МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

О.М. Пупена
О.М. Клименко

АВТОМАТИЗАЦІЯ ПОРЦІЙНИХ ВИРОБНИЦТВ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Навчальний посібник

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

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Рецензент: *Сідлецький В.М., к.т.н., доцент кафедри Автоматизації та комп'ютерних технологій систем управління НУХТ*

Відповідальний редактор: *Степанець О.В., к.т.н., доцент*

Гриф надано Методичною радою КПІ імені Ігоря Сікорського
(протокол № 1 від 02.09.2022 р.)
за поданням Вченої ради факультету
(протокол № 9 від 28.06.2022 р.)

Навчальний посібник призначений для студентів, які навчаються за освітньою програмою «Автоматизація та комп'ютерно-інтегровані технології кібер-енергетичних систем» спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології та вивчають курс «Автоматизація порційних виробництв».

Метою навчального посібника є висвітлення методики створення систем автоматизації об'єктів періодичної дії.

Реєстр. № НП 22/23-056. Обсяг 6 авт.арк.

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

ЗМІСТ

ВСТУП

Кредитний модуль «Автоматизація порційних виробництв» однойменної дисципліни надає можливість студентам ознайомитися зі створенням систем автоматизації об’єктів періодичної дії, керуванням технологічним процесом та керуванням устаткуванням.

Лабораторне заняття складається з короткого повторення теорії, проведення досліду та його аналіз. Задачі поділено за основними темами дисципліни і розглядаються у такій послідовності:

1. Підготовка робочого місця.
2. Основи роботи з Batch Control.
3. Звя’язок рецептурного та апаратного керування.
4. Виконання процедур на рівні SCADA та ПЛК. Можливості рецептів в PFC.
5. Реалізація Модулів Керування (Control Module).
6. Модель апаратурних об’єктів. Координаційне керування.
7. Керівні рецепти.

Для самостійної підготовки до лабораторного заняття необхідно вивчити матеріал за конспектом лекцій і літературними джерелами, зрозуміти методику виконання даних задач.

ЗАГАЛЬНІ ПОЛОЖЕННЯ

1 Опис завдань та проблем при керуванні порційним виробництвом

Під **порційним виробництвом** (Batch production) розуміється спосіб виготовлення продукції, за якого певна порція матеріалу (**партія**, **batch**) проходить поступову покрокову обробку відповідно до технології, що означена в рецепті. Наявність різних рецептів (регламентів) виготовлення ще не робить установку або частину її устаткування (обладнання) виробництва порційним. Ключовим моментом тут є проведення кожної технологічної дії над усією порцією (партією) матеріалу одночасно. Тому, скажімо, ємність, в якій готується суміш (змішується, проходить термічну обробку) відноситься до устаткування порційного виробництва. А пастеризаційна установка (набір теплообмінників в потоці) – ні, так як порція проходить через установку поступово, і в один момент часу обробляється тільки її частина.

Слід зазначити, що установка для виробництва одного і того самого продукту може бути як порційного так і неперервного типу. «Порційність» має певні переваги та недоліки порівняно з неперервною обробкою. Неперервний спосіб потребує обробки в потоці, тобто усі технологічні процеси проходять на різних ділянках установки. Це дає велику продуктивність, однак потребує додаткового обладнання для змішування, рециркуляцій і т.д (див.Рисунок 1). Такий спосіб виробництва передбачає постійного виготовлення продукції (великих партій) за однією технологією, тому запускається на досить тривалий період (тижні, місяці). Найбільш важкими для такого виробництва з точки зору керування є запуски, заплановані та нештатні зупинки.

Порційний спосіб передбачає проходження технологічних процесів розподілено не за потоком, а за часом. Найбільшим недоліком такого виробництва є його менша продуктивність порівняно з неперервним. Але «порційність» дає певні переваги:

1. як правило використовується менша кількість обладнання, ніж для неперервного виробництва;
2. невеликі розміри партій і відповідно час виготовлення;
3. одне і те саме обладнання можна використовувати для виробництва різних продуктів.

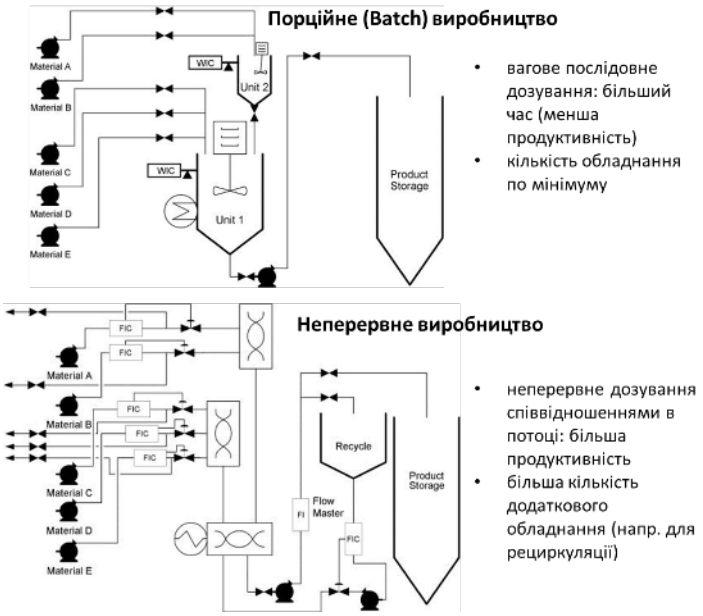


Рисунок 1 – Порівняння порційного виробництва з неперервним

Таким чином порційне виробництво дає можливість на одному і тому самому устаткованні виробляти невеликі партії продукту за різними рецептами. Це основа гнучкого виробництва, яка надає можливість швидко підлаштуватися під кон'юнктуру ринку. Не дивлячись на такі потенційні переваги самого типу виробництва, для максимального «вижимання» з нього усіх можливостей потребується просунута система керування. Деякі методи керування неперервним виробництвом тут можуть бути просто неефективними. Які ж додаткові завдання повинна реалізовувати система керування порційним виробництвом? Зупинимося на найбільш важливих.

1. Можливість виготовлення продукції на будь-якому устаткованні, що фізично на це здатне. Припустимо, якийсь апарат фізично може проводити усі необхідні технологічні дії для виготовлення продукції. Цього повинно бути достатньо, щоб використовувати його в системі керування. Однак, якщо рецепт продукту в системі не закладений, керування виготовленням можна зробити тільки вручну. Поява нового рецепту потребуватиме нового технологічного процесу, який по суті фізично може бути проведений на наявному устаткованні, але потребує підтримки зі сторони системи керування цим устаткованням. Задача спрощується, якщо в рецептах продукту змінюються тільки технологічні параметри, які можна змінювати за зміни рецепту. Але не тривіальною задачею є розробка системи керування, яка б підтримувала зміни послідовності кроків, зміни додаткових умов, або необхідність паралельного виконання технологічних дій, оскільки це потребує нового означення технологічної програми.

2. Планування та перепланування. Гнучкість порційного виробництва потребує швидкого планування. Навіть якщо на виробництві є система MES, як правило, вона забезпечує планування до рівня робочого центру, але не до технологічного устаткування всередині нього. Крім того, постійні зміни умов

проходження процесу в середині робочого центру потребують перепланування.

3. Координація ручних та автоматичних операцій. Порційне виробництво нерідко потребує використання ручних операцій, наприклад дозування або проведення замірів. Система керування повинна передбачати такі дії, зокрема формувати завдання для операторів відповідно до рецепту (наприклад, у вигляді повідомлень), отримувати підтвердження виконання та робити відмітки в журналі. Навіть якщо технологічні дії відбуваються повністю в ручному режимі, їх координація та фіксація у виробничих журналах має також проводитися системою керування.

4. Простежуваність процесів. Простежуваність виробництва включає три складові:

1) простежуваність постачальника (зовнішня простежуваність, крок назад);
2) простежуваність процесів (внутрішня простежуваність);
3) простежуваність покупця (зовнішня простежуваність, крок вперед).

Внутрішня простежуваність потребує попередньої фіксації параметрів проходження всіх технологічних процесів виробництва та задіяного при цьому устаткування з можливістю їх визначення за сировиною (пряме простежування) або за продуктом (зворотне простежування). Система керування повинна забезпечувати фіксацію виконання усіх технологічних операцій (у тому числі ручних) та виведення їх у звіти за ідентифікатором партії продукту або за ідентифікатором партії сировини.

5. Інтегрування з верхнім рівнем. На відміну від неперервних виробництв, для вищих рівнів керування (MES/MOM, або ERP) недостатньо передавати тільки агреговані та усереднені значення основних виробничих параметрів. Необхідно надавати дані про партію, зокрема про рецепт та технологічні параметри, при яких вироблялась партія. У свою чергу система керування виробничими операціями (MOM) повинна також передавати календарний план виготовлення партій, який узгоджений зі станом виробничого устаткування.

Реалізація наведених вище завдань потребує достатньо трудоємких процесів як при розробці, так і супроводженні систем керування. Вирішення їх «з нуля» навіть з використанням сучасних засобів автоматизації призводить до затяжних, а інколи невдалих проектів. З точки зору замовника, такі проекти можуть призвести до створення унікальних рішень АСКТП для кожної технологічної ділянки, що ускладнює обслуговування та інтеграцію. Замість цього, можна використовувати кращі практики затверджені в стандарті ISA-88 та його аналозі IEC-61512, які вже існують більше 20-ти років. Вони повністю означують правила побудови таких систем із забезпеченням усіх наведених вище додаткових функцій. Цей стандарт перевірений на багатьох порційних і не тільки виробництвах. Це значить, що використання його є фундаментом для розробки високоефективних систем керування порційним виробництвом. Знання його основ повинно бути обов’язковим для всіх автоматників, а також для технологів, які розробляють та супроводжують рецепти порційного виробництва.

Розділення типів керування. Основним, можна сказати навіть фундаментальним, принципом побудови систем керування порційним виробництвом згідно стандарту є розділення функцій керування на дві взаємопов'язані групи (Рисунок 2):

· керування технологічним процесом, що відповідає за виконання послідовності технологічних дій;

· керування устаткованням, що відповідає за виконання устаткованням його функцій.

Таке розділення пов'язано з тим, що номенклатура продуктів постійно змінюється, а отже виникає необхідність в створенні чи зміні перебігу технологічних процесів: додаються нові рецепти, змінюються їх властивості, видаляються старі. При цьому устаткування може залишатися тим самим і функції його також не змінюються. Може бути й інша ситуація, коли додається нове устаткування (і система керування), яка повинна вміти використовувати існуючі рецепти.

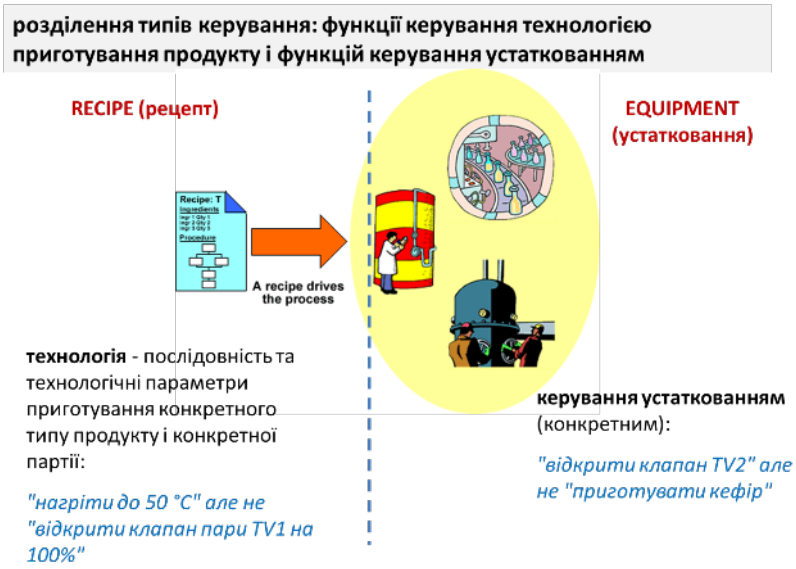


Рисунок 2 – Розділення функцій керування

Під керуванням технологічним процесом розуміється вся необхідна послідовність технологічних дій (**процедура**) з використанням заданих параметрів (**формула**), які описуються в **рецепті** для конкретного продукту. Якщо дуже спрощено, то в системі керування устаткованням реалізовані найменші **процедурні елементи** рецепту (**етапи**) типу «нагріти», «набрати», «додати», «перемішати», «охолодити». Вони будуть направлятися на виконання рецептом, який до цього створив технолог. Назва таких етапів буде співпадати для всього устаткування, але реалізація може бути зовсім різною (Рисунок 3).

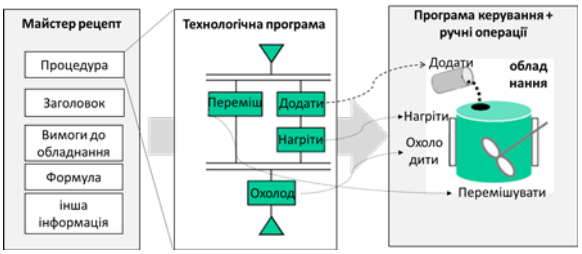


Рисунок 3 – Від рецепту до устаткування

Тобто технолог створює процедуру рецепту, набираючи технологічну програму з блоків (етапів), тестує, зберігає у бібліотеці рецептів і випускає у використання на виробництві. Така програма може мати вигляд як таблиця, або як діаграма (Рисунок 4). Послідовність роботи з рецептом можна подивитися, наприклад, за цим посиланням https://youtu.be/jCCe0jQ84fk

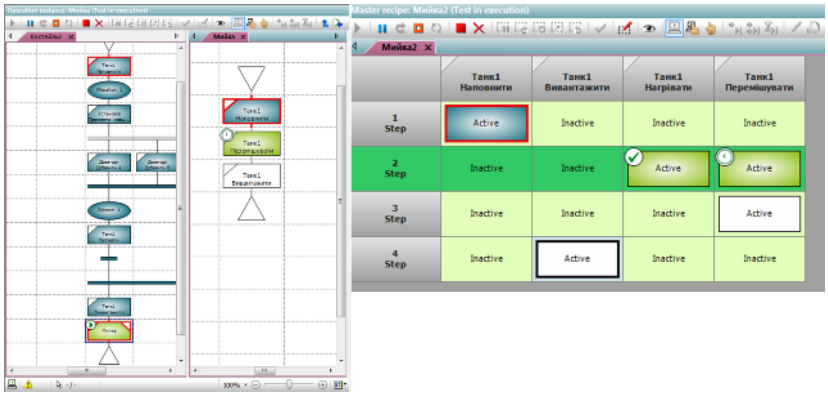


Рисунок 4 – Приклад вигляду керівного рецепту при виготовленні партії продукції

Рецепти можуть додаватися, видалятися, змінюватися, але незмінними будуть залишатися будівельні блоки, які будуть частиною системи керування устаткованням. Зрештою, якщо етапи на якомусь обладнанні мають виконуватися в ручному режимі, рецепти будуть ті самі, зміниться тільки реалізація керування устаткованням.

Керування партіями. Порційне виробництво виготовляє партію продукту за певним рецептом, який створює технолог (або хтось за його вказівкою). Цей рецепт називається **майстер рецептом**. Виготовлення конкретної партії продукту може мати певні особливості, зумовлені рядом факторів. Тому для кожної партії створюється унікальний **керівний рецепт**, який за необхідності може змінюватися. Уся інформація про історію виготовлення партії зберігається з посиланням на керівний рецепт. По суті, він відповідає за конкретну партію, тобто керує нею і залишає свій слід в архіві для простежуваності.

Керування станами та режимами. Для виготовлення конкретної партії з майстер рецепту створюється керівний рецепт, назначається йому унікальний ідентифікатор партії (Batch ID), змінюються параметри і запускається на виконання процедура рецепту. Завершення процедури керівного рецепту призводить до завершення виготовлення партії, а уся інформація залишається в історії. Така послідовність роботи процедури («початок» –> «виконується» –> «за-

вершено») буде в ідеальному випадку. А що, якщо щось піде не так при виготовленні партії? Що, якщо під час виготовлення партії:

· не вистачатиме сировини або енергії?

· потрібно призупинити роботу на певний час?

· потрібно перейти на інший крок (етап), бо фізично не може виконатися умова переходу?

· потрібно зупинити виготовлення партії?

· потрібно терміново відмінити виконання усієї процедури або одного етапу?

Стандартом передбачено, що для кожної такої ситуації у системі керування будуть означені певні дії, а сама ситуація називається **станом** (state). Побудова керування технологічним процесом на основі станів дає можливість чітко узгодити технологу та автоматнику, як себе буде вести система у кожній ситуації. Стандарт пропонує один із варіантів набору станів та умов переходу між ними (Рисунок 5). Перехід від стану до стану відбувається за командою, або за внутрішньою умовою завершення (для перехідних станів). Оскільки процедура рецепту складається з етапів, кожен з них має свій набір станів.

Наприклад, коли запускається процедура рецепту, вона переходить зі стану «ПРОСТІЙ» в стан «ВИКОНУЄТЬСЯ», і перший етап в процедурі також переходить в стан «ВИКОНУЄТЬСЯ». Якщо необхідно призупинити технологічний процес за командою оператора, або за відсутністю сировини чи енергії процедура рецепту переходить в стан «ПРИЗУПИНЯЄТЬСЯ», що призведе до переходу в такий же стан усіх активних етапів.

Зміна станів може відображатися на засобах людино-машинного інтерфейсу, наприклад зміною кольорів та написів, як це показано на Рисунку 4.

Для можливості опису різної поведінки станів в залежності від ситуації, стандарт означує поняття **режимів**. Так наприклад, оператору переводити процедуру на будь який крок можна тільки в ручному режимі, а в напівавтоматичному – після завершення одного кроку наступний запуститься тільки після підтвердження оператором.

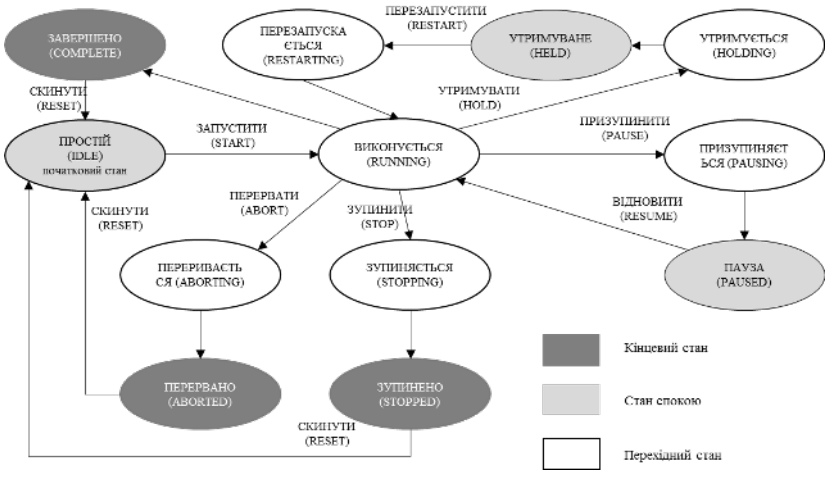


Рисунок 5 – Варіант автомату станів процедурного елементу, означеного в стандарті

Ієрархія устаткування. Для того, щоб правильно розподілити яке устаткування буде виконувати конкретні процедурні елементи (етапи) стандарт означує ієрархію устаткування (Рисунок 6). У стандарті виділяються такі рівні устаткування:

· рівень технологічної комірки (Process Cell), який відповідає за виготовлення усієї партії, тобто за виконання процедури керівного рецепту;

· рівень технологічних вузлів (Unit), в яких проходять основні технологічні операції з партією;

· рівень модулів устаткування (Equipment Module), який відповідає за додаткові технологічні операції;

· рівень модулів керування (Control Module), які невидимі для процедури, але реалізовують усі керівні дії на устаткованні;

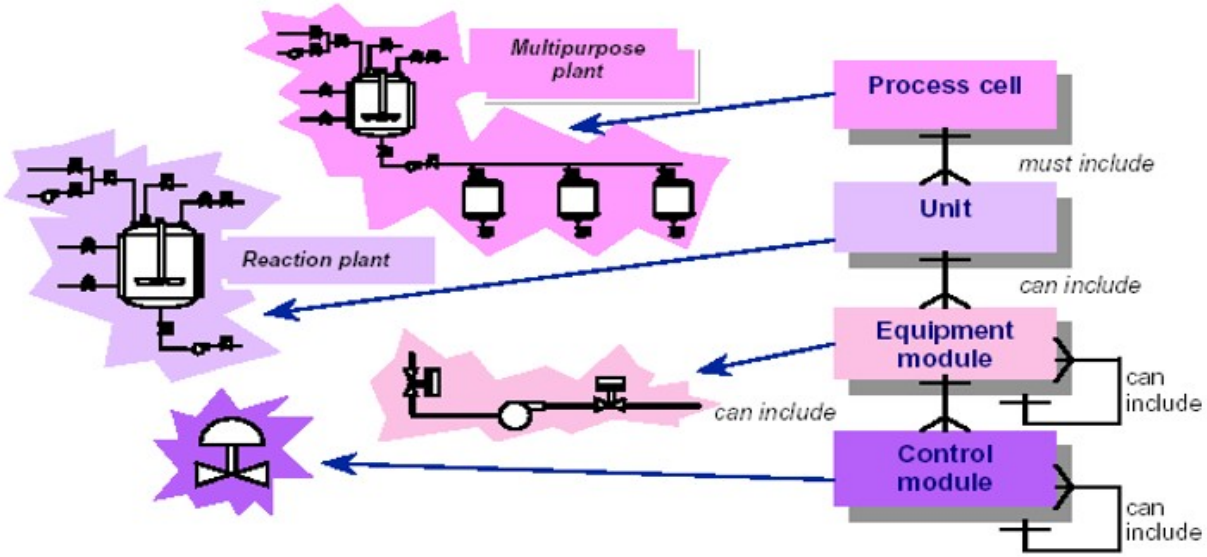


Рисунок 6 – Приклад ієрархії устаткування

Поєднання керування устаткованням з процедурою керівного рецепту забезпечує виготовлення конкретної партії (Рисунок 7). Кожен об'єкт устаткування має свій набір станів та режимів.

У стандарті також означені правила надання процедурі конкретного устаткування. Тобто етап в майстер рецепті може передбачати виконання на різному устаткуванні. Перед задіянням одного з них в керівному рецепті, устаткування надається (allocation) даному рецептурному елементу, що гарантує відсутність колізій його використання при одночасному виконанні кількох рецептів.

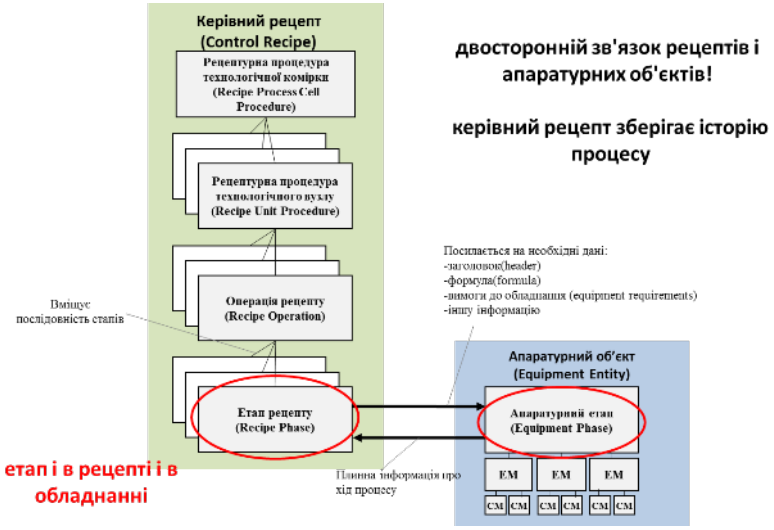


Рисунок 7 – Зв'язок етапу в рецепті з етапом в устаткованні

Ієрархія устаткування в IEC-61512 співпадає з IEC-62264 (стандарт про інтегрування MOM та ERP), що спрощує інтегрування системи з рівнем керування усім виробництвом та підприємством.

Простежуваність процесу. Стандарт передбачає, що усі дії з рецептами, переходи між станами, зміна параметрів та значення технологічних змінних фіксуються в **історії виготовлення партії** (Batch History). Таким чином, для вибраного ідентифікатора партії можна простежити за яким майстер рецептом вона була зроблена, які параметри формули змінювалися, яке устаткування було задіяне при виготовленні, коли і як змінювався стан кожного процедурного елементу. Простежуваність процесу «генетично закладено» через автомати станів, режими, послідовностне керування та керування рецептами. Навіть якщо у зміні процедури немає потреби, реалізація системи керування згідно стандарту значно спрощує механізм простежуваності.

Крім детального опису наведених вище основних концепцій, в першій частині стандарту також описані:

· різні види керування, та їх особливості
· структура рецептів
· типи рецептів на різних рівнях керування підприємством та їх зв'язок
· ієрархія процедур
· календарне планування порційного виробництва
· склад виробничої операції
· правила надання та арбітражу
· діяльності керування порційним виробництвом
· звітність про виготовлення партії
· координація роботи підсистем

Для ознайомлення з усіма функціями Ви можете прочитати навчальні матеріали за посиланням.

2 Використання підходів ієрархії устаткування при розробці програмного забезпечення

Тільки деякі з об’єктів автоматизації відносяться до порційного (Batch) типу. Тим не менше, як показує практика, багато підходів, викладених в стандарті IEC 61512 варто використовувати при впровадженні систем автоматизації незалежно від типу виробництва. Перш за все це відноситься до погляду на автоматизований комплекс з точки зору проектувальника та програміста АСКТП. Класичним є представлення автоматизованого комплексу у вигляді набору устаткування (залізо) та засобів автоматизації (набір датчиків, виконавчих механізмів, функцій/задач). Традиційний підхід до автоматизації йде через взаємопов’язані в контури функції:

«датчики -> входи регулятора (контролера) -> функції -> виходи регулятора (контролера) -> виконавчі механізми»

У противагу класичному «контурному» підходу, багато програмістів АСКТП вже давно застосовують об’єктно-орієнтовані підходи, у якому взаємопов’язані функції стають частиною якогось об’єкту, що описується та реалізовується як єдине ціле. Стандарт IEC 61512 опирається на об’єктне представлення доступного устаткування, а також технологічного процесу. Це стає зручним механізмом як для учасників життєвого циклу АСКТП, так і з позицій спеціалістів, що проводять інтегрування з верхніми рівнями керування. У цій частині посібника розглянемо використання моделей устаткування, без прив’язки до інших моделей, означених в стандарті. Коротко про призначення та зміст стандарту Ви можете прочитати в білій книзі, тут зупинимося більше на практичних аспектах.

Устаткування (Equipment)

Відповідно до стандарту, при проектуванні та розробці програмного забезпечення кожен об'єкт автоматизації розглядається як окрема сутність. У свою чергу усі об’єкти автоматизації на рівні АСКТП розділяються окремо на «технології» (як виробити продукт) і «устаткування» (на чому виробити продукт). Автоматизація «побудови технології» стосується тільки виробництв зі змінною рецептурою (для цього власне і створений стандарт), практичні рекомендації щодо неї наведені в наступних розділах. На відміну від технологічної частини, автоматизація устаткування (equipment) стосується усіх типів виробництв, навіть якщо використовуються виключно неперервні процеси з однаковою технологією.

Навіщо виділяти устаткування як окремі сутності, а не використовувати традиційні підходи на базі функціональних контурів? Перш за все слід розуміти, що у результаті виділення устаткування як окремих об’єктів, функції та навіть контури нікуди не діваються, вони стають частиною цього устаткування. По суті устаткування агрегують функції та їх зв’язки у більш загальні сутності, які сприймаються як єдине ціле. Розглянемо для прикладу, як представлений в системі автоматизації 2-х позиційний клапан або заслінка:

- орган керування (безпосередньо клапан або заслінка);

- виконавчий механізм з одним керуючим пневматичним сигналом «ВІДКРИТИ»;

- два датчика кінцевого положення «ВІДКРИТИЙ», «ЗАКРИТИЙ».

На схемі автоматизації для кожної з частин буде по одному зображенню (див. Рисунок 8), який як правило відповідає одному засобу автоматизації. Є функції для клапану, які часто не показуються на схемі автоматизації, але повинні бути передбачені в алгоритмі, наприклад:

- базові функції контролю та керування;

- функції взаємодії з HMI, такі як переключення режимів руч/авт, та ручне керування;

- функції тривожної сигналізації.

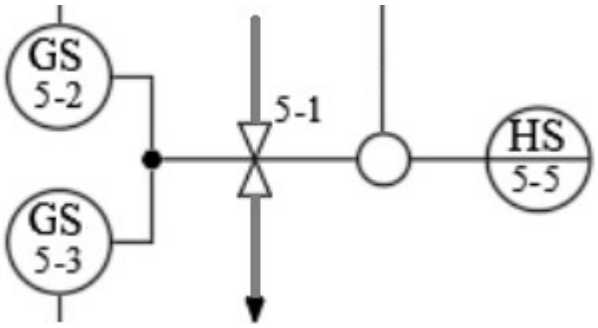


Рисунок 8 – Зображення клапана на схемі автоматизації

У програмах контролера та SCADA/HMI будуть відповідні змінні/теги та/або функції. Але з точки зору експлуатації клапан сприймається не як сукупність функцій, а в поняттях його станів, наприклад: функціонального («відкритий», «закритий»), режиму («у ручному режимі», «в автоматичному»), тривоги («не відкрився»). Ці поняття характерні для клапана в цілому, а не для його частин чи функцій. Тому на дисплеях HMI засоби автоматизації можуть показуватися як згруповані разом елементи, анімація передбачатиме використання всіх тегів, які мають відношення для клапану.

Аналогічним чином відносяться до нього інженери, що не мають відношення до автоматизації. Вони оперують поняттями станів, а не функцій чи засобів вимірювання та керування. Тому при проектуванні чи експлуатації їх не цікавить спрацювання датчиків кінцевого положення, та зрештою їх може взагалі і не бути, або бути присутнім тільки один з них.

Описане вище відношення до клапану виробничників є очевидним. Тим не менше, класичний контурний підхід при розробці ПЗ для АСКТП йде врозріз цьому підходу. Усі ці функції представлені переліком змінних (тегів), які аналізуються/змінюються в окремих частинах програми. Так, наприклад функція керування клапаном відбувається в контурі алгоритму керування процесом, а функція сигналізації – в контурах сигналізації і блокування. Таке розкидування по коду може стати дуже громіздким і робить код погано читабельним. Наведемо кілька завдань, які передбачають крос-функціональну взаємодію:

- блокування клапану при відмові одного з датчиків;

- блокування клапану, якщо він не відкрився;

- тимчасова можливість функціонування без одного з датчиків.

Для наведеного вище прикладу використання в програмному забезпеченні об’єктів типу «клапан» дає можливість інкапсулювати усю логіку виконання функцій, що стосуються його, в одну сутність. У цьому випадку взаємодія інших, зовнішніх по відношенню до нього об’єктів, буде проходити через взаємодію з ним, а не оперувати набором тегів чи функцій, які мають відношення до клапану.

Для багатьох інженерів АСКТП використання об’єктного підходу стало типовою практикою. Тим не менше, її застосовують не всі. Крім того, часто зустрічається частково-об’єктний підхід, що пов’язано з погано проробленою методологією створення ПЗ. Стандарт IEC 61512 дає певні правила виділення та оперування такими об’єктами-устаткованнями, хоч не обмежує в цьому, зокрема:

- устаткування (equipment) існує як виділена в системі керування сутність зі своїми наборами атрибутів;
- устаткування має певну роль, від якої залежить, які типи функцій воно виконує;
- устаткування формує ієрархію, розміщення в якій також впливає на керування.

Отже, окрім набору функцій та відповідних їм змінних в програмі контролера чи SCADA/HMI, передбачається наявність окремих сутностей (об’єктів) - **устаткування** (**equipment**), яке включає в себе інші об’єкти (дрібніші частини устаткування) та функції. Надалі під словом «устаткування» будуть розумітися спеціалізовані об’єкти в системі керування, які є двійниками їх фізичних сутностей. Функції устаткування надалі будемо називати **функціональними елементами**.

Взаємодія з устаткованням проводиться через його змінні стану та команди.

Стани (states)

Стан (**state**) – це загальна властивість, яка вказує на плинне становище якогось об’єкта. Оскільки устаткування включає в себе певні функціональні елементи, то зрештою його стан залежить від станів цих елементів та попереднього стану устаткування. Відповідно до принципів емерджентності, стан системи не є простою сукупністю станів її елементів. Тим не менше, наразі для простоти, будемо вважати, що це так.

Отже, стан устаткування може оцінюватися з позицій його функціональних елементів, наприклад з точки зору виконуваної операції, наявності тривог, обслуговування (ремонту) чи джерела керування. Тому під «узагальненою сукупністю» розуміється поєднання усіх станів функціональних елементів як єдиного цілого. У залежності від стану устаткування можуть змінюватися як сигнали керування, так і виконувані алгоритми.

Для прикладу розглянемо наведений вище клапан. Його стан можна розглядати з точок зору станів його функціональних елементів:

- операційна функція (функція керування/контролю позиції клапану): ВІДКРИТИЙ, ЗАКРИТИЙ, ВІДКРИВАЄТЬСЯ, ЗАКРИВАЄТЬСЯ, НЕ ВИЗНА-

ЧЕНИЙ (наприклад одночасне спрацювання обох кінцевиків) та ін.;

- функція тривожної сигналізації: НЕМАЄ ТРИВОГ, НЕ ВІДКРИВСЯ, НЕ ЗАКРИВСЯ, ДОВІЛЬНИЙ ЗСУВ та ін; у свою чергу кожна з цих тривог також описується певним станом: НЕ АКТИВНА, АКТИВНА НЕ ПІДТВЕРДЖЕНА, АКТИВНА ПІДТВЕРДЖЕНА та ін.; тому загальний стан наявності тривог клапана є певною згорткою;

- режим функціонування: АВТОМАТИЧНИЙ (алгоритми системи керування), РУЧНИЙ (дії оператора), МІСЦЕВИЙ (керування за допомогою перемикачів по місцю), БЛОКОВАНИЙ (функції керування не активні);

- функція імітації роботи (наприклад, з метою налагодження): РЕАЛЬНИЙ (зчитує значення з входів та записує виходи) та ІМІТОВАНИЙ (значення датчиків генеруються алгоритмом імітації, значення виходів не записуються);

- функція обслуговування: НА РЕМОНТІ, ФУНКЦІОНУЄ; останній час обслуговування; кількість переключень.

Перелік функціональних елементів та їх станів варіюється від вимог до контролю та керування устаткованням і не обмежуються стандартом. Оскільки устаткування можуть бути складеними, то станів може бути набагато більше, оскільки воно включає в себе кілька об’єктів. Наприклад, для устаткування типу «насос з перетворювачем частоти» стан представляється узагальнюючим набором станів функціональних елементів 2-х об’єктів: двигуна та перетворювача частоти. Крім того, з точки зору функції операційної діяльності розглядаються не лише дискретні набори станів (ВКЛЮЧЕНИЙ, ВІДКЛЮЧЕНИЙ), а й додаються аналогові значення плинної частоти (швидкості), струму, напруги тощо. У той же час на основі аналогових значень можуть формуватися набори дискретних станів функцій, такі як «працює на мінімальній частоті» або «на максимальній».

Таким чином, контроль за устаткованням відбувається через відповідні **змінні стану**, які повинні бути в програмі контролеру, SCADA/HMI чи іншого інтелектуального засобу. Для дискретних станів функцій це бітові статуси, які приймають значення TRUE/FALSE, або їх комбінація.

бітові статуси = дискретні стани або їх комбінація

Сукупність цих станів функціональних об’єктів є об’єднання (конкатенація) цих статусів. У цьому випадку всі стани функціональних об’єктів можна об’єднати в певний упорядкований набір бітів для всього устаткування – **статусне слово** (**status word**).

статусне слово = набір бітових статусів функцій елементів устаткування

Використовуючи статусне слово інша частина системи може аналізувати устаткування як єдине ціле через бітове представлення. Це дає змогу контролювати як конкретний статус функціонального елементу, звертаючись до нього як до біту, так і як бітову матрицю, використовуючи маску.

Наприклад, для клапану статусне слово може мати вигляд як в таблиці 1. Біти стану можуть взаємно виключати один одного, наприклад «ВІДКРИТИЙ» і «ЗАКРИТИЙ», а їх рівність 0 – представляти інший стан. Наприклад, якщо біти 5–8 дорівнюють нулю – це вказує на стан «НЕ ВИЗНАЧЕНИЙ».

Таблиця 1 – Приклад статусного слова для 2-х позиційного клапану

Біт	Опис
1	2
0 ALMOPN	=1 тривога НЕ ВІДКРИВСЯ
1 ALMCLS	=1 тривога НЕ ЗАКРИВСЯ
2 BLCK	=1 БЛОКОВАНИЙ
3 ALMSHFT	=1 тривога ДОВІЛЬНИЙ ЗСУВ
4 ALMSNSR	=1 тривога ПОМИЛКА ДАТЧИКА
5 OPNING	=1 ВІДКРИВАЄТЬСЯ
6 CLSING	=1 ЗАКРИВАЄТЬСЯ
7 OPNED	=1 ВІДКРИТИЙ
8 CLSED	=1 ЗАКРИТИЙ
9 DISP	=1 РУЧНИЙ режим (з ПК/ОП), =0 АВТОМАТИЧНИЙ режим
10 MANBX	=1 МІСЦЕВИЙ режим
11 ALM	=1 загальна тривога
13 FRC	=1 хоча би одна зі змінних в об’єкті форсована
14 SML	=1 режим імітації

Автомати станів (State Machines)

З точки зору іншої частини системи, устаткування змінює свій стан. Тому при написанні програми для об’єкту-устаткування необхідно реалізувати зміну його стану в залежності від станів функціональних елементів та інших об’єктів, що включені в нього. Ці стани необхідно змінювати в залежності від умов. Таку поведінку станів можна описати словесним алгоритмом, на кшталт:

якщо клапан в стані «ЗАКРИТИЙ» і прийшла команда «ВІДКРИТИ», перейти в стан «ВІДКРИВАЄТЬСЯ»

Для функції тривожної сигналізації це може виглядати так:

якщо клапан в стані «ВІДКРИВАЄТЬСЯ» і не спрацював датчик кінцевого положення і час відкриття більше максимального, то перейти в стан «НЕ ВІДКРИВСЯ»

Слід зауважити, що у даному прикладі в алгоритмі керування станами тривожної сигналізації використовуються стани операційної функції. Тобто стани різних функціональних елементів устаткування взаємопов'язані. Це одна з причин, чому функції варто групувати в устаткування.

Алгоритм, що описує для конкретної функції поведінку переходу між станами називають **автоматом станів** (**state machine**). Більш зручним описом автомату станів є графічний – **діаграма станів**. Вершинами її є стани, а ребрами переходи між станами та відповідні умови.

Автомати станів не є новаторством стандарту IEC 61512. Це класичний механізм формалізації та моделювання, який використовується у багатьох галузях, у тому числі в автоматизації. Наприклад, на Рисунку 9 показана діаграма класичного автомату станів для тривог, що описаний в стандарті IEC 62682 з певними спрощеннями. Стани тривоги представлені на рисунку колами з підписами, в яких наводиться опис, що включає комбінацію статусів: статусу тривоги та статусу підтвердження. У даному випадку, стан тривоги – це узагальнюючий показник, який залежить від плинного значення статусів та від попереднього стану. Стрілки на Рисунку 10 відповідають переходам між станами з вказаними умовами цих переходів.

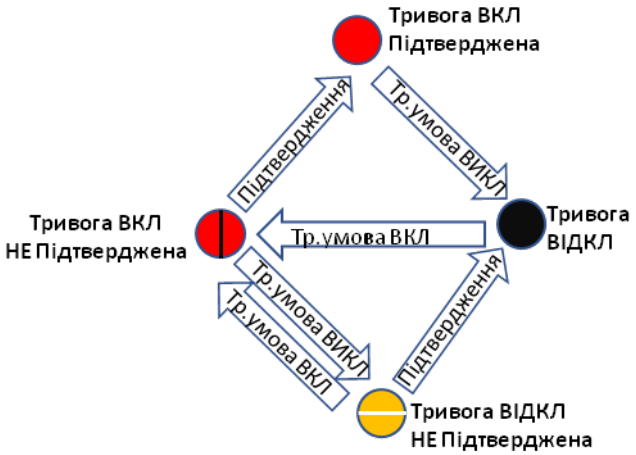


Рисунок 9 – Спрощений автомат станів тривог

Не дивлячись на те, що станів тривоги всього чотири, діаграма виглядає досить просто. Але стандарт IEC 62682 передбачає ще три можливих стани блокування, в які можна перейти з будь-якого іншого стану. Діаграма показана на Рисунку 10, але усі переходи показати там не вдасться.

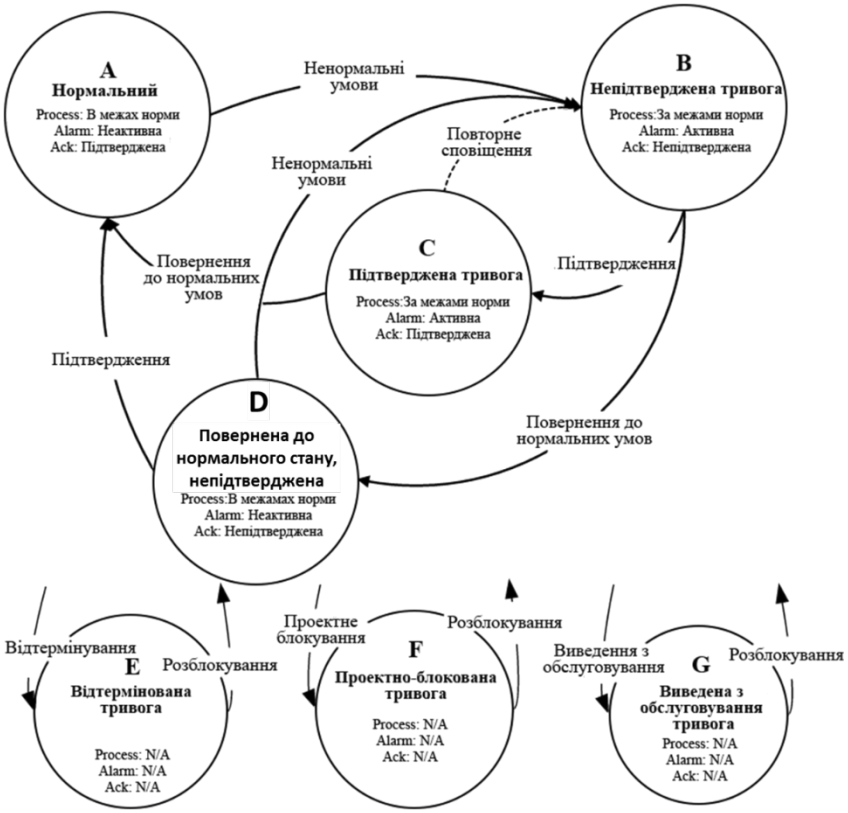


Рисунок 10 – Повний автомат станів тривог згідно IEC 62682

Розглянемо діаграму станів для операційної функції клапану. У найпростішому випадку клапан має два стани – «ВІДКРИТИЙ» та «ЗАКРИТИЙ». У залежності від наявності датчиків кінцевого положення, клапан може описуватися автоматами станів, що показані на Рисунку 11. На перший погляд, кожен з цих варіантів є самодостатнім. Однак кожен з них має ряд вад.

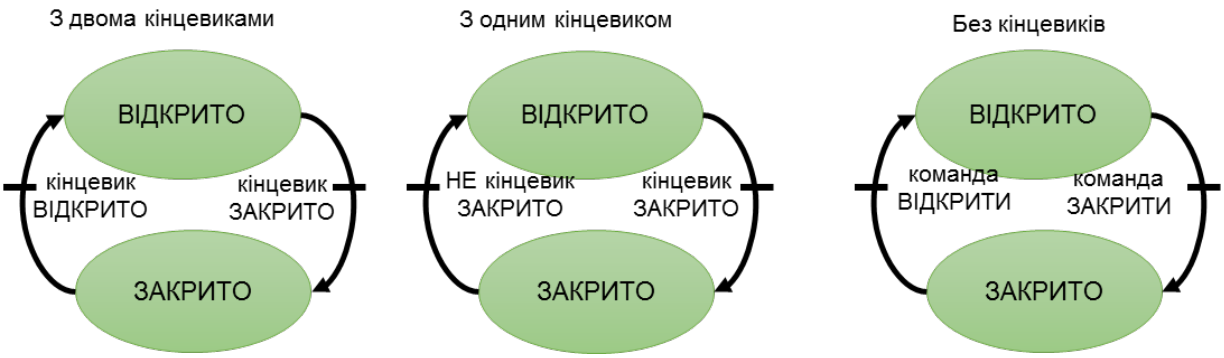


Рисунок 11 – Приклади найпростіших варіантів автомату станів для клапану

У варіанті з двома кінцевиками при виникненні несправності одного з них, жоден з них не буде дорівнювати одиниці, або спрацюють обидва. Це автоматом не передбачено. Аналогічна ситуація може відбуватися і з другим варіантом, коли датчик кінцевого положення буде несправним. Третій варіант взагалі немає контролю стану, тому в цьому випадку для алгоритмів керування мо-

жливо треба врахувати час переміщення, для уникнення, наприклад, гідроударів. Будь-який з цих варіантів не передбачає формування тривог на базі станів, так як автомату станів тривог ні на що орієнтуватися. Очевидно в наведених вище станах з датчиками кінцевого положення необхідно передбачити в якості умов переходу команд керування. Під останніми розуміється команди на об'єкт-устаткування, а не на його апаратну частину (тобто виконавчий механізм). Також, для спрощення побудови автомату станів тривожної сигналізації варто ввести додаткові перехідні стани «ВІДКРИВАЄТЬСЯ» та «ЗАКРИВАЄТЬСЯ». Крім того, варто ввести стан «НЕ ВИЗНАЧЕНО», якщо конкретну позицію неможливо ідентифікувати. З цього стану можна стартувати автомат при ініціалізації програми керування, або переходити туди при поломці датчиків положення (обидва в одиниці). У такому випадку діаграма станів операційного функціонального елементу матиме вигляд як на Рисунку 12.

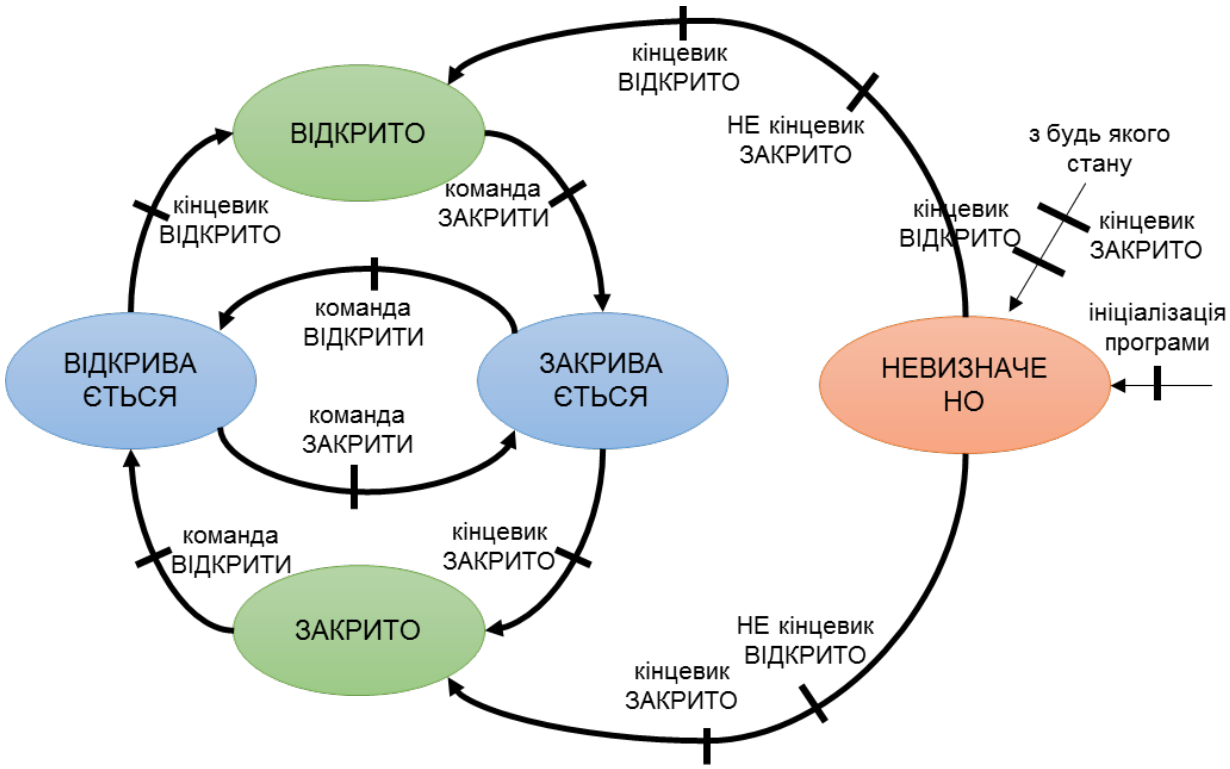


Рисунок 12 – Приклад розширеного автомату станів для операційного функціонального елементу клапану

У кожному з наведених станів можна робити певні керуючі дії. Наприклад в стані «ВІДКРИВАЄТЬСЯ» вмикати дискретний вихід контролера, що керує клапаном. Крім того, можна вмикати таймер, який буде вказувати на час активності стану, що можна буде використати для керування тривогами.

Використовуючи діаграму операційного функціонального елементу клапану можна описувати алгоритм керування, який в свою чергу може спиратися на інші автомати станів. Інші функціональні елементи цього ж клапану можуть використовувати цей автомат для формування логіки своїх автоматів. Для прикладу розглянемо автомати станів для одного з функціональних елементів тривожної сигналізації «НЕ ЗАКРИВСЯ». Для спрощення ми будемо розглядати тільки статус активності тривоги, тобто без урахування статусу пі-

дтвердження і блокування (Рисунок 13). Тривога виникає тоді, коли клапан знаходиться в операційному стані «ЗАКРИВАЄТЬСЯ» і час цього стану більше максимально дозволеного.

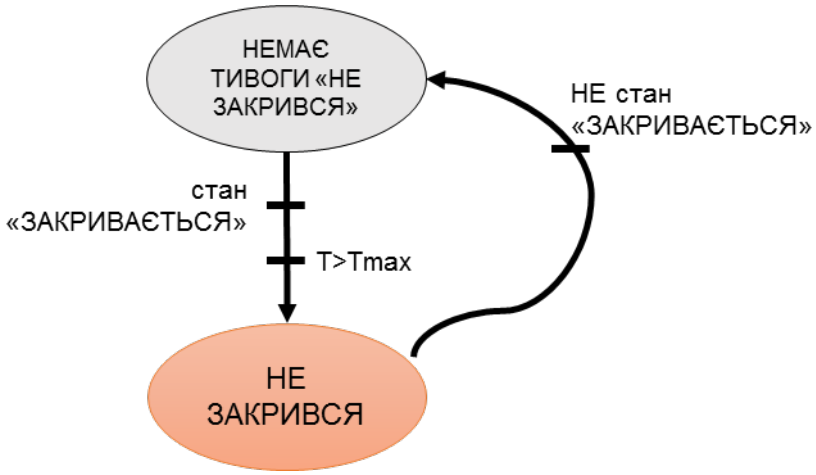


Рисунок 13 – Спрощений вигляд автомату станів для тривоги «НЕ ЗАКРИВСЯ»

Як видно, автомати станів функціональних елементів тривожної сигналізації тісно пов'язані з операційними. Тому в ряді випадків автомати станів різних функціональних елементів показують зв'язаними на одній діаграмі. Однак слід розуміти, що якщо два автомати станів з Рисунку 12 та з Рисунку 11 поєднати в одну діаграму, то в один момент часу будуть активні два стани, наприклад «ЗАКРИВАЄТЬСЯ» та «НЕ ЗАКРИВСЯ», що з графічного зображення може бути неочевидним. Тим не менше, в ряді випадків кілька автоматів станів можна звести в один, як це показано нижче на прикладі перетворювача частоти. У будь-якому випадку програмна реалізація може базуватися на станах операційного функціонування, в яких будуть реалізовані керування станами інших функціональних елементів.

Для прикладу з клапаном, устаткування можна описувати кількома взаємозалежними автоматами станів:

- операційний;
- 4 автомати для тривог («НЕ ВІДКРИВСЯ», «НЕ ЗАКРИВСЯ», «ДОВІЛЬНИЙ ЗСУВ», «ПОМИЛКА ДАТЧИКА»);
- блокування;
- режимів роботи;
- імітування.

Режими (Modes)

Набір функціональних елементів і їх станів не означується стандартом. Тим не менше, подібні до описаних вище операційних функцій відносяться до так званого базового керування (basic control), опис якого дається в наступних розділах. Крім того, у стандарті окремо виділяються функції керування **режимами** (**mode**) устаткування. Режим вказує на те, у який спосіб відбувається керування операційними функціями. Зрештою, «режими» це окремо виділені

стани, які впливають на особливість виконання (алгоритмів) функцій устаткування, а інколи – і на їх автомати станів.

Для устаткування стандарт рекомендує використовувати два режими: РУЧНИЙ та АВТОМАТИЧНИЙ. У РУЧНОМУ режимі, операційний стан устаткування означується командами з HMI, у АВТОМАТИЧНОМУ – з алгоритму керування. На практиці режимів може бути більше. Наприклад, для згаданого вище клапану, на Рисунку 14 показана діаграма з додатковими режимами «РУЧНИЙ ПО МІСЦЮ» і «ЗАБЛОКОВАНИЙ». У «РУЧНОМУ ПО МІСЦЮ» режимі, клапан керується байпасним щитком, що знаходиться біля клапану. У «ЗАБЛОКОВАНОМУ» режимі на клапан завжди подається команда «ЗАКРИТИ».

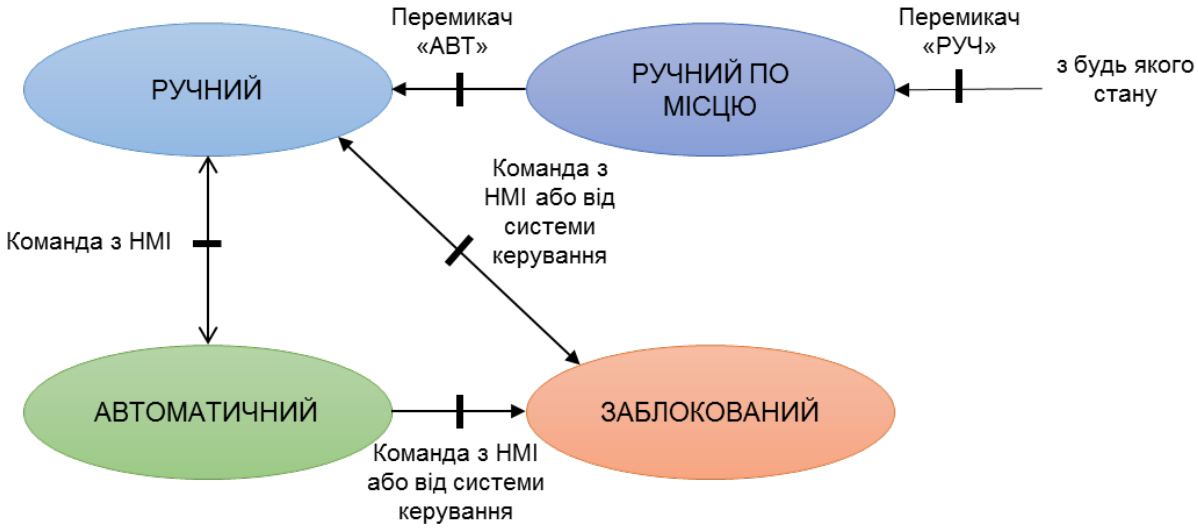


Рисунок 14 – Автомат станів переключення режимів устаткування типу клапана

У даному випадку на автомат станів, що наведений на рис. 12, будуть подаватися команди керування з різних джерел. Але, у ряді випадків автомати станів деяких функцій можуть змінюватися в залежності від режиму устаткування. Наприклад, діаграма на рис. 14 не передбачає контроль команди ВІДКРИТИ та ЗАКРИТИ у режимі «РУЧНИЙ ПО МІСЦЮ», так як ці команди не можуть бути простежені системою. Тому, для цього режиму варто продумати інший автомат.

Вище наведені приклади станів, які є взаємно-виключеними. Наприклад, стани ВІДКРИВАЄТЬСЯ і ЗАКРИВАЄТЬСЯ з Рисунку 12 ніколи не можуть бути активними одночасно. Для прикладу з 4-ма режимами, все не так однозначно, оскільки наприклад режим РУЧНИЙ (з HMI) і РУЧНИЙ ПО МІСЦЮ (з байпасу) можуть виникнути одночасно. Необхідно чітко розставити пріоритети при керуванню станами в програмі. У цьому випадку, «РУЧНИЙ ПО МІСЦЮ» має пріоритет, оскільки команди з контролеру, нівелюються.

Умови переходів та команди

Як зазначалось вище, автомати станів описуються станами та переходами, для яких задаються умови. Умовами переходу можуть бути як команди алгоритму керування (або HMI), так і вияв системою керування зміни стану устаткування, як правило за допомогою датчиків. Наприклад, на Рис. 5 перехід з ВІДКРИТО в ЗАКРИВАЄТЬСЯ відбувається по команді ЗАКРИТИ з системи керування, а з ЗАКРИВАЄТЬСЯ у ЗАКРИТО по стану датчика кінцевого положення. Слід звернути увагу, що датчик кінцевого положення, який є елементом клапану, теж є устаткованням, яке має свої стани. Він може мати стан ВІДМОВА, який може впливати на стани (або навіть режими) устаткування вищого рівня, тобто клапану.

З точки зору устаткування, будь-яка дія, націлена на нього або контроль його внутрішнього стану (команда керування, відслідковування стану), є умовою переходу. Керовані умови переходу формуються **командами**. Можна виділити кілька джерел команд: з алгоритму керування, засобів HMI, системи верхнього рівня і т.п. У деяких випадках вони можуть відпрацьовуватися за різними алгоритмами, тоді діаграма станів має це відображати. Команди можуть бути реалізовані як бітові (ВІДКРИТИ, ЗАКРИТИ) або у вигляді числового **слова команди** (**command word**), де конкретна команда задається числом. Враховуючи, що за один раз устаткованню передається одна команда, як правило достатньо одного слова (числової змінної) для передачі всіх можливих команд керування. Обробник команд може не реагувати на ті команди, які недоступні в даному стані або режимі.

Використання спільних ресурсів

З точки зору наведених вище критеріїв деяке устаткування може в окремий момент часу підпорядковується різним об’єктам верхнього рівня. Прикладом такого об’єкту може бути спільна зважувальна ємність, яка використовується декількома установками. Якщо більше ніж один об’єкт вищого рівня може оволодіти або запитувати послуги певного об’єкта, то він позначається як ***спільний ресурс***. Спільні ресурси часто присутні в складних процесах порційного виробництва.

Спільний ресурс може бути ресурсом ексклюзивного користування або колективного користуванням. Якщо ресурс позначений як ресурс ***ексклюзивного користування***, то в один момент часу тільки одне устаткування вищого рівня може використовувати цей ресурс. Прикладом ресурсу з ексклюзивним користуванням може бути спільна для установки зважувальна ємність (дозатор). Вона може бути використана тільки одним реактором в один момент часу.

Якщо спільний ресурс позначений як ресурс ***колективного користування***, кілька об’єктів можуть використовувати ресурс одночасно. Прикладами ресурсів колективного користування на заводі з порційним виробництвом можуть бути: підігрівачі, які використовуються декількома технологічними вузлами одночасно; системи розподілу сировини, які здатні забезпечити матеріалом більше ніж один технологічний вузол одночасно.

Для ресурсів ексклюзивного використання треба контролювати доступ, щоб кілька об’єктів вищого рівня одночасно не змогли ним скористатися. У стандарті ця задача має типове вирішення через ***механізм надання*** (***allocation***) спільного ресурсу. Спосіб його реалізації надання не вказується в стандарті. Зрештою, керування та контроль зайнятості може вирішуватися зміною та перевіркою відповідної властивості устаткування, яке є спільним ресурсом. У ряді наших проектів властивість використання вказувала на ідентифікатор того устаткування, якому було надано даний ресурс. При нульовому ідентифікаторі вважалося, що устаткування не задіяне і може бути за необхідності надано об’єкту вищого рівня.

Описаний вище спосіб надання був використаний на ряді об’єктів, де ми розробляли прикладне ПЗ для АСКТП. Зокрема на станції водопідготовки необхідно було, щоб один і той самий клапан мав керуватися двома незалежними об’єктами вищого рівня: CIP-мийкою та фільтром. Те устаткування, яке раніше отримувало доступ до ресурсу, перше резервувало цей клапан для своєї роботи, інше в цей час не мало до нього доступу для керування.

У стандарті IEC 61512 механізми вирішення пріоритетності надання описуються як задачі надання та арбітражу.

Процедури

Вище описані принципи використання устаткування нижнього рівня як програмних об’єктів в проектах АСКТП незалежно від типу виробництва. У стандарті IEC 61512 до його функцій ставляться певні вимоги, які забезпечують можливість виготовлення продукту порційним способом з використанням різних рецептів. Перш за все, устаткування в стандарті розглядається як незалежний від виготовлення конкретного продукту або напівпродукту набір сутностей. Іншими словами, робочий центр, який в IEC 61512 носить назву ***технологічної комірки*** (див.Рисунок 6), може робити певні технологічні операції, але не зав’язане на конкретній технології (конкретному рецепті). У свою чергу ці операції реалізовуються через певні технологічні дії, які може виконувати устаткування нижніх рівнів (технологічні вузли та модулі). Це значить, що технологічна комірка повинна спрямовувати діяльність устаткування нижніх рівнів для виготовлення конкретної партії напівпродукту відповідно до вказаного рецепту.

Таким чином, головним принципом побудови систем керування порційним виробництвом згідно стандарту IEC 61512 є початкове розділення функцій керування на дві взаємопов’язані групи:

- керування технологічним процесом, що спрямовує виконання технологічних дій у заданій послідовності;
- керування устаткованням, що спрямовує його на забезпечення виконання конкретної технологічної дії.

Враховуючи порційний характер виробництва, технологічні дії виконуються в певній послідовності. Стандарт враховує, що кожна технологічна дія може виконуватися на одному з кількох доступних одиниць устаткувань.

Таке розділення пов’язано з тим, що номенклатура продуктів постійно змінюється, тому виникає необхідність в створенні чи зміні перебігу технологічних процесів: додаються нові рецепти, змінюються їх властивості, видаляються старі. При цьому устаткування може залишатися тим самим, і його здатності виконувати технологічні дії також не змінюються. З іншого боку, при встановленні в систему нового устаткування (і системи керування), воно повинно вміти використовувати існуючі рецепти.

Щоб пов’язати технологію з устаткованням, використовуються елементи технологічних дій – ***процедури***. Технологія порційного виробництва продукту або напівпродукту передбачає виконання «технологічної програми», яка складається з кроків, кожен з яких передбачає виконання певної процедури, наприклад: 1) наповнили; 2) охолодили до 15 °С; 3) вкинули фермент в пропорції 1:100 … Ця технологія означується в ***рецепті***, який вміщує також параметри (формулу) та іншу інформацію. По суті, «технологічна програма» в рецепті є також процедурою. Тобто ***рецептурна процедура*** складається з менших процедур, а ті, в свою чергу, посилаються на конкретні екземпляри в устаткованні, яке може їх виконувати. Так відбувається зв'язок рецепту з устаткованням. Найменші процедури називаються ***етапами***.

Устаткування (об’єкти) усіх рівнів керування за виключенням СМ може містити в собі процедури. Модулі керування виконують тільки базові функції (***базове керування***), яке було описано вище, і не можуть бути вказані в рецепті, оскільки стосуються виключно конкретної одиниці устаткування. Процедури не можуть вказувати на конкретні датчики чи клапани, інакше вони б були прив’язані до конкретного устаткування. Базове керування – навпаки, відноситься виключно до конкретного устаткування. Тому процедури рецепту можуть виконуватися на різних одиницях устаткованнях, які у свою чергу, використовуючи базове керування, можуть реалізовувати необхідні дії.

Автомат станів та режими процедур

У порційному виробництві технологічна програма та конкретний її крок повинен починатися, виконуватися і завершуватися відповідно до означених правил. Тому згідно стандарту кожна процедура розглядається як виділена сутність, яка також описується автоматом станів. Наявність автомату станів для процедур робить її опис більш формалізованим та зрозумілим. Дискретність станів полегшує координацію між процедурами, взаємодію з оператором та простежуваність. Побудова керування технологічним процесом на основі станів дає можливість чітко узгодити технологу та автоматнику, як себе буде вести система у кожній ситуації.

Стандарт пропонує один із варіантів набору станів та умов переходу між ними (Рисунок 15). Перехід від стану до стану відбувається за командою, або за внутрішньою умовою завершення (для перехідних станів). Як рецептурна процедура, так і процедури, з якої вона складається, описуються автоматом станів. Наприклад, коли запускається рецептурна процедура (оператор натискає «ЗАПУСТИТИ»), вона переходить зі стану «ПРОСТІЙ» в стан «ВИКО-

НУЄТЬСЯ», і перша процедура в ній (етап) також переходить в стан «ВИКОНУЄТЬСЯ». Якщо необхідно призупинити технологічний процес за командою оператора, або за відсутністю сировини чи енергії процедура рецепту переходить в стан «ПРИЗУПИНЯЄТЬСЯ», що призведе до переходу в такий же стан усіх активних етапів.

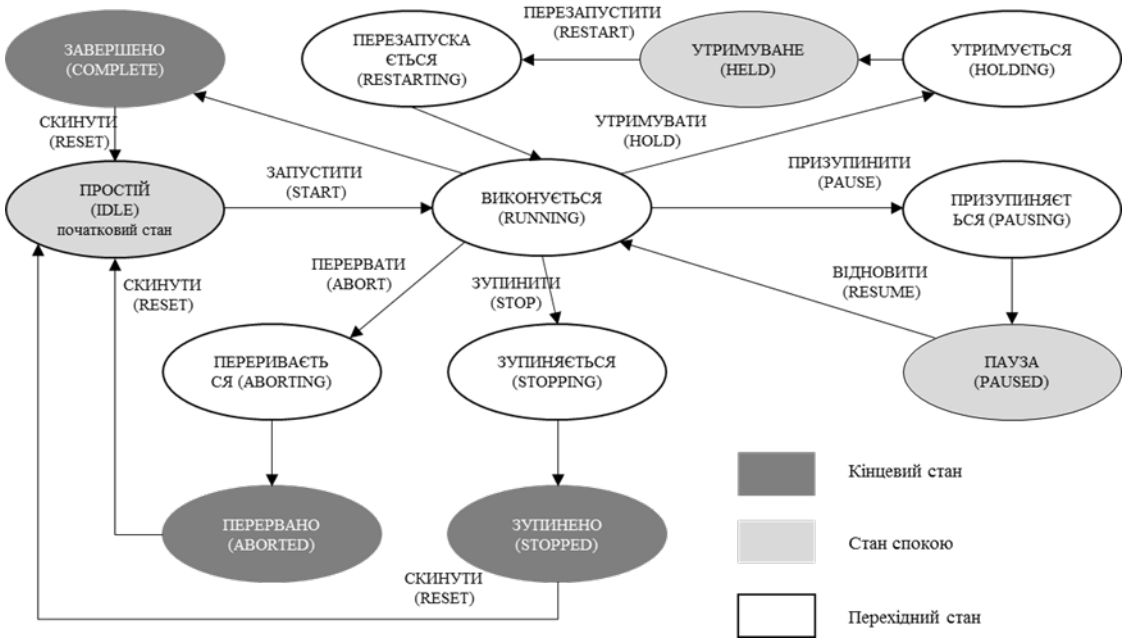


Рисунок 15 – Варіант автомату станів процедурного елементу, означеного в стандарті IEC 61512

Окрім станів для процедур в стандарті означені режими: автоматичний, ручний, напівавтоматичний. Напівавтоматичний режим задає можливість запуску наступного кроку за підтвердженням оператору. Ручний режим дає можливість оператору терміново завершити будь-який етап, самому вибрати необхідний етап для виконання.

Ієрархія процедур

Як зазначалося вище, процедури мають ієрархічну структуру. Ієрархія процедур показує яким чином менші процедури реалізують виконання процедур вищого рівня (Рисунок 16). Процедура, що означує приготування партії продукту є ***процедурою технологічної комірки*** (у стандарті її називають просто «процедурою»). Найменші процедури, які не можуть включати в себе інші, називаються етапами. Таким чином, процедури вищого рівня включають процедури нижчого, тобто координують їх роботу. Принципи взаємодії в цій ієрархії дещо схожі на ієрархію устаткування, яка розглянута вище.

Процедура технологічної комірки виконується в межах однойменного устаткування, яке є робочим центром порційного типу (див. Рисунок 11). В межах технологічної комірки можна виготовляти одну або кілька партій (Batch) одночасно. Виготовлення партії у порційному виробництві потребує технологічних вузлів (апаратів), в яких послідовно відбуваються виконання крупних тех-

нологічних дій – ***процедур апаратів***. Ті у свою чергу можуть складатися з операцій, а ті – з етапів. Наявність процедур технологічної комірки і етапів в ієрархії є обов'язковою умовою, інші – за необхідності.

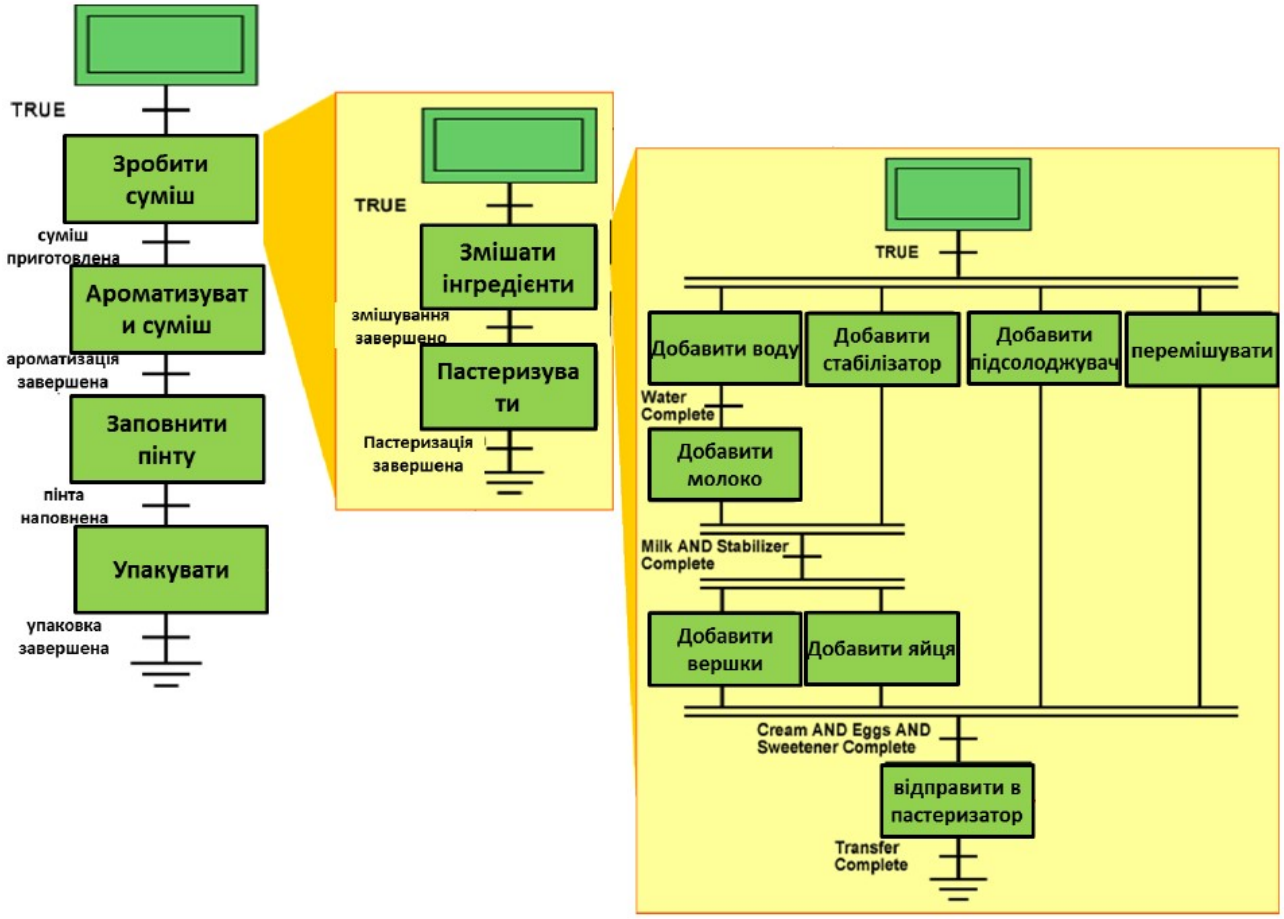


Рисунок 16 – Ієрархічне представлення процедурних елементів

По суті, процедури в програмах керування будуть функціями або функціональними блоками, які будуть реалізовувати певну технологічну операцію. Найменші процедури, тобто етапи, будуть в свою чергу керувати модулями керування, які в свою чергу реалізують базове керування. Аналогічно базовому керуванню, усі функції (або функціональні блоки), які реалізовують процедури, повинні оброблятись паралельно, тобто без взаємного вкладення. Ієрархія передбачає лише координацію між ними.

Для ієрархії процедур може використовуватись принцип поширення станів та режимів. Наприклад при переході процедури верхнього рівня в стан утримування, процедури нижнього рівня також можуть перейти в цей стан.

3.Підходи до реалізації функцій рецептурного керування

Для забезпечення процедурного керування, процедури мають бути частиною устаткування. Як зазначено вище, устаткування, яке може виконувати процедуру приготування всієї партії продукту, є технологічною коміркою. А сама процедура є процедурою технологічної комірки. Приготування певного продукту означується технологічною програмою, яка є набором взаємопов'я-

заних процедур. У найпростішому випадку ці процедури виконуються одна за одною, в складнішому – можуть мати місце альтернативні відгалуження, паралельне виконання і т.д.

Майстер рецепти та керівні рецепти

У будь-якому випадку технологія приготування продукту означується ***майстер-рецептом***. Майстер-рецепт по своїй суті є шаблоном, який включає в себе загальну процедуру приготування продукту (послідовність виконання процедурних елементів), формулу (набір параметрів) та множину устаткування, що може бути використане при виробництві даного типу продукту. Майстер-рецепт створює технолог для конкретного типу продукту, що може виготовлятися в конкретній технологічній комірці.

Майстер-рецепт не може бути безпосередньо використаний при виробництві партії продукту і на це є ряд причин. Для кожної партії продукту індивідуально необхідно:

- вказувати кількість продукту, що необхідно зробити (величина партії) згідно цього рецепту;
- вказувати індивідуальні параметри рецепту саме для цієї партії;
- конкретизувати устаткування, що буде використовуватися при виготовленні саме цієї партії;
- координувати запуск необхідних процедур у вибраному устаткованні;
- змінювати технологічну послідовність за необхідності;
- вести архівування по технологічним процесам виготовлення партії, з можливістю подальшого простежування.

Тому для виробництва конкретної партії продукту використовується ***керівний рецепт***. Відповідно до стандарту він супроводжує всю партію від початку до закінчення виготовлення. Керівний рецепт створюється як копія майстер рецепту і отримує унікальний ідентифікатор, у якому вказується шлях проходження продукту і є «маркером» для ведення архіву виготовлення партії.

Принципи реалізації рецептів з однією процедурою

У найпростішому випадку майстер-рецепт і відповідно керівний рецепт може вміщувати посилання на процедуру технологічної комірки конкретного устаткування (Рисунок 17). Таким чином, «технологічна програма» буде повністю означена в устаткованні, що характерно для деяких порційних виробництв, які не потребують змінної рецептури. Звісно, що це значно зменшує гнучкість використання устаткування, але не суперечить стандарту.

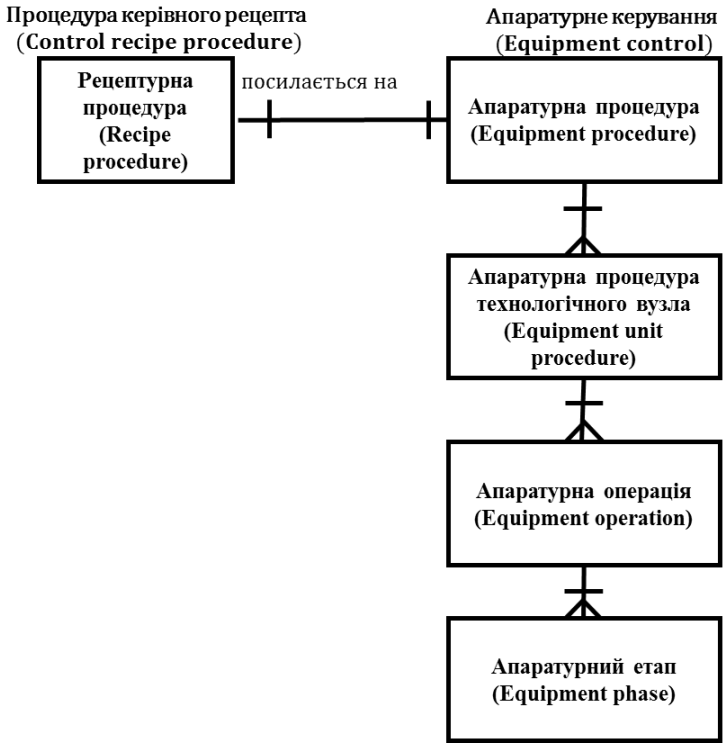


Рисунок 17 – Приклад процедури керівного рецепту з посиланням на реалізовану процедуру в устаткованні

Для таких випадків майстер-рецепт, як і керівний рецепт можна реалізувати через структурні змінні, які будуть включати певний мінімальний набір полів:

- BATCHID – унікальний ідентифікатор партії продукту, використовується для простежування виготовлення партії. Застосовні лише для керівного рецепту. Зазвичай генерується при створенні та запуску керівного рецепту на виконання з часу та дати.
- MSRECIPE_ID – унікальний ідентифікатор майстер-рецепту, використовується для посилання на шаблон як при виготовленні партії так і при простежуванні.
- Змінна стану та змінна часу стану – використовується для керування виконанням процедури технологічної комірки керівного рецепту (запуск, зупин, утримування), і зазвичай відповідає. Застосовні лише для керівного рецепту.
- Параметри формули – вміщує необхідний набір полів, які відповідають за параметри виготовлення продуктів. Міститься як в майстер-рецепті так і в керівному. При запуску керівного рецепту копіюються з відповідного майстер-рецепту. Одним із параметрів може бути номер необхідної процедури технологічної комірки.
- Статистична інформація (опційно) – вміщує набір інформації, на основі якої будуть генеруватися звіти, наприклад, кількість виготовленої продукції, інформацію про перебіг технологічного процесу і т.д. Генерується керівним рецептом в процесі його виконання.

– Вимоги до необхідного устаткування – технологічне устаткування, на якому можна виконати конкретний майстер-рецепт.

Таким чином, процедура технологічної комірки буде реалізована в програмі контролера, а рецепт матиме ідентифікатор, який посилається на дану процедуру. При необхідності приготування конкретного продукту потрібно вибрати майстер-рецепт, який посилатиметься на необхідну процедуру.

У складнішому випадку, майстер-рецепт означує «технологічну програму» через послідовність процедур. Формат опису такої процедури може бути різним, в тому числі і з використанням мови PFC. Зрештою, він може вміщувати усі типи процедурних елементів (Рисунок 18). Повні вимоги до структури керівного та майстер-рецепта і мова PFC описані в стандарті IEC 61512-2 і виходить за рамки даного посібника. Приклад такої реалізації з використанням спеціалізованого модуля показаний в лабораторному практикумі та продемонстровано на відео.

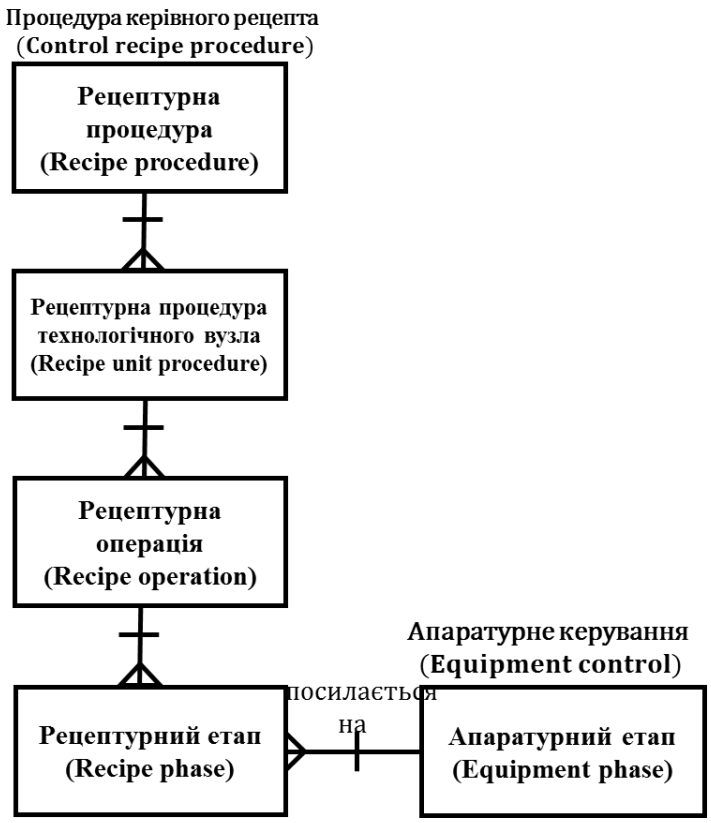


Рисунок 18 – Приклад процедури керівного рецепту з повною ієрархією рецептурних процедур

4 Мови та способи побудови рецептів

S88.01 означує термін "процедура" (procedure) - як "Стратегія для виконання процесу." Термін "процедура" використовується протягом усього документу першої частини стандарту, але правила для опису (формалізації) процедури не означені. Хоч відсутність такої формалізації залишило велику "прогалину" в 1-й частині, це було зроблено навмисно, по причині різних поглядів членів комітету SP88. Комітет визнав, що це є дуже важливим питанням, і члени до-

мовилися про два кроки для його вирішення. По-перше, Комітет дійшов висновку, що після завершення 1-ї частини стандарту буде генеруватися технічний звіт для обговорення "можливих форматів для рецептурних процедур". По-друге, було вирішено, що друга частина стандарту означуватиме спосіб зображення рецепта. У 1997 році був випущений технічний звіт ISA-TR88.0.03-1996 з назвою "Possible Recipe Procedure Presentation Formats" ("Можливі формати представлення рецептурних процедур"). Після випуску технічного звіту була продовжена робота над частиною 2 стандарту і в даний час міститься в розділі 6 проекту S88.02 під назвою "Настанова для мов".

Розділ 6 проекту стандарту S88.02 означує стандартний метод зображення процедурної логіки майстер рецепту і керівного рецепту. Цей метод був названий Procedural Function Chart (PFC) (процедурна функціональна діаграма). Нотація PFC була створена для задоволення вимог, узгоджених в одному з засідань комітету. Нотація PFC була розроблена з використанням елементів трьох різних форматів, обговорюваних в технічному звіті - це список, діаграма Ганта і мова Sequential Function Chart (SFC). На перший погляд, нотація PFC може виявитися схожою на SFC. Однак там є тонкі але істотні відмінності. Ці відмінності потрібні для задоволення вимог рецептурного процедурного керування, на відміну від послідовного виконання керуючих дій і документування, для яких використовується SFC.

Стандарт S88.01 стверджує, що рецепт містить п'ять категорій інформації, а саме: заголовку(header), вимог до обладнання (equipment requirements), формули (formula), процедури (procedure) і "як-небудь означену" іншу інформацію (" other information). У першу чергу, метод, обраний для представлення рецепта в S88.02 має справу тільки з категорією "процедура", так як це той "клей", що поєднує інші категорії, дає рецепту "глибинність" (тобто представлення багаторівневої ієрархії рецептурних процедурних елементів) а також повинен містити логіку. Решта чотири категорії лише коротко згадуються в проекті, і не стандартизуються, окрім вимог, що їх зв’язки з кожним елементом або символом в процедурі "повинні бути чітко вказані, і послідовні в межах кожного застосунку".

Нотація PFC призначена як стандартний засіб зображення процедурної логіки (procedural logic) в рецептах. Слід визнати, що поряд з PFC будуть використовуватися і альтернативні методи, так як в залежності від характеристик процедури (наприклад, розміру, складності і потреб користувача) вони можуть бути більш придатними. Перевага стандартного методу в тому, що він дозволить обмінюватися даними рецептів, зменшити криву навчання для користувачів різних систем і створити загальну основу для спілкування між користувачами і постачальниками.

Вимоги до зображення процедурної логіки

При розробці методу представлення рецепта першочерговою задачею було узгодити набір вимог, яким він повинен відповідати. Отриманий список був включений в розділ 6.11 ISA-dS88.02-1999 проекту 13, відповідно до яких, ме-

тод представлення повинен:

- бути простим для використання: легким для розуміння людиною
- бути легким для побудови: містити небагато синтаксичних вимог і символів
- бути з чітко означеними межами рецепту:повинні бути стандартизовані графічні символи початку і кінця
- мати однозначне зображення порядку виконання: послідовність, паралельність (одночасність), альтернативний вибір (дивергенція) і сходження
- виражати координаційні відношення: транспортування речовин, очікування, синхронізацію і т.д.
- забезпечувати уніфікацію на різних рівнях: стандартизовані символи для процедури технологічної комірки, процедури апарату, операції, етапу
- забезпечувати існування рівнів: уніфікований графічний символ для відображення можливості розкладання елемента в ієрархії
- бути застосовним як для майстер рецептів так і керівних рецептів
- бути застосовним для всіх рівнів: схожий набір символів і правил для всіх рівнів в рецепті
- бути незалежним від середовища розробки: зручним і зрозумілим як при реалізації олівцем на папері так і з повно-кольоровою анімацією з використанням комп'ютерної графіки.

Деякі з вимог легко оцінити (тобто стандартизовані символи для процедур технологічної комірки, процедури апарату, операції і етапу), тоді як інші носять суб'єктивний характер (наприклад легкі для розуміння людиною).

Виходячи з вимог однозначного опису порядку виконання є необхідність в масштабованості. У цьому контексті масштабованість значить діапазон складності процедур, який вони можуть мати. У найпростіших випадках процедура може мати один рецептурний процедурний елемент (Recipe Procedural Element, RPE), або простий набір таких елементів, які будуть виконуватися послідовно, один за одним, по завершенню попереднього. У складних випадках може знадобитися умовна логіка і часові обмеження.

Окрім наведених вище вимог, комітетом визнана необхідність стандарту нотації процедур для обміну даними, з використанням табличної структури з розділу 5 ISA-dS88.02-1999 проекту 13. Табличні структури (з розділу 5) а також модель даних (з розділу 4) підтримують нотацію PFC в узагальненій формі. Якщо немає якоїсь іншої причини, для надання можливості обміну рецептами потрібен стандартний метод зображення процедури.

Порівняння існуючих та запропонованих методів

Після узгодження вимог до методу зображення, комітет дослідив існуючі методи, обміркував технічний звіт і розглянув інші стандарти. У ході цього процесу було визнано, що технічний звіт представив точний аналіз варіантів, в тому числі більшість методів, що використовувалися в промисловості і комерційно доступних системах керування періодичним виробництвом (batch control systems).

Метод списку, показаний в двох різних форматах відображення на Рисунку 19 та 20, має переваги в легкості візуалізації і "очевидної точності". Слід зазначити, що історично так склалося, що більшість рецептурних процедур описані саме в текстовому форматі або списку. Ці рецепти часто виглядають як пронумерований список, який інформує оператора про те, на якому кроці виконання він знаходиться. У цьому форматі оператор зазвичай виконує до кінця один пункт, а потім переходить до наступного. Однак список корисний тільки в простих випадках. При необхідності означення умовної логіки, часових налаштувань, координації дій та візуалізації текстовий вигляд швидко стає заплутаним і схильним до помилкової інтерпретації.

ID	Phase	Phase Parameters		
		Ingredient	Set Point	Other Parameter
1	Fill	Water	1000 kg	
2	Add Manually	Salt	50 kg	
3	Heat	Steam	50 °C	
4	Add Manually	Sugar	30 kg	

Рисунок 19 – Вигляд процедури в табличному вигляді

1. Fill with 1000 kg Water
2. Manually add 50 kg Salt
3. Heat with steam to 50 °C
4. Manually add 30 kg sugar

Рисунок 20 – Вигляд процедури в форматі текстового списку

Для зображення проходження діяльності в часі корисними є Діаграми Ганта. Вони також можуть бути використані для відображення діяльностей на кількох рівнях ієрархії. Така форма зображення добре ілюструє рецептурну процедуру технологічної комірки (process cell procedure), яка складається з одного або декількох процедур апаратів (unit procedures), які по суті працюють незалежно одна від одної, координуючись між собою в певних точках. На Рисунку 21 показаний приклад з чотирма процедурами апаратів е формі діаграми Ганта.

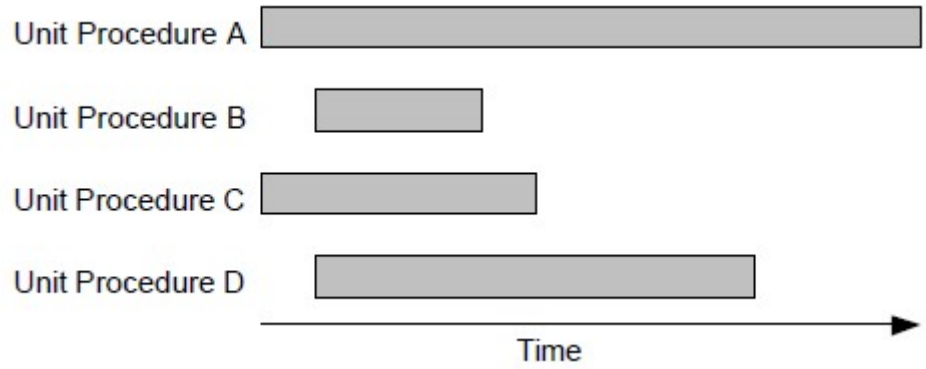


Рисунок 21 – Рецептурна процедура технологічної комірки у форматі діаграми Ганта

Така форма може бути розширеною для зображення координації між процедурами апарату, а також операцій всередині процедур апарату, як показано на Рисунку 22.

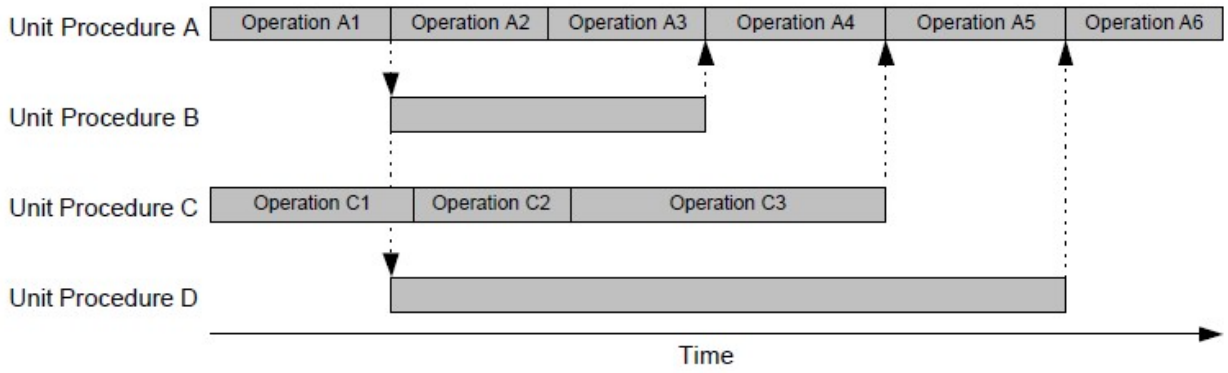


Рисунок 22 – Діаграма Ганта з зображенням Координації процедур апаратів і декількома рівнями (2-ма) процедурних елементів

Однак, коли додається складна умовна логіка, то для зображення процедури діаграма Ганта стає дуже громіздким інструментом. Щоб показати такий тип інформації у системах планування, наприклад, використання PERT-діаграми. PERT діаграми є дуже потужним інструментом і вимагають складних пакетів для планування, однак вони не так легко читаються або розуміються користувачами системи планування.

Таким чином, хоч діаграми Ганта забезпечують потужний метод зображення процедур апаратів в часі, чого, наприклад, немає в методі зображення списком, вони не забезпечують адекватної специфікації і зображення умовної логіки.

Третій метод, що обговорювався в технічному звіті – це використання Sequential Function Charts (SFCs), який означений в МЕК 61131-3. Діаграми SFC отримали широке визнання в переробній промисловості і забезпечують потужний засіб для опису умовної логіки, чого не вистачає в методах списку і

діаграми Ганта. Було відзначено, що багато хто з недавно розроблених систем керування періодичним виробництвом використовують SFC для зображення рецептурних процедур. Приклад процедури операції в форматі SFC показаний на Рисунку 23.

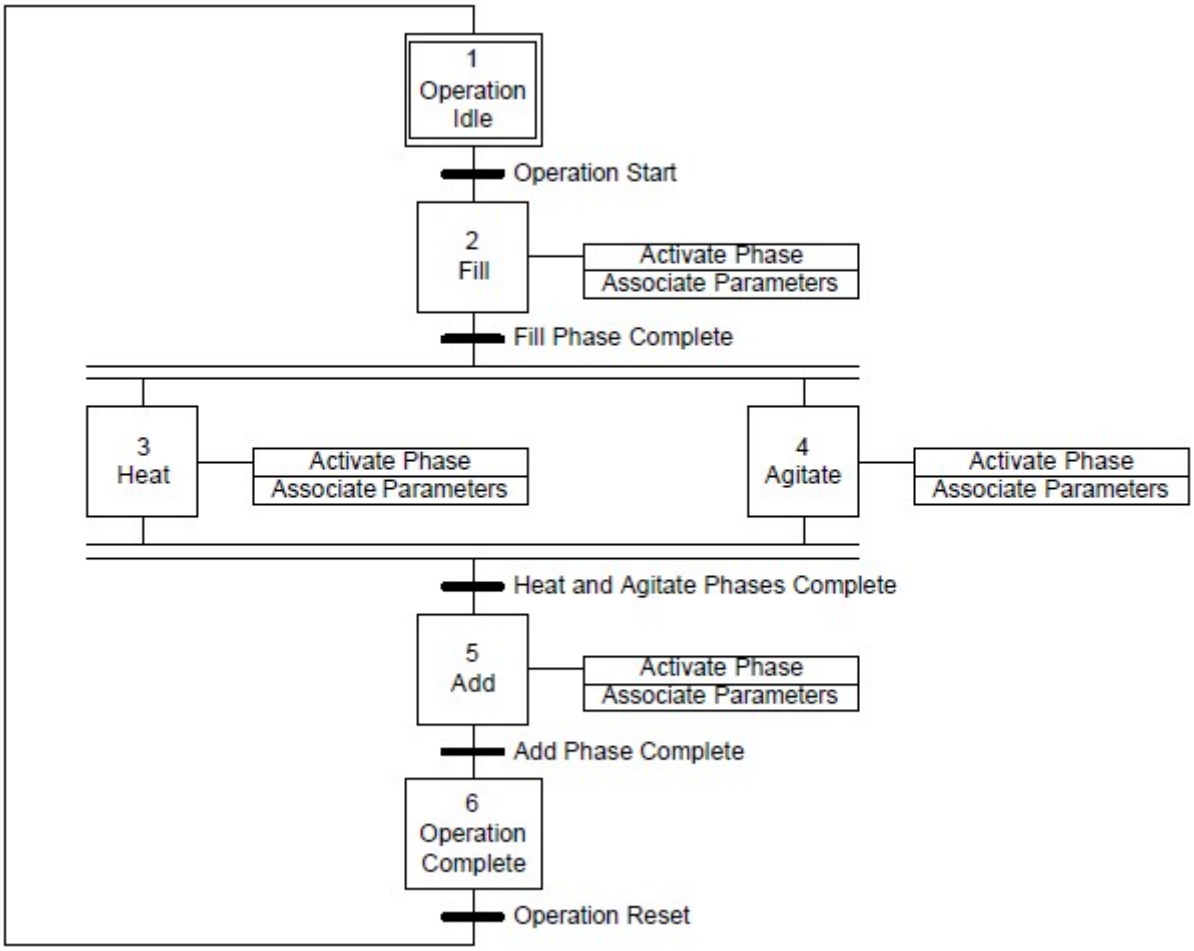


Рисунок 23 – Процедура операції (Operation Procedure) в форматі SFC

У цьому прикладі кроки були використані для забезпечення керування станами простою і завершення процедури операції. Ці кроки процедури операції не мають дій. Інші кроки процедури використані для активації рецептурних етапів (recipe phases). Кожний рецептурний етап в цьому прикладі відповідає апаратурному етапу (equipment phase).

Може бути багато стилів SFCs. У даному прикладі від кроку Завершення (Operation Complete) до кроку Простою (Operation Idle) показано повернення по циклу. Однак в деяких реалізаціях зворотній зв'язок може бути опущений разом з переходом Скидання (Operation Reset), тоді крок "Operation Complete" може бути використаний в якості кінцевої точки для SFC.

Рисунок 24 містить в форматі SFC рецептурну процедуру технологічної комірки. Вона також має петлю по циклу, хоча керівні рецепти/партії (control recipes/batches), як правило, не повторюються. Для того, щоб зупинити цей перехід, після кроку "Batch Complete" стоїть умова "FALSE", тим самим запобігаючи повторення процедури.

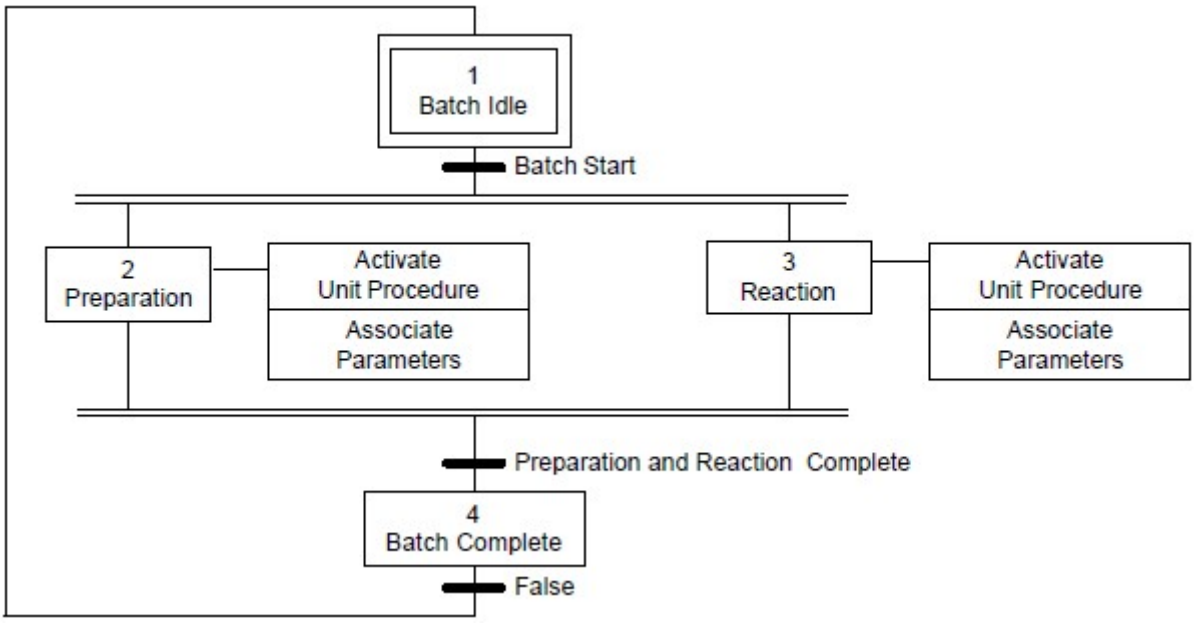


Рисунок 24 – Рецепт в SFC форматі

SFCs забезпечує придатний інструмент для зображення умовної логіки, що часто потрібно на рівні процедур операцій. У противагу цьому, використовуючи SFC на рівні рецептурних процедур технологічної комірки приводить до діаграм, що не дають багато корисної інформації. На Рисунку 24 показані 2 процедури апарату, що виконуються одночасно. Тим не менше, немає ніякої інформації щодо потоку матеріалу, синхронізації між процедурами апаратів і загального часу проведення процедур апарату по відношенню один до одного. Зображення SFC на Рисунку 24 це не показує через необхідність поставити всі процедури апарату в межах структури одночасного виконання (між горизонтальними подвійними лініями). Також Рисунку 24 не вказує час координації між процедурами апаратів. Наприклад, процедура апарату "Reaction" не повинна запускатися, поки процедура апарату "Preparation" не буде наполовину готова. Ця інформація не відображається з використанням формату SFC.

У технічному звіті зроблено висновок, що метод SFC, підтримуючи при цьому складну умовну логіку представлення, "не дозволяє користувачеві легко візуалізувати відношення між процедурами апаратів".

Два питання, які не розглядаються в технічному звіті, але надзвичайно важливі для зображення рецепта є використання кількох рівнів процедур в ієрархії і розділення виконання процедури в рецепті від виконання апаратурного процедурного елемента (equipment procedural element).

В ході цієї оцінки, були також розглянуті інші роботи. Карл-Ерік Арзен і Шарлота Джонсон опублікували роботи щодо діаграм High-Level Grafchart і як вони можуть бути використані для зображення рецепта. Крім того, вже протягом деякого часу йде робота по перегляду МЕК 60848-1988, щоб визначити, яким чином можна використовувати багаторівневі функціональні діаграми. Для розв'язання цих питань роботи передбачають означення таких структур як тупикові кроки (pit steps), тупикові переходи (pit transitions) і макро-кроки

(macro steps). Ці роботи відносяться до вирішення питання багаторівневих діаграм, які були недостатньо оброблені в попередніх означеннях Function Charts (Функціональних діаграм) і Sequential Function Charts (SFC). З цієї причини кінцева нотація PFC має схожість з High-Level Grafcharts, а в IEC 3B/WG14 запропоновано переглянути IEC 60848: 1988 (*Нова версія стандарту IEC 60848 вийшла в 2013, драфт версія стандарту доступна за* посиланням*. Крім того, на сьогоднішній день SFC підтримує макрокроки. Примітка перекладача*).

Нотація PFC

У той час, коли писалась ця стаття, нотація PFC була означена в ISA-dS88.02-1999 Draft 13. Цілком можливо, що перед випуском в якості затвердженого стандарту нотація буде змінена, оскільки буде проходити її обговорення і голосування в рамках процесів розробки стандартів ANSI/ISA та IEC. Тому пояснення нотації PFC в майбутньому слід перевіряти з остаточним стандартом, коли він буде випущений для використання. (*На даний момент в 2016 році, стандарт доступний в редакції ANSI/ISA–88.00.02–2001 Batch Control Part 2: Data Structures and Guidelines for Languages**. Нотація PFC описана в 6-му розділі стандарту, і при поверховому огляді відповідає наведеній в даній статті. Примітка перекладача*).

Дана робота не має за мету повністю пояснити нотацію PFC, це може бути зроблено в наступних публікаціях. Тут наведений короткий огляд означень та деякі приклади.

Рисунок 25 показує символи, що використовуються в нотації PFC. При створенні PFC ці символи комбінуються.

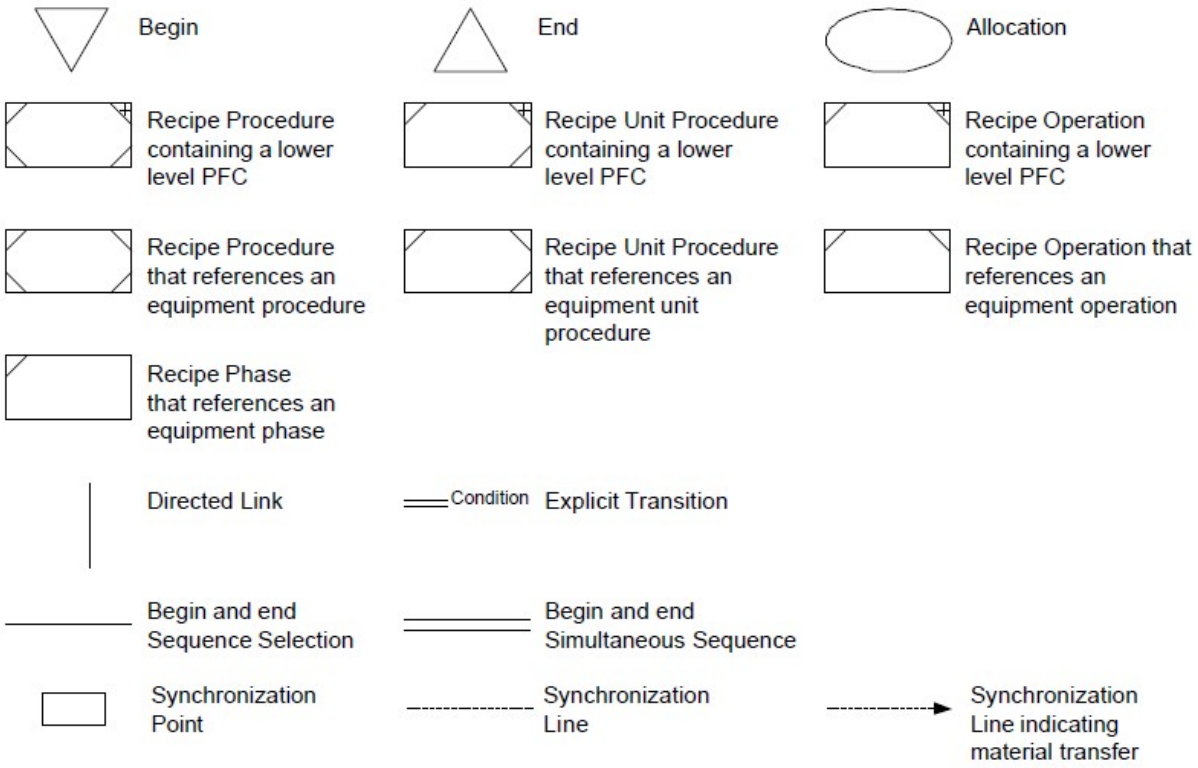


Рисунок 25 – Ключові символи Procedure Function Chart

Приклади PFC

Використовуючи символи PFC, проста операція, яка містить два етапи може бути зображена, як це показано на Рисунок 26. На цій діаграмі символ початку (begin symbol) вказує на те, де починається виконання PFC. У діаграмі PFC може бути тільки один символ початку. Символи, з’єднуючись прямими зв'язками забезпечують порядок виконання діаграми.

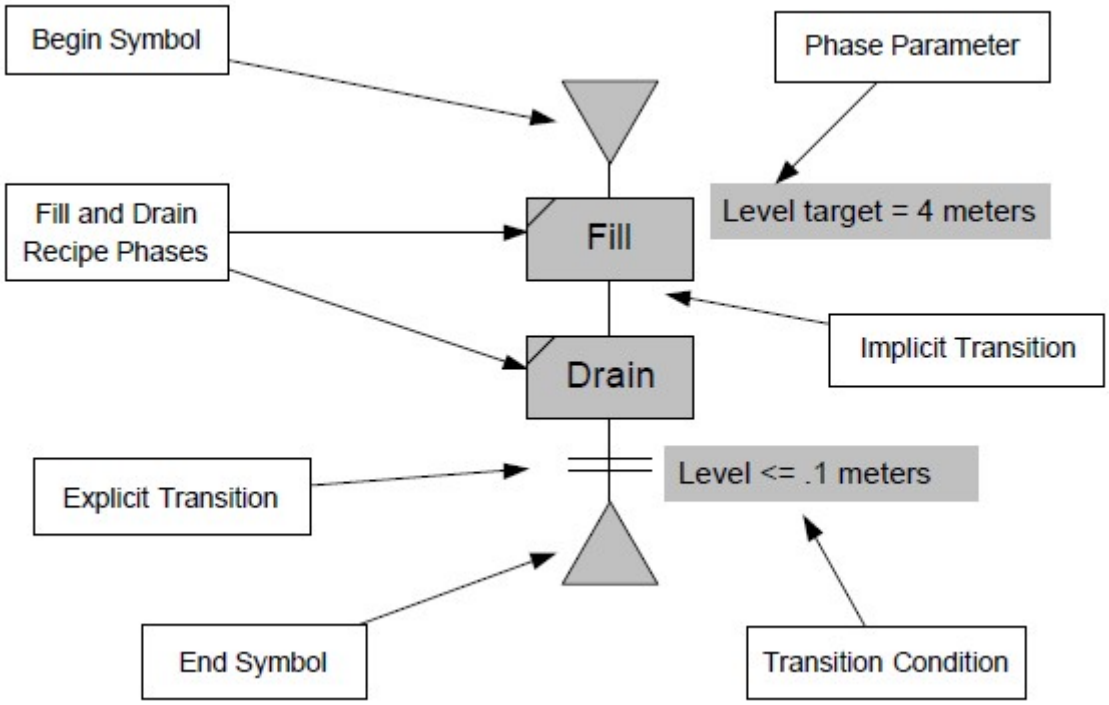


Рисунок 26 – Операції PFC

Рецептурні етапи "Fill"та "Drain" посилаються на апаратурні етапи, які зазвичай реалізовані у вигляді програмної логіки в контролері. Ці два рецептурні етапи зображені по-різному, щоб проілюструвати використання явних і неявних переходів.

Етап "Fill" використовує неявний перехід. Апаратурний етап " Fill" запрограмований таким чином, щоб при виконанні своєї мети перейти до стану "ЗАВЕРШЕНО". У цьому випадку мета полягає в тому, щоб заповнити ємність до рівня 4-х метрів. Мета була задана значенням параметра, а не умовою переходу, таким чином немає необхідності в зображенні переходу до наступного рецептурного етапу. У випадку, коли перехід не показуються його називають неявним переходом, що можна трактувати як "коли попередній етап завершив виконання". Для забезпечення сумісності з вимогою побудови переходів як "крок-перехід-крок" відповідно до IEC 61131-3, сам перехід залишається, хоч і не явно. Коли використовується неявний перехід, його умова не може містити логіку. У цьому випадку, коли апаратурний етап "Fill" переходить до стану "ЗАВЕРШЕНО", негайно запускається етап "Drain".

Рисунок 27 і Рисунок 28 розширюють концепції явних і неявних переходів. По суті, неявний перехід є більш зручною версією явного переходу, який би мав містити умову "виконання апаратурного етапу завершено", оскільки не потребує розміщення на PFC і задавання умови. У більшості за стосунків періодичного виробництва апаратурні етапи програмуються інженером так, щоб вони запускалися при виконанні рецептурної процедури, а завершувалися при досягненні якихось значень заданих в формулі або параметрах. Дуже принциповим при керуванні періодичним виробництвом є здатність апаратурного етапу самому завершувати свою логіку виконання. Звичайно, це може

включати в себе запрограмовану реакцію на явний перехід, який заданий в PFC. Проте, це повинно враховуватися в програмі, а не тільки простим бажанням автору рецепту довільно завершити виконання апаратурного процедурного елементу. Основна відмінність між цим і використанням Sequential Function Charts в тому, що в SFC після виконання умови переходу попередній крок зупиняється миттєво, а для виконання додаткових дій обслуговування в кроці немає ніякої можливості. Це раптове припинення апаратурного етапу не підходить для керування періодичним виробництвом. Цікаво, що в своєму огляді існуючих комерційно доступних batch-систем Комітет дійшов висновку, що в багатьох системах з використанням SFC для процедур цю проблему було виявлено. З метою задоволення потреб періодичного виробництва вирішували її, по суті, відхиляючись від IEC 61131-3 Sequential Function Charts.

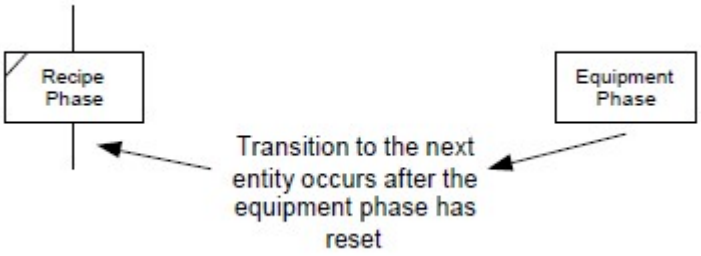


Рисунок 27 – Неявний перехід

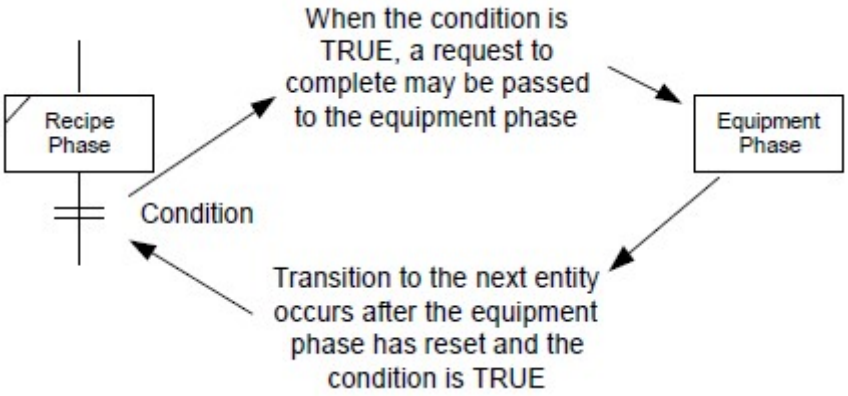


Рисунок 28 – Явний перехід

Повертаючись до Рисунку 26, якщо перехід після етапу "Fill" буде відображено, він може вміщувати умову "TRUE", "Заданий рівень досягнутого", або "етап завершився або відключився". Перехід може бути включений в якості явного переходу, якщо в ньому є потреба з точки зору кращого сприйняття.

Після рецептурного етапу "Drain" показаний явний перехід. У цьому випадку апаратурний етап "Drain" запрограмований для зливу матеріалу з ємності. Після того, як процес зливу стартував, логіка апаратурного етапу не закриває зливний клапан до тих пір, поки виконання рецепту PFC не дала команду. У цьому випадку виконання умови явного переходу "Рівень <= .1 метр" буде ініціювати завершення виконання логіки апаратурного етапу "Drain". При от-

риманні сигналу істинності переходу логіка апаратурного етапу закриває зливний клапан, виконує певні задані службові дії, і переходить в стан "ЗАВЕРШЕНО". Після завершення виконання апаратурного етапу виконання PFC йде повз явного переходу до символу кінця, що викликає завершення операції.

Як видно з цього прикладу, вибір між неявними і явними переходами віддається користувачу. Однакова логіка може бути реалізована як за допомогою використання параметрів з неявними переходам так з використанням явних переходів із задаванням умови переходу. Стандарт підтримує обидва стилі.

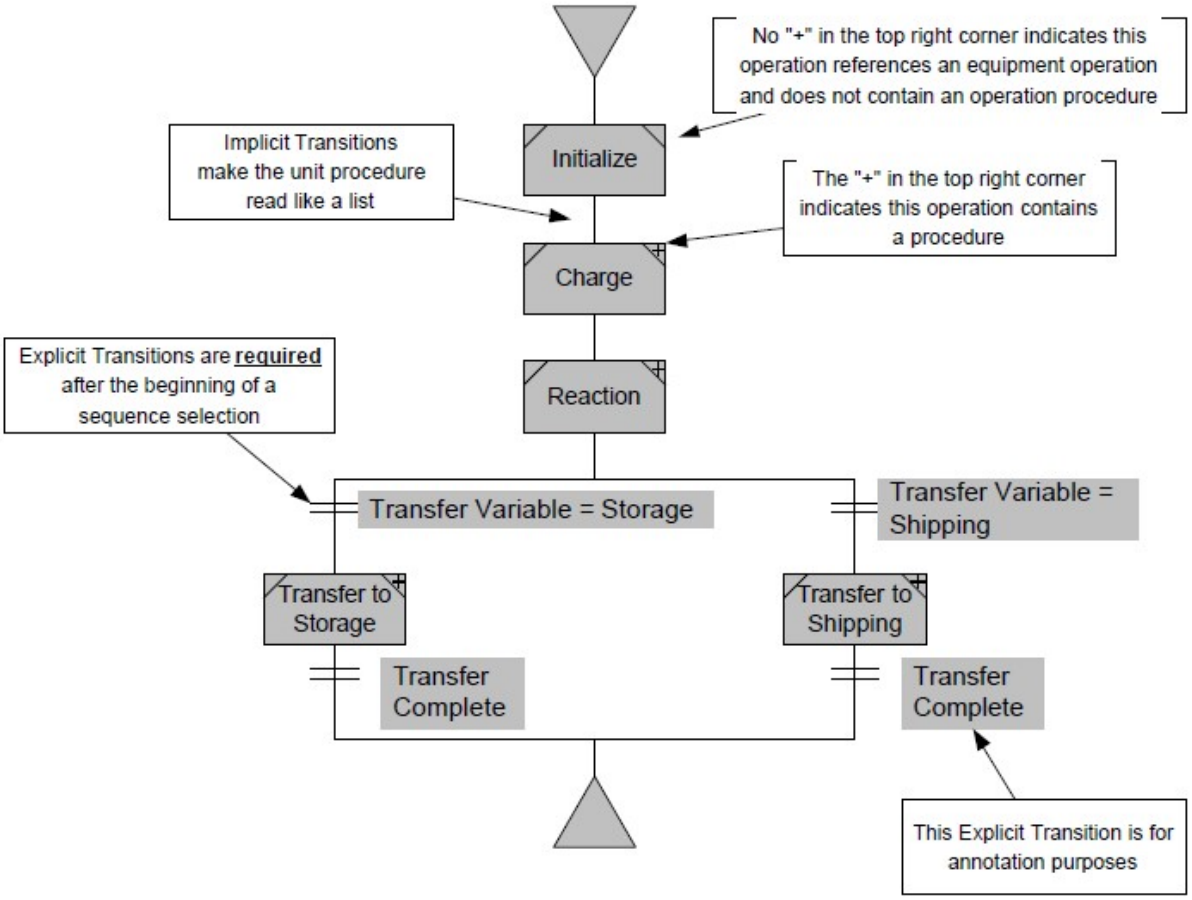
Слід зазначити, що кольори в діаграмах PFC в цій статті і зображення параметрів рецептурного етапу "Fill" тут вказані з метою демонстрації. Забарвлення не є частиною стандарту. Що стосується зображення формули/параметрів, стандартних станів, що повинні бути відображені, залежать від обставин і завжди пов'язані з рецептурним процедурним елементом. Таким чином, хоча це зображення не показано в стандарті воно відповідає вимогам стандарту.

Завершення операції в PFC показується кінцевим символом. Як і символ початку, на діаграмі він може бути тільки один. Єдиний символ кінця означає, що при використанні альтернативної або паралельної (одночасної) послідовності всі можливі шляхи повинні сходитися назад до одного шляху для досягнення єдиного символу кінця. Символи початку і кінця не виконуються, вони є просто графічними зображеннями.

На Рисунку 29 показана діаграма PFC для процедури апарату, що побудована з операцій. На цьому рівні рецепту процедурні елементи (тобто операції) можуть містити діаграму PFC з упорядкованого набору рецептурних процедурних елементів, або безпосередньо посилатися на апаратурний процедурний елемент (тобто апаратурну операцію). Для того, щоб показати, що процедурний елемент інкапсулює PFC використовується знак "+", який знаходиться у його верхньому правому кутку. Так, наприклад, рецептурна операція "Initialize" посилається на апаратурну операцію, тому вона не містить рецептурних етапів. Усі інші рецептурні операції інкапсулюють PFC, які в цьому випадку всі містять рецептурні етапи.

Використання неявних переходів після операції "Initialize" і "Charge" робить таку діаграму лаконічною (це рішення автора рецепта і інженерно-технічного персоналу, який написав логіку роботи) і робить першу частину діаграми PFC означення рецептурної процедури апарату такою ж читабельною як список.

Під символом альтернативного розходження необхідні явні переходи, неявні переходи використовуватись в цьому випадку не можуть.



Рисунку 29 – Діаграма PFC для процедури апарату

Потреби в зображенні на діаграмах PFC процедур апаратів і операцій не дуже відрізняються. Проте, потреби в зображенні на PFC рецептурних процедур технологічних комірок (procedure) значно відрізняються від цих двох. Рецептурні процедури технологічної комірки координують виконання асинхронних діяльностей процедур апаратів, які мають точки синхронізації, транспортування матеріалів і вимоги до обладнання. Крім того, в процедурі технологічної комірки необхідно надати якомога більше інформації високого рівня абстракції. На Рисунку 30 наведений приклад простої рецептурної процедури технологічної комірки. Вона включає дві процедури апарату, "Preparation" і "Reaction", які виконуються одночасно. (*Зверніть увагу, що з точки зору технологічного процесу спочатку повинна виконуватися процедура "Preparation" а потім "Reaction". Паралельність виконання необхідна для того, щоб була можливість процедурі апарату "Reaction*" синхронізуватися з процедурою "Preparation" для транспортування матеріалу. Це буде неможливо, якщо одна з процедур апарату завершила своє виконання.)

Символи виділення ресурсів (allocation) містять апаратурні вимоги до наступної за ним процедури апарату. Форма і мета символу виділення ресурсів була стандартизована в S88.02. Однак вміст був залишений неструктурованим. Він призначений для того, щоб містити правила виділення обладнання для процедури апарату. У цьому прикладі правила виділення ресурсів представ-

лені у вигляді списку можливих апаратів, які можуть бути використані відповідно до процедури апарату.

Коли явний перехід розміщується відразу після символу виділення ресурсів, він представляє собою стартову умову для наступної процедури апарату. У цьому прикладі процедура апарату "Preparation" починається тоді, коли виконається виділення ресурсу, тоді як процедура апарату "Reaction" для старту потребує втручання оператора.

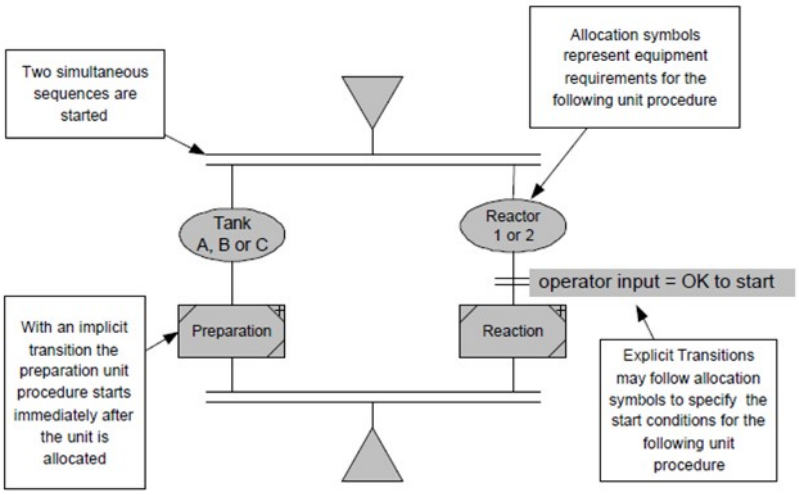
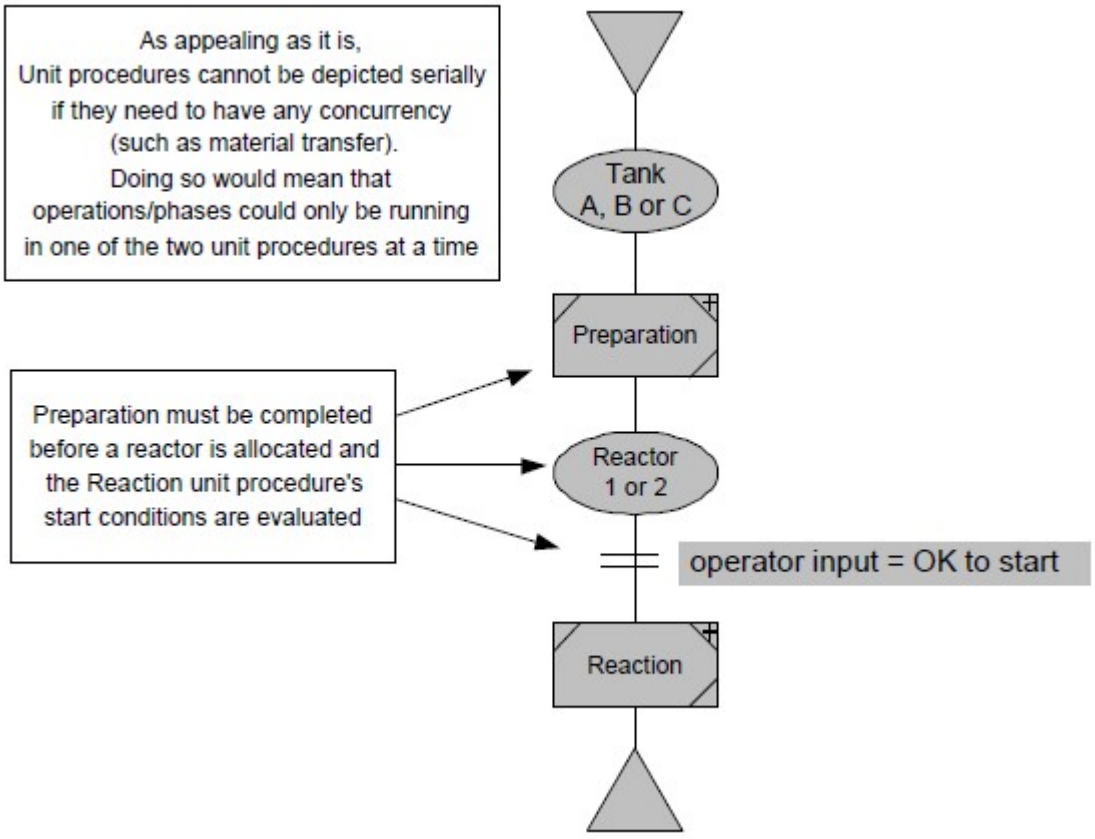


Рисунок 30 – Діаграма PFC рецептурної процедури технологічної комірки

Після того, як обидві процедури апарату завершують виконання, символ паралельного сходження викликає зближення двох активних послідовностей, що приводить до досягнення символу кінця, тим самим закінчуючи виконання рецепту.

Хоча Рисунок 30 вказує на те, що ці дві процедури апарату виконуються одночасно є багато інформації про рецептурну процедуру технологічної комірки, яка не показана на цій діаграмі. Використовуючи зображення яке показано на Рисунку 30 з більшою кількістю виконуваних процедур апаратів, як правило, призводить до того, що усі процедури апарату розміщуються як чергова гілка в паралельній структурі, а це тільки послаблює наочність цього методу опису. Крім того, на діаграмі не зображена синхронізація та транспортування матеріалу між апаратами. Перша інстинктивна думка вирішити проблему паралельності, зобразити усі процедури апаратів послідовно, як в списку. (*Це більш схоже до технологічної послідовності. Примітка перекладача.*). На Рисунку 31 наведено приклад такого зображення. На жаль, цей метод не працює. Проблема полягає в тому, що якщо процедури апаратів показати послідовно, то перша з них повинна бути завершена до того, як для другої може бути виділене обладнання і вона буде запущена на виконання. Це робить дуже важким керування транспортуванням матеріалу між апаратами.



Рисунку 31 – Діаграма PFC рецептурної процедури технологічної комірки з послідовним виконанням процедур апаратів

Для вирішення цієї проблеми необхідно зробити одну велику структуру паралельного виконання життєздатною. Для цього можна використати концепцію відносного часу на вертикальній осі, що використовується в Діаграмах Ганта. Слід зазначити, що час береться відносно майстер-рецепту, а частини керівних рецептів що ще не виконані не містять інформацію про абсолютний час, наприклад, як довго операція буде виконуватися. Так що для процедури майстер рецепту зображення розміру і початку виконання процедурних елементів (тобто процедур апаратів) є чисто відносним і задуманий як замітка. На Рисунку 32 показана та ж рецептурна процедура технологічної комірки що і на Рисунку 30, тільки нарисована таким чином, щоб показати відносний час виконання і точки синхронізації.

Початок і розміри двох процедур апаратів призначені для зображення того, що процедура апарату "Preparation" виконується раніше і в якийсь момент під час її виконання починає виконуватися правила виділення обладнання для процедури апарату "Reaction". Після виділення обладнання (Reactor 1 або 2) і підтвердження оператору відбувається запуск процедури апарату. Потім, в якийсь момент, ближче до кінця процедури "Preparation" і близько до початку "Reaction" відбувається транспортування матеріалу з "Preparation tank" до "Reactor" для використання його в процедурі "Reaction". Після транспортування, процедура "Preparation" продовжує працювати протягом деякого часу, а "Reaction" виконується протягом більшого часу. Коли обидві процедури апарату завершуються, виконання рецепту закінчується.

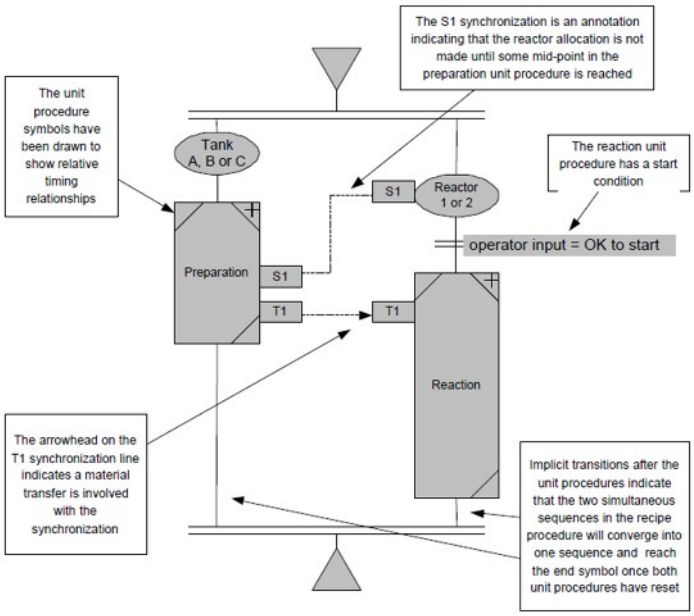


Рисунок 32 – Діаграма PFC рецептурної процедури технологічної комірки з зображенням відносних залежностей

Хоч це не конкретизує інформацію про події, Рисунок 32 дійсно забезпечує більше інформації, ніж Рисунок 30 для того ж прикладу. Для виконання цього додатково можна показати кілька рівнів діаграм на одній і тій же PFC. Так на Рисунку 33 символи процедур апаратів зображені розширеними, що показує інкапсульовані всередині діаграми PFC. Таким чином можна бачити, що процедури "Preparation" і "Reaction" включають в себе чотири операції. Точка синхронізації S1 зав’язана з відбором проби з танку Preparation (операція Sample). Так само транспортування матеріалу T1 досягається за рахунок операцій "Transfer to Reactor" і " Receive from Prep."

Метою цього стандарту є дозволити, за необхідності, застосовувати типів візуального розширення, що показані на Рисунку 32 і Рисунку 33, але це не є обов’язковим.

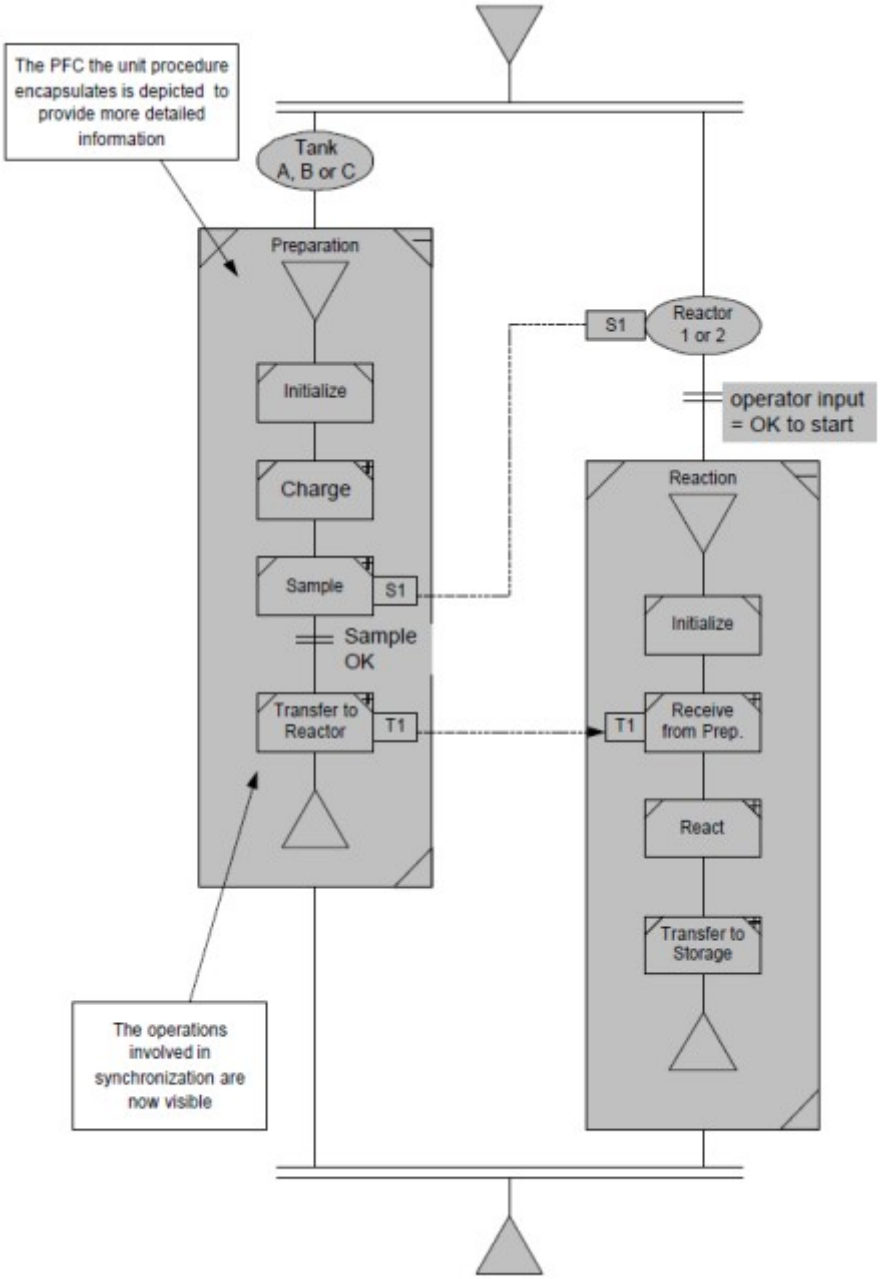


Рисунок 33 – Два рівні процедурних елементів показані на одній діаграмі PFC

Додаткові правила для PFC

Нотація PFC призначена для сприяння обміну даними рецептів між системами і спрощення вивчення кожної нової системи керування періодичним виробництвом. Проте, слід визнати, що жодна парадигма не є незмінною, а еволюція та інновації є постійними. Таким чином, стандарт допускає розширення нотації PFC. Єдиною вимогою при цьому є чітке означення таких розширень.

Як уже згадувалося раніше, процедура є "клеєм", який разом утримує різні категорії інформації в рецепті. Приклади показують варіанти зображення формули або параметрів, а символи виділення ресурсу забезпечують зображення вимог до обладнання. Зображення заголовка і "іншої" інформації залишається гнучким, як це буде визнано доцільним для застосування.

ЛАБОРАТОРНА РОБОТА 1. ПІДГОТОВКА РОБОЧОГО МІСЦЯ

Мета роботи – підготовка робочого простору SCADA zenon для подальшого використання в лабораторних роботах.

Загальні теоретичні відомості

Використання спеціального програмного забезпечення **SCADA zenon** значно спрощує розроблення прикладного програмного забезпечення для реалізації автоматизованого робочого місця (АРМ) оператора різного призначення. Таке інструментальне програмне забезпечення належить до класу **SCADA/HMI**. Основний принцип розроблення з використанням цих інструментів – це так зване *"Конфігурування замість програмування",* що різко зменшує витрачений час та вірогідність помилок, адже функціональність АРМів у своїй базовій частині мало залежить від особливостей виробництва. Програмні пакети для розроблення АРМів на базі комп'ютерів прийнято називати "**SCADA-програмами**", або просто "SCADA", а для панелей оператора – "**HMI-програмами**", або просто "HMI". Надалі ми будемо використовувати загальний термін, який об'єднує ці поняття – програми **SCADA/HMI**.

Більшість програм SCADA/HMI має типовий набір функціональних можливостей для реалізації завдань АРМів:

- збирання інформації про контрольовані технологічні параметри (отримання даних в реальному часі) з контролерів та засобів віддаленого вводу/виводу;
- графічне представлення стану технологічного процесу і обладнання в зручній для сприйняття формі у вигляді мнемосхем;
- вторинна обробка інформації (масштабування, обмеження вводу, перевірка коректності тощо);
- приймання команд оператора і передача їх на контролер або засіб віддаленого виводу;
- збереження даних реального часу в архівах даних і графічне представлення історичної інформації в зручній для сприйняття формі у вигляді графіків, гістограм тощо;
- сповіщення експлуатаційного і обслуговуючого персоналу про виявлені аварійні події в технологічному процесі і програмно-апаратних засобах;
- фіксація в електронних журналах виникнення аварійних подій у контрольованому технологічному процесі та дій експлуатаційного персоналу;
- формування звітів на основі архівної інформації, тривог та даних реального часу;
- обмін інформацією з автоматизованими системами управління виробництвом та підприємством у складі інтегрованих систем управління;
- підтримка мов програмування високого рівня, наприклад, VBA;
- захист від несанкціонованого доступу до компонент і файлів.

Лабораторний практикум базується на базових знаннях про SCADA zenon, отриманих на попередніх курсах. За необхідності повторення матеріалу пе-

рейдіть на сторінку курсу «Людино-машинні інтрфейси». Режим доступу: http://edu.asu.in.ua/login/index.php(доступ під користувачем «гість»).

Додаткові матеріали по zenon можна знайти на сайті http://www.copa-data.com.ua/

Офіційним дистрибутором компанії COPA DATA в Україні є компанія СВ Альтера.

Нижче зазначені посилання на образи інсталяції SCADA zenon.

Лабораторний практикум проводиться на базі SCADA zenon 7.2 або 7.5, нижчі версії не підтримують модуль zenon Batch. Дана версія потребує ОС Windows 7 та вище. Нижче наведені посилання на ці версії.

- Zenon 7.5 (сумісний з Windows10). Режим доступу:

http://download.copadata.com/fileadmin/user_upload/Downloads/installation_cd/Zenon_750/SP0B25796/Zenon750_FinalBuild25796_COPA-DATA.iso

- Zenon 7.20. Режим доступу:

http://download.copadata.com/fileadmin/user_upload/Downloads/installation_cd/Zenon_720/SP0B20544/Zenon720SP0_LanguageBuild20544_COPA-DATA.iso

Файл правки реєстру для продовження терміну використання (для версії 2).

Комп'ютер, на який буде інсталюватися програмне забезпечення SCADA zenon 7.20 повинен відповідати наступним вимогам:

	Мінімальні	Рекомендовані
ОС	Windows 7	Windows 7 або новіша
CPU	Pentium 4	Quad Core
RAM	2 GB	4 GB
HD	25 GB вільного місця	200 GB вільного місця

Якщо ваша система та ресурси не відповідають наведеним в таблиці, можете використовувати більш старіші версії. При цьому деякі пункти практикуму будуть виконуватися дещо іншим способом.

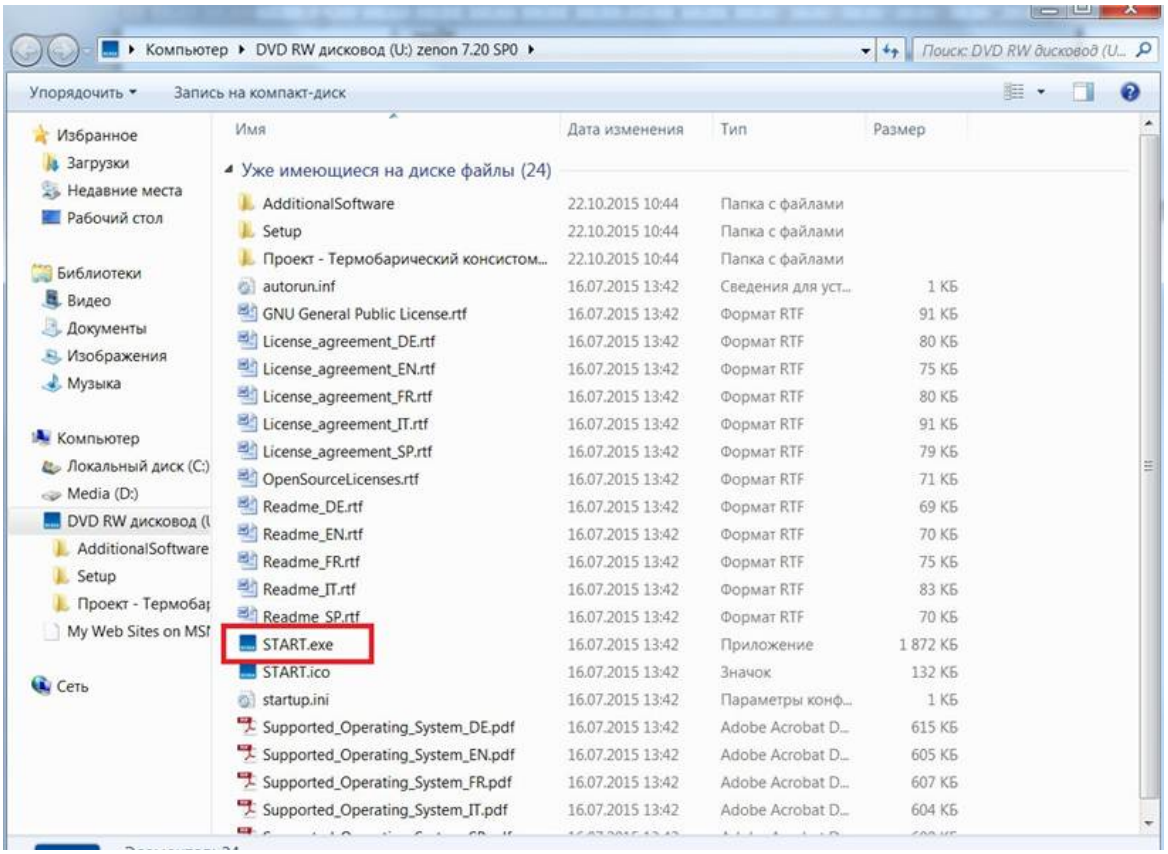
Завдання до виконання лабораторної роботи

1. Ознайомитися з програмою SCADA/HMI Zenon.
2. Підготувати робочий простір SCADA zenon.

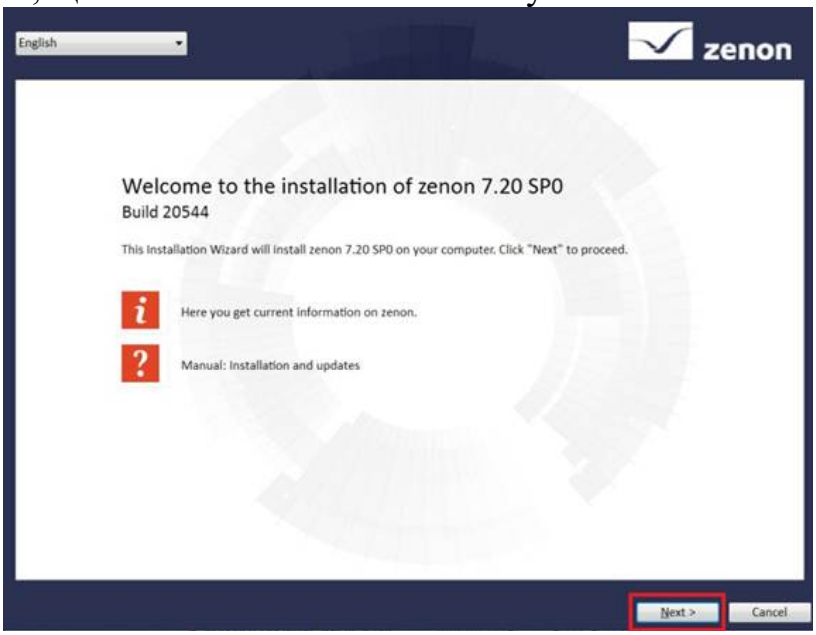
Порядок проведення роботи

1.1 Інсталюйте середовище розробки разом з середовищем виконання

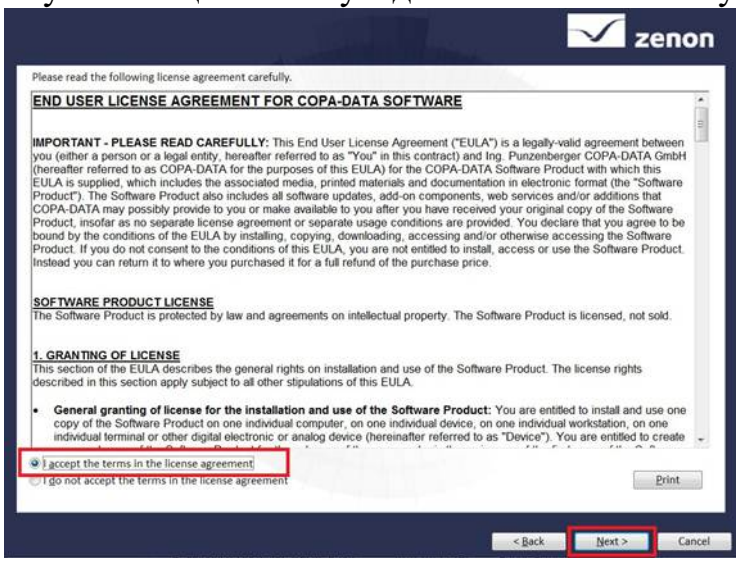
- Завантажте дистрибутив zenon задопомогою посилань, що зазначені раніше. Запустіть процес установки системи за допомогою файлу “START.exe”.



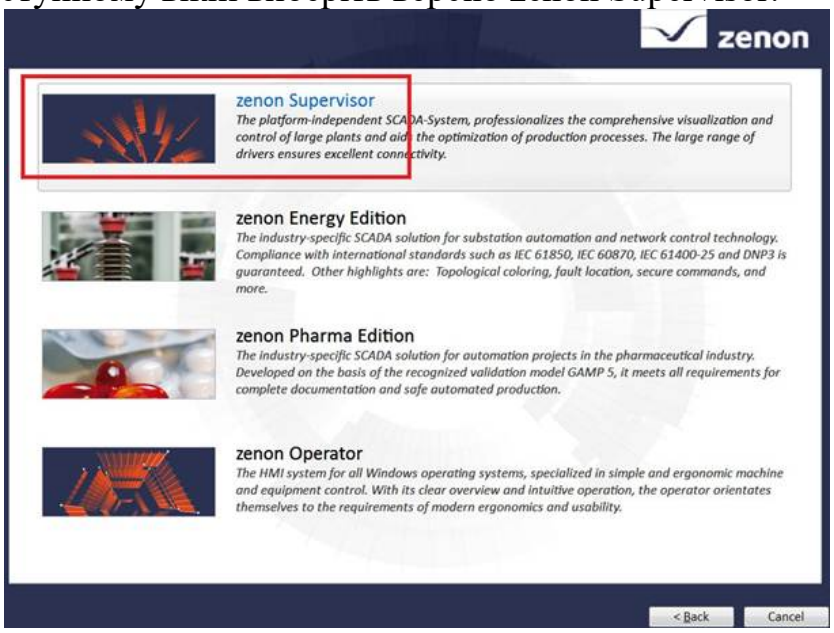
- У вікні, що з’явилось натисніть кнопку «Next».



- Прийміть умови ліцензійної угоди і натисніть кнопку «Next».



- Виберіть інсталяцію середовища розробки проекту zenon Editor
- У наступному вікні виберіть версію zenon Supervisor.



- На завершальному етапі виберіть пункт **Install now** та завершіть встановлення.

Контрольні питання

1. Що таке SCADA-програми?
2. Який типовий набір функціональних можливостей для реалізації завдань АРМів має SCADA/HMI?

ЛАБОРАТОРНА РОБОТА 2. ОСНОВИ РОБОТИ З BATCH CONTROL

Мета роботи – навчитися працювати з базовими елементами ISA-88 в zenon Batch Control: етапами та апаратами; рецептами; тегами.

Загальні теоретичні відомості

Метою даного проекту є розуміння того, як працює модуль Batch Control, як можна створювати проекти zenon з Batch Control з незалежним один від одного створенням і виконанням рецептів. У зв'язку з цим, основний акцент робиться на передачу знань про модуль Batch Control і менше про розробку функціональних рецептів, які відповідають вимогам реального виробництва.

Модуль Batch Control надає можливість автоматизувати періодичні виробничі процеси, що орієнтовані на виготовлення партій. Модуль відповідає вимогам стандарту ANSI/ISA-88.01-1995, також відомого як ANSI/ISA-S88.

Модуль розроблений таким чином, щоб бути незалежним від реалізації систем керування. Це значить, що передача даних відбувається через всі доступні драйвери zenon з будь-якими ПЛК або навіть віддаленими терміналами RTU. Ці пристрої можуть виконувати тільки дії процесу. Вся обробка рецепта проводиться на комп'ютері в виконавчій машині рецептів REE (Recipe Execution Engine). Для змін в Batch рецепті (надалі рецепт) або для створення нових майстер рецептів (Master Recipe), ніяких змін в коді ПЛК проводити не потрібно.

Модуль слідує жорсткому розділенню процедури рецепту (згідно ISA процедурної моделі керування) і виконання технологічних функцій (згідно ISA процесної моделі), як описано в ISA-S88, главі 5.2.1.

Розглянемо реальний приклад для легшого розуміння. Це проект для керування об’єктом «Танк перемішування» з використанням Batch Control. Припускаємо, що «Танк» можна наповнювати і вивантажувати, а його вміст – перемішувати. Всі інші функції і властивості «Танку» можуть бути вільно означені.

Коротко розглянемо основні поняття.

Рецепт (***Recipe***) є об'єктом, який містить мінімальний набір інформації, що однозначно ставить вимоги до виробництва для конкретного продукту. Рецепти надають спосіб для опису продуктів і способи вироблення цих продуктів. Залежно від конкретних вимог підприємства, можуть існувати різні типи рецепту. Проте, в стандарті ISA-88 обговорюються лише чотири типи рецептів:

- загальний рецепт (на рівні підприємства):
- місцевий рецепт (на рівні виробничою площадки);
- майстер рецепт (шаблонний рецепт технологічної комірки);
- керівний рецепт (конкретний екземпляр рецепту для кожної партії).

Майстер рецепт (***Master recipe***) орієнтований на конкретну технологічну комірку і являється «шаблоном» для керівного рецепту, що використовується для створення однієї партії продукту (напівпродукту). Він може бути отриманий із загального або місцевого рецепту, однак може бути створений як неза-

лежна сутність, якщо автор рецепту має знання про необхідний процес і продукт.

Центральним елементом рецептів є ***етапи*** (phase). Вони призначені для передачі значень в ПЛК і можуть керувати виконанням рецепту з використанням умов. Кожен етап розміщений в ***апарати*** (unit). Апарати представляють собою частини машин або обладнання, наприклад, таких як Танк. Цей зв'язок відображається в zenon у вигляді деревовидної структури. Ця структура відображається у вікні детального перегляду (detailwindow), якщо натиснути по розділу проекту «**Batch Control»**в дереві проекту.

На Рисунку 2.1 показана деревовидна структура апаратів в проекті zenon. Там є два апарати під головним вузлом («Unit1» і «Unit2»). «Unit1» містить дваетапи «Phase1» і «Phase2». У правій частині вікна деталізації відображаються властивості або вміст об'єкта, виділеного в дереві.

Рисунок 2.1. Деревовидна структура апаратів (Unit)

Етап може взаємодіяти з контролером за допомогою тегів. При цьому проводиться різниця між ***командними тегами*** (***command tags***) і ***зворотними тегами*** (***return tags***):

- Командні теги використовуються для запису значень в контролер.
- Зворотні тегиможуть бути використані для отримання значень з контролера.
- В обох випадках фактичний зв'язок здійснюється за допомогою змінних. Тому з кожнимтегом повинні бути пов'язані змінні.
- Командні теги в свою чергу поділяються на ***ініціальнітеги*** (***initialtags***) і ***змінні теги*** (***value tags***):
  - ініціальні теги – це командні параметри, які задаються перед запуском етапу;
  - змінні теги – це командні параметри, які передаються в контролер по зміні під час роботи етапу.

Завдання до виконання лабораторної роботи

1. Створити проект в zenon та налаштувати драйвер.
2. Розглянути та створити апарати та етапи.
3. Розробити екран Batch Control.
4. Створити і виконати рецепт.
5. Створити теги для взаємодії з ПЛК.
6. Створити умови для для хронологічного виконання рецепту.

Порядок проведення роботи

2.1 Підготовка роботи

2.1.1 Запустіть редактор zenon і створіть новий проект. Додайте в проект драйвер Modbus TCP/IP і налаштуйте його на роботу в режимі імітації (Рисунок 2.2).

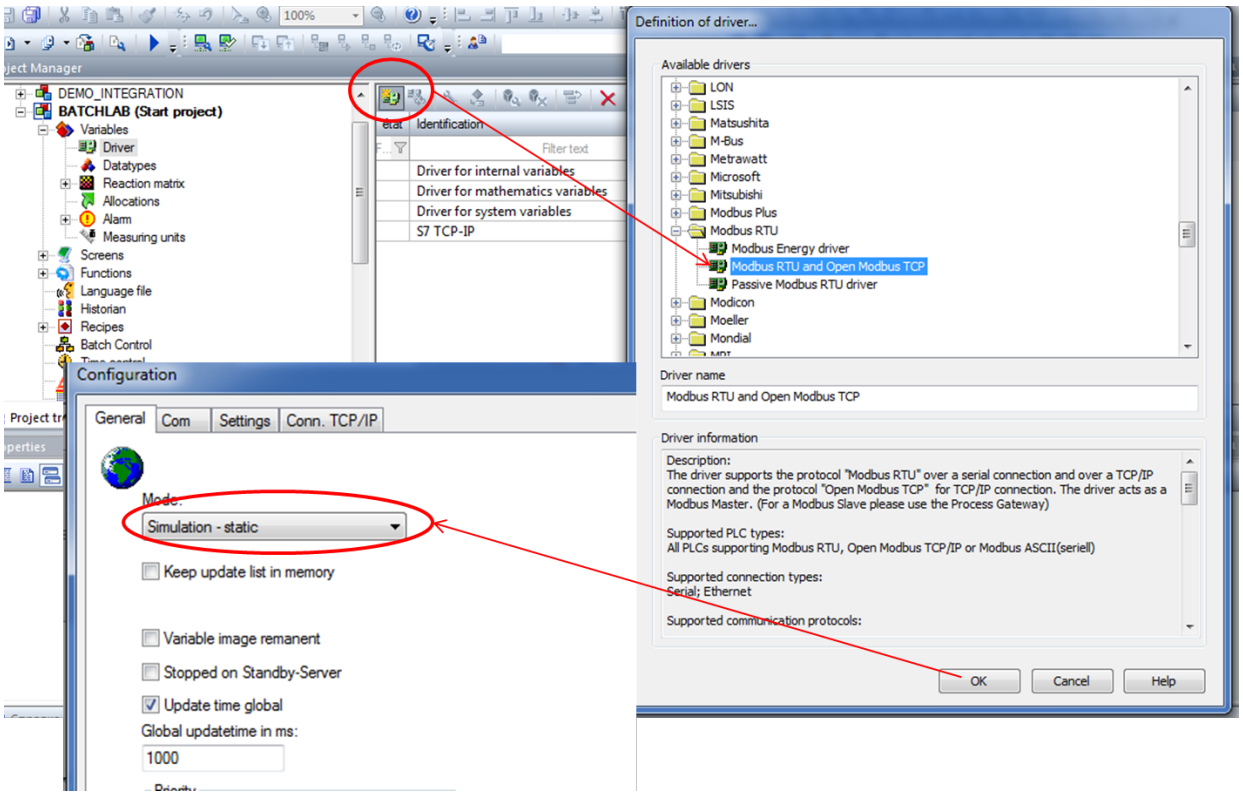


Рисунок 2.2 – Вставка драйверу Modbus TCP/IP та налаштування його на режим імітації

2.2 Апарати та етапи (Units and phases)

2.2.1 Створимо в дереві апарат «Танк», який необхідно помити. Для цього зробимо наступні дії:

- Використовуючи контекстне меню або відповідну іконку в панелі інструментів створіть новий апарат (Рисунок 2.3).

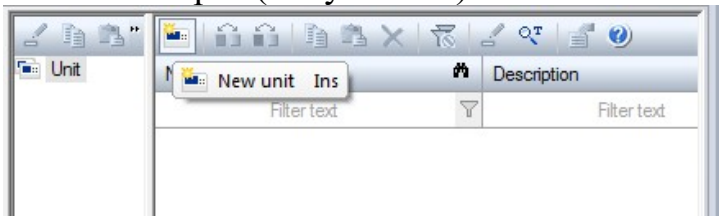


Рисунок 2.3 – Створення нового апарату (Unit)

- Змініть ім'я апарату на «Танк1».Можна для цього використати клавішу F2.
- Додайте новий етап (phase) до апарату. Для цього використовуйте

контекстне меню апарату або відповідну іконку в панелі інструментів списку.

- Зімніть назву етапу на «Перемішувати».
- Створіть ще два етапи «Наповнити» та «Вивантажити».

2.2.2 Надалі створимо перші рецепти з використанням цих етапів (Рисунок 2.4). Пізніше додамо ще декілька етапів відповідно до вимог навчального проекту.

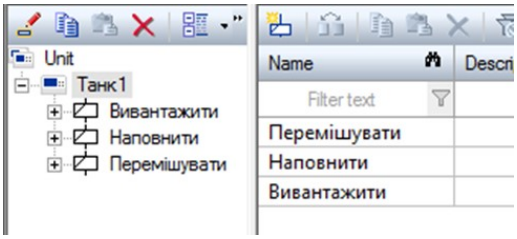


Рисунок 2.4 – Етапи апарату «Танк1»

2.3 Екран Batch Control

2.3.1 Рецепти можуть бути створені і виконані тільки в Runtime (режимі виконання). Для цього попередньо необхідно створити екран типу «BatchControl».

- Створіть два фрейми (frame) для екранів:

- «Основний шаблон екрану» – десь на 8/9 верхньої частини вікна для відображення основних екранів.
- «Шаблон меню» – десь на 1/9 нижньої частини вікна для екранів з кнопками перемикання.

- Створіть новий екран на базі шаблону меню і дайте йому ім'я «Екран меню».
- Створіть функцію перемикання на новостворений екран (ScreenSwitch)з іменем «Перемикання на екран меню».
- Зробіть даний екран стартовим, якщо він таким не являється (Властивості проекту → Graphical Design →Start Screen)
- Створіть функцію «Reload project on line». Створіть на екрані меню кнопку для виклику функції.
- Створіть новий екран на базі основного шаблону і дайте йому ім'я «Екран Batch». Змініть тип екрану на «Batch Control».
- Добавте на екран елемент керування Recipe Editor (Рисунок 2.5). Розтягніть його десь на 2/3 ширини екрану по горизонталі і на всю висоту по вертикалі.

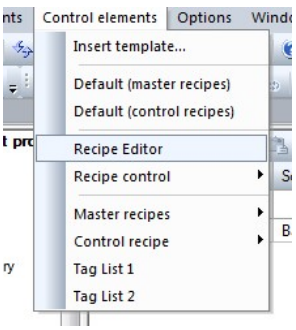


Рисунок 2.5 – Додавання елементу Recipe Editor

2.3.2 Для екранів типу Batch Control доступні і інші елементи, але зосередимося тільки на Recipe Editor, додаткові елементи керування будуть додані пізніше.

- Створіть функцію перемикання на даний екран. У вікні налаштування фільтра залиште все за замовченням і натисніть «ОК». Дайте ім'я функції, наприклад «Перемикання на екран Batch».
- На екрані меню вашого проекту створіть нову кнопку і позначте її як «Екран Batch». Прив'яжіть до неї функцію перемикання на екран Batch.

За допомогою цих кількох кроків, Ви створили основу для створення рецепта в Runtime і можливість його виконати.

2.4 Створення і виконання рецептів

2.4.1 На цьому кроці Ви будете створювати і виконувати свій перший рецепт. Для цього необхідно виконати наступні кроки:

- Запустіть zenon Runtime.
- Використовуючи кнопку на панелі меню, перейдіть на екран «Екран Batch».
- Покажіть список майстер рецептів в редакторі рецептів. Для цього натисніть на іконку в правому нижньому кутку редактора рецептів (recipe editor) і виберіть Master Recipe List (Рисунок 2.6). З'являється панель переліку майстер рецептів, яку можна рухати і закріплювати в межах редактора рецептів.

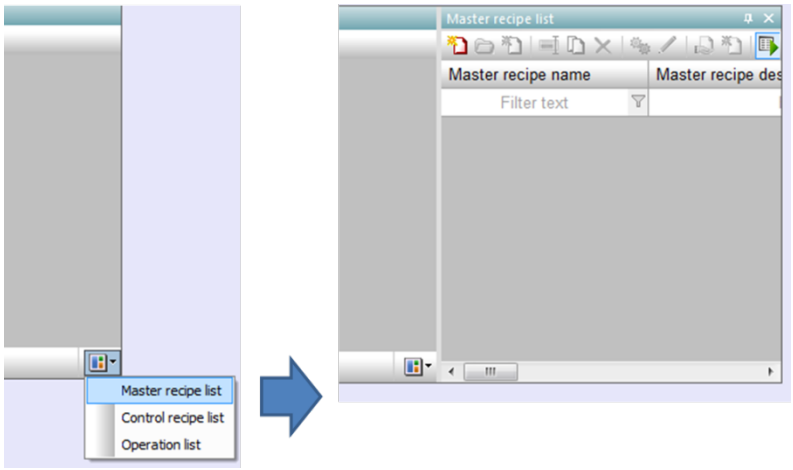


Рисунок 2.6 – Відображення списку майстер рецептів в Recipe Editor

- У списку майстер рецептів створіть новий рецепт: New Master recipe. Відображається діалог для налаштування нового рецепта (Рисунок 2.7).

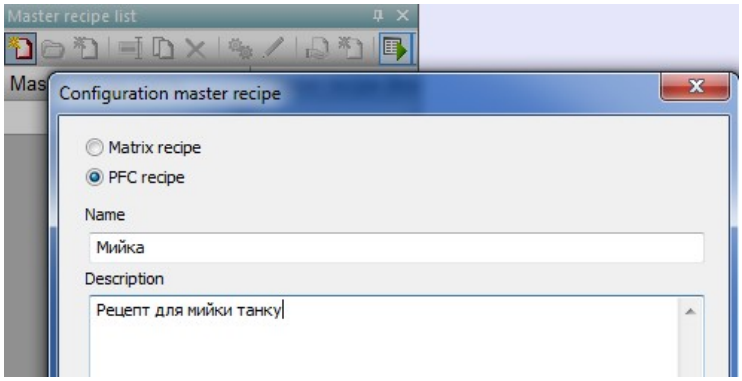


Рисунок 2.7 – Вікно створення нового рецепту

- Виберіть тип «**PFC recipe»** і введіть в полі **name** назву рецепту «Мийка». Можна також в полі Description ввести текст для опису. Після підтвердження новий рецепт автоматично відкривається в редакторі рецептів.

2.4.2 Рецепт не має ніяких інших елементів за винятком стартового елемента і кінцевого елемента. Він знаходиться в режимі редагування, що показано зображенням олівця на лівій частині панелі статусу. У цьому режимі рецепт може бути тільки відредагований. Для виконання рецепту необхідно перевести в тестовий режим.

- Додайте три етапи, які ви заздалегідь створили в Zenon Editor. Для цього натисніть на іконку «**Insert phase»** в панелі інструментів і помістіть етапи на робочу область рецепту.

При переміщенні покажчика миші на рецепт, позиція, в яку може бути вставлений елемент, буде виділена зеленим кольором. Позиція, де вставка неможлива, виділена червоним кольором. Помістіть етап в потрібне положення, натиснувши на потрібну позицію. З'являється діалог для вибору етапу.

- У діалозі виберіть етап «Наповнити».

Після того, як ви вставили етап, ви можете додати інші етапи, якізахочете. Щоб перейти з режиму вставки в режим вибору, натисніть на іконку «**Edit mode»**в панелі інструментів або натисніть клавішу Esc.

- Вставте два інші етапи.
- Переключіться в режим редагування.
- Перемістіть рецептурні елементи, як показано на Рисунку 2.8.

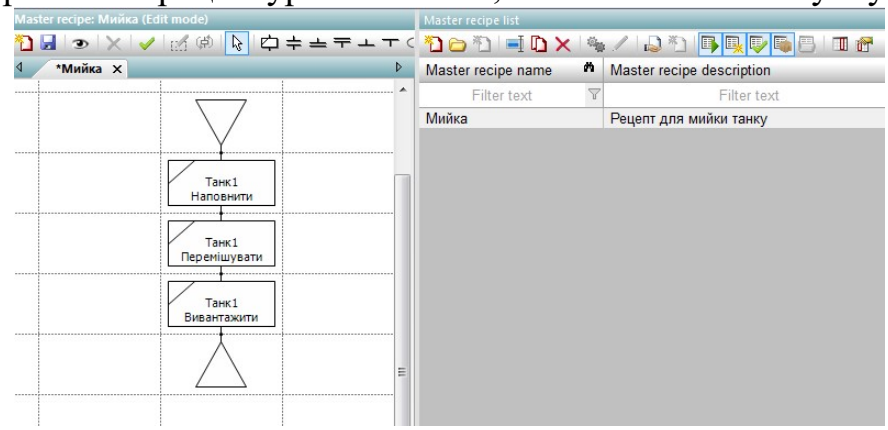


Рисунок 2.8 – Додавання етапів до рецепту

2.4.3 Рецепт створено. Тепер необхідно перевірити його роботу:

- Переведіть рецепт в режим тестування. Для цього використовуйте відповідну іконку в панелі інструментів списку рецептів або редактора рецептів (Рисунок 2.9). Підтвердьте свою дію у наступному діалоговому вікні.

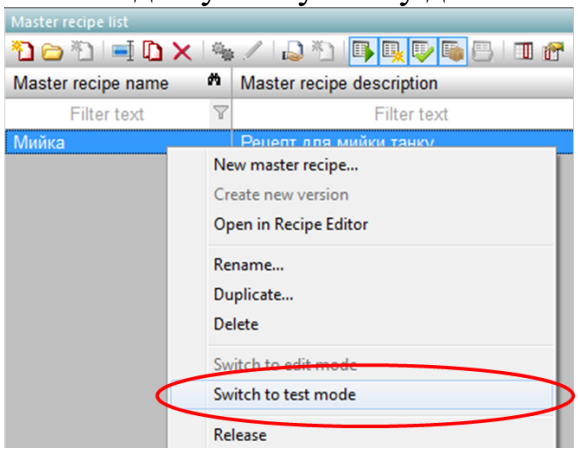


Рисунок 2.9 – Переведення в режим тестування

У рядку заголовка редактора рецептів тепер можна побачити, що рецепт знаходиться в тестовому режимі, а не в режимі редагування (Рисунок 2.10).

Рисунок 2.10 – Панель редактору в режимах редагування та тестування

- Запустіть рецепт на виконання за допомогою кнопки «Start Recipe» на панелі інструментів редактора рецептів.
- Збережіть рецепт за допомогою кнопки збереження.

Примітки:

Всі три функції змінюють колір в зелений один за іншим, а потім синій. Колір заливки показує стан етапу.

Колір	Значить
зелений	Етап зараз виконується
синій	Етап завершив виконання.

Про інші статуси етапів будуть розглянуті пізніше.

2.5 Теги

2.5.1 Ви успішно створили і виконали свій перший рецепт. Однак, навіть цей рецепт не виконує зв'язку з PLC, і, отже, не може служити для керування обладнанням. Тому потрібно створити теги, як будуть взаємодіяти з ПЛК че-

рез змінні.

Перейдемо до створення параметрів. На етапі заповнення («Наповнити») необхідно зробити заданий рівень заповнення за допомогою тега. Виконайте наступні дії:

- Перейдіть до zenon Editor.
- Натисніть на етап «Наповнити» у «Танк1». У області списку відображається список параметрів обраного етапу, який на даний момент порожній (Рисунок 2.11).
- На панелі інструментів натисніть «**New value tag**»

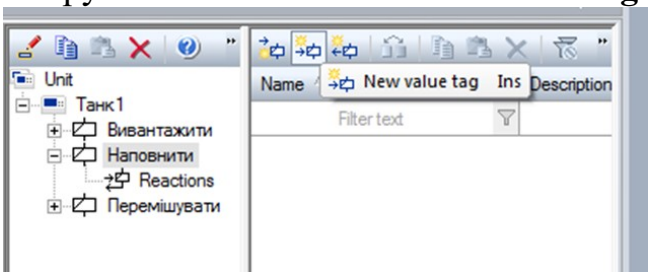
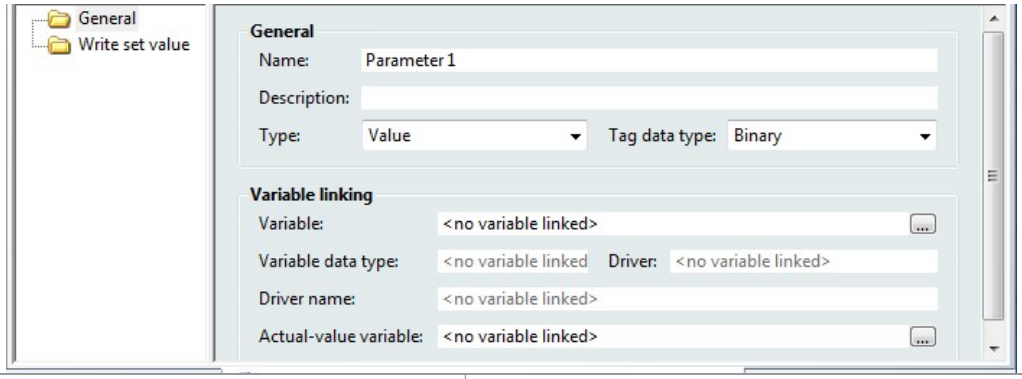
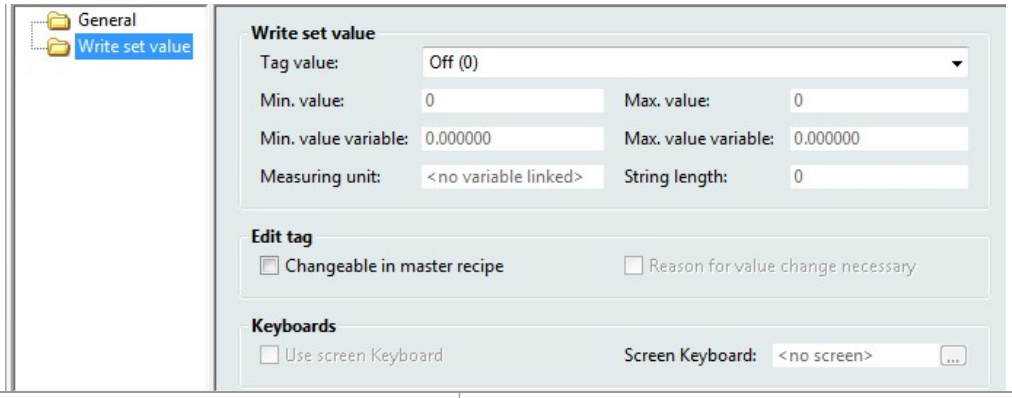


Рисунок 2.11 – Список параметрів для етапу

У список вставляється рядок, який представляє собою змінний тег (**value tag**) з іменем «Parameter1». Стовпці списку показують властивості тега, які також показані у вікні властивостей.

Розглянемо найбільш важливі властивості.

Властивість	Опис
Загальні (General) 	
Name(Ім’я)	Довільне ім’я тегу. Мусить бути унікальним в межах етапу (phase).
Description(Опис)	Текст для детального опису тегу (не обов'язково).
Type (Тип)	Тип тегу: Initial, value або return. Тип тегу може бути змінений у будь який момент часу.
Tag data type(Тип даних тегу)	Binary, numerical, string або duration (time). Означує тип тегу і фільтри для вибору змінної .

Variablelinking(Прив'язка до змінної)	Прив'язка змінної до тегу і його властивостей .
Установки записи значення (Writesetvalue) 	
TAG value (Значення тегу)	Значення яке має бути записане в змінну при виконанні етапу.
Min./Max. Value (Мін. максим значення)	Межі, в яких у режимі виконання може бути змінено значення тегу.
Min./Max. Valueofvariable (Мін. максим значення змінної)	Діапазон значень змінної. Межі тегу повинні бути в межах змінної.
Edit tag (редагування тегу)	
Change able in the master recipe (Змінюваний в майстер рецепті)	Активний: значення тегу можна редагувати в режимі виконання в рецепті.

- Введіть ім'я (**name)** для тегу «Заданий рівень наповнення».
- Введіть за бажанням опис (**description)**.
- Змініть тип даних тегів (**tag data type**) на «numerical».
- Натисніть на кнопку вибору змінної «Variable→ Variable Linking» (…). Створіть нову змінну типу realбезпосередньо в діалозі вибору змінних. Використовуйте драйвер, який Ви створили на самому початку.Дайте змінній ім'я «Заданий рівень наповненняT1».
- Зайдіть в налаштування змінної:

- змініть межі змінної 0 і 100 (Value Calculation→ Value Range PLC; (Value Calculation → adjustment linear);

- введіть % в якості одиниці вимірювання (General → Measuring unit).

- для Modbus TCP/IP (драйвер MODRTU) в області Addresing введіть:

  - Driver Object Type = Holding Register
  - Data Type = REAL
  - Net Address = 0
  - Offset = 0

- Перейдіть до групи властивостей «Write Set Value» створеного тегу етапу, де видно межі та одиниці вимірювання змінної.
- Активуйте опцію **Change able in the master recipe.**

- Введіть **tag value** = 20, **Min.Value =** 10**, Max.Value =** 100.

Як тільки етап «Наповнити» виконується, то в «Заданий рівень наповнення Т1» записується значення 20. Однак це значення може бути змінене в рецепті в межах від 10 до 100.

- Вставте на «Екран Batch» елемент **Numerical Value**, в якому відображається значення змінної «Заданий рівень наповнення Т1».
- Тепер скомпілюйте змінені файли виконання, переключіться в Zenon Runtime і натисніть кнопку Reload.
- Виконайте рецепт «Мийка» і спостерігайте за значенням змінної. Як тільки етап «Наповнити» активується відображене значення змінюється з 0% на 20%.

2.5.2 Розглянемо зміну тегу в рецепті. Якщо заданий рівень заповнення потрібен не 20%, а 50%, то це можна зробити безпосередньо в режимі Runtime, без необхідності змінювати конфігурацію проекту в редакторі. Для цього необхідно виконати наступні дії:

- У Runtime зробіть подвійний клік на етапі «Наповнити». З'являється діалогове вікно для настройки етапу.
- Переключіться на закладку **Parameters**.Тут відображається список параметрів етапу (Рисунок 2.12). За бажанням Ви можете переміщувати або приховати стовпці для можливості побачити найважливішу інформацію.

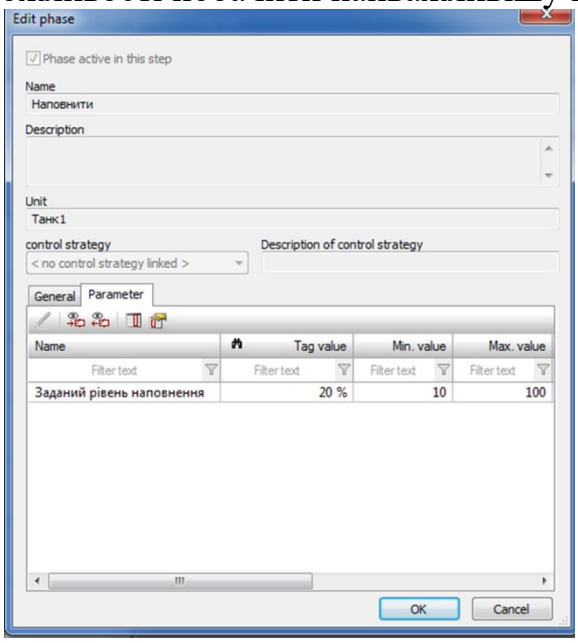


Рисунок 2.12 – Налаштування параметрів етапу

- Подвійним натисканням по імені тегу відкрийте діалогове вікно **change the tag properties** (Рисунок 2.13).

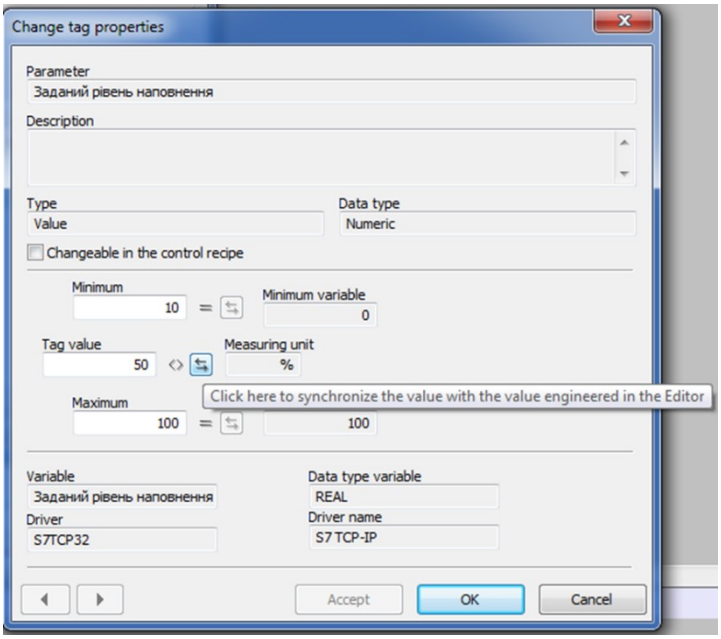


Рисунок 2.13 – Налаштування властивостей тегу в Runtime

- Змініть поле tag value рівним 50.
- Підтвердьте дії кнопкою Ок.
- Виконайте рецепт знову.

Значення змінної тепер змінено в 50%.

Значення тегів в редагованих в zenon Editor майстер рецептах завжди оновлюються тільки там. У Runtimезавжди можна синхронізуватися з цим значенням (кнопка).

2.5.3 Завдання для самостійного опрацювання.

- Створіть необхідні командні параметри і змінні для етапів «Перемішувати»і «Вивантажити»:

- для етапу «Перемішувати» – тег «Задана швидкість перемішування», змінна «Задана швидкість перемішування Т1»:
  - 0-10 об/хв
  - ModbusTCP/IP, Offset=2

- для етапу «Вивантажити» – тег «Заданий нижній рівень», змінна «Заданий нижній рівеньТ1»:
  - 0-100 %
  - ModbusTCP/IP, Offset=4

- На сторінці «Екран Batch» розмістіть Numerical Value для встановлення та зміни значення «Заданий нижній рівеньТ1»; за допомогою елементу Static Text підпишіть поля для обох заданих рівнів (наповнення та нижній).

Спробуйте скопіювати існуючий тег, а потім змінити значення.

2.6 Мінімальний час виконання

Є дві можливості, доступні для того, щоб вплинути на часові налаштування виконання етапу. Одна з них є **minimum duration of execution**. За допомогою цієї властивості етапу, можна означити, як довго етап повинен залишатися ак-

тивним. У нашому випадку це може бути корисним, наприклад, для етапу «Перемішувати». Необхідно означити, що етап повинен бути активним протягом 10 секунд. Далі потрібно виконати наступні дії:

- Відкрийте властивості етапу «Перемішувати».
- Введіть для властивості **minimum duration of execution**, в групі **General** значення 0T00:00:10.
- Скомпілюйте проект, перезавантажте **Runtime** і запустіть рецепт знову.

Тепер етап «Перемішувати» залишається активним протягом 10 секунд до того, як активізується наступний етап. Хронологічна послідовність етап показується в спливаючій підказці етапу.

2.7 Умови

2.7.1 На хронологічну послідовність виконання рецептів можуть вплинути умови. З їх допомогою можна запросити інформацію про стан устаткування і перейти до наступного етапу рецепта в потрібний час. Кожний етап передбачає дві умови, які можуть бути налаштовані в редакторі:

- ***Inputlock (Вхідне блокування*)**;
- ***phase done condition*** (***Умова закінчення етапу***).

Спочатку налаштуйте **phase done condition** для етапу «Наповнити». Цей етап повинен залишатися активним до того часу, поки не буде досягнутий заданий рівень заповнення. Тому необхідно отримувати поточний рівень заповнення.

- Добавте зворотний тег (return tag) «Плинний рівень» до етапу «Наповнити»; створіть нову змінну «Рівень T1» (для Modbus TP/IP, Offset=6).
- Відкрийте властивості **General** етапу «Наповнити» **phase done condition** і натисніть на кнопку «…**».**
- Добавте два параметри і створіть умови, як це показано на Рисунку 2.14.

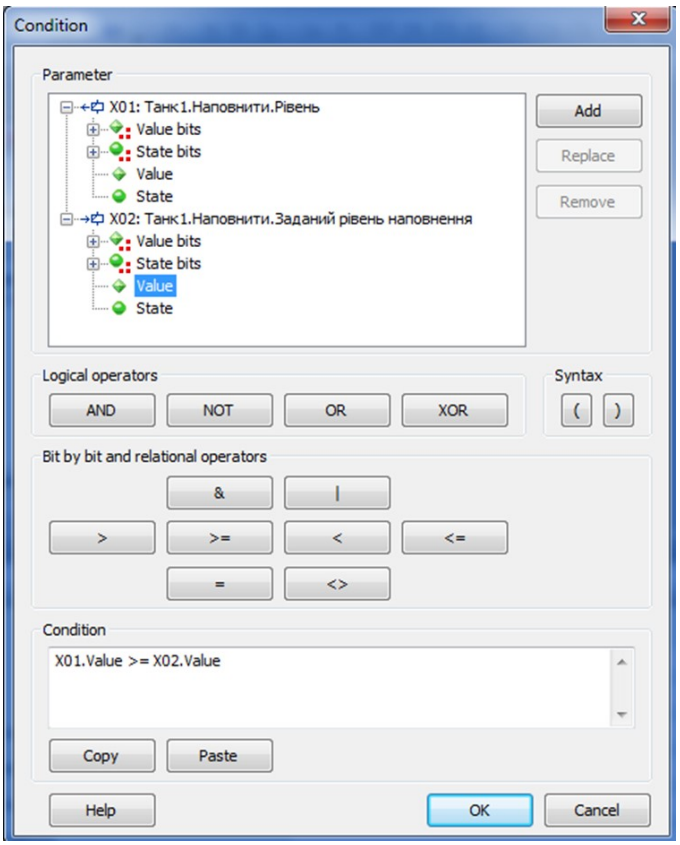


Рисунок 2.14 – Налаштування умови закінчення етапу

- Підтвердьте діалог кнопкою **OK**.
- Налаштуйте такожвідповідні умови завершення етапу для «Вивантажити».
- На сторінці «**Екран Batch**» розмістіть **Numerical Value** для встановлення та зміни значення «Рівень Т1».
- Скомпілюйте файли **Runtime**, виконайте функцію перезавантаження **Runtime** і перезавантажте рецепт.

Етап **«Наповнити»** тепер буде залишатися активним до того часу , поки значення змінної **«РівеньТ1»** не є такою ж або більше, ніж значення **«Заданий рівень наповнення Т1»**. Хронологічна послідовність етапу показаний в спливаючій підказці.

Примітки:

На практиці, перевірка досягнення заданогозначення як правило, не перевіряється в zenon. Замість цього, це робить ПЛК, і таким чиномзавершає етап, передаючи цю інформацію в **zenon**за допомогою параметра стану. Ви можете знайти можливу конфігурацію таких параметрів стану в наступному розділі: **Mnual → Batch Control → Engineering in the Editor → TAGs → Example for status tag**.

2.7.2 Період очікування

Для умов може бути налаштований час очікування. Таким чином, можна реагувати на умови, які не були виконані. Після того, як набір час очікування минув, етап отримує червону рамку, а також червоний значок помилки, який

показує, що очікуваний час був перевищений. Ця інформація також відображається в підказці. Означте **maximum execution duration** 10 секунд для заповнення резервуара. Для цього виконайте наступні дії:

- У **zenon Editor** відкрийте властивості для етапу **«Наповнити»**.
- Введіть значення 0T 00:00:10 для **Maximum execution duration** в групі **General**.
- Скомпілюйте файли **Runtime**, виконайте функцію перезавантаження **Runtime**.
- Перезавантажте рецепт. У режимі тестування перевірте роботу рецепту.
- Збережіть рецепт.

Контрольні питання

1. Яке призначення модуля Batch Control в zenon як називається його виконавча частина? Які спеціальні сторінки (екрани) існують в zenon для керування рецептами?
2. Розкажіть про сновні принципи розділення апарутрного та рецептурного керування згідно ISA-88.
3. Що представляє собою рецепт з точки зору ISA-88?
4. Які типи рецептів Ви знаєте?
5. З яких складових складаються процедури рецептів?
6. Як означується в zenon етпи з яких будуються процедури апаратів?
7. Як етапи в zenon взаємодіють з контролером?
8. Розкажіть про оснонві елементи PFC, які були задіяні в даній лабораторній роботі.
9. Які типи тегів означені для етапів, і для чого вони використовуються?
10. Поясніть відмінність між керівним рецептом та майстер рецептом.
11. Розкажіть про налаштування часових інтервалів виконання етапів.
12. Розкажіть яким чином налаштовується умова завершення етапів.

ЛАБОРАТОРНА РОБОТА 3. ЗВ'ЯЗОК РЕЦЕПТУРНОГО ТА АПАРАТУРНОГО КЕРУВАННЯ

Мета роботи – навчитися створювати взаємозв'язані етапи в SCADA та контролері, створювати реакції та події, означувати стани та команди.

Загальні теоретичні відомості

Етапи в рецептах пов'язані з аналогічними етапами в системі керування (Рисунок 3.1). Однак логіка етапу в системі керування може бути реалізовано різними способами.

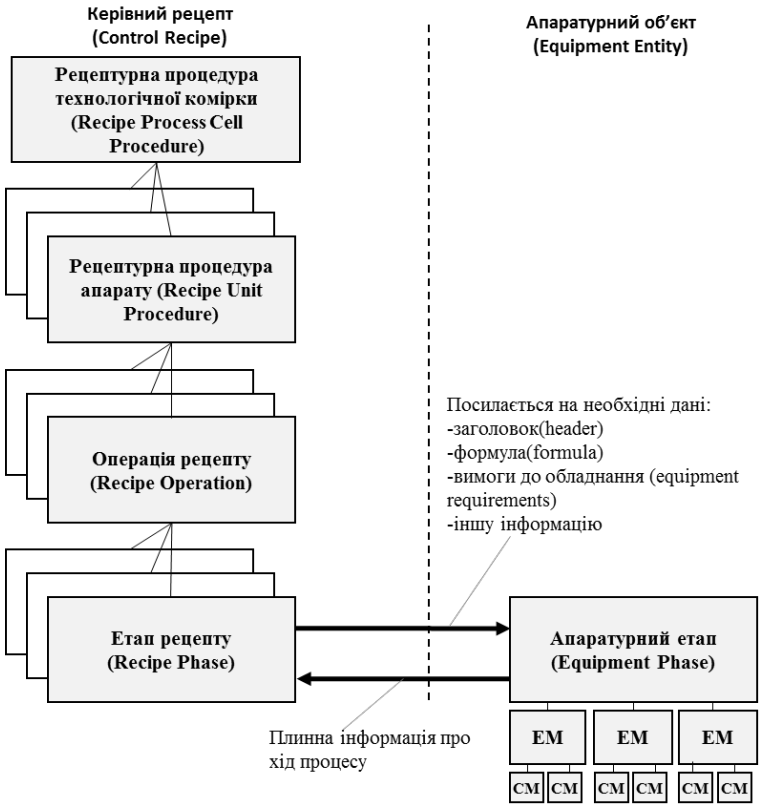


Рисунок 3.1 – Взаємодія етапів в рецепті та системі керування

У SCADA zenon етап може виконувати усю логіку виконання, взаємодіючи з CM на контролері (наприклад виконавчими механізмами та датчиками, що підключені до ПЛК). У більшості ж випадків компонувальні блоки-етапи (phase) в апаратах (Unit), з яких будується рецепт для REE в zenon надають тільки інтерфейс для взаємодії з такими саме етапами в ПЛК (Рисунок 3.2). Тобто етап в zenon при ініціалізації або зміні стану передає на ПЛК тільки задані значення, команди керування, а отримує з нього дійсні значення та стан етапу. Таким чином відбувається синхронізація між етапом в zenon та його реалізацією в ПЛК. Керування самими клапанами відбувається виключно в ПЛК через функції базового керування (base control).

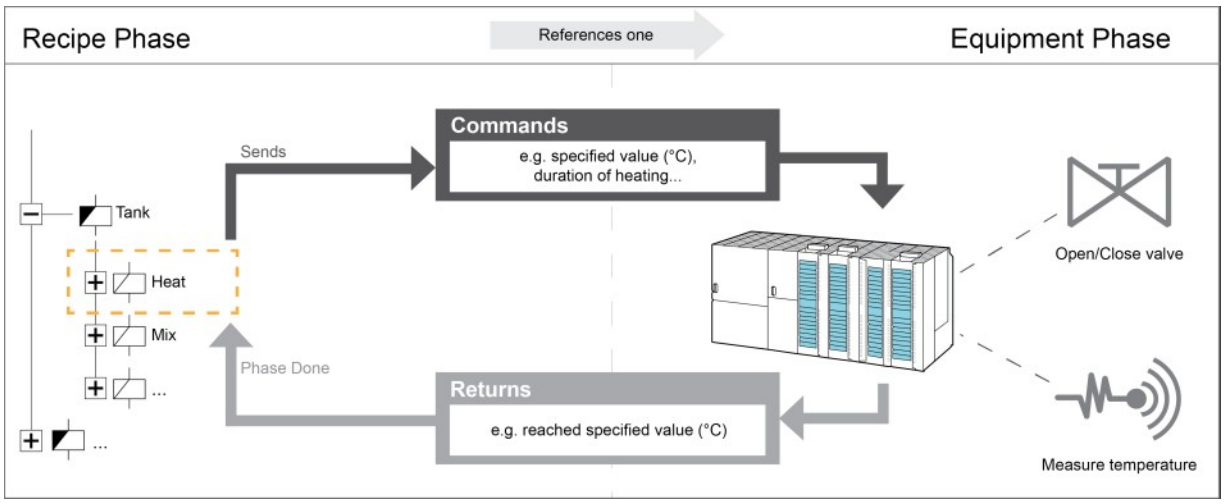


Рисунок 3.2 – Можливості взаємодії етапів в REE SCADA zenon та ПЛК

У даній лабораторній роботі буде використовуватися реалізація логіки виконання етапів тільки в zenon, а ПЛК буде виконувати функції вводу/виводу. Така конструкція дає можливість реалізувати batch-керування навіть використовуючи разом з zenon тільки модулі вводу/виводу. У наступній лабораторній роботі вся логіка виконання етапів буде записуватися в ПЛК.

У ПЛК вже створені змінні, які будуть взаємодіяти з виконавчими механізмами, а на SCADA реалізований з ними зв'язок. Тепер необхідно вписати в етапи командні теги (параметри), які будуть давати команди керування цим виконавчим механізмам.

Кожен процедурний елемент згідно ISA-88 виконується в залежності від режиму (ручний, автоматичний, напівавтоматичний) та *стану (state, status)*, в якому він знаходиться. Режими будуть розглянуті в наступній лабораторній роботі. Стан вказує на особливості виконання етапу (початок, кінець, штатний або нештатний режим і т.д.), тому в кожному з станів може бути означено свій алгоритм виконання. Перехід зі стану в стан відбувається або згідно виконання певних внутрішніх умов (наприклад, виконання умови закінчення етапу) або надсилання процедурному елементу команди.Команда може надсилатися оператором або іншим процедурним елементом (наприклад, процедура апарату надсилає команду етапу). Правила переходу між станами описується автоматом станів. У ISA-88.1 поданий типовий автомат станів, який підходить до більшості технологічний процесів (Рисунок 3.3).

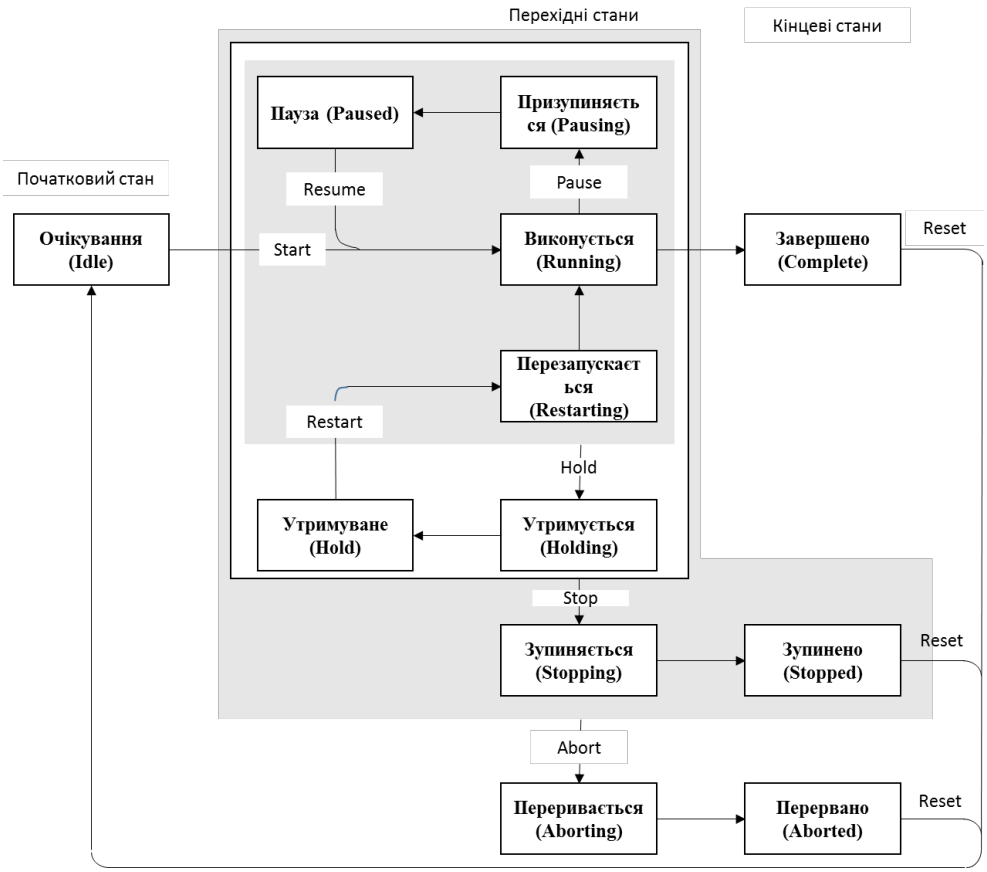


Рисунок 3.3 – Типовий автомат станів етапу

Основні стани:

- **Idle** : простоювання (очікування), не виконання діяльності процедурного керування;
- **Running**: виконується логіка процедури до завершення;
- **Paused**: стоїть в паузі;
- **Complete**: процедура завершена за внутрішньою логікою;
- **Hold**: утримування, довготривала «пауза» (як правило, в нештатних ситуаціях);
- **Stopped**: зупинено за командою;
- **Aborted**: терміново зупинено за командою відміни.

Серед станів є перехідні, назва яких закінчується на «-ing». У перехідних станах відбувається виконання логіки переходу. Навіть «**Running**» можна назвати перехідним між «**Idle**» та «**Complete**».

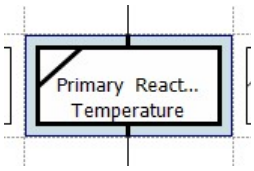
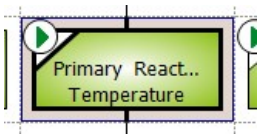
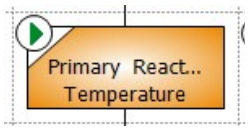
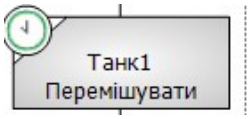
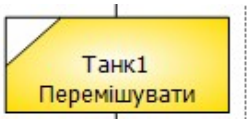
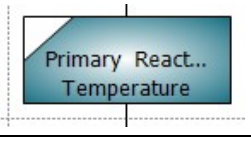
Для інформування ПЛК про *події* (*events*), пов'язані з виконанням рецептів, у Zenon використовуються *реакції* (*reactions*). Реакції можуть бути налаштовані для багатьох подій, які може ініціювати етап або інші частини REE тасередовище виконання zenon взагалі. Реакції можуть впливати на виконання рецепту, вони можуть привести до запису значень параметрів або можуть бути виконані необхідні функції zenon. Далі використаємо події та реакції на них для закривання клапанів в кінці етапів, вимиканню мішалки та для формування повідомлень в журнал CEL.

У середовищі виконання SCADA zenon логіка виконання процедур, що

прописані в рецепті, обробляється в модулі REE. Процедура комірки та апарата керується командами (*Commands*) та умовами переходу PFC, які в свою чергу передають команди етапам (phase). Типові команди в ISA-88 також означені (Рисунок 3.3). Етапи в REE пов'язані з етапами в контролері за допомогою змінних (через теги етапів), у тому числі змінні команд та статусу.

Наприклад, якщо в SCADA zenon при відправці команди «Stop» процедурі апарата, кожному етапу апарата, що зараз виконується, відправляється така сама команда, після якої етапи переходять в стан «Stopping». Для завершення переходу в кінцевий стан «Stop», повинна обробитися логіка переходу (наприклад, в програмі ПЛК), тому команда повинна відправитися етапу в контролері. Після обробки логіки переходу, етап (наприклад в контролері) переходить в стан «Stop», про що повідомляється етапу в SCADA zenon.

При виконанні рецепту в PFC, кожен етап підсвічується кольором в залежності від стану:

Колір	**Сататус:**
 білий	**Idle state** (простоювання)
 зелений	**in execution** (виконується)
 помаранчевий	**Paused** (пауза)
 сірий	**Hold** (утримування)
 жовтий	**Stopped** (зупинено)
червоний 	**Aborted** (перервано)
 світло-синій	**Completed** (завершено)
два кольори	Перехідні стани (**Transient status**)

Також в кутку етапів символами надається додаткова інформація:

Символ	Значення
	Етап запускається
	Очікування виділення апарату (unit). Апарат або етап вже використовується у іншому рецепті
	Очікування закінчення вхідного блокування.
	1. Протягом виконання етапу: очікування реакції «завершення». 2. На переході: очікування виконання умови переходу. 3. У кінці паралельного розгалуження: Очікування завершення виконання усіх гілок .
	При підготовці до старту. Після команди рестарту, коли очікується перехід до стану старту.
	1. Реакція на штатне завершення; очікування реакції completed. 2. При переходах: очікування умови переходу. 3. При виконанні паралельних гілок: очікування завершення послідовності в паралельних гілках за виконання умови.
	Очікування виконання процедури **Minimum execution duration**.
	Очікування ексклюзивного виконання .
	Тайм-аут очікування unit allocation закінчився.
	Тайм-аут очікування вхідного блокування закінчився.
	Надмірно довге очікування запису.
	Закінчився тайм-аут на реакцію completed.
	Закінчився тайм-аут на реакцію ended.

Команди можуть бути викликані оператором, використовуючи іконки в панелі керування, реакцією або через функції zenon. На Рисунку 3.4.а показані команди керування усією процедурою (апарату чи технологічної комірки) означеної рецепті. На Рисунку 3.4.б показані команди керування окремим (виділеним в PFC) етапом.

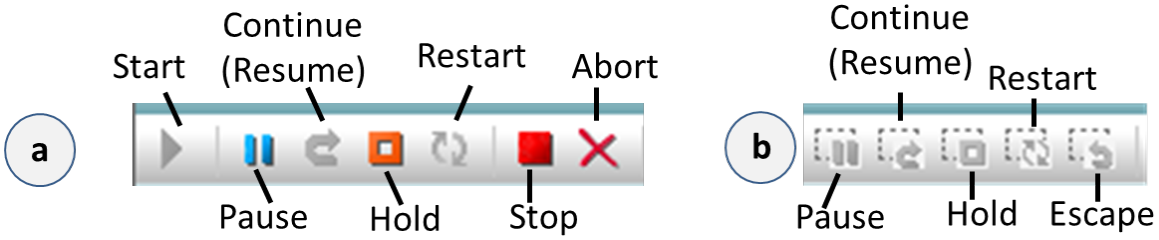


Рисунок 3.4 – Команди керування процедурою (апарату або технологічної комірки) означеної в рецепті (а) та етапі (b).

Для означення логіки, яка буде виконуватися при зміні стану етапу в Zenon, можна скористатися відповідними реакціями етапу на події, зокрема:

- Status change: In execution
- Status change: Pausing
- Status change: Paused
- Status change: Continue
- Status change: Holding
- Status change: Hold
- Status change: Restarting
- Status change: Stopping
- Status change: Stopped
- Status change: Aborting
- Status change: Aborted
- Status change: Executed.

Використайте реакції на зміну стану, для означення поведінки етапу «Наповнити» в станах Паузи та Утримання для закриття клапану набору, та відкриття його по продовженню виконання.

Завдання до виконання лабораторної роботи

1. Створити з'єднання з контролером в Unity Pro.
2. Налаштувати та виконати етапи в SCADA.
3. Розглянути та створити реакції та події.
4. Додати команди та стани.

Порядок проведення роботи

3.1 Створення з'єднання з контролером (імітатором) UNITY PRO

3.1.1 У лабораторній роботі №2 значення змінних ПЛК імітувалися в самій SCADA zenon.

У цій лабораторній роботі необхідно завантажити підготовлений проект Unity Pro з імітатором об'єкту та реалізувати з'єднання змінних SCADA zenon з зовнішнім імітатором ПЛК. Результатом повинно служити правильне відображення змінних на мнемосхемі та можливість зміни заданих значень та керування виконавчими механізмами.

Завантажте проект в ПЛК та перевірте його роботу.

- За необхідності завантажте та встановіть UNITY PRO (>v6.0).
- Образ UNITY PRO 11. Режим доступу: https://schneider-electric.box.com/s/ea0zk09lbh7x30rit7jfbscwndu5pmf3
- У назві образу вказаний серійний номер та номер PV для тріал-версії.
- Запустіть UNITY PRO.
- Відкрийте файл MES2017_BASE.XEF, скомпілюйте проект і завантажте його на виконання.
- Подивіться перелік змінних в проекті (Рисунок 3.5).

VNabor_T2_OPN	EBOOL	%i0.1.1		клапан набору Т2 відкритий
VNabor_T1_CLS	EBOOL	%i0.1.4		клапан набору Т1 закритий
VNabor_T2_CLS	EBOOL	%i0.1.5		клапан набору Т2 закритий
VSliv_T1_CLS	EBOOL	%i0.1.6		клапан зливу Т1 закритий
VSliv_T2_CLS	EBOOL	%i0.1.7		клапан зливу Т2 закритий
VNabor_D1_CLS	EBOOL	%i0.1.8		клапан набору D1 закритий
VNabor_D2_CLS	EBOOL	%i0.1.9		клапан набору D2 закритий
VSliv_D1_CLS	EBOOL	%i0.1.10		клапан зливу D1 закритий
VSliv_D2_CLS	EBOOL	%i0.1.11		клапан зливу D2 закритий
LSH_D1	EBOOL	%i0.1.12		сигналізатор верхнього рівня D1
LSL_D1	EBOOL	%i0.1.13		сигналізатор нижнього рівня D1
LSH_D2	EBOOL	%i0.1.14		сигналізатор верхнього рівня D2
LSL_D2	EBOOL	%i0.1.15		сигналізатор нижнього рівня D2
VDoz_T1_OPN	EBOOL	%I0.3.0		3-ходовий клапан в позиції Т1
VDoz_T2_OPN	EBOOL	%I0.3.1		3-ходовий клапан в позиції Т2
LE_T1	INT	%iw0.2.0		рівень Т1 (0-10000)
LE_T2	INT	%iw0.2.1		рівень Т2 (0-10000)
TE_T1	INT	%iw0.2.2		тмпература в Т1 (0-10000)
TE_T2	INT	%iw0.2.3		тмпература в Т2 (0-10000)
VNabor_T1	EBOOL	%Q0.1.17		клапан набору Т1
VSliv_T1	EBOOL	%Q0.1.18		клапан зливу Т1
VNabor_T2	EBOOL	%Q0.1.19		клапан набору Т2
VSliv_T2	EBOOL	%Q0.1.20		клапан зливу Т2
VNabor_D1	EBOOL	%Q0.1.21		клапан набору D1
VNabor_D2	EBOOL	%Q0.1.22		клапан набору D2
VSliv_D1	EBOOL	%Q0.1.23		клапан зливу D1
VSliv_D2	EBOOL	%Q0.1.24		клапан зливу D2
VDoz_T1toT2	EBOOL	%Q0.1.25		3-ходовий клапан (0 - на Т1, 1 - на Т2)
VNagrev_T1	INT	%Qw0.2.4		клапан нагрівання Т1 (0-10000)
VNagrev_T2	INT	%qw0.2.5		клапан нагрівання Т2 (0-10000)
SC1	INT	%QW0.4.4		задана швидкість перемішування Т1 (0-10000)
SC2	INT	%QW0.4.5		задана швидкість перемішування Т2 (0-10000)
KM1	EBOOL	%q0.1.26		Пускач двигуна мішалки Т1
KM2	EBOOL	%q0.1.27		Пускач двигуна мішалки Т2
GSM1	EBOOL	%i0.3.2		Контакт пускача двигуна мішалки Т2
GSM2	EBOOL	%i0.3.3		Контакт пускача двигуна мішалки Т1

Рисунок 3.5 – Перелік змінних демо-проекту UNITY PRO

- Відкрийте операторський екран «Tanks» (Рисунок 3.6), використовуючи кнопки керування:
  - наповніть танки рідиною;
  - задайте значення швидкості перемішування, що дорівнює 5000 одиниць контролера;
  - наповніть дозатори повністю;
  - подайте по одній дозі кожного компоненту в танки Т1 та Т2;
  - нагрійте рідину в танках до 4000 од. контролера;
  - злийте рідину з танків і закрийте усі клапани.

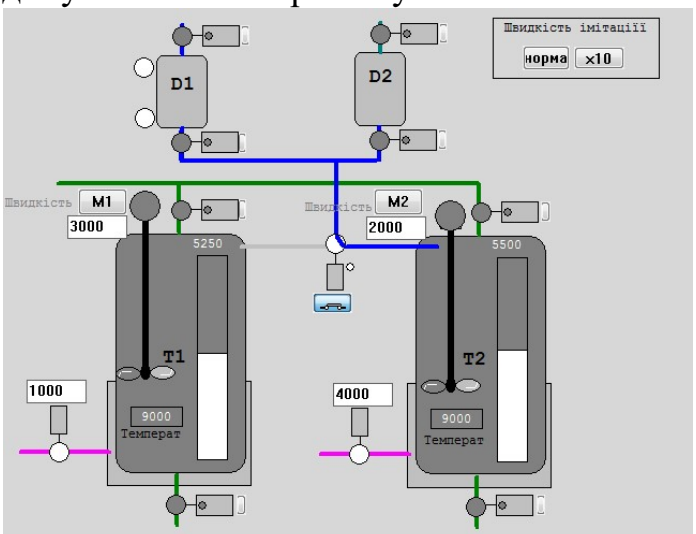


Рисунок 3.6 – Операторський екран «Tanks»

3.1.2 Створіть змінні для масштабованих значень аналогових датчиків та завдань, як це показано на Рисунок 3.7 (коментарі можна не заповнювати).

LT1	REAL	%MW6		рівень в T1 (масштабований 0-100%)
LT2	REAL	%MW8		рівень в T2 (масштабований 0-100%)
LT1_SPHI	REAL	%MW0		заданий рівень наповнення T1
LT2_SPHI	REAL	%MW22		заданий рівень наповнення T2
TT1	REAL	%MW10		температура в T1 (0-100 C)
TT2	REAL	%MW12		температура в T2 (0-100 C)
TT1_SP	REAL	%MW14		задана температура T1
TT_2SP	REAL	%MW16		задана температура T2
LT2_SPLO	REAL	%MW18		заданий нижній рівень T2
LT1_SPLO	REAL	%MW4		заданий нижній рівень T1
SC1_SP	REAL	%MW24		задана швидкість перемішування в T1 (масштабована 0-10 об/хв)
SC2_SP	REAL	%MW20		задана швидкість перемішування в T2 (масштабована 0-10 об/хв)

Рисунок 3.7 – Перелік змінних проекту

3.1.3 Створіть секції масштабування, як це показано на Рисунку 3.8, скомпілюйте код та завантажте в імітатор ПЛК.

Для зручності нижче дається готовий код, який можна скопіювати:

```
(*масштабування вхідна секція*)
TT1:=int_to_real(TE_T1)*0.01; (*0-10000 -> 0-100 C*)
TT2:=int_to_real(TE_T2)*0.01; (*0-10000 -> 0-100 C*)
LT1:=int_to_real (LE_T1) *0.01; (*0-10000 -> 0-100%*)
LT2:=int_to_real (LE_T2) *0.01; (*0-10000 -> 0-100%*)

(*масштабування вихідна секція*)
SC1:=real_to_int (SC1_SP*1000.0); (*0-10 -> 0-10000*)
SC2:=real_to_int (SC2_SP*1000.0); (*0-10 -> 0-10000*)
```

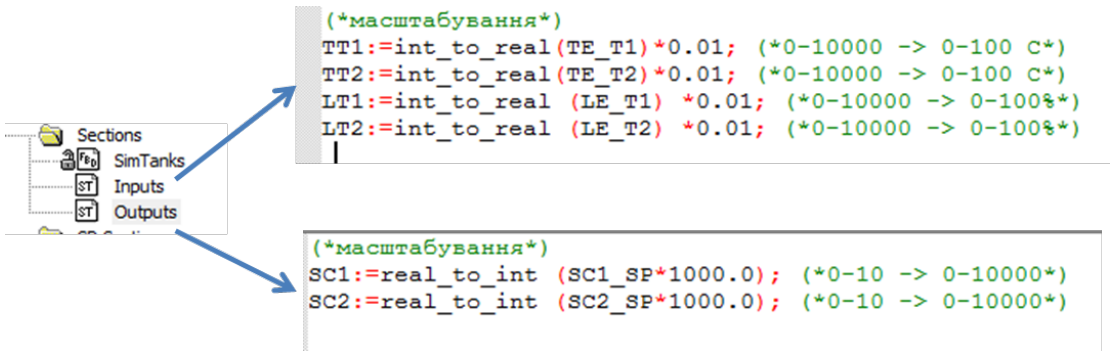


Рисунок 3.8 – Секції масштабування

3.1.4 Налаштуйте в редакторі zenon драйвер Modbus TCP/IP та перевірте його роботу зі зв'язкою з імітатором UNITY PRO.

- Запустіть на виконання zenon editor.
- Змініть налаштування драйверу «Modbus RTU and Open Modbus TCP/IP», як це показано на Рисунок 3.9.

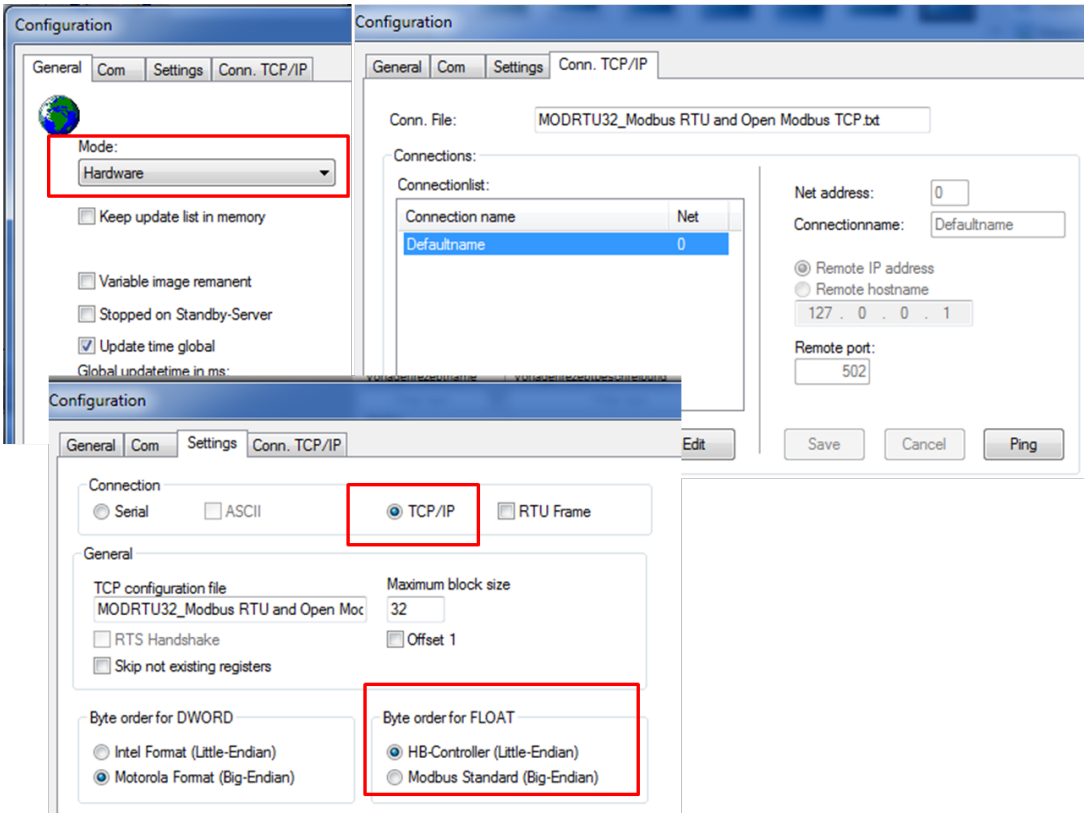


Рисунок 3.9 – Налаштування драйверу Modbus TCP/IP

- Скомпілюйте проект та запустіть на виконання.
- Перейдіть на сторінку «Екран Batch» і впевніться, що значення змінних відображаються вірно відповідно до ПЛК. Для перевірки можна набрати в «Танк Т1» рідину до певного рівня і подивитися на його значення на мнемосхемі zenon.

3.1.5 У проекті UNITY PRO створіть внутрішні змінні, що відповідають за всі дискретні виходи ПЛК (Рисунок 3.10).

- Методом копіювання-вставки скопіюйте усі змінні, що прив'язані до %Q.
- Перейменуйте змінні, давши їм суфікс М.
- Змініть адреси відповідно до Рисунку 3.10.

VNabor_T1	EBOOL	%Q0.1.17
VSliv_T1	EBOOL	%Q0.1.18
VNabor_T2	EBOOL	%Q0.1.19
VSliv_T2	EBOOL	%Q0.1.20
VNabor_D1	EBOOL	%Q0.1.21
VNabor_D2	EBOOL	%Q0.1.22
VSliv_D1	EBOOL	%Q0.1.23
VSliv_D2	EBOOL	%Q0.1.24
VDoz_T1toT2	EBOOL	%Q0.1.25
KM1	EBOOL	%q0.1.26
KM2	EBOOL	%q0.1.27

VDoz_T1toT2_M	EBOOL	%M100		3-ходовий клапан (0 - на Т1, 1 - на Т2)
VSliv_D2_M	EBOOL	%M101		клапан зливу D2
VSliv_D1_M	EBOOL	%M102		клапан зливу D1
VNabor_D2_M	EBOOL	%M103		клапан набору D2
VNabor_D1_M	EBOOL	%M104		клапан набору D1
VSliv_T2_M	EBOOL	%M105		клапан зливу T2
VNabor_T2_M	EBOOL	%M106		клапан набору T2
VSliv_T1_M	EBOOL	%M107		клапан зливу T1
VNabor_T1_M	EBOOL	%M108		клапан набору T1
KM1_M	EBOOL	%M109		Пускач двигуна мішалки T1
KM2_M	EBOOL	%M110		Пускач двигуна мішалки T2

Рис.3.10. Створення внутрішніх «дублікатів» дискретних вихідних змінних

3.1.6 У проекті Unity Pro в секції Outputs опишіть код, який буде переприсвоювати значення внутрішніх дискретних змінних вихідним (Рисунок 3.11).

```
VDoz_T1toT2 := VDoz_T1toT2_M ;
VSliv_D2 :=  VSliv_D2_M;
VSliv_D1 :=  VSliv_D1_M;
VNabor_D2 := VNabor_D2_M;
VNabor_D1 := VNabor_D1_M;
VSliv_T2 :=  VSliv_T2_M;
VNabor_T2 := VNabor_T2_M;
VSliv_T1 := VSliv_T1_M;
VNabor_T1 := VNabor_T1_M;
KM1 := KM1_M;
KM2:=  KM2_M;
```

```
(*масштабування виходів*)
SC1:=real_to_int (SC1_SP*1000.0); (*0-10 -> 0-10000*)
SC2:=real_to_int (SC2_SP*1000.0); (*0-10 -> 0-10000*)

VDoz_T1toT2 := VDoz_T1toT2_M ;
VSliv_D2 :=  VSliv_D2_M;
VSliv_D1 :=  VSliv_D1_M;
VNabor_D2 := VNabor_D2_M;
VNabor_D1 := VNabor_D1_M;
VSliv_T2 :=  VSliv_T2_M;
VNabor_T2 := VNabor_T2_M;
VSliv_T1 := VSliv_T1_M;
VNabor_T1 := VNabor_T1_M;
KM1 := KM1_M;
```

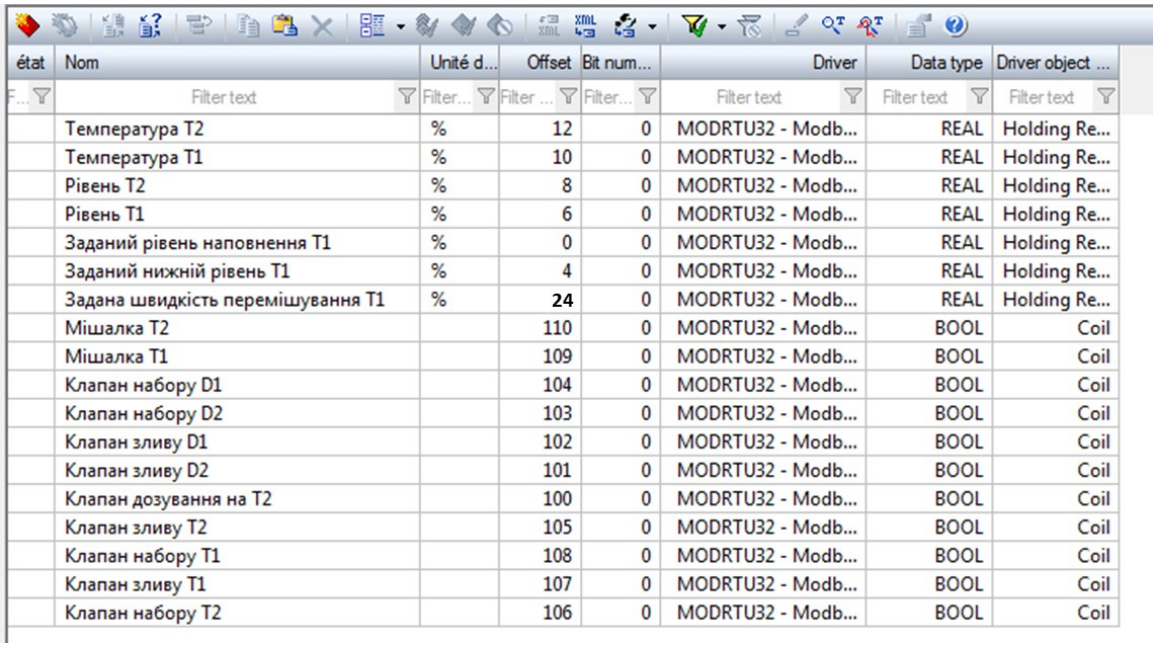
Рисунок 3.11 – Секція Outputs після вставки коду переприсвоєння дискретних змінних

3.1.7 У zenon добавте усі змінні, що відповідають за вихідні змінні ПЛК та аналогові вхідні (Рисунок 3.12).

- Створіть вихідні змінні: (приклад створення дискретних змінних показаний на Рисунку 3.13).

Name	Offset
Клапан дозування на Т2	100
Клапан зливу D2	101
Клапан зливу D1	102
Клапан набору D2	103
Клапан набору D1	104
Клапан зливу Т2	105
Клапан набору Т2	106
Клапан зливу Т1	107
Клапан набору Т1	108
Мішалка Т1	109
Мішалка Т2	110

- Створіть аналогові змінні для рівнів в танках та температур (Рисунок 3.8), задайте масштаби 0-100.



état	Nom	Unité d...	Offset	Bit num...	Driver	Data type	Driver object ...
	Температура T2	%	12	0	MODRTU32 - Modb...	REAL	Holding Re...
	Температура T1	%	10	0	MODRTU32 - Modb...	REAL	Holding Re...
	Рівень T2	%	8	0	MODRTU32 - Modb...	REAL	Holding Re...
	Рівень T1	%	6	0	MODRTU32 - Modb...	REAL	Holding Re...
	Заданий рівень наповнення T1	%	0	0	MODRTU32 - Modb...	REAL	Holding Re...
	Заданий нижній рівень T1	%	4	0	MODRTU32 - Modb...	REAL	Holding Re...
	Задана швидкість перемішування T1	%	24	0	MODRTU32 - Modb...	REAL	Holding Re...
	Мішалка T2		110	0	MODRTU32 - Modb...	BOOL	Coil
	Мішалка T1		109	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан набору D1		104	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан набору D2		103	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан зливу D1		102	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан зливу D2		101	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан дозування на T2		100	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан зливу T2		105	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан набору T1		108	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан зливу T1		107	0	MODRTU32 - Modb...	BOOL	Coil
	Клапан набору T2		106	0	MODRTU32 - Modb...	BOOL	Coil

Рисунок 3.12 – Змінні в zenon

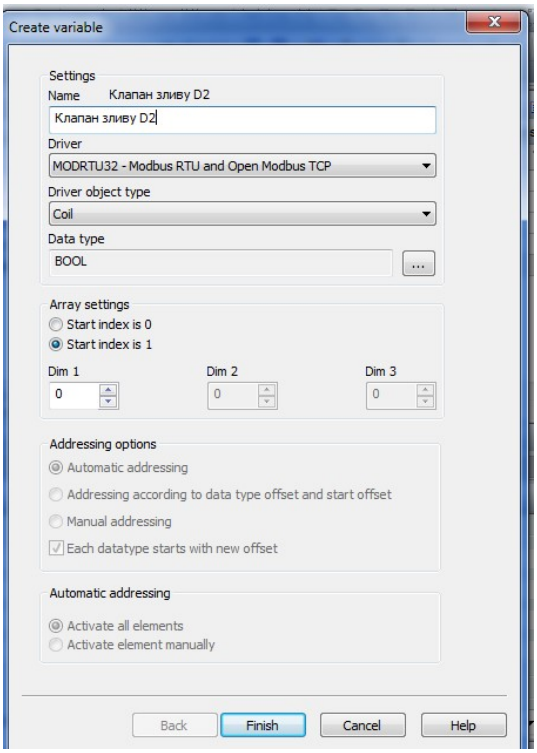


Рисунок 3.13 – Створення дискретної змінної в zenon

3.1.8 Аналогічно пунктам 3.1.5 – 3.1.7 додайте змінні, що відповідають сигналам від дискретних датчиків.

- Створіть внутрішні змінні в ПЛК.
- Зробіть переприсвоєння вхідним в секції INPUTS.
- Створіть змінні в SCADA zenon.

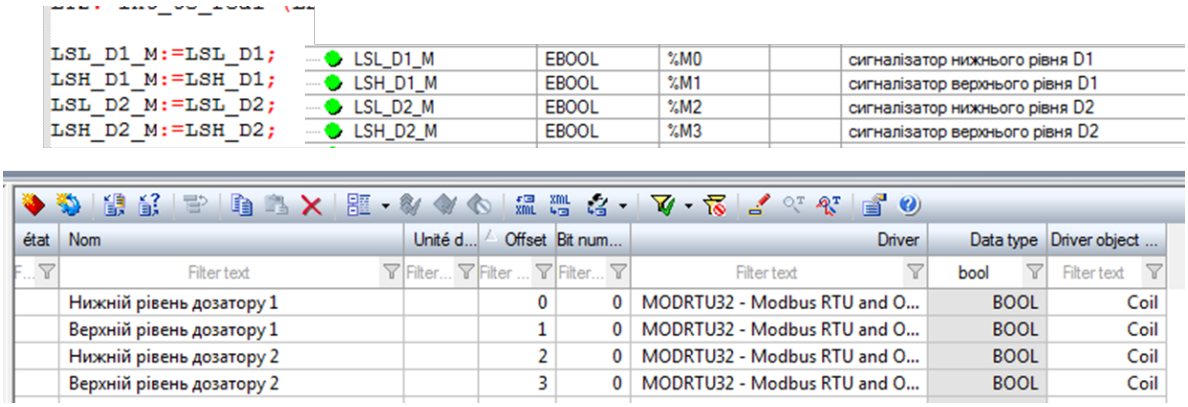


Рисунок 3.14 – Створення дискретних змінних, що відповідають за сигналізатори рівнів

3.1.9 У zenon на екрані «Batch» створіть мнемосхему керування установкою (Рисунок 3.15). Використовуйте елементи для швидкого створення, менше звертайте увагу на ергономічність. Перевірте роботу мнемосхеми в режимі виконання.

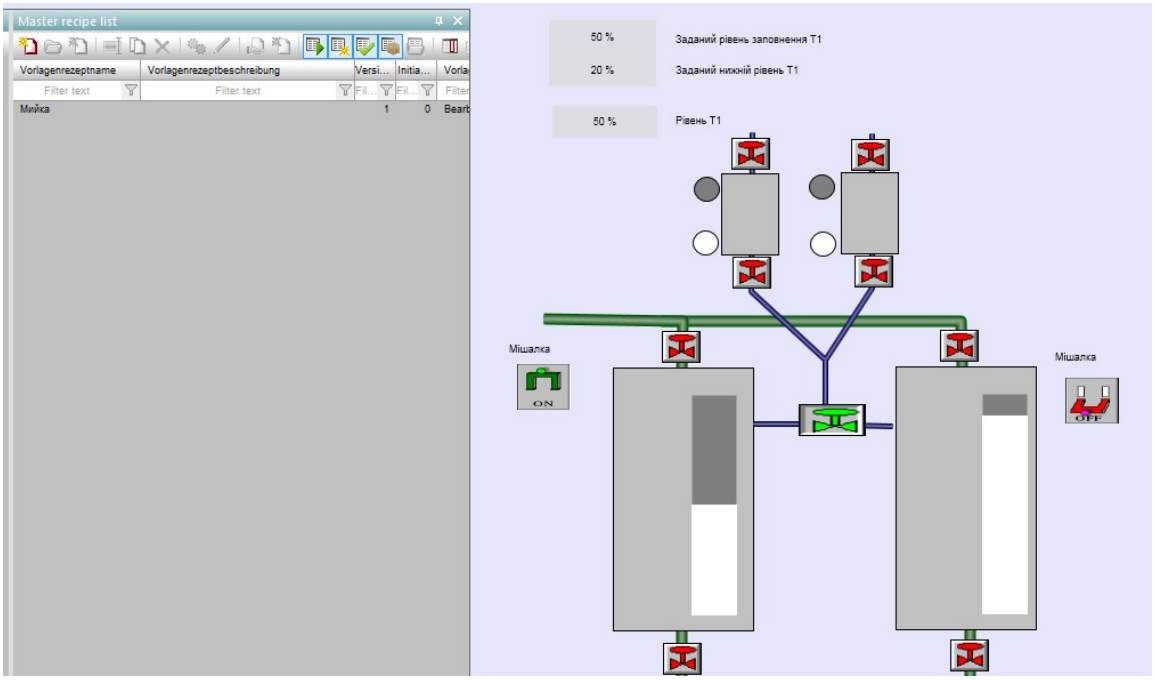


Рисунок 3.15 – Приклад чорнового варіанту загального вигляду мнемосхеми керування приготуванням продукту

3.2 Виконання етапів в SCADA

3.2.1 У zenon **batch control** для етапу «Наповнити» створіть змінний тег (**Valuetag**) з назвою «Клапан набору» (Рисунок 3.16), який буде відкриватися (on) при ініціюванні етапу і закриватися (off) при його деактивації:

- В етапі «Наповнити» створіть новий тег (параметр) з типом «**Value**» та іменем «Клапан набору».
- У групі властивостей «**General**» у властивості **Variable**, прив’яжіть тег до змінної «Клапан набору T1».

- У групі властивостей «**Write Set Value**» у властивості **Tag Value** вкажіть значення «On(1)», для того, щоб при ініціалізації етапу туди записалося значення 1.

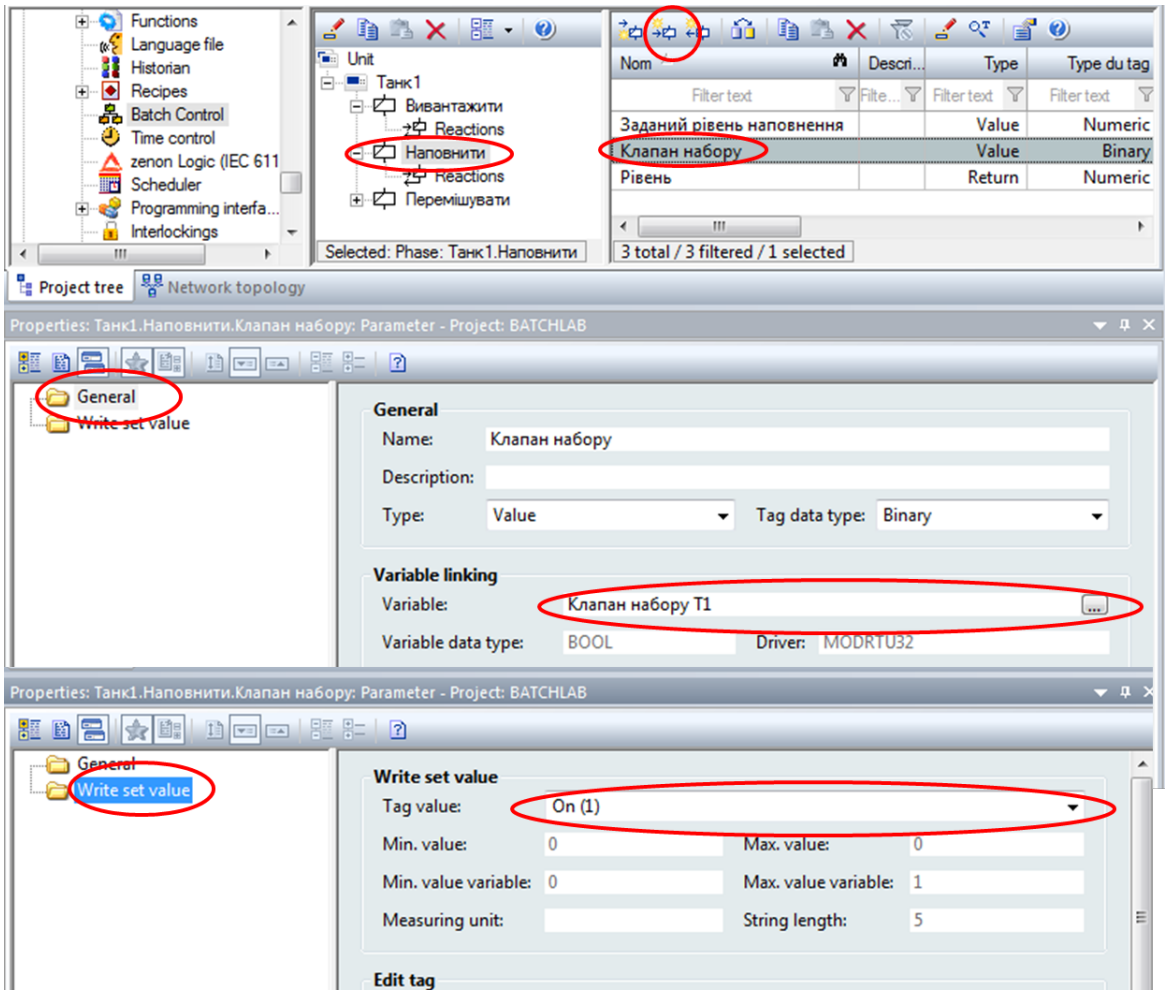


Рисунок 3.16 – Створення та налаштування тегу «Клапан набору» для етапу «Наповнити»

3.2.2 Аналогічно попередньому пункту створіть та налаштуйте:

- Тег «Клапан зливу» в етапі «Вивантажити» (Рисунок 3.17).
- Тег «Двигун мішалки» в етапі «Перемішувати» .

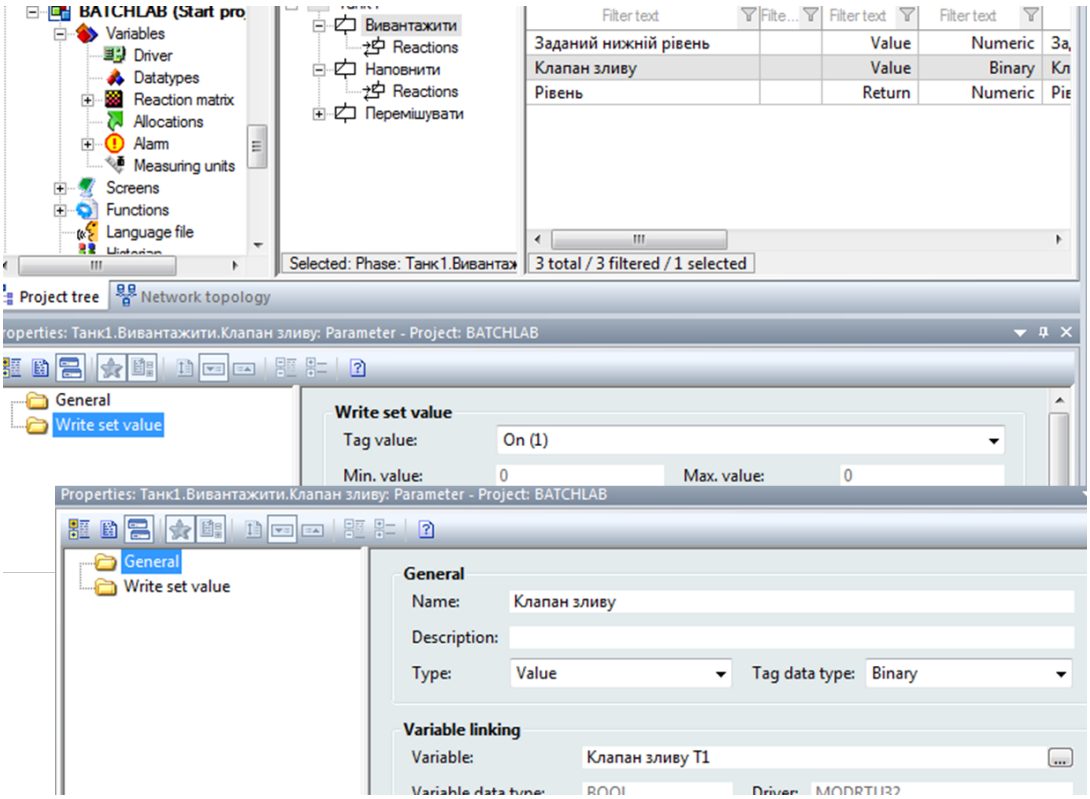


Рисунок 3.17 – Створення та налаштування тегу «Клапан зливу» для етапу «Вивантажити»

Тепер при запуску етапу «Наповнити» змінній «Клапан набору Т1» буде присвоєне значення 1, а при запуску етапу «Перемішування» – одиниця запишеться в змінну «Мішалка Т1», при запуску «Вивантажити» – одиниця – в змінну «Клапан зливу Т1». Однак, згідно логіки виконання, клапани по завершенню етапів повинні закриватися, а мішалка відключатися. За умови виконання логіки керування етапів тільки в SCADA, ці змінні повинні змінитися за певною подією. Для цього необхідно створити події та прописати для них відповідні реакції.

3.3 Події та реакції (Events and Reactions)

3.3.1 Створіть реакції на подію деактивації етапів для закривання клапанів набору та зливу танку після завершення етапу, а також перевірте їх роботу.

- Для етапу «Вивантажити» створіть реакцію на подію «**Phase deactivated**», в якій пропишіть закривання клапану зливу (Рисунок 3.18):

- для цього розверніть вузол етапу «Вивантажити» в дереві апарату (unit) так щоб вузол «**Reactions**» став видимий;

- натисніть на Reactions для відображення всіх сконфігурованих реакцій в списку;

- створіть нову реакцію натиснувши на іконці **New reaction** в панелі інструментів;

- виберіть в колонці **Event Phase deactivated**;

- введіть в поле **Parameter** значення «Танк1. Вивантажити. Клапан

зливу», а в поле **Value Boolean** значення Off(0).

- Аналогічно зробіть для клапану набору в події «**Phase deactivated**» етапу «Наповнити» та для мішалки в етапі «Перемішування».
- Запустіть на виконання **zenon Runtime**;
- Відкрийте екран «**Batch**» та завантажте рецепт «Мийка», що був створений в лабораторній роботі №2 та переведіть його в режим тестування.
- Запустіть рецепт на виконання. «Танк1» повинен заповнитися продуктом, перемішуватися і вивантажуватися після паузи.

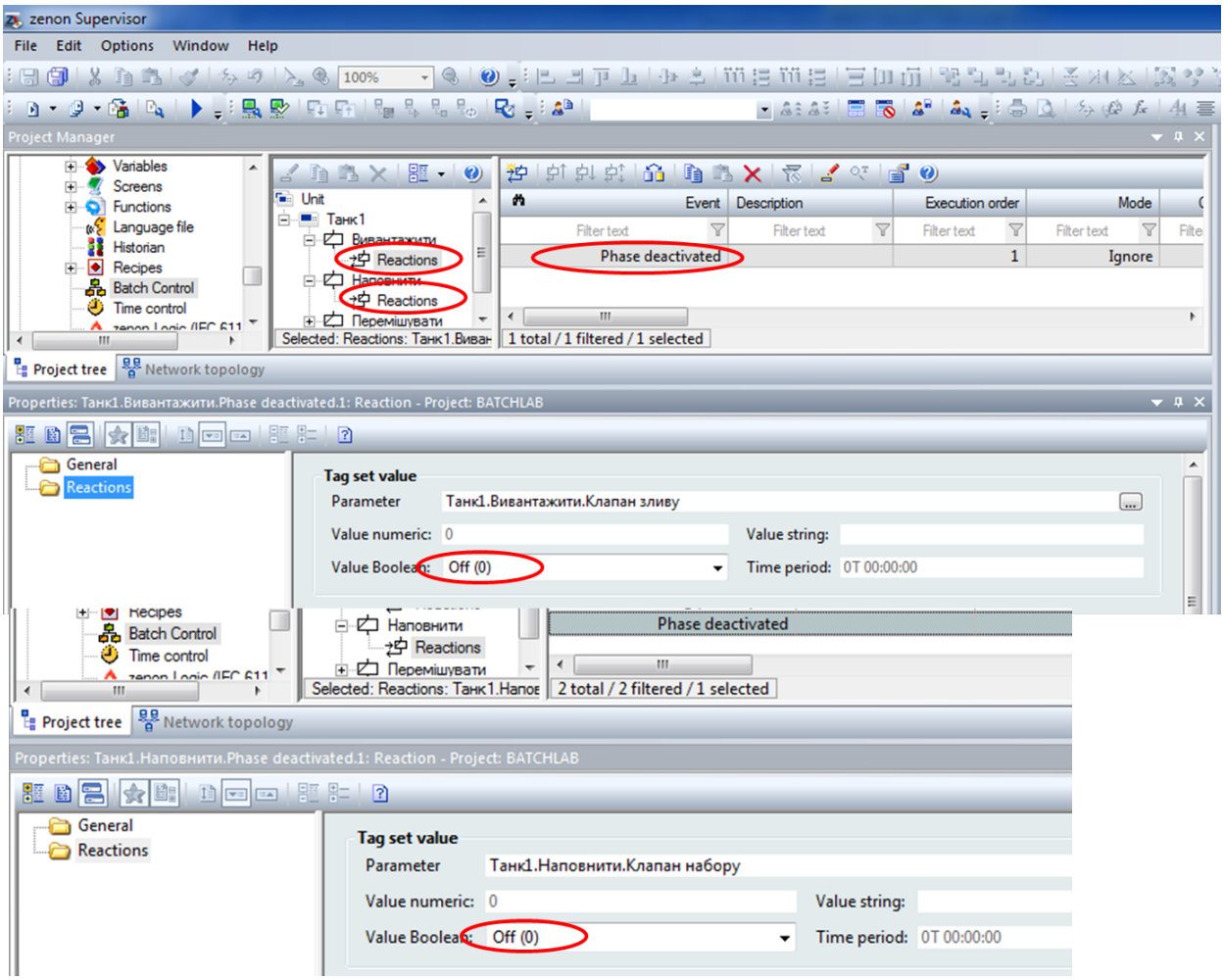


Рисунок 3.18 – Створення реакцій на події

3.3.2 Створіть реакціюна подію не заповнення Танку 1 протягом 15 секунд, яка буде створювати відповідний запис в **CEL**.

- Скопіюйте основний фрейм. Створіть новий екран «Екран **CEL**» на основі новоствореного шаблону та типу «**Chronological Event List**».

Порада. Той самий шаблон, що для екрану **Batch Control**, не рекомендується використовувати, інакше завжди прийдеться відкривати рецепти кожного разу після переключення на екран **CEL** з екрану **Batch Control**. Розмістіть елементи керування на екрані за замовченням.

- Створіть функцію для переключення екрану на **CEL** в налаштуваннях фільтру і залиште все за замовченням. Прив'яжіть функцію до кнопки,

яку розмістіть на екрані меню.

- Наступним кроком є означення **maximum execution duration**. Відкрийте властивості етапу «Наповнити» і введіть **maximum execution duration** рівним 10 секунд, якщо це ще не було зроблено до цього часу.
- Для створення запису **CEL entry**, потрібно сконфігурувати реакцію на подію **Maximum waiting period phase done condition exceeded** (Рисунок 3.19):

- розверніть вузол етапу «Наповнити»в дереві апарату (unit) так щоб вузол «**Reactions**» став видимий;

- натисніть на **Reactions** для відображення всіх сконфігурованих реакцій в списку;

- створіть нову реакцію, натиснувши на іконці New reaction в панелі інструментів;

- виберіть в колонці **Event Maximum waiting period phase done condition exceeded**;

- активуйте опцію **Create CEL** entry, і введіть в поле **CEL message text** текст «Танк1 не наповнився за коректний час»;

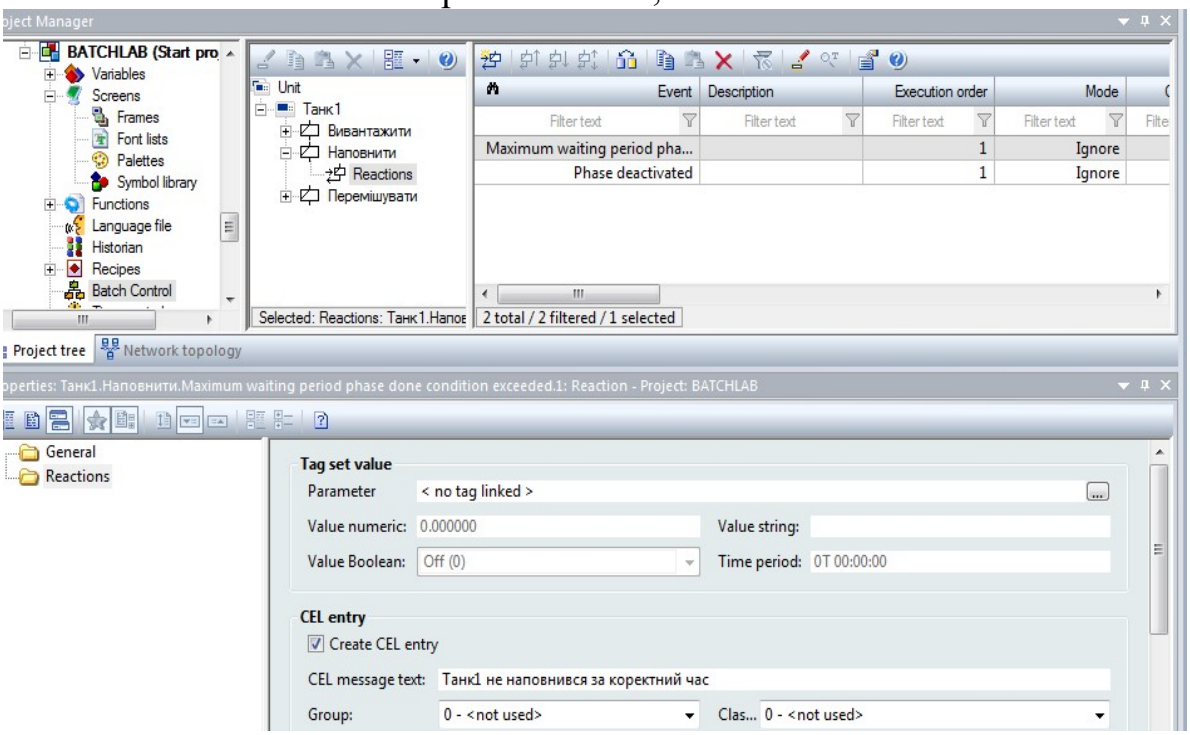


Рисунок 3.19 – Налаштування реакції

- вручну відкрийте клапан зливу Танку 1, запустіть рецепт на виконання. Оскільки наповнення танку не буде відбуватися, то після 10 секунд по запуску, буде зроблений запис в повідомлення CEL.

- відкрийте сторінку CEL і знайдіть запис.

3.4 Команди та стани (Commands and Status)

3.4.1 Для етапу «Наповнити» створіть реакції для закриття клапану набору при переході в стан «**Holding**» та «**Pausing**» а також відкриття його при вико-

нанні (Рисунок 3.20):

- Створіть реакції на події «**Status change: Pausing**» та «**Status change: Holding**», при яких клапан набору закривається.
- Створіть реакцію на подію «**Status change: In execution**», при яких клапан набору відкривається.
- Запустіть середовище виконання, запустіть рецепт в режимі тесту та перевірте його роботу на етапі «Наповнення» в станах «**Pause**» та «**Hold**», давши відповідні команди (Рисунок 3.20).
- Дайте відповідні команди для продовження виконання рецепту.

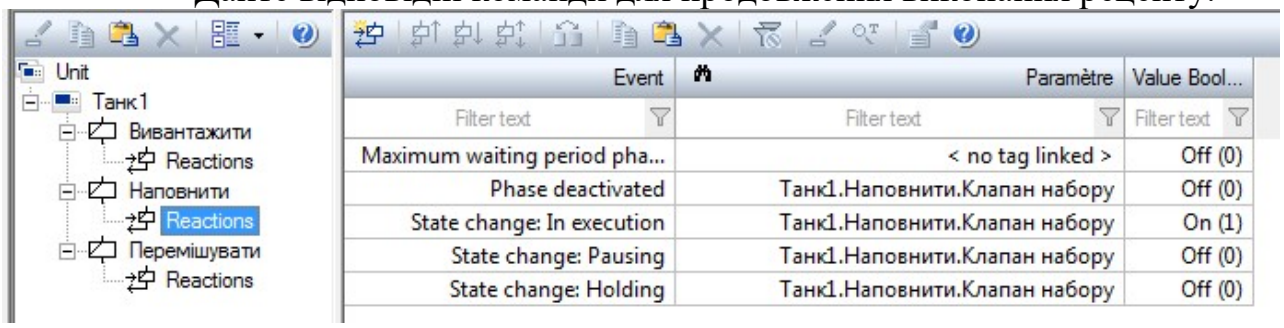


Рисунок 3.20 – Реакції для етапу «Наповнити»

Для етапу «Вивантажити» окрім керування клапаном зливу, в режимі утримування (**Holding**) необхідно також включити мішалку, щоб продукт зберігав свою консистенцію при тривалому зберіганні.

3.4.2 Для етапу «Вивантажити» при переході в стан «**Pausing**» пропишіть реакції закриття клапану зливу, в стан «**Holding**» – закриття клапану, та включення мішалки, при робочому стані відкриття клапану зливу та відключення мішалки (Рисунок 3.21):

- Для етапу «Вивантажити» створіть тег «Мішалка» та прив'яжіть його до змінної «Мішалка Т1».
- Створіть реакції на події «**Status change: Pausing**» та «**Status change: Holding**» при яких клапан зливу закривається.
- Створіть ще одну реакцію на подію «**Status change: Holding**» , при якій включається мішалка.
- Створіть дві реакції на подію «**Status change: In execution**», одна відкриває клапан набору, друга відключає мішалку.
- Запустіть середовище виконання, запустіть рецепт в режимі тесту та перевірте його роботу на етапі «Вивантаження» в станах «**Pause**»та «**Held**», давши відповідні команди (Рисунок 3.21).
- Перевірте роботу команд для окремо виділених етапів.

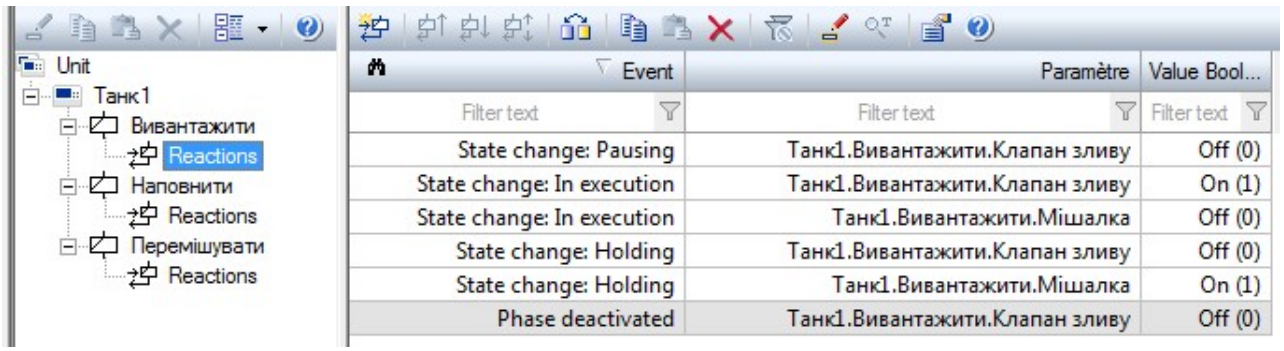


Рисунок 3.21 – Реакції для етапу «Вивантажити»

Контрольні питання

1. Опишіть взаємодію етапів в рецепті та системі керування. Як відбувається їх синхронізація?
2. Розкажіть про автомат станів процедурних елементів: призначення, переходи між станами?
3. Розкажіть про призначення проміжних та кніцевихстанів.
4. Для чого в Batch Cotrol використовуються реакції і на що вони можуть впливати? Де в лабораторній роботі це було використано?
5. При виконанні рецепту в PFC, кожен етап підсвічується кольором в залежності від стану. Використовуючи таблицю, прокоментуйте, що означають той чи інший колір та символ.
6. Які реакції етапу на події Ви знаєте, щоб описати логіку, яка буде виконуватися при зміні стану етапу в zenon? Де в лабораторній роботі це було використано?
7. Розкажіть про команди керування виконанням рецептом та етапами.

ЛАБОРАТОРНА РОБОТА 4. ВИКОНАННЯ ПРОЦЕДУР НА РІВНІ SCADA ТА ПЛК. МОЖЛИВОСТІ РЕЦЕПТІВ В PFC

Мета роботи – навчитися використовувати процедури на рівні SCADA та ПЛК. Розглянути можливості рецептів в PFC.

Загальні теоретичні відомості

У минулій лабораторній роботі вся логіка виконання етапу була реалізована в SCADA, а на ПЛК виконувались тільки базові функції реалізації керування та збір даних. Така реалізація має багато обмежень і, як правило, не застосовується. У цій лабораторній роботі використовується інший принцип, за яким логіка виконання етапів реалізована в ПЛК. Таким чином, етапи будуть функціонувати як в SCADAтак і в ПЛК, зв'язуючись через команди та стани.

Для початку, необхідно переробити вже існуючий функціонал для реалізації етапів «Наповнити», «Перемішувати» і «Вивантажити» в контролері. Для обміну між SCADA та PLC статусною інформацією про режими та стани та командами синхронізації необхідно використати структурну змінну з 3-ма полями CMD (команда), STA (статус, стан) iMODE (режим). Поле CMD (команда) – буде направляти до PLCкоманду зміни стану процедурного елементу після зміни її в REE(в SCADA), кажучи про необхідність певних дій в тому ж елементі в PLC. Таким чином, процедурні елементи (наприклад етапи) в REE будуть переходити в проміжні стани (закінчуються на –ing: Stopping, Pausing, Holding, Aborting, Restarting) і повідомляти про це ПЛК, шляхом зміни значення CMD. Етапи в ПЛК в цих проміжних станах будуть виконувати певну логіку переходу, після чого переводити їх в усталені стани (Stopped, Paused, Held, Aborted, Executed). Стан етапів в ПЛК буде передаватися через поле STAзмінної. Нижче в таблиці 4.1 наведена відповідність значення змінної STAстану етапу:

Таблиця 4.1

STA (Status)	Значення	Примітка
Idle	0	Очікування (початковий стан)
Running	1	Виконується
Complete (Executed)	2	Виконано, ініціюється в ПЛК.
Stopping	3	Зупиняється, ініціюється в SCADA.
Stopped	4	Зупинено, ініціюється в ПЛК.
Pausing	5	Призупиняється, ініціюється в SCADA.
Paused	6	Пауза, ініціюється в ПЛК.
Holding	7	Утримується, ініціюється в SCADA.
Held	8	Утримуване, ініціюється в ПЛК.
Aborting	9	Переривається, ініціюється в SCADA.
Aborted	10	Перервано, ініціюється в ПЛК.
Restarting	11	Перезапускається, ініціюється в SCADA.

Змінна **MODE** буде вказувати на режим виконання процедурного елементу (таблиця 4.2).

Таблиця 4.2

Mode	Значення	Примітка
Ignore	0	не використовується
Automatic	1	Автоматичний
Semi-automatic	2	Напівавтоматичний
Manual	3	Ручний

Для відправки команд в ПЛК, необхідно визначитися з їх значеннями. Команди будуть відправлятися в реакціях на події етапів, що пов'язані зі зміною стану або режиму. В таблиці 4.3 наведені команди та їх числові значення, які прийнято використовувати в zenon.

Таблиця 4.3

CMD (Command) –реакція на подію	Значення	Примітка
Phase started	1	Етап запущено на виконання
Finished writing command tags	2	Завершено запис значення слова команди
Phase finished: Phase done condition fulfilled and Minimum execution duration reached (if engineered)	3	Завершення виконання етапу
Phase deactivated	4	Деактивація етапу
Status change: Pausing	10	Зміна стану: Призупиняється
Status change: Resuming	11	Зміна стану: Вихід з паузи
Status change: Holding	12	Зміна стану: Утримується
Status change: Restarting	13	Зміна стану: Перезапускається
Status change: Stopping	14	Зміна стану: Зупиняється
Status change: Aborting	15	Зміна стану: Переривається
Mode change: Automatical	20	Зміна режиму: Автоматичний
Mode change: Semi-automatic	21	Зміна режиму: Напівавтоматичний
Mode change: Manual	22	Зміна режиму:Ручний
Exit Runtime initiated	30	Ініціювання виходу з середовища виконанняZenon
Runtime restart	31	Перезапуск середовища виконання Zenon
Unit allocation not possible	32	Неможливо виділити апарат (або інший ресурс обладнання)
Waiting period unit allocation exceeded	33	Перевищено період очікування виділення апарату
Input interlocking blocked	34	Умова входу заблокована
Waiting period input interlocking exceeded	35	Перевищено період очікування завершення вхідного блокування
Maximum execution period exceeded	36	Перевищено максимальний період виконання
Waiting period following condition exceeded	37	Перевищено максимальний період виконання умови переходу
Phase started multiple times	38	Етап запущено на виконання декілька раз

З попередніх лабораторних робіт відомі деякі рецептурні елементи:

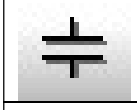
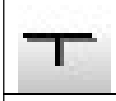
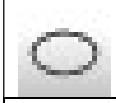
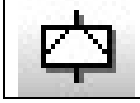
- Початковий елемент позначає початок рецепту; він повинен з'яв-

лятися в рецепті рівно один раз.

- Кінцевий елемент показує кінець рецепту; він повинен з'являтися в рецепті рівно один раз.
- Етапи є основними компонентами рецепту.
- Лінії з'єднання служать для зв'язку двох елементів.

Для того щоб мати можливість реалізувати складні рецепти в PFC є додаткові елементи:

Таблиця 4.4 – Елементи PFC

	**Transition** (Перехід): містить вільно означувані умови, що базуються на параметрах, які означені в етапах рецепту.
	**Begin simulation sequence** (Початок послідовностей одночасного виконання або паралельне розгалуження): виконання починається з цього елемента на декілька послідовностей одночасно.
	**End simulatians sequence** (Кінець послідовностей одночасного виконання або паралельне сходження): тут об'єднуються послідовності одночасного виконання.
	**Begin sequence selection** (Початок вибору послідовностей або альтернативне розгалуження): виконання продовжується в одній із двох послідовностей.
	**End sequence selection** (Кінець вибраної послідовності): кілька альтернативних шляхів з'єднуються знову
	**Jump target** (Перехід на ціль): комбіновані альтернативні шляхи; можуть також використовуватися з петлями (loops).
	**Unital location** (Виділення апарату): резервування або вивільнення апарату.
	**Operation** (Операція): Рецепт в рецепті.

Для перехідних станів можуть бути означені умови. Це створює можливість для очікування реакції від ПЛК.

Якщо команда для рецепту встановлена в **pause** (призупинитися), усі активні етапи переключаються в **pause** (призупиняється) і запускаються на обробку події. ПЛК може бути інформований про заплановану зміну стану за допомогою реакцій. ПЛК тільки повідомляє про **pause** для центру керування REE, якщо він реально перейшов в **paused** (призупинений). Цей зворотний зв'язок може бути означений через умову для **paused**. Якщо ця умова була виконана, етап перемикається з **pause** на **paused**.

Розглянемо режими для виконання рецептів.

Доступні три режими для виконання рецептів:

- **Automatical** (Автоматичний);
- **Semi-automatic** (Напівавтоматичний);
- **Manual** (Ручний).

У **автоматичному режимі (Automatical)** () рецепт виконується незалежно без будь якого впливу оператора. Однак для перевірки рецепту може знадобитися виконання етапу декілька раз, виконання вибраних етапів в довільній послідовності. У цьому випадку використовується напівавтоматичний режим.

Напівавтоматичний режим (SEMI-AUTOMATIC MODE)  . Після виконання рецептурного елементу, позиція виконання в рецепті показується стрілкою (вказівником), активні етапи рецепту призупиняються (**paused**). Тепер оператор може активувати потрібний елемент і продовжити рецепт у вказаному місці. Якщо активних елементів декілька (при одночасних послідовностях), позиція може бути переміщена і продовжена незалежно від інших. Для цього доступні іконки» **Continue recipe at all execution positions» (**  **)**та «**Continue recipe at only on selected execution positions» (**  **)**.

Ручний режим (MANUAL MODE)  .Додатково до функцій напівавтоматичного режиму, в ручному режимі можуть бути відмінені умови переходу «Skip active condition» () та доступна команда «Escape from phase» (), яка завершує виконання етапу.

Режим може бути вибраний використовуванням іконки з панелі інструментів, а також через реакції та функції zenon.

Завдання до виконання лабораторної роботи

1. У ПЛК створити структурні типи **Phase INOUT** та відповідні змінні для кожного етапу, для обміну статусом, командами та режимами між SCADA та PLC.
2. Те саме зробити в SCADA zenon.
3. Створити похідні функціональні блоки (типи та екземпляри) для реалізації етапів в ПЛК.
4. Видалити з етапів SCADA теги, що відповідають за виконавчі механізми (клапан набору, зливу, мішалка) та реакцій, які з ними пов’язані, так як керування ними буде відбуватися в ПЛК.
5. Створити теги (параметри) STA, CMD та MODE для етапів в SCADA та прив’язати їх до відповідних змінних.
6. Для кожного етапу SCADA означити умови переходу з перехідних станів в усталені стани, по значенню змінної STA в ПЛК.
7. Для кожного етапу SCADA означити реакції на події змін стану для відправки команд на ПЛК

Порядок проведення роботи

4.1 Виконання етапів на рівні SCADA та ПЛК

4.1.1 У ПЛК створіть структурні типи Phase INOUT та відповідні змінні для кожного етапу, для обміну статусом, командами та режимами між SCADA та PLC.

- Завантажте проект з попередньої лабораторної роботи в Unity Pro.
- Створіть структурний тип (DDT) PhaseINOUT з полями STA, CMD, MODE типу INT (Рисунок 4.1).

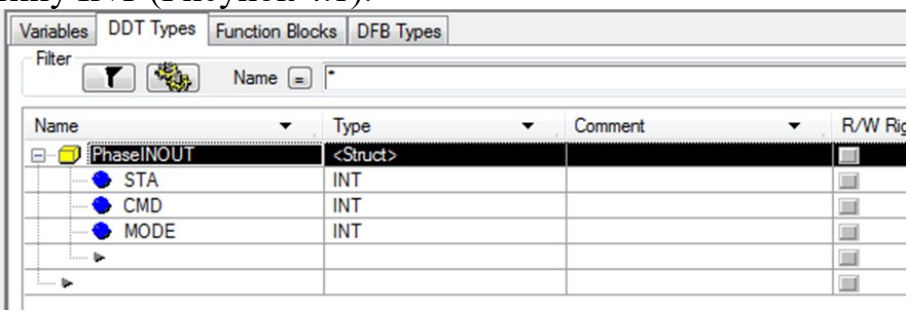


Рисунок 4.1 – Створення структурного типу Phase INOUT в Unity Pro

- Для кожного етапу створіть змінні типу Phase INOUT: phioDRAIN, phioFILL та phioMIX прив’яжіть змінні до адрес відповідно %MW100, %MW104 та %MW108 (Рисунок 4.2).

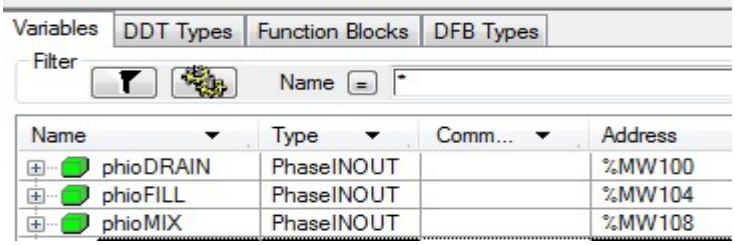


Рисунок 4.2 – Створення змінних для етапів в Unity Pro

- Скомпілюйте проект, якщо помилок немає – переходьте до наступного пункту.

4.1.2 У SCADA zenon створіть структурні типи **Phase IN OUT** та відповідні змінні для кожного етапу, для обміну статусом, командами та режимами між SCADA та PLC.

- Завантажте zenon та відкрийте проект з попередньої лабораторної роботи
- Створіть новий структурний тип (structure data type) **Phase IN OUT** з полями **STA, CMD, MODE** типу INT (Рисунок 4.3).

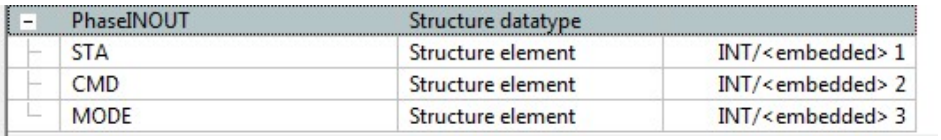
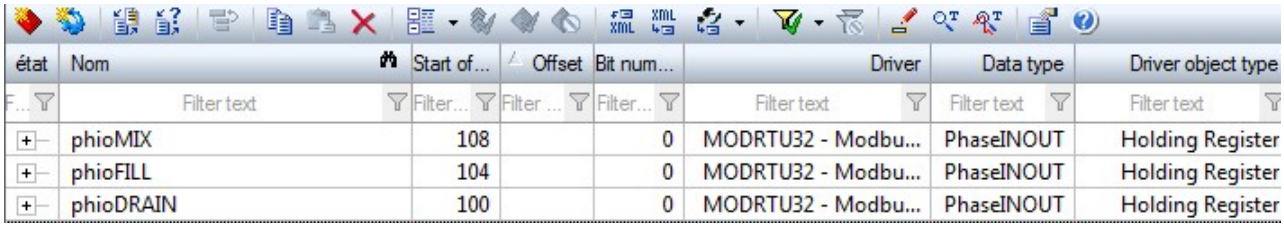


Рисунок 4.3 – Створення структурноготипу в Zenon

- Для кожного етапу створіть змінні типу **Phase IN OUT**: **phioDRAIN**, **phioFILL** та **phioMIX** прив’яжіть змінні до адрес відповідно %MW100, %MW104 та %MW108 (Рисунок 4.4).



état	Nom	Start of...	Offset	Bit num...	Driver	Data type	Driver object type
	Filter text	Filter...	Filter ...	Filter...	Filter text	Filter text	Filter text
+	phioMIX	108		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
+	phioFILL	104		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
+	phioDRAIN	100		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register

Рисунок 4.4 – Створення структурних змінних в zenon

4.1.3 Створіть похідні функціональні блоки (типи та екземпляриDFB) для реалізації етапів в ПЛК.

- В UNITY PROстворіть **DFB**-тип **phDRAIN** (Рисунок 4.5); текст коду можна скопіювати з таблиці 4.4;

DFB включає три параметри типу **IN OUT**:

- **phIO** – інтерфейсна змінна між SCADA та PLC;
- **VLV** – для керування клапаном зливу;
- **MIX** – для керування приводом мішалки.

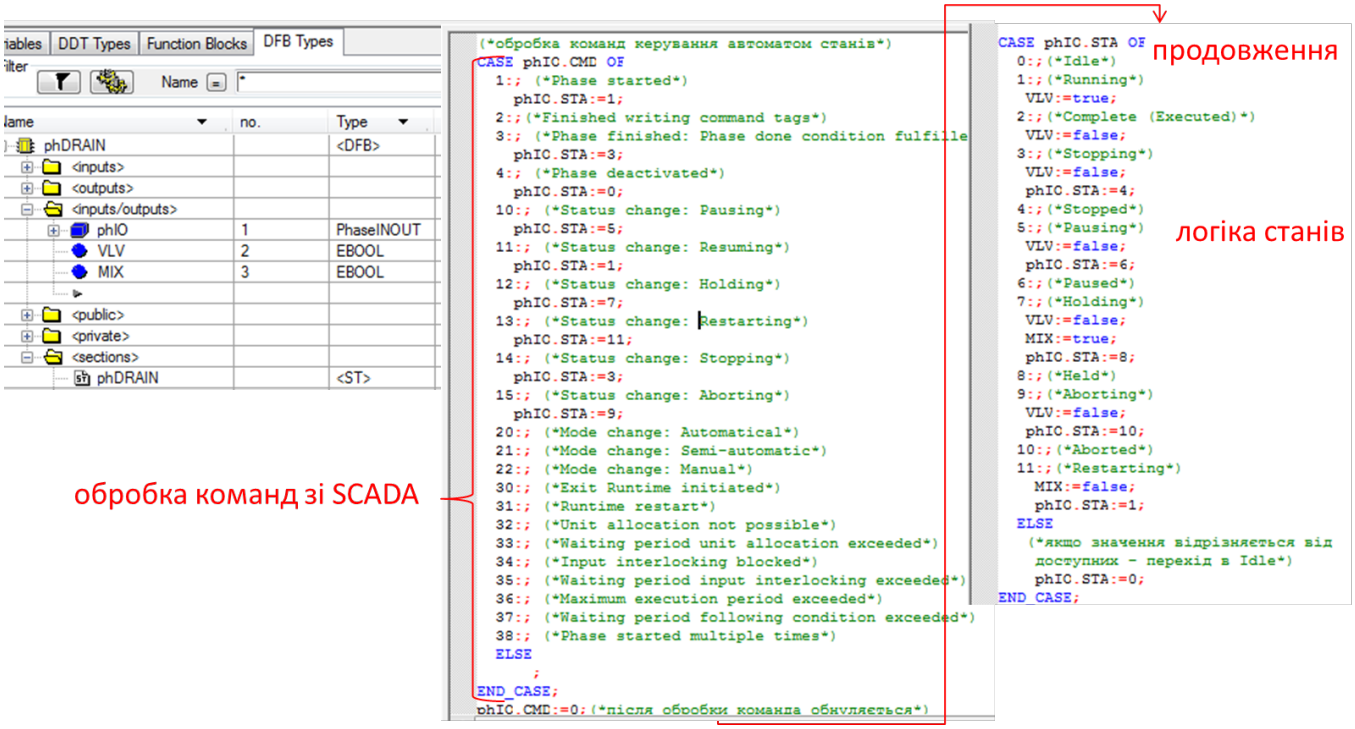


Рисунок 4.5 – Створення **DFB**-типу **phDRAIN**

Вибір типу **IN OUT** диктується міркуваннями можливості зміни цих змінних як ззовні функціонального блоку так і всередині нього.

Перша частина коду функціонального блоку вміщує реалізацію керування автоматом станів в залежності від команди, що надходить зі SCADA. Числове значення для кожної команди наведено в таблиці 4.3. Обробку деяких команд поки не будемо реалізовувати. У кінці обробника команди обнуляються, щоб повторно не обробитися на наступному циклі.

Друга частина коду функціонального блоку реалізує логіку станів, тобто дії, які буде робити етап з ВМ у кожному зі станів. Перелік станів та відповідних числових значень дано в таблиці 4.1. У логіці станів прописані тільки дії для

забезпечення такого ж функціоналу, що був в лабораторній роботі №3.

Таблиця 4.5 – Код для етапу **phDRAIN**

```
(*обробка команд керування автоматом станів етапу phDRAIN*)
CASE phIO.CMD OF
  1:; (*Phase started*)
    phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3:; (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4:; (*Phase deactivated*)
    phIO.STA:=0;
  10:; (*Status change: Pausing*)
    phIO.STA:=5;
  11:; (*Status change: Resuming*)
    phIO.STA:=1;
  12:; (*Status change: Holding*)
    phIO.STA:=7;
  13:; (*Status change: Restarting*)
    phIO.STA:=11;
  14:; (*Status change: Stopping*)
    phIO.STA:=3;
  15:; (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
  36:; (*Maximum execution period exceeded*)
  37:; (*Waiting period following condition exceeded*)
  38:; (*Phase started multiple times*)
  ELSE
      ;
END_CASE;
phIO.CMD:=0;(*після обробки команда обнуляється*)

CASE phIO.STA OF
  0:;(*Idle*)
  1:;(*Running*)
   VLV:=true;
  2:;(*Complete (Executed)*)
   VLV:=false;
  3:;(*Stopping*)
   VLV:=false;
   phIO.STA:=4;
  4:;(*Stopped*)
  5:;(*Pausing*)
   VLV:=false;
   phIO.STA:=6;
  6:;(*Paused*)
  7:;(*Holding*)
   VLV:=false;
   MIX:=true;
   phIO.STA:=8;
  8:;(*Held*)
  9:;(*Aborting*)
   VLV:=false;
```

```
    phIO.STA:=10;
  10:;(*Aborted*)
  11:;(*Restarting*)
    MIX:=false;
    phIO.STA:=1;
  ELSE
   (*якщо значення відрізняється від
    доступних - перехід в Idle*)
    phIO.STA:=0;
END_CASE;
```

- Аналогічно створіть DFB-типи **phFILL** та **phMIX** (Рисунок 4.6); текст коду можна скопіювати відповідно з таблиці 4.6 та 4.7;

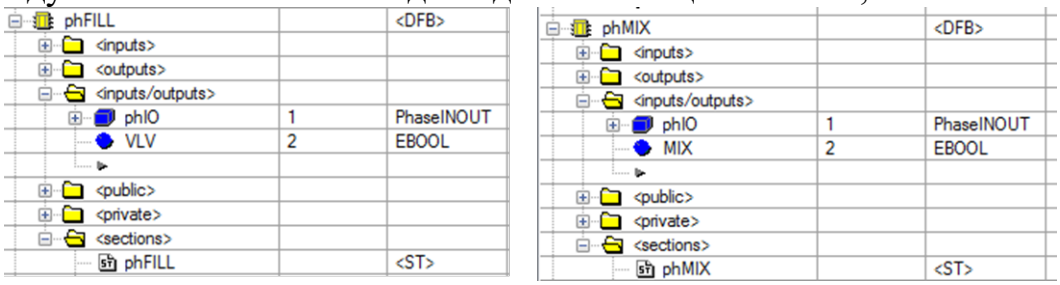


Рисунок 4.6 – Створення DFB-типівphFILL та phMIX

Таблиця 4.6 – Код для етапу **phFILL**

```
(*обробка команд керування автоматом станів етапу phFill*)
CASE phIO.CMD OF
  1:; (*Phase started*)
    phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3:; (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4:; (*Phase deactivated*)
    phIO.STA:=0;
  10:; (*Status change: Pausing*)
    phIO.STA:=5;
  11:; (*Status change: Resuming*)
    phIO.STA:=1;
  12:; (*Status change: Holding*)
    phIO.STA:=7;
  13:; (*Status change: Restarting*)
    phIO.STA:=11;
  14:; (*Status change: Stopping*)
    phIO.STA:=3;
  15:; (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
  36:; (*Maximum execution period exceeded*)
  37:; (*Waiting period following condition exceeded*)
  38:; (*Phase started multiple times*)
  ELSE
      ;
```

```
END_CASE;
phIO.CMD:=0;(*після обробки команда обнуляється*)

CASE phIO.STA OF
  0:;(*Idle*)
  1:;(*Running*)
   VLV:=true;
  2:;(*Complete (Executed)*)
   VLV:=false;
  3:;(*Stopping*)
   VLV:=false;
   phIO.STA:=4;
  4:;(*Stopped*)
  5:;(*Pausing*)
   VLV:=false;
   phIO.STA:=6;
  6:;(*Paused*)
  7:;(*Holding*)
   VLV:=false;
   phIO.STA:=8;
  8:;(*Held*)
  9:;(*Aborting*)
   VLV:=false;
   phIO.STA:=10;
  10:;(*Aborted*)
  11:;(*Restarting*)
    phIO.STA:=1;
  ELSE
     phIO.STA:=0;  (*якщо значення відрізняється від доступних - перехід в
Idle*)
END_CASE;
```

Таблиця 4.7 – Код для етапу **phMIX**

```
(*обробка команд керування автоматом станів етапу phFill*)
CASE phIO.CMD OF
  1:; (*Phase started*)
    phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3:; (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4:; (*Phase deactivated*)
    phIO.STA:=0;
  10:; (*Status change: Pausing*)
    phIO.STA:=5;
  11:; (*Status change: Resuming*)
    phIO.STA:=1;
  12:; (*Status change: Holding*)
    phIO.STA:=7;
  13:; (*Status change: Restarting*)
    phIO.STA:=11;
  14:; (*Status change: Stopping*)
    phIO.STA:=3;
  15:; (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
```

```
   36:; (*Maximum execution period exceeded*)
   37:; (*Waiting period following condition exceeded*)
   38:; (*Phase started multiple times*)
   ELSE
       ;
 END_CASE;
 phIO.CMD:=0;(*після обробки команда обнуляється*)

 CASE phIO.STA OF
   0:;(*Idle*)
   1:;(*Running*)
    MIX:=true;
   2:;(*Complete (Executed)*)
    MIX:=false;
   3:;(*Stopping*)
    MIX:=false;
    phIO.STA:=4;
   4:;(*Stopped*)
   5:;(*Pausing*)
    MIX:=false;
    phIO.STA:=6;
   6:;(*Paused*)
   7:;(*Holding*)
    MIX:=true;
    phIO.STA:=8;
   8:;(*Held*)
   9:;(*Aborting*)
    MIX:=false;
    phIO.STA:=10;
   10:;(*Aborted*)
   11:;(*Restarting*)
     phIO.STA:=1;
   ELSE
      phIO.STA:=0;  (*якщо значення відрізняється від доступних - перехід в
 Idle*)
 END_CASE;
```

- Зробіть аналіз проекту для перевірки на помилки.
- Створіть екземпляри функціональних блоків для 1-го танку (Рисунок 4.7).

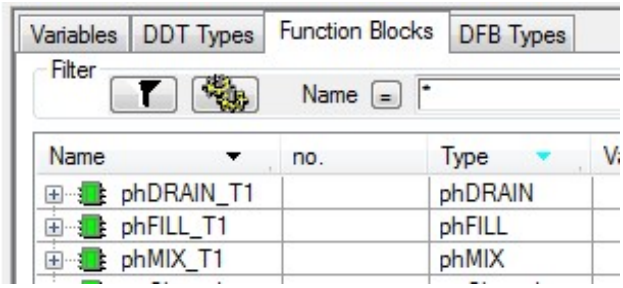


Рисунок 4.7 – Екземпляри функціональних блоків для виконання етапів для 1-го танку

- Створіть секцію Tank1 в якій реалізуйте виклик екземплярів блоків для 1-го танку (Рисунок 4.8).

Зверніть увагу, що етапи викликаються завжди безумовно, що передбачається ідеологією ISA-88.

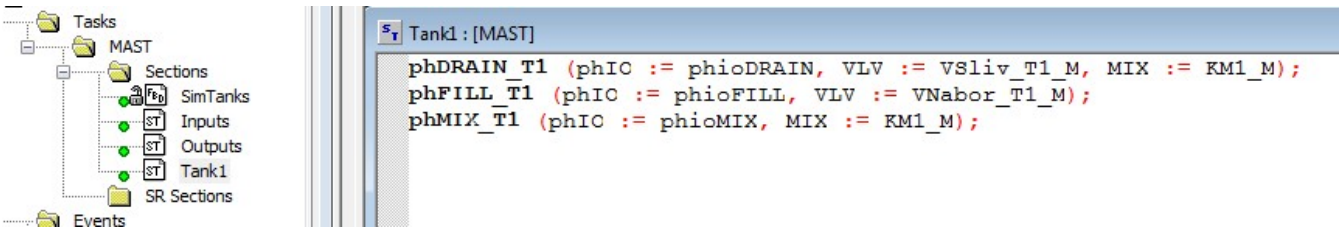


Рисунок 4.8 – Секція Tank1 з викликом етапів

- Скомпілюйте проект і запустіть на виконання в імітаторі.

4.1.4 Видаліть з етапів SCADA теги що відповідають за виконавчі механізми (клапан набору, зливу, мішалка) та реакцій, які з ними пов'язані, так як керування ними буде відбуватися з ПЛК.

- В SCADA zenon в етапах видаліть теги (параметри): «клапан набору», «клапан зливу» та «мішалка».
- Видаліть усі реакції етапів.

4.1.5 Створіть теги (параметри) **STA**, **CMD** та **MODE** для етапів в SCADA та прив'яжіть їх до відповідних змінних.

- Для етапів пропишіть теги **STA**, **CMD** та MODE (Рисунок 4.9).
- Пропишіть для них відповідні змінні (Рисунок 4.9).

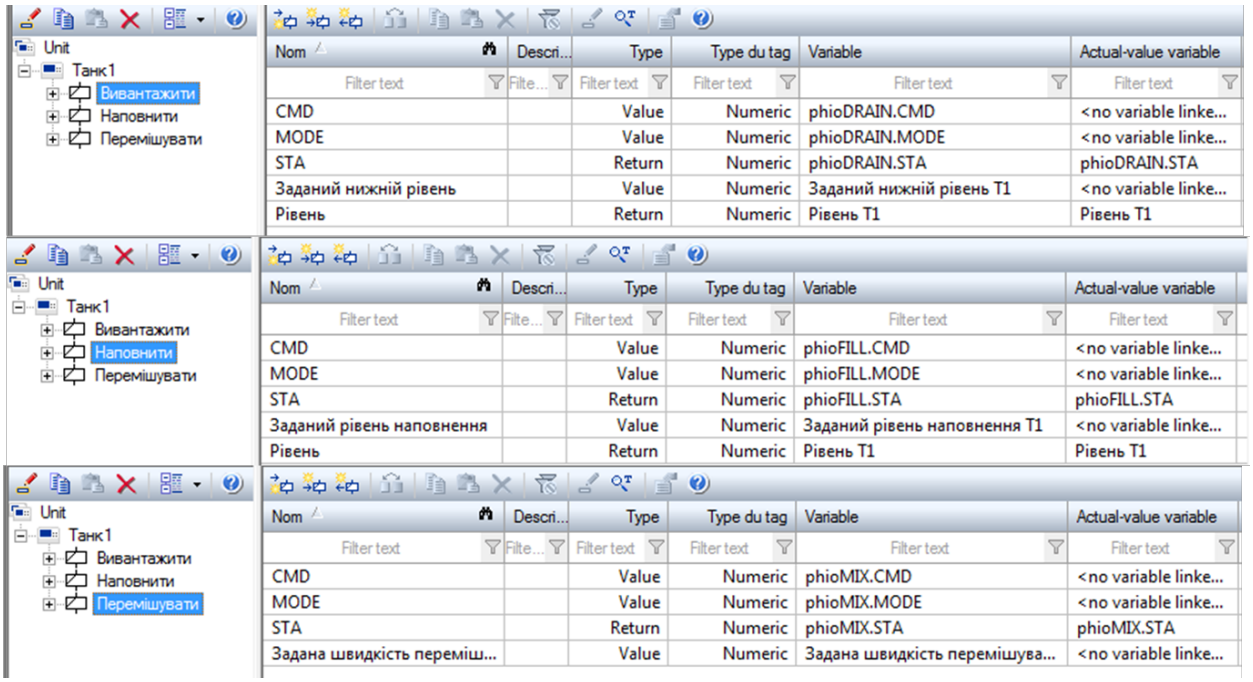


Рисунок 4.9 – Теги для етапів

4.1.6 Для кожного етапу SCADA налаштуйте умови переходу з перехідних станів в усталені стани, по значенню змінної **STA** в ПЛК.

- У групі властивостей «**Condition transienstates**» для етапу «Вивантажити», у полі «**Paused**» за допомогою редактору формул вкажіть умову переходу: Танк1.Вивантажити.STA.Value=6.

Подивіться таблицю 4.1, у якій вказані числові значення станів. Після такого налаштування етапу в zenon, у перехідному стані «**Pausing**», щоб перейти в стан «**Paused**», REE буде очікувати виконання умови рівності значення етапу STA=6, який у нашому випадку змінюється в ПЛК. Якщо ж поле залишити по-

рожнім (як це було до цього часу), перехід зі стану «**Pausing**» в «**Paused**» в етапі zenon відбувається одразу.

- Аналогічним чином пропишіть умови для станів: **Held**, **Restarted**, **Stopped**, Aborted.

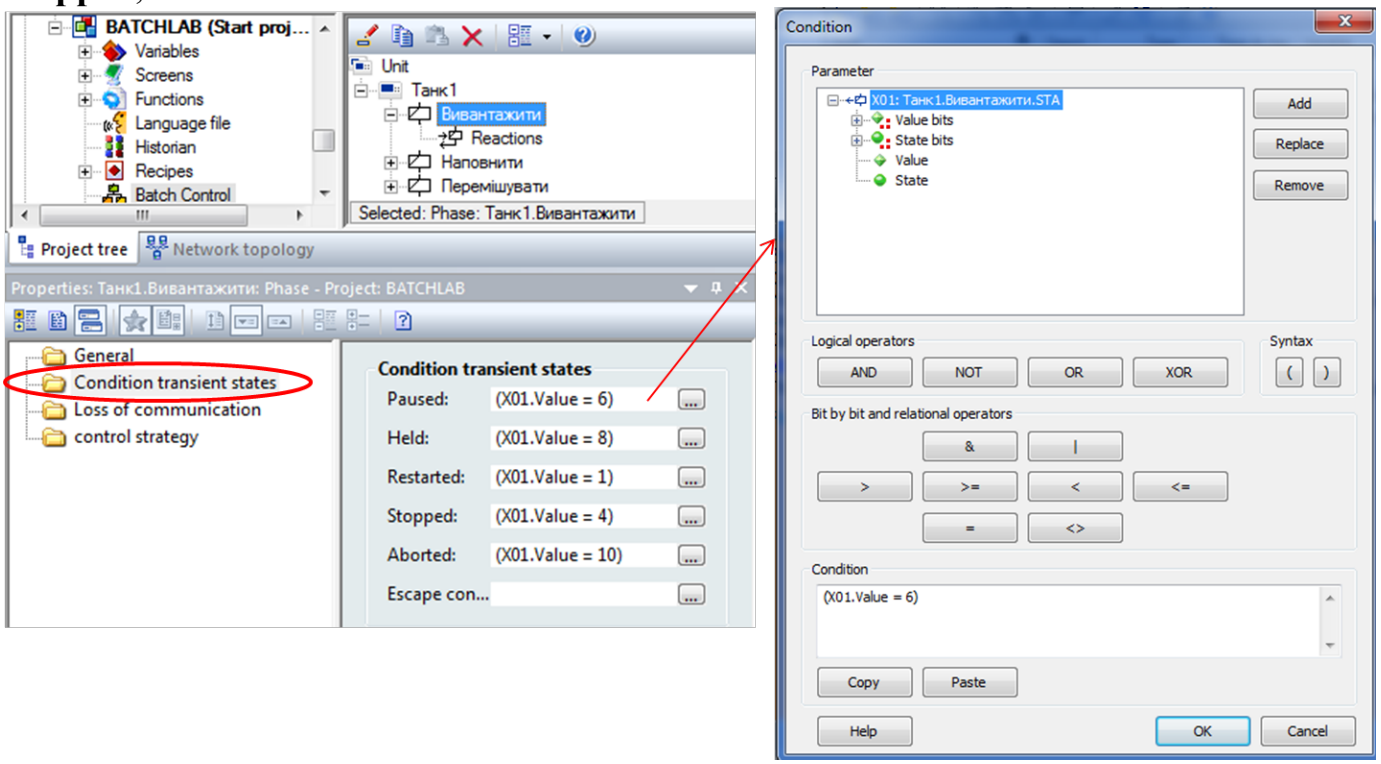


Рисунок 4.10 – Означення умов переходу для етапу «Вивантажити» Танку 1

- Аналогічними чином пропишіть умови для станів етапів «Наповнити» (зі змінною Танк1.Наповнити.STA) «Перемішувати» (зі змінною Танк1.Перемышувати.STA).

4.1.7 Для кожного етапу SCADA створіть налаштуйте реакції на події змін стану для відправки команд на ПЛК.

- Створіть реакції на події змін стану та режиму для етапу «вивантажити», як це показано Рисунок 4.11.

Таким чином, при зміні стану в змінну CMD буде записуватися команда (таблиця 4.3), а при зміні режиму, режим буде записуватися в змінну **MODE** (таблиця 4.2).

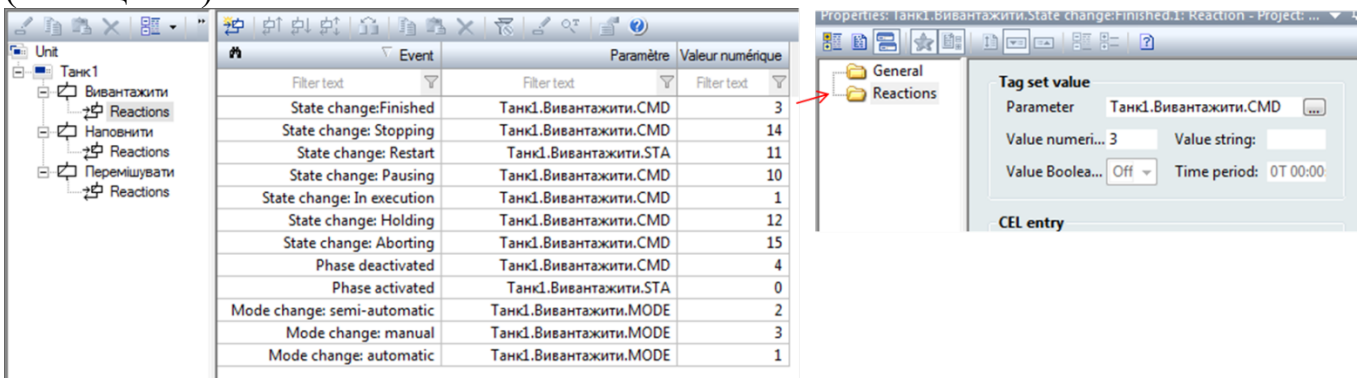


Рисунок 4.11 – Реакції на події для етапу «Вивантажити» Танку 1

- Виділіть усі реакції, скопіюйте та вставте в реакції етапу «Напов-

нити» та «Перемішувати», параметри повинні автоматично замінитися.

4.1.8 Перевірте роботу рецепту.

- Запустіть середовище виконання Zenon, відкрийте сторінку «екран Batch» запустіть рецепт «Мийка» в режимі тесту та перевірте його роботу: рецепт повинен пройти від початку до кінця аналогічно як в лабораторній роботі №3.
- Запустіть рецепт в режимі тесту ще раз; на кожному етапі перевірте правильне виконання режимів **Pause** та **Hold**.

4.2 Рецептурні елементи (Recipe elements)

4.2.1 У ПЛК створітьзмінну для етапу phHEAT, для обміну статусом, командами та режимами між SCADA та PLC.

- Створіть змінну типу **Phase INOUT**: **phioHEAT**, прив'яжіть змінну до адреси %MW112 (Рисунок 4.12).

Рисунок 4.12 – Створення змінної для етапу в Unity Pro

- Створіть масштабовані змінні для клапанів (Рисунок 4.13) та задані значення потужностей нагріву.

VNagrev_T1_M	REAL	клапан нагрівання T1 (0-100%)	%MW26
VNagrev_T2_M	REAL	клапан нагрівання T2 (0-100%)	%MW28
T1_PWRSP	REAL	відносна потужність нагрівання T1	%MW30
T2_PWRSP	REAL	відносна потужність нагрівання T2	%MW32

Рисунок 4.13 – Створення масштабованих змінних для клапанів нагріву та заданих значень потужностей нагріву (в Unity Pro)

- В секції Outputs добавте код для зміни аналогових виходів (Рисунок 4.14).

```
VNagrev_T1:=real_to_int (VNagrev_T1_M*100.0);
VNagrev_T2:=real_to_int (VNagrev_T2_M*100.0);
```

Рисунок 4.14 – Частина коду для зміни значення аналогових виходів секції **Outputs**

- Скомпілюйте проект, якщо помилок немає – переходьте до наступного пункту .

4.2.2 Створіть похідний функціональний блок (тип та екземпляр **DFB**) для реалізації етапу нагрівання в ПЛК.

- В UNITY PRO створіть **DFB**-тип **phHEAT** (Рисунок 4.15); текст коду можна скопіювати з таблиці 4.8.

DFBвключає три параметри типу **INOUT**:

- **phIO** – інтерфейсна змінна між SCADA та PLC;
- **VLV** – для керування клапаном нагрівання;
- **PWR** –задане значення потужності нагрівання (у %).

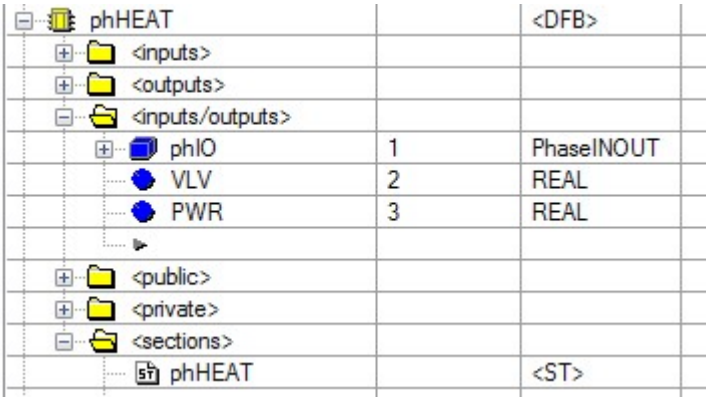


Рисунок 4.15 – Створення DFB-типу **phDRAIN**

Код забезпечує керування клапаном нагрівання VLV в залежності від заданої потужності.

Таблиця 4.8 – Код для етапу phHEAT

```
(*обробка команд керування автоматом станів етапу phHEAT*)
CASE phIO.CMD OF
  1: (*Phase started*)
    phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3: (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4: (*Phase deactivated*)
    phIO.STA:=0;
  10: (*Status change: Pausing*)
    phIO.STA:=5;
  11: (*Status change: Resuming*)
    phIO.STA:=1;
  12: (*Status change: Holding*)
    phIO.STA:=7;
  13: (*Status change: Restarting*)
    phIO.STA:=11;
  14: (*Status change: Stopping*)
    phIO.STA:=3;
  15: (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
  36:; (*Maximum execution period exceeded*)
  37:; (*Waiting period following condition exceeded*)
  38:; (*Phase started multiple times*)
  ELSE
      ;
END_CASE;
phIO.CMD:=0;(*після обробки команда обнуляється*)

CASE phIO.STA OF
  0:;(*Idle*)
  1:;(*Running*)
   VLV:=PWR;
  2:;(*Complete (Executed)*)
```

```
    VLV:=0.0;
   3:;(*Stopping*)
    VLV:=0.0;
    phIO.STA:=4;
   4:;(*Stopped*)
   5:;(*Pausing*)
    VLV:=10.0;
    phIO.STA:=6;
   6:;(*Paused*)
   7:;(*Holding*)
    VLV:=0.0;
    phIO.STA:=8;
   8:;(*Held*)
   9:;(*Aborting*)
    VLV:=0.0;
    phIO.STA:=10;
   10:;(*Aborted*)
   11:;(*Restarting*)
     phIO.STA:=1;
   ELSE
    (*якщо значення відрізняється від
     доступних - перехід в Idle*)
     phIO.STA:=0;
END_CASE;
```

- Зробіть аналіз проекту для перевірки на помилки.
- Створіть екземпляр функціонального блоку для 1-го танку (Рисунок 4.16).

Рисунок 4.16 – Екземпляр функціонального блоку для виконання етапу нагрівання для 1-го танку

- Добавте в секцію Tank1 виклик екземпляру блоку phHEAT_T1 для 1-го танку (Рисунок 4.17).

```
phDRAIN_T1 (phIC := phioDRAIN, VLV := VSliv_T1_M, MIX := KM1_M);
phFILL_T1 (phIC := phioFILL, VLV := VNabor_T1_M);
phMIX_T1 (phIC := phioMIX, MIX := KM1_M);
phHEAT_T1 (phIC := phioHEAT, VLV := VNagrev_T1_M, PWR := T1_PWRSP);
```

Рисунок 4.17 – Секція Tank1 з викликом етапів

- Скомпілюйте проект і запустіть на виконання в імітаторі.

4.2.3 У SCADA zenon створітьзмінну типу **Phase INOUT** для етапу **HEAT**, змінні для відображення стану клапану підігріву та заданого значення потужності нагрівання. Відобразіть значення змінних на екрані.

- Завантажте zenon та відкрийте проект.
- Створіть змінні для клапанів нагрівання та заданих потужностей нагріву (Рисунок 4.18).

Клапан нагрівання T1		26	0	MODRTU32 - Modbu...	REAL	Holding Register
Клапан нагрівання T2		28	0	MODRTU32 - Modbu...	REAL	Holding Register
Задана потужність нагрівання T1		30	0	MODRTU32 - Modbu...	REAL	Holding Register
Задана потужність нагрівання T2		32	0	MODRTU32 - Modbu...	REAL	Holding Register

otal / 29 filtered / 1 selected | 35 tags used / unlimited tags available

Рисунок 4.18 – Створення змінних в Zenon

- Для етапу HEAT створіть змінну типу Phase INOUT: phioHEAT прив'яжіть змінну до адреси %MW112 (Рисунок 4.19).

phioHEAT	112		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register

42 total / 30 filtered / 1 selected | 38 tags used / unlimited tags available

Рисунок 4.19 – Створення структурної змінної phioHEAT в Zenon

- В SCADA zenon на екрані **Batch** відобразіть значення для кожного з танків:
  - на клапан нагрівання;
  - задану потужність нагрівання;
  - температуру.

4.2.4 Створіть етап «Нагрівати» з відповідними параметрами.

- Способом копіювання та перейменування створіть з етапу «Наповнити»етап «Нагрівати» та відредагуйте параметри (Рисунок 4.20).

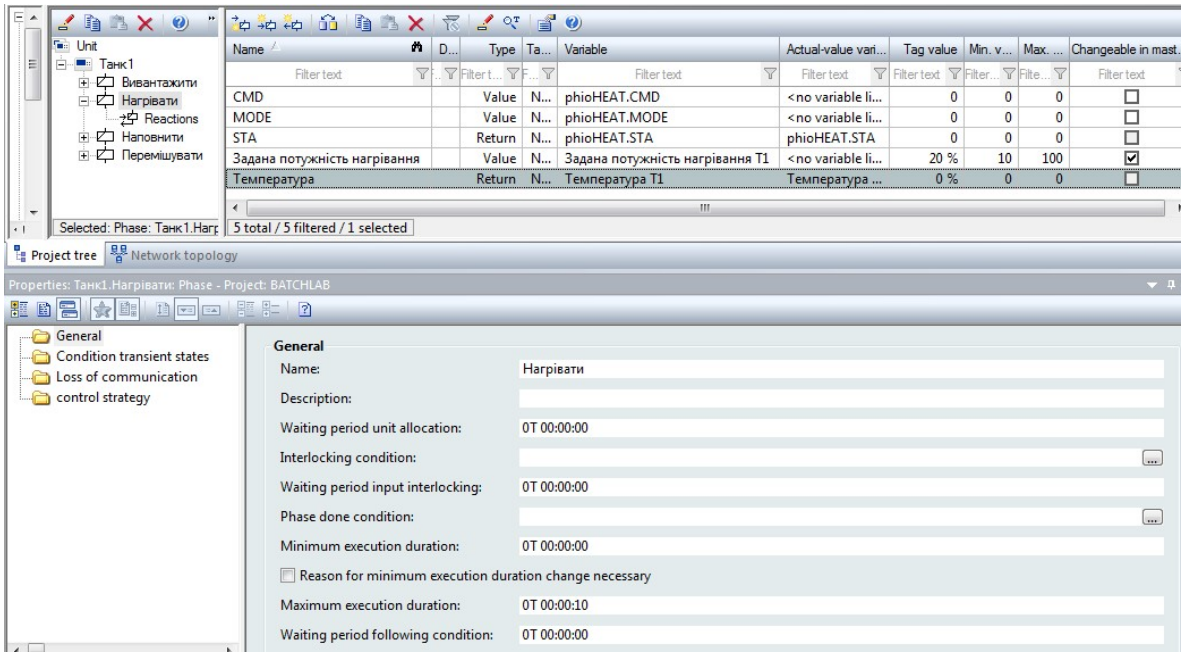


Рисунок 4.20 – Створення етапу «Нагрівати»

4.2.5 Змініть рецепт з паралельним виконанням та умовою і перевірте його роботу.

- Запустіть середовище виконання zenon, відкрийте сторінку «екран Batch» відредагуйте рецепт «Мийка» так, як це показано на Рисунок 4.21.
- Запустіть рецепт «Мийка» в режимі тесту та перевірте його роботу.

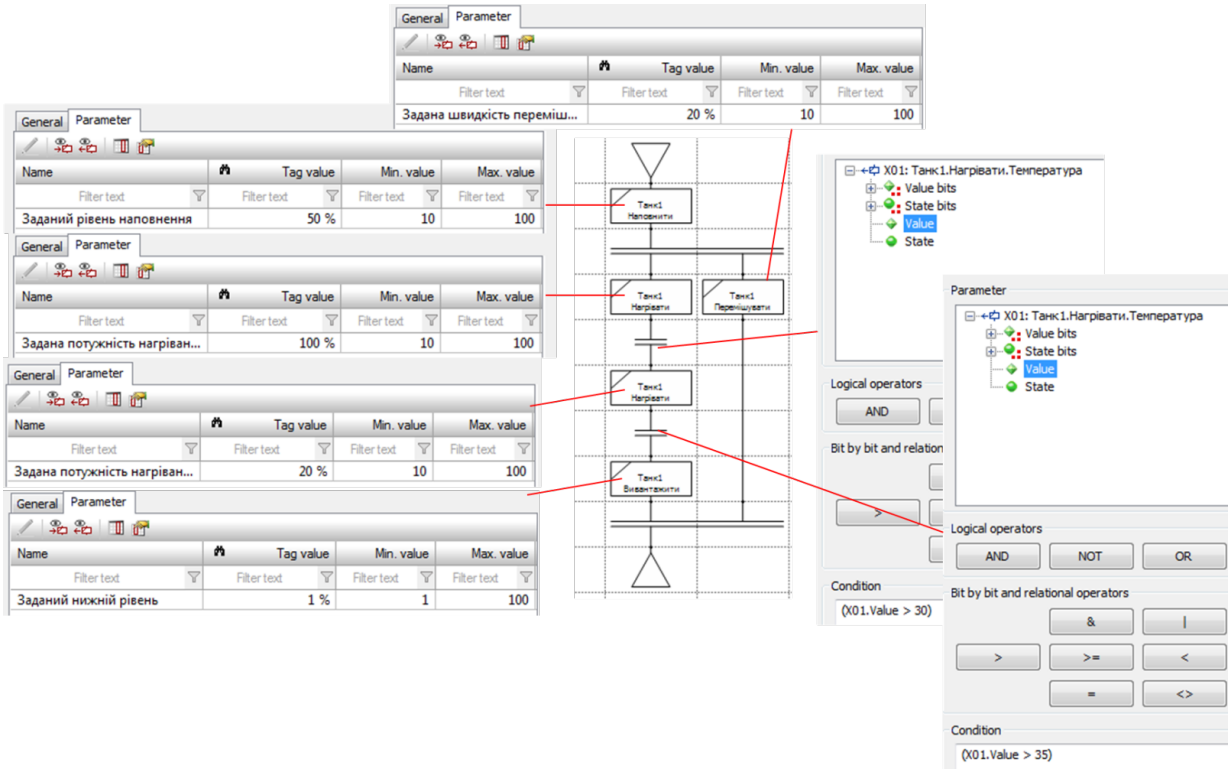


Рисунок 4.21 – Модифікація рецепту «Мийка»

4.3 Режими виконання (Execution mode)

4.3.1 Перевірте роботу рецепту в напівавтоматичному режимі.

- Запустіть середовище виконання zenon, відкрийте сторінку «екран **Batch**», відкрийте рецепт «Мийка» і переведіть його в напівавтоматичний режим.
- Запустіть рецепт на виконання в режимі тестування.
- Перед кожним переходом до наступного етапу буде засвічуватися червона стрілка, яка вказує на місце в якому призупинився рецепт; використовуючи команди продовження рецепту «**Continue recipe at all execution positions**» та «**Continue recipe at only on selected execution positions**» доведіть виконання рецепту до завершення.
- Зверніть увагу на стан активних етапів перед їх деактивацією.

4.3.2 Виконайте роботу рецепту без виконання фаз нагрівання з двома повторними наборами та зливами.

- В напівавтоматичному режимі запустіть рецепт «Мийка» на виконання.
- Після виконання етапу «Наповнити» і виконання розгалуження перемістіть вказівник до етапу «Вивантажити», а вказівник з етапу «Перемішувати» нижче нього, так як це показано на Рисунку 4.22.
- Виконайте команду «Continue recipe at all execution positions» з панелі інструментів.
- Після виконання зливу продовжте вихід з паралельного розгалу-

ження, після чого перемістіть вказівник знову на етап набору і повторіть набір і злив знову, без етапів нагрівання та перемішування.

- Після зливу доведіть рецепт до закінчення.

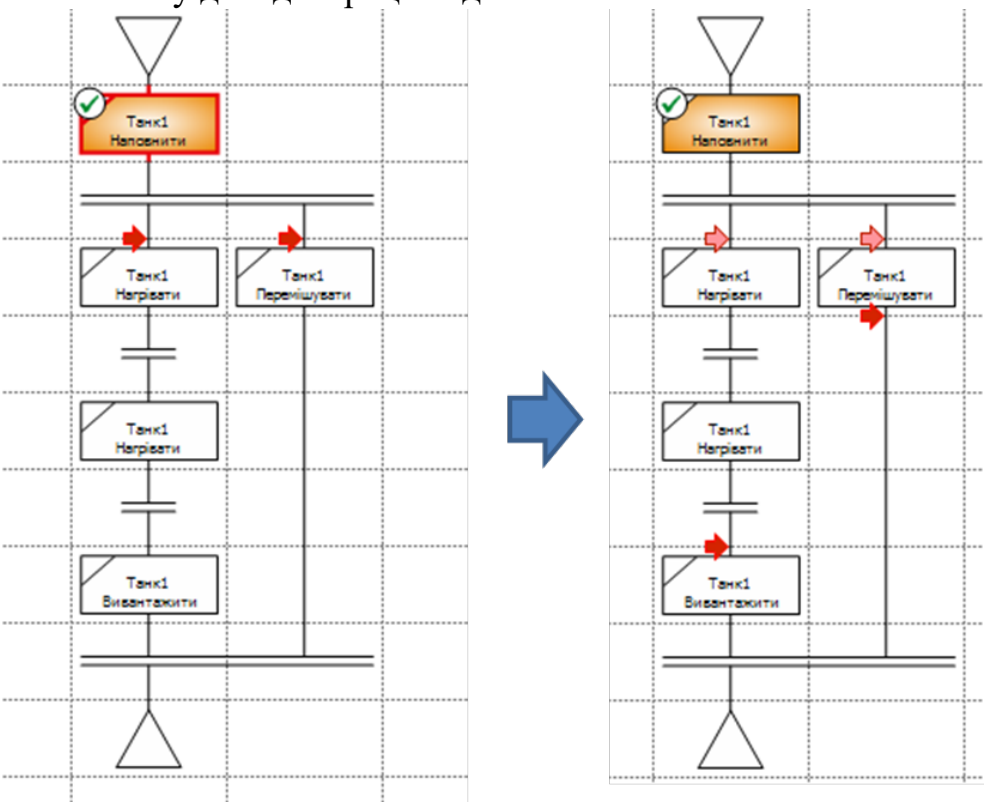


Рисунок 4.22 – Переміщення вказівників з етапу «Нагріти» до етапу «Вивантажити», та пропуск етапу «Перемішувати»

4.3.3 При наборі та нагріванні в ручному режимі виконайте вихід з етапу (закінчення етапу) та перестрибування через умову виконання.

- В автоматичному режимі запустіть рецепт «Мийка» на виконання
- При виконанні етапу «Наповнити», не чекаючи виконання умови набору, перейдіть в ручний режим і вийдіть з етапу, після чого знову перейдіть в автоматичний режим.
- При виконанні етапу нагрівання, не чекаючи умови виконання, перейдіть в ручний режим і відмініть умову переходу командою «**Escape fromphase**».
- Завершіть виконання рецепту в автоматичному режимі.

Контрольні питання

1. Які основні змінні були створені для синхронізації роботи етапів в ПЛК та SCADA?
2. Використовуючи таблицю 4.1 прокоментуйте усі значення STA.
3. Використовуючи таблицю 4.2 прокоментуйте усі значення MODE.
4. Використовуючи таблицю 4.3 прокоментуйте усі значення CMD.
5. Які додаткові елементи для рецепту в PFC Ви знаєте?
6. Які режими для виконная рецептів Ви знаєте? Коротко опишіть їх.

ЛАБОРАТОРНА РОБОТА 5. Реалізація Модулів Керування (Control Module)

Мета роботи – навчитися створювати для ПЛК логіку модулів керування відповідно до ISA-88.

Загальні теоретичні відомості

У попердніх лабораторних роботах для керування виконавчими механізмами (ВМ) використовувались безпосередньо вихідні змінні. Однак ВМ, наприклад клапан, це більш складний об'єкт, який окрім дискретних сигналів (відкрити/закрити) потребує:

- контролю стану за наявними датчиками положення;
- сигналізування при несправності: не відкрився, не закрився, зрушив зі стану;
- функціонування в режимах ручний/автоматичний: при ручному режимі команди керування видаються оператором.

Окрім цього, в залежності від типу ВМ, додатково може знадобитися ведення певної статистичної інформації про кількість включень, кількість тривог за певний час і т.д. З точки зору фізичної моделі, виконавчий механізм – це реальний фізичний пристрій, який знаходиться на найнижчому рівні ієрархії фізичних пристроїв підприємства. В термінах ISA-88, виконавчий механізм – це Модуль Керування (**Control Module**). Модуль Керування може виконувати тільки базові функції керування (**basic control**). Реалізація цих функцій в поєднані з описом стану Модуля Керування є апаратурним об'єктом (**Equipment Entity**). У даній лабораторній роботі необхідно реалізувати апаратурні об'єкти, та базові функції керування для цих об'єктів.

На Рисунку 5.1 показаний можливий варіант виділення Модулів Керування (**Control Module**, надалі **CM**) для нашого об'єкта керування, а в таблиці 5.1 наведений перелік цих об'єктів, вхідні та вихідні параметри, режими, стани та особливості керування.

Таблиця 5.1 – Перелік **CM** та параметрів їх роботи

Назва CM	Опис	Вихідні змінні (на ВМ)*	Вхідні змінні (датчики)**	Примітка
CMVnab_D1	Клапан набору D1	VNabor_D1_M	VNabor_D1_CLS_M	Можна реалізувати єдиним типом «cmVLVD»; Команди програмного керування: відкрити/закрити; Команди оператора: перевести в ручний/автоматичний, відкрити, закрити; Стани: відкритий, закритий, відкривається, закривається, невизначений; Стани тривог: не відкрився, не закрився, порушення стану; Режими: ручний/автоматичний;
CMVslv_D1	Клапан зливу D1	VSliv_D1_M	VSliv_D1_CLS_M	
CMVnab_D2	Клапан набору D2	VNabor_D2_M	VNabor_D2_CLS_M	
CMVslv_D2	Клапан зливу D2	VSliv_D2_M	VSliv_D2_CLS_M	
CMVnab_T1	Клапан набору T1	VNabor_T1_M	VNabor_T1_OPN_M VNabor_T1_CLS_M	
CMVslv_T1	Клапан зливу T1	VSliv_T1_M	VSliv_T1_OPN_M VSliv_T1_CLS_M	
CMVnab_T2	Клапан набору T2	VNabor_T2_M	VNabor_T2_OPN_M VNabor_T2_CLS_M	
CMVslv_T2	Клапан зливу T2	VSliv_T2_M	VSliv_T2_OPN_M	

			VSliv_T2_CLS_M	
CMVdoz	Клапан перемикання дозатору	VDoz_T1toT2_M	VDoz_T1_OPN_M VDoz_T2_OPN_M	
CMVngr_T1	Клапан нагрівання T1	VNagrev_T1_M	відсутні	Можна реалізувати єдиним типом «cmVLVA»; Команди програмного керування: задане значення на клапан; Команди оператора: перевести в ручний/автоматичний, ручне завдання на ВМ; Стани тривог: відсутні; Режими: ручний/автоматичний;
CMVngr_T2	Клапан нагрівання T2	VNagrev_T2_M	відсутні	
CMMix_T1	Мішалка T1	KM1_M SC1_SP	GSM1_M	Можна реалізувати єдиним типом «cmMIX»; Команди програмного керування: включити/відключити, задане значення на ПЧ; Команди оператора: перевести в ручний/автоматичний, ручне завдання на ПЧ, ручне вмикання/розмикання; Стани: вмикається, вимикається, увімкнений, вимкнений, невизначений; Стани тривог: не включився, не відключився; Режими: ручний/автоматичний;
CMMix_T2	Мішалка T2	KM2_M SC1_SP	GSM2_M	

* - вихідні змінні є проміжними внутрішніми змінними, які в секції «Outputs» присвоюються реальним виходам

** - вхідні змінні треба створити як внутрішні і переприсвоїти входами в секції «Inputs»

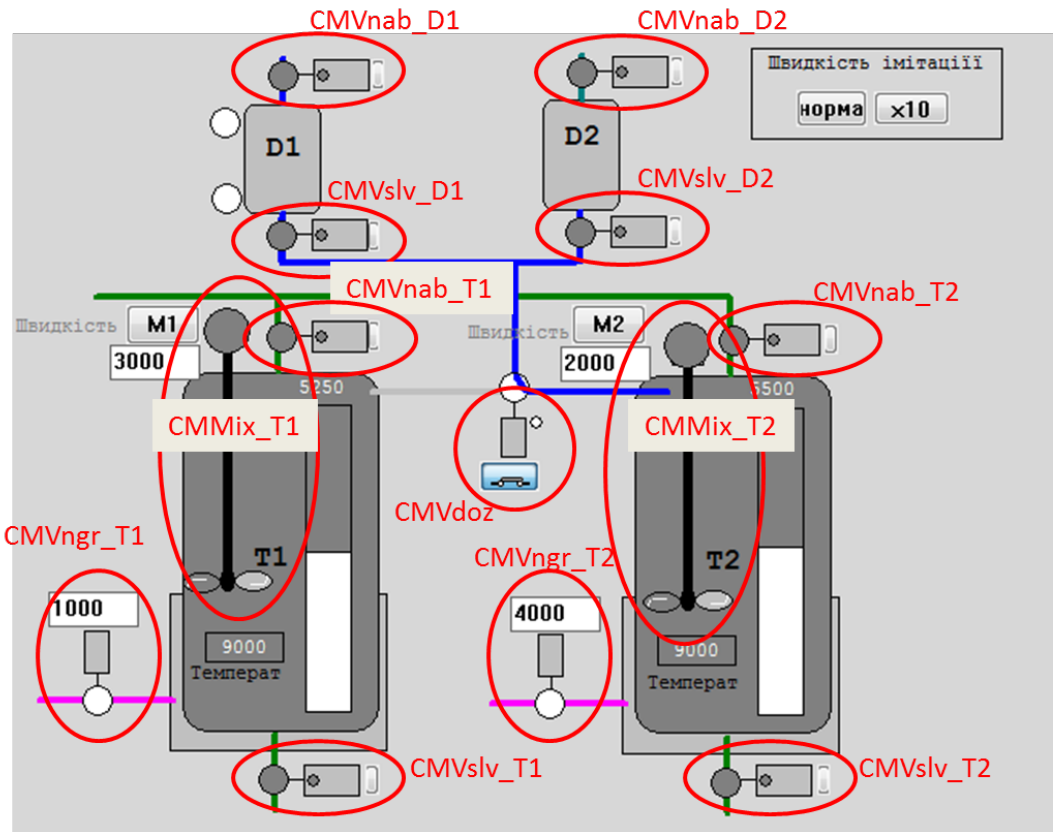


Рисунок 5.1 – Виділення Модулів Керування (СМ)

Для взаємодії ЛМІ з **СМ** стандарт ISA-88 рекомендує аналогічний до етапів механізм використання слова статусу та команди. Для цього створимо тип **CM_INOUT**, змінні на основі якого будуть повідомляти оператора про стан **СМ**, та дадуть можливість відправляти команди до **СМ**. Таким чином такий тип повинен включати принаймні два поля: **STA** (статус) та **CMD** (команда). Кожний біт статусу буде відповідати за певний стан, а команда буде задаватися певним числовим кодом.

У таблиці 5.2 показаний варіант побітового представлення статусу та значення команд які використовуватимуться в даній лабораторній роботі для дискретних клапанів. Слід зазначити, що кодування **STA** та **CMD**, як і їх обов'язкова наявність, не означена в стандарті ISA-88.

Таблиця.5.2. – Опис структуриCM_INOUT з бітовими станами для клапану

Атрибут	**Тип**	**Біт**	**Опис**
STA	INT	0 ALMOPN	=1 Не відкрився (скидаються при зміні команди або стану датчика)
		1 ALMCLS	=1 Не закрився (скидаються при зміні команди або стану датчика)
		2 ALMSNSR	=1 Помилка датчика (одночасне спрацювання обидвох датчиків положення)
		3 ALMSHFT	=1 Порушення стану (при закритому чи відкритому стані клапану зміна стану датчику без зміни команди на клапан)
		5 OPNS	=1 Відкривається
		6 CLSS	=1 Закривається
		7 OPND	=1 Відкритий
		8 CLSD	=1 Закритий
		9 MAN	=1 ручний режим
		10 GSSOPN	стан датчика відкриття
		11 GSSCLS	стан датчик закриття
		12 OUT	стан виходу наклапан
CMD	INT		Команди: 16#0001 - CMD_OPN (відкрити) 16#0002 - CMD_CLS (закрити) 16#0301 – перевести в ручний режим 16#0302 – перевести в автоматичний режим

Завдання до виконання лабораторної роботи

1. Створення в Unity Pro внутрішніх змінних для датчиків положення та прив'язка їх до реальних входів.
2. Створення в UnityPro похідних функціональних блоків (**DFB**) для реалізації логіки керування СМ, тобто апаратурних об'єктів.
3. Створення в SCADA zenon інтерфейсних змінних для взаємодії з створеними СМ та реалізація людино-машинного інтерфейсу для контролю та керування ними.
4. Модифікація етапів з прив'язкою їх до **СМ**.

Порядок проведення роботи

5.1 Реалізація для СМ дискретних клапанів в Unity Pro

5.1.1 Створіть в **UNITY PRO** тип **CM_INOUT** та змінні для **СМ** виконавчих механізмів.

- Завантажте проект **UNITY PRO** з минулої лабораторної роботи.
- Створіть внутрішні змінні (рис.5.1 та табл.5.1) та зробіть перепризначення в секції **INPUTS** зовнішнім вхідним змінним (типу **EBOOL**), що відповідають за кінцеві положення виконавчих механізмів, як це показано в таблиці 5.3.

Таблиця 5.3 – Код для секції INPUTS

```
VNabor_D1_CLS_M:=%I0.1.8;
VSliv_D1_CLS_M:=%I0.1.10;
VNabor_D2_CLS_M:=%I0.1.9;
VSliv_D2_CLS_M:=%I0.1.11;
VNabor_T1_OPN_M:=%I0.1.0;
VNabor_T1_CLS_M:=%I0.1.4;
VSliv_T1_OPN_M:=%I0.1.2;
VSliv_T1_CLS_M:=%I0.1.6;
VNabor_T2_OPN_M:=%I0.1.1;
VNabor_T2_CLS_M:=%I0.1.5;
VSliv_T2_OPN_M:=%I0.1.3;
VSliv_T2_CLS_M:=%I0.1.7;
VDoz_T1_OPN_M:=%I0.3.0;
VDoz_T2_OPN_M:=%I0.3.1;
GSM1_M:=%I0.3.2;
GSM2_M:=%I0.3.3;
```

- Створіть структурний тип даних CM_INOUT (Рисунок 5.2).
- Створіть змінні типу **CM_INOUT** для дискретних клапанів, наведених в таблиці 5.1 та прив’яжіть їх до адрес %MW (Рисунок 5.2).

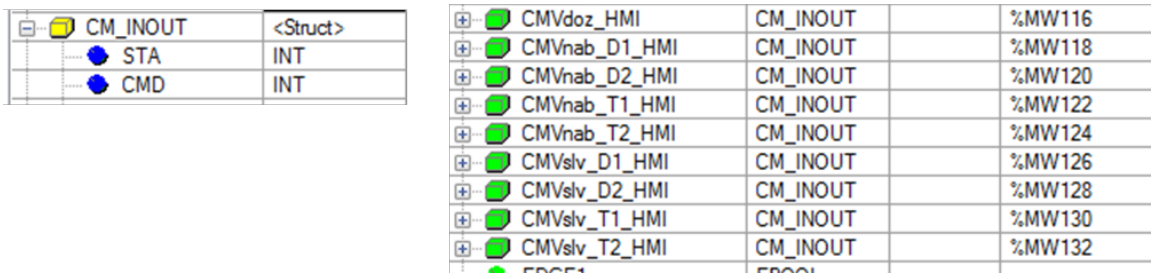


Рисунок 5.2 – Створення типу та змінних **CM_INOUT**

Окрім змінної, що забезпечує взаємодію з HMI, необхідно передбачити можливість конфігурування **СМ**, збереження його внутрішнього стану а також керування з програмної логіки (в противагу ручному керуванню). Можливим варіантом є використання для цього даних екземпляру функціонального блоку. У цій лабораторній роботі стан **СМ**, його конфігурування та керування реалізовується теж через змінну, а не через екземпляр. Опис такого структурного типу для дискретних клапанів показаний в таблиці 5.4.

Таблиця 5.4 – Опис структури **CM_VLVDCFG**

Атрибут	**Тип**	**Опис**
STA	INT	статусне слово (аналогічне CM_INOUT)
CMD	INT	слово команди (аналогічне CM_INOUT)
PRM	INT	біти параметрування: біт 0 – 1= наявність датчика відкриття біт 1 – 1= наявність датчика закриття
STEP1	INT	крок: 0 – ініціалізація 1 – невизначений стан (резерв для майбутнього застосування) 2 – закритий 3 – відкритий 4 – відкривається 5 – закривається
TSTEP1	INT	час кроку (в секундах)
TOPN_SP	INT	заданий максимальний час відкриття (в секундах), використовується для організації тривог

5.1.2 Створіть в UNITY PRO тип **CM_VLVDCFG** та змінні для **CM** виконавчих механізмів.

- Створіть структурний тип **CM_VLVDCFG** (Рисунок 5.3).
- Створіть змінні типу **CM_VLVDCFG** для дискретних клапанів, наведених в таблиці 5.3 (Рисунок 5.3).
-

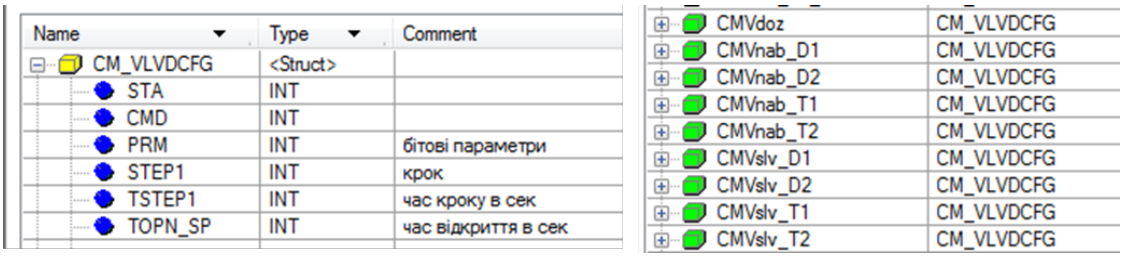


Рисунок 5.3 – Створення типу та змінних **CM_INOUT**

- Тепер необхідно створити тип функціонального блоку для реалізації CM клапанів.

- Для щойно створених змінних виставте значення при ініціалізації (**VALUE) TOPN_SP** рівним 5 секунд (приклад наведений на Рисунок 5.4). Це значення затримки очікування датчиків кінцевого положення для клапанів.

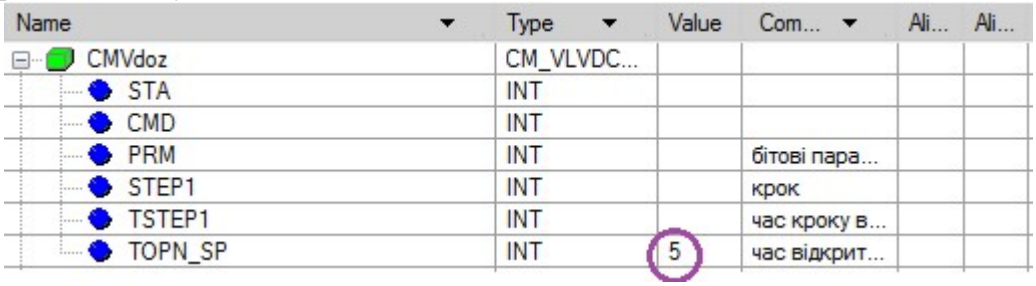


Рисунок 5.4

- Для того, щоб ці значення вступили в силу, можна проініціалізувати ПЛК. Для цього переведіть ПЛК в Stop і в меню виберіть команду PLC → Init.

5.1.3 Створіть в UNITY PRO тип **смVLVD** для СМ виконавчих механізмів.

- Ознайомтеся з описом типу та його програмною реалізацією, що наведені на Рисунку 5.5-5.7.
- Створіть **DFB** тип **cmVLVD**.
- Створіть програмну реалізацію типу **cmVLVD**, для чого можна скористатися кодом, наведеним в таблиці 5.5.
-

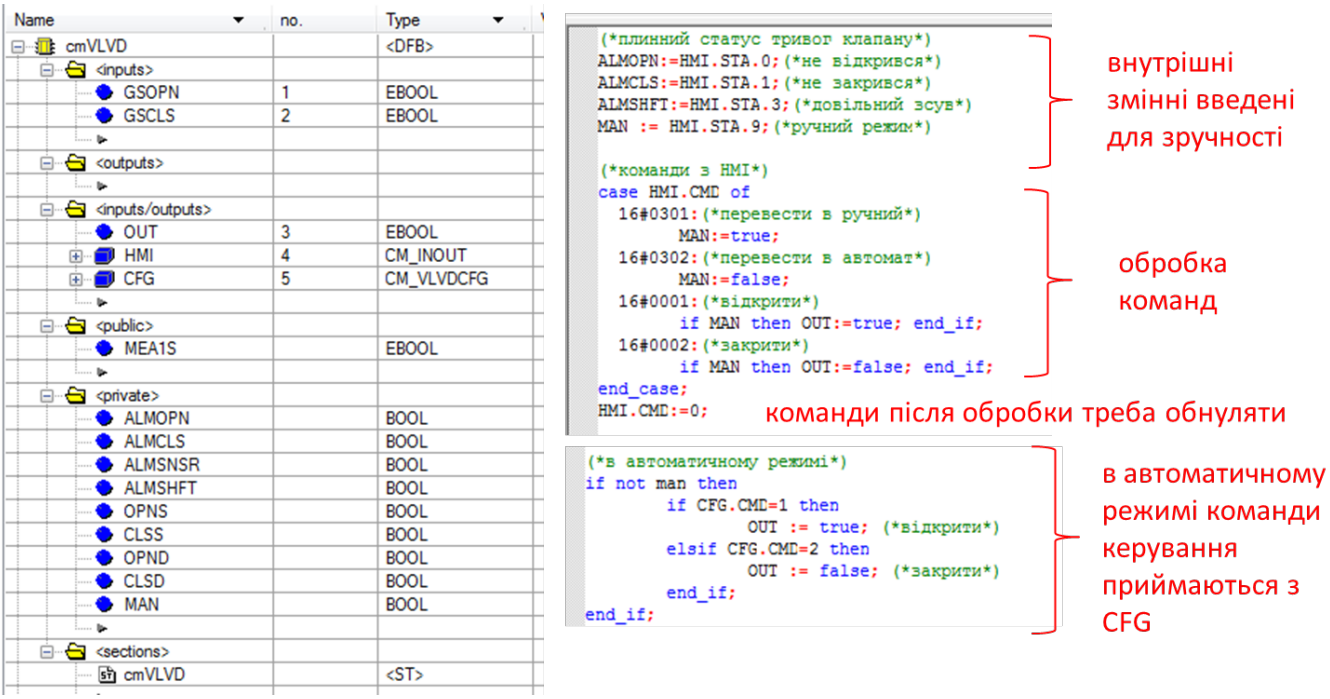


Рисунок 5.5 – Створення DFB типу та змінних cmVLVD

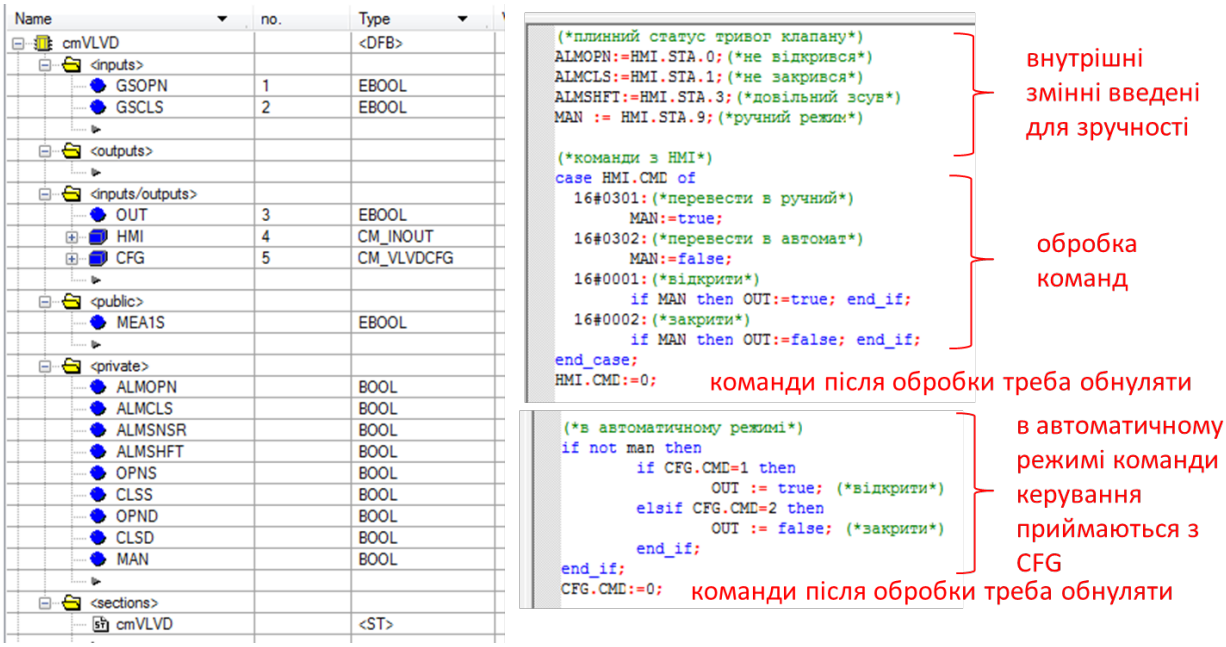


Рисунок 5.6 – Створення DFB типу та змінних cmVLVD (продовження)

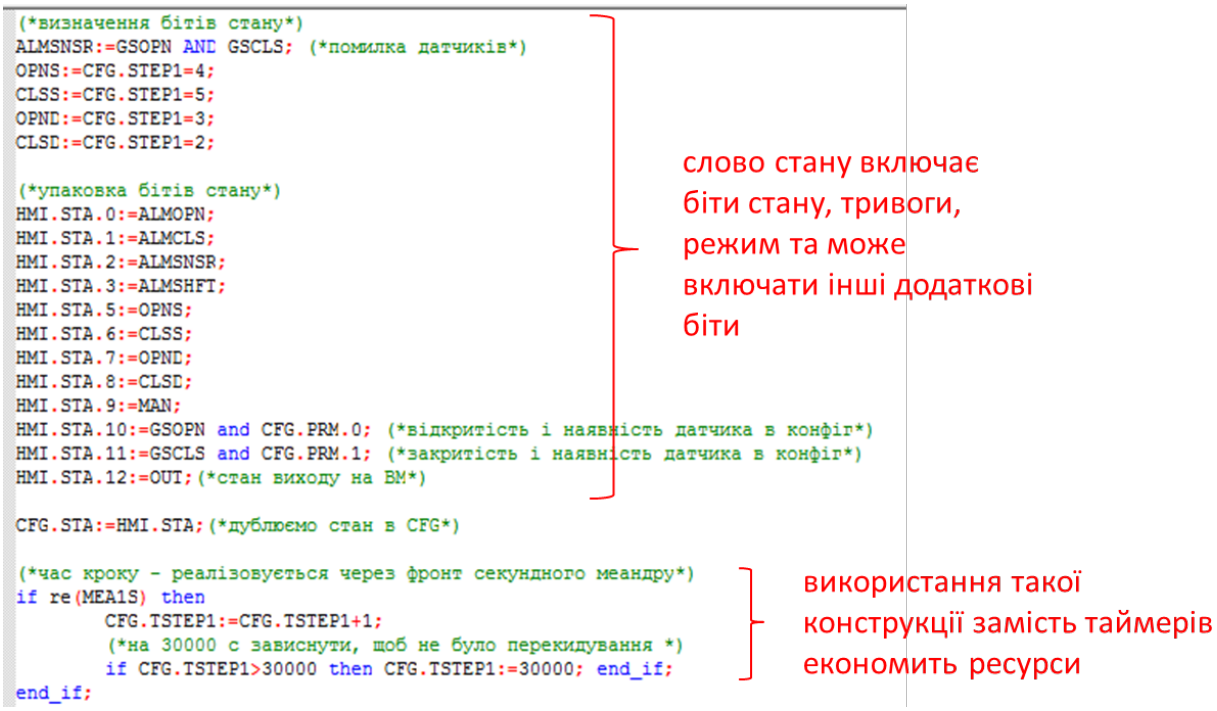


Рисунок 5.7 – Створення DFB типу та змінних cmVLVD (продовження)

Таблиця 5.5 – Код для **DFB cmVLVD**

```
(*плинний статус тривог клапану*)
ALMOPN:=HMI.STA.0;(*не відкрився*)
ALMCLS:=HMI.STA.1;(*не закрився*)
ALMSHFT:=HMI.STA.3;(*довільний зсув*)
MAN := HMI.STA.9;(*ручний режим*)

(*команди з HMI*)
case HMI.CMD of
  16#0301:(*перевести в ручний*)
     MAN:=true;
  16#0302:(*перевести в автомат*)
     MAN:=false;
  16#0001:(*відкрити*)
     if MAN then OUT:=true; end_if;
  16#0002:(*закрити*)
     if MAN then OUT:=false; end_if;
end case;
HMI.CMD:=0;

(*в автоматичному режимі*)
if not man then
     if CFG.CMD=1 then
           OUT := true; (*відкрити*)
     elsif CFG.CMD=2 then
           OUT := false; (*закрити*)
     end_if;
end_if;
CFG.CMD:=0;

CASE CFG.STEP1 OF
  0: (*ініціалізація/невизначений стан*)
     ALMOPN:=false; ALMCLS:=false; ALMSHFT := false;
     (*визначення в якому стані знаходиться клапан*)
     if GSOPN then CFG.STEP1:=3; (*відкритий*)
     elsif GSCLS then CFG.STEP1:=2; (*закритий*)
     elsif OUT then CFG.STEP1:=4; (*відкривається*)
     else CFG.STEP1:=5; (*закривається*)
```

```
      end_if;
      CFG.TSTEP1:=0; (*обнулити час кроку*)
   2: (*закритий*)
      ALMOPN:=false; ALMCLS:=false; ALMSHFT := false;
      if OUT then
            CFG.STEP1:=4; CFG.TSTEP1:=0;
      elsif NOT GSCLS and CFG.TSTEP1>1 then
            ALMSHFT:=true;(*довільний зсув*)
      end_if;
   3: (*відкритий*)
      ALMOPN:=false; ALMCLS:=false; ALMSHFT := false;
      if not OUT then
            CFG.STEP1:=5; CFG.TSTEP1:=0;
      elsif NOT GSOPN and CFG.TSTEP1>1 then
            ALMSHFT:=true;(*довільний зсув*)
      end_if;
   4: (*відкривається*)
      if GSOPN then
            CFG.STEP1:=3; CFG.TSTEP1:=0;
      elsif not OUT then
            CFG.STEP1:=5; CFG.TSTEP1:=0;
      elsif CFG.TSTEP1>=CFG.TOPN_SP then
             ALMOPN:=true;
      end_if;
   5: (*закривається*)
      if GSCLS then
            CFG.STEP1:=2; CFG.TSTEP1:=0;
      elsif OUT then
            CFG.STEP1:=4; CFG.TSTEP1:=0;
      elsif CFG.TSTEP1>=CFG.TOPN_SP then
             ALMCLS:=true;
      end_if;
   ELSE (*ініціалізація*)
      CFG.STEP1:=0; CFG.TSTEP1:=0;
END_CASE;

(*визначення бітів стану*)
ALMSNSR:=GSOPN AND GSCLS; (*помилка датчиків*)
OPNS:=CFG.STEP1=4;
CLSS:=CFG.STEP1=5;
OPND:=CFG.STEP1=3;
CLSD:=CFG.STEP1=2;

(*упаковка бітів стану*)
HMI.STA.0:=ALMOPN;
HMI.STA.1:=ALMCLS;
HMI.STA.2:=ALMSNSR;
HMI.STA.3:=ALMSHFT;
HMI.STA.5:=OPNS;
HMI.STA.6:=CLSS;
HMI.STA.7:=OPND;
HMI.STA.8:=CLSD;
HMI.STA.9:=MAN;
HMI.STA.10:=GSOPN and CFG.PRM.0; (*відкритість і наявність датчика в конфіг*)
HMI.STA.11:=GSCLS and CFG.PRM.1; (*закритість і наявність датчика в конфіг*)
HMI.STA.12:=OUT;(*стан виходу на ВМ*)

CFG.STA:=HMI.STA;(*дублюємо стан в CFG*)

(*час кроку - реалізовується через фронт секундного меандру*)
if re(MEA1S) then
     CFG.TSTEP1:=CFG.TSTEP1+1;
     (*на 30000 с зависнути, щоб не було перекидування *)
     if CFG.TSTEP1>30000 then CFG.TSTEP1:=30000; end_if;
```

```
end_if;
```

5.1.4 Реалізуйте в UNITY PRO обробку СМ для виконавчих механізмів.

- Скомпілюйте проект, якщо є помилки - виправте їх.
- Створіть один екземпляр з назвою **CMVLVD** типу **cmVLVD**.
- Створіть змінну **MEA1S** типу **EBOOL** для секундного меандру .
- Створіть секцію **VLV** з кодом, наведеним на Рисунок 5.7. Для цього можна скопіювати його з таблиці 5.6.
- Скомпілюйте проект, якщо є помилки - виправте їх.
- Завантажте проект в імітатор ПЛК та запустіть його на виконання.

```
(*використання одного екземпляру для усіх викликів дає ряд переваг:
- економя пам'яті
- використання глобальних змінних для всіх екзмеплярів
- відсутність необхідності дублювання інтерфейсних змінних в блоці і в змінних*)

CMVLVD.MEA1S:=%S6; (*бітовий меандр заводимо як глобальну змінну*)

(*конфігурвування клапанів*)
CMVnab_D1.PRM.0:=false ;(*немає датчкива відкриття*)
CMVnab_D1.PRM.1:=true;(*є датчик закриття*)
CMVnab_D2.PRM.0:=false;CMVnab_D2.PRM.1:=true;
CMVSlv_D1.PRM.0:=false;CMVSlv_D1.PRM.1:=true;
CMVSlv_D2.PRM.0:=false;CMVSlv_D2.PRM.1:=true;
CMVnab_T1.PRM.0:=true;CMVnab_T1.PRM.1:=true;
CMVnab_T2.PRM.0:=true;CMVnab_T2.PRM.1:=true;
CMVslv_T1.PRM.0:=true;CMVslv_T1.PRM.1:=true;
CMVslv_T2.PRM.0:=true;CMVslv_T2.PRM.1:=true;
CMVslv_T2.PRM.0:=true;CMVslv_T2.PRM.1:=true;
CMVdoz.PRM.0:=true;CMVdoz.PRM.1:=true;

(*для клапанів з одним датчиком кінцевого положення на інший подаєм інверсну копію існуючого*)
CMVLVD (not VNabor_D1_CLS_M , VNabor_D1_CLS_M, VNabor_D1_M, CMVnab_D1_HMI, CMVnab_D1);
CMVLVD (not VNabor_D2_CLS_M , VNabor_D2_CLS_M, VNabor_D2_M, CMVnab_D2_HMI, CMVnab_D2);
CMVLVD (not VSliv_D1_CLS_M , VSliv_D1_CLS_M, VSliv_D1_M, CMVSlv_D1_HMI, CMVSlv_D1);
CMVLVD (not VSliv_D2_CLS_M , VSliv_D2_CLS_M, VSliv_D2_M, CMVSlv_D2_HMI, CMVSlv_D2);
CMVLVD (VNabor_T1_OPN_M , VNabor_T1_CLS_M, VNabor_T1_M, CMVnab_T1_HMI, CMVnab_T1);
CMVLVD (VNabor_T2_OPN_M , VNabor_T2_CLS_M, VNabor_T2_M, CMVnab_T2_HMI, CMVnab_T2);
CMVLVD (VSliv_T1_OPN_M , VSliv_T1_CLS_M, VSliv_T1_M, CMVslv_T1_HMI, CMVslv_T1);
CMVLVD (VSliv_T2_OPN_M , VSliv_T2_CLS_M, VSliv_T2_M, CMVslv_T2_HMI, CMVslv_T2);

CMVLVD (VDoz_T1_OPN_M, VDoz_T2_OPN_M, VDoz_T1toT2_M, CMVdoz_HMI, CMVdoz);
```

Рисунок 5.8 – Секція **VLV**

Таблиця 5.6 – Код для секції **VLV**

```
(*використання одного екземпляру для усіх викликів дає ряд переваг:
- економя пам'яті
- використання глобальних змінних для всіх екзмеплярів
- відсутність необхідності дублювання інтерфейсних змінних в блоці і в змінних*)

CMVLVD.MEA1S:=%S6; (*бітовий меандр заводимо як глобальну змінну*)

(*конфігурвування клапанів*)
CMVnab_D1.PRM.0:=false ;(*немає датчкива відкриття*)
CMVnab_D1.PRM.1:=true;(*є датчик закриття*)
CMVnab_D2.PRM.0:=false;CMVnab_D2.PRM.1:=true;
CMVSlv_D1.PRM.0:=false;CMVSlv_D1.PRM.1:=true;
CMVSlv_D2.PRM.0:=false;CMVSlv_D2.PRM.1:=true;
CMVnab_T1.PRM.0:=true;CMVnab_T1.PRM.1:=true;
CMVnab_T2.PRM.0:=true;CMVnab_T2.PRM.1:=true;
CMVslv_T1.PRM.0:=true;CMVslv_T1.PRM.1:=true;
CMVslv_T2.PRM.0:=true;CMVslv_T2.PRM.1:=true;
CMVslv_T2.PRM.0:=true;CMVslv_T2.PRM.1:=true;
CMVdoz.PRM.0:=true;CMVdoz.PRM.1:=true;
```

```
(*для клапанів з одним датчиком кінцевого положення на інший подаєм інверсну
копію існуючого*)
CMVLVD (not VNabor_D1_CLS_M , VNabor_D1_CLS_M, VNabor_D1_M, CMVnab_D1_HMI,
CMVnab_D1);
CMVLVD (not VNabor_D2_CLS_M , VNabor_D2_CLS_M, VNabor_D2_M, CMVnab_D2_HMI,
CMVnab_D2);
CMVLVD (not VSliv_D1_CLS_M , VSliv_D1_CLS_M, VSliv_D1_M, CMVSlv_D1_HMI,
CMVSlv_D1);
CMVLVD (not VSliv_D2_CLS_M , VSliv_D2_CLS_M, VSliv_D2_M, CMVSlv_D2_HMI,
CMVSlv_D2);
CMVLVD (VNabor_T1_OPN_M , VNabor_T1_CLS_M, VNabor_T1_M, CMVnab_T1_HMI,
CMVnab_T1);
CMVLVD (VNabor_T2_OPN_M , VNabor_T2_CLS_M, VNabor_T2_M, CMVnab_T2_HMI,
CMVnab_T2);
CMVLVD (VSliv_T1_OPN_M , VSliv_T1_CLS_M, VSliv_T1_M, CMVslv_T1_HMI,
CMVslv_T1);
CMVLVD (VSliv_T2_OPN_M , VSliv_T2_CLS_M, VSliv_T2_M, CMVslv_T2_HMI,
CMVslv_T2);

CMVLVD (VDoz_T1_OPN_M, VDoz_T2_OPN_M, VDoz_T1toT2_M, CMVdoz_HMI, CMVdoz);
```

5.2 Реалізація інтерфесу для СМ апаратурних об’єктів в SCADA zenon.

5.2.1 Створіть типи та змінні в SCADA zenon для реалізації інтерфейс них змінних **CM_INOUT**, наведених в таблиці 5.1.

- Запустіть редактор zenon та відкрийте проект з лабораторної роботи 4.
- Створіть тип змінної **CM_INOUT** та змінні на базі нього, як це показано на Рисунку 5.9.

CM_INOUT	інтерфейсні дані для СМ					
STA	Structure element	UINT/<embedded> 3				
CMD	Structure element	UINT/<embedded> 4				
CMVdoz	116	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVnab_D1	118	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVnab_D2	120	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVnab_T1	122	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVnab_T2	124	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVslv_D1	126	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVslv_D2	128	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVslv_T1	130	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	
CMVslv_T2	132	0	MODRTU32 - Modbu...	CM_INOUT	Holding Register	

Рисунок 5.9 – Створення типів та змінних в SCADA zenon для обміну з СМ

- Для кожної змінної вкажіть коментар (коментар для кожної змінної знаходиться в описі таблиці 5.1). Наведемо приклад на Рисунку 5.10.

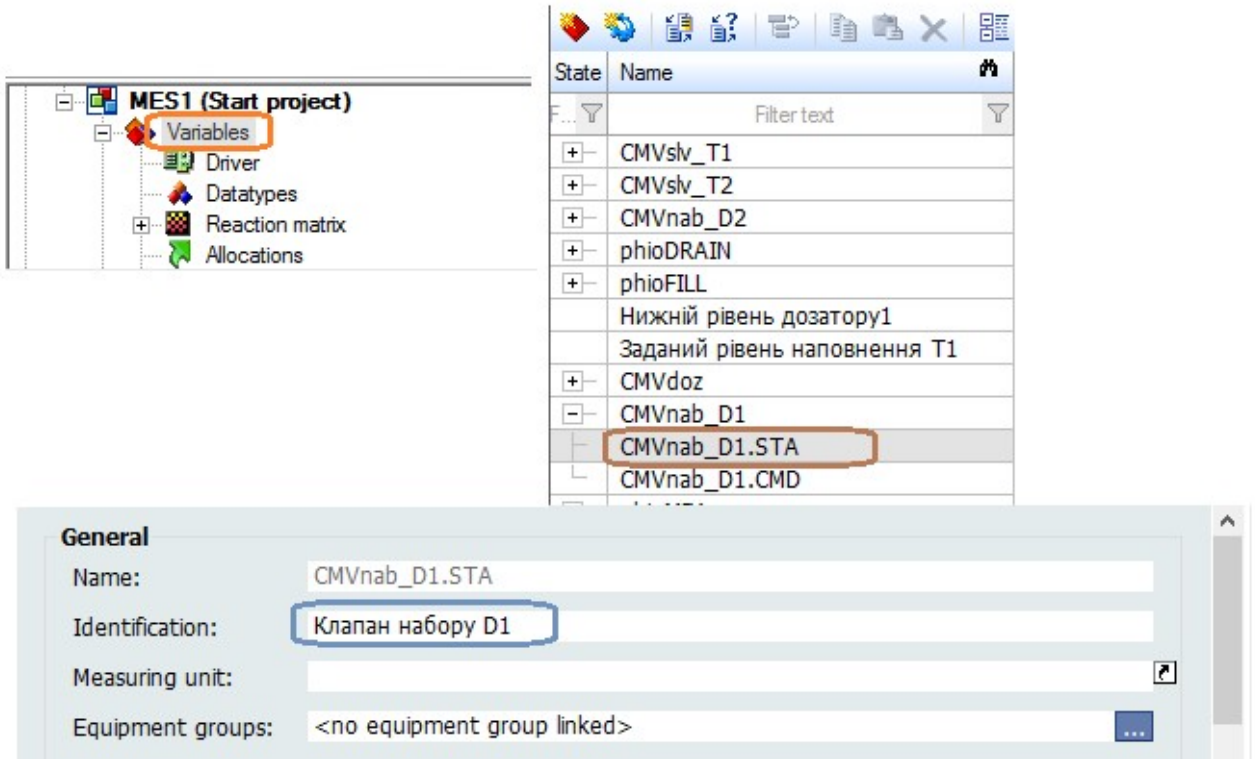


Рисунок 5.10

Стан клапану повинен відображатися на мнемосхемі комбінацією кольорів та/або тексту. У лабораторній роботі пропонується використовувати для цього заздалегідь створений символ. Для керування клапаном також створений екран, який повинен викликатися при натисканні на зображення символу. Тобто символ та сторінка вже створені, необхідно їх імпортувати та використати в своєму проекті.

5.2.2 В SCADA імпортуйте символ **VLVD** та екран для керування клапаном. Розмістіть символи для відображення станів клапанів.

- В контекстному меню редактору zenon Screens виберіть команду ImportXML і імпортуйте символ «Сивмол VLVD.XML» і екран «Вікно керування клапаном.XML».
- Розмістіть символи (як лінковані) на місцях зображення клапанів сторінки «Екран Batch», старі зображення видаліть. За необхідності розверніть символи на 90° та змініть їх масштаб.
- Замініть прив’язки до змінних відповідно до призначення символу. Для цього скористайтеся вкладкою символу «Linking rule» та кнопкою «Preview» для перевірки результату заміни (Рисунок 5.11).

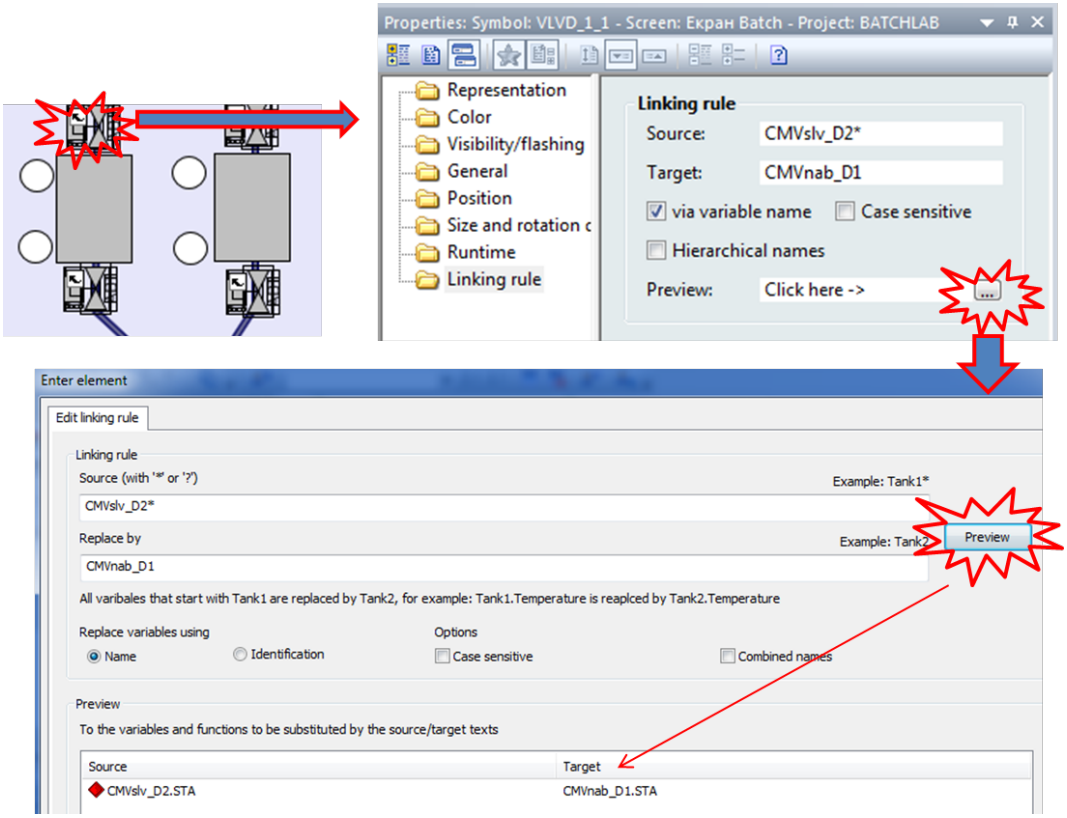


Рисунок 5.11 – Приклад заміни прив’язки для символу

Екран керування клапаном налаштований тільки на один СМ - «CMVslv_D2». Для керування іншими кранами в zenon буде використаний механізм заміни прив’язок при виклику функції «**Screen switch**». Для цього у функції вказується параметр, замість якого буде передаватися текстове значення замінника в назві змінної.

5.2.3 Створіть в zenon функції виклику та закриття сторінки клапану.

- Створіть функцію «**Screen switch**» з назвою «Відкрити вікно керування клапаном», яка відкриває в свою чергу екран «Керування клапаном».
- У параметрах функції у вкладці «Replace indices» вкажіть правило прив’язки з параметром {PARAM}, як це показано на Рисунок 5.12.

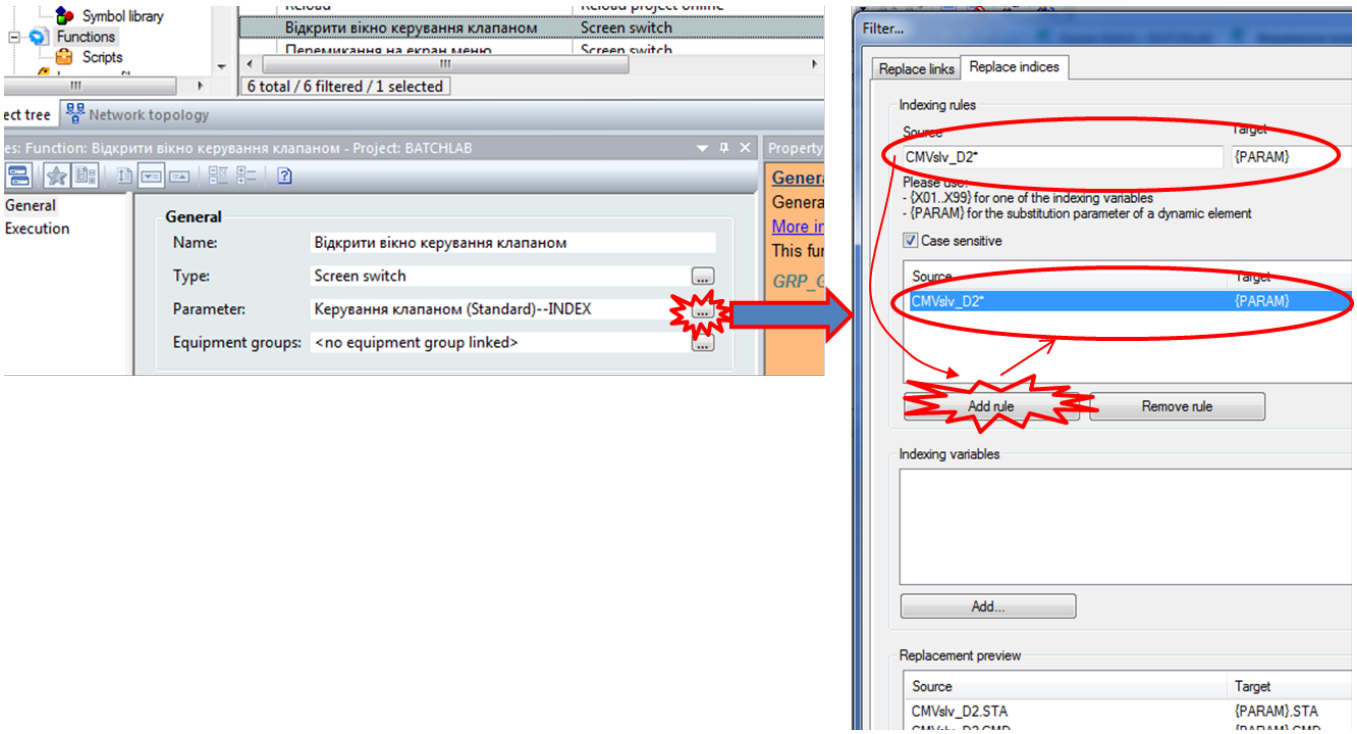


Рисунок 5.12 – Налаштування правила заміни для функції виклику екрану керування клапаном

- Створіть функцію «**Close screen**» з назвою «Закрити вікно», яка закриває поточний екран (Рисунок 5.13).

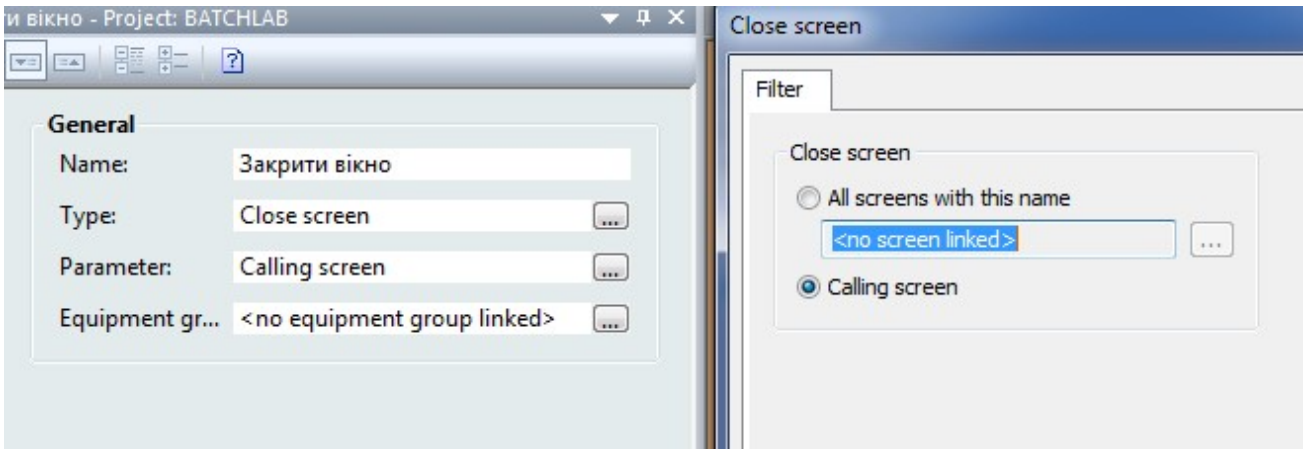


Рисунок 5.13 – Налаштування функції **Close Screen**

- Відкрийте вікно «Керування клапаном» і прив’яжіть до кнопки «Ок» функцію «Закрити вікно».

5.2.4 Розмістіть над клапанами прозорі кнопки для виклику вікон керування клапанами.

- Відкрийте «**Екран Batch**» і поверх зображення клапанів розмістіть прозорі кнопки.
- Прив’яжіть кнопки до виклику функції «Відкрити вікно керування клапаном». У налаштуваннях параметрів для кожної кнопки вкажіть назву СМ, як це показано на Рисунок 5.14 на прикладі CMVsliv_D1.

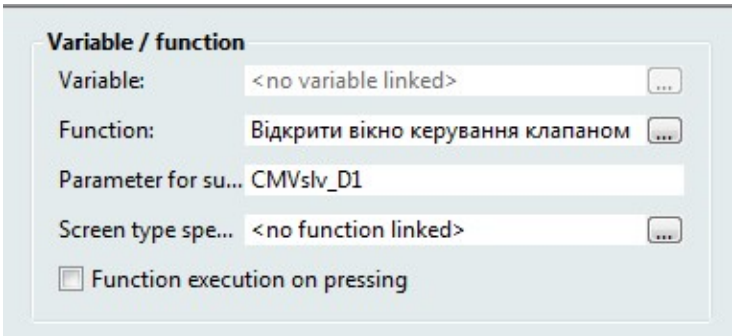


Рисунок 5.14 – Налаштування функції Close Screen

5.2.5 Перевірте роботу системи.

- Скомпілюйте проект та запустіть його на виконання. Спробуйте покерувати кранами в ручному режимі, перевірте, щоб всі вікна відкривалися для кранів вірно.
- Для клапану перемикання у певному стані буде активована тривога. Ця тривога викликана помилкою в коді програми. Крім того, для цього ж клапану не будуть відображатися стани датчиків. Знайдіть помилки в програмі і виправте їх.
- У UNITY PRO створіть таблицю анімацій, впишіть в неї адреси %I0.1.0 та %I0.1.4. За допомогою команд форсування проімітуйте тривоги «не закрився», «не відкрився» та порушення стану.

5.3 Модернізація етапів з використанням CM

Тепер необхідно модернізувати реалізацію етапів так, щоб замість звичайних змінних там використовувалися CM. Такий підхід дасть можливість гнучко керувати процесом. На даному кроці побудови системи це дає можливість використовувати ручний режими керування виконавчими механізмами. У майбутньому це також дасть можливість реалізовувати додаткову поведінку етапу в нештатних ситуаціях.

5.3.1 Модифікуйте DFB phFILL для етапу набору та phDRAIN для етапу зливу так, щоб в якості VLV використовувався тип **CM_VLVDCFG** та відповідно змініть виклик етапів. Перевірте працездатність програми.

- У **phFILL** для формального параметру «**VLV**»змініть тип з **EBOOL** на **CM_VLVDCFG**.
- Змініть реалізацію секції **phFILL** (Рисунок 5.15).
- Змініть виклик **phFILL_T1** (Рисунок 5.15).
- Зробіть те саме для етапу **phDRAIN**.

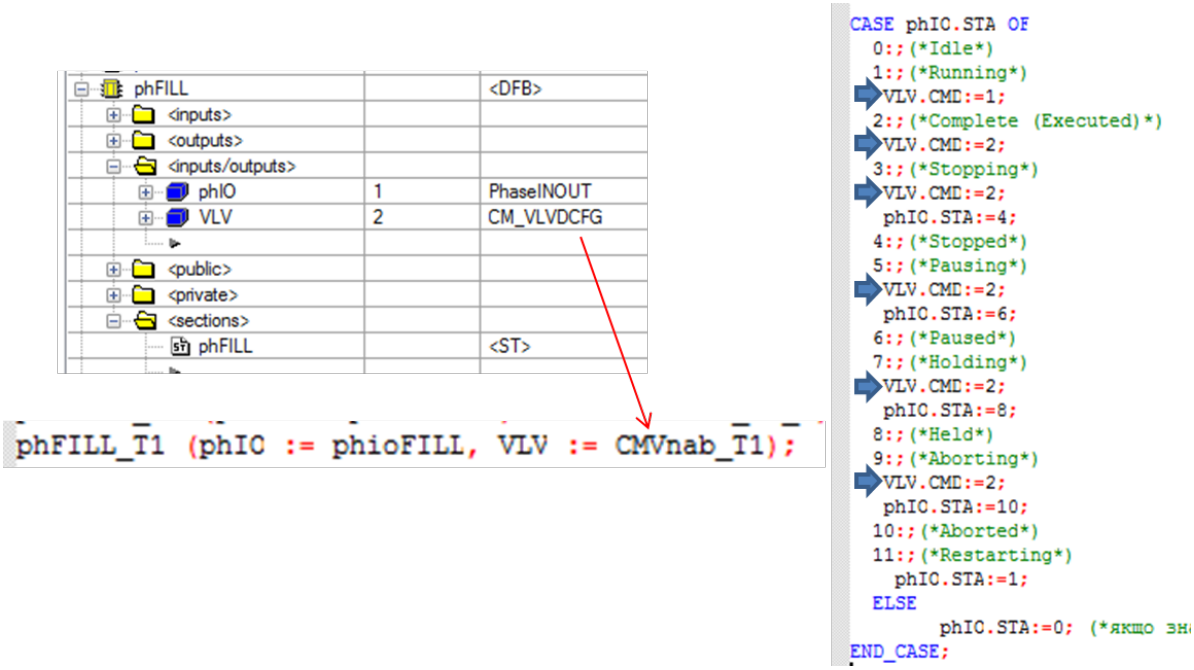


Рисунок 5.15 – Зміна інтерфейсу, реалізації та використання **CM_VLVDCFG**

- Скомпілюйте проект та запустіть на виконання zenon. Перевірте працездатність існуючого рецепту. Спробуйте перед набором перемкнути клапан набору в ручний режим та закрити його. Запустіть рецепт подивіться на стан клапану і знову переключіть його в автоматичний режим.

5.4. Створення CM для клапанів нагрівання

Окрім дискретних клапанів необхідно реалізувати CM для аналогових клапанів (таблиця 5.7). На відміну від дискретних клапанів інтерфейсні змінні окрім STA та CMD потребують ще аналогове значення виходу на клапан, яке буде використовуватися як для відображення, так і для ручної зміни (таблиця 5.7).

Для керування всередині етапів в даній лабораторній роботі можна, як і раніше, використовувати звичайне значення аналогового виходу, яке буде за необхідності в ручному режимі замінюватися значенням з HMI.

Таблиця 5.7 – Опис структури **CM_INOUTA** з бітовими станами для аналогового клапану

Атрибут	Тип	Біт	Опис
STA	INT	9 MAN	=1 ручний режим
CMD	INT		Команди: 16#0301 – перевести в ручний режим 16#0302 – перевести в автоматичний режим
VAL	REAL		стан виконавчого механізму, ручне значення вводу; в автоматичному режимі значення повторює вихід, в автоматичному – навпаки, значення задає вихід на ВМ

5.4.1 Створіть в UNITY PRO тип **CM** для аналогового клапану та реалізуйте його для клапанів нагрівання (Рисунок 5.16).

- Створіть в UNITY PRO тип **CM_INOUTA**.
- Створіть змінні **CMVngr_T1** та **CMVngr_T2** типу **CM_INOUTA** та прив'яжіть їх до реальних адрес.
- Створіть **DFB cmVLVA** та один екземпляр типу з таким саме іменем.
- У секції **VLV** пропишіть виклик **cmVLVA**.

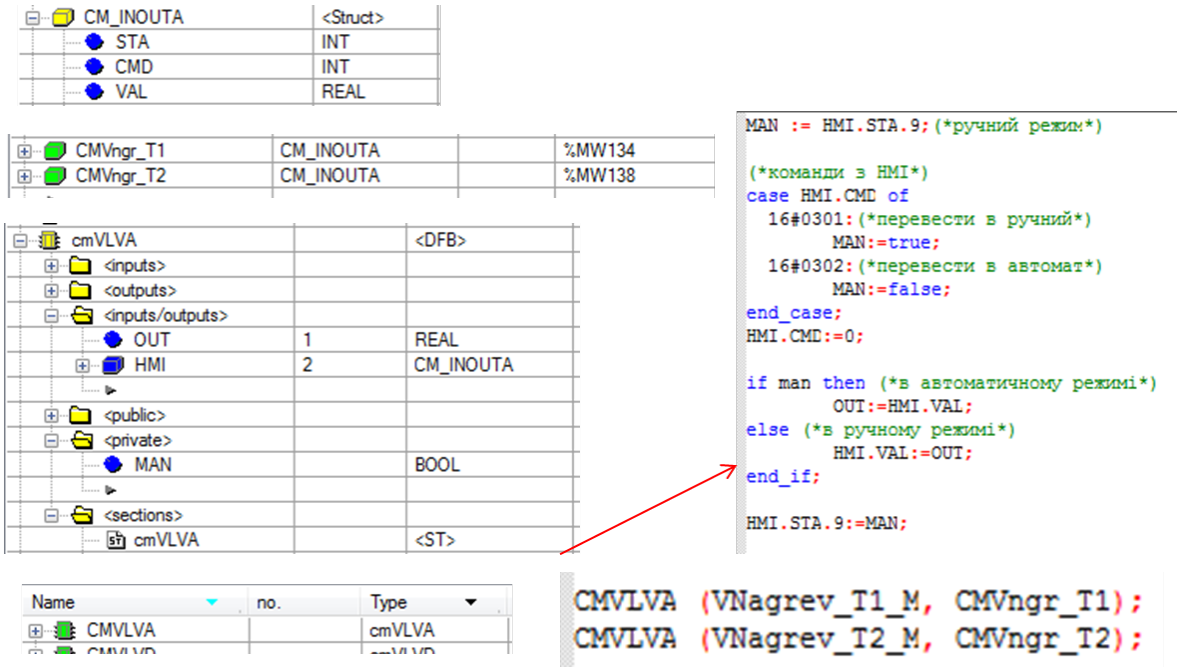


Рисунок 5.16 – Створення в UNITY PRO CM для аналогового клапану

5.4.2 Створіть в zenon тип **CM** для аналогового клапану та реалізуйте його для клапанів нагрівання (Рисунок 5.17). Перевірте роботу програми.

- У SCADA zenon створіть тип **CM_INOUTA**.
- Створіть змінні **CMVngr_T1** та **CMVngr_T2** типу **CM_INOUTA**.
- На мнемосхемі «Екран Batch» змініть прив'язку для елементу відображення значення клапану «**CMVngrT1.VAL**»та «**CMVngrT2.VAL**» з можливістю їх зміни.
- Скопіюйте з екрану «Керування клапаном» елементи відображення та зміну автоматичного/ручного режиму. Змініть прив'язки на змінні «**CMVngrT1.STA**»та «**CMVngrT2.STA**».

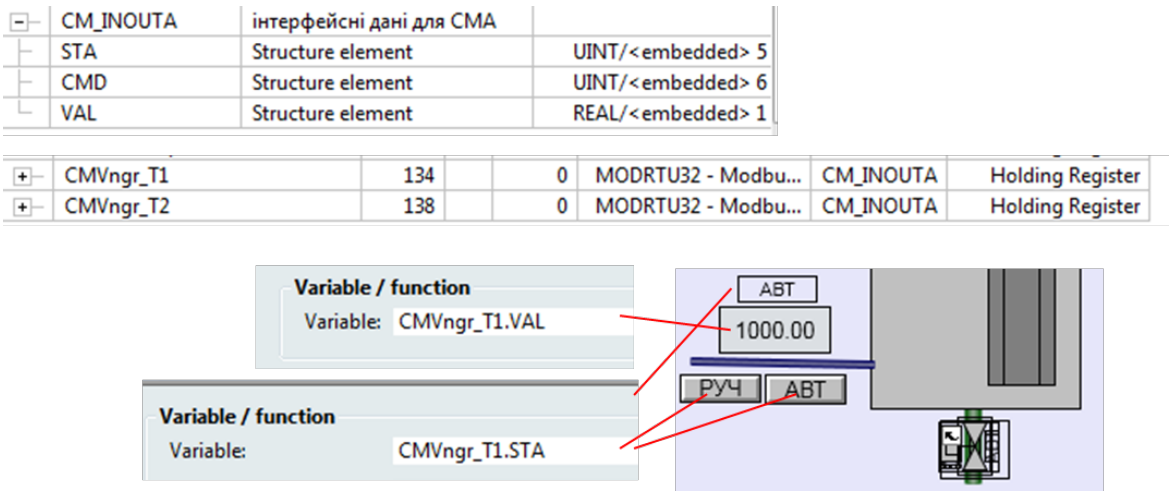


Рисунок 5.17 – Створення в zenon CM для аналогового клапану та графічного інтерфейсу для нього

- Скомпілюйте та запустіть на виконання zenon. Змініть режим роботи клапану на ручний. Задайте значення 0 %. Запустіть на виконання рецепт. На етапі нагрівання клапан буде в позиції, яку було задано в ручному режимі. Переведіть на автоматичний режим та дочекайтеся закінчення виконання рецепту.

5.5 Створення CM для приводів мішалок та модернізація етапів перемішування (phMIX) та (phHEAT)

Аналогічно необхідно реалізувати **CM** для приводу мішалок. За реалізацією вони поєднують ознаки дискретних клапанів та аналогового клапану.

5.5.1 Створіть в UNITYPRO тип **CM** для приводу мішалки та реалізуйте його для мішалки T1 та T2.

- Реалізувати самостійно.

5.5.2 Створіть в zenon тип CMдля приводу мішалки та реалізуйте його для мішалки T1 та T2. Перевірте роботу програми.

- Реалізувати самостійно.

Контрольні питання

1. Що таке виконавчий механізм, з точки зору фізичної моделі?
2. Чим може бути виконавчий механізм з точки зору фізичної моделів ISA-88?
3. Які модулі керування виділені в даній лабораторній роботі?
4. Які команди, режими і стани були виділені в даній лабораторній роботі?
5. Розкжіть про реалізацію в ПЛК вказаного викладачем CM.
6. Покажіть на прикладі, як реалізований обмін між CM в SCADA та ПЛК.

ЛАБОРАТОРНА РОБОТА 6. МОДЕЛЬ АПАРАТУРНИХ ОБ'ЄКТІВ. КООРДИНАЦІЙНЕ КЕРУВАННЯ

Мета роботи – розглянути модель апаратурних об'єктів та координаційне керування

Загальні теоретичні відомості

У попередніх лабораторних роботах в межах моделі обладнання (equipmentmodel) установки приготування продукту для танків ми створили Апарати «Танк1» та «Танк2» (Рисунок 6.1) та модулі керування для реалізації виконавчих механізмів. У даній лабораторній роботі необхідно повністю реалізувати все процедурне керування, для чого виділимо усі процедурні елементи та обладнання (equipment).

Установка, що розглядається в лабораторній роботі 6, робить повністю партію певного продукту, отже вона з точки зору ISA-88 є технологічною коміркою (**Process Cell**). Враховуючи, що продукт може готуватися одночасно в двох танках, то в технологічній комірці може одночасно виконуватися приготування двох партій продукту.

Для обох танків використовується одна установка дозування, яка включає в себе два дозатори та клапан перемикання CMVdoz. З точки зору процедурного керування установка робить дві дії – дозування певної кількості компоненту А (Дозатор D1) та В (Дозатор D2). Ці технологічні дії найбільш підходять до функцій етапу. Тому в межах обладнання необхідно створити етапи: «Добавити А» та «Добавити В», в параметрах яких задавати необхідну кількість доз компоненту.

Тепер необхідно визначитися з тим, до якого типу обладнання відноситься установка дозування. Оскільки етап – це процедурне керування, він може виконуватися в Апараті, в Агрегаті або в Технологічній Комірці. Установка дозування найкраще підходить до поняття Агрегату (**EquipmentModel**), так як в ній не готовиться продукт і вона є допоміжним обладнанням. Самі дозатори зручно буде реалізувати у вигляді модулів керування, які будуть брати на себе усю логіку підрахунку кількості доз. Таким чином, два модулі керування дозаторами **CMDozator1** та **CMDozator2** в складі Агрегату **EMDozators** будуть входити до складу Технологічної комірки.

У моделі обладнання установка дозування не належить жодному танку, оскільки може використовуватися будь яким з них. З цього слідує декілька висновків:

- агрегат «Дозатори» (EMDozators) в моделі обладнання належить безпосередньо Технологічній Комірці;
- агрегат «Дозатори» є спільним ресурсом з ексклюзивним користуванням (exclusiveusedresource), так як може використовуватися будь яким танком, але не одночасно;
- перед користуванням дозаторів рецептам необхідно буде їх йому **надати** (**allocation**), щоб під час володіння ресурсом тільки один рецепт кори-

стувався ним;

- після надавання дозаторів у використання, клапан CMVdozповинен стати у відповідну позицію; звідси слідує, що даний клапан теж належить технологічній комірці а не танкам.

Діяльність **Allocation** відноситься до координаційного керування, яке теж необхідно реалізувати.

Загальна модель апаратурних об'єктів показана на Рисунок 6.2. Для спрощення на рисунку показані не всі об'єкти.

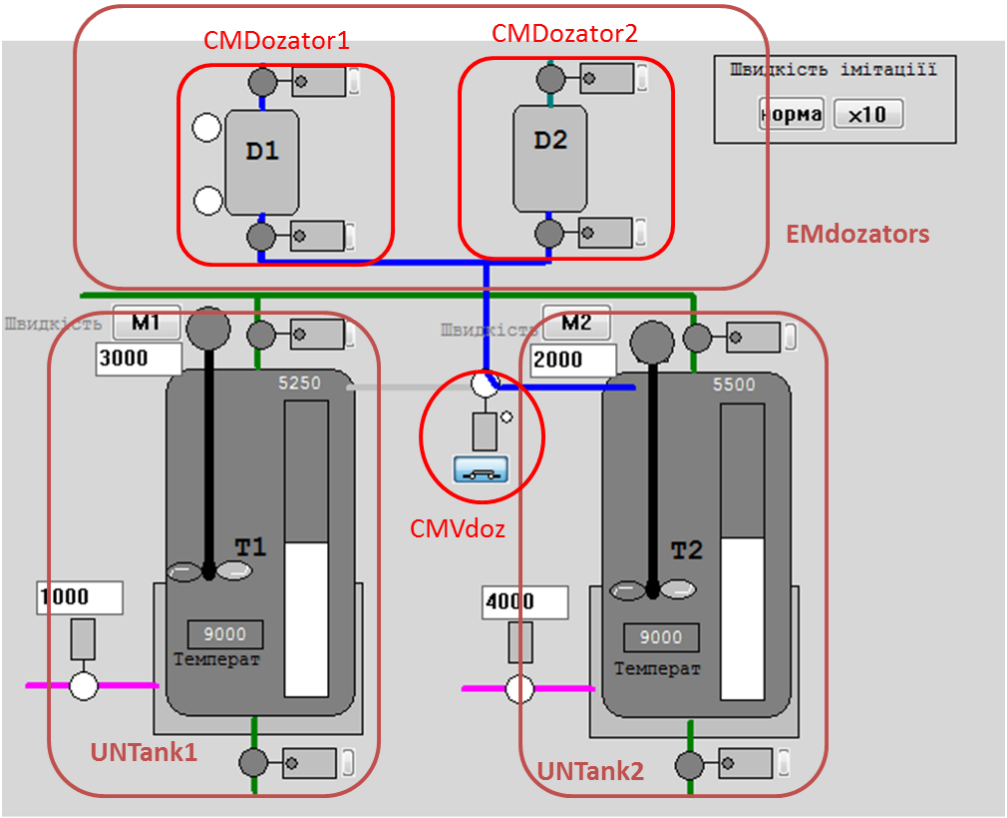


Рисунок 6.1 – Виділення Агрегату (EM)та Апаратів (UN) у технологічній комірці

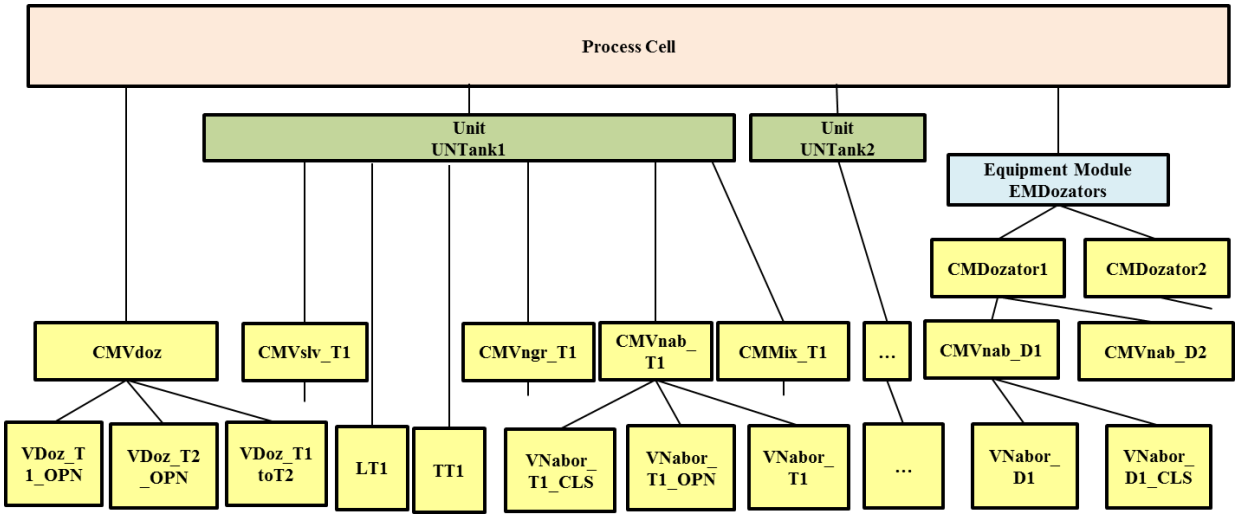


Рисунок 6.2 – Узагальнена модель апаратурних об'єктів (Equipment Entity Model)

Для можливості використання установки дозування в рецептах необхідно в межах Агрегату **EMDozators** створити етапи. Поки в явному створенні EMDozators в Unity Pro немає необхідності, тому достатньо створити етапи та викликати їх в програмі, аналогічно, як це було зроблено для етапів танків. Слід також відмітити, що в структурі означення компонувальних блоків BatchZenon немає інших типів апаратурних об'єктів, окрім Unit. Тим не менше, з точки зору означення рецепту немає різниці, як саме називається тип апаратурного об'єкту, оскільки користувач буде бачити тільки його назву. Іншими словами, як для Апаратів так і для Технологічних Комірок, і для Агрегатів в означенні компонувальних блоків Batch будуть використовуватися також Unit. Щоб користувач оперував зрозумілими для нього словами, замість EMDozators будемо використовувати назву агрегату «Дозатори».

До речі, у ISA-88.01-2010 Агрегати, етапи яких можуть безпосередньо викликатися з рецептів, називаються рецептно-обізнаними агрегатами (recipe-aware equipment module).

Для дозування сумішей з дозаторів в Танк 2 необхідно зробити перемикання 3-х ходового клапану. Враховуючи, що клапан належить технологічній комірці, саме вона повинна реалізувати таке перемикання. Це можна зробити через базове керування, або через явний виклик етапу, який буде частиною технологічної комірки. Вибір реалізації для таких задач залежить від об'єкту і вимог. У даній лабораторній роботі використаємо окремий етап «Прокласти шлях для дозування», в параметрах якого буде вказуватися номер танку, до якого необхідно перемкнути дозатори. Така реалізація дасть можливість в рецепті явно вказувати той момент, коли необхідно робити таке перемикання.Етап необхідно створити в межах технологічної комірки, так як згідно моделі апаратурних об'єктів (Рисунок 6.2) клапан належить саме їй.

Окрім керування необхідно реалізувати блокування запуску етапів та програм дозування без явного дозволу. Це загальноприйнята практика координаційного керування, коли запуск процедур повинно бути дозволено, або заборонено. Якщо оператор явно вибирає маршрут, то тільки після його прокладання можливий дозвіл на запуск дозаторів. У нашому випадку маршрутів для дозування буде лише два:

- дозатори –> танк Т1;
- дозатори –> танк Т2.

Вибір маршруту буде задаватися окремою змінною **DOZROUTE_SP**, яка буде приймати значення:

- 0 – маршрут дозування не вибраний, тому дозування блоковане;
- 1 – маршрут дозування на танк Т1, дозвіл дозування при стані 3-х ходового клапану на Т1 (аналогічно стану «закритий»);
- 2 – маршрут дозування на танк Т2, дозвіл дозування при стані 3-х ходового клапану на Т1 (аналогічно стану «відкритий»);
- інші значення – аналогічно нулю.

Блокування необхідно зробити як в програмі модулів керування дозаторів (**CMDozator**), так і в етапах дозування, наприклад в SCADA («Дозувати»). Для

CMDozator введемо додатковий біт стану **DOZENBL** (**STA.13**). Цей біт буде змінюватися ззовні **CM**, а саме в реалізації програми керування Технологічної Комірки. Цей же біт буде використовуватися для дозволу дозування в Zenon.

У ISA-88 описане вище керування, що не робить ніяких явних технологічних дій або керування обладнанням, а лише дозволяє/забороняє, змінює режим чи стан базового або процедурного керування прийнято називати Координаційним керуванням (Coordination Control).

Завдання до виконання лабораторної роботи

1. Створити CM для дозаторів (з боку ПЛК та SCADA) та перевірити їх роботу.
2. Створити EM для установки дозування(з боку ПЛК та SCADA) етапів для них, та реалізувати інтерфейс.
3. Створити етапи та апаратурні елементи для 2-го танку.
4. Забезпечити блокування етапів дозування при невірному стані 3-ходового клапану перемикання дозатору на танки.
5. Реалізувати перемикання дозаторів на необхідний танк в залежності від вибраного маршруту.
6. Реалізувати єдиний рецепт для приготування продукту у будь-якому вільному танку.

Порядок проведення роботи

6.1 Створення CM та інтерфейсу для дозаторів

6.1.1 У Unity Pro створіть тип та змінні для керування та контролю дозаторами.

- Завантажте Unity Pro та проект з минулої лабораторної роботи.
- Створіть тип **CM_Dozator** для контролю та керування дозаторами (Рисунок 6.3).
- Створіть дві змінні для **CM** дозаторів з прив’язкою до адрес (Рисунок 6.3).

CM_Dozator	<Struct>	
STA	INT	крок/стани
CMD	INT	команди
DozesSP	INT	задана кількість доз
DozesPV	INT	дійсна кількість доз
CMDozator1	CM_Dozator	%MW142
CMDozator2	CM_Dozator	%MW146

Рисунок 6.3 – Тип та змінні в Unity Pro для керування/контролю дозування

6.1.2 У Unity Pro створіть **DFB** тип, екземпляри та виклики для керування та контролю дозаторами.

- Створіть **DFB** тип **cmDozator** для контролю та керування дозато-

рами (Рисунок 6.4).

- Скопіюйте код з таблиці 6.1 в програму типу cmDozator.
- Створіть один екземпляр CMDozator типу cmDozator.

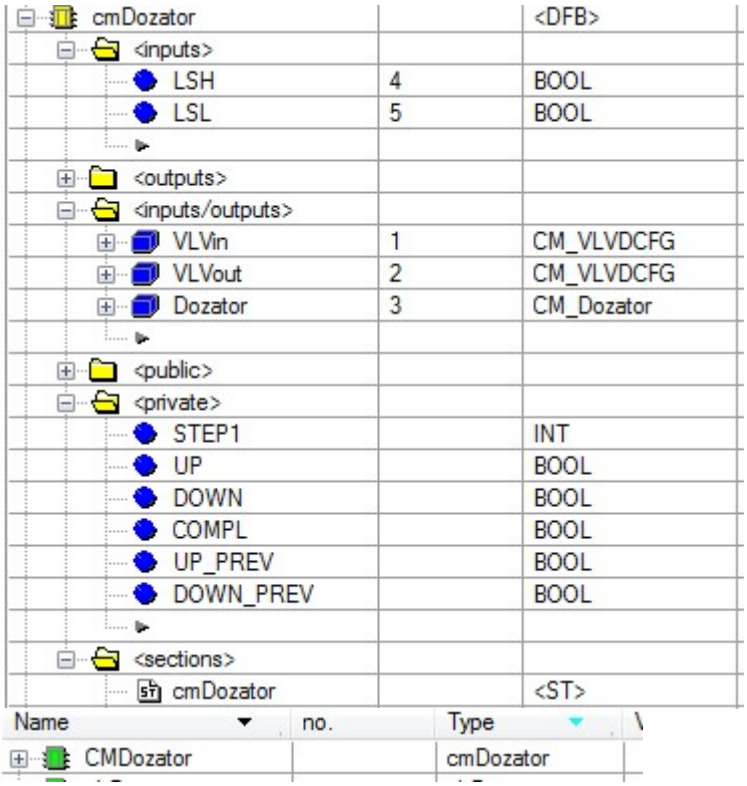


Рисунок 6.4 – Тип та екземпляр DFBдля реалізації СМ дозатору

Таблиця 6.1 – Код для секції **DFB** типу **cmDozator**

```
(*плинний статус дозатору - розпаковка бітів*)
STEP1:=Dozator.STA and 16#000F;(*тільки молодший байт використовується для
значення кроку*)
UP:=Dozator.STA.8;(*набирається*)
DOWN:=Dozator.STA.9;(*зливається*)
COMPL:=Dozator.STA.10;(*кількість доз досягнута*)
UP_PREV:=Dozator.STA.11;(*стан набору до зупинки*)
DOWN_PREV:=Dozator.STA.12;(*стан зливу до зупинки*)

if Dozator.CMD=2 then(*припинити дозування*)
     step1:=1;
     (*запамятати стан*)
     UP_PREV:=UP;DOWN_PREV:=DOWN;
end_if;
if Dozator.CMD=3 then step1:=0; end_if; (*ініціалізувати дозатор*)

if dozator.DozesPV < Dozator.DozesSP then
     COMPL:=false;
end_if;

CASE STEP1 OF
       0: (*інціалізація/невизначений стан*)
           dozator.DozesPV:=0;
           UP_PREV:=false;DOWN_PREV:=false;COMPL:=false;
           STEP1:=1;
       1: (*дозування припинено*)
           VLVin.cmd:=2;
           VLVout.cmd:=2;
```

```
            if Dozator.CMD=1 then (*продовжити/почати дозування*)
              if LSH then step1:=3; (*повний*)
              elsif NOT LSL then step1:=2; (*порожній*)
              else (*наповнюється або зливаться залежить від попереднього ста-
ну*)
                  if UP_PREV then step1:=4;
                  else step1:=5;
                  end_if;
              end_if;
            end_if;
        2: (*порожній*)
        if VLVout.STA.8 then (*якщо вихідний клапан закритий*)
            if dozator.DozesPV >= Dozator.DozesSP then
               COMPL:=true; (*кількість доз досягнута*)
               step1:=1;
            else
               step1:=4;
            end_if;
        end_if;
        3: (*повний*)
        if VLVin.STA.8 then  (*якщо вхідний клапан закритий*)
            step1:=5;
            dozator.DozesPV:=dozator.DozesPV+1; (* +1 доза*)
        end_if;
        4: (*наповнюється*)
         VLVin.cmd:=1;
        if LSH then
            step1:=3;
            VLVin.cmd:=2;
        end_if;
        5: (*зливається*)
         VLVout.cmd:=1;
        if not LSL then
            step1:=2;
            VLVout.cmd:=2;
        end_if;
        ELSE (*ініціалізація*)
        STEP1:=0;
END_CASE;

(*блокування: закриття клапанів по датчику на будь якому кроці*)
if LSH then VLVin.cmd:=2; end_if;
if not LSL then VLVout.cmd:=2; end_if;

UP:=(STEP1=4);
DOWN:=(STEP1=5);

Dozator.CMD:=0; (*обнулити команду*)

(*плинний статус дозатора - упаковка бітів*)
Dozator.STA:=0;
Dozator.STA.8:=UP;(*наповнюється*)
Dozator.STA.9:=DOWN;(*зливається*)
Dozator.STA.10:=COMPL;(*кількість доз досягнута*)
Dozator.STA.11:=UP_PREV;(*стан набору до зупинки*)
Dozator.STA.12:=DOWN_PREV;(*стан зливу до зупинки*)
Dozator.STA := Dozator.STA or STEP1;
```

- У основній задачі (MAST) створіть секцію Dozators, та запишіть виклики **CM** для дозаторів (Рисунок 6.5).
- Скомпілюйте проект UnityPro завантажте в імітатор ПЛК та запустіть на виконання.

```
CMDozator (VLVin := CMVnab_D1, VLVout := CMVslv_D1, Dozator := CMDozator1, LSH := LSH_D1, LSL := LSL_D1);
CMDozator (VLVin := CMVnab_D2, VLVout := CMVslv_D2, Dozator := CMDozator2, LSH := LSH_D2, LSL := LSL_D2);
```

Рисунок 6.5 – Програма виклику CM дозаторів в секції Dozators

6.1.3 У SCADA zenon створіть типи, змінні та ЛМІ для реалізації керування та контролю дозаторів.

- Запустіть на виконання середовище розробки zenon та завантажте проект з минулої лабораторної роботи.
- Створіть структурний тип змінної з іменем **CM_Dozator** та налаштуйте параметри, як показано на Рисунок 6.6.
- Створіть змінні **Dozator1** та **Dozator2** типу **CM_Dozator** та налаштуйте їх як показано на Рисунок 6.7.
- В контекстному меню редактору **zenon Screens** виберіть команду **Import XML** і імпортуйте символи «Символ для стану дозатору.XML» та «Символ для керування дозатором.XML».
- Розмістіть та налаштуйте імпортовані символи, як це показано на Рисунку 6.8. Для дозатору 2 налаштуйте також правило прив'язки.
- За допомогою елементів Numerical Value виведіть біля дозаторів значення уставки та плинної кількості доз для дозатору.

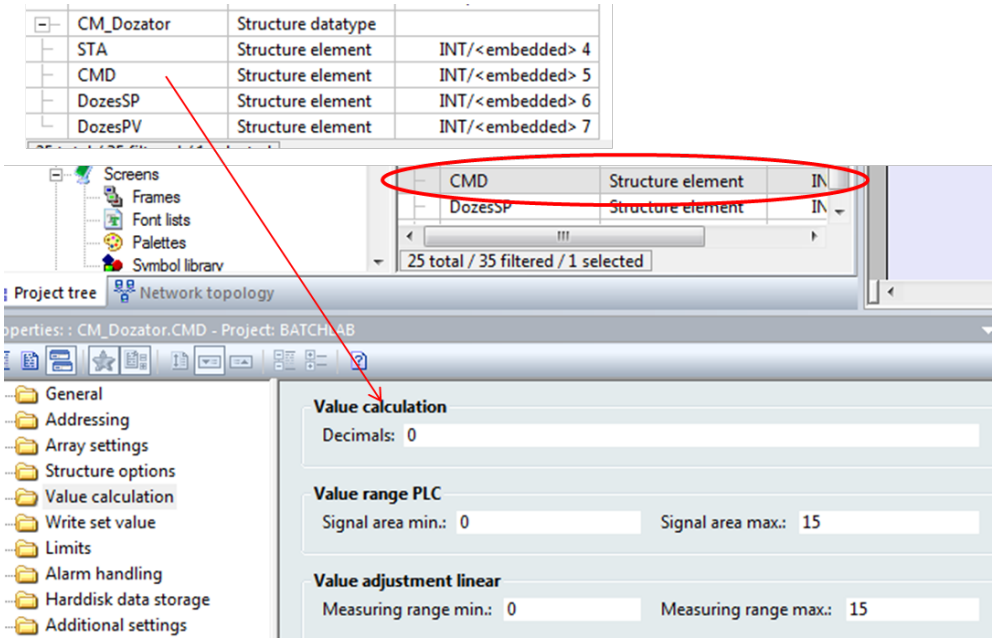


Рисунок 6.6 – Створення та налаштування в Zenonтипу CM_Dozator

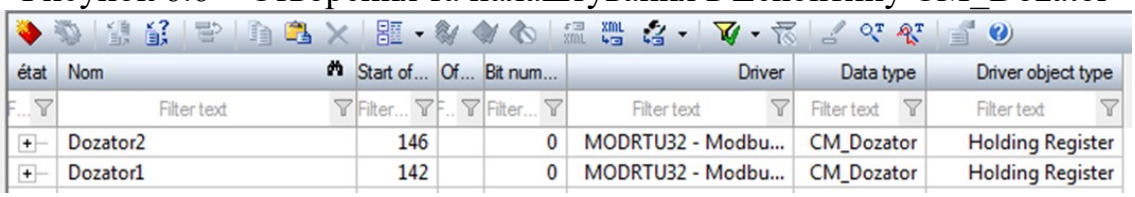


Рисунок 6.7 – Створення та налаштування в Zenon змінних CM_Dozator

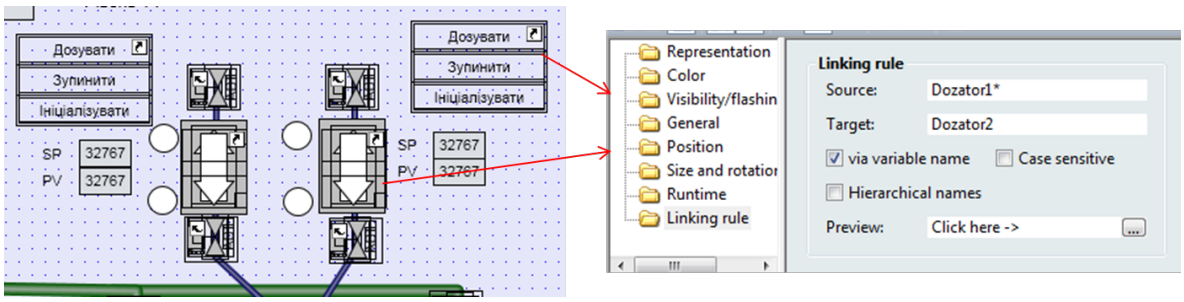


Рисунок 6.8 – Створення та налаштування в zenon ЛМІ для керування та контролю дозаторів

6.1.4 Перевірте роботу системи дозування в середовищі виконання zenon.

- Скомпілюйте проект zenon та запустіть його на виконання.
- В середовищі виконання перейдіть на екран Batch та для кожного дозатору виставте задане значення 3 та 2 відповідно.
- Для кожного дозатору натисніть команду «ініціалізувати».
- Використовуючи команди «дозувати», запустіть дозатори на дозування, дочекайтеся відпрацювання доз, після чого знову скиньте лічильники доз командою «ініціалізувати».
- На дозаторі 1 перевірте роботу команди «зупинити» та «дозувати» при наборі та зливі дози; зверніть увагу чи продовжує дозатор виконувати правильні дії зливу/набору та підрахунку доз.
- За бажанням, проаналізуйте роботу програми дозування в UnityPro, за необхідності спитайте викладача, щодо незрозумілої реалізації коду.

6.2 Створення EM, етапів та інтерфейсу для установки дозування

6.2.1 У Unity Pro створіть **DFB**-тип **phDoze** та один екземпляр цього типу для реалізації етапу «Дозувати».

- Створіть **DFB**-тип **phDoze**, як це показано на Рисунку 6.9. Код програми для секції **phDoze** скопіюйте з таблиці 6.2.
- Створіть однойменний екземпляр для типу **phDoze** (Рисунку 6.9).

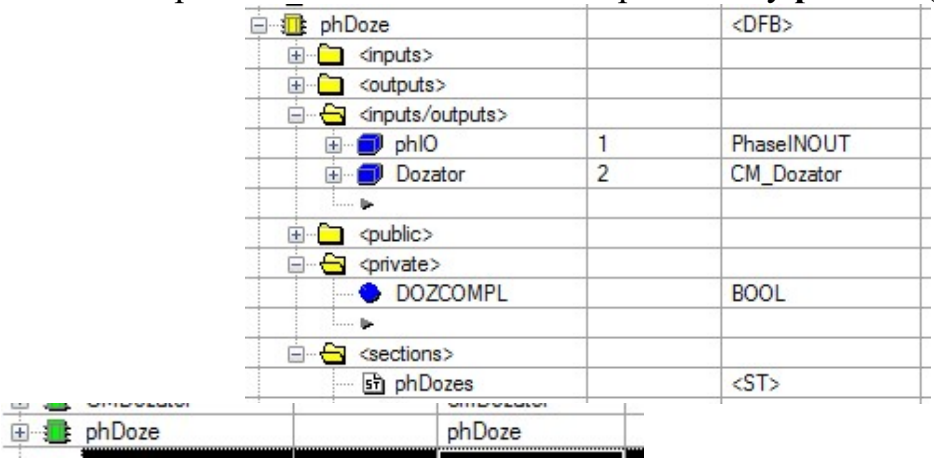


Рисунок 6.9 – Створення типу **DFB** для етапу «Дозувати»

Таблиця 6.2 – Код для секції **DFB** типу **cmDozator**

```
(*обробка команд керування автоматом станів етапу phDRAIN*)
CASE phIO.CMD OF
  1: (*Phase started*)
    phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3: (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4: (*Phase deactivated*)
    phIO.STA:=0;
    Dozator.CMD:=3; (*ініціалізувати дозатор*)
  10: (*Status change: Pausing*)
    phIO.STA:=5;
  11: (*Status change: Resuming*)
    phIO.STA:=1;
  12: (*Status change: Holding*)
    phIO.STA:=7;
  13: (*Status change: Restarting*)
    phIO.STA:=11;
  14: (*Status change: Stopping*)
    phIO.STA:=3;
  15: (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
  36:; (*Maximum execution period exceeded*)
  37:; (*Waiting period following condition exceeded*)
  38:; (*Phase started multiple times*)
  ELSE
      ;
END_CASE;
phIO.CMD:=0;(*після обробки команда обнуляється*)


CASE phIO.STA OF
  0:;(*Idle*)

  1:;(*Running*)
     Dozator.CMD:=1; (*продовжити/почати дозування*)
     if dozator.STA.10 then (*кількість доз досягнута*)
           phIO.STA:=12;
     end_if;
  2:;(*Complete (Executed)*)
     Dozator.CMD:=2; (*припинити дозування*)
  3:;(*Stopping*)
     Dozator.CMD:=2; (*припинити дозування*)
     phIO.STA:=4;
  4:;(*Stopped*)
  5:;(*Pausing*)
     Dozator.CMD:=2; (*припинити дозування*)
     phIO.STA:=6;
  6:;(*Paused*)
  7:;(*Holding*)
     Dozator.CMD:=2; (*припинити дозування*)
     phIO.STA:=8;
```

```
   8:;(*Held*)
   9:;(*Aborting*)
      Dozator.CMD:=2; (*припинити дозування*)
      phIO.STA:=10;
   10:;(*Aborted*)
   11:;(*Restarting*)
      phIO.STA:=1;
   12: (*Completed*)
      Dozator.CMD:=2; (*припинити дозування*)
   ELSE
    (*якщо значення відрізняється від
     доступних - перехід в Idle*)
     phIO.STA:=0;Dozator.CMD:=3; (*ініціалізувати дозатор*)
END_CASE;
```

6.2.2 У Unity Pro створіть інтерфейсні змінні **phioDOZE_A** та **phioDOZE_B** типу **PhaseINOUT** та реалізуйте виклик етапів.

- У Unity Pro створіть інтерфейсні змінні **phioDOZE_A** та **phioDOZE_B** типу **PhaseINOUT** та прив'яжіть їх до адрес, як це показано на Рис.6.10.
- У секції Dozators запишіть виклики **phDoze** з потрібними аргументами для кожного етапу (Рисунок 6.10).

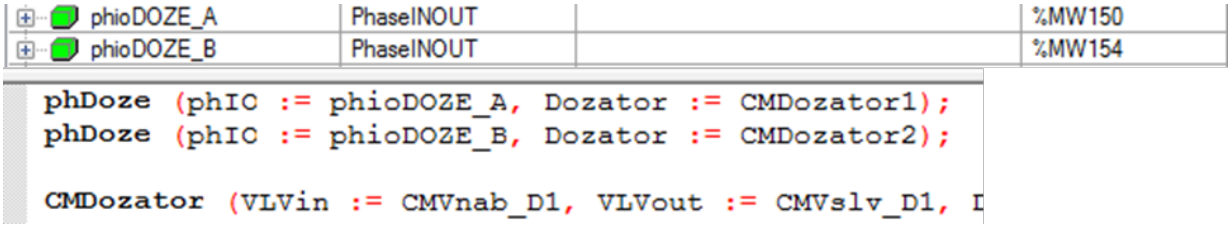


Рисунок 6.10 – Створення змінних та програми виклику етапів

6.2.3 У SCADA zenon створіть змінні та компонувальні блоки для етапів «Добавити А» та «Добавити В».

- У SCADA zenon створіть інтерфейсні змінні для етапів дозування, як це показано на Рисунок 6.11.
- В Batch Control створіть Unit «Дозатори», який буде відповідати з Агрегат установки дозування.
- Скопіюйте з Апарату «Танк1» в Unit «Дозатори» етап «Вивантажити», змініть назву на «Добавити А», перейменуйте параметр «Заданий нижній рівень» на «Задана кількість», а «Рівень» на «Дійсна кількість», прив'яжіть параметри до потрібних змінних (Рисунок 6.12).

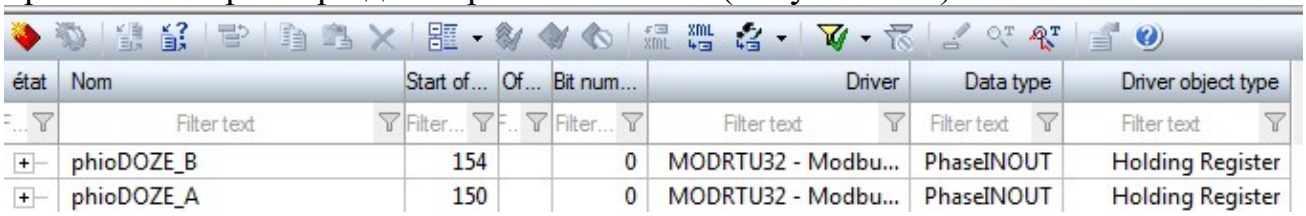


Рисунок 6.11 – Створення змінних та програми виклику етапів

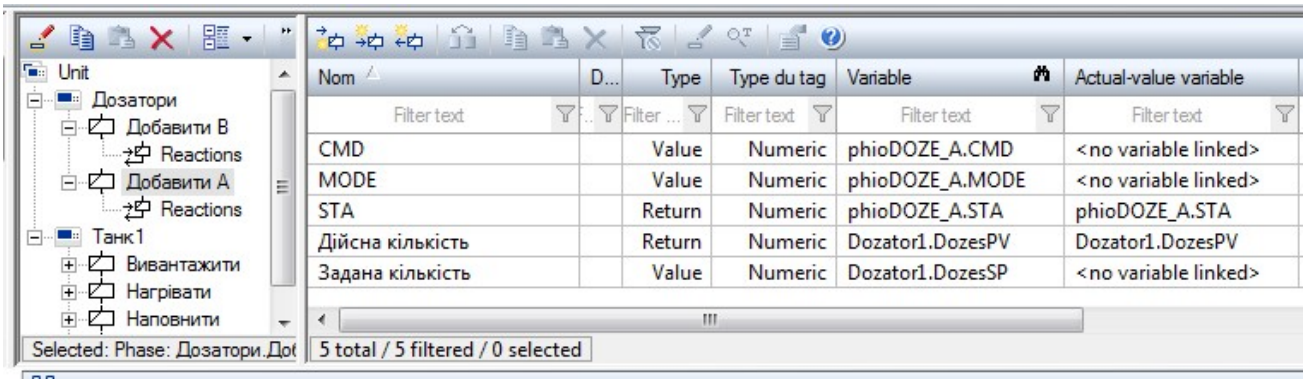


Рисунок 6.12 – Створення компонувальних блоків-етапів для дозування

- Для етапу «Добавити А» змініть умову завершення етапу на виконання рівності параметру **STA=12** (Рисунок 6.13); стан рівний 12 – це стан **Completed**, таким чином, коли кількість доз буде досягнутою, програма етапу в ПЛК дасть знати про це відображення цього етапу в SCADA.

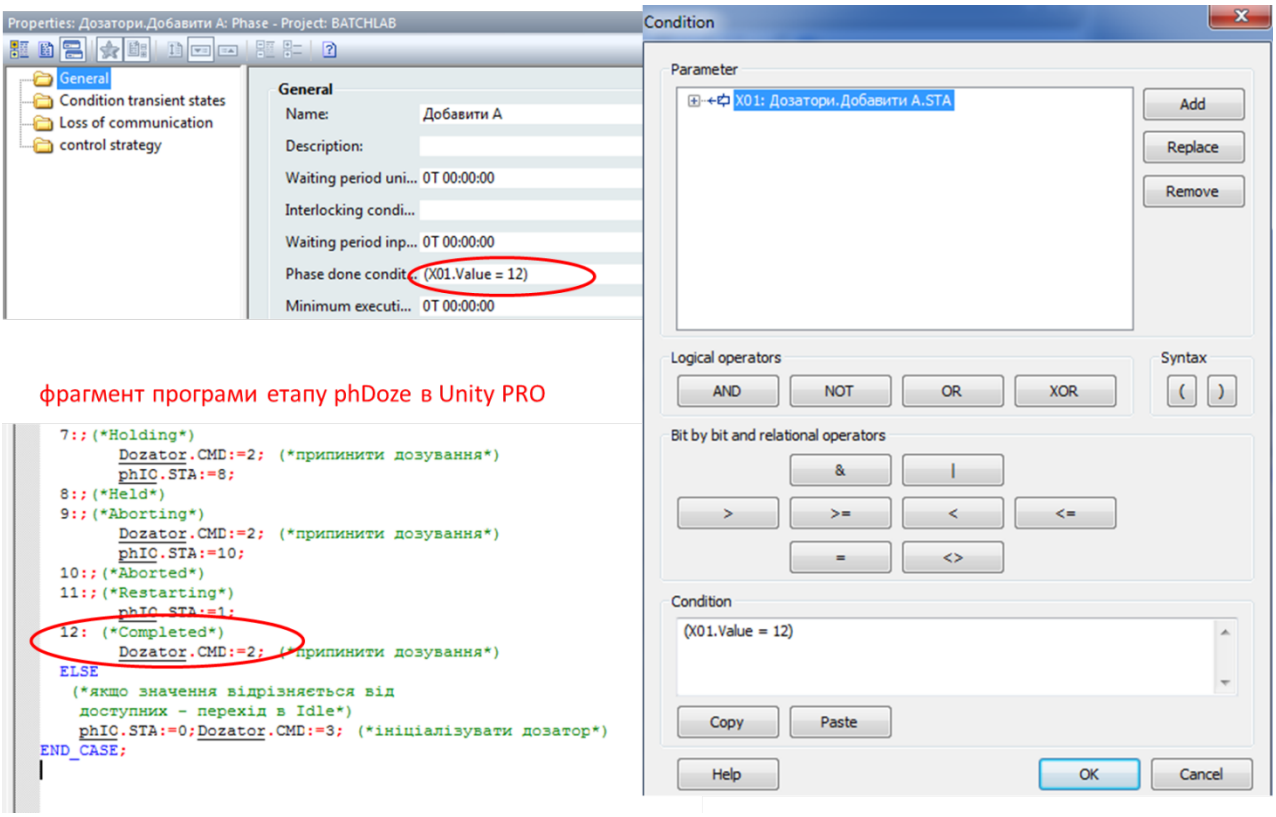


Рисунок 6.13 – Зміна умови завершення етапу

- Скопіюйте етап «Добавити А», перейменуйте в «Добавити В» та налаштуйте його аналогічно для роботи з іншим дозатором.

6.2.4 У середовищі виконання SCADA zenon створіть та перевірте роботу рецепту приготування продукту «Коктейль 1».

- Скомпілюйте проект SCADA zenon та запустіть його на виконання.
- Відкрийте «Екран Batch» і створіть новий майстер рецепт типу PFCs назвою «Коктейль 1», як показано на Рисунку 6.14.

- Зробіть наступні налаштування рецептурних блоків (таблиця 6.3).

Таблиця 6.3 – Налаштування елементів майстер рецепту «Коктейль 1»

Елемент	Налаштування	Примітка
етап «Наповнити»	MaximumExecutionDuration = 00:02:00 Заданий рівень наповнення = 70%	
етап «Добавити А»	Задана кількість = 2	
етап «Добавити В»	Задана кількість = 3	
етап «Перемішувати»	Задана швидкість перемішування = 20% Minimumexecutionduration = 00:01:00	
етап «Нагрівати»	Задана потужність нагрівання = 80%	
Умова переходу після «Нагрівати»	Танк1.Нагрівати.Температура>35	
етап «Вивантажити»	Заданий рівень = 1% (Мін=0% Макс=100%)	

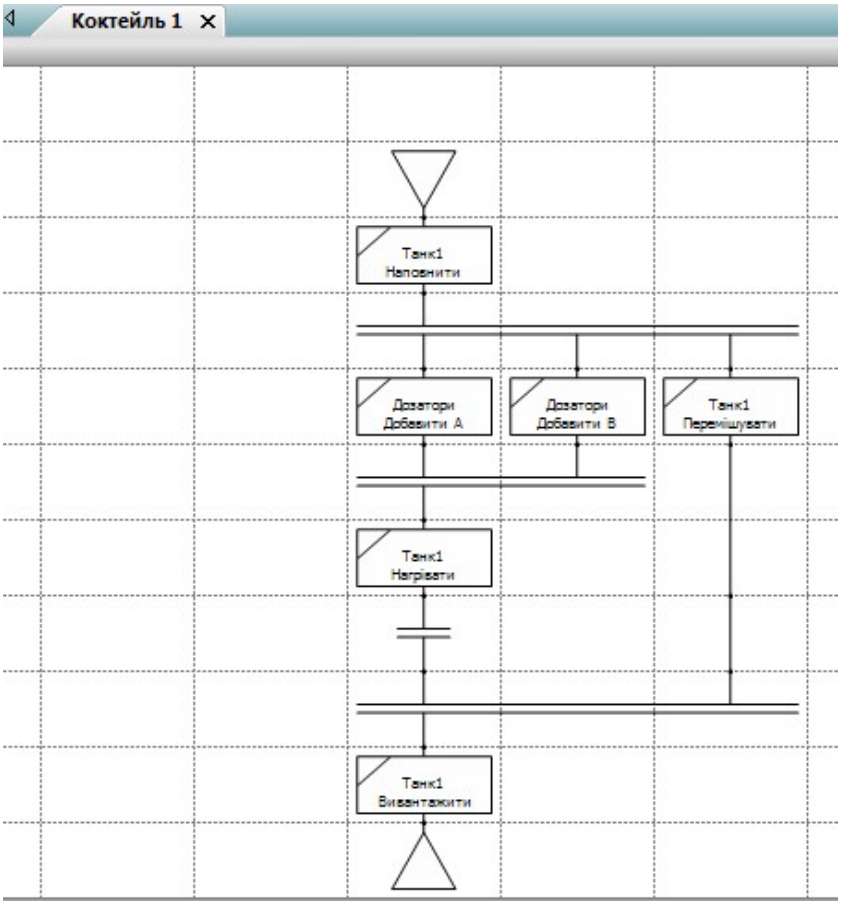


Рисунок 6.14 – Вигляд майстер-рецепту «Коктейль 1»

6.3 Створення етапів, компонувальних блоків для 2-го Танку

До сих пір етапи були створені тільки для 1-го танку. Тепер створимо етапи для 2-го танку.

6.3.1 Перейменуйте змінні для посилання їх на 1-й танк та добавте змінні для 2-го танку.

- Існуючі змінні, що відповідають за інтерфейс для етапів в UnityProта SCADA zenon перейменуйте з суфіксом T1 (Рисунок 6.15).

phioDRAIN_T1	PhaseINOUT				%MW100	
phioFILL_T1	PhaseINOUT				%MW104	
phioHEAT_T1	PhaseINOUT				%MW112	
phioMIX_T1	PhaseINOUT				%MW108	
phioHEAT_T1	112		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioMIX_T1	108		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioFILL_T1	104		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioDRAIN_T1	100		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register

Рисунок 6.15 – Перейменовані змінні в UnityProта SCADA zenon

- Добавте в UnityPro інтерфейсні змінні для етапів 2-го танку, скопіюйте змінні для 1-го танку, перейменуйте їх та прив’яжіть до нових адрес (Рисунок 6.16).
- Аналогічні дії зробіть в zenon.

phioDRAIN_T2	PhaseINOUT				%MW158	
phioFILL_T2	PhaseINOUT				%MW162	
phioMIX_T2	PhaseINOUT				%MW166	
phioHEAT_T2	PhaseINOUT				%MW170	
phioDRAIN_T2	158		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioFILL_T2	162		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioHEAT_T2	170		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register
phioMIX_T2	166		0	MODRTU32 - Modbu...	PhaseINOUT	Holding Register

Рисунок 6.16 – Інтерфейсні змінні етапів для 2-го танку

- Створіть в zenon змінні, яких не вистачає для керування та контролю за 2-м танком (Рисунок 6.17).

Примітка. Звертайте увагу на однаковість найменування в zenon, наприклад «T1» пишіть у кожній змінній на латиниці, а не кирилиці! Це вплине на автоматизацію заміни та функціонування правил прив’язки.

Заданий рівень наповнення T2		22	0
Заданий рівень наповнення T1		0	0
Заданий нижній рівень T2		18	0
Заданий нижній рівень T1		4	0
Задана швидкість перемішування T2		20	0
Задана швидкість перемішування T1		24	0
Задана потужність нагрівання T2		32	0
Задана потужність нагрівання T1		30	0

Рисунок 6.17 – Додаткові змінні в zenon для контролю та керування 2-м танком

6.3.2 Реалізуйте процедурне керування, будівельні блоки для 2-го танку та перевірте його роботу. Змініть зображення клапану перемикання дозатору на танки.

- У Unity Pro створіть секцію «Tank2», скопіюйте туди вміст секції «Tank1» та модифікуйте її для керування 2-м танком.
- В SCADA zenon в Batch Control скопіюйте Unit «Танк 1» та перейменуйте його в «Танк 2».
- Для скопійованого Апарату в контекстному меню виберіть команду «Replace linking in phases…» та замініть прив'язки з Т1 на Т2, використовуючи «Accept» та після перевірки замін – натисніть «Ok».

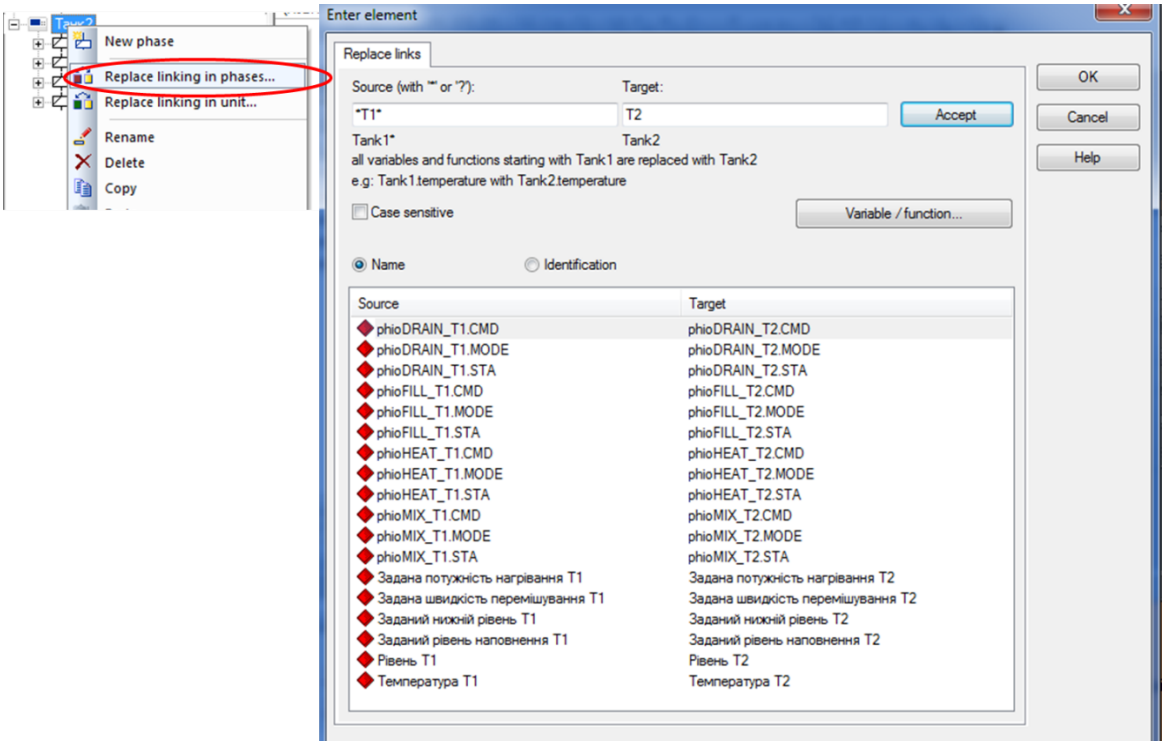


Рисунок 6.18 – Зміна зв'язків для Unit «Танк2»

- Для кращого відображення стану клапану перемикання дози, в контекстному меню редактору zenon Screens виберіть команду **Import XML** і імпортуйте символ «**Сивмол VLVDoz.XML**».
- Розмістіть символ (як лінкований) на місці зображення клапану перемикання дозатору з танку на танк на сторінці «**Екран Batch**», старе зображення видаліть.
- Замініть прив'язку до змінної відповідно до призначення символу. Для цього скористайтеся вкладкою символу «**Linking rule**» та кнопкою «**Preview**» для перевірки результату заміни.
- Скомпілюйте проекти для ПЛК та SCADA та запустіть їх на виконання.
- Скопіюйте майстер рецепт «Коктейль 1» та перейдіть в режим його модифікації; використовуючи символ «**Change Phase/Operation**»  змініть етапи з Танку 1 на аналогічні з Танку 2; для зручності можна використати

можливість упорядкування та фільтрації за полем «Обладнання».

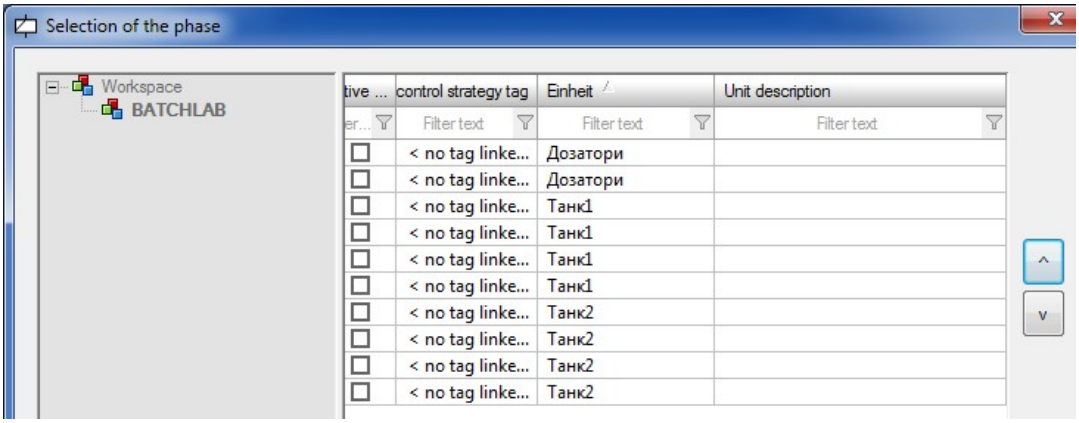


Рисунок 6.19 – Використання можливості групування та фільтрації при виборі етапу

- Перевірте роботу рецепту в режимі тесту, зверніть увагу на стан клапану перемикання дозатору. Дайте відповідь на запитання, ***чому клапан перемикання дозатору автоматично не переключається?***

6.4 Створення етапу для прокладання шляху для дозування в таки

6.4.1 У Unity Pro та zenon створіть змінну заданого маршруту для дозування, та виведіть їх на екран.

- Створіть змінні в Unity Pro та zenon, як це показано на Рисунку 6.20.
- Виведіть його на сторінку екрану Batch з можливістю введення значення.

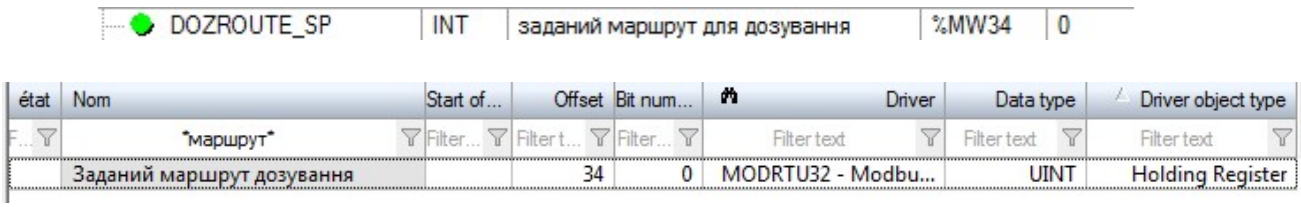


Рисунок 6.20 – Змінні заданого маршруту дозування

6.4.2 У Unity Pro реалізуйте блокування модулів керування дозування при невідповідності заданого маршруту дійсному положенню клапану.

- В Unity Pro створіть секцію Production, яка буде реалізовувати логіку обробки Технологічної Комірки, введіть туди код програми, що показана на Рисунок 6.21.

```
(*дозвіл керування дозаторами*)
                (*якщо клапан закритий (дозатор на Т1) і вибраний шлях на Т1*)
CMDozator1.STA.13:=(CMVdoz_HMI.STA.8 and DOZROUTE_SP=1)
                (*або якщо клапан відкритий (дозатор на Т2) і вибраний шлях на Т2 *)
                   or(CMVdoz_HMI.STA.7 and DOZROUTE_SP=2);
(*дозвіл для 2-го дозатору надається за тими ж правилами*)
CMDozator2.STA.13:= CMDozator1.STA.13;
```

Рисунок 6.21 – Код для секції Production

- Змініть **DFB**-тип для дозаторів **cmDozator** так, щоб команда запуску не працювала при відсутності дозволу на дозування (Рисунок 6.22), а також, щоб дозатор зупинявся при відсутності дозволу на дозування.

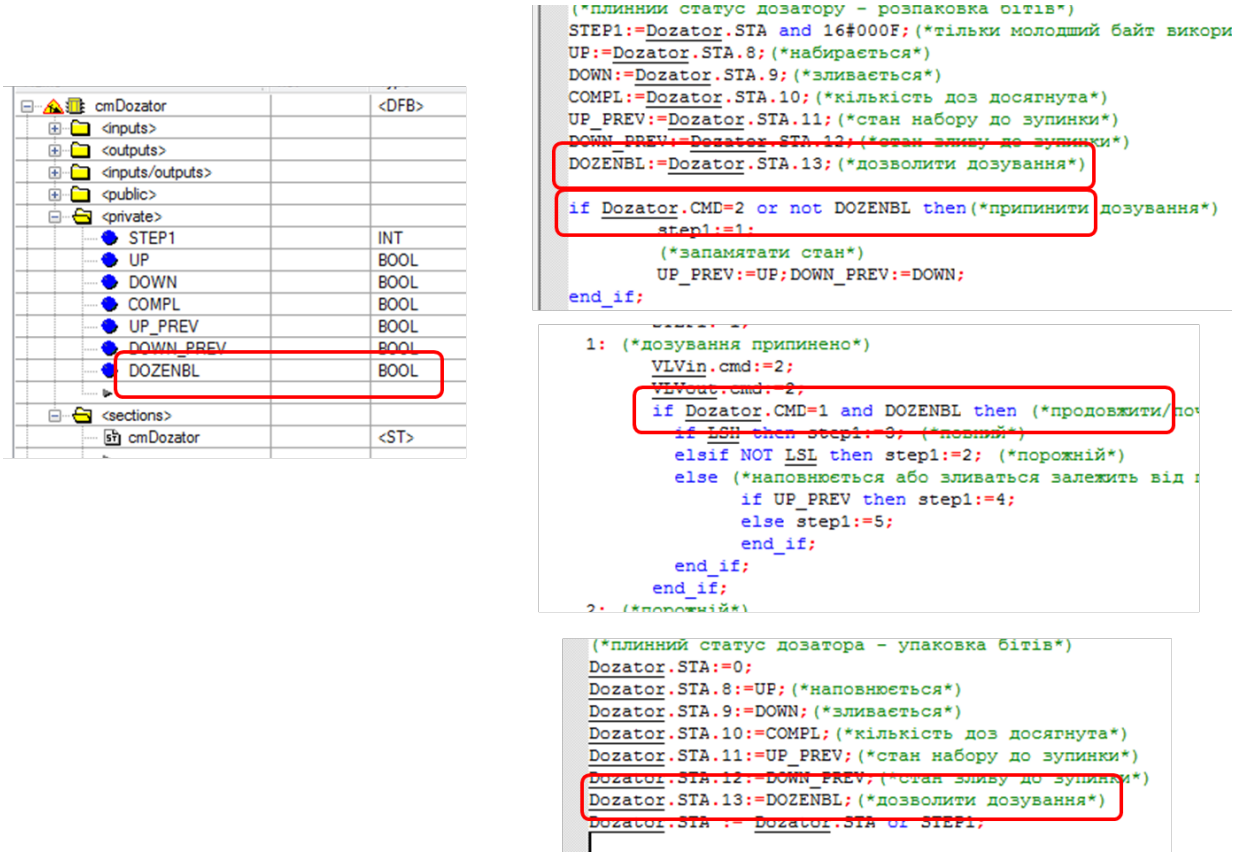


Рисунок 6.22 – Зміна типу **DFB cmDozator**

- Зробіть так, щоб на екрані Batch біля кожного дозатору висвічувався напис «**ДОЗВ**» при наявності дозволу дозування та «**БЛОК**» при відсутності(можна скористуватися Combi Element).
- Скомпілюйте проекти zenon та Unity Pro запустіть їх на виконання.
- Спробуйте різні комбінації заданого значення маршрутів та положення клапану перемикання дозатору (зміна можлива в ручному режимі), подивіться на відображення індикатору блокування та реагування на команди дозатору.

6.4.3 У zenon реалізуйте логіку блокування етапів запуску дозування при невідповідності заданого маршруту положенню клапану.

- В zenon для етапів «Добавити А» та «Добавити В» добавте параметри стану дозаторів (Рисунок 6.23).

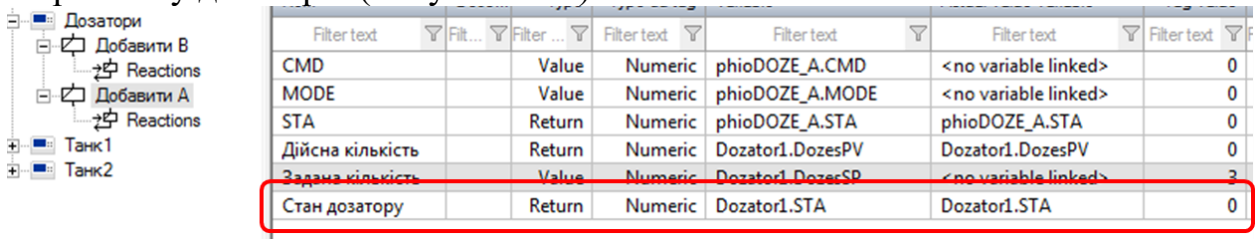


Рисунок 6.23 – Налаштування умови блокування для запуску етапу «Добавити А»

- Для цих же етапів добавте умову дозволу виконання (Рисунок 6.24).
- Зробіть компіляцію проекту, перезапустіть середовище виконання Zenon, запустіть рецепт «Коктейль 1» або «Коктейль 2» в режимі тестування і перевірте роботу рецептів при наявності та відсутності дозволу на дозування.

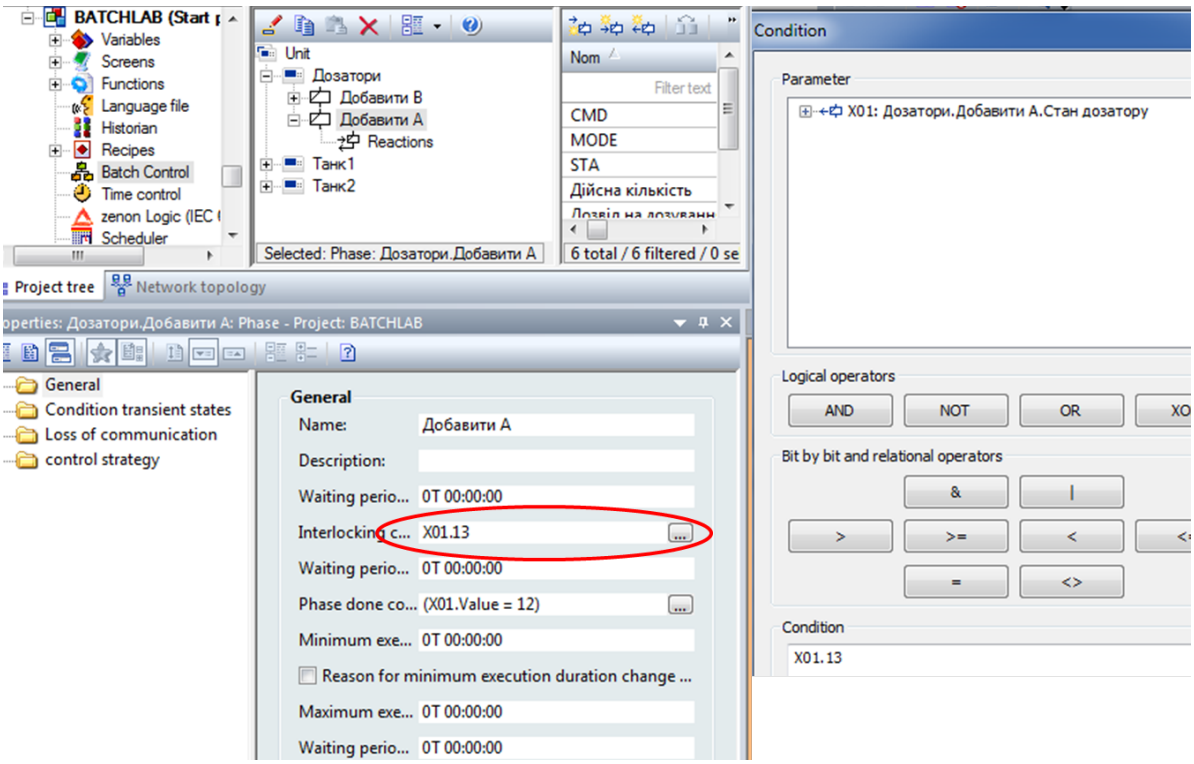


Рисунок 6.24 – Налаштування умови блокування для запуску етапу «Добавити А»

6.4.4 Реалізуйте етап прокладення маршруту дозування до танків DOZROUTE_SP та включіть його в рецепти.

- В Unity Pro та zenon створіть інтерфейсну змінну **phioDOZROUTE**, типи та екземпляр **DFB** для етапу **phDOZROUTE** (Рисунок 6.24); лістинг коду скопіюйте з таблиці 6.4; зверніть увагу, що в командах з'явилася команда при активації етапу, при якій обнуляється попереднє значення маршруту – це потрібно для запобігання спрацювання умови виконання прокладеності маршруту ще до присвоєння уставки з SCADA.

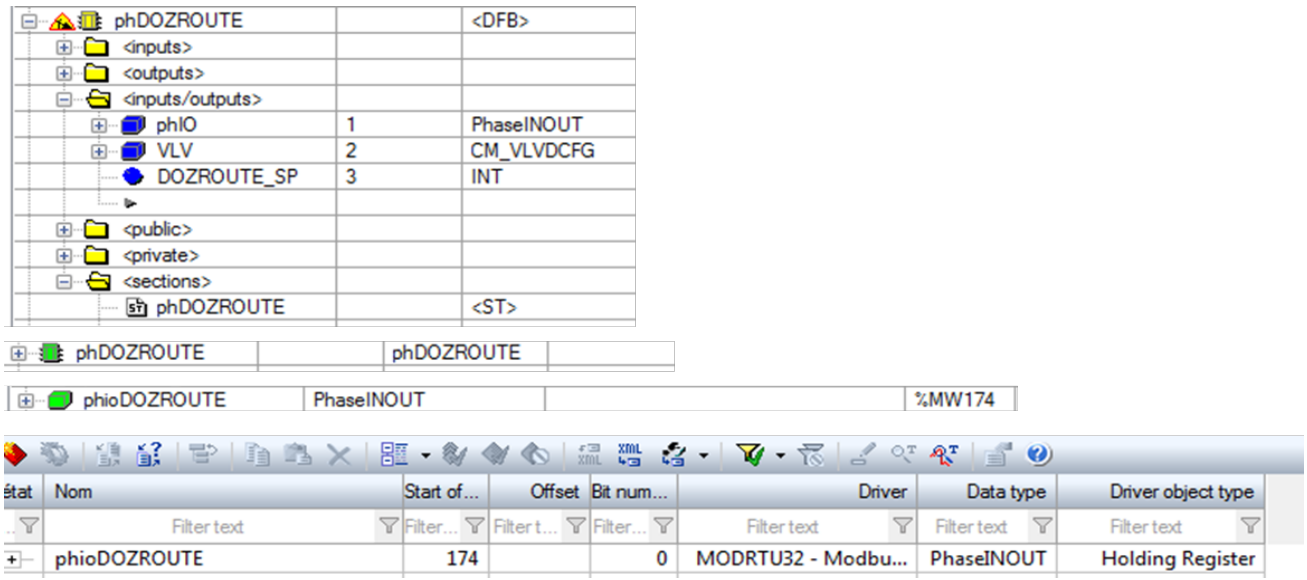


Рисунок 6.25 – Створення інтерфейс них змінних,DFB-типу та екземпляру для **phDOZROUTE**

Таблиця 6.4 – Код для секції DFB-типу етапу phDOZROUTE

```
CASE phIO.CMD OF
  1:;(*Phase started*)
       phIO.STA:=1;
  2:;(*Finished writing command tags*)
  3: (*Phase finished: Phase done condition fulfilled and Minimum execution
duration reached (if engineered)*)
    phIO.STA:=3;
  4: (*Phase deactivated*)
    phIO.STA:=0;
  5: (*Phase activated*)
    DOZROUTE_SP:=0;
  10: (*Status change: Pausing*)
    phIO.STA:=5;
  11: (*Status change: Resuming*)
    phIO.STA:=1;
  12: (*Status change: Holding*)
    phIO.STA:=7;
  13: (*Status change: Restarting*)
    phIO.STA:=11;
  14: (*Status change: Stopping*)
    phIO.STA:=3;
  15: (*Status change: Aborting*)
    phIO.STA:=9;
  20:; (*Mode change: Automatical*)
  21:; (*Mode change: Semi-automatic*)
  22:; (*Mode change: Manual*)
  30:; (*Exit Runtime initiated*)
  31:; (*Runtime restart*)
  32:; (*Unit allocation not possible*)
  33:; (*Waiting period unit allocation exceeded*)
  34:; (*Input interlocking blocked*)
  35:; (*Waiting period input interlocking exceeded*)
  36:; (*Maximum execution period exceeded*)
  37:; (*Waiting period following condition exceeded*)
  38:; (*Phase started multiple times*)
  ELSE
      ;
END_CASE;
phIO.CMD:=0;(*після обробки команда обнуляється*)
```

```
CASE phIO.STA OF
  0:;(*Idle*)

  1:;(*Running*)
     if DOZROUTE_SP=1 then (*маршрут на 1-й танк*)
           VLV.CMD:=2;
           if VLV.STA.8 then phIO.STA:=12;end_if; (*маршрут прокладено*)
     elsif DOZROUTE_SP=2 then(*маршрут на 2-й танк*)
           VLV.CMD:=1;
           if VLV.STA.7 then phIO.STA:=12;end_if; (*маршрут прокладено*)
     end_if;
  2:;(*Complete (Executed)*)
  3:;(*Stopping*)
     phIO.STA:=4;
  4:;(*Stopped*)
  5:;(*Pausing*)
     phIO.STA:=6;
  6:;(*Paused*)
  7:;(*Holding*)
     phIO.STA:=8;
  8:;(*Held*)
  9:;(*Aborting*)
     phIO.STA:=10;
  10:;(*Aborted*)
  11:;(*Restarting*)
     phIO.STA:=1;
  12:; (*Completed*)
  ELSE
   (*якщо значення відрізняється від
    доступних - перехід в Idle*)
    phIO.STA:=0;
    DOZROUTE_SP:=0; (*скинути завдання маршруту для уникнення випадкового до-
зування*)
END_CASE;
```

- В Unity Pro реалізуйте виклики етапів, як це показано на Рисунок 6.26.

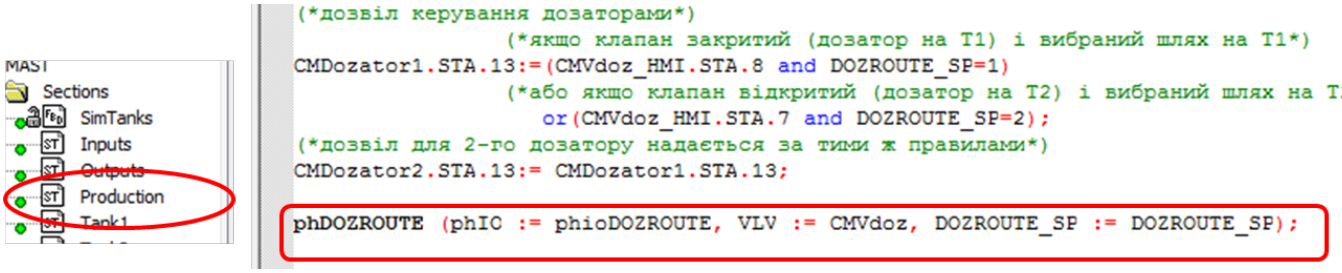


Рисунок 6.26 – Виклик етапу **phDOZROUTE**

- В zenon реалізуйте компонувальні блоки, як це показано на Рисунку 6.27:
  - для цього створіть Unit «Установка», який буде відповідати за всю Технологічну Комірку;
  - скопіюйте будь-який існуючий етап;
  - змініть параметри етапу, як це показано на Рисунку 6.27;
  - змініть реакції етапу, як це показано на Рисунку 6.28.

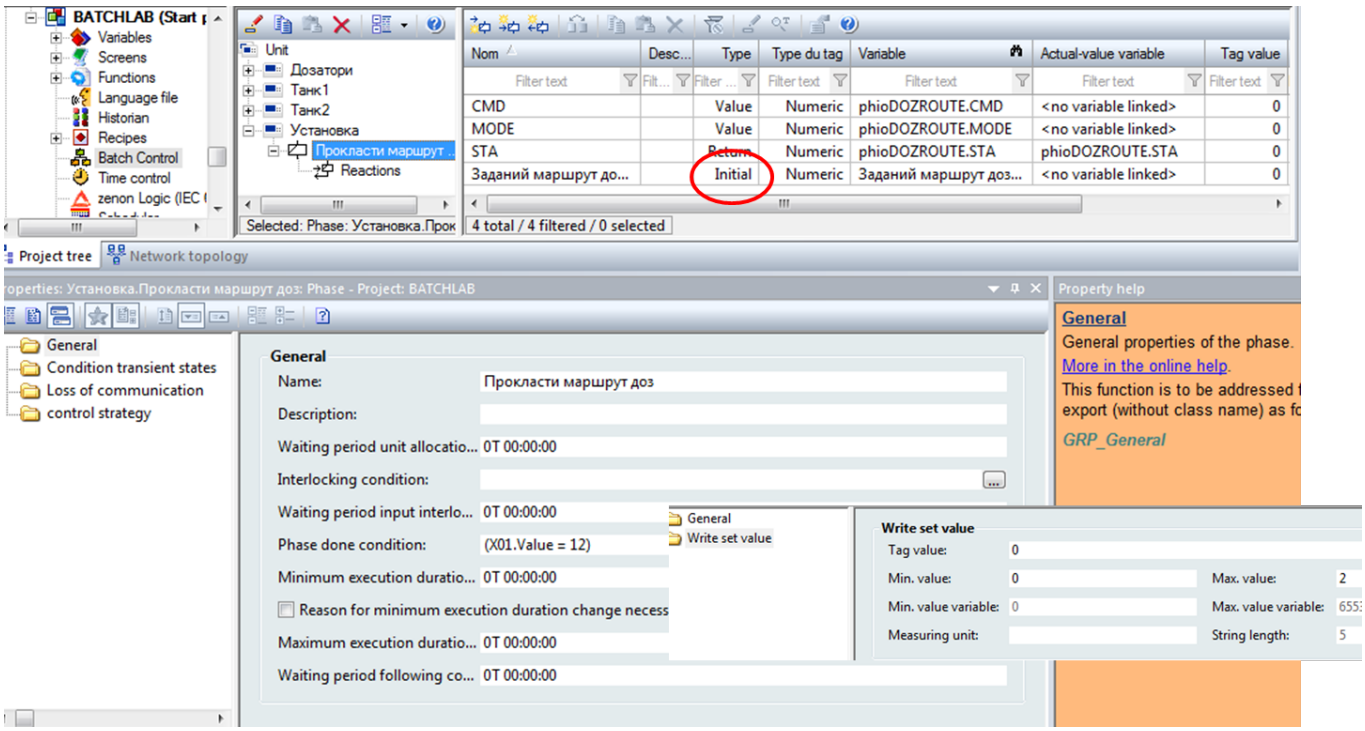


Рис.6.27. Налаштування параметрів етапу «Прокласти маршрут» в zenon

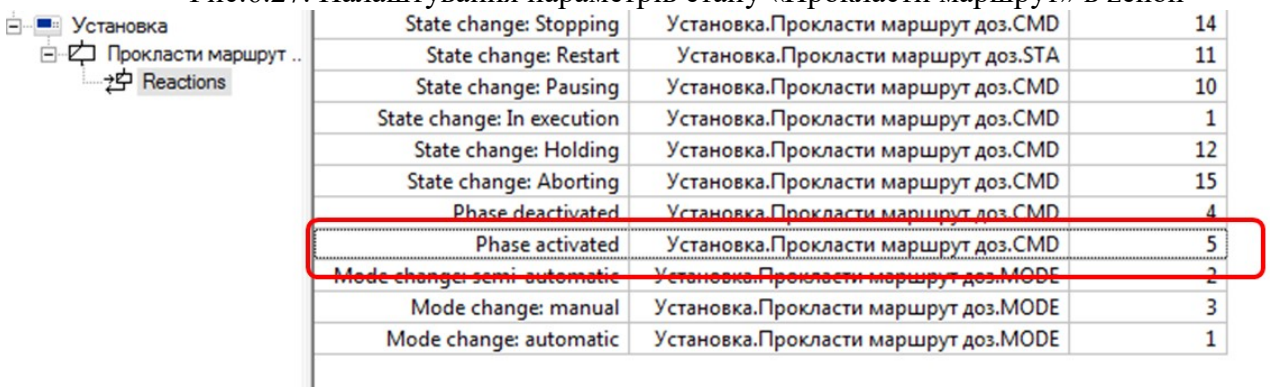


Рисунок 6.28 – Налаштування реакцій етапу «Прокласти маршрут»

- Скомпілюйте проект zenon та запустіть його на виконання.
- Модернізуйте обидва рецепти так, як це показано на Рисунку 6.29; в параметрах етапу «Прокласти маршрут» задайте вірні номера маршрутів.

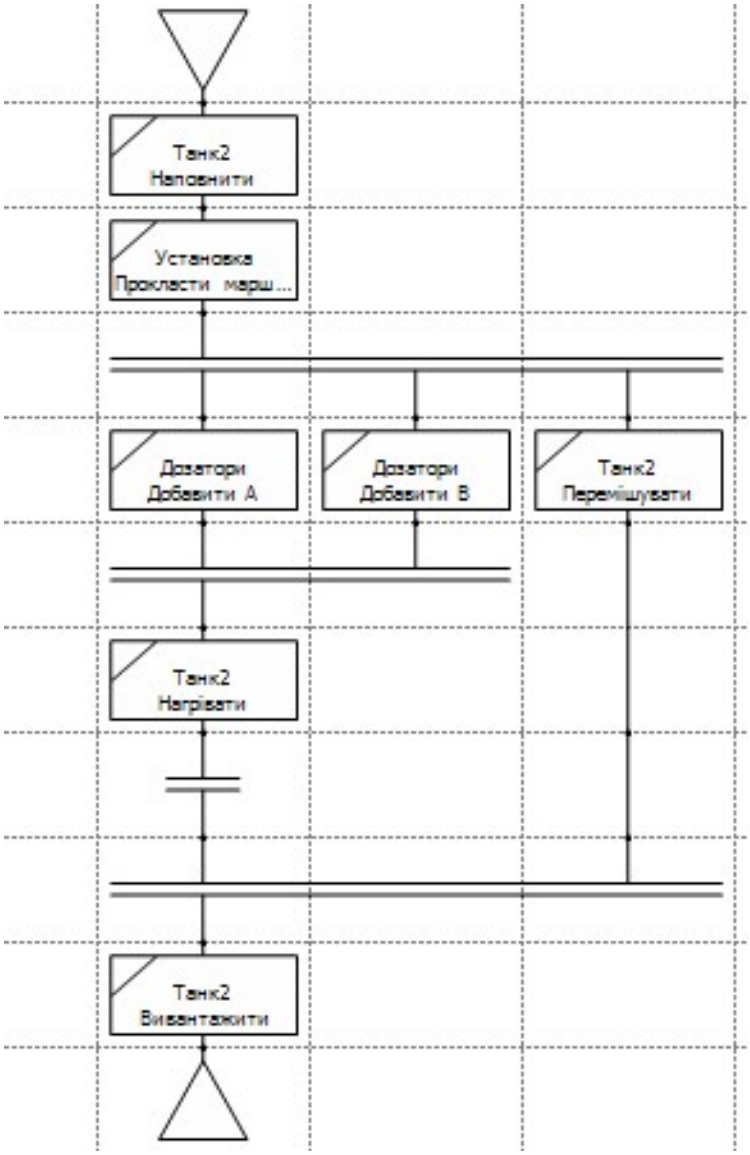


Рисунок 6.29 – Вигляд зміненого рецепту «Коктейль 2»

- Почергово запустіть рецепти на виконання і подивіться на правильність їх роботи, за необхідності виправте помилки.

6.5 Вирішення конфліктів використання ресурсів шляхом надання (**Allocation**) обладнання.

6.5.1 Запустіть одночасно два рецепти так, щоб їх етапи дозування виконувалися одночасно..

- В середовищі виконання Zenon запустіть спочатку рецепт «**Коктейль 1**», потім «**Коктейль 2**» таким чином, щоб після початку дозування для **Коктейлю 1** рецепт **Коктейлю 2** теж дійшов до дозування.
- Зверніть увагу на те, як ведуть себе етапи дозування і виясніть, чому це так.

Враховуючи, що 3-х ходовий клапан та Агрегат установки дозування є ресурсом спільного користування (common resource), то у будь-який момент часу будь-який споживач може використати його для своїх цілей. Однак, ці ресурси потребують ексклюзивного використання, тому необхідно передбачити обмеження одночасного доступу. У попередньому прикладі обмеження доступу до

агрегату визначається тим, що його етапи зайняті іншим рецептом. Однак. виконання етапу прокладення маршруту вже завершився для одного рецепту, в той час як інший використав його для себе. Таким чином, дозування відбулося в межах першого рецепту, тоді як фактично частина з цих доз була проведена в інший танк. Для передбачення ексклюзивного використання ресурсу обладнання одним споживачем (наприклад рецептом) до його вивільнення користуються такою координаційною діяльністю, як надання ресурсу (**Allocation**).

У нашому випадку перед перемиканням дозування на потрібний танк, необхідно зайняти його (Агрегат дозування). Після надання Агрегату рецепту, можна перемикати клапан на потрібний танк. Таким чином, поки Агрегат дозування буде зайнятий, ніхто не зможе не тільки скористатися його етапами (наприклад, коли дозування на одному з дозаторів закінчено), але і не перемкне клапан. Для виконання надання ресурсу рецепту в PFC передбачений спеціальний символ «**Allocation**», який використовується для попереднього займання ресурсу перед його використанням та вивільненням ресурсу.

6.5.2 Виправте рецепти так, щоб перед використанням дозування їм надавався у використання Агрегат, а після використання вивільнявся.

- В середовищі виконання zenon відредагуйте рецепти так, як це показано на Рисунок 6.30.
- Запустіть спочатку рецепт «**Коктейль 1**», потім «**Коктейль 2**» таким чином, щоб після початку дозування для **Коктейлю 1** рецепт **Коктейлю 2** теж дійшов до дозування.
- Зверніть увагу на те, як ведуть себе етапи наповнення.

При даній реалізації умова виконання наповнення в другому рецепті не виконується, тому що наступний за ним елемент **Allocation** не дає завершитися попередньому етапу. Для вирішення цієї проблеми задане значення наповнення повинно перевірятися всередині ПЛК.

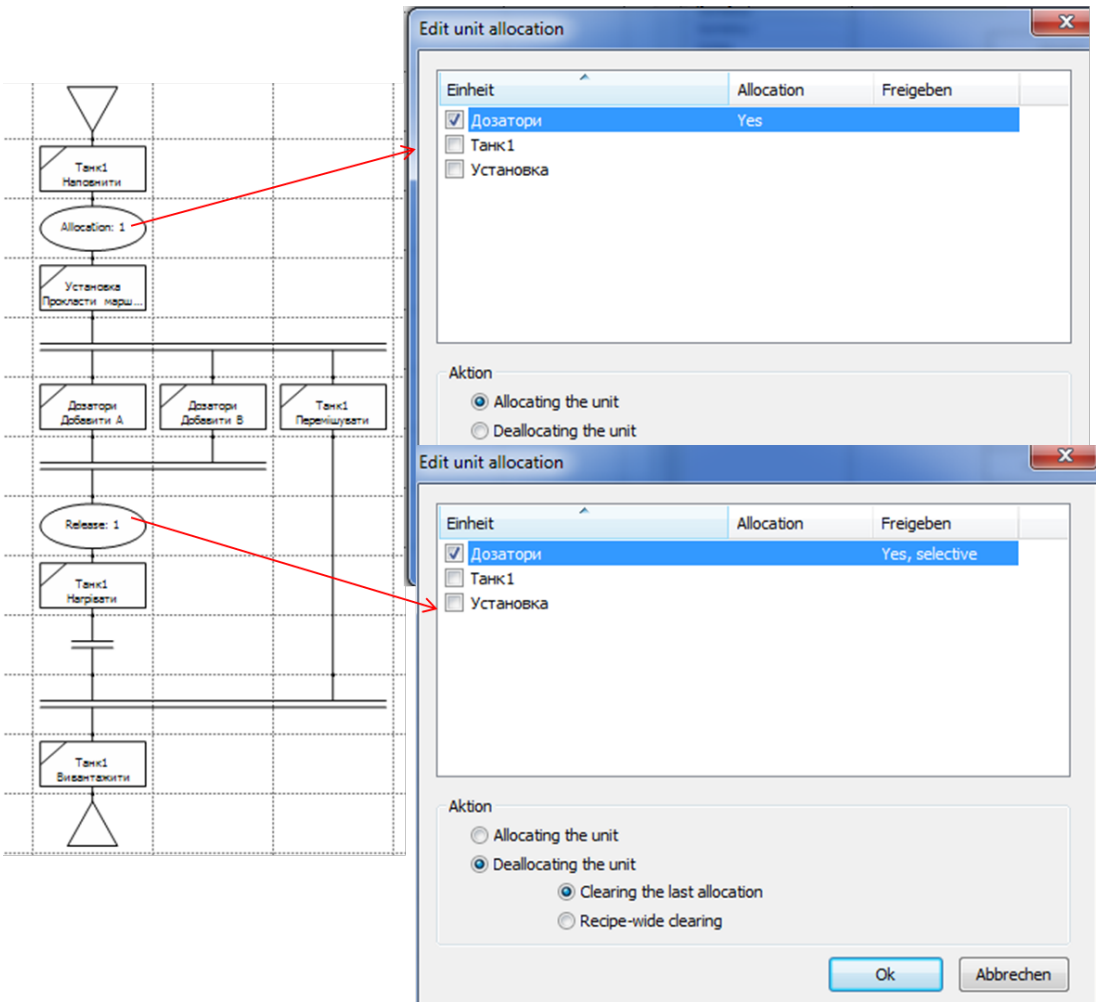


Рисунок 6.30 – Вигляд зміненого рецепту з елементами Allocation

6.5.3 Виправте компонувальні блоки етапів «Наповнити» так, щоб умова завершення перевірялася в ПЛК, а не SCADA. Перевірте одночасну роботу рецептів.

- В середовищі розробки zenon для етапів «Наповнити» відредагуйте «**Phase done condition**», щоб етап завершувався, коли значення STA=12.
- У Unity Pro відредагуйте **phFILL** таким чином, щоб він завершувався при досягненні заданого рівня:
  - добавте в інтерфейс дві змінні **Level** та **LevelSP** (Рисунок 6.32);
  - змініть програму **phFILL** (Рисунок 6.32);
  - змініть програми секцій **Tank1** та **Tank2** для передачі туди параметрів дійсних і заданих рівнів (Рисунок 6.33).
- Скомпілюйте проект і внесіть зміни в ПЛК.
- Запустіть середовище виконання zenon і запустіть одночасно обидва рецепти, подивіться на коректність їх виконання.

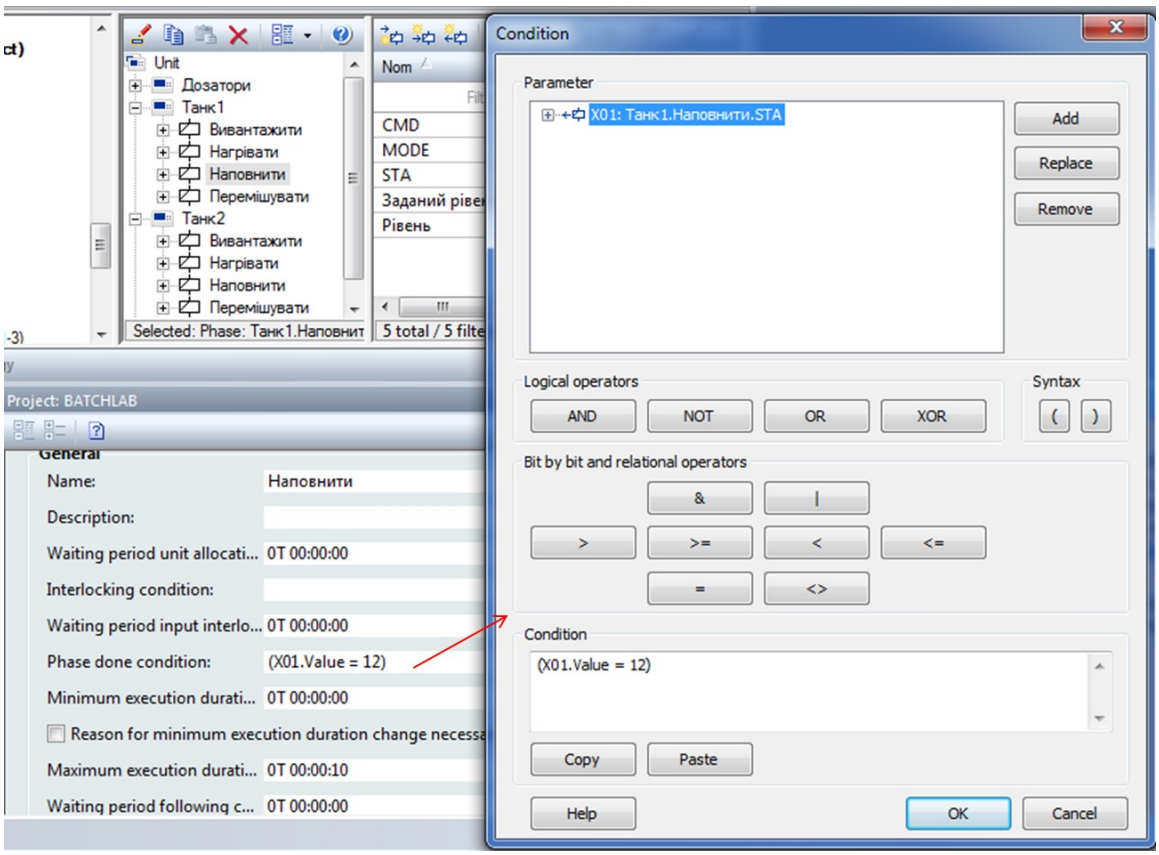


Рисунок 6.31 – Налаштування етапу наповнити для виконання умови завершення його в ПЛК

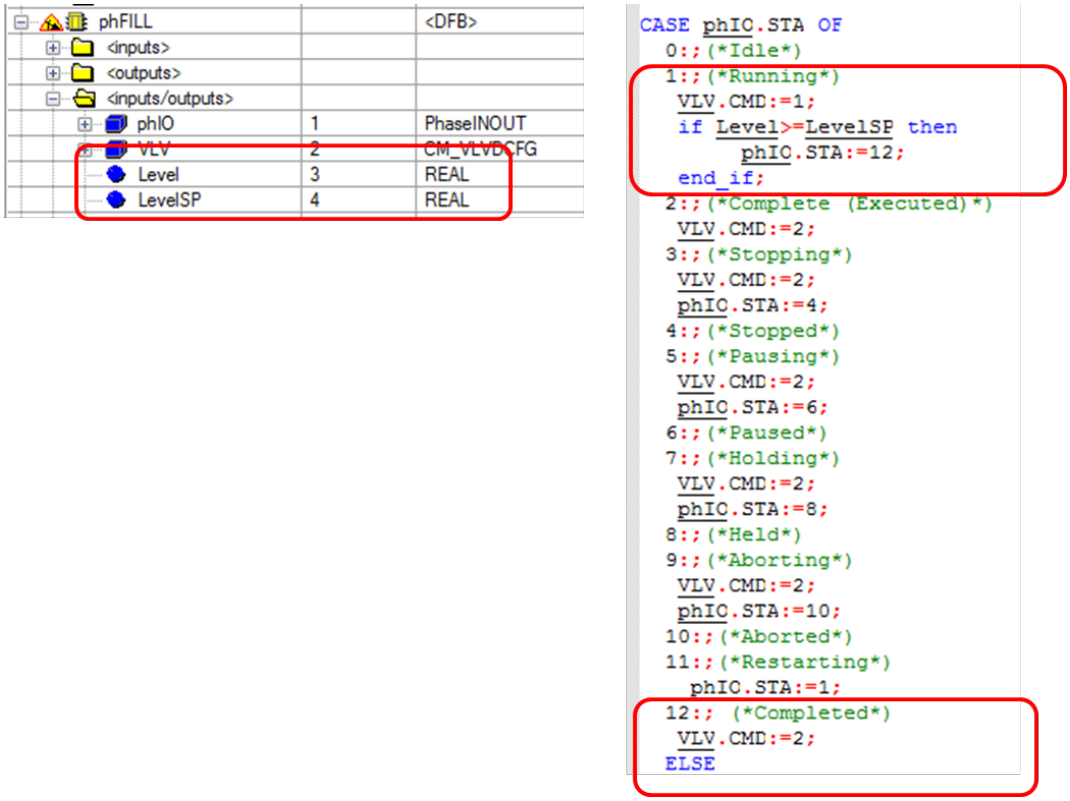


Рисунок 6.32 – Зміна **DFB**-типу **phFILL**

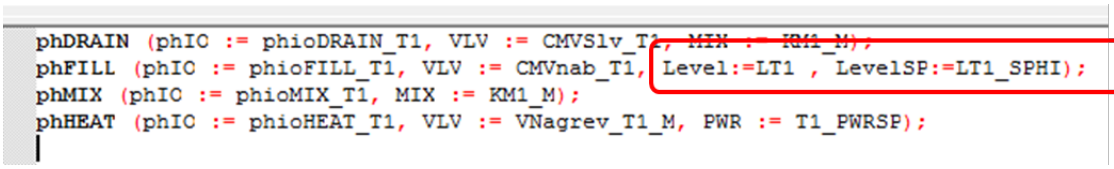


Рисунок 6.33 – Зміна секції **Tank1**

Контрольні питання

1. Опишіть узагальнену ієрархічну модель апаратурних об’єктів (Equipment Entity Model).
2. За якими принципами виділяються різні об’єкти в моделі апаратурних обєктів?
3. Де, навіщо і як в лабораторінй роботі використані агрегати ЕМ?
4. Поясніть чому в даній лабораторній роботі було проведене саме таке виділення апаратурних об’єктів, як показано на рис.6.2?
5. Навіщо потрібен allocation? Де це використовується вданій лабораторній роботі?
6. Що таке рецептно-обізнані агрегати?
7. Що таке вище керування згідно з ISA-88?
8. Поясніть за якими принципами працює алгоритм рецепту в даній лабораторній роботі.
9. Розкажіть про паралельне розгалуження та сходження в PFC на прикладі рецепту, щ обув використаний в даній лабораторній роботі.

ЛАБОРАТОРНА РОБОТА 7. КЕРІВНІ РЕЦЕПТИ

Мета роботи – розглянути та навчитися управляти керівними рецептами.

Загальні теоретичні відомості

Керівний рецепт (Control recipe) використовується для створення конкретної партії. Його створюють шляхом копіювання одного з екземплярів майстер-рецепту, а потім за необхідністю модифікують. Керівний рецепт повинен містити інформацію про продукт, яка необхідна для виготовлення конкретної партії цього продукту; про обладнання та апаратурні об'єкти технологічної комірки, що будуть використовуватися для приготування з достатньою деталізацією для їх ініціації і моніторингу. Керівний рецепт може бути змінений для врахування фактичної якості сировини і фактичного обладнання, що необхідно використовувати. Деякі приклади необхідної зміни:

- при ініціюванні партії або безпосередньо перед використанням визначають обладнання, яке насправді буде використовуватися для керівного рецепта.
- на основі якості сировини, що використовується, або проміжних аналізів якості продукції партії, додають нові або змінюють існуючі параметри;
- при нештатній ситуації може бути змінена процедура керівного рецепту.

Процедура технологічної комірки керівного рецепта складається з рецептурних процедур апаратів, рецептурних операцій і рецептурних етапів, що відносяться безпосередньо до тих, які означені в майстер рецепті. У момент створення керівного рецепта, вони «копіюються» 1:1 з майстер рецепту.

Керівний рецепт використовується для збору інформації про хід виготовлення конкретної партії.

В даній лабораторні роботі ми можемо обирати між двома типами рецептів в zenon Batch Control: **PFC** та матричний.

Ми вже повинні знати про тип **PFC** (Procedural Function Chart). Простіші процеси також можуть бути представлені у вигляді матричних рецептів. Там немає вибору послідовності (альтернатива), переходів або виділення апарату для матричних рецептів. Рецепт створюється у вигляді таблиці, в результаті чого послідовності кроків і стовпці представляють собою етапи або операції.Кожна клітинка таблиці може бути активована або деактивована незалежно одна від одної. Можна додати багато рядків і стовпців, як хотілося б.

Завдання до виконання лабораторної роботи

1. Розглянути та створити PFC та керівні рецепти.
2. Створти керівні стратегії та операції.
3. Розглянути та створити керівний рецепт.
4. Розглянути алгоритм обробки помилок.
5. Додати керівні елементи.

6. Розглянути Job ID.
7. Розглянути автоматичне створення керівних рецептів.
8. Записати статусні змінні.
9. Розглянути звітування та образів.

Порядок проведення роботи

7.1 PFC та матричні рецепти

Покажемо очищення («Мийка») **Танк1** як матричний рецепт.

- Створіть новий матричний рецепт.
- Вставте три колонки (етапи). Для цього натисніть на відповідний символ на правій частині заголовка таблиці.
- Зв'яжіть кожен етап одного з апаратів «**Танк1**» з одною колонкою.
- Добавте три рядки (кроки). Для цього використовується символ в нижньому кутку заголовка таблиці.
- Активуйте необхідні комірки, шляхом натискання лівої кнопки миші разом з натиснутою клавішею CTRL. Альтернативно ви можете також подвійним натисканням на комірці активувати властивістьPhaseactivев діалозі налаштування етапу.

Рецепт повинен виглядати, як показано на Рисунку 7.1.

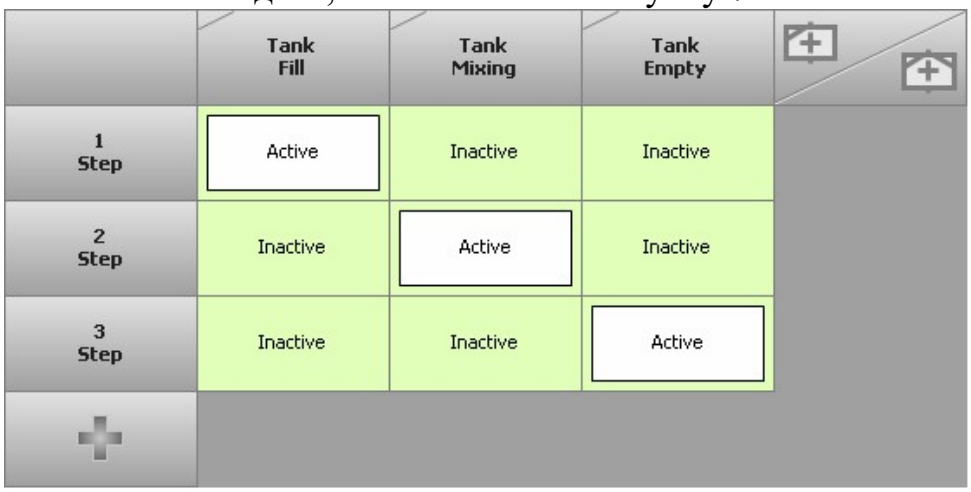


Рисунок 7.1 – PFC рецепт

- Тепер переключіться в тестовий режим і запустіть рецепт.
- Спробуйте також ручний режим.

7.2 Керівна стратегія (Control Strategy)

У апаратах часто трапляються кілька етапів, які відрізняються тільки в незначній мірі (таких як етапи «Наповнити» та «Вивантажити» у вашому прикладі).

Такі подібні етапи можуть бути замінені на єдиний етап з декількома керівними стратегіями (**control strategies**). Перевага такого рішення полягає в тому, що робота, яка необхідна для конфігурації проекту, значно знижується, як в zenon так і в програмі ПЛК.

Ми можемо тепер спробувати комбінувати етапи «Наповнити» та «Вивантажити» в єдиному етапі з керівною стратегією.

- Створіть новий етап з назвою «Наповнити» **level check**.
- Вставте всі теги і реакції етапу «Наповнити» шляхом копіювання та вставки.
- Активуйте в групі властивостей **Control strategy** властивість **Activ control strategies**.
- Створіть новий числовий параметр ініціалізації (**initialparameter**) з назвою **Control strategy** та з'єднайте з новою змінною.
- З'єднайте новий тег з властивістю **Control strategy tag**.
- Тепер розгорніть етап у вузлі апарату. Тут з'явився вузол **Control strategies.**
- Натисніть на вузлі керівних стратегій (**Control strategies**); на правій стороні з'являється перелік керівних стратегій.
- Створіть дві керівні стратегії «Наповнити» та «Вивантажити».
- Керівні стратегії тепер відображаються в вузлі апарату. Зв'яжіть змінний тег для керівних стратегій. Зробити це можна за допомогою панелі інструментів списку параметрів керівної стратегії (для керівної стратегії, обраної в дереві), або drag&drop.

Ви можете пов'язати таким чином будь-які змінні теги етапу для керівних стратегій. Значення, яке записується можуть бути встановлені незалежно для кожної стратегії керування. Якщо для керівної стратегії тег не зв'язаний, він також не береться до уваги при виконанні в рецепті керівної стратегії. Всі теги зворотного зв’язку (**return tags**)та реакції, які були налаштовані для етапу автоматично доступні у всіх стратегіях керування.

- Скомпілюйте файли проекту і перевантажте **Runtime**.
- Вставте в рецепт два екземпляри етапу «Наповнити» **level check**.
- Зробіть подвійний клік миші по етапу і виберіть в діалоговому вікні потрібну керівну стратегію.
- Знову виконайте рецепт.

7.3 Операції (Operation)

Операції можуть бути використані для інкапсулювання процесів, які відбуваються в кількох рецептах. Прикладом цього є наш рецепт очищення. Замість трьох етапів, тільки одна операція повинна бути вставлена в рецепті. В результаті цього, повний рецепт має більш чітке уявлення.

Ми створимо етап «Мийка», який потім використаємо в рецепті.

- Покажіть список операцій в редакторі рецептів.
- Створіть нову операцію.
- Вставте три етапи в операцію.

Операції не можуть бути перевірені безпосередньо. Щоб перевірити операцію, вона повинна бути вставлена в майстер рецепт. А потім це може бути перевірено.

- Збережіть операцію.
- Створіть новий майстер рецепт.
- Вставте операцію в майстер рецепт.
- Переключіть майстер рецепт в режим тестування.
- Зробіть подвійний клік по символу для операції в майстер рецепті.

Операція відкривається подвійним кліком. Проте це не шаблон операції, а екземпляр цього шаблону.

- Перемістіть екземпляр операції в свою власну групу, так щоб Ви могли побачити майстер рецепт і операцію поруч один з одним.
- Виконайте рецепт в ручному режимі.
- Зверніть увагу, як протягом виконання змінюється колір та вигляд операції, а також його спливаюча підказка.

Символ екземпляра операції представляє стан всіх етапів екземпляра операції. У підказці дається огляд помилок в екземплярі операції.

7.4 Керівний рецепт (Control recipe)

У стандарті ISA-88 проводиться відмінність між майстер рецептами і керівними рецептами. При цьому, керівні рецепти завжди походять від майстер рецептів. Ваша топологія тепер не може більше не бути зміненою; навіть значення тегів можуть бути змінені тільки, якщо це було встановлено в якості таких в майстер рецепті. **Керівні рецепти** можуть бути тільки створені та затверджені з майстер рецептів і **виконані тільки один раз.**

Виконайте наступні дії:

- Ухвалення рішення про майстер-рецепт, який буде служити основою для керівного рецепту.
- Затвердіть (**Approve**) рецепт. Для цього використовуйте іконку в панелі інструментів редактора рецептів, іконку в панелі інструментів майстер рецепта або в контекстному меню рецепта в списку.

Рецепт перевірений та затверджений. Помилкові рецепти не можуть бути затверджені в **Runtime**. Після того, як ви підтвердили рецепт, відповідно відображається статус в списку. Крім того, в рецепті зберігається інформація про користувача і дата затвердження. Розблокований рецепт (**unlocked recipe**) не може бути більше не відредагованим або перевіреним.

- Створіть керівний рецепт, базуючись на майстер рецепті. Для цього використайте відповідну іконку на панелі інструментів.
- Відкрийте список керівних рецептів.

Новостворений керівний рецепт тепер відображається в списку.

- Відкрийте керівний рецепт і виконайте його.

Після того, як рецепт завершив виконання, він не може бути виконаний повторно. Інформація про хід рецепта залишається в пам'яті. **Якщо той же самий рецепт повинен бути виконаний ще раз, за допомогою майстер рецепта повинен бути створений новий керівний рецепт.**

- Тепер знову викличте екран Batch Control і відкрийте список кері-

вних рецептів.

Керівний рецепт, створений заздалегідь, не відображається. У той час, як всі майстер рецепти наведені в списку майстер рецептів, перелік керівних рецептів повинні активно заповнюватися користувачами. Причина цього полягає в тому, що багато керівних рецептів очікуються протягом довгого часу. Завантаження всіх їх поставить навантаження на систему. Для відображення керівних рецептів:

- Виділіть керівний рецепт, який хочете відобразити.
- Означте фільтр для статусу керівного рецепта (підготовлені, у виконанні, завершені, застарілі) шляхом активації або деактивації відповідних значків на панелі інструментів списку майстер рецептів.
- Натисніть на **Show control recipes** в панелі інструментів.

Всі керівні рецепти, що відповідають фільтру, будуть показуватися в списку.

Для того, щоб запобігти подальшому створенню керівних рецептів або виконанню вже існуючих керівних рецептів, майстер рецепти можуть бути позначені, як застарілі.

- Виберіть затверджений майстер-рецепт, який ви хочете позначити як застарілий.
- Відкрийте контекстне меню і виберіть **Mark asobsolete**.
- Тепер спробуйте створити новий керівний рецепт.

Новий керівний рецепт не може бути створений.

Далі розглянемо зміну параметрів в керівних рецептах.За допомогою редагованих командних параметрів, властивості в керівному рецепті можуть бути встановленими як виправні (**asamendable**). Як результат, стане можливим перезаписувати значення командного тегу в кожному рецепті управління. Значення в керівному рецепті може бути змінено тільки в межах, які були встановлені для тега в майстер рецепті.

7.5 Обробка помилок (Error Handling)

Синхронізація між ПЛК і zenon дуже важлива для Batch Control. З цієї причини, кожен етап може виявити проблеми зв'язку з ПЛК і реагувати на них. Етап повідомляє про втрату зв'язку, якщо параметри команди не можуть бути записані або якщо виконання формули для помилки зв'язку дає істину. Якщо відбувається втрата зв'язку, етап призупиняється (**paused**) або утримується (**hold**), і спрацьовує подія, для якої можуть бути налаштовані реакції.

Невдалий запис командних параметрів. Після запису командних параметрів, значення відповідних змінних негайно зчитуються з ПЛК. Крок «**Записати командний параметр**» завершується тільки тоді, коли всі значення, які були прочитані, відповідають записаним. Якщо хоча б одна змінна значення відрізняється, то всі командні параметри записуються знову після заданого часу очікування. Якщо після третьої спроби запису, це як і раніше, не представляється можливим для всіх значень змінних, які повинні бути прочитані коректно, етап повідомляє про помилку зв'язку.

Формула для втрати зв’язку. На додаток до згаданого вище автоматичного виявлення помилок, для виявлення відмов зв'язку також можна пов'язати формулу. Таким чином, можна виявити збої не тільки при написанні значень параметрів, але кожного разу, коли етап активний. Для того, щоб означити формулу для виявлення втрати зв'язку:

- Відкрийте властивості етапу.
- Перейдіть до групи властивостей **loss of communication**.
- Натисніть на кнопку«**...**» в полі, поміченому як **Loss of communication**.
- У редакторі формул створіть необхідну формулу з параметрами з’єднаними з етапами.

Для того, щоб гарантувати безпечний повторний запуск рецепта після втрати зв'язку, можна запросити підтвердження для відновленого зв'язку:

- Введіть формулу в поле **Loss of communication acknowledged**.

7.6 Керівні елементи (Control elements)

Для того, щоб адаптувати екран Batch Control для різних користувачів або кінцевих пристроїв, доступно багато керівних елементів. Таким чином, ви маєте можливість створювати різні екрани для Batch Controlз різними вимогами.

Перейдемо до списку рецептів**. RECIPELISTS** (Список рецептів).

- Створіть новий екран типу Batch Control.
- Вставте шаблон **Standard control element**.

Шаблон містить вже відомий нам редактор рецептів і окремі елементи для списку майстер рецептів, списку керівних рецептів і кнопок для функцій, які належать до списків.

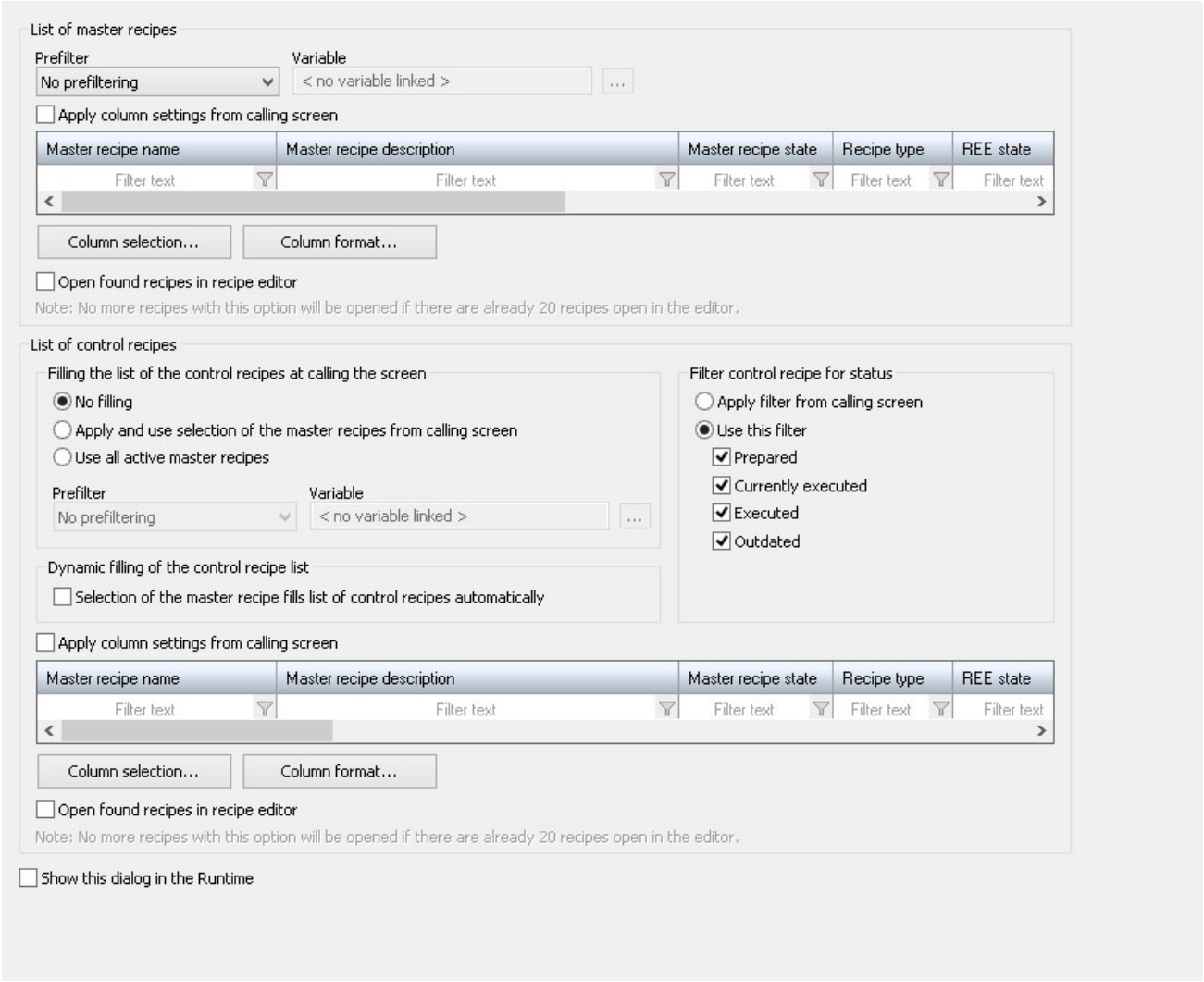
Ці елементи вільно-конфігуровані; Ви можете встановити розмір, позицію, колір, шрифт і висоту рядку за бажанням

Оскільки ми тепер маємо списки як окремі елементи, закріплювальні списки більше не потрібні в редакторі.

- Виберіть редактор рецептів для відображення його властивостей.
- Деактивуйте опцію **Dock able window** в групі властивостей **Display**.

Для більш швидшого пошуку керівних рецептів, ви можете мати оновлення списку автоматично, коли змінюється вибір в списку майстер рецептів.

- Вставте на екран керівний елемент.
- Список рецептів вже можуть бути відфільтровані або заповнені з переключенням екрану.
- Відкрийте діалог для конфігурування функцій перемикання екранів і перейдіть до вкладки Recipe list settings.



Ми хочемо, щоб відобразити всі майстер рецепти, ім'я яких відповідає критерію фільтра, який може бути означений під час виконання, а також всі супутні керівні рецепти. Для цього необхідно виконати наступні дії:

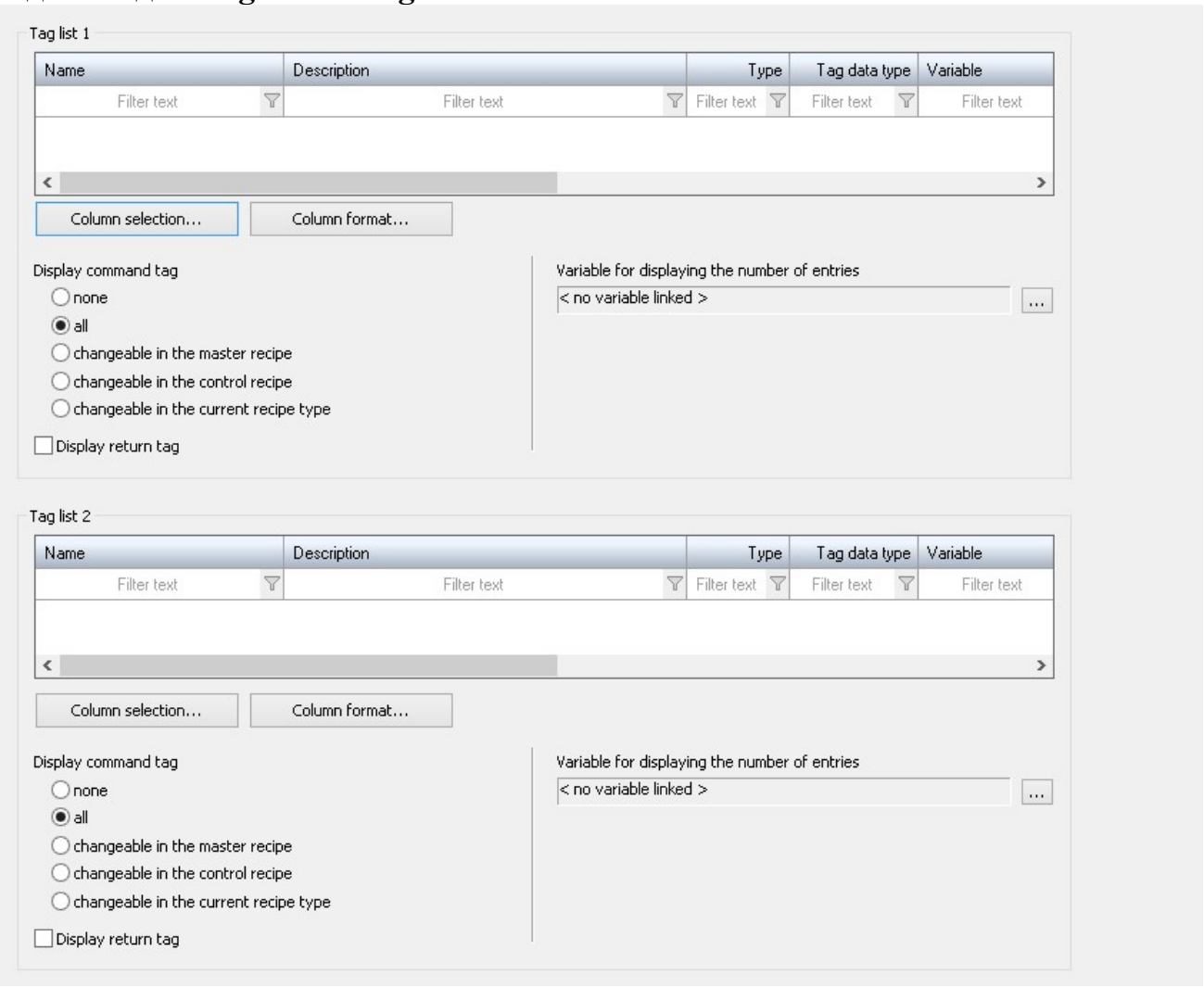
- Виберіть в **List of master recipes** – **Prefiltering** Name fromvariable.
- Відкрийте діалогове вікно, щоб зв'язати змінні і нові внутрішні змінні типу string.
- Виберіть в **«Наповнити» ing of the list of control recipes when screen is called up** Useall activated control recipes.
- Покажіть нову змінну в динамічному текстовому елементі на екрані навігації.
- Введіть потрібний рядок фільтра в Runtime для змінної фільтра і виконайте функцію перемикання екрану.

При виклику екрану, показуються тільки майстер рецепти, які відповідають параметрам фільтра. Якщо створений новий рецепт, це також показано, якщо він не відповідає умовам фільтра.

Списки тегів (TAG LISTS). Є два списки тегів (Tag lists) для того, щоб відобразити параметри в режимі виконання і мати можливість редагувати їх.

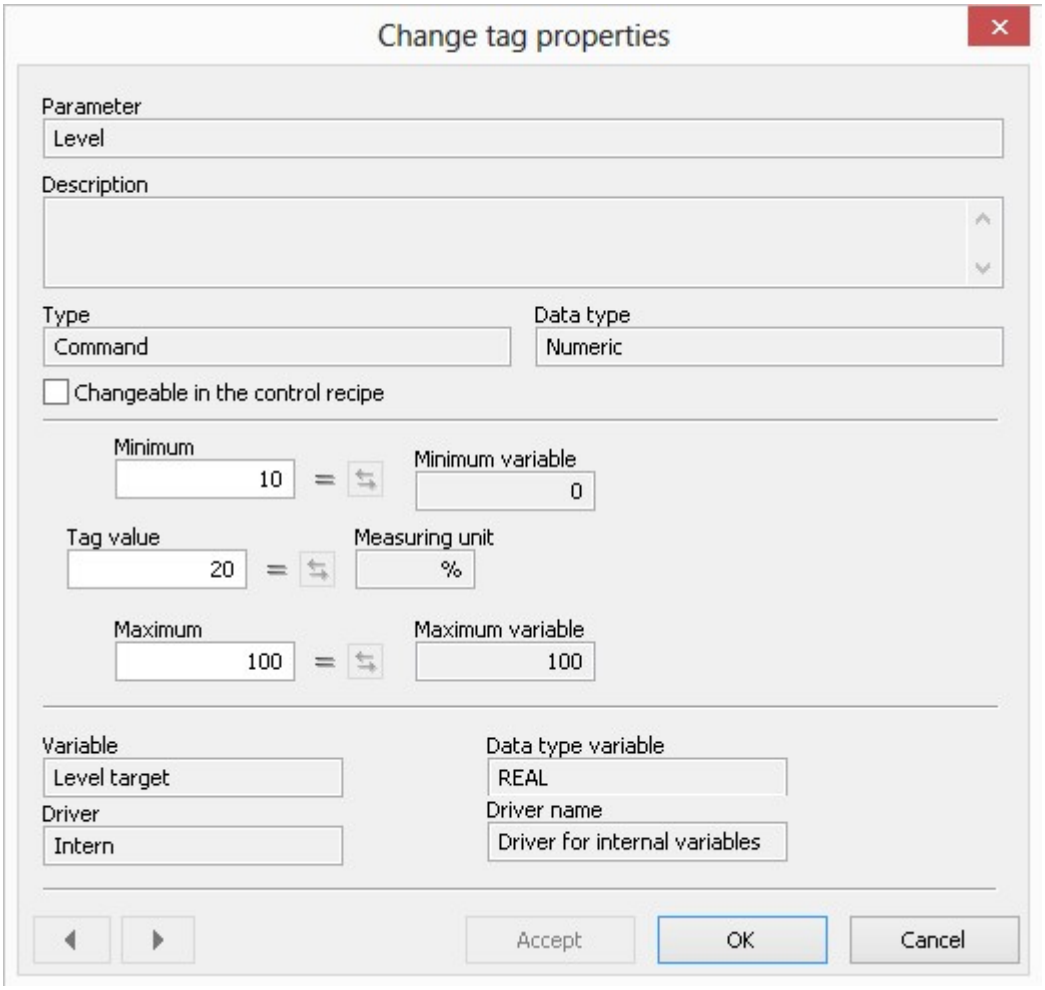
- Виберіть в контекстному меню елементів управління, два списки тегіводин за іншим і вставте їх на екран.
- Відкрийте діалог для конфігурування перемикання екрану і перей-

діть до вкладки **Tag list settings.**



- Виставте перший список тегів на **All command parameters,** а другий на **No command parameters** і **Show return tags**.
- Переключиться на Runtime, перезавантажте та виберіть фазу в редакторі рецептів.

Два списки завжди заповнюються відповідно до обраного в редакторі рецептів і фільтру, який був встановлений. Це може бути відредаговане, двічі клацнувши по тегу в списку, за умови, якщо редагування параметрів дозволено.



Діалог редагування містить інформацію про тег, таких як назва, опис і пов'язані змінні (не може бути відредагований), а також поля для введення значення тега і межі тегів. Змінити здатність тега в рецептах управління також може бути тут виконана.

Клавіатура (KEYBOARDS). Для того, щоб мати можливість редагувати теги на пристроях з сенсорним введенням, діалог редагування тегів може бути замінений на клавіатуру. Існують системні змінні, які доступні для того, щоб мати можливість відображати ту ж інформацію, що і в діалозі, на цьому екрані клавіатури.

- Добавте усі системні змінні з групи Batch Control у Ваш проект.
- Створіть новий екран клавіатури.
- Вставте на екран шаблон елементів керування Numerical minimal.
- Додайте додаткові динамічні текстові елементи і зв'яжіть системні змінні.
- Вставте елемент вводу уставки.

Екранна клавіатура завершена. Тепер вона повинна бути пов'язана зі списками тегів, замінивши діалог.

- Натисніть вузол BatchControl в дереві проекту.
- Відкрийте групу властивостей Edit parameters.

У цій групі ви можете пов'язати відповідний екран клавіатури для кожного типу даних тега.

- Зв'яжіть клавіатуру що була створена в Numerical parameters.

В якості альтернативи глобальної настройки екранну клавіатури для чотирьох типів даних тегів, ви можете зв'язати спеціальні екрани клавіатури у властивостях параметрів. Установка для тега перезаписує глобальні настройки в процесі.

7.7 JOB ID

Batch Control дає можливість призначати керівні рецепти з номером завдання (**job number**). Таким чином, ви можете призначити рецепти чіткі, відфільтровані відповідно до робочих місць рецепти або синхронізувати рецепти з архівами за партіями (lot archives).

Якщо Ви хочете використовувати job numbers, зробіть наступні дії:

- Створіть нову строкову змінну для внутрішнього драйверу.
- Зробіть змінну видимою за допомогою динамічного елемента тексту на екрані.
- У дереві проекту, клацніть на вузлі Batch Control, щоб відобразити його властивості.
- Прив’яжіть змінну до властивості **Job variable** property.
- Виставте властивість **Accept value from job variable on** в значеня creation of the control recipe.

Тепер скомпілюйте файли виконання і перезавантажити проект. Система перевіряє, чи має пов'язана змінна коректне значення при створенні керівних рецептів. Якщо це не так, то не може бути використано жодного керівного рецепту. Номер завдання (**job number**) завжди відображається в діалозі конфігурації при створенні керівних рецептів. Крім того, номер завдання відображається в списку керівних рецептів. Ви можете фільтрувати, сортувати і групувати список за номерами завдань. Крім того,коли екран Batch Control викликається, Ви можете використовувати фільтр по номеру завдання.

- Виставте властивість **Accept value from job variable on** рівною starting the control recipe.

За допомогою цієї установки, керівні рецепти також можуть бути створені без валідного значення змінної робочого завдання. Значення змінної перевіряється і переноситься в рецепт тільки при запуску рецепту.

Задача:

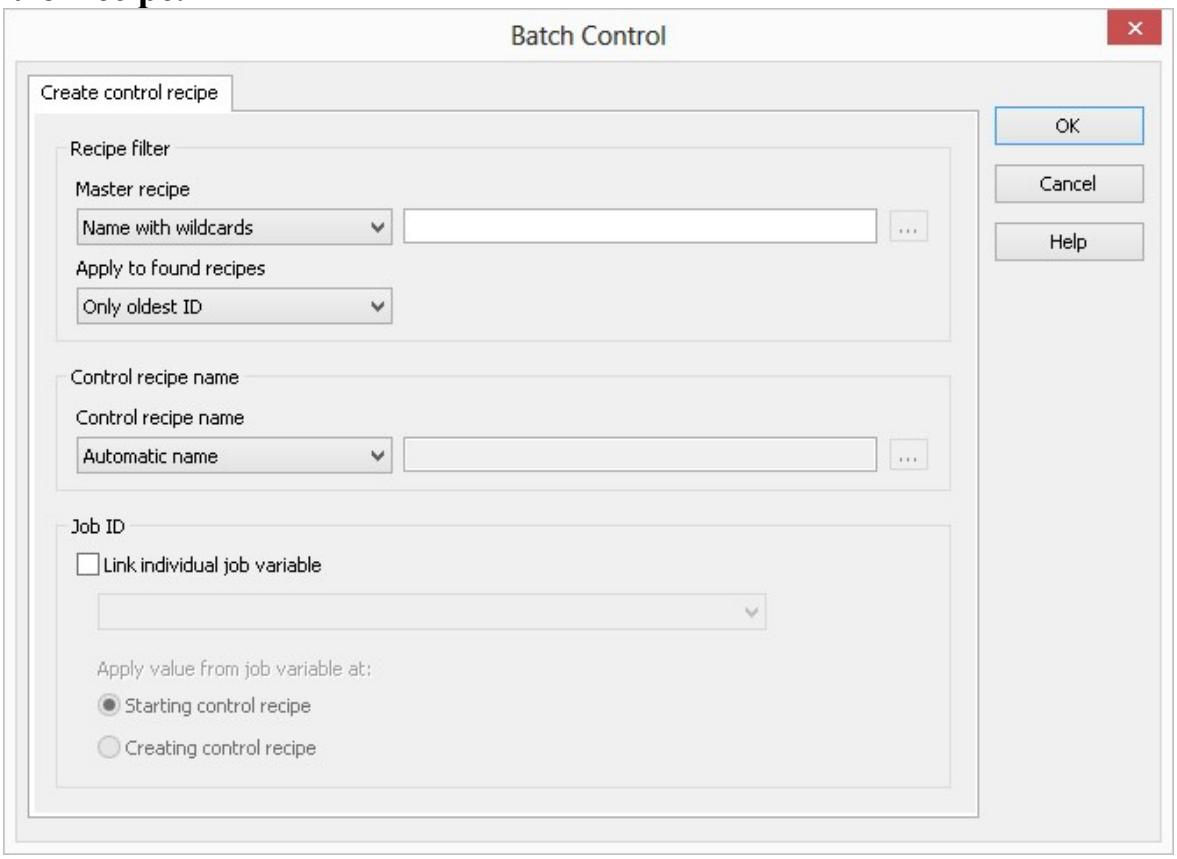
- Створіть функцію перемикання екрану, яка заповнює список керівних рецептів з усіма керівними рецептами, які належать до певного робочого завдання.

7.8 Автоматизація (Automation)

Модуль Batch Control дає можливість створювати керівні рецепти і контролювати їх прогрес не тільки вручну, але і автоматично за допомогою функції. Кілька керівних рецептів можуть, таким чином, бути одночасно створені для роботи і початися в певній точці часу. Крім того, всі рецепти, які виконуються можуть бути одночасно припинені, у разі виникнення проблем.

7.8.1 Створення керівних рецептів.

Для автоматичного створення керівних рецептів доступна функція **Create control recipe.**



В діалозі конфігурації може бути створений фільтр для вибору майстер рецептів, з яких керівний рецепт повинен бути створений. Спочатку ви маєте вказати властивість рецепту, яка повинна бути використана для фільтру. Для цього доступні **Імена** та **ідентифікатори**. **Ім'я** може бути або введено безпосередньо в діалоговому вікні, або може бути прочитане зі змінної, яку необхідно визначити. В обох випадках можна використовувати групові символи (* і %). Оскільки фільтру може відповідати кілька видів майстер рецептів, можна також стверджувати, що повинні бути використані всі чи тільки найстаріший рецепт. Якщо для фільтрації використовується **ідентифікатор,** повинна бути пов'язана змінна. У цьому випадку використання розділових знаків не допускається, отже, тільки один рецепт може відповідати фільтру.

У групі ім'я керівного рецепта (name control recipe) можна вибрати, чи буде створена назва керівного рецепта автоматично або взята з текстової змінної. При цьому автоматичне ім'я включає ім'я майстер рецепта і серійний номер. Якщо ім'я слід читати зі змінної, необхідно дотримуватися обережності, щоб гарантувати, що не існує вже рецепту з цим ім'ям, в іншому випадку ніякого керівного рецепта не буде створено.

І, нарешті, в групі номер робочого завдання (**job number group**), можна вибрати, чи використовуються глобальні параметри для номера робочого завдання, які будуть використовуватися, або окремі з них. При установці послання на окрему змінну завдання, можна вибрати зі списку декількох змінних робочого завдання (які повинні бути означені у властивостях модуля

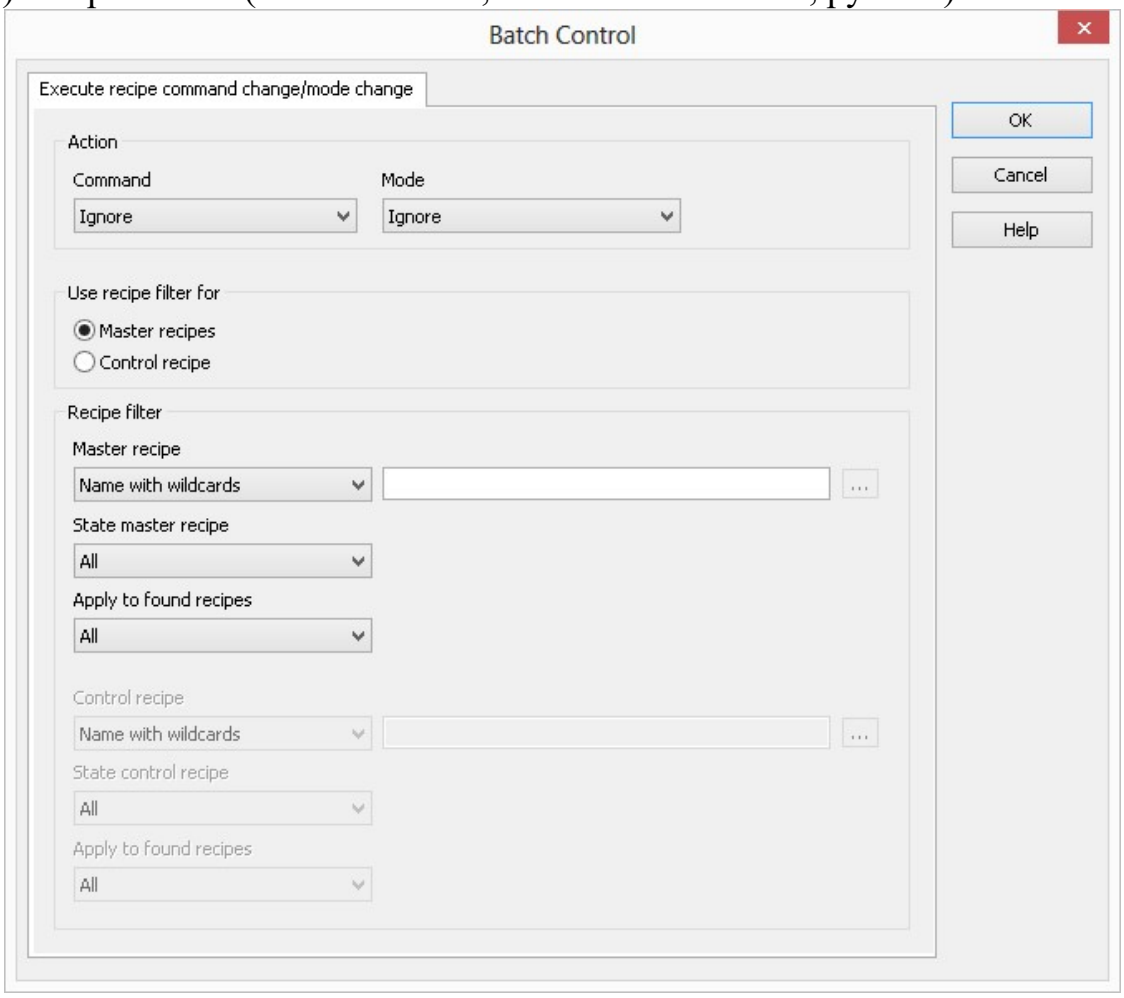
BatchControl). Крім того, можливо, незалежно від глобального параметра вибрати номер робочого завдання, чи буде записуватися в рецепті при його створенні або його запуску.

Задачі:

- Створіть функцію, яка дозволяє створити кілька керівних рецептів в той же час.
- Майстер рецепти повинні бути доступні для вибору за допомогою текстового фільтра в середовищі виконання.
- Зв'яжіть індивідуальну змінну робочого завдання і запишіть номер робочого завдання в рецепт при запуску.

7.8.2 Виконання зміни рецептурної команди/режиму

Ще одна функція дозволяє впливати на вже існуючі керівні рецепти, в якому рецепту направляються команди керування станом (старт, пауза, утримання і т.д.) або режимом (автоматичний, напівавтоматичний, ручний).



Дії (команда, перемикач режимів роботи) означуються в першому діалоговому вікні. На наступному кроці, за допомогою фільтра, що повинен бути застосований до дій, повинні бути обрані рецепти. Для цього доступні майстер рецепти і керівні рецепти. Крім параметрів фільтрів в функції the **Create control recipe**, також можна фільтрувати рецепт за статусом. Якщо для дії будуть використовуватися керівні рецепти, в фільтрі повинен бути заповнений майстер-рецепті, оскільки керівні рецепти однозначно ідентифікуються тільки разом зі своїми майстер-рецептами.

Задачі:

Створіть функцію, яка скасує всі керівні рецепти, які зараз виконуються.

Створіть сценарій, за яким керівні рецепти можуть бути створені послідовно, а потім відразу запуститися.

7.9 Адміністрування користувачів

В модулі Batch Control можуть надаватися розштрені права користувача, використовуючи **Zenon user control**. Всім діям в Batch Control може бути присвоєний рівень авторизації за допомогою функціональних властивостей авторизації проекту.

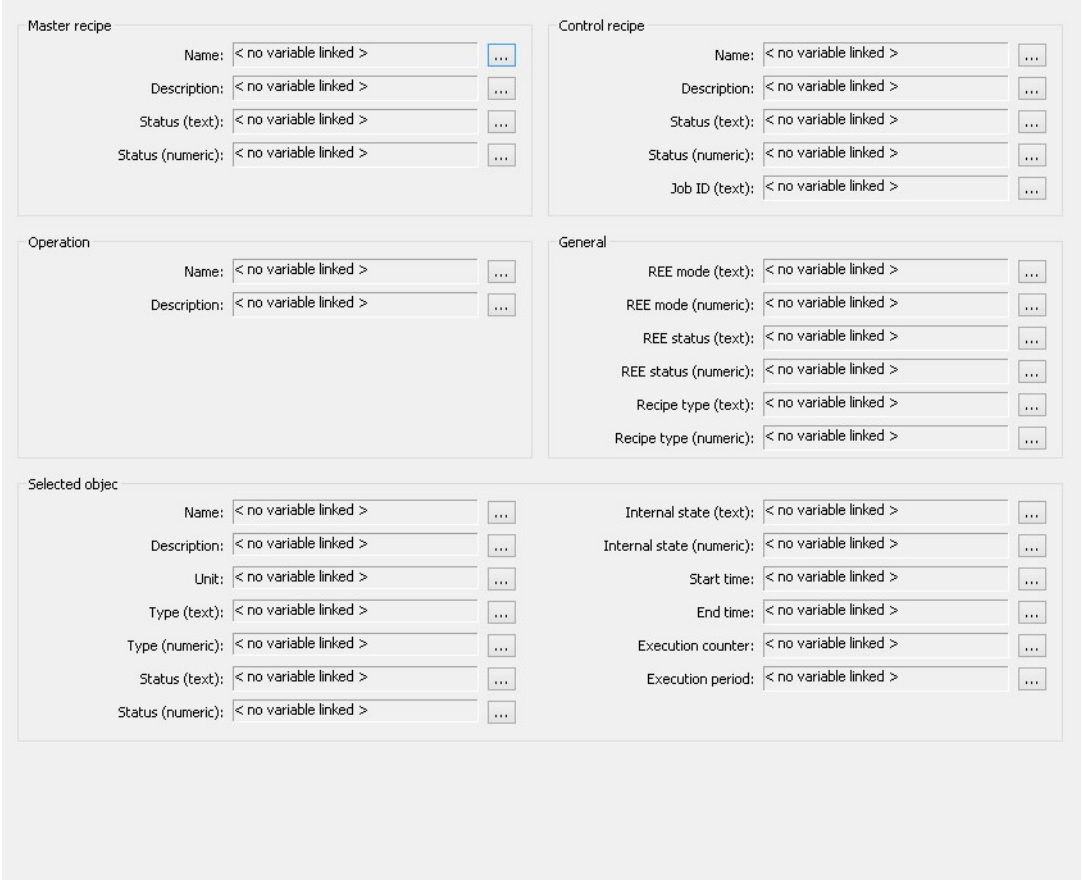
Задачі:

- Призначте рівні авторизації дії в модулі Batch Control.
- Створіть трьох користувачів (розробник рецепта, затверджувач, працівник) і призначте їм права, відповідні їх ролі.
- Перевірте функціональність в Runtime.

7.10 Запис статусних змінних

Для того, щоб мати можливість показати стан виконання двигуна, рецепту або об'єктів в рецепті, модуль Batch Control пропонує можливість записувати цю інформацію в змінних. Ці змінні можуть бути відображені або оцінені в будь-якій бажаній формі.

При цьому проводиться різниця між двома типами змінних зв'язування. По-перше, є вкладка змінної зв'язування на вкладці переключення екранів. В цій вкладці згодом може відображатися інформація про змінні рецепта, що в даний час відкритий в редакторі рецептів або об'єкт, що міститься в даний момент.



Крім того, інформація може бути пов'язана зі змінними у властивостях апарату в групі **Runtime Information**, яка містить інформацію про рецепти, які працюють в **Runtime**. Таким чином, неважко встановити рецепт, який в даний час назначений на апарат.

Задачі:

- Відобразіть ім'я і опис об'єкта, який був обраний в даний момент на екрані Batch
- Створіть функцію, яка відкриває майстер рецепт, що в даний час назначений на апарат:

- З'єднайте змінну для властивості апарату **master recipe ID.**
- Встановіть в перемикачі екранів **Prefiltering: ID from variable** і зв’яжіть з тією самою змінною.
- Активуйте опцію **Open recipes found in the recipe editor.**

7.11 Звітування

Рецепти від модуля Batch Control доступні для **Report Viewer**. Таким чином, можна створити детальну документацію по рецепту або зберегти у форматі PDF, якщо це необхідно. Дві речі, які необхідні для того, щоб мати можливість створювати такі звіти в режимі виконання: шаблон звіту (RDL-файл) і функцію, яка заповнює цей шаблон даними рецепта.

7.11.1 Шаблон рецепту

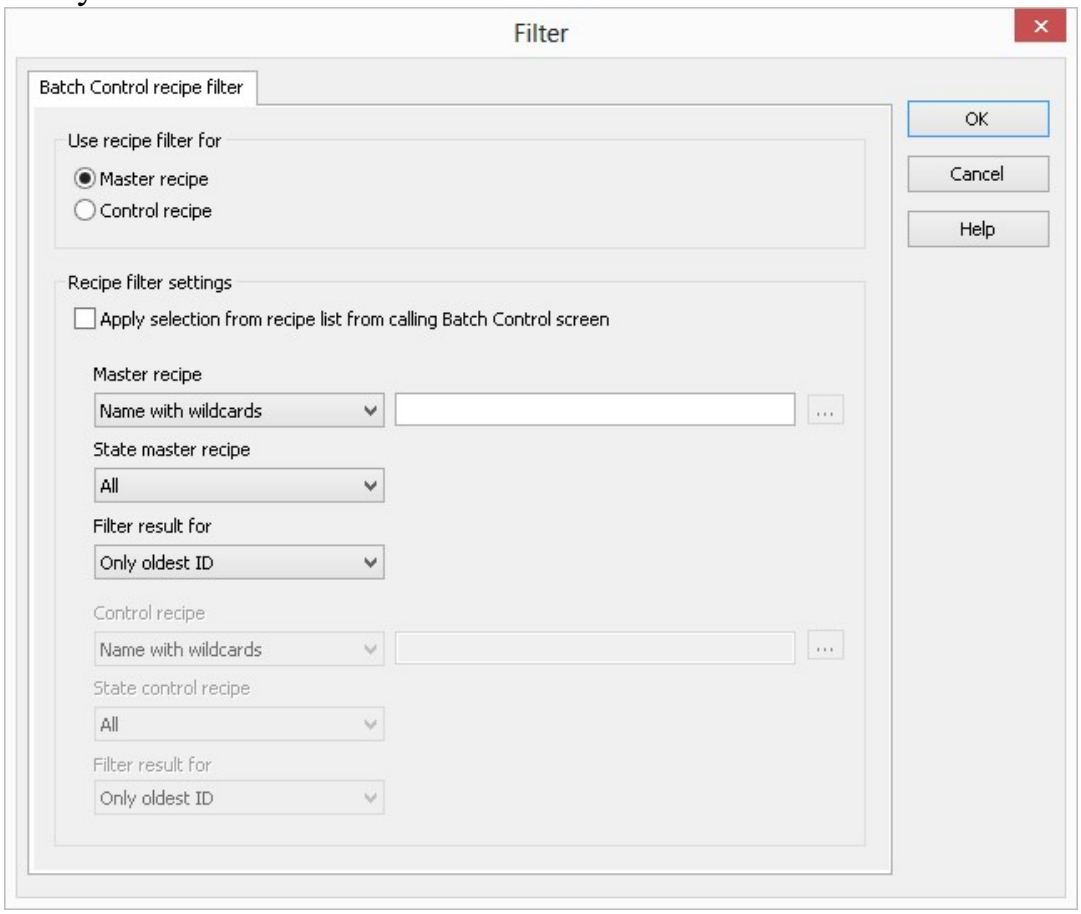
У стандартному шаблоні для звітів, який поставляється разом з Zenon, всі необхідні набори даних і таблиці, які доступні для пакетних звітів включені.

Щоб створити **batch report**, виконайте наступні дії:

- Створіть новий файл означення звітуі відкрийте його в **MSReport Builder**
- Видаліть всі непотрібні набори даних і таблиці з робочого листа
- Збережіть файл.

7.11.2 Перемикання екранів та фільтр рецептів.

При перемиканні екрану на екран Report Viewer, можна означити, використовуючи фільтр, дані якого рецепт, який повинен бути використаний для заповнення звіту.



Діалог фільтра пропонує вже знайомі настройки фільтра, такі як майстер рецепт/ керівний рецепт, фільтр відповідно до назви, ID і т.п. В якості альтернативи фільтрам для певного рецепта, також можна вибрати, щоб використовувати рецепт, обраний в списку рецептів. Для закінчення доступна настройка **Accept selection from recipe list from calling Batch Control screen**.

Задачі:

- Створіть звіт, в якому перераховані всі етапи рецепта.
- Створіть функцію, яка дозволяє заповнити звіт з даними з рецепта, який в даний час показаний.

7.12 Збереження даних

Рецепти зберігаються в Batch Control модулі в Runtime. Дані рецепта не можуть бути імпортовані в редактор. Дані, отже, не зберігаються в каталозі

Runtime. Є кілька файлів і папок під PROJECT\RT\FILES\Zenon\system\BatchRecipes:

Файл/папка	Опис
xxx.MR	Збережений майстер рецепт. xxx - IDрецепту
xxx.crd	Папка для керівного рецепту майстре рецепту. xxx– це IDмайстер рецепту
Recipe.unique	список майстер рецептів. Список збережений в пам'яті і записаний на диск як бекап
xxx.crd\yyy.CR	Збережений керівний рецепт. yyy- IDкерівного рецепту
xxx.crd\Recipe.unique	Список усіх керівних рецептів майстер рецепту
Operations	Папка в якій зберігаються усі операції
Operations\zzz.OP	Збережена операція
Operations\Recipe.unique	Список усіх операцій

7.13. Образ (Image)

Статус виконання Batch recipes можуть бути збережені циклічно в файл зображення. В результаті цього забезпечується те, що, в разі незапланованого збою Zenon **Runtime** (через відключення живлення, наприклад), всі рецепти можна продовжувати при перезапуску.

Для установки часу циклу для написання файлу зображення

- Відкрийте властивості модуля BatchControl.
- Активуйте в групі властивостей **General**властивість **Activate cyclical writing.**
- Встановіть властивість **cycle time**, значення>30.

Якщо циклічний написання буде вимкнено зображення записуються тільки якщо середовище виконання закрита належним чином.

Контрольні питання

1. Що таке керівні рецепти і для чого вони використовуються? Коли і з чого створюються керівні рецепти?
2. Коротко опишіть матричний вигляд опису рецепту.
3. Для чого можна використовувати керівну стратегію?
4. Яке призначення операцій? Де використані операції в даній лабораторній роботі?
5. Розкажіть про обробку помилок в керівних рецептах.
6. Які сторінки (екрани) в zenon Batch використовуються для адміністру-

вання керівними рецептами?
7. Для чого використовується Job ID?
8. Які механізми автоматизації створення та запуску керівних рецептів передбачені в zenon Batch?
9. Коротко розкажіть про адміністування користувача.
10. Яка інформація про керівний рецепт може бути доступна через змінні?

Глосарій до лабораторного практикуму АПВ

У модулі Zenon Batch Control використовуються наступні терміни:

Term	Definition
Unit	Physically available machine or equipment part with which phases can be carried out. (ISA 88 Unit)
Releasing the unit	Element of module Batch Control which cancels the allocation of a unit in the unit manager. With this the unit can be allocated by another recipe again.
Unit allocation	Element of module Batch Control which causes the allocation of a unit in the unit manager. An allocated unit can only be used by phases with the recipe. With this the unit is locked for phases of other recipes which are executed parallel.
Unit manager	Internal management mechanism which manages the unit allocation for all REE's in the Runtime.
Action	Used in Batch Control: all commands which are used for editing a recipe e.g. insert phase, testing recipes etc.
Begin simultaneous sequence	Element that ensures the separation of the recipe process in two or more sequence selections.
Begin sequence selection	Element that makes it possible to separate a recipe in two or more sequence selections of which only one can be active at a time. Each following sequence selection must start with a transition. The transition defines which sequence selection is executed in the recipe process.
Begin element	Element of module Batch Control with which every recipe begins.
Active element	Position in a recipe in the batch control module where the processing is interrupted in a semi-automatic and manual mode and the active elements are put into a pause status. With the command «next step» the process is resumed from this position.
Batch Control	Tool for creating master recipes and creating and executing control recipes in accordance with ISA-S88.
Batch operation	Automatic and sequential processing of a stack of single operations.
End simultaneous sequences	Element that combines the separation of the recipe process into two or more sequence selections back into one sequence selection.
End sequence selection	Element which brings together a sequence selection started by a begin sequence selection element.
End element	Element of module Batch Control with which every recipe ends.
Phase	Predefined process consisting of input interlocking, command and return tags, a phase done condition, event reactions, etc... (ISA 88: Phase)

Command	Used in Batch Control: a command which intervenes in the recipe process e.g start, stop, mode change etc.
Matrix recipe	Recipe in the Batch Control module which was created with the Matrix editor.
Simultaneous sequence	Area of module Batch Control. A simultaneous sequence starts with a begin simultaneous sequence element and is brought together with an end imultaneous sequence element to one execution branch. Between there are at least two sequence selections which are executed at the same time.
PFC recipe	Recipe in the batch control module, which was created with the PFC Editor
REE - Recipe Execution Engine	Part of module Batch Control for process control of recipes. The engine executes a control recipe and manages the entire process of the recipe.
Recipe	In recipes related data such as machine parameters or format data are summed up. This data can be transferred from the control system to the control and vice versa in one step. We differentiate between standard recipes and RGM recipes.The procedure is defined additionally to data in Batch Control Module in a recipe. It is distinguished between Matrix recipe and PFC Recipe.
Jump target	Element of the batch control module which allows a direct jump to a defined location of a sequence selection.
Control recipe	Part of the batch control module. Contains the process of a production process on basis of the batch process according to standard ISA S88. A control recipe is always derived from a template recipe and can be implemented once only. (ISA 88 Control Recipe)
Operation	Recipes can be divided into individual parts within the batchcontrol module. Operation management takes place via a centrallibrary. Instances of operations can be added within the recipe. Tags of the applied phase can be edited, the structure can only be edited in the operation template.
Transition	Element of module Batch Control which contains a condition. The element is used after phases in order to ensure a defined transition from one phase to another.
Connection line	Part of the connector in the Batch Control module: positions the connection point of the element.
Connection point	Part of the connection element in the Batch Control module. Connects two elements with eachother (e.g. phase with phase or phase with line). It changes color when the mouse pointer is on it.
Connection element	A possibility in the Batch Control of connecting elements with one another. It consists of a connection point and a connection line.
Sequence selection	Area of module Batch Control which ensures a separation in two or more sequence selections, of which only one can be active at the recipe process. It is an either/or sequence selection. A sequence selection always starts with a begin sequence selection element and ends with an end sequence selection element.

Master recipe	Part of the batch control module. Contains the process of a production process on basis of the batch process according to standard ISA S88. A recipe consists of the following components: basic functions, transitions,parallel circuits etc (ISA 88: Master recipe). Template recipes serve as templates for control recipes.
sequence selection	Execution area for the Batch Control Module Basic functions, transitions and transfer targets can be placed on it.

Перелік посилань

1. Автоматизація об’єктів періодичної діїЇ: лабораторний практикум для студентів освітнього ступеня «Магістр» спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньо-професійної програми «Комп'ютерні технології та програмування в автоматизованих системах управління» денної та заочної форм навчання. / уклад.: О. М. Пупена, О. М. Клименко, М.О.Шоронова; Нац. ун-т харч. технол. - Київ : НУХТ, 2018. - 119 с.
2. Пупена О.М. Розроблення людино-машинних інтерфейсів та систем збирання даних з використанням програмних засобів SCADA/HMI.: Навч. посіб. Київ : Видавництво Ліра-К, 2020. — 594 с.
3. Пупена О.М., Ельперін І.В. Програмування промислових контролерів у середовищі Unity Pro.Навч. посібник., — К.: Видавництво Ліра-К. — 2013. —340с.
4. Batch Control Training Documentation. 2015 Ing. Punzenberger COPA-DATA GmbH
5. David W. Brown, Introduction to S88 For the improvement of the design of batch systems, Japan Batch Forum, 2007.
6. Gordon Roney, Andrew Blankenship, How a Flexible Batch Application Can Save Costs, ISA-88 Implementation Experiences, 2010.
7. Clark L. Case, Applying ISA-88.01 to Small, Simple Processes, ISA-88 Implementation Experiences, 2010.
8. Martin Zeller, Dipl-Ing, Batch Data Analysis, ISA-88 Implementation Experiences, 2010.
9. Giovanni Godena, Igor Steiner, Janez Tancek, Marko Svetina, Design of a Batch Process Control Tool on the Programmable Logic Controller Platform, ISA-88 Implementation Experiences, 2010.
10. Dennis Brandl, Powering Project Productivity Using Best Practice Standards, CSIA Executive Conference, 2013.
11. William Hawkins, Dennis Brandl, Walt Boyes, ISA-88 Implementation Experiences, Worlds Batch Forum, 2010
12. Applying ISA-88 in Discrete and Continuous Manufacturing, 221 p., LLC 2010