

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**

**А.М.Ніколаєнко, О.М. Пупена, І.В.Ельперін**

# **Програмне забезпечення проєктованих логічних контролерів**

**Конспект лекцій.**

*для студентів напрямку 151 "Автоматизація та комп'ютерно-інтегровані  
технології" денної та заочної форм навчання.*

**СХВАЛЕНО**  
на засіданні кафедри автоматизованого  
управління технологічними процесами  
Протокол № 1 від 30.09.2020р.

Запоріжжя

# 1. Апаратне забезпечення мікропроцесорних контролерів Modicon M340

## 1.1 Архітектура M340

**Modicon M340** – промисловий контролер фірми Schneider Electric, для програмування якого використовується програмне забезпечення **UNITYPRO**. Modicon M340 – контролер модульного типу, конфігурація якого вибирається в залежності від кількості входів-виходів і алгоритму управління. Модулі кріпляться на **шасі**, яке виконує механічну та електричну функції. Така конструкція дає можливість гарячої заміни модулів без зупинки контролера. M340 може включати від 1-го до 4-х шасі з різною кількістю місць для установки модулів (від 4-х до 12-ти), об'єднаних між собою **BusX** шиною, загальною довжиною до 30 м.

Конструктивно M340 може складатись з таких основних елементів (рис.1.1):

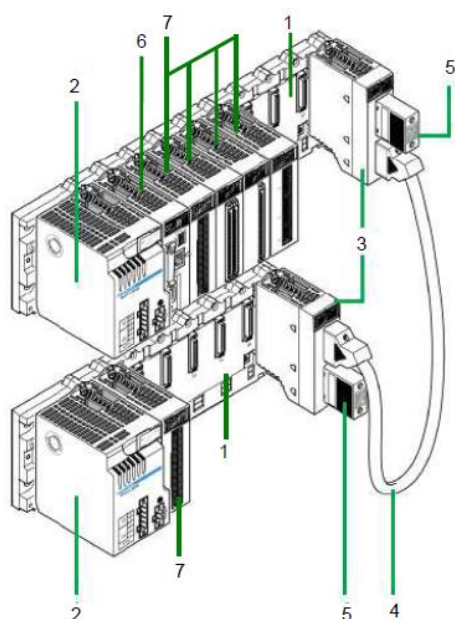
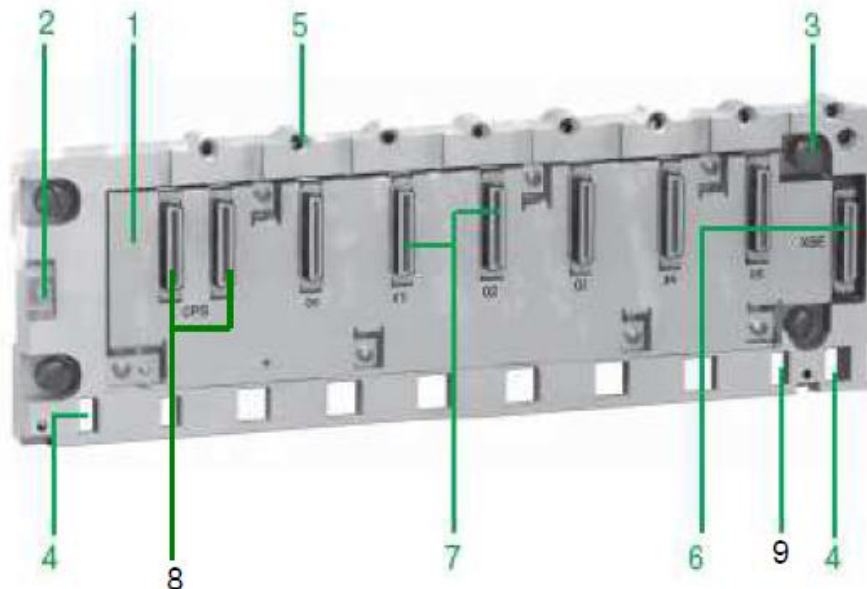


Рисунок 1.1–Архітектура Modicon M340

1. Шасі, на яких встановлюються модулі.
2. Модуль живлення, який обов'язково повинен бути присутнім в кожному шасі, і який встановлюється на спеціально відведеному місці у шасі
3. Модуль розширення для контролерів побудованих на базі декількох шасі.
4. Кабелі розширення BusX, що з'єднує модулі розширення на суміжних шасі.
5. Термінуючі резистори в кінцевих модулях розширення архітектури M340.
6. Процесорний модуль, який обов'язково розміщується в посадочному місці з номером 00 у шасі, яке має номер 0.
7. Модулі вводу/виводу та модулі спеціального призначення, які розміщуються в будь якому посадочному місці.

Основним конструктивним елементом контролера є шасі (рис.1.2). З одного боку, шасі використовується як конструктивний елемент, на якому розміщуються й закріплюються окремі модулі контролера, з іншого – шасі має загальну шину BusX, по якій відбувається як живлення модулів, установлених в шасі, так і обмін сигналами та даними між окремими модулями контролера. Шасі може кріпитися як на стандартну DIN-рейку так і з допомогою гвинтів. Шасі відрізняються за кількістю місць для встановлення модулів, відповідно на 4 (BMX XBP 0400), 6 (BMX XBP 0600), 8 (BMX XBP 0800) та 12 (BMX XBP1200) позицій. У разі необхідності використовувати велику кількість модулів контролер може складатись з декількох шасі (рис.1.2).



1- металева рама; 2- клемма заземлення; 3- отвори для кріплення шасі; 4- кріплення для заземлення екранів кабелів; 5- різьбові отвори під гвинт для закріплення кожного модуля; 6- рознім для модуля розширення (маркований як ХВЕ); 7- розніми для процесорного модуля, модулів вводу/виводу, комунікаційних модулів та модулів спеціального призначення; 8- розніми для модуля живлення (маркований як СРС); 9- отвори для установочних штирів модулів.

Рисунок 1.2 – Шасі Modicon M340

У цьому випадку в рознім ХВЕ кожного шасі встановлюються модулі розширення ХВЕ 1000, які з'єднуються BusX кабелем (кутовим ВМХ ХВС •••К або прямим TSX СВУ •••К, де ••• - довжина в дециметрах). Кожен модуль розширення має перемикач за допомогою якого виставляється адреса шасі в діапазоні від 0 до 3 (рис.1.3).

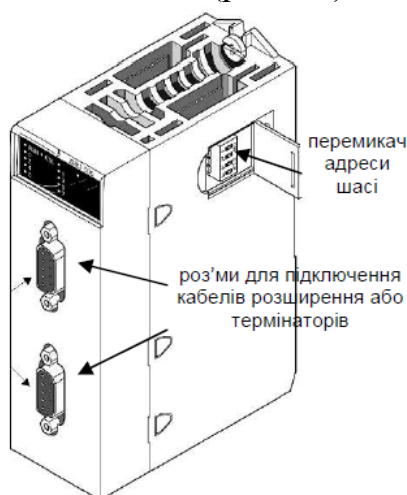


Рисунок 1.3 – Модуль розширення ХВЕ 1000ВМХ СРС•••

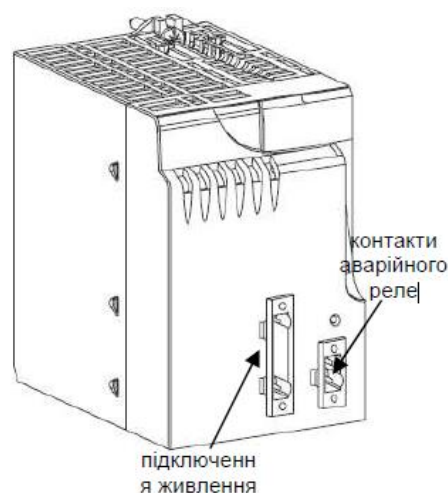


Рисунок 1.4 – Модуль живлення

Послідовність адресації шасі може не співпадати з їх фізичним розміщенням, однак процесорний модуль завжди повинен знаходитись в шасі за номером 0. В кінцевих модулях розширення встановлюють термінатори шини TSX TLY EX типу А та В, відповідно у вхідний рознім – для першого

модуля розширення та вихідний – для останнього. Всі модулі шасі, включно процесорний, живляться по внутрішній шині від модуля живлення VMX CPS... (рис.1.4). Модуль живлення підбирається по типу живлення (постійний або змінний струм) та споживаної потужності і вставляється у кожне шасі в різних з маркуванням CPS (рис.1.2). Розрахунок споживаної потужності залежить від кількості і типу модулів, які встановлюються у шасі, приклад якого розглянутий у розділі 1.5. Цей розрахунок можна також виконати у програмування UNITYPRO в процесі конфігурування апаратної частини ПЛК. Будь-який модуль живлення M340 має аварійне реле, яке відключається при зупинці контролера, або коли система самодіагностики виявить некоректне значення вихідної напруги модуля живлення. .

## 1.2 Процесорні модулі

Процесорні модулі M340 відрізняються функціональними можливостями, швидкістю обробки інструкцій, кількістю входів/виходів, які може обробляти контролер, кількістю спеціальних каналів, об'ємом доступної оперативної пам'яті та вбудованими в модуль ЦПУ комунікаційними засобами.

Таблиця 1.1. Загальні характеристики процесорних модулів

Характеристика		VMX P34 1000	VMX P34 2000	VMX P34 2010	VMX P34 2020	VMX P34 2030
Макс. кількість	шасі	2	4			
	дискретних вх+вих.	512	1024			
	аналогових вх+вих	128	256			
	лічильних каналів	20	36			
Об'єм RAM	загальний розмір	2048 Кб	4096 Кб			
	для програм, констант, символів	1792 Кб	3584			
	для даних	128 Кб	256 Кб			
Макс. кількість об'єктів	локалізовані внутрішні біти %Mi	16250	32464			
	локалізовані внутр. слова %MWi		32464			
	нелокалізовані внутрішні дані	128 Кб	256 Кб			
вбудовані комунікації	послідовний RS-485/RS-232C	+	+	+	+	-
	Ethernet TCP/IP	-	-	-	+	+
	CANOpen	-	-	+	-	+

У кожному процесорному модулі M340 є вбудований USB-інтерфейс (рис.1.5, поз 3), який призначений для підключення терміналу програмування (комп'ютер з встановленим UNITY PRO), а також для з'єднання зі операторськими станціями з встановленим програмним забезпеченням SCADA/HMI, а також з операторськими панелями. Для цього можна використати спеціальний екранований кабель, який поставляється у комплекті з процесорним модулем M340, або стандартний USB кабель з роз'ємом mini B. У будь-якому випадку довжина кабелю не може перевищувати 5 м.

У спеціальному слоті (рис.1.5, поз 4) розміщується SD-карта пам'яті. На карті, що входить у комплект стандартної поставки M340 (об'ємом 8 Мбайт), зберігається загрузочний проект, вбудовані діагностичні Веб-сторінки, а також при необхідності вихідний код проекту, константи та діалогові таблиці. Альтернативний варіант – використання карти обсягом 128 Мб, з підтримкою збереження даних користувача з прикладної програми, а також файлових операцій через FTP Сервер.



1-гвинт для закріплення модуля на шасі; 2-блок індикації;3-рознім USB mini B для підключення терміналу програмування, або засобів SCADA/НМІ;4-відсік для карти пам'яті;5-рознім RJ45 для підключення кабелю послідовного інтерфейсу RS-485 та RS-232C, поModbus RTU/ASCII або символного режиму (маркування чорним кольором);6-рознім для підключення кабелю Ethernet TCP/IP 10BASE-T/100BASE-TX (маркування зеленим кольором).

Рисунок 1.5– Процесорні модулі Modicon M340

Кожний процесорний модуль може вміщувати один або два вбудовані комунікаційні канали з комбінації (рис 1.5): послідовний Modbus Serial RS-232/RS-485, Ethernet TCP/IP та CANOpen. Крім функцій обміну з іншими пристроями системи, Modbus RTU (Serial) та Modbus TCP/IP (Ethernet) забезпечують доступ терміналу програмування UNITY PRO до контролера.

### 1.3 Дискретні модулі.

**1.3.1 Загальна характеристика.** Модулі дискретних входів/виходів M340 являють собою стандартні модулі, які займають один слот. Ці модулі відрізняються за типом каналів (вхідні, вихідні, змішані), за кількістю каналів, за типом вхідних та вихідних каналів і за способом підключення. Ці модулі можна встановлювати у будь-яке посадочне місце шасі, окрім місця для живлення (PS) та процесорного модуля. Дозволяється гаряча заміна модулів (при включеному живленні).

**1.3.2 Типи модулів.** Дискретні модулі можуть мати входи/виходи постійного струму (DC) на 24 VDC та 48 VDC з позитивною (sink) або негативною (source) логікою підключення, або змінного струму (AC) на 100-240 VAC.

Доступні модулі з транзисторними або релейними виходами. Виходи можуть бути захищені від короткого замикання. Всі дискретні входи та виходи ізолювані від внутрішньої шини. У таблиці 1.2 наведені основні технічні характеристики дискретних модулів.

### 1.3.3 Способи підключення.

Дискретні модулі за способом підключення зовнішніх сигналів можуть бути з 20-контактною з'ємною клемною колодкою (рис.1.6 варіант А) або з 40 контактними з'єднувальними рознімами (рис.1.6 варіант Б).

Таблиця 1.2 Основні технічні характеристики дискретних модулів.

Позначення модуля	Кількість каналів	Характеристики каналів	Підключення
Модулі дискретних входів			
BMX DDI1602	16	24 VDC, позитивна логіка	20-конт. з'ємна кол.
BMX DDI1603	16	48 VDC, позитивна логіка	20-конт. з'ємна кол.
BMX DAI1602	16	24 VDC негативна логіка або 24 VAC	20-конт. з'ємна кол.
BMX DAI1603	16	48 VAC	20-конт. з'ємна кол.
BMX DAI1604	16	100..120 VAC	20-конт. з'ємна кол.
BMX DDI3202K	32	24 VDC, позитивна логіка	40-конт. роз'єм
BMX DDI6402K	64	24 VDC, позитивна логіка	два 40-конт. роз'єми
Модулі дискретних входів та виходів (змішані)			
BMX DDM16022	8 Вх	24 VDC, позитивна логіка	20-конт. з'ємна кол.
	8 Вих	24 VDC, захищені, позитивна логіка, 0.5 А	
BMX DDM16025	8 Вх	24 VDC, позитивна логіка	20-конт. з'ємна кол.
	8 Вих	релейні VDC/VAC, незахищені, 2 А	
BMX DDM3202K	16 Вх	24 VDC, позитивна логіка	40-конт. роз'єм
	16 Вих	24 VDC, захищені, позитивна логіка, 0.1 А	
Модулі дискретних виходів			
BMX DDO3202K	32	24 VDC, захищені, позитивна логіка, 0.1 А	40-конт. роз'єм
BMX DDO6402K	64	24 VDC, захищені, позитивна логіка, 0.1 А	два 40-конт. роз'єми
BMX DDO1602	16	24 VDC, захищені, позитивна логіка, 0.5 А	20-конт. з'ємна кол.
BMX DDO1612	16	24 VDC, захищені, негативна логіка, 0.1 А	20-конт. з'ємна кол.
BMX DAO1605	16	тиристорні 100...240VAC, незахищені, 0.6 А	20-конт. з'ємна кол.
BMX DRA0805	8	релейні VDC/VAC, незахищені, 3 А	20-конт. з'ємна кол.
BMX DRA1605	16	релейні VDC/VAC, незахищені, 2 А	20-конт. з'ємна кол.



1- корпус; 2- маркування модуля; 3- панель індикації станів каналів; 4 – рознім для підключення з'ємної клемної колодки (варіант А) або виносної клемної колодки (варіант Б)

Рисунок 1.6– Зовнішній вигляд дискретних модулів з різними варіантами підключення

Для модулів з клемною колодкою (варіант А) додатково замовляється 20-контактна з'ємна клемна колодка BMX FTB 20•0, або готовий кабель, який на одному кінці має клемну колодкою, а на іншому вільні провідники (з розпушеними кінцями) з кольоровим маркуванням (рис.1.8, а). Існують три види 20-контактних клемних колодок:

- гвинтова клемна колодка BMX FTB 2000;

- колодка з гвинтовими зажимами ВМХ FTВ 2010;

- пружинна клемна колодка ВМХ FTВ 2020;

З'ємні клемні колодки поставляються з аксесуарами для кодування, що дає

можливість забезпечити унікальний механічний ключ для кожної пари – модуль-клемна колодка (рис.1.7). Іншими словами, кодування виключає можливість підключення клемної колодки, яка була встановлена на модулі до іншого модуля.

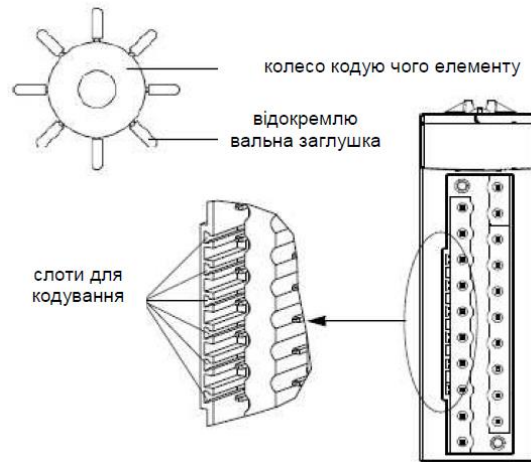


Рисунок 1.7– Механічне кодування модулів

Модулі з рознімами (варіант Б) на 32 канали мають один 40-контактний роз'єм, на 64 канали – два розніми. До таких модулів додатково замовляються спеціальні кабелі з 40-контактним з'єднувачем в одному з двох варіантів:

- FCW••3, які з іншого боку мають розпушений кінець з кольоровим маркуванням провідників (рис.1.8. б);

- FCC••3, які з іншого боку мають два з'єднувачі HE10 для підключення до

виносних клемних колодок типу Telefast ABE (рис.1.8.в).

Підключення з використанням кабелів з розпушеним кінцем проводиться через додаткову клемну колодку довільного виробника.

Підключення модулів через кабелі з HE10 з'єднувачами проводиться тільки з використанням спеціальних виносних блоків з клемними колодками системи швидкого монтажу **Telefast ABE**. Schneider Electric пропонує дуже велику гаму блоків Telefast для дискретних модулів, які відрізняються:

- кількістю та типом каналів, які обслуговує даний блок;

- типом клем (гвинтові, пружинні);

- наявністю розподілення живлення;

- наявністю гальванічних розв'язок між каналами, між блоком та дискретним модулем;

- вбудованими додатковими функціями перетворення сигналу (вбудовані або з'ємні твердотільні або електромеханічні реле на різні потужності)

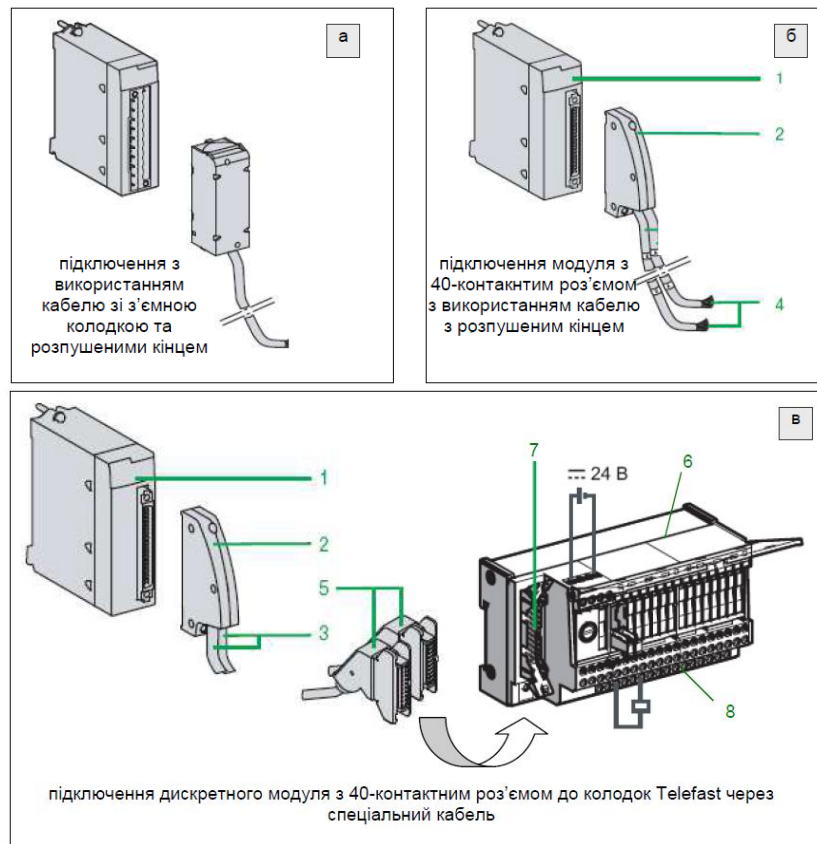
- наявністю додаткових функцій захисту;

- наявністю світлових індикаторів;

- наявністю можливості ручного включення/відключення сигналу;

- іншими додатковими опціями.

Усі блоки Telefast мають змінний плавкий запобіжник, який захищає входи/виходи модуля від перевантаження.



1 - дискретний модуль; 2 - 40-контактний рознім; 3 – кабель FCC••3; 4 – розпушений кінець кабелю; 5 – з'єднувачі типу HE10 для підключення до виносних клемних колодок типу Telefast; 6 – виносна клемна колодка типу Telefast; 7 – рознім типу HE10; 8 – клеми для підключення зовнішніх сигналів;

Рисунок 1.8– Способи підключення технічних засобів до дискретних модулів

Одним із універсальних блоків Telefast для дискретних входів/виходів є ABE7H16R21, який може підключатися до будь яких модулів з 40-контактним з'єднувачем з використанням кабелю FCC••3 (•• - залежить від довжини кабелю). Він використовується для підключення 16 дискретних входів або 16 дискретних виходів окремими парами гвинтових клем колодки.

Перелік необхідних аксесуарів для дискретних модулів зведений в таблицю 1.3. У таблиці 1.3 не наведений перелік аксесуарів для способів підключення кабелів з розпушеним кінцем та клемних колодок з підключенням до Telefast. У таблиці 1.3 також наведений тільки один варіант блоку Telefast.

Таблиця 1.3 Монтажні аксесуари для підключення дискретних модулів.



Позначення модуля	Тип підключення	Спосіб підключення
Модулі дискретних входів		
BMX DDI1602	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DDI1603	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DAI1602	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DAI1603	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DAI1604	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DDI3202K	40-контактний роз'єм	кабель FCC••3 (від 0.5 до 10 м) + Telefast ABE 7H16R21 – 2 шт.
BMX DDI6402K	40-контактний роз'єм	(кабель FCC••3 (від 0.5 до 10 м) + Telefast ABE 7H16R21 – 2 шт.) – 2 комплекти
Модулі дискретних входів та виходів (змішані)		
BMX DDM16022	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DDM16025	20-контактна з'ємна колодка	з'ємна клемна колодка BMX FTB 20•0
BMX DDM3202K	40-контактний роз'єм	кабель FCC••3 (від 0.5 до 10 м) + Telefast ABE 7H16R21 – 2 шт.
Модулі дискретних виходів		
BMX DDO3202K	40-контактний роз'єм	кабель FCC••3 (від 0.5 до 10 м) + Telefast ABE 7H16R21 – 2 шт.
BMX DDO6402K	два 40-контактні роз'єми	(кабель FCC••3 (від 0.5 до 10 м) + Telefast ABE 7H16R21 – 2 шт.) – 2 комплекти

## 1.4 Аналогові модулі

**1.4.1 Загальна характеристика.** Модулі аналогових входів/виходів M340 являють собою стандартні модулі, які займають один слот. Як і дискретні модулі, аналогові відрізняються за типом каналів (вхідні, вихідні, змішані), за кількістю каналів, за характеристикою і діапазоном сигналів (напруга, струм, термометри опору, тощо), наявністю гальванічного розподілення і за способом підключення. Ці модулі можна встановлювати у будь-яке посадочне місце шасі, окрім місця для живлення (PS) та процесорного модуля. Дозволяється гаряча заміна модулів (при включеному живленні).

**1.4.2 Типи модулів.** Перелік всіх типів аналогових модулів M340 наведений в таб.1.4. Аналогічно аналоговим модулям Modicon Premium, аналогові вхідні модулі M340 виконують функції:

- сканування вхідних каналів різного діапазону за допомогою безконтактного мультиплексування;
- аналогово-цифрове перетворення;
- фільтрація сигналів;
- моніторинг модуля: тестування ланок перетворення, вхідний контроль перевищування рівня сигналу, тест наявності клемної колодки.

Модулі аналогових виходів виконують функції:

- цифро-аналогове перетворення;

- захист каналів модулів від перевантаження;
- моніторинг модуля: тест перетворення, тест виходу за межі, тест наявності клемної колодки.

Таблиця 1.4 Основні технічні характеристики аналогових модулів.

Позначення модуля	Кількість каналів	Діапазон сигналу	Характеристики каналів	Підключення
Модулі аналогових входів				
BMX ART 0414	4	мВ, термометри опору, термопари	16-бітні, ізоляція між каналами, час опитування модуля - 400 мс	40-контактний роз'єм
BMX ART 0814	8	мВ, термометри опору, термопари	16-бітні, ізоляція між каналами, час опитування модуля - 400 мс	40-контактний роз'єм
BMX AMI 0410	4	$\pm 10В, 0...10В, 0...5В, 0...20мА, 4...20 мА$	16-бітні, ізоляція між каналами, час опитування модуля - 5 мс	20-контактна з'ємна колодка
BMX AMI 800	8	$\pm 10В, 0...10В, 0...5В, 0...20мА, 4...20 мА$	16-бітні, з загальною точкою підключення, час опитування модуля - 9 мс	28-контактна з'ємна колодка
BMX AMI 810	8	$\pm 10В, 0...10В, 0...5В, 0...20мА, 4...20 мА$	16-бітні, ізоляція між каналами, час опитування модуля - 9 мс	28-контактна з'ємна колодка
Модулі аналогових входів та виходів (змішані)				
BMX AMM 0600	4 Вх	$\pm 10В, 0...10В, 0...5В, 0...20мА, 4...20 мА$	14-бітні для U, 12-бітні для I, загальна точка, час опитування модуля - 5 мс	20-конт. з'ємна кол.
	2 Вих	$\pm 10В, 0...20мА, 4...20 мА$	12-бітні для U, 11-бітні для I, загальна точка	
Модулі аналогових виходів				
BMX AMO 0210	2	$\pm 10В, 0...20мА, 4...20 мА$	16-бітні, ізоляція між каналами	20-конт. з'ємна кол.
BMX AMO 410	4	$\pm 10В, 0...20мА, 4...20 мА$	16-бітні, ізоляція між каналами	20-конт. з'ємна кол.
BMX AMO 802	8	$0...20мА, 4...20 мА$	16-бітні, загальна точка	20-конт. з'ємна кол.

### 1.4.3 Способи підключення.

Подібно дискретним модулям за способом підключення зовнішніх сигналів, аналогові модулі можуть бути: з 20-контактною з'ємною клемною колодкою, з 28-контактною клемною колодкою або з 40-контактними з'єднувальними рознімами. З'ємні клемні колодки поставляються з аксесуарами для кодування, що дає можливість забезпечити унікальний механічний ключ для кожної пари – модуль-клемна колодка.

Для модулів, які підключаються через з'ємну клемну колодку, остання замовляється окремо як у звичайному варіанті, так і з підключеним кабелем розпушеним на кінці. 20-контактні клемні колодки для аналогових модулів такі самі як і для дискретних (BMX FTB 2000/2010/2020). Для модулів BMX AMI 800/810 замовляється 28-контактна клемна колодка BMX FTB 2820 з пружинними затискачами.

Підключення модулів BMX ART 0414/0814 з 40-контактними рознімами можна виконувати з використання блоків Telefast ABE 7CPA412 та кабелю

BMXFCA ••2, (де •• залежить від довжини). Такий спосіб підключення схематично зображений на рис.1.13.

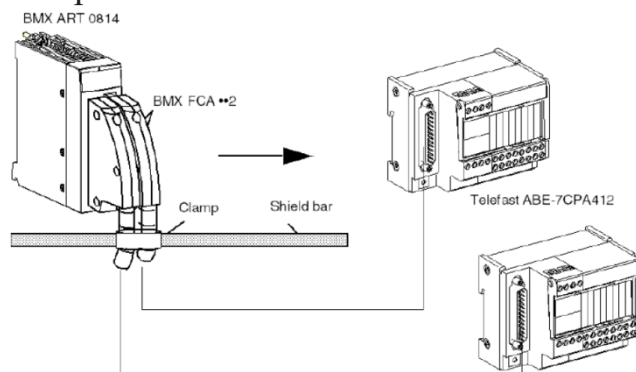


Рисунок 1.13– Підключення Telefast ABE 7CPA412 до модуля BMX ART 0814.

Перелік необхідних аксесуарів для аналогових модулів зведений в табл.1.5.

Таблиця 1.5. Монтажні аксесуари для підключення аналогових модулів.

Позначення модуля	Тип підключення	спосіб підключення
Модулі аналогових входів		
BMX ART 0414	40-конт. роз'єм	кабель BMX FCA ••2 (1.5, 2 або 5 м) + Telefast ABE 7CPA412
BMX ART 0814	два 40-конт. роз'єми	(BMX FCA ••2(1.5, 2 або 5 м) + Telefast ABE 7CPA412) – 2 комплекти
BMX AMI 0410	20-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 20•0
BMX AMI 800	28-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 2820
BMX AMI 810	28-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 2820
Модулі аналогових входів та виходів (змішані)		
BMX AMM 0600	20-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 20•0
Модулі аналогових виходів		
BMX AMO 0210	20-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 20•0
BMX AMO 410	20-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 20•0
BMX AMO 802	20-конт. з'ємна кол.	з'ємна клемна колодка BMX FTB 20•0

### 1.5 Вибір модуля живлення.

Споживана потужність, яка необхідна для модулів, встановлених на монтажному шасі, залежить від типу даних модулів. Тому для того щоб вірно вибрати модуль живлення, який забезпечить необхідне споживання необхідно проводити розрахунок енергоспоживання,

Живлення модулів по шасі проводиться через два виходи модуля живлення 24В (24V\_VAC) та 3,3 В (3V3\_VAC). Вихід 24V\_VAC використовується для живлення встановлених на монтажному шасі модулів входів-виходів та процесорного модуля, а вихід 3V3\_VAC використовується тільки для живлення модулів входів-виходів.

Модулі живлення, що живляться напругою 100...240VAC (BMX CPS 2000 та BMX CPS 3500), додатково мають зовнішній вихід 24 В (24V\_SENSORS), який можна використати для живлення датчиків або виконавчих механізмів.

Під час проведення розрахунку енергоспоживання модулів живлення BMX CPS2000/3500 необхідно враховувати наступні вимоги:

- сумарна потужність, яка споживається по виходах модуля живлення 3V3\_BAC, 24V\_BAC та 24V\_SENSORS не повинна перевищувати загальну корисну потужність модуля живлення:

$$I_{3V3\_BAC} * 3,3 \text{ В} + I_{24V\_BAC} * 24 \text{ В} + I_{24V\_SENSORS} * 24 \text{ В} \leq P_{PS} \quad (1.1)$$

- сумарна потужність, яка споживається по двом виходам модуля живлення 3V3\_BAC та 24V\_BAC не повинна перевищувати загальну корисну потужність модуля живлення по цим виходам:

$$I_{3V3\_BAC} * 3,3 \text{ В} + I_{24V\_BAC} * 24 \text{ В} \leq P_{3V3\_24V} \quad (1.2)$$

В формулах (1.1) та (1.2)  $I_{3V3\_BAC}$  – сумарний споживаний струм модулями по 3V3\_BAC,  $I_{24V\_BAC}$  – сумарний споживаний струм модулями по 24V\_BAC,  $I_{24V\_SENSORS}$

- сумарний споживаний струм по зовнішньому виходу 24V\_SENSORS,  $P_{PS}$  – загальна корисна потужність модуля живлення,  $P_{3V3\_24V}$  – максимальна сумарна потужність модуля на виходах 3V3\_BAC та 24V\_BAC.

При розрахунку енергоспоживання модулів живлення BMX CPS 2010/3020

необхідно враховувати наступну вимогу:

- сумарна потужність, яка споживається по виходах модуля живлення 3V3\_BAC та 24V\_BAC не повинна перевищувати загальну корисну потужність модуля живлення:

$$I_{3V3\_BAC} * 3,3 \text{ В} + I_{24V\_BAC} * 24 \text{ В} \leq P_{PS} \quad (1.3)$$

У таблиці 1.6 наведені характеристики потужності модулів живлення по різним виходам. Зокрема звідти можна вибрати величини  $P_{PS}$  та  $P_{3V3\_24V}$  при виборі конкретного модуля живлення.

Таблиця 1.6.Характеристик потужності модулів живлення.

Потужність	BMX CPS 2000	BMX CPS 2010	BMX CPS 3020	BMX CPS 3500
Напруга живлення	100...240 VAC	24 VDC	24...48 VDC	100...240 VAC
Загальна корисна потужність ( $P_{PS}$ )	20 Вт	17 Вт	32 Вт	36 Вт
Потужність на виході 3V3_BAC монтажного шасі	8,3 Вт (2,5 А)	8,3 Вт (2,5 А)	15 Вт (4,5 А)	15 Вт (4,5 А)
Потужність на виході 24V_BAC монтажного шасі	16,5 Вт (0,7 А)	16,5 Вт (0,7 А)	31,2 Вт (1,3 А)	31,2 Вт (1,3 А)
Максимальна сумарна потужність на виходах 3V3_BAC та 24V_BAC ( $P_{3V3\_24V}$ )	16,5 Вт	16,5 Вт	31,2 Вт	31,2 Вт
Сумарна корисна потужність на споживання зовнішніми датчиками 24V_SENOSRS	10,8 Вт (0,45 А)	-	-	21,6 Вт (0,9 А)

Величини  $I_{3V3\_BAC}$  та  $I_{24V\_BAC}$  розраховуються по середньому споживанню модулів, які наведені в таблиці 1.7. Ці значення представляють собою величину між максимальним споживаним струмом та нормальним споживаним струмом.

Використовуючи дані з цієї таблиці, можна вирахувати сумарний споживаний струм по кожному монтажному шасі для кожного виходу модуля

живлення, у відповідності з набором встановлених на монтажному шасі модулів.

## 1.6 Приклади.

### Приклад 1.1 Конфігурування контролера М340

**Завдання.** Підберіть складові ПЛК М340 для забезпечення його роботи за умов,

наведених в таблиці 1.8. Вибрані складові сформуєте у вигляді відомості.

#### **Рішення.**

Конфігурування ПЛК М340 будемо проводити у такій послідовності:

1. Вибір процесорного модуля.
2. Вибір модулів вводу/виводу.
3. Вибір аксесуарів для модулів вводу/виводу.
4. Компонування шасі, вибір додаткових модулів та аксесуарів для шасі.
5. Вибір модулів живлення.

**1. Вибір процесорного модуля.** Процесорний модуль підбираємо по наступним критеріям: кількість каналів кожного типу, тип та кількість комунікаційних каналів, об'єм пам'яті користувача.

- кількість аналогових входів + виходів:  $26+15+20+19=80$

- кількість дискретних входів + виходів:  $59+67+74+23+22+21=266$

Таблиця 1.8. Вихідні дані завдання

Вимоги	Кількість або наявність
Живлення ПЛК (24 VDC або 220 VAC)	220 VAC
Кількість дискретних входів 24 VDC, позитивна логіка	59
Кількість дискретних входів 115 VAC	67
Кількість аналогових входів 4-20 mA	26
Кількість аналогових входів 0-10 V	15
Кількість дискретних виходів струм комутації до 0,1 A 24 VDC	74
Кількість дискретних виходів струм комутації до 0,5 A 24 VDC	23
Кількість дискретних виходів струм комутації до 2 A 24 VDC	22
Кількість дискретних виходів струм комутації до 3 A 24 VDC	21
Кількість аналогових виходів 0-10 V	20
Кількість аналогових виходів 4-20 mA	19
Підключення по Modbus RTU на RS485	+
Підключення по Modbus/TCP на Ethernet	+
Підключення по CANOpen	-
Програма користувача займає (Кб)	2500

По таблиці 1.1 підбираємо процесорний модуль. По кількості каналів будь

який процесорний модуль задовольняє умовам задачі, тому орієнтуємось тільки на кількість пам'яті та наявним комунікаціям. Серед процесорних модулів М340 підходить тільки ВМХ Р34 2020.

### **2. Вибір модулів вводу/виводу проводиться по таблицям 1.2 та 1.4.**

59 ВД 24 VDC - ВМХ DDI6402K - 1 шт. (64, 5 вільних)

67 ВД 115 VAC - ВМХ DAI1604 - 4 шт. ( $5*16=80$ , 13 вільних)

21 ДВ 3 A - ВМХ DRA 0805 - 3 шт. ( $8*3=24$ , 3 вільних)

22 ДВ 2 A - ВМХ DRA 1605 - 2 шт. ( $16*2=32$ , 10 вільних)

23 ДВ 0,5 A - ВМХ DDO 1602 - 2 шт. ( $16*2=32$ , 9 вільних)

74 ДВ 0,1 А - BMX DDO 6402K – 1 шт. (64 + 9 з BMX DDO 1602 + 1 з BMXDRA 1605)

26+15 ВА – BMX AMI 810 – 5 шт. (8\*5=40) + BMX AMI 0410 – 1 шт. (4)

20 АВ U – BMX AMO 410 – 5 шт (4\*5=20)

19 АВ I - BMX AMO 802 – 3 шт. (8\*3=24)

З метою зменшення вартості системи, для дискретних вихідних каналів малих струмів комутації ми задіяли вільні канали модулів з більшими струмами комутації.

Слід зазначити, що наведений варіант компонування не єдине рішення для даної задачі, тобто може бути декілька варіантів компонування ПЛК модулями вводу/виводу.

**3. Вибір аксесуарів для модулів вводу/виводу проводиться по таблицям 1.3 та 1.5.** Зведемо аксесуари та модулі в одну таблицю.

Таблиця 1.9.

Модуль вводу/виводу		Аксесуари		
Найменування	Кількість	Найменування	Кількість	примітка
BMX DDI6402K	1	BMX FCC053	2	кабель 0.5 м, з двома HE10 з'єднувачами
		Telefast ABE 7H16R21	4	16-канальний клемний блок
BMX DAI1604	4	BMX FTB 2010	4	20 контактна з'ємна колодка з гвинтовими зажимами
BMX DRA 0805	3	BMX FTB 2010	3	20 контактна з'ємна колодка з гвинтовими зажимами
BMX DRA 1605	2	BMX FTB 2010	2	20 контактна з'ємна колодка з гвинтовими зажимами
BMX DDO 1602	2	BMX FTB 2010	2	20 контактна з'ємна колодка з гвинтовими зажимами
BMX DDO 6402K	1	BMX FCC053	1	кабель 0.5 м, з двома HE10 з'єднувачами
		Telefast ABE 7H16R21	2	16-канальний клемний блок
BMX AMI 810	5	BMX FTB 2820	5	28 контактна з'ємна клемна колодка
BMX AMI 0410	1	BMX FTB 2010	1	20 контактна з'ємна колодка з гвинтовими зажимами
BMX AMO 410	5	BMX FTB 2010	5	20 контактна з'ємна колодка з гвинтовими зажимами
BMX AMO 802	3	BMX FTB 2010	3	20 контактна з'ємна колодка з гвинтовими зажимами

#### **4. Вибір шасі, додаткових модулів та аксесуарів для шасі.**

Загальна кількість модулів разом з процесорним: 1 CPU + 10 DI + 3 DO + 6 AI + 8 AO = 28. Таким чином можна вибрати 2 монтажних шасі на 12 місць (BMX XBP 1200) та 1 шасі на 4місця (BMX XBP 0400). До кожного шасі необхідно по 1-му модулю розширення, які зв'язуються BusX кабелями. Тобто додатково треба замовити модулі XBE 1000 -3 шт., кабелі TSX СВУ 010K – 2 шт. та один набір з двох термінаторів TSX TLYEX.

Розміщення модулів в шасі вибираємо довільно. Однак на практиці, модулі з найбільшим споживанням струму розподіляють рівномірно між шасі.

Для зручності підрахунку споживаної потужності кожним модулем, розміщення покажемо у вигляді таблиці 1.10. В цій таблиці заповнюємо тільки поля шасі,місце та найменування модулів ( за виключенням модулів живлення - місце PS).

#### **5. Вибір модулів живлення.**

За умови задачі, живлення ПЛК проводиться напругою 220 ВАС. Таким чином, в залежності від споживаної потужності модулів на шасі, необхідно вибрати один з двох модулів BMX CPS 2000 або BMX CPS 3500.

Вважаємо, що вихід модуля для живлення датчиків та виконавчих механізмів(24V\_SENOSRS) не використовується. Для кожного модуля шасі за таблицею 1.7 визначаємо середні споживані струми і вписуємо їх в табл.1.10.

Таблиця 1.10. Споживаний струм використаних модулів

Шасі	Модуль		Середній споживаний струм, в мА	
	місце	Найменування модуля	На виході 3V3_BAC	На виході 24V_BAC
00	PS	BMX CPS 2000 $P_{3V3\_24V} = 16,5$ Вт	$I_{3V3\_BAC}=300$ мА, P=0.99Вт	$I_{24V\_BAC}=260$ мА, P=6.24Вт
			$\Sigma P=7,23$ Вт	
	00	BMX P34 2020	0	95
	01	BMX DRA 0805	100	55
	02	BMX DRA 0805	100	55
	03	BMX DRA 0805	100	55
	XBE			
01	PS	BMX CPS 2000 $P_{3V3\_24V} = 16,5$ Вт	$I_{3V3\_BAC}=1480$ мА, P=4,88 Вт	$I_{24V\_BAC}=460$ мА , P=11,04 Вт
			$\Sigma P=15,92$ Вт	
	00	BMX DDI 6402K	200	0
	01	BMX DAI 1604	90	0
	02	BMX DAI 1604	90	0
	03	BMX DAI 1604	90	0
	04	BMX DAI 1604	90	0
	05	BMX DRA 1605	100	95
	06	BMX DRA 1605	100	95
	07	BMX DDO 1602	90	0
	08	BMX DDO 1602	90	0
	09	BMX DDO 6402K	240	0
	10	BMX AMO 802	150	135
	11	BMX AMO 802	150	135
	XBE			
02	PS	BMX CPS 3500 $P_{3V3\_24V} = 31,2$ Вт	$I_{3V3\_BAC}=1800$ мА, P=5,94 Вт	$I_{24V\_BAC}=675$ мА , P=16,2 Вт
			$\Sigma P=21,96$ Вт	
	00	BMX AMI 810	150	54
	01	BMX AMI 810	150	54
	02	BMX AMI 810	150	54
	03	BMX AMI 810	150	54
	04	BMX AMI 810	150	54
	05	BMX AMI 0410	150	45
	06	BMX AMI 0410	150	45
	07	BMX AMI 0410	150	45
	08	BMX AMI 0410	150	45
	09	BMX AMI 0410	150	45
	10	BMX AMI 0410	150	45
11	BMX AMO 802	150	135	
	XBE			

В рядках з місцем PS в колонках 3V3\_BAC та 24V\_BAC вказуються сумарні споживані струми по кожному шасі, та споживані потужності ( $I*U$ ). По таблиці 1.6та з використанням формули (1.2) визначається який з модулів живлення підходить для живлення даного шасі.

Як видно з таблиці, для шасі 00 та 01 задовольняє модуль живлення BMX CPS2000, оскільки сумарна потужність по виходам 3V3\_VAC та 24V\_VAC не перевищує показники  $P_{3V3\_24V}$  та  $P_{PS}$  для даного модуля. Для шасі 02 сумарна потужність по цим виходам 21.96 Вт, що більше ніж максимально дозволена в BMX CPS 2000 (16,4Вт), отже необхідно вибирати модуль живлення BMX CPS 3500.

### **Результат.**

Перелік всіх необхідних складових M340 для даної задачі наведемо у вигляді відомості в таблиці 1.11[1].

Таблиця 1.11. Відомість обладнання для компонування ПЛК M340.

Позиція	Найменування	Кількість	Примітка
	BMX XBP 1200	2	шасі на 12 місць
	BMX XBP 0400	1	шасі на 4 місця
	XBE 1000	3	модуль розширення
	TSX TLY EX	1	комплект з двох термінаторів BusX
	TSX CBY 010K	2	BusX кабель 1 м
	BMX CPS 2000	2	модуль живлення 100...240 VAC, 20 Вт
	BMX CPS 3500	1	модуль живлення 100...240 VAC, 36 Вт
	BMX P34 2020	1	процесорний модуль, Serial + Ethernet
	BMX DDI6402K	1	дискретний вхідний модуль на 64 канали
	BMX DAI1604	4	дискретний вхідний модуль на 16 каналів
	BMX DRA 0805	3	дискретний вихідний модуль на 8 каналів
	BMX DRA 1605	2	дискретний вихідний модуль на 16 каналів
	BMX DDO 1602	2	дискретний вихідний модуль на 16 каналів
	BMX DDO 6402K	1	дискретний вихідний модуль на 64 канали
	BMX AMI 810	5	аналоговий вхідний модуль на 8 каналів
	BMX AMI 0410	1	аналоговий вхідний модуль на 4 канали
	BMX AMO 410	5	аналоговий вихідний модуль на 4 канали
	BMX AMO 802	3	аналоговий вихідний модуль на 8 каналів
	BMX FTB 2010	20	20 контактна з'ємна колодка з гвинтовими зажимами
	BMX FTB 2820	5	28 контактна з'ємна клемна колодка
	Telefast ABE 7H16R21	6	16-канальний клемний блок
	BMX FCC053	3	кабель 0.5 м, з двома NE10 з'єднувачами

## **2. Апаратне забезпечення контролера Modicon M580**

### **2.1 Центральні процесори M580**

Після більш ніж 20-річного використання на ринку промислової автоматизації загальнопромислових ЦПУ Quantum (140CPU \*\*\*\*\*) компанія Schneider Electric у 2019р. зняла їх з виробництва. На заміну контролерів Modicon Quantum для АСУТП було зроблено новітній контролер Modicon M580.

Modicon M580 - це перший в світі ePAC (programmable automation controller), основою якого є Ethernet. Процесор SPEAr має вбудований детермінований (*певний, ясний, конкретний, чіткий, точний; певний*) стандарт зв'язку Ethernet, який використовується для комунікацій з магістральною і внутрішньою шиною шасі, що спрощує інтеграцію обладнання в єдину систему. В результаті досягається новий рівень прозорості та продуктивності



без необхідності ручного налаштування кожного підключеного пристрою. Протокол Ethernet також використовується для роботи з будь-якими пристроями в мережі, наприклад: пристрої розподілу електроенергії, низьковольтні щити, які разом складають єдину відкриту систему управління підприємством.

Ядром M580 є сучасний двоядерний ARM-процесор сімейства Cortex-A9 SPEAr, який забезпечує найвищий рівень обчислювальної потужності і широкі можливості підключень, дозволяючи реалізовувати функції безпеки в контролері і підвищуючи його стійкість до кіберзагрозам. Можливість змінювання конфігурації в режимі реального часу не потребує зупинки процесу, для додавання або видалення модулів, зміни архітектури, навіть модифікування додатку.

Набір модулів ЦПУ представлений сімома типами процесорних пристроїв, три з яких доступні до замовлення як в стандартному, так і у виконанні для важких умов експлуатації (символ «Н» в кінці замовного номера):

ВМЕР581020 і ВМЕР581020Н; ВМЕР582020 і ВМЕР582020Н;  
ВМЕР582040 і ВМЕР582040Н; ВМЕР583040; ВМЕР584020;  
ВМЕР584040.

Всі ЦПУ підтримують роботу пристроїв розподіленого вводу-виводу (DIO) в мережі «Device Network». Ця можливість забезпечується службою «DIO scanner». Робота зі станціями віддаленого вводу-виводу (RIO) вимагає наявності у ЦПУ служби «RIO scanner». Тільки три типи ЦПУ з лінійки M580 мають дану службу в наявності:

- ВМЕР582040 і ВМЕР582040Н; ВМЕР583040; ВМЕР584040.

Основні характеристики продуктивності ЦПУ з точки зору використання станцій вводу-виводу, наведені в таблиці 2.1. З точки зору виконання процесором кількості інструкцій, ЦПУ характеризуються показниками наведеними в таблиці 2.2:

Таблиця 2.1 Характеристики продуктивності ЦПУ ПЛК M580

	BME P58 ....						
	1020(H)	2020(H)	2040(H)	3020	3040	4020	4040
Максимальное кол-во каналов дискретного ввода-вывода	1024	2048	2048	3072	3072	4098	4098
Максимальное кол-во каналов аналогового ввода-вывода	256	512	512	768	768	1024	1024
Максимальное кол-во модулей в сети RIO	-	-	64	-	128	-	128
Максимальное кол-во устройств DIO	64	128	64	128	64	128	64
Максимальное кол-во каналов Ethernet, исключая ЦПУ	2	2	2	3	3	4	4
Максимальное кол-во локальных шасси ввода-вывода (главное + расширение)	4	4	4	8	8	8	8
Максимальное кол-во островов удалённого ввода-вывода (до 2-х шасси на одну станцию)	-	-	8	-	16	-	16
Порты Ethernet							
Сервисный	1	1	1	1	1	1	1
RIO / DIO	-	-	2	-	2	-	2
DIO	2	2	-	2	-	2	-

Таблица 2.2 Характеристики продуктивности ЦПУ ПЛК M580

	BME P58 ....						
	1020(H)	2020(H)	2040(H)	3020	3040	4020	4040
Кол-во логических инструкций [Кинстр./мс.]	10	10	10	20	20	40	40
Кол-во типичных инструкций [Кинстр./мс.]	7,5	7,5	7,5	15	15	30	30

## 2.2 Переваги технології Ethernet IO

Технологія Ethernet Remote I/O об'єднує переваги відкритості, продуктивності і гнучкості промислової Ethernet мережі з можливостями ефективної інтеграції і детермінізму мережі віддаленого вводу-виводу (RIO). До переваг технології Ethernet також відносяться:

- можливість використання непатентованої і недорогої кабельної продукції як фізичного середовища передачі даних (включаючи одномодовий і багатомодовий оптичний кабель);
- більш висока продуктивність у порівнянні з коаксіальним середовищем при одночасному забезпеченні детермінізму передачі даних, тобто час скана «дозволу» логіки віддаленого вводу-виводу може бути визначено з достовірною точністю;
- можливість застосування архітектури мережі «відмовостійке кільце» як з використанням комутаторів, так і без таких;

- можливість інтегрувати недетерміновані пристрої мережі вводу-виводу, такі як Ethernet DIO і HMI-панелі в єдину мережу;
- можливість прямого доступу до вузлів ERIO, пристроїв EDIO і HMI з рівня мережі управління (мережа верхнього рівня, АРМ інженера);
- конфігурація всієї мережі ERIO в єдиному середовищі Unity Pro (за винятком сторонніх DIO пристроїв).

Крім того нові поліпшені можливості продуктивності Ethernet IO забезпечують:

- можливість використання RIO і Ethernet DIO в пристроях єдиній мережі;
- детермінованість обміну даними між ПЛК та RIO пристроями, виражена у визначенні точного часу відгуку системи не залежно від кількості DIO пристроїв в мережі;
- можливість інтегрувати недетерміновані пристрої мережі вводу-виводу, такі як Ethernet DIO і HMI-панелі в єдину мережу;
- можливість прямого доступу до вузлів ERIO, пристроїв EDIO і HMI з рівня мережі управління (мережа верхнього рівня, АРМ інженера);
- конфігурація всієї мережі ERIO в єдиному середовищі Unity Pro (за винятком сторонніх DIO пристроїв).
- час самовідновлення мережі в разі обриву кабелю або несправності модуля 50 мс.

Обмеження за кількістю переданих даних для RIO пристроїв становить до 400 вхідних слів і 400 вихідних слів, швидкість передачі даних складає 100 Мб/с, а максимальні відстані між вузлами мережі RIO:

- 100 м по мідному кабелю (між вузлами мережі);
- 2 км по многомодовому оптичному кабелю;
- 15 км по одномодовому оптичному кабелю.

Детермінізм мережі вимірюється максимальним часом, витраченим на доставку за призначенням критично важливих пакетів даних. Детермінізм забезпечує передачу даних по мережі Ethernet RIO за раніше встановлений фіксований період часу. У мережі Ethernet RIO прийом/передача максимального розміру пакета даних з пристроїв віддаленого вводу-виводу, займає не більше 9 мс.

### 2.3 Монтажні шасі

Локальне монтажне шасі - шасі, в якому встановлений модуль центрального процесорного пристрою (ЦПУ). В архітектурі мережі eRIO може бути встановлено тільки одне локальне шасі (за винятком випадків систем з «гарячим» резервуванням).

Шасі станції віддаленого вводу-виводу (віддалене шасі) - це шасі, в яке встановлено модуль-адаптер станції віддаленого вводу-виводу.

Шасі розширення - шасі, яке підключається до спеціального розніму локального або віддаленого шасі і служить для збільшення максимально можливої кількості модулів вводу-виводу.

Основна роль монтажного шасі - забезпечити шину обміну даними між модулями, встановленими в даний шасі. Шина обміну даними вбудована всередину монтажного шасі і називається внутрішньою шиною.

В архітектурі системи автоматизації, побудованої на базі контролера Modicon M580, можуть бути використані два типи внутрішнього шини – E-bus і X-bus. Монтажні шасі контролера Modicon M580 мають в своєму складі обидва типи внутрішніх шин. Монтажні шасі контролера Modicon M340 в якості внутрішньої шини мають тільки шину X-bus, але, тим не менш, можуть бути використані в архітектурі системи автоматизації, побудованої на базі контролера Modicon M580 з деякими обмеженнями.

Внутрішня шина E-bus обов'язково є у випадках:

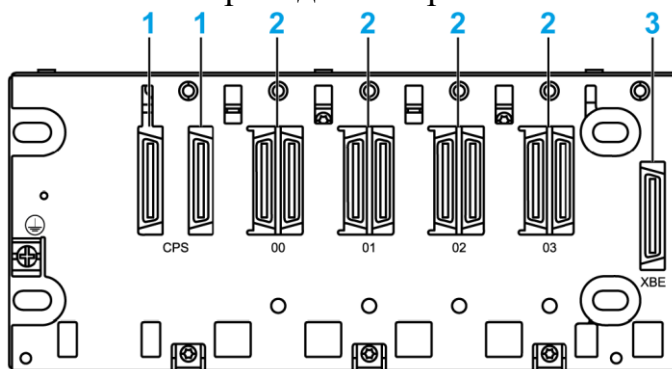
- використання модулів сімейства eX80 (наприклад, eX80 HART модулі);
- використання модулів ваговимірювання;
- використання комунікаційних Ethernet-модулів, пов'язаних з ЦПУ.

В інших випадках дозволяється застосовувати шасі контролера Modicon M340, при цьому необхідно враховувати, що номер версії випуску даного типу стійок повинен бути не нижче PV: 02. Тільки в цьому випадку гарантується нормальна робота даного типу шасі в системі автоматизації на основі контролера Modicon M580. Нижче в таблиці представлені типи монтажних шасі контролера Modicon M580. Всі ці типи шасі доступні як в стандартному, так і в спеціальному виконанні (символ «Н» в кінці замовного номера).

ВМЕ ХВР .... (Н)	Кол-во модульних слотів
0400 / 0400Н	4
0800 / 0800Н	8
1200 / 1200Н	12

У 12-ти слотовому монтажному шасі, тільки 8 слотів мають розніми обох типів внутрішньої шини. Слоти ж з номерами 02, 08, 10 і 11 мають розніми тільки внутрішньої шини X-bus.

4-х слотове монтажне шасі приведено на рис 2.1.

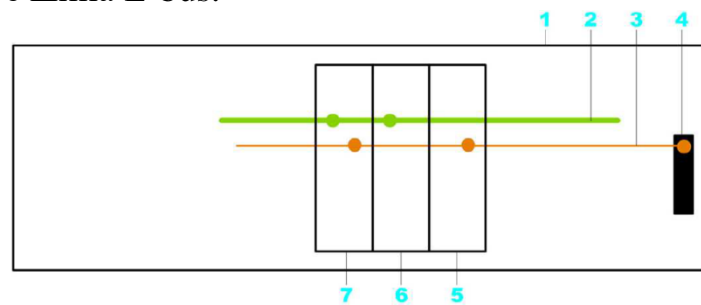


1 - розніми для підключення модуля джерела живлення; 2 - розніми задніх шин стійки (лівий - E-bus; правий - X-bus); 3 - рознімим для підключення модуля розширення.

Рисунок 2.1.– 4-х слотове монтажне шасі ПЛК M580

Шасі розширення використовуються в системах автоматизації для збільшення кількості використовуваних модулів, а також розширення області, що покривається шасі, для забезпечення можливості розташування модулів вводу-виводу максимально близько до контрольованого обладнання.

При використанні шасі розширення слід враховувати, що навіть в разі використання в якості основного шасі контролера Modicon M580 шасі, що має наявності обидва типи внутрішньої шини, фактично розширюється тільки шина X-bus. Ця обставина забороняє використання в шасі розширення модулів, в яких внутрішньою є шина E-bus.



1 - основне шасі; 2 - внутрішня шина E-bus; 3 - внутрішня шина X-bus; 4 - рознім для підключення модуля розширення; 5 - модуль, що використовує лише шину X-bus; 6 - модуль, що використовує лише шину E-bus; 7 - модуль, що використовує оба види внутрішньої шини.

Рисунок 2.2– Продовження внутрішньої шини в шасі розширення

В якості локального шасі розширення можуть бути використані шасі сімейства контролера Modicon M340 або Modicon Premium, тоді як для розширення шасі станції віддаленого вводу-виводу – тільки шасі сімейства контролера Modicon M340.

Залежно від ЦПУ, що використовується в системі автоматизації, локальне шасі може мати до 3-х (ВМЕ Р58 1020, ВМЕ Р58 2020, ВМЕ Р58 2040) або до 7 (ВМЕ Р58 3020, ВМЕ Р58 3040, ВМЕ Р58 4020, ВМЕ Р58 4040 ) шасі розширення сімейства контролера Modicon M340.

Припустимо одночасне використання в якості шасі розширення шасі сімейства контролера Modicon M340 і шасі сімейства контролера Modicon Premium. При цьому до локального шасі спочатку підключаються шасі розширення сімейства контролера Modicon M340, а потім - шасі розширення сімейства контролера Modicon Premium.

При використанні в якості шасі розширення шасі сімейства контролера Modicon M340 загальна довжина кабелів розширення не повинна перевищувати 30 м. Тоді як у разі використання в якості шасі розширення шасі сімейства контролера Modicon Premium - 100 м.

## 2.4 Комунікаційні модулі

Комунікаційні модулі використовуються в системах автоматизації для забезпечення зв'язку ЦПУ з мережевими пристроями. Найчастіше система АСУТП містить в своєму складі розподілені по території об'єкта автоматизації станції вводу-виводу сигналів, які мають мережеві інтерфейси зв'язку. З іншого боку часто постає завдання організації каналу обміну даними з ЦПУ суміжної ділянки або комп'ютерами системи візуалізації.

Модуль BMX NOC 0402 - комунікаційний модуль для мереж Ethernet, що надає можливість ПЛК Modicon M580 обмінюватися даними з іншими Ethernet-пристроями, використовуючи протоколи EtherNet/IP або Modbus TCP. Функціонально цей комунікаційний модуль ідентичний BMX NOC 0401, який широко використовується в системах автоматизації, виконаних на базі ПЛК Modicon M340.

Основне завдання комунікаційного модуля BMX NOC 0402 - забезпечити інтеграцію розподілених пристроїв за допомогою виділення їх в окрему мережу, не пов'язану з мережею віддаленого вводу-виводу (Ethernet Remote I/O network). Це виділення дозволяє оптимально використовувати мережеві ресурси системи автоматизації. Однак цей модуль не використовує внутрішню шину монтажного шасі E-bus. Тому фірмою Schneider Electric планується виробництво комунікаційного модуля BME NOC, який в повній мірі буде використовувати потенціал нового ПЛК Modicon M580.

## **2.5 Модулі ваговимірювання**

У системах автоматизації часто виникають завдання вимірювання ваги. Підтримка ПЛК Modicon M580 модуля ваговимірювання PME SWT 0100, виробником якого є партнер Schneider Electric, фірма SCAIME, закриває потребу в засобах ваговимірювання.

Модуль PME SWT 0100 є повнофункціональним контролером ваговимірювання, який може бути використаний у всіх системах автоматизації Schneider Electric, де необхідно ваговимірювання для забезпечення вимог технологічного процесу.

Основними перевагами цього модуля є:

- використання в локальних/віддалених шасі ПЛК Modicon M580;
- повноцінне використання внутрішньої шини E-bus;
- конфігурація з програмного пакета Unity Pro v8.0;
- висока точність ваговимірювання (0,01%);
- наявність дискретних входів/виходів;
- можливість роботи з 4-х або з 6-ю провідним підключенням ваговимірювальних засобів;
- наявність інтерфейсу RS485 для підключення локального терміналу ЧМІ;
- можливість використання у вибухонебезпечних зонах (ATEX).

## **2.6 Модулі з підтримкою HART-протоколу**

HART (Highway Addressable Remote Transducer) - один з декількох протоколів зв'язку, які активно використовуються в промисловій автоматизації. HART-протокол - це протокол типу «Master-Slave». Цей протокол визнаний кращим при отриманні додаткових даних від польових пристроїв, а також діагностичної інформації. Причому здійснюється це за допомогою стандартного струмового сигналу 4..20мА. Звичайний датчик аналогового

сигналу здатний передавати тільки значення змінної процесу, при цьому немає впевненості, що це значення є вірним. У разі використання HART- протоколу передається змінна процесу, а також додаткова інформація:

- стан пристрою + діагностичні події;
- змінні процесу + одиниці виміру;
- основні конфігураційні параметри;
- ідентифікатори виробника / пристрої.

HART-інтерфейсними модулями в складі ПЛК Modicon M580 є:

а) BME АНІ 0802:

8 каналів аналогового вводу; базовий сигнал - 4..20мА; міжканальна ізоляція = 1000 В; підтримка HART-протоколу версій v5, v6, v7; робота з ПЗ: КПО / UnityPro / SCADA.

б) BME АНО 0412:

4 канали аналогового виводу; базовий сигнал - 4..20мА; міжканальна ізоляція = 1000 В; підтримка HART-протоколу версій v5, v6, v7; робота з ПЗ: КПО / UnityPro / SCADA.

## **2.7 Використання станцій розподіленого вводу-виводу ПЛК Modicon M340**

Програмований логічний контролер Modicon M580 дозволяє з мінімальними витратами перевести АСУТП, що побудовані на базі ПЛК Modicon M340, на нову платформу. При наявності в системі АСУТП станцій розподіленого вводу-виводу на базі ПЛК Modicon M340 можна підключити їх до модернізованої системі в незмінному вигляді (DIO-станції) або конвертувати в станції віддаленого вводу-виводу (RIO-станції).

Конвертація станції розподіленого вводу-виводу в станцію віддаленого вводу-виводу забезпечує ряд переваг:

- пріоритетний обмін даними з ЦПУ;
- час відновлення мережі в межах 50 мс;
- використання спеціалізованих модулів (NOM, NOR, ERT і т.д.);
- наявність буфера подій з мітками часу (10 мс.)

Конвертація станції розподіленого вводу-виводу в станцію віддаленого вводу-виводу досягається шляхом заміни модуля ЦПУ (в разі використання в даній станції модуля VMXP \*\*\*\*\*) або модуля адаптера (в разі використання в даній станції модуля VMXPRA0100) на модуль адаптера ( VMXCRA31200/ VMXCRA31210/VMCRA31210) станції віддаленого вводу-виводу. Слід зазначити, що, в разі використання шасі розширення станції розподіленого вводу-виводу, повноцінна конвертація можлива тільки, якщо використовується не більше одного шасі розширення.[2]

### **Контрольні запитання.**

1. Які конструктивні особливості має ПЛК Modicon M580?
2. Скількома типами ЦПУ представлений ПЛК Modicon M580 і скільки з них підтримують роботу пристроїв розподіленого вводу-виводу (DIO)?

3. Яку максимальну кількість каналів дискретного і аналогового вводу-виводу підтримує ЦПУ ModiconM580?
4. Яку кількість локальних модулів розширення та станцій віддаленого вводу-виводу підтримує ЦПУ ModiconM580?
5. Яка максимальна швидкість обробки логічних інструкцій ЦПУ ModiconM580?
6. В чому основні переваги технології Ethernet IO?
7. Які типи внутрішньої шини використовуються в монтажних шасі ModiconM580?
8. Скільки модульних слотів мають шасі ModiconM580?
9. Яка з внутрішніх шин основного шасі контролера M580 розширюється при використанні шасі розширення?
10. Чи можуть використовуватися шасі контролера M340 для розширення локального шасі або шасі станції віддаленого вводу-виводу ПЛК ModiconM580?
11. Скільки шасі розширення може мати ПЛК ModiconM580?
12. Яке призначення має комунікаційний модуль BMX NОC 0402 і яке його основне завдання?
13. Які переваги має модуль ваговимірювання PMESWT 0100?
14. Які можливості HART-протоколу?
15. Чи можливо перевести АСУТП, що побудована на базі ПЛК Modicon M340, нанову платформу з контролером M580?

## **2. Програмування контролерів у середовищі UNITYPRO.**

### **2.1 Загальні положення.**

Для програмування контролерів Modicon Quantum, Premium, M340 та M580 використовується єдине програмне забезпечення **UNITYPRO**. Цей інструментарій дає користувачу такі можливості:

1. Створення апаратної конфігурації та програми користувача для контролерів Modicon, а саме:
  - використання мультизадачного режиму: одна MAST, одна FAST, декілька EVT, одна AUX (тільки для QUANTUM та PREMIUM);
  - використання 5-ти мов програмування згідно стандарту MEK 61131-3: LD, ST, IL, FBD, SFC;
  - поділу програми користувача на секції (Section), кожна з яких може бути написана на різних мовах програмування MEK;
  - використання підпрограм (SR);
  - функціональне структурування проекту користувача;
  - доступу до великої бібліотеки функцій та функціональних блоків (FFB), які доступні на будь якій мові програмування;
  - створення функціональних блоків користувача (DFB);
  - використання поряд з локалізованими (located, прив'язаними до конкретної комірки пам'яті) та нелокалізованими (unlocated, не прив'язаними до конкретної комірки) даних;
  - використання масивів, структурних типів користувача;



2. Відладгодження програми користувача, а саме:

- використання програмного емулятору (simulator) контролера з підтримкою

більшості функцій UNITY та можливості доступу з інших програмно-технічних засобів до нього по Modbus/TCP;

- анімації змінних безпосередньо в редакторах за допомогою кольору, відображення числових та текстових значень;

- управління та контролю змінних за допомогою таблиць анімацій (Animation Table);

- перегляду стану кроків мови SFC;

- використання точок переривання (Break Point), покрокового виконання програми, точки спостереження (Watch point);

- зміни програми користувача в режимі виконання контролером програми управління.

3. Експлуатації та обслуговування, а саме:

- створення та використання графічних сторінок (Operator Screens) за допомогою технологічного процесу (подібно засобам HMI);

- використання вбудованих діагностичних засобів для контролю стану будь-якої частини контролера;

- використання вбудованого вікна тривоги Alarm Viewer для перегляду стану діагностичного буферу контролера;

4. Автоматичного створення документації по проекту.

5. Імпорту та експорту частин проекту в форматі \*.XML для можливості їх використання в інших програмних засобах.

## 2.2 Структура програми користувача.

**Задачі.** Програма користувача для контролерів Modicon може виконуватись в контексті однієї або декількох *задач* (Tasks) та може включати обов'язкову основну задачу MAST та додаткові FAST та EVENTS. При конфігуруванні апаратного забезпечення до кожної задачі, "прив'язуються" входи, які будуть обпитуватися на початку виконання задачі, та виходи – в кінці виконання задачі. Кожна задача запускається на виконання тільки при умові знаходження ПЛК в режимі виконання (**RUN**). У режимі зупинки (**STOP**) відбувається тільки циклічне опитування входів.

**Задача MAST.** У ПЛК завжди функціонує як мінімум одна задача **MAST**, яка може виконуватись у циклічному чи періодичному режимах (рис.2.1). Обидва режими передбачають циклічне виконання програми задачі, яка починається зі зчитування входів та закінчується записом виходів. У **циклічному режимі**, початок наступного виклику задачі MAST починається відразу по закінченню обробки попередньої. Таким чином, час між викликами задачі MAST залежить від тривалості її виконання. У **періодичному режимі**, інтервал між викликами задачі задається в проекті UNITY PRO. Цей інтервал вибирається зазвичай більшим за максимальний час обробки задачі. Тобто, в більшості випадків, між викликами задачі MAST контролер буде виконувати внутрішню обробку (діагностичні операції, комунікаційний обмін, тощо).

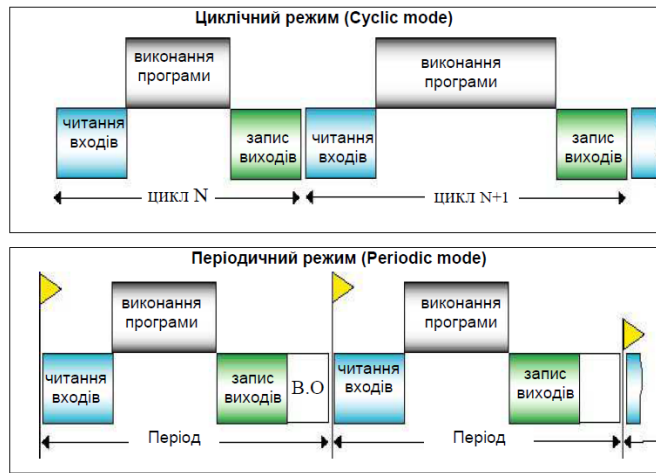


Рисунок 2.1–Циклічний та періодичний режим роботи задачі MAST

Програма задачі MAST представляє собою набір *секцій* (рис.2.2), кожна з яких може бути написана на будь-якій з мов MEK 61131-3: IL, LD, ST, FBD чи SFC.

Секції виконуються одна за одною, в порядку розміщення їх в гілці Sections (рис.2.2). У свою чергу, програми в секціях можуть викликати підпрограми, які записуються в SR Sections цієї ж задачі.

MAST задача контролюється *сторожовим таймером* (WATCH DOG), який конфігурується в UNITY PRO. У випадку перевищення тривалості виконання задачі MAST за встановлене максимально допустиме значення, контролер перейде в режим STOP, що не допустить його "зависання".

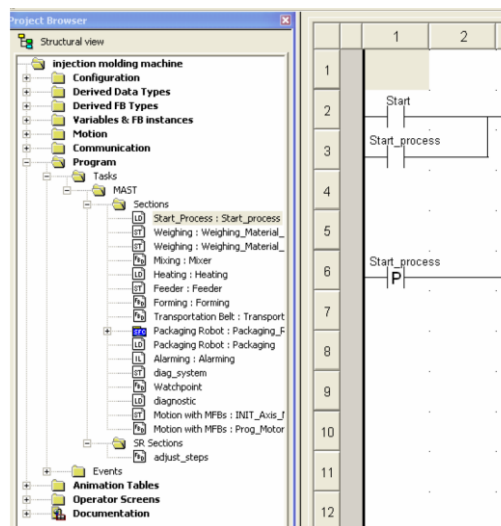
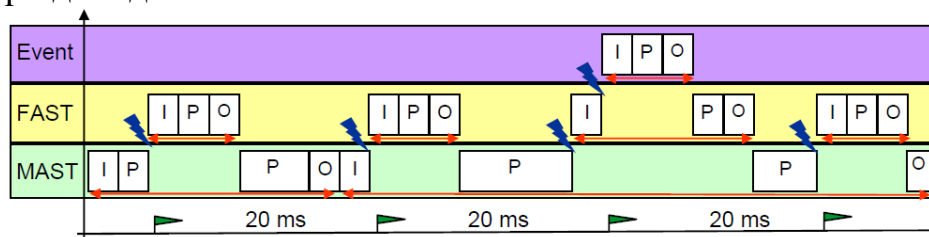


Рисунок 2.2–Приклад вигляду структури задачі MAST.

**Задача FAST.** У багатозадачній структурі програми M340, крім MAST можуть функціонувати також одна задача FAST та декілька задач Event. Задача *FAST* запускається завжди в періодичному режимі і має вищий пріоритет за задачу MAST. Вона призначена для процесів які потребують частішої обробки ніж ті, які обробляються в задачі MAST. Аналогічно до структури програми MAST, програма FAST також записується у секціях, однак використання мови SFC для задачі FAST не допускається.

**Задачі EVENT.** Задачі *Event* викликаються коли відбувається певна подія: *EVT<sub>i</sub>* – по апаратним подіям, *TIMER<sub>i</sub>* – по таймерним. Задачі *EVT<sub>i</sub>* (де *i* – номер задачі даного типу) прив'язуються до певної події, які фіксуються модулями спеціального призначення. Задачі *TIMER<sub>i</sub>* (де *i* – номер задачі даного типу) прив'язуються до таймера спеціального призначення, який запускається в програмі користувача спеціальною функцією ITCNTRL.

**Пріоритетність задач.** Задачі *TIMER<sub>i</sub>* мають вищий пріоритет ніж *FAST*, а *EVT<sub>i</sub>* – вище ніж *TIMER<sub>i</sub>*. Задачі з вищим пріоритетом будуть переривати виконання менш пріоритетних задач, які будуть продовжувати виконання задачі після закінчення виконання більш пріоритетної задачі. На рис.2.3 показаний приклад роботи контролера в багатозадачному режимі, з тривалістю періода задачі *FAST* - 20 мс.



I – опитування входів "прив'язаних" до задачі; O – запис виходів; P – виконання програми задачі

Рисунок 2.3– Приклад роботи контролера у багатозадачному режимі

## 2.3 Структура пам'яті ПЛК

**Розподіл пам'яті прикладної програми.** Пам'ять, яку займає прикладна програма UNITY (Application), складається з таких розділів:

- локалізовані та нелокалізовані дані – дані користувача;
- системні дані;
- програма користувача, включно символи та коментарі;
- константи.

При увімкненому контролері ці дані розміщуються в оперативній пам'яті ROM процесорного модуля (рис.2.4). Додатково у ній також виділена області пам'яті, для можливості внесення зміни у програму користувача в режимі он-лайн, тобто не зупиняючи роботу контролера. Для збереження програми користувача та констант при вимкненому живленні контролера використовується SD карта. При завантаженні або зміні прикладної програми в контролері, вона автоматично зберігається на карті пам'яті SD, а при увімкненні контролера – зчитується з неї. Крім прикладної програми на карті

зберігаються дані WEB-сервера для доступу до контролера через порт Ethernet, а також файли користувача (тільки для карт типу MPF). Для збереження значень локалізованих та нелокалізованих змінних при вимкненому живленні, використовується внутрішня флеш пам'ять процесорного модуля.

**Локалізовані дані користувача.** Програма користувача може оперувати локалізованими та нелокалізованими даними. Розміщення *локалізованих даних (located data)* у пам'яті наперед визначене, що дає можливість звернутися до них за адресою. При створенні змінних, можна вказати комірку розміщення даних для неї в конкретній області, що дає можливість оперувати

з локалізованими даними не за адресою а за символічним ім'ям змінної. Процес прив'язки змінних до конкретної комірки пам'яті будемо називати *локалізацією*. Змінні, які прив'язані до локалізованої області пам'яті будемо називати *локалізованими змінними*.

У залежності від призначення, локалізовані дані розміщені в декількох областях:

- **%M** – область даних для внутрішніх булевих (Boolean) змінних;
- **%MW** – область даних для внутрішніх числових змінних;
- **%S** – область даних для системних булевих змінних;
- **%SW** – область даних для системних числових змінних;
- **%I, %IW, %Q, %QW** – область даних асоційованих з каналами модулів ПЛК;
- **%KW** – область констант.

Області внутрішніх змінних призначені для довільного використання, наприклад, для збереження проміжних результатів розрахунку, мережного обміну, тощо. Кожна комірка області адресується унікальним номером, починаючи від 0 та закінчуючи номером останньої сконфігурованої змінної даної області. Область **%M** адресується побітно, а **%MW** - 16-розрядними словами. Так, наприклад, адреса наступної комірки після **%MW16** є комірка з адресою **%MW17**.

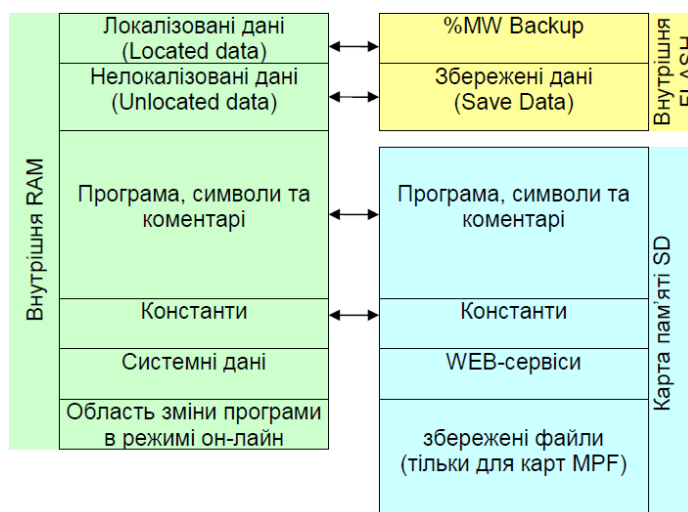


Рисунок 2.4 – Структура пам'яті M340

В областях системних змінних **%S** та **%SW** знаходиться інформація про функціонування контролера. Кожна комірка має своє призначення і адресується номером. Так, наприклад, системний біт **%S0** переводиться у стан логічної "1" при холодному старті (при увімкненні живлення), а на наступний цикл задачі **MAST** переводиться в "0". А у змінній **%SW53** знаходиться інформація про поточний рік.

Область **%S** адресується побітно, а **%SW** - 16-розрядними словами.

В області даних, що асоційована з каналами модулів ПЛК, знаходиться інформація про стан кожного каналу: числове значення, інформація про помилку, конфігураційна інформація, тощо. Адреса та розмір комірок цієї

області пов'язані з адресами та типом каналів, за які вони відповідають. Більш детально адресацію каналів розглянемо в наступному підрозділі.

В області констант розміщені ті дані, які не можуть змінюватися в режимі

виконання контролером програми користувача. Область %KW адресується 16-розрядними словами. Кожна комірка має унікальний номер, починаючи з 0.

**Нелокалізовані дані користувача.** Розміщення *нелокалізованих даних (unlocated data)* невідоме користувачу і вибирається середовищем UNITYPRO при кожній побудові (**BUILD**) проекту. Це значить, що звернення до нелокалізованих даних за адресою неможливе.

Якщо при описі змінної в UNITY PRO не вказується конкретне її розміщення в пам'яті, вона буде знаходитись в області нелокалізованих даних. Такі змінні будемо називати *нелокалізованими змінними*. Використання нелокалізованих змінних звільняє користувача від необхідності слідкування за виділенням області даних. Одним з недоліків використання нелокалізованих змінних є ускладнення при забезпеченні обміну ними по промисловим мережам.

## 2.4 Робота з даними в UNITY PRO.

**Типи даних.** Прикладна програма користувача може оперувати даними наступним чином:

- звертаючись до них за адресою (тільки для локалізованих даних);
- через змінні, звертаючись до них по імені змінної;
- через екземпляр функціонального блоку;

У будь якому випадку у прикладній програмі UNITY операції з даними проводяться чітко згідно їх типу. **Тип даних** визначає структуру, формат, набір атрибутів та визначених операцій над цими даними. Всі типи даних UNITY можна поділити на чотири різні категорії:

EDT (Elementary Data Type) – елементарні типи даних ;

DDT (Derived Data Type) – похідні типи даних;

EFB (Elementary Function Block) – елементарні функціональні блоки;

DFB (Derived Function Block) – похідні функціональні блоки.

EFB та DFB функціональні блоки розглянуті в розділі 3.5.

Розглянемо елементарні та похідні типи даних.

В UNITY представлені наступні *елементарні типи даних (EDT)*: BOOL, EBOOL, INT, DINT, UINT, UDINT, BYTE, WORD, DWORD, REAL, TIME, DATE, TIME\_OF\_DAY, DATE\_AND\_TIME, STRING, STRING[n]. Їх характеристики наведені у табл. 2.1.

Таблиця.2.1 Характеристика деяких елементарних типів даних (EDT).

Тип	Призначення	кількість біт, формат	Діапазон значень	форми відображення константи
<b>BOOL</b>	булевий	8	TRUE/FALSE	<b>TRUE</b> або <b>1</b> , <b>FALSE</b> або <b>0</b>
<b>EBOOL</b>	булевий з можливістю	8	TRUE/FALSE	<b>TRUE</b> або <b>1</b> , <b>FALSE</b> або <b>0</b>

	форсування та визначення фронтів			
<b>INT</b>	ціле, числові операції	16, 16-й біт - знак	-32768...32767	десятькова, двійкова (2#), вісімкова(8#), шіснадцяткова(16#), наприклад: <b>3</b> – десяткова; <b>2#0000000000000011</b> - :двійкова; <b>8#000003</b> - вісімкова; <b>16#0003</b> - шіснадцяткова.
<b>DINT</b>	подвійне ціле, числові операції	32, 31-й біт - знак	0...65535	десятькова, двійкова (2#), вісімкова(8#), шіснадцяткова(16#)
<b>WORD</b>	слово, побітові операції	16	16#0...16#FFFF	двійкова (2#), вісімкова(8#), шіснадцяткова(16#)
<b>REAL</b>	реальне, число з плаваючою комою	32, 31-й біт – знак, 8 біт експоненційна, 23 біт - мантіса	Нормальні значення: 3.4028235e+38... 1.1754944e-38, -0...0, 1.1754944e-38... 3.4028235e+38	0.456 аналогічно -1.32e12 1.0E+6 аналогічно 1000000  Не гарантований результат <b>DEN</b> : -1.1754944e-38 ... 1.1754944e-38  Нескінченність: < 3.4028234e+38 <b>-INF</b> >+3.4028234e+38 <b>+INF</b>  <b>QNaN</b> та <b>SNAN</b> – невірний формат числа
<b>TIME</b>	беззнакове подвійне ціле	32, зберігає значення в мілісекундах	0...4294967295 мс, або T#0MS...T#49D_17H_2M_47S_295MS	Часова константа <b>T#</b> , де D – дні, H – години, M – хвилини, S – секунди, MS – мілісекунди, Наприклад: T#16S_500MS – 16,5 с

**Похідні типи даних (DDT)** – це складені типи даних, на базі елементарних типів. Це можуть бути масиви та структури, як вже визначені в UNITY та доступні через бібліотеку, так і визначені користувачем в процесі створення проекту. Змінні на основі похідних типів даних також будемо називати **структурними**.

**Структурні типи даних користувача.** Структурні типи даних користувача (належить до DDT) створюється в розділі проекту (Derived Data Types). Структурному типу дається ім'я, а так створюються поля. При створенні типу він перебуває в режимі конструкції, що відображається на піктограмі імені типу (рис.2.5 зверху).

Після створення всіх потрібних файлів необхідно викликати команду Build→ *Analyze* для перевірки правильності структури та внесення змін до проекту. При відсутності помилок піктограма «конструктор» зникає (рис.2.5 знизу).

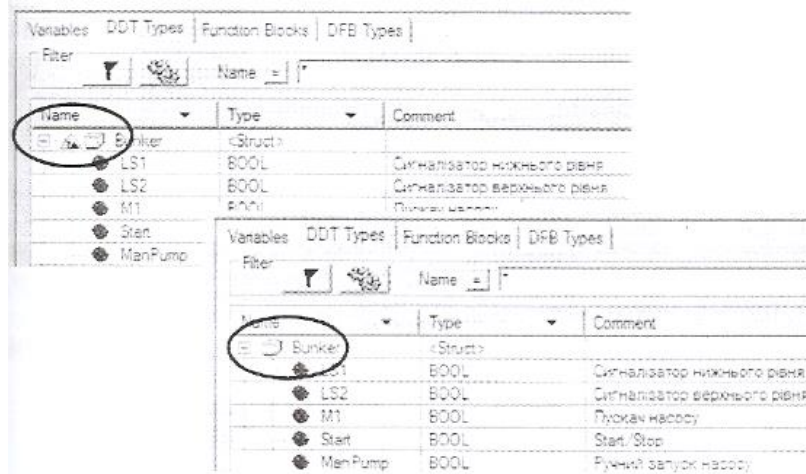


Рис. 2.5 Створення структурного типу користувача DDT угорі–до команди *Analyze*, внизу – після

Структурні змінні створюється аналогічна до змінних елементарних типів даних (рис.2.6). Дозволено також створювати масиви на основі структурного типу DDT.

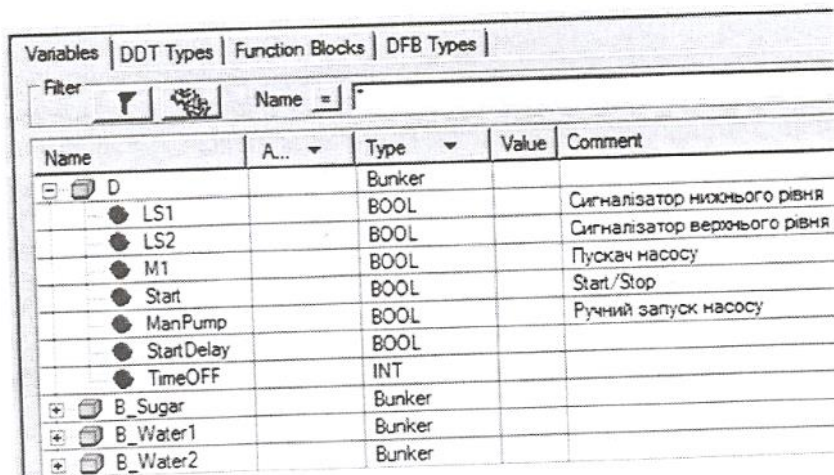


Рис.2.6 Структурні змінні.

**Робота зі змінними в UNITY.** UNITY дає можливість оперувати з даними через змінні. Змінні створюються у середовищі UNITY PRO у розділі "Variables & FB instances". При створенні змінної визначається її ім'я та тип. Ім'я повинно бути унікальним, містити символи латинського алфавіту, не містити службові символи та пробіли. Тип вибирається з доступних в UNITY елементарних та похідних типів.

Серед властивостей змінної доступна адреса розміщення даних (Address). Якщо при визначенні змінної, у властивості Address вказати комірку пам'яті (дивись локалізовані змінні користувача), то змінна буде локалізованою. В іншому випадку, змінна буде нелокалізована, отже адреса розміщення її буде невідомою, тобто до даних можна буде звертатися тільки через ім'я змінної. Для кожної змінної можна вказати початкове значення, тобто значення при ініціалізації.

Приклади вибору змінних та визначення їх властивостей наведені на рис.2.7.

Name	Type	Ad...	Value	Comment
Array1	ARRAY[0..1] OF INT			нелокалізована змінна-масив з 2-х елементів типу INT
Array1[0]	INT		16#23F2	0-й елемент масиву зі значенням ініціалізації в 16-ковому форматі
Array1[1]	INT		2568	1-й елемент масиву зі значенням ініціалізації в 10-ковому форматі
Array2	ARRAY[5..6] OF INT	%Mw100		змінна-масив з 2-х елементів типу INT, прив'язаний до комірок %Mw100 та %Mw101
Array2[5]	INT	%Mw100	2#000111100001...	елемент масиву з номером 5 зі значенням ініціалізації в 2-ковому форматі
Array2[6]	INT	%Mw101	100	елемент масиву з номером 6 зі значенням ініціалізації в 10-ковому форматі
Bool1	BOOL		TRUE	нелокалізована змінна типу BOOL
Ebool2	EBOOL		FALSE	нелокалізована змінна типу EBOOL
Bool3	EBOOL	%M200	TRUE	змінна типу EBOOL, прив'язана до %M200
Bool4	BOOL	%Mw200.7		змінна типу BOOL, прив'язана до 7-го біту слова %Mw200
Real1	REAL		16.5	нелокалізована змінна типу REAL
Real2	REAL	%Mw150	1.25e+4	змінна типу REAL, прив'язана до комірок %Mw150 та %Mw151
Int1	INT		2345	нелокалізована змінна типу INT
Int2	INT	%Mw160	16#ABCD	змінна типу INT, прив'язана до %Mw160
Int3	INT	%IW0.1.1		змінна типу INT, прив'язана до %IW0.1.1
Time1	TIME		T#25s350ms	нелокалізована змінна типу TIME зі значенням ініціалізації 25 секунд 350 мілісекунд
Time2	TIME	%Mw170	T#2h16m34s	змінна типу TIME зі значенням ініціалізації 2 год 16 хв 34 сек, прив'язана до комірок %Mw170 та %Mw171

Рис. 2.7 Вибір змінних та визначення їх властивостей

Для нелокалізованих змінних в полі Address (3 колонка) нічого не пишеться.

Масиви вибираються з визначенням типу елементів, а також початкового та кінцевого індексу. При локалізації масиву, вказується тільки адреса початкової комірки, всі інші комірки прив'язуються автоматично починаючи з вказаної.

Кількість зайнятих комірок залежить від типу та кількості елементів масиву. Типи даних DWORD, DINT, REAL, TIME займають два слова, тому при локалізації розміщуються в двох суміжних комірках.

При створенні структурних даних спочатку визначають структурний тип(DDT), а потім створюють на основі цього типу змінну. Для звернення до елемента структурної змінної (поля), вказують спочатку назву змінної, а потім через крапку - поле змінної.

Звернення до елементів масиву проводиться через квадратні дужки. Так наприклад для звернення до 0-го елемента масиву *Array1* (рис.2.7) необхідно написати: *Array1[0]*.

Можливе побітове звернення до змінних. Для цього після назви змінної через крапку вказується номер біта. Наприклад для звернення до 7-го біта змінної *Int1*(рис.2.7) необхідно записати: *Int1.7*. Аналогічно можна звернутися також до біта у локалізованій області пам'яті. Наприклад запис *%MW100.4*, означає що йде звернення до 4-го біта комірки *%MW100*.

**Адресація каналів вводу/виводу.** Задачі MAST та FAST починаються зі зчитування даних з вхідних каналів ПЛК а закінчуються записом даних у вихідні канали. Процес зчитування та запису проходить автоматично, неявно для користувача. Прив'язка каналів до задач проводиться при конфігурації апаратної частини ПЛК.

Інформація про стан та значення вхідних каналів ПЛК Modicon знаходиться у комірках *%I* та *%IW*, а значення вихідних – у комірках *%Q* та *%QW*. Кожна комірка з даних областей пам'яті відповідає за конкретний канал, в залежності від розміщення модуля у шасі та номеру каналу. Тобто топологічна адреса розміщення даних, які відповідають за значення каналу, визначається: номером шасі, на якому розташований модуль; розміщенням модуля у шасі та каналом на модулі. Таким чином:

*% Ir.m.c* – адреса відповідає за дискретний вхід на шасі з номером *r*, модулі на посадочному місці *m*, каналу *c*.



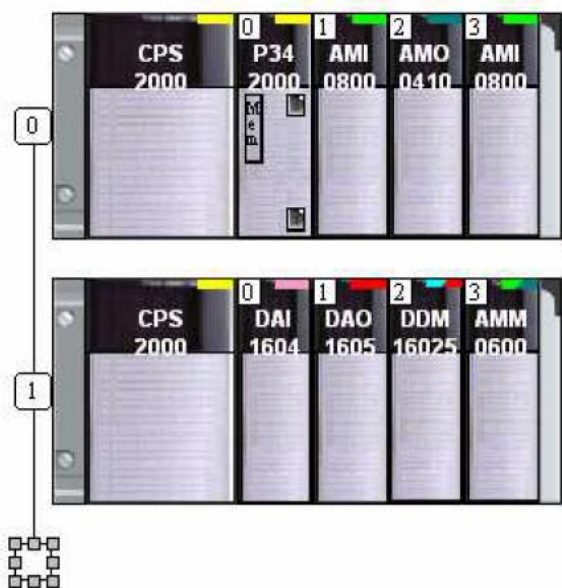
$\%IW_r.m.c$  - адреса відповідає за аналоговий вхід на шасі з номером  $r$ , модуліна посадочному місці  $m$ , каналу  $c$ .

$\%Qr.m.c$  – адреса відповідає за дискретний вихід на шасі з номером  $r$ , модуліна посадочному місці  $m$ , каналу  $c$ .

$\%QWr.m.c$  – адреса відповідає за аналоговий вихід на шасі з номером  $r$ , модуліна посадочному місці  $m$ , каналу  $c$ .

На рис.3.5 показані приклади топологічних адрес які відповідають за значення каналів. Для модулів з однотипними каналами (тільки входи або тільки виходи)номер каналу співпадає з порядковим номером входу або виходу (якщо рахувати з 0).

Для дискретних змішаних модулів, перші 16 каналів (0-15) виділяються під входи, а виходи починають адресуватися з 16-го. Навіть якщо у змішаному модулі тільки 8 входів (як на рис.2.8), адресація дискретних виходів все одно починається з16-го. Для аналогових змішаних модулів АММ 0600, перші 4-ри канали (0-3)виділяються під входи, а виходи починають нумеруватися з 4-го.



номер шасі	Місце модуля	Тип модуля	Номер каналу	змінна
0	1	AMI 0800	0	$\%IW0.1.0$
			...	...
			7	$\%IW0.1.7$
	2	AMO 0800	0	$\%QW0.2.0$
			...	...
			7	$\%QW0.2.7$
	3	AMI 0800	0	$\%IW0.3.0$
			...	...
			7	$\%IW0.3.7$
1	0	DAI 1604	0	$\%I1.0.0$
			...	...
			15	$\%I1.0.15$
	1	DAO 1605	0	$\%Q1.1.0$
			...	...
			15	$\%Q1.1.15$
	2	DDM 16025 (входи)	0	$\%I1.2.0$
			...	...
			7	$\%I1.2.7$
		DDM 16025(виходи)	15	$\%Q1.2.15$
			...	...
			31	$\%Q1.2.31$
3	AMM 0600 (входи)	0	$\%IW1.3.0$	
		...	...	
	AMM 0600 (виходи)	3	$\%IW1.3.3$	
		4	$\%QW1.3.4$	
			7	$\%QW1.3.5$

Рисунок 2.8– Приклади топологічних адрес, які відповідають за значення каналів вводу/виводу вUNITY PRO.

### 3. Бібліотека елементарних функцій EF/блоків EFB, та похідних блоків DFB. 3.1 Основні поняття.

В програмі користувача можна використовувати різні типи програмних блоків, в які можна передавати фактичні параметри та отримувати на виході результати розрахунку. Можна користуватися як бібліотечними готовими блоками (функції, процедури, елементарні функціональні блоки) так і розробити власні (похідні функціональні блоки).

**Функція** (Elementary Functions - EF) - бібліотечний програмний блок, який може мати декілька входів і тільки один вихід. Функція не має внутрішньої пам'яті, тобто вона не може між викликами "всередині себе" зберігати розраховані значення.

Це значить, що вихід функції завжди однозначно залежить тільки від станів її входів.

**Процедура** (Procedure) – бібліотечний програмний блок, який може мати декілька входів і декілька виходів. Крім наявності декількох виходів, відмінність процедур від функцій є в можливості використання параметрів типу VAR\_IN\_OUT(вхід/вихід).

**Елементарний функціональний блок** (Elementary Function Block, *EFB*) – готовий бібліотечний програмний блок, який має внутрішню пам'ять, тобто може зберігати проміжні розрахункові значення між викликами. Перед використанням функціонального блоку у програмі користувача, спочатку створюють **екземпляр функціонального блоку** (FB instance), якому дають унікальне ім'я. Перегляд, створення та видалення функціональних блоків в UNITY PRO виконується у розділі "Variables & FB Instances" у підрозділі "Elementary FB Instances". Екземпляр функціонального блоку буде вмішувати всі внутрішні дані, тобто його пам'ять. Дані для екземплярів функціональних блоків розміщуються в нелокалізованій області пам'яті, тому за адресою до них звернутися неможливо.

**Похідний функціональний блок** (Derived Function Block, *DFB*) – функціональний блок, який розроблений користувачем засобами UNITY PRO.

Спочатку в UNITY PRO у розділі проекту Derived FB Types розробляється **DFB тип** (DFB Type), на основі якого потім створюються екземпляри. При створенні DFB типу визначають: ім'я типу; інтерфейс блоку (вказують ім'я та тип входних та вихідних параметрів функціонального блоку); внутрішні змінні та внутрішні екземпляри функціональних блоків, в яких будуть зберігатися проміжні результати між викликами; створюють програму для функціонального блоку на мовах ST, IL, LD або FBD. Використання екземплярів DFB аналогічно як і EFB.

Виходи EFB та DFB є змінними екземпляру, тобто до них можна звертатись як до інших змінних. Для цього вказується ім'я екземпляру а потім через крапку назва виходу. Так наприклад запис "Timer1.Q" означає, що йде звернення до виходу Q екземпляру з іменем "Timer1".

### 3. 2 Використання бібліотечних блоків.

В **бібліотеці типів** UNITY (Types Library) доступна велика кількість функцій, процедур та елементарних функціональних блоків. Вони можуть бути використані в будь-якій із мов програмування. Так наприклад функція ADD призначена для додавання в мові FBD, однак її можна викликати і в ST, LD та IL, хоча для додавання там зручніше використовувати відповідний оператор "+".

Найбільш загальні програмні блоки зведені у розділі стандартної бібліотеки. Частина з них наведено в таб.3.1. Ряд блоків з бібліотеки управління, які використовуються при реалізації алгоритмів регулювання наведено у таб.3.2.

Таблиця 3.1.Стандартна бібліотека (Standard Library, часткова вибірка)

<b>Математичні</b> (сімейство Mathematics)		
ADD – додавання	SUB – віднімання	SQRT – квадратний корінь
MUL – множення	MOVE – присвоєння	
DIV – ділення	NEG – зміна знаку	
<b>Порівняння</b> (сімейство Comparison)		
EQ – дорівнює (=)	GE – більше або дорівнює ( $\geq$ )	LE – менше або дорівнює ( $\leq$ )
NE – не дорівнює ( $\neq$ )	GT – більше ( $>$ )	LT – менше ( $<$ )
<b>Логічні</b> (сімейство LOGIC)		
AND – логічне ТА	SR – SR тригер	FE – визначення переднього фронту
OR – логічне АБО	RS – RS тригер	RE – визначення заднього фронту
XOR – виключне АБО	SET – встановлення в лог. "1"	
NOT – логічне НІ	RESET – встановлення в лог. "0"	
<b>Статистичні</b> (сімейство Statistical)		
MUX – мультиплексор	LIMIT – обмеження, LIMIT_IND – обмеження з індикацією	MAX – вибір максимального
SEL – бінарний вибір	AVE – середнє значення	MIN – вибір мінімального
<b>Таймери та лічильники</b> (сімейство Timers & Counters)		
CTD – лічильник вниз	CTUD – лічильник вгору/вниз	TOF – таймер з затримкою на виключення
CTU – лічильник вгору	TP – таймер формування імпульсу	TON – таймер з затримкою на включення
<b>Перетворення типів</b> (сімейство Type to Type)		
INT_TO_REAL	WORD_TO_BIT	WORD_TO_INT
REAL_TO_INT	BIT_TO_WORD	INT_TO_WORD
<b>Цілочисельне регулювання</b> (сімейство CLC_INT)		
PID_INT – ПІД регулятор	PWM_INT – широтно-імпульсне перетворення	SERVO_INT – управління серводвигунами

Таблиця 3.2Бібліотека управління (Control Library, часткова вибірка).

PI_B – базовий ПІ регулятор	PWM1 – широтно-імпульсна модуляція
PIDFF – повний ПІД регулятор	SERVO – управління серводвигунами
LAG_FILTER – фільтрація (аперіодична ланка)	SAMPLETM – шаблон часу для періодичного виклику блоків регулювання
SCALING – масштабування	DTIME – транспортне запізнення

**Математичні функції.** Математичні функції призначені для виконання таких базових операцій як додавання, віднімання, множення, ділення, присвоєння, а також тригонометричних, експоненційних та інших математичних функцій. Базові математичні операції в основному призначені для FBD, оскільки в інших мовах для цього використовуються відповідні оператори.

Більшість функцій у бібліотеці представлені для різних типів даних таких як INT, DINT, UINT, UDINT та REAL. Так для додавання цілих чисел можна використати функцію ADD\_INT, або універсальну функцію ADD. Тим не менше, використання універсальної (за типом даних) функції не дозволяє використовувати в якості її параметрів різні за типом дані. Надалі ми будемо розглядати тільки універсальні за типом даних функції.

Всі розглянуті в таб.3.1 функції мають один вихід – результат виконання математичної функції. Функції SUB (віднімання) та DIV(ділення) мають по два

входи, функції ADD (додавання) та MUL (множення) можуть мати від 2-х до 32-х входів (вибирається після встановлення), всі інші – один вхід. На рис.3.1 показаний приклад використання математичних функцій для реалізації залежностей:

$$X = \frac{A \cdot B \cdot C + D - E + \sqrt{F} + G}{H};$$

$$Y = -(A \cdot B \cdot C + D - E + \sqrt{F} + G);$$

$$Z = A \cdot B \cdot C + D - E + \sqrt{F} + G;$$

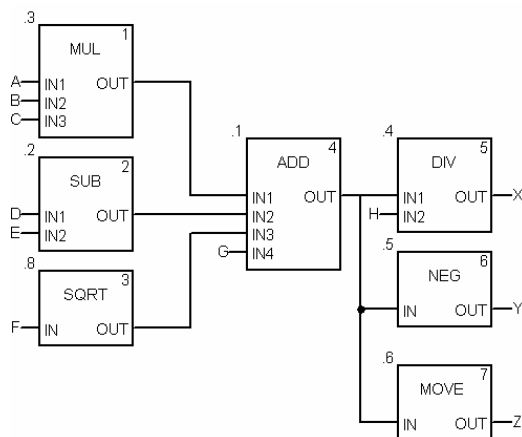


Рис.3.1. Приклад використання математичних функцій в FBD

**Функції порівняння.** Дані функції використовуються для порівняння двох або більше вхідних величин в мові FBD та LAD (однак дозволяється їх використання і в інших мовах). Результат порівняння є булевою величиною ("0" або "1"). За допомогою функцій порівняння та входів EN зручно проводити управління виконання логікою програми.

На рис.3.2 показаний приклад використання функцій порівняння для виконання логічного управління. При умові що  $A=B=C$ , змінній Y буде присвоєне значення 0. При умові, що  $D \geq F$ , виконається дія  $Y := D + F$ .

Альтернативою функціям

порівняння в FBD може бути ST вираз, який повертає булевий результат.

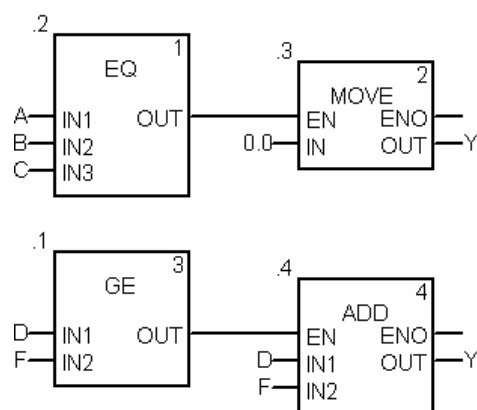


Рис.3.2. Приклад використання функцій порівняння в FBD

**Логічні блоки.** Логічні функції AND, OR, XOR, NOT використовуються для логічних операцій переважно в мові FBD, так як у мові ST для цього використовуються оператори, а в LD – спеціальні графічні оператори. Функції SET та RESET, не мають входів (за винятком EN). При виклику функції SET, вихідний параметр виставляється у логічну одиницю. Аналогічно, при виклику функції RESET вихідний параметр виставляється в логічний нуль.

Функції FE та RE служать для відлову відповідно заднього та переднього фронтів у мовах FBD, ST та IL. У якості вхідного параметру використовується змінні типу EBOOL.

Функціональні блоки SR та RS є тригерами. Тобто при поданні логічної "1" на вхід S на виході по передньому фронту встановлюється логічна "1", а на R – логічний "0". SR та RS тригери відрізняються пріоритетністю входів S та R при одночасному поданні на них логічних "1". У RS тригері пріоритетний вхід R, а у SR, вхід S. Нагадаємо, що функціональні блоки потребують створення **екземплярів**.

На рис.3.3 показаний приклад використання логічних блоків в FBD. Змінна Bool1 встановиться в "1" в тому випадку, коли  $A \geq B$  або  $C > D$ . В іншому випадку, ніяких дій зі змінною Bool1 проводитись не буде. Змінна Bool3 виставиться в "1", коли  $C > D$  і встановиться у "0" по передньому фронту змінної ebool2, навіть якщо C буде більше D, оскільки вхід R1 має вищий пріоритет. Однак, якщо на наступний цикл після виникнення логічної "1" в ebool2, C буде більше ніж D, умова фронту сигналу вже не буде працювати, тому Bool3 знову переведеться в логічну "1". У разі виникнення переднього фронту у змінній ebool2 змінна B буде збільшуватися на 1.0.

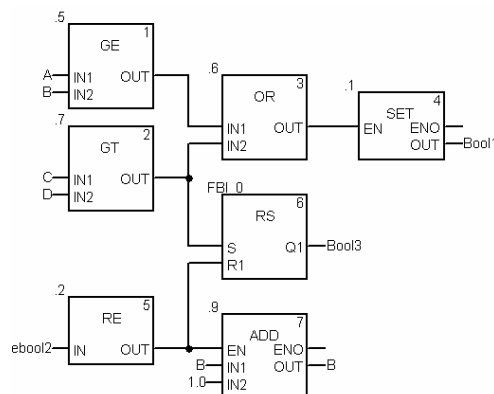


Рис.3.3. Приклад використання логічних блоків в FBD

**Статистичні функції.** Функції MIN та MAX призначені для запису відповідно мінімального та максимального значення серед вхідних сигналів у вихідний сигнал. Кількість входів може бути змінена до 32.

Функція MUX є мультиплексором, вихід OUT якої перемикається на один із входів IN0-IN30, по номеру що подається на вхід K. Тобто якщо  $K=2$   $OUT:=IN2$ .

Функція SEL перемикає вихід на один із двох входів IN, в залежності від значення вхідного параметру G. Тобто OUT:=IN0 при G=FALSE, OUT:=IN1 при G=TRUE.

Функція AVE реалізує розрахунок зваженого середнього за формулою:

$$Y = \frac{\sum (K_i \cdot X_i)}{\sum (K_i)}$$

де  $K_i$  – коефіцієнт  $i$ -го вхідного значення;  $X_i$  –  $i$ -те вхідне значення.

Коефіцієнт і значення вхідного сигналу являються парою вхідних параметрів. Тобто три пари сигналів потребують шість входів ( $K\_X1\dots K\_X6$ ). Непарні входи є коефіцієнтами, парні – значеннями сигналів.

Функція LIMIT забезпечує обмеження вхідної величини IN по мінімуму (вих MN) та по максимуму (вхід MX). Процедура LIMIT\_IND працює аналогічно LIMIT однак має додаткові виходи Y\_MAX та Y\_MIN, який виставляється в логічну "1" при досягненні відповідно максимального та мінімального значень на вході.

Приклад використання статистичних функцій в FBD наведені на рис.3.4. Для кращого розуміння приклад показаний як в режимі редагування та і в режимі виконання програми з анімацією, де для всіх вхідних та вихідних параметрів показані числові значення. На виході блоку ".4" формується зважене середнє 3-х сигналів ( $K\_X2:=D$ ,  $K\_X4:=E$ ,  $K\_X8:=F$ ) з коефіцієнтами ( $K\_X1:=0.1$ ,  $K\_X3:=0.25$ ,  $K\_X5:=0.5$ ). Блок ".1" вибирає максимальне значення серед 4-рьох входів (OUT:=98.0), блок ".5" переключує вихід на перший вхід (OUT:=IN1). Серед двох вхідних сигналів блоку ".2" вибирається 0-вий (OUT:=IN0), оскільки  $G=FALSE$  (для блока ".6" не справджується умова  $IN1 < IN2$ ). Блок ".7" обмежує по максимуму вхідну величину ( $MX:=95$ ), тому на виході OUT:=95, а на виходах MN\_IND:=FLASE та MX\_IND:=TRUE.

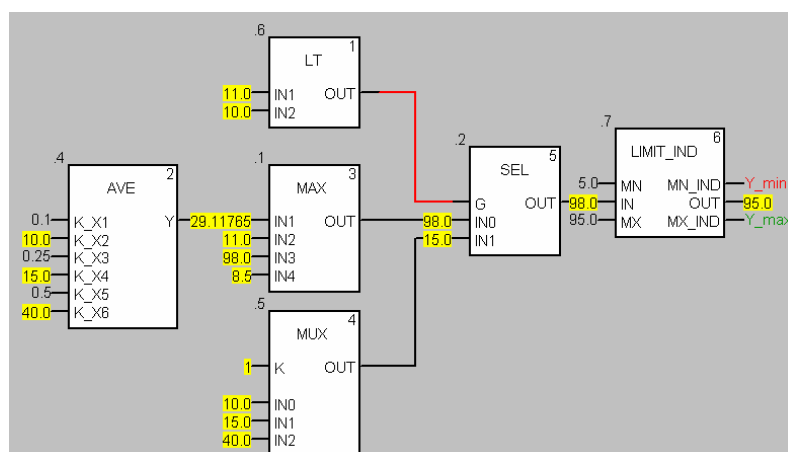


Рис.3.4. Приклад використання статистичних функцій в FBD в режимі анімації

**Таймери та лічильники.** Таймери та лічильники в UNITY реалізовані у вигляді функціональних блоків, тому потребують попереднього створення екземплярів.

Лічильники доступні у вигляді 3-х функціональних блоків: STU – підрахунок імпульсів зі збільшенням, STD – підрахунок імпульсів зі зменшенням, STUD – реверсивний лічильник.

Лічильник STU збільшує плинне значення CV, по передньому фронту сигналу на вході CU. На вході PV задається уставка. При досягненні плинного значення  $CV \geq PV$ , вихід Q:=TRUE. При подачі на вхід R:=TRUE, скидає плинне значення в нуль.

Лічильник STD зменшує плинне значення CV, по передньому фронту сигналу на вході CD. На вході PV задається уставка. При досягненні плинного значення  $CV \leq 0$ , вихід Q:=TRUE. При подачі на вхід LD:=TRUE, у плинне значення записується значення уставки, тобто  $CV:=PV$ .

Лічильник STUD, об'єднує в собі функції двох лічильників STU та STD. Входи CU та CD призначені відповідно для збільшення та зменшення плинного значення лічильника CV. При  $CV \geq PV$ , вихід QU:=TRUE. При  $CV \leq 0$ , вихід QD:=TRUE. Якщо LD=TRUE,  $CV:=PV$ . Якщо R=TRUE,  $CV:=0$ .

Таймери TON, TOF та TP мають вхід IN – сигнал запуску таймеру, PT – часова уставка таймеру, вихід таймера Q, та плинне значення таймеру ET. Діаграми роботи даних таймерів показані на рис.3.5.

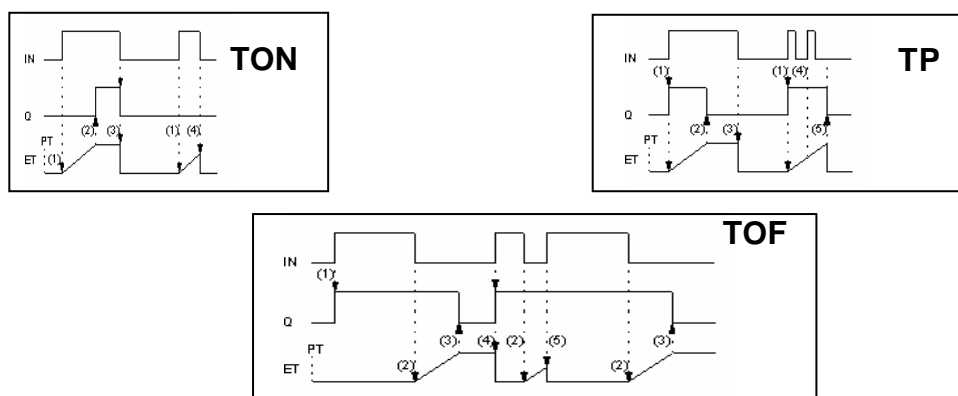


Рис.3.5. Графіки режимів роботи таймерів TON, TOF та TP.

Приклад використання лічильника STUD та таймера TON в FBD показаний на рис.3.6. Вихід QU лічильника "CounterUD" спрацює при досягненні CV уставки ( $CV=20$ ). Після досягнення уставки лічильника змінна Y\_bool:=TRUE і запуститься таймер Timer1. Через 2с 100мс після запуску таймера скинеться плинне значення лічильника "CounterUD", тобто CounterUD.CV:=0. Слід звернути увагу на використання виходу EFB "Timer1.Q" в якості фактичного параметру на вході R блоку "CounterUD".

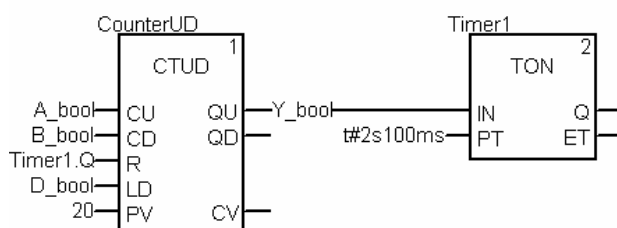


Рис.3.6. Приклад використання функціональних блоків TON та STUD

**Функції перетворення типів.** Функція `INT_TO_REAL` перетворює ціле значення в дійсне, а `REAL_TO_INT` – навпаки. Дані функції необхідні у випадках, коли над одними даними треба проводити як цілочисельні операції так і операції з плаваючою комою. На рис.3.7 показаний приклад, де аналогову вхідну змінну необхідно помножити на коефіцієнт `A` (дійсне число) після чого результат подати на аналоговий вихід. Враховуючи, що значення аналогових входів записується у цілочисельні змінні `%IW`, необхідно спочатку його перетворити в тип `REAL` (функція `INT_TO_REAL`), після чого помножити на `A`. Враховуючи, що значення аналогових виходів зберігається в цілочисельних змінних `%QW`, розрахований добуток попередньо перетворюється в цілочисельний формат (функція `REAL_TO_INT`).

Функції `INT_TO_WORD` та `WORD_TO_INT` забезпечують перетворення відповідно цілочисельних даних `INT` в формат слова `WORD` та навпаки. Формат `WORD` використовується для побітових операцій над змінними.

Функція `WORD_TO_BIT` призначена для побітового розкладення входу `IN` типу `WORD`. Виходи `BIT0...BIT15` виставляються в `TRUE` або в `FALSE` відповідно до значення бітів у вхідному слові. Біти у слові рахуються від молодшого (0-й) до старшого (15-й). Функція `BIT_TO_WORD` – навпаки, по значенням входів `BIT0...BIT15` формує вихідне значення `OUT` типу `WORD`. На рис.2.14 показаний приклад, де значення змінної `%IW0.1.1` інтерпретується як набір бітів. Спочатку значення перетворюється в тип `WORD` (функція `INT_TO_WORD`), після чого значення 0-го біту записується в `a_bool`, 1-го в `b_bool`, 7-го в `c_bool`.

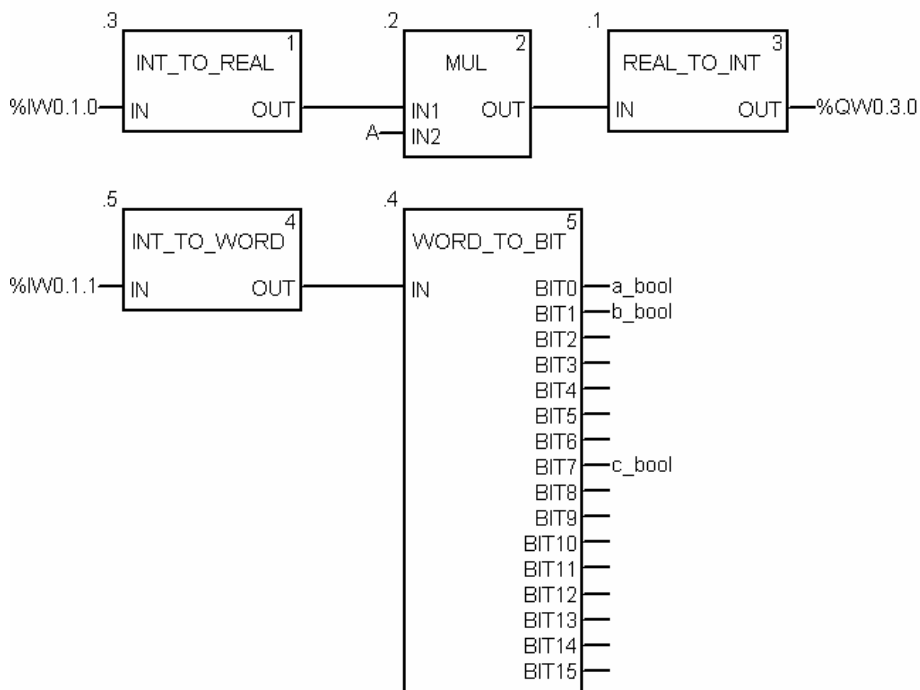


Рис.3.7. Приклад використання функцій перетворення типів в FBD

### 3.3 Створення функціональних блоків користувача

Часто виникають задачі, які передбачають обробку даних за однаковим алгоритмом. Для того, щоб не повторювати однаковий код програми з різними



даними декілька раз, є сенс викликати один і той самий код, однак з різними фактичними параметрами. За таким самим принципом реалізовані функції, процедури та функціональні блоки, які зберігаються у бібліотеці.

При необхідності виклику частин програм декілька раз можна використати підпрограми (SR). Однак в Unity Pro в підпрограми не можна безпосередньо передавати параметри, тобто вони завжди викликаються як частина коду.

Для можливості передачі параметрів до частини програми у багатьох програмних середовищах сучасних ПЛК існують функції та функціональні блоки користувача. В Unity Pro немає можливості створення функцій користувача, однак є ймовірність створення функціональних блоків користувача, які носять назву Derived Function Block (DFB). У принципі якщо DFB-типі не описати змінні, що організують внутрішню пам'ять то в першому наближенні він може бути використаний як функція.

Для використання функціональних блоків користувача в Unity Pro, спочатку у розділі проекту Derived FB Types розробляється DFB-тип, на основі якого потім створюються екземпляри. Екземпляри функціональних блоків користувача створюються та використовуються аналогічна EFB.

Процес створення DFB типу схожий на створення структурного DDT. При цьому вказується ім'я типу, описується інтерфейс та внутрішня структура типу (рис.3.1). Інтерфейс блока (формальні параметри) описується входами (Inputs) виходами (Outputs) та входами і виходами (Inputs/Outputs), використання яких у програмі аналогічно як для елементарних функціональних блоків. На рис.3.1 показано створення DFB-типу, для якого описані 4 вхідні формальні параметри (Start, LS1, LS2, ManPump) та один вихідний (M1). Порядок розміщення інтерфейсних параметрів вказується у властивості «по».

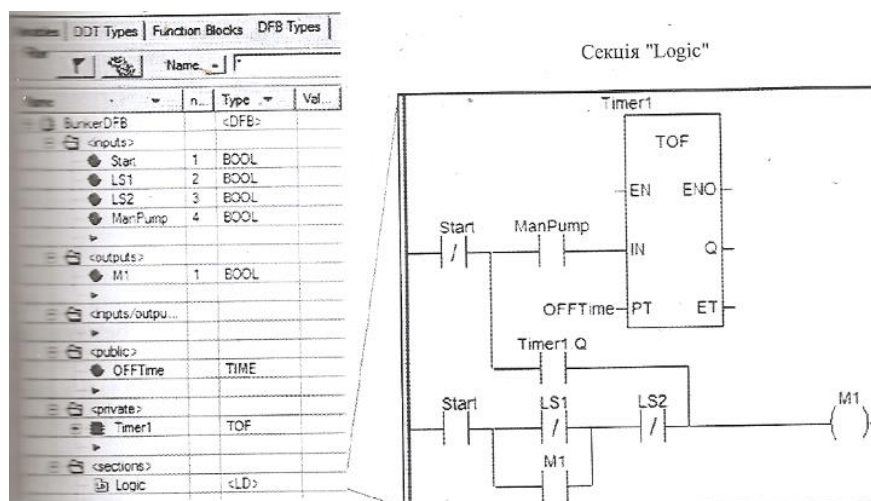


Рис 3.1 Створення типу функціонального блока користувача DFB.

Програма функціонального блока описуються в розділі Sectins. Секції можуть бути створені з використанням мов LD, ST, IL або FBD. У секціях можна використовувати тільки входи, виходи, входи/виходи, локальні змінні або екземпляри та глобальні змінні функціонального блока, бібліотечні функції і

процедури. Дозволяється також використовувати системні біти та слова. На рис. 3.1 функціональний блок вміщує тільки одну програмну секцію – «Logic».

Крім інтерфейсної частини, тип може вміщувати *локальні дані*, які розташовуються в розділі «private». Локальні дані по суті є пам'яттю функціонального блока, яка інкапсульована в його екземпляр і недосяжна із зовнішньої програми. У прикладі на рис.3.1 в якості локальних даних використовується екземпляр таймера з іменем *Timer1*. Слід звернути увагу, що налаштування *Timer1* і доступ до його полів можливий тільки із середини функціонального блока, тобто з його програмних секцій.

Для можливості доступу до змінних функціонального блока не описуючи їх в інтерфейсі, ці змінні визначають як глобальні в розділі *Public*. У прикладі на рис.3.1 визначена глобальна змінна *OFFTime*, за допомогою якої індивідуально можна налаштувати таймер *Timer1* у програмній секції функціонального блока. При створенні екземпляра значення глобальної змінної при ініціалізації можна прописати в поле *Value*. У програмі на рис. 3.2 під час ініціалізації блока глобальна змінна *OFFTime* буде отримувати значення T#5s. До глобальної змінної блока в програмі можна також звернутися як до виходу блока тобто через крапку (Рис.3.2).

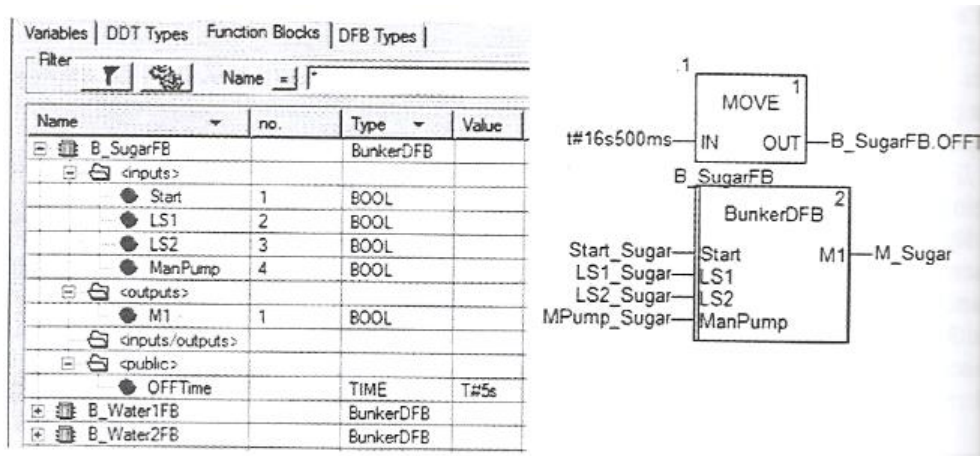


Рис.3.2 Створення екземплярів функціонального блока користувача DFB та їх використання в програмі

## 4. Загальні принципи використання бібліотечних FFB для побудови контурів регулювання.

### 4.1 Структура контурів регулювання

Узагальнена функціональна структура контурів регулювання з використанням ПЛК показана на рис.4.1. Бібліотеки UNITY PRO нараховують велику кількість блоків для реалізації наведених на рис.4.1 функцій. Частина з них присутні у бібліотеці для сумісності з проектами, які конвертуються з середовищ CONCEPT та PL7. Це такі сімейства блоків:

- CLC\_INT бібліотеки *Base Lib* (PID\_INT, PWM\_INT, SERVO\_INT);
- CLC бібліотеки *Obsolete Lib* (PI1, PID1, PIDP1 та інші);
- CLC PRO бібліотеки *Obsolete Lib* (PI, PID, PID\_P, PIP, PPI, PWM та інші).

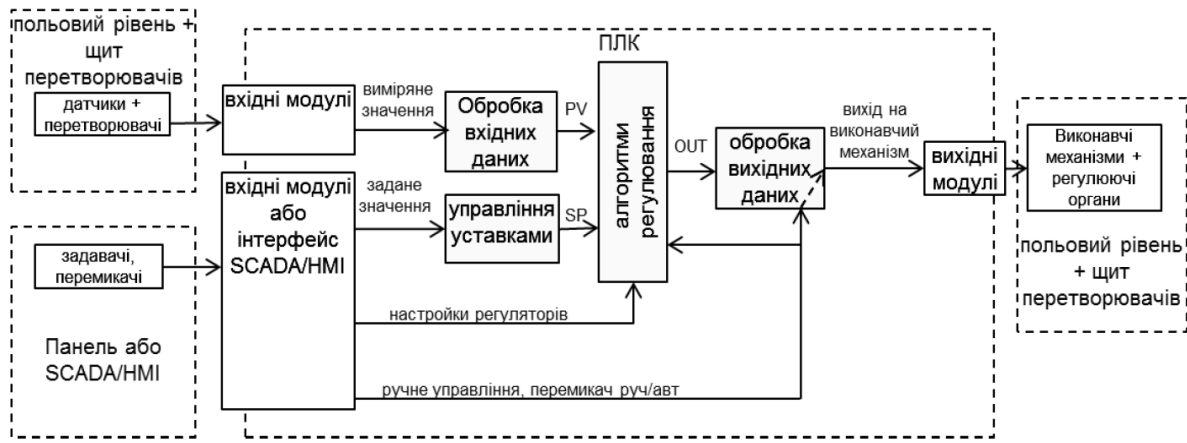


Рисунок 4.1– Структура контурів регулювання

Сімейства бібліотеки *Obsolete Lib* не рекомендується використовувати у новостворюваних проектах UNITY PRO. Процедури сімейства *CLC\_INT* бібліотеки *StandardLib* реалізують цілочисельне регулювання аналогічне тому, яке використовувалось в PL7 PRO.

Більшість з наведених на рис.4.1 функцій реалізуються з використанням елементів FFB бібліотеки "*Control Library*", яка поставляється разом з UNITY PRO. Використанню FFB з цієї бібліотеки якраз і присвячений даний розділ.

У центрі контурів регулювання знаходиться регулятор (алгоритми регулювання), який в UNITY PRO може бути реалізований через один із доступних функціональних блоків із сімейства "*Controller*": *PI\_B* (ПІ-регулятор), *PIDFF* (ПІД-регулятор), *STEP2* (2-позиційний регулятор), *STEP3* (3-позиційний регулятор). Окрім самих регуляторів додатково з цього ж сімейства можуть бути використані блоки *AUTOTUNE* (автонастроювання), *IMC* (коректор моделі), *SAMPLETM* (диспетчер виклику функціональних блоків).

Значення регульованої величини, яке поступає з датчиків на програмний регулятор, попередньо повинен бути оброблений. Це зв'язано з тим, що всі реалізації регуляторів сімейства "*Controller*" оперують зі значеннями типу *REAL*, а оцифроване значення з аналогових вхідних модулів має тип *INT* (в діапазоні 0-10000). Крім того вимірювальне значення може бути зашумлене та потребувати додаткової обробки. Для обробки вхідних даних контурів управління можуть бути використані блоки сімейств "*Conditioning*", "*Measurement*" та "*Mathematics*".

Контури регулювання повинні мати можливість працювати в ручному режимі, при цьому повинна бути забезпечена безударність переходу. Крім того, для деяких типів виконавчих механізмів необхідне додаткове перетворення сигналу. Для такого типу перетворення вихідного сигналу регуляторів призначені блоки сімейства "*Output Processing*".

Для формування та управління уставками регуляторів можна скористатися FFB сімейства "*Setpoint management*".

На будь якому з етапів перетворення та алгоритмічної обробки даних можуть знадобитися блоки додаткової обробки з сімейства "*Conditioning*".

## 4.2. Режим слідування (Tracking)

Багато функціональних блоків бібліотеки управління підтримують управління операційним режимом. Доступні такі режими:

- Tracking (режим слідування);
- Manual/Automatic (ручний/автомат)

Режим слідування дає можливість переводити функціональний блок в стан управління його виходом із зовні (рис.4.2).

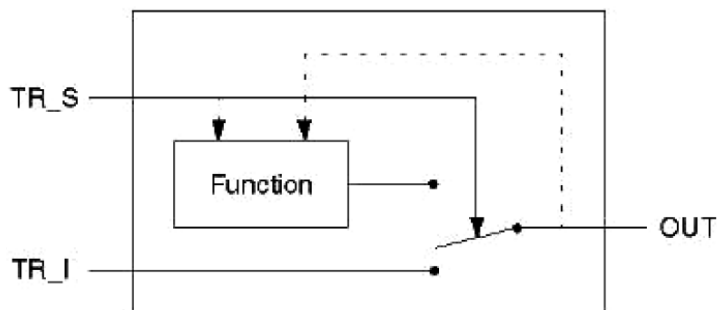


Рисунок 4.2– Режим слідування

Управління режимом проводиться сигналом  $TR\_S$  (TRacking Switch). У **нормальному режимі** ( $TR\_S = 0$ ) значення виходу функціонального блоку  $OUT$  визначається закладеним в нього алгоритмом ( $Function$ ). У **режимі слідування** ( $Tracking$ ,  $TR\_S = 1$ ) вихід  $OUT$  дорівнює значенню входу  $TR\_I$  ( $TRacking Input$ ).

Для забезпечення безударності переходу між режимами, внутрішній алгоритм відслідковує значення виходу. Тобто в момент переходу з режиму  $Tracking$  в нормальний режим, вихід алгоритму буде дорівнювати входу  $TR\_I$ .

Режим  $Tracking$  може бути використаний в наступних ситуаціях:

- 1) ініціалізація функціонального блоку в початковій фазі функціонування, тобто при першому виконанні блоку;
- 2) режим слідування функціонального блоку в дубльованому ПЛК (в системах Hot Standby TSX Premium/Quantum), для гарантування безударності запуску резервного контролера;
- 3) безпосереднє управління виходом функціонального блоку, тобто коли вихід блоку повинен визначатися зовнішнім алгоритмом.

## 4.3. Режими Ручний/Автомат (Manual/Automatic)

Вибраний режим ручний/автомат визначається значенням входу  $MAN\_AUTO$  (рис.4.3). У автоматичному режимі ( $MAN\_AUTO=1$ ) вихід функціонального блоку  $OUT$  дорівнює виходу внутрішнього алгоритму ( $Function$ ).

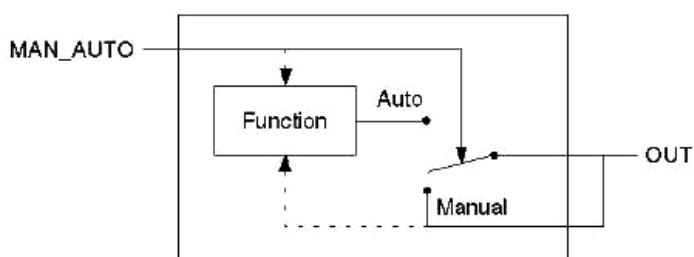


Рисунок 4.3. Режим Руч/Авт

У ручному режимі ( $MAN\_AUTO=1$ ) вихід  $OUT$  не залежить від розрахункового значення закладеного алгоритмом, і може бути змінена ззовні (наприклад засобами HMI).

Для забезпечення безударності переходу між режимами, внутрішній алгоритм відслідковує значення виходу. Тобто, при переході з режиму з  $Manual$  в  $Auto$ , вихід алгоритму буде дорівнювати останньому значенню виходу.

Якщо функціональний блок підтримує обидва типи операційних режимів  $Tracking$  і  $Manual/Automatic$ , режим  $Tracking$  має вищий пріоритет (рис.4.4). Це значить, що в режимі  $Tracking$  вихід  $OUT$  буде дорівнювати  $TR\_I$  незалежно від стану  $MAN\_AUTO$ .

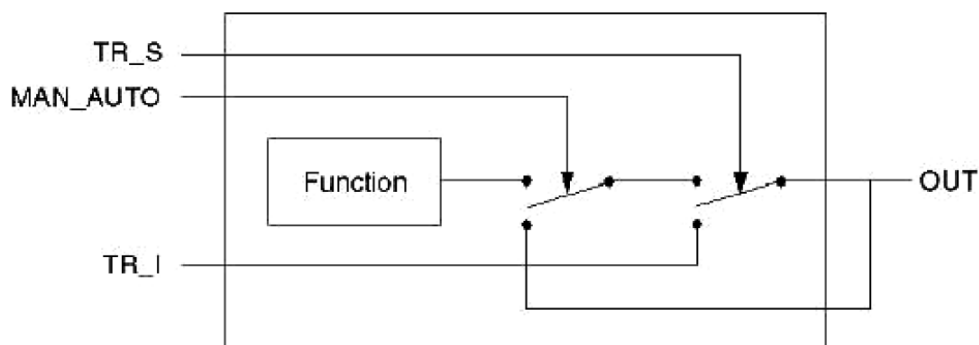


Рисунок 4.4– Пріоритетність режимів.

#### 4.4. Періодичність виклику FFB та контроль за помилками

Багато функціональних блоків зав'язані на часових інтервалах між викликами, наприклад для розрахунку інтегральної та диференційної складової в ПІД- регуляторі. У багатьох ПЛК для правильної роботи таких блоків необхідно викликати їх періодично. У UNITY PRO функціональні блоки бібліотеки управління  $ControlLIB$  розраховують ці інтервали автоматично, що дає можливість викликати їх без забезпечення періодичності. Тим не менше, в деяких випадках бажано задати періодичність виклику, наприклад:

- оптимізація часу виконання циклів, розподіливши виклик операцій регулювання між різними циклами;
- покращення якості управління в контурах з серводвигунами, для зменшення частоти обробки блоку  $SERVO$ ;
- мінімізація навантаження на виконавчі механізми (збільшення періодичності виклику – зменшення частоти зміни положення);

Для управління періодичністю виклику функціональних блоків можна використати  $EFB$  типу  $SAMPLETM$ , вихід якого може управляти входом  $EN$  потрібного функціонального блоку управління.

Контроль виконання багатьох FFB бібліотеки проводиться шляхом аналізу вихідного слова  $STATUS$ . Призначення перших 8-ми біт цього слова (0..7) однакові для всіх функціональних блоків, призначення інших 8-ми (8..15) залежить від функціонального блоку.

Таблиця 4.1.

Bit 0 = 1	Помилка при розрахунку значення з плаваючою комою (наприклад отримання квадратного кореня з від'ємного значення)
Bit 1 = 1	Недозволене значення було записане у вхідне значення з плаваючою комою із за того, що: значення не являється з плаваючою комою; значення являється нескінченністю
Bit 2 = 1	Ділення на 0 при розрахунку з плаваючою комою
Bit 3 = 1	Переповнення пам'яті при розрахунку з плаваючою комою
Bit 4 = 1	Вхідне значення виходить за діапазон; використовується значення обмежене блоком
Bit 5 = 1	Основний вихід функціонального блоку досяг нижньої межі
Bit 6 = 1	Основний вихід функціонального блоку досяг верхньої межі
Bit 7 = 1	Верхня та нижня межі однакові

## Реалізація законів регулювання в UNITY PRO

### 5. Огляд блоків сімейства Controller.

#### 5.1 Загальне представлення сімейства Controller (бібліотека Control Lib).

Бібліотека UNITY нараховує велику кількість блоків для реалізації стандартних законів регулювання. Частина з них присутні у бібліотеці для сумісності з проектами, які конвертуються з середовищ CONCEPT та PL7. Це такі сімейства блоків:

- CLC\_INT бібліотеки Base Lib (PID\_INT, PWM\_INT, SERVO\_INT);
- CLC бібліотеки Obsolete Lib (PI1, PID1, PIDP1 та інші);
- CLC PRO бібліотеки Obsolete Lib (PI, PID, PID\_P, PIP, PPI, PWM та інші).

Наведені сімейства блоків не рекомендується використовувати у новостворюваних проектах UNITY PRO. Для реалізації алгоритмів регулювання пропонується використовувати бібліотеку Control Lib зокрема блоки сімейств Controller, Output Processing, Setpoint Management. Сімейство Controller включає такі блоки:

- PI\_V – реалізовує ПІ закон регулювання;
- PIDFF – реалізовує ПІД закон регулювання;
- AUTOTUNE – для автонастройки PI\_V та PIDFF;
- SAMPLETM – для періодичного виклику алгоритмів регулювання;
- STEP2 – двохпозиційний регулятор;
- STEP3 – трьохпозиційний регулятор;

Серед блоків сімейства Output Processing можна виділити такі основні блоки:

- PWM1 – реалізовує ШІМ-алгоритм (широотно-імпульсна модуляція) для управління дискретними виконавчими механізмами по аналоговому сигналу та може бути використаний разом з PI\_V або PIDFF;
- SERVO – реалізовує ШІМ алгоритм з двома дискретними виходами для управління виконавчими механізмами з прямим і зворотнім ходом (МЕК,МЕО) та може бути використаний разом з PI\_V або PIDFF;

Блоки сімейства Setpoint Management призначені для забезпечення алгоритмів регулювання можливістю управління завданням. Зокрема функціональний блок SP\_SEL призначений для можливості переключення завдання з віддаленого на локальний при зв'язаному регулюванні.

Всі блоки бібліотеки *Control Lib*, алгоритм яких передбачає використання часових інтервалів (наприклад для інтегрування або диференціювання), розраховують ці інтервали як різницю між плинним та попереднім часом виклику блоку. Враховуючи значне використання ресурсів контролера даними блоками, для оптимізації роботи програми контролера рекомендується викликати їх періодично, зсунутими у часі відносно контурів, в яких вони використовуються. Так, наприклад, при наявності 10-ти контурів регулювання, можна викликати зв'язані в контурі блоки з періодичністю 100 мс, але зсунуті один відносно одного на один цикл.

Тобто через кожні 50 мс, протягом 10 циклів будуть оброблені всі контури. Періодичний виклик зі зсувом по часу можна забезпечити функціональним блоком *SAMPLETM*.

### Функціональний блок *SAMPLETM*

Всі блоки бібліотеки *Control Lib*, алгоритм яких передбачає використання часових інтервалів (наприклад для інтегрування або диференціювання), розраховують ці інтервали як різницю між плинним та попереднім часом виклику блоку. Це значить, що їх можна викликати аперіодично. Однак алгоритми регулювання потребують значні часові ресурси, що займає значну частину часу Задачі (*Task*), в якій вони викликаються. З іншого боку, більшість задач регулювання не потребують частої обробки, а отже їх контури можуть оброблятися рідше, ніж кожний цикл. Таким чином для оптимізації роботи програми контролера рекомендується обробляти контури періодично, і зсунутими у часі відносно один одного.

Так, наприклад, при наявності 10-ти контурів регулювання, можна викликати зв'язані в контурі блоки з періодичністю 100 мс, але зсунуті один відносно одного на один цикл. Тобто через кожні 100 мс, протягом 10 циклів будуть оброблені всі контури. Періодичний виклик зі зсувом по часу можна забезпечити функціональним блоком *SAMPLETM*.

Функціональний блок *SAMPLETM* з періодичністю, яка визначається вхідним параметром *INTERVAL*, на один цикл Задачі виставляє в значення *TRUE* вихід *Q*. Вхідний параметр *DELSCANS* визначає зміщення в циклах запуску внутрішнього таймеру блоку відносно першого циклу контролеру (після холодного старту).

На рис.5.1 показаний приклад використання 2-х екземплярів *SAMPLETM*, виходи *Q* яких з періодичністю однієї секунди будуть виставлятися на один цикл в *TRUE*. Включення цих виходів буде зміщене на один цикл один відносно одного.

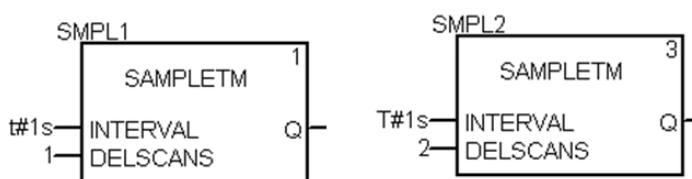


Рисунок 5.1– Використання 2-х екземплярів *SAMPLETM* зі зсувом на один цикл відносно

## 5.2. III регулятор PI\_V

Функціональний блок *PI\_V* реалізує III алгоритм регулювання (5.1).

$$OUT = kp \cdot \left( 1 + \frac{1}{ti \cdot p} \right) \cdot DEV \quad (5.1)$$

Функціональна схема III-регулятора зображена на рис.5.2. Параметри *PI\_V* наведені в таблиці табл.5.1. Як видно з рис.5.2, в алгоритмі спочатку розраховується розузгодження ( $DEV = PV - SP$ ), яке пройшовши через блок нечутливості (з зоною *dband*), використовується для розрахунку інтегральної складової (налаштовується *ti*) та пропорційної складової (налаштовується *kp*). Задане значення *SP* обмежується по мінімуму (*pv\_inf*) та по максимуму (*pv\_sup*). Якщо необхідно управляти ВМ в зворотному напрямку (команда *rev\_dir=TRUE*), розраховане значення інвертується (змінює знак).

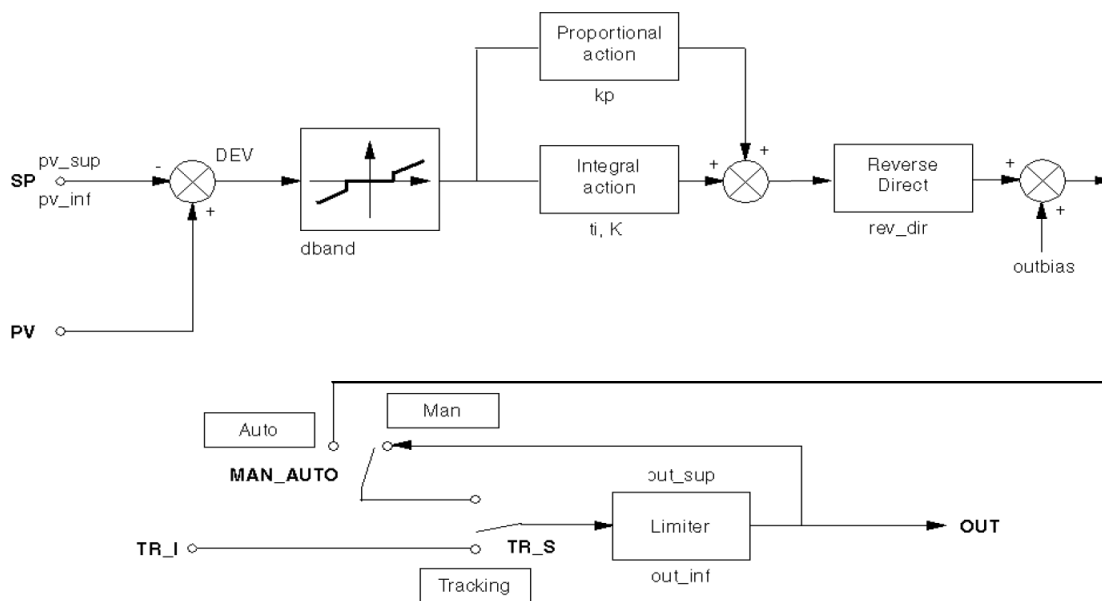


Рисунок 5.2– Функціональна схема III-регулятора *PI\_V*.

У автоматичному режимі ( $MAN\_AUTO=TRUE$ ) розраховане значення, пройшовши через блоки *Tracking* та *Limiter* подається на вихід *OUT*. В ручному режимі ( $MAN\_AUTO=FALSE$ ) вихід *PI\_V* регулюється ззовні функціонального блока.

Режим *Tracking* (слідкування) використовується для прямого регулювання *OUT* зовнішнім алгоритмом. У нормальному режимі *TR\_S* повинен бути *FALSE*, тобто *Tracking* вимкнений.

Параметрами *out\_inf* та *out\_sup* налаштовуються обмеження відповідно по мінімуму та максимуму вихідної величини (при  $en\_rcpu=TRUE$  обмеження не діють).

Функціональний блок *PI\_V* може працювати в режим III та II регулятора. Виставивши параметр  $ti=0s$ , блок *PI\_V* переключається в режим II-регулювання. У режимі III регулювання блок розраховує вихід *OUT* по інкрементальному розрахунку, а в II режимі регулювання – по абсолютному.



У П режимі регулювання (режим абсолютного розрахунку) вихідне значення розраховується за формулою:

$$OUT = kp \cdot DEV + outbias \quad (5.2)$$

Параметр *outbias* потрібен для того, щоб в П режимі регулятора задати зміщення початкової точки (при нульовому розузгодженні). Тобто, в залежності від знака розузгодження, вихід на виконавчий механізм буде зміщуватися в меншу або більшу сторону відносно *outbias*. Значення цього параметру як правило близький до 50%.

Всі команди, плинні, задані та вихідні значення доступні як параметри функціонального блоку. Налаштування алгоритму проводиться через структурний параметр *PARA* типу *Para\_PI\_B*, поля якого описані в таб.5.1.

На рис.5.3 показаний приклад використання функціонального блоку *PI\_B* для регулювання температури в контурі з умовною назвою *TIC1*. Для періодичного виклику ПІ-регулятора використовується функціональний блок типу *SAMPLETM*.

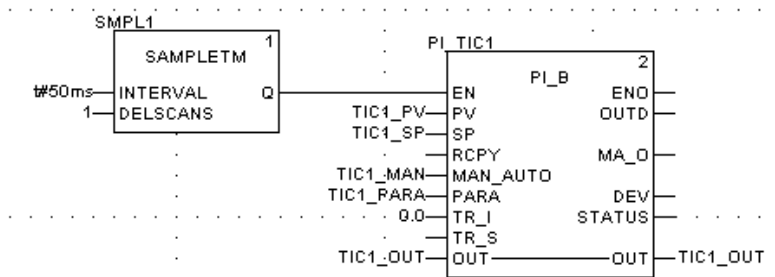
Настройка регулятора проводиться через структурну змінну *TIC1\_PARA*, яка попередньо створюється на базі типу *Para\_PI\_B*.

Таблиця 5.1 Параметри функціонального блоку *PI\_B*.

<b>Вхідні параметри</b>		
<i>PV</i>	<i>REAL</i>	значення вимірювальної величини (плинне значення)
<i>SP</i>	<i>REAL</i>	задане значення (уставка)
<i>RCPY</i>	<i>REAL</i>	копія значення регульованого виходу як база для інкрементального розрахунку (наприклад стан виконавчого механізму)
<i>MAN_AUTO</i>	<i>BOOL</i>	Режим роботи ПІ-регулятора: 1 : Автоматичний режим 0 : Ручний режим
<i>PARA</i>	<i>Para_PI_B</i>	Параметри регулятора (див. таб.6.4)
<i>TR_I</i>	<i>REAL</i>	Вхід слідкування/ініціалізації ( <i>Tracking</i> )
<i>TR_S</i>	<i>BOOL</i>	Команда на включення режиму слідкування/ініціалізації (1: Включити вхід <i>TR_I</i> )
<b>Вхідні/вихідні параметри</b>		
<i>OUT</i>	<i>REAL</i>	абсолютне значення виходу регулятора (в ручному режимі може змінюватися з зовні <i>PI_B</i> )
<b>Вихідні параметри</b>		
<i>OUTD</i>	<i>REAL</i>	інкрементальне значення виходу регулятора (див. <a href="#">праграф 6.2.5</a> )
<i>MA_O</i>	<i>BOOL</i>	Поточний режим виконання регулятора 1: Автоматичний режим 0: інший режим (ручний або режим слідкування)
<i>DEV</i>	<i>REAL</i>	Значення розузгодження ( <i>PV - SP</i> )
<i>STATUS</i>	<i>WORD</i>	Слово статусу (використовується для контролю за помилками виконання <i>PI_B</i> )

Таблиця 5.2. Опис структурного типу *Para\_PI\_B*

<i>id</i>	<i>UINT</i>	Використовується для алгоритму автопідстройки ( <i>AUTOTUNING</i> )
<i>pv_inf</i>	<i>REAL</i>	обмеження по мінімуму вхідної величини завдання
<i>pv_sup</i>	<i>REAL</i>	обмеження по максимуму вхідної величини завдання
<i>out_inf</i>	<i>REAL</i>	обмеження по мінімуму вихідної величини блоку
<i>out_sup</i>	<i>REAL</i>	обмеження по максимуму вихідної величини блоку
<i>rev_dir</i>	<i>BOOL</i>	0: пряма робота регулятора ( <i>SP-PV</i> ) 1: зворотна робота регулятора ( <i>PV-SP</i> )
<i>en_rcpy</i>	<i>BOOL</i>	1: використати вхід <i>RCPY</i>
<i>kp</i>	<i>REAL</i>	Коефіцієнт пропорційності
<i>ti</i>	<i>TIME</i>	Час інтегрування
<i>dband</i>	<i>REAL</i>	Зона нечутливості
<i>outbias</i>	<i>REAL</i>	зміщення виходу регулятора в П-режимі функціонування (при $ti=0s$ )



Name	Type	#	Value	Comment
TIC1_MAN	BOOL			режим руч/авт для контуру TIC1
TIC1_PARA	Para_PI_B			Парметри настройки контуру TIC1 (стабілізація температури)
id	UINT			ідентифікатор (службовий, не заповнюється)
pv_inf	REAL		0.0	обмеження по мінімуму вхідної величини
pv_sup	REAL		100.0	обмеження по максимуму вхідної величини
out_inf	REAL		0.0	обмеження по мінімуму вихідної величини
out_sup	REAL		100.0	обмеження по максимуму вихідної величини
rev_dir	BOOL		false	робота в прямому/зворотному режимі
en_rcpy	BOOL		false	вкл/відкл RCPY
kp	REAL		1.0	коефіцієнт пропорційності
ti	TIME		t#5s	час інтегрування
dband	REAL		0.1	зона нечутливості
outbias	REAL		50.0	зміщення для П-режиму (при $ti=0s$ )
TIC1_OUT	REAL			вихід на ВМ подачі пари
TIC1_PV	REAL			значення вимірювальної величини температури
TIC1_SP	REAL			задане значення температури для контуру TIC1

Рисунок 5.3– Приклад використання функціонального блоку *PI\_B*: зверху – фрагмент програми, знизу – опис змінних в редакторі Data Editor .

### 5.3 ПІД регулятор PIDFF

Функціональний блок *PIDFF* реалізовує ПІД алгоритм регулювання та має ряд додаткових функцій. Функціональна схема регулятора наведена на рис.5.4. Параметри *PID\_FF* наведені в таблиці табл.5.3, приклад виклику на рис.5.5. Нижче наведемо особливості регулятора.

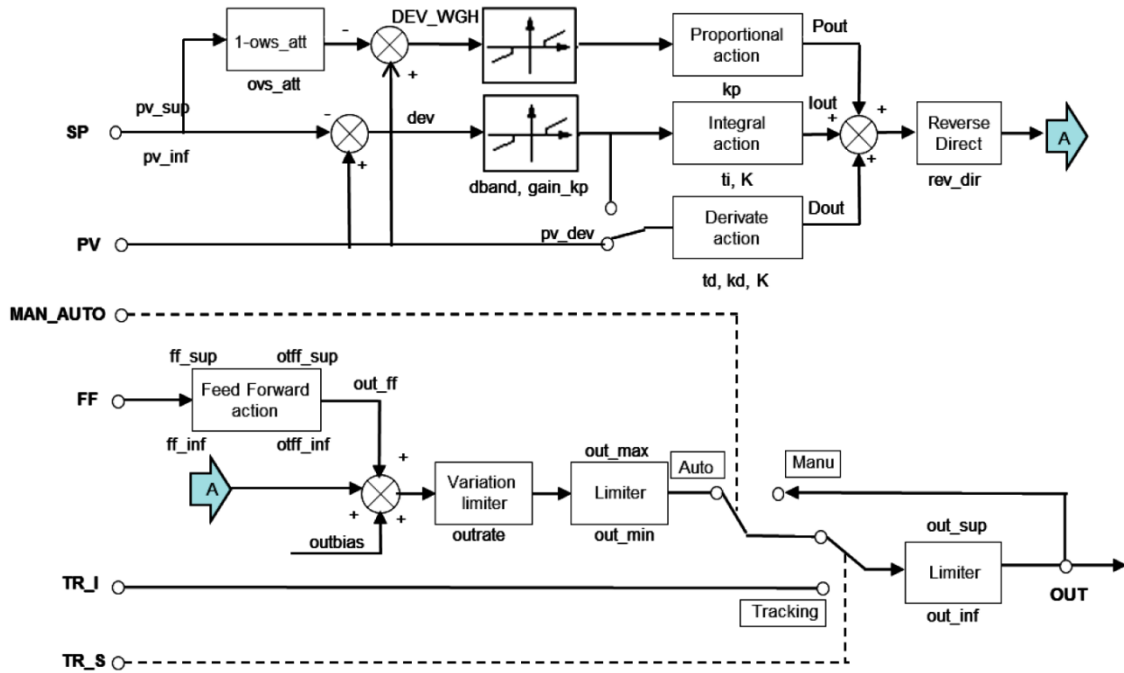


Рисунок 5.4– Функціональна схема ПІД-регулятора PIDFF.

$$OUT = kp \cdot \left( 1 + \frac{1}{ti \cdot p} + \frac{td \cdot p}{1 + \left(\frac{td}{kd}\right) \cdot p} \right) \cdot dev \quad (5.3)$$

Розрахунок виходу регулятора з паралельною (*parallel*) структурою наведений в (4.4) .

$$OUT = \left( kp + \alpha \cdot \frac{1}{ti \cdot p} + \alpha \cdot \frac{td \cdot p}{1 + \left(\frac{td}{kd}\right) \cdot p} \right) \cdot dev \quad (5.4)$$

У формулах (5.3) та (5.4)  $kp$ - коефіцієнт пропорційності,  $ti$ - час інтегрування,  $td$ - час диференціювання,  $kd$ - коефіцієнт диференціювання,  $dev = (PV - SP)$ - розузгодження,  $\alpha$ - коефіцієнт масштабування, який розраховується за формулою

$$\alpha = \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf} \quad (5.5)$$

Формули (5.3) та (5.4) справедливі тільки для певних режимів роботи *PIDFF*.

У змішаній структурі ( $mix\_par = FALSE$ ) коефіцієнти підсилення  $K$  інтегральної та диференційної складової (рис.5.4) будуть рівними коефіцієнту пропорційності  $K=kp$ . У паралельній структурі ( $mix\_par = TRUE$ ) коефіцієнти підсилення розраховуються за формулою

$$K = \alpha = \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf} \quad (5.6)$$

При включеній інтегральній складовій  $ti > 0$ , вхід для пропорційної складової розраховується за формулою:

$$DEV\_WGH = (PV - (1 - ovs\_att) \cdot SP) \quad (5.7)$$

де  $ovs\_att$  – ваговий коефіцієнт (від 0 до 1), який призначений для запобігання перерегулювання при зміні завдання.

При  $ovs\_att = 0$ , розузгодження для пропорційної складової рахується за класичними формулами (5.3) або (5.4). При  $ovs\_att = 1$  зміна завдання ніяк не впливає на розрахунок пропорційної складової, оскільки на вхід її подається тільки виміряне значення. Тобто в цьому режимі розузгодження використовується тільки в інтегральній та диференційній (при  $pv\_dev = TRUE$ ) складових. Значення в діапазоні 0-1 вказує наскільки пропорційна складова регулятора буде чутливою до зміни завдання.

Регулятор має диференційну складову (*Derivate action*) яка може працювати як по розузгодженню (при  $pv\_dev = TRUE$ ) так і по плинному значенню (при  $pv\_dev = FALSE$ ). Аналогічно до попередньої настройки, режим роботи  $pv\_dev = FALSE$ , виключає реакцію диференційної складової на зміну завдання. Слід також звернути увагу, що інтегральна складова в формулах (5.3) та (5.4) включає додатковий фільтр 1-го порядку, який налаштовується параметром  $kd$ .

На відміну від  $PI\_B$ , блоки нечутливості для  $dev$  та  $DEV\_WGH$  окрім зони нечутливості ( $dband$ ) мають настройку  $gain\_kp$ , яка вказує на коефіцієнт підсилення (послаблення) розузгодження, коли його значення знаходиться в цій зоні. При  $gain\_kp = 0$  блок нечутливості працює аналогічно як в  $PI\_B$ . Статичні характеристики блоку при інших значеннях  $gain\_kp$  показані на рис.5.5.

Ланка упередження (*Feed Forward*) дає можливість включити в контур управління сигнал упередження по збуренню, тим самим покращити динамічні характеристики процесу регулювання. Сигнал упередження масштабується відповідно до формули:

$$out\_ff = \frac{(FF - ff\_inf) \cdot (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf \quad (5.8)$$

Формули 5.3 та 5.4 наведені в спрощеному вигляді для розуміння суті роботи ПД регулятора. Функціональний блок  $PIDFF$  може працювати в режим ПД/ПІ або П/ПД регулятора, що впливає на особливості розрахунків.

Приклад виклику екземпляру  $PIDFF$  показаний на рис.5.6.

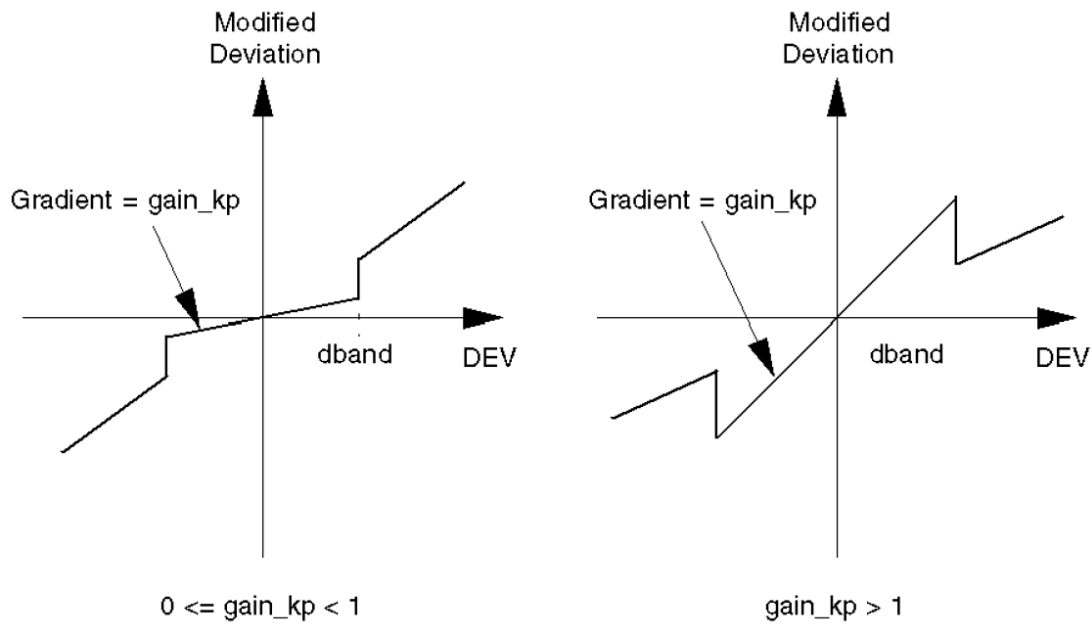


Рисунок 5.5– Налаштування зони нечутливості.

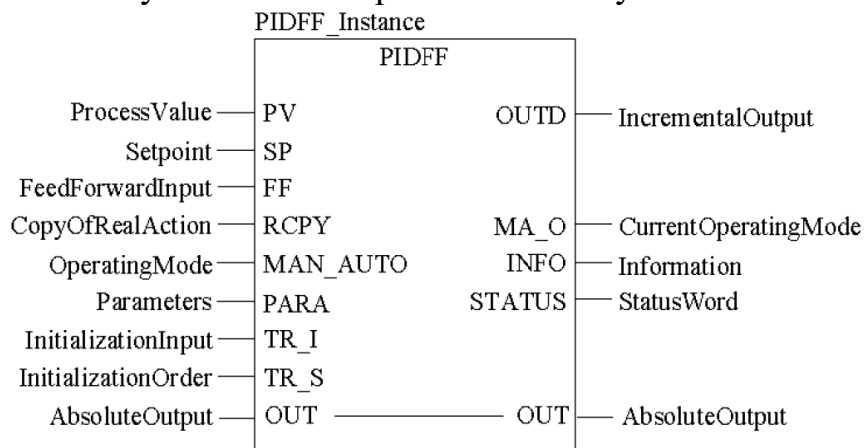


Рисунок 5.6– Приклад виклику екземпляру "PIDFF\_Instance" типу PIDFF в секції на мові FBD.

Таблиця 5.3 Параметри функціонального блока PIDFF.

Вхідні параметри		
<i>PV</i>	<i>REAL</i>	значення вимірювальної величини (плинне значення)
<i>SP</i>	<i>REAL</i>	задане значення (уставка)
<i>FF</i>	<i>REAL</i>	сигнал упередження
<i>RCPY</i>	<i>REAL</i>	копія значення регульованого виходу як база для інкрементального розрахунку (наприклад стан виконавчого механізму)
<i>MAN_AUTO</i>	<i>BOOL</i>	Режим роботи регулятора: 1 : Автоматичний режим 0 : Ручний режим
<i>PARA</i>	<i>Para_PIDFF</i>	Параметри регулятора (див. таб.6.6)
<i>TR_I</i>	<i>REAL</i>	Вхід слідкування/ініціалізації

<i>TR_S</i>	<i>BOOL</i>	Команда на включення режиму слідкування/ініціалізації (1: Включити вхід <i>TR_I</i> )
<b>Вхідні/вихідні параметри</b>		
<i>OUT</i>	<i>REAL</i>	абсолютне значення виходу регулятора
<b>Вихідні параметри</b>		
<i>OUTD</i>	<i>REAL</i>	інкрементальне значення виходу регулятора (див. <a href="#">параграф 6.2.5</a> )
<i>MA_O</i>	<i>BOOL</i>	Поточний режим виконання регулятора 1: Автоматичний режим 0: інший режим (ручний або режим слідкування)
<i>INFO</i>	<i>Info_PIDFF</i>	інформація про стан роботи <i>PIDFF</i> (див. таб.6.7)
<i>STATUS</i>	<i>WORD</i>	Слово статусу (використовується для контролю за помилками виконання <i>PIDFF</i> )

Таблиця 5.4. Опис структурного типу *Para\_PIDFF*.

<i>id</i>	<i>UINT</i>	Використовується для алгоритму автопідстройки ( <i>AUTOTUNING</i> )
<i>pv_inf</i>	<i>REAL</i>	обмеження по мінімуму вхідної величини завдання
<i>pv_sup</i>	<i>REAL</i>	обмеження по максимуму вхідної величини завдання
<i>out_inf</i>	<i>REAL</i>	обмеження по мінімуму вихідної величини блоку
<i>out_sup</i>	<i>REAL</i>	обмеження по максимуму вихідної величини блоку
<i>rev_dir</i>	<i>BOOL</i>	0: пряма робота регулятора ( <i>SP-PV</i> ) 1: зворотна робота регулятора ( <i>PV-SP</i> )
<i>mix_par</i>	<i>BOOL</i>	1: регулятор з паралельною структурою 0: регулятор зі змішаною структурою
<i>aw_type</i>	<i>BOOL</i>	1: активація безударного режиму антинасичення
<i>en_rcpy</i>	<i>BOOL</i>	1: використати вхід <i>RCPY</i>
<i>kp</i>	<i>REAL</i>	Коефіцієнт пропорційності
<i>ti</i>	<i>TIME</i>	Час інтегрування
<i>td</i>	<i>TIME</i>	Час диференціювання
<i>kd</i>	<i>REAL</i>	Коефіцієнт фільтрації диференційної складової
<i>kd</i>	<i>REAL</i>	Коефіцієнт фільтрації диференційної складової
<i>pv_dev</i>	<i>BOOL</i>	Режим розрахунку диференційної складової: 1: по розузгодженню ( <i>dev</i> ) 0: по плинному значенню ( <i>PV</i> )
<i>bump</i>	<i>BOOL</i>	1: відсутність безударності при переході в автоматичний режим 0: активація безударності переходу в автоматичний режим
<i>dband</i>	<i>REAL</i>	Зона нечутливості
<i>gain_kp</i>	<i>REAL</i>	коефіцієнт підсилення(ослаблення) розузгодження в зоні нечутливості
<i>ovs_att</i>	<i>REAL</i>	значення вагового коефіцієнта (від 0 до 1), що призначений для запобігання перерегулювання при зміні завдання
<i>outbias</i>	<i>REAL</i>	зміщення виходу регулятора в П/ПД-режимі функціонування

		(при $t_i=0s$ )
<i>out_min</i>	REAL	нижня межа для виходу регулятора
<i>out_max</i>	REAL	верхня межа для виходу регулятора
<i>outrate</i>	REAL	обмеження по швидкості зміни на виході одиниць/сек ( $\geq 0$ )
<i>ff_inf</i>	REAL	нижня межа значення <i>FF</i>
<i>ff_sup</i>	REAL	верхня межа значення <i>FF</i>
<i>otff_inf</i>	REAL	нижня межа значення <i>out_ff</i>
<i>otff_sup</i>	REAL	верхня межа значення <i>out_ff</i>

Таблиця 5.5. Опис структурного типу Info\_PIDFF .

<i>dev</i>	REAL	розузгодження ( <i>PV-SP</i> )
<i>out_ff</i>	REAL	значення упередження

### 5.4 Двохпозиційний регулятор STEP2

Функціональний блок *STEP2* реалізує 2-х позиційний закон регулювання з заданими порогами відхилення. Функціональна схема регулятора показана на рис.5.7, графік залежності виходу від входів – на рис.5.8, приклад виклику - на рис.5.9.

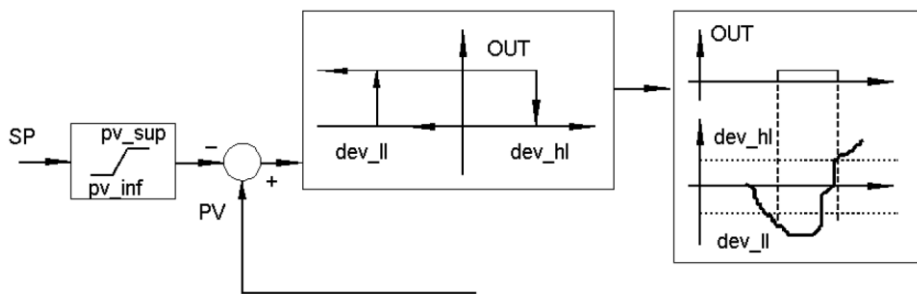


Рисунок 5.7– Функціональна схема 2-х позиційного регулятора *STEP2*

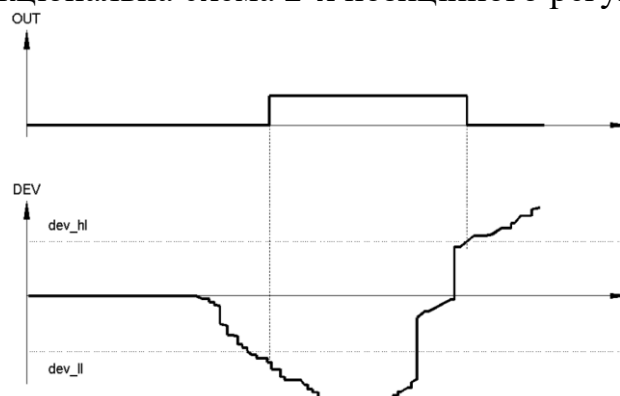


Рисунок 5.8– Графік залежності виходу *OUT* від розузгодження *DEV*

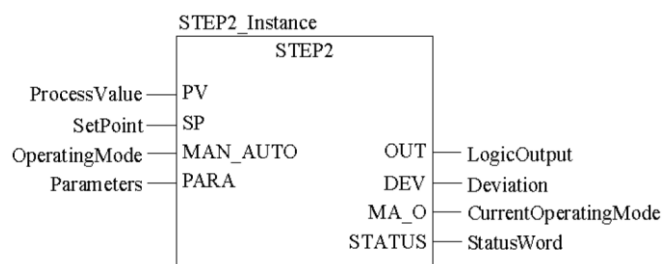


Рисунок 5.9– Приклад використання функціонального блоку *STEP2*

Таблиця 5.6 Параметри функціонального блока *STEP2*.

<b>Вхідні параметри</b>		
<i>PV</i>	<i>REAL</i>	значення вимірювальної величини (плинне значення)
<i>SP</i>	<i>REAL</i>	задане значення (уставка)
<i>MAN_AUTO</i>	<i>BOOL</i>	Режим роботи регулятора: 1 : Автоматичний режим 0 : Режим зупинки
<i>PARA</i>	<i>Para_STEP2</i>	Параметри регулятора (див. таб.6.9)
<b>Вихідні параметри</b>		
<i>OUT</i>	<i>BOOL</i>	значення виходу регулятора
<i>MA_O</i>	<i>BOOL</i>	Поточний режим виконання регулятора 1: Автоматичний режим 0: Режим зупинки
<i>DEV</i>	<i>REAL</i>	Значення розузгодження ( <i>PV - SP</i> )
<i>STATUS</i>	<i>WORD</i>	Слово статусу

Таблиця 5.7. Опис структурного типу *Para\_STEP2* .

<i>dev_ll</i>	<i>REAL</i>	нижній поріг відхилення від заданої величини ( $\geq 0$ )
<i>dev_hl</i>	<i>REAL</i>	верхній поріг відхилення від заданої величини ( $\leq 0$ )
<i>pv_inf</i>	<i>REAL</i>	обмеження по мінімуму вхідної величини завдання
<i>pv_sup</i>	<i>REAL</i>	обмеження по максимуму вхідної величини завдання

## 6. Обробка вхідних даних контурів регулювання.

### 6.1 Індикатор меж *INDLIM* (сімейство *Measurement*)

Функціональний блок *INDLIM* призначений для відслідковування переходу вхідного значення через визначені межі. На рис.6.1 показана діаграма роботи блоку, а на рис.6.2 – приклад виклику.

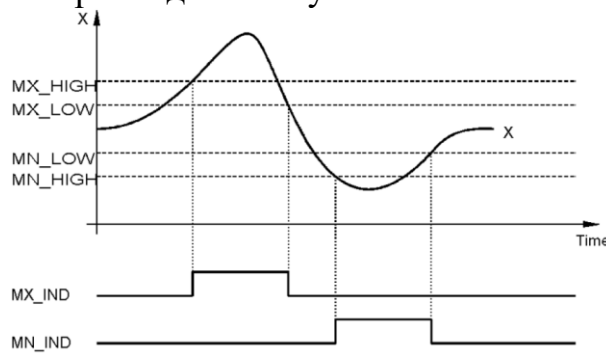


Рис.6.1. Робота блоку *INDLIM*

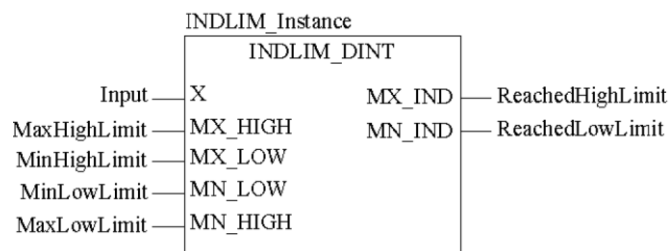


Рис.6.2. Приклад виклику блоку *INDLIM* .



Таблиця 6.1 Параметри функціонального блока *INDLIM*.

Вхідні параметри		
<i>X</i>	<i>INT, DINT, UINT, UDINT, REAL</i>	Вхідна змінна
<i>MX_HIGH</i>	<i>INT, DINT, UINT, UDINT, REAL</i>	Максимальна верхня межа
<i>MX_LOW</i>	<i>INT, DINT, UINT, UDINT, REAL</i>	Максимальна нижня межа
<i>MN_LOW</i>	<i>INT, DINT, UINT, UDINT, REAL</i>	Мінімальна нижня межа
<i>MN_HIGH</i>	<i>INT, DINT, UINT, UDINT, REAL</i>	Мінімальна верхня межа
Вихідні параметри		
<i>MX_IND</i>	<i>BOOL</i>	Відображення досягнення верхньої межі
<i>MN_IND</i>	<i>BOOL</i>	Відображення досягнення нижньої межі

## 6.2 Кусково-лінійна інтерполяція *LOOKUP\_TABLE1* (сімейство *Measurement*)

Процедура *LOOKUP\_TABLE1* використовується для кусочно-лінійної інтерполяції. У залежності від входу *X* та заданої вузловими точками залежності  $X_i Y_i$  (2 значення на кожену точку) формується вихід *Y*. Кількість вузлових точок варіюється до 15, кожна задається парою значень  $X_i Y_i$ : перша для *X* (непарні номери *j*), друга для *Y* (парні номери *j*). Таким чином процедура може мати до  $30+1$  входів.

Аналitична залежність *X* від *Y* та заданих вузлових точок показана в (6.1)

$$Y = \begin{cases} Y_1, \text{при } X < X_1 \\ Y_i + \frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} \cdot (X - X_i), \text{при } X_1 \leq X \leq X_m \\ Y_m, \text{при } X > X_m \end{cases} \quad (6.1)$$

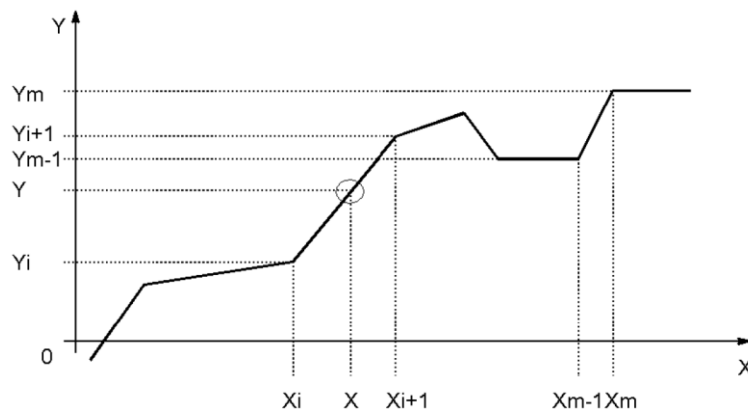


Рисунок 6.2– Робота процедури *LOOKUP\_TABLE1*

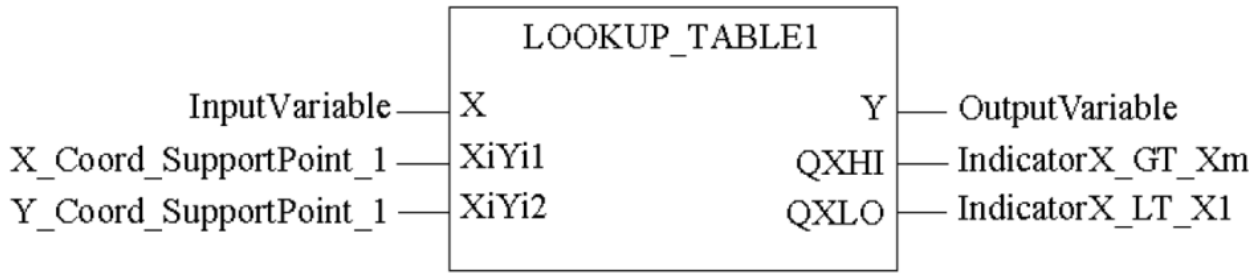


Рисунок 6.3– Робота функції *LOOKUP\_TABLE1*

Таблиця 6.1 Параметри процедури *LOOKUP\_TABLE1*.

Вхідні параметри		
<i>X</i>	<i>REAL</i>	Вхідна змінна
<i>XiYi1</i>	<i>REAL</i>	координата $X_1$ .
<i>XiYi2</i>	<i>REAL</i>	координата $Y_1$ .
<i>XiYi(n-1)</i>	<i>REAL</i>	координата $X_{n/2}$ ; $n=\max 30$
<i>XiYi(n)</i>	<i>REAL</i>	координата $Y_{n/2}$ ; $n=\max 30$
Вихідні параметри		
<i>Y</i>	<i>REAL</i>	Вихідна змінна
<i>QXHI</i>	<i>BOOL</i>	Індикація $X > X_m$
<i>QXLO</i>	<i>BOOL</i>	Індикація $X < X_l$

### 6.3. Фільтр ковзного середнього *AVGMV* (сімейство *Measurement*)

Функціональний блок *AVGMV* реалізує алгоритм розрахунку фільтру ковзного середнього по формулі (6.2).

$$Y = \frac{\sum_{i=0}^{N-1} X_i}{N} \quad \text{або} \quad Y = Y_{old} + \frac{X - X_{N-1}}{N} \quad (6.2)$$

де  $N$  – кількість значень в буфері (ширина вікна),  $X$  – вхідне не фільтроване значення,  $Y$  – середнє вихідне (фільтроване) значення,  $Y_{old}$  - значення виходу на попередньому виклику.

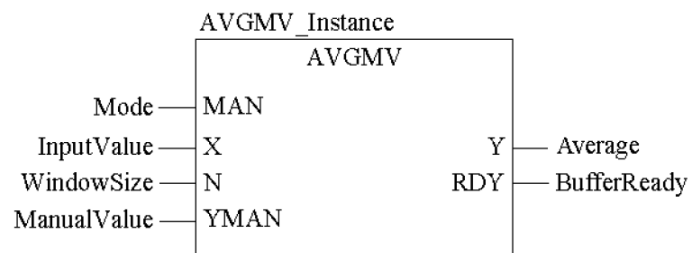


Рисунок 6.4. Виклик функціонального блоку *AVGMV*

Вхідні параметри		
<i>MAN</i>	<i>BOOL</i>	0 = Автоматичний режим 1 = Ручний режим
<i>X</i>	<i>REAL</i>	Вхід
<i>N</i>	<i>INT</i>	Ширина вікна (кількість значень в буфері; максимум 50)

<i>YMAN</i>	<i>REAL</i>	Ручне значення
<b>Вихідні параметри</b>		
<i>Y</i>	<i>REAL</i>	Середнє значення
<i>RDY</i>	<i>BOOL</i>	1 = буфер заповнений (готовий) 0 = буфер не заповнений (не готовий)

При ініціалізації функціонального блоку буфер скидується. З кожним викликом буфер наповнюється новими значеннями. Поки кількість значень в буфері не буде рівним  $N$ , буфер не буде готовий ( $RDY=FALSE$ ) і фільтр працює з неповним буфером. Коли буфер заповниться ( $RDY=TRUE$ ) блок буде видавати коректне відфільтроване значення.

#### 6.4 Розрахунок витрати по перепаду тиску $K\_SQRT$ (сімейство *Mathematics*) та $MFLOW$ (сімейство *Conditioning*)

Об'ємна витрата може бути розрахована за наступними залежностями:

$$Q = 0.01252 \cdot \alpha \cdot \varepsilon \cdot d^2 \sqrt{\frac{p_1 - p_2}{\rho}} \quad (6.3)$$

де  $Q$ -об'ємна витрата (м3/год),  $d$  – діаметр отвору (мм),  $\rho$  -густина (кг/м3),  $(p_1-p_2)$  – перепад тиску на звужуючому пристрої (кгс/м2),  $\varepsilon$ – коефіцієнт розширення (для рідин=1),  $\alpha$ - коефіцієнт витрати який розраховується за формулою:

$$\alpha = \frac{\mu \cdot \sqrt{\psi}}{\sqrt{\xi + \kappa_2 - \kappa_1 \cdot m^2 \cdot \mu^2}} \quad (6.4)$$

Коефіцієнти розраховуються (підбираються) в залежності від середовища та типу звужуючого пристрою. Таким чином формулу для розрахунку можна записати:

$$Q = K \cdot \sqrt{dP} \quad (6.5)$$

де  $dP=p_1-p_2$ ,  $K$  – ваговий коефіцієнт. Ваговий коефіцієнт  $K$  можна розрахувати або підібрати при відомих витратах на етапі налагодження.

У UNITY PRO для реалізації розрахункової формули 6.5 можна скористатися функцією  $SQRT$  (вилучення квадратного кореню), однак для цієї цілі зручніше використовувати спеціалізовану функцію  $K\_SQRT$  або функціональний блок  $MFLOW$ .

Функція  $K\_SQRT$  знаходиться в бібліотеці *Control Library* сімейства *Mathematics*. Приклад використання функції показаний на рис.6.5. Функція повертає:

$$OUT = \begin{cases} K \cdot \sqrt{IN}, \text{ при } IN \geq CUTOFF \\ 0, \text{ при } IN < 0 \text{ або } IN < CUTOFF \end{cases} \quad (6.6)$$

Таким чином, крім вагового коефіцієнта  $K$ , функція враховує нижню межу вхідного значення  $CUTOFF$ . Тобто, якщо перепад тиску на вході є від'ємним або менше нижньої межі, то функція повертає 0, в той час як  $SQRT$  повернув би -1.#NAN, а це в свою чергу спрощує використання цієї функції для даної задачі.

Густина газів залежить від температури та тиску. Для врахування цих параметрів, треба їх ввести в розрахункову формулу для масової витрати:

$$Q_m = K \cdot \sqrt{dP \cdot \frac{P}{T}} \quad (6.7)$$

Функціональний блок *MFLOW* сімейства *Conditioning*, призначений для розрахунку масової витрати газів з урахуванням їх температури та тиску згідно формули (6.8):

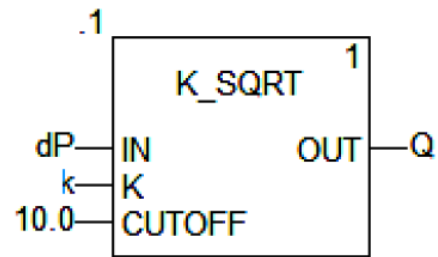


Рисунок 6.5. Приклад використання функції *K\_SQRT*

$$OUT = K \cdot \sqrt{\frac{IN \cdot PA}{TA}} \quad (6.8)$$

де *PA*-абсолютний тиск газу, *TA* – абсолютна температура в градусах Кельвіна.

Приклад використання функціонального блоку *MFLOW* показаний на рис.6.6. Вхід *IN*, вихід *OUT* та коефіцієнт *K* мають той же зміст, що і в попередньому варіанті. Параметри роботи блоку задаються на вході *PARAM*.

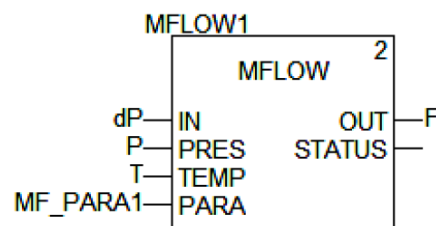
Вхід *PRES* – це тиск, який може задаватися як в абсолютних одиницях так і відносно атмосферного. У випадку відносного тиску (*PARAM.pr\_pa=FALSE*), абсолютний тиск розраховується за формулою :

$$PA = PRES + p_i \quad (6.9)$$

де *p<sub>i</sub>* задається в тій же структурі *PARAM*.

Вхід *TEMP*, це температура, яка може задаватися в градусах Цельсія (*PARAM.tc\_tf=FALSE*) або Фаренгейта (*PARAM.tc\_tf=TRUE*). У блоці розраховується абсолютна температура *TA* в градусах Кельвіна.

Корекція по тиску активується параметром *PARAM.en\_pres=TRUE*, по температурі - *PARAM.en\_temp=TRUE*. Якщо *PARAM.en\_sqrt=FALSE* то добуток буде розраховуватись без квадратного кореня.



MF_PARA1	Para_MFLOW	
● k	REAL	ваговий коефіцієнт
● en_pres	BOOL	1 - активація корекції по тиску
● pr_pa	BOOL	0 - відносний тиск, 1 - абсолютний тиск
● pu	REAL	величина =1 атмосфері
● en_temp	BOOL	1 - активація корекції по температурі
● tc_tf	BOOL	0 - розрахунок в Цельсіях, 1 - розрахунок в Фаренгейтах
● en_sqrt	BOOL	1 - активація квадратного кореня

Рис.6.6. Приклад використання функціонального блоку *MFLOW*.

## 7 Обробка вихідних даних контурів регулювання (сімейство *Output Processing*)

### 7.1 Блок управління реверсивним двигуном (*SERVO*)

Функціональний блок *SERVO* призначений для реалізації управляючих дій з регуляторів *PI\_B/PIDFF* або інших блоків з використанням виконавчих механізмів типу реверсивного двигуна (серводвигун, наприклад *MEO*). Для управління серводвигунами використовуються два виходи - *RAISE* ("більше") та *LOWER* ("менше"), на яких сигнал формується в залежності від значення входу *IN* (або *INPD*) та налаштування блоку (рис.7.1). У таблиці 7.1 показані параметри блоку *SERVO*.

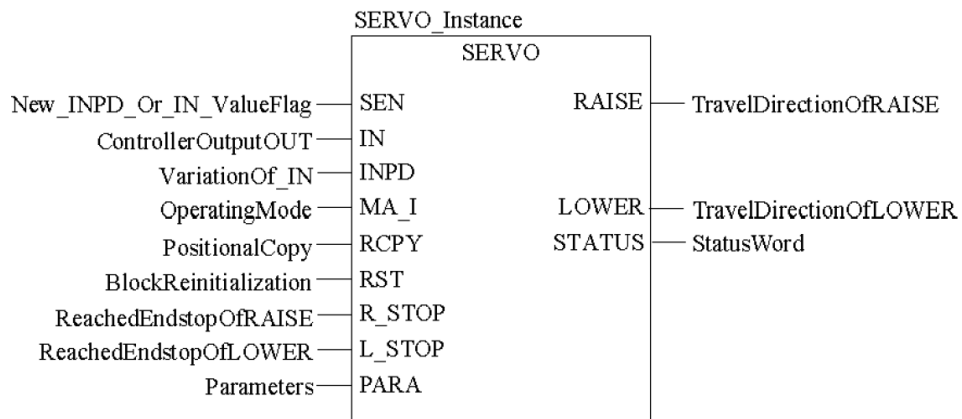


Рис.7.1. Виклик функціонального блоку *SERVO*.

Таблиця 7.1 Параметри блоку *SERVO*

<b>Вхідні параметри</b>		
<i>SEN</i>	<i>BOOL</i>	1 : є нові значення на входах <i>INPD</i> або <i>IN</i> 0 : немає нових значень на входах <i>INPD</i> або <i>IN</i>
<i>IN</i>	<i>REAL</i>	значення з абсолютного виходу блоку управління <i>OUT</i> (0 до 100%)
<i>INPD</i>	<i>REAL</i>	значення з інкрементального виходу блоку управління <i>OUTD</i> (-100% до 100%)
<i>MA_I</i>	<i>BOOL</i>	значення операційного режиму блоку регулятора до якого підключений <i>SERVO</i> (вихід <i>MA_O</i> ) 1 : Автоматичний режим 0 : інший режим (ручний або слідкування)
<i>RCPY</i>	<i>REAL</i>	Зворотній зв'язок по позиції ВМ (0 до 100%)
<i>RST</i>	<i>BOOL</i>	1 : Ініціалізація функціонального блоку (відключення виходів та обнуління внутрішнього статусу блоку)
<i>R_STOP</i>	<i>BOOL</i>	кінцева позиція "відкритий" (досягнутий поріг по <i>RAISE</i> )
<i>L_STOP</i>	<i>BOOL</i>	кінцева позиція "закритий" (досягнутий поріг по <i>LOWER</i> )
<i>PARA</i>	<i>Para_SERVO</i>	Параметри (див. таб 6.24)

Вихідні параметри		
<i>RAISE</i>	<i>BOOL</i>	Вихід "більше" ("відкрити")
<i>LOWER</i>	<i>BOOL</i>	Вихід "менше" ("закрити")
<i>STATUS</i>	<i>WORD</i>	Слово статусу

Таблиця 7.2 Опис структурного типу *Para\_SERVO*

<i>en_rcpy</i>	<i>BOOL</i>	1 : Використовується зворотній зв'язок по позиції ВМ (включити <i>RCPY</i> )
<i>rcpy_rev</i>	<i>BOOL</i>	1 : Інвертувати <i>RCPY</i> (0 – це 100%, 100 – це 0%) 0 : Не інвертувати <i>RCPY</i>
<i>t_motor</i>	<i>TIME</i>	час повного відкриття регулюючого органу
<i>t_mini</i>	<i>TIME</i>	мінімальна довжина імпульсу

Функціональний блок може використовуватись як з сигналом зворотного зв'язку по положенню виконавчого механізму (*RCPY*) так і без нього.

При використанні сигналу зворотного зв'язку ( $en\_rcpy = TRUE$ ), в якості управляючого входу *SERVO* треба використовувати вхід *IN*, який повинен бути зв'язаний з абсолютним виходом *OUT* регулятора *PI\_B/PIDFF*. Для кожного нового значення виходу *OUT*, який формує регулятор, функціональний блок *SERVO* генерує дискретний вихід *RAISE* або *LOWER* з довжиною імпульсу пропорційною різниці *IN-RCPY*. При необхідності, вхід *RCPY* можна інвертувати ( $rcpy\_rev=1$ ), тобто коли показник положення регулюючого органу показує не процент відкриття, а процент закриття. Приклад діаграми роботи *SERVO* зі зворотним зв'язком по положенню ВМ показаний на рис. 7.2.

Якщо зворотний сигнал по положенню ВМ не використовується ( $en\_rcpy = FALSE$ ) то в якості управляючого входу *SERVO*, треба використовувати вхід *INPD*, який повинен бути зв'язаний з виходом *OUTD* регулятора *PI\_B/PIDFF*. Для кожного нового значення виходу *OUTD* регулятора, блок *SERVO* генерує дискретний вихід *RAISE* або *LOWER* з довжиною імпульсу пропорційною *INPD*. Для правильного функціонування операційного режиму, вхід *MA\_I* блоку *SERVO* повинен бути з'єднаний з виходом *MA\_O* блоку регулятора.

Для формування імпульсу з тривалістю, пропорційною величині управляючого сигналу, необхідно вказати час повного відкриття регулюючого органу ( $t\_motor$ ). Тобто, наприклад, при формуванні на вході *INPD* значення 100%, блок *SERVO* сформує імпульс на виході *RAISE* рівним значенню  $t\_motor$ . А при *INPD* рівним -10%, *SERVO* сформує імпульс на виході *LOWER* рівним значенню  $t\_motor/10$ .

Для зменшення навантаження на двигун, тобто збільшення його терміну служби, задається мінімальний час імпульсу ( $t\_mini$ ). Якщо тривалість розрахованого імпульсу менша ніж  $t\_mini$ , імпульс формуватися не буде, однак він буде врахований на наступних циклах.

Для точності регулювання, блок *SERVO* повинен викликатися з кожним циклом Задачі. Однак весь контур регулювання може оброблятися з меншою частотою, наприклад при використанні *SAMPLETM*. Для того щоб вказати блоку *SERVO* що регулятор *PI\_B/PIDFF* оброблений і сформовані нові значення

(особливо це стосується виходу *OUTD* регулятора), і його вихід *OUT/OUTD* а отже і вхід *SERVO IN/INPD* отримав нові значення, на вхід *SEN* подається *TRUE*. Таким чином, якщо в контурі для задавання періодичності обробки використовується блок *SAMPLETM*, то його вихід підключається паралельно до входів *EN* блоків *PI\_B/PIDFF* і до входу *SEN* блоку *SERVO*.

У ручному режимі (*MA\_I=FALSE*) блок *SERVO* обробляє входи *IN/INPD* в кожному циклі незалежно від значення входу *SEN*, що треба враховувати при написанні програми. Тобто при ручній зміні виходу *OUT* регулятора, він автоматично розрахує *OUTD*, який протримається на виході аж до наступного виклику регулятора. За цей час, зв'язаний з цим виходом вхід *INPD* блоку *SERVO*, буде оброблений декілька (*n*) раз, що затягне імпульс в *n* раз довше, ніж потрібно. Для подолання цієї проблеми, в ручному режимі після кожного виклику регулятора необхідно обнулювати змінну, прив'язану до *OUTD* та *INPD*.

При використанні сигналу зворотного зв'язку (*en\_rcpy = TRUE*) у ручному режимі блок *SERVO* буде видавати команди "більше" та "менше" на виконавчий механізм доти, поки входи *RCPY* та *IN* не будуть рівними. Це може негативно сказатися на процесі а також на роботі приводу та регулюючого органу. При досягненні крайнього положення регулюючим органом, тобто коли *R\_STOP=TRUE* або *L\_STOP=TRUE*, відповідний вихід виставляється в логічний нуль (*RAISE=FALSE* або *LOWER=FALSE*) незалежно від значення *IN* або *INPD*.

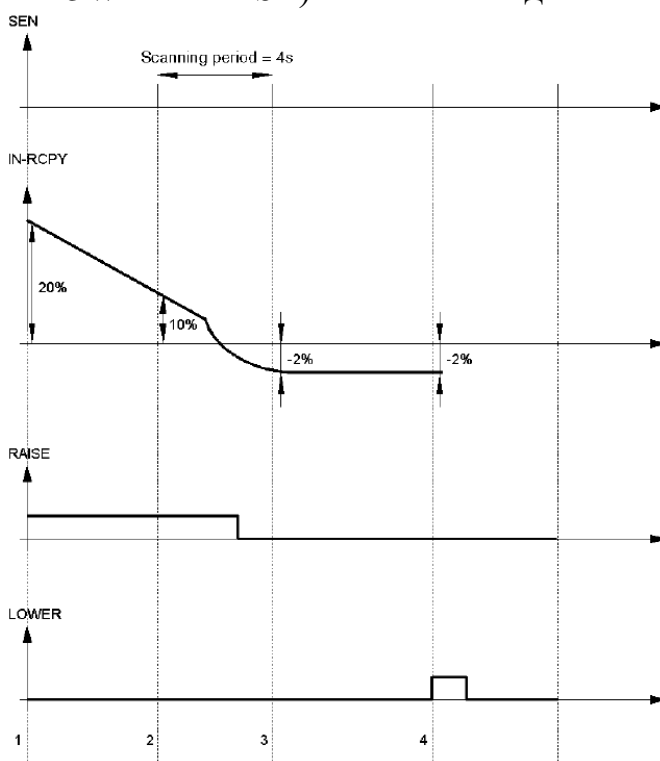


Рисунок 7.2– Приклад діаграми роботи блоку *SERVO* зі зворотним зв'язком по положенню *BM*.

Розглянемо роботу *SERVO* в режимі використання зворотного зв'язку по положенню, тобто при *en\_rcpy=TRUE* (рис.7.2). У цьому прикладі використовуються такі настройки:  $t_{motor}=t\#25s$ ,  $t_{mini}=t\#1s$ , періодичність виклику основного регулятора (періодичність активації *SEN*) - 4s. Контур працює

в автоматичному режимі ( $MA\_I=TRUE$ ). На діаграмі цифрами мітками позначені наступні ситуації.

1. При початковому скануванні контуру значення ( $IN-RCPY$ )= $+20\%$  : на виході  $RAISE$  генерується імпульс тривалістю  $5s$  ( $=20\%$  від  $25s$ );

2. При наступному скануванні контур значення ( $IN-RCPY$ )= $+10\%$ : на виході  $RAISE=TRUE$  залишається ще протягом  $2.5s$  ( $=10\%$  від  $25s$ ); одна секунда, яка залишилась від попереднього разу вже не враховується;

3. На 3-му циклі сканування контуру ( $IN-RCPY$ )= $-2\%$ : це відповідає імпульсу  $0.5s$  на виході  $LOWER$ , однак оскільки це менше ніж  $t\_mini$ , то вихід  $LOWER=FALSE$ ; тривалість  $0.5s$  для  $LOWER$  залишається збереженим до наступного циклу перерахунку;

На 4-му циклі сканування контуру ( $IN-RCPY$ )= $-2\%$ : це відповідає імпульсу  $0.5s$  на виході  $LOWER$ , враховуючи попередній імпульс  $0.5s$ , загальна тривалість імпульсу  $1s$ , тобто на виході  $LOWER=TRUE$  протягом однієї секунди.

Розглянемо роботу  $SERVO$  без використання зворотного зв'язку по положенню, тобто при  $en\_rcpy=FALSE$  (рис.7.3). У цьому прикладі використовуються такі настройки:  $t\_motor=t\#25s$ ,  $t\_mini=t\#1s$ , контур працює в ручному режимі ( $MA\_I=FALSE$ ). На діаграмі цифрами мітками позначені наступні ситуації.

1. При  $INPD=+20\%$ : на виході  $RAISE$  генерується імпульс тривалістю  $5s$  ( $=20\%$  від  $25s$ );

2. При появі значення  $INPD=+2\%$ : це відповідає імпульсу  $0.5s$  на виході  $RAISE$ , однак оскільки це менше ніж  $t\_mini$ , то вихід  $RAISE=FALSE$ ; тривалість  $0.5s$  для  $RAISE$  залишається збереженим до наступного перерахунку;

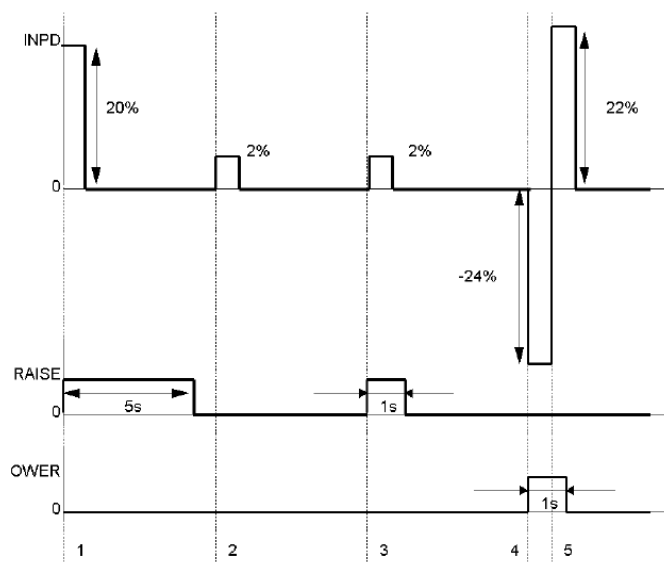


Рисунок 7.3— Приклад діаграми роботи блоку  $SERVO$  без зворотного зв'язку по положенню ВМ.

3. При появі значення  $INPD=+2\%$ : це відповідає імпульсу  $0.5s$  на виході  $RAISE$ , враховуючи попередній імпульс  $0.5s$ , загальна тривалість імпульсу  $1s$ , тобто на виході  $RAISE=TRUE$  протягом однієї секунди;



4. При появі значення  $INPD = -24\%$ : це відповідає імпульсу  $bs$  на виході  $LOWER$ ;

5. Перед закінченням наступної секунди наступна модифікація  $INPD = +22\%$ : загальна модифікація  $= -24\% + 22\% = -2\%$ , це менше ніж мінімальний імпульс (4%), тому вихід  $LOWER = FALSE$ .

## 7.2 Управління виконавчими механізмами з використанням широтно- імпульсного перетворення $PWM1$

Функціональний блок  $PWM1$  призначений для перетворення числового значення на вході блоку  $IN$  в 2 дискретні сигнали  $OUT\_POS$  та  $OUT\_NEG$  з певною шириною та частотою імпульсу (рис.7.4).

*Ініціалізація блоку обов'язково повинна проводитись при першому циклі ПЛК!*

Весь час ділиться на періоди з тривалістю  $t\_period$  (рис.7.5). У залежності від значення входу  $IN$ , на кожному періоді формується сигнал  $OUT\_POS$  (при  $IN > 0$ ) або  $OUT\_MIN$  (при  $OUT < 0$ ) з тривалістю  $T\_on$ . Залежність  $T\_on$  від значення  $IN$  описується формулами (7.1)-(7.2), та графічно показана на рис.7.6.

$$T\_on(OUT\_POS) = t\_period \cdot \frac{IN}{in\_max} \quad \text{при } 0 \leq IN \leq in\_max$$

$$T\_on(OUT\_NEG) = t\_period \cdot \frac{|IN|}{in\_max} \quad \text{при } 0 \leq -IN \leq in\_max$$

(7.1), (7.2)

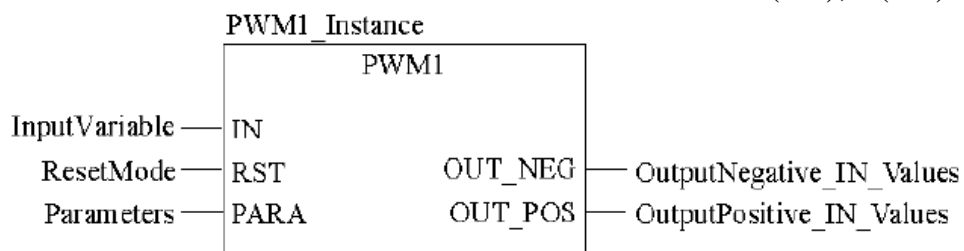


Рисунок 7.4– Приклад виклику блоку  $PWM1$ .

У таблиці 7.3 та 7.4 наведені параметри блоку  $PWM1$ .

Таблиця 7.3 Параметри блоку  $PWM1$

Вхідні параметри		
$IN$	$REAL$	Вхідна змінна
$RST$	$BOOL$	1 = Скинути блок
$PARA$	$Para\_PWM1$	Параметри
Вихідні параметри		
$OUT\_NEG$	$BOOL$	Вихід для від'ємного значення $IN$
$OUT\_POS$	$BOOL$	Вихід для додатного значення $IN$

Таблиця 7.4 Опис структурного типу  $Para\_PWM1$

$t\_period$	$TIME$	Довжина (тривалість) періоду
$t\_min$	$TIME$	Мінімальний час імпульсу
$in\_max$	$REAL$	обмеження по максимуму (по модулю) значення $IN$

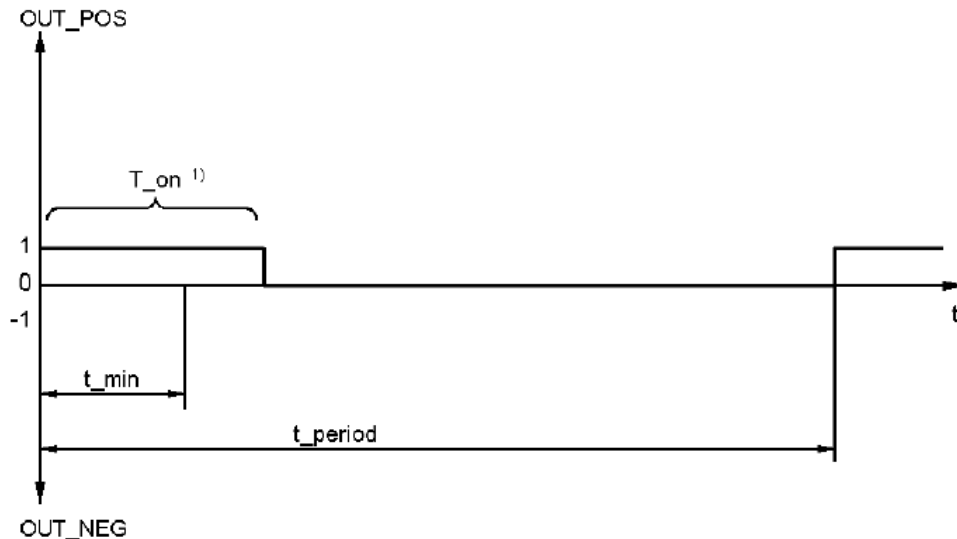


Рисунок 7.5– Діаграма зміни виходів блоку *PWM1*.

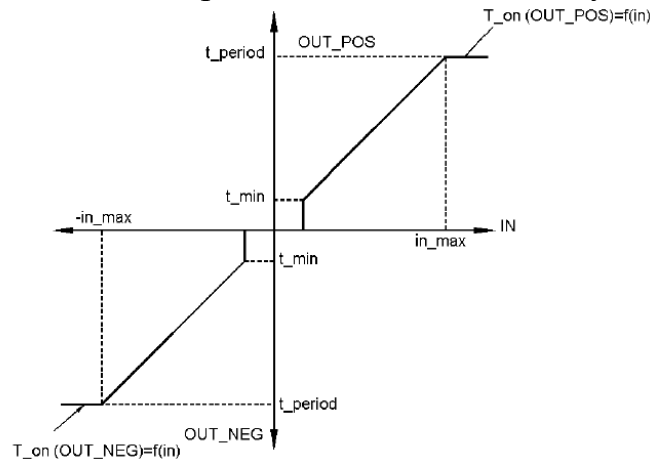


Рисунок 7.6– Статична характеристика  $t\_period(IN)$  для *PWM1*.

Параметр  $t\_min$  вказує мінімальну тривалість імпульсу. Тобто якщо розрахований час  $T\_on$  за (7.1) та (7.2) буде менше часу  $t\_min$ , то імпульс на даному періоді формуватися не буде.

При команді  $RST=1$  всі виходи блоку скидаються в *FALSE*, а блок починає відраховувати час з моменту, коли  $RST=0$ .

Якщо блок *PWM1* обробляє вихід регуляторів *PI\_B/PIDFF*, то рекомендується  $t\_period$  виставляти рівним часу виклику блоку регулятора. Це гарантує, що кожний новий сигнал на ВМ з регулятора буде оброблений блоком *PWM1*. Час періоду виклику самого *PWM1* рекомендується виставляти як мінімум в 10 раз менше ніж час періоду  $t\_period$ .

Блок *PWM1* може використовуватися для управління двигуном, швидкість якого керується шириною імпульсу *OUT\_POS* в один бік, і *OUT\_NEG* в інший бік. Іншим прикладом виконавчого механізму є *TEH* (трубчатий електронагрівач), однак при цьому буде використовуватись тільки вихід *OUT\_POS*.

Приклад діаграми роботи *PWM1* з настройками  $t\_period=t\#4s$ ,  $t\_min=t\#0.5s$ ,  $in\_max=10$  показаний на рис.7.7.

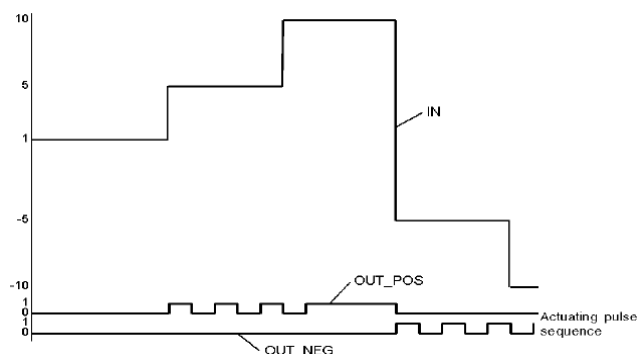


Рисунок 7.7– Приклад діаграми роботи PWM1.

### 4.3.3. Блок ручного управління MS

Функціональні блоки регуляторів, в тому числі  $PI\_B/PIDFF$ , мають вбудовані засоби переключення їх в ручний режим, з забезпеченням безперебійності. Нижче наведені випадки, для яких може знадобитися використання блоку ручного управління MS:

- для управління аналоговими виходами, які управляються НЕ через алгоритми зі зворотним зв'язком (не за відхиленням);
- для управління аналоговими виходами, які в ручному і в автоматичному режимах формуються окремими блоками (наприклад при каскадному регулюванні, переключенні між алгоритмами);
- для управління серводвигунами в ручному режимі без використання регуляторів  $PI\_B/PIDFF$ ;

На рис.7.8 показана функціональна схема блоку MS, на рис.7.9 – приклад виклику блоку в FBD.

Вихід  $OUT$  в автоматичному режимі формується як сума входів  $IN$  та  $outbias$ , якщо зміщення активоване ( $use\_bias=TRUE$ ). У ручному режимі вихід міняється безпосередньо. В обох режимах абсолютне значення  $OUT$  обмежується по мінімуму і максимуму, а інкрементальне  $OUTD$  – не обмежується.

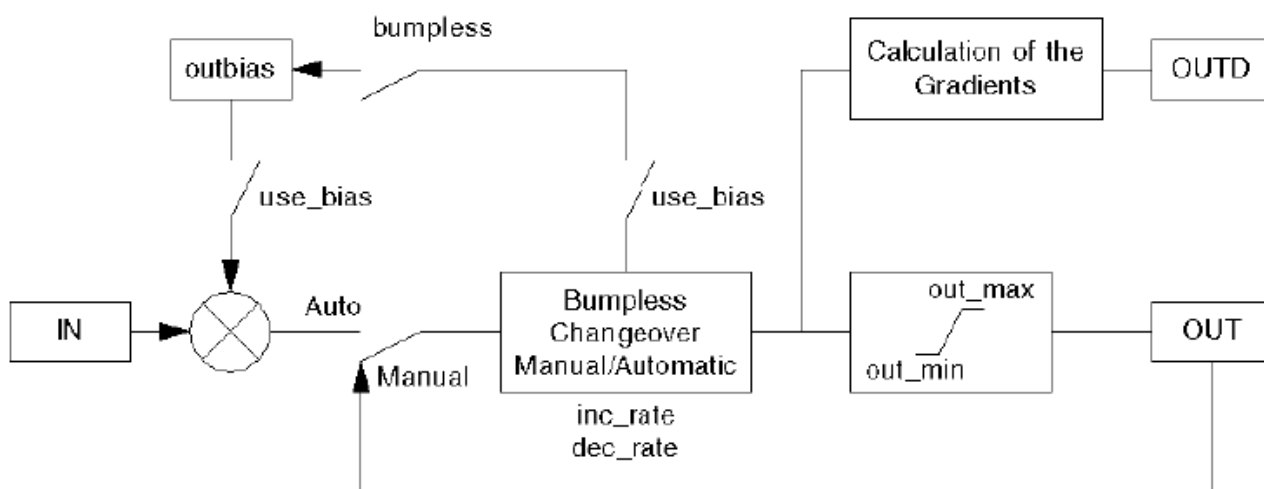


Рис.7.8.Функціональна схема блоку MS.

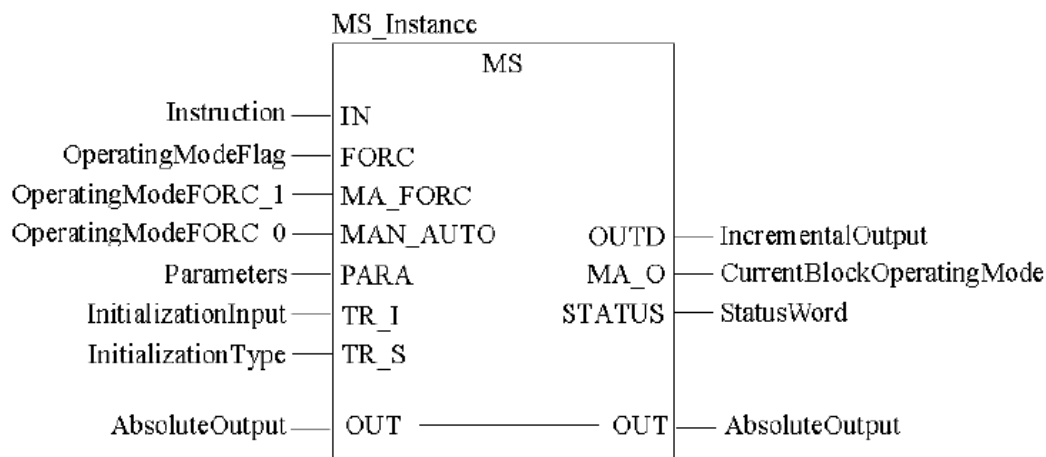


Рисунок 7.9—Приклад виклику блоку *MS*

Таблиця 7.5 Параметри блоку *MS*

<b>Вхідні параметри</b>		
<i>IN</i>	<i>REAL</i>	Значення змінної в автоматичному режимі
<i>FORC</i>	<i>BOOL</i>	1: Режим ручний/автомат визначається входом <i>MA_FORC</i> 0: Режим ручний/автомат визначається входом <i>MAN_AUTO</i>
<i>MA_FORC</i>	<i>BOOL</i>	Режим ручний/автомат для <i>FORC = 1</i> 1: Автоматичний режим 0: Ручний режим
<i>MAN_AUTO</i>	<i>BOOL</i>	Режим ручний/автомат для <i>FORC = 0</i> 1: Автоматичний режим 0: Ручний режим
<i>PARA</i>	<i>Para_MS</i>	Параметр
<i>TR_I</i>	<i>REAL</i>	Вхід ініціалізації
<i>TR_S</i>	<i>BOOL</i>	Команда на включення ініціалізації (1: Включити вхід ініціалізації)
<b>Вхідні/вихідні параметри</b>		
<i>OUT</i>	<i>REAL</i>	абсолютне значення виходу
<b>Вихідні параметри</b>		
<i>OUTD</i>	<i>REAL</i>	інкрементальне значення виходу: різниця між вихідною величиною в плинному і попередньому циклах перерахунку <i>OUTD</i>
<i>MA_O</i>	<i>BOOL</i>	Плинний режим роботи регулятора (0: Ручний, 1: Автоматичний)
<i>STATUS</i>	<i>WORD</i>	Слово статусу

Таблиця 7.6 Опис структурного типу Para\_MS

<i>out_min</i>	REAL	нижня межа для виходу регулятора
<i>out_max</i>	REAL	верхня межа для виходу регулятора
<i>inc_rate</i>	REAL	Швидкість наростання значення сигналу при переключенні руч./авт. (одиниць на секунду)
<i>dec_rate</i>	REAL	Швидкість спадання значення сигналу при переключенні руч./авт. (одиниць на секунду)
<i>outbias</i>	REAL	значення зміщення
<i>use_bias</i>	BOOL	1: Активувати зміщення
<i>bumpless</i>	BOOL	1: активувати безударність переходу руч/авт

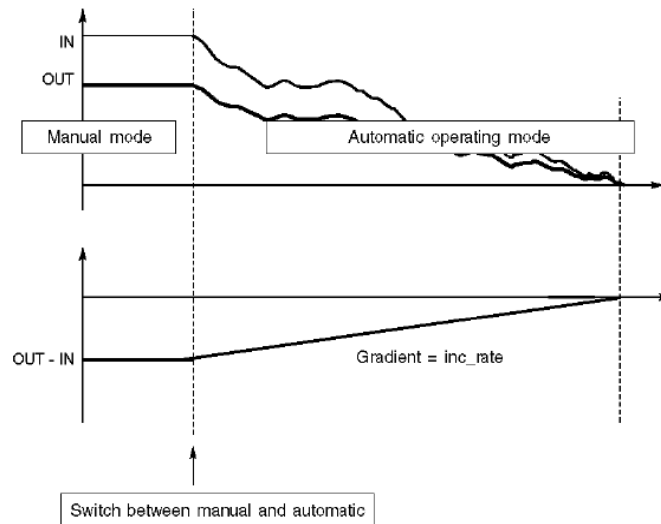


Рисунок 7.10–Діаграма роботи безударного переходу руч./авт. в MS.

Для реалізації безударності переходу з ручного в автоматичний режим (*bumpless=TRUE*) використовуються лінійні переходи з обмеженням по швидкості, які настраюються параметрами *inc\_rate* та *dec\_rate*. Таким чином, в момент переходу в автоматичний режим, різниця між IN та OUT буде зменшуватися відповідно до заданої швидкості (рис.7.10).

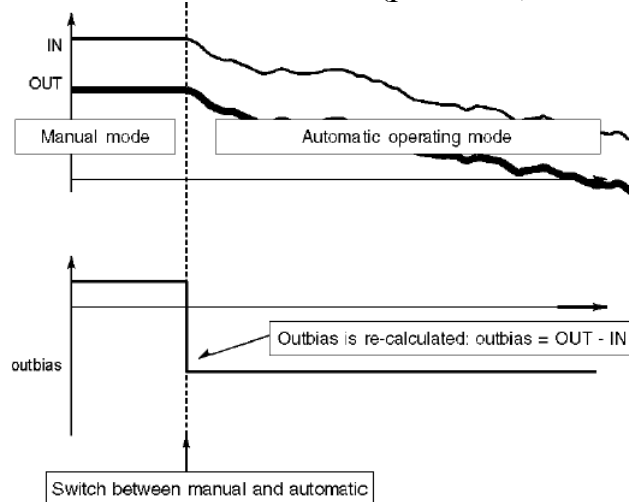


Рисунок 7.11. Діаграма роботи безударного переходу руч./авт. зі зміщенням в MS.

Якщо активований параметр *outbias* (зміщення) тобто *use\_bias=TRUE*, то безударність досягається за рахунок використання нового перерахованого значення зміщення (рис.7.11).

Безударний перехід є сенс активувати тоді, коли вхід *IN* підключений до регуляторів або інших блоків, в яких не реалізований безударний перехід.

## Організація управління уставками та блоки додаткової обробки

### 8. Управління уставками.

#### 8.1 Перемикач уставок *SP\_SEL*

Функціональний блок призначений для переключення уставки (заданого значення *SP*) для регуляторів *PI\_B/PIDFF* або аналогічних (рис.8.1). У локальному режимі (*local mode*, коли *SP\_RSP=FALSE*) уставка змінюється безпосередньо через вхід/вихід *SP*, який повинен бути підключений до однойменного входу регулятора. У дистанційному режимі (*remote mode*, коли *SP\_RSP=TRUE*) уставка змінюється через вхід *RSP*.

Функціональний блок має наступні властивості:

- 1) переключення між уставками в безударному режимі;
- 2) операції з уставками проводяться навіть в ручному режимі;
- 3) проводиться обмеження уставки по верхній і нижній межі;

При переключенні уставки з локального режиму в дистанційний, безударність забезпечується за рахунок плавної зміни виходу з швидкістю, заданою параметром *rate*.

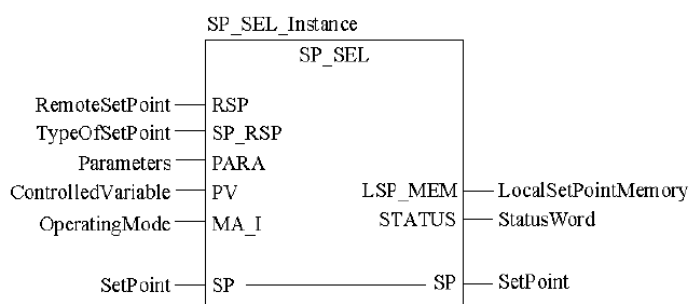


Рисунок 8.1– Приклад виклику блоку *SP\_SEL*.

Таблиця 8.1 Параметри блоку *SP\_SEL*

Вхідні параметри		
<i>RSP</i>	<i>REAL</i>	Дистанційна уставка
<i>SP_RSP</i>	<i>BOOL</i>	Вибір уставки 1 : Дистанційна уставка 0 : Локальна уставка
<i>PARA</i>	<i>Para_SP_SEL</i>	Параметри
<i>PV</i>	<i>REAL</i>	Плинне значення змінної управління
<i>MA_I</i>	<i>BOOL</i>	Операційний режим регулятора, до якого під'єднаний перемикач 1 : Автоматичний режим 0 : Ручний режим
Вхідні/вихідні параметри		
<i>SP</i>	<i>REAL</i>	Уставка регулятора
Вихідні параметри		
<i>LSP_MEM</i>	<i>REAL</i>	Локальна уставка в пам'яті
<i>STATUS</i>	<i>WORD</i>	Слово статусу

Таблиця 8.2 Опис структурного типу *Para\_SP\_SEL*

<i>sp_min</i>	<i>REAL</i>	Обмеження по мінімуму уставки
<i>sp_max</i>	<i>REAL</i>	Обмеження по максимуму уставки
<i>bump</i>	<i>BOOL</i>	У процесі зміни уставки локальної/дистанційної 1 : вихід <i>SP</i> форсується значенням <i>LSP_MEM</i> 0: безударний перехід
<i>track</i>	<i>BOOL</i>	1 : значення <i>SP</i> копіюється зі входу <i>PV</i> (тільки в локальному режимі)
<i>rate</i>	<i>REAL</i>	швидкість збільшення <i>SP</i> протягом зміни локальне/дистанційне в одиницю/секунду ( $\geq 0$ )

## 8.2 Задатчик співвідношення (*RATIO*)

Функціональний блок *RATIO* забезпечує розрахунок завдання на виході за формулою:

$$SP = K \cdot PV\_TRACK + bias \quad (8.1)$$

де *PV\_TRACK* – значення змінної, відносно якої розраховується завдання; *bias* – зміщення.

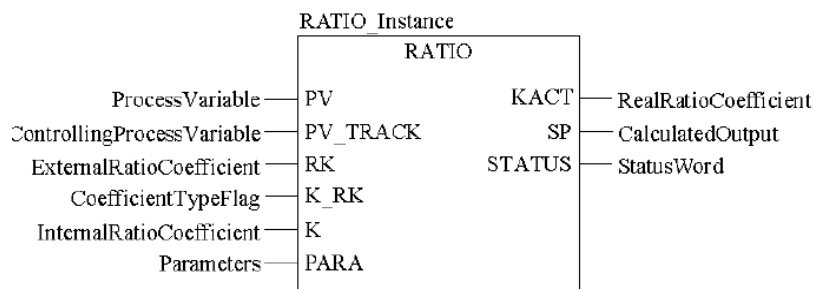


Рисунок 8.2– Приклад виклику блоку *RATIO*.

Таблиця 8.3 Параметри блоку RATIO

Вхідні параметри		
<i>PV</i>	<i>REAL</i>	Значення вимірювальної величини, що подається на регулятор (тільки для розрахунку <i>KACT</i> )
<i>PV_TRACK</i>	<i>REAL</i>	значення змінної, відносно якої розраховується завдання
<i>RK</i>	<i>REAL</i>	дистанційний коефіцієнт співвідношення
<i>K_RK</i>	<i>BOOL</i>	Вибір джерела для коефіцієнта: "1": коефіцієнт задається дистанційно входом <i>RK</i> "0": локальний коефіцієнт, задається входом <i>K</i>
<i>K</i>	<i>REAL</i>	локальний коефіцієнт співвідношення
<i>PARA</i>	<i>Para_RATIO</i>	Параметри
Вихідні параметри		
<i>KACT</i>	<i>REAL</i>	Реальний коефіцієнт співвідношення
<i>SP</i>	<i>REAL</i>	Розраховане завдання
<i>STATUS</i>	<i>WORD</i>	Слово стану

Таблиця 8.4 Опис структурного типу Para\_RATIO

<i>k_min</i>	<i>REAL</i>	обмеження по мінімуму <i>K</i> та <i>RK</i>
<i>k_max</i>	<i>REAL</i>	обмеження по максимуму <i>K</i> та <i>RK</i>
<i>sp_min</i>	<i>REAL</i>	обмеження по мінімуму для розрахованого <i>SP</i>
<i>sp_max</i>	<i>REAL</i>	обмеження по максимуму для розрахованого <i>SP</i>
<i>bias</i>	<i>REAL</i>	Зміщення

Функціональний блок може бути використаний при слідкуючому регулюванні (наприклад регулятор співвідношення) для формування завдання регуляторам *PI\_B/PIDFF*.

Крім реалізації лінійної залежності, блок має додаткові можливості (рис.8.3):

- дистанційну зміну коефіцієнту *K* або *RK*;
- обмеження по мінімуму та максимуму для *K* або *RK*;
- обмеження по мінімуму та максимуму для розрахованого *SP*;
- розрахунок реального коефіцієнта:  $KACT = (PV - bias) / PV\_TRACK$ ;

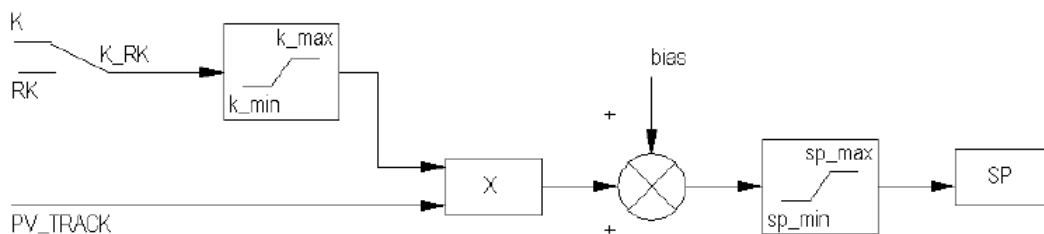


Рисунок 8.3– Функціональна схема RATIO.

### 8.3 Зміна уставки з постійною швидкістю (*RAMP*)

Функціональний блок *RAMP* забезпечує плавну зміну виходу *SP* від попереднього значення до значення входу *RSP* (рис.8.4). Швидкість зміни



налаштовується двома параметрами з *Para\_RAMP*: на збільшення – *inc\_rate*, на зменшення - *dec\_rate*. Рівність  $SP=RSP$  сигналізується виходом *DONE*. Блок *RAMP* може бути використаний для плавної зміни завдання або виходу регулятора, створеного користувачем.

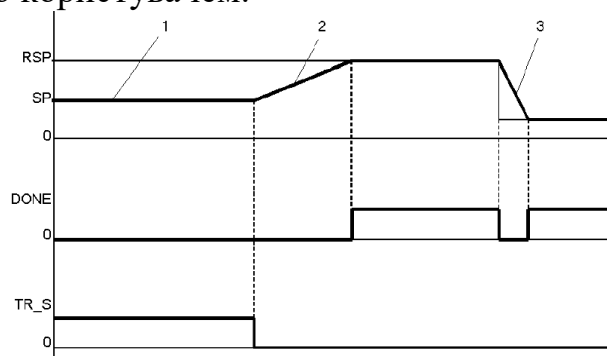


Рис.8.4. Приклад виклику блоку RAMP.

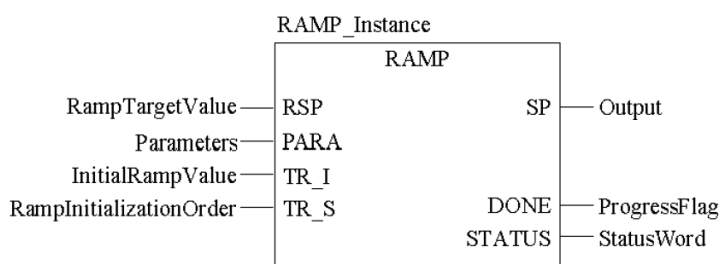


Рис.8.5. Приклад виклику блоку RAMP.

Таблиця 8.5

Параметри блоку *RAMP*

Вхідні параметри		
<i>RSP</i>	<i>REAL</i>	Задане значення уставки
<i>PARAM</i>	<i>Para_RAMP</i>	Параметри
<i>TR_I</i>	<i>REAL</i>	Значення ініціалізації
<i>TR_S</i>	<i>BOOL</i>	1 – включити ініціалізацію
Вихідні параметри		
<i>SP</i>	<i>REAL</i>	Вихідна уставка
<i>DONE</i>	<i>BOOL</i>	1: значення досягнуло уставки
<i>STATUS</i>	<i>WORD</i>	Слово стану

Таблиця 8.6

Опис структурного типу *Para\_RAMP*

<i>inc_rate</i>	<i>REAL</i>	Швидкість нарощування в одиницях/секунду ( $\geq 0$ )
<i>dec_rate</i>	<i>REAL</i>	Швидкість спаду в одиницях/секунду ( $\geq 0$ )

## 9. Блоки додаткової обробки

### 9.1 SCALING (сімейство *Conditioning*)

Даний функціональний блок призначений для масштабування числових величин. Він реалізовує лінійну залежність вихідної величини (*OUT*) від вхідної (*IN*) за формулою:

$$OUT = (IN - in\_min) \cdot \frac{(out\_max - out\_min)}{(in\_max - in\_min)} + out\_min \quad (9.1)$$

Графічно залежність виходу  $OUT$  від входу  $IN$  показана на рис.9.1. Мінімальні та максимальні вхідні ( $in\_min$ ,  $in\_max$ ) та вихідні ( $out\_min$ ,  $out\_max$ ) величини, відносно яких проводиться масштабування, задаються у вхідному параметрі  $PARAM$  типу  $Para\_SCALING$ . Тип даних  $Para\_SCALING$  включає 4-ри поля типу  $REAL$  для завдання вхідних та вихідних меж, а також одне поле " $clip$ " типу  $BOOL$  для визначення необхідності обмеження вихідної величини (див. рис.4.44).

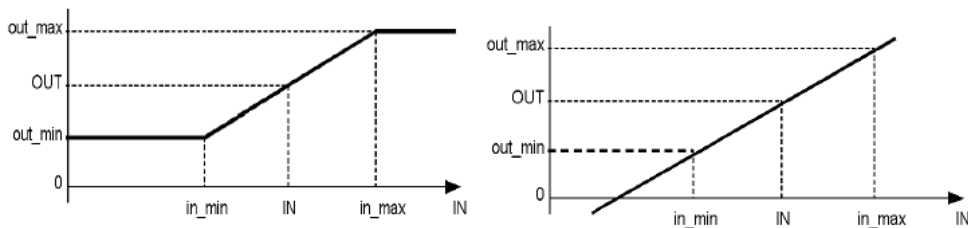


Рис.9.1. Залежність масштабованого вихідного значення ( $OUT$ ) від вхідного ( $IN$ ) при обмеженні на вихідний сигнал (зліва,  $Clip=1$ ) та без обмеження (праворуч,  $Clip=0$ ) для блока  $SCALING$ .

Функціонування блоку продемонструємо на прикладі масштабування вхідного аналогового сигналу від датчика температури з діапазоном  $0-150^{\circ}C$ , який підключений до  $\%IW0.1.1$ . Результат масштабування необхідно записати в змінну  $T1\_R$ .

По замовченню, при опитуванні, сигнали від універсальних аналогових вхідних модулів перетворюються в діапазон  $0-10000$ . Тобто вхідні межі будуть  $0-10000$ , а вихідні  $0-150$ . Для параметрів масштабування створюємо змінну  $T\_PARAM$  типу  $Para\_SCALING$ , властивості  $VALUE$  для полів заповнюємо відповідно до рис.9.2 (зверху). Присвоїмо поле  $clip:=TRUE$  для обмеження по мінімуму та максимуму вихідної (масштабованої величини).

Вигляд програми користувача на FBD показаний на рис.9.2 (внизу). Створюється екземпляр функціонального блоку  $SCALE\_T1$  типу  $SCALING$ . Попередньо  $\%IW0.1.1$  перетворюється в тип  $REAL$ , відповідно до типу параметру  $IN$  функціонального блоку  $SCALING$ . Вихідний параметр  $STATUS$  потрібен для контролю за помилками, в прикладі не використовується.

Name	Type	Value	Comment
T_PARAM	Para_SCALING		
in_min	REAL	0.0	мінімальне вхідне значення
in_max	REAL	10000.0	максимальне вхідне значення
out_min	REAL	0.0	мінімальне масштабоване значення
out_max	REAL	150.0	максимальне масштабоване значення
clip	BOOL	true	активувати обмеження по виходу

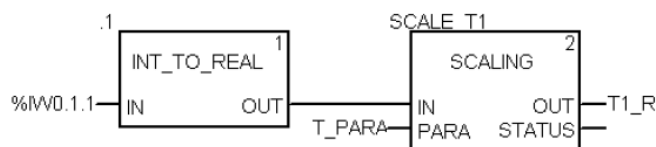


Рисунок 9.2– Приклад використання функціонального блоку  $SCALING$  для масштабування аналогового вхідного сигналу: зверху – опис структурної змінної  $T\_PARAM$  типу  $Para\_SCALING$ , знизу – приклад програми на FBD.

Аналогічним чином можна проводити масштабування вихідної величини.

## 9.2. Ланка транспортного запізнювання *DTIME* (сімейство *Conditioning*)

Функціональний блок *DTIME* призначений для реалізації ланки чистого (транспортного) запізнювання між входом *IN* та виходом *OUT*. Час запізнювання визначається значенням *T\_DELAY* (рис.9.3)

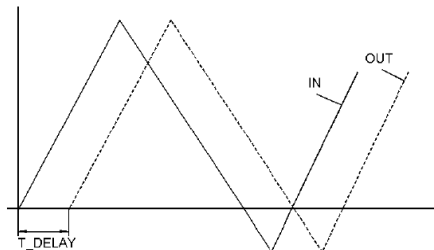


Рисунок 9.3– Діаграма роботи *DTIME*.

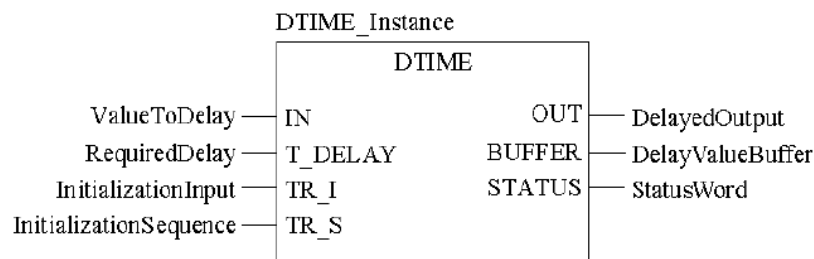


Рисунок 9.4– Приклад виклику блоку *DTIME*.

Таблиця 9.1 Параметри блоку *DTIME*

<b>Вхідні параметри</b>		
<i>IN</i>	<i>REAL</i>	Вхідне числове значення
<i>T_DELAY</i>	<i>TIME</i>	Час запізнення
<i>TR_I</i>	<i>REAL</i>	Вхід ініціалізації
<i>TR_S</i>	<i>BOOL</i>	Команда ініціалізації
<b>Вихідні параметри</b>		
<i>OUT</i>	<i>REAL</i>	Вихідне значення
<i>BUFFER</i>	<i>ANY</i>	Пам'ять для збереження значень (завжди повинен бути підключений до змінної)
<i>STATUS</i>	<i>WORD</i>	Слово стану

Для розміщення даних транспортного запізнення між входом і виходом використовується змінна, яка підключається до виходу *BUFFER*. Ця змінна має тип *ANY*, враховуючи що розмір буфер може бути різним. Однак в більшості випадків розмір буфера буде більше ніж *REAL*, оскільки кількість значень в буфері буде більше 1-го. Тому в якості змінної рекомендується використовувати масив типу *REAL*. Наприклад, масив *ARRAY [0..10] of REAL* може вміщувати до 11 елементів.

Максимальна величина затримки розраховується за формулою:

$$T\_DELAY_{max} = n \cdot T\_Period \quad (9.1)$$

де  $n$  - кількість значень, які можуть бути збережені в  $BUFFER$ ,  $T\_Period$ -інтервал виклику функціонального блоку.

### 9.3. Аперіодична ланка $LAG\_FILTER$ (сімейство $Conditioning$ )

Функціональний блок  $LAG\_FILTER$  призначений для реалізації аперіодичної ланки (1-го порядку), де вихід розраховується за формулою:

$$OUT = OUT_{old} + \frac{dt}{LAG + dt} \cdot (GAIN \cdot \frac{IN_{old} + IN}{2} - OUT_{old})$$

де змінні з індексами  $old$  – значення на попередньому виклику,  $dt$  – інтервал між викликами блоку, інші параметри наведені в таблиці 9.2.

Даний блок повинен обов'язково викликатися на першому циклі ПЛК.

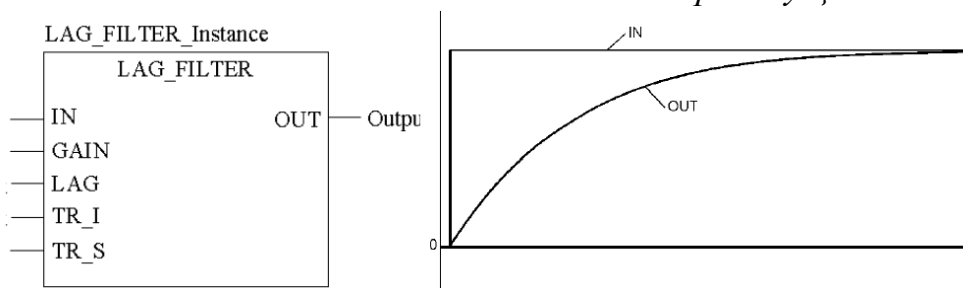


Рис.9.4. Приклад виклику блоку  $LAG\_FILTER$  та діаграма його роботи.

Таблиця 9.2

Параметри блоку  $LAG\_FILTER$

Вхідні параметри		
$IN$	$REAL$	Вхідне значення
$GAIN$	$REAL$	Коефіцієнт підсилення
$LAG$	$TIME$	Стала часу
$TR\_I$	$REAL$	Вхід ініціалізації (слідкування)
$TR\_S$	$BOOL$	1 = режим слідкування 0 = автоматичний режим
Вихідні параметри		
$OUT$	$REAL$	Вихід

## Програмування ПЛК систем локальної автоматизації. Програмування САР рівня рідини та температури продукту.

### 10.1. Стабілізаційне регулювання рівня рідини

#### Завдання.

Створити проект для ПЛК М340, для реалізації програми управління установкою, що описується наступним алгоритмом (рис.10.1). Після нажимання кнопки "СТАРТ" відкривається клапан набору першого продукту. Після

досягнення середнього рівня клапан 1-го продукту закривається, відкривається клапан набору 2- го продукту. Після спрацювання сигналізатору верхнього рівня закривається клапан набору 2-го продукту, відкривається клапан пари на 100% (діапазон виходу 0-100%). Після досягнення температури 95°C (діапазон датчику 0-150°C) включається етап витримки. Витримка повинна тривати 3 хвилини, в цей час регулятор повинен підтримувати температуру на заданому рівні. Задане значення температури визначає оператор. Програму перевірити та налагодити з використанням операторських екранів.

Після закінчення витримки, рідина зливається з апарату. Після відключення сигналізатору нижнього рівня, цикл повторюється у випадку якщо кнопка СТОП не нажата. Якщо СТОП нажата – клапан зливу закривається. У ПЛК поступає сигнал від датчика рівня з діапазоном вимірювання 0-5 м.

### Рішення.

Модифікована частина програми показана на рис.10.2, перелік змінних та екземплярів функціональних блоків наведений на рис.10.3 та рис.10.4.

Змінна що відповідає за управління виконавчим механізмом *VR\_par\_R* управляється регулятором *TIC1*, який реалізує ПІ закон управління. Регулятор викликається кожні 100 мс, оскільки його вхід *EN* підключений до блоку *SMPL1*. Ручне управління клапаном *VR\_par\_R* забезпечується через функцію *SEL* (".16"), який перемикає *VR\_par\_R* на значення *VR\_par\_MAN*. Для безударності переходу замість *SEL* можна використання блок *MS*.

Регулятор *TIC1* працює в режимі регулювання тільки при (*StepProg=4*), в інших випадках він працює в режимі слідкування, тобто на вихід *TIC1.OUT* буде подаватися значення *TIC1.TR\_I*. Таким чином на усіх етапах роботи програми *VR\_par\_R* буде управлятися значенням з *MUX*, тільки на етапі витримки буде включатися режим регулювання.

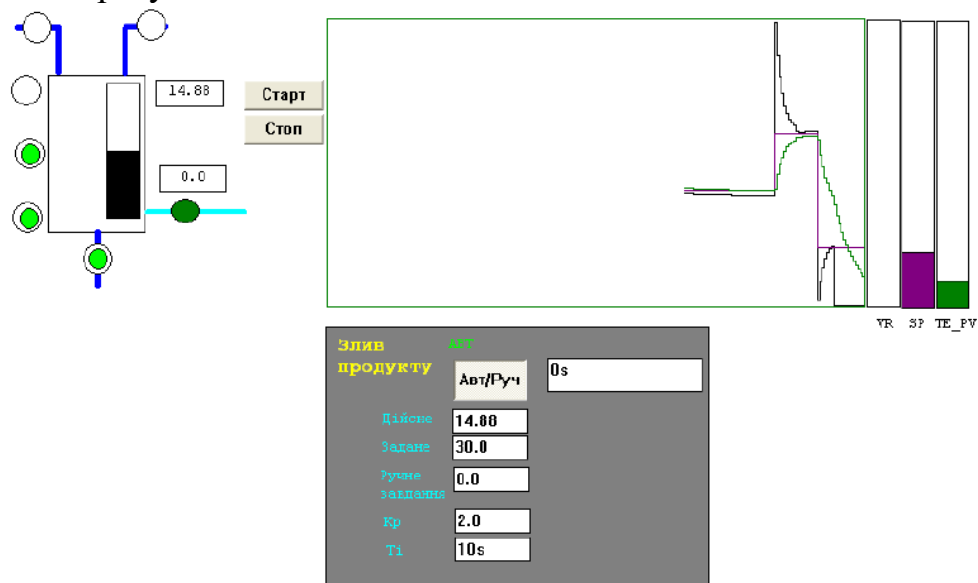


Рисунок 10.1– Приклад операторського екрану до поставленої задачі.

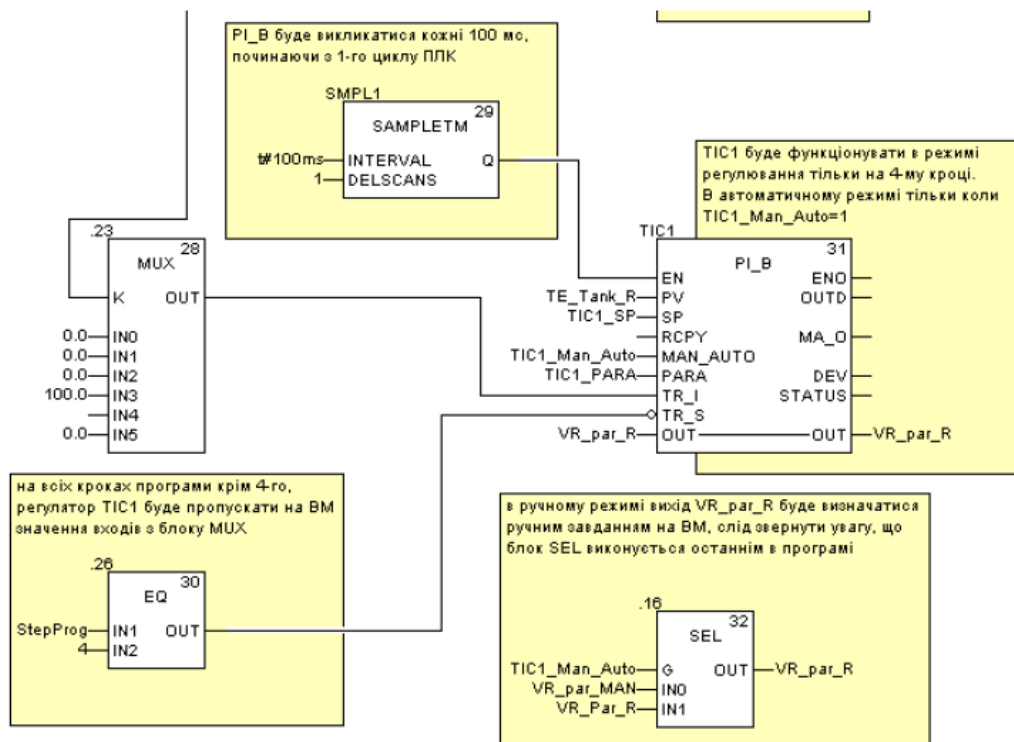


Рисунок 10.2– Модифікована частина секції програми

Name	Type	Addr...	Value	Comment
LE_Tank	INT	%Iw0.1.0		Датчик рівня
LE_Tank_R	REAL			Відмасштабований Рівень 0-5 м
LS_nyz	EBOOL	%I0.2.0		Сигналізатор нижнього рівня
LS_ser	EBOOL	%I0.2.1		Сигналізатор середнього рівня
LS_verh	EBOOL	%I0.2.2		Сигналізатор верхнього рівня
SB_Start	BOOL			Кнопка запуску процесу
SB_stop	BOOL			Кнопка зупинки процесу
SCALE_PARA_L	Para_SCALING			
SCALE_PARA_T	Para_SCALING			
StepProg	INT		0	Крок програми
TE_Tank	INT	%Iw0.1.1		Датчик температури
TE_Tank_R	REAL			Відмасштабована Температура 0-150 гр.С
TIC1_Man_Auto	EBOOL		1	Режим руч/авт для регулятора температури
TIC1_PARA	Para_PI_B			
id	UINT			
pv_inf	REAL		0.0	обмеження по мінімуму вхідної величини
pv_sup	REAL		150.0	обмеження по максимуму вхідної величини
out_inf	REAL		0.0	обмеження по мінімуму вихідної величини
out_sup	REAL		100.0	обмеження по максимуму вихідної величини
rev_dir	BOOL		false	режим прямої роботи ПІ-регулятора
en_rcpy	BOOL		false	не використовувати RCPY
kp	REAL		1.0	коефіцієнт пропорційності
ti	TIME		t#0s	час інтегрування
dband	REAL		0.2	зона нечутливості
outbias	REAL		50.0	зміщення виходу регулятора в П-режимі
TIC1_SP	REAL		80.0	Задане значення на регулятор температури
VA_nabor1	EBOOL	%Q0.2.16		Клапан набору 1-го продукту
VA_nabor2	EBOOL	%Q0.2.17		Клапан набору 2-го продукту
VA_sliv	EBOOL	%Q0.2.18		Клапан зливу
VR_par	INT	%Qw0.1.4		Клапан пари
VR_par_MAN	REAL			Вихід на ВМ в ручному режимі
VR_Par_R	REAL			Відмасштабоване значення ВМ 0-100%

Рисунок 10.3– Перелік змінних.

Name	r	Type	v	Comment
RS_Nabor1		RS		Тригер для управління VA_Nabor1
RS_Nabor2		RS		Тригер для управління VA_Nabor2
RS_Sliv		RS		Тригер для управління VA_Sliv
SCALE_L		SCALING		екземпляр функц блока масштабування для LE_Tank
SCALE_T		SCALING		екземпляр функц блока масштабування для TE_Tank
SMPL1		SAMPLETM		
T_DELAY		TON		екземпляр таймера на витримку
TIC1		PI_B		екземпляр функц болка ПІ-регулятора

Рисунок 10.4– Перелік екземплярів функціональних блоків.

## 10.2. Каскадне регулювання температури Завдання.

Необхідно створити проект в UNITY PRO для реалізації поставленої задачі (рис.10.5) з використанням FFB бібліотеки *ControlLIB*.

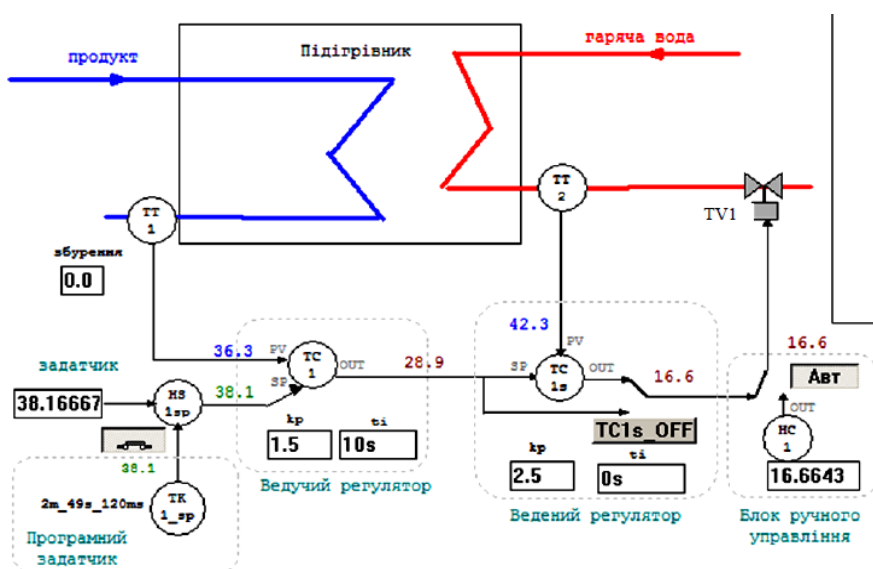


Рисунок 10.5– Операторський екран для контролю та управління процесом нагрівання

Технологічна установка являє собою теплообмінник для підігріву рідкого продукту (рис.10.5) за допомогою гарячої води. Температура продукту на 4 виході вимірюється датчиком температури  $TT1$  (0-100°C, вхід ПЛК  $\%IW0.1.2$ ) який вмонтований на трубопроводі на відстані кількох метрів від підігрівника, а регулюється витратою гарячої води регулюючим клапаном  $TV1$  на виході підігрівника (0-100% ХРО, вихід  $\%QW0.2.0$ ). Додатково вимірюється також температура води в трубопроводі безпосередньо на виході підігрівника за допомогою датчика  $TT2$  (0-100°C, вхід ПЛК  $\%IW0.1.1$ ). Інерційність об'єкта по каналу  $TT2$  менша ніж по  $TT1$ .

Система управління повинна забезпечити регулювання температури  $TT1$  з підтримкою наступних функцій:

- 1) стабілізація температури на виході підігрівника на заданому оператором або програмним задатчиком значенні, з використанням каскадного регулятора, де:
  - а. ведучий регулятор (ІІІ режим) стабілізує температуру  $TT1$ ;
  - б. ведений регулятор (ІІ режим) служить для швидкої реакції контуру регулювання на зміну малоінерційного  $TT2$ ;
- 2) можливість настройки коефіцієнтів  $Kp$  та  $Ti$  з операторського екрану;
- 3) можливість ручного управління виконавчим механізмом  $TV1$  з операторського екрану при переключенні в ручний режим з забезпеченням безударності переходу ;
- 4) можливість формування завдання програмним задатчиком в залежності від часу по залежності показаній на діаграмі рис. 10.6.
- 5) можливість переключення з ручного завдання на програмне управління та навпаки в будь який момент часу з забезпеченням безударності переходу;
- б) можливість виключення веденого регулятора з каскаду (з забезпеченням безударності переходу), тобто переводу контуру в режим прямого ІІІ-регулювання по температурі  $TT1$ ;
- 7) при ручному управлінні виконавчим механізмом ведучий регулятор повинен фіксувати своє вихідне значення в останньому положенні.

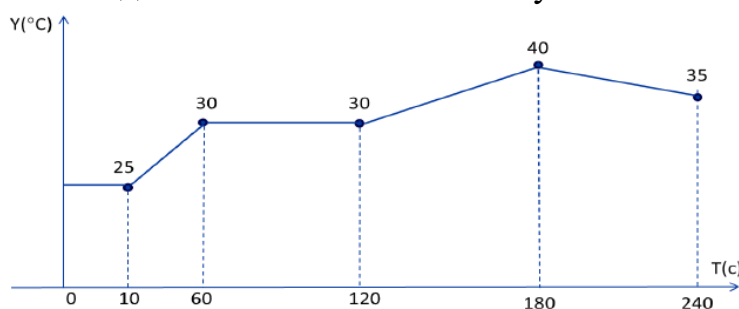


Рисунок 10.6– Залежність завдання температури від часу

### Рішення.

**Загальні принципи роботи контуру.** Функціональна схема контуру регулювання температури продукту показана на операторському екрані (рис.10.5).

Контур включає:

- канали вимірювання ( $TT1$ ,  $TT2$ );
- канал формування завдання, який складається з ручного задатчика, програмного задатчика  $TK1\_sp$  (формування програмного управління по часу), перемикача ручного/програмного завдання  $HS1sp$  (включена кнопка – запуск програмного задатчика) ;
- ведучого регулятора  $TC1$  (ІІІ закон), який на основі сформованого завдання (вхід  $SP$ ) та вимірювального значення ( $TT1$  на вхід  $PV$ ), формує сигнал управління (вихід  $OUT$ ), який подається на ведений регулятор  $TC1s$ ; для регулятора налаштовуються  $Kp$  та  $Ti$ ;
- веденого регулятора  $TC1s$  (ІІ закон), який на основі сформованого завдання ( $TC1.OUT$  на вхід  $SP$ ) та вимірювального значення ( $TT2$  на вхід  $PV$ ), формує сигнал управління (вихід  $OUT$ ), який подається на виконавчий механізм  $TV1$  через перемикач  $TC1s\_OFF$ ; перемикач  $TC1s\_OFF$  дозволяє переключити



*TVI* на вихід ведучого регулятора *TC1.OUT*, тим самим відключивши ведений регулятор з контуру управління; для регулятора налаштовується  $K_p$  а також  $T_i$ , ненульове значення якого дозволяє перевести регулятор в ПІ-режим;

- блок ручного управління *HC1*, який дає можливість переключитися в режим ручного управління виконавчим механізмом *TVI* з операторського екрану; кнопка "АВТ" для переключення режиму (нажата - автоматичний режим);

- канал управління (*TVI*);

Перелік змінних та екземплярів функціональних блоків наведений на рис.10.7.

The figure consists of three screenshots of a software interface, likely a control system design tool, showing variable declarations and parameter lists.

**Top Screenshot: Function Blocks**

Name	r	Type	Comment
HC1		MS	Блок ручного управління
smHeater1		smHeatExch	Імітаційна модель підігрівача
HS1sp		SP_SEL	Перемикач ручного/програмного завдання
smkl1		smValve	Імітаційна модель клапану
smDEL1		smDELAY	Імітаційна модель блоку запізнення
SMPL1		SAMPLETM	Диспетчер виклику
TC1		PI_B	Ведучий регулятор
TC1s		PI_B	Ведений регулятор
TK1		TON	Таймер програмного задатчика

**Middle Screenshot: Variables**

Name	Type	Value	Comment
smInit	BOOL		Ініціалізація імітаційної моделі
TC1_AUTO	BOOL		1 - включити автоматичний режим роботи контуру
TC1s_OFF	BOOL		1 - відключити ведений регулятор
TK1_ON	BOOL		1 - включити програмний задатчик
smZ1	REAL		імітація збурення
TC1_OUT	REAL		Вихід ведучого регулятора
TC1_SP	REAL	20.0	Уставка для ведучого регулятора
TC1s_OUT	REAL		Вихід веденого регулятора
TK1_SP	REAL		Вихід завдання з програмного задатчика
TT1	REAL		T продукту на виході підігрівача
TT2	REAL		T гарячої води на виході підігрівача
TV1	REAL		Клапан подачі гарячої води

**Bottom Screenshot: Parameters**

Name	Type	Value	Comment
HC1_PARA	Para_MS		Параметри ручного задатчика
out_min	REAL	0.0	
out_max	REAL	100.0	
inc_rate	REAL		
dec_rate	REAL		
outbias	REAL		
use_bias	BOOL	FALSE	
bumpless	BOOL	FALSE	
TC1_PARA	Para_PI_B		Параметри вдеучого регулятора
TC1s_PARA	Para_PI_B		Параметри веденого регулятора
HS1sp_PA...	Para_SP_SEL		Параметри перемикача завдань
sp_min	REAL	0.0	
sp_max	REAL	100.0	
bump	BOOL		
track	BOOL		
rate	REAL		

Рисунок 10.7– Змінні проекту

**Загальні принципи роботи програми.** Для реалізації даної задачі використовуються 4-ри секції (рис.10.8): секція "INPUTS" – для обробки входних каналів вимірювання (датчики температури); секція "OUTPUTS" – для обробки вихідного каналу правління; секція "CTRL1" – для реалізації контуру регулювання. Секція "Simulation" призначена тільки для імітації об'єкта при налагодженні програми і є необов'язковою.

У секції "INPUTS" оцифровані значення аналогових входів масштабуються шляхом множення на коефіцієнт (діапазон 0-10000 в 0-100°C). У секції "OUTPUTS" аналогічним чином проводиться масштабування для значення виконавчого механізму TV1, тільки в зворотному напрямку (діапазон 0-100%XPO в 0-10000).

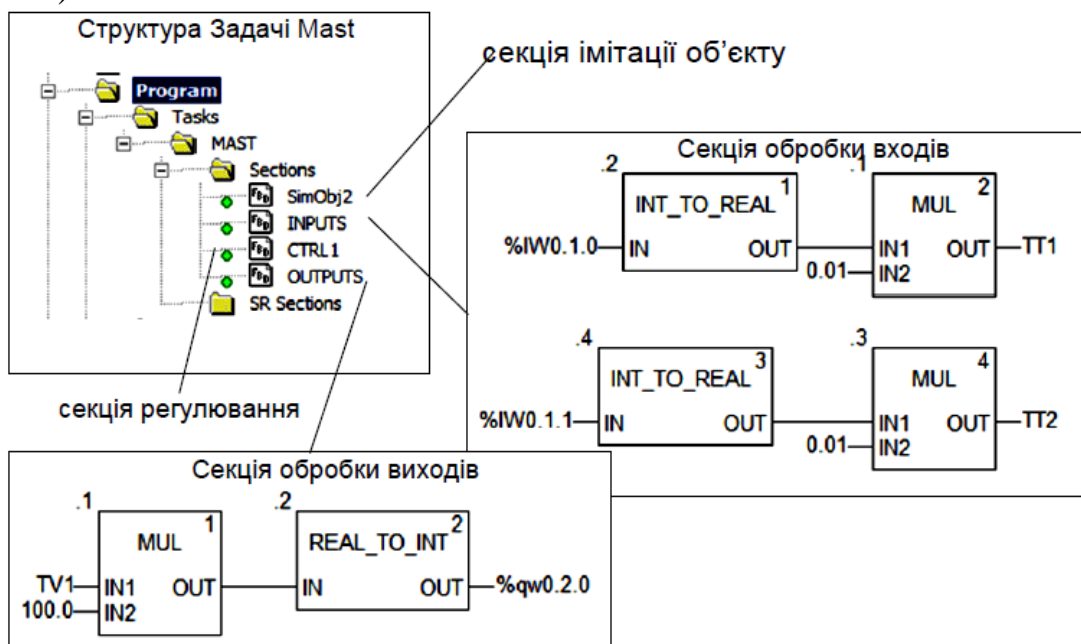


Рисунок 10.8– Структура задачі MAST

**Реалізація програмного задатчика.** Для реалізації програмного задатчика використана процедура *LOOKUP\_TABLE1*, яка реалізовує кусочно-лінійну інтерполяцію за заданими вузловими точками. Вузлові точки задаються парами значень  $X_i Y_i$ , перше з яких – час в секундах, друга – задане значення температури в цій точці. Таким чином на вході X процедура буде отримувати значення часу роботи програмного задатчика в форматі REAL, в залежності від якого на виході Y буде формуватися задане значення (уставка для регулятора) *TK1\_SP*.

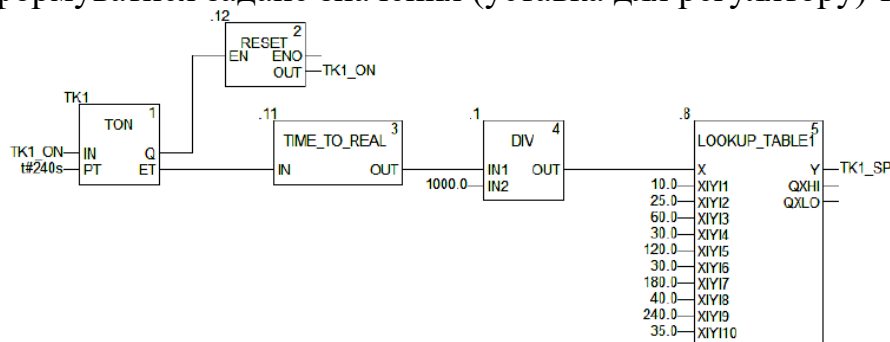


Рисунок 10.9– Реалізація програмного за датчика

Для формування часу роботи задатчика використовується таймер *TK1*, який стартує по команді запуску задатчика *TK1\_ON*, та налаштований на повний час його роботи (240с). Спрацювання виходу *TK1.Q*, тобто по закінченню роботи програмного задатчика, приводить до скидання команди *TK1\_ON*. У зв'язку з тим, що вихід таймеру *TK1.ET* видає значення типу *TIME*, воно перетворюється в тип *REAL*, а отримані мілісекунди шляхом ділення на 1000 переводяться в секунди.

**Загальні принципи реалізації контуру регулювання.** Фрагмент програми реалізації контуру регулювання показана на рис.10.10. Слід зазначити, що старші версії UNITY PRO (<6.0) не підтримують можливість безпосереднього з'єднання в FBD виходів типу *INOUT* та *IN* (або *OUT*).

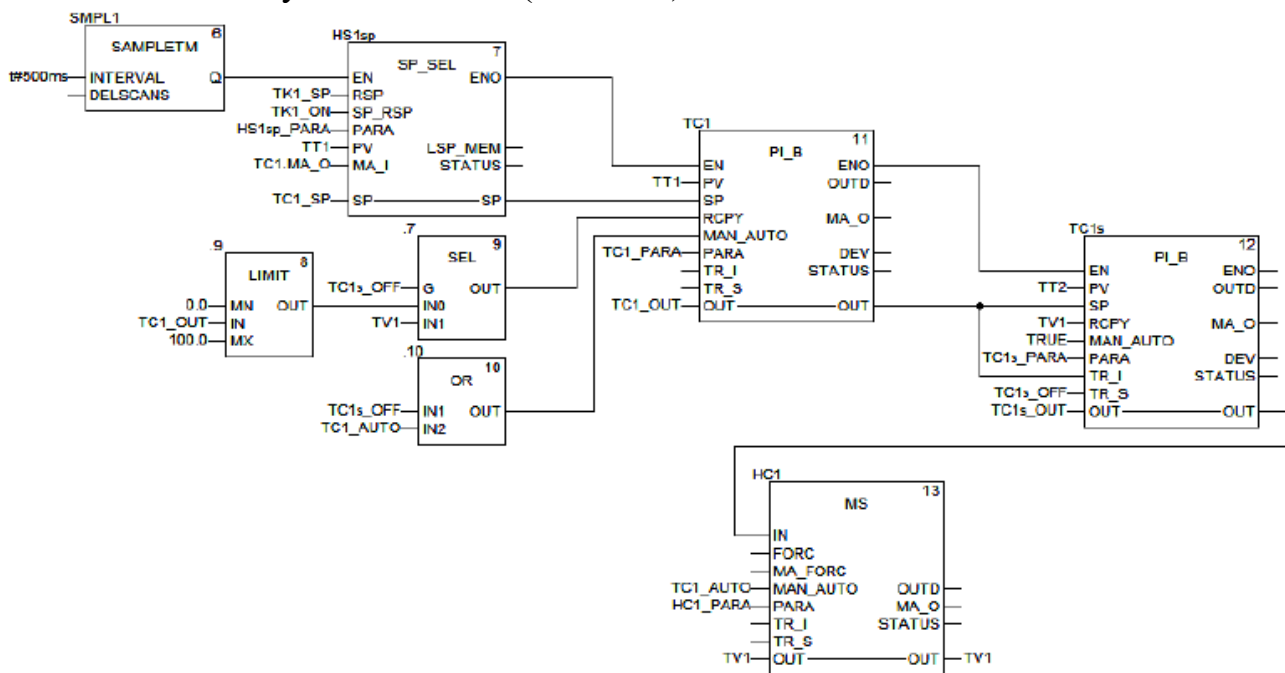


Рисунок 10.10– Фрагмент секції з реалізацією контуру регулювання (для UNITY PRO  $\geq 6.0$ )

Використовуючи *SMPL1* та входи *EN* функціональних блоків, блоки *HS1sp*(перемикача завдань), *TC1*(ведучий регулятор) та *TC1s*(ведений регулятор) викликаються періодично з часом 500 мс, послідовно один за одним. Це дає змогу зменшити навантаження на ресурси ПЛК та виконавчих механізмів в автоматичному режимі. Однак для ручного режиму необхідна швидка реакція системи на дії оператора, тому блок ручного завдання викликається з кожним циклом (вхід *EN* не задіяний).

**Реалізація каналу формування завдання.** Вихід блоку *HS1sp.SP* через змінну *TC1\_SP* формує завдання для блоку *TC1*, вихід якого в свою чергу формує завдання *TC1s*. Таким чином формується каскадне регулювання.

Блок перемикавання завдання *HS1sp.SP* на вході *RSP* (дистанційне завдання) отримує значення в змінній *TK1\_SP* з програмного задатчика. При активації оператором програмного управління (*TK1\_ON=TRUE*), через вхід *SP\_RSP* блок *HS1sp* перемикається в дистанційний режим, при якому *HS1sp.SP=HS1sp.TK1\_SP*. Це ж значення копіюється в змінну *TC1\_SP*, так як вона прив'язана до входу/виходу *SP*. У локальному режимі (*TK1\_ON=FALSE*) оператор може змінювати уставку безпосередньо змінюючи *TIC1*.

**Реалізація каскадного регулятора.** Ведучий регулятор  $TC1$  реалізований через функціональний блок типу  $PI\_B$ . Його задача забезпечити значення температури  $TT1$  на рівні заданому  $TC1\_SP$ .

Ведений регулятор  $TC1s$  теж реалізований через функціональний блок типу  $PI\_B$ , однак по замовченню він працює в П-режимі, так як інтегральна складова відключена ( $Ti=0$ ). Його задача швидко зреагувати на зміну менш інерційної змінної процесу  $TT2$ . Таким чином велику частину збурень швидко компенсує ведений регулятор, а ведучий регулятор забезпечить точність регулювання.

Враховуючи вимоги до функцій контуру регулювання, ведучий регулятор повинен мати можливість працювати в каскаді з веденим регулятором ( $TC1s$ ), або самостійно без нього. У каскадному режимі вихід регулятора підключається до  $TC1s.SP$  через змінну  $TC1\_OUT$ , тим самим формуючи завдання веденому регулятору. Виведення з каскаду веденого регулятора проводиться шляхом переключення  $TC1s$  в режим слідкування (*Tracking*), подавши на його вхід  $TR\_s$  змінну  $TC1s\_OFF$  ( $TC1s.TR\_s=TC1s\_OFF$ ). Тобто в режимі слідкування на вихід  $TC1s.OUT$  буде подаватися значення  $TC1s.TR\_I=TC1\_OUT$ .

Враховуючи вимоги до безударності переходів, та різні режими роботи контуру регулювання, обидва регулятори  $TC1$  та  $TC1s$  працюють в режимі використання входу  $RCPY$  ( $en\_rcpy=TRUE$ ). Це значить, що нове значення виходу  $OUT$  регулятори будуть розраховувати на базі значення входу  $RCPY$ . Для веденого регулятора  $TC1s$  на вхід  $RCPY$  завжди подається значення, яке йде на виконавчий механізм  $TVI$ . Тобто, якщо ведений регулятор виведений з каскаду, він все одно буде формувати нове значення виходу внутрішнього регулятора на базі плинного  $TVI$ , що забезпечить безударний перехід при повторному вводити його в каскад. Для ведучого регулятора значення  $RCPY$  залежить від того, чи включений в контур управління ведений регулятор  $TC1s$ . Якщо ведений регулятор виведений із каскаду, то  $RCPY=TVI$ , так як  $TC1$  безпосередньо управляє виконавчим механізмом, отже нове розраховане значення базується на значенні  $TVI$ . Якщо ведений регулятор в складі контуру, то  $RCPY=LIMIT(TC1\_OUT)$ . Функція обмеження використовується тому, що в режимі ( $en\_rcpy=TRUE$ ) обмеження на вихід блоку  $PI\_B$  не діють.

**Реалізація ручного управління.** Ручне управління контуру реалізоване через блок ручного управління  $HCI$  та відповідних режимів роботи регуляторів  $TC1$  та  $TC1s$ . Значення виходу  $HCI.OUT$  зв'язане зі змінною  $TVI$ . У автоматичному режимі роботи контуру ( $TC1\_AUTO=TRUE$ ) вихід  $HCI.OUT=TC1s\_OUT$ . У ручному режимі вихід  $HCI.OUT$  може бути змінений оператором. При цьому, враховуючи що значення  $TVI$  повторюється на входах  $RCPY$  блоків регулювання забезпечується безударний перехід.

Ведений регулятор завжди знаходиться в автоматичному режимі, так як для ручного режиму передбачений блок  $HCI$ . Ведучий регулятор, якщо він знаходиться в режимі каскаду з веденим, в ручному режимі повинен фіксувати своє вихідне значення в останньому положенні. Це забезпечується шляхом переключення його в ручний режим. Таким чином ведучий регулятор працює в автоматичному режимі, коли працює в автоматичному режимі весь контур, або коли ведений регулятор виведений з каскаду.

### 10.3. Регулювання з використанням виконавчих механізмів типу реверсивний двигун

#### Завдання.

Необхідно створити проект в UNITY PRO для реалізації поставленої задачі (рис.10.11 та рис.10.12) з використанням FFB бібліотеки *ControlLIB*. Налаштування проекту зробити з використанням готового програмного імітатора об'єкту та операторського екрану.

Технологічна установка являє собою два теплообмінника для підігріву рідкого продукту за допомогою гарячої води. У першому підігрівнику (рис.10.11) температура продукту на виході вимірюється датчиком температури *TT1* (0-100°C, вхід ПЛК *%IW0.1.2*) який вмонтований на трубопроводі на відстані кількох метрів від підігрівника, а регулюється витратою гарячої води на виході підігрівника з використанням клапану *TV1* і приводом типу МЕО (вихід ПЛК *%Q0.3.16* – "більше", *%Q0.3.17* – "менше"). Додатково вимірюється також температура води в трубопроводі безпосередньо на виході підігрівника за допомогою датчика *TT1a* (0-100°C, вхід ПЛК *%IW0.1.3*). Виконавчий механізм *TV1* має показник положення регулюючого органу (0-100%, вхід ПЛК *%IW0.1.4*). Час повного відкриття клапану – 10с, мінімальний імпульс – 250 мс.

Температура продукту на виході другого підігрівника (рис.3.70) вимірюється датчиком температури *TT2* (0-100°C, вхід ПЛК *%IW0.1.6*), а регулюється витратою гарячої води на виході підігрівника з використанням клапану *TV2* з приводом типу МЕО (вихід ПЛК *%Q0.3.18* – "більше", *%Q0.3.19* – "менше"). Додатково вимірюється також температура води в трубопроводі безпосередньо на виході підігрівника за допомогою датчика *TT2a* (0-100°C, вхід ПЛК *%IW0.1.7*). Виконавчий механізм *TV2* має датчики кінцевого положення регулюючого органу: "повністю відкритий" - вхід ПЛК *%IO.3.0*, "повністю закритий" - вхід ПЛК *%IO.3.1*. Час повного відкриття клапану – 10с, мінімальний імпульс – 250 мс.

Система управління повинна забезпечити регулювання температури *TT1* та *TT2* з підтримкою наступних функцій:

- 1) стабілізація температур на виходах підігрівників на заданому оператором значенні, з використанням ПІ регуляторів та блоків управління серводвигунами;
- 2) можливість настройки коефіцієнтів *Kp* та *Ti* з операторського екрану;
- 3) можливість ручного управління виконавчими механізмами *TV1* та *TV2* з операторського екрану при переключенні в ручний режим з забезпеченням безударності переходу.

#### Рішення.

Перелік змінних та екземплярів функціональних блоків наведені на рис.10.13.

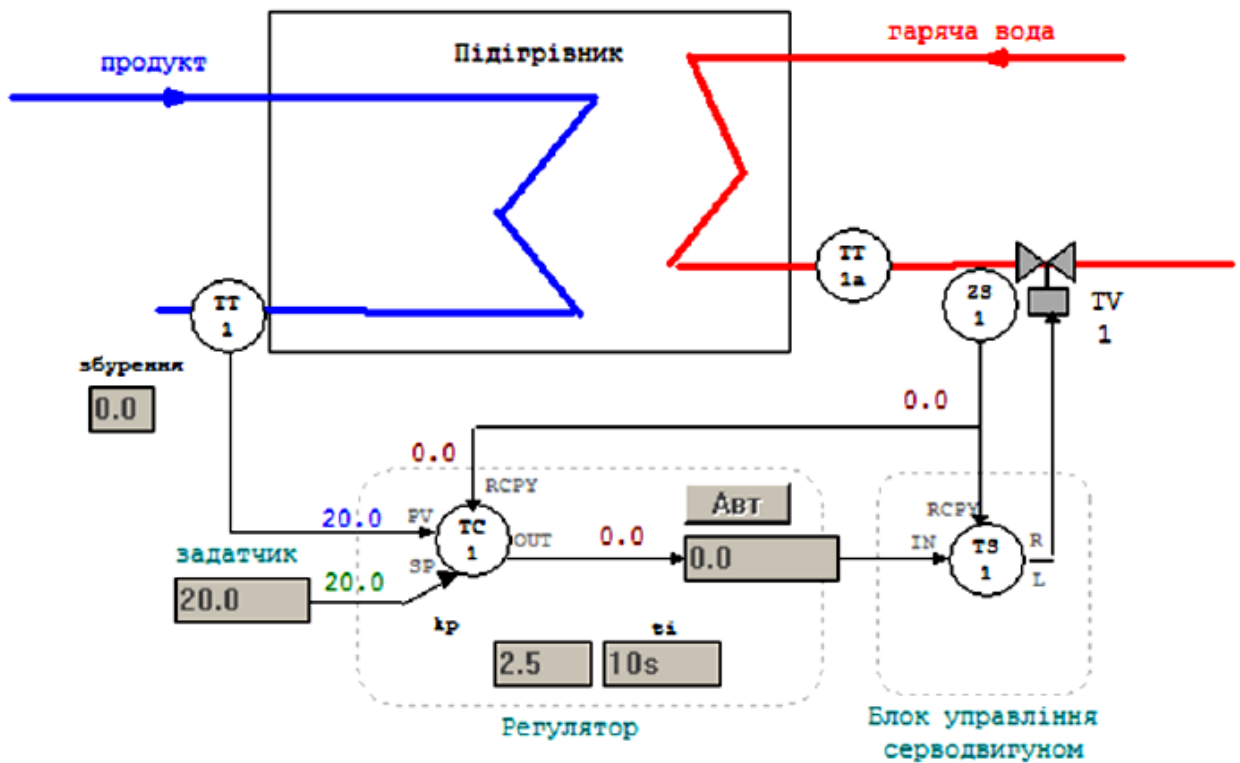


Рисунок 10.11– Операторський екран для контролю та управління процесом нагрівання в підігрівнику 1

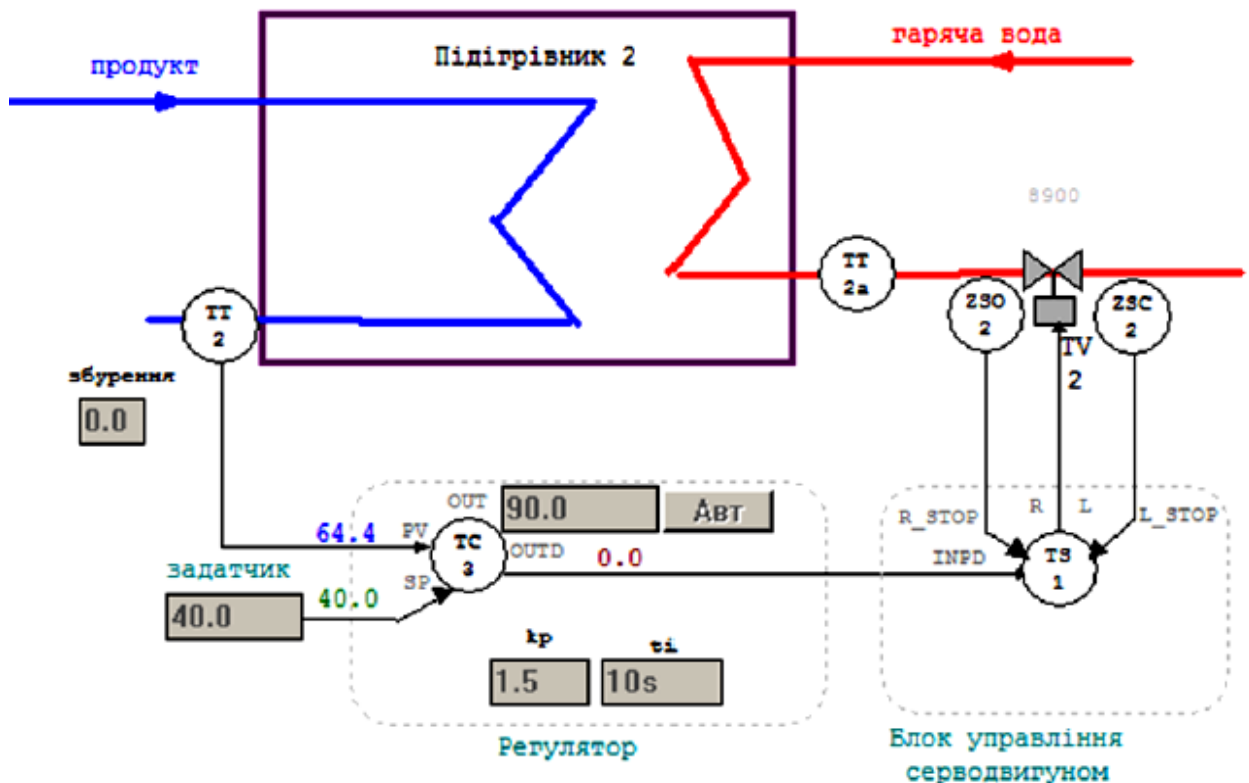


Рисунок 10.12– Операторський екран для контролю та управління процесом нагрівання в підігрівнику 2

Variables   DDT Types   Function Blocks   DFB Types				
Filter <input type="text" value="Name ="/>				
Name	/	Type	Val...	Comment
smInit		BOOL		Ініціалізація імітаційної моделі
TC1_AUTO		BOOL		1 - включити автоматичний режим роботи контуру TC1
TC2_AUTO		BOOL		1 - включити автоматичний режим роботи контуру TC2
TV1_CLS		BOOL		1 - команда на закриття Кл1
TV1_OPN		BOOL		1 - команда на відкриття Кл1
TV2_CLS		BOOL		1 - команда на закриття Кл2
TV2_OPN		BOOL		1 - команда на відкриття Кл2
ZSC2		BOOL		1 - спрацював датчик повного закриття
ZS02		BOOL		1 - спрацював датчик повного відкриття
smZ1		REAL		збурення для TT1
smZ2		REAL		збурення для TT2
TC1_OUT		REAL		Вихід регулятора TC1
TC1_SP		REAL	20.0	Уставка регулятора TC1
TC2_OUT		REAL		Вихід регулятора TC2
TC2_OUTD		REAL		Інкрементальний вихід регулятора TC2
TC2_SP		REAL	20.0	Уставка регулятора TC1
TT1		REAL		T продукту на виході підігрівача 1
TT1a		REAL		T гарячої води на виході підігрівача 1
TT2		REAL		T продукту на виході підігрівача 2
TT2a		REAL		T гарячої води на виході підігрівача 2
ZS1		REAL		показчик положення регулюючого органу Кл1

Variables   DDT Types   Function Blocks   DFB Types				
Filter <input type="text" value="Name ="/>				
Name	/	Type	Value	Comment
TC1_PARA		Para_PI_B		Параметри регулятора
id		UINT		
pv_inf		REAL	0.0	
pv_sup		REAL	100.0	
out_inf		REAL	0.0	
out_sup		REAL	100.0	
rev_dir		BOOL		
en_rcpy		BOOL	1	
kp		REAL	2.5	
ti		TIME	t#10s	
dband		REAL	0.2	
outbias		REAL		
TC2_PARA		Para_PI_B		Параметри регулятора
TS1_PARA		Para_SERVO		Параметри блоку управління серводвигуном
en_rcpy		BOOL	1	
rcpy_rev		BOOL		
t_motor		TIME	t#10s	
t_mini		TIME	t#250ms	
TS2_PARA		Para_SERVO		Параметри блоку управління серводвигуном
en_rcpy		BOOL	0	
rcpy_rev		BOOL		
t_motor		TIME	t#10s	
t_mini		TIME	t#250ms	

Variables   DDT Types   Function Blocks   DFB Types				
Filter <input type="text" value="Name &lt;&gt; sm*"/>				
Name	n..	Type	\	Comment
TC1		PI_B		Регулятор контуру 1
TC2		PI_B		Регулятор контуру 2
TS1		SERVO		Блок управління серводвигуном контуру 1
TS2		SERVO		Блок управління серводвигуном контуру 2

Рисунок 10.13– Змінні проекту

### Загальні принципи роботи контуру управління температурю в підігрівнику 1.

Функціональна схема контуру регулювання температури продукту показана на операторському екрані (рис.10.11).

Контур включає:

- канали вимірювання ( $TT1$ ,  $TT1a$ );
- регулятор  $TC1$  (ПІ закон), який на основі сформованого завдання (вхід  $SP$ ) та вимірювального значення ( $TT1$  на вхід  $PV$ ), формує сигнал управління (вихід

*OUT*), що подається на блок управління серводвигуном *TS1*; на вхід *RCPY* регулятора заводиться сигнал зворотного зв'язку по положенню регулюючого органу *ZS1*; для регулятора налаштовуються  $Kp$  та  $Ti$ ; регулятор може працювати в автоматичному або ручному режимі (вмикається/вимикається кнопка "АВТ"); в ручному режимі вихід *OUT* задається безпосередньо оператором;

- блок управління серводвигуном *TS1*, який перетворює числовий сигнал що поступає на вхід *IN* в діапазоні 0-100% у дискретні сигнали відповідної тривалості типу "більше" (вихід R) та "менше" (вихід L); на вхід *RCPY* блоку заводиться сигнал зворотного зв'язку по положенню регулюючого органу *ZS1*;

- канал управління, який окрім клапану з виконавчим механізмом TV1 включає показчик положення регулюючого органу (*ZS1*);

**Структура програми.** Для реалізації даної задачі використовуються 5-ть секцій (рис.10.14): секція "*INPUTS*" – для обробки вхідних каналів вимірювання; секція "*OUTPUTS*" – для обробки вихідних каналів управління; секція "*CTRL1*" та "*CTRL2*" – для реалізації контурів управління температурою відповідно в підігрівнику 1 та підігрівнику 2. Секція "*Simulation*" призначена тільки для імітації об'єкта.

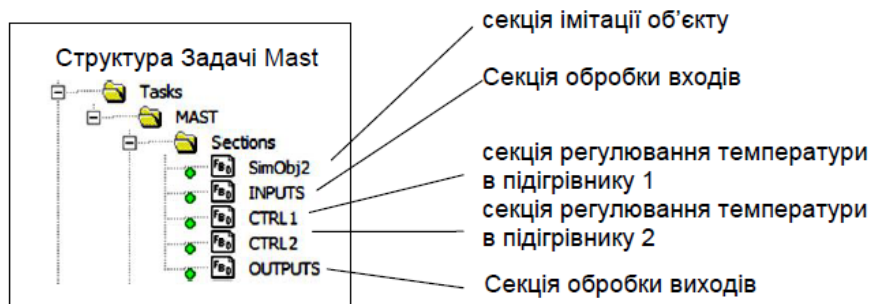


Рисунок 10.14– Структура Задачі MAST

У секції "*INPUTS*" (рис.10.15) оцифровані значення аналогових входів масштабуються шляхом множення на коефіцієнт (діапазон 0-10000 в 0-100°C). До входів контуру з підігрівачем 1 належить також показник положення *ZS1* (діапазон 0-10000 в 0-100 %XPO). До входів контуру з підігрівачем 2 належать також датчики кінцевого положення типу "відкрито" - *ZSO2*, та "закрито" - *ZSC2*.

У секції "*OUTPUTS*" (рис.10.16) для кожного виконавчого механізму на виходи ПЛК подаються сигнали "більше" (*TV1\_OPN*, *TV2\_OPN*) та "менше" (*TV1\_CLS*, *TV2\_CLS*).

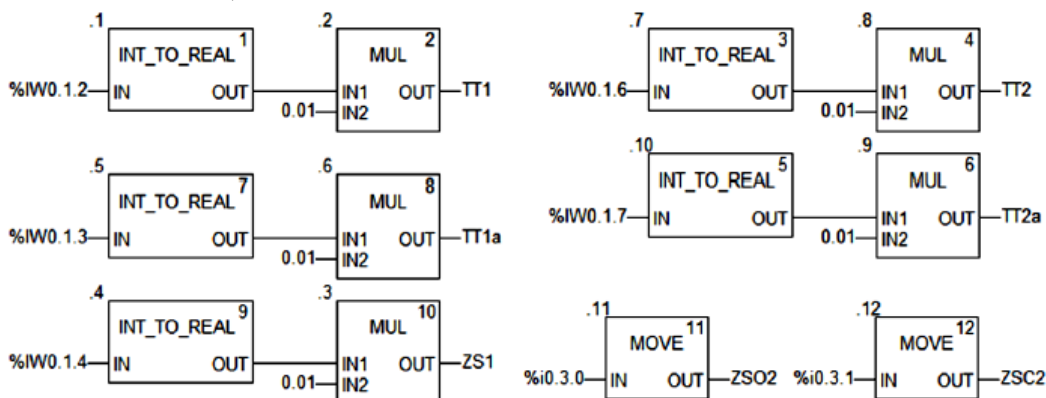


Рисунок 10.15– Секція обробки входів



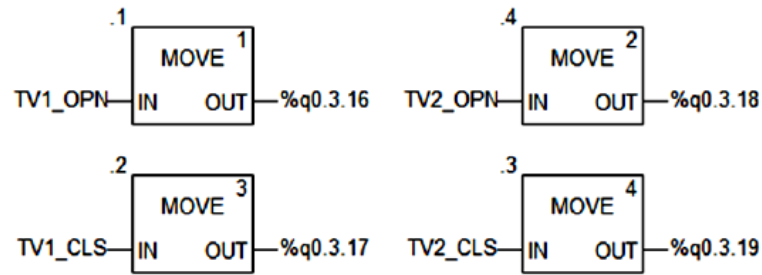


Рисунок 10.16– Секція обробки виходів

**Опис роботи програми реалізації контуру управління температурою в підігрівнику 1.** Програма секції "CTRL1" наведена на рис.10.17. Для UNITY PRO V<6.0 вхід/вихід *TC1.OUT* треба розірвати від *TS1* та *SERVO*, а зв'язок реалізувати через змінну *TC1\_OUT*.

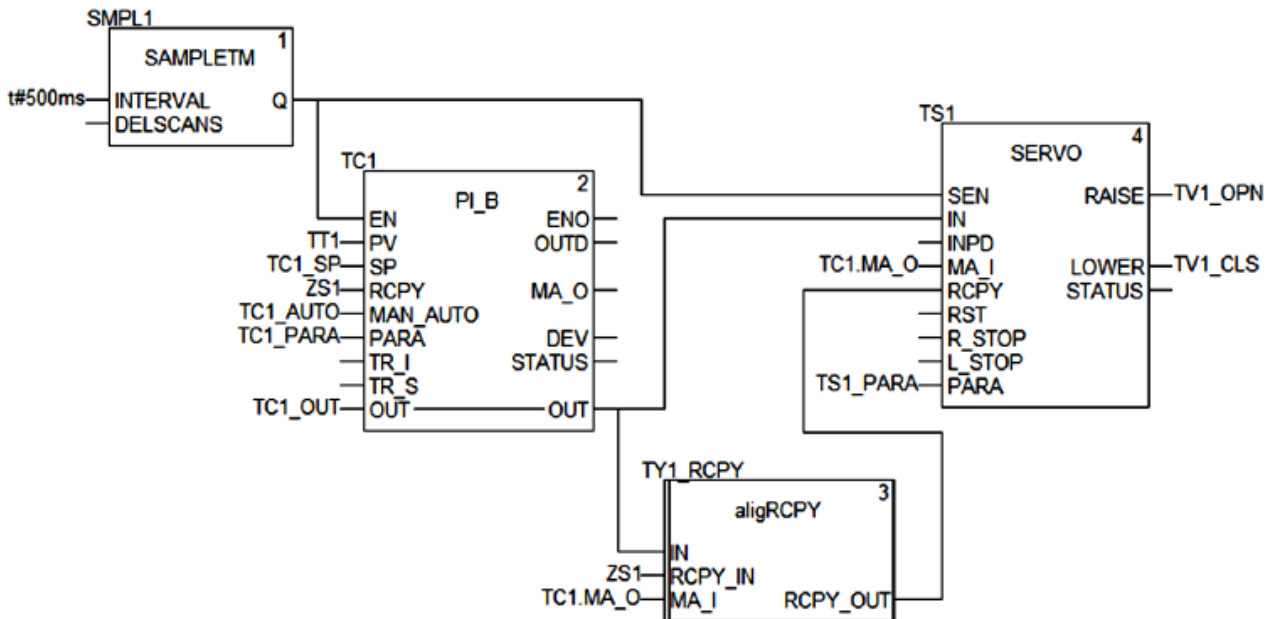


Рисунок 10.17– Секція CTRL1

Основу контуру складає ПІ-регулятор *TC1*, який на основі плинної температури *TT1* та завдання *TC1\_SP* формує на виході *OUT* (прив'язаний до *TC1\_OUT*) числове значення, яке подається на блок управління серводвигуном *TS1* для формування імпульсів "більше" та "менше". Регулятор *TC1* працює в режимі використання входу *RCPY* (*TC1\_PARA.en\_rcpy=TRUE*). Це значить, що нове значення виходу *OUT* регулятор буде розраховувати на базі значення входу *RCPY*, на який подається значення показчика положення *ZS1*.

Блок управління серводвигуном *TS1* теж працює в режимі використання входу *RCPY* (*TS1\_PARA.en\_rcpy=TRUE*). Це значить, що він буде перетворювати значення різниці *IN-RCPY* у дискретний сигнал *RAISE* або *LOWER* відповідної тривалості. Значення параметрів блоку дорівнюють *TS1\_PARA.t\_motor=t#10s* та *TS1\_PARA.t\_mini=t#250ms* відповідно до умов задачі.

Алгоритм роботи блоку *SERVO* працює таким чином, що у ручному режимі він буде видавати сигнали "більше" та "менше" до тих пір, поки *IN* та *RCPY* не будуть рівними. Для того, щоб блок управління серводвигуном в ручному режимі

припиняв управління в зоні наближеній до положення *RCPY*, можна створити та використати блок вирівнювання (рис.10.18). Принцип роботи алгоритму заключається в прирівнюванні виходу *RCPY\_OUT=IN* в тому випадку, коли в ручному режимі (*MA\_I=TRUE*) вхід *IN* буде в зоні наближення до *RCPY*. Величина зони наближення визначається параметром *deadb*, збільшення значення якого зменшує кількість рухів однак збільшує похибку позиціонування.

Name	no.	V...	Comment
<b>&lt;DFB&gt;</b>			
<b>&lt;inputs&gt;</b>			
IN	1	REAL	управляючий вихід на виконавчий механізм
RCPY_IN	2	REAL	положення виконавчого механізму
MA_I	3	BOOL	стан РУЧ/АВТ регулятора
<b>&lt;outputs&gt;</b>			
RCPY_OUT	3	REAL	вирівняне положення виконавчого механізму
<b>&lt;inputs/outputs&gt;</b>			
<b>&lt;public&gt;</b>			
deadb		REAL	0,25 зона нечутливості
<b>&lt;private&gt;</b>			
<b>&lt;sections&gt;</b>			
aligRCPY			

```

if ABS(IN-RCPY_IN)<deadb and NOT MA_I then
    RCPY_OUT:=IN;
ELSE
    RCPY_OUT:=RCPY_IN;
end_if;
    
```

Рисунок 10.18– Структура і програма DFB типу aligRCPY.

Блок вирівнювання *TYI\_RCPY* включається в схему між *TC1.OUT* та *SERVO.RCPY*. Тобто, коли *ZS1* знаходиться в зоні *TC1\_OUT*, на вхід *TS1.RCPY* буде подаватися значення *TC1\_OUT* а не *ZS1*, що приведе до відключення виходів LOWER та RAISE.

**Опис роботи програми реалізації контуру управління температурою в підігрівнику 2.** Програма секції "CTRL2" наведена на рис.10.19.

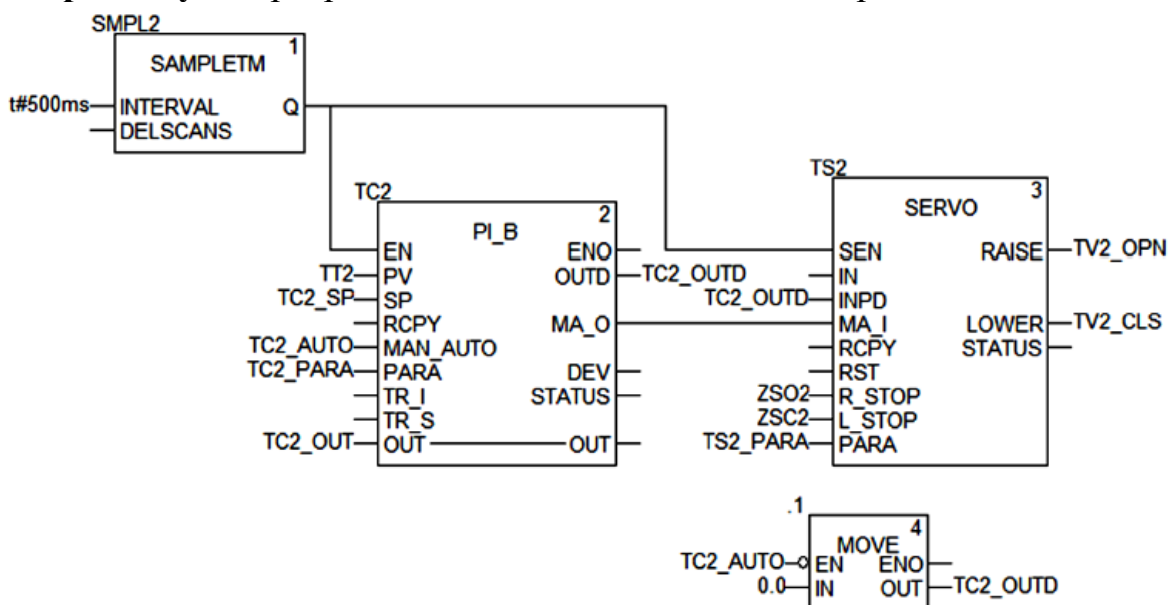


Рисунок 10.19– Секція CTRL2

Основу контуру складає ПІ-регулятор  $TC2$ , який на основі плинної температури  $TT2$  та завдання  $TC2\_SP$  формує на інкрементальному виході  $OUTD$  (прив'язаний до  $TC2\_OUTD$ ) числове значення, яке подається на блок управління серводвигуном  $TS2$  для формування імпульсів "більше" та "менше". Регулятор  $TC2$  працює в режимі без використання входу  $RCPY$  ( $TC1\_PARA.en\_rcpy=FALSE$ ), саме тому використовується зв'язка інкрементального виходу  $TC2.OUTD$  та  $TS2.INPD$ . Це значить, що на кожному циклі блок  $TC2$  буде розраховувати нове інкрементальне значення виходу  $OUTD$ , а абсолютне значення  $OUT$  використовуватися не буде.

Блок управління серводвигуном  $TS2$  теж працює в режимі без використання входу  $RCPY$  ( $TS1\_PARA.en\_rcpy=FALSE$ ). Це значить, що він буде перетворювати значення входу  $INPD$  у дискретний сигнал  $RAISE$  або  $LOWER$  відповідної тривалості. Значення параметрів блоку дорівнюють  $TS1\_PARA.t\_motor=t\#10s$  та  $TS1\_PARA.t\_mini=t\#250ms$  відповідно до умов задачі.

Алгоритм роботи блоку  $SERVO$  працює таким чином, що у ручному режимі він буде видавати сигнали "більше" та "менше" відповідно до значення входу  $INPD$  на кожному циклі. Враховуючи що змінна  $TC2\_OUTD$  обновлюється з періодичністю виклику  $TC2$  (500 мс), а блок  $TS2$  з кожним циклом, протягом 500 мс блок  $TS2$  буде формувати нові імпульси. Для того щоб уникнути цього ефекту, в ручному режимі ( $TC2\_AUTO=FALSE$ ) змінна  $TC2\_OUTD$  обнуляється після обробки контуру.

## **Програмування контролерів у складі SCADA-системи**

### **11. Технологія та математичне забезпечення АСУТП.**

#### **11.1 Технологія виробництва алюмінієвої катанки**

Технологічна лінія з виробництва алюмінієвої катанки зазвичай складається з пічної дільниці, ливарної машини, лінії транспортування литої заготовки, прокатного стана і моталок (рис.11.1). За допомогою скіпа завантажувальної машини 1 алюмінієві брикети подаються у газову плавильну піч шахтного типу 2. Після розплавлення метал прямує в одну з двох роздавальних печей 3, де він очищується від шлаку, піддається дегазації 4 та направляється у жолоб ливарного колеса 5, охопленого сталеву стрічкою. Під впливом охолодної води розплавлений алюміній кристалізується у виливниці колеса, що обертається, та у вигляді безперервного зливка подається у прокатний стан за допомогою правильного 6 і переднього тягового 7 пристроїв. У разі незадовільної якості лиття заготовку розрізають гідравлічними ножицями 8 на дрібні мірні шматки до тих пір, поки вона не відповідатиме встановленим вимогам.

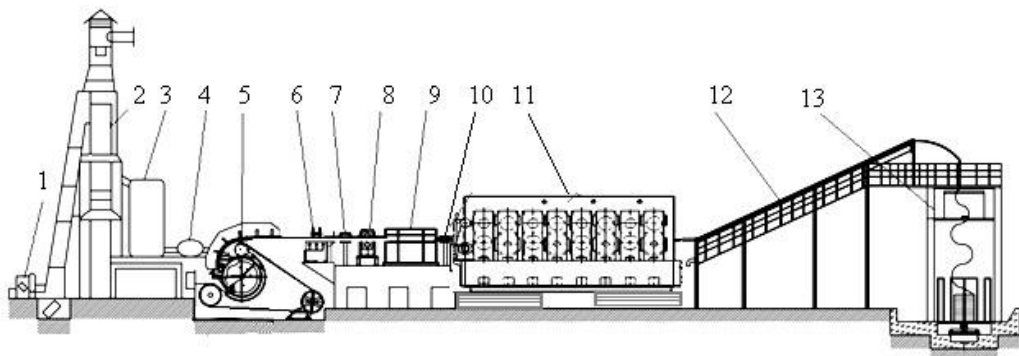


Рисунок 11.1 - Технологічна лінія неперервного лиття та прокатки

Під час виробництва дроту з алюмінієвих сплавів заготовку перед прокаткою додатково нагрівають індукційним нагрівачем 9 і за допомогою пристрою 10 подають у прокатний стан 11. Після прокатки катанка піддається процесам гартування, охолодження та сушіння 12 і подається на здвоєну моталку кошикового типу 13. Затискний тяговий пристрій разом з пристроєм утворення кілець укладає катанку в приймальний кошик, після наповнення якого відбувається заміна його на новий.

Відповідно до технології, для одержання високоякісної катанки необхідно підтримувати задану температуру розплаву у шахтній та роздавальних печах, забезпечити стабільні умови кристалізації рідкого металу у ливарному колесі, регулювати довжину алюмінієвої заготовки перед прокаткою, керувати охолодженням катанки під час прокатування у стані та процесом формування бунта у моталці. Оскільки властивості алюмінієвої катанки формуються протягом всього технологічного циклу, від плавильної печі до моталки, виконання усіх перелічених умов має відбуватися якісно. Тому до того, як розпочинати імітаційне моделювання АСУТП з виробництва алюмінієвої катанки, необхідно визначитися з математичними моделями об'єктів автоматизації, формулами та рівняннями, що описують зв'язок між окремими параметрами, а також алгоритмами керування технологічним процесом, за допомогою яких можна зрозуміти особливості виробництва катанки, створити системи автоматизації, дослідити вплив тих чи інших параметрів на якість регулювання і налагодити роботу ливарно-прокатного агрегату на бажаний кінцевий результат.

## 11.2. Математичне забезпечення АСУТП з виробництва алюмінієвої катанки

Основними вихідними величинами, що характеризують процес неперервного лиття та прокатки є температура рідкого алюмінію в шахтній та роздавальних печах, температура та довжина заготовки на виході з ливарного колеса, температура катанки на виході з прокатного стану та після гартування і охолодження. Для побудови систем автоматичного регулювання динамічні характеристики усіх перелічених параметрів апроксимовані аперіодичними ланками з транспортним запізнюванням:

$$W_{(p)} = \frac{K}{T \cdot p + 1} \cdot e^{-p\tau}, \quad (11.1)$$

де  $K$  – коефіцієнт передачі, вихідна величина / % ходу регулювального органа;  $T$  – стала часу, с;  $\tau$  – транспортне запізнювання, с.

Коефіцієнти зазначених об'єктів автоматизації приведені у таблиці 11.1.

Зазвичай температура розплаву у шахтних та роздавальних печах регулюється шляхом коригування витрати спалюваного газу. А охолодження заготовки під час прокатування її у стані та охолодження катанки у процесі гартування здійснюється шляхом керування витратою охолодної речовини. Оскільки на згаданих об'єктах автоматизації значення вихідних величин визначаються виключно величинами вхідних параметрів, при імітаційному моделюванні АСУТП були використані одноконтурні системи автоматичного регулювання.

Таблиця 11.1. Коефіцієнти об'єктів автоматизації

Пара- метри  Коефі- цієнти	Темпера- тура рідко- го алю- мінію у шахт- ній печі	Темпера- тура рідко- го алю- мінію у роздава- льній печі	Темпера- тура заготов- ки після ливарно- го колеса	Темпер- атура катан- ки на виході з прокат- ного стана	Темпер- атура катан- ки після гартува- ння
$K$	2,7	15	1	2	1.5
$T$	32	25	15	15	15
$\tau$	13	10	7	4	7

Разом з цим температура та довжина заготовки на виході з ливарного колеса, залежать від багатьох параметрів, серед яких швидкість лиття, температура розплаву в роздавальній печі, витрата води на охолодження та ін. Так само процес формування бунта катанки залежить від декількох параметрів: діаметра катанки, кроку укладання витків в шарі, внутрішнього та зовнішнього радіусів бунта. Тому, аби забезпечити якісне керування вказаними параметрами та можливість проведення досліджень технологічного процесу на імітаційній моделі, необхідно при моделюванні АСУТП використовувати більш складні алгоритми керування, які б базувались на математичних формулах і рівняннях, що визначають зв'язок між регульованими параметрами та тими, що впливають на їх величину.

### 11.2.1. Охолодження зливка та стабілізація теплового режиму кристалізатора ливарної машини

Якість алюмінієвої заготовки і продуктивність ливарно-прокатного агрегату в значній мірі залежать від роботи системи охолодження зливка, головною метою якої є безперервне і поступове видалення тепла в процесі перетворення розплавленого алюмінію в тверду заготовку. Відбувається це за рахунок подачі води в певні зони охолодження через регульовані форсунки, що розташовані по

периметру та довжині виливниці. При цьому загальна витрата води і розподіл її по зонах повинні бути такими, щоб фронт кристалізації металу у виливниці мав форму латинської літери "V", а твердіння зливка закінчувалося у момент видалення його з ливарного колеса. Проте в технічних документаціях на технологічні лінії безперервного лиття та прокатки алюмінію виробники не дають конкретних відомостей з налагодження систем охолодження, що суттєво ускладнює запуск ливарно-прокатних агрегатів у роботу. У зв'язку з цим виникає необхідність у розробленні рекомендацій щодо визначення загальної витрати води на охолодження зливка і алгоритму розподілу її між окремими зонами кристалізатора. Задля цього проведено аналіз теплового балансу кристалізатора машини неперервного лиття алюмінію:

$$Q + Q_{кр} = Q_{вод} + Q_{вил} + Q_{стс} + Q_{стд} + Q_{заз} + Q_{заб} + Q_{дов} + Q_{заг}, \quad (11.2)$$

де  $Q$  – загальна кількість теплоти, яка підводиться з рідким алюмінієм у виливницю ливарного колеса;  $Q_{кр}$  – теплота, що виділяється при кристалізації зливка;  $Q_{вод}$  – кількість теплоти, що втрачається з охолодною водою,  $Q_{вил}$  – кількість теплоти, що витрачається на нагрів виливниці;  $Q_{стс}$  – кількість теплоти, що витрачається на нагрів сталевої стрічки, що охоплює мідну виливницю;  $Q_{стд}$  – кількість теплоти, що витрачається на нагрів сталевих дисків ливарного колеса;  $Q_{заз}$  – кількість теплоти, що витрачається на нагрів газового зазору між стінками виливниці і зливком;  $Q_{заб}$  – втрата теплоти на термічних опорах забруднень (накипу);  $Q_{дов}$  – втрата теплоти в довкілля;  $Q_{заг}$  – кількість теплоти, що виводиться з кристалізатора із затверділою заготовкою.

З рівняння видно, якщо розрахувати приходну частину теплового балансу, кількість теплоти, яка виводиться з кристалізатора затверділою заготовкою при заданих температурі та швидкості ливарного колеса, кількість теплоти, яка витрачається на нагрів виливниці, газового зазору та термічних опорів забруднень, а також втрату теплоти в навколишнє середовище, тоді розв'язуючи рівняння теплового балансу щодо  $Q_{вод}$ , можна за її величиною знайти витрату води на охолодження зливка:

$$G_{кр} = \frac{Q_{вод}}{c_g \cdot \Delta T_g}, \quad (11.3)$$

де  $G_{кр}$  – витрата води, яка подається в кристалізатор, кг/с;  $c_g$  – теплоємність води, Дж/(кг·К);  $\Delta T_g$  – різниця між температурою води, що подається  $T_{вх}$  у кристалізатор та відводиться  $T_{вих}$  від нього, К.

Оскільки інтенсивність тепловідведення від розплаву залежить не тільки від витрати води на охолодження, а і від умов теплообміну в кристалізаторі, для забезпечення рівномірного твердіння рідкого металу за поперечним перерізом і довжиною зливка, витрата води в окремі зони кристалізатора має відбуватися з урахуванням коефіцієнтів теплопередачі його багаточислової стінки в цих зонах.

Для визначення алгоритму розподілу витрати води між окремими зонами кристалізатора припустимо, що коефіцієнти теплопередачі за напрямками охолодження в окремих зонах однакові і дорівнюють середньоарифметичному  $K_{cp} = K_{сум} / n$ . Тоді відсоток витрати води на кожну зону охолодження від загального обсягу, що подається в кристалізатор, складе:

$$G = \frac{K_{cp}}{K_{сум}} \cdot 100\% , \quad (11.4)$$

де  $K_{сум} = \sum_1^n K_i$  – сумарний коефіцієнт теплопередачі багатозонаної стінки кристалізатора за всіма зонами охолодження.

Якщо коефіцієнт теплопередачі  $K1$  більше  $K_{cp}$  на  $\Delta 1\%$ , тоді реальна витрата води у першу зону охолодження має бути менше за уявну на  $\Delta 1\%$ :

$$G1 = G - \Delta 1, \%$$

Якщо коефіцієнт теплопередачі  $K2$  менше  $K_{cp}$  на  $\Delta 2\%$ , тоді реальна витрата води у другу зону має бути більше за уявну на  $\Delta 2\%$ :

$$G2 = G + \Delta 2, \%$$

І так за всіма коефіцієнтами теплопередачі.

Виходячи з цього, для рівномірного твердіння металу у роторній ливарній машині за поперечним перерізом та довжиною зливка, необхідно визначити зони охолодження кристалізатора та коефіцієнти теплопередачі його багатозонаної стінки за напрямками охолодження у кожній з них і воду, що подається у кристалізатор, розподілити поміж визначеними зонами охолодження, відповідно до рівняння:

$$G_i = \frac{K_{cp}}{K_{сум}} \cdot 100\% - \frac{K_i - K_{cp}}{K_{cp}} \cdot 100\% = \left( 1 + \frac{1}{n} - \frac{K_i}{K_{cp}} \right) \cdot 100\% , \quad (11.5)$$

де  $G_i$  – витрата води на охолодження в  $i$ -ту зону кристалізатора;  $n$  – кількість зон охолодження;  $K_i$  – коефіцієнт теплопередачі багатозонаної стінки кристалізатора за напрямком охолодження в  $i$ -тій зоні;  $K_{cp}$  – середній коефіцієнт теплопередачі багатозонаної стінки кристалізатора за всіма зонами охолодження.

Розроблений алгоритм розподілу витрати води між окремими зонами кристалізатора був перевірений на прикладі роботи роторної ливарної машини ЛПА «LY-1600» китайського виробництва, на якій виготовляється алюмінієва заготовка. Було вибрано три зони охолодження за напрямками: «розплав-стінка виливниці», «розплав-сталева стрічка» і «розплав-сталеві диски».

За результатами розрахунків коефіцієнти теплопередачі багатозонаної стінки кристалізатора по зонах склали:

- напрям «розплав-стінка виливниці» –  $K_1 = 776 \text{ Вт}/(\text{м}^2 \cdot \text{К})$ ;
- напрям «розплав-сталева стрічка» –  $K_2 = 817 \text{ Вт}/(\text{м}^2 \cdot \text{К})$ ;
- напрям «розплав-сталеві диски» –  $K_3 = 660 \text{ Вт}/(\text{м}^2 \cdot \text{К})$ .

При таких значеннях сумарний коефіцієнт теплопередачі за всіма зонами охолодження дорівнює  $K_{\text{сум}}=2253\text{Вт}/(\text{м}^2\cdot\text{К})$ , а середньоарифметичний –  $K_{\text{ср}}=751\text{Вт}/(\text{м}^2\cdot\text{К})$ .

Для забезпечення рівномірного твердіння зливка при  $K_{\text{ср}}$  за всіма зонами охолодження, відсоток витрати води у кожній зоні складе:

$$G1=G2=G3= 100\% / 3 = 33,3\%$$

Оскільки  $K1=776\text{Вт}/(\text{м}^2\cdot\text{К})$  більше  $K_{\text{ср}}=751\text{Вт}/(\text{м}^2\cdot\text{К})$  на 3,3%, то витрата води для реальних умов теплообміну у зоні з напрямом охолодження «розплав-стінка виливниці» має бути:

$$G1=33,3\% - 3,3\% = 30\%$$

Оскільки  $K2=817\text{Вт}/(\text{м}^2\cdot\text{К})$  більше  $K_{\text{ср}}=751\text{Вт}/(\text{м}^2\cdot\text{К})$  на 8,8%, то витрата води для реальних умов теплообміну у зоні з напрямом «розплав-сталева стрічка» має бути:

$$G2=33,3\% - 8,8\% = 24,5\%$$

Оскільки  $K3=660\text{Вт}/(\text{м}^2\cdot\text{К})$  менше  $K_{\text{ср}}=751\text{Вт}/(\text{м}^2\cdot\text{К})$  на 12,1%, то витрата води для реальних умов теплообміну у зоні з напрямом «розплав-сталеві диски» має бути:

$$G3=33,3\% + 12,1\% = 45,4\%.$$

Розрахунок температур за поперечним перерізом зливка при знайдених відсотках розподілу витрати води поміж зонами охолодження проводився в інструментальній системі Comsol 3.5a на підставі розробленої математичної моделі. В результаті розв'язання нестационарної задачі теплопровідності отримано картини розподілу температурного поля в алюмінієвому зливку на протязі його кристалізації, які підтверджують рівномірність твердіння металу за поперечним перерізом і довжиною зливка.

Таким чином, завдяки розподілу витрати води поміж зонами охолодження з урахуванням коефіцієнтів теплопередачі багатошарової стінки кристалізатора за напрямками охолодження в окремих його зонах, забезпечується однакова швидкість кристалізації та рівномірне твердіння розплаву з усіх боків зливка на протязі всього часу кристалізації.

Після визначення загальної витрати води на охолодження зливка, а також розподілу її за зонами кристалізатора, стабілізувати його тепловий режим за наявності збурень можна тільки з використанням системи автоматичного регулювання.

Зазвичай на подібних агрегатах стабілізацію теплового режиму реалізують за рахунок коригування витрати води, що подають на охолодження зливка. Проте недоліком таких систем автоматизації є те, що регулювання теплового режиму кристалізатора здійснюють не за прямими показниками його стану, а шляхом регулювання охолодження зливка за непрямими показниками – температурою води до та після охолодження. Тому, для підвищення якості регулювання бажано використовувати у системі автоматизації параметр, що напряму характеризує тепловий режим кристалізатора.

Виходячи з рівняння теплового балансу, таким параметром можна вважати теплоту, що виводиться з кристалізатора із затверділою заготовкою  $Q_{\text{заг}}$ :



$$Q_{zag} = \rho \frac{V}{60} \cdot F \cdot h_{roz} - G_{roz} \cdot (h_{roz} - h_{zag}), \quad (11.6)$$

де  $G_{\delta i_c} \cdot (h_{\delta i_c} - h_{\zeta \grave{a} \bar{a}})$  – теплота, що віддається гарячим теплоносієм, Вт;  $G_{\delta i_c}$  – витрата розплаву, що визначається продуктивністю агрегату та швидкістю лиття, кг/с;  $h_{\delta i_c} = L + c_i \cdot T_{\delta i_c}$  – ентальпія сплаву на початку виливниці, Дж/кг;  $h_{\zeta \grave{a} \bar{a}} = L + c_e \cdot T_{\zeta \grave{a} \bar{a}}$  – ентальпія сплаву у кінці виливниці, Дж/кг;  $T_{roz}$  – початкова та  $T_{zag}$  – кінцева температура алюмінієвого сплаву у виливниці, °С;  $c_n$ ,  $c_k$  – теплоємність алюмінієвого сплаву, відповідно, для початкової та кінцевої температур, Дж/(кг·°С).

При цьому, якщо в якості допоміжного параметра використовувати кількість теплоти, що втрачається з охолодною водою  $Q_{вод}$ :

$$Q_{вод} = c_g \cdot G_g \cdot \Delta T_g \quad (11.7)$$

тоді можна реалізувати систему автоматичного регулювання теплового режиму кристалізатора, яка буде коригувати витрату води на охолодження зливка не тільки за відхиленням основного параметру  $Q_{zag}$ , а і за збуреннями, що надходять з боку  $Q_{вод}$ . На рис.5.2 приведено структурну схему системи автоматичного регулювання теплового режиму кристалізатора.

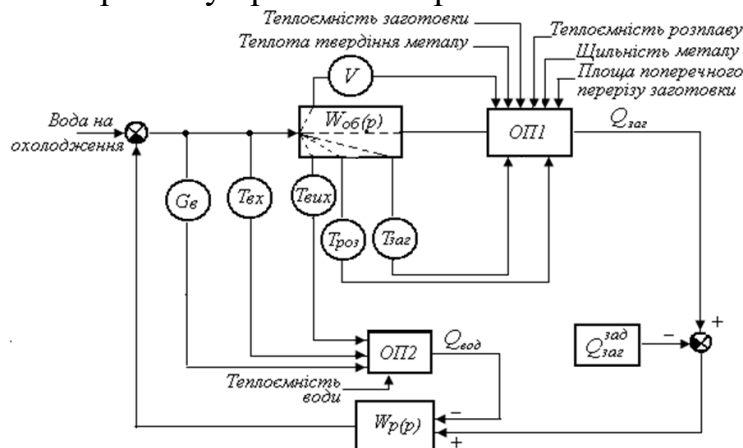


Рисунок 11.2 – Структурна схема системи автоматичного регулювання теплового режиму кристалізатора

На підставі інформації з датчиків швидкості лиття  $V$ , температури розплаву  $T_{roz}$  та температури затверділої заготовки  $T_{zag}$  з урахуванням відомих значень решти параметрів, обчислювальний пристрій ОП1 за формулою (11.6) обчислює поточне значення теплоти, що виводиться заготовкою з кристалізатора  $Q_{zag}$ , яку порівнюють з розрахованим заздалегідь заданим значенням  $Q_{zag}^{зад}$  для встановленої швидкості лиття та теплофізичних характеристик розплаву, що кристалізують. Сигнал, пропорційний різниці цих величин прямує на перший вхід регулятора  $Wp(p)$ . Під час не збігання розрахованих параметрів  $Q_{zag}$  і  $Q_{zag}^{зад}$  регулятор змінює подавання води на охолодження зливка до повної компенсації розбалансу на його вході.

Під час змінювання вхідних параметрів  $G_b$ ,  $T_{вх}$  або  $T_{вих}$ , що впливають на

ступінь охолодження зливка, у системі регулювання з'являється збурення з боку теплоти, що втрачають з охолодною водою  $Q_{вод}$ . Обчислювальний пристрій ОП2 за формулою (11.7) обчислює поточну величину  $Q_{вод}$  і подає сигнал пропорційний цьому параметру на другий вхід регулятора  $Wp(p)$ , що здійснює випереджувальне коригування витрати охолодної води у бік компенсації збурення  $Q_{вод}$ . Оскільки запізнювання за каналом регулювання  $Q_{вод}$  на порядок менше запізнювання за каналом регулювання  $Q_{заг}$ , використання інформації про теплоту, яку втрачають з охолодною водою, підвищить якість регулювання, що призведе до стабілізації процесу кристалізації та покращенню структури заготовки.

## 11.2.2. Формування заготовки та бунта готової катанки

### 11.2.2.1 Алгоритм регулювання довжини заготовки

В більшості випадків регулювання довжини заготовки здійснюється одноконтурною САР, яка реалізує ПІ-закон. Проте така система не забезпечує необхідну якість регулювання. Тому для імітаційного моделювання АСУТП виробництва алюмінієвої катанки застосована комбінована система автоматичного регулювання, рис.112, , яка на підставі математичної моделі:

$$L = L_{заг} + \left( V_{Л} \cdot c_1 (T_{вих} - T_{вх}) + V_{Л} - \frac{c_2 n}{\lambda} \right) \cdot \Delta t, \quad (11.8)$$

де  $L$ - загальна довжина алюмінієвої заготовки, м;  $L_{заг} = V_{Л} \cdot \Delta t$  – базова довжина отриманого зливка за цикл лиття, м;  $V_{Л}$  – лінійна швидкість заготовки на виході з ливарного колеса, м/с;  $c_1$ – коефіцієнт температурного подовження алюмінію;  $T_{вих}$ – температура зливка на виході з ливарного колеса, К;  $T_{вх}$  – температура металу на вході в ливарне колесо, К;  $c_2$ – коефіцієнт передачі швидкості прокатного стана;  $n$  – швидкість обертання електропривода;  $\lambda$  - коефіцієнт витяжки прокатного стана;  $\Delta t$  – цикл лиття, с;

працює не тільки за довжиною заготовки  $L$ , а і за збуреннями, які виникають при відхиленнях лінійної швидкості заготовки  $V_{Л}$  на виході з ливарного колеса, температур металу на вході  $T_{вх}$  і виході  $T_{вих}$  з ливарного колеса, а також коефіцієнта витяжки прокатного стан  $\lambda$ .

Вираз у дужках цього рівняння визначається нестабільністю роботи ливарного колеса й прокатного стана, а тому характеризує величину прирощення заготовки  $\Delta L$  під впливом збурень, діючих в ЛПА.

### 11.2.2.2 Алгоритм керування роботою моталки

Аби метал, який безперервно прокатують, набував не лише компактної форми, а і заданих властивостей за всією його довжиною, при створенні імітаційної моделі АСУТП використано спосіб, за допомогою якого можна коригувати швидкість виткоутворювача моталки під час формування бунта на кожному його шарі з урахуванням діаметра змотуваної катанки та бажаного кроку укладання витків. При цьому швидкість для  $i$ -того витка змінюється ступінчасто, відповідно до виразу:

$$\omega_i = \frac{V}{R_{\max(\min)} \mp (i-1)d \cdot \alpha}, \quad (11.9)$$

а тривалість керування, яка дорівнює часу формування  $i$ -того витка, визначається із співвідношення:

$$T_i = \frac{2\pi}{\omega_i}, \quad (11.10)$$

де  $\omega_i$  – кутова швидкість обертання проводки, рад/с;  $V$  – швидкість подачі катанки, м/с;  $R_{\max}$  – радіус максимального витка, м;  $R_{\min}$  – радіус мінімального витка, м;  $d$  – діаметр катанки, м;  $1 \leq i \leq n$ ,  $n = \frac{R_{\max} - R_{\min}}{d \cdot \alpha}$  – кількість витків у шарі,  $\alpha$  – крок укладання витків,  $1 \leq \alpha \leq \frac{R_{\max} - R_{\min}}{2d}$ , « $R_{\max}$  –» – при укладанні шару від периферії до центру, « $R_{\min}$ +» – при укладанні шару від центру до периферії.

Як видно з наведених формул, рівнянь та алгоритмів керування вони відображають зв'язки між окремими технологічними параметрами і відповідають вимогам, висунутим до математичного забезпечення створюваної імітаційної моделі АСУТП.

### 11.3 Програмна реалізація імітаційної моделі АСУТП з виробництва алюмінієвої катанки.

Імітаційна модель технологічної лінії неперервного лиття і прокатки алюмінієвої катанки розроблена на одній обчислювальній машині з використанням пакетів програмування Unity Pro виробництва Schneider Electric і TIA Portal компанії Siemens. Зв'язок між цими пакетами було створено за допомогою відповідних налаштувань у кожному з них. В Unity Pro за допомогою мов FBD і ST реалізовані програми моделювання та управління роботою шахтної та роздавальної печі, ливарного колеса, кристалізатора і прокатного стану, моталки і гартівного пристрою, а також програма управління станом об'єктів візуалізації. В пакеті TIA Portal створено чотири екрани візуалізації: стартовий, шахтної та роздавальної печей, ливарної машини і прокатного стану, гартівного пристрою та моталки. Як додаток до них розроблено дев'ять екранів відображення трендів основних технологічних параметрів і установлення налаштувань характеристик відповідних агрегатів і засобів автоматизації.

Всього у проекті задекларовано 314 змінних. З них 6 – типу ARRAY OF REAL, 176 – типу BOOL, 9 – типу INT, 123 – типу REAL, 64 – символічні і 250 – локалізовані.

Архітектура імітаційної моделі АСУТП з виробництва алюмінієвої катанки на ливарно-прокатному агрегаті приведена на рисунку 11.3. Основою для спілкування людини з імітаційною моделлю є екран ЕОМ, в якому за допомогою миші та меню можна вибрати три базових вікна візуалізації окремих ділянок технологічної лінії. В кожному з цих вікон зображені відповідні агрегати та механізми з технологічними лініями зв'язку, а також пульти керування з

кнопками вибору режимів роботи, ручного або автоматичного, та засобами контролю за технологічними параметрами.

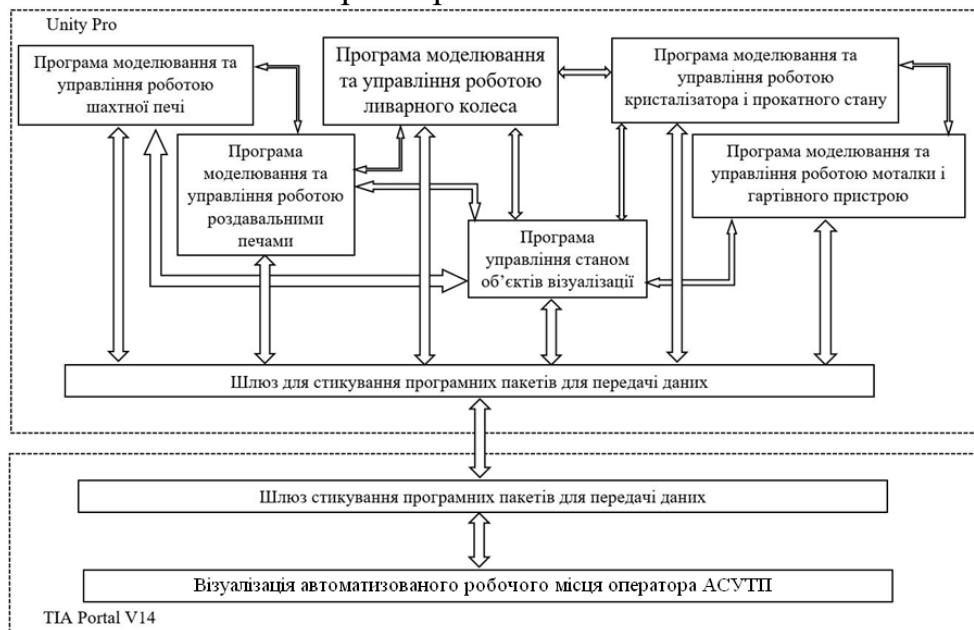


Рисунок 11.3 – Архітектура імітаційної моделі АСУТП виробництва алюмінієвої катанки

Для налагодження та дослідження роботи технологічної лінії в кожному з вікон можна викликати панель коригування коефіцієнтів окремих об'єктів автоматизації, налаштування параметрів регулятора та активізації збурень, що впливають на хід технологічного процесу. Результат досліджень представляється у вигляді графіків зміни контрольованих величин.

У першому вікні імітаційної моделі зображені шахтна та дві роздавальні печі з відповідними технологічними лініями зв'язку, а також пульти керування з кнопками вибору режимів роботи, ручного керування та засобами контролю за рівнем і температурою металу в печах (рис. 11.4). Програмою передбачено завантаження шахтної печі алюмінієвим ломом, увімкнення її в роботу, ручне і автоматичне регулювання температури розплаву на рівні 750 °С, зливання його в одну з роздавальних печей з автоматичною стабілізацією температури (700°С) та подачу розплавленого алюмінію у ливарну машину. Рівень металу в печах відслідковується синім кольором.

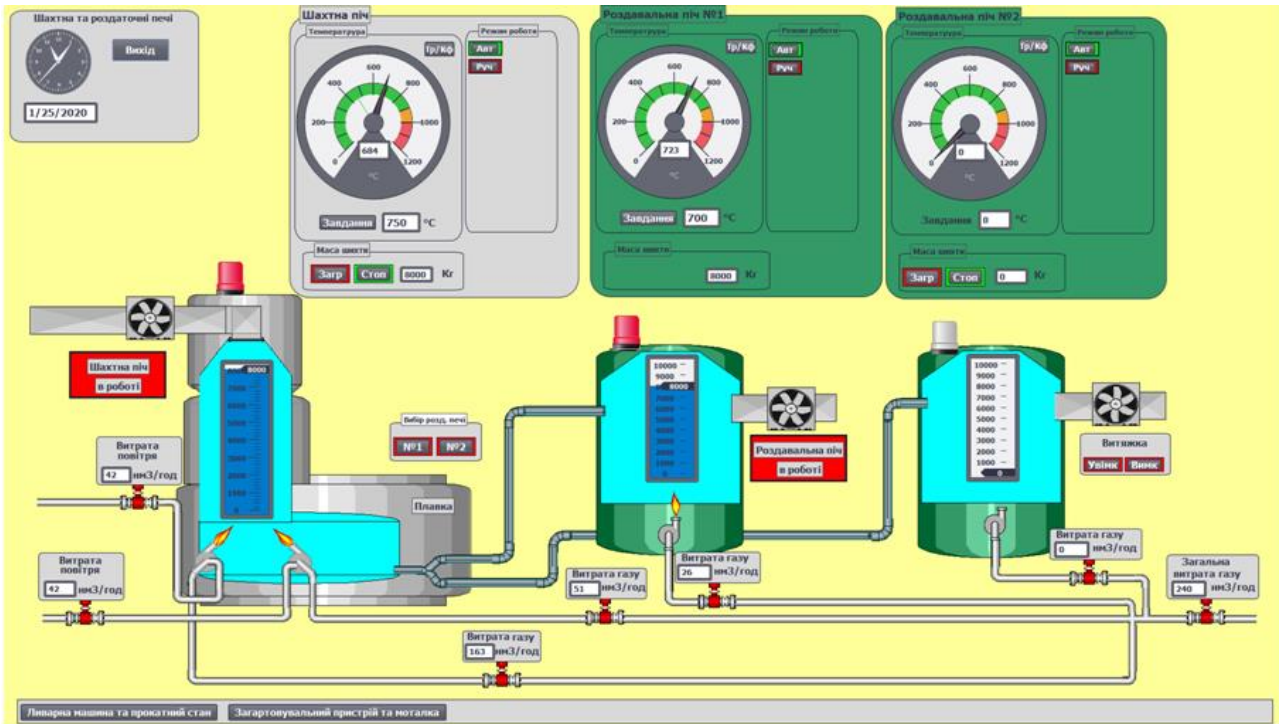


Рисунок 11.4 – Вікно візуалізації пічної ділянки

Робочий режим печей попереджається відповідним повідомленням на червоному фоні під вентиляторами, а біля пальників з'являється полум'я. Після злиття розплаву печі зупиняють, попередження про їх робочий стан зникає, а на його місці з'являються кнопки керування роботою вентиляторів, які призначені для провітрювання внутрішнього простору печей.

Програма моделювання роботи шахтної печі приведена на рисунку 11.5.

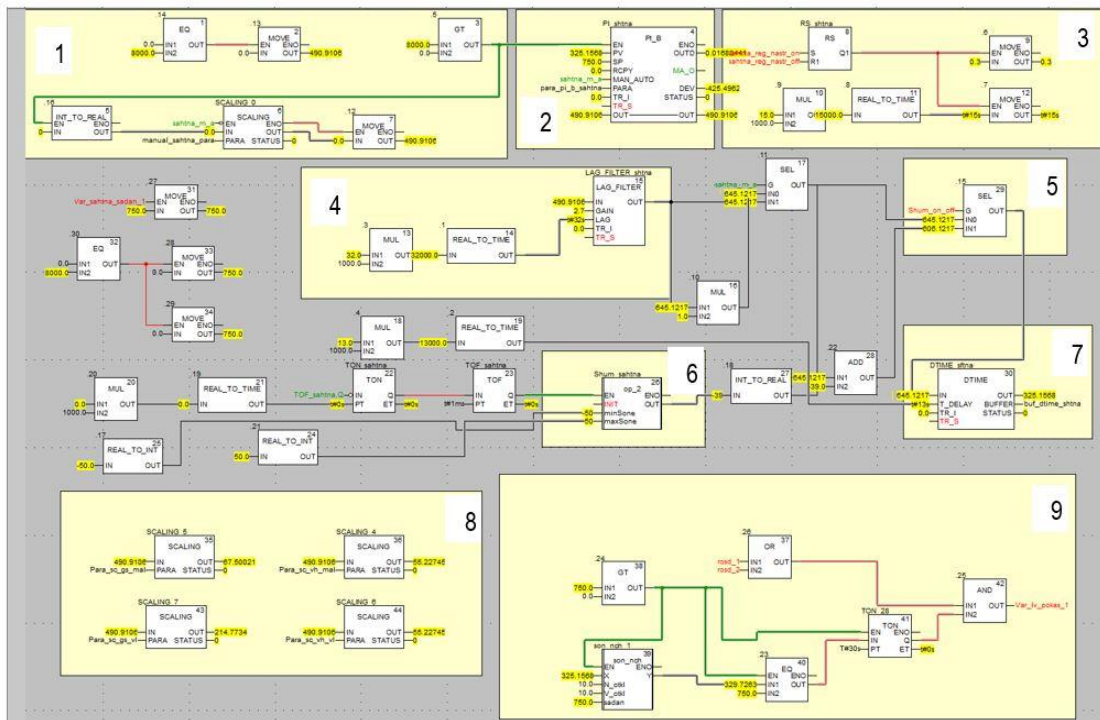


Рисунок 11.5 – Програма моделювання роботи та анімації шахтної печі

Функціональний блок 1 реалізує пуск програми в роботу, блоками 2 і 3 запрограмовані ПІ регулятор і алгоритм керування його настройками. Інерційною

ланкою 4 і транспортним запізнюванням 7 представлена шахтна піч. Формування збурень, що випадково з'являються у системі регулювання, реалізовано у блоці 6, а у фрагменті 8 вихідні сигнали регуляторів перетворюється у реальну витрату газу та повітря, що надходять до пальників. Дозвіл на злив розплаву з шахтної печі реалізовано у блоці 9.

Панель установлення параметрів об'єкта, настройок регулятора та активізації збурень, що випадково з'являються у системах автоматизації, показана на рисунку 11.6.

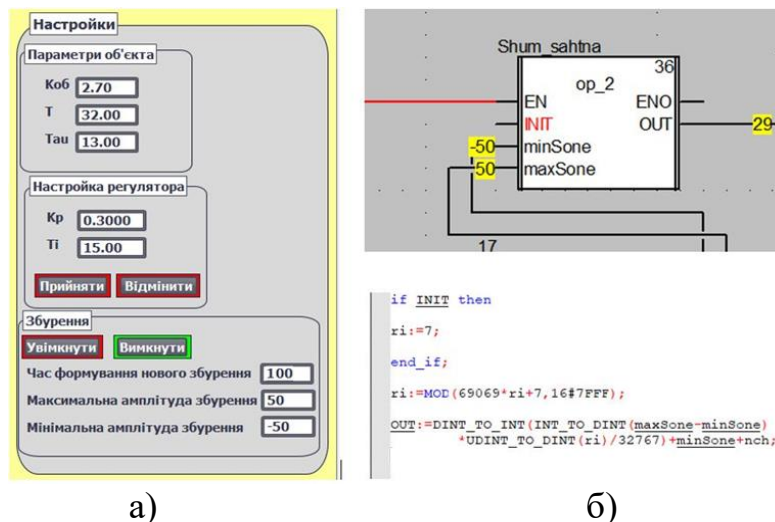


Рисунок 11.6 – Панель настроювання системи автоматичного регулювання (а) та програмний блок формування випадкових збурень (б)

У другому вікні показані ливарна машина із заготовкою та прокатний стан з пультами керування роботою агрегатів і приладами, що показують значення регульованих параметрів: теплоти, яка виноситься заготовкою з ливарної машини і температури заготовки до та після прокатного стану (рис. 11.7). Ливарна машина і прокатний стан запускаються в роботу в ручному режимі. При цьому ливарне колесо на екрані розпочинає мигати, а фоновий колір прокатного стану змінюється з червоного на зелений. На виході ливарного колеса з'являється заготовка, яка за певною траєкторією збільшується у розмірі. На початковому етапі роботи ливарної машини якість заготовки не задовільна, тому при зростанні її довжини до 10 м, яка відображається у круглому вікні під петлею заготовки, гідравлічні ножиці відрізають браковану частину. Для охолодження зливка у ливарній машині та заготовки у прокатному стані, під цими агрегатами показані ємності з водою та емульсією, рівень яких у синьому кольорі підтримується в заданих межах.

Програмне забезпечення цього вікна дозволяє в реальному масштабі часу здійснити з анімацією вмикання в роботу ливарної машини та прокатного стану, керування їх швидкістю в ручному режимі, відслідковування вторинними приладами значень контрольованих і регульованих параметрів, формування траєкторії руху катанки від кристалізатора до прокатного стану, відрізання гідравлічними ножицями бракованої частини заготовки та автоматичне регулювання теплового режиму кристалізатора, довжини заготовки і температури катанки після прокатного стану.

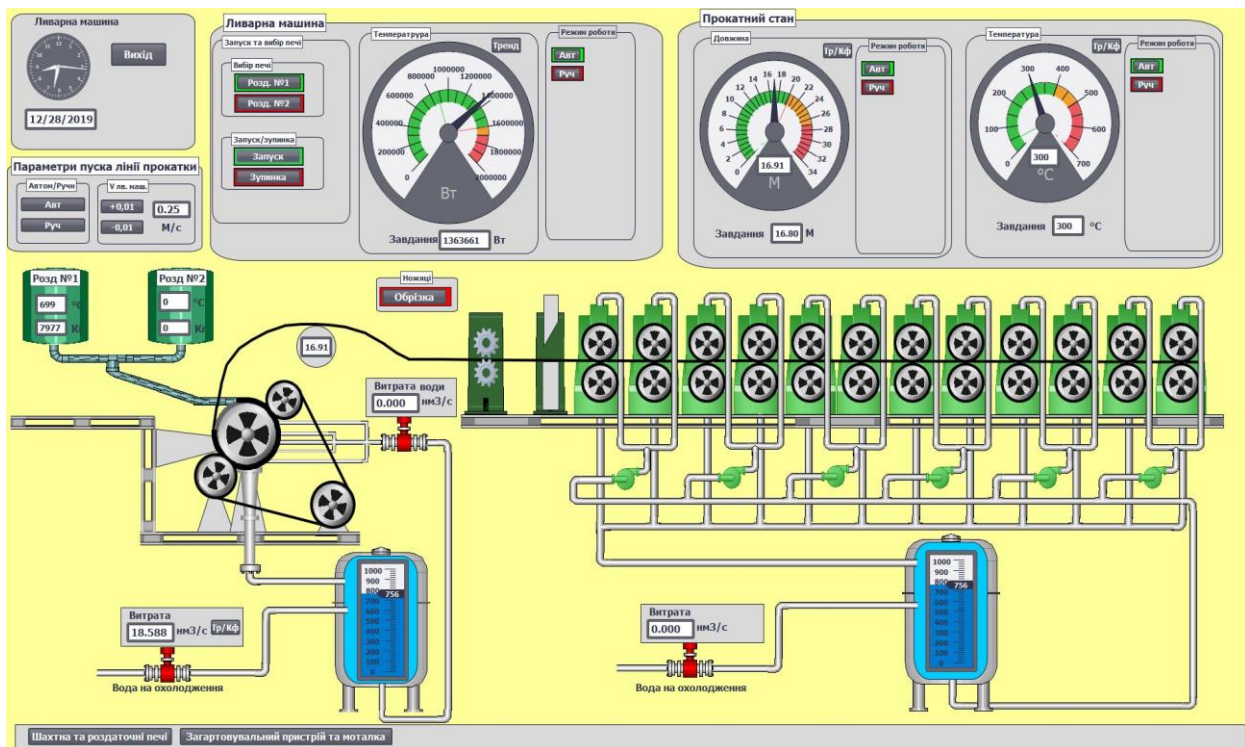


Рисунок 11.7 – Вікно візуалізації ливарної машини та прокатного стану

Програма регулювання теплового режиму кристалізатора за розрахованими значеннями кількості теплоти, що виноситься із заготовкою з кристалізатора, і кількості теплоти, що втрачається з охолодною водою, приведена на рисунку 11.8.

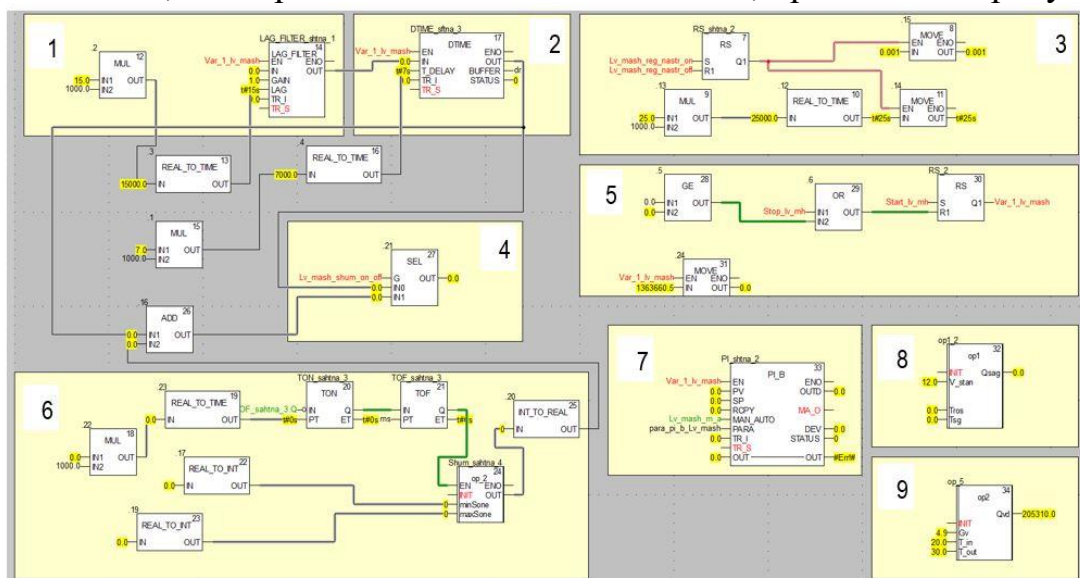


Рисунок 11.8 – Програма регулювання теплового режиму кристалізатора

Програма включає 9 основних фрагментів: моделі кристалізатора 1 і транспортного запізнення 2, блоки формування 6 і увімкнення 4 збурень, блоки розрахунку кількості теплоти, що виноситься із заготовкою з кристалізатора 8, і кількості теплоти, що втрачається з охолодною водою 9, та блоки ПІ-регулятора 3 і зміни його налаштувань 7.

Функціональний блок розрахунку кількості теплоти, що виноситься із заготовкою з кристалізатора за формулою (6) приведено на рисунку 11.9.

op1		<DFB>	
<b>&lt;inputs&gt;</b>			
INIT	1	BOOL	
V_stan	2	REAL	
Tros	5	REAL	
Tsg	6	REAL	
<b>&lt;outputs&gt;</b>			
Qsag	1	REAL	
<b>&lt;inputs/outputs&gt;</b>			
<b>&lt;public&gt;</b>			
C_sg		REAL	1027.0
L		REAL	362300
C_met		REAL	1250.0
p_met		REAL	2700.0
F		REAL	0.0023
G_ros		REAL	1.26

```

if V_stan>0.0 and Tros>0.0 and Tsg>10.0 and init then
Qsag:=((p_met*V_stan)/60.0)*F*(L+C_met*(Tros+273.0))-
(G_ros*((L+C_met*(Tros+273.0))-(L+C_sg*(Tsg+273.0))));
else
Qsag:=0.0;
end_if;

```

а)

б)

Рисунок 11.9 – Оголошені змінні (а) та код блока розрахунку кількості теплоти, що виноситься із заготовкою з кристалізатора (б)

Функціональний блок розрахунку кількості теплоти, що втрачається з охолодною водою, за формулою (7) приведено на рисунку 11.10.

op2		<DFB>	
<b>&lt;inputs&gt;</b>			
INIT	2	BOOL	
Gv	3	REAL	
T_in	4	REAL	
T_out	5	REAL	
<b>&lt;outputs&gt;</b>			
Qvd	1	REAL	
<b>&lt;inputs/outputs&gt;</b>			
<b>&lt;public&gt;</b>			
C_vd		REAL	4190.0

```

if INIT then
Qvd:=0.0;
end_if;
if Gv>0.0 and T_in>0.0 and T_out>0.0 then
Qvd:=C_vd*Gv*abs(T_in-T_out);
end_if;

```

а)

б)

Рисунок 11.10 – Оголошені змінні (а) та код (б) блока розрахунку кількості теплоти, що втрачається з охолодною водою

На рисунку 11.11 показана панель настроювання САР теплового режиму кристалізатора і графік перехідного процесу після встановлення заданого значення кількості теплоти, що виноситься із заготовкою з кристалізатора, при появі випадкових збурень та після зменшення завдання.



а)

б)

Рисунок 11.11 – Панель настроювання САР теплового режиму кристалізатора (а) та графік перехідного процесу при зміні завдання та під впливом випадкових збурень (б)



Програма імітації роботи системи автоматичного регулювання довжини заготовки приведена на рисунку 11.12. Основними у її структурі є функціональні блоки, що формують петлю заготовки та розраховують її довжину між ливарним колесом і прокатним станом 2, моделюють ПІ-регулятор 1 і надають можливість коригувати його настройки 3.

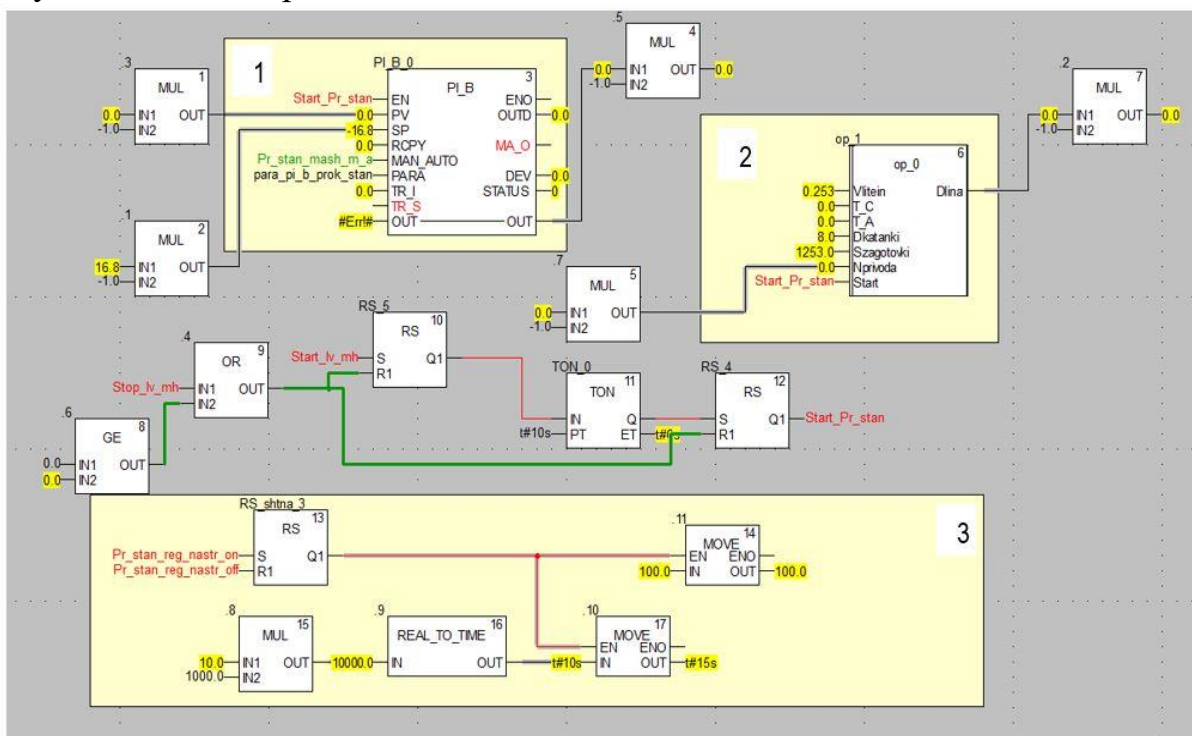


Рисунок 11.12 – Програма імітації формування заготовки та регулювання її довжини

Блок розрахунку поточної довжини заготовки, за формулою (8), а також керування роботою механічних ножиць, за допомогою яких відрізаються перші 10 метрів неякісної заготовки на початку роботи ливарного колеса, приведено на рисунку 11.13.

op1		<DFB>
<div style="display: flex; justify-content: space-between;"> <div style="width: 30%;"> <p><b>inputs</b></p> <ul style="list-style-type: none"> <li>INIT 1 BOOL</li> <li>V_stan 2 REAL</li> <li>Tros 5 REAL</li> <li>Tsg 6 REAL</li> </ul> <p><b>outputs</b></p> <ul style="list-style-type: none"> <li>Qsag 1 REAL</li> </ul> </div> <div style="width: 65%;"> <pre> if V_stan&gt;0.0 and Tros&gt;0.0 and Tsg&gt;10.0 then Qsag:=((p_met*V_stan)/60.0)*F*(L+C_met*(Tros+273.0)) -(G_ros*((L+C_met*(Tros+273.0))-(L+C_sg*(Tsg+273.0)))); else Qsag:=0.0; end_if;                     </pre> </div> </div>		
<p><b>public</b></p> <ul style="list-style-type: none"> <li>C_sg REAL 1027.0</li> <li>L REAL 362300</li> <li>C_met REAL 1250.0</li> <li>p_met REAL 2700.0</li> <li>F REAL 0.0023</li> <li>G_ros REAL 1.26</li> </ul>		

а)

б)

Рисунок 11.13 – Оголошені змінні (а) та код (б) блока розрахунку поточної довжини заготовки

У третьому вікні імітатора представлено пристрій для гартування й охолодження катанки та здвоєна моталка кошикового типу. Також передбачені візуальні засоби контролю та керування відповідними технологічними параметрами (рис.11.14).

З пульта оператора за допомогою кнопок можна увімкнути в роботу та вимкнути моталку, настроїти її роботу на відповідний діаметр катанки із заданим

кроком укладання. При цьому на екрані відбувається емуляція її роботи та графік зміни швидкості виткоутворювача залежно від установлених параметрів формування бунта.

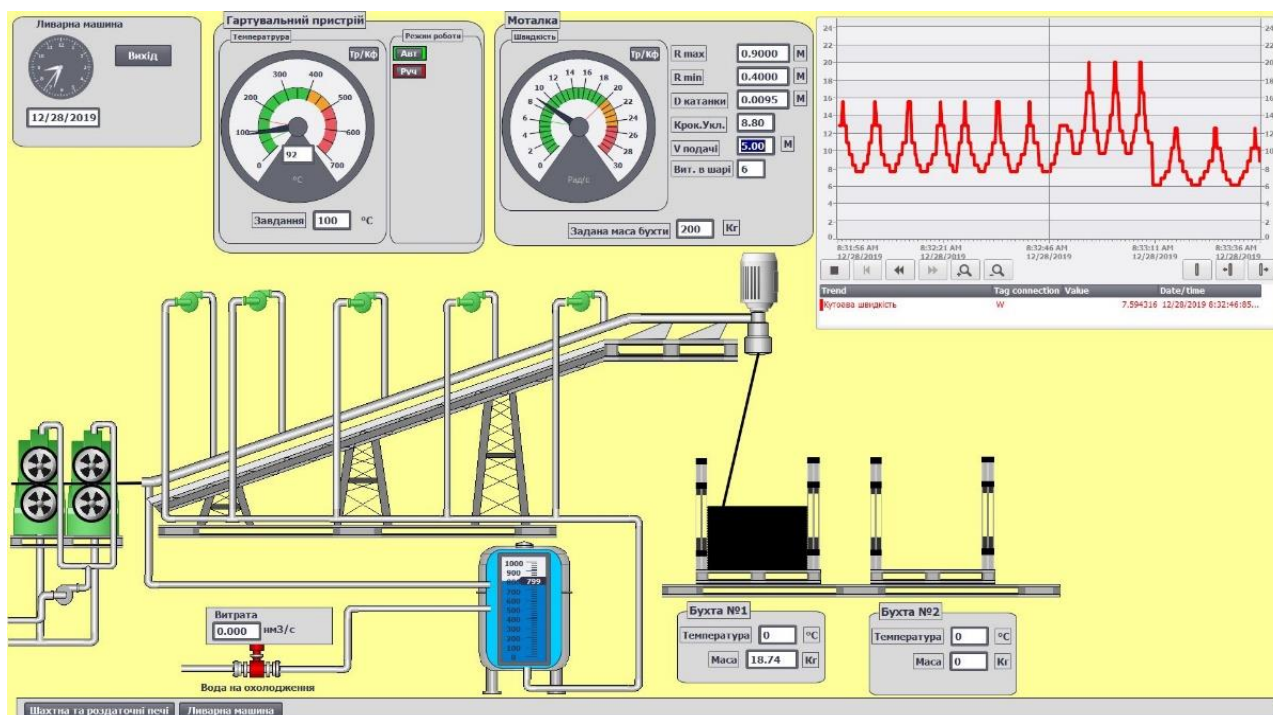


Рисунок 11.14 – Вікно візуалізації гартувального пристрою та моталки

Код програми, що дискретно керує швидкістю виткоутворювача моталки, приведено на рисунку 11.15.

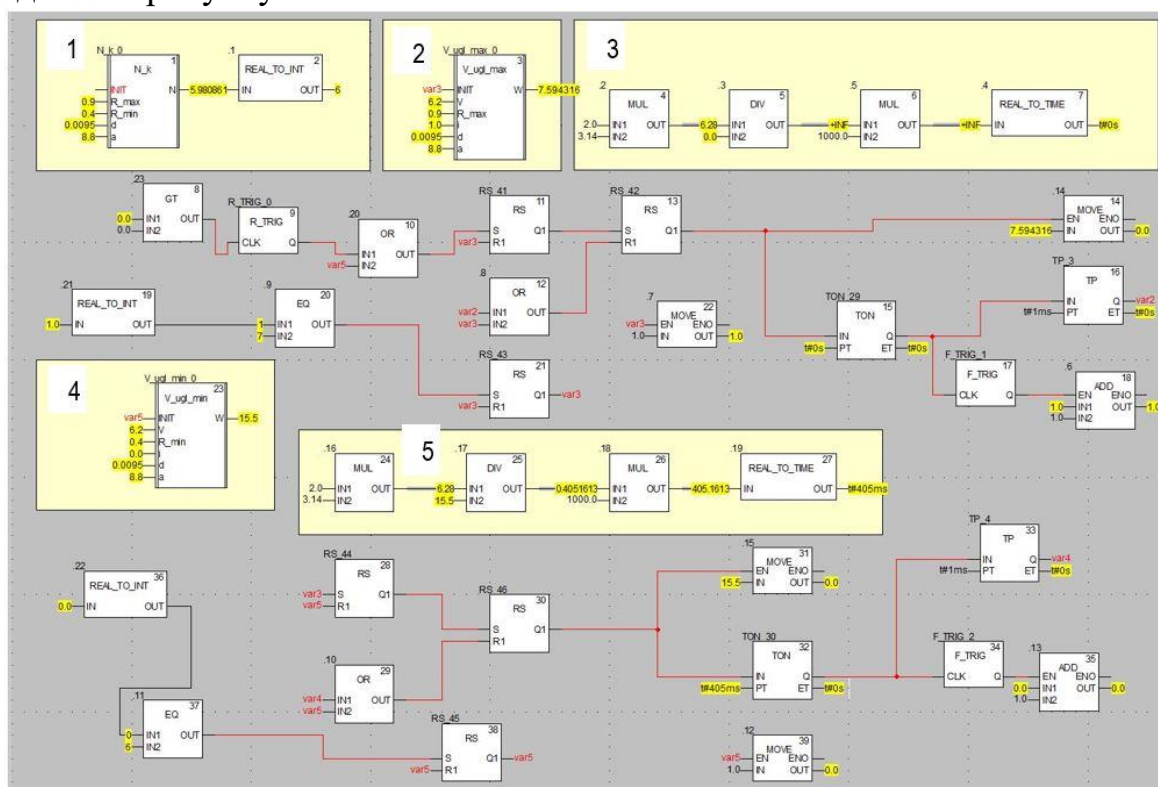


Рисунок 11.15 – Програма керування швидкістю виткоутворювача моталки

У блоці 1 розраховується кількість витків у шарі, у блоках 2 і 4 за формулою (9) – кутова швидкість проводки виткоутворювача при укладанні витків, відповідно, від периферії до центру і навпаки, а у блоках 3 і 5 за формулою (10) розраховується час формування витків при кожній швидкості виткоутворювача.

Функціональний блок, що розраховує кутову швидкість обертання виткоутворювача, приведено на рисунку 11.16.

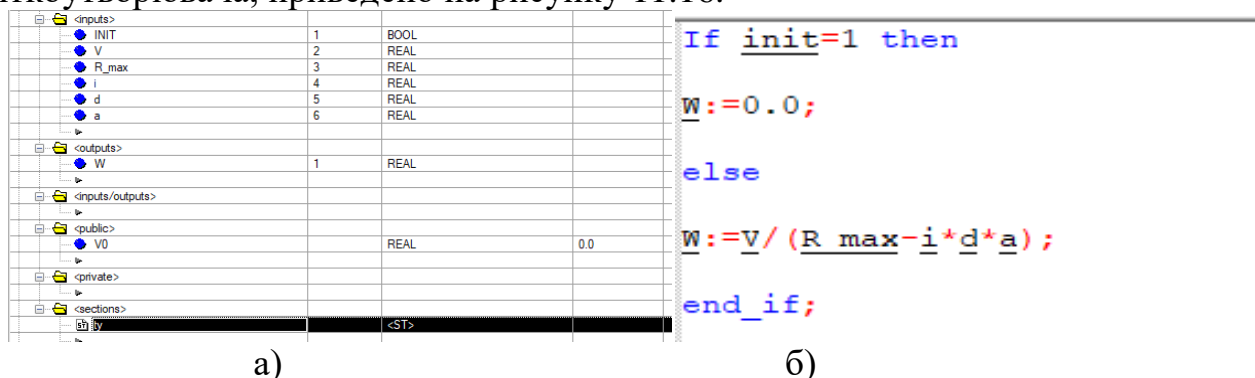


Рисунок 11.16 – Оголошені змінні (а) та код (б) блока розрахунку кутової швидкості виткоутворювача

Програмне забезпечення імітаційної моделі АСУТП з виробництва алюмінієвої катанки дозволяє реалізувати два ручних режими керування ливарно-прокатним агрегатом – вільний і заблокований, а також автоматичний. При реалізації вільного режиму управління, кожна група механізмів управляється самостійно, що дозволяє оператору вручну синхронізувати їх роботу. При заблокованому режимі робота всіх механізмів прив'язана до швидкості ливарного колеса, яка визначає продуктивність ливарно-прокатного агрегату. В автоматичному режимі регулювання технологічних параметрів та управління виробничими процесами відбувається у реальному часі. Передбачена можливість коригування коефіцієнтів в об'єктах автоматизації, зміни значень параметрів в алгоритмах керування, настройок регуляторів та активізації збурень, що впливають на хід технологічного процесу. Усе це створює можливості для імітування різних ситуацій у роботі пічної ділянки та ливарно-прокатного агрегату, дослідити вплив тих чи інших параметрів на хід технологічного процесу, а також відпрацювати дії оператора-технолога АРМ та наладників АСУТП.

## 12.2. Дослідження роботи імітаційної моделі АСУТП з виробництва алюмінієвої катанки

Для підтвердження можливостей розробленої імітаційної моделі АСУТП з виробництва алюмінієвої катанки методом неперервного лиття та прокатки проведені деякі дослідження у технологічному процесі та у роботі систем автоматичного регулювання.

Для одержання заготовки бажаної якості необхідно забезпечувати стабільні умови кристалізації рідкого металу або здійснювати управління процесом його твердіння. У зв'язку з цим проведені дослідження впливу швидкості обертання ливарного колеса, температури металу на вході і виході з кристалізатора на величину теплоти, що виноситься заготовкою з ливарної машини, а також

довжину заготовки на шляху до прокатного стана. Для цього в імітаційній моделі у ручному режимі роботи по черзі змінювали швидкість ливарного колеса у діапазоні 0,19 – 0,21 м/с, температуру розплаву від 963К до 983К та температуру заготовки на виході з ливарного колеса з 723К до 773К і за показаннями приладів у другому вікні імітаційної моделі відслідковували кількість теплоти, що виноситься заготовкою з ливарного колеса та довжину заготовки. За результатами досліджень встановлено, що найбільший вплив на кількість теплоти, що виноситься з заготовкою має швидкість ливарного колеса та температура заготовки на його виході. При коливаннях швидкості ливарного колеса в межах (0,19 – 0,21)м/с та незмінних умовах охолодження розплаву, кількість теплоти, що виноситься заготовкою з ливарної машини, змінюється з 1029273Вт до 1194647Вт тобто на 13,8%. При підвищенні температури заготовки на виході з ливарної машини з 723К до 773К, кількість теплоти, що виноситься заготовкою з ливарного колеса збільшується з 1111960Вт до 1192411Вт, на 6,8%. У той же час вплив температури алюмінієвого розплаву незначний. При коливаннях його температури від 963К до 983К, кількість теплоти, що виноситься заготовкою змінюється від 1114764Вт до 1140656Вт, на 2,3%.

Результати досліджень, що до впливу розглянутих параметрів на довжину заготовки показали, що найбільший вплив має швидкість ливарного колеса. Коливання її в межах  $\pm 0,5\%$  від базового 0,2м/с приводять до зміни довжини заготовки на 0,18м або на 1,07%. Зміна температури розплаву від 963К до 983К та температури заготовки в діапазоні (723–773)К практично не впливають на довжину заготовки. Її коливання складають долі процента. Додаткові дослідження впливу коефіцієнта витяжки металу при його прокатуванні показали, що цей параметр має такий самий вплив на довжину заготовки, як і швидкість ливарного колеса. При зміні коефіцієнта витяжки від 22,4 до 21,4 довжина заготовки між ливарною машиною та прокатним станом збільшилась на 0,16м.

Якість роботи систем автоматичного регулювання можна дослідити за будь-яким технологічним параметром від шахтної печі до моталки. Для цього на кожному з трьох основних екранів імітаційної моделі необхідно установити автоматичний режим роботи і за допомогою настроювальної панелі вибрати настройки регулятора, активізувати появу збурень і змінити задане значення. Результати роботи системи автоматичного регулювання можна спостерігати на відповідному тренді. В роботі приведені дослідження систем автоматичного регулювання теплоти, що виноситься заготовкою з ливарного колеса, та довжини заготовки, а також системи автоматичного керування роботою моталки. На рисунку 11.17 послідовно показані перехідні процеси в системі автоматичного регулювання теплоти, що виноситься заготовкою з ливарного колеса, при зміні завдання, при регулюванні тільки за відхиленням теплоти, що виноситься заготовкою, та при регулюванні за відхиленням теплоти і збуренням, що надходить з боку системи охолодження зливка. Як видно з рисунку, якість регулювання у комбінованій системі автоматичного регулювання краща за всіма показниками: максимальне динамічне відхилення, перерегулювання та тривалість перехідного процесу. Це суттєво сприяє стабілізації теплового режиму в кристалізаторі та одержанню заготовки з необхідними властивостями.



Рисунок 11.17 – Перехідні процеси в САР теплоти, що виноситься заготовкою з ливарного колеса

На рисунку 11.18 приведені перехідні процеси, що отримані у системі автоматичного регулювання довжини заготовки, яка працює тільки за відхиленням регульованого параметра і за алгоритмом, що використаний у імітаційній моделі. Аналіз приведених графіків показав, що якість регулювання за всіма показниками краща при комбінованій системі автоматичного регулювання (останній фрагмент графіка).



Рисунок 11.18 – Перехідні процеси в САР довжини заготовки

При дослідженні системи автоматичного керування укладанням катанки в бунт на екрані, що відображає гартівник та моталку, установили діаметр катанки 9,5мм і по черзі змінювали крок укладання, так, щоб у кожному шарі бунта формувалось послідовно – 6, 3 і 18 витків. На рисунку 11.19 видно, що керування швидкістю виткоутворювача відбувається дискретно, чим більше кількість витків у шарі, тим більше частота зміни швидкості роботи виткоутворювача.

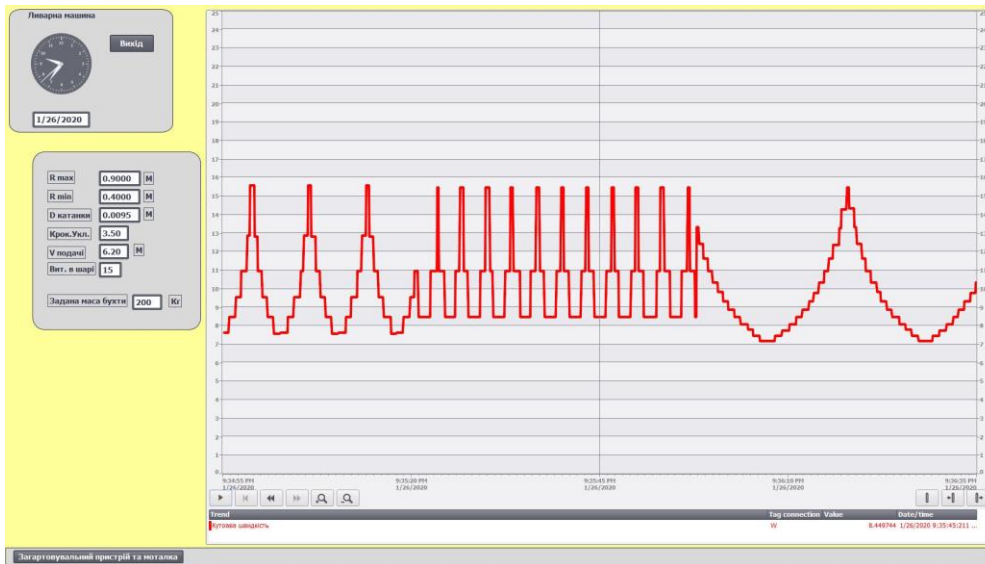


Рисунок 11.19 – Графік зміни швидкості виткоутворювача при кількості витків катанки у шарі, відповідно 6, 3 і 18

## Література

1. Пупена О.М., Ельперін І.В. Програмування промислових контролерів у середовищі UNITY PRO. Навч. Посібник.–К.Видавництво: Ліра-К, 2015.– 376 с.
2. Сухих Я.А, Колбин І.В. ПЛК Modicon M580. Обзор аппаратных средств и возможностей по построению систем АСУТП. SCHNEIDER ELECTRIC