МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІНЖЕНЕРНИЙ НАВАЛЬНО-НАУКОВИЙ ІНСТИТУ

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

Програмування промислових контролерів

Конспект лекцій.

для студентів напряму 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.Програмований мікропроцесорний контролер Modicon TSX Premium

Мікропроцесорний контролер TSX Premium призначений для керування складними технологічними або виробничими процесами, які вимагають обробки великої кількості інформації і керування великою кількістю виконавчих механізмів.

Modicon TSX Premium, рис.1, – це потужний контролер, який може

Рисунок 1 – Мікропроцесорний контролер Modicon TSX Premium

працювати з тисячами дискретних і сотнями аналогових входів-виходів. Він має розподілену фізичну структуру, широкий діапазон модулів вводу-виводу, багатозадачну операційну систему, потужну систему обробки переривань, а також вбудовані функції контурів регулювання, широкий діапазон модулів керування рухом, спеціальні модулі, вбудовану мережу Fipio та ін.

Архітектурно TSX Premium складається з одного або кількох з'єднаних між собою окремих шасі, на яких установлюються різноманітні процесори, блоки живлення, модулі дискретних і аналогових входів-виходів, лічильники, комунікаційні модулі та інші.

1.1 Шасі

Основним конструктивним елементом контролера є шасі. Шасі виконує дві основні функції - механічну і електричну. З одного боку, шасі використовується як конструктивний елемент, на якому розміщуються і закріплюються окремі модулі контролера, з іншого - шасі має загальну Bus X шину, по якій відбувається живлення модулів, що установлені у шасі, і обмін сигналами та даними між окремими модулями контролера.

Для того щоб більш повно відповідати вимогам користувача, контролер Premium комплектується двома типами шасі – стандартним і розширюваним, кожне з яких може мати 4, 6, 8 або 12 місць для встановлення модулів. Стандартне шасі використовується у тому разі, коли контролер складається з кількості модулів не більше 12, а розширюване призначене для систем автоматизації, проектне компонування яких передбачає використання більше 12 модулів.

Розширюване шасі на відміну від стандартного має мікроперемикачі для кодування адреси шасі, а також розніми, через які, за допомогою спеціального кабелю, внутрішні Bus X шини окремих шасі об'єднуються у загальну Bus X шину, рис.2. Це дає змогу модулям різних шасі, кількістю не більше 16, обмінюватися сигналами і даними між собою. На кінцях Bus X шини мають бути встановлені термінатори TSX TLY EX з індексами A і B. Розташування їх може бути у будь-якому порядку, але на одній шині повинні бути термінатори з різним літерним позначенням. Загальна довжина Bus X шини не повинна перевищувати 100 метрів.

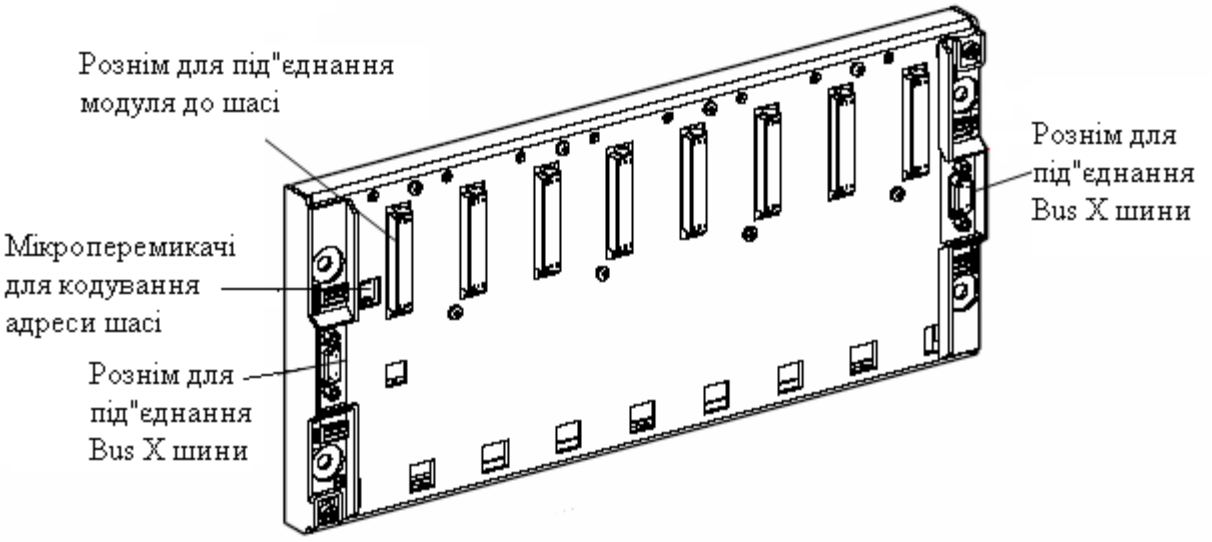


Рисунок 2 – Розширюване восьмипозиційне шасі

Для побудови більш складних систем автоматизації використовуються спеціальні модулі розширення Bus X шини - TSX REY 200, які є основою розгалуженої сегментної структури контролера, рис. 3.

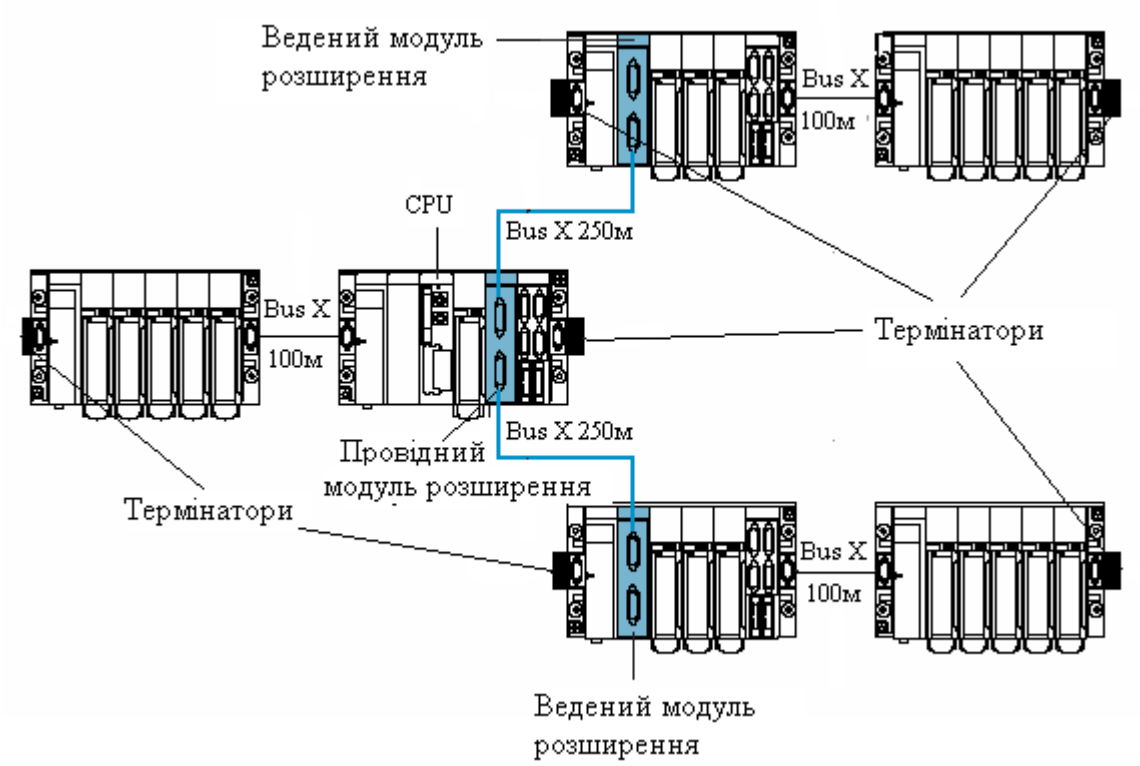


Рисунок 3 – Розгалужена структура контролера з віддаленими шасі.

Модуль TSX REY 200 має два канали для підключення віддалених сегментів Bus X шини кабелем TSX CBRY 2500 довжиною не більше 250м. Якщо модуль TSX REY 200 встановлений у шасі з центральним процесором, то він є провідним і виконує функцію "Майстра" по відношенню до аналогічних модулів встановлених в шасі розширювання і, які мають назву "Ведених".

Кожне шасі, яке входить до складу контролера, має свою унікальну адресу. Якщо контролер складається з одного стандартного шасі, воно по замовчуванню має адресу 00. Якщо контролер складається з кількох розширюваних шасі, то адреса кожного з них у двійковому коді виставляється за допомогою мікроперемикачів, які розташовані на шасі. Порядок розташування шасі на Bus X шині не залежить від їхньої адресації.

Для всіх стандартних і розширюваних шасі використовується географічний принцип адресації модулів, тобто адреса модуля залежить від його позиції на шасі. Адреса кожної позиції вказується на шасі нижче кожного з'єднувача. Перший з'єднувач з позначкою PS завжди призначений для блока живлення.

У зв'язку з тим що модулі живлення і процесорні модулі можуть бути як стандартного, так і подвійного формату, то може змінюватись як кількість місць, які відводяться для встановлення інших модулів, так і адреси, які вони можуть мати.

Наприклад, якщо блок живлення має подвійний формат, то він займає місця з адресами PS і 00. Тоді процесорний модуль може бути встановлений тільки на місце під номером 01, а інші модулі можуть займати місця, починаючи з номера 02. Якщо ж і процесорний модуль має подвійний формат, то модулі можуть займати місця, починаючи з номера 03.

Скомпоновані шасі встановлюються на DIN- рейки або на перфоровані монтажні пластини.

1.2 Модулі процесорів

Процесорні модулі, якими компонується контролер, умовно можна поділити на дві групи - TSX P57 xx3M і TPCX 57 xx3M .

Процесори TSX P57 103M, TSX P57 203M, TSX P57 303M, TSX P57 403M установлюються на шасі контролера, а TPCX 57 203M, TPCX 57 353M - на ISA-шині промислового або офісного PC і працюють у середовищі Windows 95/98, Windows 2000 або Windows NT.

Модулі процесорів розрізняються функціональними можливостями, основними з яких є:

- кількість шасі, які можуть входити до складу ПЛК(процесори TSX P57 xx3M – 2, 4, 6 або 16, а TPCX 57 xx3M – 8 або 16);
- кількість входів-виходів, які може обробити контролер (процесори TSX P57 xx3M – 512, 1024, 2048 дискретних і 24, 80, 128, 256 – аналогових, а процесори TPCX 57 xx3M – 1024 дискретних і 80 або 128 аналогових);

• кількість спеціальних модулів – лічильники, модулі вимірювання ваги та ін.

(від 8 до 64);

• кількість і типи мереж, до яких може приєднуватись ПЛК (від 1 до 4);

• кількість конфігурованих контурів регулювання (від 10 до 20);

• види і ємність пам'яті (вбудованої ОЗП – від 32Кбайт до 128Кбайт, а ОЗП розширення від 64 Кбайт до 512Кбайт).

Процесорні модулі TSX P57 15M, TSX P57 253M, TSX P57 353M і TSX P57 453M відрізняються від решти процесорів групи TSX P57 xx3M тим, що в ці модулі інтегрована польова шина FIPIO, яка може використовуватися для підключення від 1 до 127 пристроїв віддалених модулів вводу-виводу типу Momentum.

За розміром процесорні модулі TSX P57 xx3M випускаються двох форматів - стандартного і подвійного. На рис.4 показано загальний вигляд процесорного модуля подвійного формату, який встановлюється на шасі контролера, а на рис.5 - загальний вигляд процесорного модуля T PCX 57 xx3M, що встановлюється на ISA-шині комп’ютера.

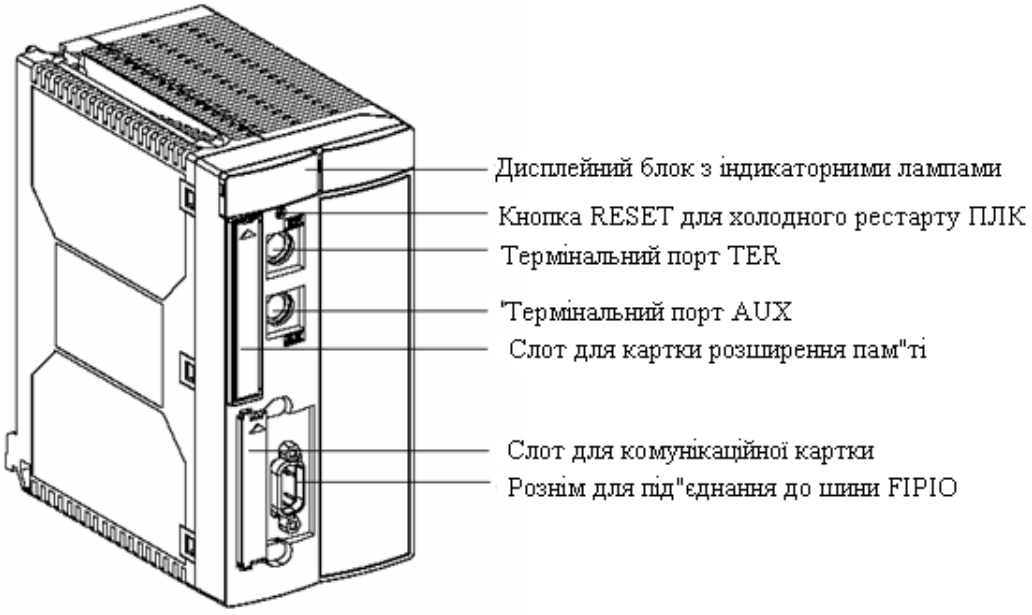


Рисунок 4 – Загальний вигляд процесорного модуля TSX P57 xx3M подвійного формату

Кожний процесор має термінальний порт (не ізольований зв'язок RS485) з двома рознімами (у TPCX57 – один). Рознім TER призначений для приєднання PC- сумісного термінала, або для підключення ПЛК до мережі UNI- TELWEY, MODBUS. Цей порт дає можливість надати живлення 5В на пристрої, що до нього підключені. Рознім AUX використовується для підключення периферійних пристроїв, які мають власне джерело живлення.

Кожний процесор має ОЗП для зберігання програми користувача, а також слот для установлення комунікаційної картки формату PCMCIA, яка дає можливість зв’язатися з процесором, а також збільшити пам'ять контролера.

Внутрішня память процесора TSX P57 може бути захищена батарейкою, яка встановлена у блоці живлення. В процесорах TPCX57 батарейка встановлюється на платі самого процесора.

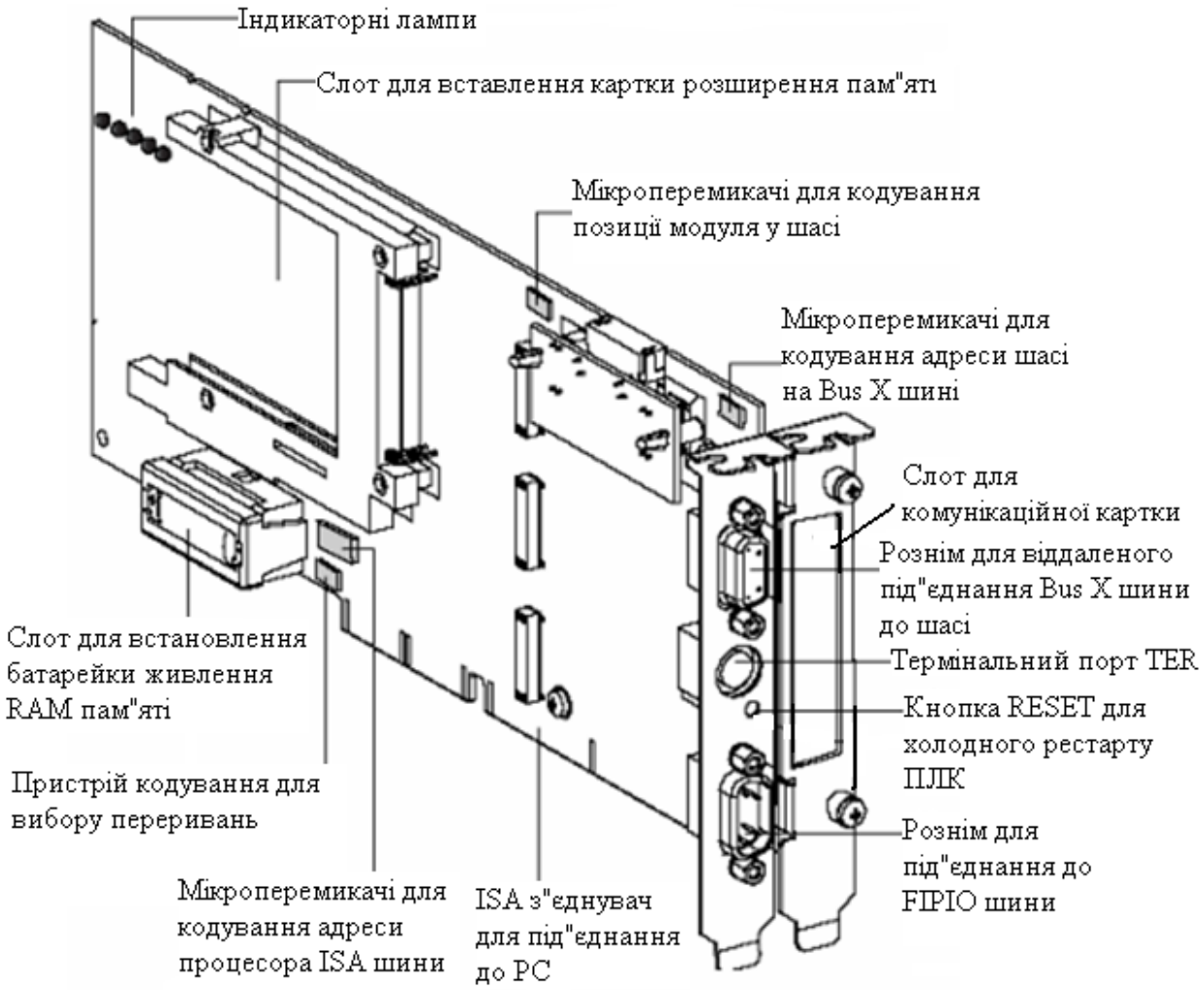


Рисунок 5 – Загальний вигляд процесорного модуля TPCX 57xx3M, що встановлюється на ISA-шині EOM

Кожний процесор має кнопку RESET, натиснення на яку призводить до холодного рестарту ПЛК. Процесорний модуль T PCX 57 xx3M додатково має мікроперемикачі для кодування модуля у шасі, кодування адреси процесора ISA шини і кодування адреси шасі на BUS X шині.

В усіх процесорах є годинники реального часу, робота яких підтримується батарейкою. Годинники відлічують поточну дату і час, дату і час останньої зупинки програми. Дата і час відлічуються і при вимкнутому контролері, якщо джерело живлення має резервну батарейку.

При виборі процесорного модуля, який встановлюється на ISA-шині комп’ютера, місце для встановлення процесорного модуля на шасі залишається вільним. Інші модулі контролера займають наступні після вільного місця шасі..

1.3 Блоки живлення

У кожному шасі повинен бути встановлений блок живлення TSX PSY.

Пропонуються різні типи блоків, які розрізняються живленням від напруги змінного ~100-240В (TSX PSY2600M, TSX PSY5500M, TSX PSY8500M) або постійного =24В (TSX PSY1610M, TSX PSY3610M, TSX PSY5520M) струму, потужністю, а також розмірами. Блоки, що живляться змінною напругою, мають додатковий вихід для живлення кіл датчиків напругою 24В постійного струму. Окрім того кожне джерело живлення має дисплейний блок, сигнальне реле, слот для установлення батарейки, яка забезпечує зберігання даних у пам'яті процесора, кнопка під олівець RESET для теплого рестарту додатку ПЛК, живлення датчиків =24В.

Сигнальне реле, що встановлене у кожному блоці живлення, виконує кілька функцій:

• якщо блок живлення розташований у шасі з установленим модулем процесора, то за нормальної роботи контролера контакт сигнального реле замкнений;

• якщо з якоїсь причини виконання програми припиняється і контролер переходить у режим STOP, контакт реле розмикається.

У блоках живлення, встановлених у інші шасі контролера, контакт сигнального реле замкнений у разі, якщо блок живлення працює нормально. В іншому разі цей контакт розмикається. Отже, контакти сигнального реле можна використовувати у системах безпеки контролера і системи керування.

Блок живлення для кожного шасі вибирається виходячи з типів і кількості модулів, які планується встановити у шасі. Для цього використовуються дані, наведені в інструкції з експлуатації, про потужність, яку споживає кожний модуль при напрузі ±5 і ±24 В. Після цього розраховується загальна потужність, яку споживають всі модулі, що встановлені у шасі, і підбирається блок живлення, який може задовольнити цим потребам. Існує два формати блоків живлення – стандартний і подвійний. До стандартного відносяться блоки TSX PSY 1610M і TSX PSY 2600M, до подвійного - більш потужні TSX PSY 3610M, TSX PSY 5520M, TSX PSY 5500M і TSX PSY 8500M. Модулі стандартного формату встановлюються у перший слот кожного шасі TSX PKY і займають позицію PS, а подвійного формату – у перші два слоти з позиціями PS і 00.

Окрім блоків живлення контролера випускається широка гама модулів для живлення напругою =24В (TBX SUP 10, TSX SUP 1011, TBX SUP 1021, TBX SUP 1051 та ін.) периферійних пристроїв, таких як датчики, виконавчі пристрої, операторські панелі, пульти настроювання та ін., а також напругою =30В компонентів, що приєднуються до польовій шині AS-i (TSX SUP A02, TSX SUP A05). Ці модулі живлення встановлюються окрема на спеціальної перфарированої пластині або на DIN-рейці.

1.4 Модулі дискретних вводів-виводів

Для задоволення різноманітних потреб користувача випускається широкий діапазон дискретних модулів вводу (TSX DEY08D2, TSX DEY16FK, TSX DEY32DK, TSX DEY64D2K та ін.) і виводу (TSX DSY08T2, TSX DSY16R5, TSX DSYУ32T2K, TSX DSYУ64T2K та ін.), які розрізняються:

• кількістю каналів – 8(наприклад, TSX DEY08D2, TSX DSY08T2) 16 (наприклад, TSX DEY16D2, TSX DSY16R5) 28, 32(наприклад, TSX DEY32DK, TSX DSYУ32T2K або 64 (наприклад, TSX DEY64D2K, TSX DSYУ64T2K);

• типами входів:

- модулі із входами постійного струму (DC) - 24VDC(TSX DEY 08D2, TSX DEY 16D2), 48VDC(TSX DEY 16D3, TSX DEY 16A3, TSX DEY 32D3K) та ін.;

- модулі із входами змінного струму (AC) - 24VAC (TSX DEY 16A2), 48VAC (TSX DEY 16A3), 110VAC (TSX DEY 16A4), 240VAC (TSX DEY 16A5) та ін.

• типами виходів: .

- модулі з релейними виходами (TSX DSY 08R5, TSX DSY 16R5, TSX DSY 08R5A, TSX DSY 08R40);

- модулі з безконтактними виходами постійного струму (DC) -24VDC/0.1A - 0,5A - 2A (TSX DSY08T2, TSX DSY08T22, TSX DSY16T2 ; 48VDC/ 0.25A - 1a (TSX DSY32D3K);

- модулі з безконтактними виходами змінного струму (AC**)** -24VAC/ 1A (TSX DSY16S4); I30VAC/1A (TSX DSY16S4); 48VAC/2A (TSX DSY08S5, TSX DSY16S5); 240VAC/2A;

• типами приєднання - гвинтова клемна колодка (TSX DEY08D2, TSX DSY16R5 та ін.) або з'єднувач HE10 (TSX DEY64D2K, TSX DSYУ32T2K та ін.).

Серед модулів дискретного вводу є модулі з так званими швидкими входами (TSX DEY 16FK, T3X DMY 28FK та TSX DMY 28RFK). Входи цих модулів можна використовувати як звичайні дискретні входи, входи із запам'ятовуванням або входи для обробки подій.

Модулі вводу-виводу мають пластмасовий корпус стандартного формату і займають 1 слот. На дисплейному блоці кожного дискретного модуля розташовані індикатори його стану - RUN (зелений), ERR і I/O (червоний), а також індикатори з позначенням номерів каналів входів-виходів. Кількість цих індикаторів відповідає кількості каналів модуля. Максимальна кількість таких індикаторів - 32. Якщо модуль розрахований на більшу кількість каналів (64), то у нижній частині дисплея розташована кнопка перемикання на іншу групу з 32 каналів. При цьому у верхній частині дисплея загоряється індикатор +32.

Індикатори каналів висвітлюються при спрацьовуванні відповідного вхідного або вихідного каналу. У нормальному стані модуля повинен горіти тільки індикатор RUN. Висвітлення індикаторів ERR або I/O сигналізує про виявлення системою самодіагностування відмови модуля або окремих його каналів.

Особливістю гвинтових клемних колодок на модулях вводу-виводу є те, що їх можна знімати, рис.6. Окрім цього вони мають спеціальний кодувальний пристрій, який автоматично займає відповідне положення при першому встановленні клемної колодки на модуль.

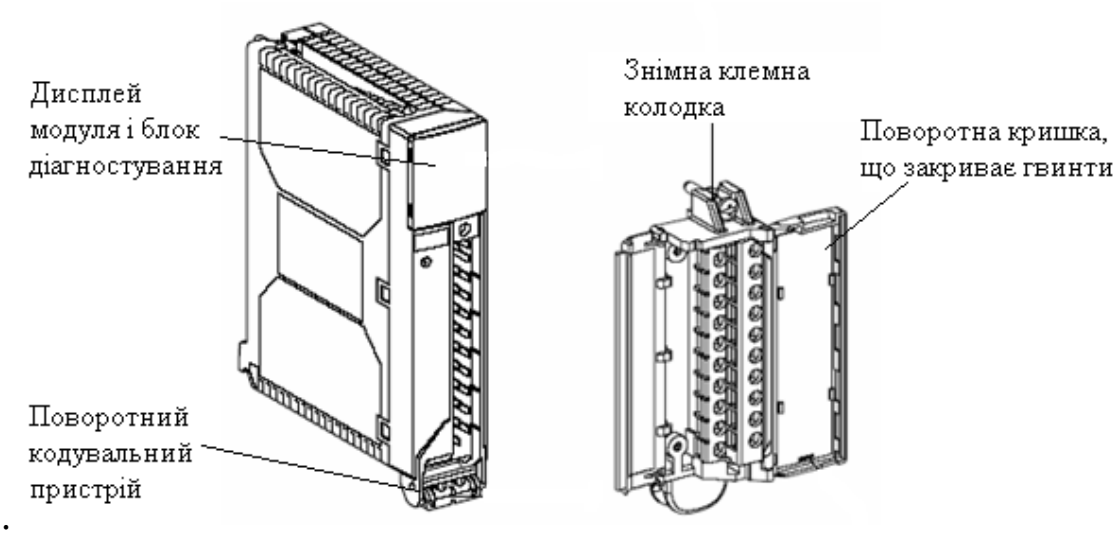


Рис.6 Модуль з підключенням через гвинтову клемну колодку

Кожний тип дискретного модуля має свій код, тому неможливо помилково встановити клемну колодку одного типу модулів на інші. При з'єднанні клемна колодка спочатку вставляється у кодувальний пристрій, а потім гвинтом закріплюється на модулі. Клемна колодка має кришку, яка закриває доступ до клем і має змінний ярлик, на якому із зовнішнього боку вказується тип модуля і можуть бути внесені позначення входів-виходів. З внутрішнього боку кришки показана схема підключення до входів-виходів модуля.

Для приєднання до модулів з HE10 - з'єднувачами використовуються спеціальні кабелі, які обладнані відповідними рознімами. При цьому можливі два варіанти приєднання.

У першому варіанті, рис.7, використовується кабель, який з одного кінця має HE10- рознім, за допомогою якого кабель приєднується до модуля, а з іншого кінця розташовані кодовані кольором вільні кінці проводу.

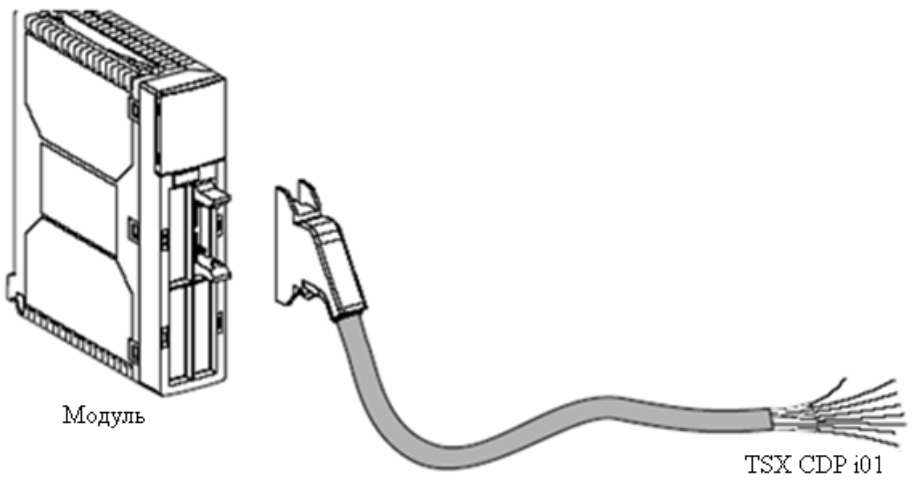


Рисунок 7 – Сполучний кабель з HE10 рознімом і вільними кінцями проводу

В іншому варіанті**,** рис.8, до модулів з HE10–з'єднувачами за допомогою кабелю з

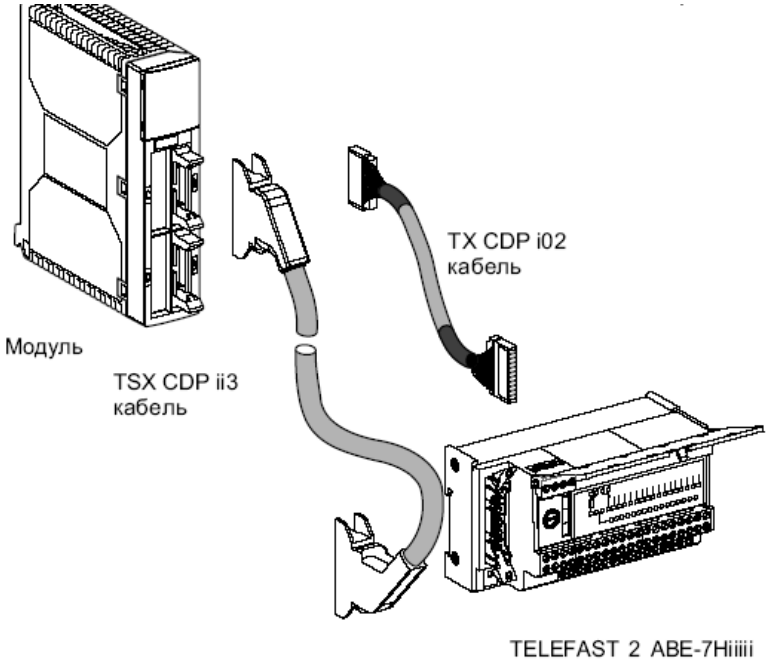


Рисунок 8 – Пристрій для швидкого монтажу TELEFAST 2

рознімами приєднуються пристрої для швидкого монтажу TELEFAST-2.

Система TELEFAST-2 – це група виробів для швидкого підключення дискретних модулів до зовнішніх кіл. Вони замінюють гвинтові клемні колодки і забезпечують віддалене розташування блоків з клемниками, до яких вже безпосередньо приєднуються датчики і виконавчі пристрої.

Кількість HE10- з'єднувачів (конекторів), розташованих на лицевій панелі модуля, залежить від кількості каналів, з якими працює цей модуль. Так, для модуля, розрахованого на 64 канали, кількість конекторів чотири, а для модуля, розрахованого на роботу з 16 каналами – один.

1.5 Модулі аналогових вводів-виводів

Модулі аналогових вводів (TSX AEY1600, TSX AEY 800,TSX AEY 810, TSX AEY 420, T5X AEУ 414, TSX ASY 1614) і виводів (TSX ASY 410, TSX ASY 800) в Premium – це модулі стандартного формату, тому займають одну позицію у будь-якому слоті, окрім слота, призначеного для встановлення блока живлення, рис.9.

На лицевій панелі вони мають дисплейний блок з трьома індикаторними лампами RUN, ERR і I/O, які відображають режим роботи та можливі несправності, рознім для гвинтової колодки і кодовий ключ або SUB-D рознім для датчиків і виконавчих пристроїв.

Різноманітність аналогових модулів задовольняє більшості вимогам користувача. Аналогові модулі розрізняються:

• кількістю каналів 4 (TSX AEY 420,T5X AEУ 414, TSX ASY 410), 8 (TSX AEY 800, TSX AEY 810,TSX ASY 800) і 16 (TSX AEY1600, TSX ASY 1614);

• характеристиками і діапазонами сигналів – термометри опору (T5X AEУ 414), термопара (T5X AEУ 414, TSX ASY 1614), напруга, струм – решта модифікацій модулів вводу-виводу;
• наявністю гальванічного розділення;
• типами приєднання – клемна колодка (T5X AEУ 414,TSX ASY 410) або 25-штирковий SUB-D конектор – решта модифікацій модулів вводу-виводу.

Максимальна кількість аналогових каналів залежить від модуля процесора, який встановлений у контролері.

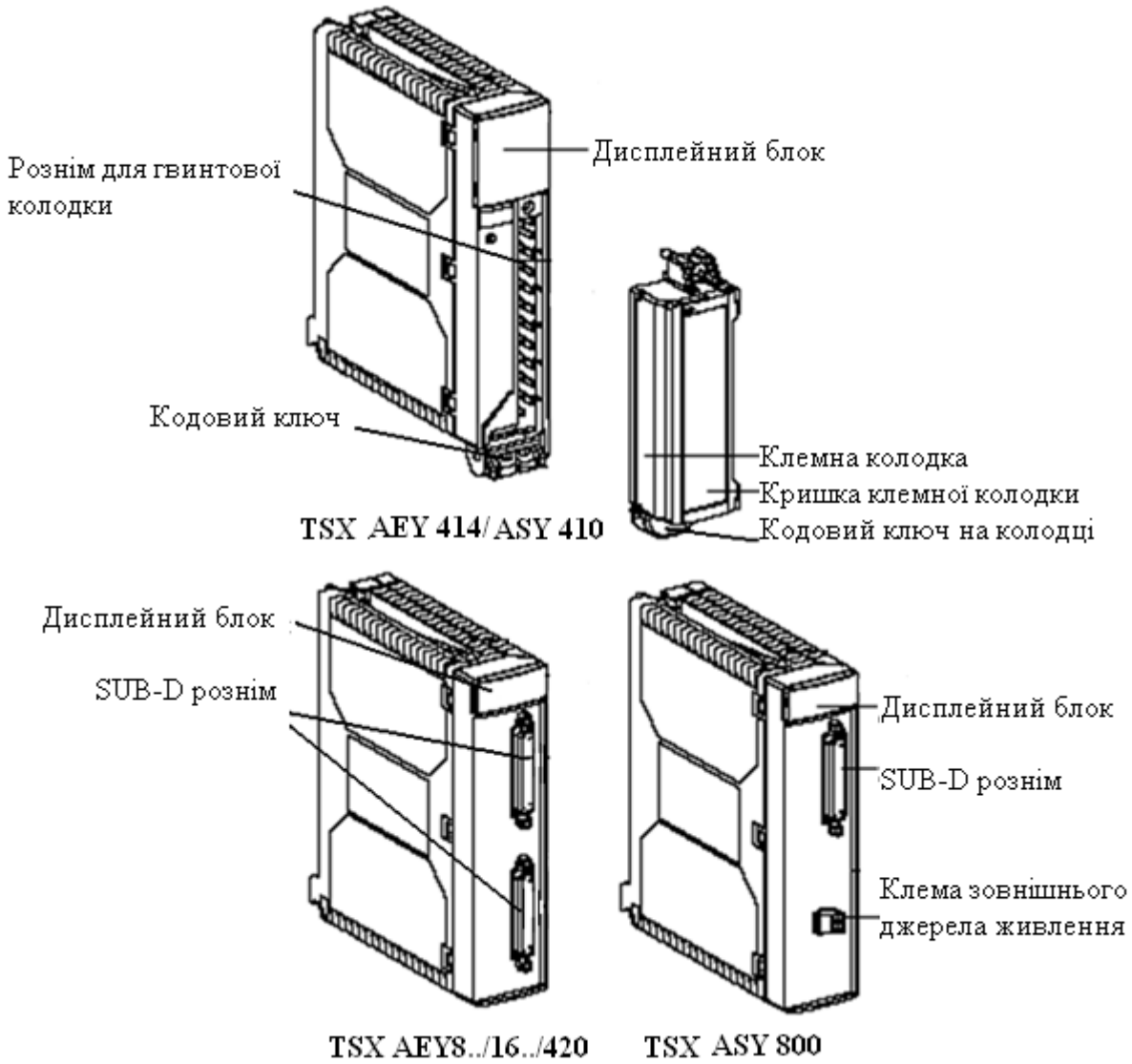


Рисунок 9 – Модулі аналогових вводів-виводів TSX AEY 414/ ASY 410,TSX AEY 8../16../420 і TSX ASY 800

Особливістю застосування модуля аналогового виходу TSX ASY 800 є те, що кількість таких модулів, встановлених в одне шасі, обмежено двома модулями. Це викликано тим, що цей модуль споживає великий струм при напрузі 24В. Тому в разі потреби використання більшої кількості таких модулів необхідно забезпечити їх додаткове живлення зовнішнім джерелом живлення. Для цього на лицьовій панелі модуля розташовані спеціальні клеми.

Час опитування модулів залежить від вибраного в процесі конфігурування режиму. За нормального режиму час опитування становить 27, 51 і 80 мс, а за швидкого режиму – 2 або 3 мс на канал.

Для модулів аналогових входів з термінала програмування можна виконати перекалібрування каналів, щоб усунути похибки вимірювання, які виникли із-за наявності дрейфу характеристик вимірювальних каналів.

Зовнішні кола приєднуються до модулів з SUB-D конектором за допомогою блоків швидкого монтажу TELEFAST.

З'єднання виконується за допомогою екранованого кабелю TSX CAP 030 довжиною 3м з рознімами на обох кінцях.

Блок TELEFAST ABE-7CPA02 забезпечує підключення восьми каналів.

Блок TELEFAST ABE-7CPA03/31 забезпечує підключення восьми каналів, а також живлення каналів напругою =24В, провідність струмових кіл при від'єднанні SUB-D конектора, захист струмового шунта всередині модуля від перевантаження.

Блок TELEFAST ABE-7CPA12 забезпечує підключення 16 термопар. На ньому встановлений датчик температури для компенсації холодних спаїв.

Аналогові модулі можна знімати і встановлювати у слоти шасі при підключеному живленні контролера.

1.6 Модулі лічильників і спеціальних функцій

Для реалізації функцій підрахунку імпульсів у ПЛК TSX Premium використовуються три типи модулів лічильників - TSX CTY 2A, TSX CTY 4A та TSX CTY 2C. Модулі CTY 2A/4A, відповідно, на два і чотири канали використовуються для підрахунку імпульсів із максимальною частотою 40 кГц у режимах прямого, зворотного і реверсивного ліченя. Модуль TSX CTY 2C використовується як реверсивний лічильник і для вимірювання імпульсів із частотою до 1 МГц.

Параметри лічильників встановлюються програмним шляхом у процесі конфігурування. До цих модулів можна приєднувати датчики з безконтактними виходами (інкрементні декодери, фотоелектричні датчики, датчики положення та ін.) і датчики з механічними контактами (у цьому разі припустима частота імпульсів не повинна перевищувати 100 Гц)[1,16,17].

До переліку модулів, які виконують спеціальні функції, відносяться модулі TSX CAY 21/22, TSX CAY 41/42 та TSX CAY 33. Вони використовуються для розв'язання задач керування серводвигунами з метою досягнення ефективного керування складним рухом машин і агрегатів.

Модулі TSX CAY 21/22 керують переміщенням по двом незалежним осям (канал 0 і 1), модулі TSX CAY 41/42 можуть контролювати керування чотирма незалежними осями, а модуль TSX CAY 33 — трьома осями. Модулі дають можливість контролювати положення робочих органів машин, змінювати швидкість і прискорення їх переміщення, контролювати їх запуск і зупинку.

Для керування машинами, які застосовують крокові двигуни, використовуються модулі TIX CAH 11/21.

Для розв'язання задач автоматичного вимірювання ваги, дозування, регулювання подачі матеріалу на конвеєр тощо, використовується інтегрована система вимірювання ваги IPS Plus, до складу якої входять модуль вимірювання ваги TSX ISP H100, індикатор ваги TSX XBT H100 і тензодатчики.

Індикатор ваги TSX XBT H100 поставляється разом із модулем TSX XBT H100. Він попередньо сконфігурований і на ньому відображаються всі операції вимірювання ваги.

Для роботи з операціями вимірювання ваги додатково поставляється спеціалізоване програмне забезпечення.

1.7 Віддалені та розподілені вводи-виводи

На базі контролерів TSX Premium можна реалізувати схеми віддаленого та розподіленого вводу-виводу. Організація віддаленого вводу-виводу забезпечується за допомогою вбудованого в процесор порту шини FIPIO і модуля TSX SAY 100, який надає контролеру можливості застосовувати AS-i шину у якості провідного.

AS-i шина призначена для передачі дискретної інформації між модулем TSX SAY 100 і датчиками та виконавчими пристроями, що виступають у ролі підпорядкованих. Фізично всі елементи з'єднані між собою або двопровідним спеціальним плоским кабелем, або стандартним двопровідним круглим кабелем. Загальна довжина усіх гілок шини не повинна перевищувати 100м без використовування повторювачів. Для живлення компонентів, що приєднані до AS-i шини, напругою =30В, використовуються спеціальні модулі контролерів TSX Premium – TSX SUP A02 і TSX SUP A05. Ця напруга подається по тому ж кабелю, що використовується для обміну інформацією. Приклад топології AS-i шини наведений на рис.10.

Рисунок 10 – Приклад топології AS-i шини

При створені віддаленого вводу-виводу за допомогою шини FIPIO, може використовуватися проста архітектура, коли працює тільки одна станція або

більш складна, в якій кілька сегментів FIPIO можуть управлятися локальною мережею більш високого рівня, наприклад FIPWAY або Ethernet TCP_IP.

Пропонується велика гама модулів дискретних і аналогових входів-виходів типу TBX, які приєднуються до контролера через мережу Fipio інтегровану у більшість процесорних модулів. До окремих модулів TBX може підключатись до 16 дискретних вхідних-вихідних сигналів і до шести аналогових. Адресація у мережі Fipio виконується за допомогою спеціальних перемикачів. Кожний модуль має дисплейний блок, за допомогою якого виконується діагностика модуля. Швидкість передачі даних по шині FIPIO – 1Мбод. Кількість вузлів, об'єднаних екранованою витою парою, не повинна перевищувати 128. В мережі за допомогою повторювачів допускається створювати до 15 сегментів довжиною 1000м.

З 1998 р. компанія Schneider Electric почала випускати гаму модулів розподіленого керування - Modicon TSX Momentum, до складу якої входять модулі розподілених входів-виходів, процесорів, комунікаційних адаптерів і адаптерів розширення. TSX Momentum може приєднуватись до ПЛК за допомогою багатьох поширених відкритих промислових мереж - Modbus Plus, FIPIO, Ethernet, Interbus, Profibus, DeviceNet ControlNet. Оскільки до складу TSX Momentum входить процесорний модуль, то на базі цих модулів може бути побудована самостійна невелика розподілена система керування. Модулі вводу-виводу можуть також безпосередньо приєднуватись до процесорного модуля TSX Premium через інтегровану мережу FIPIO.

Взагалі до центрального процесора через вбудований порт FIPIO можуть бути підключені модулі дискретних і аналогових входів-виходів типу TBX, віддалений ввід-вивід контролерів Momentum, панель управління оператора CCX17, перетворювачі змінної швидкості ATV 16, персональні комп'ютери та інші пристрої.

2. Програмування контролерів у середовищи 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 МЕК 61131-3: LD,ST, IL, FBD, SFC;
- поділу програми користувача на секції (Section), кожна з яких може бути написана на різних мовах програмування МЕК;
- 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 нелокалізованих (unlocated, не прив’язаних 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) занімацією технологічного процесу (подібно засобам HMI);

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

- використання вбудованого вікна тривог 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 починаєтьсязі зчитування входів та закінчується записом виходів. У ***циклічному режимі***,початок наступного виклику задачі MAST починається відразу по закінченнюобробки попередньої. Таким чином, час між викликами задачі MAST залежить відтривалості її виконання. У ***періодичному режимі***,

інтервал між викликами задачізадається в проекті UNITY PRO. Цей інтервал вибирається завідомо більшим замаксимальний час обробки задачі. Тобто, в більшості випадків, між викликамизадачі MAST контролер буде виконувати внутрішню обробку (діагностичні операції,комунікаційний обмін, тощо).

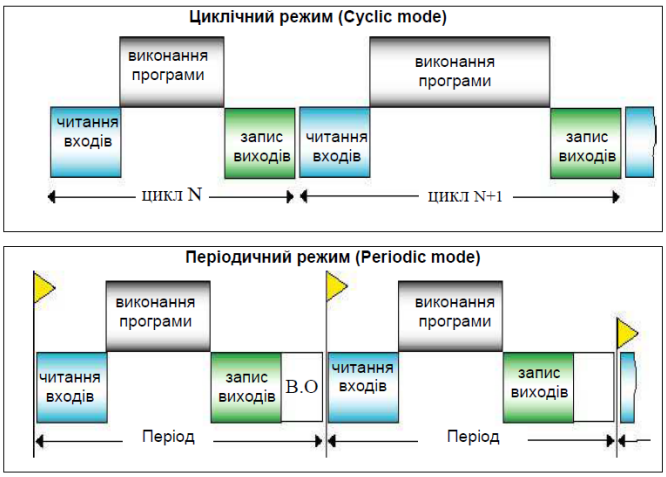


Рисунок 2.1–Циклічний та періодичний режим роботи задачі MAST

Програма задачі MAST представляє собою набір ***секцій*** (рис.2.2), кожна з яких може бути написана на будь-якій з мов МЕК 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".

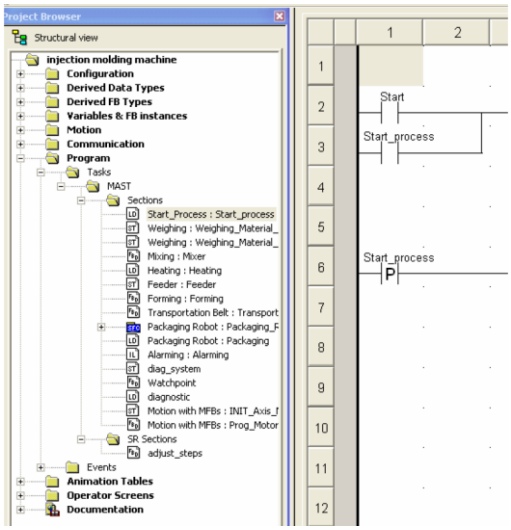


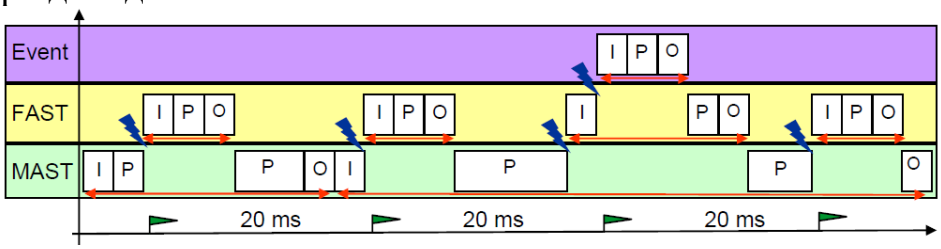
Рисунок 2.2–Приклад вигляду структури задачі MAST.

Задача FAST. У багатозадачній структурі програми М340, 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: ***EVTi*** – по апаратним подіям, ***TIMERi*** – по таймерним. Задачі EVTi (де і – номер задачі даного типу) прив'язуються до певної події, які фіксуються модулями спеціального призначення. Задачі TIMERi (де і – номер задачі даного типу)прив'язуються до таймера спеціального призначення, який запускається в програмі користувача спеціальною функцією ITCNTRL.

Пріоритетність задач. Задачі TIMERi мають вищий пріоритет ніж FAST, а EVTi – вище ніж TIMERi. Задачі з вищим пріоритетом будуть перериватиконання менш пріоритетних задач, які будуть продовжувати виконання задачіпісля закінчення виконання більш пріоритетної задачі. На рис.2.3 показанийприклад роботи контролера в багатозадачному режимі, з тривалістю періода задачіFAST - 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 увімкнені 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 символьним 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.

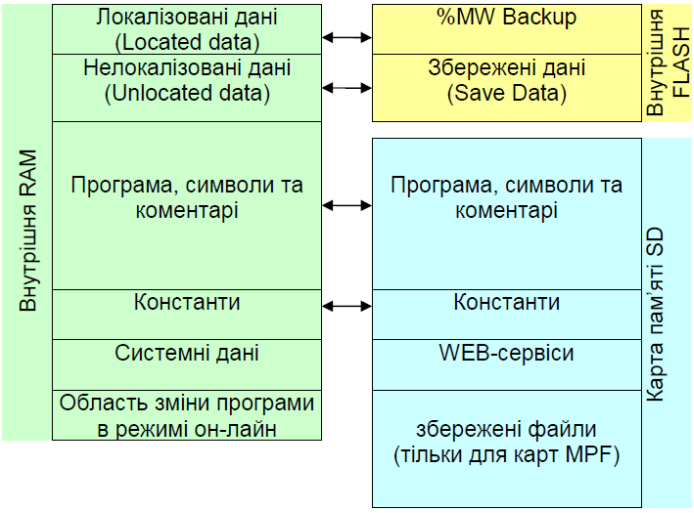


Рисунок 2.4 – Структура пам'яті M340

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.2.1Характеристика деяких елементарних типів даних (EDT).

Тип	Призначення	кількість біт, формат	Діапазон значень	форми відображення константи
BOOL	булевий	8	TRUE/FALSE	***TRUE*** або ***1***, ***FALSE*** або ***0***
EBOOL	булевий з можливістю форсування та визначення фронтів	8	TRUE/FALSE	***TRUE*** або ***1***, ***FALSE*** або ***0***
INT	ціле, числові операції	16, 16-й біт - знак	-32768...32767	десяткова, двійкова (2#), вісімкова(8#), шіснадцяткова(16#), наприклад: 3 – десяткова; ***2***#0000000000000011 - :двійкова; ***8***#000003 - вісімкова; ***16***#0003 - шіснадцяткова.
DINT	подвійне ціле, числові операції	32, 31-й біт - знак	0...65535	десяткова, двійкова (2#), вісімкова(8#), шіснадцяткова(16#)
WORD	слово, побітові операції	16	16#0…16#FFFF	двійкова (2#), вісімкова(8#), шіснадцяткова(16#)
REAL	реальне, число з плаваючою комою	32, 31-й біт – знак, 8 біт експонен-ційна, 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 ***DEN***: -1.1754944e-38 … 1.1754944e-38 Нескінченність: < 3.4028234e+38 –***INF*** >+3.4028234e+38 +***INF*** ***QNAN*** та ***SNAN*** – невірний формат числа
TIME	беззнакове подвійне ціле	32, зберігає значення в мілісекун-дах	0...4294967295 мс, або T#0MS...T#49D_17H_2M_47S_295MS	Часова константа ***T#***, 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).

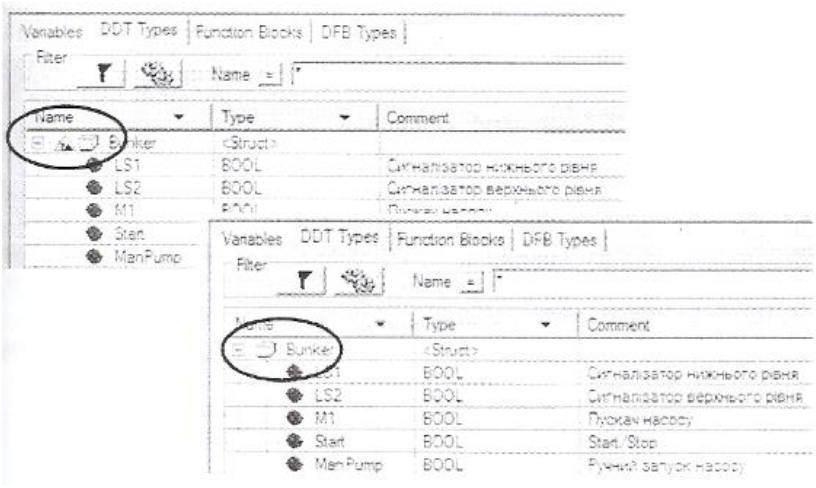


Рис. 2.5 Створення структурного типу користувача DDT угорі–до команди *Analyze*, внизу – після

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

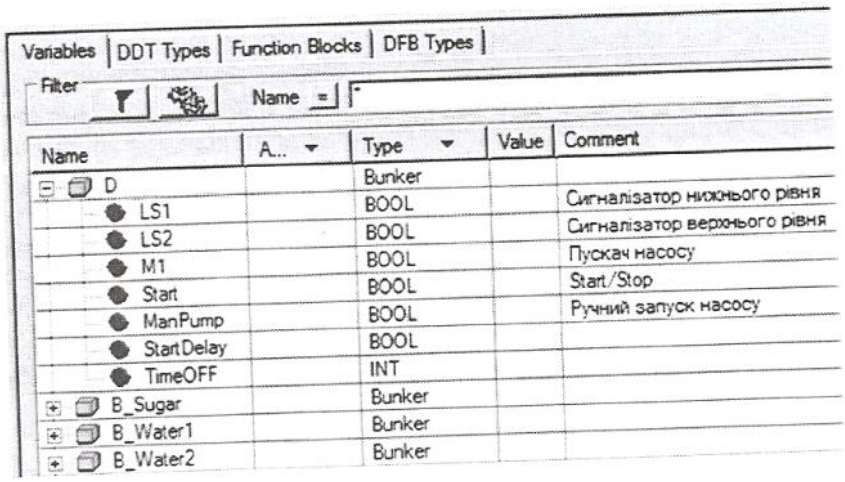
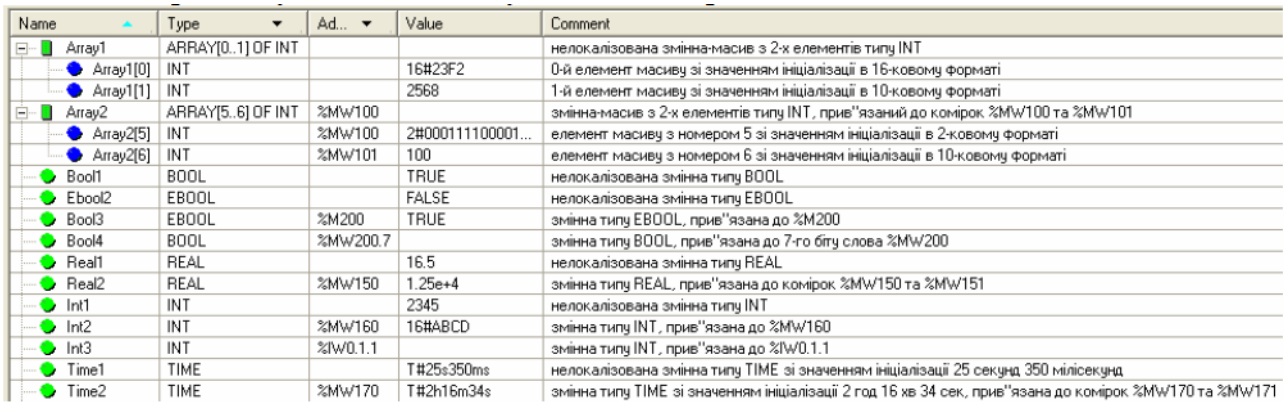


Рис.2.6 Структурні змінні.

Робота зі змінними в UNITY. UNITY дає можливість оперувати зданими 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 доданих 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#0001111000001...	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 елементу 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 елементу 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*.

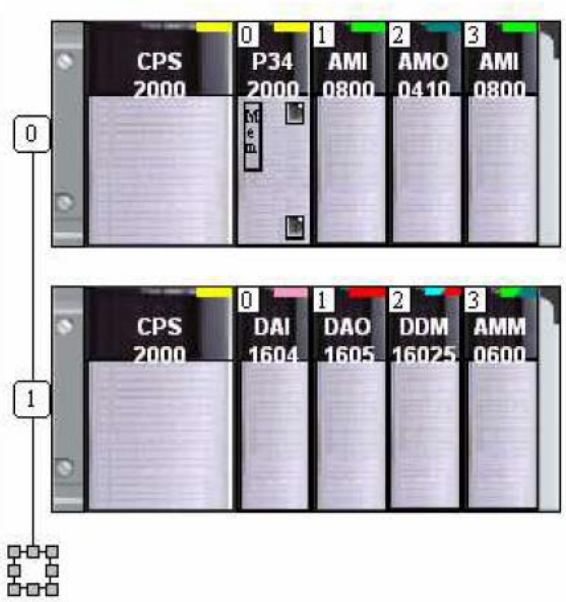
%IWr.m.c - адреса відповідає за аналоговий вхід на шасі з номером *r*, модуліна посадочному місці *m*, каналу *c*.

%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*,модулі на 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 AMM 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
			…	…
			3	%IW1.3.3
		AMM 0600 (виходи)	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 вміщувати всі внутрішні дані, тобто його пам'ять. Данідля 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 вхідних 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. Частину з них наведено в таб.3.1. Ряд блоків з бібліотеки управління, які використовуються при реалізації алгоритмів регулювання наведені у таб.3.2.

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

Математичні (сімейство Mathematics)		
ADD – додавання	SUB – віднімання	SQRT – квадратний корінь
MUL – множення	MOVE – присвоєння	
DIV – ділення	NEG – зміна знаку	
Порівняння (сімейство Comparison)		
EQ – дорівнює (=)	GE – більше або дорівнює (≥)	LE – менше або дорівнює (≤)
NE – не дорівнює (≠)	GT – більше (>)	LT – менше (<)
Логічні (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"	
Статистичні (сімейство Statistical)		
MUX – мультиплексор	LIMIT – обмеження, LIMIT_IND – обмеження з індикацією	MAX – вибір максимального
SEL – бінарний вибір	AVE – середнє значення	MIN – вибір мінімального
Таймери та лічильники (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
Перетворення типів (сімейство Type to Type)		
INT_TO_REAL	WORD_TO_BIT	WORD_TO_INT
REAL_TO_INT	BIT_TO_WORD	INT_TO_WORD
Цілочисельне регулювання (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 – маштабування	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:

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

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

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

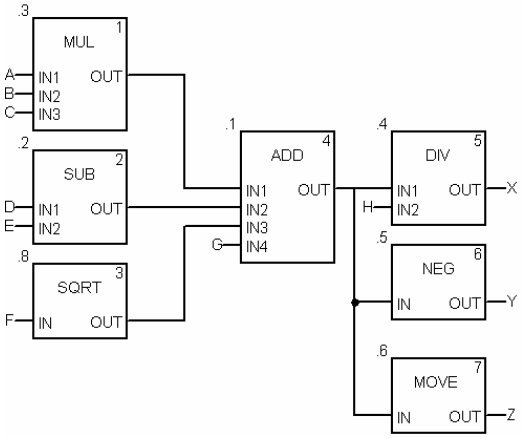


Рис.3.1. Приклад використання математичних функцій в FBD

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≥F, виконається дія Y:=D+F.

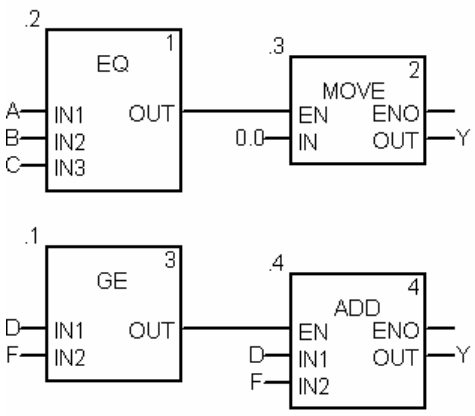


Рис.3.2. Приклад використання функцій порівняння в FBD

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

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≥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.

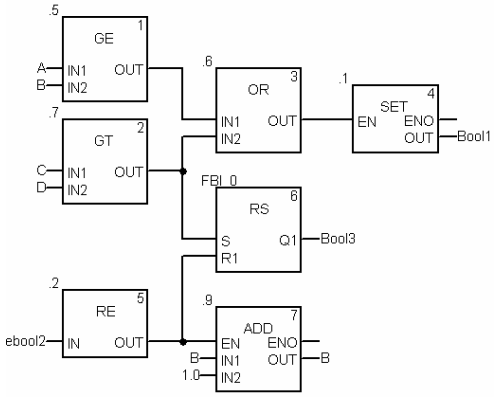


Рис.3.3. Приклад використання логічних блоків в FBD

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:

$$Y = \frac{\sum (Ki \cdot Xi)}{\sum (Ki)}$$

де Ki – коефіцієнт і-го вхідного значення; Xi – і-те вхідне значення.

Коефіцієнт і значення вхідного сигналу являються парою вхідних параметрів. Тобто три пари сигналів потребують шість входів (K_X1…K_X6). Непарні входи є коефіцієнтами, парні – значеннями сигналів.

Функція LIMIT забезпечує обмеження вхідної величини IN по мінімуму (вік 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" переключає 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.

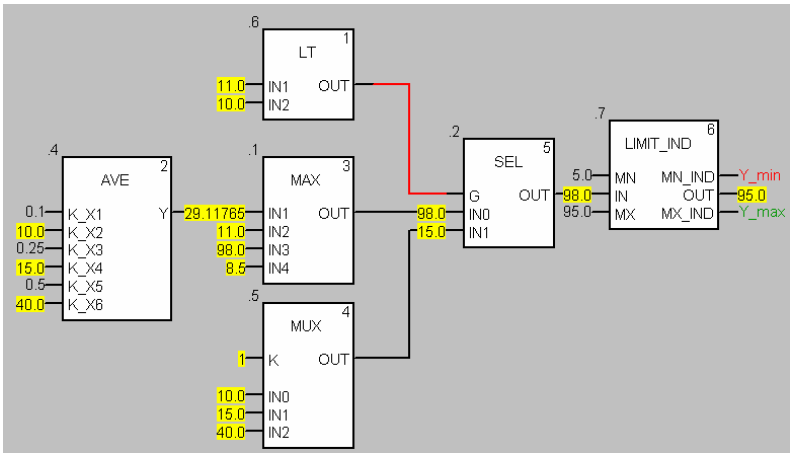


Рис.3.4. Приклад використання статистичних функцій в FBD в режимі анімації

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: CTU – підрахунок імпульсів зі збільшенням, CTD – підрахунок імпульсів зі зменшенням, CTUD – реверсивний лічильник.

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

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

Лічильник CTUD, об’єднує в собі функції двох лічильників CTU та CTD. Входи CU та CD призначені відповідно для збільшення та зменшення плинного значення лічильника CV. При CV≥PV, вихід QU:=TRUE. При CV≤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.

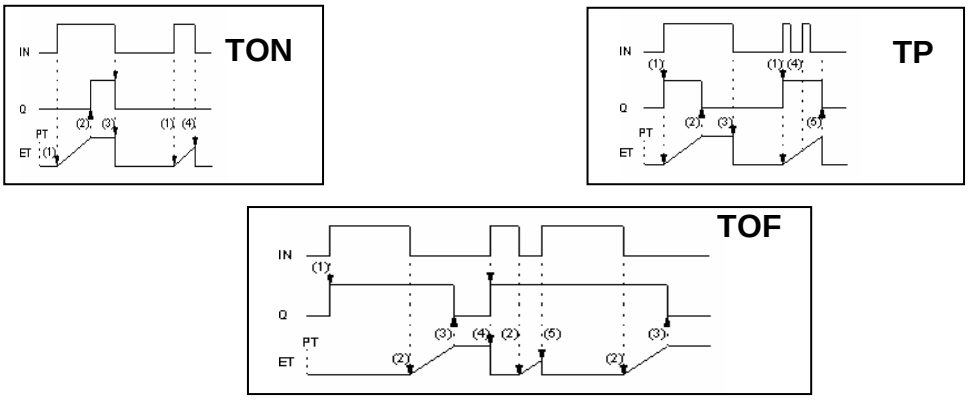


Рис.3.5. Графіки режимів роботи таймерів TON, TOF та TP.

Приклад використання лічильника CTUD 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 100 мс 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".

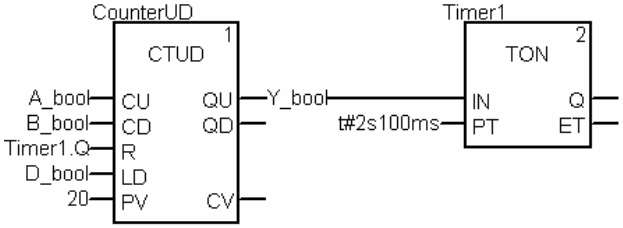


Рис.3.6. Приклад використання функціональних блоків TON та CTUD

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.

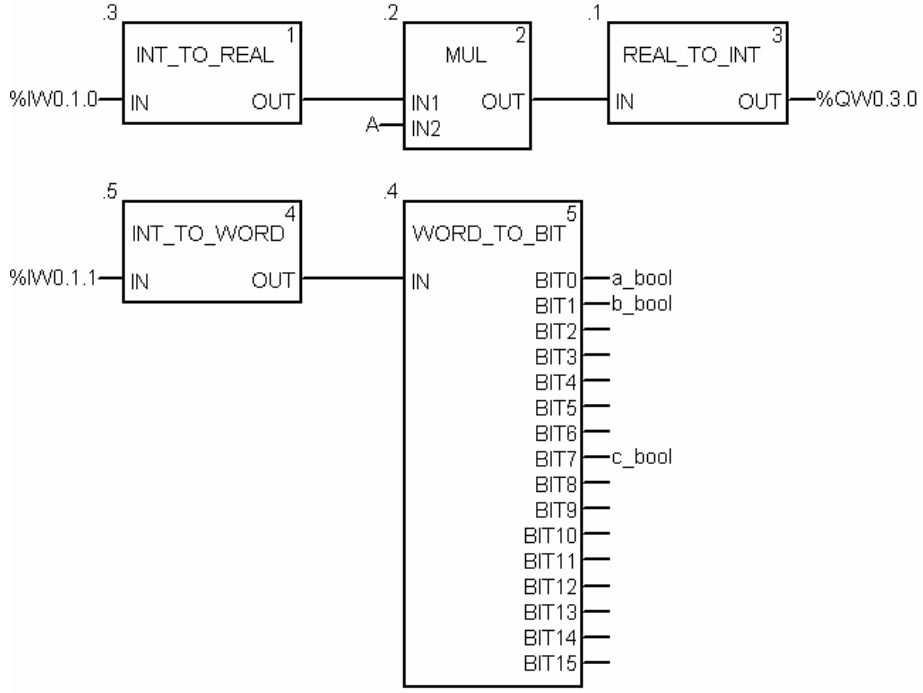


Рис.3.7. Приклад використання функцій перетворення типів в FBD

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 організовують 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 «no».

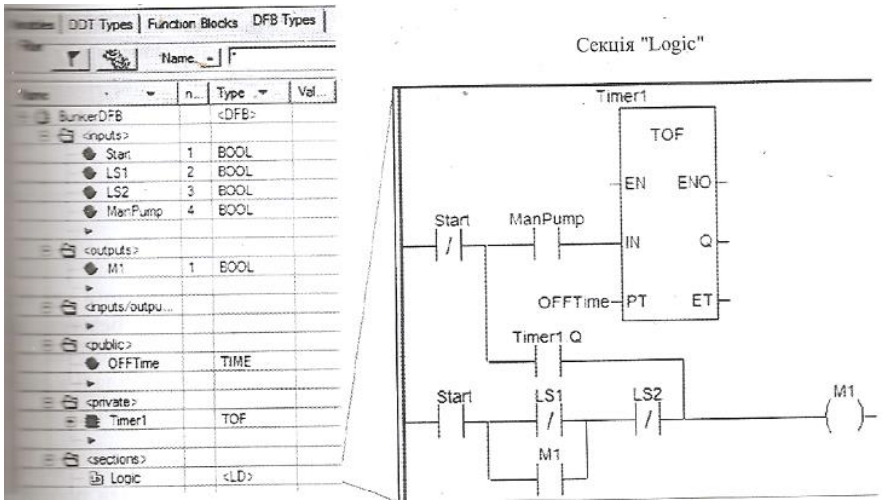


Рис 3.1 Створення типу функціонального блока користувача DFB.

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 «prіvate». 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).

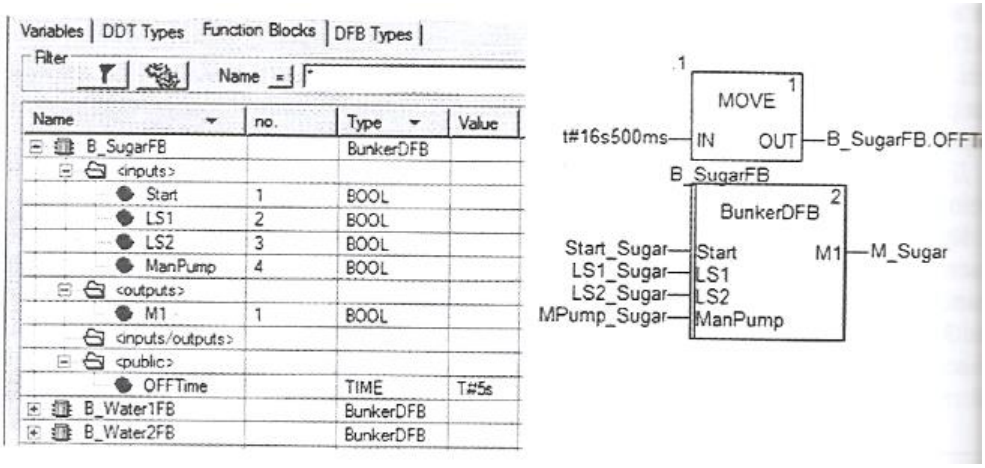


Рис.3.2 Створеення екземплярів функціонального блока користувача DFB та їх використання в програмі

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*).

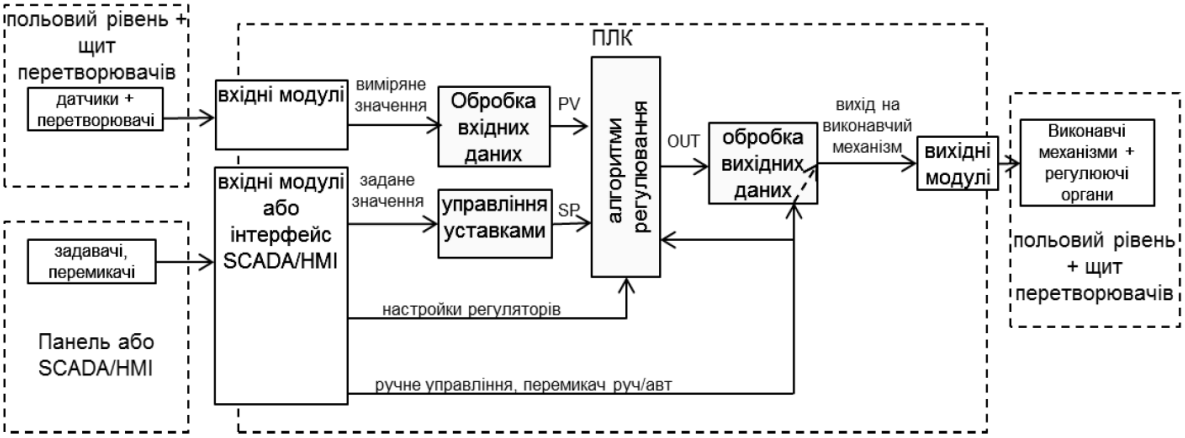


Рисунок 4.1– Структура контурів регулювання

Сімейства бібліотеки *Obsolete Lib* не рекомендується використовувати у новостворюваних проектах UNITY PRO. Процедури сімейства *CLC_INT* бібліотеки *StandardLib* реалізовують 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* (з-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 слідкування (Tracking)

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

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

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

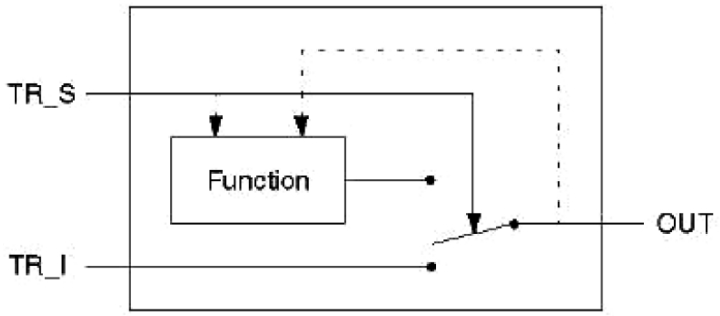


Рисунок 4.2– Режим слідкування

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 слідкування*** (***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 виконанню блоку;

2) режим слідкування функціонального блоку в дубльованому ПЛК (в системах Hot Standby TSX Premium/Quantum), для гарантування безударності запуску резервного контролеру;

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*).

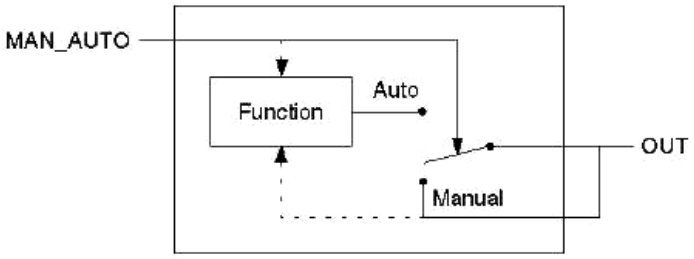


Рисунок 4.3. Режим Руч/Авт

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*.

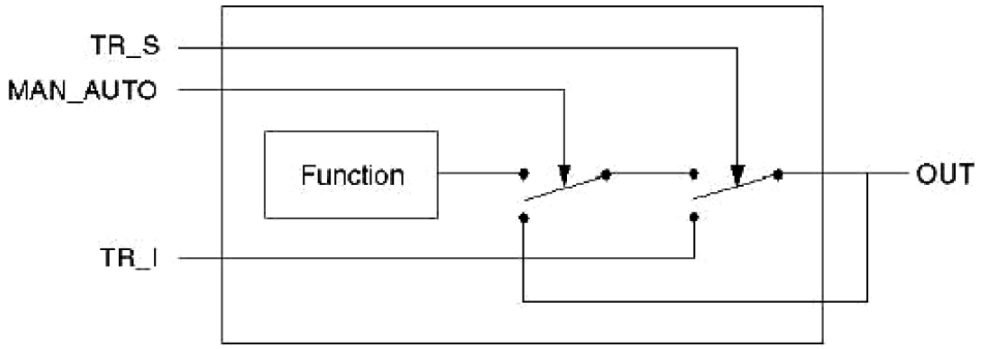


Рисунок 4.4– Пріоритетність режимів.

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.

5.1Загальне 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_B – реалізовує ПІ закон регулювання;
- PIDFF – реалізовує ПІД закон регулювання;
- AUTOTUNE – для автонастройки PI_B 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 (широтно-імпульсна модуляція) для управління дискретними виконавчими механізмами по аналоговому сигналу та може бути використаний разом з PI_B 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_B 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 налокальний 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.

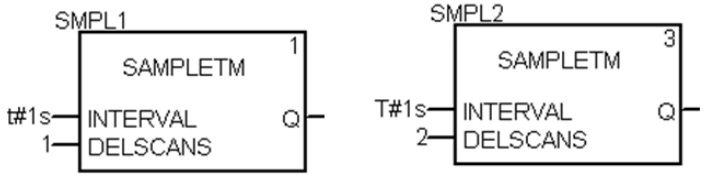


Рисунок 5.1– Використання 2-х екземплярів SAMPLETM зі зсувом на один цикл відносно

5.2. ПІ регулятор PI_B

Функціональний блок *PI_B* реалізовує ПІ алгоритм регулювання (5.1).

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

Функціональна схема ПІ-регулятора зображена на рис.5.2. Параметри *PI_B* 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).

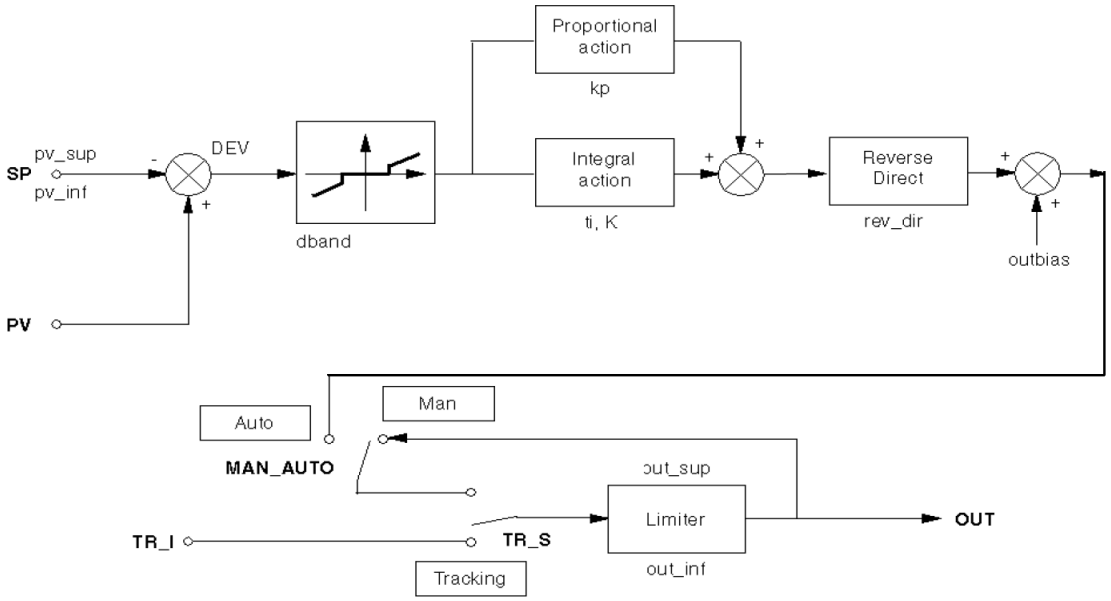


Рисунок 5.2– Функціональна схема ПІ-регулятора PI_B.

У автоматичному режимі (*MAN_AUTO=TRUE*) розраховане значення, пройшовши через блоки *Tracking* та *Limiter* подається на вихід *OUT*. В ручному режимі (*MAN_AUTO=TRUE*) вихід *PI_B* 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_rcpy=TRUE* обмеження не діють).

Функціональний блок *PI_B* може працювати в режим ПІ та П регулятора. Виставивши параметр *ti=0s*, блок *PI_B* переключається в режим П-регулювання. У режимі ПІ регулювання блок розраховує вихід *OUT* по інкрементальному розрахунку, а в П 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-регулятору використовується функціональний блок типу *SAMPLETM*.

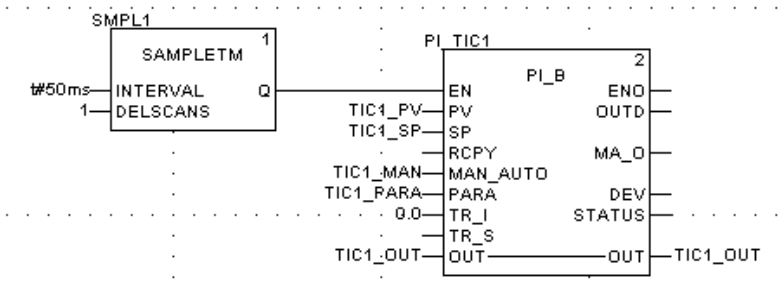
Настройка регулятора проводиться через структурну змінну *TIC1_PARA*, яка попередньо створюється на базі типу *Para_PI_B*.

Таблиця 5.1 Параметри функціонального блока *PI_B*.

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

Таблиця 5.2. Опис структурного типу *Para_PI_B*

id	*UINT*	Використовується для алгоритму автопідстройки (*AUTOTUNING*)
pv_inf	*REAL*	обмеження по мінімуму вхідної величини завдання
pv_sup	*REAL*	обмеження по максимуму вхідної величини завдання
out_inf	*REAL*	обмеження по мінімуму вихідної величини блоку
out_sup	*REAL*	обмеження по максимуму вихідної величини блоку
rev_dir	*BOOL*	0: пряма робота регулятору (*SP-PV*) 1: зворотна робота регулятору (*PV-SP*)
en_rcpy	*BOOL*	1: використати вхід *RCPY*
kp	*REAL*	Коефіцієнт пропорційності
ti	*TIME*	Час інтегрування
dband	*REAL*	Зона нечутливості
outbias	*REAL*	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 функції. 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.

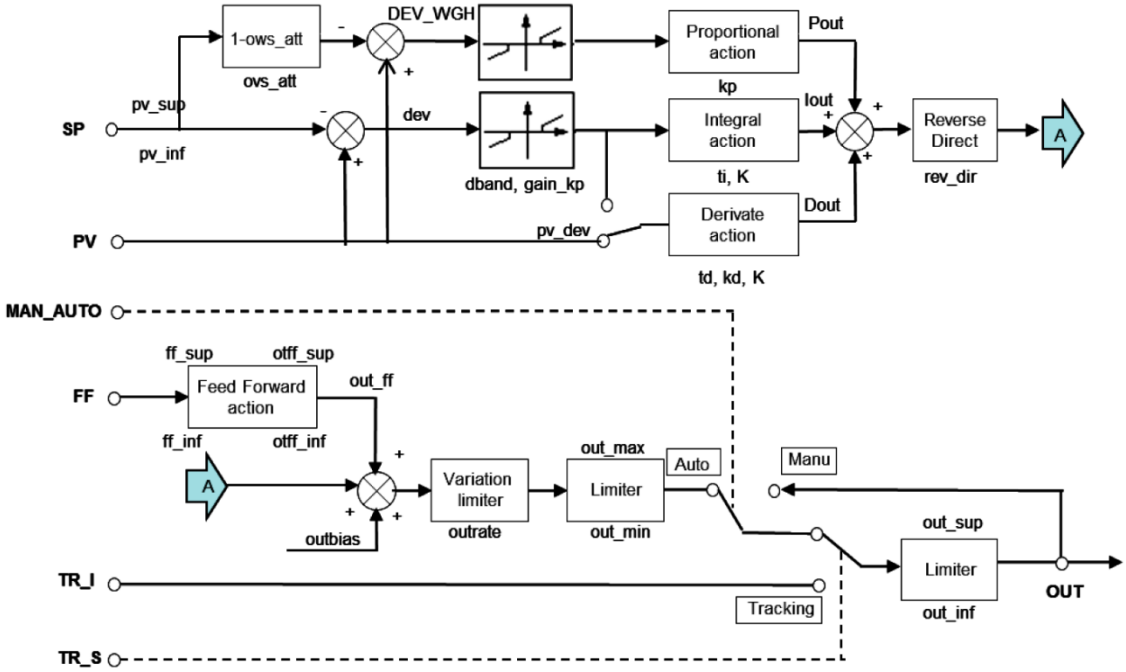


Рисунок 5.4– Функціональна схема ПІД-регулятора PIDFF.

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

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

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

У формулах (5.3) та (5.4) k_p- коефіцієнт пропорційності, t_i- час інтегрування, t_d- час диференціювання, k_d- коефіцієнт диференціювання, $dev = (PV - SP)$- розузгодження, α- коефіцієнт масштабування, який розраховується за формулою

$$\alpha = \frac{out\_\sup - out\_\inf}{pv\_\sup - pv\_\inf} \qquad (5.5)$$

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

$$K = \alpha = \frac{out\_\sup - out\_\inf}{pv\_\sup - pv\_\inf} \qquad (5.6)$$

При включеній інтегральній складовійt_i>0, вхід для пропорційної складової розраховується за формулою:

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

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 регулятору 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:

$$out\_ff = \frac{(FF - ff\_\inf) \cdot (otff\_\sup - otff\_\inf)}{(ff\_\sup - ff\_\inf)} + otff\_\inf \qquad (5.8)$$

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

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

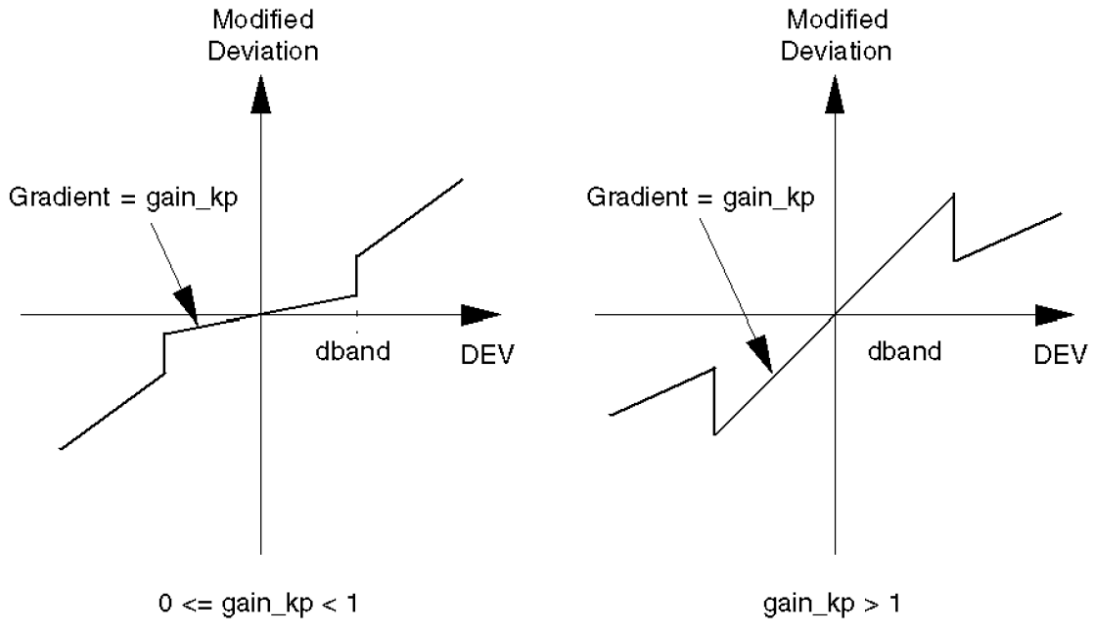


Рисунок 5.5– Настройка зони нечутливості.

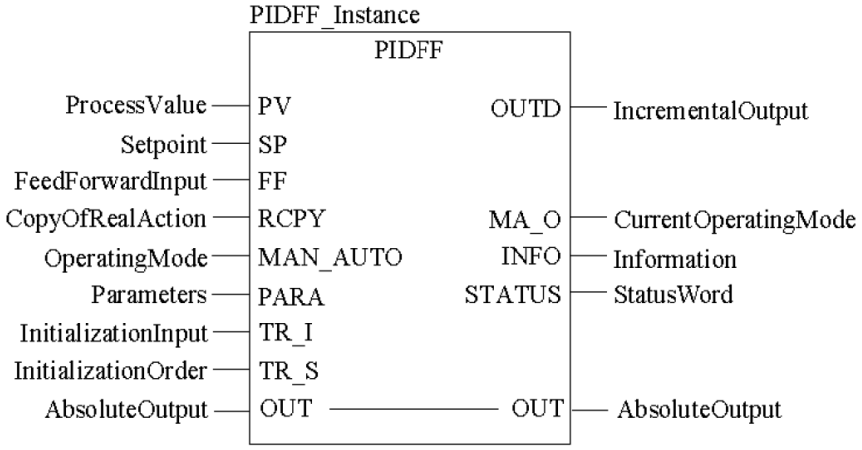


Рисунок 5.6– Приклад виклику екземпляру "*PIDFF_Instance*" типу *PIDFF* в секції на мові FBD.

Таблиця 5.3 Параметри функціонального блока *PIDFF*.

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

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

Таблиця 5.4. Опис структурного типу *Para_PIDFF*.

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

		(при *ti=0s*)
out_min	*REAL*	нижня межа для виходу регулятору
out_max	*REAL*	верхня межа для виходу регулятору
outrate	*REAL*	обмеження по швидкості зміни на виході одиниць/сек (≥0)
ff_inf	*REAL*	нижня межа значення *FF*
ff_sup	*REAL*	верхня межа значення *FF*
otff_inf	*REAL*	нижня межа значення *out_ff*
otff_sup	*REAL*	верхня межа значення *out_ff*

Таблиця 5.5. Опис структурного типу Info_PIDFF .

dev	REAL	розузгодження (*PV-SP*)
out_ff	REAL	значення упередження

5.4 Двохпозиційний регулятор *STEP2*

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

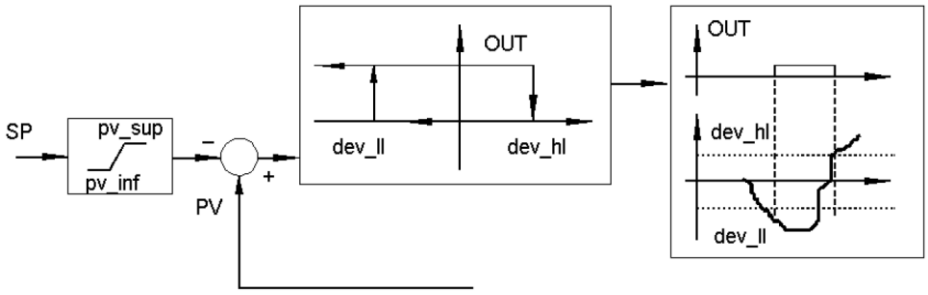


Рисунок 5.7– Функціональна схема 2-х позиційного регулятора *STEP2*

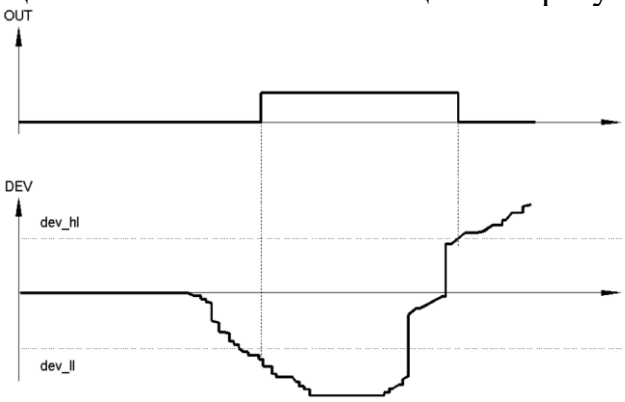


Рисунок 5.8– Графік залежності виходу *OUT* від розузгодження *DEV*

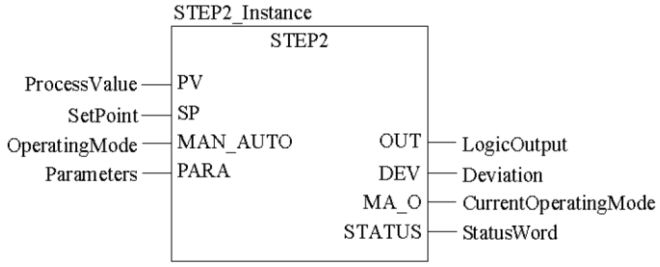


Рисунок 5.9– Приклад використання функціонального блоку *STEP2*

Таблиця 5.6 Параметри функціонального блока *STEP2*.

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

Таблиця 5.7. Опис структурного типу *Para_STEP2* .

dev_ll	*REAL*	нижній поріг відхилення від заданої величини *(≥0)*
dev_hl	*REAL*	верхній поріг відхилення від заданої величини *(≤0)*
pv_inf	*REAL*	обмеження по мінімуму вхідної величини завдання
pv_sup	*REAL*	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.

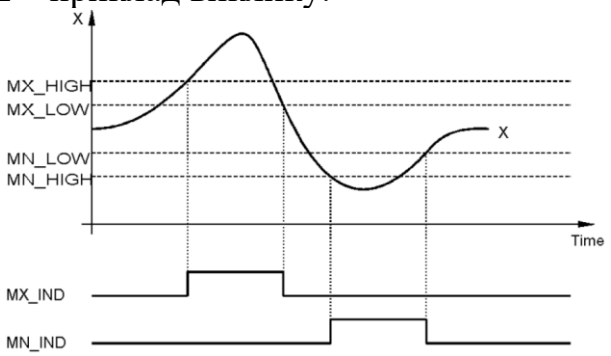


Рис.6.1. Робота блоку *INDLIM*

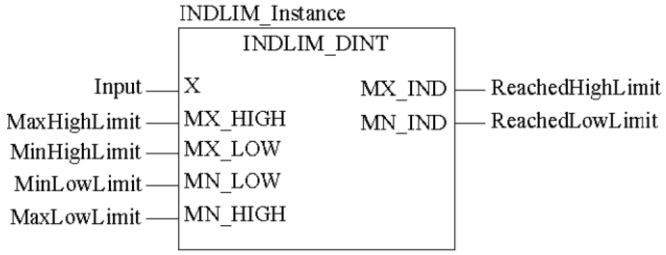


Рис.6.2. Приклад виклику блоку *INDLIM* .

Таблиця 6.1 Параметри функціонального блока *INDLIM*.

Вхідні параметри		
X	*INT, DINT, UINT, UDINT, REAL*	Вхідна змінна
MX_HIGH	*INT, DINT, UINT, UDINT, REAL*	Максимальна верхня межа
MX_LOW	*INT, DINT, UINT, UDINT, REAL*	Максимальна нижня межа
MN_LOW	*INT, DINT, UINT, UDINT, REAL*	Мінімальна нижня межа
MN_HIGH	*INT, DINT, UINT, UDINT, REAL*	Мінімальна верхня межа
Вихідні параметри		
MX_IND	*BOOL*	Відображення досягнення верхньої межі
MN_IND	*BOOL*	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 *XiYi* (2 значення на кожну точку) формується вихід *Y*. Кількість вузлових точок варіюється до *15*, кожна задається парою значень *XiYij*: перша для *X* (непарні номера *j*), друга для *Y*(парні номера *j*). Таким чином процедура може мати до *30+1* входів.

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

$$Y = \begin{cases} Y_1, при\_X < X_1 \\ Y_i + \dfrac{Y_{i+1} - Y_i}{X_{i+1} - X_i} \cdot (X - X_i), при\_X_1 \le X \le X_m \\ Y_m, при\_X > X_m \end{cases} \quad (6.1)$$

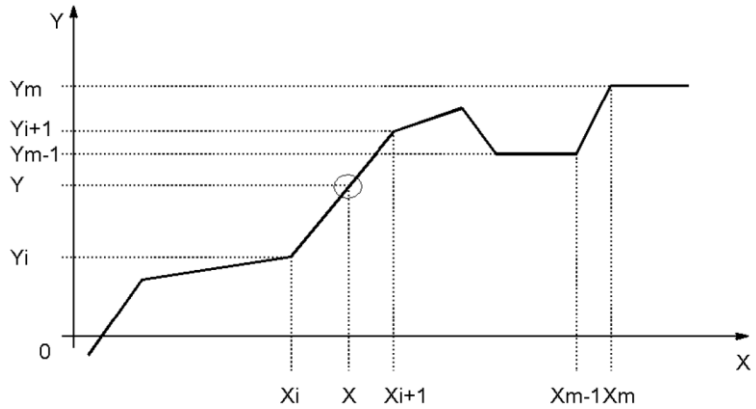


Рисунок 6.2– Робота процедури *LOOKUP_TABLE1*

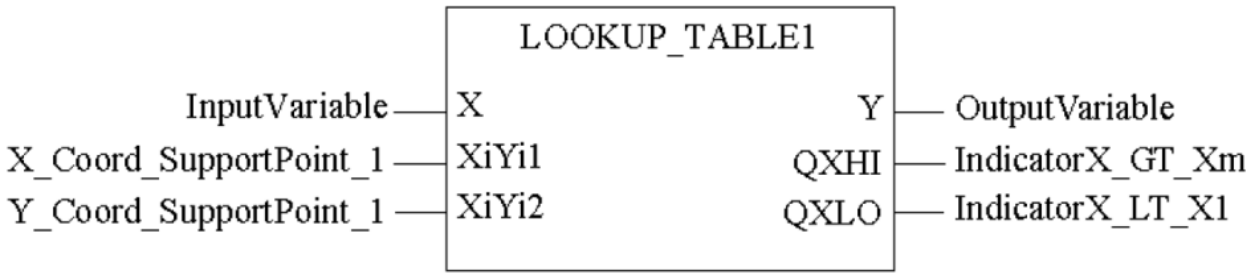


Рисунок 6.3– Робота функції *LOOKUP_TABLE1*

Таблиця 6.1 Параметри процедури *LOOKUP_TABLE1*.

Вхідні параметри		
X	*REAL*	Вхідна змінна
XiYi1	*REAL*	координата X_1.
XiYi2	*REAL*	координата Y_1.
XiYi(n-1)	*REAL*	координата $X_{n/2}$; *n=max 30*
XiYi(n)	*REAL*	координата $Y_{n/2}$; *n=max 30*
Вихідні параметри		
Y	*REAL*	Вихідна змінна
QXHI	*BOOL*	Індикація $X > Xm$
QXLO	*BOOL*	Індикація $X < X1$

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

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

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

де *N* – кількість значень в буфері (ширина вікна), *X* – вхідне не фільтроване значення, *Y* – середнє вихідне (фільтроване) значення, *Yold* - значення виходу на попередньому виклику.

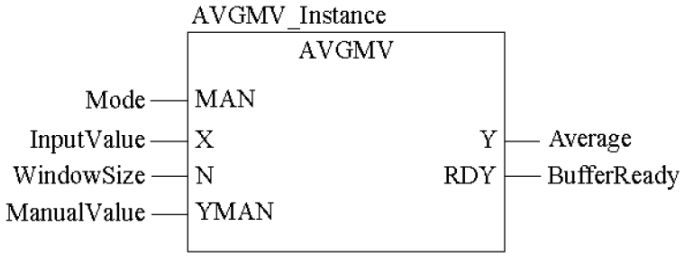


Рисунок 6.4. Виклик функціонального блоку *AVGMV*

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

YMAN	*REAL*	Ручне значення
Вихідні параметри		
Y	*REAL*	Середнє значення
RDY	*BOOL*	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:

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

де *Q*-об'ємна витрата (м3/год), *d* – діаметр отвору (мм), *p* -густина (кг/м3), *(p1-p2)* – перепад тиску на звужуючому пристрої (кгс/м2), *ε*– коефіцієнт розширення (для рідин=1), *α*- коефіцієнт витрати який розраховується за формулою:

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

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

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

де *dP=P1-P2*, *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:

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

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:

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

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

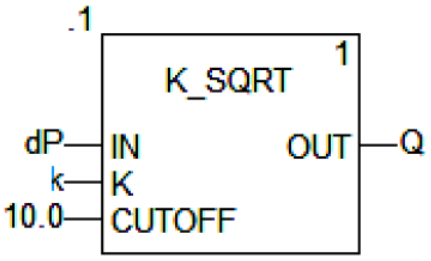


Рисунок 6.5. Приклад використання функції K_SQRT

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

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 *PARA*.

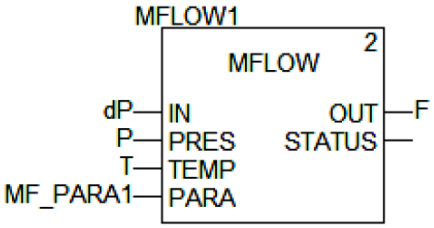
Вхід *PRES* – це тиск, який може задаватися як в абсолютних одиницях так і відносно атмосферного. У випадку відносного тиску (*PARA.pr_pa=FALSE*), абсолютний тиск розраховується за формулою :

$$PA=PRES+pu \qquad (6.9)$$

де *pu* задається в тій же структурі *PARA*.

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

Корекція по тиску активується параметром *PARA.en_pres=TRUE*, по температурі - *PARA.en_temp=TRUE*. Якщо *PARA.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 корня

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 управляння 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*.

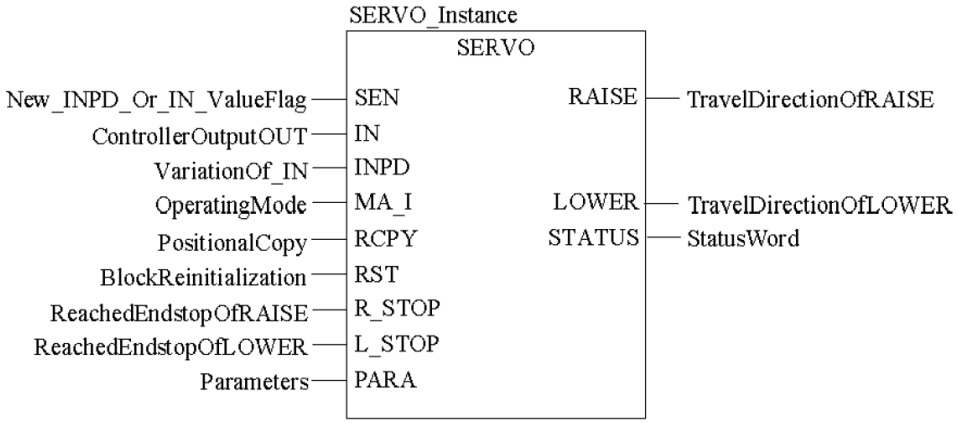


Рис.7.1. Виклик функціонального блоку *SERVO*.

Таблиця 7.1 Параметри блоку *SERVO*

Вхідні параметри		
SEN	*BOOL*	1 : є нові значення на входах *INPD* або *IN* 0 : немає нових значень на входах *INPD* або *IN*
IN	*REAL*	значення з абсолютного виходу блоку управління *OUT* (0 до 100%)
INPD	*REAL*	значення з інкрементального виходу блоку управління *OUTD* (-100% до 100%)
MA_I	*BOOL*	значення операційного режиму блоку регулятору до якого підключений *SERVO* (вихід *MA_O*) 1 : Автоматичний режим 0 : інший режим (ручний або слідкування)
RCPY	*REAL*	Зворотній зв'язок по позиції ВМ (0 до100%)
RST	*BOOL*	1 : Ініціалізація функціонального блоку (відключення виходів та обнуління внутрішнього статусу блоку)
R_STOP	*BOOL*	кінцева позиція "відкритий" (досягнутий поріг по *RAISE*)
L_STOP	*BOOL*	кінцева позиція "закритий" (досягнутий поріг по *LOWER*)
PARA	*Para_SERVO*	Параметри (див. таб 6.24)

Вихідні параметри		
RAISE	*BOOL*	Вихід "більше" ("відкрити")
LOWER	*BOOL*	Вихід "менше" ("закрити")
STATUS	*WORD*	Слово статусу

Таблиця 7.2 Опис структурного типу *Para_SERVO*

en_rcpy	*BOOL*	1 : Використовується зворотній зв'язок по позиції ВМ (включити *RCPY*)
rcpy_rev	*BOOL*	1 : Інвертувати *RCPY* (0 – це 100%, 100 – це 0%) 0 : Не інвертувати *RCPY*
t_motor	*TIME*	час повного відкриття регулюючого органу
t_mini	*TIME*	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* регулятору *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* регулятору *PI_B/PIDFF*. Для кожного нового значення виходу *OUTD* регулятору, 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 регулятору.

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* регулятору), 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* регулятору, він автоматично розрахує *OUTD*, який протримається на виході аж до наступного виклику регулятору. 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 регулятору 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*.

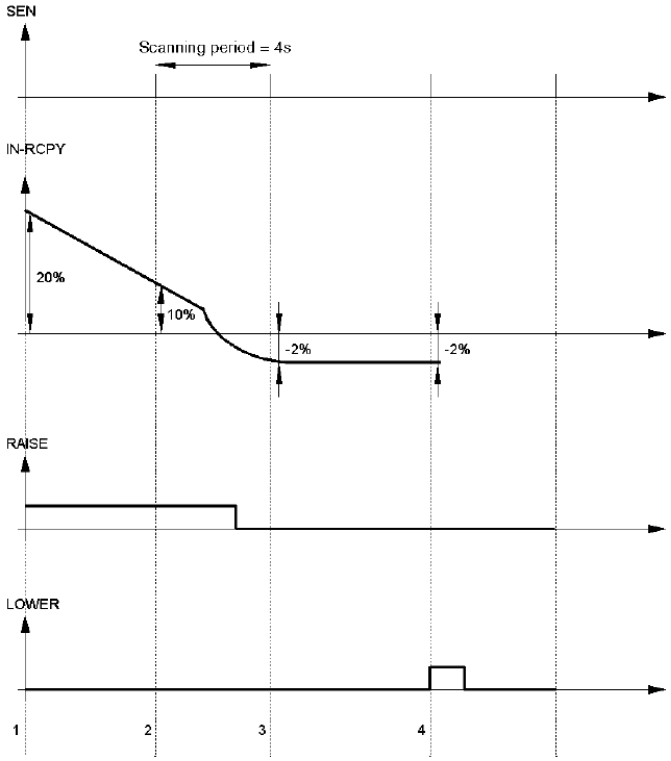


Рисунок 7.2– Приклад діаграми роботи блоку SERVO зі зворотним зв'язком по положенню ВМ.

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 регулятору (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 одної 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;

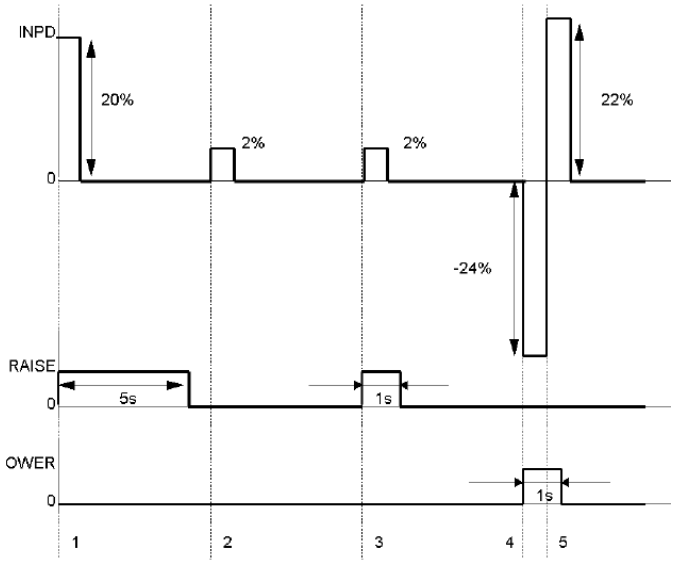


Рисунок 7.3– Приклад діаграми роботи блоку SERVO без зворотного зв'язку по положенню ВМ.

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 одної секунди;

4. При появі значення *INPD= -24%:* це відповідає імпульсу 6s 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.

$$T\_on(OUT\_POS) = t\_period \cdot \frac{IN}{in\_\max} \text{ при } 0 \le IN \le in\_\max$$

$$T\_on(OUT\_NEG) = t\_period \cdot \frac{|IN|}{in\_\max} \text{ при } 0 \le -IN \le in\_\max \quad (7.1), \quad (7.2)$$

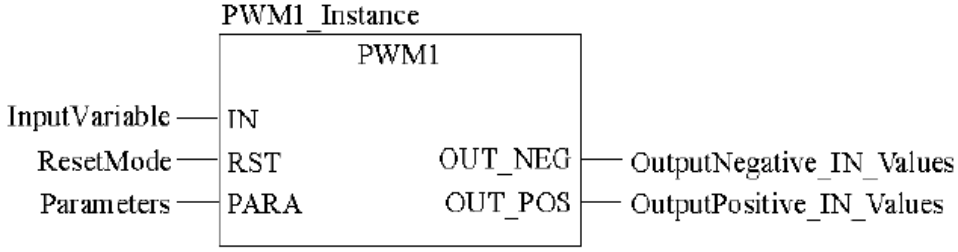


Рисунок 7.4– Приклад виклику блоку PWM1.

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*

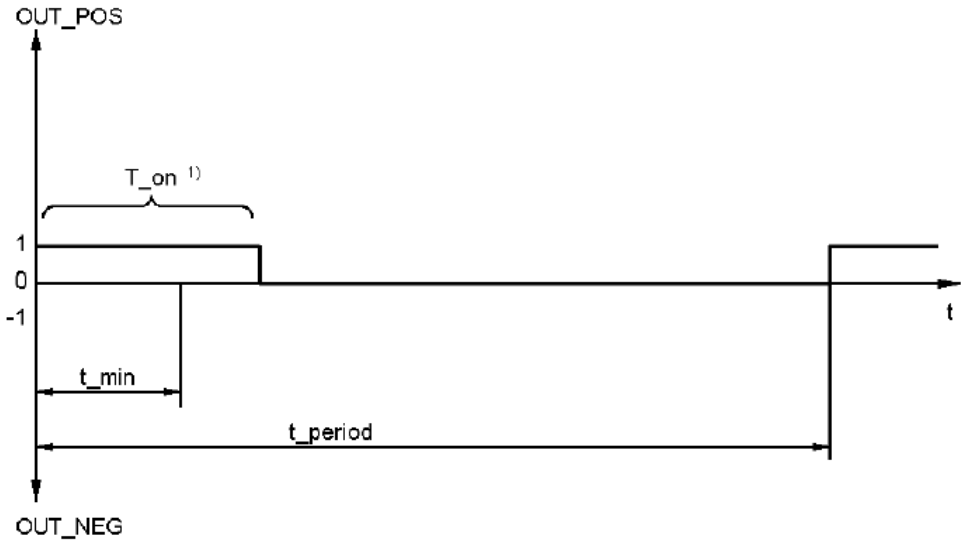


Рисунок 7.5– Діаграма зміни виходів блоку *PWM1*.

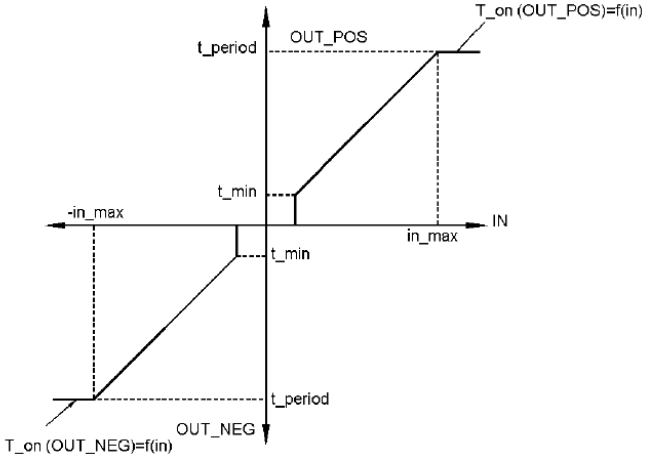


Рисунок 7.6– Статична характеристика *t_period(IN)* для *PWM1*.

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 регулятору. 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 *ТЕН* (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.

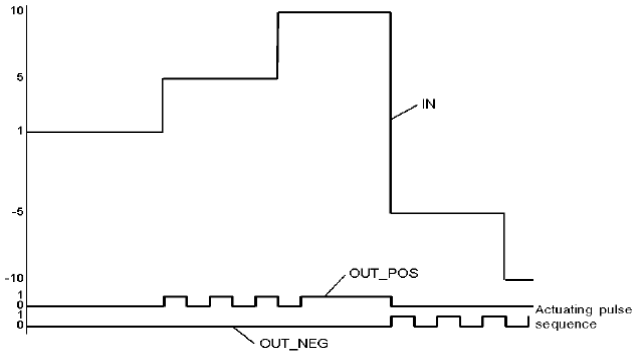


Рисунок 7.7– Приклад діаграми роботи PWM1.

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.

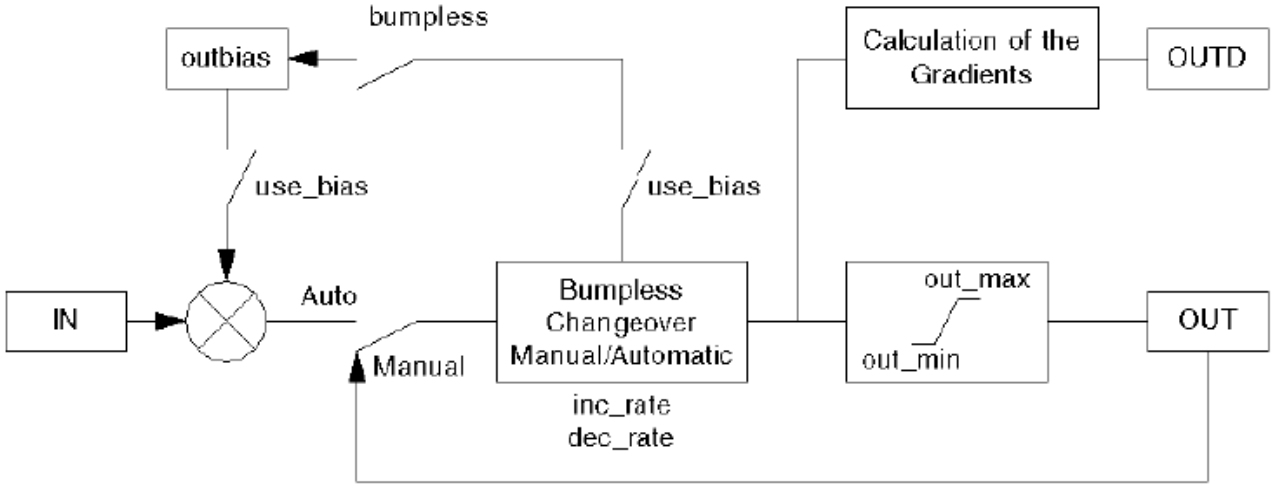


Рис.7.8.Функціональна схема блоку *MS*.

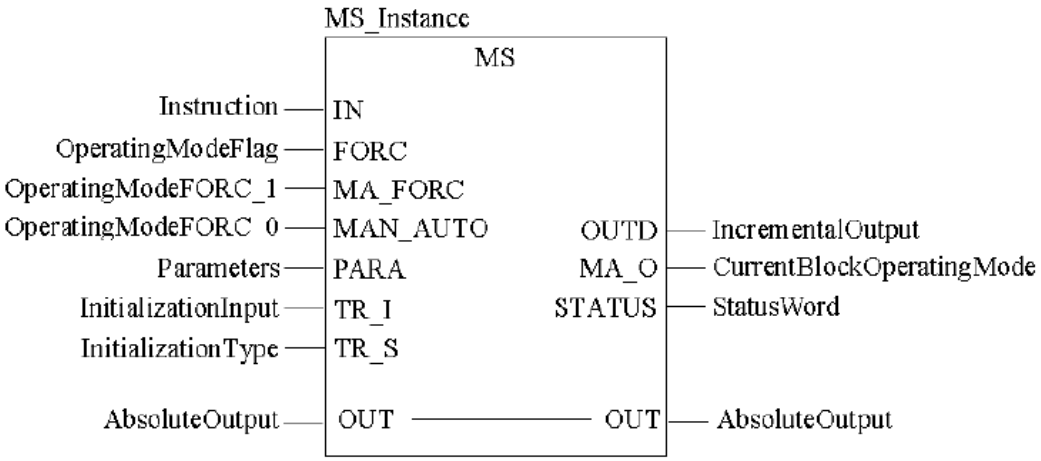


Рисунок 7.9–Приклад виклику блоку *MS*

Таблиця 7.5 Параметри блоку *MS*

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

Таблиця 7.6 Опис структурного типу Para_MS

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

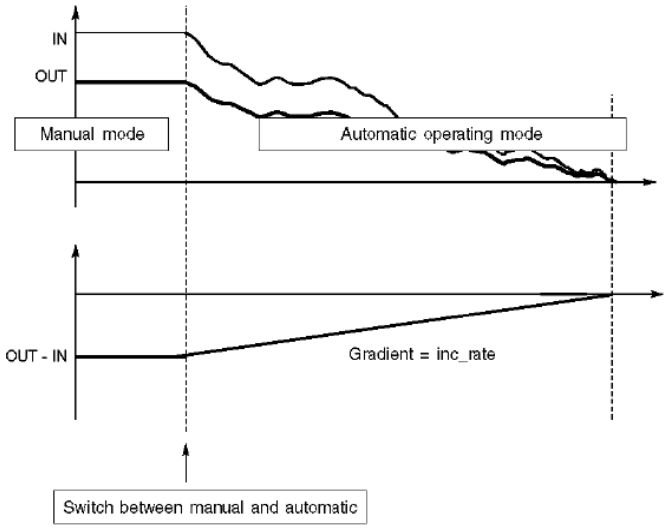


Рисунок 7.10–Діаграма роботи безударного переходу руч./авт. в *MS*.

Для реалізації безударності переходу з ручного в автоматичний режим (*bumpless=TRUE*) використовуються лінійні переходи з обмеженням по швидкості, які настроюються 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).

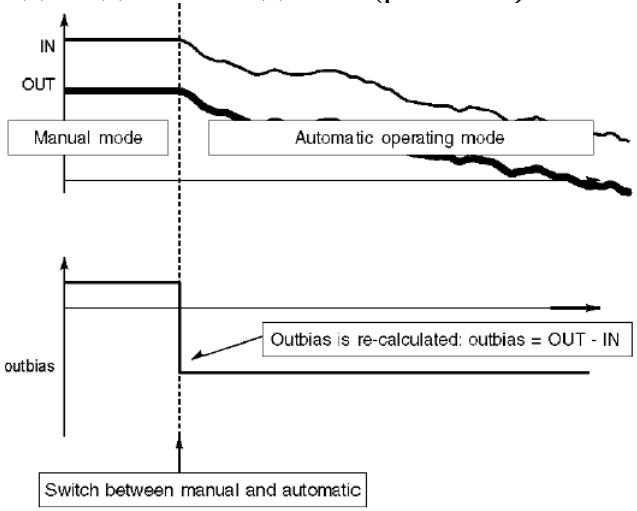


Рисунок 7.11. Діаграма роботи безударного переходу руч./авт.
зі зміщенням в *MS*.

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 регулятору. 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*.

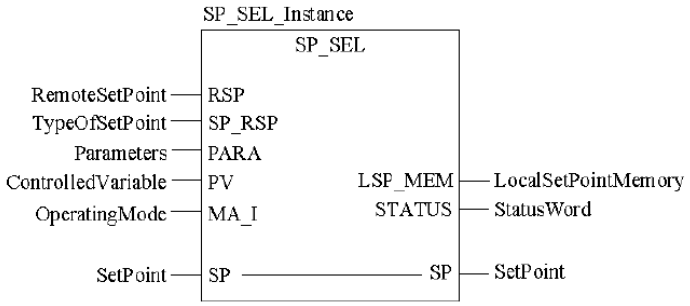


Рисунок 8.1– Приклад виклику блоку *SP_SEL*.

Таблиця 8.1 Параметри блоку *SP_SEL*

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

Таблиця 8.2 Опис структурного типу *Para_SP_SEL*

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

8.2 Задатчик співвідношення (*RATIO*)

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

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

де *PV_TRACK* – значення змінної, відносно якої розраховується завдання; *bias* – зміщення.

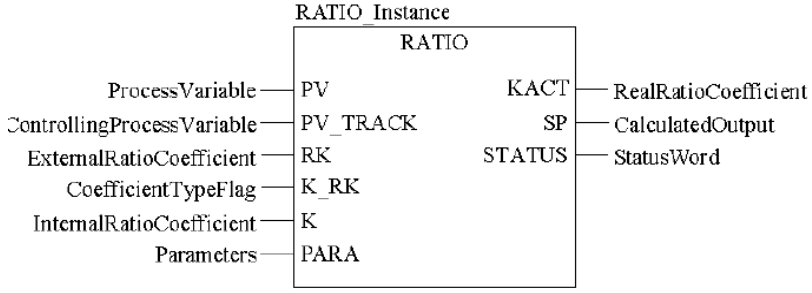


Рисунок 8.2– Приклад виклику блоку *RATIO*.

Таблиця 8.3 Параметри блоку RATIO

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

Таблиця 8.4 Опис структурного типу Para_RATIO

k_min	*REAL*	обмеження по мінімуму *K* та *RK*
k_max	*REAL*	обмеження по максимуму *K* та *RK*
sp_min	*REAL*	обмеження по мінімуму для розрахованого *SP*
sp_max	*REAL*	обмеження по максимуму для розрахованого *SP*
bias	*REAL*	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*;

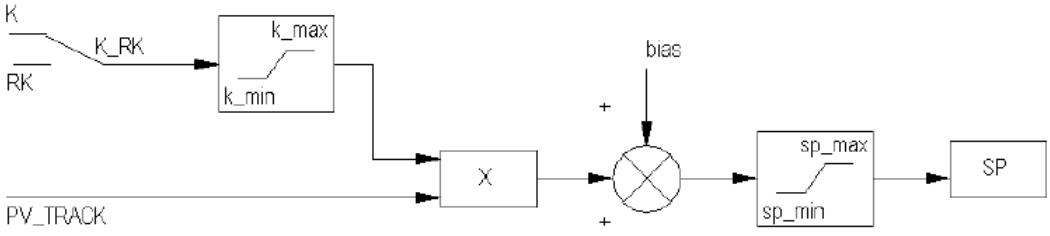


Рисунок 8.3– Функціональна схема RATIO.

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 регулятору, створеного користувачем.

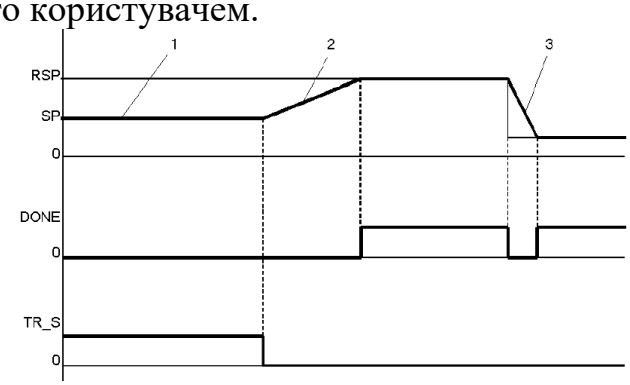


Рис.8.4. Приклад виклику блоку RAMP.

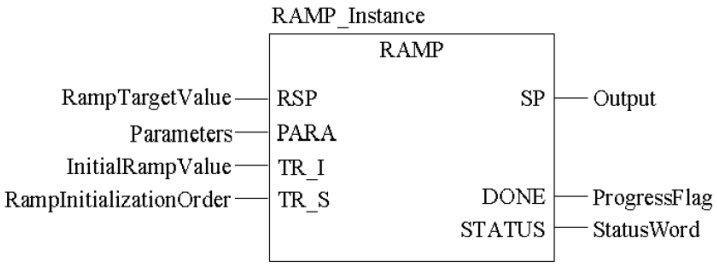


Рис.8.5. Приклад виклику блоку RAMP.

Таблиця 8.5

Параметри блоку *RAMP*

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

Таблиця 8.6

Опис структурного типу *Para_RAMP*

inc_rate	*REAL*	Швидкість нарощування в одиницях/секунду (≥0)
dec_rate	*REAL*	Швидкість спаду в одиницях/секунду(≥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:

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

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 *PARA* 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).

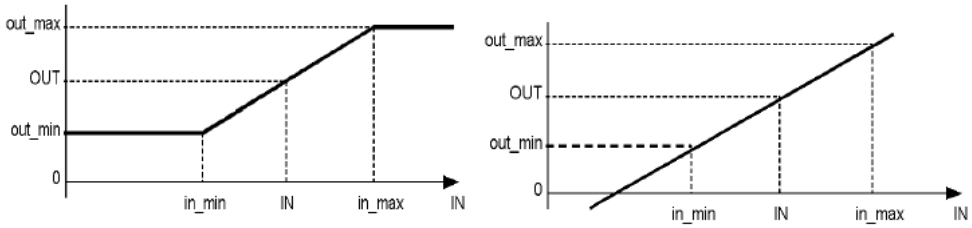


Рис.9.1. Залежність масштабованого вихідного значення (*OUT*) від вхідного (*IN*) при обмеженні на вихідний сигнал (зліва, *Clip*=1) та без обмеження (праворуч, *Clip*=0) для блока *SCALING*.

Функціонування блоку продемонструємо на прикладі масштабування вхідного аналогового сигналу від датчика температури з діапазоном *0-150°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_PARA* 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_PARA	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

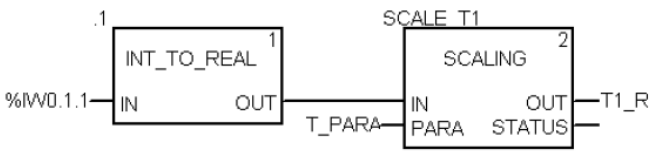


Рисунок 9.2– Приклад використання функціонального блока SCALING для масштабування аналогового вхідного сигналу: зверху – опис структурної змінної T_PARA типу Para_SCALING, знизу – приклад програми на FBD.

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*)*

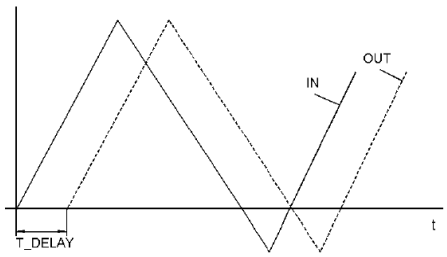


Рисунок 9.3– Діаграма роботи DTIME.

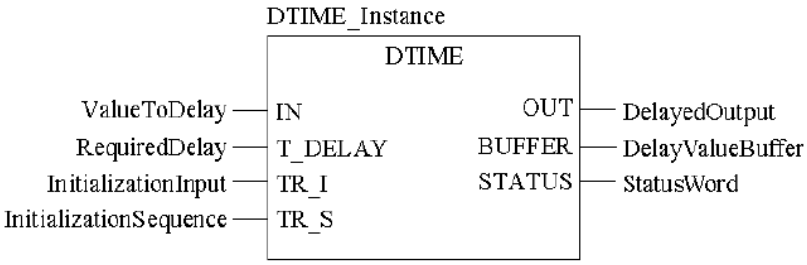


Рисунок 9.4– Приклад виклику блоку DTIME.

Таблиця 9.1 Параметри блоку DTIME

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

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

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.

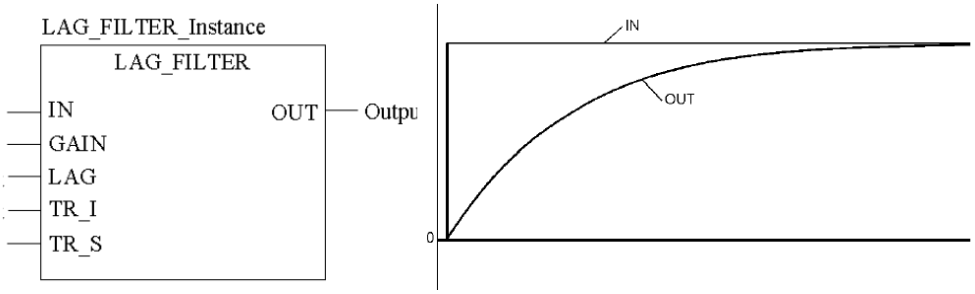


Рис.9.4. Приклад виклику блоку *LAG_FILTER* та діаграма його роботи.

T *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 95ºС (діапазон датчику 0-150ºС) 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 реалізовує 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.

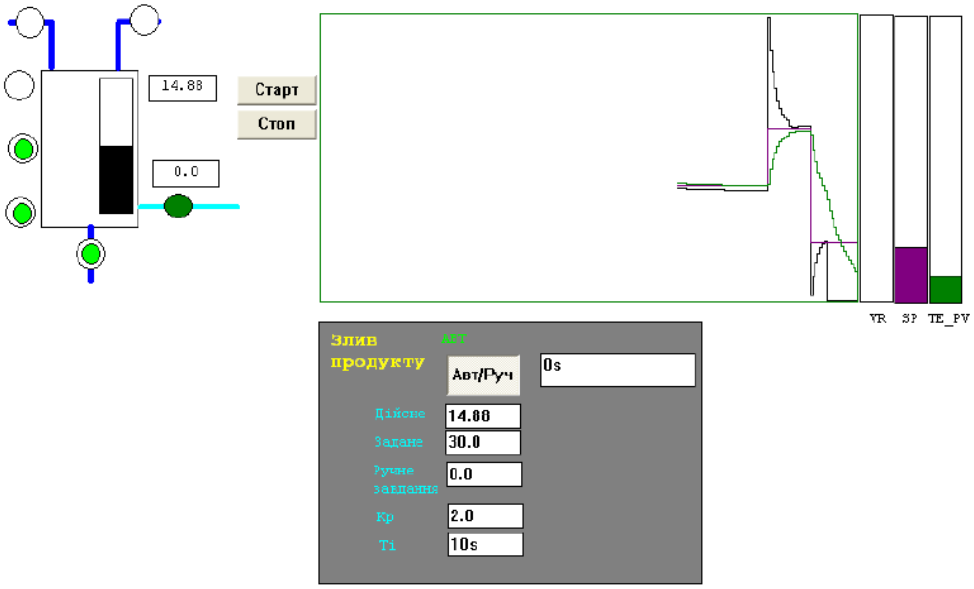


Рисунок 10.1– Приклад операторського екрану до поставленої задачі.

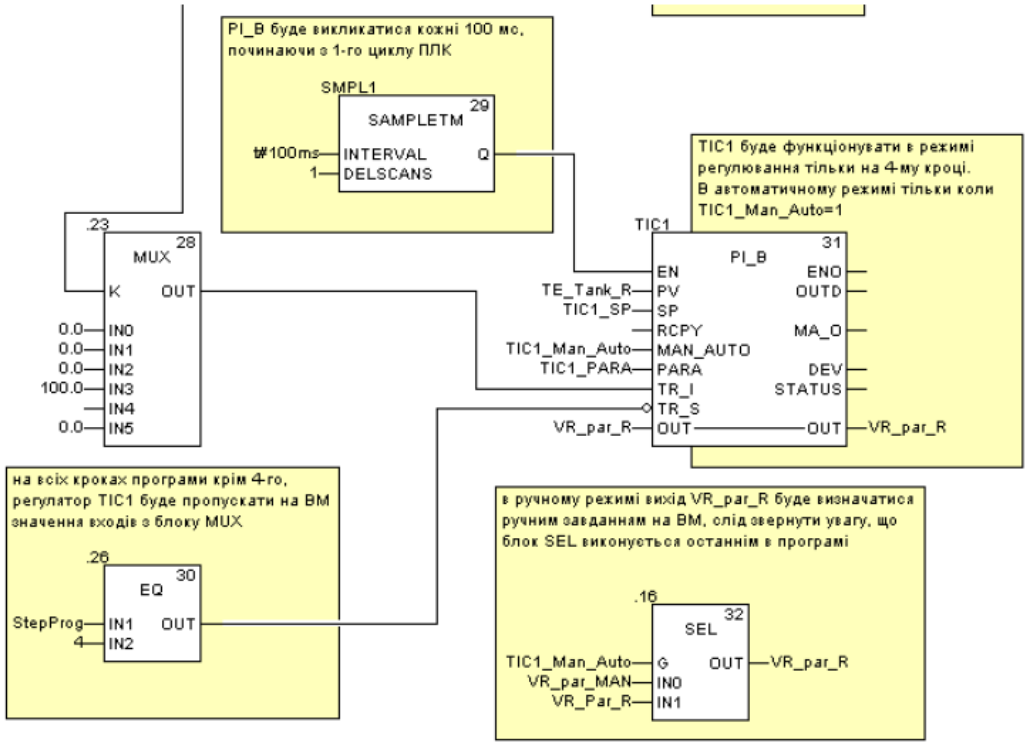


Рисунок 10.2– Модифікована частина секції програми

Variables | DDT Types | Function Blocks | DFB Types

Filter

Name =

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 процеса
SB_stop	BOOL			Кнопка зупинки процеса
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 велчини
pv_sup	REAL		150.0	обмеження по максимуму вхідної велчини
out_inf	REAL		0.0	обмеження по мінімуму вихідної велчини
out_sup	REAL		100.0	обмеження по максимуму вихідної велчини
rev_dir	BOOL		false	режим прямої роботи ПІ-регулятору
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 продукта
VA_nabor2	EBOOL	%Q0.2.17		Клапан набору 2-го продукта
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 значеня ВМ 0-100%

Рисунок 10.3– Перелік змінних.

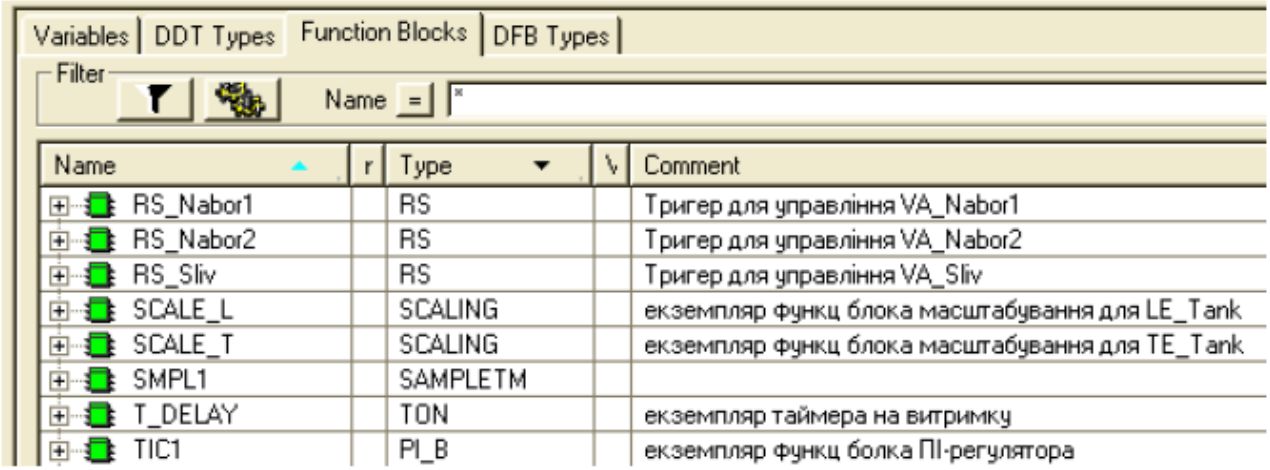


Рисунок 10.4– Перелік екземплярів функціональних блоків.

10.2. Каскадне регулювання температури

Завдання.

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

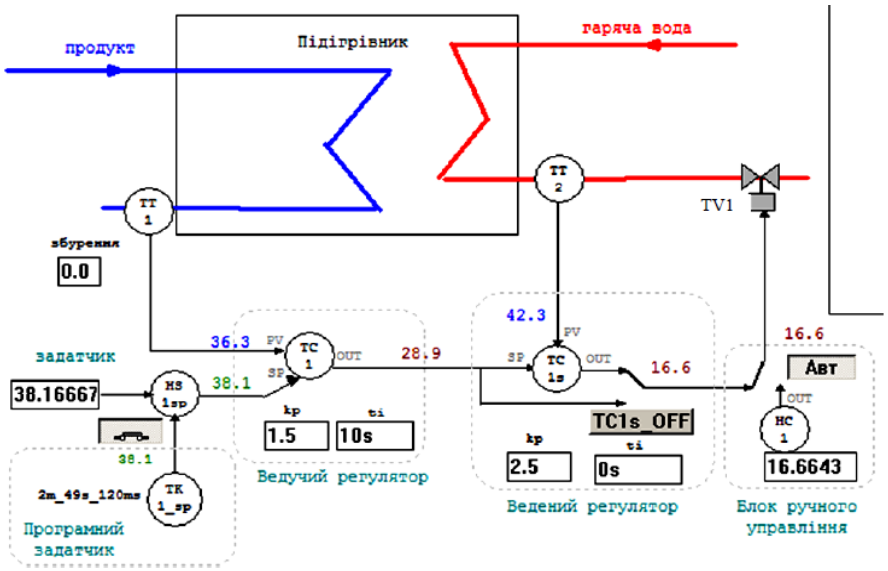


Рисунок 10.5– Операторський екран для контролю та управління процесом нагрівання

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 значені, з використанням каскадного регулятору, де:

a. ведучий регулятор (ПІ режим) стабілізує температуру *TT1*;

b. ведений регулятор (П 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;

6) можливість виключення веденого регулятору з каскаду (з забезпеченням безударності переходу), тобто переводу контуру в режим прямого ПІ-регулювання по температурі *TT1*;

7) при ручному управлінні виконавчим механізмом ведучий регулятор повинен фіксувати своє вихідне значення в останньому положенні.

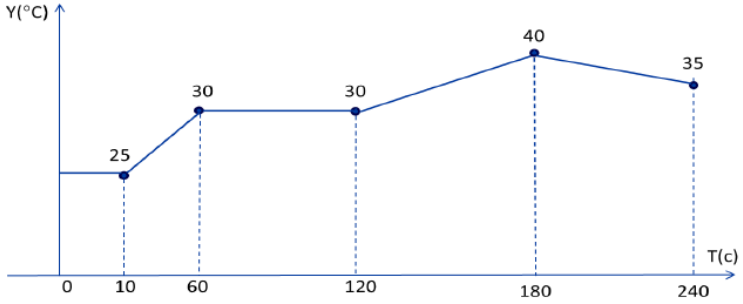


Рисунок 10.6– Залежність завдання температури від часу

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 регулятору *TC1* (ПІ 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 регулятору налаштовуються *Kp* та *Ti*;

- веденого регулятору *TC1s* (П 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

TV1 на вихід ведучого регулятору *TC1.OUT*, тим самим відключивши ведений регулятор з контуру управління; для регулятору налаштовується *Kp* а також *Ti*, ненульове значення якого дозволяє перевести регулятор в ПІ-режим;

- блок ручного управління *HC1*, який дає можливість переключитися в режим ручного управління виконавчим механізмом *TV1* з операторського екрану; кнопка "АВТ" для переключення режиму (нажата - автоматичний режим);

- канал управління (*TV1*);

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

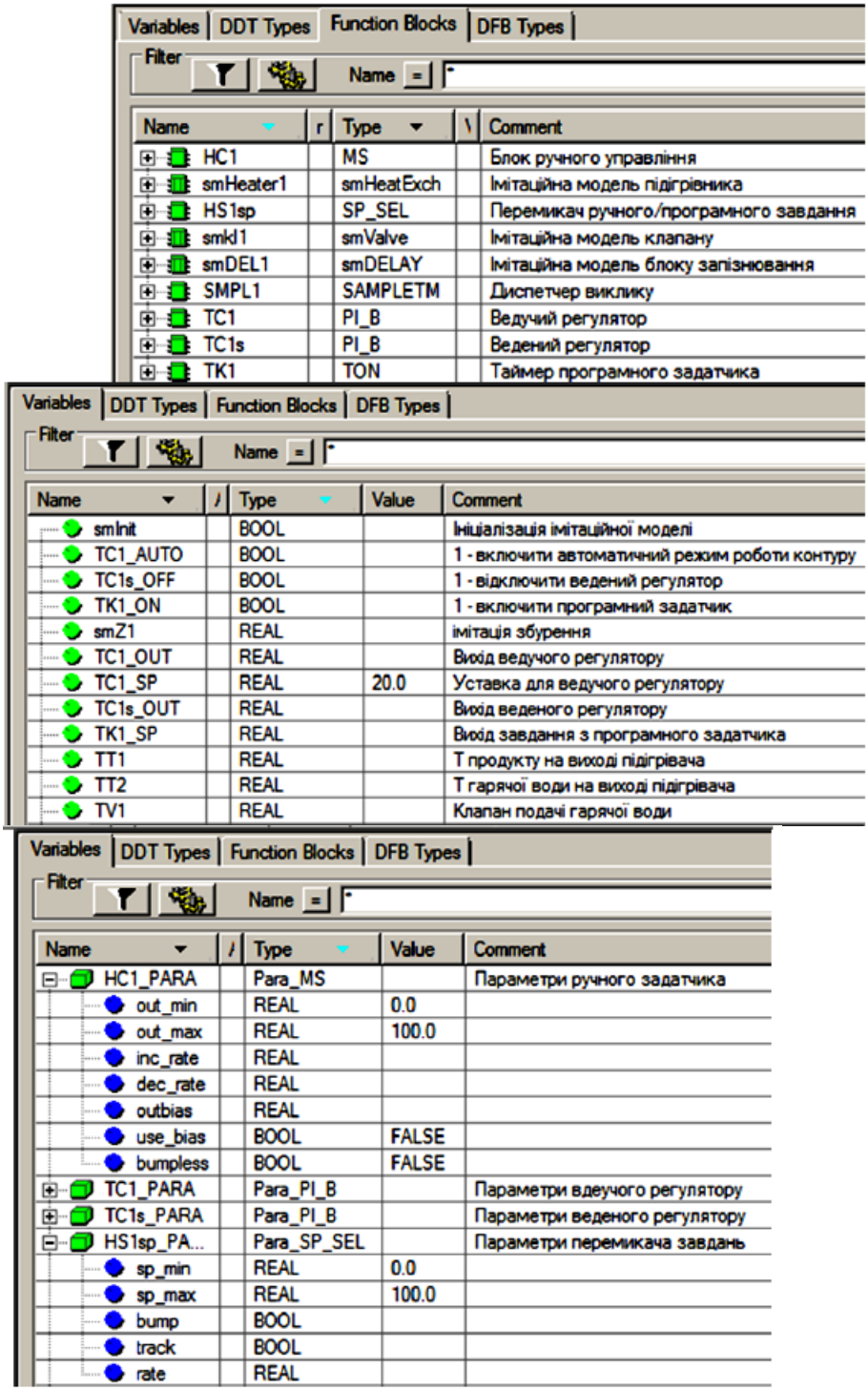


Рисунок 10.7– Змінні проекту

Загальні принципи роботи програми. Для реалізації даної задачі використовуються 4-ри секції (рис.10.8): секція "*INPUTS*" – для обробки вхідних 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 "*OTPUTS*"аналогічним чином проводиться масштабування для значення виконавчого механізму *TV1*, тільки в зворотному напрямку (діапазон 0-100%ХРО в 0-10000).

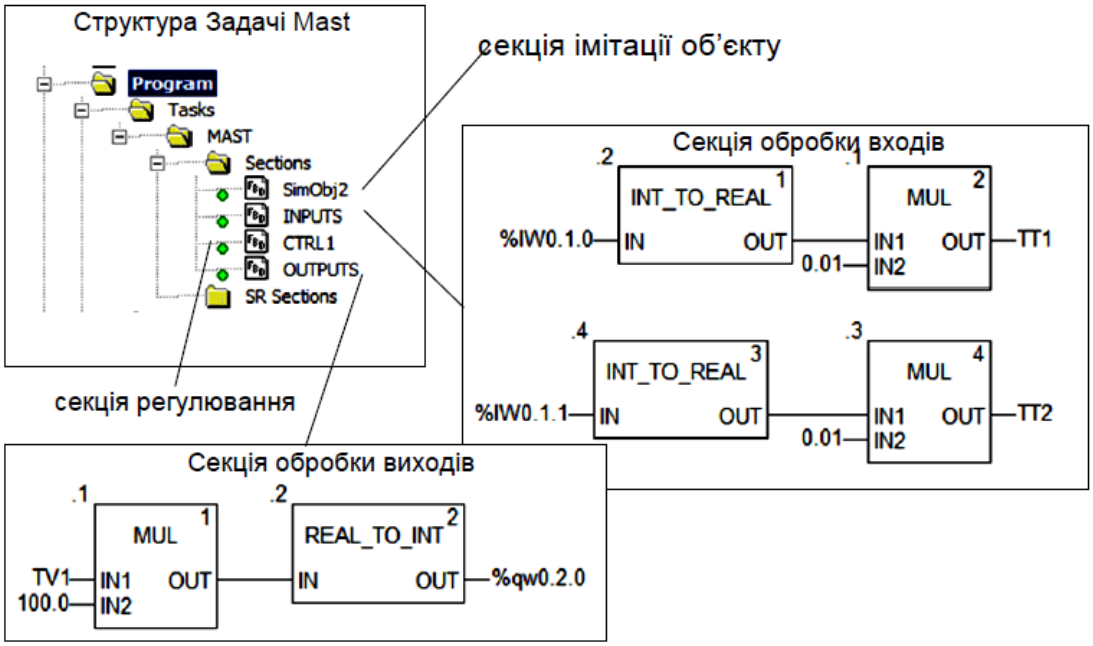


Рисунок 10.8– Структура задачі MAST

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 *XiYi*, 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 регулятору) *TK1_SP*.

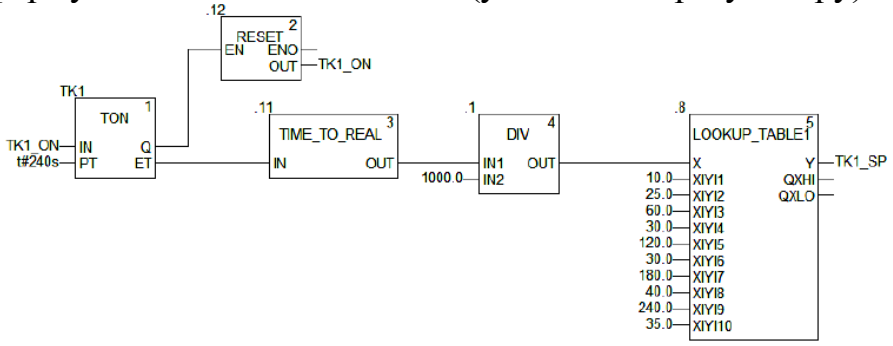


Рисунок 10.9– Реалізація програмного за датчика

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*).

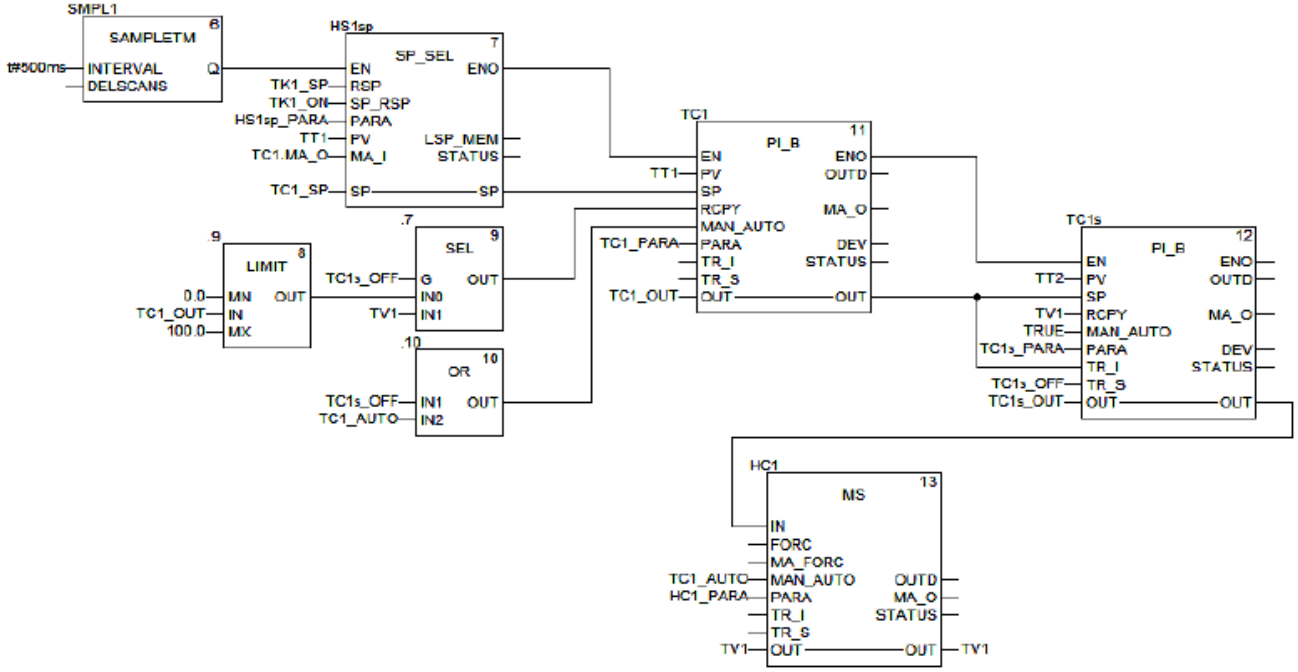


Рисунок 10.10– Фрагмент секції з реалізацією контуру регулювання (для UNITY PRO ≥6.0)

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 оператору, 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 перемикання 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* переключається 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 регулятору. 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 регулятору підключається до *TC1s.SP* через змінну *TC1_OUT*, тим самим формуючи завдання веденому регулятору. Виведення з каскаду веденого регулятору 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 регулятору *TC1s* на вхід *RCPY* завжди подається значення, яке йде на виконавчий механізм *TV1*. Тобто, якщо ведений регулятор виведений з каскаду, він все одно буде формувати нове значення виходу внутрішнього регулятору на базі плинного *TV1*, що забезпечить безударний перехід при повторному вводі його в каскад. Для ведучого регулятору значення *RCPY* залежить від того, чи включений в контур управління ведений регулятор *TC1s*. Якщо ведений регулятор виведений із каскаду, то *RCPY=TV1*, так як *TC1* безпосередньо управляє виконавчим механізмом, отже нове розраховане значення базується на значенні *TV1*. 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 *HC1* та відповідних режимів роботи регуляторів *TC1* та *TC1s*. Значення виходу *HC1.OUT* зв’язане зі змінною *TV1*. У автоматичному режимі роботи контуру (*TC1_AUTO=TRUE*) вихід *HC1.OUT=TC1s_OUT*. У ручному режимі вихід *HC1.OUT* може бути змінений оператором. При цьому, враховуючи що значення *TV1* повторюється на входах *RCPY* блоків регулювання забезпечується безударний перехід.

Ведений регулятор завжди знаходиться в автоматичному режимі, так як для ручного режиму передбачений блок *HC1*. 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*. Налагодження проекту зробити з використанням готового програмного імітатору 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 показчик 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 *%I0.3.0*, "повністю закритий" - вхід ПЛК *%I0.3.1*. 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 значені, 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.

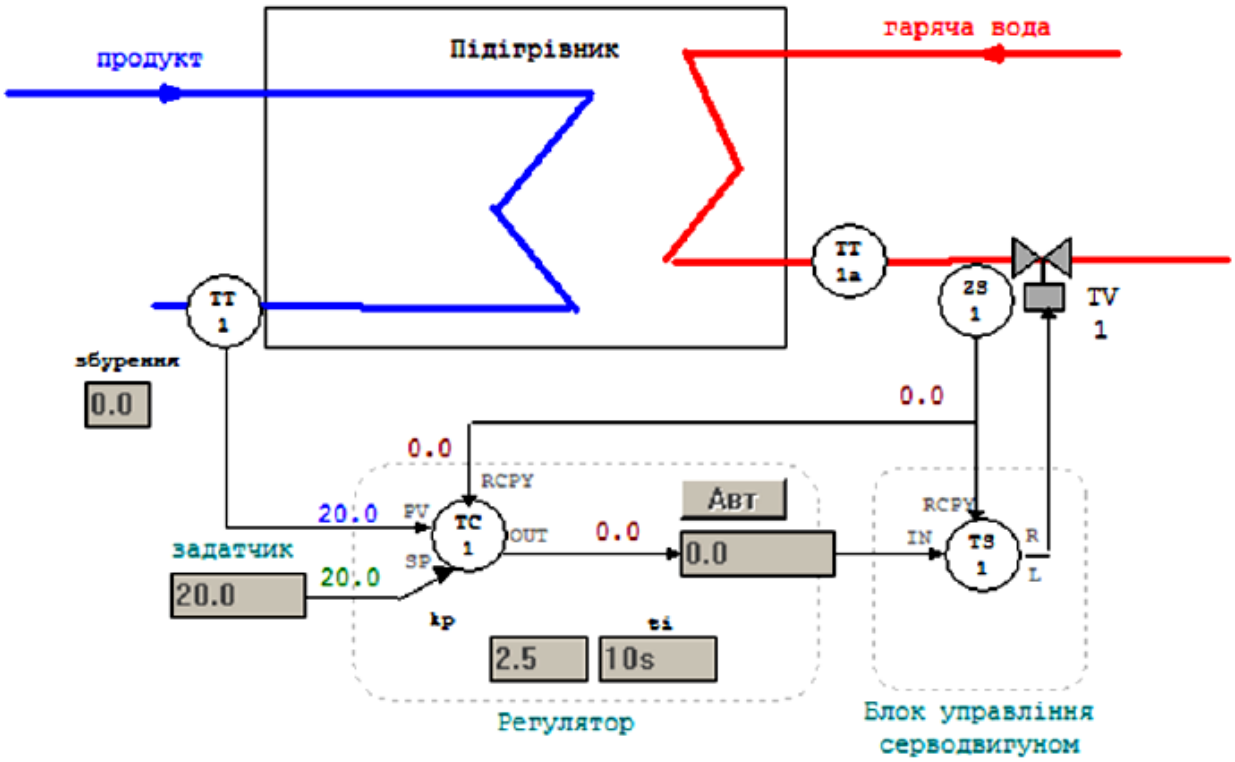


Рисунок 10.11– Операторський екран для контролю та управління процесом нагрівання в підігрівнику 1

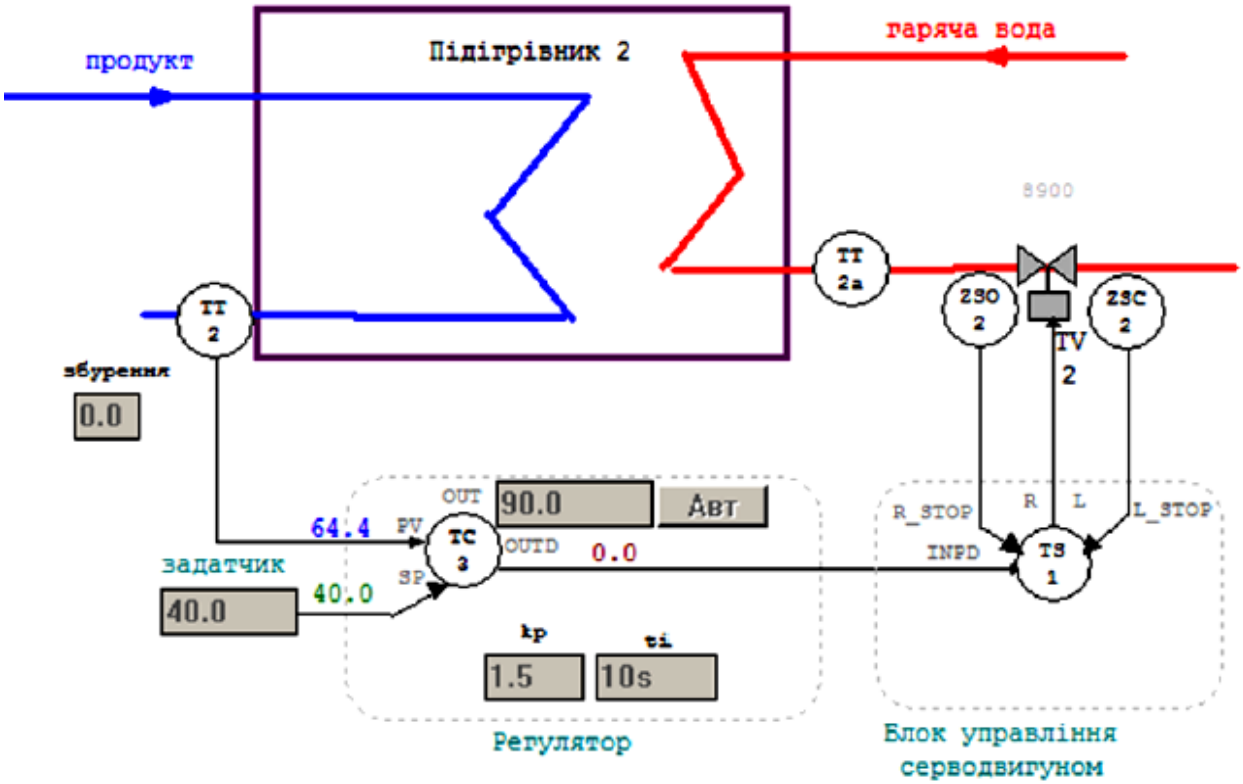


Рисунок 10.12– Операторський екран для контролю та управління процесом нагрівання в підігрівнику 2

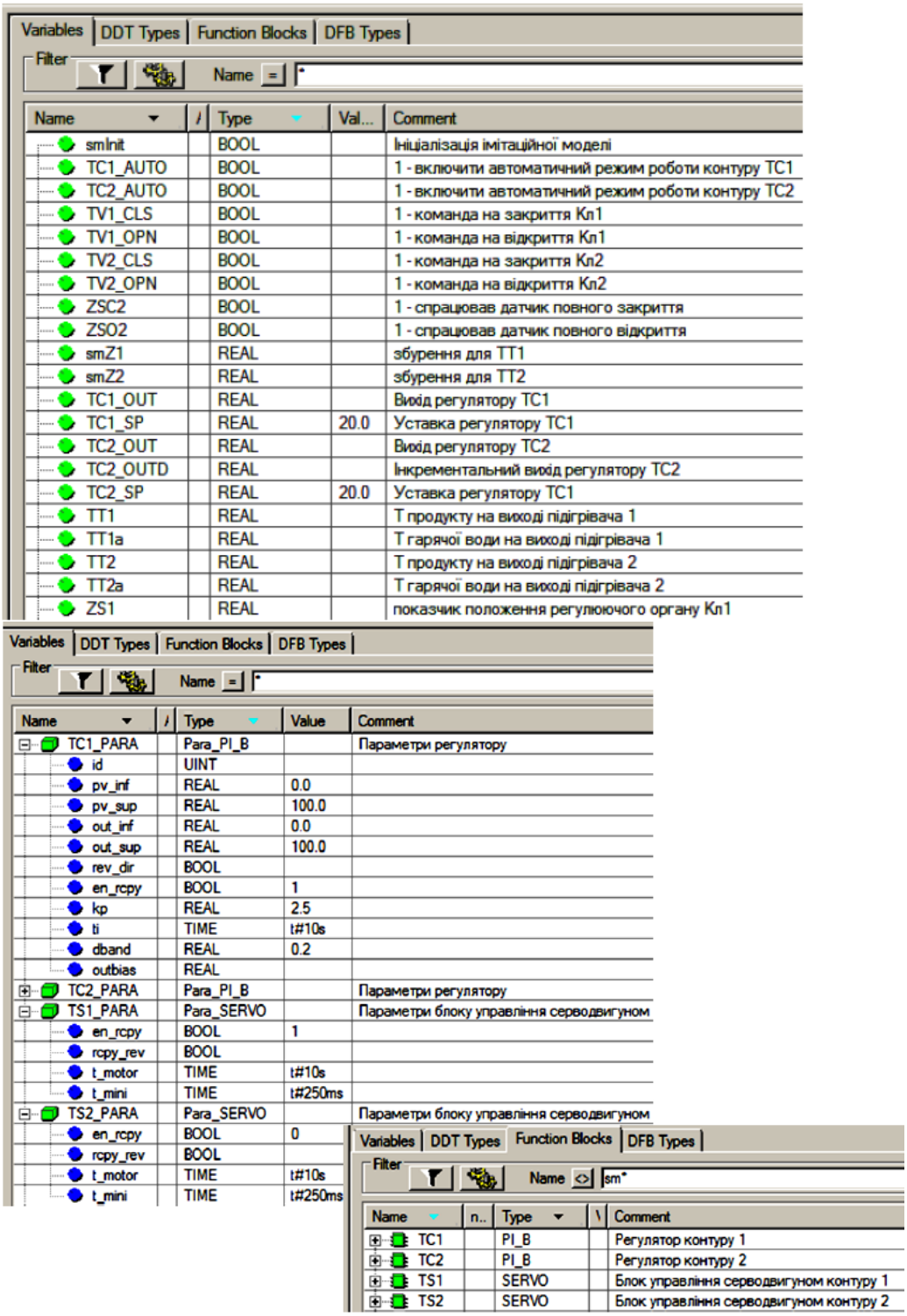


Рисунок 10.13– Змінні проекту

Загальні принципи роботи контуру управління температурою 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* регулятору заводиться сигнал зворотного зв'язку по положенню регулюючого органу *ZS1*; для регулятору 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 правління; секція "*CTRL1*" та "*CTRL2*" – для реалізації контурів управління температурою відповідно в підігрівнику 1 та підігрівнику 2. Секція "*Simulation*" призначена тільки для імітації об'єкта.

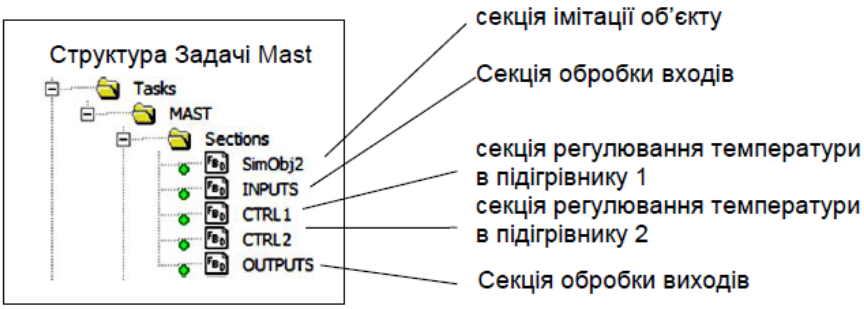


Рисунок 10.14– Структура Задачі MAST

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 %ХРО). До входів контуру з підігрівачем 2 належать також датчики кінцевого положення типу "відкрито" - *ZSO2*, та "закрито" - *ZSC2*.

У секції "*OTPUTS*" (рис.10.16) для кожного виконавчого механізму на виходи ПЛК подаються сигнали "більше" (*TV1_OPN*, *TV2_OPN*) та "менше" (*TV1_CLS*, *TV2_CLS*).

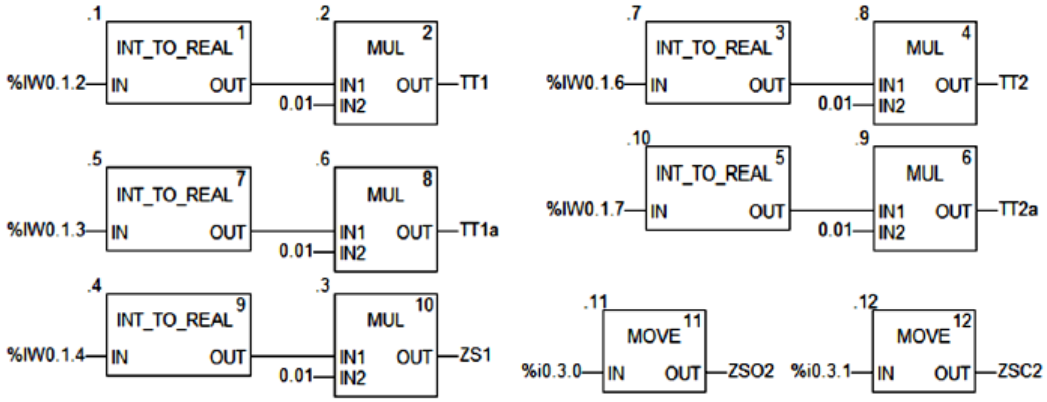


Рисунок 10.15– Секція обробки входів

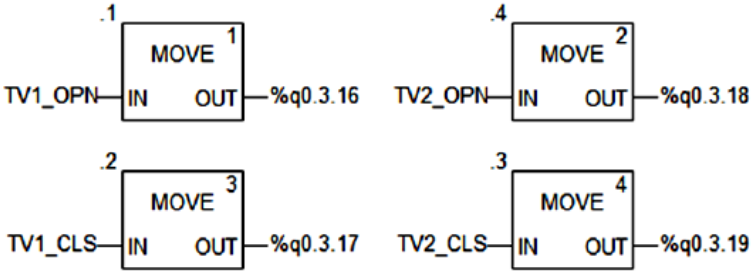


Рисунок 10.16– Секція обробки виходів

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*.

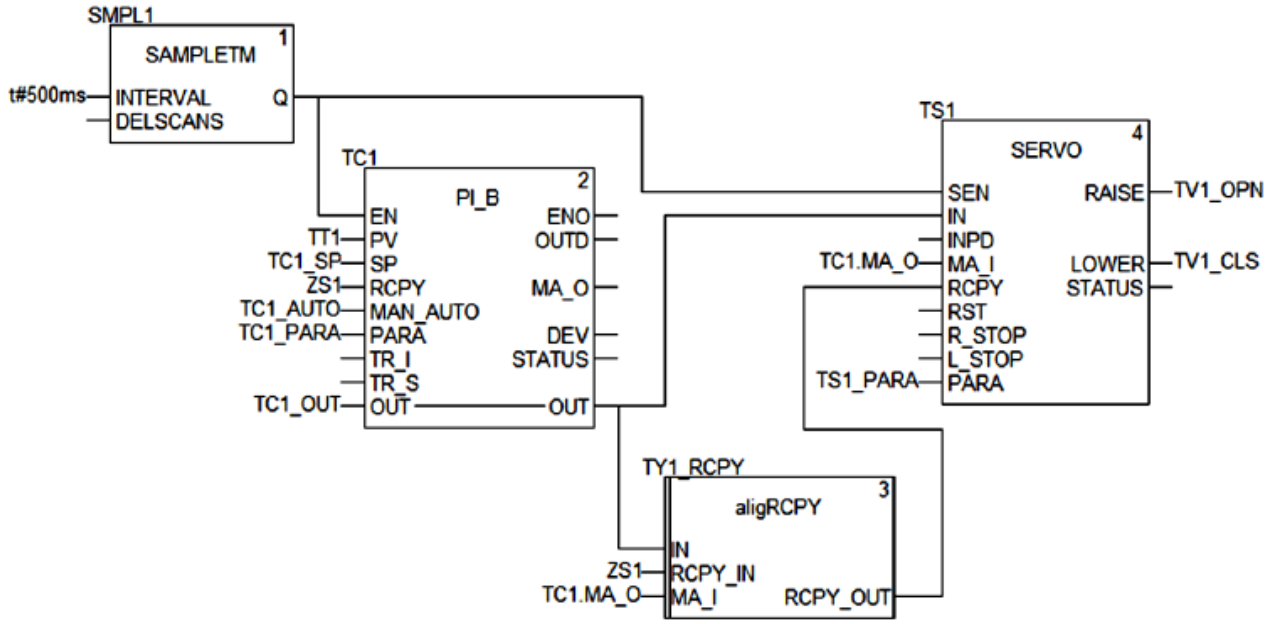


Рисунок 10.17– Секція CTRL1

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 покажчика 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.

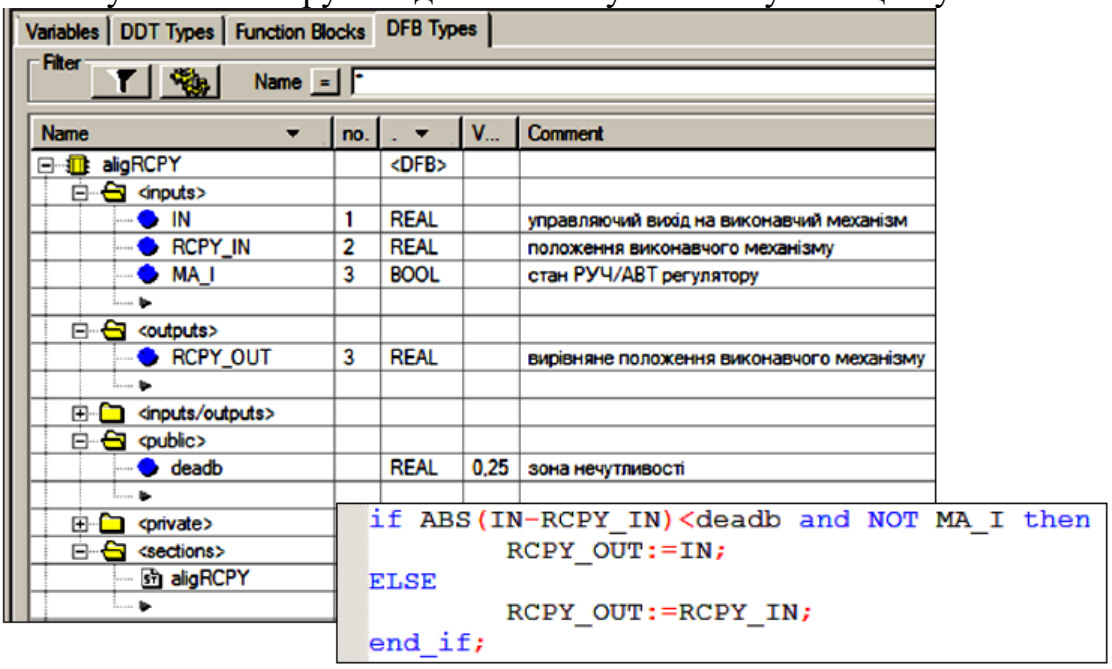


Рисунок 10.18– Структура і програма DFB типу aligRCPY.

Блок вирівнювання *TY1_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.

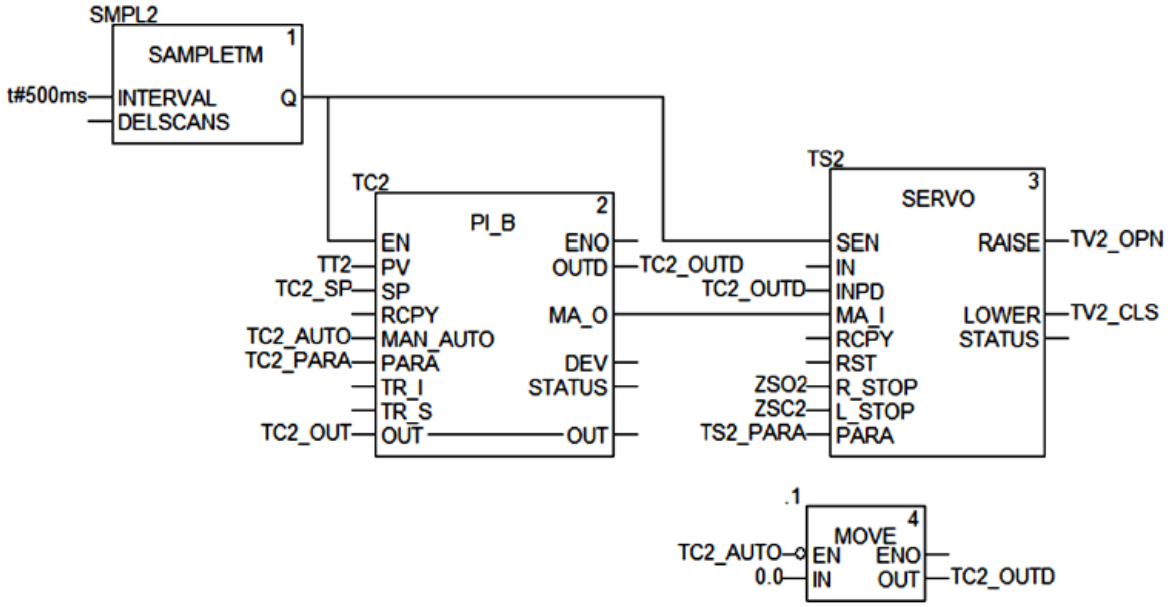


Рисунок 10.19– Секція *CTRL2*

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 циклу. 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.

11.1Технологія 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 сталевою 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.

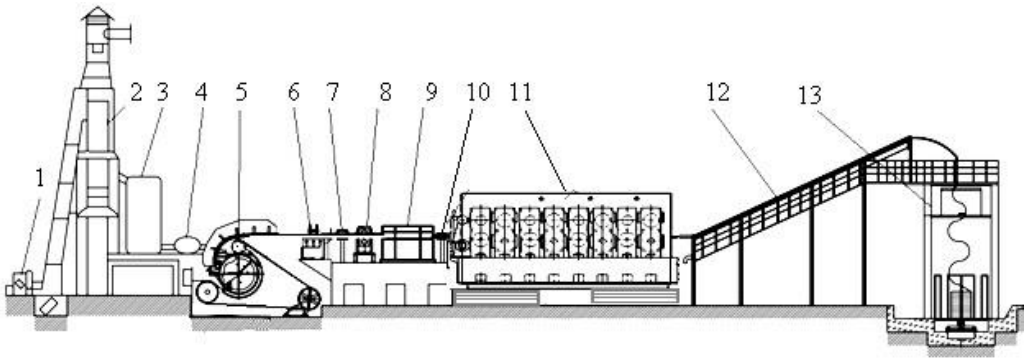


Рисунок 11.1 - Технологічна лінія неперервного лиття та прокатки

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:

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

де K – коефіцієнт передачі, вихідна величина / % ходу регулювального органа; T – стала часу, с; τ – 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

Параметри / Коефіцієнти	Температура рідкого алюмінію у шахтній печі	Температура рідкого алюмінію у роздавальній печі	Температура заготовки після ливарного колеса	Температура катанки на виході з прокатного стана	Температура катанки після гартування
K	2,7	15	1	2	1.5
T	32	25	15	15	15
τ	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:

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

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:

$$G_{кр} = \frac{Q_{вод}}{c_в \cdot \Delta T_в}, \qquad (11.3)$$

де $G_{кр}$ – витрата води, яка подається в кристалізатор, кг/с; $c_в$ –теплоємність води, Дж/(кг·К); $\Delta T_в$ – різниця між температурою води, що подається *Твх* у кристалізатор та відводиться *Твих* 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 багатошарової стінки в цих зонах.

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

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

де $K_{сум} = \sum_{1}^{n} K_i$ – сумарний коефіцієнт теплопередачі багатошарової стінки кристалізатора за всіма зонами охолодження.

Якщо коефіцієнт теплопередачі *К1* більше *Кср* на *Δ1*%, тоді реальна витрата води у першу зону охолодження має бути менше за уявну на *Δ1*%:

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

Якщо коефіцієнт теплопередачі *К2* менше *Кср* на *Δ2*%, тоді реальна витрата води у другу зону має бути більше за уявну на *Δ2*%:

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

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 багатошарової стінки за напрямами охолодження у кожній з них і воду, що подається у кристалізатор, розподілити поміж визначеними зонами охолодження, відповідно до рівняння:

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

де *Gi*– витрата води на охолодження в *i*-ту зону кристалізатора; *n* – кількість зон охолодження; *Ki* – коефіцієнт теплопередачі багатошарової стінки кристалізатора за напрямом охолодження в *i*-тій зоні; *Кср* – середній коефіцієнт теплопередачі багатошарової стінки кристалізатора за всіма зонами охолодження.

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

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

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

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

Для забезпечення рівномірного твердіння зливка при *Кср* за всіма зонами охолодження, відсоток витрати води у кожній зоні складе:

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

Оскільки *К1*=776Вт/(м$^{2\cdot}$°К) більше *Кср*=751Вт/(м$^{2\cdot}$°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 *К2*=817Вт/(м$^{2\cdot}$°К) більше *Кср*=751Вт/(м$^{2\cdot}$°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 *К3*=660Вт/(м$^{2\cdot}$°К) менше *Кср*=751Вт/(м$^{2\cdot}$°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 нестаціонарної 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 напрямами 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 $Q_{заг}$:

$$Q_{заг} = \rho \frac{V}{60} \cdot F \cdot h_{роз} - G_{роз} \cdot \left(h_{роз} - h_{заг}\right), \qquad (11.6)$$

де $G_{ðîç} \cdot \left(h_{ðîç} - h_{çàã}\right)$ – теплота, що віддається гарячим теплоносієм, Вт; $G_{ðîç}$ – витрата розплаву, що визначається продуктивністю агрегату та швидкістю лиття, кг/с; $h_{ðîç} = L + c_{ï} \cdot T_{ðîç}$ – ентальпія сплаву на початку виливниці, Дж/кг; $h_{çàã} = L + c_{ê} \cdot T_{çàã}$ – ентальпія сплаву у кінці виливниці, Дж/кг; $T_{роз}$ – початкова та $T_{заг}$ – кінцева температура алюмінієвого сплаву у виливниці, °С; $c_{п}$, $c_{к}$ – 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 $Q_{вод}$:

$$Q_{вод} = c_{в} \cdot G_{в} \cdot \Delta T_{в} \qquad (11.7)$$

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

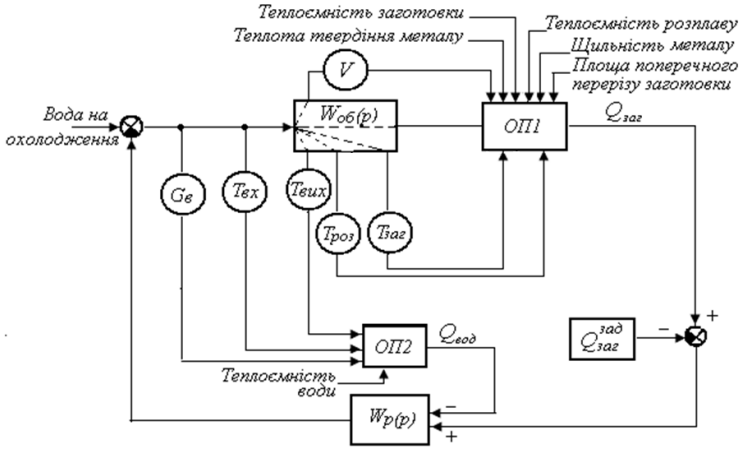


Рисунок 11.2 – Структурна схема системи автоматичного регулювання теплового режиму кристалізатора

На підставі інформації з датчиків швидкості лиття *V*, температури розплаву $T_{роз}$ та температури затверділої заготовки $T_{заг}$ з урахуванням відомих значень решти параметрів, обчислювальний пристрій *ОП*1 за формулою (11.6) обчислює поточне значення теплоти, що виводиться заготовкою з кристалізатора $Q_{заг}$, яку порівнюють з розрахованим заздалегідь заданим значенням $Q_{заг}^{зад}$ 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 $Q_{заг}$ і $Q_{заг}^{зад}$ регулятор змінює подавання води на охолодження зливка до повної компенсації розбалансу на його вході.

Під час змінювання вхідних параметрів $G_{в}$, $T_{вх}$ 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 *ОП*2 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:

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

де L- загальна довжина алюмінієвої заготовки, м; $L_{заг} = V_Л \cdot \Delta t$ – базова довжина отримуваного 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; λ - коефіцієнт витяжки прокатного стана; Δt – 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 λ.

Вираз у дужках цього рівняння визначається нестабільністю роботи ливарного колеса й прокатного стана, а тому характеризує величину прирощення заготовки ΔL 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 створені 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:

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

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

$$T_i = \frac{2\pi}{\varpi_i}, \qquad (11.10)$$

де ϖ_i – кутова швидкість обертання проводки, рад/с; *V* – швидкість подачі катанки, м/с; *Rmax*– радіус максимального витка, м; *Rmin* – радіус мінімального витка, м; *d*– діаметр катанки, м; $1 \le i \le n, n = \frac{R_{\max} - R_{\min}}{d \cdot \alpha}$ – кількість витків у шарі, α – крок укладання витків, $1 \le \alpha \le \frac{R_{\max} - R_{\min}}{2d}$, «*Rmax* –» – при укладанні шару від периферії до центру, «*Rmin*+» – 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 настройок у кожному з них. В Unity Pro за допомогою мов FBD і ST реалізовані програми моделювання та управління роботою шахтної та роздавальними печами печі, ливарного колеса, кристалізатора і прокатного стана, моталки і гартівного пристрою, а також програма управління станом об'єктів візуалізації. В пакеті TIA Portal створено чотири екрана візуалізації: стартовий, шахтної та роздавальних печей, ливарної машини і прокатного стана, гартівного пристрою та моталки. Як додаток до них розроблено дев'ять екранів відображення трендів основних технологічних параметрів і установлення настройок 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 – символьні 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 дільниць 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, настроювання 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.

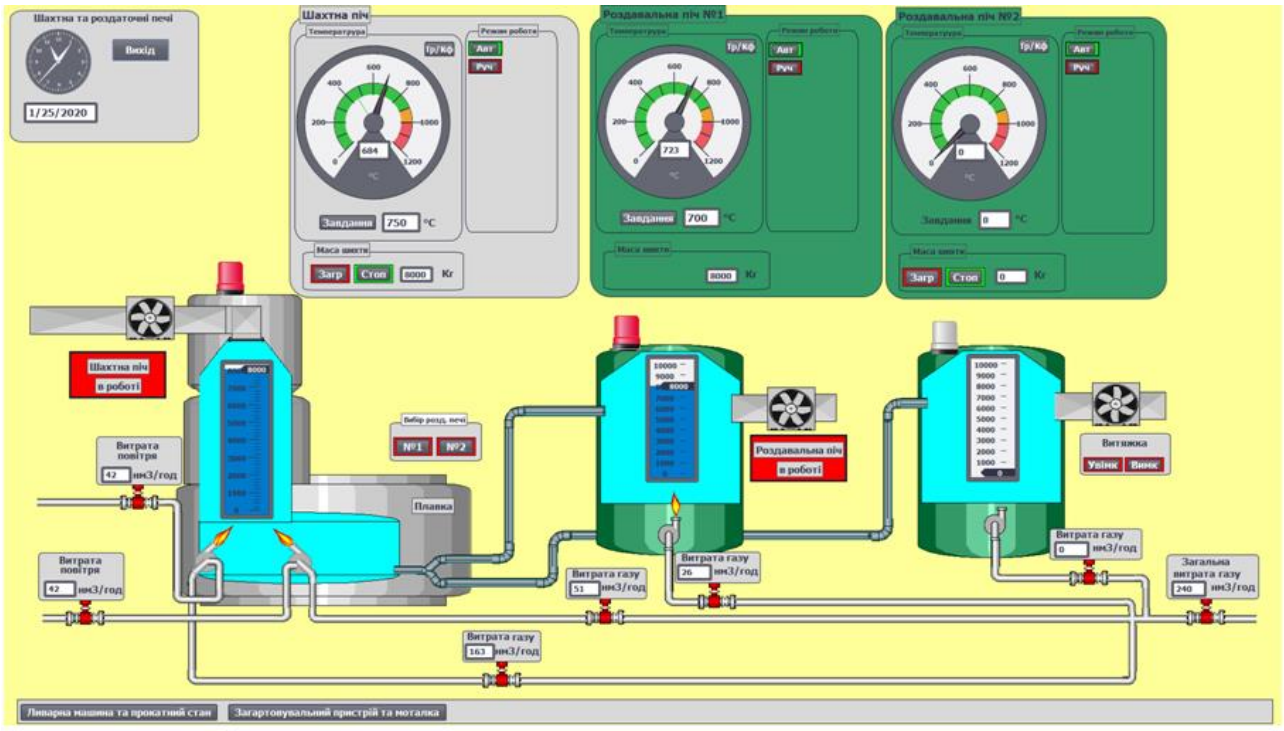


Рисунок 11.4 – Вікно візуалізації пічної дільниці

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.

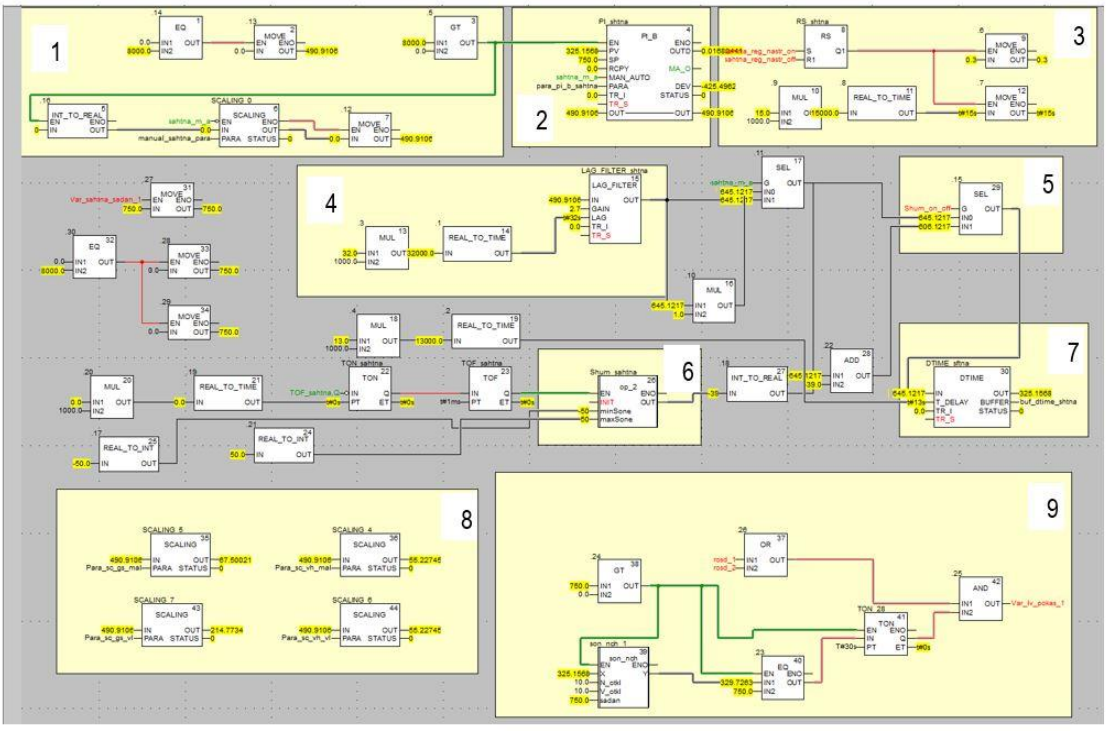
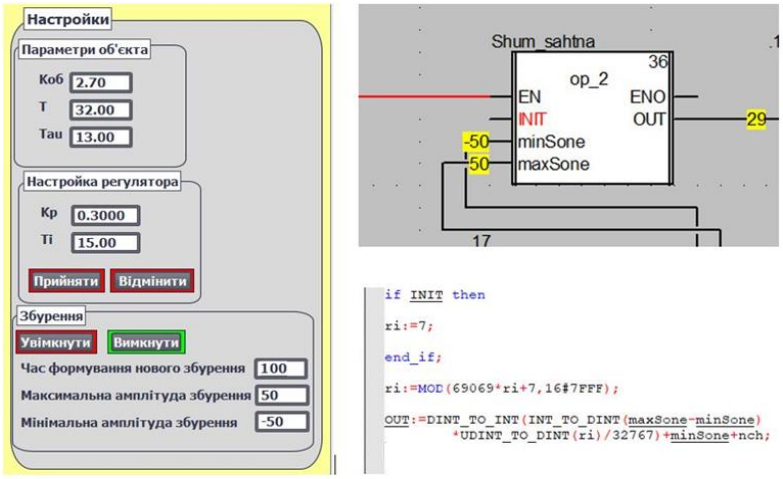


Рисунок 11.5 – Програма моделювання роботи та анімації шахтної печі

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.



а) б)

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 стана, керування їх швидкістю в ручному режимі, відслідковування вторинними приладами значень контрольованих і регульованих параметрів, формування траєкторії руху катанки від кристалізатора до прокатного стана, відрізання гідравличними ножицями бракованої частини заготовки та автоматичне регулювання теплового режиму кристалізатора, довжини заготовки і температури катанки після прокатного стану.

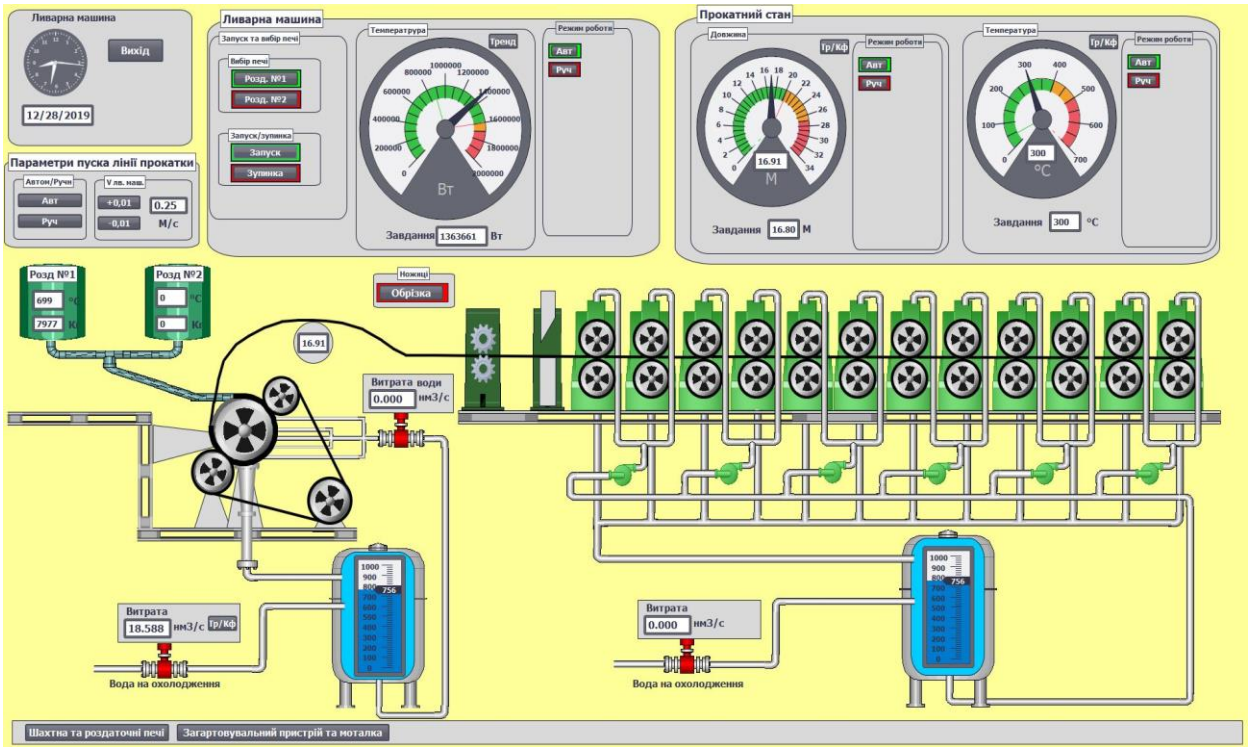


Рисунок 11.7 – Вікно візуалізації ливарної машини та прокатного стану

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.

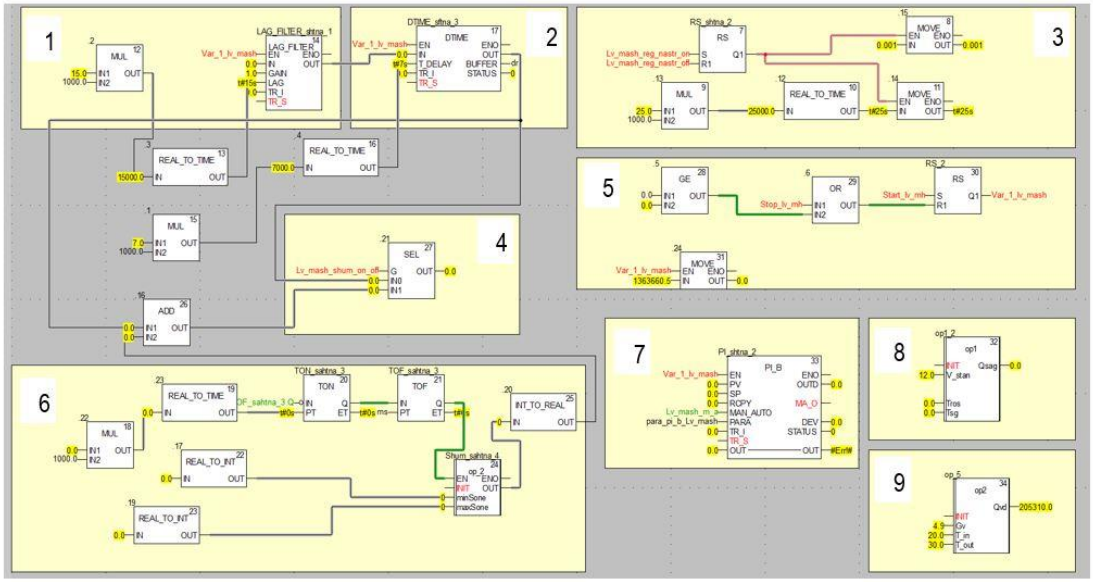
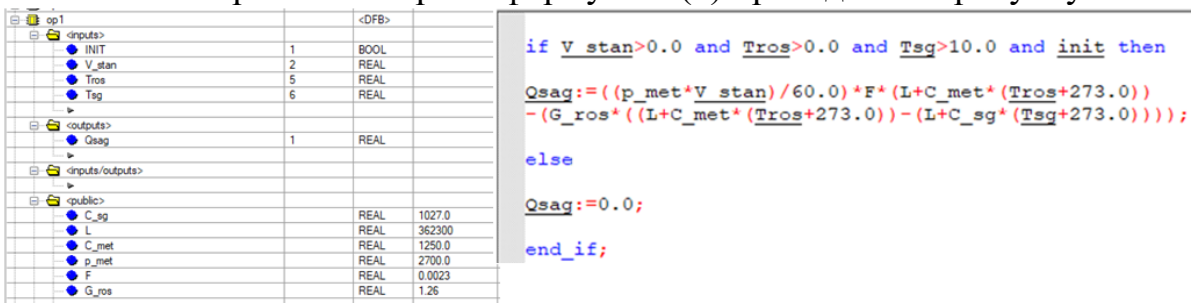


Рисунок 11.8 – Програма регулювання теплового режиму кристалізатора

Програма включає 9 основних фрагментів: моделі кристалізатора 1 і транспортного запізнювання 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 настройок 7.

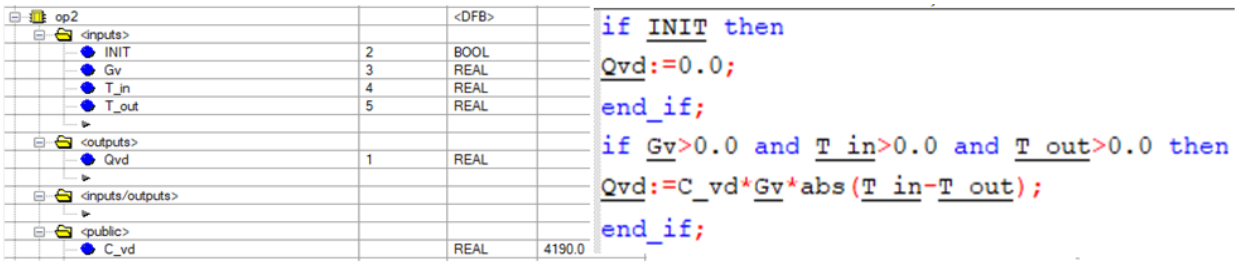
Функціональний блок розрахунку кількості теплоти, що виноситься із заготовкою з кристалізатора за формулою (6) приведено на рисунку 11.9.



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.



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.

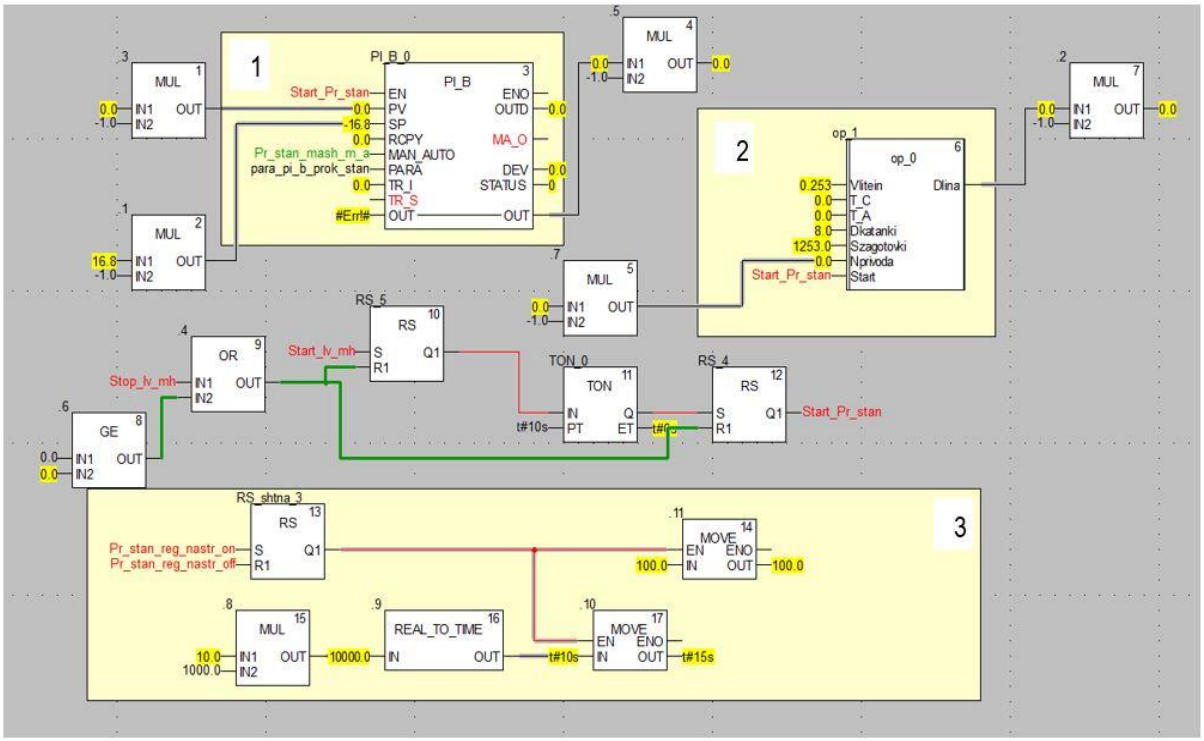
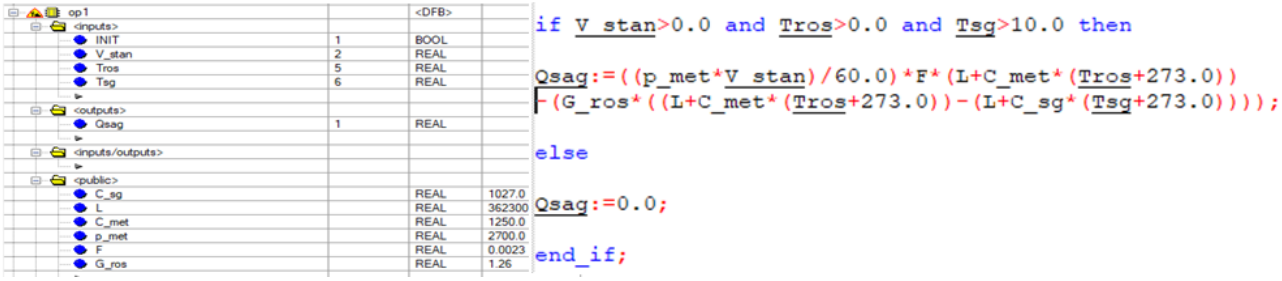


Рисунок 11.12 – Програма імітації формування заготовки та регулювання її довжини

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.



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.

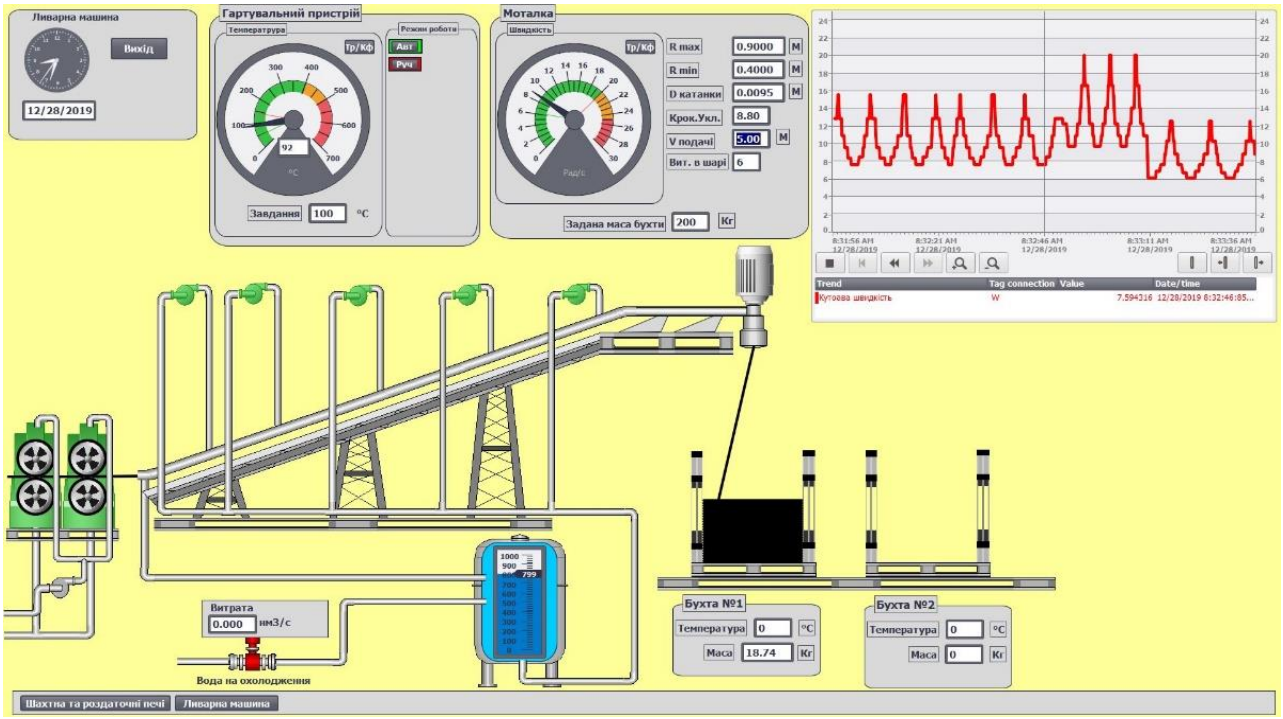


Рисунок 11.14 – Вікно візуалізації гартувального пристрою та моталки

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

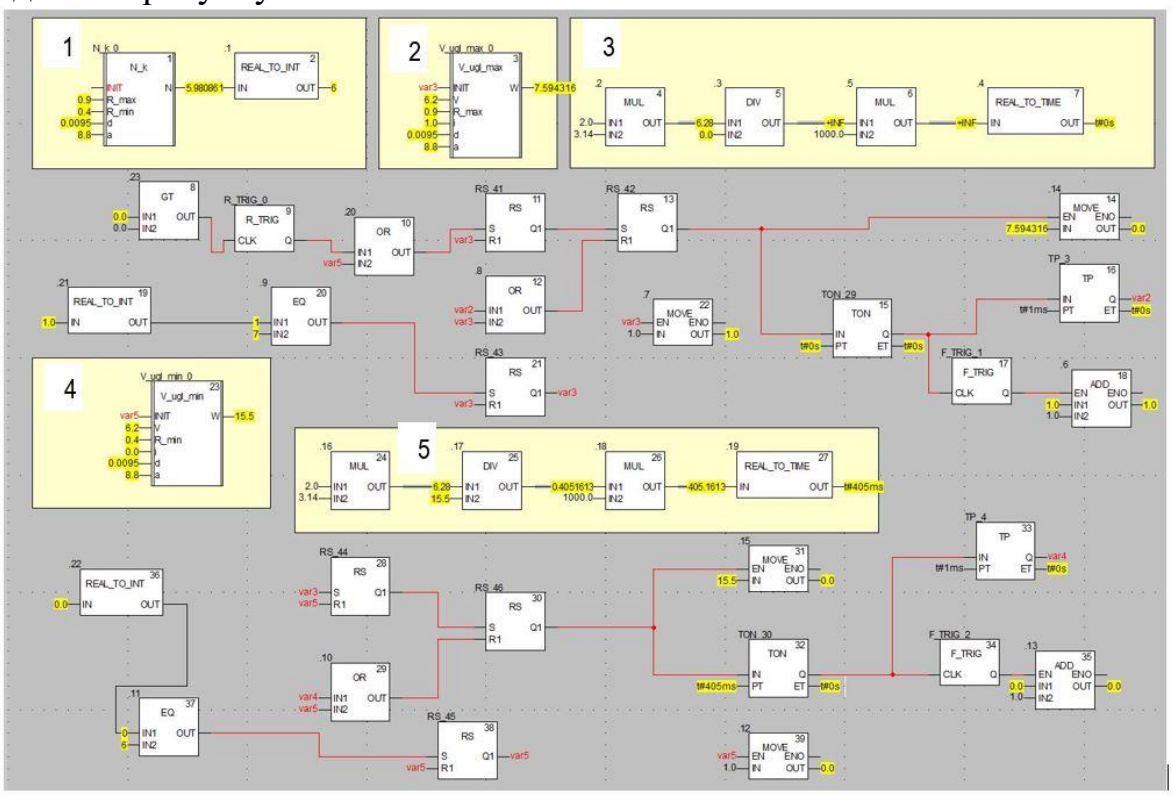
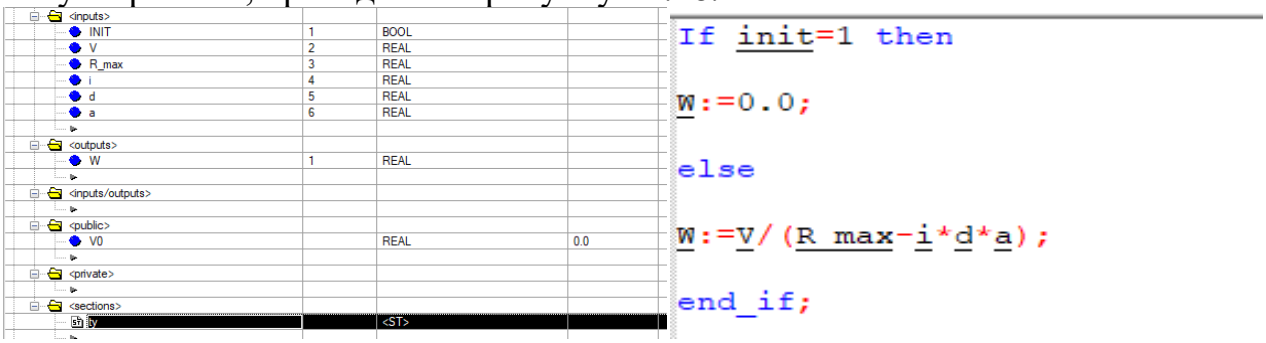


Рисунок 11.15 – Програма керування швидкістю виткоутворювача моталки

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.



а) б)

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 зблокований, а також автоматичний. При реалізації вільного режиму управління, кожна група механізмів управляється самостійно, що дозволяє оператору вручну синхронізувати їх роботу. При зблокованому 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 дільниці та ливарно-прокатного агрегату, дослідити вплив тих чи інших параметрів на хід технологічного процесу, а також відпрацьовувати 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 розплава 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 установлено, 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 підвищені 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 ± 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 металлу 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 регулювання 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.

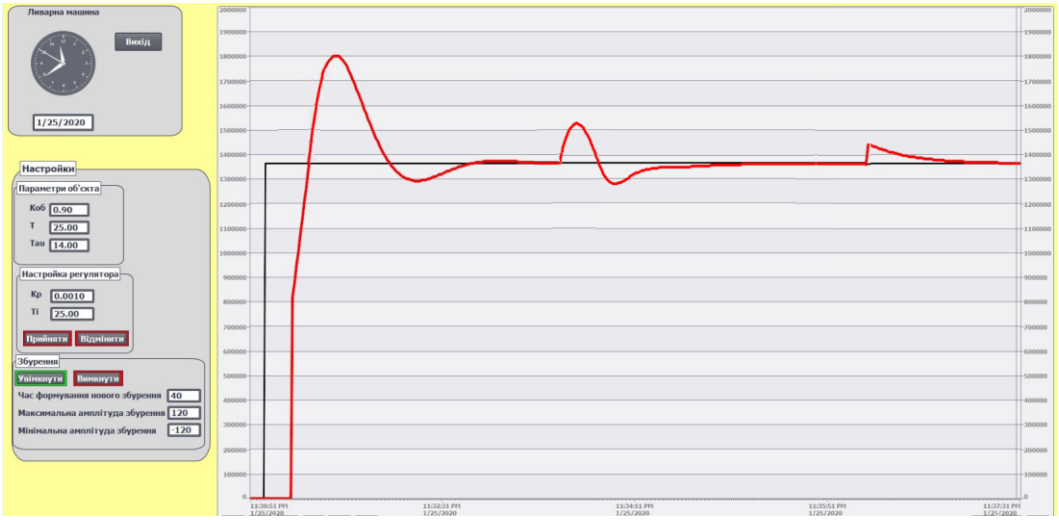


Рисунок 11.17 – Перехідні процеси в САР теплоти, що виноситься заготовкою з ливарного колеса

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 системи автоматичного регулювання (останній фрагмент графіка).

Рисунок 11.18 – Перехідні процеси в САР довжини заготовки

При досліджені 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 И.В. ПЛК Modicon M580. Обзор аппаратных средств и возможностей по построению систем АСУТП. SCHNEIDER ELECTRIC