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.

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.

4 Выполнение работы

Исходный код программы, выполняющей поставленную задачу, приведен ниже.

```
001 #include <msp430.h>
002 // --- debounce settings ---
003 #define BTN_SAMPLES 10
004 #define BTN_THRESHOLD 8
005
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
015 #define BTN_S2_PORT     P2IN
016 #define BTN_S2_BIT      BIT2    //S2 = P2.2
017
018 // --- led ---
019 #define LED1_PORT_OUT   P1OUT
020 #define LED1_PORT_DIR   P1DIR
021 #define LED1_PORT_SEL   P1SEL
022 #define LED1_PORT_SEL2  P1SEL2
023 #define LED1_BIT        BIT0    // LED1 = P1.0
024
025 #define LED2_PORT_OUT   P8OUT
026 #define LED2_PORT_DIR   P8DIR
027 #define LED2_PORT_SEL   P8SEL
028 #define LED2_PORT_SEL2  P8SEL2
029 #define LED2_BIT        BIT1    // LED1 = P8.1
030
031 #define LED3_PORT_OUT   P8OUT
032 #define LED3_PORT_DIR   P8DIR
033 #define LED3_PORT_SEL   P8SEL
034 #define LED3_PORT_SEL2  P8SEL2
035 #define LED3_BIT        BIT2    // LED1 = P8.2
036
037 // ----
038 #define BLINK_TICKS 20      // led blinkings
039 unsigned char mode = 0;         // working mode
040
041
```

```
042 void init_hw(void) {
043     WDTCTL = WDTPW+WDTHOLD;     // Stop WDT
044 
045     //led1
046     LED1_PORT_DIR |= LED1_BIT; //out
047     LED1_PORT_SEL &= ~LED1_BIT;//gpio
048 #ifdef P1SEL2
049     LED1_PORT_SEL2 &= ~LED1_BIT;
050 #endif
051     LED1_PORT_OUT |= LED1_BIT;//enable
052 
053     //led3
054     LED3_PORT_DIR |= LED3_BIT;//out
055     LED3_PORT_SEL &= ~LED3_BIT;//gpio
056 #ifdef P8SEL2
057     LED3_PORT_SEL2 &= ~LED3_BIT;
058 #endif
059     LED3_PORT_OUT |= LED3_BIT;//enable
060 
061     // btn1
062     BTN_S1_PORT_DIR &= ~BTN_S1_BIT;//in
063     BTN_S1_PORT_REN |= BTN_S1_BIT; //ren
064     BTN_S1_PORT_OUT |= BTN_S1_BIT; //ren up
065     BTN_S1_PORT_SEL &= ~BTN_S1_BIT; //GPIO
066 #ifdef P1SEL2
067     BTN_S1_PORT_SEL2 &= ~BTN_S1_BIT;
068 #endif
069 }
070 
071 
072 unsigned char debounce_S1(void) {
073     unsigned int i, j, pressed_count = 0;
074     for (i=0;i<BTN_SAMPLES;i++){
075         if((BTN_S1_PORT_IN & BTN_S1_BIT)==0) {
076             pressed_count++;
077         }
078         for (j=0;j<10;j++) {;}
079     }
080     return pressed_count >=BTN_THRESHOLD;
081 }
082 
083 int main(void)
084 {
085   init_hw();
086 
087   unsigned char btn_prev = 0, blink_phase=0;
088   unsigned int tick = 0;
089   while(1) {
090 
091       // debounce
092       unsigned char btn_now = debounce_S1();
093       if (btn_prev == 1 && btn_now ==0) {
094           mode = (mode+1) & 3; //0-1-2-0
095       }
096       btn_prev=btn_now;
```

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

5 Вывод

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