МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

# ТЕСТУВАННЯ І ДІАГНОСТИКА ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ

Навчальний посібник

*Рекомендувала Науково-методична рада*
*Національного університету “Львівська політехніка”*

Львів
Видавництво Львівської політехніки
2021

УДК 004
Г 55

**Автори:**
**В. С. Глухов,** доктор технічних наук, професор, професор кафедри ЕОМ;
**М. О. Хомуляк,** старший викладач кафедри ЕОМ;
**Г. В. Бойко,** асистент кафедри ЕОМ;
**І. М. Жолубак,** асистент кафедри ЕОМ

**Рецензенти:**
**Оліярник Б. О.,** доктор технічних наук, професор, головний конструктор Державного підприємства "Львівський державний завод "ЛОРТА", лауреат Державної премії України в галузі науки і техніки;
**Лукенюк А. А.,** кандидат технічних наук, директор Львівського центру Інституту космічних досліджень НАН та ДКА України;
**Парамуд Я. С.,** кандидат технічних наук, доцент, доцент кафедри електронних обчислювальних машин Національного університету "Львівська політехніка"

*Рекомендувала Науково-методична рада*
*Національного університету "Львівська політехніка"*
*як навчальний посібник*
*для студентів спеціальності 123 "Комп'ютерна інженерія"*
*(протокол № 60 від 8.12.2021 р.)*

**Глухов В. С.**
Г 55 Тестування і діагностика програмно-апаратних засобів: навч. посібник / В. С. Глухов, М. О. Хомуляк, Г. В. Бойко, І. М. Жолубак. – Львів: Видавництво Львівської політехніки, 2021. – 140 с. Режим доступу: http://eom.lp.edu.ua/textbooks/np-tdpaz.pdf вільний. – Заголовок з екрана.
ISBN 978-966-941-695-7

До посібника увійшли лабораторний практикум з навчальної дисципліни "Тестування і діагностика програмно-апаратних засобів", завдання для самостійної роботи та термінологічний словничок. Методичні вказівки до лабораторних робіт містять теоретичні матеріали, покрокові інструкції для виконання, варіанти індивідуальних завдань, контрольні запитання, вимоги до звітів, електронні джерела інформації, рекомендовану літературу.

Навчальний посібник призначено для студентів спеціальності "Комп'ютерна інженерія" галузі знань "Інформаційні технології".

**УДК 004**



ISBN 978-966-941-695-7

**ЗМІСТ**

# ВСТУП

Навчальний посібник з дисципліни “Тестування і діагностика програмно-апаратних засобів” призначений для студентів першого (бакалаврського) рівня вищої освіти спеціальності 123 “Комп’ютерна інженерія”.

Лабораторні роботи орієнтовано на вивчення основних принципів та базових засобів діагностики й тестування функціональних вузлів електронних пристроїв, які є основою переважної більшості комп’ютерних систем.

Виконання лабораторних робіт побудовано на моделюванні нештатних ситуацій шляхом внесення несправностей у створені електронні вузли, а також на дослідженні особливостей взаємодії апаратури та програмних засобів. При цьому опрацьовуються такі питання:

– ознайомлення з методами та засобами тестування комбінаційних схем на прикладі арифметико-логічного пристрою;

– тестування цифрових схем із пам’яттю на прикладі оперативного запам’ятовувального пристрою;

– тестування цифрових схем із пам’яттю на прикладі цифрового автомата;

– ознайомлення з порядком розгортання та засобами програмного конфігурування та перевірки функціональних вузлів дослідницько-налагоджувальної плати з мікроконтролером;

– дослідження способів перевірки працездатності оперативної пам’яті налагоджувальної плати з використанням універсального асинхронного інтерфейсу.

Проведення лабораторних занять передбачає використання комп’ютерної бази і спеціального апаратного та програмного забезпечення навчальних лабораторій кафедри електронних обчислювальних машин Національного університету “Львівська політехніка”. Водночас лабораторний практикум максимально адаптовано до занять в умовах дистанційного навчання. Методичні матеріали призначено для допомоги студентам у підготовці до лабораторних занять, виконанні індивідуальних завдань, захисті отриманих результатів.

Завдання для самостійної роботи та термінологічний словничок дають змогу студентам перевірити свої знання та вміння при підготовці до контрольного заходу.

**Лабораторна робота № 1**

## ТЕСТУВАННЯ АРИФМЕТИКО-ЛОГІЧНОГО ПРИСТРОЮ

МЕТА РОБОТИ: ознайомлення із загальною схемою тестування цифрової техніки. Засвоєння методів та засобів тестування комбінаційних схем на прикладі арифметико-логічного пристрою (АЛП).

## ОЧІКУВАНИЙ РЕЗУЛЬТАТ РОБОТИ

Необхідно розробити програму тестування АЛП, описати поведінку еталонного АЛП (створити еталон), за результатами тестування АЛП визначити несправні елементи досліджуваного пристрою, несправні контакти визначених елементів та тип несправності (обрив, закоротка).

## ТЕОРЕТИЧНА ЧАСТИНА

### 1. Основні принципи тестування, налагодження і контролю

Основний принцип тестування, налагодження і контролю – це порівняння результатів роботи досліджуваного взірця (об’єкта) тестування й еталонного взірця – еталона (рис. 1).

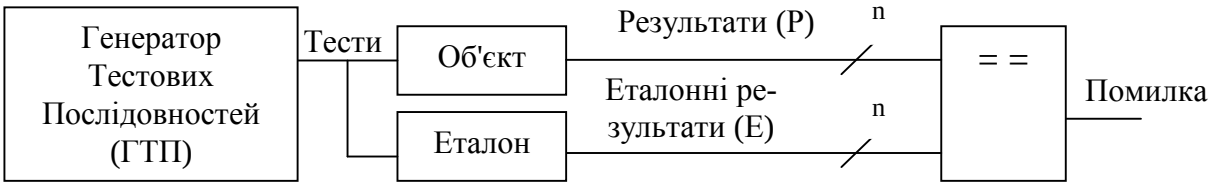



*Рис. 1. Структурна схема тестування, налагодження і контролю*

Тестові послідовності з генератора тестових послідовностей (ГТП) подаються одночасно на об’єкт тестування і на еталон. Схема порівняння перевіряє на збіг результати роботи об’єкта з еталонними результатами. У випадку невідповідності результатів на виході схеми порівняння формується ознака несправності – сигнал “Помилка”.

Еталони можуть бути найрізноманітнішими (рис. 2).

Еталон-модель може бути на папері – у технічних умовах (ТУ) на об’єкт і тоді порівняння результатів робить людина. При використанні засобів автоматизації модель може зберігатися у вигляді файлів. Наприклад, якщо об’єкт має підраховувати значення y=sin(x), то для кожного тестового значення засоби автоматизації також можуть обраховувати значення синуса (функціональна модель). Або ці значення вже наперед обраховані та зберігаються як модель (таблична модель).

Еталон-взірець може повністю збігатися за своїми характеристиками з об’єктом. Може бути простішим за нього. Наприклад, якщо перевіряється робота об’єкта, який має у своєму складі канал RS-232 і контролер ЛОМ, то під час перевірки роботи по каналу RS-232 немає необхідності, щоб еталон мав у собі контролер ЛОМ.

Еталон може бути складнішим за об’єкт, наприклад, під час перевірки реакції об’єкта на нештатну ситуацію – збій парності при роботі по каналу, яку неускладнений працездатний об’єкт ніколи допустити не може.

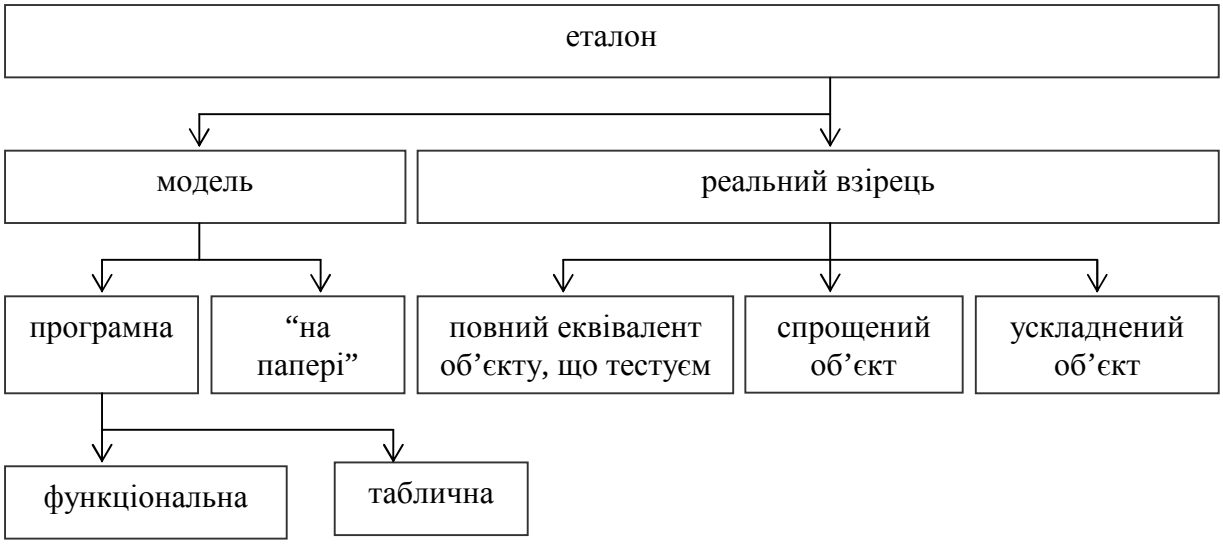



*Рис. 2. Класифікація еталонів*

## 2. Загальна схема випробовування цифрових пристроїв

При налагодженні цифрових пристроїв найчастіше користуються еталоном-моделлю. У цьому випадку загальна схема випробовування цифрових пристроїв набуває вигляду, як показано на рис. 3, де позначено:

ГТП – генератор тестових послідовностей;

ГЕП – генератор еталонних послідовностей;

ГМП – генератор масочних послідовностей.

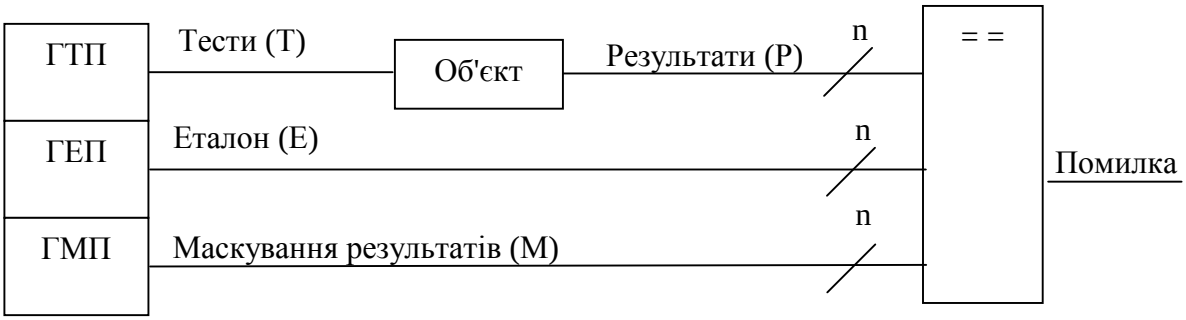



*Рис. 3. Загальна схема випробовування цифрових пристроїв*

Маскування дозволяє проводити порівняння лише коли відомі еталонні значення (коли усунуті всі невизначені стани об’єкта). Маскування дозволяє також перевіряти тільки визначений набір вихідних сигналів об’єкта. Вузол порівняння формує сигнал помилки згідно з формулою:

Помилка = $( P_0 \neq E_0 ) \& M_0 \vee ( P_1 \neq E_1 ) \& M_1 \vee \ldots \vee ( P_{n-1} \neq E_{n-1} ) \& M_{n-1},$

де n – кількість двійкових розрядів результату (еталона, маски).

Генератори можуть мати різну складність – від простих тумблерів і кнопок до спеціалізованих комп'ютерів. Схема порівняння також може бути реалізована по різному. Це може бути просто візуальна перевірка стану індикаторів, яку робить людина, і це може бути спеціалізований комп'ютер, який у реальному масштабі часу накопичує результати роботи об'єкта і робить їхнє апаратне або програмне порівняння з еталоном.

### 3. "Потужності" (приорітети) цифрових сигналів

Приорітети цифрових сигналів мають значення при визначенні результату взаємодії двох сигналів при виникненні закорочень. На рис. 4 показано такий випадок: два вихідні логічні сигнали A і B, кожний з яких може набуваати значення як логічної одниниці (1) так і логічного нуля (0), унаслідок помилки виявилися з'єднаними один з одним. Який рівень сигналу буде на об'єднаному зв'язку?

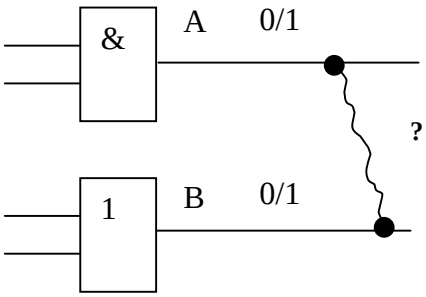



*Рис. 4. Закорочення цифрових сигналів*

За приорітетом цифрові сигнали поділяються на групи. Це означає, що сигнали з вищим приорітетом при закороченні з сигналами з нижчим приорітетом переважають їх. Далі наведені визначеня груп у порядку зменшення пріорітету.

Першу групу з найвищим приорітетом складають сигнали типу S (source) – джерела живлення та загальний провід.

Другу групу складають сигнали типу D (digital) – виходи цифрових мікросхем.

Третю групу складають сигнали з резистивним виходом R (rezistor) – виходи вузлів із відкритим колектором (ВК) чи емітером (ВЕ).

Четверту групу (з найнижчим приорітетом) складають сигнали з високоімпедансних виходів – Z.

### 4. Визначення рівня сигналу при об'єднанні кількох цифрових виходів

При з'єднанні цифрових виходів рівень напруги буде залежати в першу чергу від типу виходів, що з'єднуються. Результат закоротки двох сигналів з різними приорітетами можна визначити за допомогою таблиці 1, де позначено X –невизначений стан.

Результат взаємодії двох сигналів перебуває на перетині відповідних рядка та стовпця таблиці, наприклад, при закоротці сигналів типу S та Z результуючий рівень сигналу буде визначатися сигналом типу S.

Якщо взаємодіють два сигнали з однаковим приорітетом, то результуюче значення можна визначити за допомогою таблиці 2, де позначено:

0 – логічний 0, низький ТТЛ-рівень сигналу;

1 – логічна 1, високий ТТЛ-рівень сигналу.

Результат взаємодії двох сигналів перебуває на перетині відповідних рядка та стовпця таблиці. Результат взаємодії сигналу з рінем 0 і сигналу з рівнем 1 може бути різним для різних схем, і в таблиці 3.2 наведено орієнтовні дані. Найчастіше такий сигнал буде сприйматися як логічний 0.

*Таблиця 1*

**Конфлікти сигналів різного приорітету**

| **B\A** | **S** | **D** | **R** | **Z** |
|---|---|---|---|---|
| **S** | X | S | S | S |
| **D** | S | X | D | D |
| **R** | S | D | X | R |
| **Z** | S | D | R | X |

*Таблиця* 2

**Конфлікти сигналів однакового приорітету**

| **A/B** | **0** | **1** | **Z** |
|---|---|---|---|
| **0** | 0 | 0–90 %<br>1–10 % | 0 |
| **1** | 0–90 %<br>1–10 % | 1 | 1 |
| **Z** | 0 | 1 | Z |

**5. Обриви ТТЛ і КМОН входів цифрових мікросхем.**

У процесі виробництва чи під час експлуатації можливе виникнення обривів та закоротк провідників. Сприймання стану обірваних входів залежнить від технології виготовлення мікросхем (рис. 5):

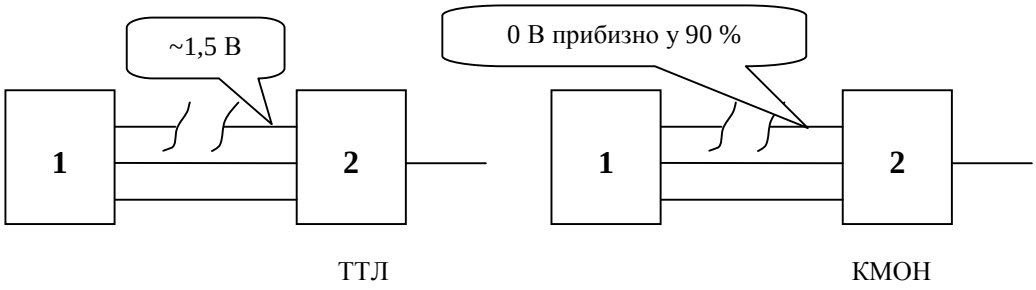



*Рис. 5. Обриви зв'язків*

У ТТЛ мікросхемах на обірваному вході (на інженерному жаргоні – "висить у повітрі") буде напруга приблизно 1,5 В, що зазвичай сприймається як логічна "1";

У КМОН важко навіть наближено вказати рівень напруги на обірваному вході. Вона може коливатися від 0 В до рівня напруги живлення. Приблизно в 90 % обрив сприйметься як логічний "0", у решті 10 % – як логічна "1".

Варто мати на увазі, що сприймання цифровими мікросхемами стану обірваних входів як логічного "0" або логічної "1" не є постійним ні в часі, ні для різних взірців, і може змінюватися під впливом багатьох факторів.

Ідеального тесту для виявлення обривів нема і обрив на вході складного цифрового вузла локалізувати без продзвонювання досить важко. Проте можна виявити обрив, запустивши ряд тестів (біжучий “0”, біжуча “1”, хвиля “0” та “1”) та проаналізувавши отримані результати. Оскільки протягом усього тесту на обірваному вході скоріше за все буде стале значення (“1” – в ТТЛ, “0” – в КМОН), то отримане значення функції від входу буде відрізнятись від еталонного, що б мало бути отриманим при проходженні такого тесту.

## 6. Рекомендована послідовність перевірки цифрових пристроїв

Рекомендують таку послідовність перевірки цифрових пристроїв:

1) візуальний огляд, виявлення механічних пошкоджень, невідповідність виробу конструкторській документації – відсутності елементів, неправильного їх встановлення і таке інше;

2) продзвонювання зв’язків – виявлення обривів;

3) продзвонювання зв’язків – виявлення закорочень. На практиці часто виконують спрощений варіант продзвонювання зв’язків потужних силових зв’язків – тільки живлення, особливо більше за +5 В і менше за 0 В, з сигнальними зв’язками – виявлення найбільш небезпечних закороток на живлення;

4) подача напруг живлення на виріб, перевірка напруг живлення;

5) подача напруг живлення на виріб, технологічне тренування;

6) подача напруг живлення на виріб, перевірка частот усіх генараторів і формувачів синхроімпульсів;

7) подача напруг живлення на виріб, перевірка його по тестам – виявлення усіх невиявлених раніше помилок:

8) для цифрових схем – перевірка шин об’єкта тестування, які з’єднують його з зовнішнім середовищем, тобто перевірка шляхів надходження інформації до вузлів об’єкта і зняття інформації з вузлів об’єкта. Для аналогових схем – перевірка шляхів надходження і зняття аналогових сигналів;

9) перевірка роботи вузлів об’єкта за тестами – перевірка логічного функціонування в нормальних умовах;

10) перевірка логічного функціонування в граничних умовах зовнішнього середовища (згідно з вимогами технічного завдання) – при підвищеній та понижених робочих температурах і таке інше.

Найбільш грубі і небезпечні помилки знаходяться саме на перших трьох етапах налагодження.

## 7. Тестові послідовності

Під час перевірки дослідного взірця виробу на його вхід подаються тестові послідовності сигналів. Серед них найбільш поширені такі:

біжуча 1;

біжучий 0;

хвиля 1;

хвиля 0;

шаховий код.

Ці послідовності ілюструються табл. 3 для випадку 8-розрядної шини даних.

Припустимо, що:

перевіряється буфер, який передає інформацію з входу на вихід;

старший розряд називається D7, а молодший D0.

На шині даних можуть бути такі помилки:

а) закоротка одного з розрядів, наприклад, D4, із шиною землі (із логічним 0);

б) закоротка одного з розрядів, наприклад, D4, із шиною +5 В (із логічною 1);

в) закоротка двох розрядів, наприклад, D4 і D5, до того ж рівень логічного 0 має більший приоритет.

Тоді після послідовної подачі на вхід буфера даних згідно з табл. 3 на його виході отримаємо дані згідно з табл. 4а…4в (відповідно до типу помилки).

Для кожної послідовності у таблицях виділене незбігання записаної та прочитаної інформації.

Як видно, кожна з тестових послідовностей дозволяє однозначно локалізувати місце, де є помилка.

Зауважмо, що шаховий код дозволяє виявляти закоротки тільки розрядів з різними значеннями (сусідні, або тих, що розділені 2*n бітами, де n=0, 1, 2…).

*Таблиця 3*

**Тестові послідовності**

| N | Біжуча 1 | Біжучий 0 | Хвиля 1 | Хвиля 0 | Шаховий код |
|---|---|---|---|---|---|
| 0 | 00000000 | 11111111 | 00000000 | 11111111 | 01010101 |
| 1 | 00000001 | 11111110 | 00000001 | 11111110 | 10101010 |
| 2 | 00000010 | 11111101 | 00000011 | 11111100 | 01010101 |
| 3 | 00000100 | 11111011 | 00000111 | 11111000 | 10101010 |
| 4 | 00001000 | 11110111 | 00001111 | 11110000 | 01010101 |
| 5 | 00010000 | 11101111 | 00011111 | 11100000 | 10101010 |
| 6 | 00100000 | 11011111 | 00111111 | 11000000 | 01010101 |
| 7 | 01000000 | 10111111 | 01111111 | 10000000 | 10101010 |
| 8 | 10000000 | 01111111 | 11111111 | 00000000 | 01010101 |

**8. Основна вимога до алгоритму перевірки працездатності цифрових схем.**

Найкращою перевіркою цифрової схеми була б така, що дозволяє розглядати усі виходи елементів схеми як розряди одного слова і дозволяє прогнати по розрядам цього слова одну з тестових послідовностей (біжучий “0”, біжуча “1” або інші). На жаль, практично такий підхід реалізувати неможливо. Наприклад, неможливо, щоб на прямому та інверсному виходах тригера була однакова інформація (0), що необхідно при перевірці за допомогою біжучої “1”.

Тому однією з основних вимог до алгоритмів перевірки працездатності цифрової схеми є забезпечення можливості отримання усіх можливих станів усіх виходів пристрою, що перевіряється. Наприклад, якщо на якомусь виході можуть бути стани логічного “0”, логічної “1” і високоімпедансний рівень, то тест має бути побудований так, щоб можна було перевірити наявність усіх цих станів.

Також важливою вимогою до алгоритму є можливість перевірки реакціїї кожного елемента пристрою на усі можливі стани і зміни станів на кожному з його входів. Наприклад, для тригера, що спрацьовує по перепаду сигналу на його синхровході з високого рівня на низький, необхідно перевіряти, як себе буде поводити тригер при наявності на вході синхронізації логічного “0”, логічної “1” і відповідного перепаду.

*Таблиця 4а*

**Виявлення помилок за допомогою тестових послідовностей**

| N | Біжуча 1 | Біжучий 0 | Хвиля 1 | Хвиля 0 | Шаховий код |
|---|---|---|---|---|---|
| 0 | 00000000 | 111**0**1111 | 00000000 | 111**0**1111 | 010**0**0101 |
| 1 | 00000001 | 111**0**1110 | 00000001 | 111**0**1110 | 10101010 |
| 2 | 00000010 | 111**0**1101 | 00000011 | 111**0**1100 | 010**0**0101 |
| 3 | 00000100 | 111**0**1011 | 00000111 | 111**0**1000 | 10101010 |
| 4 | 00001000 | 111**0**0111 | 00001111 | 111**0**0000 | 010**0**0101 |
| 5 | 000**0**0000 | 11101111 | 000**0**1111 | 11100000 | 10101010 |
| 6 | 00100000 | 110**0**1111 | 001**0**1111 | 11000000 | 010**0**0101 |
| 7 | 01000000 | 101**0**1111 | 011**0**1111 | 10000000 | 10101010 |
| 8 | 10000000 | 011**0**1111 | 111**0**1111 | 00000000 | 010**0**0101 |

*Таблиця 4б*

| N | Біжуча 1 | Біжучий 0 | Хвиля 1 | Хвиля 0 | Шаховий код |
|---|---|---|---|---|---|
| 0 | 000**1**0000 | 11111111 | 000**1**0000 | 11111111 | 01010101 |
| 1 | 000**1**0001 | 11111110 | 000**1**0001 | 11111110 | 101**1**1010 |
| 2 | 000**1**0010 | 11111101 | 000**1**0011 | 11111100 | 01010101 |
| 3 | 000**1**0100 | 11111011 | 000**1**0111 | 11111000 | 101**1**1010 |
| 4 | 000**1**1000 | 11110111 | 000**1**1111 | 11110000 | 01010101 |
| 5 | 00010000 | 111**1**1111 | 00011111 | 111**1**0000 | 101**1**1010 |
| 6 | 001**1**0000 | 11011111 | 00111111 | 110**1**0000 | 01010101 |
| 7 | 010**1**0000 | 10111111 | 01111111 | 100**1**0000 | 101**1**1010 |
| 8 | 100**1**0000 | 01111111 | 11111111 | 000**1**0000 | 01010101 |

*Таблиця 4в*

| N | Біжуча 1 | Біжучий 0 | Хвиля 1 | Хвиля 0 | Шаховий код |
|---|---|---|---|---|---|
| 0 | 00000000 | 11111111 | 00000000 | 11111111 | 010**0**0101 |
| 1 | 00000001 | 11111110 | 00000001 | 11111110 | 10**0**01010 |
| 2 | 00000010 | 11111101 | 00000011 | 11111100 | 010**0**0101 |
| 3 | 00000100 | 11111011 | 00000111 | 11111000 | 10**0**01010 |
| 4 | 00001000 | 11110111 | 00001111 | 11110000 | 010**0**0101 |
| 5 | 000**0**0000 | 11**0**01111 | 000**0**1111 | 11**0**00000 | 10**0**01010 |
| 6 | 00**0**00000 | 110**0**1111 | 00111111 | 11000000 | 010**0**0101 |
| 7 | 01000000 | 10111111 | 01111111 | 10000000 | 10**0**01010 |
| 8 | 10000000 | 01111111 | 11111111 | 00000000 | 010**0**0101 |

Важливою є також можливість алгоритму виявляти закоротки сигналів. Перевірка на можливі закоротки виконується за допомогою тестових послідовностей (біжучий “0”, біжуча “1” або інші), але стосовно окремих елементів пристрою, що перевіряються (найчастіше шин).

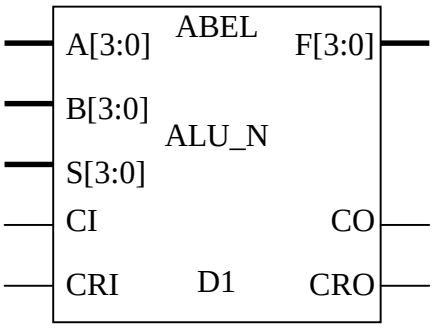



*Рис. 6. Арифметико-логічний пристрій*

### 9. Особливості тестування комбінаційних схем (на прикладі АЛП)

Комбінаційні схеми – це вузли цифрової техніки без пам’яті, вихід яких залежить тільки від стану їхніх входів у даний момент часу. До комбінаційних схем можна віднести вузли, що реалізують функції алгебри логіки, буфери, АЛП, ПЗП, дешифратори, мультиплексори тощо.

Особливістю тестування арифметико-логічних пристроїв (АЛП, рис. 3.6) є те, що АЛП – це комбінаційна схема і результати роботи АЛП виникають на виходах відразу після подачі на входи тестового набору.

АЛП має групові входи:

– першого n-розрядного операнда (A);

– другого n-розрядного операнда (B);

– k-розрядного коду операції (S).

АЛП має однорозрядні входи:

– арифметичного переносу (CI);

– переносу при зсуві праворуч (CRI).

АЛП має n-розрядний вихід результату (F) і однорозрядні виходи арифметичного переносу (CO) і переносу при зсуві праворуч (CRO).

Спочатку перевіряють шинні структури АЛП (перевіряють, чи доходять до АЛП оператори і чи знімаються з АЛП результати). Для цього обирається операція трансляції даних з одного входів (A або B) на вихід F:

F := A або F :=B.

На входи A (B) подається одна з тестових послідовностей, результат перевіряється на виході F. Помилка може виникнути на входах і на виходах АЛП. Якщо помилка на виході, то результати перевірки трансляції даних з обидвох входів і виявлені при цьому помилки будуть повторюватися.

Після перевірки шин перевіряється правильність виконання кожної з операцій АЛП (операції визначаються кодом на входах S). Особлива увага має приділятися перевірці вхідних і вихідних переносів АЛП.

## ХАРАКТЕРИСТИКА РОБОЧОГО МІСЦЯ

Лабораторна робота виконується у середовищі Xilinx ISE WebPack.

При тестуванні створюються такі вузли.

1. Еталонний вузол.
2. Еталонний вузол з наперед заданою помилкою.
3. Вузол порівняння.
4. Генератор тестових послідовностей.

Розглянемо приклад тестування на основі дешифратора.

1. Створюємо проєкт в Xilinx ISE WebPack. Натискаємо кнопку File->New Project. З'являється вікно, в якому потрібно вказати назву проєкту, місце на диску і робочий каталог. Зі списку "Top-level source type" вибираємо "Schematic" і натискаємо кнопку "Next". Потім вибираємо сімейство FPGA – "Spartan3A and Spartan3AN", її тип – "XC3550A" і корпус – "TQ144", а також інструменти для синтезу – "XST (VHDL/Verilog)" і симуляції – "Isim (VHDL/Verilog)". Натискаємо кнопку "Next" і завершуємо створення проєкту. Ми створили середовище, в якому можна створювати цифрові вузли.

2. Для отримання еталонного вузла створюємо файли двох типів: графічний (схема) з розширенням *.sch і текстовий (мовою VHDL) з розширенням *.vhd. Як приклад наводиться схема дешифратора 2x4, описана мовою VHDL.

```
Library ieee;
use ieee.std_logic_1164.all;

  entity DC_2X4 is
        port(
              A : in std_logic_vector(1 downto 0);
              S : out std_logic_vector(3 downto 0));
  end DC_2X4;

  architecture Arch_DC_2X4 of DC_2X4 is
        signal    Y: std_logic_vector(3 downto 0);
  begin
        process(A,Y)
        begin
              case A is
                    when «00» => Y <= «0001»;
                    when «01» => Y <= «0010»;
                    when «10» => Y <= «0100»;
                    when «11» => Y <= «1000»;
                    when others => Y <= «XXXX»;
              end case;
              S <= Y;
        end process;
  end Arch_DC_2X4;
```

3. Створюємо ще один такий самий вузол, але з помилкою.
4. Створюємо вузол, який буде виконувати порівняння.

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Comparison is
port (  C : in std_logic_vector (3 downto 0);
        D : in std_logic_vector (3 downto 0);
        Output : out std_logic
       );
end Comparison;

architecture Arch_Comparator of Comparison is

begin
   Output <= '1' when (C=D) else '0';
end Arch_Comparator;
```

5. Створюємо генератор тестових послідовностей (TestBench) і порівнюємо результати.

## ПОРЯДОК РОБОТИ

Змоделювати арифметико-логічний пристрій та протестувати його при різних помилках. Арифметико-логічний пристрій має виконувати такі операції (варіанти завдань):

1) операція віднімання;
2) операція І;
3) операція АБО;
4) операція виключне АБО;
5) інкремент А;
6) інкремент В;
7) інверсія А;
8) інверсія В;
9) логічний зсув ліворуч А;
10) логічний зсув ліворуч В;
11) зсув праворуч А;
12) логічний зсув праворуч В;
13) декремент А;
14) декремент В;
15) константа 0.

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.

2. Теоретична частина у тезовому викладі (стисло, без рисунків і таблиць).

3. Індивідуальне завдання.

4. Опис послідовності виконання роботи. Результати проілюструвати скриншотами зі схемами та часовими діаграмами.

5. Висновки.

6. Додаток. Тексти програмних модулів всіх створених вузлів.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Який основний принцип варто використовувати для тестування, налагодження і контролю?

2. Типовий склад стенда для здійснення тестування, налагодження і контролю.

3. Якими можуть бути еталони?

4. Які особливості має випробовування цифрових пристроїв?

5. На які групи за пріоритетом поділяються цифрові сигнали при їх закороченні?

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. Москва: Горячая линия-Телеком, 2003. 624 с.

2. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. Москва: Горячая линия-Телеком, 2005. 252 с.

3. Соловьев В. В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. Москва: Горячая линия-Телеком, 2008. 376 с.

## Лабораторна робота № 2

## ТЕСТУВАННЯ ОЗП

МЕТА РОБОТИ: ознайомлення із загальною схемою тестування цифрової техніки. Засвоєння методів та засобів тестування цифрових схем із пам'яттю на прикладі оперативного запам'ятовувального пристрою (ОЗП).

## ОЧІКУВАНИЙ РЕЗУЛЬТАТ РОБОТИ

Необхідно розробити програму тестування ОЗП, за результатами тестування ОЗП визначити несправні виводи та біти ОЗП, а також тип несправності (обрив, закоротка).

## ТЕОРЕТИЧНА ЧАСТИНА

**Особливості тестування схем з пам'яттю (на прикладі ОЗП).**

Вихід цифрового вузла з пам'яттю залежить як від стану його входів у цей момент часу, так і від їхнього стану у попередні моменти часу. Ця обставина значно ускладнює тестування таких вузлів порівняно з комбінаційними схемами, оскільки реакція на тестовий набір виникає на виході вузла зі значною затримкою у часі і ця затримка залежить від функції і внутрішньої структури вузла. До вузлів із пам'яттю можна віднести оперативний запам'ятовувальний пристрій (ОЗП), цифровий автомат тощо.

ОЗП (рис. 3.1) має такі групові входи:

– адреси (A);

– вхідних даних (DI – Data Input);

– входи дозволу видачі інформації на вихід (OE – Output Enable), запису (WR) і читання (RD);

– вхід, який забезпечує нарощування об'єму ПЗП (CS – Chip Select).

ОЗП має груповий вихід даних DO – Data Output.

Основні кількісні характеристики ОЗП:

– кількість слів $N = 2^n$;

– об'єм пам'яті $V = m*N = m*2^n$ біт.

Перевірка ОЗП складається з кількох етапів. Спочатку перевіряється, чи інформація доходить до входів мікросхем ОЗП і знімається з виходів (це перевіряється так званими тестами шини адреси та шини даних). У цих тестах задіяні тільки деякі комірки (біти) запам'ятовуючого масиву мікросхем пам'яті. На другому етапі перевіряється відсутність помилок у всьому запам'ятовуючому масиві мікросхем ОЗП – перевіряються усі його комірки (біти). Це так званий тест на збереження інформації.

Кожна перевірка ОЗП складається з рознесених у часі циклів запису інформації до ОЗП (коли на ОЗП подається тестова послідовність сигналів) і циклів читання інформації з ОЗП (коли власне на виході даних ОЗП з'являється результат його роботи).

Основна мета тесту шини даних – пересвідчитися, що немає закороток і обривів на шині даних. Для цього обирається одна фіксована адреса ОЗП, наприкалад, 0, і послідовно

записується до неї і зчитується з неї одна з тестових послідовністей кодів. Такий тест вимагає проведення m циклів запису до ОЗП і m циклів читання з ОЗП.

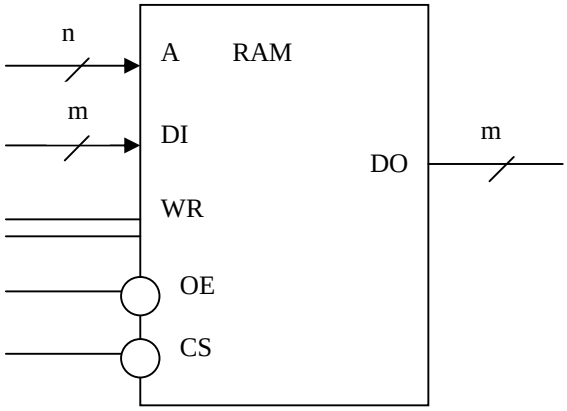



*Рис. 1. Умовне графічне позначення ОЗП*

Основна мета тесту шини адреси – пересвідчитися, що немає закороток і обривів на шині адреси. Для цього обирається група фіксованих адрес, наприклад, адреси, серед розрядів яких є тільки по одній логічній одиниці – група адрес з біжучою одиницею.

Далі виконується перевірка за таким алгоритмом (рис. 2):

а) за всіма адресами цієї групи записують фіксовану інформацію – “фон” (наприклад, двійкові нулі 0..0);

б) за першою адресою групи записують протилежну фону інформацію (“не фон” – наприклад, двійкові одиниці 1..1);

в) перевіряють наявність фону у решті адрес групи;

г) якщо фон за всіма адресами не змінився, відновлюють фон у комірці, де був записаний “не фон”. Якщо фон десь змінився, то це ознака помилки;

д) записують “не фон” до наступної адреси групи і повторюють виконання пунктів в) і г), доки “не фон” не буде послідовно прописаний за всіма адресами групи.

Такий тест шини адреси вимагає проведення 2*n циклів запису до ОЗП і $n^2$ циклів читання з ОЗП.

Оскільки при перевірках шини адреси і шини даних задіяні лише деякі адреси пам'яті, то слід перевірити доступність даних до кожного біта пам’яті. Для цього виконується перевірка на збереження інформації:

а) за всіма адресами пам’яті прописують двійкові нулі 0..0;

б) перевіряють, що за всіма адресами пам’яті прописано 0..0. Якщо десь з’являється двійкова 1 – це ознака помилки.

В) за всіма адресами пам’яті прописують двійкові одиниці 1..1;

б) перевіряють, що за всіма адресами пам’яті прописано 1..1. Якщо десь з’являється двійковий 0 – це ознака помилки.

У цьому тесті запис даних до пам’яті і перевірка вмісту пам’яті можуть бути суміщені. Наприклад, перед записом до якоїсь комірки двійкових одиниць 1..1 робиться перевірка вмісту цієї комірки (цикл читання з неї). При нормальній роботі там мають бути двійкові 0.

| Адреса | Дані |
|---|---|
| 0…00 | 1…1 |
| 0…01 | 0…0 |
| | |
| i-1 | 0…0 |
| i | 0…0 |
| i+1 | 0…0 |
| | |
| F…FE | 0…0 |
| F…FF | 0…0 |

Початок

| Адреса | Дані |
|---|---|
| 0…00 | 0…0 |
| 0…01 | 0…0 |
| | |
| i-1 | 0…0 |
| i | 1…1 |
| i+1 | 0…0 |
| | |
| F…FE | 0…0 |
| F…FF | 0…0 |

Середина

| Адреса | Дані |
|---|---|
| 0…00 | 0…0 |
| 0…01 | 0…0 |
| | |
| i-1 | 0…0 |
| i | 0…0 |
| i+1 | 0…0 |
| | |
| F…FE | 0…0 |
| F…FF | F…F |

Кінець

*Рис. 2. Зміна вмісту комірок ОЗП під час тестування*

Такий тест збереження інформації вимагає проведення $2^{n+1}$ циклів запису до ОЗП і $2^{n+1}$ циклів читання з ОЗП.

Перевірка РПЗП нічим суттєвим не відрізняється від перевірки ОЗП, але треба пам’ятати, що кількість циклів запису до РПЗП зазвичай обмежена, і часте тестування РПЗП, на відміну від ОЗП, недопустиме. Тестування РПЗП, як ПЗП (за контрольними сумами), може виконуватися практично без обмежень.

## ХАРАКТЕРИСТИКА РОБОЧОГО МІСЦЯ

Лабораторна робота виконується у середовищі Xilinx ISE WebPack.

При тестуванні створюються такі вузли:

1. Генератор тестових послідовностей, який перебирає всі можливі послідовності.
2. Еталонний вузол.
3. Еталонний вузол з наперед заданою помилкою.
4. Вузол порівняння.

Розглянемо приклад тестування на основі дешифратора.

1. Створюємо проєкт в Xilinx ISE WebPack. Натискаємо кнопку File>New Project. З’являється вікно, в якому потрібно вказати назву проєкту, місце на диску і робочий каталог. Зі списку “Top-level source type” вибираємо “Schematic” і натискаємо кнопку “Next”. Потім вибираємо сімейство FPGA – “Spartan3A and Spartan3AN”, її тип – “XC3550A” і корпус – “TQ144”, а також інструменти для синтезу – “XST (VHDL/Verilog)” і симуляції – “Isim (VHDL/Verilog)”. Натискаємо кнопку “Next” і завершуємо створення проєкту. Ми створили середовище, в якому можна створювати цифрові вузли.

2. Для отримання еталонного вузла створюємо файли двох типів: графічний (схема) з розширенням *.sch і текстовий (мовою VHDL) із розширенням *.vhd. Як приклад наводиться схема дешифратора 2x4, описана мовою VHDL.

```
Library ieee;
use ieee.std_logic_1164.all;

  entity DC_2X4 is
        port(
              A : in std_logic_vector(1 downto 0);
              S : out std_logic_vector(3 downto 0));
  end DC_2X4;

  architecture Arch_DC_2X4 of DC_2X4 is
        signal    Y: std_logic_vector(3 downto 0);
  begin
        process(A,Y)
        begin
              case A is
                    when “00” => Y <= “0001”;
                    when “01” => Y <= “0010”;
                    when “10” => Y <= “0100”;
                    when “11” => Y <= “1000”;
                    when others => Y <= “XXXX”;
              end case;
              S <= Y;
        end process;
  end Arch_DC_2X4;
```

3. Створюємо ще один такий самий вузол, але з помилкою.
4. Створюємо вузол, який буде виконувати порівняння.

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Comparison is
port ( C: in std_logic_vector (3 downto 0);
       D: in std_logic_vector (3 downto 0);
       Output: out std_logic
      );
end Comparison;

architecture Arch_Comparator of Comparison is
```

```
begin
   Output <= ‘1’ when (C=D) else ‘0’;
end Arch_Comparator;
```

5. Створюємо генератор тестових послідовностей (TestBench) відповідно до вимог.

## МОДЕЛЬ ПОМИЛОК

1. Стенд працює без помилок.
2. Модель помилок у досліджуваному вузлі (варіанти завдань):
   1) обрив вхідного сигналу;
   2) закоротка вхідного сигналу на корпус;
   3) закоротка вхідного сигналу на живлення +5 В;
   4) закоротка вхідного сигналу з будь-яким іншим логічним сигналом;
   5) обрив вихідного сигналу;
   6) закоротка вихідного сигналу на корпус;
   7) закоротка вихідного сигналу на живлення +5 В;
   8) закоротка вихідного сигналу з будь-яким іншим логічним сигналом;
   9) обрив внутрішнього сигналу;
   10) закоротка внутрішнього сигналу на корпус;
   11) закоротка внутрішнього сигналу на живлення +5 В;
   12) закоротка внутрішнього сигналу з будь-яким іншим логічним сигналом.
3. Локалізація помилок.

Помилки можуть бути:

– на шині даних;
– на шині адреси;
– у внутрішньому масиві пам’яті вузла.

## ПОРЯДОК РОБОТИ

Змоделювати ОЗП та протестувати його при різних помилках. Реалізувати відповідно до варіанту вузол генерування тестових послідовностей:

1) шини адреси;
2) шини даних;
3) перевірки на збереження інформації.

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.
2. Теоретична частина у тезовому викладі (стисло, без рисунків і таблиць).
3. Індивідуальне завдання.
4. Опис послідовності виконання роботи. Результати проілюструвати скриншотами зі схемами та часовими діаграмами.
5. Висновки.
6. Додаток. Тексти програмних модулів всіх створених вузлів.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які особливості має тестування цифрового вузла з пам’яттю?
2. Із яких етапів складається перевірка ОЗП і мета кожного з них?
3. Який алгоритм перевірки шини даних?
4. Який алгоритм перевірки шини адрес?
5. Який алгоритм тесту на збереження інформації?

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. Москва: Горячая линия-Телеком, 2003. 624 с.

2. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. Москва: Горячая линия-Телеком, 2005. 252 с.

3. Соловьев В. В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. Москва: Горячая линия-Телеком, 2008. 376 с.

## Лабораторна робота № 3

## ТЕСТУВАННЯ ЦИФРОВОГО АВТОМАТА

МЕТА РОБОТИ: ознайомлення із загальною схемою тестування цифрової техніки. Засвоєння методів та засобів тестування цифрових схем з пам'яттю на прикладі цифрового автомата (ЦА).

## ОЧІКУВАНИЙ РЕЗУЛЬТАТ РОБОТИ

Необхідно розробити програму тестування ЦА, описати поведінку еталонного ЦА (створити еталон), за результатами тестування ЦА знайти прояви помилки, визначити тип несправності (неправильне формування вихідних сигналів, неправильний перехід до наступного стану), визначити причину несправності (обрив або закоротка вхідного чи вихідного сигналу).

## ТЕОРЕТИЧНА ЧАСТИНА

**Особливості тестування цифрових автоматів.**

Цифровий автомат є прикладом пристрою із пам'яттю.

Цифровий автомат характеризується:

– набором вхідних сигналів {X};

– набором вихідних сигналів {Y};

– набором внутрішніх станів {A};

– початковим станом $a_0$;

– правилом формування вихідних сигналів;

– правилом формування наступного внутрішнього стану.

Існують такі способи завдавання автоматів, які описують залежність відповідно до наступного стану автомата і його виходу від теперешнього стану автомата, його входів:

– табличний – задаються дві таблиці: станів (переходів) і виходів;

– за допомогою часових діаграм;

– аналітичний, у тому числі мовами опису апаратної частини цифрових вузлів – Hardware Discription Language (HDL);

– за допомогою алгоритму роботи автомата;

– за допомогою графа автомата (рис. 1);

– опис звичайною мовою.

Цифрові автомати поділяються на автомати:

– Мура, в яких вихідні сигнали залежать тільки від стану автомата;

– Мілі, в яких вихідні сигнали залежать як від стану автомата, так і від вхідних сигналів.

Здебільшого цифрові автомати бувають синхронними, тобто зміна стану автомата здійснюється фронтом синхроімпульсів.

На рис. 3.1 представлено граф синхронного автомата Мура, який має:

– вісім внутрішніх станів (S0, …, S7), що кодуються трьома розрядами ($s_0$, …, $s_2$);

– три вхідні сигнали (pusk, $x_0$, $x_1$);

– два вихідні сигнали ($y_0$, $y_1$).

Названий автомат переводиться до початкового стану за допомогою асинхронного сигналу скиду (Reset). Синхроімпульси на графі не показано.

Основний принцип тестування цифрових автоматів полягає у перевірці того, що автомат:

– здійснює усі позначені стрілочками на графі переходи і не здійснює не позначених;

– формує усі позначені на графі вихідні сигнали в потрібних станах і не формує не позначених.

Для перевірки усіх цих вимог потрібно декілька разів повертати автомат до початкового стану і повторно проводити його перевірку з іншою послідовністю вхідних сигналів.

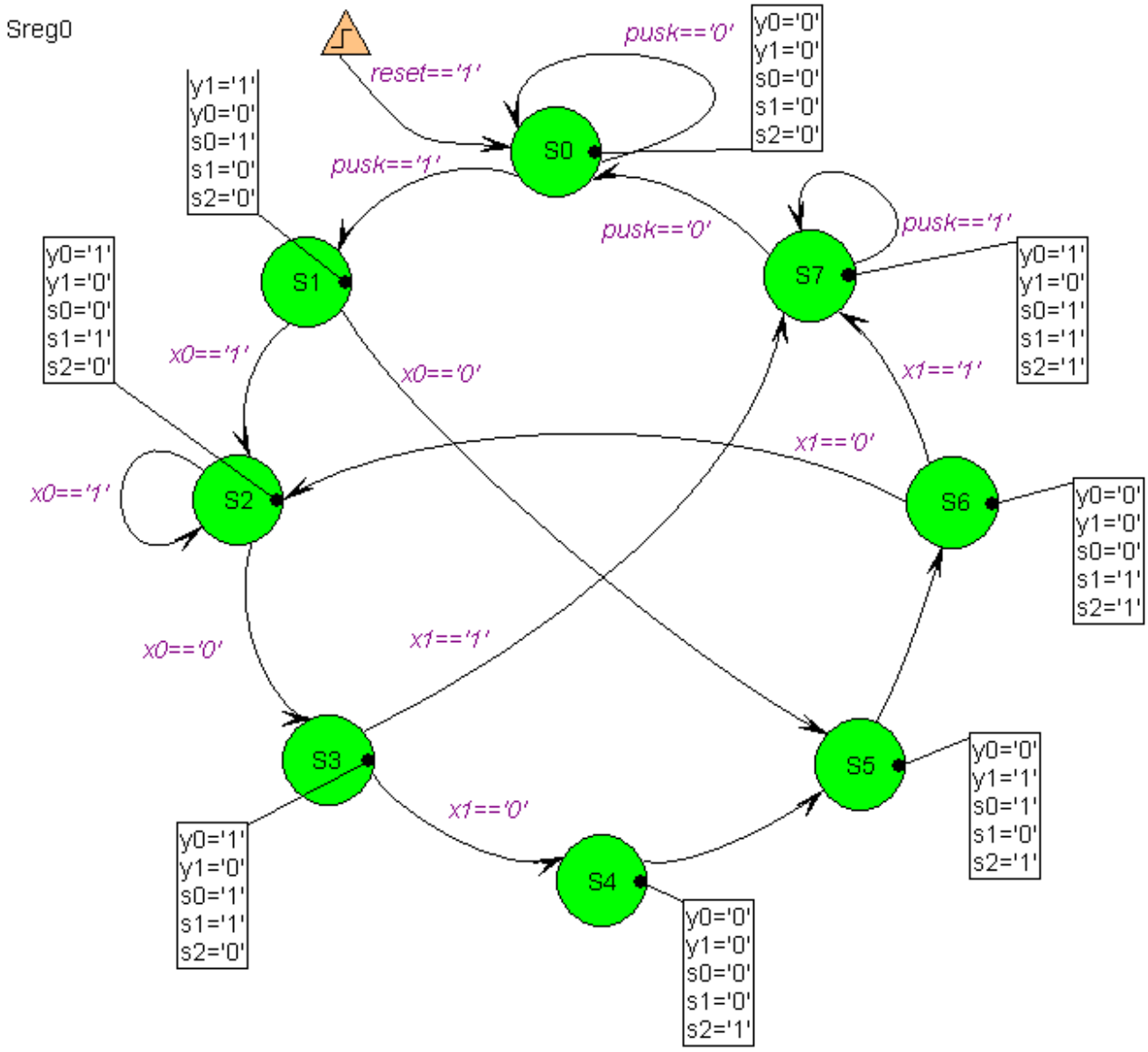



*Рис. 1. Граф автомата*

## ХАРАКТЕРИСТИКА РОБОЧОГО МІСЦЯ

У цій роботі використовується еталон у вигляді табличної моделі, тобто у файлі SIMUL.CMD у вигляді таблиці зберігаються результати роботи об’єкта при подаванні йому на вхід деякого обмеженого набору тестових даних.

Робоче місце моделюється за допомогою пакета Xilinx ISE WebPack, призначеного для проєктування програмованих логічних інтегральних схем (ПЛІС) і описується набором файлів відповідного формату (табл. 1).

*Таблиця 1*

**Формати файлів**

| РОЗШИРЕННЯ | ПРИЗНАЧЕННЯ ФАЙЛА |
|---|---|
| .SCH | Схема електрична принципова |
| .ABL | Опис роботи бібліотечного елемента (моделі, мовою ABEL) |
| .CMD | Програма моделювання досліджуваного пристрою |

Структурно модель робочого місця (рис. 2 – State_st1.SCH) складається з досліджуваного пристрою (мікросхема D3) і стенда.

Досліджуваний пристрій може бути еталонним (state_s), алгоритм його роботи міститься у файлі state_s.asf.

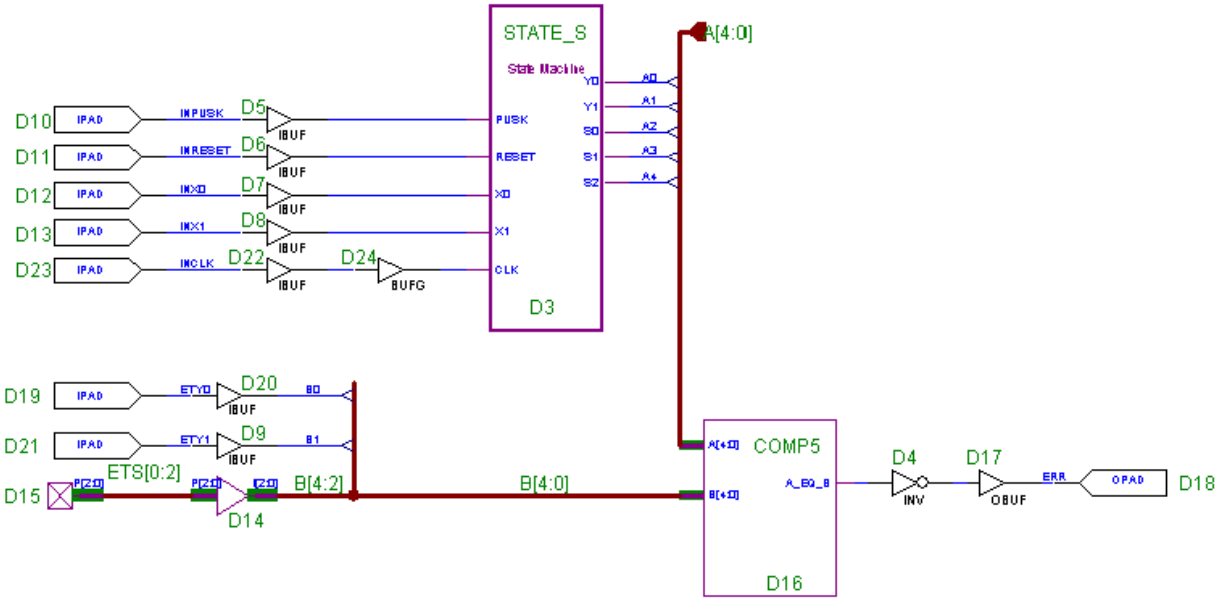



*Рис. 2. Функціональна схема стенда*

Стенд складається з таких функціональних вузлів:

– зовнішнього генератора тестових послідовностей та еталонних результатів (файл SIMUL.CMD);

– схеми порівняння еталонних результатів та реальних (елемент D16 COMP5);

– вхідних (IBUF) та вихідних (OBUF) буферів, а також контактів (IPAD, OPAD).
– може виконуватися практично без обмежень.

Лабораторна робота виконується у середовищі Xilinx ISE WebPack.
При тестуванні створюються такі вузли:
1. Генератор тестових послідовностей, який перебирає всі можливі послідовності.
2. Еталонний вузол.
3. Вузол з наперед заданою помилкою.
4. Вузол порівняння.
Розглянемо приклад тестування на основі дешифратора.
1. Створюємо проєкт і діаграму станів в Xilinx ISE WebPack, згідно зі зразком в теоретичній частині
3. Створюємо ще один такий самий вузол, але з помилкою.
4. Створюємо вузол, який буде виконувати порівняння.

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Comparison is
port (  C : in std_logic_vector (3 downto 0);
        D : in std_logic_vector (3 downto 0);
        Output : out std_logic
       );
end Comparison;

architecture Arch_Comparator of Comparison is

begin
   Output <= ‘1’ when (C=D) else ‘0’;
end Arch_Comparator;
```

5. Створюємо генератор тестових послідовностей відповідно до вимог.

## МОДЕЛЬ ПОМИЛОК

1. Стенд працює без помилок.
2. Модель помилок у досліджуваному взірці (варіанти завдань):
   1) обрив вхідного сигналу;
   2) закоротка вхідного сигналу на корпус;
   3) закоротка вхідного сигналу на живлення +5 В;
   4) закоротка вхідного сигналу з будь-яким іншим логічним сигналом;
   5) обрив вихідного сигналу;
   6) закоротка вихідного сигналу на корпус;
   7) закоротка вихідного сигналу на живлення +5 В;

8) закоротка вихідного сигналу з будь-яким іншим логічним сигналом;

9) обрив внутрішнього сигналу;

10) закоротка внутрішнього сигналу на корпус;

11) закоротка внутрішнього сигналу на живлення +5 В;

12) закоротка внутрішнього сигналу з будь-яким іншим логічним сигналом.

## ПОРЯДОК РОБОТИ

Змоделювати діаграму станів та протестувати її при всіх можливих комбінаціях.

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.
2. Теоретична частина у тезовому викладі (стисло, без рисунків і таблиць).
3. Індивідуальне завдання.
4. Опис послідовності виконання роботи. Результати проілюструвати скриншотами зі схемами та часовими діаграмами.
5. Висновки.
6. Додаток. Тексти програмних модулів всіх створених вузлів.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Чим характеризується цифровий автомат?
2. Якими способами можна задавати (описувати) цифрові автомати?
3. Чим відрізняються цифрові автомати Мура і Мілі?

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. Москва: Горячая линия-Телеком, 2003. 624 с.

2. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. Москва: Горячая линия-Телеком, 2005. 252 с.

3. Соловьев В. В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. Москва: Горячая линия-Телеком, 2008. 376 с.

**Лабораторна робота № 4**

# ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ ІНСТРУМЕНТАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

МЕТА РОБОТИ: ознайомитися з порядком реєстрації користувача та встановлення і налаштування інструментального програмного забезпечення.

## ТЕОРЕТИЧНА ЧАСТИНА

### Короткий огляд програмних та апаратних засобів

**STM32CubeMX** є візуальним графічним редактором для конфігурування мікроконтролерів сімейства STM32, що дозволяє генерувати код на основі мови С, використовуючи для цього графічних помічників.

Створений компанією STMicroelectronics програмний пакет STM32CubeMX є справжнім автоматизованим робочим місцем для розробників систем на базі 32-бітових мікроконтролерів STM32, виконаних на основі ядер ARM Cortex. Це зручне середовище для повноцінної настройки конфігурації мікроконтролера з видачею пакета файлів ініціалізації, придатних для подальшого використання в ряді систем розробки і налагодження керуючого коду мікроконтролера. STM32CubeMX значно спрощує створення вбудованого програмного забезпечення, прискорює цей процес, не вимагає від фахівців-початківців досконального знання документації на мікроконтролер, дозволяє обійтися початковими даними про апаратну і програмну архітектури контролера, про можливості бібліотек програмного забезпечення. STM32CubeMX частково дозволяє ефективно поєднати роботу над своїми першими проєктами з вивченням можливостей мікроконтролера.

**Keil MDK-ARM**. Програмний пакет розробки RealView Microcontroller Development Kit (MDK-ARM) компанії Keil об'єднує компілятор C/C++ ARM RealView та інтегроване середовище розробки (IDE) Keil µVision. Цей продукт надає у розпорядження розробника багатофункціональний засіб проєктування, оптимізований для роботи з широкою лінійкою мікроконтролерів на базі ядра ARM. MDK-ARM забезпечує підтримку пристроїв, що базуються на ARM7, ARM9 і Cortex-M таких виробників, як Analog Devices, Atmel, Freescale, Luminary, OKI, NXP, Samsung, Sharp, STMicroelectronics, Texas Instruments тощо. Використання MDK-ARM дозволяє значно зменшити цикл проєктування та істотно скоротити час виходу продукту на ринок. MDK-ARM – ідеальний засіб, стандартизований для промислових проєктів з розвиненою системою налагодження і підтримкою реального часу.

**ST-Link V2** є внутрісхемним відладчиком-програматором для мікроконтролерів серії STM32 і STM8. Він здатний виконувати операції програмування і налагодження через інтерфейс SWIM (що зручно для роботи з мікроконтролерами STM8), а також через інтерфейси SWD і JTAG (підходять для мікроконтролерів серії STM32).

Із мікроконтролерами серії STM8 є можливість працювати за допомогою програмного забезпечення ST Visual Develop (STVD) або не менш популярного ST Visual Program (STVP). З мікроконтролерами серії STM32 можна працювати за допомогою інтегрованих засобів розробки, таких як IAR, Atollic, Keil MDK, CooCox і Tasking.

Програматор підтримує повноцінну роботу з інтерфейсом USB версії 2.0, тому передача даних між програматором і комп'ютером може відбуватися на високій швидкості.

Перетворювач **CP2102** – багатофункціональний перехідник USB – UART на основі мікросхеми фірми Silicon Labs. Використовується для програмування мікросхем і мікроконтролерів, що підтримують рівні сигналів TTL. Призначений для забезпечення зв'язку між пристроєм з послідовним інтерфейсом UART і персональним комп'ютером за допомогою порту USB.

## ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

### 1. Встановлення STM32CubeMX

1.1. У пошуковому полі браузера набрати: STM32CubeMX.

1.2. Перейти на вебсторінку "STM32CubeMX" фірми-виробника STMicroelectronics і послідовно натиснути спочатку верхню (рис. 1.1), а згодом, після автопрокрутки, і нижню (рис. 1.2) кнопки "Get Software" відповідно до наявної на комп'ютері операційної системи.

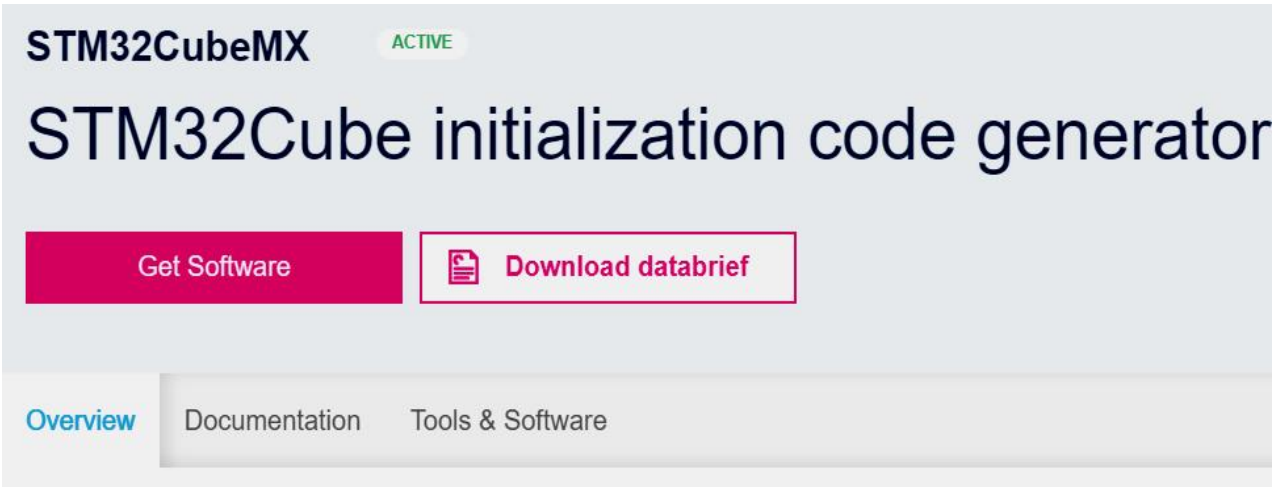



*Рис. 1.1. Фрагмент вебсторінки STMicroelectronics*


*Рис. 1.2. Фрагменти вебсторінки з варіантами завантажень*

1.3. Прийняти ліцензійну угоду, натиснувши “ACCEPT”.

1.4. У діалоговому вікні (рис. 1.3) у відповідних полях набрати свої ім’я, прізвище, адресу електронної поштової скриньки та, поставивши мітку про ознайомлення з умовами продажу, використання і політикою конфіденційності, натиснути кнопку “Download”.

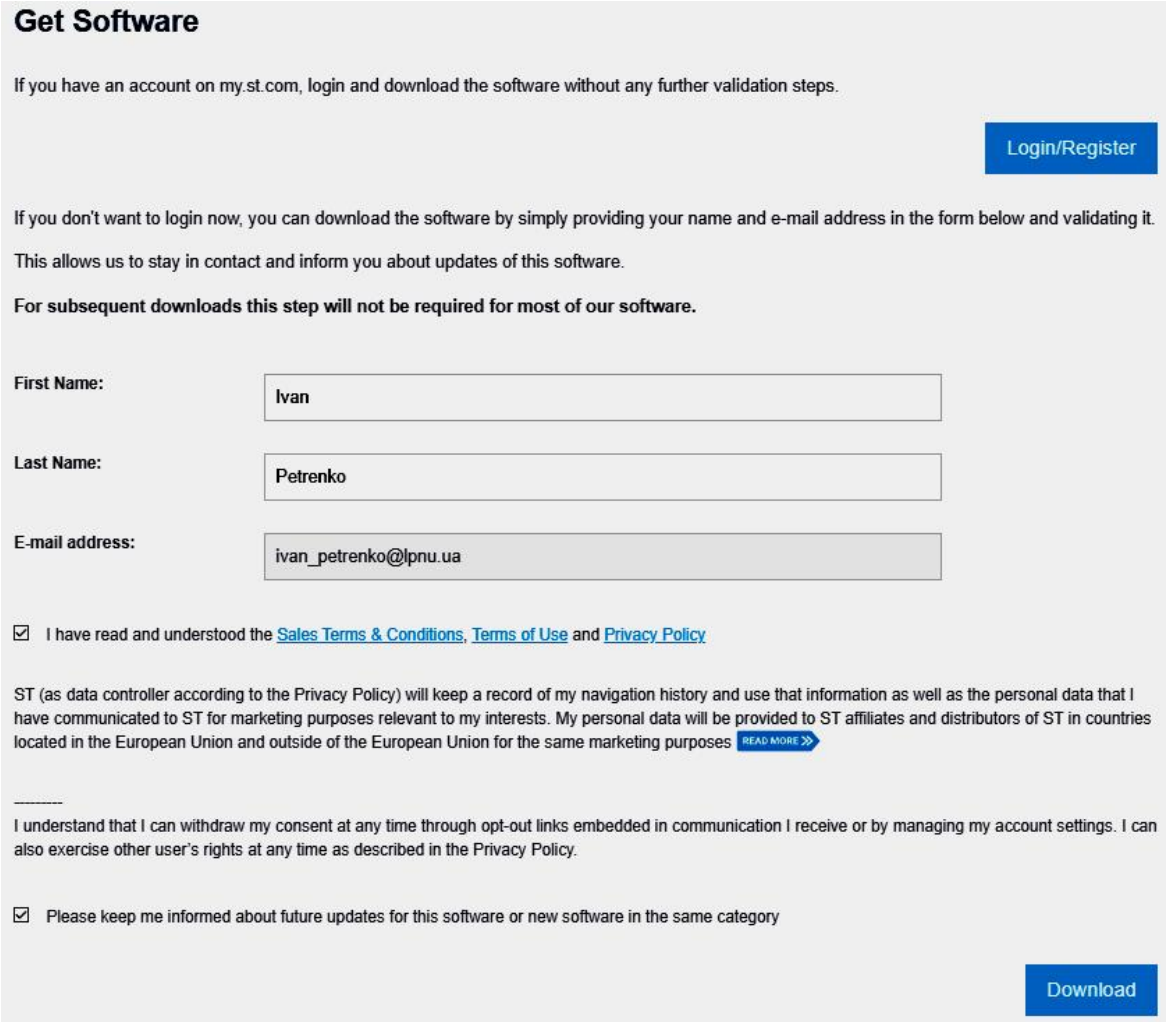



*Рис. 1.3. Діалогове вікно фірми STMicroelectronics з реєстраційними даними користувача*

1.5. Після повідомлення про успішну реєстрацію (рис. 1.4) зайти на свою електронну скриньку, в отриманому листі натиснути “Download now” і завантажити інсталяційний пакет на комп’ютер.

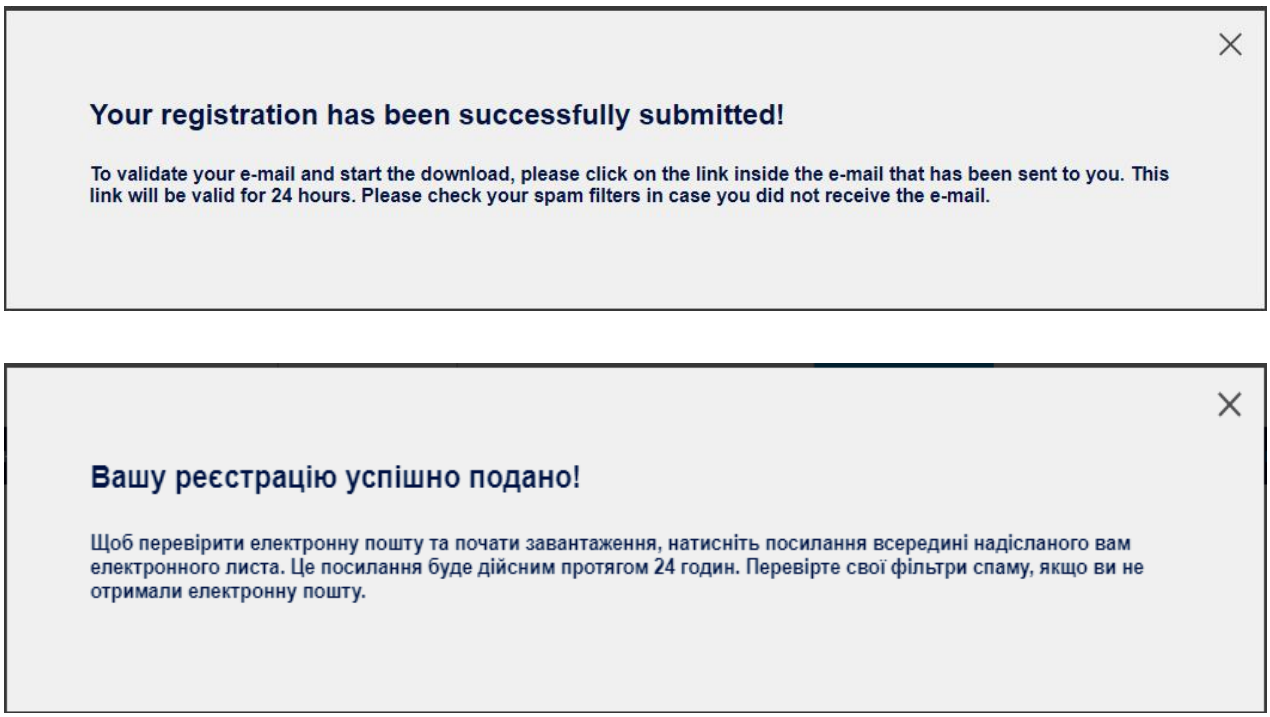



*Рис. 1.4. Повідомлення про успішну реєстрацію*

1.6. Вміст завантаженого інсталяційного архіву поточної версії STM32CubeMX (наприклад, для Windows – en.stm32cubemx-win_v6-2-0.zip) розпакувати у визначений вами каталог, у ньому запустити на виконання EXE-файл (SetupSTM32CubeMX-6.2.0-Win.exe).

1.7. Якщо на комп’ютері не встановлена відповідна версія Java, то необхідно погодитися з показаним на рис. 1.5 повідомленням.

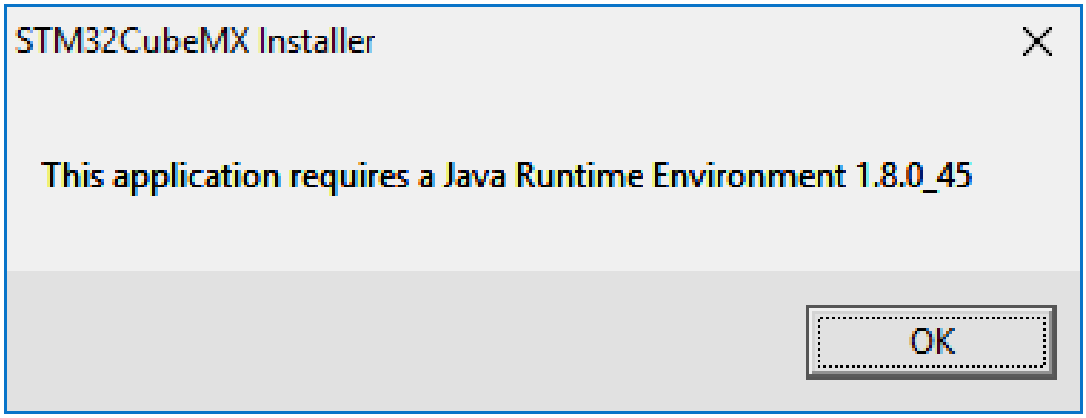



*Рис. 1.5. Повідомлення “Цей додаток вимагає середовища виконання Java”*

Відкриється типовий для операційної системи браузер з домашньою сторінкою Java (рис. 1.6), з якої можна завантажити інсталяційний пакет і встановити на комп'ютер.


*Рис. 1.6. Фрагмент вебсторінки Java*

Якщо при встановленні Java операційна система повідомляє про якусь помилку, то варто на вебсторінці Java зліва внизу знайти посилання "Trouble downloading? Try the offline installer" ("Проблеми із завантаженням? Спробуйте встановити офлайн") – на рис. 1.6 виділено синім кольором (див. вище). Натискання на ньому дозволить завантажити повний пакет для встановлення Java.

Ще один спосіб – унизу сторінки Java знайти рядок "Not the right operating system? See all Java downloads" ("Не та операційна система? Переглянути всі завантаження Java") – на рис. 1.7 виділено синім кольором. Перейти за цим посиланням і в наступному вікні (рис. 1.8) вибрати необхідний інсталяційний пакет Java під наявну операційну систему та завантажити його на комп'ютер.

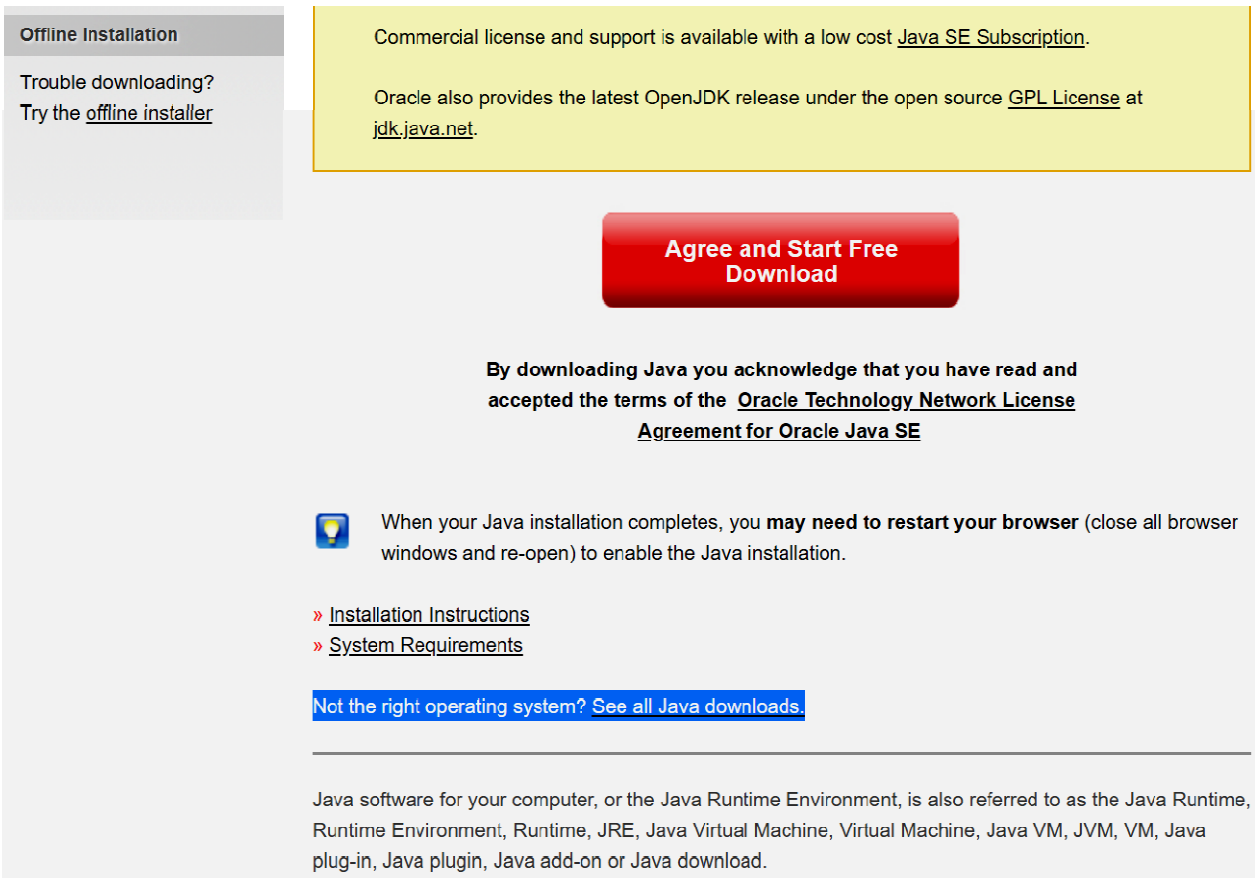



*Рис. 1.7. Рядок“Не та операційна система?”*
*(фрагмент вебсторінки Java)*

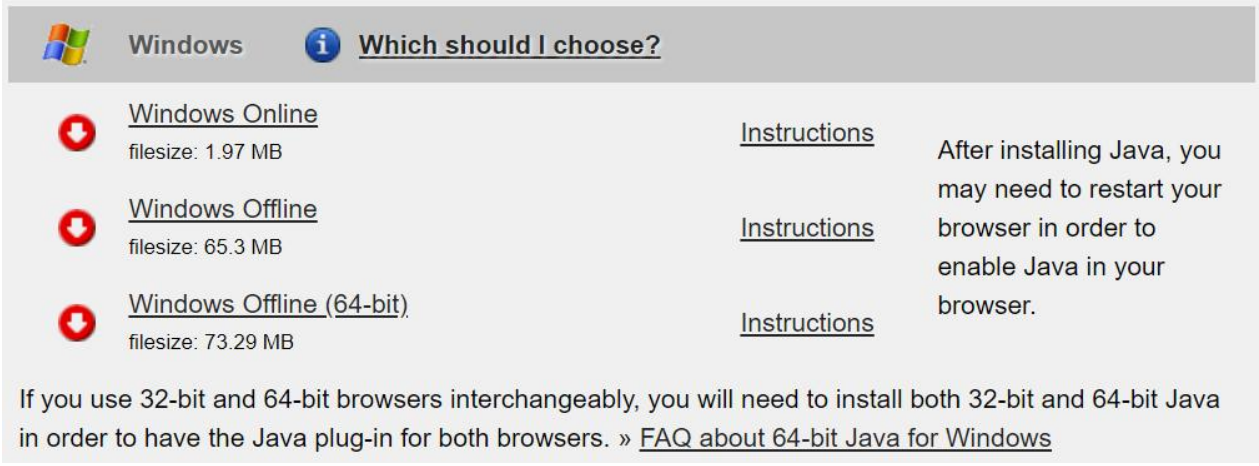



*Рис. 1.8. Вибір варіанта завантаження*
*(фрагмент вебсторінки Java)*

1.8. Встановлення Java відбувається у три етапи (рис. 1.9–1.11).

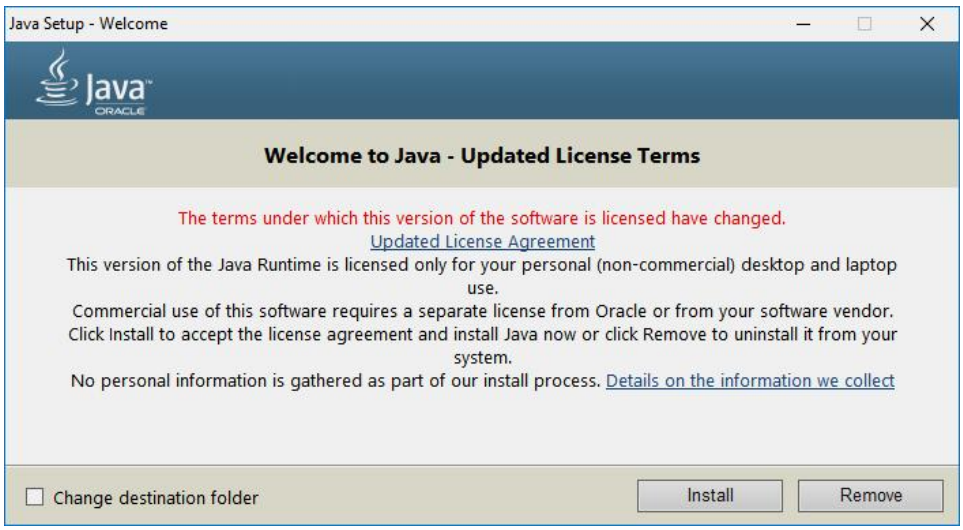



*Рис. 1.9. Ініціалізація встановлення Java*


*Рис. 1.10. Процес встановлення Java*

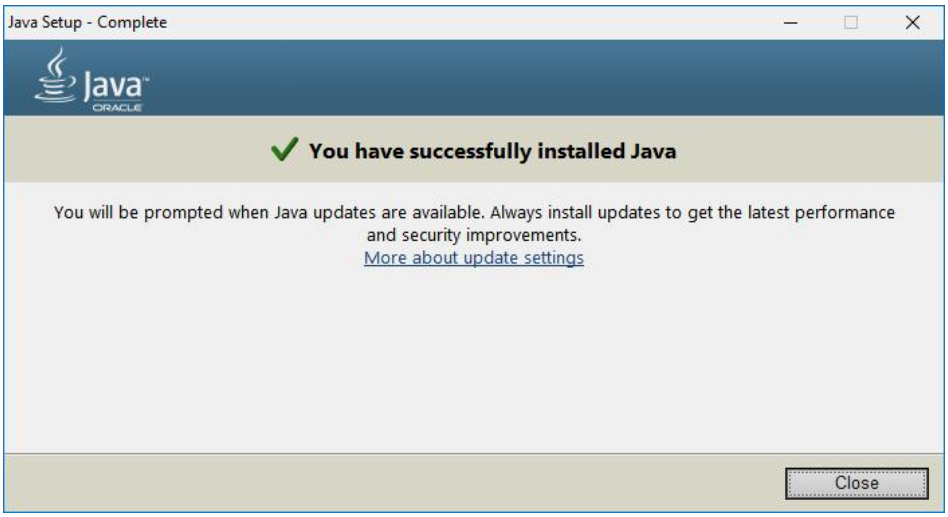



*Рис. 1.11. Успішне встановлення Java*

1.9. Тепер можна продовжити встановлення STM32CubeMX (рис. 1.12).

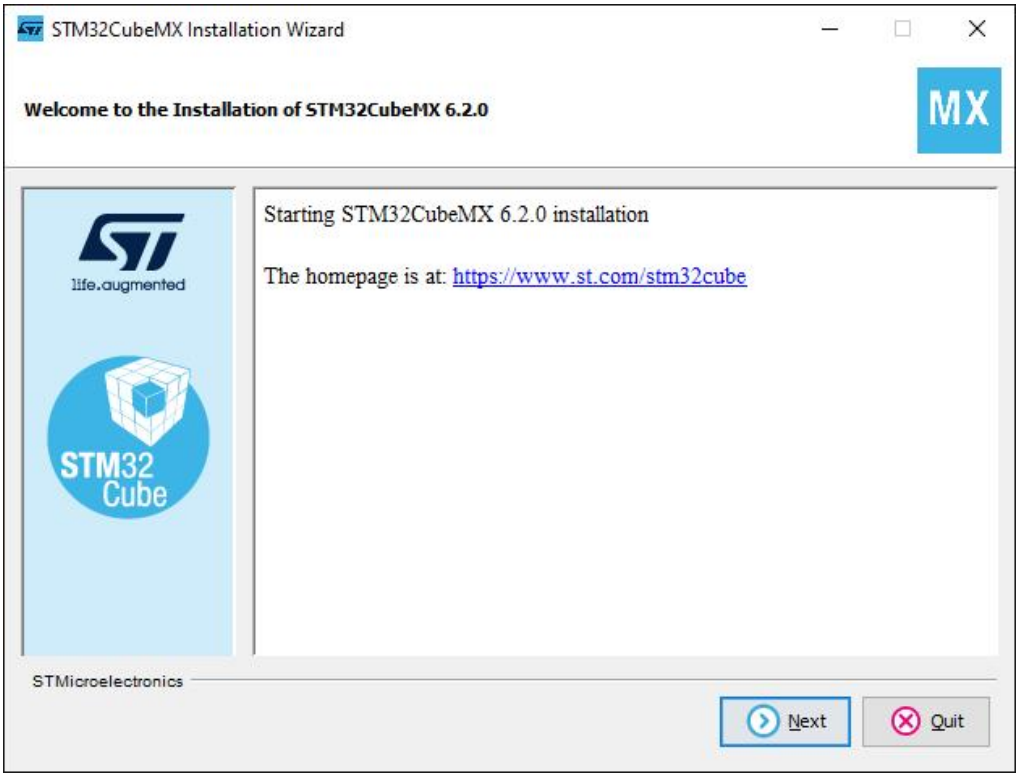



*Рис. 1.12. Ініціалізація встановлення STM32CubeMX*

1.10. Погодитися з умовами ліцензування STM32CubeMX (рис. 1.13).

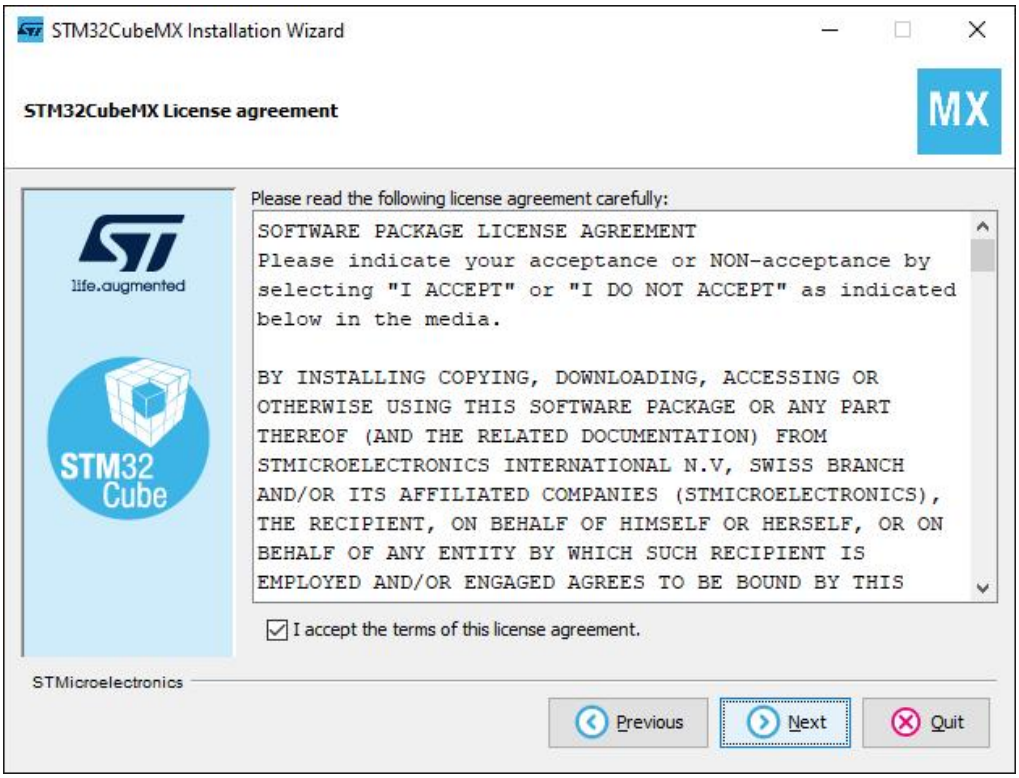



*Рис. 1.13. Умови ліцензування STM32CubeMX*

1.11. Поставити мітки на запитах щодо конфіденційності та умов використання і щодо дозволу на збирання статистичної інформації про використання функцій програмного продукту STMicroelectronics (рис. 1.14).

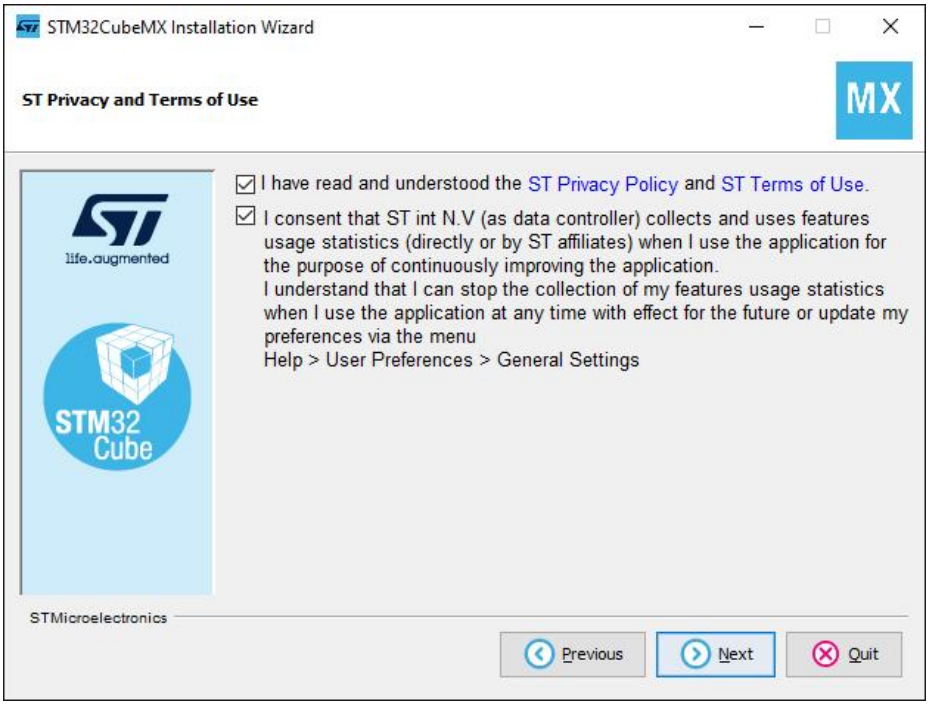



*Рис. 1.14. Вікно погодження з умовами* STMicroelectronics

1.12. Вибрати місце на диску, куди буде встановлюватись STM32CubeMX (рис. 1.15).

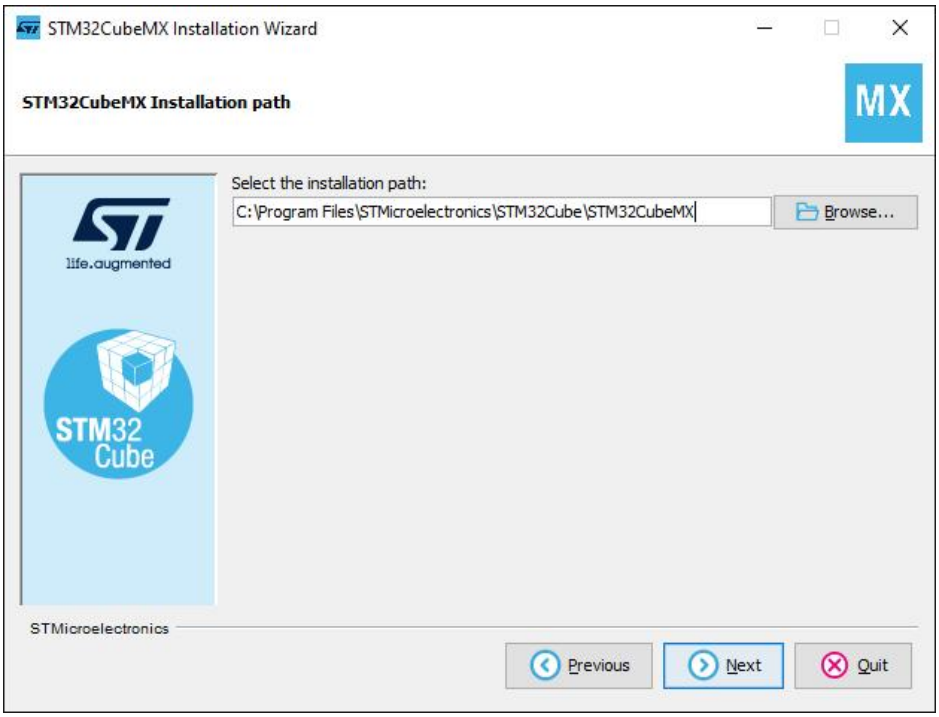



*Рис. 1.15. Розміщення програмного продукту STM на комп'ютері*

1.13. Погодитися на створення цільового каталогу (рис. 1.16).

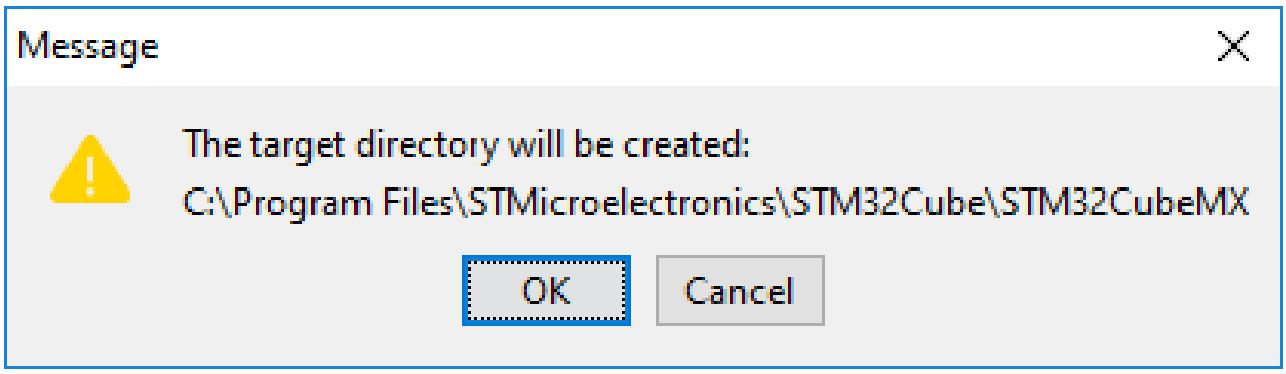



*Рис. 1.16. Повідомлення щодо цільового каталогу*

1.14. Визначитися з наявними користувачами (рис. 1.17).

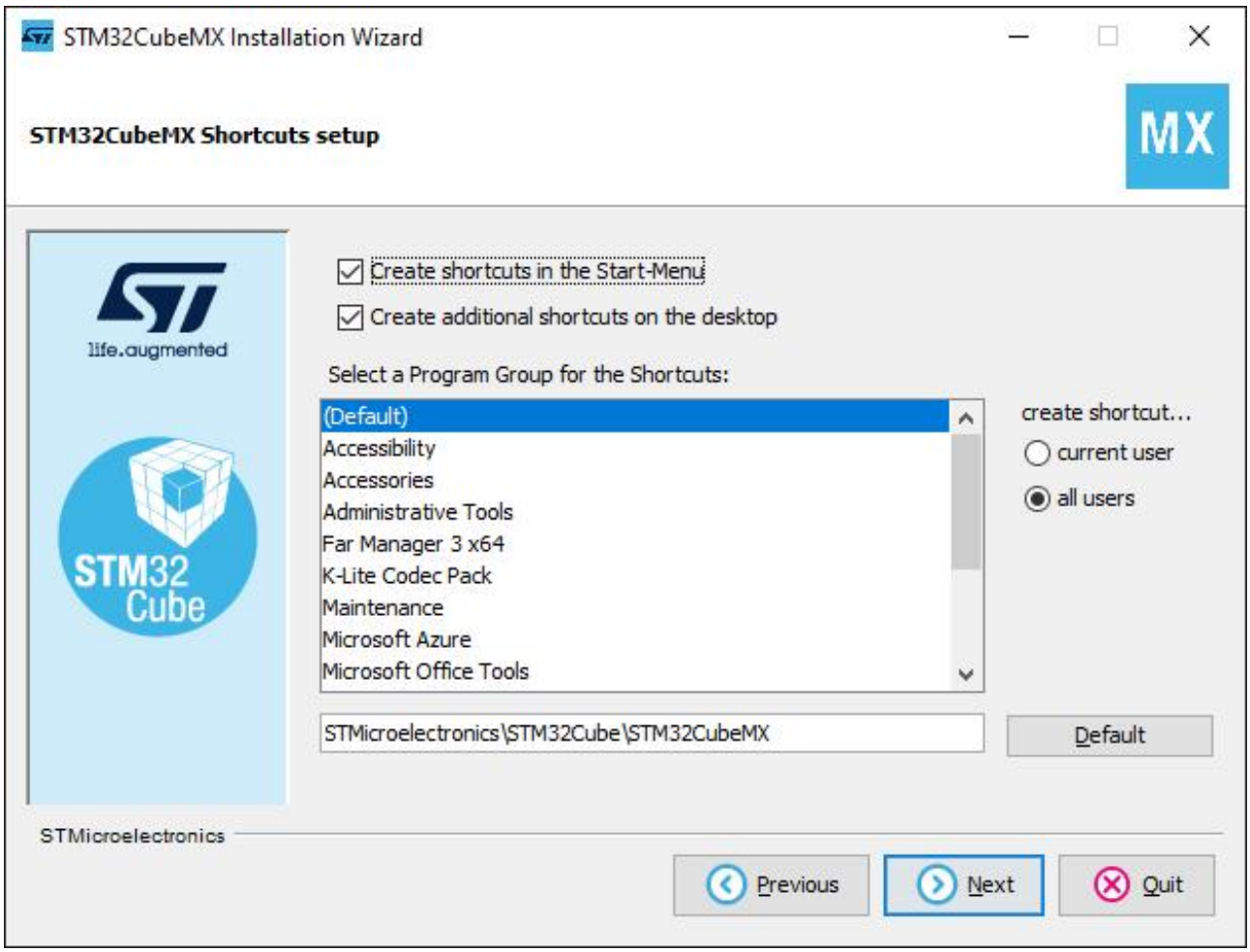



*Рис. 1.17. Доступність для всіх користувачів*

1.15. Спостерігати за роботою інсталятора (рис. 1.18).

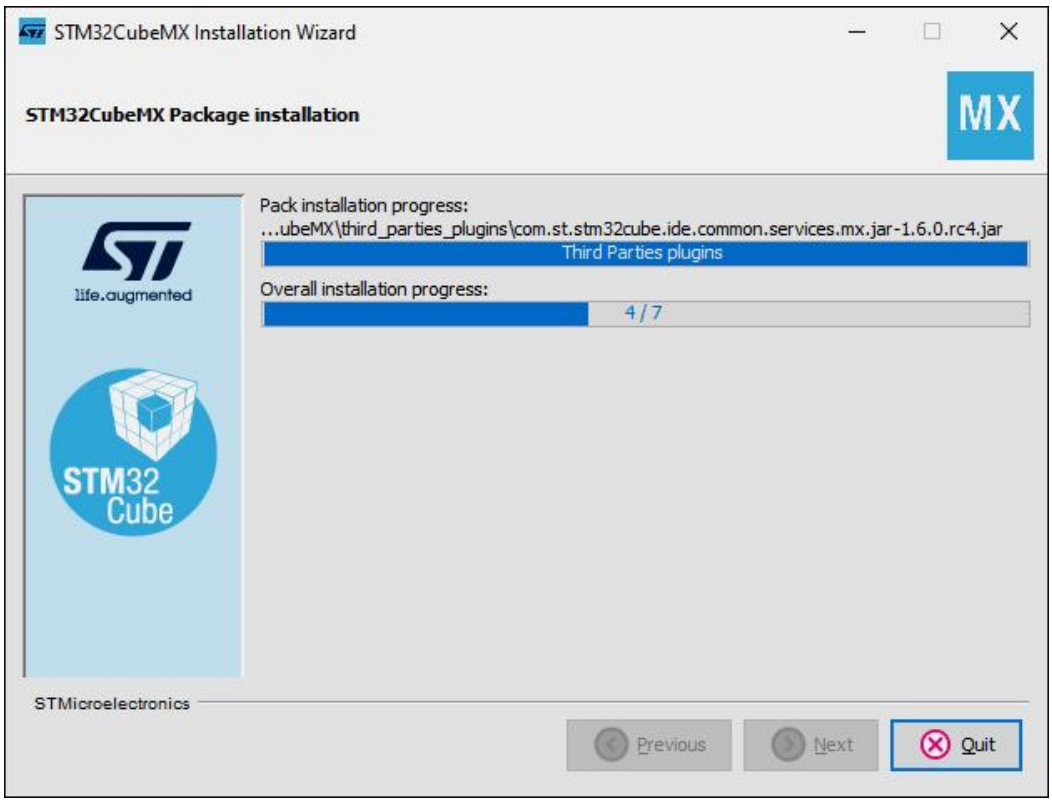



*Рис. 1.18. Процес встановлення STM32CubeMX*

1.16. Завершити роботу інсталятора (рис. 1.19).

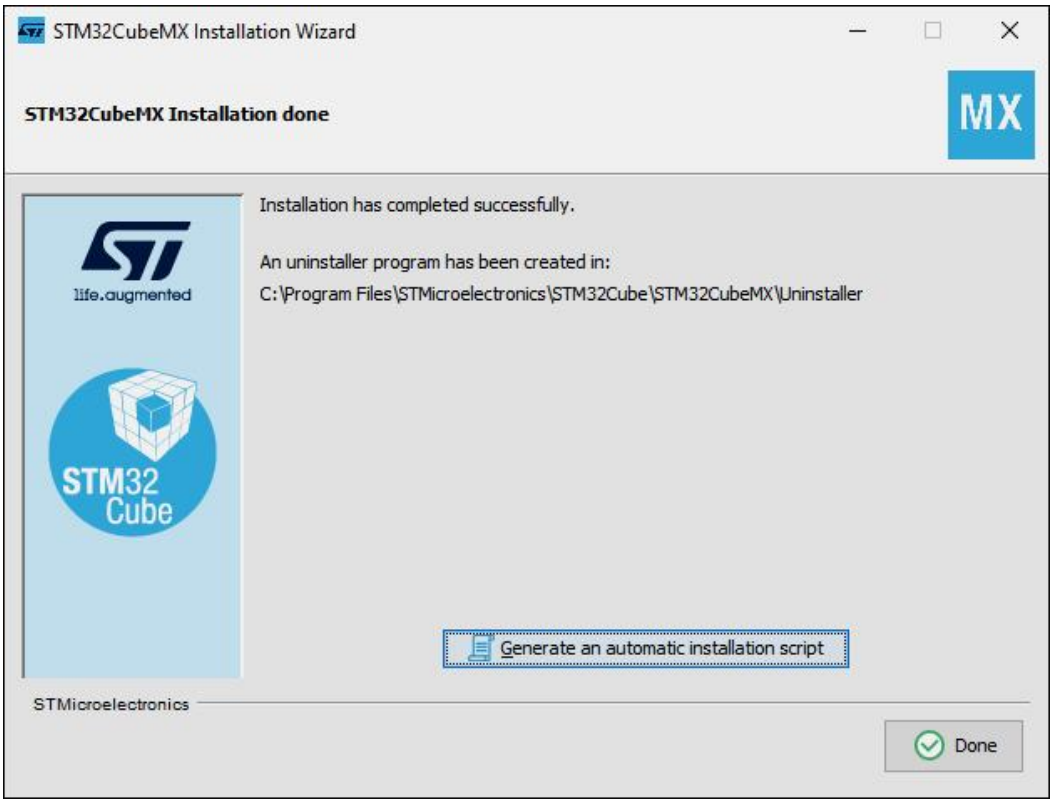



*Рис. 1.19. Успішне встановлення STM32CubeMX*

1.17. Погодитися із запитом щодо допомоги для вдосконалення продукції фірми STMicroelectronics (рис. 1.20).

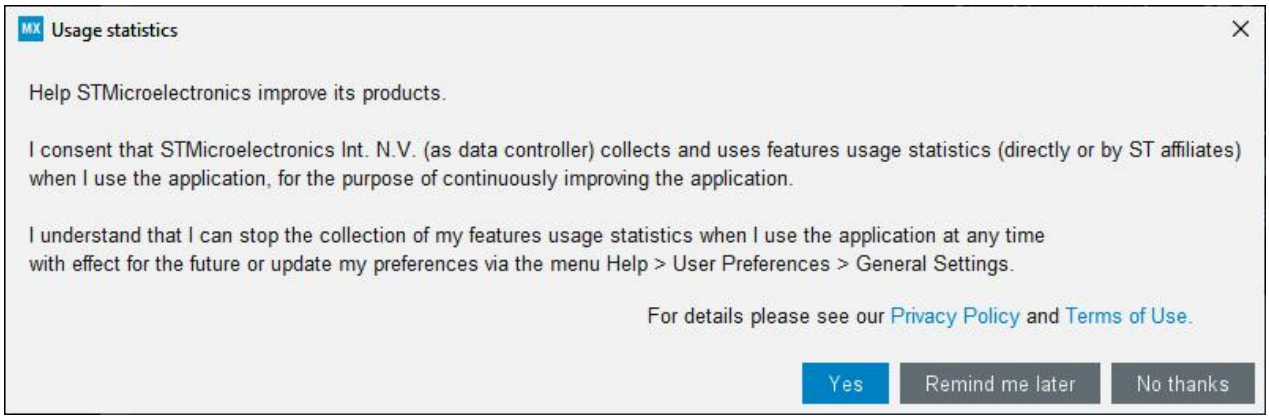



*Рис. 1.20. Запит від фірми-виробника*

1.18. Впевнитися, що STM32CubeMX запускається (рис. 1.21).

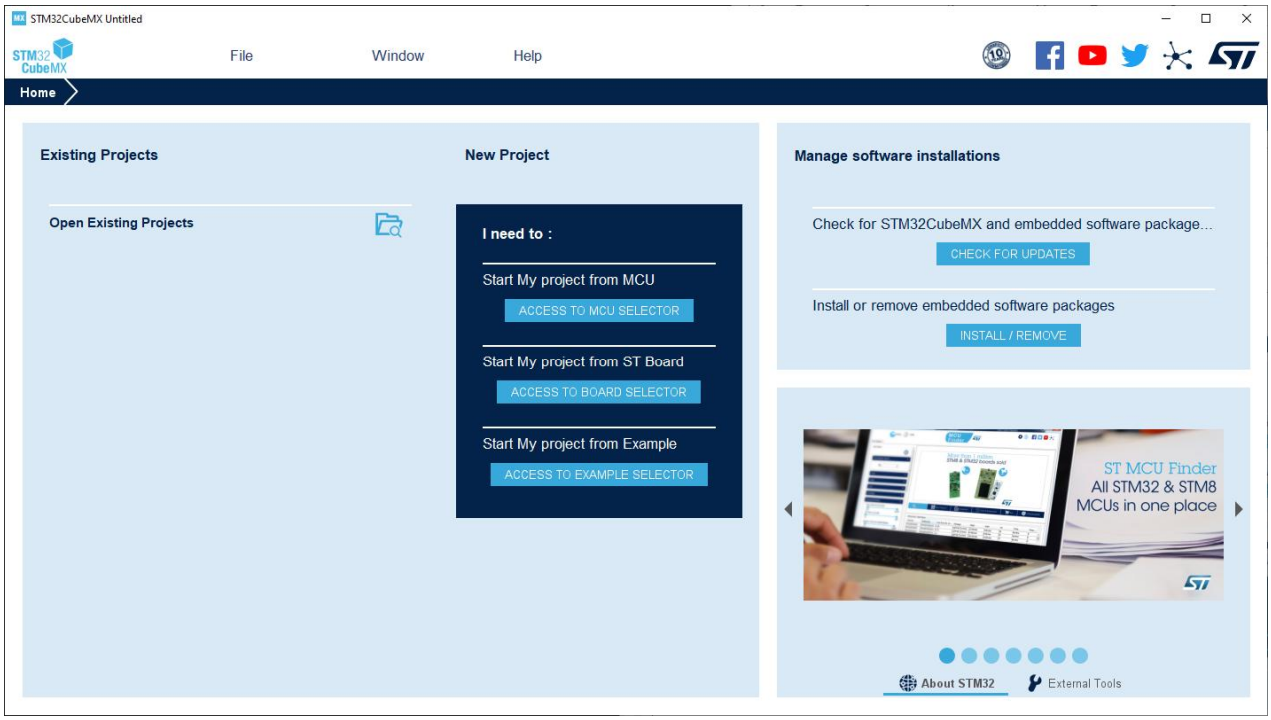



*Рис. 1.21. Стартове вікно STM32CubeMX*

1.19. Закрити STM32CubeMX: File > Exit (Ctrl+X, Alt+F4).

## 2. Встановлення Keil MDK-ARM

2.1. У пошуковому полі браузера набрати: Keil. В отриманому результаті натиснути на посиланні “MDK” (рис. 2.1).

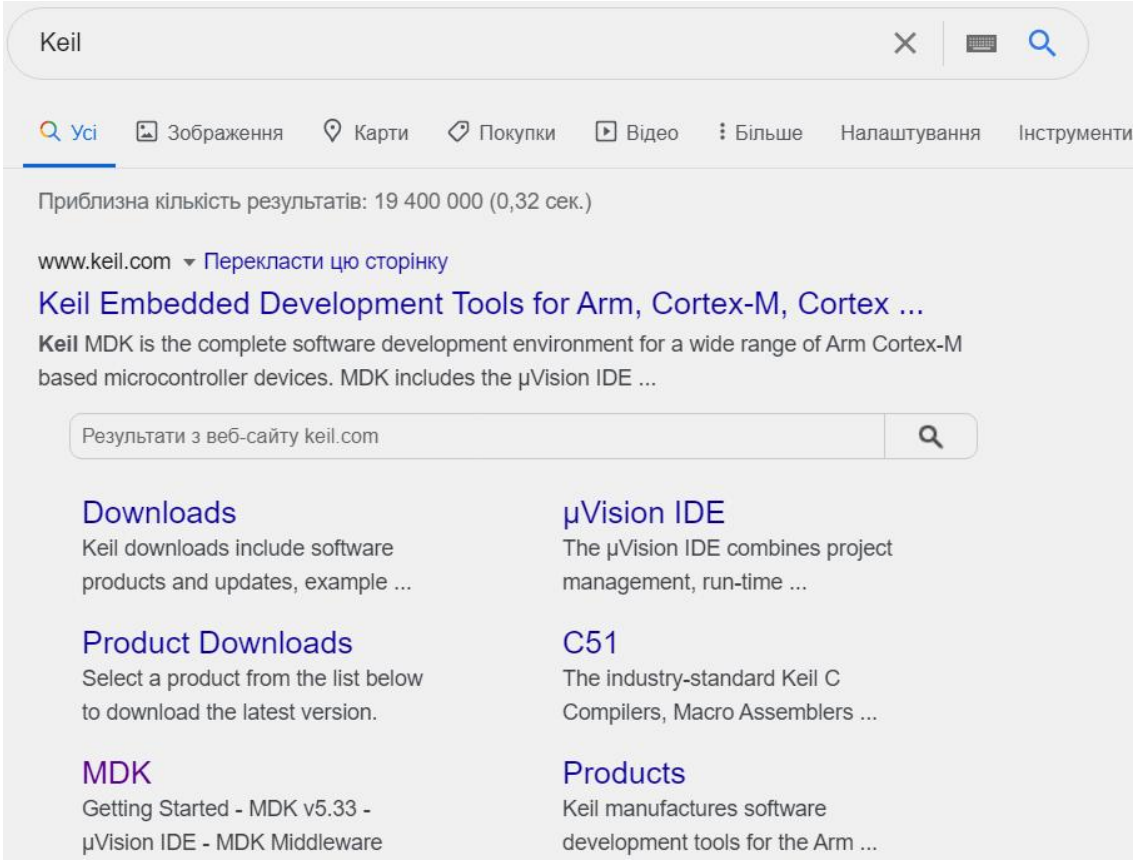



*Рис. 2.1. Результат пошуку слова “Keil” (фрагмент екрана)*

2.2. На вебсторінці “MDK Microcontroller Development Kit” фірми Keil натиснути кнопку “Download MDK” (рис. 2.2).

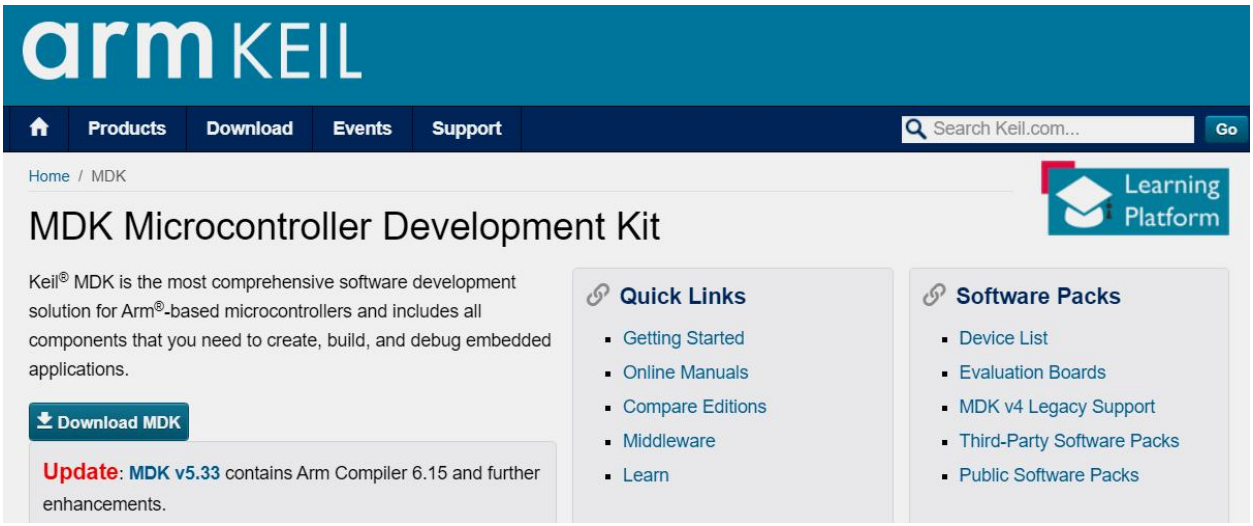



*Рис. 2.2. Фрагмент вебсторінки фірми Keil*

2.3. Сформувати запит для оцінювання програмного забезпечення MDK-ARM. Для цього в діалоговому вікні (рис. 2.3) у відповідних полях набрати свої ім'я, прізвище, адресу електронної поштової скриньки, а також назву чи абревіатуру організації, вибрати зі списку країну, набрати номер свого телефону, вказати сімейство мікроконтролерів STM32 та натиснути кнопку "Submit".

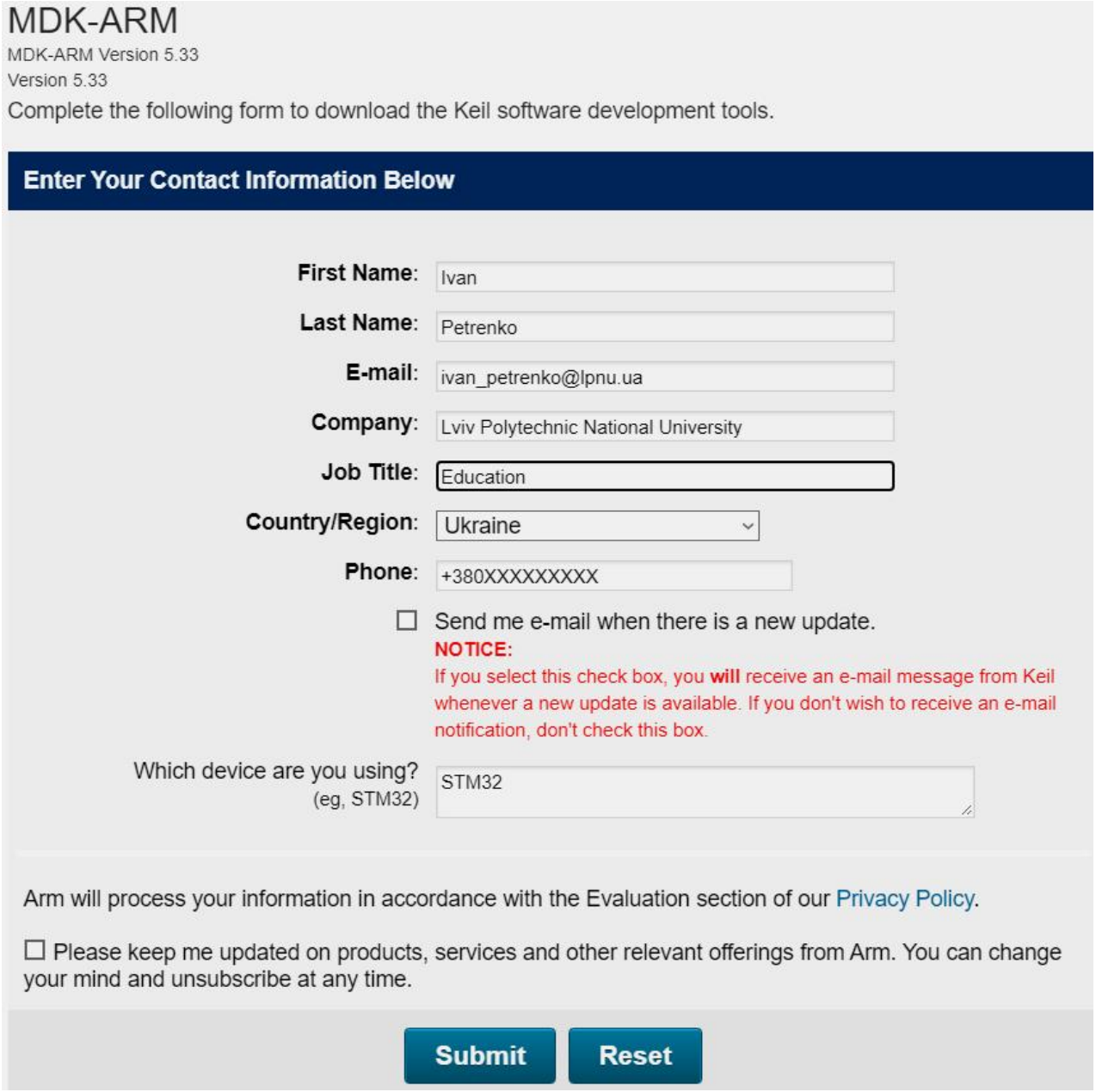



*Рис. 2.3. Діалогове вікно фірми Keil з реєстраційними даними користувача*

2.4. Ознайомитися з вимогами до апаратного та програмного забезпечення (рис. 2.4) і натиснути на виділеній кольором поточній версії "MDK533.EXE". Розмір файлу 945 МБ. Завантажити його на комп'ютер.

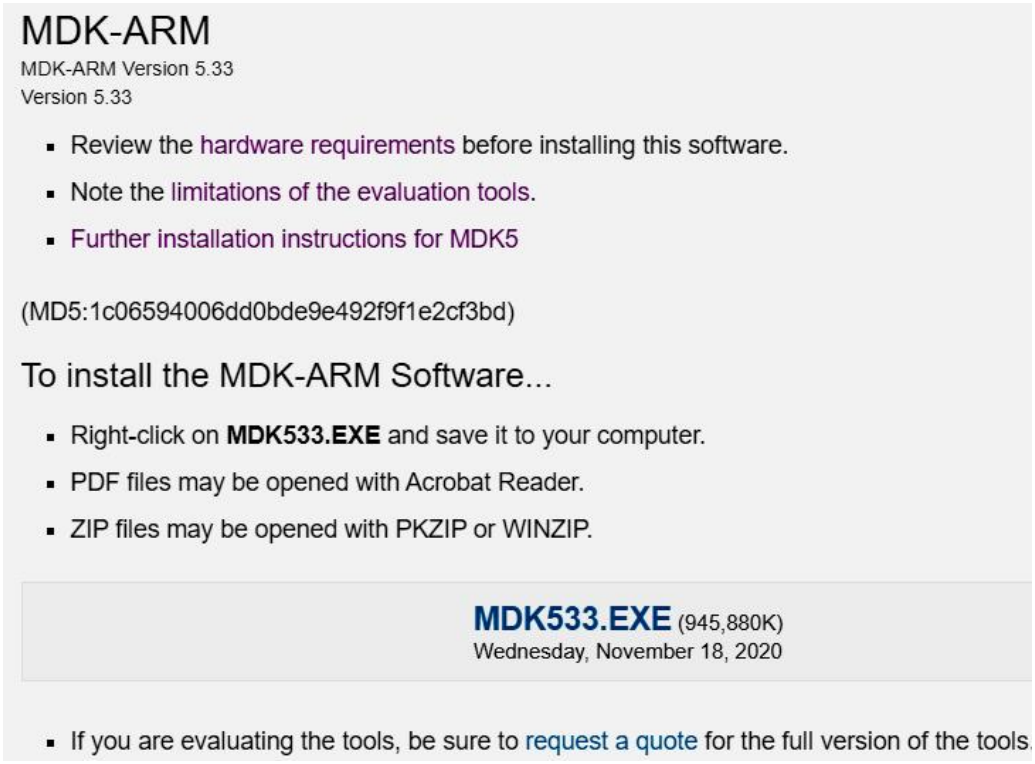



*Рис. 2.4. Вимоги до обладнання*

2.5. Завантажений файл MDK533.EXE запустити на встановлення (рис. 2.5) і виконувати подальші кроки.

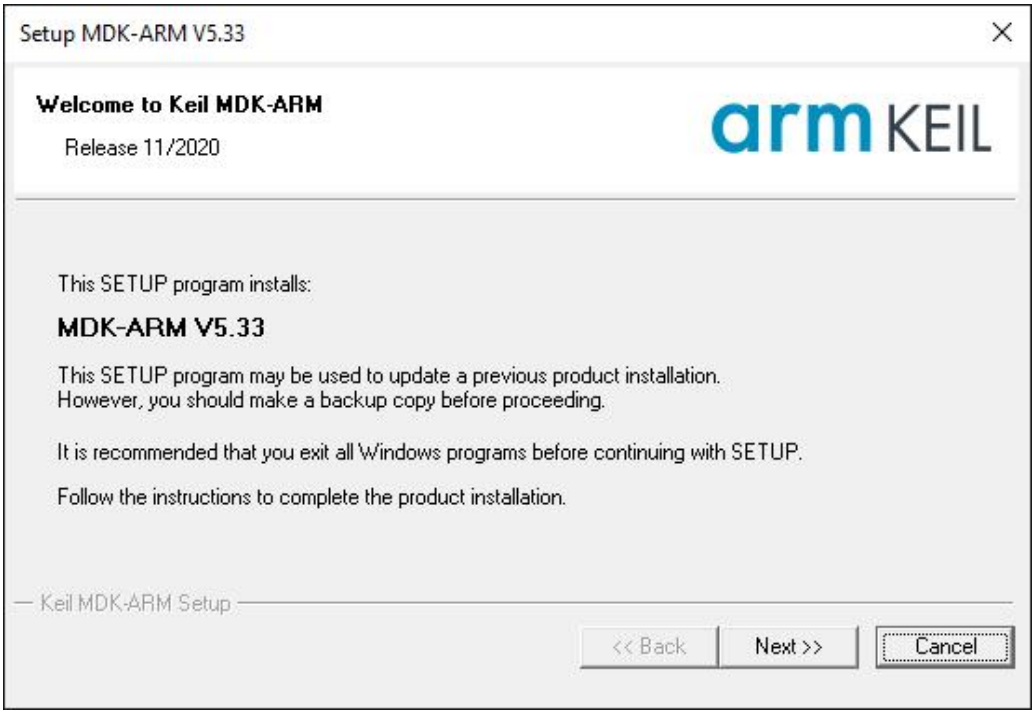



*Рис. 2.5. Ініціалізація встановлення Keil MDK-ARM*

2.6. Погодитися з умовами ліцензування Keil MDK-ARM (рис. 2.6).

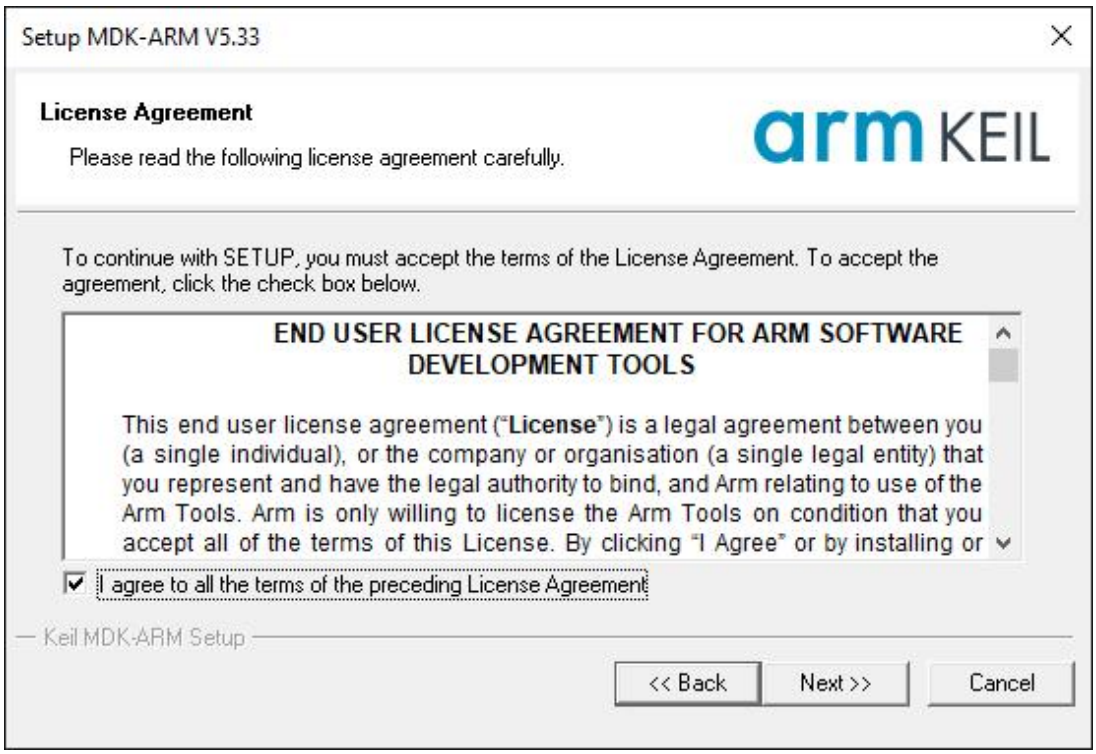



*Рис. 2.6. Умови ліцензування Keil MDK-ARM*

2.7. Вибрати місце на диску, куди буде встановлюватися Keil MDK-ARM (рис. 2.7).

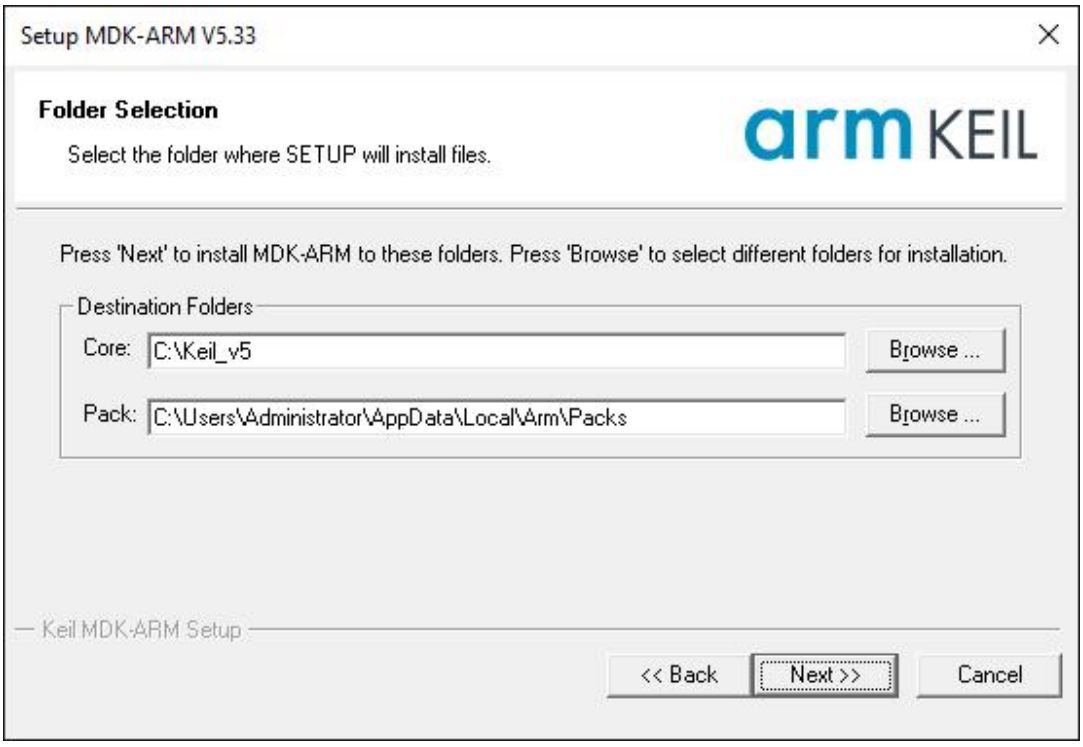



*Рис. 2.7. Розміщення програмного продукту Keil на комп'ютері*

2.8. У відповідних полях набрати свої імя, прізвище, адресу електронної поштової скриньки, а також назву організації (рис. 2.8).

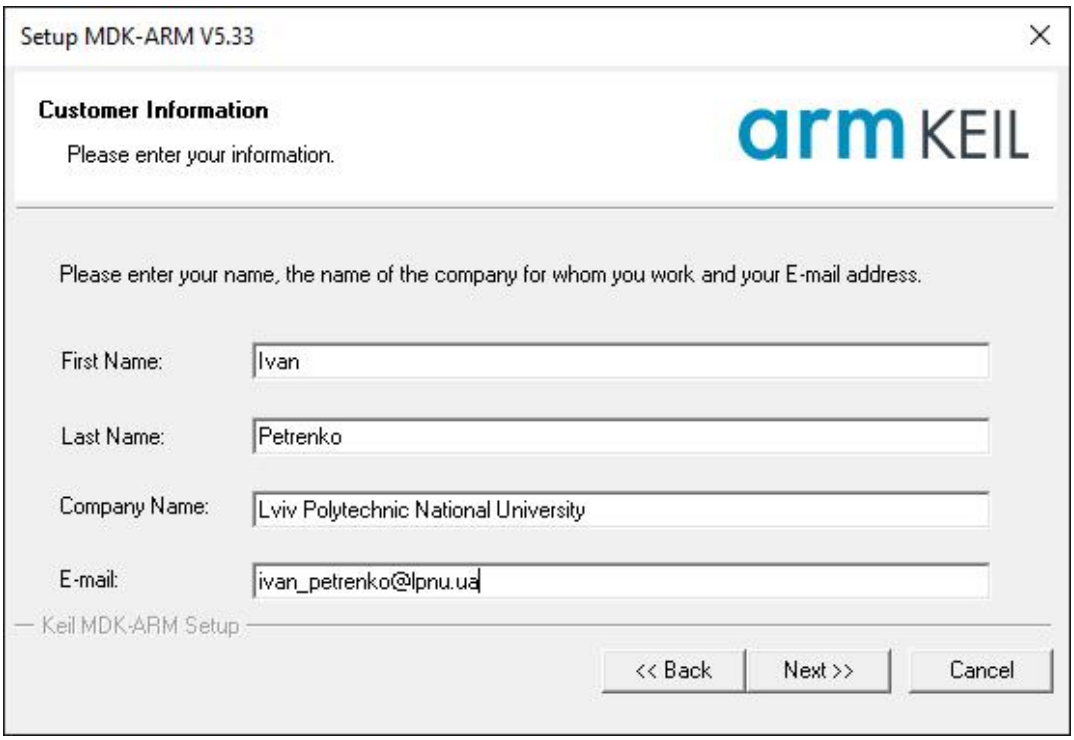



*Рис. 2.8. Реєстраційні дані для встановлення*

2.9. Спостерігати за роботою інсталятора (рис. 2.9).

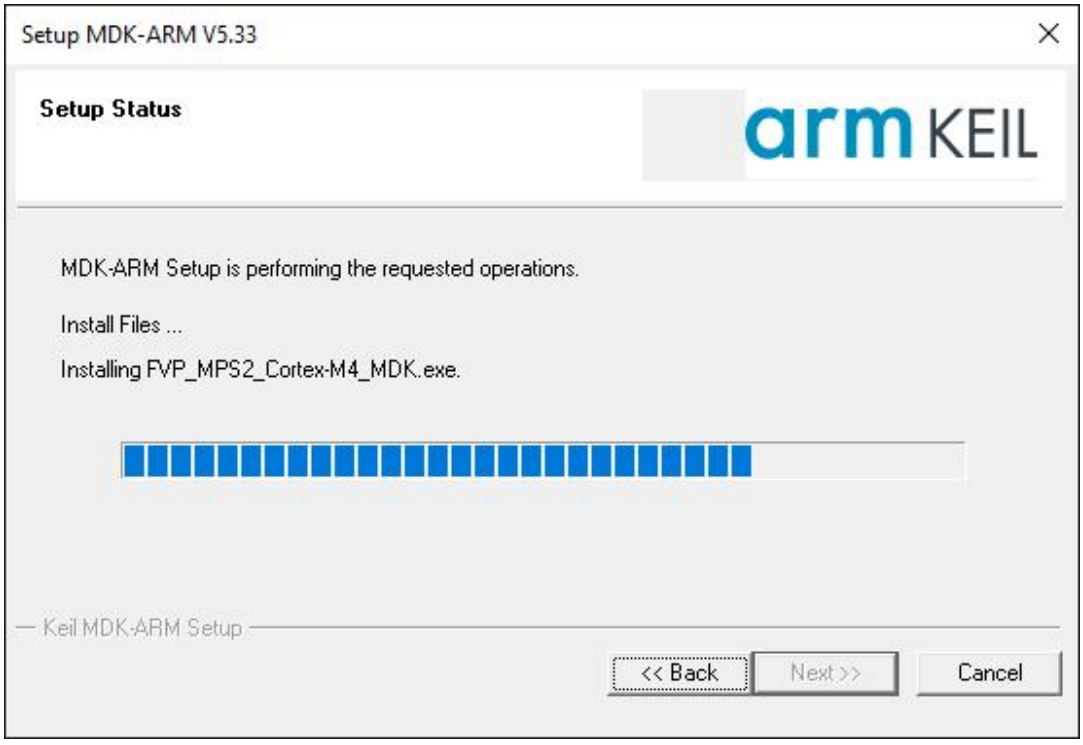



*Рис. 2.9. Процес встановлення Keil MDK-ARM*

2.10. Запевнити операційну систему, що складники Keil MDK-ARM не є для неї загрозою. Для цього натиснути кнопку “Інсталяція” (рис. 2.10).

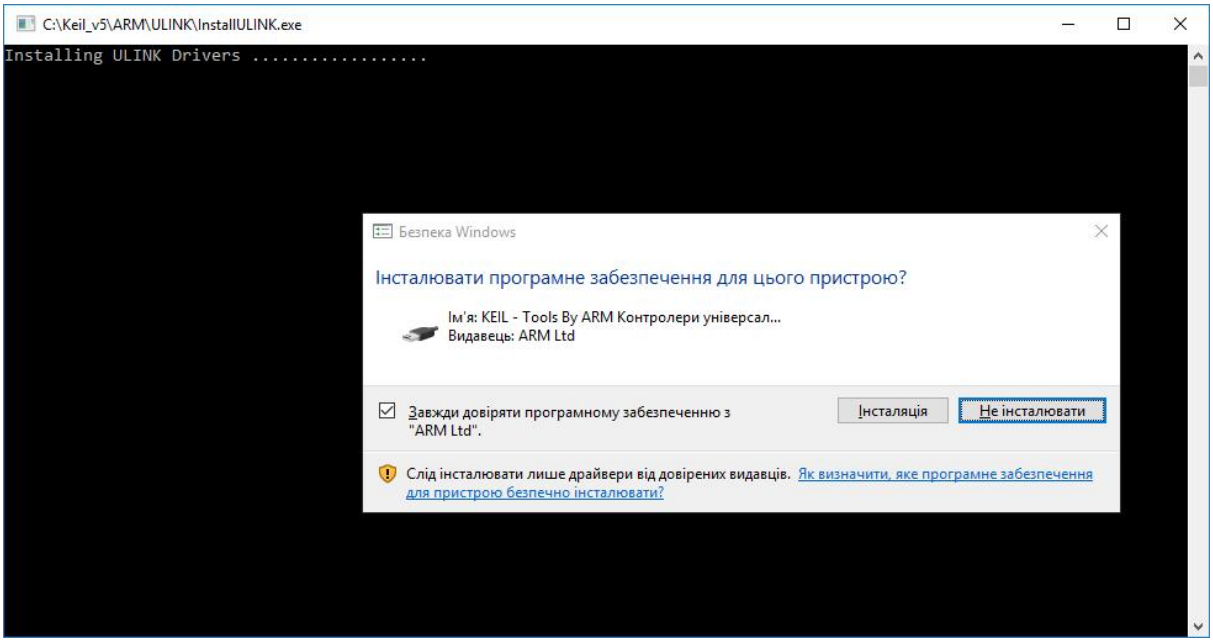



*Рис. 2.10. Запит від системи безпеки Windows*

2.11. Завершити роботу інсталятора (рис. 2.11).

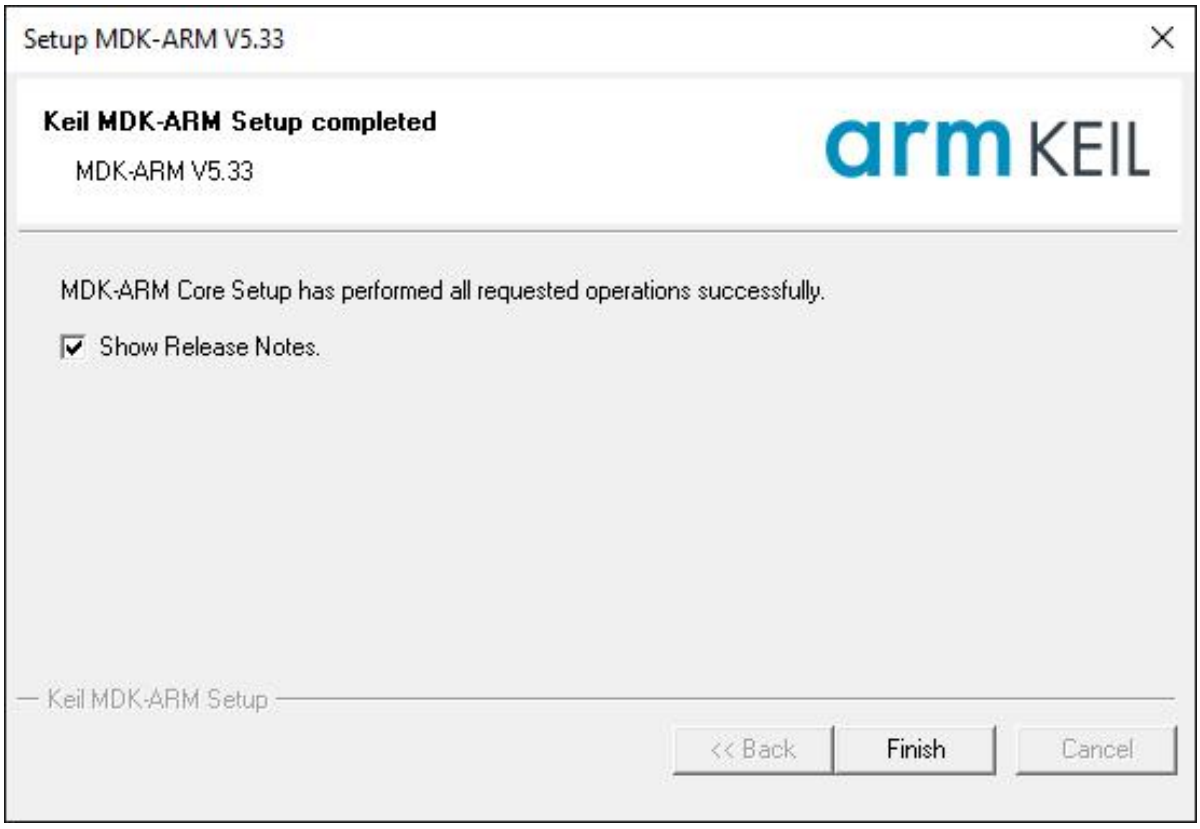



*Рис. 2.11. Успішне встановлення Keil MDK-ARM*

2.12. Після встановлення запуститься менеджер пакетів “Pack Installer” (рис. 2.12). Погодитися на роботу з ним.

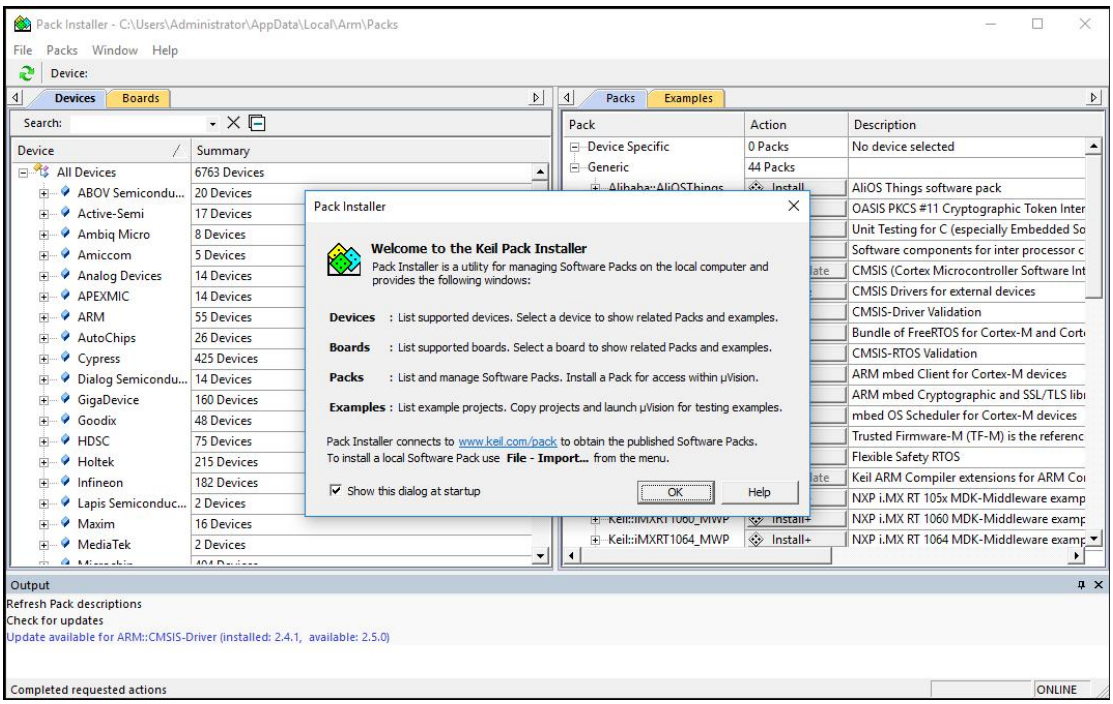



*Рис. 2.12. Вікно запрошення до Pack Installer*

2.13. На вкладці “Devices” знайти рядок “STMicroelectronics” і, послідовно розгортаючи дерево каталогів, знайти та вибрати мікроконтролер STM32F407VGTx (рис. 2.13).

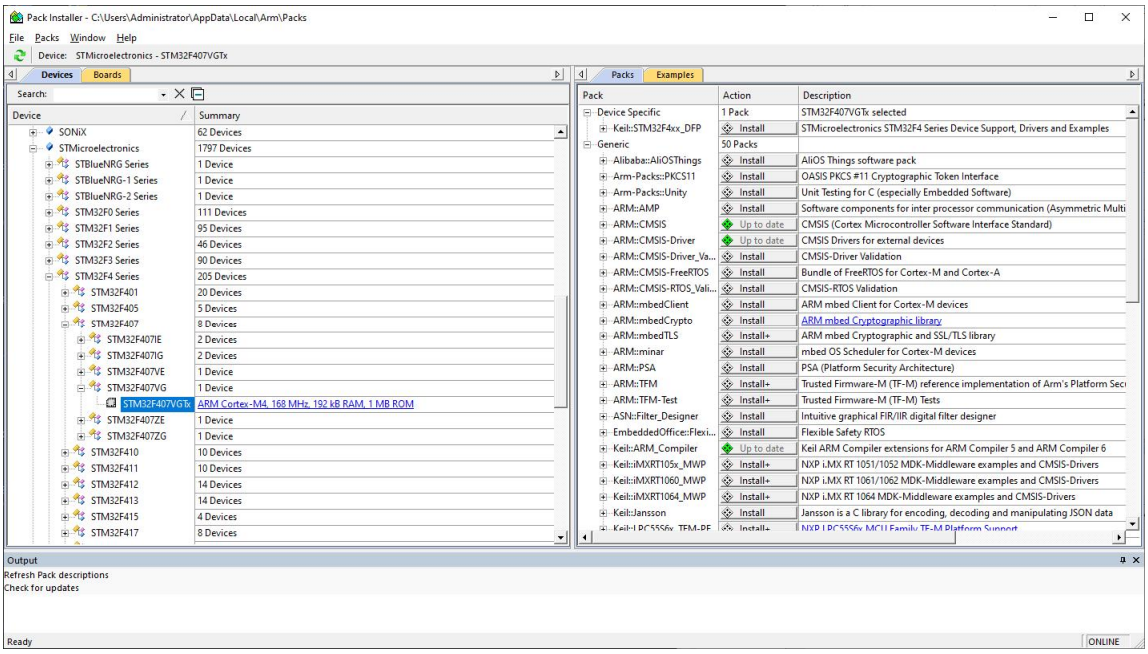



*Рис. 2.13. Вибір мікроконтролера*

2.14. На вкладці “Packs” виділити рядок “Keil::STM32F4xx_DFP” і справа від нього натиснути кнопку “Install”. У правому нижньому куті “Pack Installer” можна спостерігати за процесом встановлення необхідних для вибраного мікроконтролера програмних пакетів (рис. 2.14).

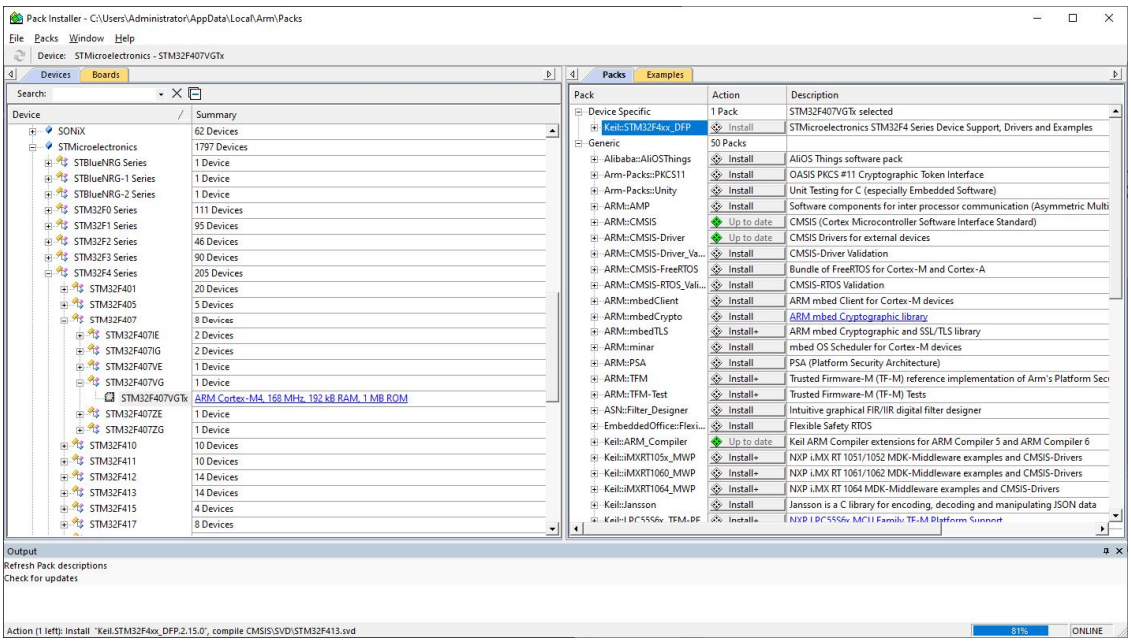



*Рис. 2.14. Завантаження і встановлення пакетів*

2.15. Запустити Keil µVision  , який є складником інтегрованого середовища розробки MDK-ARM (рис. 2.15).

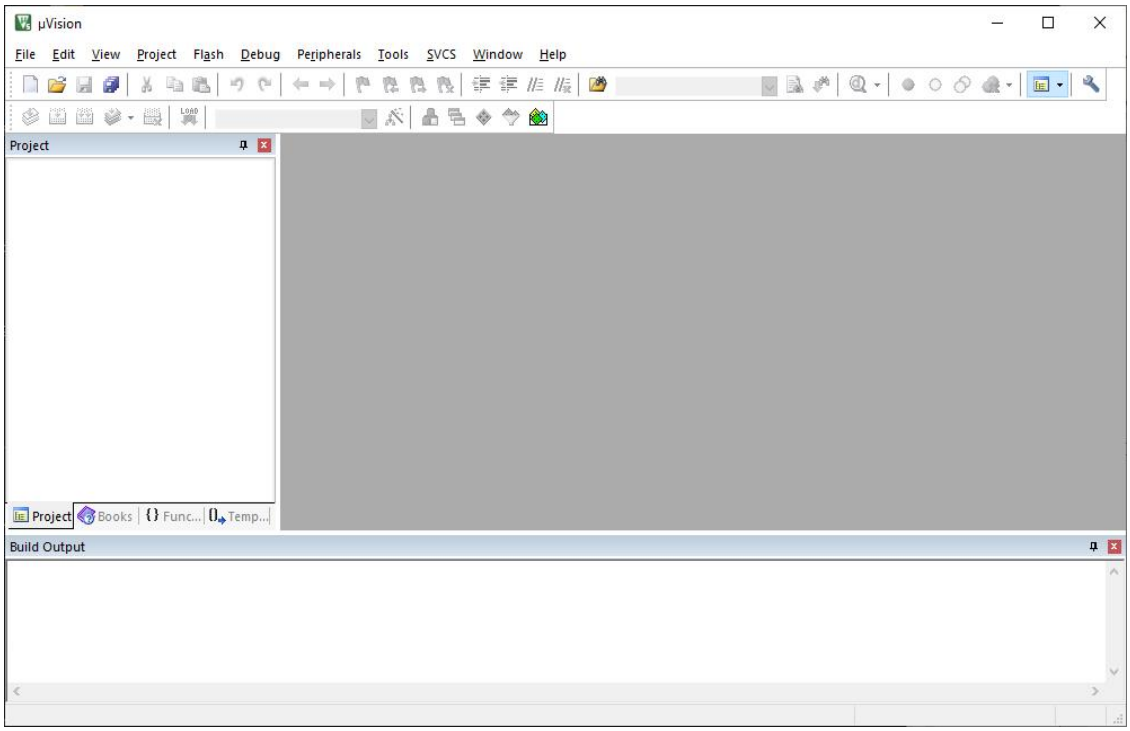



*Рис. 2.15. Стартове вікно Keil µVision*

2.16. Якщо необхідно налаштуватись на роботу з іншим мікроконтролером чи мікропроцесором, то запустити менеджер пакетів можна з Keil µVision, натиснувши піктограму на панелі інструментів, чи пройшовши в меню шлях Project > Manage > Pack Installer (рис. 2.16).

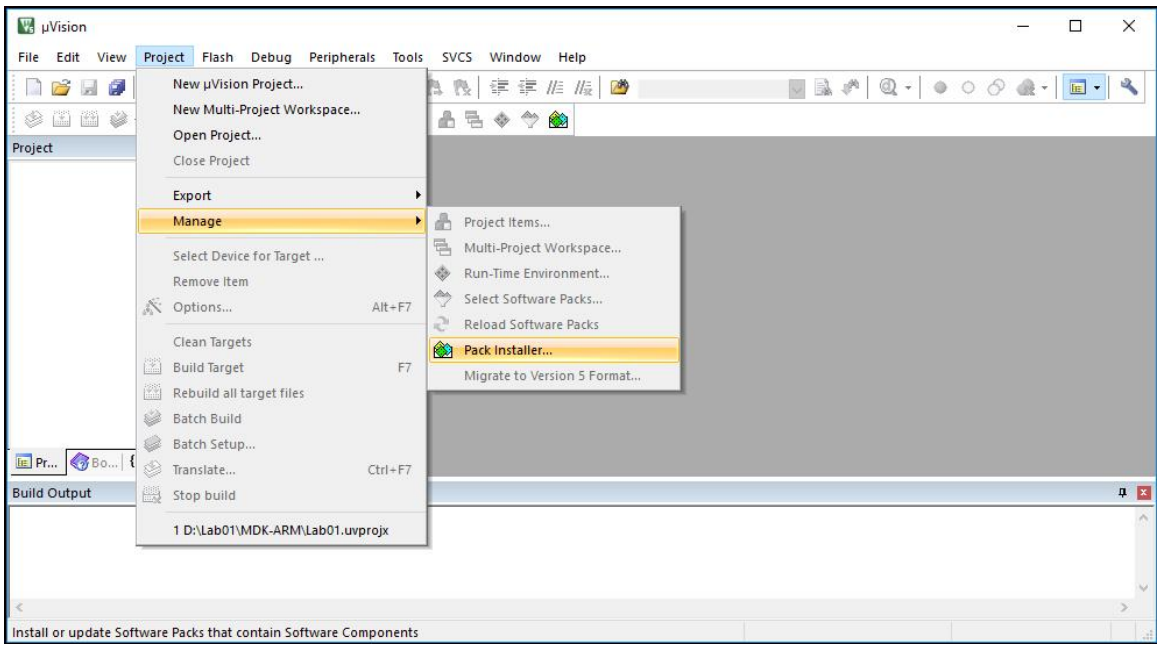



*Рис. 2.16. Запуск менеджера пакетів з Keil µVision*

2.17. Створити новий проєкт через меню Project > New µVision Project. При цьому визначитись з робочим каталогом і назвою проєкту (рис. 2.17).

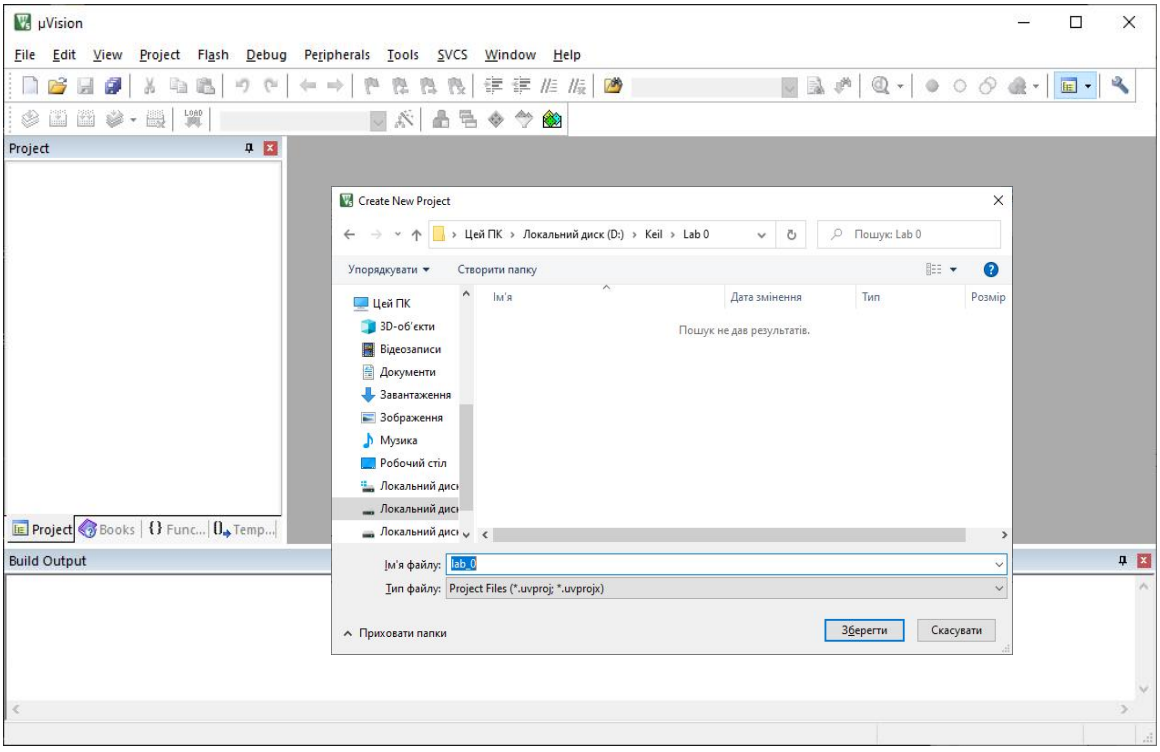



*Рис. 2.17. Створення проєкту в Keil µVision*

2.18. У вікні вибору пристрою (рис. 2.18) послідовно розгорнути дерево каталогів STMicroelectronics, знайти та вибрати мікроконтролер STM32F407VGTx (рис. 2.19).

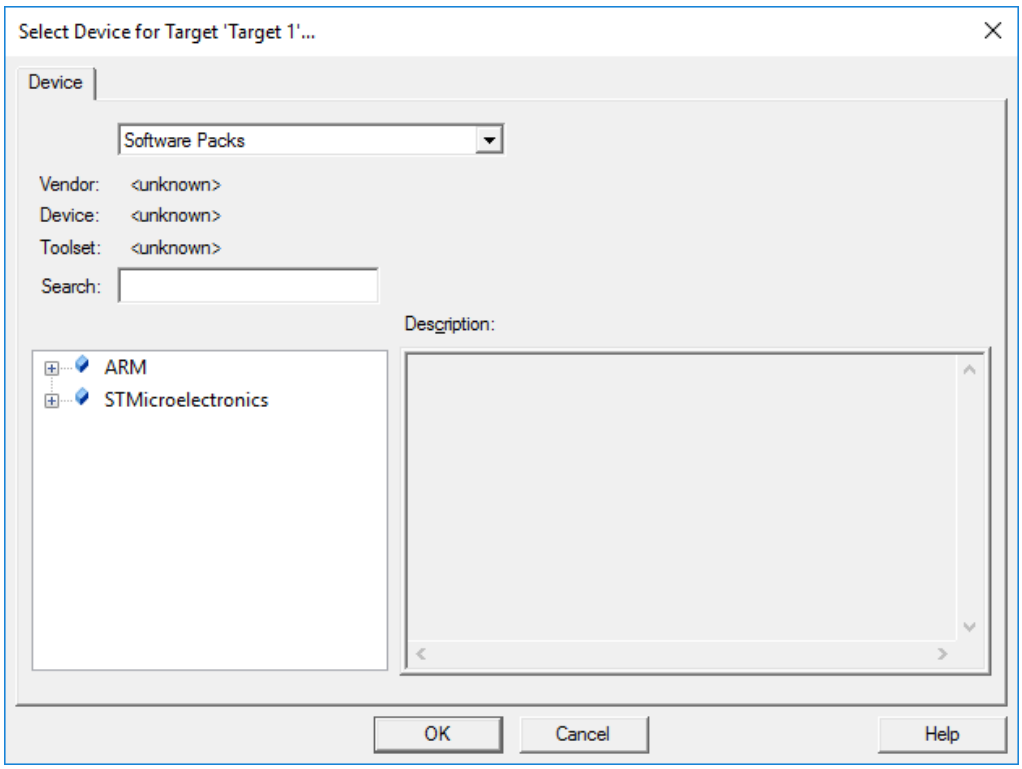



*Рис. 2.18. Вікно вибору пристрою*

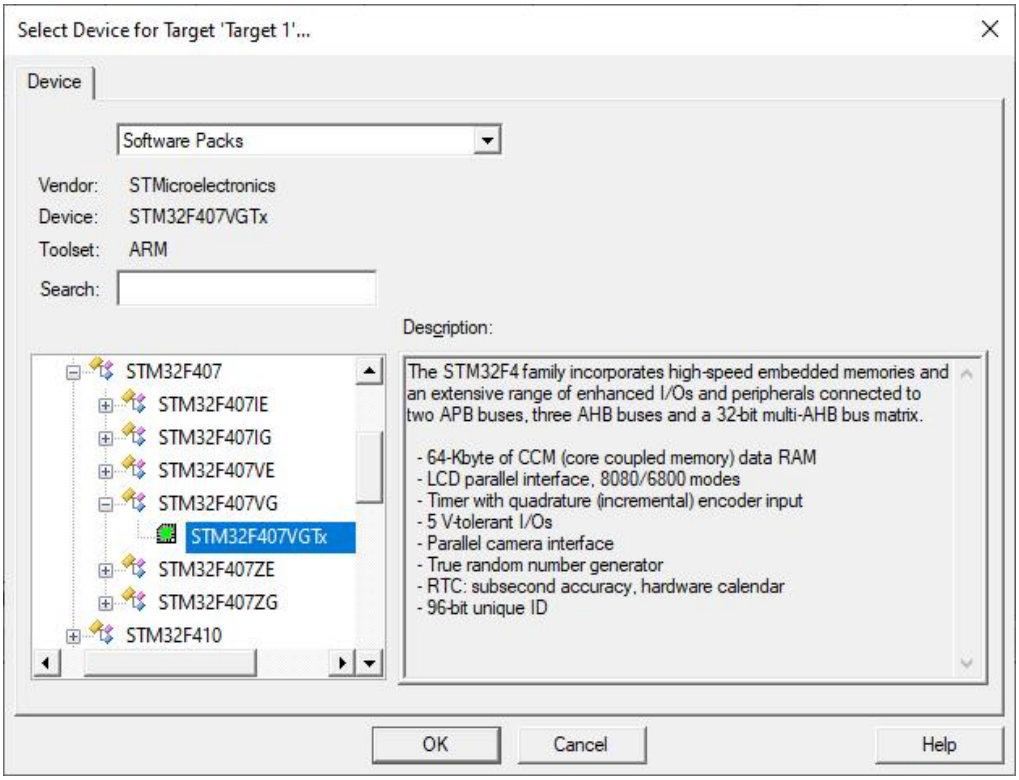



*Рис. 2.19. Мікроконтролер STM32F407VGTx у дереві каталогів*

2.19. У вікні “Manage Run-Time Environment” (MRTE) вибрати компоненти, які будуть використовуватись у проєкті (рис. 2.20). У подальшому для додавання чи видалення

компонентів середовище MRTE можна викликати за допомогою меню Project > Manage > Run-Time Evironment.

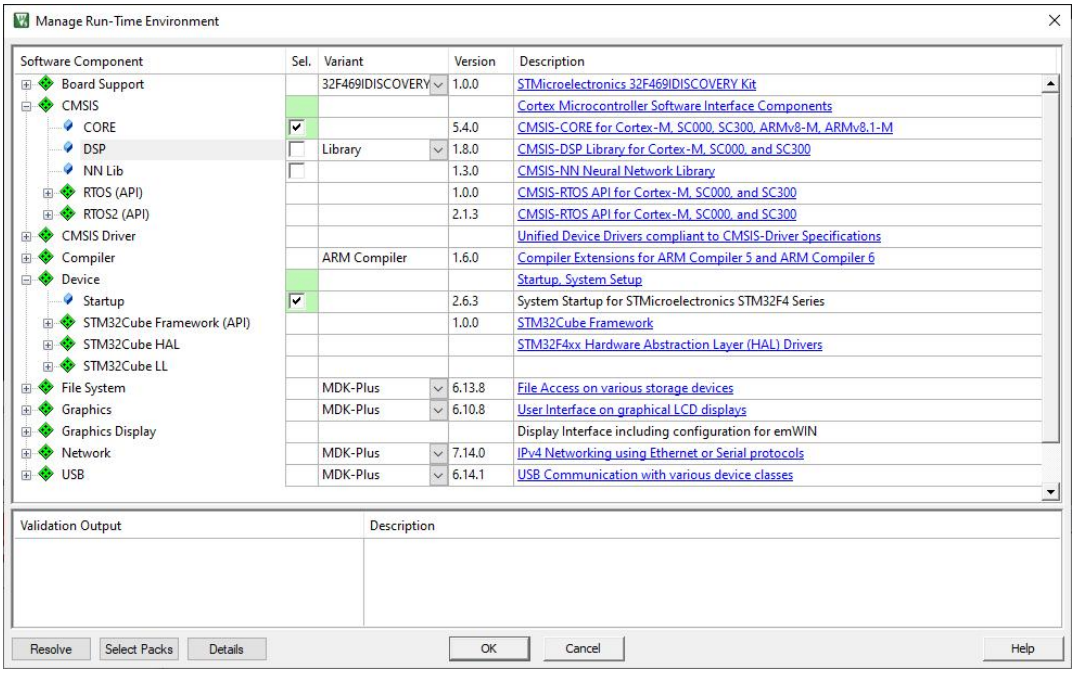



*Рис. 2.20. Вибір компонентів у вікні MRTE*

2.20. Створити основний файл проєкту. Для цього у вікні Keil µVision в панелі структури проєкту (рис. 2.21) правою кнопкою миші натиснути на “Source Group 1” і вибрати “Add New Item to ‘Source Group 1’” (рис. 2.22).

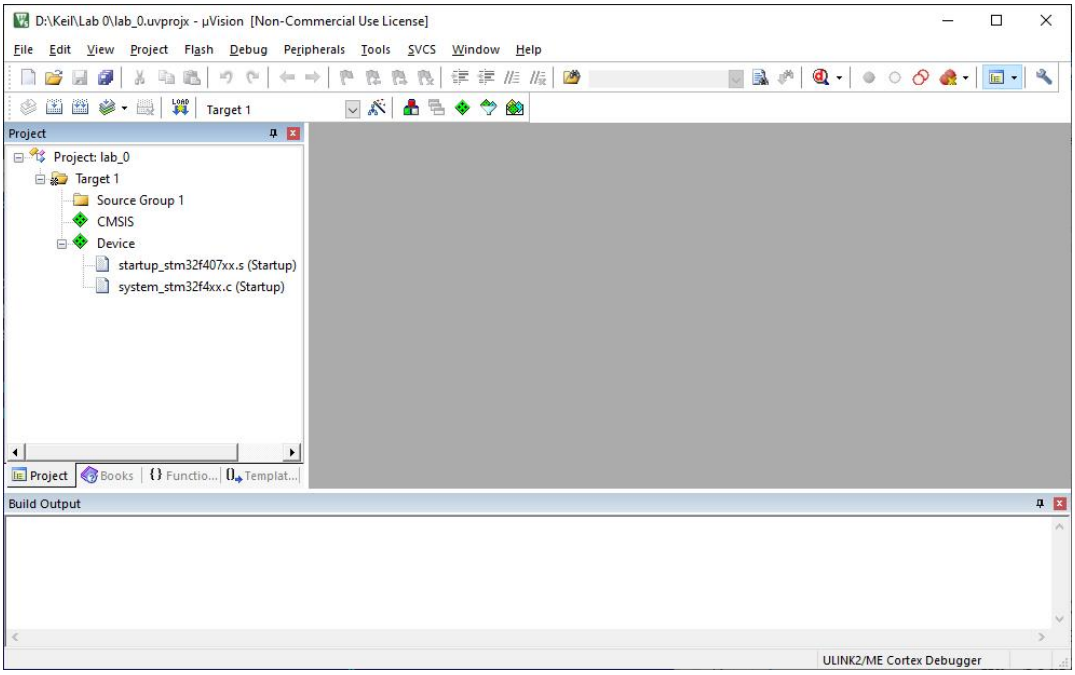



*Рис. 2.21. Структура проєкту в Keil µVision*

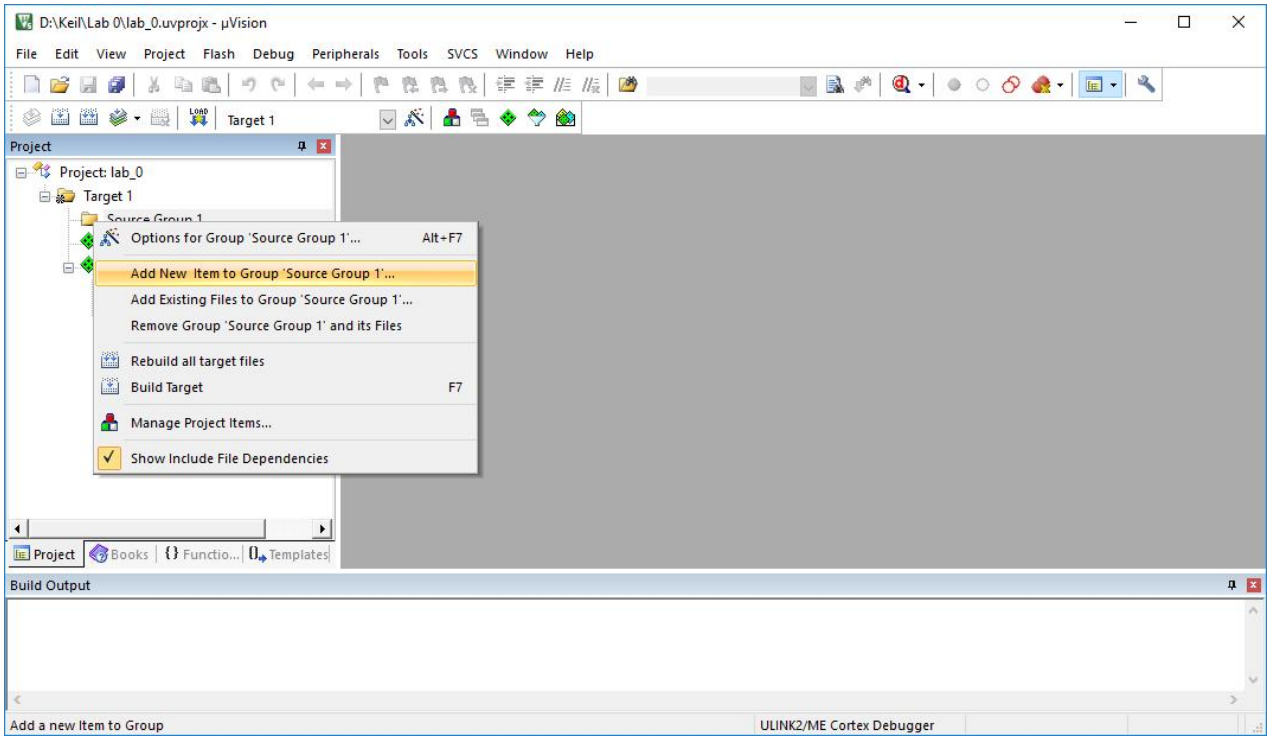



*Рис. 2.22. Створення основного файлу проєкту*

2.21. Вказати тип і назву файлу (рис. 2.23).

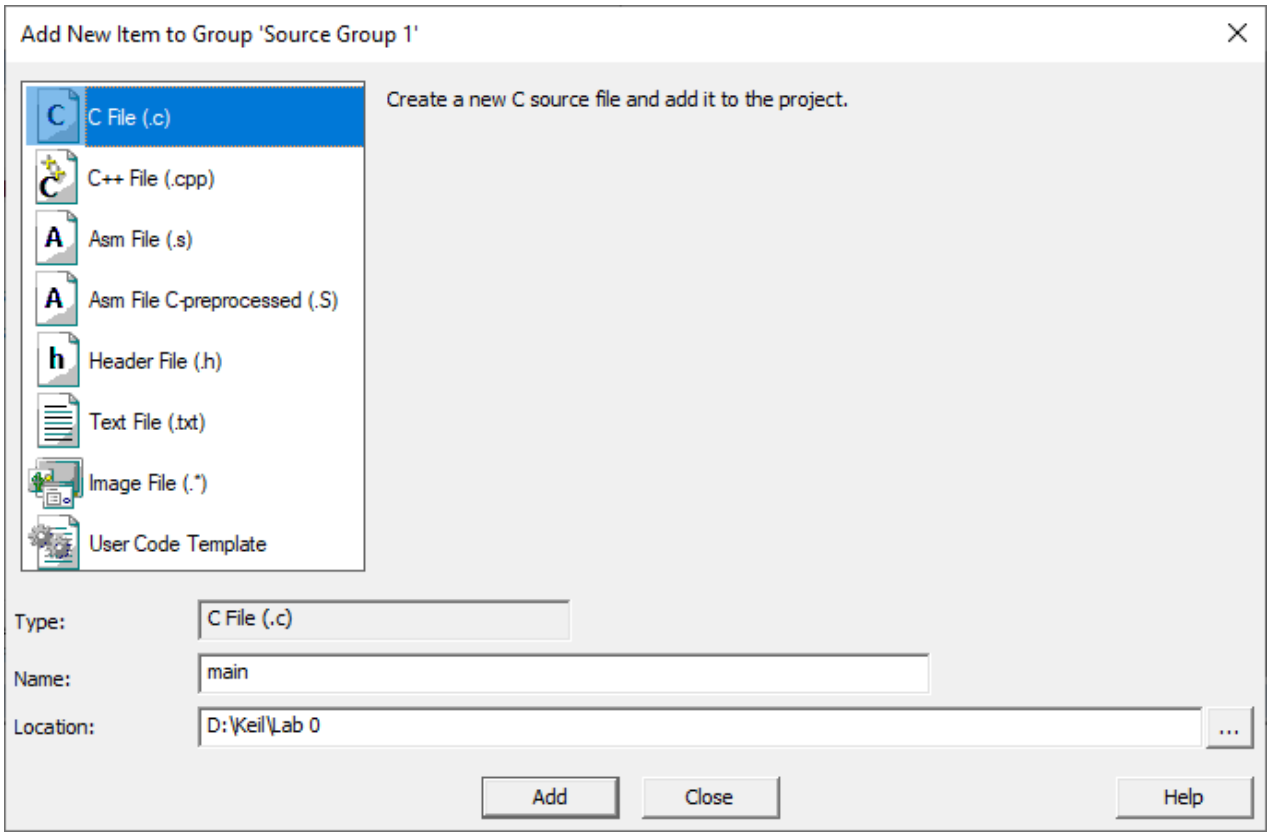



*Рис. 2.23. Файл* **main** *мовою С*

2.22. До файлу дописати кілька рядків програми (рис. 2.24), яка у коментарях має містити ім’я та прізвище студента латиницею. Натиснути на функціональну клавішу F7 (Project > Build Target), щоб переконатися у відсутності помилок.

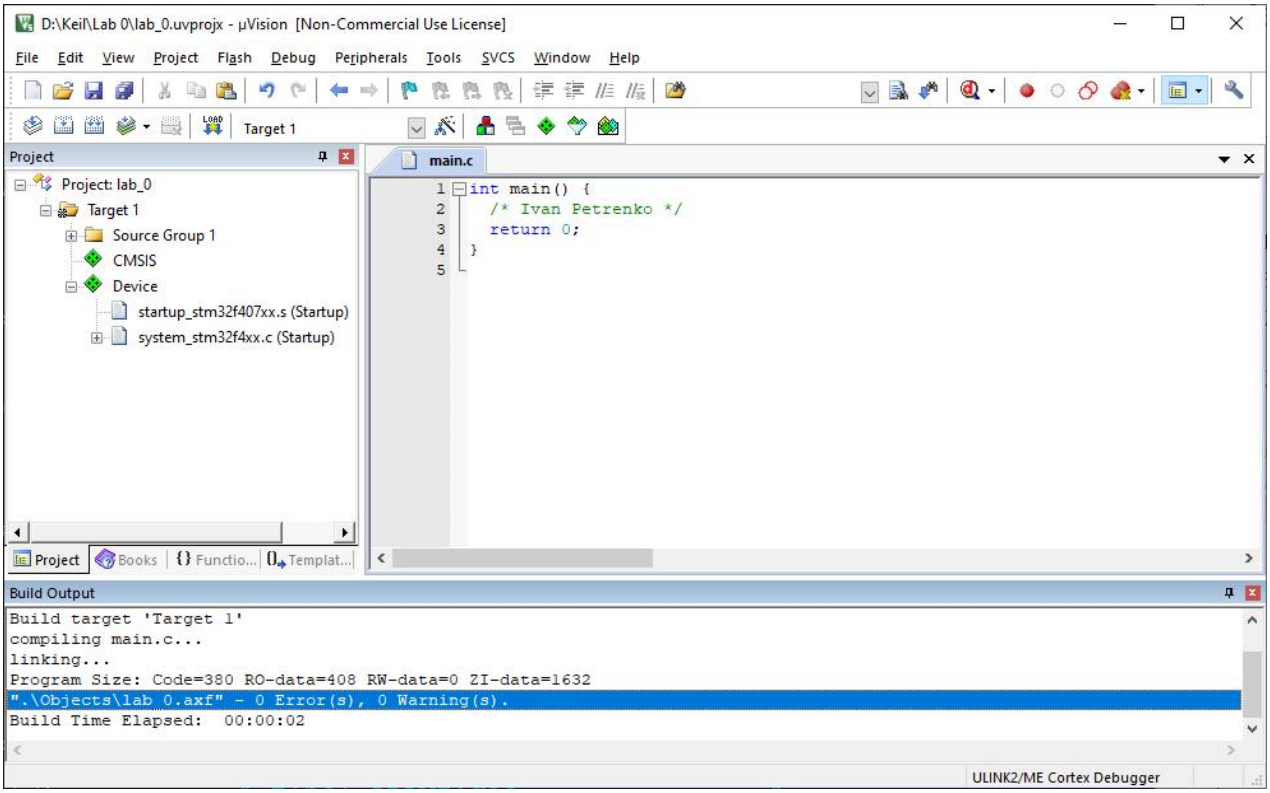



*Рис. 2.24. Початковий файл проєкту без помилок*

2.23. Закрити Keil µVision: File > Exit (Alt+F4).

## 3. Встановлення USB-драйвера ST-Link V2

3.1. У пошуковому полі браузера набрати: ST-Link V2. В отриманому результаті, який стосується компанії STMicroelectronics, натиснути на посиланні “STSW-LINK009”.

3.2. На вебсторінці “STSW-LINK009” фірми-виробника STMicroelectronics послідовно натиснути спочатку верхню, а згодом, після автопрокрутки, і нижню кнопки “Get Software”.

3.3. Прийняти ліцензійну угоду, натиснувши “ACCEPT”.

3.4. У діалоговому вікні (рис. 3.1) у відповідних полях набрати свої ім’я, прізвище, адресу електронної поштової скриньки та, поставивши мітку про ознайомлення з умовами продажу, використання і політикою конфіденційності, натиснути кнопку “Download”.

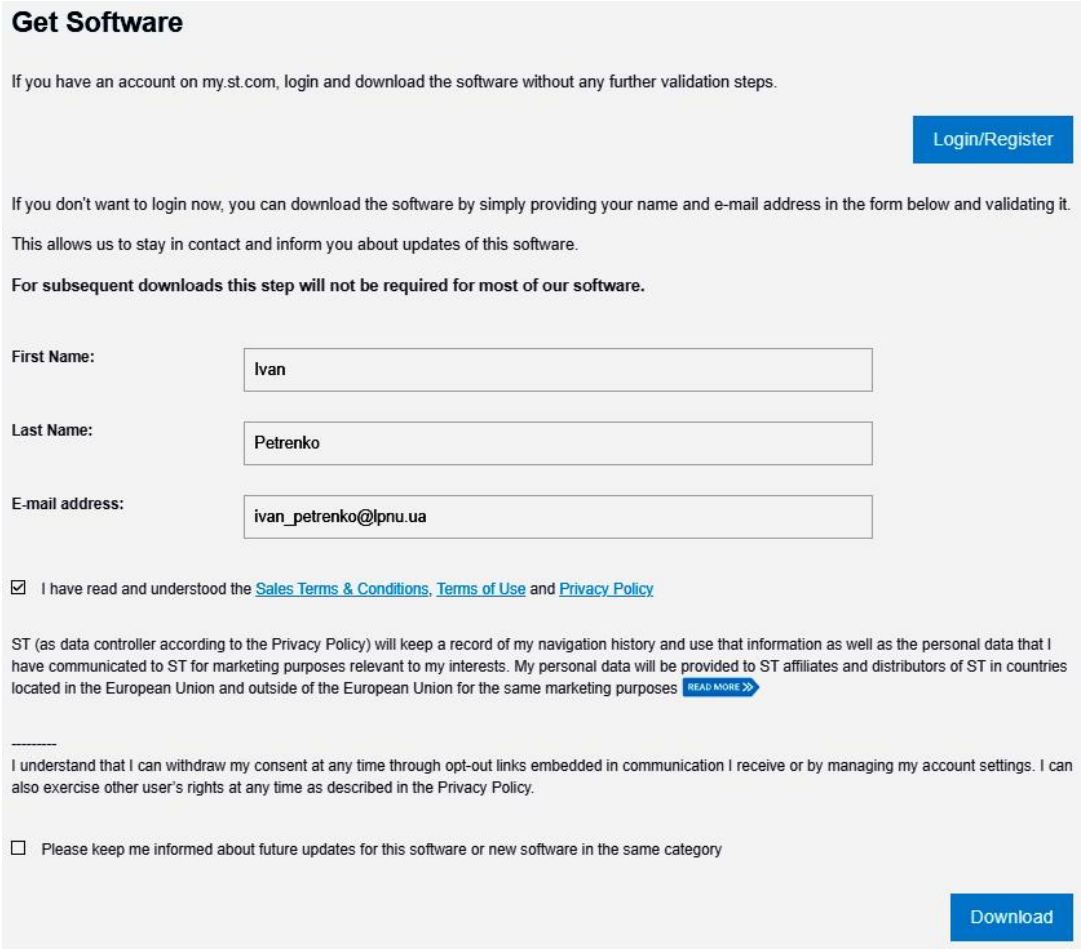



*Рис. 3.1. Діалогове вікно фірми STMicroelectronics з реєстраційними даними користувача*

3.5. Після повідомлення про успішну реєстрацію (рис. 3.2) зайти на свою електронну скриньку, в отриманому листі натиснути “Download now” і завантажити інсталяційний пакет на комп’ютер.

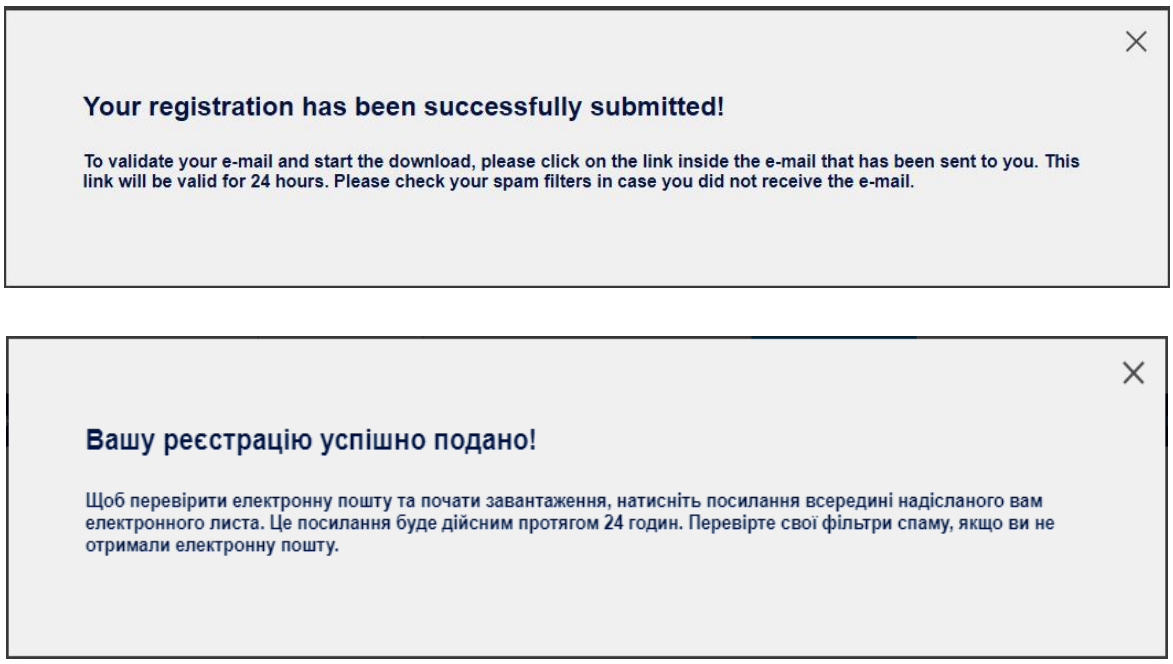



*Рис. 3.2. Повідомлення про успішну реєстрацію*

3.6. Вміст завантаженого інсталяційного архіву (en.stsw-link009.zip) розпакувати у визначений вами каталог і в ньому запустити на виконання EXE-файл, що відповідає вашій операційній системі, наприклад, dpinst_amd64.exe. Відкриється початкове вікно інсталятора (рис. 3.3).

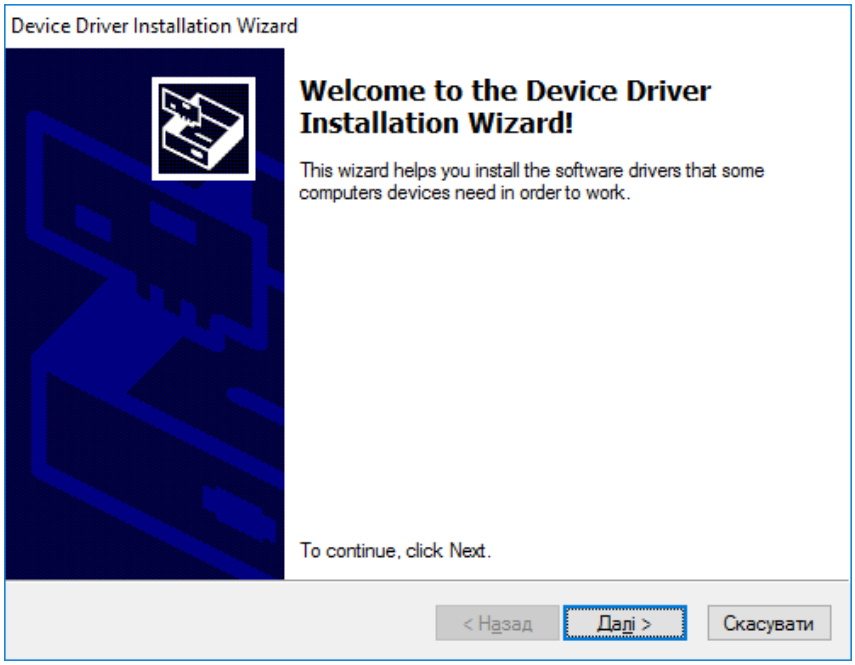



*Рис. 3.3. Ініціалізація встановлення USB-драйвера ST-Link V2*

3.7. Натиснувши кнопку “Інсталяція” (рис. 3.4), запевнити операційну систему, що драйвер ST-Link V2 не є для неї загрозою.

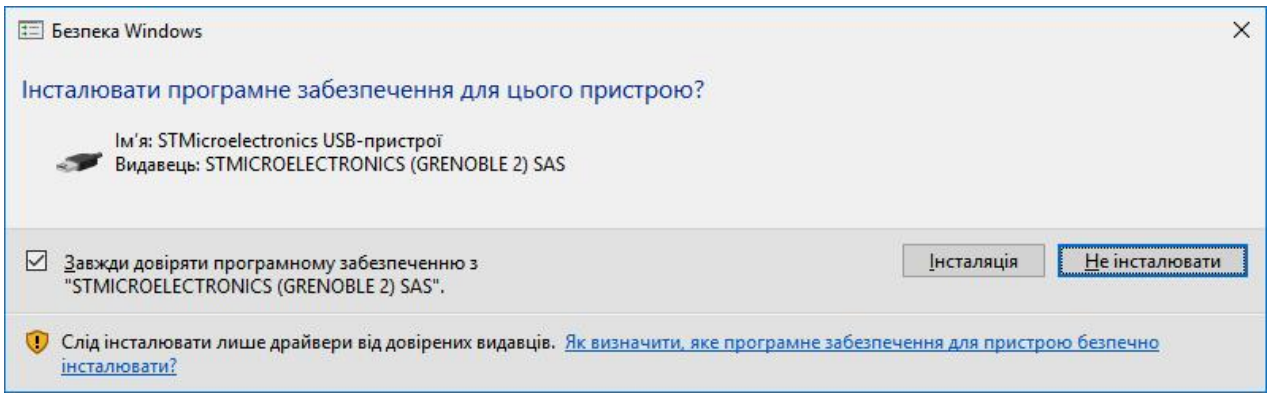



*Рис. 3.4. Запит від системи безпеки Windows*

3.8. Завершити роботу інсталятора (рис. 3.5).


*Рис. 3.5. Успішне встановлення USB-драйвера ST-Link V2*

## 4. Встановлення драйвера перетворювача USB – UART TTL CP2102

4.1. У пошуковому полі браузера набрати: Silicon Labs. В отриманому результаті натиснути на посиланні “Software and Tools” (рис. 4.1).

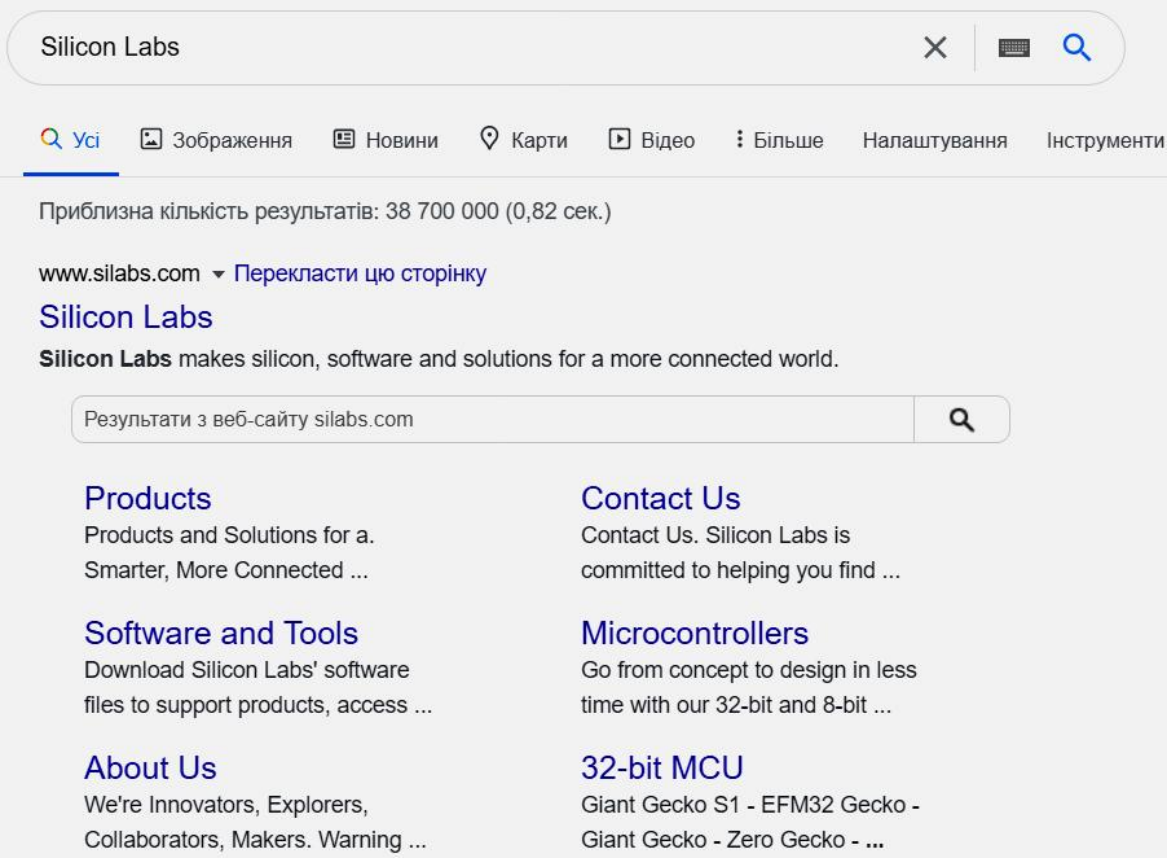



*Рис. 4.1. Результат пошуку словосполучення “Silicon Labs” (фрагмент екрана)*

4.2. На вебсторінці фірми Silicon Labs з використанням прокрутки знайти розділ “INTERFACE > Featured Interface Software” (рис. 4.2) і натиснути на “CP210x VCP Drivers”.

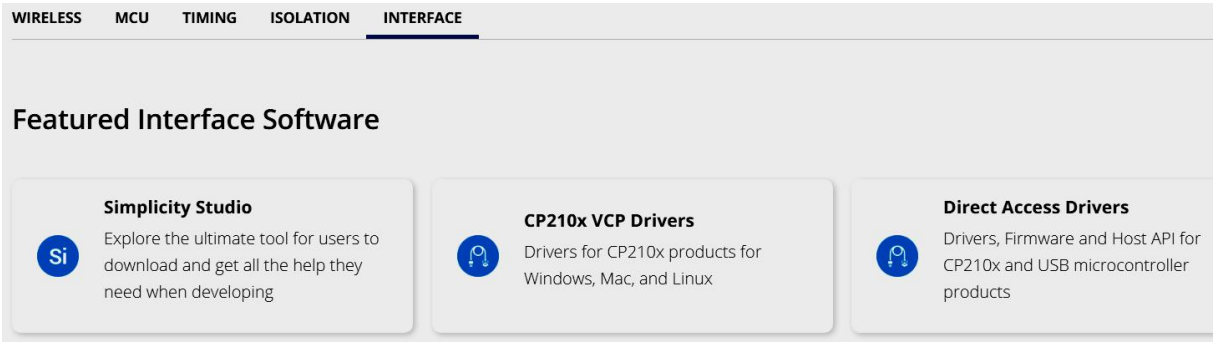



*Рис. 4.2. Фрагмент вебсторінки фірми Silicon Labs*

4.3. У розділі “DOWNLOADS > Download and Install VCP Drivers” (рис. 4.3) вибрати і завантажити необхідний драйвер під наявну операційну систему.

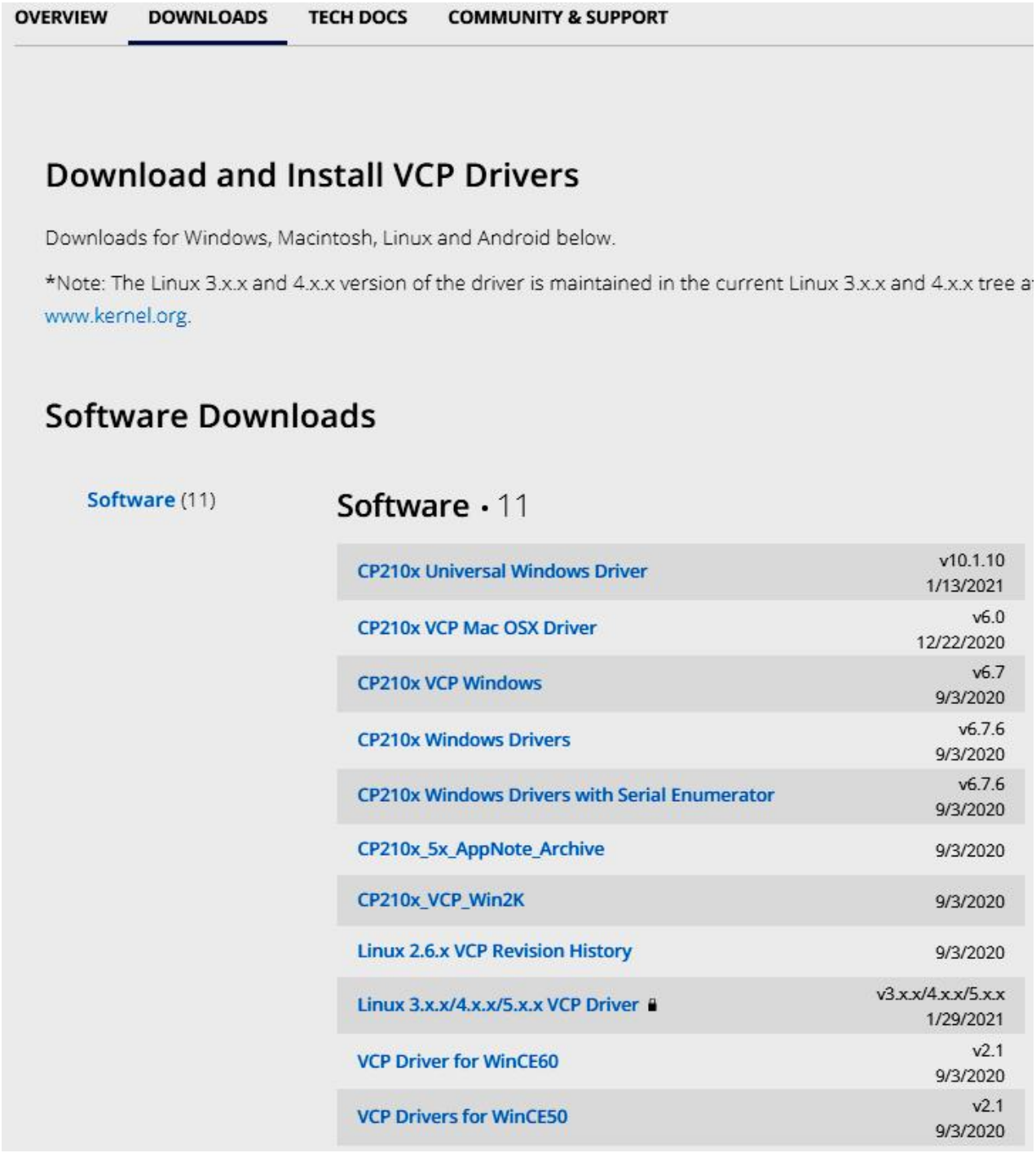



*Рис. 4.3. Вибір варіанта завантаження*

4.4. Вміст завантаженого інсталяційного архіву (наприклад, CP210x_Universal_Windows_Driver.zip) розпакувати у визначений вами каталог і в ньому запустити на виконання EXE-файл, що відповідає вашій операційній системі, наприклад, CP210xVCPInstaller_x64.exe. Відкриється початкове вікно інсталятора (рис. 4.4).

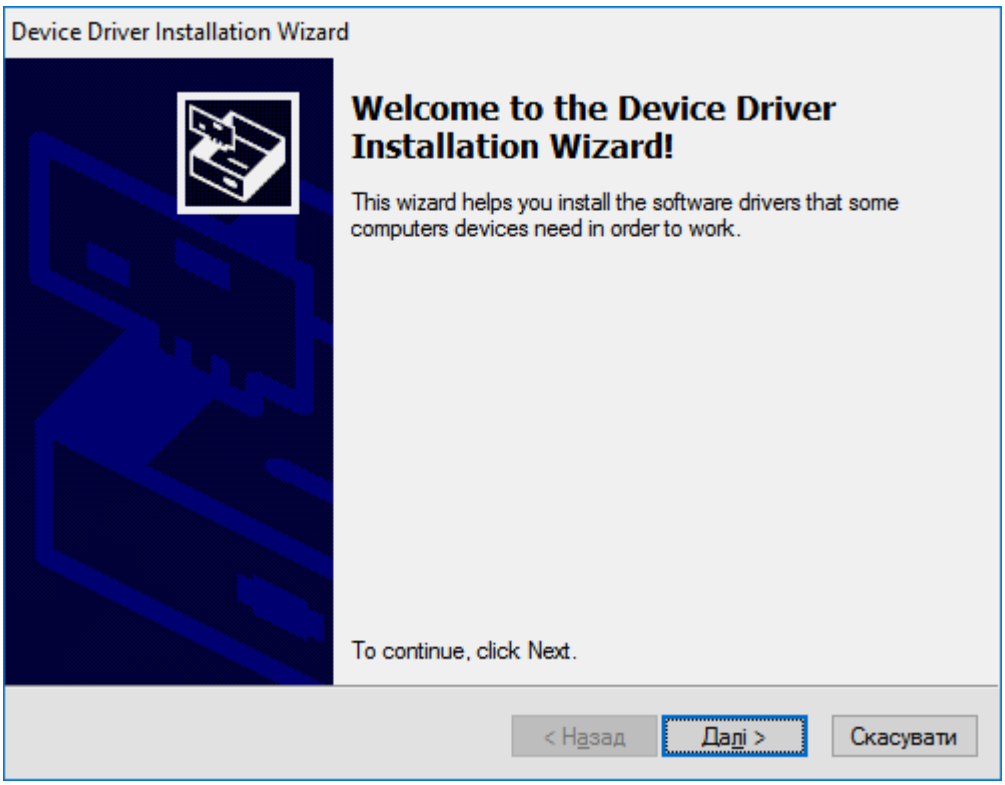



*Рис. 4.4. Ініціалізація встановлення драйвера CP2102*

4.5. Після встановлення завершити роботу інсталятора (рис. 4.5).

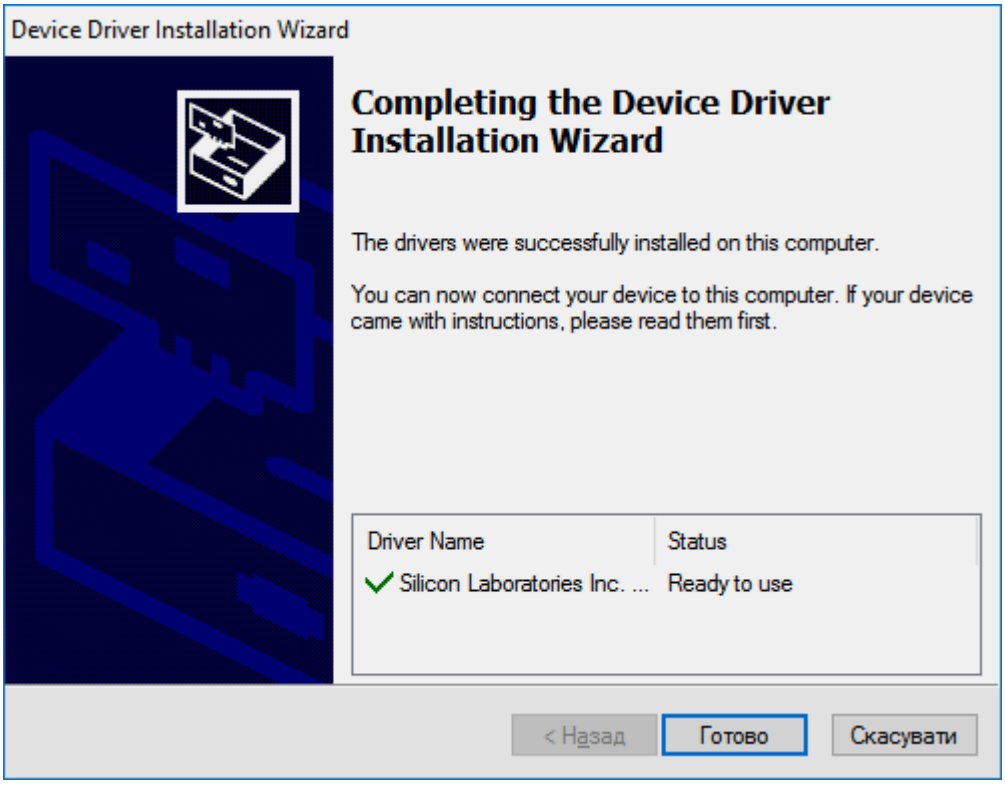



*Рис. 4.5. Успішне встановлення драйвера CP2102*

## ЗАВДАННЯ

Після реєстрації на вебсторінках виробників завантажити і встановити на комп'ютері загальнодоступні версії спеціалізованого програмного забезпечення. Здійснити попередні налаштування.

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.
2. Теоретична частина у тезовому викладі.
3. Опис послідовності виконання роботи. Результати проілюструвати скриншотами діалогових вікон від фірм-виробників із реєстраційними даними студента, а також кінцевими й важливими проміжними скриншотами основних етапів роботи.
4. Висновки.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого призначений програмний пакет STM32CubeMX і які його особливості?
2. Чим особливий Keil MDK-ARM і що дозволяє зробити його використання?
3. Які функції виконує ST-Link V2 і які його особливості?
4. Для чого призначений CP2102?
5. Який програмний пакет при встановленні вимагає відповідної версії Java?
6. Який із програмних пакетів вимагає подвійної реєстрації користувача?

## ДЖЕРЕЛА ІНФОРМАЦІЇ

1. STM32CubeMX – STM32Cube initialization code generator – STMicroelectronics. URL: https://www.st.com/en/development-tools/stm32cubemx.html.
2. MDK Microcontroller Development Kit. URL: https://www2.keil.com/mdk5.
3. ST-LINK V2 програматор STM. URL: https://uamper.com/ST-LINK-V2-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0%D1%82%D0%BE%D1%80-STM?search=ST-LINK%20V2.
4. Перетворювач USB – UART CP2102. URL: https://3v3.com.ua/product_7034.html.

**Лабораторна робота № 5**

# ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ВЗАЄМОДІЇ АПАРАТНИХ І ПРОГРАМНИХ ЗАСОБІВ

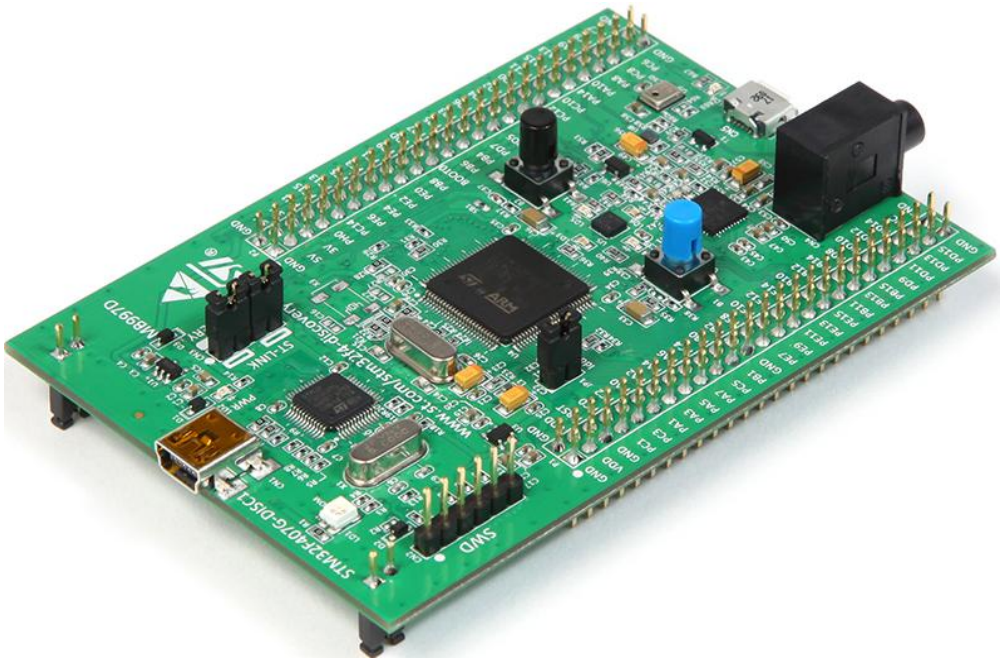
МЕТА РОБОТИ: ознайомитися з порядком розгортання та засобами програмного конфігурування і перевірки функціональних вузлів дослідницько-налагоджувальної плати на базі мікроконтролера STM32F407VGT6.

## ТЕОРЕТИЧНА ЧАСТИНА

### Опис плати розробника STM32F407G-DISC1 для програмування і налагодження проєктів на мікроконтролерах STM32F407VGT6

#### 1. Плата розробника

Плата розробника STM32F407G-DISC1 побудована на 32-бітовому мікроконтролері STM32F407VGT6 з ядром Cortex-M4F, яке можна розігнати до 168 МГц. Об'єм оперативної пам'яті становить 192 КБ. Флеш пам'ять програм – 1 МБ. Є підтримка USB, причому контролер може бути як пристроєм так і хостом. Є апаратний генератор випадкових чисел. Крім звичних інтерфейсів DAC, ADC, SPI, I2C, PWM, RTC у чіпа ще є DCMI для підключення камери і SDIO для підключення SD карт. Також є підтримка Ethernet.




*Плата розробника STM32F407G-DISC1*

У плату STM32F407G-DISC1 вбудований цифровий мікрофон MP45DT02 і стереофонічний 24-бітовий ЦАП CS43L22 для відтворення звуку з вбудованим підсилювачем потужності класу D, регулятором гучності, тембру і т. ін., що дозволяє побудувати на цій платі звукову карту.

Плата оснащена трьохосьовим цифровим акселерометром LIS302DL. Крім стандартних функцій, акселерометр додатково вміє перемикати діапазони вимірювання прискорення, апаратно обробляти кліки і подвійні кліки (наприклад, постукування), генерувати переривання. Має два інтерфейси – SPI та I2C, а також виводи для переривань. Для керування і контролю на платі є 4 різноколірні світлодіоди, розташовані довкола акселерометра, і дві кнопки, призначені для користувача і для скидання контролера.

Живиться контролер через стабілізатор напруги LD3985M33R. Для початкової роботи його струму у 150 мА цілком досить, проте для складних пристроїв з більшою кількістю периферії потрібен потужніший зовнішній стабілізатор напруги 3,3 В.

Для налагодження програми на платі розміщений програматор-відладчик ST-Link. На відміну від програматорів інших плат DISCOVERY, світлодіод даної плати – двоколірний, що дозволяє розрізняти режими роботи налагодження та програмування.

## 2. Виводи і з'єднувачі

Із двох сторін плати виведено по дві пари з'єднувачів-гребінок по 25 виводів кожен з кроком 2,54 мм. На них розташовані 80 штирів для портів вводу-виводу загального призначення і додаткових інтерфейсів. Через з'єднувач mini-USB підключається відладчик ST-LINK, через з'єднувач micro-USB під'єднується контролер. Живиться плата розробника STM32F407G-DISC1 від з'єднувача mini-USB.

Логічна напруга платформи 3,3 В і не всі виводи толерантні до 5 В. Треба бути уважним під час підключення модулів і сенсорів. Якщо подати на такі виводи більше, ніж 3,3 В, мікроконтролер вийде з ладу.

## 3. Середовища розробки

RealView Development Suite (ARM C / C++ Compiler).
Keil MDK-ARM (ARM C / C++ Compiler).
IAR Embedded Workbench for ARM (IAR C / C++ Compiler).
MULTI IDE for ARM (Green Hills C / C++ Compiler).
TASKING VX-toolset for ARM (Altium C / C++ Compiler).
Sourcery CodeBench (GCC Compiler).
Rowley CrossWorks for ARM (GCC Compiler).
Atollic TrueSTUDIO (GCC Compiler).
RAISONANCE Ride7 IDE for ARM (GCC Compiler).
CooCox CoIDE (GCC Compiler).
MikroC for ARM (MikroElektronika C Compiler).

## 4. Характеристики мікроконтролера

Мікроконтролер: STM32F407VGT6 з 32-бітовим ARM Cortex M4.
Корпус: LQFP100.
Тактова частота: 168 МГц.
Обсяг Flash-пам'яті: 1 МБ.
Обсяг SRAM-пам'яті: 192 КБ.

Портів вводу/виводу: 80.
Портів толерантних до 5 В: 78.
Портів з АЦП: 3 х 12 біт.
Кількість каналів АЦП: 24.
Портів з ЦАП: 2 х 12 біт.
Таймери ШІМ: 17.
Апаратні інтерфейси: 3 х SPI, 3 х I2C, 6 х UART, 2 х CAN, 1 х SDIO.
Номінальна робоча напруга: 3,3 В.
Максимальний струм з виводу або на вивід: 25 мА.
Максимальний струм з виводу 3,3 В: 150 мА.

## 5. Особливості плати

Вбудований внутрісхемний відладчик/програматор ST-LINK/V2-A.
Можливість використання ST-LINK/V2-A як окремого пристрою.
Трьохосьовий акселерометр LIS302DL.
Цифровий мікрофон MP45DT02.
Зовнішній ЦАП CS43L22 з підсилювачем класу D.
Вісім світлодіодів:
LD1 (червоний/зелений) для індикації обміну через USB,
LD2 (червоний) – індикація живлення 3,3 В,
Чотири призначені для користувача/розробника: LD3 (помаранчевий), LD4 (зелений), LD5 (червоний) і LD6 (синій),
Два світлодіоди для USB OTG: LD7 (зелений) – VBUS і LD8 (червоний) – індикатор струмового перевантаження.
Дві кнопки: скидання програми і для користувача.
USB OTG зі з’єднувачем micro-USB.
Регулятор напруги з виходом 3,3 В і струмом до 150 мА.
Габарити: 97 х 66 мм.

## ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

### 1. Створення проєкту

1.1. Запустити інтегроване середовище розробки STM32CubeMX (рис. 1.1).

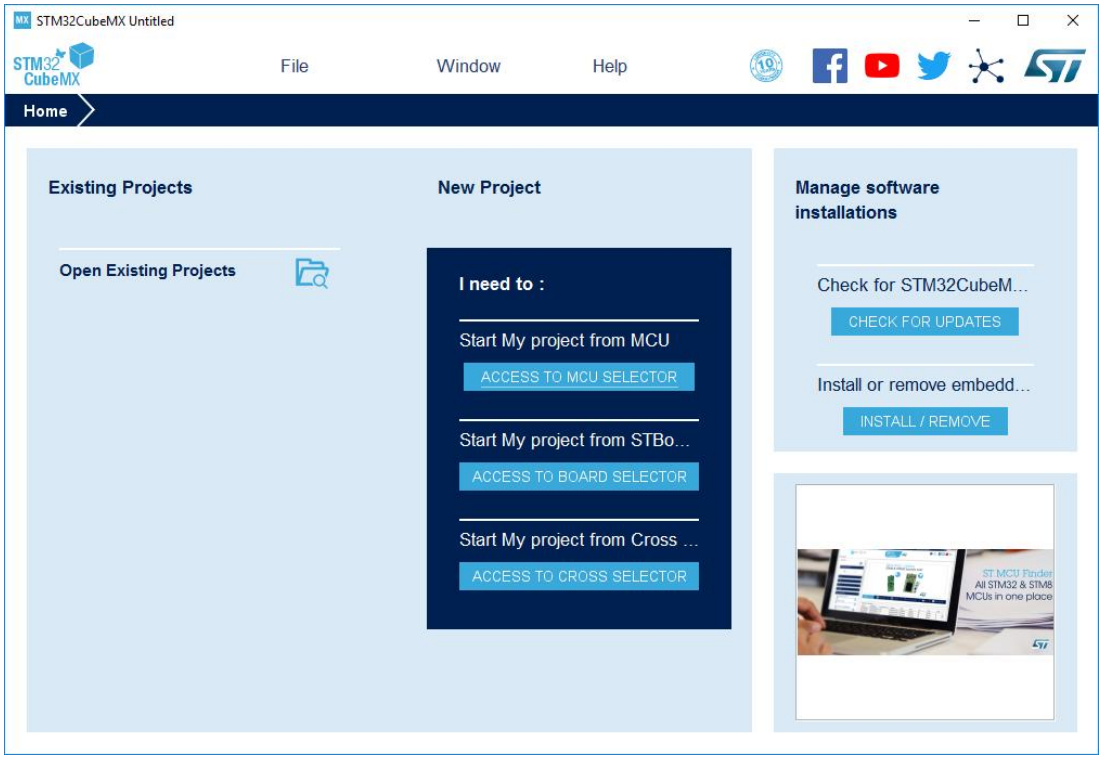



*Рис. 1.1. Стартове вікно STM32CubeMX*

1.2. У меню “New Project” вибрати роботу з платами ST “Start My project from STBoard”, натиснувши кнопку доступу до вибору плати “ACCESS TO BOARD SELECTOR” (див. рис. 1.1). Дочекатися завантаження необхідної інформації з Інтернету (рис. 1.2) і відкриття вікна вибору плати (рис. 1.3).

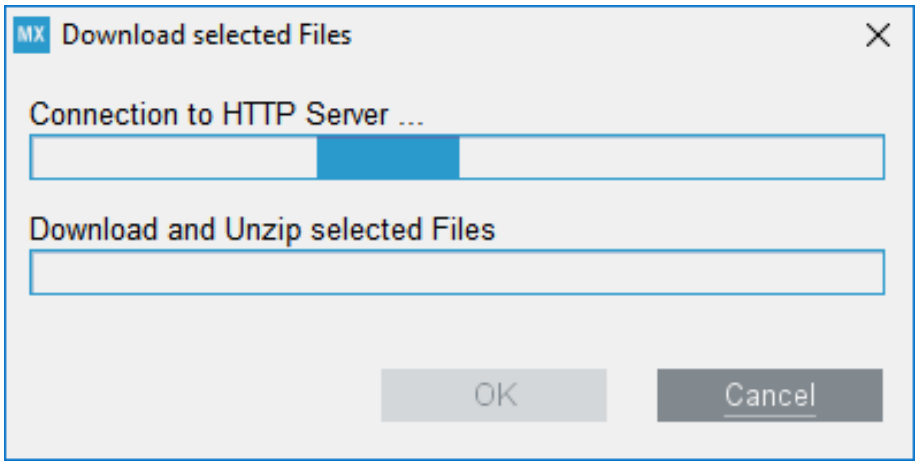



*Рис. 1.2. Оновлення даних*

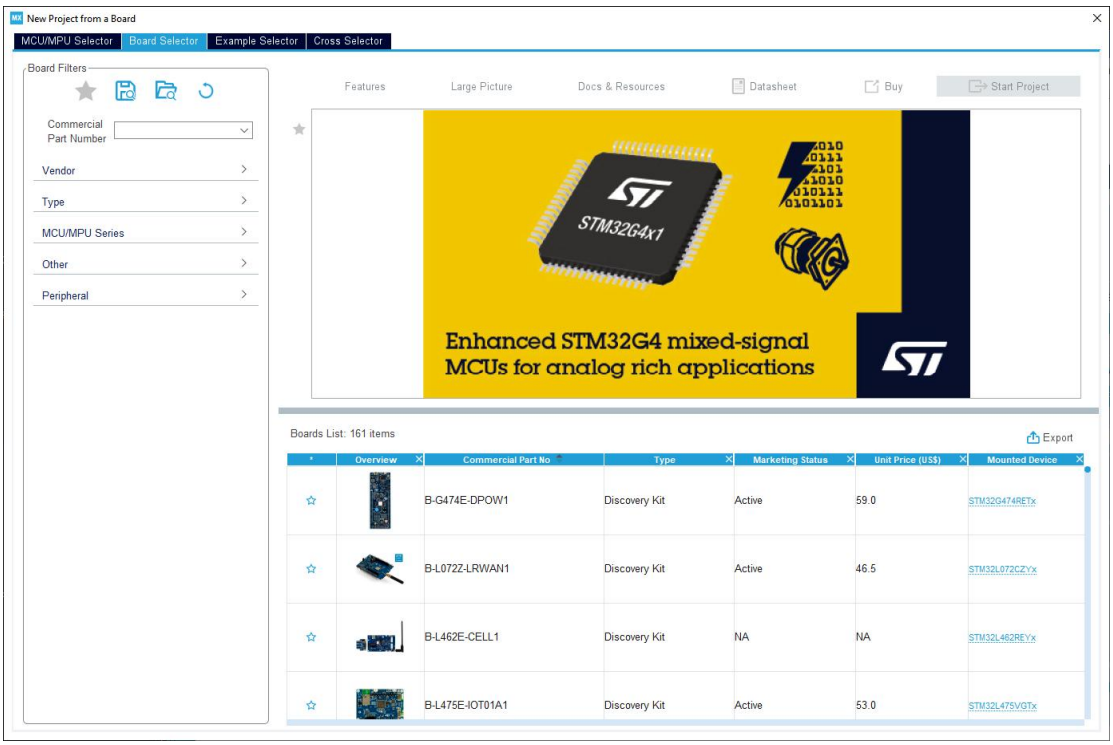



*Рис. 1.3. Вікно вибору плати*

1.3. У вкладці “Board Selector” у розділі “Board Filters” розкрити список “Commercial Part Number” і вибрати плату: STM32F407G-DISC1 (рис. 1.4).

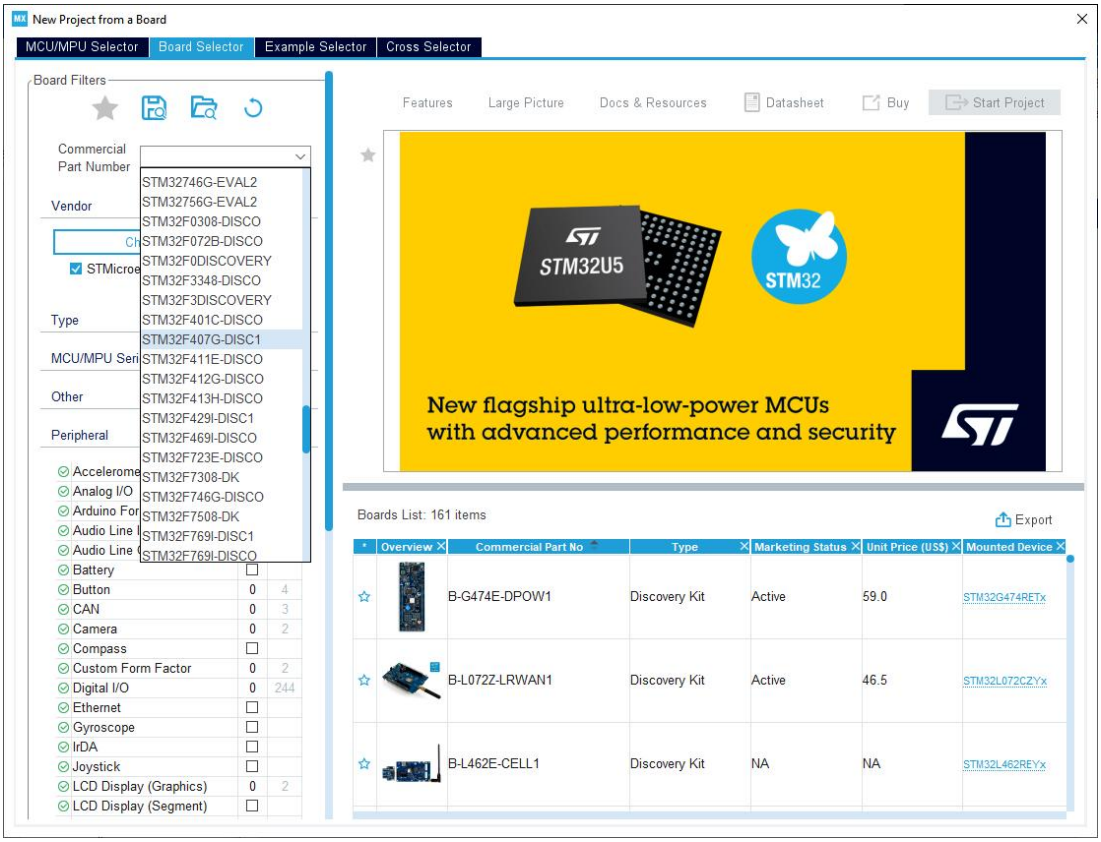



*Рис. 1.4. Вибір необхідного апаратного модуля*

1.4. У розділі “Board List” (рис. 1.5, внизу) натиснути на зображенні плати чи на назві “STM32F407G-DISC1” і отримати її короткий опис та основні характеристики (рис. 1.6). У правому верхньому куті вікна натиснути кнопку “Start Project”.

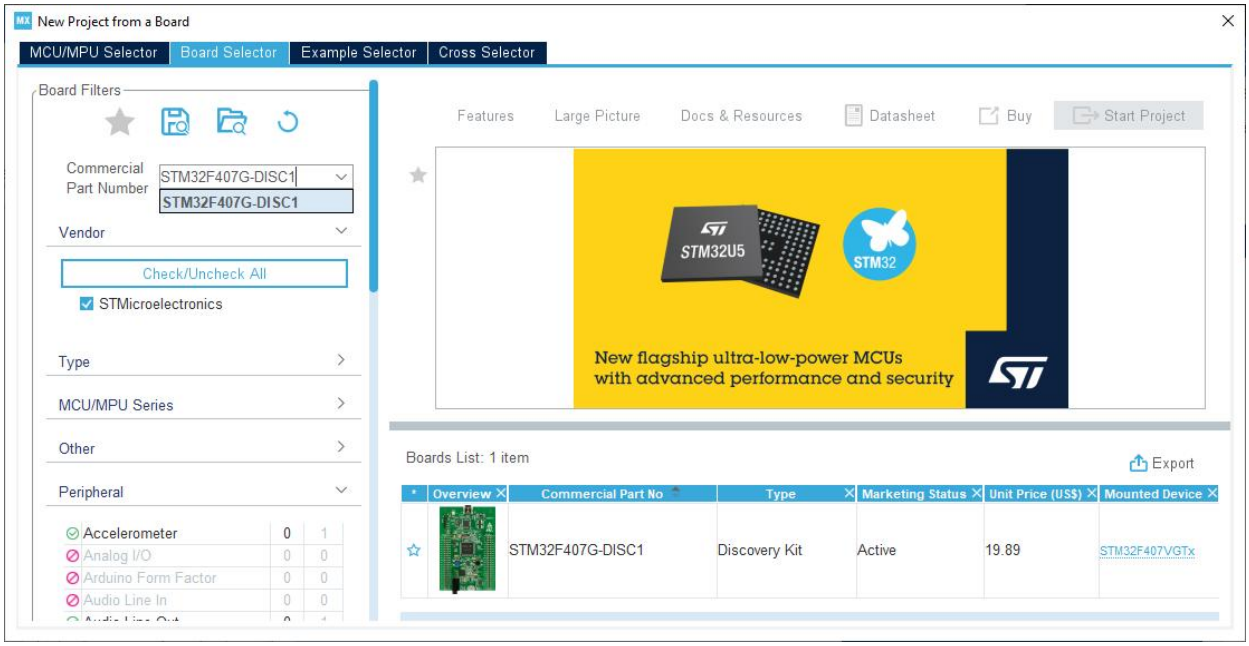



*Рис. 1.5. Вікно вибраної плати*

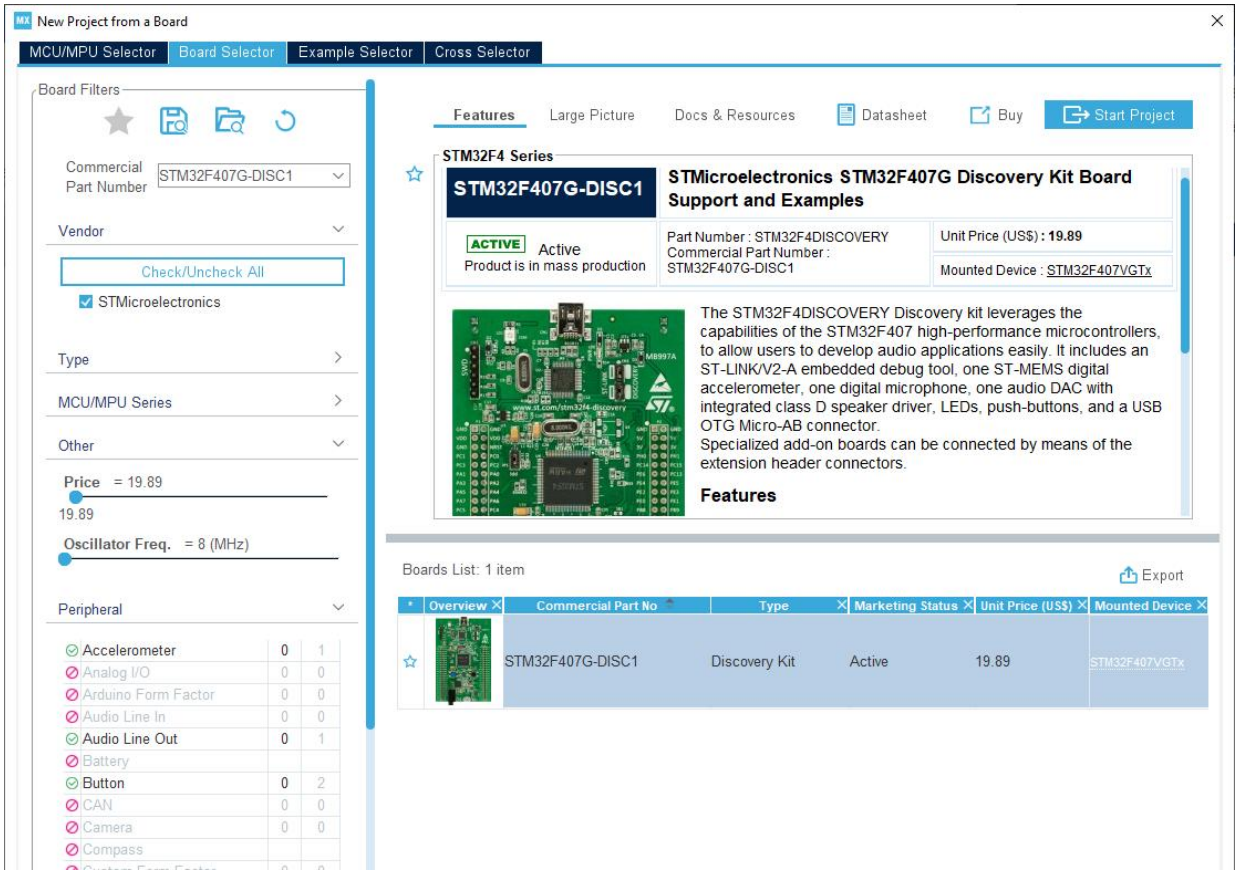



*Рис. 1.6. Короткий опис плати STM32F407G-DISC1*

1.5. Погодитися з пропозицією завантаження та початкової ініціалізації типових параметрів плати (рис. 1.7). Їх оновлення відбудеться через інтернет.

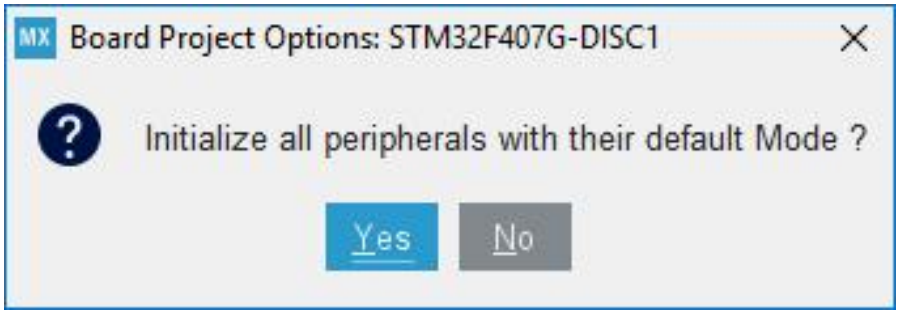



*Рис. 1.7. Пропозиція ініціалізації параметрів плати*

1.6. У вікні заготовки проєкту з типовими параметрами плати (рис. 1.8) у вкладці “Categories” розкрити меню “System Core” і натиснути на “RCC” – Reset and Clock Controler (рис. 1.9).

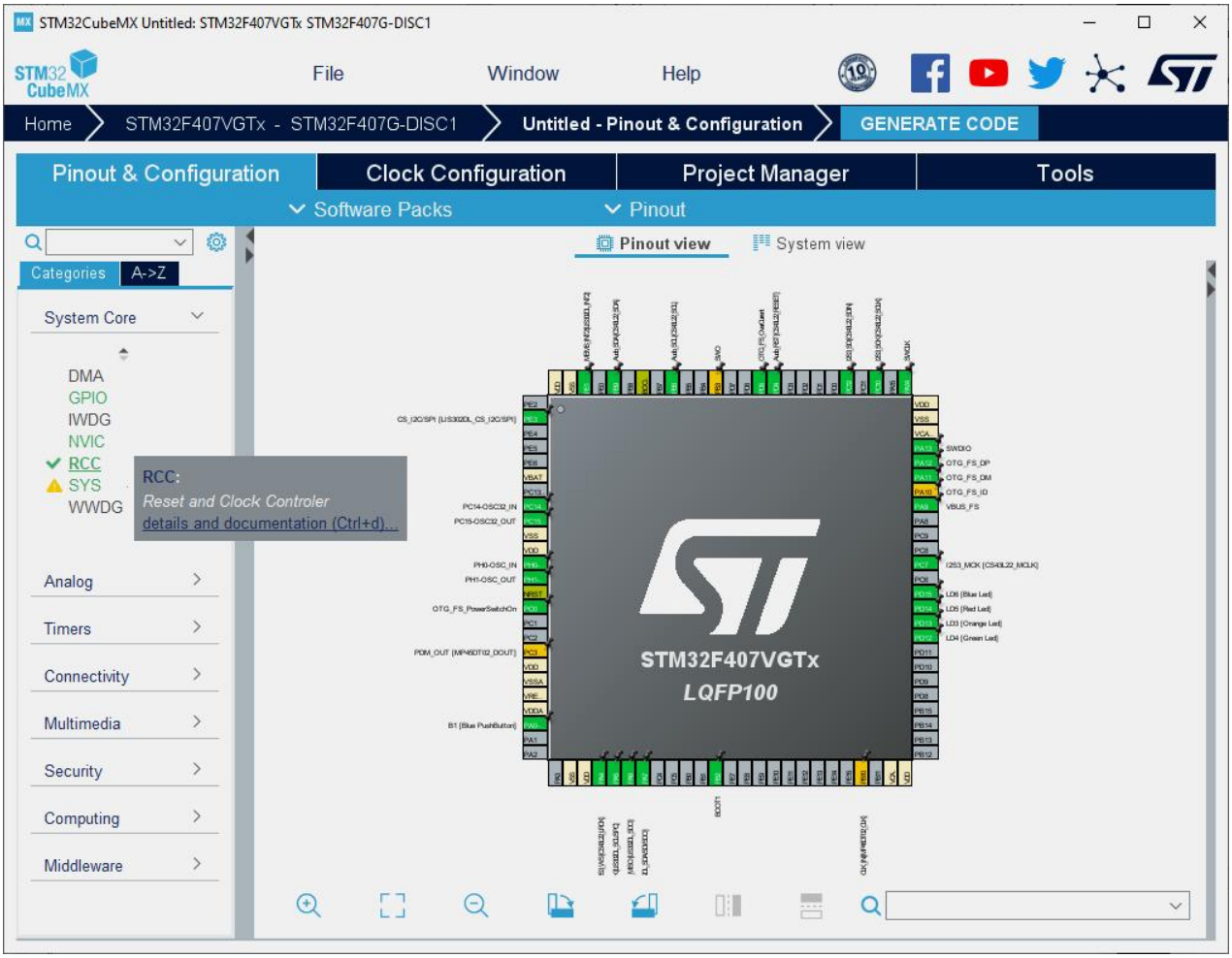



*Рис. 1.8. Проєкт з типовими параметрами плати*

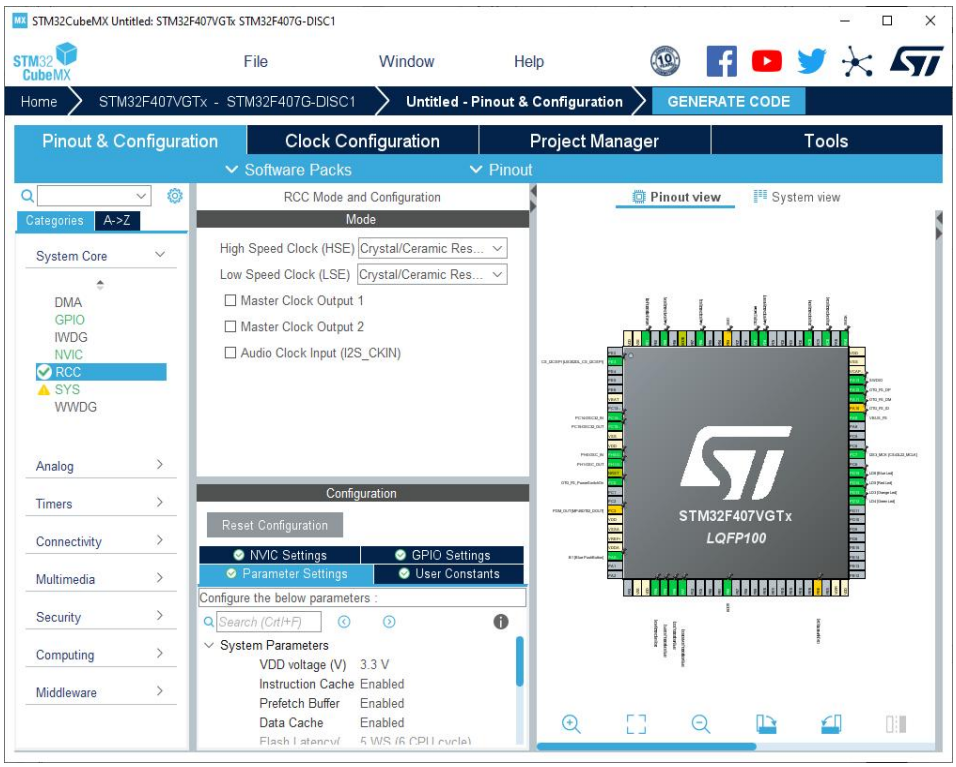



*Рис. 1.9. Вибір пункту меню "Reset and Clock Controler"*

1.7. У розділі "RCC Mode and Configuration" (рис. 1.10) знайти рядок "Low Speed Clock" і встановити значення "Disable". Цей крок забороняє синхронізацію схемних вузлів від кварцу частотою 32 кГц, оскільки плата STM32F407G-DISC1 не містить цього елемента (позиція X3 на платі порожня).

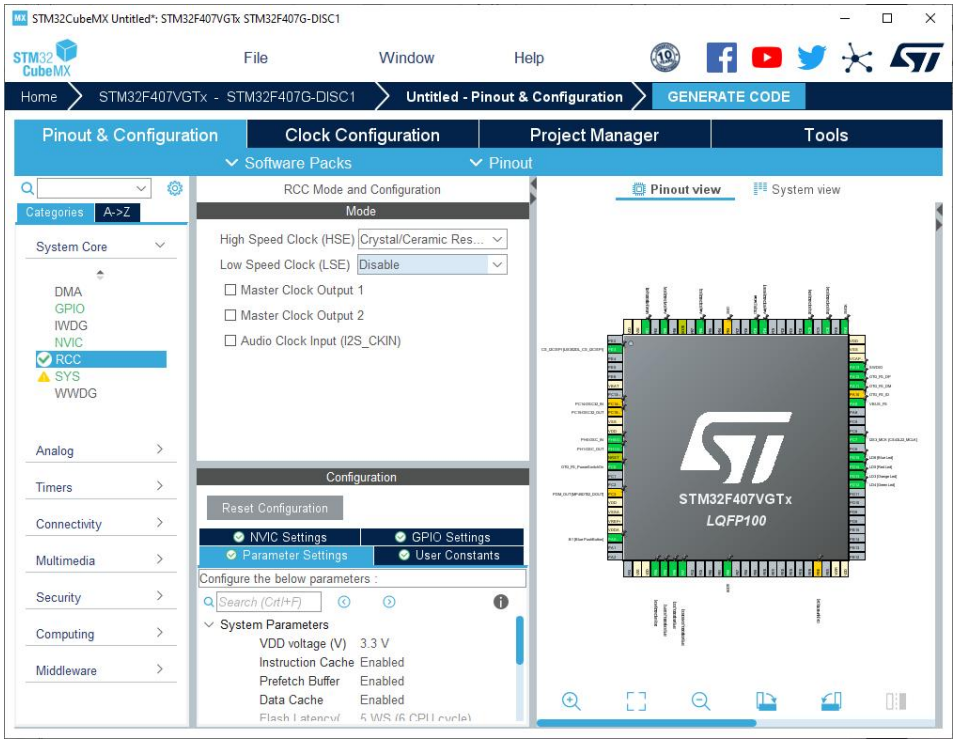



*Рис. 1.10. Відключення кварцу 32 кГц*

1.8. Звернути увагу і занотувати, які саме порти мікроконтролера виділено у проєкті для світлодіодів різних кольорів LD3 – LD6, що призначені для використання розробником (див. рис. 1.10).

1.9. У правому верхньому куті вікна конфігурування проєкту (див. рис. 1.10) натиснути кнопку “GENERATE CODE”. Погодитися з попередженням про необхідність присвоєння проєкту назви та внесення основних параметрів (рис. 1.11).


*Рис. 1.11. Попередження менеджера проєктів*

1.10. Літерами англійської абетки присвоїти назву проєкту (без пробілів), вибрати для нього місце на диску, а також вказати тип компілятора, для якого готуємо пакет проєкту – IDE MDK-ARM (рис. 1.12).

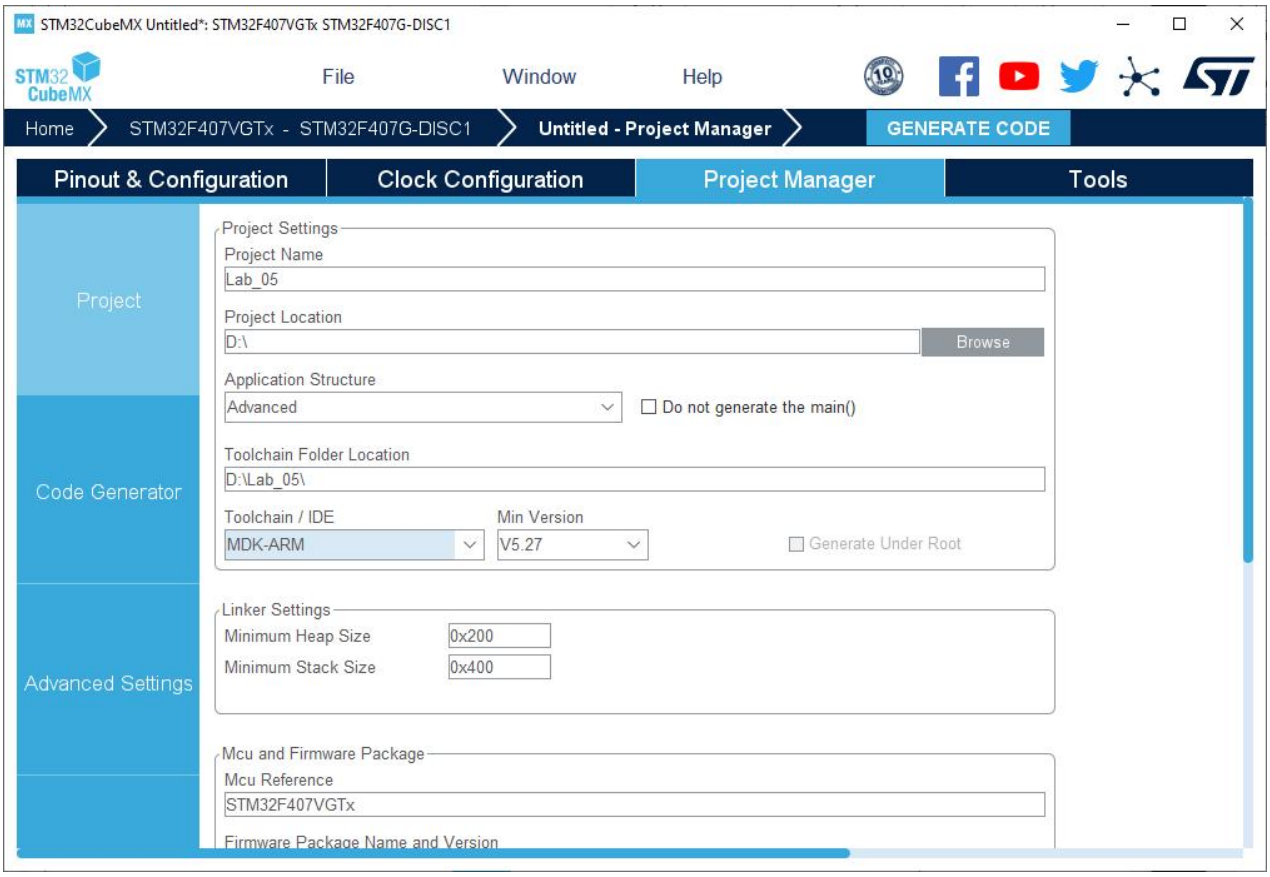



*Рис. 1.12. Основні параметри проєкту*

1.11. Знову натиснути кнопку GENERATE CODE і спостерігати за процесом формування пакету проєкту для вибраного компілятора та копіюванням необхідних бібліотек (рис. 1.13).

Увага! Можлива ситуація (зазвичай виникає, якщо місце для проєкту вибирається не у профілі користувача, а в окремому каталозі), коли з першої спроби не завантажились усі необхідні для даного апаратного модуля файли (рис. 1.14). Тоді треба погодитися на завантаження решти інформації (рис. 1.15). І лише згодом продовжиться створення проєкту (див. рис. 1.13).

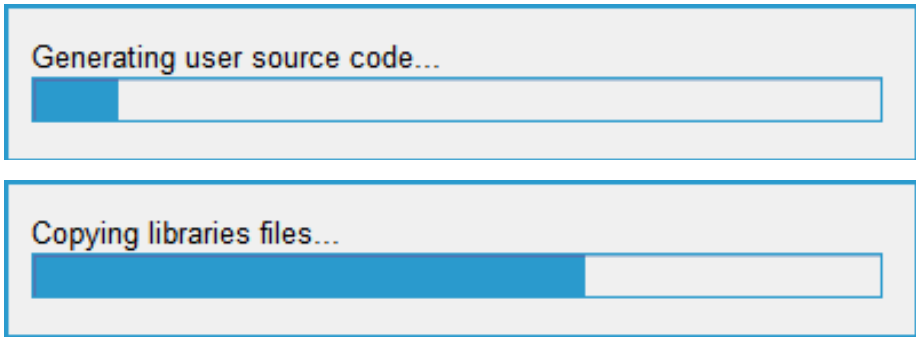



*Рис. 1.13. Процес створення проєкту*

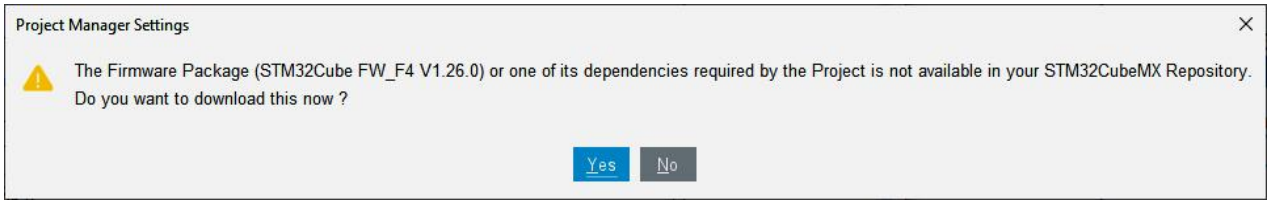



*Рис. 1.14. Запит на додаткове завантаження*

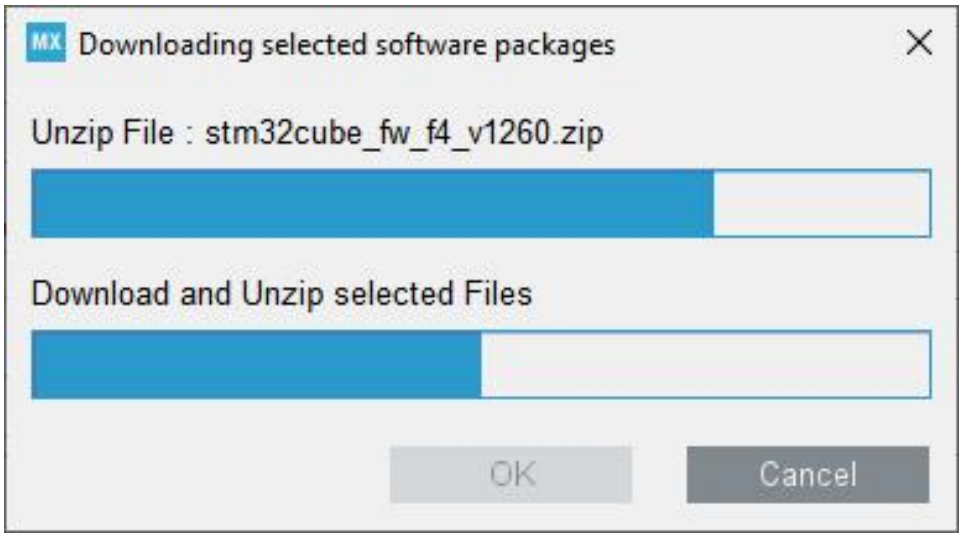



*Рис. 1.15. Завантаження додаткових файлів*

1.12. Погодитися на відкриття проєкту, натиснувши у запиті кнопку “Open Project” (рис. 1.16), щоб запустити MDK-ARM (рис. 1.17).

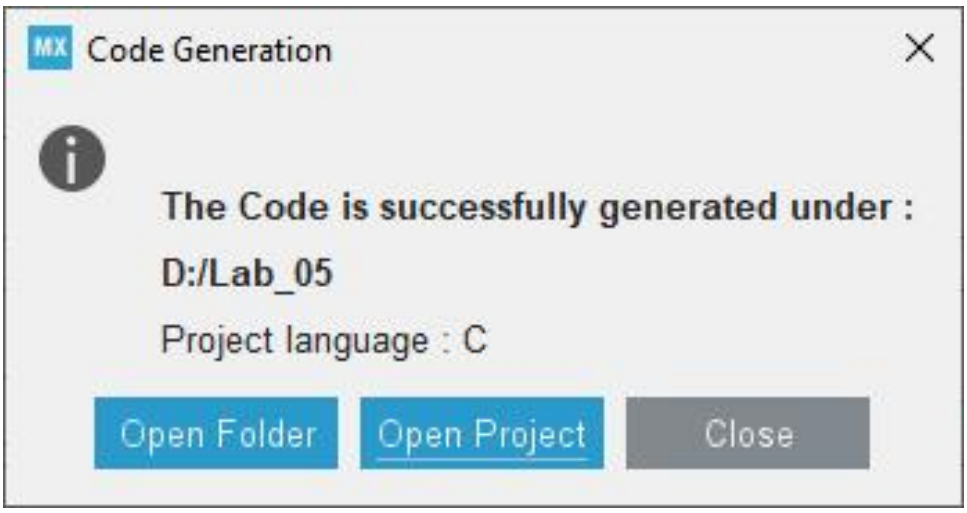



*Рис. 1.16. Запит на відкриття проєкту в MDK-ARM*

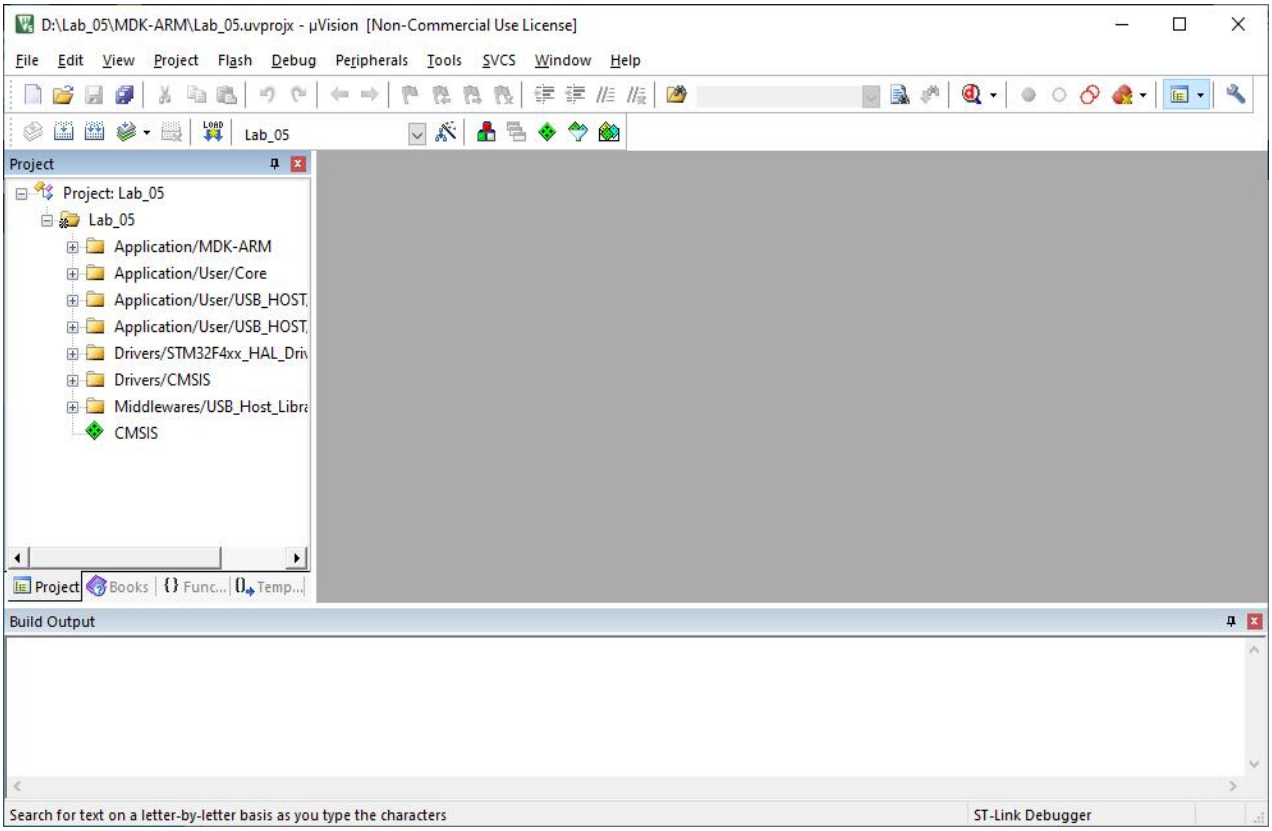



*Рис. 1.17. Вікно MDK-ARM (IDE Keil µVision)*

1.13. В одній із вкладених папок проєкту знайти і відкрити файл **main.c** з основними налаштуваннями проєкту (рис. 1.18).

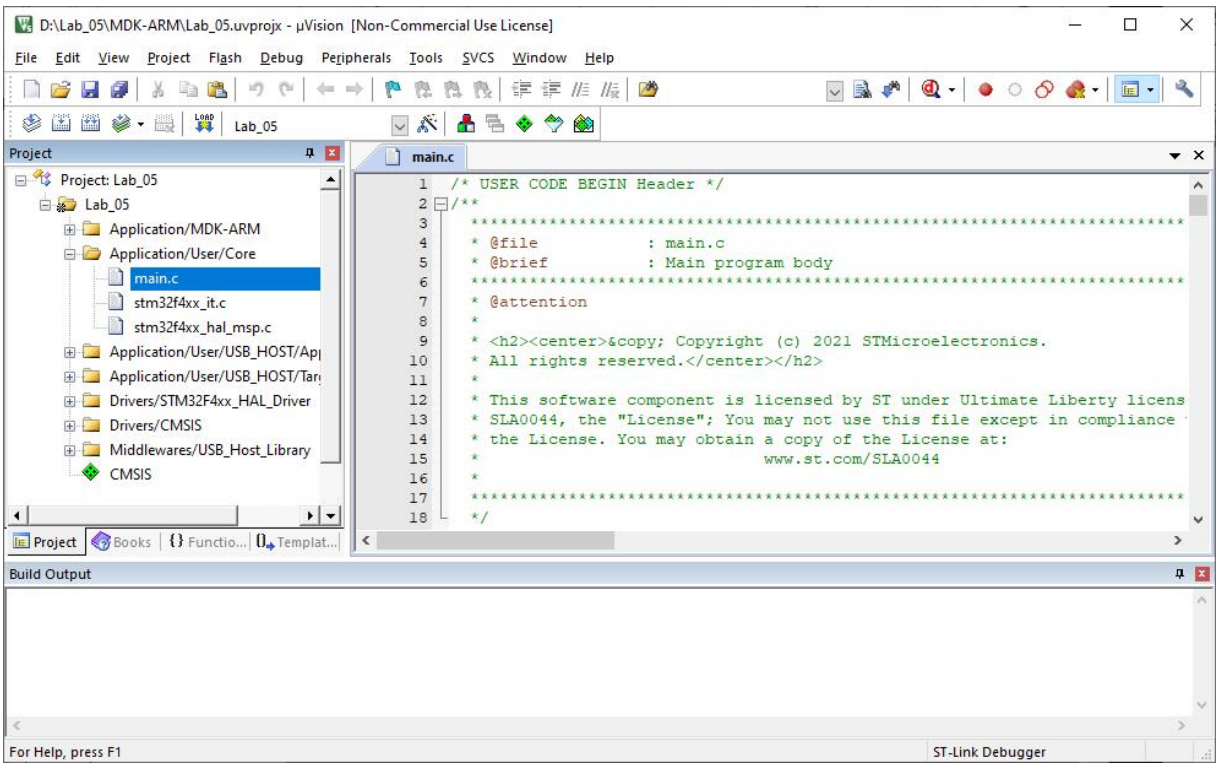



*Рис. 1.18. Головний файл проєкту*

1.14. Натиснути функціональну клавішу F7 (Project > Build Target). Після компілювання переконатися у відсутності помилок в об’єктному коді заготовки проєкту (рис. 1.19).

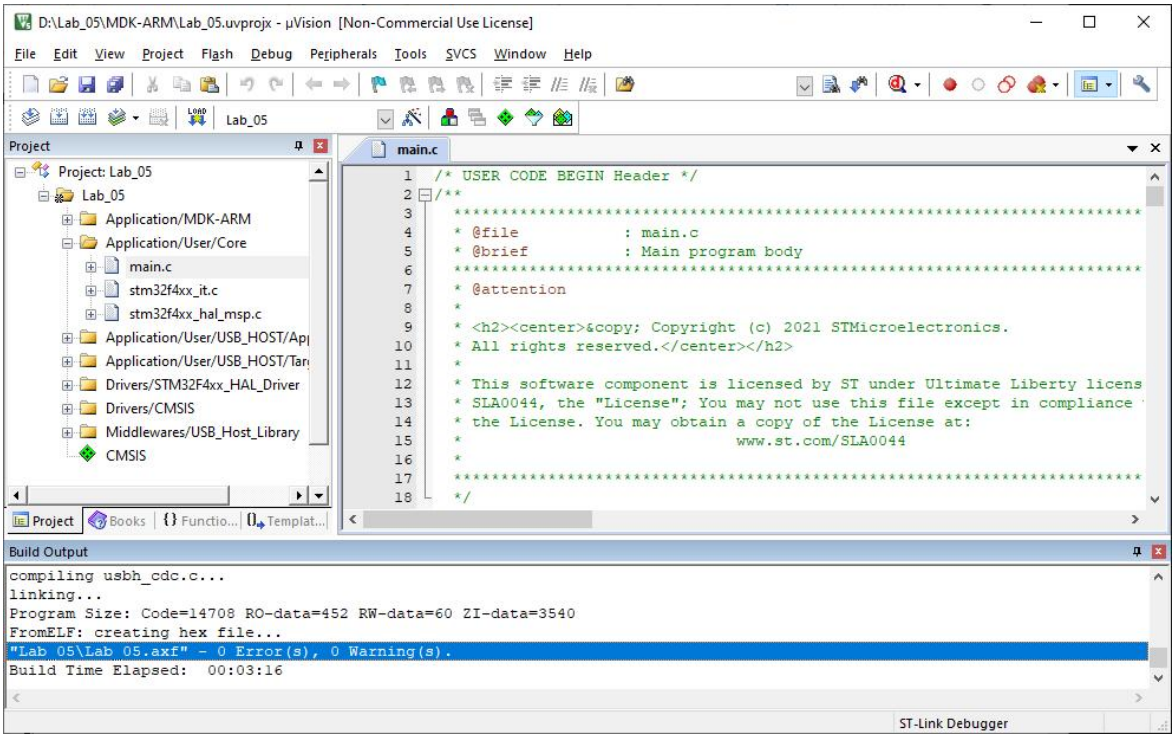



*Рис. 1.19. Успішна компіляція*

1.15. Закрити проєкт (Project > Close Project).

## 2. Програмування плати

2.1. Для перевірки можливості програмування мікроконтролера плати STM32F407G-DISC1 кабельний з’єднувач mini-USB під’єднати до відповідного порту плати, а з’єднувач USB звичайного розміру – до порту комп’ютера.

Увага! На цьому етапі може з’явитися запит на перевірку версії драйвера внутрісхемного відладчика-програматора ST-Link V2 (рис. 2.1).

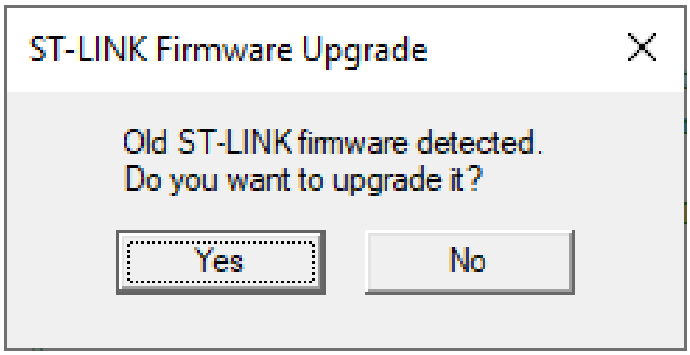



*Рис. 2.1. Запит на перевірку*

Погодитися на перевірку та завантаження і встановлення актуального на цей момент драйвера. Для цього у наступному вікні (рис. 2.2) спочатку натиснути кнопку “Device Connect”, а згодом, після отримання результату перевірки (рис. 2.3), натиснути кнопку “Yes>>>>”.


*Рис. 2.2. Ініціалізація перевірки*

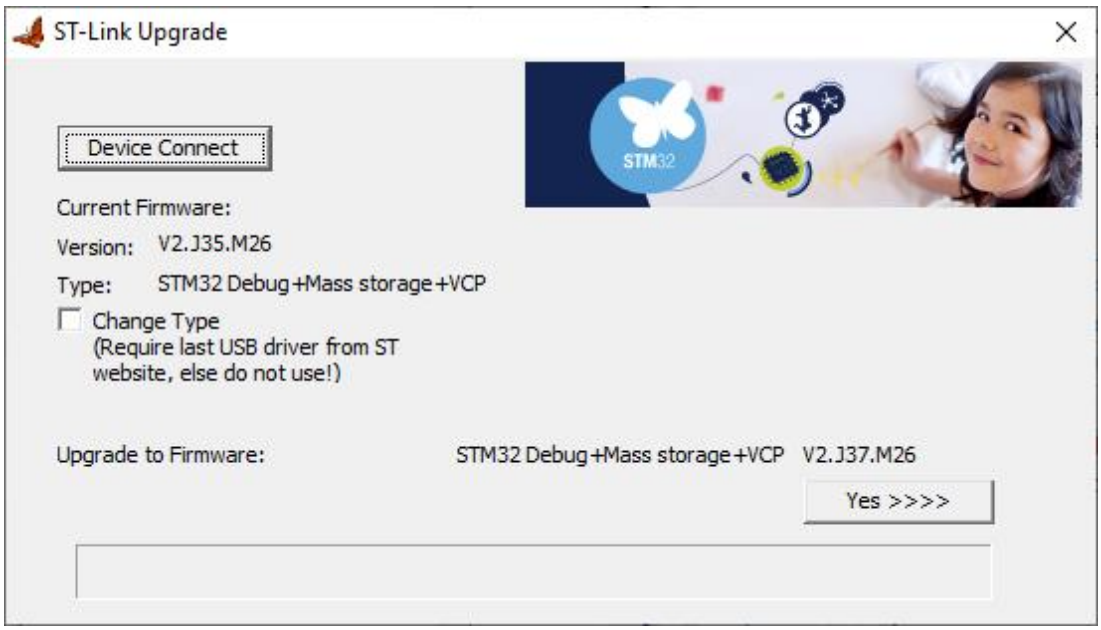



*Рис. 2.3. Результат перевірки*

Дочекатися оновлення драйвера (рис. 2.4).

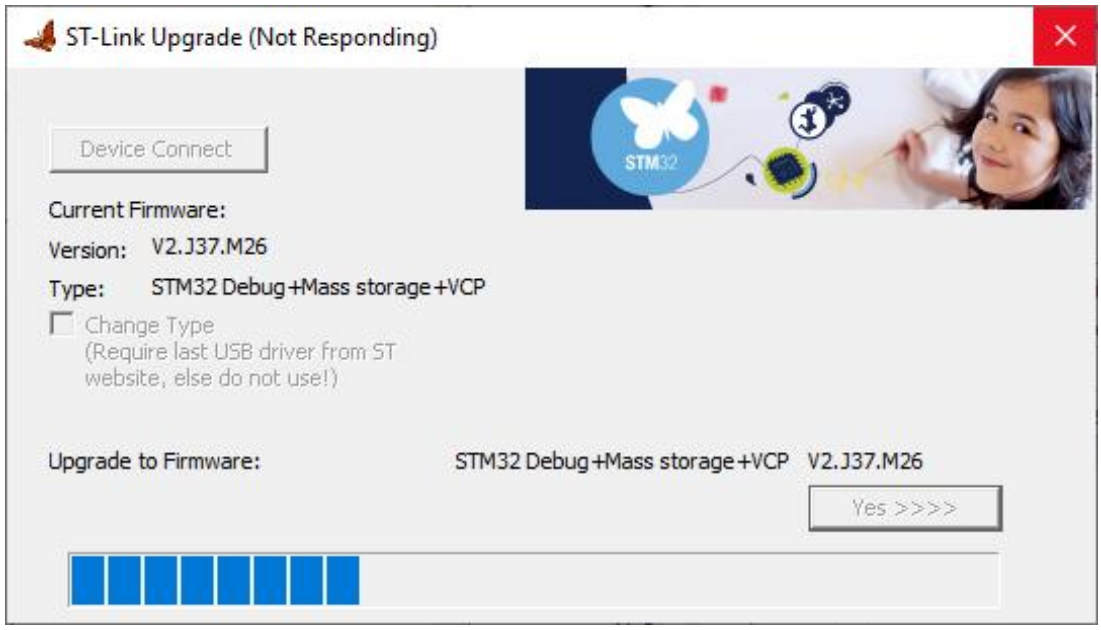



*Рис. 2.4. Процес оновлення драйвера*

2.2 В MDK-ARM відкрити створений проєкт (Project > Open Project) і повторити пункт 1.14 (F7, Project > Build Target). Потім натиснути функціональну клавішу F8 (Flash > Download). У лівому нижньому куті можна побачити процес запису об'єктного коду до пам'яті програм налагоджувальної плати (рис. 2.5). При цьому відбувається блимання світлодіода LD1 (червоний/зелений), призначеного для індикації обміну через USB (рис. 3.2, рис. 3.3).

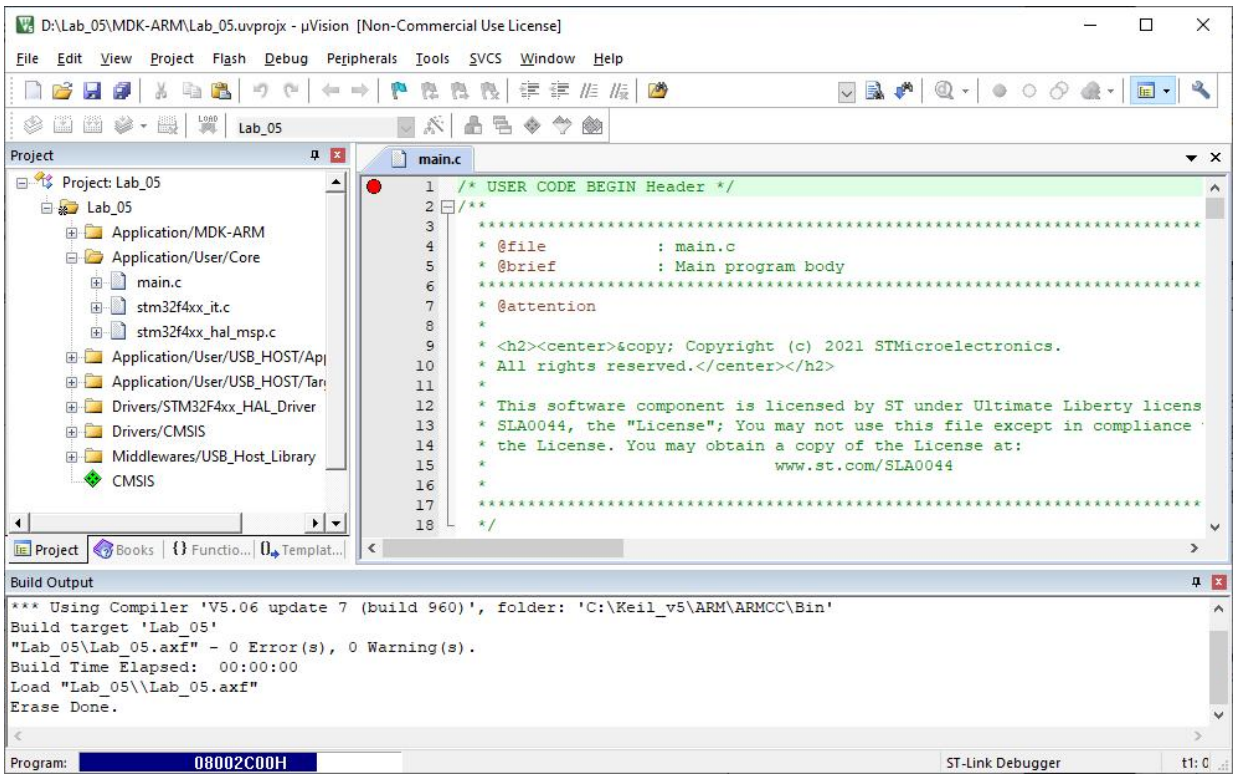



*Рис. 2.5. Процес програмування плати*

2.3. Про успішне завантаження об’єктного коду може свідчити висновок у секції “Build Output” (рис. 2.6).

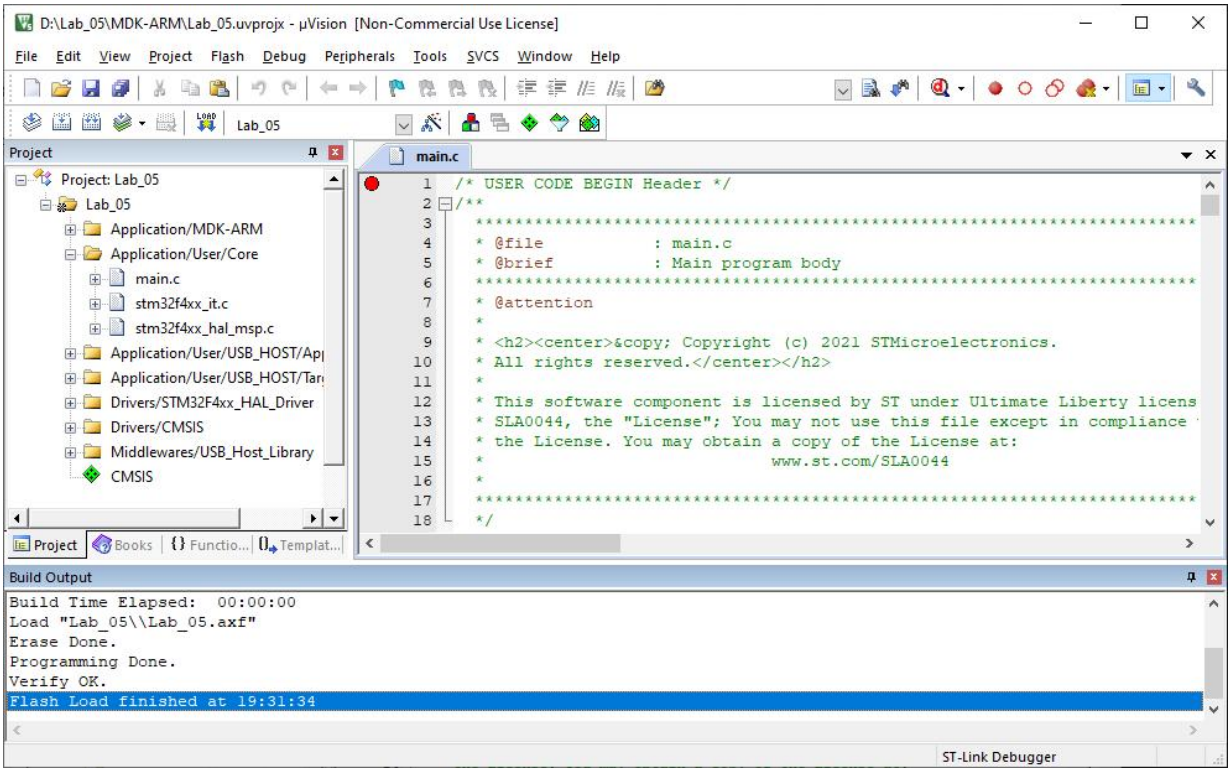



*Рис. 2.6. Успішне програмування плати*

## 3. Тестування світлодіодів

3.1. У тіло головної програми після коментаря “/* USER CODE BEGIN 3 */” вставити два рядки програмного коду для керування світлодіодами користувача/розробника:

```
HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_**);
HAL_Delay(200);
```

3.2. Замість символів “**” вставити один з номерів портів, занотованих у пункті 1.8. Наведений на рис. 3.1 фрагмент програми зумовлює блимання світлодіода з інтервалом приблизно в 1 секунду.

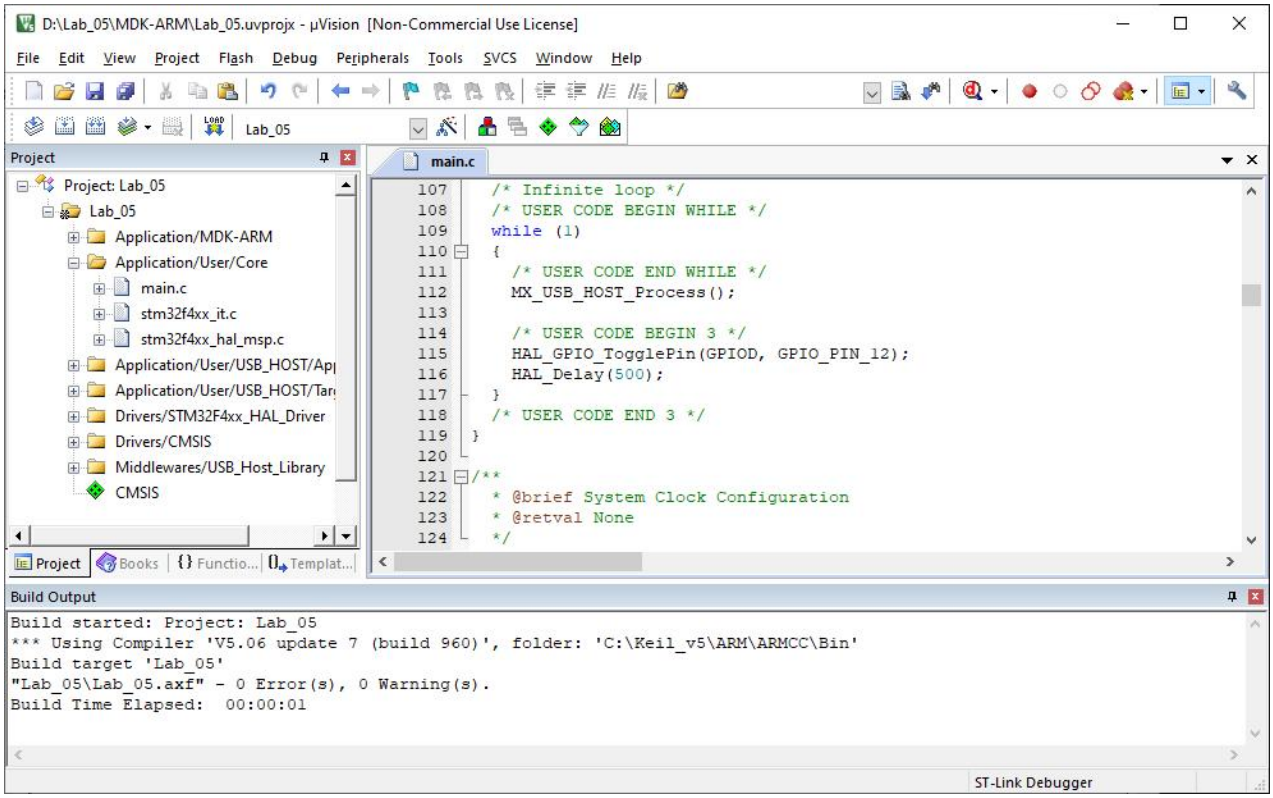



*Рис. 3.1. Фрагмент програмного коду для керування світлодіодом*

3.3. Скомпілювати код програми (F7, Project > Build Target) і запрограмувати плату (F8, Flash > Download).

3.4. На платі натиснути кнопку чорного кольору “Reset”. Зауважити блимання одного зі світлодіодів.

3.5. Послідовно змінюючи у програмі номери портів, перевірити справність решти світлодіодів користувача/розробника (див. рис. 3.2 та рис. 3.3).

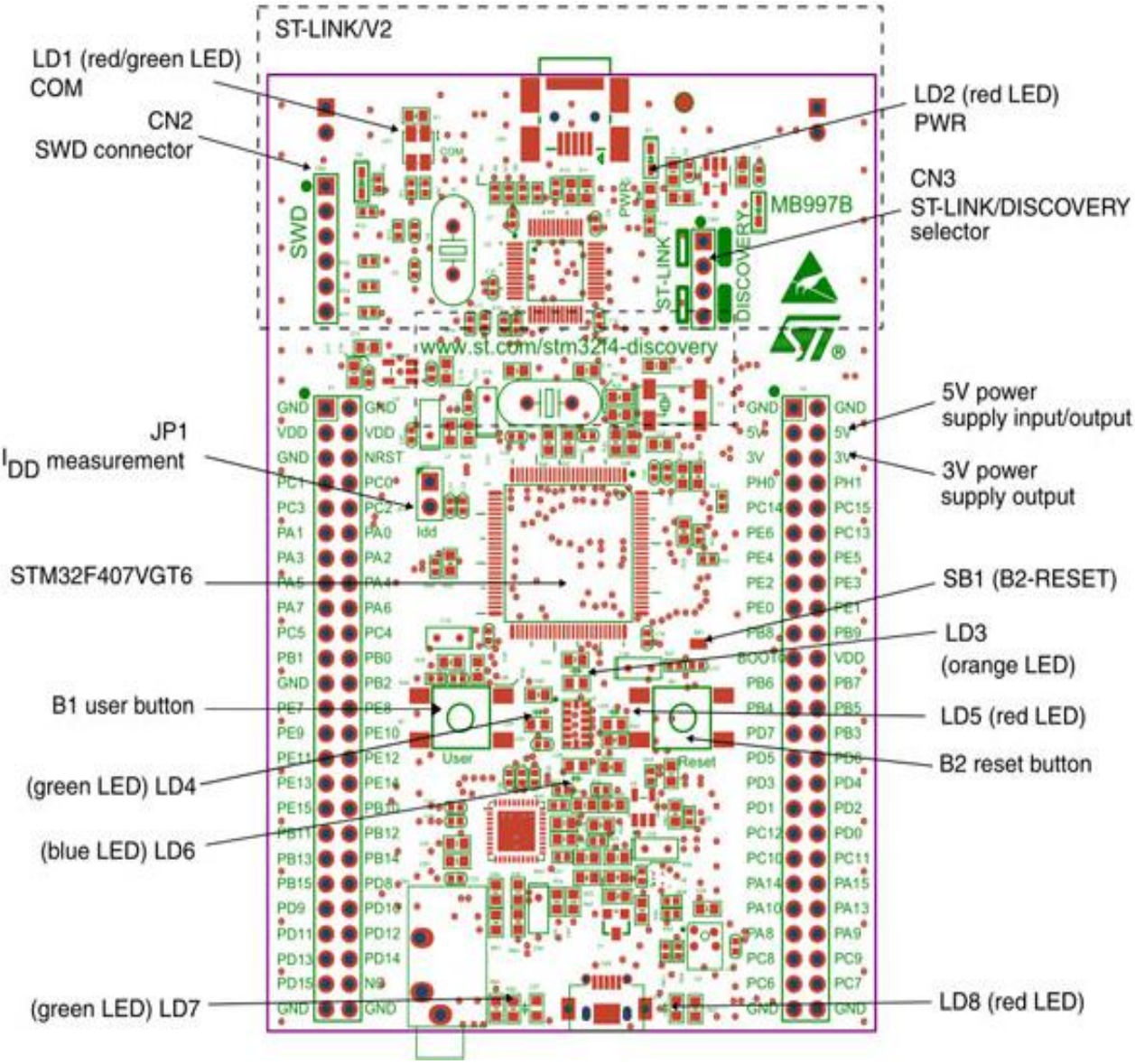



*Рис. 3.2. Розташування окремих компонентів на платі*

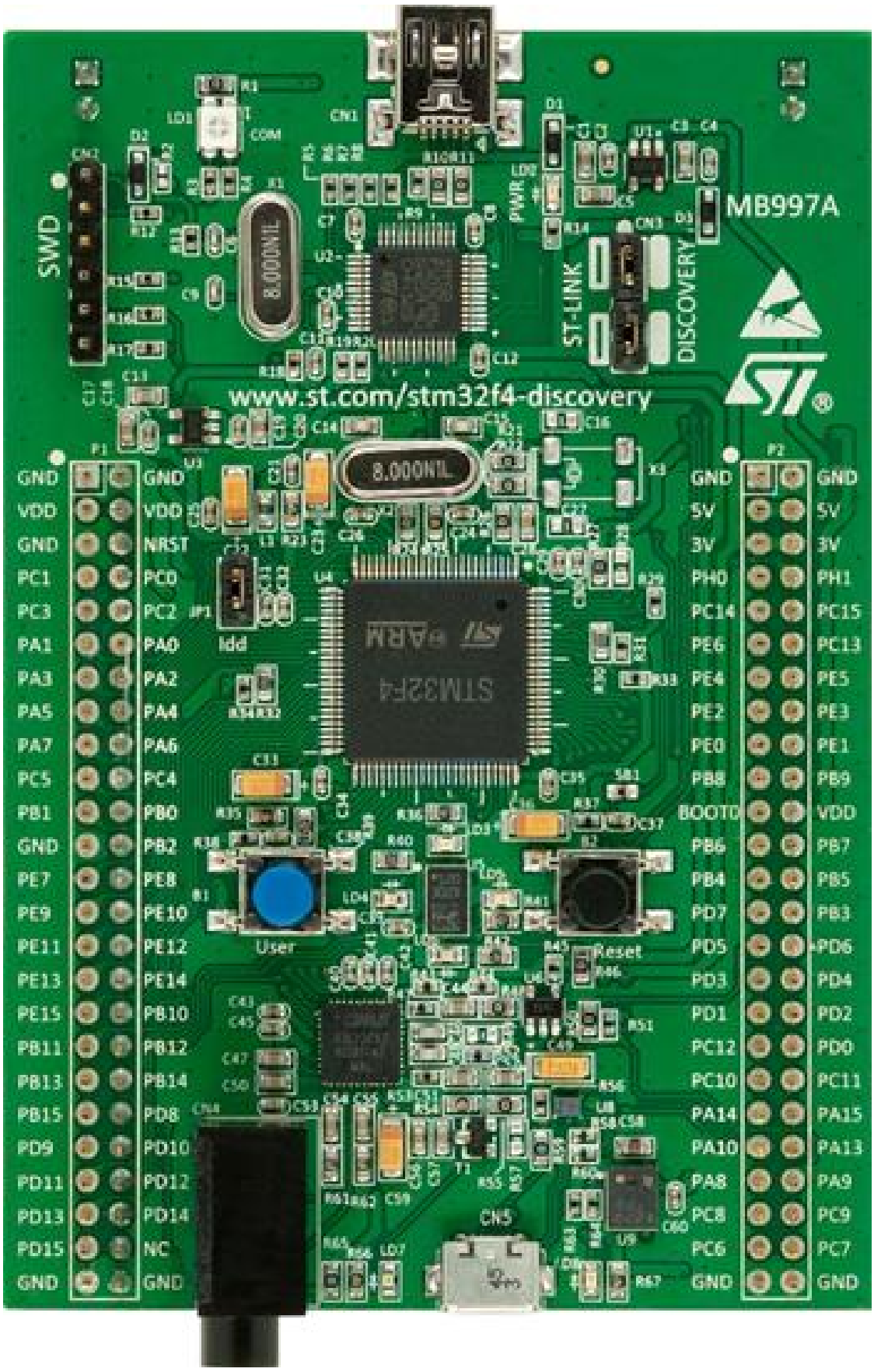



*Рис. 3.3. Зображення плати*

## ЗАВДАННЯ

1. Створити проєкт за допомогою STM32CubeMX і перевірити його правильність компілюванням в Keil MDK-ARM.

2. Перевірити можливість програмування плати STM32F407G-DISC1.

3. Послідовно перевірити роботу світлодіодів користувача / розробника LD3 – LD6.

4. Модифікувати програму відповідно до індивідуального завдання.

| Варіант завдання | Поєднання світлодіодів за кольором | Інтервал блимання для кожного світлодіода у секундах |
|---|---|---|
| 1 | зелений, червоний | 0,1 |
| 2 | червоний, помаранчевий | 0,2 |
| 3 | помаранчевий, синій | 0,3 |
| 4 | синій, зелений | 0,4 |
| 5 | зелений, помаранчевий | 0,5 |
| 6 | червоний, синій | 0,6 |
| 7 | червоний, зелений | 0,7 |
| 8 | помаранчевий,червоний | 0,8 |
| 9 | синій, помаранчевий | 0,9 |
| 10 | зелений, синій | 1,0 |
| 11 | помаранчевий, зелений | 1,1 |
| 12 | синій, червоний | 1,2 |
| 13 | зелений, червоний | 1,3 |
| 14 | червоний, помаранчевий | 1,4 |
| 15 | помаранчевий, синій | 1,5 |
| 16 | синій, зелений | 1,6 |
| 17 | зелений, помаранчевий | 1,7 |
| 18 | червоний, синій | 1,8 |
| 19 | червоний, зелений | 1,9 |
| 20 | помаранчевий,червоний | 2,0 |
| 21 | синій, помаранчевий | 2,1 |
| 22 | зелений, синій | 2,2 |
| 23 | помаранчевий, зелений | 2,3 |
| 24 | синій, червоний | 2,4 |
| 25 | зелений, червоний | 2,5 |
| 26 | червоний, помаранчевий | 2,6 |
| 27 | помаранчевий, синій | 2,7 |
| 28 | синій, зелений | 2,8 |
| 29 | зелений, помаранчевий | 2,9 |
| 30 | червоний, синій | 3,0 |

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.
2. Теоретична частина у тезовому викладі (стисло, без рисунків).
3. Індивідуальне завдання.
4. Опис послідовності виконання роботи. Результати проілюструвати скриншотами.
5. Висновки.
6. Додаток. Повний текст програми.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Який обсяг оперативної пам'яті міститься на платі STM32F407G-DISC1?
2. Код якого максимального розміру можна завантажити до пам’яті програм?
3. Якого типу пам’ять програм на платі?
4. На якій тактовій частоті працює мікроконтролер?.
5. Яка розрядність мікроконтролера?
6. Які компоненти плати забезпечують роботу зі звуком?
7. Завдяки якому компоненту на платі можна відслідковувати зміну положення у просторі?
8. Яким чином можна збільшити обсяг пам’яті для плати?

## ДЖЕРЕЛА ІНФОРМАЦІЇ

1. UM1472. User manual. Discovery kit with STM32F407VG MCU. URL: https://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf.

2. STM32F405xx. STM32F407xx. Datasheet – production data. URL: https://www.st.com/resource/en/datasheet/stm32f407vg.pdf.

3. Налагоджувальний комплекс DISCOVERY STM32F407G-DISC1. URL: http://www.kosmodrom.com.ua/el.php?name=STM32F407G-DISC1.

4. Налагоджувальна плата STM32F4 Discovery (STM32F407G-DISC1). URL: https://xcraft.com.ua/stm32f4-discovery-stm32f407g-disc1.

**Лабораторна робота № 6**

# ТЕСТУВАННЯ ОПЕРАТИВНОЇ ПАМ'ЯТІ ТА ПОСЛІДОВНОГО ІНТЕРФЕЙСУ

МЕТА РОБОТИ: ознайомитися зі способом перевірки працездатності оперативної пам'яті налагоджувальної плати STM32F407G-DISC1 з використанням універсального асинхронного інтерфейсу.

## ТЕОРЕТИЧНА ЧАСТИНА

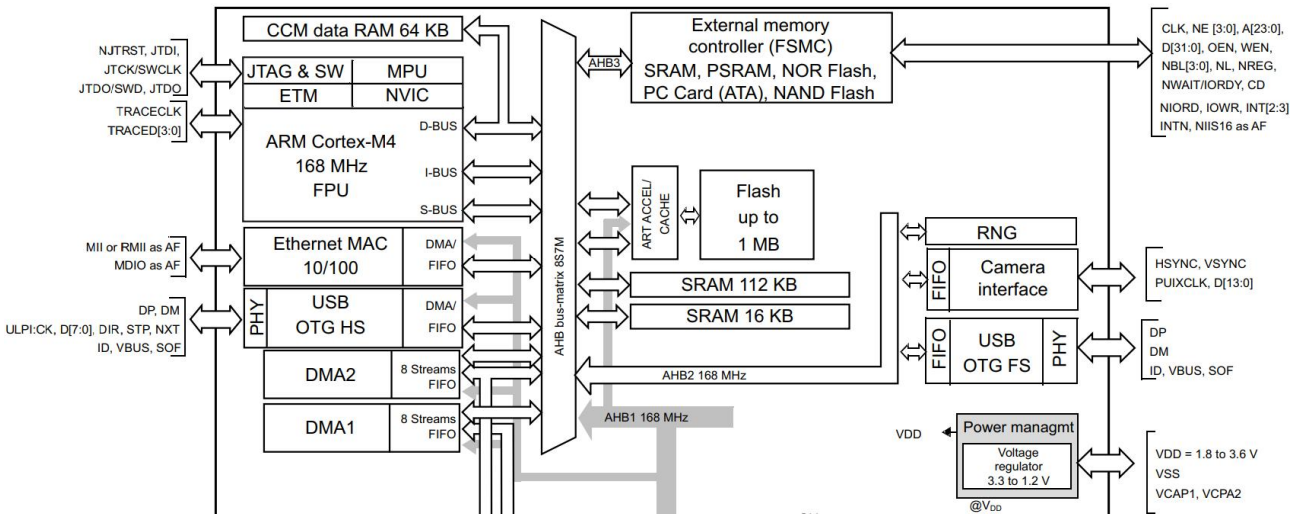
### Призначення та особливості функціонування окремих вузлів мікроконтролера STM32F407VGT6

Мікроконтролери STM32F407VGT6 базуються на високоефективному 32-бітному ядрі ARM Cortex-M4 RISC, що працює на частоті до 168 МГц. Ядро Cortex-M4 може виконувати операції з рухомою комою (FPU) з числами одинарної точності. Воно підтримує всі інструкції та типи обробки даних ARM одинарної точності, а також реалізує повний набір інструкцій DSP та містить блок захисту пам'яті (MPU), що підвищує безпеку програми.

Мікроконтролери STM32F407VGT6 містять високошвидкісну вбудовану пам'ять (флеш-пам'ять до 1 Мбайт, SRAM до 192 Кбайт), ще 4 Кбайт SRAM для зберігання резервних копій змінних, а також широкий спектр розширених ліній вводу-виводу та периферійних пристроїв.

Адаптивний акселератор пам'яті реального часу (ART Accelerator) – це прискоррювач пам'яті, оптимізований для використання у стандартних мікроконтролерах з ядром ARM Cortex-M4F, до яких входить сімейство STM32F4xx. Він призначений для вирівнювання продуктивності ядра і швидкодії флеш-пам'яті, яка, зазвичай, вимагає очікування з боку процесора, що працює на високій частоті. Для забезпечення повної продуктивності процесора 210 DMIPS на частоті 168 МГц акселератор здійснює попереднє вибирання інструкцій та організовує кеш переходів у 128-розрядний буфер пам'яті, що прискорює виконання коду програми. Як показують результати тестування CoreMark, продуктивність, яка досягається завдяки ART-акселератору, відповідає нульовому часу очікування вибирання інструкції з флеш-пам'яті для процесора з частотою до 168 МГц.

Блок захисту пам'яті (MPU) використовується для керування доступом CPU до пам'яті, щоб запобігати можливості небажаної зміни областей пам'яті, використовуваних іншим завданням. Масив пам'яті, керований блоком MPU, розділений на 8 захищених областей. Розмір захищеної області пам'яті може перебувати в діапазоні між 32 байт і 4 Гбайт. Блок MPU особливо корисний для додатків, у яких деякий критичний чи сертифікований код необхідно захищати від доступу з боку інших завдань, які зазвичай керуються однією з ОС реального часу (RTOS). Якщо програма намагається отримати доступ до області пам'яті, захищеної блоком MPU, то до RTOS надсилається спеціальний сигнал, який призводить до виконання заздалегідь заданих дій із обробки позаштатної ситуації. У робочому оточенні RTOS її ядро може динамічно оновлювати налаштування блоку MPU, ґрунтуючись на виконуваних завданнях. Блок MPU є необов'язковим і його можна відключити, щоб не застосовувати для виконання поточних завдань.




*Фрагмент функціональної схеми мікроконтролера*

Вбудована флеш-пам’ять. Мікроконтролери STM32F4xx можуть мати 256, 512, 768 Кбайт чи 1 Мбайт флеш -пам'яті для зберігання програм і даних. У мікроконтролері STM32F407VGT6 її обсяг становить 1 Мбайт.

Вбудований ОЗП (SRAM). Усі мікроконтролери STM32F4xx мають до 192 Кбайт системної SRAM, включаючи 64 Кбайт тісно пов'язаного (CCM) ОЗП даних (data RAM), до якого з боку процесора є прямий доступ. Доступ до всіх областей ОЗП може здійснюватися на частоті CPU з нульовим часом очікування. Додатково є область резервного ОЗП (backup SRAM) розміром 4 Кбайт. Доступ до цієї області здійснюється тільки з боку CPU. Її вміст захищено від можливих небажаних спроб запису.

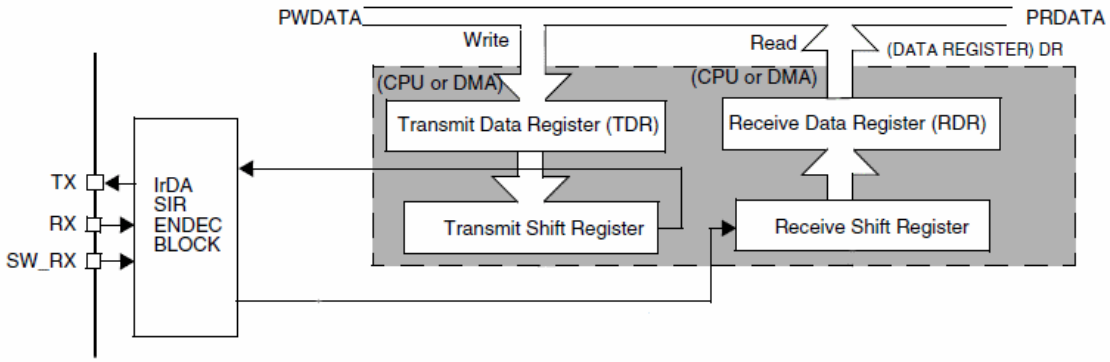
Контролер прямого доступу до пам'яті (DMA). Мікроконтролер містить два двопортові контролери DMA загального призначення (DMA1 і DMA2), кожен із яких може керуватися 8 потоками даних, такими як пам'ять-пам'ять, периферія-пам'ять і пам’ять-периферія. Контролери мають спеціалізовані буфери FIFO для периферійних модулів на шинах APB/AHB, підтримують пакетну передачу даних і спроєктовані для забезпечення максимальної пропускної здатності цих периферійних модулів. Два DMA-контролери підтримують кругову організацію буферів даних – таким чином не потрібно специфічного програмного коду для відстеження початку і кінця буфера. Крім того, підтримується функція подвійної буферизації, яка автоматизує використання і перемикання двох буферів пам'яті без написання будь-якого додаткового коду. Кожен потік даних має власний апаратний канал DMA-запитів з підтримкою програмного запуску. Конфігурування блоків DMA здійснюється програмним чином і не залежить від розміру масиву даних, що передаються. Контролер DMA може використовуватися з усіма основними периферійними модулями: SPI, I2S; I2C, USART, таймерами, АЦП, ЦАП, SDIO, криптоакселератором, інтерфейсом відеокамери (DCMI).

Гнучкий контролер статичної пам'яті (FSMC), вбудований у мікроконтролери STM32F4xx, має 4 виходи Chip Select, що підтримують такі режими: PCCard/Compact Flash, SRAM, PSRAM, NOR Flash і NAND Flash. Операції запису виконуються з використанням буфера FIFO. Максимальна частота доступу до зовнішніх пристроїв (fCLK) становить 60 МГц.

Універсальні синхронні/асинхронні приймачі-передавачі пропонують гнучкі засоби повнодуплексного обміну даними із зовнішніми пристроями. Вони забезпечують передавання та приймання послідовних даних у широкому діапазоні стандартних швидкостей. Програмована довжина слова даних – 8 або 9 біт. Підтримується налаштування стоп-бітів – 1 або 2.

Мікроконтролери STM32F407VGT6 містять чотири універсальні синхронні/асинхронні приймачі-передавачі (USART1, USART2, USART3, USART6) та два універсальні асинхронні приймачі-передавачі (UART4, UART5). Ці шість інтерфейсів забезпечують асинхронний зв'язок, підтримку IrDA SIR ENDEC, багатопроцесорний режим зв'язку, одножильний напівдуплексний режим зв'язку та мають можливість LIN Master/Slave. Інтерфейси USART1 та USART6 здатні спілкуватися зі швидкістю до 10,5 Мбіт/с. Інші доступні інтерфейси спілкуються на швидкості до 5,25 Мбіт/с. Інтерфейси USART1, USART2, USART3 та USART6 також забезпечують апаратне керування CTS- і RTS-сигналами, режим смарт-карт (сумісний із ISO 7816) та SPI-подібний зв'язок. Всі інтерфейси можуть обслуговуватися контролером DMA.

Передавальна і приймальна частини інтерфейсів UART/USART працюють незалежно одна від одної. Спільний у них лише тактовий генератор. Тобто можуть бути задіяні тільки приймач, або лише передавач. Приймач і передавач можуть працювати з різними пристроями і різними протоколами верхнього рівня. Однаковими у них мають бути швидкість і формат даних.




*Фрагмент функціональної схеми UART без вузлів керування*

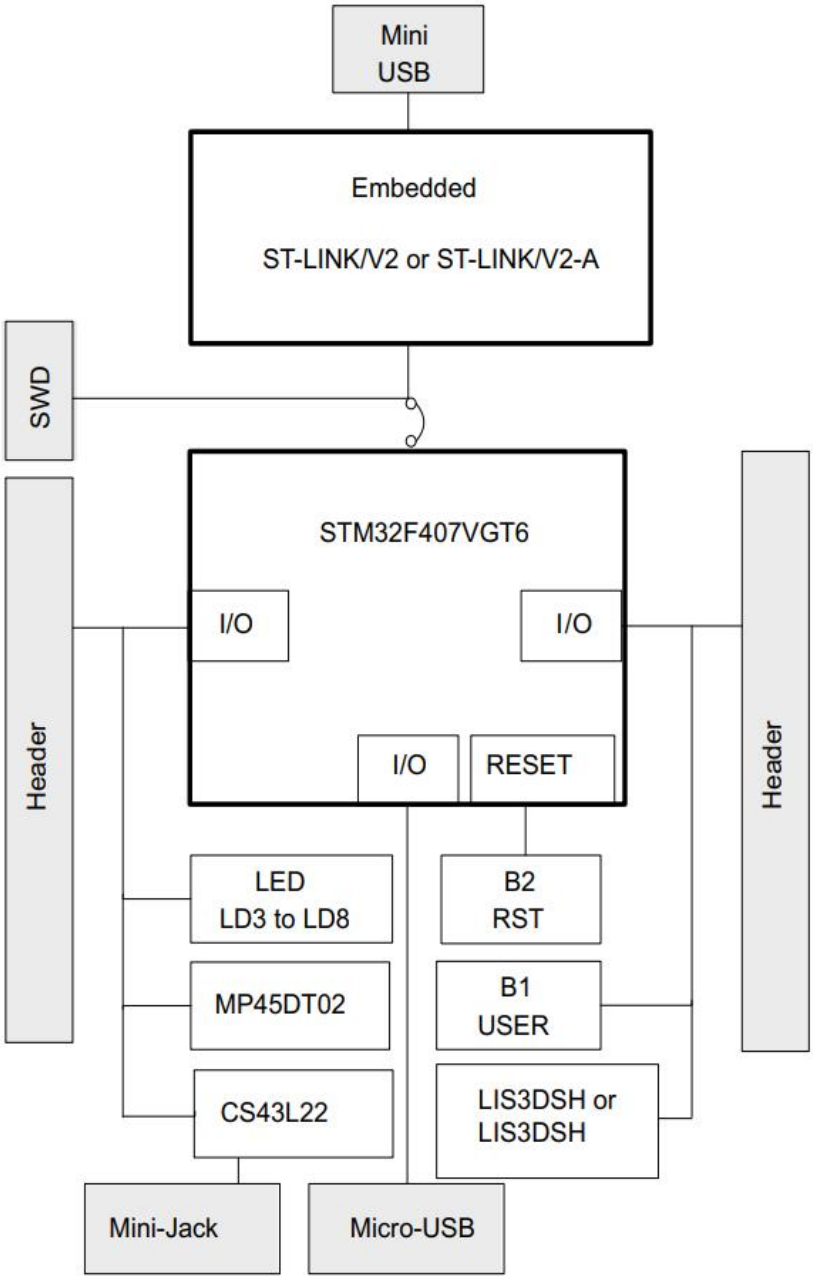
Передавач складається з 2-х регістрів: регістра зсуву (Transmit Shift Register) і буферного регістра (TDR). Слово даних завантажується у буферний регістр передавача (програмно доступний). Якщо передавання попереднього слова даних закінчено і регістр зсуву порожній, то наступне слово завантажується в нього, зсувається і побітово надходить на вихід TX. Щойно слово даних завантажено до регістра зсуву, буферний регістр звільняється і в нього може бути завантажено нове слово. Воно буде очікувати закінчення передачі та автоматично завантажиться у зсувний регістр.

Таким чином відбувається буферизація даних передавача. Це дозволяє реалізувати передавання без пауз між словами і дані будуть передаватися суцільним потоком.

Про закінчення передачі даних регістром зсуву і звільнення буферного регістра повідомляють спеціальні прапорці. За їх станом можуть бути сформовані переривання.

Приймальна частина пристрою теж складається з двох регістрів: регістра зсуву (Receive Shift Register) і буферного регістра (RDR). Зі входу RX дані побітово надходять у зсувний регістр і після формування повного слова завантажуються у буферний регістр приймача (програмно доступний). З нього зчитується отримане слово даних. Якщо дані надходять суцільним потоком, то після отримання слова має бути час, щоб прочитати його з буферного регістра. В іншому разі надійде нове слово даних, а старе буде втрачено.

Мікроконтролер STM32F407VGT6 є основним компонентом, довкола якого об’єднано інші складники налагоджувальної плати STM32F407G-DISC1.




*Функціональний склад налагоджувальної плати*

## ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

### 1. Створення консольної програми

1.1. Запустити Microsoft Visual Studio (рис. 1.1).

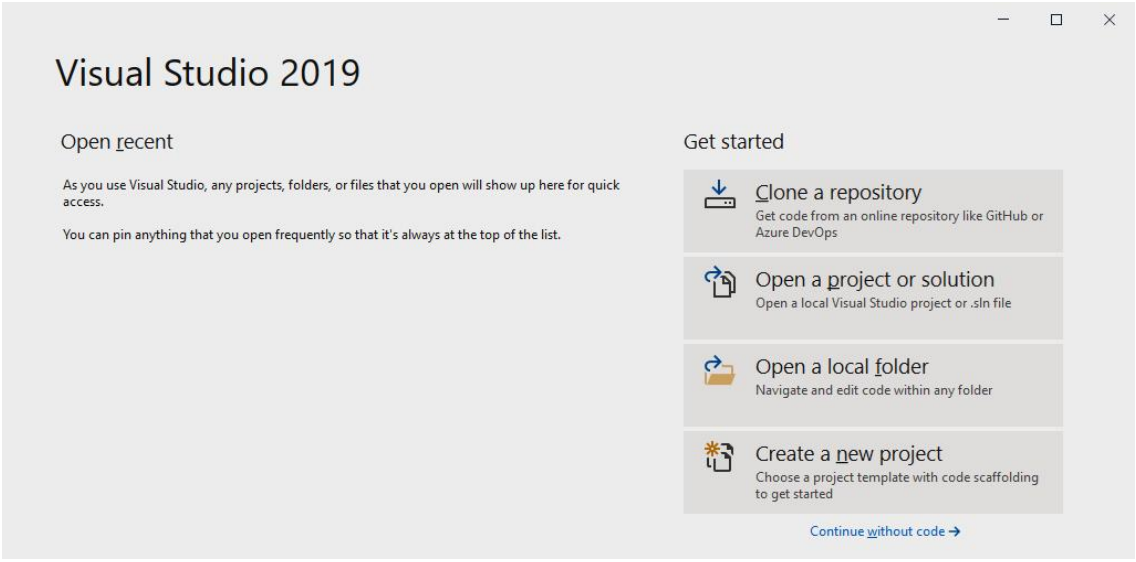



*Рис. 1.1. Стартове вікно Visual Studio*

1.2. Створити (рис. 1.2) та налаштувати (рис. 1.3) новий консольний проєкт мовою C++.

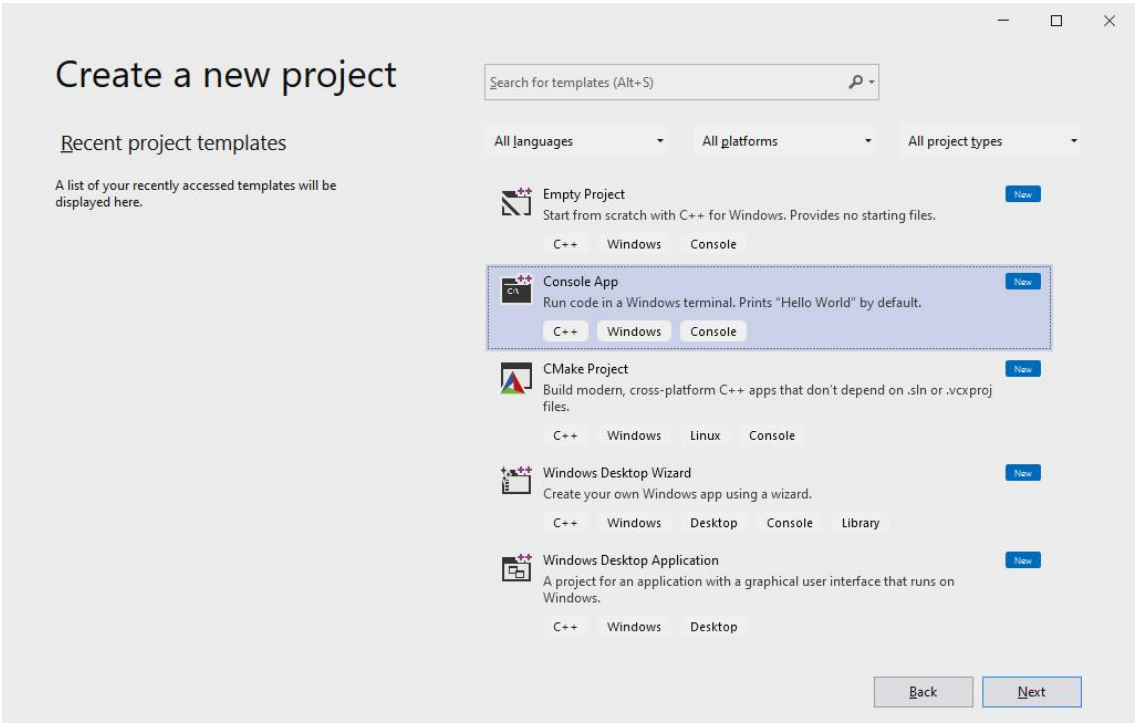



*Рис. 1.2. Вибір варіанта проєкту*

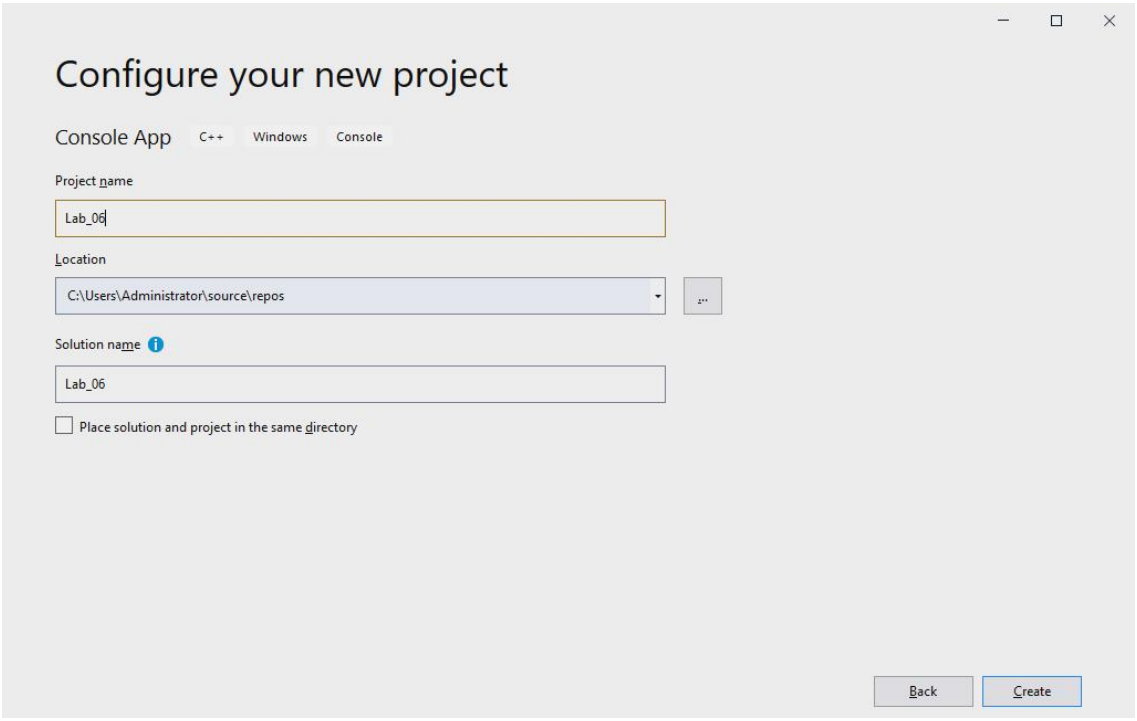



*Рис. 1.3. Попередні налаштування проєкту*

1.3. У заготовці проєкту (рис. 1.4) в полі директив препроцесора існуючі рядки замінити на необхідні бібліотеки та визначення:

```
#include <Windows.h>
#include <stdio.h>
#define  BUFFERLENGTH 256
```

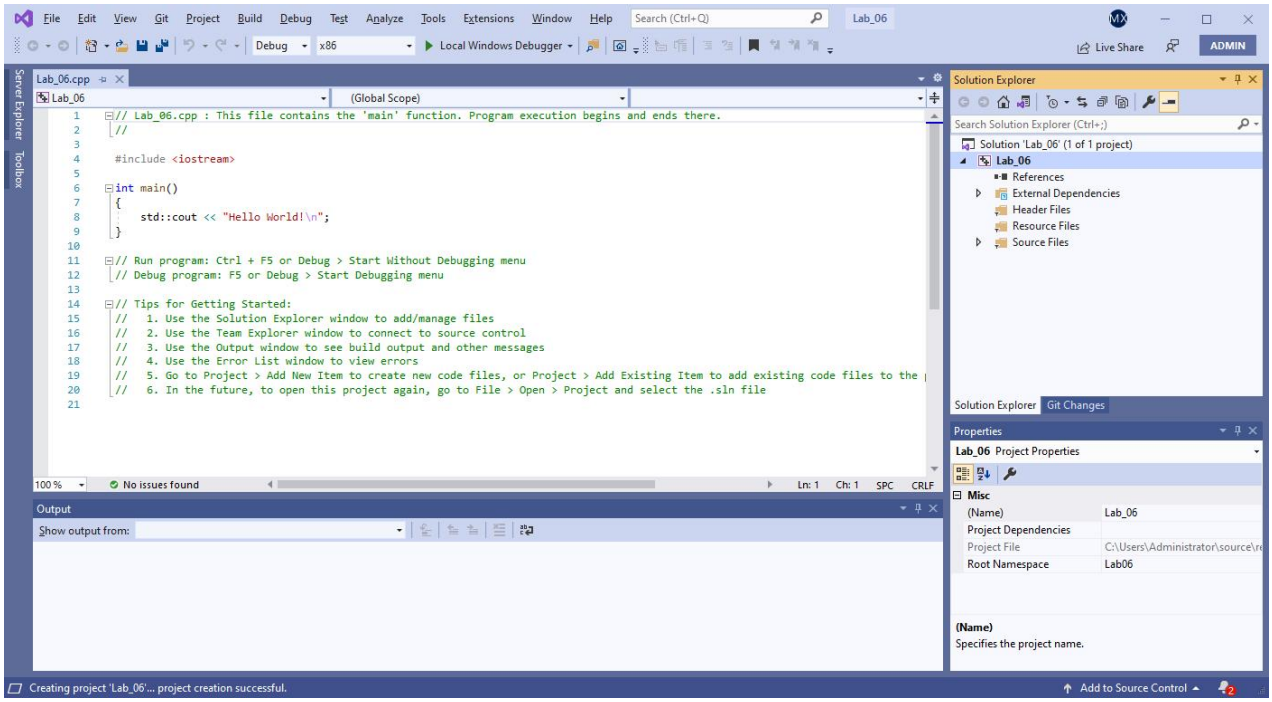



*Рис. 1.4. Заготовка проєкту*

1.4. Туди також додати блок глобальних змінних (рис. 1.5):

```
HANDLE hComm = INVALID_HANDLE_VALUE;         // Handle to the
Serial port
//char   ComPortName[] = “\\\\.\\COM24”;     // Name of the
Serial port(May Change) to be opened,
char   cComPortName[] = “COM5”;         // Name of the Serial
port(May Change) to be opened,
wchar_t   wComPortName[] = L”COM5”;          // Name of the
Serial port(May Change) to be opened,
BOOL   Status;
DCB dcbSerialParams = { 0 };            // Initializing DCB
structure
COMMTIMEOUTS timeouts = { 0 };

char   lpSendBuffer[BUFFERLENGTH] = { 0 };   // lpSendBuffer
should be  char or byte array
DWORD  dNoOFBytestoWrite;               // No of bytes to write
into the port
DWORD  dNoOfBytesWritten = 0;           // No of bytes written to
the port

DWORD dwEventMask;                  // Event mask to trigger
char  lpReceiveBuffer[BUFFERLENGTH] = { 0 }; // Buffer Containing
Rxed Data
char  lpRxBuffer[BUFFERLENGTH] = { 0 }; // Buffer Containing Rxed
Data
DWORD NoBytesRead;                  // Bytes read by ReadFile()
int i, j, n;

int iTime1, iTime2, iTime3, iTime4;
```

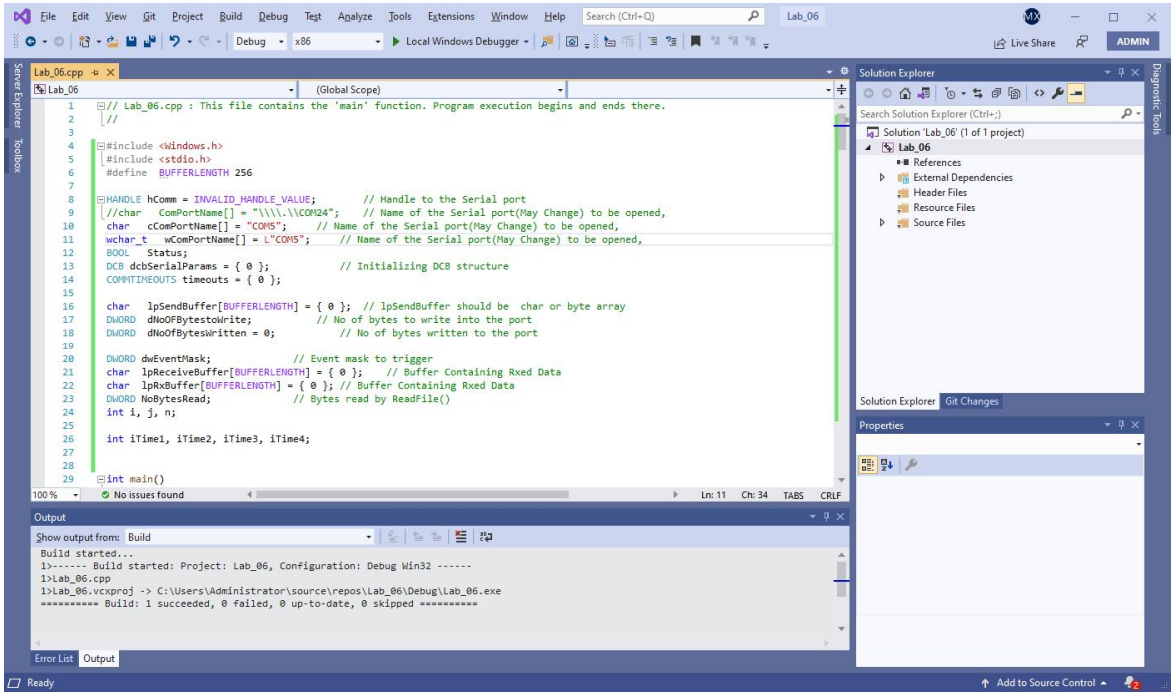



*Рис. 1.5. Блок глобальних змінних проєкту*

1.5. Замість рядка для виведення у консоль привітання в тіло головної функції вставити скопійований з методички програмний код (рис. 1.6):

```
    for (i = 0; i < BUFFERLENGTH; i++) {
        lpSendBuffer[i] = (char)i;
    }

    printf("\n\n +=========================================+");
    printf("\n |  Serial Transmission (Win32 API)        |");
    printf("\n +=========================================+\n");

    /*--------------- Opening the Serial Port -----------------
--------------------------*/

    hComm = ::CreateFile(wComPortName,      // Name of the Port
to be Opened
        GENERIC_READ | GENERIC_WRITE,       // Read/Write Access
        0,                          // No Sharing, ports can't be
shared
        NULL,                           // No Security
        OPEN_EXISTING,                  // Open existing port
only
        0,                          // No Overlapped I/O
        NULL);                          // Null for Comm Devices

    if (hComm == INVALID_HANDLE_VALUE) {
```

```
          if (GetLastError() == ERROR_FILE_NOT_FOUND)
          {
               printf("\n\n   Error! – Port %s can't be opened",
cComPortName);
          }
          else {
               printf("\n\n   Error! – invalid handle value!");
          }
          goto l_exit_proc;
     }
     else {
          printf("\n   Port %s Opened\n", cComPortName);
     }

     /*------------ Setting the Parameters for the SerialPort ----
---------------------------*/

     dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

     Status = ::GetCommState(hComm, &dcbSerialParams);
//retreives  the current settings

     if (Status == FALSE) {
          printf("\n\n   Error! In GetCommState()");
          goto l_exit_proc;
     }

     dcbSerialParams.BaudRate = CBR_115200;       //  baud rate
     dcbSerialParams.ByteSize = 8;                //  data size,
xmit and rcv
     dcbSerialParams.Parity = NOPARITY;           //  parity bit
     dcbSerialParams.StopBits = ONESTOPBIT;

     dcbSerialParams.fBinary = TRUE;
     dcbSerialParams.fDsrSensitivity = false;
     //dcbSerialParams.fParity = fParity;
     dcbSerialParams.fOutX = false;
     dcbSerialParams.fInX = false;
     dcbSerialParams.fNull = false;
     dcbSerialParams.fAbortOnError = TRUE;
     dcbSerialParams.fOutxCtsFlow = FALSE;
     dcbSerialParams.fOutxDsrFlow = false;
     dcbSerialParams.fDtrControl = DTR_CONTROL_DISABLE;
     dcbSerialParams.fDsrSensitivity = false;
     dcbSerialParams.fRtsControl = RTS_CONTROL_DISABLE;
     dcbSerialParams.fOutxCtsFlow = false;
```

```
     dcbSerialParams.fOutxCtsFlow = false;

     //Configuring the port according to settings in DCB
     Status = ::SetCommState(hComm, &dcbSerialParams);

     if (Status == FALSE) {
          printf("\n\n   Error! In Setting DCB Structure");
          goto l_exit_proc;
     }
     else {
          printf("\n   Setting DCB Structure Successfull\n");
          printf("\n       Baudrate = %d",
dcbSerialParams.BaudRate);
          printf("\n       ByteSize = %d",
dcbSerialParams.ByteSize);
          printf("\n       StopBits = %d",
dcbSerialParams.StopBits);
          printf("\n       Parity   = %d",
dcbSerialParams.Parity);
     }

     /*---------------- Setting Timeouts ------------------------
--------------------------*/

     timeouts.ReadIntervalTimeout = 50;
     timeouts.ReadTotalTimeoutConstant = 50;
     timeouts.ReadTotalTimeoutMultiplier = 10;
     timeouts.WriteTotalTimeoutConstant = 50;
     timeouts.WriteTotalTimeoutMultiplier = 10;

     if (::SetCommTimeouts(hComm, &timeouts) == FALSE) {
          printf("\n\n   Error! In Setting Time Outs");
          goto l_exit_proc;
     }
     else {
          printf("\n\n   Setting Serial Port Timeouts
Successfull\n");
          printf("\n       ReadIntervalTimeout         = %d",
timeouts.ReadIntervalTimeout);
          printf("\n       ReadTotalTimeoutConstant    = %d",
timeouts.ReadTotalTimeoutConstant);
          printf("\n       ReadTotalTimeoutMultiplier  = %d",
timeouts.ReadTotalTimeoutMultiplier);
          printf("\n       WriteTotalTimeoutConstant   = %d",
timeouts.WriteTotalTimeoutConstant);
```

```
          printf("\n       WriteTotalTimeoutMultiplier = %d",
timeouts.WriteTotalTimeoutMultiplier);
     }

     ::PurgeComm(hComm, PURGE_TXCLEAR | PURGE_RXCLEAR);

     /*--------- Writing a Character to Serial Port----------
-------------------------*/
     n = 1;
l_loop:;
     printf("\n\n %d pass.
==========================================\n", n);
     dNoOFBytestoWrite = sizeof(lpSendBuffer); // Calculating the
no of bytes to write into the port

     iTime1 = ::GetTickCount();

     Status = ::WriteFile(hComm,   // Handle to the Serialport
          lpSendBuffer,       // Data to be written to the port
          dNoOFBytestoWrite,  // No of bytes to write into the
port
          &dNoOfBytesWritten, // No of bytes written to the port
          0);

     if (Status == TRUE) {
          // WriteFile completed immediately.
          Printf("\n    %d bytes – Written to %s",
dNoOfBytesWritten, cComPortName);
     }
     else {
          printf("\n\n   Error %d in Writing to Serial Port",
GetLastError());
          goto l_exit_proc;
     }

     /*---------------- Setting Send Mask ----------------------
-------------------------*/

     Status = ::SetCommMask(hComm, EV_TXEMPTY); //Configure
Windows to Monitor the serial device for transmit buffer

     if (Status == FALSE) {
          printf("\n\n    Error! In Setting EV_TXEMPTY
CommMask");
          goto l_exit_proc;
     }
```

```
     //else {
          //printf("\n\n    Setting EV_TXEMPTY CommMask
successfull");
     //}

     /*---------------- Setting WaitComm() Event   --------------
-------------------------*/

     printf("\n    Waiting for Send Data");

     Status = ::WaitCommEvent(hComm, &dwEventMask, NULL); //Wait
for the character to be received

     if (Status == FALSE)
     {
          printf("\n\n    Error! In Setting WaitCommEvent()");
          goto l_exit_proc;
     }
     //else //If  WaitCommEvent()==True Read the Rxed data using
ReadFile();
     //{
     //}

     iTime2 = ::GetTickCount();
     printf(" %d miliseconds", iTime2 – iTime1);

     /*---------------- Setting Receive Mask ------------------
-------------------------*/

     Status = ::SetCommMask(hComm, EV_RXCHAR); //Configure
Windows to Monitor the serial device for Character Reception

     if (Status == FALSE) {
          printf("\n\n    Error! In Setting EV_RXCHAR CommMask");
          goto l_exit_proc;
     }
     //else {
          //printf("\n\n    Setting EV_RXCHAR CommMask
successfull");
     //}

     /*--------------- Setting WaitComm() Event   --------------
-------------------------*/

     printf("\n    Waiting for Data Reception");
```

```
     Status = ::WaitCommEvent(hComm, &dwEventMask, NULL); //Wait
for the character to be received

     /*------ Program will Wait here till a Character is received
------------------------*/

     if (Status == FALSE) {
          printf("\n\n    Error! In Setting WaitCommEvent()");
          goto l_exit_proc;
     } // if (Status == FALSE)
     else {

          iTime3 = ::GetTickCount();
          printf(" %d miliseconds", iTime3 - iTime2);

          i = 0;
          do {
               //Status = ::ReadFile(hComm, lpReceiveBuffer,
BUFFERLENGTH, &NoBytesRead, NULL);
               Status = ::ReadFile(hComm, lpReceiveBuffer, 1,
&NoBytesRead, NULL);
               if (Status) {
                    for (j = 0; j < (int)NoBytesRead; j++) {
                         lpRxBuffer[i++] = lpReceiveBuffer[j];
                    }
               }
          } while (NoBytesRead > 0);

          iTime4 = ::GetTickCount();
          printf("\n    %d bytes Received from %s.\n\nTotal time
is %d miliseconds", i, cComPortName, iTime4 - iTime1);

          n++;
          if (n < 10) goto l_loop;
          //        if (n < 1024) goto l_loop;
     } // if (Status == FALSE)

l_exit_proc:;
     if (hComm != INVALID_HANDLE_VALUE) CloseHandle(hComm);
     //Closing the Serial Port
     printf("\n ========================================\n");

     system("pause");

     return 0;
```

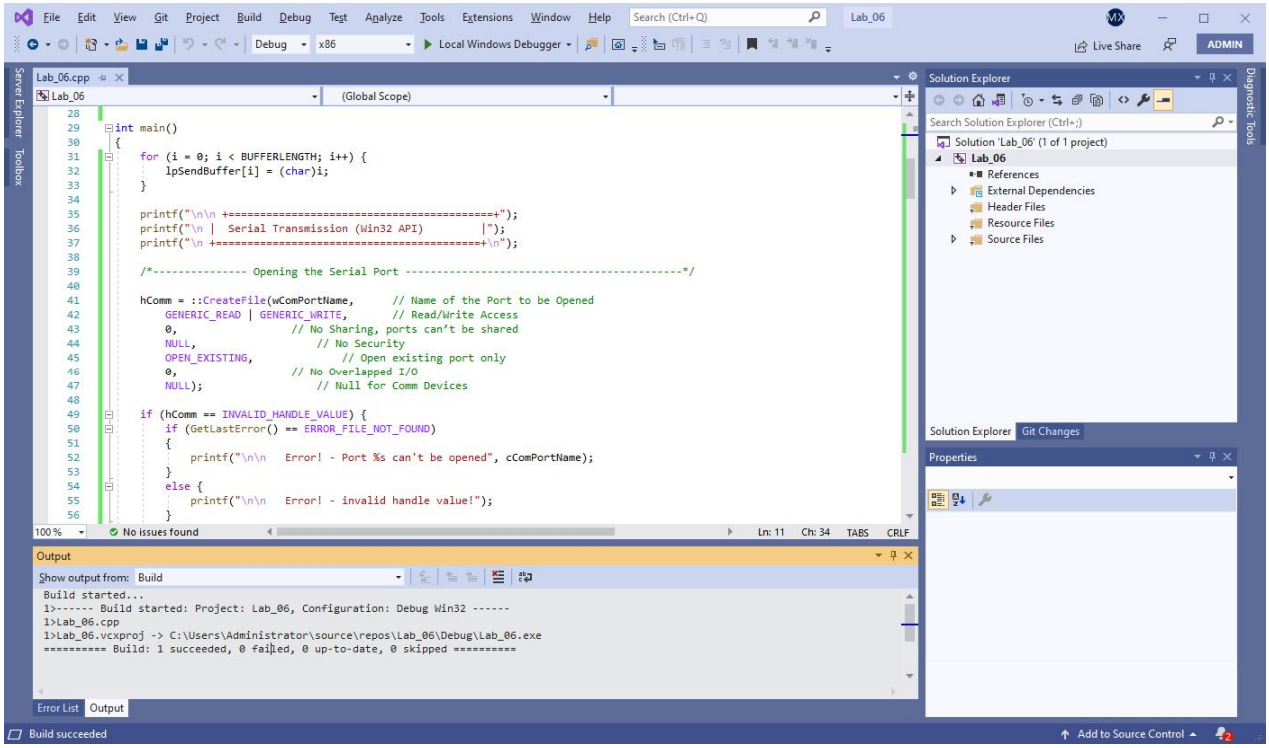



*Рис. 1.6. Фрагмент головної функції проєкту*

1.6. Скомпілювати проєкт, щоб переконатись у відсутності помилок (див. рис. 1.6) і запустити для тестового виконання (рис. 1.7 та рис. 1.8).

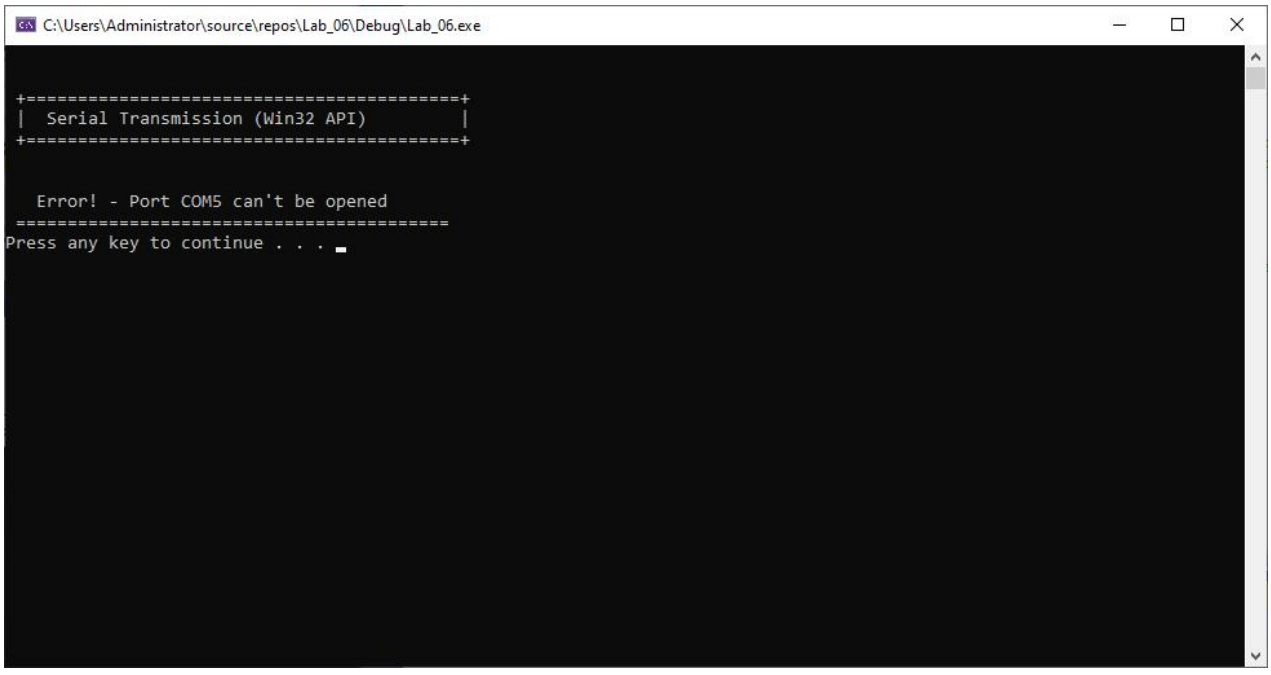



*Рис. 1.7. Тестове виконання консольної програми (порт не відкритий)*

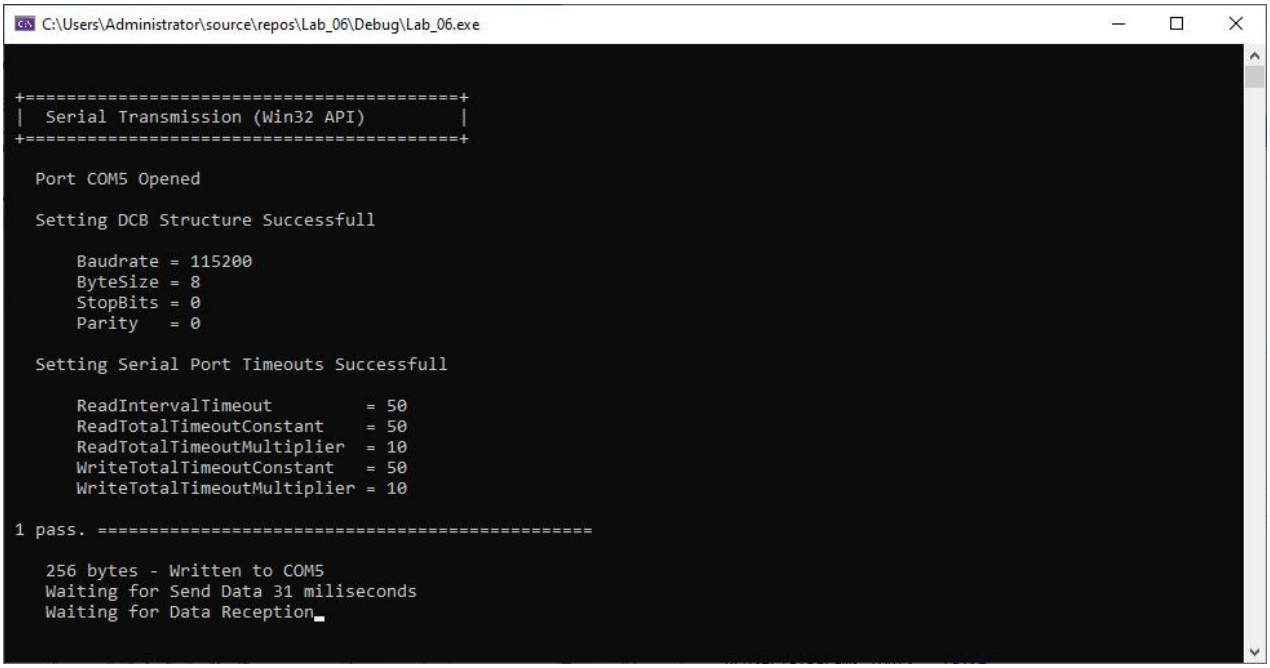



*Рис. 1.8. Тестове виконання консольної програми (порт відкритий)*

1.7. Закрити Visual Studio.

## 2. Підготовка проєкту для налагоджувальної плати

2.1. Запустити автоматизоване робоче місце розробки STM32CubeMX (рис. 2.1).

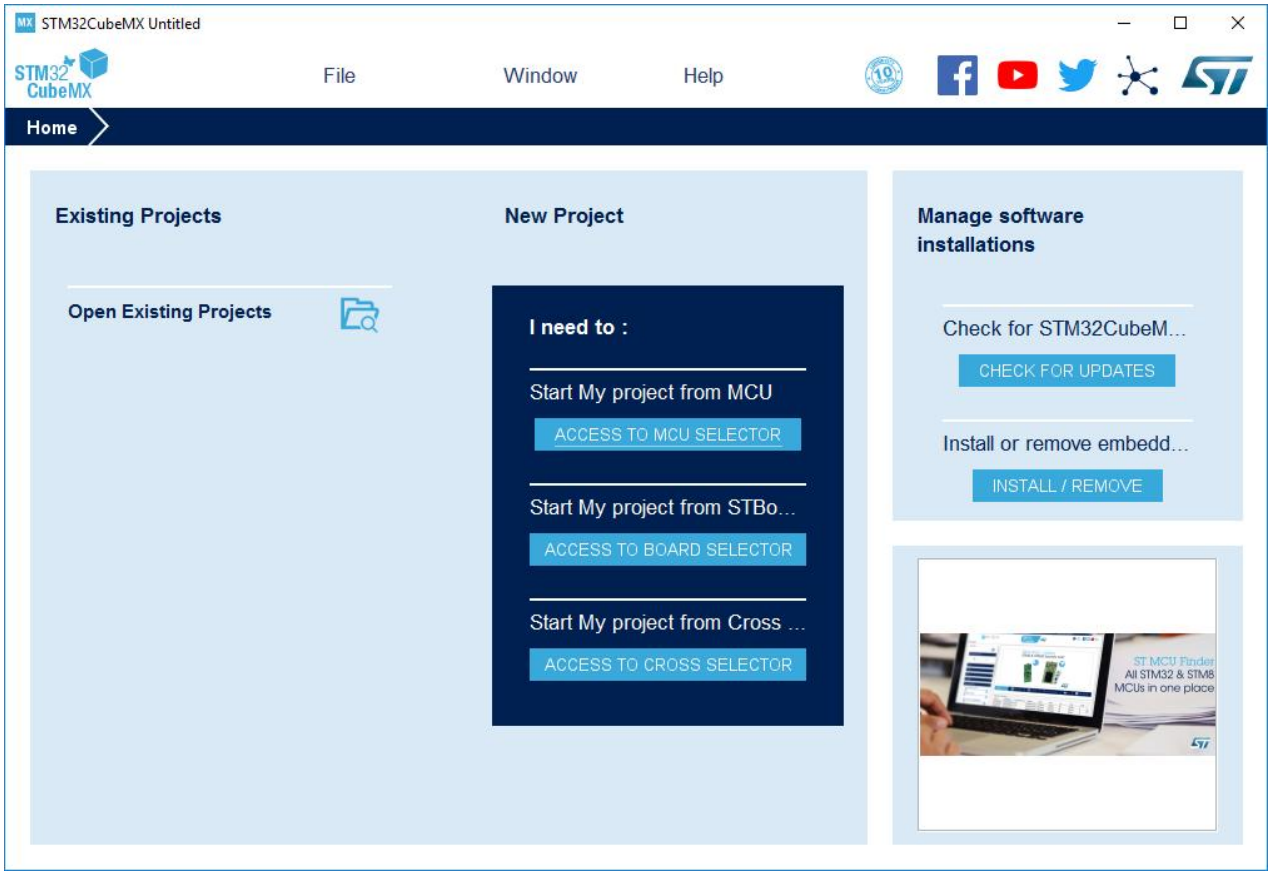



*Рис. 2.1. Стартове вікно STM32CubeMX*

2.2. У меню “New Project” вибрати роботу з платами ST “Start My project from STBoard”, натиснувши кнопку доступу до вибору плати “ACCESS TO BOARD SELECTOR” (див. рис. 2.1). Дочекатися завантаження необхідної інформації з Інтернету (рис. 2.2) і відкриття вікна вибору плати (рис. 2.3).

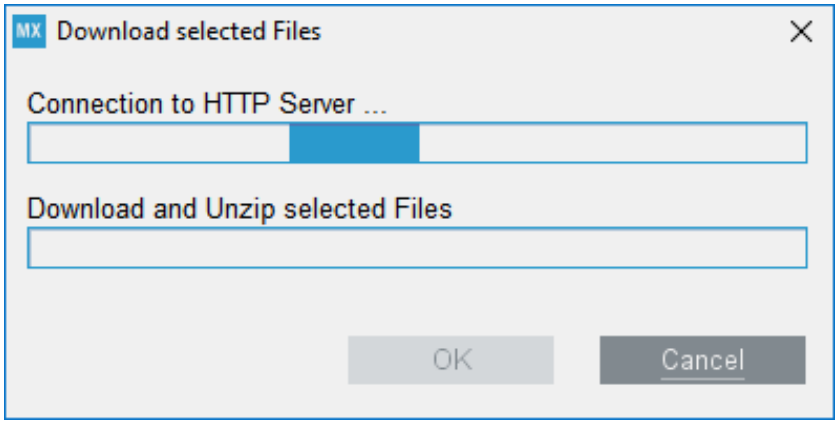



*Рис. 2.2. Оновлення даних*

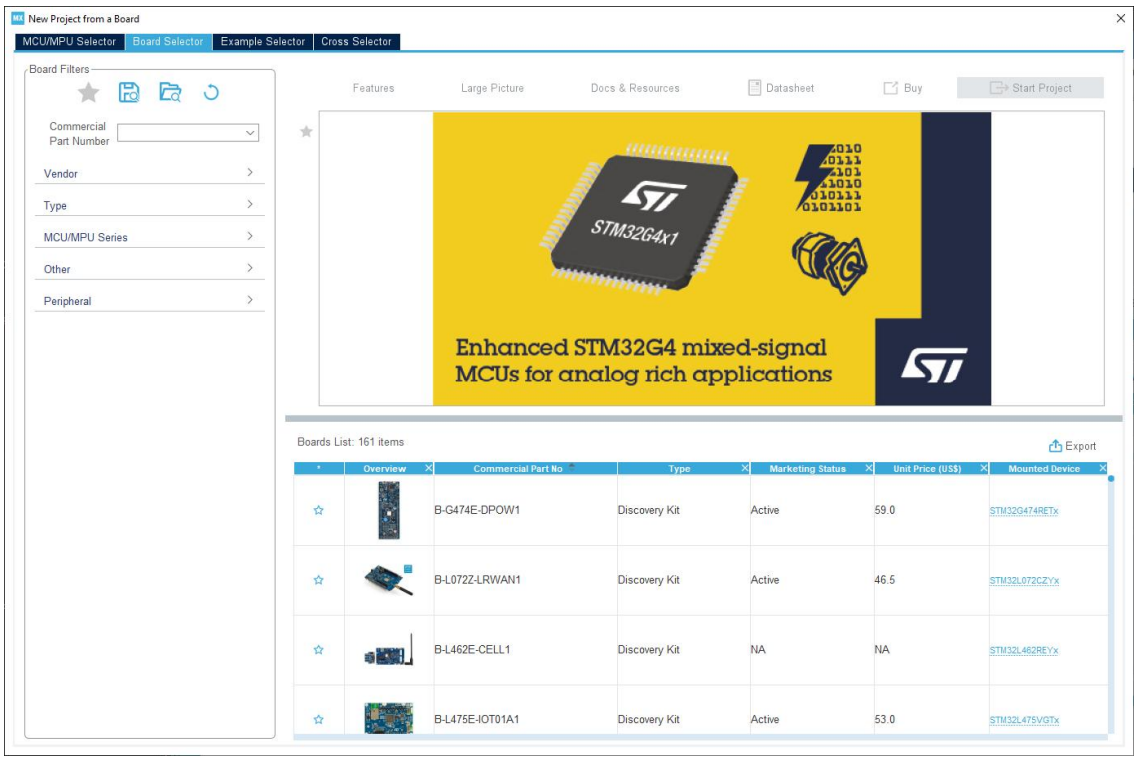



*Рис. 2.3. Вікно вибору плати*

2.3. У вкладці “Board Selector” в розділі “Board Filters” розкрити список “Commercial Part Number” і вибрати плату: STM32F407G-DISC1 (рис. 2.4).

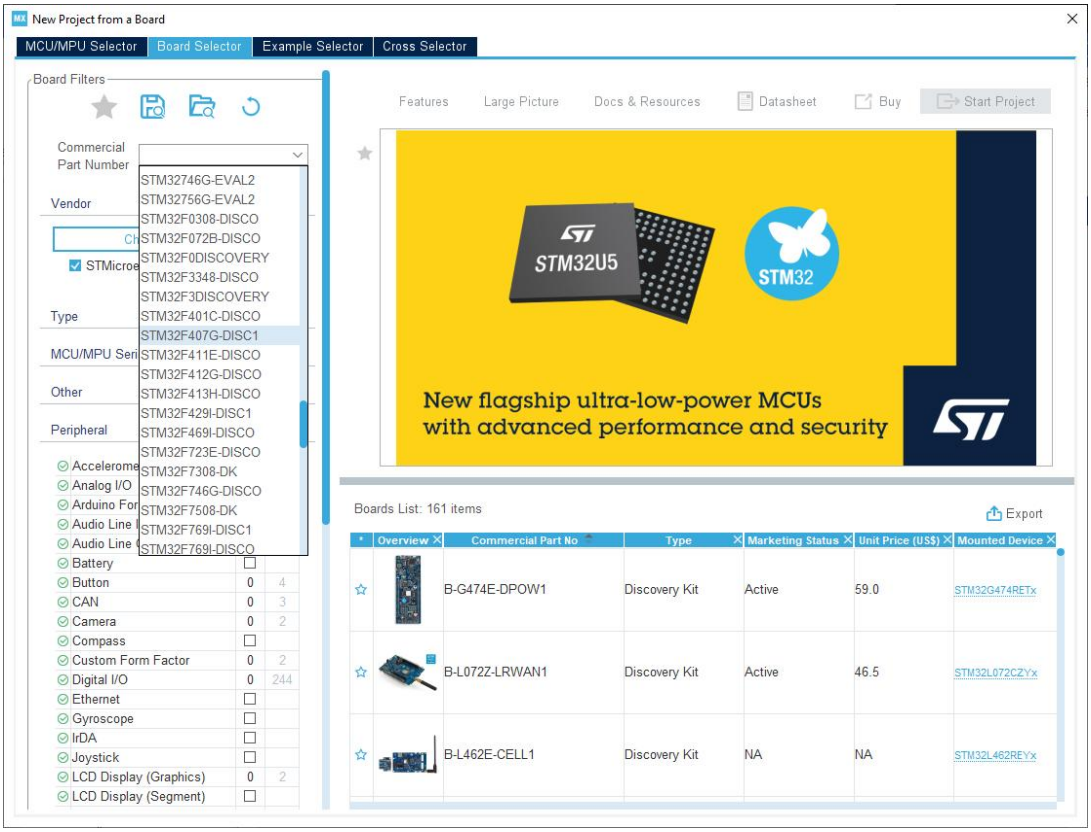



*Рис. 2.4. Вибір необхідного апаратного модуля*

2.4. У розділі “Board List” (рис. 2.5, внизу) натиснути на зображенні плати чи на назві “STM32F407G-DISC1” і отримати її короткий опис та основні характеристики (рис. 2.6). У правому верхньому куті вікна натиснути кнопку “Start Project”.

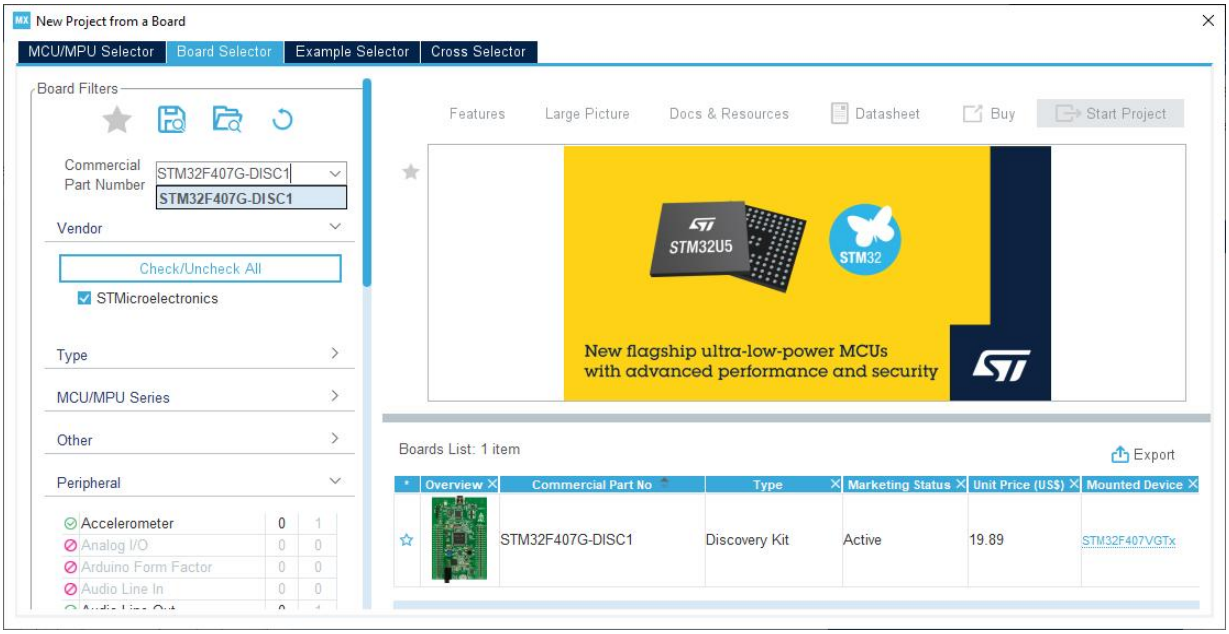



*Рис. 2.5. Вікно вибраної плати*

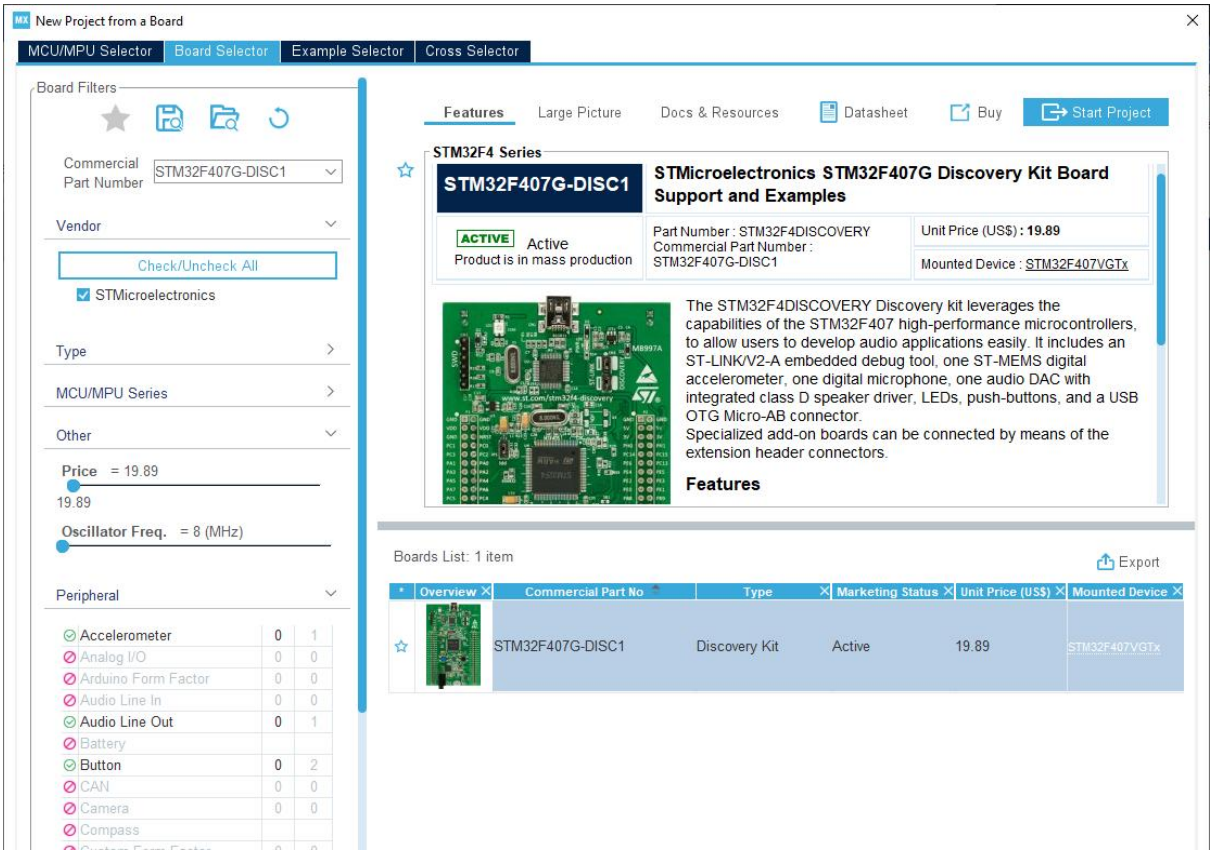



*Рис. 2.6. Короткий опис плати STM32F407G-DISC1*

2.5. Погодитися із пропозицією завантаження та початкової ініціалізації типових параметрів плати (рис. 2.7). Їх оновлення відбудеться через інтернет.

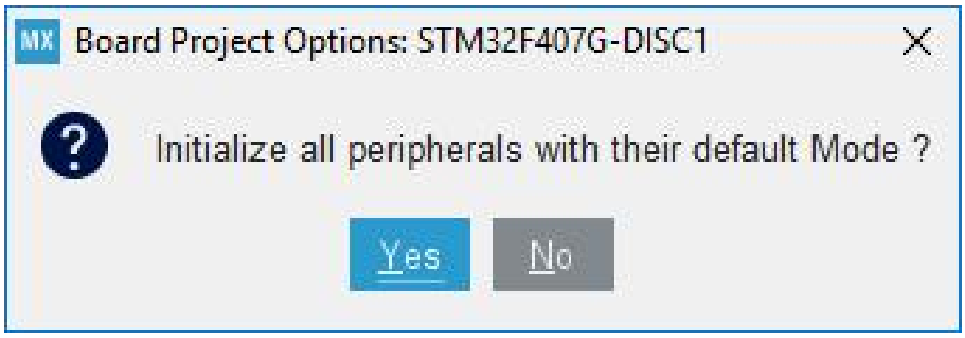



*Рис. 2.7. Пропозиція ініціалізації параметрів плати*

2.6. У вікні заготовки проєкту з типовими параметрами плати (рис. 2.8) у вкладці “Categories” розкрити меню “System Core” і натиснути на “RCC” – Reset and Clock Controler (рис. 2.9).

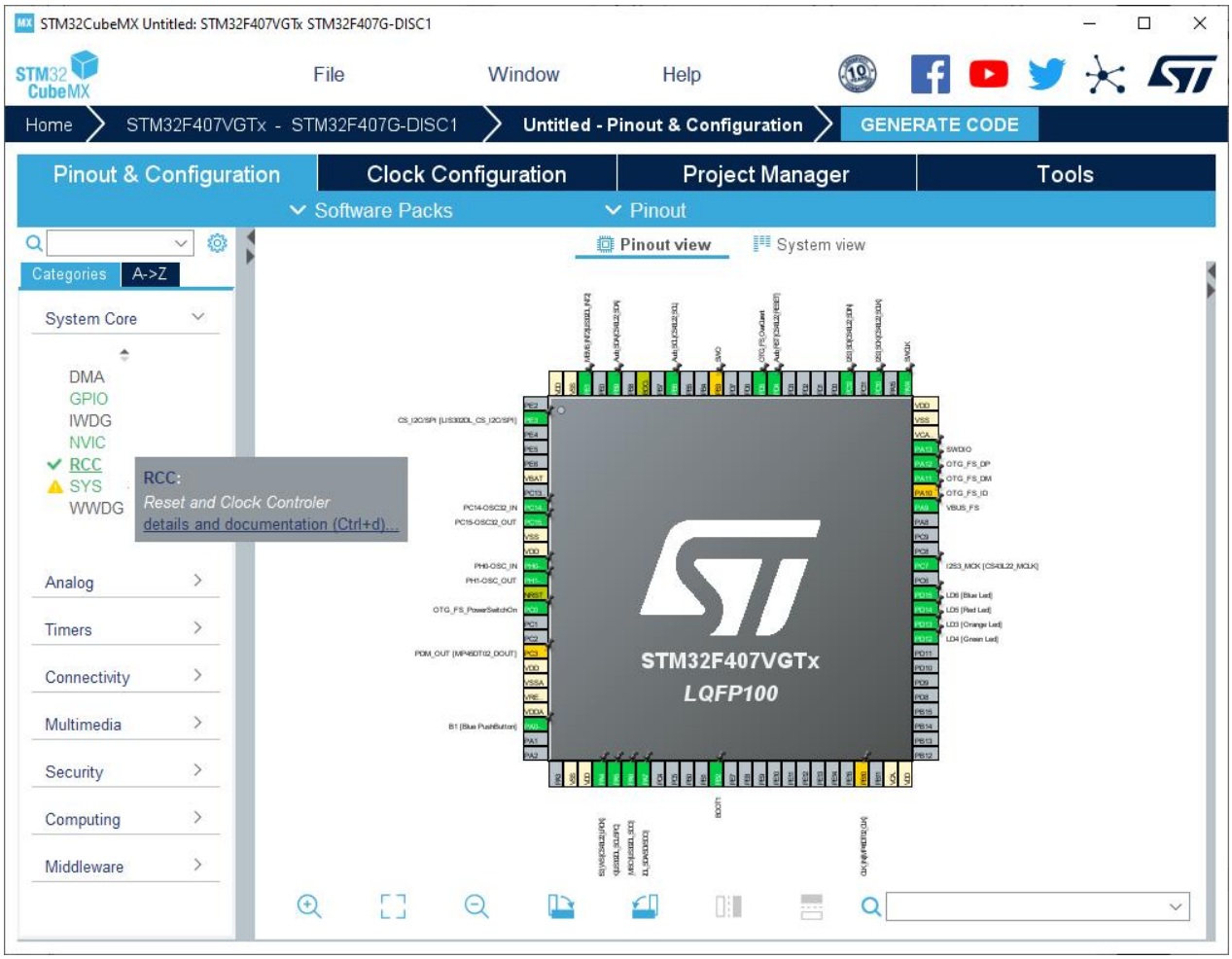



*Рис. 2.8. Проєкт з типовими параметрами плати*

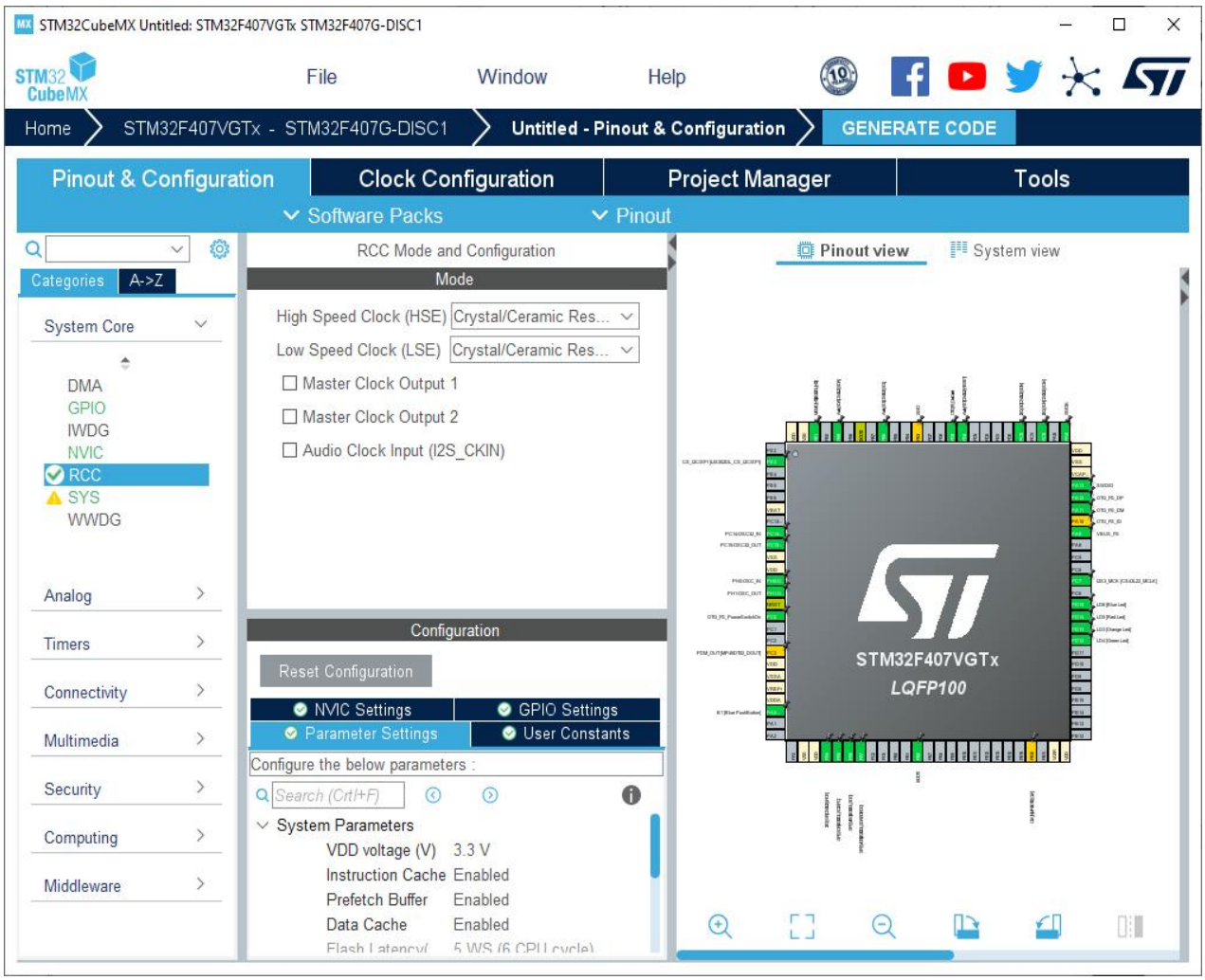



*Рис. 2.9. Вибір пункту меню “Reset and Clock Controler”*

2.7. У розділі “RCC Mode and Configuration” (рис. 2.10) знайти рядок “Low Speed Clock” і встановити значення “Disable”. Цей крок забороняє використання кварцового резонатора частотою 32,768 кГц, призначеного для синхронізації годинника реального часу мікроконтролера, оскільки плата STM32F407G-DISC1 не містить цього елемента (позиція X3 на платі порожня).

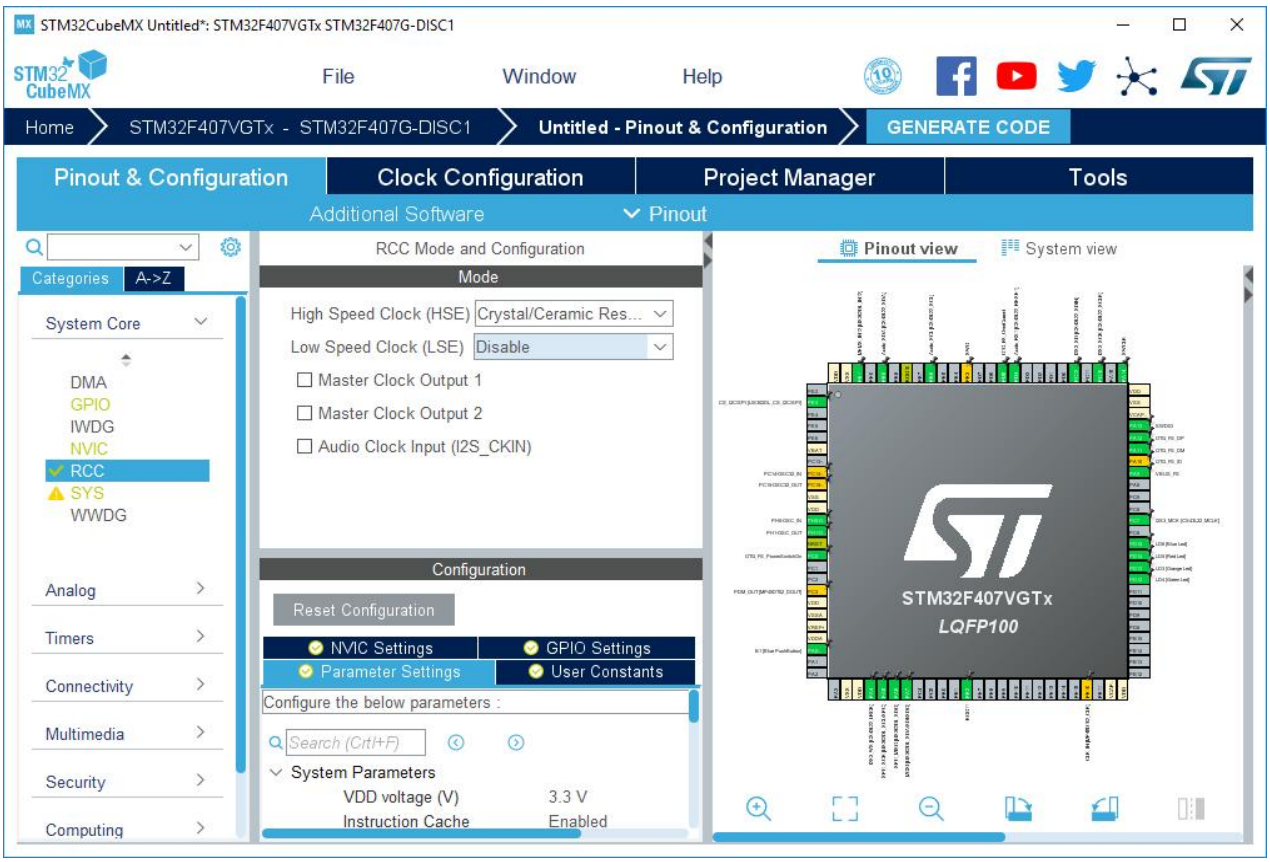



*Рис. 2.10. Відключення кварцу 32 кГц*

2.8. У вкладці “Categories” розкрити меню “Connectivity” і вибрати вільний пристрій USART2 (рис. 2.11).

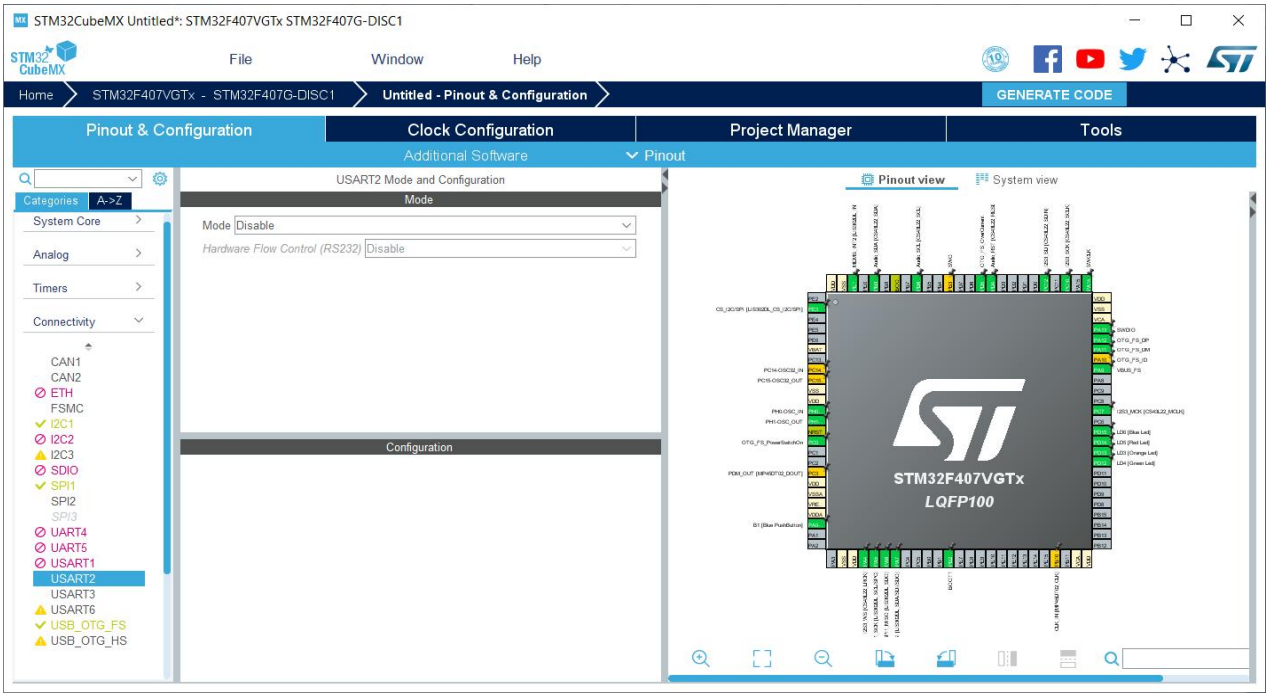



*Рис. 2.11. Вибір послідовного інтерфейсу*

2.9. У розділі “USART Mode and Configuration” вибрати роботу в асинхронному режимі (рис. 2.12).

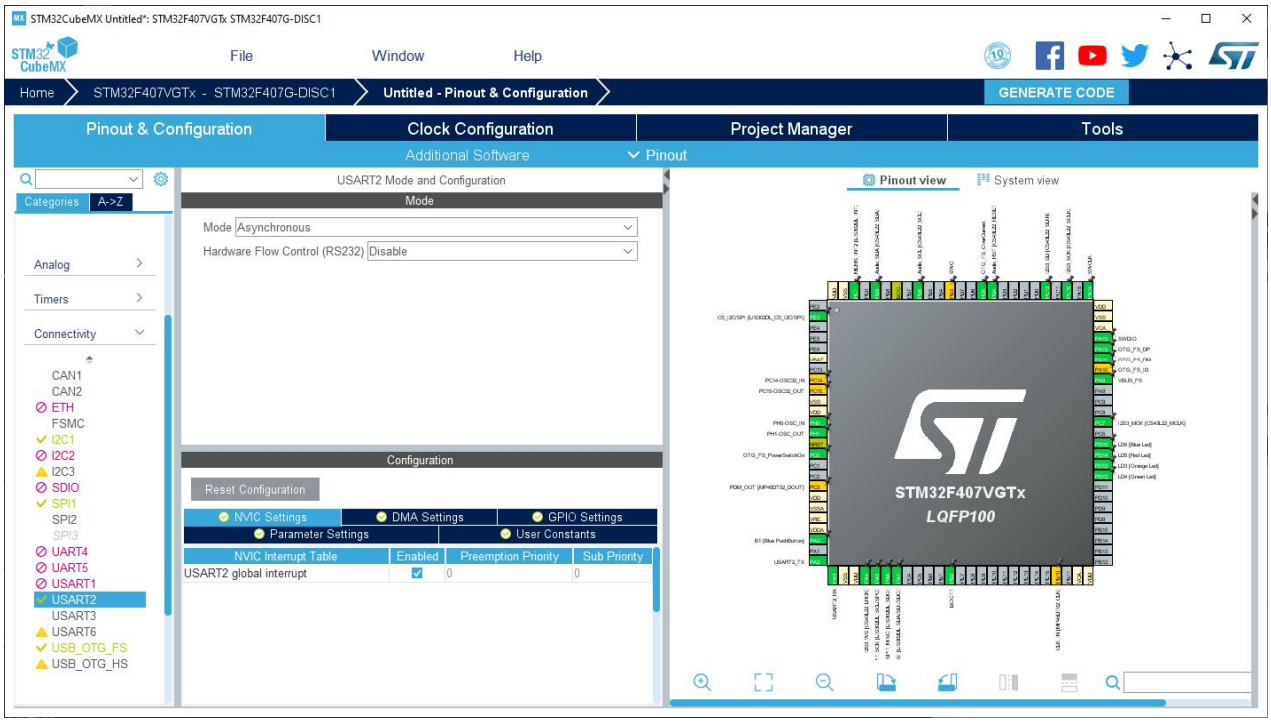



*Рис. 2.12. Вибір роботи в асинхронному режимі*

2.10. Звернути увагу, що на зображенні мікроконтролера з’явилися два нові задіяні виводи (рис. 2.13).

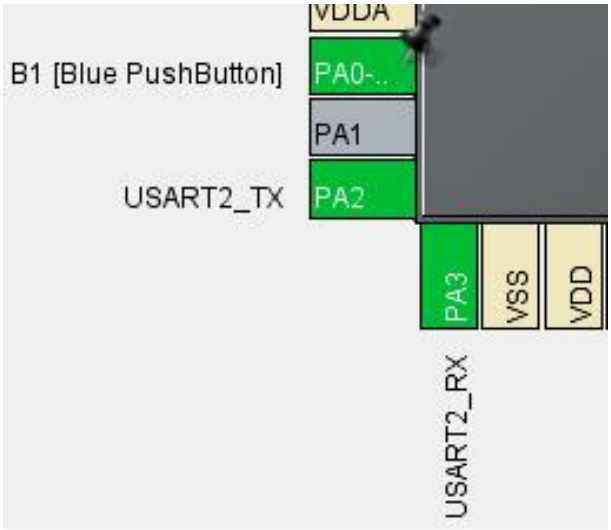



*Рис. 2.13. Виводи USART2 мікроконтролера*

2.11. Для реалізації послідовного інтерфейсу з використанням апаратних переривань внизу розділу “USART Mode and Configuration” у вкладці “NVIC Settings” встановити позначку “Enabled” (див. рис. 2.12).

2.12. У правому верхньому куті вікна конфігурування проєкту (див. рис. 2.12) натиснути кнопку “GENERATE CODE”. Погодитися з попередженням про необхідність присвоєння проєкту назви та внесення основних параметрів (рис. 2.14).


*Рис. 2.14. Попередження менеджера проєктів*

2.13. Літерами англійської абетки ввести назву проєкту (без пробілів), вибрати для нього місце на диску, а також вказати тип компілятора, для якого готуємо пакет проєкту – IDE MDK-ARM (рис. 2.15).

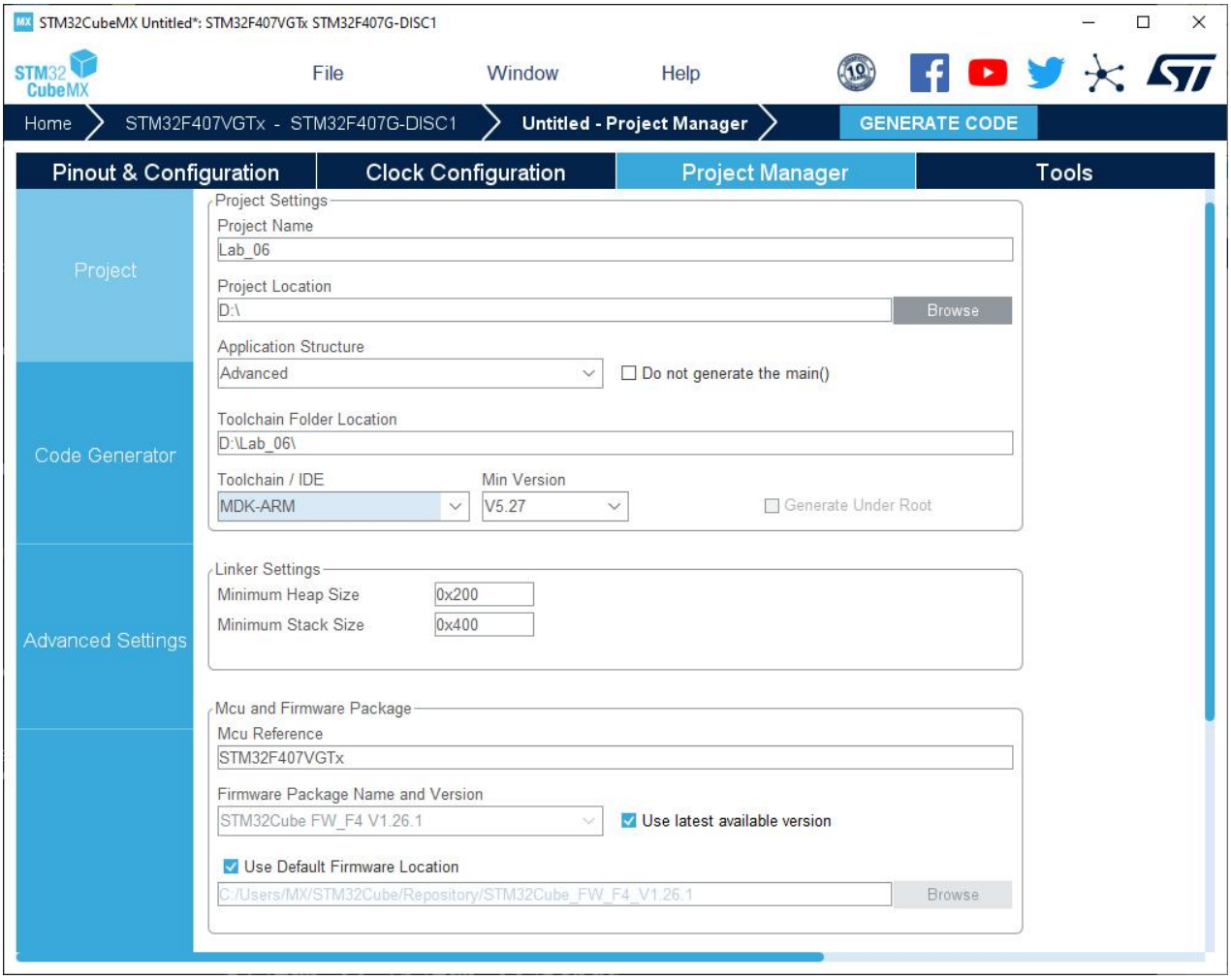



*Рис. 2.15. Основні параметри проєкту*

2.14. Знову натиснути кнопку GENERATE CODE і спостерігати за процесом формування заготовки проєкту для вибраного компілятора та копіюванням необхідних бібліотек (рис. 2.16).

Увага! Можлива ситуація (зазвичай виникає, якщо місце для проєкту вибирається не у профілі користувача, а в окремому каталозі), коли з першої спроби не завантажились усі необхідні для даного апаратного модуля файли (рис. 2.17). Тоді треба погодитися на завантаження решти інформації (рис. 2.18). І лише згодом продовжиться створення проєкту (див. рис. 2.16).

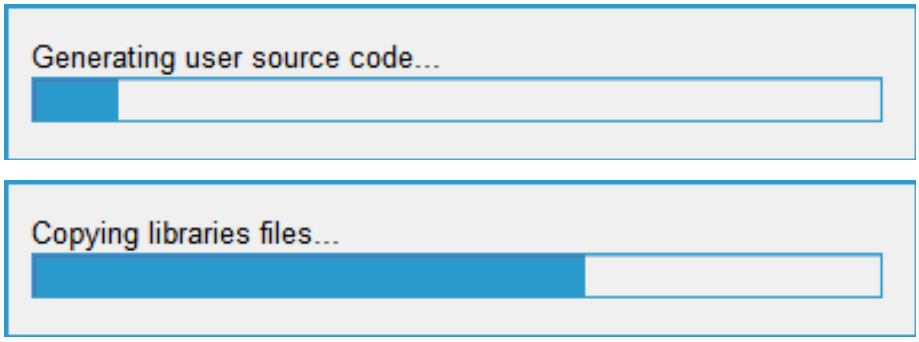



*Рис. 2.16. Процес створення проєкту*

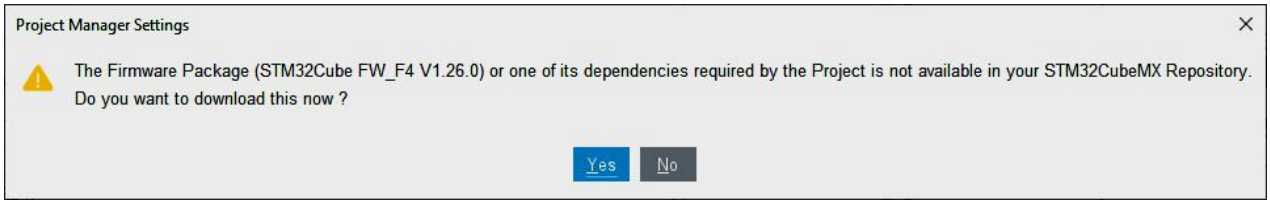



*Рис. 2.17. Запит на додаткове завантаження*

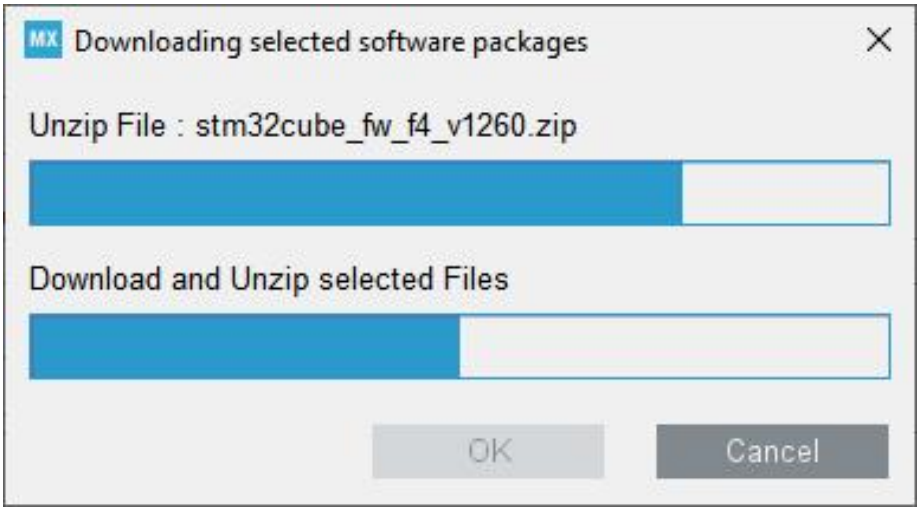



*Рис. 2.18. Завантаження додаткових файлів*

2.15. Погодитися на відкриття проєкту, натиснувши у запиті кнопку “Open Project” (рис. 2.19), щоб запустити MDK-ARM (рис. 2.20).

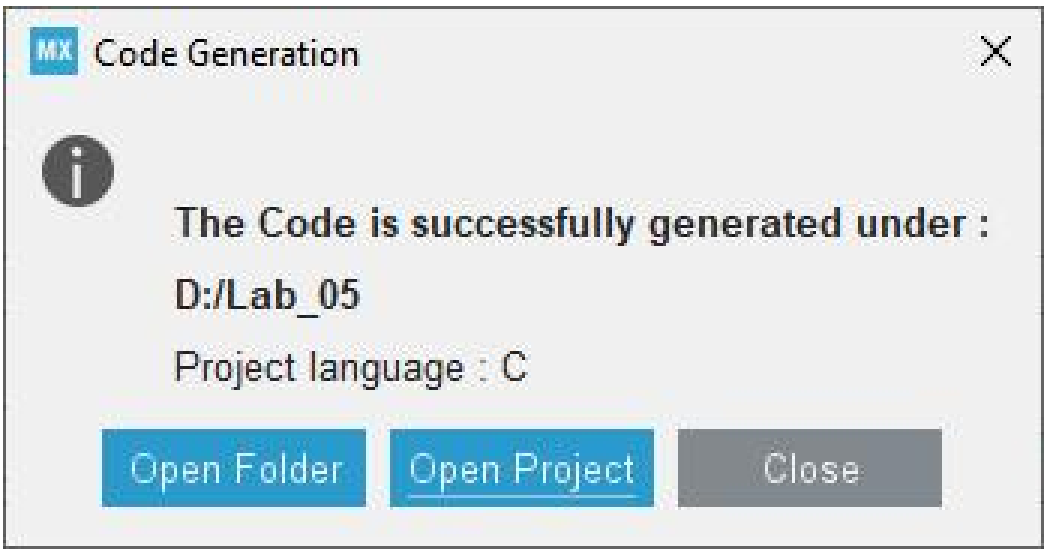



*Рис. 2.19. Запит на відкриття проєкту в MDK-ARM*

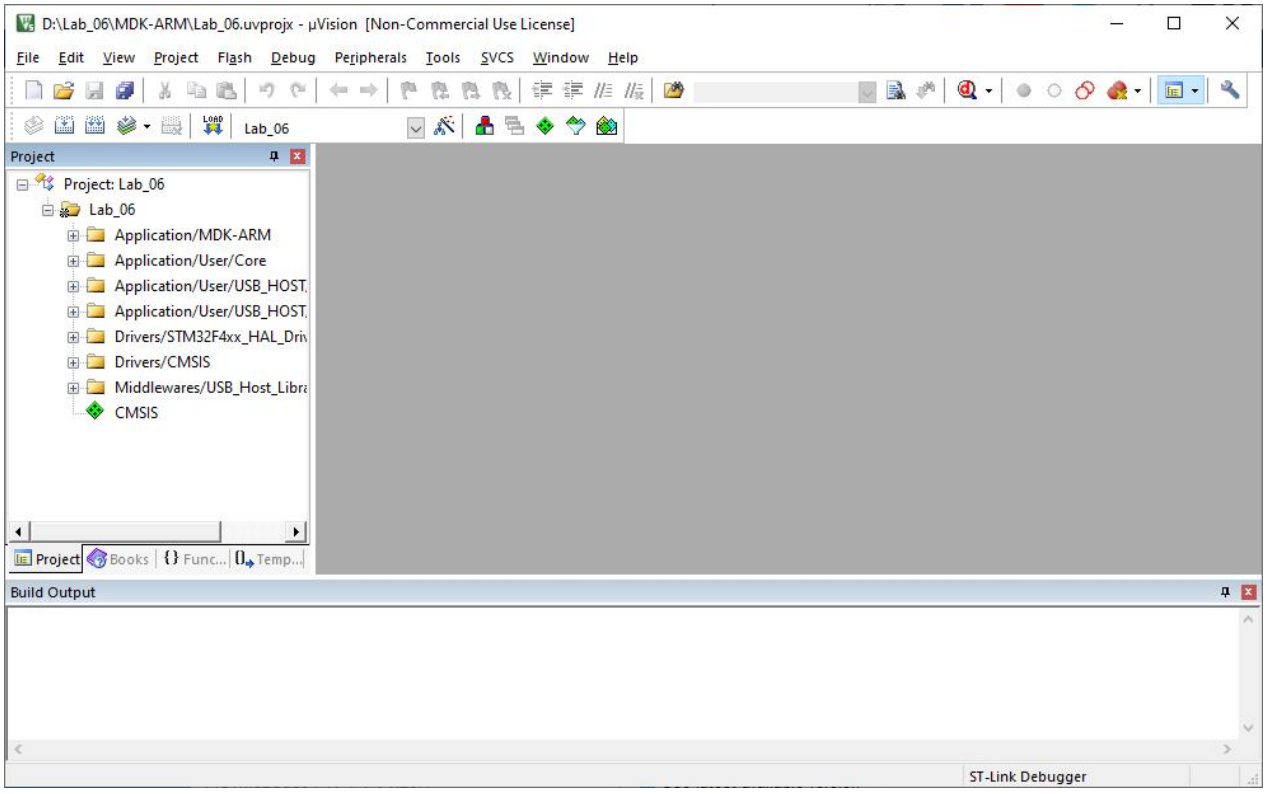



*Рис. 2.20. Вікно MDK-ARM (IDE Keil µVision)*

2.16. В одній із вкладених папок проєкту знайти і відкрити файл **main.c** з основними налаштуваннями проєкту (рис. 2.21).

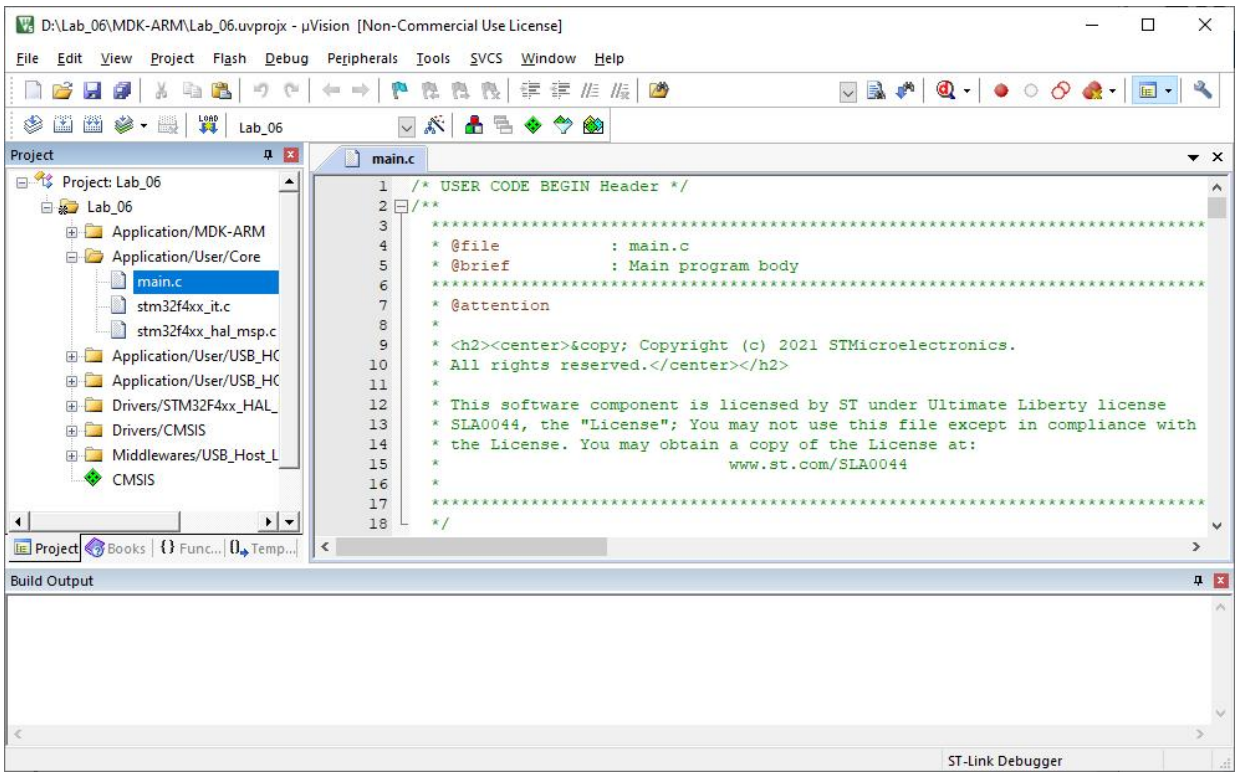



*Рис. 2.21. Головний файл проєкту*

2.17. Натиснути на функціональну клавішу F7 (Project > Build Target). Після компілювання переконатись у відсутності помилок в об’єктному коді заготовки проєкту (рис. 2.22).

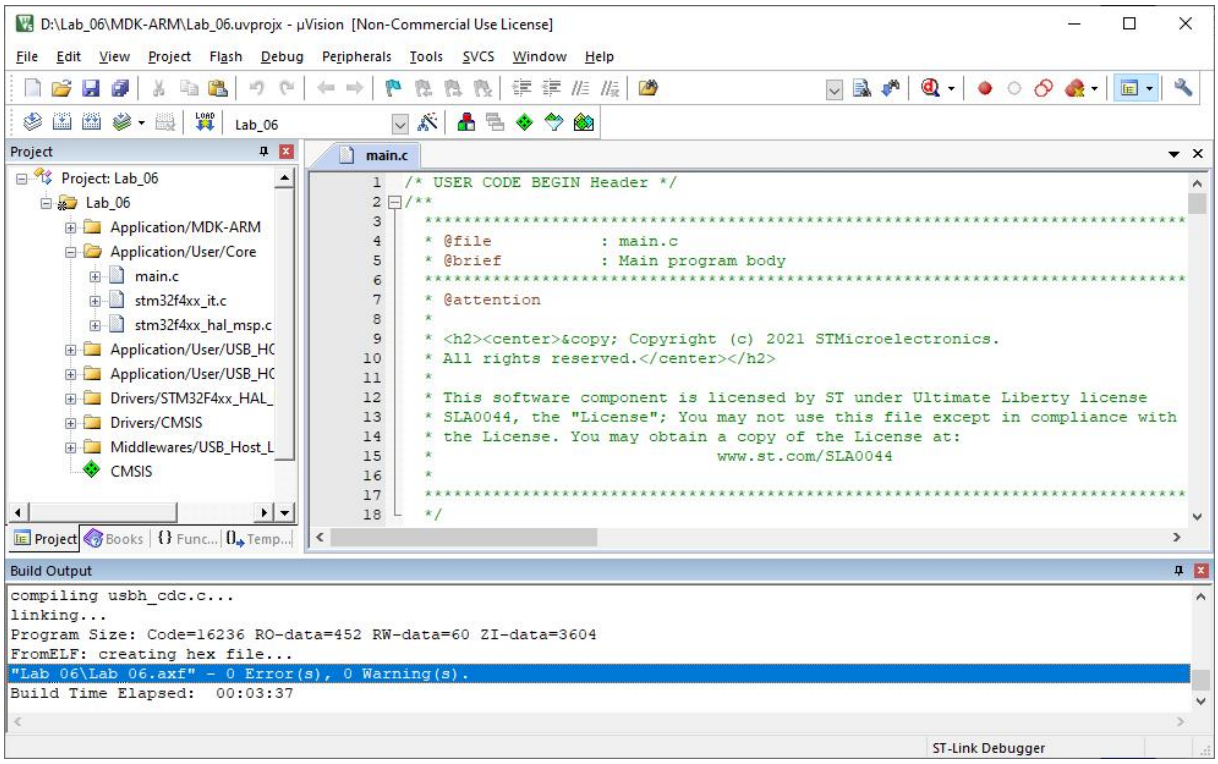



*Рис. 2.22. Успішна компіляція*

2.18. У тіло головної програми у розділ приватних визначень (рис. 2.23) вставити рядки:

```
/* Private define ------------------------------------------------
-------------*/
/* USER CODE BEGIN PD */
#define GreenLedPin GPIO_PIN_12
#define OrangeLedPin GPIO_PIN_13
#define RedLedPin GPIO_PIN_14
#define BlueLedPin GPIO_PIN_15

#define DataBufferSize 256
/* USER CODE END PD */
```

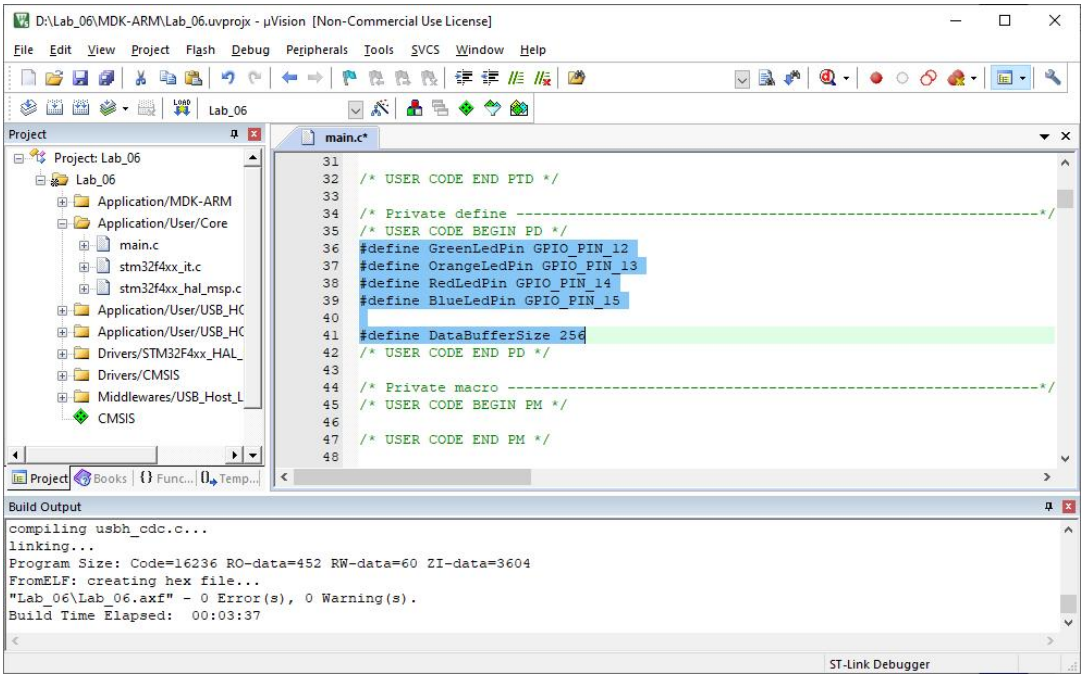



*Рис. 2.23. Додаткові визначення розробника*

2.19. Звернути увагу на останній рядок додаткових визначень. У ньому здійснюється погодження розміру виділеної пам’яті для обміну даними між комп’ютером і мікроконтролером (див. пункт 1.3). Числові значення “DataBufferSize” в Keil µVision та “BUFFERLENGTH” у Visual Studio мають збігатися. У наведених прикладах задано довжину блоку даних – 256 байтів.

2.20. В тіло головної програми у розділ циклу типу ”WHILE” вставити разом з рядками коментарів (!!!) робочий код:

```
/* USER CODE BEGIN WHILE */
  uint8_t u8_DataBuffer[DataBufferSize];

if(HAL_UART_Receive_IT(&huart2,&u8_DataBuffer[0],DataBufferSize)
!= HAL_OK)
```

```
  {
    while (HAL_UART_GetState(&huart2) != HAL_UART_STATE_READY) {}
  }
  while (1)
  {
    if(huart2.RxXferCount==0) {
      HAL_GPIO_TogglePin(GPIOD, OrangeLedPin);

if(HAL_UART_Receive_IT(&huart2,&u8_DataBuffer[0],DataBufferSize)
!= HAL_OK){
        while (HAL_UART_GetState(&huart2) !=
HAL_UART_STATE_READY) {}
      }
      if(HAL_UART_Transmit_IT(&huart2, &u8_DataBuffer[0],
DataBufferSize) != HAL_OK){
        while (HAL_UART_GetState(&huart2) !=
HAL_UART_STATE_READY) {}
      }
    }
    /* USER CODE END WHILE */
```

Він реалізує приймання даних від комп’ютера та повернення їх назад (рис. 2.24).

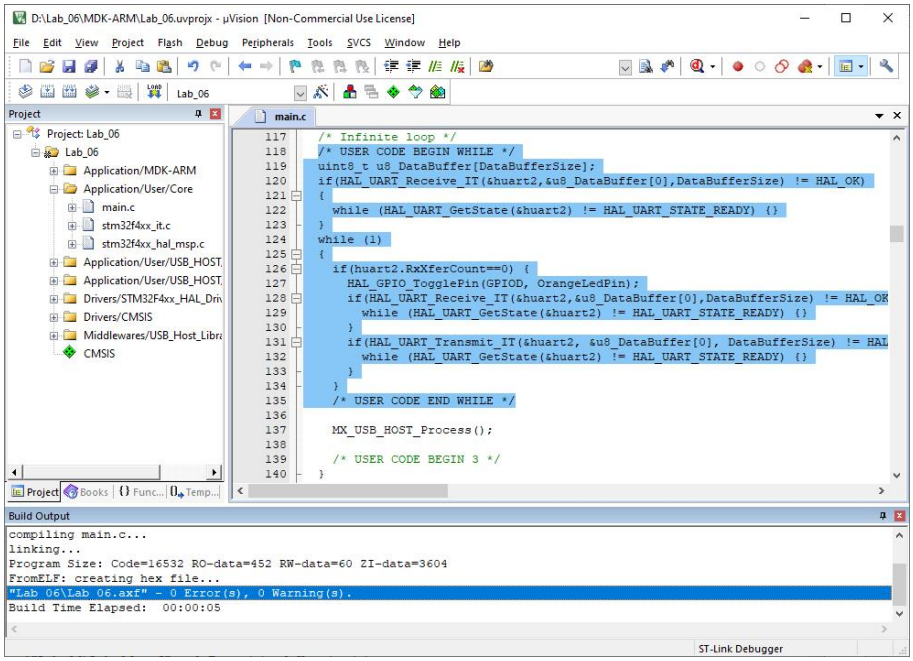



*Рис. 2.24. Цикл обміну даними*

2.21. Скомпілювати проєкт (F7, Project > Build Target) і переконатись у відсутності помилок в об’єктному коді (див. рис. 2.24).

2.22. Закрити IDE Keil µVision та STM32CubeMX.

### 3. Програмування плати та перевірка пам’яті і послідовного інтерфейсу

Оскільки на платі жоден з інтерфейсів UART мікроконтролера не з’єднаний з UART відладчика-програматора ST-LINK/V2, зв’язок послідовним каналом з комп’ютером доведеться забезпечувати через перехідник-перетворювач USB-UART CP2102.

3.1. Вивід GND модуля CP2102 (рис. 3.1) приєднати до виводу GND порту P1 налагоджувальної плати (рис. 3.2 і рис. 3.3).

3.2. Вивід RXD модуля CP2102 (див. рис. 3.1) приєднати до виводу PA2 (USART2_TX) налагоджувальної плати (див. рис. 3.2 і рис. 3.3).

3.3. Вивід TXD модуля CP2102 (див. рис. 3.1) приєднати до виводу PA3 (USART2_RX) налагоджувальної плати (див. рис. 3.2 і рис. 3.3).

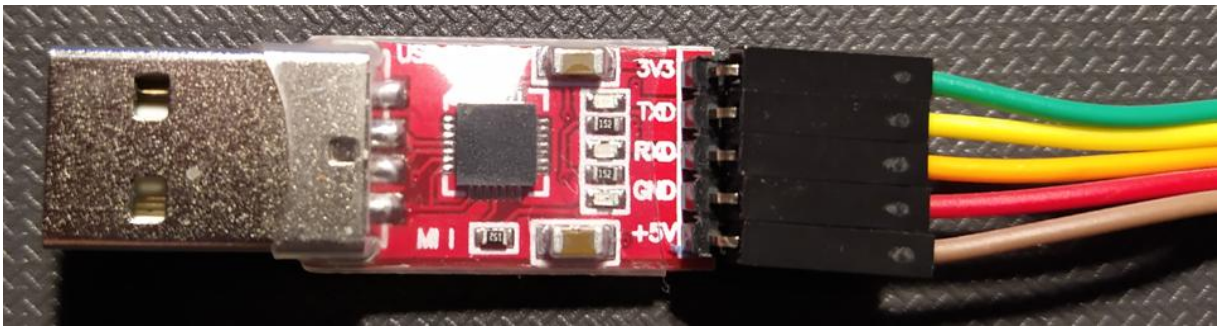



*Рис. 3.1. Приєднання провідників до модуля CP2102*

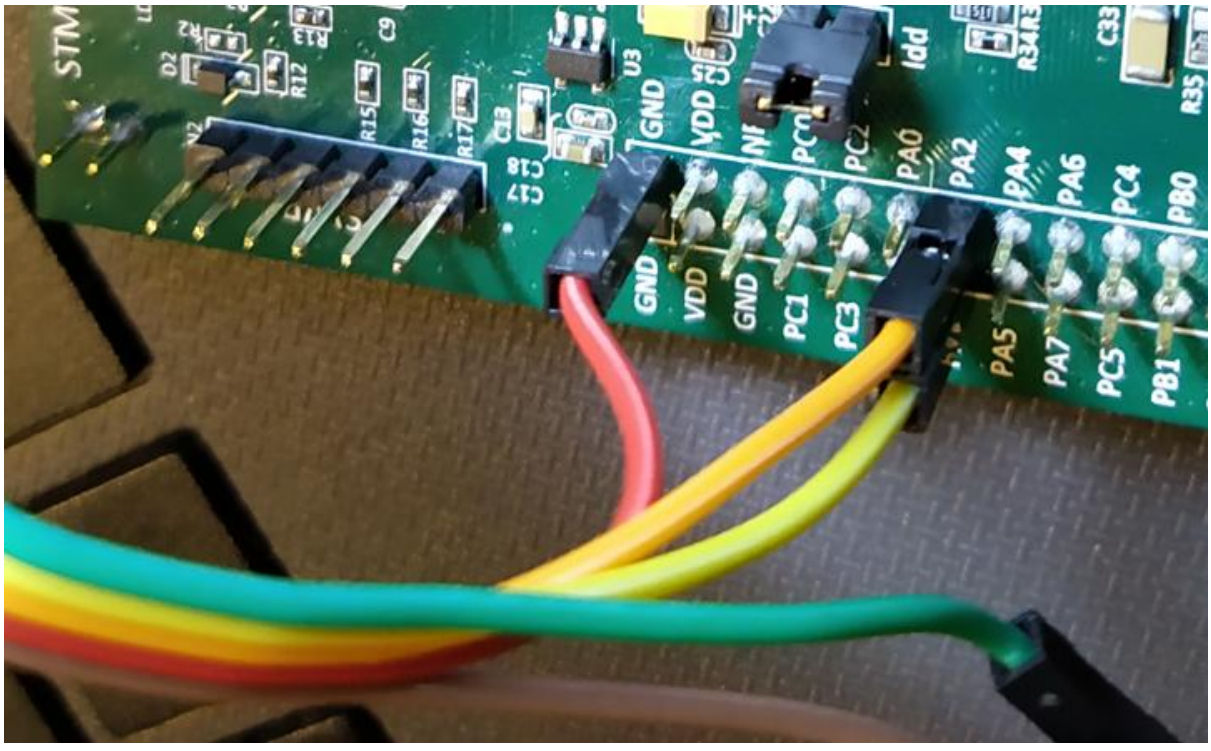



*Рис. 3.2. Приєднання провідників до налагоджувальної плати*

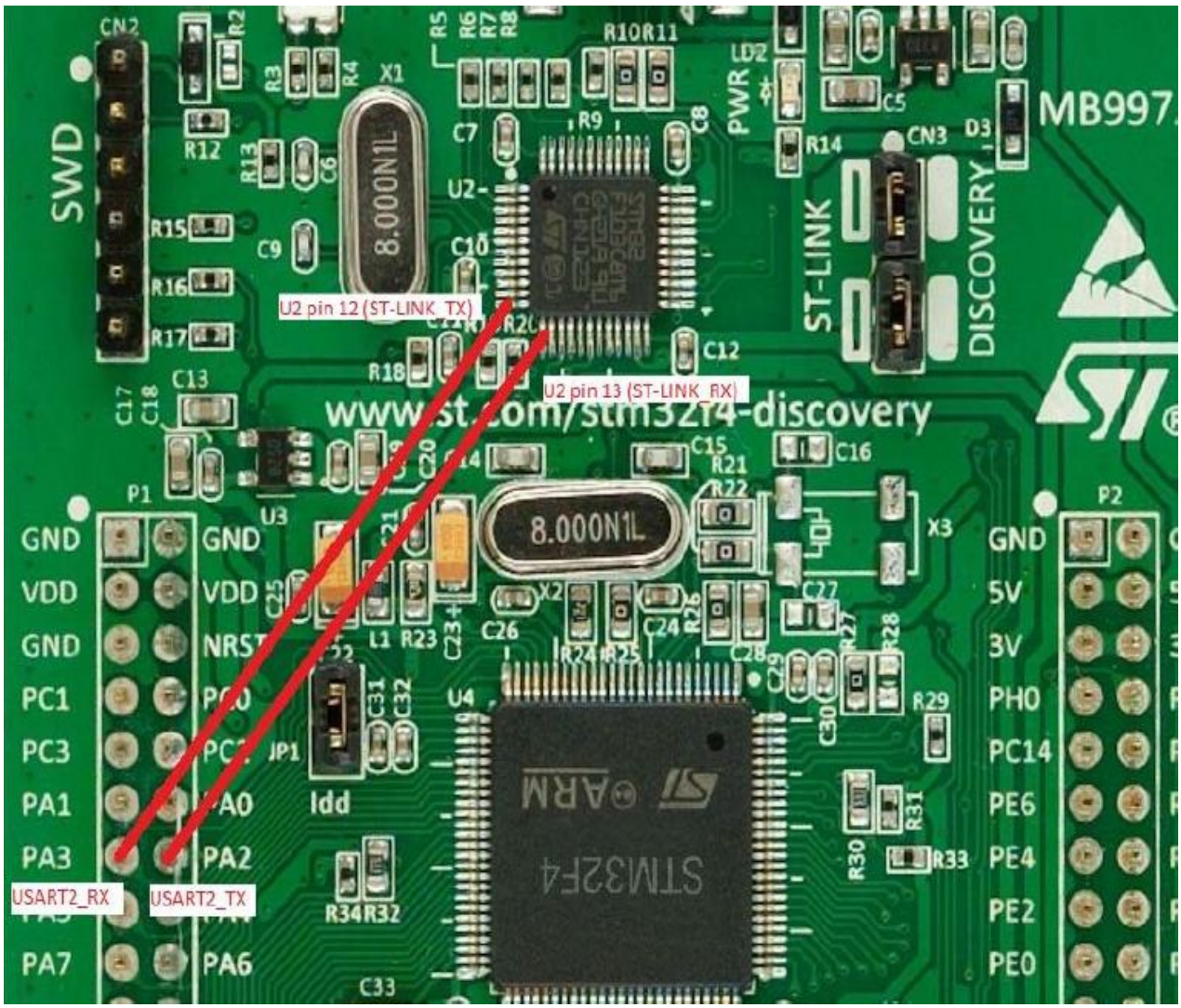



*Рис. 3.3. Фрагмент зображення налагоджувальної плати*

3.4. Кабельний з’єднувач mini-USB під’єднати до відповідного порту плати, а з’єднувач USB звичайного розміру – до вільного порту комп’ютера.

3.5. З’єднувач USB модуля CP2102 під’єднати до вільного порту комп’ютера.

3.6. Запустити Microsoft Visual Studio і відкрити підготовлену консольну програму.

3.7. Запустити сервіс Windows “Керування комп’ютером” (для Windows 8.1/10 – правою клавішею миші клацнути на кнопці “Пуск” і лівою клавішею миші вибрати вказаний сервіс з меню; для решти версій Windows – правою клавішею миші клацнути на ярлику “Мій комп’ютер” і лівою клавішею миші вибрати сервіс з меню). У вікні “Керування комп’ютером” відкрити “Диспетчер пристроїв” (рис. 3.4). Розгорнувши категорію “Порти (COM та LPT)”, з’ясувати, який номер порту відповідає перетворювачу USB-UART Silicon Labs CP2102 (COM?). У 3-му і 4-му рядках блоку глобальних змінних консольної програми Visual Studio (див. пункт 1.4) виправити номер COM-порту, щоб він збігався з номером у диспетчері пристроїв.

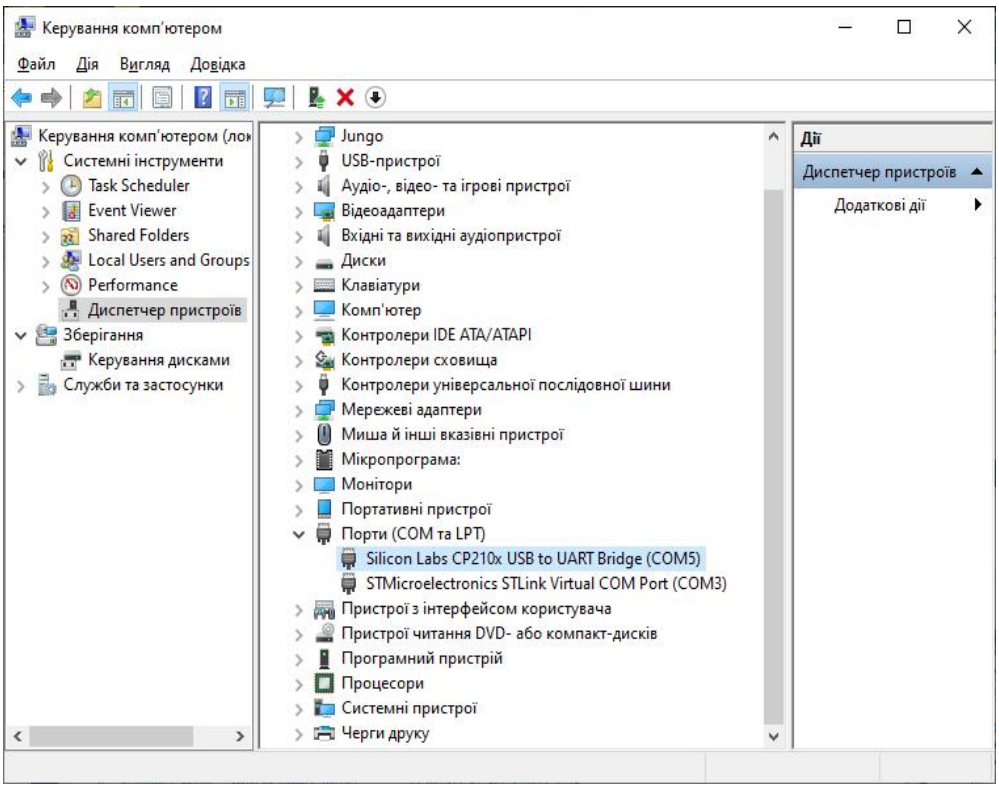



*Рис. 3.4. Сервіс Windows "Керування комп'ютером"*

3.8. Увага! Цей крок можна виконати лише з правами адміністратора. У вікні "Керування комп'ютером" (див. рис. 3.4) правою клавішею миші клацнути на рядку " Silicon Labs CP210x USB to UART Bridge (COM?)". У меню вибрати "Властивості" і у вікні з такою назвою перейти на вкладку "Параметри порту" (рис. 3.5). Встановити необхідне значення швидкості обміну даними через COM-порт. У наведеному прикладі – 115200 біт/с.

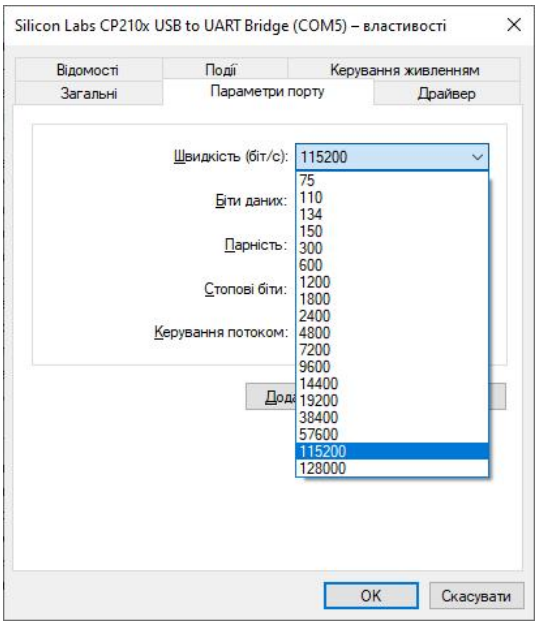



*Рис. 3.5. Вікно властивостей COM-порту*

Таке саме значення швидкості має міститися у рядку “dcbSerialParams.BaudRate = CBR_115200;” консольної програми (див. пункт 1.5 та рядок № 74 на рис. 3.6), а також у рядку “huart2.Init.BaudRate = 115200;” головної програми Keil µVision (рядок № 317 на рис. 3.7).

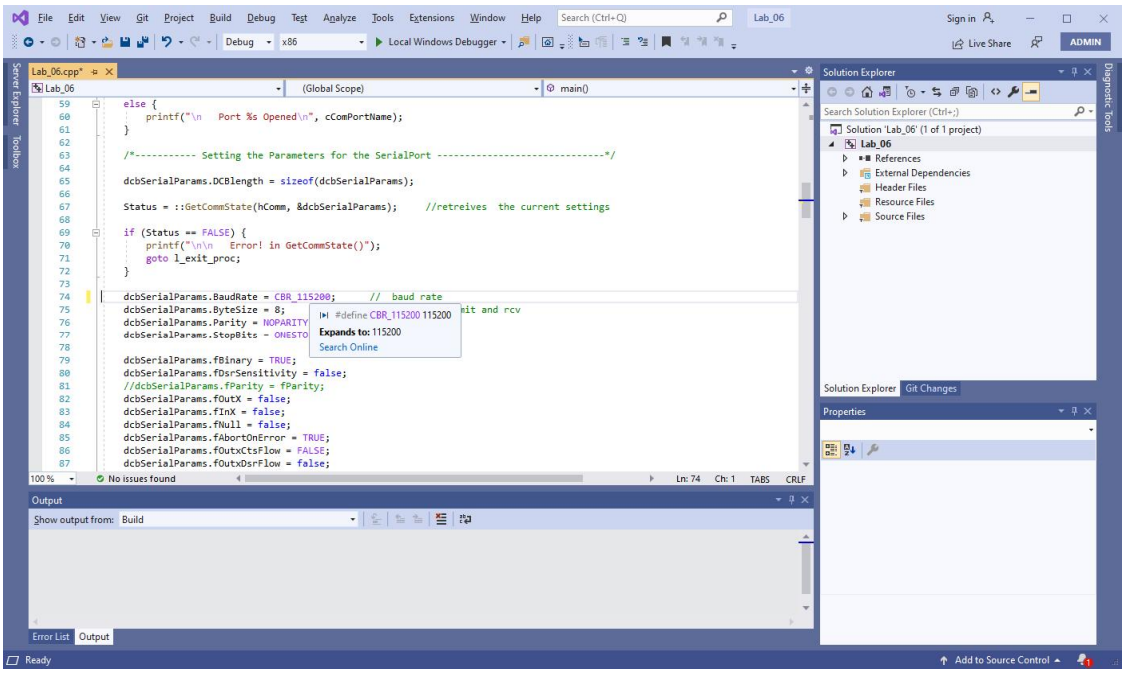



*Рис. 3.6. Зміна швидкості у Visual Studio*

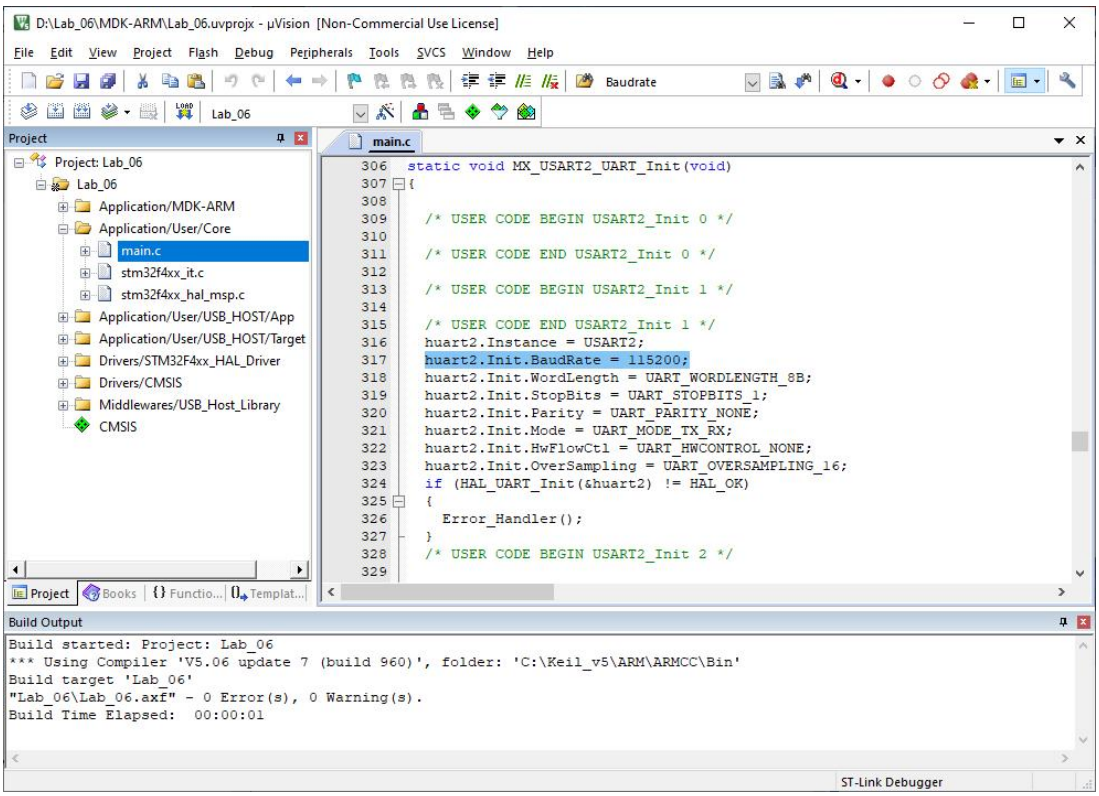



*Рис. 3.7. Зміна швидкості в Keil µVision*

3.8. В Keil µVision відкрити створений проєкт, налаштувати задані значення швидкості та довжини блоку даних. Скомпілювати код програми і завантажити на плату (рис. 3.8).

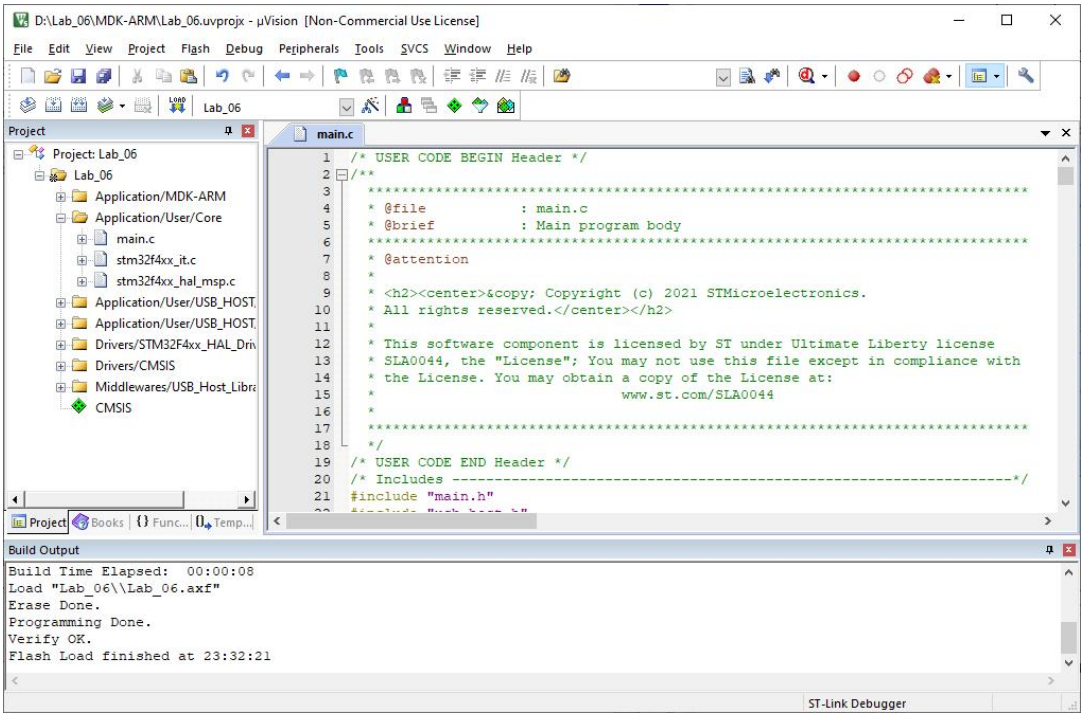



*Рис. 3.8. Успішне програмування плати*

3.9. На платі натиснути кнопку чорного кольору “Reset” – мікроконтролер готовий до роботи.

3.10. У Visual Studio налаштувати задані значення швидкості та довжини блоку даних, скомпілювати і запустити консольну програму обміну даними з мікроконтролером (рис. 3.9 та рис. 3.10).

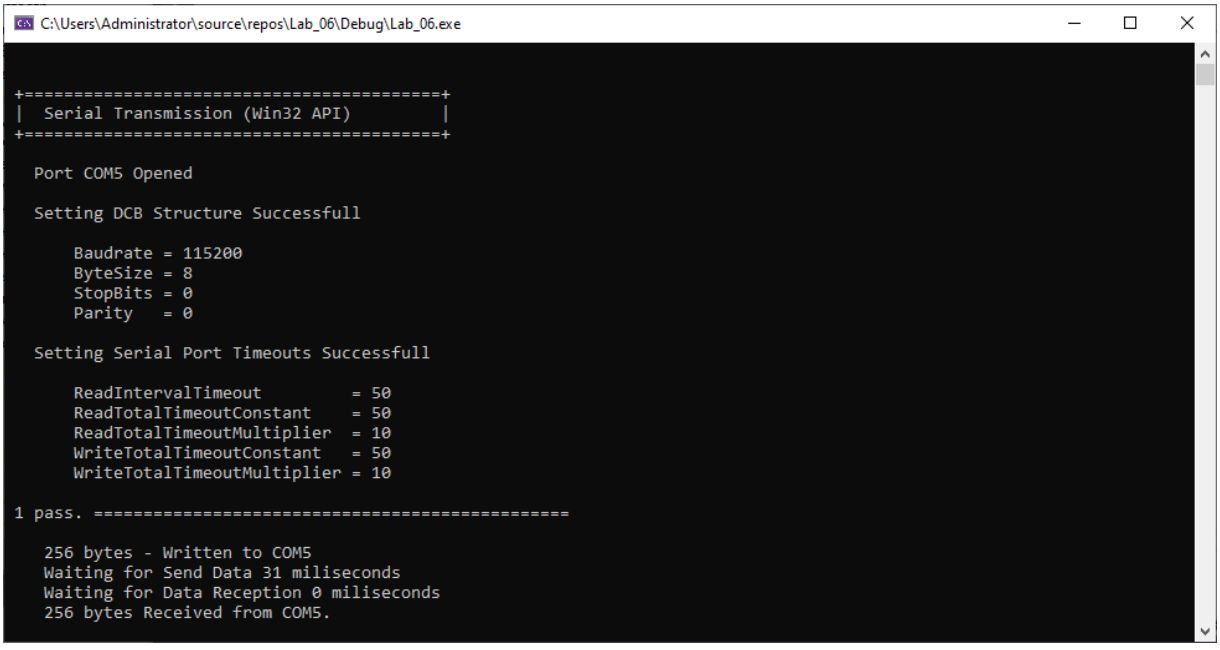



*Рис. 3.9. Результат виконання консольної програми (початок)*

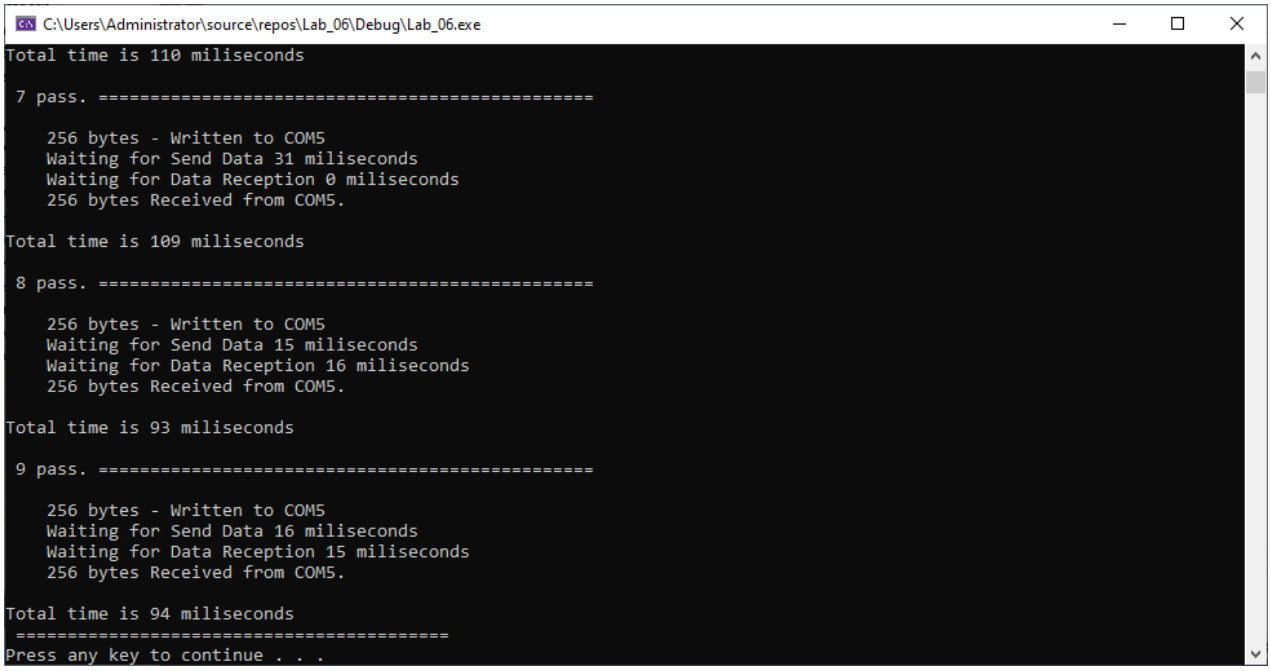



*Рис. 3.10. Результат виконання консольної програми (кінець)*

3.11. Існуючу індикацію успішного завершення обміну даними, яка здійснюється засвіченням помаранчевого світлодіода, замінити на почергове блимання двох світлодіодів згідно з варіантом індивідуального завдання до лабораторної роботи № 5. Для цього необхідно модифікувати програму в Keil µVision.

## ЗАВДАННЯ

1. У Microsoft Visual Studio створити консольну програму для обміну даними.
2. У середовищах розробки STM32CubeMX та Keil MDK-ARM підготувати проєкт для налагоджувальної плати.
3. Модифікувати програми відповідно до індивідуального завдання.
4. Виконати необхідні електричні з’єднання апаратних компонентів.
5. Запрограмувати мікроконтролер.
6. Перевірити можливість обміну даними між комп’ютером та оперативною пам’яттю налагоджувальної плати за допомогою універсального асинхронного інтерфейсу.

| Варіант завдання | Розмір блоку даних, байт | Швидкість обміну даними, біт/с |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 256 | 128000 |
| 2 | 288 | 115200 |
| 3 | 320 | 57600 |
| 4 | 352 | 38400 |
| 5 | 384 | 19200 |
| 6 | 416 | 14400 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 7 | 448 | 9600 |
| 8 | 480 | 4800 |
| 9 | 512 | 2400 |
| 10 | 544 | 1200 |
| 11 | 576 | 128000 |
| 12 | 608 | 115200 |
| 13 | 640 | 57600 |
| 14 | 672 | 38400 |
| 15 | 704 | 19200 |
| 16 | 736 | 14400 |
| 17 | 768 | 9600 |
| 18 | 800 | 4800 |
| 19 | 832 | 2400 |
| 20 | 864 | 1200 |
| 21 | 896 | 128000 |
| 22 | 928 | 115200 |
| 23 | 960 | 57600 |
| 24 | 992 | 38400 |
| 25 | 1024 | 19200 |
| 26 | 1088 | 14400 |
| 27 | 1120 | 9600 |
| 28 | 1152 | 4800 |
| 29 | 1184 | 2400 |
| 30 | 1216 | 1200 |

## ЗМІСТ ЗВІТУ

1. Номер, назва і мета роботи.
2. Теоретична частина у тезовому викладі (стисло, без рисунків).
3. Індивідуальне завдання.
4. Опис послідовності виконання роботи. Результати проілюструвати скриншотами.
5. Висновки.
6. Додаток. Повні тексти програм.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Який складник мікроконтролера використовується для вирівнювання продуктивності ядра і швидкодії флеш-пам’яті?

2. Для чого передбачений блок захисту пам’яті – Memory Protection Unit?

3. До якого розміру ОП є прямий доступ з боку процесора?

4. Який розмір резервної області ОП?

5. Скільки контролерів прямого доступу до пам'яті містить мікроконтролер?

6. Скільки і якого типу універсальні приймачі-передавачі містяться в мікроконтролері?

7. Із якою максимальною швидкістю здатні спілкуватися універсальні приймачі-передавачі мікроконтролера?

## ДЖЕРЕЛА ІНФОРМАЦІЇ

1. STM32F405xx. STM32F407xx. Datasheet – production data. URL: https://www.st.com/resource/en/datasheet/stm32f407vg.pdf.

2. UM1472. User manual. Discovery kit with STM32F407VG MCU. URL: https://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf.

3. RM0090. Reference manual. STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs. URL: https://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf.

4. UM1718. User manual. STM32CubeMX for STM32 configuration and initialization C code generation. URL: https://www.st.com/content/ccc/resource/technical/document/user_manual/10/c5/1a/43/3a/70/43/7d/DM00104712.pdf/files/DM00104712.pdf/jcr:content/translations/en.DM00104712.pdf.

5. Getting started with MDK. Create applications with µVision® for ARM® Cortex®-M microcontrollers. URL: https://armkeil.blob.core.windows.net/product/gs_MDK5_4_en.pdf.

## ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

| № | Завдання | Правильна відповідь |
|---|---|---|
| 1 | 2 | 3 |
| 1 | Стенд для тестування комбінаційних схем має 11 входів та 8 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні) – 30 | 2 |
| 2 | Стенд для тестування комбінаційних схем має 12 входів та 7 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні) – 119 | 0 |
| 3 | Стенд для тестування комбінаційних схем має 10 входів та 7 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 21 | 3 |
| 4 | Стенд для тестування комбінаційних схем має 11 входів та 7 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 201 | 4 |
| 5 | Стенд для тестування комбінаційних схем має 10 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 124 | 1 |
| 6 | Стенд для тестування комбінаційних схем має 9 входів та 7 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 175 | 5 |
| 7 | Стенд для тестування комбінаційних схем має 6 входів та 6 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 104 | 5 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 8 | Стенд для тестування комбінаційних схем має 9 входів та 8 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 81 | 5 |
| 9 | Стенд для тестування комбінаційних схем має 10 входів та 6 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 154 | 3 |
| 10 | Стенд для тестування комбінаційних схем має 6 входів та 10 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 84 | 4 |
| 11 | Стенд для тестування комбінаційних схем має 6 входів та 11 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 63 | 3 |
| 12 | Стенд для тестування комбінаційних схем має 11 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 165 | 1 |
| 13 | Стенд для тестування комбінаційних схем має 9 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 91 | 6 |
| 14 | Стенд для тестування комбінаційних схем має 10 входів та 8 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 97 | 0 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 15 | Стенд для тестування комбінаційних схем має 12 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 93 | 0 |
| 16 | Стенд для тестування комбінаційних схем має 6 входів та 9 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 192 | 7 |
| 17 | Стенд для тестування комбінаційних схем має 9 входів та 10 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 16 | 0 |
| 18 | Стенд для тестування комбінаційних схем має 10 входів та 8 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 232 | 1 |
| 19 | Стенд для тестування комбінаційних схем має 11 входів та 11 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 75 | 0 |
| 20 | Стенд для тестування комбінаційних схем має 10 входів та 11 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 157 | 5 |
| 21 | Стенд для тестування комбінаційних схем має 10 входів та 10 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 72 | 0 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 22 | Стенд для тестування комбінаційних схем має 10 входів та 11 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 183 | 4 |
| 23 | Стенд для тестування комбінаційних схем має 11 входів та 6 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 70 | 0 |
| 24 | Стенд для тестування комбінаційних схем має 6 входів та 6 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 244 | A |
| 25 | Стенд для тестування комбінаційних схем має 10 входів та 10 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 33 | 3 |
| 26 | Стенд для тестування комбінаційних схем має 10 входів та 11 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 155 | 1 |
| 27 | Стенд для тестування комбінаційних схем має 9 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 195 | 6 |
| 28 | Стенд для тестування комбінаційних схем має 11 входів та 6 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 85 | 3 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 29 | Стенд для тестування комбінаційних схем має 9 входів та 10 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 183 | 7 |
| 30 | Стенд для тестування комбінаційних схем має 8 входів та 12 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 57 | 4 |
| 31 | Стенд для тестування комбінаційних схем має 6 входів та 9 виходів. На скільки необхідно збільшити кількість його входів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), –226 | A |
| 32 | Стенд для тестування комбінаційних схем має 12 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 56 | 0 |
| 33 | Стенд для тестування комбінаційних схем має 11 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 68 | 3 |
| 34 | Стенд для тестування комбінаційних схем має 7 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 34 | 0 |
| 35 | Стенд для тестування комбінаційних схем має 7 входів та 11 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 19 | 0 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 36 | Стенд для тестування комбінаційних схем має 11 входів та 8 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 247 | 7 |
| 37 | Стенд для тестування комбінаційних схем має 10 входів та 6 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 122 | 8 |
| 38 | Стенд для тестування комбінаційних схем має 10 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 123 | 4 |
| 39 | Стенд для тестування комбінаційних схем має 11 входів та 6 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 37 | 6 |
| 40 | Стенд для тестування комбінаційних схем має 6 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 55 | 2 |
| 41 | Стенд для тестування комбінаційних схем має 11 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 63 | 4 |
| 42 | Стенд для тестування комбінаційних схем має 10 входів та 10 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 22 | 3 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 43 | Стенд для тестування комбінаційних схем має 7 входів та 10 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 159 | 6 |
| 44 | Стенд для тестування комбінаційних схем має 7 входів та 11 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 207 | 0 |
| 45 | Стенд для тестування комбінаційних схем має 10 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 50 | 0 |
| 46 | Стенд для тестування комбінаційних схем має 7 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 191 | 6 |
| 47 | Стенд для тестування комбінаційних схем має 12 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 28 | 6 |
| 48 | Стенд для тестування комбінаційних схем має 7 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 170 | 6 |
| 49 | Стенд для тестування комбінаційних схем має 11 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 172 | 9 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 50 | Стенд для тестування комбінаційних схем має 8 входів та 10 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 25 | 3 |
| 51 | Стенд для тестування комбінаційних схем має 7 входів та 11 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 100 | 1 |
| 52 | Стенд для тестування комбінаційних схем має 6 входів та 11 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 64 | 3 |
| 53 | Стенд для тестування комбінаційних схем має 9 входів та 6 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 3, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 220 | 5 |
| 54 | Стенд для тестування комбінаційних схем має 7 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 8, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 87 | 8 |
| 55 | Стенд для тестування комбінаційних схем має 7 входів та 11 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 6, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 60 | 1 |
| 56 | Стенд для тестування комбінаційних схем має 9 входів та 7 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 8, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 144 | 6 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 57 | Стенд для тестування комбінаційних схем має 8 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 92 | 1 |
| 58 | Стенд для тестування комбінаційних схем має 6 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 5, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 194 | 1 |
| 59 | Стенд для тестування комбінаційних схем має 10 входів та 6 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 244 | 8 |
| 60 | Стенд для тестування комбінаційних схем має 9 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 3, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 20 | 3 |
| 61 | Стенд для тестування комбінаційних схем має 11 входів та 12 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 4, кількість виходів – 5, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 114 | 0 |
| 62 | Стенд для тестування комбінаційних схем має 10 входів та 8 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 7, кількість виходів – 7, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 114 | 6 |
| 63 | Стенд для тестування комбінаційних схем має 12 входів та 9 виходів. На скільки необхідно збільшити кількість його виходів, щоб на цьому стенді можна було тестувати комбінаційну схему цифрового автомата на основі D-тригерів, у якого кількість входів – 6, кількість виходів – 4, а кількість станів, в яких автомат може перебувати (при їх двійковому кодуванні), – 243 | 5 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 64 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам'яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 33, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 65 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 14 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 19, якщо розмір кеш-пам’яті 16 слів? | 3 |
| 66 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті у 12 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 18, якщо розмір кеш-пам’яті 16 слів? | 3 |
| 67 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам'яті в 10 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 19, якщо розмір кеш-пам’яті 16 слів? | 3 |
| 68 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 17, якщо розмір кеш-пам’яті 16 слів? | 5 |
| 69 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 14 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 32, якщо розмір кеш-пам’яті 16 слів? | 1 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 70 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 29, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 71 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті у 12 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 22, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 72 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 16 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 17, якщо розмір кеш-пам’яті 16 слів? | 6 |
| 73 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 8 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 19, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 74 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 15 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 20, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 75 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті у 12 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 24, якщо розмір кеш-пам’яті 16 слів? | 1 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 76 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 26, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 77 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 14 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 31, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 78 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 22, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 79 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 9 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 21, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 80 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 16 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 29, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 81 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 13 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 29, якщо розмір кеш-пам’яті 16 слів? | 1 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 82 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті у 12 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 17, якщо розмір кеш-пам'яті 16 слів? | 5 |
| 83 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті в 9 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 16, якщо розмір кеш-пам'яті 16 слів? | 9 |
| 84 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті в 14 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 23, якщо розмір кеш-пам'яті 16 слів? | 2 |
| 85 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті в 14 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 29, якщо розмір кеш-пам'яті 16 слів? | 1 |
| 86 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті в 13 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 32, якщо розмір кеш-пам'яті 16 слів? | 1 |
| 87 | Цикл звернення до кеш-пам'яті з прямим відображенням менший за цикл звернення до основної пам'яті в 11 разів. Під час тестування кеш-пам'яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам'яті, під час другого частина даних може читатися з кеш-пам'яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 21, якщо розмір кеш-пам'яті 16 слів? | 2 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 88 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 11 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 32, якщо розмір кеш-пам’яті 16 слів? | 1 |
| 89 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 14 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 21, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 90 | Цикл звернення до кеш-пам’яті з прямим відображенням менший за цикл звернення до основної пам’яті в 14 разів. Під час тестування кеш-пам’яті здійснюється два читання того самого блоку даних: перший раз читання відбувається з основної пам’яті, під час другого частина даних може читатися з кеш-пам’яті, якщо вони там є. У скільки разів зменшиться час повторного читання блоку даних з кількістю слів 20, якщо розмір кеш-пам’яті 16 слів? | 2 |
| 91 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 77, 62, 89 градусів? | 0 |
| 92 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 87, 46, 88 градусів? | 0 |
| 93 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 7 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 44, 56, 52 градусів? | 3 |
| 94 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 66, 10, 44 градусів? | 0 |
| 95 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 84, 13, 83 градусів? | 0 |
| 96 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 1 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 79, 44, 80 градусів? | 0 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 97 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 30, 46, 73 градусів? | 6 |
| 98 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 69, 16, 48 градусів? | 0 |
| 99 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 56, 42, 38 градусів? | 1 |
| 100 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 70, 45, 30 градусів? | 1 |
| 101 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 80, 53, 47 градусів? | 0 |
| 102 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 79, 39, 70 градусів? | 0 |
| 103 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 60, 17, 54 градусів? | 0 |
| 104 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 67, 46, 59 градусів? | 0 |
| 105 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 22, 85, 78 градусів? | 0 |
| 106 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 6 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 21, 21, 44 градусів? | 0 |
| 107 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 7 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 17, 83, 0 градусів? | 0 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 108 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 64, 6, 17 градусів? | 0 |
| 109 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 62, 87, 34 градусів? | 0 |
| 110 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 35, 57, 7 градусів? | 7 |
| 111 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 1 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 14, 4, 22 градусів? | 2 |
| 112 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 34, 65, 86 градусів? | 0 |
| 113 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 5 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 26, 21, 10 градусів? | 0 |
| 114 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 1 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 82, 36, 43 градусів? | 0 |
| 115 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 29, 66, 1 градусів? | 2 |
| 116 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 1, 14, 20 градусів? | 0 |
| 117 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 75, 80, 23 градусів? | 0 |
| 118 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 5 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 56, 7, 59 градусів? | 8 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 119 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 5 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 41, 36, 72 градусів? | 4 |
| 120 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 30, 62, 86 градусів? | 0 |
| 121 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 81, 70, 29 градусів? | 0 |
| 122 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 1 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 58, 84, 30 градусів? | 0 |
| 123 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 6 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 40, 79, 47 градусів? | 3 |
| 124 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 55, 41, 88 градусів? | 0 |
| 125 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 58, 48, 23 градусів? | 4 |
| 126 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 14, 74, 78 градусів? | 0 |
| 127 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 5 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 8, 5, 44 градусів? | 0 |
| 128 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 2 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 37, 51, 24 градусів? | 5 |
| 129 | Скільки разів квантовий комп’ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 54, 68, 46 градусів? | 2 |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 130 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 85, 75, 86 градусів? | 0 |
| 131 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 7 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 63, 73, 17 градусів? | 1 |
| 132 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 3 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 67, 86, 37 градусів? | 1 |
| 133 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 4 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 30, 78, 0 градусів? | 0 |
| 134 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 7 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 15, 63, 41 градусів? | 0 |
| 135 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 44, 89, 89 градусів? | 0 |
| 136 | Скільки разів квантовий комп'ютер, що складається з 3 кубіт, видасть результат 0 в серії з 16 однакових дослідів, якщо його кубіти (старший, середній та молодший) наприкінці кожного досліду зупиняються в положенні 87, 76, 3 градусів? | 0 |
| 137 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 7, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 68, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | E |
| 138 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду B4, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 0 |
| 139 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 7, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 61, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | E |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 140 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – С, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 6, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | B |
| 141 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 2, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду CE, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 5 |
| 142 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 6, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 12, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | C |
| 143 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 1, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 80, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 3 |
| 144 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – С, утворюючий поліном - (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 9A, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | C |
| 145 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 1, утворюючий поліном - (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду B8, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 3 |
| 146 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – E, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 41, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | F |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 147 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – D, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 1A, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 9 |
| 148 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном - (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 22, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 7 |
| 149 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 6, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 9B, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | D |
| 150 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 8, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду D1, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 4 |
| 151 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 4, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 6F, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 8 |
| 152 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 5, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду B4, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | B |
| 153 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 5, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 3B, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | A |

*Продовження табл.*

| 1 | 2 | 3 |
|---|---|---|
| 154 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра - 2, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 5D, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 4 |
| 155 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 1, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 2C, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 2 |
| 156 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – A, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду A2, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |
| 157 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 3, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 6E, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |
| 158 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду A6, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |
| 159 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду D0, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |
| 160 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном - (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 8F, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |

*Закінчення табл.*

| 1 | 2 | 3 |
|---|---|---|
| 161 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – C, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 99, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | A |
| 162 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 5, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду DB, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | B |
| 163 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 9, утворюючий поліном – (4, 2, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду E8, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 6 |
| 164 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 8, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду 21, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 3 |
| 165 | Для визначення циклічної контрольної суми використовується регістр зсуву з лінійними зворотніми зв'язками. Поточний стан регістра – 1, утворюючий поліном – (4, 1, 0) (числа відповідають показникам степеня членів полінома). На вхід регістра подаються в цей момент розряди 8-бітного коду D4, записаного 16-ковими символами. Яким буде стан регістра після першого зсуву? | 3 |

# ТЕРМІНОЛОГІЧНИЙ СЛОВНИЧОК

| Термін | Тлумачення |
|---|---|
| **1** | **2** |
| Алгоритм | Скінчена послідовність команд, направлених на виконання певного завдання. |
| Алфавіт | Упорядкований набір символів деякої природної мови. |
| Алфавіт мови програмування | Символьний набір, заданий описом конкретної мови програмування для подання вихідних програм та даних. |
| Байт | Група з 8 біт, використовується як міра інформації. |
| Біт | Двійкова одиниця кількості інформації, яка відповідає тій інформації, яка одержується при прийомі одного з двох рівноймовірних повідомлень. |
| Буфер | Поле пам'яті або пристрій, що використовується для узгодження між швидкостями обміну, розмірами блоків даних та моментами виникнення подій при обміні даними. |
| Величина | Об'єкт, із яким пов'язується певна множина значень. |
| Вираз | Формульний запис у програмі, що визначає операції над даними, виконуючи які, процесор одержує значення; правила запису виразів визначаються конкретною мовою програмування. |
| Дані | Інформація, подана у вигляді зручному для формального опрацювання автоматичними пристроями або людиною. |
| Діаграма | Графічне зображення співвідношення між величинами, об'єктами. |
| Довжина рядка | Кількість символів, які справді містяться в рядковій величині на час її опрацювання. |
| Елемент | Складова частина складного цілого. |
| ЕОМ | Електронно-обчислювальна машина (див. Комп'ютер) – цифрова обчислювальна машина, основні вузли якої реалізовані засобами електроніки. |
| Залежність | Відношення між функцією та її аргументами. |
| Запам'ятовувальний пристрій | Пристрій для фіксації даних таким чином, щоб вони потім могли бути автоматично відтворені. |
| Змінна | Тріада: об'єкт, що може змінювати значення під час виконання програми; ім'я змінної; значення змінної. |
| Значення | Смисл, який приписується об'єкту людиною. |
| Інформатика | Технічна наука, що систематизує прийоми створення збереження, накопичення, опрацювання та передачі даних засобами обчислювальної техніки, а також принципи функціонування цих засобів і методи керування ними. |
| Інформація | Продукт взаємодії даних та методів, який розглядається в контексті цієї взаємодії. |
| Кілобайт | Одиниця виміру інформації, яка містить 1024 байт. |
| Код | Набір знаків в сукупності зі схемою кодування для подання інформації у вигляді даних. |
| Кодування даних | Процес перетворення даних до єдиної форми подання. |
| Комп'ютер | Універсальний електронний пристрій, призначений для автоматизації накопичення, збереження, опрацювання, передачі та відтворення даних. |

*Продовження табл.*

| 1 | 2 |
|---|---|
| Константа | Величина, що не змінює своє значення за час виконання програми. |
| Контролер | Пристрій, призначений для керування зовнішніми пристроями. |
| Лінійний алгоритм | Алгоритм, в якому оператори виконуються в порядку їх слідування. |
| Масив | Структура даних, що являє собою впорядкований набір однотипних даних. |
| Мегабайт | Одиниця виміру інформації, містить 1024 кілобайти або приблизно 1000000 байт. |
| Множина | Сукупність деяких об'єктів, які мають характерну властивість для всіх об'єктів. |
| Модем (модулятор – демодулятор) | Пристрій для перетворення даних для передачі їх за певними лініями зв'язку. |
| Обчислювальна техніка | Сукупність пристроїв, призначених для автоматизації опрацювання даних. |
| Обчислювальна система | Конкретний набір програмно-апаратних пристроїв, призначених для виконання певного класу завдань. |
| Операнд | Аргумент операції. |
| Оперативна пам'ять | Набір мікросхем, призначених для зберігання даних під час їх безпосереднього опрацювання; |
| Оператор | Граматична конструкція, визначення якої уточнюється конкретною мовою програмування і яка виражає певну закінчену послідовність машинних операцій; інтерпретується як вказівка виконати ці команди. |
| Оператор вводу | Оператор, призначений для вводу даних з клавіатури або файлу. |
| Оператор виводу | Оператор, призначений для виводу даних на екран дисплею, принтер або у файл. |
| Оператор присвоювання | Оператор, що обчислює значення виразу та присвоює його змінній. |
| Оператор умовного переходу | Оператор, що змінює природну послідовність виконання операторів залежно від наведеної в операторі умови. |
| Оператор безумовного переходу | Оператор, що дає значенням мітку наступного опертора, що має виконуватись. |
| Оператор вибору | Оператор із переліком альтернатив, в яких задано дії відповідно до наведених умов. |
| Оператор переходу | Оператор, що змінює природну послідовність вибірки операторів програми при її виконанні. |
| Оператор присвоювання | Оператор, у результаті виконання якого змінній надається значення відповідно до заданого виразу. |
| Оператор циклу | Оператор, що визначає тіло циклу, параметри циклу та/або умови повтору виконання тіла циклу. |
| Операція | Деяка закінчена послідовність дій людини або машини. |
| Опис | Декларативне речення мови програмування, що оголошує об'єкти, їх імена, типи та області існування. |
| Організація | Внутрішня впорядкованість, узгодженість та взаємодія частин цілого, обумовлена його будовою. |

*Закінчення табл.*

| 1 | 2 |
|---|---|
| Пам’ять | Загальна назва для будь-яких засобів фіксації та збереження інформації, довільного реального або абстрактного механізму, що забезпечує можливість фіксувати сигнали. |
| Параметр | Змінна величина, від якої залежать значення інших величин. |
| Постійна пам’ять | Мікросхема, яка призначена для постійного зберігання інформації, у тому числі і при вимкненому живленні. |
| Пристрій | Сукупність деталей, вузлів, елементів, якій властива конструктивна та функціональна єдність. |
| Процесор (центральний процесор) | Пристрій опрацювання даних в комп’ютері; виконує функції арифметичної та логічної обробки даних та загального керування роботою інших пристроїв комп’ютера . |
| Розгалужений алгоритм | Алгоритм, у якому здійснюється вибір між декількома групами операторів. |
| Символ | Літерний ланцюжок з одного або декількох літер для позначення чого-небудь. |
| Система | 1. Сукупність абстрактних або матеріальних об’єктів разом з відомими або заданими властивостями і відношеннями, які утворюють в певному сенсі єдине ціле. 2. Сукупність матеріальних або абстрактних органів, пов’язаних загальною функцією, метою, призначенням. |
| Система числення | Спосіб вираження та позначення чисел. |
| Структура | 1. Сукупність стійких зв’язків між частинами об’єкта, що забезпечують його цілісність та тотожність самому собі, тобто, збереження його характеристичних властивостей при внутрішніх та зовнішніх змінах. 2. Екземпляр моделі даних. |
| Циклічний алгоритм | Алгоритм, що являє собою послідовність операторів, яка повинна виконатись декілька раз. |
| Швидкодія | Параметр, що характеризує продуктивність обчислювальної системи кількістю операцій, які виконуються за певний проміжок часу. |
| Шина | Набір провідників для передачі даних та сигналів керування. |

ЕЛЕКТРОННЕ НАВЧАЛЬНЕ ВИДАННЯ

**Глухов Валерій Сергійович**
**Хомуляк Мирослав Олегович**
**Бойко Геннадій Володимирович**
**Жолубак Іван Михайлович**

# ТЕСТУВАННЯ І ДІАГНОСТИКА ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ

Навчальний посібник

Редактор *Ірина Черевко*
Комп'ютерне верстання *Марти Гарасимів*

Режим доступу:
http://eom.lp.edu.ua/textbooks/np-tdpaz.pdf

Видавець і виготівник: Видавництво Львівської політехніки
*Свідоцтво суб'єкта видавничої справи ДК № 4459 від 27.12.2012 р.*

*вул. Ф. Колесси, 4, Львів, 79013*
тел. + 380 32 2584103, факс +380 32 2584101
vlp.com.ua, ел. пошта: vmr@vlp.com.ua