Белорусский Государственный Университет
Информатики и Радиоэлектроники
Кафедра ЭВМ

Отчет по лабораторной работе № 2

Тема: «Прерывания. Таймеры»

Выполнили:

ст. гр. 950503
Зарубо Д. Ю
Ященко В.П

Проверил:
Шеменков В.В

Минск 2022

1 ЦЕЛЬ РАБОТЫ

Ознакомиться с работой подсистемы прерываний и таймерами микроконтроллера MSP430F5529. Написать программу, используя таймеры и прерывания в соответствии с заданием варианта

2 ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

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.

Вариант 12

12	LED4 – LED8, S1, TA2	Движе-ние	Режим включен, только когда кнопка нажата. Диоды включаются друг за другом с некоторой задержкой, потом гаснут друг за другом с некоторой задержкой.

3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Таблица 2.5. Регистр управления TAxCTL

Бит	Поле	Тип	Сброс	Описание	Определения флагов в msp430f5529.h
9-8	TASSEL	RW	0h	Выбор источника тактирования: 00b TACLK 01b ACLK 10 SMCLK 11 INCLK	 TASSEL__TACLK TASSEL__ACLK TASSEL__SMCLK TASSEL__INCLK
7-6	ID	RW	0h	Входной делитель. Эти биты позволяют выбрать коэффициент деления для входной тактовой частоты. 00b /1 01b /2 10b /4 11b /8	 ID__1 ID__2 ID__4 ID__8
5-4	MC	RW	0h	Выбор режима. Установка MCx=00h, когда таймер не используется, позволяет уменьшить потребляемую мощность. 00b остановка: таймер остановлен 01b прямой счет: вверх к TAxCCR0 10b непрерывный: вверх к 0FFFFh 11b реверсивный: вверх к TAxCCR0, затем вниз к 0000h	 MC__STOP MC__UP MC__CONTINOUS MC__UPDOWN
2	TACLR	RW	0h	Очистка таймера A. Установка этого бита сбрасывает TAxR, IDx и MCx. Бит TACLR автоматически сбрасывается и всегда читается как нуль.	TACLR
1	TAIE	RW	0h	Разрешение прерывания от таймера A. Этот бит разрешает запрос прерывания TAIFG. 0b Запрещение прерывания 1b Разрешение прерывания	TAIE
0	TAIFG	RW	0h	Флаг прерывания Таймера A 0b Прерывание не ожидается 1b Ожидается прерывание	TAIFG

Таблица 2.6. Регистр управления захватом/сравнением TAxCCTLn

Бит	Поле	Тип	Сброс	Описание	Определения флагов в msp430f5529.h
15-14	CM	RW	0h	Выбор режима захвата 00 Нет захвата 01 Захват по нарастающему (переднему) фронту 10 Захват по заднему фронту (сбросу) 11 Захват как по переднему, так и по заднему фронтам	 CM_1 CM_2 CM_3 CM_4
13-12	CCIS	RW	0h	Выбор входа захвата/сравнения. Этими битами выбирается входной сигнал TAxCCR0. 00 CCIxA 01 CCIxB 10 GND 11 VCC	 CCIS_0 CCIS_1 CCIS_2 CCIS_3
11	SCS	RW	0h	Синхронизация источника захвата. Используется для синхронизации входного сигнала захвата с тактовым сигналом таймера 0 Асинхронный захват 1 Синхронный захват	SCS
10	SCCI	RW	0h	Синхронизация входа захвата/сравнения. Выбранный входной сигнал CCI фиксируется по сигналу EQUx и может быть прочитан через этот бит	CCIS0 CCIS1
8	CAP	RW	0h	Выбор режима захвата 0 Режим сравнения 1 Режим захвата	CAP
7-5	OUTMOD	RW	0h	Выбор режима выхода. Режимы 2, 3, 6 и 7 не пригодны для TAxCCR0, поскольку EQUx=EQU0. 000 Значение бита OUT 001 Установка 010 Переключение/сброс 011 Установка/сброс 100 Переключение 101 Сброс 110 Переключение/установка 111 Сброс/установка	 OUTMOD_0 OUTMOD_1 OUTMOD_2 OUTMOD_3 OUTMOD_4 OUTMOD_5 OUTMOD_6 OUTMOD_7

Бит	Поле	Тип	Сброс	Описание	Определения флагов в msp430f5529.h
4	CCIE	RW	0h	Разрешение прерывания по захвату/сравнению. Этот бит разрешает запрос прерывания от соответствующего флага CCIFG. 0 Запрещение прерывания 1 Разрешение прерывания	CCIE
3	CCI	R	0h	Вход захвата/сравнения. Выбранный входной сигнал может быть прочитан этим битом.	CCI
2	OUT	RW	0h	Выход. Этот бит указывает состояние выхода. Если выбран режима вывода 0, этот бит напрямую управляет состоянием выхода. 0 Низкий уровень выхода 1 Высокий уровень выхода	OUT
1	COV	RW	0h	Переполнение захвата. Этот бит указывает, что произошло переполнение захвата. Бит COV должен быть сброшен программно 0 Нет переполнения захвата 1 Произошло переполнение захвата	COV
0	CCIFG	RW	0h	Флаг прерывания захвата/сравнения 0 Прерывание не ожидается 1 Ожидается прерывание	CCIFG

4 ВЫПОЛНЕНИЕ РАБОТЫ

Код программмы:

```
#include <msp430.h>
volatile long int sysMillis = 0;
// debounce (times)
volatile int DEBOUCE_TRESHOLD = 2;
// Led blink poll period
volatile int ledBlinkPollPeriod = 500;
// button poll period
volatile int buttonPollPeriod = 5;
// Leds off poll period
volatile char ledOffPollPeriod = 100;
// Is button pressed or released poll period
volatile char pressReleasePollPeriod = 50;

int leds[5] = { BIT1, BIT2, BIT3, BIT4, BIT5 };
volatile char ledsPrevStates[5] = { };
volatile int ledIndex = 0;
volatile char ledsWereOn = 0;
```

```
volatile char buttonState, button_debounce_counter;
volatile char buttonPrevState;
volatile char isButtonPressed = 0;
volatile char isButtonReleased = 1;

void setupPins()
{
    // S1 input
    P1DIR &= ~BIT7;
    // pull enable
    P1REN |= BIT7;
    //pull-up
    P1OUT |= BIT7;

    //led 1 on p1.0
    //out
    P1DIR |= BIT1;
    //off
    P1OUT &= ~BIT1;

    //led 2 on 1.1
    //out
    P1DIR |= BIT5;
    //off
    P1OUT &= ~BIT5;

    //led 3 on 1.2
    //out
    P1DIR |= BIT2;
    //off
    P1OUT &= ~BIT2;
    //led 3 on 1.3
    //out
    P1DIR |= BIT3;
    //off
    P1OUT &= ~BIT3;
    //led 4 on 1.4
    //out
    P1DIR |= BIT4;
    //off
    P1OUT &= ~BIT4;
}

// т.к нет других входных clk
// base MCLK = 1MHz
// период в 1ms = 1000 тиков при предделителе 1,
// значит TAxCCR0 = 500 при предделителе 2 - биты ID1 = 01

// TASSEL__SMCLK 10
// MC_1 = Управление режимом таймера = 01 - прямой счет до TAxCCR0
// TACLR - начальное обнуление
//
void setupTimers()
{
    TA2CCR0 = 500 - 1;
    // Разрешаем прерывание таймера по достижению значения TA1CCR0
    TA2CCTL0 = CCIE;
```

```
    // Настройка режима работы таймера Timer_A
    TA2CTL = TASSEL__SMCLK | ID_1 | MC_1 | TACLR;
    _enable_interrupt();

}

// 1 ms interrupt
#pragma vector = TIMER2_A0_VECTOR
__interrupt void CCR0_ISR(void)
{
    static long int prevLedBlinkPollEntry = 0;
    static long int prevButtonPollEntry = 0;
    static long int prevLedOffPollEntry = 0;
    static long int prevPressReleasePollPeriod = 0;

    sysMillis++;

    if (sysMillis - prevButtonPollEntry >= buttonPollPeriod)
    {
        prevButtonPollEntry = sysMillis;
        if (!(P1IN & BIT7))
        {
            if (button_debounce_counter > DEBOUCE_TRESHOLD)
            {
                buttonState = 1;
                button_debounce_counter = 0;
            }
            else
            {
                button_debounce_counter++;
            }
        }
        else
        {
            button_debounce_counter = 0;
            buttonState = 0;
        }

    }

    if (sysMillis - prevLedBlinkPollEntry >= ledBlinkPollPeriod && isButtonPressed)
    {
        if (!ledsWereOn)
        {
            char i;
            for (i = 0; i < 5; i++)
            {
                if (ledsPrevStates[i])
                {
                    P1OUT |= leds[i];
                }

            }
            ledsWereOn = 1;
        }
        prevLedBlinkPollEntry = sysMillis;
        P1OUT ^= leds[ledIndex];
```

```
            ledIndex++;
            if (ledIndex > 4)
                ledIndex = 0;
        }



        if (sysMillis - prevPressReleasePollPeriod >= pressReleasePollPeriod)
        {
            prevPressReleasePollPeriod = sysMillis;
            if (buttonPrevState != buttonState)
            {
                if (buttonState)
                {
                    isButtonPressed = 1;
                    isButtonReleased = 0;
                }
                if (!buttonState)
                {
                    isButtonPressed = 0;
                    isButtonReleased = 1;
                }
            }
            buttonPrevState = buttonState;
        }

        if (sysMillis - prevLedOffPollEntry >= ledOffPollPeriod)
        {
            if (isButtonReleased)
            {
                if (ledsWereOn)
                {
                    char i;
                    for (i = 0; i < 5; i++)
                    {
                        ledsPrevStates[i] = P1IN & leds[i];
                    }
                    ledsWereOn = 0;
                }
                P1OUT &= ~BIT1;
                P1OUT &= ~BIT2;
                P1OUT &= ~BIT3;
                P1OUT &= ~BIT4;
                P1OUT &= ~BIT5;
            }
        }
        return;
    }

    int main(void)
    {
        WDTCTL = WDTPW | WDTHOLD;
        setupPins();
        setupTimers();
        while (1);
    }
```

ВЫВОД

В ходе выполнения лабораторной работы была написана программа с использованием таймеров и их прерываний. Для опроса кнопки и взаимодействия со светодиодами было использовано прерывание таймера A.