Вінницький національний технічний університет
Факультет інфокомунікацій, радіоелектроніки та наносистем
Кафедра радіотехніки

ПОЯСНЮВАЛЬНА ЗАПИСКА
до магістерської кваліфікаційної роботи
Рівень вищої освіти другий (магістерський)

на тему:
УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ
08-36.МКР.002.00.000 ПЗ

Виконав: ст. 2–го курсу, групи РТ–19м
Спеціальність 172 – Телекомунікації та радіотехніка Освітня програма: Радіотехніка
_____Онофрійчук В.А.

Керівник: д.т.н., проф. каф. РТ
_____ Осадчук О.В.
«____» _____ 2021 р.

Рецензент: к.т.н., доцент. каф. ТКСТБ
_____
«____» _____ 2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет

Факультет інфокомунікацій, радіоелектроніки та наносистем
Кафедра Радіотехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 172 – Телекомунікації та радіотехніка
Освітня програма – Радіотехніка
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри РТ
д.т.н., професор О.В. Осадчук
“_10_” ____03_____2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Онофрійчука Володимира Анатолійовича
(прізвище, ім’я, по батькові)

1. Тема роботи «Універсальний програматор для сучасних мікроконтролерів»

керівник роботи Осадчук Олександр Володимирович, д.т.н., професор, зав. каф. РТ
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “09” 03 2021 року №64
2. Строк подання студентом роботи 06 червня 2021 року.
3. Вихідні дані до роботи: Номінальна напруга +5 В; +3,3 В; споживана потужність не повинна перевищувати 0,05 Вт; струм споживання знаходиться в межах 1-15 мА; діапазон робочих частот становить 50 кГц - 6500 кГц; діапазон робочих температур складає 0…+60 °С.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити: Вступ. Сучасний стан розвитку пристроїв для програмування мікроконтролерів. Розробка апаратної частини універсального програматора для сучасних мікроконтролерів. Схемотехнічна та програмна реалізація пристрою для програмування універсального програматора для сучасних мікроконтролерів. Економічна частина. Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік посилань. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Автономні програматори. Будова автономного комп’ютерного програматора. Склад областей пам’яті мікроконтролера виробництва Atmel. Різновиди систем програмування. Спрощена електрична схема USB з’єднання. Паралельне програмування ATTiny2313. Схема з’єднання програматора з МК AVR при програмуванні через SPI. Структурна схема пристрою.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	д.т.н., професор Осадчук О.В.		
Охорона праці та безпека в надзвичайних ситуаціях	Професор кафедри БЖДПБ, доцент, д.п.н., Дембіцька С.В.		
Економічна частина	ст. викл. каф. ЕПВМ к.е.н., Кавецький В.В.		

7. Дата видачі завдання 11 березня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Огляд літературних джерел. Вибір, узгодження та затвердження теми МКР	03.09.2020-31.12.2021	
2.	Аналіз літературних джерел. Попередня розробка основних розділів	07.01.2021-14.02.2021	
3.	Затвердження теми. Розробка технічного завдання	15.02.2021-10.03.2021	
4.	Аналіз вирішення поставленої задачі. Розробка структурної схеми	01.03.2021-21.03.2021	
5.	Електричні розрахунки. Експериментальне дослідження	22.03.2021-11.04.2021	
6.	Розділ моделювання	12.04.2021-19.04.2021	
7.	Розробка графічної частини МКР	20.04.2021-30.04.2021	
8.	Аналіз економічної ефективності розробки	01.05.2021-14.05.2021	
9.	Охорона праці (ОП)	15.05.2021-20.05.2021	
10	Оформлення пояснювальної записки та графічної частини	21.05.2021-26.05.2021	
11.	Нормоконтроль	27.05.2021-31.05.2021	
12.	Попередній захист МКР, доопрацювання, рецензування МКР	01.06.2021-06.06.2021	
13.	Захист МКР ЕК	07.06.2021-11.06.2021	

Студент _____ Онофрійчук В. А.
(підпис)

Керівник роботи _____ Осадчук О.В.
(підпис)

РЕФЕРАТ

УДК 621.397

Онофрійчук В.А. Універсальний програматор для сучасних мікроконтролерів. Магістерська кваліфікаційна робота. – Вінниця: ВНТУ, 2021. – 161с. На українській мові. Бібліогр.: 51 назв; Рисунок 45.

У магістерській кваліфікаційній роботі проведено аналіз публікацій, присвячених теоретичним і експериментальним дослідженням схемотехнічної та програмної реалізації програматорів для мікроконтролерів. Досліджено протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів. Розглянуто набір команд завантажувача їх алгоритми та принцип роботи. Розглянуто інтерфейси програмування мікроконтролерів STM32. Мікроконтролери STM32 в залежності від моделі підтримують від двох до шести способів завантаження програми у пам’ять. Максимальна кількість можливих варіантів bootloader у одному мікроконтролері – 5, а саме через протоколи USART, I2C, SPI, CAN, DFU.

Розроблено пристрій для програмування восьми та тридцяти двох розрядних мікроконтролерів, він складається із чотирьох структурних блоків. Розроблений пристрій складається із трьох інтерфейсів обміну даними та основного блоку. Приймач/передавач USB відповідає за перетворення прийнятих від USB порту комп’ютера сигналів у зрозумілі основному блокові. Даний блок працює у режимі Full Speed. Блоки UART та SPI здійснюють передачу сигналів програмування від основного блоку до мікроконтролерів та у зворотному напрямі.

Розроблено структурну та принципову схему пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів. Розроблено друковану плату пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів в пакеті програм DipTrace, а також розроблено 3D модель друкованої плати з елементами. Розглянуто алгоритм прошивання мікроконтролера STM32F103C8, як основного елемента розробленого пристрою.

У четвертому розділі описано рекомендації щодо охорони праці та безпеки при роботі з даним пристроєм.

У п’ятому розділі проведено розрахунок кошторису витрат на виробництво пристрою та ефективність вкладених інвестицій.

У роботі проведено розрахунки економічної частини, а також розділу охорони праці та безпеки в надзвичайних ситуаціях.

Ключові слова: мікроконтролер, програматор, інтерфейси обміну даними, алгоритм роботи пристрою.

ABSTRACT

Onofriychuk V.A. Universal programmer for modern microcontrollers. - Master's thesis. - Vinnytsia: VNTU, 2021. - 161p. In Ukrainian. Bibliogr .: 51 titles; Figure 45.

In the master's qualification work the analysis of the publications devoted to theoretical and experimental research of circuit engineering and software realization of programmers for microcontrollers is carried out. Microcontroller programming protocols and types of devices for microcontroller programming are studied. The set of bootloader commands, their algorithms and the principle of operation are considered. The programming interfaces of STM32 microcontrollers are considered. Depending on the model, STM32 microcontrollers support two to six ways to load the program into memory. The maximum number of possible bootloader options in one microcontroller is 5, namely via USART, I2C, SPI, CAN, DFU protocols.

A device for programming eight and thirty-two bit microcontrollers has been developed, it consists of four structural units. The developed device consists of three interfaces of data exchange and the main block. The USB receiver / transmitter is responsible for converting the signals received from the computer's USB port into a clear main unit. This unit operates in Full Speed mode. The UART and SPI units transmit programming signals from the main unit to the microcontrollers and vice versa.

The structural and schematic diagram of the device for programming of eight and thirty-two bit microcontrollers is developed. The printed circuit board of the device for programming of eight and thirty-two bit microcontrollers in the DipTrace software package was developed, and also the 3D model of the printed circuit board with elements was developed. The algorithm for flashing the STM32F103C8 microcontroller as the main element of the developed device is considered.

The fourth section describes the recommendations for occupational safety and health when working with this device.

The fifth section calculates the estimated cost of manufacturing the device and the efficiency of the investment.

The calculations of the economic part, as well as the section on labor and safety in emergency situations, were carried out.

Keywords: microcontroller, programmer, data exchange interfaces, device operation algorithm.

ЗМІСТ

ВСТУП

У сучасному світі важко уявити своє життя без електроніки. Електронні пристрої є важливим помічником сучасної людини, так як вони виконують більшу частину складної рутинної роботи. Але сучасні пристрої мають бути по можливості невеликими за розмірами, та обов’язково надійні у своїй роботі. Пристрої, побудовані на дискретних елементах низького ступеню інтеграції уже не можуть забезпечити потреб сучасної людини, тож в наш час перевага надається електронним пристроям, побудованим на мікроконтролерах.

Мікроконтролери використовується для керування електронними пристроями. По суті, це однокристальний комп'ютер, здатний виконувати прості завдання. Один і той же пристрій, який раніше збирався на традиційних елементах, будучи зібраним із застосуванням мікроконтролерів, стає простішим. Він не потребує регулювання і суттєво менший за розмірами. Використання однієї мікросхеми значно знижує розміри, енергоспоживання і вартість пристроїв, побудованих на базі мікроконтролерів.

Мікроконтролери можна зустріти в багатьох сучасних приладах, таких як телефони, пральні машини, вони відповідають за роботу двигунів і систем гальмування сучасних автомобілів, з їх допомогою створюються системи контролю і системи збору інформації. Переважна більшість процесорів, що випускаються у світі — мікроконтролери [1].

Перший мікроконтролер був розроблений в 1971 році інженером Gary W. Boone, співробітником «Texas Instruments». У 1980 році фірма Intel випустила мікроконтролер «i8048». Трохи пізніше в цьому ж році «Intel» випустила наступний мікроконтролер: «i8051». Вдалий набір периферійних пристроїв, можливість гнучкого вибору зовнішньої або внутрішньої програмної пам'яті і прийнятна ціна забезпечили цьому мікроконтролеру успіх на ринку. З погляду технології мікроконтролер i8051 був для свого часу дуже складним виробом — у кристалі було використано 128 тисяч транзисторів, що в 4 рази перевищувало кількість транзисторів в 16-розрядному мікропроцесорі i8086 [2].

На сьогоднішній день існує більше двохсота модифікацій мікроконтролерів, що сумісні з i8051, які випускаються двома десятками

компаній, і велика кількість мікроконтролерів інших типів. Популярністю у розробників користуються 8-бітові мікроконтролери PIC від фірми «Microchip Technology» і «AVR» від фірми «Atmel». Також останнім часом дуже популярним у розробників стали мікроконтролери побудовані на ядрі ARM, зокрема заслуженою популярністю користуються тридцяти двох бітні мікроконтролери фірми STMicroelectronics STM32.

Задача розробки пристроїв із використанням мікроконтролерів потребує знання та розуміння принципів їх роботи, але головне – вміння створювати керуючі програми. Без програми мікроконтролер просто непотрібний шматок кремнію.

Мікроконтролери можна програмувати за допомогою кількох різних інтерфейсів. Найбільш поширеним із них являється ISP, тобто внутрішньо схемне програмування. Що ж стосується тридцяти двох бітних мікроконтролерів – найпростішим та найпоширенішим способом їх програмування являється інтерфейс UART.

Внутрішньо схемне програмування (англ. In-System Programming, скор. ISP) - технологія програмування електронних компонентів (ПЛІС, мікроконтролери і т. п.), що дозволяє програмувати компонент, вже встановлений у пристрій.

До появи цієї технології компоненти програмувались перед установкою в пристрій, для їх перепрограмування потрібно було діставати контролер з пристрою. Головною перевагою технології є можливість об'єднання процесу програмування та тестування при виробництві, виключивши окрему фазу програмування компонентів перед остаточною збіркою. Технологія також дозволяє виробникам пристроїв обійтися без закупівлі заздалегідь запрограмованих компонентів, виконуючи програмування прямо в процесі виробництва. Це дозволяє знизити вартість виробництва і вносити зміни в програмовану частину пристрою без зупинки виробництва. Програмування по ISP-інтерфейсу відбувається по п'яти лініях зв'язку: MOSI, MISO, SCK, RESET і GND.

Існують два основних способи ISP:

- програматор працює з ПЗП і EEPROM мікроконтролера як із зовнішньою

пам’яттю, самостійно розміщуючи байти прошивки за потрібними адресами. Ядро МК при цьому не задіяно, а виводи переведені в високоімпедансний стан;

- використовується bootloader - невелика програма, записана, зазвичай, наприкінці пам’яті мікроконтролера. У цьому випадку при старті контролера bootloader перевіряє наявність заздалегідь визначених умов і якщо умови не співпадають, передає управління основній програмі. Якщо ж умови збігаються, bootloader переходить в режим програмування, готовий приймати дані через будь-який інтерфейс і розміщувати їх в пам’яті. При цьому МК програмує «сам себе».

UART (англ. universal asynchronous receiver/transmitter — універсальний асинхронний приймач/передавач) — тип асинхронного приймача-передавача, компонентів комп'ютерів та периферійних пристроїв, що передає дані між паралельною та послідовною формами. UART звичайно використовується спільно з іншими комунікаційними стандартами, такими як EIA RS-232 [3].

Актуальність теми

Нажаль, Україна не входить до складу країн, які виробляють пристрої для програмування мікроконтролерів, так як вигідніше закуповувати готові плати із таких країн як Китай чи Америка. Пристрої для програмування мікроконтролерів із Америки більш надійні, але і ціна їх досить висока, натомість, пристрої для програмування мікроконтролерів виготовлені у Китаї мають набагато меншу ціну, але разом із тим, і надійність їх невелика. Тож створенням та розробкою даних пристроїв займаються радіоаматори. В інтернеті є велика кількість схем пристроїв для програмування мікроконтролерів. Але недоліком більшості USB програматорів є те, що вони призначені для програмування мікроконтролерів конкретного виробника, та містять у своєму складі мікроконтролер від того ж самого виробника. Перевагою розроблюваного пристрою є те, що він дозволяє зручно і просто розширювати реалізацію можливих протоколів програмування без зміни апаратної частини або мікропрограми. Це дозволяє досить легко додавати нові можливості пристрою, так як проект на даний момент не закінчений та активно розвивається.

Актуальність теми полягає у створенні простого у використанні та зібраного на недорогих елементах пристрою для програмування

мікроконтролерів, який міг би програмувати мікроконтролери кількох різних виробників та різної архітектури.

Тому необхідність розробки пристрою з USB інтерфейсом для програмування 8 та 32 розрядних мікроконтролерів, включно із розробкою схеми, програмного забезпечення, експериментального дослідження параметрів пристрою та впровадження готового пристрою у виробництво є актуальним на даний час.

Мета і задачі дослідження

Метою роботи є створення універсального програматора для сучасних мікроконтролерів.

Об'єктом дослідження є універсальний програматор для сучасних мікроконтролерів виробництва Atmel та STMicroelectronics.

Предметом дослідження – протоколи програмування мікроконтролерів різних виробників та різної архітектури.

Для досягнення поставленої мети у магістерській кваліфікаційній роботі розв'язуються такі *задачі*:

- проаналізувати існуючі пристрої для програмування мікроконтролерів та обґрунтувати переваги пристрою, який розробляється по відношенню до існуючих;
- дослідити протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів;
- дослідити можливості мікроконтролера STM32F103C8;
- створення керуючої програми для мікроконтролера STM32F103C8 мовою C++;
- виконати експериментальну перевірку розробленого пристрою для програмування мікроконтролерів;
- здійснити метрологічну оцінку похибок програмування розробленого пристрою для програмування 8 та 32 розрядних мікроконтролерів.

Методи дослідження ґрунтуються на використанні:

- технічного опису протоколів передачі даних USB, UART та SPI;
- теорії розрахунку електричних кіл з використанням законів Ома для розрахунку номіналів елементів схеми;
- теорії ймовірності для оцінки похибок вимірювання.

Наукова новизна одержаних результатів

Наукова новизна роботи полягає в тому, що удосконалено структурну та принципову схему пристрою для програмування мікроконтролерів у порівнянні із існуючими аналогічними пристроями.

Практичне значення одержаних результатів

Практична цінність роботи полягає в тому, що:

1. Досліджено протоколи передачі даних USB, UART та SPI та дано їх докладний опис українською мовою із приведенням прикладів їх використання у реальних пристроях;
2. Вивчено особливості програмування мікроконтролерів AVR та STM32;
3. Розроблено закінчений пристрій з USB інтерфейсом для програмування 8 та 32 розрядних мікроконтролерів та керуючу програму на стороні персонального комп’ютера.

Результати роботи можуть використовуватись під час навчального процесу у дисциплінах, пов’язаних із мікропроцесорними пристроями: «Цифрові пристрої та мікропроцесорні системи», «Цифрова обробка аудіо та відео інформації».

Особистий внесок здобувача

Основні положення і результати магістерської кваліфікаційної роботи отримані автором самостійно.

1 СУЧАСНИЙ СТАН РОЗВИТКУ ПРИСТРОЇВ ДЛЯ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ

1.1 Програмування мікроконтролерів

Мікроконтролер – це виконана у вигляді мікросхеми спеціалізована мікропроцесорна система, що включає мікропроцесор, блоки пам'яті для збереження коду програм і даних, порти вводу-виводу і блоки зі спеціальними функціями (лічильники, компаратори, АЦП та інші) [1].

Мікроконтролер використовується для керування електронними пристроями. По суті, це – однокристальний комп'ютер, здатний виконувати прості завдання. Використання однієї мікросхеми значно знижує розміри, енергоспоживання і вартість пристроїв, побудованих на базі мікроконтролерів.

Мікроконтролер відрізняється від звичайної мікросхеми тим, що його після покупки потрібно попередньо запрограмувати, тобто записати у пам’ять мікроконтролера певну послідовність двійкових даних. Незапрограмований мікроконтролер у виробі марний. Він буде справно споживати струм, але програмно зациклиться в «бігу на місці», не виконуючи ніякої осмисленої роботи.

Щоб запрограмувати мікроконтролер потрібно виконати певну послідовність дій, в залежності від протоколу програмування. Хоча ці дії можна виконувати і в ручну, усе ж простіше скористатись спеціальним пристроєм для програмування. При масовому виробництві зазвичай використовують промислові пристрої для програмування мікроконтролерів (рисунок 1.1) та наведена в додатку Б, що мають самостійні органи управління, контролю, індикації.

Для дрібносерійного і одиничного виробництва раціональніше використовувати комп'ютерні програматори. В їх основі лежить проста формула: «Пристрій для програмування» = <Електричний адаптер> + <Комп'ютер> + <Програма>.

Розрізняють автономні (рисунок 1.2) танаведена в додатку Б, і внутрішньо схемні (рисунок 1.3) пристрої для програмування мікроконтролерів. Перші з них працюють в незалежному режимі. Мікроконтролер вручну встановлюється в панельку, програмується,

потім витягується з панельки і переноситься в готовий виріб. Другі – використовують спеціальний технологічний роз’єм, розташований на платі виробу. Через нього, не випаюючи мікроконтролер, можна в будь-який момент змінити керуючу програму.

Рисунок 1.1 – Автономні програматори

Рисунок 1.2 – Будова автономного комп’ютерного програматора

Усі сучасні мікроконтролери мають у своєму складі чотири основні типи пам’яті, а саме: пам’ять програм, пам’ять даних, енергонезалежна пам’ять даних та конфігураційні біти. Також до мікроконтролерів можна підключати зовнішню пам’ять. Робота із такою пам’яттю організовується за допомогою програмного коду, а використовувати її можна як завгодно.

Для зовнішнього програмування доступні три області пам'яті з чотирьох (рисунок 1.4), та наведена в додатку В. Інформація в них зберігається при вимиканні джерела живлення.

Пам’ять програм (FLASH) у мікроконтролері містить команди для керування процесором, а також може використовуватися для зберігання

константних табличних даних. FLASH пам’ять у 8 розрядних мікроконтролерах має 16-бітну організацію пам’яті та її розмір сягає від 1 до 256 КБайт, у залежності від моделі. FLASH пам’ять допускає перезапис до 10 тис. разів.

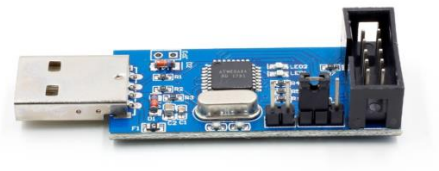


Рисунок 1.3 – Внутрішньо схемний комп’ютерний програматор

Пам’ять даних (SRAM) являє собою статичну оперативну пам'ять з довільним доступом (static random access memory). По суті це напівпровідникова оперативна пам'ять, в якій кожен двійковий розряд зберігається в схемі з додатним зворотним зв'язком, що не потребує регенерації, необхідної в динамічній пам'яті. Але зберігати дані без перезапису SRAM можливо тільки поки є живлення, тобто SRAM є енергозалежним типом пам'яті. Довільний доступ (RAM – random access memory) – можливість вибирати для запису/зчитування будь-який з бітів (частіше байтів, залежить від особливостей конструкції) [4]. EEPROM (енергонезалежна пам’ять даних) використовується для тривалого зберігання налаштувань, зібраних даних і т.п., тобто всіх даних, які слід зберегти до наступного запуску мікроконтролера. Кількість циклів перезапису сягає 100 тис. Читання даних у восьми бітних мікроконтролерах триває близько 4 тактів, однак запис відбувається дуже довго – близько 2-9 мсек. (це час виконання декількох тисяч команд) [5].

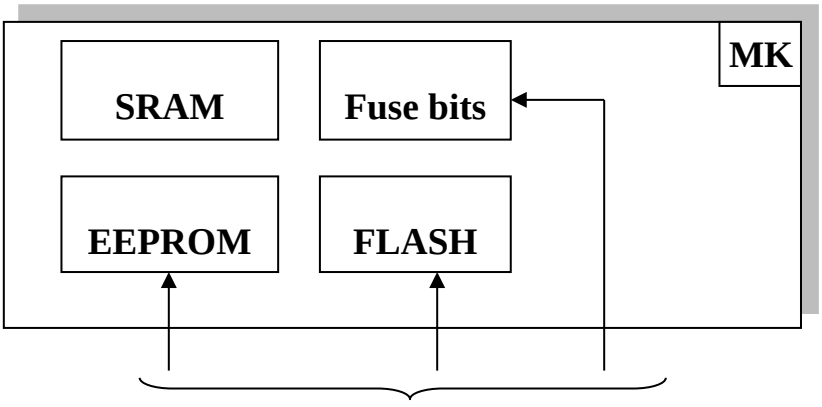


Рисунок 1.4 – Склад областей пам’яті мікроконтролера виробництва Atmel

Конфігураційні біти (fuse bits) наявні у 8 бітних мікроконтролерів використовуються для керування режимами роботи мікроконтролера, наприклад щоб мікроконтролер використовував зовнішній тактовий генератор (за замовчування мікроконтролери використовують внутрішній тактовий генератор невеликої частоти). Розшифровка призначень конфігураційних бітів наводиться в даташитах. Конфігураційними бітами можна також вважати і біти захисту LockBits. Вони дозволяють захистити коди програми від копіювання, перегляду та несанкціонованої зміни. Фізично конфігураційні біти виділяються в окрему область пам'яті, щоб їх було важко (або неможливо) змінити в процесі експлуатації, а також при хакерських атаках [6].

У мікроконтролерах STM32 інший спосіб організації пам’яті, коли для усіх блоків пам’яті використовується єдиний адресний простір. Також у даних мікроконтролерах відсутня EEPROM пам’ять, тому в якості енергонезалежної пам’яті для налаштувань і зібраних даних використовується flash пам’ять. На відміну від мікроконтролерів AVR, у мікроконтролерах STM32 режими роботи налаштовуються спеціальними внутрішніми регістрами в процесі роботи, що дає певні переваги (наприклад безвідмовна робота у разі, якщо у тактовому генераторі стався збій). На рисунку 1.5 показано області пам’яті мікроконтролера STM32F103C8T6 [7].

Із рисунка 1.5 видно, що всі апаратні складові мікроконтролера STM32F103C8T6 знаходяться в одному адресному просторі.

У різних мікроконтролерних платформах при програмуванні застосовуються свої фірмові засоби, технології і навіть назви, а саме:

ISP (In-System Programming), ICSP (In-Circuit Serial Programming), ISSP (In-System Serial Programming) - низьковольтне послідовне програмування (рисунок 1.6, а);

- JTAG (Joint Test Action Group) - низьковольтне шлейфне програмування за стандартом IEEE 1149.1 з можливістю налагодження (рисунок 1.6, б);
- debugWire - низьковольтне однопровідне програмування з можливістю налагодження (рисунок 1.6, в);
- BootLoader - низьковольтне програмування, при якому в ПЗП спочатку заносяться коди завантажувача на іншому програматорі (рисунок 1.6, г,д);

– HVSP (High Voltage Serial Programming) - високовольтне послідовне програмування (рисунок 1.6, е);
– HVPP (High Voltage Parallel Programming) - високовольтне паралельне програмування (рисунок 1.6, ж), та наведена в додатку Д.

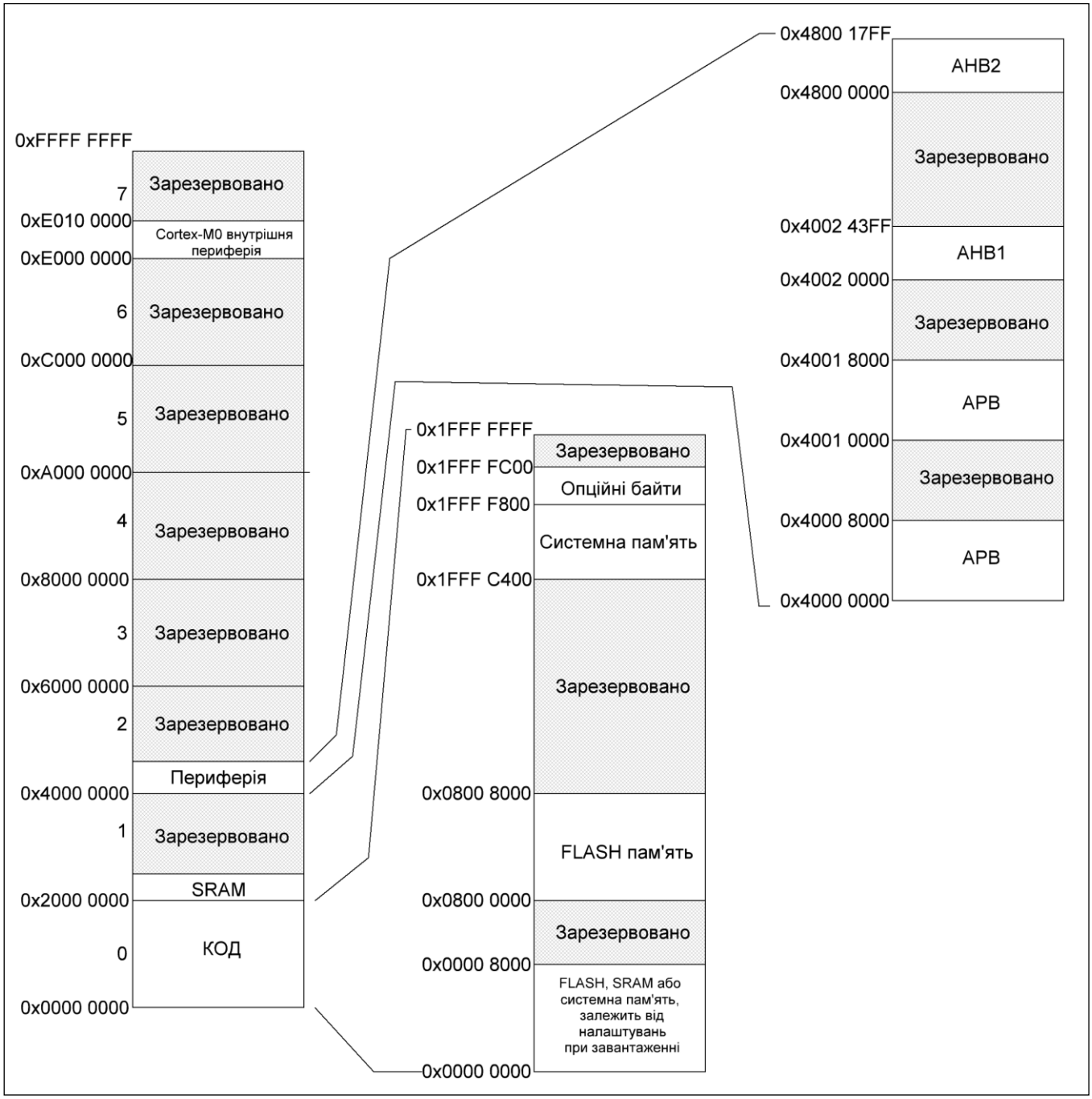


Рисунок 1.5 – Області пам’яті мікроконтролера STM32F103C8T6

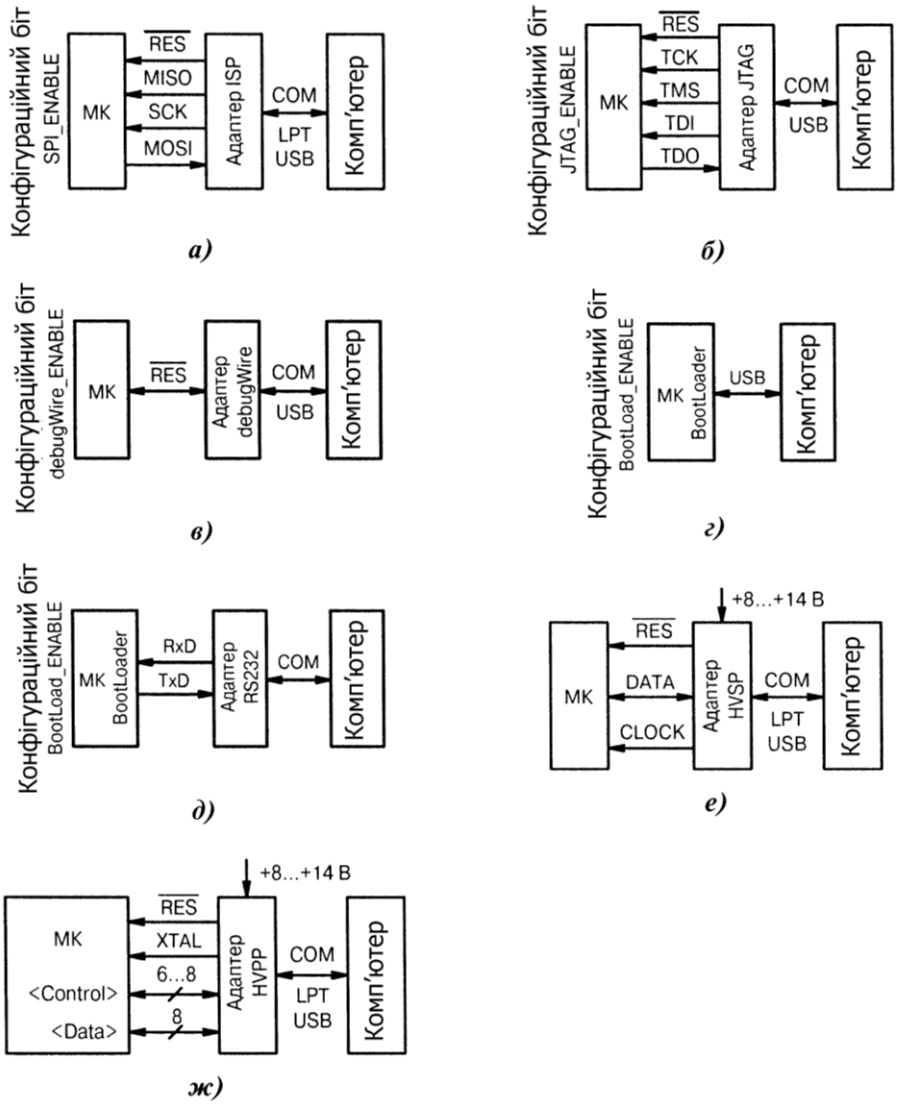


Рисунок 1.6 – Різновиди систем програмування

Розподіл на низько та високовольтне програмування стався історично. Перші мікроконтролери програмувались підвищеною напругою +8…14/+24…27 В, і потребували вилучення мікросхеми із панелі на платі пристрою. По мірі вдосконалення технології інженери-конструктори навчились вмонтовувати підвищуючі DC/DC – перетворювачі напруги прямо на підкладку кристала мікросхеми. Програмування стало низьковольтним від +5 В і внутрішньо схемним, оскільки тепер не потрібно було вилучати мікроконтролер із панелі.

Розподіл на паралельне і послідовне програмування також має історичні корні. На початку мікроконтролерної ери споживачі старались зробити виробництво дешевшим, замовляючи на заводі-виробнику великі партії мікроконтролер із певною версією програми. Промислове виробництво

в заводських умовах мало бути швидким, а високу швидкість передачі даних забезпечує саме багато провідний паралельний режим.

З широким впровадженням низьковольтних адаптерів виявилося, що вигідніше здійснювати програмування невеликих партій мікроконтролерів у себе на фірмі. Однак прощатися з паралельним режимом ще рано, тому що через нього в деяких мікроконтролерах, наприклад Atmel ATmega, «прошиваються» важливі конфігураційні біти.

Наступною помітною сходинкою стало об’єднання процесу програмування із налагодженням програми. Технології JTAG та debugWire дають програмісту повний контроль над апаратною начинкою мікроконтролера, дозволяють зупиняти роботу програми в потрібних точках, перевіряти стан портів і регістрів в режимі реального часу. Як правило, такі налагоджувачі мають вигляд заводських фірмових модулів зі своїм програмним забезпеченням. Аматорські розробки теж існують, але вони вимагають високої кваліфікації виконавця.

У мікроконтролерах із ядром ARM, до яких відносяться мікроконтролери серії STM32, з’явився свій підвид інтерфейсу JTAG, а саме SWD.

Послідовне провідне налагодження (анг. Serial Wire Debug, SWD) дозволяє використовувати режим налагодження у мікроконтролерах із обмеженою кількістю виводів та в складних інтегральних схемах, де обмеження виводів мікроконтролера є критичним і може бути визначальним фактором у розрахунку ціни пристрою.

SWD замінює стандартний 5-контактний JTAG забезпечуючи всі звичайні функції налагодження JTAG і плюс доступ до системної пам'яті в режимі реального часу без зупинки процесора і не вимагаючи будь-якого коду цільового мікроконтролера. SWD використовує стандартний двонаправлений провідний протокол ARM, визначений у інтерфейсі налагодження ARM v5, для передачі даних між налагоджувачем та цільовою системою високоефективним і стандартним способом.

Переваги SWD:

- Потрібно тільки 2 виводи – вагоме значення для мікроконтролерів із малою кількістю виводів;
- Забезпечує налагоджування сумісне з контролерами, підтримуючими JTAG;

- Висока швидкість передачі даних – 4 Мбайт/с на частоті 50 МГц;
- Низький рівень споживання енергії – немає необхідності в додаткових виводах живлення або заземлення;
- Надійність – вбудоване виявлення помилок;
- Безпечність – захист від збоїв на виводах, коли інструменти налагодження не підключені.

Іще один цікавий напрямок - це віддалене програмування через програму-завантажувач (BootLoader). Попередньо в мікроконтролер «прошивається» коротка завантажувальна програма, яка за певних умов може перепрограмувати свою власну флеш-пам’ять та комірку EEPROM. Коди нової прошивки передаються в мікроконтролер з комп’ютера по одному з доступних каналів зв'язку: USB, COM, I^2C, IrDA і т.д. Якщо комп'ютер програматора буде підключений до Інтернету, то змінити прошивку можна з іншого віддаленого комп'ютера, знаходячись в будь-якій точці Земної кулі [6].

Мікросхем, які підтримують всі відомі способи «послідовно-паралельного» програмування, не існує. Зазвичай доступні один, два або три різних способи. Найважливіше, що всі сучасні мікроконтролери можуть програмуватись внутрішньо схемно. Це дозволяє використовувати прості і надійні програматори, що працюють за протоколом SPI (Serial Programming Interface).

1.2 Інтерфейс SPI

SPI (англ. Serial Peripheral Interface, SPI bus – послідовний периферійний інтерфейс, шина SPI) – послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для забезпечення простого і недорогого сполучення мікроконтролерів і периферії [8].

Інтерфейс SPI – це один з найпопулярніших на сьогоднішній день послідовних інтерфейсів. Він був розроблений фірмою Motorola і дуже швидко завоював популярність завдяки своїй винятковій простоті і високій швидкості. При цьому, SPI, напевно, не можна назвати повною мірою інтерфейсом, скоріше це просто принцип зв’язку, оскільки все, що мається на увазі під SPI - це логіка передачі даних між двома пристроями ("Ведучий" - "Ведений") [9].

На відміну від стандартного послідовного порту, SPI є синхронним інтерфейсом, в якому будь-яка передача синхронізована із загальним тактовим сигналом, що генерується ведучим пристроєм (процесором). Приймаюча (ведена) периферія синхронізує отримання бітової послідовності з тактовим сигналом. До одного послідовного периферійного інтерфейсу ведучого пристрою-мікросхеми може приєднуватися кілька мікросхем. Ведучий пристрій вибирає ведений для передачі, активуючи сигнал «вибір кристалу» (англ. chip select) на веденій мікросхемі. Периферія, не обрана процесором, не бере участі в передачі по SPI.

У SPI використовуються чотири цифрових сигнали:

- MOSI – вихід ведучого, вхід веденого (англ. Master Out Slave In). Служить для передачі даних від ведучого пристрою веденого;
- MISO – вхід ведучого, вихід веденого (англ. Master In Slave Out). Служить для передачі даних від веденого пристрою ведучому;
- SCLK – послідовний тактовий сигнал (англ. Serial Clock). Служить для передачі тактового сигналу для ведених пристроїв;
- CS або SS – вибір мікросхеми, вибір веденого (англ. Chip Select, Slave Select).

Передача здійснюється пакетами (рисунок 1.7). Довжина пакету як правило становить 1 байт (8 біт). Ведучий пристрій ініціює цикл зв'язку установкою низького рівня на виводі вибору підлеглого пристрою ($\overline{SS}$) того пристрою, з яким необхідно встановити з'єднання. При низькому рівні сигналу $\overline{SS}$:

- схемотехніка веденого пристрою знаходиться в активному стані;
- вивід MISO переводиться в режим «вихід»;
- тактовий сигнал SCK від ведучого пристрою сприймається веденим і викликає зчитування на вході MOSI значень бітів переданих від ведучого і зсув регістра веденого пристрою.

Дані, які підлягають передачі ведучий і ведений пристрої поміщають в регістри зсуву. Після цього, ведучий пристрій починає генерувати імпульси синхронізації на лінії SCLK, що призводить до взаємного обміну даними. Передача даних здійснюється біт за бітом від ведучого по лінії MOSI і від веденого по лінії MISO. Передача здійснюється як правило починаючи зі старших бітів, але деякі виробники допускають зміну порядку передачі бітів програмними методами. Після передачі кожного пакета даних, ведучий

пристрій, з метою синхронізації веденого пристрою, може перевести лінію SS у високий стан.

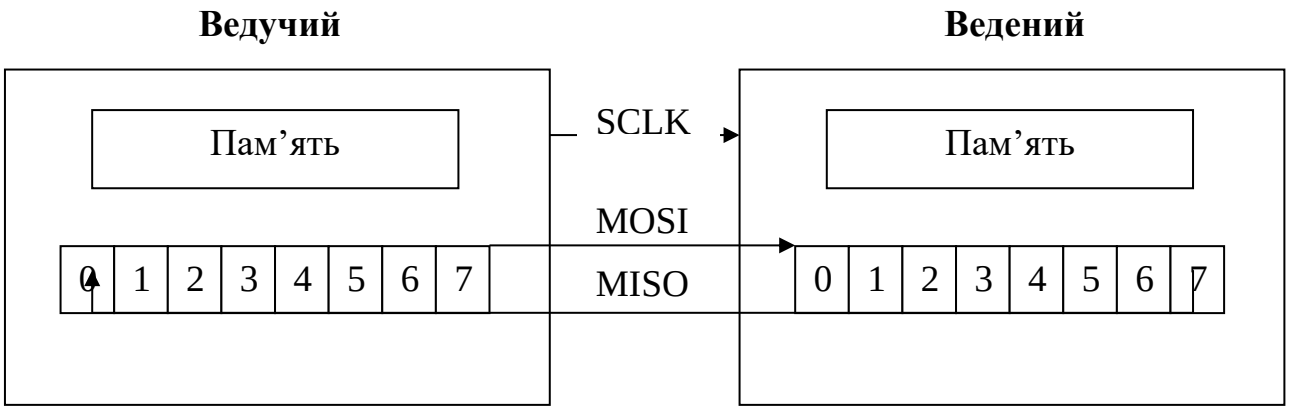


Рисунок 1.7 – Типова структура зв’язків і ліній інтерфейсу SPI

Можливі чотири комбінації фази (CPHA) і полярності (CPOL) сигналу SCLK по відношенню до сигналів даних. Режими роботи визначаються комбінацією бітів CPHA і CPOL:

CPOL = 0 – сигнал синхронізації починається з низького рівня;

CPOL = 1 – сигнал синхронізації починається з високого рівня;

CPHA = 0 – вибірка даних проводиться по передньому фронту сигналу синхронізації;

CPHA = 1 – вибірка даних проводиться по задньому фронту сигналу синхронізації.

Позначення режимів роботи інтерфейсу SPI прийнято наступне:

– режим 0 (CPOL = 0, CPHA = 0);
– режим 1 (CPOL = 0, CPHA = 1);
– режим 2 (CPOL = 1, CPHA = 0);
– режим 3 (CPOL = 1, CPHA = 1).

Мікроконтролери фірми Atmel під час програмування використовують режим 0. Часові діаграми представлені на рисунку 1.8.

1.3 Інтерфейс UART

Універсальний асинхронний приймач-передавач (УАПП, англ. Universal Asynchronous Receiver-Transmitter, UART) - вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими

пристроями. Перетворює передані дані в послідовний вигляд так, щоб було можливо передати їх по цифровій лінії іншому аналогічному пристрою. Метод перетворення добре стандартизований і широко застосовується в комп'ютерній техніці (особливо у вбудованих пристроях і системах на кристалі (SoC)).

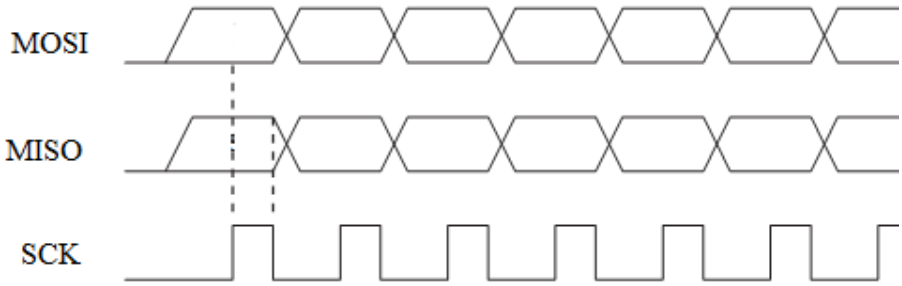


Рисунок 1.8 – Часові діаграми роботи інтерфейсу SPI при програмуванні мікроконтролерів фірми Atmel

UART – це логічна схема, з одного боку підключена до шини обчислювального пристрою, а з іншого має два або більше виводів для зовнішнього з’єднання.

UART може бути окремою мікросхемою (наприклад, Intel I8251, I8250) або бути частиною великої інтегральної схеми (наприклад, мікроконтролера). Використовується для передачі даних через послідовний порт комп'ютера, часто вбудовується в мікроконтролери [3].

Передача даних в UART здійснюється по одному біту в рівні проміжки часу. Цей часовий проміжок визначається заданою швидкістю UART і для конкретного з’єднання вказується в бодах (що в даному випадку відповідає бітам в секунду). Існує загальноприйнятий ряд стандартних швидкостей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод. Швидкість (S, бод) і тривалість біта (T, секунд) пов’язані співвідношенням T = 1/S. Швидкість в бодах іноді називають сленговим словом бітрейт.

Крім інформаційного потоку, UART автоматично вставляє в потік синхронізуючі мітки, так звані стартовий і стоповий біти. При прийомі ці зайві біти видаляються з потоку. Зазвичай стартовий і стоповий біти оточують один байт інформації (8 біт), проте зустрічаються реалізації UART, які дозволяють передавати по 5, 6, 7, 8 або 9 біт. Оточені стартовим і

стоповим бітом біти є мінімальною посилкою. Деякі реалізації UART дозволяють вставляти два стопових біта при передачі для зменшення ймовірності розсинхронізації приймача і передавача при щільному трафіку. Приймач ігнорує другий стоповий біт, сприймаючи його як коротку паузу на лінії [3].

Прийнято угоду, що пасивним (за відсутності потоку даних) станом входу і виходу UART є логічна 1. Стартовий біт завжди логічний 0, тому приймач UART чекає перепаду з 1 в 0 і відраховує від нього часовий проміжок в половину тривалості біта (середина передачі стартового біта). Якщо в цей момент на вході все ще 0, то запускається процес прийому мінімальної посилки. Для цього приймач відраховує 9 бітових тривалостей поспіль (для 8-бітних даних) і в кожен момент фіксує стан входу. Перші 8 значень є прийнятими даними, останнє значення перевірочне (стоп-біт). Значення стоп-біта завжди 1, якщо реально прийняте значення інше, UART фіксує помилку. Приклад потоку байтів UART представлено на рисунку 1.9.

Для формування часових інтервалів передавальний і приймальний UART мають джерело точного часу (тактування). Точність цього джерела повинна бути такою, щоб сума похибок (приймача і передавача) установки часового інтервалу від початку стартового імпульсу до середини стопового імпульсу не перевищувала половини (а краще хоча б чверті) бітового інтервалу [10].

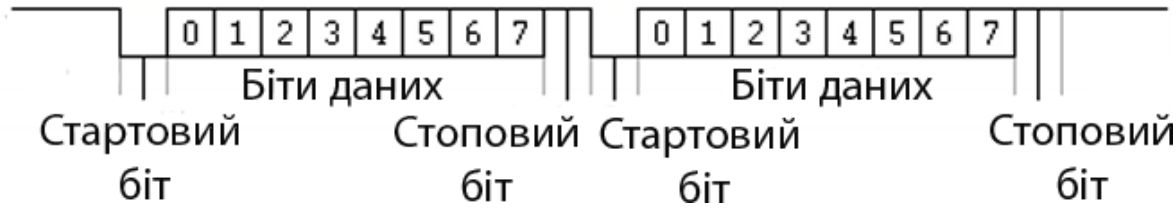


Рисунок 1.9 – Потік байтів UART

Оскільки синхронізуючі біти займають частину бітового потоку, то результуюча пропускна здатність UART не дорівнює швидкості з'єднання. Наприклад, для 8-бітних посилок формату 8-N-1 синхронізуючі біти займають 20% потоку, що для фізичної швидкості 115 200 бод дає бітову швидкість даних 92 160 біт / с або 11 520 байт / с.

Багато реалізацій UART мають можливість автоматично контролювати цілісність даних методом контролю бітової парності. Якщо ця функція увімкнена, останній біт даних в мінімальної посилці («біт парності») контролюється логікою UART і містить інформацію про парність кількості одиничних біт в цій мінімальній посилці. Розрізняють контроль на парність (англ. Even parity), коли сума кількості одиничних біт в посилці є парним числом, і контроль на непарність (англ. Odd parity), коли ця сума непарна. При прийомі такої посилки UART може автоматично контролювати біт парності і виставляти відповідні ознаки правильного або неправильного прийому.

1.4 Типи пристроїв для програмування мікроконтролерів

Спершу розглянемо типи пристроїв для програмування мікроконтролерів, які використовуються для програмування восьми бітних мікроконтролерів. Частину із них можна застосовувати і для програмування 32 бітних мікроконтролерів завдяки схемам узгодження та спеціальному програмному забезпеченні.

З електричної точки зору виводи, що беруть участь в програмуванні, є звичайними входами/виходами КМОП-елементів.

Серед програматорів виділяються наступні групи:

- USB - найшвидші і перспективні, але відносно складні;
- LPT - найпростіші у виготовленні, але не завжди стабільно працюючі;
- COM - популярні в аматорському середовищі, але мають низьку швидкість програмування [6].

Пристрої для програмування мікроконтролерів через LPT порт є найпростішими у виготовленні. Прикладом такого пристрою можна привести програматор STK200 (рисунок 1.10).

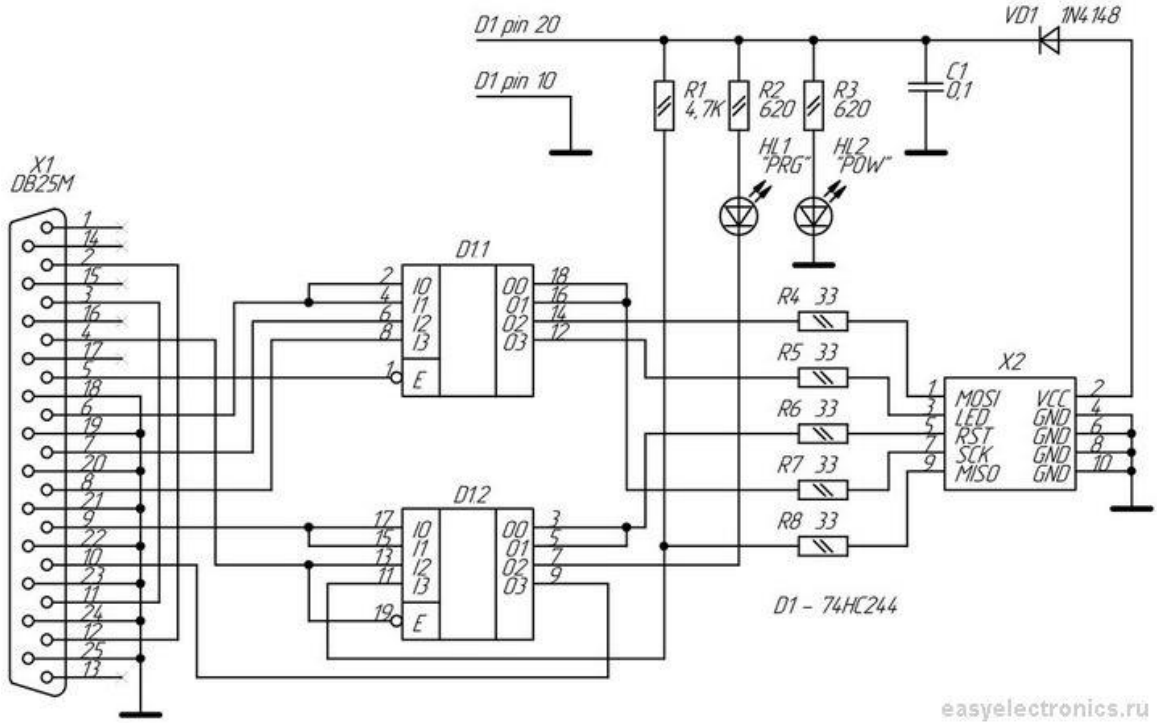


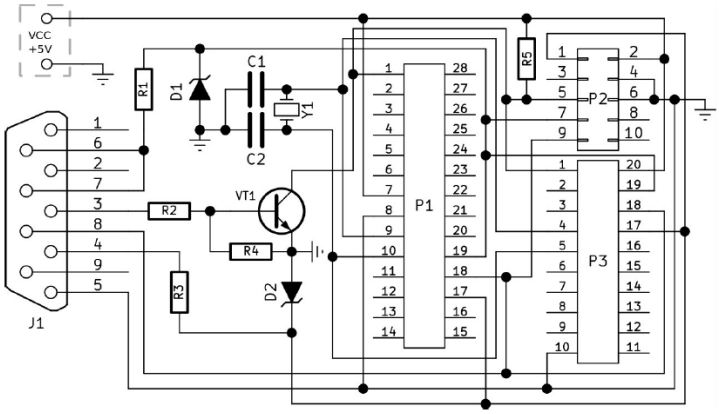
Рисунок 1.10 – Схема програматора STK200

Як видно із схеми, щоб виготовити даний програматор, потрібно лише LPT порт та 4 резистора. Живлення мікроконтролера, який програмується, здійснюється від комп’ютера. Перевагою такого пристрою для програмування мікроконтролерів є його простота і велика кількість керуючих програм, із якими він працює. До недоліків можна віднести низьку стабільність роботи, необхідність пошуку варіантів для підключення живлення. Головним недоліком є те, що порт LPT на сьогодні є застарілим та його можна зустріти у дедалі меншій кількості комп’ютерів [11].

Ще одним нескладним, в плані виготовлення, є пристрій для програмування мікроконтролерів через COM порт. За умови використання альтернативного режиму COM порту Bitbang, відпадає необхідність у перетворенні інтерфейсу RS232 COM порту в SPI, необхідний для програмування. Залишається тільки привести рівні сигналів COM порту (-12В, + 12В) до необхідних (0, + 5В). Цього можна досягти за допомогою простою схеми, приведеної рисунку 1.11.

Такий пристрій для програмування мікроконтролерів називається програматором К-156. Схема ця в набагато надійніша, а також легко перетворюється в програматор різних EEPROM мікросхем (типу 24Cxx).

Так як режим Bit bang нестандартний для COM порту комп'ютера, то можливі збої в роботі. Якщо використовувати програму Algorithm Builder, то цей програматор можна використовувати для внутрішньосхемного налагодження кристала (програмний JTAG) [12]. До головної переваги даного програматора можна віднести його простоту та високу надійність, порівняно із STK200. До недоліків можна віднести те, що COM порти є застарілими, та усе рідше трапляються на комп’ютерах, а на сучасних ноутбуках їх взагалі немає. Також недоліком є те, що COM порти у комп’ютера можуть легко вийти з ладу, якщо пристрій для програмування підключати до комп’ютера, який увімкнений.



а)

б)

Рисунок 1.11 – Схема (а) та зовнішній вигляд (б) програматора К-156

Враховуючи що USB порт є сьогодні найпоширенішим а швидкість передачі даних сягає 10 Гбіт/с, пристрої для програмування мікроконтролерів через USB порт є найбільш перспективними. Як приклад, можна привести USB програматор AVR910, схема якого представлена на рисунку 1.12 [13].

Як видно із схеми, даний пристрій є більш складним у виготовленні, порівняно із COM та LPT програматорами. До переваг даного пристрою можна віднести те, що він також підтримується великою кількістю керуючих

програм, має порівняно високу швидкість програмування кристалів, високу надійність у роботі та при роботі із ним схема, яку потрібно запрограмувати може живитись від USB порту комп'ютера. До недоліків можна віднести більш складну схему.

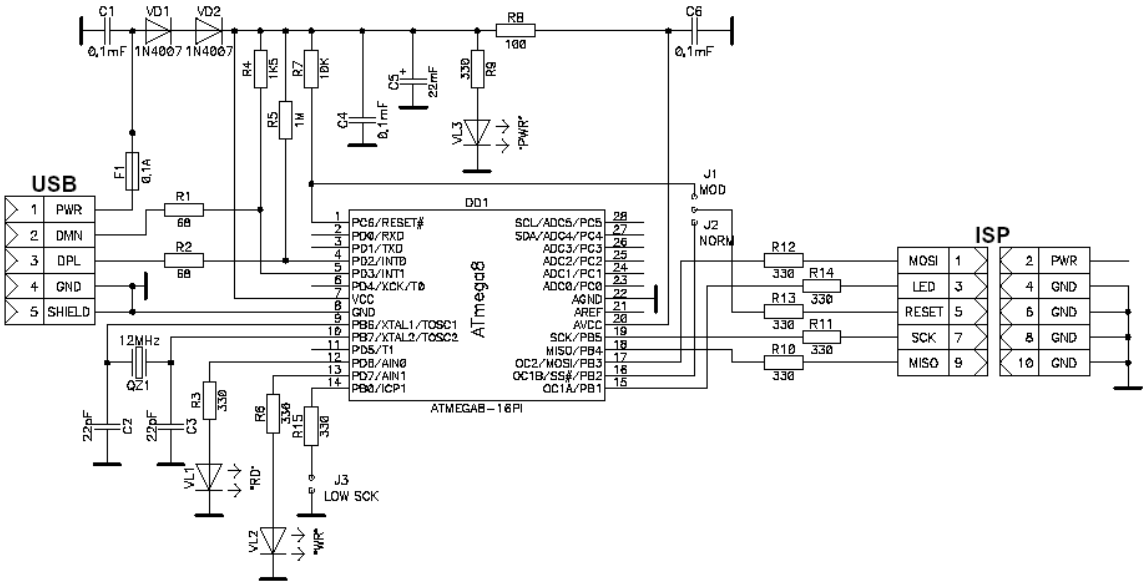


Рисунок 1.12 – Схема програматора AVR910

В якості іще одного USB програматора можна представити програматор для мікроконтролерів PIC GTP-USB (Grabador TodoPic-USB). Схему даного програматора показано на рисунку 1.13.

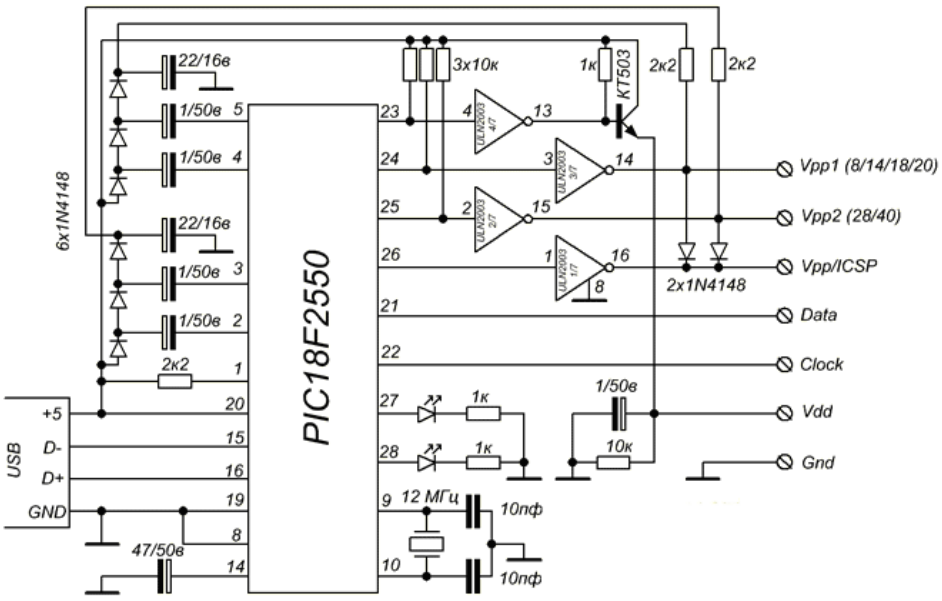


Рисунок 1.13 – Схема програматора GTP-USB

Як видно із рисунку 1.13 схема програматора іще більше ускладнилась завдяки додатковим інверторам, конденсаторам та транзистору. Працює даний програматор за допомогою програми WinPic800 [14].

До переваг даного програматора можна віднести порівняно високу швидкість програмування мікроконтролерів та надійність даного програматора. До недоліків – більш складну схему та необхідність від'єднувати мікроконтролер, який потрібно запрограмувати, від схеми, в якій він знаходиться.

Прикладом найпростішого пристрою для програмування мікроконтролерів через USB порт є пристрій на мікросхемі FT232RL. Ця мікросхема дозволяє без зайвих операцій створити USB програматор так як у ній на апаратній частині реалізовано протокол USB та написані виробником драйвера. Також ця мікросхема підтримує режим BitBang який дозволяє програмно із комп'ютера керувати виводами мікросхеми. Саме на цій можливості мікросхеми побудовано пристрій для програмування мікроконтролерів, схема якого зображена на рисунку 1.14 [15].

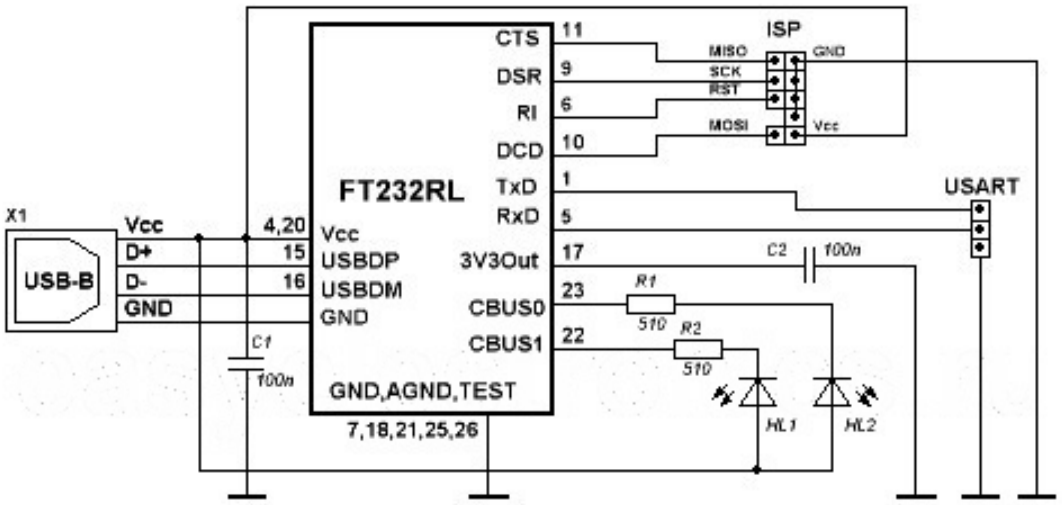


Рисунок 1.14 – Схема пристрій для програмування мікроконтролерів на мікросхемі FT232RL

Як можна побачити із рисунка 1.14, цей програматор є найпростішим серед USB програматорів. До його переваг відноситься дуже проста схема ввімкнення, висока надійність у роботі. Для роботи із даним програматором в інтернеті можна знайти спеціально налаштовану програму AVRDude.

Головним недоліком цієї програми є те, що вона призначена для роботи у командному рядку, і кожного разу під час програмування нового мікроконтролера потрібно у командний рядок вводити спеціальні команди. Це сповільнює роботу та дає високу імовірність помилки при введенні команди. Також недоліком є те, що після програмування за допомогою AVRDude програматор не відпускає лінію RESET, через що для запуску мікроконтролера потрібно від'єднувати контролер від програматора.

Основним недоліком цього пристрою для програмування мікроконтролерів є те, що керуюча мікросхема на сьогоднішній день суттєво піднялась в ціні, її вартість становить близько 150 гривень. Якщо використовувати дану мікросхему у пристрої, то його ціна зросте приблизно на 30%.

Розглянемо ще один програма тор, який призначений для програмування різних мікроконтролерів, а також прошивання мікросхем SPI Flash і I2C Eeprom і т.п. Цей програматор є розвитком програматора, описаного раніше. Робота програматора описана вище і сильно не змінилася, додалися лише підтримувані мікросхеми, а також функції читання FLASH і EEPROM. Також даний програматор може працювати як імітатор і аналізатор послідовних протоколів [6].

Програматор побудований на мікросхемі FT232R (Рис.1.15), яка дозволяє управляти пінами в режимі бітбенг. Для того, щоб прошивання займало якомога менший час, реалізована пакетна передача даних. Програматор не має вихідного буфера, тому що FT232R дозволяє живити свої порти зовнішнім напругою і переводити їх в стан Z (3-третій стан) після програмування, тому він не конфліктує з периферією контролера.

Програматор має два роз'єми для внутрисхемного програмування: AVR-ISP і PIC-ISP. З назви вже зрозуміло призначення роз'ємів. Також програматор має панельку для програмування SPI Flash і I2C Eeprom. З одного боку плати - під мікросхеми в DIP-8, з іншого - SO-8 (на прищіпку). Також є 2 джампера: один для зміни режимів живлення програматора і програмованої мікросхеми, інший перемикає режим виходу 12 В (програмування PIC контролерів або робота в режимі аналізатора).

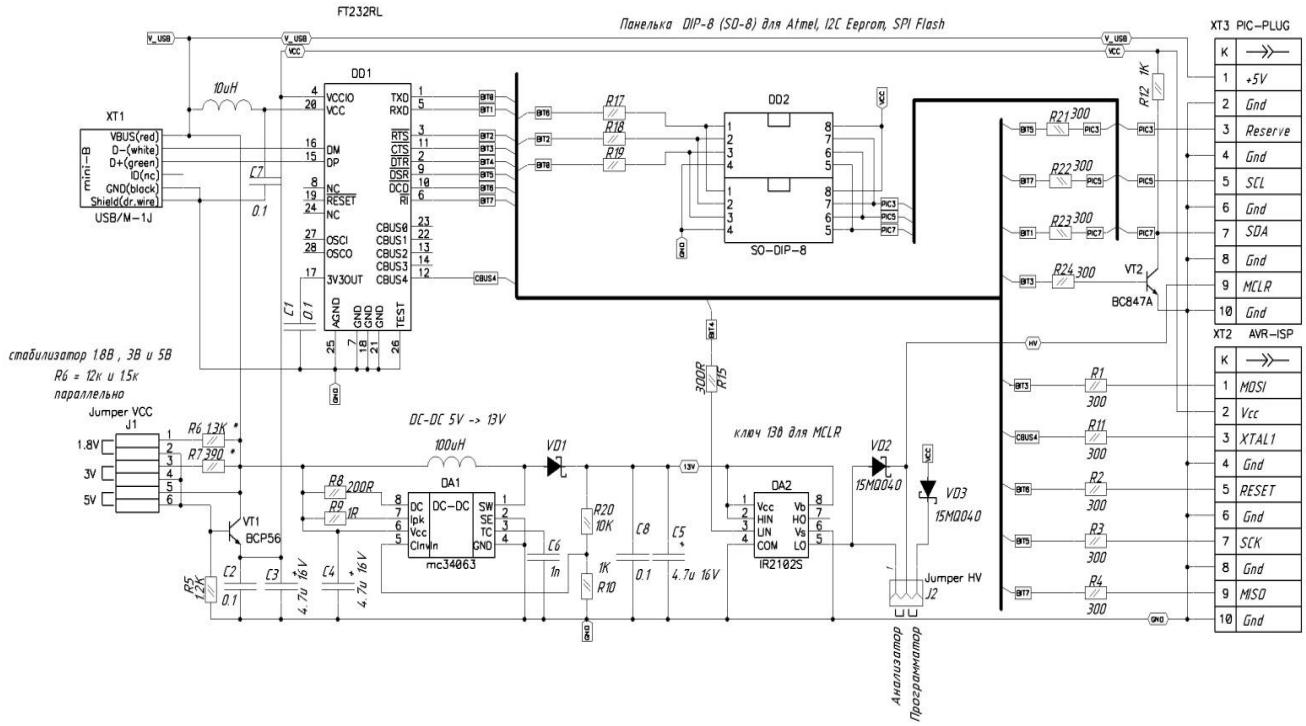


Рисунок 1.15 – Схема пристрій для програмування мікроконтролерів на мікросхемі FT232R

Програмна оболонка для такого програма тора представлена на рис.1.16. Інтерфейс програматора інтуїтивно зрозумілий. Є табло з балкою операцій (праворуч) і табло з закладками для програмування різних чіпів (зліва). Фьюз підписані згідно даташіта, галочка означає '1 ".

Основним недоліком більшості приведених пристроїв для програмування мікроконтролерів є те, що вони призначені для якоїсь конкретної серії мікроконтролерів, або для мікроконтролерів певного виробника.

Для програмування 32 бітних мікроконтролерів STM32 можна використовувати програматор ST-LINK. Схема даного програматора показана на рисунку 1.17. Програматор ST-Link V2 у формфакторі флешки сумісний з контролерами лінійок STM32 і STM8. Підтримуються інтерфейси SWD, SWIM. ST-LINK дозволяє по кроках виконувати програму мікроконтролера і стежити за значеннями всіх регістрів, що сильно полегшує роботу з проектами. Програматором можна записати в пам'ять

мікроконтролера і сам завантажувач, за допомогою якого МК прошивається через USB/UART, і таким чином відновити контролер.

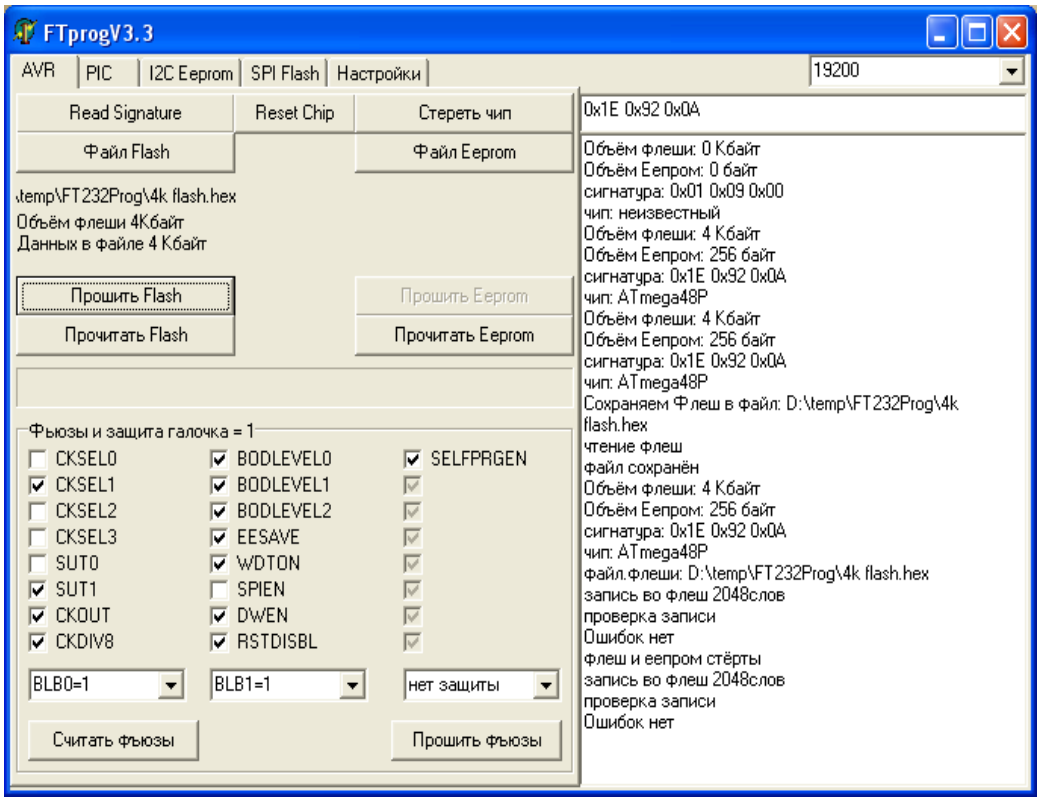
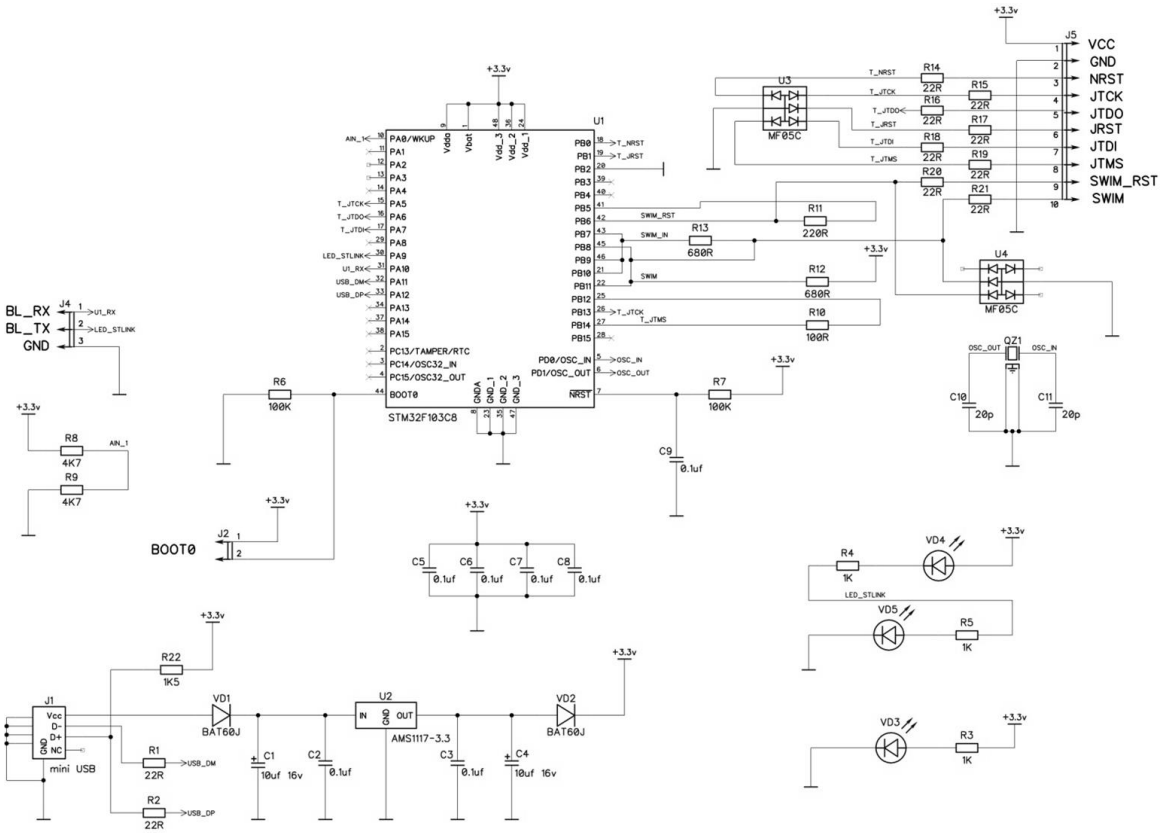


Рисунок.1.16 – Програмна оболонка

STLINK V2 - це внутрішньо-схемний програматор і відладчик для мікроконтролерів серії STM8 і STM32. За допомогою ST-LINK MINI можна програмувати і виконувати налагодження по інтерфейсів SWIM (для мікроконтролерів STM8), SWD і JTAG (для мікроконтролерів STM32). Всі комунікаційні інтерфейси програм (SWIM, SWD, JTAG) доступні для використання і виведені на 10-ти контактний роз'єм програматора.

а)



б)

Рисунок 1.17 – Зовнішній вигляд а), схема б) програматора ST-LINK

Як видно на рисунку 1.17 схема даного програматора складніша за USB програматори для 8 бітних мікроконтролерів. Головною перевагою даної схеми є можливість роботи із протоколом SWD. Недоліком є необхідність запрограмувати керуючий мікроконтролер та складність виготовлення друкованої плати в домашніх умовах, так як мінімальна товщина провідників складає 0,3 мм. Також недоліком є те, що керуюча програма мікроконтролера не поставляється у відкритому вигляді, що робить неможливим модифікацію програми для підтримки протоколу програмування ISP.

Найпростіший спосіб запрограмувати мікроконтролер STM32 це використати вбудований bootloader та протокол UART. Такий тип програмування підтримується усією лінійкою мікроконтролерів STM32 та легкий у реалізації.

Для того, щоб запрограмувати мікроконтролер STM32 необхідно на вивід BOOT0 подати напругу живлення (3,6 В), вивід BOOT1 приєднати до корпусу та перезавантажити мікроконтролер. Після того, як мікроконтролер перейде в режим програмування вивід BOOT0 можна від’єднати від напруги живлення. Схема приєднання до UART показана на рисунку 1.18 [16].

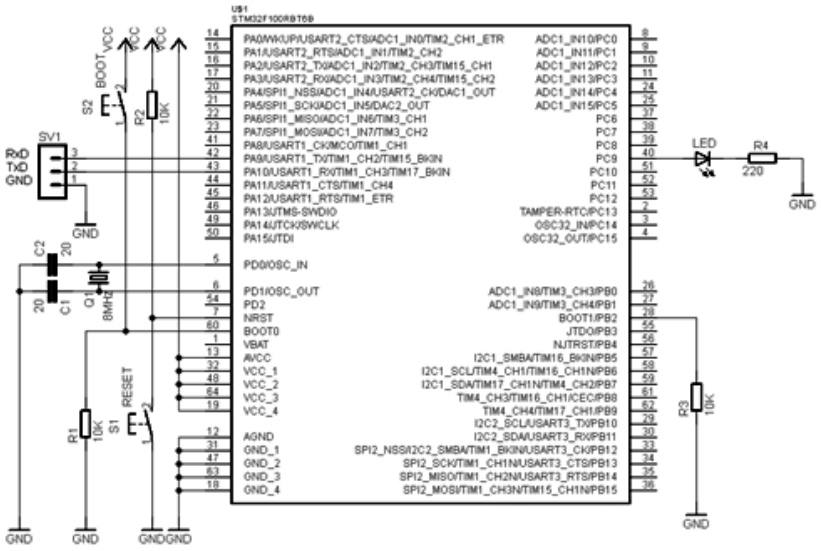


Рисунок 1.18 – Схема приєднання мікроконтролерів STM32 до UART

Для програмування можна скористатись стандартною програмою Flash Loader Demonstrator, яку безкоштовно постачає виробник. Процес завантаження програми до мікроконтролерів STM32 за допомогою даної програми простий та інтуїтивно зрозумілий.

1.5 Висновки до розділу

Проведено аналіз існуючих пристрої для програмування мікроконтролерів та обґрунтовано недоліки та переваги пристроїв для програмування МК. Поставлено задачі, які необхідно розв’язати при розробці програма тора для мікроконтролерів по відношенню до існуючих. Досліджено протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ УНІВЕРСАЛЬНОГО ПРОГРАМАТОРА ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

2.1 Обґрунтування вибору керуючого мікроконтролера

Пристрій з USB інтерфейсом для програмування 8 та 32 розрядних мікроконтролерів, який розроблено у магістерській кваліфікаційній роботі побудований на мікроконтролері STM32F042K6T6.

STM32F042 представник сімейства бюджетних мікроконтролерів STM32F0 на 32-бітному ядрі Cortex-M0 від ST Microelectronics. У даному мікроконтролері присутня периферія USB, CAN, HDMI, а також збільшилася кількість стандартної периферії - USART, SPI, I²C і таймерів.

USB інтерфейс у мікроконтролері STM32F042K6T6 може працювати за рахунок внутрішнього генератора 48 МГц з автоматичним налаштуванням. Синхронізація даного генератора відбувається автоматично по потоку даних на шині USB, що дозволяє обходитися без додаткового зовнішнього кварцового генератора. Така технологія позитивно позначається на надійності і вартості готового пристрою.

Хоча мікроконтролер STM32F042K6T6 може використовувати внутрішній генератор 48 МГц з автоматичним налаштуванням для тактування внутрішньої периферії, для більш стабільної та надійної роботи усе ж прийнято рішення використати зовнішній кварцовий генератор частотою 8 МГц та внутрішніми помножувачами розігнати частоту до 48 МГц.

Основні технічні характеристики мікроконтролера STM32F042K6T6:

- Вид монтажу: SMD;
- Тип корпусу: LQFP-32;
- Ядро: ARM Cortex M0;
- Ширина шини даних: 32 біти;
- Максимальна тактова частота: 48 МГц;
- Розмір програмної пам'яті: 32 кБ;
- Розмір RAM даних: 6 кБ;
- Розрядність АЦП: 12 бітів;
- Робоча напруга живлення: від 2 В до 3,6 В;
- Максимальна робоча температура: +85 ^{O}C;

- Серія процесора: STM32F0;
- Напруга живлення для аналогових пристроїв: від 2 В до 3,6 В;
- Тип ОЗП: SRAM;
- Вхідна/вихідна напруга: від 1,65 В до 3,6 В;
- Тип інтерфейсу: CAN, CEC, I2C, SPI, USART, USB;
- Мінімальна робоча температура: - 40 ^{O}C;
- Кількість каналів АЦП: 13;
- Кількість ліній входів/виходів: 26;
- Кількість таймерів/лічильників: 5 x 16 біт, 1 x 32 біти [16].

Даний мікроконтролер на сьогоднішній день є найдешевшим, у порівнянні із мікроконтролерами інших виробників (серед тих, які містять у своєму складі апаратно реалізований інтерфейс USB) та коштує близько 70 гривень. Також даний мікроконтролер у своєму складі має апаратно реалізовані інтерфейси передачі даних SPI та USART, які використовуються у розробленому пристрої. Використання апаратно реалізованого інтерфейсу USB обумовлено складною програмною реалізацією даного інтерфейсу та ймовірністю його хибної роботи.

Усі приведені фактори дозволяють спростити розробку пристрою використовуючи стандартні можливості мікроконтролера заміть того, щоб реалізовувати передачу даних по протоколах програмно. Також це збільшує надійність пристрою. Розмір оперативної та постійної пам’яті, яка присутня у складі мікроконтролера, дозволяє записати в пам’ять мікроконтролера керуючу програму та безперебійно функціонувати.

Тож можна зробити висновок, що застосування даного мікроконтролера у пристрої із USB інтерфейсом для програмування 8 та 32 розрядних мікроконтролерів є доцільним.

2.2 Опис протоколу передачі даних USB

USB (Universal Serial Bus – «універсальна послідовна шина») – послідовний інтерфейс для підключення периферійних пристроїв до обчислювальної техніки. Отримав широке розповсюдження і фактично став основним інтерфейсом підключення периферії до побутової цифрової техніки [18].

Інтерфейс дозволяє не тільки обмінюватися даними, а й забезпечувати електроживлення периферійного пристрою. Мережева архітектура дозволяє підключати велику кількість периферії навіть до пристрою з одним роз’ємом USB.

Розробка специфікацій USB проводиться в рамках міжнародної некомерційної організації USB Implementers Forum (USB-IF), яка об'єднує розробників і виробників обладнання з шиною USB. У процесі розвитку вироблено кілька версій специфікацій. Проте розробникам вдалося зберегти високу ступінь сумісності обладнання різних поколінь. Специфікація інтерфейсу охоплює безпрецедентно широке коло питань підключення та взаємодії периферійних пристроїв з обчислювальною системою:

- Уніфікацію роз’ємів і кабелів;
- Нормування енергоспоживання;
- Протоколи обміну даними;
- Уніфікацію функціональності і драйверів пристроїв [18].

Перші специфікації для USB 1.0 були представлені в 1994-1995 роках. Розробка USB підтримувалася фірмами Intel, Microsoft, Philips, US Robotics. USB став «спільним знаменником» під трьома не пов’язаними один з одним прагненнями різних компаній:

- Розширення функціональності комп'ютера. На той момент для підключення зовнішніх периферійних пристроїв до персонального комп'ютера використовувалося кілька «традиційних» (англ. Legacy) інтерфейсів (PS/2, послідовний порт, паралельний порт, порт для підключення джойстика, SCSI), і з появою нових зовнішніх пристроїв розробляли і новий роз'єм. Передбачалося, що USB замінить їх всі і призупинить розробку не традиційних пристроїв.
- Підключити до комп'ютера мобільний телефон. У той час мобільні мережі переходили на цифрову передачу голосу, і жоден з наявних інтерфейсів не підходив для передачі з телефону на комп'ютер як мови, так і даних.
- Простота для користувача. Старі інтерфейси (наприклад, послідовний (COM) і паралельний (LPT) порти) були вкрай прості для розробника, але не давали справжнього «підключи і працюй». Були потрібні нові механізми взаємодії комп'ютера з низько- і середньошвидкісними

зовнішніми пристроями - можливо, більш складні для конструкторів, але надійні і придатні до «гарячого» підключення.

Підтримка USB вийшла у вигляді патча до Windows 95b, в подальшому вона увійшла в стандартне постачання Windows 98. У перші роки пристроїв було мало, тому шину жартома називали «Useless serial bus» - «марна послідовна шина» [19]. Втім, виробники швидко усвідомили користь USB, і вже до 2000 року більшість принтерів і сканерів працювали з новим інтерфейсом.

Hewlett-Packard, Intel, Lucent (нині Alcatel-Lucent), Microsoft, NEC і Philips спільно виступили з ініціативою щодо розробки більш швидкісний версії USB. Специфікація USB 2.0 була опублікована в квітні 2000 року, і в кінці 2001 року ця версія була стандартизована USB Implementers Forum. USB 2.0 є зворотно сумісною з усіма попередніми версіями USB.

Поки відбувалося поширення USB-портів другої версії, виробники зовнішніх жорстких дисків вже «вперлися» в обмеження USB 2.0 - і по струму, і по швидкості. Знадобився новий стандарт, який і вийшов в 2008 році. 4 дроти стало замало, тож додали 5 нових проводів. Перші материнські плати з підтримкою USB 3.0 вийшли в 2010 році. До 2013 року USB 3.0 став масовим.

Специфікації протоколу передачі даних USB

На сьогоднішній день існує кілька поширених специфікацій, які приведені у таблиці 2.1.

Таблиця 2.1 – Список специфікацій USB

Специфікація	**Швидкість**	**Стандарт USB**
Low-Speed	до 1,5 Мбіт/с	USB 1.0
Full-Speed	до 12 Мбіт/с	USB 1.0
High-speed	до 480 Мбіт/с	USB 2.0
SuperSpeed	до 5 Гбіт/с	USB 3.0 / USB 3.1 Gen 1
SuperSpeed 10Gbps	до 10 Гбіт/с	USB 3.1 Gen 2

Метод зв’язку USB

USB є мережею з одним майстром (хостом) і будь-якою кількістю підлеглих пристроїв (device). Топологія мережі – активне дерево. Активне означає що в кожному вузлі дерева знаходиться спеціальний пристрій –

концентратор (хаб). Хаб займається електричним узгодженням кабелів, маршрутизацією пакетів, виявленням підключення/відключення пристроїв і іншими функціями. Всі з'єднання в мережі електрично і протокольно ідентичні.

USB дозволяє виконувати гаряче підключення і відключення окремих пристроїв або сегментів мережі. Гаряче означає що робота мережі при цьому не порушується, а майстер здатний визначати факт зміни конфігурації мережі автоматично, в режимі реального часу. Оскільки вся мережа отримує електроживлення від майстра то підтримується можливість автоматичного контролю енергопостачання мережі: пристрій повідомляє майстру про свої потреби, а майстер може заборонити роботу пристрою якщо енергетичні можливості мережі можуть бути перевищені.

Фізичний рівень USB

Спрощена електрична схема USB з’єднання показана на рисунку 2.1, та наведена в додатку Е. Коли до хосту ніхто не підключений, обидві сигнальні лінії D + і D- підтягнуті резисторами 15 кОм до мінуса живлення. При підключенні пристрою одна з ліній підтягується до +3,3 В через резистор 1,5 кОм. Пристрої Low Speed підтягують лінію D-, а пристрої Full Speed - D +. Таким чином хост визначає факт підключення і тип підключеного пристрою. Пристрої High Speed в момент підключення працюють як Full Speed, перемикаючись в високошвидкісний режим після обміну візитками.

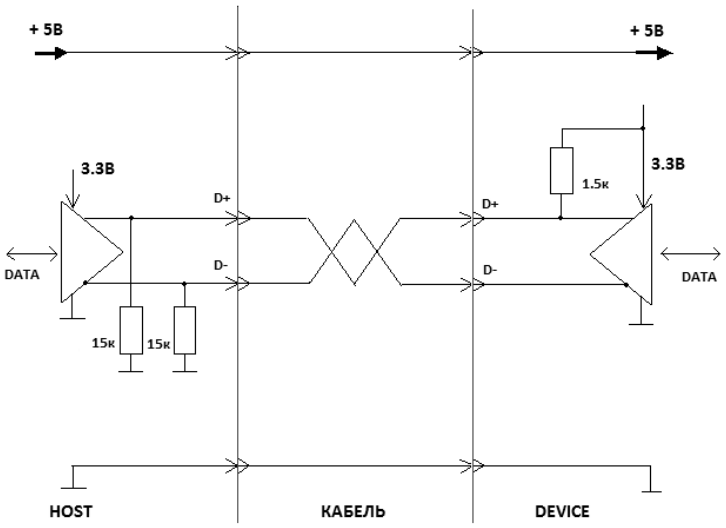


Рисунок 2.1 – Спрощена електрична схема USB з’єднання

Стан диференційної пари, визначений резисторами підтяжки, в специфікації іменується Idle. Той же стан при включеному драйвері позначається буквою J. Протилежний йому стан – буквою K. Замикання обох ліній на мінус іменується Single Ended 0, скорочено SE0; замикання на плюс – SE1.

Дані кодуються за методом NRZI (Non-return-to-zero inverted). За цим методом кожному нульового біту вхідних даних відповідає зміна стану диференційної пари (J → K або K → J), а при одиниці зміни немає. Щоб виключити втрату синхронізації на тривалих одиничних послідовності, застосовують бітстаффінг, тобто примусово вставляють в потік даних нуль на кожні 6 одиниць поспіль.

Стан шини SE0 довше 10 мс трактується пристроєм як скидання (Reset) і вимагає від пристрою реініціалізації USB стека. Стан Idle довше 3 мс поспіль трактується пристроєм як зупинка шини (Suspend) і формально вимагає від пристрою самообмеження в споживанні електроенергії від шини USB. Вихід з Suspend відбувається або по відновленню активності хосту, або пристрій може, при необхідності, подати спеціальний сигнал Resume. Сигнал Resume складається зі стану K на кілька мілісекунд, що завершується послідовністю SE0, SE0, J, де кожен стан триває один бітовий інтервал згідно швидкісного режиму пристрою [17].

Структура пакетів USB

Обмін відбувається короткими пакетами. Кожен пакет починається з послідовності Start of Packet, для Low і Full Speed це KJKJKJKK. Далі завжди йде спеціальний ідентифікатор пакета PID (англ. Packet IDentifier), який вказує на тип пакета. Усього є 16 різних типів пакетів, тому PID має розмірність 4 біта. Однак для надійності значення цього поля дублюється в інверсному вигляді, тому довжина поля PID в пакеті 8 біт. Закінчується пакет послідовністю End of Packet: SE0, SE0, J. Мінімальний інтервал часу між пакетами ~ 0,1 мкс (для Full Speed).

Залежно від типу пакета між PID і EoP може міститися ряд інших полів з параметрами пакета і/або даними. Всі ці поля (включаючи PID) передаються молодшим бітом вперед (LSB first).

Пакети типу IN, OUT, SETUP є заголовками багато пакетної транзакції з обміном даними. Вони містять поля адреси пристрою для транзакції і

номера кінцевої точки (Endpoint), з якою буде обмін. Цілісність пакетів засвідчує поле CRC5.

Пакети типу DATA містять поле даних і поле контролю цілісності даних CRC16. Стандарт обмежує максимальну дозволену довжину даних: 8 байт для не налаштованих пристроїв, 64 байта для пристроїв Low Speed, 1023 байта для пристроїв Full Speed і 1024 байти для пристроїв High Speed. Пристрій може встановити свою максимальну довжину даних, меншу дозволеної. Хост зобов'язаний підтримувати максимальну дозволену довжину даних. Пакети типу ACK, NACK, STALL не містять ніяких додаткових полів.

Адреса USB

USB є свого роду мережею, тобто до одного хосту може підключатися кілька пристроїв. Кожному пристрою в процесі початкового конфігурування в момент підключення призначається унікальна адреса. Розмірність адреси 7 біт, нульове значення зарезервовано, відповідно до одного хосту може підключатися до 127 пристроїв. Поле адреси містять тільки ті пакети, що починають транзакцію (IN, OUT, SETUP).

Крім адресації фізично підключених пристроїв, USB пропонує логічну адресацію всередині пристрою. Логічна адресація дозволяє розділити потоки даних по різному функціоналу всередині одного пристрою. Наприклад, клавіатура з тачпадом може мати один канал даних для натискань клавіш, а інший - для даних тачпада. У стеці TCP/IP є пряма аналогія endpoint - порти.

Поле endpoint має розмірність 4 біта, тобто можливі до 16 endpoint. Кожен endpoint може незалежно працювати як приймальний і як передавальний, тому іноді їх налічують 32. Поле endpoint містять тільки ті пакети, що починають транзакцію (IN, OUT, SETUP). У момент підключення в рамках початкового конфігурування пристрій повинен передати хосту інформацію про задіяні endpoint і їх призначення. Ця інформація повинна узгоджуватися з відповідними каналами даних програмного драйвера пристрою хоста. Звернення до незадіяних endpoint викликає відповідь STALL. Пакети SETUP можуть приходити тільки на нульовий endpoint.

Специфікація USB містить поняття тимчасових фреймів і мікрофреймів. Для Low і Full Speed пристроїв кожну мілісекунду хост передає спеціальний пакет SOF (Start of Frame). Цей період називається фрейм. Для High Speed цей пакет передається кожні 125 мкс, такий період

називається мікрофрейм. Специфікація USB вимагає, щоб підтримувалося таке планування транзакцій і пакетів, щоб періодичність розсилки SOF не порушувалася.

Принципи обміну даними USB

Обмін даними відбувається так званими транзакціями – нерозривними послідовностями з декількох пакетів. Ініціатором обміну завжди є хост. Він передає короткий пакет (token), що повідомляє про початок нової транзакції. У цьому пакеті-токені хост вказує напрямок транзакції (IN або OUT), адресу пристрою і номер endpoint. Наприклад, токен OUT означає, що негайно за токеном піде пакет з даними від хоста до пристрою (DATA0 або DATA1). Пакетів з даними може бути кілька в одній транзакції, якщо кожен з них має максимально допустиму для цього пристрою довжину даних. Закінчення пересилання даних визначається по довжині пакета, не рівної максимальної. Як тільки приходить укорочений пакет, пристрій негайно передає у відповідь пакет-підтвердження (handshake), наприклад, ACK (всі дані успішно прийнято), NACK (не зміг прийняти, наприклад, переповнений вхідний буфер), STALL (дані адресовані відключеному endpoint). Всі пакети в транзакції передаються практично разом, максимальна пауза між пакетами не повинна перевищувати ~ 1 мкс (для Full Speed), інакше транзакція буде визнана помилковою.

Аналогічно відбувається передача даних від пристрою до хосту. Хост ініціює передачу токеном IN. Якщо пристрій не має готових даних для передачі, то відповідає NACK і транзакція завершується. Якщо дані готові, пристрій починає передавати пакети DATA0/DATA1. Принцип закінчення передачі аналогічний: неповна довжина пакета з даними. Отримавши неповний пакет, хост відповідає пристрою пакетом-підтвердженням ACK.

Транзакція з токеном SETUP повністю аналогічна транзакції OUT, відмінності лише в логіці сприйняття даних пристроєм: це параметри з’єднання, які керують роботою USB стека пристрою [19].

Специфікація USB надає кілька способів з’єднання. Кожному включеному endpoint повинен відповідати будь-який з методів. Control, Interrupt і Bulk використовують протокол обміну з підтвердженням. Метод bulk дозволяє хосту вільно обмінюватися даними з пристроєм на свій розсуд. Метод control аналогічний bulk, але обмінюється з пристроєм лише спеціальними даними, які керують роботою USB-протоколу відповідно до

специфікації (в рамках транзакцій типу SETUP). Оскільки периферійні пристрої не можуть ініціювати обмін, то для передачі даних які раптово виникають у пристрої придумали метод interrupt, який дозволяє опитувати пристрій з заданим періодом. Метод interrupt широко застосовується для опитування клавіатур і мишок. Окремо стоїть метод isochronous, що дозволяє зарезервувати частину смуги пропускання USB-шини для таких даних, як аудіо або відео. Isochronous не підтримує контролю цілісності передачі (пакети ACK і NACK не передаються), а значить, не передбачені повтори в разі помилок: невірно прийняті дані пропадають.

Ініціалізація пристроїв USB

У момент підключення хост запитує у пристрою ряд стандартизованих відомостей (дескрипторів), на підставі яких приймає рішення, як з цим пристроєм працювати. Дескриптори містять відомості про виробника і тип пристрою, на підставі яких хост підбирає програмний драйвер. Таблиці дескрипторів і призначення полів докладно описані в розділі 9 специфікації USB.

Після цього хост проводить зміну швидкості (якщо пристрій High Speed) і призначає пристрою адресу.

Стандартні класи USB пристроїв

Призначення USB-пристроїв може визначатися кодами класів, які повідомляються USB-хосту для завантаження необхідних драйверів. Коди класів дозволяють уніфікувати роботу з однотипними пристроями різних виробників. Пристрій може підтримувати один або кілька класів, максимальна кількість яких визначається кількістю доступних endpoints.

Опис кодів стандартних класів приведено у таблиці 2.2 [21].

Розроблений у магістерській кваліфікаційній роботі пристрій з USB інтерфейсом для програмування 8 та 32 бітних мікроконтролерів є композитним пристроєм, який одночасно відноситься до класу Custom HID Device та Virtual CDC Device. Клас Custom HID Device підтримується усіма сучасними операційними системами, включно із Linux та Windows, та дозволяє без розробки власних драйверів реалізувати обмін між USB пристроєм та персональним комп’ютером. Єдиним мінусом даного класу є те, що максимальна швидкість обміну із хостом дорівнює 64 кБіт/с, що суттєво менше максимальної швидкості, яку підтримує вибраний керуючий мікроконтролер (STM32F042K6T6 працює за специфікацією Full-Speed із

максимальною швидкістю передачі даних 12 Мбіт/с). Проте, враховуючи те, що для програмування більшості мікроконтролерів такої швидкості передачі даних більш ніж достатньо, даний клас пристроїв нам підходить ідеально. Клас Virtual CDC Device дозволяє операційній системі обмінюватися даними із мікроконтролером як із звичайним COM портом.

Таблиця 2.2 – Опис кодів стандартних класів USB пристроїв

Код	**Назва**	**Приклади використання/примітки**
00h	N/A	Не задано
01h	Audio	Звукова карта, MIDI
02h	Communication Device (CDC)	Модем, Мережева карта, COM-порт
03h	Human Interface Device (HID)	Клавіатура, Мишка, Джойстик
05h	Physical Interface Device (PID)	Джойстик с підтримкою Force feedback
06h	Image	Веб-камера, Сканер
07h	Printer	Принтер
08h	Mass Storage Device (MSD)	USB-накопичувач, карта пам’яті, кардридер, цифрова фотокамера
09h	USB hub	USB-хаб
0Ah	CDC Data	Використовується спільно з класом CDC
0Bh	Smart Card Reader (CCID)	Зчитувач смарт-карт
0Dh	Content security	Біометричний сканер
0Eh	Video Device Class	Веб-камера
0Fh	Personal Healthcare	Індикатор пульсу, медичне обладнання
DCh	Diagnostic Device	Використовується для перевірки сумісності з USB
E0h	Wireless Controller	Bluetooth-адаптер
EFh	Miscellaneous	ActiveSync-пристрій
FEh	Application-specific	IrDA-пристрій, режим оновлення прошивки (DFU)
FFh	Vendor-specific	На розгляд виробника

Розроблений у магістерській кваліфікаційній роботі пристрій з USB інтерфейсом для програмування 8 та 32 бітних мікроконтролерів є композитним пристроєм, який одночасно відноситься до класу Custom HID Device та Virtual CDC Device. Клас Custom HID Device підтримується усіма сучасними операційними системами, включно із Linux та Windows, та дозволяє без розробки власних драйверів реалізувати обмін між USB пристроєм та персональним комп'ютером. Єдиним мінусом даного класу є те, що максимальна швидкість обміну із хостом дорівнює 64 кБіт/с, що суттєво менше максимальної швидкості, яку підтримує вибраний керуючий мікроконтролер (STM32F042K6T6 працює за специфікацією Full-Speed із максимальною швидкістю передачі даних 12 Мбіт/с). Проте, враховуючи те, що для програмування більшості мікроконтролерів такої швидкості передачі даних більш ніж достатньо, даний клас пристроїв нам підходить ідеально. Клас Virtual CDC Device дозволяє операційній системі обмінюватися даними із мікроконтролером як із звичайним COM портом.

Для того, щоб операційна система розпізнала розроблений пристрій правильно, використовуються драйвери від ST Microelectronics для підтримки Virtual CDC Device, та відредагований *.inf файл, у якому вказано VID та PID пристрою. Після того як системі вказати на місцезнаходження даного *.inf файлу розроблений пристрій розпізнається як композитний пристрій.

2.3 Інтерфейс програмування МК Atmel

AVR-мікроконтролери надають користувачеві кілька різних інтерфейсів для програмування. Це послідовне програмування при низькій напрузі через SPI, паралельне програмування при високій напрузі та програмування по інтерфейсу JTAG. Останній тип програмування доступний лише деяким моделям старшого сімейства.

Переважна більшість AVR-мікроконтролерів володіють також здатністю само-програмування. Крім цього FLASH-пам'ять може бути перепрограмована в режимі налагодження через одно провідний інтерфейс DebugWIRE, наявний в ряді моделей ATmega і у всіх нових моделях ATtiny.

Паралельне програмування при високій напрузі вимагає значного числа виводів МК і додаткового джерела напруги 12 В. З цієї причини конструкція

програматорів досить складна. При високовольтному програмуванні досягається найбільша швидкість запису і надається максимальний доступ до ресурсів AVR. На рисунку 2.2 показана схема підключення МК ATTiny2313 при паралельному програмуванні, та наведена в додатку Ж.

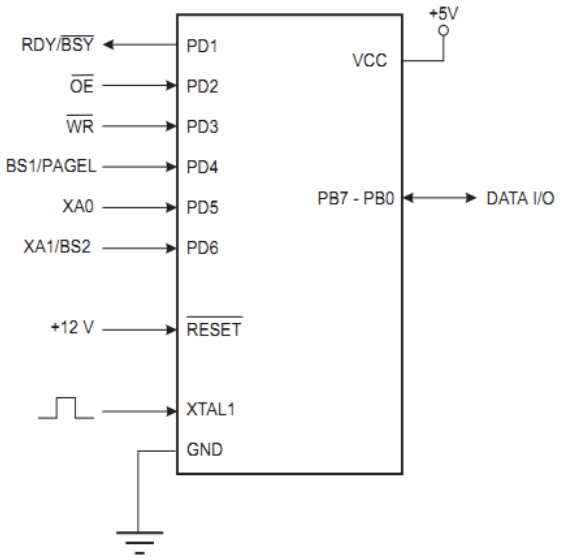


Рисунок 2.2 – Паралельне програмування ATTiny2313

Як видно із рисунку 2.2, для програмування у паралельному режимі МК ATTiny2313 використовує 16 виводів. Для прикладу, МК старшого сімейства ATMega16 використовує 18 виводів. Враховуючи велику кількість виводів, які використовуються при програмуванні, керуюча програма та схема програматора досить складні. А необхідність використання джерела високої напруги робить неможливим програмування МК внутрішньо схемно.

Інтерфейс JTAG дуже зручно використовувати в тих випадках, коли необхідно вести програмування і налагодження в одному циклі розробки.

Низьковольтне послідовне програмування через SPI, найбільш поширене. Цей спосіб варто визнати основним при програмуванні AVR-мікроконтролерів. Його підтримують всі моделі з ядром AVR, за винятком двох застарілих представників молодшого сімейства ATtiny11x і ATtiny28x.

Для взаємодії програматора з МК при послідовному низьковольтному програмуванні використовується апаратний модуль SPI. Це дуже практичне рішення, що дозволяє використовувати мінімальну кількість виводів і змінювати алгоритми роботи пристрою попередньо запаяного на плату. Саме тому програмування через SPI називають також ще внутрішньо схемним

програмуванням або ISP (In System Programming). На рисунку 2.3 показана схема з’єднання програматора з МК AVR при програмуванні через SPI, та наведена в додатку К.

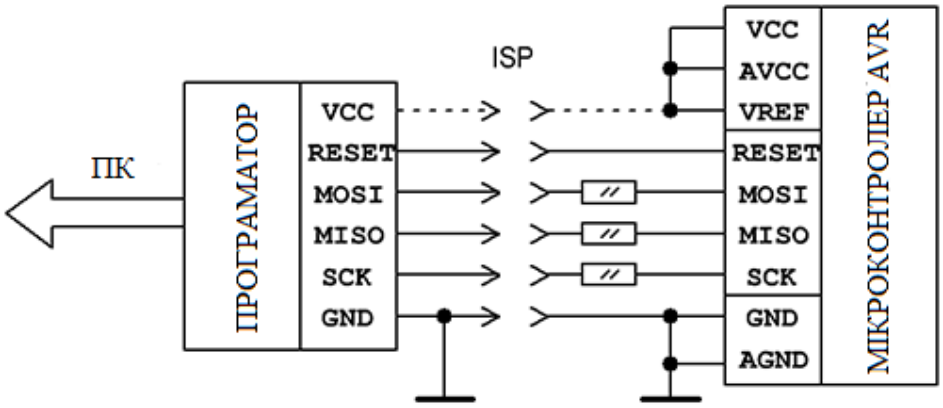


Рисунок 2.3 – Схема з’єднання програматора з МК AVR при програмуванні через SPI

Для більш надійної роботи послідовно до ліній MISO, MOSI, SCK рекомендується включати резистори невеликого номіналу.

При внутрішньо схемному програмуванні для читання і запису доступні FLASH-пам'ять програм, EEPROM-пам'ять даних, біти захисту та керуючі FUSE-біти [22].

Все спілкування програматора з МК складається з відправки 32-бітових команд і прийому відповідей контролера. Повний перелік команд наводиться в даташитах. Для програмування МК потрібно виконати наступні операції:

- Переведення контролера в режим програмування;
- Читання ідентифікатора пристрою;
- Стирання;
- Запис flash;
- Перевірка записаного;

Режим програмування вмикається подачею «0» на вивід RESET. Компанія Atmel рекомендує спочатку виставити на виводах RESET і SCK низький рівень, а тільки потім подавати на контролер живлення. Якщо такої можливості немає, потрібно після включення живлення подати «0» на SCK, а потім позитивний імпульс на RESET. Цей процес показано на рисунку 2.4.

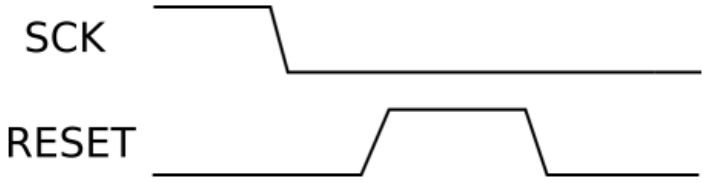


Рисунок 2.4 – Переведення МК в режим програмування

Після цього необхідно зачекати як мінімум 20 мс, а потім передати команду ввімкнення режиму програмування $AC $53 $00 $00. Відправлення команди «Program Enable» показано на рисунку 2.5.

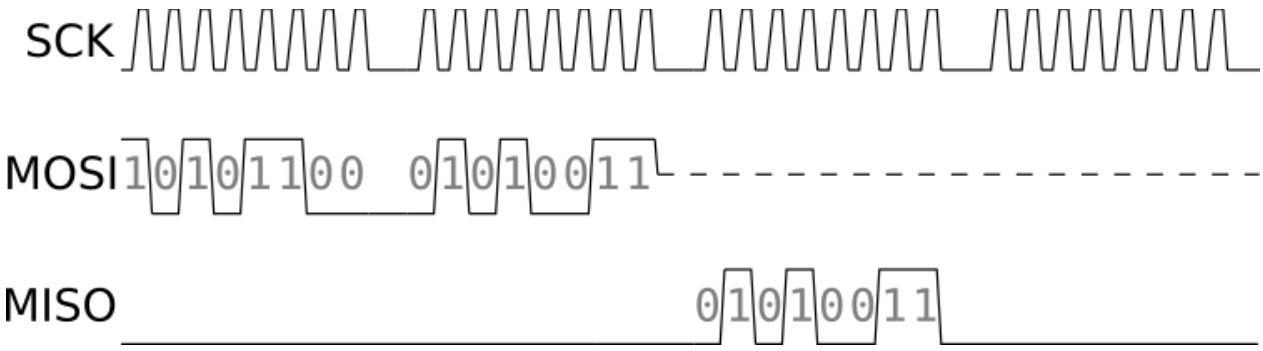


Рисунок 2.5 – Відправлення команди «Program Enable»

Під час передачі третього байта контролер повинен переслати назад другий байт ($53). Якщо це сталося, значить, все добре, команда прийнята, контролер чекає подальших інструкцій. Якщо відповідь відрізняється, потрібно перезавантажити МК і спробувати все спочатку.

Перш ніж що-небудь писати в пам'ять МК, потрібно прочитати ідентифікатор. Кожна модель контролера має свій трьох байтний ідентифікатор (Signature). Прочитати його можна командами виду $30 $00 0000 00aa $00 Замість aa (третій байт команди) слід підставити 00 для першого байта ідентифікатора, 01 - для другого і 10 - для третього. Відповідний байт ідентифікатора буде переданий контролером при відправці 4-го байта команди (рисунок 2.6).

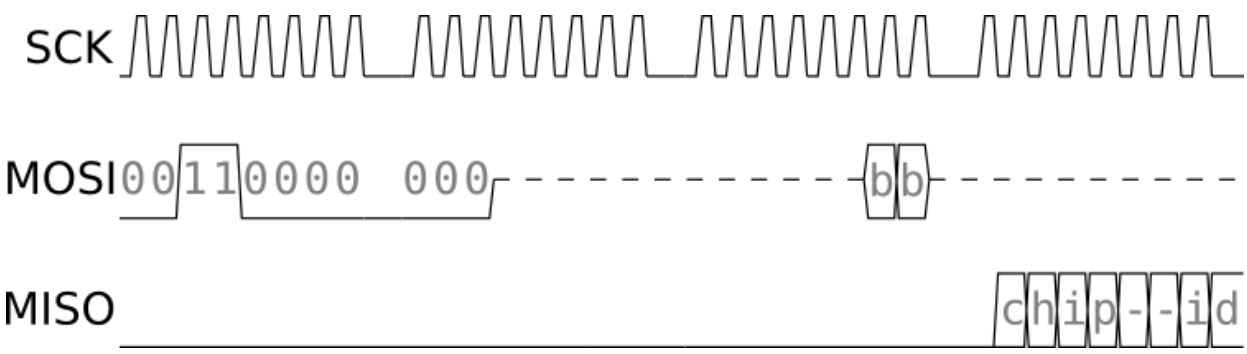


Рисунок 2.6 – Команда «Read Signature Byte»

Як приклад, для МК ATTiny2313 значення ідентифікатора рівне $1E $91 $0A:

1) 0x1E – показує що вироблено компанією Atmel;
2) 0x91 – показує 2 кб Flash пам’яті;
3) 0x0A – показує що підключено ATTiny2313, якщо другий біт 0x91.

Далі потрібно виконати операцію очистки пам’яті яка здійснюється відправкою команди «Chip Erase» $AC $80 $00 $00. Цією командою виконується стирання вмісту Flash і EEPROM (всі комірки будуть містити FF), а також зняття lock-бітів, якщо вони встановлені (рисунок 2.7).

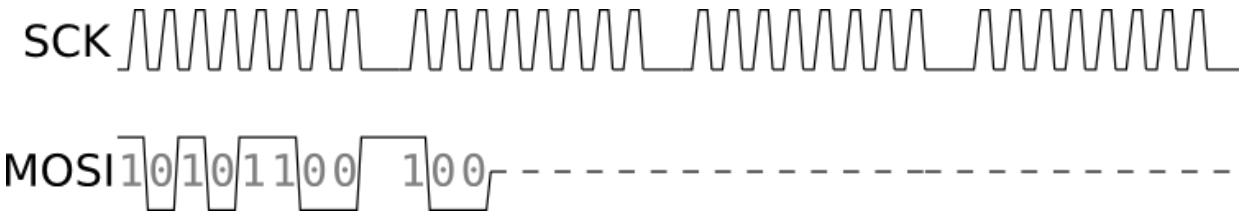


Рисунок 2.7 – Команда «Chip Erase»

Після очистки пам’яті МК потрібно зачекати як мінімум 5 мс, після чого можна починати запис у Flash пам’ять. У МК AVR Flash-пам'ять розділена на сторінки, кожна сторінка має свій розмір. Наприклад МК ATTiny2313 має 2 кб Flash пам’яті, яка поділена на 64 сторінки по 16 слів на сторінці. Кожне слово складається із двох байт. Запис у flash здійснюється в два етапи.

Спочатку необхідно завантажити дані в буфер сторінки, для цього використовується команда «Load Program Memory Page» 0100H000 000xxxxx xxxxbbbb iiiiiiii. Для завантаження молодшого байта слова замість H потрібно

відправити 0, для старшого байта - 1. 4 молодших біта 3-го байта команди bbbb - адреса слова на сторінці, iiiiiiii – даня які потрібно завантажити. Спочатку завжди повинен завантажуватися молодший байт слова, а потім - старший байт того ж слова (рисунок 2.8).

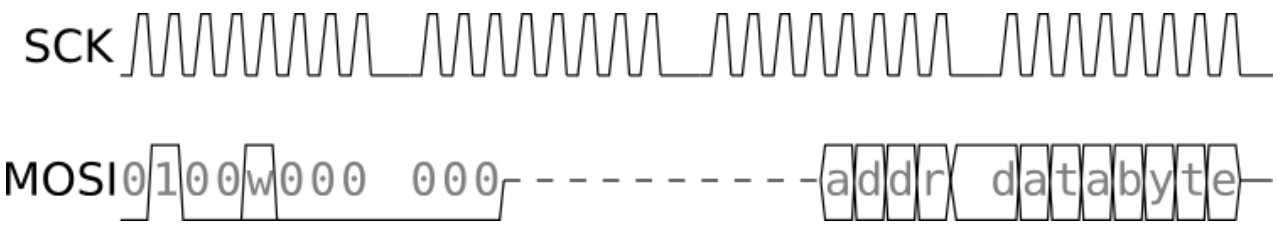


Рисунок 2.8 – Команда «Load Program Memory Page»

Після того, як буфер сторінки завантажений, потрібно виконати команду «Write Program Memory Page» 01001100 000000aa bbbbxxxx xxxxxxxx для запису сторінки безпосередньо в пам'ять контролера. Два молодших біти другого байта і старші 4 біти третього аа: bbbb - номер сторінки для запису (рисунок 2.9). Дана команда змінюється в залежності від об’єму пам’яті МК. Перед завантаженням в буфер сторінки нових даних необхідно зачекати мінімум 5 мс.

Після того як вміст Flash завантажено, потрібно перевірити що записалось у МК і якщо щось записалось із помилкою, повторити запис знову. Для перевірки необхідно прочитати вміст Flash пам’яті і порівняти із оригіналом.

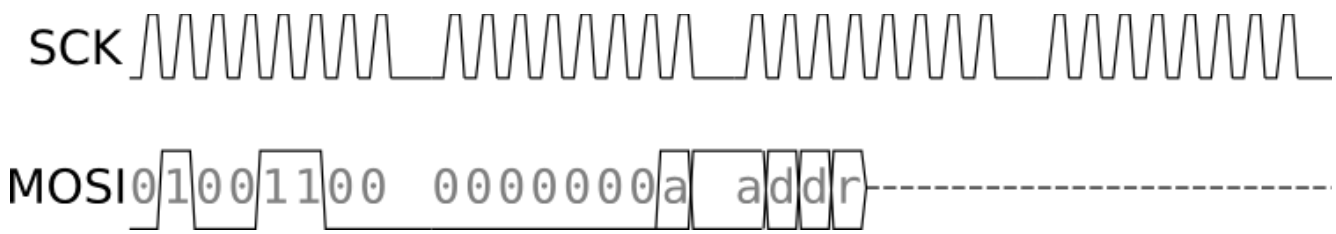


Рисунок 2.9 – Команда «Write Program Memory Page»

Читання Flash виконується по одному байту. Команда «Read Program Memory» виглядає так: 0010H000 000000aa bbbbbbbb oooooooo. Для читання молодшого байта замість H потрібно надіслати 0, старшого – 1. Два молодших біти другого байта і весь третій байт аа: bbbbbbbb - адреса слова в

пам'яті. Прочитаний байт повертається під час передачі 4-го байта команди. Дана команда змінюється в залежності від об’єму пам’яті МК. Команда «Read Program Memory» показана на рисунку 2.10.[22]

Після запису Flash можна записувати дані у EEPROM МК. Запис у EEPROM можна виконувати як по одному байту, так і по сторінці.

Для запису по одному байту потрібно надіслати команду «Write EEPROM Memory» 11000000 000xxxxx xbbbbbb iiiiiiii. У цій команді третій біт – адреса комірки EEPROM, у яку потрібно записати дані із четвертого біта iiiiiiii. Команда «Write EEPROM Memory» показана на рисунку 2.11. Дана команда змінюється в залежності від об’єму EEPROM пам’яті МК. Після запису кожного байта необхідно зачекати мінімум 5 мс.

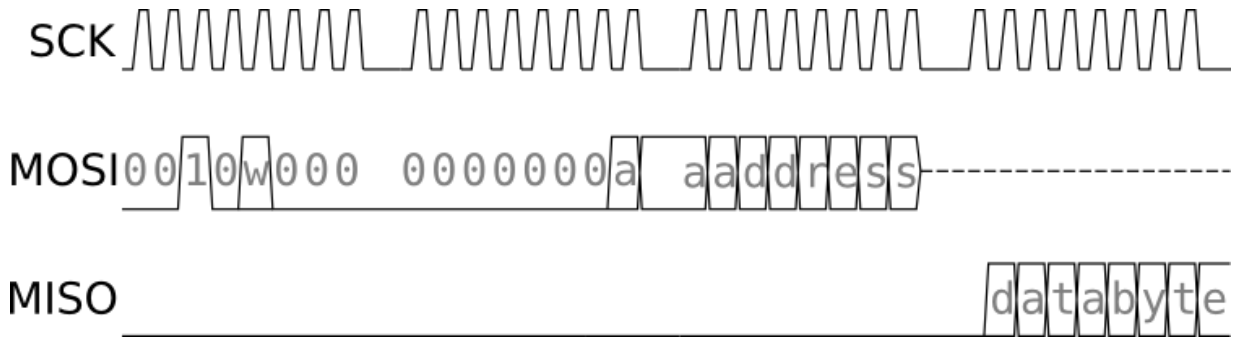


Рисунок 2.10 – Команда «Read Program Memory»

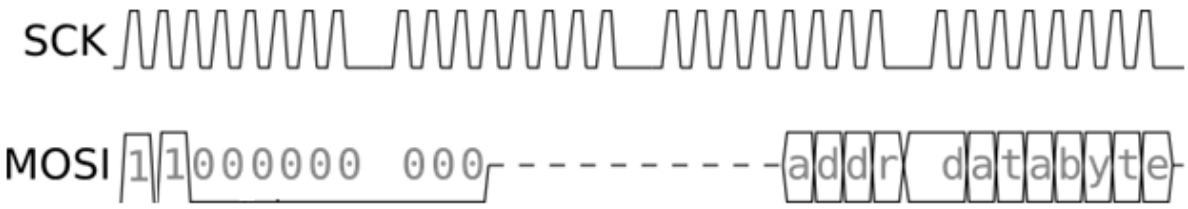


Рисунок 2.11 – Команда «Write EEPROM Memory»

Для запису EEPROM у режимі сторінок спершу потрібно завантажити дані до буфера EEPROM командою «Load EEPROM Memory Page» 11000001 00000000 000000bb iiiiiiii. У даній команді bb – адреса байту iiiiiiii на сторінці (рисунок 2.12).

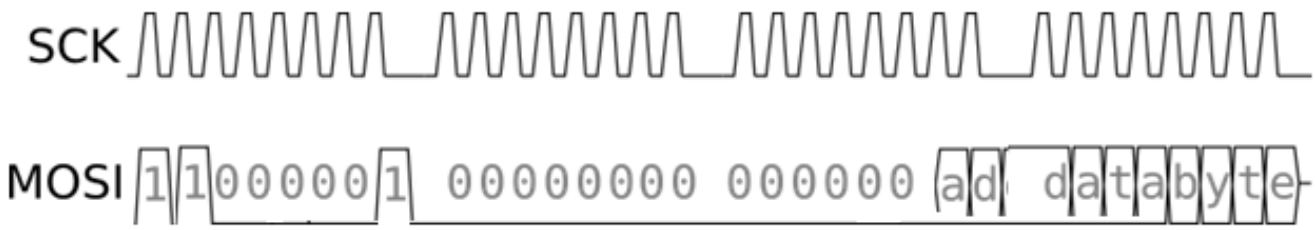


Рисунок 2.12 – Команда «Load EEPROM Memory Page»

Потім, для завантаження сторінки EEPROM у потрібно надіслати команду «Write EEPROM Memory Page» 11000010 00xxxxxx xbbbbb00 xxxxxxxx. У даній команді bbbbb – номер сторінки EEPROM (рисунок 2.13). Перед завантаженням у буфер нової сторінки потрібно зачекати мінімум 5мс.

Після запису EEPROM для перевірки записаних даних, потрібно так само як і для Flash прочитати вміст EEPROM із МК і зрівняти із оригіналом.

Читання EEPROM виконується по одному байту командою «Read EEPROM Memory». Дана команда виглядає наступним чином: 10100000 000xxxxx xbbbbbbb oooooooo, де bbbbbbb – адреса байту, який необхідно прочитати. Прочитаний байт повертається із МК при відправці четвертого байта. Команда «Read EEPROM Memory» показана на рисунку 2.14.

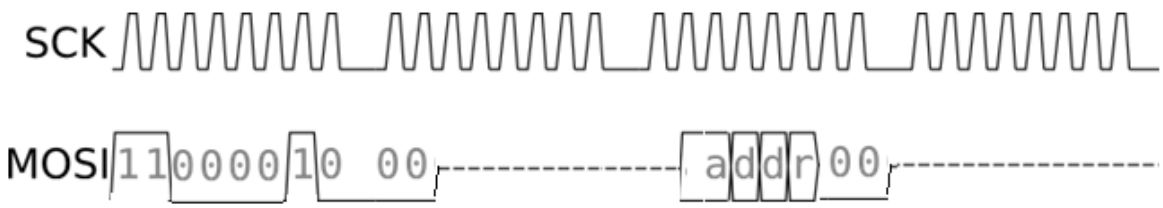


Рисунок 2.13 – Команда «Write EEPROM Memory Page»

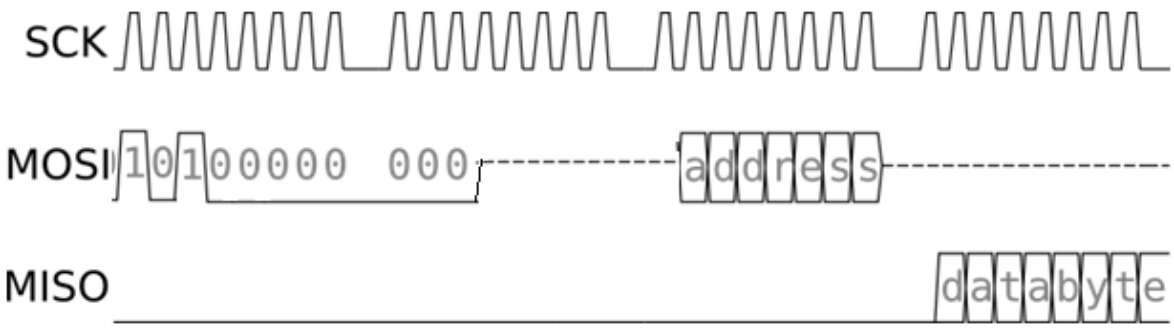


Рисунок 2.14 – Команда «Read EEPROM Memory»

Щоб завершити програмування і перевести МК в робочий режим, потрібно подати на RESET логічний рівень «1». Контролер запуститься і буде працювати за новою програмою.

МК AVR також містять Fuse та Lock біти. Програмувати їх потрібно обережно, так як неправильне їх виставлення може заблокувати МК для подальшої роботи із ним за допомогою SPI інтерфейсу. Щоб розблокувати заблокований МК потрібно скористатись паралельним високовольтним програматором.

Fuse біти зазвичай складаються із трьох областей: Fuse Low bits, Fuse High bits, Fuse Extended bits. Щоб їх прочитати потрібно у режимі програмування до МК надіслати команди:

- Fuse Low bits – 01010000 00000000 xxxxxxxx oooooooo;
- Fuse High bits – 01010000 00001000 xxxxxxxx oooooooo;
- Fuse Extended bits – 01010000 00001000 xxxxxxxx oooooooo.

Зчитане значення Fuse бітів повертається при надсиланні четвертого бійта.

Щоб записати Fuse біти необхідно у режимі програмування до МК надіслати команди:

- Fuse Low bits – 01011100 10100000 xxxxxxxx iiiiiiii;
- Fuse High bits – 01011100 10101000 xxxxxxxx iiiiiiii;
- Fuse Extended bits – 10101100 10100100 xxxxxxxx iiiiiiii.

Нові значення Fuse бітів надсилаються в четвертому байті.

Для читання Lock бітів потрібно у режимі програмування надіслати до МК команду 01011000 00000000 xxxxxxxx xxoooooo. Значення Lock бітів повертаються під час відправки четвертого байта. Для запису Lock бітів потрібно надіслати команду 10101100 111xxxxx xxxxxxxx 11iiiiii, де замість iiiiii потрібно надсилати нові значення Lock бітів [22].

2.4 Мікроконтролери STM32

На сьогоднішній день восьми розрядні мікроконтролери є застарілими. Якщо рівняти восьми розрядні мікроконтролери та тридцяти двох розрядні у одному ціновому діапазоні, можна побачити, що восьми бітні мікроконтролери мають набагато менше периферії, працюють на менших

частотах, мають менше пам’яті. У таблиці 2.3 приведено порівняння восьми та тридцяти двох бітні мікроконтролери вартістю до 100 гривень [23].

Споживання енергії – вказані в оптимальних умовах відповідно до технічного опису (мінімальна напруга для даної частоти + відключена зайва периферія). На «високих» частотах у деяких мікроконтролерів споживання на МГц вище так як потрібно більш високу напругу. Також потрібно взяти до уваги, що значення MIPS у ARM, та у 8/16-и бітних мікроконтролерів не рівноцінні, так ARM здатні виконати більш складні операції, такі як множення 32х32 за один такт, на що восьми та шістнадцяти бітним мікроконтролерам потрібно чотири такти.

Таблиця 2.3 – Порівняння можливостей мікроконтролерів

Мікроконтролер	Ядро	Ціна, грн	Flash пам’ять (кб)	RAM пам’ять (кб)	Продуктивність (MIPS)	Споживання енергії (мА/МГц)
STM32F100C4T6B	ARM 32-bit Cortex-M3	58,00	16	4	30	0.23
AtTiny13A	Atmel 8bit	15,50	1	0.064	20	0.3 (1МГц) 0.55 (20 МГц)
Atmega48	Atmel 8bit	53,0	4	0.5	20	0.3 (1 МГц) 0.55 (20 МГц)
MSP430F2101IPW	TI 16bit MSP430	73,00	1	0.12	16	0.25 (1 МГц) 0.4 (16 МГц)
PIC16F676	PIC16 8bit	48,50	2	0.22	5	0,1*4
PIC12F509	PIC12 8bit	29,75	2	0.041	5	0,087*4
PIC10F200T	PIC10 8bit	10,00	0.384	0.016	1	0,043*4

По енергоспоживанню 8-й і 16-й бітні контролери відчутно програють, а якщо врахувати що ARM може більше перебувати в режимі очікування при виконанні тієї ж задачі, відрив буде ще сильніше. PIC в цифрах виглядає

добре, але цифру в їхньому випадку треба множити на 4 (тому що потрібно по 4 такти на виконання команд).

Зважаючи на приведену інформацію, багато розробників переходять з восьми розрядних (або шістнадцяти розрядних) мікроконтролерів на тридцяти двох розрядні. Однією з причин переходу є те, що завдання ускладнюються, а потужність восьми розрядних мікроконтролерів обмежена, при цьому їх вартість порівнянна з новими 32-бітними мікроконтролерами. Іще однією перевагою мікроконтролерів STM32 є те, що в середині одного сімейства мікроконтролерів, їх можна замінити на більш потужний (або слабший) без зміни конструкції друкованої плати завдяки pin-to-pin сумісності, якої дотримуються виробники.

Якщо, наприклад, розробнику знадобився корпус LQFP 64 і мінімальний набір пам'яті і функціоналу, то його вибір зупиниться на мікросхемі STM32F100R4 (16 Кбайт FLASH, 4 Кбайт RAM, 24 МГц). Далі потрібен більш швидкий АЦП і флеш-пам'ять розміром 128 кбайт. Тоді він переходить на мікросхему STM32F101RB. Всі вводи-виводи в цих мікросхемах ідентичні, тобто плату чіпати не треба. Необхідно тільки оновлення програмного коду. Далі в виробі потрібен USB-інтерфейс. Тоді на допомогу приходить мікроконтролер STM32F103RE, теж без змін на рівні друкованої плати. Нарешті, на основі цього виробу вирішили зробити портативний прилад. Тоді розробник переходить на мікроконтролер STM32L151RB, який має більш низьке споживання і зберігає всі характеристики STM32F103RE по пам'яті і інтерфейсів. В останнім випадку розробник повинен звернути увагу на одну незначну відмінність (сигнал Vbat стає VLCD), але швидше за все, йому також не буде потрібно нічого змінювати на платі. Можна подивитися на процес і зі зворотного боку. Наприклад, при розробці розглядався багатий набір функціоналу для кінцевого вироби, але вже при виході на виробництво з'ясувалося, що велика частина цього функціоналу надлишкова. Можна взяти більш простий мікроконтролер з лінійки, скоротивши витрати на комплектуючі. Якщо під час розробки по максимуму передбачити всі майбутні варіанти свого вироби на основі pin-to-pin сумісності, то можна з великою ефективністю запускати у виробництво безліч різноманітних виробів. У підсумку, витративши свої зусилля на одну розробку, розробник має можливість масштабувати свої вироби і досить швидко виконувати вимоги ринку.

2.5 Інтерфейси програмування мікроконтролерів STM32

Мікроконтролери STM32 в залежності від моделі підтримують від двох до шести способів завантаження програми у пам’ять. Більшість із вказаних способів це
використання завантажувача bootloader. Максимальна кількість можливих варіантів bootloader у одному мікроконтролері – 5, а саме через протоколи USART, I2C, SPI, CAN, DFU. Хоча не всі моделі STM32 підтримують по п’ять протоколів, кожна із них підтримує протокол USART, що і стало причиною вибору даного протоколу для реалізації завдання.

Завантажувач зберігається у внутрішній завантажувальній пам’яті (системній пам’яті) пристроїв STM32. Він програмується ST під час виробництва. Його головне завдання полягає в тому, щоб завантажити програму у внутрішню флеш-пам’ять за допомогою одного з доступних послідовних периферійних протоколів. Протокол зв’язку, визначається для кожного послідовного інтерфейсу, з сумісним набором команд і послідовностей [24].

Іще одним варіантом завантаження програми у пам’ять STM32 є використання протоколу SWD. У даній роботі він не розглядається, так як він є більш складним та на його розгляд не вистачило часу при підготовці магістерської кваліфікаційної роботи. Так як проект іще не завершений та продовжує розвиватись – протокол SWD буде реалізовано у наступних версіях програмного забезпечення для пристрою.

Протокол USART, який використовується в завантажувачі STM32

Після того, як мікроконтролер перейшов у режим завантажувача USART і мікроконтролер STM32 був налаштований, код завантажувача починає сканувати лінію USARTx_RX, чекаючи, на отримання кадру 0x7F: один стартовий біт, біти даних 0x7F, біт парності і один стоп-біт.

Тривалість цього кадру даних вимірюється за допомогою системного таймера. Підраховане значення таймера потім використовується для обчислення відповідного коефіцієнта швидкості обміну інформацією по відношенню системного таймера.

Далі, код налаштовує послідовний інтерфейс відповідним чином. Використовуючи цю розрахункову швидкість передачі даних, байт

підтвердження 0x79 (ACK) повертається до хосту, та сигналізує про те, що STM32 готовий до прийому команд.

Обчислення швидкості послідовної передачі даних для USARTx, в залежності від довжини першого байта, який отриманий, використовується для управління завантажувачем в широкому діапазоні швидкостей передачі даних. Проте, верхні і нижні межі повинні бути збережені, для того щоб забезпечити належну передачу даних.

Для правильної передачі даних від хосту на мікроконтролер, максимальне відхилення між внутрішньою налаштованою швидкістю передачі даних для USARTx і реальною швидкістю передачі даних хосту повинно бути нижче 2,5%. Відхилення (f_B, у відсотках) між швидкістю передачі даних хосту і швидкості передачі даних мікроконтролера може бути розраховане за наступною формулою [25]

$$f_B = \left| \frac{швидкість\ STM32 - швидкість\ хосту}{швидкість\ STM32} \right| \cdot 100\%, де\ f_B \leq 2{,}5\%.$$

Це відхилення швидкості передачі є нелінійної функцією в залежності від частоти процесора і швидкості передачі даних хосту. Максимум функції (f_B) зростає зі збільшенням швидкості передачі даних хосту. Це пов'язано з меншою швидкістю попереднього подільника передачі даних, та спричиняє помилку квантування.

Найнижча перевірена швидкість передачі (B_{low}) – 1200. Швидкість передачі даних нижче B_{low} викличе переповнення системного таймера. У цьому випадку, USARTx не буде правильно налаштовано.

B_{High} є найвищою швидкості передачі даних, для якоъ відхилення як і раніше не перевищує межі. Всі швидкості передачі даних між B_{low} і B_{High} нижче межі відхилення. Найвища перевірена швидкість передачі даних становить 115 200.

2.6 Набір команд завантажувача

Список команд завантажувача приведено у таблиці 2.4.

Всі команди від інструменту програмування (ПК) до пристрою перевіряються:

1. Контрольною сумою: отримані блоки байт даних перевіряються за допомогою операції XOR. Байт, що містить обчислене операцією XOR значення всіх попередніх байтів додається в кінці кожного сеансу зв’язку (байт контрольної суми). Після обрахунку операцією XOR всі отримані байти (дані + контрольна сума) повинні дати в результаті 0x00.

Таблиця 2.4 – Набір команд завантажувача

Команда	**Код**	**Опис команди**
Get	0x00	Отримує версію і дозволені команди, які підтримуються поточною версією завантажувача
Get Version & Read Protection Status	0x01	Отримує версію завантажувача і статус захисту від читання флеш-пам’яті
Get ID	0x02	Отримує ідентифікатор мікроконтролера
Read Memory	0x11	Читає до 256 байт пам'яті, починаючи з адреси, вказаної далі
Go	0x21	Перехід до коду програми користувача, розташованого у внутрішній флеш-пам'яті або в SRAM
Write Memory	0x31	Записує до 256 байт в RAM або флеш-пам’ять, починаючи з адреси, вказаної далі
Erase	0x43	Стирає від однієї до всіх сторінок флеш-пам’яті
Extended Erase	0x44	Стирає від однієї до всіх сторінок флеш-пам'яті з використанням двох байтного режиму адресації (доступно тільки для версії завантажувача USART v3.0 і вище)
Write Protect	0x63	Включає захист від запису для деяких секторів
Write Unprotect	0x73	Відключення захисту від запису для всіх секторів флеш-пам'яті
Readout Protect	0x82	Забезпечує захист від читання
Readout Unprotect	0x92	Відключення захисту від читання

2. Для кожної команди хост посилає байт та його доповнення (XOR = 0x00).

3. UART: активна перевірка парності (контроль парності).

Кожен пакет або приймається (відповідь ACK) або відкидається (відповідь NACK) [25]:

- ACK = 0x79;
- NACK = 0x1F.

Команда Get

Команда Get дозволяє отримати версію завантажувача і підтримувані команди. Коли завантажувач отримує команду Get, він передає версію завантажувача і підтримувані ним коди команд на хост, як це описано на рисунках 2.15, 2.16.

STM 32 відправляє байти наступним чином:

Байт 1: ACK;

Байт 2: N = 11 = число байтів, які будуть далі – 1 виключаючи поточний байт та байти підтвердження;

Байт 3: Версія bootloader (0<Версія<255), наприклад 0x10=Версія 1.0;

Байт 4: 0x00 – Команда Get;

Байт 5: 0x01 – Команда Get Version & Read Protection Status;

Байт 6: 0x02 – Команда Get ID;

Байт 7: 0x11 – Команда Read Memory;

Байт 8: 0x21 – Команда Go;

Байт 9: 0x31 – Команда Write Memory;

Байт 10: 0x43 або 0x44 – Команда Erase або Extended Erase (мікроконтролер підтримує одну із них);

Байт 11: 0x63 – Команда Write Protection;

Байт 12: 0x73 – Команда Write Unprotect;

Байт 13: 0x82 – Команда Readout Protect;

Байт 14: 0x92 – Команда Readout Unprotect;

Байт 15: ACK.

Рисунок 2.15 – Команда Get: сторона хосту

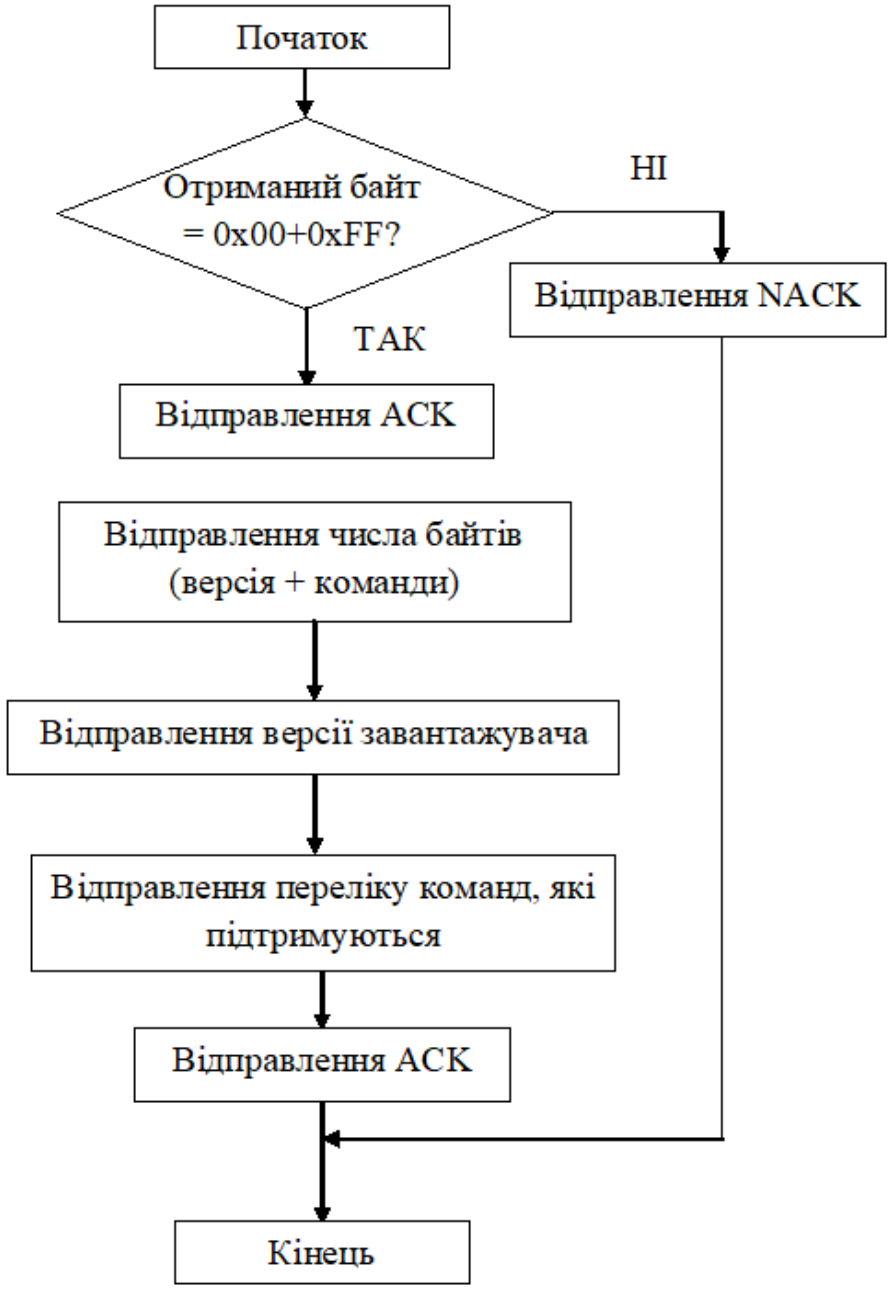


Рисунок 2.16 – Команда Get: сторона мікроконтролера

Команда Get Version & Read Protection Status

Команда Get Version & Read Protection Status використовується для отримання версії завантажувача і інформації про стан захисту від читання. Коли завантажувач отримує команду, він передає інформацію, зазначену нижче (версія, захист від читання) на хост. Процес передачі описано на рисунках 2.17, 2.18.

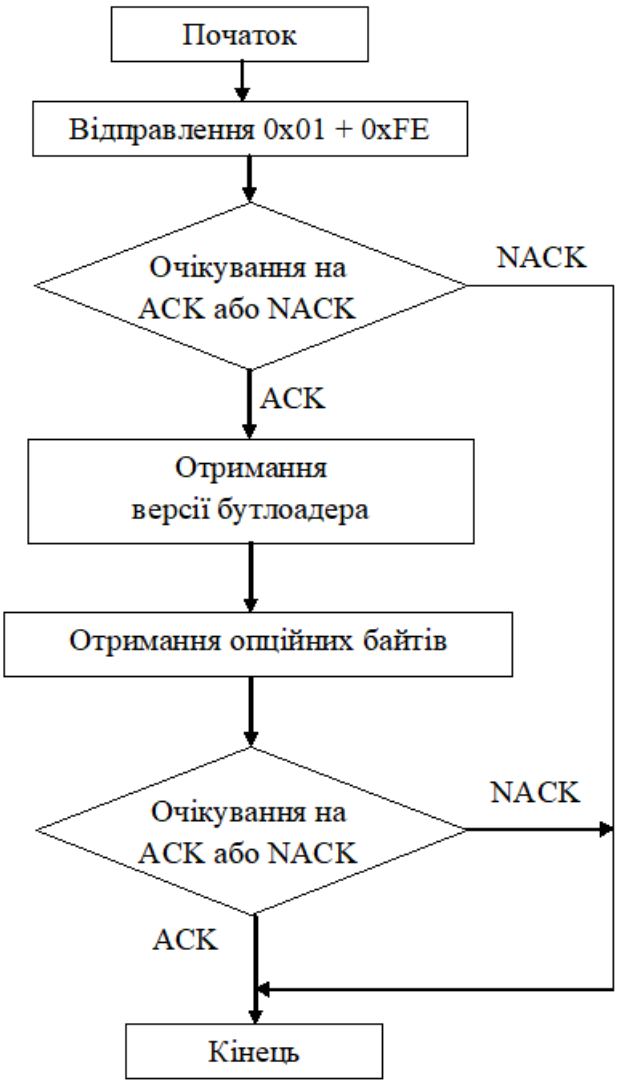


Рисунок 2.17 - Команда Get Version & Read Protection Status: сторона хосту

STM 32 відправляє байти наступним чином:

Байт 1: ACK;

Байт 2: Версія bootloader (0<Версія<255), наприклад 0x10=Версія 1.0;

Байт 3: Опційний байт 1 – для збереження сумісності із загальним протоколом завантажувача;

Байт 4: Опційний байт 2 – для збереження сумісності із загальним протоколом завантажувача;

Байт 5: ACK.

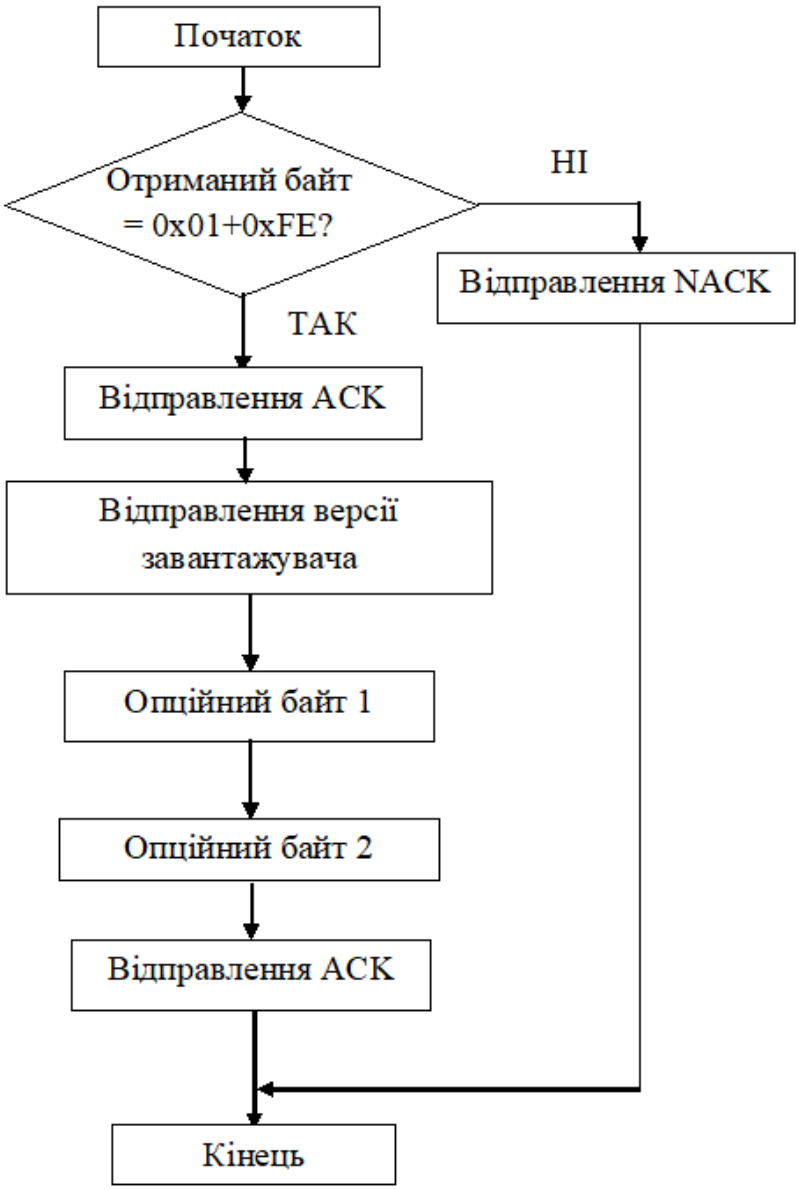


Рисунок 2.18 – Команда Get Version & Read Protection Status: сторона мікроконтролера

Команда Get ID

Команда Get ID використовується, щоб отримати версію ID (ідентифікатор) мікроконтролера. Коли завантажувач отримує команду, він передає ідентифікатор продукту хосту. Процес передачі описано на рисунках 2.19, 2.20.

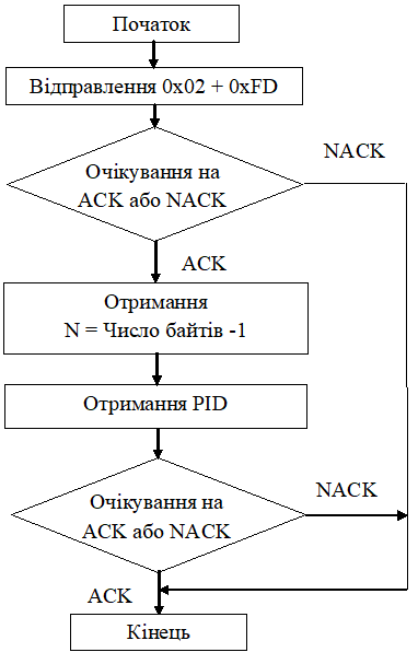


Рисунок 2.19 – Команда Get ID: сторона хосту

STM 32 відправляє байти наступним чином:

Байт 1: ACK; Байт 2: N = кількість байтів - 1 (N = 1 для STM32), за винятком поточного байту та байтів підтвердження; Байт 3-4: PID байт 3 – 0x04 (старший байт), байт 4 – 0xXX (молодший байт); Байт 5: ACK.

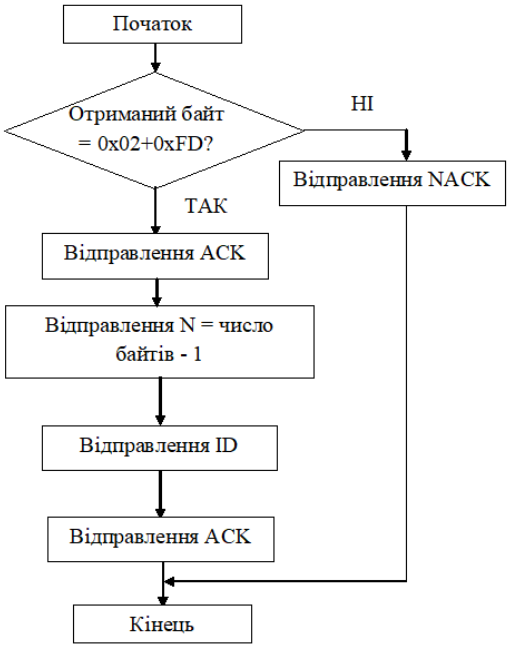


Рисунок 2.20 – Команда Get ID: сторона мікроконтролера

Команда Read Memory

Команда Read Memory використовується для читання даних з будь-якої дійсної адреси пам’яті в RAM, Flash пам’яті і інформаційний блок (зони системної пам’яті або опційних байтів). На рисунках 2.21 та 2.22 показано процес передачі.

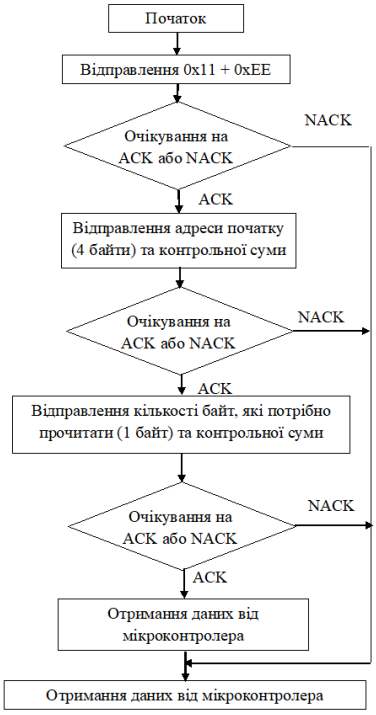


Рисунок 2.21 – Команда Read Memory: сторона хосту

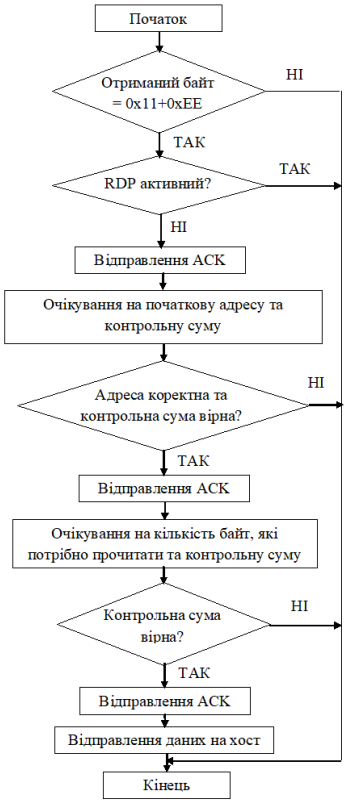


Рисунок 2.22 – Команда Read Memory: сторона мікроконтролера

Коли завантажувач отримує команду Read Memory, він передає ACK байт до хосту. Після передачі ACK байту, завантажувач очікує адреси (4 байти, байт 1 є старшим байтом, байт 4 - молодший байт) і байт контрольної суми. Після отримання завантажувач перевіряє прийняту адресу. Якщо адреса є дійсною, і контрольна сума вірна, то завантажувач передає ACK, в іншому випадку він передає NACK і перериває команду.

Якщо адреса є дійсним, і контрольна сума вірна, то завантажувач очікує кількості байтів для передачі - 1 (N байт) і його байту доповнення (контрольна сума). Якщо контрольна сума вірна вона потім передає необхідні дані ((N + 1) байт) до хосту, починаючи від отриманої адреси. Якщо контрольна сума не вірна, то він посилає NACK перед скасуванням команди.

Хост відправляє байти наступним чином:

Байти 1-2: 0x11 + 0xEE;

Чекає на ACK;

Байти 3-6: початкова адреса (байт 3 – старші розряди, байт 6 – молодші розряди);

Байт 7: контрольна сума – XOR (байт 3, байт 4, байт 5, байт 6);

Чекає на ACK;

Байт 8: Кількість байтів, які потрібно прочитати – 1 ($0<N\leq255$);

Байт 9: Контрольна сума – XOR байту 8 (доповнення до байту 8).

Команда GO

Команда Go використовується для виконання завантаженого коду або будь-якого іншого коду, переходячи за адресою, вказаною хостом. Коли завантажувач отримує команду Go, він передає ACK байт до хосту. Після передачі ACK байта, завантажувач очікує адреси (4 байта, перший байт є старшим байтом адреси, а четвертий байт є молодшим байтом адреси) і байт контрольної суми. Після цього завантажувач перевіряє отриману адресу. Якщо адреса є коректною, і контрольна сума вірна, то завантажувач передає ACK байт, в іншому випадку він передає байт NACK і перериває команду. Опрацювання даної команди показано на рисунках 2.23 та 2.24.

Рисунок 2.23 – Команда Go: сторона хосту

Якщо адреса коректна, і контрольна сума вірна, вбудоване програмне забезпечення початкового завантажувача виконує наступні операції:

- налаштовує регістри периферійних пристроїв, що використовуються завантажувачем на їхні значення по замовчуванню;
- налаштовує основний покажчик стека;
- переходить до комірки пам’яті, адреса якої дорівнює «адреса + 4».

Наприклад, якщо отримано адресу 0x0800 0000 завантажувач перейде в комірку пам’яті за адресою 0x0800 0004.

Хост відправляє байти наступним чином:

Байти 1-2: 0x21 + 0xDE;

Чекає на ACK;

Байти 3-6: початкова адреса (байт 3 – старші розряди, байт 6 – молодші розряди); Байт 7: контрольна сума – XOR (байт 3, байт 4, байт 5, байт 6).

Команда Write Memory

Команда Write Memory використовується для запису даних у будь-яку коректну адресу пам'яті RAM, флеш-пам'яті, області опційних байтів. Коли завантажувач отримує команду Write Memory, він передає ACK байт до хосту. Після передачі ACK байта, завантажувач очікує на адресу (4 байта, перший байт є старшим байтом адреси, а четвертий байт є молодшим байтом адреси) і байт контрольної суми. Після цього хост перевіряє отриману адресу. Для області опційних байтів, початкова адреса має бути базовою адресою області опійного байту, щоб уникнути написання недоречного в цій області.

Якщо прийнята адреса коректна, і контрольна сума вірна, то завантажувач передає ACK байт, в іншому випадку він передає байт NACK і перериває команду. Якщо адреса коректна, і контрольна сума вірна, то завантажувач:

- отримує байт, N, який містить кількість байтів даних, які потрібно отримати;
- приймає дані ((N + 1) байтів) та контрольну суму (XOR з N і всіх байтів даних);
- записує дані в пам'ять, починаючи з прийнятої адреси;
- в кінці команди, якщо операція запису пройшла успішно, завантажувач передає ACK байт; в іншому випадку він передає байт NACK для програми та перериває команду.

Максимальна довжина блоку, який потрібно записати, для STM32 становить 256 байт.

Якщо запис пам’яті відбувається у область опційних байтів, все опції будуть видалені перед записом нових значень, і в кінці команди завантажувач генерує скидання системи, щоб активувати нові конфігураційні байти.

Алгоритм опрацювання команди Write Memory показано на рисунках 2.24 та 2.25.

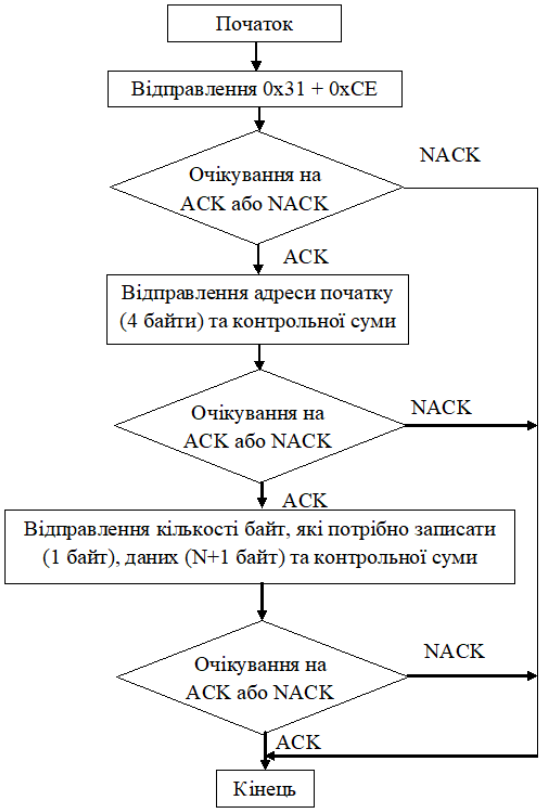


Рисунок 2.24 – Команда Write Memory: сторона хосту

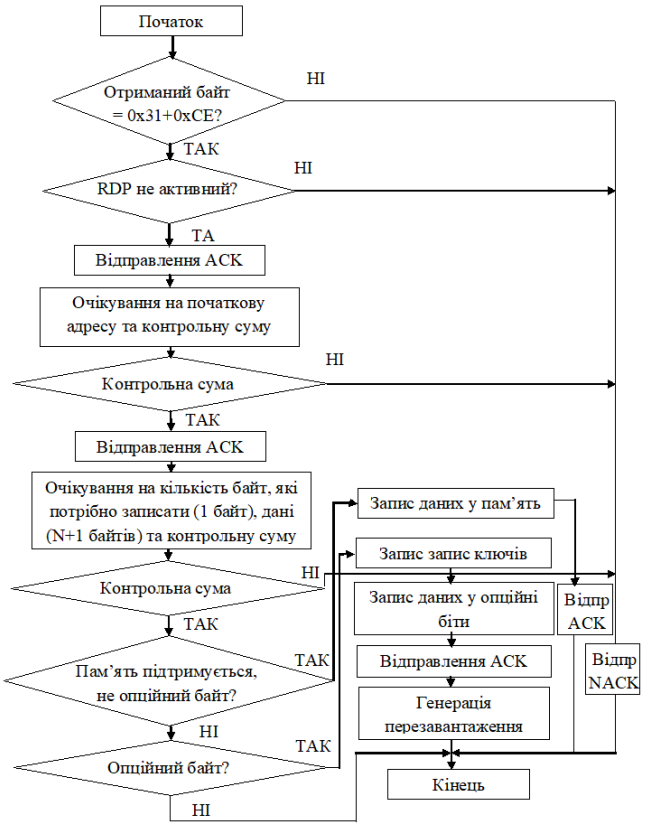


Рисунок 2.25 – Команда Write Memory: сторона мікроконтролера

Хост відправляє байти наступним чином:

Байт 1: 0x31

Байт 2: 0xCE;

Чекає на ACK;

Байти 3-6: початкова адреса (байт 3 – старші розряди, байт 6 – молодші розряди);

Байт 7: контрольна сума – XOR (байт 3, байт 4, байт 5, байт 6);

Чекає на ACK;

Байт 8: Кількість байтів, які потрібно відправити ($0<N\leq255$);

N + 1 байтів даних: (максимально 256 байтів)

Байт 9: Контрольна сума – XOR байту 8 (N, N+1 байтів даних).

Команда Erase Memory

Команда Erase Memory дозволяє хосту стерти сторінки флеш-пам'яті. Коли завантажувач отримує команду Erase Memory, він передає ACK байт на хост. Після передачі ACK байта, завантажувач отримує один байт (кількість сторінок, які будуть стерті), номер сторінки FLASH пам’яті і контрольна сума байт; якщо контрольна сума правильна, то завантажувач стирає пам'ять і посилає ACK байт на хост, в іншому випадку він посилає байт NACK на хост і команда переривається.

Специфікації команди Erase Memory:

1. Завантажувач отримує один байт, який містить N, кількість сторінок, які потрібно стерти -1.

$N = 255$ зарезервований для глобальних запитів стирання. При $0 \leq N \leq 254$, N+1 сторінок видаляються.

2. Завантажувач отримує (N + 1) байт, кожен байт, що містить номер сторінки.

Хост відправляє байти наступним чином:

Байт 1: 0x43;

Байт 2: 0xBC;

Чекає на ACK;

Байти 3: 0xFF або номера сторінок, які потрібно стерти -1 ($0\leq N\leq$максимальна кількість сторінок);

Байт 4: 0x00 (у випадку повного стирання) або N+1 байтів (номера сторінок) та контрольна сума – XOR (N, N+1 байтів).

На рисунках 2.26 та 2.27 показано алгоритм роботи даної команди.

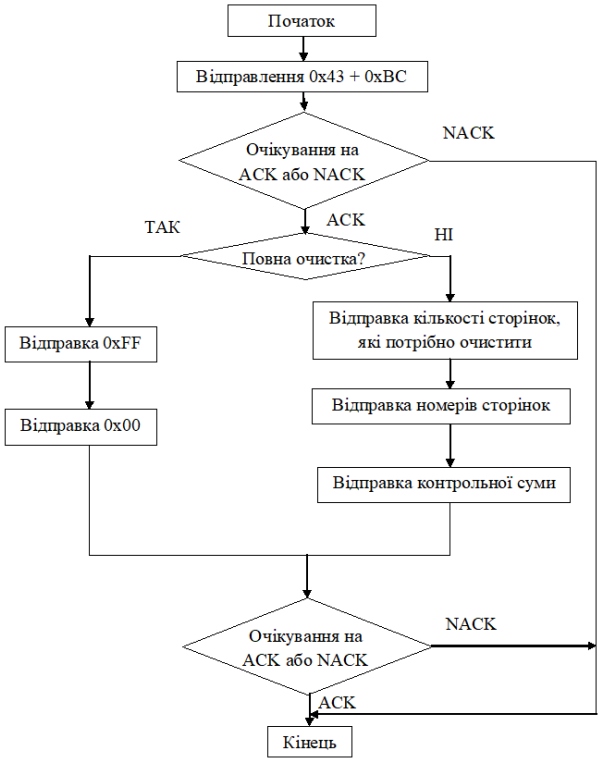


Рисунок 2.26 – Команда Erase Memory: сторона хосту

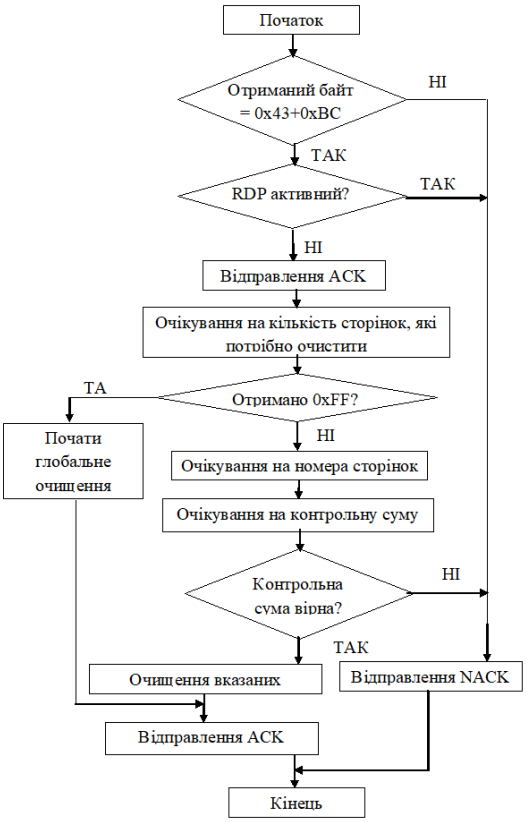


Рисунок 2.27 – Команда Erase Memory: сторона мікроконтролера

Команда Extend Erase Memory

Команда Extend Erase Memory дозволяє хосту стерти сторінки флеш-пам'яті, використовуючи два байта режиму адресації. Коли завантажувач отримує команду Extended Memory Erase, він передає ACK байт на хост. Після передачі ACK байта, завантажувач отримує два байта (кількість сторінок, які потрібно стерти), коди Flash-сторінок пам'яті (кожен з яких кодується з двох байтів, старша частина адреси перший байт) і контрольна сума байт (XOR переданого байта); якщо контрольна сума вірна, то завантажувач стирає пам'ять і відправляє ACK хосту. В іншому випадку він посилає NACK на хост і команда переривається.

Характеристики команди Extend Erase Memory:

1. Завантажувач отримує одне півслово (два байти), які містять N, кількість сторінок, які потрібно стерти:

а) для N = 0xFFFY (де Y від 0 до F) спеціальне стирання виконується:

- 0xFFFF для глобальної очистки;
- 0xFFFE для глобальної очистки частини 1;
- 0xFFFD для глобальної очистки частини 2;
- Коди від 0xFFFC до 0xFFF0 зарезервовані.

б) Для інших значень, де 0 ≤ N <максимальної кількості сторінок: очищається N + 1.

2. Завантажувач отримує:

а) У разі спеціального очистки, один байт: контрольна сума попередніх байтів:

- 0x00 для 0xFFFF;
- 0x01 для 0xFFFE;
- 0x02 для 0xFFFD.

б) У випадку очистки N + 1 сторінок, завантажувач отримує (2 x (N + 1)) байтів, кожне півслово, що містить номер сторінки (закодоване з двох байтів, старша частина адреси перший байт). Після цього контрольна сума всіх попередніх байтів (в один байт).

На рисунках 2.28 та 2.29 показано алгоритми виконання команди Extend Erase Memory.

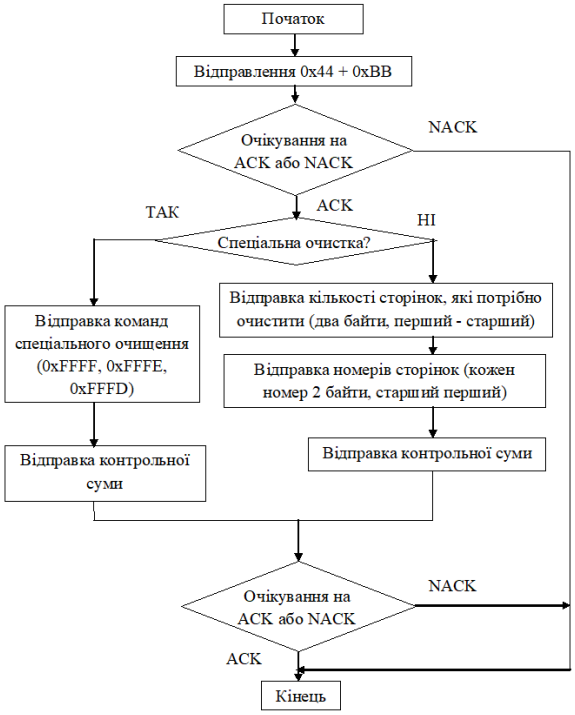


Рисунок 2.29 – Команда Extend Erase Memory: сторона хосту

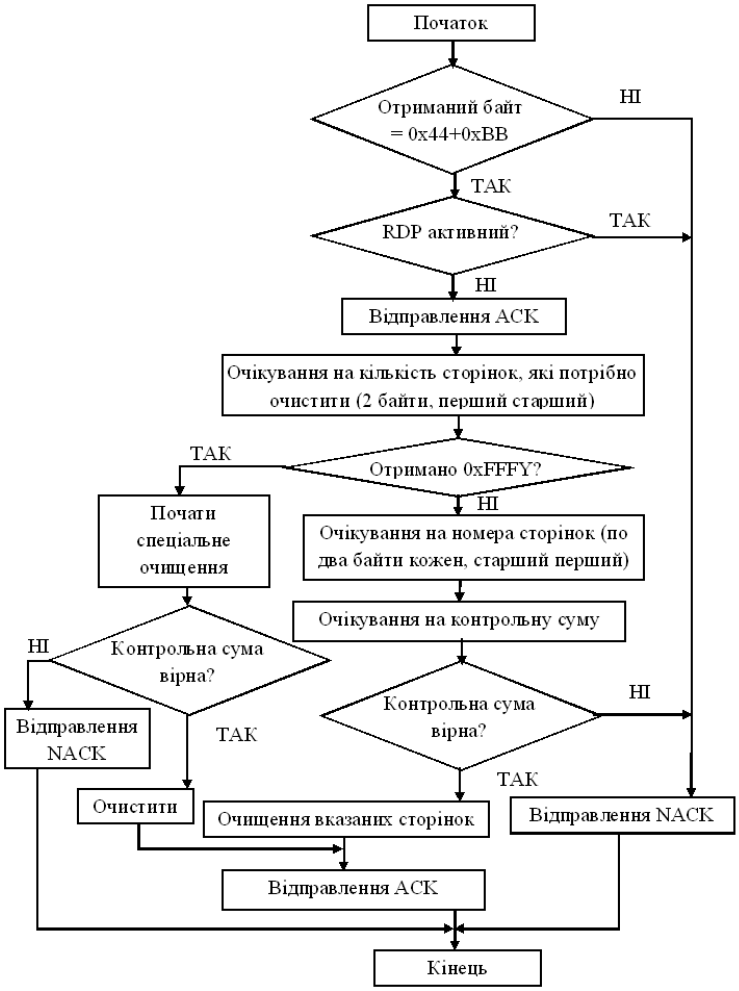


Рисунок 2.29 – Команда Extend Erase Memory: сторона мікроконтролера

Чекає на ACK; Байти 3-4: Спеціальна очистка (0xFFFF, 0xFFFE, 0xFFFD), або кількість сторінок які потрібно очистити (N+1, де: 0≤N≤максимальної кількості сторінок);

Решта байтів: контрольна сума байтів 3, 4 у випадку спеціальної очистки (0x00 якщо 0xFFFF або 0x01 якщо 0xFFFE або 0x02 якщо 0xFFFD), або (2x(N+1)) байтів (номера сторінок закодовані по два байти, перший старший) і потім контрольна сума байтів 3-4 та наступних байтів.

2.7 Висновки до розділу

Досліджено протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів. Розглянуто набір команд завантажувача їх алгоритми та принцип роботи.

Розглянуто інтерфейси програмування мікроконтролерів STM32. Мікроконтролери STM32 в залежності від моделі підтримують від двох до шести способів завантаження програми у пам’ять. Більшість із вказаних способів це використання завантажувача bootloader. Максимальна кількість можливих варіантів bootloader у одному мікроконтролері – 5, а саме через протоколи USART, I2C, SPI, CAN, DFU. Хоча не всі моделі STM32 підтримують по п’ять протоколів, кожна із них підтримує протокол USART, що і стало причиною вибору даного протоколу для реалізації пристрою.

3 СХЕМОТЕХНІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПРИСТРОЮ ДЛЯ ПРОГРАМУВАННЯ УНІВЕРСАЛЬНОГО ПРОГРАМАТОРА ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

3.1 Розробка структурної та принципової схеми пристрою

Пристрій для програмування восьми та тридцяти двох розрядних мікроконтролерів, який розроблено у магістерській кваліфікаційній роботі, складається із чотирьох структурних блоків. Структурна схема пристрою показана на рис.3.1, та наведена вдодатку Л.

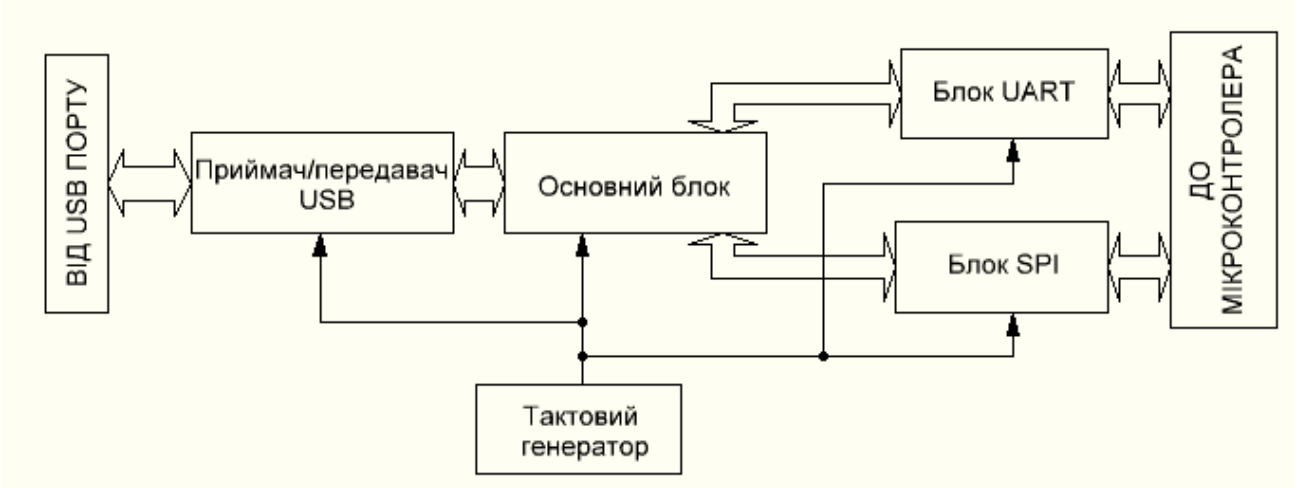


Рисунок 3.1 – Структурна схема пристрою

Як видно із рис.3.1, розроблений пристрій складається із трьох інтерфейсів обміну даними та основного блоку. Приймач/передавач USB відповідає за перетворення прийнятих від USB порту комп'ютера сигналів у зрозумілі основному блокові. Даний блок працює у режимі Full Speed. Блоки UART та SPI здійснюють передачу сигналів програмування від основного блоку до мікроконтролерів та у зворотному напрямі.

Основний блок відповідає за налаштування блоків передачі даних та за доставку даних до конкретного блоку. Також основний блок здійснює попередню обробку даних, які передаються між блоками, відкидаючи зайві біти та складаючи дані пакети, готові для відправки до наступного блоку пристрою.

Всі чотири основні структурні блоки у розробленому пристрої розташовані фізично у керуючому мікроконтролері STM32F103C8. Блоки

передачі даних реалізовані у керуючому мікроконтролері апаратно. Основним блоком у пристрої є ядро керуючого мікроконтролера. Дані між ядром та блоками передачі даних передаються за допомогою апаратних переривань.

Враховуючи, що напруга живлення мікроконтролера STM32F103C8 становить 2,0 – 3,6 В, а живлення на пристрій буде подаватись від USB порту комп'ютера, прийнято рішення використати стабілізатор напруги. Серед усіх доступних варіантів було обрано стабілізатор на напругу 3,3 В LM1117 завдяки його надійності, малим розмірам (корпус SOT-223), низькій ціні, та простій схемі увімкнення. Даний стабілізатор вмонтовано у пристрій за стандартною схемою ввімкнення, приведеною у технічному описі [26].

Для індикації роботи пристрою прийнято рішення додати у схему 3 світлодіоди: індикація напруги живлення, індикація передачі даних через інтерфейс UART/SPI та індикація прийому даних через інтерфейс UART/SPI. Світлодіоди було вибрано SMD у корпусі 0805. Так як струм споживання обраних світлодіодів становить 10 мА, за законом Ома розрахуємо опір резисторів

$$R = \frac{3{,}3\text{В}}{0{,}01\text{А}} = 330\ (\text{Ом}).$$

Так як 10 мА це максимальний струм споживання світлодіодів а максимальне навантаження на порт мікроконтролера STM32F103C8 25 мА, прийнято рішення зменшити струм світлодіодів до 6,4 мА, встановивши SMD резистори в корпусі 0805 номіналом 510 Ом. Дане рішення зменшить навантаження на порти керуючого мікроконтролера.

У пристрій вмонтовано дві кнопки, які призначені для переведення мікроконтролера в режим bootloader та перезавантаження мікроконтролера.

У якості USB роз'єму використовується miniUSB. Також на платі розташовано кварцовий резонатор на частоту 8 МГц.

3.2 Розроблення принципової схеми та друкованої плати пристрою

Принципова схема розробленого пристрою для програмування 8 та 32 розрядних мікроконтролерів показана на рис.3.2.

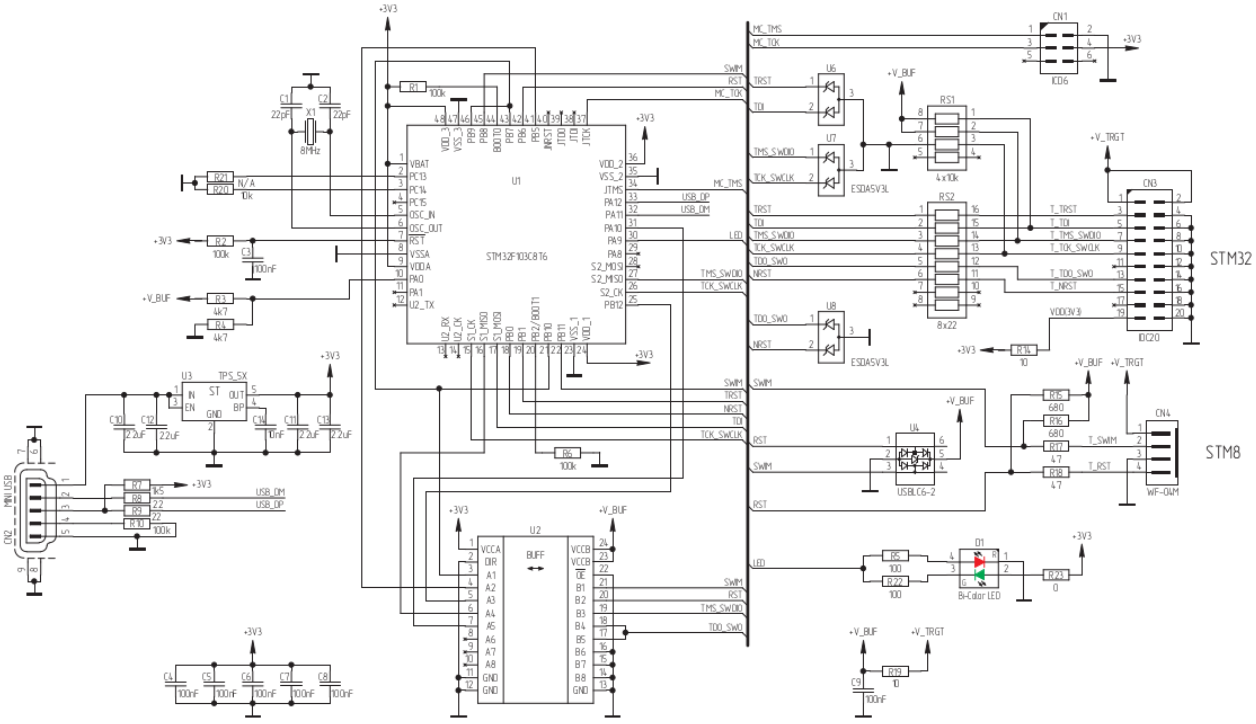


Рисунок 3.2 – Принципова схема розробленого пристрою для програмування 8 та 32 розрядних мікроконтролерів

Для розробки друкованої плати пристрою вибираємо програмний пакет DipTrace. Програмний пакет DipTrace є повнофункціональну систему для розробки принципових схем і друкованих плат. Він включає в себе чотири програмних продукта:

1. Schematic - створення принципових схем з подальшою можливістю переведення їх в друковані плати.

2. PCB Layout - проектування плат зі зручною інтерактивним та автоматичним трасуванням.

3. ComEdit - редактор корпусів для друкованої плати.

4. SchemEdit - редактор компонентів. Рисування символів електронних компонентів і зв'язування їх з корпусами.

Робота супроводжується підсвічуванням редагованих об'єктів і пов'язаних з ними для поліпшення наочного сприйняття плати або принципової схеми. Редагування одного об'єкта веде за собою відповідні зміни пов'язані з ним: система не допускає "висячих" зв'язків, оскільки при побудові відразу створюється логічна структура принципової схеми або

плати, змінювати яку можна як в наочному режимі, так і вказуючи зв'язку в табличному вигляді.

Створено мінімальну кількість режимів роботи з максимальною функціональністю кожного: так наприклад в Default режимі PCB можливе виділення, редагування та переміщення компонентів, трас, меж плати, побудова зв'язків. Перехід з будь-якого режиму побудови або редагування в Default здійснюється кліком правої кнопки миші в області побудови.

У DipTrace застосована зручна система роботи з шарами, кількість яких фактично відповідає числу сигнальних шарів друкованої плати. Провідники встановлюються в поточний сигнальний шар, при побудові трас можна переходити в інший шар. Після побудови будь-який фрагмент траси, вся траса або мережу можна переміщати в інший шар, при цьому автоматично створюються міжшарові переходи, які не існують в програмі як окремі об'єкти - їх властивості фактично є властивостями точки, яка є перехідною. Графічні елементи, текст і растрові зображення є окремими об'єктами які можуть створюватися як графіка, маркування або провідник поточного сигнального шару. Після створення можна змінювати розташування цих об'єктів.

Програма містить вбудований сітковий оптимізаційний автотрасувальник, який може створювати кілька варіантів трасування плати і вибирати кращий. Існує чотири режими настройки автотрасувальника різних по швидкості і якості. Є перевірка на помилки трасування (перетину, занадто близьке розташування провідників). Підтримується експорт в формати Gerber і N/C Drill.

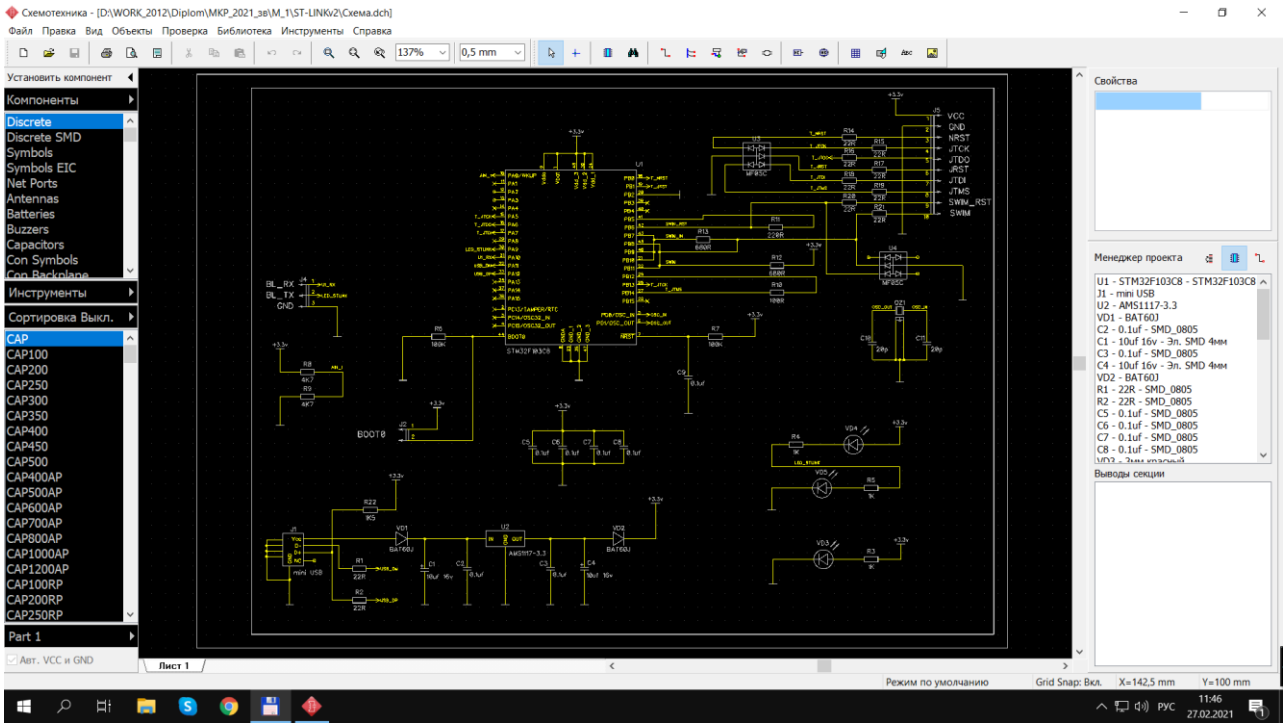


Рисунок 3.3 – Принципова схема розробленого пристрою у DipTrace

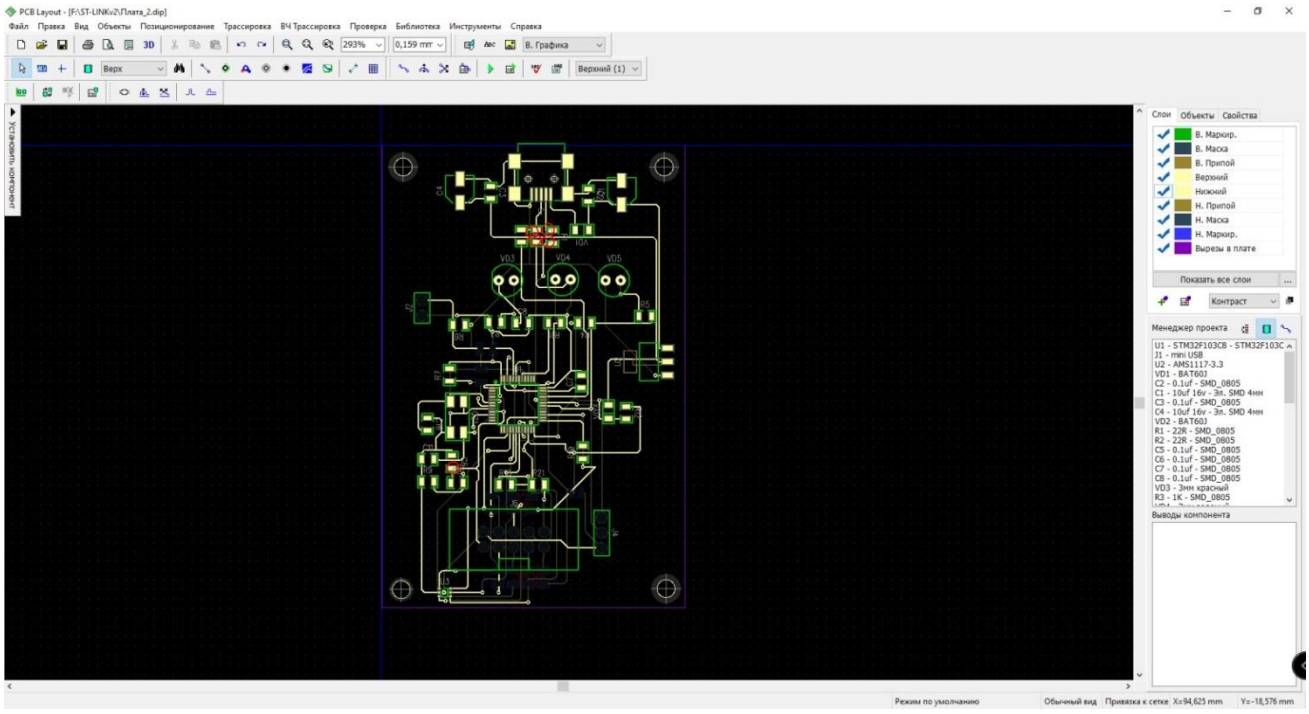


Рисунок 3.4 – Верхня сторона друкованої плати розробленого пристрою у DipTrace

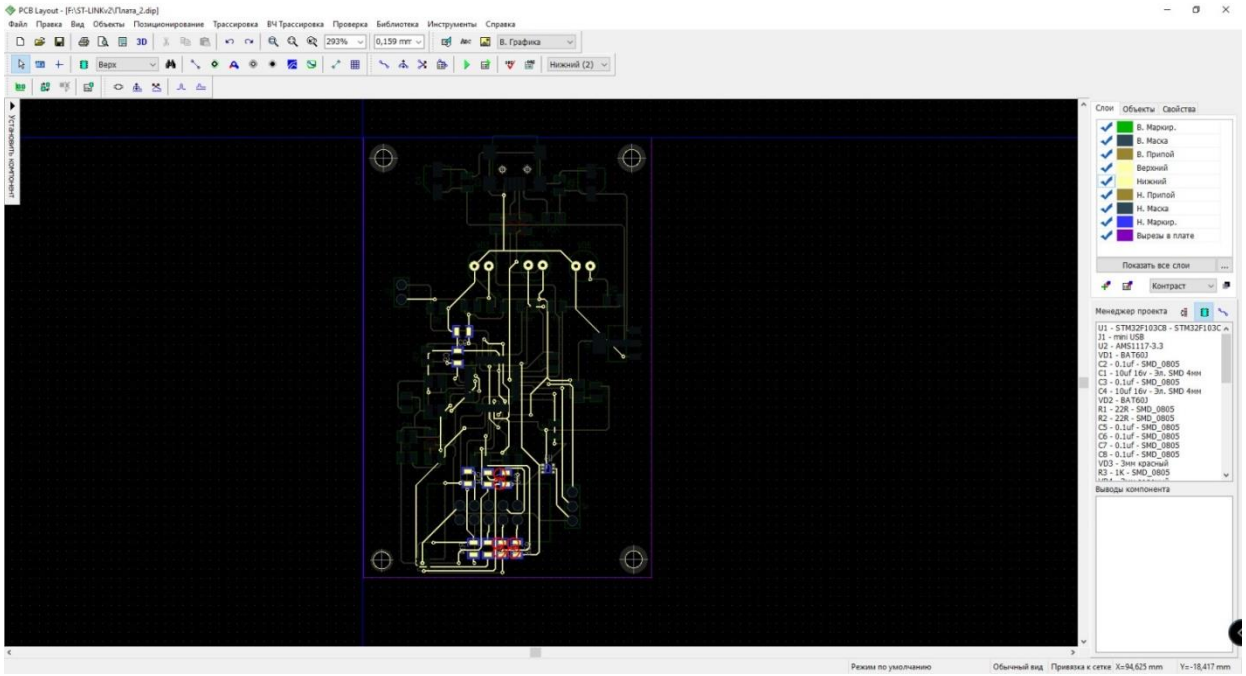


Рисунок 3.5 – Нижня сторона друкованої плати розробленого пристрою у DipTrace

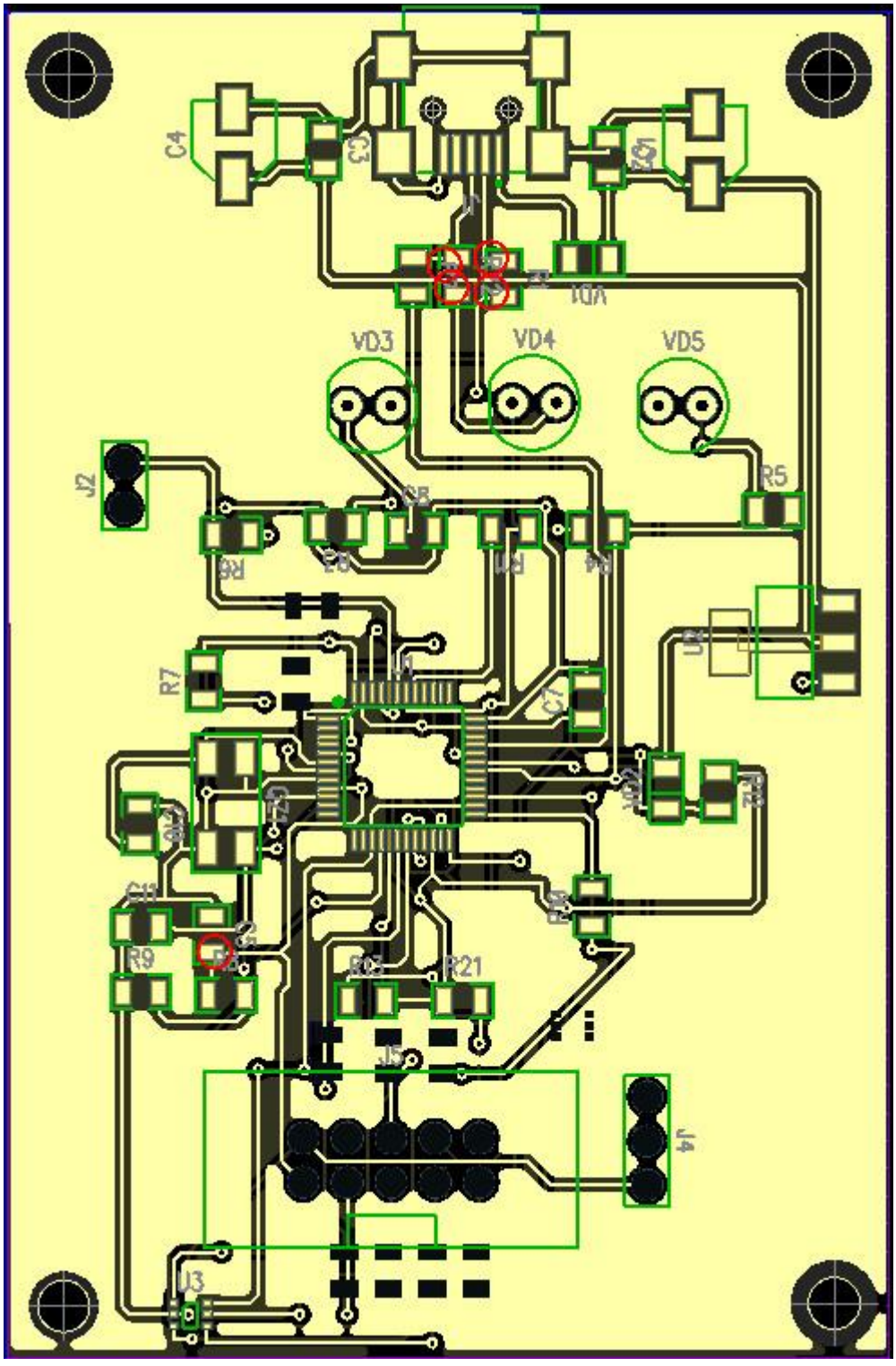


Рисунок 3.6 – Друкована плата програматора

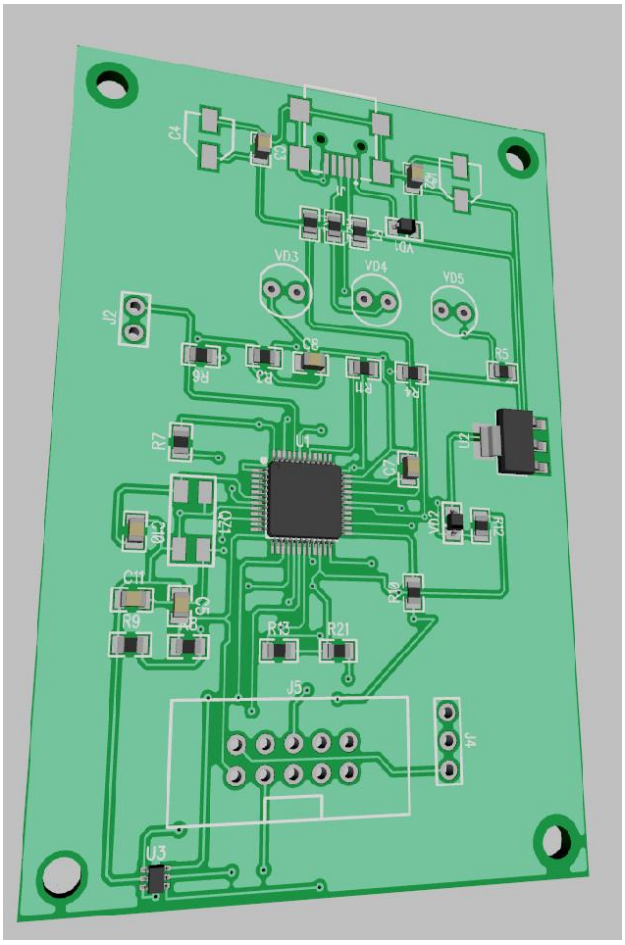


Рисунок 3.7 – 3D модель друкованої плати програматора (верхня сторона)

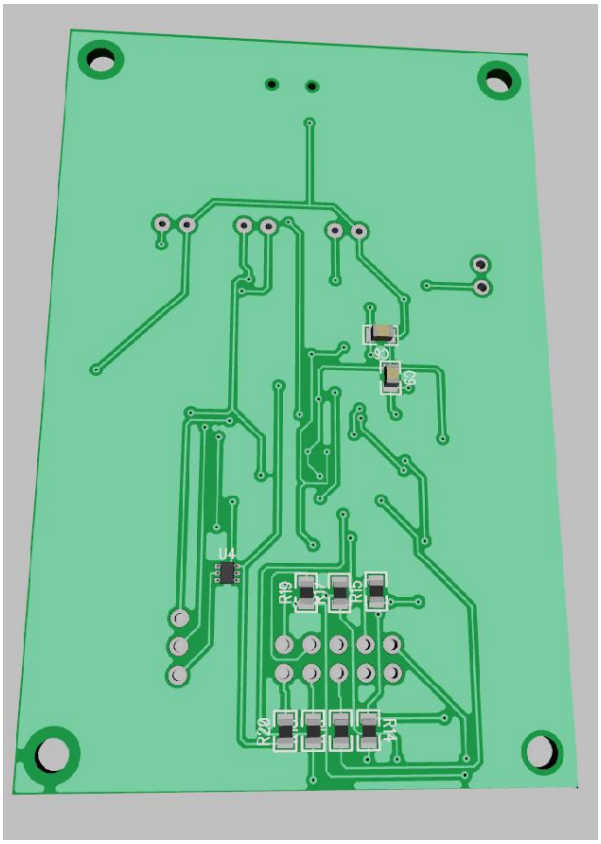


Рисунок 3.8 – 3D модель друкованої плати програматора (нижня сторона)

3.3 Програмування мікроконтроллера програматора

Розроблено програмне забезпечення мовою C++ для прошивки мікроконтролера STM32F103C8, лістинг якої наведено у додатку Б. Розглянемо алгоритм прошивання мікроконтролера STM32F103C8, основного елемента розробленого пристрою.

Біля основного роз'єму є три піна, це Rx, Tx і GND. До них потрібно підключити COM-порт. Для програмування мікроконтролера програматора потрібен тільки COM-порт. Використаємо фізичний перехідник на TTL. Живлення потрібно подати 3,3 В на 1 пін основного роз'єму. Найкраще живлення програма тора здійснювати від USB роз’єму. Також можна здійснювати живлення через USB-хаб з можливістю підключення до зовнішнього джерела живлення.

Після того як підключили живлення, якщо все спаяно нормально і без помилок, мікроконтролер повинен бути готовий до прошивання. Далі запускаємо програму Flash Loader Demonstrator (рис.3.9).

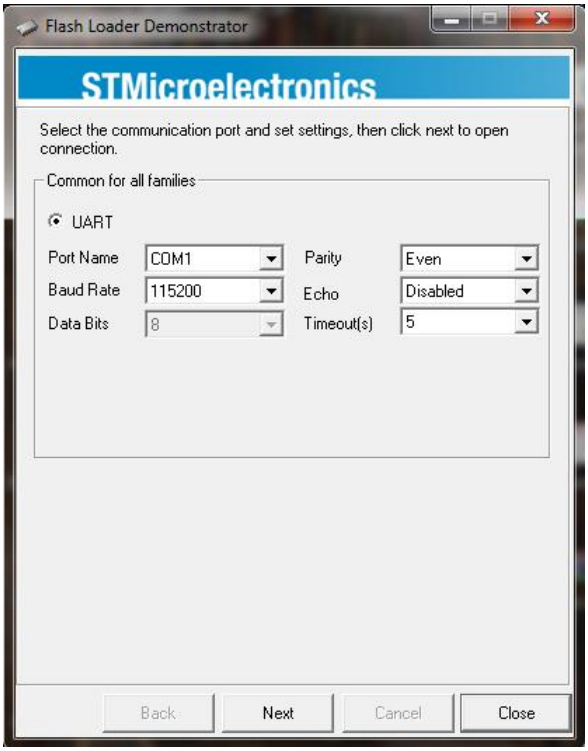


Рисунок 3.9 – Зовнішній вигляд програми Flash Loader Demonstrator

Далі вибираємо віртуальний COM порт і швидкість передачі даних. Рекомендується вибирати швидкість 115200 біт/с, натискаємо кнопку Next.

Повинне з'явитися наступне вікно. Якщо світлофор буде зелений, значить все зроблено правильно (рис.3.10). В іншому випадку шукаємо помилку в схемі.

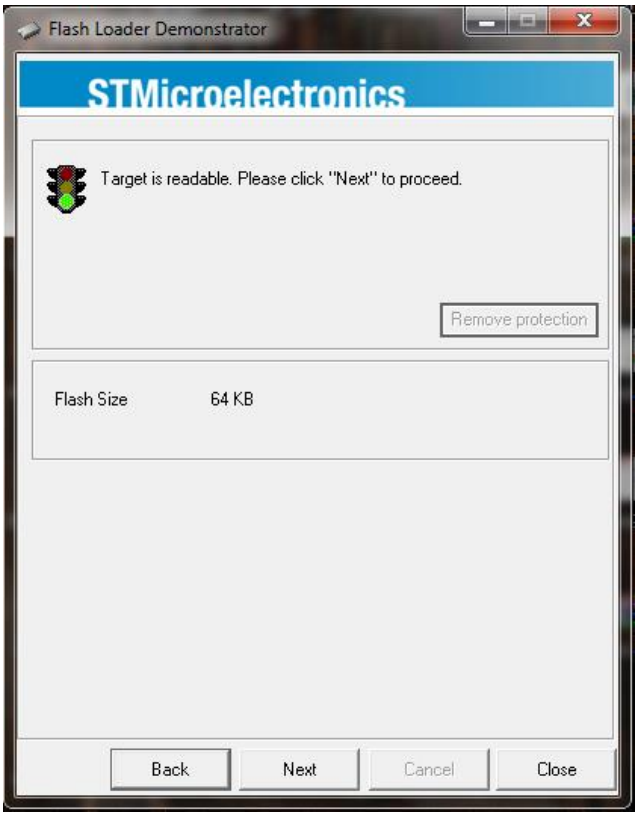


Рисунок 3.10

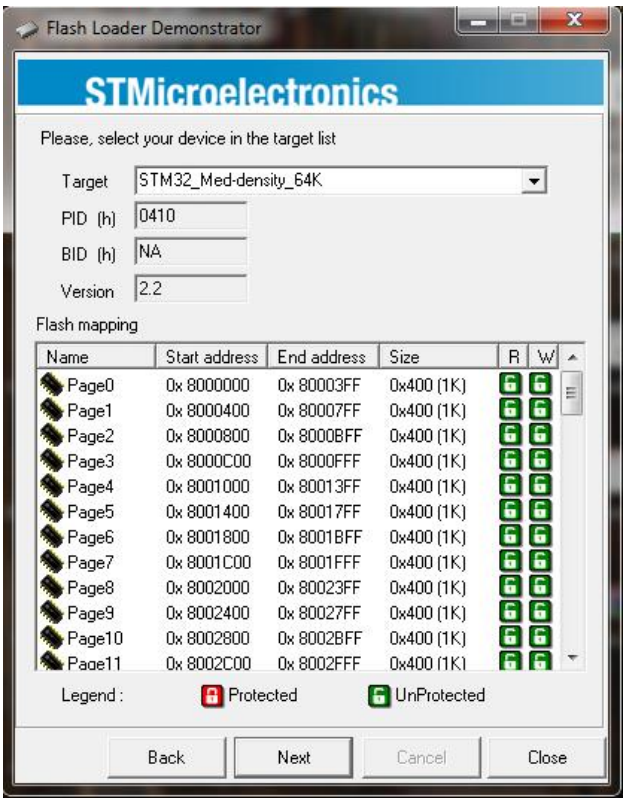


Рисунок 3.11

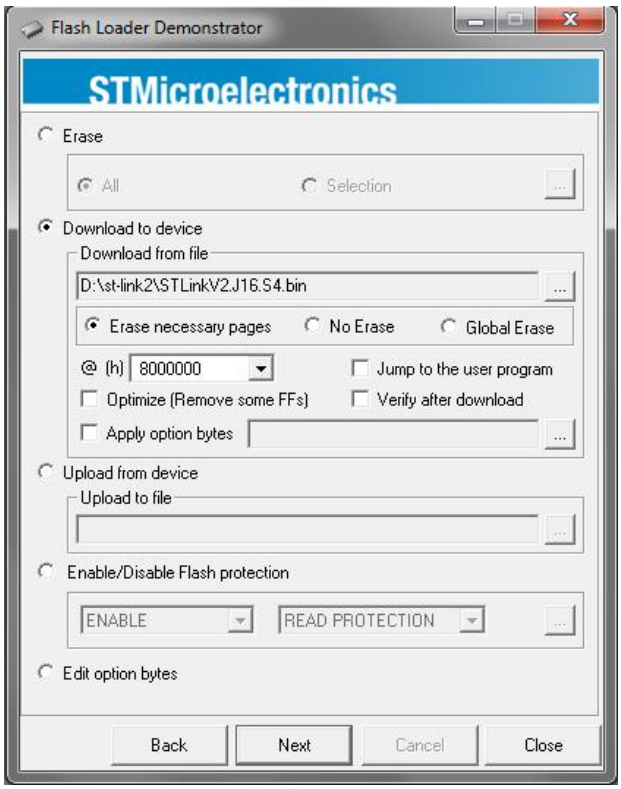


Рисунок 3.12

У цьому вікні потрібно вказати завантажуваний файл. Вибираємо з архіву файл STLinkV2.J16.S4 і тиснемо Next. Після завантаження вікно буде виглядати наступним чином.

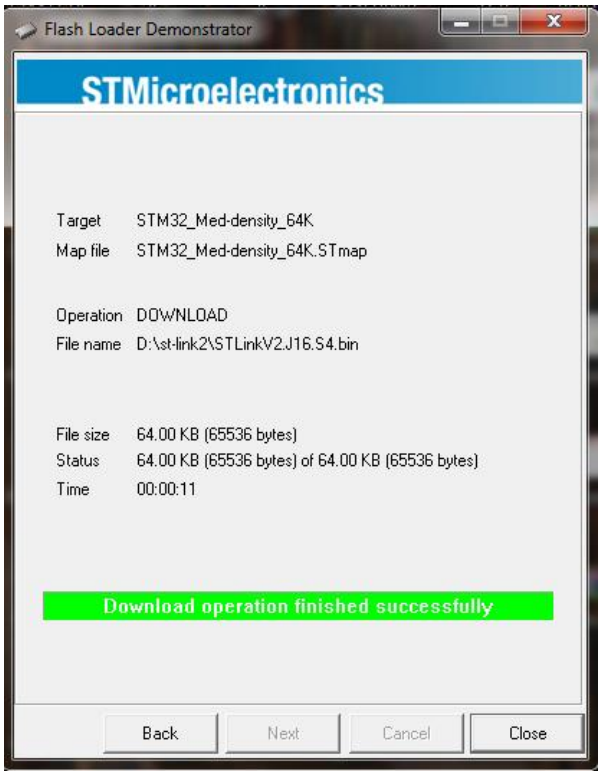


Рисунок 3.13

Після цього знімаємо джампер і відключаємо COM-порт. Наступним кроком встановлюємо драйвер st-link_v2_usbdriver. Також можна завантажити більш новіший драйвер з сайту STMicroelektronics. Після установки драйвера підключаємо програматор до USB роз'єму комп'ютера. Якщо все до цього моменту було зроблено правильно, Windows побачить розроблений пристрій і встановить для нього драйвер.

Рисунок 3.14

3.4 Висновки до розділу

Розроблено пристрій для програмування восьми та тридцяти двох розрядних мікроконтролерів, він складається із чотирьох структурних блоків. Розроблений пристрій складається із трьох інтерфейсів обміну даними та основного блоку. Приймач/передавач USB відповідає за перетворення прийнятих від USB порту комп'ютера сигналів у зрозумілі основному блокові. Даний блок працює у режимі Full Speed. Блоки UART та SPI здійснюють передачу сигналів програмування від основного блоку до мікроконтролерів та у зворотному напрямі.

Розроблено структурну та принципову схему пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів.

Розроблено друковану плату пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів в пакеті програм DipTrace, а також розроблено 3D модель друкованої плати з елементами.

Розглянуто алгоритм прошивання мікроконтролера STM32F103C8, як основного елемента розробленого пристрою.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу результатів НДДКР. В результаті оцінювання можна зробити висновок щодо напрямів (особливостей) організації подальшого впровадження результатів з врахуванням встановленого рейтингу.
Рекомендується здійснювати оцінювання комерційного потенціалу розробки за 12-ма критеріями, наведеними в таблиці 4.1. [39]

Таблиця 4.1 - Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність проду-кту в реаль-них умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки зведемо до таблиці 4.2.

Таблиця 4.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	експерт		
	1	2	3
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	2	2	3
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	2	1	2
4. Ринкові переваги (технічні властивості)	1	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	3	3
6. Ринкові перспективи (розмір ринку)	3	2	2
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	2	3	2
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових	3	3	4
11. Практична здійсненність (термін реалізації)	2	2	2
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	29	29	32
Середньоарифметична сума балів СБ	**30,0**		

За даними таблиці 4.2 зробимо висновок щодо рівня комерційного потенціалу дослідження. При цьому доцільно користуватися рекомендаціями, наведеними в таблиці 4.3. [39]

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі	Рівень комерційного потенціалу розробки
0 - 10	Низький
11 - 20	Нижче середнього
21 - 30	Середній
31 - 40	Вище середнього
41 - 48	Високий

Згідно проведених досліджень рівень комерційного потенціалу розробки становить 30,0 бала, що, згідно таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки середній).

4.2 Розрахунок витрат на проведення НДДКР з дослідження універсального програматора для сучасних мікроконтролерів

В техніко-економічному обґрунтуванні представленому в першому розділі даної магістерської кваліфікаційної роботи було приблизно обґрунтовано доцільність проведення НДДКР. Тому в даному розділі будуть проведені більш детальні розрахунки витрат на проведення НДДКР стосовно дослідження універсального програматора для сучасних мікроконтролерів.

Для економічного розрахунку проведення НДДКР потрібно скласти кошторис витрат, який передбачає розрахунок визначених основних статей витрат.

Основна заробітна плата дослідників та розробників, яка розраховується за формулою [39]

$$З_о = \frac{M}{T_р} \cdot t \quad , \tag{4.1}$$

де M – місячний посадовий оклад конкретного розробника (дослідника), грн.;
$T_р$ – число робочих днів в місяці, 21 дн;
t – число днів роботи розробника (дослідника).
Проведені розрахунки зводимо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата дослідників та розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
1. Керівник проекту	11150,00	530,95	21	11150,00
2. Інженер-конструктор радіоелектронної апаратури	10250,00	488,10	12	5857,14
3. Науковий співробітник	10300,00	490,48	12	5885,71
4. Лаборант	6100,00	290,48	8	2323,81
Разом				25216,67

Витрати на основну заробітну плату працівників (Зр), що здійснюють підготовку робочих місць необхідних для дослідження універсального програматора для сучасних мікроконтролерів, підготовку та формування баз даних, підготовку та монтаж обладнання, макетів, виготовлення дослідних зразків тощо, розраховуються на основі норм часу, які необхідні для виконання даної роботи, за формулою [39]

$$З_р = \sum_{1}^{n} t_i \cdot С_i \cdot К_с \quad , \tag{4.2}$$

де t_i - норма часу (трудомісткість) на виконання конкретної роботи, годин;
n - число робіт по видах та розрядах;
$К_с$ - коефіцієнт співвідношень, який установлений в даний час Генеральною тарифною угодою між Урядом України і профспілками, $К_с$ = 1,5;
Ci - погодинна тарифна ставка робітника відповідного розряду, який виконує відповідну роботу, грн./год.
$С_i$ визначається за формулою [39]

$$С_i = \frac{М_м \cdot К_i}{Т_р \cdot Т_{зм}}, \tag{4.3}$$

де, $М_н$ – прожитковий мінімум працездатної особи, грн., $М_н$ = 2270,00 грн.;
$К_i$ - тарифний коефіцієнт робітника відповідного розряду;
$Т_р$ - число робочих днів в місяці, $Т_р$ = 21 дн;

$T_{зм}$ - тривалість зміни, $Т_{зм}$ = 8 годин.
Проведені розрахунки винесемо до таблиці 4.5.

Таблиця 4.5 – Витрати на основну заробітну плату працівників

Найменування робіт	Трудомісткість нормо-годин	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати, грн.
1. Розміщення обладнання	5,00	2	1,1	22,29	111,47
2. Контроль компонентів РЕА	2,00	5	1,7	34,46	68,91
3. Монтаж обладанання	3,00	3	1,35	27,36	82,08
4. Монтаж досліджуваних блоків	4,00	5	1,7	34,46	137,82
5. Налаштування системи	2,00	3	1,35	27,36	54,72
6. Підготовка робочих місць проектантів	3,00	2	1,1	22,29	66,88
Разом					521,90

Додаткова заробітна плата розробників, дослідників та працівників, які приймали участь в дослідженнях та розробці НДДКР розраховується як 11% від основної заробітної плати розробників та працівників

$$З_д = З_о *\cdot 11 / 100\% \quad (4.4)$$

$З_д$ = (25216,67+ 521,90) *· 11 / 100 % = 2831,24 (грн.)

Нарахування на заробітну плату дослідників та працівників.

Згідно діючого законодавства нарахування на заробітну плату складають 22% від суми основної та додаткової заробітної плати

$$Н_з = (З_о + З_д) \cdot * 22\% / 100\% \quad (4.5)$$

$Н_з$ = (25216,67+ 521,90+ 2831,24) * 22% / 100% = 6285,36 (грн.)

Витрати на матеріали на даному етапі проведення НДДКР пов’язані з використанням моделей елементів та моделювання роботи і досліджень за допомогою комп’ютерної техніки та створення експериментальних блоків і компонентів, тому дані витрати формуються на основі як офісних витратних матеріалів так і обмеженого переліку матеріалів.

Витрати на матеріали, що були використані при проведенні досліджень, розраховуються по кожному виду матеріалів за формулою [39]

$$M = \sum_{1}^{n} H_i \cdot Ц_i \cdot K_i \quad , \tag{4.6}$$

де, - H_i - витрати матеріалу *i*-го найменування, кг;
$Ц_i$ - вартість матеріалу *i*-го найменування, грн./кг.;
K_i - коефіцієнт транспортних витрат, $К_і$ = 1,1;
n - кількість видів матеріалів,
Проведені розрахунки зводимо до таблиці 4.6.

Таблиця 4.6 – Витрати на основні матеріали

Найменування матеріалу, марка, тип, сорт	Одиниця виміру	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
1. Папір канцелярський офісний	уп.	90,00	4	360,00
2. Компакт-диски (CD)	шт.	10,00	2	20,00
3. Офісна тека ЕКОС	шт.	66,00	3	198,00
4. Канцелярські товари (ручки, файли, бокси)	компл.	180,00	4	720,00
5. Тонер для принтера	кг	5400,00	0,02	108,00
6. Плата друкована	шт.	150,00	1	150,00
Всього				1556,00

З врахуванням транспортних витрат вартість матеріалів складе

М = 1556,00 * 1,11 = 1727,16 грн.

Витрати на комплектуючі (основне обладнання, емулятори, моделі, комплектуючі макетів), що були використані при дослідженні універсального програматора для сучасних мікроконтролерів, розраховуються за формулою

$$Н = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \ , \qquad (4.7)$$

де: *Нi* - кількість комплектуючих *i*-го виду, шт.;
Цi - покупна ціна комплектуючих *i*-го виду, грн.;
$К_i$ - коефіцієнт транспортних витрат, $К_i = 1,11$;
n - кількість видів матеріалів.
Проведені розрахунки зводимо до таблиці 4.7.

Таблиця 4.7 – Витрати на комплектуючі для формування компонентів для НДДКР

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн.	Сума, грн.
Корпус програматора	1	43,00	43,00
Інтерфейс АХ-120	1	55,00	55,00
Блок живлення	1	111,00	111,00
Інші комплектуючі	1	250,00	250,00
Всього			459,00

Витрати на комплектуючі з урахуванням транспортних витрат складають

$$Н = 459,00 \cdot 1,11 = 509,49 \ (грн.)$$

Амортизація обладнання для проведення досліджень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню можуть бути розраховані з використанням прямолінійного методу амортизації за формулою

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, приміщень тощо, які використовувались для розробки нового технічного рішення, грн.;
$t_{вик}$ – термін використання обладнання, приміщень під час розробки, місяців;
$Т_в$ – строк корисного використання обладнання, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці 4.8.

Таблиця 4.8 - Величина амортизаційних відрахувань

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, міс.	Величина амортиза-ційних відрахувань, грн
Комп'ютеризована система проектування	20000,00	4	1	416,67
Програмне забезпечення	9000,00	3	1	250,00
Офісна оргтехніка	12000,00	5	1	200,00
Автоматизоване робоче місце проектувальника	8500,00	5	1	141,67
Приміщення лабораторії	200000,00	20	1	833,33
Всього				1841,67

Витрати на силову електроенергію на проведення досліджень розраховують за формулою [39]

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.9)$$

де, $В$ – вартість 1 кВт-години електроенергії, $В$ = 3,5 грн./кВт –година;
$П$ – встановлена потужність обладнання, кВт.;
$Ф$ – фактична кількість годин роботи обладнання, годин. ;
$К_п$ – коефіцієнт використання потужності.
Всі проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Витрати на електроенергію при проведенні досліджень

Найменування обладнання	Кількість годин роботи обладнання, год.	Встановлена потужність, кВт	Коефіцієнт використання потужності	Величина оплати
Комп'ютеризована система проектування	160	0,86	0,95	457,52
Офісна оргтехніка	15	1,25	0,95	62,34
Автоматизоване робоче місце	90	0,12	0,95	35,91
Дослідна система	90	0,08	0,95	23,94
Всього				579,71

Інші витрати охоплюють: загальновиробничі витрати, адміністративні витрати, витрати на відрядження, матеріали, окремі непередбачені витрати, зв’язок, витрати на інтернет-послуги тощо.
Інші витрати доцільно приймати як 200...300% від суми основної заробітної плати дослідників та робітників. Величина інших витрат складе

$$I = (25216,67 + 521,90) * 200\% / 100\% = 51477,13 \text{ (грн.)}$$

Загальні витрати на проведення науково-дослідної роботи.
Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи

$$B=25216,67+521,90+2831,24+6285,36+1556,00+509,49+1841,67+579,71+ +51477,13=90819,16 \text{ (грн.)}$$

Загальна (повна) вартість всієї НДДКР визначається за формулою

$$B_{заг} = \frac{B}{\alpha}, \qquad (4.10)$$

де α - частка витрат, які безпосередньо здійснює виконавець даної НДДКР, у відносних одиницях.

$$B_{заг} = \frac{B}{\alpha} = \frac{90819,16}{1} = 90819,16 \text{, грн.}$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної НДДКР здійснюється за формулою

$$ЗВ = \frac{B_{заг}}{\beta}, \qquad (4.11)$$

де β - коефіцієнт, який характеризує етап (стадію) виконання даної НДДКР (від 0,1… до 0,9)

$$ЗВ = \frac{B_{заг}}{\beta} = \frac{90819,16}{0,9} = 100910,00 \text{, грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство (організація) від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства (організації). Зростання чистого прибутку ми можемо оцінити у теперішній вартості грошей.

Саме зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності та виплатити кредити (якщо вони потрібні для впровадження результатів розробки).

При проведенні даної розробки не можливо прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою

$$\Delta П_i = \sum (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}) \qquad (4.12)$$

де $\Delta Ц_о$ - покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N - основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN - покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_о$ - основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n - кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ - коефіцієнт, який враховує сплату податку на додану вартість. У 2021 р. ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0,8333$.

ρ - коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $p = 0,2...0,3$;

υ - ставка податку на прибуток. У 2021 році $\upsilon = 18\%$.

В результаті впровадження результатів наукової розробки покращується якість нашої розробки, що дозволяє підвищити ціну її реалізації на 120 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року - на 110 шт., протягом другого року - ще на 140 шт., протягом третього року - ще на 120 шт., а протягом четвертого року – на 100 шт. Орієнтовно: реалізація аналогічного пристрою до впровадження результатів наукової розробки складала 4000 шт., а її ціна - 990 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства протягом першого року складе

$$\Delta П_1 = \left[120 \cdot 4000 + (990 + 120) \cdot 110\right] \cdot 0,8333 \cdot 0,25 \cdot (1 - \frac{18}{100}) = 102855,00 \text{ грн.}$$

Збільшення чистого прибутку підприємства протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе

$$\Delta П_2 = \left[120 \cdot 4000 + (990 + 120) \cdot (110 + 140)\right] \cdot 0,8333 \cdot 0,25 \cdot (1 - \frac{18}{100}) = 129401,00\,(\text{грн}).$$

Збільшення чистого прибутку підприємства протягом третього року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе

$$\Delta П_3 = \left[120 \cdot 4000 + (990 + 120) \cdot (110 + 140 + 120)\right] \cdot 0,8333 \cdot 0,25 \cdot (1 - \frac{18}{100}) = 152155,00\,(\text{грн}).$$

Збільшення чистого прибутку підприємства протягом четвертого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе

$$\Delta П_4 = \left[120 \cdot 4000 + (990 + 120) \cdot (110 + 140 + 120 + 100)\right] \cdot 0,8333 \cdot$$
$$\cdot 0,25 \cdot (1 - \frac{18}{100}) = 171117,00\,(грн).$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1. Розраховують теперішню вартість інвестицій, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ=**100910,00** грн. на виконання та впровадження результатів НДДКР.
2. Розраховують очікуване збільшення прибутку, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження.
3. Для спрощення подальших розрахунків будують вісь часу, на яку наносять всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуються у ті терміни, коли вони здійснюються.

Проведемо відповідні розрахунки.

У першому році підприємство отримає збільшення чистого прибутку на **102855** грн відносно базового року.
У другому році - збільшення чистого прибутку на **129400** грн (відносно базового року).
У третьому році - збільшення чистого прибутку на **152155** грн (відносно базового року),
У четвертому - на **171117** грн (відносно базового року).
Тоді рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис 4.1.

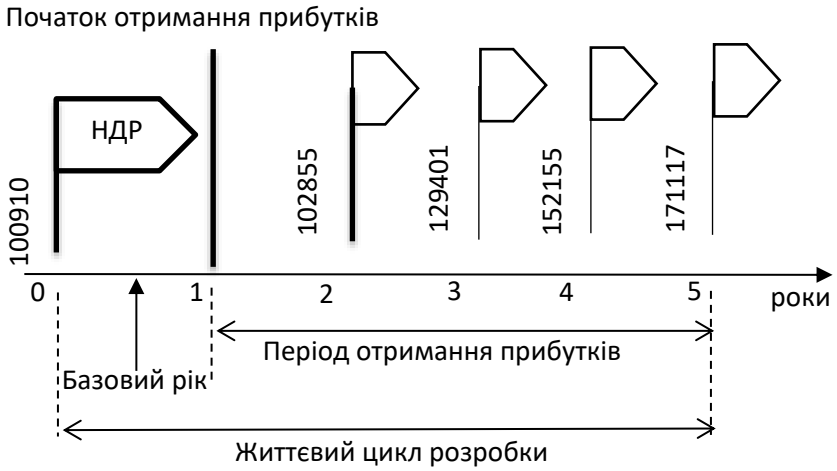


Рисунок 4.1 - Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4. Розраховують абсолютну ефективність вкладених інвестицій $E_{абс}$. Для цього використаємо формулу [39-41]

$$E_{абс} = (ПП - PV), \qquad (4.13)$$

де *ПП* - приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;
PV - теперішня вартість інвестицій $PV = ЗВ$, грн.
У свою чергу, приведена вартість всіх чистих прибутків *ПП* розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.14)$$

де *ΔП* - збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;
Т - період часу, протягом якого виявляються результати впровадженої НДДКР, роки;
τ - ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,12;
t - період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{102855}{(1+0,12)^2} + \frac{129401}{(1+0,12)^3} + \frac{152155}{(1+0,12)^4} + \frac{171117}{(1+0,12)^5} =$$
$$= 81995 + 92105 + 96697 + 97096 = 367893\ (грн).$$

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Отримаємо:

$$E_{абс} = (367893 - 100910) = 266984\ (грн).$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.15)$$

де $E_{абс}$ - абсолютна ефективність вкладених інвестицій, грн;
PV - теперішня вартість інвестицій *PV = ЗВ*, грн;
Т - життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 = \sqrt[5]{1 + \frac{266984}{100910}} - 1 = 0,30$$

Розраховану величину $E_в$ порівнюємо з мінімальною ставкою дисконтування, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування визначається за формулою

$$\tau = d + f, \tag{4.16}$$

де d - середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні (0,08...0,12);
f - показник, що характеризує ризикованість вкладень (0,05...0,1).

$$\tau = d + f = 0,1 + 0,1 = 0,2.$$

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою

$$T_{ок} = \frac{1}{E_в}.$$

$$T_{ок} = \frac{1}{E_в} = \frac{1}{0,30} = 3,39 \text{ року}.$$

Якщо $T_{ок}$ < 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

4.5 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки становить 30,0 балів, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки середній).

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство (організація) від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства (організації). Зростання чистого прибутку становить 367893,00 грн.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Абсолютна ефективність вкладених коштів для даної розробки складе 266984,00 грн., а відносна ефективність забезпечить прибутковість в межах 30%, що більше мінімальної яка складає 20%.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $Т_{ок}$=3,39 року, що менше нормативного терміну.

Враховуючи наведені показники діяльності можна зробити висновок про доцільність проведення науково-дослідної роботи.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Праця є цілеспрямованою діяльністю людини, що орієнтована на створення за допомогою знарядь виробництва матеріальних і духовних цінностей, необхідних для життя людей. Сучасне виробництво базується на застосуванні складної техніки, машин, устаткування та певних технологій, отже, виробниче середовище стає агресивнішим стосовно працівника, збільшується ризик отримати травму або захворювання в процесі його взаємодії із засобами праці і компонентами виробництва, а тому, він потребує відповідних заходів з охорони життя та здоров'я. Трудова діяльність є не тільки найбільш відповідальною сферою життя суспільства, а й запорукою самого існування людства. Праця є основою життєдіяльності людини як індивіда і члена суспільства, вона безпосередньо впливає на формування і розвиток суспільних відносин. Завдяки цьому найважливішим завданням будь-якої демократичної держави є створення найбільш сприятливих умов для плідної праці та її охорони.

Розробка універсального програматора для сучасних мікроконтролерів відбувалася в приміщені, яке обладнане комп'ютеризованими робочими місцями. В приміщенні, згідно ГОСТ 12.0.003-74, можуть бути наявні вплив такі небезпечні та шкідливі виробничі фактори:

1. Фізичні: підвищена запиленість та загазованість повітря робочої зони; підвищений рівень шуму на робочому місці; підвищена чи понижена вологість повітря; підвищений рівень статичної електрики; підвищений рівень електромагнітного випромінювання; недостатня освітленість робочої зони.

2. Психофізіологічні: розумове перевантаження; перенапруга аналізаторів; статичне перевантаження.

Відповідно до визначених факторів формуємо рішення щодо безпечного виконання роботи, а також з гігієни праці та виробничої санітарії, які забезпечують безпечні та комфортні виробничі умови.

5.1 Технічні рішення щодо безпечного виконання роботи

5.1.1 Обладнання приміщення та робочого місця

Загальні вимоги до умов праці на підприємствах встановлено законодавством про працю. Відповідно до ч. 1 ст. 6 Закону України «Про охорону праці» від 14.10.92 р. № 2694-XII умови праці на робочому місці,

безпека технологічних процесів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам законодавства.

Більшість нормативів щодо умов праці працівників, які використовують ПК та відповідне програмне забезпечення для виконання роботи, встановлено на рівні державних стандартів. Основними з них є:

- Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 2.3.6.037-99, затверджені постановою Головного державного санітарного лікаря України від 01.12.99 р. № 37;
- Державні санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджені постановою Головного державного санітарного лікаря України від 01.12.99 р. № 42;
- Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджені постановою Головного державного санітарного лікаря України від 10.12.98 р. № 7;
- Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників, затверджені наказом МНС від 25.01.2012 р. № 67.

Організація робочого місця розробника універсального програматора для сучасних мікроконтролерів повинна забезпечувати відповідність всіх його елементів і їхнього розташування ергономічним вимогам та особливостям трудової діяльності.

Приміщення в якому відбувалася розробка універсального програматора для сучасних мікроконтролерів (рис.5.1) є має загальну площу 15,75 $м^2$ і об'єм – 50,4 $м^3$.

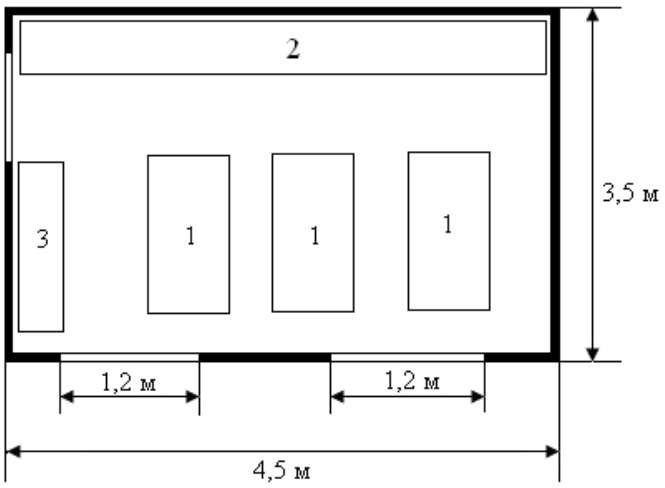


Рисунок 5.1 - Схема приміщення

Умовні позначення: 1 – робочий стіл; 2 – меблі для документації; 3 – стіл для копіювальної техніки.

В приміщенні працює 3 працівника. На одного працівника припадає 5,25 $м^2$ площі та 16,8 $м^3$ об'єму повітря робочої зони. Вимоги НПАОП 0.00-1.28-10 [42] щодо вказаних параметрів не дотримані.

Розміри робочого столу: висота – 725 мм, ширина – 600 -1400 мм, глибина – 800 – 1000 мм, що відповідає вимогам ДСанПіН 3.3.2.007-98. Робочий стіл для ПК має простір для ніг висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм, на рівні витягнутої ноги – не менше 650 мм.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці забезпечує добру видимість монітору, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті 900 – 1300 мм, по глибині 400 – 500 мм.

Оскільки в приміщенні лише три робочі місця, обладнані ПК, тому необхідності відділяти їх одне від одного перегородками висотою 1,5 – 2 м немає.

Мінімальні вимоги безпеки під час роботи:

– щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;

– після закінчення роботи екранні пристрої слід відключати від електричної мережі;

– у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;

– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

5.1.2 Електробезпека приміщення

В досліджуваному приміщенні лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ й устаткування для обслуговування, ремонту й налагодження ЕОМ виконана як окрема групова трипровідна мережа, шляхом прокладання фазових, нульових робочих і нульового захисного провідників. Нульовий захисний провідник використається для заземлення електроприймачів.

Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники відповідають номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту.

Персональні комп’ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту відповідають класу зони, мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію.

Приміщення, в якому відбувалося дослідження за класом електробезпеки – це приміщення без підвищеної небезпеки, оскільки фактори підвищеної та особливої небезпеки на робочому місці відсутні.

Персональні комп’ютери, периферійні пристрої повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки, окрім контактів фазового та нульового робочого провідників, повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання

контактів фазових провідників з контактами нульового захисного провідника. Неприпустимим є підключення комп’ютерів, периферійних пристроїв до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв [43].

Є неприпустимими**:**

– експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією;

– застосування саморобних подовжувачів, застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;

– користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;

– використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів.

5.2 Технічні рішення з гігієни праці та виробничої санітарії

5.2.1 Мікроклімат

Суттєвий вплив на стан організму працівника, його працездатність здійснює мікроклімат (метеорологічні умови) у виробничих приміщеннях, під якими розуміють клімат внутрішнього середовища цих приміщень, що визначається діючою на організм людини сукупністю температури, вологості, руху повітря та випромінювання від нагрітих або охолоджених поверхонь. Мікроклімат виробничих приміщень впливає на стан організму працюючих та їх теплообмін з навколишнім середовищем.

Робота розробника універсального програматора для сучасних мікроконтролерів за енерговитратами відноситься до категорії 1а.

Допустимі параметри мікроклімату для категорії 1а наведені в табл.5.1.

Таблиця 5.1 – Параметри мікроклімату [44]

Період року	Допустимі		
	t,°C	W, %	V, м/с
Теплий	22-28	55	0,1-0,2
Холодний	21-25	75	0,1

Для підтримки допустимих значень мікроклімату та концентрації позитивних і негативних іонів необхідно передбачати установки або прилади зволоження та/або штучної іонізації, кондиціювання повітря. В Україні відсутні затверджені на законодавчому рівні гранично допустимі норми вмісту вуглекислого газу в повітрі для житлових, офісних та громадських споруд. Під час перевищення припустимих значень робочий день співробітників повинен бути скорочений мінімум на 10 % у відповідності до нормативних вимог.

Для підтримки оптимального рівня мікроклімату в приміщенні передбачено систему кондиціонування повітря з індивідуальним регулюванням температури, систему центрального опалення та систематичне вологе прибирання приміщення.

5.2.2 Склад повітря робочої зони

Шкідливі речовини, що потрапили в організм людини спричинюють порушення здоров’я лише в тому випадку, коли їхня кількість в повітрі перевищує граничну для кожної речовини величину. В приміщенні, де здійснюється розробка універсального програматора для сучасних мікроконтролерів, можливими шкідливими речовинами у повітрі є фенол, пил, озон та вуглекислий газ. Джерелами цих речовин є офісна техніка. Пил потрапляє у приміщення ззовні. ГДК шкідливих речовин, згідно ДСН 3.3.6.042-99, які знаходяться в досліджуваному приміщені, наведені в таблиці 5.2.

Таблиця 5.2 – ГДК шкідливих речовин у повітрі [45]

Назва речовини	ГДК, мг/м^3		Клас небезпечності
	Максимально разова	Середньо добова	
Фенол	0,01	0,01	3
Пил нетоксичний	0,5	0,15	4
Озон	0,16	0,03	4
Вуглекислий газ	3	1	4

Параметри іонного складу повітря на робочому місці, що обладнане ПК, повинні відповідати допустимим нормам (табл.5.3).

Таблиця 5.3 – Рівні іонізації повітря приміщень при роботі на ПК [46]

Рівні	Кількість іонів в 1 $см^3$	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально необхідні	50000	50000

Забезпечення складу повітря робочої зони здійснюється за допомогою системи кондиціювання, регулярного провітрювання, та вологого прибирання.

5.2.3 Виробниче освітлення

Як відомо, тривала робота за комп'ютером та з документами при недостатньому рівні освітленості може призвести до значного перенапруження зору, тому вимоги до освітлення є досить важливими. Додатково, окрім вже перелічених документів, вимоги до освітлення встановлено ДБН В.2.5-28:2018 «Природне і штучне освітлення». Норми освітленості при штучному освітленні та КПО при природному та сумісному освітленні зазначені у таблиці 5.4.

Таблиця 5.4 - Норми освітленості в приміщенні [46]

Характеристика зорової роботи	Найменший розмір об'єкта розрізнювання	Розряд зорової роботи	Підрозряд зорової роботи	Контраст об'єкта розрізнення з фоном	Характеристика фона	Освітленість, лк		КПО, $e_н$,%			
						Штучне освітлення		Природнє освітлення		Сумісне освітлення	
						Комбіно-ване	Загальне	Верхнє або верхнє і бокове	Бокове	Верхнє або верхнє і бокове	Бокове
Дуже високої точності	Від 0,15 до 0,3	II	г	великий	світлий	1000	300	7	2,5	4,2	1,5

Робоче місце необхідно розміщувати таким чином, щоб уникнути попадання прямого світла в очі. Відносно вікон робоче місце необхідно

організовувати так, щоб природне світло було з лівого боку. Для забезпечення захисту за необхідності використовують локальні світлофільтри (засоби індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

Штучне освітлення приміщення має здійснюватись системою загального рівномірного освітлення. У приміщеннях при переважній роботі з документами допускається використання системи комбінованого освітлення, тобто встановлення світильників місцевого освітлення додатково до загального.

Як джерела штучного освітлення необхідно використовувати люмінесцентні лампи. Допускається застосування ламп розжарювання у світильниках місцевого освітлення та, у разі влаштування відбитого освітлення у виробничих чи адміністративно-громадських приміщеннях, металогалогенних ламп потужністю 250 Вт.

Світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Для забезпечення нормованих значень освітленості у приміщеннях відповідно необхідно мити вікна і світильники не рідше 2 разів на рік, а також своєчасно замінювати лампи, що перегоріли.

5.2.4 Виробничий шум

Вплив шуму на організм людини пов’язаний в основному із застосуванням нового, високопродуктивного устаткування, з механізацією й автоматизацією виробничих процесів, переходом на високі швидкості під час експлуатації верстатів та агрегатів. Джерелами шуму можуть бути двигуни, насоси, компресори, турбіни, пневматичні інструменти, молоти, дробарки, верстати та інші установки, що мають у своєму складірухомі механізми та обертові деталі. Шум з точки зору охорони праці розглядається як стресовий чинник і загальний біологічний подразник, який негативно впливає на всі органи і системи організму, передусім на центральну нервову і серцевосудинну системи.

Нормативним документом, який регламентує рівні шуму для різних категорій робочих місць службових приміщень, є ДСН 3.3.6.037-99.

Таблиця 5.5 - Рівень звукового тиску [48]

Характер робіт	Допустимі рівні звукового тиску (дБ) в стандартизованих октавних смугах зі середньогеометричними частинами (Гц)									Допуст-мий рівень звуку, дБА
	32	63	125	250	500	1000	2000	4000	8000	
Виробничі приміщення	86	71	61	54	49	45	42	40	38	50

Для забезпечення дотримання допустимих рівнів шуму на робочих місцях застосовуються засоби звукопоглинання, вибір яких обґрунтовується спеціальними інженерно-акустичними розрахунками (п. 3.3.3 ДСанПіН 3.3.2.007-98).

Перелік організаційно-технічних заходів щодо обмеження несприятливого впливу шуму та вібрації на працюючих наведено в ДСН 2.3.6.037-99 та ДСН 3.3.6.039-99, серед яких зменшення шуму та вібрації на шляху розповсюдження засобами ізоляції та поглинання, наприклад, за рахунок використання гумових, поролонових, інших шумо- чи вібропоглинаючих матеріалів, або інших матеріалів аналогічного призначення, що дозволені для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду.

5.2.5 Виробничі випромінювання

Вимоги щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних та магнітних полів встановлюються відповідно до ДСанПіН 3.3.2.007-98, а також Вимог до роботодавців щодо захисту працівників від шкідливого впливу електромагнітних полів, затверджених наказом Міненергетики від 05.02.2014 р. № 99, ДСанПіН 3.3.6.096-2002.

Значення напруженості електростатичного поля на робочих місцях (як у зоні екрана дисплея, так і на поверхнях обладнання, клавіатури, друкувального пристрою) мають не перевищувати гранично допустимих відповідно до встановлених норм.

Ступінь впливу електромагнітних випромінювань від ПК на організм працівника залежить від діапазону частот, тривалості опромінення, характеру опромінення, режиму опромінення, розмірів поверхні тіла, яке опромінюється, та індивідуальних особливостей організму. Гранично

допустимі рівні електромагнітного поля для працівника становлять наведені в таблиці 5.6.

Таблиця 5.6 - Допустимі параметри електромагнітних неіонізуючих випромінювань і електростатистичного поля [50]

Види поля	Допустимі параметри поля		Допустима поверхнева щільність потоку енергії, Вт/кв.м
	за електричною складовою (Е), В/м	за магнітною складовою (Н), А/м	
Напруженість електромагнітного поля 60 кГц до 3 мГц	50	5	
Напруженість електромагнітного поля 3 кГц до 30 мГц	20		
Напруженість електромагнітного поля 30 кГц до 50 мГц	10	0,3	
Напруженість електромагнітного поля 30 кГц до 300 мГц 5	5		
Напруженість електромагнітного поля 300 кГц до 300 гГц			10Вт/кв. м
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-С (220 — 280 мм)			0,001
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-В (280 — 320 мм)			0,01
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-А (320 — 400 мм)			10,0
Електромагнітне поле оптичного діапазону в видимій частині спектру 400 — 760 мм			10,0

Продовження таблиці 5.6 - Допустимі параметри електромагнітних неіонізуючих випромінювань і електростатистичного поля

Електромагнітне поле оптичного діапазону в інфрачервоній частині спектру 0,76 — 10,0 мкм			35,0 — 70,0
Напруженість електричного поля відеодисплейного терміналу			20кВ/м

Окрім цього, наслідком сучасного технічного прогресу є зростання з кожним роком енергоспоживання та збільшення навантаження на кабелі, що в свою чергу призводить до збільшення напруги електромагнітних полів, несприятлива дія яких може призвести до погіршення стану здоров'я працівників. Для обмеження впливу ЕМП на розробника слід використовувати лише якісну техніку із сертифікатом якості і дотримуватися встановленого часу роботи за ПК.

5.2.6 Психофізіологічні фактори

Оцінка психофізіологічних факторів під час розробки універсального програматора для сучасних мікроконтролерів здійснюється відповідно до Гігієнічної класифікацією праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу.

Загальні енергозатрати організму: до 174 Вт.

Стереотипні робочі рухи (кількість за зміну): до 40 000.

Робоча поза: вільна зручна поза, можливість зміни пози («сидячи – стоячи») за бажанням працівника; перебування в позі «стоячи» до 40% часу зміни.

Нахили тулуба (вимушені, більше 30°), кількість за зміну: до 50 раз.

Класи умов праці за показниками напруженості праці:

Інтелектуальні навантаження:

–зміст роботи – творча діяльність, що вимагає вирішення складних завдань за відсутності алгоритму;

–сприймання інформації та їх оцінка – сприймання сигналів з наступним порівнянням фактичних значень параметрів з їх номінальними значеннями. Заключна оцінка фактичних значень параметрів;

–розподіл функцій за ступенем складності завдання – обробка, виконання завдання та його перевірка.

Сенсорні навантаження:

–зосередження (%за зміну) – до 5-75%;

–щільність сигналів (звукові за1 год) – до 150;

–навантаження на слуховий аналізатор (%) – розбірливість слів та сигналів від 50 до 80 %;

–спостереження за екранами відеотерміналів (годин на зміну) – 4-6год.

–навантаження на голосовий апарат (протягом тижня) – від 16 до 20.

Емоційне навантаження:

ступінь відповідальності за результат своєї діяльності – є відповідальним за функціональну якість основної роботи; Ступінь ризику для власного життя – вірогідний;

Режим праці:

– тривалість робочого дня – більше 8 год;

– змінність роботи – однозмінна (без нічної зміни).

За зазначеними показниками важкості та напруженості праці, робота, яка виконується належить до допустимого класу умов праці (напруженість праці середнього ступеня).

5.3 Безпека у надзвичайних ситуаціях. Дослідження безпеки роботи програматора для сучасних мікроконтролерів в умовах дії загрозливих чинників надзвичайних ситуацій

Система передачі даних спрямована на забезпечення послугами цифрового зв’язку для різних задач. У зв’язку з тим, що це має важливе значення для воєнної сфери, то на них можуть справляти значний вплив загрозливі чинники надзвичайні ситуації різного типу і необхідно провести дослідження безпеки роботи системи пілот-тоном. До таких НС можна віднести: стихійні лиха (землетруси, блискавка, зливи), а особливо впливовими на РЕА мають іонізуючі випромінювання та ЕМІ. Тому при забезпеченні даних пристроїв слід забезпечити найвищий рівень захисту від тої чи іншої НС, оскільки кожна НС має свій вплив на дану систему.

Тож, в даній частині розділу необхідно провести дослідження безпеки роботи програматора для сучасних мікроконтролерів та розробку заходів по підвищенню стійкості роботи програматора для сучасних мікроконтролерів в умовах дії іонізуючих випромінювань та електромагнітного імпульсу.

В РЕА застосовуються елементи, до складу яких входять такі матеріали: метали, неорганічні матеріали (в основному діелектрики),

провідники і різноманітні органічні сполуки (діелектрики, смоли і т.д.). Серед цих матеріалів метали найбільш чутливі до впливу іонізуючих випромінювань, оскільки їм властива висока концентрація вільних носіїв.

В радіоелектронній апаратурі іонізуючі випромінювання, викликають зворотні і незворотні процеси, внаслідок яких можуть відбуватися порушення роботи електричних елементів схеми, що призводять до виходу з ладу апаратури. Так, проходячи через елементи РЕА, потік гамма-випромінень створює в них вільні носії електричних зарядів, в результаті переміщення яких виникає помилковий імпульс, який призводить до спрацьовування пристрою. При великих дозах випромінювання втрачають працездатність комплектуючі елементи систем радіоелектроніки і електроавтоматики. В результаті опромінення у транзисторах змінюється обернений струм і коефіцієнт підсилення, у конденсаторах знижуються напруги пробою та опір стікання, змінюється провідність і внутрішній нагрів; руйнується електрична ізоляція дротів з полімерних матеріалів. Неорганічні матеріали менш чутливі до впливу іонізуючих випромінювань [49].

Для інженерної практики найбільший інтерес представляє перший випадок, тобто оцінка безпеки роботи РЕА при перебуванні її в зараженій радіоактивними речовинами місцевості протягом певного часу після випадання радіоактивних речовин у даній місцевості.

ЕМІ ушкоджує напівпровідникові прилади, резистори, конденсатори. Це являє велику небезпеку для апаратури, добре захищеної від впливу інших загрозливих чинників. Тому слід пам’ятати про те, що захист апаратури від механічних ушкоджень не захищає від впливу ЕМІ. Апаратура може втратити працездатність, знаходячись у надійних захисних спорудженнях [50].

5.3.1 Дослідження безпеки роботи програматора для сучасних мікроконтролерів в умовах дії іонізуючих випромінювань.

За критерій безпеки роботи програматора для сучасних мікроконтролерів в цих умовах приймається таке граничне значення рівня ($Р_{зв}$, Р/год), при якому можуть виникнути тимчасові зміни, але пристрій буде працювати з потрібною якістю.

Приймаючи до уваги елементну базу, що використовується для реалізації програматора для сучасних мікроконтролерів , складається таблиця потужностей експозиційної дози опромінення для кожного елемента $Р_{зв.і}$, що

викликають початок зворотних змін Отримані значення занесемо до таблиці 5.7.

Визначається елемент, який найбільшою мірою піддається впливу випромінюванням, тобто елемент із мінімальним значенням $Р_{зв}$.

$$Р_{зв} = 10^3 \text{ Р/год.}$$

Таблиця 5.7 – Потужність граничної експозиційної дози для програматора для сучасних мікроконтролерів

№	Елементи програматора для мікроконтролерів	$Р_{зв.і}$, Р/год	$Р_{зв.}$, Р/год
1	Процесори, інтегральні мікросхеми	10^3	10^3
2	Діоди загального призначення	10^4	
3	Транзистори загального призначення	10^4	
4	Мікросхеми	10^5	
5	Конденсатори	10^7	
6	Резистори	10^8	

В якості критерію стійкості роботи РЕА використовується граничне значення рівня іонізуючих випромінювань

$$Р_{гр} = К_{над*} Р_{зв*} К_{посл}, \quad (5.1)$$

де $Р_{зв}$ - рівень радіації незворотних змін пристрою в цілому;

$К_{над}$ - коефіцієнт надійності ($К_{над}$ =0,9÷0,95);

$К_{посл}$ - коефіцієнт послаблення.

$$Р_{гр} = 0,95* 10^3 *2 = 7,65 \text{ Р/год.}$$

З наведеної таблиці слідує, що мінімальні значення граничних рівнів радіації елементів, при яких в елементній базі можливі необоротні зміни мають інтегральні мікросхеми великої ступені інтеграції та мікропроцесори – $Р_{зв}=10^3$, $k_{посл}=1$.

Визначаємо допустимий час роботи пристрою

$$t_{доп} = \left(\frac{Д_{гр} \cdot К_{осл} + 2 \cdot P_1 \cdot \sqrt{1}}{2 \cdot P_1} \right)^2, \quad (5.2)$$

Таким чином, допустимий час роботи програматора для сучасних мікроконтролерів складатиме 17248 годин при максимальному рівні радіації 7,65 Р/год.

5.3.2 Дослідження безпеки роботи програматора для сучасних мікроконтролерів в умовах дії електромагнітного імпульсу.

За критерій безпеки роботи програматора для сучасних мікроконтролерів в умовах дії електромагнітного імпульсу можна прийняти коефіцієнт безпеки

$$K_б = 20\lg\frac{U_д}{U_г} \geq 40 \ [дБ],$$

де $U_д$ – допустиме коливання напруги живлення (для мікросхем 5 В);

$U_г$ – напруга наведена за рахунок електромагнітного імпульсу у вертикальних (горизонтальних) струмопровідних частинах, В.

Допустимі коливання напруги живлення

$$U_д = U_ж + \frac{Uж}{100} * N = 5 + \frac{5}{100} = 5{,}25(B).$$

В зв’язку з тим, що окремі елементи програматора можуть мати різні значення коефіцієнтів безпеки, то стійкість роботи пристрою в цілому визначається мінімальним значенням коефіцієнта безпеки.

З рівняння (5.1) визначаємо

$$U_г = \frac{Uд}{10^{\frac{40}{20}}} = \frac{5{,}25}{100} = 0{,}05(B).$$

Приймемо максимальну довжину горизонтальних струмопровідних частин $l_г$=0,58 м. Тоді горизонтальна складова напруженості електричного поля визначається за формулою

$$E_г = U_в / l_в = 2{,}63/0{,}58 = 4{,}53 \text{ (В/м)}.$$

Звідси, зважаючи, що вертикальна складова в тисячу разів вища, будемо мати її оціночне значення $E_в$=4,530 (кВ/м).

Таким чином, робота програматора для сучасних мікроконтролерів можлива у випадку, якщо не перевищується значення вертикальної складової напруженості електричного поля 4,530 (кВ/м).

5.3.3 Розробка заходів по підвищенню безпеки роботи програматора для сучасних мікроконтролерів в умовах надзвичайних ситуацій.

З метою зменшення негативного впливу на пристрій можна використати наступні методи.

Для захисту розробки, як і любих радіоелектронних пристроїв від дії іонізуючих випромінювань можна використати алюмінієві сплави, леговані елементами з високим атомним номером (лантаноїдами і рідкоземельними елементами), сплави на основі тугоплавких і рідкоземельних елементів і багатошарові матеріали. Також для боротьби з впливом іонізуючого випромінювання можна використати новітній вітчизняний метод, що полягає в захисному покритті радіоелектронної апаратури, що розміщується на поверхнях даних елементів, які піддаються впливу іонізуючого випромінювання, відмінним тим, що захисне покриття виконане у вигляді наноструктури, яка включає сукупність атомів рідкоземельних елементів, введених в структуру армованої атомно-молекулярної металічної матриці, або утворює її захисний шар.

Найкращим для захисту від електромагнітного імпульсу є захищене металічним екраном приміщення, в якому розміщена радіоелектронна апаратура. Оскільки такий захист в ряді випадків неможливо виконати, то використовуються менш надійні засоби захисту, такі як струмопровідні сітки та плівкові покриття вікон, стільникові металеві конструкції для повітрозбірників та вентиляційних отворів і контактні пружинні прокладки, що розміщуються по периметру дверей і люків. Також для захисту кабельних вводів використовують в їх конструкції фільтри та встановлення вбудованих зенерівських діодів. Сучасний ефективний спосіб захисту від EMI – це використання по внутрішній поверхні пластикового корпусу композитних фарб на основі пласких часточок срібла є найкращий, хоч і вимагає спеціальної технології покриття. Цей спосіб найбільше використовують у військових апаратах та системах подвійного призначення.

В ході виконання було розглянуто вплив іонізуючого випромінювання та EMI на компоненти схеми, виконано розрахунки з яких видно, що ні один з класів елементів схеми не зазнає більшого впливу за граничне значення,

також розраховано термін безпечної роботи системи, який складає 17248год при максимальному значені потужності експозиційної дози 7,65 Р/год.

Що стосується впливу електромагнітного імпульсу, то з урахуванням необхідного рівня коефіцієнта безпеки було розраховано значення напруженості електричного поля. Безпечна робота програматора для сучасних мікроконтролерів можлива у випадку, якщо не перевищується значення вертикальної складової напруженості електричного поля 4,530 (кВ/м). Для підвищення безпеки роботи програматора для сучасних мікроконтролерів наведено основні заходи боротьби з впливом загрозливих чинників НС.

Отже основною метою даної частини розділу було дослідження безпеки роботи програматора для сучасних мікроконтролерів та розробка дієвих заходів по підвищенню безпеки роботи цієї системи в умовах надзвичайних ситуацій.

ВИСНОВКИ

Проведено аналіз існуючих пристрої для програмування мікроконтролерів та обґрунтовано недоліки та переваги пристроїв для програмування МК. Поставлено задачі, які необхідно розв’язати при розробці програма тора для мікроконтролерів по відношенню до існуючих. Досліджено протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів. Розглянуто набір команд завантажувача їх алгоритми та принцип роботи.

Розглянуто інтерфейси програмування мікроконтролерів STM32. Мікроконтролери STM32 в залежності від моделі підтримують від двох до шести способів завантаження програми у пам’ять. Більшість із вказаних способів це використання завантажувача bootloader. Максимальна кількість можливих варіантів bootloader у одному мікроконтролері – 5, а саме через протоколи USART, I2C, SPI, CAN, DFU. Хоча не всі моделі STM32 підтримують по п’ять протоколів, кожна із них підтримує протокол USART, що і стало причиною вибору даного протоколу для реалізації пристрою.

Розроблено пристрій для програмування восьми та тридцяти двох розрядних мікроконтролерів, він складається із чотирьох структурних блоків. Розроблений пристрій складається із трьох інтерфейсів обміну даними та основного блоку. Приймач/передавач USB відповідає за перетворення прийнятих від USB порту комп’ютера сигналів у зрозумілі основному блокові. Даний блок працює у режимі Full Speed. Блоки UART та SPI здійснюють передачу сигналів програмування від основного блоку до мікроконтролерів та у зворотному напрямі.

Розроблено структурну та принципову схему пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів.

Розроблено друковану плату пристрою для програмування восьми та тридцяти двох розрядних мікроконтролерів в пакеті програм DipTrace, а також розроблено 3D модель друкованої плати з елементами.

Розглянуто алгоритм прошивання мікроконтролера STM32F103C8, як основного елемента розробленого пристрою.

В економічній частині основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Абсолютна ефективність вкладених коштів для даної розробки складе

266984,00 грн., а відносна ефективність забезпечить прибутковість в межах 30%, що більше мінімальної яка складає 20%.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$=3,39 року, що менше нормативного терміну.

Враховуючи наведені показники діяльності можна зробити висновок про доцільність проведення науково-дослідної роботи.

В розділі охорони праці та безпеки надзвичайних ситуацій було розглянуто вплив іонізуючого випромінювання та EMI на компоненти схеми, виконано розрахунки з яких видно, що ні один з класів елементів схеми не зазнає більшого впливу за граничне значення, також розраховано термін безпечної роботи системи, який складає 17248год при максимальному значені потужності експозиційної дози 7,65 Р/год.

ПЕРЕЛІК ПОСИЛАНЬ

1. А.В. Евстигнеев «Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL», М. Издательский дом «Додэка-XXI», 2005.
2. Future Technology Devices International Ltd. “FT2232D Dual USB UART/FIFO I.C.” , Datasheet, 2006.
3. Future Technology Devices International Ltd. “Software Application Development D2XX Programmer's Guide” , Document, 2009.
4. Future Technology Devices International Ltd. “Programmers Guide for High Speed FTCJTAG DLL” , Application note AN_110, 2009.
5. Future Technology Devices International Ltd. “Programmers Guide for High Speed FTCSPI DLL” , Application note AN_111, 2009.
6. Эндрю Троелсен «C# и платформа .NET» М.,С-П. Питер, 2007.
7. Geoffrey Brown. Discovering the STM32 Microcontroller . 2016, –244 p.
8. ARM Limited. Cortex-M3 technical reference manual, 2006.
9. ARM Limited. Procedure call standard for the ARM® architecture, October 2009. IHI 0042D.
10. ChaN. FatFs generic FAT file system module, 2011. http://elm-chan. org/fsw/ff/00index_e.html. Accessed April 2012.
11. F. Foust. Secure digital card interface for the MSP430, 2004. http://alumni.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_ appnote_foust.pdf. Accessed July 2012.
12. J. Jiu. The Definitive guide to the ARM Cortext-M3. Newnes, 2010.
13. Microchip. 25AA160A/25LC160A 16K SPI bus serial EEPROM, 2004.
14. NXP. ı2c bus specification and user manual (rev 03), June 2007. UM10204.
15. C. Philips. Read wii nunchuck data into arduino. http://www. windmeadow.com/node/42. Accessed February 4, 2012.
16. SD specifications, part 1, physcial layer simplified specification, version 3.01, May 2010.
17. F. Semiconductor. Tilt sensing using linear accelerometers, 2012.
18. Sitronix Technology Corporation. Sitronix st7735r 262k color single-chip TFT controller/driver, 2009.
19. STMicroelectronics. How to get the best ADC accuracy in the STM32F10xxx devices, November 2008.
20. STMicroelectronics. STM32F10xxx ı2c optimized examples, 2010.

21. Козаченко В.Ф. Практический курс микропроцессорной техники на базе процессорных ядер ARM-Cortex-M3/M4/M4F" 2019 М.: МЭИ, –256 с.
22. Kento Watanabe. Introduction to STM32 ARM Microcontroller with STM HAL-Library & SW4STM32. 2017. –99 p.
23. Белов А. В. Программирование микроконтроллеров для начинающих и не только. НиТ, 2016, –352 с.
24. Микроконтроллеры AVR: от простого к сложному / М. С. Голубцов - М.: СОЛОН-Пресс, 2003. – 288 с.
25. Электроника и схемотехника. Основы электроники: конспект лекций для высшего профессионального образования / В.Т. Еременко, А.А. Рабочий, И.И. Невров, А.П. Фисун, А.В. Тютякин, В.М. Донцов, О.А. Воронина, А.Е. Георгиевский. – Орел: ФГБОУ ВПО «Госуниверситет - УНПК», 2012. – 290 с.
26. Опадчий, Е.Ф. Аналоговая и цифровая электроника: учебник для вузов / Ю.Ф. Опадчий, О.П. Глудкин, А.И. Гуров; под ред. О.П. Глудкина. – М.: Горячая линия – Телеком, 2002. – 768 с.
27. Гутников, В.С. Интегральная электроника в измерительных устройствах учебное пособие/ В.С. Гутников. – Л.: Энергоатомиздат, 1988. – 304 с.: ил. 7. Гусев, В.Г. Электроника: учебное пособие для приборостроительных специальностей вузов / В.Г. Гусев, Ю.М. Гусев. – 2-е изд.– М.: Высш. шк., 1991. – 662 с.
28. Ефимов, И.Е. Микроэлектроника. Физические и технологические основы, надежность / И.Е. Ефимов, И.Я. Козырь, Ю.И. Горбунов. – М.: Высш. шк., 1986. – 464 с.
29. Жеребцов, И.П. Основы электроники / И.П. Жеребцов. – Л.: Энергоатомиздат, 1989. – 352 с.
30. Цифровая схемотехника и архитектура комп’ютера/ Второе издание Дэвид М. Харрис и Сара Л. Харрис. Издательство Morgan Kaufman. English Edition 2013. –1662 с.
31. Новиков Ю. В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. - М.: Мир, 2001. - 379 с.
32. Чижма С.Н. Основы схемотехніки. Учебное пособие для вузов. — Омск: Апельсин, 2008. — 424 с.
33. Хоровиц П., Хилл У. Искусство схемотехники. (The Art of Electronics). Монография. Издание 7-е. Москва: Издательство «Бином», 2015. –704 с.

34. Осадчук О.В. Мікроелектронні частотні сенсори на основі транзисторних структур з від'ємним опором. – Вінниця: «УНІВЕРСУМ–Вінниця», 2000.–303с.
35. Осадчук В.С. Индуктивный эффект в полупроводниковых приборах. - К.: Вища школа, 1987. – 155 с.
36. Осадчук В.С., Осадчук О.В. Реактивні властивості транзисторів і транзисторних схем. –Вінниця: «Універсум-Вінниця», 1999. – 275 с.
37. Осадчук В.С., Осадчук О.В., Семенов А.О., Коваль К.О. Функціональні вузли радіовимірювальних приладів на основі реактивних властивостей транзисторних структур з від'ємним опором. – Вінниця: ВНТУ, 2011. – 336с.
38. Осадчук В.С., Осадчук О.В., Семенов А.О. Генератори електричних коливань на основі транзисторних структур з від'ємним опором. Монографія. –Вінниця: «Універсум-Вінниця», 2009. – 182 с.
39. Методичні вказівки до виконання студентами-магістрантами наукового напрямку економічної частини магістерських кваліфікаційних робіт / Уклад. В.О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.
40. Козловський В.О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах. Навчальний посібник. – Вінниця : ВДТУ, 2003. – 75с.
41. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.
42. Наказ від 08.04.2014 № 248 Про затвердження Державних санітарних норм та правил Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу - [Електронний ресурс] - Режим доступу: http://online.budstandart.com/ua/catalog/topiccatalogua/labor-protection/14._nakazy_ta_rozpor_183575/248+58074-detail.html
43. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация. . - [Електронний ресурс] - Режим доступу: http://www.znaytovar.ru/gost/2/GOST_12000374_SSBT_Opasnye_i_v.html
44. ДБН В.2.5-28:2018 Природне і штучне освітлення - [Електронний ресурс] - Режим доступу: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=79885

45. ДНАОП 0.00-1.21-98 Правила безпечної експлуатації електроустановок споживачів - [Електронний ресурс] - Режим доступу: http://dnop.com.ua/dnaop/act3167.htm
46. ДСанПіН 3.3.2.007-98 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ЕОМ - [Електронний ресурс] - Режим доступу: http://document.ua/derzhavni-sanitarni-pravila-i-normi-roboti-z-vizualnimi-disp-nor4881.html.
47. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. - [Електронний ресурс] - Режим доступу: http://document.ua/sanitarni-normi-virobnichogo-shumu-ultrazvuku-ta-infrazvuku-nor4878.html.
48. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. - [Електронний ресурс] - Режим доступу: http://mozdocs.kiev.ua/view.php?id=1972.
49. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. - [Електронний ресурс] - Режим доступу: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php.
50. Конституція України від 28.06.1996 № 254к/96-ВР.
51. Про охорону праці Закон України від 14.10.1992 № 2694-XII - [Електронний ресурс] - Режим доступу: http://zakon2.rada.gov.ua/laws/show/2694-12

Додаток А
(обов'язковий)

ЗАТВЕРДЖУЮ
Завідувач кафедри РТ
д.т.н., професор О.В. Осадчук

«_15__» __03____2021 р.
(підпис)

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ
08-36.МКР.002.00.000 ТЗ

Керівник МКР д.т.н.,проф.
О.В. Осадчук
_____ «____» _____2021 р.
(підпис)

Розробив студент гр. РТ-19м з/в
Онофрійчук В.А.
_____ «____» _____2021 р.
(підпис)

Вінниця ВНТУ 2021

1. ПІДСТАВА ДЛЯ ВИКОНАННЯ РОБОТИ

Робота проводиться на підставі наказу ректора по Вінницькому національному технічному університету № 64 “09 ” 03 2021 року та індивідуального завдання на магістерську кваліфікаційну роботу.

Дата початку роботи: “03 ” 09 2020 р.

Дата закінчення: “ 06 ” червень 2021 р.

2. МЕТА І ПРИЗНАЧЕННЯ МКР

Мета і задачі дослідження

Метою роботи є створення універсального програматора для сучасних мікроконтролерів.

Об’єктом дослідження є універсальний програматора для сучасних мікроконтролерів виробництва Atmel та STMicroelectronics.

Предметом дослідження – протоколи програмування мікроконтролерів різних виробників та різної архітектури.

Для досягнення поставленої мети у магістерській кваліфікаційній роботі розв’язуються такі *задачі*:

- проаналізувати існуючі пристрої для програмування мікроконтролерів та обґрунтувати переваги пристрою, який розробляється по відношенню до існуючих;
- дослідити протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів;
- дослідити можливості мікроконтролера STM32F103C8;
- створення керуючої програми для мікроконтролера STM32F103C8 мовою C++;
- створення керуючої програми пристрою для програмування мікроконтролерів мовою Java призначеної для роботи на операційній системі Windows та Linux;
- виконати експериментальну перевірку розробленого пристрою для програмування мікроконтролерів;
- здійснити метрологічну оцінку похибок програмування розробленого універсального програматора для сучасних мікроконтролерів.

Методи дослідження ґрунтуються на використанні:

- технічного опису протоколів передачі даних USB, UART та SPI;

- теорії розрахунку електричних кіл з використанням законів Ома для розрахунку номіналів елементів схеми;
- теорії ймовірності для оцінки похибок вимірювання.

4 ДЖЕРЕЛА РОЗРОБКИ

1. Осадчук О.В. Мікроелектронні частотні сенсори на основі транзисторних структур з від'ємним опором. – Вінниця: «УНІВЕРСУМ–Вінниця», 2000.–303с.
2. Осадчук В.С. Индуктивный эффект в полупроводниковых приборах. - К.: Вища школа, 1987. – 155 с.
3. Осадчук В.С., Осадчук О.В. Реактивні властивості транзисторів і транзисторних схем. –Вінниця: «Універсум-Вінниця», 1999. – 275 с.
4. Методичні вказівки до виконання студентами-магістрантами наукового напрямку економічної частини магістерських кваліфікаційних робіт / Уклад. В.О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.
5. Козловський В.О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах. Навчальний посібник. – Вінниця : ВДТУ, 2003. – 75с.
6. Наказ від 08.04.2014 № 248 Про затвердження Державних санітарних норм та правил Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу - [Електронний ресурс] - Режим доступу: http://online.budstandart.com/ua/catalog/topiccatalogua/labor-protection/14._nakazy_ta_rozpor_183575/248+58074-detail.html

ВИКОНАВЕЦЬ

Вінницький національний технічний університет, кафедра радіотехніки, студент групи РТ-19м з/в Онофрійчук Володимир Анатолійович.

5 Технічні вимоги

5.1 Варіанти виконання

Пристрій має бути виконаний у вигляді окремого функціонального блоку та з'єднуватись за допомогою роз'ємів з іншими пристроями. У варіанті виконання із застосуванням МК повинен використовуватись STM32. Номінальна напруга живлення +5 В.

5.2 Вид цифрових елементів

В якості цифрових елементів використовується мікроконтролер STM32.

5.3 Джерело електроенергії

номінальна напруга +5 В; +3,3 В.

5.4 Споживана потужність

Споживана потужність не повинна перевищувати 0,05 Вт.

5.5 Час готовності до роботи

Час готовності до роботи, не більше 1 с

5.6 Параметри джерела живлення:

Номінальна напруга живлення +5 В; 3,3 В. Струм споживання знаходиться в межах 1-15 мА.

5.7 Робоча частота:

Діапазон робочих частот становить 50 кГц - 6500 кГц;

5.8 Діапазон робочих температур:

Діапазон робочих температур складає 0…+60 °С.

6. ЕТАПИ МКР І ТЕРМІНИ ЇХ ВИКОНАННЯ

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання		Очікувані результати	Звітна документація
1.	Огляд літературних джерел. Вибір та узгодження теми МКР	03.09.2020	31.12.2021	Проведено огляд літературних джерел. Вибрана тема	Узгодження теми МКР на кафедрі
2.	Аналіз літературних джерел. Попередня розробка основних розділів	07.01.2021	14.02.2021	Проведений аналіз літературних джерел по даній тематиці. Підготовлений матеріал основних розділів	Вступ
3.	Затвердження теми. Розробка технічного завдання	15.02.2021	06.03.2021	Розроблене ТЗ	Наказ ВНТУ про затвердження теми. Додаток А
4.	Аналіз вирішення поставленої задачі. Розробка структурної схеми	01.03.2021	21.03.2021	Проведений аналіз. Розроблені схеми пристрою	Звіт по переддипломній практиці Вступ Розділ 1-2

5.	Електричні розрахунки. Експериментальне дослідження	22.03.2021	11.04.2021	Проведені розрахунки та дослідження	Розділ 3
6.	Розділ моделювання	12.04.2021	19.04.2021	Проведено моделювання	Результати моделювання
7.	Розробка графічної частини МКР	20.04.2021	30.04.2021	Плакати. Структурні та електричні схеми	Графічна частина
8.	Аналіз економічної ефективності розробки	01.05.2021	14.05.2021	Економічна частина	Розділ 4
9.	Охорона праці (ОП)	15.05.2021	20.05.2021	Частина БЖДПБ	Розділ 5
10.	Оформлення пояснювальної записки та графічної частини	21.05.2021	26.05.2021	Оформлена документація	ПЗ та графічна частина
11.	Нормоконтроль	27.05.2021	31.05.2021	Підпис нормоконтроля	Оформлена ПЗ та графічна частина
12.	Попередній захист МКР, доопрацювання, рецензування МКР	01.06.2021	06.06.2021	Позитивні відзиви	Відзив. Рецензія
13.	Захист МКР ЕК	07.06.2021	11.06.2021	Позитивний захист	Протокол ЕК

7. ОЧІКУВАНІ РЕЗУЛЬТАТИ ТА ПОРЯДОК РЕАЛІЗАЦІЇ МКР

В результаті виконання роботи будуть розроблені:

- аналіз існуючі пристрої для програмування мікроконтролерів та обґрунтування переваг пристрою, який розробляється по відношенню до існуючих;
- протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів;
- дослідження можливості мікроконтролера STM32F103C8;
- керуюча програма для мікроконтролера STM32F103C8 мовою C++;
- експериментальна перевірка розробленого пристрою для програмування мікроконтролерів;
- розділ безпеки життєдіяльності.
- розділ економічної частини.

8 МАТЕРІАЛИ, ЯКІ ПОДАЮТЬСЯ ПІСЛЯ ЗАКІНЧЕННЯ РОБОТИ ТА ПІД ЧАС ЕТАПІВ

За результатами виконання МКР до ЕК подаються пояснювальна записка, графічна частина МКР, відзив і рецензія.

9 ПОРЯДОК ПРИЙМАННЯ МКР ТА ЇЇ ЕТАПІВ

Поетапно результати виконання МКР розглядаються керівником роботи та обговорюються на засіданні кафедри.

Захист магістерської кваліфікаційної роботи відбувається на відкритому засіданні ЕК.

10. ВИМОГИ ДО РОЗРОБЛЮВАНОЇ ДОКУМЕНТАЦІЇ

Документація, що розробляється в процесі виконання досліджень повинна містити:

- структурна та принципова схеми мікропроцесорного програматора;
- протоколи програмування мікроконтролерів та види пристроїв для програмування мікроконтролерів;
- керуюча програма для мікроконтролера STM32F103C8 мовою C++;
- розділ безпеки життєдіяльності;
- розділ економічної частини.

11. ВИМОГИ ЩОДО ТЕХНІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ З ОБМЕЖЕНИМ ДОСТУПОМ

У зв`язку з тим, що інформація не є конфіденційною, заходи з її технічного захисту не передбачаються.

Додаток Б
(обов’язковий)

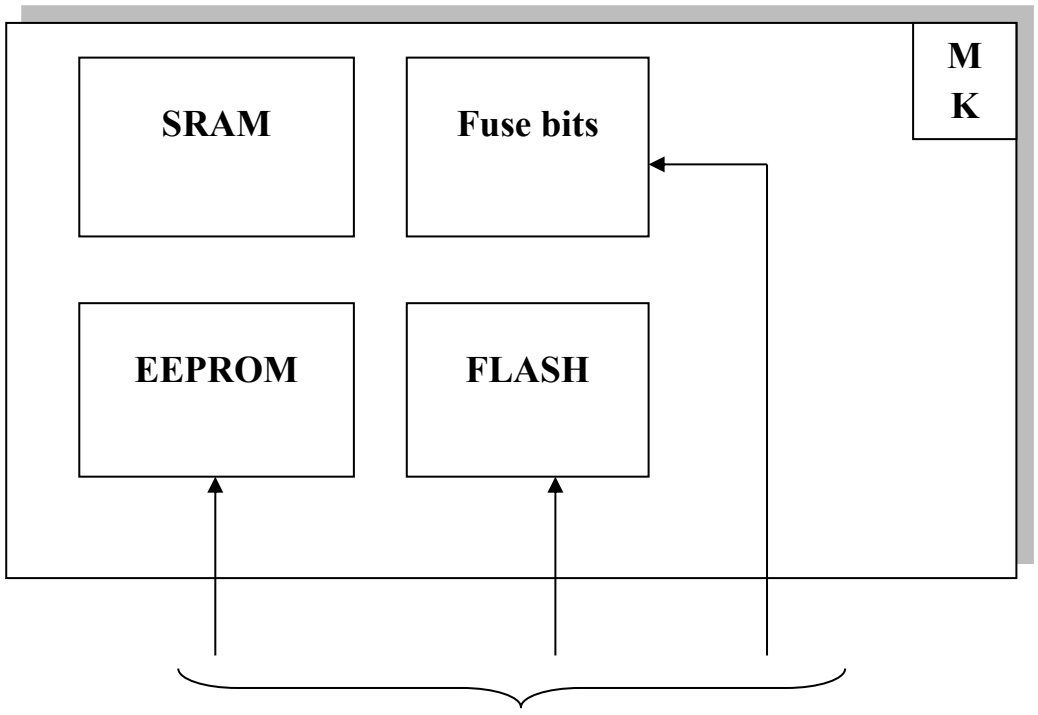
УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

Автономні програматори
Будова автономного комп’ютерного програматора

Додаток В
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

Склад областей пам’яті мікроконтролера виробництва Atmel

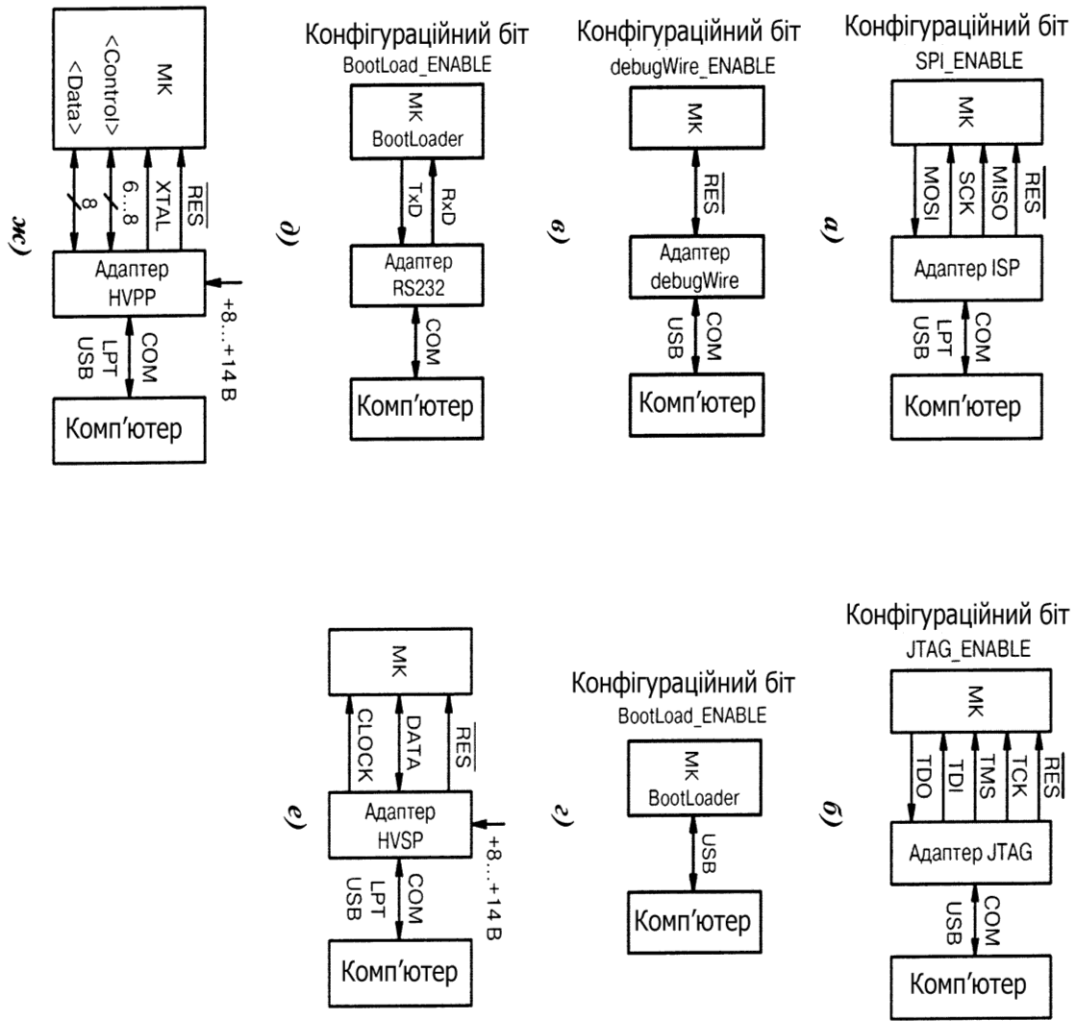

M
K
SRAM
Fuse bits
EEPROM
FLASH

Додаток Д
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

Різновиди систем програмування

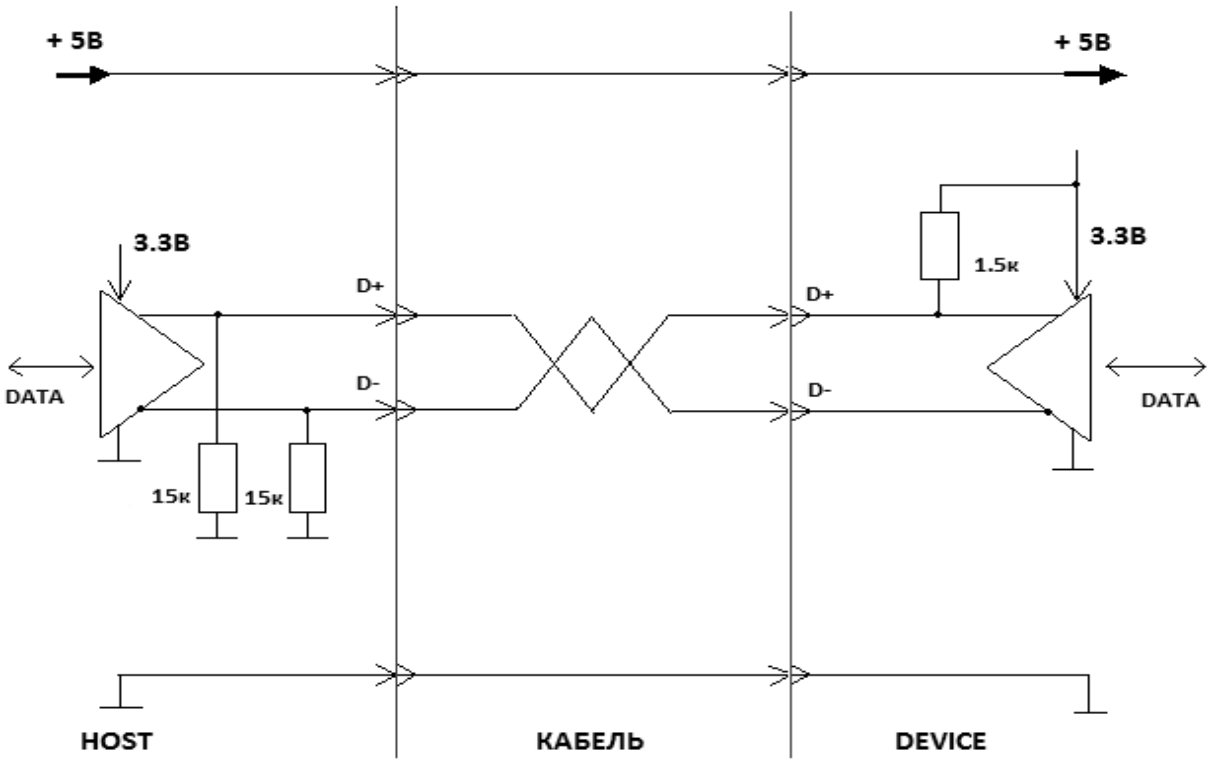
Різновиди систем програмування



Додаток Е
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

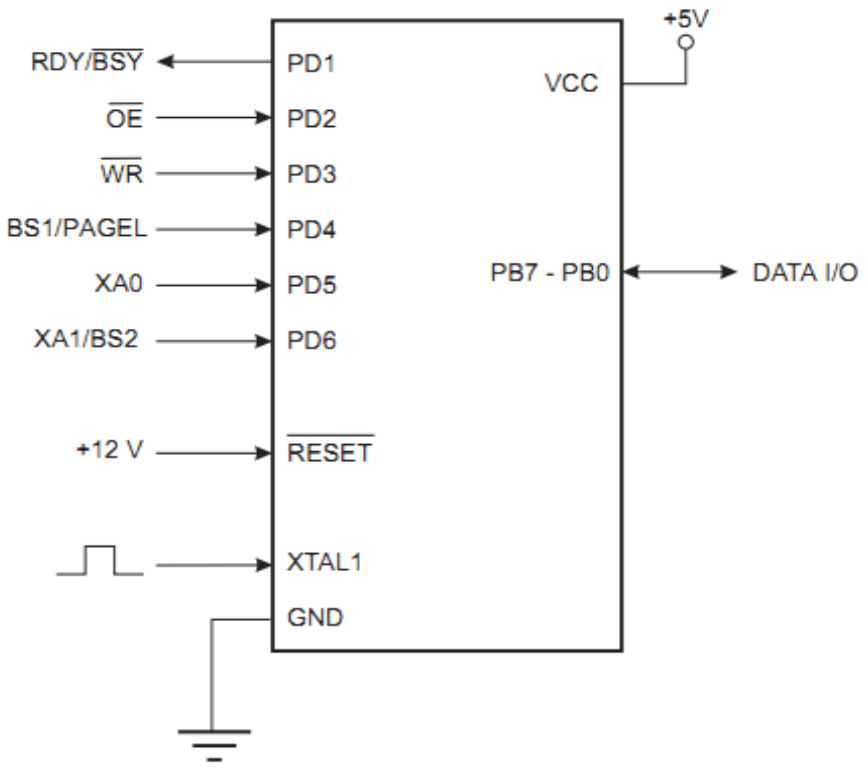
Спрощена електрична схема USB з’єднання


+ 5В
+ 5В
3.3В
3.3В
1.5к
D+
D+
D-
D-
DATA
DATA
15к
15к
HOST
КАБЕЛЬ
DEVICE

Додаток Ж
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

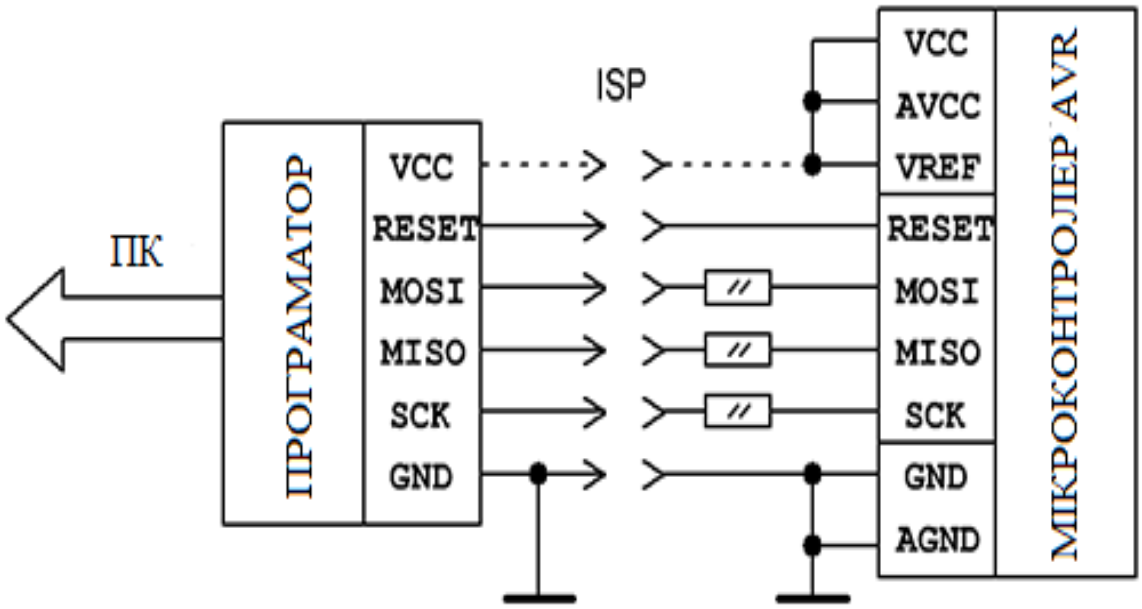
Паралельне програмування ATTiny2313


+5V
RDY/BSY
PD1
VCC
OE
PD2
WR
PD3
BS1/PAGEL
PD4
PB7 - PB0
DATA I/O
XA0
PD5
XA1/BS2
PD6
+12 V
RESET
XTAL1
GND

Додаток К
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

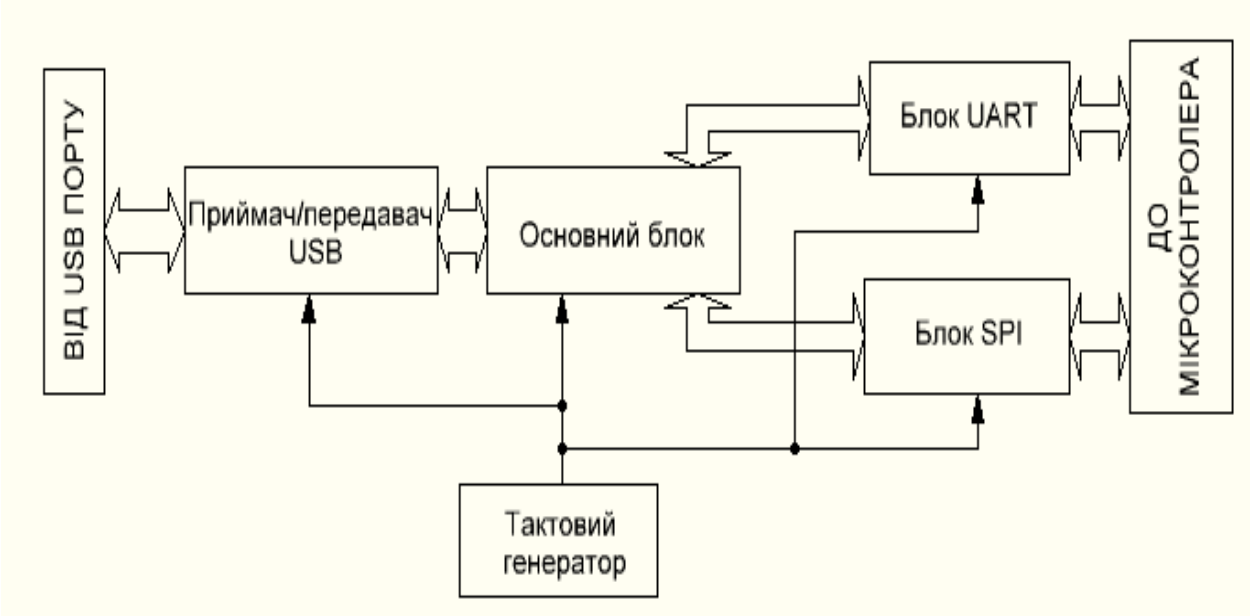
Схема з’єднання програматора з МК AVR при програмуванні через SPI


ПК
ПРОГРАМАТОР
VCC
RESET
MOSI
MISO
SCK
GND
ISP
VCC
AVCC
VREF
RESET
MOSI
MISO
SCK
GND
AGND
МИКРОКОНТРОЈЕР AVR

Додаток Л
(обов’язковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

Структурна схема


ВІД USB ПОРТУ
Приймач/передавач USB
Основний блок
Блок UART
Блок SPI
ДО МІКРОКОНТРОЛЕРА
Тактовий генератор

Додаток М
(довідниковий)

УНІВЕРСАЛЬНИЙ ПРОГРАМАТОР ДЛЯ СУЧАСНИХ МІКРОКОНТРОЛЕРІВ

Лістинг програми для мікроконтролера

Лістинг програми для мікроконтролера

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************/
/* Includes ------------------------------------------------------------------*/
#include "stm32f0xx_hal.h"
#include "spi.h"
#include "tim.h"
#include "usart.h"
#include "usb_device.h"
#include "gpio.h"
/* USER CODE BEGIN Includes */
/*For using CDC and CustomHID class functions*/
#include "usbd_cdc_hid.h"
#include <stdio.h>
#include "Handler.h"
/* USER CODE END Includes */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
//blink LEDs var
uint8_t rxLed=0,txLed=0;
uint8_t uartBusy = 0;
char tmp[BUF_SIZE];
/*USBHID functions and variables declaration*/
extern USBD_HandleTypeDef hUsbDeviceFS;
/*CDC buffers declaration*/
char uart_tx[BUF_SIZE];
uint8_t countTx=0;
uint8_t writePointerTx=0, readPointerTx=0;
char uart_rx[BUF_SIZE];
uint8_t countRx=0;
uint8_t writePointerRx=0, readPointerRx=0;
/*SPI buffers declaration*/
char spi_tx[BUF_SIZE];
uint8_t spiCountTx=0;
uint8_t spiWritePointerTx=0, spiReadPointerTx=0;
char spi_rx[BUF_SIZE];
uint8_t spiCountRx=0;
uint8_t spiWritePointerRx=0, spiReadPointerRx=0;
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void Error_Handler(void);
/* USER CODE BEGIN 0 */
void clearStr(char *buf){
	int i;
	for(i = 0; i<BUF_SIZE; i++){
		buf[i]=0;
	}
}
void blinkTX(){
	txLed=5;
	TIM3->CCR3=65535;
}
void blinkRX(){
	rxLed=5;
	TIM3->CCR4=65535;
}
```

```
//
void cdcTransmit(uint8_t size){
	blinkRX();
	for(int i=0;i<size;i++){
		tmp[i]=uart_rx[readPointerRx];
		readPointerRx=(readPointerRx+1)%BUF_SIZE;
		if(i==49){
			CDC_Transmit_FS((uint8_t*)tmp, 50);
			countRx-=50;
			size-=50;
			i=-1;
		}
	}
	if(size!=0){
		CDC_Transmit_FS((uint8_t*)tmp, size);
		countRx-=size;
	}
}
///
void usbTransmit(uint8_t id, uint8_t* readPointer, char* buffer, uint8_t* counter, uint8_t size){
	blinkRX();
	tmp[0]=id;
	for(int i=2;i<size+2;i++){//tmp[0] - id, tmp[1] - size, tmp[2...31] - data
		tmp[i]=buffer[*readPointer];
		*readPointer=(*readPointer+1)%BUF_SIZE;
		if(i==(USB_PACKET_SIZE-1)){
			tmp[1]=USB_PACKET_SIZE-2;
			USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, (uint8_t*)tmp,
USB_PACKET_SIZE);
			*counter-=tmp[1];
			size-=tmp[1];
			i=1;
		}
	}
	if(size!=0){
		tmp[1]=size;
		USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, (uint8_t*)tmp, USB_PACKET_SIZE);
		*counter-=size;
	}
}
///
void uartTransmit(uint8_t size){
	blinkTX();
	for(int i=0;i<size;i++){
		tmp[i]=uart_tx[readPointerTx];
		readPointerTx=(readPointerTx+1)%BUF_SIZE;
		if(i==31){
			HAL_UART_Transmit(&huart1, (uint8_t*)tmp, 32,0xFF);
			countTx-=32;
			size-=32;
			i=-1;
		}
	}
	if(size!=0){
		countTx-=size;
		HAL_UART_Transmit(&huart1, (uint8_t*)tmp, size,0xFF);
	}
}
///SPI////
void spiTransmit(uint8_t size){
	blinkTX();
	for(uint8_t i=0; i<size;i+=4){
```

```
                HAL_SPI_TransmitReceive(&hspi1, (uint8_t*)&spi_tx[spiReadPointerTx],
                                                        (uint8_t*)&spi_rx[spiWritePointerRx], 4, 0xFF);
                spiReadPointerTx=(spiReadPointerTx+4)%BUF_SIZE;
                spiWritePointerRx=(spiWritePointerRx+4)%BUF_SIZE;
        }
        spiCountRx+=size;
        spiCountTx-=size;
}
/* USER CODE END 0 */
int main(void)
{
 /* MCU Configuration--------------------------------------------------------*/
 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();
 /* Configure the system clock */
 SystemClock_Config();
 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_SPI1_Init();
 MX_TIM3_Init();
 MX_TIM14_Init();
 MX_USART1_UART_Init();
 MX_USB_DEVICE_Init();
 /* USER CODE BEGIN 2 */
 HAL_TIM_Base_Start_IT(&htim14);
 HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
 HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4);
 HAL_UART_Receive_IT(&huart1, (uint8_t*)uart_rx, BUF_SIZE);
 /* USER CODE END 2 */
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
                if(rxLed){
                        rxLed--;
                        if(rxLed==0)
                                TIM3->CCR4 = 0;
                }
                if(txLed){
                        txLed--;
                        if(txLed==0)
                                TIM3->CCR3 = 0;
                }
                //send received data to uart
                uint8_t cn = countTx;
                if(cn>0){
                        uartTransmit(cn);
                }
                //receive data from uart
                cn = countRx;
                if(cn>0){
                        if(uartBusy){
                                usbTransmit(USB_UART_RX, &readPointerRx, uart_rx, &countRx, cn);
                        }else{
                                cdcTransmit(cn);//use cdc for data from uart
                        }
                }
                ////SPI////
                cn=spiCountTx;
                if(cn>=4){
                        uint8_t size = cn-(cn%4);
                        spiTransmit(size);
```

```
                        usbTransmit(USB_SPI_RX, &spiReadPointerRx, spi_rx, &spiCountRx, size);
                }


                HAL_Delay(1);
  /* USER CODE END WHILE */
  }
}
/** System Clock Configuration*/
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInit;
  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.PLLMUL = RCC_PLL_MUL6;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USB|RCC_PERIPHCLK_USART1;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK1;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}



/**
  *
  ================================================================================
  *                          composite CDC_HID
  *
  ================================================================================
    */
/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc_hid.h"
/**   CDC Device library callbacks**/
extern uint8_t  USBD_CDC_Init (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
extern uint8_t  USBD_CDC_DeInit (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
extern uint8_t  USBD_CDC_Setup (USBD_HandleTypeDef *pdev, USBD_SetupReqTypedef *req);
extern uint8_t  USBD_CDC_DataIn (USBD_HandleTypeDef *pdev, uint8_t epnum);
```

```
extern uint8_t  USBD_CDC_DataOut (USBD_HandleTypeDef *pdev, uint8_t epnum);
extern uint8_t  USBD_CDC_EP0_RxReady (USBD_HandleTypeDef *pdev);
extern uint8_t  *USBD_CDC_GetFSCfgDesc (uint16_t *length);
extern uint8_t  *USBD_CDC_GetHSCfgDesc (uint16_t *length);
extern uint8_t  *USBD_CDC_GetOtherSpeedCfgDesc (uint16_t *length);
extern uint8_t  *USBD_CDC_GetDeviceQualifierDescriptor (uint16_t *length);
/**Custon HID Device library callbacks**/
extern uint8_t  USBD_CUSTOM_HID_Init (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
extern uint8_t  USBD_CUSTOM_HID_DeInit (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
extern uint8_t  USBD_CUSTOM_HID_Setup (USBD_HandleTypeDef *pdev, USBD_SetupReqTypedef *req);
extern uint8_t  *USBD_CUSTOM_HID_GetCfgDesc (uint16_t *length);
extern uint8_t  *USBD_CUSTOM_HID_GetDeviceQualifierDesc (uint16_t *length);
extern uint8_t  USBD_CUSTOM_HID_DataIn (USBD_HandleTypeDef *pdev, uint8_t epnum);
extern uint8_t  USBD_CUSTOM_HID_DataOut (USBD_HandleTypeDef *pdev, uint8_t epnum);
extern uint8_t  USBD_CUSTOM_HID_EP0_RxReady (USBD_HandleTypeDef  *pdev);
/**CDC_Custon HID Device library callbacks**/
static uint8_t USBD_CDC_HID_Init (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
static uint8_t USBD_CDC_HID_DeInit (USBD_HandleTypeDef *pdev, uint8_t cfgidx);
static uint8_t USBD_CDC_HID_Setup (USBD_HandleTypeDef *pdev, USBD_SetupReqTypedef *req);
static uint8_t USBD_CDC_HID_DataIn (USBD_HandleTypeDef *pdev, uint8_t epnum);
static uint8_t USBD_CDC_HID_DataOut (USBD_HandleTypeDef *pdev, uint8_t epnum);
static uint8_t USBD_CDC_HID_EP0_RxReady (USBD_HandleTypeDef *pdev);
static uint8_t *USBD_CDC_HID_GetCfgDesc (uint16_t *length);
static uint8_t *USBD_CDC_HID_GetDeviceQualifierDesc (uint16_t *length);
USBD_CDC_HID_ItfTypeDef CDC_HID =
{
        &USBD_CustomHID_fops_FS,
        &USBD_Interface_fops_FS,
};
/* CDC_HID interface class callbacks structure */
USBD_ClassTypeDef  USBD_CDC_HID =
{
        USBD_CDC_HID_Init,
        USBD_CDC_HID_DeInit,
        USBD_CDC_HID_Setup,
        NULL,              /* EP0_TxSent, */
        USBD_CDC_HID_EP0_RxReady, /*EP0_RxReady*/ /* STATUS STAGE IN */
        USBD_CDC_HID_DataIn,    /*DataIn*/
        USBD_CDC_HID_DataOut,
        NULL,    /*SOF */
        NULL,
        NULL,
        USBD_CDC_HID_GetCfgDesc,
        USBD_CDC_HID_GetCfgDesc,
        USBD_CDC_HID_GetCfgDesc,
        USBD_CDC_HID_GetDeviceQualifierDesc,
};
/* USB CDC_HID device Configuration Descriptor */
__ALIGN_BEGIN static uint8_t USBD_CDC_HID_CfgDesc[USB_CDC_HID_CONFIG_DESC_SIZ]
__ALIGN_END =
{
        0x09, /* bLength: Configuration Descriptor size */
        USB_DESC_TYPE_CONFIGURATION, /* bDescriptorType: Configuration */
        USB_CDC_HID_CONFIG_DESC_SIZ,
        /* wTotalLength: Bytes returned */
        0x00,
        0x03,       /*bNumInterfaces: 3 interfaces (2 for CDC, 1 for HID)*/
        0x01,       /*bConfigurationValue: Configuration value*/
        0x00,       /*iConfiguration: Index of string descriptor describing the configuration*/
        0xC0,       /*bmAttributes: bus powered */
        0x32,       /*MaxPower 100 mA: this current is used for detecting Vbus*/
        /********** Descriptor of CUSTOM HID interface **********/
```

```
0x09,       /*bLength: Interface Descriptor size*/
USB_DESC_TYPE_INTERFACE,/*bDescriptorType: Interface descriptor type*/
HID_INTERFACE,       /*bInterfaceNumber: Number of Interface*/
0x00,       /*bAlternateSetting: Alternate setting*/
0x02,       /*bNumEndpoints*/
0x03,       /*bInterfaceClass: CUSTOM_HID*/
0x00,       /*bInterfaceSubClass : 1=BOOT, 0=no boot*/
0x00,       /*nInterfaceProtocol : 0=none, 1=keyboard, 2=mouse*/
0,          /*iInterface: Index of string descriptor*/
/********************* Descriptor of CUSTOM_HID **************************/
0x09,       /*bLength: CUSTOM_HID Descriptor size*/
CUSTOM_HID_DESCRIPTOR_TYPE, /*bDescriptorType: CUSTOM_HID*/
0x11,       /*bCUSTOM_HIDUSTOM_HID: CUSTOM_HID Class Spec release number*/
0x01,
0x00,       /*bCountryCode: Hardware target country*/
0x01,       /*bNumDescriptors: Number of CUSTOM_HID class descriptors to follow*/
0x22,       /*bDescriptorType*/
USBD_CUSTOM_HID_REPORT_DESC_SIZE,/*wItemLength: Total length of Report descriptor*/
0x00,
/********************* Descriptor of Custom HID endpoints *********************/
0x07,        /*bLength: Endpoint Descriptor size*/
USB_DESC_TYPE_ENDPOINT, /*bDescriptorType:*/
CUSTOM_HID_EPIN_ADDR,     /*bEndpointAddress: Endpoint Address (IN)*/
0x03,        /*bmAttributes: Interrupt endpoint*/
CUSTOM_HID_EPIN_SIZE, /*wMaxPacketSize: 64 Byte max */
0x00,
0x01,        /*bInterval: Polling Interval (1 ms)*/
0x07,            /* bLength: Endpoint Descriptor size */
USB_DESC_TYPE_ENDPOINT,  /* bDescriptorType: */
CUSTOM_HID_EPOUT_ADDR,  /*bEndpointAddress: Endpoint Address (OUT)*/
0x03,     /* bmAttributes: Interrupt endpoint */
CUSTOM_HID_EPOUT_SIZE,    /* wMaxPacketSize: 2 Bytes max  */
0x00,
0x01,     /* bInterval: Polling Interval (1ms) */
/******** /IAD should be positioned just before the CDC interfaces  IAD to associate the two CDC
interfaces */
0x08, /* bLength */
0x0B, /* bDescriptorType */
0x01, /* bFirstInterface */
0x02, /* bInterfaceCount */
0x02, /* bFunctionClass */
0x02, /* bFunctionSubClass */
0x01, /* bFunctionProtocol */
0x02, /* iFunction (Index of string descriptor describing this function) */
/************************  CDC interface ********************************/
0x09,   /* bLength: Interface Descriptor size */
USB_DESC_TYPE_INTERFACE,  /* bDescriptorType: Interface */
/* Interface descriptor type */
CDC_INTERFACE0,   /* bInterfaceNumber: Number of Interface */
0x00,   /* bAlternateSetting: Alternate setting */
0x01,   /* bNumEndpoints: One endpoints used */
0x02,   /* bInterfaceClass: Communication Interface Class */
0x02,   /* bInterfaceSubClass: Abstract Control Model */
0x01,   /* bInterfaceProtocol: Common AT commands */
0x00,   /* iInterface: */
/*Header Functional Descriptor*/
0x05,   /* bLength: Endpoint Descriptor size */
0x24,   /* bDescriptorType: CS_INTERFACE */
0x00,   /* bDescriptorSubtype: Header Func Desc */
0x10,   /* bcdCDC: spec release number */
0x01,
/*Call Management Functional Descriptor*/
```

```
            0x05,   /* bFunctionLength */
            0x24,   /* bDescriptorType: CS_INTERFACE */
            0x01,   /* bDescriptorSubtype: Call Management Func Desc */
            0x00,   /* bmCapabilities: D0+D1 */
            0x01,   /* bDataInterface: 1 */
            /*ACM Functional Descriptor*/
            0x04,   /* bFunctionLength */
            0x24,   /* bDescriptorType: CS_INTERFACE */
            0x02,   /* bDescriptorSubtype: Abstract Control Management desc */
            0x02,   /* bmCapabilities */
            /*Union Functional Descriptor*/
            0x05,   /* bFunctionLength */
            0x24,   /* bDescriptorType: CS_INTERFACE */
            0x06,   /* bDescriptorSubtype: Union func desc */
            0x00,   /* bMasterInterface: Communication class interface */
            0x01,   /* bSlaveInterface0: Data Class Interface */
            /*Endpoint 2 Descriptor*/
            0x07,                   /* bLength: Endpoint Descriptor size */
            USB_DESC_TYPE_ENDPOINT,   /* bDescriptorType: Endpoint */
            CDC_CMD_EP,               /* bEndpointAddress */
            0x03,                   /* bmAttributes: Interrupt */
            LOBYTE(CDC_CMD_PACKET_SIZE),     /* wMaxPacketSize: */
            HIBYTE(CDC_CMD_PACKET_SIZE),
            0x10,                   /* bInterval: */
            /*Data class interface descriptor*/
            0x09,   /* bLength: Endpoint Descriptor size */
            USB_DESC_TYPE_INTERFACE,  /* bDescriptorType: */
            CDC_INTERFACE1,   /* bInterfaceNumber: Number of Interface */
            0x00,   /* bAlternateSetting: Alternate setting */
            0x02,   /* bNumEndpoints: Two endpoints used */
            0x0A,   /* bInterfaceClass: CDC */
            0x00,   /* bInterfaceSubClass: */
            0x00,   /* bInterfaceProtocol: */
            0x00,   /* iInterface: */
            /*Endpoint OUT Descriptor*/
            0x07,   /* bLength: Endpoint Descriptor size */
            USB_DESC_TYPE_ENDPOINT,      /* bDescriptorType: Endpoint */
            CDC_OUT_EP,                  /* bEndpointAddress */
            0x02,                   /* bmAttributes: Bulk */
            LOBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),  /* wMaxPacketSize: */
            HIBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),
            0x00,                   /* bInterval: ignore for Bulk transfer */
            /*Endpoint IN Descriptor*/
            0x07,   /* bLength: Endpoint Descriptor size */
            USB_DESC_TYPE_ENDPOINT,      /* bDescriptorType: Endpoint */
            CDC_IN_EP,                   /* bEndpointAddress */
            0x02,                   /* bmAttributes: Bulk */
            LOBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),  /* wMaxPacketSize: */
            HIBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),
            0x00                    /* bInterval: ignore for Bulk transfer */
} ;
/* USB Standard Device Descriptor */
__ALIGN_BEGIN static uint8_t USBD_CDC_HID_DeviceQualifierDesc[USB_LEN_DEV_QUALIFIER_DESC]
__ALIGN_END =
{
  USB_LEN_DEV_QUALIFIER_DESC,
  USB_DESC_TYPE_DEVICE_QUALIFIER,
  0x00,
  0x02,
  0x00,
  0x00,
  0x00,
```

```
  0x40,
  0x01,
  0x00,
};
/**
  * @brief  USBD_CDC_HID_Init
  *         Initialize the CDC and HID interfaces
  */
static uint8_t USBD_CDC_HID_Init (USBD_HandleTypeDef *pdev, uint8_t cfgidx)
{
          /* HID initialization */
          USBD_CUSTOM_HID_Init(pdev, cfgidx);
          /* CDC initialization */
          USBD_CDC_Init(pdev, cfgidx);
          return USBD_OK;
}
/**
  * @brief  USBD_CDC_HID_DeInit
  *         DeInitialize the CDC and HID interfaces
  * @param  pdev: device instance
  * @param  cfgidx: Configuration index
  * @retval status
  */
static uint8_t USBD_CDC_HID_DeInit (USBD_HandleTypeDef *pdev, uint8_t cfgidx)
{
          /* HID deinitialization */
          USBD_CUSTOM_HID_DeInit(pdev, cfgidx);
          /* CDC deinitialization */
          USBD_CDC_DeInit(pdev, cfgidx);
          return USBD_OK;
}
/**
  * @brief  USBD_CDC_HID_Setup
  *         Handle the CDC and HID specific requests
  * @param  pdev: instance
  * @param  req: USBD requests
  * @retval status
*/
static uint8_t USBD_CDC_HID_Setup (USBD_HandleTypeDef *pdev, USBD_SetupReqTypedef *req)
{
          if (req->wIndex == HID_INTERFACE)
                    return (USBD_CUSTOM_HID_Setup (pdev, req));
          else
                    return (USBD_CDC_Setup(pdev, req));
}
/**
  * @brief  USBD_CDC_HID_DataIn
  *         handle data IN Stage
  * @param  pdev: device instance
  * @param  epnum: endpoint index
  * @retval status
 */
static uint8_t USBD_CDC_HID_DataIn (USBD_HandleTypeDef *pdev, uint8_t epnum)
{
          /*DataIN can be for CDC or HID */
          if (epnum == (CUSTOM_HID_EPIN_ADDR&~0x80))
                    return (USBD_CUSTOM_HID_DataIn(pdev, epnum));
          else
                    return (USBD_CDC_DataIn(pdev, epnum));
}
/**
  * @brief  USBD_CDC_HID_DataOut
```

```
  *        handle data OUT Stage
  * @param  pdev: device instance
  * @param  epnum: endpoint index
  * @retval status
  */
static uint8_t USBD_CDC_HID_DataOut (USBD_HandleTypeDef *pdev, uint8_t epnum)
{
          /*DataOUT can be for CDC or HID */
          if (epnum == (CUSTOM_HID_EPOUT_ADDR&~0x80))
                    return (USBD_CUSTOM_HID_DataOut(pdev, epnum));
          else
                    return (USBD_CDC_DataOut(pdev, epnum));
}
/**
  * @brief  USBD_CDC_HID_EP0_RxReady
  *        Handles control request data.
  * @param  pdev: device instance
  * @retval status
*/
static uint8_t USBD_CDC_HID_EP0_RxReady (USBD_HandleTypeDef *pdev)
{
          USBD_CUSTOM_HID_EP0_RxReady(pdev);
          USBD_CDC_EP0_RxReady(pdev);
          return USBD_OK;
}
/**
  * @brief  USBD_CDC_HID_GetCfgDesc
  *        return configuration descriptor
  * @param  length : pointer data length
  * @retval pointer to descriptor buffer
  */
static uint8_t *USBD_CDC_HID_GetCfgDesc (uint16_t *length)
{
          *length = sizeof (USBD_CDC_HID_CfgDesc);
          return USBD_CDC_HID_CfgDesc;
}
/**
* @brief  DeviceQualifierDescriptor
*        return Device Qualifier descriptor
* @param  length : pointer data length
* @retval pointer to descriptor buffer
*/
static uint8_t *USBD_CDC_HID_GetDeviceQualifierDesc (uint16_t *length)
{
          *length = sizeof (USBD_CDC_HID_DeviceQualifierDesc);
          return USBD_CDC_HID_DeviceQualifierDesc;
}
/**
* @brief  USBD_CDC_HID_RegisterInterface
  */
uint8_t  USBD_CDC_HID_RegisterInterface  (USBD_HandleTypeDef   *pdev)
{
          pdev->pUserData= &CDC_HID;
  USBD_CUSTOM_HID_RegisterInterface(pdev, &USBD_CustomHID_fops_FS);
          USBD_CDC_RegisterInterface(pdev, &USBD_Interface_fops_FS);

          return USBD_OK;
}



 / * @file          : usbd_custom_hid_if.c
```

```
  * @brief        : USB Device Custom HID interface file.*/
/* Includes ------------------------------------------------------------------*/
#include "usbd_custom_hid_if.h"
/* USER CODE BEGIN INCLUDE */
#include "Handler.h"

__ALIGN_BEGIN static uint8_t
CUSTOM_HID_ReportDesc_FS[USBD_CUSTOM_HID_REPORT_DESC_SIZE] __ALIGN_END =
{
  /* USER CODE BEGIN 0 */
          //from PC
  0x06,0x00,0xFF,   //USAGE_PAGE (GENERIC DESCTOP)
    0x09, 0x01,                 // USAGE (Vendor Usage 1)
    0xa1, 0x01,                 // COLLECTION (Application)
    0x85, 0x01,                 //   REPORT_ID (1)
    0x09, 0x01,                 //   USAGE (Vendor Usage 1)
    0x15, 0x00,                 //   LOGICAL_MINIMUM (0)
    0x25, 0xFF,                 //   LOGICAL_MAXIMUM (1)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32)
    0xB1, 0x82,                 //   FEATURE (Data,Var,Abs,Vol)
    0x85, 0x01,                 //   REPORT_ID (1)
    0x09, 0x01,                 //   USAGE (Vendor Usage 1)
    0x91, 0x82,                 //   OUTPUT (Data,Var,Abs,Vol)
0x85, 0x03,                   //   REPORT_ID (3)
    0x09, 0x03,                 //   USAGE (Vendor Usage 3)
    0x15, 0x00,                 //   LOGICAL_MINIMUM (0)
    0x25, 0xFF,                 //   LOGICAL_MAXIMUM (255)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32)
    0xB1, 0x82,                 //   FEATURE (Data,Var,Abs,Vol)
    0x85, 0x03,                 //   REPORT_ID (3)
    0x09, 0x03,                 //   USAGE (Vendor Usage 3)
    0x91, 0x82,                 //   OUTPUT (Data,Var,Abs,Vol)
0x85, 0x05,                   //   REPORT_ID (5)
    0x09, 0x05,                 //   USAGE (Vendor Usage 5)
    0x15, 0x00,                 //   LOGICAL_MINIMUM (0)
    0x25, 0xFF,                 //   LOGICAL_MAXIMUM (255)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32)
    0xB1, 0x82,                 //   FEATURE (Data,Var,Abs,Vol)
    0x85, 0x05,                 //   REPORT_ID (5)
    0x09, 0x05,                 //   USAGE (Vendor Usage 5)
    0x91, 0x82,                 //   OUTPUT (Data,Var,Abs,Vol)
//to PC
0x85, 0x02,                   //   REPORT_ID (2)
    0x09, 0x02,                 //   USAGE (Vendor Usage 2)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32)
    0x81, 0x82,                 //INPUT (Data,Var,Abs,Vol)
    0x85, 0x04,                 //   REPORT_ID (4)
    0x09, 0x04,                 //   USAGE (Vendor Usage 4)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32)
    0x81, 0x82,                 //INPUT (Data,Var,Abs,Vol)
    0x85, 0x06,                 //   REPORT_ID (6)
    0x09, 0x06,                 //   USAGE (Vendor Usage 6)
    0x75, 0x08,                 //   REPORT_SIZE (8)
    0x95, 0x1F,                 //   REPORT_COUNT (32) 0x04
    0x81, 0x82,                 //   INPUT (Data,Var,Abs,Vol)
  /* USER CODE END 0 */
  0xC0    /*     END_COLLECTION                     */
```

```
};
/* USER CODE BEGIN PRIVATE_VARIABLES */
extern char uart_tx[BUF_SIZE], spi_tx[BUF_SIZE];
extern uint8_t countTx, writePointerTx, spiCountTx, spiWritePointerTx;
/* USER CODE END PRIVATE_VARIABLES */
 extern USBD_HandleTypeDef hUsbDeviceFS;
/* USER CODE BEGIN EXPORTED_VARIABLES */
extern uint8_t uartBusy;
/* USER CODE END EXPORTED_VARIABLES */
/** @defgroup USBD_CUSTOM_HID_Private_FunctionPrototypes
  * @{
  */
static int8_t CUSTOM_HID_Init_FS     (void);
static int8_t CUSTOM_HID_DeInit_FS   (void);
static int8_t CUSTOM_HID_OutEvent_FS (uint8_t event_idx, uint8_t state);
USBD_CUSTOM_HID_ItfTypeDef USBD_CustomHID_fops_FS =
{
  CUSTOM_HID_ReportDesc_FS,
  CUSTOM_HID_Init_FS,
  CUSTOM_HID_DeInit_FS,
  CUSTOM_HID_OutEvent_FS,
};
/* Private functions ---------------------------------------------------------*/
/**
  * @brief  CUSTOM_HID_Init_FS
  *         Initializes the CUSTOM HID media low layer
  * @param  None
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CUSTOM_HID_Init_FS(void)
{
  /* USER CODE BEGIN 4 */
  return (0);
  /* USER CODE END 4 */
}
/**
  * @brief  CUSTOM_HID_DeInit_FS
  *         DeInitializes the CUSTOM HID media low layer
  * @param  None
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CUSTOM_HID_DeInit_FS(void)
{
  /* USER CODE BEGIN 5 */
  return (0);
  /* USER CODE END 5 */
}
/**
  * @brief  CUSTOM_HID_OutEvent_FS
  *         Manage the CUSTOM HID class events
  */
static int8_t CUSTOM_HID_OutEvent_FS  (uint8_t event_idx, uint8_t state)
{
  /* USER CODE BEGIN 6 */
            USBD_CUSTOM_HID_HandleTypeDef *hhid =
(USBD_CUSTOM_HID_HandleTypeDef*)hUsbDeviceFS.pClassData;
            uint8_t id = hhid->Report_buf[0];
            uint8_t informByte = hhid->Report_buf[1];
            if(id==USB_SETTINGS){
                        uint8_t tmp[USB_PACKET_SIZE] = "\0";
                        tmp[0]=USB_ANSWER;
                        if(informByte==UART_BUSY_RESET || informByte==UART_BUSY_SET)//0|1
```

```
                uartBusy = informByte;
            else if (informByte==CS_SET)//2
                cs_set();
            else if (informByte==CS_RESET)//3
                cs_reset();
            else if (informByte==USB_DISCONNECT){//4
                cs_reset();
                uartBusy = UART_BUSY_RESET;
            }
            tmp[1]=informByte;
            USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, tmp, USB_PACKET_SIZE);
        }else if (id==USB_UART_TX){
            for(uint8_t i=2;i<informByte+2;i++){//informByte contains length
                uart_tx[writePointerTx]=hhid->Report_buf[i];
                writePointerTx=(writePointerTx+1)%BUF_SIZE;
                countTx++;
            }
        }else if (id==USB_SPI_TX){
            for(uint8_t i=2;i<informByte+2;i++){//informByte contains length
                spi_tx[spiWritePointerTx]=hhid->Report_buf[i];
                spiWritePointerTx=(spiWritePointerTx+1)%BUF_SIZE;
                spiCountTx++;
            }
        }
  return (0);
  /* USER CODE END 6 */
}
/* USER CODE BEGIN 7 */
/**
  * @brief  USBD_CUSTOM_HID_SendReport_FS
  *         Send the report to the Host
/*
static int8_t USBD_CUSTOM_HID_SendReport_FS ( uint8_t *report,uint16_t len)
{
  return USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, report, len);
}
*/
/* USER CODE END 7 */


/**
  *
==========================================================================
  *                      Handler.c
  *
==========================================================================
   */
#include "Handler.h"
extern uint8_t countRx;
extern uint8_t writePointerRx;

void USER_UART_IRQHandler(UART_HandleTypeDef *huart){
        /* UART in mode Receiver ---------------------------------------------------*/
  if((__HAL_UART_GET_IT(huart, UART_IT_RXNE) != RESET) &&
(__HAL_UART_GET_IT_SOURCE(huart, UART_IT_RXNE) != RESET))
  {
        /* Check that a Rx process is ongoing */
            if(huart->RxState == HAL_UART_STATE_BUSY_RX)
            {
                huart->pRxBuffPtr[writePointerRx] = (uint8_t)(huart->Instance->RDR & huart->Mask);
                writePointerRx=(writePointerRx+1)%BUF_SIZE;
                countRx++;
```

```
                        if(huart->RxXferCount == 0)
                        {
                                __HAL_UART_DISABLE_IT(huart, UART_IT_RXNE);
                                /* Disable the UART Parity Error Interrupt */
                                __HAL_UART_DISABLE_IT(huart, UART_IT_PE);
                                /* Disable the UART Error Interrupt: (Frame error, noise error, overrun error)
*/
                                __HAL_UART_DISABLE_IT(huart, UART_IT_ERR);
                                /* Rx process is completed, restore huart->RxState to Ready */
                                huart->RxState = HAL_UART_STATE_READY;
                                HAL_UART_RxCpltCallback(huart);
                                return;
                        }
                        return;
                }
  }
  /* UART in mode Transmitter ------------------------------------------------*/
  if((__HAL_UART_GET_IT(huart, UART_IT_TXE) != RESET) &&(__HAL_UART_GET_IT_SOURCE(huart,
UART_IT_TXE) != RESET))
  {
    UART_Transmit_IT(huart);
  }
  /* UART in mode Transmitter (transmission end) -----------------------------*/
 if((__HAL_UART_GET_IT(huart, UART_IT_TC) != RESET) &&(__HAL_UART_GET_IT_SOURCE(huart,
UART_IT_TC) != RESET))
  {
    UART_EndTransmit_IT(huart);
  }
}
```