Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 6

з дисципліни: «Тестування та діагностика програмно-апаратних засобів»

на тему: «Тестування оперативної пам’яті та послідовного інтерфейсу»

Виконав:

ст. гр. КІ-303

Порубайміх О.Є.

Перевірив:

Ст.викл. Хомуляк М.О.

Львів – 2024

Зміст

[Зміст 2](#__RefHeading___Toc55759_1171611487)

[Перелік рисунків 2](#__RefHeading___Toc55761_1171611487)

[РОЗДІЛ 1. Мета 3](#__RefHeading___Toc14349_1995045277)

[РОЗДІЛ 2. Теоретичні відомості 4](#__RefHeading___Toc14355_1995045277)

[РОЗДІЛ 3. Індивідуальне завдання 6](#__RefHeading___Toc14353_1995045277)

[РОЗДІЛ 4. Виконання завдання 7](#__RefHeading___Toc14351_1995045277)

[4.1. Створення консольної програми. 7](#__RefHeading___Toc14357_1995045277)

[4.2. Підготовка проєкту для налагоджувальної плати. 9](#__RefHeading___Toc14369_1995045277)

[Висновок 15](#__RefHeading___Toc14347_1995045277)

[Список використаних джерел 16](#__RefHeading___Toc14345_1995045277)

[Додаток А. Код програми у Visual Studio 17](#__RefHeading___Toc8664_2297683834)

[Додаток Б. Код програми у Keil μVision. 21](#__RefHeading___Toc4403_358829861)

Перелік рисунків

[Рис. 4.1. Стартове вікно Visual Studio. 8](#%25D0%25A0%25D0%25B8%25D1%2581._4.!0|seq)

[Рис. 4.2. Попередні налаштування проєкту. 8](#%25D0%25A0%25D0%25B8%25D1%2581._4.!1|seq)

[Рис. 4.3. програмний код. 9](#%25D0%25A0%25D0%25B8%25D1%2581._4.!2|seq)

[Рис. 4.4. Налаштування швидкості передачі даних у Visual Studio. 9](#%25D0%25A0%25D0%25B8%25D1%2581._4.!3|seq)

[Рис. 4.5. Тестове виконання консольної програми. 10](#%25D0%25A0%25D0%25B8%25D1%2581._4.!4|seq)

[Рис. 4.6. Короткий опис плати STM32F407G-DISC1. 10](#%25D0%25A0%25D0%25B8%25D1%2581._4.!5|seq)

[Рис. 4.7. Відключення кварцу 32 кГц. 11](#%25D0%25A0%25D0%25B8%25D1%2581._4.!6|seq)

[Рис. 4.8. Вибір роботи в асинхронному режимі. 11](#%25D0%25A0%25D0%25B8%25D1%2581._4.!7|seq)

[Рис. 4.9. Встановлення режиму з використанням апаратних переривань. 12](#%25D0%25A0%25D0%25B8%25D1%2581._4.!8|seq)

[Рис. 4.10. Основні параметри проєкту. 12](#%25D0%25A0%25D0%25B8%25D1%2581._4.!9|seq)

[Рис. 4.11. Вікно MDK-ARM (IDE Keil μVision). 13](#%25D0%25A0%25D0%25B8%25D1%2581._4.!10|se)

[Рис. 4.12. Успішна компіляція. 13](#%25D0%25A0%25D0%25B8%25D1%2581._4.!11|se)

[Рис. 4.13. Додаткові визначення розробника. 14](#%25D0%25A0%25D0%25B8%25D1%2581._4.!12|se)

[Рис. 4.14. Цикл обміну даними. 14](#%25D0%25A0%25D0%25B8%25D1%2581._4.!13|se)

[Рис. 4.15. Налаштування швидкості передачі даних в Keil μVision. 15](#%25D0%25A0%25D0%25B8%25D1%2581._4.!14|se)

[Рис. 4.16. Успішна компіляція. 15](#%25D0%25A0%25D0%25B8%25D1%2581._4.!15|se)

1. Мета

Ознайомитися зі способом перевірки працездатності оперативної пам’яті налагоджувальної плати STM32F407G-DISC1 з використанням універсального асинхронного інтерфейсу.

1. Теоретичні відомості

Мікроконтролери STM32F407VGT6 базуються на високопродуктивному 32-бітному ядрі ARM Cortex-M4 RISC, що працює на частоті до 168 МГц. Цей чіп пропонує широкий спектр функцій, які роблять його ідеальним для широкого кола застосувань, включаючи промислове управління, медичні прилади, побутову техніку та багато іншого.

**Основні характеристики.**

* Ядро ARM Cortex-M4 RISC з частотою до 168 МГц
* 1 МБ флеш-пам'яті
* 192 КБ SRAM
* DMA з 8 каналами
* Гнучкий контролер статичної пам'яті (FSMC)
* Чотири універсальні синхронні/асинхронні приймачі-передавачі (USART)
* Два універсальні асинхронні приймачі-передавачі (UART)
* Інтерфейс Ethernet
* Криптографічний прискорювач
* Інтерфейс камери

**Детальний опис вузлів.**

* Ядро ARM Cortex-M4 RISC: Це ядро ​​забезпечує високу продуктивність і енергоефективність, що робить STM32F407VGT6 ідеальним вибором для ресурсомістких застосувань. Ядро ​​підтримує всі інструкції та типи обробки даних ARM одинарної точності, а також реалізує повний набір інструкцій DSP.
* Флеш-пам'ять: 1 МБ флеш-пам'яті використовується для зберігання програмного коду та даних.
* SRAM: 192 КБ SRAM використовується для зберігання тимчасових даних та змінних.
* DMA: DMA (Direct Memory Access) використовується для автоматичного переміщення даних між пам'яттю та периферійними пристроями, що зменшує навантаження на процесор.
* FSMC: FSMC (Flexible Static Memory Controller) використовується для підключення до зовнішньої статичної пам'яті, такої як SRAM, PSRAM, NOR Flash та NAND Flash.
* USART: Чотири USART забезпечують гнучкі можливості послідовного зв'язку. Вони підтримують різні швидкості передачі даних, формати даних та режими роботи.
* UART: Два UART забезпечують додаткові можливості послідовного зв'язку.
* Ethernet: Інтерфейс Ethernet дозволяє мікроконтролеру підключатися до мережі Ethernet.
* Криптографічний прискорювач: Криптографічний прискорювач використовується для апаратного прискорення криптографічних операцій.
* Інтерфейс камери: Інтерфейс камери дозволяє підключати до мікроконтролера камеру.

Мікроконтролер STM32F407VGT6 є потужним і універсальним чіпом, який може використовуватися в широкому колі застосувань. Його висока продуктивність, широкий спектр функцій та енергоефективність роблять його ідеальним вибором для розробників, які шукають надійне та гнучке рішення для вбудованих систем.

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

Варіант завдання:

* Розмір блоку даних: 832 байта;
* Швдикість обміну даними: 2400 біт/с.

1. Виконання завдання
   1. Створення консольної програми.

Запустив інтегроване середовище розробки Visual Studio.

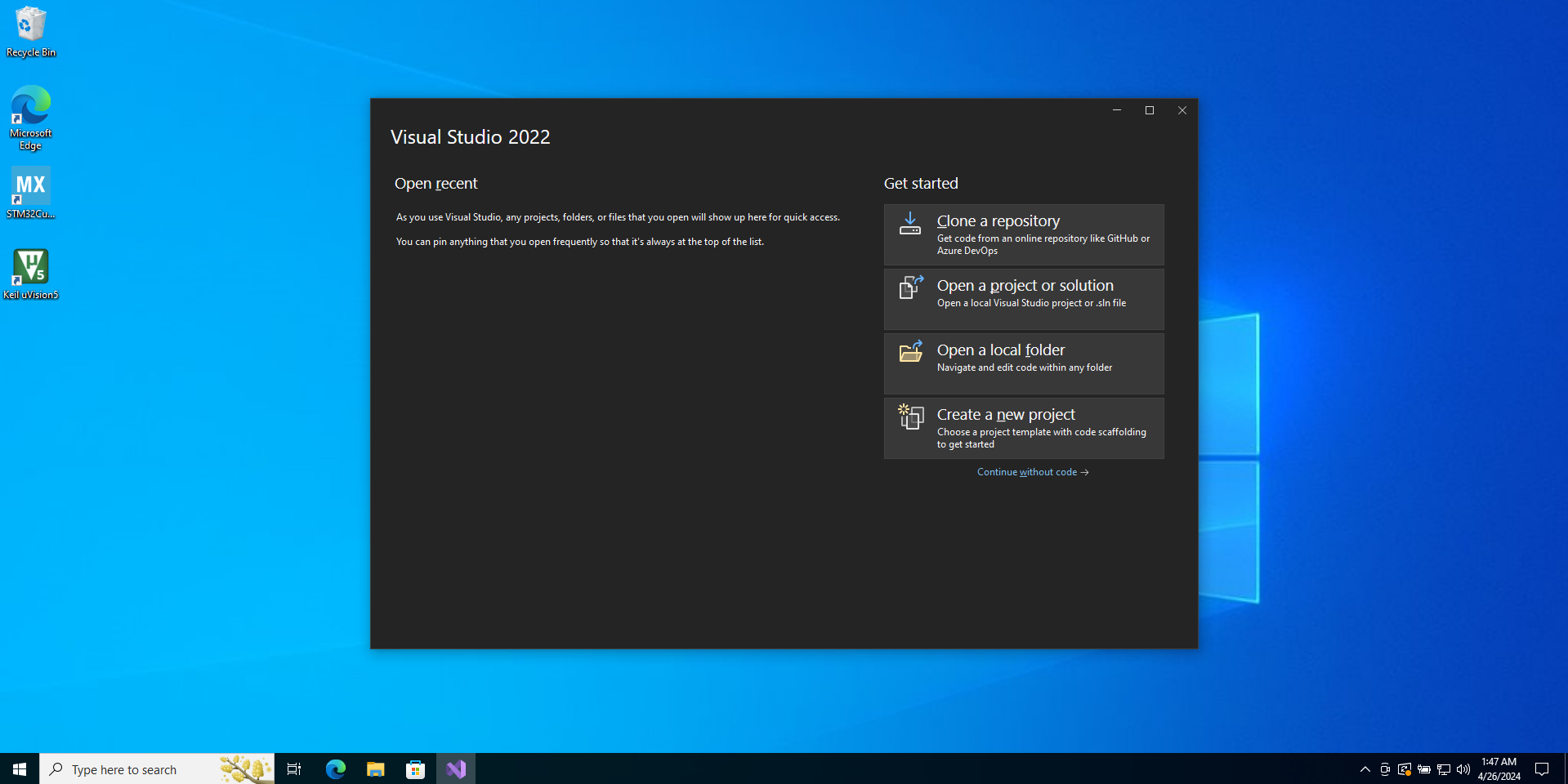


Рис. 4.1. Стартове вікно Visual Studio.

Створив та налаштував новий консольний проект.

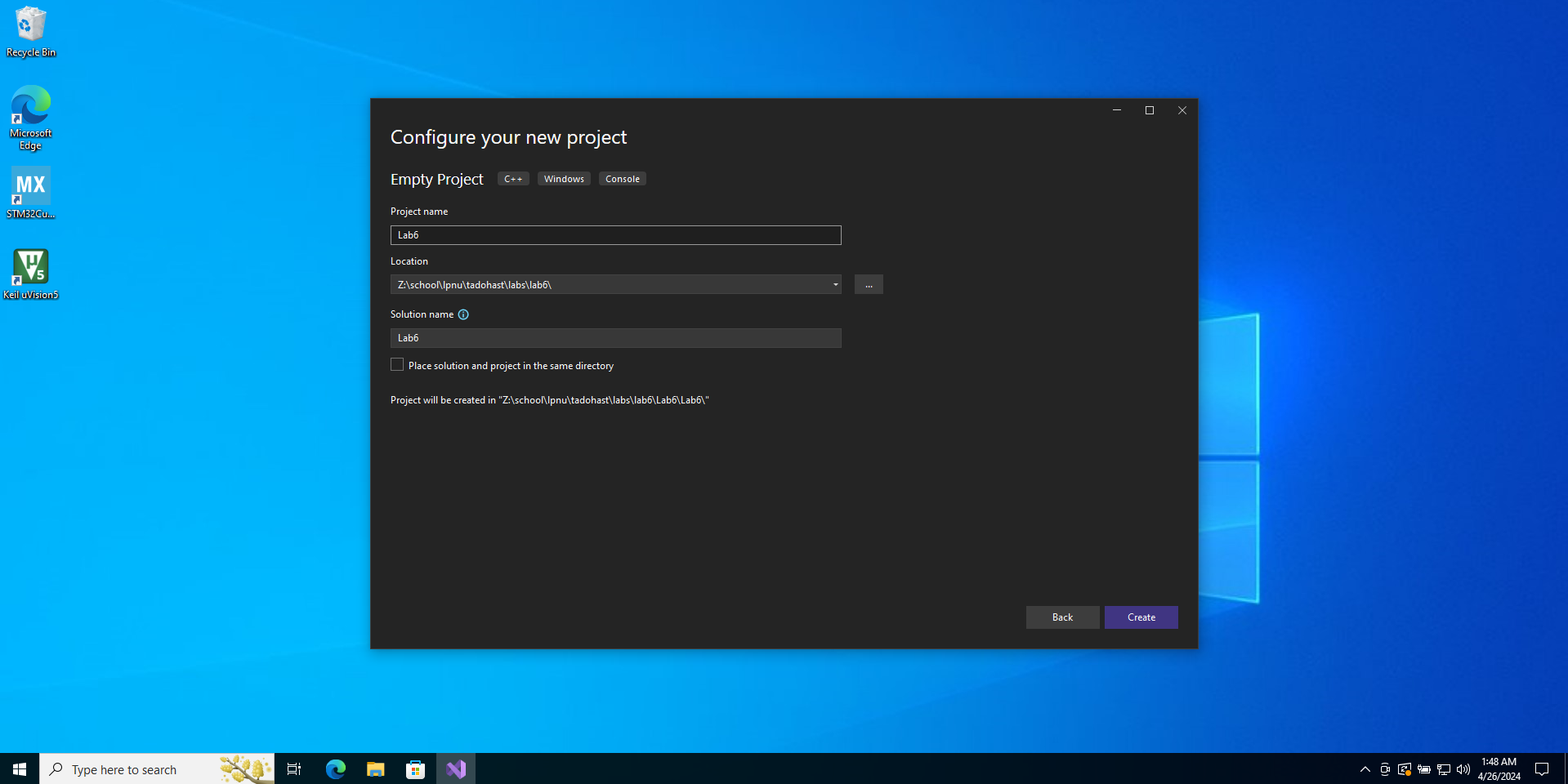


Рис. 4.2. Попередні налаштування проєкту.

У файл main.cpp вставив програмний код з методичних вказівок, попередньо змінивши розмір блоку даних і швидкість передачі.

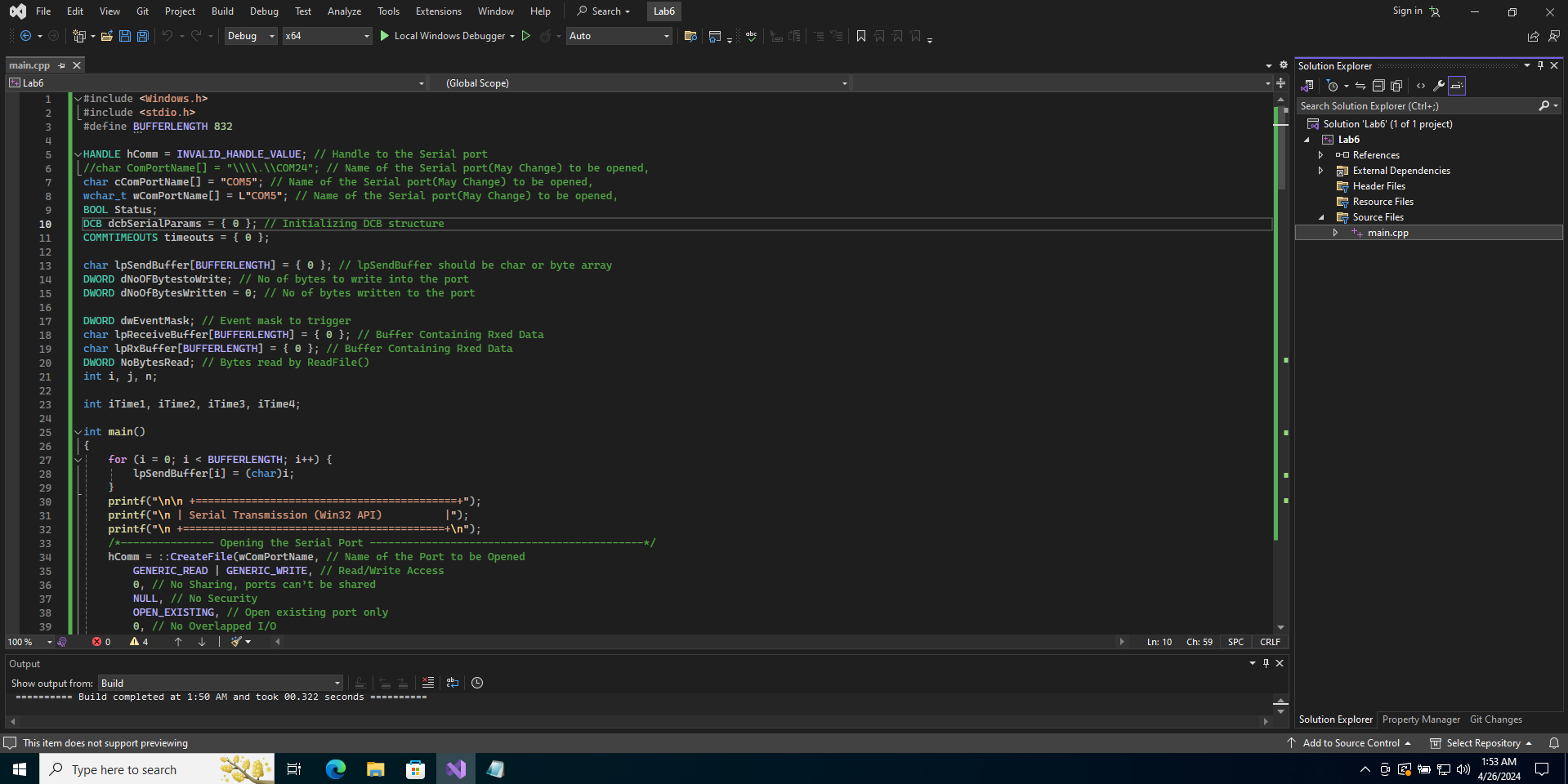


Рис. 4.3. програмний код.

Налаштування швидкості видно на наступному рисунку.

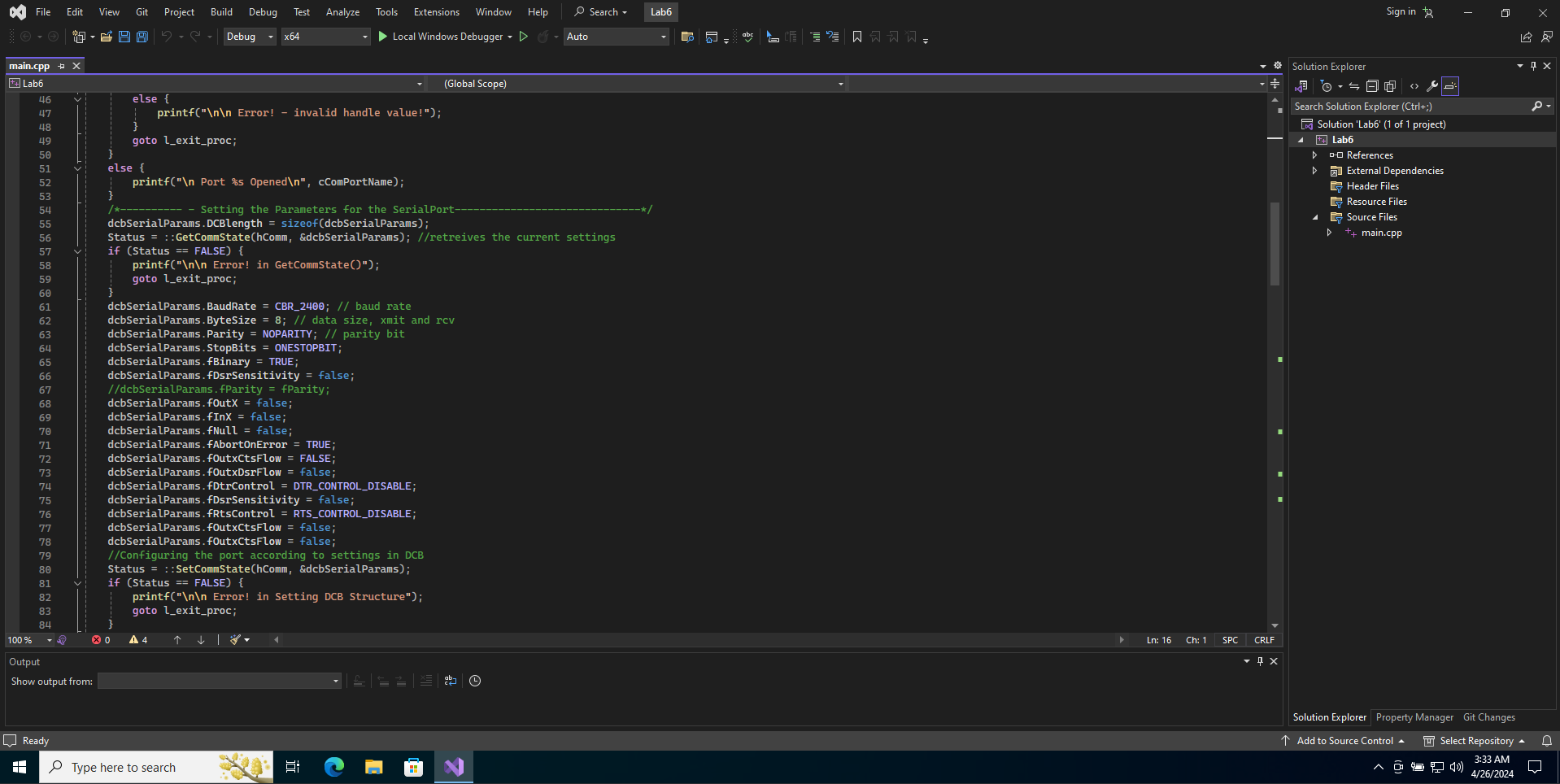


Рис. 4.4. Налаштування швидкості передачі даних у Visual Studio.

Скомпілював і запустив програму для перевірки відсутності помилок.

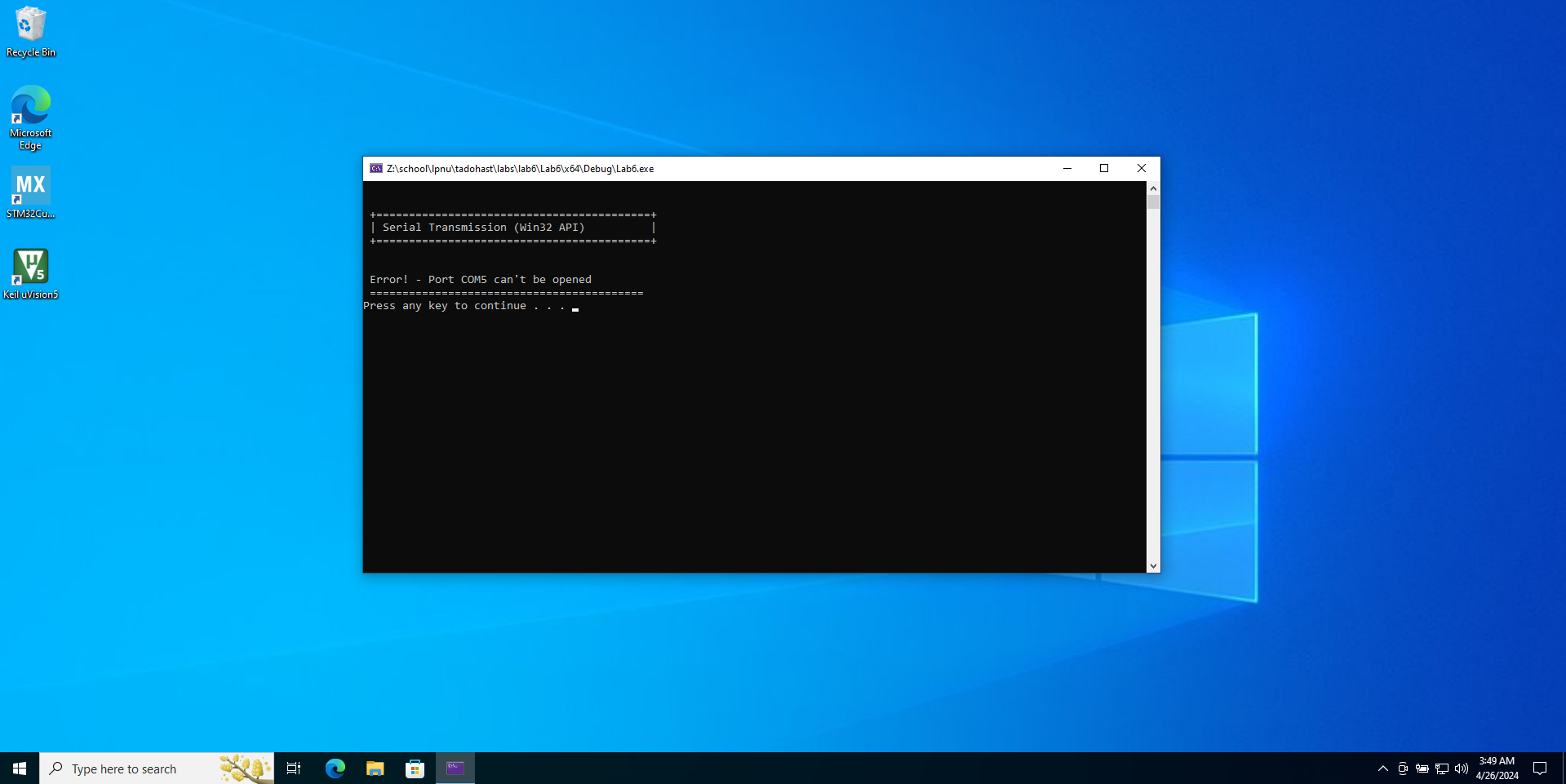


Рис. 4.5. Тестове виконання консольної програми.

* 1. Підготовка проєкту для налагоджувальної плати.

Запустив автоматизоване робоче місце розробки STM32CubeMX. В меню «New Project» обрав роботу з платами ST «Start My project from STBoard», натиснувши кнопку допступу до вибору плати. Дочекався завантаження необхідної інформації з інтернету. У вкладці «Board Selector» в розділі «Board Filters» обрав плату STM32F407G-DISC1 і натиснув на неї у списку плат.

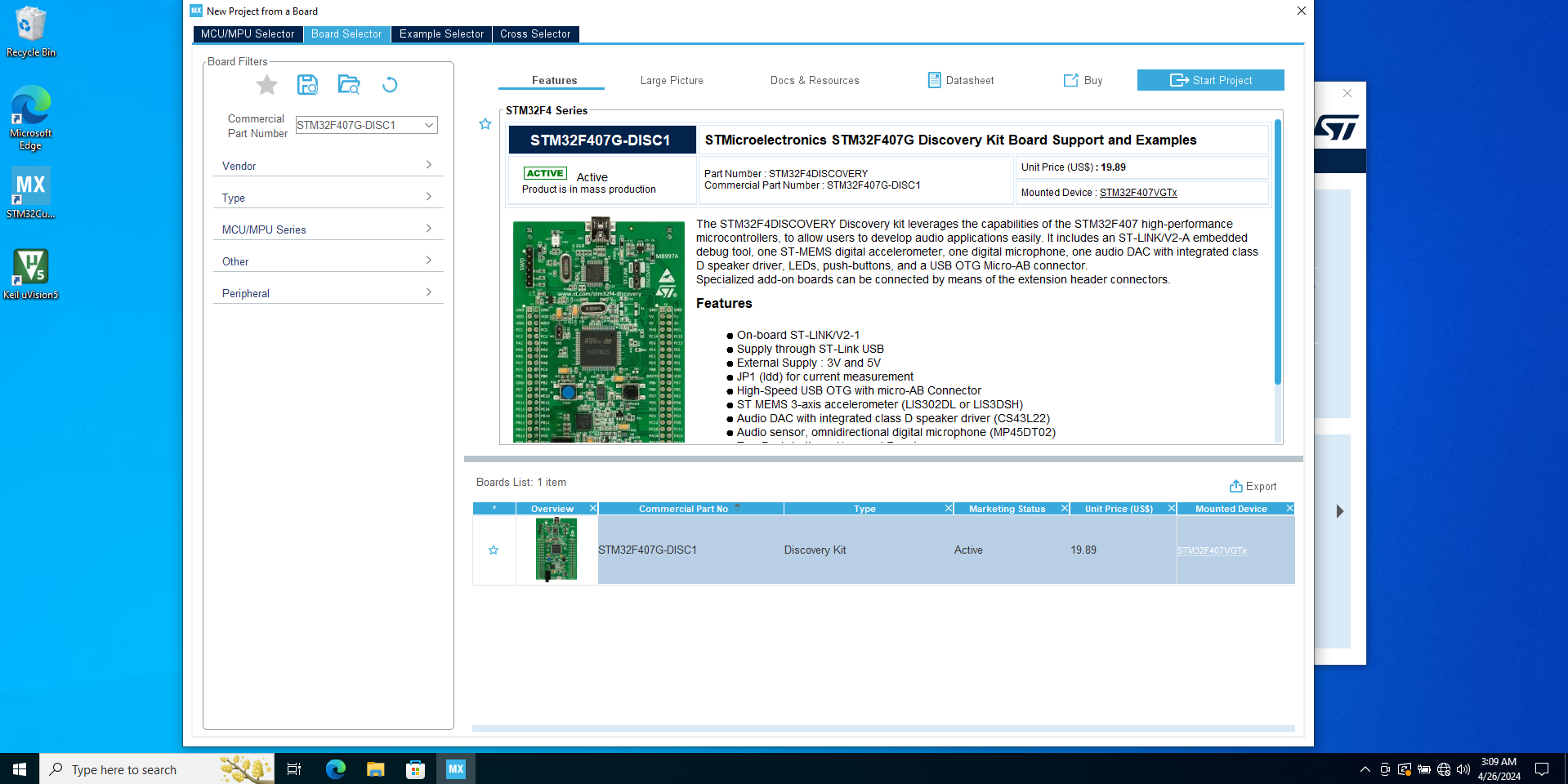


Рис. 4.6. Короткий опис плати STM32F407G-DISC1.

В розділі «System Core» обрав пункт «RCC». В розділі «RCC Mode and Configuration» встановив значення «Disable» для пункту «Low Speed Clock».

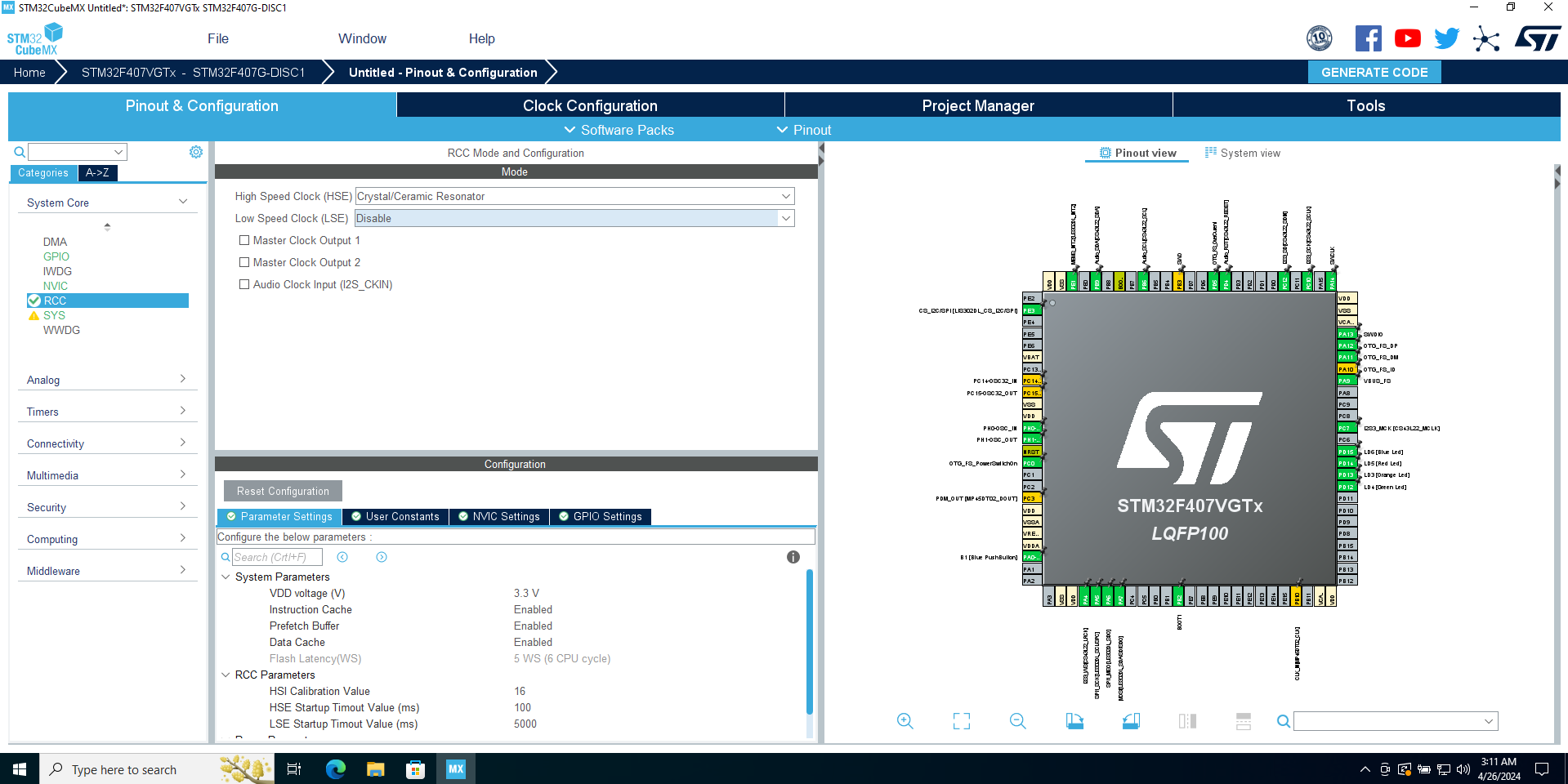


Рис. 4.7. Відключення кварцу 32 кГц.

На вкладці «Categories» розкрив меню «Connectivity» і вибрав пристрій «USART2». Встановив асинхронний режим роботи.

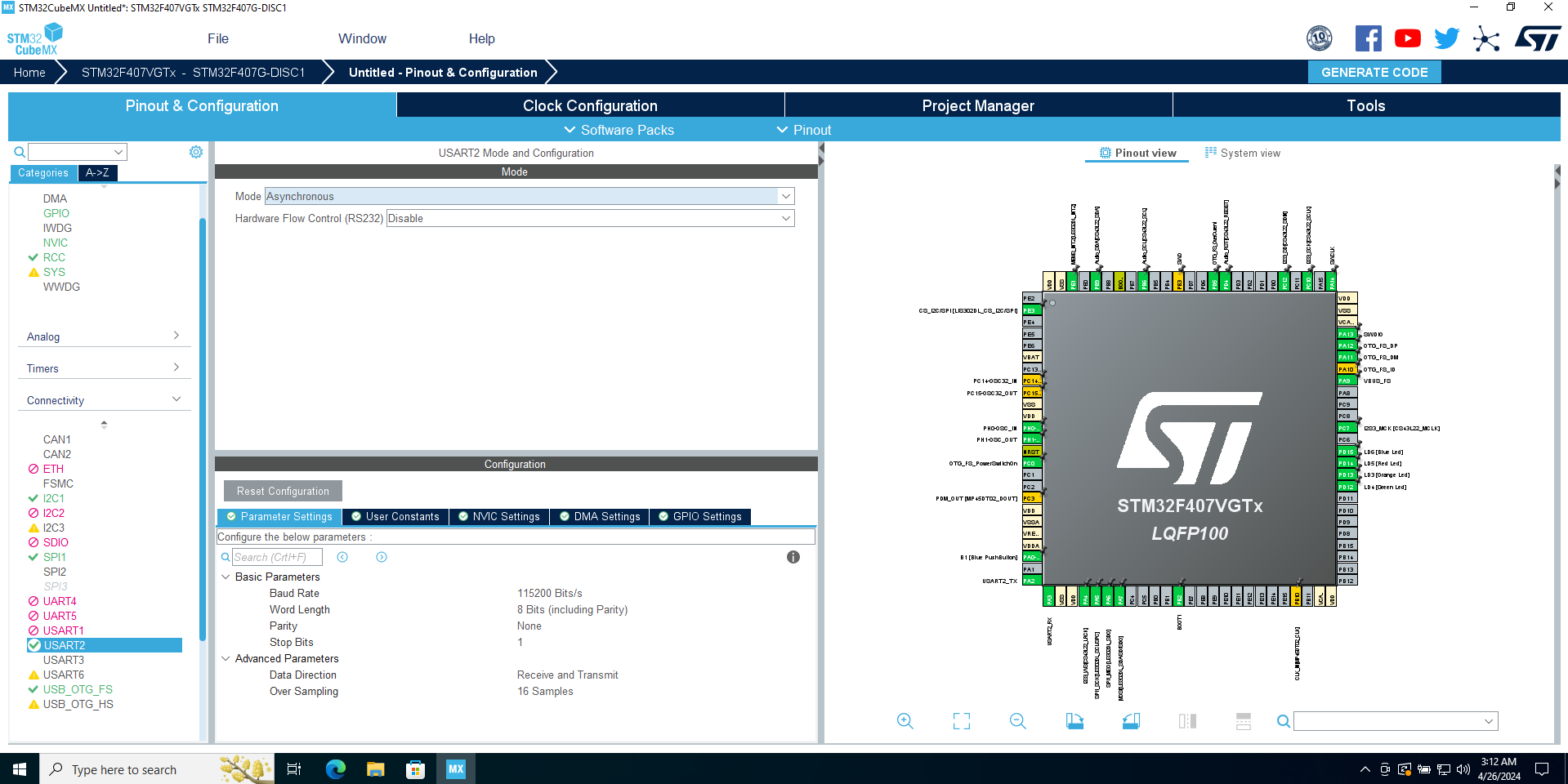


Рис. 4.8. Вибір роботи в асинхронному режимі.

На вкладці «NVIC Settings» встановив позначку «Enabled» для увімкнення підтримки апартних переривань.

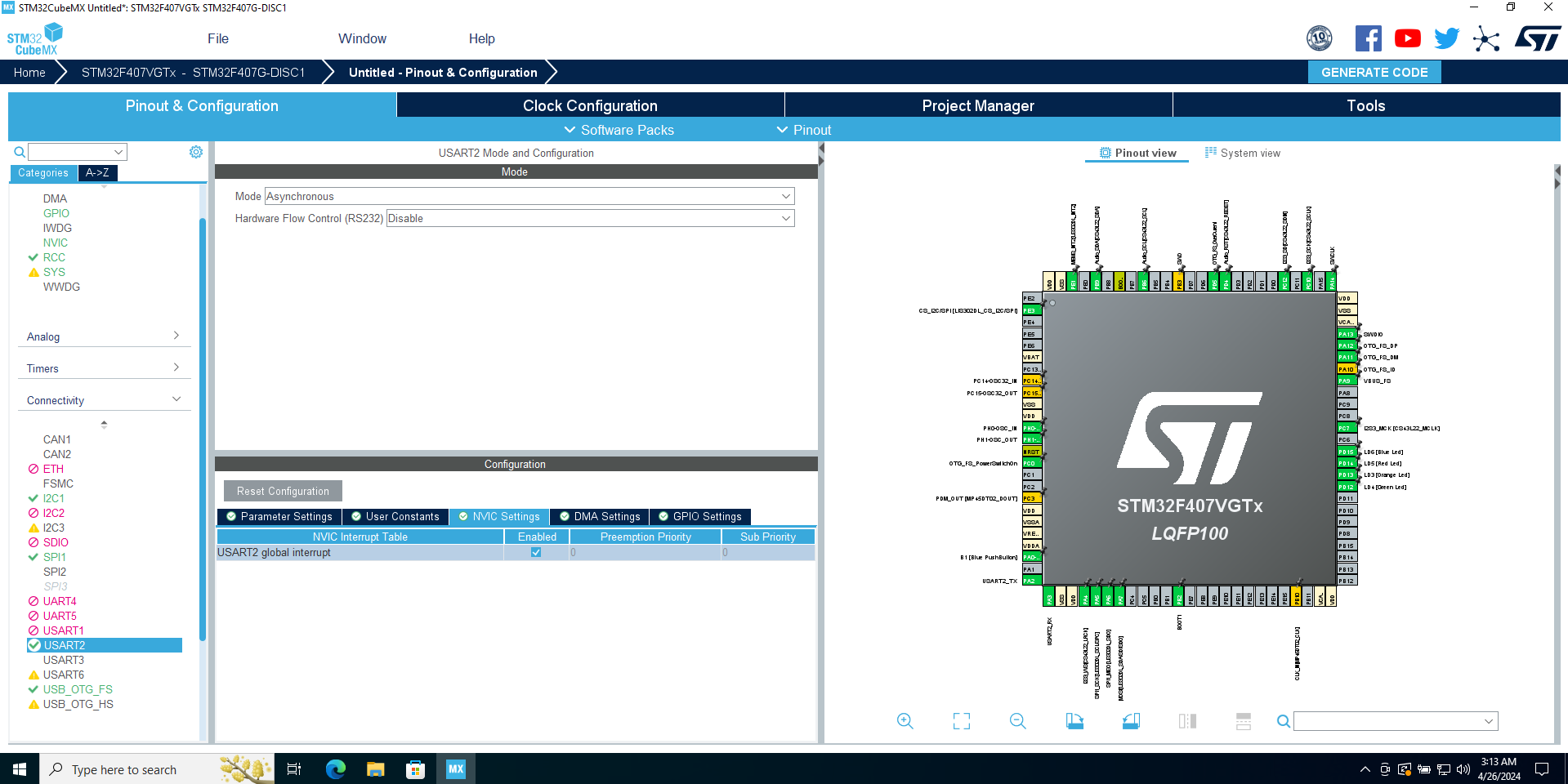


Рис. 4.9. Встановлення режиму з використанням апаратних переривань.

Перейшов на вкладку «Project Manager» і налаштував основні параметри проекту.

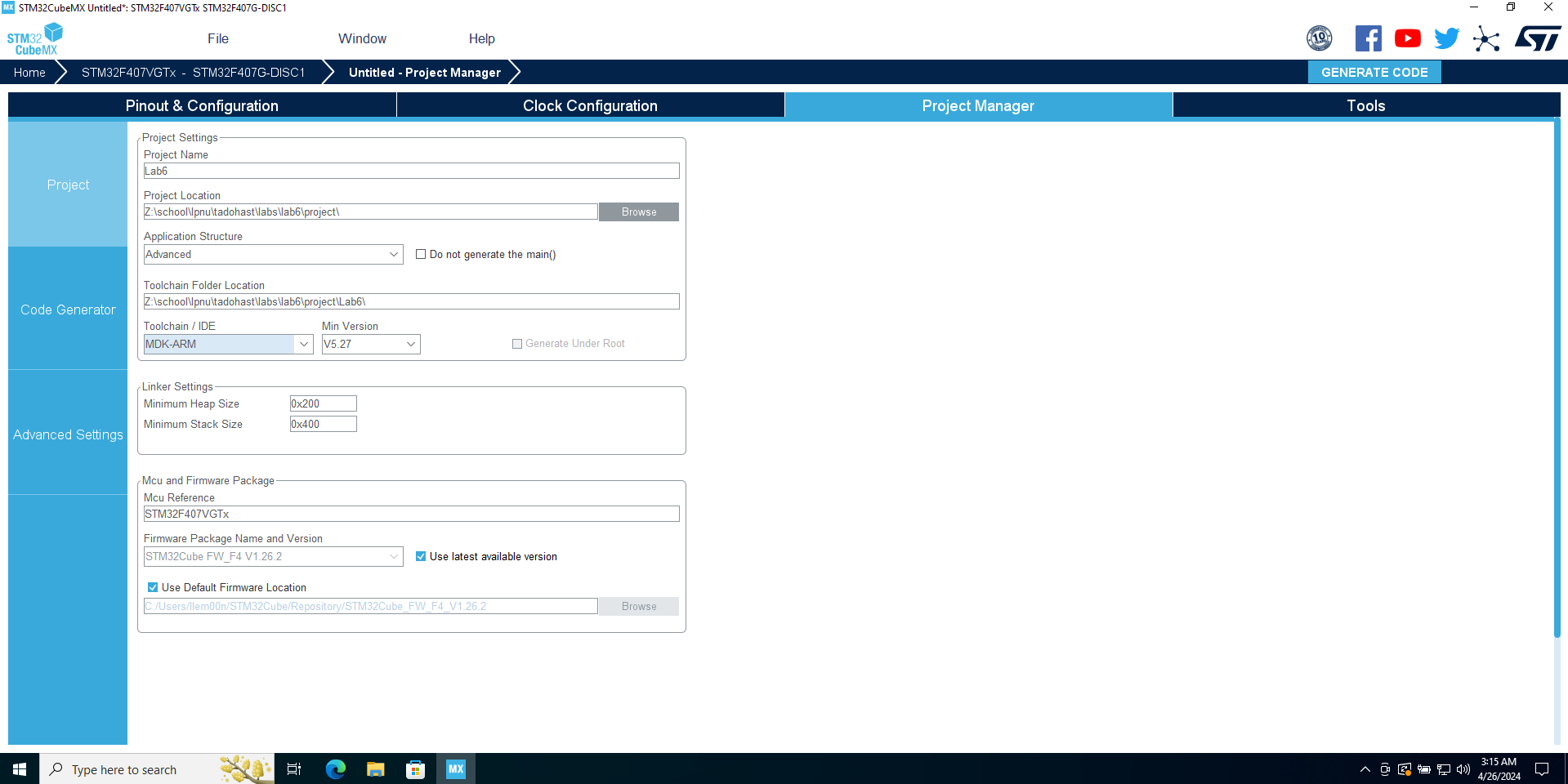


Рис. 4.10. Основні параметри проєкту.

Натиснув кнопку «GENERATE CODE», дочекався створення проекту і відкрив його у Keil μVision.

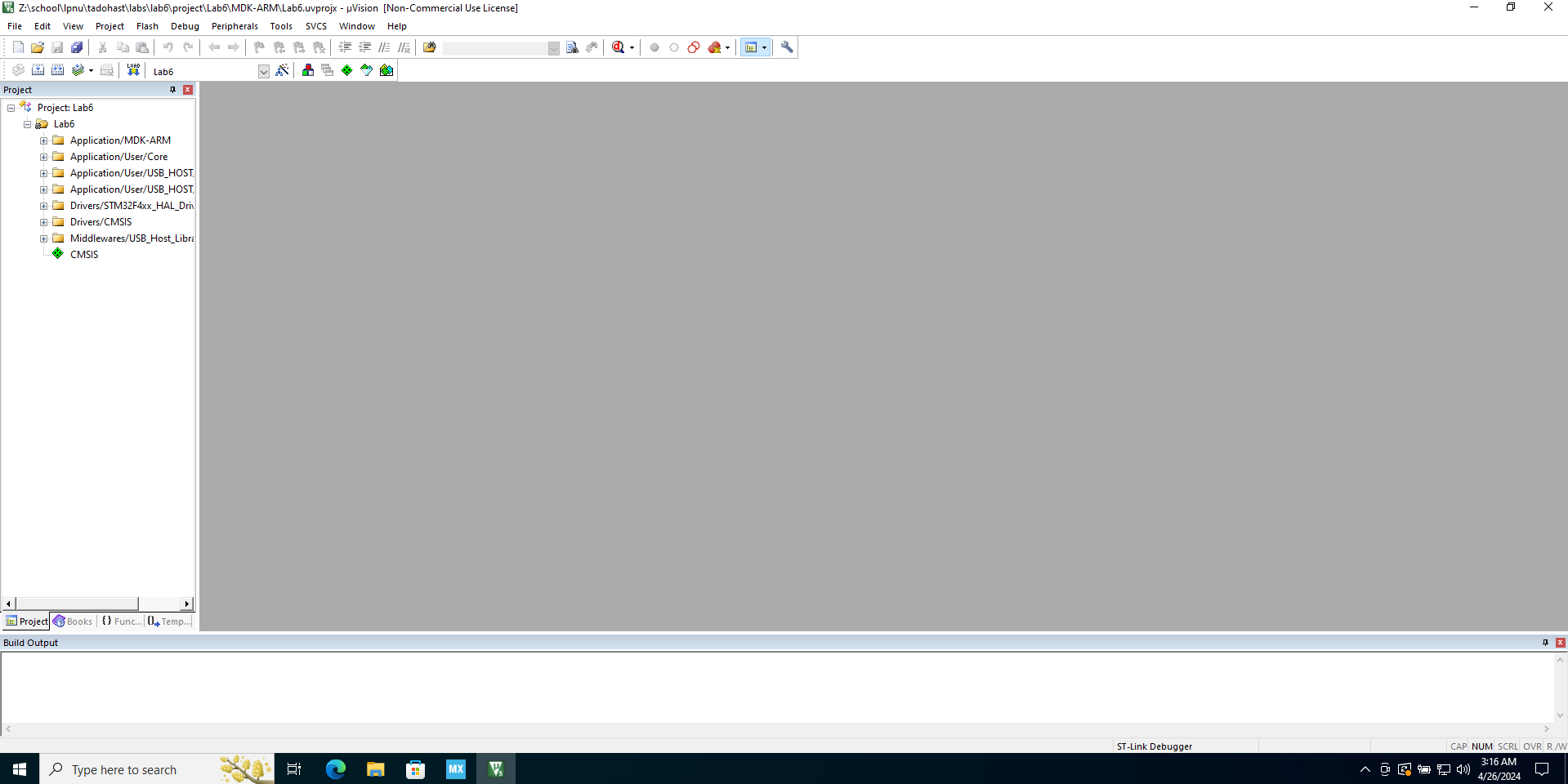


Рис. 4.11. Вікно MDK-ARM (IDE Keil μVision).

Скомпілював проект для перевірки відсутності помилок.

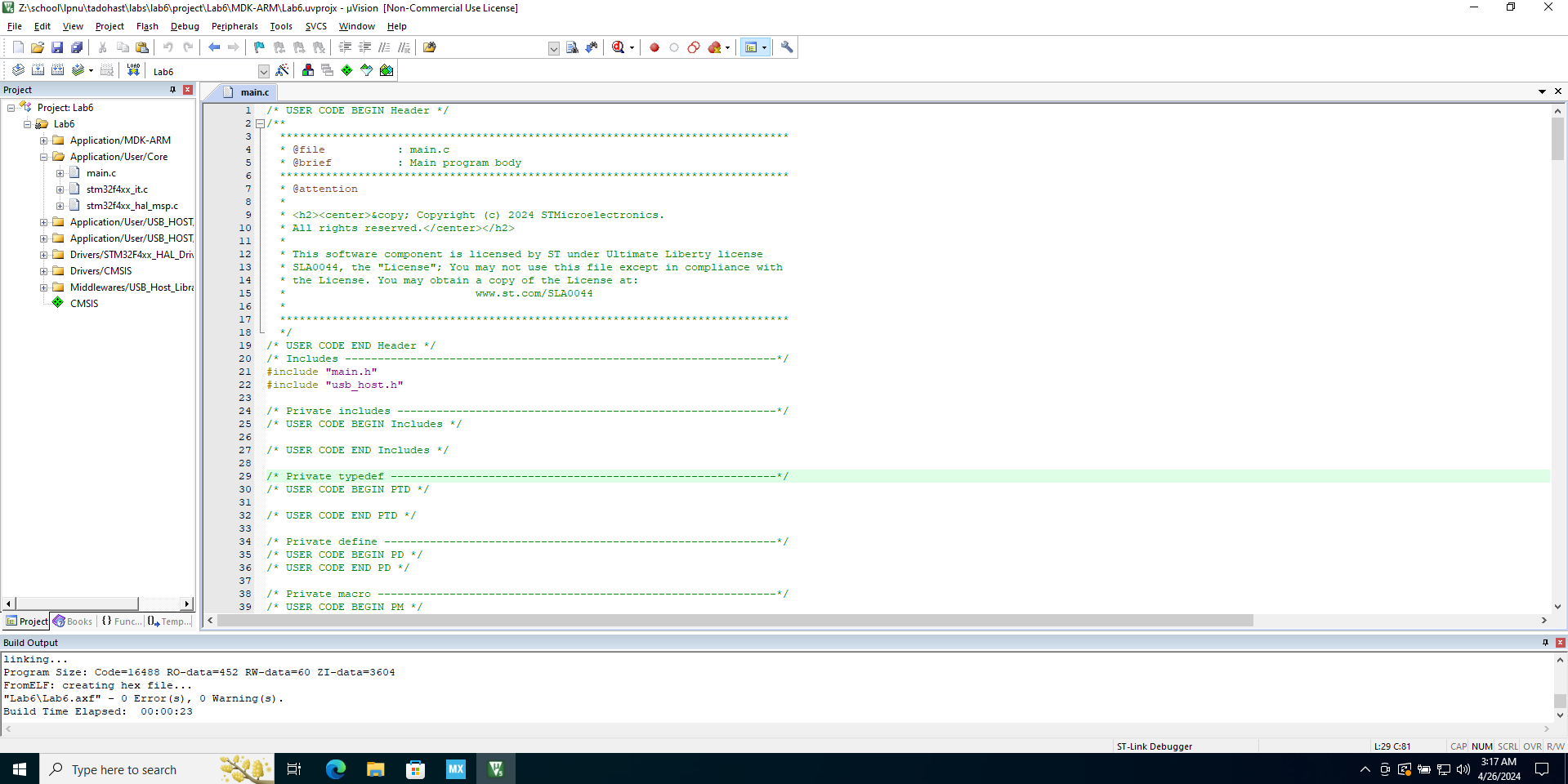


Рис. 4.12. Успішна компіляція.

Перейшов у розділ приватних оголошень, і додав визначення згідно свого варіанту.

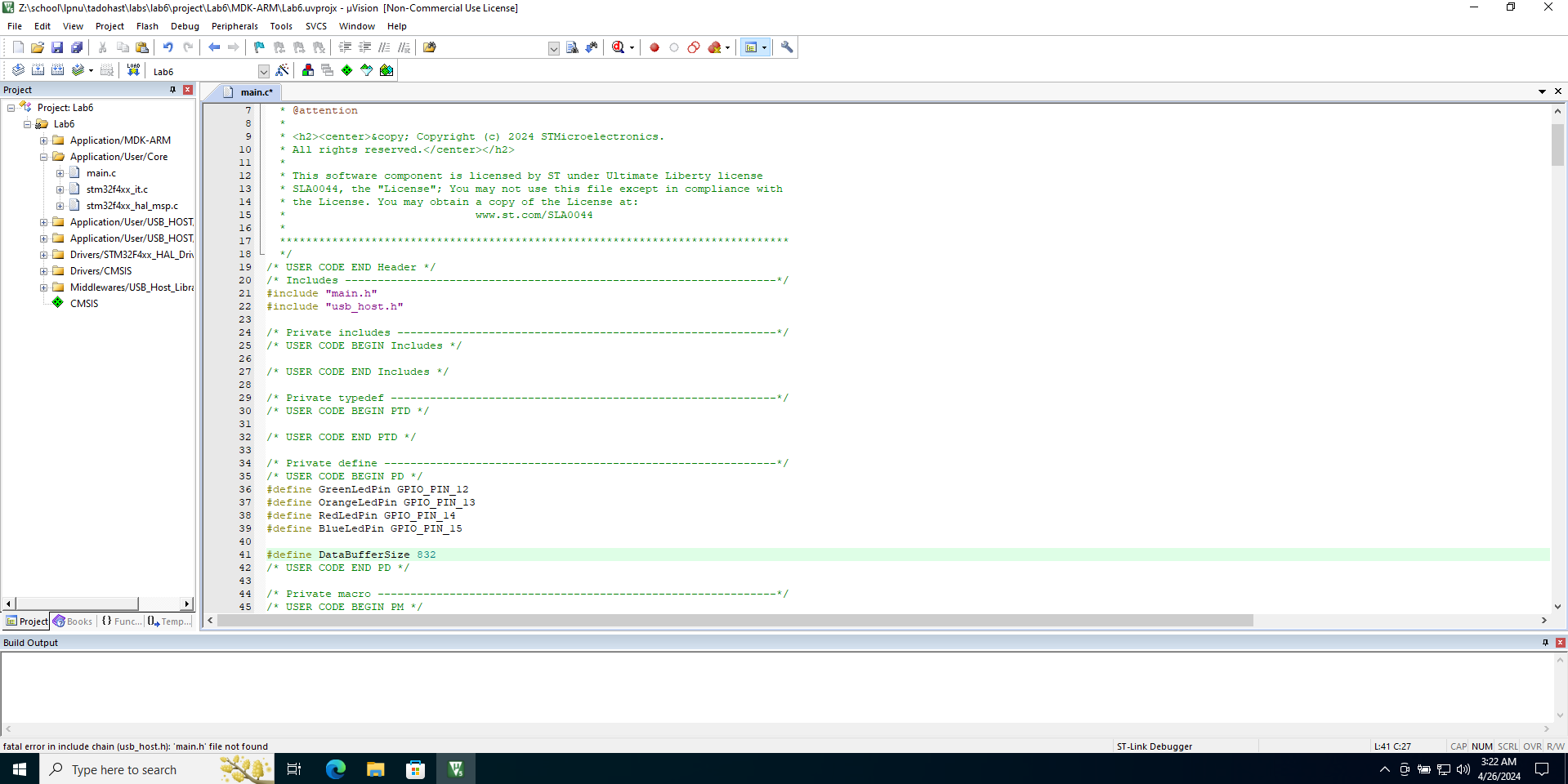


Рис. 4.13. Додаткові визначення розробника.

Перейшов до розділу основного циклу програми, і вставив код з методичних вказівок.

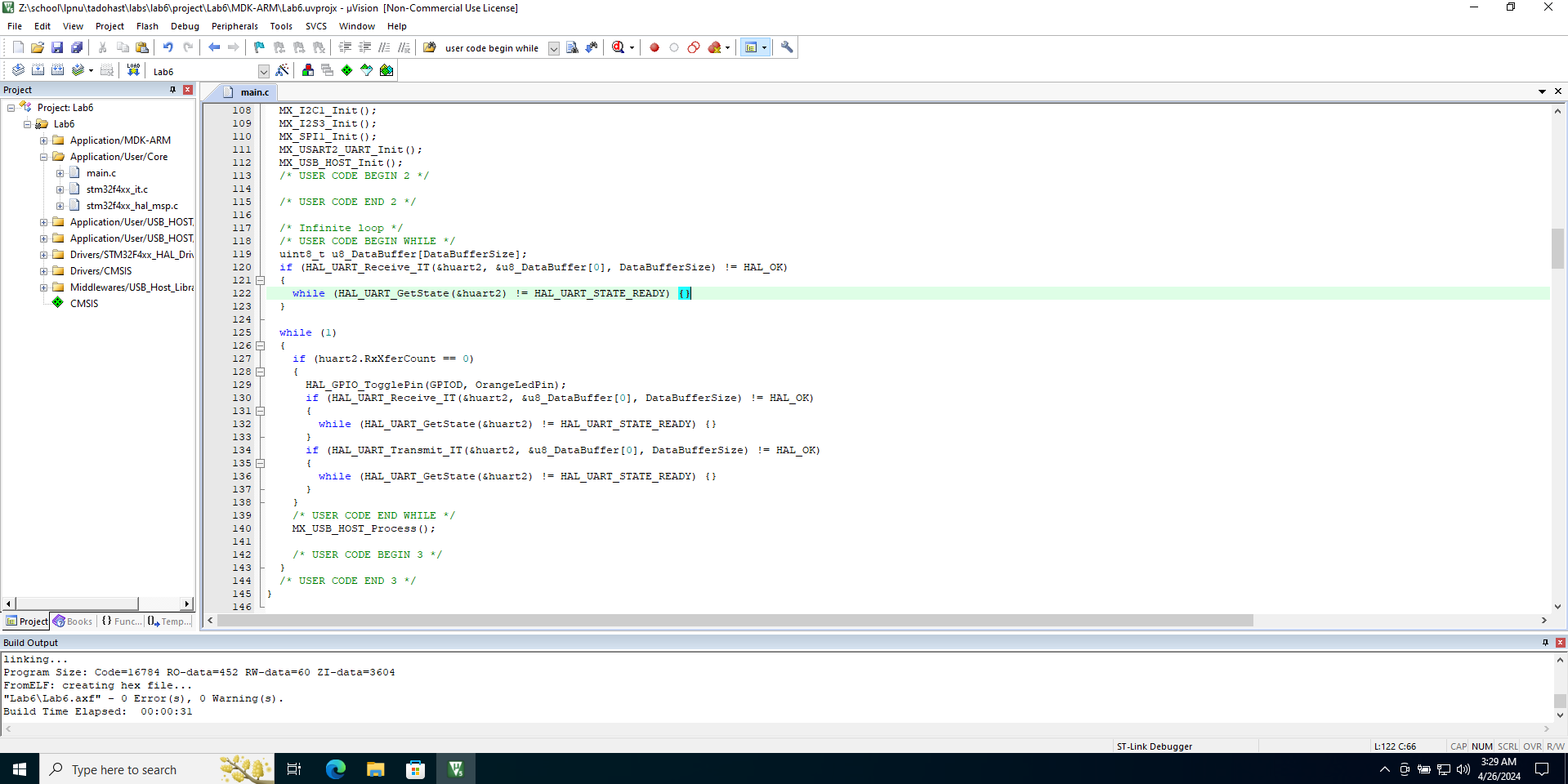


Рис. 4.14. Цикл обміну даними.

Встановив швидкість передачі даних згідно варіанту.

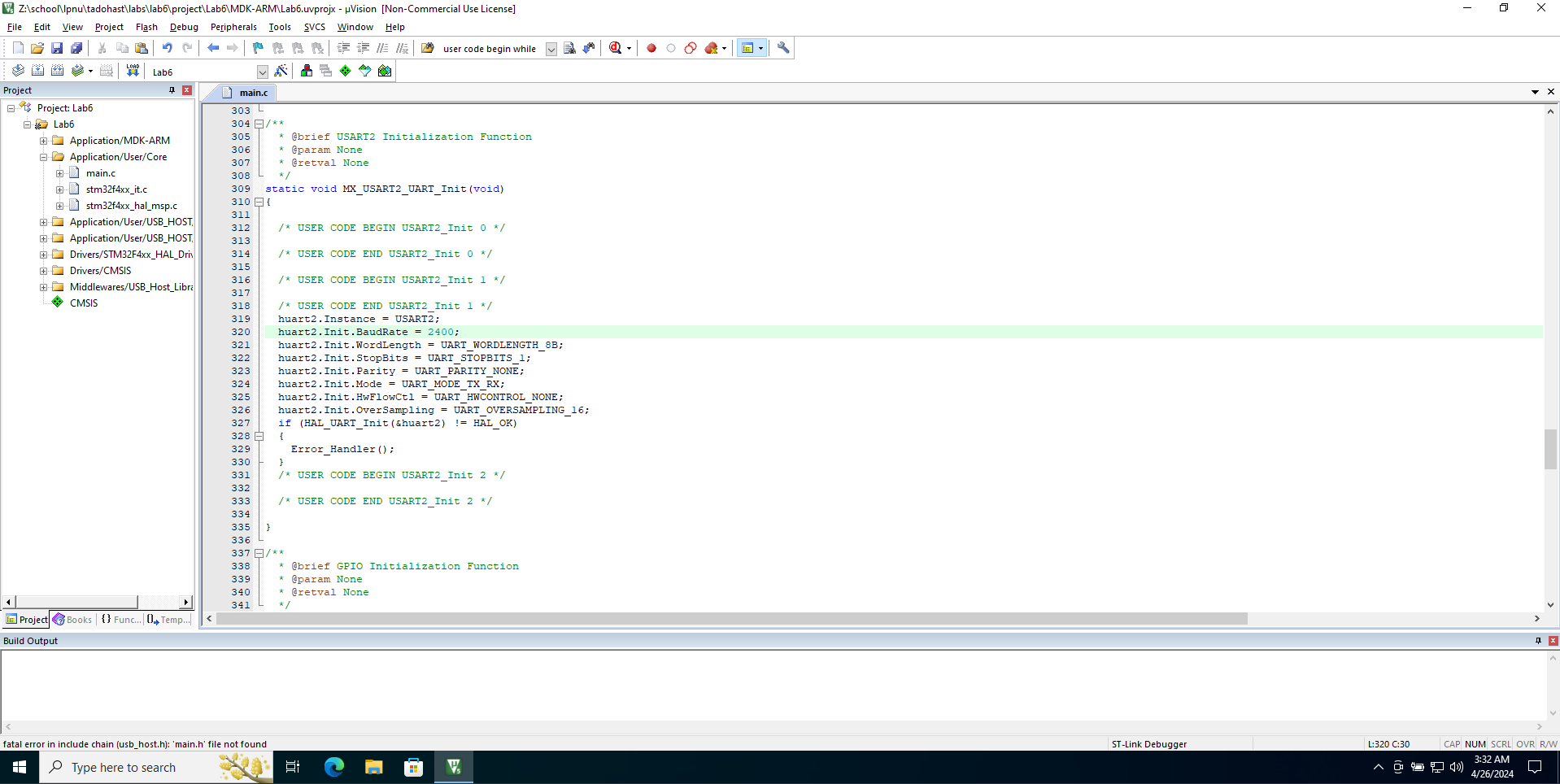


Рис. 4.15. Налаштування швидкості передачі даних в Keil μVision.

Скомпілював програму для перевірки відсутності помилок.

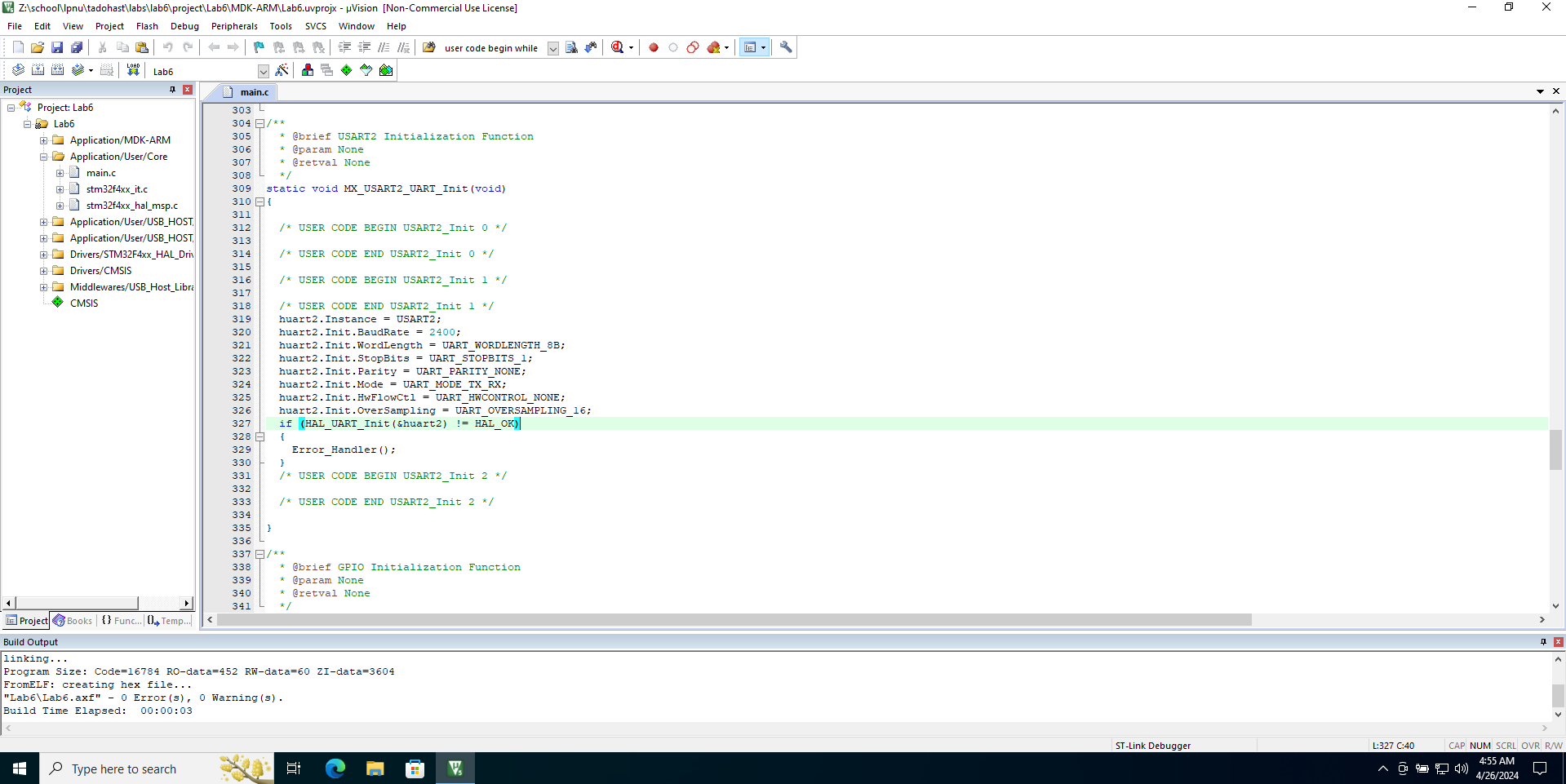


Рис. 4.16. Успішна компіляція.

Висновок

Я ознайомився зі способом перевірки працездатності оперативної пам’яті налагоджувальної плати STM32F407G-DISC1 з використанням універсального асинхронного інтерфейсу.

Список використаних джерел

1. STM32F405xx. STM32F407xx. Datasheet – production data [Електронний ресурс] – Режим доступу: https://www.st.com/resource/en/datasheet/stm32f407vg.pdf.
2. UM1472. User manual. Discovery kit with STM32F407VG MCU [Електронний ресурс] – Режим доступу: 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 [Електронний ресурс] – Режим доступу: 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 [Електронний ресурс] – Режим доступу: 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 [Електронний ресурс] – Режим доступу: https://armkeil.blob.core.windows.net/product/gs\_MDK5\_4\_en.pdf.

Додаток А. Код програми у Visual Studio

#include <Windows.h>

#include <stdio.h>

#define BUFFERLENGTH 832

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;

int main()

{

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\_2400; // 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;

}

Додаток Б. Код програми у Keil μVision.

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2024 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "usb\_host.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* 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 832

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

I2C\_HandleTypeDef hi2c1;

I2S\_HandleTypeDef hi2s3;

SPI\_HandleTypeDef hspi1;

UART\_HandleTypeDef huart2;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_I2C1\_Init(void);

static void MX\_I2S3\_Init(void);

static void MX\_SPI1\_Init(void);

static void MX\_USART2\_UART\_Init(void);

void MX\_USB\_HOST\_Process(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_I2C1\_Init();

MX\_I2S3\_Init();

MX\_SPI1\_Init();

MX\_USART2\_UART\_Init();

MX\_USB\_HOST\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* 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 \*/

MX\_USB\_HOST\_Process();

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_I2S;

PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;

PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief I2C1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C1\_Init(void)

{

/\* USER CODE BEGIN I2C1\_Init 0 \*/

/\* USER CODE END I2C1\_Init 0 \*/

/\* USER CODE BEGIN I2C1\_Init 1 \*/

/\* USER CODE END I2C1\_Init 1 \*/

hi2c1.Instance = I2C1;

hi2c1.Init.ClockSpeed = 100000;

hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C1\_Init 2 \*/

/\* USER CODE END I2C1\_Init 2 \*/

}

/\*\*

\* @brief I2S3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2S3\_Init(void)

{

/\* USER CODE BEGIN I2S3\_Init 0 \*/

/\* USER CODE END I2S3\_Init 0 \*/

/\* USER CODE BEGIN I2S3\_Init 1 \*/

/\* USER CODE END I2S3\_Init 1 \*/

hi2s3.Instance = SPI3;

hi2s3.Init.Mode = I2S\_MODE\_MASTER\_TX;

hi2s3.Init.Standard = I2S\_STANDARD\_PHILIPS;

hi2s3.Init.DataFormat = I2S\_DATAFORMAT\_16B;

hi2s3.Init.MCLKOutput = I2S\_MCLKOUTPUT\_ENABLE;

hi2s3.Init.AudioFreq = I2S\_AUDIOFREQ\_96K;

hi2s3.Init.CPOL = I2S\_CPOL\_LOW;

hi2s3.Init.ClockSource = I2S\_CLOCK\_PLL;

hi2s3.Init.FullDuplexMode = I2S\_FULLDUPLEXMODE\_DISABLE;

if (HAL\_I2S\_Init(&hi2s3) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2S3\_Init 2 \*/

/\* USER CODE END I2S3\_Init 2 \*/

}

/\*\*

\* @brief SPI1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_SPI1\_Init(void)

{

/\* USER CODE BEGIN SPI1\_Init 0 \*/

/\* USER CODE END SPI1\_Init 0 \*/

/\* USER CODE BEGIN SPI1\_Init 1 \*/

/\* USER CODE END SPI1\_Init 1 \*/

/\* SPI1 parameter configuration\*/

hspi1.Instance = SPI1;

hspi1.Init.Mode = SPI\_MODE\_MASTER;

hspi1.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi1.Init.DataSize = SPI\_DATASIZE\_8BIT;

hspi1.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi1.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi1.Init.NSS = SPI\_NSS\_SOFT;

hspi1.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

hspi1.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi1.Init.TIMode = SPI\_TIMODE\_DISABLE;

hspi1.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

hspi1.Init.CRCPolynomial = 10;

if (HAL\_SPI\_Init(&hspi1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN SPI1\_Init 2 \*/

/\* USER CODE END SPI1\_Init 2 \*/

}

/\*\*

\* @brief USART2 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART2\_UART\_Init(void)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 2400;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

if (HAL\_UART\_Init(&huart2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(CS\_I2C\_SPI\_GPIO\_Port, CS\_I2C\_SPI\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, OTG\_FS\_PowerSwitchOn\_Pin, GPIO\_PIN\_SET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : CS\_I2C\_SPI\_Pin \*/

GPIO\_InitStruct.Pin = CS\_I2C\_SPI\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(CS\_I2C\_SPI\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_PowerSwitchOn\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_PowerSwitchOn\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : PDM\_OUT\_Pin \*/

GPIO\_InitStruct.Pin = PDM\_OUT\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(PDM\_OUT\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : BOOT1\_Pin \*/

GPIO\_InitStruct.Pin = BOOT1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(BOOT1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : CLK\_IN\_Pin \*/

GPIO\_InitStruct.Pin = CLK\_IN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(CLK\_IN\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : LD4\_Pin LD3\_Pin LD5\_Pin LD6\_Pin

Audio\_RST\_Pin \*/

GPIO\_InitStruct.Pin = LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_OverCurrent\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_OverCurrent\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(OTG\_FS\_OverCurrent\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : MEMS\_INT2\_Pin \*/

GPIO\_InitStruct.Pin = MEMS\_INT2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(MEMS\_INT2\_GPIO\_Port, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/