МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

О.М. Павловський

Мікроконтролери та мікропроцесорна техніка

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерно-інтегровані технології та системи навігації і керування» та «Комп'ютерно-інтегровані системи та технології в приладобудуванні»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти: *Галаган Р.М.* — канд. техн. наук, доцент, доцент кафедри приладів і систем неруйнівного контролю КПІ ім. Ігоря Сікорського
Головач С.В. — канд. техн. наук, головний конструктор напрямку АТ «Елміз»

Відповідальний редактор: *Лакоза С.Л.*, канд. техн. наук, доцент кафедри приладів і систем орієнтації і навігації КПІ ім. Ігоря Сікорського

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 8 від 24.06.2021 р.) за поданням Вченої ради Приладобудівного факультету (протокол № 5/21 від 31.05.2021 р.)

Електронне мережне навчальне видання

Павловський Олексій Михайлович, канд. техн. наук, доц.

МІКРОКОНТРОЛЕРИ ТА МІКРОПРОЦЕСОРНА ТЕХНІКА

Лабораторний практикум

Мікроконтролери та мікропроцесорна техніка. Лабораторний практикум. [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / О.М. Павловський; КПІ ім. Ігоря Сікорського, 2021. — 104 с.

Навчальний посібник містить теоретичні відомості та практичні рекомендації до виконання лабораторних робіт з дисципліни «Мікроконтролери та мікропроцесорна техніка».

У процесі виконання лабораторних робіт студент познайомиться із трьома обчислювальними ядрами, 32 розрядними мікроконтролерами сімейства ARM, AT91SAM7X фірми Atmel та STM32F303VC фірми STMicroelectronics, та апаратною платформою Arduino.

ЗМІСТ

ОСНОВНІ СКОРОЧЕННЯ

ARM – Advanced RISC Machine, сімейство мікроконтролерів, сучасна архітектура.

ASCII – American standard code for information interchange, американська система кодування інформації, таблиця в якій символи кодуються порядковими номерами.

FPU – процесорне ядро для виконання операцій із плаваючою комою.

IDE – інтегроване середовище розробки.

KSPS – тисяч вибірок за секунду (характеристика швидкодії АЦП) .

MEMS – мікроелектромеханічні системи, на базі яких будуються сучасні чутливі елементи.

MIPS – мілiонів ітерацій за секунду, характеристика, що описує продуктивність обчислювального ядра.

PIO – контролер паралельного вводу-виводу МК AT91SAM7X.

PMC – контролер живлення МК AT91SAM7X.

АЦП – аналого-цифровий перетворювач.

БІНС – безплатформна інерціальна навігаційна система.

МК – мікроконтролер.

НОЗП – над оперативний запам’ятовуючий пристрій.

ЦАП – цифро-аналоговий перетворювач.

ШІМ – широтно-імпульсна модуляція.

ВСТУП

Із розвитком науки і техніки, з’являється все більша кількість електронних цифрових пристроїв. Сфера їх застосування не обмежується однією галуззю і простягається від цивільного використання, до військових застосунків. Із підвищенням попиту на такі пристрої з’явилась необхідність універсалізації виконавчих елементів, обчислювальних пристроїв, керуючих модулів та чутливих елементів. І не дивлячись, що такі універсальні чіпи у вигляді мікропроцесорів та мікроконтролерів існують вже понад 50 років, дійсно широкого вжитку та розповсюдження вони набули за останні десятиріччя. Відтак, сьогодні мікроконтролери присутні у більшості цифрової техніки і не дивлячись на масовість випуску, мають суттєві відмінності за своїми характеристиками, архітектурою призначенням та ін.

Таким чином, метою даного посібника є якомога ширше охоплення сучасних мікропроцесорних ядер, з метою виокремлення спільних рис, відмінностей та особливостей їх програмування, взаємодії із периферійними пристроями та стандартизованими інтерфейсами.

У процесі виконання лабораторних робіт студент познайомиться із трьома обчислювальними ядрами: 32-розрядними мікроконтролерами сімейства ARM – AT91SAM7X фірми Atmel та STM32F303VC фірми STMicroelectronics, та апаратною платформою Arduino. Дані обчислювальні ядра є найбільш типовими для ринку сучасних мікроконтролерів.

ЛАБОРАТОРНА РОБОТА № 1
СТВОРЕННЯ НОВОГО ПРОЕКТУ В СЕРЕДОВИЩІ KEIL uVISION

Мета роботи: ознайомитись з середовищем Keil uVision та навчитись створювати новий проект.

1.1. Теоретичні відомості

Для написання програмного коду, його відлагодження та перевірки на коректність, а також завантаження у пам’ять мікроконтролера (МК), будемо використовувати середовище Keil uVision IDE. Основними модулями Keil uVision IDE, що будуть використані при створенні програмного забезпечення та виконання лабораторних робіт, є менеджер проектів, редактор і відладчик. Під час редагування початкового коду можна конфігурувати відладчик, а в процесі відладки - відкоригувати початковий код. У режимі симуляції, відладчик моделює усю систему команд ARM і симулює усю периферію мікроконтролера. Таким чином, це дозволяє перевіряти коректність і працездатність створеного коду, фізично не маючи відладочної плати або мікроконтролера, що дозволяє знайомитись із особливостями даного МК не лише у лабораторії, а й віддалено.

Для детального тестування створеного коду, з точки зору безпеки, використовується аналізатор ефективності Code Coverage. Він виконує статистичний аналіз виконання програми, а також сприяє оптимізації створеного програмного забезпечення. База підтримуваних uVision мікроконтролерів Device Database автоматично конфігурує засоби розробки для усіх підтримуваних МК.

Keil uVision IDE має велику кількість прикладів, що дозволяє швидко почати роботу з МК сімейства ARM.

Пакет RealView Compilation Tools формує об'єктні файли, які містять повну інформацію для відладки, використовуючи вихідні файли із

розширенням *.c, отримані за допомогою uVision Debugger або внутрішньосхемного емулятора.

Компілятор дозволяє сумісне використання команд ARM і Thumb в одному проекті (Режим ARM потрібний для обробки переривань і швидких алгоритмів обробки сигналів, а режим Thumb - забезпечує мінімальний розмір коду). Також компілятор виконує вставки коду на мові ассемблер, за рахунок чого досягається повна оптимізація програми, хоча основний код створюється в межах синтаксису мов C/C++.

Відладчик (uVision Debugger) симулює вбудовану периферію ARM (I2C, CAN, UART, SPI, переривання, порти, АЦП, ЦАП і ШІМ), а також дозволяє вести відладку програм, написаних на мовах С і асемблері, або в змішаному форматі, зберігає історію трасування. Це особливо важливо при роботі із об’ємними проектами.

Вбудований в uVision Debugger аналізатор продуктивності (Performance Analyzer) відслідковує час виконання частин програми, що дозволяє більш гнучко створювати проект і оптимізувати систему для максимальної швидкодії-або ефективності.

uVision IDE пропонує широкі можливості по використанню простих і умовних точок зупинки. Умовою зупинки може бути або результат виразу, або операція звернення до сегменту пам’яті (читання, запис, доступ). Точки зупинки можуть тимчасово зупинити виконання програми або запустити команду/послідовність дій відладчика.

1.2. Створення нового проекту в середовищі Keil uVision

Після запуску програми Keil uVision, відкривається головне вікно, загальний вигляд якого представлено на рис.1.1.

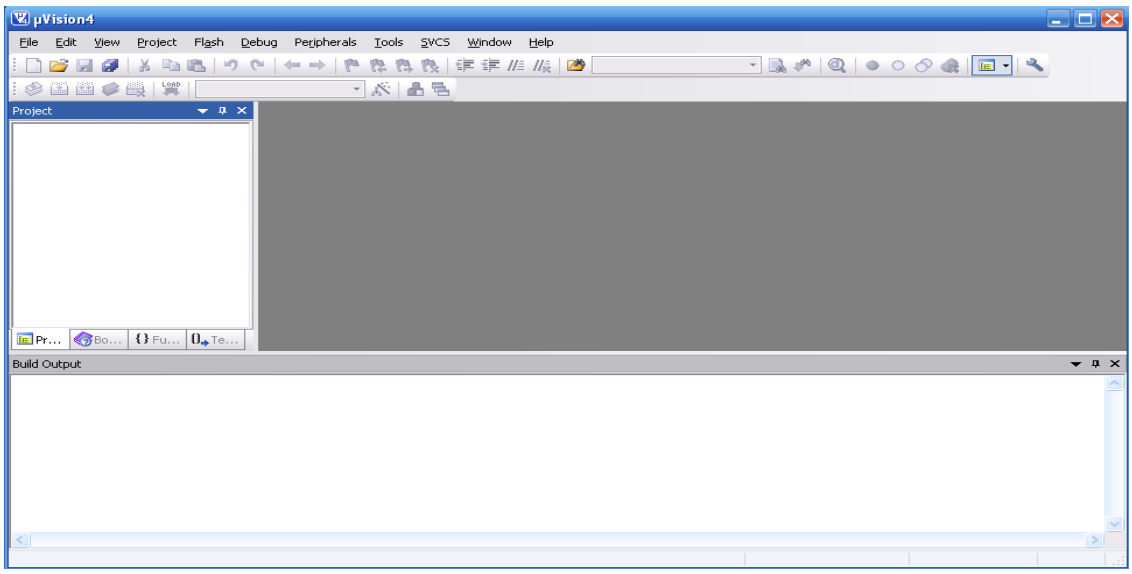


Рис.1.1. Основне вікно програми Keil uVision

Для створення нового проекту необхідно зробити наступні кроки:

Відкрити меню Project -> New uVision Project як показано на рис. 1.2. Вибрати папку призначення і назвати проект.

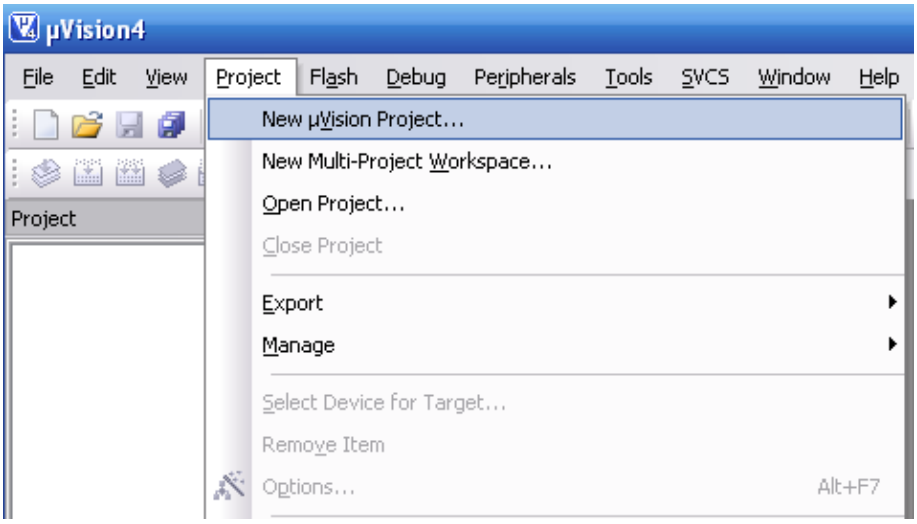


Рис.1.2. Створення нового проекту

Далі обрати папку фірми виробника, у нашому випадку Atmel, а з розгорнутого списку необхідно вибрати мікроконтролер AT91SAM7x256 і натиснути кнопку "ОК" (Рис. 1.3). Необхідно заначити, що при виборі мікроконтролера середовищем Keil uVision надається стисла інформація про основні характеристики і модулі мікроконтролера.

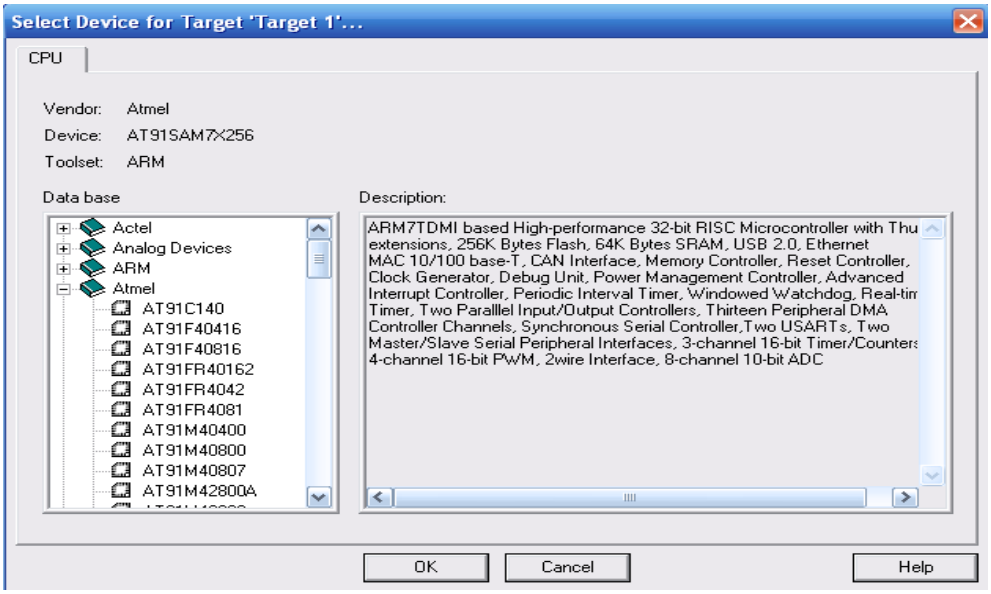


Рис. 1.3. Вибір мікроконтролера

Спочатку необхідно зберегти проект на ПК. Для цього відкриваємо меню File - > Save As…

У відкритому новому проекті є лише директорія Target 1 та файли проекту. Для додавання файлів, необхідно зробити подвійний клік на Source Group1. Далі у папці проекту, створюємо текстовий файл з ім'ям ***main.c.*** Другий варіант додавання файлів, а також налаштування проекту, є виклик меню (Components, Environment and Books). У вікні "Components, Environment and Books", в колонці "Project Targets" необхідно зробити дві категорії «Simulation» та «Flash», як наведено на рис. 1.4.

Перша категорія буде використовуватись при відладці проектів у режимі симуляції, тобто коли відладочна плата контролера фізично не під'єднана.

Для цього, у вкладці Project вибираємо команду Options for Target 'simulator'... . У вікні, що відкрилося, переходимо на вкладку Debug і вибираємо Use Simulator .

У колонку Files необхідно додати файл main.c (якщо цього не було зроблено раніше), після чого він з'явиться у вікні Project Workspace.

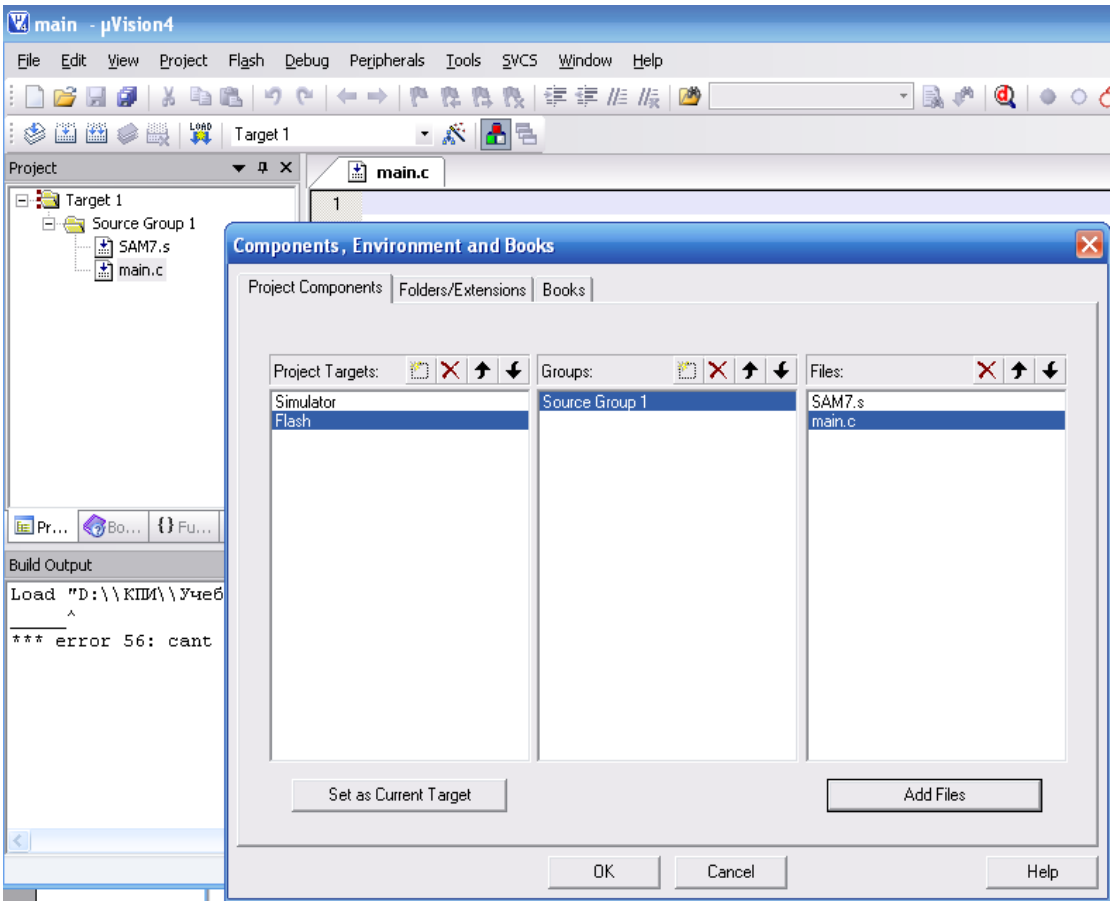
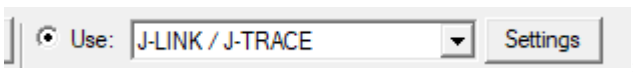


Рис. 1.4. Налаштування нового проекту Keil

Тепер необхідно сконфігурувати проект для завантаження створеного коду в МК за допомогою зовнішнього програматора. Будемо використовувати універсальний програматор фірми Segger з інтерфейсом JTAG. Зовнішній вигляд якого представлено на рис. 1.5.

Для цього, в основному вікні середовища Keil uVision вибираємо

і повторюємо попередню дію, вибираючи у вкладці Debug опцію



, відповідно вибираючи тип підключення (Рис. 1.6.).

а) б)

Рис. 1.5. Програматор Segger SAM-ICE: а) Загальний вигляд; б) Інтерфейс JTAG

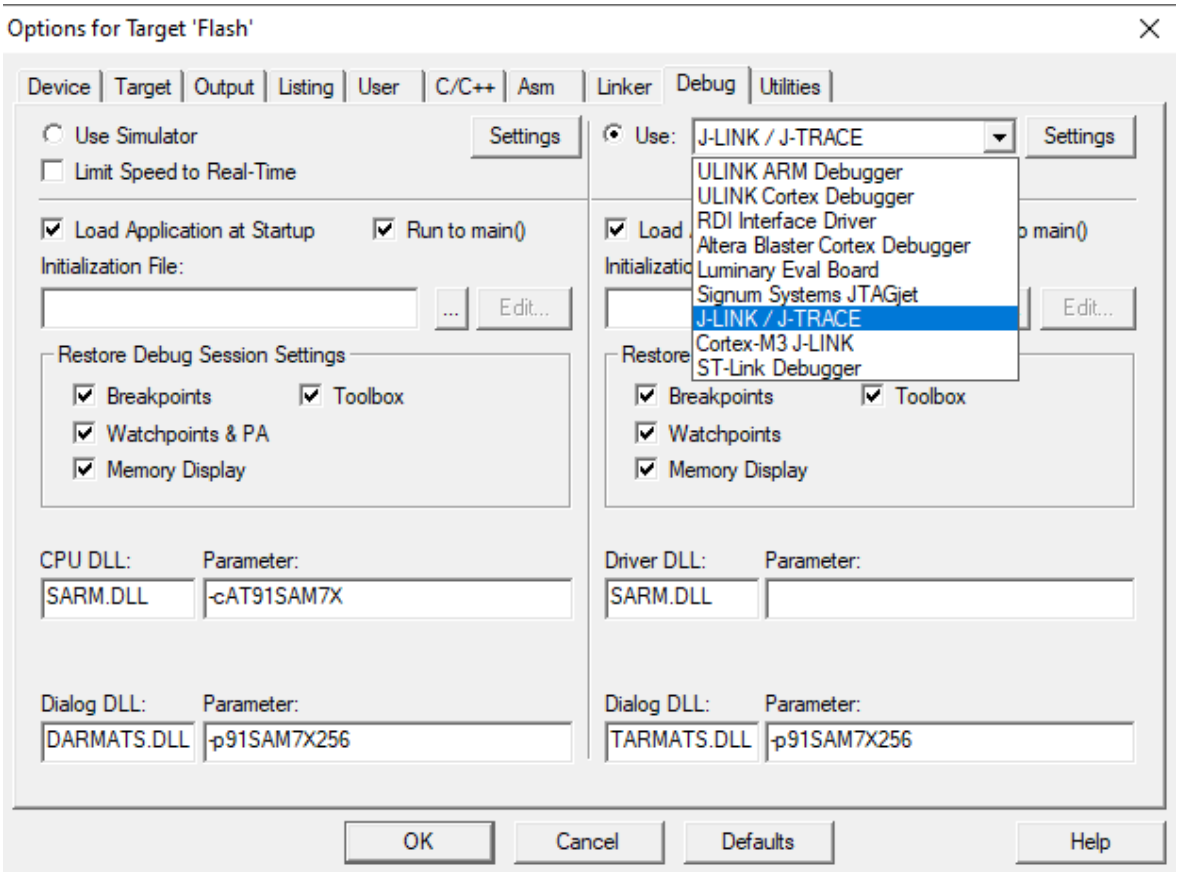


Рис.1.6. Вибір інтерфейсу програматора

Після цього натискаємо на кнопку «Settings» і у відкритому вікні із списку «Reset Straregy» обираємо пункт Software, for Atmel AT91SAM7 MCUs (рис. 1.7). І натискаємо кнопку «Ок».

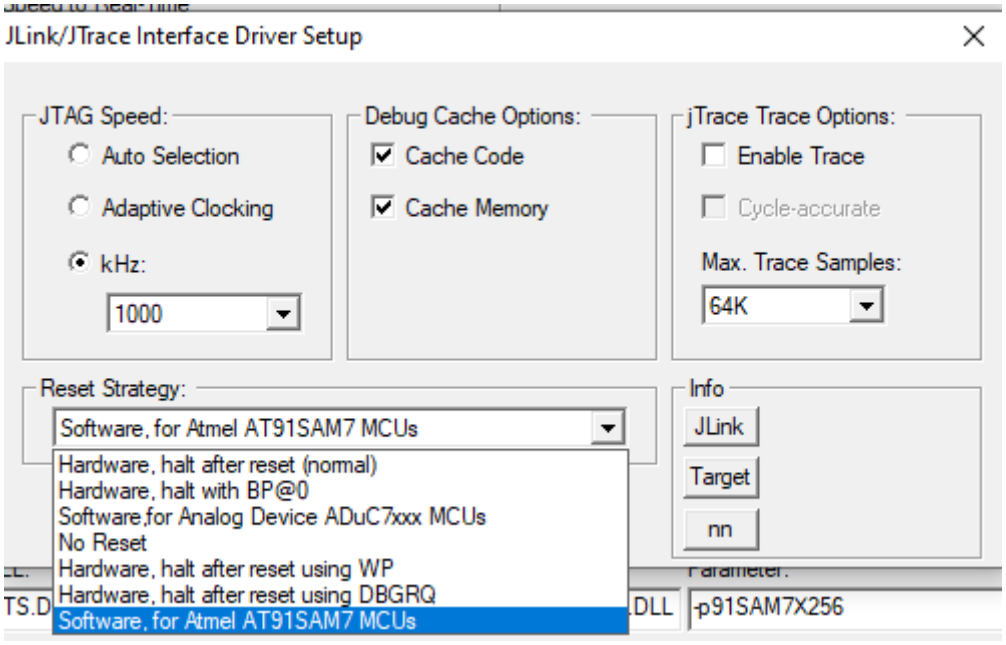


Рис.1.7. Налаштування програматора

Останнім кроком налаштування, переходимо у вкладку «Utilities», та обравши інтерфейс J-Link/J-Trace, як показано на рис. 1.8., натискаємо кнопку «Settings».

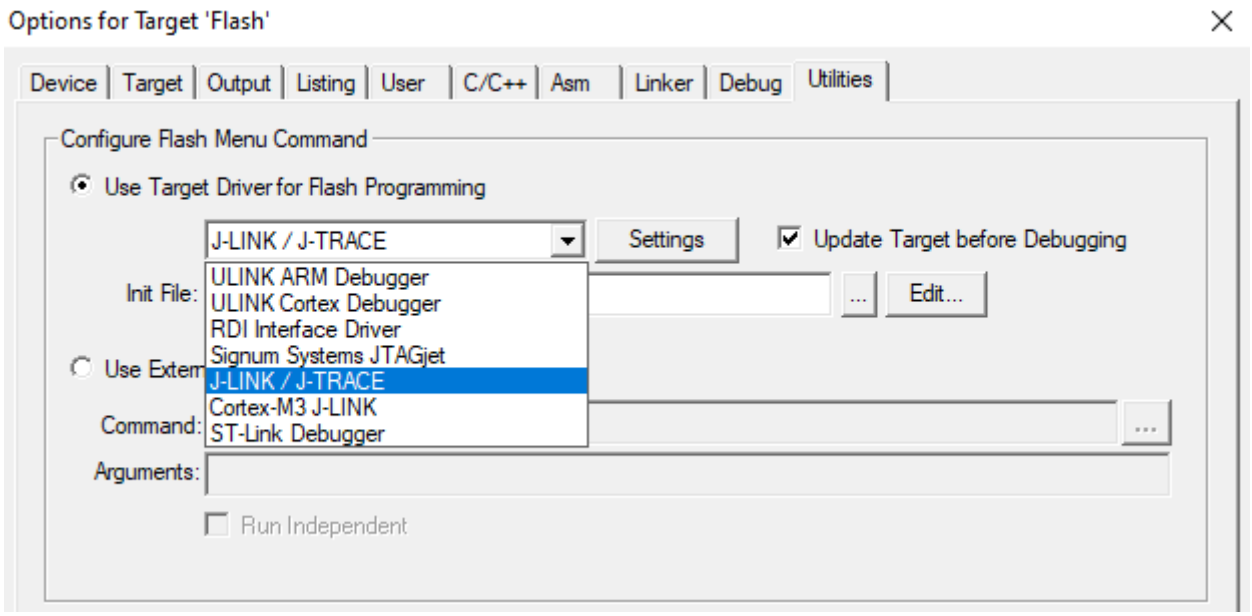


Рис.1.8. Налаштування програматора, вкладка «Utilities»

У відкритому вікні необхідно натиснути кнопку «Add» та обрати тип пам'яті для завантаження коду. Обираємо AT91SAM7 256kB Flash .

Наразі налаштування проекту завершено.

Тепер перейдемо до налаштувань самого коду. У вікні Project Workspace обираємо файл "main.c", та у розгорнутому пустому вікні починаємо писати код. Спершу, використовуючи директиву препроцесора #include <AT91SAM7X256.H>, відтак підключаємо головний заголовочний файл (бібліотеку) мікроконтролера. Для успішного приєднання бібліотеки до проекту, натисніть кнопку Rebuild All Targets .

Додамо головну функцію – для цього у строці 3 напишемо:

```
int main (void)
    {
    }
```

І знову натиснемо кнопку Rebuild All Targets .

Таким чином, завершений новий проект повинен виглядати наступним чином:

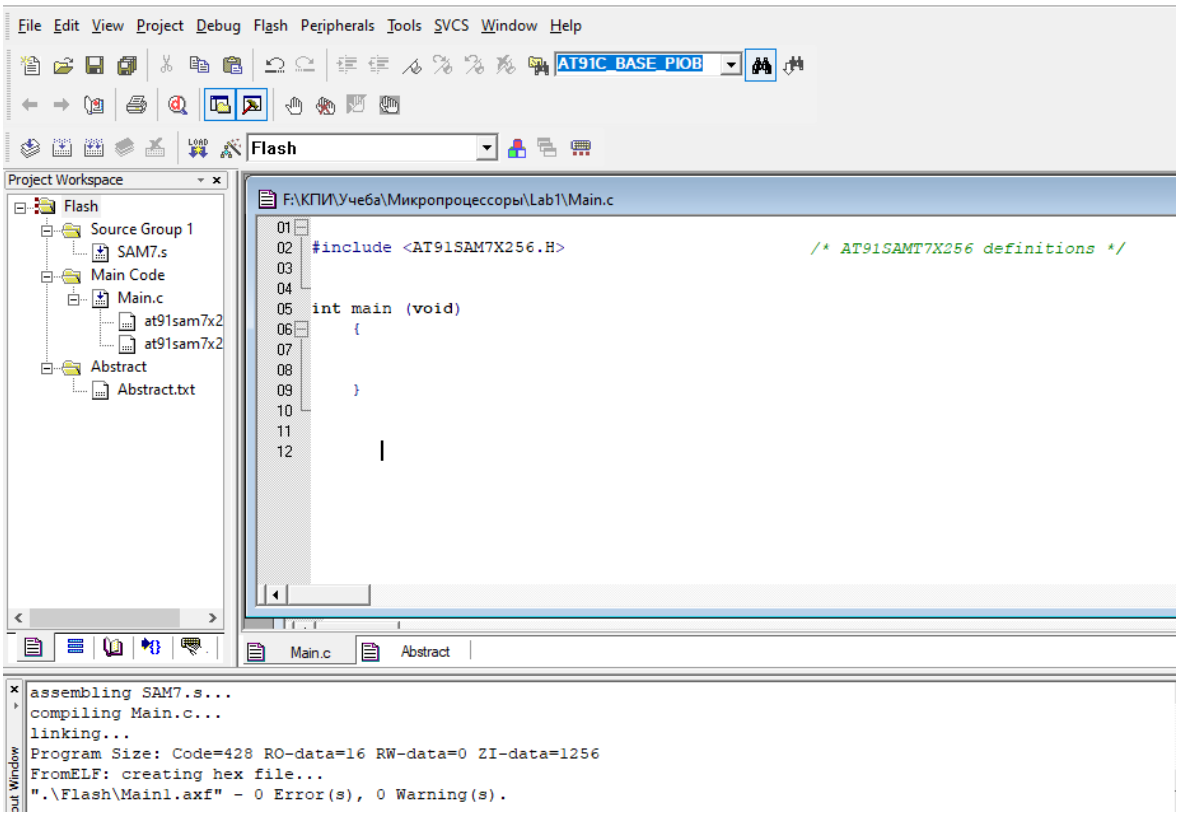


Рис. 1.9. Підключення бібліотеки проекту Keil

Правильність виконання дій, можна контролювати за допомогою сповіщень компілятора у нижньому вікні рис. 1.9. (0 Error (s))

Для відкриття вже існуючого проекту, з верхнього меню Project, оберіть 

. У середовищі Keil uVision4 файл проекту має розширення *. Uvproj, у ранніх версіях *.uv2, *.uv3. Проекти збережені у Keil uVision4 не мають зворотної сумісності із ранніми версіями. Ще однією відмінністю Keil uVision4 є відсутність бібліотеки функцій для МК AT91SAM7X, що спрощує написання коду. Таким чином, для виконання лабораторних робіт по МК AT91SAM7X пропонується використовувати Keil uVision 3.7.

1.3. Завдання

Повторити створення нового проекту у середовищі Keil uVision (2-3 рази). За допомогою кнопки Rebuild All Targets впевнитись в коректності виконання створеного проекту і відсутності помилок.

1.4. Зміст звіту

Зробити висновки по роботі.

1.5. Контрольні запитання

1. Що таке RealView Compilation Tools, навіщо потрібен цей модуль?
2. Із яких основних модулів складається пакет середовища uVision IDE?
3. Для чого створювати два типи запуску програм?
4. У чому особливості створення нового проекту?
5. Які настройки відповідають за налагодження програматора?
6. Що таке #include <AT91SAM7X256.H>?
7. У чому відмінність файлів проектів uVision IDE різних версій, чи сумісні ці проекти?

ЛАБОРАТОРНА РОБОТА № 2.
РОБОТА ІЗ КОНТРОЛЕРОМ ПАРАЛЕЛЬНОГО ВВОДУ/ВИВОДУ (PIO)

Мета роботи: Отримати базові навички програмування у середовищі Keil uVision, навчитися використовувати контролер паралельного вводу / виводу (PIO).

2.1. Теоретичні відомості

2.1.1. Загальні відомості про МК AT91SAM7X256

Мікроконтролер AT91SAM7X256 — це сучасний 32 розрядний мікроконтролер фірми Atmel, сімейства ARM7. Загальний вигляд мікросхеми МК на відладочній платі показано на рис. 2.1., а основними характеристиками є наступні:

Тактова частота AT91SAM7X256 складає 55 МГц. Не дивлячись на відносно невелику тактову частоту, контролер має швидкодію 30–60 MIPS, що достатньо для створення більшості сучасних пристроїв, від embedded до керування дронами та мультикоптерами.

Рис.2.1. МК AT91SAM7X256 на відладочній платі

ПЗП (Flash) – 256 кБ, та НОЗП (RAM) - 64 кБ, що достатньо для зберігання доволі потужних проектів і невеликих буферів інформації.

Мікроконтролер має роздільне живлення ядра і периферійних модулів – 1,8 В і 3,3 В відповідно. Таке поєднання параметрів дозволяє застосовувати мікроконтролер в найрізноманітніших програмах та проєктах, в тому числі для побудови систем з низьким енергоспоживанням, що працюють у реальному часі. Максимальний струм споживання, при частоті ядра 55 МГц, становить приблизно 30 мА, в режимі очікування - 16 мкА.

Для обміну інформації із зовнішніми системами передбачено низку стандартних інтерфейсів, серед яких такі найбільш розповсюджені як USART та UART, SPI, I2C, SSC, CAN.

Обробка аналогових сигналів здійснюється 10-бітним 8-канальним АЦП із швидкістю 384 KSPS (при частоті тактування АЦП 5 МГц) та 4-канальним 16-бітним ШІМ-контролером.

Лінії вводу/виводу працюють із напругою 3,0–3,6 В, проте всі лінії можуть працювати із 5В логікою, тобто можуть сприймати вхідний сигнал із рівнем логічної 1 – 5В.

Мікросхему AT91SAM7X256 представлено в корпусі QFP100 із 100 виводами.

З урахуванням складності МК AT91SAM7X256 тут наведені лише основні характеристики, які не є вичерпними, проте достатніми для виконання лабораторних робіт.

2.1.2. Відладочна плата

У даному курсі лабораторних робіт будемо використовувати відладочну плату AT91SAM7X-EK, загальний вигляд якої наведено на рис. 2.2. Наявні інтерфейси, їх виводи, а також периферійні пристрої, що спрощують ознайомлення із МК (світлодіоди, джойстик, т.д.), представлені у вигляді структурної схеми на рис. 2.3.

2.1.3. Контролер паралельного вводу / виводу (PIO)

Для обміну інформацією із зовнішніми системами МК AT91SAM7X256 використовує модулі контролерів паралельного вводу / виводу PIO. МК має у своєму складі два контролери паралельного вводу / виводу PIO: PIOA та PIOB.

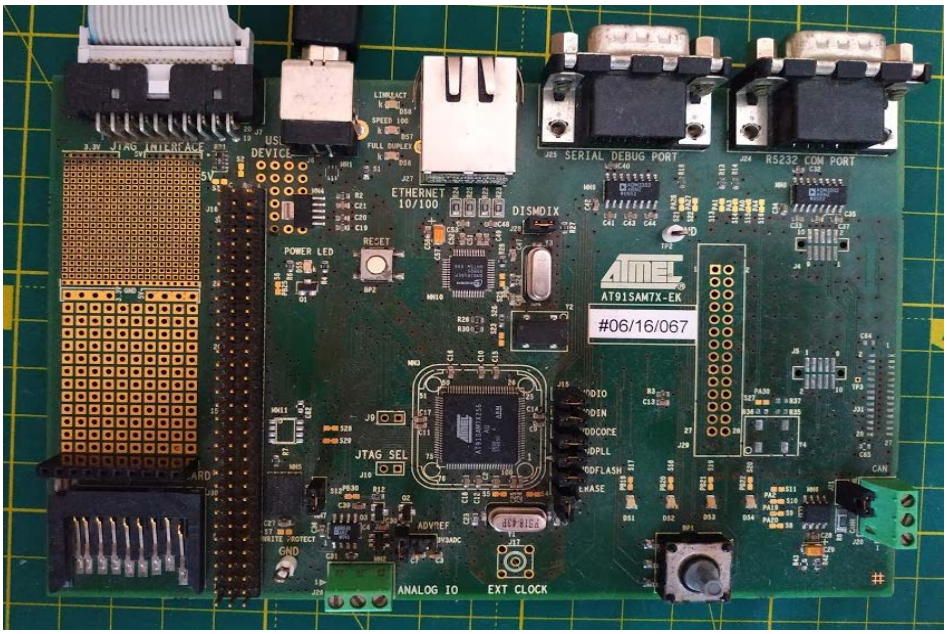


Рис.2.2. Загальний вигляд відладочної плати AT91SAM7X-EK

Кожен контролер PIO керує 32 лініями. Кожна із цих ліній може бути сконфігурована як лінія вводу / виводу загального призначення, або на виконання деяких функцій, що відноситься до вбудованого периферійного пристрою. Кожна лінія вводу / виводу має свою унікальну адресу в межах 32-бітових регістрів 32-бітного інтерфейсу користувача PIO контролера. Кожна лінія вводу / виводу контролера PIO має такі особливості та функції:

• Забезпечує слідкування за рівнем вхідного сигналу і керування вихідним сигналом.

• Генерує запит переривання по зміні стану на вході, що дозволяє легко виявляти зміну зовнішнього сигналу на будь-якій лінії вводу / виводу.

• Має вбудований фільтр, що дозволяє відфільтрувати завади або випадкові імпульси зовнішніх сигналів з тривалістю менше половини тактового циклу.

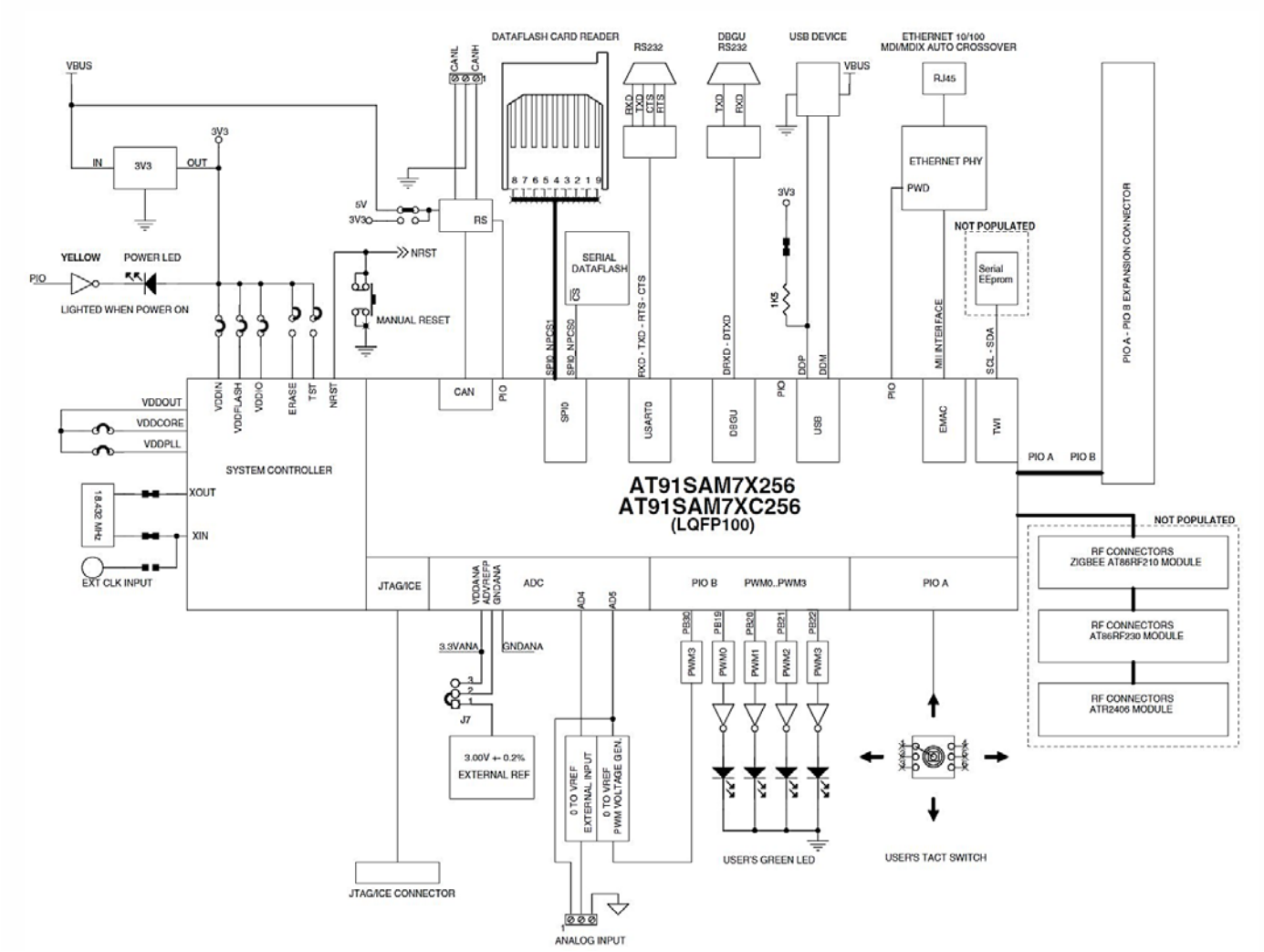


Рис.2.3. Структурна схема відладочної плати AT91SAM7X-EK

• Може керуватись від декількох драйверів, оскільки має організацію виходу з відкритим стоком.

• Має вбудований «підтягуючий до плюса живлення» резистор, який керується програмно.

Контролер PIO має також механізм синхронного виведення, що забезпечує виведення до 32 біт даних однією операцією запису.

У перших лабораторних роботах, для перевірки працездатності написаного коду, будемо використовувати у якості індикативних елементів, вбудовані на плату світлодіоди. Вони підключені до ліній № 19-22 PIOB контролера, що показано на структурній схемі (рис. 2.3).

Увага! Задля того, щоб при задіянні зазначених ліній, світлодіоди не збільшували струм споживання, і відповідно, при рівні логічної одиниці не змінювали напругу логічного рівня, світлодіоди підключені через інвертор!

Структурна схема PIO контролера, карта регістрів та їх опис, доступні в мережевому оточенні комп'ютерного класу.

2.2. Завдання до виконання

2.2.1. Код програми

У новому проєкті набрати приведений нижче код:

```
#include <AT91SAM7X256.H>
#include <lib_AT91SAM7X256.h>
#include "AT91SAM7X-EK.h"

    int main (void) {

AT91F_PIO_CfgOutput(AT91C_BASE_PIOB, AT91B_LED_MASK);

AT91F_PIO_SetOutput(AT91C_BASE_PIOB, AT91B_LED_MASK);
}
```

Увага! Код необхідно набрати самотужки, а не скопіювати. Такий підхід сприяє кращому запам'ятовуванню послідовності дій, набуття навичок програмування МК, а також кращому розумінню коду.

2.2.2. Пояснення тексту програми

З початку виконується підключення необхідних для роботи МК заголовочних файлів/бібліотек:

- « #include <AT91SAM7X256.H> » - бібліотека контролера. У цій бібліотеці містяться основні регістри контролера, адреси головних модулів та периферії;

- « #include <lib_AT91SAM7X256.h>» - бібліотека функцій. Для спрощення написання програмного коду, звернення до регістрів представлено у вигляді функцій швидкого керування, опис функцій наведено у коментарях в даному файлі;
- « #include "AT91SAM7X-EK.h"» - бібліотека відладочної плати. Бібліотека містить ключові значення для роботи із вбудованими периферійними пристроями плати (світлодіоди, кнопки, ін.).

Оскільки бібліотека копіюється у папку проекту, то вона доступна для модифікації, а її втрата або спотворення коду, не призведе до втрати всіх проектів Keil.

Перша строка у головній функції int main (void), функція AT91F_PIO_CfgOutput. Ця функція використовується, щоб сконфігурувати лінії PIO контролерів на вихід. Функція має два аргументи. Перший AT91C_BASE_PIOB – базова адреса PIOB контролера, що визначає початкову адресу із якої починаються адреси основних регістрів PIOB контролера користувальницького інтерфейсу. По суті, вказує, що використовувати будемо лінії, що керуються PIOB контролером. Другий аргумент - значення, що вказує які лінії треба сконфігурувати. У даному випадку значення AT91B_LED_MASK вказує на лінії №19-22, що відповідають лініям, до яких підключені світлодіоди на платі. Впевнитись у цьому можна відкривши бібліотеку AT91SAM7X-EK.h, двічі клікнувши по ній у вікні Project Workspace.

Там, у строці №31 побачимо наступний запис #define AT91B_LED_MASK (AT91B_LED1| AT91B_LED2| AT91B_LED3| AT91B_LED4), тобто це об'єднання визначень номерів ліній всіх чотирьох наявних на платі світлодіодів (зауважимо на використанні оператора «побітового Або» |).

У свою чергу, визначення у строці №26 бібліотеки AT91SAM7X-EK.h #define AT91B_LED1 (1<<19) слід читати як одиниця зміщена вліво на 19

розрядів. Це одна із форм записів двійкових значень, що використовується у середовищі Keil uVision.

З врахуванням того, що у пізніх версіях Keil uVision (починаючи з 4) бібліотека lib_AT91SAM7X256.h відсутня, то дослідивши вміст функції, наприклад, AT91F_PIO_CfgOutput, можемо перейти до запису безпосередньо через регістри.

Для цього потрібно виконати наступні кроки:

1. Відкрити бібліотеку lib_AT91SAM7X256.h, двічі клікнувши по ній у вікні Project Workspace.
2. Активувати функцію пошуку та знайти строку з назвою даної функції (AT91F_PIO_CfgOutput). Результат пошуку повинен збігатися із рис. 2.4.

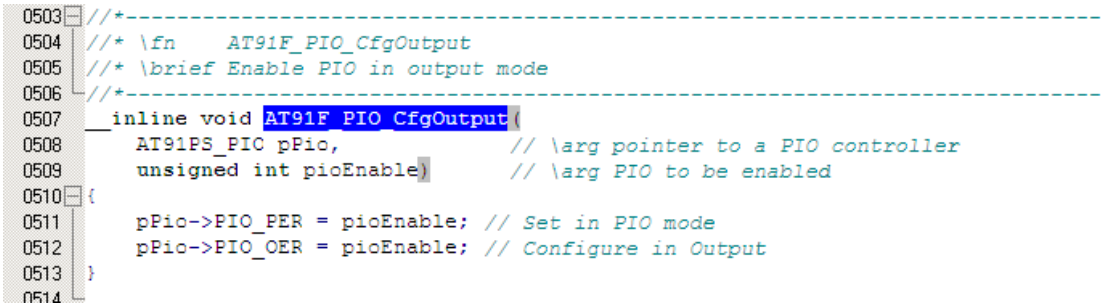


Рис. 2.4. Пошук функції у бібліотеці

Після пошуку та ознайомлення із тілом функції AT91F_PIO_CfgOutput () видно, що дана функція виконується з двома параметрами: AT91PS_PIO pPio та unsigned int pioEnable. Перший параметр відповідає за вибір PIOA або PIOB, другий – за те, які лінії будуть сконфігуровані. Функція записує значення у два регістри: PIO_PER – регістр активації лінії PIO контролера, та PIO_OER - вказує які лінії будуть сконфігуровані на вихід.

Отже записи

AT91F_PIO_CfgOutput(AT91C_BASE_PIOB, AT91B_LED_MASK)

та

AT91C_BASE_PIOB->PIO_PER=AT91B_LED_MASK;

AT91C_BASE_PIOB->PIO_OER=AT91B_LED_MASK;

будуть еквівалентними.

Далі у рядку

AT91F_PIO_SetOutput(AT91C_BASE_PIOB, AT91B_LED_MASK)

виконується встановлення на лінії №19-22 PIOB контролера логічних 1.

Для перевірки коректності коду використаємо відладчик (Debugger) середовища Keil uVision. Для цього, після збирання проєкту (впевнившись, що помилки відсутні), для запуску відладчика необхідно натиснути кнопку (Debugger) або Ctrl+F5. Після відкриття операційного простору відладчика необхідно відкрити меню Periferals -> Parallel I/O Controller-> PIOB. І натиснути на кнопку (Run) для запуску програми. Реакція відладчика на коректний код має виглядати як показано на рис. 2.5.

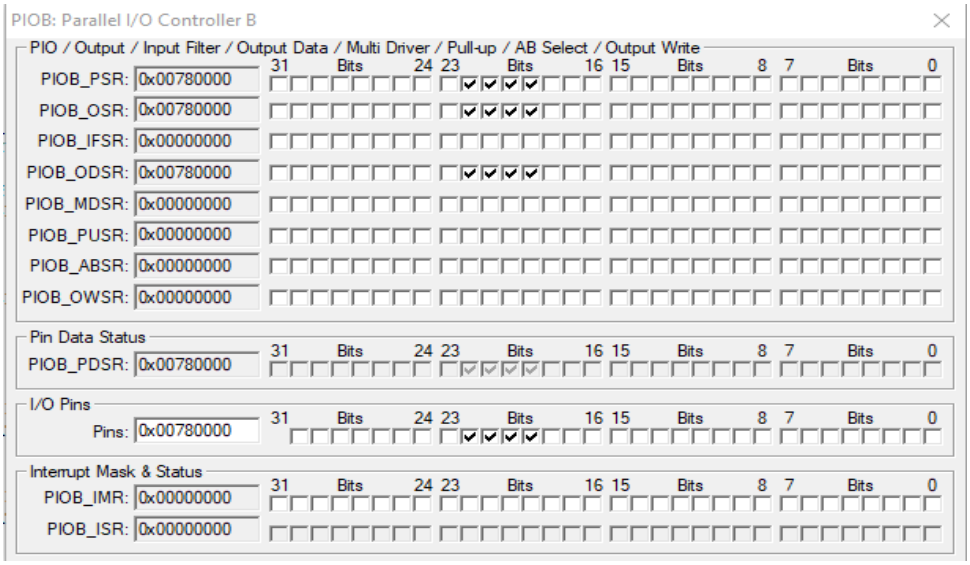


Рис. 2.5. Результат роботи програми у відладчику

Необхідно зазначити, що з врахуванням того, що світлодіоди підключені через інвертор, то при подальшому завантаженні даного коду на МК світлодіоди будуть виключені.

2.2.3. Порядок виконання лабораторної роботи

1. Запустити Keil uVision

2. Створити та зберегти новий проект.
3. Скопіювати у директорію проекту файл AT91SAM7X-EK.h.
4. Надрукувати текст програми у файлі з розширенням *.c.
5. Прочитати та розібратися у тексті програми.
6. Натиснути на кнопку  (Rebuild All Rargets) для підключення всіх бібліотек.
7. Перевірити правильність роботи програми у відладчику.
8. Змінити значення AT91B_LED_MASK на значення лінії, що відповідає номеру варіанта по списку.
9. Замінити функцію AT91F_PIO_SetOutput () на AT91F_PIO_ClearOutput() з тими ж параметрами, зробити висновки.
10. Створити другий проект, в якому буде відсутня бібліотека lib_AT91SAM7X256.h та використана форма запису через регістри, проте функціонал повинен залишатись не змінним.

2.3. Зміст звіту

1. Код програми, використовуючи функції і запис через регістри.
2. Фото результату відладки (перенести за допомогою Print Screen).
3. Загальні висновки по роботі.

2.4. Контрольні запитання

1. Яке призначення PIO контролера?
2. У чому різниця між PIOA і PIOB контролерами?
3. За допомогою якої функції лінії конфігуруються на вивід/як виходи?
4. За допомогою якої функції можна встановити логічний “0” на лінії?
5. За допомогою якої функції можна встановити логічну “1” на лінії?
6. Яке призначення регістрів PIO_PER та PIO_OER?
7. Які регістри використовують функції AT91F_PIO_SetOutput () та AT91F_PIO_ClearOutput(), яке їхнє призначення?
8. Як можна перевірити правильність роботи програми?

ЛАБОРАТОРНА РОБОТА № 3.
КЕРУВАННЯ ПЕРИФЕРІЙНИМИ СТРУКТУРАМИ ВІДЛАДОЧНОГО МОДУЛЯ. ПРОГРАМА «ГІРЛЯНДА»

Мета роботи: Закріпити навички використання регістрів PIO контролера. Навчитись завантажувати код у Flash-пам'ять МК різними способами.

3.1. Теоретичні відомості

3.1.1. Програмування МК за допомогою універсального програматора SEGGER

Для завантаження готового коду на відладочну плату існує два алгоритми дій. Перший спосіб потребує наявності програматора, другий – використовує модуль завантажувальник SAM-BA.

Перший варіант передбачає наступне: підключити модуль програматора до відладочної плати, використовуючи інтерфейс JTAG (див. рис. 3.1).

За допомогою двох USB кабелів підключити програматор та відладочну плату до ПК.

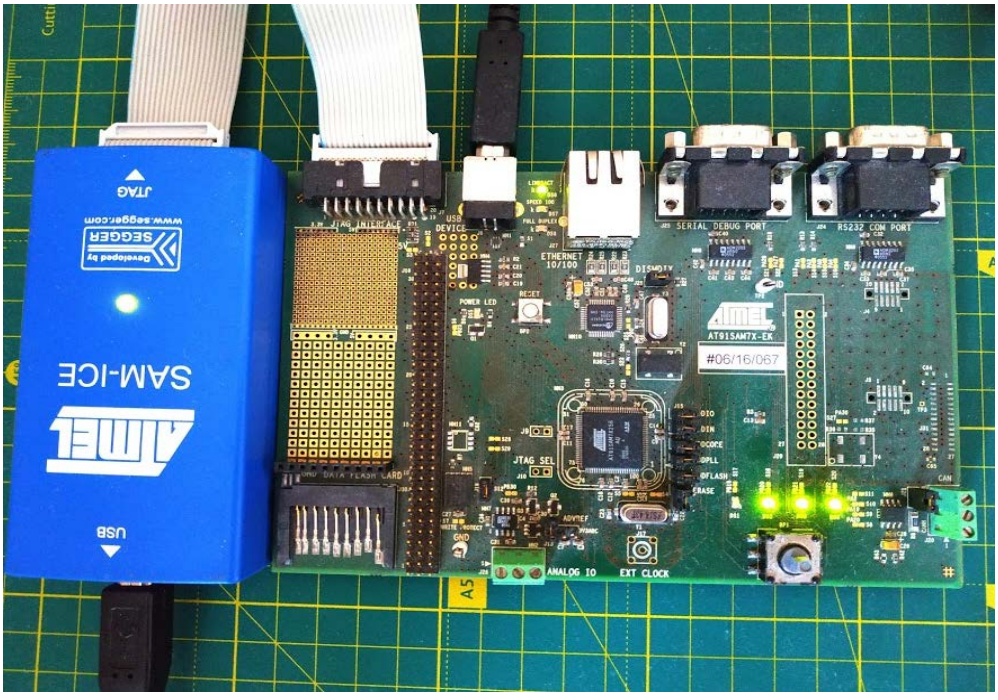


Рис. 3.1. Підключення відладочної плати і програматора

При першому ввімкненні впевнитись, що проінстальовані драйвери програматора, якщо ні — провести інсталяцію, запустивши файл JLink_v3.exe (доступний у мережевому оточенні в комп'ютерному класі).

Із випадаючого списку режимів (їх створення і налагодження описане у лабораторній роботі №1) 

обираємо режим Flash.

Після цього, необхідно зберегти і перезібрати остаточну версію проекту, натиснувши на кнопку  (Rebuild All Rargets). Якщо помилок немає, натиснути на кнопку  Download з меню Flash. Якщо середовище не вивело написи з помилками, то це означає, що програма успішно завантажена у Flash-пам'ять МК та готова для виконання. Для її запуску необхідно повторно подати живлення на плату (перепідключивши кабель USB) або натиснути на відладочній платі кнопку Reset.

Для отримання можливості керування завантаженою програмою після збирання проекту, запустимо відладчик. Для цього необхідно натиснути кнопку  (Debugger) або Ctrl+F5, і натиснути на кнопку  (Run) для запуску програми.

3.1.2. Програмування МК за допомогою модуля SAM-BA

Другий варіант завантаження використовується коли програматор відсутній. Для нього необхідно додатково підготувати як відладочну плату, так і проект у середовищі Keil uVision.

Спочатку у папку проекту необхідно скопіювати файл fromelf.exe (доступний у мережевому оточенні). Це програма конвертор, що сформує бінарний файл, який буде завантажений у пам'ять МК.

Далі у середовищі Keil uVision у вкладці Project вибираємо команду 

та переходимо у вкладку User, далі у строці, що показана на рис. 3.2, вводимо наступну команду: fromelf --bin -o XXX.bin YYY.axf. Де XXX – ім'я вихідного бінарного файлу, наприклад, out.bin, а YYY - ім'я

наявного *.axf файлу із папки проекту. Також необхідно поставити маркер в полі Run #1, як показано на рис. 3.2.

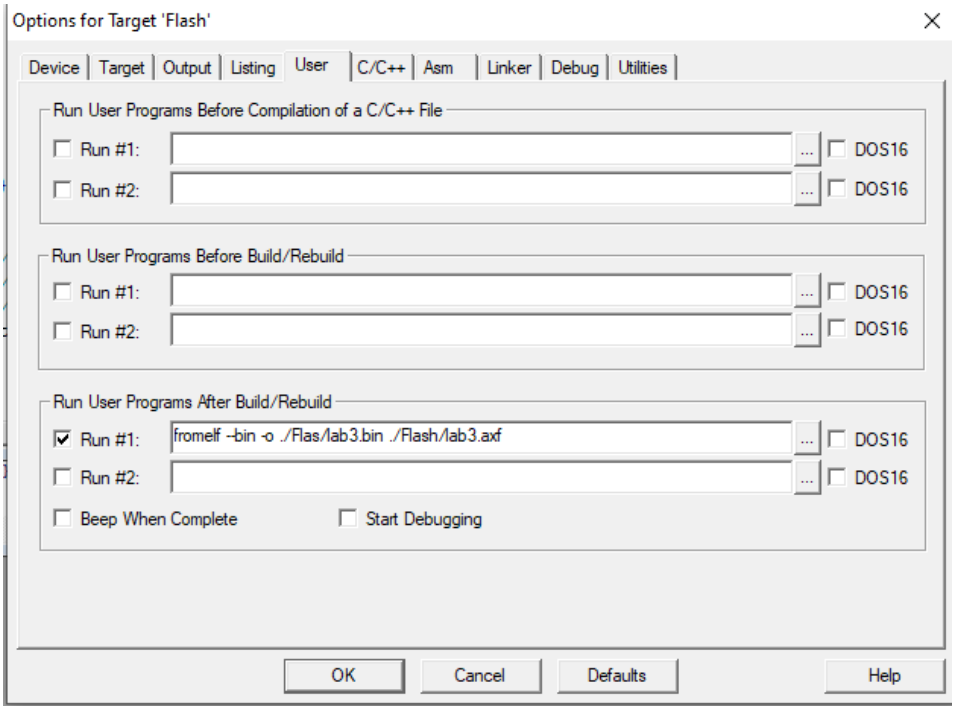
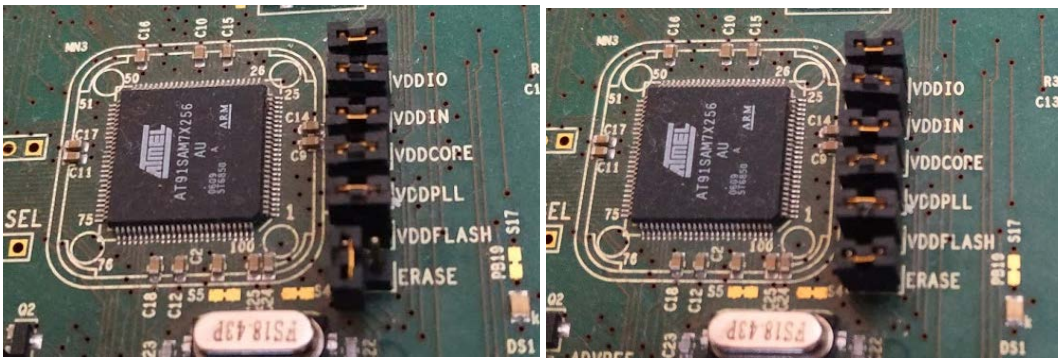


Рис. 3.2. Налаштування проекту, меню User

Перевірити чи сформувався у папці проекту файл із розширенням *.bin.

Наступним кроком необхідно підготувати Flash-пам’ять МК для запису. Для цього, при підключеному живленні плати, необхідно замкнути перемичку Erase, як показано на рис. 3.3. Після цього відключити живлення і зняти перемичку.



а) б)

Рис. 3.3. Положення перемички Erase:

а) Перемичка розімкнена; б) Перемичка замкнута

Запустити програму Sam-Ba_2_16.exe (в залежності від операційної системи, версії можуть змінюватись) і, обравши at91sam7x-ek, натиснути Connect (рис. 3.4.).

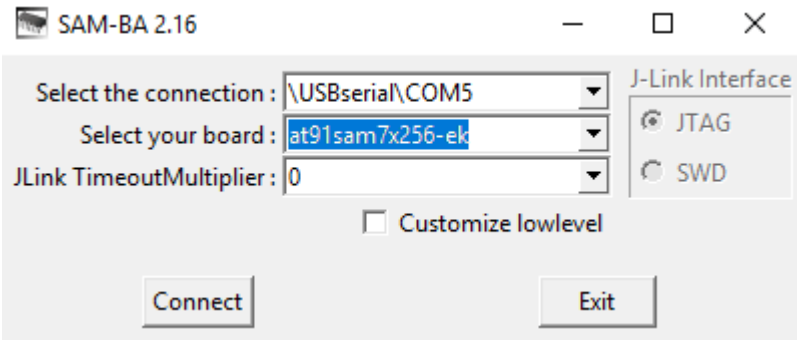


Рис. 3.4. Меню запуску SAM-BA

Загальний вигляд вікна SAM-BA показано на рис. 3.5.

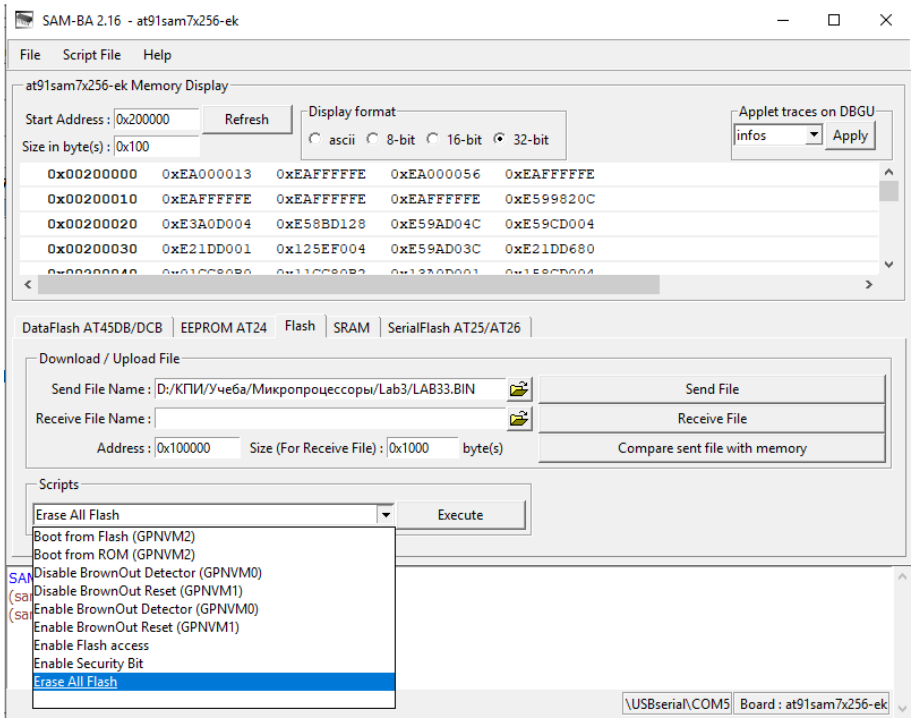


Рис. 3.5. Головне вікно SAM-BA

У відкрившомуся вікні із меню Scripts необхідно обрати Erase All Flash та натиснути Execute. Ця операція проводить попередню очистку секторів Flash-пам’яті для запису нової програми.

Далі обираємо Boot From Flash та Execute. Ці операції гарантовано очистять пам’ять МК і вкажуть, яку саме пам’ять використовувати для завантаження програми.

У меню Send File Name обрати створений бінарний файл і натиснути Send File.

Для запуску програми на МК на відладочній платі необхідно натиснути Reset.

3.2. Завдання до виконання

3.2.1. Принцип функціонування

Після подачі живлення на плату, відбувається почергове звернення до ніжок PIOB контролера, до яких приєднані вбудовані світлодіоди. У результаті світлодіоди із 1 по 4 почергово запалюються та гаснуть.

3.2.2. Код програми

З врахуванням того, що реалізація даного завдання не потребує використання нових функцій або регістрів, які стосуються МК або PIO-контролера, то код створюється власноручно.

Для спрощення створення коду наведемо найпростіший код, який буде реалізовувати затримку для можливості спостереження за блиманням світлодіодів. Реалізуємо функцію wait (void), що буде використана у основній програмі.

```
void wait (void)
{
  unsigned int n;
  for (n = 0; n < AT91B_MAIN_OSC/20; n++);
}
```

AT91B_MAIN_OSC – частота головного кварцевого резонатора, що розміщений на платі. Вона дорівнює 18.43 МГц. Данна величина в циклі взята

для прикладу і може бути змінена власноручно, використовується для керування швидкістю блимання.

3.2.3. Додаткове завдання

- До гребінки ніжок PIOB контролера підключити три додаткові світлодіоди (використовувати макетні плати). Номер ніжок визначається за наступною формулою: 22+ № варіанту+№ світлодіода-1. Тобто, наприклад, для 2-го варіанту світлодіоди слід підключити до ліній 24, 25 та 26. Якщо значення більше ніж 32, то переходимо до лінії №1. Схема підключення світлодіодів приведена на рис. 3.6., а загальний вигляд на рис. 3.7.

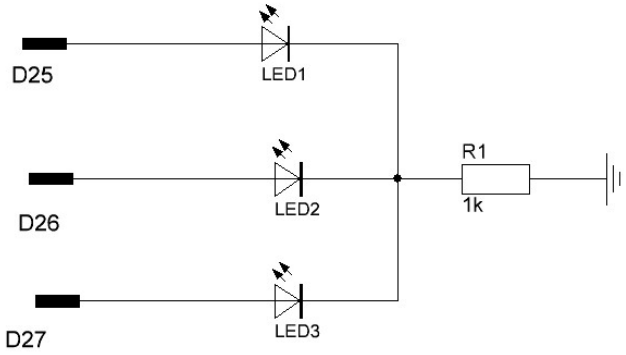


Рис. 3.6. Схема підключення додаткових світлодіодів

Зазначимо, що нумерація гребінок на платі починається із номера 1, а значення розрядів в регістрах з 0. Це треба враховувати при написанні коду!

- Додати підключені світлодіоди у загальну послідовність «Гірлянди».
- Замінити всі функції безпосередньо на записи через регістри та впевнитись в працездатності програми.

3.3. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Створити код програми у головному файлі (з розширенням *.c).
3. Перевірити правильність роботи програми у відладчику.

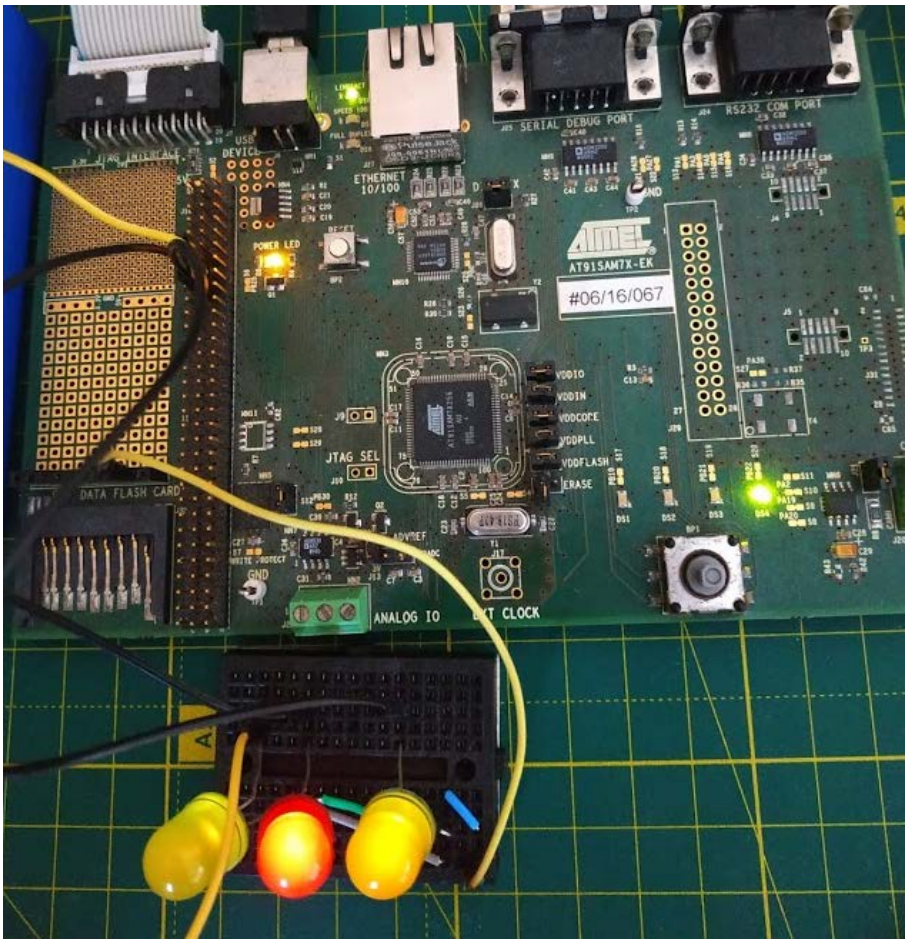


Рис. 3.7. Відладочна плата з підключеними додатковими світлодіодами

4. Оптимізувати код програми, уникаючи повторюваних частин коду (використовувати цикли).
5. Завантажити програму в МК двома способами.
6. Створити другий проект, в якому буде відсутня бібліотека lib_AT91SAM7X256.h та використана форма запису через регістри.

3.4. Зміст звіту

1. Код програми, використовуючи функції і запис через регістри.
2. Фото результату відладки.
3. Висновки.

3.5. Контрольні запитання

1. Чим відрізняється код звернення для зовнішніх світлодіодів та світлодіодів, що розміщені на відладочній платі? Як це пояснити?
2. За допомогою якої функції лінії конфігуруються на вихід, навіщо це робити?
3. Скільки периферійних пристроїв можна підключити до цифрових ліній вводу/виводу?
4. Як можна перевірити правильність роботи програми?
5. У чому відмінності двох наведених способів програмування МК, які їх недоліки та переваги?

ЛАБОРАТОРНА РОБОТА № 4.
КОНТРОЛЕР ЖИВЛЕННЯ (PMC)

Мета роботи: Ознайомитись з особливостями використання регістрів контролера живлення (PMC), на прикладі операції зчитування стану ліній вводу/виводу.

4.1. Теоретичні відомості

4.1.1. Контролер живлення (PMC)

Контролер живлення (PMC – power management controller) — один із периферійних модулів мікроконтролера AT91SAM7X256, який може керувати споживаною мікросхемою енергією за рахунок тактових синхросигналів. Інакше кажучи, PMC керує всіма системними і користувальницькими периферійними тактовими сигналами, керує (підключає або відключає) входами тактових сигналів для багатьох периферійних пристроїв і ядра процесора ARM. Контролер живлення забезпечує керування наступними синхросигналами:

• Задаюча тактова частота MCK (master clock), значення якої може бути встановлено від 100 Гц до максимальної тактової частоти ядра мікросхеми. Ця частота подається на модулі, що функціонують постійно, наприклад, контролер пам'яті.

• Тактова частота процесора PCK (processor clock). Може бути відключена при переході мікросхеми в режим очікування.

• Периферійна тактова частота, що дорівнює, як правило, MCK та забезпечує тактування вбудованих периферійних пристроїв (USART, SPI, TWI, і т. д.) і керується незалежно.

• Частота UDP (UDPCK), необхідна для тактування порту USB-пристроїв.

• Програмно керовані виходи синхронізації, які можуть бути обрані з джерел синхронізації тактового генератора.

4.1.2. Контролер тактування периферії

Контролер живлення керує тактовими сигналами кожного вбудованого периферійного пристрою МК за допомогою контролера тактування периферії. Користувач може індивідуально ввімкнути або вимкнути задану частоту для всіх периферійних пристроїв шляхом запису в регістр дозволу тактування периферії (PMC_PCER) і регістр заборони тактування периферії (PMC_PCDR). Які з периферійних пристроїв тактовані, можна дізнатися, опитавши регістр статусу тактування периферії – PMC_PCSK.

Коли тактова частота периферійного пристрою заблокована, тактування і, відповідно, функціонування периферії негайно зупиняються. Тактова частота периферійних модулів МК за замовчуванням заблокована.

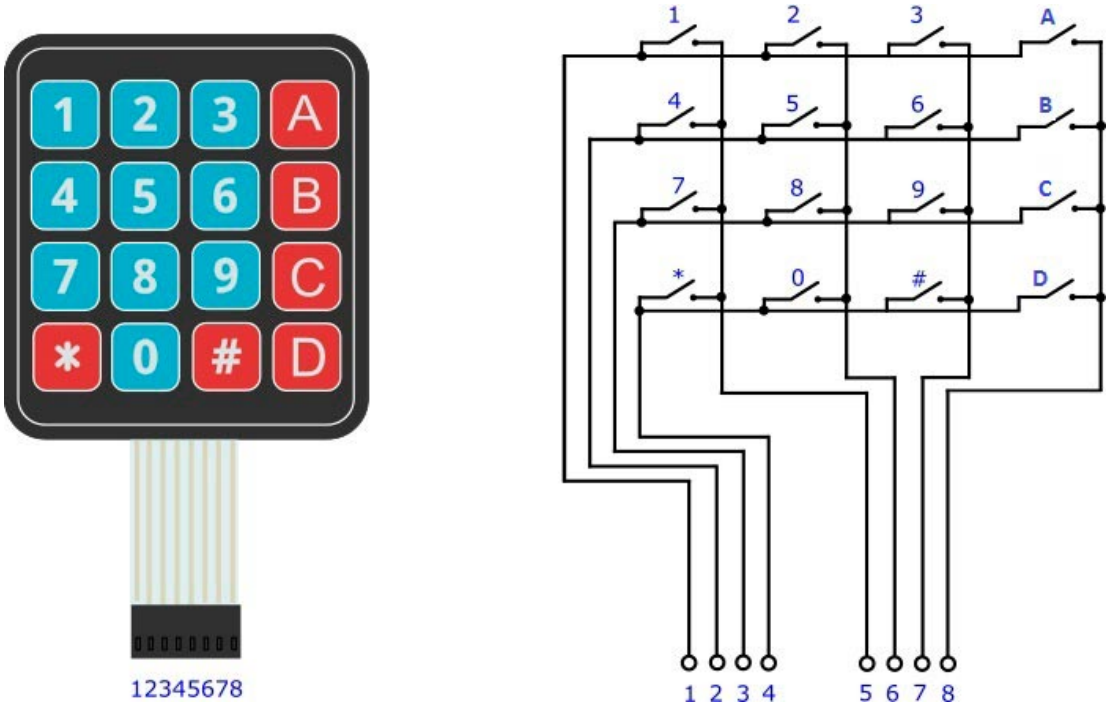
Для проведення коректної зупинки периферійного модуля, рекомендується програмне очікування завершення його останньої операції перед відключенням тактування. Це дозволить уникнути порушення цілісності даних або не коректної роботи системи.

Бітовий номер, що міститься в регістрах керування тактуванням периферії (PMC_PCER, PMC_PCDR і PMC_PCSR), називається ідентифікатором периферійного пристрою. Значення цих ідентифікаторів визначаються на етапі виготовлення мікросхеми і не можуть бути програмно змінені. У загальному випадку для кожного периферійного пристрою цей бітовий номер відповідає номеру джерела переривання, призначеного для цього периферійного пристрою.

4.1.3. Матрична клавіатура

У даній лабораторній роботі, для формування логічних рівнів на лініях вводу/виводу, будемо використовувати матричну клавіатуру. Матрична або

мембранна клавіатура – простий пристрій для тактильного вводу даних, що представляє собою матрицю вимикачів, виконаних на єдиній основі. Розміри матриць можуть відрізнятись в залежності від сфер застосування, у даній роботі будемо використовувати клавіатури розміром 3x4 або 4x4. Загальний вигляд матричної клавіатури 4x4 та її електрична схема показані на рис. 4.1.



а) б)

Рис. 4.1. Матрична клавіатура 4x4

а) Загальний вигляд; б) електрична схема

4.2. Завдання до виконання

4.2.1. Принцип функціонування програми

При підключенні живлення до відладочної плати всі 4 вбудовані світлодіоди вимкнені. При натисканні на першу позицію джойстика (вгору), відбувається ввімкнення першого світлодіода. Світлодіод повинен бути ввімкненим доки утримується джойстик. При натисканні на другу позицію джойстика – запалюється другий світлодіод, і т.д. При натисканні п’ятої позиції джойстика (центральне положення) всі вбудовані світлодіоди світяться.

4.2.2. Код програми

Код програми приведений лише для роботи із першим положенням джойстика і, відповідно одним світлодіодом, оскільки принцип роботи із іншими положеннями джойстика – такий самий.

```
#include <AT91SAM7X256.H>
#include <lib_AT91SAM7X256.h>
#include "AT91SAM7X-EK.h"

	int main (void) {
// Ввімкнення тактування периферії
AT91F_PMC_EnablePeriphClock(AT91C_BASE_PMC, 1 << AT91C_ID_PIOA);
// Конфігурація необхідних ліній PIO
AT91F_PIO_CfgOutput(AT91C_BASE_PIOB, AT91B_LED_MASK);
	for(;;)
{
//Вимикаємо світлодіоди
AT91F_PIO_SetOutput(AT91C_BASE_PIOB, AT91B_LED_MASK);
// Опитуємо лінії PIOA
if ((AT91F_PIO_GetInput(AT91C_BASE_PIOA) & AT91B_SW1) == 0)
AT91F_PIO_ClearOutput(AT91C_BASE_PIOB, AT91B_LED1);
} }
```

4.2.3. Опис нових функцій

AT91F_PMC_EnablePeriphClock() – функція ввімкнення тактування периферії.

Зауважимо, що опитування ліній вводу / виводу може здійснюватися лише тоді, коли дозволено подачу на PIO контролер синхроімпульсів від контролера живлення PMC.

AT91F_PIO_GetInput () – функція, що вичитує всі значення 32 бітного регістра PIO_PDSR (регістр статуса вхідних даних). Тобто, ми отримуємо статус одразу всіх 32 ліній PIO контролера. Для виокремлення необхідного біта, що відповідає за статус певної ніжки, використаний оператор бітового множення “&”.

Отже, строка

```
if ((AT91F_PIO_GetInput(AT91C_BASE_PIOA) & AT91B_SW1) == 0)
```

слідкує за тим, чи нажате перше положення джойстика. Зауважимо, що при натисканні на певне положення джойстика, на відповідній йому лінії з’являється логічний 0 (див. рис. 4.2).

4.2.4. Додаткове завдання

1) Підключити до гребінки виходів PIOA контролера матричну клавіатуру. Необхідно задіяти мінімум 4 клавіші.

Номер виводів ліній для підключення клавіатури визначається за наступною формулою: № варіанту+№ клавіші-1. Тобто, наприклад для 1-го варіанту клавіатуру слід підключити до ліній PIOA № 1, 2, 3 та 4 (Нагадування! Приєднати загальний дріт до контакту GND). Якщо значення більше ніж 20, то переходимо до лінії №1. Схема комутації показана на рис. 4.2.

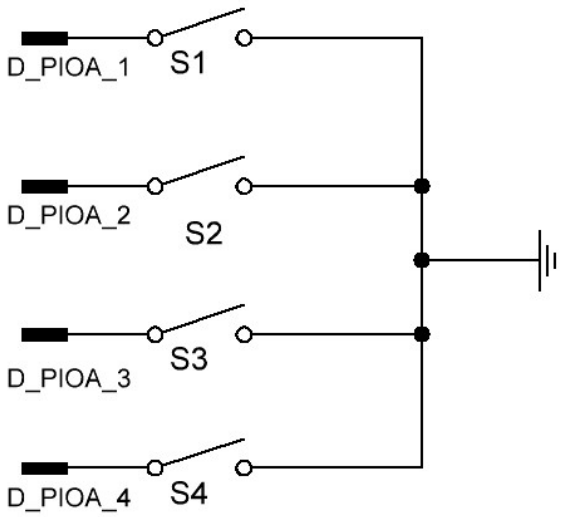


Рис. 4.2. Схема підключення додаткових кнопок матричної клавіатури

До гребінки ніжок PIOB контролера підключити три додаткові світлодіоди (використовувати макетні плати). Номер ніжок визначається за наступною формулою: 22+ № варіанту+№ світлодіода-1. Додати підключені модулі у загальну послідовність роботи програми, відповідно змінивши її код.

2) *Для непарних варіантів.* Виконати керування вбудованими світлодіодами за допомогою підключеної клавіатури, та джойстиком – підключеними зовнішніми світлодіодами.

Для парних варіантів. Виконати керування вбудованими світлодіодами за допомогою джойстика, а підключеними зовнішніми світлодіодами – за допомогою підключеної клавіатури.

Загальний принцип залишається таким, як наведено в п.п. 4.2.1.

3) *Для непарних варіантів.* Підключити клавіатуру та світлодіоди лише до виходів PIOB контролера, не змінюючи загальний алгоритм роботи програми.

Для парних варіантів. Підключити клавіатуру та світлодіоди лише до виходів PIOA контролера, не змінюючи загальний алгоритм роботи програми.

Оптимізувати код програми, застосувавши навички програмування.

Замінити всі функції безпосередньо на записи через регістри та впевнитись в працездатності програми.

4.3. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Створити код програми у головному файлі (з розширенням *.c).
3. Перевірити правильність роботи програми у відладчику.
4. Оптимізувати код програми, уникаючи повторюваних частин коду (використовувати цикли).
5. Завантажити програму на МК двома способами.

6. Створити другий проект, в якому буде відсутня бібліотека lib_AT91SAM7X256.h та використана форма запису через регістри.

4.4. Зміст звіту

1. Код програми використовуючи функції і запис через регістри.
2. Фото результату відладки.
3. Висновки.

4.5. Контрольні запитання

1. Основні функції контролера живлення?
2. Як пов’язано тактування та споживана потужність?
3. Що таке тактування периферії?
4. Навіщо застосовувати функцію AT91F_PMC_EnablePeriphClock()?
5. Чому функцію AT91F_PMC_EnablePeriphClock() не використовували у минулих роботах?
6. Чому функція AT91F_PIO_GetInput має лише один аргумент?
7. Чи можна взагалі не використовувати бібліотеки відладочної плати та функцій, як зміниться код програми?

ЛАБОРАТОРНА РОБОТА №5. СТВОРЕННЯ КЕРОВАНИХ ЧАСОВИХ ПОСЛІДОВНОСТЕЙ ТА КЕРУВАННЯ ПОСЛІДОВНОСТЯМИ ВИХІДНИХ ЦИФРОВИХ СИГНАЛІВ. ПРОГРАМА «ГІРЛЯНДА 2».

Мета роботи: Вдосконалити навички програмування у середовищі Keil uVision щодо роботи з контролером тактування периферії, циклами та умовами. Створення програмного забезпечення із змінним значенням часу затримок між операціями.

5.1. Завдання до виконання

5.1.1. Принцип функціонування

При поданні живлення на плату, спочатку функціонування програми збігається із програмою «Гірлянда». Тобто, бачимо послідовне блимання світлодіодів 1-2-3-4-3-2. При натисканні на першу позицію джойстика (вгору), відбувається прискорення блимання світлодіодів. При натисканні другої позиції джойстика (до низу) – уповільнення. При натисканні на п’яту позицію (центральне положення) – зупинка поточної позиції. Максимальна і мінімальна швидкості блимання повинні бути обмежені, ці значення слід підібрати експериментально.

5.1.2. Код програми

З врахуванням того, що використання нових функцій або регістрів, які стосуються МК, реалізація даного завдання не потребує, то код створюється власноручно.

Для спрощення створення коду, наведемо найпростіший код, що буде реалізовувати зупинку послідовності блимання «гірлянди». Код повинен бути реалізований у функції керування блиманням світлодіодів.

Також додамо, що мінімальне і максимальне значення затримок повинні бути обмежені.

```
while (((AT91F_PIO_GetInput(AT91C_BASE_PIOA) & AT91B_SW5) == 0));
```

Принцип роботи наступний: поки натиснене центральне положення джойстика (AT91B_SW5) цикл while продовжує роботу і, як наслідок, не дозволяє перейти до наступного етапу блимання «гірлянди».

5.1.3. Додаткове завдання

Підключити до гребінки виходів PIOA контролера додаткову клавіатуру (3x4 або 4x4), задіяними мають бути мінімум 3 клавіші клавіатури, а до виходів PIOB контролера 3 додаткові світлодіоди.

Принцип підключення, номери ліній і т.д. описані у попередніх лабораторних роботах і залишаються не змінними.

Додати підключені модулі у загальну послідовність роботи програми, відповідно змінивши її код.

Клавіатура і джойстик повинні працювати однаково. Тобто при натисканні на кнопку 1 клавіатури, відбувається прискорення блимання (як і при натисканні на першу позицію джойстика).

Для парних варіантів:

а) При натисканні на обрану кнопку матричної клавіатури або незадіяну позицію джойстика, послідовність «гірлянди» зупиняється, а всі задіяні світлодіоди мають передавати сигнал «SOS» доки утримується кнопка. Після цього послідовність «гірлянди» продовжується.

б) Підключити клавіатуру та світлодіоди лише до виходів PIOB контролера, не змінюючи загальний алгоритм роботи програми.

Для непарних варіантів:

а) При натисканні на обрану кнопку клавіатури або незадіяну позицію джойстика, всі світлодіоди (і на платі, і підключені) загоряються. Далі відбувається почергове виключення світлодіодів з кроком приблизно 1с, поки не вимкнуться всі. Після цього послідовність «гірлянди» продовжується.

б) Підключити клавіатуру та світлодіоди лише до виходів PIOA контролера, не змінюючи загальний алгоритм роботи програми.

5.2. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Створити код програми у головному файлі (з розшименням *.c)
3. Перевірити правильність роботи програми у відладчику.
4. Оптимізувати код програми, уникаючи повторюваних частин коду (використовувати цикли).
5. Завантажити програму в МК двома способами.
6. Створити другий проект, в якому буде відсутня бібліотека lib_AT91SAM7X256.h та використана форма запису через регістри.

5.3. Зміст звіту

1. Код програми використовуючи функції і запис через регістри.
2. Фото результату відладки.
3. Висновки.

5.4. Контрольні запитання

1. Як більш точно визначити час затримки?
2. Як додати функцію затримки до бібліотеки?
3. Чи можна керувати лінією до якої підключений світлодіод індикації живлення плати?
4. У яких випадках можна не застосовувати функцію AT91F_PMC_ Enable PeriphClock ()?

ЛАБОРАТОРНА РОБОТА №6. ВИКОРИСТАННЯ ТАЙМЕРА/ЛІЧИЛЬНИКА. ЗНАЙОМСТВО ІЗ СВІТЛОДІОДНИМИ ІНДИКАТОРАМИ

Мета роботи: На прикладі створення пристроїв відліку часу ознайомитись із роботою таймера/лічильника для керування сигналами.

6.1. Теоретичні відомості

6.1.1. Таймер/лічильник

Таймер/лічильник (Timer-counter TC) призначений для виконання широкого діапазону функцій, таких як вимірювання частоти, відлік та вимірювання інтервалів між подіями, генерація імпульсів, формування часових затримок і широтно-імпульсна модуляція.

Таймер/лічильник містить три ідентичних 16-бітних канали. Кожен канал можна програмувати незалежно від інших. У свою чергу, кожний канал має три зовнішніх тактових входи, п'ять внутрішніх тактових входів і два багатофункціональних вхідних / вихідних сигнали, які можуть бути сконфігуровані користувачем. Кожен канал керує внутрішнім сигналом переривання, який використовується для генерації переривань процесора.

Блок TC має два глобальних регістра, які пов’язані із всіма трьома каналами, це:

Block Control Register – однією інструкцією може дозволити запуск всіх каналів одночасно.

Block Mode Register – визначає зовнішні джерела тактування для кожного каналу, дозволяючи поєднувати їх в ланцюжок.

Структурна схема таймера/лічильника представлена на рис. 6.1.

Зазначимо декілька ключових моментів. Так як TC тактується через контролер живлення (PMC), тому для коректної роботи таймера/лічильника

необхідно спочатку сконфігурувати PMC, щоб дозволити подачу тактових синхроімпульсів на TC.

Виводи, які використовуються для підключення зовнішніх пристроїв, можуть бути мультиплексовані з лініями вводу/виводу, а отже, для цього спочатку необхідно запрограмувати контролери PIO, щоб призначити виводи таймера/лічильника на виконання їх периферійних функцій.

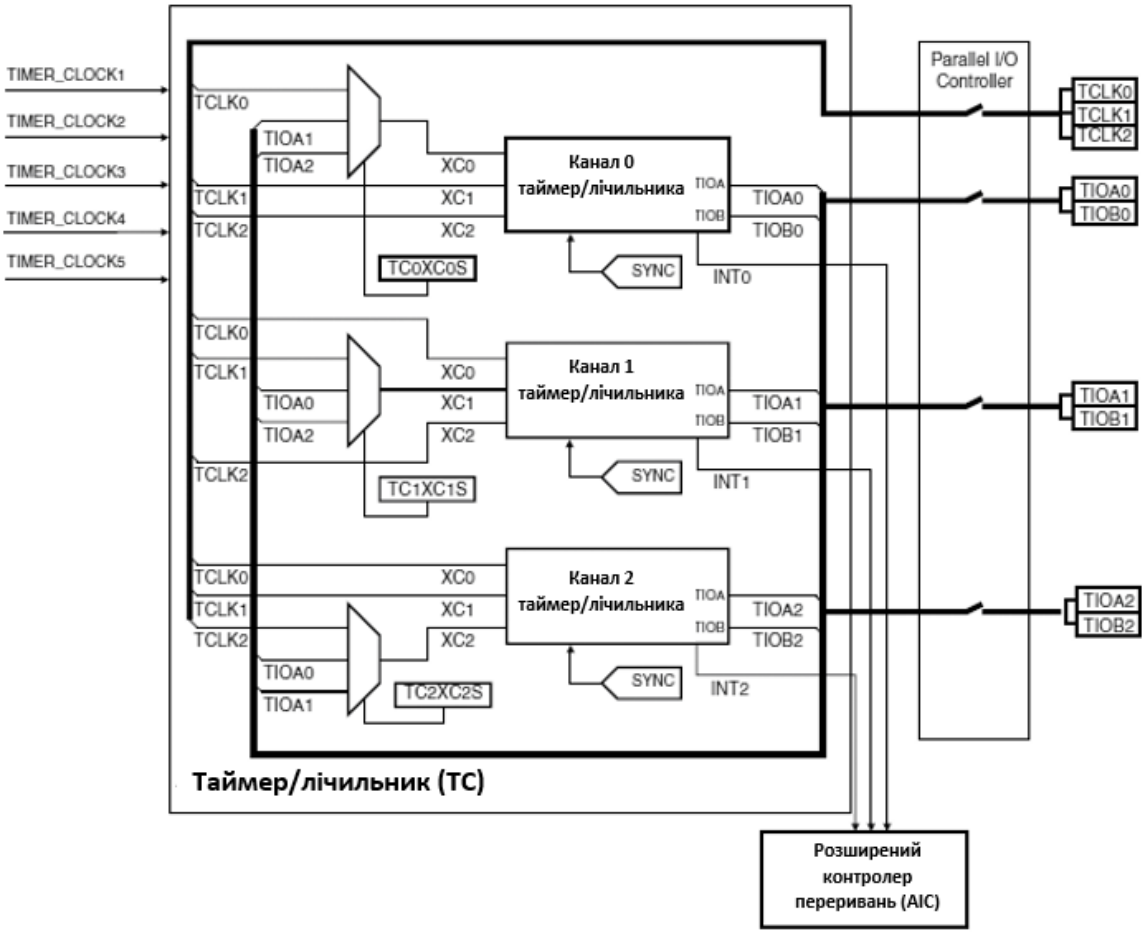


Рис. 6.1. Структурна схема таймера/лічильника

Так як структура TC відносно складна, то студентам пропонується використовувати створену бібліотеку **delay.** Для цього у папку проекту необхідно додати файли delay.h та delay.c.

Бібліотека дає можливість використовувати дві функції, прототипи яких void delay_config(void); - для конфігурації таймера/лічильника.

Та void delay (int millis); - безпосередньо реалізує затримку на вказану у дужках кількість мілісекунд. Максимальне значення затримки – 1 с.

Далі, для прикладу, наведемо лістинг коду (рис. 6.2.), що реалізує блимання світлодіода із частотою 10 Гц, тобто із періодом 100 мс.

```
01
02  #include <AT91SAM7X256.H>
03  #include <lib_AT91SAM7X256.h>
04  #include "AT91SAM7X-EK.h"
05  #include "delay.h"
06
07  int main (void) {
08  int i= 100U;
09  AT91F_PIO_CfgOutput(AT91C_BASE_PIOB, AT91B_LED1);
10  delay_config();
11
12  for(;;)
13  {
14  AT91F_PIO_ClearOutput(AT91C_BASE_PIOB, AT91B_LED1);
15  delay(i);
16  AT91F_PIO_SetOutput(AT91C_BASE_PIOB, AT91B_LED1);
17  delay(i);
18  }
19  }
20
21
22
23
```

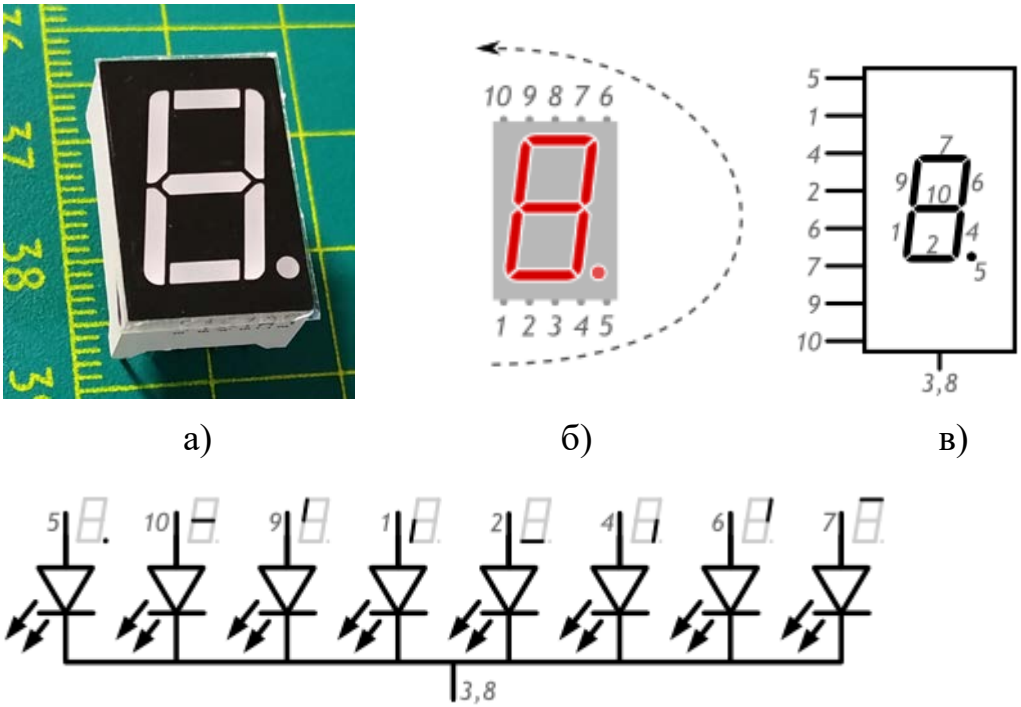
Рис. 6.2. Приклад використання функцій бібліотеки *delay*

6.1.2. Семисегментний індикатор

Семисегментний індикатор – простий світлодіодний пристрій, призначений для формування зображення у вигляді арабських цифр та деяких літер. Семисегментні індикатори розрізняються за розмірами, розрядністю, схемою включення, проте принцип роботи залишається однаковим. Це 7 світлодіодів виконаних у вигляді смуг, які розміщені таким чином, щоб формувати зображення символу. Додатково присутній світлодіод у вигляді десяткової крапки, що використовується для формування зображень чисел більшої розрядності.

У роботі будемо використовувати однорозрядний червоний індикатор з спільним катодом, що представлено на рис.6.3.

Індикатор має 10 виводів, нумерація яких іде проти годинникової стрілки, що показано на рис. 6.3. б. На рис. 6.3.в та рис. 6.3.г представлені принципові схеми, які необхідні для підключення індикатора до плати.



а) б) в)

г)

Рис. 6.3. Семисегментний індикатор: а) Загальний вигляд; б) Напрям рахунку виводів; в) відповідність виводів сегментам індикатора; г) принципова схема.

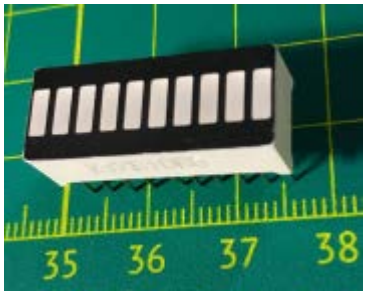
Також зауважимо, що для підключення семисегментного індикатора до виводів мікроконтролера, необхідно використовувати струмообмежувальні резистори номіналом від 200 Ом до 1 КОм.

6.1.3. 10 сегментний світлодіодний індикатор рівня

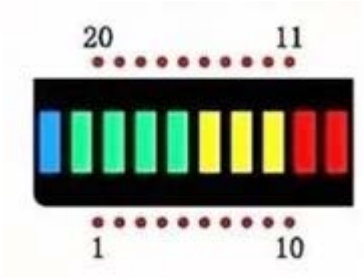
Дані індикатори призначені для виведення візуальної інформації для відображень гістограми, рівня, послідовності, відліку часу, т. і. 10 сегментний світлодіодний індикатор представлений на рис. 6.4.

По суті, такий індикатор представляє собою 10 незалежних світлодіодів, розміщених в одному корпусі. Як і для семисегментного індикатора, для

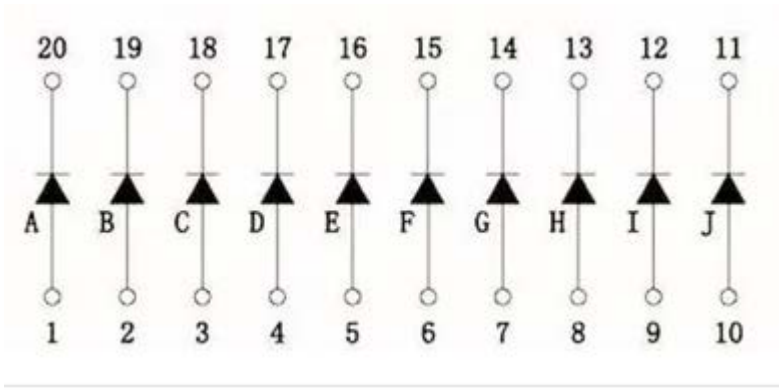
підключення до мікроконтролера необхідно використовувати струмообмежувальні резистори.



а)



б)



в)

Рис. 6.4. 10-сегментний індикатор:
а) Загальний вигляд; б) відповідність виводів сегментам індикатора;
в) принципова схема.

6.1.4. П’єзодинамік

П'єзокерамічні випромінювачі (п’єзодинаміки) - електроакустичні пристрої відтворення звуку. Принцип роботи полягає у наступному: коли на п’єзолектричний кристал подається напруга, він змінює свої розміри, при поданні напруги певної частоти, мембраною, що поєднана із кристалом, – генерується звук. У лабораторній роботі представлені п’єзодинаміки двох видів — активні та пасивні. Активний п‘єзодинамік має вбудовану систему збудження. Це означає, що для генерації звуку достатньо подати рівень логічної одиниці від мікроконтролера. Пасивний п’єзодинамік, для генерації звуку, потребує постійної зміни логічного рівня з певною частотою. Не

зважаючи на складність керування, перевагою пасивного п’єзодинаміка є можливість змінювати тональність в залежності від частоти сигналу.

6.2. Завдання до виконання

Для парних варіантів:

Семисегментний індикатор підключений до ліній PIOB контролера за схемою, що описується таблицею 1.

Таблиця 1.

Послідовність підключення семисегментного індикатора до виводів мікроконтролера

№ ніжки PIOB	№ виводу індикатора
0	1
1	2
2	4
3	6
4	7
5	9
6	10

Пасивний п’єзодинамік підключений до виводу PIOB мікроконтролера на лінію, що дорівнює номеру варіанту+6 (при умові, що лінія не задіяна на інший функціонал).

При ввімкненні живлення на семисегментному індикаторі загоряється цифра 9, через 500мс лунає короткий звуковий сигнал (писк) і індикатор висвічує цифру 8, так продовжується до цифри 0. При кожному переключенні чутно короткий писк. Коли висвічується «0», писк стає постійним. Алгоритм виконується 1 раз.

Для непарних варіантів:

Десятисегментний індикатор рівня підключити до ліній PIOA контролера за схемою, що описується таблицею 2. Для спрощення підключення буде задіяно лише 7 світлодіодів із 10.

Таблиця 2.

Підключення 10-сегментного індикатора до виводів МК

№ ніжки PIOA	№ виводу індикатора
0	10
1	9
2	8
3	7
4	6
5	5
6	4

Активний п'єзодинамік підключений до 7+(номер варіанту) виводу PIOA мікроконтролера.

При ввімкненні живлення загораються 7 сегментів 10-сегментного індикатора рівня. Два червоних, три жовтих та два зелених (рис. 6.4.б). Через 200 мс вимикається крайня зелена смуга, чутно звуковий сигнал, далі ще через 200 мс вимикається наступна смуга і чутно писк, і т.д. Чим менше стає смуг, тим коротші і частіші стають звукові сигнали. Коли вимкнувся останній світлодіод у індикаторі – писк постійний. Алгоритм виконується 1 раз.

6.3. Додаткове завдання

При натисканні на SW1 – алгоритм починає працювати. При натисканні на SW5 – відлік починається спочатку.

6.4. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Зібрати схему за завданням.
3. Оптимізувати код програми, застосувавши навички програмування.
4. Виконати відлагодження програми за допомогою вбудованого відладчика.

5. Перевірити працездатність та коректність програми на відладочній платі, завантаживши її в МК двома способами.
6. Створити другий проект, в якому буде відсутня бібліотека lib_AT91SAM7X256.h та використана форма запису через регістри.

6.5. Зміст звіту

1. Код програми використовуючи функції і запис через регістри.
2. Фото результату відладки.
3. Висновки.

6.6. Контрольні запитання

1. Яке джерело сигналів вибрано для відліку таймера/лічильника?
2. Що саме робить функція delay_config?
3. Які регістри використовуються у delay.c, навіщо?
4. Як розрахувати струмообмежуючі резистори для даних індикаторів?
5. Які способи формування цифр на семисегментому індикаторі ви знаєте?
6. Покажіть на прикладі, як можна змінити тональність пасивного п’єзодинаміка?
7. Чи можна за допомогою п’єзодинаміка програти мелодію? Як це зробити? Який тип динаміка, активний чи пасивний, краще використовувати?

ЛАБОРАТОРНА РОБОТА №7
ЗНАЙОМСТВО ІЗ МІКРОКОНТРОЛЕРОМ STM32F303VCT6

Мета роботи: познайомитись із основними характеристиками мікроконтролера STM32F303VCT6, навчитися працювати з портами вводу/виводу.

7.1.Теоретичні відомості

7.1.1. Основні характеристики МК STM32F303VCT6

У даній лабораторній роботі познайомимось із новим мікроконтролером STM32F303VCT6 фірми STmicroelectronics. Це сучасний 32-бітний мікроконтролер сімейства ARM Cortex-M4, загальний вигляд якого наведено на рис. 7.1.

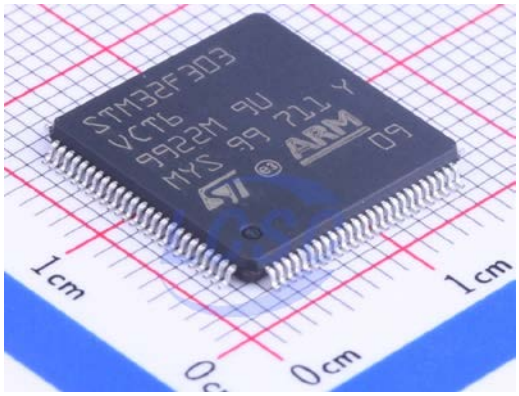


Рис. 7.1. МК STM32F303VCT6

МК має наступні характеристики:

- корпус LQFP-100;
- максимальну тактову частоту 72 МГц;
- 256 Кб Flash пам’яті (ПЗП) та 48 Кб RAM пам’яті (ОЗП);
- 87 входів/виходів, 45 з яких розраховані на рівень логічної 1 - 3.3В, інші 42 є сумісними із логікою, що працює із рівнем логічної 1 - 5В;
- інтерфейси зв’язку: CAN, I²C, IrDA, LIN, SPI, UART/USART, USB;
- 4 12-бітних АЦП;

- 2 12-бітних ЦАП;
- 4 операційних підсилювача;
- модуль для роботи з дробовими числами (FPU);
- блок захисту пам’яті (MPU);
- 10 таймерів загального використання.

7.1.2. Відладочна плата STM32F3 DISCOVERY

Вивчення МК STM32F303VCT6 буде проводитись на відладочній платі STM32F3DISCOVERY, загальний вигляд якої представлено на рис. 7.2.

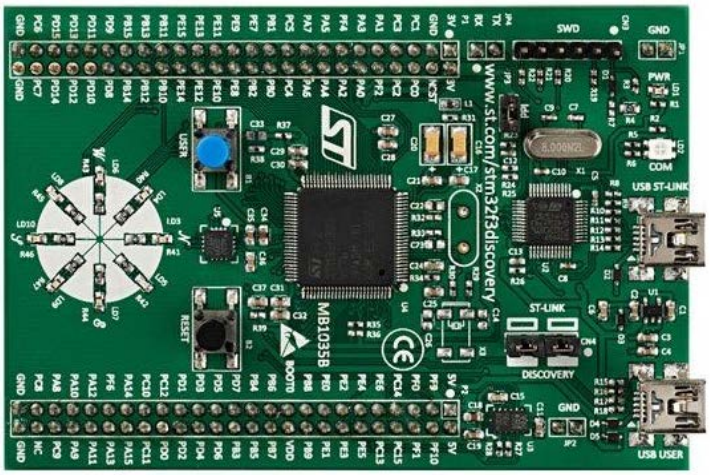


Рис. 7.2. Відладочна плата STM32F3DISCOVERY

До основних особливостей відладочної плати можна віднести наступні:

- MEMS датчики - 3-осьовий гіроскоп з цифровим виходом, комбінований датчик 3-осьового магнітометра та 3-осьового гіроскопа.
- Вісім користувальницьких світлодіодів: LD3 / 10 (червоний), LD4 / 9 (синій), LD5 / 8 (помаранчевий) і LD6 / 7 (зелений) та 1 кнопка.
- Роз'єми плати:

Роз'єм USB Mini-B – може бути використаний для підключення зовнішнього накопичувача, реалізації сервісних функцій і т.ін.

Роз'єм ST-LINK Mini-B USB – для швидкого програмування МК за допомогою вбудованого відладчика / програматора ST-LINK / V2.

Парні гребінки із 100 виводів, що призначені для швидкого підключення відладочної плати плати та макетування.

- Живлення може відбуватись через роз’єми ST-LINK USB або користувальницький роз'єм USB Mini-B. Зовнішня напруга живлення може бути або 3 В, або 5 В.

Для написання та відладки коду, як і раніше, будемо використовувати Keil uVision IDE. Проте, для більш глибокого засвоєння матеріалу, порівняння середовищ, пропонується змінити версію Keil на 4.7 або 5 і новіше.

Не зважаючи на те, що існує більше десяти різновидів відладочних плат серії Discovery, STM32F3DISCOVERY обрана з врахуванням того, що максимально забезпечує потреби і специфіку роботи кафедри комп’ютерно-інтегрованих оптичних та навігаційних систем, оскільки комплект присутніх інерціальних MEMS чутливих елементів дозволяє створити, наприклад, БІНС. При додатковому підключенні модулів бездротової передачі даних для з’єднання з ПК, отримаємо польотний контролер квадрокоптера. Порівняльна характеристика існуючих відладочних плат серії Discovery представлена в таблиці 7.1.

7.1.3. Робота з портами вводу/виводу

Для роботи з периферією та, безпосередньо, з портами вводу/виводу існує декілька бібліотек:

CMSIS (Cortex Microcontroller Software Interface Standard) — стандартна бібліотека для Cortex®-M. Вона забезпечує підтримку периферії і спрощує програмування інтерфейсів.

SPL (Standard Peripherals Library) – бібліотека, створена компанією STMicroelectronics. Має функції, структури та макроси для полегшення роботи з периферією мікроконтролера.

Таблиця 7.1.

Характеристики відладочних плат Discovery

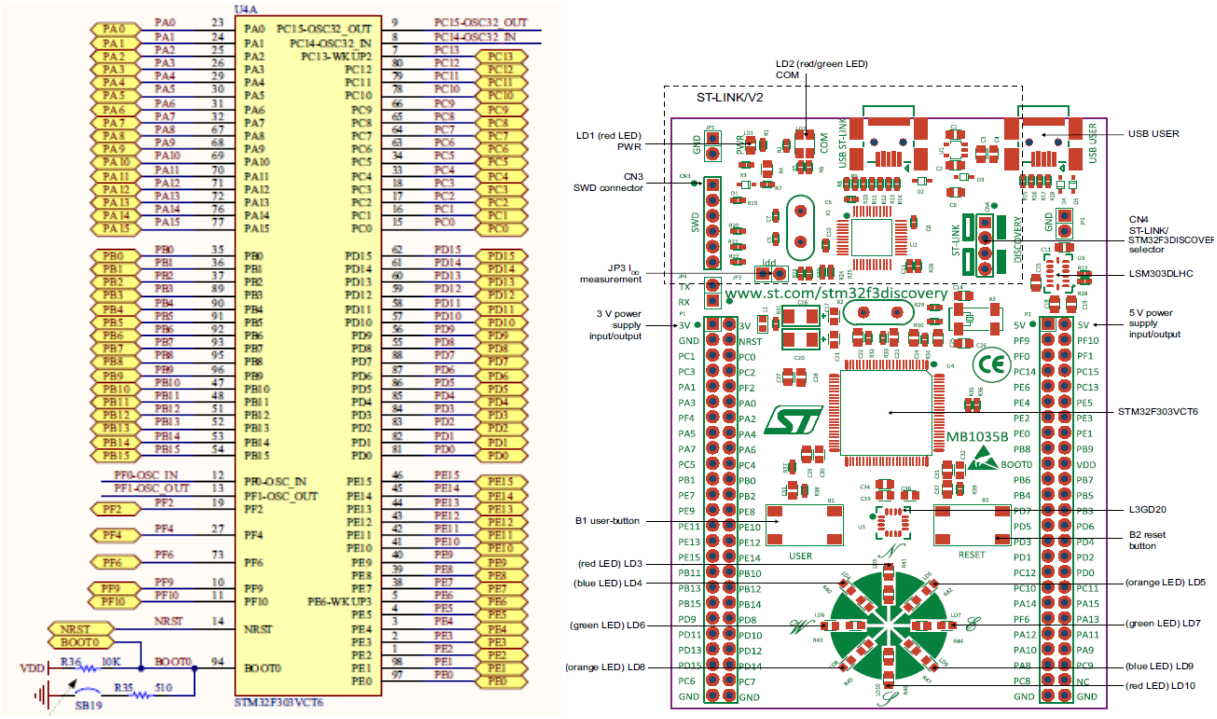
Відладочна плата	STM32f3 DISCOVERY	32F411E DISCOVERY	32F072B DISCOVERY	32F429I DISCOVERY
Мікроконтролер	32-бітний STM32F303VCT6 сімейства Cortex-M4	32-бітний STM32F411VET6 сімейства Cortex-M4	32-бітний STM32F072RB T6 сімейства Cortex-M0	32-бітний STM32F429ZI T6 сімейства Cortex-M4
Тактова частота, МГц	72	100	48	180
Відладчик-програматор	ST-LINK/V2			
ПЗП, Кб	256	512	128	2048
ОЗП, Кб	48	128	16	256
Периферія	Трьохвісьовий мікроелектромеханічний гіроскоп L3GD20			
	Мікроелектромеханічний акселерометр – магнітометр LSM303DLHC		Один лінійний датчик дотику	2.4" QVGA TFT LCD дисплей
	8 світлодіодів	4 світлодіода	4 світлодіода	2 світлодіода
		Звуковий підсилювач класа D CS43L22		Додатковий модуль пам’яті об’ємом 64 Мбіт
		Цифровий мікрофон MP45DT02		
	Кнопки user та reset			

За замовчуванням для меншого енергоспоживання вся периферія МК вимкнена. Тому для початку роботи необхідно скористатися наступним алгоритмом:

- Дозволити тактування порта/лінії.
- Обрати режим роботи лінії, та сконфігурувати її на вхід або вихід.
- Подавати дані у вигляді логічних значень, шляхом запису у відповідні регістри, або зчитувати стан ліній вводу/виводу.

Як бачимо, не дивлячись на відмінність мікроконтролерів фірм Atmel та ST – загальний алгоритм роботи з лініями вводу/виводу залишається не змінним.

Всі лінії вводу виводу загального призначення (у англомовній літературі зазначаються як GPIO. **Увага!** Не плутати із контролером паралельного вводу-виводу PIO) розбиті по 16, на 5 портів — GPIOA-GPIOE. На схемах, що наведені на рис. 7.3, лінії позначаються у форматі P (букви A-E) + цифра, наприклад, PE8.



а) б)

Рис. 7.3. Відладочна плата STM32F3DISCOVERY: а) спрощена електрична принципова схема підключення виводів МК; б) принципова схема

Як було вказано раніше, відладочна плата має 8 користувальницьких світлодіодів, які підключені до GPIOE (рис.7.3. б). Про це свідчать номера виводів PE8 – PE15, де PE – порт E. Схема підключення світлодіодів із зазначенням кольору представлено на рис. 7.4.

Отже, першою ітерацією наведеного алгоритму буде тактування порту E. За тактування відповідає модуль RCC та регістр *RCC_AHBENR*, карта якого представлена на рис. 7.5. Відтак, проаналізувавши підписи бітів регістру – бачимо, що за тактування порта E відповідає 21-ий біт, підписаний як IOPEEN.

Тож запис `RCC->AHBENR = 1<<21;` призведе до подачі тактових сигналів на порт E. Замість запису `1<<21;` можна скористатися існуючим

визначенням RCC_AHBENR_GPIOEEN, що у подальшому спростить читання та аналіз коду.

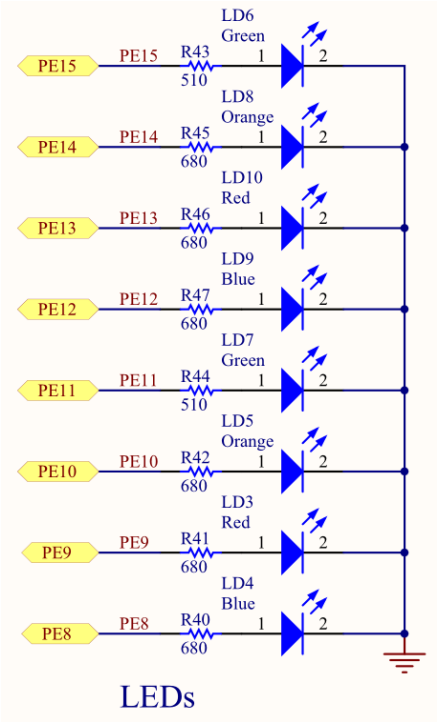


Рис.7.4. Схема підключення світлодіодів на відладочній платі STM32F3DISCOVERY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	ADC34 EN	ADC12EN	Res	Res	Res	TSCEN	IOPG EN(1)	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	IOPH EN(1)
		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	CRC EN	FMC EN(1)	FLITF EN	Res	SRAM EN	DMA2 EN	DMA1 EN
									rw	rw	rw		rw	rw	rw

1. Only on STM32F303xDxE.

Bit 21 **IOPEEN:** I/O port E clock enable(STM32F303xB/C and STM32F358xC devices only)
Set and cleared by software.
0: I/O port E clock disabled
1: I/O port E clock enabled.

Рис.7.5. Карта регістра *RCC_AHBENR* (Reset and clock control)

За режими роботи ліній відповідає регістр *MODER*, карта якого представлена на рис. 7.6. Регістр MODER умовно розділений на пари бітів. Кожна пара відповідає за конфігурацію режиму роботи однієї із 16 ліній. Отже,

можливо 4 варіанти запису: 00 – режим вводу даних (режим входу, лінія сконфігурована як вхід), це стан за замовчуванням; 01 – режим виходу; 10 – режим альтернативних функцій; 11 – аналоговий режим роботи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Рис.7.6. Карта регістра MODER

Для формування логічних рівнів на лініях портів відповідає регістр *ODR* (рис.7.7). Зазначимо, що в наведеному регістрі використовуються лише перші 16 бітів (0…15).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис.7.7. Карта регістру ODR

7.2. Завдання до виконання

Синій світлодіод постійно блимає з частотою приблизно 2Гц.

Код програми буде виглядати наступним чином:

```
        RCC->AHBENR    |=RCC_AHBENR_GPIOEEN;
        GPIOE->MODER   &= ~(3UL << 2* 8);
        GPIOE->MODER   |=  (1UL << 2* 8);
for(;;)
{
         GPIOE->ODR= 1<<8;
```

```
for (n=0;n<360000;n++);
        GPIOE->ODR = 0x0000;
for (n=0;n<360000;n++);
}
```

Строка GPIOE->MODER &= ~(3UL << 2* 8); записує код 00, попередньо очищаючи пару бітів у позиціях 16 та 17, для наступного запису значень.

Інші наведені строки, по принципу написання, повністю співпадають з відповідними строками із попередніх лабораторних робіт МК AT91SAM7X.

7.4. Завдання до виконання

Всі 8 користувальницьких світлодіодів блимають із частотою приблизно 2 Гц.

Для парних варіантів. Всі світлодіоди ввімкнені. Починаючи із першого, світлодіоди вимикаються проти годинникової стрілки. Після вимкнення всіх світлодіодів—алгоритм перезапускається.

Для непарних варіантів. Всі світлодіоди вимкнені. Перший світлодіод блимає 8 разів і вимикається, другий – 7 разів, третій – 6 і т.д. Після завершення блимання всіма світлодіодами алгоритм зупиняється.

7.5. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Створити код програми у головному файлі (з розшименням *.c).
3. Перевірити правильність роботи програми на платі, сфотографувавши, або продемонструвавши результат викладачу.
4. Оптимізувати код програми, уникаючи повторюваних частин коду (використовувати цикли).
5. Завантажити програму на МК двома способами.

7.6. Зміст звіту

1. Код програми.
2. Фото результату виконання (не обов’язково).
3. Висновки.

7.7. Контрольні запитання

1. Які основні характеристики МК STM32F303VCT6?
2. Які відмінності МК STM32F303VCT6 та AT91SAM7X256?
3. Які переваги та недоліки відладочної плати STM32F3DISCOVERY?
4. Наведіть основні етапи алгоритму роботи із лініями вводу / виводу.
5. Які можливі режими роботи ліній вводу/виводу?
6. Наведіть відмінності регістрів MODER, та відповідних, що використовувались у AT91SAM7X256.
7. Наведіть відмінності регістрів ODR, та відповідних, що використовувались у AT91SAM7X256.

ЛАБОРАТОРНА РОБОТА №8

ОПИТУВАННЯ ЦИФРОВИХ ПОРТІВ STM32F303VCT6

Мета роботи: вдосконалити навички роботи із цифровими портами вводу / виводу. Познайомитись із альтернативним способом програмування МК.

8.1. Теоретичні відомості

8.1.1. STM32 ST-LINK Utility

Утиліта ST-LINK utility призначена для програмування микроконтролерів STM32 через програматор ST-LINK або ST-LINK/V2. Цей спосіб необхідний, якщо відладочна плата із вбудованим програматором відсутня, або на відладочній платі не передбачено його наявності. У такому випадку програматор підключається окремо, такий підхід використовується для реалізації готових рішень та проєктів, або коли розміри відладочної плати завеликі для корпусу приладу.

Для програмування пам’яті мікроконтролера за допомогою STM32 ST-LINK Utility необхідно у налаштуваннях проекту поставити галку «Create HEX File»(Рис.8.1). Таким чином автоматично згенерується бінарний файл програми, який буде розміщений у папці проєкту.

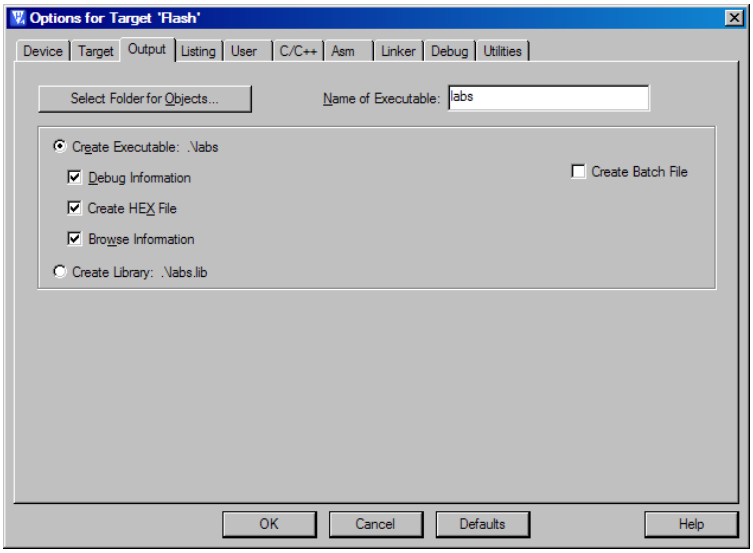


Рис.8.1. Налаштування для створення бінарного файлу

Після чого запустити STM32 ST-LINK Utility (програма доступна у мережевому оточенні комп'ютерного класу) і обрати пункт меню Target -> Connect (рис. 8.2)

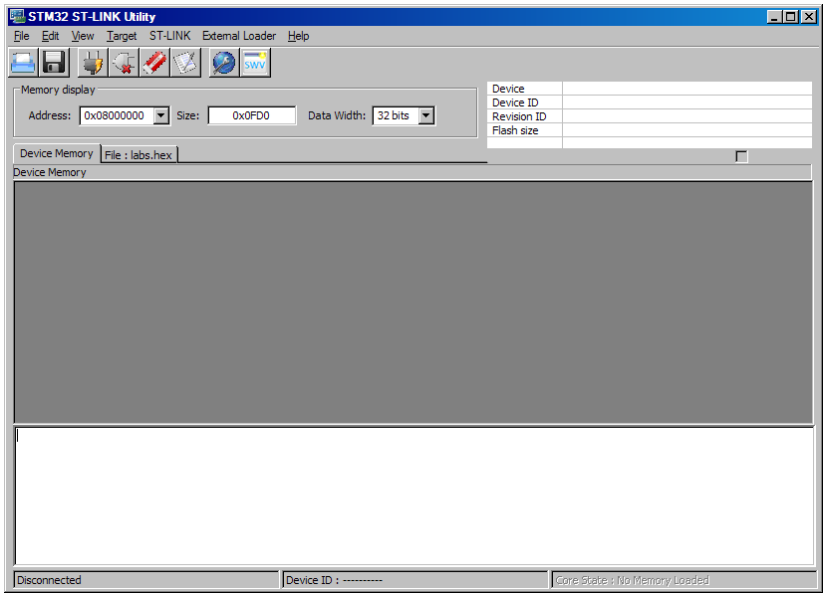


Рис.8.2. STM32 ST-LINK Utility

При правильному підключенні у вікні відобразиться вміст пам'яті мікроконтролера, як показано на рис. 8.3.

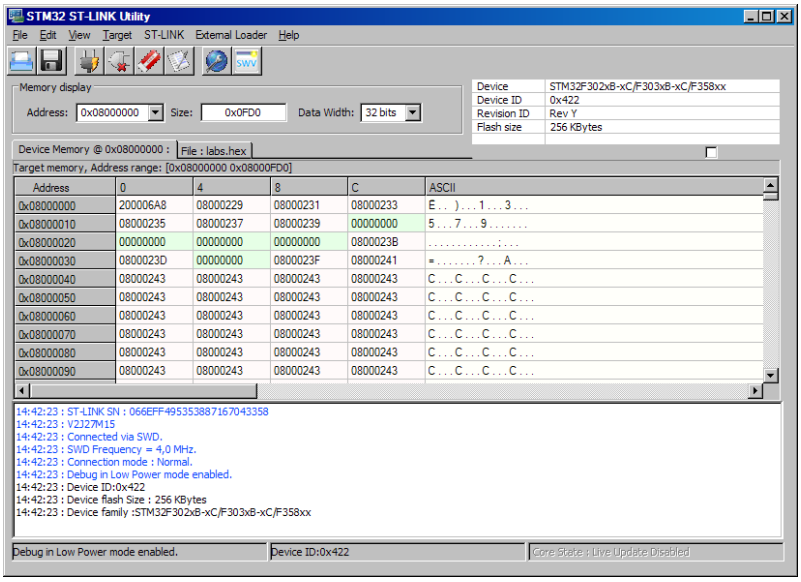


Рис.8.3. Вміст пам'яті мікроконтролера

Далі, для подальшого запису, проведемо процедуру очистки пам'яті. Для цього виконаємо Target -> Erase Chip (рис.8.4).

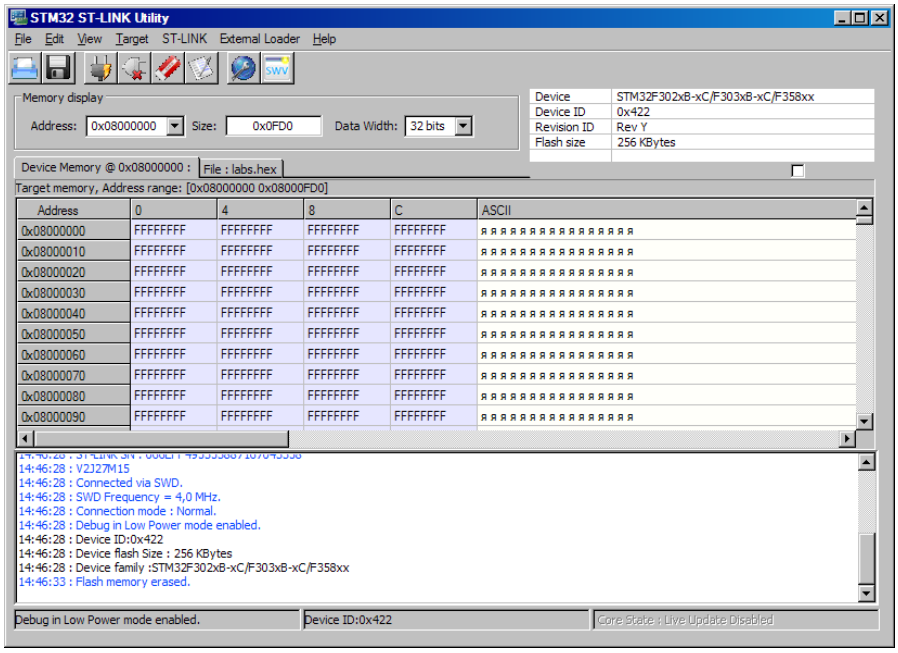


Рис.8.4. Відображення вмісту пам’яті після стирання

Обираємо програму у форматі *.hex за допомогою команди File -> Open file… Після чого програма доступна для перегляду (рис. 8.5).

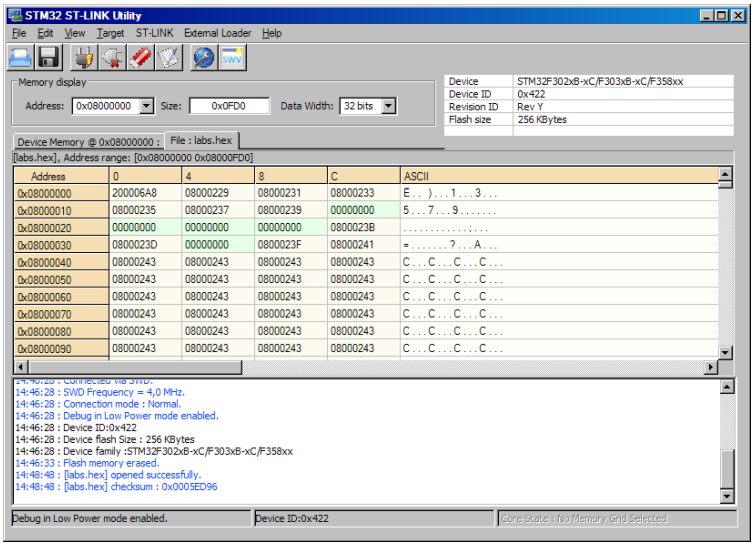


Рис.8.5. Відображення обраного файлу програми

Виконавши операцію Target -> Programm & Verify -> Start, завантажуємо файл у Flash-пам’ять мікроконтролера.

8.1.2. Опитування цифрових ліній

За зчитування даних із обраного порта відповідає регістр *IDR (*input data register*)*, карта якого показана на рис. 8.6. Як видно, регістр сформований по аналогії із регістром ODR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data bit (y = 0..15)
These bits are read-only. They contain the input value of the corresponding I/O port.

Рис.8.6. Карта регістра *IDR*

Для приєднання різноманітних пристроїв на кожній лінії кожного порта передбачені програмно керовані підтягуючі резистори, що описуються функціоналом регістру *PUPDR* (**p**ull-**u**p, **p**ull-**d**own **r**egister, рис. 8.7).

Регістр умовно розділений на пари бітів. Кожна пара відповідає за конфігурацію режиму роботи однієї із 16 ліній. Отже, можливо 3 варіанти запису: 00 – підтягуючий резистор відключений; 01 – підтягування до живлення, на лінії буде високий рівень напруги; 10 – підтягування до заземлення; 11 – зарезервовано.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Рис.8.7. Карта регістра *PUPDR*

8.2. Завдання до виконання

Світлодіоди реалізують послідовність “блимаючого по колу вогника”, тобто вмикається перший світлодіод, горить приблизно 0.5-0.1с і вимикається. За годинниковою стрілкою запалюється наступний світлодіод, і т.д. При натисканні та утриманні користувальницької кнопки – послідовність блимання зупиняється. При відпусканні – продовжується (саме продовжується, а не перезапускається).

Код програми буде виглядати наступним чином:

```
unsigned int n=200000;
unsigned long led_mask[] = {1UL <<  8, 1UL <<  9, 1UL << 10, 1UL << 11, 1UL << 12, 1UL << 13, 1UL << 14, 1UL << 15 };

void int_all (void)
{
    RCC->AHBENR     |=RCC_AHBENR_GPIOEEN;
    GPIOE->MODER    &= ~((3UL << 2* 8) | (3UL << 2* 9) |
                (3UL << 2*10) | (3UL << 2*11) |
                (3UL << 2*12) | (3UL << 2*13) |
                (3UL << 2*14) | (3UL << 2*15)  );
    GPIOE->MODER    |=  ((1UL << 2* 8) | (1UL << 2* 9) |
                (1UL << 2*10) | (1UL << 2*11) |
                (1UL << 2*12) | (1UL << 2*13) |
                (1UL << 2*14) | (1UL << 2*15)  );

 RCC->AHBENR   |= RCC_AHBENR_GPIOAEN;
 GPIOA->MODER    &= ~((3UL << 2*0)  );
 GPIOA->PUPDR    &= ~((3UL << 2*0)  );
}

void wait (void)
            {
            int i;
            for (i=1;i<n;i++);
            while (GPIOA->IDR & (1UL << 0)==1);
            }

int main (void)
    {
    int y;
    int_all();
    for(;;)
{
for (y=0;y<8;y++)
    {
    GPIOE->ODR = led_mask[y];
    wait();
```

```
GPIOE->ODR = 0x0;
wait(); }}}
```

Користувальницька кнопка підключена до нульової лінії порта А, отже, вираз RCC->AHBENR |= RCC_AHBENR_GPIOAEN; тактує порт А.

GPIOA->MODER &= ~((3UL << 2*0)); – лінія сконфігурована на вхід, для отримання логічного рівня із кнопки.

GPIOA->PUPDR &= ~((3UL << 2*0)); – підтягуючий резистор відключений, оскільки кнопка має свою схему підключення, що показана на рис. 8.8.

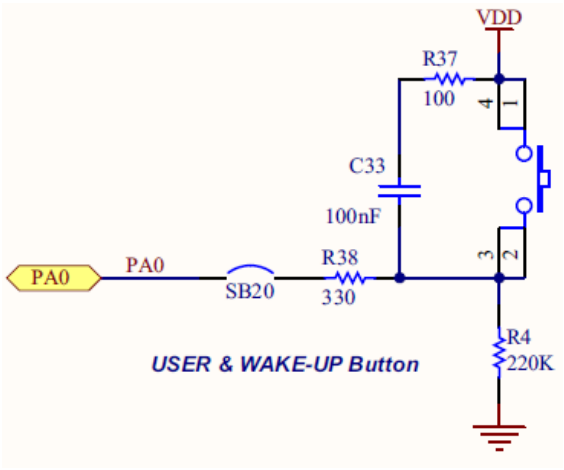


Рис.8.8. Схема підключення користувальницької кнопки

Цикл while (GPIOA->IDR & (1UL << 0)==1); працює наступним чином: поки натиснута кнопка, цикл продовжує працювати і не дозволяє завершити функцію wait, і є аналогією коду, що був використаний у п.п.5.1.2.

8.3. Додаткове завдання

Підключити матричну клавіатуру до мікроконтролера. Схема підключення може бути обрана самостійно або повторювати схему, що наведена на рис. 4.2. п.п. 4.1.3.

Лінії для підключення обираються наступним чином: підключення проводити до порту А, номер лінії відповідає номеру варіанту. Якщо варіант більше 15, то починаємо спочатку порта А. Тобто, 16 варіант – лінія №1, 17 – №2, т. д.

Додатково підключити 3 світлодіоди (електрична схема показана на рис.3.6.) до портів B, C та D. Якщо варіант з 1 по 10 – до порта B, з 11 по 20 – до C, з 21 по 30 – до D. Номери ліній відповідають номеру розряду одиниць в номері варіанту, тобто, наприклад, 28 - номери ліній будуть 8, 9 та 10 порта C, Варіант 23 – номери 3, 4 та 5 порта D.

До будь-якого вільного виводу підключити активний п’єзодинамік.

Для парних варіантів. При натисканні користувальницької кнопки, підключені світлодіоди блимають сигналом «SOS», при кожному блиманні чутний писк. Причому, час звуку завжди постійний. При натисканні на будь-яку підключену кнопку матричної клавіатури– сигнал «SOS» зупиняється.

Для парних варіантів. Натиснута клавіша «1» на матричній клавіатурі вмикає всі світлодіоди на платі (і вони продовжують світитись, навіть якщо кнопка вже не натиснута), «2» – загораються підключені світлодіоди, «3» - всі світлодіоди вимикаються. При кожному натисканні на будь-яку кнопку чутний короткий писк.

8.4. Порядок виконання лабораторної роботи

1. У середовищі Keil uVision створити новий проект.
2. Створити код програми у головному файлі (з розширенням *.c).
3. Перевірити правильність роботи програми на платі, сфотографувавши, або продемонструвавши результат викладачу.
4. Оптимізувати код програми, уникаючи повторюваних частин коду.

8.5. Зміст звіту

1. Код програми.
2. Фото результату виконання (не обов’язково).
3. Висновки.

8.6. Контрольні запитання

1. GPIO регістри та їх призначення.
2. Коли треба використовувати підтягуючі резистори?
3. Як зміниться код програми при опитуванні регістру IDR, якщо резистор підтягнутий до живлення, або до землі? (Електрична схема буде узгоджена із кожним варіантом підключень).
4. Що відбудеться якщо ввімкнути тактування всієї периферії? Поясніть на прикладах.
5. Що відбудеться, якщо замість GPIOE->MODER|=0x55550000; записати GPIOE->MODER=0x55550000;
6. Якими способами можна записати значення у регістр? Наведіть приклади.

ЛАБОРАТОРНА РОБОТА № 9.
ЗНАЙОМСТВО З АПАРАТНОЮ ПЛАТФОРМОЮ ARDUINO

Мета роботи: знайомство з основними характеристиками апаратної платформи Arduino та з програмним середовищем розробки Arduino IDE.

9.1. Теоретичні відомості

9.1.1. Основні характеристики платформи

Arduino — апаратна обчислювальна платформа, основними компонентами якої є проста плата вводу/виводу і середовище розробки мовою Processing/Wiring. Платформа створена для простого та універсального застосування, у першу чергу для створення сучасних електронних пристроїв з керуванням від мікропроцесорного ядра, яке використовується для прийому сигналів від різних цифрових і аналогових датчиків, а також, керування різними виконавчими пристроями.

У якості обчислювального ядра Arduino використовує мікроконтролери Atmel AVR. У курсі вивчається плата Arduino Uno та її різновиди (рис. 9.1), що використовує МК ATmega328. Платформа має 14 цифрових входів/виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, кварцовий генератор із тактовою частотою 16 МГц, USB порт та силовий роз'єм, кнопку перезавантаження. Не зважаючи на доволі скромний набір бортових периферійних пристроїв, платформа дозволяє підключити велику кількість датчиків, модулів і т.п.

До основних характеристик, на прикладі відладочної плати Arduino Uno, можна віднести наступні:

Мікроконтролер	ATmega328
Розрядність	8 біт
Тактова частота	16 Мгц
Робоча напруга	5 В
Напруга живлення	7-12 В
Гранична напруга живлення	6-20 В

Цифрові лінії	14 (6 як виходи ШІМ)
Аналогові входи, кількість	6
Максимальний струм на цифрових лініях	40 мА
Максимальний струм для виводу 3.3 В	50 мА
Флеш-пам`ять	32 Кб
ОЗП	2 Кб
EEPROM	1 Кб

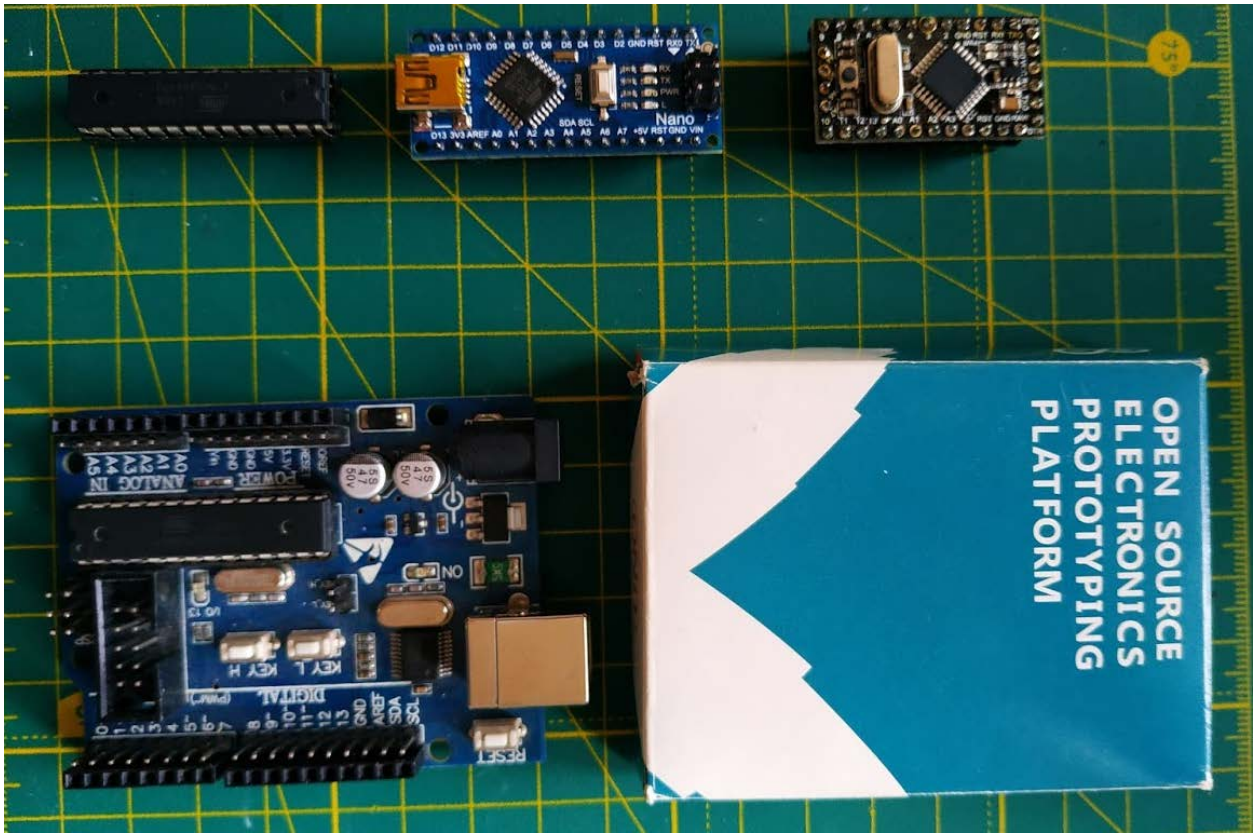


Рис. 9.1 Різновиди відладочних плат Arduino

Живлення плати:

Відладочні плати Arduino можуть отримувати живлення через USB порт або від зовнішнього джерела живлення. Джерело живлення вибирається автоматично. Зовнішнє живлення підключається за допомогою роз'єму 2.1 мм із центральним додатнім контактом. Напруга живлення коливається у межах 6-20В, причому як змінного так і постійного струму.

Виводи живлення:

Для підключення більшості модулів, датчиків та чутливих елементів, застосовується стандартизовані напруги живлення для пристроїв ТТЛ логіки — 5В та 3.3В постійного струму.

5V. Стабілізоване джерело живлення, що використовується для живлення мікроконтролера і компонентів на платі. Живлення може подаватися від виводу Vin через стабілізатор напруги або від USB порта.

3V3. Напруга на виводі 3.3 В формується вбудованим стабілізатором типу AMS1117, проте максимальний струм на даному виводі не може перевищувати 50мА.

GND - виводи заземлення.

Пам'ять:

Мікроконтролер ATmega328 має 32кб Flash-пам'яті, з яких 0,5кб використається для зберігання завантажувача, а також 2 кб ОЗП (SRAM) і 1 Кб EEPROM.

Входи і Виходи:

Зазвичай, для комутації із цифровими пристроями, на платі Arduino UNO використовується 14 цифрових ліній загального призначення, які можуть бути сконфігуровані як вхід або вихід. На платі позначені D0-D13. Напруга логічної 1 відповідає 5 В. Кожна лінія має програмно керований навантажувальний резистор опором 20-50 кОм. Максимальний струм на кожній лінії - до 40 мА.

Послідовна шина:

0 (RX) і 1 (TX). Для обміну даними із ПК використовується послідовний інтерфейс UART, для UNO використовуються лінії D0 для отримання (RX) і D1 - передачі даних (TX).

Широтно-імпульсна модуляція (ШІМ): D3, D5, D6, D9, D10, і D11 виводи плати. Кожен із зазначених виводів забезпечує формування сигналу ШІМ з розрядністю 8 біт за допомогою функції analogWrite().

Стандартизовані периферійні інтерфейси:

SPI: D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK). За допомогою даних виводів здійснюється зв'язок по шині SPI, для цього використовується відповідна бібліотека SPI.h

I2C: D4 (SDA) і D5 (SCL). За допомогою виводів здійснюється зв'язок I2C (TWI), для створення якого використається бібліотека Wire.h

Входи АЦП. На відладочній платі Arduino Uno присутній 6 канальний 10 бітний аналого-цифровий перетворювач (входи A0 - A5). Верхня межа перетворюваної напруги становить 5 В, проте вона може бути змінена за допомогою контакту AREF і функції analogReference().

LED: На платі присутній 1 користувальницький вбудований світлодіод, підключений до цифрового виводу D13.

Програмування:

МК ATmega328 має записаний завантажувальник, що дозволяє програмування МК без використання зовнішніх програматорів. Зв'язок забезпечується за допомогою протоколу STK500.

Автоматичне (програмне) перезавантаження:

При кожному перезаписі керуючої програми МК відбувається автоматичне перезавантаження МК. Також перезавантаження автоматично відбувається після закінчення запису програми для її запуску.

Струмовий захист USB:

У платі Arduino Uno передбачено вбудований автоматичний запобіжник, що захищає USB порт комп'ютера від струмів короткого замикання і перевантаження. Запобіжник спрацьовує при перевищенні струмом порогу 500 мА через USB порт і розмикає ланцюг. Після відновлення значень у допустимих меж, з’єднання відновлюється.

9.1.2. Програмне середовище Arduino IDE

Платформа Arduino програмується власним програмним середовищем: Arduino IDE - це кросплатформене середовище створене за допомогою мови Java, що включає в себе редактор коду, компілятор і модуль передачі керуючої програми в МК. Середовище створено на основі мови програмування Processing, а, отже, використовує C/C++ подібний синтаксис, що зручно для роботи із кодом і бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC. Основне вікно програми показане на рис. 9. 2. Програма, написана в середовищі Arduino, називається скетч.

Arduino IDE у порівнянні із середовищем Keil uVision має набагато примітивніший інтерфейс, що пояснюється відсутністю необхідності емуляції підсистем та ядра МК.

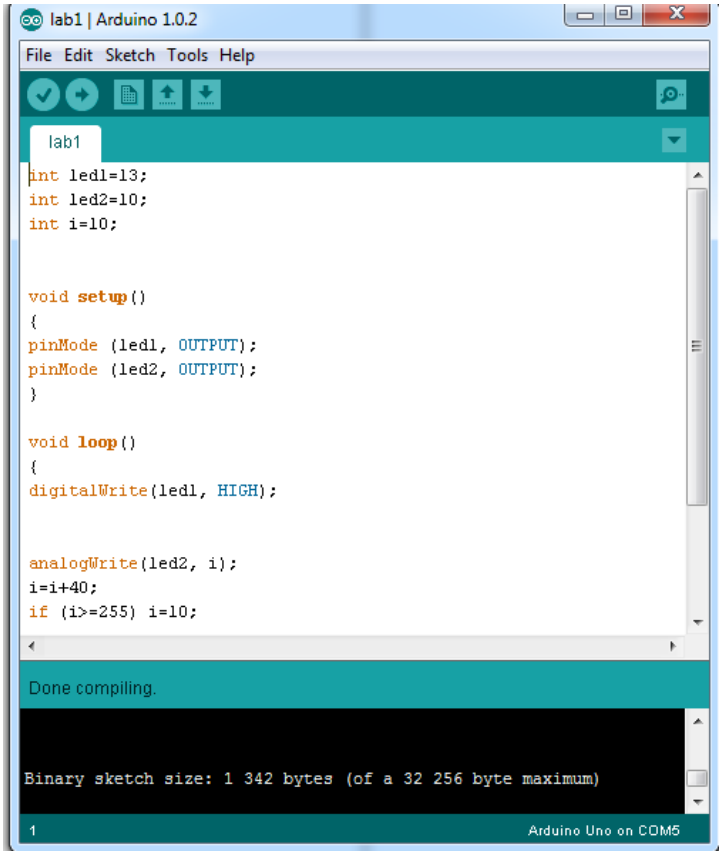


Рис. 9.2. Програмне забезпечення Arduino

Основні елементи керування головного вікна Arduino IDE наступні:

 - перевірка коректності синтаксису коду;

 - перевірка синтаксису та завантаження коду на МК;

 - іконки створення нового скетчу, відкриття і збереження відповідно;

 - монітор послідовного порта (Serial-інтерфейс).

9.2. Завдання на виконання

Блимання вбудованим світлодіодом із частотою, що відповідає наступній формулі: f=1/T, де T=100мс*№ варіанту.

9.3. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.
2. Набрати наведений код програми:

```
int led1=13;

void setup()
{
pinMode (led1, OUTPUT);
}

void loop()
{
digitalWrite(led1, HIGH);
delay(500);
digitalWrite(led1, HIGH);
delay(500);

}
```

У функцію delay() вказати значення f згідно із завданням.

3. Підключити відладочну плату Arduino UNO до ПК, використовуючи кабель USB.

4. Завантажити записаний скетч на плату, переконатися в працездатності програми.

9.4. Зміст звіту

1. Повторити створення нового проекту у середовищі Arduino та програмування мікроконтролера (2-3 рази).
2. Зробити висновки по роботі.

9.5. Питання для самоконтролю

1. Який мікроконтролер використовує апаратна платформа Arduino UNO?
2. Яка відладочна плата Arduino розглянута в лабораторній роботі?
3. Які основні характеристики відладочної плати?
4. Яка мова використається для написання програм у ПЗ Arduino?
5. Як називається програма написана в середовищі розробки Arduino?

ЛАБОРАТОРНА РОБОТА № 10.
ЗНАЙОМСТВО ІЗ ЦИФРОВИМИ ВХОДАМИ/ВИХОДАМИ

Мета роботи. Навчитися працювати із цифровими входами/виходами. Навчитися взаємодіяти із периферійними пристроями.

10.1. Теоретичні відомості

Структура скетчу, основні функції

З попередньої лабораторної роботи відомо, що Arduino використовує мову Processing, яка є наслідницею C/C++. Отже, основні конструкції синтаксису мови C/C++ залишаються без змін і докладно описані не будуть.

Тепер познайомимось із структурою скетча, що був створений у минулій лабораторній роботі.

int led1=13; – оголошується глобальна цілочисельна змінна зі значенням 13. Ця змінна потрібна для звернення до єдиного вбудованого світлодіода, який під'єднаний до лінії D13.

void setup() – функція призначена для попередніх настройок та конфігурації елементів скетчу. У даному випадку функція призначена для конфігурації ліній контролера. Також у цю функцію зручно помістити конструкції, які виконуються однократно.

pinMode (led1, OUTPUT); Цифрові виводи платформи Arduino. Можуть працювати як входи, або як виходи, тому функція *pinMode* встановлює режим роботи заданого входу/виходу як вхід, або як вихід.

Синтаксис pinMode(pin, mode), де

pin: номер лінії входу/виходу(pin),

mode: INPUT або OUTPUT. Значення, що конфігурує ніжку на вхід або вихід відповідно.

void loop() – основна функція скетчу. Тіло даної функції повторюється постійно.

digitalWrite(led1, HIGH); – подає HIGH (1) або LOW(0) на цифровий вхід/вихід.

Синтаксис digitalWrite(pin, value), де

pin: номер входу/виходу(pin),

value: значення HIGH або LOW.

delay(500); – функція створення затримки, значення якої задається в мс.

Розглянемо функцію, що дозволяє зчитувати значення з цифрової лінії:

digitalRead(pin) – функція зчитує значення із заданого цифрового входу.

pin: номер цифрової лінії. Функція повертає бульове значення, що знаходиться на обраному цифровому вході.

10.2. Завдання на виконання

10.2.1. Завдання №1. Кнопка і світлодіод

При натисканні на кнопку світлодіод гасне, при повторному натисканні загоряється знову, і т.д.

Зібрати схему наведену на рис. 3.

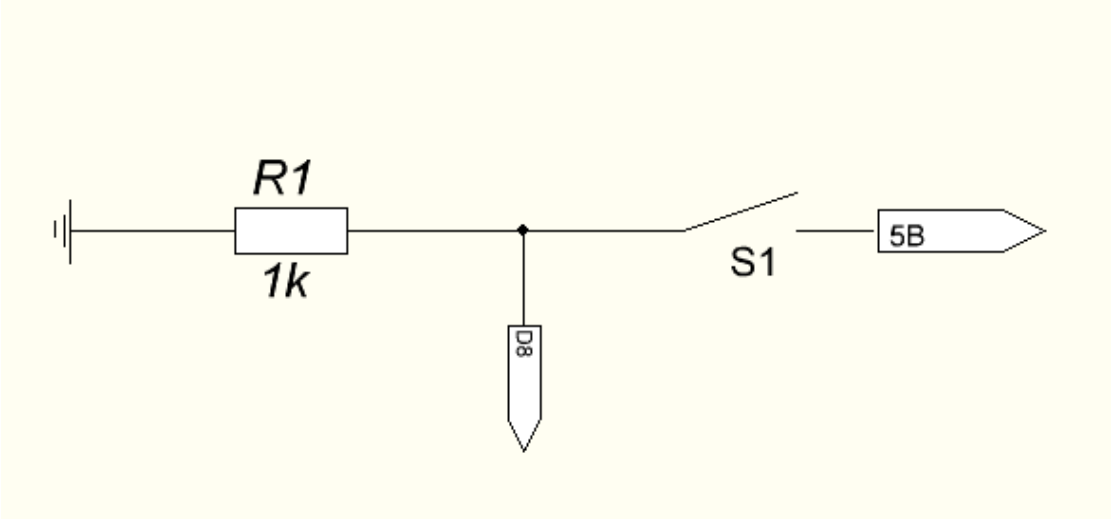


Рис.10.1. Схема до завдання

Створити новий проект і набрати приведений код програми:

```
int switchPin=8;
int ledZhol=13;
boolean lastButton=LOW;
boolean ledOn=LOW;
```

```
void setup ()
{
  pinMode(switchPin, INPUT);
  pinMode(ledZhol, OUTPUT);
}

void loop()
{

  if (digitalRead(switchPin)==HIGH && lastButton==LOW )
  {

 ledOn=!ledOn;
 lastButton=HIGH;
  }
  else {
 lastButton=digitalRead(switchPin);
      }
digitalWrite(ledZhol, ledOn);
 }
```

Змінити лінію, до якої підключена кнопка на наступну: №лінії = №варіанта -1. Якщо значення більше 13, то починаємо з 0. Тобто варіант 15 відповідає лінії №0.

10.2.2. Завдання №2. Сфітлофор

Алгоритм роботи:

- Спочатку горить зелений світлодіод, жовтий і червоний вимкнені.
- При натисканні на кнопку, зелений мигає 2 рази (по 500 мс) і гасне.
- На 2с вмикається жовтий.
- На 5с вмикається червоний.
- На 2с вмикається жовтий.
- Вмикається зелений.

Для виконання завдання необхідно зібрати схему наведену на рис. 10.2.

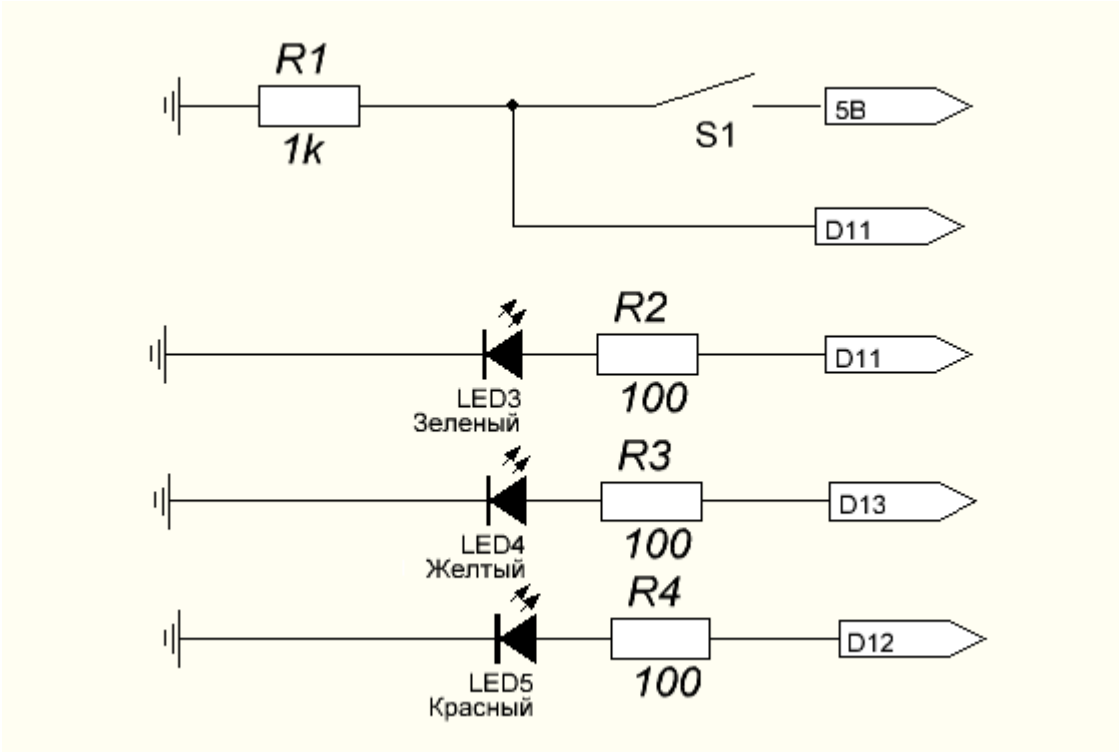


Рис.10.2. Схема до завдання 2

10.3. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.

2. Набрати код програми, що наведений у пп. 10.2.1., з врахуванням номеру варіанту.

3. Завантажити записаний скетч на плату, переконатися в працездатності програми.

4. Зібрати схему приведену на рис. 10.2, та створити код скетчу згідно зі завданням із пп. 10.2.2.

5. Завантажити записаний скетч в МК, переконатися в працездатності програми.

10.4. Зміст звіту

1. Перенести створені коди по завданням з пп. 10.2.1. та 10.2.2.
2. Дати рекомендація по оптимізації коду програми.
3. Зробити висновки по роботі.

10.5. Питання для самоконтролю

1. Для чого потрібна функція setup()?
2. У чому особливість використання функції loop()?
3. Які значення аргументів може мати функція pinMode?
4. Для чого потрібна функція затримки?
5. Для чого використовуються змінні lastButton і ledOn?

ЛАБОРАТОРНА РОБОТА № 11.
ЗНАЙОМСТВО З АНАЛОГОВИМИ ВХОДАМИ/ВИХОДАМИ ПЛАТФОРМИ ARDUINO

Мета роботи. Навчитися працювати з виходами, що підтримують ШІМ. Навчитися працювати з АЦП МК.

11.1. Теоретичні відомості

11.1.1. Дільник напруги

Для зменшення значення вхідної напруги використовують резистивний дільник напруги. Його вихідна напруга Uвих залежить від значення вхідної напруги Uвх і значень опору резисторів. Дільник напруги (Рис. 11.1) – одна із найбільш розповсюджених схем, і також буде використана для даної лабораторної роботи.

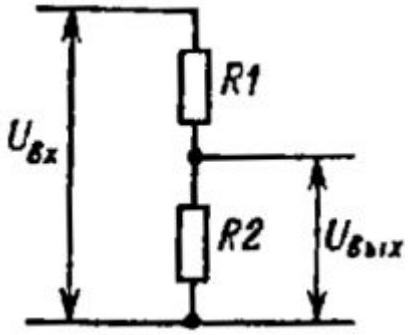


Рис. 11.1. Схема дільника напруги

Його вихідна напруга описується наступною формулою:

$$\boldsymbol{U}_{\text{вих}} = \boldsymbol{U}_{\text{вх}} \frac{\boldsymbol{R_2}}{\boldsymbol{R_1 + R_2}}$$

Одним із головних недоліків резистивного дільника напруги є те, що номінал опорів дільника повинен бути на декілька порядків більший, ніж номінальний опір навантаження.

Малі значення опорів приводять до виникнення великих струмів у дільнику. Знижується ККД схеми через нагрівання опорів.

Резистивний дільник напруги не можна використовувати для підключення потужних електричних приладів: електричні машини, нагрівальні елементи і т.д.

11.1.2. Аналого-цифровий перетворювач

Як було зазначено раніше, на відладочній платі Arduino Uno присутній 6 канальний 10-бітний аналого-цифровий перетворювач (входи A0 - A5). Розглянемо його характеристики більш детально:

Розрядність АЦП характеризується кількістю дискретних значень, на які АЦП може розбити діапазон вхідних значень. Для 10 бітного АЦП це значення відповідає 2^{10}=1024 дискрета. Стандартний вхідний діапазон напруг для АЦП Arduino Uno становить від 0 до 5В. Це означає, що напруга, яка буде подана на аналоговий вхід, буде перетворена в значення від 0 до 1023 (всього 1024 значення).

Роздільна здатність аналого-цифрового перетворювача – це мінімальна зміна величини аналогового сигналу, що може бути перетворена АЦП. Цей параметр безпосередньо пов'язаний з розрядністю АЦП. Роздільна здатність дорівнює сумарному максимальному діапазону вимірюваної напруги, поділеному на кількість вихідних дискретних значень.

Напруга подана на аналоговий вхід (від 0 до 5В) буде перетворена в значення від 0 до 1023, тобто 1024 рівні квантування. А отже, роздільна здатність АЦП буде дорівнювати (5-0)/1024 = 0,0049 В ≈ 5 мВ.

Розберемо декілька прикладів.

Приклад 1:

Діапазон вхідних значень — від 0 до 10 В.

Розрядність двійкового АЦП 12 біт: 2^{12} = 4096 рівнів квантування.

Роздільна здатність двійкового АЦП по напрузі:

(10-0)/4096 = 0,00244 В = 2,44 мВ.

Приклад 2:

Діапазон вхідних значень — від -10 до +10 В.

Розрядність двійкового АЦП 14 біт: $2^{14} = 16\ 384$ рівнів квантування.

Роздільна здатність двійкового АЦП по напрузі:

(10-(-10))/16384 = 20/16384 = 0,00122 В = 1,22 мВ.

Приклад 3:

На вхід 10-бітного АЦП із діапазоном вимірюваних напруг від 0 до 3.3В, прийшло значення 1В. Який код АЦП буде сформований?

(1В*1023)/3.3В = 310.

Опитування значення з аналогового входу і його перетворення займає приблизно 100 мкс (0.0001 сек). Тобто максимальна частота зчитування дорівнює приблизно 10000 разів у секунду, або 10кГц.

11.1.3. Широтно-імпульсна модуляція

Широтно-імпульсна модуляція (ШІМ) — це операція одержання аналогового значення на цифрових лініях і є одним із найпростіших видів цифро-аналогового перетворювача (ЦАП). На лініях формуються прямокутні імпульси — сигнал, що постійно перемикається між значеннями логічної 1 і 0. Даний сигнал моделює напругу між максимальним значенням (5 В) і мінімальним (0 В), змінюючи при цьому тривалість часу включення високого рівня відносно низького. Тривалість включення максимального значення називається шириною імпульсу. Для одержання різних аналогових величин змінюється ширина імпульсу. На виході кожної такої лінії, що підтримує ШІМ, встановлена інтегруюча ланка, яка згладжує прямокутні імпульси до сталого значення із діапазону 0В і 5В, таким чином сформувавши аналоговий сигнал. Arduino UNO використовує 8 розрядний ШІМ, отже, коди, що будуть перетворені у певну аналогову напругу, будуть змінюватись у діапазоні від 0 до 255 (всього 256 значень).

На рис.11.2. зелені лінії відлічують постійні часові періоди. Тривалість періоду обернено пропорційна частоті ШІМ. Тобто, якщо частота ШІМ становить 500 Гц (за замовчуванням для плати Arduino UNO вона дорівнює 488,28 Гц), то зелені лінії будуть відзначати інтервали тривалістю в 2мс.

Виклик функції analogWrite(255) буде відповідати 100% робочого циклу (постійне включення 5 В), а значення analogWrite(127) – 50% робочого циклу і напрузі 2,5В.

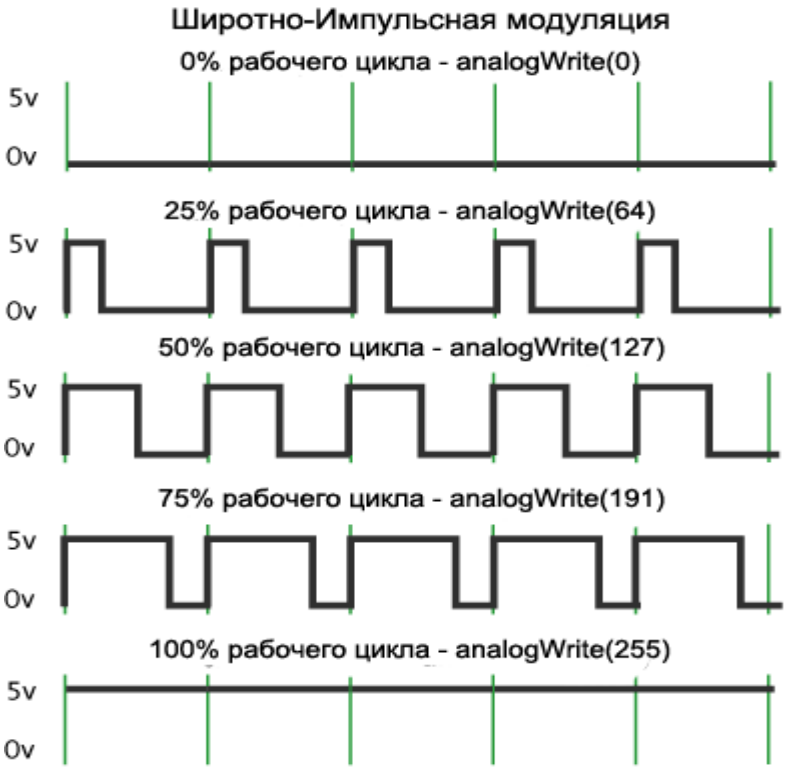


Рис. 11.2. Приклади широтно-імпульсної модуляції

11.2. Опис основних функцій

analogRead(pin); – функція, яка зчитує значення із заданого аналогового входу і повертає значення у вигляді коду АЦП.

де pin: номер аналогового входу A0-A5. Функція повертає значення у діапазоні від 0 - 1023.

analogWrite(pin, value), де

pin: номер входу/виходу (лише лінії, що підтримують ШІМ),

value: значення 0-255.

constrain(x, a, b) – функція обмежує діапазон значень змінної x. Аргументи функції: x – вхідна змінна, a – мінімальне значення діапазону, b – максимальне. Тобто якщо x<a, то функція повертає a, якщо x>b – b.

Якщо a<x<b - то x.

map(x, a1, b1, a2, b2) – функція перевизначає значення числа x з поточного діапазону [a1,b1] в новий [a2,b2].

11.3. Завдання на виконання

11.3.1. Завдання №1. Розумний нічник

Пристрій моделює алгоритм роботи смарт ліхтарів-нічників. Він працює за наступним алгоритмом: при денному світлі – світлодіод вимкнений, чим темніше стає – тим яскравіше загоряється світлодіод.

Для цього необхідно зібрати схему наведену на рис. 11.3. У якості світлочутливого елементу будемо використовувати фоторезистори, а схему підключення до МК будемо реалізовувати за схемою дільника напруги.

У лабораторній роботі студентам надається 2 типи фоторезисторів (GL5528, GL5516). Тому, використовуючи мультиметр, необхідно визначити опір фоторезистора при денному освітленні і при темряві (закрити рукою).

Далі, визначити значення напруги за формулою дільника напруги за умови, що резистор R2 має номінал 1 кОм *№Варіанта.

Визначити прогнозовані значення напруг у коді АЦП.

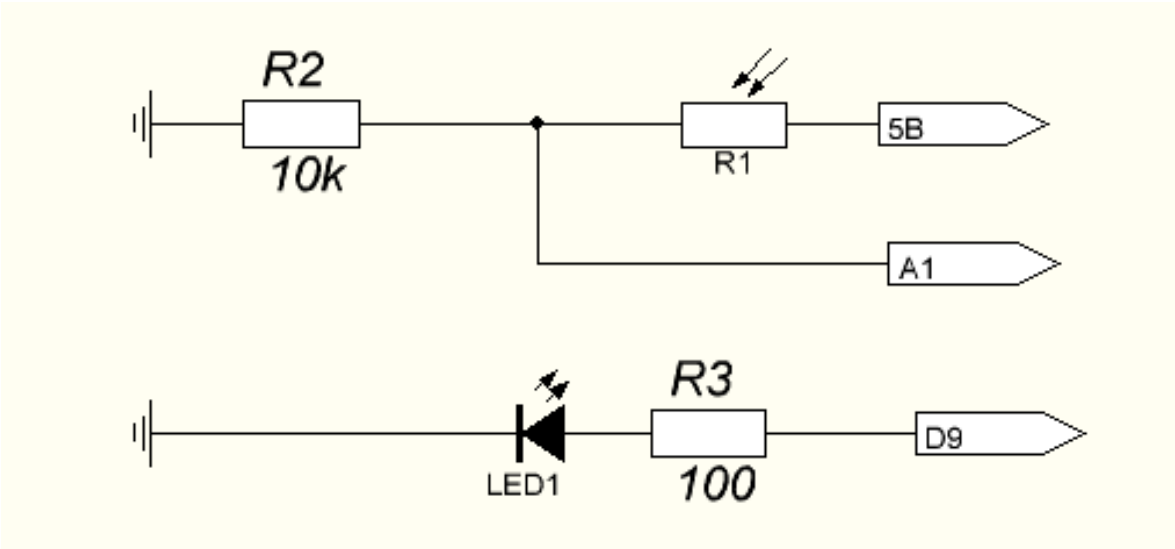


Рис. 11.3. Схема до завдання

11.3.2. Завдання №2. Розумний вентилятор

Алгоритм роботи: коли прохолодно – вентилятор вимкнений, чим тепліше стає, тим швидше обертається вентилятор. Зміна температури моделюється затисненням термочутливого елемента у долоні.

Використовуючи рис. 11.3, зібрати схему, замінивши фоторезистор R1 на термістор. До цифрової ніжки D9, замість світлодіода, підключити мікропотужний двигун. Термістор і двигун буде надано викладачем.

Алгоритм виконання завдання повністю збігається із алгоритмом із завдання №1.

11.4. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.
2. Створити код програми у відповідності до завдання із пп. 11.3.1;
3. Завантажити записаний скетч в МК, переконатися в працездатності програми при різних рівнях освітленості.
4. Зібрати схему, приведену на рис. 11.3, та створити код скетчу за завданням із пп. 11.3.2.
5. Завантажити записаний скетч в МК, переконатися в працездатності програми.

11.5. Зміст звіту

1. Перенести написаний код для завдань №1 і №2.
2. Виділити основні відмінності у написаних кодах.
3. Зробити висновки по роботі.

11.6. Питання для самоконтролю

1. Яка розрядність використовуваного АЦП?
2. Яка чутливість використовуваного АЦП?
3. Яка розрядність використовуваного ШІМ?

4. Яку мінімальну напругу можливо одержати на ніжці із ШІМ?
5. Для чого застосовується функція constrain?
6. Коли не можна застосовувати резисторний дільник напруги?
7. Як ще можна використовувати зазначені чутливі елементи?

ЛАБОРАТОРНА РОБОТА № 12.
УЛЬТРАЗВУКОВИЙ ДАЛЬНОМІР

Мета роботи. Створити прилад для вимірювання відстані на базі ультразвукового дальноміра. Навчитися використовувати послідовний комунікаційний інтерфейс.

12.1. Теоретичні відомості

12.1.1. Ультразвуковий дальномір

Загальний принцип функціонування ультразвукового дальноміра наступний: передавач випромінює короткий ультразвуковий імпульс, що відбивається від об'єкта і приймається сенсором-приймачем. Відстань до об’єкта розраховується виходячи із часу від початку випромінювання до приходу відбитого сигналу з врахуванням швидкості звуку в середовищі.

У лабораторній роботі розглядається модуль ультразвукового далекоміра на прикладі HC-SRO4, загальний вид якого наведений на рис. 12.1. Основні характеристики датчика представлені у табл.12.1. Модуль живиться від джерела постійного струму напругою від 3.3 до 5В, для цього призначені лінії Vcc і Gnd. Інформаційними лініями є лінія Trig, що призначення для активації послідовності вимірювання, та лінія Echo – призначена для передачі відповіді МК про відстань до об’єкта.

Рис. 12.1. Загальний вид чутливого елемента ультразвукового дальноміра

Якщо на сигнальну ніжку (Trig) подається імпульс тривалістю 10-15мкс, то ультразвуковий модуль буде випромінювати вісім пакетів ультразвукового сигналу із частотою 40кГц і чекати їхній відгук.

Таблиця 12.1. Основні характеристики датчика HC-SRO4

Напруга живлення	5В
Струм спокою	< 2 мА
Ефективний кут огляду	< 15 °
Діапазон вимірів	2 см - 500 см
Точність	0,3 см

На рис. 12.2. наведено циклограму роботи модуля.

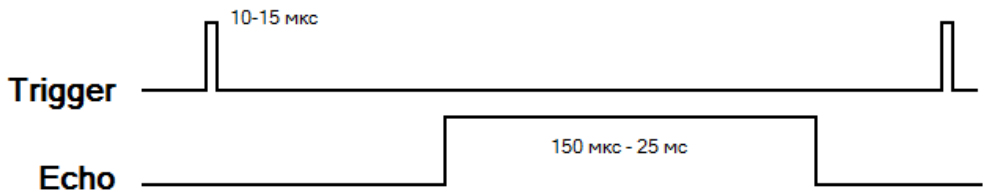


Рис.12.2. Часова діаграма роботи датчика

Після випромінювання сигнальних пакетів, на лінії Echo з’являється рівень логічної одиниці. При одержанні відповіді у вигляді еха, датчик знімає логічну одиницю з лінії Echo. Знаючи час випромінювання і прийому відповіді, одержуємо тривалість сигналу в “польоті”. Маючи інформацію про швидкість звуку в середовищі, легко одержати відстань до об'єкта.

Швидкість звуку в повітрі обчислюється по формулі:

$$V=(331+0,60T) \text{ м/с}$$

де Т – температура в ^{o}С.

Наприклад, при 20 ^{o}С: V=[331+(0,60) *(20)] м/с = 343 м/с.

12.1.2. Serial - Інтерфейс

Набір функцій Serial служить для зв'язку пристрою Arduino з комп'ютером або іншими пристроями, що підтримують послідовний інтерфейс обміну даними. Всі плати Arduino мають хоча б один послідовний порт (UART). Для обміну даними Serial використовує цифрові лінії вводу/виводу 0 (RX) і 1 (TX), а також USB порт. Важливо враховувати, що якщо використовуються функції Serial, то не можна одночасно використати лінії 0 і 1 для інших цілей, наприклад, для підключення чутливих елементів.

Середовище розробки Arduino має вбудований монітор послідовного інтерфейсу (Serial monitor).

12.1.3. Опис основних функцій

Serial.begin() – ініціює послідовне з'єднання і задає швидкість передачі даних у біт/с(бод). Для обміну даними з комп'ютером використовуються наступні значення: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200 бод.

Приклад: Serial.begin(9600);

Для початку обміну даними необхідно запустити монітор порта натисканням кнопки Serial monitor і виставити ту ж швидкість зв'язку (baud rate), з якою викликана функція begin().

Serial.print() – передає текстові дані використовуючи ASCII. Зазначимо, що функція дозволяє виводити данні різних типів, проте вони будуть перетворені у символи із таблиці ASCII кодів.

Приклад: Serial.print("Hello world");

Serial.println() – те ж, що і Serial.print, проте записане значення буде виведено з нового рядка. Аналог дії керуючого символу “\n”.

12.2. Завдання на виконання

12.2.1. Завдання №1. Ультразвуковий дальномір

Алгоритм роботи: за допомогою модуля HC-SRO4, вивести відстань до об'єкта на екран ПК.

Для цього необхідно зібрати схему наведену на рис. 12.2. Набрати та проаналізувати наведений нижче код:

```
int TP = 4;//пін для лінії trig

int EP = 7;//пін для лінії echo

void setup() {
  Serial.begin(9600);
   pinMode(TP, OUTPUT);
   pinMode(EP, INPUT);
}

void loop() {
 digitalWrite(TP, LOW);
  delayMicroseconds(2);
  digitalWrite(TP, HIGH);
  delayMicroseconds(10);
  digitalWrite(TP, LOW);
  long mKs = pulseIn(EP, HIGH); // час відгуку в мкс
  float cm=  mKs /29.412 / 2 ;  //швидкість свуку 340 м/с, або 29,412 мкс/см,
  //з врахуванням часу польоту в 2 сторони, ділимо навпіл
  Serial.print("Vidstn do objektu = ");
  Serial.print(cm);
  Serial.println("cm");

 delay(200);
}
```

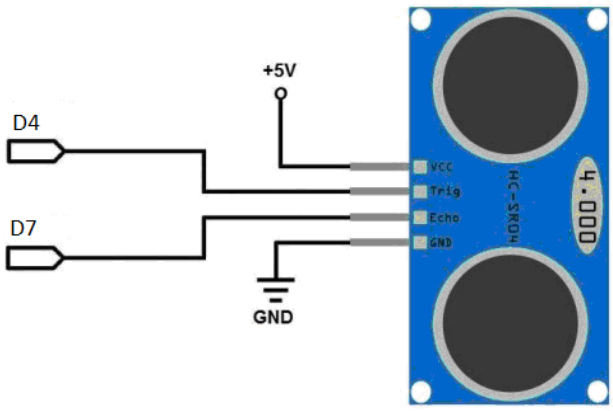


Рис. 12.2. Схема до завдання 1

Для парних варіантів: вивести значення до об’єкту у дюймах.

Для непарних варіантів: вивести значення до об’єкту у сантиметрах при температурі повітря -20 °С та +30 °С.

12.2.2. Завдання №2. Парктронік

Пристрій буде моделювати роботу автомобільного парктроніка з індикацією відстані на світлодіоді індикатори. Алгоритм роботи наступний: якщо відстань більше 2м, то горить зелений світлодіод. Якщо відстань від 2м до 1м – жовтий. Якщо від 1м до 20 см – горить червоний. Якщо менше 20 см - червоний мигає. Для цього необхідно зібрати схему із рис. 12.3, створити код скетчу та перевірити його працездатність.

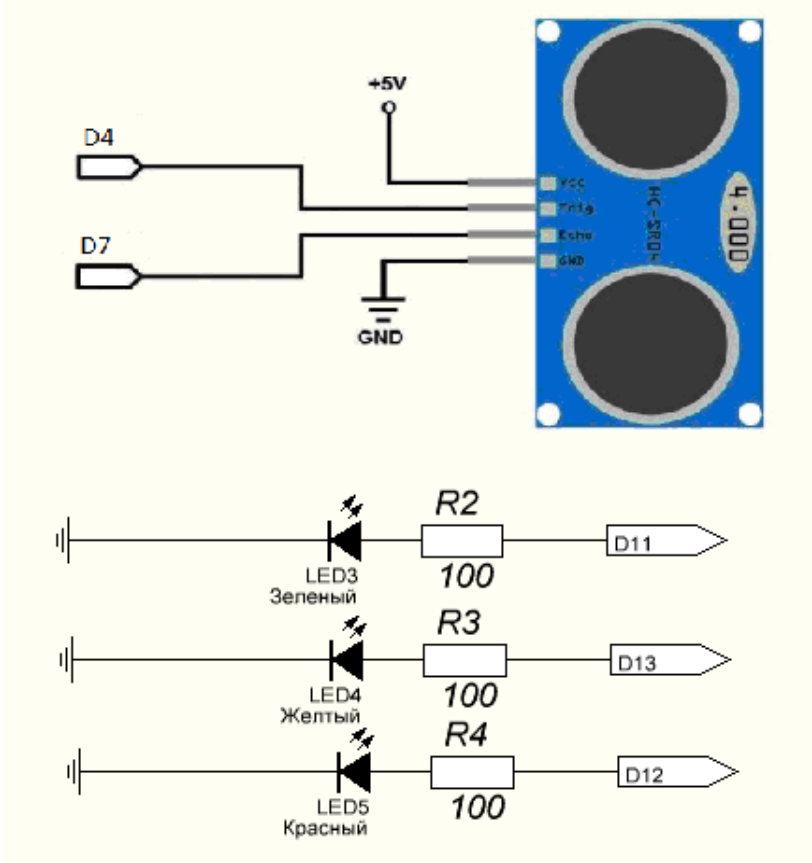


Рис. 12.3. Схема до завдання 2

12.3. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.
2. Створити код програми у відповідності до завдання із пп. 12.2.1.
3. Завантажити створений скетч в МК, переконатися в працездатності програми при різних відстанях, та направляючи далекомір на об’єкти під різними кутами.
4. Створити код програми у відповідності до завдання із пп. 12.2.2.
5. Завантажити записаний скетч в МК, переконатися в працездатності програми.

Зміст звіту

1. Перенести написаний код для завдань №1 і №2.
2. Зробити висновки по роботі.

Питання для самоконтролю

1. Принцип роботи ультразвукового дальноміра.
2. Для чого подається високий рівень на ніжку Trig?
3. Як розрахувати відстань до об'єкта?
4. Які недоліки та переваги ультразвукових далекомірів, як їх позбутися?
5. Для чого використається Serial-інтерфейс?
6. У чому різниця функцій Serial.print() і Serial.println()?
7. Чому швидкість, що зазначається у функції Serial.begin() повинна бути однаковою із швидкістю, що задається в моніторі порта?
8. Чи можна використовувати цифрові входи D0 і D1, якщо відбувається обмін даними із ПК, чому?

ЛАБОРАТОРНА РОБОТА № 13.
РОБОТА З АКСЕЛЕРОМЕТРОМ

Мета роботи. Вивчити принцип роботи акселерометра ADXL-335. Засвоїти принципи побудови пристроїв на базі ADXL-335.

13.1. Теоретичні відомості

13.1.1. Акселерометр ADXL-335

У лабораторній роботі буде розглянуто акселерометр ADXL-335, загальний вид якого наведений на рис. 13.1. ADXL335 – це мікроелектромеханічний (МЕМС), малоспоживаючий, повнофункціональний трьохосьовий акселерометр з аналоговим вихідним сигналом. Мінімальний діапазон вимірюваних прискорень становить ±3 g із смугою пропускання до 1.6 кГц. Акселерометр чутливий до статичного прискорення, що викликане силою тяжіння. Отже, може бути використаний для вимірювання кутів нахилу, а також динамічного прискорення, викликаного рухом, ударами чи вібрацією.

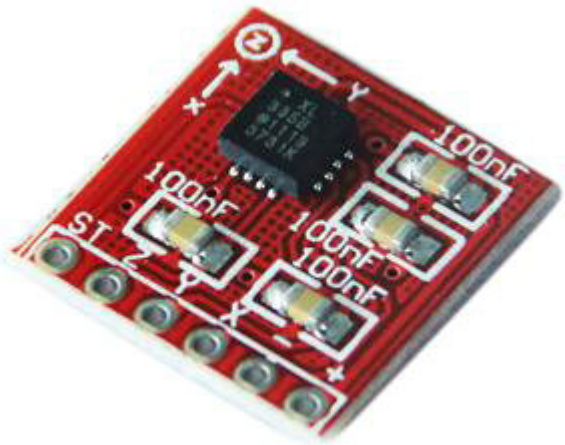


Рис. 13.1. Модуль акселерометра ADXL-335

Ширину смуги пропускання акселерометра можна регулювати за допомогою додавання на виходи X_{OUT}, Y_{OUT} та Z_{OUT} конденсаторів, змінюючи смугу пропускання у діапазоні від 0.5 Гц до 1.6 кГц для осей X та Y, та до

0.5 кГц для осі Z. ADXL335 випускається в мініатюрному пластиковому корпусі ромірами 4 мм × 4 мм × 1.45 мм. з 16-ма виводами.

ADXL335 знайшов застосування у таких галузях техніки та електроніки, як малоспоживаючі схеми вимірювання нахилу, вимірювання параметрів руху, мобільні пристрої, ігрові системи, системи віброзахисту пристроїв, стабілізація зображень, спортивні гаджети.

13.1.2. Основні характеристики

- 3 осі вимірювань із мінімальним перехресним зв'язком.
- Мініатюрний пластиковий корпус LFCSP, 4 мм × 4 мм × 1.45 мм.
- Максимальний струм споживання: 350 мкА;
- Живлення від 1.8 В до 3.6 В.
- Максимальні ударні навантаження до 10000 g.
- Висока температурна стабільність.
- Налаштування ширини смуги пропускання за допомогою одного конденсатора на вісь.
- Відповідає вимогам RoHS/WEEE.

За умови того, що живлення акселерометра ADXL-335 знаходиться у діапазоні 1.8 - 3.6 В, можна провести підключення датчика до плати Arduino Uno, не використовуючи додаткових узгоджуючи пристроїв. Схема підключення каналів живлення і інформаційних ліній представлені на рис. 13.2 та у таблиці 13.1. Контакт Vcc живлення акселерометра може бути під'єднаний або до ніжки живлення 3.3V, або до ніжки AREF, із можливістю задання опорної напруги для корекції похибок акселерометра.

Таблиця 13.1. Відповідність контактів ADXL-335 і платформи Arduino

ADXL335	Arduino UNO
X	A0
Y	A1
Z	A2
Vcc	3V3 (Aref)
GND	GND

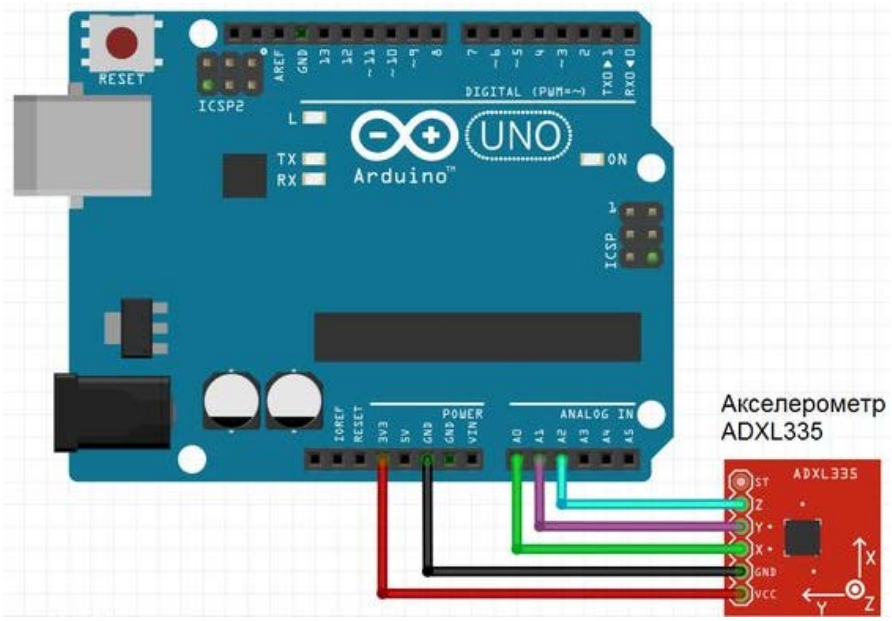


Рис. 13.2. Схема підключення акселерометра ADXL-335 і платформи Arduino

13.2. Завдання на виконання

13.2.1. Завдання №1. Отримати дані з акселерометра

Алгоритм роботи: вивести кути нахилу платформи із датчиком відносно горизонталі та вертикалі через Serial-інтерфейс.

а) Підключити акселерометр по схемі на рис. 13.1.

б) Створити і зберегти новий проект.

в) Написати код програми, використовуючи функції з попередніх лабораторних робіт;

г) Встановити відповідність отриманих даних коду АЦП та куту нахилу.

д) Вивести інформацію через Serial-інтерфейс.

13.2.2. Завдання №2. Цифровий рівень

Пристрій представляє собою індикатор нахилу, інформація про нахил виводиться на 3 світлодіоди. Алгоритм роботи: коли платформа знаходиться в горизонтальному положенні – світиться жовтий світлодіод. Якщо кут нахилу більше $+10^0$ – жовтий світлодіод вимикається і починає світитися зелений

світлодіод. Якщо кут нахилу більше -10^0 – червоний. Чим більший кут нахилу, тим яскравіше світяться світлодіоди. Схема включення світлодіодів представлена на рис.10.2. Створити код скетчу в середовищі розробки та перевірити його працездатність.

Для парних варіантів: Задіяти вісь X.

Для непарних варіантів: Задіяти вісь Y.

13.3. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.
2. Створити код програми у відповідності до завдання із пп. 13.2.1.
3. Завантажити створений скетч в МК, переконатися в працездатності програми при різних кутах нахилу і для різних осей.
4. Створити код програми у відповідності до завдання із пп. 13.2.2.
5. Завантажити записаний скетч в МК, переконатися в працездатності програми.

13.4. Зміст звіту

1. Перенести написаний код для завдань №1 і №2.
2. Дати рекомендації по підвищенню точності показів акселерометра.
3. Зробити висновки по роботі.

13.5.Питання для самоконтролю

1. Принцип роботи акселерометра.
2. Яка напруга живлення акселерометра?
3. Які ударні навантаження витримує акселерометр?
4. Для чого використається Serial-інтерфейс?
5. Як встановлюється залежність кута нахилу платформи від даних на виході акселерометра?
6. Як підвищити точність визначення кута нахилу?
7. Які типи акселерометрів ви знаєте?

ЛАБОРАТОРНА РОБОТА № 14

ЕКРАН NOKIA 5110

Мета роботи. Ознайомитись з принципом роботи та підключення LCD дисплею Nokia 5110, навчитись створювати пристрої із використанням LCD дисплею.

14.1. Теоретичні відомості

14.1.1. Екран Nokia 5110

Монохромний LCD дисплей Nokia 5110 (рис.14.1) з роздільною здатністю 84x48 пікселя та діагоналлю 1.6 дюйма, використовується для виведення графічної інформації від контролерів, чутливих елементів та інших периферійних приладів.

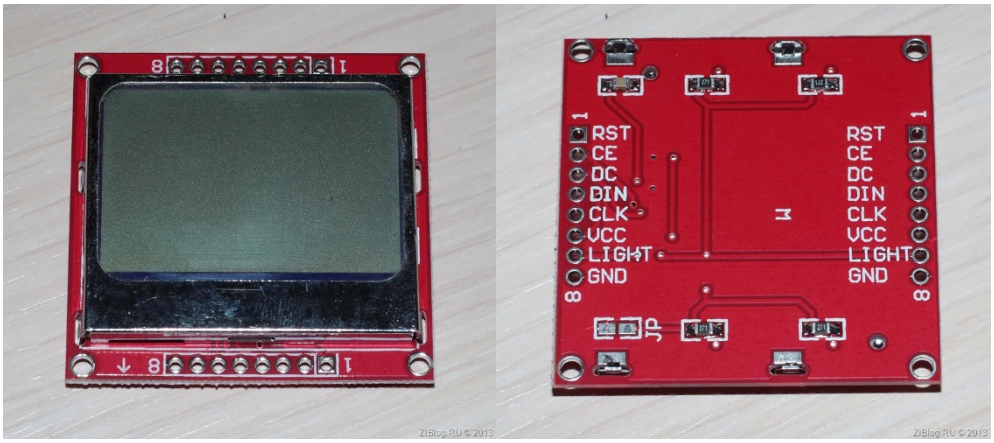


Рис. 14.1. Загальний вигляд екрану Nokia 5110

Графічний LCD дисплей 84x48 Nokia 5110 керується контролером PCD8544, має дві колодки контактів для підключення дисплею до живлення (2,7 – 3,3В) та керуючих сигналів від мікроконтролера. Для взаємодії із МК використовується модифікований SPI інтерфейс. Назви та функції кожного з виходів наведемо нижче:

RST (external reset input) - лінія для перезавантаження дисплею;

CE (chip enable) - дозволяє або забороняє введення даних в контролер дисплея, вибір контроллера екрана при наявності декількох дисплеїв;

DC (data command) – вибір перемикання між вводом даних, або команд;

DIN (serial data input) – лінія введення даних;

CLK (serial clock input) – лінія тактування для послідовного інтерфейсу SPI;

VCC (supply voltage) – живлення контролера дисплея 2,7-3,3 В;

Light (LCD supply voltage) – підсвічування дисплею, для ввімкнення необхідно приєднати до загального контакту;

Схема підключення контактів LCD дисплею до відладочної плати наведена на рис. 14.2. З врахуванням того, що контролер дисплею PCD8544 працює із 3-вольтовими логічними рівнями, то для стабільної роботи дисплею пропонується всі інформаційні лінії підключати через опори 1к-4.7к Ом.

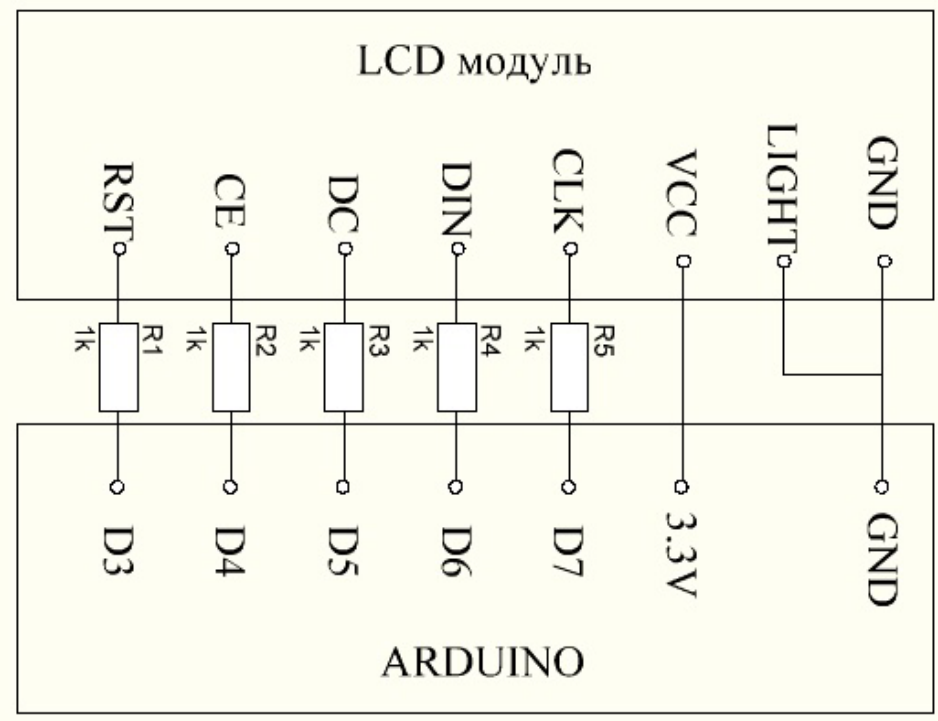


Рис.14.2. Схема підключення контактів дисплею до плати Arduino

14.1.2. Код програми та опис основних функцій

Наведемо тестовий код, що одноразово виводить на екран напис «Hello World». Для формування послідовності пікселів у вигляді символів використовується система ACSII. У наведеному коді вона представлена частково, лише для пояснення основного принципу.

```
#define PIN_SCE   7
#define PIN_RESET 6
#define PIN_DC    5
#define PIN_SDIN  4
#define PIN_SCLK  3

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48

static const byte ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
……
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
……..
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
……
};

void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {    LcdWrite(LCD_D, ASCII[character - 0x20][index]);  }
  LcdWrite(LCD_D, 0x00);
}

void LcdClear(void)
{  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {    LcdWrite(LCD_D, 0x00);  }}

void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x21 );  // LCD Extended Commands.
  LcdWrite(LCD_C, 0xB1 );  // Set LCD Vop (Contrast).
  LcdWrite(LCD_C, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite(LCD_C, 0x14 );  // LCD bias mode 1:48. //0x13
  LcdWrite(LCD_C, 0x0C );  // LCD in normal mode.
```

```
  LcdWrite(LCD_C, 0x20 );
  LcdWrite(LCD_C, 0x0C );
}

void LcdString(char *characters)
{  while (*characters)
  {    LcdCharacter(*characters++);  }}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void setup(void)
{
  LcdInitialise(); /ініціалізація екрану
  LcdClear();/очищення екрану
  LcdString("Hello World!");  /вивід рядка
}

void loop(void)
{}
```

LcdCharacter – функція для виведення на екран символів із таблиці ASCII кодів.

LcdClear – функція очищення даних на екрані.

LcdInitialise – функція ініціалізації екрану, за допомогою якої можна змінити параметри відображення на екрані. *За замовчуванням не змінюється.*

LcdString – функція, що циклічно виводить символи, викликаючи функцію **LcdCharacter**, тобто виведення рядка.

LcdWrite – має два аргументи. Перший – бульове значення 0 або 1, не записувати і записувати відповідно. Другий аргумент шістнадцяткове дворозрядне число, що формує 8 пікселів (стовпчик).

Тепер детально розглянемо, яким чином формується символ.

Наприклад, код LcdCharacter('H'); виведе на екран латинську букву N. Код тотожний семикратному виклику функції LcdWrite із наступними значеннями аргументів:

```
LcdWrite(1,0x00);
LcdWrite(1,0x7f);
LcdWrite(1,0x04);
LcdWrite(1,0x08);
LcdWrite(1,0x10);
LcdWrite(1,0x7f);
LcdWrite(1,0x00);
```

Якщо провести аналіз роботи функції, то звернення буде відбуватись до строки ACSII коду {0x7f, 0x04, 0x08, 0x10, 0x7f}, – що відповідає літері «N» (рис. 14.3).

Для розшифрування коду потрібно перевести даний код в двійкову систему числення.

7f – 01111111 – зафарбовані всі лунки 2-го стовпчика, крім 1-ї;

04 – 00000100 – зафарбована третя лунка зверху 3-го стовпчика;

08 – 00001000 – зафарбована четверта лунка зверху 4-го стовпчика;

10 – 00010000 – зафарбована п’ята лунка зверху 5-го стовпчика;

7f – 01111111 – зафарбовані всі лунки 6-го стовпчика, крім 1-ї.

Перший та останній стовпчики завжди пусті. Таким чином формується відстань між символами.

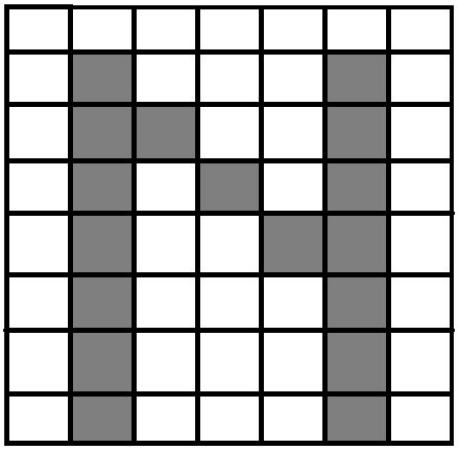


Рис. 14.3 Літера “N” як послідовність пікселів, виведених на екран

Такий підхід дозволяє не лише використовувати сформовані у таблиці символи, а й формувати свої, наприклад, кириличні, спецсимволи, або малюнки.

14. 2. Завдання на виконання

14.2.1. Завдання 1. Виведення кириличних символів

На екран вивести власне ім’я та прізвище українською мовою. *Наприклад:* Олексій ПАВЛОВСЬКИЙ. Для цього необхідно:

а) Підключити екран до плати Arduino по схемі на рис. 14.2.

б) Завантажити базовий код програми (видається викладачем).

в) За допомогою алгоритму, що описаний в пп. 14.1.2, сформувати коди символів.

г) Створити код програми для виведення прізвища.

14.2.2. Завдання 2. Виведення даних з чутливих елементів

Вивести на екран дані із чутливих елементів.

Для парних варіантів: чутливий елемент – акселерометр, вивести кут нахилу по осям X та Y (Приклад: X = -24^0

Y = 45^0).

Для непарних варіантів: чутливий елемент – ультразвуковий дальномір, вивести інформацію про відстань до об’єкта у сантиметрах і відповідний напис (Приклад: vidstan` do objektu = 50sm).

Для цього необхідно:

а) підключити чутливі елементи у відповідності до свого варіанту;

б) завантажити базовий код програми (видається викладачем) та змінити його для виконання поставленої задачі використовуючи функції з попередніх лабораторних робіт;

в) переконатися у працездатності програми.

14.3. Порядок виконання лабораторної роботи

1. Запустити ПЗ Arduino, створити і зберегти новий скетч.
2. Створити код програми у відповідності до завдання із пп. 14.2.1.

3. Завантажити створений скетч в МК, переконатися в працездатності програми.

4. Створити код програми у відповідності до завдання із пп. 14.2.2.

5. Завантажити записаний скетч на плату, переконатися в працездатності програми, особливо звернути увагу на коректність виведених із ЧЕ показів.

Зміст звіту

1. Перенести написаний код у звіт.
2. Зробити висновки по роботі.

Питання для самоконтролю

1. Який принцип формування зображення на дисплеї?
2. Який інтерфейс використовує дисплей Nokia 5110?
3. Назвіть основні характеристики дисплею Nokia 5110.
4. Опишіть призначення основних функцій програми.
5. Яка система числення використовується в ASCII?
6. Як розшифрувати символи по коду ASCII?
7. Як створити символ, якого немає в таблиці ASCII?

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Мікропроцесорна техніка. Частина 1. Мікроконтролер ARM7: Методичні вказівки до виконання лабораторних робіт для студентів напряму підготовки 6.051003 – "Приладобудування" / Уклад.: О.М. Павловський – К.: НТУУ "КПІ", 2015. – 36 с.

2. Мікропроцесорна техніка. Частина 2. Платформа Arduino: Методичні вказівки до виконання лабораторних робіт для студентів напряму підготовки 6.051003 – "Приладобудування" / Уклад.: О.М. Павловський, Д.О. Півторак – К.: НТУУ "КПІ", 2015. – 37 с.

3. Редькин П. П. 32/16-битные микроконтроллеры ARM7 семейства AT91SAM7 фирмы Atmel. Руководство пользователя.- М.: Издательский дом “Додэка-XXI”, 2008.- 704 с.: ил..

4. Embedded Development Tools. Datasheet[Електронний ресурс].- режим доступу: https://www.keil.com/dd/docs/datashts/atmel/at91sam7x128_256_pc.pdf

5. Микроконтроллеры ARM (ARM7 и ARM9) [Електронний ресурс].- режим доступу: http://www.phyton.ru/pages/page41.html. – Назва з екрана.

6. Шпак Ю. А. Программирование на языке С для AVR и PIC микроконтроллеров.- М.: Издательский дом “МК-Пресс”, 2009.

7. Артюхов В.Г. Проектирование микропроцессорной електронно-вычислительной аппаратуры: справочник / В.Г. Артюхов, А.А. Будняк, В.Ю. Лапий. – К.:Технічна література, 1988. – 263 с.

8. MCU STM32F303VCT6 Datasheet [Електронний ресурс].- режим доступу: https://www.st.com/en/microcontrollers-microprocessors/stm32f303vc.html

9. STM32F4DISCOVERY. User Manual [Електронний ресурс].- режим доступу: https://www.st.com/en/evaluation-tools/stm32f4discovery.html

10. Новиков Ю.В. Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов. – Интернет-университет информационных технологий: ИНТУИТ.ру, БИНОМ, Лаборатория знаний, 2008. – 358 с.

11. Баранов В.Н., Применение Микроконтроллеров AVR: схемы алгоритмы программы. — М.: Издательский дом «Додэка-XXI», 2006. – 186с.

12. Евстифеев А. В., Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL, 5-е изд., стер. — М.: Издательский дом «Додэка XXI», 2008. — 560 с.

13. Офіційний сайт Arduino. Tutorials [Електронний ресурс]. - режим досту-пу:https://www.arduino.cc/en/Tutorial/HomePage

14. Навчальні матеріали по Arduino - огляд [Електронний ре-сурс]. - режим доступу: http://downloads.oreilly.com/make/arduinoMAKE07.pdf . – Назва з ек-рана.

15. Цикл статей по Arduino [Електронний ресурс]. - режим досту-пу: www.arduino.cc ; amperka.ru – Назва з екрана.