МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Обслуговування комп'ютерних систем і мереж»

Група: 4КС-55

Дипломний проект

здобувача освіти денної форми навчання
КС.55.21.000.ДП

СИТНІКОВА
АНДРІЯ СЕРГІЙОВИЧА

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: «**Обслуговування комп'ютерних систем і мереж»**

Група: **4КС-55**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR

Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах).

Дипломник _____ (Ситніков А.С.)

Керівник _____ (Кірєєв І.А.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Скорнякова О.В.)

До захисту допущений

Голова циклової комісії _____ (Скорнякова О.В.)

Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____

Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ІІІ**

Спеціальність **123 «Комп'ютерна інженерія»**

Освітня програма **«Обслуговування комп'ютерних систем і мереж»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР

Беркань І.В.

"_____" _____ 2022 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Ситнікову Андрію Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR

затверджена наказом по коледжу від "____" _____ 202___ р. № _____

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) 1. Швидкість передачі даних через порт USB та SPI – до 6 Мбіт/с;
2. Кількість переданих біт в посилці – 8, 9 біт; режим роботи – дуплексний;
3. Тактова частота роботи програматору – 12 МГц;
4. Рівні логічних сигналів при роботі програматора – CMOS, USB;
5. Операційна система для середовища програмування – Linux, Windows.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1.1. Основні функції та архітектура мікроконтролерів сімейства AVR

1.2. Засоби програмування мікроконтролерів сімейства AVR

1.3. Розробка структурної схеми програматору для мікроконтролерів ATMEL з ядром AVR

1.4. Розробка принципової електричної схеми програматору для мікроконтролерів ATMEL

1.5 Розробка програми ініціалізації програматору для мікроконтролерів ATMEL з ядром AVR

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Базова структура мікроконтролера AVR. Структурна схема мікроконтролера сімейства AVR. Взаємодія прикладної системи із середовищем розробки. Елементи середовища розробки. Середовище розробки CodeVisionAVR. Структурна схема розроблюваного програматору. Загальна блок-схема USB-контролеру. Стани контролера USB після скидання. Послідовність дій для активації каналу. Функціональна схема інтерфейсу SPI. Схема електрична принципова розроблюваного програматору. БСА програми для ініціалізації програматору.

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	Кірєєв І.А.		
розділ			
2.Екон. частина	Копайгородська Т.Г.		
3.Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		

7. Дата видачі завдання _____

Керівник Кірєєв І.А. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	Вступ. Постановка задачі проектування		
2.	Класифікація програматорів		
3.	Способи підключення програматору до ПК		
4.	Огляд програмного забезпечення програматору		
5.	Вивчення особливостей програмування USB-інтерфейсу		
6.	Вивчення послідовного периферійного інтерфейсу SPI		
7.	Короткий опис процесора ATMega8		
8.	Загальний опис схеми ISP-програматору		
9.	Установка драйверів і програм		
10.	Вибір програматора в програмі		
11.	Розрахунок струму споживання ISP-програматору		
12.	Розробка програми ініціалізації ISP-програматору		
13.	Розрахунок собівартості ISP-програматору		
14.	Виконання розділу охорони праці та ТБ		
15.	Виконання графічної частини проекту		
16.	Підготовка проекту до захисту		

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 55. 21 000. 00 ДП	Дипломний проект		
А4			КС 55. 21 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 55. 21 000. 00 ДП		
Розробив		Ситніков А.С.			Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR	Літ.	Аркуш 5 / Аркушів 60
Перевірив		Кірєєв І.А.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4КС-55	
Затверд.		Скорнякова О.В.					

ЗМІСТ

ВСТУП

Застосування мікроконтролерів можна розділити на два етапи: перший – програмування, коли користувач розробляє програму і прошиває її безпосередньо у кристал, і другий – узгодження спроектованих виконавчих пристроїв із запрограмованим мікроконтролером. Значно полегшує відлагодження програми на першому етапі симулятор, який наочно моделює роботу мікропроцесора. На другому етапі для відлагодження використовується внутрішньосхемний емулятор, який є складним і дорогим пристроєм.

Дипломний проект присвячений програмуванню мікроконтролерних пристроїв, які мають SPI-інтерфейс. У даній роботі виконується розробка апаратних-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR.

Під час роботи над дипломним проектом необхідно вивчити принципи побудови та способи програмування мікроконтролерів. Проект має включати структурну і функціональну схему пристрою, схему електричну принципову, її опис, алгоритм та програму ініціалізації пристрою, а також розрахунок струму живлення пристрою. Результатом роботи має бути пакет документації для створення програматору для мікроконтролерів ATMEL з ядром AVR.

Інтерфейс USB є стандартним інтерфейсом для зв'язку між електронними пристроями (OTG) і електронними пристроями з ПК. Один із способів реалізації USB-інтерфейсу – це використання мікроконтролерів з апаратною підтримкою цього інтерфейсу. Тому основні виробники мікроконтролерів:

- Atmel Corporation;
- STMicroelectronics;
- Microchip - Freescale Semiconductor,

що розробили серії мікроконтролерів, які підтримують USB, і, звичайно, "рекомендації", що полегшують розробникам реалізацію власних пристроїв.

Декілька основних переваг інтерфейсу USB:

- При використанні USB не вимагається виконувати процедури інсталяції і конфігурації ні для інтерфейсної карти, ні для самого пристрою; USB не викликає конфліктів по перериваннях, відпадає необхідність в драйверах пристроїв;

- USB-концентратори забезпечують додаткові USB-порти (більшість ПК підтримують тільки два USB-порти). Ці порти дозволяють за бажанням користувача підключати найрізноманітніші пристрої, такі як клавіатура, миша, принтер, сканер, зовнішні накопичувачі, джойстик і так далі.

Основною метою дипломного проектування є:

- розглянути і вивчити процес програмування мікроконтролерів сімейства ATMEL з ядром AVR;
- підвищити ефективність використання програматору для мікроконтролерів ATMEL з ядром AVR.

Для досягнення поставленої мети необхідно вирішити ряд задач:

1. Виконати аналіз концепції побудови програматорів для мікроконтролерів;

2. Розробити структурну схему програматору для мікроконтролерів ATMEL з ядром AVR;

3. Розробити схему електричну принципову програматору для мікроконтролерів ATMEL з ядром AVR;

4. Розрахувати величини струму споживання програматору для мікроконтролерів ATMEL з ядром AVR;

5. Розробити програму ініціалізації програматору.

Актуальність поставленої задачі підтверджується Законом України «Про Концепцію Національної програми інформатизації», «Про Національну програму інформатизації», «Про основні засади розвитку інформаційного суспільства в Україні», а також рекомендаціями Міжнародного союзу електрозв'язку стандартами 802.1-802.12, в яких особлива увага приділена інформаційним технологіям.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Основні функції та архітектура мікроконтролерів сімейства AVR

Загальна структура мікроконтролера показана на рис. 1.1. Ця структура дає уявлення про те, як мікроконтролер зв'язується із зовнішнім світом.

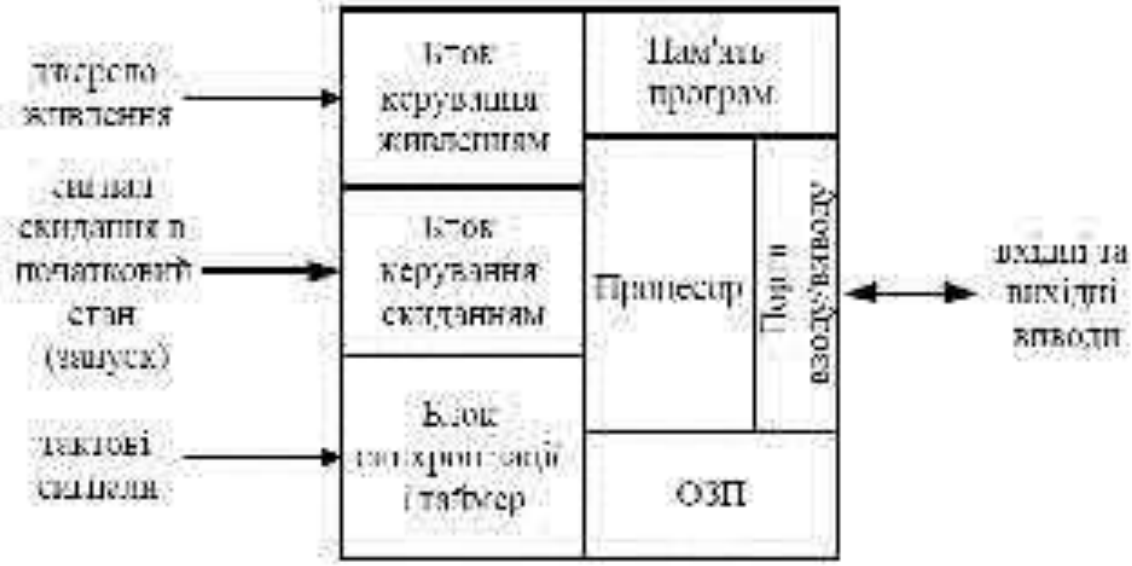

Рисунок 1.1. Базова структура мікроконтролера AVR

Відмінні риси мікроконтролерів сімейства AVR наступні:

- продуктивність, близька до 1 MIPS/МГц;
- вдосконалена RISC архітектура;
- розділені шини пам'яті команд і даних, 32 регістра загального призначення;
- Flash ПЗП програм з можливістю внутрішньосхемного перепрограмування, 1000 циклів стирання/запису;
- блокування режиму програмування;
- вбудований аналоговий компаратор, сторожовий таймер;
- повністю статичні пристрої – працюють при тактовій частоті від 0 Гц до 20 МГц;
- діапазон напруги живлення від 1,8 В до 6,0 В;
- режими енергозбереження: стоповий та пасивний [1].

Мікроконтролери сімейства AVR мають єдину базову структуру. Узагальнена структурна схема мікроконтролера (МК) зображена на рис. 1.2.

До складу мікроконтролера входять:

- генератор тактового сигналу (GCK);

- процесор (CPU);

- постійний запам'ятовуючий пристрій для зберігання програми, виконаної за технологією Flash, (Flash ROM);

- оперативний запам'ятовуючий пристрій статичного типу для зберігання даних (SRAM);

- постійний запам'ятовуючий пристрій для зберігання даних (EEPROM);

- набір периферійних пристроїв для вводу та виводу даних, керуючих сигналів та виконання інших функцій [3].

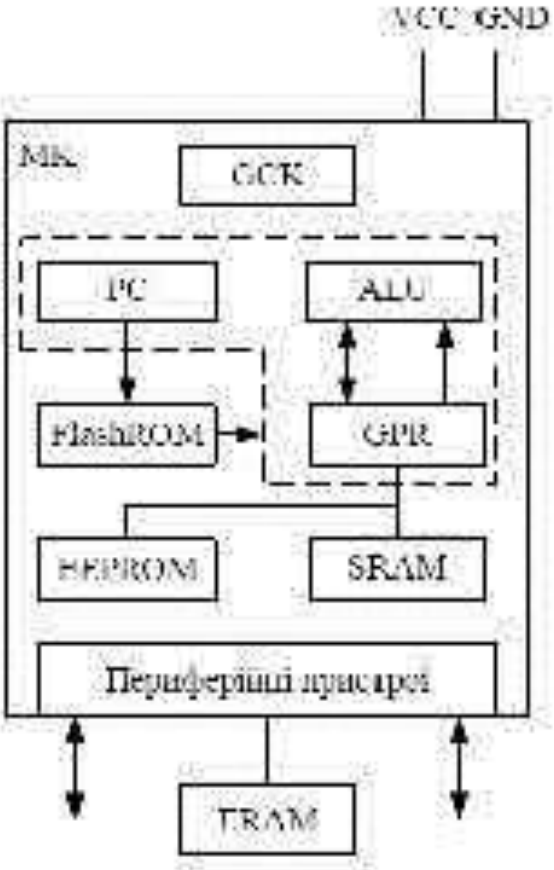

Рисунок 1.2. Структурна схема мікроконтролера сімейства AVR

У МК серії AVR використовується принципи так званої RISC-архітектури, коли пам'ять програм і пам'ять даних, з якими програма оперує, не тільки

розділені фізично, але й мають різні канали зв'язку з ядром (різні шини). Окрім цього, є ще одна цікава особливість, яка називається конвеєризацією. Конвеєризація дозволяє одночасно і виконувати команду, і готувати до виконання наступну. Гарвардська архітектура, а також конвеєризація і деякі інші спеціальні заходи дозволяють виконувати інструкції дуже швидко – за один машинний цикл. Це, в свою чергу, означає, що при рівній частоті тактового генератора забезпечується продуктивність у 12 разів більше продуктивності попередніх мікроконтролерів на основі CISC-архітектури. З іншого боку, в рамках одного додатку із заданою швидкодією, AVR-мікроконтролер може тактуватись у 12 разів меншою тактовою частотою, забезпечуючи при цьому рівномірну швидкодію, але споживаючи набагато меншу потужність.

Таким чином, AVR-мікроконтролери надають більш широкі можливості по оптимізації продуктивності та енергоспоживання. Мікроконтролери забезпечують продуктивність до 16 млн. операцій за секунду і підтримують флеш-пам'ять програм різної ємності 1...256 кбайт [4].

Мікроконтролери серії AVR мають у своєму складі компаратори, АЦП, пристрої ШІМ, послідовні порти вводу/виводу. Звичайно, різні типономінали включають різні пристрої, але їх набір на сьогодні є нормою.

32 регістри загального призначення утворюють регістровий файл швидкого доступу, де кожен регістр безпосередньо пов'язаний з АЛП. За один такт з реєстрового файлу вибираються два операнди, виконується операція, і результат повертається у регістровий файл. АЛП підтримує арифметичні та логічні операції з регістрами, між регістром і константою або безпосередньо з регістром.

Базовий набір команд AVR містить 120 інструкцій (в залежності від моделі може становити від 90 до 133). Більшість команд займають тільки одну комірку пам'яті та виконуються за один такт. Керування периферійними пристроями здійснюється через адресний простір даних. Інструкції бітових операцій включають інструкції установки, очищення і тестування бітів.

Всі мікроконтролери AVR мають вбудовану FLASH ROM з можливістю внутрішньосхемного програмування через послідовний 4-провідний інтерфейс.

Периферія мікропроцесорів AVR включає: таймери-лічильники, широтно-імпульсні модулятори, підтримку зовнішніх переривань, аналогові компаратори, 10-розрядний 8-канальний АЦП, паралельні порти (від 3 до 48 ліній вводу/виводу), інтерфейси UART і SPI, сторожовий таймер і пристрій скидання по включенню живлення [5].

1.2 Засоби програмування мікроконтролерів сімейства AVR

Програмування мікроконтролерів є невід'ємною частиною розробки самостійного електронного пристрою. Технологія, що дозволяє програмувати електронні компоненти систем – внутрішньосхемне програмування.

Головною перевагою даного типу програмування є можливість об'єднання процесу програмування та тестування на виробництві, що дозволяє виключити окрему фазу програмування. Крім того, це дає можливість внесення оперативних змін без зупинки процесу виробництва, що суттєво знижує собівартість самого виробництва.

Мікросхеми з можливістю внутрішньосхемного програмування зазвичай мають вбудовану схему генерації напруги, необхідної для програмування та схему для комунікації з програматором через послідовний інтерфейс [6].

Програмування мікроконтролерів зазвичай відбувається наступним чином. У регістр програматору завантажується значення, яке необхідно розмістити за певною адресою, потім включається схема, яка пересилає вміст цього регістра по заданій адресі, проходить якийсь час очікування, поки завершиться процес програмування обраної комірки пам'яті і, нарешті, виконується перевірка, тобто перевіряється правильність записаного значення. Програмування всього пристрою може зайняти від декількох секунд до декількох хвилин в залежності від розміру пам'яті і алгоритму програмування.

Часто устаткування для програмування виявляється занадто дорогим, особливо для радіоаматорів і підприємств малого бізнесу. Але в деяких випадках для програмування потрібне просте і доступне обладнання. Наприклад, деякі моделі PIC і мікроконтролерів AVR програмуються дуже легко. Існують також пристрої, які не вимагають устаткування для програмування, наприклад, МК Basic STAMP, або мають вбудований апаратно-програмний блок, що рятує від необхідності використання зовнішнього обладнання, крім джерела підвищеної напруги для програмування, як МК 68HC05 [7].

Якщо МК допускає можливість внутрішньосистемного програмування (ВСП), In-System Programming (ISP), то це означає, що він може бути змонтований на плату з порожньою пам'яттю програм, яка потім може бути запрограмована без будь-якого впливу на інші компоненти схеми. Це може стати важливою обставиною при виборі МК. Використання ВСП позбавляє МК від необхідності купувати спеціальний програматор, дає можливість оновлювати програмне забезпечення без зміни розташованих на платі апаратних засобів і дозволяє виробникам створювати запас готових виробів, які можуть легко модифікуватися згідно з замовленнями [8].

Найбільш широкого розповсюдження досягли засоби розробки програм для AVR мікроконтролерів IAR, CodeVision AVR, AVR-GCC, AVR Studio. Порівняно новими вважаються графічні середовища Flowcode, GIDE. Середовища розробки на основі Basic, Pascal отримали обмежене застосування. Деяке число компіляторів реалізовано для програм на мові С, для яких є числені бібліотеки, які мають функції, необхідні для вирішення прикладних задач [10].

Для мікроконтролерної прикладної системи необхідний файл, що завантажується у flash-пам'ять, який зазвичай має розширення *hex. Такий файл і створюється у середовищі розробки. На рис. 13 наведено приклад прикладної системи з типовими елементами інтерфейсу оператора – LCD та пристроями вводу/виводу і їх взаємодія з середовищем розробки.

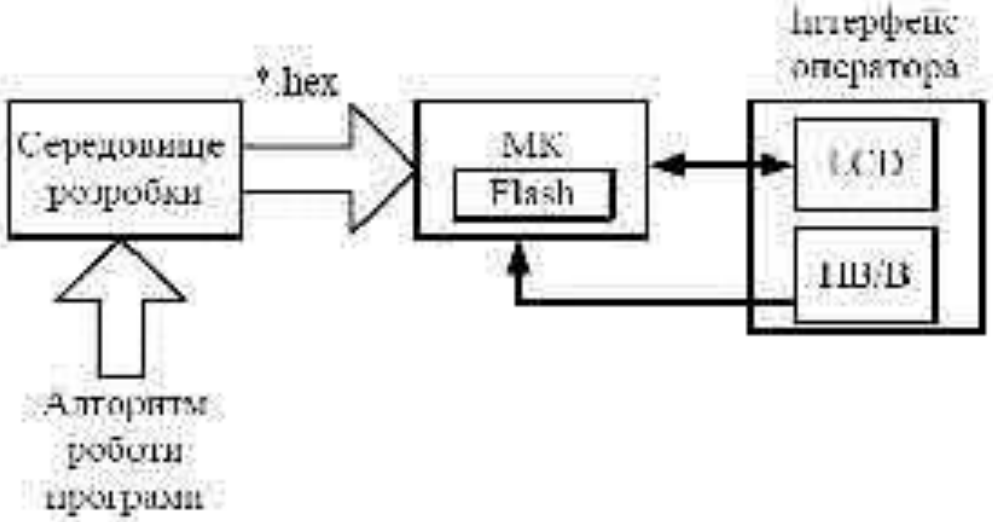

Рисунок 13. Взаємодія прикладної системи із середовищем розробки

Основним елементом середовища розробки (рис. 1.4) є компілятор, який визначає розмір коду та час виконання програми. Сучасні компілятори володіють широкими можливостями оптимізації програм з метою підвищення швидкості виконання програм або зменшення розмірів коду.

Рисунок 1.4. Елементи середовища розробки

Для кожного виду мікроконтролерів є вузькоспрямоване середовище програмування. Це пов'язано із внутрішньою структурою мікроконтролерів та технічною реалізацією запису програми до їхньої пам'яті [11].

Популярним та умовно безкоштовним середовищем програмування AVR-мікроконтролерів є CodeVisionAVR. Його особливість є об'єднання в собі С-

подібної моки програмування та Асемблеру. Функції програми дозволяють самостійно прошивати мікроконтролери. Кінцевим результатом розробки програмного забезпечення під мікроконтролер є створення *hex, *bin або *rom файлу для прошивки мікроконтролера за допомогою програматору [13]. Вигляд вікна робочої області середовища розробки Code VisionAVR зображено на рис. 1.5. Саме його буде використано у даному проекті.

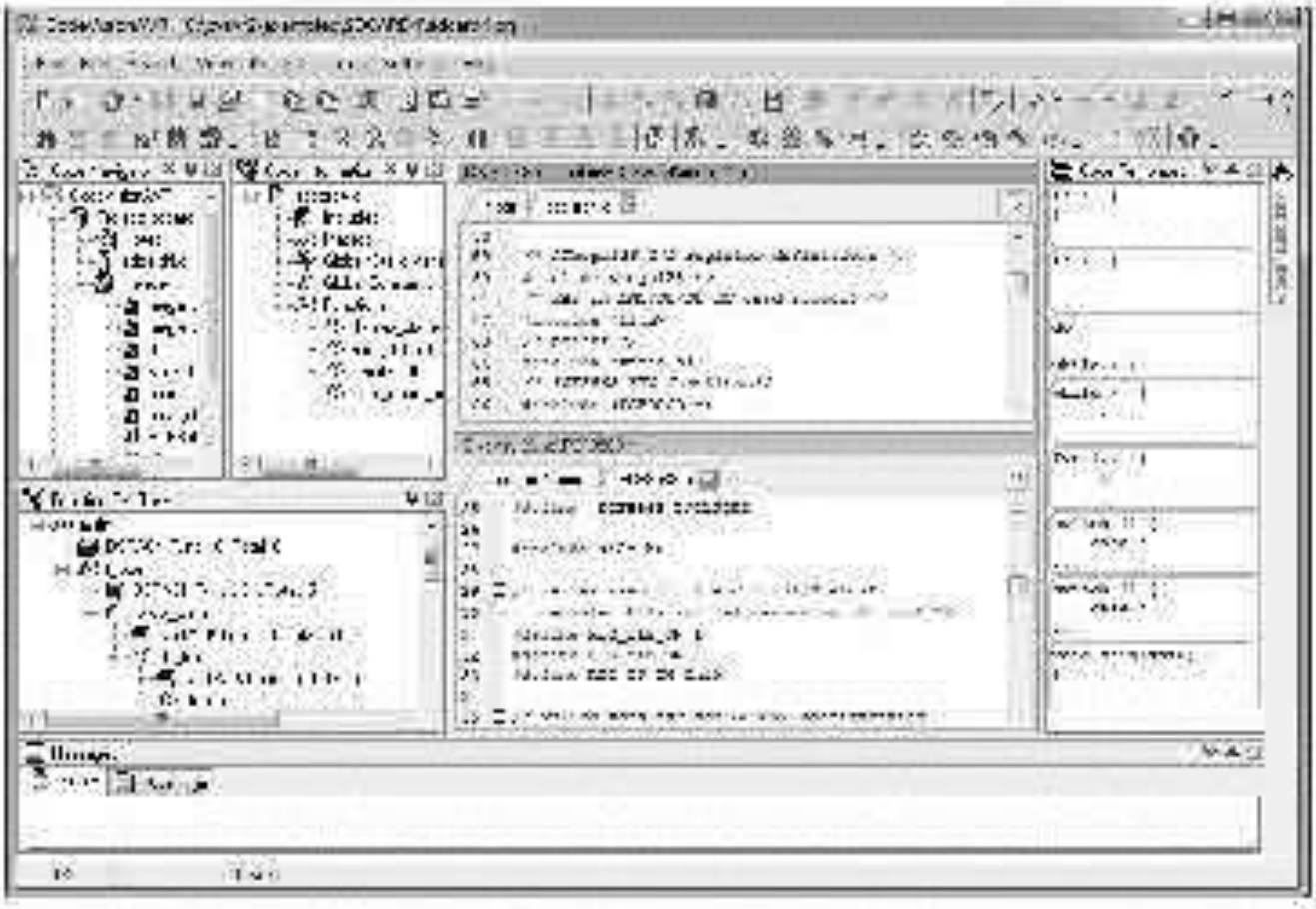

Рисунок 1.5. Середовище розробки CodeVisionAVR

1.3 Розробка структурної схеми програматор для мікроконтролерів ATMEL з ядром AVR

Створюваний програматор призначений для програмування мікроконтролерних пристроїв ATMEL з ядром AVR, які мають інтерфейс SPI. Цей програматор забезпечує перетворення інтерфейсу USB в SPI і навпаки. Структурна схема програматора представлена на рис. 1.6, де:

а) USB інтерфейс – роз'єм типу гаджет-USB;

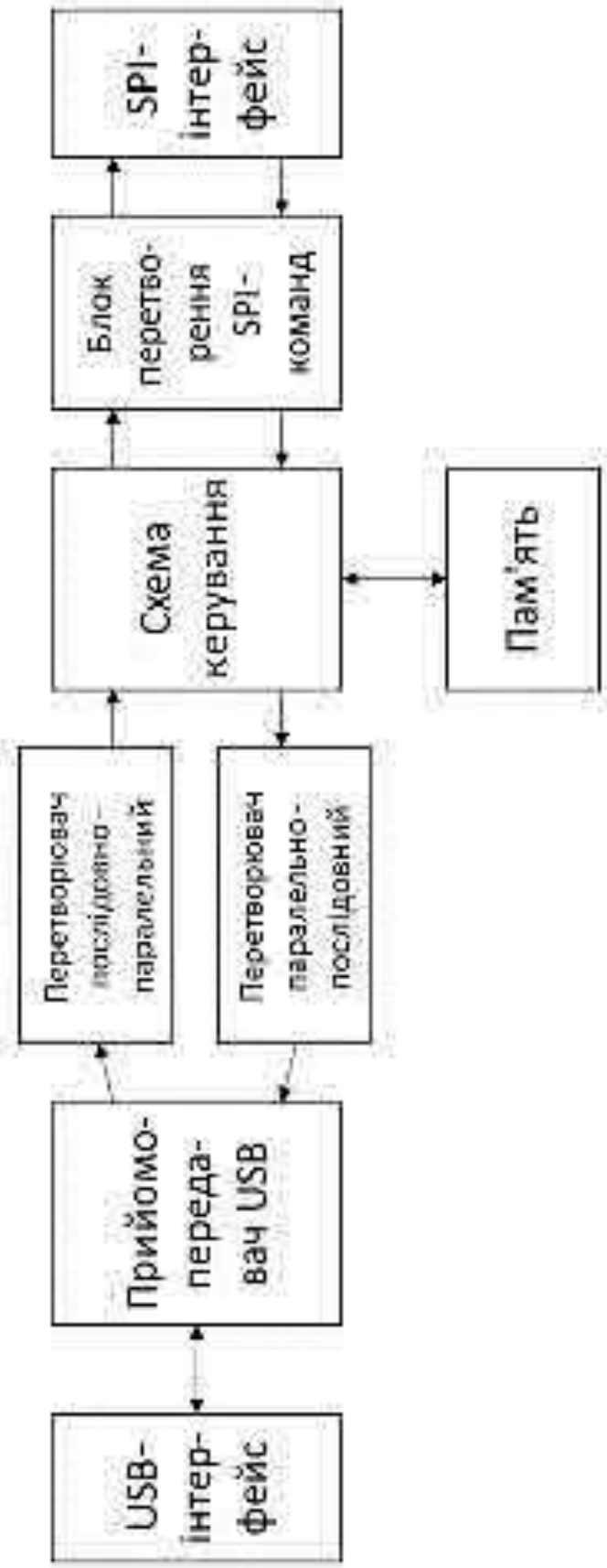

Рисунок 1.6. Структурна схема розроблюваного програматору для мікроконтролерів ATMEL з ядром AVR

б) Прийомо-передавач USB призначений для передачі сигналу фізичного рівня по двопровідних симетричних лініях;

в) Перетворювач послідовного коду в паралельний потрібний для ефективнішої підтримки протоколу USB;

г) Інтерфейс SPI є трипровідним інтерфейсом з лініями прийому даних, передачі даних і синхро-частоти;

д) Блок перетворення SPI-команд, призначений для організації пакетного режиму передачі інформації;

е) Схема керування і пам'ять забезпечує перетворення протоколів.

1.3.1 USB-інтерфейс

USB-інтерфейс підтримує наступні режими:

1) Підтримка full speed і low speed;

2) Підтримка режиму Ping-pong (подвійний банк даних);

3) 832 байти DPRAM:

а) 1 кінцева точка максимального розміру 64 байти (кінцева точка за умовчанням керувча);

б) 1 кінцева точка максимального розміру 256 байт (одинарний або подвійний банк даних);

в) 5 кінцевих точок максимального розміру 64 байти (одинарний або подвійний банк даних).

Складемо загальну блок-схему USB-контролера (рис. 1.7). USB-контролер забезпечує апаратний зв'язок між USB-шиною і даними в подвійній пам'яті порту (double port RAM, DPRAM).

Для роботи USB-контролера потрібна опорна частота 48 МГц±0,25 яка є виходом внутрішнього блоку фазового автоналаштування частоти (ФАНЧ). ФАНЧ генерує високу внутрішню частоту (48 МГц) для USB-інтерфейсу, використовуючи зовнішню низку частоту (генератор на кристалі або зовнішній генератор, підключений до входу XTAL1; тільки це джерело тактових сигналів

відповідає вимогам USB по точності частоти та джитера і забезпечує нормальну роботу USB-контролера).

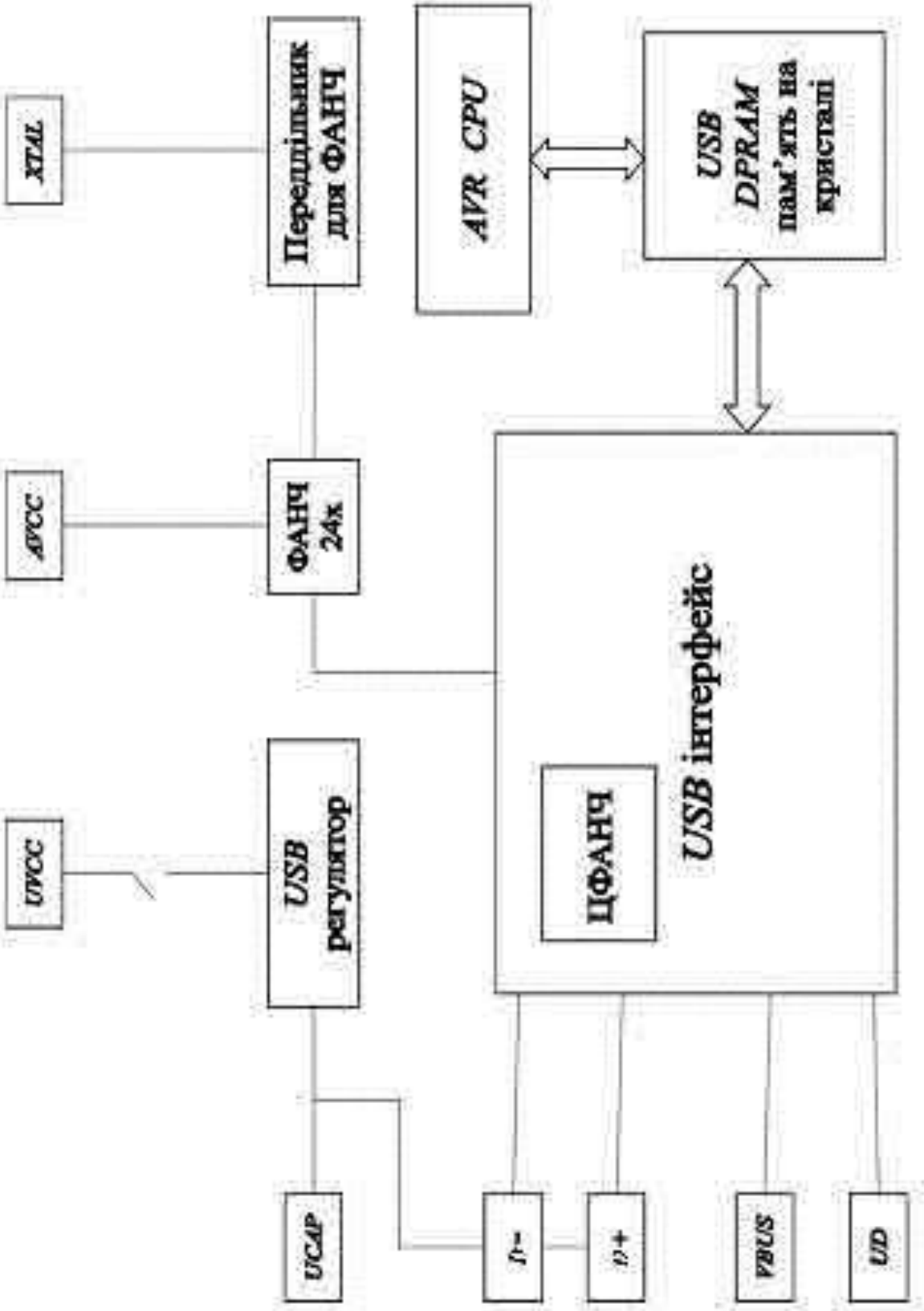

Рисунок 1.7. Загальна блок-схема USB-контролеру

Тактовий сигнал частотою 48 МГц використовується для виділення 12 МГц (для Full Speed) тактового сигналу синхронізації бітів з отриманих диференціальних даних (або 1,5 МГц для Low Speed) і для передачі даних відповідно до вибраної швидкості роботи USB-пристрою. Відновлення тактового сигналу відбувається за допомогою цифрової системи ФАНЧ (DPLL), яка відповідає специфікації на джитер для USB-шини. Згідно електричної специфікації USB, USB виводи (D+ або D-) мають бути підключені до напруги живлення в діапазоні від 3,0 до 3,6 В.

ATmega32U6, AT90USB64/128 може використати напругу живлення до 5,5В, внутрішній регулятор забезпечує живлення виводів USB.

Розглянемо режими роботи програмного забезпечення USB.

Залежно від режиму роботи USB програмне забезпечення повинне виконувати деякі з нижче приведених функцій:

Увімкнення живлення USB-інтерфейсу:

- включення регулятора живлення виводів USB;
- конфігурація інтерфейсу ФАНЧ;
- дозвіл ФАНЧ і очікування синхронізації ФАНЧ;
- дозвіл USB-інтерфейсу;
- конфігурація USB-інтерфейсу (швидкість, конфігурація кінцевих точок);
- очікування з'єднання на лінії VBUS;
- підключення USB-пристроїв.

Вимкнення живлення USB інтерфейсу:

- від'єднання USB-інтерфейсу;
- відключення USB-інтерфейсу;
- відключення ФАНЧ;
- відключення регулятора напруги виводів USB.

Призупинення USB-інтерфейсу:

- скидання біта призупинення (suspend);
- зупинка тактування USB;

- відключення ФАНЧ;

- контроль за наявністю дозволених переривань для виходу з режиму призупинення;

- введення процесора в sleep-режим.

Відновлення (resume) роботи USB-інтерфейсу:

- увімкнення ФАНЧ;

- синхронізація ФАНЧ;

- запуск тактування USB;

- очищення інформації про відновлення.

USB-контролер в режимі пристрою підтримує дві швидкості передачі даних: full speed і low speed. Додатково до обов'язкової контрольної кінцевої точки підтримується ще 6 інших кінцевих точок, які можуть бути конфігуровані як керуючі, суцільні (bulk), точки переривань або ізохронні:

- кінцева точка 0: програмований розмір FIFO до 64 байт, за умовчанням керуюча;

- кінцева точка 1: програмований розмір FIFO до 256 байт в режимі ping-pong;

- кінцеві точки з 2 до 6: програмований розмір FIFO до 64 байт, в режимі ping-pong.

Контролер розпочинає свою роботу з режиму "idle". У цьому режимі споживання виводів зведене до мінімуму. Нижче наведена схема показує стани контролера USB в режимі очікування при включенні (рис. 1.8). Стан скидання контролера пристрою полягає в наступному:

- тактування зупинене для зменшення загального споживання (встановлений біт FRZCLK);

- внутрішній стан контролера USB-пристрою "скидання" (значення усіх регістрів встановлено за умовчанням. Біт DETACH встановлений);

- банки цих кінцевих точок скинуті;

- підтяжки на лініях D+ або D- не активовані (режим від'єднання (Detach)).

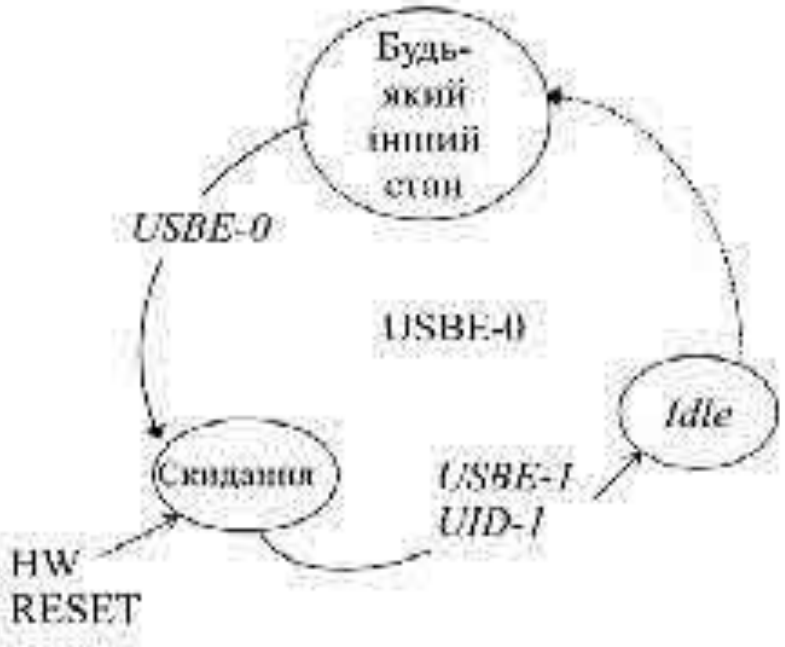

Рисунок 1.8. Стани контролера USB у режимі чікування після скидання

Підтяжки на лініях D+ або D- активуються при скиданні біта DETACH і наявності напруги VBUS. Загальний стан контролера після скидання – 'Idle' з мінімальним споживанням потужності, тому для входу в цей стан непотрібна активація ФАНЧ. Контролер USB-пристрою може бути у будь-який момент скинутий за допомогою скидання біта USBE (заборона USB-інтерфейсу).

Розглянемо скидання кінцевих точок.

Кінцева точка може бути скинута у будь-який момент за допомогою установки в регістрі UERST, що відповідає її біту (EPRSTx). Відбувається скидання:

- внутрішнього кінцевого автомата обраної кінцевої точки;
- банки даних Rx і Tx очищені, а їх внутрішні покажчики відновлені;
- значення в регістрах UEINTX, UESTA0X і UESTA1X відповідають значенню за умовчанням.

Поле "data toggle" залишається без змін, як і інші регістри. Конфігурація кінцевої точки залишається активною і кінцева точка все ще працює. Скидання кінцевої точки пов'язане із скиданням біта RSTDT (команда "data toggle"), що є відповіддю на команду USB CLEAR_FEATURE.

Розглянемо процес скидання USB.

Коли виявляється скидання на лінії USB, контролер здійснює наступні дії:

- усі кінцеві точки припиняють свою роботу;
- керуюча точка (за умовчанням) залишається конфігурованою.

Розглянемо вибір кінцевих точок.

Кінцева точка має бути вибрана перед будь-якою операцією, виконуваною ЦПП. Це здійснюється шляхом установки бітів EPNUM2: 0 (регістр UENUM) відповідно до номера кінцевої точки, з якою працюватиме ЦПП. Після цього ЦПП має доступ до усіх регістрів і даних кінцевої точки.

Розглянемо активацію кінцевої точки.

Кінцева точка залишається в стані скидання доки не буде встановлений біт EPEN. Для активації кінцевої точки має бути виконаний наступний порядок дій (рис. 1.9).

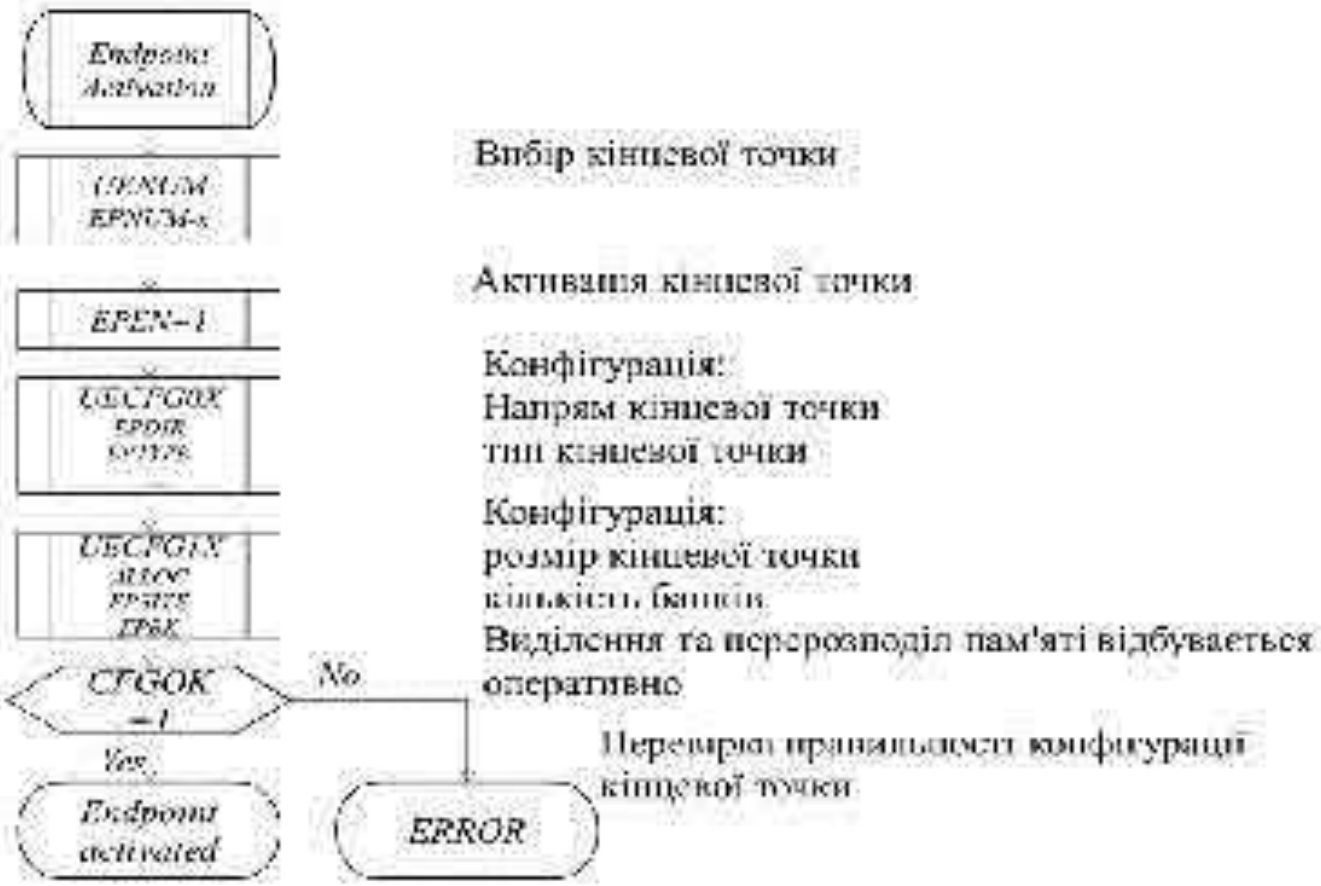

Рисунок 1.9. Порядок дій для активації кінцевої точки

Поки кінцева точка не конфігурована правильно (біт CFGOK скинутий) у відповідь на пакети, послані хостом, не приходитиме підтвердження (acknowledgement). Біт CFGOK не буде встановлений, якщо розмір кінцевої точки більший, ніж розмір DPRAM.

Скидання біта EPEN діє як скидання кінцевої точки. Це також призводить до наступного:

- конфігурація кінцевої точки зберігається (зберігаються EPSIZE, EPBK, ALLOC);

- скидання поля "data toggle ";

- область пам'яті DPRAM, виділена для кінцевої точки, залишається зарезервованою.

Розглянемо режими роботи USB хосту.

Цей режим можливий тільки для мікроконтролерів AT90USB647/1287.

Для USB-хост-контролера замість терміну "кінцева точка", який застосовується для контролера USB-пристрою, використовується термін "канал". Канал хоста відповідає кінцевій точки пристрою, як це описано в специфікації USB (рис. 1.10).

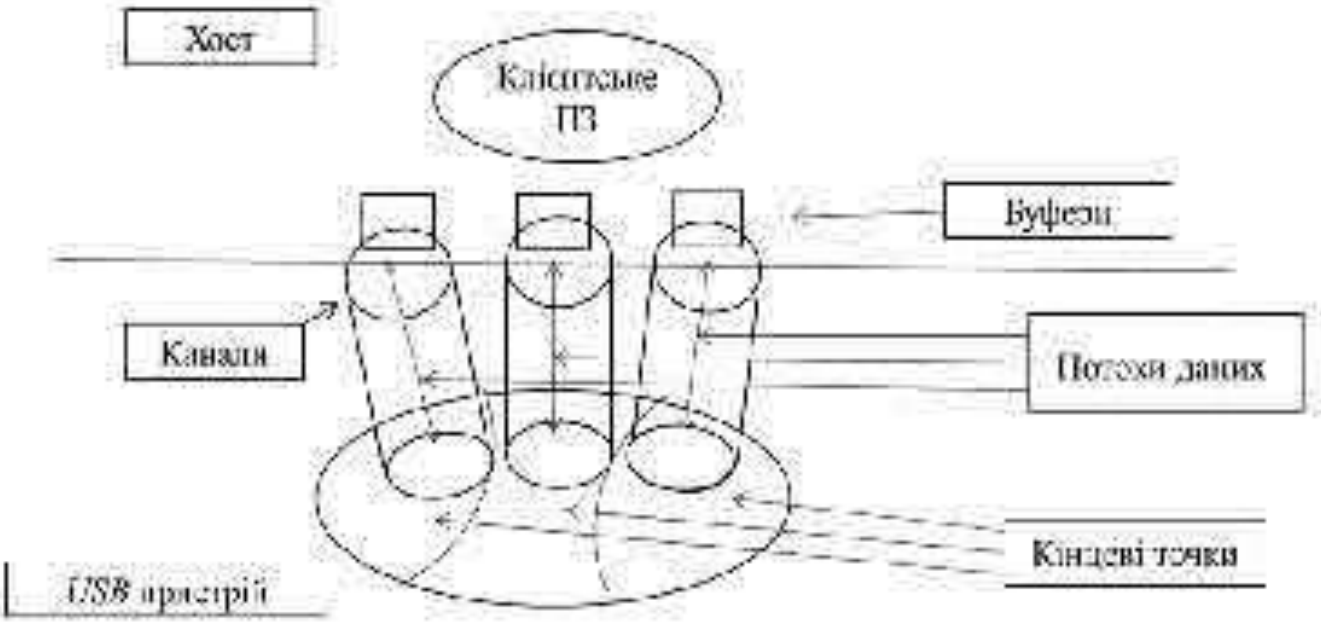

Рисунок 1.10. Канали та кінцеві точки в системі USB

У USB-хост-контролері канал асоціюється з кінцевою точкою пристрою відповідно до дескриптора конфігурації пристрою.

При від'єднанні значення біта DETACH після скидання дорівнює 1. Тому програмне забезпечення повинне скинути цей біт перед переміканням в режим хоста (установкою біта HOST).

Нижченаведена схема пояснює основні стани контролера USB-хоста при увімкненні (рис. 1.11).

Рисунок 1.11. Стан USB-хост-контролера після скидання

Стан USB-хост-контролера після скидання змінюється наступним чином. Коли USB-контролер дозволений і вибраний режим хоста, USB-контролер знаходиться у стані "idle". У цьому стані USB-хост-контролер чекає підключення пристрою, при цьому споживаючи мінімум потужності. Виводи USB мають бути у стані "idle". Для переходу в режим "host ready" немає необхідності активувати ФАНЧ. Хост-контролер переходить в призупинений (suspend) стан

коли USB-шина знаходиться в призупиненому стані, тобто коли хост-контролер не генерує початок фрейму (Start of Frame).

У цьому стані споживання USB-контролера мінімальне. Хост-контролер виходить з призупиненого стану, коли починає генерувати SOF на лінії USB.

Визначення пристрою виконується наступним чином. Пристрій визначається USB-контролером, коли стан USB-шини відрізняється від низьких рівнів на обох лініях D+ і D-. Іншими словами, коли USB-хост-контролер виявляє підтяжку (що відповідає пристрою) на лінії D+. Для дозволу цього виявлення хост-контролер повинен забезпечити живлення пристрою по лінії Vbus. Від'єднання пристрою виявляється хост-контролером за станом "idle", що відповідає низьким рівням на USB-лініях D+ і D-.

Вибір каналу виконується наступним чином. Перед будь-якою операцією, що проводиться ЦПП, має бути вибраний канал. Це здійснюється установкою бітів PNUM2: 0 (регістр UPNUM) відповідно до номера каналу, з яким працюватиме ЦПП. Після чого у ЦПП є доступ до регістрів каналу і даних.

Наступна послідовність дій має бути виконана для активації каналу (рис. 1.12).

Після того, як канал активізований (встановлений біт EPEN), контролер готовий до посилки запитів пристрою. Після конфігурації (CFGOK = 1), може змінюватися тільки маркер каналу (PTOKEN) і інтервал опитування для ізохронного каналу. Керуючий канал підтримує тільки 1 банк. Будь-яке інше значення приведе до помилки конфігурації (CFGOK = 0).

Скидання PEN призводить до скидання конфігурації каналу. Усі пов'язані з каналом регістри набувають значень за умовчанням. Програмне забезпечення повинне конфігурувати керуючий канал з наступними параметрами:

- тип: керуючий;
- маркер: SETUP;
- банк даних: 1;
- розмір: 64 байти.

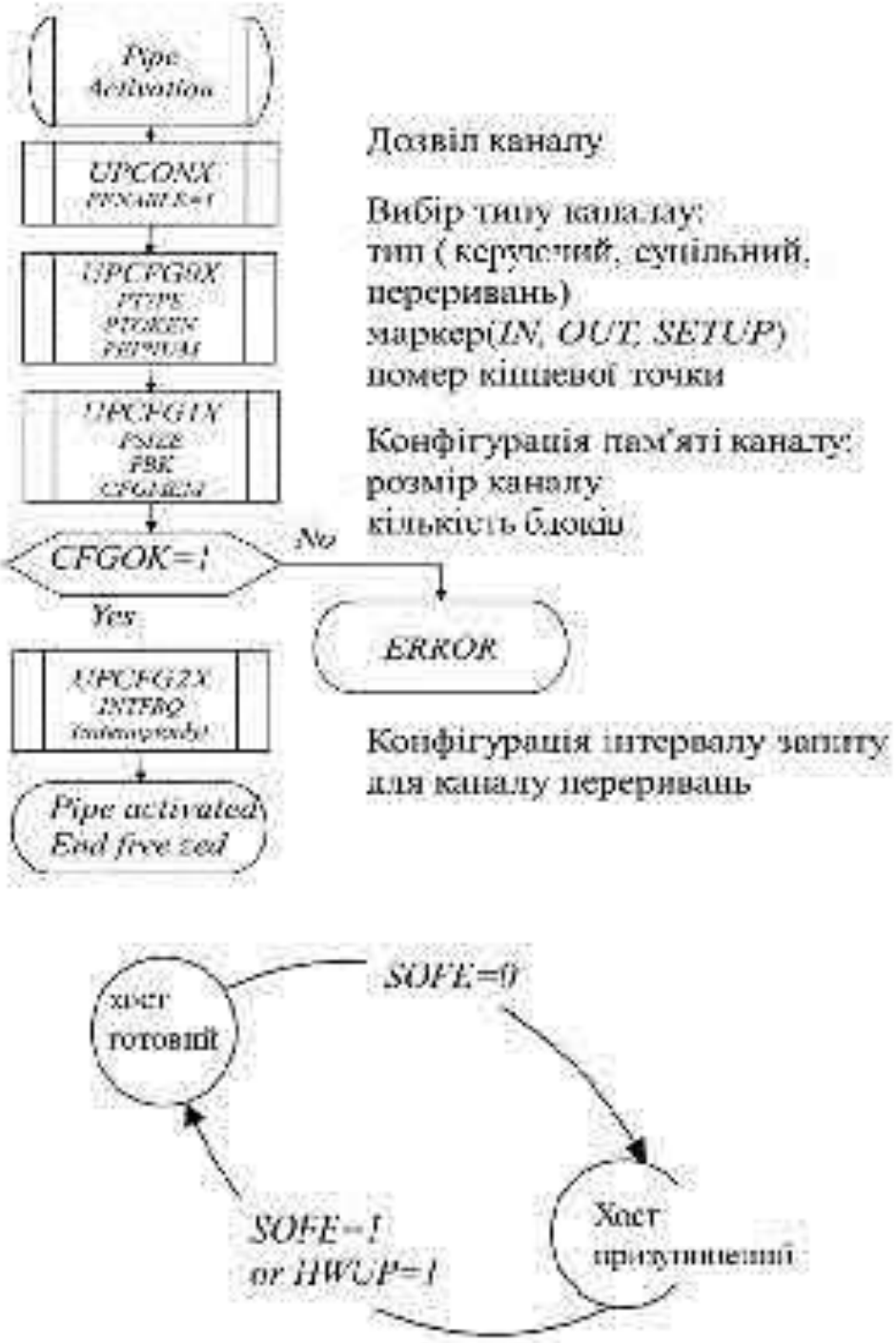

Рисунок 1.12. Послідовність дій для активації каналу

Програмне забезпечення запитує 8 байт дескриптора пристрою (Device Descriptor) за допомогою посилки запиту GET_DESCRIPTOR. Ці байти містять максимальний розмір пакету (MaxPacketSize) контрольної точки пристрою, а програмне забезпечення реконфігурує розмір керуючого каналу відповідно до отриманих даних.

USB-контролер посилає USB-скидання коли програмно встановлюється біт RESET. Біт RSTI встановлюється апаратно, коли запит на USB-скидання послано. Це призводить до переривання, якщо біт RSTE встановлений. Коли USB-скидання відправлене, конфігурація усіх каналів і розподіл пам'яті скидається. Регістр дозволу загальних USB-переривань залишається без змін.

Якщо перед цим лінія була в призупиненому стані (SOFEN=0), USB контролер автоматично перемикається в режим відновлення (resume) (встановлюється біт HWUPI) і апаратно встановлюється біт SOFEN для генерації відразу після USB скидання SOF. Після того, як USB-пристрій відповів на перший запит хосту з адресою за умовчанням (0), хост привласнює пристрою нову адресу. Хост-контролер повинен послати обладнанню USB-скидання і послати керуючий запит SET ADDRESS з новою адресою, яку повинен використати пристрій. Після закінчення цього керуючого запиту програмне забезпечення повинне записати нову адресу в регістр UHADDR. Усі подальші запити по усіх каналах здійснюватимуться з використанням цієї адреси..

Коли хост-контролер посилає USB-скидання, регістр UHADDR скидається апаратно і наступний запит хоста використовуватиме адресу за умовчанням (0).

При скиданні біта SOFEN хост-контролер переходить в призупинений режим. Жоден старт фрейма (Start Of Frame) більше не посилається в лінію USB і USB-пристрій переходить в призупинений режим на 3 мс пізніше. Пристрій поновлює роботу хоста посилкою запиту на віддалене пробудження (Upstream Resume). Хост-контролер виявляє не-idle стан на шині USB і встановлює біт HWUPI. Якщо не-idle стан відповідає віддаленому пробудженню (стан K), то апаратно встановлюється біт RXRSMI.

1.3.2 Послідовний периферійний інтерфейс – SPI

Інтерфейс SPI дозволяє організувати послідовну синхронну високошвидкісну передачу даних між ATmega128 і іншим периферійним пристроєм або між декількома AVR-мікроконтролерами.

Відмінні особливості інтерфейсу SPI в ATmega128:

- повнодуплексна, трипровідна синхронна передача даних;
- провідна або підпорядкована робота;
- передача першим молодшого або старшого біта;
- сім програмованих швидкостей зв'язку;
- прапор переривання для індикації закінчення передачі даних;
- захисний прапор при поверховому записі;
- пробудження з режиму холостого ходу (Idle);
- режим ведучого (майстра) SPI з подвоєнням швидкості (CK/2).

Зовнішні з'єднання між ведучим (майстром) і підлеглим ЦПП через інтерфейс SPI показані на рис. 1.13. Система складається з двох регістрів зсуву і генератора провідної синхронізації. Ведучий SPI ініціює сеанс зв'язку поданням низького рівня на вхід SS того підпорядкованого пристрою, з яким необхідно обмінюватися даними. Обидва респонденти (ведучий і підлеглий) готують дані до передачі у своєму регістрі зсуву, при цьому на стороні ведучого генеруються також імпульси синхронізації на лінії SCK. По лінії MOSI завжди здійснюється передача даних від ведучого до підлеглого, а по MISO, навпаки, від підлеглого до майстра. Після закінчення передачі кожного пакету даних ведучий SPI повинен синхронізувати підлеглого шляхом подання високого рівня на лінію SS (вибір підпорядкованого інтерфейсу). Якщо SPI налагоджений як ведучий (майстер), то керування лінією SS відбувається не автоматично. Ця операція має бути виконана програмно перед початком сеансу зв'язку. Після цього, запис в регістр даних SPI ініціює генерацію синхронізації і апаратний зсув 8-ми розрядів в підпорядкований пристрій. Після закінчення зсуву одного байту генератор синхронізації SPI зупиняється, при цьому встановлюючи прапор закінчення передачі (SPIF). Якщо встановлений біт SPIE в регістрі SPCR, то дозволяється переривання SPI і після закінчення передачі байту генеруватиметься запит на переривання. Майстер може продовжити зсувати наступний байт, якщо записати його в регістр SPDR, або подати сигнал закінчення пакету шляхом установки низького рівня на лінії SS.

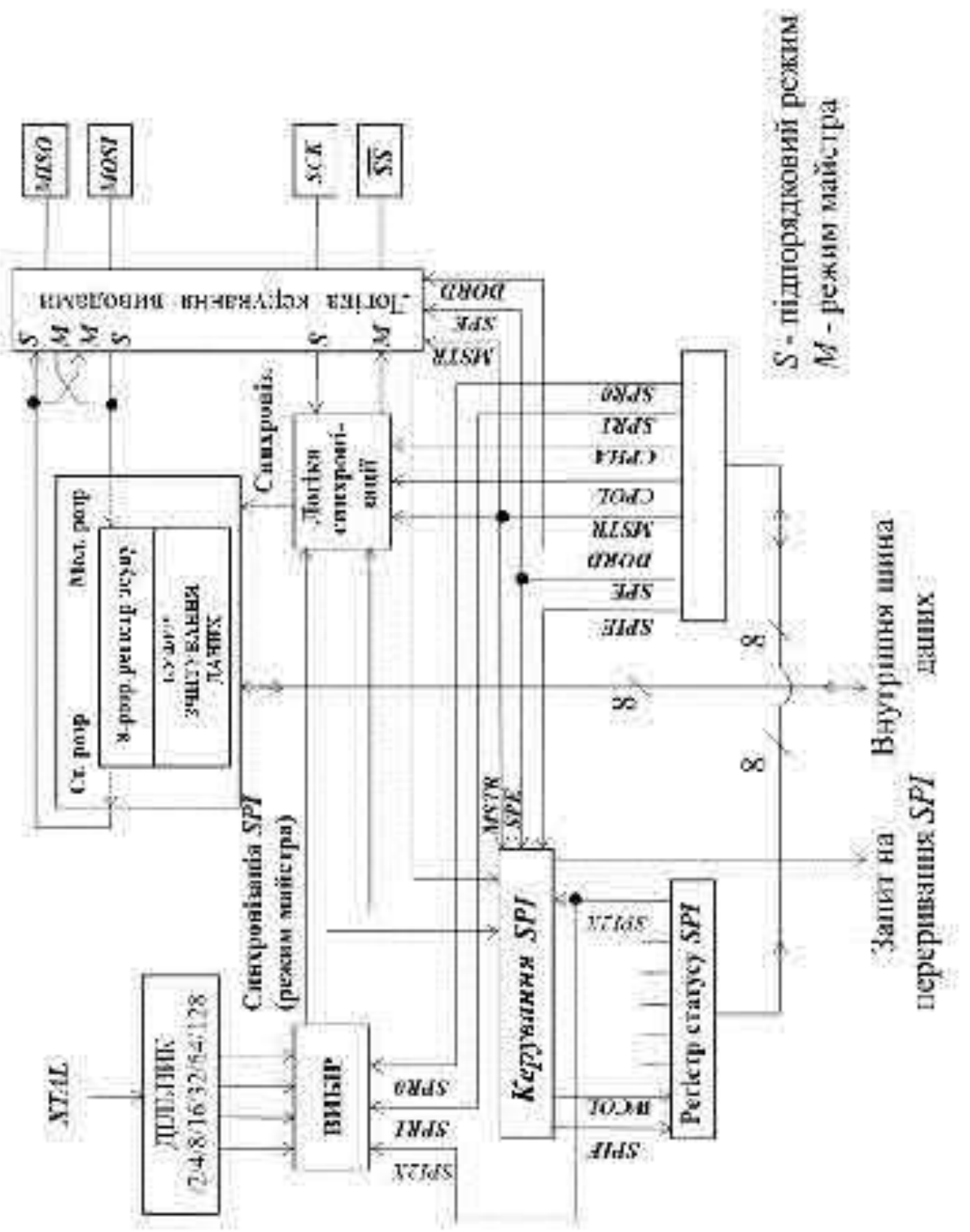

Рисунок 1.13. Функціональна схема інтерфейсу SPI

У режимі підлеглого, інтерфейс SPI знаходиться в стані очікування, в якому MISO переводиться в третій стан, до тих пір, поки на виводі SS є

присутнім високий рівень. У цьому стані програма може оновлювати вміст регістра даних SPI (SPDR), але імпульси синхронізації, що при цьому входять, не зсувають дані до подання низького рівня на вивід SS. Після того, як один байт був повністю зсунутий, встановлюється прапор закінчення передачі SPIF. Якщо встановлений біт дозволить переривання SPI (SPIE) в регістрі SPCR, то установка прапора SPIF призводить до генерації запиту на переривання. Підлеглий може продовжувати розміщувати нові дані для передачі в регістр SPDR перед читанням даних, що входять. Останній прийнятий байт зберігається у буферному регістрі (рис. 1.14).

У напрямку передачі даних система виконана як однобуферна, а у напрямку прийому використовується подвійна буферизація. Це означає, що передані байти не можуть бути записані в регістр даних SPI, перш ніж повністю завершиться цикл зсуву. Під час прийому даних необхідно стежити, щоб прийнята посилка була зчитана з регістра даних SPI, перш ніж завершиться цикл зсуву нової посилки, що надійшла. Інакше перший байт буде втрачений.

У підлеглому режимі SPI керуюча логіка здійснює вибірку сигналу SCK, що надійшов. Щоб гарантувати коректність вибірки тактового сигналу необхідно використати частоту синхронізації SPI не більше fosc/4.

Якщо робота SPI дозволена, то дозволяється альтернативний напрям виводів MOSI, MISO, SCK і SS (таблиця. 1.1).

Таблиця 1.1. Напрям виводів SPI

Вивід	*Напрям для ведучого SPI*	*Напрям для підлеглого SPI*
MOSI	Визначається користувачем	Вхід
MISO	Вхід	Визначається користувачем
SCK	Визначається користувачем	Вхід
SS	Визначається користувачем	Вхід

Флеш-пам'ять і ЕСППЗП можуть бути запрограмовані через послідовний інтерфейс SPI, коли вхід RESET переведений в низький стан. Послідовний інтерфейс складається з наступних сигналів: SCK, MOSI (вхід) і MISO (вихід).

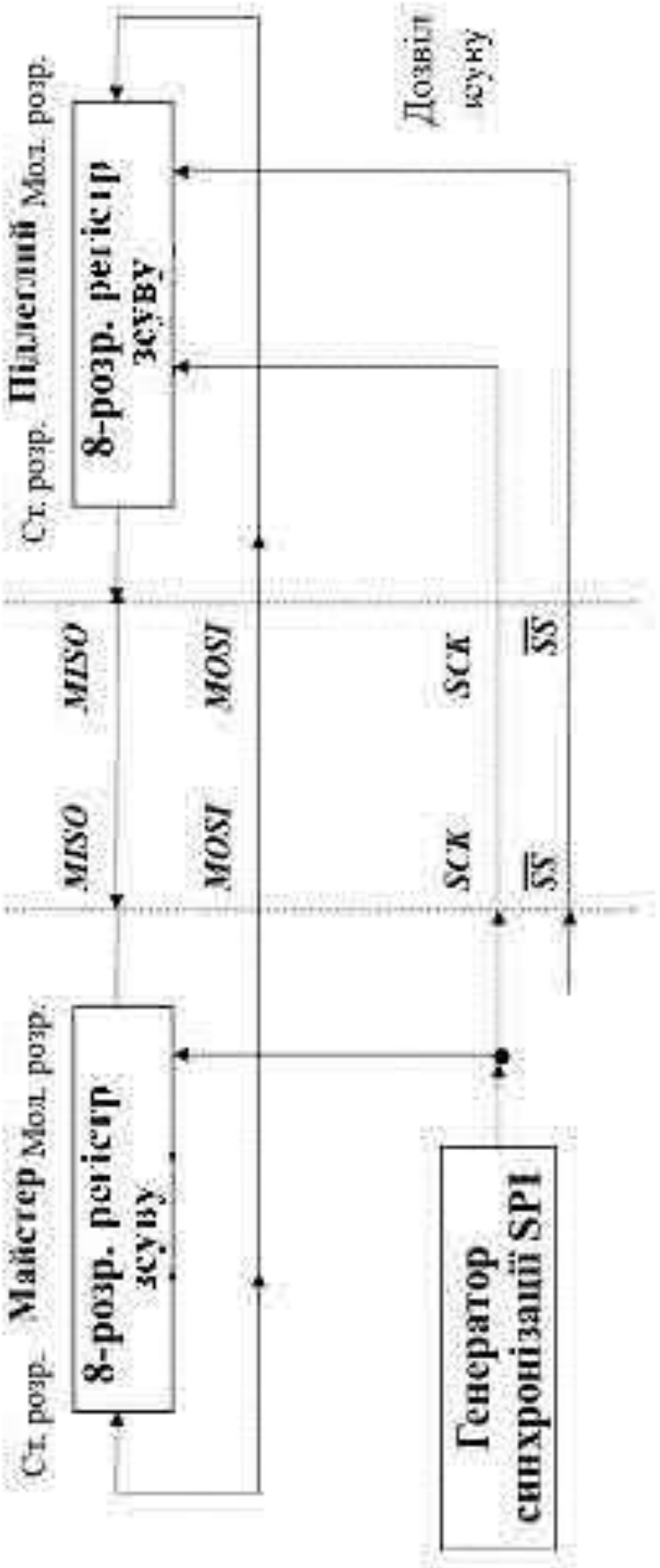

Рисунок 1.14. Зовнішнє з'єднання ведучого (майстра) і підлеглого SPI

Після подання низького рівня на вхід RESET необхідно виконати інструкцію дозволу програмування. У таблиці 1.2 представлено опис сигналів програмування. Не усі виводи послідовного програмування співпадають з виводами внутрішнього інтерфейсу SPI.

Таблиця 1.2. Виводи інтерфейсу SPI при послідовному програмуванні

Позначення	*Вивід*	*Напрям*	*Опис*
MOSI (PDI)	*PE0*	Введення	Послідовне введення даних
MISO (PDO)	*PE1*	Виведення	Послідовне виведення даних
SCK	*PB1*	Введення	Синхронізація послідовного зв'язку

Також слід зазначити, що всюди при описі послідовного програмування використовуються найменування MOSI і MISO для опису послідовного введення і виведення даних, відповідно. Для ATmega128 відповідні виводи програмування іменуються PDI і PDO.

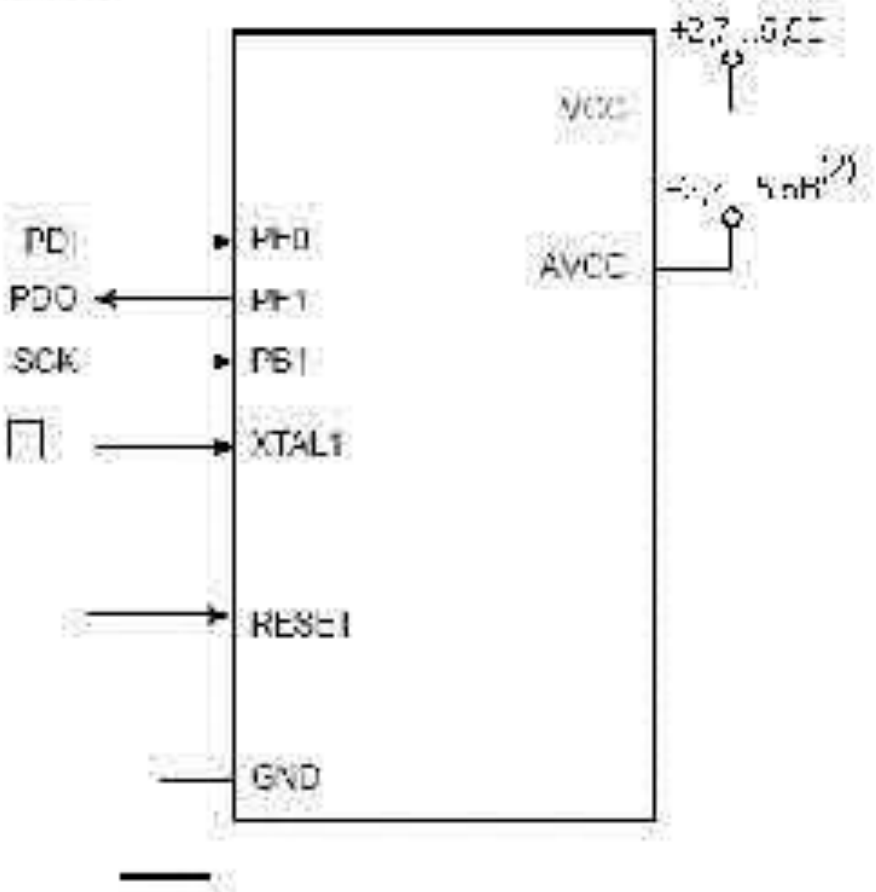

Рисунок 1.15. Послідовне програмування та перевірка

По причте, що при послідовному програмуванні використовується той же модуль SPI, що і при звичайній роботі мікроконтролера, є одна важлива відмінність: виводи MOSI/MISO модуля введення-виведення SPI, які поєднані з PB2 і PB3, не використовуються при програмуванні. Замість них використовуються PE0 і PE1 для введення і виведення даних при послідовному програмуванні (рис. 1.15).

1. Якщо мікроконтролер тактується внутрішнім генератором, то немає необхідності підключати тактове джерело до виводу XTAL1.

2. VCC - 0.3В < AVCC < VCC 0.3В, але AVCC повинен знаходитись в межах 2,7 – 5,5В.

Розглянемо алгоритм послідовного програмування через SPI.

Під час послідовного запису в ATmega128 дані тактуються наростаючим фронтом SCK. Під час читання даних з ATmega128 дані тактуються фронтом SCK, що падає.

Для програмування і перевірки пам'яті ATmega128 в режимі послідовного програмування через SPI рекомендується дотримуватися наступної послідовності (чотирьохбайтовий формат наведено в таблиці 1.3):

1. Послідовність подання живлення: подати напругу живлення між VCC і GND, коли на входах RESET і SCK є присутнім рівень 0. У деяких системах, програматор не може гарантувати, що SCK = 0 при поданні живлення. В цьому випадку необхідно сформувати позитивний імпульс на RESET тривалістю не менше двох тактів ЦПП після того, як SCK прийняв низький стан. Альтернативно сигналу RESET можна використати вивід PEN. В цьому випадку буде важливо тільки значення PEN під час скидання при подачі живлення. Якщо програматор не гарантує, що при подачі живлення SCK=0, то використання PEN неприпустиме. При використанні цього методу мікроконтролер може повернутися до нормального режиму роботи тільки зняттям і відновленням живлення.

Таблиця 1.3. Набір інструкцій послідовного програмування через SPI

Інструкція	Формат інструкції				Функція
	Байт 1	Байт 2	Байт 3	Байт 4	
Дозвіл програмування	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Дозвіл послідовного програмування після подання рівня 0 на *RESET*
Стирання кристала	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирання ЕСППЗП і флеш-пам'яті
Читання пам'яті програм	0010 H'00'	aaaa aaaa	bbbb bbbb	oooo oooo	Читання старшого (H'1') або молодшого (H'0') байта даних з пам'яті програм за адресою a : b
Завантаження сторінки пам'яті програм	0100 H'00'	xxxx xxxx	xbbb bbbb	iiii iiii	Запис старшого (H'1') або молодшого (H'0') байта даних в сторінку пам'яті програм за адресою b. Мол. байт даних має бути завантажений перед старшим байтом за тією ж адресою
Запис сторінки пам'яті програм	0100 1100	aaaa aaaa	bxxx xxxx	xxxx xxxx	Запис сторінки пам'яті програм за адресою a : b
Читання ЕСППЗП	1010 0000	xxxx aaaa	bbbb bbbb	oooo oooo	Читання даних з ЕСППЗП за адресою a : b
Запис ЕСППЗП	1100 0000	xxxx aaaa	bbbb bbbb	iiii iiii	Запис даних в ЕСППЗП за адресою a : b
Читання біт-захисту	0101 1000	0000 0000	xxxx xxxx	xxoo oooo	Читання біту захисту. "0" - запрограм., " 1" - не запрограм.
Запис біт-захисту	1010 1100	111x xxxx	xxxx xxxx	1Iii iiii	Запис біту захисту. Запис " 0" призводить до програмування біта захисту
Читання сигнатурного байту	0011 0000	xxxx xxxx	xxxx xxbb	oooo oooo	Читання сигнатурного байту за адресою b
Запис конфігураційних біт	1010 1100	1010 0000	xxxx xxxx	iiii iiii	"0" для програмування, "1" для стирання
Запис старших конфігураційних біт	1010 1100	1010 1000	xxxx xxxx	iiii iiii	"0" для програмування, "1" для стирання
Запис розширених конфігураційних біт	1010 1100	1010 0100	xxxx xxxx	xxii iiii	"0" для програмування, "1" для стирання
Читання конфігураційних біт	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Читання конфігураційних біт. "0" - запрограм., " 1" - не запрограм.
Читання старших конфігураційних біт	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Читання старших конфігураційних біт. "0" - запрограм., " 1" - не запрограм.
Читання розширених конфігураційних біт	0101 0000	0000 1000	xxxx xxxx	oooo oooo	Читання розширених конфігураційних біт. "0" - запрограм., " 1" - не запрограм.
Читання калібрувального байту	0011 1000	xxxx xxxx	0000 00bb	oooo oooo	Читання калібрувального байта за адресою b

a - адреса старших розрядів;

b - адреса молодших розрядів;

H – (0) мол. байт, (1) ст. байт;

о – виведення даних;

і – введення даних;

х – довільне значення.

2. Пауза не менше 20 мс і дозвіл послідовного програмування шляхом запису команди дозволу послідовного програмування через вхід MOSI.

3. Інструкції послідовного програмування не виконуються, якщо послідовний зв'язок не увійшов до синхронізації. Вхід в синхронізацію відображає набуття значення другого байта ($53) при записі третього байта інструкції дозволу послідовного програмування. Залежно від того, коректні або ні набуті значення, передаються усі чотири байти інструкції. Якщо набуте значення не рівне $53, то формується позитивний імпульс на вході RESET і вводиться нова команда дозволу послідовного програмування.

4. Флеш-пам'ять програмується посторінково. Розмір сторінки показаний в таблиці. Сторінка пам'яті завантажується побайтно, при цьому в інструкції "завантаження сторінки пам'яті програм" вказуються дана адреса в семи молодших розрядах. Щоб гарантувати коректність завантаження сторінки спочатку необхідно записати молодший байт, а потім старший байт даних за кожною адресою. Запис сторінки пам'яті програм ініціюється введенням інструкції "запис сторінки пам'яті програм", де в 9-ти старших розрядах вказується адреса сторінки.

Якщо опитування не використовується, то програміст повинен передбачити затримку не менше tWD_FLASH перед введенням нової сторінки (табл. 14).

5. Масив пам'яті ЕСППЗП програмується побайтно, при цьому в інструкції запису вказується адреса і дані. Перед записом даних спочатку автоматично стирається комірка ЕСППЗП, що адресується. Якщо опитування не використовується, то програміст повинен передбачити затримку tWD_EEPROM перед введенням наступного байту (табл. 14). Після стирання пам'яті мікроконтролера необхідно записувати тільки дані, нерівні $FF.

Таблиця 1.4. Мінімальна тривалість затримок перед записом чергової комірки флеш-пам'яті і ЕСППЗП

Позначення	*Мінімальна затримка*
tWD_FLASH	4.5 мс
tWD_EEPROM	9.0 мс
tWD_ERASE	9.0 мс

6. Будь-який елемент пам'яті можна перевірити використанням інструкції читання, яка повертає вміст комірки за вказаною адресою шляхом послідовної передачі на виході MISO.

7. Після закінчення програмування вхід RESET має бути переведений у високий стан для відновлення нормальної роботи.

8. Послідовність зняття живлення (при необхідності): установка RESET="1", відключити живлення VCC.

Після того, як сторінка повністю запрограмована у флеш-пам'ять, при читанні по адресах в межах запрограмованої сторінки повертається $FF. Мікроконтролер готовий до запису нової сторінки, якщо запрограмоване значення зчитане коректно. Це використовується для визначення моменту, коли може бути завантажена наступна сторінка. При цьому виконується запис усієї сторінки одночасно і будь-яка адреса в межах сторінки може використовуватися для опитування. Опитування даних флеш-пам'яті не діє для значення $FF, оскільки при записі цього значення користувач може не вводити затримку tWD_FLASH перед програмуванням нової сторінки. Ця можливість пояснюється тим, що очищена пам'ять мікроконтролера містить $FF в усіх комірках. Значення tWD_FLASH представлене в таблиці 1.4.

При читанні значення за адресою, яка використовувалася для запису нового байту і подальшого його програмування в ЕСППЗП, повертається значення $FF. В цей же час, мікроконтролер готовий до запису нового байту, якщо запрограмоване значення коректно зчитується. Це використовується для визначення моменту, коли може бути здійснений запис наступного байту.

Наведене вище не поширюється на значення $FF, але програміст повинен звернути увагу на наступне: оскільки очищена пам'ять заповнена $FF по усіх адресах, то програмування комірки значенням $FF може бути пропущене. Пропуск можна робити, якщо ЕСППЗП перепрограмується без попереднього стирання усієї пам'яті. В цьому випадку, значення $FF не можна використати для опитування даних і програміст повинен передбачити затримку неменше tWD_EEPROM перед програмуванням наступного байту.

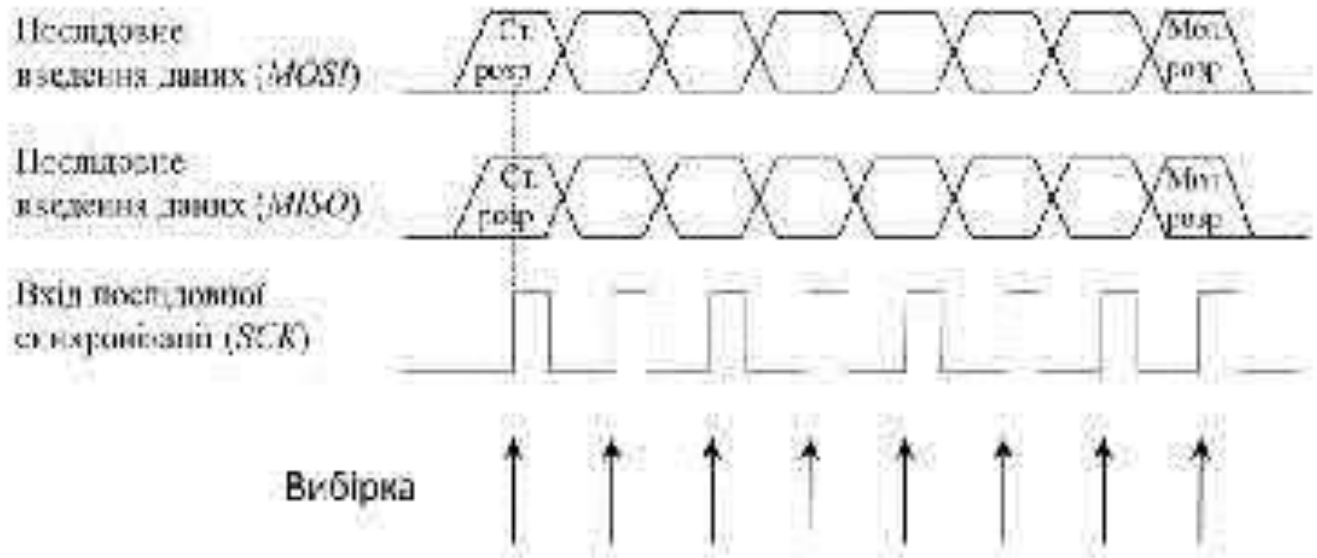

Рисунок 1.16. Осцилограми сигналів послідовного програмування інтерфейсу SPI

1.4 Розробка принципової електричної схеми програматору для мікроконтролерів ATMEL з ядром AVR

Схема програматору наведена на рис.1.17. Запобіжник F1 служить для захисту ліній живлення порту USB від випадкового замикання по ланцюгах живлення програматора. Діоди VD1, VD2 – звичайні випрямні, з прямим падінням напруги ~0,6-0,7 В, призначені для зниження напруги живлення мікроконтролера DD1 до 3,6 В. Світодіоди VL1, VL2 сигналізують про поточні дії програматора, і, відповідно, означають режими читання і запису. Світодіод VL3 служить для сигналізації подачі живлення на програматор. Джампер J1-J2 служить як для початкового програмування (замкнути J1 – MODify), так і для використання у якості роз'єму програматора (замкнути J2 – NORMal).

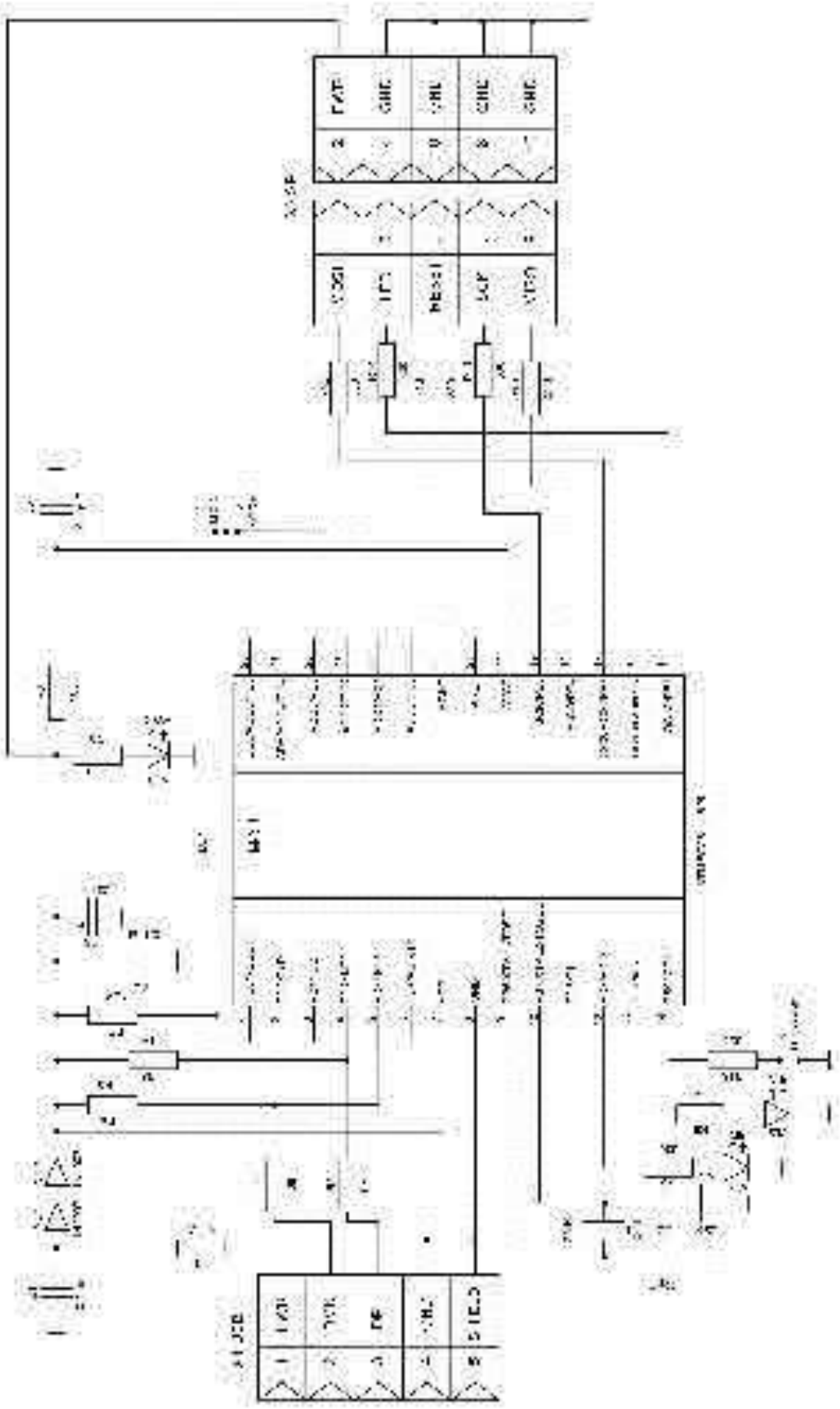

Рисунок 1.17. Схема електрична принципова ISP-програматору

Резистори R10-R14 призначені для узгодження рівнів сигналів контролера програматора і програмованого контролера.

За допомогою J3 LOW SCK можливо знижувати тактову частоту порту SPI МК програматора до ~20 кГц. При розімкненому джампері частота SPI нормальна, при замкнутому – знижена. Перемикати джампер можна "на ходу", оскільки керуюча програма МК програматора перевіряє стан лінії PB0 при кожному зверненні до порту SPI. Не рекомендується перемикати джампер при запущеному процесі запису/читання програмованого МК, оскільки, швидше за все, це приведе до спотворення операції запису/читання. Цей джампер введений для можливості програмування МК AVR, тактованих від внутрішнього генератора 128 кГц.

Швидкість роботи порту SPI МК програматора при розімкненому джампері J3 дорівнює 187,5 кГц. Це дозволяє програмувати контролери з тактовою частотою приблизно від 570 кГц для tiny/mega, 750 кГц для 90S і 7,5 МГц для 89S. Контролери програмуються від 10 до 30 секунд разом з верифікацією залежно від об'єму FLASH-пам'яті і тактової частоти. На вихід LED роз'єму ISP виведений мелдр з частотою 1 МГц для "пожвавлення" МК, у яких були помилково запрограмовані шпюки, що відповідають за тактування. Сигнал генерується постійно і не залежить від режиму роботи програматора.

У цій модифікації програматора встановлена додаткова знімна перемичка (джампер), яка дозволяє подавати живлення 5 В на програмовану схему від USB порту комп'ютера через контакт 2 роз'єма ISP (на схемі не показано). Цей джампер розташований з краю плати.

1.4.1 Загальний опис схеми програматору

ISP-програматор призначений для внутрішньо-схемного програмування контролерів серії AVR фірми ATMEL. ISP (In System Programming) – "Програмування в системі". Програматор розроблений на основі оригінального програматора AVR910 і сумісний з ним по можливостях і системі команд.

Відмінністю цього програматора від оригінального AVR910 є робота по шині USB. При цьому організовується віртуальний COM-порт.

Користувач ISP-програматору повинен мати уявлення про методи програмування AVR-контролерів, а також самостійно вивчити керуючі програми, які є продуктами фірми ATMEL або сторонніх фірм, а також технічну документацію на програмовані контролери.

Підключення програматора до комп'ютера здійснюється до USB-порту за допомогою стандартного шнура USB,що додається(A – USB – B). Живлення 5В на програматор подається від USB-шини комп'ютера. Наявність живлення відображає світлодіод зеленого кольору.

Програматор з'єднується з схемою програмованого контролера за допомогою плоского кабелю. Контакти роз'єму ISP: 1 – MOSI; 3 – LED; 5 – RESET; 7 – SCK; 9 – MISO; 2 – 5V (через Jumper); контакти 4,6,8,10 сполучені із загальним дротом (GND); перший провідник в кабелі відмічений червоним кольором.

Програматор дозволяє програмувати контролери AVR, які підтримують ISP(In System Programming),а також МК серії 89S – 89S53 і 89S8252.

Кожна конкретна програма підтримує свій список мікроконтролерів, який може оновлюватися в наступних версіях. За допомогою ISP-програматору і програми AVR Prog з пакету AVR Studio можливе програмування наступних мікроконтролерів фірми ATMEL: ATtiny10, ATtiny12, ATtiny15(L), ATtiny26(L), AT89S53, AT89S8252, AT90S1200, AT90S2313, AT90S2323, AT90LS2323, AT90S2333, AT90LS2333,AT90S2343, AT90LS2343, AT90S4414, AT90S4433, AT90LS4433, AT90S4434,AT90LS4434, AT90S8515, AT90S8535, AT90LS8535, ATmega8(L), ATmega16(L), ATmega32(L), ATmega64(L), ATmega103(L), ATmega128(L), ATmega161(L), ATmega163(L), ATmega169(L, V), ATmega8515(L), ATmega8535(L).

Програми Chipvision і ChipBlaster мають ширший список підтримуваних мікроконтролерів.

1.4.2 Встановлення драйверів і програм

Для ОС сімейства Windows необхідно виконати наступні дії.

1) Підключити програматор до PC через вільний роз'єм USB. ОС знайде новий пристрій – AVR910 USB Programmer.
2) Після пропозиції автоматично знайти драйвер, відмовитися, і вказати шлях до файлу prottoss.avr910 usb.inf (вказати шлях до диска).
3) При попередженні, що драйвер не має цифрового підпису, ігнорувати. Номер створеного віртуального COM-порту має бути в діапазоні COM1 – COM4, інакше, програма AVRProg не зможе знайти програматор. Якщо Windows присвоїть інший номер порту, то змінити його можна, якщо зайти в диспетчер пристроїв: AVR910 USB Programmer > Властивості > Параметри порту > Додатково > Номер COM-порту.

Нижче показано встановлення номера порту в диспетчері пристроїв (рис. 1.18).

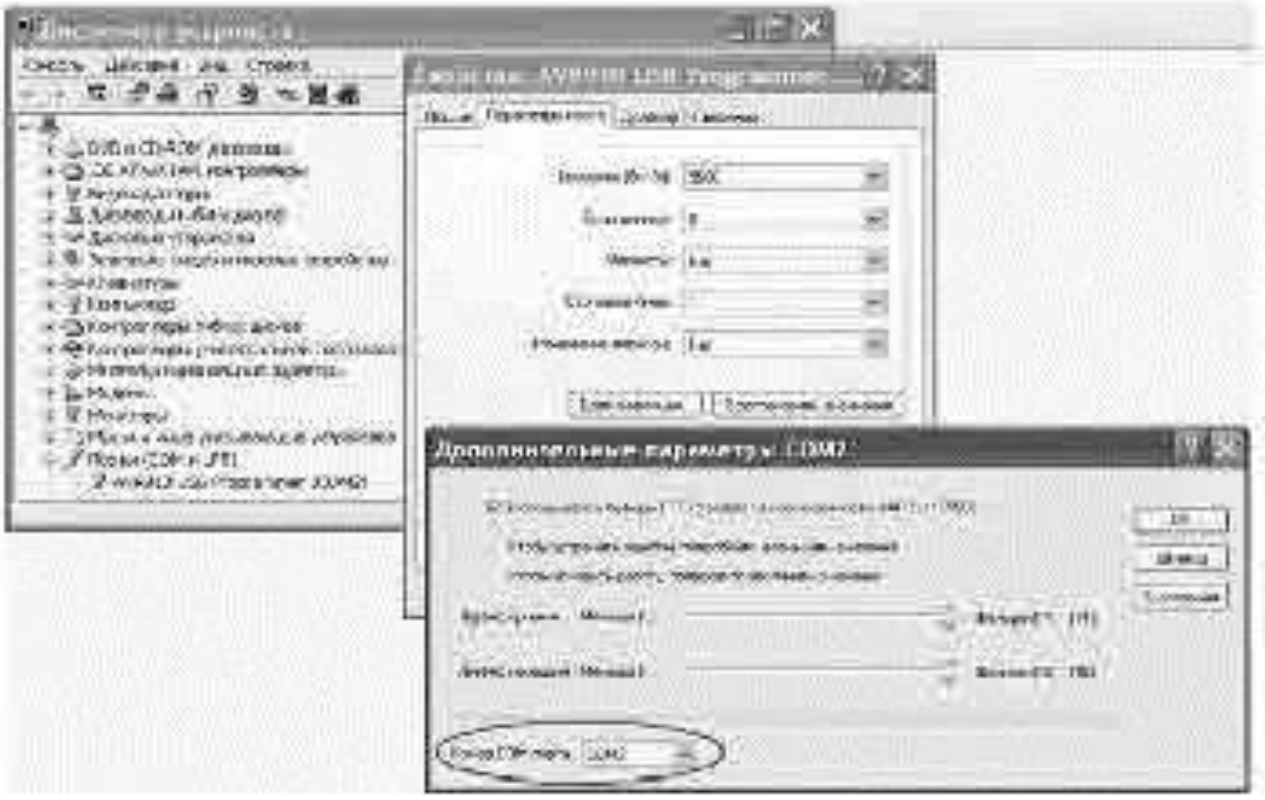

Рисунок 1.18. Встановлення номера порту у диспетчері

1.4.3 Основні налаштування програматору

У програмному забезпеченні, що додається, необхідно правильно встановити тип пристрою (програматора) – обрати AVR910 або AVRISP, а також номер організованого віртуального COM-порту. Нижче наведені приклади встановлення цих параметрів в програмах ChipBlaster/ChipVision і AVR Prog (AVR Studio) (рис. 1.19).

Рисунок 1.19. Встановлення параметрів програматору

Розглянемо прошивку FUSE BITS. Для нормального функціонування контролера в схемі необхідно, щоб були запрограмовані (встановлені в "0") біти SPIEN, CKOPT і SUT0. Зазвичай МК, що поставляються від виробника, тобто нові, мають вже запрограмований біт SPIEN. Також бажано (але не обов'язково) запрограмувати (встановити в "0") біт BODEN, що дозволить роботу вбудованого в МК brown-out детектора. При незапрограмованому біті BODLEVEL встановлення спрацьовування brown-out детектора буде на рівні 2,4...2,9 Вольт. Інші біти мають бути незапрограмовані (встановлені в "1").

Розглянемо можливі несправності програматора і методи їх усунення.

У разі неправильної роботи програматора можливо буде потрібно виконати наступні дії:

1. Змінити швидкість роботи віртуального COM-порту (у диспетчері пристроїв Windows, а також в налаштуваннях керуючих програм) – до 115200 бод.

2. Відключити кабель програматора від USB-порту і через 10-15 секунд підключити знову (особливо може знадобитися при перемиканні між декількома керуючими програмами).

1.4.4 Розрахунок величини струму споживання програматору

Розрахунок струму споживання схеми здійснюється грунтуючись на наступних етапах.

1. Аналіз елементної бази на основі технічної документації з метою визначення максимального струму споживання мікросхемами.

2. Аналіз виводів мікросхем, запрограмованих на передачу інформації з метою визначення витікаючих або впадаючих струмів.

3. Розрахунок сторонніх струмів, присутніх в схемі.

В якості основного елементу обрано мікроконтролер типу AVR, який має наступні характеристики:

- робоча температура (-55…155 °C);
- температура зберігання (-65…150 °C);
- низький рівень на вході RESET (- 0,5V…0,5V);
- допустима напруга на вході RESET (-13V…13V);
- максимальна робоча напруга 6 V;
- максимальний струм одного виводу 40 мА;
- максимальний струм споживання 200 мА;
- максимальна тактова частота 16 МГц.

Струм споживання мікроконтролера при напрузі 5 V і тактовій частоті 8 МГц складає 12 мА, в режимі очікування – 5,5 мА, у вимкненому стані – 15 мкА. Якщо частота вище 8 МГц, то значення струму споживання визначається за допомогою графіку, наведеного в документації.

Для розробленої в даному проекті схеми вибираємо максимальний струм споживання $I_в$=12 мА.

Розрахунок витікаючих струмів через вивід мікроконтролера виконується наступним чином.

PD6 (вивід 12), при логічній 1 на цьому виводі витікаючий струм складає:

$$I_1 = \frac{VCC - VLED}{R_3} = \frac{5V - 2.5V}{330\,Ом} \approx 8(мА) \quad (1.1)$$

де VCC – напруга джерела живлення, яка дорівнює 5 V;

VLED – падіння напруги на світлодіод 2,5 V;

R3 – опір на виході PD6.

PD7 (вивід 13) при логічній 1 на цьому виводі витікаючий струм складає:

$$I_2 = \frac{VCC - VLED}{R_6} = \frac{5V - 2.5V}{330\,Ом} \approx 8(мА), \quad (1.2)$$

де VCC – напруга джерела живлення, яка дорівнює 5 V;

VLED – падіння напруги на світлодіоді 2,5 V;

R6 – опір на виході PD7.

PB5 (вивід 19) при логічній 1 на цьому виводі витікаючий струм складає:

$$I_3 = \frac{VCC - VLED}{R_{11}} = \frac{5V - 2.5V}{330\,Ом} \approx 8(мА), \quad (1.3)$$

де VCC – напруга джерела живлення, яка дорівнює 5 V;

VLED – падіння напруги на світлодіоді 2,5 V;

R11 – опір на виході PB5.

PB3 (вивід 17) при логічній 1 на цьому виводі витікаючий струм складає:

$$I_4 = \frac{VCC - VLED}{R_{12}} = \frac{5V - 2.5V}{330\,Ом} \approx 8(мА), \quad (1.4)$$

де VCC – напруга джерела живлення, яка дорівнює 5 V;

VLED – падіння напруги на світлодіоді 2,5 V;

R12 – опір на виході PB3.

PB2 (вивід 16), при логічній 1 на цьому виводі витікаючий струм складає:

$$I_5 = \frac{VCC - VLED}{R_{13}} = \frac{5V - 2.5V}{330\,Ом} \approx 8(мА), \quad (1.5)$$

де VCC –напрута джерела живлення, яка дорівнює 5 V;

VLED – падіння напрути на світлодіоді 2,5 V;

R13 – опір на виході PB2.

PB1 (вивід 15), при логічній 1 на цьому виводі вигіскаючий струм складає:

$$I_6 = \frac{VCC - VLED}{R_{13}} = \frac{5V - 2,5V}{330\,Ом} \approx 8(мА), \qquad (1.6)$$

де VCC –напрута джерела живлення, яка дорівнює 5 V;

VLED – падіння напрути на світлодіоді 2,5 V;

R14 – опір на виході PB1.

Розрахунок струму споживання по USB інтерфейсу виконується наступним чином.

$$I_7 = \frac{VCC - VD1 - VD2}{R_2 + R_3} = \frac{5V - 0,7V - 0,7V}{1000\,Ом + 68\,Ом} \approx 3(мА), \qquad (1.7)$$

де VCC –напрута джерела живлення, яка дорівнює 5 V;

VD1, VD2 – падіння напрути на діоді 0,7 V;

R1, R4 – опір інтерфейсу USB.

Розрахунок струму через світлодіод VL3 виконується наступним чином.

$$I_8 = \frac{VCC - VD1 - VD2 - VLED}{R_9} = \frac{5V - 0,7V - 0,7V - 2,5V}{430\,Ом} \approx 3(мА), \qquad (1.8)$$

де VCC –напрута джерела живлення, яка дорівнює 5 V;

VD1, VD2 – падіння напрути на діоді 0,7 V;

VLED – падіння напрути на світлодіоді 2,5 V;

R9 – опір на світлодіоді VL3.

Таким чином, сумарний струм споживання складатиме:

$$I_{стр} = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7 + I_8 + I_{світ}, \qquad (1.9)$$

$$I_{стр} = 8 + 8 + 8 + 8 + 8 + 8 + 3 + 3 + 12 = 66\,мА.$$

1.5 Розробка програми ініціалізації програматору для мікроконтролерів ATMEL з ядром AVR

Програмний код для ініціалізації програматора створено в середовищі розробки Code Vision AVR на мові С. Блок-схема алгоритму програми для ініціалізації створюваного програматору наведено на рис. 1.20.

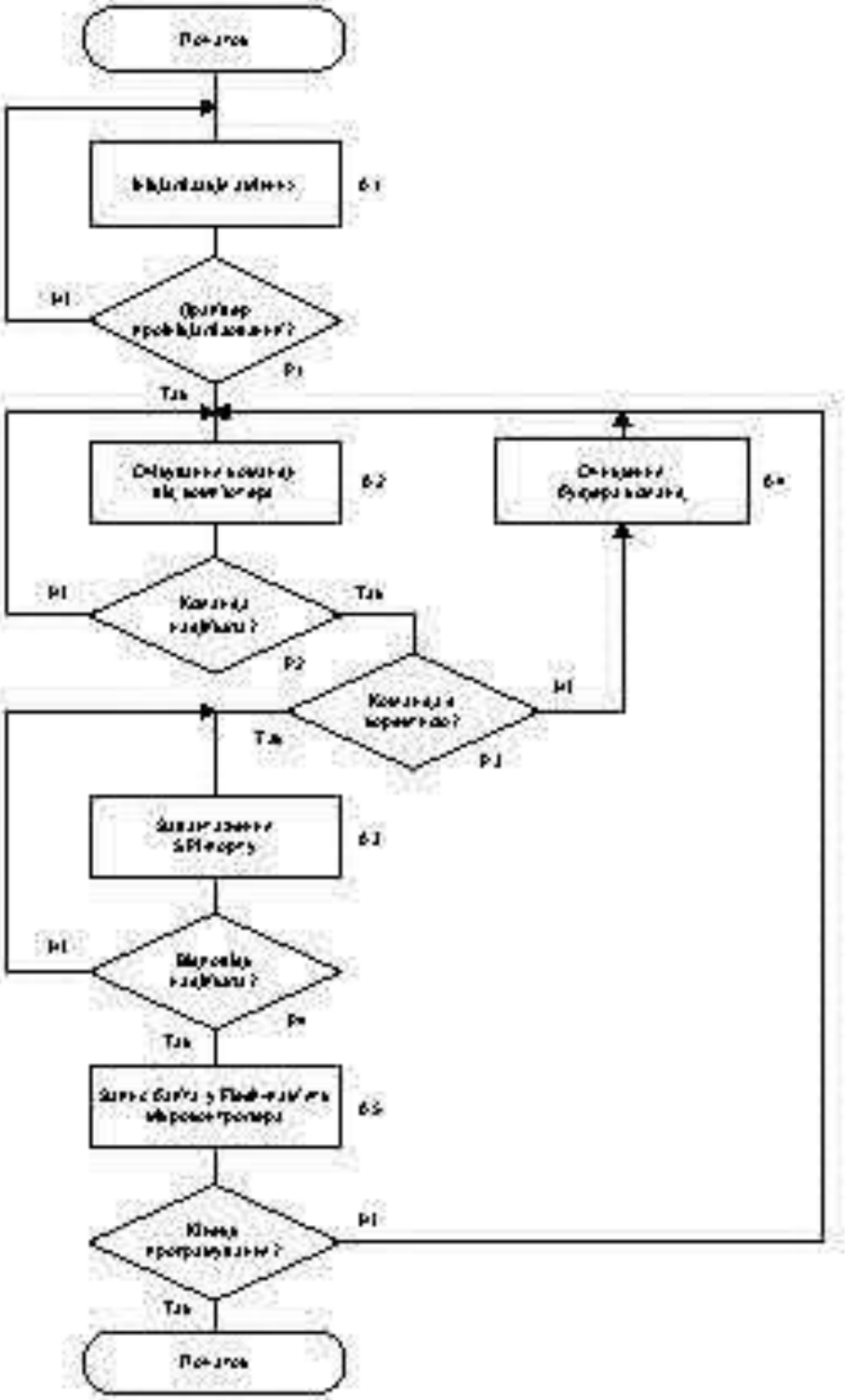

Рисунок 1.20. БСА програми для ініціалізації програматору

Ініціалізація програматору виконується наступним чином (рис.1.20):

А1 – установки констант і елементів пам'яті для початку роботи;

Р1 – завантаження драйвера в пам'ять комп'ютера для роботи з USB-програматором;

А2 – очікується інформаційний пакет для програматору, який складається з команди та необхідних операндів;

Р2 – одна з команд, яка приймає участь у програмуванні мікроконтролеру;

Р3 – перевірка на коректність прийнятої команди;

А3 – завантаження чергового байту для відправлення у програмуємий мікроконтролер;

А4 – підготовка для прийому очікуваної команди;

Р4 – очікування відповіді від периферійного програматора;

А5 – програмування мікроконтролера з використанням інтерфейсу SPI та підтримкою команд.

Вихідними файлами середовища розробки CodeVisionAVR є:

- HEX, BIN або ROM-файл для завантаження в мікроконтролер за допомогою програматора;
- COFF-файл, що містить додаткову інформацію про компіляцію.

Розглянемо процес програмування мікроконтролеру.

Для того, щоб запрограмувати ("прошити") мікроконтролер, необхідний відповідний програматор. Програматор являє собою програмно-апаратний комплекс, який складається безпосередньо з пристрою, що зв'язує мікроконтролер з комп'ютером, і програми, яка цим пристроєм керує. Програматор заносить підготовлену для мікроконтролеру програму (див. додаток) в його пам'ять.

Найбільш поширеним способом програмування для AVR є внутрішньосхемне програмування (функція ISP – in-circuit serial programming) через комунікаційний інтерфейс SPI. Цією можливістю володіють всі мікроконтролери AVR, окрім Tiny11 і Tiny28. Даний режим зручний тим, що

дозволяє програмувати AVR, розташований в готовому пристрої, тобто не потрібно витягувати мікроконтролер з плати кожного разу, коли треба його перепрограмувати. Інтерфейс SPI (Serial Peripheral Interface) являє собою 3 лінії: SCK, MISO і MOSI.

Лінія SCK (SPI ClocK) призначена для тактового сигнал, який програматор формує на лінії SCK.

MOSI (Master Out, Slave In – вхід відомого, вихід ведучого) являє собою лінію передачі даних від програм (ведучий) до програмованого мікроконтролеру (відомий). Під час кожного імпульсу на лінії SCK передається один біт від програматора до програмованого мікроконтролеру по лінії MOSI.

MISO (Master In, Slave Out – вихід відомого, вхід ведучого) являє собою лінію передачі даних від програмованого мікроконтролеру (відомий) до програматора (ведучий). По кожному імпульсу на лінії SCK передається один біт від мікроконтролеру до програматора по лінії MISO.

Для забезпечення нормального зв'язку за трьома SPI-лініями необхідно з'єднати спільну землю (GND) на програматорі та програмованому пристрої. Для входу та перебування в режимі послідовного програмування використовується лінія скидання (RESET). Вона повинна утримуватися в активному стані (низький рівень) під час програмування AVR. Також при стиранні мікросхеми на лінії RESET повинен бути сформований імпульс в кінці циклу стирання. Крім того, може використовуватися вивід контролера XTAL1 для тактування контролера програматором за відсутності кварцового резонатора.

При програмуванні AVR-програматор завжди функціонує як ведучий пристрій, а мікроконтролер як відомий.

Текст програми на мові програмування С, що відповідає наведеному на рис. 1.20 алгоритму, наведено у Додатку А. Розглянемо процес програмування мікроконтролера з використанням розробленого програматору (рис. 1.20).

Програмування здійснювалося наступним чином: необхідно підключити програматор, а потім запустити програму PonyProg. У меню «Настройки»

вибирати «Калибровка» і виконати її. Після цього, у меню «Установки» вибирати «Настройка оборудования» і виставити «Последовательный, COM1 и SI Prog API». Решту полів можна залишити порожніми. Далі треба натиснути «Проверка», ОК. На цьому налаштування завершене. Якщо воно пройшло успішно, можна приступати до програмування самого мікроконтролеру.

Далі необхідно вставити мікроконтролер до роз'єму. У вікні інтерфейсу програми вибрати AVR Micro, у сусідньому – Atmega8. Тепер необхідно клацнути «Команды», «Читать всё». Після цього почнеться читання з мікроконтролеру. По його закінченні з'явиться повідомлення про його успішне завершення. Таке читання, навіть чистої (без записаної програми) мікросхеми, дозволяє налаштувати зв'язок комп'ютер-програматор-мікроконтролер і, якщо процес пройде успішно, можна виконувати програмування мікроконтролера.

Після цього необхідно завантажити в програматор прошивку. По натисненні «Файл, Открыть» треба обрати потрібний файл з розширенням Hex. Після його відкриття, його вміст відобразиться у вікні програми. Тепер дуже важливо правильно конфігурувати сам мікроконтролер. Мікроконтролер є універсальним і на його основі можна створювати різні пристрої. Якраз під прошивку для цього пристрою і буде потрібно встановити конфігураційні біти (фузи) мікроконтролеру так, щоб дана прошивка керувала мікроконтролером коректно. Необхідно натиснути на піктограму з замком, після чого у вікні виставити прапорці для завдання роботи від внутрішнього генератора на 8 МГц, натиснути ОК.

Далі необхідно натиснути «Команды, Записать всё». Після цього почнеться процес запису, а потім перевірка. По її закінченні з'явиться повідомлення «Запись выполнена», після чого треба вийняти мікроконтролер і встановити його у пристрій, де має використовуватись даний мікроконтролер.

2 ЕКОНОМІЧНА ЧАСТИНА

Метою даних розрахунків є обчислення вартості виконання науково-дослідної роботи "Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR " Основна мета даного дипломного проекту є: Розробити апаратно-програмні засоби програматору для мікроконтролерів ATMEL з ядром AVR.

Розподіл робіт по етапах і видах виконавців

Розподіл робіт по етапах і видах виконавців починається . З оцінка якості розробленого проекту в який включає визначення трудомісткості і вартості його створення. Та розрахунок трудомісткості НДР здійснений в наступній послідовності:

1) Складений перелік всіх етапів і видів робіт, які необхідно виконати в ході даної НДР. Після узгодження з керівником проекту допущено виключення, доповнення, об'єднання окремих етапів і видів робіт;

2) По кожному виду робіт визначений кваліфікаційний рівень виконавців.. Розподіл робіт по етапах і видах виконавців виробленний формою, наведено в таблиці.

Етап проведення НДР	Вид робіт	Посада виконавця
Розробка технічного завдання (ТЗ)	1.Складання і затвердження ТЗ для НДР «Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR »	Дипломник, керівник
Вибір напряму дослідження	1. Збір і вивчення науково-технічної літератури. 2. Формулювання можливих напрямів вирішення завдань, поставлених в технічному завданні НДР . 3. Вибір напряму проведення досліджень для подальшої розробки. 4. Розробка плану проведення досліджень для подальшої розробки.	Дипломник керівник

Теоретичні і експериментальні дослідження	1. Технологічний розділ. 1.1Основна функція та архітектура мікроконтролерів. 1.2 Засоби програмування мікроконтролерів AVR. 1.3Розробка структурної схеми програматора. 1.4Розробка принципової електронної схеми програматора для мікроконтролерів. 1.4.1 Загальний опис схеми програматора 1.4.2 встановлення драйверів і програм. 1.5 Розробка програми ініціалізації програматора для мікроконтролерів ATMEL з ядром AVR. 2. Економічна частина 3. Охорона праці	Дипломник керівник консультанти
Узагальнення і оцінка результатів досліджень	1. Узагальнення результатів попередніх етапів роботи. 2. Складання і оформлення звіту. Розгляд результатів проведеної НДР і прийняття результатів в цілому.	Дипломник керівник консультанти

Очікувана трудомісткість робіт

В умовах відсутності нормативної бази тривалість виконання окремих робіт розраховується на основі вірогідних оцінок робіт, що задаються виконавцями.

Вигляд роботи	Очікуваний час виконання
1. Складання і затвердження ТЗ для НДР «Розробка апаратно-програмних засобів програматору для мікроконтролерів ATMEL з ядром AVR»	1
2. Збір і вивчення науково –технічної літератури, технічної документації і інших матеріалів.	4
3. Формулювання можливих напрямів вирішення завдань, поставлених в технічному завданні НДР і їх порівняльна оцінка.	2
4. Вибір напряму проведення досліджень і способів вирішення поставлених завдань. Розробка плану проведення досліджень для подальшої розробки.	2
5. Технологічний розділ.	16
6. Економічна частина	2
7. Охорона праці	2
8 Висновки з переліком використаних джерел.	1
Всього:30	

Результатом виконання НДР є науково-технічна продукція,що є закінчені науково – дослідницькі роботи, виконані відповідно до вимог, передбачених договором, і прийнятими замовником. Розрахунок собівартості і ціни виконання НДР включає наступні статті витрат: витрати на матеріали, основна і додаткова заробітна плата, відрахування до єдиного соціального фонду страхування, витрати на роботи,що виконуються сторонніми організаціями, і деякі інші.

1) Витрати на матеріали,купувальні комплектуючі, напівфабрикати визначають на основі розрахунку потреби в них за оптовими цінами,що діють і складають (ПАПІР формат А4 + друк) = 244 грн.

2) До витрат «Основна заробітна плата» відносяться оплата праці виконавців, безпосередньо притягнених до її виконання. Розмір основної зарплати встановлюється виходячи з чисельності різних категорій виконавців, трудомісткості,що витрачається ними на виконання різних видів робіт, а також їх середньої заробітної плати (ставки) за один робочий день.

Відповідно до статті 8 «Закону про Державний бюджет України на 2022» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39,26 грн. Середня зарплата за один робочий день для кожного виконавця визначена по формулі: Зден= п.т.с. * 8 , де п т с – погодинна тарифна ставка, грн.. 8 – тривалість робочого дня,год. Зден дипломника = 39.26*8 = 314,08 грн. Зден керівника = 50 *8 = 400 грн. Зден консультантів = 50*8= 400грн.

Витрати на основну заробітну плату, НДР,що включаються в собівартість, приведені .

Виконавець	Погодинна тарифна ставка, грн	Денна ставка, грн	Трудомісткість робочих днів	Сума основної зарплати, грн
Дипломник	39,26	314.08	30	9422.04
Керівник	50	400	1	400
Консультант по економічній частині	50	300	0,25	300
Консультант по охороні праці	50	300	0,25	300
Нормоконтроль	50	300	0,25	300
Всього (Зо)				10722.04

3) Витрати на додаткову заробітну плату визначаються у відсотках від основної і враховують виплати за час, що не пропрацював, встановлений законом. У наукових закладах додаткова заробітна плата складає 10-12% від основної

заробітної плати. Зд=10% Зо; Зд= 0 грн

4)До складу собівартості НДР включаються податки. Відрахування до єдиного соціального внеску складає. Зесв=0,22*(Зо + Зд)= 10722.04 + 0; Зесв= 2358грн

5) До накладних витрат відносять витрати на управління і господарське обслуговування, що відноситься до всіх виконуваних НДР.. У наукових закладах накладні витрати складають 40 - 120% від основної і додаткової заробітної плати. Рнакл= (Зо+Зд)*0,4= (10722.04 + 0)*0,4 Рнакл= 4288 грн

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому НДР за формою ,приведеною в табл.

Калькуляція планової собівартості

Статті витрат	Сума ,грн.
1. Матеріали	244 грн..
2. Основна заробітна плата	10722.04грн
3. Додаткова заробітна плата	0 грн
4. Відрахування до єдиного соціального внеску	2358грн
5. Накладні витрати	4288 грн
Планова собівартість (Спл)	17612.04грн

Плановий прибуток визначений по формулі:

Ппл = (0,1*Спл) = 0,1*17612.04 = 1761.20 грн

Де 0,1 – норматив, який враховує граничний рівень рентабельності, встановлений чинним законодавством для науково-технічної продукції.

Договірна ціна визначається по формулі:

Цнір = (Спл + Ппл)=17612.04+1761.20 = 19373.24грн

Звідси ціна реалізації становить:

ПДВ(податок на додану вартість)=(Цнір *0,2)= 19373.24 *0,2

Цр = (Цнір + ПДВ)= (Цнір+ Цнір*0,2) = 19373.24 + 19373.24 *0,2 = 23247.88грн

Ціна роботи = 23247.88грн.

3 ОХОРОНА ПРАЦІ

Охорона праці це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних умов . Яка є частиною будь-якого виробництва чи підприємства у сфері її. За допомогою якого працівник має права з Конституції та закону України про охорону праці, основ трудового законодавства та законодавчих актів на збереження життя, здоров'я та працездатності у процесі трудової діяльності інженера. С початку треба визначити на основі аналізу умов праці за додатком. Та охарактеризувати шкідливі та небезпечні фактори виробничого середовища фізичні та психофізіологічні , що можуть мати місце під час експлуатації устаткування, оцінити ступінь впливу кожного фактора на працівника. Потім еще визначити заходи та засоби захисту від негативного впливу цих факторів.

3.1 Аналіз та умови безпека праці на робочому місці

Спочатку визначаємо небезпечні і шкідливі виробничі фактори у приміщенні, де проходила дипломна робота. А саме фізичні, хімічні, психофізіологічні фактори .

У приміщенні де проходила дипломне проектування було виявлені такі фактори: Фізичні небезпечні шкідливі виробничі фактори :

Це підвищений рівень статичної електрики, підвищений вміст позитивних та негативних аероіонів у повітрі робочої зони та нерівномірність розподілу яскравості у полі зору.

Під час роботи виконуються паяльні роботи, які можуть призвести до можливості отримати опіки середньої або високої тяжкості.

3.2 Хімічні небезпечні й шкідливі виробничі фактори

Виявлено що на робочому місті знаходиться припій у склад якого входить свинець. При роботі з ним виділяються пари, які можуть потрапити в організм людини.

Ще можна підкреслити що каніфоль теж виділяє пари смоли, які можуть при вдиханні людини залишаться в організмі, а саме в легенях. Може ще

привернути увагу ортофосфорна кислота для паяння. При не дотриманні правил експлуатації можна отримати опіки від середніх до тяжких опіків.

3.2.1 Психофізіологічні небезпечні виробничі фактори

При роботі працівник має нервово-психічні перевантаження. А саме розумову перенапругу, перенапругу аналізаторів; монотонність праці. Наприклад, через делікатну роботу з мікросхемами, де треба усидливість і акуратність. Ще можна додати роботу з комп'ютером, де треба виявляти уважність.

3.2.2 Розробка заходів з охорони праці. Мікроклімат

Параметри мікроклімату в робочій зоні приміщення.

Період року	Температура °С	Відносна вологість	Швидкість руху кисню, м/с
Холодний Холодний	Оптимальна 21-23	Допустима на робочих місцях 75	Допустима на постійних та непостійних не більш 0,2
	Допустима на робочих місцях 21-25	Оптимальна 45-60	Оптимальна 0,1
Теплий	Оптимальна 22-24	Допустима на робочих місцях 55 при 27°С	Допустима на постійних та непостійних не більш 0,1-0,3
Теплий	Допустима на робочих місцях 21-28	Оптимальна 40-60	Оптимальна 0,2

Приміщення відноситься до категорії Іб. Тобто приміщення за нормованими параметрами відповідає стандартами згідно до витягу із ГОСТ 12.0. 003-74..

3.2.3 Різні іонізації повітря приміщень при роботі на ВДТ та ПЕОМ

При праці з приладами або ЕВМ, для стабільної роботи приладів, приміщення має добре провітрюватись .Також для гарного стану людини ,щоб не було нестачі повітря в приміщеннях. Відповідно до ГОСТ 12.1.005-88 вміст озону в повітрі робочої зони не повинен перевищувати 0,1мг/м, вміст оксидів азоту – 5 мг/м, вміст пилу – 4 мг/м.

В даному приміщенні рівень іонізації повітря становить 1500-3000 іонів в 1 куб.см повітря. Тобто це приміщення відповідає нормам ГОСТ.

3.2.4 Електробезпека

Це приміщення відноситься до приміщень без підвищеної небезпеки, в яких відсутні умови, що створюють підвищену або особливу небезпеку. Як можна побачити в даному приміщені є електростатичні вимики та аварійне відключення при замиканні. Відповідно до ДНАОП 0.00-1.32.01 Дане приміщення відповідає нормам стандарту.

3.3 Пожежна безпека речовин

Під час праці використовуються горючі речовини. А саме це :

1 Алюміній (аерозоль) $t_{самозаймання}{}^0$ С = 360

2 Ебоніт (аерозоль) $t_{самозаймання}{}^0$ С = 460

3 Формалін технічний $t_{сп}$,^{0}С = 38, $t_{самозаймання}{}^0$ С= 435

4 Спирт $t_{сп}$,^{0}С = 67 , $t_{самозаймання}{}^0$ С =750

Ці суміші і технічних продуктів відносяться (ГОСТ 12.1.004 – 85)

Відповідно до НАПБ Б.03.002-2007. приміщення де проводиться робота відповідно технологічної частини дипломного проекту відноситься до категорії Д , так як в даному приміщенні немає таких хімічних речовин, які в холодному стані можуть вибухнути або самозайматися.

Але в цьому приміщені знаходяться та використовуються матеріали , які можуть спалахнути, як наприклад гума або пластик, діоди, транзистори, резистори.

Первинні засобами пожежогасіння

У будь якому приміщені повинні бути засоби пожежонегасіння. Це приміщення відноситься до категорії В ,за відсутності горючих газів і рідин.

Для цього в приміщенні обов'язково необхідні порошкові, повітряно-пінні, вуглекислотні вогнегасники або пісок . Переносний вогнегасник має масу із зарядом вогнегасної речовини, 2 кг . Ще підкреслимо, що вогнегасник повинен бути на видному і доступному місці.

Згідно до типовими нормами належності вогнегасників ПАПБ Б.03.001-2004:

ВИСНОВКИ

У даному дипломному проекті була поставлена задача розробити апаратно-програмні засоби програматору для мікроконтролерів ATMEL з ядром AVR. В результаті виконання роботи реалізовані наступні етапи:

- сформульовано мету та задачі роботи;
- доведена актуальність тематики даного дипломного проекту;
- проаналізовано технічну доцільність використання ISP-програматору;
- розроблено структурну та принципову схеми пристрою;
- виконано розрахунок величини струму споживання ISP-програматору;
- розроблено програму ініціалізації для ISP-програматору;
- сформульовано техніко-економічне обґрунтування використання даного пристрою;
- передбачені заходи безпеки при роботі з розробленим пристроєм.

Розроблений ISP-програматор дозволяє запрограмувати мікроконтролер серії AVR фірми ATMEL. Можливості таких мікроконтролерів дуже широкі, наприклад, мікроконтролеру можна доручити вимір різноманітних величин, обробку різних сигналів і керування широким спектром різних виконавчих пристроїв. Основною перевагою є використання одного мікроконтролера, в якому інтегровані усі необхідні компоненти, замість декількох окремих мікросхем (процесор, ОЗП, ПЗП, периферія), що дозволяє знизити вартість пристрою, зменшити його розміри, енергоспоживання, а також витрати на розробку і збірку. Використання розробленого ISP-програматору дозволяє ще більше знизити вартість розробки, тому що професійні програматори мають на порядок більшу вартість. Окрім того, після проведених розрахунків струму споживання можна зробити висновок, що розроблений ISP-програматор може отримувати живлення безпосередньо з USB-порту, що дозволить відмовитись від зовнішнього джерела живлення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Є. М. Смирнов – Розробка радіоелектронних схем на основі мікроконтролерів (на прикладі AVR мікроконтролерів фірми Atmel): Київ: Радіофізичний факультет КНУ ім. Тараса Шевченка, 2013. – 74 с.
2. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. / В. Н. Баранов – Издательство: Додэка. – 2004. - 289 с.
3. Atmel Flash Microcontrollers. Product Portfolio – Atmel corporation. – 2012. - 28 p.
4. Абраш Р. О. Книга по работе с WinAVR и AVR Studio. – Підбірка статей у журналі «Радиолюбитель» за період 01/2010–05/2011. – 88 стр.
5. Мортон Дж. Микроконтроллеры AVR. Вводный курс / Джон Мортон - Издательство: Издательский дом Додэка – XXI. – 2006. – 272 с.
6. Плахтеев А. П. Сравнительная оценка сред разработки устройств на AVR микроконтроллерах / А. П. Плахтеев, Н. С. Шангалова – Вестник ХНАДУ, випуск – 2011. – 53 с.
7. Code Vision AVR High Performance C Compiler for Atmel AVR [Електронний ресурс]. – Режим доступу : http://www.x-graph.be/codevision.html
8. AVR Studio 4 [Електронний ресурс]. – Режим доступу : http://atmel.com/dyn/products/
9. AVR Libc Development Pages [Електронний ресурс]. – Режим доступу : http://www.nongnu.org/avr-libc
10. WinAVR [Электронный ресурс]. – Режим доступу : http://winavr.sourceforge.net/index.html (англ.).
11. Программирование микроконтроллеров AVR [Електронний ресурс]. Режим доступу : http://www.mytobot.ru/-stepbystep/mc_programming.php.

ДОДАТОК А. Текст програми на мові С для ініціалізації програматору

```
#include "stdafx.h"
#include "usbdrv.h"
#include "usb_descriptors.h"
#include "avr910.h"
#include <string.h>

/* Команда USB CDC-хосту */
enum
{
    SEND_ENCAPSULATED_COMMAND = 0,
    GET_ENCAPSULATED_RESPONSE,
    SET_COMM_FEATURE,
    GET_COMM_FEATURE,
    CLEAR_COMM_FEATURE,
    SET_LINE_CODING = 0x20,
    GET_LINE_CODING,
    SET_CONTROL_LINE_STATE,
    SEND_BREAK
};
static UCHAR        intr3Status; // керування передачами
//кінцевих точок переривання
static UCHAR        baudsetup[8];
/*************************************************************************/
UCHAR usbFunctionSetup(UCHAR data[8])
{
        usbRequest_t *rq = (void *)data;
   if((rq->bmRequestType & USBRQ_TYPE_MASK)== USBRQ_TYPE_ CLASS)
   {
        // тип запиту класу
                if( rq->bRequest== GET_LINE_CODING ||
rq->bRequest= =SET_ LINE_CODING )
      {
         GET_LINE_CODING -> usbFunctionRead()
         SET_LINE_CODING -> usbFunctionWrite()
                return 0xff;
      }
      if(rq->bRequest == SET_CONTROL_LINE_STATE)
      {
         // Повідомляти про послідовний стан
         //(визначення несучої)
         // tty-пристрої можуть бути відкритими
         //лише коли встановлене визначення несучої.
         intr3Status = 2;
      }
   }
   return 0;
}
/*************************************************************************/
```

```
UCHAR usbFunctionRead( UCHAR *data, UCHAR len )
{
        memcpy(data, baudsetup, len);
        return 7;
}
/*---------------------------------------------------------------------------*/
UCHAR usbFunctionWrite( UCHAR *data, UCHAR len )
{
        SET_LINE_CODING
        memcpy(baudsetup, data, len);
        return 1;
}

/*---------------------------------------------------------------------------*/
void usbFunctionWriteOut(UCHAR *data, UCHAR len)
{
        usbDisableAllRequests();
    AVR910_Command(data, len);
// Для кожного блоку, який отримано від хоста,
// запускається порожній блок у протилежному напрямку.
// Це виправляє зависання у Windows.
// Виконувати лише при необхідності.
#if 1
    if(usbInterruptIsReady())
        usbSetInterrupt(0, 0);
#endif
}
/*---------------------------------------------------------------------------*/
static void hardwareInit(void)
{
        /* відключаємо pull-ups від ліній USB */
    USB_CFG_IOPORT &= ~((1<<USB_CFG_DMINUS_BIT)|
(1<<USB_CFG_DPLUS_BIT));
    // завантажуємо параметри за умовчанням для емуляції послідовного порту
(*(ULONG *)(baudsetup[0])) = (ULONG)115200; /* швидкість обміну 115200 бод */
    baudsetup[4] = 0; /* один stopbit */
        baudsetup[5] = 0; /* немає паритету */
    baudsetup[6] = 8; /* 8 біт даних в кадрі */
        /* затримка >10ms для скидання USB */
    __delay_cycles(150000);
}
/*---------------------------------------------------------------------------*/
__C_task main(void)
{
        hardwareInit();
        AVR910_Init();
    usbInit();
    SEI();
    usbSetInterrupt(0, 0);
    // головний цикл подій
    for(;;)
    {
        UCHAR *data;
        UCHAR bytesRead;
```

```
        usbPoll();

        if(usbAllRequestsAreDisabled())
          usbEnableAllRequests();

        if(usbInterruptIsReady() && (bytesRead = AVR910_IsRxData(&data)) > 0)
        {
        // зменшення кількості даних до розміру посилки USB
          if(bytesRead > 7)
                if(bytesRead == 8)
                bytesRead = 7;
            else
                bytesRead = 8;
  // встановлення буферу даних для драйвера USB і
  //з вільнення буферу даних програматора від надісланих даних
           usbSetInterrupt(data, bytesRead);
           AVR910_ClearTXData(bytesRead);
        }
        // Необхідно відправити rx та tx пакети після спроби відкриття порту
        if(intr3Status != 0 && usbInterruptIsReady3())
        {
          static UCHAR serialStateNotification[10] = {0xa1, 0x20, 0, 0, 0, 0, 2, 0, 3, 0};
           if(intr3Status == 2)
                usbSetInterrupt3(serialStateNotification, 8);
           else
                usbSetInterrupt3(serialStateNotification+8, 2);
           intr3Status--;
        }
    }
}
```