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*).

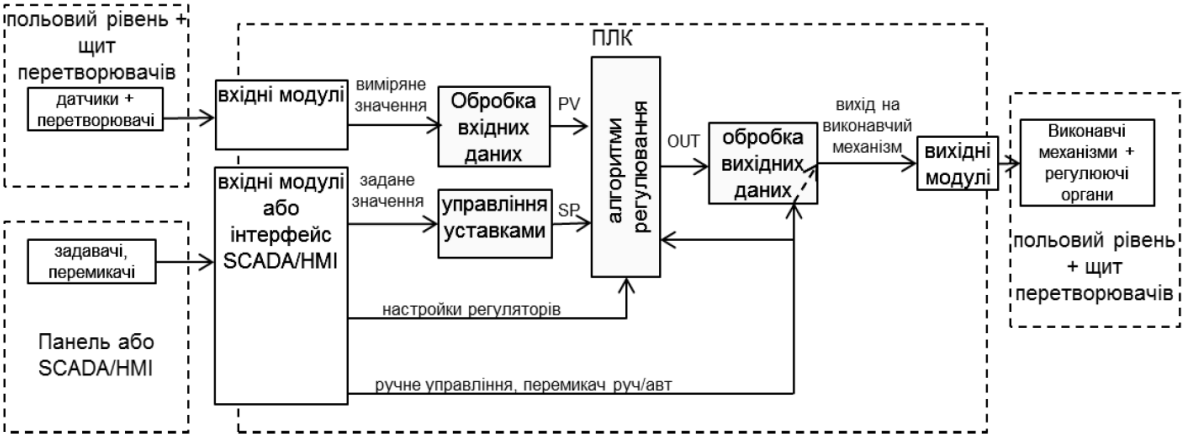

Рисунок 4.1– Структура контурів регулювання

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).

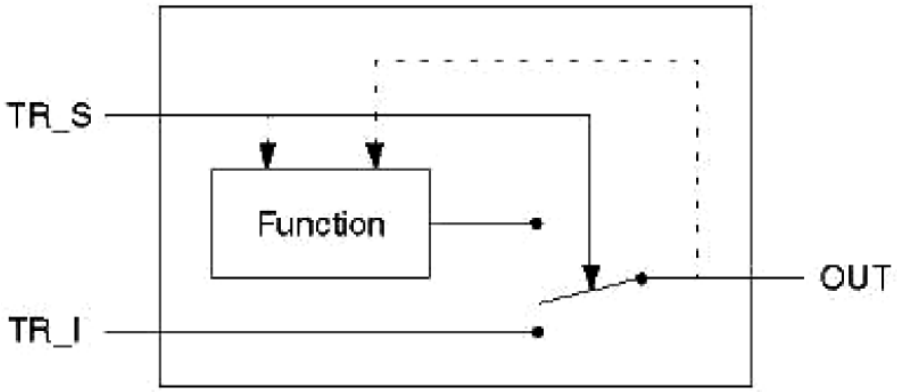

Рисунок 4.2– Режим слідкування

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*).

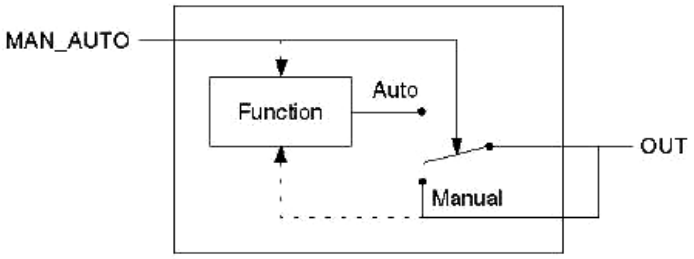

Рисунок 4.3. Режим Руч/Авт

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*.

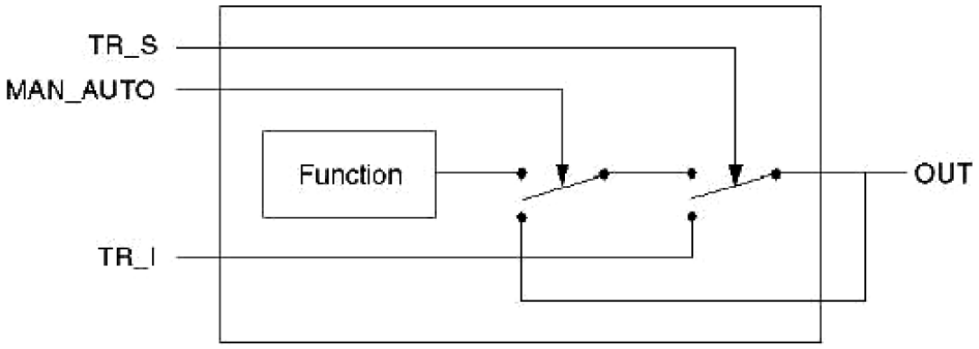

Рисунок 4.4– Пріоритетність режимів.

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

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).

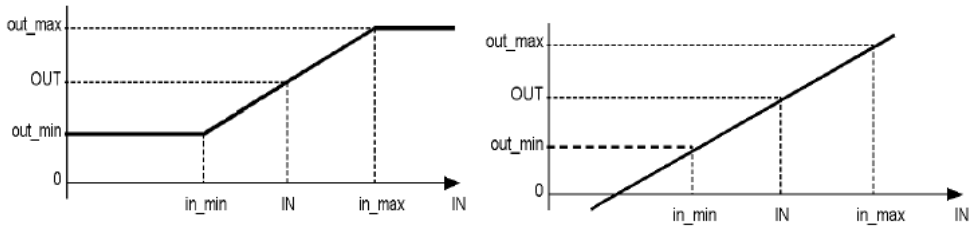

Рис.9.1. Залежність масштабованого вихідного значення (*OUT*) від вхідного (*IN*) при обмеженні на вихідний сигнал (зліва, *Clip*=1) та без обмеження (праворуч, *Clip*=0) для блока *SCALING*.

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

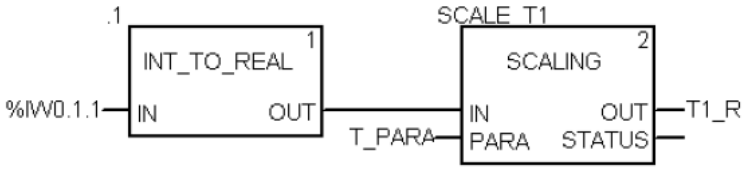

Рисунок 9.2– Приклад використання функціонального блока SCALING для масштабування аналогового вхідного сигналу: зверху – опис структурної змінної T_PARA типу Para_SCALING, знизу – приклад програми на FBD.

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*)*

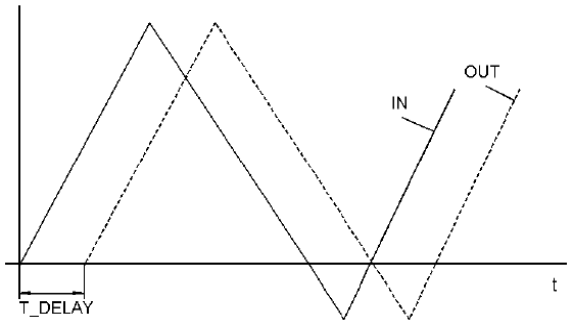

Рисунок 9.3– Діаграма роботи DTIME.

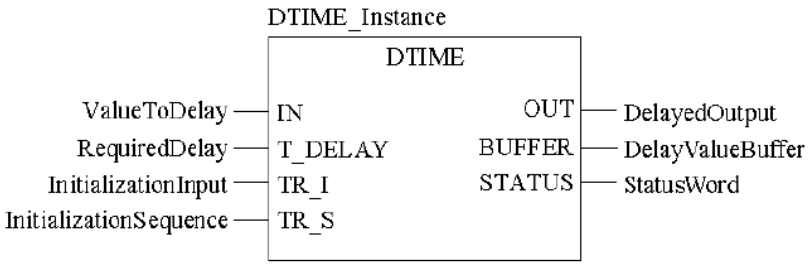

Рисунок 9.4– Приклад виклику блоку DTIME.

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.

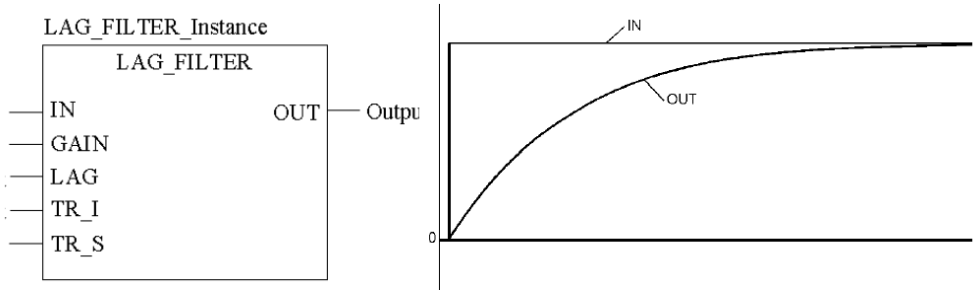

Рис.9.4. Приклад виклику блоку *LAG_FILTER* та діаграма його роботи.

Т *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