Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

# Побудова спеціалізованих комп'ютерних систем на мікроконтролерах ARM-7

Методичні вказівки до виконання лабораторних робіт
з дисципліни «Програмне забезпечення
спеціалізованих комп'ютерних систем»
для студентів спеціальності
121 «Інженерія програмного забезпечення»

Обговорено та рекомендовано
на засіданні кафедри електроніки, автоматики, робототехніки
та мехатроніки.
Протокол № 4 від 29.11.2019 р.

Чернігів – 2019

Побудова спеціалізованих комп’ютерних систем на мікроконтролерах ARM-7. Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмне забезпечення спеціалізованих комп’ютерних систем» для студентів спеціальності 121 «Інженерія програмного забезпечення». – Чернігів: НУЧП, 2019. – 88 с.

Укладач: Войтенко Володимир Павлович, канд. техн. наук, доц.

Відповідальний за випуск – Денисов Юрій Олександрович, докт. техн. наук, проф., завідувач кафедри електроніки, автоматики, робототехніки та мехатроніки

Рецензент: Ревко Анатолій Сергійович, канд. техн. наук, доц., доцент кафедри електроніки, автоматики, робототехніки та мехатроніки Національного університету «Чернігівська політехніка»

## Зміст

## Перелік умовних скорочень

МК – мікроконтролер.
МП – мікропроцесор (мікропроцесорний).
МПС – мікропроцесорна система.
ОЗП – оперативний запам‘ятовуючий пристрій.
РКД – рідинно-кристалічний дисплей.
ШІМ – широтно-імпульсна модуляція.
ФНЧ – фільтр нижніх частот.
ARM – мікроконтролери сімейства ARM.
ARM-7, ARM7-МК – мікроконтролери сімейства ARM-7.
ISR – Interrupt Service Routine (підпрограма обробки переривань).

## Вступ

Розробки в галузі інтернету речей та штучного інтелекту вимагають все більшої продуктивності та обсягів резидентної пам'яті від мікроконтролерів і, водночас, доступної ціни на ці специфічні радіоелектронні компоненти. Британська фірма *Advanced* (або *Acorn*) *RISC Machines* на початку 1990-х років запропонувала ринку ліцензію топології мікропроцесорного ядра *ARM*, доопрацьовані версії якого сьогодні використовуються в мікроконтролерах багатьох відомих виробників: *Atmel*, *Cirrus Logic*, *FreeScale*, *Intel*, *Marvell*, *NXP*, *Oki*, *Qualcomm*, *Samsung*, *Sony Ericsson*, *Texas Instruments* та ін.

Найголовніша ознака ядра *ARM* – це досить велика довжина машинного слова (*32* розряди), що, поряд із широким набором периферійних модулів традиційних мікроконтролерів (МК) та підтримкою великих об'ємів адресного простору, надає суттєвих переваг в таких сферах, як пристрої промислової автоматики, медична апаратура, торгівельні автомати, телекомунікаційні засоби, бездротовий зв'язок, перетворювачі протоколів тощо. За рахунок масовості випуску та наявності найсучасніших мікроелектронних технологій у виробників для цих МК вдається утримувати украй невисоку ціну, яка у багатьох випадках наближається до 8-розрядних приладів. Таким чином, застосовуючи МК з ядром *ARM*, розробник дістає унікальну можливість значно збільшити функціональність своєї системи, не виходячи за рамки обмеженого бюджету.

Дані методичні вказівки призначені для самостійної підготовки здобувачів вищої освіти до виконання лабораторних робіт на навчальному стенді *LPC2148 Education Board* фірми *Embedded Artists* і присвячені вивченню архітектури та практичному дослідженню характерних особливостей мікропроцесорних систем на базі МК *LPC2148* фірми *NXP*.

Процес підготовки до виконання завдань потребує обов'язкової самостійної роботи з рекомендованою літературою та лекційним матеріалом. Після теоретичної частини наведено список контрольних питань, за допомогою яких оцінюється ступінь опанування теоретичного матеріалу і готовність здобувача вищої освіти до виконання лабораторної роботи. Далі наводиться перелік завдань, які потрібно вирішити в ході виконання роботи.

Результати виконання робіт оформлюються у вигляді звітів на аркушах формату *A4*. Звіт повинен містити таку інформацію:

1) короткі відомості про об'єкт вивчення (*15…20* рядків), в тому числі – схему підключення певного периферійного модуля;

2) рукописні відповіді на контрольні запитання;

3) схему, текст програми для МК і стислий опис алгоритму її дії;

4) висновки щодо отриманих результатів.

Пункти 1 та 2 оформлюються і показуються викладачеві на початку заняття, що є умовою допуску до виконання роботи в навчальній лабораторії. Весь звіт надається безпосередньо на захист роботи.

# 1 Лабораторна робота №1. Основи програмування МК *ARM-7*

***Мета роботи***: дослідити побудову, можливості навчального стенду та ознайомитися з методами завантаження програми до МК ARM-7.

## *1.1 Призначення і склад навчального стенду LPC2148 Education Board*

Навчальний стенд *LPC2148 Education Board* фірми *Embedded Artists* призначений для ознайомлення з мікроконтролерами *ARM7TDMI* виробництва фірми *NXP*, а також розробки і швидкого запуску пристроїв на базі цих МК. Безпосередньо на платі реалізовані унікальні можливості взаємодії МК з типовими для електронних систем об'єктами керування, що істотно полегшує і підвищує наочність процесу розробки. Крім того, через інтерфейси розширення також доступне підключення різноманітних за функціями дочірніх модулів: *Ethernet, Bluetooth, ZigBee, UART* та *RS485, MP3, LCD* тощо. Для демонстрації можливостей в резидентну пам'ять мікроконтролера занесено демонстраційну програму керування двигуном, виведенням інформації на РКІ, одиничний багатоколірний та матричний світлодіодні індикатори. Під час відлагодження програмного забезпечення для курсових та дипломних проектів також може бути задіяний опціональний *JTAG*-інтерфейс.

Розглянемо особливості стенду. Схему розміщення складових частин навчального стенду *LPC2148 Education Board* наведено на рисунку 1.1.

• Встановлений мікроконтролер *LPC2148* ARM7TDMI-S™ [1] має розрядність 32/16 біт, резидентну *Flash*-пам'ять програм ємністю 512 кБ, ОЗП 32+8 кБ, USB 2.0, годинник реального часу, вісім 10-бітових АЦП, ЦАП, 2 порти UART, два $I^2C$, SPI, два 32-бітові таймери, вісім каналів захвату/зберігання, ШІМ (шість виходів), сторожовий таймер, входи/виходи, сумісні з 5В, тактова частота процесора до 60 МГц (вбудована ФАПЧ з перемноженням до 5) (1 на рисунку 1.1).

• Резонатор 12 МГц для встановлення максимальної тактової частоти 60 МГц (5х) та стандартних швидкостей послідовної передачі (2).

• Резонатор годинника реального часу (RTC) 32768 Гц (3).

• Периферійні пристрої на платі:

– рідинно-кристалічний дисплей 2х16 символів зі світлодіодним підсвічуванням і можливістю відключення (4);

– джойстик (5);

– міст інтерфейсів UART – USB 2.0 для внутрішньосхемного програмування МК через UART #0 (6);

– інтерфейс USB 2.0 (роз'єм mini-B) для зовнішніх приладів (7);

– трибарвний світлодіод з управлінням ШІМ-сигналом (8);

– вісім призначених для користувача світлодіодів (9);

– датчик температури (LM75) на шині $I^2C$ (10);

– кнопка на порту P0.14 (вхід переривання) (11);

– світлодіодна матриця 8х8, керована регістром зсуву SPI (12);

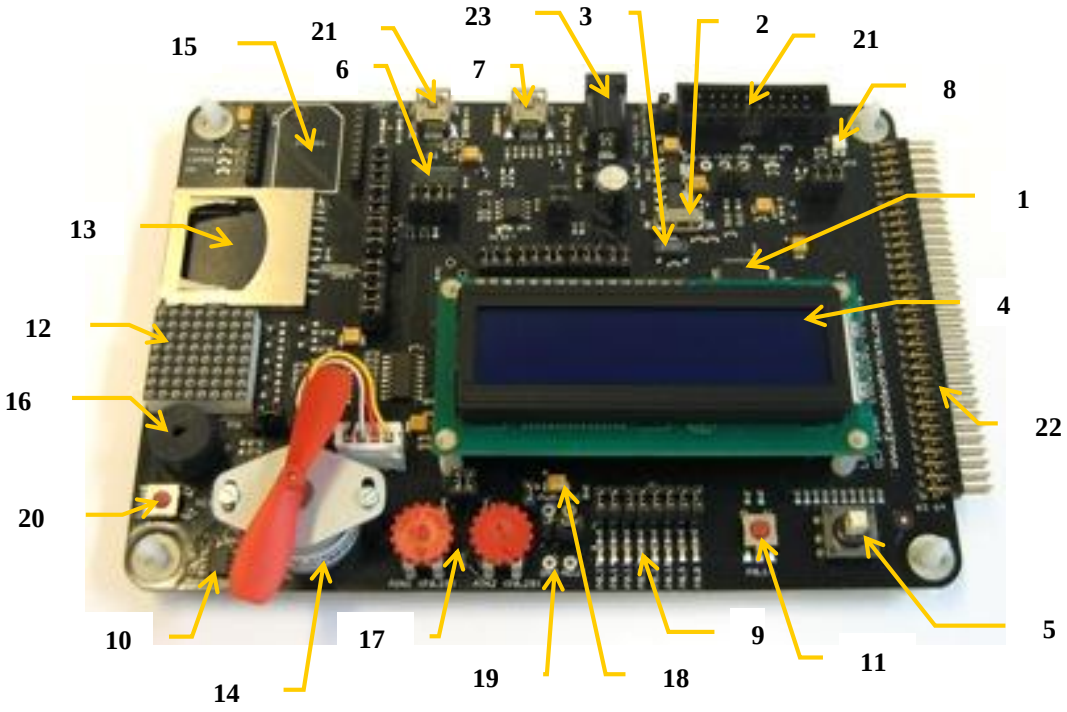



Рисунок 1.1 – Розміщення складових частин LPC2148 Education Board

– інтерфейс карт пам'яті MMC/SD (13);
– кроковий двигун (біполярне керування) (14);
– інтерфейс модуля XBEE™ Max Stream (15);
– п'єзоелектричний зумер (16);
– два аналогові входи з приєднаними потенціометрами (17);
– фільтр нижніх частот для ШІМ-сигналу (18);
– аналоговий вихід (19);
– кнопка скидання (20).

• Роз'єми, що встановлені на платі.

– роз'єм типу USB mini-B для мосту інтерфейсів UART – USB через UART #0 (21). **При виконанні лабораторних робіт до цього роз'єму підключається кабель, через який від персонального комп'ютера здійснюється живлення стенду та завантаження пам'яті МК**;

– роз'єм типу USB mini-B для зв'язку *LPC2148* з зовнішніми приладами (7). Може бути підключений ще один кабель (пристрій);

– роз'єм карт пам'яті MMC/SD (13);

– роз'єм налагоджувального інтерфейсу JTAG (21);

– роз'єм розширення на 64 контакти, на якому доступні всі ніжки вводу/виводу мікроконтролера *LPC2148* (22);

– роз'єм живлення діаметром 2,1 мм (23).

• Пам'ять EEPROM з послідовним доступом $I^2C$ ємністю 2 кбіти для збереження енергонезалежних параметрів.

• Джерело живлення (вхід 4,5...6В або безпосередньо від USB, вихід +3,3В) з супервізором і виходом на зовнішні кола (до 300мА); генерує +5В,

якщо стенд живиться від постійної напруги 9…15 В ≥200 мА через роз'єм 2,1 мм.

• Просте та автоматичне завантаження програми (ISP) через канал мосту UART – послідовний інтерфейс;

• Габаритні розміри плати складають 156х110мм. Використаний чотирьохшаровий матеріал FR-4 для кращої завадостійкості.

***Попередження*. Щоб запобігти пошкодженням стенду від електростатичних розрядів, доторкніться обома руками до металевих частин USB або MMC/SD роз'ємів раніше, ніж до інших компонентів.**

Більш докладну інформацію про *LPC2148 Education Board* розміщено в [2]. Для полегшення практичного застосування стенду під час виконання лабораторних робіт розглянемо особливості підключення до МК деяких периферійних пристроїв.

Через виводи порту МК LPC2148 може протікати струм до 4 мА, що достатньо для безпосереднього керування світлодіодами (СВД, LED) (рисунок 1.2). Світлодіоди червоного кольору підключено до виводів P0.8 – P0.15 МК. Резистори номіналом в 1 кОм обмежують електричний струм на рівні приблизно 2 мА. Засвічування можливе, коли на лінії порту встановлений низький рівень. Кожен з світлодіодів можна індивідуально відключити за допомогою відповідних джамперів.

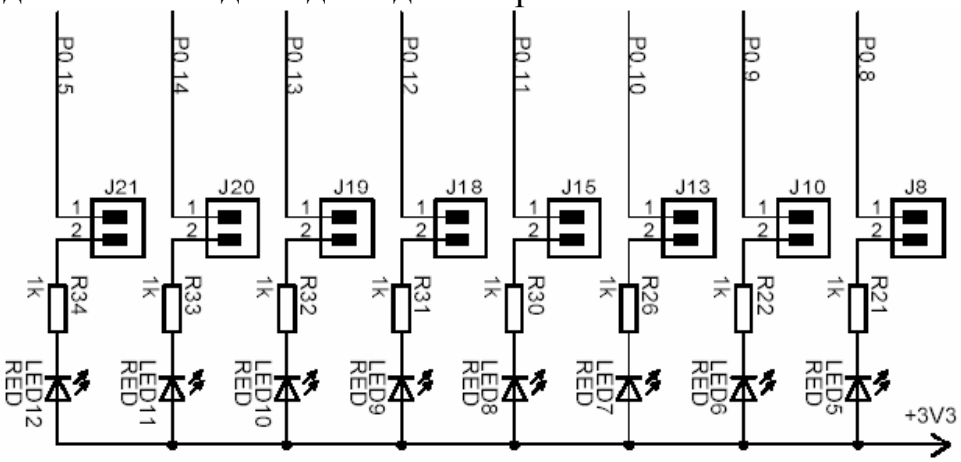



Рисунок 1.2 – Схема підключення одиничних світлодіодів

Сигнал P0.14 може бути конфігурований, як вхід запиту зовнішнього переривання. За допомогою кнопки SW1 на цьому виводі МК можна встановити сигнал низького рівня (рисунок 1.3). Підтягуючий резистор (10 кОм) також потрібний для реалізації функції внутрішньосхемного програмування (ISP). Для того, щоб запустити нормальне виконання програми, після скидання сигнал P0.14 має бути високим. Інакше буде активований внутрішній завантажувач (bootloader).

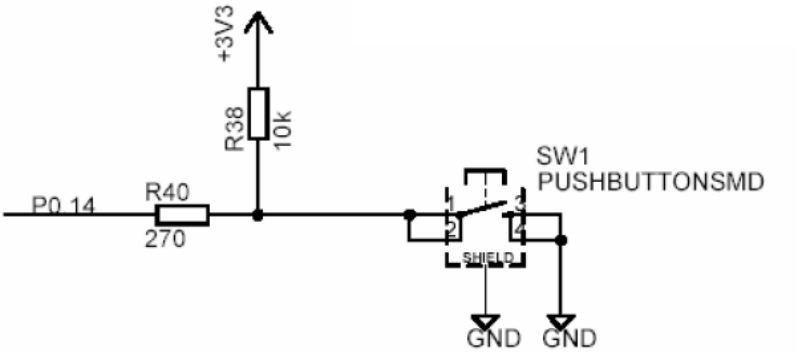



Рисунок 1.3 – Схема підключення кнопки користувача

На платі стенду встановлений джойстик (рисунок 1.4). У ньому вмонтовано п'ять внутрішніх вимикачів, по одному для чотирьох напрямів та один центральний типу кнопки. Всі виводи МК (P0.16 – P0.20) можуть бути запрограмовані або безпосередньо як входи переривання EINTx кожний, або опосередковано через вивід МК CAPture, який, в свою чергу, може генерувати переривання.

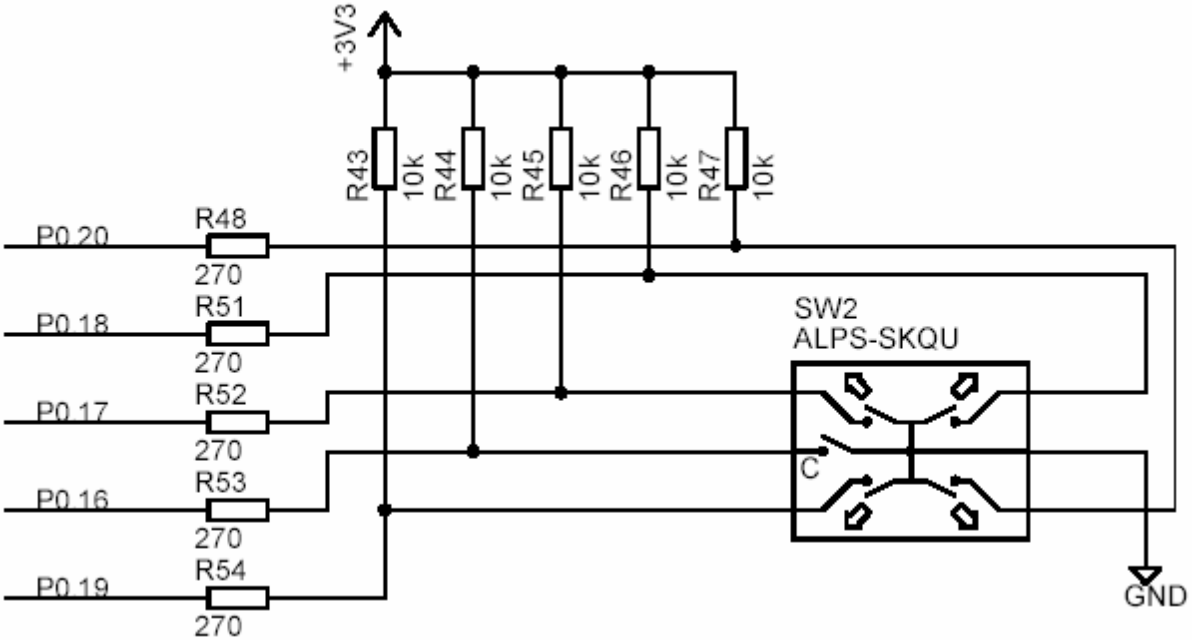



Рисунок 1.4 – *Схема підключення джойстика*

В розпорядженні користувача є повнокольоровий RGB-світлодіод, яким можна керувати за допомогою сигналів ШІМ (широтно-імпульсна модуляція, PWM). Змінюючи ширину імпульсу, можна керувати яскравістю кожного з кольорів. Підключення сигналів здійснено наступним чином:

- червоний СВД керується сигналом P0.7 – PWM2
- синій СВД керується сигналом P0.8 – PWM4
- зелений СВД керується сигналом P0.9 – PWM6

Сигнали ШІМ є альтернативними для виводів та генеруються резидентним периферійним пристроєм – блоком PWM у складі МК. Рисунок 1.5 ілюструє схему вузла RGB-світлодіоду.

Зазначимо, що керування світлодіодами здійснюється від джерела живлення VIN = +5 В тому, що пряме падіння напруги на синьому світлодіоді звичайно наближається до 3,5 В, що більше, ніж напруга живлення МК LCP2148 3,3 В.

RGB-світлодіод можна відключити від МК, якщо видалити три перемички на J16. Крім того зазначимо, що сигнали P0.7 – P0.9 використовуються також в інших частинах стенду, наприклад для того, щоб керувати зумером, двигуном, аналоговим виводом, модулем XBEE, а також індивідуальним світлодіодами (рисунок 1.2).

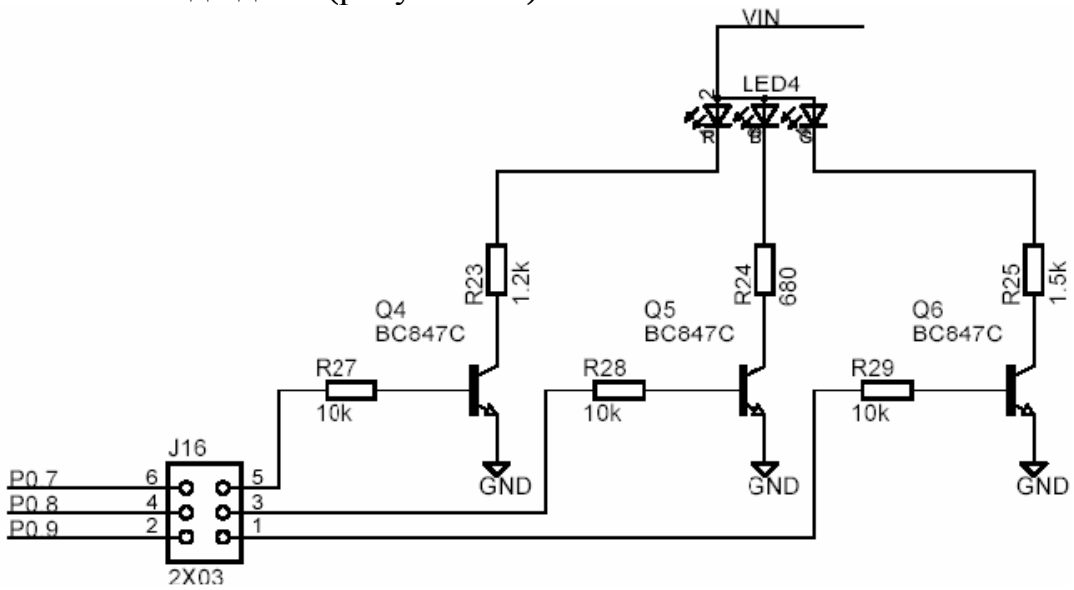



Рисунок 1.5 – Схема вузла трикольорового світлодіоду

На стенді *LPC2148 Education Board* також встановлений маленький п'єзоелектричний зумер (рисунок 1.6). Подаючи на вивід МК P0.7 низький рівень сигналу, ми забезпечим протікання струму через зумер і почуємо відносно різкий тон однієї частоти.

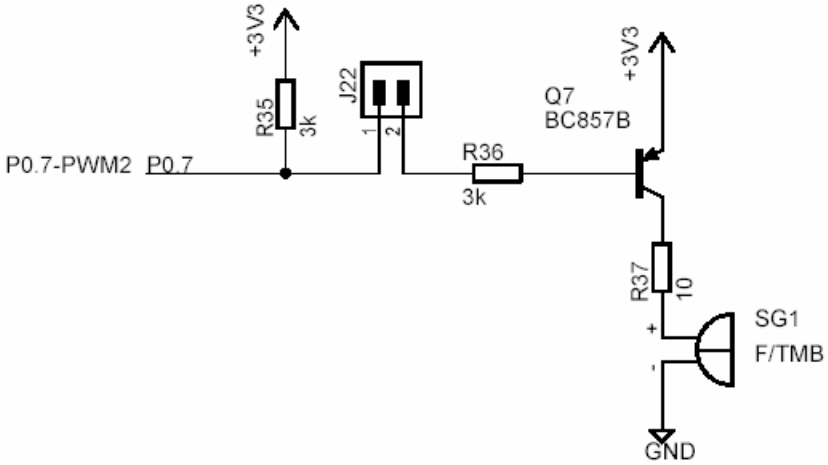



Рисунок 1.6 – *Схема вузла зумера*

Для того, щоб промодулювати зумер коливаннями різних частот, можна задіяти альтернативну функцію ШІМ виводу P0.7 (сигнал PWM2).

За допомогою зміни ширини імпульсу можна змінити тільки гучність звуку, але ж ніяк не частоту. Для того, щоб змінити частоту сигналу зумера, треба змінювати частоту ШІМ. Зумер можна відключити, якщо видалити перемичку на перехіднику J22. Це зроблено за замовчанням, оскільки звук зумера може бути досить подразнюючим.

Зауважимо, що сигнал P0.7 використовують декілька інших пристроїв стенду, наприклад, червоний світлодіод в трикольоровому світлодіоді або аналоговий вивід.

На рисунку 1.7 проілюстровано підключення термодатчика LM75 до інтерфейсу I²C. Зауважимо, що для того, щоб підключити інтерфейс I²C до LPC2148, мають бути встановлені перемички на J25 (див. схемотехніку скидання та I²C-EEPROM, рисунок 1.8). Для детального опису процедури зчитування температури треба ознайомитися з технічним описом LM75 [3].

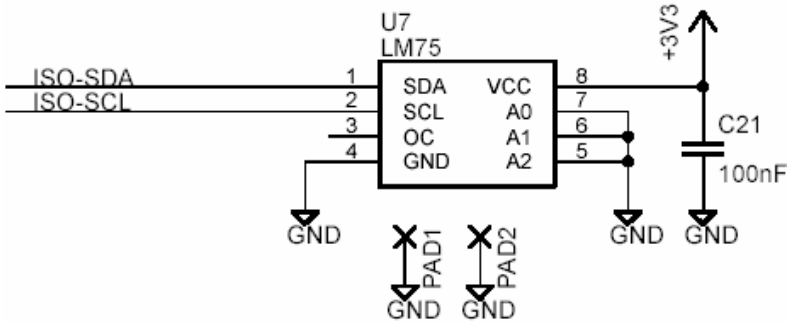



Рисунок 1.7 – *Схема вузла термодатчика*

Генерація сигналу скидання здійснюється мікросхемою змішаного сигналу *CAT1025* виробництва *Catalyst Semiconductor* [4]. Сигнал скидання підтримується активним (тобто низького рівня), доки напруга живлення +3,3 В лежить поза припустимих меж. Типова тривалість скидання складає 200 мс. Вивід сигналу скидання є виводом типу відкритий колектор/відкритий стік. Генератором скидання може також керувати зовнішнє джерело скидання. На рисунку 1.8 показано кнопку скидання (SW1), а також обидва сигнали, доступні на з'єднувачі розширення RES_IN та RES_OUT.

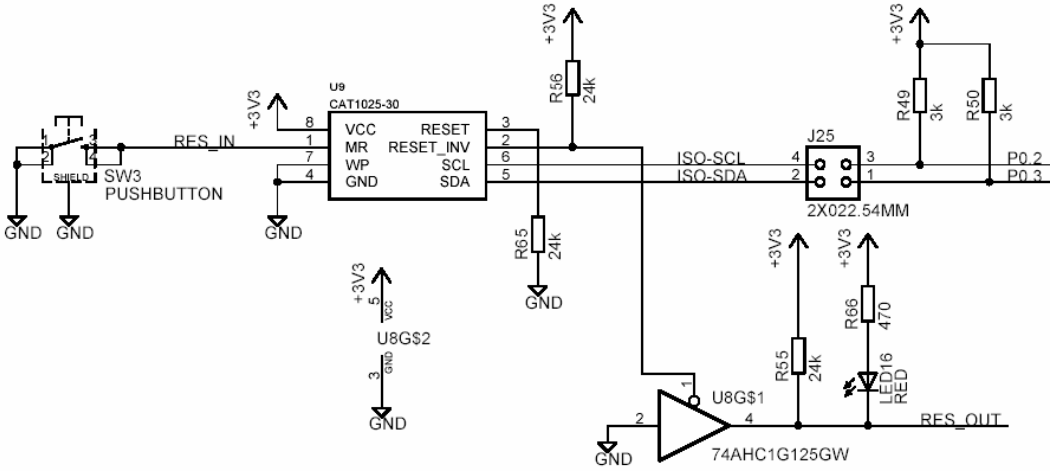



Рисунок 1.8 – Схема вузла скидання та I²C E²PROM

Зауважимо, що зовнішній драйвер цих сигналів має бути відкритим колектором/стоком. Коли сигнал скидання активний, засвічується черво-

ний світлодіод.

В МК LPC2148 є два резидентних канали зв'язку $I^2C$. Канал #0 використовується для того, щоб спілкуватися з $E^2PROM$. Інший канал можна вільно використовувати на платі розширення. До двохпровідної шини $I^2C$ легко підключити багато периферійних пристроїв, аби тільки не конфліктували адреси. Адреса $E^2PROM$ на 2 кілобіти – 0xA0. Інтерфейс $I^2C$ з CAT1025 можна відключити від LPC2148, якщо видалити перемички на J25.

Зауважимо, що резистори підтягування (які завжди потрібні на шинах $I^2C$) впаяні в плату. Номінали цих резисторів складають по 3000 Ом. Якщо використовується другий канал $I^2C$, також не забудьте підключити резистори підтягування до відповідних сигналів. Зауважимо, що це має бути зроблено, навіть якщо $I^2C$ не включений/дозволений. У виводів МК P0.11 та P0.14 відкриті стоки, тому, коли їх конфігурують в якості виходів, потрібно встановлювати резистори підтягування. Це легко забути, проте вони впаяні в стенд *LPC2148 Education Board*.

Шину SPI LPC2148 можна використати для зв'язку з карточкою пам'яті MMC/SD, працюючи в більш простому режимі MMC. Вивід МК/сигнал P0.11 використовується для вибору пристрою. NXP видав керівництво із застосування, в якому описано, як здійснити інтерфейс нижнього рівня (читання та запис сектора на карточці пам'яті). Деталі див. AN10406: Доступ до карти SD/MMC з використанням SPI у LPC2000. Схемотехніку інтерфейсу MMC/SD можна побачити на рисунку 1.9.

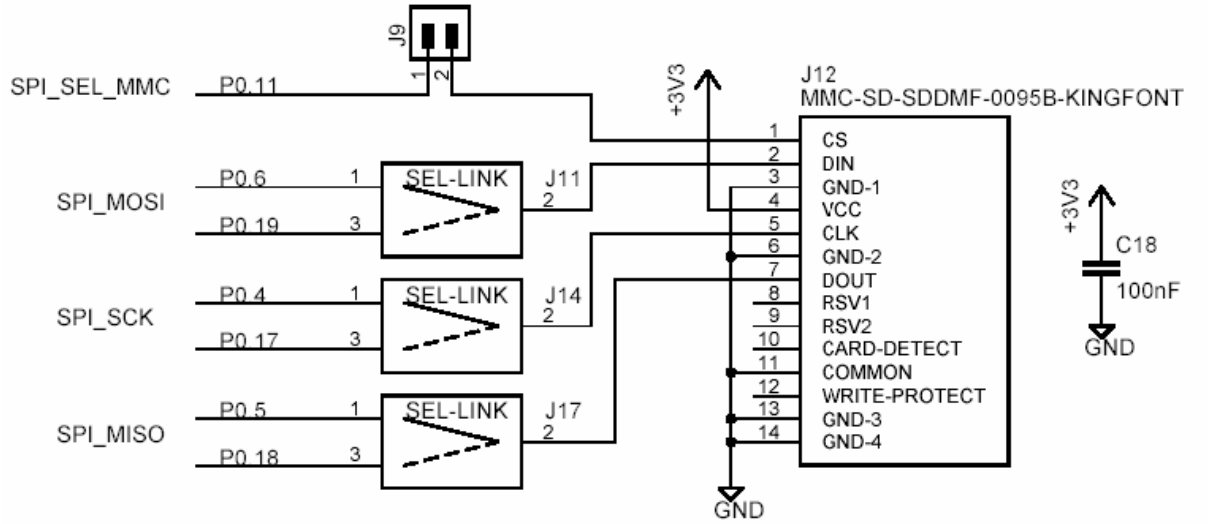



Рисунок 1.9 – Схема інтерфейсу MMC/SD

Інтерфейс можна відключити від LPC2148, знявши перемички J9, J11, J14 та J17. Також зауважимо, що нижній джампер на J24 (SPI_MISO до P0.5 – інтерфейс SPI для світлодіодної матриці 8x8, Рисунок 1.10) не повинен бути встановленим, оскільки цей вивід не має третього стану, а тому буде конфлікт із сигналом MISO (DOUT на інтерфейсі MMC/SD). Звичайно це не проблема, оскільки в установці цього джампера немає потреби.

Якщо до інтерфейсу карточки пам'яті MMC/SD потрібно підключити

канал SPI #1, а не #0, треба змінити перемички J11, J14 и J17. Значення за умовчанням – канал SPI #0.

Зауважимо, що ключі «пошуку карти» та «захисту від запису» в цій розробці взагалі не підключені.

Схему керування світлодіодною матрицею 8x8 проілюстровано на рисунку 1.10. Два регістри зсуву 74HC595 (послідовне введення, паралельне виведення) використовуються для керування вісьмома стовпчиками та рядками, відповідно. Щоб оновити ці два регістри зсуву, треба зробити 16 зсувів (два байти). Перший байт надає керування рядками, а другий байт – стовпчиками. Нульові біти у першому байті засвітять світлодіоди відповідного рядка, а нульові біти у другому байті – світлодіоди відповідного стовпчика. Щоб відобразити на дисплеї довільні інформаційні моделі, треба реалізувати механізм часового мультиплексування, коли кожен стовпчик (або рядок) оновлюється по одному циклічно. Щоб не бачити мерехтіння світлодіодів, частота оновлення має бути щонайменше 100 Гц.

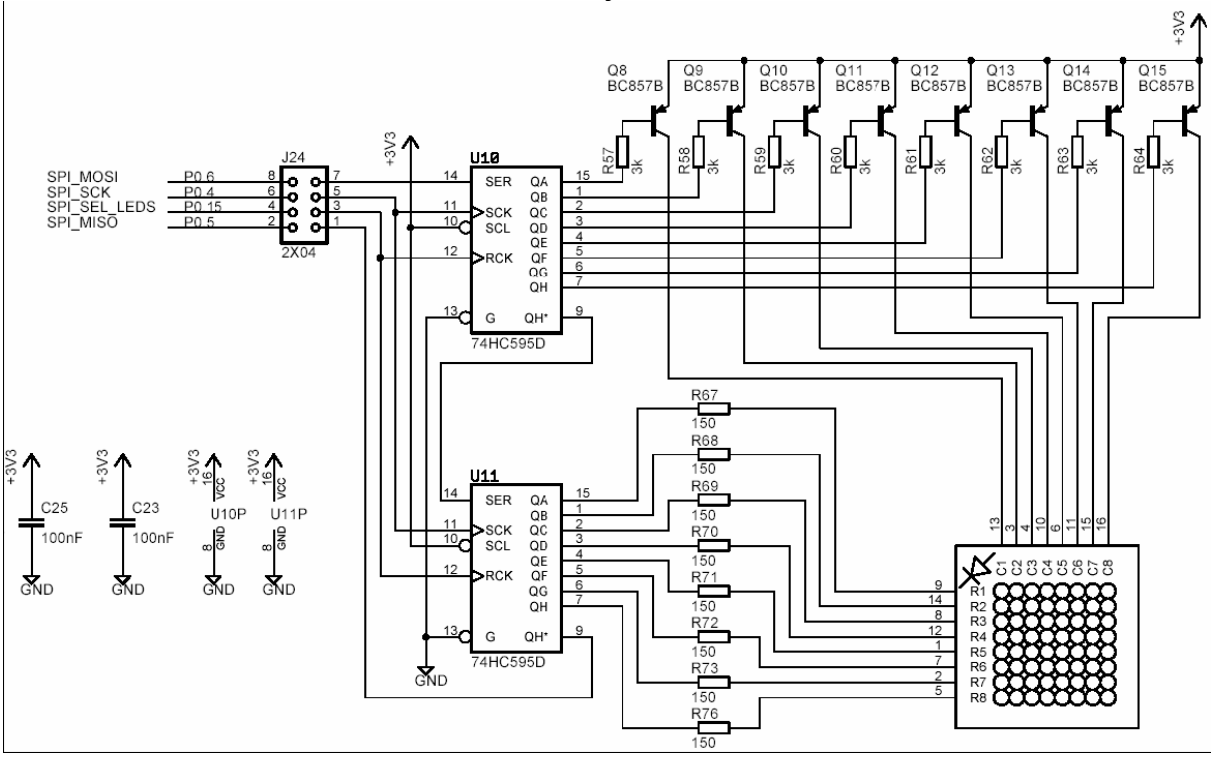



Рисунок 1.10 – Схема керування світлодіодною матрицею 8x8

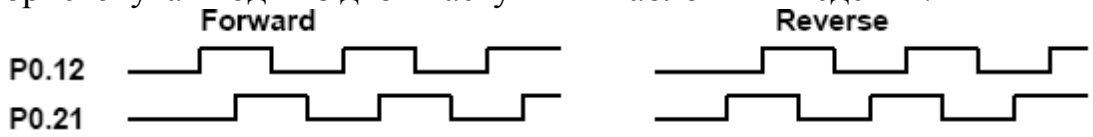
Зауважимо, що для забезпечення максимальної яскравості індикатора більш доцільним є використання режиму одночасного висвічування стовпчика. З цією метою спочатку треба завантажити байт графіки стовпчика з „0" у тих розрядах, де потрібне підсвічування. Далі завантажується байт з одним (!) нулем для включення одного з транзисторів Q8…Q15, які виконують функцію підсилювача струму. Після затримки тривалістю в 1/8 періоду кадру (приблизно 10/8 = 1,25 мс) аналогічну процедуру потрібно повторити для усіх наступних стовпчиків.

У МК LPC2148 є два резидентних канали послідовної передачі даних

SPI. До регістрів зсуву U10, U11 звертаються через шину SPI, канал #0, а сигнал P0.15 використовується в якості виводу вибору пристрою. Щоб зсунути дані у регістри зсуву, можна використати режим 0 SPI. Канал SPI #0 також підтримує інтерфейс MMC/SD.

Схему матриці світлодіодів 8x8 можна відключити від каналу SPI #0, якщо видалити перемички на J24. Зазвичай нижня перемичка на J24 (SPI_MISO на P0.5) не повинна бути вставленою. Річ в тім, що виводи U11 не мають третього стану, коли регістри зсуву заборонено сигналом вибору пристрою P0.15. За наявності нижньої перемички на J24 можливий конфлікт виводу QH мікросхеми U11 з виводом даних (DOUT) від інтерфейсу MMC/SD. Єдина причина вставити цю перемичку могла б полягати в тому, аби прочитати останні два байти, зсунуті у регістри зсуву (проте це рідко, коли потрібно).

Стенд містить маленький двофазний кроковий двигун з пропелером, який полегшує наочність руху. Для керування цим двигуном використовуються сигнали P0.12 та P0.21. Двофазне керування кроковим двигуном означає, що електричний струм має текти через обмотки в одному (з двох можливих) напрямів. Два вихідних буфери (драйвери типу 4428) забезпечують коректність керування обмотками двигуна. Для обертання вісі треба використовувати один з двох наступних шаблонів виведення.




На рисунку 1.11 показано фрагмент стенду з компонентами схеми керування кроковим двигуном.

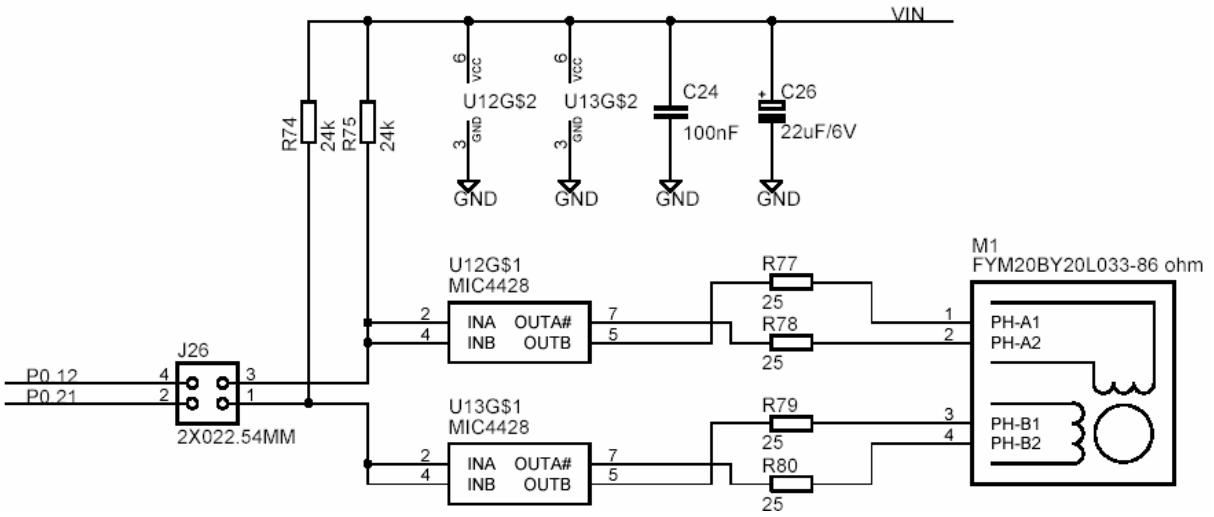



Рисунок 1.11 – Схема керування кроковим двигуном

МК LPC2148 має 14 аналогових входів, які підключені до двох різних 10-бітових аналого-цифрових перетворювачів (АЦП, ADC). Час перетворення не перевищує 2,44 мкс, а в якості опорних використовуються виводи МК VREF та VSSA. На рисунку 1.12 показано схему використання аналогових входів. Як видно з рисунку, для генерації змінюваної напруги

на аналогових входах #1 (P0.28) та #2 (P0.29) використовуються два вимірювальних потенціометри. Зауважимо, що аналогові входи нумеруються, починаючи з #0, проте цей вхід в даному стенді не використовуються.

Якщо аналогові входи не задіяні, сигнали P0.28 та P0.29 можна використовувати в якості універсальних виводів МК. В цьому випадку аналогові напруги легко відключити, якщо видалити дві перемички на J28.

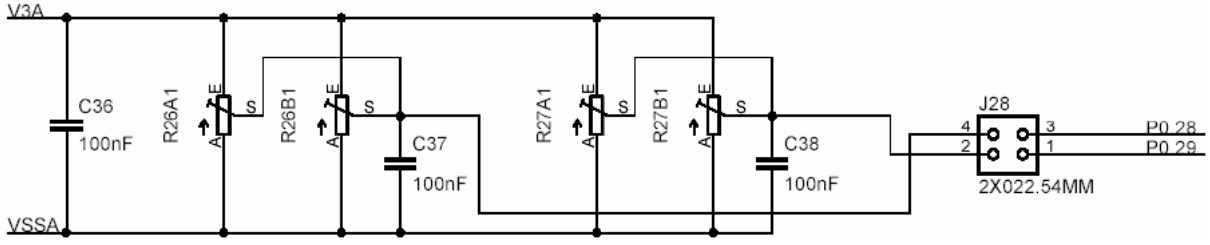



Рисунок 1.12 – Схема використання аналогових входів

МК LPC2148 має один аналоговий вихід, доступний як альтернативна функція на виводі МК P0.25. На рисунку 1.13 показано схему використання аналогового виходу. Через перемичку J30 можна підключити аналоговий вихід до аналогового входу #3 (P0.30). Для того, щоб полегшити вимірювання аналогової напруги, на платі також присутні спеціальні гнізда (PAD6 та PAD9).

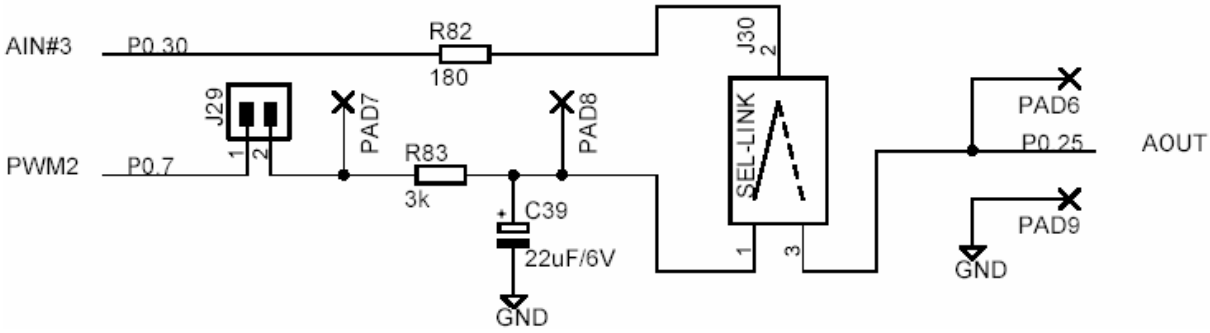



Рисунок 1.13 – Схема використання аналогового виходу

Крім того можна генерувати аналогову вихідну напругу, фільтруючи сигнал ШІМ (PWM). В цьому випадку за допомогою фільтра нижніх частот (ФНЧ) фільтрується альтернативний для виводу МК P0.7 сигнал PWM2. Результат можна виміряти за допомогою аналогового входу #3. Велику перевагу надає використання осцилографу для вимірювання сигналу PWM до та після фільтрації за допомогою ФНЧ. Частота зрізу $1/(2\pi RC)$ приблизно дорівнює 2 Гц, а це означає, що будь-який ШІМ-сигнал з розумно високою частотою буде адекватно відфільтрованим ФНЧ. Також можна виміряти, чи відповідають залишкові пульсації тому рівню, що можна чекати від простого фільтра нижніх частот першого порядку.

Зручним пристроєм на стенді *LPC2148 Education Board* є символьний рідинно-кристалічний дисплей (РКД або LCD) типу LMB162A з організацією 2 рядки по 16 символів у кожному. Інтерфейс РКД моделює шину пам'яті. Для утворення 8-бітового інтерфейсу потрібні 11 виводів: 8 інфо-

рмаційних розрядів (D0-D7), 1 біт адреси (RS), 1 біт для сигналу читання/запису (R/W) та один сигнал керування (E). Також передбачена можливість керування світлодіодом підсвічування за допомогою сигналу P0.30. Контролер РКД – схожий на стандартний KS0070B або еквівалентний, інтерфейс якого є дуже добре відомим у світі символьних РКД.

На рисунку 1.14 показано схему підключення РК-дисплея та виводи МК, які використовуються для інтерфейсу. Усі виводи за необхідності можна відключити від інтерфейсу перемичками J31 – J42. РКД живиться від джерела живлення +3,3В. Настроювання контрасту дисплея (за допомогою R86) в реальному стенді не передбачене, оскільки використаний РКД не потребує корегування.

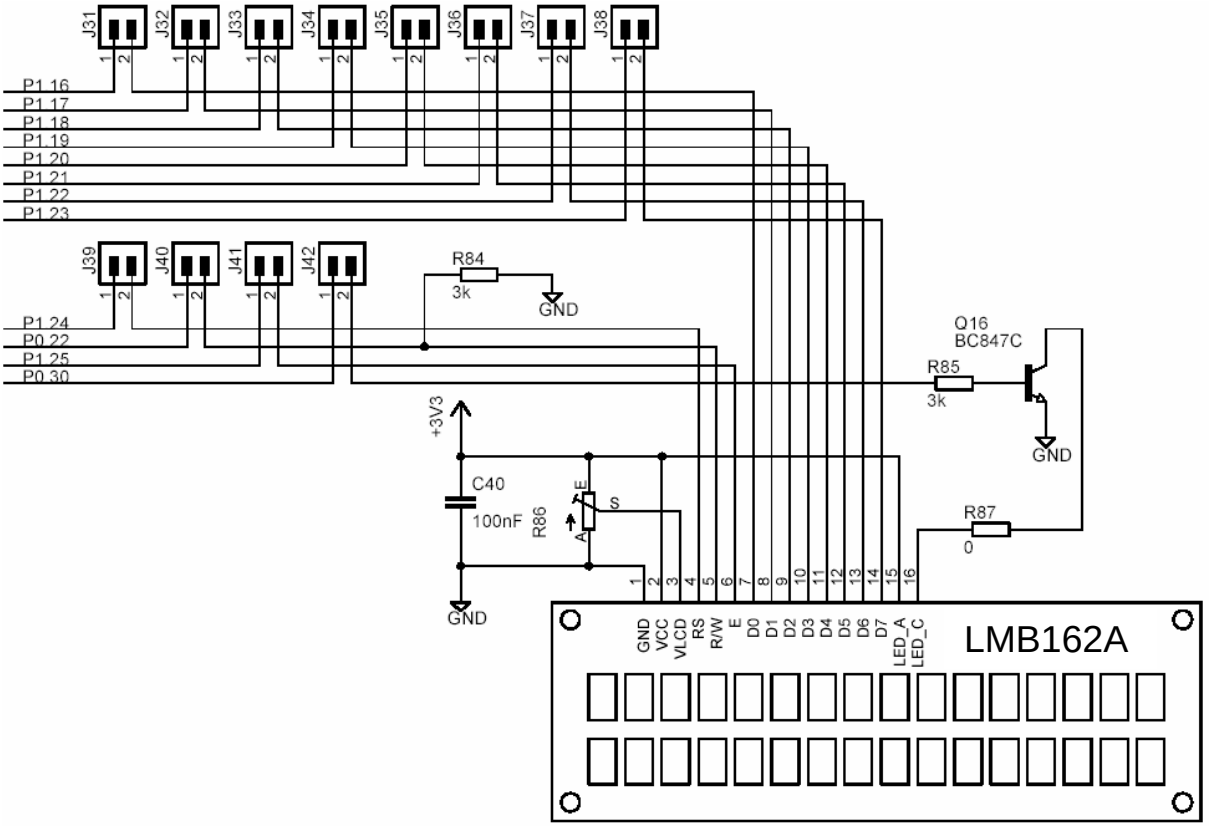



Рисунок 1.14 – Схема підключення РК-дисплея 16x2

Щоб скоротити кількість виводів МК, задіяних для інтерфейсу, можна зробити наступне.

• Не використовувати сигнал R/W, зекономивши один вивід (P0.22). Звичайно є потреба тільки записувати в контролер дисплея. В такому випадку сигнал R/W може бути постійно низьким (R84), а перемичка J40 видалена.

• Використовувати 4-бітовий інтерфейс замість 8-бітового. Контролер РКД може працювати і з 4-бітовою шиною даних, використовуючи тільки D4 – D7. В такому разі 4 виводи МК (P1.16 – P1.19) будуть звільнені для іншого застосування. Разом з попереднім «вдосконаленням» можна

вивільнити п'ять виводів.

Одним з важливих периферійних пристроїв для LPC2148 у стенді є вузол інтерфейсу USB 2.0 (рисунок 1.15). Фільтри низьких частот додані для завадостійкості. Інтерфейс USB підтримує «м'яке підключення» (*Soft Connect*) та слідкування за напругою [1]. М'яке підключення керується виводом МК P0.31 та активується, коли між сигналом D+ та +3,3 В встановлений резистор 1,5 кОм. При цьому також світиться зелений світлодіод LED1. Режим слідкування за напругою реалізується за допомогою виводу МК P0.23, який пов'язаний з напругою живлення інтерфейсу USB.

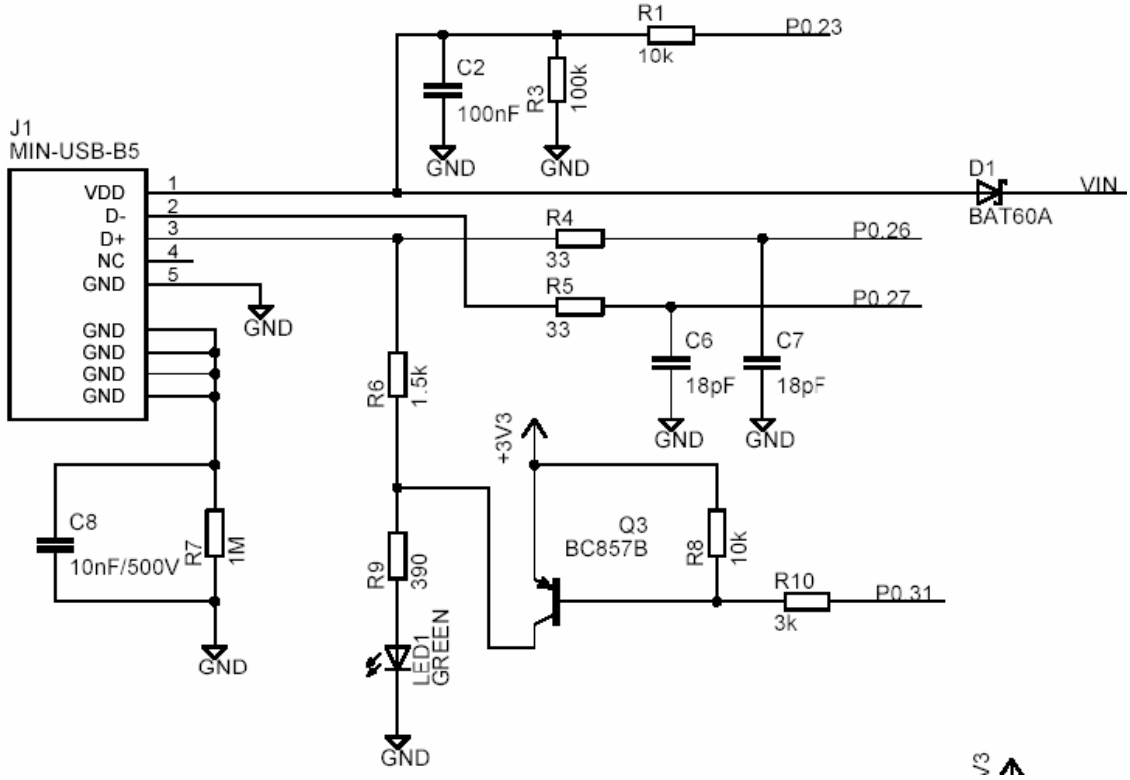



Рисунок 1.15– *Схема вузла інтерфейсу* USB 2.0

На стенді LPC2148 Education Board розміщено багато джамперів, що дозволяє підключати/відключати окремі частини зовнішніх вузлів. На рисунку 1.16 показані всі перемички та наведено пояснення, до якої частини стенду вони належать.

Відмітимо, що деякі перемички взаємно виключають одна одну і не мають бути вставлені одночасно.

• Сигнал MISO (P0.5) на каналі SPI #0. Одночасно не повинно бути перемички на J24, оскільки цей сигнал також використовується інтерфейсом MMC/SD. Див. пояснення вище.

• P0.7 – це сигнал, що використовується для зумера та червоного світлодіода в RGB-LED.

• Більшість сигналів для індивідуальних світлодіодів (P0.8 – P0.15) використовується також в інших частинах стенду. Це не пошкоджує компоненти, якщо світлодіоди підключені, а в деяких випадках може бути навіть корисним мати світлодіодну індикацію окремих сигналів.

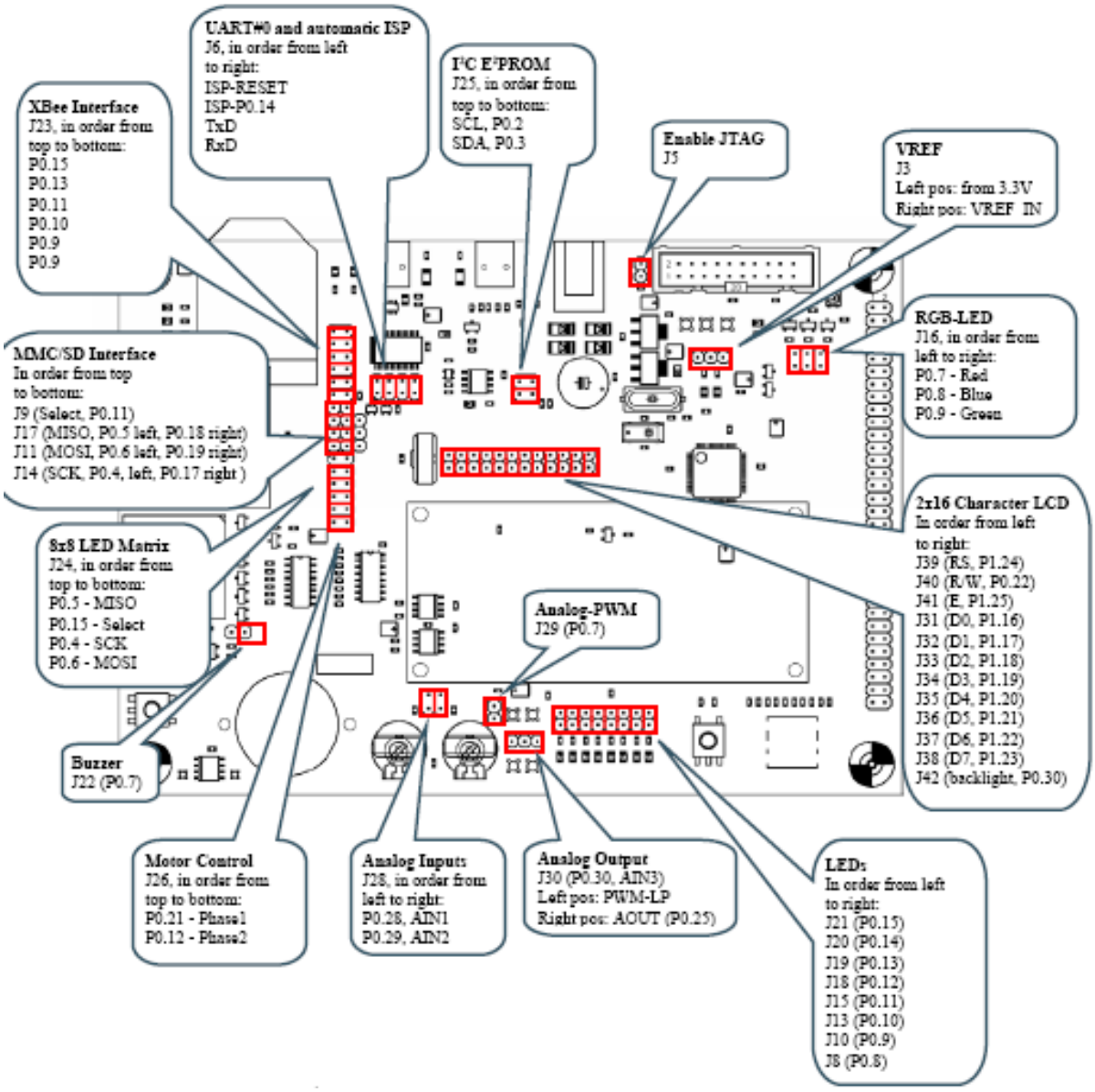



Рисунок 1.16– Джампери на стенді *LPC2148 Education Board*

## 1.2 Вступ до програмування ARM7-МК

### 1.2.1 Завантаження програми через ISP

Утиліта *FlashMagic* від *ES Academy* – це рекомендована NXP програма для завантаження застосувань користувача до внутрішньої пам'яті МК у режимі внутрішньосхемного програмування (*In-System Programming*, ISP). Якщо на комп'ютері встановлені драйвери мікросхеми-мосту USB-RS232, під час підключення стенду автоматично визначається певний послідовний порт (наприклад, UART#0), який можна використати для завантаження коду програми у внутрішню FLASH-пам'ять МК LPC2148.

У диспетчері пристроїв треба подивитися, до якого саме послідовного порту підключився наш пристрій (наприклад, `USB Serial (Port COM4)`).

Режим ISP дозволяється в тому випадку, коли під час скидання МК утримується низький рівень на виводі P0.14. Щоб автоматизувати включення ISP, в стенді *LPC2148 Education Board* передбачено керування і сигналом скидання, і P0.14 через міст *USB-to-serial* (мікросхема *FT232RL* фірми *FTDI*). Скидання керується від сигналу *DTR* послідовного порту, а P0.14 – від сигналу *RTS*. Останні версії утиліти *FlashMagic* можна завантажити з:

http://www.flashmagictool.com/

На рисунку 1.16 та Figure 29 в описі стенду [2] показані установки перемичок для автоматичної реалізації режиму ISP. Частота задаючого генератора у вікні настроювання *FlashMagic* має відповідати частоті кварцового резонатора на стенді, тобто 12 МГц.

### 1.2.2 Програма тестування *LPC2148 Education Board*

Стенд *LPC2148 Education Board* поставляється з тестовою програмою, яка дозволяє перевірити правильність роботи пристрою.

•Завантажте тестову програму, використовуючи шістнадцятковий файл:

`testprogram_lpc2148_edu_v2_1(прошивка від виробника).hex`

Цей файл можна знайти на робочому комп'ютері в лабораторії, або за посиланням [10] (Інтранет).

- Вставте всі перемички у позиції за умовчанням
- Натисніть кнопку скидання
- Подивіться на «одиницю, що біжить» на індивідуальних світлодіодах
- Також подивіться друковану інформацію на терміналі (див. нижче)
- Після цього починається цикл тестування:
  - На РКД друкується повідомлення та мерехтить лампа підсвічування
  - На світлодіодній матриці 8x8 прокручується повідомлення
  - Плавно мерехтять RGB-світлодіоди із зміною кольору
  - Кроковий двигун прокручується вперед та назад
  - По USB за допомогою джойстика емулюється PC-мишка. Для реалізації цієї функції потрібно підключитися до комп'ютера через роз'їм інтерфейсу USB 2.0 (mini-B) для зовнішніх приладів *LPC2148 Education Board* (поз. 7 на рисунку 1.1)

Термінальна програма на персональному комп'ютері взаємодіє з навчальним стендом *LPC2148 Education Board* за допомогою роз'єднувача USB mini-B (USB-послідовний UART). У вікні терміналу можна побачити (рисунок 1.17), як тестова програма виведе випробувальну інформацію щодо перевірки інтерфейсу $I^2C$ та $E^2PROM$, тест MMC/SD, а також тест резонатора годинника реального часу. Ручне повертання потенціометрів платі (поз. 17 на рисунку 1.1) призводить до відповідної зміни вхідних рівнів АЦП (P0.28 = AIN1, P0.29 = AIN2). Крім того, можна перевірити канал

UART/USB, друкуючи символи в термінальній програмі. Параметри налаштування термінальної програми: 38,4 кбіт/с, 8 інформаційних розрядів, без бітів парності та один стоповий біт (тобто 8N1). Після запуску терміналу, можливо, доведеться скинути *LPC2148 Education Board*.

Зауважимо, що можна використовувати, як зовнішні термінальні програми (*Terminal*, *PCcomm Terminal Emulator*), так і ту, яку можна запустити з вкладки *Tools* утиліти *Flash Magic*.

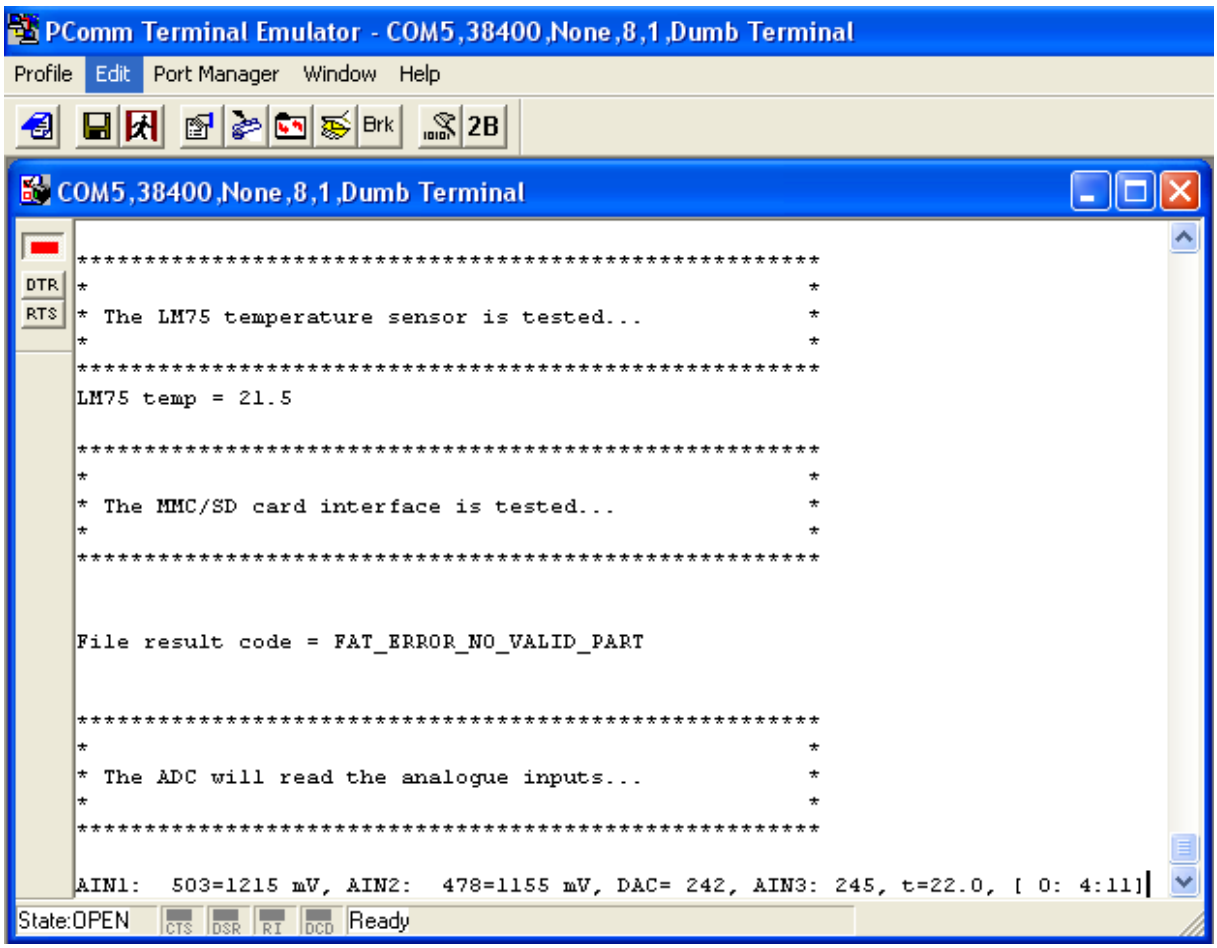



Рисунок 1.17 – Вікно термінальної програми *PCcomm Terminal Emulator* під час виконання тесту *LPC2148 Education Board*

## 1.3 Контрольні питання

1. Навести призначення та склад навчального стенду *LPC2148 Education Board*.
2. Дати стислу характеристику застосованому МК (назва, розрядність, резидентна пам'ять, периферійні пристрої, інтерфейси, які підтримуються, тактова частота.
3. Указати на платі стенду розміщення окремих вузлів.
4. Навести схеми підключення зовнішніх пристроїв до МК та схарактеризувати особливості роботи з цими пристроями.

5. Дати характеристику програмним та апаратним ресурсам, які використовуються для завантаження програми користувача до навчального стенду *LPC2148 Education Board*.

## *1.4 Хід роботи*

### 1.4.1 Підготовчі стадії

Поза навчальною лабораторією треба зробити наступне:

1. Користуючись розділом 1.1, а також рекомендованою літературою, ознайомитися із навчальним стендом *LPC2148 Education Board* та особливостями його складових частин.

2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвища студентів бригади, мету лабораторних досліджень та дати письмові відповіді на контрольні питання (п. 1.3).

### 1.4.2 Етап лабораторних досліджень

1. Перевірити працездатність апаратних та програмних засобів відлагодження програмного забезпечення МК, виконавши дії, описані в п. 1.2.2.

2. Зробити скрін-шоти роботи тестової програми з термінальною програмою.

## *1.5 Вимоги до звіту по роботі*

Звіт про виконання роботи повинен містити:

1. Титульний аркуш встановленого в університеті зразку.

2. Письмові відповіді на контрольні питання.

3. Скрін-шот роботи тестової програми з термінальною програмою.

4. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Які саме конкретно апаратні та програмні засоби (перерахувати по пунктах) дозволяє досліджувати навчальний стенд *LPC2148 Education Board*?

# 2 Лабораторна робота №2. Дослідження можливостей інтегрованого середовища розробки *Keil µVision*

***Мета роботи***: виявити можливості інтерфейсу користувача, навчитися складати новий проект та редагувати його налаштування, вносити зміни до початкового коду, запускати просту програму на тестовій платі.

## *2.1 Короткі теоретичні відомості*

*Microcontroller Development Kit Keil® MDK* – одне з найбільш повних рішень з розробки програмного забезпечення для мікроконтролерів на основі ядра *ARM®*, що включає всі компоненти, які потрібні для створення, збирання та налагодження вбудованих додатків [11].

Середовище розробки *Keil µVision* містить менеджер проектів, редактор вхідних текстів, компілятор, засоби відлагодження та утиліти для повної симуляції МК.

*Keil® MDK* складається (рисунок 2.1) з інструментальної частини (*MDK-Tools*) та специфічної частини (*Software Packs*).

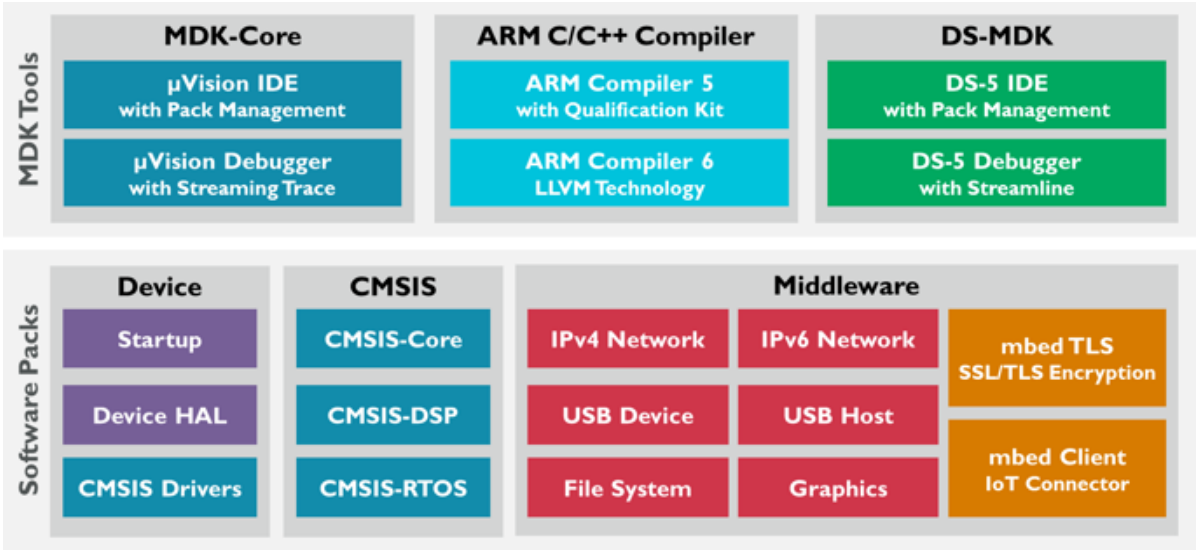



Рисунок 2.1 – Компоненти *Microcontroller Development Kit Keil® MDK*

*MDK-Core* базується на IDE *µVision* з підтримкою як найновіших, так і традиційних МК з архітектурою *ARM*. *DS-MDK* містить IDE/ Відлагоджувальник *DS-5* на основі *Eclipse* і підтримує 32-розрядні процесори *Cortex-A* або гібридні системи з 32-бітними *Cortex-A* та *Cortex-M* (доступно з липня 2016 р.). МДК включає в себе два компілятори ARM C/C++ з асемблером, компонувальником і бібліотеками, спеціально оптимізованими під час виконання програми між розміром коду та продуктивністю.

*Software Packs* (програмні пакети) додають компоненти підтримки пристроїв і програмного забезпечення, які можна використовувати в якості будівельних блоків для розроблюваної програми. Програмні пакети можуть бути додані в будь-який час до *МДК-Core* або *DS-MDK*, забезпечуючи підтримку нових пристроїв і оновлення проміжного програмного забез-

печення незалежно від інструментального ПЗ. Вони містять підтримку пристроїв, бібліотеки, проміжне програмне забезпечення, підтримку плат відлагодження ПЗ, шаблони коду, а також приклади проектів.

## 2.2 Встановлення Keil® MDK

Для того, щоб виконати на стенді *LPC2148 Education Board* свій перший проект для *LPC2148 ARM7TDMI-S™*, необхідно зробити декілька кроків, які дозволять Вам підготувати робоче місце та врешті-решт завантажити резидентну флеш-пам'ять цільового процесора шістнадцятковим файлом для виконання.

1. По-перше, треба завантажити та встановити середовище відлагодження програмного забезпечення *MDK-Lite* [11].

Процедура розпочинається з реєстрації користувача, для чого треба заповнити відповідні поля. На час написання цього навчального видання доступна версія MDK524A.EXE (796,513K), July 4, 2017.

Корисним буде дослідити директорію, де встановлене IDE (наприклад, – `D:\Keil_v5\`). У директорії D:\Keil_v5\UV4\ Ви знайдете власне *IDE µVision* (файл `UV4.exe`), а також іншу корисну програму – `PackInstaller.exe`, яка призначена для установки, оновлення і видалення програмних пакетів, і може бути запущена, як зсередини *IDE µVision*, так і автономно, поза *µVision*.

2. Оскільки даний цикл лабораторних робіт базується на МК сімейства ARM7, також необхідно завантажити пакет підтримки цих контролерів на сторінці розробника *µVision* [12]. Для цього необхідно натиснути кнопку `Download Legacy support for ARM7, ARM9 & Cortex-R` (для останньої версії основної програми), або обрати потрібне посилання (для більш старих версій). Після встановлення *Legacy SupportMDK Version 5* здатна підтримувати проекти *MDK Version 4*. Це додасть підтримку приладів ARM7, ARM9 та ARM Cortex-R4. Найбільше нас наразі цікавить МК LPC2148 від фірми NXP, підтримку якого, зокрема, розміщено в [13].

Результат інсталяції потрібних програм буде приблизно такий, як показано в таблиці 2.1.

Таблиця 2.1 – Потрібні завантаження

| **Назва пакету** | **Файл завантаження** | | **Обсяг пакету після установки, ГБ** |
|---|---|---|---|
| | **Назва** | **Обсяг, МБ** | |
| 1. MDK Tools | MDK524a.exe | 816 | 2,936 |
| 2. MDK ARM7 & ARM9 legacy support | MDK79524.exe | 127 | 3,351 |

## 2.3 Хід роботи

### 2.3.1 Підготовчі стадії

1. Запустити *Keil µVision*.

2. Ознайомитися з елементами керування. Середовище розробки *Keil µVision* являє собою типове застосування *Windows*. Його вигляд з основними елементами керування зображений на рисунку 2.2.

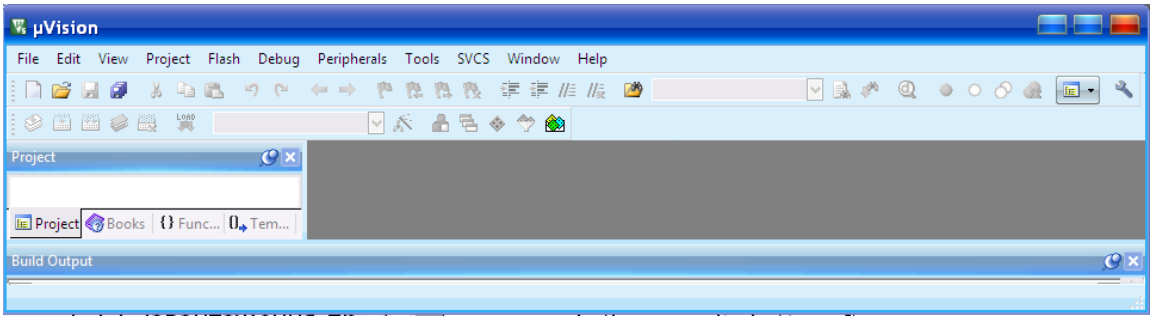



Рисунок 2.2 – Інтерфейс користувача середовища *Keil µVision*.

3. Створити новий проект (рисунок 2.3).

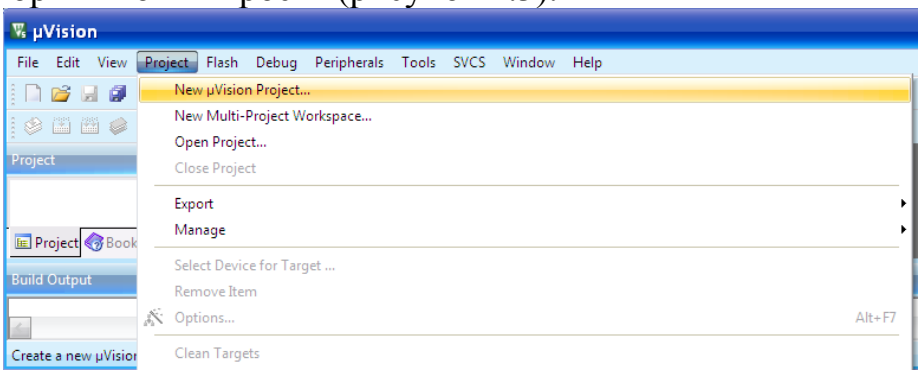



Рисунок 2.3 – Створення нового проекту

4. Зберегти проект на диску. Вибрати потрібний тип МК (рисунок 2.4). `Vendor: NXP (Philips)`, `Device: LPC2148`. Ознайомитися із списком периферійних пристроїв. Погодитися на створення *Startup*-файлу та автоматичне додання його до проекту.

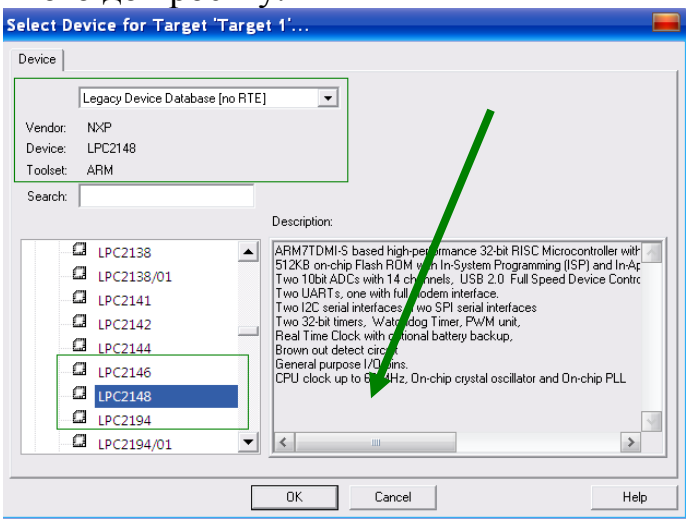



Рисунок 2.4 – Вибір типу МК

5. Відкрити вкладку проглядання вмісту проекту (рисунок 2.5) та вибрати `Startup.s`. Уважно прочитати текстову інформацію на початку та ознайомитися із вмістом цього файлу.

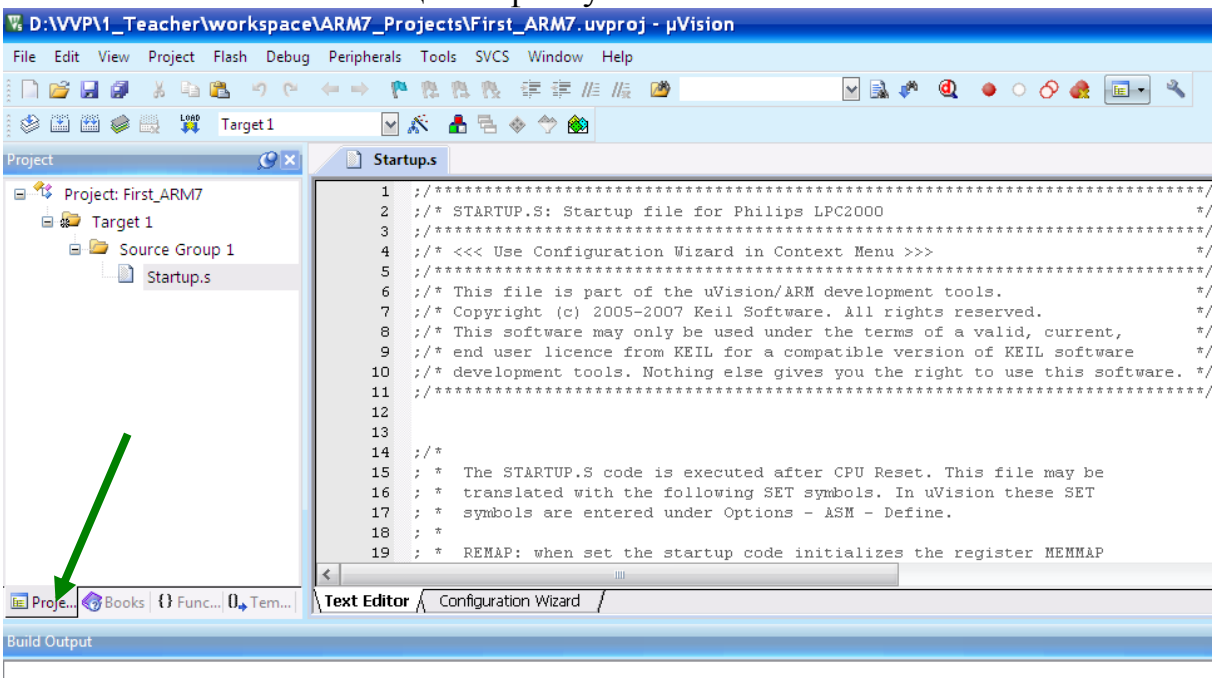



Рисунок 2.5 – Проглядання вмісту проекту

6. Вибрати вкладку `Edit -> Advanced -> Toggle Text Editor/Configuration Wizard (Shift+F8)` та ознайомитися із списком налаштувань, які можна редагувати у середовищі *Keil* (рисунок 2.6).

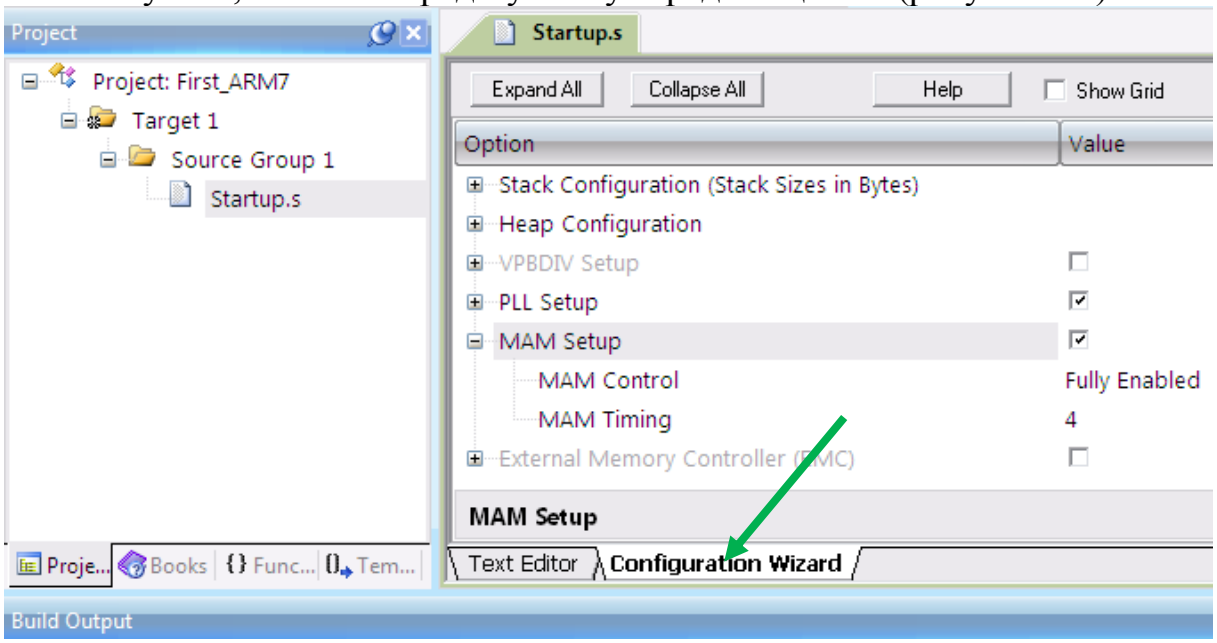



Рисунок 2.6 – Майстер налаштувань стартової програми проекту

7. Вказавши мишкою на проект (`Project/target`), розпочати налаштування: `Project -> Options for target` '…' (або Alt+F7);

8. На вкладці "`Debug`" (рисунок 2.7) вибрати "`Use Simulator`" та встановити "`Run to main()`".

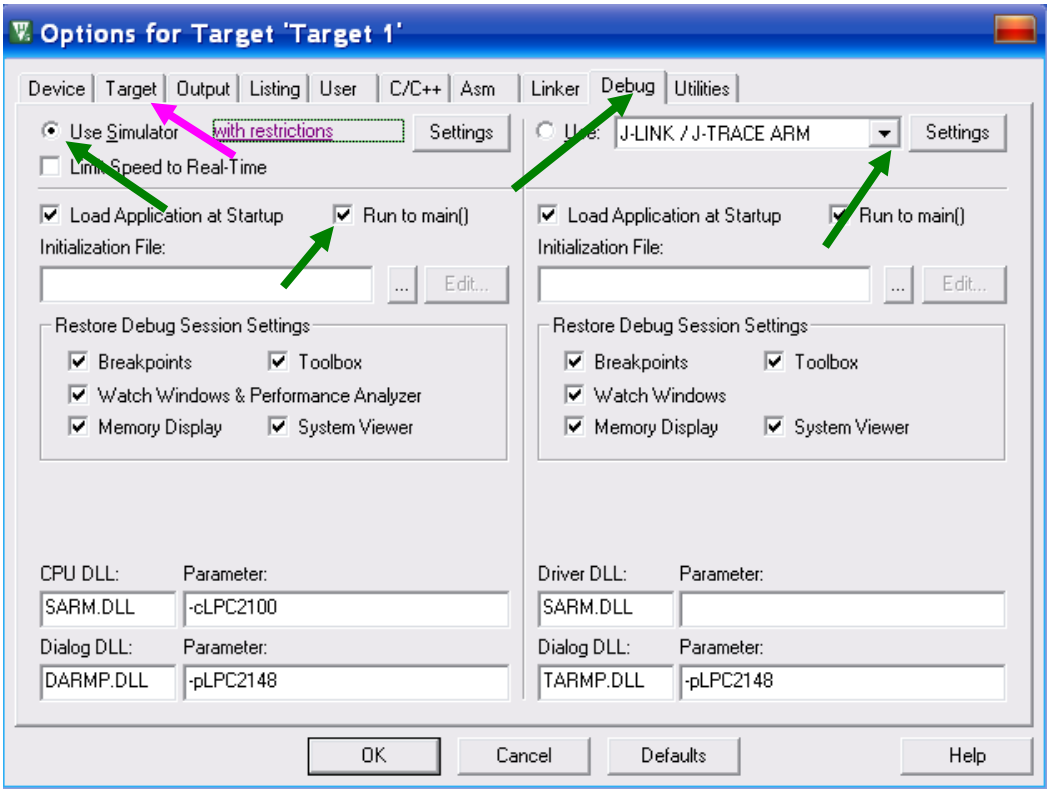



Рисунок 2.7 – Налаштування проекту

9. Також потрібно проконтролювати частоту кварцового резонатора на вкладці `Target`, яка має відповідати `Xtal (MHz) = 12.0`. На вкладці `Linker` необхідно активізувати опцію `Use Memory Layout from Target Dialog`.

10. У випадку завантаження програми користувача до резидентної пам'яті МК за допомогою утиліти *Flash Magic* на вкладці `Output` потрібно відмітити `Create HEX File`.

11. Якщо наявний апаратний JTAG-емулятор, на вкладці "`Debug`" треба обрати потрібну апаратну платформу, наприклад, `Use: J-LINK / J-Trace ARM`, а далі провести настроювання після натискання `Settings`.

12. Треба дослідити вміст вкладки `Books` у робочому просторі *Keil® MDK* та уважно прочитайте документ `Getting Started`. З'ясуйте, для чого призначені та з чого складаються компоненти `CMSIS`.

13. З'ясуйте, для чого призначені та з чого складаються компоненти `Middleware`.

14. Треба дослідити документ `ARM Development Tools`. З'ясуйте, чим відрізняються редакції *MDK* (*Microcontroller Development Kit Editions*).

15. З'ясуйте, чим відрізняється розміщені у розділі `Device Data Books` вкладки `Books` *Keil® MDK* документи `User Manual` від `Data Sheet`.

### 2.3.2 Етап розробки програмного забезпечення

1. Створити новий файл (`File -> New ...`) та зберегти його в поточній директорії під назвою `main.c`. Додати цей файл до проекту (подвійне на-

тискання на "Source Group 1", рисунок 2.8. Написати текст програми та зберегти файл (лістинг 2.1).

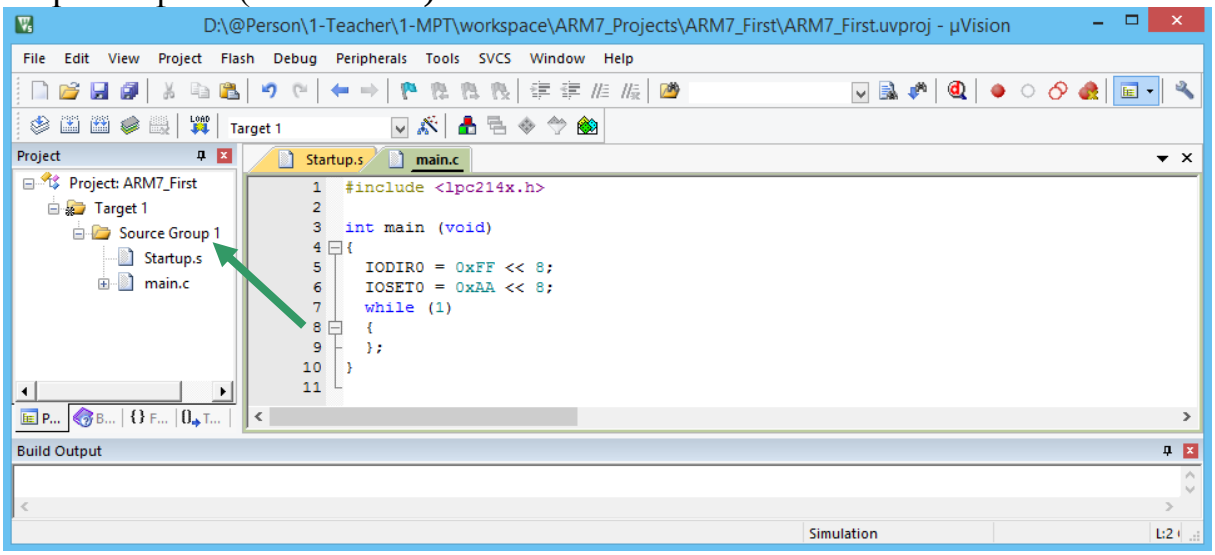



Рисунок 2.8– Додавання програми до проекту

main.c Лістинг 2.1.

```
01  #include <lpc214x.h>
02
03  int main (void)
04  {
05          IODIR0 = 0xFF << 8;
06          IOSET0 = 0xAA << 8;
07          while (1)
08          {
09          };
10  }
11
```

2. Користуючись технічною документацією на досліджуваний МК, наприклад, розділ 8 *General Purpose Input/Output ports* (*GPIO*) керівництва користувача [5], а також знаннями, отриманими під час виконання лабораторної роботи №1, з'ясувати, що саме має зробити програма в стенді *LPC2148 Education Board*.

3. Скомпілювати програму. Для цього вибрати `Project -> Build Target`, або нажати F7. У випадку наявності помилок їхній список буде відображений у вікні "`Build Output`".

4. Запустіть програму на відлагодження `Debug -> Start/Stop Debug Sesion` (або Ctrl+F5). Засвоїти використання стандартних функцій відлагоджувальника (`Step In`, `Step Out`, `Step Over`, `Run`, `Stop` – рисунок 2.9).

5. Використовуючи утиліту *Flash Magic*, завантажте програму до резидентної флеш-памяті МК, запустіть на виконання та подивіться на результати її роботи на навчальному стенді.

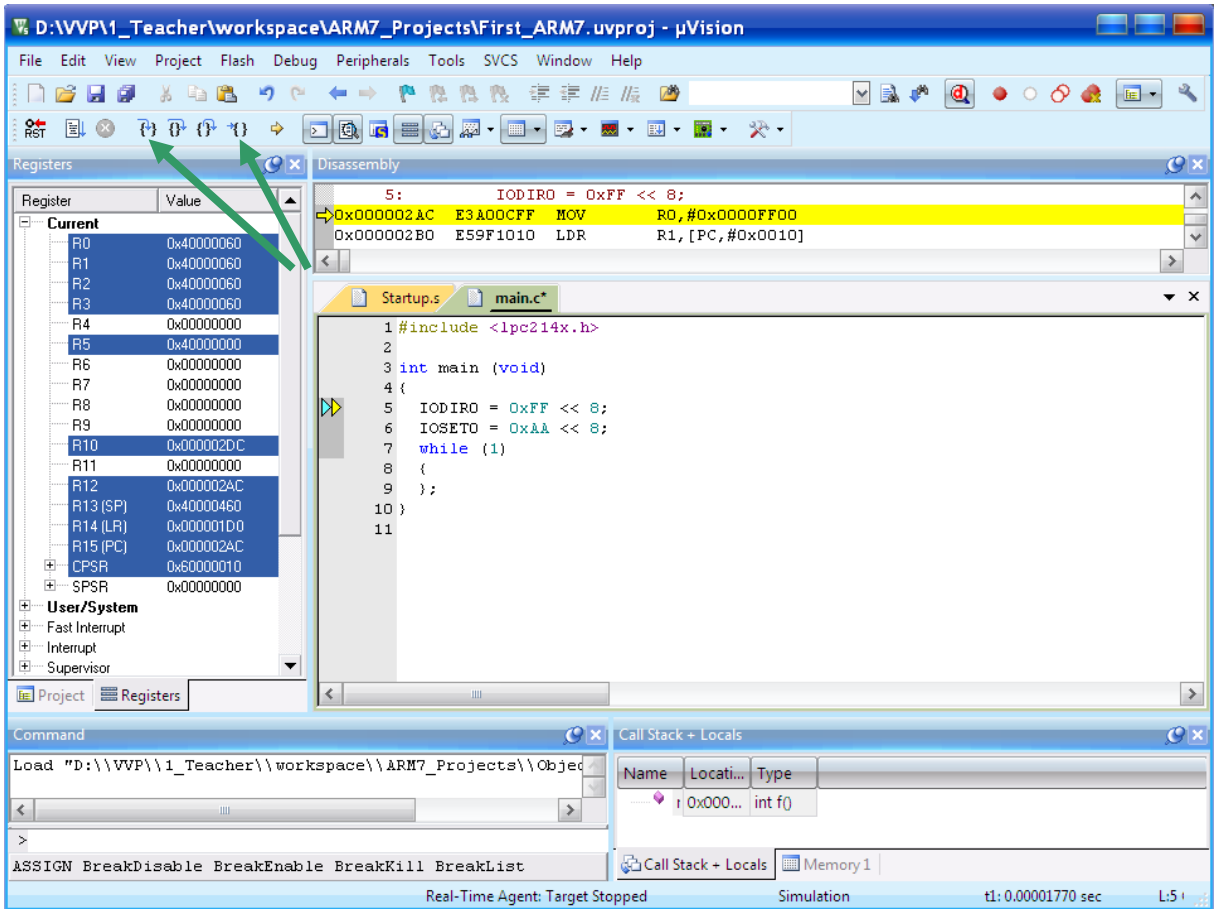



Рисунок 2.9 – Відлагодження програми

6. Розробіть та виконайте перевірку програми відповідно до варіанту завдання (підрозділ 2.6).

## 2.4 Контрольні питання

1. Пояснити призначення та охарактеризувати можливості IDE *Keil µVision*.
2. Пояснити призначення пунктів меню `File` *Keil µVision*.
3. Пояснити призначення пунктів меню `Edit` *Keil µVision*.
4. Пояснити призначення пунктів меню `View` *Keil µVision*.
5. Пояснити призначення пунктів меню `Project` *Keil µVision*.
6. Пояснити призначення пунктів меню `Debug` *Keil µVision*.
7. Пояснити призначення пунктів меню `Flash` *Keil µVision*.
8. Пояснити призначення пунктів меню `Peripherals` *Keil µVision*.
9. Пояснити призначення пунктів меню `Tools` *Keil µVision*.
10. Пояснити призначення окремих рядків тестової програми.
11. Для чого призначені та з чого складаються компоненти `CMSIS`?
12. Для чого призначені та з чого складаються компоненти `Middleware`?
13. Чим відрізняються редакції *MDK-Lite* від *MDK-Professional*?

14.Чим відрізняється `User Manual` від `Data Sheet`?

## *2.5 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні питання.
3. Схему, коментований текст програми вирішеної задачі та схему підключення індикатора.
5. Висновки, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) дозволяє робити *Keil µVision*?

## *2.6 Орієнтовні варіанти завдань*

Завдання: Організувати засвічування наступних світлодіодів:

Варіант 1

●☼●●●●●●

Варіант 2

●☼●☼●●●●

Варіант 3

●☼●☼●☼●●

Варіант 4

●☼●☼●☼●☼

Варіант 5

☼●☼●☼●☼●

Варіант 6

Задається викладачем.

# 3 Лабораторна робота №3. Дослідження особливостей портів введення/виведення загального призначення

***Мета роботи***: виявлення особливостей портів введення/виведення загального призначення та ознайомлення з методами роботи з швидкими портами (FIO)

## *3.1 Короткі теоретичні відомості*

Порти введення/виведення загального призначення (*General Purpose Input Output – GPIO*) призначені для обміну даними або для керування зовнішніми пристроями. Кожний з портів може бути налаштований як на передачу, так і на прийом інформації. Деякі з портів можна налаштувати в якості входів переривань, коли сигнал із зовнішнього пристрою ініціює певну дію МК, що відмінна від основної програми.

Для ліній введення/виведення загального призначення мікроконтролерів сімейства LPC2000 характерні наступні особливості:

– управління напрямком передачі даних (введення або виведення) для кожної лінії індивідуальне;

– роздільне управління установкою і скиданням стану лінії;

– всі лінії за замовчуванням після скидання налаштовуються як вхідні.

Порти введення/виведення МК *NXP LPC2148* можуть бути підключені до шини периферійних пристроїв VPB (VLSI peripheral bus) або до локальної шини (ARM7 local bus). У другому випадку порти називаються «швидкими» (*Fast General Purpose I/O – FIO*), і частота їх перемикання може бути суттєво підвищена.

Якщо програма користувача не задіє ніяких альтернативних функцій виводів МК LPC2148, доступні 45 ліній введення/виведення загального призначення: 29 ліній введення/виведення порту 0 (всі лінії, крім P0.24, P0.26, P0.27) і 16 ліній введення/виведення порту 1 (P1.16…P1.31).

На стенді порти введення/виведення виконують декілька функцій:

1. Керування світлодіодами.
2. Керування символьним рідинно-кристалічним дисплеєм (РКД) *LMB162A*.
3. Керування роботою крокового двигуна.
4. Керування регістрами, які підключені до світлодіодної матриці.

Розглянемо застосування портів введення/виведення з метою керування лінійкою світлодіодів. Схему підключення світлодіодів у стенді наведено на рисунку 1.2. Щоб засвітити світлодіод, потрібно конфігурувати відповідний порт на вихід і подати на нього **нульовий** рівень сигналу. З схеми видно, що світлодіоди підключені до виводів 8…15 порту 0.

## *3.2 Опис регістрів GPIO/FIO*

Налаштування *GPIO* відбувається під час виведення певної комбінації бітів у регістри, які відповідають за конкретний порт x = 0 або 1.

IOxPIN, FIOxPIN — Регістри, з яких можна отримати інформацію про поточний стан портів вводу/виводу, незалежно від напряму передавання та призначення ніжок. Нульовий біт відповідає порту Px.0, 31-й – Px.31. Використовують тільки для читання

IOxDIR, FIOxDIR — Керуючий регістр портів введення/виведення. Цей регістр указує напрям введення/виведення. Нульовий біт відповідає порту Px.0, 31й – Px.31. Використовують для запису та читання

IOxSET, FIOxSET — Регістр виведення. Контролює стан виводу ніжок спільно з регістром IOCLRx. Запис 1 дає високий рівень на відповідній ніжці. Запис 0 не має ефекту. Нульовий біт відповідає порту Px.0, 31й – Px.31. Використовують для запису та читання

IOxCLR, FIOxCLR — Очищуючий регістр виведення. Цей регістр контролює стан виводу ніжок. Запис 1 дає низький рівень на виході та очищує відповідний біт в регістрі IOSETx. Запис 0 не має ефекту. Нульовий біт відповідає порту Px.0, 31й – Px.31. Використовують тільки для запису

FIOxMASK — Регістр маски для швидкого порту. Дозволяє писати, встановлювати, скидати та читати з порту (за допомогою запису у FIOxPIN, FIOxSET і FIOxCLR та читання з FIOxPIN) тільки тих бітів, для яких у цей регістр завантажений нуль

Регістри IOxPIN та FIOxPIN призначені для відображення поточного стану виводів порту. Біти регістрів відображають будь-які зміни зовнішніх сигналів, проте тільки для виводів МК, сконфігурованих, як лінії введення/виведення загального призначення. При налаштуванні виводів для виконання альтернативних функцій контроль їх стану з використанням цих регістрів не забезпечить достовірного результату. Після скидання МК значення бітів цих регістрів є невизначеним, що відповідає переведенню ліній введення/виведення загального призначення в Z-стан.

Регістри IOxDIR та FIOxDIR призначені для управління напрямком виводів порту, якщо вони конфігуруються для введення/виведення загального призначення. Запис "1" в задані біти регістрів встановлює відповідні виводи, як вихідні (для передачі даних). Запис "0" встановлює відповідні виводи, як вхідні (для прийому даних). Після скидання МК значення бітів цих регістрів дорівнює "0".

Регістри IOxSET та FIOxSET використовуються для установки виводів порту, якщо вони налаштовані як лінії введення/виведення загального призначення і для них заданий режим виходів. Запис "1" в задані біти регістрів викликає установку відповідних виводів портів. Запис "0" не має ефекту. Якщо виводи сконфігуровані, як вхідні, або для них обрані альтернативні функції, то зміна станів відповідних бітів цих регістрів не має ефекту. Читання регістра IOxSET повертає значення, яке визначено попередніми записами в IOxSET та IOxCLR (або IOxPIN). Це значення не відображує вплив зовнішніх сигналів на лінію введення/виведення. Після скидан-

ня мікроконтролера значення бітів цих регістрів дорівнюють "0".

Регістри IOxCLR та FIOxCLR використовуються для скидання виводів порту, якщо вони налаштовані як лінії введення/виведення загального при-значення, і для них заданий режим виходів. Запис "1" в задані біти регістрів викликає скидання відповідних виводів портів і скидає відповідні біти в регістрах IOxSET та FIOxSET. Запис "0" не має ефекту. Якщо виводи сконфігуровані, як вхідні або для них обрані альтернативні функції, то зміна станів відповідних бітів цих регістрів не має ефекту. Після скидання мікроконтролера значення бітів цих регістрів дорівнюють "0".

## 3.3 Контрольні питання

1. Для чого призначені порти введення/виведення загального призначення МК (*General Purpose Input Output – GPIO*)?
2. Які особливості характерні для ліній введення/виведення загального призначення МК сімейства LPC2000?
3. Яким чином групуються лінії введення/виведення загального призначення МК сімейства LPC2000?
4. Яким чином використовуються лінії введення/виведення загального призначення МК LPC2148 у лабораторному стенді?
5. Яким чином засвітити світлодіод, на лабораторному стенді?
6. Дайте характеристику регістрам IOxPIN та FIOxPIN: номенклатура, призначення, особливості та застосування у лабораторному стенді.
7. Дайте характеристику регістрам IOxDIR та FIOxDIR: номенклатура, призначення, особливості та застосування у лабораторному стенді.
8. Дайте характеристику регістрам IOxSET та FIOxSET: номенклатура, призначення, особливості та застосування у лабораторному стенді.
9. Дайте характеристику регістрам IOxCLR та FIOxCLR: номенклатура, призначення, особливості та застосування у лабораторному стенді.

## *3.4 Хід роботи*

### 3.4.1 Підготовчі стадії

1. Ознайомитися з *LPC2148 User Manual*, Частина 8 [5].
2. Запустити проект *GPIO_FIO*.
3. Проаналізувати логіку роботи програми.
4. Запустити програму на стенді. Порівняти швидкості роботи звичайних і швидких портів введення/виведення (результати відображуються на екрані РКД – кількість перемикань портів введення/виведення за 0,1с).

### 3.4.2 Коментарі до програми

Текст головної функції програми наведено у лістингу 3.1.

main.c Лістинг 3.1

```
01 int main (void)
02 {
```

```
03        int GPIO, FIO;
04        unsigned char result1 [30];
05        unsigned char result2 [30];
06
07        /*************************************************/
08        /*           Приклад запису числа до LCD         */
09        /*************************************************/
10
11        InitLCD ();
12        SetBacklight (1);
13        LCDTextOut ("   Hands On 3   ", "    GPIO/FIO    ");
14
15        /*************************************************/
16        /*                  Приклад коду                 */
17        /*************************************************/
18
19        GPIO = GPIOPerformanceCheck ();
20        sprintf (result1, "GPIO %d ticks", GPIO);
21        FIO = FIOPerformanceCheck ();
22        sprintf (result2, "FIO  %d ticks", FIO);
23        LCDTextOut (result1, result2);
24
25        while (1);
26
27        return 0;
28  }
```

Рядки 11-13 призначені для виведення на РКД номера та назви лабораторної роботи. Аналогічні рядки будуть зустрічатися у наступних лабораторних роботах.

В рядках 19 та 21 виконується вимір продуктивності портів введення/виведення, підключених до периферійної шини та до локальної шини (докладніше див. лістинги 3.2 и 3.3).

Рядок 23 виводить результати на символьний дисплей.

У лістингах 3.2 та 3.3 наведені фрагменти файлу gpio.c з функціями вимірювання продуктивності переключення портів введення/виведення.

gpio.c (Перевірка швидкості GPIO) Лістинг 3.2

```
42  /* Перевірка швидкості GPIO
43   повертає час перемикання входив */
44  int GPIOPerformanceCheck (void)
45  {
46        int count = 0;
47        int prev_SCS = SCS;
48
49        /*************************************************/
50        /*          Ініціалізація GPIO                   */
51        /*************************************************/
52        SCS = 0x00;                   // Use GPIO (instead of FIO)
53        IO0DIR |= 0x0000FF00;   // Встановити P0.8 - P0.15 як виходи
54        IO0SET |= 0x0000FF00;         // Встановити P0.8 - P0.15
55
56        InitTimer0 ();                // Запуск таймера (Лаб. роб. №4)
57
58        while (!TimerIsUp)      // Доки таймер не дорахував до переривання
59        {
60              // Установити P0.12=1
61              IO0SET |= (1 << 12);
62              while (!(IO0PIN & (1 << 12))); // Чекати доки P0.12 високий
63
64              // Скинути P0.12=0
65              IO0CLR |= (1 << 12);
66              while (IO0PIN & (1 << 12));   // Чекати доки P0.12 низький
67              count ++;         // Підрахунок
68        }
69
70        // Відновлення попереднього стану SCS
71        SCS = prev_SCS;
72
73        return count;
```

```
74 }
```

Функції вимірювання продуктивності переключення портів введення/виведення запускають таймер та за час його роботи переключають стан портів з 1 в 0 та з 0 в 1 послідовно.

В рядках 52 та 86 обох функцій виконується установка регістра *System Control Status* (SCS), щоб підключити порти введення/виведення або до периферійної, або до локальної шини. Рядки 53, 54 і 87, 88 налаштовують виводи 8-15 порту 0 на вихід і установлюють їх у початковий одиничний стан. Рядки 61, 65 та 94, 96 подають на один з виводів сигнал логічної 1 або логічного 0, відповідно. В рядках 62, 66 і 95, 97 очікується факт переключення. Робота з таймером, його ініціалізація та обробка переривання є предметом дослідження іншої лабораторної роботи.

gpio.c (Перевірка швидкості FIO) Лістинг 3.3

```
76  /* Перевірка швидкості FIO
77  */
78  int FIOPerformanceCheck (void)
79  {
80        int count = 0;
81        int prev_SCS = SCS;
82
83        /*************************************************/
84        /*         Ініціалізація  FIO                    */
85        /*************************************************/
86        SCS = 0x01;                    // Використання FIO (замість GPIO)
87        FIO0DIR |= 0x0000FF00; // Налаштувати P0.8 - P0.15 як виходи
88        FIO0SET |= 0x0000FF00; // Встанови
89  [5].ти P0.8 - P0.15
90
91        InitTimer0 ();                        // Запуск таймера (Лаб. роб. №4)
92
93        while (!TimerIsUp)
94        {
95              FIO0SET |= (1 << 11);
96              while (!(FIO0PIN & (1 << 11)));// Чекати, поки P0.11 високий
97              FIO0CLR |= (1 << 11);
98              while (FIO0PIN & (1 << 11));// Чекати, поки P0.11 низький
99              count ++;
100       }
101
102       // Відновлення попереднього стану SCS
103       SCS = prev_SCS;
104
105       return count;
      }
```

*LPC2148 User Manual*, Частина 8

### *3.5 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні питання.
3. Схему програми вирішеної задачі зі схемою підключення портів.
4. Текст програми з коментарями, які відповідають схемі програми.
5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) вдалося дослідити: які особливості портів введення/виведення загального призначення виявили та як пройшло ознайомлення з методами роботи з

швидкими портами (*FIO*).

### *3.6 Орієнтовні варіанти завдань*

1. На світловипромінюючих діодах вивести вогник, що біжить зліва праворуч.
2. На світловипромінюючих діодах вивести вогник, що біжить справа ліворуч.
3. На світловипромінюючих діодах вивести тінь, що біжить зліва праворуч.
4. На світловипромінюючих діодах вивести тінь, що біжить справа ліворуч.
5. На світловипромінюючі діоди вивести інформаційну модель у вигляді двійкового лічильника.
6. Вивести інформаційну модель, задану викладачем, на світловипромінюючі діоди.

*Примітка*. Під час виконання зазначених завдань затримку можна реалізувати за допомогою циклічної процедури типу:

```
for (i = 0, i < DelayTime, i++)   {};
```

# 4 Лабораторна робота №4. Дослідження контролера переривань

***Мета роботи***: ознайомлення з поняттям векторного переривання та практичне дослідження контролера векторних переривань

## *4.1 Короткі теоретичні відомості*

Для того, щоб ввести чи вивести дані з периферійного пристрою, можна застосувати т. зв. полінг, тобто періодично зчитувати регістр обміну цього пристрою, аналізувати відповідний прапорець готовності даних та виконувати операцію обміну. В якості прикладу можна навести очікування натискання кнопки, коли програма має постійно звертатися до зчитування стану лінії порту, до якої кнопку підключено. Натискання кнопок порівняно рідкі, а момент натискання невідомий. Аби не пропустити натискання та сформувати вчасну реакцію на запит оператора, доводиться завантажувати процесор величезним обсягом холостої роботи, повертаючись в циклі до зчитування порту введення раз за разом. Аналогічні приклади можна навести із сфери обробки сигналів різноманітних інтерфейсів: від UART до USB.

Переривання – це ефективний механізм асинхронної взаємодії процесорного ядра з периферійними пристроями, коли, на відміну від полінгу (опитування) пристрій генерує запит на переривання, який потрапляє на спеціальний і окремий від процесора блок обробки переривань. Якщо дане переривання дозволене, процесор призупиняє роботу за поточною програмою та переходить до підпрограми обробки переривань (*Interrupt Service Routine, ISR*) для цього конкретного запиту. Після завершення обміну з периферійним пристроєм (скажімо, зчитування стану кнопки або виведення в порт) відбувається повернення до раніше перерваної програми.

Слід зауважити, що джерелом запитів переривань може бути також і підсистема обліку часу, яку в МК зазвичай реалізують за допомогою таких периферійних пристроїв, як таймери-лічильники. Після того, як таймер підрахує задану під час прогамного настроювання кількість тактових імпульсів, сигнал переповнення ініціює запит на переривання. Це дозволяє, зокрема, забезпечувати переключення центрального процесора на виконання різних задач за певним часовим розкладом. Такий незалежний від центрального процесора апаратний блок таймера, наприклад, використовують операційні системи реального часу.

В системі *ARM7* доступні 32 лінії запитів переривань. Переривання програмно призначаються трьома категоріями:

1. Швидке переривання (*FIQ*) для швидкої обробки переривання.
2. Векторне переривання (*vectored IRQ*) для загальних переривань.
3. Безвекторне переривання (*non-vectored IRQ*.)

Швидке переривання може спрацьовувати всього лише за 12 машинних циклів (200 *нс* при тактовій частоті процесора 60 *МГц*) на відміну від

звичайного векторного переривання, яке спрацьовує за 25 циклів (416 *нс* при 60 *МГц*). Це досягається за рахунок того, що обробник *FIQ* розташовується відразу після таблиці векторів переривань, тому немає необхідності робити зайві переходи.

Існує 16 векторних переривань. Векторні і невекторні переривання (*IRQ*) викликають обробник переривання (*ISR*). Читання з векторного регістра адреси переривання `VICVectAddr` дає адресу *ISR* і оновлює налаштування контролера, який маскує будь-які запити на переривання, що мають такий же, або більш низький пріоритет. Запис в регістр `VICVectAddr` вказує контролеру, що поточне переривання обслуговано. Це дозволяє активізуватися перериванню з більш низьким пріоритетом та розпочати його обробку. Переривання *FIQ* мають найвищий пріоритет. Векторні *IRQ* переривання мають найнижчий пріоритет.

Узагальнену структуру контролера переривань в складі МК LPC2000 наведено на рисунку 4.1.

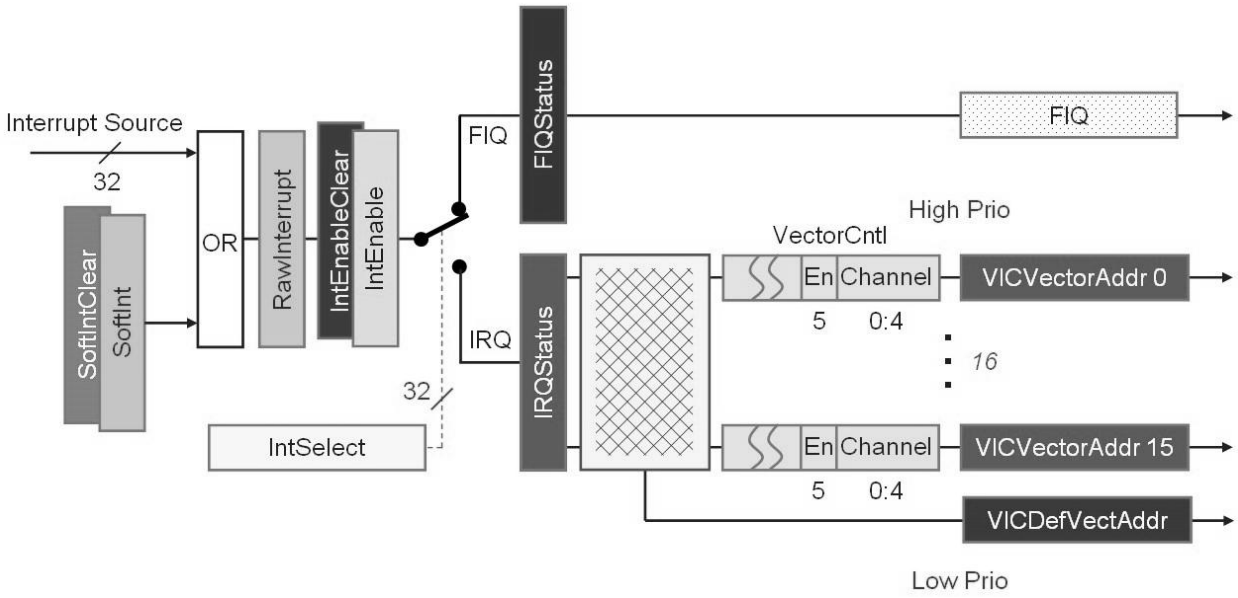



Рисунок 4.1 – Структура векторного контролера преривань

Організація роботи з перериваннями у власному проекті передбачає налаштування шляхом обміну з регістрами контролера переривань, які коротко описані в табл. 4.1.

Таблиця 4.1 – Опис регістрів контролера переривань

| **Назва** | **Опис** | **Доступ** |
|---|---|---|
| `VICIRQStatus` | Показує стан запитів переривань, які активовані і класифіковані, як *IRQ*. 1 вказує, на те, що переривання активне, і проводиться передача переривання процесору | RO |
| `VICFIQStatus` | Показує стан запитів переривань, які активовані і класифіковані як *FIQ*. 1 вказує, на те, що переривання активне, і проводиться передача переривання процесору. | RO |

| Назва | Опис | Доступ |
|---|---|---|
| VICRawIntr | Показує стан 32 ліній запиту переривань (в тому числі, – і програмних), незалежно від активації і класифікації переривання. 1 вказує на те, що переривання активне, і проводиться передача переривання процесору | RO |
| VICIntSelect | Вказує контролеру переривань, який тип переривання (*FIQ* або *IRQ*) робить джерело запиту. 1 вказує на *FIQ* переривання «0» – на *IRQ*. | R/W |
| VICIntEnable | Включає лінії запиту переривань, маскуючи джерела переривань для переривань IRQ<br>1 = Переривання дозволено.<br>0 = Переривання заборонено.<br>Запис 1 у відповідний біт дозволяє переривання. Запис 0 не має ніякого ефекту. (Див. VICIntEnClr) | R/W |
| VICIntEnClr | Очищує біти в регістрі VICIntEnable.<br>Запис 1 у біт очищає відповідний біт у регістрі VICIntEnable. Запис 0 не має ніякого ефекту | W |
| VICSoftInt | Використовується для того, щоб зробити програмне переривання Установка біта в 1 генерує програмне переривання для встановленого джерела переривання (до маскування). Високий біт встановлює відповідний біт в регістрі VICIntEnable. Низький біт не має ніякого ефекту. | R/W |
| VICSoftIntClear | Очищує біти в регістрі VICSoftInt. Високий біт очищує відповідний біт в регістрі VICSoftInt. Низький біт не має ніякого ефекту | W |
| VICProtection | Дозволяє або забороняє захищений доступ до регістрів. Коли він включений, можливі тільки привілейовані способи доступу (читання і запис) до регістрів контролера переривань. Коли вимкнений, можливі привілейовані і призначені для користувача способи доступу (читання і запис) до регістрів контролера переривань | R/W |
| VICVectAddr | Містить адресу обробника (*ISR*) активного переривання. Будь-які записи в регістр очищують переривання | R/W |
| VICDefVectAddr | Містить адресу *ISR* для безвекторних *IRQ* переривань | R/W |
| VICVectAddr0-15 | Містить адреси вектора *ISR* для і-го переривання | R/W |

| Назва | Опис | Доступ |
|---|---|---|
| VICVectCntl0-15 | Вибирають джерело переривання для і-го векторного переривання<br>0-4: Вибір джерела переривань. Можна вибрати будь-яке з 32 джерел переривань<br>5: дозволяє векторні переривання | R/W |

Докладну інформацію про контролер векторних переривань в складі МК LPC2000 можна знайти в рекомендованій літературі [3 - 7].

## 4.2 Контрольні питання

1. Яким чином можна оцінити стан кнопки у лабораторному стенді?
2. Які два основних механізми застосовують для обміну між центральним процесором і периферійним пристроєм? В чому полягають переваги та недоліки кожного з них?
3. Що таке переривання та яким чином воно реалізується в МК?
4. Які апаратні пристрої всередині МК дозволяють формувати функції часу незалежно від центрального процесора?
5. Якими категоріями призначаються переривання в МК *ARM7*?
6. В чому перевага швидкого переривання та як вона реалізується?
7. В чому різниця векторних і невекторних переривань?

## *4.3 Хід роботи*

### 4.3.1 Підготовча стадія

1. Користуючись керівництвом користувача *LPC2148 User Manual*, Частина 5 [5], дослідити особливості організації системи переривань в ARM7-МК.
2. Користуючись керівництвом користувача, Частина 15 [5], дослідити особливості роботи таймерів в ARM7-МК.
3. Відкрити проект 01_GPIO.
4. Дослідити логіку і послідовність роботи частини програми, що стосується роботи з перериванням за таймером.

### 4.3.2 Практична стадія

5. Скомпілювати проект і запустити програму на навчальному стенді *LPC2148 Education Board*.
6. Модифікувати проект відповідно до завдання.

### 4.3.3 Коментарі до програми

Код, що відповідає за ініціалізацію контролера переривань і таймера, наведений у лістингу 4.1.

gpio.c Лістинг 4.1

```
01 int TimerIsUp = 0;      // Скидання прапорця
02
03 /*************************************************/
04 /*          Обробник переривань від Таймера 0        */
05 /*************************************************/
06 __irq void Timer0ISR (void)       //Timer0 ISR
07 {
08       TimerIsUp = 1;              //Встановити ознаку факту переривання
09       T0IR = 0x01;                //Скинути прапорець переривання
10       VICVectAddr = 0;            //Скинути VIC
11       return;__
12 }
13
14 void InitTimer0 (void)
15 {
16       /*************************************************/
17       /*                  Ініціалізація VIC              */
18       /*************************************************/
19       VICDefVectAddr = (unsigned int) &Timer0ISR;
20       VICIntEnable = 0x10;    // Channel#4 - це Timer0
21       VICIntSelect = 0x00;    // Усі переривання - IRQ
22
23       /*************************************************/
24       /*                 Ініціалізація Timer0            */
25       /*************************************************/
26       T0MR0 = 1500000;      // Співпадіння (Timer match ~ 0.1 секунди)
27       T0MCR = 0x03;      // Переривання по Match0, скинути за співпадінням
28       T0PC = 0x01;            // Прескалер = 2
29       T0TC = 0x00;            // Скинути Timer counter
30       T0TCR = 0x01;           // Дозволити Timer
31       /*************************************************/
32       /*                  Скинути прапор                 */
33       /*************************************************/
34       TimerIsUp = 0;    // Скинути прапор у нуль
35
36       return;
37 }
38
```

У рядку 1 задається прапор, який прийме значення 1 в момент спрацювання таймера. Будь-яка функція може відстежувати стан цього прапора. Це – стандартний прийом для обробки переривання, тому що сам обробник не повинен виконувати будь-яких дій, а тільки сигналізувати про наявність переривання.

В даному випадку обробник переривання – це функція `Timer0ISR()`. Вона викликається у момент виникнення переривання, виставляє прапор наявності переривання для інших функцій (рядок 8), скидає прапор переривання таймера (рядок 9) і скидає контролер переривань (рядок 10). Рядки 9 і 11 потрібні для того, щоб обробник переривань викликався один раз при появі запиту на переривання.

Розглянемо функцію ініціалізації таймера і установки обробника переривання. У рядку 19 задається адреса функції, яка є обробником переривання по 4-му каналу (рядок 20). У рядку 21 вказується, що всі переривання є звичайними, а не швидкими.

Рядки 26…30 налаштовують таймер на спрацьовування через 0,1 секунди після запуску (рядок 26), виклик переривання і скидання таймера в 0 (рядок 27) та запуск таймера (рядок 30). Детальніше роботу таймера буде розглянуто в лабораторній роботі, присвяченій модулю широтно-імпульсної модуляції.

Приклад роботи з таким джерелом переривання наведений в лістингу 4.2.

main.c Лістинг 4.2

```
01 InitTimer0 ();                    // Start timer
02
03 while (!TimerIsUp)
04 {
05       // Очікуємо спрацювання таймера
06 }
```

Замість рядка 5 можуть виконуватися будь-які дії під час очікування спрацьовування таймера (див. попередню лабораторну роботу).

### *4.4 Вимоги до звіту по роботі*

Звіт про виконання роботи повинен містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні питання.
3. Схему програми вирішеної задачі зі схемою підключення індикатора.
4. Текст програми з докладними коментарями, які відповідають схемі програми.
5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. З чим саме конкретно (перерахувати по пунктах) вдалося ознайомитися та які особливості ARM7-МК вдалося дослідити?

### *4.5 Орієнтовні варіанти завдань*

Із використанням переривань від таймерів реалізувати затримки для наступної інформаційної моделі:

1. На світловипромінюючих діодах вивести вогник, що біжить зліва праворуч.
2. На світловипромінюючих діодах вивести вогник, що біжить справа ліворуч.
3. На світловипромінюючих діодах вивести тінь, що біжить зліва праворуч.
4. На світловипромінюючих діодах вивести тінь, що біжить справа ліворуч.
5. На світловипромінюючі діоди вивести інформаційну модель у вигляді двійкового лічильника.
6. На світловипромінюючі діоди вивести інформаційну модель у вигляді «потяга». [redacted], [redacted], , [redacted], [redacted], …
7. Вивести інформаційну модель, задану викладачем, на світловипромінюючі діоди.

Із використанням переривання за допомогою кнопки змінити напрям руху для заданої інформаційної моделі.

# 5 Лабораторна робота №5. Дослідження модуля ШІМ (PWM)

***Мета роботи***: практичне засвоєння методів роботи з модулем широтно-імпульсної модуляції

## *5.1 Короткі теоретичні відомості*

Широтно-імпульсний модулятор (ШІМ) – це модуль, по суті, схожий на таймер загального призначення, але з додатковою можливістю періодично генерувати імпульси певної тривалості. За допомогою ШІМ можна, наприклад, керувати яскравістю світлодіоду, подаючи на нього живлення в певні моменти часу. У даній роботі модуль ШІМ керує яскравістю трьох різнокольорових світлодіодів, створюючи при цьому світіння певного кольору.

Схему роботи модуля ШИМ наведено на рисунку 5.1. Часові діаграми наведені на рисунку 5.2.

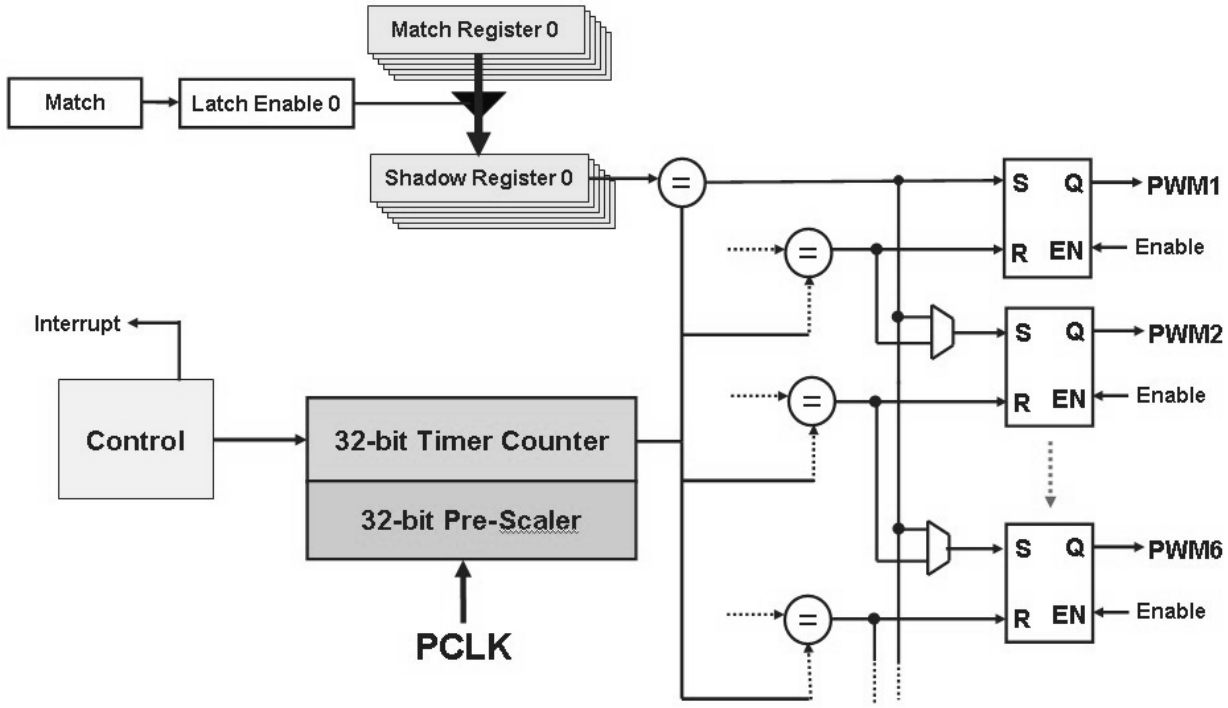



Рисунок 5.1 – Схема роботи модуля ШІМ.

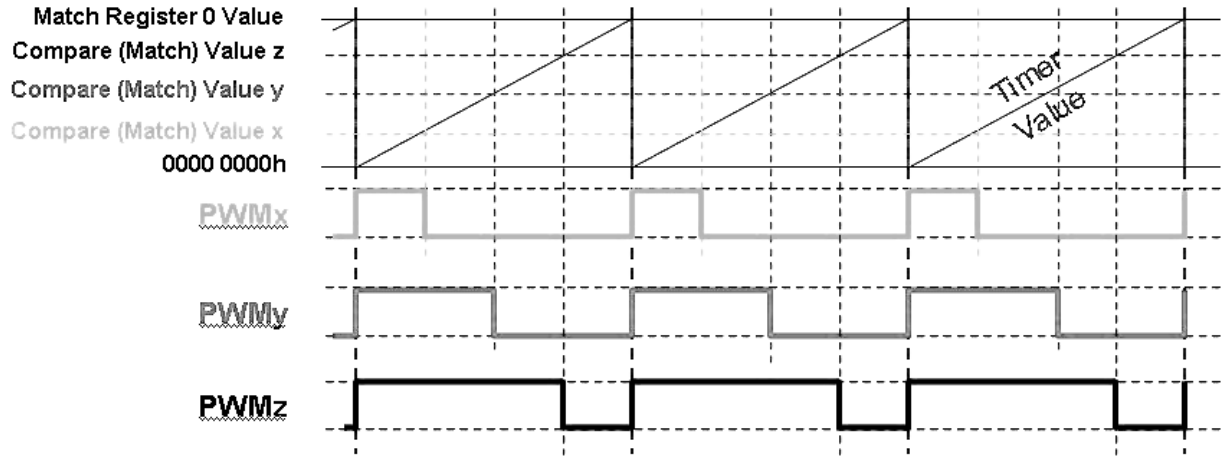



Рисунок 5.2 – Часові діаграми роботи модуля ШІМ

Схема підключення кольорових світлодіодів приведена на рисунку 1.3. Як видно, вони підключені до виводів 7…9 порту 0, які програмно можна сполучити із модулем ШІМ всередині мікроконтролера.

## *5.2 Опис регістрів модуля ШІМ*

Опис регістрів модуля ШІМ наведений у таблиці 5.1.

Таблиця 5.1 – Опис регістрів модуля ШІМ

| Опис | Доступ | При скиданні |
|---|---|---|
| **PWM_TC (**PWM Timer Counter**)** | | |
| 32-бітний таймер-лічильник PWM. Інкрементується з кожним PR+1 циклом тактової частоти. TC керується через TCR | RW | 0 |
| **PWM_TCR** | | |
| Регістр керування таймером PWM (PWM Timer Control Register). Використовується для керування функціями таймера-лічильника. За допомогою TCR таймер-лічильник може бути заблокований або встановлений в стан "0" | R/W | 0 |

32-розрядний таймер-лічильник PWM (PWMTC) збільшується, коли лічильник попереднього дільника (прескалер) досягає свого кінцевого значення. Поки він не скинувся при досягненні верхньої межі, PWM_TC буде рахувати в прямому напрямку до величини 0xFFFF FFFF, а потім автоматично перейде назад до величини 0x0000 0000. Ця подія не викликає переривання, а регістр співпадіння (збігу, *match*) може бути використаний для визначення переповнення, коли це необхідно.

Функція кожного з бітів регістра керування часом PWM (PWM_TCR). вказана в таблиці 5.2.

Таблиця 5.2 – Опис бітів регістра керування таймером PWM (PWM_TCR)

| Біт PWM_TCR | Функція | Опис | Скинуте значення |
|---|---|---|---|
| 0 | Дозвіл | Коли 1, дозволено рахувати як основному лічильнику PWM, так і попередньому лічильнику (прескалеру) PWM. Коли 0, лічильники заблоковані. | 0 |
| 1 | Скидання | Коли 1, лічильник PWM та прескалер PWM синхронно скидаються за наступним фронтом PCLK. Лічильники залишаються скинутими, доки TCR[1] не повернеться до "0". | 0 |
| 2 | Зарезервовано | Програми користувача не повинні записувати одиниці в резервовані біти. | NA |
| 3 | Дозвіл PWM | Коли 1, режим PWM дозволено. Режим PWM змушує тіньові регістри працювати разом з регістрами співпадіння. Програмний запис до регістра співпадіння не діятиме доти, поки не буде встановлений відповідний біт у PWMLER, а потім відбудеться подія PWM Match 0. Регістр співпадіння PWM, який визначає швидкість PWM, має бути налаштованим до того, як режим PWM буде дозволеним. | 0 |

**PWM_PR** (PWM Prescale Register)

| Опис | Доступ | Скинуте значення |
|---|---|---|
| Регістр прескалера PWM. Зростає з кожним PR+1 циклом тактування. | R/W | 0 |

32-бітний регістр прескалера (масштабування) PWM задає роздільну здатність та максимальне значення лічильника прескалера PWM_PC в тактах синхрогенератора.

**PWM_PC** (PWM Prescale Counter register)

| Опис | Доступ | Скинуте значення |
|---|---|---|
| Лічильник масштабування ШІМ. 32 бітний лічиль- | R/W | 0 |

| Опис | Доступ | Скинуте значення |
|---|---|---|
| ник прескалера ШІМ PWM_PC збільшується до значення, що зберігається в PWM_PR. | | |

32-розрядний лічильник прескалера ШІМ PWM_PC (PWM Prescale Counter) контролює діленням PCLK на деяку постійну величину, перш ніж тактові імпульси потраплять до таймера-лічильника ШІМ. Це дозволяє контролювати співвідношення роздільної здатності таймера у порівнянні з максимальним часом перш, ніж таймер переповниться. Цей регістр збільшується на кожному такті PCLK. Коли він досягає значення, що зберігається в регістрі попереднього масштабування ШІМ PWM_PR, лічильник таймера PWM збільшується, а PWM_PC скидається на наступному PCLK. Це призводить до збільшення PWM TC на кожному PCLK, коли PWMPR = 0, кожні 2 PCLK, коли PWMPR = 1 і т. д.

**PWMMR0 – PWMMR6** (PWM Match Registers)

Значення 32-розрядних регістрів збігу PWM (PWM Match Registers) постійно порівнюються з значенням PWM Timer Counter. Коли два значення рівні, автоматично можуть спрацьовувати дії. Можливості дії полягають у створенні переривання, скидання лічильника таймера PWM або зупинення таймера. Дії контролюються параметрами у регістрі PWMMCR.

**PWM_PCR** (PWM Control Register)

| Опис | Доступ | Скинуте значення |
|---|---|---|
| Регістр керування ШІМ. Дозволяє виведення PWM і вибирає типи каналів PWM: контроль одного фронту або двох фронтів. | R/W | 0 |

Регістр керування ШІМ використовують для дозволу та вибору кожного каналу ШІМ. Функції кожного з біт вказані в таблиці:

| PWMPCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 1:0 | Зарезервовано | Програма користувача не повинна записувати одиниці в зарезервовані біти. Значення, прочитані з зарезервованих біт, не визначені | NA |
| 2 | PWMSEL2 | При нулі обирається режим PWM2 з керуванням одним фронтом вихідного сигналу. При 1 обирається режим керування двома фронтами PWM2. | 0 |
| 3 | PWMSEL3 | При нулі обирається режим PWM3 з керуванням одним фронтом вихідного сигналу. При 1 обирається режим | 0 |

| PWMPCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| | | керування двома фронтами PWM3. | |
| 4 | PWMSEL4 | При нулі обирається режим PWM4 з керуванням одним фронтом вихідного сигналу. При 1 обирається режим керування двома фронтами PWM4. | 0 |
| 5 | PWMSEL5 | При нулі обирається режим PWM5 з керуванням одним фронтом вихідного сигналу. При 1 обирається режим керування двома фронтами PWM5. | 0 |
| 6 | PWMSEL6 | При нулі обирається режим PWM6 з керуванням одним фронтом вихідного сигналу. При 1 обирається режим керування двома фронтами PWM6. | 0 |
| 8:7 | Зарезервовано | Програма користувача не повинна записувати одиниці в зарезервовані біти. Значення, прочитані з зарезервованих біт, не визначені | NA |
| 9 | PWMENA1 | При 1 дозволяються дані на виході PWM1. При 0 дані на виході PWM1 блокуються. | 0 |
| 10 | PWMENA2 | При 1 дозволяються дані на виході PWM2. При 0 дані на виході PWM2 блокуються. | 0 |
| 11 | PWMENA3 | При 1 дозволяються дані на виході PWM3. При 0 дані на виході PWM3 блокуються. | 0 |
| 12 | PWMENA4 | При 1 дозволяються дані на виході PWM4. При 0 дані на виході PWM4 блокуються. | 0 |
| 13 | PWMENA5 | При 1 дозволяються дані на виході PWM5. При 0 дані на виході PWM5 блокуються. | 0 |
| 14 | PWMENA6 | При 1 дозволяються дані на виході PWM6. При 0 дані на виході PWM6 блокуються. | 0 |
| 15 | Зарезервовано | Програма користувача не повинна записувати одиниці в зарезервовані біти. Значення, прочитані з зарезервованих біт, не визначені | NA |

**PWM_MCR** (PWM Match Control Register)

| Опис | Доступ | Скинуте значення |
|---|---|---|
| Регістр керування співпадінням PWM. MCR використовується для керування, коли виникає співпадіння при генерації переривань та установці TC в "0". | R/W | 0 |

Регістр керування співпадінням PWMMCR використовується для контролю того моменту, коли один з регістрів співпадіння PWM дорівнює вмісту лічильника-таймера. Функції кожного з біт вказані в таблиці:

| PWM MCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 0 | Перериван-ня на PWMMR0I | При 1: генерується переривання, коли PWMMR0 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 1 | Скидання на PWMMR0R | При 1: PWMTC скинеться, якщо PWMMR0 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 2 | Зупинка на PWMMR0S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR0 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |
| 3 | Перериван-ня на PWMMR1I | При 1: генерується переривання, коли PWMMR1 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 4 | Скидання на PWMMR1R | При 1: PWMTC скинеться, якщо PWMMR1 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 5 | Зупинка на PWMMR1S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR1 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |
| 6 | Перериван-ня на PWMMR2I | При 1: генерується переривання, коли PWMMR2 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 7 | Скидання на PWMMR2R | При 1: PWMTC скинеться, якщо PWMMR2 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 8 | Зупинка на PWMMR2S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR2 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |
| 9 | Перериван-ня на PWMMR3I | При 1: генерується переривання, коли PWMMR3 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 10 | Скидання на PWMMR3R | При 1: PWMTC скинеться, якщо PWMMR3 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 11 | Зупинка на | При 1: PWMTC і PWMPC будуть зу- | 0 |

| PWM MCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| | PWMMR3S | пинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR3 співпаде з PWMTC. При 0 ця дія заблокована. | |
| 12 | Перериван-ня на PWMMR4I | При 1: генерується переривання, коли PWMMR4 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 13 | Скидання на PWMMR4R | При 1: PWMTC скинеться, якщо PWMMR4 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 14 | Зупинка на PWMMR4S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR4 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |
| 15 | Перериван-ня на PWMMR5I | При 1: генерується переривання, коли PWMMR5 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 16 | Скидання на PWMMR5R | При 1: PWMTC скинеться, якщо PWMMR5 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 17 | Зупинка на PWMMR5S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR5 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |
| 18 | Перериван-ня на PWMMR6I | При 1: генерується переривання, коли PWMMR6 співпаде із значенням у PWMTC. При 0 це переривання заблоковане. | 0 |
| 19 | Скидання на PWMMR6R | При 1: PWMTC скинеться, якщо PWMMR6 співпаде із ним. При 0 це скидання заблоковане. | 0 |
| 20 | Зупинка на PWMMR6S | При 1: PWMTC і PWMPC будуть зупинені та PWMTCR[0] буде встановлений в 0, якщо PWMRR6 співпаде з PWMTC. При 0 ця дія заблокована. | 0 |

**PWM_LER** (PWM Latch Enable Register)

| Опис | Доступ | Скинуте значення |
|---|---|---|
| Регістр дозволу защілки PWM. Дозволяє використання нових співпавших значень PWM | R/W | 0 |

Регістр дозволу защілки PWM використовується для керування оновленням регістрів співпадінь (PWM Match), коли вони використовуються

для генерації ШІМ. Коли програма записує у місце розташування регістра PWM Match, якщо таймер знаходиться в режимі PWM, значення зберігається в тіньовому регістрі. Коли відбувається подія PWM Match 0 (зазвичай також скидається таймер у режимі PWM), вміст тіньових регістрів буде перенесено до фактичних регістрів збігів, якщо відповідний біт у регістрі дозволу защібки встановлено. У цей момент нові значення набудуть чинності та визначать дію наступного циклу ШІМ. Після передачі нових значень всі біти LER будуть автоматично очищені. Поки не буде встановлено відповідний біт у PWMLER та відбудеться подія PWM Match 0, будь-яке значення, записане в регістри PWM Match, не впливає на роботу PWM.

Наприклад, якщо PWM2 налаштовується для роботи з подвійним фронтом і виконується в даний час, типовою послідовністю подій для зміни часових інтервалів буде:

•Записати нове значення до регістру PWM Match1.

•Записати нове значення до регістру PWM Match2.

•Записати до PWMLER, встановлюючи біти 1 і 2 одночасно.

•Змінені значення набудуть чинності при наступному скиданні таймера (коли відбувається подія PWM Match 0).

Порядок запису двох регістрів PWM Match не важливий, оскільки жодні значення не будуть використовуватися, доки не буде записано PWMLER. Це гарантує, що обидва значення вступають в дію одночасно, якщо це потрібно. Одне значення може бути змінено таким же чином, якщо це необхідно.

Функції кожного з біт PWMLER вказані в таблиці:

| **PWM LER** | **Функції** | **Опис** | **Скинуте значення** |
|---|---|---|---|
| 0 | Дозвіл защібки співпадіння PWM 0 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 0, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 1 | Дозвіл защібки співпадіння PWM 1 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 1, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 2 | Дозвіл защібки співпадіння PWM 2 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 2, стати діючим, коли таймер скидається наступною | 0 |

| PWM LER | Функції | Опис | Скинуте значення |
|---|---|---|---|
| | | подією співпадіння PWM. | |
| 3 | Дозвіл защібки співпадіння PWM 3 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 3, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 4 | Дозвіл защібки співпадіння PWM 4 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 4, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 5 | Дозвіл защібки співпадіння PWM 5 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 5, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 6 | Дозвіл защібки співпадіння PWM 6 | Запис 1 у цей біт дозволяє останньому значенню, записаному в регістр співпадіння PWM 6, стати діючим, коли таймер скидається наступною подією співпадіння PWM. | 0 |
| 7 | Зарезервовано | Програма користувача не повинна записувати одиниці в зарезервовані біти. Значення, прочитані з зарезервованих біт, не визначені | NA |

## 5.3 Контрольні питання

1. Для чого призначений широтно-імпульсний модулятор (ШІМ) в складі МК *ARM*7?
2. Наведіть часові діаграми роботи модуля ШІМ.
3. За допомогою чого можна змінити період імпульсів, які генерує модуль ШІМ?
4. За допомогою чого можна змінити скважність імпульсів, які генерує модуль ШІМ?
5. Для чого призначений регістр прескалера (масштабування) широтно-імпульсного модулятора (ШІМ) в складі МК *ARM*7?

## *5.4 Хід роботи*

### 5.4.1 Підготовча стадія

1. Користуючись керівництвом користувача, Частина 16 [5],, дослідити особливості організації модуля ШІМ в ARM7-МК.
2. Користуючись керівництвом користувача, дослідити особливості режимів ШІМ в ARM7-МК.
3. Відкрити проект 02_PWM.
4. Дослідити логіку і послідовність роботи частини програми, що стосується налаштування та роботи модуля ШІМ.

### 5.4.2 Практична стадія

1. Скомпілювати проект і запустити програму на навчальному стенді *LPC2148 Education Board*.
2. Модифікувати проект відповідно до завдання.

### 5.4.3 Коментарі до програми

Програма виконується у вигляді трьох послідовних циклів змінювати яскравості кольорових світлодіодів.

Код, що відповідає за налаштування і роботу з модулем ШІМ, наведений у лістингу 5.1.

pwm.c Лістинг 5.1

```
01 void InitPWM (void)
02 {
03       /*****************************************************/
04       /*   Альтернативна функція PIN GPIO 0.7, 8, 9        */
05       /*****************************************************/
06       PINSEL0 |= (2 << 14);         // GPIO 0.7 to PWM2
07       PINSEL0 |= (2 << 16);         // GPIO 0.8 to PWM4
08       PINSEL0 |= (2 << 18);         // GPIO 0.9 to PWM6
09
10       /*****************************************************/
11       /* Встановлення регістрів співпадіння (match)        */
12       /*****************************************************/
13       PWMMR0 = 0x00000FFF;          // Relative signal length
14       PWMMR2 = 0x00000000;          // Default set to zero
15       PWMMR4 = 0x00000000;          // Default set to zero
16       PWMMR6 = 0x00000000;          // Default set to zero
17
18       /*****************************************************/
19       /*      Скидання лічильника PWM за співпадінням      */
20       /*****************************************************/
21       PWMMCR = (1 << 1);
22
23       /*****************************************************/
24       /*  Копіювання регістрів співпадіння з тіньової пам'яті   */
25       /*****************************************************/
26       PWMLER = (1) | (1 << 2) | (1 << 4) | (1 << 6);
27
28       /*****************************************************/
29       /*               Set PWM2,4,6 enabled                */
30       /*****************************************************/
31       PWMPCR |= (1 << 10);          // Set PWM2 enabled
32       PWMPCR |= (1 << 12);          // Set PWM4 enabled
33       PWMPCR |= (1 << 14);          // Set PWM6 enabled
34
35       /*****************************************************/
36       /*               Enable counter and PWM              */
37       /*****************************************************/
```

```
38        PWMTCR = (1) | (1 << 3);
39  }
40
41
42
43  void SetRGB (int Red, int Green, int Blue)
44  {
45        /*****************************************************/
46        /*              Fill match registers with values     */
47        /*****************************************************/
48        PWMMR2 = Red;
49        PWMMR4 = Blue;
50        PWMMR6 = Green;
51
52        /*****************************************************/
53        /*          Copy match registers from shadow memory  */
54        /*****************************************************/
55        PWMLER = (1) | (1 << 2) | (1 << 4) | (1 << 6);
56  }
```

Розглянемо ініціалізацію модуля ШІМ (функція InitPWM). Рядки 6…8 відключають виводи від GPIO і підключають їх до модуля ШІМ. Рядок 13 встановлює період генеруємого сигналу, а рядки 14…16 встановлюють значення регістра збігу в 0, тобто фактично блокують сигнал з ШІМ. Інші функції, які використовують ШІМ, будуть налаштовувати ці регістри на необхідні значення.

Рядок 21 вказує на те, що лічильник модуля ШІМ повинен скинутися за однаковості його вмісту регістру збігу.

Рядок 26 необхідний для можливості одночасного включення всіх каналів ШИМ, коли це необхідно.

Рядок 38 включає лічильник модуля ШИМ.

Мета функції SetRGB() – встановлення заданої яскравості кольорових світлодіодів. Тут все аналогічно функції ініціалізації, проте в регістри збігу записуються ненульові значення. Якщо, наприклад, PWMMR0 = 0xFFF (= 4095), і задати PWMMR2 = 0x200 (= 512), то живлення на світлодіод буде подаватися протягом (512/4095) * 100% = 12,5% часу. Відповідно, його яскравість буде дорівнювати 12,5% максимальної.

### *5.5 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.

2. Письмові відповіді на контрольні питання.

3. Схему програми вирішеної задачі зі схемою підключення індикатора.

4. Текст програми з докладними коментарями, які відповідають схемі програми.

5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Які саме особливості ШІМ конкретно (перерахувати по пунктах) дозволяє досліджувати навчальний стенд?

### *5.6 Орієнтовні варіанти завдань*

**Завдання 1**. Використовуючи ШІМ, виконати завдання з попередньої роботи – біжучий вогник (тінь) з заданою викладачем яскравістю.

**Завдання 2 (варіанти)**.

1) Засвітити світлодіоди R та G на повну потужність. За кожним натисканням на кнопку збільшувати яскравість синього світлодіоду на 25% до 100%, а потім повертатися в початкове положення.

2) Засвітити світлодіоди B та G на повну потужність. За першим натисканням на кнопку протягом 5 сек. збільшувати яскравість червоного світлодіоду до 100%. За другим натисканням на кнопку повернутися в початкове положення.

3) Засвітити червоний світлодіод на повну потужність. За першим натисканням на кнопку протягом 5 сек. збільшувати яскравість зеленого та синього світлодіодів до 100%. За другим натисканням на кнопку повернутися в початкове положення.

4) Погасити усі світлодіоди. При кожному натисканні на кнопку збільшувати яскравість червоного світлодіоду на 25% до 100%, а потім повертатися в початкове положення. За допомогою джойстика змінювати яскравість зеленого (вгору – униз) та синього (вправо – вліво) світлодіодів.

5) В режимі очікування кожен RGB-світлодіод загорається і гасне (мерехтить) з однаковою частотою 0,5 Гц. При натисненні на кнопку частота збільшується на 5%.

6) В режимі очікування кожен RGB загорається і гасне (мерехтить) з однаковою частотою 1 Гц. При першому натисненні на кнопку мерехтіння припиняється, при повторному – поновлюється початковий стан. За допомогою джойстика змінювати частоту мерехтіння червоного (вгору – униз) та зеленого (вправо – вліво) світлодіодів.

**Завдання 3.** Використовуючи ШІМ, реалізувати «біжучий вогник» на RGB-світлодіоді одночасно з обертанням крокового двигуна, повний оберт якого буде відповідати одному проходу біжучого вогника. При обертанні двигуна проти годинникової стрілки біжучий вогник рухається в іншому напрямку. При зміні швидкості обертання змінювати і швидкість біжучого вогню.

# 6 Лабораторна робота №6. Дослідження модуля АЦП (ADC)

***Мета роботи***: практичне засвоєння методів роботи з вбудованим аналогово-цифрового перетворювачем (АЦП)

## *6.1 Короткі теоретичні відомості*

У мікроконтролерах сімейства LCP2xxx АЦП має розрядність 10 біт і працює за принципом послідовного наближення. АЦП може мати від 6 до 14 незалежних каналів, в залежності від конкретної моделі МК. Так, кристал LPC2148 дозволяє працювати з 14 аналоговими сигналами.

Схему підключення двох аналогових входів AD0.2 та AD0.1 наведено на рисунку 1.12. У технічній документації на МК номери модулів АЦП 1 та 2 скорочено позначають символом «x».

**Короткий опис регістрів АЦП**

| **Регістр** | **Опис** |
|---|---|
| ADxCR | Керуючий регістр АЦПx |
| ADxGDR | Містить ознаку закінчення перетворення |
| ADxSTAT | Містить інформацію про поточний стан АЦП |
| ADxINTEN | Містить інформацію про переривання від сигналів АЦП |
| ADxDR0… ADxDR7 | Містить результати перетворення по кожному з каналів |

Опис окремих бітів регістру ADxCR для налаштування АЦП наведений у таблиці.

| **Розряди** | **Значення** |
|---|---|
| 7:0 | Вибір робочих каналів АЦП |
| 15:8 | Дільник шини периферійних пристроїв |
| 16 | При 1 перетворення повторюється самостійно, при 0 – за запитом |
| 19:17 | Визначає розрядність перетворення.<br>000 – 10 біт<br>…<br>111 – 3 біта |
| 20 | Зарезервовано |
| 21 | 1 – вмикання живлення |
| 23:22 | Зарезервовано |
| 26:24 | Визначає момент початку перетворення при біті 16, рівному 0 |
| 27 | 1 – перетворення по зростаючому фронту керуючого сигналу, 0 – по спадаючому |
| 31:28 | Зарезервовано |

Зчитування значення з АЦП (регістр ADxGDR)

| Розряди | Значення |
|---|---|
| 15:6 | Перетворене значення |
| 26:24 | Канал АЦП, з якого отримано значення |
| 30 | 1, якщо хоча б одне перетворення не було зчитано, і тому перезаписано |
| 31 | 1, коли перетворення закінчено |

## 6.2 Контрольні питання

1. Для чого призначений аналого-цифровий перетворювач (АЦП) в складі МК *ARM7*? Наведіть його основні характеристики.
2. Дайте коротку характеристику регістрам модуля АЦП в складі МК *ARM7* (номенклатура та призначення).
3. Яким чином можна використати модуль АЦП в складі МК *ARM7* в рамках роботи із навчальним стендом LPC2148 Education Board?

## *6.3 Хід роботи*

### 6.3.1 Підготовчі стадії

1. Запустіть тестову програму проекту 03_ADC.
2. Запустіть тест на платі та спостерігайте результати роботи.

### 6.3.2 Коментарі до тестової програми

Програма зчитує значення з АЦП і засвічує один із світлодіодів відповідно рівню вхідного сигналу.

main.c Лістинг 6.1

```
01 int main (void)
02 {
03       /*************************************************/
04       /*            Write Hands On number to LCD             */
05       /*************************************************/
06
07       InitLCD ();
08       SetBacklight (1);
09       LCDTextOut ("   Hands On 3   ", "  ADC Converter ");
10
11       /*************************************************/
12       /*                    Hands On code                      */
13       /*************************************************/
14
15       InitADC ();             // see "adc.c"
16
17       IODIR0 |= (0xFF) << 8;  // Set LEDs as outputs (for indication)
18
19       while (1)
20       {
21             IOSET0 = (0xFF) << 8;              // Clear leds
22
23             // ADC is 10-bit width
24             // values are stored in 0..1023 range
25
26             // Fill leds with ADC value
```

```
27              IOCLR0 = (1 << (ADCReadValue () >> 7)) << 8;
28              Sleep (100);
29        }
30
31        return 0;
32  }
33
```

У рядку 15 ініціалізується АЦП (див. Лістинг 6.2). Рядок 17 встановлює в якості виходів виводи, до яких підключені світлодіоди.

У рядку 27 відбувається зчитування значення з АЦП (див. Лістинг 6.2). Зсув праворуч на 7 необхідний для приведення діапазону 0...1023 до діапазону 0...7, тому що на платі всього 7 світлодіодів. Зсув 1 на отримане значення ліворуч вибирає номер світлодіоду, що запалюється. Зсув результату ліворуч на 8 переносить отриманий біт в розряди, що відповідають ніжкам МК P0.8..P0.15 для запалювання світлодіодів.

adc.c Лістинг 6.2.

```
01  void InitADC (void)
02  {
03        /*************************************************************/
04        /*            Connect PIN connect block to ADC0.1            */
05        /*************************************************************/
06        PINSEL1 |= (1 << 24);
07
08        /*************************************************************/
09        /*                      Configure ADC                        */
10        /*************************************************************/
11        ADCR =      (1 << 1) |        // Select ADC0.1 channel is active
12                    (4 << 8) |        // Set Clock divider
13                    (1 << 16) |       // Set BURST
14                    (0 << 17) |       // Set ADC resolution
15                    (1 << 21) |       // Power on ADC
16                    (1 << 24);        // Start conversion now
17  }
18
19  int ADCReadValue (void)
20  {
21        int ADCValue;
22
23        /*************************************************************/
24        /*                    Read value from ADC                    */
25        /*************************************************************/
26        while (((ADCValue = ADDR) & 0x80000000) == 0);
27
28        /*************************************************************/
29        /*       Separate ADC value from other information            */
30        /*************************************************************/
31        return (ADCValue >> 6) & (0x3FF);
32  }
33
```

Розглянемо ініціалізацію АЦП (InitADC). Рядок 6 підключає вхід до АЦП. Рядки 11…16 вибирають режим роботи АЦП: активний канал 1, дільник частоти = 4, режим автоматичного повтору перетворення, роздільна здатність АЦП – 10 біт.

Функція ADCReadValue очікує на ознаку закінчення перетворення (рядок 26) і видає перетворене значення (рядок 31). Зсув праворуч на 6 необхідний, щоб перенести перетворене значення в молодші біти з розрядів

15:6. Логічне множення на 0x3FFF видаляє всю іншу інформацію тому, що в цьому випадку вона не потрібна.

### *6.4 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні питання.
3. Схему програми вирішеної задачі зі схемою підключення зовнішнього пристрою.
4. Текст програми з докладними коментарями, які відповідають схемі програми.
5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) вдалося дослідити у вбудованих аналогово-цифрових перетворювачах?

### *6.5 Орієнтовні варіанти завдань*

1. Реалізувати регулювання яскравості світіння заданої комбінації світлодіодів LED5…LED12 за допомогою потенціометра.
2. Реалізувати регулювання яскравості світіння трикольорового світлодіоду за допомогою двох потенціометрів.
3. Реалізувати регулювання яскравості світіння світлодіодної матриці 8x8 за допомогою потенціометра.
4. Реалізувати регулювання швидкості обертання крокового двигуна за допомогою потенціометра.
5. Реалізувати регулювання гучності зумера за допомогою потенціометра.

Приклади комплексних завдань знаходяться в Додатку Г (Табл. Г.4).

# 7 Лабораторна робота №7. Дослідження послідовного інтерфейсу UART

***Мета роботи***: практичне засвоєння методів роботи з послідовним інтерфейсом передачі даних UART

## *7.1 Короткі теоретичні відомості*

UART є модулем послідовної передачі даних з підтримкою апаратного контролю та модемних функцій. Основні особливості:

- 16-бітна черга прийому-передачі.
- Регістр розміщення відповідає промисловому стандарту 550.
- FIFO тригер приймача показує 1, 4, 8,14 біт.
- Вбудований контролер швидкості передачі.

**Опис портів**

| **Назва порту** | **Тип** | **Опис** |
|---|---|---|
| RxD0 | Введення | Послідовне введення (даних). Отримувані послідовно дані. |
| TxD0 | Виведення | Послідовне виведення (даних). Передані послідовно дані. |

**Опис регістрів UART**

Карта регістрів представлена в таблиці 7.1.

**UART0_RBR**

| **Опис** | **Доступ** | **Скинуте значення** | **Адреса** |
|---|---|---|---|
| Буферний регістр прийому | RO | Не визначене | 0×E000C000 |

UORBR є старшим байтом FIFO-стеку приймача UART0 Rx. Старший байт з Rx FIFO містить "найстаріший" отриманий символ і може бути прочитаний через інтерфейс шини. Найменший біт (LSB, біт 0) представляє "найстаріший" отриманий біт. Якщо отриманий символ менше 8 біт, то невикористовувані старші біти (MSB) заповнюються нулями.

**UART0_DLL**

| **Опис** | **Доступ** | **Скинуте значення** | **Адреса** |
|---|---|---|---|
| Дільник регістра-защібки LSB | R/W | 0×01 | 0×E000C000 |

Таблиця 7.1 – Карта регістрів UART

| Name | Description | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 | Access | Reset Value* | Address |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| U0RBR | Receiver Buffer Register | MSB | | | READ DATA | | | | LSB | RO | un-defined | 0xE000C000 DLAB = 0 |
| U0THR | Transmit Holding Register | MSB | | | WRITE DATA | | | | LSB | WO | NA | 0xE000C000 DLAB = 0 |
| U0IER | Interrupt Enable Register | 0 | 0 | 0 | 0 | 0 | Enable Rx Line Status Interrupt | Enable THRE Interrupt | Enable Rx Data Available Interrupt | R/W | 0 | 0xE000C004 DLAB = 0 |
| U0IIR | Interrupt ID Register | FIFOs Enabled | | 0 | 0 | IIR3 | IIR2 | IIR1 | IIR0 | RO | 0x01 | 0xE000C008 |
| U0FCR | FIFO Control Register | Rx Trigger | | Reserved | | - | Tx FIFO Reset | Rx FIFO Reset | FIFO Enable | WO | 0 | 0xE000C008 |
| U0LCR | Line Control Register | DLAB | Set Break | Stick Parity | Even Parity Select | Parity Enable | Number of Stop Bits | Word Length Select | | R/W | 0 | 0xE000C00C |
| U0LSR | Line Status Register | Rx FIFO Error | TEMT | THRE | BI | FE | PE | OE | DR | RO | 0x60 | 0xE000C014 |
| U0SCR | Scratch Pad Register | MSB | | | | | | | LSB | R/W | 0 | 0xE000C01C |
| U0DLL | Divisor Latch LSB | MSB | | | | | | | LSB | R/W | 0x01 | 0xE000C000 DLAB = 1 |
| U0DLM | Divisor Latch MSB | MSB | | | | | | | LSB | R/W | 0 | 0xE000C004 DLAB = 1 |

| **U0DLL** | **Функції** | **Опис** | **Скинуте значення** |
|---|---|---|---|
| 7:0 | Дільник регістра-защібки LSB | Дільник регістра-защібки LSB UART0 разом з регістром U0DLM визначає швидкість двійкової передачі | 0×01 |

**UART0_DLM**

| **Опис** | **Доступ** | **Скинуте значення** | **Адреса** |
|---|---|---|---|
| Дільник регістра-защібки MSB | R/W | 0 | 0×E000C004 |

Дільник регістра-защібки UART0 є частиною генератора-контролера швидкості передачі UART0 і зберігає значення, яке використовується для поділу тактів VPB так, щоб зробити швидкість двійкової передачі в тактах 16×бажану швидкість передачі. Регістри U0DLL і U0DLM утворюють 16 бітний дільник, де U0DLL містить нижні 8 біт, а U0DLM – верхні 8 біт дільника.

| **U0DLM** | **Функції** | **Опис** | **Скинуте значення** |
|---|---|---|---|

| U0DLM | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 7:0 | Дільник регістра-защібки MSB. | Дільник регістра-защібки MSB UART0 разом з регістром U0DLL визначає швидкість двійкової передачі | 0 |

**UART0_FCR**

| Опис | Доступ | Скинуте значення | Адреса |
|---|---|---|---|
| Регістр керування FIFO | R/W | 0 | 0×E000C008 |

U0FCR керує діями UART0 Rx і TX FIFO.

| U0FCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 0 | Розблокування FIFO | Дозволяє дії UART0 R×, T×FIFO та доступ до U0FCR 7:1. Цей біт має бути встановленим для задіяння UART0. Будь-який перехід цього біту автоматично очистить FIFO-стек UART0. | 0 |
| 1 | Скидання R× FIFO | Запис логічної 1 до U0FCR1 скине усі біти в UART0 R×FIFO, а також логічний покажчик. Цей біт самоскидаючийся. | 0 |
| 2 | Скидання T× FIFO | Запис логічної 1 до U0FCR2 скине усі біти в UART0 T×FIFO, а також логічний покажчик. Цей біт самоскидаючийся. | 0 |
| 5:3 | Зарезервовано | Програмні засоби користувача не повинні записувати одиниці в резервовані біти. Значення, прочитані звідси, не визначені | NA |
| 7:6 | R×Вибір рівня перекключення | 00: Рівень перемикання 0 (0×01h)<br>01: Рівень перемикання 1 (0×04h)<br>10: Рівень перемикання 2 (0×08h)<br>11: Рівень перемикання 3 (0×0eh)<br>Ці два біта визначають, скільки символів має бути записано до того, як відбудеться переривання. Чотири рівня перемикання, зазначені користувачем при компіляції, дозволяють йому відрегулювати обрану глибину FIFO. | 0 |

**UART0_LCR**

| Опис | Доступ | Скинуте значення | Адреса |
|---|---|---|---|
| Лінійний регістр керування | R/W | 0 | 0×E000C00C |

U0LCR визначає формат символів інформації, які передаються або приймаються.

| UOLCR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 1:0 | Вибір довжини слова | 00: довжина символу 5 біт<br>01: довжина символу 6 біт<br>10: довжина символу 7 біт<br>11: довжина символу 8 біт | 0 |
| 2 | Вибір стоп-біта | 0: 1 стоп-біт<br>0: 2 стоп-біта | 0 |
| 3 | Дозвіл по парності | 0: Блокування генерування і перевірки парності<br>1: Дозвіл генерування і перевірки парності | 0 |
| 5:4 | Вибір парності | 00: Перевірка на непарність<br>01: Перевірка на парність<br>10: Примусова "1" для біта парності<br>11: Примусовий "0" для біта парності | 0 |
| 6 | Керування перериванням | 0: Блокування переходу по перериванню<br>1: Дозвіл переходу по перериванню | 0 |
| 7 | Доступ дільника регістра-защібки | 0: Блокування доступу до дільника регістра-засувки<br>1: Дозвіл доступу до дільника регістра-засувки | 0 |

**UART0_LSR**

| Опис | Доступ | Скинуте значення | Адреса |
|---|---|---|---|
| Лінійний регістр стану пристрою | R0 | 0×60 | 0×E000C014 |

UOLSR – регістр тільки для читання, який забезпечує інформацію про стан блоків UART0 Rx та Tx.

| UOLSR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 0 | Приймач готових даних (RDR) | 0: U0RBR незайнятий<br>1: U0RBR містить правильні дані | 0 |
| 1 | Помилка переповнення (OE) | 0: Стан переповнення неактивний<br>1: Стан переповнення активний | 0 |
| 2 | Помилка парності (PE) | 0: Стан помилки парності неактивний<br>1: Стан помилки парності активний | 0 |
| 3 | Помилка кодування (FE) | 0: Стан помилки кодування неактивний<br>1: Стан помилки кодування активний | 0 |

| UO LSR | Функції | Опис | Скинуте значення |
|---|---|---|---|
| 4 | Зупинка переривання (BI) | 0: Стан зупинки переривання неактивний<br>1: Стан зупинки переривання активний | 0 |
| 5 | Перетворювач блокування незайнятого регістра (THRE) | 0: U0THR незайнятий<br>1: U0THR містить правильні дані | 1 |
| 6 | Перетворювач (TEMT) | 0: U0THR та/або U0TSR містить правильні дані<br>1: U0THR та U0TSR незайняті | 1 |
| 7 | Помилка в R× FIFO (R×FE) | 0: U0RBR не містить UART0 R× помилки або U0FCR0=0<br>1: UART0 RBR містить принаймні одну UART0 R×помилку | 0 |

## 7.2 Контрольні питання

1. Для чого призначений та які основні характеристики модуля UART в складі МК *ARM*7?
2. Дайте коротку характеристику регістрам модуля UART в складі МК *ARM*7 (номенклатура та призначення).
3. Яким чином можна використати модуль UART в складі МК *ARM*7 в рамках роботи із навчальним стендом LPC2148 Education Board?

## *7.3 Хід роботи*

### 7.3.1 Підготовчі стадії

1. Запустити проект 04_UART.
2. Зрозуміти логіку і послідовність роботи програми.
3. Запустити програму на стенді, видаливши перемички J5 і J7 (розділ 1.1).
4. Запустити програму HyperTerminal, налаштувати її на режим роботи 19200 8-N-1 без апаратного контролю.

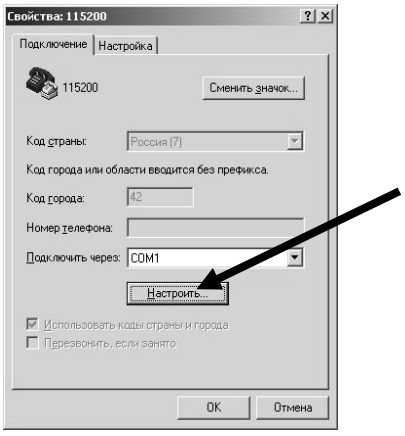



Рисунок 7.1 – Налаштування Hyper Terminal

Для цього вибрати пункт меню "Файл-Властивості" (рисунок 7.1). Вибрати порт, до якого підключений стенд, і натиснути кнопку «Налаштувати ...».

Вибрати наступні параметри:

Швидкість: 19200

Біти даних: 8

Парність: Немає

Стопові біти: 1

Управління потоком: Немає

*Зауваження*: Так налаштовано у функції ініціалізації UART. У разі зміни в коді програми необхідно провести відповідні зміни і в налаштуваннях Hyper Terminal.

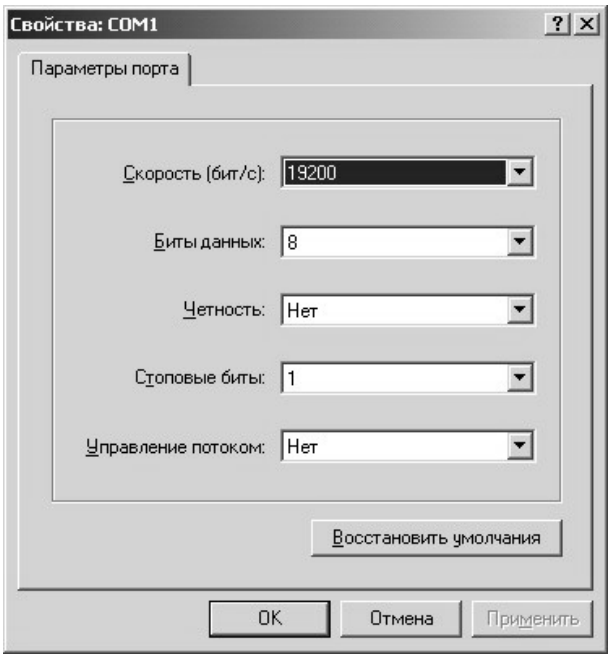



Рисунок 7.2 – Налаштування параметрів COM-порту

5. Переконатися в працездатності програми, натискаючи клавіші на клавіатурі (див. рисунок 7.3).

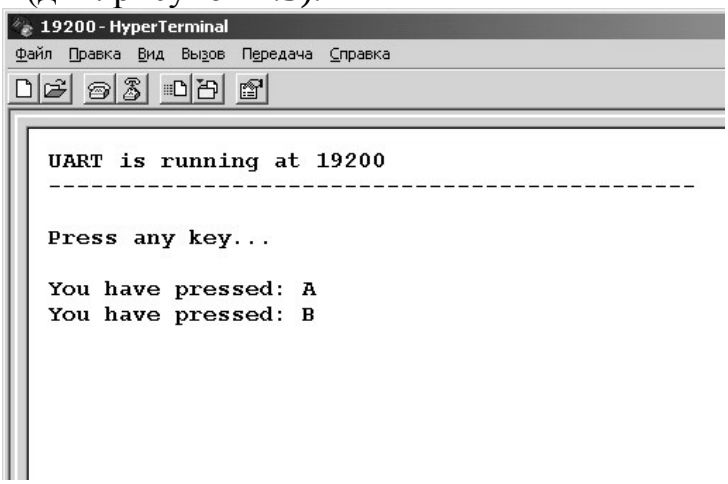



Рисунок 7.3 – Приклад роботи програми

### 7.3.2 Коментарі до програми

Основну програму наведено у лістингу 7.1.

main.c Лістинг 7.1.

```
01 int main (void)
02 {
03       char ch;
04       char text [128];
05
06       /*****************************************************/
07       /*           Write Hands On number to LCD            */
08       /*****************************************************/
09
10       InitLCD ();
11       SetBacklight (1);
12       LCDTextOut ("   Hands On 4   ", "      UART      ");
13
14       /*****************************************************/
15       /*                   Hands On code                   */
16       /*****************************************************/
17
18       InitUART (19200);
19       UARTTextOut ("\r\nUART is running at 19200\r\n------------\r\n");
20       UARTTextOut ("Press any key...\r\n\r\n");
21
22       while (1)
23       {
24             ch = UARTReadChar ();
25             sprintf(text, "You have pressed: %c (Code = %d)\r\n", ch, ch);
26             UARTTextOut (text);
27       }
28
29
30       return 0;
31 }
32
```

У рядку 18 UART ініціалізується на роботу на швидкості 19200 бод/с (докладніше – див. лістинг 7.2.).

Цикл в рядках 22-27 читає передані з терміналу комп’ютера дані та повертає код натиснутої клавіші.

main.c Лістинг 7.2.

```
01 void InitUART (int baud_rate)
02 {
03       /*****************************************************/
04       /*           Connect PIN connect block to UART       */
05       /*****************************************************/
06       PINSEL0 = 0x05;
07
08       /*****************************************************/
09       /*              Activate and clear FIFOs             */
10       /*****************************************************/
11       U0FCR = 0x07;
12
13       /*****************************************************/
14       /*    Set 8N1 mode, allow access to divider-latches  */
15       /*****************************************************/
16       U0LCR = 0x83;
17
18       /*****************************************************/
19       /*                 Calculate baud rate               */
20       /*****************************************************/
21       U0DLL = (12000000 * 5) / (baud_rate * 16 * 4);
22
23       /*****************************************************/
```

```
24        /*    Set 8N1 mode, forbid access to divider-latches      */
25        /*********************************************************/
26        U0DLM = 0x00;
27        U0LCR = 0x03;
28  }
29
30
31  void UARTPutchar(int buffer)
32  {
33        /*********************************************************/
34        /*              Write char to Tx register                */
35        /*********************************************************/
36        U0THR = buffer;
37
38        /*********************************************************/
39        /*            Wait until the char is sent                */
40        /*********************************************************/
41        while(!(U0LSR & 0x40));
42  }
43
44  void UARTTextOut (char * string)
45  {
46        int i, len = strlen (string);
47
48        for (i = 0; i < len; i++)
49              UARTPutchar (string [i]);
50  }
51
52  char UARTReadChar (void)
53  {
54        /*********************************************************/
55        /*                 Wait char from terminal               */
56        /*********************************************************/
57        while(!( U0LSR & 0x1 )) {}
58        return U0RBR;               //Read character from input buffer
59  }
```

Функція ініціалізації (InitUART) підключає виводи МК до модуля UART (рядок 6), активує та очищує черги на прийом і передачу (рядок 11), встановлює режим передачі і надає їм доступ до делителя-защібки (рядок 16), розраховує необхідне значення U0DLL для роботи на заданій швидкості (рядок 21) та забороняє доступ до дільника-защібки.

Формула, наведена в рядку 21, стандартна для всіх контролерів сімейства LPC2xxx:

U0DLL = (CPU_Freq) / (baud_rate * 16 * 4),
где CPU_Freq = частота кварцу * множник PLL,
baud_rate – необхідна швидкість роботи, бод/с.

Функція UARTPutChar пересилає один символ по інтерфейсу UART. В рядку 36 вона записує необхідний символ у відповідний регістр, в рядку 41 очікує відправки даного символу.

Функція UARTReadChar очікує на появу символу у вхідному буфері, отриманого з терміналу комп'ютера (або будь-якого іншого пристрою), читає цей символ і повертає підпрограмі, що викликала (рядки 57, 58).

### *7.4 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні питання.

3. Схему програми вирішеної задачі зі схемою підключення UART.

4. Текст програми з докладними коментарями, які відповідають схемі програми.

5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) дозволяє виконувати UART?

# 8 Лабораторна робота №8. Дослідження послідовного інтерфейсу SPI

***Мета роботи***: практичне засвоєння методів роботи з послідовним інтерфейсом передачі даних SPI

## *8.1 Короткі теоретичні відомості*

SPI – інтерфейс передачі даних, що підтримує головні "master" і підлеглі "slave" пристрої на шині. У мікроконтролерах сімейства LPC2xxx інтерфейс SPI може працювати як головним, так і підпорядкованим пристроєм. Інтерфейс SPI містить 4 сигнали: SPI_SCK, SPI_SSEL – сигнали синхронізації, SPI_MISO (master input - slave output), SPI_MOSI (master output - slave input) – канали передачі даних. Часові діаграми циклу передачі даних в різних режимах наведені на рисунку 8.1.

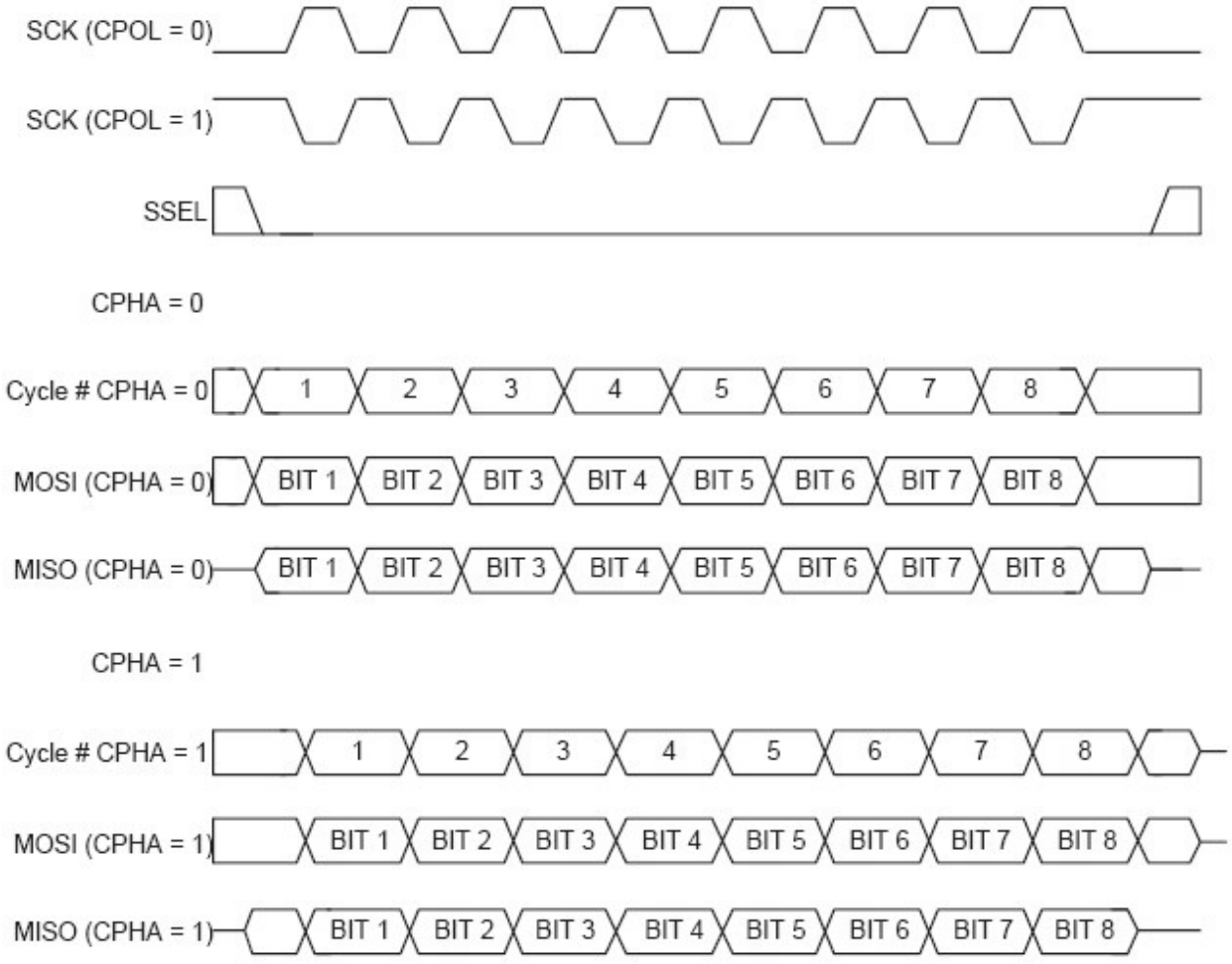



Рисунок 8.1 – Часові діаграми циклу передачі даних

На платі навчального стенду *LPC2148 Education Board* дані інтерфейсу SPI передаються на два послідовно з'єднаних регістра, керуючих

світлодіодною матрицею (рисунок 1.10). Перший регістр вибирає стовпці, подаючи сигнал 1 на відповідні виходи, другий дозволяє протікати струму через світлодіоди, якщо виставити сигнал 0 на відповідних виходах.

Переміщення точки на матриці, що світиться, здійснюється за допомогою джойстика, схема підключення якого наведена на рисунку 1.4.

### 8.1.1 Опис регістрів SPI

| **Регістр** | **Опис** |
|---|---|
| S0SPCR | Регістр керування SPI |
| S0SPSR | Регістр стану |
| S0SPDR | Двонапрялений регістр даних |
| S0SPCCR | Лічильник тактових імпульсів |
| S0SPINT | Регістр переривання |

Опис регістра керування

| **Біти** | **Опис** |
|---|---|
| 1:0 | Зарезервовано |
| 2 | 0 – усі передачі по 8 біт; 1 – по стільки біт, скільки вказано в бітах 11:8 |
| 3 | CPHA. Визначає моменти видачі даних. Див. рисунок 7.1 для пояснення |
| 4 | CPOL. Полярність. Див. рисунок 7.1 для пояснення |
| 5 | 1 – режим “master”, 0 – режим “slave”. |
| 6 | 0 – дані передаються в режимі Big Endian; 1 – в режимі Little Endian |
| 7 | Режим переривання |
| 11:8 | Кількість біт за одну передачу при біті 2, рівному 0.<br>1000 – 8 біт<br>1001 – 9 біт<br>…<br>1111 – 15 біт<br>0000 – 16 біт |
| 15:12 | Зарезервовано |

## 8.2 Контрольні питання

1. Для чого призначений та які основні характеристики модуля SPI в складі МК *ARM7*?
2. Дайте коротку характеристику регістрам модуля SPI в складі МК *ARM7* (номенклатура та призначення).
3. Яким чином можна використати модуль SPI в складі МК *ARM7* в рамках роботи із навчальним стендом LPC2148 Education Board?

## *8.3 Хід роботи*

### 8.3.1 Підготовчі стадії

1. Запустити проект 05_SPI.
2. Зрозуміти логіку і послідовність роботи програми.
3. Запустити програму на платі і спостерігати результати її роботи (управляти роботою інтерфейсу SPI можна за допомогою джойстика, що знаходиться на платі).

   Код, що описує логіку роботи програми, приведений в лістингу 8.1.

main.c Лістинг 8.1

```
01 int main (void)
02 {
03       int X = 4, Y = 4;
04
05       /*************************************************/
06       /*           Write Hands On number to LCD        */
07       /*************************************************/
08       InitLCD ();
09       SetBacklight (1);
10       LCDTextOut ("   Hands On 5   ", "    SPI Usage   ");
11
12       /*************************************************/
13       /*                  Hands On code                */
14       /*************************************************/
15
16       InitSPI ();       // see "spi.c"
17
18       while (1)
19       {
20             switch ((IOPIN0 & (0x1F << 16)) >> 16) // Read joystick
21             {
22                   case 0x1D:  if (Y < 8) Y++; break;  // Up
23                   case 0x0F:  if (Y > 1) Y--; break;  // Down
24                   case 0x17:  if (X > 1) X--; break;  // Left
25                   case 0x1B:  if (X < 8) X++; break;  // Right
26                   default: break;
27             }
28
29             SPIPutDot (X, Y);       // put dot to LED matrix, see "spi.c"
30             Sleep (300);
31       }
32
33
34       return 0;
35 }
```

У рядку 16 ініціалізується модуль SPI (див. Лістинг 8.2).

У рядку 20 відбувається опитування джойстика. Відповідно до його положення, вибираються нові координати точки на світлодіодній матриці і передаються в модуль SPI (рядок 29, див. Лістинг 8.2).

spi.c Лістинг 8.2

```
01 void InitSPI (void)
02 {
03       /*************************************************/
04       /*                  Pin Select to use SPI        */
05       /*************************************************/
06       PINSEL0 |= (1 << 8) | (1 << 10) | (1 << 12);
07
08       /*************************************************/
```

```
09        /*                        Init SPI                          */
10        /************************************************************/
11        S0SPCR =    (0 << 2) |  // set bits per transfer (0-8bit, 1-defined)
12                    (1 << 3) |  // CPHA mode
13                    (1 << 4) |  // SCK is active high
14                    (1 << 5) |  // Set SPI as master
15                    (0 << 8);   // 16 bits per transfer
16
17        S0SPCCR = 64;  // Set SPI speed
18  }
19
20
21  void SPIPutDot (int x, int y)
22  {
23        IOSET0 = (1 << 15);                 // Pull up P0.15
24
25        /************************************************************/
26        /*          Send value for first shift register    */
27        /************************************************************/
28        S0SPDR = ~(1 << (y - 1));
29        while ((S0SPSR & (1 << 7)) == 0);   // Wait until data is sent
30
31        /************************************************************/
32        /*          Send value for second shift register   */
33        /************************************************************/
34        S0SPDR = ~(1 << (8 - x));
35        while ((S0SPSR & (1 << 7)) == 0);   // Wait until data is sent
36
37        IOCLR0 = (1 << 15);     // Pull down P0.15 ("apply new settings")
38  }
```

Функція ініціалізації модуля SPI (InitSPI) підключає виводи мікроконтролера до SPI (рядок 6), налаштовує режим роботи модуля (рядки 11-15) і вибирає дільник швидкості (рядок 17).

Функція малювання точки (SPIPutDot) дозволяє роботу регістрів зсуву (рядок 23), відправляє на них значення (рядки 28 і 34) і чекає відправки цих значень (рядки 29 і 35). Рядок 37 забороняє роботу зсувних регістрів.

### *8.4 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.

2. Письмові відповіді на контрольні питання.

3. Схему програми вирішеної задачі зі схемою підключення індикатора.

4. Текст програми з докладними коментарями, які відповідають схемі програми.

5. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) дозволяє виконувати SPI?

### *8.5 Індивідуальне завдання до виконання*

Реалізувати задану викладачем інформаційну модель на світлодіодній матриці 8x8. Приклади завдань знаходяться в Додатку Г (Табл. Г.3).

# 9 Лабораторна робота №9. Дослідження послідовного інтерфейсу USB

***Мета роботи***: практичне засвоєння методів роботи з USB контролером

## *9.1 Короткі теоретичні відомості*

USB (Universal Serial Bus) – це широко поширений 4-провідний інтерфейс, що підтримує зв'язок між USB-хостом і великою кількістю периферійних пристроїв (максимально – 127). Модуль USB на мікроконтролері LPC2148 є клієнтським ("slave"-контролером) і не містить USB-хоста. Це дозволяє конструювати такі пристрої, як USB-миші, клавіатури, звукові карти, накопичувачі інформації та багато інших. При цьому даний мікроконтролер підтримує стандарт USB 2.0 Full Speed і може працювати з максимальною швидкістю 12 Мб/с.

Модуль USB мікроконтролерів LPC2xxx має 8 Кб пам'яті, виділеної для прямого доступу до пам'яті (DMA). Крім того, це єдині мікроконтролери зі своєї категорії, які підтримують 32 кінцеві точки, що дозволяє реалізувати практично будь-який USB пристрій. Структура контролера USB наведена на рисунку 9.1.

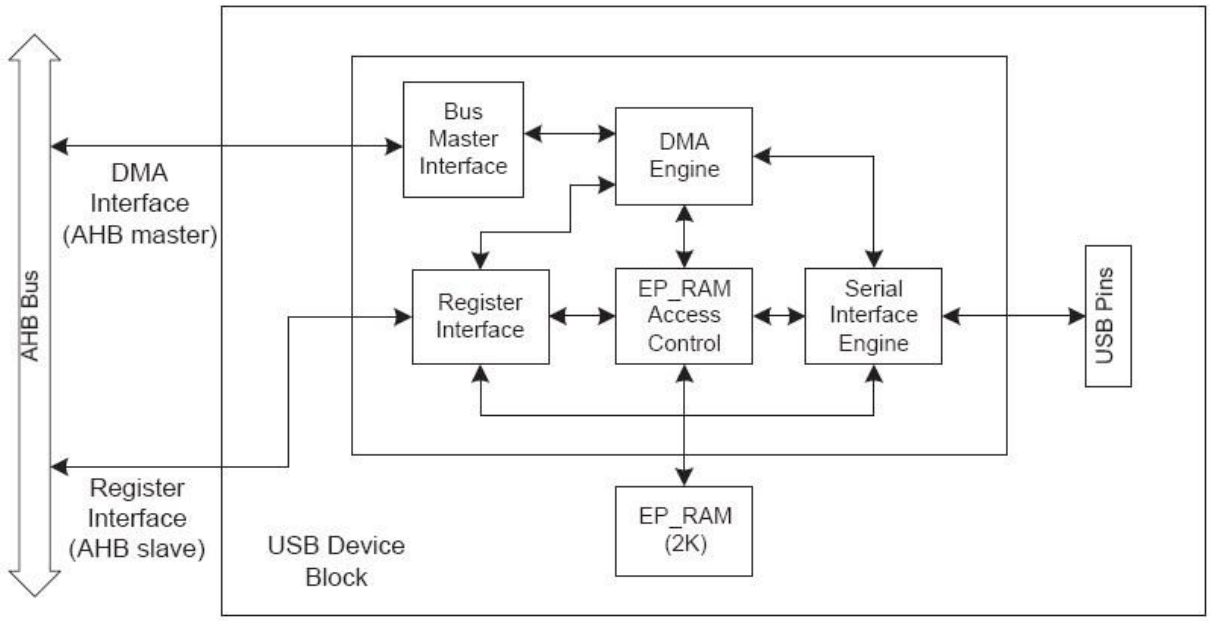



Рисунок 9.1 – Структура контролера USB.

Створення програмного забезпечення для USB пристроїв – непросте завдання, яке вимагає глибокого розуміння інтерфейсу USB. За необхідності, можна самостійно вивчити специфікацію USB та порядок роботи програми.

## *9.2 Хід роботи*

### 9.2.1 Підготовчі стадії

1. Відкрити проект 06_USB.
2. Скомпілювати і запустити програму на платі навчального стенду *LPC2148 Education Board*.
3. При живленні не від USB-порту, з'єднати USB кабелем плату навчального стенду *LPC2148 Education Board* і комп'ютер.
4. Почекати, поки в комп'ютері не виявиться новий пристрій. Як правило, в операційних системах сімейства Windows, починаючи з Win2000, немає необхідності встановлювати додаткові драйвери.
5. Вивчити диск, що з'явився. Спробувати прочитати і записати на нього інформацію.

## *9.3 Вимоги до звіту по роботі*

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.

2. Схему програми вирішеної задачі зі схемою підключення індикатора.

3. Текст програми з докладними коментарями, які відповідають схемі програми.

4. Висновки по роботі, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно (перерахувати по пунктах) дозволяє виконувати модуль USB, вбудований в мікроконтролери LPC2xxx?

## *9.4 Індивідуальне завдання до виконання*

Реалізувати заданий викладачем обмін з хостом.

## Рекомендована література

1. LPC2148 Education Board. QuickStart Program Development User's Guide. – Embedded Artists AB, 2005 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\Магістри\Мікропроцесорна техніка _ARM\QuickStart_Program_Development_Users_Guide-Version_1.0_Rev_A.pdf

2. LPC2148 Education Board. User's Guide. – Embedded Artists AB, 2007 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\Магістри\Мікропроцесорна техніка _ARM \LPC2148_Education_Board_Users_Guide-Version_2.1_Rev_B.pdf

3. Мартин Т. Микроконтроллеры ARM7. Семейство LPC2000 компании Philps. Вводный курс/ Пер. с англ. – М.: Издательский дом «Додэка-XXI», 2006. – 240с.

4. Редькин П.П. Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя. – М.: Издательский дом «Додэка-XXI», 2007. – 560с.

5. LPC214x User Manual. Document UM10139. – Philips, 2005 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive \Kources \Магістри \Мікропроцесорна техніка _ARM \lpc2148_ usermanual.pdf

6. ARM7TDMI-S. Technical Reference Manual. Revision: r4p3. – ARM Limited, 2001 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\Магістри\Мікропроцесорна техніка _ARM \ARM7TDMI-S.pdf

7. ARM 7TDMI Data Sheet. Document Number:ARM DDI 0029E. – Advanced RISC Machines Ltd (ARM), 1995 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources \Магістри\Мікропроцесорна техніка _ARM \ ARM 7TDMI_Data_sheet.pdf

8. LM75. Digital Temperature Sensor and Thermal Watchdog with Two-Wire Interface. – National Semiconductor Corporation TL/H/12658, 1996 [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\\Магістри\Мікропроцесорна техніка _ARM \\LM75.pdf

9. CAT1024, CAT1025. Supervisory Circuits with I2C Serial 2k-bit CMOS EEPROM and Manual Reset. – Catalyst Semiconductor, Inc., 2004. [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\Магістри\Мікропроцесорна техніка _ARM \cat1025.pdf

10. Тестова програма стенду *LPC2148 Education Board* [Електронний ресурс]. – Режим доступу (Інтранет): \\Inel\archive\Kources\Магістри\Мікропроцесорна техніка _ARM \ARM_Software \testprogram_lpc2148_edu_v2_1.hex

11. MDK Microcontroller Development Kit [Електронний ресурс]. – Режим доступу : http://www2.keil.com/mdk5/

12. MDK v4 Legacy Support [Електронний ресурс]. – Режим доступу : http://www2.keil.com/mdk5/legacy/
13. NXP (founded by Philips) LPC2148 [Електронний ресурс]. – Режим доступу : http://www.keil.com/dd/chip/3880.htm
14. MDK Microcontroller Development Kit [Електронний ресурс]. – Режим доступу : http://www2.keil.com/mdk5
15. Специфікація інтерфейсу USB 2.0.
16. Програмування мікроконтролерних систем на базі ядра ARM7 на мові С. Методичні вказівки до виконання лабораторних робіт з дисципліни "Мікроконтролерні системи автоматики та управління" для студентів напряму підготовки 6.050102 – "Комп'ютерна інженерія"./ Укл. А.І. Роговенко, О.В. Красножон, А.В. Красножон – Чернігів: ЧДТУ, 2014. – 51 с.

# Додатки

## *Додаток А – Програми керування периферійними пристроями*

**Програма А.1 – Засвічування лінійки світлодіодів**

Схема підключення наведена на рисунку 1.2.

```
#include <lpc21xx.h>
int main(void) {
     IODIR0 |= 0x0000FF00;
     IOCLR0 |= 0x0000FF00;
     while(1) {;}
     }
```

**Програма А.2 – Опитування стану кнопки**

При кожному натисканні кнопки, підключеної відповідно до рисунку 1.3, вмикається, а потім вимикаються світлодіоди.

```
#include <lpc21xx.h>
void delay(int milisecs) {
     unsigned int i;
     for (i = 0; i < milisecs * 10000/6; ++i) {};
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     while(1) {
          if (!(IOPIN0 & (1<<14))) {
               IOCLR0 |= 0x0000FF00;
               delay(80);
               IOSET0 |= 0x0000FF00;
          }
     }
}
```

*Примітка*. При роботі з кнопкою слід пам'ятати про брязкіт контактів і застосовувати методи його пригнічення (затримку 1…100 мс, повторне опитування, лічильник повторних підтверджень тощо).

**Програма А.3 – Опитування стану джойстика**

Кнопковий контакт джойстика (рисунок 1.4) може знаходитись в одному з п'яти положень: зсунутий ліворуч, праворуч, вгору, донизу або натиснутий так само, як кнопка. Джойстик, як елемент людино-машинного інтерфейсу, можна використовувати для керування іншими пристроями. У прикладі за умови кожного натискання на джойстик виконується певна дія.

```
#include <lpc21xx.h>
int main(void) {
     IODIR0 |= 0x0000FF00;
```

```
     while(1) {
          switch ((IOPIN0 & (0x1F<<16)) >> 16) {
               case 0x1E: <дія «Push»> break;
               case 0x1D: <дія «↓»> break;
               case 0x1B: <дія «↑»> break;
               case 0x17: <дія «←»> break;
               case 0x0F: <дія «→»> break;
               default: break;
          }
     }
     }
```

**Програма А.4 – Керування трьохкольорним світлодіодом**

Трьохкольорний світлодіод (рисунок 1.5) являє собою 3 напівпровідникових світлодіоди з різним кольором світіння (R-червоний, G-зелений, B-голубий), які розміщено в одному корпусі. Засвічуванням та гасінням кожного з цих світлодіодів можна керувати окремо, що дозволяє генерувати різні кольорові комбінації. У прикладі відбувається почергове роздільне засвічування кожного з кольорів.

```
#include <lpc21xx.h>
...
int main(void) {
     IODIR0 |= 0x0000380;
     IOCLR0 |= 0x0000380;
     while(1) {
          IOSET0 |= (1<<7);
          delay (100);
          IOCLR0 |= 0x0000380;
          IOSET0 |= (1<<8);
          delay (100);
          IOCLR0 |= 0x0000380;
          IOSET0 |= (1<<9);
          delay (100);
          IOCLR0 |= 0x0000380;
     }
}
```

**Програма А.5 – Керування зумером**

Частота звуку, що генерується зумером (рисунок 1.6), буде тим вище, чим більшу кількість разів за період переключатиметься транзисторний ключ.

```
#include <lpc21xx.h>
…
int main(void) {
     IODIR0 |= (1<<7);
     IOSET0 |= (1<<7);
     while(1) {
```

```
          IOCLR0 |= (1<<7);
          delay(3);
          IOSET0 |= (1<<7);
          delay(2);
     }
     }
```

**Програма А.6 – Керування кроковим двигуном**

Двигун (рисунок 1.11) обертається за годинниковою стрілкою:

```
#include <lpc21xx.h>
…
int main(void) {
     unsigned char flag = 0;
     IODIR0 |= (1 << 12) | (1 << 21);
     IOCLR0 |= (1 << 12) | (1 << 21);
     while(1) {
               IOSET0 |= (1<<12);
               delay (12);
               IOSET0 |= (1<<21);
               delay (12);
               IOCLR0 |= (1<<12);
               delay (12);
               IOCLR0 |= (1<<21);
               delay (12);
     }
     }
```

**Програма А.7 – Відправка команди до РКІ-модуля**

```
void SetCommLCD(uchar  bT) {
     IODIR1 |= (0xFF << 16); //Set data bus as outputs
     IODIR1 |= (1 << 24); //Set RS as output
     IODIR1 |= (1 << 25); //Set E as output
     IODIR0 |= (1 << 30); //Set Backlight control as output
     IODIR0 |= (1 << 22); //Set R/W as output
     IOCLR1 = (1 << 25); //E down
     IOCLR1 = (1 << 24); //RS = 0
     IOCLR0 = (1 << 22); //RW = 0
     IOCLR1 = (0xFF) << 16; //Clear data bus
     IOSET1 = (bT) << 16; //Write data to bus
     IOSET1 = (1 << 25); //E up
     Sleep (10); //Wait
     IOCLR1 = (1 << 25); //E down
}
```

**Програма А.8 – Відправка даних (коду символа) до РКІ-модуля**

```
void SetDataLCD(uchar  bT) {
     IODIR1 |= (0xFF << 16); //Set data bus as outputs
     IODIR1 |= (1 << 24); //Set RS as output
     IODIR1 |= (1 << 25); //Set E as output
     IODIR0 |= (1 << 30); //Set Backlight control as output
     IODIR0 |= (1 << 22); //Set R/W as output
     IOCLR1 = (1 << 25); //E down
     IOSET1 = (1 << 24); //RS = 1
     IOCLR0 = (1 << 22); //RW = 0
     IOCLR1 = (0xFF) << 16; //Clear data bus
     IOSET1 = (bT) << 16; //Write data to bus
     IOSET1 = (1 << 25); //E up
     Sleep (10); //Wait
     IOCLR1 = (1 << 25); //E down
     }
```

**Програма А.9 – Робота із матричним індикатором**

Два послідовно-паралельні регістри керування знакосинтезуючим індикатором (рисунок 1.10) обмінюються інформацією з мікроконтролером за допомогою інтерфейсу SPI. У стенді реалізовано каскадне (послідовне) підключення ведених пристроїв інтерфейсу SPI. У прикладі функції в деякому рядку та стовпчику індикатора засвічується світлодіод.

```
void SPIPutDot (int x, int y) {
     IOSET0 = (1 << 15);
     S0SPDR = ~(1 << (y - 1));
     while ((S0SPSR & (1 << 7)) == 0);
     S0SPDR = ~(1 << (8 - x));
     while ((S0SPSR & (1 << 7)) == 0);
     IOCLR0 = (1 << 15);
}
```

**Програма А.10 – Функція для ініціалізації АЦП**

```
void InitADC (void) {
     PINSEL1 |= (1 << 24);
          /*Write 01 to 25:24 bits of PINSEL1 to select the
          ADC-input function for P0.28 - AD0.1*/
     PINSEL1 |= (1 << 26);
          /*Write 01 to 27:26 bits of PINSEL1 to select the
          ADC-input function for P0.29 - AD0.*/
     AD0CR &= 0x00000000; //Reset ADC0
     //Setup ADC – AD0.1 and AD0.2 input, 11 clocks/10 bits
     AD0CR |= (1 << 2); //Select ADC channel
     AD0CR |= (1 << 16); //Automatically repeat transform
     AD0CR |= (1 << 21); //Power-on ADC
}
```

**Програма А.11 – Приклад функції, яка читає дані з АЦП після закінчення перетворення**

```
int ADCReadValue (void) {
     int ADCValue= 0;
     AD0CR = 0x00210002; //Setup ADC AD0.1 input (P0.28)
     AD0CR |= 0x01000000; //Start of transformation
     //Waiting for the end of transformation
     while (AD0GDR == 0x8000000);
     ADCValue = AD0DR1; //Read all data from ADC
     //Read 10-bits result of transformation
     ADCValue = ((ADCValue >>6) & 0x03FF);
     return ADCValue;
     }
```

## *Додаток Б – Приклади програм обробки переривань*

**Програма Б.1 – Конфігурування й обробка переривання за співпадінням для таймера TIMER0**

```
#include <lpc21xx.h>

int led_count = 8;
//Обробщик переривання від Timer0
__irq void Timer0ISR(void) {
     if (++led_count == 16)
          led_count = 8;
     IOSET0 |= 0x0000FF00;
     T0IR = 0x01;
     VICVectAddr = 0;
     return;
}
}

void InitTimer0(void) {
     /*****************************************************/
     /*              Ініціалізація VIC                    */
     /*****************************************************/
     VICDefVectAddr = (unsigned int) &Timer0ISR;
     VICIntEnable = 0x10; //Channel#4 is the Timer0
     VICIntSelect = 0x00; //all interrupts are IRQs
     /*****************************************************/
     /*          Ініціалізація Timer0                     */
     /*****************************************************/
     T0MR0 = 4500000; //Timer match (~ 0.1 second)
     T0MCR = 0x03; //Interrupt on Match0, reset timer on match
     T0PC = 0x01; // Prescaler to 2
     T0TC = 0x00; // reset Timer counter
     T0TCR = 0x01; // enable Timer
     return;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     InitTimer0();
     while(1) {
          IOCLR0 |= (1 << led_count);
     }
     }
```

**Програма Б.2 – Конфігурування й обробка зовнішнього переривання EINT1, яке генерується за натисканням кнопки SW1**

```
#include <lpc21xx.h>

unsigned char flag = 0;

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

__irq void eint1(void) {
     EXTINT |= 0x02;
     VICVectAddr = 0;
     flag = 1;
}

void init_eint1(void) {
     PINSEL0 |= 0x20000000;
     VICVectCntl0 = 0x0000002f;
     VICVectAddr0 = (unsigned)&eint1;
     VICIntEnable = 0x00008000;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     init_eint1();
     while(1) {
          if (flag == 1) {
               IOCLR0 |= 0x0000FF00;
               delay(200);
               IOSET0 |= 0x0000FF00;
               flag = 0;
          }
     }
     }
```

## *Додаток В – Система команд ARM*

| **Мнемоніка** | **Назва команди** | **Дія** |
|---|---|---|
| ADC | Додавання з перенесенням | Rd := Rn + Op2 + Carry |
| ADD | Додавання | Rd := Rn + Op2 |
| AND | Логічне «І» | Rd := Rn AND Op2 |
| B | Перехід | R15 := address |
| BIC | Очищення біта | Rd := Rn AND NOT Op2 |
| BL | Перехід із збереженням адреси повернення | R14 := R15, R15 := address |
| BX | Перехід і обмін | R15 := Rn, T bit := Rn[0] |
| CDP | Обробка даних співпроцесором | (Coprocessor-specific) |
| CMN | Негативне порівняння | CPSR flags := Rn + Op2 |
| CMP | Порівняння | CPSR flags := Rn - Op2 |
| EOR | Виключне «АБО» | Rd := (Rn AND NOT Op2) OR (op2 AND NOT Rn) |
| LDC | Запис співпроцесора з пам'яті | Coprocessor load |
| LDM | Запис в регістри | |
| LDR | Завантаження регістру з пам'яті | Stack manipulation (Pop) |
| MCR | Копіювання регістра центрального процесора в регістр співпроцесора | |
| MLA | Множення з переповненням | Rd := (Rm * Rs) + Rn |
| MOV | Переміщення регістра або константи | Rd : = Op2 |
| MRC | Переміщення з регістра співпроцесора до регістру центрального процесора | Rn := cRn {<op>cRm} |
| MRS | Переміщення PSR статусу/прапорців до регістру | Rn := PSR |
| MSR | Переміщення регістра в PSR статусу/прапорців | PSR := Rm |
| MUL | Множення | Rd := Rm * Rs |
| MVN | Переміщення від’ємного регістра | Rd := 0xFFFFFFFF EOR Op2 |
| ORR | Логічне «АБО» | Rd := Rn OR Op2 |
| RSB | Зворотне віднімання | Rd := Op2 - Rn |
| RSC | Зворотне віднімання з перенесенням | Rd := Op2 - Rn - 1 + Carry |
| SBC | Віднімання з переносом | Rd := Rn - Op2 - 1 + Carry |
| STC | Збереження регістра співпроцесора в пам'ять | address := CRn |
| STM | Багатократне збереження | Stack manipulation (Push) |
| STR | Збереження регістра в пам'яті | <address> := Rd |

| **Мнемоніка** | **Назва команди** | **Дія** |
|---|---|---|
| SUB | Віднімання | Rd := Rn - Op2 |
| SWI | Програмне переривання | OS call |
| SWP | Обмін регістра з пам'яттю | Rd := [Rn], [Rn] := Rm |
| TEQ | Перевірка на порозрядну рівність | CPSR flags := Rn EOR Op2 |
| TST | Перевірка бітів | CPSR flags := Rn AND Op2 |

## *Додаток Г – Приклади програм керування периферійними пристроями*

В цьому додатку наведені комплексні завдання для складання програм керування периферійними пристроями на стенді. Їх можна використовувати для реалізації курсових і розрахунково-графічних робіт, а також бонусних лабораторних робіт відповідно до графіку навчального процесу у конкретному семестрі. Крім того, робота над цими завданнями дозволяє отримати практичний досвід та навички, які в подальшому можна використати у випускових кваліфікаційних роботах.

Таблиця Г.1 – Завдання із керування за допомогою джойстика

| **Варіант** | **Опис завдання** |
|---|---|
| 1 | Реалізувати алгоритм управління світлодіодним індикатором за допомогою джойстика SW2. При натисканні джойстика вліво або вправо деякий світлодіод переміщається в обраному напрямку, при натисканні вгору або вниз – залишається на колишній позиції. Переміщення світлодіоду відбувається по спрацьовуванню переривання від таймера (частота вибирається довільно). |
| 2 | Реалізувати алгоритм відображення на матричному світлодіодному індикаторі кількості оброблених зовнішніх переривань, які генеруються при натисканні на кнопку SW1. Брязкіт контактів повинен усуватися програмним способом. |
| 3 | Реалізувати алгоритм плавного засвічування і затухання матричного світлодіодного індикатора. |
| 4 | Реалізувати алгоритм світіння матричного світлодіодного індикатора з різними рівнями яскравості. Сусідні світлодіоди повинні відрізнятися за яскравістю на 1 крок градації – 1/8 від максимального значення, тобто перший світлодіод світиться з яскравістю 1/8, другий – 2/8 і т. д. Останній світлодіод - з максимальною яскравістю. |
| 5 | Реалізувати алгоритм управління частотою звуку за допомогою джойстика. Крок зміни частоти 100 Гц. Одне натискання джойстика має відповідати одному кроку зміни частоти в залежності від напрямку натискання: вгору – частота збільшується, вниз – зменшується. |

Таблиця Г.2 – Завдання із керування рідкокристалічним індикатором

| **Варіант** | **Опис завдання** |
|---|---|
| 1 | Реалізувати алгоритм виведення даних на РКІ. Дані у вигляді цілих чисел від 0 до 9 циклічно надходять і виводяться в довільному місці індикатора при спрацьовуванні переривання від таймера TIMER0, частота роботи якого складає 2 Гц |
| 2 | Реалізувати алгоритм виведення даних на РКІ. Дані у вигляді власноруч створених символів (не менше 2-х символів) повинні виводитися в довільні знакомісця індикатора. |
| 3 | Реалізувати алгоритм відображення стану електронного секундоміра (у форматі мм:сс) на РКІ, використовуючи 4-х бітний інтерфейс обміну. При цьому на РКІ повинен відображатися поточний хід хвилин і секунд. Управління роботою секундоміра проводитися за допомогою натискань джойстика SW2: вправо – запуск секундоміра, вліво – зупинка, вниз - зупинка і скидання. |
| 4 | Реалізувати алгоритм керування курсором на РКІ за допомогою джойстика. Роль курсора відіграє власноруч створений символ. Курсор переміщується тільки в межах видимої області РКІ. |
| 5 | Реалізувати алгоритм виведення біжучого рядка, який складається з прізвищ студентів бригади, на РКІ. Дані рухаються по індикатору в напрямку, який задається джойстиком. Зсув джойстика вгору прискорює біжучий рядок на 25%, а вниз – уповільнює. |

Таблиця Г.3 – Завдання із керування интерфейсом SPI

| Варіант | Опис завдання |
|---|---|
| 1 | Реалізувати алгоритм світіння / гасіння на знакосинтезуючому індикаторі "шахівниці" при спрацьовуванні зовнішнього переривання EINT1, яке генерується після натискання кнопки SW1. Брязкіт контактів повинен усуватися програмним способом. |
| 2 | Реалізувати алгоритм виведення на знакосинтезуючий індикатор послідовності цифр від 0 до 9. Цифри починають виводитися при спрацьовуванні зовнішнього переривання EINT1, яке генерується після натискання кнопки SW1. Брязкіт контактів повинен усуватися програмним способом. Повторне натискання на SW1 призупиняє процес виведення, при цьому індикатор зберігає поточний стан. |
| 3 | Реалізувати алгоритм управління світлодіодом на знакосинтезуючому індикаторі за допомогою джойстика SW2. Спочатку в довільному місці індикатора загоряється один світлодіод. Після кожного натискання джойстика з затримкою 0,2 с запалений світлодіод зміщується в заданому напрямку. |
| 4 | Реалізувати алгоритм переміщення вогню на знакосинтезуючому індикаторі. Спочатку в довільному місці індикатора загоряється світлодіод. Надалі він рухається (частота 2 Гц) по індикатору під кутом 45 °. При досягненні краю "м'ячик" повинен відбиватися в протилежну сторону під кутом 45°. Переміщення світлодіоду призупиняється після натискання кнопки SW1. При цьому виводиться кількість "ударів" об край знакосинтезуючого індикатора на семисегментний світлодіодний індикатор. |
| 5 | Реалізувати алгоритм відображення шкали на знакосинтезуючому індикаторі. Залежно від кількості натискань джойстика SW2 загоряється певна кількість рядків індикатора. Кожне натискання джойстика вгору додає по одному рядку, що світиться, вниз – зменшує. При досягненні граничних станів (вся шкала заповнена, або вся шкала порожня) відповідні натискання джойстика вгору або вниз не здійснюють впливу. |

Таблиця Г.4 – Завдання із керування АЦП

| Варіант | Опис завдання |
|---|---|
| 1 | Реалізувати алгоритм читання даних з АЦП і виведення їх на знакосинтезуючий індикатор у вигляді шкали. Залежно від діапазону, в якому знаходиться поточне значення вхідної напруги, загоряється певна кількість рядків індикатора. |
| 2 | Реалізувати алгоритм читання даних з АЦП і виведення їх на РКІ в форматі: "Voltage: *. * V". При збільшенні або зменшенні вхідної напруги за допомогою потенціометра значення, що відображається на РКІ, змінюється. |
| 3 | Реалізувати алгоритм читання даних з АЦП і виведення їх на світлодіодний індикатор у вигляді шкали. Залежно від діапазону, в якому знаходиться поточне значення вхідної напруги, загоряється певна кількість світлодіодів. |
| 4 | Реалізувати алгоритм читання даних з АЦП і виведення їх на синтезатор тону. Залежно від діапазону, в якому знаходиться поточне значення вхідної напруги, синтезатор генерує звук певної частоти. Якщо вхідна напруга лежить в діапазоні 0 В – 1 В, генерується звук з частотою 100 Гц; якщо в діапазоні 1 В – 2 В, генерується звук з частотою 300 Гц; якщо в діапазоні 2 В – 3 В, генерується звук з частотою 500 Гц |
| 5 | Реалізувати алгоритм читання даних з АЦП і виведення їх на світлодіодний індикатор. Залежно від діапазону, в якому знаходиться поточне значення вхідної напруги, всі світлодіоди індикатора блимають з певною частотою. Крок зміни частоти між сусідніми діапазонами – 1 Гц. |