

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Микропроцессорные средства и системы

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

на тему

Среда разработки Code Composer Studio. Плата MSP-EXP430F5529.  
Цифровой ввод-вывод

Выполнил  
студент гр. 250541

В.Ю. Бобрик

Проверил  
доцент, к.т.н. каф. ЭВМ

И.Л. Селезнев

Минск 2025

## СОДЕРЖАНИЕ

1 Цель работы.....	3
2 Исходные данные к работе.....	3
3 Теоретические сведения.....	3
4 Выполнение работы.....	5
5 Вывод.....	7

## **1 Цель работы**

Написать программу по управлению цифровым вводом-выводом для микроконтроллера макетной платы MSP-EXP430F5529.

## **2 Исходные данные к работе**

В лабораторной работе применяется макетная плата MSP-EXP430F5529. Программа должна работать следующим образом:

- после запуска должны гореть 2 светодиода;
- после первого нажатия на кнопку – первый светодиод не горит, а второй мигает;
- после второго нажатия на кнопку – первый мигает, а второй не горит;
- после третьего нажатия на кнопку – горят оба светодиода;
- режимы работы должны меняться по отпусканию кнопки;
- не допускатьдребезга кнопки.

## **3 Теоретические сведения**

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

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; 7 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, обозначаемые как P1.7 и P2.2. Также программно доступны 8 светодиодов, три из которых (LED1 – LED3) 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. Схема управления выводами кнопок и светодиодов приведена на рисунке 1.

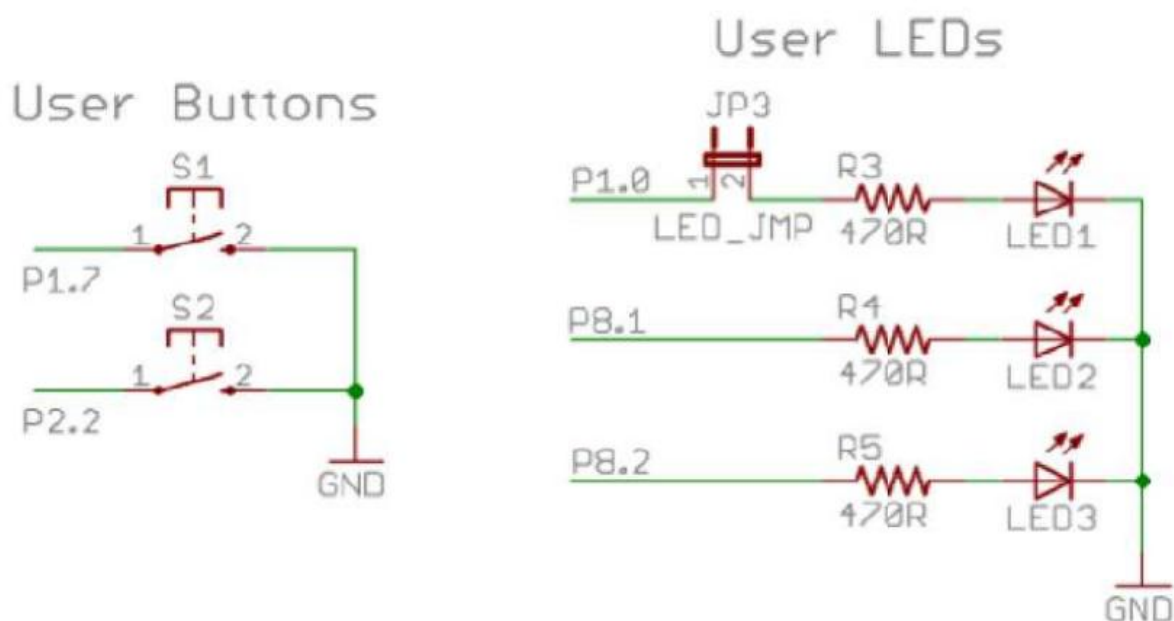


Рисунок 1 – Управление выводами кнопок и светодиодов

Следует обратить внимание, что регистр R<sub>x</sub>OUT управляет подключением подтягивающего резистора, если вывод сконфигурирован как цифровой I/O, направление — выход, и разрешен подтягивающий резистор.

При написании кода следует учесть несколько моментов. Вначале следует подключить заголовочный файл `mcp430.h`, который в свою очередь подключает файл `mcp430f5529.h`, содержащий необходимые константы в соответствии с архитектурой контроллера. Далее, поскольку после сброса запускается сторожевой таймер, его следует отключить.

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 случаях 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

Исходный код программы, выполняющей поставленную задачу, приведен ниже.

```
000 // Бобрик В.Ю. 250541 L1
001 // Цифровой ввод-вывод
002 #include <msp430.h>
003 // --- настройки антидребезга ---
004 #define BTN_SAMPLES 10 // всего опросов
005 #define BTN_THRESHOLD 8 // 8 из 10 должны показать 0 на кнопке
006 // --- регистры кнопки ---
007 #define BTN_S1_PORT_IN P1IN
008 #define BTN_S1_PORT_DIR P1DIR
009 #define BTN_S1_PORT_OUT P1OUT
010 #define BTN_S1_PORT_REN P1REN
011 #define BTN_S1_PORT_SEL P1SEL
012 #define BTN_S1_BIT BIT7 // кнопка S1 = P1.7
013 // --- led ---
014 #define LED1_PORT_OUT P1OUT
015 #define LED1_PORT_DIR P1DIR
016 #define LED1_PORT_SEL P1SEL
017 #define LED1_PORT_SEL2 P1SEL2
018 #define LED1_BIT BIT0 // LED1 = P1.0
019 #define LED2_PORT_OUT P8OUT
020 #define LED2_PORT_DIR P8DIR
021 #define LED2_PORT_SEL P8SEL
022 #define LED2_PORT_SEL2 P8SEL2
023 #define LED2_BIT BIT1 // LED2 = P8.1
024 #define LED3_PORT_OUT P8OUT
025 #define LED3_PORT_DIR P8DIR
026 #define LED3_PORT_SEL P8SEL
027 #define LED3_PORT_SEL2 P8SEL2
028 #define LED3_BIT BIT2 // LED3 = P8.2
029 // ----
030 #define BLINK_TICKS 20 // длительность мигания
031 unsigned char mode = 0; // рабочий режим
032 void init_hw(void)
033 {
034     WDTCTL = WDTPW + WDTHOLD; // остановить WDT
035     // led1
036     LED1_PORT_DIR |= LED1_BIT; // режим выход
037     LED1_PORT_OUT |= LED1_BIT; // включить режим выхода
038     LED1_PORT_SEL &= ~LED1_BIT; // gpio
039     // led3
040     LED3_PORT_DIR |= LED3_BIT; // режим выход
```

```

041     LED3_PORT_OUT |= LED3_BIT; // включить режим выхода
042     LED3_PORT_SEL &= ~LED3_BIT; // gpio
043     // кнопка 1
044     BTN_S1_PORT_DIR &= ~BTN_S1_BIT; // режим вход
045     BTN_S1_PORT_REN |= BTN_S1_BIT; // резистор
046     BTN_S1_PORT_OUT |= BTN_S1_BIT; // подтяжка резистора вверх
047     BTN_S1_PORT_SEL &= ~BTN_S1_BIT; // gpio
048 }
049 unsigned char debounce_S1(void)
050 {
051     unsigned int i, j, pressed_count = 0;
052     for (i = 0; i < BTN_SAMPLES; i++)
053     {
054         if ((BTN_S1_PORT_IN & BTN_S1_BIT) == 0)
055         {
056             pressed_count++;
057         }
058         for (j = 0; j < 10; j++)
059         {
060             ;
061         }
062     }
063     return pressed_count >= BTN_THRESHOLD;
064 }
065 int main(void)
066 {
067     init_hw();
068     unsigned char btn_prev = 0, blink_phase = 0;
069     unsigned int tick = 0;
070     while (1)
071     {
072         // антидребезг
073         unsigned char btn_now = debounce_S1();
074         if (btn_prev == 1 && btn_now == 0)
075         {
076             mode = (mode + 1) & 3; // 0-1-2-0
077         }
078         btn_prev = btn_now;
079         // мигания
080         tick++;
081         if (tick >= BLINK_TICKS)
082         {
083             tick = 0;
084             blink_phase ^= 1; // xor led
085         }
086         // режимы
087         switch (mode)
088         {
089             case 0: // led1 + led3
090                 LED1_PORT_OUT |= LED1_BIT;
091                 LED3_PORT_OUT |= LED3_BIT;
092                 break;
093             case 1: // led3 мигает
094                 LED1_PORT_OUT &= ~LED1_BIT;
095                 if (blink_phase)

```

```

096         LED3_PORT_OUT |= LED3_BIT;
097     else
098         LED3_PORT_OUT &= ~LED3_BIT;
099     break;
100 case 2: // led1 мигает
101     LED3_PORT_OUT &= ~LED3_BIT;
102     if (blink_phase)
103         LED1_PORT_OUT |= LED1_BIT;
104     else
105         LED1_PORT_OUT &= ~LED1_BIT;
106     break;
107 case 3: // led 1 + led 3
108     LED1_PORT_OUT |= LED1_BIT;
109     LED3_PORT_OUT |= LED3_BIT;
110     mode = 0; // идем в режим 0
111     break;
112 }
113 // задержка
114 volatile unsigned int d;
115 for (d = 0; d < 1000; d++)
116 {
117     ;
118 }
119 }
120 }

```

## 5 Вывод

В ходе выполнения лабораторной работы написана программа по управлению цифровым вводом-выводом для микроконтроллера макетной платы MSP-EXP430F5529.