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

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 допускать дребезга 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.

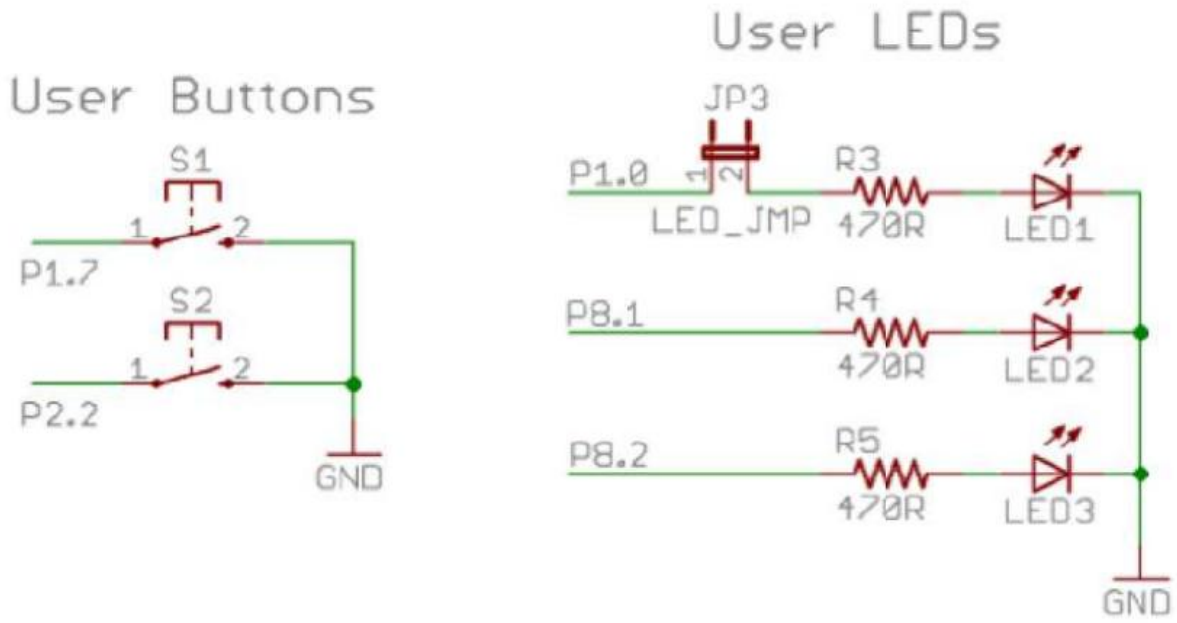

Рисунок 1 – Управление выводами кнопок и светодиодов

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

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.

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

Исходный код программы, выполняющей поставленную задачу, приведен ниже.

```
000 // Бобрик В.Ю. 250541 L1
001 // Цифровой ввод-вывод
002 #include <msp430.h>
003 // --- debounce settings ---
004 #define BTN_SAMPLES 10
005 #define BTN_THRESHOLD 8
006 // --- btn ---
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_PORT_SEL2 P1SEL2
013 #define BTN_S1_BIT BIT7 // S1 = P1.7
014 #define BTN_S2_PORT P2IN
015 #define BTN_S2_BIT BIT2 // S2 = P2.2
016 // --- led ---
017 #define LED1_PORT_OUT P1OUT
018 #define LED1_PORT_DIR P1DIR
019 #define LED1_PORT_SEL P1SEL
020 #define LED1_PORT_SEL2 P1SEL2
021 #define LED1_BIT BIT0 // LED1 = P1.0
022 #define LED2_PORT_OUT P8OUT
023 #define LED2_PORT_DIR P8DIR
024 #define LED2_PORT_SEL P8SEL
025 #define LED2_PORT_SEL2 P8SEL2
026 #define LED2_BIT BIT1 // LED2 = P8.1
027 #define LED3_PORT_OUT P8OUT
028 #define LED3_PORT_DIR P8DIR
029 #define LED3_PORT_SEL P8SEL
030 #define LED3_PORT_SEL2 P8SEL2
031 #define LED3_BIT BIT2 // LED3 = P8.2
032 // ----
033 #define BLINK_TICKS 20  // led blinkings
034 unsigned char mode = 0; // working mode
035 void init_hw(void)
036 {
037     WDTCTL = WDTPW + WDTHOLD; // Stop WDT
038     // led1
039     LED1_PORT_DIR |= LED1_BIT;  // out
040     LED1_PORT_OUT |= LED1_BIT; // enable
```

```
041     LED1_PORT_SEL &= ~LED1_BIT; // gpio
042 #ifdef P1SEL2
043     LED1_PORT_SEL2 &= ~LED1_BIT;
044 #endif
045     // led3
046     LED3_PORT_DIR |= LED3_BIT;  // out
047     LED3_PORT_OUT |= LED3_BIT; // enable
048     LED3_PORT_SEL &= ~LED3_BIT; // gpio
049 #ifdef P8SEL2
050     LED3_PORT_SEL2 &= ~LED3_BIT;
051 #endif
052     // btn1
053     BTN_S1_PORT_DIR &= ~BTN_S1_BIT; // in
054     BTN_S1_PORT_REN |= BTN_S1_BIT;  // ren
055     BTN_S1_PORT_OUT |= BTN_S1_BIT;  // ren up
056     BTN_S1_PORT_SEL &= ~BTN_S1_BIT; // GPIO
057 #ifdef P1SEL2
058     BTN_S1_PORT_SEL2 &= ~BTN_S1_BIT;
059 #endif
060 }
061 unsigned char debounce_S1(void)
062 {
063     unsigned int i, j, pressed_count = 0;
064     for (i = 0; i < BTN_SAMPLES; i++)
065     {
066         if ((BTN_S1_PORT_IN & BTN_S1_BIT) == 0)
067         {
068             pressed_count++;
069         }
070         for (j = 0; j < 10; j++)
071         {
072             ;
073         }
074     }
075     return pressed_count >= BTN_THRESHOLD;
076 }
077 int main(void)
078 {
079     init_hw();
080     unsigned char btn_prev = 0, blink_phase = 0;
081     unsigned int tick = 0;
082     while (1)
083     {
084         // debounce
085         unsigned char btn_now = debounce_S1();
086         if (btn_prev == 1 && btn_now == 0)
087         {
088             mode = (mode + 1) & 3; // 0-1-2-0
089         }
090         btn_prev = btn_now;
091         // blinks
092         tick++;
093         if (tick >= BLINK_TICKS)
094         {
095             tick = 0;
```

```
096             blink_phase ^= 1; // xor led
097         }
098         // mods
099         switch (mode)
100         {
101         case 0 // led1 + led3
102             LED1_PORT_OUT |= LED1_BIT;
103             LED3_PORT_OUT |= LED3_BIT;
104             break;
105         case 1 // led3 blink
106             LED1_PORT_OUT &= ~LED1_BIT;
107             if (blink_phase)
108                 LED3_PORT_OUT |= LED3_BIT;
109             else
110                 LED3_PORT_OUT &= ~LED3_BIT;
111             break;
112         case 2 // led1 blink
113             LED3_PORT_OUT &= ~LED3_BIT;
114             if (blink_phase)
115                 LED1_PORT_OUT |= LED1_BIT;
116             else
117                 LED1_PORT_OUT &= ~LED1_BIT;
118             break;
119         case 3 // led 1 + led 3
120             LED1_PORT_OUT |= LED1_BIT;
121             LED3_PORT_OUT |= LED3_BIT;
122             mode = 0; // goto mode 0
123             break;
124         }
125         // delay
126         volatile unsigned int d;
127         for (d = 0; d < 1000; d++)
128         {
129             ;
130         }
131     }
132 }
```

5 Вывод

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