БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 1
Тема: «Знакомство с CCS. Цифровой ввод-вывод»
Вариант №8

Выполнил:
студент группы 150503 Ходосевич М.А.

Проверил:
ассистент каф. ЭВМ _____ Шеменков В.В.

Минск
2024

1. Постановка задачи

Написать программу по управлению цифровым вводом-выводом в соответствии с заданием варианта. Задание 8 варианта: реализовать режим потухания с использованием LED1-LED3 и S2. При нажатии на кнопку все диоды включаются, при отпускании - гаснут друг за другом с небольшой задержкой.

2. Теоретические сведения

Экспериментальная плата MSP-EXP430F5529 разработана на основе микроконтроллера MSP430F5529 компании Texas Instruments. Это серия процессоров для обработки смешанных сигналов со сверхнизким энергопотреблением.

Основные особенности архитектуры:

– 16-разрядная ортогональная RISC архитектура;

– Фон-Неймановская адресная шина общей памяти и шина данных памяти;

– 27 (51) команд + 37 расширенных инструкций (20-бит адрес) + 11 адресных инструкций (20-бит операнды, но ограничения в режимах адресации);

– 7 согласованных способов адресации;

– полный программный доступ к регистрам, включая счетчик команд (PC), регистр состояния (SR), указатель стека (SP);

– однотактные регистровые операции;

– большой размер регистрового файла, уменьшающий количество обращений к памяти;

– 20-битная шина адреса, 16-битная шина данных;

– генератор констант (6);

– пересылки память-память без промежуточного сохранения в регистре;

– гибкая система тактирования;

– несколько режимов пониженного энергопотребления;

– моментальный переход в активный режим (порядка 6 мкс).

Микроконтроллер обладает следующими характеристиками:

– производительность до 25 MIPS;

– напряжение питания 1,8-3,6 В;

– ток утечки вывода 50 нА;

– потребление в режиме хранения данных 0,1 мкА;

– потребление в режиме часов реального времени 2,5 мкА.

Микроконтроллер включает в свой состав:

– флеш-память 128 Кб, SRAM 8 Кб;

– 80 выводов, 63 линии входа/выхода;

– 4 асинхронных 16-разрядных таймера/счетчика (7,5,3,3 регистров захвата соответственно);

– сторожевой таймер (WDT) и таймер часов реального времени (RTC);

– модуль управления питанием PMM с блоками защиты от падений напряжения (BOR) и контроля напряжения питания (SVS);

– универсальный последовательный коммуникационный интерфейс USCI 2 x UART/LIN/IrDA/SPI + 2 x I2C/SPI;

– 3 канала DMA;

– умножитель-накопитель MPY 32 x 32 бита;

– компаратор;

– 12 разрядный АЦП (ADC 12A), 16 каналов;

– полноскоростной USB 2.0 (12Мб/с), до 8 линий в/в со встроенным 3,3 В стабилизатором (питание от 5 В шины, обеспечивает ток 12 мА);

– интерфейс для измерения линейных и угловых перемещений (SIF);

– LCD контроллер до 128 сегментов;

– внутренний генератор частоты с цифровым управлением.

Обобщенная архитектура микроконтроллера представлена на рис. 2.1. Элементы архитектуры микроконтроллера будут описаны по мере выполнения лабораторных работ. Внешний вид экспериментальной платы представлен на рис. 2.2, а назначение основных элементов - на рис. 2.3.

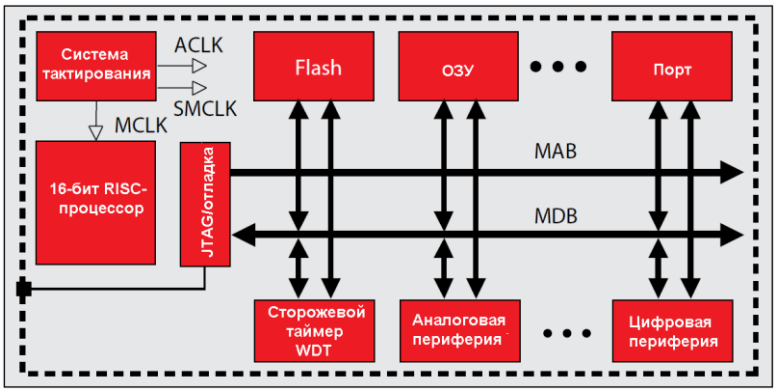

Рисунок 2.1 – Архитектура микроконтроллера MSP430

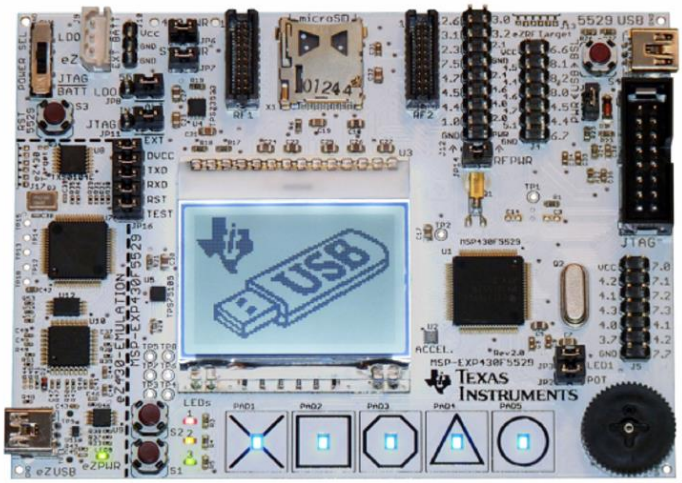

Рисунок 2.2 – Внешний вид экспериментальной платы

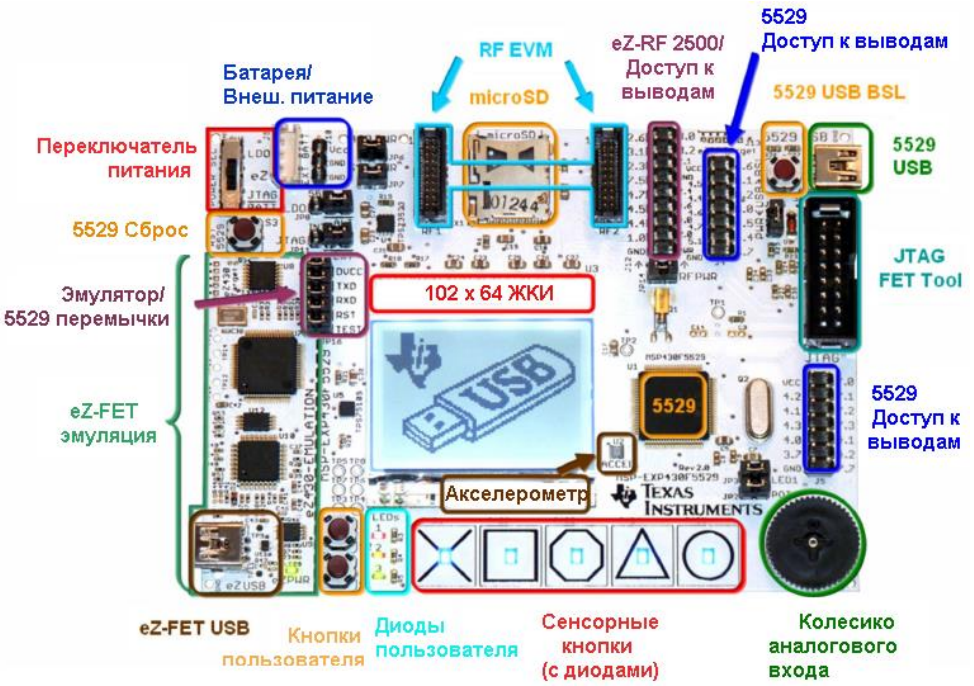

Рисунок 2.3 – Назначение элементов экспериментальной платы

Плата MSP-EXP430F5529 подключается к USB-порту ПК через разъем ezUSB платы. При исследовании возможностей экспериментальной платы для управления меню будут использоваться пользовательские кнопки и колесико.

8-разрядные порты P1, P2, P3,…,P8, PJ управляют выводами контроллера. Выводы программируются либо как I/O, либо как вход/выход периферии. Порты могут объединяться в пары: P1 и P2 = PA, P3 и P4 = PB, P5 и P6 = PC, P7 и P8 = PD. При работе с прерываниями порты в пары не объединяются. Для порта могут быть доступны регистры:

PxIN – чтение данных с вывода;

PxOUT – установка значения выхода;

PxDIR – выбор направления: 0 – вход, 1 – выход;

PxREN – разрешение подтягивающего резистора;

PxDS – выбор допустимой силы вывода;

PxSEL – выбор функции вывода: 0 – I/O, 1 – периферия;

PxIV – генерирует значение для изменения счетчика команд, соответствующее прерыванию с максимальным приоритетом;

PxIES – выбор направления перепада для генерации запроса на прерывание: 0 – по фронту, 1 – по спаду;

PxIE – разрешение прерывания;

PxIFG – флаг прерывания.

Пользователю программно доступны две кнопки S1 и S2, подключенные соответственно к выводу 7 порта 1 и выводу 2 порта 2 (см. рис. 2.4). В дальнейшем такое подключение будем обозначать как P1.7 и P2.2 соответственно. Также программно доступны 8 светодиодов, три из которых (LED1 – LED3, см. рис. 2.4) размещены рядом с кнопками и подключены

соответственно к выводам P1.0, P8.1, P8.2. Еще 5 светодиодов (LED4 – LED8) размещаются в блоке сенсорных кнопок и подключены к выводам P1.1 – P1.5 соответственно.

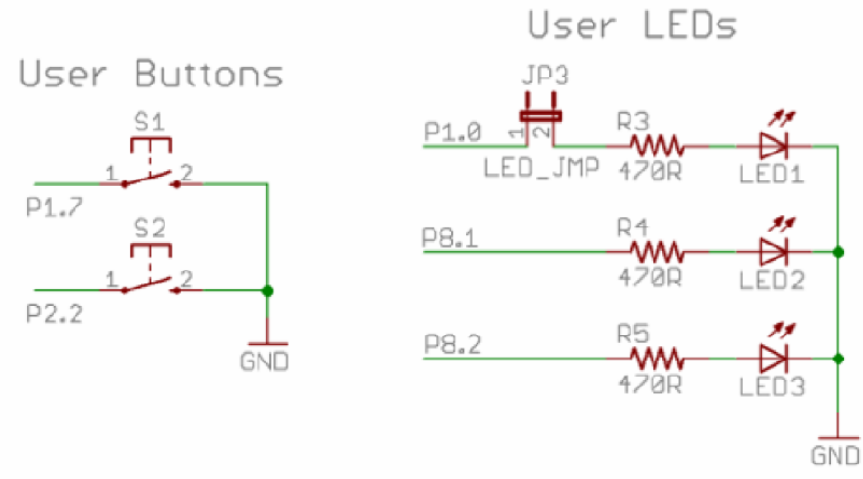

Рисунок 2.4 – Подключение пользовательских кнопок и светодиодов

При написании кода следует учесть несколько моментов. Вначале следует подключить заголовочный файл msp430.h, который в свою очередь подключает файл msp430f5529.h, содержащий необходимые константы в соответствии с архитектурой контроллера. Далее, поскольку после сброса запускается сторожевой таймер, его следует отключить (иначе через какое-то время сработает сброс).

Константы и определения заданы как для портов, так и для отдельных полей и их значений. Поэтому работа с портами становится максимально удобной для программиста. Так, например, запись P8DIR |= BIT2; означает, что в порт P1DIR, отвечающий за выбор направления выводов порта 1, заносится новое значение, которое получено логическим ИЛИ его текущего состояния и бита 2. Фактически, это устанавливает бит 2 в заданном порту.

Следует обратить внимание, что при наименовании констант использовались следующие принципы:

– константа, соответствующая биту поля-флага именуется по имени поля, например, полю CPUOFF регистра состояния процессора SR (бит 4) соответствует константа CPUOFF;

– константа соответствующая биту n в поле NNN именуется NNNn;

– константа, соответствующая номеру x выбранного варианта для поля NNN именуется NNN_x;

– константа, соответствующая выбранному режиму zz для поля NNN именуется NNN__zz.

Так, например, для 3-битного поля SELA, константа, соответствующая 0 биту поля, именована SELA0, вариант выбора 0 (SELA = 000) именован SELA_0, а режим, соответствующий данному варианту именован SELA__XT1CLK. В некоторых случая поля задают делители либо множители,

соответствующие степени двойки. Тут надо быть особо внимательным и не спутать похожие мнемоники, например, NN4 (четвертый бит, т.е. 10000), NN_4 (четвертый вариант, т.е. 00100), NN__4 (режим деления на 4, т.е. 00011).

3. Листинг программы

```
#include <msp430.h>

int button_pressed(int button_status) {
    if(!(P2IN & BIT2)) {            //если кнопка нажата
        if(!button_status) {
            button_status = 1;      //переключение состояния кнопки
        }
    } else {
        if(button_status) {
            button_status = 0;      //сброс состояния кнопки
        }
    }

    return button_status;
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD; //отключаем сторожевой таймер

    //настройка портов для кнопок и светодиодов
    P1DIR |= BIT0;  //устанавливаем P1.0 как выход(LED1)
    P8DIR |= BIT1;  //устанавливаем P8.1 как выход(LED2)
    P8DIR |= BIT2;  //устанавливаем P8.2 как выход(LED3)
#include <msp430.h>

int button_pressed() {
    return !(P2IN & BIT2); // возвращаем 1, если кнопка нажата
}

void delay_ms(unsigned int ms) {
    while (ms--) {
        __delay_cycles(1000);  // задержка в 1 миллисекунду при 1 МГц
тактовой частоте
    }
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD; // отключаем сторожевой таймер

    // настройка портов для светодиодов
    P1DIR |= BIT0;  // устанавливаем P1.0 как выход (LED1)
    P8DIR |= BIT1;  // устанавливаем P8.1 как выход (LED2)
    P8DIR |= BIT2;  // устанавливаем P8.2 как выход (LED3)
    P2DIR &= ~BIT2; // устанавливаем P2.2 как вход (S2)

    P2REN |= BIT2;  // включаем подтягивающий резистор для P2.2 (S2)
    P2OUT |= BIT2;  // подтягиваем P2.2 к Vcc

    // Начальное состояние: выключаем все светодиоды
    P1OUT &= ~BIT0;
    P8OUT &= ~BIT1;
    P8OUT &= ~BIT2;

    while (1) {
        if (button_pressed()) {  // если кнопка нажата
```

```
            // Зажигаем все три светодиода
            P1OUT |= BIT0;  // включаем LED1
            P8OUT |= BIT1;  // включаем LED2
            P8OUT |= BIT2;  // включаем LED3
            delay_ms(500);   // задержка 500 мс

        } else {  // если кнопка отпущена
            // Последовательно гасим светодиоды с задержкой
            P1OUT &= ~BIT0;  // выключаем LED1
            delay_ms(500);   // задержка 500 мс
            P8OUT &= ~BIT1;  // выключаем LED2
            delay_ms(500);   // задержка 500 мс
            P8OUT &= ~BIT2;  // выключаем LED3
        }
    }
}
```

4. Заключение

В ходе выполнения лабораторной работы удалось ознакомиться с интегрированной средой разработки Code Composer Studio и с основными функциональными возможностями платы MSP-EXP430F5529. Удалось написать программу по управлению цифровым вводом-выводом (светодиодами и кнопками) в соответствии с вариантом №8.