Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ЭВМ

Лабораторная работа № 1
«Цифровой ввод/вывод»
Вариант №13

Выполнил:
ст. гр. 050503 Кириллов В.И.

Проверил:
ассистент Шеменков В.В.

Минск 2023

1. Цель работы

Цели работы:

– ознакомиться с интегрированной средой разработки Code Composer Studio;

– ознакомиться с основными функциональными возможностями платы MSP-EXP430F5529;

– изучить приемы работы с цифровыми портами ввода/вывода;

–

2. Исходные данные

Для выполнения работы используется плата MSP-EXP430F5529 и интегрированная среда разработки Code Composer Studio.

Необходимо выполнить задание варианта № 13.

3. Теоретические сведения

3.1. Интегрированная среда разработки Code Composer Studio

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

Подробную информацию и руководства можно найти на сайте компании-разработчика Texas Instruments: http://www.ti.com/tool/ccstudio. Здесь будут рассмотрены только основные приемы работы.

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):

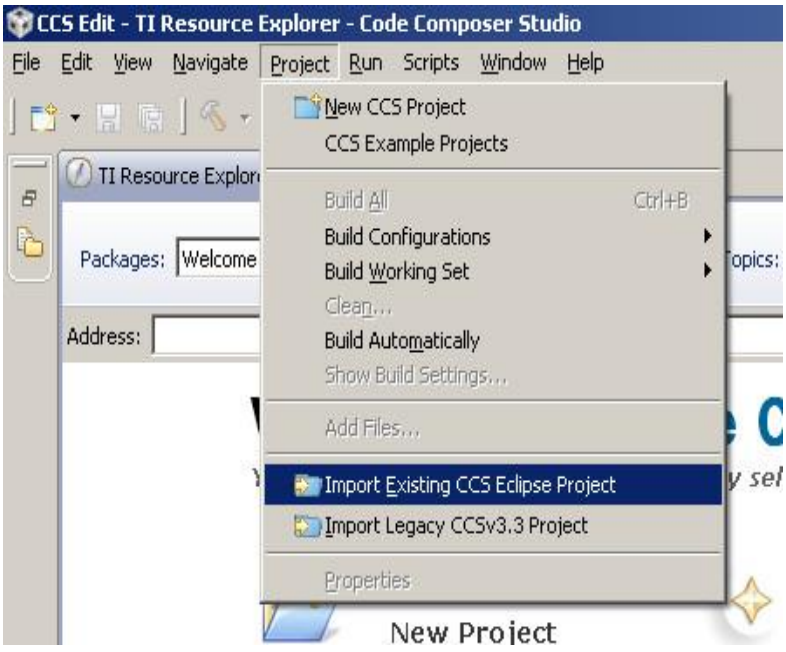

Рис. 1.1 Выбор проекта с примером

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 устройство MSP430F5529:

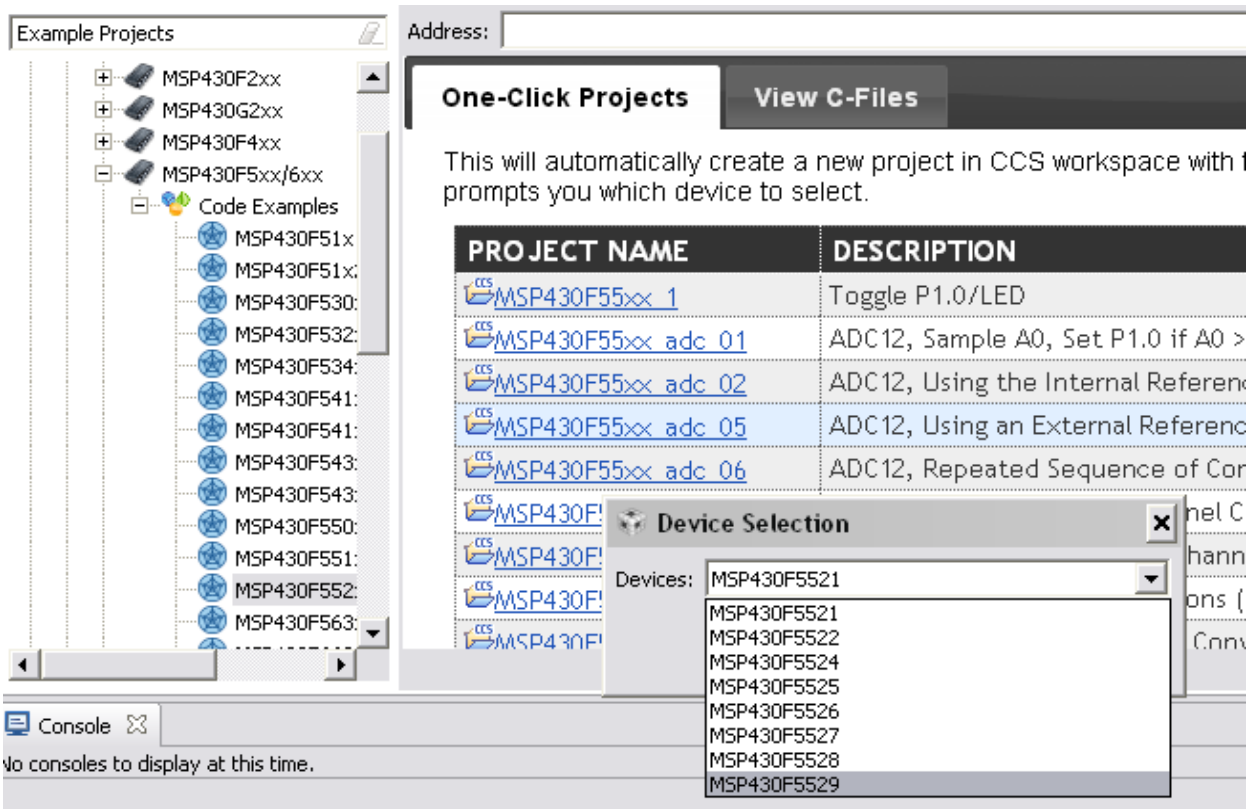

Рис. 1.2 Выбор примера и устройства

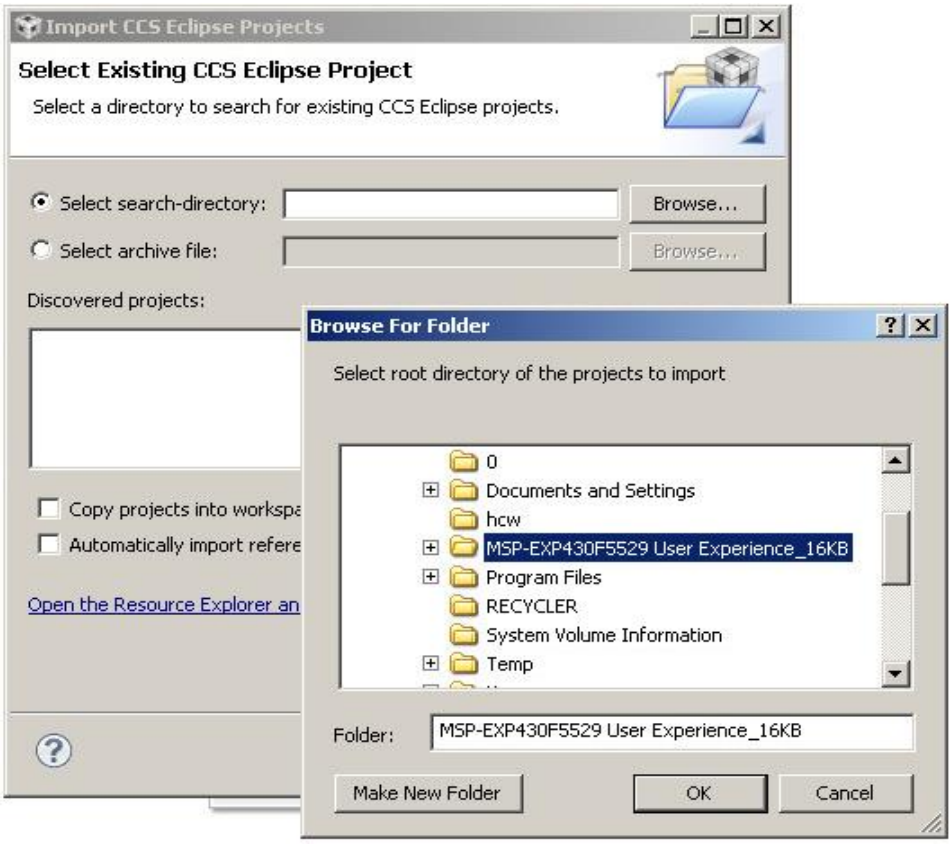

Рис. 1.3 Выбор Eclipse проекта

При выборе Eclipse проекта, в открывшемся окне следует нажать кнопку Browse... и выбрать папку, содержащую проект (MSP-EXP430F5529 User Experience_16KB_Cut), как показано на рис. 1.3.

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.

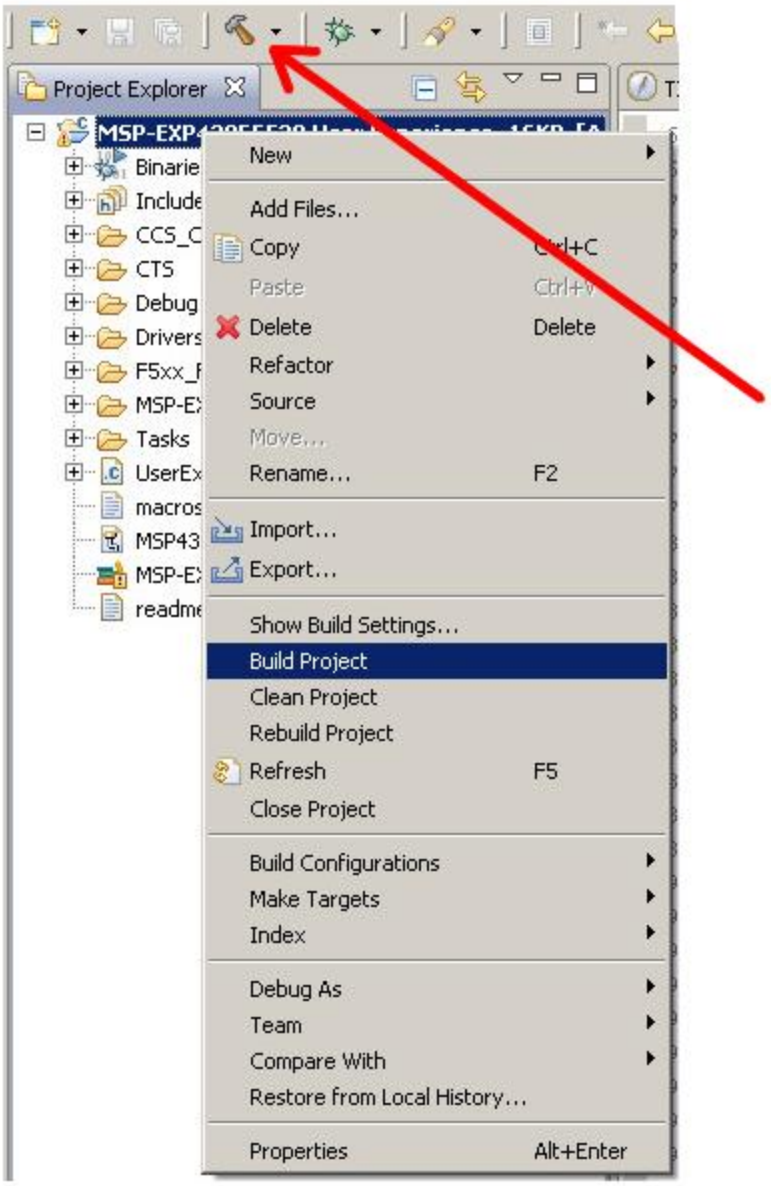

Рис. 1.4 Два варианта сборки проекта

Сборка проекта может быть запущена двумя способами:

- используя контекстное меню по щелчку правой кнопкой мыши на папке проекта в поле Project Explorer (рис. 1.4);

- 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 (рис. 1.5).

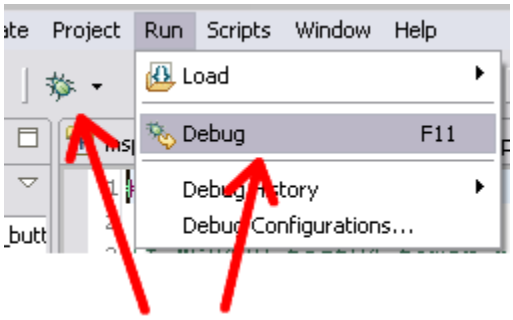

Рис. 1.5 Запуск загрузки программы в микроконтроллер

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.

3.2. Плата MSP-EXP430F5529

Подробную информацию и руководства можно найти на сайте компании-разработчика Texas Instruments: http://www.ti.com/tool/msp-exp430f5529. Здесь будут рассмотрены только основные возможности.

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. 1.6. Элементы архитектуры микроконтроллера будут описаны по мере выполнения лабораторных работ. Более подробную информацию можно найти на http://www.ti.com/product/msp430f5529. Внешний вид экспериментальной платы представлен на рис. 1.7, а назначение основных элементов - на рис. 1.8.

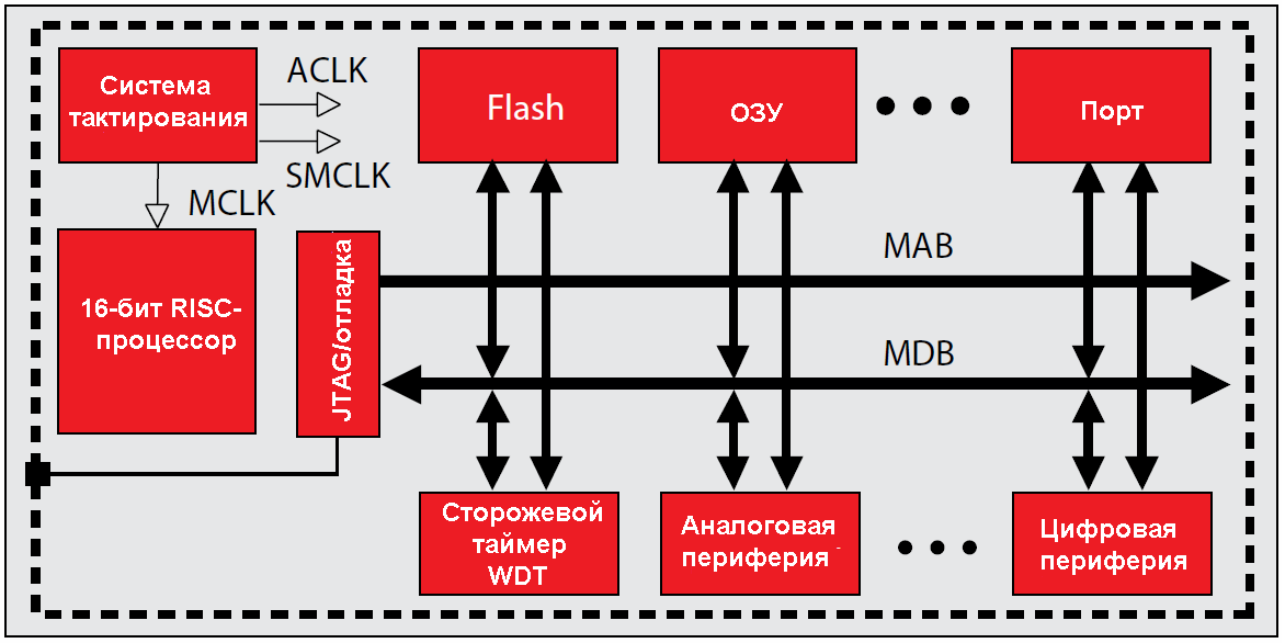

Рис. 1.6 Архитектура микроконтроллера MSP430

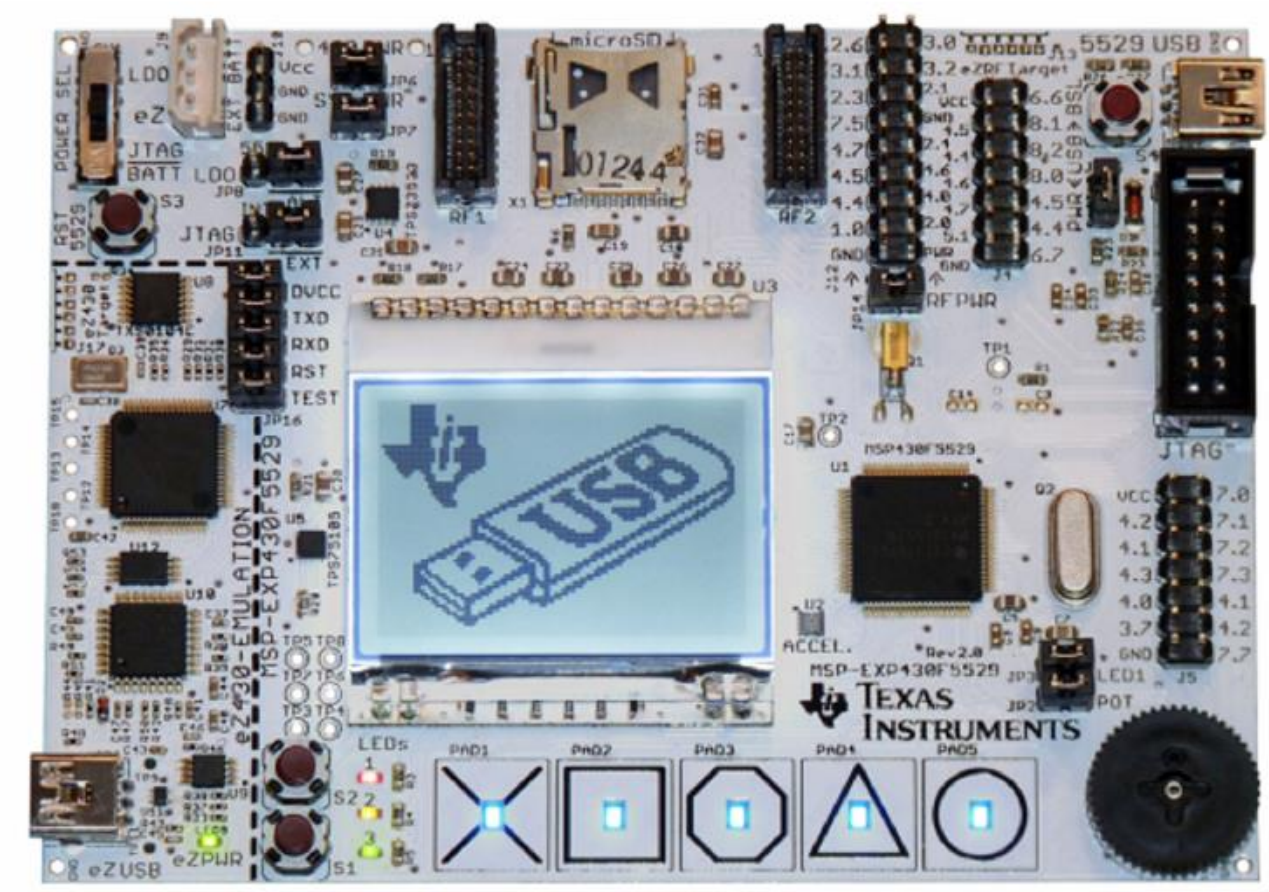

Рис. 1.7 Внешний вид экспериментальной платы MSP-EXP430F5529

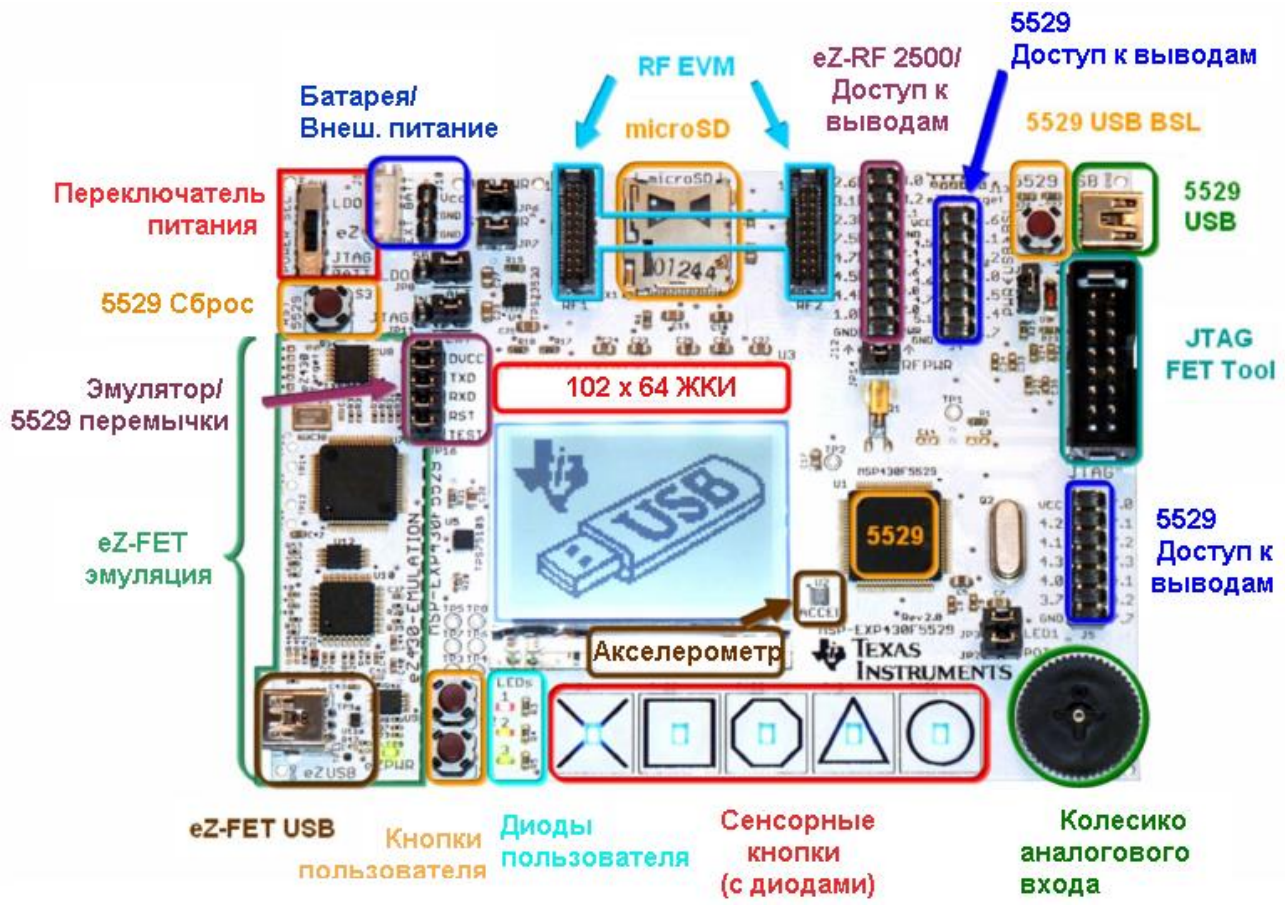

Рис. 1.8 Назначение элементов экспериментальной платы MSP-EXP430F5529

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.

3.3. Цифровой ввод-вывод

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.

Адреса соответствующих портов представлены в таблице:

Таблица 1.1 — Адреса портов ввода-вывода

№ порта	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**J**
База	200h		0220h		0240h		0260h		0320h
PxIN		1	0	1	0	1	0	1	0
PxOUT	2	3	2	3	2	3	2	3	2
PxDIR	4	5	4	5	4	5	4	5	4
PxREN	6	7	6	7	6	7	6	7	6
PxDS	8	9	8	9	8	9	8	9	8
PxSEL	A	B	A	B	A	B	A	B	-
PxIV	E	1E	-	-	-	-	-	-	-
PxIES	18	19	-	-	-	-	-	-	-
PxIE	1A	1B	-	-	-	-	-	-	-
PxIFG	1C	1D	-	-	-	-	-	-	-

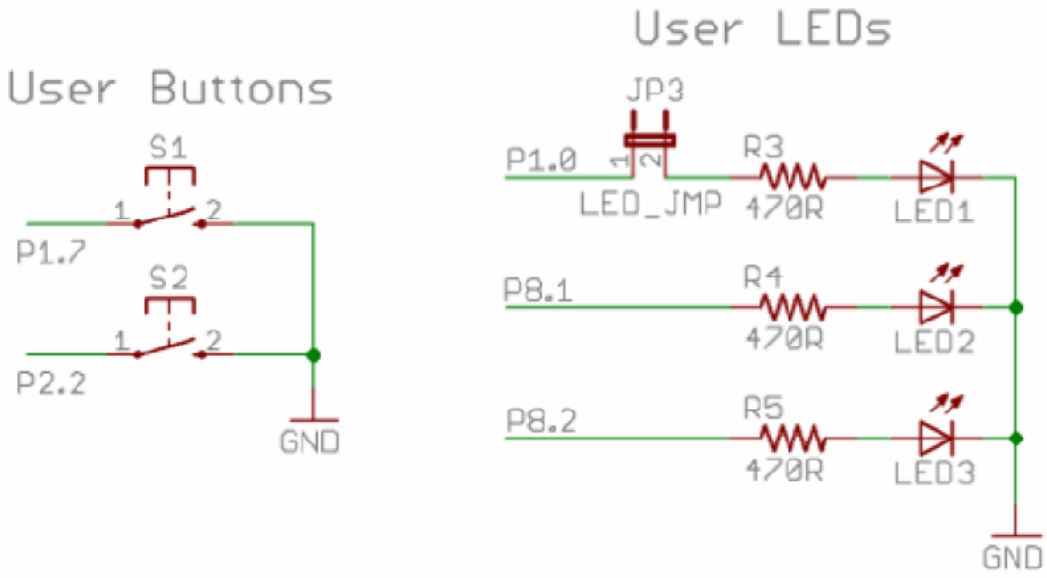

Рис. 1.9 Подключение пользовательских кнопок и светодиодов

Пользователю программно доступны две кнопки S1 и S2, подключенные соответственно к выводу 7 порта 1 и выводу 2 порта 2 (см. рис. 1.9). 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. 1.9) 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.

Логика управления выводом на примере порта 1 представлена на рисунке ниже. Для других портов схемотехника может несколько отличаться, в зависимости от особенностей подключаемой к выводу периферии микроконтроллера.

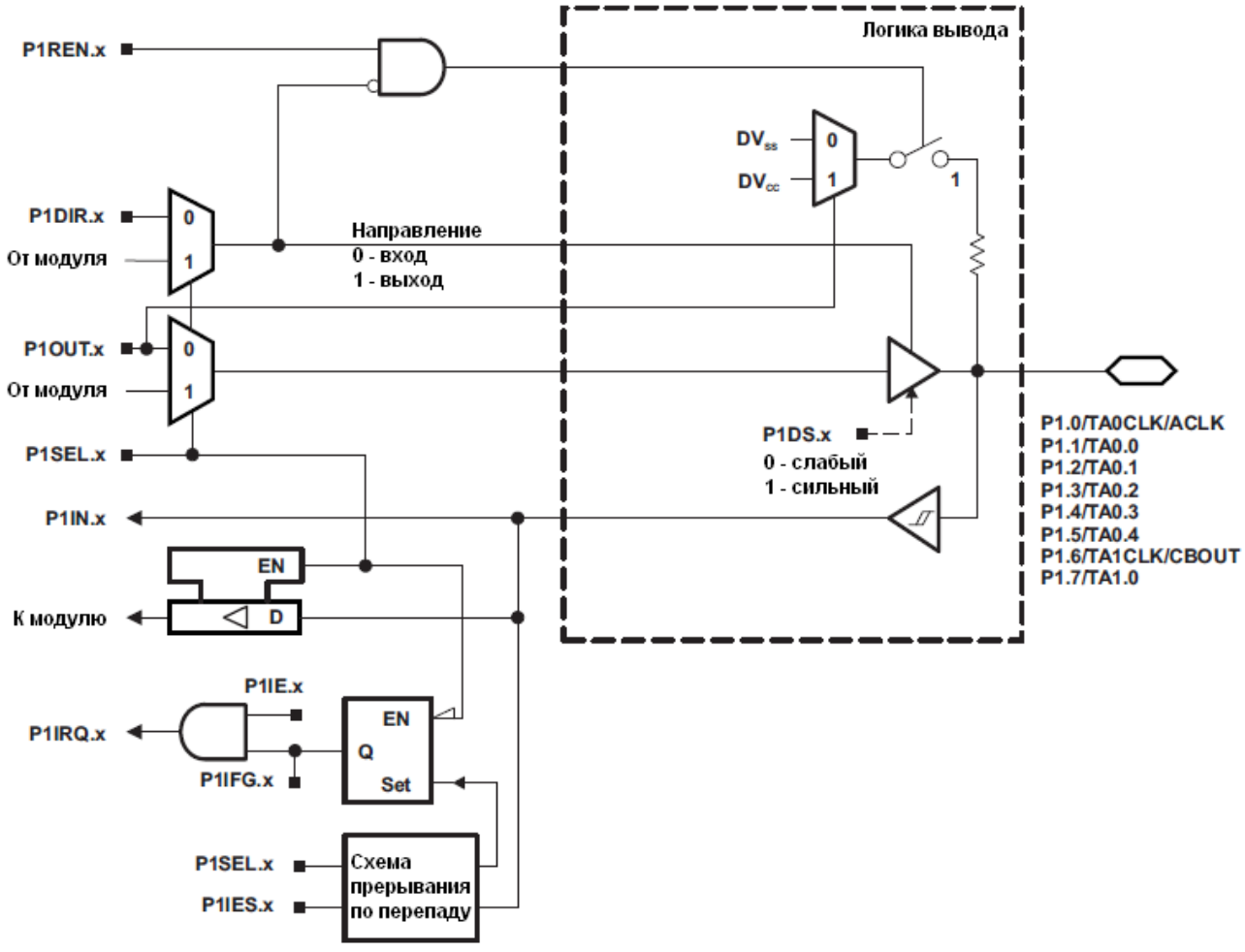

Рис. 1.10 Организация входа-выхода с триггером Шмидта на примере порта 1

Следует обратить внимание, что регистр PxOUT управляет подключением подтягивающего резистора, если вывод сконфигурирован как цифрой I/O, направление — выход, и разрешен подтягивающий резистор. В случае, если вывод сконфигурирован как вывод периферии

микроконтроллера, прерывания не генерируются. Отметим также, что после сброса цифровые выводы конфигурируются на вход, кроме того запускается сторожевой таймер в сторожевом режиме.

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).

4. Выполнение работы

4.1. Задание

При нажатии на кнопку S1 включается очередной диод (LED1 - LED3). Если все диоды горят — то при нажатии на кнопку S1 они гаснут. Нажатие и отжатие кнопок должны обрабатываться корректно:

- одно нажатие должно обрабатываться, как только одно нажатие (аналогично с отжатием);

- если было несколько нажатий, ни одно не должно быть пропущено (аналогично с отжатием).

Программа должна быть написана:

- без использования прерываний;

Не допускается подключение к проекту каких-либо файлов, за исключением:

- “msp430.h”;
- библиотек языка С;
- написанных самостоятельно.

4.2. Программа по управлению цифровым вводом-выводом без использования прерываний

```
#include <msp430.h>

#define LED1 BIT0
#define LED2 BIT1
#define LED3 BIT2
#define BUTTON BIT7

#define MIN_PRESSED_AMOUNT 7

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

    // устанавливаем пины LED1, LED2 и LED3 как выходы
    P1DIR |= LED1;
    P8DIR |= (LED2 | LED3);
    P8OUT &= ~(LED2 | LED3);

    // выключаем все светодиоды
    P1OUT &= ~LED1;
    P8OUT &= ~(LED2 | LED3);

    // устанавливаем пин BUTTON как вход
    P1DIR &= ~BUTTON;
    // включаем подтягивающий резистор на пине BUTTON
    P1REN |= BUTTON;
    // устанавливаем подтягивающий резистор на пине BUTTON
    P1OUT |= BUTTON;

    unsigned int led_num = 0;
    unsigned int button_check = 0;
    unsigned int i = 0;

    while(1)
    {
      // цикл проверки на корректность нажатия
      for (i = 0; i < 10; i++) {
            // если кнопка нажата
            if (!(P1IN & BUTTON)) {
                  button_check++;
            }
      }
```

```
        if (button_check >= MIN_PRESSED_AMOUNT)  // если кнопка нажата
        {
             __delay_cycles(100000);
             // если кнопка все еще нажата
            if (!(P1IN & BUTTON))
            {
             // зажигаем LED1
                if (led_num == 0)
                {
                    P1OUT |= LED1;
                    led_num++;
                }
                // зажигаем LED2
                else if(led_num == 1)
                {
                    P8OUT |= LED2;
                    led_num++;
                }
                // зажигаем LED3
                else if(led_num == 2)
                {
                    P8OUT |= LED3;
                    led_num++;
                }
                // сбрасываем счетчик и гасим все лампы
                else if(led_num == 3)
                {
                    P1OUT &= ~LED1;
                    P8OUT &= ~(LED2 | LED3);
                    led_num = 0;
                }
            }
            // ждем, пока кнопка не отпущена
            while(!(P1IN & BUTTON));
            // обнуляем счетчики
            i = button_check = 0;
        }
    }
}
```

5. Выводы

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 №13 без использования прерывани.