Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Автоматики і комп'ютеризованих технологій_____
(повна назва)

Кафедра _Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки_
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

_____перший (бакалаврський)_____
(рівень вищої освіти)

SPIN-бібліотека для відображення інформації з мультипроцесорної системи
(тема бакалаврської роботи)

Виконав:
здобувач 4 курсу, групи АКТСІ-20-3
_____Логінов М.О._____ ____

Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології

Тип програми освітньо-професійна

Освітня программа Системна інженерія

Керівник проф. Овчаренко В.Є.

Допускається до захисту

Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024

Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет Автоматики і комп'ютеризованих технологій

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Освітньо-кваліфікаційний рівень перший (бакалаврський)

Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
(шифр і назва)

Освітньо-професійна програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ
Зав. кафедри КІТАМ_____

"_____" _____2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачу Логінову Микиті Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи SPIN-бібліотека для відображення інформації з мультипроцесорної системи

затверджені наказом по університету від 03.06.2024 р. № 545 Ст.

2. Термін подання здобувачем роботи 20.06.2024

3. Вихідні дані до роботи мікроконтролер Parallax Propeller P8X32A, VGA-інтерфейс, SPIN-мова програмування.

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

4.1 Вступ

4.2 Аналіз апаратних ресурсів, здатних формувати відповідні сигнали інтерфейсу VGA для подальшого відображення їх на дисплеї

4.2 Проведення аналізу алгоритмів та існуючих функцій мовою SPIN

4.3 Розробка SPIN-бібліотеки

4.4 Аналіз небезпечних і шкідливих виробничих факторів

4.5 Практична реалізація бібліотеки

4.6 Налаштування генерації відео сигналу через інтерфейс VGA

4.6 Висновки

4.7 Перелік посилань

4.8 Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt)

6. Консультанти розділів роботи

Найменування Розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	04.06.2024	
2	Аналіз технічного завдання	05.06.2024	
3	Огляд літератури з теми дослідження	09.06.2024	
4	Аналіз апаратних ресурсів, здатних формувати відповідні сигнали інтерфейсу VGA для подальшого відображення їх на дисплеї	11.06.2024	
5	Проведення аналізу алгоритмів та існуючих функцій мовою SPIN	14.06.2024	
6	Розробка SPIN-бібліотеки та практична реалізація бібліотеки	16.06.2024	
7	Налаштування генерації відео сигналу через інтерфейс VGA	17.06.2024	
8	Оформлення пояснювальної записки	19.06.2024	
9	Подання атестаційної роботи в ЕК	20.06.2024	

Дата видачі завдання 01.04.2024

Студент _____ Логінов М.О.
(підпис) (прізвище та ініціали)

Керівник роботи (проекту) _____ проф. Овчаренко В.Є.
(підпис) (посада,прізвище та ініціали)

Я, Логінов Микита Олександрович, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав та не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

20.06.2024  Логінов М.О.

РЕФЕРАТ

Пояснювальна записка: 90 сторінок, 9 таблиць, 16 рисунків, 2 додатки, 30 джерел.

МЕТОДИ, АЛГОРИТМ, ТЕХНОЛОГІЯ, ЗАСОБИ, МІКРОКОНТРОЛЕРИ, СИСТЕМИ ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ.

Бакалаврська кваліфікаційна робота присвячена розробці SPIN-бібліотеки для відображення інформації з мультипроцессорної системи побудованої на основі мікроконтролера Parallax Propeller P8X32A. Метою роботи є розробка SPIN-бібліотеки, об’єктом дослідження мікроконтролер Parallax Propeller P8X32A, а предметом дослідження підсистема відображення інформації.

В основній частині роботи досліджено принцип функціонування обчислювальних ядер і процедура перемикання між ними. Поведено аналіз архітектури мікроконтролера Parallax Propeller і мов програмування. Зроблено висновок про доцільність застосування SPIN-бібліотеки для відображення інформації з мультипроцесорної системи побудованої на основі мікроконтролера Propeller. Також, у роботі розроблена алгоритмічна і програмна структура SPIN-бібліотеки. програмної частини бібліотеки. Основними функціями розробленої бібліотеки є вивід графічної і текстової інформації на екран по інтерфейсу VGA. Кожному з примітивів бібліотеки можна задавати необхідний колір і колір фону.

У розділі Охорона праці був проведений аналіз небезпечних і шкідливих виробничих факторів, розглянуті питання промислової безпеки, виробничої санітарії та пожежної безпеки в умовах виробництва.

Отримані результати бакалаврської кваліфікаційної роботи орієнтовані на практичне використання розробленої бібліотеки для мікроконтролерів серії Propeller.

ABSTRACT

The explanatory note contains: 90 pages, 9 tables, 16 drawings, 2 applications, 30 sources.

METHODS, ALGORITHMS, TECHNOLOGIES, TOOLS, MICROCONTROLLERS, DATA DISPLAY SYSTEM.

Bachelor's qualification work is devoted to the development of SPIN-library to display information from a multiprocessor system built on the basis of microcontroller Parallax Propeller P8X32A. The purpose of the work is the development of the SPIN library, the object of research is the Parallax Propeller P8X32A microcontroller, and the object of research is the information display subsystem.

In the main part of the analysis of the microcontroller, namely the principle of the studied cores and how to switch between them. Also analyzed the Parallax Propeller microcontroller architecture and programming languages. The conclusion about expediency of SPIN-library for displaying information from a multiprocessor system built on the basis of microcontroller Propeller. Also in the developed algorithmic and program structure SPIN-library. program of the library. The main functions of the developed library is the conclusion of graphic and text information on the screen is the interface VGA. Each of the primitives library, you can specify the desired color, and background color.

In the section "Health" was analyzed dangerous and harmful factors, the analysis of the "man-machine-environment", the issues of industrial safety, industrial hygiene and fire safety in production.

The results of baccalaureate work focused on the practical use of the developed library for microcontroller series Propeller.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЦП – аналогово-цифровий перетворювач;

ДЖ – джерело живлення;

ІМС – інтегральна мікросхема;

МК – мікроконтролер;

ПЕОМ – персональна електронно-обчислювальна машина;

ПЗ – програмне забезпечення;

ПЗП – постійно запам'ятовуючий пристрій;

ПЛІС – програмована логічна інтегральна схема;

ОЗП – оперативно запам'ятовуючий пристрій;

ОЦ – обчислювальний центр;

РСФ – регістр спеціальних функцій;

САУ – система автоматичного управління;

ЦАП – цифро-аналоговий перетворювач;

ЦП – центральний процесор;

ШІМ – широтно-імпульсна модуляція;

COM – послідовний порт комп'ютера;

RISC (англ. Reduced Instruction Set Computing) – обчислення зі скороченим набором команд;

USART – універсальний синхронно асинхронний приймач;

VGA – стандарт моніторів та відеоадаптерів.

ВСТУП

Сьогодні керування засобами відображенням інформації переходить від персональних комп'ютерів до мікроконтролерів. Сама інформація стає основним об'єктом торгівлі, нагромадження, обміну. У таких умовах першочергового значення набувають засоби збору, відображення та передачі інформації. Засоби відображення інформації є технічною основою побудови інформаційної моделі будь-якої системи моніторингу та управління.

За допомогою різних технічних елементів індикації створюють засоби відображення інформації, які можуть бути виконані у вигляді табло, мнемосхем, панелей приладів, щитів. У водночас елементом управління системою індикації служить, зазвичай персональний комп'ютер, чи програмований логічний контролер, чи потужний микроконтроллер. З перерахованого вище можна зробити висновок, що найбільш економічно обгрунтованим вибором є система відображення інформації побудована на основі мікроконтролера.

Аналізуючи сучасний ринок мікроконтролерів можна дійти невтішного висновку, що мікроконтролер Propeller компанії Parallax сильно відрізняється від інших за багатоядреності і присутності у архітектурі відео-генератора. Варто відзначити, що в архітектурі мікроконтролера Propeller важливою є взаємодія ядер і перемикачів між ними, чим робить його практично унікальним як у функціональності, так і у відповідній ціні.

В результаті відокремленості мікроконтролера та унікальності мови програмування SPIN актуальною є завдання розробки SPIN-бібліотеки, здатної вирішити задачу відображення інформації, отриманої мультипроцесорною системою.

Таким чином метою роботи є розробки SPIN-бібліотеки.

Об’єктом дослідження є мікроконтролер Parallax Propeller P8X32A.

Предметом дослідження – підсистема відображення інформації.

Для виконання мети роботи необхідно:

– аналіз апаратного пристрою мікроконтролера Parallax Propeller;

– розробити алгоритмічну та програмну структуру SPIN-бібліотеки.

Таким чином в основній частині бакалаврської дипломної роботи проведено аналіз апаратного пристрою мікроконтролера Parallax Propeller, а також розроблено алгоритмічну та програмну структуру SPIN-бібліотеки. Основними функціями розробленої бібліотеки є виведення графічної та текстової інформації на екран за інтерфейсом VGA. Кожному з примітивів бібліотеки можна задавати потрібний колір та колір фону.

У розділі Охорона праці розглядаються питання безпеки життя та діяльності співробітників обчислювального центру. Було проведено аналіз небезпечних та шкідливих виробничих факторів, розглянуто питання промислової безпеки, виробничої санітарії та пожежної безпеки в умовах виробництва.

Робота виконана згідно ДСТУ 3008 – 2015 [1] та рекомендацій [2-3]. Сам мікроконтролер Parallax P8X32A Propeller є унікальним багатоядерним мікроконтролером, розробленим компанією Parallax Inc. Він відомий своїми особливостями, які роблять його придатним для різних задач, включаючи робототехніку, обробку сигналів, ігрові консолі та системи реального часу. Мікроконтролер Parallax P8X32A Propeller залишається актуальним на 2024 рік завдяки своїй унікальній багатоядерній архітектурі, що забезпечує високу продуктивність і багатозадачність. Його гнучкі можливості введення/виведення, проста в освоєнні мова програмування SPIN і активна підтримка спільноти розробників роблять його привабливим для різних застосувань. Крім того, його здатність використання в системах реального часу та енергоефективність забезпечують конкурентоспроможність у сучасних вбудованих системах та системах автоматизації.

1 АНАЛІЗ І РОЗРОБКА АПАРАТНИХ ЗАСОБІВ ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ

1.1 Аналіз технічного завдання та постановка завдання розробки SPIN-бібліотеки

Метою бакалаврської кваліфікаційної роботи є розробка SPIN-бібліотеки для відображення інформації з мультипроцесорної системи. Основне завдання – розробка алгоритмічних та програмних засобів для виведення інформації отриманої мікропроцесорної системою на екран.

Основною вимогою, що пред'являється до бібліотеки, що розробляється, є можливість підключення монітора або РК-дисплея на основі інтерфейсу VGA.

У зв'язку з цим у бакалаврській роботі передбачається реалізація задачі, що складається з кількох рівнів:

– аналіз апаратних ресурсів, здатних формувати відповідні сигнали VGA інтерфейсу для подальшого відображення їх на дисплеї;

– проведення аналізу алгоритмів та існуючих функцій мовою SPIN для відображення інформації на дисплеї;

– власне, розробка SPIN-бібліотеки.

Існуючі функції мовою SPIN для VGA інтерфейсу мають обмежену функціональність і, зазвичай, реалізовують одне певне завдання, наприклад – виведення текстової інформації чи виведення массиву, як бітової матриці. У той же час розробка повнофункціональної SPIN-бібліотеки для завдання синхронної мульти-потокової обробки в реальному часі є актуальною.

1.2 Аналіз роботи мікроконтролера Parallax Propeller

Серед безліч представлених на ринку мікроконтролерів, мікроконтролер Propeller компанії Parallax сильно відрізняється від решти. Багато мікроконтролерів містять одне обчислювальне ядро будь-якої архітектури та розрядності, функціональну систему переривань, пам'ять програм, пам'ять

даних і певний набір периферійних пристроїв. Основна відмінність Propeller – це його багатоядерність, що важливо для технічних засобів автоматизації [4-5].

Мікроконтролер Parallax Propeller розроблений для забезпечення високошвидкісної обробки даних у вбудованих системах. Інтегральна мікросхема Propeller (P8X32A) забезпечує гнучкість і продуктивність за рахунок своїх восьми процесорів-ядер[6]. Ці процесори можуть одночасно виконувати незалежні чи спільні завдання.

Слід зазначити, що на ринку мікроконтролерів[7], що бурхливо розвивається, мікроконтролер Propeller від компанії Parallax якісно відрізняється від інших. Оскільки він має вісім 32-розрядних незалежних абсолютно однакових обчислювальних модулів або cog'ів (cog – зубець шестерні), схема роботи такої архітектури наведена на рисунку 1.1. [8].

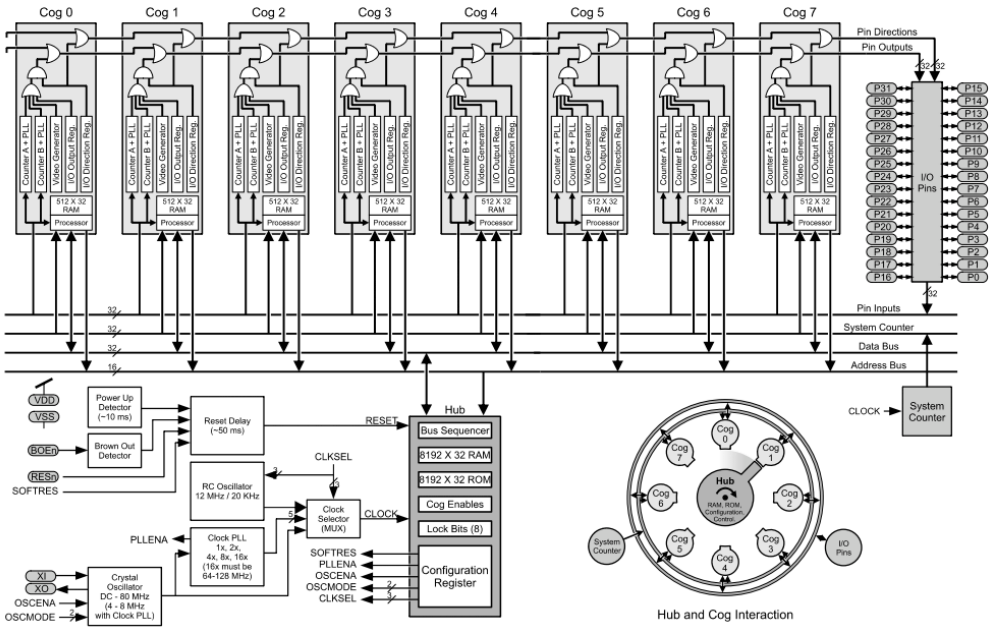
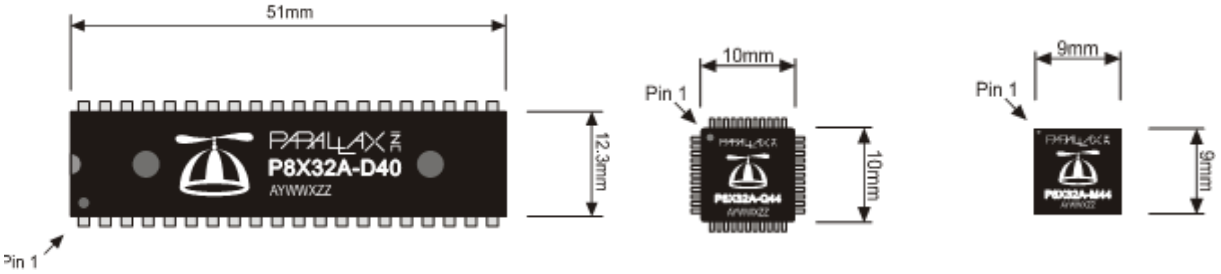


Рисунок 1.1 – Архітектура мікроконтролера Parallax Propeller P8X32A

Варто відзначити, що в архітектурі мікроконтролера Propeller важливим є взаємодія ядер (Cog) і перемикачів (Hub) між ними[8]. Саме перемикач контролює, який з ядер мікроконтролера може отримати доступ до

взаємовиключних ресурсів, таких, як основні ОЗП/ПЗП, регістри конфігурації та інші ресурси. Перемикач надає окремий доступ кожному ядру в момент часу за круговою схемою незалежно від того, скільки ядер в даний момент запущено, підтримуючи таким чином синхронізацію.

Конструктивне виконання мікроконтролера Parallax можливе у DIP, LQFP та QFN корпусах, що надає великий вибір для проектування готової мікропроцесорної системи відображення інформації, рисунок 1.2.



а) DIP-корпус б) LQFP-корпус в) QFN-корпус

Рисунок 1.2 – Зовнішній вигляд реалізації корпусів мікроконтролера Parallax

Асинхронні процеси обробляти простіше, ніж у пристроях, які використовують для цього переривання. Мікроконтролер Propeller не потребує переривань; йому потрібно лише призначити деяким з Cog-ов свої власні високошвидкісні завдання, тоді як інші матимуть цілком вільні ресурси. В результаті виходить система з малим часом відгуку, яку легше адаптувати для відображення інформації.

Наступна особливість: дуже гнучка система таймерів[9]. У мікроконтролера по 2 лічильники-таймери на ядро і кожен таймер має свій PLL, що дозволяє працювати останньому на частотах до 128 МГц. Крім звичайних режимів рахунку, генерації ШІМ та управління виводами, таймери мають досить специфічні режими, такі як генерація TV сигналу, що вкрай необхідно для реалізації системи відображення інформації.

У мікроконтролері досить особливе розбиття пам'яні, а саме індивідуальна пам'ять ядра – 2кБ (є у кожного ядра) та загальна пам'ять на 64кБ.

Завдяки таким можливостям, Parallax Propeller може бути застосований у завданнях, де без ПЛІС не можливо було обійтися, або там, де зазвичай потрібно DSP-процесор.

1.2.1 Аналіз роботи обчислювальних ядер

До складу мікроконтролера Propeller P8X32A входить вісім обчислювальних ядер, званих Cog, з номерами від 0 до 7. Кожен Cog складається з однакових компонентів, рисунок 1.3, при цьому всі обчислювальні ядра виконані абсолютно однаковими і можуть виконувати завдання незалежно один від одного[9].

Кожне ядро – це 32-розрядний обчислювальний модуль. Даний модуль має свою локальну пам'ять, доступну тільки йому і копію регістрів введення-виведення. Банк локальної пам'яті має обсяг 512 машинних слів, тобто, 2048 байт. У цій пам'яті розміщується виконувана ядром мікропрограма, у ній відображаються 16 регістрів вводу-виводу, у ній повинні бути всі локальні дані з якими виконуються дії. Так як такий блок регістрів в ядрі відсутня, дії виробляються прямісінько з осередками пам'яті.

Кожен Cog має свою власну ОЗП, звану Cog RAM, яка складається з 512-ти 32-х бітних регістрів. Все ОЗП (Cog RAM) є пам'яттю загального призначення, крім останніх 16 регістрів – регістрів спеціальних функцій (РСФ) [9].

Пам'ять Cog RAM використовується для зберігання коду, даних і змінних, а на останні 16 адрес служать інтерфейсом до системного лічильника, ліній вводу/виводу і локальної периферії ядра.

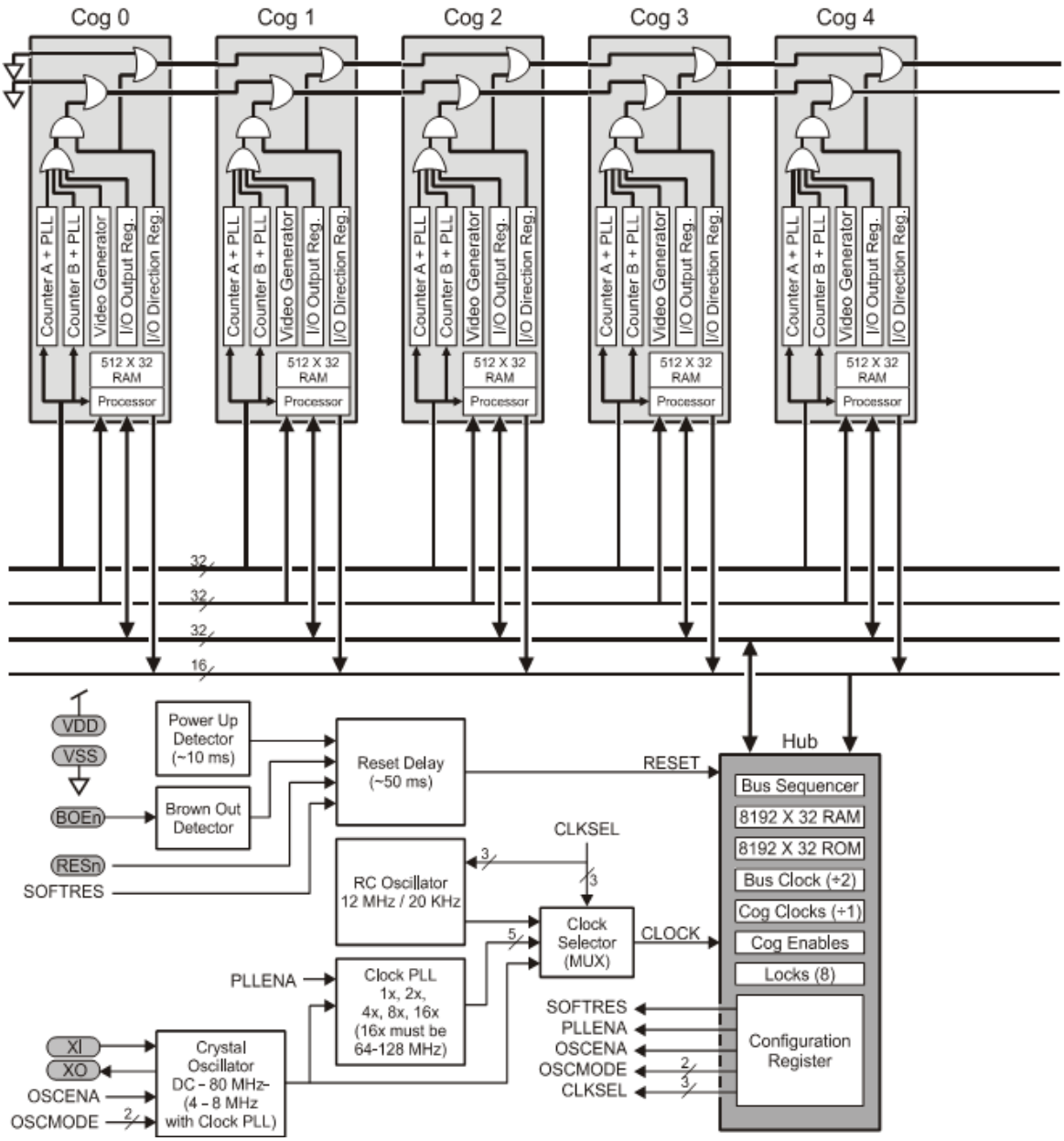


Рисунок 1.3 – Компоненти обчислювальних ядер мікроконтролера

При завантаженні ядра адреси з 0 ($000) до 495 ($1EF) послідовно завантажуються з основної ОЗП/ПЗП, а його регістри спеціальних функцій з адресами від 496 ($1F0) до 511 ($1FF) обнуляються. Після завантаження, Cog починає виконання інструкцій, починаючи з $000 в його ОЗП (Cog RAM). Він продовжує виконувати код до тих пір, поки не буде зупинений, перевантажений самим собою або іншим ядром, або поки не виникне сигнал «Скинути».

У кожного ядра також є невеликий набір периферії: доступ до портів введення-виводу і 2 однакових таймера-лічильника.

Усі вісім ядер тактуються від джерела системної частоти, тому в кожного їх одна й та сама тимчасова база; всі активні Cog-і виконують інструкції одночасно. Кожне ядро може бути запущено або зупинено під час виконання програми, може бути запрограмоване для одночасного виконання спільних завдань: незалежно або скоординовано з іншими Cog-ом через основне ОЗП. Незалежно від природи використання Cog-ів, розробник додатків для IMC Propeller має повний контроль над тим, як і коли кожен з процесорів буде задіяний; ні компілятор, ні якась операційна система не займаються поділом завдань між Cog-ами. Така побудова системи дає можливість абсолютно чіткого узгодження ядер у часі, контроль споживання та відгуку на зовнішні події при розробці вбудованих систем.

1.2.2 Аналіз роботи перемикачів ядер

Перемикач ядер мікроконтролера Propeller послідовно комутує кожне ядро до загальних ресурсів, таких, як загальна пам'ять. Така комутація організовується в чергу для доступу до ресурсів між обчислювальними ядрами від Cog0 до Cog7 і від Cog0 за круговою схемою. Саме з цієї причини, інструкції, що працюють із загальними ресурсами, непередбачувані за часом виконання і можуть займати від 7 тактів (у найкращому випадку, коли HUB на підході) до 22 тактів у гіршому випадку (коли HUB тільки обробив дане ядро) [9].

Маючи кілька обчислювальних ядер принципи програмування під такий мікроконтролер змінюються. Можна не заводити окреме переривання по таймеру для опитування зовнішньої клавіатури та оновлення даних на екрані, а виділити окреме ядро і займатися в ньому лише однією елементарною дією. У деяких випадках немає необхідності в операційній системі: багато її функцій вже реалізовано в апаратній частині.

Необхідно пам'ятати, що Hub-інструкції одного Cog-а не заважають виконанню інструкцій інших ядер завдяки архітектурі самого перемикача ядер (рисунок 1.4).

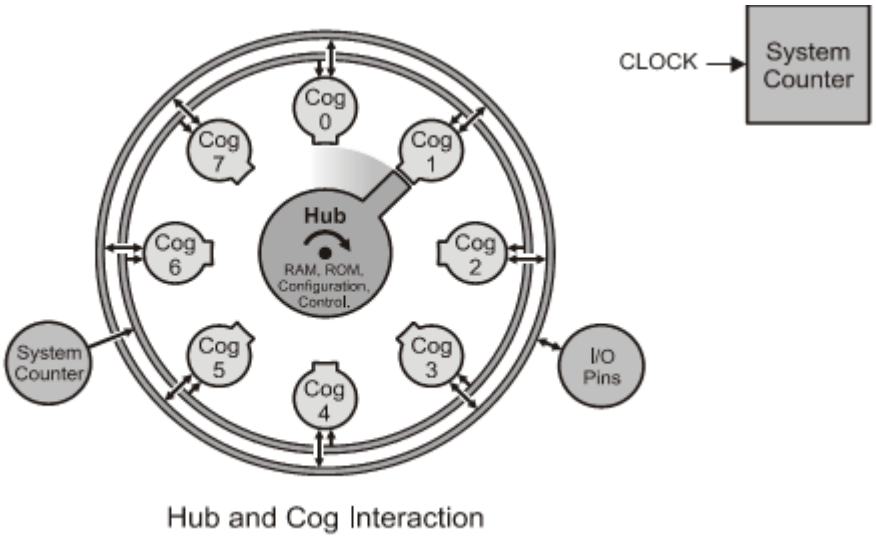


Рисунок 1.4 – Архітектура перемикача ядер

Для отримання максимальної ефективності від асемблерних процедур, які потребують частого доступу до взаємовиключних ресурсів, необхідно чергувати Hub-інструкції з інструкціями, які не потребують доступу до Hub, щоб зменшити кількість циклів очікування наступного вікна доступу. Оскільки більшість асемблерних інструкцій виконуються за 4 цикли, між суміжними Hub-інструкціями можуть бути виконані дві такі інструкції.

1.2.3 Реєстри спеціальних функцій та ОЗП

Як такий блок регістрів в ядрі мікроконтролера відсутня, події виконуються безпосередньо з осередками пам'яті. Існує лише 16 регістрів спеціальних функцій, що знаходяться в ОЗП кожного ядра, писання функцій кожного наведено в таблиці 1.1.

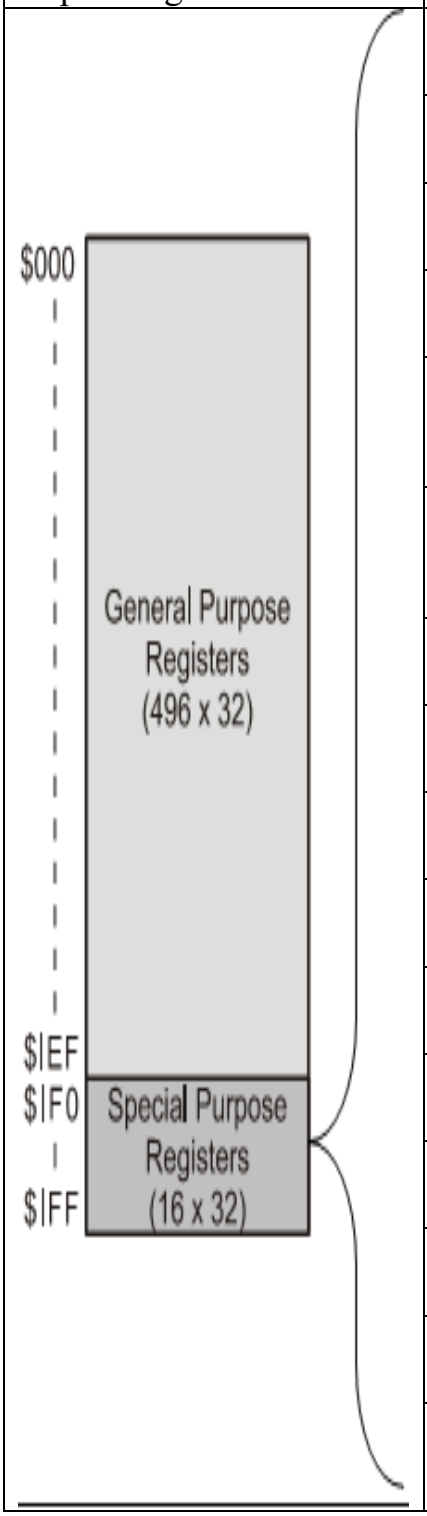
Будь-який з регістрів спеціальних функцій може бути доступний через:

– його фізичну адресу;

– його зумовлене ім'я;

– змінну типу масив з індексом від 0 до 15.

Таблиця 1.1 – Регістри спеціальних функцій мікроконтролера Propeller P8X32A

Карта Cog RAM	Адреса	Ім'я	Тип	Опис
	$1F0	PAR	Читання	Параметр завантаження
	$1F1	CNT	Читання	Системний Лічильник
	$1F2	INA	Читання	Стан входів P31 - P0
	$1F3	INB	Читання	Стан входов P63- P32
	$1F4	OUTA	Читання/Запис	Значення виходів P31 - P0
	$1F5	OUTB	Читання/Запис	Значен. виходів P63 – P32
	$1F6	DIRA	Читання/Запис	Напрямок P31 - P0
	$1F7	DIRB	Читання/Запис	Напрямок P63 - P32
	$1F8	CTRA	Читання/Запис	Управління лічильником A
	$1F9	CTRB	Читання/Запис	Управління лічильником B
	$1FA	FRQA	Читання/Запис	Частота лічильника A
	$1FB	FRQB	Читання/Запис	Частота лічильника B
	$1FC	PHSA	Читання/Запис	ФАПЧ лічильника A
	$1FD	PHSB	Читання/Запис	ФАПЧ лічильника B
	$1FE	VCFG	Читання/Запис	Налаштування відео
	$1FF	VSCL	Читання/Запис	Масштаб відео

Основна пам'ять у мікроконтролері Propeller – це блок пам'яті об'ємом 64кБ (16к х 32біта), який доступний усім обчислювальним ядрам, як взаємовиключний ресурс – через перемикач. Основна пам'ять складається з 32 кБ основного ОЗП (RAM) та 32 кБ основного ПЗП (ROM). Блок основного ОЗП є пам'яттю загального призначення, а також місцезнаходженням додатка, завантаженого з хоста, або завантаженого із зовнішнього 32 кБ ЕСППЗП. Блок основного ПЗП містить код і дані, дуже важливі для функціонування мікроконтролера Propeller, а сааме, набори символів, таблиці log, anti-log і sin, а також завантажувач і інтерпретатор мови Spin. Організація основної пам'яті показана на рисунку 1.5.

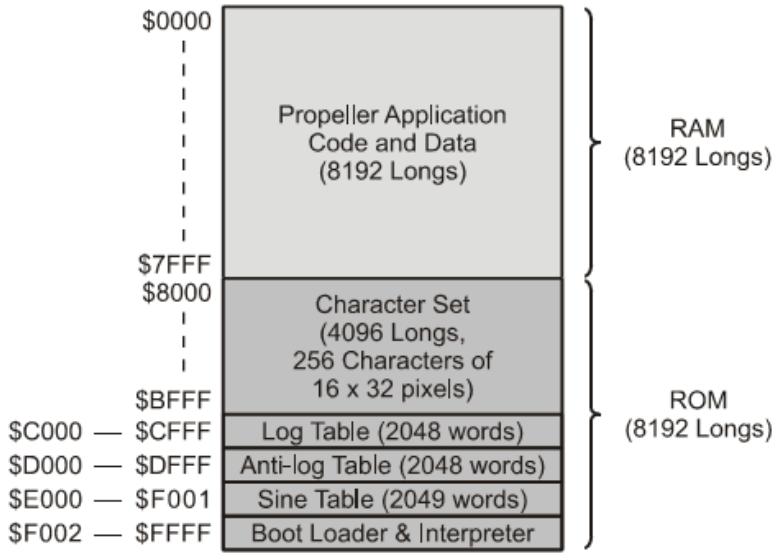


Рисунок 1.5 – Карта основної пам'яті мікроконтролера Propeller

Перша половина основної пам'яті – це ОЗП. Цей простір використовується для програм, даних, змінних та стека.

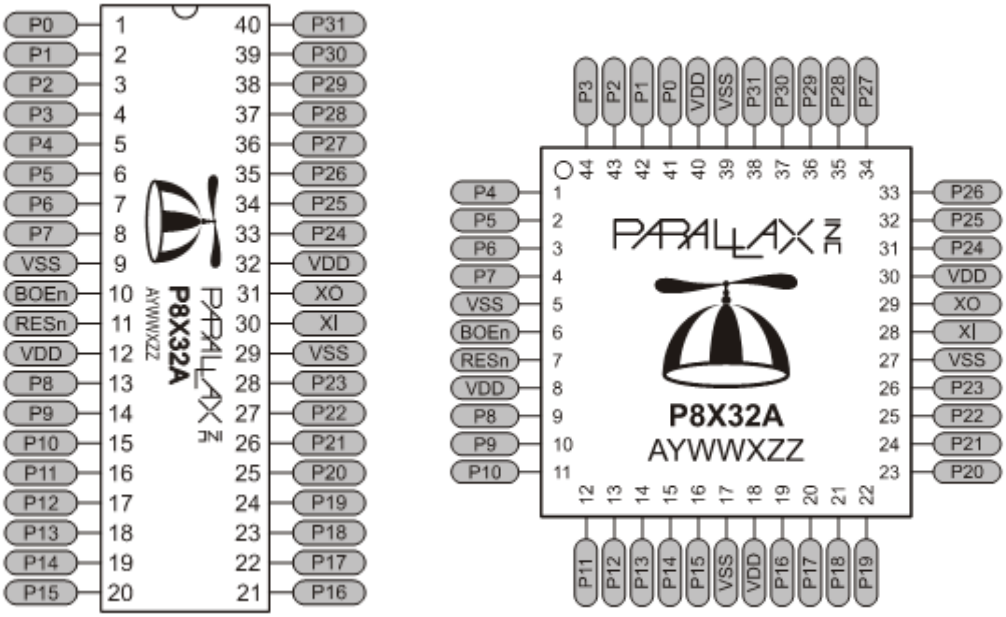
Перші 16 адрес, $0000 – $000F, містять ініціалізаційні дані, які використовуються завантажувачем та інтерпретатором. Користувальницька програма починається з адреси $0010, адресний простір після коду, що

здійснюється, що простягається до $7FFF, використовується для змінних і стека.

Друга половина основної пам'яті – це ПЗП, яка використовується для зберігання таблиці символів, математичних функцій, завантажувача та інтерпретатора мови.

1.2.4 Архітектура портів введення/виводу

Архітектура мікроконтролера Propeller P8X32A має 32 лінії [8]вводу/виводу, як показано на рисунку 1.6, 28 з яких повністю є портами загального призначення. Чотири лінії введення/виводу (28 - 31) мають спеціальне призначення при початковому завантаженні та доступні як порти загального призначення після неї.



а) DIP-корпус б) LQFP или QFN корпус

Рисунок 1.6 – Позначення портів вводу/виводу різної реалізації корпусів мікроконтролера Parallax

При включенні живлення або виникненні сигналу скидання, лініями P30 і P31 відбувається зв'язок з хостом для програмування, а P28 і P29

виробляється доступ до зовнішньої 32 кБ ЕСППЗУ. Після завантаження, будь-яка кількість ліній вводу/виводу може бути використана будь-яким Cog в будь-який момент часу, оскільки лінії вводу/виводу є одним із загальних ресурсів.

Необхідно стежити за тим, щоб під час виконання програми два ядра не використовували один і той же порт вводу/виводу для різних цілей.

Кожне ядро має свій власний 32-бітний регістр напряму та 32-бітовий регістр вихідних значень. Стан регістру напряму кожного ядра складається з логічної операції «ИЛИ» із значенням цього регістру у попереднього ядра. Таким же чином стан регістру вихідних значень для кожного ядра є логічним «АБО» з таким у попереднього ядра[9].

Необхідно відзначити, що значення вихідного регістру у кожного ядра виходять шляхом додавання по логічній функції «АБО» їх внутрішніх апаратних значень та подальшого множення по логічній функції «І» зі значеннями свого регістра напряму. В результаті напрям і стан виходу кожного порту вводу/виводу з'єднані «апаратним АБО» всієї групи ядер. Між ядрами немає електричних з'єднань, проте вони все ж таки можуть керувати портами вводу/виводу одночасно.

Результат такої структури з'єднань портів введення/виводу може бути легко описаний такими простими правилами:

– вивід є входом тільки якщо жоден з активних ядер не встановлює його як вихід;

– на виводі буде низький рівень тільки в тому випадку, коли всі активні ядра, які встановили його як вихід, встановлять його в нуль.

– на виводі буде високий рівень, якщо будь-який з активних ядер, що встановили його виходом, встановить його в одиницю.

Регістр напряму та регістр стану виходів будь-якого неактивного (відключеного) ядра встановлюються в нуль, таким чином виключаючи дане ядро з ланцюга впливу на стан портів вводу/висновки, які контролюються активними ядрами, що залишилися.

Кожне ядро має свій власний 32-бітний регістр введення. Цей вхідний регістр насправді є псевдореєстром; щоразу, під час читання з цього регістру, повертається реальний стан портів, незалежно від своїх напрями[10].

1.2.5 Аналіз мов програмування

Мікроконтролер Parallax Propeller P8X32A програмується за допомогою двох мов програмування[11]:

– мова Spin, об'єктно-орієнтована мова верхнього рівня;

– Propeller Асемблер, оптимізована мова нижнього рівня.

У мові асемблер є безліч апаратно-орієнтованих команд, які мають прямі еквіваленти у мові Spin[12]. Це дозволяє легко впоратися і з вивченням обох мов, і з використанням мікроконтролера Propeller взагалі[13].

Мова Spin компілюється програмним забезпеченням Propeller Tool в пакети даних (tokens), які обробляються в процесі виконання вбудованим в IMC Propeller інтерпретатором Spin. Вважається, що мова Spin добре пристосована для багатьох застосувань. За допомогою Spin можна вирішувати низькошвидкісні, високорівневі завдання та створити код для більш швидкісних застосувань, таких як асинхронний послідовний зв'язок до 19200 бод.

Мова Propeller Ассемблер асемблюється[14] програмою Propeller Tool у чистий машинний код і виконується у чистому вигляді по ходу роботи. На асемблері можна вирішувати високошвидкісні завдання[15] із застосуванням дуже невеликого обсягу коду[16].

Мова Spin компілюється не в асемблерний код, а в байт код[17], який виконується у віртуальній машині. Технологія схожа на Java та всякі C#. Віртуальна машина для виконання Spin-коду вбудована в ПЗП мікроконтролера, тобто при його використанні немає втрати швидкодії, на відміну від застосування мови CI та її подальшої компіляції в асемблер. Мова Spin є об'єктно-орієнтованою та служить базою для будь-якого Propeller

програми. Об'єкти в мікроконтролері Propeller складаються з коду Spin і коду Propeller ассемблера (рисунок 1.7).

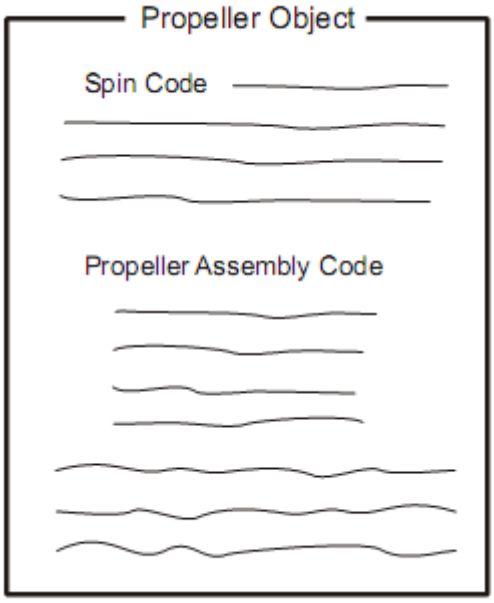


Рисунок 1.7 – Об'єкт у програмній реалізації мікроконтролера Propeller

Кожен об'єкт може розглядатися як будівельний блок програми. Об'єкт може використовувати один або більше об'єктів для побудови більш складних додатків.

При компіляції додатків створюється двійковий образ коду, що можна здійснити. Двійковий образ – це те, що завантажується в мікроконтролер Propeller. Відкомпільований код кожного з об'єктів, що використовуються додатком, додається в цей двійковий образ разом з областями змінних для кожного екземпляра кожного такого об'єкту. Під час виконання, програма може використовувати будь-який об'єкт будь-яку кількість часу; одні можуть використовуватися постійно, інші – за потребою, але всі вони займають незмінний обсяг пам'яті для свого коду та змінних.

Час життя об'єкта не змінюється, незалежно від цього, чи активно він використовується; він вимагає певну кількість пам'яті в двійковому образі докладання. Об'єкти створюються під час компіляції і ніколи не створюються

і не знищуються під час виконання, оскільки такі дії будуть фрагментувати пам'ять і призводити до невизначеностей у поведінці вбудованих систем реального часу.

Кожен об'єкт Propeller має внутрішню структуру, що включає до шести спеціальних блоків: CON, VAR, OBJ, PUB, PRI та DAT.

Розглянемо покажчики блоків об'єктів:

– CON – оголошує блок констант;

– VAR – оголошує блок змінних;

– OBJ – оголошує блок посилань на об'єкти;

– PUB – оголошує блок методів Public;

– PRI – оголошує блок методів Private;

– DAT – оголошує блок даних.

Остання секція, переважно ОЗП, містить програми завантажувача та інтерпретатора мови Spin.

Завантажувач відповідає за ініціалізацію мікроконтролера при включенні живлення або скидання. Коли розпочато процедуру завантаження, завантажувач копіюється в ОЗП Cog 0, і цей процесор виконує код, починаючи з адреси 0.

Програма завантажувача спочатку перевіряє лінії зв'язку з хостом та ЕСППЗП для завантаження коду/даних, потім відповідним чином обробляє отриману інформацію, і в кінці – або запускає програму інтерпретатора в ОЗП Cog 0 (перезаписуючи її поверх себе) для запуску програми користувача, або переводить мікроконтролер в режим зупинки.

Програма інтерпретатора Spin витягує та виконує Propeller-додаток користувача з основного ОЗП.

Залежно від коду програми це може призвести до запуску додаткових ядер для виконання додаткових частин Spin або асемблерного коду.

1.3 Розробка апаратних засобів відображення інформації

1.3.1 Принцип роботи генератора

Половина ПЗП мікроконтролера призначена для зберігання набору з 256 визначень символів. Кожне визначення символу має 16 пікселів завширшки і 32 пікселі заввишки. Ці визначення можуть використовуватися для відео дисплеїв, графічних РКІ, друку і т.д. Набір символів базується на Північно-Американському/Західно-Європейському кодуванні з великою кількістю додаткових спеціальних символів. Спеціальні символи призначені для малювання форм сигналів та електричних схем, відображення грецьких символів, що використовуються в електроніці, а також кількох стрілок та маркерів. Таблиця символів шрифту мікроконтролера Propeller представлена на рисунку 1.8.

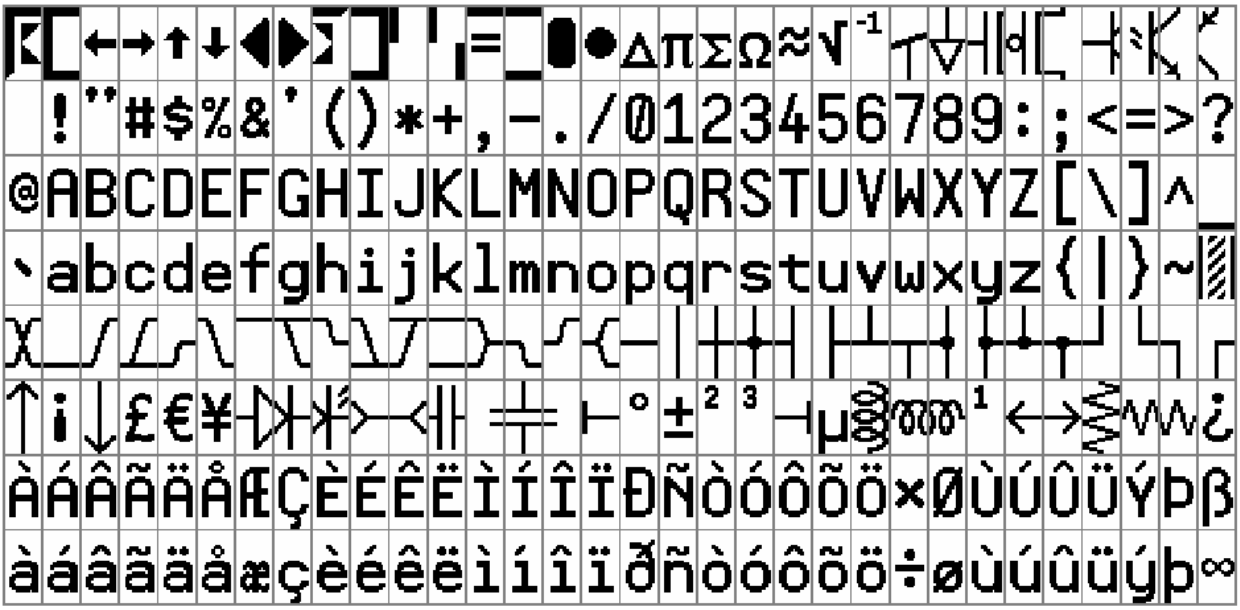


Рисунок 1.8 – Символи шрифту мікроконтролера Propeller

Перша пара символів розташована за адресою $8000-$807F. Друга пара займає байти $8080-$80FF, і так далі до останньої пари за адресою $BF80-$BFFF. Програма Propeller Tool включає інтерактивну таблицю символів, яка має переглядач ПЗП, що вказує, де і як розміщений у пам'яті кожен символ.

Пари символів злиті рядок до рядка таким чином, що кожні 16 горизонтальних пікселів одного символу перемежуються через один із пікселями свого сусіда так, що пікселі парного символу займають біти 0, 2, 4, ...30, а непарного - біти 1, 3, 5, ...31. Пікселі зліва займають молодші біти, праворуч – старші, як показано на рисунку 1.9.

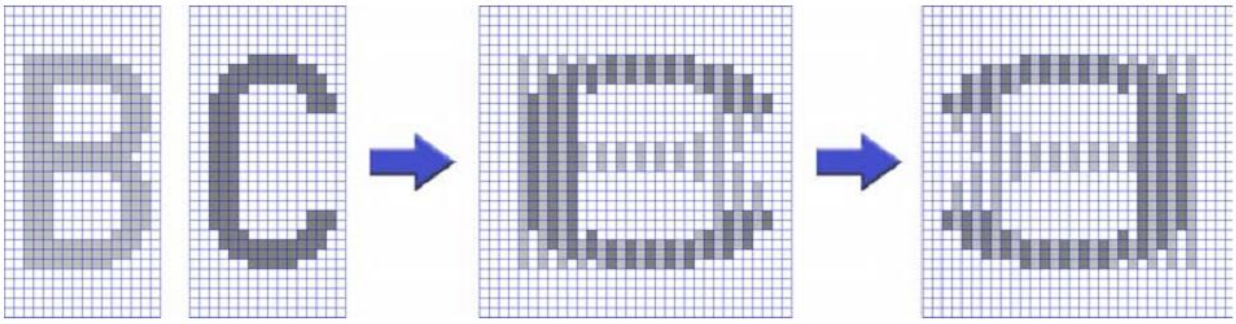


Рисунок 1.9 – Чергування символів у мікроконтролері Propeller

Такий формат виділяє 1 long (4 байти) на кожен рядок пікселів для кількох символів. У результаті 32 таких long-а, побудованих від верху символів до низу, забезпечують повне визначення злитої пари символів.

Визначення закодовані в такому вигляді для того, щоб апаратна відео-підсистема процесорів могла оперувати злитими long-ами безпосередньо, використовуючи вибір кольору для відображення парного або непарного символу.

Такий формат дає перевагу і для отримання «інтерактивних» пар символів, що є чотириколірними символами.

1.3.2 Організація генерація відеосигналу через VGA

Для створення відеосигналу через VGA необхідно підключити VGA-роз'єм (15-контактний субмініатюрний роз'єм для підключення аналогових моніторів за стандартом VGA) до мікроконтролера Propeller, як показано на рисунку 1.10.

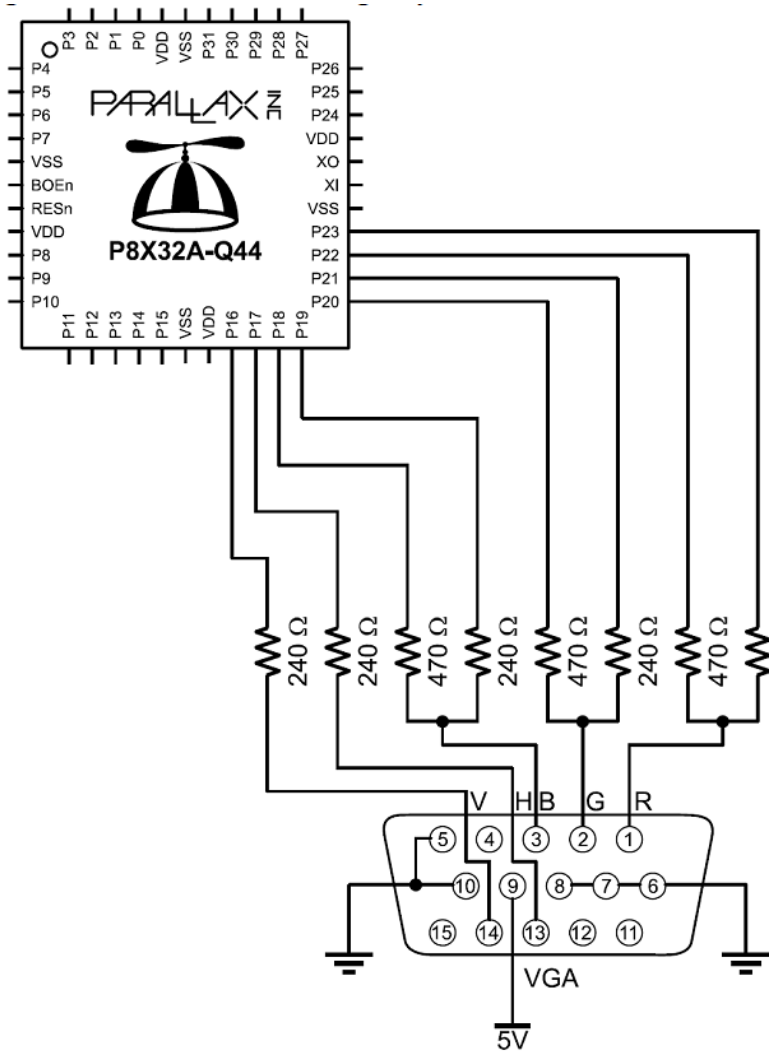


Рисунок 1.10 – Схема підключення VGA-роз'єму до мікроконтролера Propeller

Для генерації відеосигналу існують бібліотеки як низького: vga, vga512x384_Bitmap, так і рівня: vga_text і vga_hires_text, що забезпечують виведення текстової інформації.

Розглянемо два текстові модулі: один працює в невисокій роздільній здатності та споживає небагато ресурсів, а другий навпаки – у високому (1024*768) і для роботи вимагає набагато більше ресурсів.

Перетворення відео сигналу з VGA на HDMI в тому випадку, якщо необхідно використовувати цифровий інтерфейс можливо за допомогою спеціального перетворювача, рисунок 1.11.

Рисунок 1.11 – Зовнішній вигляд перетворювача VGA HDMI

Наприклад, розглянемо модуль vga_text:

```
CON
_clkmode = xtal1 + pll16x
_xinfreq = 5_000_000

OBJ
text : "vga_text"

PUB start | i
```

```
text.start(16)
i := 0
repeat
text.out($C) ' set color
text.out(i)
text.str(string("Test... "))
i++
if i > 32
i := 0
waitcnt(CNT + 80_000_000 / 3)
```

Приклад виводить на екран напис «Test...» різними кольорами тричі на секунду. Це забезпечується можливостями vga_text, що дозволяють задавати кольори, а також позиції виведення на екрані, очищати екран.

Необхідно оцінити ресурсоємність наведеного коду, для цього проводимо компіляцію, результат якої представлений на рисунку 1.12.

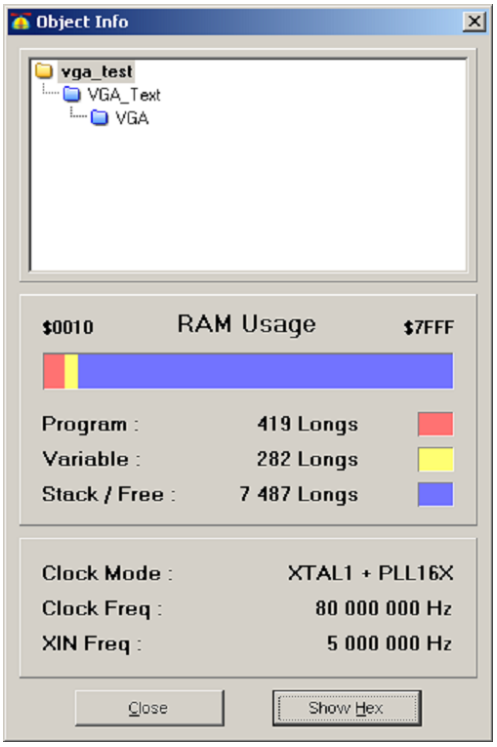


Рисунок 1.12 – Ресурсоємність модуля vga_text

Як можна помітити, використано (419+282)*4 =2084 байт загальної пам'яті. Для модуля, що генерує текстовий відеосигнал, це значення не велике. У той же час, розміри текстової області всього 40*15 символів, тому що використовується вбудований якісний шрифт з розміром символу 16*32. Якщо подивитися вихідний код модуля VGA, то видно, так само задіюється тільки одне додаткове ядро, яке і займається генерацією, основний код продовжує працювати на 1-му ядрі.

Це дає можливість згенерувати сигнал одночасно на 6-7 моніторів, але враховуючи кількість портів введення-виводу можна лише на 3.

Наведемо приклад генерації сигналу на кілька моніторів:

```
CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000
OBJ
  text  : "vga_text"  ' monitor 1
  text2 : "vga_text" ' monitor 2
PUB start | i
  text.start(16)
  text2.start(8)
  i := 0
  repeat
    text.out($C)  ' set color
    text.out(i)
    text.str(string("Test 1. "))
    text2.out($C) ' set color
    text2.out(32-i)
    text2.str(string("Test 2. "))
    i++
    if i > 32
      i := 0
    waitcnt(CNT + 80_000_000 / 3)
```

Як можна побачити, модуль vga_text імпортується двічі під різними іменами. Це дозволяє створити 2 його копії, що мають різні екземпляри змінних та буферів. Це дублює і код функцій. Якщо створювати копії на різних ядрах з окремими стеками, можна не дублювати код самих функцій.

1.3.3 Налаштування роботи відеогенератора

В архітектурі мікроконтролера Parallax Propeller P8X32A існує два регістри (VCFG і VSCL), які впливають на функціонування відеогенератора ядра.

Кожен із ядер має модуль відеогенератора, який спрощує передачу даних відеозображення на постійній швидкості. Регістр VCFG містить конфігураційні установки відеогенератора, зведено до таблиці 1.2.

Таблиця 1.2 – Структура регістру VCFG

Біти VCFG									
31	30..29	28	27	26	25..23	22..12	11..9	8	7..0
-	VMode	CMode	Chroma1	Chroma0	AuralSub	-	VGroup	-	VPins

Двобітове поле VMode (video mode, режим відео) вибирає тип та орієнтацію відео-виходу, якщо він використовується, зведено до таблиці 1.3.

Поле CMode (color mode, режим кольоровості) вибирає двоколірний чи чотириколірний режим. При CMode = 0 – двоколірний, дані пікселів – це 32 біти на 1 біт і можуть використовуватися лише колір 0 і 1. При CMode = 1 – чотириколірний режим, дані пікселів – це 16 біт на 2 біти та використовуються кольори від 0 до 3.

Біт Chroma1 (радіосигнал хромо) включає або вимикає хромо (колір) у радіочастотному сигналі. 0 = вимкнено, 1 = увімкнено.

Біт Chroma0 (відеосигнал хромо) включає або вимикає хромо (колір) у відеосигналі. 0 = вимкнено, 1 = увімкнено.

Таблиця 1.3 – Значення поля VMode

VMode	Відео-режим
00	Вимкнено, відеосигнал не генерується
01	VGA-режим; 8-бітовий паралельний висновок на лініях VPins7:0
10	Композитний режим 1; радіосигнал на VPins 7:4, відеосигнал на VPins 3:0
11	Композитний режим 2; відеосигнал на VPins 7:4, радіосигнал на VPins 3:0

Поле AuralSub (aural sub-carrier, звукова піднесуча) вибирає джерело для ЧС-модуляції звукової піднесе. Джерелом може бути PLLA одного ізядер, обраний відповідно до значення AuralSub (таблиця 1.4).

Таблиця 1.4 – Поле AuralSub

AuralSub	Джерело частоти піднесе
000	PLLA Cog 0
001	PLLA Cog 1
001	PLLA Cog 2
011	PLLA Cog 3
100	PLLA Cog 4
101	PLLA Cog 5
110	PLLA Cog 6
111	PLLA Cog 7

Поле VGroup (група ліній відеовиходу) вибирає, яка з груп по 8 ліній виводитиме відеосигнал (зведено до таблиці 1.5). Поле VPins (лінії відео-виходу) є маскою, що застосовується до ліній VGroup, яка вказує, на яких лініях виводити відео сигнал, як зведено в таблиці 1.6. Регістр VCFG може бути списан/записаний, як будь-які інші регістри або зумовлені змінні. Наприклад:

VCFG := %0_10_1_0_1_000_00000000000_001_0_00001111.

Таблиця 1.5 – Значення поля VGroup

Значення поля VGroup	Група ліній
000	Група 0: P7..P0
001	Група 1: P15..P8
010	Група 2: P23..P16
011	Група 3: P31..P24
100-111	< зарезервовано>

Таблиця 1.6 – Значення поля VPins

Значення поля VPins	Результат
00001111	Виведення відео на молодші 4 біти, композит
11110000	Виведення відео на старші 4 біти, композит
11111111	Виведення відео на всі 8 біт; режим VGA

Наведений вище програмний код встановлює регістр конфігурації відео для включення відео в композитному режимі 1 з 4 кольорами, радіосигнал хром (колір) включений, на группе ліній 1, нижні 4 лінії (тобто лінії P11:8).

Регістр масштабу відео. VSCL встановлює швидкість, на якій генеруються відео. Він повертає поточне значення регістру масштабу відео ядра, якщо використовується як змінна-джерело. Призначення конфігураційних установок регістру. VSCL наведено у таблиці 1.7.

Таблиця 1.7 – Призначення конфігураційних установок регістру VSCL

Біти VSCL		
31..20	19..12	11..0
-	PixelClocks	FrameClocks

Поле PixelClocks з 8 біт вказує кількість тактів на піксель: кількість тактів, яке має пройти перед тим, як кожен піксель висувається модулем відео-генератора. Ці такти – такти PLLA, а не системного генератора.

Поле FrameClocks з 12 бітів вказує кількість тактів на кадр: кількість тактів, яке має пройти перед тим, як новий кадр буде висунуто модулем відеогенератора. Ці такти – такти PLLA, а чи не системного генератора. Фрейм – це подвійне слово даних для пікселів. Оскільки дані пікселів можуть бути або 16 на 2 біт, або 32 на 1 біт (тобто відповідно 16 біт завширшки з 4 кольорами, або 32 пікселя завширшки з 2 квітами), то поле FrameClocks зазвичай у 16 або 32 рази більше за величину поля PixelClocks.

Поле VSCL може бути прочитано/записано як будь-які інші регістри або зумовлені змінні. Наприклад:

VSCL := %000000000000_10100000_101000000000

Цей код встановлює регістр масштабу відео значення 160 для поля PixelClocks і 2560 для поля FrameClocks (для 16-піксельного на 2 біти, кольорового кадру). Звичайно, реальна швидкість, на якій виводяться пікселі, залежить від частоти PLLA у комбінації з масштабним фактором.

1.3.4 Розробка програматора мікроконтролера Propeller

Для запуску програм достатньо лише сам чіп і послідовний порт (COM/UART/RS$232) з TTL рівнями на виході. Для цього можна використовувати перетворювач із MAX232 або конвертори USB-UART.

Повна схема підключення мікроконтролера Propeller для програмування за допомогою ПК представлена рисунку 1.13. На ній схематично показано підключення контролера Propeller у корпусі DIP-40. Кварцовий резонатор краще використовувати на 5МГц, що дасть змогу отримувати максимальну частоту тактування ядра 80МГц, але можна будь-

якої частоти 4-20МГц. Мікроконтролер Propeller має 2 внутрішні генератори, тому зовнішній кварцовий резонатор не завжди обов'язковий. На схемі також показана зовнішня EEPROM для зберігання конфігурації.

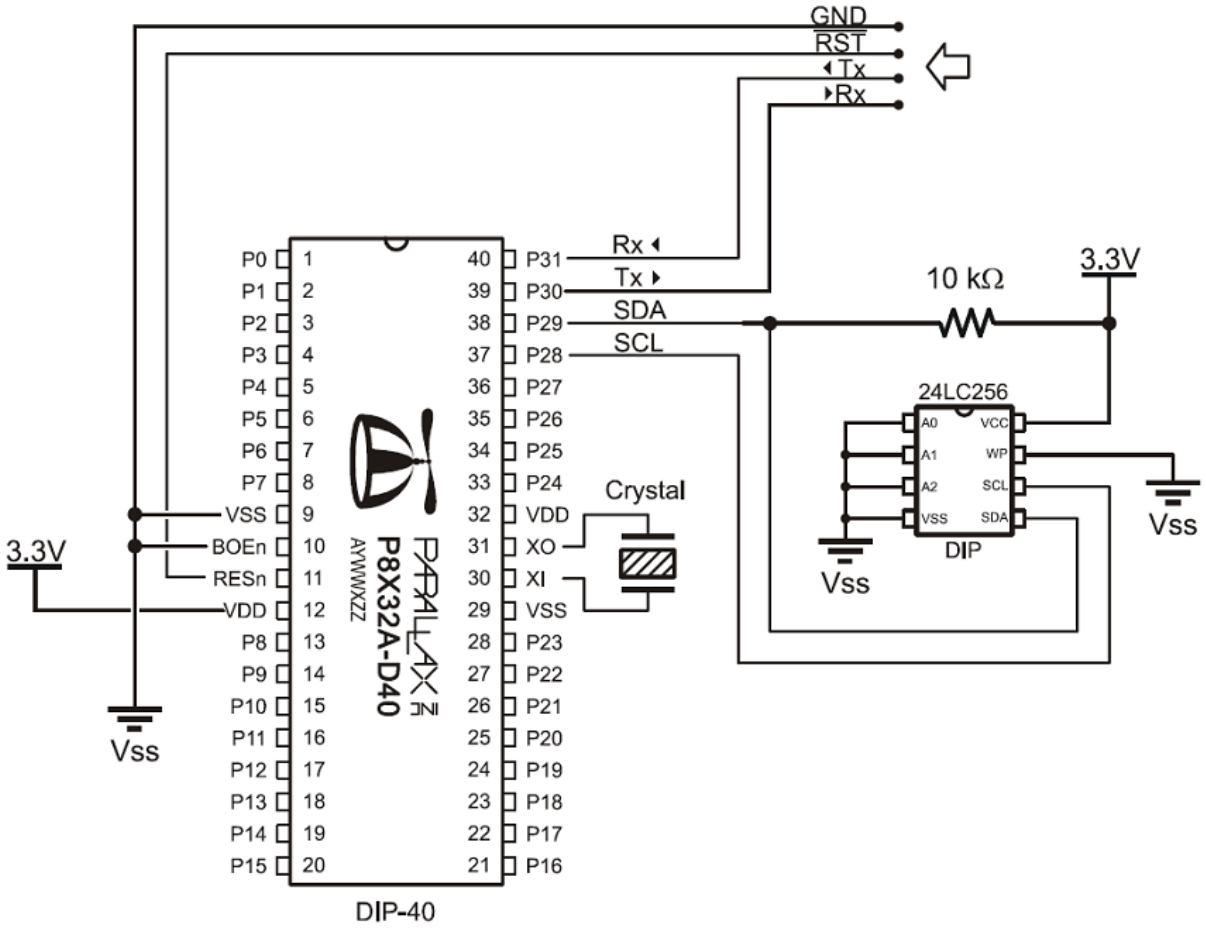


Рисунок 1.13 – Схема підключення мікроконтролера Propeller для програмування

Для програмування потрібно 3 лінії: прийом, передача та скидання. Перші 2 – це RXD, TXD, а 3я може бути як DTR, так і RST у ком-порті. (можна налаштувати серед Propeller Tool).

Основні операції в середовищі Propeller Tool:

– зібрати додаток та переглянути інформацію;

– завантажити додаток в ОЗП;

– завантажити додаток до зовнішнього EEPROM.

Єдиний істотний недолік програмного забезпечення Propeller Tool повне нерозуміння кирилиці.

Для підключення мікроконтролера через USB слід використовувати схему, наведену на рисунку 1.14, яка реалізовує віртуальний комп'ютер на основі мікросхеми FT232RQ.

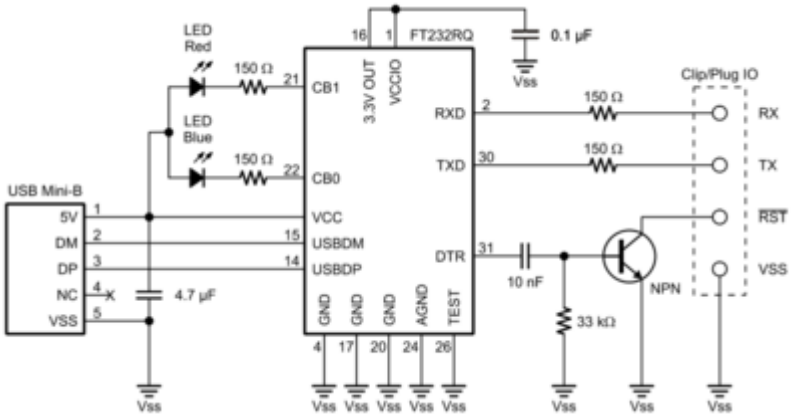


Рисунок 1.14 – Схема підключення мікроконтролера Propeller через USB

Мікросхема FTDI FT232R (FT232RL або FT232RQ) є високоінтегрованим перехідником USB – COM[17], що дозволяє використовуючи мінімум зовнішніх компонентів організувати послідовний обмін даними між зовнішнім пристроєм на мікроконтролері та комп'ютером через шину USB. Порівняно з попередніми версіями мікросхеми, у FT232R на кристал інтегровані тактовий генератор, енергонезалежна пам'ять EEPROM, частина зовнішніх пасивних компонентів.

Мікросхема може працювати в режимі послідовного обміну та в режимі bit-bang[18-19].

Підключення до послідовного порту ПК наведено на рисунку 1.15.

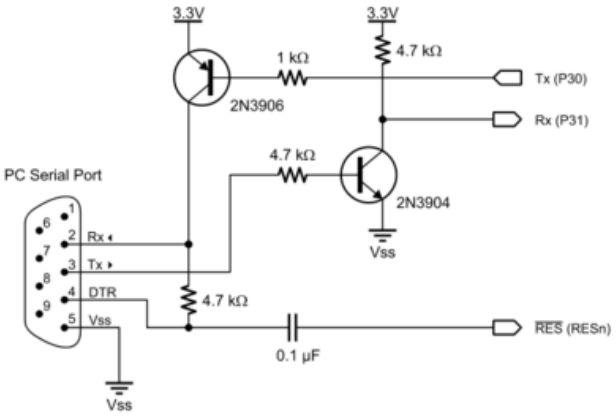


Рисунок 1.15 – Схема підключення до послідовного порту ПК

2 РОЗРОБКА АЛГОРИТМІЧНИХ І ПРОГРАМНИХ ЗАСОБІВ SPIN-БІБЛІОТЕКИ

Провівши аналіз апаратного пристрою мікроконтролера необхідно розпочати практичну розробку алгоритмічної та програмної структури SPIN-бібліотеки. Для цього знадобиться драйвер роботи з інтерфейсом VGA. Безперечною перевагою даного мікроконтролера є апаратна підтримка інтерфейсу VGA[20-21]. І драйвер для роботи з цим інтерфейсом вже є у програмному середовищі Propeller Tool. Програмний код даного драйвера наведено у додатку А.

2.1 Аналіз алгоритмів для реалізації SPIN-бібліотеки

Алгоритми машинної графіки можна розділити на два рівні: нижній та верхній[22-24]. Група алгоритмів нижнього рівня призначена для реалізації графічних примітивів (ліній, кіл, заповнень тощо). Ці алгоритми або подібні до них відтворені в графічних бібліотеках мов високого рівня (BGI в Турбо - Паскалі) або реалізовані апаратно в графічних процесорах робочих станцій (Silicon Graphics, Parallax Propeller та ін.).

Розглянемо групи алгоритмів нижнього рівня.

Найпростіші у сенсі використовуваних математичних методів і які відрізняються простотою реалізації. Як правило, такі алгоритми не є найкращими за обсягом обчислень, що виконуються, або необхідним ресурсам пам'яті.

Тому можна виділити другу групу алгоритмів, які використовують складніші математичні передумови (але часто й евристичні) і відрізняються більшою ефективністю.

До третьої групи слід віднести алгоритми, які можуть бути без великих труднощів реалізовані апаратно (допускають розпаралелювання, рекурсивні, що реалізуються в найпростіших командах). До цієї групи можуть потрапити і алгоритми, представлені у перших двох групах.

Нарешті, до четвертої групи можна зарахувати алгоритми зі спеціальним призначенням (наприклад, усунення сходового ефекту).

До алгоритмів верхнього рівня належать насамперед алгоритми видалення невидимих ліній та поверхонь. Завдання видалення невидимих ліній та поверхонь продовжує залишатися центральним у машинній графіці. Від ефективності алгоритмів, що дозволяють вирішити це завдання, залежать якість та швидкість побудови тривимірного зображення.

До завдання видалення невидимих ліній та поверхонь примикає завдання побудови (зафарбовування) напівтонових (реалістичних) зображень, тобто. обліку явищ, пов'язаних з кількістю та характером джерел світла, обліку властивостей поверхні тіла (прозорість, заломлення, відображення світла).

Однак при цьому не слід забувати, що виведення об'єктів у алгоритмах верхнього рівня забезпечується примітивами, що реалізують алгоритми нижнього рівня, тому не можна ігнорувати проблему вибору та розробки ефективних алгоритмів нижнього рівня.

Для різних галузей застосування машинної графіки першому плані можуть висуватися різні властивості алгоритмів. Для наукової графіки велике значення має універсальність алгоритму, швидкодія може відходити другого план. Для систем моделювання, що відтворюють об'єкти, що рухаються, швидкодія стає головним критерієм, оскільки потрібно генерувати зображення практично в реальному масштабі часу.

Особливості растрової графіки пов'язані з тим, що звичайні зображення, з якими стикається людина у своїй діяльності (креслення, графіки, карти, художні картини тощо), реалізовані на площині, що складається з нескінченного набору точок. Екран же растрового дисплея є матрицею дискретних елементів, що мають конкретні фізичні розміри. При цьому кількість їх суттєво обмежена. Тому не можна провести точну лінію з однієї точки до іншої, а можна виконати тільки апроксимацію цієї лінії з відображенням її на дискретній матриці (площині). Таку площину також

називають цілою решіткою, растрової площиною або растром. Ця решітка є квадратною сіткою з кроком 1. Відображення будь-якого об'єкта на цілу решітку називається розкладанням його в растр або просто растровим уявленням.

2.1.1 Побудова ліній, кіл, еліпсів

Найпростіше накреслити лінію можна за допомогою рівняння y=ax+b. При цьому результати треба заокруглювати до цілих, тому пряма буде нерівна.

Якщо навпаки, потрібно з'єднати 2 точки із заданими координатами (x1, y1), (x2, y2), то:

$$a = (y2-y1) / (x2-x1);\ \ b = y1 - a*x1. \qquad (2.1)$$

Рівняння кола виглядає так:

$$x = xc + r*\cos(a);\ \ y = yc + r*\sin(a),$$

де (xc, yc) – координати центру кола,

r – радіус,

a – кут для поточної точки (x, y).

Можна будувати коло прямо за цим рівнянням, задавши певний крок по кутку (a = 0.360 з кроком DA). Однак, якщо крок буде занадто малий, коло за рахунок округлення координат буде нерівне і деякі точки висвічуватимуться по кілька разів. Зазвичай крок по куту вибирається рівним 1/r радіан, (найчастіше крок зміни кута має бути змінним для того, щоб уникнути розривів або відсутності зміни координат).

Можна здійснити простий алгоритм апроксимації відрізками.

Наприклад, координати відрізків 6 виходять з кроком 60 градусів, потім вони з'єднуються прямими лініями.

Для швидкої побудови використовується симетрія кола (обчислюються координати точок лише 1/8 частини кола – для сегмента від 0 до 45 градусів). Крім того, можна уникнути операцій синуса/косинусу, якщо виразити координати наступної точки кола з попередньої (рекурентна формула):

$$x2 = xc + (x1-xc) * CA + (y1-yc) * SA; \quad (2.2)$$
$$y2 = yc + (y1-yc) * CA - (x1-xc) * SA,$$

де SA = sin(DA),

CA = cos(DA).

Початкова точка для рекурентної формули: x1 = xc, y1 = yc+r.

При побудові кіл слід враховувати, що розміри пікселів по вертикалі та горизонталі не збігаються (крім VGA 640x480) і кола будуть витягуватися в еліпси. Щоб уникнути цього, потрібно вводити по вирівнюючі коефіцієнти (які можна визначити за GetAspectRatio). Самі ж еліпси описуються ураженнями:

$$x = xc + rx * \cos(a); \; y = yc + ry * \sin(a), \quad (2.3)$$

де rx, ry – напіврадіуси.

Загальні принципи побудови їм самі, що й кіл.

2.1.2 Алгоритм Брезенхема

Алгоритм Брезенхема (Bresenham) був розроблений в 1965 для цифрових графопобудівників, а потім став використовуватися для растрових дисплеїв.

Ідея алгоритму полягає в тому, що одна координата змінюється на одиницю, а інша або не змінюється, або змінюється на одиницю залежно від

розташування відповідної точки від найближчого вузла координатної сітки. Відстань від точки відрізка до найближчого вузла по відповідній ортогональній координаті називається помилкою. Алгоритм організований в такий спосіб, що з обчислення другої координати потрібно лише визначати знак цієї помилки. Величину помилки Delta можна визначити відповідно до наступного виразу:

$$Delta = Yуз - Yот \ , \qquad (2.4)$$

де Yуз – координата Y найближчого вузла при X = 1,

Yот – координата Y відрізка при X = 1.

Якщо при X = 1 координата Y точки відрізка дорівнює 1/2, то вузли координатної сітки (1,0) і (1,1) знаходяться на однаковій відстані від відрізка і як "найближчий" вибирається вузол (1,1). Отже, якщо Yот >= 1/2, то Delta >=0, інакше, при Yот < 1/2, Delta < 0. Для організації обчислень зручніше користуватися величиною Delta, а інший, визначається як величина d = Delta - 1/2. При зміні координати X на 1 величина d змінюється значення кутового коефіцієнта: d = d + dY / dX. На кожному кроці алгоритму провадиться обчислення координати X, величини d і виконується аналіз знака d. При цьому, якщо виявиться, що d> = 0, то здійснюється збільшення Y на 1, а значення d коригується шляхом віднімання з неї 1.

З урахуванням викладеного алгоритм апроксимації відрізка у першому октанті можна подати у такому вигляді:

```
X := X1; Y : =  Y1;
dX := X2 - X1;
dY := Y2 - Y1;
d : = - 1/2;
while X =< X2 do
PutPixel (X, Y);
X := X + 1;
d := d + dY  / dX;
if d >= 0 then
```

```
begin
Y := Y +  1;
d := d -  1
end
end while.
```

Представлений алгоритм використовує речові числа та операцію поділу. Обидва недоліки можна виключити заміною величини d на іншу, що дорівнює D = 2 * dX * d. Відповідно до цього арифметичні вирази, в яких бере участь d, модифікуються шляхом множення обох частин на величину 2 * dX.

Тоді алгоритм набуває вигляду:

```
X := X1; Y := Y1;
dX := X2 - X1;
dY := Y2 - Y1;
D := - dX;
while X =< X2 do
PutPixel (X, Y);
X := X + 1;
D := D + 2  * dY;
if D >= 0 then
begin
Y := Y +  1;
D := D -  2 * dX;
end
end while.
```

Останній варіант алгоритму можна покращити, якщо операцію множення на 2 замінити операцією зсуву вліво на один розряд. Крім того, якщо обчислення 2 * dX і 2 * dY виконати перед циклом, то операцію зсуву потрібно виконати лише двічі.

Тоді алгоритм набуває остаточного вигляду:

```
X := X1; Y := Y1;
dX := X2 - X1;
dY := Y2 - Y1;
D := - dX;
```

```
Dx := dX shl 1;
Dy := dY shl 1;
while X =< X2 do
PutPixel (X, Y);
X := X + 1;
D := D + Dy;
if D >= 0 then
begin
Y := Y +  1;
D := D -  Dx
end
end while.
```

Оцінюючи переваги отриманого алгоритму, можна відзначити, що він виконує оптимальну апроксимацію відрізку, використовуючи при цьому цілочисленну арифметику та мінімальну кількість операцій складання та віднімання. Крім того, алгоритм дозволяє використовувати його та в інших октантах площини.

У алгоритмі Брезенхема для всі октантів лінія розглядається як набір сегментів двох типів: тих, які розташовані діагонально і тих, які розташовані горизонтально чи вертикально. Для ліній з нахилом більше 1 прямі сегменти вертикальні <= 1 – горизонтальні. Перше завдання алгоритму полягає у обчисленні нахилу. Потім обчислюється вирівнюючий фактор, який стежить, щоб деяке число прямих сегментів мало більшу довжину, ніж інші. У циклі по більшому відрізку (по x або y) BX по черзі приймає то позитивні, то негативні значення, відзначаючи який тип сегмента виводиться – діагональний або прямий.

В алгоритмі проводиться лінія від (x1, y1) до (x2, y2). Використовуються деякі регістри та змінні, призначення яких коментується.
Початок алгоритму.
CX=1, DX=1 – початкові інкременти X і Y
{обчислення вертикальної дистанції}
DI=y2-y1 > 0? Якщо так, то на крок 4, інакше – на слід.

DX=-1, DI=-DI (дистанція Y повинна бути > 0)

DIAG_Y=DX – збільшення Y для діагональних ел-тов

{обчислення горизонтальної дистанції}

SI = x2-x1 > 0? Якщо так, то на крок 7, інакше – на слід.

CX=-1, SI=-SI (дистанція X повинна бути > 0)

DIAG_X=CX - збільшення по X для діагональних елементів

{ вертикальні або горизонтальні прямі сегменти }

SI >= DI? Якщо так, то на крок 9, інакше крок 10.

DX=0 – для прямих сегментів Y не змінюється (тобто вони горизонтальні).

Перехід на крок 11.

CX=0 – прямі сегменти вертикальні.

SI <-> DI – обмін SWAP, щоб більший відрізок був у SI.

{обчислення фактора, що вирівнює}

SHORT=DI – коротший відрізок

STR_X=CX, STR_Y=DX – інкременти для прямих сегментів (один з них = 0)

STR_COUNT = 2*SHORT

DIAG_COUNT= 2*SHORT – 2*SI

BX=2*SHORT-SI – початкове значення перемикача

{підготовка до виведення лінії}

CX = x1, DX = y1 – початкові координати

SI = SI + 1 – додати один піксель для кінця відрізка

{ виведення лінії }

SI=SI-1

SI = 0? Якщо так, то кінець алгоритму – крок 19.

Виведення точки з координатами CX, DX через BIOS чи ПДП.

BX < 0? Якщо так, то крок 17, інакше 18.

{наступний сегмент прямий – підготовка координат для виведення}

CX=CX+STR_X

DX=DX+STR_Y

BX=BX+STR_COUNT

перехід на крок 13

{Слід. сегмент діагональний }

CX=CX+DIAG_X

DX=DX+DIAG_Y

BX=BX+DIAG_COUNT

перехід на крок 13

Кінець алгоритму.

2.2 Розробка програмної моделі SPIN-бібліотеки

Надалі необхідно розпочати розробку безпосередньо програмної частини бібліотеки. Виходячи із завдання, основними функціями бібліотеки має бути виведення графічної та текстової інформації на екран. Для цього необхідно визначиться з дозволом інформації, що виводиться. Виходячи з того, що застосування даного мікроконтролера оптимально у вбудовуваних системах і невеликими екранами, оптимальним буде роздільна здатність 640*480 пікселів. Таким чином, ми матимемо 40 колонок та 30 рядків, що складаються з блоків 16*16 пікселів.

Створено бібліотеку для виведення на дисплей найпростіших графічних примітивів, таких як: текст, точки, лінії, прямокутники. Ця бібліотека називається "VGA_Draw.spin". Кожному з примітивів можна встановити потрібний колір і колір фону. Надалі будемо послідовно розглядати та описувати роботу функцій, що входять до цієї бібліотеки. Повний програмний код цієї бібліотеки наведено у додатку Б.

2.2.1 Завдання констант, змінних та підключення бібліотек

По-перше, що потрібно було зробити, це встановити константи, які будуть використовуватися в програмі. В мові SPIN для цього є особливий

блок CON. Необхідні константи це: частота роботи мікроконтролера (80 МГц), кількість стовпців і колонок, розміри дисплея (з урахуванням кількості стовпців і колонок). Надалі представлено програмний код з визначення констант:

```
CON

                        _clkmode = xtal1+pll16x
                         _clkfreq = 80_000_000

                            vga_params = 21
                               cols = 32
                               rows = 16
                      screensize = cols * rows
```

Наступне, що потрібно було зробити, це задати необхідні змінні. Для змінних існує спеціальний блок коду, який називається VAR. Програмний код оголошення змінних:

```
VAR

  long  vga_status      'status: off/visible/invisible  read-only
(21 contiguous longs)
  long  vga_enable      'enable: off/on                 write-only
  long  vga_pins        'pins: byte(2),topbit(3)        write-only
  long  vga_mode        'mode: interlace,hpol,vpol      write-only
  long  vga_videobase   'video base @word               write-only
  long  vga_colorbase   'color base @long               write-only
  long  vga_hc          'horizontal cells               write-only
  long  vga_vc          'vertical cells                 write-only
  long  vga_hx          'horizontal cell expansion      write-only
  long  vga_vx          'vertical cell expansion        write-only
  long  vga_ho          'horizontal offset              write-only
  long  vga_vo          'vertical offset                write-only
  long  vga_hd          'horizontal display pixels      write-only
  long  vga_hf          'horizontal front-porch pixels  write-only
  long  vga_hs          'horizontal sync pixels         write-only
  long  vga_hb          'horizontal back-porch pixels   write-only
  long  vga_vd          'vertical display lines         write-only
  long  vga_vf          'vertical front-porch lines     write-only
```

```
  long  vga_vs          'vertical sync lines           write-only
  long  vga_vb          'vertical back-porch lines     write-only
  long  vga_rate        'pixel rate (Hz)               write-only

  word  screen[screensize]

  long  col, row, color
  long  boxcolor,ptr
  long  stack[100]
```

Підключення драйвера для роботи з інтерфейсом VGA. Для під'єднання бібліотек використовується блок OBJ. Код підключення бібліотеки:

```
OBJ

  vga : "vga"
```

2.2.2 Ініціалізація дисплея та зупинка роботи

Для реалізації функцій існують два блоки: PUB та PRI. Ідентифікатор PUB служить для оголошення методу Public. Метод Public – це секція коду, що виконує певну функцію і повертає значення результату. Public методи можуть бути доступні ззовні об'єкту та забезпечують інтерфейс з об'єктом. Метод Private у всьому схожий на метод Public, не зважаючи на те, що вони оголошені як PRI і не доступні ззовні об'єкту.

Першою та найважливішою функцією бібліотеки є функція ініціалізації дисплея. Вона викликається один раз на початку основної програми. Функція ініціалізує ядро, з яким працюватиме бібліотека (дана бібліотека задіяє лише одне ядро мікроконтролера). Програмний код цієї функції мовою SPIN:

```
PUB start(pins)

  print($100)
  longmove(@vga_status, @vgaparams, vga_params)
  vga_pins := pins
  vga_videobase := @screen
  vga_colorbase := @vgacolors
```

```
  result := vga.start(@vga_status)
```

Для випадків, коли необхідно повністю зупинити роботу з дисплеєм, реалізовано функцію зупинки. Вона просто відключає ядро, з яким працює ця бібліотека. Програмний код функції зупинки:

```
PUB stop

'' Stop VGA driver - frees a cog

  if cog
    cogstop(cog~ - 1)
```

2.1.3 Функція виведення текстової інформації

Була реалізована функція щодо виведення текстової інформації. Функція має назву print(c). За допомогою цієї функції відбувається не лише виведення безпосередньо тексту, а і завдання кольору для подальшої інформації. Недоліком цієї реалізації, яку, на жаль, не вдалося виправити, є те, що текст не можна виводити рядками, а лише за одним символом. Обійти це обмеження можна за допомогою створення масивів текстової інформації, яка в циклі по одному символу буде виводитися на екран. Символи беруться із таблиці знакогенератора, зашитої в пам’ять мікроконтролеру. Їх можна програмно задавати як безпосередньо символами, наприклад, `"H", "j", "•", "∑", "┐"`, так і за їх адресами у просторі знакогенератору `$20, $fa, $44, $1b` и т.д.

Також, за допомогою функції print(c) можна очистити екран, перейти на новий рядок, стерти попередній символ. Програмний код, що реалізує цю функцію:

```
PUB print(c) | i, k
  case c
    $00..$FF:           'character?
      k := color << 1 + c & 1
      i := k << 10 + $200 + c & $FE
      screen[row * cols + col] := i
      screen[(row + 1) * cols + col] := i | 1
```

```
      if ++col == cols
        newline

    $100:               'clear screen?
      wordfill(@screen, $200, screensize)
      col := row := 0

    $108:               'backspace?
      if col
        col--

    $10D:               'return?
      newline

    $110..$11F:         'select color?
      color := c & $F


' New line
PRI newline : i
  col := 0
  if (row += 2) == rows
    row -= 2
    'scroll lines
    repeat i from 0 to rows-3
'      wordmove(@screen[i*cols], @screen[(i+2)*cols], cols)
    'clear new line
'    wordfill(@screen[(rows-2)*cols], $200, cols<<1)
```

Далі докладно опишемо принцип роботи та застосування цієї функції. Як видно з коду, то залежно від того, який параметр передається у функцію, її робота відрізнятиметься.

Якщо функція отримала число в проміжку від $00 до $FF, на екран буде виведено символ, відповідний цій адресі таблиці знакогенератора.

Якщо функція отримала $100, то буде очищено екран.

Якщо функція отримала $108, буде видалено попередній виділений символ.

Якщо функція отримала значення $10D, то буде зміщення на один рядок вниз. Формальний аналог Enter.

Якщо функція отримала $110..$11F, буде визначено колір тексту, що вводиться після виклику цієї функції. В бібліотеку у поточній редакції «зашито» вісім комбінацій кольору для тексту:

– $110 – червоний текст на чорному фону;

– $111 – синій текст на сірому фону;

– $112 – темно-зелений текст на чорному фону;

– $113 – білий текст на чорному фону;

– $114 – білий текст на червоному фону;

– $115 – білий текст на зеленому фону;

– $116 – білий текст на блакитному фону;

– $117 – чорний текст на сірому фону.

Таким чином, щоб написати слово «Hello» білим кольором на чорному тлі, програмний код буде виглядати наступним чином:

```
Print($113)
Print (string("H","e","l","l","o"))
```

А щоб після цього написати фразу my name is Propeller темно-зеленим кольором на чорному тлі і з нового рядка, то необхідно вже до написаного коду додати:

```
Print($10D)
Print($112)
Print(string("m","y",$20,"n","a","m","e",",$20,"I",
 "s",$20,"P","r","o","p","e","l","l","e","r"))
```

Далі представлена функція виводу шістнадцяткового числа hex(value, digits). Ця функція зручна під час візуалізації лічильників. Параметр value – це саме шістнадцяткове число, а параметр digits – це кількість розрядів числа. Програмний код цієї функції:

```
PUB hex(value, digits)

'' Print a hexadecimal number

  value <<= (8 - digits) << 2
  repeat digits
```

```
    out(lookupz((value <-= 4) & $F : "0".."9", "A".."F"))
```

Функція виведення двійкового числа bin(value, digits). Принцип роботи цієї функції повністю відповідає попередньої функції, крім того, ще число перетворюється не в шістнадцяткову форму уявлення, а двійкову. Програмний код функції:

```
PUB bin(value, digits)

'' Print a binary number

  value <<= 32 - digits
  repeat digits
    out((value <-= 1) & 1 + "0")
```

Функція виведення рядка називається str(stringptr). Принцип її роботи полягає в тому, що вона необхідна кількість разів (рівне кількості переданих у неї параметрів або символів) викликає функцію print(c) і створена за великим рахунком лише для полегшення читабельності програмного коду. Безпосередньо сама функція:

```
PUB str(stringptr)

'' Print a zero-terminated string

  repeat strsize(stringptr)
    out(byte[stringptr++])
```

2.2.4 Функція реалізації графічних примітивів

Вивід простої текстової інформації, безсумнівно, є важливою частиною інформаційного процесу. Але, все ж таки, необхідно наявність можливості створення хоча б найпростіших графічних примітивів, таких як, коло, лінія, точка, прямокутник, багатокутник і т.д. Для цього було програмно реалізовано кілька функцій, які відповідають цим вимогам. Перша – це

виведення на дисплей одного пікселя. Функція називається plotPixel(pixelValue, xPixel, yPixel, displayBase). Програмний код:

```
      PUB plotPixel(pixelValue, xPixel, yPixel, displayBase) '' 7 Stack Longs

      ''
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////
      '' // Plots a 1x1 pixel on screen.
      '' //
      '' // PixelValue - The pixel to plot on screen. Between %%0 and %%1 or
between %%0, %%1, %%2, and %%3 depending on color mode.
      '' // XPixel - The X cartesian pixel coordinate, will be forced to be
multiple of 1. Y will be forced to be a multiple of 1.
      '' // YPixel - The Y cartesian pixel coordinate. Note that this axis is
inverted like on all other graphics drivers.
      '' // DisplayBase - The address of the display buffer to draw to.
      ''
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////

        xPixel := ((xPixel <# (horizontalPixels - 1)) #> 0)
        displayBase += (((horizontalLongs * ((yPixel <# (verticalPixels - 1))
#> 0)) + (xPixel >> (5 - bitsPerPixel))) << 2)


        xPixel := ((xPixel & ($1F >> bitsPerPixel)) << bitsPerPixel)
        yPixel := (!((1 + (bitsPerPixel << 1)) << xPixel))


        long[displayBase] := ((long[displayBase] & yPixel) | (((pixelValue <#
(1 + (bitsPerPixel << 1))) #> 0) << xPixel))
```

Аргументами служать:

– `PixelValue` – буквально означає значення пікселя. За фактом це колір виведеного пікселя;

– `XPixel` – Координата пікселя по осі X;

– `YPixel` – Координата пікселя по осі Y.

Потенціал використання цієї функції величезний. За допомогою неї можна будувати графіки, що динамічно змінюються.

Функція box(left,top,width,height) здатна виводити на екран прямокутник. За її допомогою можна продавати імпровізовані кнопки, таблиці і т.д. Код функції:

```
PUB box(left,top,width,height) | x, y, i


  ptr := top * cols + left
```

```
  boxchr($0)
  repeat i from 1 to width
    boxchr($C)
  boxchr($8)
  repeat i from 1 to height
    ptr := (top + i) * cols + left
    boxchr($A)
    ptr += width
    boxchr($B)
  ptr := (top + height + 1) * cols + left
  boxchr($1)
  repeat i from 1 to width
    boxchr($D)
  boxchr($9)

PRI boxchr(c): i

  screen[ptr++] := boxcolor << 10 + $200 + c
```

Аргументами є:

- `left` – задає колонку, в якій перебуватиме верхній лівий кут прямокутника;
- `top` – задає рядок, у якому буде верхній лівий кут прямокутника;
- `width` – задає ширину прямокутника;
- `height` – задає висоту прямокутника.

2.2.5 Блок даних

Останнім блоком у програмі є блок визначення даних. Там зберігаються усі дані, зарезервовані для використання пам'яті, а також код мови Propeller асемблер. У цьому випадку зберігаються дані для налаштування дисплея, таблиця колірних кодів і таблиці символьних комбінацій для виведення на екран. Програмний код блоку:

```
DAT
```

```
vgaparams               long    0               'status
                        long    1               'enable
                        long    %010_111        'pins
                        long    %011            'mode
                        long    0               'videobase
                        long    0               'colorbase
                        long    cols            'hc
                        long    rows            'vc
                        long    1               'hx
                        long    1               'vx
                        long    0               'ho
                        long    0               'vo
                        long    512             'hd
                        long    16              'hf
                        long    96              'hs
                        long    48              'hb
                        long    380             'vd
                        long    11              'vf
                        long    2               'vs
                        long    31              'vb
                        long    20_000_000      'rate

vgacolors               long
                        long    $C000C000       'red
                        long    $C0C00000
                        long    $08A808A8       'green
                        long    $0808A8A8
                        long    $50005000       'blue
                        long    $50500000
                        long    $FC00FC00       'white
                        long    $FCFC0000
                        long    $FF80FF80       'red/white
                        long    $FFFF8080
                        long    $FF20FF20       'green/white
                        long    $FFFF2020
                        long    $FF28FF28       'cyan/white
                        long    $FFFF2828
                        long    $00A800A8       'grey/black
                        long    $0000A8A8
                        long    $C0408080       'redbox
spcl                    long    $30100020       'greenbox
                        long    $3C142828       'cyanbox
                        long    $FC54A8A8       'greybox
                        long                      $3C14FF28
'cyanbox+underscore
                        long    0
```

3 ТЕОРІЯ АВТОМАТИЧНОГО УПРАВЛІННЯ

Будь-яка цілеспрямована діяльність або процес мають потребу в керуванні. Якщо керування здійснюється технічними засобами без участі людини (або іншого живого організму) – це автоматичне керування.

Системи автоматичного управління (САУ) призначені для керування технічними процесами без безпосереднього втручання або участі людини.

У складі САУ розрізняють власне об'єкт керування й керуючі пристрої. Дуже часто цей розподіл частково або повністю умовно і в основному базується на зручності аналізу системи, а не на фізичному поділі об'єкта й пристрою керування. Об'єкт і керуючий пристрій зв'язані через виконавчі механізми, за допомогою яких на об'єкт передаються керуючі впливи, і через вимірювальну апаратуру, від якої керуючі пристрої одержують сигнали про стан об'єкта. Технічно в САУ входить також апаратура і лінії зв'язку між вище зазначеними елементами, і їхнім можливим впливом на сигнали в системі не можна нехтувати.

При побудові та аналізу САУ приділяється значна увага саме інформаційним потокам та зв'язкам, а також тим перетворенням (і спотворенням) сигналів, які відбуваються в об'єкті, в керуючому пристрої, в зв'язковій апаратурі. Сукупність правил і процедур, по яких керуючий пристрій обробляє інформацію для вироблення керуючих впливів, називається алгоритмом функціонування САУ.

3.1 Призначення САУ

Система автоматичного управління (САУ) підтримує або поліпшує функціонування керованого об'єкта. У ряді випадків допоміжні для САУ операції (пуск, зупинка, контроль, налагодження) також можуть бути автоматизовані. САУ функціонує в основному в складі виробничого чи якогось іншого комплексу.

Структури управління поділяють на два великі класи:

– Автоматизована система управління (АСУ) – з участю людини в контурі управління;

– Система автоматичного управління (САУ) – без участі людини в контурі управління.

Система автоматичного управління, як правило, складається з двох основних елементів – об'єкта управління і керуючого пристрою.

У замкнутих системах автоматичного регулювання керуючий вплив формується в безпосередній залежності від керованої величини. Зв'язок виходу системи з його входом називається зворотним зв'язком. Сигнал зворотного зв'язку віднімається з задає впливу. Такий зворотній зв'язок називається негативним.

Сутність принципу разомкнутого управління полягає в жорстко заданою програмою управління. Тобто, управління здійснюється « наосліп», без контролю результату, ґрунтуючись лише на закладеній в САУ моделі керованого об'єкта. Приклади таких систем: таймер, блок управління світлофора, автоматична система поливу газону, автоматична пральна машина і т. п.

Залежно від опису змінних системи діляться на лінійні та нелінійні. До лінійним відносяться системи, що складаються з елементів опису, які задаються лінійними алгебраїчними або диференціальними рівняннями.

Якщо всі параметри рівняння руху системи не змінюються в часі, то така система називається стаціонарною. Якщо хоча б один параметр рівняння руху системи змінюється в часі, то система називається нестаціонарної або із змінними параметрами.

Системи, в яких визначені зовнішні (задаючи) і описуються безперервними або дискретними функціями в часі, відносяться до класу детермінованих систем.

Системи, в яких має місце випадкові сигнальні або параметричні впливу і описуються стохастичними диференціальними або різницевими рівняннями, відносяться до класу стохастичних систем.

Якщо в системі є хоча б один елемент, опис якого задається рівнянням приватних похідних, то система відноситься до класу систем з розподіленими змінними.

Системи, в яких безперервна динаміка, породжувана в кожен момент часу, перемежовується з дискретними командами, що посилаються ззовні, називаються гібридними системами.

3.2 Поняття стійкості системи

У статичному режимі роботи всі складові вектора стану САУ не залежать від моменту часу їх розгляду і залишаються постійними, відповідними умові рівноваги системи. Це стан залежно від структури і параметрів САУ може бути стійким або нестійким.

Якщо після зміни вектора зовнішніх дій система приходить в стан, при якому всі складові вектора її стану стають постійними, тобто система повертається в положення рівноваги, то це стан рівноваги є стійким.

У разі, коли після зміна вхідного сигналу або обурення, система не прагне в первинний стан, а вектор вихідних сигналів змінюється незалежно від зовнішньої дії, то такий стан є нестійким. В цьому випадку система автоматичного управління є нестійкою.

4 ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці

Проаналізуємо умови праці та характеристики приміщення, що використовується під час розробки SPIN-бібліотеки для відображення інформації з мультипроцесорної системи. Бакалаврська робота виконується в приміщенні, що є обчислювальним центром (ВЦ). Розміри приміщення 11м х 8,5 м х 4 м, що становить площу 93,5 м^2, об`єм – 374 м^3. Кількість працюючих – дванадцять осіб. У приміщенні розміщено 12 ПЕОМ типу IBMPC/AT з рідкокристалічними моніторами.

Приміщення ОЦ, виходячи з норм на окремі робочі місця, відповідає вимогам НПАОП 0.00-1.28-10 – на одного працюючого доводиться 7,8 м^2 площі та 31,2 м^3 обсягу за норми 6 м^2 й 20 м^3 на одного працюючого.

До елемента Предмет праці, як відомо, відносяться вироби, що обробляються. У нашому випадку це матеріали для SPIN-бібліотеки.

Відповідно до ГОСТ 12.0.003-74 на користувача ПЕОМ у процесі трудової діяльності впливають такі небезпечні та шкідливі виробничі фактори (НШВФ):

а) фізичні НШВФ:

– підвищений рівень шуму;

– підвищена або знижена температура, вологість та підвищена рухливість повітря в робочій зоні;

– недостатня освітленість робочої зони,

– відсутність або нестача природного світла;

– ідвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини;

б) психофізіологічні НШВФ:

– статичні навантаження;

– перенапруга зорових аналізаторів;

– монотонність праці;

– розумова перенапруга;

в) хімічні НШВФ – відсутні;

г) біологічні НШВФ – відсутні.

Домінуючий фактор – підвищене значення напруги в мережі та, як наслідок, існує можливість виникнення спалаху. Відповідно до цього фактора було проведено розрахунок кількості вогнегасників у приміщенні ОЦ.

4.2 Виробнича безпека у приміщенні ОЦ

Електроустаткування в ОЦ живиться від трифазної чотирипровідної мережі з глухозаземленою нейтраллю змінного струму напругою 380/220 В частотою 50 Гц.

За рівнем небезпеки ураження електричним струмом за Правила пристрою електроустановок – 2011 приміщення відноситься до приміщень без підвищеної небезпеки, так як приміщення сухе, підлога ізолююча (покрита лінолеумом), а також відсутня можливість одночасного дотику людини до заземлених металоконструкцій будівлі з одного боку та до корпусів електрообладнання з іншого [25].

Для захисту людей від ураження електричним струмом слід передбачати блокувальні пристрої, електричний поділ мережі, занулення, подвійну ізоляцію, захисне вимкнення. (ГОСТ 12.1. 019-79, НПАОП 40.1-1.32-01).

Відповідно до вимог електробезпеки, у приміщенні, що розглядається, повинні бути вжиті наступні заходи. Лінія електромережі для живлення ЕОМ повинна бути виконана як окрема трипровідна групова мережа шляхом прокладання фазного, нульового робочого і нульового захисного провідників.

Площа поперечного перерізу нульового робочого та нульового

захисного провідників повинна бути не меншою за площу поперечного перерізу фазного провідника. Кабельні провідники ЕОМ та засоби передачі інформації захищені спеціальними чохлами.

Мережа повинна бути обладнана апаратурою захисту від короткого замикання та інших аварійних режимів, що забезпечує автоматичне відключення електроустановок від мережі при виникненні небезпеки ураження людини струмом за 0,2 с [26].

Повинно бути виконане занулення – з'єднання металевих нетоковедущих частин електрообладнання з нульовим проводом, а для зниження потенціалу, винесеного на занулені корпуси при обриві нульового робочого провідника живильної лінії, рекомендується організувати повторне заземлення нульового проводу електричної мережі за допомогою штучного заземлювача.

Також необхідно проводити заходи щодо перевірки ізоляції струмопровідних частин не рідше 1 разу на рік, опір якої має становити не менше 500 кОм.

4.3 Виробнича санітарія у приміщенні ОЦ

Робота в ОЦ проводиться сидячи і не вимагає систематичної фізичної напруги. Відповідно до ГОСТ 12.1.005-88 та ДСН 3.3.6.042-99 робота відноситься до категорії легкої 1а і для неї встановлені параметри мікроклімату, наведені в таблиці 4.1.

Таблиця 4.1 – Оптимальні норми мікроклімату

Пора року	Температура повітря, град. С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодне	22-24	40-60	≤ 0.1
Тепле	23-25	40-60	≤ 0.1

Для підтримки параметрів повітряного середовища в допустимих межах, що забезпечують надійну роботу ПЕОМ та комфортні умови праці для персоналу в приміщенні, повинна використовуватися: у холодний період року – опалення, у теплий період року – установка для кондиціювання повітря згідно зі СНиП 2.04.05-86 [27].

Роботи з набору, редагування текстів та іншої інформації за допомогою ПК відносяться до III розряду зорових робіт, над документами, Е = 300 лк при загальному освітленні [28].

Рівень шуму відповідно до ДСН 3.3.6-0.37-99 не повинен перевищувати 50 дБ. А при перевищенні цього рівня використовують звукоізоляційні та звукопоглинаючі матеріали для облицювання стін та стель.

4.4 Пожежна безпека виробничого приміщення ОЦ

По вибухопожароопасности в обчислювальний центр відноситься до пожежонебезпечної категорії В так як є тверді спалені матеріали, з температурою спалаху 61° С, будинок II ступеня вогнестійкості згідно з ДБН В 1.1.7-2002, виконано з вогнестійкого матеріалу – цегли. За ПУЕ-2011 приміщення відноситься до класу П-IIа.

Причиною пожежі приміщення ОЦ можуть бути коротке замикання електропроводки; несправність ПЕОМ та іншого електрообладнання; нагрівання провідників; куріння в недозволеному місці.

Для евакуації людей під час пожежі використовується робочий вихід шириною 1,5 м та висотою 2 м. Відстань від дверей до віддаленого місця в приміщенні не перевищує 25 м, відповідно до ДБН В.1.1.7-2002. Ширина коридору 2 м.

У приміщенні знаходиться схема евакуації у разі пожежі, розміщена на видному місці, яка наведена на рисунку 4.1.

Щоб забезпечити безпеку людей при роботі з комп'ютерами, слід дотримуватись загальних вимог пожежної безпеки, визначених у ОСТ 12.1.004-91 [29].

Відповідно до НАПБ Б.03.001-2004 виробничі, складські, лабораторні, адміністративні та побутові будинки та приміщення об'єктів різного призначення, громадські будівлі та споруди повинні бути оснащені переносними або пересувними вогнегасниками, відповідно до норм таблиці 4.2.

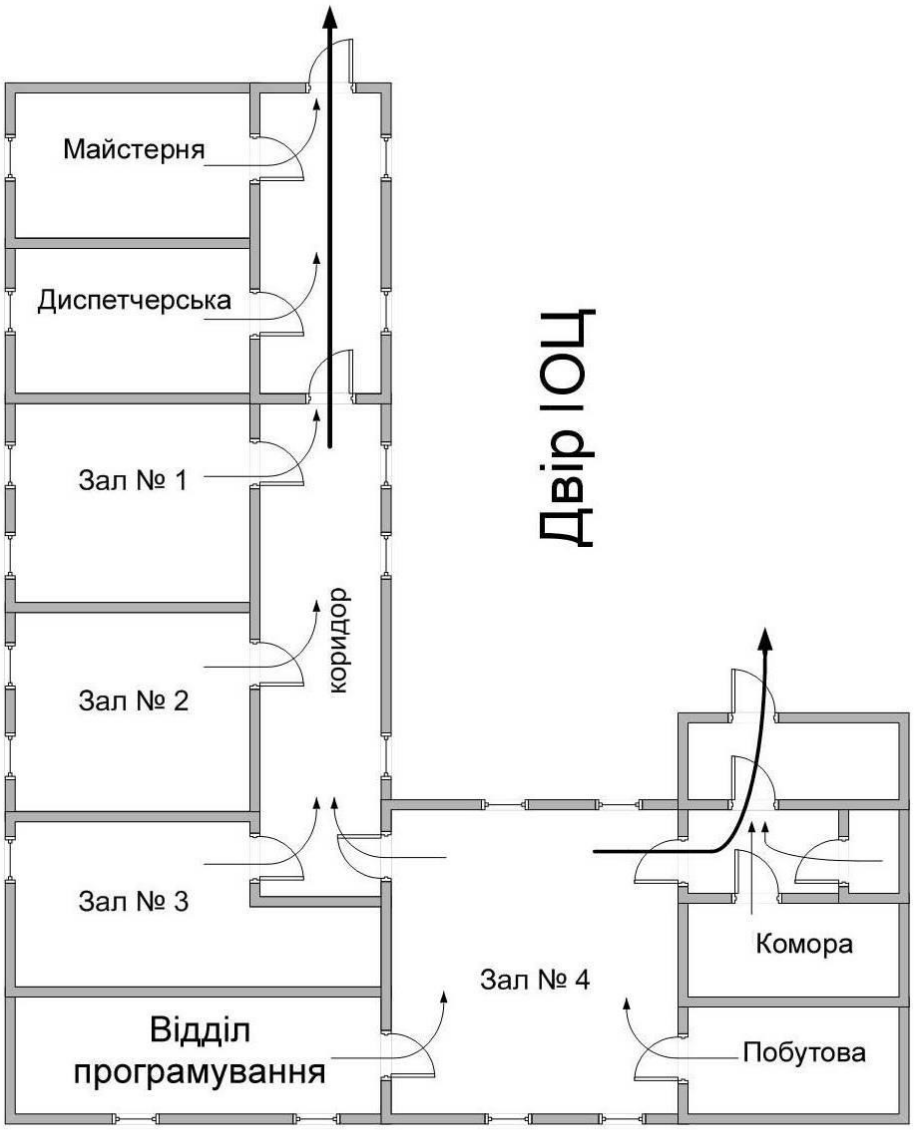


Рисунок 4.1 – План евакуації людей та майна з аудиторії №151 ІОЦ

Таблиця 4.2 – Норми оснащення приміщень вогнегасниками

Категорія приміщення	Гранична площа, що захищається, кв. м	Клас пожежі	Пінні та водні вогнегасники місткістю 10 л	Порошкові вогнегасники місткістю, л			Хладонові вогнегасники місткістю, л	Вуглекислотні вогнегасники, місткістю, л	
				2	5	10	2 (3)	2	5 (8)
А, Б, В (горючі гази та рідини)	200	А	2++	-	2+	1++	-	-	-
		Б	4+	-	2+	1++	4+	-	-
		С	-	-	2+	1++	4+	-	-
		Д	-	-	2+	1++	-	-	-
		(Е)	-	-	2+	1++	-	-	2++
В	400	А	2++	4+	2++	1+	-	-	2+
		Д	-	-	2+	1++	-	-	-
		(Е)	-	-	2++	1+	2+	4+	2++
Г	800	В	2+	-	2++	1+	-	-	-
		С	-	4+	2++	1+	-	-	-
Г, Д	1800	А	2++	4+	2++	1+	-	-	-
		Д	-	-	2+	1++	-	-	-
		(Е)	-	2+	2++	1+	2+	4+	2++

Примітки до таблиці 4.2:

– для гасіння пожеж різних класів порошкові вогнегасники повинні мати відповідні заряди: для класу А – порошок АВС (Е); для класів В, С та (Е) - ВС (Е) або АВС (Е) та класу Д – Д;

– знаком "++" позначені об'єкт, що рекомендуються до оснащення, вогнегасники, знаком "+" - вогнегасники, застосування яких допускається за відсутності рекомендованих і при відповідному обґрунтуванні, знаком "-" - вогнегасники які не допускаються для оснащення даних об'єктів;

– у замкнутих приміщеннях обсягом трохи більше 50 куб. м для гасіння пожеж замість переносних вогнегасників, або додатково до них, можуть бути використані вогнегасники самоспрацьовують порошкові.

При захисті приміщення від пожежі з наявністю ПЕОМ згідно з НАПБ Б.03.001-2004 слід використовувати вуглекислотні вогнегасники.

Приміщення ОЦ, в яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 (старе позначення – ОУ-2) або ВВК-2 (старе позначення – ОУ-3) або один ВВПА-400 на три ПЕОМ але не менше ніж один вогнегасник зазначених типів на приміщення.

Для розрахунку кількості вогнегасників у приміщенні ОЦ необхідно знати [30].

а) категорії приміщення. Визначено раніше – категорія В;

б) клас пожежі. Визначаємо за стандартом ISO №3941-77:

– клас А – пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, текстиль, папір);

– клас В – пожежі горючих рідин або твердих речовин, що плавляться;

– клас С – пожежі газів;

– клас D – пожежі металів та його сплавів;

– клас Е – пожежі, пов'язані з горінням електроустановок.

Можливі пожежі у ОЦ відносяться до класу Е або А.

Загальна площа аудиторії, яку займає аудиторія ОЦ кладає S = 93,5 ($м^2$). У таблиці 4.2 найближча (велика) гранична площа, що захищається 200 $м^2$. Для категорії приміщення В, класу пожежі Е, та займаної площею до 200 $м^2$ рекомендовано два вуглекислотні вогнегасники ємністю 5 (8) л.

ВИСНОВКИ

У кваліфікаційній роботі виконано аналіз архітектури мікроконтролера Paralax Propeller, а саме досліджено принцип функціонування обчислювальних ядер та процедуру перемикання між ними. Також проведено аналіз мов програмування. В результаті проведеного аналізу встановлено, що при використанні мови Spin немає втрати швидкодії, на відміну від застосування мови Сі та її подальшої компіляції в асемблер, а всі об'єкти в мікроконтролері Propeller складаються з коду Spin і коду Propeller асемблера. Таким чином, зроблено висновок про доцільність застосування SPIN-бібліотеки для відображення інформації з мультипроцесорної системи, побудованої на основі мікроконтролера Propeller.

Також у роботі розроблено алгоритмічну та програмну структуру SPIN-бібліотеки, програмної частини бібліотеки. Основними функціями розробленої бібліотеки є виведення графічної та текстової інформації на екран за інтерфейсом VGA. Кожному з примітивів бібліотеки можна встановити потрібний колір і колір фону. Таким чином було виконано мету роботи, а сме проведено розробку SPIN-бібліотеки для мікроконтролера Parallax Propeller P8X32A.

У розділі “Охорона праці” було розглянуті питання промислової безпеки, виробничої санітарії та пожежної профілактики, які стосуються безпосередньо теми дипломної роботи. Виділено небезпечні та шкідливі виробничі фактори, визначено заходи, що забезпечують вимоги безпечної праці при роботі на ПЕОМ.

Отримані результати кваліфікаційної роботи орієнтовані на практичне використання розробленої бібліотеки для мікроконтролерів серії Propeller. Розробка SPIN-бібліотеки для відображення інформації з мультипроцесорної системи на базі мікроконтролера Parallax P8X32A Propeller демонструє значний потенціал у підвищенні ефективності та зручності роботи з багатоядерними системами. Завдяки використанню багатоядерної

архітектури Propeller, бібліотека забезпечує паралельну обробку даних і швидкий відгук системи, що є критично важливим для задач реального часу.

Таким чином, ця робота не лише підтверджує актуальність і ефективність мікроконтролера Propeller, але й відкриває нові можливості для його застосування у різноманітних сферах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.ДСТУ 3008 – 2015. Документація. Документація, звіти у сфері науки. Структура і правила оформлення. Введ.2015-06-22. К. Держстандарт України, 2017. 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Системна інженерія» / Упоряд.: І. Ш. Невлюдов, О. М. Цимбал, О. В. Токарєва, А. І. Бронніков. Харків: ХНУРЕ, 2022. 66 с.

3.Диплмне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології»: навч. посібник / І. Ш. Невлюдов, В. А. Андрусевич, О. В. Токарєва, Г.В. Пономарьова, Київ, 2018. 320 с.

4. Невлюдов І. Ш. Механізми технічних засобів автоматизації (довідкові матеріали з курсового і дипломного проектування): навчальний посібник. / І.Ш. Невлюдов, В.І. Роменський, І.О. Яшков. – Харків: ХНУРЕ, 2021. – 292 с.

5. Невлюдов І. Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг : Криворізький коледж НАУ, 2019. – 366 с.

6. Баскаков С. С. Мікроконтролер Propeller від Parallax / С.С. Баскаков. // Новини електроніки. – 2008. – №2. – С. 30-35.

7. Мікроконтролер Parallax Propeller [Електронний ресурс] – Режим доступу: http://www.kosmodrom.com.ua/data/PropellerManual-v1.1.pdf

8. Contributed Code for the Propeller Microcontroller [Електронний ресурс] – Режим доступу до ресурсу: http://obex.parallax.com/projects.

9. Міні-комп'ютер на базі мікроконтролера Parallax Propeller Propeller [Електронний ресурс] – Режим доступу до ресурсу: Режим доступу: http://habrahabr.com/post/159847/.

10. GUIDemo – A full VGA Library for the Propeller [Електронний ресурс] – Режим доступу до ресурсу: http://www.yourwarrantyisvoid.com/2010/11/19/guidemo-a-full-vga-library-for-the-propeller/.

11. Propeller Education Kit Labs Propeller [Електронний ресурс] – Режим доступу до ресурсу: http://www.parallax.com/Portals/0/Downloads/docs/prod/prop/PEKitLabs-v1.2.pdf.

12. Programming and Customizing the Multicore Propeller Microcontroller, 2010. – 496 p.

13. Harprit Sandhu Programming the Propeller with Spin: A Beginner's Guide to Parallel Processing – New York, 2010. – 368 p.

14. Євстіфєєв А. В. Мікроконтролери AVR сімейств Tiny та Mega фірми ATMEL / А. В. Євстіфєєв. – М: Видавничий дім Додека-XXI, 2004. – 560 с.

15. Пушкарьов О. MSP430 та продукція Chipcon / О. Пушкарьов. // Новини електроніки. – 2007. – №3. – С. 11-14.

16. Джозеф Ю. Повний посібник з ARM Cortex-M3 / Ю. Джозеф., 2007. – 520 с.

17. Євстіфєєв А. В. Сімейство мікроконтролерів MSP430x2xx. Архітектура. Програмування. Розробка додатків / А. В. Євстіфєєв. – М: Видавничий дім Додека-XXI, 2010. – 544 с.

18. Зотов В. Ю. Проектування мікропроцесорних систем, що вбудовуються, на основі ПЛІС фірми XILINX / В. Ю. Зотов., 2006. – 522 с.

19. Bailey D. G. Design for embedded image processing on FPGAs / D. J. Bailey. – Singapore: John Wiley & Sons (Asia), 2011. – 341 p.

20. Woods R. FPGA-based implementation of signal processing systems / Woods. – Chichester, U.K.: John Wiley & Sons, 2008. – 217 p.

21. Rueggeberg J. Programming VGA graphics / J. Rueggeberg., 2009. – 655 с.

22. Фролов А. Програмування відеоадаптерів CGA, EGA та VGA / А. Фролов. – М: Діалог-МІФІ, 1992. – 287 с.

23. Рачков М. Ю. Технічні засоби автоматизації / М. Ю. Рачков. – М: МДІУ, 2009. – 185 с.

24. Алексєєв К. Б. Мікроконтролерне керування електроприводом / К. Б. Алексєєв, К. А. Палагута. – М: МДІУ, 2008. – 302 с.

25. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин. Чинний від 2000-01-01. Вид. офіц. Київ : Вид-во стандартів. 10 с.

26. НПАОП 0.00-4.12-05. Типове положення про навчання, інструктаж та перевірку знань працівників з питань охорони праці установок. Чинний від 2006-01-01. Вид. офіц. Київ : Вид-во стандартів, 2005. 7 с.

27. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень установок. Чинний від 2000-01-01. Вид. офіц. Київ : Вид-во стандартів, 1999. 11 с.

28. ДБН В.2.5-28-2006. Природне та штучне освітлення. Чинний від 2007-01-01. Вид. офіц. Київ : Вид-во стандартів, 2006. 62 с.

29. ДБН В.1.1.7-2002. Захист від пожежі. Пожежна безпека будівництва. Чинний від 2003-05-01. Вид. офіц. Київ : Вид-во стандартів, 2002. 42 с.

30. Дзюндзюк Б. В. Охорона праці. Збірник завдань / Б. В. Дзюндзюк, В. Г. Іванов. – Харків: НВП центр ХНУРЕ, 2006. – 242 с.