БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 1
Тема: «Среда разработки Code Composer Studio. Плата MSP-EXP430F5529. Цифровой ввод-вывод»

Вариант №7

Выполнил:

.

Проверил:

Минск
2024

1. ПОСТАНОВКА ЗАДАЧИ

Написать программу по управлению цифровым вводом-выводом в соответствии с заданием варианта.

2. АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Разработка проектов для лабораторного макета MSP-EXP430F5529 проводится в интегрированной среде разработки Code Composer Studio.

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

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 рисунке 1.1.

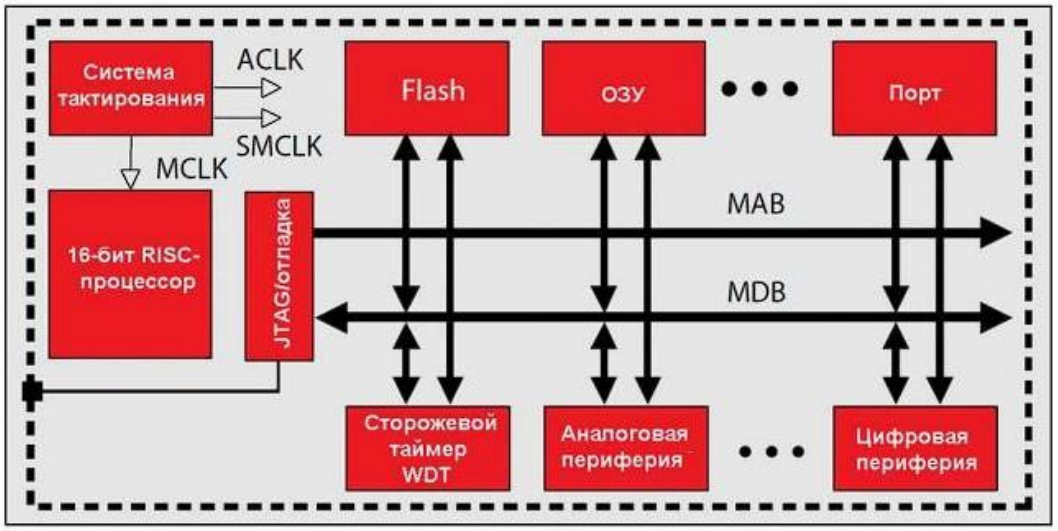

Рисунок 2.1 – Обобщенная архитектура микроконтроллера

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.

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.

Чтобы открыть проект примера Code Composer Studio требуется в среде разработки в меню выбрать Project → CCS Example Projects (либо Project → Import Existing CCS Eclipse Project). В открывшейся вкладке в дереве поиска необходимо выбрать серию устройств (MSP430ware → Devices → MSP430F55xx/6xx → Code Examples → MSP430FF552x), выбрать требуемый пример, и в выпавшем списке выбрать 1 устройство MSP430F5529. При выборе Eclipse проекта, в открывшемся окне следует нажать кнопку Browse... и выбрать папку, содержащую проект (MSP-EXP430F5529 User Experience_16KB_Cut). После выбора проекта он отображается слева в поле Project Explorer. Двойной щелчок по нужному файлу открывает его в редакторе. В редакторе исходного кода при удержании клавиши Ctrl и при помощи щелчка мыши на вызове функции или использовании переменной можно перейти к ее объявлению или определению. Сборка проекта может быть запущена двумя способами: - используя контекстное меню по щелчку правой кнопкой мыши на папке проекта в поле Project Explorer; 3 - щелчок мышью на панели инструментов по кнопке Build (на рис. 1.4 показан стрелкой). Запись собранного проекта на контроллер и запуск на выполнение в режиме отладки выполняется вызовом из основного меню среды разработки Run → Debug или на панели инструментов кнопкой Debug. После прошивки и запуска CCS переходит в режим отладки, открывая соответствующее окно. Продолжить выполнение, войти внутрь вызываемой функции, выйти на уровень выше или остановить выполнение программы можно используя кнопки отладки на панели приложения. Точки останова ставятся в окне отладки слева от кода двойным щелчком. После остановки выполнения среда возвращается в режим редактирования.

Загрузим Code Composer Studio. Изучим интерфейс пользователя и основные возможности. В соответствии с вариантом загрузим и скомпилируем демонстрационный пример, загрузим его в микроконтроллер. Загрузим прошивку MSP-EXP430F5529 User Experience. Изучим возможности экспериментальной платы. В соответствии с вариантом, не используя прерываний и таймеров, запрограммируем кнопки и светодиоды.

3. ЛИСТИНГ ПРОГРАММЫ

```
#include <msp430.h> // Подключаем заголовочный файл MSP430

#define LED1 BIT0   // Светодиод 1 подключен к P1.0
#define LED2 BIT1   // Светодиод 2 подключен к P8.1
#define LED3 BIT2   // Светодиод 3 подключен к P8.2
#define BUTTON BIT7  // Кнопка подключена к P1.7

void delay(unsigned int cycles); // Прототип функции задержки

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;     // Останавливаем watchdog таймер

    // Настройка портов для светодиодов
    P1DIR |= LED1;                // Устанавливаем LED1 как выход
    P1OUT &= ~LED1;               // Изначально LED1 выключен

    P8DIR |= LED2 + LED3;         // Устанавливаем LED2 и LED3 как выходы
    P8OUT &= ~(LED2 + LED3);      // Изначально LED2 и LED3 выключены

    // Настройка кнопки
    P1DIR &= ~BUTTON;             // Устанавливаем кнопку как вход
    P1REN |= BUTTON;             // Включаем внутренний резистор подтяжки
кнопки
    P1OUT |= BUTTON;              // Подтягиваем резистор к Vcc

    unsigned char button_state = 0; // Переменная для отслеживания
состояния кнопки

    while (1)
    {
        if ((P1IN & BUTTON) == 0) // Проверяем, нажата ли кнопка (низкий
уровень)
        {
            __delay_cycles(20000); // Задержка для антидребезга

            if ((P1IN & BUTTON) == 0) // Дополнительная проверка кнопки
            {
                button_state ^= 1; // Инвертируем состояние кнопки

                if (button_state == 1) // Если кнопка нажата первый раз
                {
                    P1OUT |= LED1;     // Включаем LED1
                    delay(10000);      // Задержка
                    P8OUT |= LED2;     // Включаем LED2
                    delay(10000);      // Задержка
                    P8OUT |= LED3;     // Включаем LED3
                }
                else // Если кнопка нажата повторно
                {
                    P1OUT &= ~LED1;    // Выключаем LED1
                    P8OUT &= ~(LED2 + LED3); // Выключаем LED2 и LED3
                }

                while ((P1IN & BUTTON) == 0); // Ожидаем, пока кнопку
отпустят
            }
        }
    }
```

```
// Функция для создания задержки
void delay(unsigned int cycles)
{
    while (cycles--)
    {
        __no_operation(); // Ничего не делаем, просто ждем
    }
}
```

4. ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был установлен Code Composer Studio. Изучен интерфейс пользователя и основные возможности. В соответствии с вариантом загружен и скомпилирован демонстрационный пример, загрузили его в микроконтроллер. Прошивку MSP- EXP430F5529 User Experience загружена. Изучены возможности экспериментальной платы. В соответствии с вариантом, не используя прерываний и таймеров, запрограммированы кнопки и светодиоды.