

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

Подсистема прерываний микроконтроллера MSP430F5529

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 допускать дребезга кнопки.

Обработка кнопки должна быть основана на прерываниях.

3 Теоретические сведения

Различают системные немаскируемые (SMNI), пользовательские немаскируемые (UNMI) и маскируемые прерывания. К системным немаскируемым относятся: сигнал RST/NMI в режиме NMI, сбой генератора, ошибка доступа Flash памяти. К пользовательским немаскируемым - сбой напряжения питания (от подсистемы PMM), доступ к несуществующей (vacant) памяти, события с буфером (mailslot) JTAG интерфейса. Маскируемые прерывания могут быть отключены (замаскированы) индивидуально или все сразу (бит GIE регистра состояния SR).

Задержка от возникновения запроса на прерывание до начала выполнения обработчика составляет 6 циклов. При этом заканчивается выполнение текущей инструкции, счетчик команд PC сохраняется в стеке (указывает на следующую команду), регистр состояния SR сохраняется в стеке, выбирается прерывание с максимальным приоритетом (если поступило несколько запросов), автоматически сбрасывается флаг запроса от отдельного прерывания (сброс общего флага запроса должен осуществляться программно). Далее, все биты SR сбрасываются, за исключением SCG0, так как останавливаются все режимы с низким питанием. Так как бит GIO при этом устанавливается в 0, все прерывания запрещаются. Наконец, вектор (адрес обработчика) загружается в PC.

Из-за конвейерной архитектуры процессора, команда, следующая за EINT (разрешение прерывания), всегда выполняется, даже если запрос на прерывание возник до его разрешения. Если за EINT сразу следует DINT, прерывание, ожидающее обработки может быть не обслужено. Команды, следующие за DINT в этом случае могут сработать некорректно. Аналогичные

последствия вызываются альтернативными командами, которые устанавливают и сразу сбрасывают флаг GIE регистра состояний. Рекомендуется вставлять хотя бы одну команду между EINT и DINT. Возврат из прерывания выполняется командой RETI, которая выполняется за 5 циклов и загружает из стека SR, PC. Таблица векторов прерываний располагается по адресам 0FFFFh – 0FF80h и содержит 64 вектора. Бит SYSRIVECT регистра SYSCTL позволяет определить альтернативную таблицу векторов, в старших адресах RAM. По сигналу сброса этот бит автоматически сбрасывается.

За прерывания отвечают ряд системных регистров. Пользовательские маскируемые прерывания рассматриваются отдельно при обсуждении соответствующего функционального узла архитектуры микроконтроллера, в частности, ранее уже рассматривались регистры для работы с прерываниями от цифровых портов ввода-вывода.

Работа с прерываниями достаточно проста. Вначале необходимо разрешить соответствующее прерывание, например, $P1IE \neq 0$; - разрешает прерывание по входу 7 вывода порта 1, в экспериментальной плате к нему подключена кнопка S1.

MSP430F5529 содержит 32-разрядный сторожевой таймер WDT (базовый адрес 015Ch), 3 таймера TAx (базовые адреса соответственно 0340h, 0380h, 0400h), таймер TBx (базовый адрес 03C0h) и таймер часов реального времени RTC_A (базовый адрес 04A0h).

Таймер А – это 16-разрядный таймер/счетчик с широкими возможностями по использованию прерываний, которые могут генерироваться счетчиком в случае переполнения и от каждого регистра захвата/сравнения. Таймер А обладает следующими возможностями: – асинхронный 16-битный таймер/счетчик с четырьмя рабочими режимами; – выбираемый и конфигурируемый источник счетного импульса; – три конфигурируемых регистра захвата/сравнения (в таймере ТА0 их 5); – возможность множественного захвата/сравнения; – конфигурируемые выходы с возможностью широтно-импульсной модуляции; – асинхронная фиксация (защелка) входа и выхода; – счет по фронту тактового импульса; – возможность генерации прерываний при переполнении; – регистр вектора прерываний для быстрого декодирования всех прерываний таймера А. Источниками входного импульса для таймера А могут быть следующие тактовые сигналы: ACLK, SMCLK, внешние CAxCLK, INCLK. На входе имеется программно доступный делитель частоты, который позволяет снижать частоту в 2,3,4,5,6,7,8 раз. Режимы работы таймера: остановка, прямой счет (до уровня TAxCCR0) (Up Mode), непрерывный режим (Continuous Mode), реверсивный счет (Up/Down mode). Таймер В имеет ряд отличий от таймера А: – 7 регистров захвата/сравнения; – разрядность счетчика программируется (8, 10, 12, 16 бит); – регистр TBxCCRn с двойной буферизацией и может быть сгруппирован; – все выходы имеют высокоимпедансное состояние; – не поддерживается бит SCCI. Таймер часов

реального времени RTC_A представляет собой конфигурируемые часы реального времени с функцией календаря и счетчика общего назначения. Поддерживает выбор формата BCD или двоичный в режиме часов реального времени, имеет программируемый будильник, подстройку коррекции времени, возможность прерываний.

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_IE  P1IE
012 #define BTN_S1_PORT_IES P1IES
013 #define BTN_S1_PORT_IFG P1IFG
014 #define BTN_S1_PORT_SEL P1SEL
015 #define BTN_S1_PORT_SEL2 P1SEL2
016 #define BTN_S1_BIT      BIT7      //S1 = P1.7
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 volatile unsigned char mode = 0;      // working mode
040 volatile unsigned char blink_phase = 0;
041 volatile unsigned int  blink_cnt = 0;
```

```

042 volatile unsigned char btn_prev = 0;
043
044 void init_hw(void) {
045     WDTCTL = WDTPW+WDTHOLD;      // Stop WDT
046
047     //led1
048     LED1_PORT_DIR |= LED1_BIT; //out
049     LED1_PORT_OUT |= LED1_BIT; //enable
050
051     //led3
052     LED3_PORT_DIR |= LED3_BIT; //out
053     LED3_PORT_OUT |= LED3_BIT; //enable
054
055     // btn1
056     BTN_S1_PORT_DIR &= ~BTN_S1_BIT; //in
057     BTN_S1_PORT_REN |= BTN_S1_BIT; //ren
058     BTN_S1_PORT_OUT |= BTN_S1_BIT; //ren up
059     BTN_S1_PORT_SEL &= ~BTN_S1_BIT;
060
061     //timerA0
062     TA0CCR0 = 1250-1;
063     TA0CTL = TASSEL_2 | ID_3 | MC_1 | TACLRL;
064     TA0CCTL0 = CCIE;
065 }
066
067
068 unsigned char debounce_S1(void) {
069     unsigned int i, j, pressed_count = 0;
070     for (i=0; i<BTN_SAMPLES; i++){
071         if((BTN_S1_PORT_IN & BTN_S1_BIT)==0) {
072             pressed_count++;
073         }
074         for (j=0; j<10; j++) {;}
075     }
076     return pressed_count >=BTN_THRESHOLD;
077 }
078
079 int main(void)
080 {
081     init_hw();
082     __enable_interrupt();
083     __low_power_mode_3();
084     return 0;
085 }
086
087 #pragma vector=TIMER0_A0_VECTOR
088 __interrupt void Timer_A0(void) {
089     unsigned char btn_now = debounce_S1();
090     if (btn_prev == 1 && btn_now == 0) {
091         mode = (mode+1) & 3;
092     }
093     btn_prev = btn_now;
094
095     blink_cnt++;
096     if (blink_cnt >=25) {

```

```

097         blink_cnt = 0;
098         blink_phase ^= 1;
099     }
100
101     switch(mode) {
102     case 0:
103         LED1_PORT_OUT |= LED1_BIT;
104         LED3_PORT_OUT |= LED3_BIT;
105         break;
106     case 1:
107         LED1_PORT_OUT &= ~LED1_BIT;
108         if(blink_phase) LED3_PORT_OUT |= LED3_BIT;
109         else LED3_PORT_OUT &= ~LED3_BIT;
110         break;
111     case 2:
112         LED3_PORT_OUT &= ~LED3_BIT;
113         if(blink_phase) LED1_PORT_OUT |= LED1_BIT;
114         else LED1_PORT_OUT &= ~LED1_BIT;
115         break;
116     case 3:
117         LED1_PORT_OUT |= LED1_BIT;
118         LED3_PORT_OUT |= LED3_BIT;
119         mode = 0;
120         break;
121     }
122 }

```

5 Вывод

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