УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра спеціалізованих комп'ютерних систем

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних та самостійних робіт

з дисципліни
«АРХІТЕКТУРА ТА ПРОГРАМУВАННЯ ПРОМИСЛОВИХ СИСТЕМ КЕРУВАННЯ»

Харків – 2021

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри спеціалізованих комп'ютерних систем 17 лютого 2020 р., протокол № 9.

Методичні вказівки призначено для студентів факультету ІКСТ зі спеціальності 123 «Комп'ютерна інженерія» першого рівня вищої освіти (бакалавр) усіх форм навчання.

Укладач

доц. В. М. Бутенко

Рецензент

проф. В. І. Мойсеєнко

ЗМІСТ

ВСТУП

Метою методичних вказівок є вивчення практичних аспектів вибору архітектури контролерів промислової системи управління та отримання практичних навичок на основі алгоритмізації базових обчислювальних процесів [1] з програмуванням зазначених систем символьною мовою ST [2]. У подальшому перевірка їх роботи шляхом моделювання у середовищі програмування Unity Pro 3.0 або вище для створення елементів технологічного спрямування будь-якої предметної галузі.

Для створення програмних модулів спеціалізованих комп'ютерних систем промислового рівня використовується множина мов програмування, яка відповідає EN 61131/IEC 61131-3:2013 [3]. Лабораторний курс розглядається як вибіркова фахова складова та передбачає ознайомлення з відповідними методиками створення й дослідження моделей і підпрограм відповідно до навчальної програми бакалавра.

Побудова й дослідження моделей [4] і програмного забезпечення під час розроблення нових складних технічних систем промислового призначення часто є значною частиною у загальній сукупності робіт, які виконують для створення комплексних проектів створення/модернізації виробництва на базі кіберфізичних систем.

Для виконання лабораторних робіт у лабораторіях кафедри СКС університету встановлено програмне забезпечення (ПЗ) Unity PRO XL v 3.0, яке надала фірма Шнайдер Електрик Україна (http://www.schneider-electric.ua), що функціонує у навчально-демонстраційному режимі. Для самостійної роботи студентів можливе застосування будь-якої версії легального ПЗ Unity PRO. Однак у разі невідповідності версії на студентському та лабораторному ПК саме на студента покладається відповідальність за демонстрацію самостійно розроблених програмних модулів під час виконання лабораторної роботи (ЛР). Практика показує, що краще або застосовувати версію 3.0, або приносити переносний ПК для демонстрації самостійно розроблених схем-програм лабораторного практикуму.

ЛАБОРАТОРНА РОБОТА 1
Конфігурування контролера визначеної архітектури та створення нових проектів мовою ST в Unity Pro

Мета роботи: отримання практичних навичок установлення, створення проектів і програм мовою **ST** (Structured Text) в Unity Pro.

Обладнання та ПЗ: персональна електронно-обчислювальна машина (ПЕОМ) із системним програмним забезпеченням (СПЗ) Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР [2, 5, 6].

2 Створити проект за послідовністю, викладеною нижче, та ввести програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Увести програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити висновки й отримати бали за ЛР у викладача.

Послідовність створення проекту

Unity PRO – це програмне середовище конфігурації, програмування, налагодження і діагностування виконавчої системи та промислових контролерів, яке є результатом розвитку двох програмних продуктів: PL7 PRO – середовище програмування першої групи контролерів, Concept – середовище програмування другої групи контролерів.

Виконавча система Unity – це ПЗ, яке виконується у контролері. Виконавча система базується на операційній системі (ОС) Unity, яка вже перебуває у завантаженні програмного логічного контролера (ПЛК) та бере участь у всіх операціях, але

нас цікавить моделювання роботи схем булевої алгебри в зазначеному середовищі.

На робочому столі знаходимо ярлик програми Unity PRO, запускаємо його (або через меню "Пуск"–програм). Потім створюємо (**File-New**) на диску ПЕОМ (у лабораторії кафедри D:\SE) новий проект (використавши для його назви тільки цифри й латинські літери ASCII таблиці) та обираємо процесорний модуль зазначеної архітектури **TSX P57 2634M**, вибравши тип шасі (X-Bas) і, **за бажанням,** скомпонувавши ПЛК модулями введення-виведення за допомогою панелі, яка розміщена у лівому нижньому вікні (рисунок 1), для прикладу навчального класу **Schnider Electric УкрДУЗТ –** модулі дискретні TSX DEY 32D2K TSX, DSY 32T2K та аналоговий TSX AEY 414.

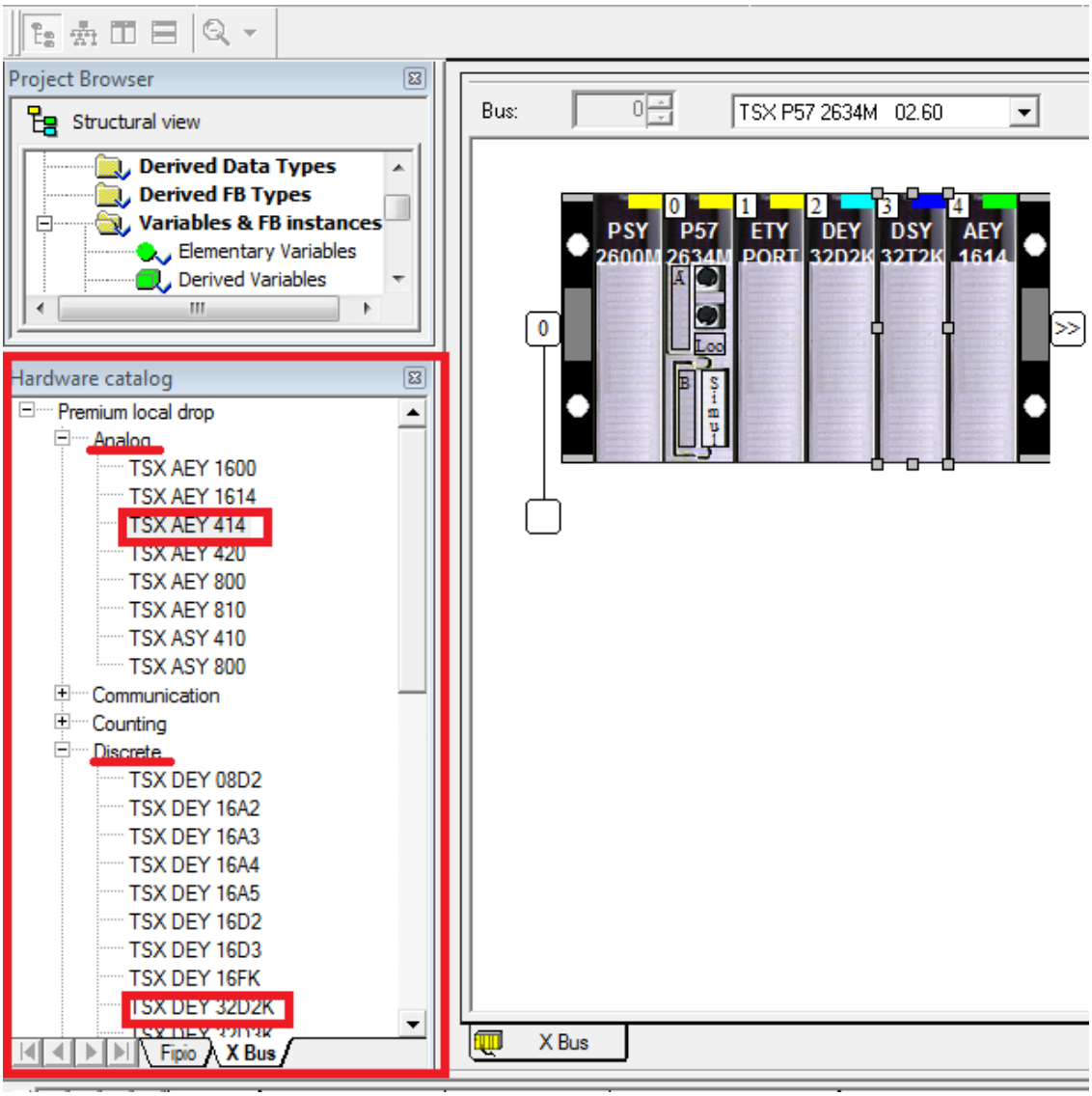


Рисунок 1 – Конфігурування ПЛК у проекті

Після завершення компонування заданих проектом або планом ЛР модулів (як показано для ЛР на рисунку 2) активізуємо модуль ETY PORT і відкриємо його для налаштування (рисунок 3). За бажанням, аналогічно налаштовуються інші модулі ПЛК.

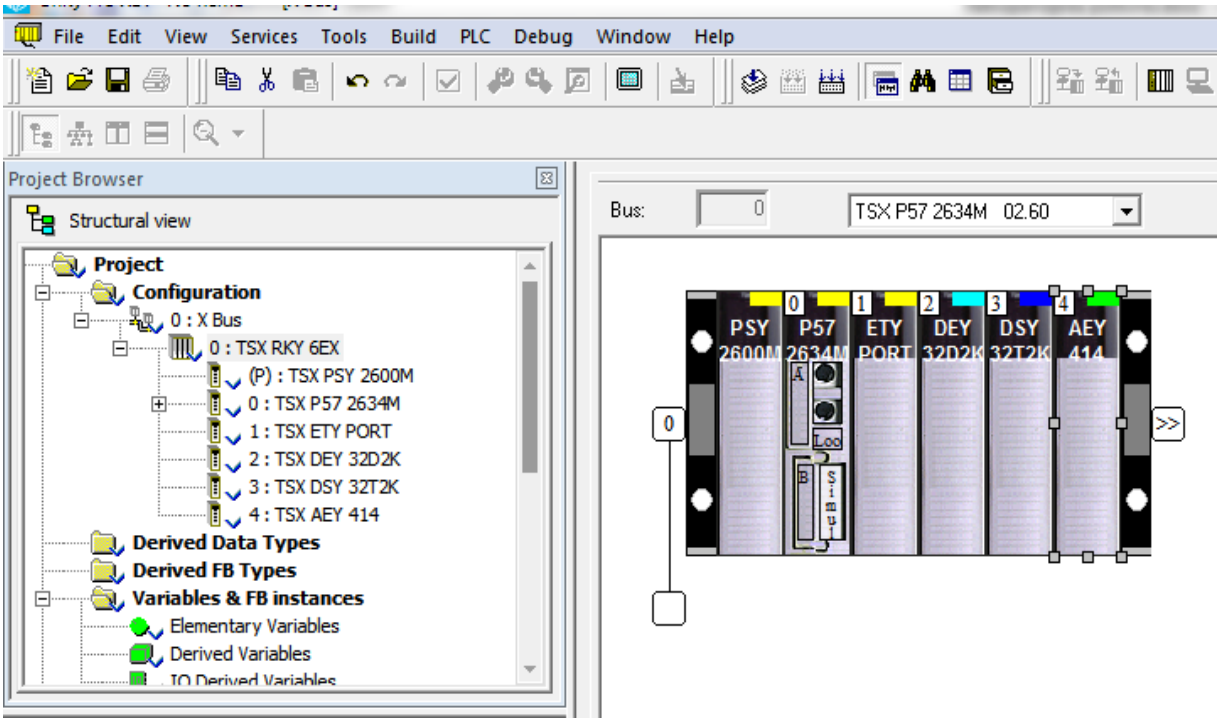


Рисунок 2 – Вибір модуля ПЛК

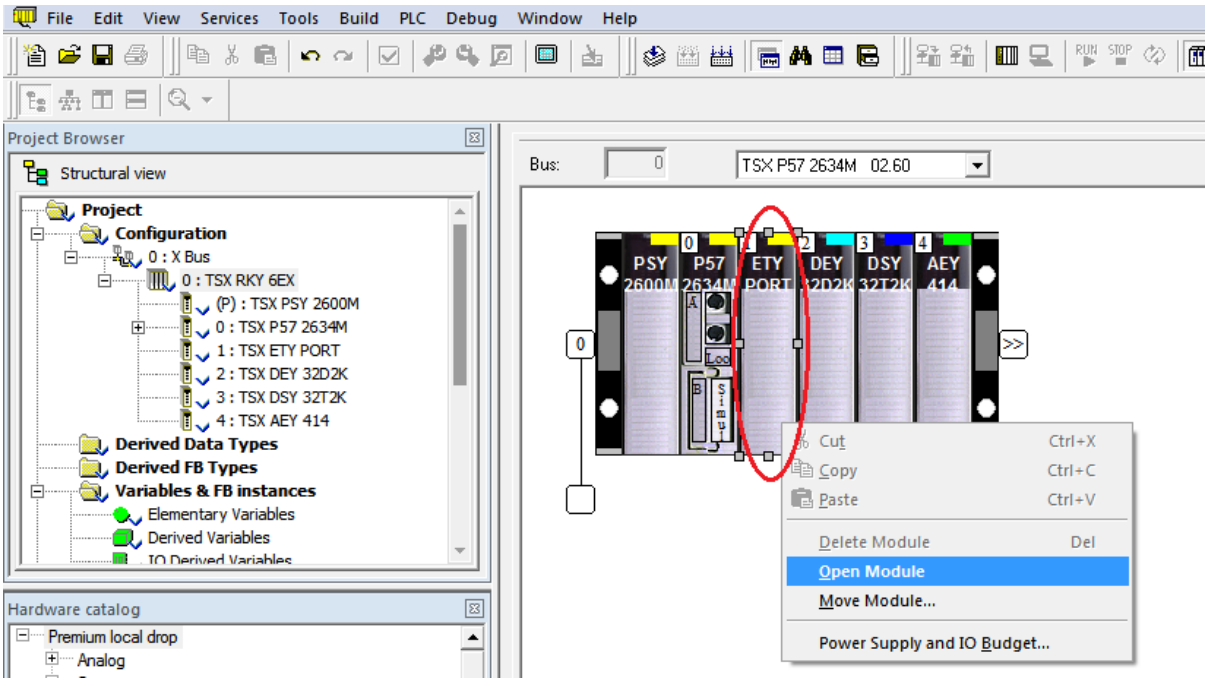
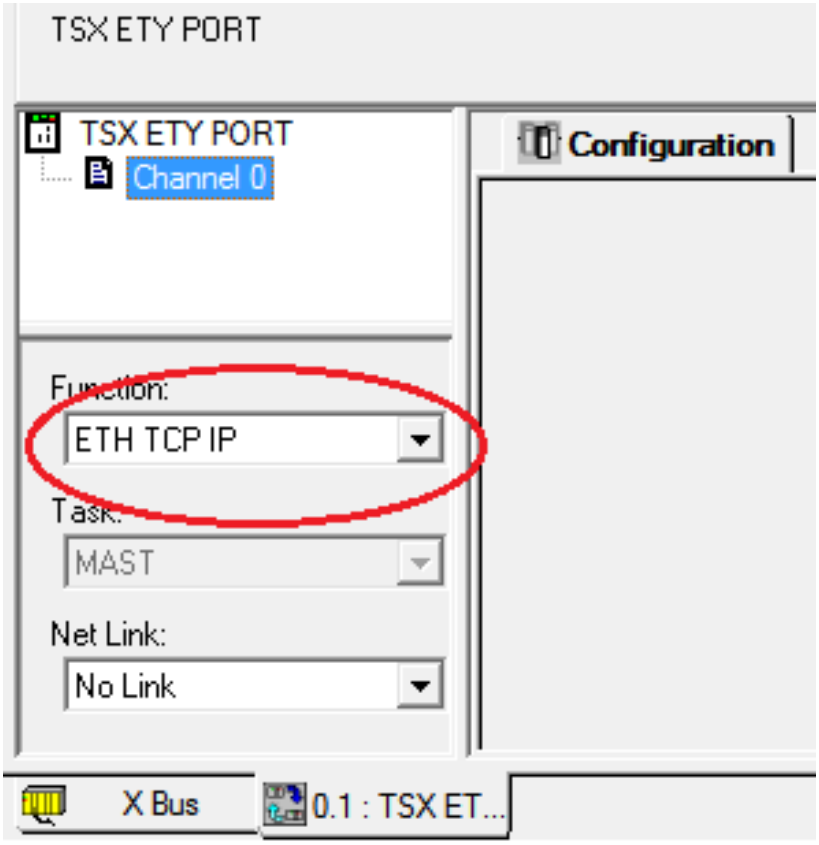


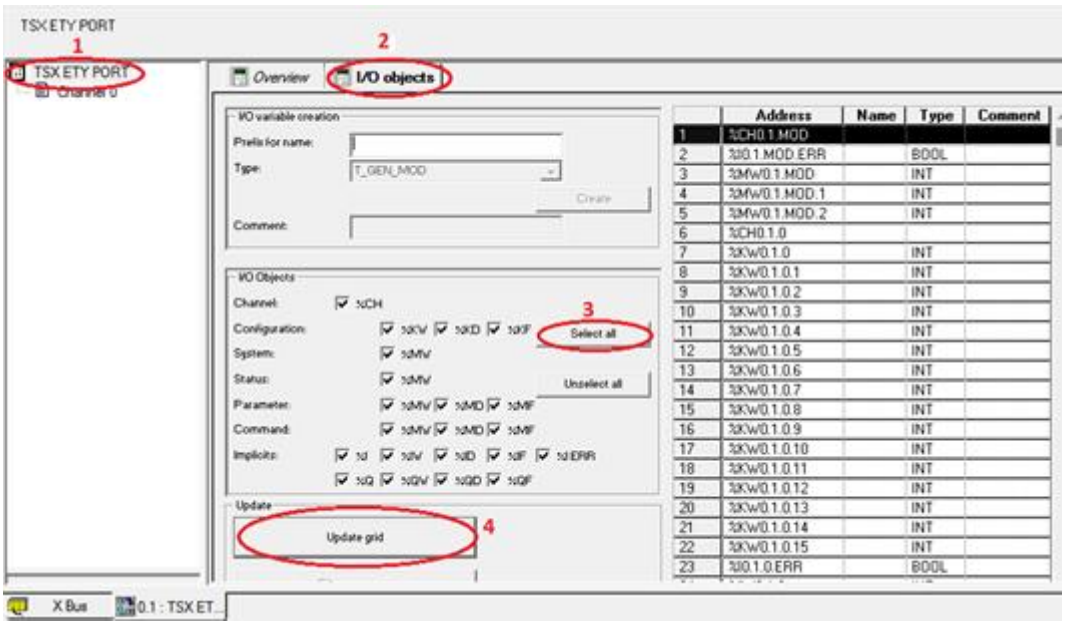
Рисунок 3 – Налаштування модуля TSX ETY PORT

У цьому самому вікні обираємо TSX ETY PORT (рисунок 4, а) (Open Module –> Channel 0 –> знизу Function), перемикаємо на ETH TCP IP і, зробивши налаштування у закладці I/O objects (рисунок 4, б), зберігаємо налаштування.

а



б



а – вибір протоколу ETH TCP IP;
б – налаштування у закладці I/O objects
Рисунок 4 – Налаштування модуля TSX ETY PORT

Далі переходимо в меню PLC (англ. *Programmable Logic Controller (PLC)*), як показано на рисунках 5, 6.

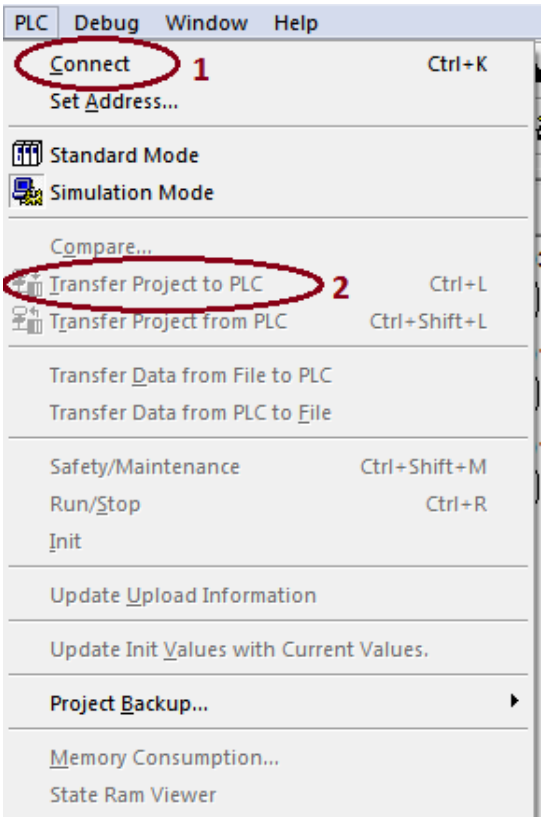


Рисунок 5 – Зв'язок із контролером

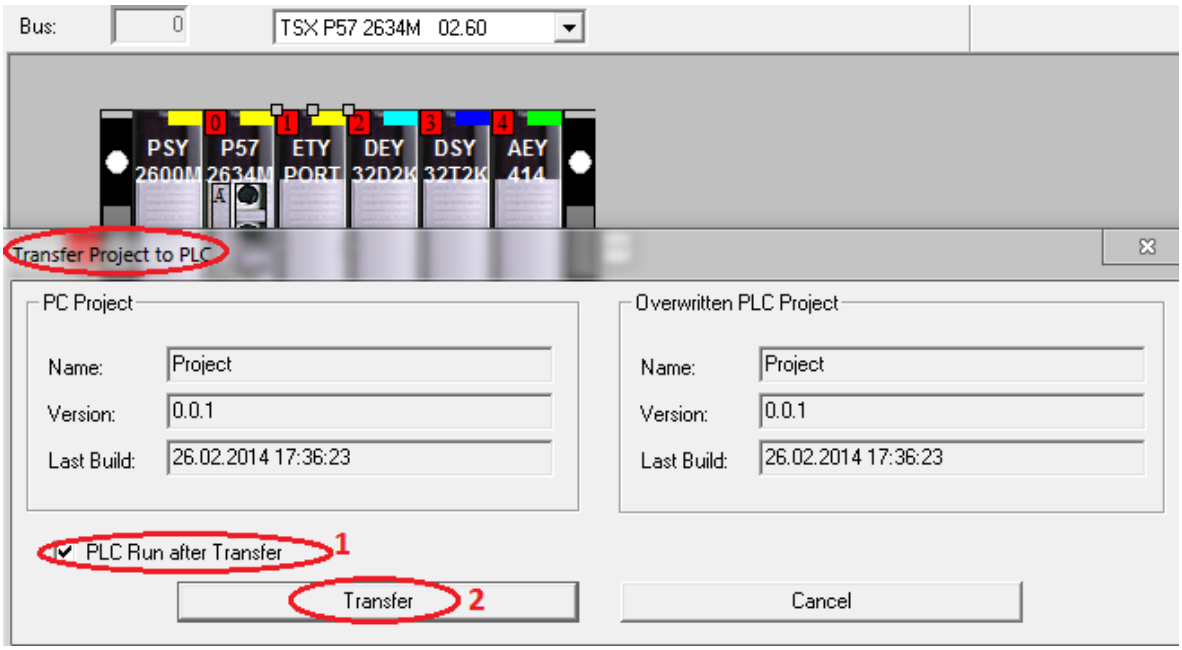


Рисунок 6 – Передача проекту до ПЛК/PLC

Якщо фізичний ПЛК не підключений до ПК, то необхідно перевірити встановлення опції «Simulation mode».

Для того, щоб зупинити програму, обираємо меню PLC і натискаємо Stop, потім знову PLC та обираємо Disconnect.

Послідовність дій введення типового завдання проекту

Типове завдання ЛР 1. Розробити схему-програму мовою ST кодування стану чотирьох кнопок (булеві змінні a0, a1, a2, a3) у дворозрядний код (булеві змінні x1, x0) та візуалізувати коди x1x0 (00, 01, 10, 11) на стандартному семисегментному індикаторі (рисунок 7, а) із сегментами (a, b, c, d, e, f, g) у вигляді цифр 0, 1, 2, 3 відповідно до значень, показаних у таблиці 1.

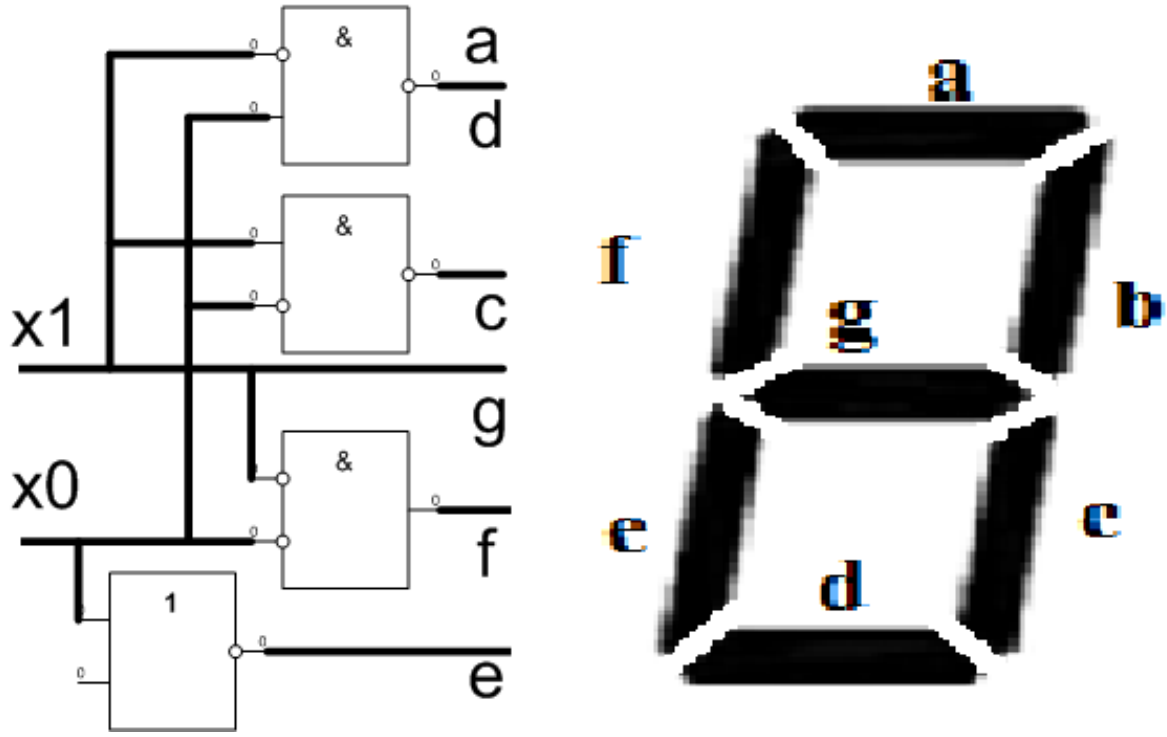
Таблиця 1 – Відповідність значень булевих змінних у різних елементах схеми типового завдання ЛР 1

	Код		Сегменти							Цифра
	x1	x0	a	b	c	d	e	f	g	
Значення	0	0	1	1	1	1	1	1	0	0
	0	1	0	1	1	0	0	0	0	1
	1	0	1	1	0	1	1	0	1	2
	1	1	1	1	1	1	0	0	1	3

За отриманими раніше аналітичними виразами визначення a0= $\overline{a1 + a2 + a3}$ та кодування x0=a3+a2, x1= $\overline{a0}\ \overline{a1}$ сформовано a=d=$\overline{\overline{x1}\ x0}$, b=1, c= $\overline{x1\overline{x0}}$, e=$\overline{x0}$, f= $\overline{x0}\ \overline{x1}$, g=x1. На рисунку 7, б зображено семисегментний індикатор з позначенням сегментів та комбінаційну схему перекодування a0, a1, a2, a3 в x0 та x1 із увімкненням усіх сегментів на рисунку 7, а.

При описі цієї роботи було використано ПЗ Unity Pro 3.0, і саме скриншоти з нього наведено в тексті, але в окремих випадках використовувались зображення з інших версій з подібними повідомленнями та ідентичними функціями.

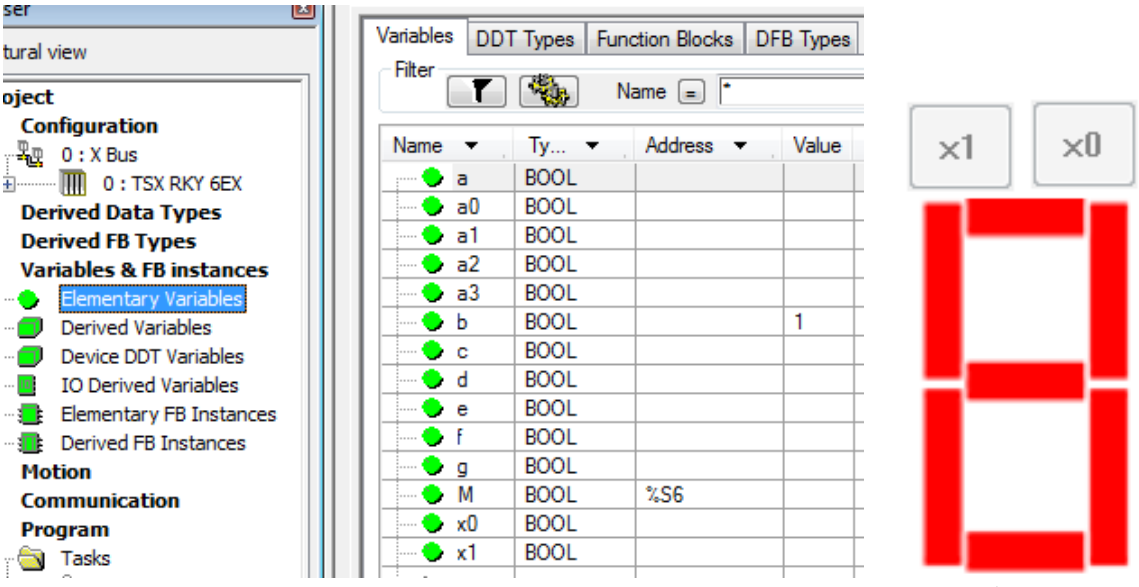
Наступний крок виконання ЛР – перейти у вікно **Project Browser** (один з варіантів – комбінацією клавіш Alt+1), на вкладці якого знайти рядок **Elementary Variables,** вибрати його та ввести 14 змінних типу **BOOL** як показано на рисунку 8.



а б

а – схема перекодування x0, x1 із увімкненням сегментів;
б –зображення семисегментного індикатора з позначенням сегментів

Рисунок 7 – Зображення індикації



а б

а – зображення вікна створення змінних типу BOOL;
б – спрощене зображення семисегментного індикатора та двох кнопок шин сигналів x0 та x1

Рисунок 8 – Змінні програми

Ще один крок виконання ЛР – у вікні **Project Browser** (один із варіантів – комбінацією клавіш Alt+1), на вкладці якого знайти рядок Program, обрати Tasks, потім MAST, далі Section і правою кнопкою миші (ПКМ) викликати контекстне меню і вибрати New Section (рисунок 9).

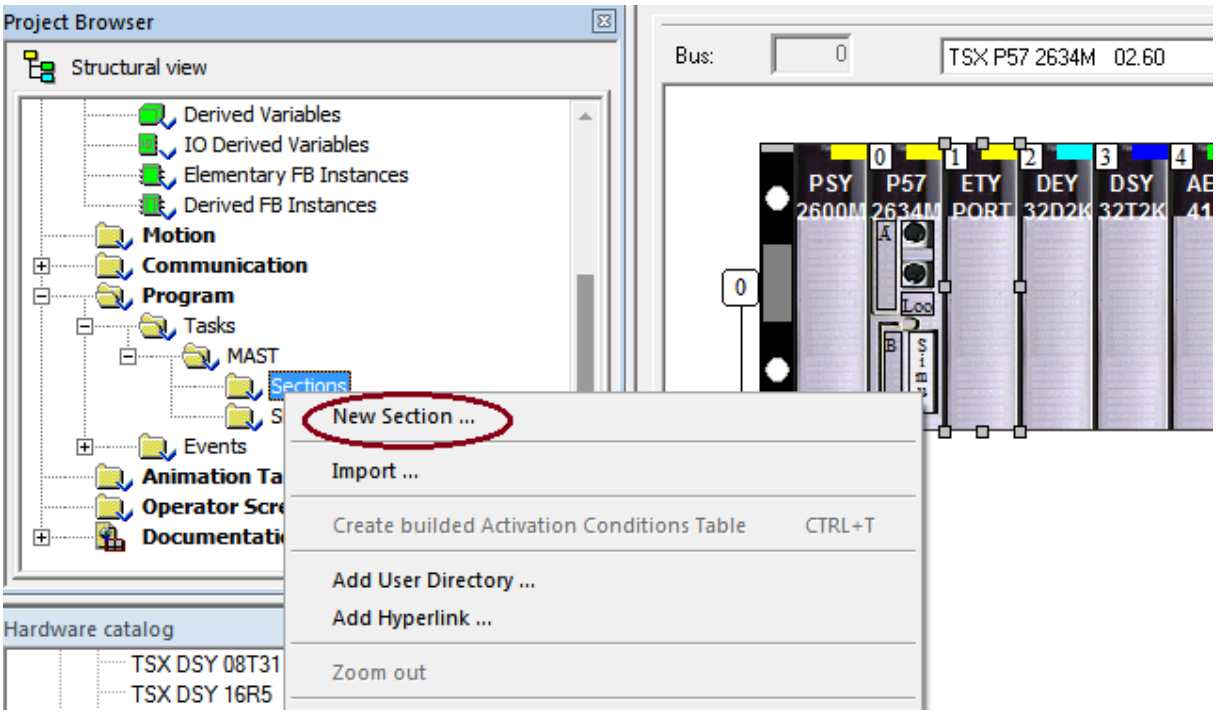


Рисунок 9 – Створення секції програми в групі MAST

Далі вводимо ім’я секції (використовувати тільки латинські літери з можливим додаванням цифр, наприклад **laba01**) та обираємо параметр мови програмування (ST) (рисунок 10).

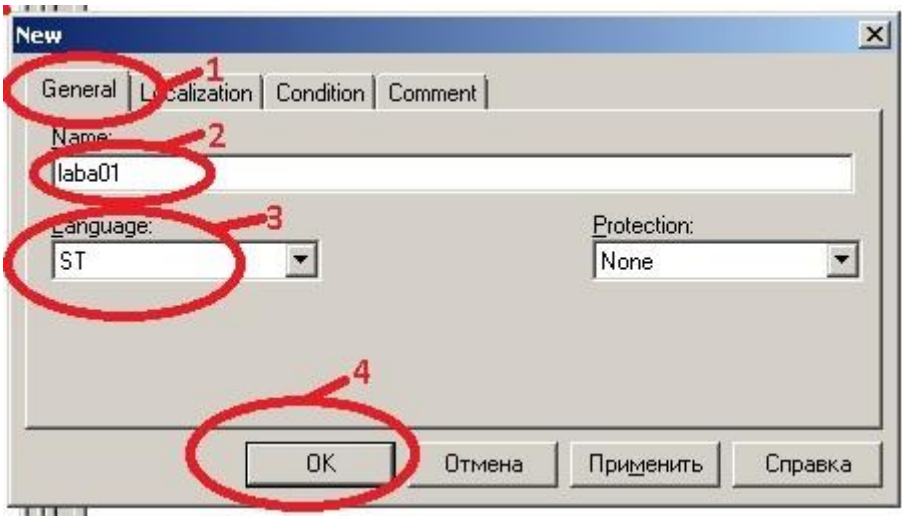
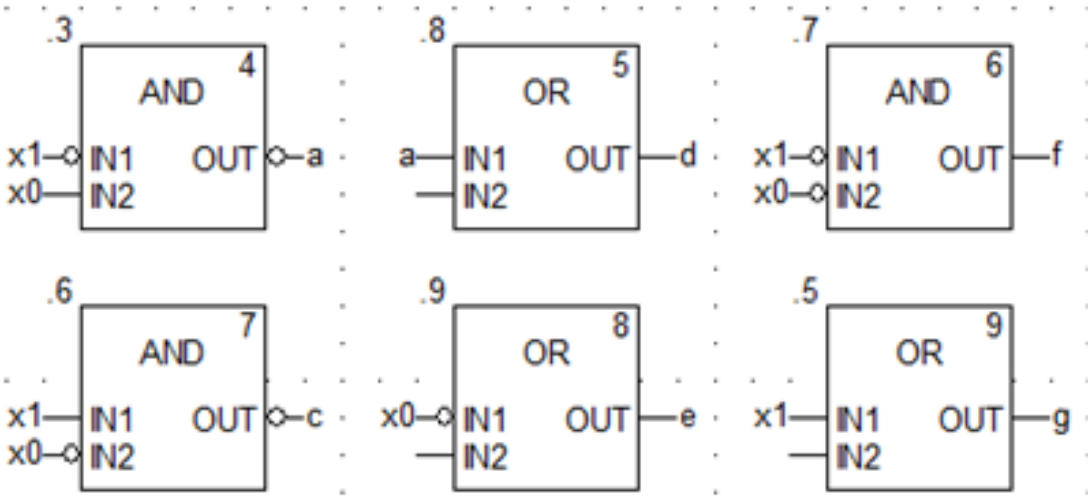


Рисунок 10 – Введення назви секції та вибір мови програмування

За допомогою кнопок (наприклад, ) на верхній допоміжній панелі будуємо схему-програму мовою FBD або мовою ST як показано на рисунку 11. Для вибору блоків **OR**-АБО, **AND**-ТА можливе використання комбінації клавіш Ctrl+I.

а



б

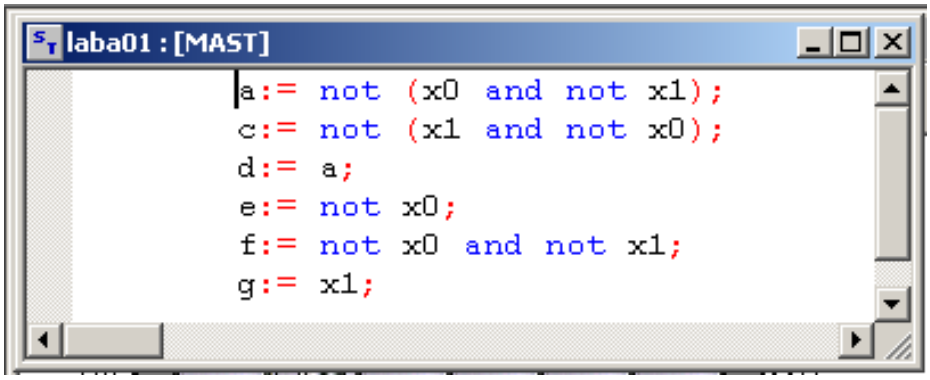


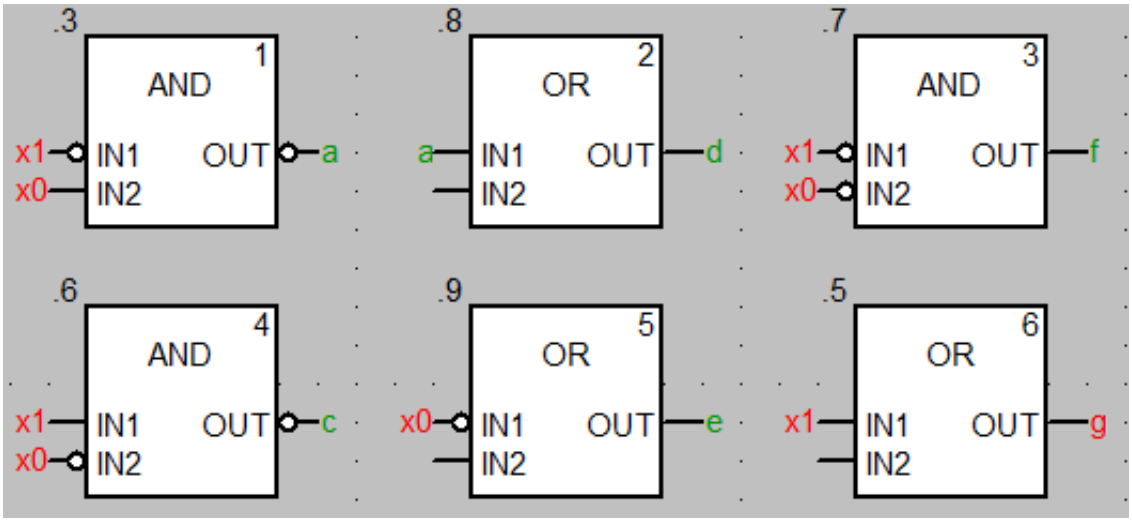
Рисунок 11 – Приклад програми мовою FBD (а) мовою ST (б)

Після компіляції (Ctrl+B), з'єднання з ПЛК (Ctrl+K), передачі/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуску (Run) її у віртуальному ПЛК (Ctrl+R) має з’явитись зображення схеми-програми мовою FBD (рисунок 12) з індикацією зеленим кольором булевих змінних, які мають стан «1», та червоним кольором – змінних, які дорівнюють «0».

Після перевірки правильності роботи схеми за таблицею істинності (таблиця 1) рекомендується зупинити ПЛК (повторним натисканням Ctrl+R) та від'єднатися від нього (Ctrl+K). Далі стає можливим візуалізація роботи схеми-програми на екрані користувача «Operator Screen».

Наступний крок виконання ЛР – у вікні **Project Browser** (один з варіантів виклику – комбінацією клавіш Alt+1) на лівому вікні (рисунок 13, а) обираємо «Operator Screen», ПКМ викликаємо меню і вибираємо «New Screen».

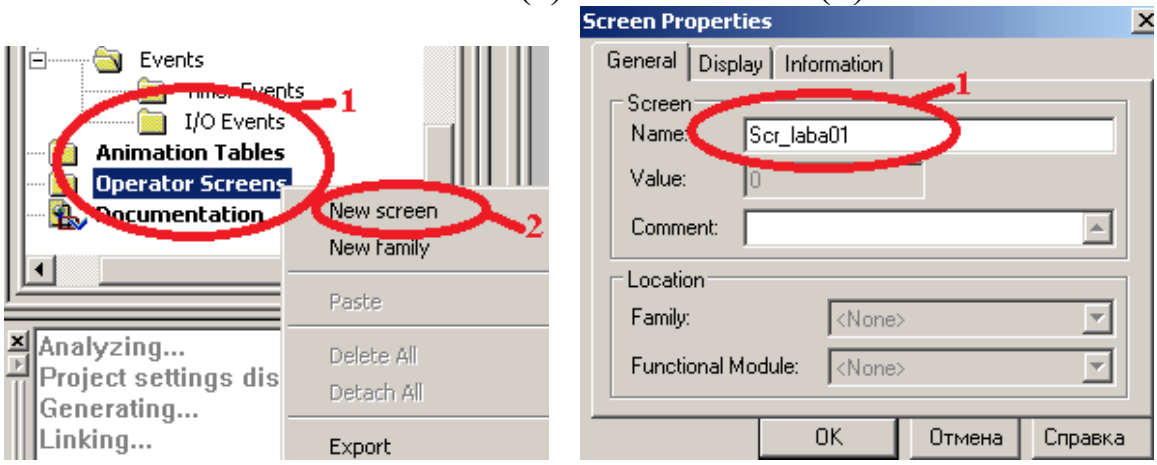
а



б

```
a:= not (x0 and not x1);
c:= not (x1 and not x0);
d:= a;
e:= not x0;
f:= not x0 and not x1;
g:= x1;
```

Рисунок 12 – Зображення працюючої програми мовою FBD (а) та мовою ST (б)



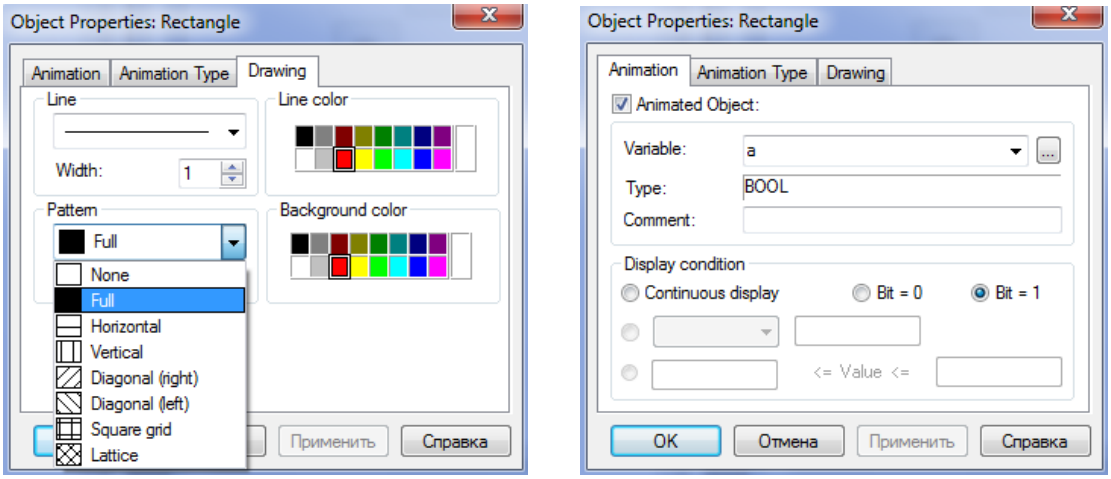
а б

а – створення нового елемента екранної візуалізації;
б – створення назви вікна налаштування візуалізації

Рисунок 13 – Параметри візуалізації

У вікні, яке з’явилось, записуємо назву Scr_laba01 (рисунок 13, б). На полі цього вікна проектуємо спрощене зображення семисегментного індикатора й двох кнопок шин сигналів x0 та x1 (рисунок 8, б).

Для цього в утвореному вікні візуалізації за допомогою ПКМ створюємо новий об’єкт–сегмент «a» типу Rectangle – «прямокутник» та, знову натискаючи ПКМ, за допомогою опції Properties задаємо параметри як на рисунках 14, 15.



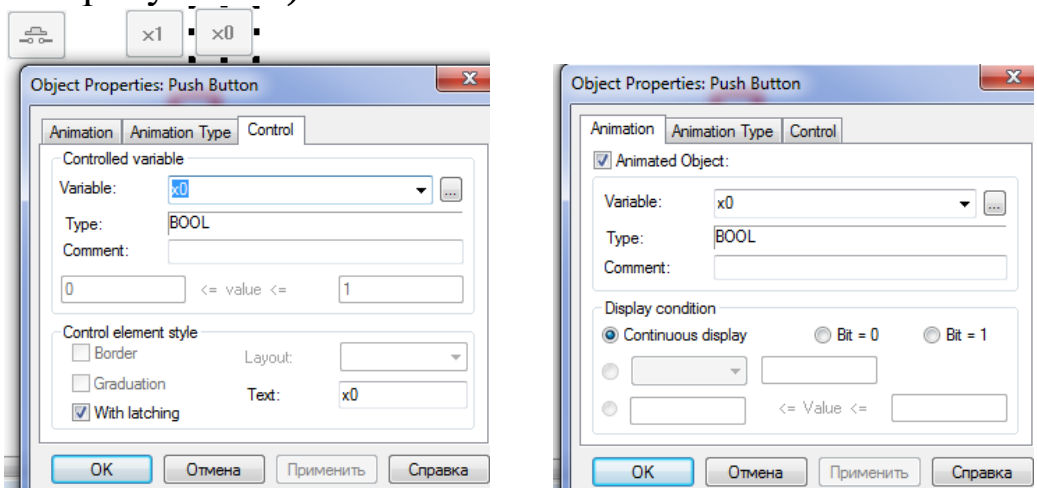
а б

а – візуалізація; б – анімація об'єкта – сегмента «a»

Рисунок 14 – Налаштування форми зображення елемента

За аналогічною процедурою створюємо решту сегментів.

Наступним кроком візуалізації є створення кнопок типу Push Button із назвами x0 та x1 для емуляції сигналів x0 та x1 (зразок – рисунок 15).



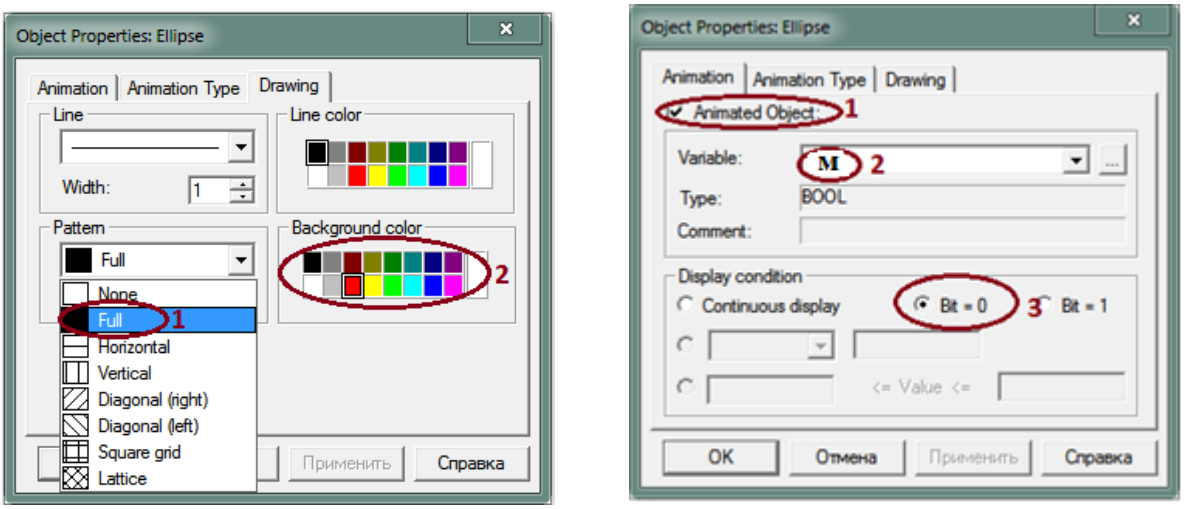
а б

а – візуалізація; б – анімація об'єкта – кнопка шини «x0»

Рисунок 15 – Налаштування форми зображення елемента

За аналогічною процедурою створюється кнопка шини «x1». Наступним кроком виконання роботи є повторне з'єднання із ПЛК (Ctrl+K), передача/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуск (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад Scr_laba01) та після натискання F7 або увімкнувши режим «Enable Variable Modification» іншим способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин x1x0, 00, 01, 11, 10 та перевіряти правильність зображень цифр за таблицею 1. У разі невідповідності виконати коригування схеми-програми або налаштування об’єктів екранів оператора та повторну перевірку.

На операторському екрані, у вікні візуалізації, за допомогою ПКМ створюємо новий об’єкт «коло» та, знову натискаючи ПКМ, за допомогою опції Properties задаємо параметри як на рисунках 16.



а б

а – візуалізація; б – анімація об'єкта

Рисунок 16 – Налаштування форми зображення елемента «коло»

Підсилене завдання. Зобразити у зошиті схему-програму мовою **ST** шифрування сигналів від натискання кнопок a1, a2, a3 в код змінних x1, x0. Відображення на семисегментному індикаторі має здійснюватися автоматично за таблицею 1.

Відповісти на контрольні питання, виконати індивідуальне завдання.

Контрольні питання

1 Як сконфігурувати архітектуру ПЛК з різними модулями в проекті?

2 Які типи та категорії модулів виробляються?

3 Як активувати модуль ETY PORT?

4 Які мови програмування підтримуються Unity PRO?

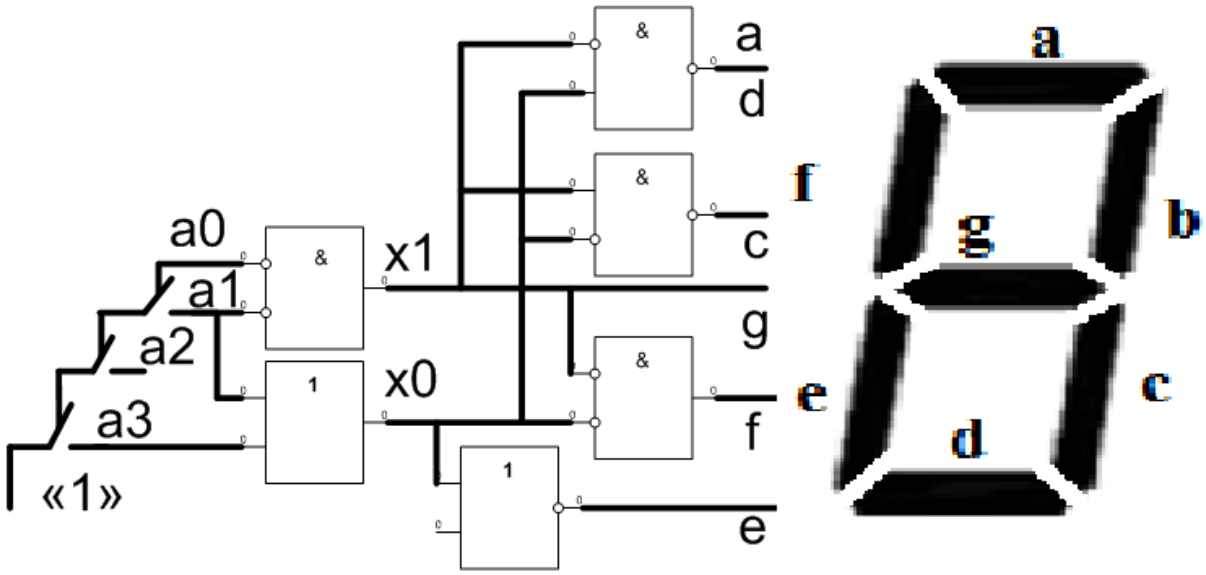
5 У якому вікні можливо зображати кола, прямокутники?

6 Як зробити візуалізацію, залежну від змінних проекту?

7 Які розширення мають файли з проектом Unity PRO?

Індивідуальне завдання студента

Розширити завдання ЛР, створивши програму перетворення вхідних сигналів a1, a2, a3 в сигнали x0 та x1 та a0 за прикладом завдання 1.



Завдання 1 – Розширення вмикання індикатора

Додатково створити необхідні вхідні та вихідні змінні та в окремій секції програм реалізувати функцію, відобразивши результат на тому самому операторському екрані (наприклад Scr_laba01). Приймаючи символ «∩»/∧ за кон'юнкцію («і»), символ «U»/V за диз'юнкцію, реалізувати функції за варіантами (вихідні змінні візуалізувати у вигляді кружечків зеленого (для «1») та червоного (для «0») кольорів:

1) $y1 = x1 \cap x2 \cap x3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x6 \cup \bar{x}7$;

2) $y2 = \bar{x}1 \cap x2 \cap \bar{x}3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = \bar{x}6 \cup \bar{x}7$;

3) $y3 = \bar{x}1 \cap \bar{x}2 \cap \bar{x}3 \cap (x4 \cup \bar{x}5) \cup y19, y19 = \bar{x}6 \cup x7 \cap x9$;

4) $y4 = \bar{x}1 \cap x2 \cap \bar{x}3 \cap \bar{x}4 \cap x5 \cup y100, y100 = \bar{x}6 \cup x7$;

5) $y5 = x1 \cap x2 \cap x3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}7$;

6) $y6 = x1 \cap x2 \cap x9 \cap (\bar{x}4 \cup \bar{x}6) \cap y19, y19 = x8 \cup \bar{x}7$;

7) $y7 = x9 \cap x2 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}7 \cup \bar{x}3$;

8) $y8 = x9 \cap x2 \cap x3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}7 \cup \bar{x}1$;

9) $y9 = x1 \cap x9 \cap x3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}7 \cup \bar{x}2$;

10) $yA = x1 \cap x2 \cap x9 \cap (\bar{x}4 \cup \bar{x}3) \cap y19, y19 = x8 \cup \bar{x}7 \cup \bar{x}5$;

11) $yB = x1 \cap x2 \cap x3 \cap (\bar{x}9 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}9 \cup \bar{x}7$;

12) $yC = x9 \cap \bar{x}2 \cap x3 \cap (\bar{x}4 \cup \bar{x}5) \cap y19, y19 = x8 \cup \bar{x}7 \cup \bar{x}1$.

ЛАБОРАТОРНА РОБОТА 2
Проектування програми вмикання семисегментного індикатора CCI_3x8

Мета роботи: отримання практичних навичок синтезу комбінаційних схем і розроблення програм мовою ST в Unity Pro, налагодження елементарних функцій алгебри логіки.

Обладнання і ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за наведеною після теоретичного матеріалу послідовністю та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Увести програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

Теоретичний матеріал

У пристроях залізничної автоматики, телемеханіки, обчислювальної техніки, зокрема у мікропроцесорах, є багато комбінаційних схем. Під комбінаційними схемами розуміють логічні схеми, сигнал на виході яких у кожний момент часу визначається комбінацією вхідних сигналів у той самий момент часу.

Синтез комбінаційних схем полягає у визначенні таких способів поєднання деяких найпростіших схем, названих логічними елементами (таблиця 2), за допомогою яких побудований пристрій реалізує поставлене завдання із перетворення вхідної двійкової інформації.

Синтез комбінаційних схем поділяють на чотири етапи:

1) утворення таблиці істинності для функції алгебри-логіки (ФАЛ), яка описує роботу проектованої логічної схеми (частіше за все на підставі словесного опису принципу роботи);

2) утворення математичної формули для ФАЛ, що описує роботу схеми, яку синтезують, у вигляді досконалої диз'юнктивної нормальної форми (ДДНФ) або досконалої кон'юктивної нормальної форми (ДКНФ) (на підставі таблиці істинності);

3) аналіз отриманої ФАЛ для побудови різних варіантів її математичного виразу (на основі законів булевої алгебри) та знаходження найкращого з них відповідно до того чи іншого критерію. На цьому етапі здійснюється мінімізація ФАЛ;

4) утворення функціональної (логічної) схеми пристрою з елементів, які становлять вибраний базис. **Але у нашому випадку синтез комбінаційних схем реалізується схемою-програмою мовою FBD для ПЗ Unity Pro 3.0 або вище.** У таблиці 2 наведено функції й еквівалентні їм FFB-типи мови FBD, а також їх запис за допомогою операцій І, АБО, НІ.

Диз'юнкти́вна норма́льна фо́рма (**ДНФ**) у булевій логіці — нормальна форма, у якій булева формула має вигляд диз'юнкції декількох кон'юнктів (де кон'юнктами називаються кон'юнкції декількох пропозиційних символів або їх заперечень).

ДДНФ булевої функції називається диз'юнкція тих конституент одиниці, які перетворюються в одиницю на тих самих наборах змінних, що й задана функція. ДДНФ повинна задовольняти такі умови:

- у ній немає однакових доданків;
- жоден із доданків не містить двох однакових співмножників;
- жоден із доданків не містить змінну разом із її запереченням;
- у кожному окремому доданку є як співмножник або змінна x_i, або її заперечення для будь-якого i = 1, 2, …, n.

У практиці перетворювання логічних формул є чіткий порядок виконання дій. Якщо у виразі немає дужок, першими мають виконуватися операції інверсії (заперечення), потім — операції кон'юнкції (логічного множення), останніми — операції диз'юнкції (логічного складання).

Наявність у виразі дужок змінює порядок дій, і тоді в першу чергу виконуються операції у дужках.

Кількість різних наборів значень аргументів ФАЛ кінцева, зважаючи на це будь-яка ФАЛ може бути задана таблицею з 2^N рядками. Зліва у таблиці (таблиця 3) проставляються номери рядків (номери цифр, що відображатимуться), потім – значення функції семи сегментів для кожного з наборів змінних, а справа — набори значень аргументів функції. Такий спосіб задання функції для семи сегментів названо табличним.

Синтез логічних пристроїв у різних базисах розглянемо на прикладі побудови схем увімкнення сегментів семисегментного індикатора з трьома входами x2, x1, x0.

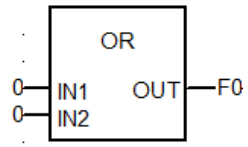
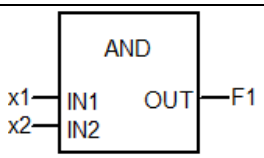
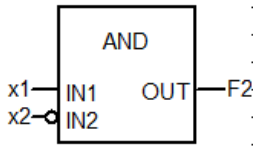
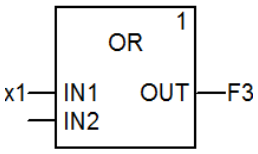
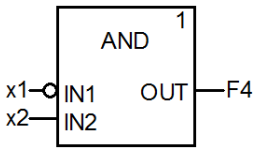
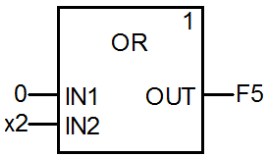
Послідовність створення проекту

Для виконання ЛР 2 необхідно відкрити створений проект ЛР 1 та зберегти його як laba02 або самостійно створити проект за процедурою, наведеною у ЛР 1, із назвою laba02.

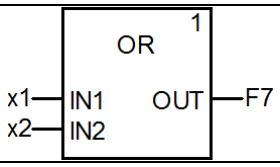
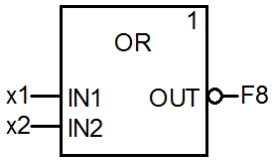
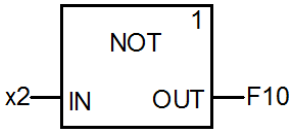
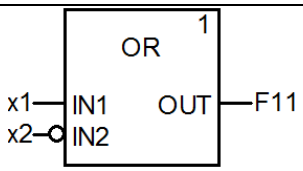
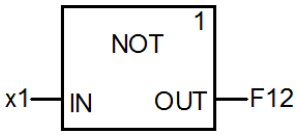
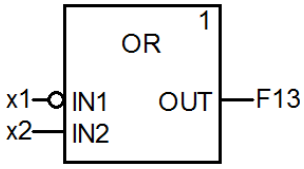
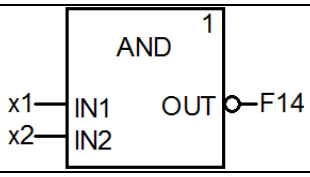
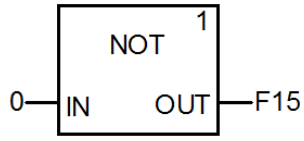
Послідовність дій введення типового завдання проекту

Розробити схему-програму мовою FBD кодування стану восьми кнопок (булеві змінні a0, a1, a2, a3, a4, a5, a6, a7) у дворозрядний код (булеві змінні x2, x1, x0) та візуалізувати коди x2x1x0 (000, 001, 010, 011, 100, 101, 110, 111) на стандартному семисегментному індикаторі із сегментами (a, b, c, d, e, f, g) у вигляді цифр, зображення яких пропонується виводити в такому вигляді:  , відповідно до значень, показаних у таблиці 3.

Таблиця 2 – Функції та еквівалентні їм FFB-типи мови FBD

Логічні агументи і функції	Значення аргументів і функцій				Функція (з назвою у дужках) та її запис за допомогою операцій І, АБО, НІ	FFB-тип логічного елемента схеми-програми мовою FBD для ПЗ Unity Pro
Аргум x_1	0	0	1	1		
Аргум x_2	0	1	0	1		
1	**2**	**3**	**4**	**5**	**6**	**7**
F_0	0	0	0	0	Константа нуль (генератор нуля) $F_0 = 0$	
F_1	0	0	0	1	Кон'юнкція (кон'юнктор) $F_1 = x_1 \cdot x_2$	
F_2	0	0	1	0	Заборона 2-го аргументу (елемент заборони) $F_2 = x_1 \cdot \overline{x_2}$	
F_3	0	0	1	1	Повторення1-го аргументу (повто-рювач) $F_3 = x_1$	
F_4		1	0	0	Заборона 1-го аргументу (елемент заборони) $F_4 = \overline{x_1} \cdot x_2$	
F_5	0	1	0	1	Повторення 2-го аргументу (повторювач) $F_5 = x_2$	
F_6	0	1	1	0	Нерівнозначність (суматор за модулем 2) $F_6 = \overline{x_1}\, x_2 + x_1 \overline{x_2} = x_1 \oplus x_2$	

Продовження таблиці 2

1	2	3	4	5	6	7
F_7	0	1	1	1	Диз'юнкція (диз'юнктор) $F_7 = x_1 + x_2$	
F_8	1	0	0	0	Операція Пірса (елемент Пірса) $F_8 = \overline{x_1 + x_2}$	
F_9	1	0	0	1	Рівнозначність (еквівалентор) $F_9 = x_1 x_2 + \overline{x_1}\,\overline{x_2}$	
F_{10}	1	0	1	0	Заперечення 2-го аргументу (інвертор) $F_{10} = \overline{x_2}$	
F_{11}	1	0	1	1	Імплікація від 2-го аргументу до 1-го $F_{11} = x_1 + \overline{x_2}$	
F_{12}	1	1	0	0	Заперечення 1-го аргументу (інвертор) $F_{12} = \overline{x_1}$	
F_{13}	1	1	0	1	Імплікація від 1-го аргументу до 2-го (імплікатор) $F_{13} = \overline{x_1} + x_2$	
F_{14}	1	1	1	0	Операція Шеффера (елемент Шеффера) $F_{14} = \overline{x_1 \cdot x_2}$	
F_{15}	1	1	1	1	Константа одиниця (генератор одиниці) $F_{15}=1$	

Для синтезу схем, що реалізують різні ФАЛ, найбільш зручним є аналітичний спосіб задання ФАЛ, коли ФАЛ має вигляд алгебраїчного виразу, який отримують у результаті використання яких-небудь логічних операцій до змінних функції алгебри логіки. Для знаходження ДДНФ з таблиці вибирають

тільки ті рядки, де стоять набори змінних, які перетворюють функцію в 1. Це 0, 2, 3, 5, 6 і 7-й рядки. Виписують кон'юнкції, які відповідають вибраним рядкам. З огляду на це, якщо аргумент x_i входить до цього набору як 1, він записується у кон'юнкцію без змін; якщо ж x_i входить до цього набору як 0, то у відповідну кон'юнкцію записується його заперечення. Наприклад, для сегмента «a» в ДДНФ він буде мати вигляд такого алгебраїчного виразу:

$a_{ДДНФ}=$

$=\overline{x2}\ \overline{x1}\ \overline{x0} + \overline{x2}\ x1\ \overline{x0} + \overline{x2}\ x1x0 + x2\ \overline{x1}\ x0 + x2\ x1\ \overline{x0} + x2x1x0.$

Цю ж функцію для сегмента «a» можна записати у вигляді ДКНФ. Для цього із таблиці істинності вибирають набори аргументів, на яких значення функції дорівнює 0. Виписують диз'юнкції, які відповідають названим наборам аргументів. Зважаючи на це, якщо аргумент x_i входить у цей набір як 0, він вписується у диз'юнкцію, яка відповідає набору, без змін; якщо ж x_i входить у набір як 1, то у відповідну диз'юнкцію вписують його заперечення. ДКНФ буде мати вигляд такого алгебраїчного виразу: $a_{ДКНФ}= (\overline{x2} + x1 + x0) \wedge (x2 + x1 + \overline{x0})$.

Таблиця 3 – Задання ФАЛ за допомогою таблиці істинності для сегментів ССІ 3х8

Номер рядка	Значення функцій ССІ 3х8							Значення аргументів			Примітка
	a	b	c	d	e	f	g	x_2	x_1	x_0	
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	
1	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	
2	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	
3	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	
4	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	
5	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	
6	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	
7	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	

Очевидно, що для реалізації ***а*** ($а_{ДКНФ}$) раціональніше вибрати схему в програмі, яка буде містити тільки три блоки з трьома входами (аналог F11, F13) та з двома входами F14 з таблиці 2.

За аналогічною процедурою студентам необхідно скласти ДДНФ та ДКНФ для решти сегментів таблиці 3 і вибрати, яка функція є більш раціональною для програмування. Збільшення кількості входів у блоків OR, AND та інших реалізується шляхом «розтягування» блока ПКМ за нижню межу блока.

За відомою з ЛР 1 процедурою створюються додаткові змінні та кнопки шини «x2», «x1», «x0». Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передача/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуску (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад Scr_laba02) та після натискання F7 або увімкнувши режим «Enable Variable Modification» іншим способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин x2x1x0, 001, 011, 010, 000, 100, 101, 111, 110 та перевіряти правильність зображень цифр згідно з таблицею 3. У разі невідповідності провести коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку.

Контрольні питання

1 Яка концепція функціональних блоків FBD та ST?

2 Яким чином реалізовуються зв'язки між блоками на FBD та ST?

3 Яке призначення входів і виходів функціональних блоків?

4 Як розташовуються функціональні блоки в редакторі програми FBD, а також мовою ST?

5 Яке визначення булевої функції?

6 Як тлумачиться термін ДНФ?

7 Як тлумачиться термін КНФ?

8 Як тлумачиться термін ДДНФ?

9 Як тлумачиться термін ДКНФ?

10 Як реалізувати з'єднання з ПЛК?

11 Як реалізувати передачі/завантаження (Load) схеми-програми до ПЛК?

12 Як реалізувати запуск (Run) програми у віртуальному ПЛК?

13 Як скласти таблицю істинності для ССІ 4х10, 4х16(F)?

14 Як скласти таблицю істинності для ССІ 5х32, 6х46?

15 Яка потенційна межа відображення символів на ССІ?

Індивідуальне завдання

Скласти таблицю істинності, зробити для ССІ 4х16(F) аналітичний запис і схему-програму за варіантом:

1) сегмента "a" у досконалій ДНФ;
2) сегмента "a" у досконалій КНФ;
3) сегмента "b" у досконалій ДНФ;
4) сегмента "b" у досконалій КНФ;
5) сегмента "c" у досконалій ДНФ;
6) сегмента "c" у досконалій КНФ;
7) сегмента "d" у досконалій ДНФ;
8) сегмента "d" у досконалій КНФ;
9) сегмента "e" у досконалій ДНФ;
10) сегмента "e" у досконалій КНФ;
11) сегмента "f" у досконалій ДНФ;
12) сегмента "f" у досконалій КНФ.

ЛАБОРАТОРНА РОБОТА 3
Створення схем вмикання для синтезу ССІ 4х10 на комутаторах MUX

Мета роботи: отримання практичних навичок синтезу комбінаційних схем, розроблення схем-програм мовою ST в Unity Pro та налагодження елементарних функцій алгебри логіки на комутаторах **MUX**.

Обладнання та ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за наведеною після теоретичного матеріалу послідовністю та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Ввести програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

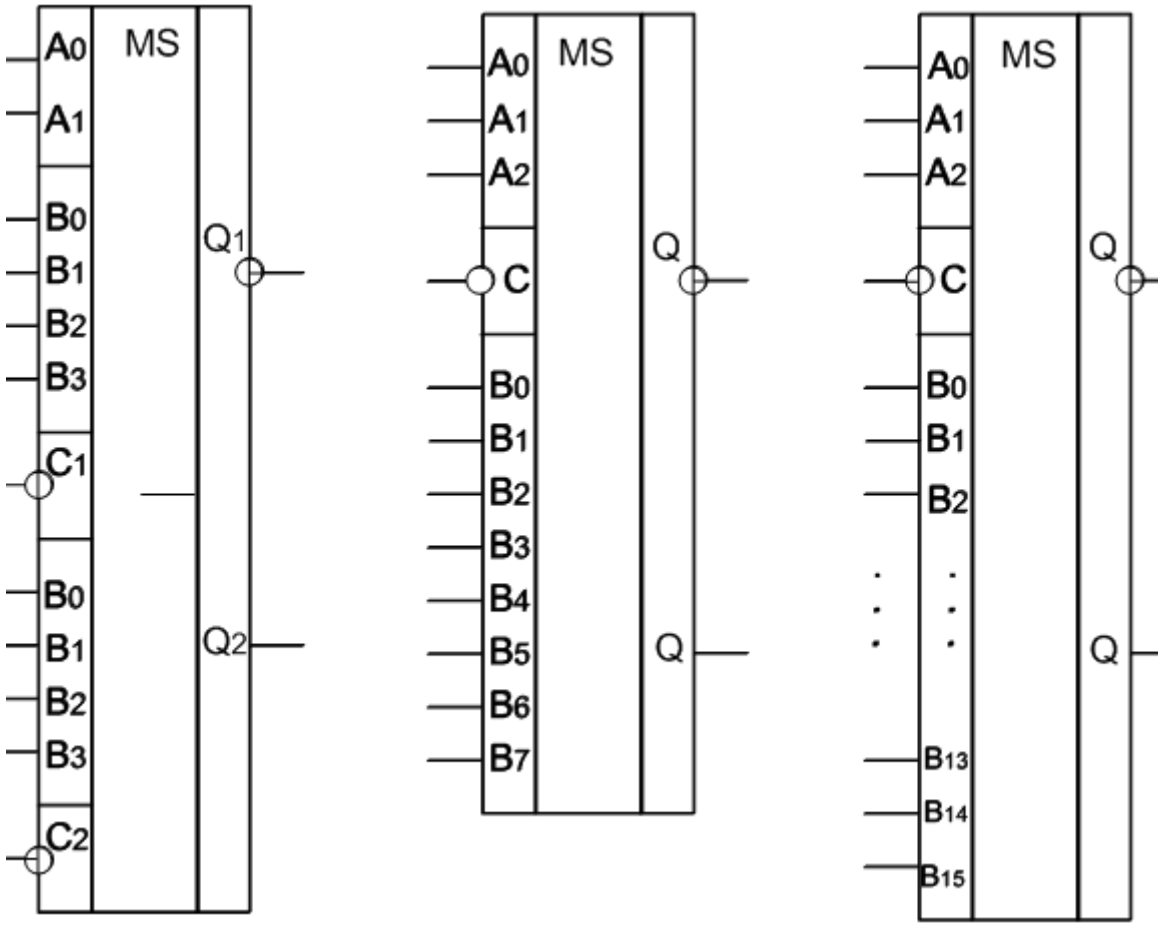
Теоретичний матеріал

У складі різних серій мікросхем, що застосовуються у пристроях залізничної автоматики та телемеханіки, є елементи середнього ступеня інтеграції – комутатори (мультиплексори – MUX). Мультиплексор – це схема з одним виходом Q, k+1 керуючими входами $A_0, A_1, \ldots, A_к$, n+1 інформаційними входами $B_0, B_1, \ldots, B_n$ та входом дозволу роботи мультиплексора C для подачі синхронізуючого сигналу. Якщо подавати на керуючі входи відповідні сигнали у вигляді двійкового коду, до виходу комутатора підключається один з його інформаційних входів. Є комутатори, які здійснюють вибір одного з 4, 8 або 16 інформаційних сигналів. Умовне позначення таких комутаторів наведено на рисунку 17. Якщо немає сигналу на вхід C, вихідна змінна мультиплексора Q повторює змінну інформаційного входу B_n з номером $n=2^{k+1}$, що задається двійковим кодом на керуючих входах. Якщо немає синхронізуючого сигналу (C=0, якщо вхід прямий), то немає й зв'язку між інформаційними входами і виходом.

Функціонування мультиплексора визначається за таблицею 4.

Таблиця 4 – Функціонування мультиплексора

Керуючі входи		Вхід дозволу С	Вихід
A_1	A_0	С	Q
~	~	0	0
0	0	1	B_0
0	1	1	B_1
1	0	1	B_2
1	1	1	B_3



а б в

а – К155КП2; б – К155КП7; в – К155КП1

Рисунок 17 – Типи комутаторів серії К155

Рівняння комутаторів 1 з 4, 1 з 8, 1 з 16 можна записати таким чином:

$$F_{1\text{-}4} = \overline{A_1}\,\overline{A_0}B_0 + \overline{A_1}\,A_0B_1 + A_1\overline{A_0}B_2 + A_1A_0B_3; \qquad (1)$$

$$F_{1\text{-}8} = \overline{A_2}\,\overline{A_1}\,\overline{A_0}B_0 + \overline{A_2}\,\overline{A_1}\,A_0B_1 + \overline{A_2}\,A_1\,\overline{A_0}B_2 + \\ + \overline{A_2}\,A_1\,A_0B_3 + A_2\,\overline{A_1}\,\overline{A_0}B_4 + A_2\,\overline{A_1}\,A_0B_5 + \\ + A_2\,A_1\,\overline{A_0}B_6 + A_2\,A_1\,A_0B_7; \qquad (2)$$

$$F_{1\text{-}16} = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}B_0 + \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,A_0B_1 + \overline{A_3}\,\overline{A_2}\,A_1\,\overline{A_0}B_2 + \\ + \overline{A_3}\,\overline{A_2}\,A_1\,A_0B_3 + \overline{A_3}\,A_2\,\overline{A_1}\,\overline{A_0}B_4 + \overline{A_3}\,A_2\,\overline{A_1}\,A_0B_5 + \\ + \overline{A_3}\,A_2\,A_1\,\overline{A_0}B_6 + \overline{A_3}\,A_2\,A_1\,A_0B_7 + A_3\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}B_8 + \\ + A_3\,\overline{A_2}\,\overline{A_1}\,A_0B_9 + A_3\,\overline{A_2}\,A_1\,\overline{A_0}B_{10} + A_3\,\overline{A_2}\,A_1\,A_0B_{11} + \\ + A_3\,A_2\,\overline{A_1}\,\overline{A_0}B_{12} + A_3\,A_2\,\overline{A_1}\,A_0B_{13} + A_3\,A_2\,A_1\,\overline{A_0}B_{14} + \\ + A_3\,A_2\,A_1\,A_0B_{15}. \qquad (3)$$

У виразах (1) – (3) через B_i й A_j позначені відповідно сигнали, які подаються на інформаційні та керуючі входи мультиплексорів.

Нехай ми маємо довільні ФАЛ 3, 4 і 5 змінних, тоді

$$F_3(x_3, x_2, x_1); \\ F_4(x_4, x_3, x_2, x_1); \qquad (4) \\ F_5(x_5, x_4, x_3, x_2, x_1).$$

Позначимо x_5, x_4, x_3, x_2 відповідно через A_3, A_2, A_1, A_0, а x_1 – через B.

$$F_3(A_1, A_0, B); \\ F_4(A_2, A_1, A_0, B); \qquad (5) \\ F_5(A_3, A_2, A_1, A_0, B).$$

Використовуючи метод функціональної декомпозиції, подамо функції (5) у вигляді таких наборів:

$$F_3(A_1, A_0, B) = F_3(0, 0, B)\overline{A_1}\,\overline{A_0} + F_3(0, 1, B)\overline{A_1}\,A_0 + \\ + F_3(1, 0, B)A_1\overline{A_0} + F_3(1, 1, B)A_1A_0; \qquad (6)$$

$$F_4(A_2, A_1, A_0, \hat{A}) = F_4(0,0,0,\hat{A})\overline{A_2}\ \overline{A_1}\ \overline{A_0} + F_4(0,0,1,\hat{A}) \wedge$$
$$\wedge \overline{A_2}\ \overline{A_1}\ A_0 + F_4(0,1,0,\hat{A})\overline{A_2}\ A_1\overline{A_0} + F_4(0,1,1,\hat{A})\overline{A_2}\ A_1A_0 +$$
$$+F_4(1,0,0,\hat{A})A_2\ \overline{A_1}\ \overline{A_0} + F_4(1,0,1,\hat{A})A_2\ \overline{A_1}\ A_0 + F_4(1,1,0,\hat{A}) \wedge$$
$$\wedge A_2\ A_1\overline{A_0} + F_4(1,1,1,\hat{A})A_2\ A_1A_0; \qquad (7)$$

$$F_5(A_3, A_2, A_1, A_0, B) = F_5(0,0,0,0,B)\overline{A_3}\ \overline{A_2}\ \overline{A_1}\ \overline{A_0} +$$
$$+F_5(0,0,0,1,B)\overline{A_3}\ \overline{A_2}\ \overline{A_1}\ A_0 + F_5(0,0,1,0,B)\overline{A_3}\ \overline{A_2}\ A_1\overline{A_0} +$$
$$+F_5(0,0,1,1,B)\overline{A_3}\ \overline{A_2}\ A_1A_0 + F_5(0,1,0,0,B)\overline{A_3}\ A_2\ \overline{A_1}\ \overline{A_0} +$$
$$+F_5(0,1,0,1,B)\overline{A_3}\ A_2\ \overline{A_1}\ A_0 + F_5(0,1,1,0,B)\overline{A_3}\ A_2\ A_1\overline{A_0} +$$
$$+F_5(0,1,1,1,B)\overline{A_3}\ A_2\ A_1A_0 + F_5(1,0,0,0,B)A_3\ \overline{A_2}\ \overline{A_1}\ \overline{A_0} +$$
$$+F_5(1,0,0,1,B)A_3\ \overline{A_2}\ \overline{A_1}\ A_0 + F_5(1,0,1,0,B)A_3\ \overline{A_2}\ A_1\overline{A_0} +$$
$$+F_5(1,0,1,1,B)A_3\ \overline{A_2}\ A_1A_0 + F_5(1,1,0,0,B)A_3\ A_2\ \overline{A_1}\ \overline{A_0} +$$
$$+F_5(1,1,0,1,B)A_3\ A_2\ \overline{A_1}\ A_0 + F_5(1,1,1,0,B)A_3\ A_2\ A_1\overline{A_0} +$$
$$+F_5(1,1,1,1,B)A_3\ A_2\ A_1A_0. \qquad (8)$$

Розглядаючи вирази (6) – (8), легко переконатися у тому, що всі значення функцій F_3, F_4, F_5, які містяться у правій частині цих виразів, можуть набувати значення тільки з множини {0, 1, B }.

Вирази (6) – (8) описують будь-яку ФАЛ відповідно 3, 4 і 5 змінних.

Попарне зіставлення виразів (1) і (6), (2) і (7), (3) і (8) показує, що коли на керуючі входи комутаторів подавати змінні $A_1\ A_0$; $A_2\ A_1\ A_0$; $A_3\ A_2\ A_1\ A_0$, а на відповідні інформаційні входи — величини з множини {0, 1, B або $\overline{B}$} за таблицею істинності заданої функції, то можна записати такі рівності:

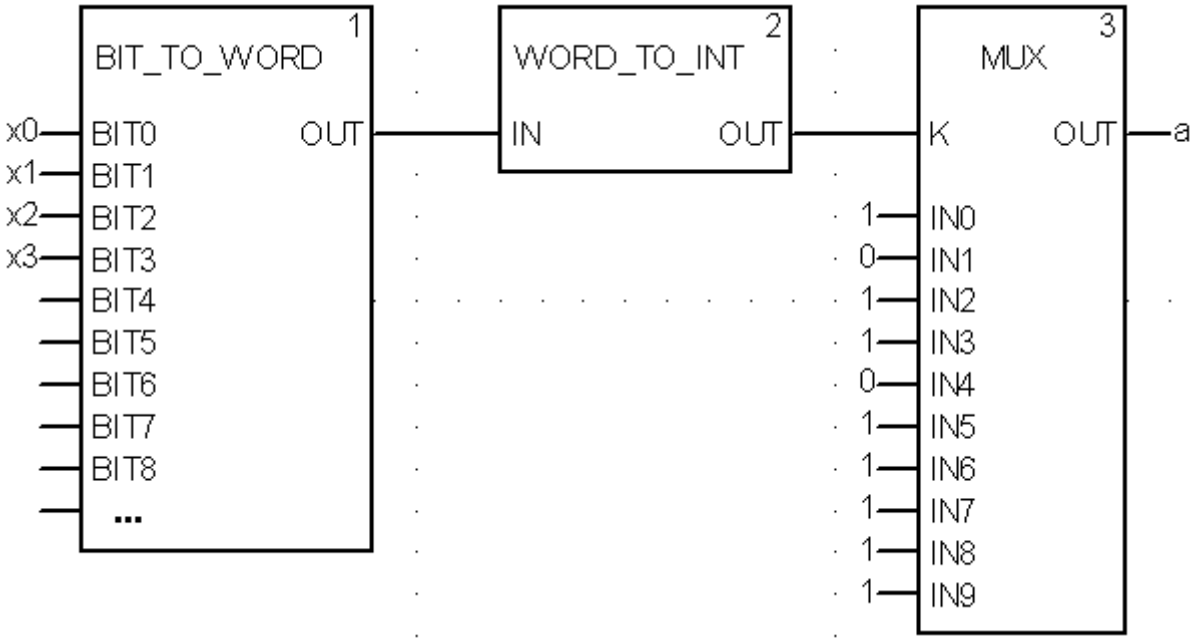
$$F_{1\text{-}4} = F_3; \quad F_{1\text{-}8} = F_4; \quad F_{1\text{-}16} = F_5. \qquad (9)$$

Ці вирази свідчать про те, що комутатори 1 з 4, 1 з 8, 1 з 16 є універсальними логічними елементами, що реалізують будь-яку ФАЛ відповідно 3, 4 і 5 змінних [8].

Але у нашому випадку синтез комбінаційних схем реалізується програмою мовою ST для ПЗ Unity Pro 3.0 або вище з широким різноманіттям функцій, процедур і

функціональних блоків (ST). У **ПЗ Unity Pro** є бібліотечні програмні блоки (EF-модуль/блок) FFB типу, які називаються MUX і моделюють роботу мультиплексора 1 з N (де N – від 2 до 16) з вибором (комутацією) каналу через змінну типу INT. Отже, для перетворення керуючих булевих (BOOL) сигналів A_3 A_2 A_1 A_0 (x3, x2, x1, x0) в тип INT для входу K (керуючого) EF-блока MUX (комутатора) необхідно перекодувати змінні. Оскільки в бібліотеці **Unity Pro** немає EF-блока BIT_TO_INT, то його потрібно або створити самостійно, або замінити двома послідовно з'єднаними BIT_TO_WORD і WORD_TO_INT. На рисунку 18 наведено вигляд схеми/програми функції комутатора.

а



б

```
adr := WORD_TO_INT(BIT_TO_WORD(X0,X1,X2,X3));
  (*                    0 1 2 3 4 5 6 7 8 9 A b c d E F     *)
        a := mux(adr,1,0,1,1,0,1,1,1,1,1,1,0,0,0,1,1);
```

Рисунок 18 – Схема комутатора 1 із 10, реалізована для увімкнення сегмента "a" індикатора мовою FBD (а) та мовою ST (б)

Синтез логічних пристроїв на базі мультиплексора (MUX) розглянемо на прикладі побудови схем увімкнення десяти цифр сегментами семисегментного індикатора з чотирма входами x3, x2, x1, x0.

Послідовність створення проекту

Для виконання ЛР 3 необхідно відкрити створений проект ЛР 1, зберегти його як laba03 або самостійно створити проект за процедурою, наведеною в ЛР 1, із назвою laba03.

Послідовність дій введення типового завдання проекту

Розробити програму мовою ST кодування стану десяти кнопок (булеві змінні k0, k1, k2, k3, k4, k5, k6, k7, k8, k9) у дворозрядний код (булеві змінні x3, x2, x1, x0 або A3, A2, A1, A0) та візуалізувати коди x3x2x1x0 (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001) на стандартному семисегментному індикаторі із сегментами (a, b, c, d, e, f, g) у вигляді цифр, зображення яких пропонується виводити в такому вигляді:  , відповідно до значень, показаних у таблиці істинності за аналогією з таблицею 3.

Синтез схем-програм ФАЛ на базі комутаторів, при якому ФАЛ має вигляд стандартної схеми-програми як на рисунку 18, є найбільш зручним. Однак є невелика відмінність. Входи IN0...IN9 при прямому кодуванні (до 16 входів) фактично є табличними значеннями булевої функції виходу (вихід OUT). Тому для побудови схеми ФАЛ потрібно скопіювати необхідну кількість (сім – від кількості сегментів) EF-блоків типу MUX для кожного із сегментів (a, b, c, d, e, f, g). Виходи (OUT) семи EF-блоків типу MUX приєднати до булевих змінних (a, b, c, d, e, f, g), а на входи IN0...IN9 подати 0 або 1 у суворій відповідності до таблиці істинності. Набір змінних x3x2x1x0 засобами двох послідовно з'єднаних блоків BIT_TO_WORD і WORD_TO_INT перетворюють у змінну Adr формату INT. Саме змінну Adr формату INT необхідно подати на вхід "K" семи EF-блоків типу MUX.

За відомою з ЛР 1 процедурою створюються додаткові змінні та кнопки шини «x3», «x2», «x1», «x0». Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передачі/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуску (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад Scr_laba03) та після натискання F7, або увімкнувши режим «Enable Variable Modification» іншим

способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин x3x2x1x0, 0001, 0011, 0010, 0000, 0100, 0101, 0111, 0110, 1000, 1001 та перевіряти правильність зображень цифр за таблицею істинності (створеною самостійно). У разі невідповідності провести коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку.

Контрольні питання

1 Яке визначення процедури та функції?

2 Яке визначення FFB блока та підпрограми?

3 Яке визначення EF функції?

4 Яке визначення FFB блока та функції в мові ST?

5 Як тлумачиться термін MUX?

6 Як тлумачиться термін DMX?

7 Як тлумачиться термін WORD?

8 Як тлумачиться термін INT?

9 Як реалізувати з'єднання з ПЛК?

10 Як реалізувати передачі/завантаження (Load) схеми-програми до ПЛК?

11 Як реалізувати запуск (Run) програми у віртуальному ПЛК?

12 Як скласти таблицю істинності для CCI 4x10, 4x16(F)?

13 Як скласти таблицю істинності для CCI 5x32, 6x46?

14 Яка потенційна межа відображення символів на CCI?

Індивідуальне завдання

Скласти таблицю істинності, зробити для CCI 4x16(F) аналітичний запис і схему-програму для реалізації на комутаторах MUX таких ФАЛ:

1) сегмента "a" у досконалій ДНФ;

2) сегмента "a" у досконалій КНФ;

3) сегмента "b" у досконалій ДНФ;

4) сегмента "b" у досконалій КНФ;

5) сегмента "c" у досконалій ДНФ;

6) сегмента "c" у досконалій КНФ;

7) сегмента "d" у досконалій ДНФ;

8) сегмента "d" у досконалій КНФ;

9) сегмента "e" у досконалій ДНФ;
10) сегмента "e" у досконалій КНФ;
11) сегмента "f" у досконалій ДНФ;
12) сегмента "f" у досконалій КНФ.

ЛАБОРАТОРНА РОБОТА 4
Створення функціональних блоків користувача (DFB) для синтезу модуля вмикання CCI 4x16(F)min

Мета роботи: отримання практичних навичок створення функціональних блоків користувача (DFB) та синтезу комбінаційних схем з розробленням програм мовою ST й використанням DFB-блоків в Unity Pro і їх налагодження.

Обладнання та ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за послідовністю, викладеною нижче, та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання і ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Увести програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

Послідовність створення проекту

Для виконання ЛР 4 необхідно відкрити створений проект ЛР 1, зберегти його як laba04 або самостійно створити проект за процедурою, наведеною у ЛР 1, із назвою laba04.

Послідовність дій введення типового завдання проекту

Типове завдання ЛР 3. Як і в попередній роботі, компонуємо всі необхідні апаратні модулі контролера й активуємо та налагоджуємо їх за схемою підключення. Наступний крок – відкриваємо вікно Derived FB Types, щоб побудувати новий DFB-тип (рисунок 19), на основі якого потім будуть створюватися екземпляри. Крім унікальної назви типу (за правилами створення ідентифікаторів, латинські букви без пробілів), необхідно створити локальні вхідні і вихідні змінні (рисунок 19) в DFB-типі та логіку «перетворення» вхідних змінних у вихідні, у разі імітації бітового меандра з періодом $T_{імп}$ = 1 с (D_MT-меандр часу, у якого $t_{імп}$= 0,5 с, $t_{пауз}$= 0,5 с) (рисунок 20). Мовою ST будуємо схему (рисунок 20), натискаючи на об'єкт Sec_MT два рази ПКМ. У процесі створення цих елементів буде відображатись знак 

, це означає, що елемент у процесі обробки. Після завершення створення треба скомпілювати програму, щоб не було помилок, за допомогою символу  .

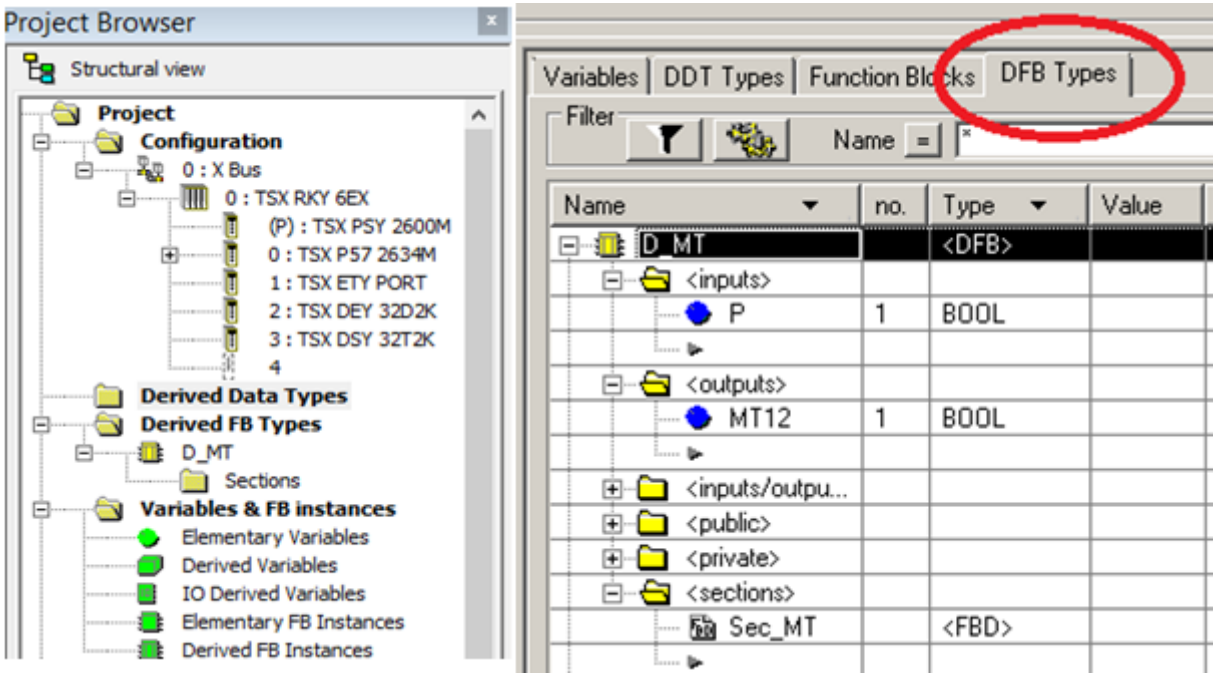


Рисунок 19 – Створення у розділі проекту Derived FB Types DFB-типу з назвою «D_MT»

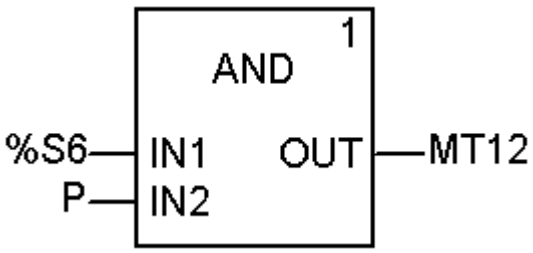


Рисунок 20 – Внутрішня схема (логіка) секції Sec_MT DFB-блока створюваного типу "D_MT"

У MAST секції (Laba04) викликаємо меню вибору (створення) блока Ctrl+I (рисунок 21) і створюємо екземпляр функціонального блока користувача ввівши в поле "FFB" значення «D_MT», а в поле «Instance/екземпляр» – назву екземпляра блока типу «D_MT» (можливо MT або MT1 за замовчуванням D_MT_0, D_MT_1, ...). Якщо на вхід «P» такого блока подати «сигнал 1», то на виході MT12 буде відтворюватися імпульсний сигнал – меандр.

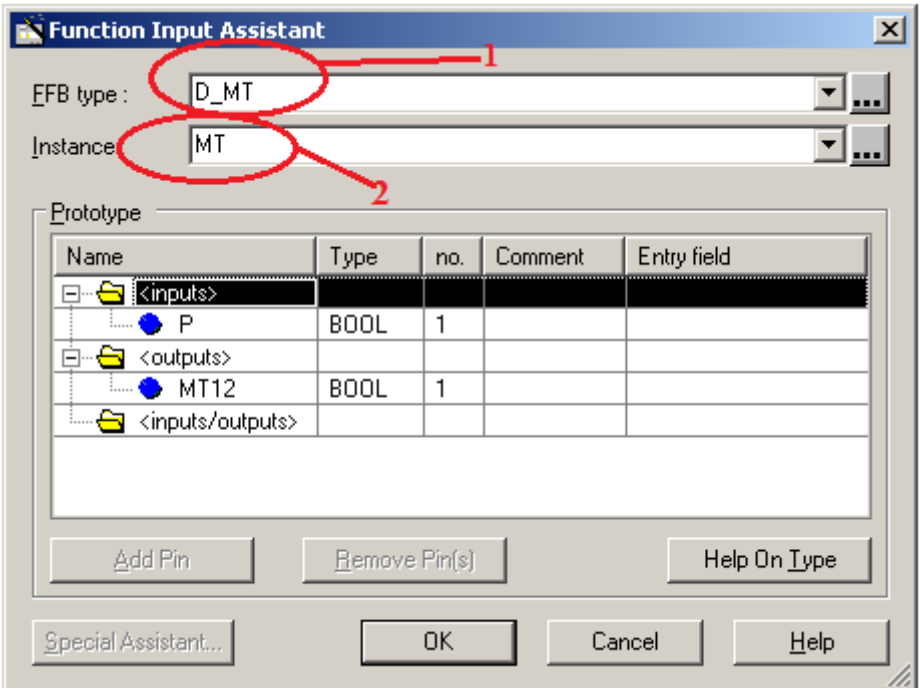


Рисунок 21 – Створення екземпляра з ім'ям «MT» функціонального блока користувача DFB-типу з ім'ям «D_MT» зі сформованими раніше вхідними і вихідними змінними та програмою перетворення змінних

Відкриваємо/створюємо операторський екран Screen_laba04 і зображуємо об'єкт типу «еліпс» як на рисунку 22, задіємо кружечок для відображення восьмого «dp» сегмента модифікованого CCI, який буде залежати від вихідної змінної типу BOOL з іменем «MT12» блока з іменем «MT» DFB-типу. Така змінна матиме параметр запису MT.MT12 із можливістю задавати її в полі Variable (рисунок 22).

Можемо використовувати вихідну змінну MT12 блока MT як елемент складної змінної MT.MT12 у параметрах анімації «еліпс» – восьмого сегмента CCI або інших секціях програм.

Послідовність дій розроблення залікового завдання лабораторної роботи

Після успішного виконання вищенаведеного типового завдання для розуміння створення функціональних блоків користувача (DFB-тип) на прикладі створення генератора меандрових імпульсів (із ім'ям "MT"), з періодом в одну секунду необхідно синтезувати модуль управління семисегментним індикатором (DFB-тип із ім'ям "I4xF"), який буде перетворювати двійковий чотирирозрядний код (вхідні булеві змінні x3, x2, x1, x0) на стандартному семисегментному індикаторі (рисунок 8, б) з виходами a, b, c, d, e, f, g.

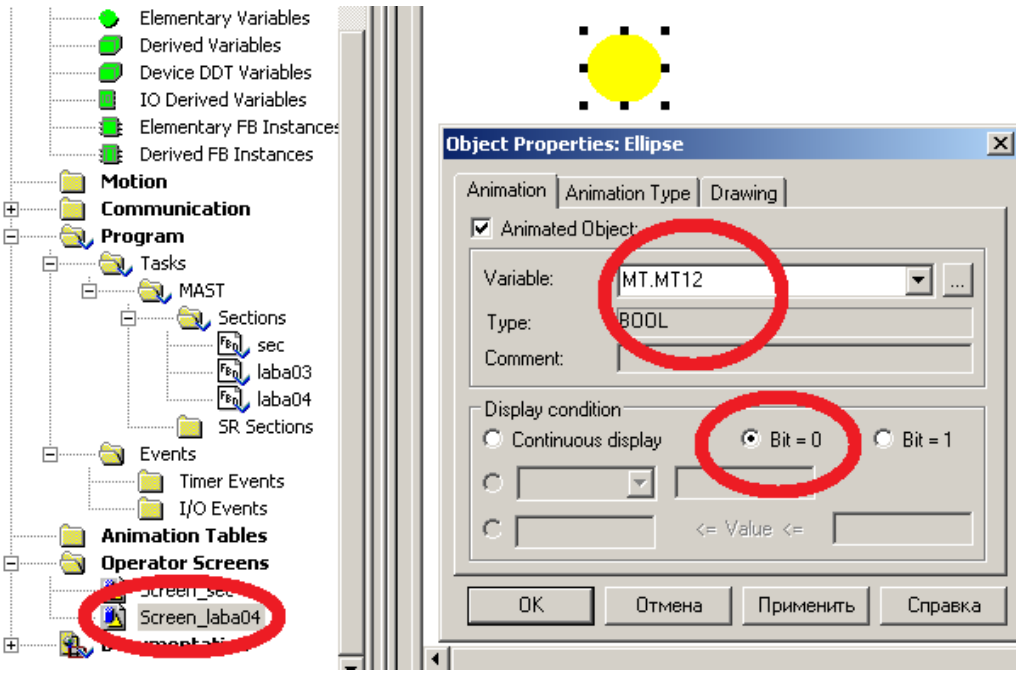
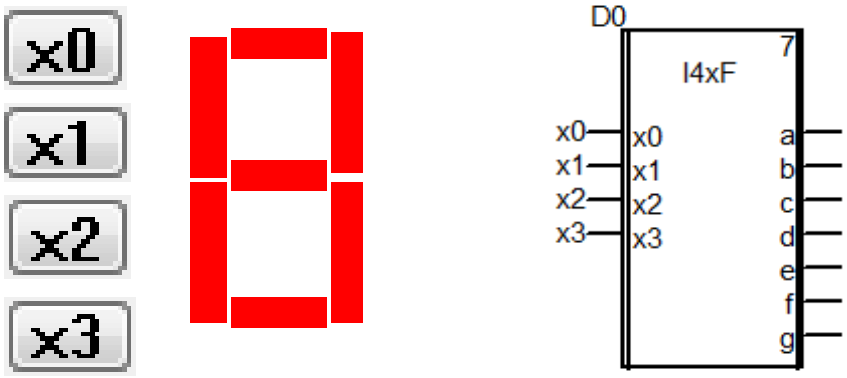


Рисунок 22 – Зображення і налаштування восьмого сегмента "dp" модернізованого CCI

Використавши створений функціональний блок користувача (DFB-тип із ім'ям "I4xF"), створити його екземпляр із ім'ям "D0" (Digital0) (рисунок 23). Підключити до вхідних булевих змінних x3, x2, x1, x0 блока D0 типу I4xF кодові булеві сигнали, що генерують таку послідовність: 0001, 0011, 0010, 0000, 0100, 0101, 0111, 0110, 1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100.

За відомою з ЛР 1 процедурою створюються додаткові змінні та кнопки шини «x3», «x2», «x1», «x0». Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передача/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуску (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад, Scr_laba04), після натискання F7, або увімкнувши режим «Enable Variable Modification» іншим способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин x3x2x1x0, 0001, 0011, 0010, 0000, 0100, 0101, 0111, 0110, 1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100 та перевірити правильність зображень цифр за таблицею істинності (створеною самостійно). У разі невідповідності провести коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку.

У результаті тестування мають з'явитися всі 16 цифр, зображення яких має виводитися у такому вигляді:  , відповідно до значень таблиці істинності, раніше самостійно сформованої (таблиця 5). Для отримання відмінної оцінки за виконання ЛР ФАЛ усіх сегментів мають бути оптимізовані.



а б

Рисунок 23 – Зображення фрагмента операторського екрана (а) та фрагмент секції схеми-програми завдання (б)

Для перевірки роботи розробленого модуля можливе застосування системної змінної типу «слово» (WORD), %SW50, яка містить інформацію про секунди реального часу ПЛК у форматі #16SS00 і блок WORD_TO_BIT.

Таблиця 5 – Форми подання чисел для перевірки

Форми подання чисел		
десяткова	двійкова	шістнадцяткова
1	**2**	**3**
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Схема перевірки зображена на рисунку 24.

Контрольні питання

1 Яка послідовність створення функціонального блока користувача (DFB)?

2 Як створити новий блок DFB-типу?

3 Як створити новий екземпляр блока?

4 Як мінімізувати ФАЛ?

5 Як раціонально виділити частину ФАЛ у блок користувача?

6 Як тлумачиться термін WORD?

7 Як тлумачиться термін INT?

8 Як реалізувати з'єднання з ПЛК?

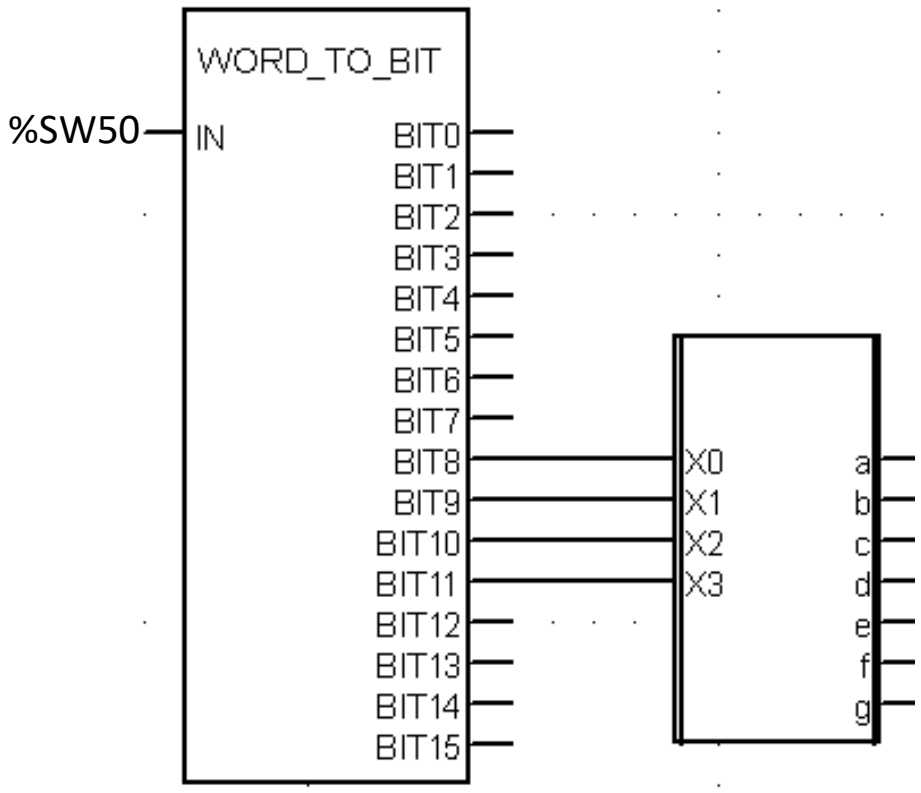
9 Як реалізувати передачі/завантаження (Load) схеми-програми до ПЛК?

10 Як скласти таблицю істинності для ССІ 4x10, 4x16(F)?

11 Як скласти таблицю істинності для ССІ 5x32, 6x46?

12 Яка потенційна межа відображення символів на ССІ?

а



б

```
WORD_TO_BIT (%SW50, x8,x8,x8,x8,x8,x8,x8,x8, x0, x1, x2,
                    x3,x4, x5, x6, x7);
D0 (x0, x1, x2, x3);
```

Рисунок 24 – Зображення фрагмента схеми-програми перетворення системних секунд ПЛК в сигнали ССІ мовою FBD (а) та мовою ST (б)

Індивідуальне завдання

Скласти таблицю істинності, зробити для ССІ 5х32 аналітичний запис і схему-програму для реалізації ФАЛ за варіантом:

1) сегмента "a" у досконалій ДНФ;
2) сегмента "a" у досконалій КНФ;
3) сегмента "b" у досконалій ДНФ;
4) сегмента "b" у досконалій КНФ;
5) сегмента "c" у досконалій ДНФ;
6) сегмента "c" у досконалій КНФ;
7) сегмента "d" у досконалій ДНФ;
8) сегмента "d" у досконалій КНФ;
9) сегмента "e" у досконалій ДНФ;
10) сегмента "e" у досконалій КНФ;
11) сегмента "f" у досконалій ДНФ;
12) сегмента "f" у досконалій КНФ.

ЛАБОРАТОРНА РОБОТА 5
Синтез комбінаційних схем вмикання декількох ССІ 4х16(F) для виведення перекодованих чисел різних систем числення

Мета роботи: отримання практичних навичок створення комбінаційних схем вмикання декількох ССІ 4х16(F) для виведення перекодованих чисел різних систем числення з використанням функціональних блоків користувача (DFB) та синтезу комбінаційних схем в Unity Pro, мінімізованих засобами карт Карно, налагодження мінімізованих ФАЛ.

Обладнання та ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за послідовністю, викладеною нижче, та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Ввести програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

Послідовність створення проекту

Для виконання ЛР 5 необхідно відкрити створений проект ЛР 1, зберегти його як laba05 або самостійно створити проект за процедурою, наведеною у ЛР 1, із назвою laba05.

Послідовність дій введення типового завдання проекту

Як і в попередній роботі, компонуємо всі необхідні апаратні модулі контролера та активуємо, налагоджуємо їх за схемою підключення. Наступний крок – відкриваємо створені блоки в попередній ЛР або вікно Derived FB Types, щоб створити новий DFB-тип (рисунок 19), на основі якого потім будуть створюватися необхідні екземпляри блоків користувача, щоб синтезувати модуль управління семисегментним індикатором (DFB-тип із ім'ям I4xF), який буде перетворювати двійковий чотирирозрядний код (вхідні булеві змінні x3, x2, x1, x0 всіх комбінацій) на стандартному семисегментному індикаторі (рисунок 8, б) з виходами a, b, c, d, e, f, g. Для перевірки роботи розробленого модуля можливе застосування системної змінної типу «слово» (WORD), %SW50, яка містить інформацію про секунди реального часу ПЛК у форматі #16SS00 і блок WORD_TO_BIT.

Послідовність дій для розроблення індивідуального завдання лабораторної роботи

Після успішного виконання вищенаведеного типового завдання для розуміння створення функціональних блоків

користувача (DFB-тип) необхідно створити схему-програму перетворення двійкового восьмирозрядного двійкового коду (вхідні булеві змінні x7, x6, x5, x4, x3, x2, x1, x0 будь-якої комбінації) у трирозрядний десятковий код та у дворозрядний шістнадцятковий код. Доречно для побудови схем перекодування використовувати описані раніше блоки перетворення типів даних (рисунок 18) і стандартні блоки: ділення (DIV), перетворення типів INT_TO_WORD, WORD_TO_BIT або інші, якщо буде потреба. Приклад використання таких типів наведено на рисунку 25.

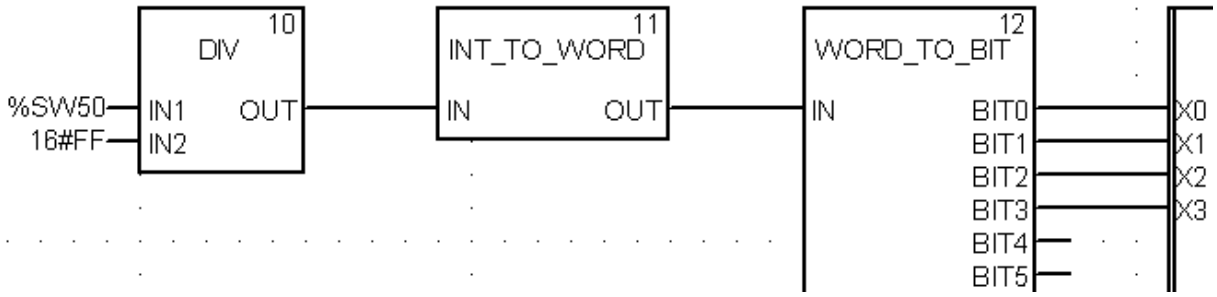


Рисунок 25 – Варіант перетворення системних секунд ПЛК у сигнали ССІ у вигляді фрагмента схеми-програми

Обов'язково використовувати розроблений раніше модуль управління семисегментним індикатором (DFB-тип із ім'ям I4xF), створюючи на базі нього будь-яку кількість екземплярів розробленого типу.

За відомою з ЛР 1 процедурою створюються додаткові змінні та кнопки шини «x7», «x6», «x5», «x4», «x3», «x2», «x1», «x0». На операторському екрані користувача подати (рисунок 26) двійковий код числа (зверху), його десятковий аналог (ліворуч червоними символами) і шістнадцятковий аналог (праворуч зеленими символами) засобами ССІ.

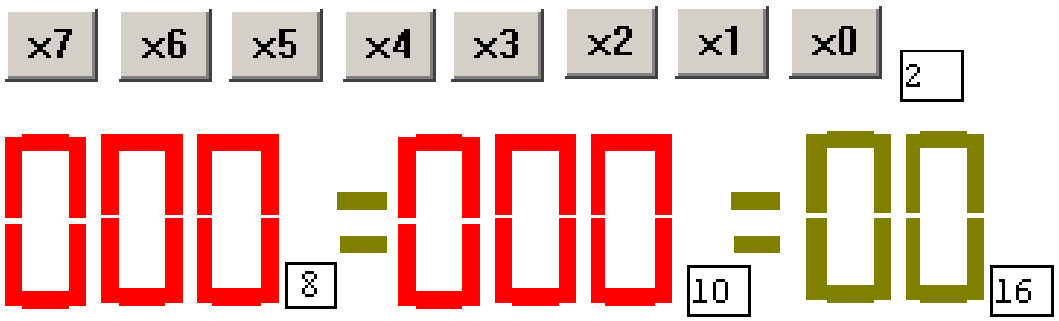
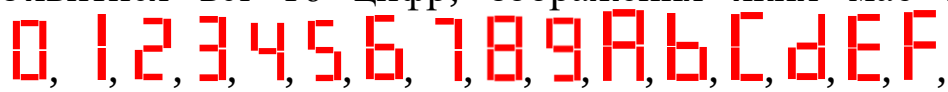


Рисунок 26 – Зображення на операторському екрані користувача двійкового коду числа (зверху), його вісімкового еквівалента (зліва червоними символами), десяткового аналога (посередині червоними символами) і шістнадцяткового аналога (зеленими символами (праворуч) засобами CCI

Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передачі/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуску (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад, Scr_laba04), після натискання F7 або увімкнувши режим «Enable Variable Modification» іншим способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин та перевіряти правильність зображень цифр за таблицею істинності (створеною самостійно). У разі невідповідності провести коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку.

У результаті тестування молодших чотирьох розрядів мають з'явитися всі 16 цифр, зображення яких має такий вигляд:  , відповідно до значень таблиці істинності, раніше самостійно сформованої, і таблиці 6.

Таблиця 6 – Форми подання восьмибітового числа тетрадами

Форми подання чисел		
десяткова	двійкова	шістнадцяткова
1	**2**	**3**
0	0000 0000	00
1	0000 0001	01
2	0000 0010	02
3	0000 0011	03
4	0000 0100	04
5	0000 0101	05
6	0000 0110	06
7	0000 0111	07
8	0000 1000	08
10	0000 1010	0A
15	0000 1111	0F
16	0001 0000	10
32	0010 0000	20
64	0100 0000	40
128	1000 0000	80
192	1100 0000	C0
202	1100 1010	CA
240	1111 0000	F0
255	1111 1111	FF

Контрольні питання

1 Яка послідовність створення функціонального блока користувача (DFB)?

2 Як створити новий блок DFB-типу?

3 Як створити новий екземпляр блока?

4 Як мінімізувати ФАЛ?

5 Як раціонально виділити частину ФАЛ у блок користувача?

6 Як додати до зображення (рисунок 26) вісімковий еквівалент числа?

ЛАБОРАТОРНА РОБОТА 6
Синтез дискретних автоматів із пам'яттю засобами RS-тригерів

Мета роботи: отримання практичних навичок створення дискретних автоматів із пам'яттю засобами **RS-тригерів** в Unity Pro та налагодження функціонування цифрових автоматів.

Обладнання та ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за наведеною після теоретичного матеріалу послідовністю та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою ST).

5 Ввести схему-програму мовою ST для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

Теоретичний матеріал

В арифметичних і логічних пристроях для збереження інформації найчастіше використовують тригери – пристрої з двома стійкими станами по виходу, які містять елементарну запам’ятовувальну комірку (бістабільна схема – БС) і схему керування (СК) – дискретні автомати з пам'яттю (ДАП). Такі автомати характеризуються тим, що стан їх виходів залежить як від сигналів, які діють на їх входах у цей момент часу, так і від послідовності сигналів, що надійшли на входи автомата в попередні моменти часу. У загальному вигляді ДАП мають схему, показану на рисунку 27.

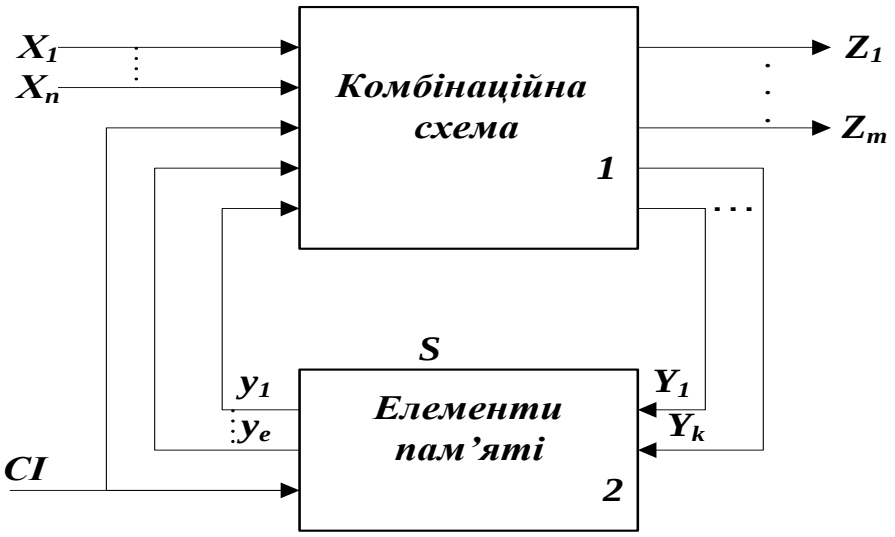


Рисунок 27 – Зображення схеми ДАП у загальному вигляді

Для опису ДАП необхідно оперувати двома різними функціями: φ та f. Функція φ (функція переходів) описує зміну вмісту пам'яті залежно від того, що в ній зберігалось і які відбувалися зміни вхідних сигналів. Функція f (функція виходів) описує зміну вихідних сигналів автомата під впливом вхідних сигналів залежно від того, що було записано в пам'яті ДАП.

ДАП складається з комбінаційної схеми (КС) 1 та елементів пам'яті (ЕП) 2. На входи КС надходять сигнали $X_1, \ldots, X_n$. Крім вихідних сигналів $Z_1, \ldots, Z_m$, КС1 формує сигнали $Y_1, \ldots, Y_k$, що здійснюють переключення ЕП2 в різні стани *S*. Вихідні сигнали ЕП2 $y_1, \ldots, y_e$ надходять на КС1 [8].

При реалізації ДАП як однобітових елементів пам'яті використовуються тригери різних типів. Найбільш поширеними на практиці є *RS*–, $\overline{R}\overline{S}$ –, *D*–, *T*–, *JK*–тригери. Умовні позначення, призначення входів, відповідні таблиці переходів і рівняння тригерів наведені в таблицях 7, 8.

Із таблиці 8 може бути отримана таблиця збудження тригерів (таблиця 7), що показує, які сигнали необхідно подавати на входи тригерів для забезпечення всіх можливих переходів тригерів. У таблиці 7 y – стани виходів тригерів у моменти часу *t* і *t+1;* Y_S , Y_R , $Y_{\bar{S}}$, $Y_{\bar{R}}$, Y_D , Y_T , Y_J , Y_K – вхідні сигнали відповідних тригерів; «~» означає 0 або 1.

Таблиця 7 – Таблиця збудження тригерів

y		RS		$\bar{R}\bar{S}$		D	T	JK	
t	t+1	Y_S	Y_R	$Y_{\bar{S}}$	$Y_{\bar{R}}$	Y_D	Y_T	Y_J	Y_K
0	0	0	~	1	~	0	0	0	~
0	1	1	0	0	1	1	1	1	~
1	0	0	1	1	0	0	1	~	1
1	1	~	0	~	1	1	0	~	0

У нашому випадку синтез автоматів із пам'яттю реалізується програмою мовою ST для ПЗ Unity Pro 3.0 або вище з широким різноманіттям функцій, процедур і функціональних блоків (FFB).

У ПЗ Unity Pro є бібліотечні програмні блоки (EF-модуль/блок) FFB - типу, що називаються "RS" (рисунок 28) і моделюють роботу RS-тригера із входами (типу BOOL) S та R1, які відповідно встановлюють вихід тригера (типу BOOL) в стан Q1=1 або Q1=0. При першому старті схеми-програми і відсутності значення "1" булевих змінних KeyS та KeyR стан виходу Q1 встановлюється на значення "0".

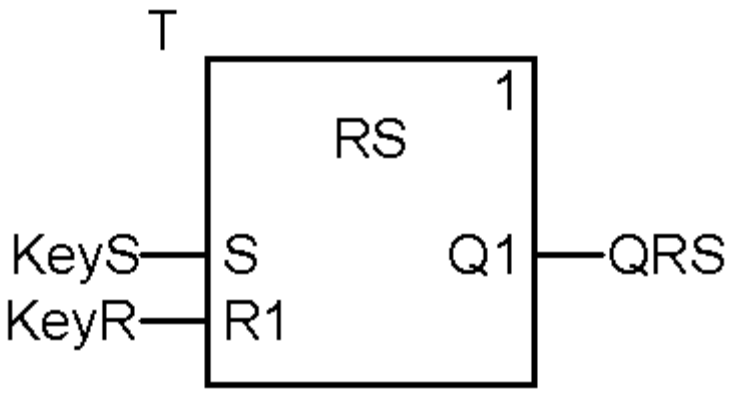
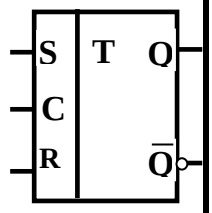
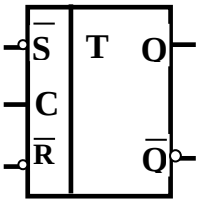
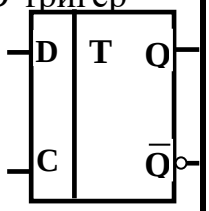
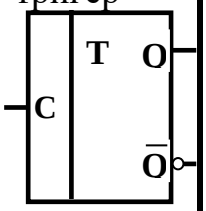
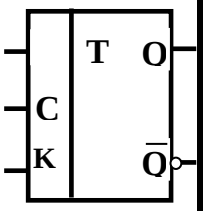


Рисунок 28 – Зображення схеми-програми підключення EF-модуля/блока RS-тригера

Таблиця 8 – Види функціонального призначення тригерів

<table>
<tr><th rowspan="3">Умовне позначення</th><th rowspan="3">Призначення входів</th><th colspan="5">Таблиця переходів*</th><th rowspan="3">Рівняння тригера</th></tr>
<tr><th>Такт t</th><th colspan="4">Такт (t+1)</th></tr>
<tr><th>y(t)=Q(t)</th><th colspan="4">y(t+1)=Q(t+1)</th></tr>
<tr><td rowspan="5">RS-тригер
 
</td><td rowspan="5">S - вхід установлення 1

R - вхід установлення 0

C-вхід синхронізації</td><td></td><td colspan="4">S(t)R(t)</td><td rowspan="5">y(t+1)=
=y(t) $\bar{R}$(t)+S(t)</td></tr>
<tr><td></td><td>00</td><td>01</td><td>11</td><td>10</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>–</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>–</td><td>1</td></tr>
<tr><td></td><td colspan="4">$\bar{S}$(t) $\bar{R}$(t)</td></tr>
<tr><td rowspan="4">$\bar{R}\bar{S}$ -тригер
 
</td><td></td><td>00</td><td>01</td><td>11</td><td>10</td></tr>
<tr><td>0</td><td>–</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>–</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td rowspan="4">D-тригер
 
</td><td rowspan="4">D - вхід установлення 0 або 1
C - вхід синхронізації</td><td></td><td colspan="2">D(t)</td><td rowspan="4">y(t+1)=D(t)</td></tr>
<tr><td></td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td rowspan="4">T-тригер
 
</td><td rowspan="4">C - лічильний вхід</td><td></td><td colspan="2">C(t)</td><td rowspan="4">y(t+1)=
=y(t)$\bar{N}$(t)+
+ $\bar{y}$(t)C(t)</td></tr>
<tr><td></td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td rowspan="5">JK-тригер
 
</td><td rowspan="5">J-вхід установлення 1
K-вхід установлення 0

C - вхід синхронізації</td><td></td><td colspan="4">J(t)K(t)</td><td rowspan="5">y(t+1)=
=y(t) $\bar{K}$(t)+
+ $\bar{y}$(t)J(t)</td></tr>
<tr><td></td><td>00</td><td>01</td><td>11</td><td>10</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td colspan="7">* знак “–” означає заборонений стан</td></tr>
</table>

Послідовність створення проекту

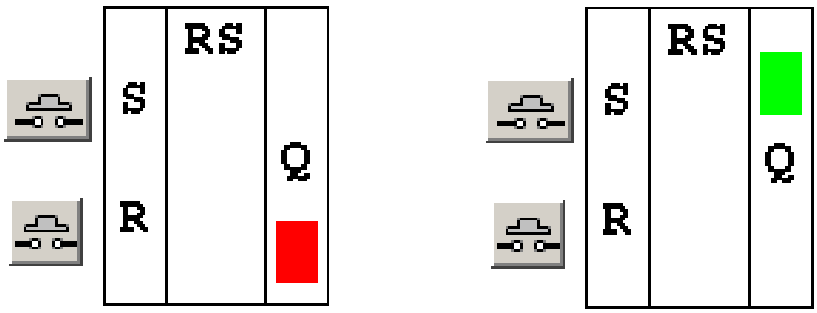
Для виконання ЛР 6 необхідно відкрити створений проект ЛР 1, зберегти його як laba06 або самостійно створити проект за процедурою, наведеною у ЛР 1, із назвою laba06.

Типове завдання ЛР 6. Розробити схему-програму мовою FBD або ST для моделювання роботи ДАП на базі RS-тригера відповідно до значень таблиці істинності з таблиць 7, 8.

Послідовність дій введення типового завдання проекту

Синтез схем-програм ДАП на базі RS-тригера відповідно до значень таблиці істинності з таблиць 7, 8, при якому ДАП має вигляд стандартної схеми-програми як на рисунку 28, є найбільш зручним. Однак є невелика відмінність. Входи S та R1, які відповідно встановлюють вихід тригера (типу BOOL) в стан Q1=1 або Q1=0, фактично є табличними значеннями входів S, R та Q.

На рисунку 29 наведено операторський екран користувача RS-тригера в початковому стані Q1=0 та в переключеному стані Q1=1.



а б

Рисунок 29 – Зображення на операторському екрані користувача RS-тригера в початковому стані Q1=0 (а) та у переключеному стані Q1=1 (б)

Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передача/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуск (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад Scr_laba06), після натискання F7, або увімкнувши режим «Enable Variable Modification» іншим способом – натиснувши ЛКМ на піктограмі , моделювати натискання кнопок S (змінна KeyS) та R (змінна

KeyR) і перевіряти правильність зображень стану виходу за таблицею істинності (таблиці 7, 8). У разі невідповідності провести коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку.

Контрольні питання

1 Яка послідовність створення ДАП на базі RS-тригера?

2 Як створити ДАП на базі RS-тригера?

3 Як створити новий екземпляр ДАП на базі RS-тригера?

4 Як мінімізувати роботу ФАЛ?

5 Як раціонально виділити частину ФАЛ у блок користувача?

6 Якого типу вхідні змінні RS-тригера?

7 Якого типу вихідні змінні RS-тригера?

8 Чому кнопки вхідних змінних RS-тригера без фіксації?

Індивідуальне завдання

За варіантом студента скласти таблицю істинності, створити схему-програму та зображення на операторському екрані для:

1) $\overline{R}\overline{S}$ – тригера;
2) D – тригера;
3) T – тригера;
4) JK – тригера;
5) $\overline{R}\overline{S}$ – тригера;
6) D – тригера;
7) T – тригера;
8) JK – тригера;
9) $\overline{R}\overline{S}$ – тригера;
10) D – тригера;
11) T – тригера;
12) JK – тригера.

ЛАБОРАТОРНА РОБОТА 7
Синтез лічильників імпульсів із довільним модулем засобами мови ST в Unity Pro

Мета роботи: отримання практичних навичок проектування/створення лічильників імпульсів із довільним модулем, розроблення програм мовою **ST** з використанням DFB блоків в Unity Pro та налагодження проектів.

Обладнання та ПЗ: цифрова ПЕОМ із СПЗ Windows XP або вище та прикладне ПЗ Unity Pro 3.0 або вище.

Хід виконання роботи

1 Вивчити весь теоретичний матеріал, необхідний для досягнення цілей ЛР.

2 Створити проект за послідовністю, викладеною нижче, та ввести схему-програму типового завдання.

3 Визначити та вирішити індивідуальне завдання.

4 Оформити «заготовку» до ЛР (номер, назва, мета, обладнання та ПЗ, принципова схема підключення контактів (виводів) контролера (уявного), схема-програма вирішення індивідуального завдання для Unity Pro мовою **ST** або FBD).

5 Увести програму мовою **ST** для Unity Pro в лабораторії, отримати результати та захистити у викладача.

6 Оформити звіт з висновками й отримати бали за ЛР у викладача.

Типове завдання: побудувати синхронний лічильник зворотної лічби на три розряди.

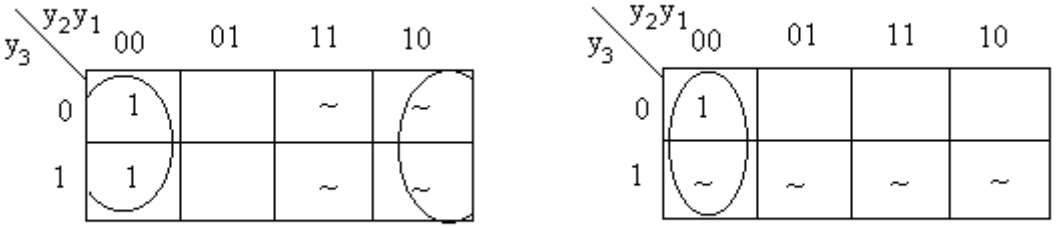
Для проектування заданого лічильника використовуємо ті самі методи, що й для проектування звичайних двійкових лічильників (етапи синтезу показано на рисунку 30).

Для синтезу лічильників як ДАП схеми реалізуються програмною мовою **ST** для ПЗ Unity Pro 3.0 або вище з широким різноманіттям функцій, процедур і функціональних блоків (FFB).

t			t+1			Y_{J3}	Y_{K3}	Y_{J2}	Y_{K2}	Y_{J1}	Y_{K1}
y_3	y_2	y_1	y_3	y_2	y_1						
0	0	0	1	1	1	1	~	1	~	1	~
0	0	1	0	0	0	0	~	0	~	~	1
0	1	0	0	0	1	0	~	~	1	1	~
0	1	1	0	1	0	0	~	~	0	~	1
1	0	0	0	1	1	~	1	1	~	1	~
1	0	1	1	0	0	~	0	0	~	~	1
1	1	0	1	0	1	~	0	~	1	1	~
1	1	1	1	1	0	~	0	~	0	~	1

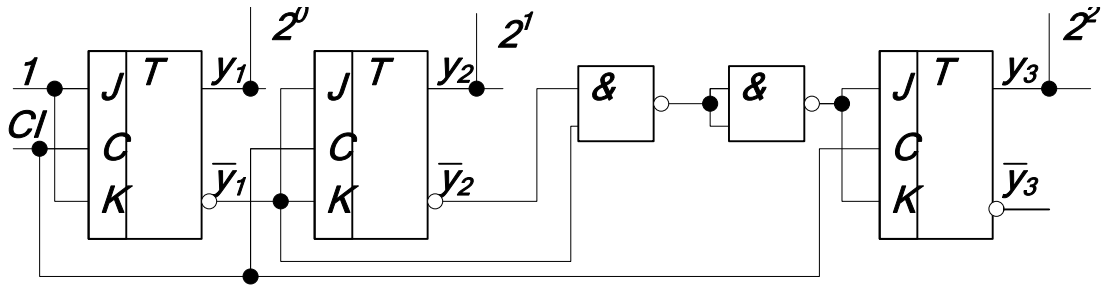
$$Y_{J1}=Y_{K1}=1$$

а



$$Y_{J2}=Y_{K2}=\bar{y}_1 \qquad Y_{J3}=Y_{K3}=\bar{o}_2\bar{y}_1$$

б



в

а – таблиця істинності; б – карта Карно з виділеними підкубами; в – схема

Рисунок 30 – Етапи синтезу лічильника

У ПЗ Unity Pro є бібліотечні програмні блоки (EF-модуль/блок) FFB-типу. Один з них називається «TP» і моделює запуск за переднім фронтом на вході IN. При цьому поточне значення ET змінюється з 0 ms до значення уставки «PT». Вихідне значення таймера переходить у TRUE, коли таймер

запускається, та інвертується, коли поточне значення досягне значення PT. Фактично таймер витримує імпульс на виході заданої тривалості, якщо на вході IN одразу після увімкнення з'явиться "0". Переведення входу IN в "1" одразу перезапустить таймер. Зображення блока таймера типу TP та його діаграма роботи наведені на рисунку 31.

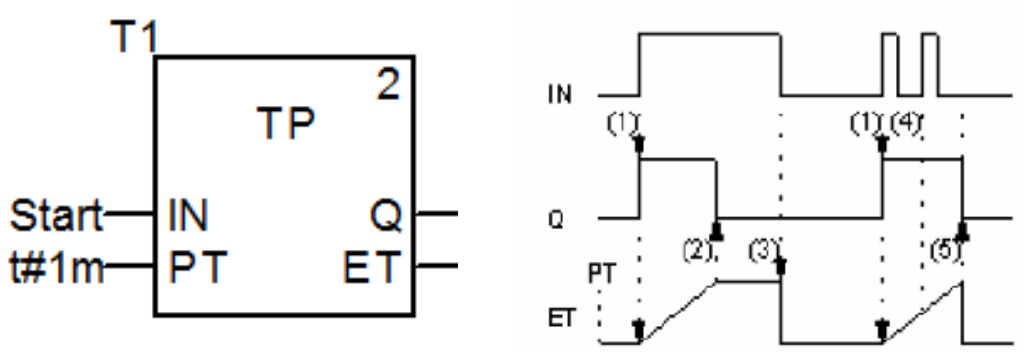
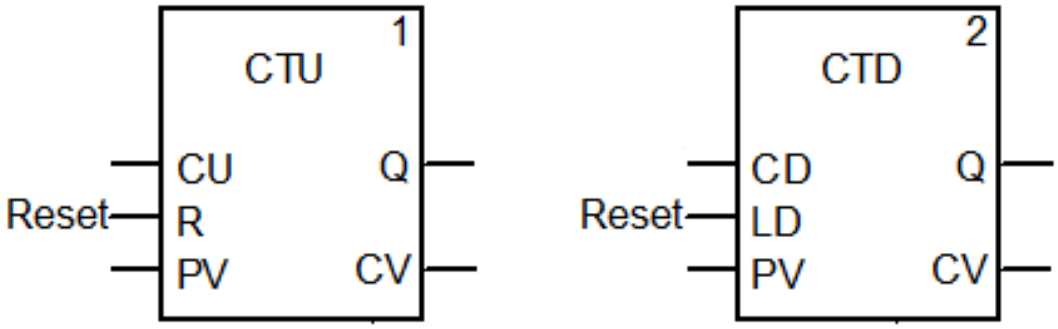


Рисунок 31 – Зображення блока таймера типу TP
та його діаграма роботи

Оскільки в бібліотеці **Unity Pro** є три різновиди лічильників (CTU, CTD, CTUD), розглянемо їх по черзі. Лічильник CTU (прямий) за переднім фронтом сигналу на вході CU збільшує поточне значення CV на 1 (рисунок 32, а). На вході PV задається уставка. При досягненні поточного значення CV ≥ PV виходу Q присвоюється TRUE. Подача на вхід блока сигналу R=TRUE скидає поточне значення в нуль ("0").



а б

а – CTU (прямий); б – CTD (зворотний)
Рисунок 32 – Зображення лічильників за переднім фронтом

Лічильник CTD (зворотний) за переднім фронтом сигналу на вході CD зменшує на 1 поточне значення CV (рисунок 32, б). На вході PV задається уставка. При досягненні поточного значення $CV \leq 0$ виходу Q присвоюється TRUE. Подача на вхід блока сигналу LD=TRUE реалізує записування у значення уставки поточного значення (CV:=PV) [2].

Лічильник CTDU (реверсивний) об'єднує у собі функції обох описаних раніше лічильників CTU та CTD. Входи CU та CD призначені відповідно для збільшення і зменшення плинного значення лічильника CV на 1. На вході PV задається уставка. Решта компонентів лічильника аналогічна описаним раніше.

Засобами **Unity Pro** із застосуванням двох різних блоків CTD, INT_TO_WORD реалізовано схему зворотної лічби за переднім фронтом імпульсів змінної «Imp» від 7 із записом до змінної «K» типу «WORD» (рисунок 33).

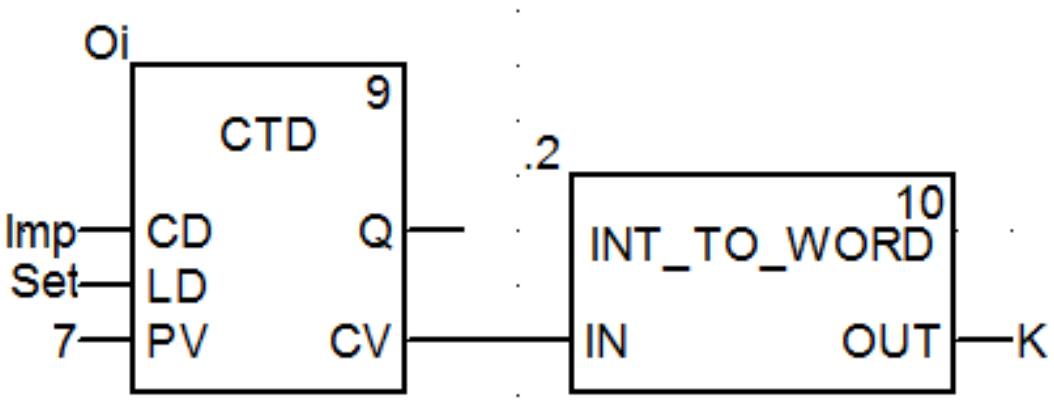


Рисунок 33 – Зображення схеми зворотної лічби
за переднім фронтом від 7 на зменшення

За відомою із ЛР 1 процедурою створюються додаткові змінні та кнопки шини «Imp», «Set». Реалізовані в попередніх ЛР семисегментні індикатори можливо задіяти для індикації змінної «K» (здійснивши відповідні перетворення типів). Наступним кроком виконання роботи є повторне з'єднання з ПЛК (Ctrl+K), передача/завантаження (Load) схеми-програми до ПЛК (Ctrl+L) і запуск (Run) її у віртуальному ПЛК (Ctrl+R). Вибрати екран користувача (наприклад Scr_laba07), після натискання F7, або увімкнувши режим «Enable Variable Modification» іншим

способом – натиснувши ЛКМ на піктограмі  , моделювати комбінації шин «Imp» та «Set» і перевіряти правильність зображень цифр за таблицею істинності (створеною самостійно). У разі невідповідності виконати коригування схеми-програми або налаштування об’єктів екранів оператора та зробити повторну перевірку. Приблизне зображення екрана користувача наведено на рисунку 34. Після натискання кнопки 

на індикаторі має з’явитися  . А з кожним натисканням кнопки 

операторського екрана на стандартному семисегментному індикаторі із сегментами (a, b, c, d, e, f, g) мають з’явитися зображення у вигляді цифр, які пропонується виводити в такому вигляді:  , відповідно до значень, показаних у таблиці істинності, самостійно створеній, якщо була потреба. Створення змінних і кнопок шини «x3», «x2», «x1», «x0» та необхідність їх візуалізації студент обирає на свій розсуд.

Рисунок 34 – Зображення екрана користувача для перевірки зворотного лічильника від 7 на зменшення

Для перевірки роботи розробленого модуля або розроблення більш складних завдань можливе застосування системної змінної типу «слово» (WORD) – %SW50, яка містить інформацію про секунди реального часу ПЛК у форматі #16SS00, блок WORD_TO_BIT й ін.

Подальші дії слід здійснювати шляхом перекладання мовою ST усіх програм технологічного спрямування [7].

Контрольні питання

1 Як створити лічильник прямої лічби засобами тригерів і логічних елементів?

2 Яка послідовність створення функціонального блока користувача (DFB)?

3 Як створити новий DFB-тип?

4 Як створити лічильник прямої лічби засобами **Unity Pro**?

5 Як мінімізувати ФАЛ?

6 Як раціонально виділити частину ФАЛ у блок користувача в **Unity Pro**?

7 Як тлумачиться термін WORD в **Unity Pro**?

8 Як тлумачиться термін INT в **Unity Pro**?

9 Як реалізувати з'єднання з ПЛК в **Unity Pro**?

10 Як створити лічильник зворотної лічби засобами **Unity Pro**?

11 Як створити лічильник прямої лічби засобами **Unity Pro**?

12 Як створити лічильник прямої лічби з довільною кількістю засобами **Unity Pro**?

Індивідуальне завдання

Скласти таблицю істинності та схему-програму мовою ST із зображенням на операторському екрані:

1) лічильника зворотної лічби засобами **Unity Pro** з модулем 12 (від 12 до 0 по колу);

2) лічильника прямої лічби засобами **Unity Pro** з модулем 14 (від 0 до 14 по колу);

3) лічильника прямої лічби засобами **Unity Pro** з модулем 23 (від 0 до 23 по колу);

4) лічильника прямої лічби засобами **Unity Pro** з модулем 59 (від 0 до 59 по колу);

5) лічильника зворотної лічби засобами **Unity Pro** з модулем 6 (від 6 до 0 по колу);

6) лічильника зворотної лічби засобами **Unity Pro** з модулем 23 (від 23 до 0 по колу);

7) лічильника зворотної лічби засобами **Unity Pro** з модулем 17 (від 17 до 0 по колу);

8) лічильника прямої лічби засобами **Unity Pro** з модулем 19 (від 0 до 19 по колу);

9) лічильника прямої лічби засобами **Unity Pro** з модулем 30 (від 0 до 30 по колу);

10) лічильника прямої лічби засобами **Unity Pro** з модулем 31 (від 0 до 31 по колу);

11) лічильника зворотної лічби засобами **Unity Pro** з модулем 5 (від 5 до 0 по колу);

12) лічильника зворотної лічби засобами **Unity Pro** з модулем 28 (від 28 до 0 по колу);

Завдання для самостійної роботи студентів з дисципліни

1 Створити схему включення контролером виконавчих пристроїв збільшеного струму споживання/підвищеною напругою живлення/ промислової частоти живлення та одно/двонаправленою передачею контрольної/вимірювальної або діагностичної інформації.

2 Створити контролерну систему з двома контролерами та одно/двонаправленою передачею між ними булевих сигналів в однакових/різних фізичних величинах

3 Розробити схеми контролерних еквівалентів класичних електромагнітних пристроїв типу «нейтральне реле», «комбіноване реле», «поляризоване реле», «аварійне реле» або ін.

4 Спроектувати схему та програму використання контролерних систем для моделювання тризначного кодового автоматичного блокування (КАБ) з використанням однониткових/двониткових ламп прохідних світлофорів

5 Розробити схему та програму використання контролерних систем для моделювання чотиризначного кодового автоматичного блокування (КАБ) з використанням однониткових/двониткових ламп прохідних світлофорів

6 Створити схему та програму використання контролерних систем для діагностики чотиризначного кодового автоматичного

блокування (КАБ) з використанням однониткових/двониткових ламп прохідних світлофорів

7 Вирішити задачу програмного захисту від несанкціонованого доступу до контролерних систем з дистанційною діагностикою чотиризначного кодового автоматичного блокування (КАБ)

СПИСОК ЛІТЕРАТУРИ

1 Меркулов В. С., Бутенко В. М. Основи алгоритмізації базових обчислювальних процесів : навч. посіб. Харків : УкрДАЗТ, 2008. 163 с.

2 Пупена О. М., Ельперін І. В. Програмування промислових контролерів у середовищі Unity Pro : навч. посіб. Київ : Ліра-К, 2013. 376 с.

3 IEC 61131-3:2013 Programmable controllers Part 3: Programming languages.

4 Математичні методи та моделі в розрахунках на ЕОМ : навч. посіб. / М. І. Данько, В. С. Меркулов, В. О. Гончаров та ін.; за заг. ред. М. І. Данька. Харків : УкрДАЗТ, 2008. 172 с.

5 IEC 60617-12:1997 **Withdrawn Graphical symbols for diagrams - Part 12: Binary logic elements**

6 Булева алгебра з двома елементами. URL : https://uk.wikipedia.org/wiki/.

7 Бутенко В. М. Методичні вказівки до виконання лабораторних робіт з дисципліни «Автоматизація технологічних процесів» Schnider Electric. Харків : УкрДАЗТ, 2015. 54 с.

8 Загарій Г. І., Бушевська Л. В. Методичні вказівки з дисципліни «Електроніка та мікросхемотехніка». Розд. «Синтез комбінаційних схем». Харків :УкрДАЗТ, 2008. Ч. 2. 23 с.

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних та самостійних робіт

з дисципліни
«АРХІТЕКТУРА ТА ПРОГРАМУВАННЯ ПРОМИСЛОВИХ СИСТЕМ КЕРУВАННЯ»

Відповідальний за випуск Бутенко В. М.

Редактор Еткало О. О.

Підписано до друку 07.07.20 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 4,0. Тираж 5. Замовлення №

Видавець та виготовлювач Український державний університет залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.