Лабораторная работа №1

Цифровой ввод-вывод
Прерывания

Задание на ЛР1

В соответствии с вариантом задания написать программу, которая бы включала и выключала заданные светодиоды в зависимости от комбинации состояния кнопок.

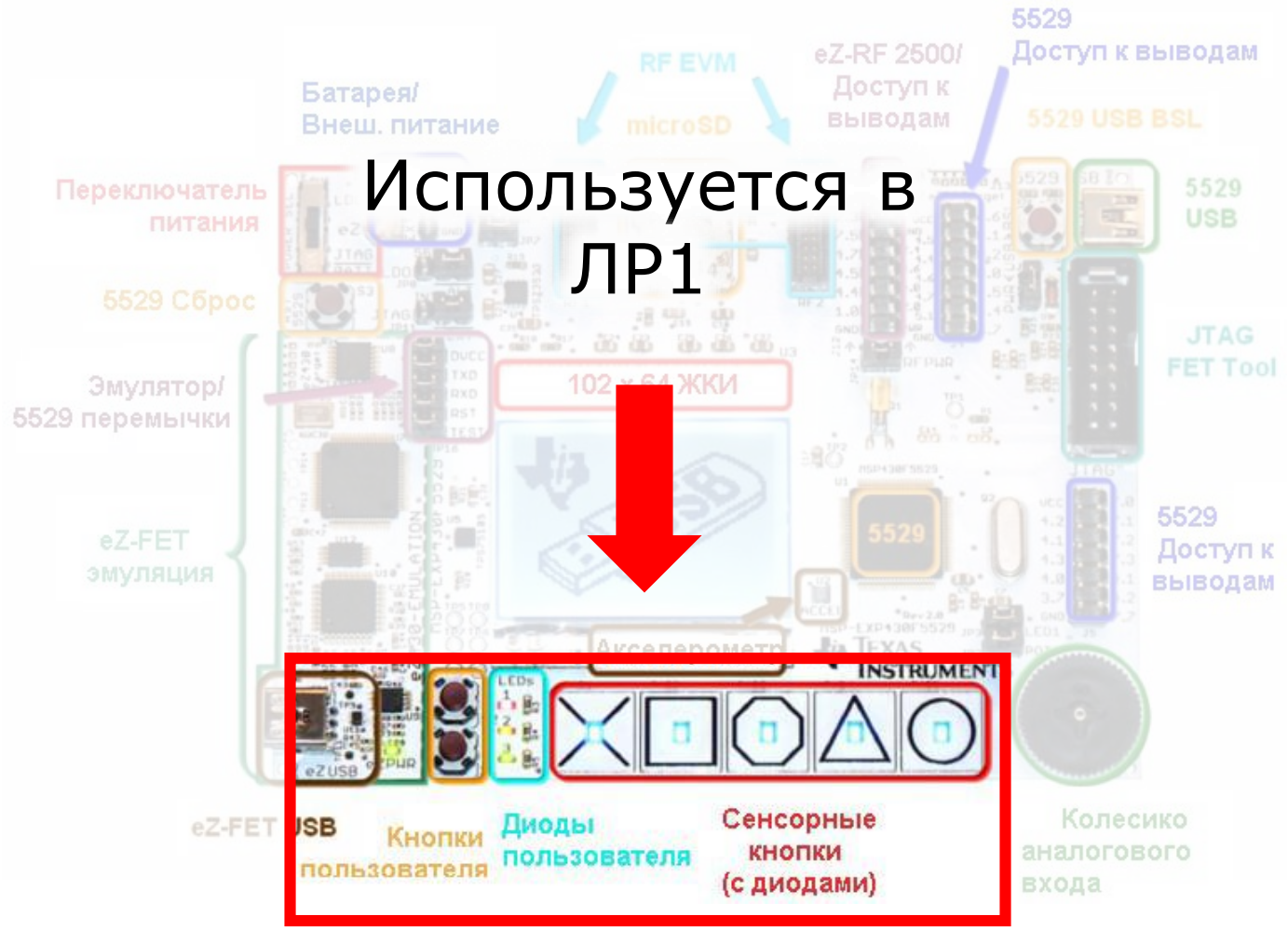
МОЖНО ПОДКЛЮЧАТЬ ТОЛЬКО:

“msp430.h”

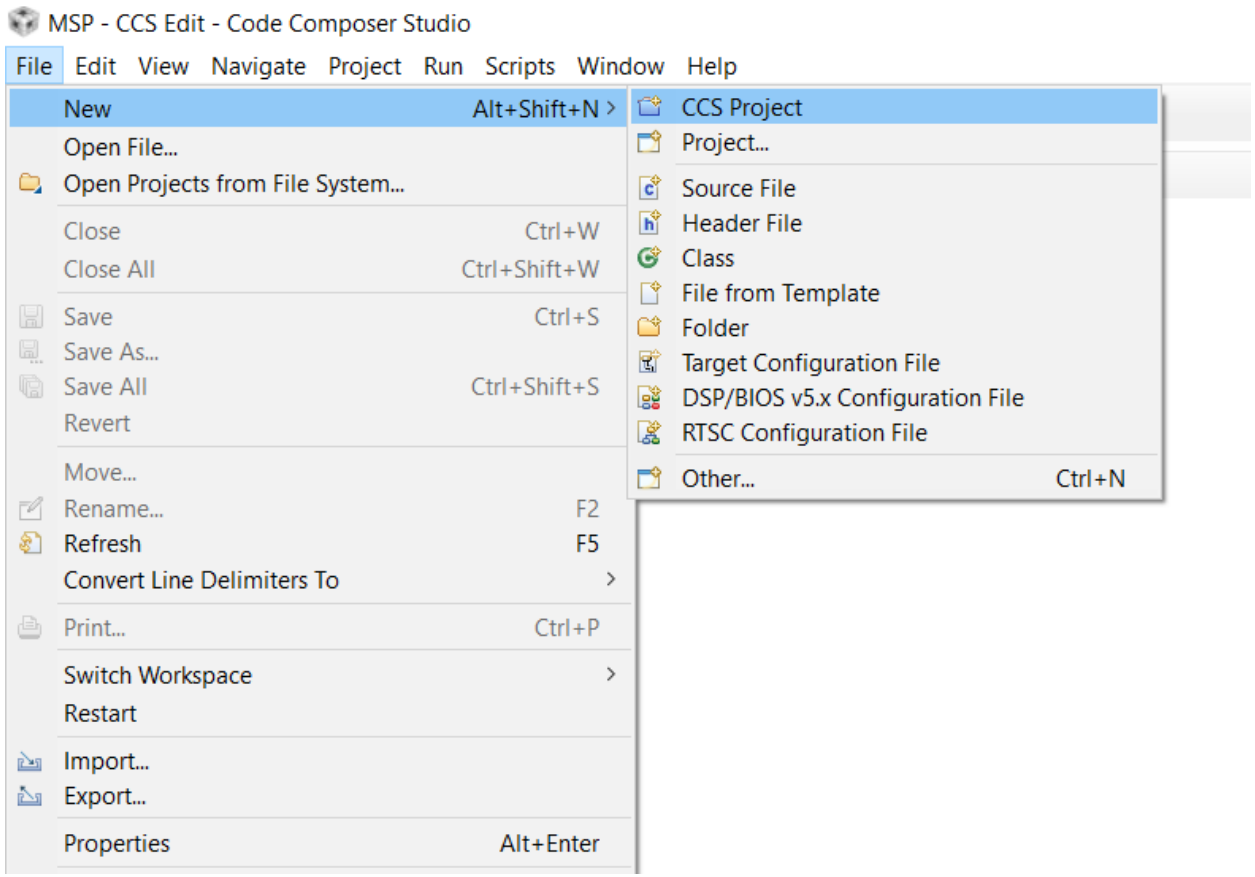
библиотеки языка С

файлы, написанные самостоятельно

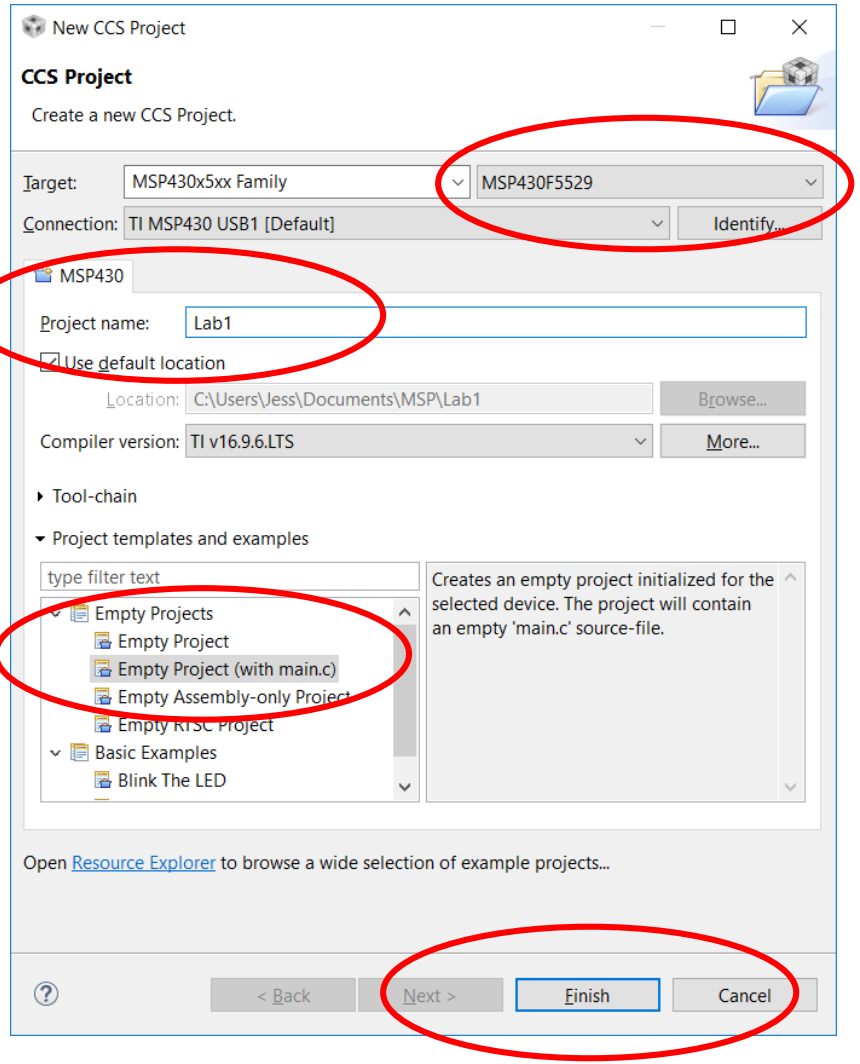
Плата MSP-EXP430F5529



Создание проекта в CCS

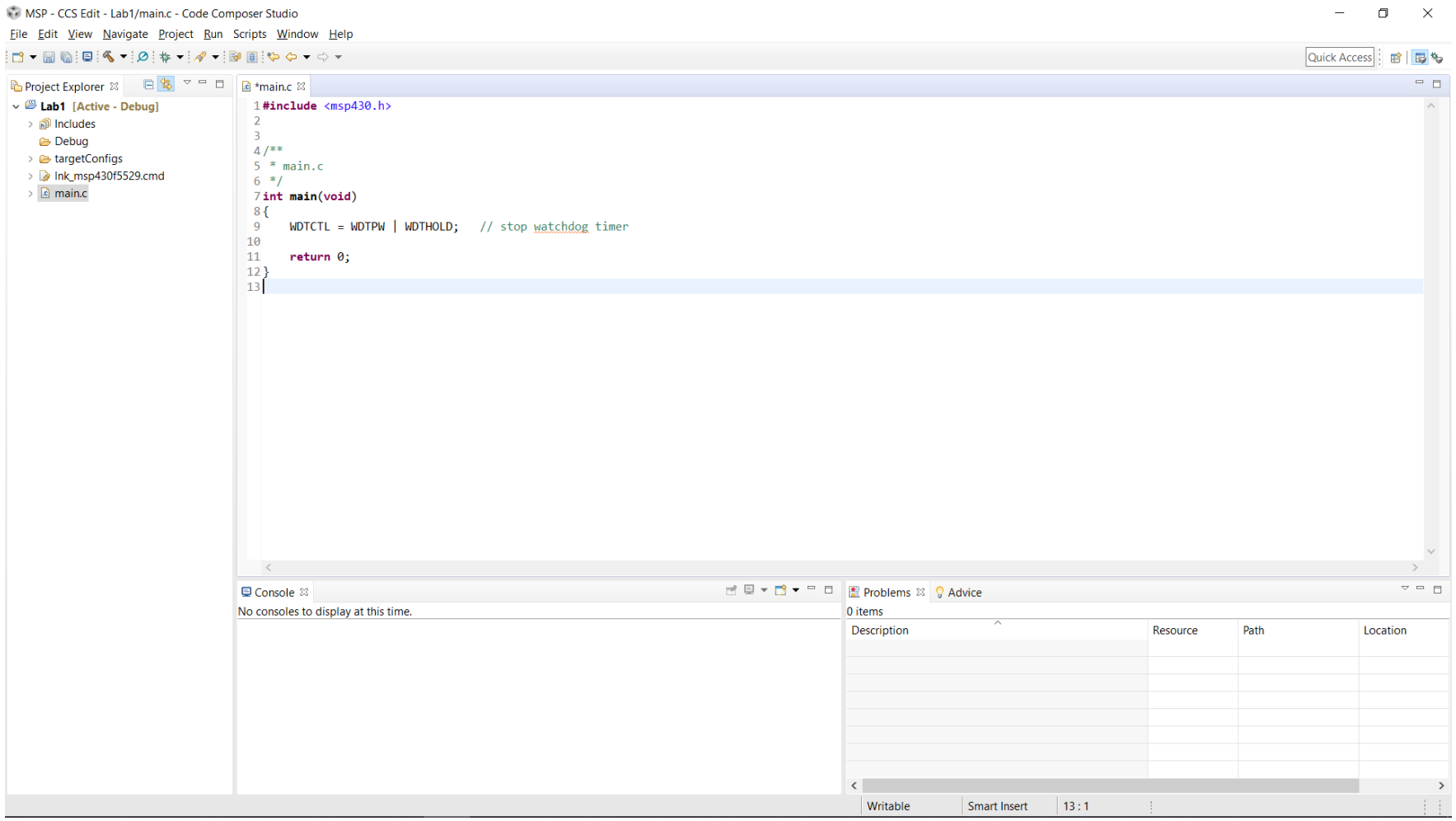


Создание проекта в CCS



Указать устройство
MSP430F5529

Проект CCS


MSP - CCS Edit - Lab1/main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
Lab1 [Active - Debug]
Includes
Debug
targetConfigs
lnk_msp430f5529.cmd
main.c
*main.c
#include <msp430.h>
/**
* main.c
*/
int main(void)
{
WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
return 0;
}
Console
No consoles to display at this time.
Problems
Advice
0 items
Description
Resource
Path
Location
Writable
Smart Insert
13 : 1

Сторожевой(Watchdog) таймер

Отключить:

*main.c

```c
#include <msp430.h>


/**
 * main.c
 */
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer

    return 0;
}
```

Порты

Порт X.Y

X – порт	Y – пин
Устройство ввода/вывода (набор пинов)	«Ножка» микроконтроллера
Регистры	Биты регистра

Типы регистров (необходимых для ЛР1)

PxIN – чтение данных с вывода;

PxOUT – установка значения выхода;

PxDIR – выбор направления: 0 – вход, 1 – выход;

PxREN – разрешение подтягивающего резистора;

PxSEL – выбор функции вывода: 0 – I/O, 1 – периферия;

Обращение к регистрам - запись

Порт X.Y

PxOUT (*операция*)= BITy

Арифметические и логические
(необязательно)

Можно изменять сразу нескольких битов в регистрах

Обращение к регистрам - запись

В бит 3 регистра PxOUT порта 1 (P1.3) записать 1

```
P1OUT |= BIT3
```

В бит 6 регистра PxDIR порта 2 (P1.3) записать 0

```
P2OUT &= ~BIT6
```

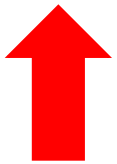
Обращение к портам - чтение

Порт X.Y

PxIN – чтение ВСЕГО регистра

Если необходимо обратиться к отдельному пину:

PxIN операция BITy



Логические

Обращение к портам - чтение

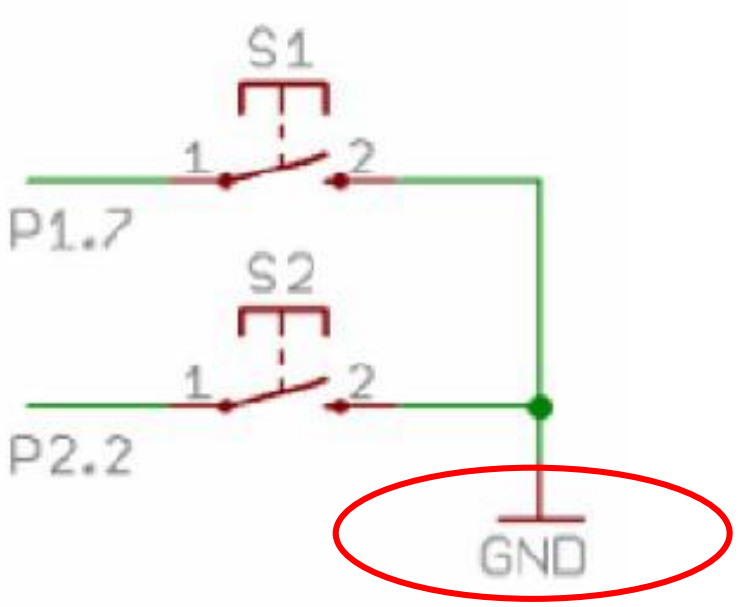
Записать байт из регистра PxIN порта 2 в переменную

```
t = P2IN
```

Выделить значение бита 7 регистра PxIN порта 1 с помощью битовой маски

```
P1IN&BIT7
```

Кнопки

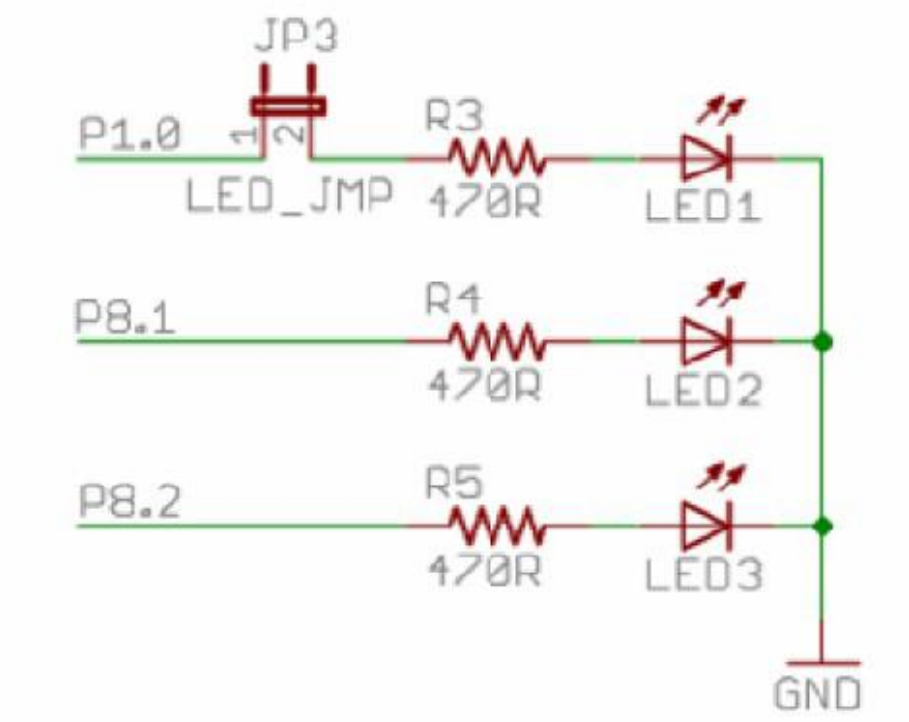


Кнопка S1 – порт 1.7

Кнопка S2 – порт 2.2

При замыкании кнопка подключается к «нулю»

Светодиоды



Активный уровень «единица»

LED1 – порт 1.0
LED2 – порт 8.1
LED3 – порт 8.2

Расположены рядом с S1 и S2

LED4 – порт 1.1
LED5 – порт 1.2
LED6 – порт 1.3
LED7 – порт 1.4
LED8 – порт 1.5

Расположены в блоке сенсорных кнопок

Особенности

Мыслить как программист

Я написал код:

- Знание алгоритмов говорит мне, что это должно работать
- В отладке все работает как задумывалось

НО НА ДЕЛЕ ОНО РАБОТАЕТ НЕ ТАК, КАК НУЖНО!!!

Мыслить как разработчик аппаратуры

Нужно учитывать, что:

Устройства имеют определенные физические характеристики

Изменение сигналов происходит не мгновенно

Существуют переходные состояния

FAQ

Решение не собирается

Ищите ошибки в коде :)

Если долго не можете найти ошибку – зовите, спрашивайте, свежим взглядом найти ошибку проще

FAQ

Решение собралось, но на плату не заливается

Проверить, подключена ли плата

Проверить, указано ли нужное устройство (MSP430F5529)

FAQ

Решение залилось на плату, но ничего не происходит (не горят диоды, кнопки не реагируют на нажатие)

Проверить, отключен ли Watchdog таймер

Проверить инициализацию портов

Заменить плату

Советы

Вспомнить особенности арифметических и логических операций

Не забывать мыслить как разработчик аппаратуры

Не надеяться на значения по умолчанию

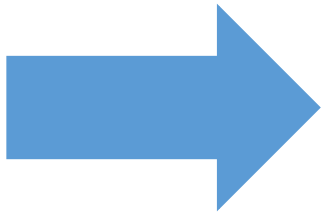
Выходное значение лучше указывать после остальной инициализации

Учитывать, что на разных платах программа может выполняться по-разному

Прерывания

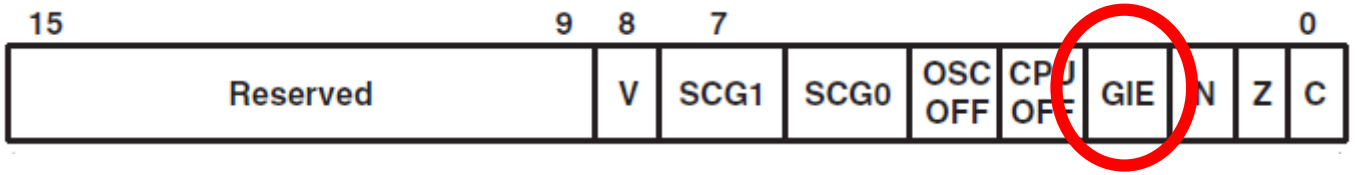
Особенность запуска в отладке

```
while (true)
{
…
};
```



```
__no_operation();
```

Регистр состояния SR



~~*SR |= mask*~~ → *__bis_SR_register(mask);*

Разрешить пользовательские маскируемые прерывания!

Типы регистров

PxIN – чтение данных с вывода;
PxOUT – установка значения выхода;
PxDIR – выбор направления: 0 – вход, 1 – выход;
PxREN – разрешение подтягивающего резистора;
PxSEL – выбор функции вывода: 0 – I/O, 1 – периферия;

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

Работа с прерываниями I/O портов

1. Инициализация порта для работы с кнопкой (см. ЛР1)
2. Разрешение прерывания (PxIE |= BITy)
3. Выбор направления перепада для генерации запроса (если необходимо)
4. Сброс флага прерывания (лучше это делать явно всегда в конце инициализации)

Обработчик прерывания I/O портов

```
#pragma vector = PORTx_VECTOR
__interrupt void ISR(void)
{...}
```

x – номер порта (1 для S1 и 2 для S2)

ISR – название функции обработчика прерывания

Особенности

PxIFG не сбрасывается автоматически при завершении обработчика прерывания

Запись в регистры PxOUT, PxDIR и PxREN может быть причиной установки значений в регистре PxIFG

Не запрещается изменять направление перепада для генерации прерывания внутри обработчика (если это необходимо)

Необходимо помнить о необходимости разрешения пользовательских маскируемых прерываний

Вопросы на защите

1. Особенности архитектуры микроконтроллера MSP430;
2. Физические принципы работы портов;
3. Типы регистров, доступных портам, и их назначение;
4. Общая теория по прерываниям (прерывание, вектор прерывания, таблица векторов прерываний, типы прерываний)
5. Особенности работы с кнопками и диодами.