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).

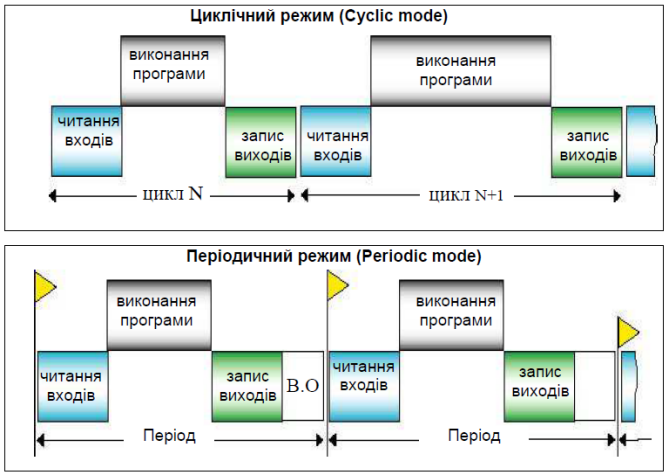


Рисунок 2.1–Циклічний та періодичний режим роботи задачі MAST

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".

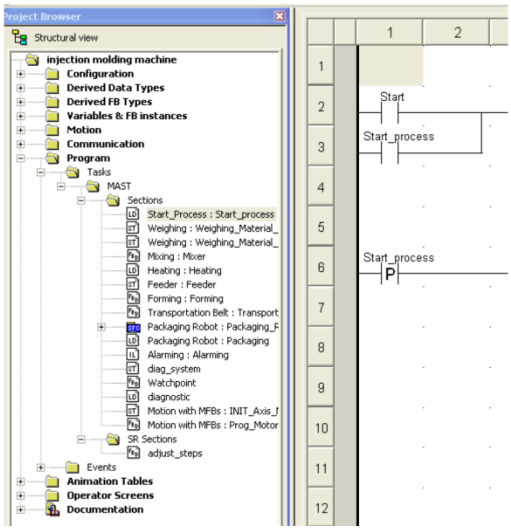
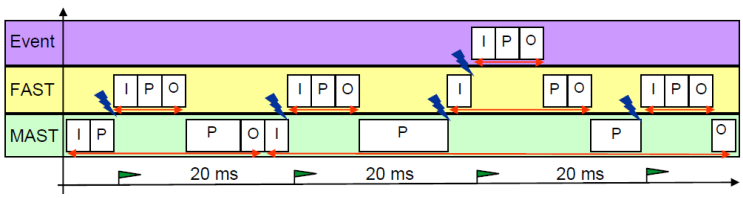


Рисунок 2.2–Приклад вигляду структури задачі MAST.

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

Задачі EVENT. Задачі ***Event*** викликаються коли відбувається певна подія: ***EVTi*** – по апаратним подіям, ***TIMERi*** – по таймерним. Задачі EVTi (де i – номер задачі даного типу) прив'язуються до певної події, які фіксуються модулями спеціального призначення. Задачі TIMERi (де i – 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 переривативиконання 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.

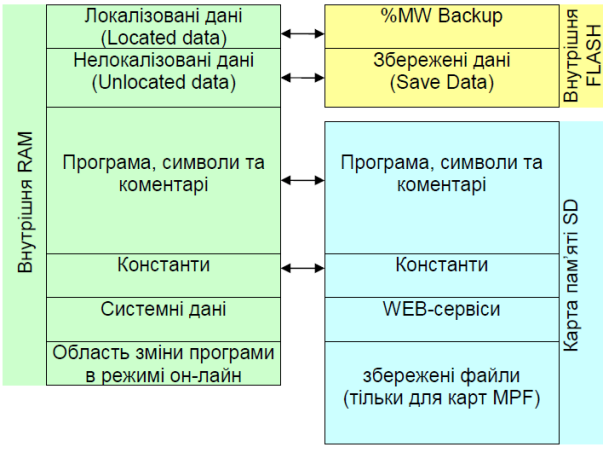


Рисунок 2.4 – Структура пам’яті M340

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 ескпонен-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).

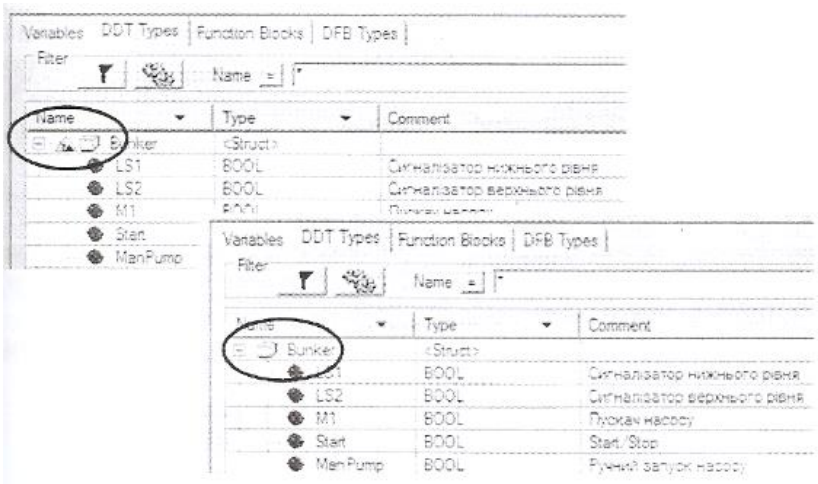


Рис. 2.5 Створення структурного типу користувача DDT угорі–до команди *Analyze*, внизу – після

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).

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

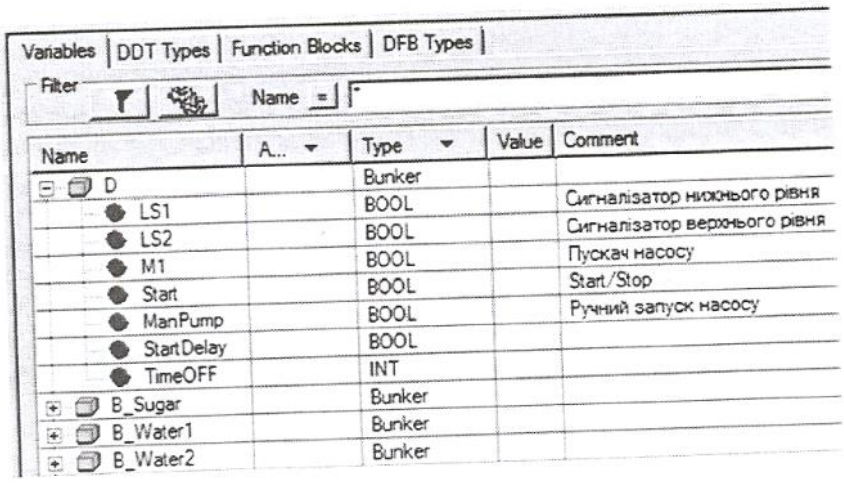


Рис.2.6 Структурні змінні.

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*.

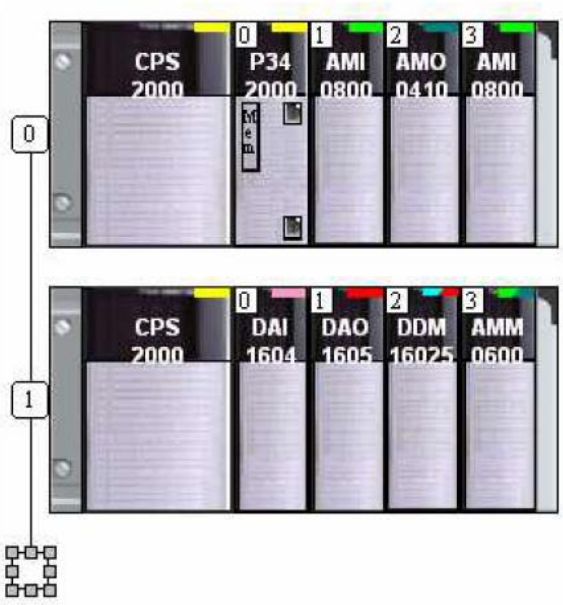
%IWr.m.c - адреса відповідає за аналоговий вхід на шасі з номером *r*, модуліна посадочному місці *m*, каналу *c*.

%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.