БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №2

Тема: «Подсистема прерываний микроконтроллера MSP430F5529»

Выполнил:

студент группы 350531 Козяков А.И.

Проверил:

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Селезнёв И.Л.

Минск

2018

**Цель:**

Ознакомиться с работой подсистемы прерываний микроконтроллера MSP430F5529.

**Постановка задачи:**

Разработать программу для микроконтроллера макетной платы MSP-EXP430F5529. Алгоритм работы программы:

1. По отпусканию кнопки S2 переключать своё состояние между «выключено» и «включено».
   1. При переходе в состояние «выключено» программа должна перестать реагировать на кнопку S1, а светодиоды должны погаснуть.
   2. При переходе в состояние «включено» разрешается работа кнопки S1, а светодиоды должны зажечься и гореть непрерывно.
2. По отпусканию кнопки S1 должен переключаться режим работы светодиодов между «мерцанием» и «непрерывным горением».

При этом не допускается подвисание устройства при нажатии и удерживании кнопок S1 и S2, а так же не должны использоваться прерывания и аппаратные таймеры.

**Теоретические сведения:**

### Экспериментальная плата MSP-EXP430F5529 разработана на основе микроконтроллера MSP430F5529 компании Texas Instruments. Это серия процессоров для обработки смешанных сигналов со сверхнизким энергопотреблением.

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;
* PxIES – выбор направления перепада для генерации запроса на прерывание: 0 – по фронту, 1 – по спаду;
* PxIE – разрешение прерывания;
* PxIFG – флаг прерывания.



Рисунок 1. Подключение пользовательских кнопок и светодиодов

Пользователю программно доступны две кнопки S1 и S2, подключенные соответственно к выводу 7 порта 1 и выводу 2 порта 2. В дальнейшем такое подключение будем обозначать как P1.7 и P2.2 соответственно. 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. Логика управления выводом на примере порта 1 представлена на рисунке ниже. Для других портов схемотехника может несколько отличаться, в зависимости от особенностей подключаемой к выводу периферии микроконтроллера.

### 

### Рисунок 2. Организация входа-выхода с триггером Шмидта на примере порта 1

### Прерывания

### Различают системные немаскируемые (SMNI), пользовательские немаскируемые (UNMI) и маскируемые прерывания. К системным немаскируемым прерываниям относятся: сигнал RST/NMI в режиме NMI, сбой генератора, ошибка доступа Flash памяти. К пользовательским немаскируемым прерываниям – 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). Кратко рассмотрим, как происходит обработка прерывания.

### 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. После того, как режим инициализирован, хорошим тоном считается перевод контроллера в режим пониженного энергопотребления. Сделать это можно, используя вызов \_\_bis\_SR\_register, например, следующий фрагмент переводит контроллер в режим LPM0 с разрешением прерываний:

### \_\_bis\_SR\_register(LPM0\_bits + GIE);

### Еще одной особенностью запуска в среде отладки Code Composer Studio является необходимость вызова \_\_no\_operation() перед завершением функции main, если она не использует некоторого цикла. Без этого вызова с завершением функции main завершится и выполнение кода в оболочке. Собственно обработчик прерывания описывается с использованием директивы #pragma vector. Например, фрагмент кода ниже описывает обработчик прерывания от порта ввода-выода 1:

### #pragma vector=PORT1\_VECTOR

### \_\_interrupt void PORT1\_ISR(void)

### {...}

### Алгоритм решения задачи

Разработанная программа работает по следующему алгоритму:

1. Инициализация
2. В бесконечном цикле:
   1. Если произошло событие отпускания кнопки, то
      1. Обработать его
      2. Сбросить событие
   2. Обновить состояние светодиодов.

В прерывании по отпусканию кнопки:

1. Подождать завершения дребезга
2. Если порт кнопки в высоком состоянии – зафиксировать событие отпускания
3. Иначе сбросить флаг прерывания и выйти

Обработка событий кнопок:

1. Если была отпущена кнопка S2, то
   1. Если работа устройства разрешена, то
      1. Погасить светодиоды
      2. Запретить работу устройства
   2. Иначе
      1. Разрешить работу устройства
      2. Включить светодиоды.
2. Иначе если была отпущена кнопка S1 и работа устройства разрешена, то
   1. Если мигание включено, то
      1. Выключить мигание
      2. Зажечь светодиоды
   2. Иначе включить мигание

### Листинг программы

#include <msp430.h>

/\*

Программа для микроконтроллера макетной платы MSP-EXP430F5529

Алгоритм работы программы:

- по отпусканию кнопки S2 переключать своё состояние между «выключено» и «включено».

При переходе в состояние «выключено» программа должна перестать реагировать на кнопку S1, а светодиоды должны погаснуть.

При переходе в состояние «включено» разрешается работа кнопки S1, а светодиоды должны зажечься и гореть непрерывно.

- по отпусканию кнопки S1 должен переключаться режим работы светодиодов между «мерцанием» и «непрерывным горением»

Условия:

- не допускается подвисание устройства при нажатии и удерживании кнопок

\*/

/\*

========== Initial variables ==========

\*/

volatile unsigned int enabled = 0;

volatile unsigned int blinking = 0;

volatile unsigned int delay = 0;

/\*

=========================================

\*/

// InitLeds

static void InitLeds()

{

//set ports to OUT mode

P8DIR |= BIT1;

P1DIR |= BIT0;

P8DIR |= BIT2;

}

// InitButtons

static void InitButtons()

{

//set port to IN

P1DIR &= !(BIT7);

P2DIR &= !(BIT2);

//enable pull-up

P1OUT |= BIT7; // set button 1 to output

P1REN |= BIT7; // enable pull-up

P2OUT |= BIT2;

P2REN |= BIT2;

}

// Turn leds to opposite state

static void TurnLedsToOppositeState()

{

P8OUT ^= BIT1;

P1OUT ^= BIT0;

P8OUT ^= BIT2;

}

// Turn leds On

static void TurnLedsOn()

{

P1OUT |= BIT0;

P8OUT |= BIT1;

P8OUT |= BIT2;

}

// Turn leds Off

static void TurnLedsOff()

{

P1OUT &= ~(BIT0);

P8OUT &= ~(BIT1);

P8OUT &= ~(BIT2);

}

// Enable interrupts for buttons ports

static void EnableInterrupts()

{

P1IES &= ~(BIT7); // Enable interrupt Low-High

P1IE |= BIT7; // Enable interrupts for the first port

P2IES &= ~(BIT2); // Enable interrupt Low-High

P2IE |= BIT2; // Enable interrupts for the second port

\_\_bis\_SR\_register(GIE); // Enable processing of interrupts

}

static void processBlinking()

{

if (--delay == 0)

{

if (blinking > 0 && enabled > 0) {

// blinking is ON -> switch leds to opposite state

TurnLedsToOppositeState();

} else if (blinking == 0 && enabled > 0) {

//blinking is OFF -> leds is ON

TurnLedsOn();

} else {

//Programm is OFF -> lights is OFF

TurnLedsOff();

}

delay = 6000;

}

}

int main(void)

{

WDTCTL = WDTPW | WDTHOLD;

// init states of controls

InitLeds();

InitButtons();

// turn leds off

TurnLedsOff();

// enable interrupts

EnableInterrupts();

while (1)

{

// process the programm state

processBlinking();

}

}

//Interrupt event for the S1 button

#pragma vector = PORT1\_VECTOR

\_\_interrupt void PORT1\_ISR(void) {

for (int i = 0; i < 30; i++);

// load button S1 state

if (P1IN & BIT7) {

if (enabled == 0) {

// If programm is OFF -> set programm to ON state

enabled = 1;

} else if (enabled == 1) {

// If programm is ON -> set programm to OFF state, blinking is OFF

enabled = 0;

blinking = 0;

}

}

// Reset interrupt flag

P1IFG = 0;

}

//Interrupt event for the S2 button

#pragma vector = PORT2\_VECTOR

\_\_interrupt void PORT2\_ISR(void) {

for (int i = 0; i < 30; i++);

// load button S2 state

if (P1IN & BIT7) {

if (blinking == 0 && enabled == 1) {

// If programm is ON and blinking is OFF -> set blinking to ON state

blinking = 1;

} else if (blinking == 1 && enabled == 1) {

// If programm is ON and blinking is ON -> set blinking to OFF state

blinking = 0;

}

}

// Reset interrupt flag

P2IFG = 0;

}

**Заключение**

В результате выполнения работы была написана требуемая программа и изучена работа с подсистемой прерываний микроконтроллера MSP430F5529.