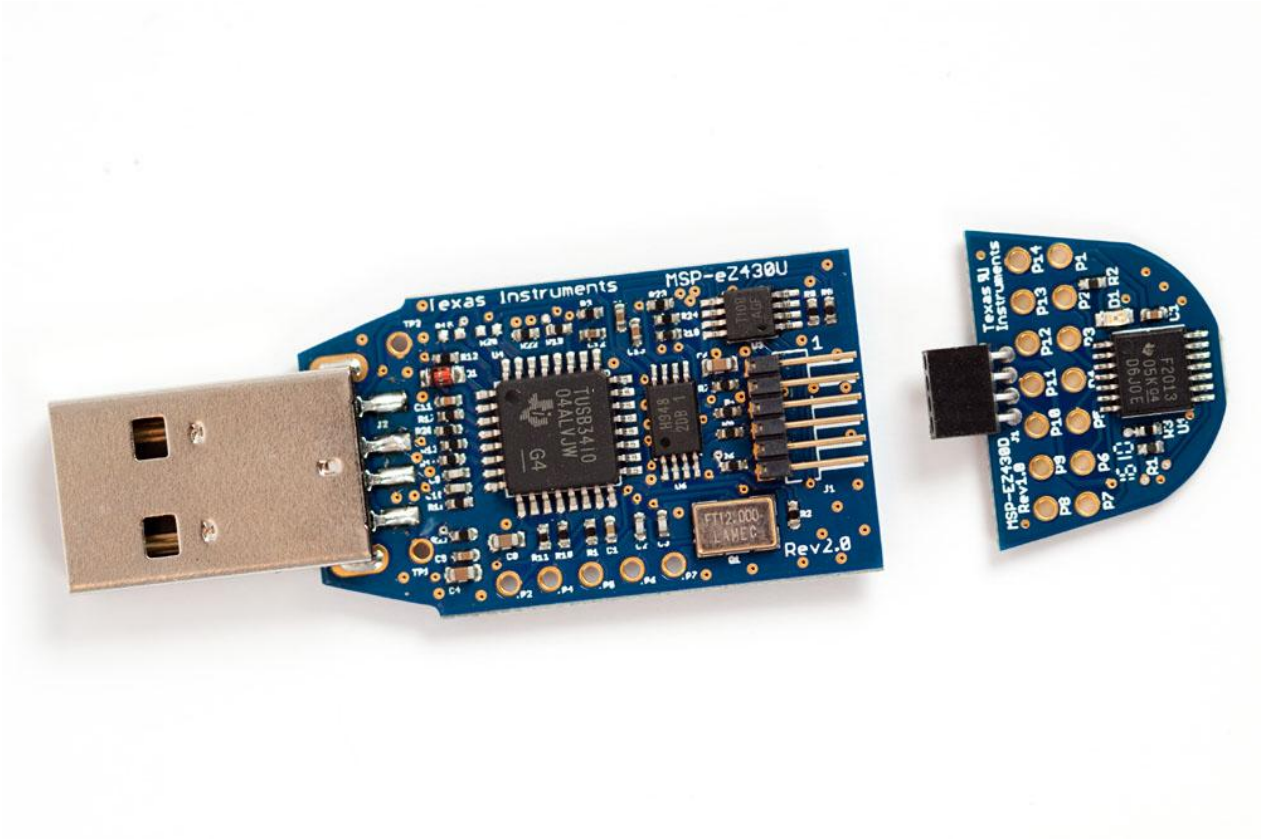
Texas Instruments MSP430

Texas Instruments MSP430

Мікропроцесорні системи.

Лабораторний практикум



Міністерство освіти і науки, молоді та спорту України
Вінницький національний технічний університет

МІКРОПРОЦЕСОРНІ СИСТЕМИ. ЛАБОРАТОРНИЙ ПРАКТИКУМ

Вінниця
ВНТУ
2013

УДК 004.3(075)
ББК 32.973-04 я 73
М91

Рекомендовано до видання Вченою Радою Вінницького національного технічного університету як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом підготовки «Системна інженерія»

Рецензенти:

І. І. Хаймзон, доктор технічних наук, професор

В.М. Лисогор, доктор технічних наук, професор

М. П. Мусієнко, доктор технічних наук, професор

М91 **Мікропроцесорні системи. Лабораторний практикум:** Навчальний посібник / [Квєтний Р.Н., Маслій Р. В., Гармаш В.В., Бойко О.Р.] – Вінниця : ВНТУ, 2013. – 106 с.

Навчальний посібник присвячений вивченню основ програмування пристроїв на базі мікроконтролерів MSP430 на мові C/C++. Розглянуті середовища програмування Code Composer Essential та IAR Embedded Workbench. В лабораторних роботах здійснюється дослідження пристрою eZ430-F2013.

Навчальний посібник призначений для студентів напряму підготовки «Системна інженерія».

УДК 004.3(075)

ББК 32.973-04 я 73

Навчальне видання

Квєтний Роман Наумович
Маслій Роман Васильович
Гармаш Володимир Володимирович
Бойко Олексій Романович

Мікропроцесорні системи. Лабораторний практикум

Навчальний посібник

Редактор В. Дружиніна
Оригінал-макет підготовлено Маслієм Р.В.

Підписано до друку
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. др. арк.
Наклад 300 прим. Зам. № 2012-

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95.
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95.
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-38.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 00.00.2012 р.

Міністерство освіти і науки, молоді та спорту України
Вінницький національний технічний університет

Мікропроцесорні системи. Лабораторний практикум

Усі цитати, цифровий, фактичний матеріал та бібліографічні відомості перевірні, написання відповідає стандартам. Зауваження рецензентів враховані

Вимогам, які висуваються до навчальної літератури, відповідає.
До друку і в світ дозволяю на підставі § 2 п. 15

Автори: Р.Н. Квєтний

Р.В. Маслій

В.В. Гармаш

О.Р. Бойко

Перший проректор з науково-педагогічної роботи по організації навчального процесу та його науково-методичного забезпечення

О.Н. Романюк

Затверджено на засіданні кафедри АІВТ.

Протокол № ___ від «____» _____ 20____ р.

Зав. кафедри _____ Р.Н. Квєтний

Вінниця ВНТУ 2013

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ACLK – Auxiliary CLocK (допоміжний тактовий сигнал)

ADC – Analog-to-Digital Converter (аналого-цифровий перетворювач, АЦП)

AM – active mode (активний режим)

BOR – Brown Out Reset (скидання при пониженні напруги живлення)

CCE – Code Composer Essentials

CPU – Central Processing Unit – центральний процесор

DAC – Digital-to-Analog Converter (цифро-аналоговий перетворювач, ЦАП)

DCO – Digitally Controlled Oscillator (генератор з цифровим управлінням)

GIE – General Interrupt Enable (загальний дозвіл переривань)

IDE – Integrated Development Environment (інтегроване середовище розробки)

I/O – Input/Output (введення/виведення)

ISR – Interrupt Service Routine (процедура обробки переривань)

ISP – In-System Programming (внутрішньосхемне програмування)

LSB – Least-Significant Bit (молодший значущий біт)

LSD – Least-Significant Digit (молодший значущий розряд)

LPM – Low-Power Mode (режим пониженого енергоспоживання)

MAB – Memory Address Bus (шина адреси)

MCLK – Master CLocK (основний тактовий сигнал)

MDB – Memory Data Bus (шина даних)

MSB – Most-Significant Bit (старший значущий біт)

MSD – Most-Significant Digit (старший значущий розряд)

NMI – Non-Maskable Interrupt (немасковане переривання)

PC – Program Counter (лічильник команд)

POR – Power-On Reset (скидання по ввімкненню живлення)

PUC – Power-Up Clear (очищення по ввімкненню живлення)

ROM – Read Only Memory (пам'ять тільки для читання)

RAM – Random Access Memory (пам'ять з довільним доступом)

SFR – Special Function Register (регістр спеціальних функцій)

SMCLK – Sub-system Master CLocK (додатковий тактовий сигнал)

SP – Stack Pointer (покажчик стека)

SR – Status Register (регістр стану)

TOS – Top-of-Stack (вершина стека)

WDT – WatchDog Timer (сторожовий таймер)

R/W – Read/Write (зчитування/запис)

R/O – Read only (тільки зчитування)

ОЗП – оперативно запам’ятовувальний пристрій

ПЗП – постійно запам’ятовувальний пристрій

ВСТУП

Інструмент розробника *(development tool)* eZ430-F2013 призначений для розробки і налагодження додатків *(applications)* на базі мікроконтролера *(microcontroller)* фірми Texas Instruments MSP430F2013. Він включає в себе програмні *(software)* та апаратні засоби *(hardware)* в зручному форм-факторі USB-stick. Для розробки додатків і забезпечення повної емуляції в інструмент розробника входять інтегровані середовища розробки *(integrated development environment)* IAR Embedded Workbench та Code Composer Essential. Крім того, конструктивом інструменту розробника передбачена заміна з'ємної плати *(detachable target board)* зовнішнім пристроєм за допомогою спеціально відведених виводів *(pins)*. Так як інструмент розробника призначений для використання з мікроконтролерами з наднизьким споживанням *(ultra-low power)*, живлення пристрою забезпечується USB-інтерфейсом *(interface)* комп'ютера *(computer)*, і зовнішні джерела при цьому не потрібні.

Відмінними рисами інструменту розробника є:

- інтерфейс USB;
- змінна конфігурація;
- світлодіодний індикатор стану *(LED)*, доступний для програмного управління;
- легкознімний прозорий корпус;
- зневаджувальний інтерфейс *(debugging interface)*, який підтримує всі мікросхеми серії MSP430F20xx;
- комплекс програмних засобів для розробки та зневадження *(debagging)* програм, що включає компілятор *(compilator)* асемблера *(assembler)* і C, компонувальник *(linker)*, емулятор *(emulator)* і зневаджувач *(debugger)*.

У комплект поставки інструменту розробника входять:

- пристрій *(device)* eZ430-F2013;
- IDE IAR Embedded Workbench;
- IDE Code Composer Essential;
- компакт-диск з програмним забезпеченням і документацією.

Даний посібник є продовженням серії, присвяченої вивченню мікроконтролерів MSP430. Крім того він є першим у серії посібників з лабораторного практикуму.

У лабораторних роботах *(laboratory works)* представлених у посібнику здійснюється програмування пристрою eZ430-F2013 на мові C/C++.

В першому розділі наведений короткий опис IDE IAR Embedded Workbench, покрокова інструкція створення, компіляції, зневадження та запуску проекту *(project)* на виконання, а також розглянута послідовність виконання програми у інтегрованому середовищі розробки.

В другому розділі наведений короткий опис IDE Code Composer Essentials та покрокова інструкція створення, компіляції, зневадження та запуску проекту на виконання.

В третьому розділі представлений лабораторний практикум.

В підрозділі 3.1 представлена лабораторна робота присвячена ознайомленню з лабораторним стендом на базі інструменту розробника eZ430-F2013.

В підрозділі 3.2 представлена лабораторна робота, в якій досліджується система переривань *(interrupt system)* та режими роботи *(operation modes)* мікроконтролера MSP430F2013.

В підрозділі 3.3 представлена лабораторна робота, в якій досліджується порти введення/виведення *(input/output ports)* роботи мікроконтролера MSP430F2013.

В підрозділі 3.4 представлена лабораторна робота, в якій досліджується модуль синхронізації *(synchronization module)* Basic Clock мікроконтролера MSP430F2013.

В підрозділі 3.5 представлена лабораторна робота, в якій досліджується робота сторожового таймера *(watchdog timer)* мікроконтролера MSP430F2013.

В підрозділі 3.6 представлена лабораторна робота, в якій досліджується робота Timer_A мікроконтролера MSP430F2013.

В підрозділі 3.7 представлена лабораторна робота, в якій досліджується робота флеш-пам'яті *(flash memory)* мікроконтролера MSP430F2013.

В підрозділі 3.8 представлена лабораторна робота, в якій досліджується модуль цифро-аналогового перетворення *(digital-to-analog conversation)* SD16_A мікроконтролера MSP430F2013.

Лабораторний практикум представлений у посібнику може використовуватися у дисциплінах: «Мікропроцесорні системи», «Проектування та програмування МПС», «Основи збору, передавання та обробки інформації», «Цифрові системи», «Функціональні перетворювачі СУА» а також використовуватися студентами напрямку «Системна інженерія», студентами суміжних напрямів.

Авторський колектив висловлює подяку студенткам Барченко К.В. та Васаженко А.С. за допомогу у перекладі технічної документації та оформленні матеріалів лабораторного практикуму.

1 ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ IAR EMBEDDED WORKBENCH

1.1 Загальна характеристика IDE IAR Embedded Workbench

IAR Embedded Workbench для MSP430 – це потужне інтегроване середовище розробки і зневадження програм для мікроконтролерів MSP430 за допомогою мов C, C++ та асемблера. Середовище забезпечує розширену підтримку пристроїв MSP430 і створює дуже компактний і ефективний код *(program code)*.

В комплект IAR Embedded Workbench входять:

- компілятор мови C/C++;
- асемблер;
- компонувальник;
- зневаджувач;
- редактор.

IAR Embedded Workbench забезпечує можливість взаємодії з зовнішніми програмами *(program)*.

Особливості компілятора мови C/C++:

- один з кращих компіляторів по ефективності коду;
- повна сумісність з ANSI C;
- кілька моделей для ефективного розподілу пам'яті;
- алгоритми оптимізації спеціально для мікроконтролерів MSP430;
- розширення мови для вбудованих систем *(embedded systems)*.

Особливості асемблера:

- інтегрований макроасемблер для додатків реального часу *(real time)*;
- включає препроцесор для компілятора C.

Особливості компонувальника:

- підтримує повне компонування, розміщення, і створення формату;
- підтримує більше 30 стандартних вихідних форматів для використання спільно з внутрішньосхемними емуляторами;
- завантаження модулів тільки при необхідності.

Особливості зневаджувача:

- зневадження кодів C, C++ та асемблера;
- різні точки зупину *(breakpoint)*;
- мова опису периферії та операцій введення/виведення;
- перегляд областей CODE, DATA, EEPROM та регістрів *(registers)* введення/виведення;
- обробка переривань з прогнозом;
- контроль будь-яких змінних і стека *(stack)*;
- комплексні типи даних.

Особливості редактору вихідного тексту:

- зручний інтерфейс користувача;
- автоматичне виділення помилок;
- зручна панель інструментів;
- виділення директив C/C++;
- розвинені засоби пошуку.

Вбудований редактор спеціально налаштований на синтаксис мови С, а додаткові утиліти і вбудована система допомоги додатково полегшують написання програм.

1.2 Створення проекту в IDE IAR Embedded Workbench

Нижче приводиться опис покрокової процедури *(procedure)* створення проекту в інтегрованому середовищі IAR Embedded Workbench.

1) Запустити середовище IAR Embedded Workbench. Створити новий проект: Project → Create New Project. У меню, що відкрилося, необхідно вибрати Tool chain: MSP430 (рис. 1.1). Натиснути «OK».

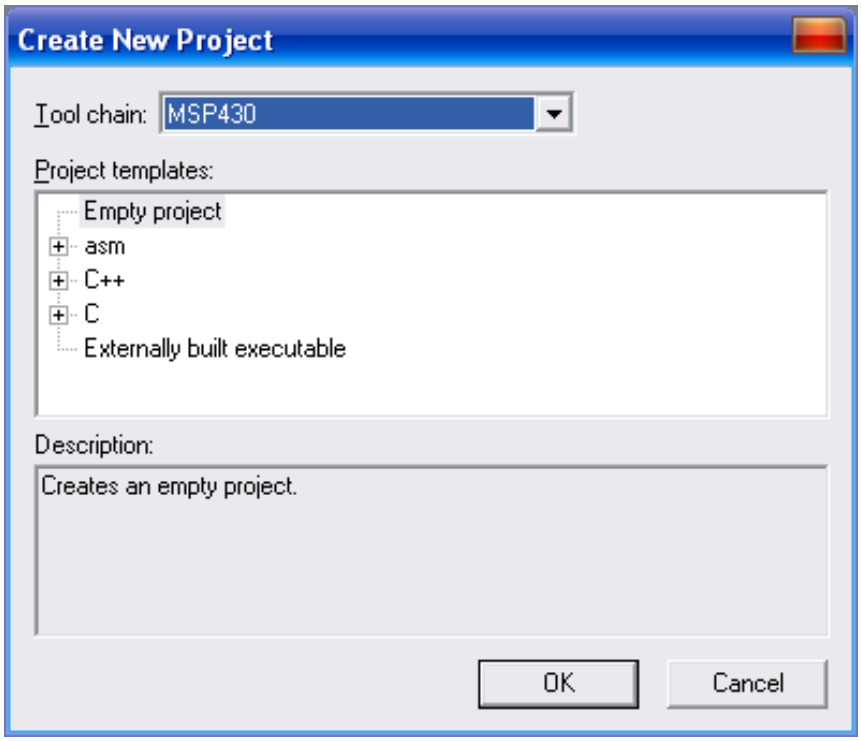


Рисунок 1.1 – Створення нового проекту

2) Ввести назву проекту, наприклад, «LED Blink» і вказати теку, в якій зберігатиметься проект, наприклад, "My Projects\LED Blink" (рис. 1.2). Далі натиснути «Сохранить».

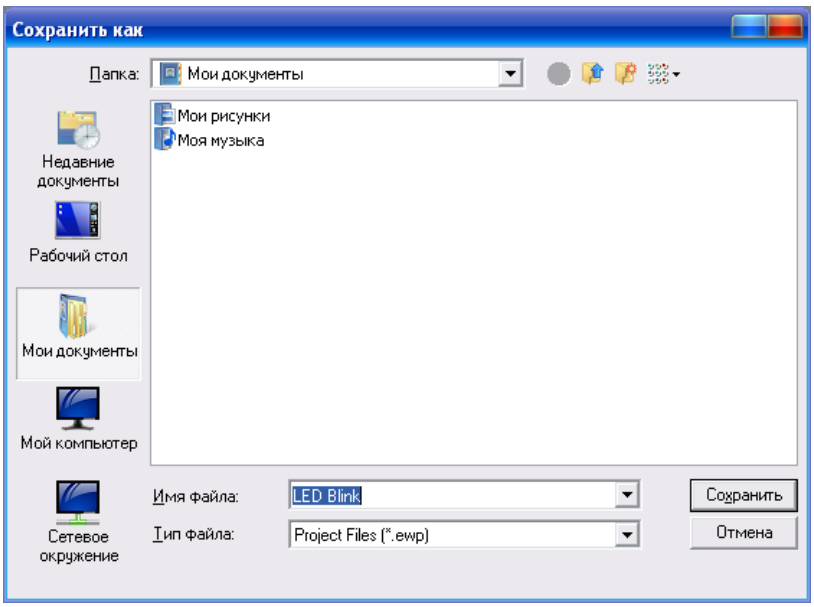


Рисунок 1.2 – Збереження проекту

3) У вікні середовища розробки, зліва, повинно з’явитися вікно Workspace, воно відображає файли проекту. Якщо вікна немає, необхідно вибрати View → Workspace. Серед файлів можна побачити файл LED Blink (рис. 1.3). Клацнути по ньому правою кнопкою мишки і вибрати Options.

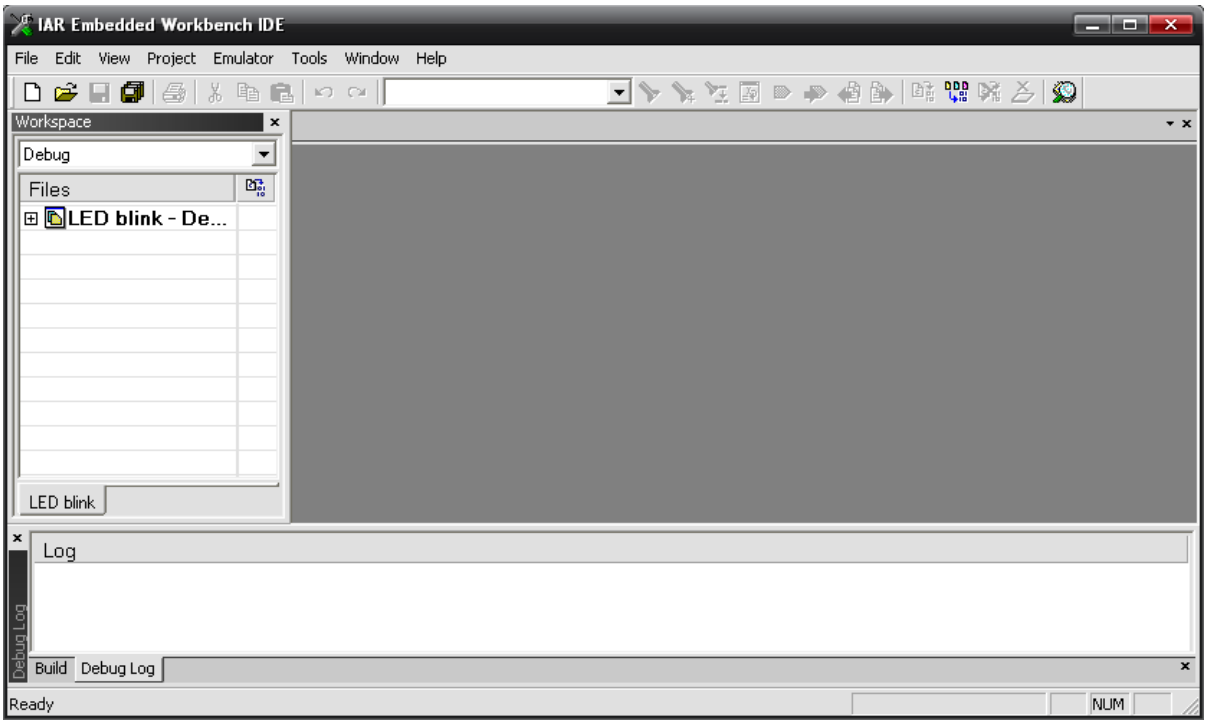


Рисунок 1.3 – Вікно Workspace

4) У категорії General Options в графі Device вибрати мікроконтролер з яким необхідно працювати: MSP430F2013 (рис. 1.4).

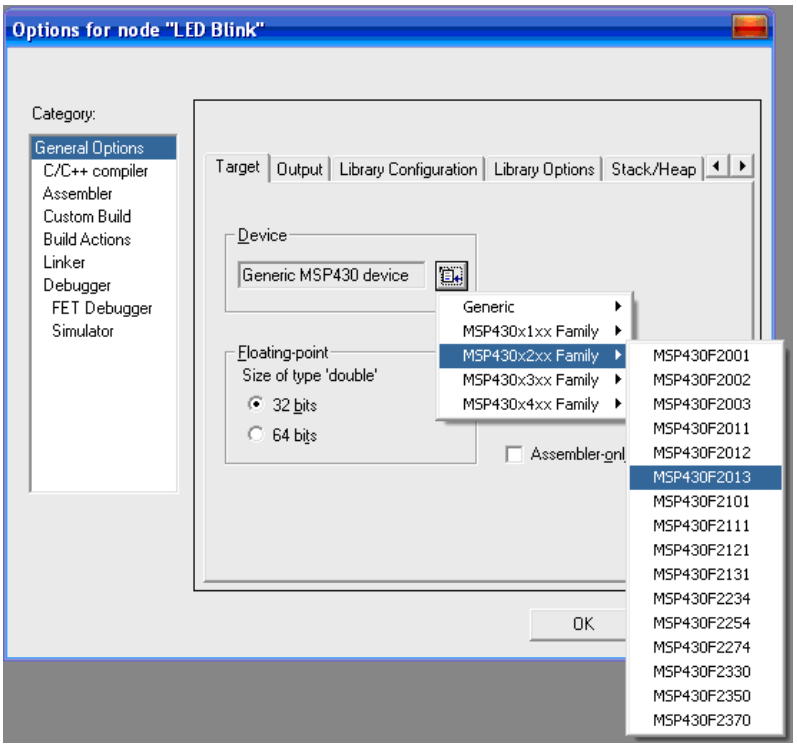


Рисунок 1.4 – Вибір мікроконтролера

5) У категорії Debugger, закладці Setup, графі Driver, вибрати FET Debugger (рис. 1.5). Натиснути «OK».

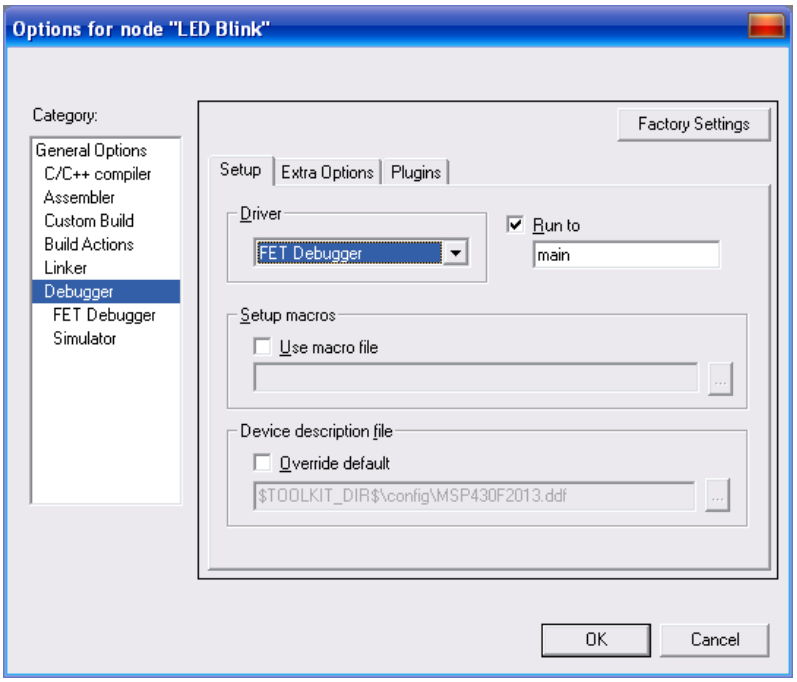


Рисунок 1.5 – Налаштування категорії Debugger

6) Тепер необхідно створити файл, в якому буде розміщена програма: File → New → File. Відразу необхідно зберегти його: File → Save As. Назвати його, наприклад "LED Blink.c".

7) Скопіювати у файл “LED Blink.c” наступний код програми (рис. 1.6):

```
#include  <msp430x20x3.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;/*Зупинка сторожового
                                    таймера */
  P1DIR |= 0x01; /* Налаштування порту P1.0 на
                     виведення */
  for (;;)
  {
    volatile unsigned int i;
    P1OUT ^= 0x01;/* Перемикання порту P1.0 з
                    використанням виключного АБО */
    i = 50000;     // Затримка
    do
    {
      (i--);
    }
    while (i != 0);
  }
}
```

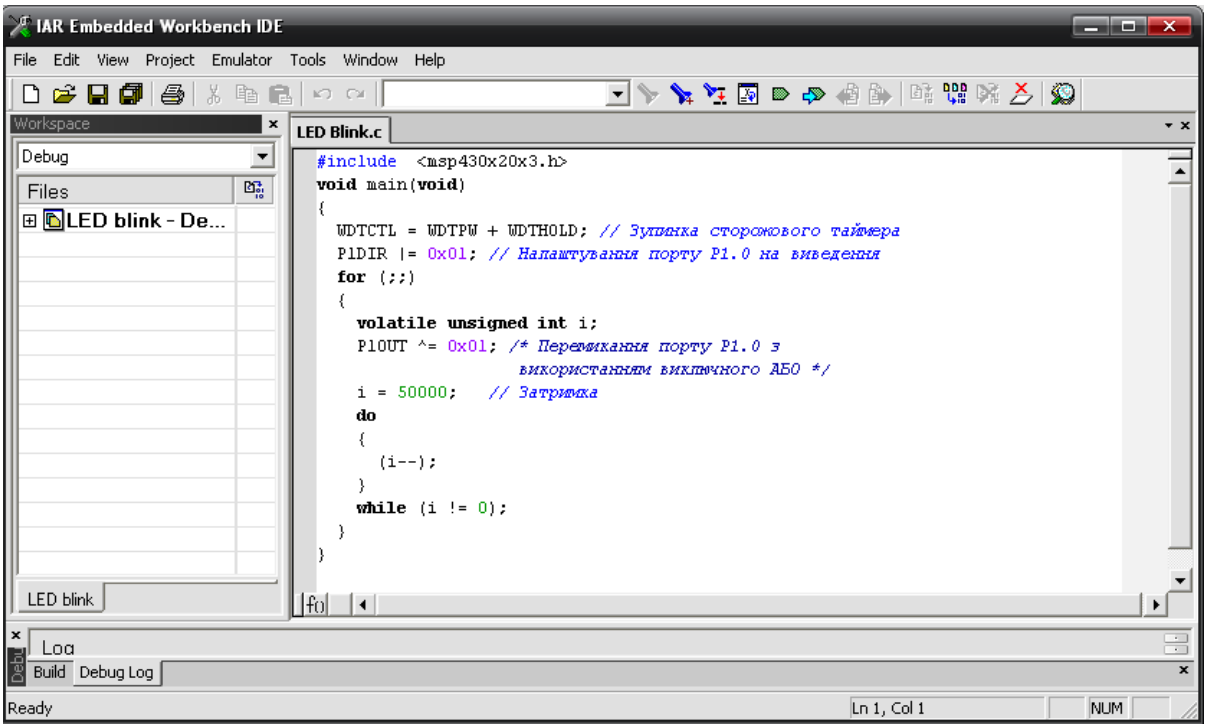


Рисунок 1.6 – Програма "LED Blink.c"

8) Зберегти файл "LED Blink.c" після змін. Підключити його до проекту: правою кнопкою мишки клацнути у вікні Workspace по назві проекту, вибрати Add → Add Files, потім файл "LED Blink.c" (рис. 1.7).

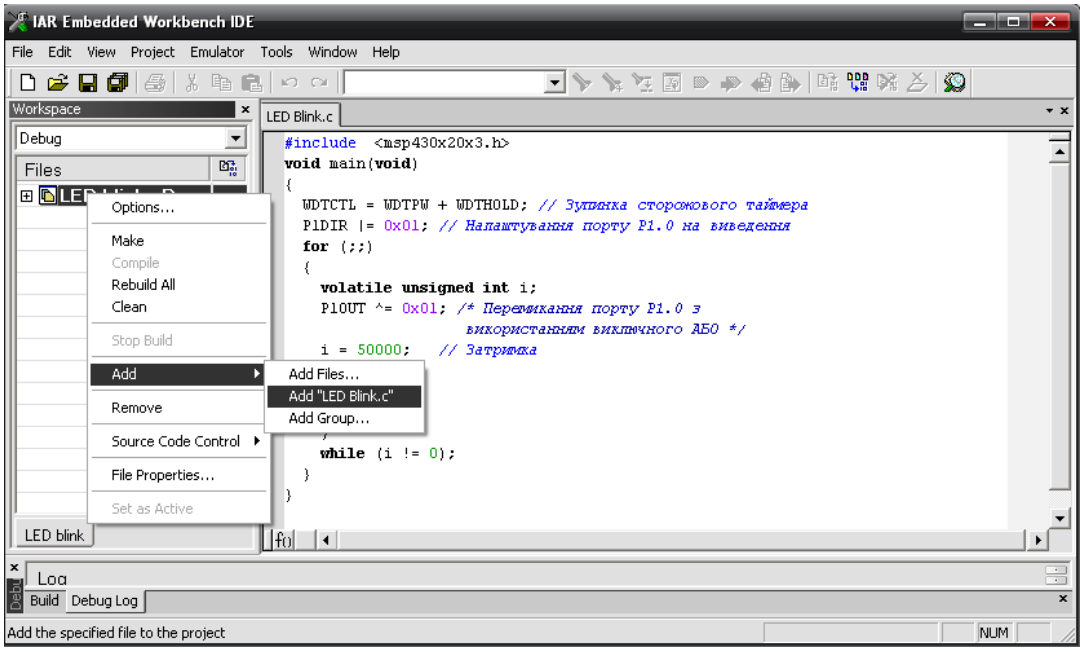


Рисунок 1.7 – Підключення файлу до проекту

9) Компіляція проекту. Вибрати пункт головного меню: Project → Rebuild All (рис. 1.8). Відкриється вікно з пропозицією зберегти робочу область: Save Workspace As. Зберегти робочу область в тій же теці My Projects\LED Blink. Натиснути «Save».

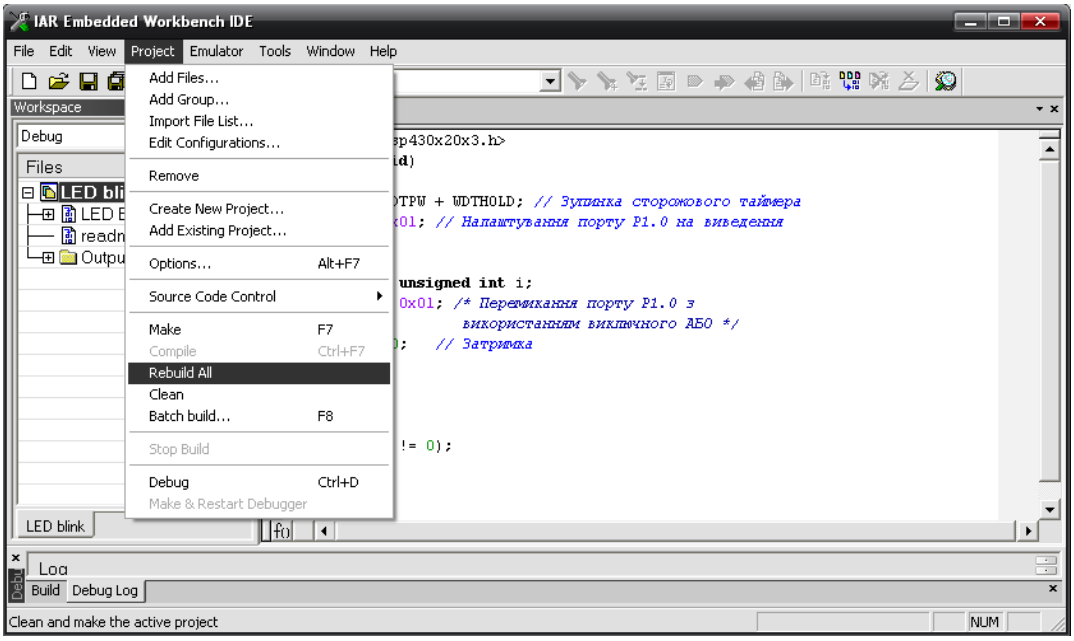


Рисунок 1.8 – Компіляція проекту

10) Зневадження проекту та запуск на виконання. Для зневаження пристрій eZ430-F2013 повинен бути підключений до USB порту. Натиснути Project → Debug (рис. 1.9). Натиснути F5 і спостерігати як блимає світлодіод. Для зупинки блимання світлодіоду потрібно натиснути Debug → Break, для виходу з режиму зневадження: Debug → Stop Debugging.

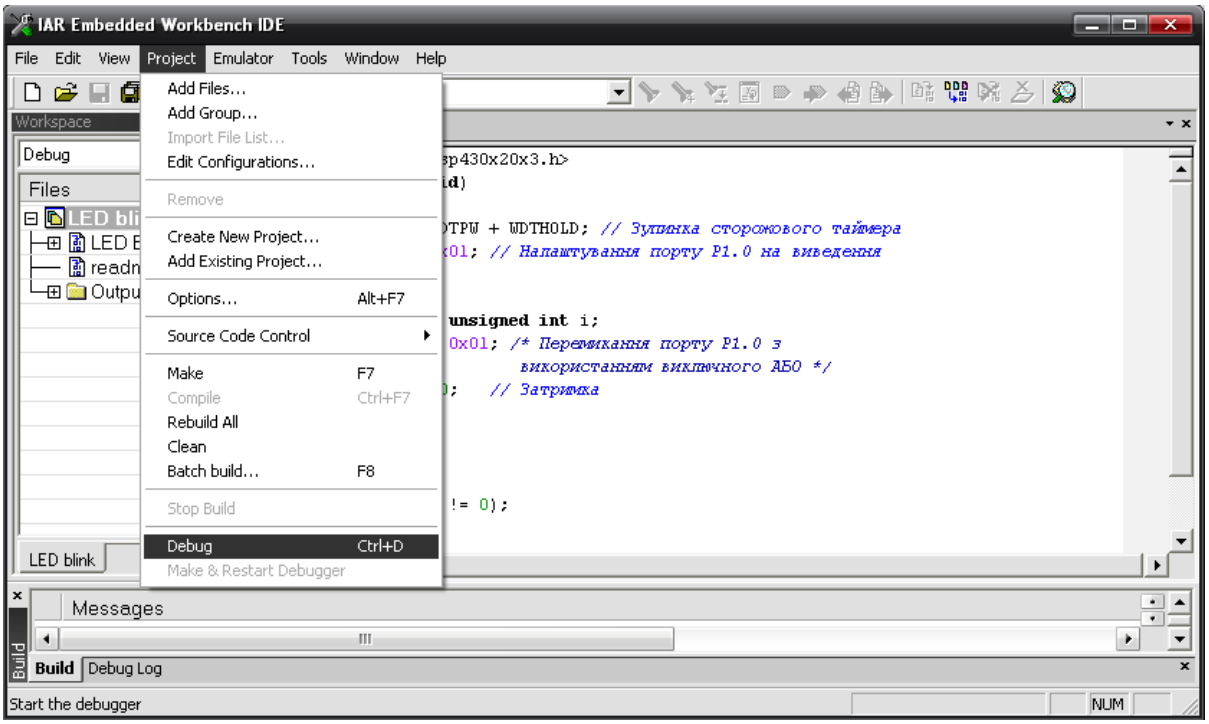


Рисунок 1.9 – Зневадження проекту

1.3 Виконання програми на C/C++ у інтегрованому середовищі розробки

На рис. 1.10 наведена послідовність виконання програми на C/C++ у інтегрованому середовищі розробки.

Основні елементи послідовності виконання програми на C/C++ у IDE:

- файли C/C++;
- компілятор C/C++;
- код асемблера;
- асемблер;
- об’єктні файли *(object files)*;
- компонувальник;
- виконуваний об’єктний файл;
- MSP430.

Інші елементи наведені на рис. 1.10 є опціональними.

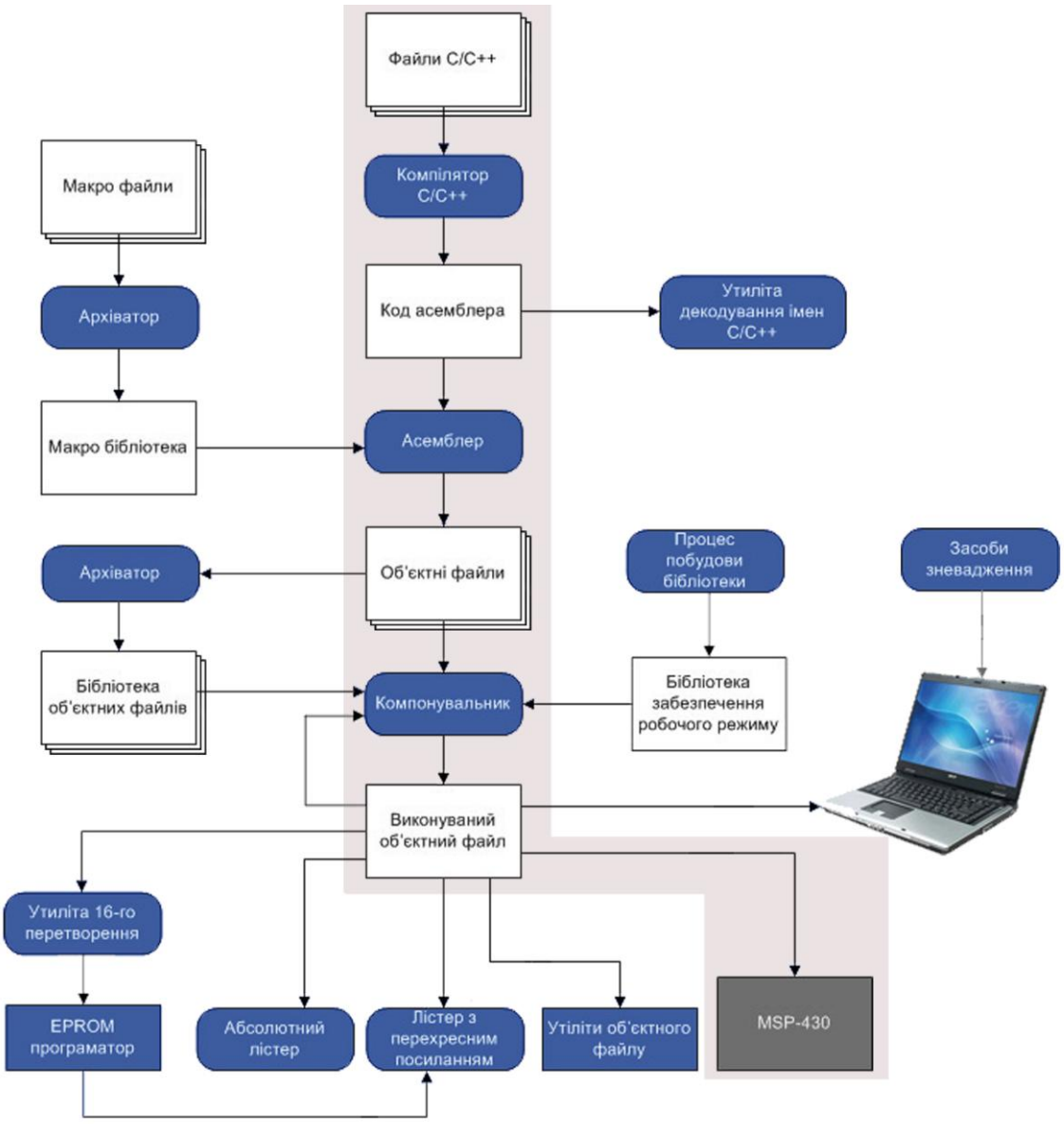


Рисунок 1.10 – Послідовність виконання програми на C/C++ у IDE

Компілятор перетворює код C/C++ у код асемблера MSP430.
Тобто якщо записати в програмі наступний код:

```
int a,b,c;
a = 3;
b = 5;
c = a+b;
```

Компілятор перетворить цей код наступним чином:

```
mov.w   #0x3,R15
mov.w   #0x5, R14
add.w   R14,R15
```

Завдяки компілятору, програміст може використовувати мову C/C++, що значно спрощує процес розробки програмного забезпечення. Саме у об'ємі компільованого коду виробники IDE для MSP430 вносять обмеження для того, щоб надати безкоштовну тестову версію.

Результат роботи компілятора можна побачити у вікні Disassembly, коли запускаємо режим зневадження проекту (Project → Debug) в середовищі розробки IAR Embedded Workbench IDE.

Асемблер – це перекладач з мови асемблера на мову машинних кодів, тобто в об'єктний код. Таким чином, отриманий код асемблера:

```
mov.w   #0x3,R15
mov.w   #0x5, R14
add.w    R14,R15
```

на мові машинних кодів буде мати наступний вигляд:

```
00F80E    403F 0003
00F812    403E 0005
00F816    5E0F
```

Компонувальник об’єднує всі об'єктні файли проекту в один виконуваний об'єктний модуль.

Архіватор *(arhiver)* – дозволяє створювати бібліотеки функцій, які часто використовуються, а також модифікувати вже існуючі бібліотеки. На сайті Texas Instruments *www.ti.com* можна безкоштовно завантажити бібліотеки оптимізованих функцій для роботи з деякими периферійними пристроями *(peripheral devices)*.

Контрольні запитання

1. Що входить в комплект IAR Embedded Workbench?
2. Які особливості компілятора мови C/C+?
3. Які особливості асемблера?
4. Які особливості компонувальника?
5. Які особливості зневаджувача?
6. Які особливості редактору вихідного тексту?
7. Як відбувається створення проекту в IDE IAR Embedded Workbench?
8. Яким чином здійснюється компіляція проекту?
9. Яким чином здійснюється зневадження проекту та запуск на виконання?
10. Які основні елементи послідовності виконання програми на C/C++ у IDE ?
11. Яке призначення асемблера?
12. Яке призначення компонувальника?
13. Яке призначення архіватора?

2 ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ CODE COMPOSER ESSENTIALS

2.1 Загальна характеристика IDE Code Composer Essentials

Code Composer Essentials – інтегроване середовище розробки і зневадження для мікроконтролерів сімейства MSP430.

Середовище CCE засноване на відкритій платформі Eclipse, яка призначена для створення інструментарію розробки і надає розробнику вільний вибір засобів інтеграції, моделювання та тестування, вибір мови програмування, платформи та постачальника цих засобів.

Використання платформи Eclipse дає розробнику спеціалізовані засоби і набір додаткових програмних модулів. Цей інструментарій покликаний спростити інтеграцію програмних засобів від різних постачальників і зменшити час розробки та її вартість. Платформа Eclipse забезпечує широкий діапазон вибору сумісних продуктів, дозволяє розробнику використовувати різні операційні системи, зневаджувачі і компілятори, виходячи з власних уподобань.

У складі CCE є весь необхідний набір відповідних засобів розробки:

- компілятор C, асемблер і лінкер для ядра MSP430;
- зневаджувач на рівні вихідного коду;
- вбудований менеджер проектів;
- підтримка апаратних і програмних точок зупину;
- вбудований текстовий редактор.

Особливості текстового редактору:

- підсвічування синтаксису мови;
- контроль коректності при написанні коду;
- автоматичне підсвічування інформації про об'єкти (функції, змінні тощо);
- інформація, доступна під час зневадження:
    - змінні і обчислювані вирази,
    - комірки пам'яті,
    - внутрішні регістри ядра MSP430.

У версії CCE v2.0 введені поліпшення для зручності роботи з середовищем:

- поліпшена стабільність роботи зневаджувача;
- зручний менеджер проектів для початку роботи, що запускається при запуску середовища;
- підтримка мікроконтролерів з об'ємом флеш-пам'яті більше 64 Кбайт *(byte)* (зокрема, MSP430F461x);
- розширені можливості управління точками зупину:
- точка зупину при переповненні стека;
- точки зупину для змінних;

● можливість завдяки використанню відкритого графічного інтерфейсу Eclipse, підключати різні плагіни: наприклад, підтримку системи контролю версій SubVersion;

● вдосконалений компілятор С.

Особливості компілятора С:

● дозволяє формувати додаткову інформацію, в якій повідомляється розмір використовуваного стека, граф викликів функцій тощо;

● вбудована скриптова мова, яка дозволяє, наприклад, з'ясувати точний мінімальний розмір стека, необхідного для роботи програми;

● можливість підключення до CCE продуктів інших виробників – компіляторів С та компонувальників;

● підтримка роботи під операційною системою Linux.

Нове середовище розробки CCE v3.0 передбачає значні оновлення, спрямовані на збільшення продуктивності, імітації функцій і зручність використання при достатньо низькій вартості. CCE спрощує процес розробки за рахунок інтуїтивного інтерфейсу в поєднанні з високою щільністю коду та потужними можливостями зневадження.

Використовуючи експлуатаційну гнучкість і функціональну сумісність, яку надає платформа з відкритим кодом Eclipse, розробники можуть легко створювати зручні для застосування замовні IDE шляхом інтегрування плагінів.

Основні поліпшення включають оптимізацію і зручності, орієнтовані на підвищення якості розробки, що спрощують зневадження і верифікацію і прискорюють час виведення виробів на ринок, при цьому вартість системи утримується на низькому рівні.

Середовище розробки CCE v3.0 характеризується:

● зростанням зручності у використанні: добре налагоджені діалогові вікна скорочують число кроків, необхідних для часто виконуваних завдань при розробці. Додатково, розробники мають спільний доступ до інших компіляторів, таких як MSPGCC GNU з допомогою єдиного середовища;

● розвиненим вбудованим емулятором: повна підтримка можливостей мікроконтролерів MSP430 при мінімальному втручанні у виконання коду, апаратна підтримка реального часу, включаючи точки зупину і відстеження стека, що приводить до більш швидкої емуляції і більш ефективного зневадження у всіх режимах роботи;

● високопродуктивним зневаджувачом: здійснена заміна зневаджувача на TI CCStudio Debug Server, який на 200 відсотків збільшує продуктивність в таких завданнях, як завантаження додатків, послідовне виконання та оновлення регістрів і змінних;

● оновленим синтаксисом коду С: дозволяє розробникам прямо імпортувати код без модифікації з великого набору прикладів, бібліотек і демонстрацій від TI і сторонніх фірм;

• підтримкою для всіх мікроконтролерів MSP430, яка дозволяє розробникам використовувати оптимальний вибір для своїх потреб, включаючи розширену пам'ять (до 1 Мбайт флеш-пам'яті), наприклад, в серії MSP430F261x.

CCE v3.0 є безкоштовною версією і підтримує до 16 Кбайт програмного коду. CCE Pro v3.0 характеризується необмеженим розміром програмного коду при вартості від $ 499.

2.2 Створення проекту в IDE Code Composer Essentials

Нижче приводиться опис покрокової процедури створення проекту з використанням середовища CCE.

1) Запустити середовище CCE. Створити новий проект: File → New → Managed Make C/ASM Project (рис. 2.1).

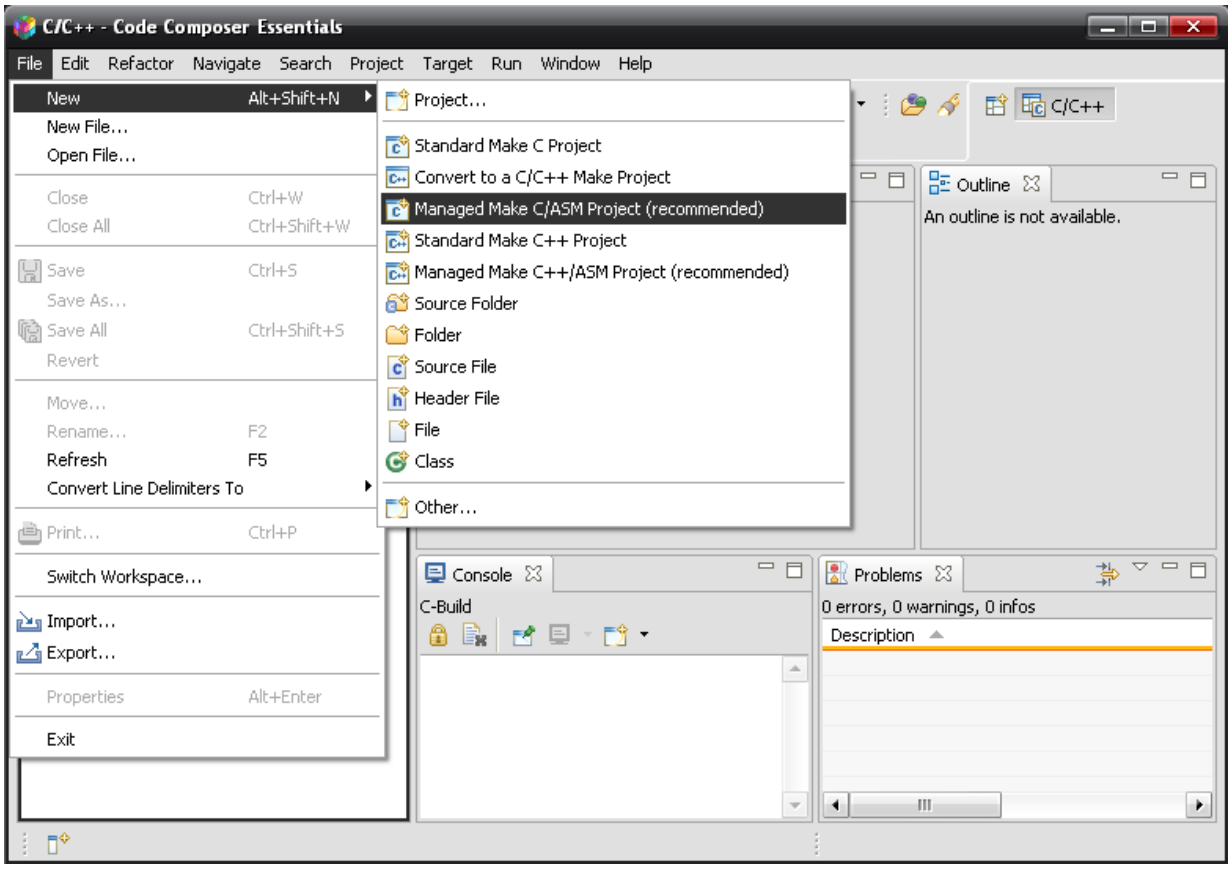


Рисунок 2.1 – Вибір пункту меню для створення проекту

2) У поле “Project name” ввести назву проекту (рис. 2.2). Натиснути "Next".

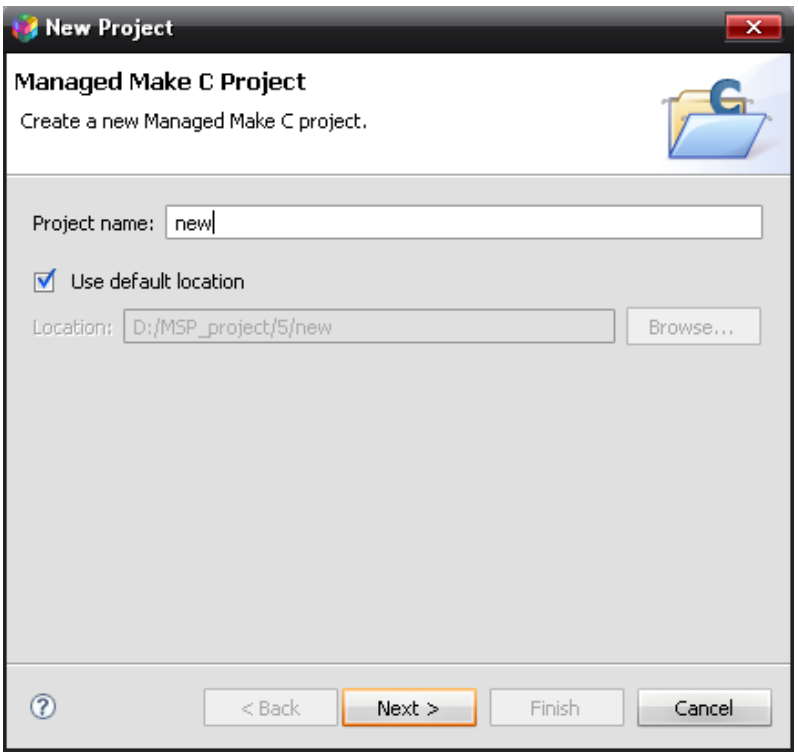


Рисунок 2.2 – Введення ім’я проекту

3) Перевірити, що у полі «Project Type» установлене значення "MSP430 Executable", а також, що вибрані конфігурації Debug та Release (рис. 2.3).

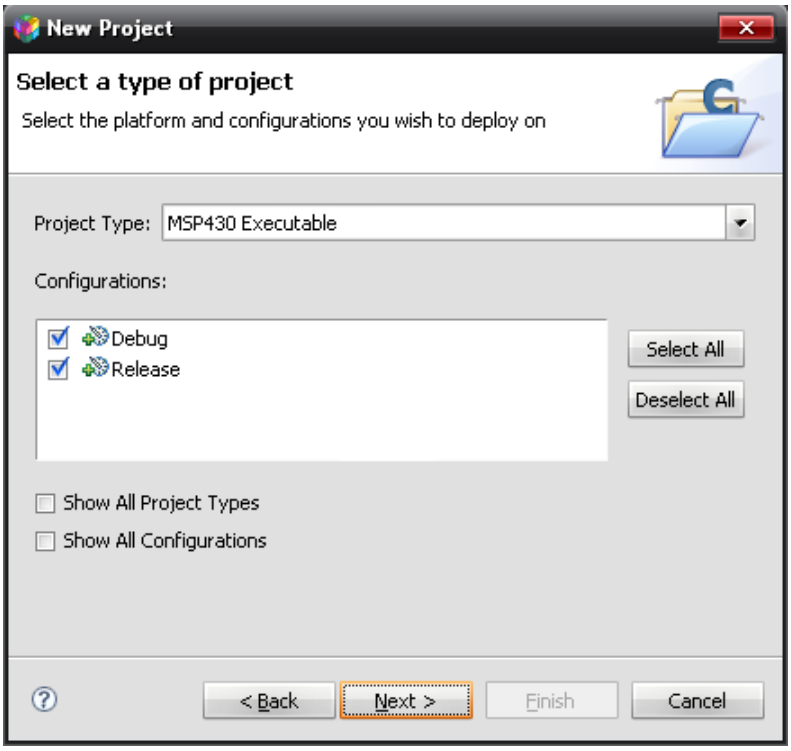


Рисунок 2.3 – Вибір типу проекту

4) У вкладці Projects перевірити відсутність посилань на проекти C/C++ (рис. 2.4). У вкладці C/C++ Indexer (рис. 2.5) вибрати "Full C/C++ Indexer (slow but accurate)". Після цього натиснути «Next».

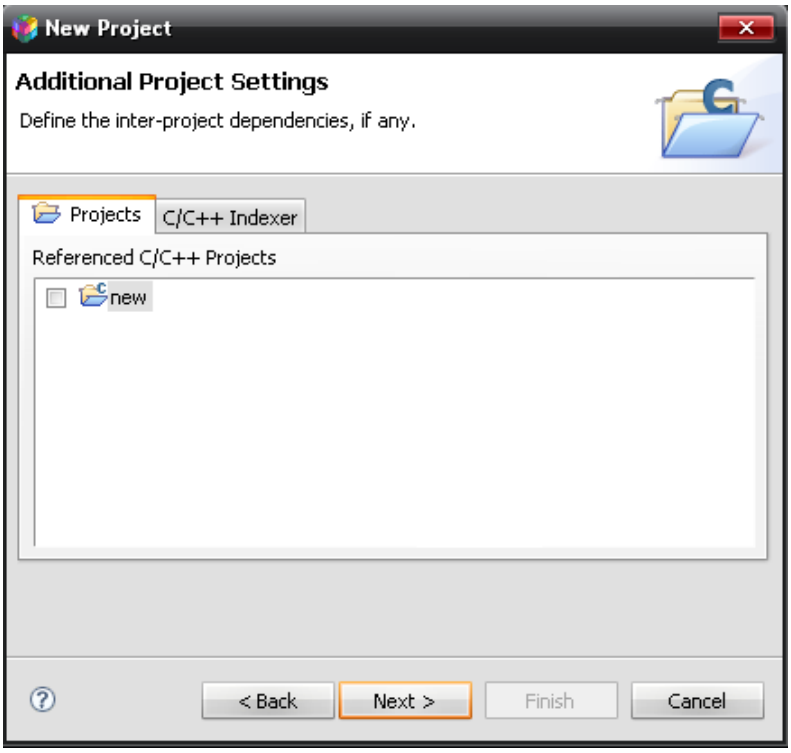


Рисунок 2.4 – Налаштування додаткових параметрів у вкладці Projects

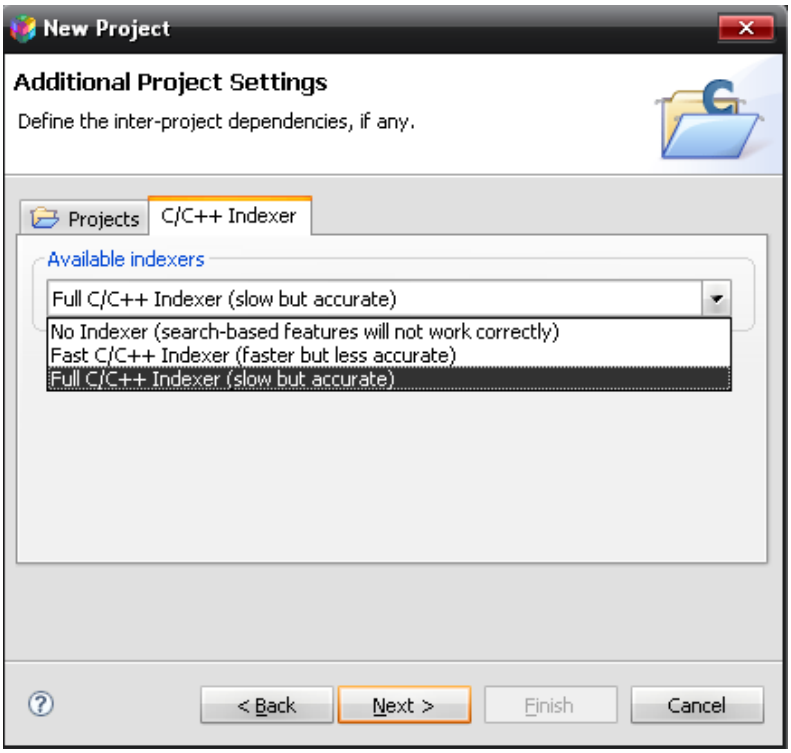


Рисунок 2.5 – Налаштування додаткових параметрів у вкладці C/C++Indexer

5) У списку «Device Variant» вибрати необхідний пристрій, наприклад MSP430F2013 (рис. 2.6). Після вибору пристрою автоматично відображається відповідний командний файл лінкер (Linker Command File). У випадку створення проекту з кодом асемблера відмітити галочкою поле "Configure as an assembly only project". У випадку створення проекту з кодом C/C++ це поле відмічати не потрібно. Далі натиснути "Finish".

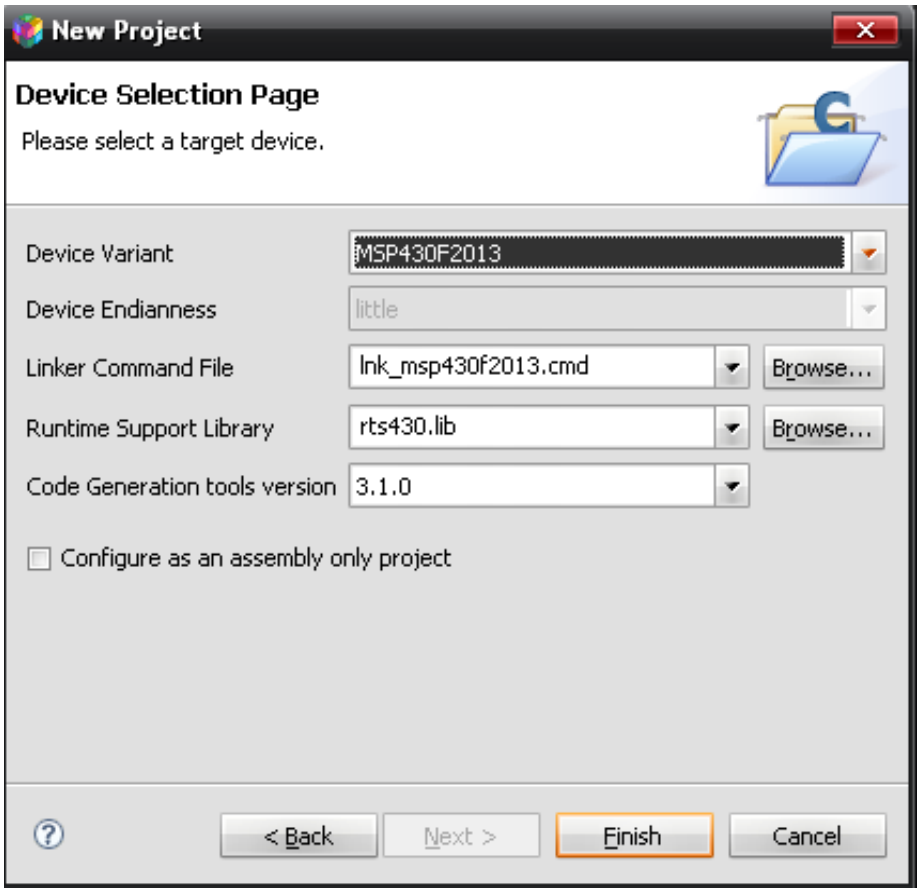


Рисунок 2.6 – Сторінка вибору пристрою

6) Для встановлення властивостей проекту необхідно клацнути правою кнопкою миші на назві проекту. Вибрати Properties → TI Debug Settings → Setup. В полі «Connection» вибрати тип з'єднання (рис. 2.7). Натиснути "OK".

7) Додавання файлу у проект: File → New → Source file. Ввести назву файлу (розширення ".c" або ".asm").

8) Компіляція проекту. Вибрати пункт головного меню: Project → Build Active Project.

9) Зневадження проекту. Для завантаження коду в MSP430 вибрати пункт головного меню: Run → Debug Active Project. Як варіант, можна вибрати кнопку Debug Active Project (рис. 2.8).

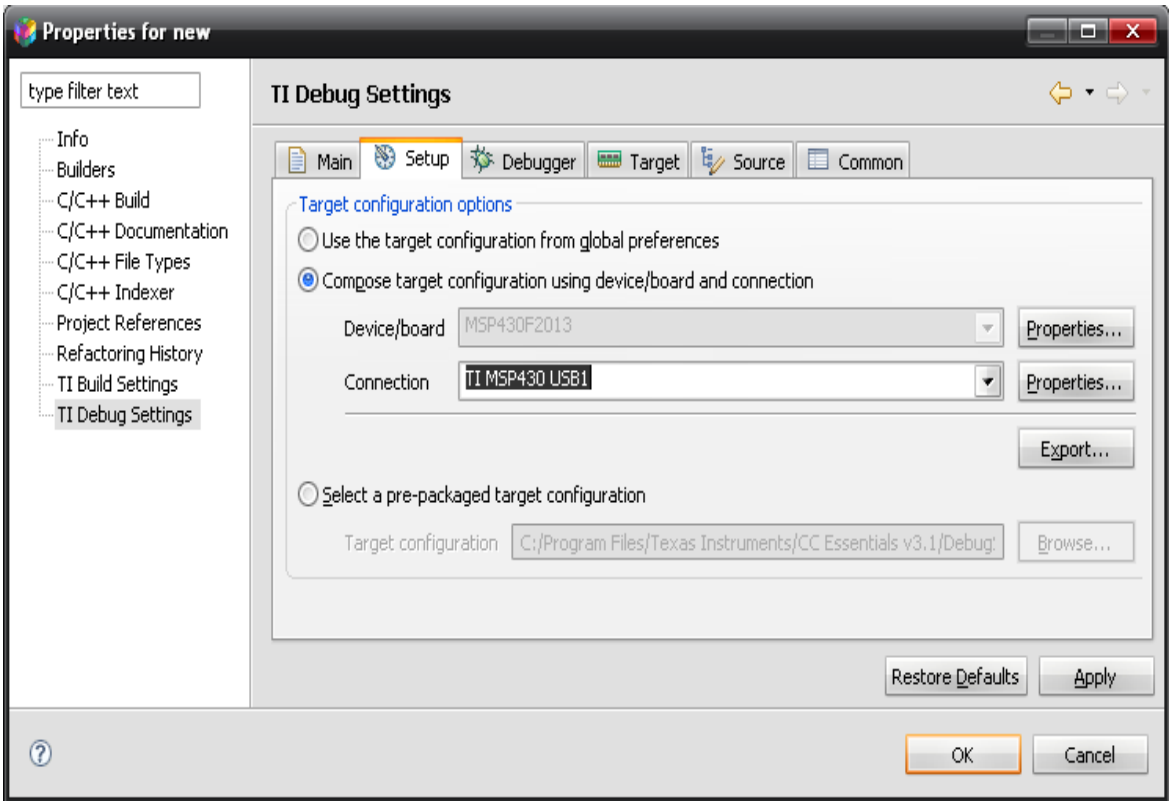


Рисунок 2.7 – Налаштування властивостей проекту

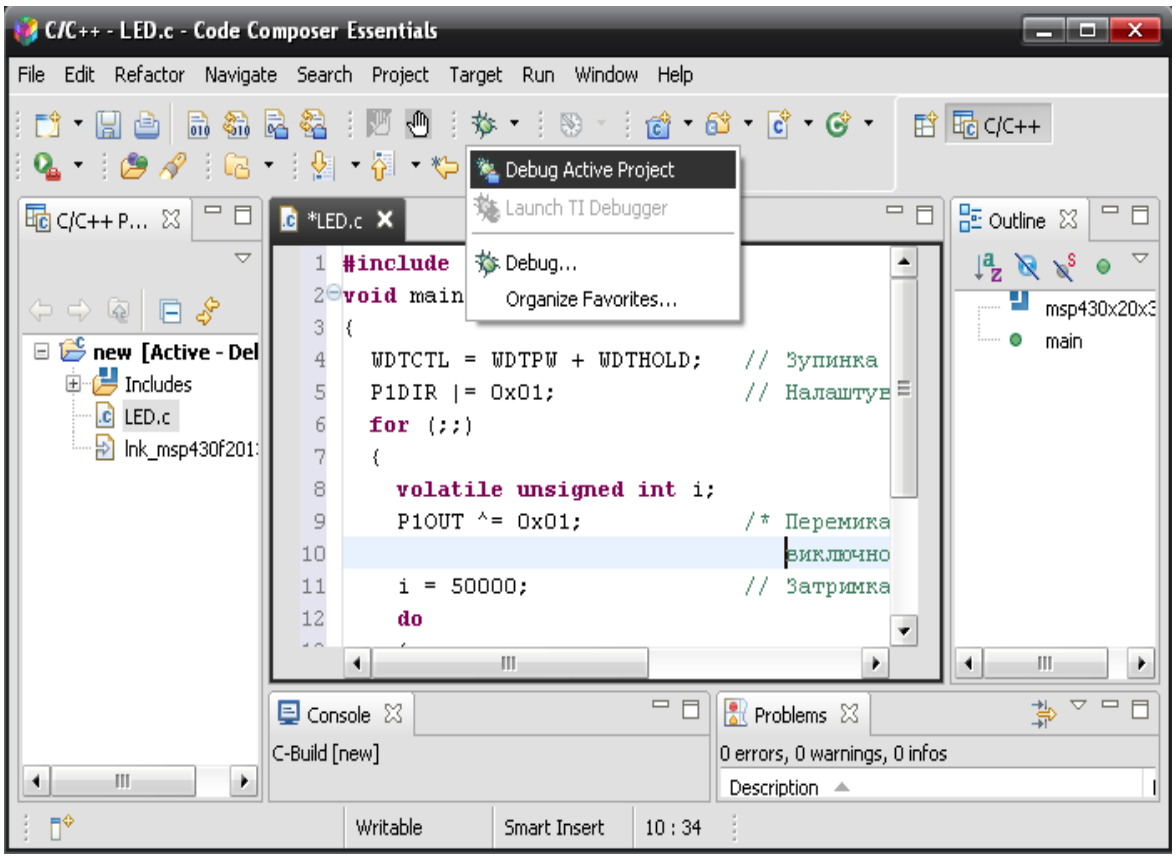


Рисунок 2.8 – Завантаження коду проекту на пристрій

10) Запуск проекту на виконання. Після успішної побудови й завантаження коду у пристрій відображається режим зневадження. Для виконання цього коду необхідно натиснути F8 або вибрати пункт головного меню Run → Run (рис. 2.9).

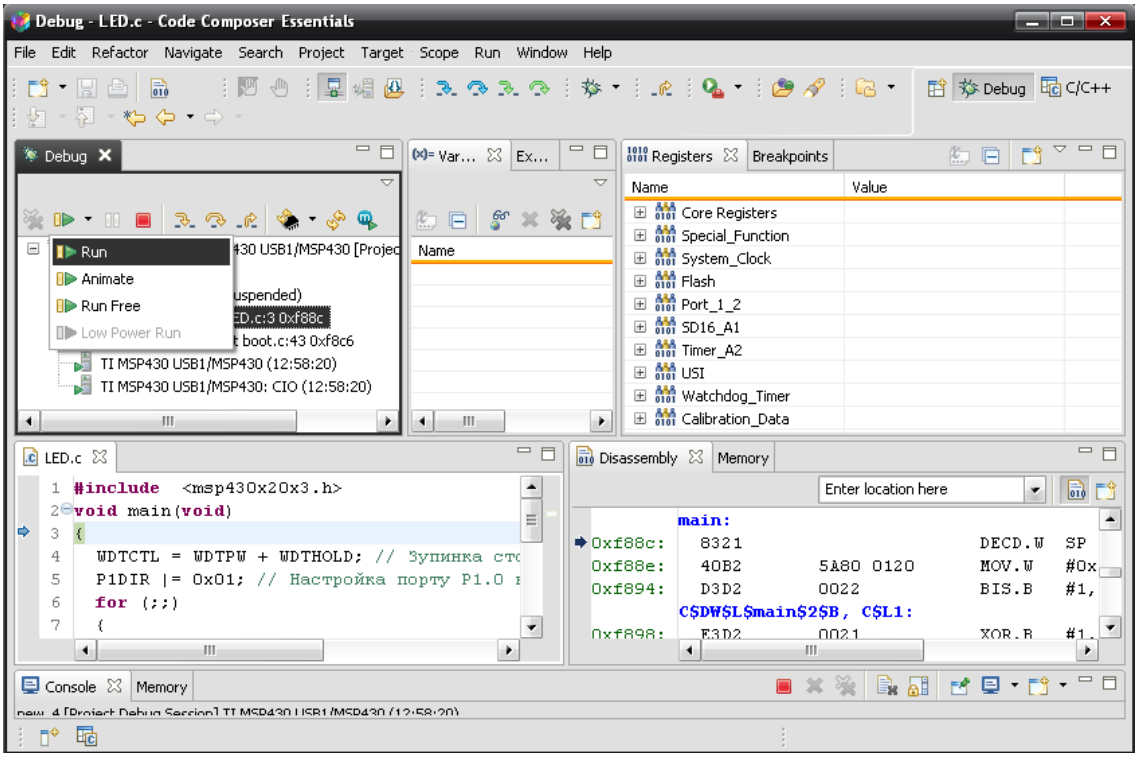


Рисунок 2.9 – Запуск проекту на виконання

Контрольні запитання

1. Що входить до складу Code Composer Essentials?
2. Яка інформація, доступна під час зневадження ?
3. Які особливості текстового редактору?
4. Які поліпшення введені для зручності роботи з середовищем у версії CCE v2.0?
5. Які особливості компілятора C у IDE CCE v2.0?
6. Чим характеризується середовище розробки CCE v3.0?
7. Як відбувається створення проекту в IDE Code Composer Essentials?
8. Яким чином вибирається необхідний пристрій?
9. Яким чином здійснюється компіляція проекту?
10. Яким чином здійснюється зневадження проекту та запуск на виконання?

3 ЛАБОРАТОРНИЙ ПРАКТИКУМ

3.1 Лабораторна робота №1. Ознайомлення з лабораторним стендом

Мета роботи: ознайомлення з можливостями та використанням інструменту розробника eZ430-F2013.

Теоретичні відомості

Інструмент розробника eZ430-F2013 складається з:

- пристрою eZ430-F2013;
- IDE IAR Embedded Workbench;
- IDE CCE;
- компакт-диску з програмним забезпеченням і документацією.

Лабораторний стенд складається з пристрою eZ430-F2013, комп'ютера та середовища розробки (CCE або IAR Embedded Workbench). Пристрій eZ430-F2013 є мініатюрною комплексною системою розробки додатків на мікроконтролерах.

Назва пристрою eZ430-F2013 розшифровується таким чином:

- eZ означає «easy» (англ. – легкий, зручний);
- 430 означає, що у пристрої використовується мікроконтролер з сімейства MSP430, представники якого характеризується низькою вартістю та наднизьким споживанням енергії;
- F2013 вказує на те, що у пристрої використовується один з наймініатюрніших мікроконтролерів сімейства MSP430.

Пристрій eZ430-F2013 підключається безпосередньо до стандартного USB-порту комп'ютера, живиться від нього ж, не вимагаючи більш ніяких кабелів і джерел живлення. У корпусі пристрою розміщуються інтерфейсна плата і з'ємна плата з мікроконтролером MSP430F2013. Ця універсальність і відрізняє eZ430 від інших систем подібного призначення. Пристрій використовує фірмовий інтерфейс Spy-Bi-Wire.

Для того, щоб побачити технічну будову пристрою, потрібно відкрити корпус за допомогою викрутки. У корпусі eZ430-F2013 (рис. 3.1) розміщені дві мініатюрні друковані плати:

- інтерфейсна плата eZ430U;
- з'ємна плата eZ430D.

До складу інтерфейсної плати eZ430U входить роз'єм USB і ланцюги, необхідні для підтримки інтерфейсу з програмним забезпеченням інтегрованого середовища розробки.

У інтерфейсній платі використовується контролер USB – Tusb3410 і регулятор живлення Tps77301, що подає напругу 3 В.

Додатково на нижній стороні інтерфейсної плати розміщений процесор MSP430F1612, призначений для прискорення взаємодії зі з'ємною платою.

До складу з’ємної плати входить мікроконтролер MSP430F2013, резистор установки сигналу *(signal)* «Reset» і розділовий конденсатор *(capacitor)*. Крім того на платі розміщений світлодіод для безпосереднього зворотного зв'язку при розробці коду і зневадженні. Всі 14 виводів мікроконтролера MSP430F2013 виведено на додаткові вивідні майданчики друкованої плати з крізними отворами кроком 0,1 дюйм і можуть взаємодіяти із зовнішніми сигналами для даного додатку або з іншими пристроями.

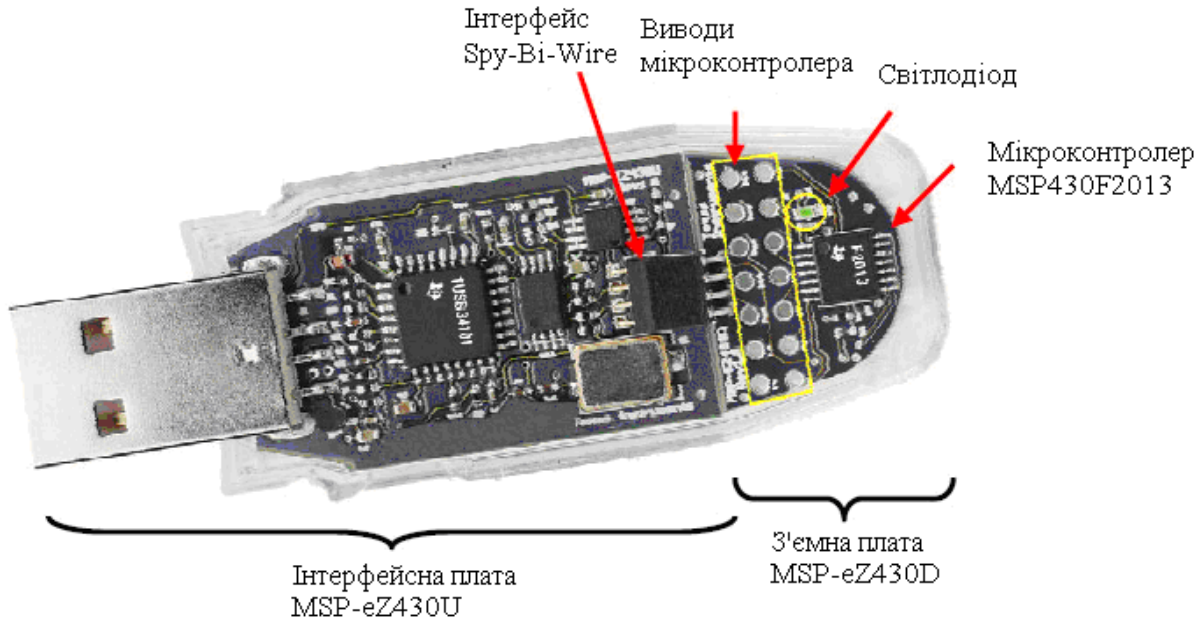


Рисунок 3.1 – Пристрій eZ430-F2013

Інтерфейсна плата eZ430U (зневаджувальний інтерфейс) і з’ємна плата eZ430D з'єднуються за допомогою 4-х вивідного роз'єму через інтерфейс Spy-Bi-Wire:

- сигнальною лінією SBWTCK (TEST);
- сигнальною лінією SBWTDIO (RESET);
- лінією живлення Vcc – 3 В;
- загальною лінією заземлення.

Втім, існує також можливість підключити інтерфейсну плату eZ430U до будь-якого з інших пристроїв MSP430F20xx через інтерфейс Spy-Bi-Wire.

З’ємна плата eZ430D від’єднується від інтерфейсної плати eZ430U шляхом обережного відтягування плат одна від одної (рис. 3.2).

Інтерфейс Spy-Bi-Wire призначений для програмування та зневадження, а також забезпечує повноцінну внутрисистемну емуляцію з використанням всього лише двох сигнальних виводів, не вживаних при звичайній роботі: TEST і RESET (SBWTCK і SBWTDIO).

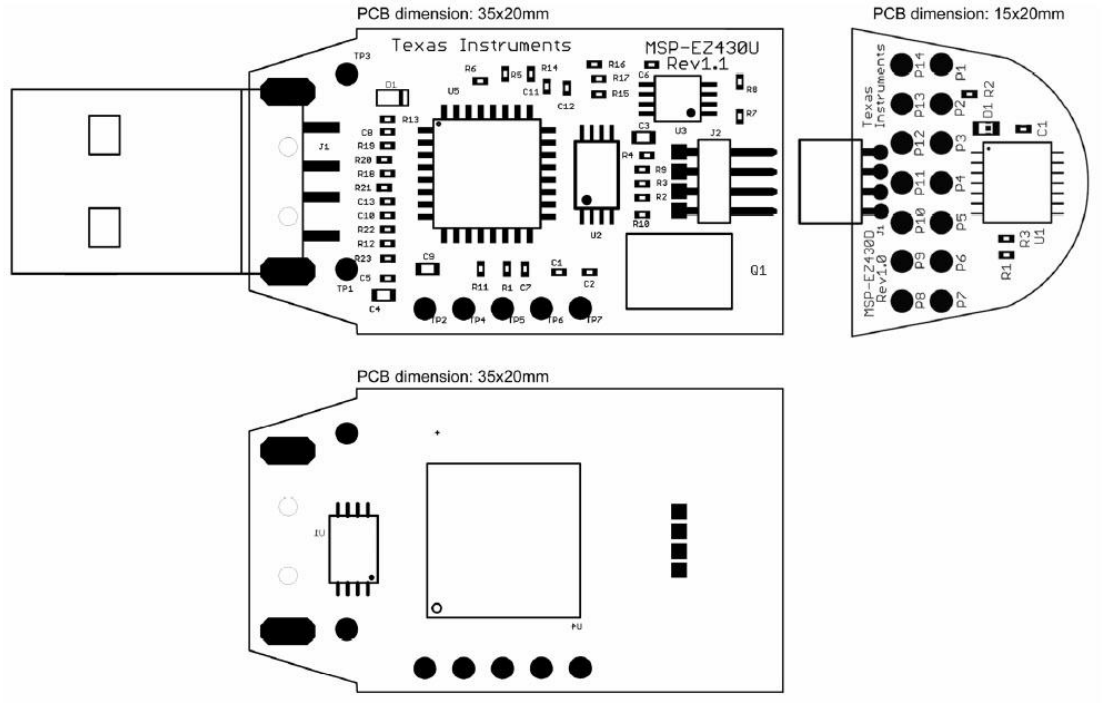


Рисунок 3.2 – Вигляд друкованих плат пристрою eZ430-F2013

Сигнали інтерфейсу Spy-Bi-Wire декодуються відповідно до стандарту JTAG MSP430F20xx. Емуляція усередині додатку зберігає час, оскільки розробка здійснюється в точності при тих же електричних характеристиках, які мають місце в кінцевому застосуванні, і не є обтяжливою в ситуаціях, пов'язаних з перенесенням. Емуляція, яку забезпечує інтерфейс Spy-Bi-Wire пристроїв MSP430F20xx, повністю сумісна з апаратними засобами компанії Texas Instruments на базі USB і програмно сумісна зі всіма доступними IDE сторонніх постачальників інструментальних засобів.

Комплексна внутрішньосистемна розробка, що включає програмування і зневадження на рівні асемблера або кодів на мові С, покрокове виконання, велику кількість апаратних встановлюваних контрольних точок, виконання на повній робочій швидкості і доступ до периферійних пристроїв — все це повною мірою підтримується усередині системи з використанням інтерфейсу Spy-Bi-Wire.

Типовий мікроконтролерний пристрій розробки показано на рисунку 3.3. Модуль емуляції звичайно є апаратною частиною, що керується головним модулем (host), наприклад, таким, як персональний комп’ютер. Програмне забезпечення керуючого модуля перетворює задані операції у послідовність JTAG-команд. JTAG-машина перетворює ці команди і відсилає їх у вигляді сигналів JTAG-інтерфейсу. Керуючий модуль разом із модулем емуляції ініціює усі зв’язки із кінцевим пристроєм. Таким чином, керуючий модуль виступає як основний пристрій (master) у системі master-slave, де кінцевий пристрій виступає як підлеглий (slave).

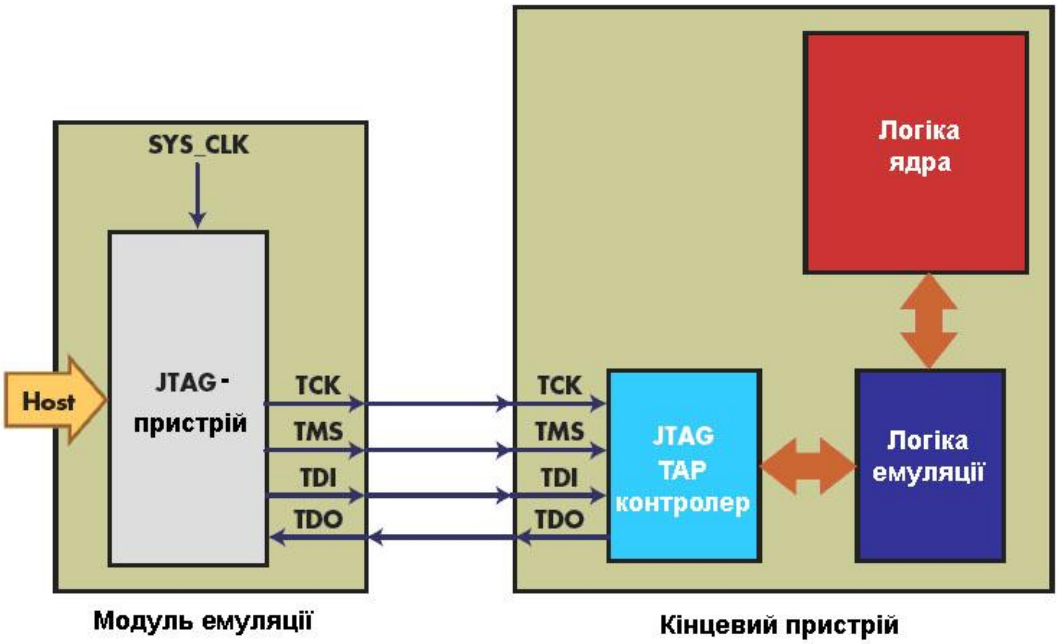


Рисунок 3.3 – Типовий мікроконтролерний пристрій розробки

Зміст завдання

1. Лабораторна робота присвячена ознайомленню з можливостями та використанням пристрою eZ430-F2013.

2. Завдання містить код програми, який необхідно відкомпілювати, зневадити та запустити на виконання.

3. При виконанні завдання необхідно змінювати значення часової затримки.

Код програми

```
#include  <msp430x20x3.h>
void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */
  P1DIR |= 0x01;  // налаштування P1.0 на виведення
  for (;;){
    volatile unsigned int i;
    P1OUT ^= 0x01; /* перемикання P1.0 з
                      використанням виключного АБО */
    i = 50000;     // затримка
    do {
      i--;
    }
    while (i != 0);
  }
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп’ютера.

2. Запустіть середовище розробки вказане викладачем (CCE або IAR).

3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми.

4. Здійсніть компіляцію та зневадження проекту.

5. Запустіть проект на виконання. Занесіть у звіт результат роботи програми.

6. Встановіть значення часової затримки i = 20000. Повторіть дії описані в пунктах 4-5.

7. Встановіть значення часової затримки i = 5000. Повторіть дії описані в пунктах 4-5.

8. Закрийте середовище розробки та від’єднайте пристрій eZ430-F2013 від USB-порту комп’ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- короткий опис інтегрованого середовища розробки;
- результати виконання;
- висновки.

Контрольні запитання

1. З чого складається інструмент розробника eZ430-F2013?
2. З чого складається лабораторний стенд?
3. Як розшифровується назва пристрою eZ430-F2013?
4. Як підключається та живиться пристрій eZ430-F2013?
5. Технічна будова пристрою eZ430-F2013.
6. З чого складається інтерфейсна плата eZ430U?
7. З чого складається з’ємна плата eZ430D?
8. Для чого призначений процесор MSP430F1612?
9. Яким чином з’єднуються інтерфейсна плата і з’ємна плата?
10. Для чого призначений інтерфейс Spy-Bi-Wire?
11. Відповідно до якого стандарту декодуються сигнали інтерфейсу Spy-Bi-Wire?
12. В чому заключається процес емуляції?

3.2 Лабораторна робота №2. Дослідження системи переривань та режимів роботи

Мета роботи: Ознайомлення з системою переривань та режимами роботи мікроконтролера MSP430F2013, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Переривання – це сигнал, що повідомляє систему про здійснення якої-небудь події. При цьому виконання поточної послідовності команд *(command)* припиняється і управління передається спеціальній функції – ISR, обробникові переривання. Якщо необхідно, щоб пристрій на базі MSP430 споживав якомога менше електроенергії, потрібно правильно використовувати переривання.

В ході виконання програми мікроконтролером зазвичай відбуваються різні події: АЦП завершив перетворення вхідного сигналу, таймер відлічив заданий інтервал, осцилятор вийшов з ладу тощо. Для відстеження цих подій і вчасної їх обробки може застосовуватися два підходи:

- перший підхід. У безперервному циклі відстежувати регістри статусу периферійних пристроїв і ядра. Недоліки: ядро мікроконтролера постійно включене, що приводить до збільшення енергоспоживання, програмне відстежування подій вимагає багато часу, що може привести до втрат інформації;
- другий підхід. Використовувати переривання. В цьому випадку мікроконтролер може знаходитися в режимі низького енергоспоживання, при виникненні певної події, ядро переходить в активний режим, включаються необхідні периферійні пристрої.

Пріоритети переривань фіксовані і залежать від місцезнаходження модуля відносно CPU. Чим ближче розташований модуль до CPU, тим вище пріоритет його переривання. Пріоритети визначають порядок обробки переривань при одночасній генерації декількох запитів.

У MSP430 є три типи переривань:

- скидання системи;
- немасковані *(non maskable interrupt)*;
- масковані.

Немасковані переривання

Немасковані переривання NMI не маскуються бітом *(bit)* загального дозволу переривань GIE, однак можуть бути окремо дозволені/заборонені за допомогою індивідуальних бітів дозволу переривання (NMIE, ACCVIE, OFIE).

При виникненні немаскованого переривання всі біти дозволу цього переривання автоматично скидаються. Виконання програми продовжується з адреси *(address)*, що міститься у векторі немаскованого переривання,

0FFFCh. Для повторного дозволу переривання користувача програма повинна знову встановити необхідні біти. Графічна схема обробки джерел немаскованого переривання приведена на рис. 3.4. Немасковане переривання може бути викликане трьома подіями:

- поява активного фронту на виводі RST/NMI при конфігурації цього в виводу в режимі NMI;
- виникнення несправності тактового генератора *(clock generator)*;
- порушення доступу до флеш-пам'яті.

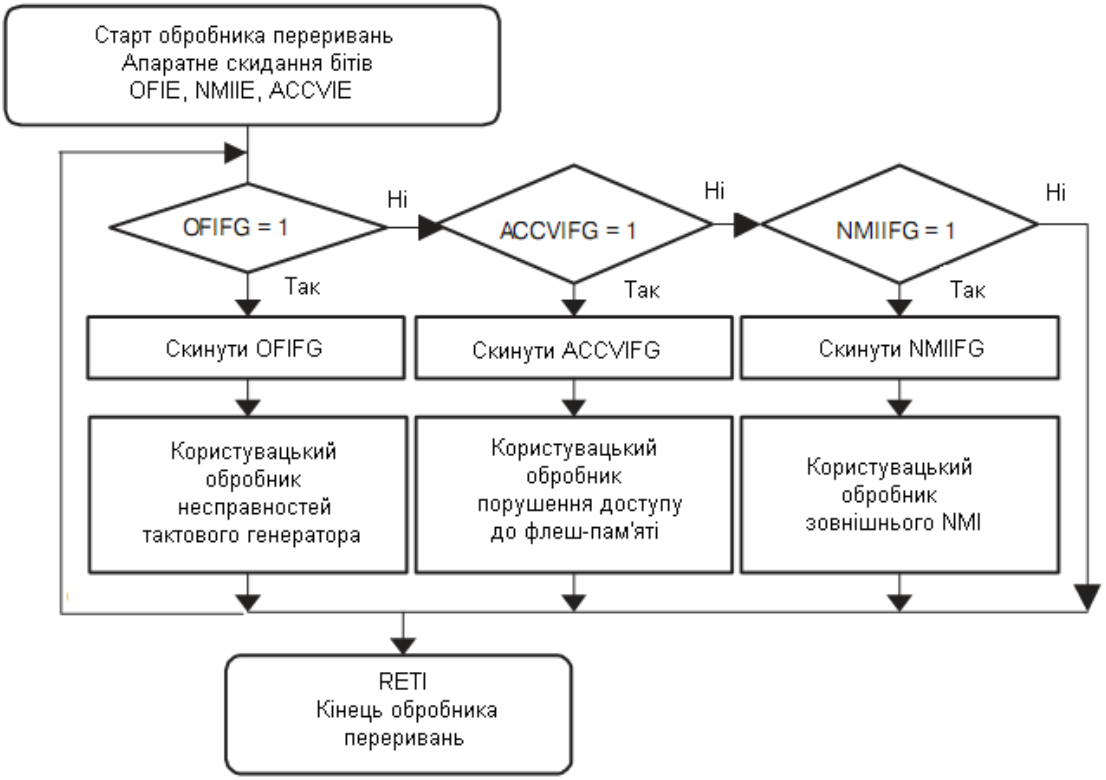


Рисунок 3.4 – Графічна схема обробки джерел немаскованого переривання

Вивід RST/NMI

При вмиканні мікроконтролера вивід RST/NMI конфігурується як вхід апаратного скидання. Призначення цього виводу задається в регістрі управління сторожового таймера WDTCTL. Якщо вивід RST/NMI використовується в якості входу скидання, то CPU буде утримуватися в стані скидання до тих пір, поки на цьому виводі буде присутній сигнал низького рівня. При подачі на вхід сигналу високого рівня CPU починає виконувати програму з адреси, яка зберігається у векторі скидання (0FFFEh). Одночасно з цим встановлюється прапорець *(flag)* RSTIF. Якщо вивід RST/NMI налаштований в програмі як вхід немаскованого переривання, то поява на цьому виводі активного фронту (задається бітом WDTNMIES) при встановленому біті NMIIE викликає генерацію немаскованого переривання. Також встановлюється прапорець NMIIFG.

Порушення доступу до флеш-пам'яті

При порушенні доступу до флеш-пам'яті встановлюється прапорець ACCVIFG. Генерація немаскованого переривання при виникненні такої ситуації дозволяється встановленням біта ACCVIE. У процедурі обробки немаскованого переривання можна перевірити прапорець ACCVIFG, щоб визначити, чи було переривання викликано саме порушенням доступу до флеш-пам'яті.

Несправність тактового генератора

Сигнал несправності генератора дозволяє запобігти помилкам, пов'язаним з неправильним функціонуванням кварцового генератора *(crystal oscillator)*. Генерація немаскованого переривання при виявленні несправності генератора дозволяється встановленням біта OFIE. У процедурі обробки немаскованого переривання можна перевірити прапорець OFIFG, щоб визначити, чи було переривання викликано саме збоєм в роботі генератора. Сигнал про несправність генератора може бути викликаний появою сигналу скидання PUC, оскільки останній переводить генератор LFXT1 з режиму HF в режим LF.

Приклад процедури обробки немаскованого переривання

Немасковане переривання може генеруватися різними джерелами. При виникненні переривання біти дозволу NMIIE, OFIE і ACCVIE автоматично скидаються. Процедура обробки немаскованого переривання скидає прапорці переривання і повторно дозволяє генерацію переривання від необхідних джерел відповідно до вимог додатку, як показано на рис. 4.1.

Масковані переривання

Масковані переривання генеруються периферійними пристроями, які мають таку можливість. У тому числі, масковані переривання можуть генеруватися за переповнення сторожового таймера при роботі останнього в режимі інтервального таймера *(interval timer)*. Переривання від кожного з джерел можуть бути заборонені з допомогою індивідуальних бітів дозволу переривань. Крім того, всі масковані переривання можуть бути заборонені за допомогою біта загального дозволу переривань GIE регістра стану SR.

Обробка переривання

При виникненні запиту переривання від периферійного пристрою, якщо встановлені біт дозволу переривання від цього пристрою і біт загального дозволу переривань GIE, викликається процедура обробки переривання. Для виклику обробника немаскованого переривання досить встановленого індивідуального біта дозволу конкретного переривання.

Приймання запиту переривання

Затримка обробки переривання, тобто час з моменту прийняття запиту переривання до початку виконання першої команди процедури обробки переривання (рис. 3.5), становить 6 тактів CPU.

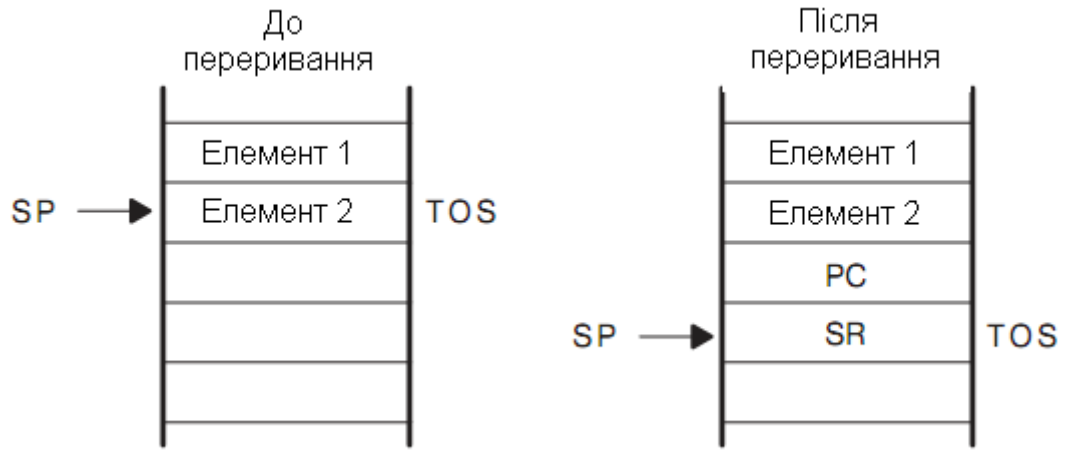


Рисунок 3.5 – Обробка запиту на переривання

Обробка запиту на переривання проводиться в такій послідовності:

1. Очікується завершення команди, що виконується в даний момент.
2. Вміст лічильника *(counter)* команд PC, що вказує на наступну команду, поміщається в стек.
3. Вміст регістра стану SR поміщається в стек.
4. Якщо за час виконання останньої команди було сформовано кілька запитів на переривання, то вибирається переривання з найбільшим пріоритетом.
5. Якщо переривання має одне джерело, то прапорець переривання автоматично скидається. Якщо переривання може генеруватися кількома джерелами то прапорці переривання залишаються встановленими для подальшої обробки в програмі.
6. Регістр стану SR очищається. У результаті процесор переходить з режиму зниженого споживання в активний режим. Оскільки біт GIE скидається, наступні переривання забороняються.
7. Вміст вектора переривання завантажується в лічильник команд PC і починається виконання процедури обробки переривання, розташованої за цією адресою.

Повернення з переривання

Процедура обробки переривання завжди завершується командою: RETI (повернення з процедури обробки переривання) Для повернення з переривання потрібно 5 (MSP430) тактів CPU, необхідних для виконання таких дій (рис. 3.6):

1. Відновлення вмісту регістра SR зі стека. У результаті вступають в дію всі попередні установки бітів GIE, CPUOFF тощо, незалежно від їх установок, що використовувалися в процедурі обробки переривання.

2. Вміст лічильника команд PC витягується зі стека, і виконання програми продовжується з того місця, де вона була перервана.

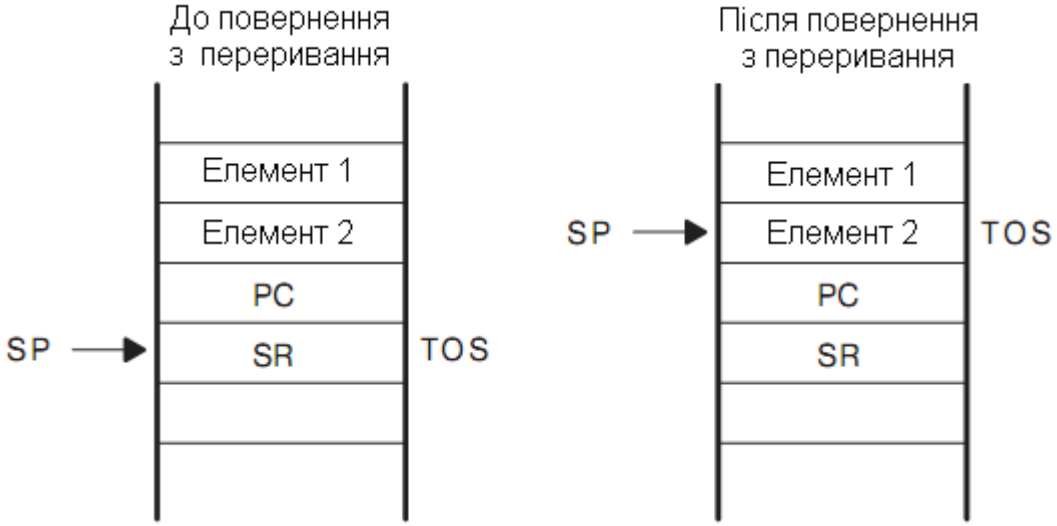


Рисунок 3.6 – Повернення з переривання

Вкладені переривання

Вкладені переривання дозволяються установкою біта GIE у процедурі обробки переривання. При цьому будь-яке переривання, що виникло під час виконання процедури обробки переривання, перерве її виконання, незалежно від пріоритетів переривання, що обслуговується та нового переривання.

Вектори переривань

Вектори переривань і вектор скидання мікроконтролера MSP430F2013 розташовуються в діапазоні адрес 0FFFh ... 0FFC0h, як показано в табл. 4.1. Кожен вектор програмується користувачем шляхом запису в нього 16-ти бітної адреси відповідної процедури обробки переривання. Рекомендується передбачати процедури обробки переривань для всіх переривань, реалізованих в конкретному мікроконтролері. При цьому всі не використані в програмі вектори можуть вказувати на порожній обробник переривань, що містить єдину команду RETI. Незадіяні вектори переривань при необхідності можна використовувати для розміщення програмного коду. Деякі біти включення периферійних модулів, біти дозволу переривань і прапорці переривань знаходяться в регістрах спеціальних функцій SFR. Ці регістри є 8-ми бітними і розташовуються в молодших адресах адресного простору. Звертатися до регістрів спеціальних функцій необхідно за допомогою команд, що працюють з однобайтними операндами. Конфігурація області SFR наводиться в довідковій документації на конкретні моделі.

В табл. 3.1 наведені джерела, прапорці та вектори переривань мікроконтролера MSP430F2013.

Таблиця 3.1 – Джерела, прапорці та вектори переривань мікроконтролера MSP430F2013

Джерело переривання	Прапорець переривання	Системне переривання	Адреса	Пріоритет
Вимкнення живлення, зовнішнє скидання, сторожовий таймер, пароль флеш-пам'яті, вибірка команди по некоректній адресі	PORIGF RSTIFG WDTIFG KEYV	Скидання	0FFFEh	31
NMI-переривання, несправність генератора, порушення доступу до флеш-пам'яті	NMIIFG	Немасковане	0FFFCh	30
	OFIFG	Немасковане		
	ACCVIFG	Немасковане		
Сторожовий таймер	WDTIFG	Масковане	0FFF4h	26
Таймер A	TACCR1 CCIFG TAIFG	Масковане	0FFF0h	24
SD16_A	SD16CCTL0 SD16OVIFG SD16IFG	Масковане	0FFEAh	21
USI	USIIFG USISTTIFG	Масковане	0FFE8h	20
Порт I/O P2	P2IFG.6 - P2IFG.7	Масковане	0FFE6h	19
Порт I/O P1	P2IFG.0 - P2IFG.7	Масковане	0FFE4h	18

Режими роботи мікроконтролера

Сімейство MSP430 було розроблено для додатків з наднизьким живленням, тому мікроконтролери сімейства мають різні режими роботи (рис. 3.7).

При реалізації цих режимів враховувалися наступні вимоги:

- наднизьке споживання;
- швидкість *(speed)* і пропускна здатність;
- мінімізація струму споживання окремих периферійних модулів.

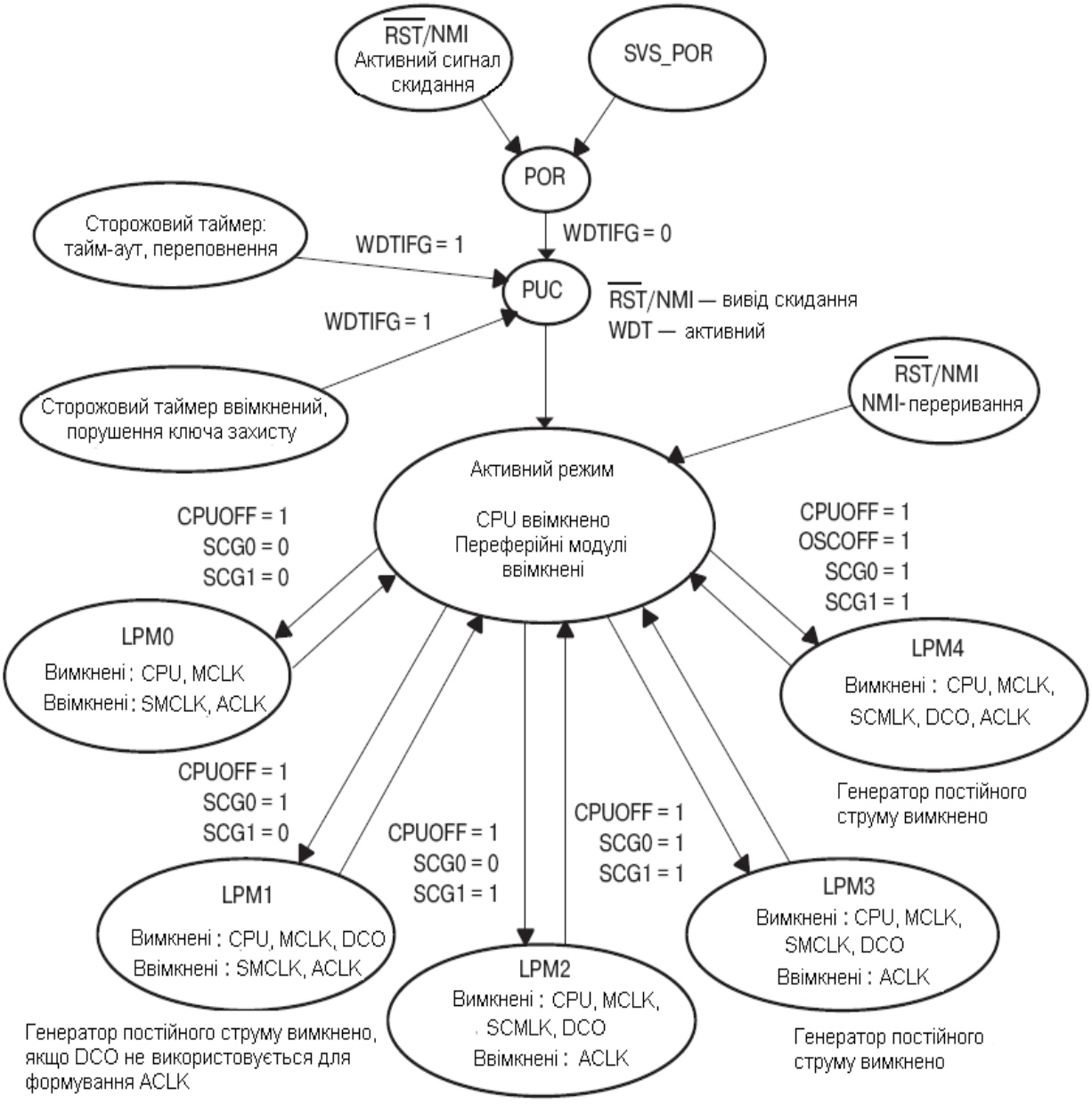


Рисунок 3.7 – Режими роботи мікроконтролера MSP430F2013

Режими зниженого енергоспоживання (LPM0 – LPM4) конфігуруються за допомогою бітів CPUOFF, OSCOFF, SCG0 і SCG1 регістра стану SR. Перевага розміщення бітів управління режимом роботи в регістрі стану полягає в тому, що поточний режим роботи зберігається в стеку на час виконання процедури обробки переривання. Якщо в обробнику збережене значення SR не змінювалося, то після завершення процедури обробки переривання попередній режим роботи відновлюється. Виконання програми може продовжитися і в іншому режимі, якщо процедура обробки переривання змінить значення регістра стану, що знаходиться в стеку. Звернення до бітів управління режимом роботи та до стека може проводитися за допомогою команди будь-якого типу.

При установці будь-якого з бітів управління режимом мікроконтролер відразу ж переключається в обраний режим. При відключенні будь-якого тактового сигналу *(clock signal)* також блокується робота периферійних пристроїв, які його використовують. Периферійні пристрої можуть відключатися і окремо за допомогою відповідних регістрів управління. Стан портів введення/виведення і вміст ОЗП і регістрів при зміні режиму роботи залишаються незмінними. Вихід з режиму зниженого споживання можливий при будь-якому дозволеному перериванні.

Вхід в режими зниженого енергоспоживання та вихід з них

Типове споживання струму мікроконтролерами сімейства MSP430 показане на рис. 3.8.

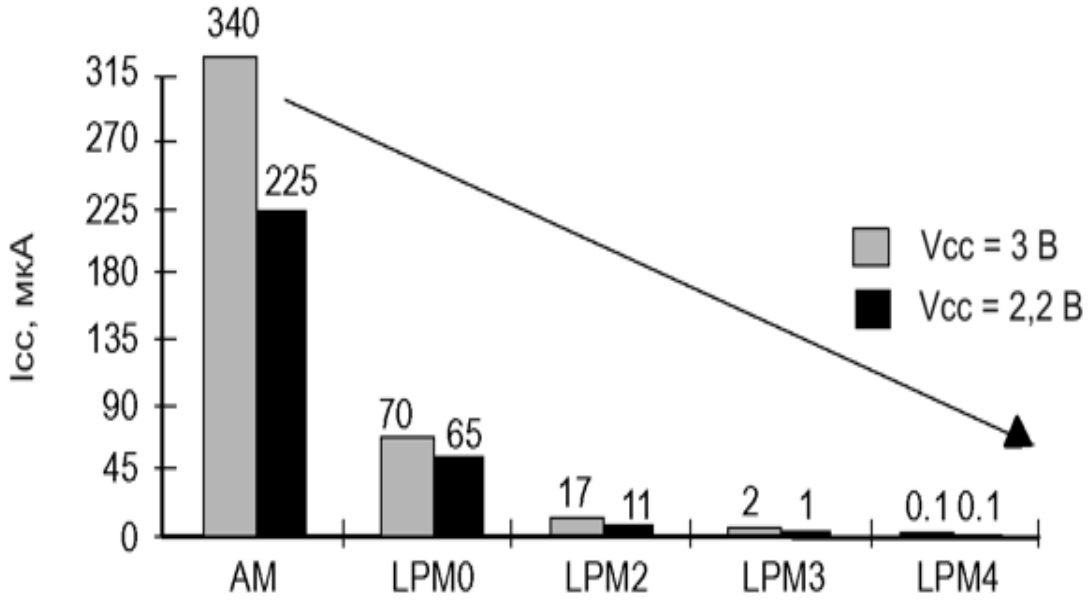


Рисунок 3.8 – Типове споживання струму мікроконтролерами сімейства MSP430 в залежності від режиму роботи

Вихід з будь-якого режиму зниженого споживання проводиться при появі якого-небудь дозволеного переривання. При цьому виконуються наступні дії:

- вхід в процедуру обробки переривання:
  - лічильник команд PC і регістр стану SR зберігаються в стеку;
  - біти CPUOFF, SCG1 і OSCOFF автоматично скидаються.
- варіанти повернення з процедури обробки переривання:
  - вихідне значення регістру SR витягується зі стека, відновлюючи попередній режим роботи;
  - біти регістра SR, збережені в стеку, можуть бути змінені у процедурі обробки переривання. У цьому випадку при виконанні команди RETI мікроконтролер перемкнеться в інший режим роботи.

Зміст завдання

1. Лабораторна робота присвячена ознайомленню з системою переривань та режимами роботи мікроконтролера MSP430F2013.

2. Завдання містить код програми, який необхідно відкомпілювати, зневадити та запустити на виконання.

3. При виконанні завдання необхідно змінювати значення регістру TACCR0.

Код програми

```
#include <msp430x20x3.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */

  P1DIR |= 0x01;// налаштування P1.0 на виведення

  CCTL0 = CCIE; // дозвіл переривання CCR0

  CCR0 = 30000;

  TACTL = TASSEL_2 + MC_1;// SMCLK, режим «вгору»

  _BIS_SR(LPM0_bits + GIE);/* перехід в режим
                              LPM0 і очікування
                              переривання */
}


#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P1OUT ^= 0x01;  // перемикання P1.0
}
```

Опис програми

У програмі використовується Timer_A, який налаштований на режим прямого рахунку. В цьому режимі Timer_A циклічно рахує до тих пір, поки його значення не стане рівним вмісту регістру порівняння TACCR0. Коли значення таймера досягає величини, яка записана в регістрі TACCR0, таймер здійснює скидання і рахунок починається з нуля. Прапорець переривання CCIFG, відповідає регістру TACCR0, встановлюється при досягненні таймером в процесі рахунку значення, записаного в цьому регістрі. Прапо-

рець переривання TAIFG встановлюється при рахуванні таймера від значення, яке міститься в регістрі TACCR0, до нуля.

Переривання генерується у випадку переповнення регістру лічильника таймера, у функції обробнику переривання здійснюється перемикання P1.0.

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп'ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми.
4. Здійсніть компіляцію та зневадження проекту.
5. Запустіть проект на виконання.
6. Занесіть у звіт значення регістру TACCR0.
7. Змініть у коді програми значення регістру TACCR0 на 60000:

```
CCR0 = 60000;
```

8. Повторіть дії описані в пунктах 4-6.
9. Порівняйте результати роботи програми при значеннях регістру TACCR0 = 30000 та TACCR0 = 60000 та занесіть їх у звіт.
10. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- лістинг програми;
- результати виконання;
- висновки.

Контрольні запитання

1. Що таке переривання?
2. Які підходи використовуються для відстеження подій і вчасної їх обробки?
3. Які типи переривань є у MSP430?
4. Якими подіями викликається немасковане переривання?
5. Як конфігурується Вивід RST / NMI?
6. Що відбувається при порушенні доступу до флеш-пам'яті?
7. Як генеруються масковані переривання?
8. Як відбувається обробка переривання?
9. В якій послідовності проводиться обробка запиту на переривання?
10. Як відбувається повернення з переривання?
11. Що таке вектори переривань і як вони використовуються?
12. У яких режимах працює мікроконтролер?

3.3 Лабораторна робота №3. Дослідження портів введення/ виведення

Мета роботи: Дослідити порти введення/виведення загального призначення мікроконтролера MSP430F2013, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Мікроконтролер MSP430F2013 містить 2 порти введення/виведення P1 та P2: 10 виводів – порт P1 (8 біт) і порт P2 (2 біти) (рис. 3.9).

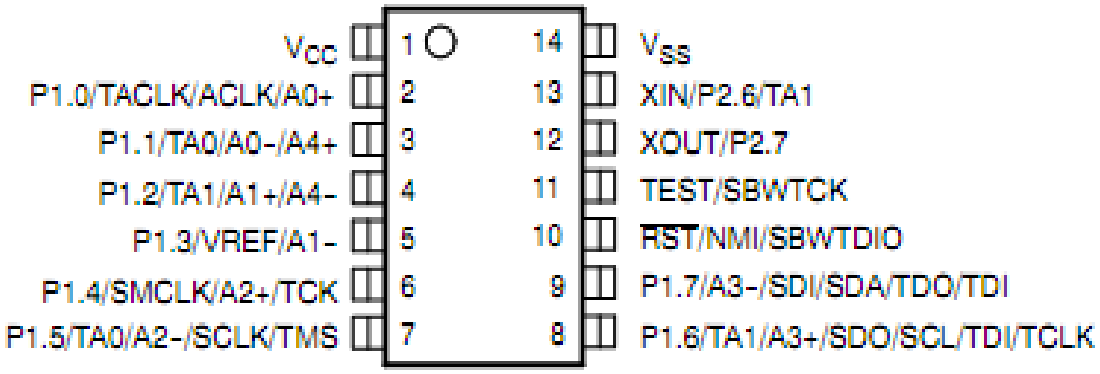


Рисунок 3.9 – Мікросхема мікроконтролера MSP430F2013

В табл. 3.2 наведені призначення та опис виводів портів введення/виведення P1 та P2 мікроконтролера MSP430F2013.

Порти P1 і P2 підтримують зовнішні переривання. Для кожного з виводів портів P1 і P2 можна індивідуально дозволити переривання і налаштувати таким чином, щоб переривання генерувалося по наростаючому чи спадаючому фронту вхідного сигналу. Всі виводи порту P1 призначені одному вектору переривань, а всі виводи порту P2 – іншому вектору.

Цифрові порти введення/виведення мають такі особливості:

- незалежні індивідуально програмовані виводи;
- будь-які комбінації входів та виходів;
- індивідуальне налаштування переривань від виводів портів P1 та P2;
- окремі регістри даних для входів і виходів;
- індивідуальне налаштування резисторів змінного опору.

Окрім того виводи порту можуть бути індивідуально налаштовані в якості спеціальних функцій введення/виведення (див. табл. 3.2)

Таблиця 3.2 – Призначення та опис портів введення/виведення P1 та P2 мікроконтролера MSP430F2013

Вивід	Номер виводу	I/O	Опис
P1.0/ TACLK/ ACLK/ A0+	2	I/O	Цифровий вивід *(digital pin)* I/O загального призначення; Timer_A: вхідний сигнал синхронізації TACLK; Вихідний сигнал ACLK; SD16_A: позитивний аналоговий вхід *(analog input)* A0.
P1.1/ TA0/ A0–/ A4+/	3	I/O	Цифровий вивід I/O загального призначення; Timer_A: режим захоплення *(capture mode)* – вхід CCI0A, режим порівняння *(compare mode)* – вихід Out0; SD16_A: негативний аналоговий вхід A0; SD16_A: позитивний аналоговий вхід A4.
P1.2/ TA1/ A1+/ A4–	4	I/O	Цифровий вивід I/O загального призначення; Timer_A: режим захоплення – вхід CCI1A, режим порівняння – вихід Out1; SD16_A: позитивний аналоговий вхід A1; SD16_A: негативний аналоговий вхід A4.
P1.3/ VREF/ A1–	5	I/O	Цифровий вивід I/O загального призначення; Вхід для зовнішньої опорної напруги *(reference voltage)*/вихід для внутрішнього джерела опорної напруги; SD16_A: негативний аналоговий вхід A1.
P1.4/ SMCLK/ A2+/ TCK	6	I/O	Цифровий вивід I/O загального призначення; Вихідний сигнал SMCLK; SD16_A: позитивний аналоговий вхід A2; Тестування синхронізації JTAG, вхід для пристроїв програмування і тестування.
P1.5/ TA0/ A2–/ SCLK/ TMS	7	I/O	Цифровий вивід I/O загального призначення; Timer_A: режим порівняння - вихід Out0; SD16_A: негативний аналоговий вхід A2; USI: вхід зовнішньої синхронізації у режимі SPI чи I2C, вихід синхронізації у режимі SPI; Вибір режиму тестування JTAG, вхід для пристроїв програмування і тестування.

Продовження таблиці 3.2.

Вивід	Номер виводу	I/O	Опис
P1.6/ TA1/ A3+/ SDO/ SCL/ TDI/ TCLK	8	I/O	Цифровий вивід I/O загального призначення; Timer_A: режим захоплення – вхід CCI1B, режим порівняння – вихід Out1; SD16_A: позитивний аналоговий вхід A3; USI: виведення даних у режимі SPI; Синхронізація I2C у режимі I2C; Вхід JTAG тестування даних; Вхід тестування синхронізації під час програмування і тестування.
P1.7/ A3–/ SDI/ SDA/ TDO/ TDI	9	I/O	Цифровий вивід I/O загального призначення; SD16_A: негативний аналоговий вхід A3; USI: введення даних у режимі SPI, введення даних I2C у режимі I2C; Вихід для JTAG тестування даних; Вхід тестування даних під час програмування і тестування.
XIN/ P2.6/ TA1	13	I/O	Вхід для кварцового генератора; Цифровий вивід I/O загального призначення; Timer_A: режим порівняння – вихід Out1.
XOUT/ P2.7	12	I/O	Вихід для кварцового генератора; Цифровий вивід I/O загального призначення.
$\overline{\text{RST}}$/ NMI/ SBWTDIO	10	I	Скидання немаскованого переривання; Вхід немаскованого переривання; I/O Spy-Bi-Wire тестування даних під час програмування і тестування.
TEST/ SBWTCK	11	I	Вибір тестового режиму для виводів JTAG порту P1. Запобіжник підключений до TEST. Вхід тестування синхронізації Spy-Bi-Wire під час програмування і тестування
V_{CC}	1		Напруга живлення *(supply voltage)*
V_{SS}	14		Заземлення

Регістри

Конфігурація роботи цифрових портів введення/виведення визначається в програмному забезпечені, використовуючи наступні регістри.

Регістр напрямку PxDIR

Значення кожного біту регістра PxDIR визначає напрямок передачі даних відповідного виводу порту, незалежно від обраної для цього виводу функції. Якщо вивід порту використовується яким-небудь периферійним модулем, то біт регістру PxDIR має бути встановлений у відповідності з вимогами даного модуля.

Конфігурація PxDIR:

- 1 – вивід порту встановлено на виведення;
- 0 – вивід порту встановлено на введення.

Регістр введення PxIN

Кожен біт регістра PxIN, доступний лише в режимі зчитування, відображає рівень вхідного сигналу на відповідному виводі порту, якщо цей вивід налаштований в якості введення/виведення загального призначення.

Конфігурація PxIN:

- 1 – високий рівень вхідного сигналу;
- 0 – низький рівень вхідного сигналу.

Регістр виведення PxOUT

Значення кожного біту регістра PxOUT визначає стан відповідного виводу порту, якщо цей вивід налаштований як цифровий вихід, і не використовується внутрішній резистор змінного опору *(pull-up/pull-down resistor).*

Конфігурація PxOUT:

- 1 – високий рівень вихідного сигналу;
- 0 – низький рівень вихідного сигналу.

Якщо використовується внутрішній резистор змінного опору, то значення біту регістра PxOUT визначає тип «підтяжки» на відповідному виводі порту:

- 1 – вивід підтягується до живлення;
- 0 – вивід підтягується до загального проводу.

Регістр дозволу резисторів змінного опору PxREN

Кожен біт регістра дозволяє включити або відключити резистор змінного опору відповідного виводу порту.

Конфігурація PxREN:

- 1 – резистор змінного опору включений;
- 0 – резистор змінного опору відключений.

Регістри вибору функції PxSEL, PxSEL2

Більшість виводів портів використовуються різними периферійними модулями. Альтернативні функції виводів наведені в табл. 5.1. Кожен з регістрів PxSEL та PxSEL2 використовуються для вибору функції відповідного виводу мікроконтролера.

У випадку налаштування портів P1 і P2 для функції периферійного модуля переривання відключені.

В табл. 3.3 наведена конфігурація портів P1 та P2 комбінацією значень регістрів PxSEL та PxSEL2.

Таблиця 3.3 – Конфігурація портів P1 та P2 комбінацією PxSEL та PxSEL2

PxSEL	PxSEL2	Функція виводу
0	0	Вивід порту введення/виведення
0	1	Вивід основного периферійного модуля
1	0	Зарезервовано
1	1	Вивід додаткового периферійного модуля

Порти переривання P1 і P2

Кожен вивід портів P1 і P2 здатний генерувати запит на переривання. Конфігурація цієї функції здійснюється за допомогою регістрів PxIFG, PxIE і PxIES. Всі виводи порту P1 зв’язані з одним вектором переривання, а всі виводи порту P2 – з іншим вектором. Для визначення конкретного джерела переривання від портів P1 чи P2 можна перевірити вміст відповідного регістру PxIFG.

Регістри дозволу переривань PxIE

Кожен біт регістру PxIE дозволяє генерацію переривання при встановленні відповідного прапорця переривання PxIFG.

Конфігурація PxIE:

- 1 – переривання дозволено;
- 0 – переривання заборонено.

Регістри вибору фронту переривання PxIES

Значення кожного біту регістра PxIES визначає по якому фронту сигналу *(signal front)* буде генеруватися переривання від відповідного виводу порту.

Конфігурація PxIES:

- 1 – прапорець переривання встановлюється по спадаючому фронту;
- 0 – прапорець переривання встановлюється на наростаючому фронту.

Регістри цифрових портів введення/виведення

Регістри цифрових портів введення/виведення наведені в табл. 3.4.

Таблиця 3.4 – Регістри цифрових портів введення/виведення

Порт	Регістр	Позначення	Адреса	Тип регістра	Початковий стан
P1	Вхід	P1IN	020h	R/O	-
	Вихід	P1OUT	021h	R/W	Не змінюється
	Напрямок	P1DIR	022h	R/W	Скидається після PUC
	Прапорець переривання	P1IFG	023h	R/W	Скидається після PUC
	Фронт переривання	P1IES	024h	R/W	Не змінюється
	Дозвіл переривання	P1IE	025h	R/W	Скидається після PUC
	Вибір функції	P1SEL	026h	R/W	Скидається після PUC
	Вибір функції 2	P1SEL2	041h	R/W	Скидається після PUC
	Включення резистора	P1REN	027h	R/W	Скидається після PUC
P2	Вхід	P2IN	028h	R/O	-
	Вихід	P2OUT	029h	R/W	Не змінюється
	Напрямок	P2DIR	02Ah	R/W	Скидається після PUC
	Прапорець переривання	P2IFG	02Bh	R/W	Скидається після PUC
	Фронт переривання	P2IES	02Ch	R/W	Не змінюється
	Дозвіл переривання	P2IE	02Dh	R/W	Скидається після PUC
	Вибір функції	P2SEL	02Eh	R/W	Скидається після PUC
	Вибір функції 2	P2SEL2	042h	R/W	Скидається після PUC
	Включення резистора	P2REN	02Fh	R/W	Скидається після PUC

Регістри прапорця переривання PxIFG

Кожен з бітів PxIFGx є прапорцем переривання від відповідного виводу порту і встановлюється при появі на виводі заданого фронту сигналу. Будь-який з прапорців PxIFGx генерує запит переривання, якщо встановлений

відповідний біт регістра PxIE і біт загального дозволу переривань GIE. Всі прапорці PxIFG повинні скидатися програмно. Крім того, будь-який прапорець PxIFG може бути встановлений вручну для програмної генерації переривання.

Конфігурація PxIFG:

- 0 – не було переривання;
- 1 – було переривання.

Переривання генерується тільки по фронту сигналу а не по його рівню. Якщо будь-який з прапорців PxIFGx буде встановлено під час виконання підпрограми обробки переривання від порту Px чи вже після виконання команди RETI даної підпрограми, то такий прапорець визве генерацію нового переривання. Це гарантує обробку всіх змін рівня сигналу.

Конфігурація виводів, які не використовуються

Невикористані виводи мікроконтролера необхідно налаштувати як виходи портів введення/виведення і залишити непідключеними, щоб уникнути появи «плаваючих» входів і знизити струм споживання пристрою. Значення біта PxOUT для такого виводу може бути будь-яким, оскільки вивід нікуди не підключений. В якості альтернативи, щоб уникнути появи «плаваючого» входу, можна до невживаного виводу підключити внутрішній резистор змінного опору, встановивши відповідний біт регістра PxREN.

Зміст завдання

1. Лабораторна робота присвячена ознайомленню з роботою портів введення/виведення загального призначення мікроконтролера MSP430F2013.

2. Завдання містить код програми 1, який необхідно відкомпілювати, зневадити та запустити на виконання.

3. Завдання містить код програми 2, в якому необхідно визначити значення наступних регістрів: P1IE, P1IES, P1IFG. Після цього програму необхідно відкомпілювати, зневадити та запустити на виконання.

Код програми 1

```
#include  <msp430x20x3.h>

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* Зупинка сторожового
                              таймера */
  P1DIR |= 0x01;/* Налаштування порту P1.0 на
                виведення */
  while (1) {      // Тестування P1.4
    if ((0x10 & P1IN)) P1OUT |= 0x01;
    else P1OUT &= ~0x01;
  }
 }
```

Код програми 2

```
#include  <msp430x20x3.h>

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */
  P1DIR |= 0x01;// налаштування P1.0 на виведення
  P1IE _______; // дозвіл на переривання для P1.4
  P1IES ______; /* переривання на Hi/lo переході
                 для P1.4 */
  P1IFG = _______; /* очищення біту регістра прапорця
                    переривання IFG для P1.4 */
  _BIS_SR(LPM4_bits + GIE);/* перехід в режим
                              LPM4 і очікування
                              переривання */
}
// Port1 ISR
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){
  P1OUT ^= 0x01; // перемикання P1.0
  P1IFG = _____; /* очищення біту регістра
                  прапорця переривання IFG для
                  P1.4 */
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп’ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми 1.
4. Здійсніть компіляцію та зневадження проекту.
5. Запустіть проект на виконання, перегляньте значення регістрів P1DIR, P1IN, P1OUT та занесіть ці значення у звіт.
6. Створіть новий проект, додайте в нього файл з розширенням ".c" та скопіюйте в нього код програми 2.
7. Код програми є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів, а саме:
   - встановити дозвіл на переривання P1.4:

     ```
     P1IE |= _____;
     ```

   - встановити прапорець переривання регістру P1IES для P1.4 по спадаючому фронту сигналу:

     ```
     P1IES ______;
     ```

○ здійсніть очищення біту регістру прапорця переривання IFG для P1.4

```
P1IFG = ________;
```

8. Здійсніть компіляцію та зневадження проекту.

9. Запустіть проект на виконання, перегляньте значення регістрів P1IE, P1IES, P1IFG, P1OUT та P1DIR та занесіть ці значення у звіт.

10. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- лістинги програм;
- результати виконання;
- висновки.

Контрольні запитання

1. Скільки портів містить мікроконтролер MSP430F2013?
2. Чи підтримують порти P1 і P2 зовнішні переривання?
3. Які можливості мають цифрові порти введення/виведення?
4. Призначення та опис портів введення/виведення P1 мікроконтролера MSP430F2013.
5. Призначення та опис портів введення/виведення P2 мікроконтролера MSP430F2013.
6. Які регістри використовує конфігурація роботи порту визначена в програмному забезпечені?
7. Основна характеристика та конфігурація регістрів напрямку (PxDIR).
8. Основна характеристика та конфігурація регістрів введення (PxIN).
9. Основна характеристика та конфігурація регістрів виведення (PxOUT).
10. Основна характеристика та конфігурація регістрів дозволу резисторів змінного опору (PxREN).
11. Основна характеристика та конфігурація регістрів вибору функції: (PxSEL) and (PxSEL2).
12. Дайте коротку характеристику портам переривання P1 та P2.
13. Які регістри використовують порти переривання P1 та P2?
14. Основна характеристика та конфігурація регістрів вибору фронту переривання (PxIES).
15. Основна характеристика та конфігурація регістрів прапорця переривання (PxIFG).
16. Яким чином конфігуруються виводи, які не використовуються?

3.4 Лабораторна робота №4. Дослідження модуля синхронізації BASIC CLOCK

Мета роботи: ознайомлення з режимами роботи модуля синхронізації Basic Clock мікроконтролера MSP430F2013, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Модуль синхронізації Basic Clock дозволяє знизити вартість кінцевої системи і забезпечує наднизьке споживання пристрою. Використовуючи один з трьох тактових сигналів, які формуються модулем, користувач може досягнути оптимального співвідношення продуктивності і енергоспоживання.

Конфігурація модулю синхронізації може бути здійснена програмно для роботи без використання додаткових зовнішніх елементів з одним зовнішнім транзистором *(transistor)* чи з одним або двома зовнішніми кварцевими чи керамічними резонаторами *(resonator).*

Модуль синхронізації містить три джерела тактового сигналу:

• LFXT1CLK – низькочастотний/високочастотний генератор, який може працювати з «часовим» кварцевим резонатором чи зовнішнім сигналом частотою *(signal frequency)* 32768 Гц або зі звичайними кварцевими/керамічними резонаторами чи зовнішнім сигналом синхронізації частотою від 400 кГц до 16 МГц;

• DCOCLK – вбудований генератор з цифровим управлінням;

• VLOCLK – вбудований низькочастотний генератор з наднизьким споживанням, який працює на частоті 12 кГц.

Модуль синхронізації формує три тактових сигнали:

• ACLK – допоміжний тактовий сигнал. Джерело вибирається програмно: LFXT1CLK чи VLOCLK. При використанні дільника *(frequencies divider)* частота сигналу від вибраного джерела зменшується в 1, 2, 4 чи 8 разів. Сигнал ACLK може програмно назначатися для окремих периферійних модулів;

• MCLK – основний тактовий сигнал. Джерело вибирається програмно: LFXT1CLK, VLOCLK чи DCOCLK. При використанні дільника частота сигналу від вибраного джерела зменшується в 1, 2, 4 чи 8 раз. Сигнал MCLK використовується для тактування CPU і системи;

• SMCLK – додатковий тактовий сигнал. Джерело вибирається програмно: LFXT1CLK, VLOCLK чи DCOCLK. При використанні дільника частота сигналу від вибраного джерела зменшується в 1, 2, 4 чи 8 раз. Сигнал SMCLK може програмно назначатися для окремих периферійних модулів.

На рис. 3.10 зображена схема електрична функціональна модуля синхронізації System Clock.

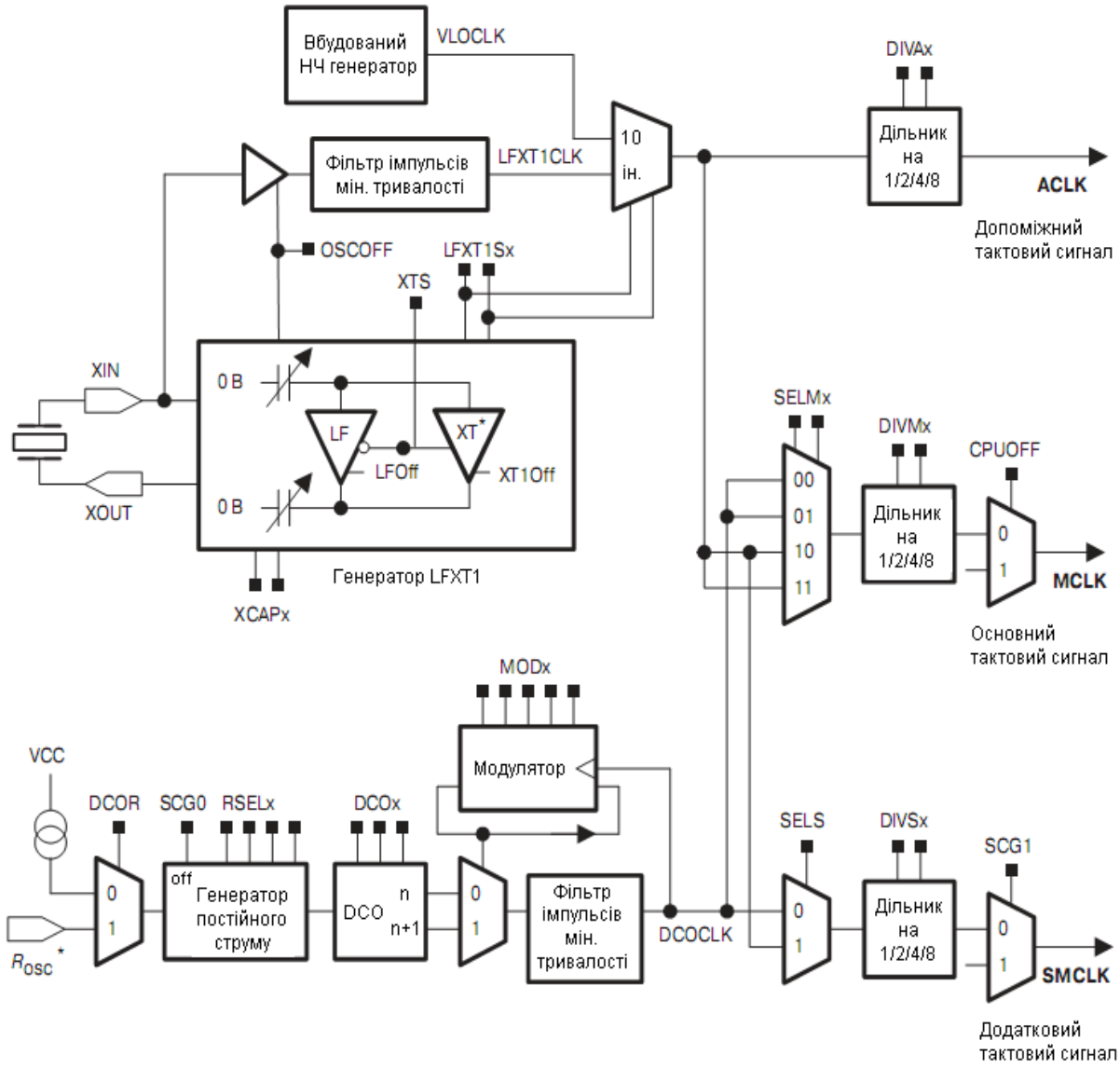


Рисунок 3.10 – Схема електрична функціональна модуля синхронізації System Clock

Функціонування модуля синхронізації

Після сигналу PUC тактові сигнали MCLK і SMCLK формуються з DCOCLK, що має частоту близько 1.1 МГц. Сигнал ACLK формується з LFXT1CLK, при цьому генератор LFXT1 працює в режимі LF з використанням вбудованих конденсаторів (навантажувальна ємність - 6 пФ). Біти управління SCG0, SCG1, OSCOFF і CPUOFF регістра стану визначають режими роботи ядра MSP430 і включають або відключають окремі вузли модуля синхронізації. Конфігурація модуля здійснюється за допомогою регістрів DCOCTL, BCSCTL1, BCSCTL2 і BCSCTL3.

Можливості модуля синхронізації і додатків з наднизьким енергоспоживанням

До системи синхронізації пристроїв з батарейним живленням часто пред'являють суперечливі вимоги:

- наявність тактового сигналу низької частоти для зниження енергоспоживання і реалізації функцій відліку часу;
- наявність тактового сигналу високої частоти для забезпечення швидкої реакції на події та швидкої обробки даних *(data processing)*;
- забезпечення стабільності частоти тактового сигналу при зміні температури і напруги живлення.

Модуль синхронізації Basic Clock розроблений з урахуванням перерахованих вимог і дозволяє користувачеві вибирати будь-який з трьох доступних тактових сигналів: ACLK, MCLK і SMCLK. Для досягнення оптимальної продуктивності при незначному споживанні в якості джерела сигналу ACLK може використовуватися економічний «часовий» кварц частотою 32768 Гц, при цьому забезпечується стабільний синхросигнал і низьке споживання в режимі очікування. Якщо точність формування часових інтервалів не критична, то в якості джерела сигналу ACLK може використовуватися вбудований низькочастотний генератор.

Сигнал MCLK може формуватися вбудованим DCO, який у разі потреби активується обробниками переривань при настанні відповідних подій. Сигнал SMCLK може формуватися кварцевим генератором із зовнішнім резонатором або DCO в залежності від вимог, які пред'являються периферійними модулями. Гнучка система тактування і наявність дільників тактових сигналів забезпечують можливість точного налаштування модуля синхронізації відповідно до вимог конкретного застосування.

Вбудований низькочастотний генератор з наднизьким споживанням

Вбудований низькочастотний генератор з наднизьким споживанням (VLO) працює на частоті 12 кГц без зовнішнього кварцового резонатора. Використання VCOCLK як джерела тактового сигналу задається бітами LFXT1Sx = 10 при XTS = 0. Біт OSCOFF відключає генератор для переходу в режим зниженого енергоспоживання LPM4. При використанні генератора VLO кварцовий генератор LFXT1 відключається, знижуючи тим самим споживання мікроконтролера. Якщо генератор VLO не використовується, то його струм споживання дорівнює нулю.

Генератор LFXT1

Генератор LFXT1 має дуже мале споживання в режимі LF (XT = 0) при використанні «часового» кварцу частотою 32768 Гц. Кварцовий резонатор підключають до виводів XIN і XOUT мікроконтролера без вико-

ристання додаткових елементів. Величина навантажувальної ємності для кварцового резонатора при роботі генератора LFXT1 в режимі LF задається бітами XCAPx, які конфігуруються програмно. Ця ємність може набувати значення з ряду: 1 пФ, 6 пФ, 10 пФ або 12.5 пФ. При необхідності допускається підключення зовнішніх конденсаторів.

У режимі HF (XTS = 1, XCAPx = 00) генератор LFXT1 підтримує використання високочастотних кварцових і керамічних резонаторів. Високочастотний резонатор підключається до виводів XIN і XOUT, при цьому на обох виводах повинні бути присутніми зовнішні навантажувальні конденсатори. Ємність даних конденсаторів вибирається відповідно до специфікації використовуваного резонатора. Біти LFXT1Sx визначають робочий діапазон частот при роботі генератора LFXT1 в режимі HF.

У будь-якому з режимів генератор LFXT1 може працювати із зовнішнім тактовим сигналом, що подається на вхід XIN (LFXT1Sx = 11, OSCOFF = 0, XCAPx = 00). При використанні зовнішнього сигналу його частота повинна відповідати специфікації мікроконтролера для вибраного режиму. Якщо частота зовнішнього сигналу виявиться нижче мінімально допустимої величини, то може бути встановлено біт LFXT1OF, що запобігає тактування CPU сигналом LFXT1CLK. Якщо LFXT1CLK не використовується в якості джерела тактового сигналу SMCLK або MCLK, то генератор LFXT1 може бути відключений установкою біта OSCOFF.

Генератор з цифровим управлінням

DCO є вбудованим генератором з цифровим управлінням. Частота DCO може підлаштовуватися програмно за допомогою бітів DCOx, MODx і RSELx.

Відключення DCO

Якщо DCO не використовується для формування тактового сигналу SMCLK або MCLK в активному режимі мікроконтролера, то він може бути відключений встановленням біта SCG0.

Налаштування частоти DCO

Після сигналу PUC в біти управління DCO завантажуються такі значення (RSELx = 7, DCOx = 3), які забезпечують запуск DCO на середній частоті робочого діапазону. При цьому тактові сигнали MCLK і SMCLK формуються з сигналу DCOCLK. Оскільки CPU тактується сигналом MCLK, а DCO має дуже малий час запуску, програма починає виконуватися менш ніж через 2 мкс після появи сигналу PUC. Типові значення діапазонів і дискретних інтервалів для бітів RSELx і DCOx показані на рис. 3.11.

Частота сигналу DCOCLK встановлюється наступним чином:

• Чотири біта RSELx визначають один із шістнадцяти робочих діапазонів DCO. Значення частот для цих діапазонів наводяться в документації на конкретні моделі.

• Три біта DCOx ділять вибраний діапазон на 8 дискретних значень, які відрізняються один від одного приблизно на 10%.

• П'ять бітів MODx визначають частоту перемикань з поточної частоти, заданої бітами DCOx, на наступну частоту, відповідну значенню DCOx+1. При DCOx = 07h вміст бітів MODx ігнорується, оскільки в цьому випадку DCO вже працює на максимально можливій для вибраного діапазону RSELx частоті.

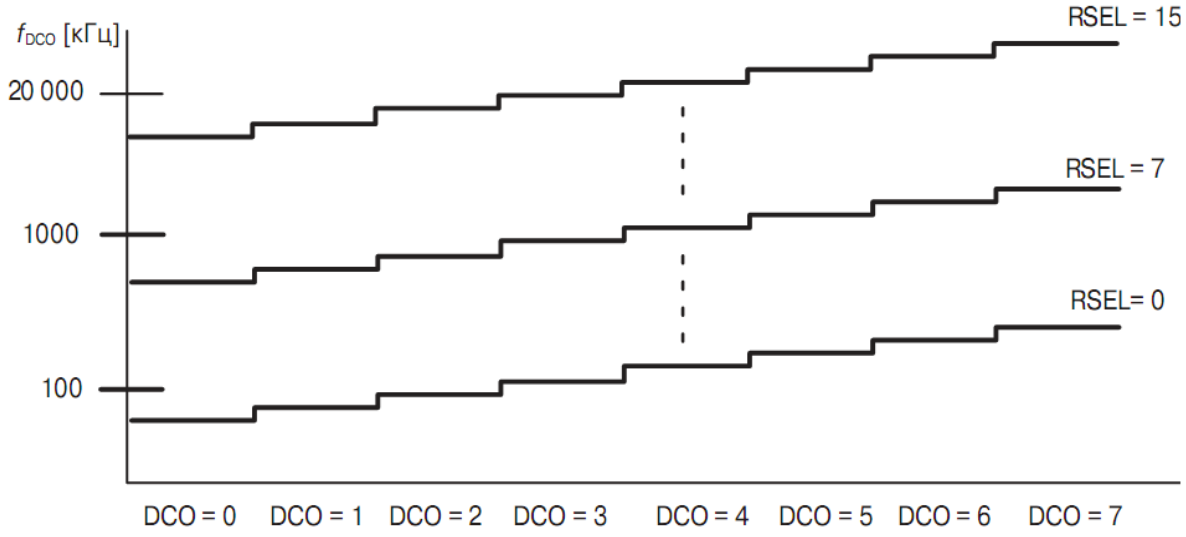


Рисунок 3.11 – Типові значення діапазонів і дискретних інтервалів для бітів RSELx і DCOx

Всі мікроконтролери сімейства MSP430F2xx містять в сегменті *(segment)* A інформаційної секції флеш-пам'яті *(information memory)* значення регістрів DCOCTL і BCSCTL1, відкалібровані для конкретних частот. Щоб використовувати калібровані значення, їх копіюють з флеш-пам'яті у відповідні регістри. ці значення визначають стани бітів DCOx, MODx, RSELx і скидають всі інші біти регістрів, за винятком біта XT2OFF, який встановлюється в 1. Інші біти регістра BCSCTL1 можуть бути індивідуально встановлені або скинуті.

Використання кварцевого генератора для формування MCLK

Після скидання при вмиканні живлення (PUC) тактовий сигнал MCLK формується з сигналу DCOCLK. При необхідності для формування MCLK можна використати генератор LFXT1. Для перемикання тактового сигналу MCLK з DCOCLK на сигнал від кварцевого генератора (LFXT1CLK чи XT2CLK) необхідно виконати наступні дії:

1. Ввімкнути генератор і вибрати необхідний режим роботи.
2. Скинути прапорець OFIFG.
3. Почекати не менше 50 мкс.

4. Перевірити прапорець OFIFG – якщо він встановлений, то повторити етапи 1…4.

В табл. 3.5 наведені регістри модуля Basic Clock.

Таблиця 3.5 – Регістри модуля Basic Clock

Регістр	Позначення	Тип регістру	Адреса	Вихідний стан
Регістр управління DCO	DCOCTL	R/W	056h	060h після PUC
Регістр управління 1 модуля синхронізації	BCSCTL1	R/W	057h	087h після POR
Регістр управління 2 модуля синхронізації	BCSCTL2	R/W	058h	Скидається після PUC
Регістр управління 3 модуля синхронізації	BCSCTL3	R/W	053h	005h після PUC
Регістр дозволу переривань 1	IE1	R/W	0000h	Скидається після PUC
Регістр прапорців переривань 1	IFG1	R/W	0002h	Скидається після PUC

В табл. 3.6 наведені біти регістру управління 1 модуля синхронізації BCSCTL1.

Таблиця 3.6 – Регістр управління 1 модуля синхронізації BCSCTL1

Біти	Позначення	Опис
7	XT2OFF	Вмикання XT2. Цей біт керує роботою генератора XT2: 0 – XT2 ввімкнено 1 – XT2 вимкнено і не використовується для формування MCLK чи SMCLK
6	XTS	Режим LFXT1. Цей біт визначає режим роботи генератора LFXT1: 0 – низькочастотний режим (LF) 1 – високочастотний режим (HF)
5..4	DIVAx	Дільник для ACLK: 00 – /1 01 – /2 10 – /4 11 – /8
3..0	RSELx	Вибір діапазону. Доступно 16 частотних діапазонів. Нижчому діапазону відповідає значення RSELx = 0. Біт RSEL3 ігнорується при DCOR = 1

В табл. 3.7 наведені біти регістру управління DCOCTL.

Таблиця 3.7 – Регістр управління DCOCTL

Біти	Позначення	Опис
7..5	DCOx	Вибір частоти DCO. Ці біти визначають одно з восьми дискретних значень частоти в межах вибраного діапазону, що задається бітами RSELx
4..0	MODx	Налаштування модулятора. Ці біти визначають, як часто сигнал частоти f_{DCO+1} буде використовуватися на протязі періоду, рівного 32 тактам DCOCLK. В час що залишився (MOD - 32) буде використовуватися сигнал частотою f_{DCO}. Ці біти не використовуються про DCOx = 7

В табл. 3.8 наведені біти регістру управління 2 модуля синхронізації BCSCTL2.

Таблиця 3.8 – Регістр управління 2 модуля синхронізації BCSCTL2

Біти	Позначення	Опис
7..6	SELMx	Вибір MCLK. Ці біти визначають джерело тактового сигналу MCLK: 00 – DCOCLK 01 – DCOCLK 10 – LFXT1CLK або VLOCLK 11 – LFXT1CLK або VLOCLK
5..4	DIVMx	Дільник для MCLK: 00 – /1 01 – /2 10 – /4 11 – /8
3	SELS	Вибір SMCLK: 0 – DCOCLK 1 – LFXT1CLK або VLOCLK
2..1	DIVSx	Дільник для SMCLK: 00 – /1 01 – /2 10 – /4 11 – /8
0	DCOR	Вибір резистора DCO: 0 – внутрішній резистор 1 – зовнішній резистор

В табл. 3.9 наведені біти регістру управління 3 модуля синхронізації BCSCTL3.

Таблиця 3.9 – Регістр управління 3 модуля синхронізації BCSCTL3

Біти	Позначення	Опис
7..6	XT2Sx	Вибір діапазону XT2. Ці біти визначають частотний діапазон генератора XT2: 00 – кварцевий/керамічний резонатор 0,4..1 МГц 01 – кварцевий/керамічний резонатор 1..3 МГц 10 – кварцевий/керамічний резонатор 3..16 МГц 11 – зовнішний сигнал частотою 0,4..16 МГц
5..4	LFXT1Sx	Вибір низькочастотного джерела тактового сигналу LFXT1. При XTS = 0 ці біти визначають джерело тактового сигналу (LFXT1 чи VLO). XTS = 0: 00 – кварцовий резонатор 32768 Гц в генераторі LFXT1 01 – зарезервовано 10 – VLOCLK 11 – зовнішній сигнал синхронізації
3..2	XCAPx	Вибір навантажувальної ємності. При XTS = 0 ці біти визначають величину ємності для резонатора, який підключається до генератора LFXT1. Якшо LFXT1Sx = 11, то біти XCAPx повинні бути рівні 00. 00 – 1 пФ 01 – 6 пФ 10 – 10 пФ 11 – 12,5 пФ
1	XT2OF	Несправність генератора XT2: 0 – генератор функціонує нормально 1 – виявлено збій генератора
0	LFXT1OF	Несправність генератора LFXT1: 0 – генератор функціонує нормально 1 – виявлено збій генератора

Зміст завдання

1. Лабораторна робота присвячена дослідженню модуля синхронізації Basic Clock.

2. Завдання містить код програми 1, в якому необхідно визначити значення наступних регістрів: BCSCTL1, DCOCTL, BCSCTL2. Після цього код необхідно відкомпілювати, зневадити та запустити на виконання.

3. Завдання містить код програми 2, в якому необхідно визначити значення наступних регістрів: BCSCTL3, BCSCTL2. Після цього код необхідно відкомпілювати, зневадити та запустити на виконання.

Код програми 1

```
#include  <msp430x20x3.h>

void main(void)
{
  volatile unsigned int i = 0;

  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */
   P1DIR |= 0x01;/* налаштування порту P1.0 на
                  виведення */
   BCSCTL1 |= _______; /* налаштування діапазону
                        DCO на ~100 кГц */
   DCOCTL |= ________; /* вибір частоти
                        DCO = 100кГц */
   BCSCTL2 |= _______; /* DCO – джерело для MCLK,
                        ділення на 8 */
  for (;;)
  {
    P1OUT ^= 0x01; // перемикання порту P1.0
    i = 6250;      // затримка
     do
    {
      (i--);
    }
    while (i != 0);
  }
}
```

Код програми 2

```
#include  <msp430x20x3.h>

void main(void){
  volatile unsigned int i;
```

```
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                           таймера */
  BCSCTL3 |= _________;    // LFXT1 = VLO
  IFG1 &= ~OFIFG; // очищення прапорця OSCFault
  __bis_SR_register(SCG1 + SCG0);// Зупинка DCO
  BCSCTL2 |= _________;          // MCLK = LFXT1/8
  P1DIR = 0xFF;   // всі P1.x на вихід
  P1OUT = 0;    // скинути всі P1.x
  P2DIR = 0xFF; // всі P1.x на вихід
  P2OUT = 0;    // скинути всі P1.x

  for (;;)
  {
    P1OUT |= 0x01;  // встановлення P1.0
    for (i = 10; i > 0; i--);  // затримка 1x
    P1OUT &= ~0x01;            // скидання P1.0
    for (i = 1000; i > 0; i--);// затримка 100x
  }
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп’ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми 1.
4. Код програми 1 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів, а саме:
   - налаштуйте діапазон DCO на ~ 100 кГц:

     ```
     BCSCTL1 |= ________;
     ```
   - виберіть частоту DCO = 100 кГц:

     ```
     DCOCTL |= _________;
     ```
   - виберіть DCOCLK у якості джерела для MCLK та встановіть дільник частоти на 8:

     ```
     BCSCTL2 |= ________;
     ```

5. Здійсніть компіляцію та зневадження проекту.
6. Запустіть проект на виконання, перегляньте значення регістрів BCSCTL1, DCOCTL, BCSCTL2 занесіть ці значення у звіт.
7. Створіть новий проект, додайте в нього файл з розширенням ".c" та скопіюйте в нього код програми 2.

8. Код програми 2 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів:

○ виберіть у якості джерела низькочастотного тактового сигналу для LFXT1 VLOCLK:

```
BCSCTL3 |= __________;
```

○ виберіть VLOCLK у якості джерела для MCLK та встановіть дільник частоти на 8:

```
BCSCTL2 |= __________;
```

9. Здійсніть компіляцію та зневадження проекту.

10. Запустіть проект на виконання, перегляньте значення регістрів BCSCTL3, BCSCTL2 занесіть ці значення у звіт.

11. Порівняйте різні режими синхронізації, та опишіть роботу пристрою в різних режимах.

12. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- схема функціональна модуля синхронізації System Clock;
- лістинги програм;
- результати виконання;
- опис роботи пристрою в різних режимах синхронізації;
- висновки.

Контрольні запитання

1. Для чого призначений модуль синхронізації Basic Clock?
2. Які джерела тактового сигналу містить модуль синхронізації Basic Clock?
3. Які тактові сигнали формує модуль синхронізації Basic Clock?
4. Яким чином відбувається функціонування модуля синхронізації?
5. Які вимогу пред'являють до модуля синхронізації і додатків з над-низьким енергоспоживанням?
6. Як працює вбудований НЧ генератор з наднизьким споживанням?
7. Як працює генератор LFXT1?
8. Як працює генератор з цифровим управлінням (DCO)?
9. Як використовується кварцовий генератор для формування MCLK?
10. Які Ви знаєте регістри модуля Basic Clock?

3.5 Лабораторна робота №5. Дослідження роботи сторожового таймера

Мета роботи: ознайомлення з принципами роботи сторожового таймера мікроконтролера MSP430F2013, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Основна функція модуля WDT+ полягає у виконанні керованого перезапуску системи при некоректному функціонуванні програми. При «зависанні» програми на час, що перевищує заданий інтервал, модуль генерує сигнал системного скидання. Якщо сторожовий таймер в додатку не потрібно, то модуль може використовуватися в якості звичайного інтервального таймера, що генерує переривання із заданою періодичністю. Модуль вдосконаленого сторожового таймера WDT + має такі особливості:

- чотири часових інтервали, обраних програмно;
- режим сторожового таймера;
- режим інтервального таймера;
- парольний захист доступу до регістру управління модуля;
- управління функціонуванням виведення RST/$\overline{\text{NMI}}$;
- можливість вибору джерела тактового сигналу;
- можливість зупину для зменшення струму споживання;
- захист від зникнення тактового сигналу.

Функціонування сторожового таймера

Модуль WDT+ може бути налаштований для роботи в режимі сторожового або інтервального таймера за допомогою регістра WDTCTL. Крім того, в регістрі WDTCTL містяться біти, що визначають конфігурацію виведення RST/NMI. Регістр WDTCTL – це захищений паролем 16-ти бітний регістр, доступний як для читання, так і для запису. Будь-які звернення до даного регістру повинні проводитися з використанням команд, що оперують 2-х байтними операндами, причому при операціях запису в старшому байті записуваного значення має міститися число 0A5h. Запис в регістр WDTCTL значення, старший байт якого не дорівнює 0A5h, викличе порушення ключа захисту з наступним формуванням сигналу скидання PUC, незалежно від режиму роботи модуля. При читанні регістра WDTCTL в старшому байті повертається значення 069h. Частота тактового сигналу сторожового таймера повинна бути менше або дорівнює частоті системного тактового сигналу (MCLK).

На рис. 3.12 наведена схема електрична функціональна сторожового таймера.

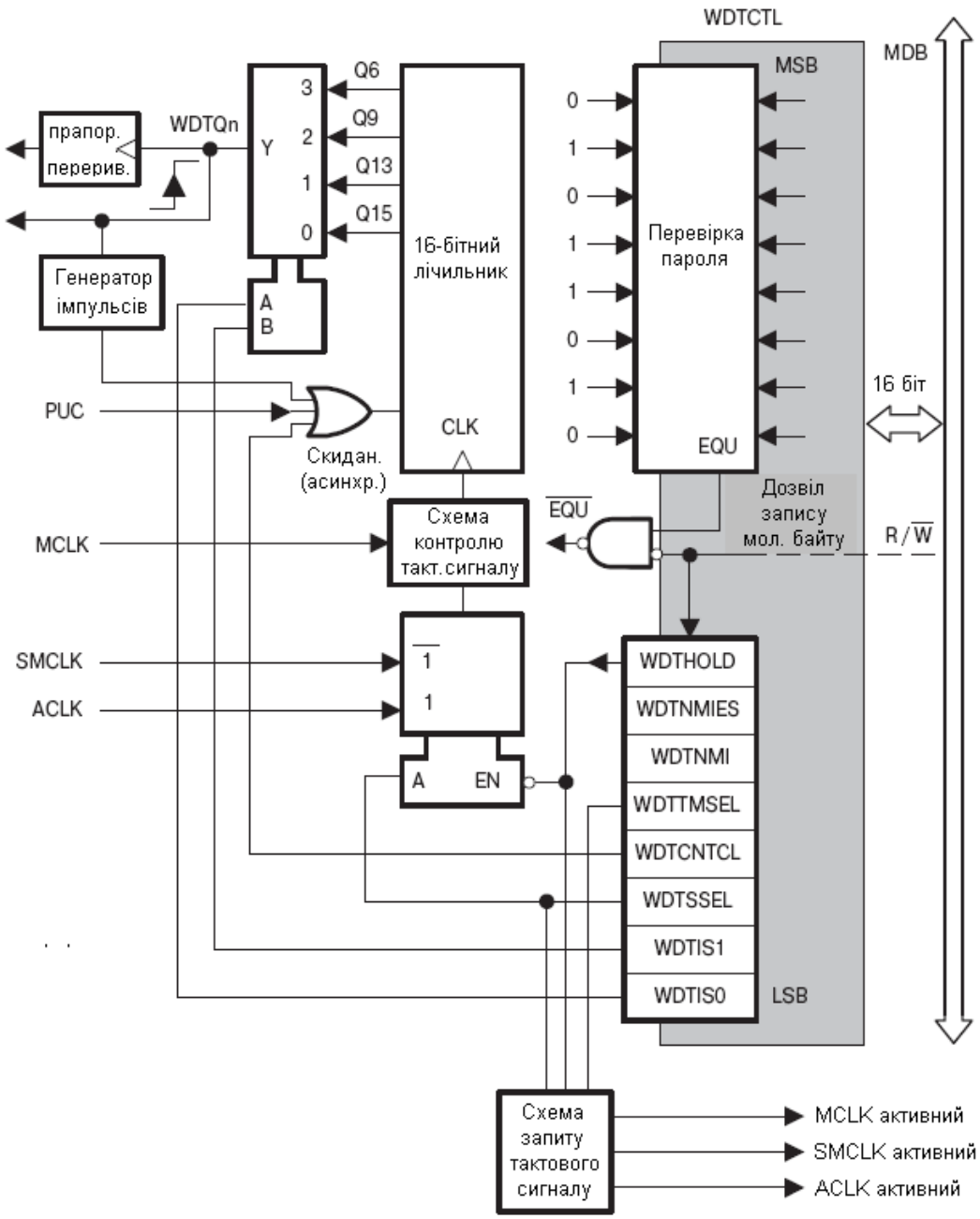


Рисунок 3.12 – Схема електрична функціональна сторожового таймера

Лічильник сторожового таймера

Лічильник сторожового таймера WDTCNT являє собою 16-ти бітний сумуючий лічильник, який безпосередньо з програми не доступний. Управління лічильником, в тому числі завдання часових інтервалів, здійснюється за допомогою регістра керування WDTCTL. Лічильник WDTCNT може тактуватися від сигналів ACLK або SMCLK джерел тактового сигналу задається бітом WDTSSEL.

Режим сторожового таймера

Після скидання при включенні живлення модуль WDT+ починає працювати в режимі сторожового таймера з інтервалом скидання, рівним

32768 тактам сигналу DCOCLK. Користувач повинен повторно налаштувати, зупинити або скинути сторожовий таймер до закінчення цього інтервалу, в іншому випадку сигнал скидання PUC буде згенеровано повторно. При роботі модуля в режимі сторожового таймера сигнал скидання PUC генерується або при записі в регістр WDTCTL з некоректним паролем, або після закінчення обраного інтервалу часу. При появі сигналу PUC модуль WDT+ скидається в початковий стан, а вивід RST/NMI конфігурується як вхід апаратного скидання.

Режим інтервального таймера

Встановлення біта WDTTMSEL в 1 переводить модуль WDT+ в режим інтервального таймера. Цей режим може використовуватися для періодичної генерації переривань. У режимі інтервального таймера після закінчення обраного інтервалу встановлюється прапорець WDTIFG. Сигнал скидання PUC в даному режимі не генерується, а стан біта дозволу переривання WDTIE регістра WDTIFG не змінюється.

Якщо біти WDTIE і GIE встановлені, то при установці прапорця WDTIFG генерується запит переривання. Прапорець переривання WDTIFG скидається автоматично при виклику процедури обробки цього переривання або може бути скинутий програмно. Адреса вектора переривання в режимі інтервального таймера відрізняється від адреси вектора в режимі сторожового таймера.

Переривання сторожового таймера

Для управління переривань модуля WDT + використовуються два біти регістрів спеціальних функцій:

- прапорець переривання WDTIFG (IFG1.0);
- біт дозволу переривання WDTIE (IE1.0).

При роботі модуля WDT + в режимі сторожового таймера прапорець WDTIFG є одним з джерел вектора скидання. Цей прапорець можна використовувати в процедурі обробки скидання для визначення причини скидання мікроконтролера. Встановлений прапорець WDTIFG означає, що скидання було ініційовано сторожовим таймером після закінчення заданого інтервалу часу або в результаті порушення ключа захисту. Якщо ж прапорець WDTIFG скинутий, то причина скидання була іншою.

При роботі модуля WDT + в режимі інтервального таймера прапорець WDTIFG встановлюється після закінчення обраного інтервалу часу і, якщо встановлені біти WDTIE і GIE, генерує запит переривання. Вектор переривання інтервального таймера не збігається з вектором скидання, який використовується в режимі сторожового таймера. У режимі інтервального таймера прапорець WDTIFG скидається автоматично при обробці переривання або може бути скинутий програмно.

В табл. 3.10 наведені біти регістру сторожового таймера WDTCTL.

Таблиця 3.10 – Регістр сторожового таймера WDTCTL

Біт (и)	Позначення	Опис
15-8	WDTPW	Ключ захисту WDT+. Завжди зчитується як 069h. Має записуватися як 0A5, в іншому випадку буде генеруватися сигнал PUC
7	WDTHOLD	Останов модуля WDT+. Установка біту WDTHOLD=1 при не використанні модуля дозволяє зменшити струм живлення мікроконтролера. 0 – модуль WDT+ не зупинено 1 – модуль WDT+ зупинено
6	WDTNMIES	Вибір фронту NMI. Цей біт використовується для вибору активного фронту немаскованого переривання при WDTNMI=1. Зміна біту може викликати генерацію немаскованого переривання. Щоб уникнути цього WDTNMIES потрібно змінити при WDTIE=1. 0 – немасковане переривання генерується по зростаючому фронту 1 – немасковане переривання генерується по спадаючому фронту
5	WDTNMI	Вибір NMI. За допомогою цього біту задається режим роботи виводу RST/NMI мікроконтролера. 0 – вхід апаратного скидання 1 – вхід немаскованого переривання
4	WDTTMSEL	Вибір режиму модуля WDT+: 0 – режим сторожового таймера 1 – режим інтервального таймера
3	WDTCNTCL	Скидання лічильника модуля WDT+. Встановлення біта WDTCNTCL = 1 викликає завантаження в лічильник значення 0000h. Біт WDTCNCTL скидається автоматично: 0 – нема дії 1 – WDTCNT = 0000h
2	WDTSSEL	Вибір тактового сигналу модуля WDT+: 0 – SMCLK 1 – ACLK
1..0	WDTISx	Вибір інтервалу модуля WDT+. Ці біти визначають часовий інтервал, по закінченню якого встановлюється прапорець WDTIFG і/або генерується сигнал PUC. 00 – частота тактового сигналу /32767 01– частота тактового сигналу /8192 10– частота тактового сигналу /512 11– частота тактового сигналу /64

Функціонування в режимах зниженого енергоспоживання

Мікроконтролери MSP430 мають кілька режимів зниженого енергоспоживання. У різних режимах користувачеві доступні різні джерела тактових сигналів. Відповідно, конфігурація модуля WDT + визначається вимогами конкретного застосування і режимом роботи модуля синхронізації мікроконтролера. Наприклад, якщо користувач передбачає задіяти режим LPM3, то модуль WDT+ не повинен використовуватися в режимі сторожового таймера з тактуванням від сигналу SMCLK, оскільки модуль WDT+ не дозволить виключити джерело цього сигналу, що призведе до підвищеного споживання в режимі LPM3. Якщо сторожовий таймер не використовується, то за допомогою біта WDTHOLD можна зупинити лічильник WDTCNT, зменшуючи тим самим споживання мікроконтролера.

Зміст завдання

1. Лабораторна робота присвячена дослідженню роботи сторожового таймера мікроконтролера MSP430F2013.

2. Завдання містить код програми 1, в якому необхідно визначити значення наступних регістрів: WDTCTL, IE1. Після цього код необхідно відкомпілювати, зневадити та запустити на виконання.

3. Завдання містить код програми 2, в якому необхідно визначити значення наступних регістрів: WDTCTL, IE1. Після цього код необхідно відкомпілювати, зневадити та запустити на виконання.

Код програми 1

```
#include <msp430x20x3.h>

void main(void){
  WDTCTL = __________;/* встановлення інтервалу
                      сторожового таймера ~30мс */
  IE1 |= ___________;/* дозвіл переривання
                      сторожового таймера */
  P1DIR |= 0x01;      /* налаштування P1.0 на
                      виведення */

  _BIS_SR(LPM0_bits + GIE);/* перехід в режим
                              LPM0 */
}

// WDT ISR

#pragma vector=WDT_VECTOR
|__interrupt void watchdog_timer(void){
  P1OUT ^= 0x01;  // Перемикання світлодіода
}
```

Код програми 2

```
#include <msp430x20x3.h>

void main(void){
  WDTCTL = _____; /* WDT 250 мс, ACLK, режим
                  інтервального таймера */
  IE1 |= _____;   // дозвіл переривання WDT

  P1DIR |= 0x01;  // налаштування P1.0 на виведення
  _BIS_SR(LPM3_bits + GIE); /* Перехід в режим
                            LPM3 і очікування
                            переривання */
}
// WDT ISR

#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void){
  P1OUT ^= 0x01;  // Перемикання світлодіода
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп'ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми 1.
4. Код програми 1 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів, а саме:
   - встановіть ключ захисту, біт скидання лічильника модуля WDT+, режим інтервального таймера. У якості тактового сигналу для модуля WDT+ використайте SMCLK, задайте часовий інтервал ~30мс:

     ```
     WDTCTL = ___________;
     ```

   - встановіть дозвіл переривання сторожового таймера:

     ```
     IE1 |= _____________;
     ```

5. Здійсніть компіляцію та зневадження проекту.
6. Запустіть проект на виконання, перегляньте значення регістрів WDTCTL, IE1 занесіть ці значення у звіт.
7. Створіть новий проект, додайте в нього файл з розширенням ".c" та скопіюйте в нього код програми 2.

8. Код програми 2 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів:

○ встановіть ключ захисту, біт скидання лічильника модуля WDT+, режим інтервального таймера. У якості тактового сигналу для модуля WDT+ використайте ACLK, задайте часовий інтервал ~250 мс:

```
WDTCTL = _____;
```

○ встановіть дозвіл переривання сторожового таймера:

```
IE1 |= _____;
```

9. Здійсніть компіляцію та зневадження проекту.

10. Запустіть проект на виконання, перегляньте значення регістрів WDTCTL, IE1 занесіть ці значення у звіт.

11. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- схема електрична функціональна сторожового таймера;
- лістинги програм;
- результати виконання;
- опис та порівняння роботи програми 1 та програми 2;
- висновки.

Контрольні запитання

1. Яке призначення сторожового таймера?
2. Які особливості модуля сторожового таймера WDT+?
3. Як функціонує сторожовий таймер?
4. Для чого необхідний лічильник сторожового таймера?
5. Як працює модуль WDT+ в режимі сторожового таймера?
6. Як працює модуль WDT+ в режимі інтервального таймера?
7. Що таке переривання?
8. Як відбуваються переривання сторожового таймера?
9. Які є біти регістру сторожового таймера WDTCTL?
10. Яким чином відбувається функціонування в режимах зниженого енергоспоживання?

3.6 Лабораторна робота №6. Дослідження роботи TIMER_A

Мета роботи: Дослідити роботу Timer_A використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Timer_A є 16-ти бітним таймером/лічильником з трьома регістрами захоплення/порівняння. Цей таймер може забезпечити кілька каналів захоплення/порівняння, генерацію сигналів з ШІМ та формування часових інтервалів. Крім того, Timer_A має розвинену підтримку переривань. Переривання можуть генеруватися як регістром лічильника таймера (при його переповненні), так і кожним з регістрів захоплення/порівняння.

Timer_A має такі особливості:

- асинхронний 16-бітний таймер/лічильник з чотирма робочими режимами:
  - довжина Timer_A: 16 біт;
  - регістр таймера/лічильника TAR збільшується або зменшується (залежно від робочого режиму) з кожним фронтом тактового сигналу;
  - в разі переповнення таймера, він може генерувати переривання;
  - широкий діапазон інтервалу переривань: від 1/MCLK до 32 секунд;
- джерело тактових імпульсів з можливістю вибирати та налаштувати:
  - ACLK, SMCLK, або ззовні через TxCLK або INCLK (обирається бітами TASSELx);
  - обране джерело тактових імпульсів може бути додатково поділено на 2, 4, 8 (конфігурація бітів IDx).
- регістри збору/порівняння з можливістю налаштування:
  - Timer_A має три або п’ять регістрів збору/порівняння;
- виходи з можливістю налаштування та декілька внутрішніх зв’язків з іншими модулями, що дозволяє швидше відповісти, оскільки не марнується жодних циклів поки ISR завантажує/виконує і уникає “пробудження” CPU, який в свою чергу не витрачає енергію:
  - виходи з можливістю модуляції ширини імпульсу;
  - Comparator_A;
  - прямий доступ до пам’яті (DMA);
- асинхронні входи і виходи, що замикаються:
  - регістри збору/порівняння Timer_A не буферизовані, негайно оновлюються при записі;
- регістр вектору переривань для швидкого декодування всіх переривань Timer_A:

- вектор переривання TACCR0 для TACCR0 CCIFG;
- вектор переривання TAIV для решти прапорців CCIFG і TAIFG.

На рис. 3.13 зображена схема електрична функціональна Timer_A.

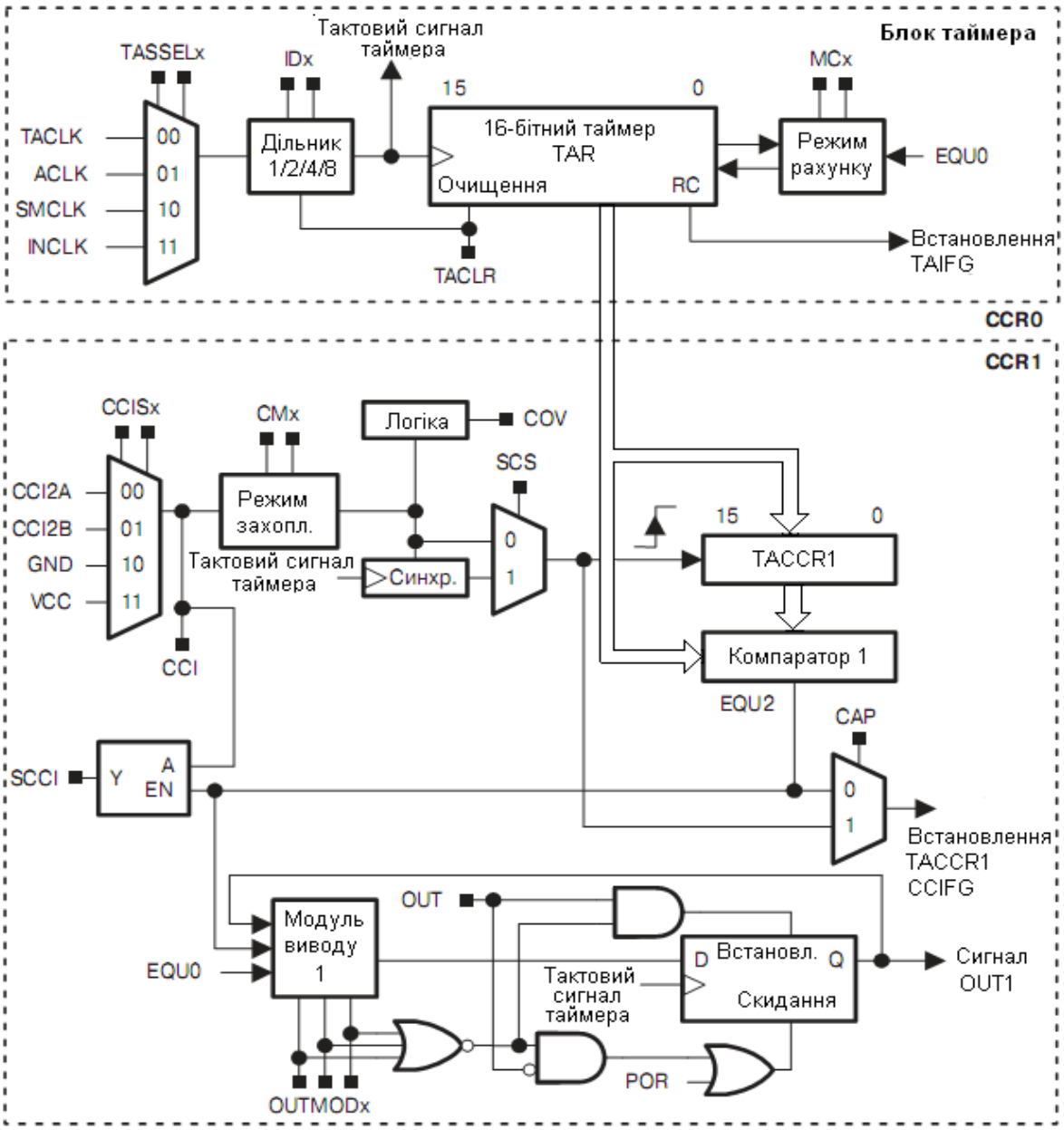


Рисунок 3.13 – Схема електрична функціональна Timer_A

Режими роботи Timer_A наведені в табл. 3.11.

Таблиця 3.11 – Режими роботи Timer_A

MCx	Режим	Опис
0 0	Стоп	Таймер зупинено
0 1	Вгору	Режим повторюваного підрахунку (від 0x0000 до значення в регістрі TACCR0)
1 0	Безперервний	Режим повторюваного безперервного підрахунку (від 0x0000 до 0xFFFF)
1 1	Вгору/вниз	Режим повторюваного підрахунку вгору/вниз (від 0x0000 до значення в регістрі TACCR0 і назад до нуля)

Регістри Timer_A наведено в табл. 3.12.

Таблиця 3.12 - Регістри Timer_A

Регістр	Позначення	Тип регістру	Адреса	Початковий стан
Регістр управління Timer_A	TACTL	R/W	0160h	Скидається після POR
Регістр лічильника Timer_A	TAR	R/W	0170h	Скидається після POR
Регістр управління блоком захоплення/порівняння 0 Timer_A	TACCTL0	R/W	0162h	Скидається після POR
Регістр захоплення/порівняння 0 Timer_A	TACCR0	R/W	0172h	Скидається після POR
Регістр управління блоком захоплення/порівняння 1 Timer_A	TACCTL1	R/W	0164h	Скидається після POR
Регістр захоплення/порівняння 1 Timer_A	TACCR1	R/W	0174h	Скидається після POR
Регістр вектора переривання Timer_A	TAIV	R/O	012Eh	Скидається після POR

Біти регістру управління Timer_A TACTL наведені в табл. 3.13.

Таблиця 3.13 – Регістр управління Timer_A TACTL

Біт (и)	Позначення	Опис
15..10	Unused	Біти не використовуються
9..8	TASSELx	Вибір джерела тактового сигналу: 00 – TACTL 01 – ACLK 10 – SMCLK 11 – INCLK
7..6	IDx	Коефіцієнт ділення вхідного дільника. Ці біти визначають коефіцієнт ділення задаючого тактового сигналу: 00 – /1 01 – /2 10 – /4 11 – /8
5..4	MCx	Управління режимом таймера (див. табл. 8.2). Установка MCx = 00h при невикористанні таймера зменшує загальне живлення мікроконтролера.
3	Unused	Не використовується
2	TACLR	Очищення таймера. Встановлення цього біту очищує регістр TAR, дільник і скидає ознаку напрямку рахунку. Біт TACLR автоматично скидається; завжди читається як 0.
1	TAIE	Дозвіл переривання таймера. Цей біт дозволяє генерацію запиту переривання TAIFG: 0 – переривання заборонено 1 – переривання дозволено
0	TAIFG	Прапорець переривання таймера: 0 – переривання не було 1 – переривання було

В табл. 3.14 описані режими роботи модуля виведення.

Таблиця 3.14 – Режими роботи модуля виведення

OUTMODx	Режим	Опис
000	Виведення	Вихідний сигнал OUTx визначається станом біту OUTx. Вихідний сигнал змінюється зразу ж після зміни біту OUTx.
001	Встановлення	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx вихід встановлюється. Цей стан зберігається до тих пір, поки таймер не буде скинутий або поки не буде заданий новий режим роботи модуля виводу з наступними впливом на вихід
010	Перемикання /скидання	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx, вихід перемикається. Вихід скидається при досягненні таймером значення, записаного в регістрі TACCR0
011	Встановлення /скидання	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx, вихід встановлюється. Вихід скидається при досягненні таймером значення, записаного в регістрі TACCR0
100	Перемикання	Вихід перемикається при досягненні таймером значення, записаного в регістрі TACCRx. Період вихідного сигналу рівний подвоєному періоду таймера
101	Скидання	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx, вихід скидається. Цей стан зберігається до тих пір, поки не буде заданий новий режим роботи модуля виводу з наступними впливом на вихід
110	Перемикання/ встановлення	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx, вихід перемикається. Вихід встановлюється при досягненні таймером значення, записаного в регістрі TACCR0
111	Скидання/ встановлення	При досягненні таймером в процесі рахунку значення, записаного в регістрі TACCRx, вихід скидається. Вихід встановлюється при досягненні таймером значення, записаного в регістрі TACCR0

Біти регістру захоплення/порівняння Timer_A TACCRx наведені в табл. 3.15.

Таблиця 3.15 – Регістри захоплення/порівняння Timer_A TACCRx

Біт (и)	Позначення	Опис
15..0	TACCRx	Режим порівняння: регістр TACCRx містить число, яке порівнюється зі змістом регістра лічильника TAR. Режим захоплення: вміст регістру TAR копіюється в регістр TACCRx в момент виконання операції захоплення

Біти регістру захоплення/порівняння Timer_A TACCRx наведені в табл. 3.16.

Таблиця 3.16 – Регістри управління блоком захоплення/порівняння Timer_A TACCTLx

Біт (и)	Позначення	Опис
15..14	CMx	Режим захоплення. 00 – немає захоплення 01 – захоплення по наростаючому фронті 10 – захоплення по спадаючому фронті 11 – захоплення по обом фронтам
13..12	CCISx	Вибір входу захоплення/порівняння. Ці біти визначають вхідний сигнал блока захоплення/порівняння. 00 – CCIxA 01 – CCIxB 10 – GND 11 – Vcc
11	SCS	Синхронізація захоплення. Цей біт використовується для синхронізації сигналу захоплення з тактовим сигналом таймера.
10	SCCI	Синхронізований вхід захоплення/порівняння. Вхідний сигнал блоку захоплення/порівняння фіксується по сигналу EQUx і може бути зчитаний за допомогою цього біту.
9	Unused	Не використовується. Завжди 0.

Продовження таблиці 3.16.

Біт (и)	Позначення	Опис
8	CAP	Режим роботи блоку захоплення/порівняння: 0 – Режим захоплення 1 – Режим порівняння
7..5	OUTMODx	Режим роботи модуля виведення (див. табл. 8.5). Для регістру TACCR0 використання режимів 2, 3, 6 і 7 не є доцільним, оскільки у цьому випадку EQUx = EQU0.
4	CCIE	Дозвіл переривання захоплення/порівняння. Цей біт дозволяє генерацію запиту переривання при встановленні відповідного прапорця CCIFG. 0 – переривання заборонено 1 – переривання дозволено
3	CCI	Вхід захоплення/порівняння. З допомогою біту можна визначити значення вхідного сигналу блоку захоплення/порівняння
2	OUT	Стан виводу. При роботі модуля виведення в режимі 0 цей біт напряму управляє станом виводу. 0 – на виводі низький рівень 1 – на виводі високий рівень
1	COV	Переповнення захоплення. Цей біт показує переповнення при операціях захоплення. Біт COV повинен скидатися програмно. 0 – не було переповнення при захоплені 1 – було переповнення при захоплені
0	CCIFG	Прапорець переривання захоплення/порівняння: 0 – переривання не було 1 – переривання було

Скидання таймера

Таймер може бути скинуто, використовуючи наступні дії:

- запис 0 в регістр TAR;
- запис 0 в регістр TACCR0 у випадку якщо таймер знаходиться не у безперервному режимі;
- встановлення біта TACLR в регістрі TACTL.

Переривання Timer_A

З модулем 16-ти бітного Timer_A пов'язані два вектора переривання:

● вектор переривання TACCR0 для біта CCIFG, який відповідає регістру TACCR0;

● вектор переривання TAIV для всіх інших прапорців CCIFG і прапорця TAIFG.

У режимі захоплення будь-який з прапорців CCIFG встановлюється в момент збереження значення таймера у відповідному регістрі TACCRx. У режимі порівняння будь-який з прапорців CCIFG встановлюється при досягненні таймером в процесі рахунку значення, записаного у відповідному регістрі TACCRx. Крім того, будь-який прапорець CCIFG можна встановити або скинути програмно. При установці прапорця CCIFG, якщо встановлений відповідний біт CCIE і біт загального дозволу переривань GIE, генерується запит переривання.

У табл. 3.17 представлені можливі значення вектора переривань таймера у регістрі TAIV.

Таблиця 3.17 – Можливі значення вектора переривань таймера у регістрі TAIV

Вміст TAIV	Джерело переривання	Прапорець переривання	Пріоритет переривання
00h	Не було переривання		
02h	Захоплення/порівняння 1	TACCR1 CCIFG	Вищий
04h	Зарезервовано		
06h	Зарезервовано		
08h	Зарезервовано		
0Ah	Переповнення таймера	TAIFG	
0Ch	Зарезервовано		
0Eh	Зарезервовано		Нижчий

Характеристики режимів роботи

Режим «Вгору»

Основними особливостями цього режиму є:

● TAR підраховує кожен тактовий імпульс до тих пір, поки значення в ньому не стає рівним значенню в регістрі TACCR0;

● прапорець переривання CCIFG TACCR0 встановлюється, коли таймер рахує до значення TACCR0;

• коли він досягає цього значення, EQU0 = 1 (перезапускає підрахунок TAR нуля);

• прапорець переривання TAIFG встановлено, коли таймер рахує від значення в TACCR0 до нуля.

• період переривання: t_{INT} = 1/[f_{CLK}/дільник частоти/(TxCCR0+1)];

• t_{INT} – період переривання TxIFG (сек);

• f_{CLK} – частота джерела тактових імпульсів (Гц);

• дільник частоти (біти IDx).

Безперервний режим

Основними характеристики цього режиму є:

• TxR підраховує кожен тактовий імпульс до 0xFFFF (65536 раз);

• коли він досягає цього значення, на наступному тактовому імпульсі він перезапустить рахунок TxR з нуля;

• прапорець переривання TxIFG встановлено, коли таймер рахує від 0xFFFF до нуля.

• період переривання: t_{INT} = 1/[f_{CLK}/дільник частоти /65536].

Режим «Вгору/Вниз»

Основні характеристики цього режиму:

• TxR рахує кожен тактовий імпульс до тих пір, поки він не досягне значення в регістрі TxCCR0;

• прапорець переривання CCIFG TxCCR0 встановлено, коли таймер рахує від TxCCR0 − 1 до TxCCR0;

• коли він досягає цього значення, підрахунок інвертовано, починається на наступному тактовому імпульсі для зменшення до нуля;

• прапорець переривання TxIFG встановлено, коли таймер завершує підрахунок від 0x0001 да 0x0000;

• період переривання: t_{INT} = 1/[f_{CLK}/дільник частоти/(TxCCR0×2)].

Зміст завдання

1. Лабораторна робота присвячена дослідженню роботи Timer_A.

2. Завдання містить код програми 1, в якому необхідно визначити значення регістру TACTL, вибрати джерело тактових імпульсів, встановити режим роботи. Після цього програму необхідно відкомпілювати, зневадити та запустити на виконання.

3. Завдання містить код програми 2, в якому необхідно визначити значення наступних регістрів: P1SEL, P1DIR, CCTL0, CCTL1, TACTL. Після цього програму необхідно відкомпілювати, зневадити та запустити на виконання.

4. Для кожної програми необхідно обчислити частоту перемикання світлодіода.

Код програми 1

```
#include  <msp430x20x3.h>

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */
  P1DIR |= 0x01;// налаштування P1.0 на виведення
  TACTL = ___________;
  _BIS_SR(_______+ GIE); /* перехід у відповідний
                          режим і очікування
                          переривання */
}
// Timer_A1 ISR
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void){
  switch( TAIV ){
    case  2:  break; // CCR1 не використовується
    case 10:  P1OUT ^= 0x01;    // переповнення
              break;
  }
}
```

Код програми 2

```
#include  <msp430x20x3.h>

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */
  P1SEL |= _______;
  P1DIR |= _______;
  CCTL0 = ________;
  CCTL1 = ________;
  TACTL = ________; /* SMCLK, безперервний режим,
                     дозвіл переривань  */
  _BIS_SR(LPM0_bits + GIE);/* перехід в режим
                              LPM0 і очікування
                              переривання */
}
// Timer_A0 ISR
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A0 (void){
  CCR0 += 200; // додавання значення до CCR0
}
// Timer_A1 ISR
#pragma vector=TIMERA1_VECTOR
```

```
__interrupt void Timer_A1(void){
  switch( TAIV ){
  case 2: CCR1+=1000; /* додавання значення до CR0 */
          break;
  case 10: P1OUT ^= 0x01;  // переповнення
          break;
 }
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп'ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми 1.
4. Код програми 1 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів, а саме:
   - встановіть відповідні біти регістра TACTL: джерело тактового сигналу – ACLK, режим роботи – безперервний, дозвіл переривання:

     ```
     TACTL = _____________;
     ```

   - виберіть режим роботи LPM3:

     ```
     _BIS_SR( _______ + GIE);
     ```

5. Здійсніть компіляцію та зневадження проекту.
6. Запустіть проект на виконання, перегляньте значення регістрів TACTL та TAIV, обчисліть частоту перемикання світлодіода та занесіть ці значення у звіт.
7. Повторіть дії описані в пунктах 4-6, вибравши у якості джерела тактового сигналу SMCLK та режим роботи LPM0.
8. Створіть новий проект, додайте в нього файл з розширенням ".c" та скопіюйте в нього код програми 2.
9. Код програми 2 є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів, а саме:
   - встановіть біти в регістрі P1SEL для виводів P1.1 та P1.2:

     ```
     P1SEL |= ____;
     ```

   - налаштуйте виводи P1.0 - P1.2 на виведення:

     ```
     P1DIR |= ____;
     ```

   - для регістру TACCL0 виберіть режим роботи модуля виведення – перемикання. Встановіть біт дозволу переривання захоплення/порівняння:

     ```
     CCTL0 = ___________;
     ```

○ для регістру TACCL1 виберіть режим роботи модуля виведення – перемикання. Встановіть біт дозволу переривання захоплення/порівняння:

```
CCTL1 =_____________;
```

○ встановіть відповідні біти регістра TACTL: джерело тактового сигналу – SMCLK, режим роботи – безперервний, дозвіл переривання:

```
TACTL = _____________;
```

10. Запустіть проект на виконання, перегляньте значення регістрів TACTL та TAIV, обчисліть частоту перемикання світлодіода та занесіть ці значення у звіт.

11. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- схема електрична функціональна Timer_A;
- лістинги програм;
- результати виконання;
- опис роботи програми 2;
- висновки.

Контрольні запитання

1. Що представляє собою Timer_A?
2. Які особливості має Timer_A?
3. Які регістри входять до складу Timer_A?
4. В яких режимах роботи працює Timer_A?
5. Які режими роботи модуля виведення?
6. Які основні особливості режиму «Вгору»?
7. Які основні особливості безперервного режиму?
8. Які основні особливості режиму «Вгору/Вниз»?
9. Як відбувається скидання таймера?
10. Яким чином відбуваються переривання Timer_A?
11. Які можуть бути можливі значення вектора переривань таймера у регістрі TAIV?
12. Які біти регістру управління Timer_A TACTL можна встановити?
13. Як працюють регістри захоплення/порівняння Timer_A TACCR?
14. Як працюють регістри управління блоком захоплення/порівняння Timer_A TACCTL?

3.7 Лабораторна робота №7. Дослідження роботи флеш-пам’яті

Мета роботи: Дослідити організацію флеш-пам'яті та роботу контролера флеш-пам'яті мікроконтролера MSP430F2013, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Флеш-пам'ять є гібридом ROM та RAM пам’яті.

Особливості флеш-пам’яті наступні:

- невисока ціна;
- електрично-програмована;
- швидке зчитування даних;
- висока щільність;
- не втрачає даних при вимкненні живлення.

Саме тому, флеш-пам’ять це одна з найбільш популярних технологій для зберігання програмного коду та значень констант. Мікроконтролери сімейства MSP430 містять інтегровану флеш-пам'ять. Флеш-пам'ять у цих мікроконтролерах може адресуватися і записуватися побітно, побайтно або послівно. Модуль флеш-пам'яті має власний контролер, який керує операціями програмування і стирання пам'яті. Контролер має чотири регістри, тактовий генератор, а також генератор напруги, який формує напруги, необхідні для виконання операцій запису і стирання. Контролер флеш-пам'яті має такі особливості:

- вбудований генератор напруги програмування;
- побітове, побайтове або послівне програмування;
- наднизьке споживання;
- підтримується стирання сегментів і загальне стирання;
- два режими зчитування при граничних умовах (опціонально).

Структурна схема контролера флеш-пам'яті наведена на рис. 3.14.

Сегментна організація флеш-пам'яті

У мікроконтролерах MSP430 вся флеш-пам'ять поділена на сегменти. Записати у флеш-пам'ять можна окремі біти, байти або ж слова, проте найменшою одиницею флеш-пам'яті, яку можна стерти, є сегмент. Крім того, флеш-пам'ять містить дві секції – основну і інформаційну. Обидві секції функціонують абсолютно однаково. І програмний код, і дані можуть розташовуватися в обох секціях. Різниця між цими секціями полягає в різних значеннях розміру сегмента і фізичних адрес.

Інформаційна секція містить чотири 64-х байтних сегменти, в той час як основна секція містить два чи більше сегментів розміром 512 байт. Сегменти в свою чергу поділені на блоки. На рис. 3.15 показана сегментна організація модуля флеш-пам'яті мікроконтролера MSP430F2013 об’ємом 32 КБ, який має чотири основних і чотири інформаційних сегмента.

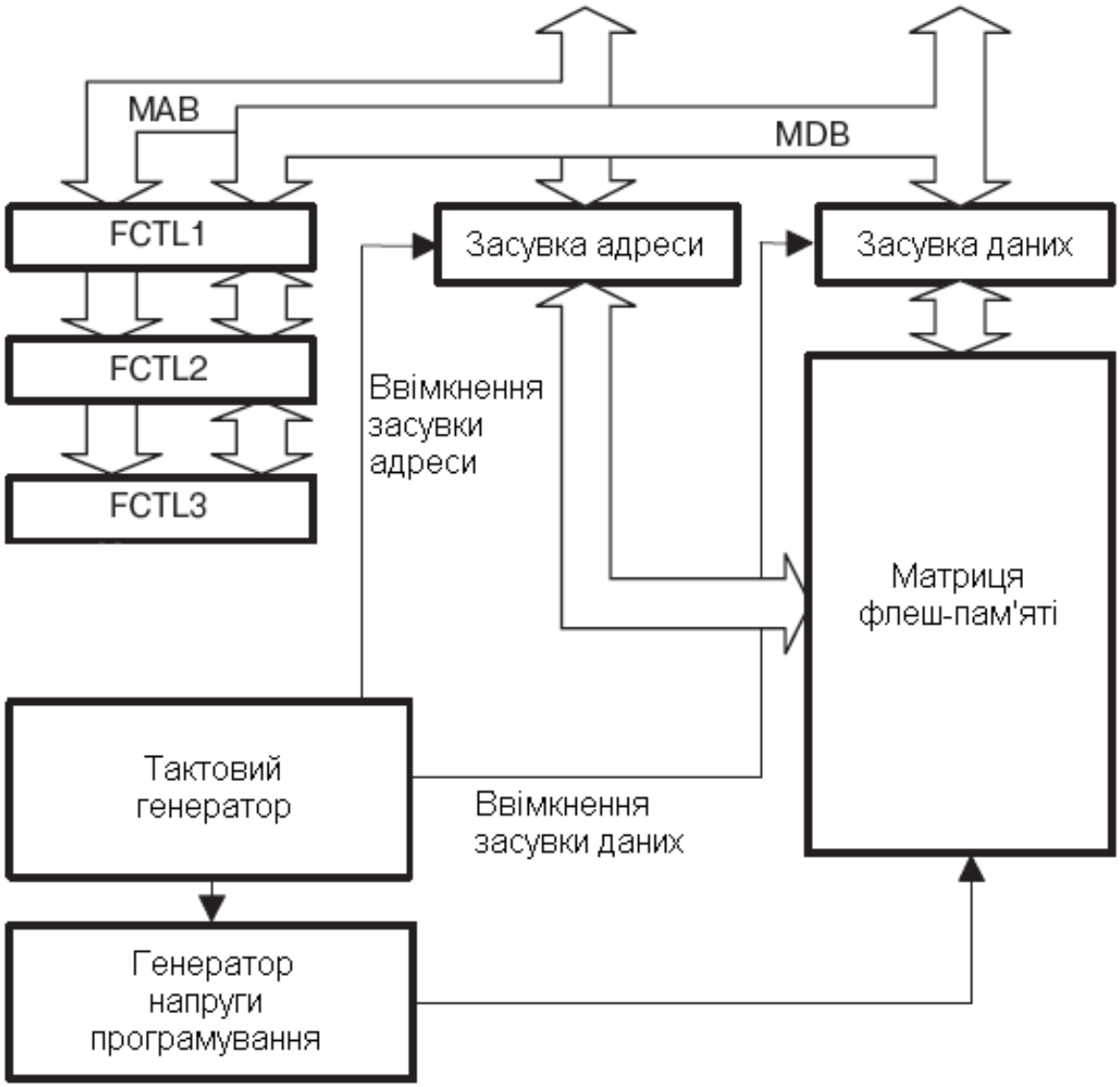


Рисунок 3.14 – Структурна схема контролера флеш-пам'яті

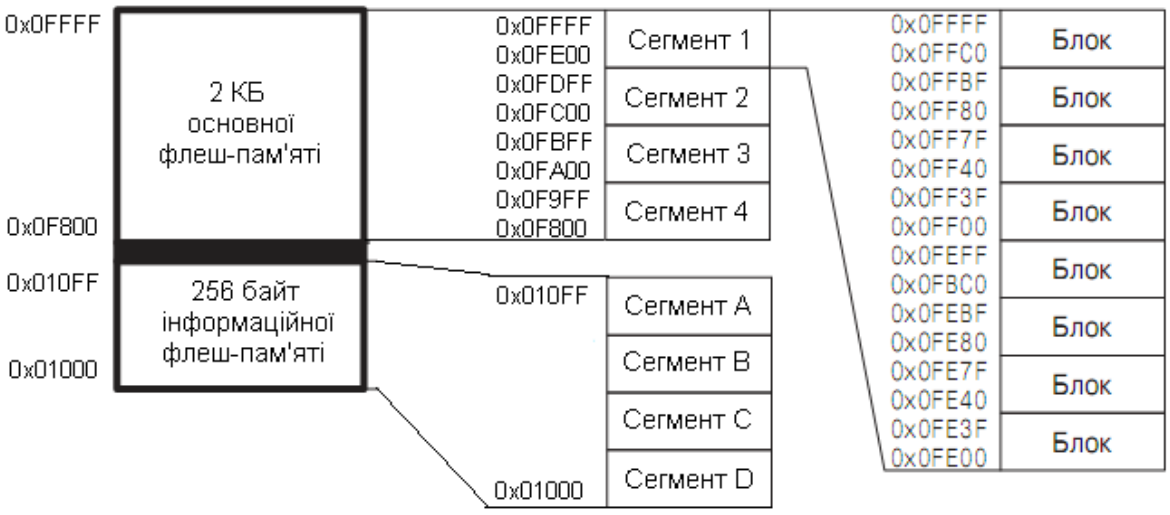


Рисунок 3.15 – Сегментна організація флеш-пам'яті

Функціонування флеш-пам'яті

За замовчуванням флеш-пам'ять знаходиться в режимі читання. У цьому режимі стирання чи запис флеш-пам'яті заблоковані, а тактовий генератор і генератор напруги вимкнені – пам'ять працює подібно ПЗП.

Флеш-пам'ять сімейства MSP430 підтримує внутрішньосхемне програмування (ISP) і не вимагає додаткового джерела напруги. CPU може здійснювати програмування власної флеш-пам'яті.

При використанні бітів BLKWRT, WRT, MERAS і ERASE можуть бути вибрані наступні режими запису / стирання флеш-пам'яті:

- запис байта/слова;
- запис блоку;
- стирання сегмента;
- загальне стирання (стирання всіх сегментів основної секції пам'яті);
- повне стирання (стирання всієї пам'яті).

Читання або запис флеш-пам'яті під час її програмування чи стирання заборонені. Якщо під час запису або стирання пам'яті потрібно виконання програми, то виконуваний код повинен розташовуватися в ОЗП. Процес зміни флеш-пам'яті може бути ініційований як з флеш пам'яті, так і з ОЗП.

Тактовий генератор контролера флеш-пам’яті

Операції запису і стирання управляються тактовим генератором контролера, схема електрична функціональна якого наведена на рис. 3.16. Робоча частота генератора f_{FTG} повинна знаходитися в діапазоні від 257 кГц до 476 кГц.

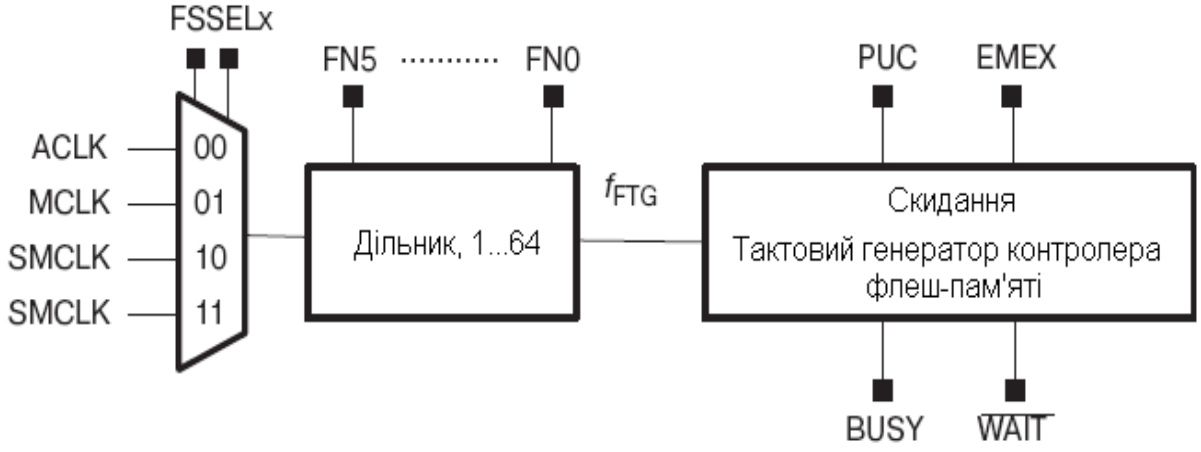


Рисунок 3.16 – Схема електрична функціональна тактового генератора контролера флеш-пам'яті

Вибір опорного сигналу тактового генератора контролера флеш-пам’яті

Тактовий генератор контролера флеш-пам'яті може тактуватися від будь-якого з тактових сигналів ACLK, SMCLK або MCLK. З вибраного

сигналу при використанні дільника, коефіцієнт ділення якого задається бітами FNx, формується тактовий сигнал контролера f_{FTG}. Якщо під час запису або стирання частота цього сигналу буде відрізнятися від зазначеної в специфікації, то результат операції може виявитися непередбачуваним або ж робочі параметри флеш-пам'яті можуть вийти за межі, що гарантують надійну роботу. При виявленні несправності тактового генератора під час операції запису або стирання дана операція переривається і встановлюється прапорець FAIL. Результат операції при цьому буде непередбачуваним.

Під час виконання операції запису або стирання вибране джерело тактового сигналу не можна виключити, перевівши мікроконтролер MSP430 в режим зниженого енергоспоживання. Це джерело залишиться в активному стані до завершення операції, після чого буде вимкнуте.

Стирання флеш-пам’яті

При стиранні біти флеш-пам'яті встановлюються в 1. Стан кожного окремого біта при програмуванні може бути змінено з 1 на 0, проте для зміни його стану з 0 на 1 потрібно цикл стирання. Найменшим елементом флеш-пам'яті, що допускають незалежне стирання, є сегмент. Передбачено три режими стирання, які задаються бітами ERASE і MERAS відповідно до табл. 3.18.

Таблиця 3.18 – Режими стирання

MERAS	ERASE	Режим стирання
0	1	Стирання сегменту
1	0	Загальне стирання
1	1	LOCKA = 0 – стирання основної і інформаційної секцій флеш-пам'яті LOCKA = 1 – стирання тільки основної секції флеш-пам'яті

Запис у флеш-пам'ять

Можливі режими запису задаються бітами WRT та BLKWRT у відповідністю з табл. 3.19.

Таблиця 3.19 – Режими запису

BLKWRT	WRT	Режим запису
0	1	Запис байта/слова
1	1	Запис блоку

Сегмент А

Сегмент інформаційної секції флеш-пам'яті може бути заблокований незалежно від інших секцій за допомогою біта LOCK. При LOCK = 1 запис або стирання сегмента А заборонені, а весь вміст інформаційної секції захищений від стирання при виконанні операції загального стирання або при програмуванні мікроконтролера. При LOCK = 0 запис і стирання сегмента А здійснюються нарівні з будь-якими іншими секціями флеш-пам'яті, а вміст інформаційної секції при виконанні загального стирання або при програмуванні мікроконтролера стирається.

Стан біта LOCKA змінюється шляхом запису в нього 1. Запис 0 в біт LOCKA не змінює його стану. Це дозволяє використовувати без змін існуючі процедури програмування флеш-пам'яті.

Сегмент А містить дані калібрації. Після вимикання сегмент А має захист від програмування і стирання. Захист може бути знятий, але потрібно проявляти обережність, щоб не стерти цей сегмент, у випадку необхідності використання даних калібрації.

Регістри контролера флеш-пам'яті

Список регістрів контролера флеш-пам’яті наведений в таблиці 3.20.

Таблиця 3.20 – Регістри контролера флеш-пам’яті

Регістр	Позначення	Тип регістру	Адреса	Вихідний стан
Регістр управління 1 контролера флеш-пам'яті	FCTL1	R/W	0128h	09600h після PUC
Регістр управління 2 контролера флеш-пам'яті	FCTL2	R/W	012Ah	09642h після PUC
Регістр управління 3 контролера флеш-пам'яті	FCTL3	R/W	012Ch	09658h після PUC
Регістр дозволу переривань 1	IE1	R/W	0000h	Скидається після PUC
Регістр прапорців переривань 1	IFG1	R/W	0002h	Скидається після PUC

У таблиці 3.21 наведений опис бітів регістру FCTL1.

Таблиця 3.21 – Опис регістру FCTL1

Біт (и)	Позначення	Опис
15-8	FRKEY / FWKEY	Ключ захисту FCTLx. Завжди зчитується як 096h. Має записуватися як 0A5, в іншому випадку буде генеруватися сигнал PUC
7	BLKWRT	Режим блочного запису. Біт BLKWRT автоматично скидається при встановленні біта EMEX: 0 – режим блочного запису вимкнений 1 – режим блочного запису ввімкнений
6	WRT	Режим запису. Біт BLKWRT автоматично скидається при встановленні біта EMEX: 0 – режим запису вимкнений 1 – режим запису ввімкнений
5	Reserved	Зарезервований. Завжди зчитується як 0.
4	EEIEX	Аварійне завершення роботи контролера під час переривання
3	EEI	Дозвіл переривання під час стирання
2	MERAS	Вибір режиму стирання. Див. табл. 9.1. Біти MERAS і ERASE автоматично скидаються при встановленні біта EMEX.
1	ERASE	
0	Reserved	Зарезервований. Завжди зчитується як 0.

У таблиці 3.22 наведений опис бітів регістру FCTL2.

Таблиця 3.22 – Опис регістру FCTL2

Біти	Позначення	Опис
15-8	FRKEY / FWKEY	Ключ захисту FCTLx. Завжди зчитується як 096h. Має записуватися як 0A5, в противному випадку буде генеруватися сигнал PUC
7-6	FSSELx	Вибір джерела тактового сигналу контролера флеш-пам'яті: FSSEL1 FSSEL0 = 00 -> ACLK FSSEL1 FSSEL0 = 01 -> MCLK FSSEL1 FSSEL0 = 10 -> SMCLK FSSEL1 FSSEL0 = 11 -> SMCLK
5-0	FNx	Подільник тактового сигналу контролера флеш-пам'яті: FNx = 00h -> /1 …. FNx = 03Fh -> /64

У таблиці 3.23 наведений опис регістру FCTL3.

Таблиця 3.23 – Опис регістру FCTL3

Біт(и)	Позначення	Опис
15-8	FRKEY / FWKEY	Ключ захисту FCTLx. Завжди зчитується як 096h. Має записуватися як 0A5, в противному випадку буде генеруватися сигнал PUC
7	FAIL	Збій при виконанні операції. Цей біт встановлюється при знаходженні несправності джерела тактового сигналу f_{FTG} чи при аварійному завершенні операції з флеш-пам'яттю в результаті переривання, коли EEIEX=1. Біт FAIL має скидатися програмно: 0 – збою не було 1 – збій був
6	LOCKA	Блокування сегменту А та інших сегментів інформаційної пам'яті: 0 – сегмент А розблокований, вся інформаційна пам'ять стирається при загальному стиранні 1 – сегмент А заблокований, інформаційна пам'ять захищена від стирання при загальному стиранні
5	EMEX	Екстрене завершення роботи з пам'яттю: 0 – не використовувати екстрене завершення операції 1 – екстрено завершити операцію
4	LOCK	Блокування флеш-пам'яті. Цей біт розблоковує флеш-пам'ять для операцій запису чи стирання: 0 – розблоковано 1 – заблоковано
3	WAIT	Прапорець очікування. Цей біт показує що в даний момент відбувається запис у флеш-пам'ять: 0 – флеш-пам'ять не готова до запису наступного байта/слова 1 – флеш-пам'ять готова до запису наступного байта/слова
2	ACCVIFG	Прапорець переривання при порушенні доступу: 0 – немає переривання 1 – є переривання
1	KEYV	Прапорець порушення ключа захисту FCTLx: 0 – був записаний правильний ключ 1 – був записаний неправильний ключ
0	BUSY	Прапорець зайнятості. Цей біт показує стан тактового генератора контролера флеш-пам'яті: 0 – не зайнятий 1 – зайнятий

Зміст завдання

1. Лабораторна робота присвячена дослідженню організації флеш-пам'яті та роботи контролера флеш-пам'яті.

2. Завдання містить код програми, в якому необхідно визначити значення наступних регістрів контролера флеш пам'яті: FCTL1, FCTL2, FCTL3. Після цього програму необхідно відкомпілювати, зневадити та запустити на виконання.

Код програми

Для виконання лабораторної роботи використайте наступний код:

```
#include  <msp430x20x3.h>

char value; /*  8-ми бітне значення для запису у
            сегмент A */

// прототипи функцій
void write_SegC (char value);
void copy_C2D (void);

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              Timer */
  if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
  {
    while(1);
  }

  BCSCTL1 = CALBC1_1MHZ;    /* встановлення DCO
                               як 1 МГц */
  DCOCTL = CALDCO_1MHZ;
  FCTL2 = _________________;  /* встановлення
                               MCLK/3 */
  value = 0;                //ініціалізація значення

  while(1){
    write_SegC(value++);/* запис сегменту C,
                           інкрементація змінної
                           value*/
    copy_C2D(); // копіювання сегменту C в D
    _NOP();     // тут встановити точку зупину
  }
}
```

```
void write_SegC (char value){
  char *Flash_ptr;
  unsigned int i;
  Flash_ptr = (char *) 0x1040;  /* ініціалізація
                                   покажчика */
  FCTL1 = _____________;        /* встановлення
                                   біту ERASE */
  FCTL3 = _____________;        /* очищення біту
                                   LOCK */
  *Flash_ptr = 0;               /* стирання
                                   сегменту */
  FCTL1 = ____________; /* встановлення біту WRT
                           для запису */

  for (i=0; i<64; i++){
    *Flash_ptr++ = value; /* запис значення у
                             флеш-пам'ять */
  }
  FCTL1 = _____________;  // очищення біту WRT
  FCTL3 = _____________;  // встановлення біту LOCK
}

void copy_C2D (void){
  char *Flash_ptrC;      // покажчик на сегмент C
  char *Flash_ptrD;      // покажчик на сегмент D
  unsigned int i;
  Flash_ptrC = (char *) 0x1040; /* ініціалізація
                                   покажчика на
                                   сегмент C */
  Flash_ptrD = (char *) 0x1000; /* ініціалізація
                                   покажчика на
                                   сегмент D */
  FCTL1 = _____________;     /* встановлення біту
                                ERASE */
  FCTL3 = _____________;    // очищення біту LOCK
  *Flash_ptrD = 0;          // стирання сегменту
  FCTL1 = _____________;  /* встановлення біту WRT
                             для запису */
  for (i=0; i<64; i++){
    *Flash_ptrD++ = *Flash_ptrC++;
  }
  FCTL1 = __________;    // очищення біту WRT
  FCTL3 = __________;    // встановлення біту LOCK
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп’ютера.

2. Запустіть середовище розробки вказане викладачем (CCE або IAR).

3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми.

4. Код програми є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів.

5. У функції main() налаштуйте конфігурацію регістра FCTL2 для встановлення тактової частоти генератора контролера на рівні MCLK/3.

```
FCTL2 = _________________;
```

6. У функції write_SegC здійсніть наступні налаштування:

   ○ Налаштуйте конфігурацію регістра FCTL1 для встановлення біту ERASE та конфігурацію регістра FCTL3 для очищення біту LOCK.

```
FCTL1 = _________________;

FCTL3 = _________________;
```

   ○ Налаштуйте конфігурацію регістра FCTL1 для встановлення біту WRT.

```
FCTL1 = _________________;
```

   ○ Налаштуйте конфігурацію регістра FCTL1 для очищення біту WRT та конфігурацію регістра FCTL3 для встановлення біту LOCK.

```
FCTL1 = _____________;

FCTL3 = _____________;
```

7. Повторіть дії описані в пункті 6 для функції copy_C2D.

8. Здійсніть компіляцію та зневадження проекту.

9. Поставте точку зупину на рядку коду з _NOP (). Відкрийте властивості breakpoint та в пунктах «Skip count» та «Current count» поставте значення вказані викладачем.

10. Запустіть проект на виконання, перегляньте значення регістрів FCTL1, FCTL2, FCTL3, сегментів пам’яті C та D, та занесіть ці значення у звіт.

11. Закрийте середовище розробки та від’єднайте пристрій eZ430-F2013 від USB-порту комп’ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- схема електрична функціональна контролера флеш-пам'яті;
- результати виконання;
- висновки.

Контрольні запитання

1. Що таке флеш-пам’ять?
2. Які особливості флеш-пам’яті?
3. Які особливості контролера флеш-пам’яті?
4. Поясніть структурну схему контролера флеш-пам’яті.
5. Що собою представляє сегментна організація флеш-пам'яті?
6. Яким чином відбувається функціонування флеш-пам'яті?
7. Яким чином працює тактовий генератор контролера флеш-пам’яті?
8. Як відбувається стирання флеш-пам’яті?
9. Як відбувається запис у флеш-пам’ять?
10. Як працює Сегмент A?
11. Які існують регістри контролера флеш-пам'яті?
12. Для чого призначений біт BLKWRT регістру управління 1 контролера флеш-пам'яті FCTL1?
13. Для чого призначені біти FSSELx регістру управління 2 контролера флеш-пам'яті FCTL2?
14. Для чого призначені біти FNx регістру управління 2 контролера флеш-пам'яті FCTL2
15. Для чого призначений біт LOCK регістру управління 3 контролера флеш-пам'яті FCTL3?

3.8 Лабораторна робота №8. Дослідження модуля цифро-аналогового перетворення SD16_A

Мета роботи: Дослідити модуль цифро-аналогового перетворення SD16_A на прикладі реалізації логера температури, використовуючи інструмент розробника eZ430-F2013.

Теоретичні відомості

Інструмент розробника eZ430-F2013 містить модуль SD16_A який підтримує 16-розрядні аналого-цифрові перетворення. До складу модуля SD16_A входить сигма-дельта АЦП, що має буфер з високим вхідним опором, і внутрішнє джерело опорної напруги. Модуль має до восьми мультиплексованих повністю диференціальних аналогових каналів, включаючи канали для вимірювання сигналу від вбудованого датчика температури і напруги живлення. АЦП реалізовано на базі сигма-дельта модулятора другого порядку і децимуючого цифрового фільтра. Для децимації використовується гребінчастий фільтр з програмованим коефіцієнтом передискретизації (до 1024). Додаткова фільтрація може здійснюватися програмно.

Модуль SD16_A має такі особливості:

- 16-ти бітна сигма-дельта архітектура;
- до восьми мультиплексованих диференціальних аналогових входів на канал;
- програмно активується вбудований генератор опорної напруги (1.2 В);
- зовнішнє чи внутрішнє джерело опорної напруги (вибирається програмно);
- вбудований датчик температури;
- вхідна частота модулятора до 1.1 МГц;
- вхідний буфер з високим вхідним опором;
- програмно активується режим перетворення зі зниженим живленням.

Аналого-цифрове перетворення здійснюється однобітним сигма-дельта модулятором 2-го порядку. Однобітний компаратор, що входить до складу модулятора, виконує квантування вхідного сигналу з частотою модуляції f_M. Отриманий в результаті бітовий потік усереднюється цифровим фільтром *(digital filter)* для формування результату перетворення.

Конфігурація модуля SD16_A здійснюється програмно.

Максимальний розмах вхідної напруги для кожної пари аналогових входів залежить від встановленого коефіцієнта підсилення (PGA = 1, 2, 4, 8, 16 і 32). Вхідний сигнал повинен знаходитися в діапазоні ± VFSR, при цьому величина VFSR визначається виразом:

$$V_{FSR} = \frac{V_{REF} / 2}{GAIN_{PGA}}$$

На рис. 3.17 зображена схема електрична функціональна модуля SD16_A.

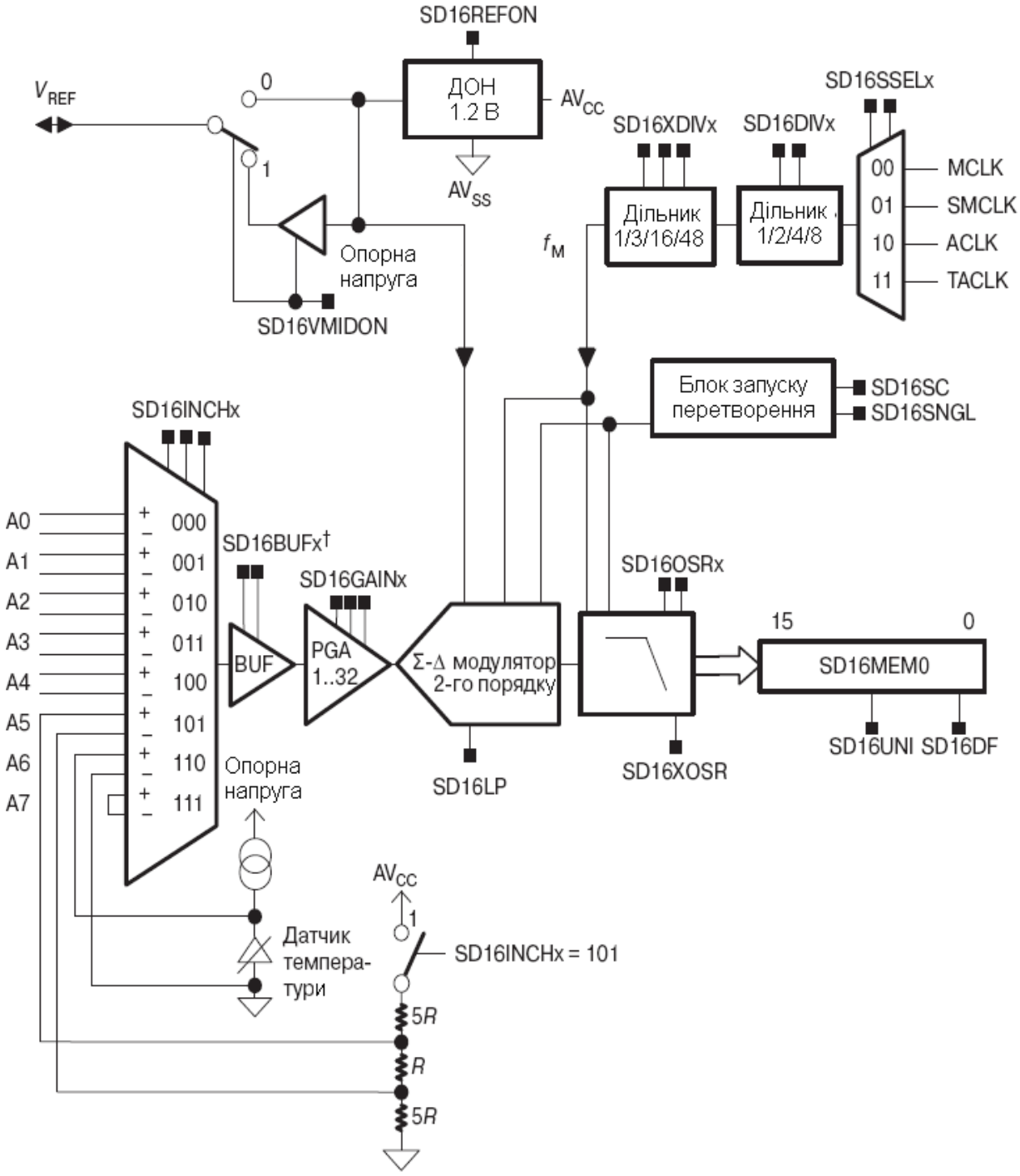


Рисунок 3.17 – Схема електрична функціональна модуля SD16_A

При опорній напрузі 1.2 В і одиничному коефіцієнті підсилення максимальний розмах вхідного сигналу дорівнює:

$$\pm V_{FSR} = \frac{1.2B/2}{1} = \pm 0.6B.$$

На рис. 3.18 зображений діапазон аналогового входу SD16_A.

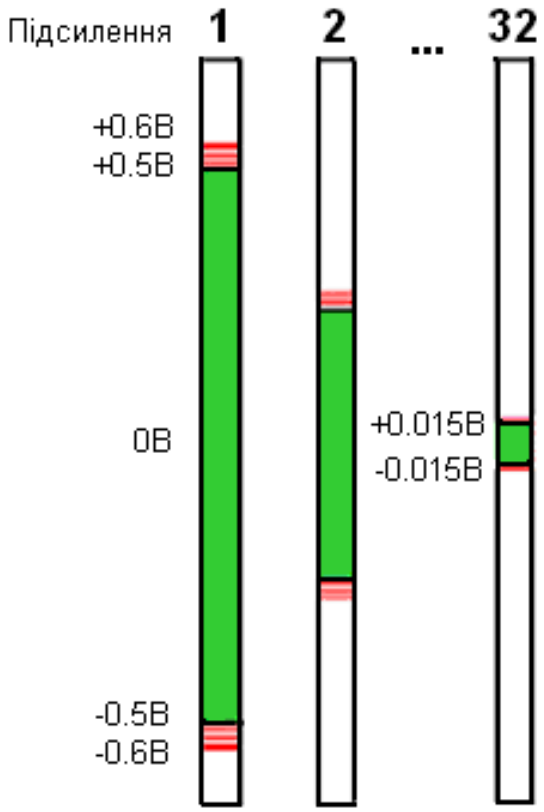


Рисунок 3.18 – Діапазон аналогового входу SD16_A

Генератор опорної напруги

Модуль SD16_A має вбудоване джерело опорної напруги 1.2 В, яке вмикається встановленням біта SD16REFON. Для зниження рівня шумів при використанні цього джерела рекомендується між виводами VREF та AVSS підключати зовнішній конденсатор ємністю 0.1 мкФ. Напруга внутрішнього джерела може використовуватися поза мікроконтролером при SD16VMIDON = 1. Максимальна навантажувальна здатність буфера вбудованого джерела опорної напруги становить 1 мА. При використанні внутрішньої опорної наруги поза мікроконтролером між виводами VREF і AVSS необхідно увімкнути конденсатор ємністю 0.47 мкФ.

Автоматичне відключення

Модуль SD16_A розроблений для використання в додатках з низьким енергоспоживанням. Тому він автоматично відключається в ті періоди часу, коли перетворення не виконується, і автоматично включається при запуску перетворення. Генератор опорного напруги автоматично не відключається, проте його можна відключити вручну, скинувши біт SD16REFON. При вимкнених ядрі АЦП і генераторі опорної напруги їх струм споживання дорівнює нулю.

В табл. 3.24 наведені регістри модуля SD16_A.

Таблиця 3.24 – Регістри модуля SD16_A

Регістр	Позначення	Тип регістру	Адреса	Вихідний стан
Регістр управління модуля SD16_A	SD16CTL	R/W	0100h	Скидається після PUC
Регістр вектора переривань модуля SD16_A	SD16IV	R/W	0110h	Скидається після PUC
Регістр управління каналу 0 модуля SD16_A	SD16CCTL0	R/W	0102h	Скидається після PUC
Регістр даних модуля SD16_A	SD16MEM0	R/W	0112h	Скидається після PUC
Регістр управління входом модуля SD16_A	SD16INCTL0	R/W	0B0h	Скидається після PUC
Регістр дозволу аналогових входів модуля SD16_A	SD16AE	R/W	0B7h	Скидається після PUC

В табл. 3.25 описані біти регістру управління входом модуля SD16_A SD16CCTL0.

Таблиця 3.25 – Регістр управління входом модуля SD16_A SD16CCTL0

Біт(и)	Позначення	Опис
15	Reserved	Зарезервований. Читається як 0.
14..13	SD16BUFx	Режим роботи буфера з високою вхідною напругою: 00 – буфер відключено 01 – низька швидкодія / малий струм 10 – середня швидкодія / великий струм 11 – висока швидкодія / великий струм
12	SD16UNI	Вибір однополярного режиму: 0 – двополярний режим; 1 – однополярний режим.
11	SD16XOSR	Розширений діапазон коефіцієнта пере дискретизації. Цей біт сумісно з бітами SD16OSRx визначає коефіцієнт передискретизації.
10	SD16SINGL	Вибір режиму однократного перетворення 0 – режим неперервного перетворення 1 – режим однократного перетворення

Продовження таблиці 3.25.

Біт(и)	Позначення	Опис
9..8	SD16OSRx	Коефіцієнт передискретизації. При SD16XOSR = 0: 00..11 – 256, 128, 64, 32 При SD16XOSR = 1: 00..01 – 512, 1024 10..11 – зарезервовано
7	SD16LSBTOG	Перемикання біту доступу до молодших бітів результату: 0 – SD16LSBACC не змінюється при зчитуванні регістру SD16MEM0 1 – Стан SD16LSBACC змінюється при кожному зчитуванні регістру SD16MEM0
6	SD16LSBACC	Цей біт дозволяє звертатися до старших чи молодших 16 біт результату перетворення АЦП: 0 – SD16MEMx містять старші 16 біт 1 – S16MEMx містять молодші 16 біт
5	SD16OVIFG	Прапорець переривань при переповненні SD16MEM0: 0 – не було запиту переривання 1 – є запит переривання
4	SD16DF	Представлення результату перетворення: 0 – змішаний двійковий код 1 – додатковий код
3	SD16IE	Дозвіл переривання модуля SD16_A: 0 – переривання заборонено 1 – переривання дозволено
2	SD16IFG	Прапорець переривання модуля SD16_A скидається автоматично при зчитуванні регістру SD16MEM0 чи може бути скинутий програмно: 0 – не було запиту переривання 1 – є запит переривання
1	SD16SC	Запуск перетворення: 0 – не починати перетворення 1 – починати перетворення
0	Reserved	Зарезервований. Читається як 0.

В табл. 3.26 описані біти регістру управління модуля SD16_A SD16CTL.

Таблиця 3.26 – Регістр управління модуля SD16_A SD16CTL

Біт(и)	Позначення	Опис
15..12	Reserved	Зарезервований. Зчитується як 0.
11..9	SD16XDIVx	Коефіцієнти ділення додаткового дільника тактового сигналу модуля SD16_A: 000 – /1 001 – /3 010 – /16 011 – /48 1xx – зарезервовано
8	SD16LP	Режим пониженого енергоспоживання. Цей біт вмикає режим роботи модуля SD16_A з пониженою швидкістю перетворення і зменшеним енергоспоживанням: 0 – режим пониженого енергоспоживання вимкнутий; 1 – режим пониженого енергоспоживання ввімкнутий. Максимальна тактова частота модуля SD16_A зменшена.
7..6	SD16DIVx	Коефіцієнт ділення основного дільника тактового сигналу модуля SD16_A: 00 – /1 01 – /2 10 – /4 11 – /8
5..4	SD16SSELx	Вибір джерела тактового сигналу модуля SD16_A: 00 – MCLK 01 – SMCLK 10 – ACLK 11 – зовнішній сигнал TACLK
3	SD16VMIDON	Ввімкнення буфера V_{MID}: 0 – буфер вимкнено 1 – буфер ввімкнено
2	SD16REFON	Ввімкнення генератора опорної напруги: 0 – генератор опорної напруги вимкнено 1 – генератор опорної напруги ввімкнено
1	SD16OVIE	Дозвіл переривання при переповненні: 0 – переривання заборонено 1 – переривання дозволено
0	Reserved	Зарезервований. Зчитується як 0.

В табл. 3.27 описані біти регістру управління входом модуля SD16_A SD16INTCTL0.

Таблиця 3.27 – Регістр управління входом модуля SD16_A SD16INTCTL0

Біти	Позначення	Опис
7..6	SD16INTDLYx	Затримка генерації переривання після запуску перетворення. Ці біти визначають затримку генерації першого переривання після запуску переривання. 00 – Переривання генерується після 4-го перетворення 01 – Переривання генерується після 3-го перетворення 10 – Переривання генерується після 2-го перетворення 11 – Переривання генерується після 1-го перетворення
5..3	SD16GAINx	Коефіцієнт підсилення вхідного підсилювача: 000..101 – ×1, ×2, ×4, ×8, ×16, ×32 110.. 111 – зарезервовано
2..0	SD16INCHx	Вибір вхідного каналу: 000 – A0 001 – A1 010 – A2 011 – A3 100 – A4 101 – A5 – (AVcc - AVss)/11 110 – A6 – датчик температури 111 – A7 – коротко замкнутий вхід для вимірювання зміщення PGA

Зміст завдання

1. Лабораторна робота присвячена дослідженню роботи модуля цифро-аналогового перетворення SD16_A.

2. Завдання містить код програми, яка реалізує логер температурних даних. В коді програми необхідно визначити значення наступних регістрів: `DCOCTL`, `BCSCTL1`, `BCSCTL3`, `SD16CTL`, `SD16CCTL0`, `SCD16INCTL0`, `TACCTL0`, `TACCR0`, `TACTL`. Після цього програму необхідно відкомпілювати, зневадити та запустити на виконання.

3. На основі температурних даних отриманих логером побудувати графік зміни температури в часі.

Код програми
Для виконання лабораторної роботи використайте наступний код:

```
/* Логер температурних даних з використанням SD16_A
*/
#include  <msp430x20x3.h>
#define InfoMemoB 0x1040
#define InfoMemoC 0x1080

unsigned int counter;  /* підрахунок WDT
                          переривань */
unsigned int min;      // лічильник хвилин
unsigned int memo_ptr; // лічильник пам'яті

// Стирання сегменту флеш-пам'яті
void erase_segment(int address)
{
  int *Flash_ptr;

  Flash_ptr = (int *)address; /* ініціалізація
                                 Flash_ptr */
  FCTL1 = _________;          /* встановлення
                                 біту Erase */
  FCTL3 = _________;          // очищення біту Lock

  *Flash_ptr = 0;     // стирання сегменту флеш

  FCTL3 = _________;  // встановлення біту Lock
}

// Запис цілого числа у флеш-пам'ять
void write_int_flash(int address, int value)
{
  int *Flash_ptr;

  Flash_ptr = (int *)address; /* ініціалізація
                                 Flash_ptr */
  FCTL3 = ________;

  FCTL1 = _________;      /* встановлення WRT біту
                          для запису */

  *Flash_ptr = value;    /* запис значення у
                          флеш-пам'ять */
```

```
  FCTL1 = _______;       // очищення біту WRT

  FCTL3 = ________;      // встановлення біту LOCK
}
// ISR SD16_A
#pragma vector=SD16_VECTOR
__interrupt void SD16ISR(void){
 unsigned int temperature;
  if (min <= 60){
    temperature = (SD16MEM0-0x8000)/84 - 232;
    write_int_flash(memo_ptr,temperature);
    memo_ptr += 2;
  }
  else
  {
      _NOP();
  }
}

// ISR Timer_A
#pragma vector=TIMERA0_VECTOR
__interrupt void TimerA0_ISR (void){
  counter++;
  P1OUT ^= 0x01;        // перемикання LED

  if (counter == ____) /* встановлення часу
                       виміру в хвилинах */
  {
  min++;
  counter = 0;

  SD16CCTL0 |= SD16SC;  /* початок SD16
                           перетворення */
  }
}

void main(void){
  WDTCTL = WDTPW + WDTHOLD;/* зупинка сторожового
                              таймера */

  if ( CALBC1_1MHZ == 0xFF || CALDCO_1MHZ == 0xFF )
  {
    while(1);
  }
```

```
  DCOCTL =  _______________;

  BCSCTL1 = _______________; // ACLK = 1.5 kHz

  BCSCTL3 = _______________; /* встановлення VLOCLK
                               (12 kHz) */

  FCTL2 = FWKEY + FSSEL0 + FN1; /* встановлення
                                   MCLK/3 */
  P1DIR |= 0x01;  /* налаштування порту P1.0 на
                   виведення */
  SD16CTL = ____________________; /* 1.2В опорна
                                    напруга, SMCLK */
  SD16INCTL0 = _______________; /* датчик
                                   температури */
  SD16CCTL0 = ________________;

  TAR = 0;        // очищення лічильника Timer A

  CCTL0 = _______________; /* CCR0 переривання
                           дозволено */
  CCR0 = ________________; /* число має
                           відповідати 1 сек. */
  TACTL = _______________; /* ACLK, режим «вгору»
                           до CCR0 */
   counter = 0;
  min = 0;

  erase_segment(InfoMemoB);
  erase_segment(InfoMemoC);

  memo_ptr = InfoMemoB;

  _BIS_SR(LPM3_bits + GIE);/* перехід в режим
                            LPM3 і очікування
                            переривання */
}
```

Порядок виконання

1. Підключіть пристрій eZ430-F2013 до USB-порту комп’ютера.
2. Запустіть середовище розробки вказане викладачем (CCE або IAR).
3. Створіть новий проект, додайте в нього файл з розширенням ".c" (див. розділ 1-2) та скопіюйте в нього код програми.

4. Код програми є неповним, використовуючи теоретичні відомості та специфікацію потрібно визначити значення регістрів.

5. У функції main() здійсніть наступні налаштування:

○ встановіть частоту DCO на рівні 1 МГц за допомогою каліброваних констант:

```
DCOCTL =_____________;
```

○ налаштуйте конфігурацію регістру BCSCTL1 для встановлення дільника ACLK - /8:

```
BCSCTL1 = _________________;
```

○ налаштуйте конфігурацію регістру BCSCTL3 для вибору у якості низькочастотного джерела тактового сигналу LFXT1 – VLOCLK:

```
BCSCTL3 = _________________;
```

○ налаштуйте конфігурацію регістру FCTL2 для вибору у якості джерела тактового сигналу контролера флеш-пам'яті MCLK, та встановлення дільника частоти MCLK/3:

```
FCTL2 =_____________;
```

○ налаштуйте конфігурацію регістру SD16CTL для ввімкнення генератора опорної напруги:

```
SD16CTL=_____________;
```

○ налаштуйте конфігурацію регістру SD16INCTL0 для вибору вхідного каналу A6 – датчик температури:

```
SD16INCTL0 = ____________;
```

○ налаштуйте конфігурацію регістру SD16CCTL0 для вибору режиму однократного перетворення та дозволу переривання модуля SD16_A.

```
SD16CCTL0 = ____________;
```

○ налаштуйте регістр Timer_A для переривання кожну секунду. Використайте тактовий сигналу ACLK як джерело тактових сигналів. Цей таймер налаштований в режимі лічильника для підра-

хунку до тих пір поки значення TAR не досягне значення TACCR0.

```
CCTL0 = _________________;

CCR0 = _________________;

TACTL = _________________;
```

6. У функції erase_segment() здійсніть наступні налаштування:
   - налаштуйте конфігурацію регістра FCTL1 для встановлення біту ERASE та конфігурацію регістра FCTL3 для очищення біту LOCK:

```
FCTL1 = _________________;

FCTL3 = _________________;
```

   - налаштуйте конфігурацію регістра FCTL3 для встановлення біту LOCK:

```
FCTL3 = _________________;
```

7. У функції write_int_flash() здійсніть наступні налаштування:
   - налаштуйте конфігурацію FCTL3 для очищення біту LOCK:

```
FCTL3 = _________________;
```

   - налаштуйте конфігурацію регістра FCTL1 для встановлення біту WRT:

```
FCTL1 = _________________;
```

   - налаштуйте конфігурацію регістра FCTL1 для очищення біту WRT та конфігурацію регістра FCTL3 для встановлення біту LOCK:

```
FCTL1 = _____________;

FCTL3 = _____________;
```

8. У функції TimerA0_ISR задайте період виміру в хвилинах (визначається викладачем).

```
if (counter == ___ )
```

9. Після компіляції проекту, розпочніть сеанс зневадження і перед запуском програми, поставте точку зупину (breakpoint) на рядку коду з _NOP(). Відкрийте властивості breakpoint та в пункті «Action» виберіть дію «Write data to file». Назвіть файл «Temp.dat» та визначте формат даних як integer. Визначте початкову адресу даних як 0x01040, та вкажіть розмір даних в залежності від періоду виміру. Запустіть програму на виконання.

Використовуйте обігрівач чи вентилятор, щоб змусити температуру коливатись протягом періоду виміру. Коли виконання досягає точки зупину, здійснюється запис значень температури за період виміру у файл.

10. Побудуйте графік зміни температури в часі в ET Excel, використавши дані отримані логером, та занесіть цей графік у звіт.

11. Закрийте середовище розробки та від'єднайте пристрій eZ430-F2013 від USB-порту комп'ютера.

Зміст звіту

- титульний аркуш;
- мета роботи;
- схема електрична функціональна модуля SD16_A;
- результати виконання;
- графік зміни температури в часі;
- висновки.

Контрольні запитання

1. Які основні особливості модуля SD16_A включені в eZ430-F2013?
2. Які функції виконує Ядро SD ADC?
3. Який діапазон аналогового входу і PGA?
4. Поясніть принцип робити модуля SD16_A використовуючи схему електричну функціональну.
5. Яким чином значення напруги перетворюється у значення температури?
6. Для чого призначений Timer_A?
7. Які ресурси використовуються додатком?
8. Назвіть регістри модуля SD16_A?
9. Для чого призначені біти SD16SSELx регістру управління модуля SD16_A SD16CTL?
10. Для чого призначені біти SD16INCHx регістру управління входом модуля SD16_A SD16INTCTL0?
11. Для чого призначені біти SD16DIVx регістру управління модуля SD16_A SD16CTL?
12. Для чого призначені біти SD16BUFx регістру управління входом модуля SD16_A SD16CCTL0?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Семенов Б.Ю. Микроконтроллеры MSP430: первое знакомство / Б.Ю. Семенов. – М.: СОЛОН-ПРЕСС, 2006. – 128 с.

2. Семейство микроконтроллеров MSP430x2xx. Архитектура, программирование, разработка приложений / пер. с англ. Евтифеева А.В. – М.: Додэка-XXI. – 2010. – 544 с.

3. MSP430. [Електронний ресурс]: Режим доступу: http:// chinapads.ru/c/s/msp430

4. MSP430. [Електронний ресурс]: Режим доступу: http:// wikipedia.org/wiki/msp430

5. Семейство MSP430: опыт применения. [Електронний ресурс]: Режим доступу:– http://www.compitech.ru/html.cgi/arhiv/01_03/stat_64.htm

6. Семейство микроконтроллеров MSP430 Texas Instruments. [Електронний ресурс] : Режим доступу: http://www. gaw.ru/ html. cgi/ txt/ic/Texas_Instruments/micros/msp430/start.htm

7. Семейство микроконтроллеров MSP430 Texas Instruments с Flash-памятью. [Електронний ресурс]: Режим доступу:http:// kazus.ru/articles/193.html

8. Практика работы с MSP430. [Електронний ресурс] / С. Борщ. – Режим доступу: http://www.chipinfo.ru/literature/chipnews/200101/44.html

9. Гук И. Краткий обзор микроконтроллеров семейства MSP430 компании Texas Instruments / И. Гук // Компоненты и технологии. – № 6. – 2006.

10. MSP430. [Електронний ресурс]: Режим доступу:http:// emproj.com/MSP430StartPage

11. Особенности программирования микроконтроллеров в среде программирования IAR Embedded Workbench . [Електронний ресурс] : Режим доступу: http:// www.support17.com/ component/ content/ 248.html? task=view

12. Охотин А.П. Семейство MSP430: опыт применения [Електронний ресурс]: Рынок микроэлектроники. – Режим доступу: http:// www.compitech.ru/html.cgi /arhiv/01_03/ stat_64.htm

13. Архитектура MSP430 [Електронний ресурс]: Рынок микроэлектроники. – Режим доступу: http:// www. gaw.ru/ html.cgi /txt /doc /micros/msp430/arh/1.htm

14. Texas Instruments. MSP430FG4618/F2013 Experimenter’s Board: User’s Guide [Електронний ресурс]: Режим доступу:http:// www.ti.com/litv/pdf/slau213a

15. Средства разработки для микроконтроллеров MSP430 [Електронний ресурс]: Компэл. – Режим доступу: http://mcu.compel.ru/arcticle/11

16. Texas Instruments. Руководство пользователя по компилятору CCE [Електронний ресурс]: Режим доступу: http://focus.ti.com /lit/ug/ slau132c/slau132c.pdf

17. MSP430. Система команд [Електронний ресурс]: Режим доступу: http://www.gaw.ru/html.cgi/txt/doc/micros/msp430/asm/start.htm

18. Микроконтроллеры MSP430: отличительные особенности семейства MSP430x2xx от MSP430x1xxr [Електронний ресурс]: Режим доступу: http://www.compitech.ru/html.cgi/arhiv/06_06/stat_116.htm

19. MSP430x2xx Family User's Guide. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2008. – 693с. – Режим доступу: http://www.ti.com/lit/ug/slau144i/slau144i.pdf

20. Микроконтроллеры семейства MSP430 [Електронний ресурс]: Режим доступу:http://www.gaw.ru/html.cgi/txt/doc/micros/msp430/index.htm

21. MSP430 Assembly Language Tools v 4.1 User's Guide. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2012. – 284с. – Режим доступу: http:// www.ti.com/ general/docs/ lit/getliterature.tsp? literatureNumber=slau131g&fileType=pdf

22. MIXED SIGNAL MICROCONTROLLER. [Електронний ресурс]: Режим доступу: www.ti.com/lit/ds/slas774a/slas774a.pdf

23. Ultra-low Power Motion Detection using the MSP430F2013. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2005. – 6с. – Режим доступу: https://www.olimex.com/Products/MSP430/Starter/MSP430-PIR/resources/slaa283.pdf

24. Code composer Studio (CCStudio) Integrated Developent Environment (IDE). [Електронний ресурс]: Режим доступу: http:/ /www.ti.com/tool / ccstudio

25. Power Management Solutions for Ultra-Low-Power 16-Bit MSP430 TM MCUs. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2012. – 3с. – Режим доступу: www.ti.com/lit/sg/slyt345d/slyt345d.pdf

26. Migrating from the MSP430F2xx and MSP430G2xx Families to the MSP430FR58xx and MSP430FR59xx Family. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2012. – 14с. – Режим доступу: www. ti. com/ it/an/slaa559/slaa559.pdf

27. MSP430F2013 Device Erratasheet. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2012. – 11с. – Режим доступу: www. ti. com /lit/er/slaz156a/slaz156a.pdf

28. MSP430 Assembly Language Tools v 3.3 User's Guide. [Електронний ресурс]: Austin: Texas Instruments Incorporated. – 2010. – 280с. – Режим доступу: www.ti.com/lit/ug/slau131e/slau131e.pdf

СЛОВНИК НАЙБІЛЬШ ВЖИВАНИХ ТЕРМІНІВ

Адреса – address
Аналоговий вхід – analog input
Апаратні засоби – hardware
Архіватор - arhiver
Асемблер - assembler
Байт – byte
Біт – bit
Вбудувана система – embedded system
Введення/виведення – input/output
Вектор переривання – interrupt vector
Вивід – pin
Вихід – output
Вхід – input
Генератор – generator
Дільник частоти – frequencies divider
Додаток – application
Емулятор – emulator
З'ємна плата – detachable target board
Зневадження – debugging
Зневаджувальний інтерфейс – debugging interface
Зневаджувач – debugger
Інструмент розробника – development tool
Інтегроване середовище розробки – integrated development environment
Інтервальний таймер – interval timer
Інтерфейс – interface
Інформаційна пам'ять – information memory
Кварцовий генератор – crystal oscillator
Код програми – source code
Команда – command
Комп'ютер – computer
Компілятор – compilator
Компіляція – compilation
Компонувальник – linker
Конденсатор – capacitor
Лабораторна робота – laboratory work
Лічильник – counter
Мікроконтролер – microcontroller
Модуль – module
Модуль синхронізації – synchronization module
Наднизьке споживання – ultra-low power
Напруга живлення – supply voltage

Немасковане переривання - non maskable interrupt
Об’єктні файли – object files
Обробка даних – data processing
Опорна напруга – reference voltage
Переривання – interrupt
Периферійний пристрій – peripheral device
Порт – port
Прапорець –flag
Пристрій - device
Програма – program
Програмні засоби – software
Проект – project
Процедура – procedure
Реальний час – real time
Регістр – register
Режим захоплення – capture mode
Режим порівняння – compare mode
Режими роботи – operation modes
Резистор змінного опору – pull-up/pull-down resistor
Резонатор – resonator
Світлодіод - LED
Сегмент – segment
Сигнал – signal
Синхронізація – synchronization
Система переривань – interrupt system
Стек – stack
Сторожовий таймер - watchdog timer
Тактовий генератор – clock generator
Тактовий сигнал – clock signal
Точка зупину – breakpoint
Транзистор – transistor
Флеш-пам'ять – flash memory
Фронт сигналу – signal front
Цифро-аналогове перетворення – digital-to-analog conversation
Цифровий вивід – digital pin
Цифровий фільтр – digital filter
Частота сигналу – signal frequency
Швидкість – speed