Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ЭВМ

# Лабораторная работа № 2
# «Прерывания. Таймеры»
# Вариант №2

Выполнил:
ст. гр. 050503 Кириллов В.И.

Проверил:
ассистент Шеменков В.В.

Минск 2023

## 1. Цель работы

Цели работы:

Ознакомиться с работой подсистемы прерываний и таймерами микроконтроллера MSP430F5529.

## 2. Исходные данные

| | | | |
|---|---|---|---|
| 13 | LED1 – LED3, S1, TA1 | Непре-рывный | Изменение состояния диода, номер диода определяется количеством нажатий кнопки за некоторое время |

Для выполнения работы используется плата MSP-EXP430F5529 и интегрированная среда разработки Code Composer Studio.

## 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).

Таблица 2.1. Регистры для работы с прерываниями

| **Регистр** | **Адрес** | **Назначение** |
|---|---|---|
| **SFRIE1** | 0100h | Разрешение прерываний |
| **SFRIFG1** | 0102h | Флаги прерываний |
| **SYSCTL** | 0180h | Регистр управления |
| **SYSBERRIV** | 0198h | Генератор вектора ошибок шины |
| **SYSUNIV** | 019Ah | Генератор вектора пользовательских NMI |
| **SYSSNIV** | 019Ch | Генератор вектора системных NMI |
| **SYSRSTIV** | 019Eh | Генератор вектора сброса |

Таблица 2.2. Поля регистров для работы с прерываниями

| Регистр | Биты | Поле | Назначение |
|---|---|---|---|
| SFRIE1 | 7 | JMBOUTIE | Разрешение прерываний выхода JTAG |
| | 6 | JMBINIE | Разрешение прерываний входа JTAG |
| | 5 | ACCVIE | Разрешение прерываний нарушения доступа Flash |
| | 4 | NMIIE | Разрешение прерываний вывода NMI |
| | 3 | VMAIE | Разрешение прерываний доступа к несуществующей памяти |
| | 1 | OFIE | Разрешение прерываний сбоя генератора |
| | 0 | WDTIE | Разрешение прерываний сторожевого таймера |
| SFRIFG1 | 7 | JMBOUTIFG | Флаг прерывания выхода JTAG |
| | 6 | JMBINIFG | Флаг прерывания входа JTAG |
| | 4 | NMIIFG | Флаг прерывания NMI |
| | 3 | VMAIFG | Флаг прерывания доступа к несуществующей памяти |
| | 1 | OFIFG | Флаг прерывания сбоя генератора |
| | 0 | WDTIFG | Флаг прерывания сторожевого таймера |
| SYSCTL | 0 | SYSRIVECT | Вектор прерывания при выходе за пределы RAM (64K или полностью) |
| SYSUNIV | 0-15 | SYSUNIV | Вектор пользовательского NMI |
| SYSSNIV | 0-15 | SYSSNIV | Вектор системного NMI |
| SYSRSTIV | 0-15 | SYSRSTIV | Вектор прерываний сброса |
| SYSBERRIV | 0-15 | SYSBSLOFF | Вектор прерываний ошибки системной шины |

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).

## 4. Выполнение работы

### 4.1. Задание

В соответствии с вариантом, используя прерывания и таймеры, запрограммировать кнопки и светодиоды. Для работы с кнопками

использовать только прерывания. Не использовать опросы флагов состояния в цикле и циклы задержки (активное ожидание). Не допускается использовать иные заголовочные файлы, кроме msp430, не допускается также использовать высокоуровневые библиотеки. При выполнении задания особое внимание уделить грамотному выбору режима работы таймера. Комментарии в тексте программы обязательны, они должны пояснять что именно делает данные фрагмент.

### 4.2. Программа по управлению цифровым вводом-выводом с использованием прерываний

```
#include <msp430.h>

#define LED1 BIT0
#define LED2 BIT1
#define LED3 BIT2
#define BUTTON BIT7

static volatile unsigned int button_intr = 0;
static volatile unsigned int timer_counter = 0;

void main(void) {
   // отключаем сторожевой таймер
   WDTCTL = WDTPW | WDTHOLD;

   // устанавливаем пины LED1, LED2 и LED3 как выходы
   P1DIR |= LED1;
   P8DIR |= (LED2 | LED3);
   P8OUT &= ~(LED2 | LED3);

   // выключаем все светодиоды
   P1OUT &= ~LED1;
   P8OUT &= ~(LED2 | LED3);

   // устанавливаем пин BUTTON как вход
   P1DIR &= ~BUTTON;
   // включаем подтягивающий резистор на пине BUTTON
   P1REN |= BUTTON;
   // устанавливаем подтягивающий резистор на пине BUTTON
   P1OUT |= BUTTON;

   P1IE |= BUTTON;         // Разрешение прерывания для пина BUTTON
   P1IES |= BUTTON;        // Настройка прерывания по спаду
   P1IFG = 0;
```

```
    TA1CCTL0 = CCIE;    // Разрешение прерывания для таймера TA1
    TA1CCR0 = 1047;
    TA1CTL = TASSEL_2 + MC_1 + TACLR;

    __enable_interrupt();   // Разрешение глобальных прерываний
    __no_operation();
}

#pragma vector=PORT1_VECTOR
__interrupt void Port1_ISR(void) {
        button_intr = 1;
        P1IFG &= ~BIT7;
}

#pragma vector=TIMER1_A0_VECTOR
__interrupt void Timer1_A0_ISR(void) {
        static int is_button_pressed = 0;
        static int debounce_total_counter = 0;
        static int debounce_pressed_counter = 0;
        static int pressed_count = 0;

        static int is_counter_allowed = 0;
        static int last_pressed_time_counter = 0;

        timer_counter++;
        if(button_intr){
                P1IE &= ~BIT7;
                debounce_total_counter++;
                debounce_pressed_counter += (P1IN & BIT7) == 0;
                if(debounce_total_counter == 10){
                        is_button_pressed = debounce_pressed_counter > 7;
                        if (is_button_pressed) {
                                is_counter_allowed = 1;
                                last_pressed_time_counter = timer_counter;
                                pressed_count++;
                        }
                        button_intr = 0;
                        debounce_total_counter = 0;
                        debounce_pressed_counter = 0;

                        if(is_button_pressed)
                                P1IES &= ~BIT7;
                        else
```

```
                    P1IES |= BIT7;
                P1IE |= BIT7;
            }
        }
        if (!is_counter_allowed) {
            timer_counter = 0;
        }
        if (timer_counter - last_pressed_time_counter > 2000) {
            if (pressed_count == 1) {
                P1OUT ^= LED1;
            } else if (pressed_count == 2) {
                P8OUT ^= LED2;
            } else if (pressed_count == 3) {
                P8OUT ^= LED3;
            }

            pressed_count = 0;
            timer_counter = 0;
            is_counter_allowed = 0;
            last_pressed_time_counter = 0;
        }
}
```

## 5. Выводы

В ходе выполнения лабораторной работы удалось ознакомиться с интегрированной средой разработки Code Composer Studio и с основными функциональными возможностями платы MSP-EXP430F5529. Удалось написать программу по управлению цифровым вводом-выводом (светодиодами и кнопками) в соответствии с вариантом №2 с использованием прерываний.