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

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 |= BIT7; - 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 A – 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 A 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 TA0 их 5); возможность множественного захвата/сравнения; конфигурируемые выводы 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 A. Источниками входного импульса для таймера A 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 B имеет ряд отличий от таймера A: 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.

```
000 // Бобрик В.Ю. 250541 L2
001 // Подсистема прерываний
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_PORT_IE P1IE
013 #define BTN_S1_PORT_IES P1IES
014 #define BTN_S1_PORT_IFG P1IFG
015 #define BTN_S1_BIT BIT7 // кнопка S1 = P1.7
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 LED3_PORT_OUT P8OUT
023 #define LED3_PORT_DIR P8DIR
024 #define LED3_PORT_SEL P8SEL
025 #define LED3_PORT_SEL2 P8SEL2
026 #define LED3_BIT BIT2 // LED3 = P8.2
027 // ----
028 #define BLINK_COUNT 15
029 volatile unsigned char mode = 0; // режим работы
030 volatile unsigned char blink_phase = 0;
031 volatile unsigned int blink_cnt = 0;
032 void init_hw(void) // стартовая настройка
033 {
034     WDTCTL = WDTPW + WDTHOLD; // остановить WDT
035     // led1
036     LED1_PORT_DIR |= LED1_BIT; // режим выход
037     LED1_PORT_OUT |= LED1_BIT; // включить диод
038     LED1_PORT_SEL &= ~LED1_BIT; // gpio
039     // led3
040     LED3_PORT_DIR |= LED3_BIT; // режим выход
041     LED3_PORT_OUT |= LED3_BIT; // включить диод
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     // прерывание по кнопке
049     BTN_S1_PORT_IE |= BTN_S1_BIT;   // разрешить прерывание
050     BTN_S1_PORT_IES |= BTN_S1_BIT;  // сначала ловим задний фронт
051     BTN_S1_PORT_IFG &= ~BTN_S1_BIT; // сброс флага
052     // таймер A0
053     TA0CCR0 = 1250 - 1; // от 0 до 1249 (ровно 10 мс)
054     //      SMCLK | делитель 8 | счет вверх | очистить
055     TA0CTL = TASSEL_2 | ID_3 | MC_1 | TACLR;
056     TA0CCTL0 = CCIE; // разрешить прерывание по равенству
057 }
058 unsigned char debounce_S1(void)
059 {
060     unsigned int i, j, pressed_count = 0;
061     for (i = 0; i < BTN_SAMPLES; i++)
062     {
063         if ((BTN_S1_PORT_IN & BTN_S1_BIT) == 0)
064         {
065             pressed_count++;
066         }
067         for (j = 0; j < 10; j++)
068         {
069             ;
070         }
071     }
072     return pressed_count >= BTN_THRESHOLD;
073 }
074 int main(void)
075 {
076     init_hw();              // стартовая настройка
077     __enable_interrupt();   // разрешить прерывания
078     __low_power_mode_3();   // режим низкого потребления
079     return 0;
080 }
081 // регистрация обработчика прерывания по таймеру A0
082 #pragma vector = TIMER0_A0_VECTOR
083 __interrupt void Timer_A0(void)
084 {
085     // мигание
086     blink_cnt++;
087     if (blink_cnt >= BLINK_COUNT)
088     {
089         blink_cnt = 0;
090         blink_phase ^= 1;
091     }
092     // режим работы
093     switch (mode)
094     {
095     case 0: // led1 + led3
096         LED1_PORT_OUT |= LED1_BIT;
097         LED3_PORT_OUT |= LED3_BIT;
098         break;
```

```
099     case 1: // led3 мигает
100         LED1_PORT_OUT &= ~LED1_BIT;
101         if (blink_phase)
102             LED3_PORT_OUT |= LED3_BIT;
103         else
104             LED3_PORT_OUT &= ~LED3_BIT;
105         break;
106     case 2: // led1 мигает
107         LED3_PORT_OUT &= ~LED3_BIT;
108         if (blink_phase)
109             LED1_PORT_OUT |= LED1_BIT;
110         else
111             LED1_PORT_OUT &= ~LED1_BIT;
112         break;
113     case 3: // led 1 + led 3
114         LED1_PORT_OUT |= LED1_BIT;
115         LED3_PORT_OUT |= LED3_BIT;
116         mode = 0; // переход в режим 0
117         break;
118     }
119 }
120 #pragma vector=PORT1_VECTOR
121 __interrupt void Port_1(void)
122 {
123     if (BTN_S1_PORT_IFG & BTN_S1_BIT) // если есть прерывание S1
124     {
125         if (BTN_S1_PORT_IES & BTN_S1_BIT) // если это задний фронт
126         {
127             if (debounce_S1())
128             {
129                 // переключаем на передний фронт (отпускание)
130                 BTN_S1_PORT_IES &= ~BTN_S1_BIT;
131             }
132         }
133         else // если это передний фронт
134         {
135             if (!debounce_S1())
136             {
137                 mode = (mode + 1) & 3; // смена режима
138                 // переключаем на задний фронт (нажатие)
139                 BTN_S1_PORT_IES |= BTN_S1_BIT;
140             }
141         }
142         BTN_S1_PORT_IFG &= ~BTN_S1_BIT; // сброс флага
143     }
144 }
```

5 Вывод

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