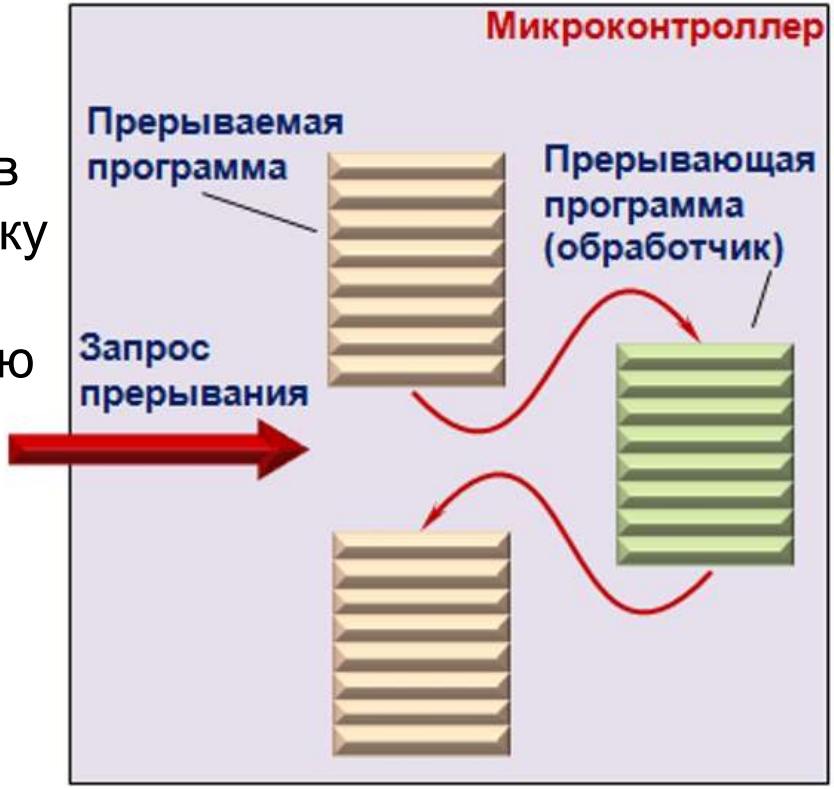
Лекция 3

Внешние прерывания

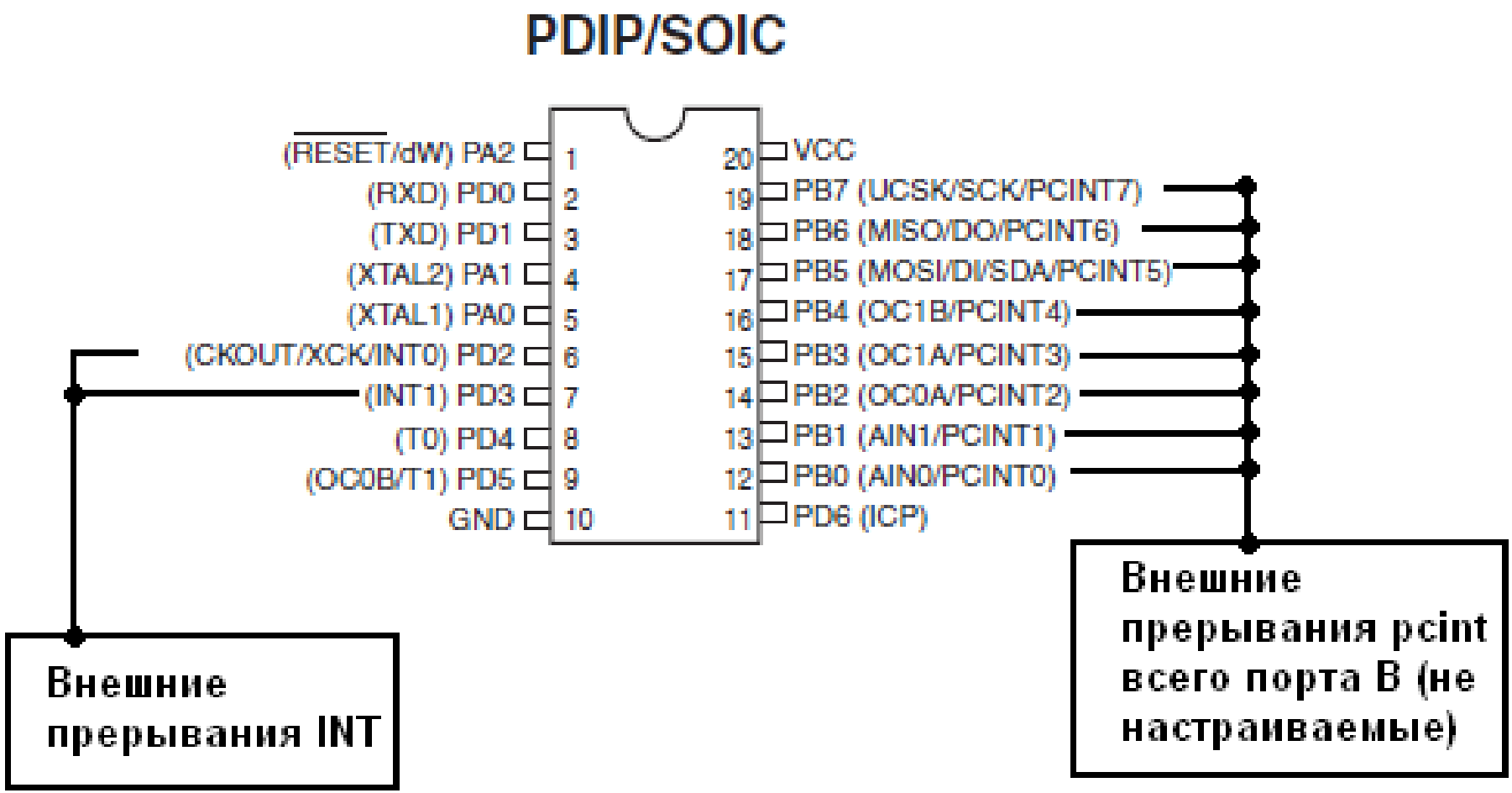
Понятие прерывания

Прерывание (interrupt) – событие, требующие немедленной реакции со стороны процессора. Реакция состоит в том, что процессор прерывает обработку текущей программы (***прерываемой программы***) и переходит к выполнению некоторой другой программы (***прерывающей программы***), специально предназначенной для данного события. По завершении этой программы процессор возвращается к выполнению прерванной программы.

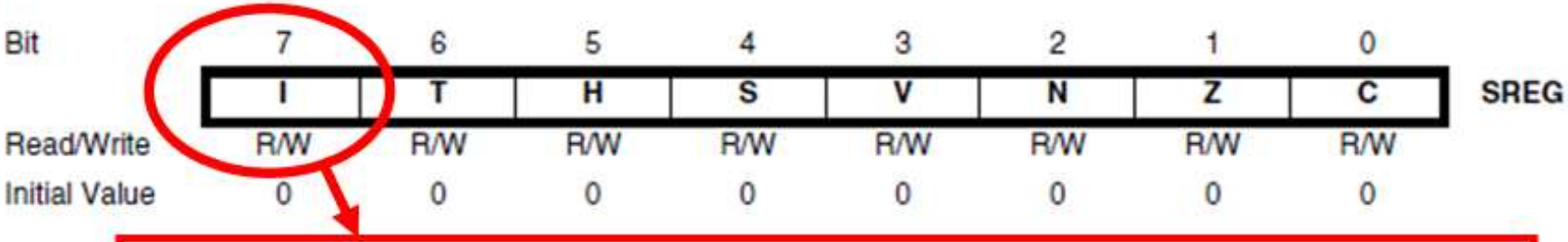


Прерывания бывают внутренними и внешними.

Внешние прерывания микроконтроллера ATtiny2313



Глобальное разрешение и глобальный запрет прерываний в регистре статуса SREG

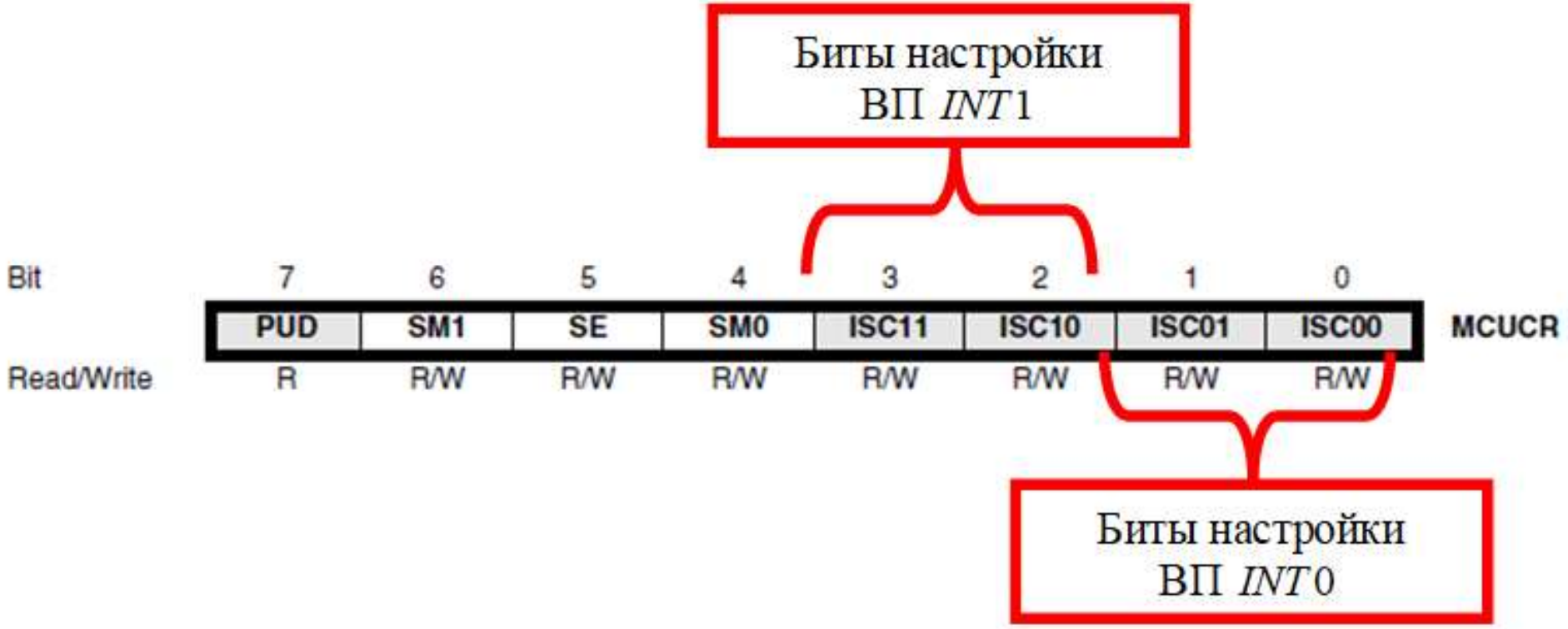


Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Бит *I* установлен – глобальное разрешение прерываний (язык Си: sei(); язык Ассемблера: sei)

Бит *I* сброшен – глобальный запрет прерываний (язык Си: cli(); язык Ассемблера: cli)

Регистр управления внешними прерываниями микроконтроллера - MCUCR



Биты 1 и 0 - ISC01, ISC00: для настройки прерывания INT0

Биты 3 и 2 - ISC11, ISC10 : для настройки прерывания INT1

Режимы настройки ВП INT0

Биты регистра MCUCR		Описание
ISC01	ISC00	
0	0	ВП вызывается по низкому уровню
0	1	ВП вызывается по изменению логического уровня на выводе
1	0	ВП вызывается по заднему (спадающему) фронту
1	1	ВП вызывается по переднему (нарастающему) фронту

Режимы настройки ВП INT1

Биты регистра MCUCR		Описание
ISC11	ISC10	
0	0	ВП вызывается по низкому уровню
0	1	ВП вызывается по изменению логического уровня на выводе
1	0	ВП вызывается по заднему (спадающему) фронту
1	1	ВП вызывается по переднему (нарастающему) фронту

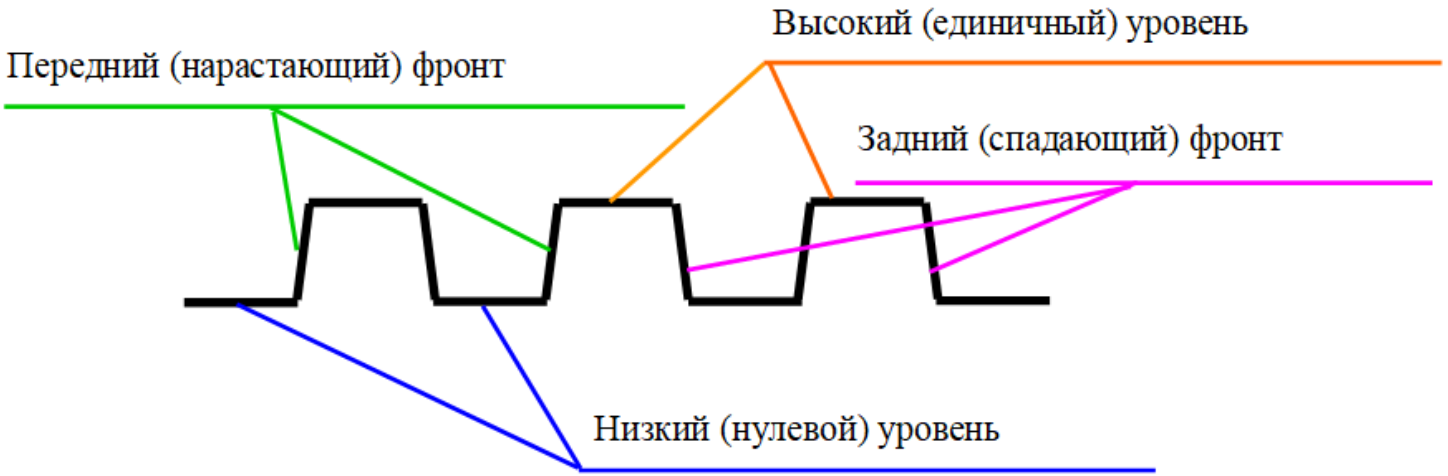
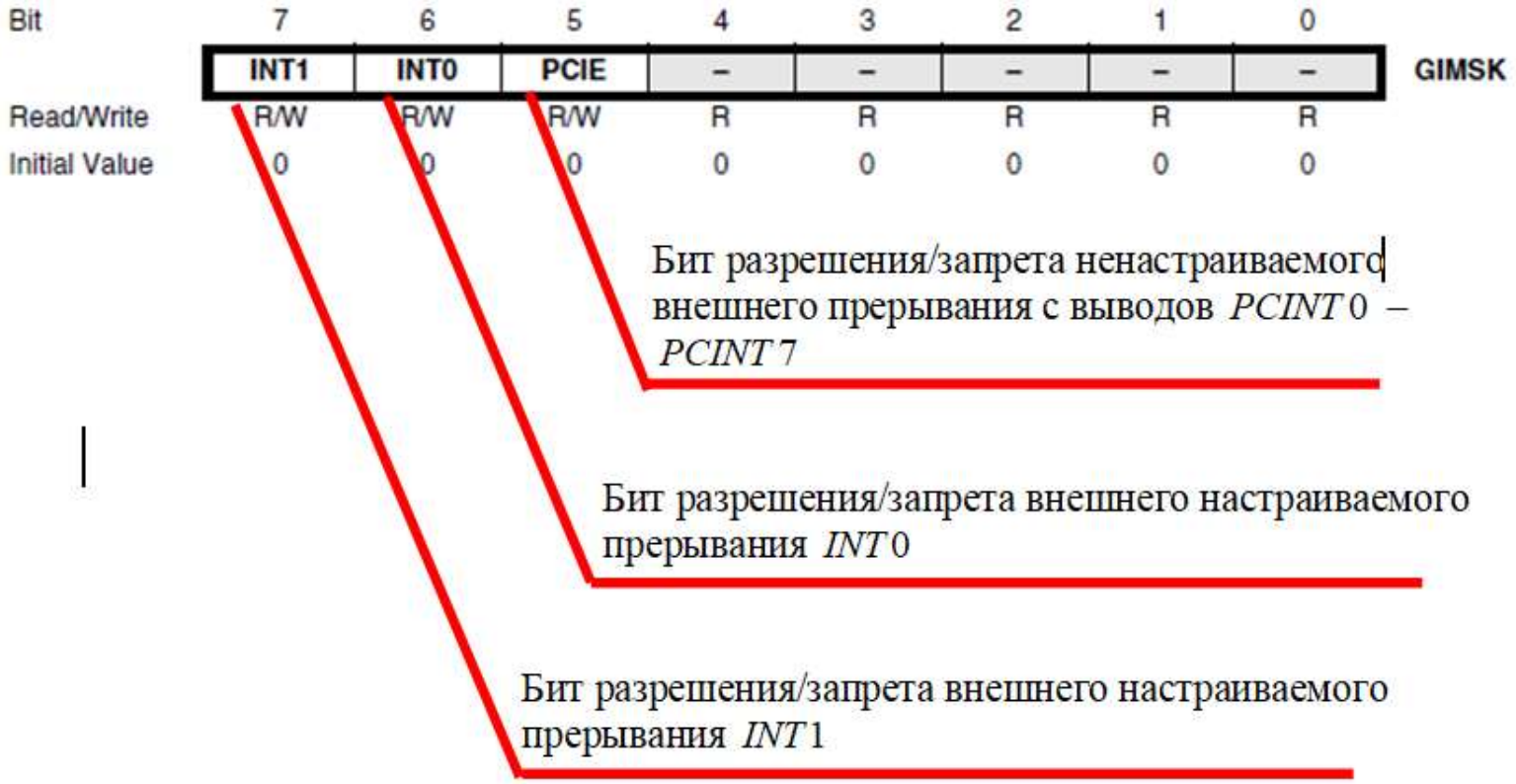


Рисунок - Элементы цифрового сигнала

Регистр маски прерываний – GIMSK



Бит **7 - INT1 :** Разрешение внешнего Прерывания **INT1**
Вызов прерывания на выводе INT1 разрешен, если бит INT1 текущего регистра и бит I в регистре SREG установлены в лог. 1.
Биты ISC11 и ISC10 в регистре MCUCR определяют в каком режиме будет считываться сигнал прерывания на выводе INT1.

Бит **6 – INT0 :** Разрешение внешнего Прерывания **INT0**

Бит **5 - PCIE:** Разрешение прерывания по изменению состояния выводов

Если бит PCIE установлен в единицу, и при этом установлен Флаг I регистра SREG, то прерывания по изменению состояния любого контакта разрешено.

Какие именно контакты будут вызывать прерывание, определяется индивидуально установкой регистра PCMSK.

Регистр Маски прерываний по изменению на любом из контактов PCINT7..0 - PCMSK

Каждый бит из PCINT7..0 отвечает за свой вывод.

Установка какого-нибудь бита из PCINT7..0 разрешает соответствующему I/O-выводу работать в качестве источника прерывания, при условии, что установлен бит PCIE в GIMSK.

Если какого-нибудь бит из PCINT7..0 очищен (ноль), то соответствующий I/O-вывод не будет работать в качестве источника прерывания.

Регистр флагов внешних прерываний – EIFR

Бит **7 - INTF1:** Флаг внешнего прерывания **INT1**
Когда происходит изменение логического уровня на выводе INT1, то флаг INTF1 устанавливается в 1, благодаря чему вызывается прерывание. Этот флаг очищается аппаратно при запуске процедуры обработки прерывания.

Бит **6 – INTF0:** Флаг внешнего прерывания **INT0**

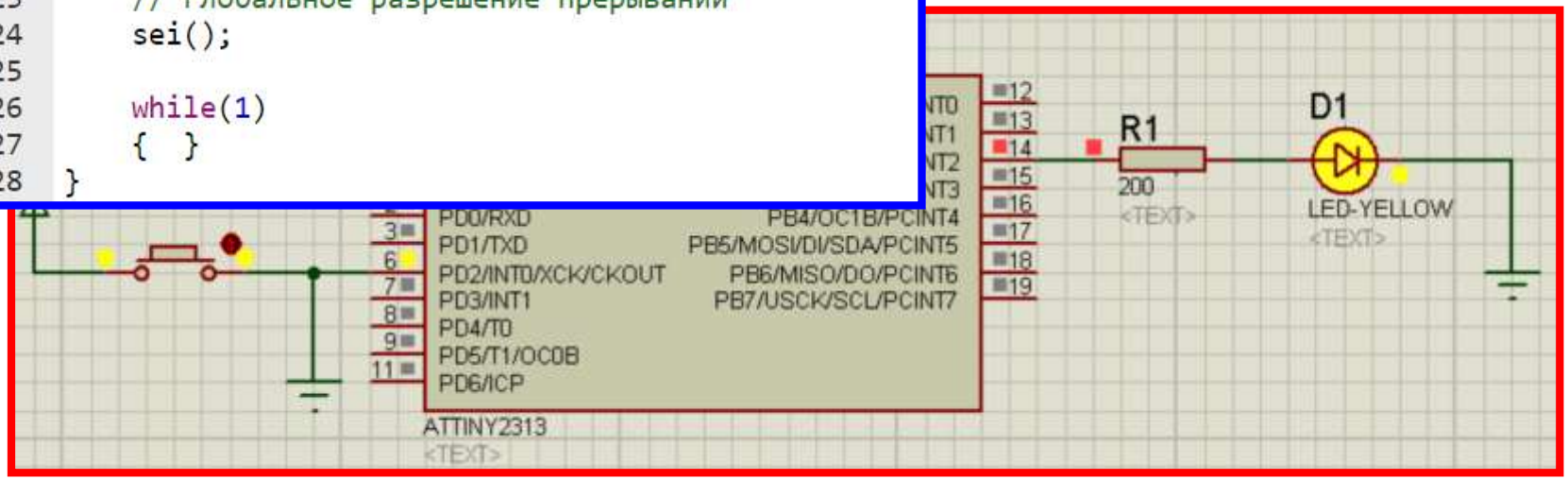
Бит **5 - PCIF:** Флаг прерывания по изменению состояния одного из выводов

Пример структуры программного кода для обработки внешних прерываний

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(INT0_vect) //обработка прерывания INT0
// или SIGNAL(SIG_INTERRUPT0)
{ cli(); //запрещение прерываний на время обработки прерывания
  // подпрограмма, которая должна выполнятся при срабатывании прерывания INT0
  sei(); //разрешение прерываний
}

ISR(INT1_vect) //обработка прерывания INT1
// или SIGNAL(SIG_INTERRUPT1)
{ /*подпрограмма, которая должна выполнятся при срабатывании прерыв. INT1*/  }

int main(void)
{
  GIMSK=0b11000000;  //разрешаем прерывание INT0 и INT1
  MCUCR=0b00001111;

  sei(); //Разрешение прерываний глобально по всей программе
  while(1)
  {  /*код программы, которая будет выполняться в основном цикле*/  }
}
```

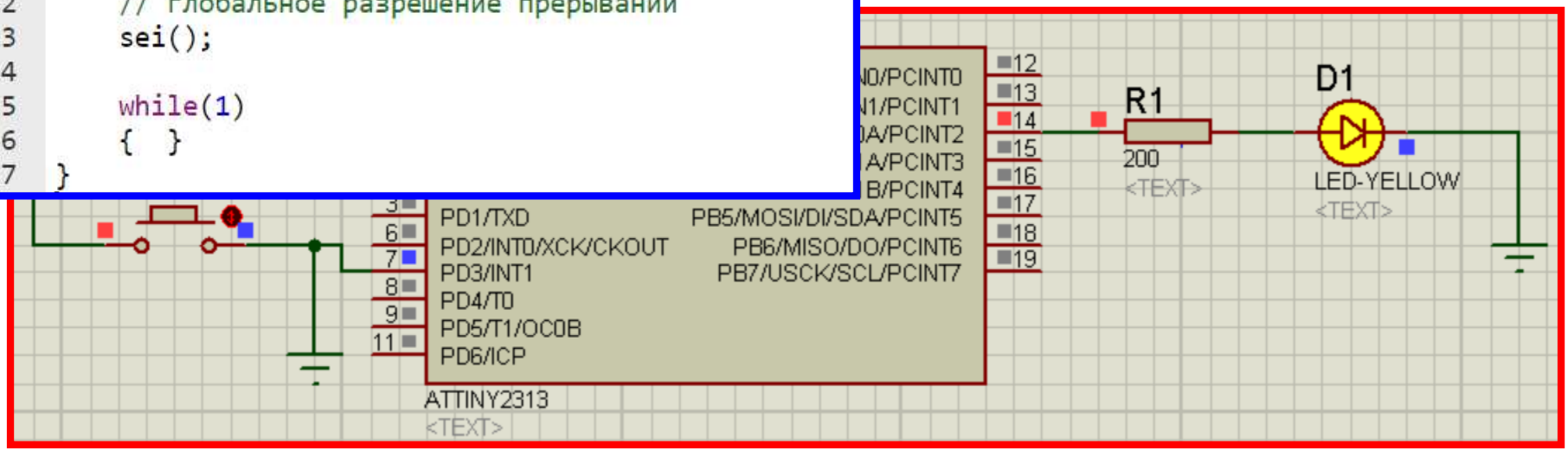
```c
#include <avr/io.h>
#include <avr/interrupt.h> //библиотека прерываний

// подпрограмма обработки ВП INT0
ISR(INT0_vect)
{
    // включение/выключение светодиода
    PORTB = PORTB ^ (1 << 2);
}

int main(void)
{
    // настройка вывода PB2 на вывод данных
    DDRB = (1 << 2);
    PORTB = 0;

    //настройка ВП INT0 (передний фронт)
    MCUCR = 0b11;

    //разрешение прерывания INT0
    GIMSK = (1 << 6);

    // глобальное разрешение прерываний
    sei();

    while(1)
    { }
}
```

Пример программы, демонстрирующей работу ВП INT0: включение/выключение светодиода (вывод микроконтроллера PB2) нажатием кнопки (вывод микроконтроллера INT0).



```c
#include <avr/io.h>
#include <avr/interrupt.h> //библиотека прерываний

// подпрограмма обработки ВП INT1
ISR(INT1_vect)
{
    // включение/выключение светодиода
    PORTB = PORTB ^ (1 << 2);
}

int main(void)
{
    // настройка вывода PB2 на вывод данных
    DDRB = (1 << DDB2);

    //настройка ВП INT1 (задний фронт)
    MCUCR = (1 << ISC11);

    //разрешение прерывания INT1
    GIMSK = (1 << INT1);

    // глобальное разрешение прерываний
    sei();

    while(1)
    { }
}
```

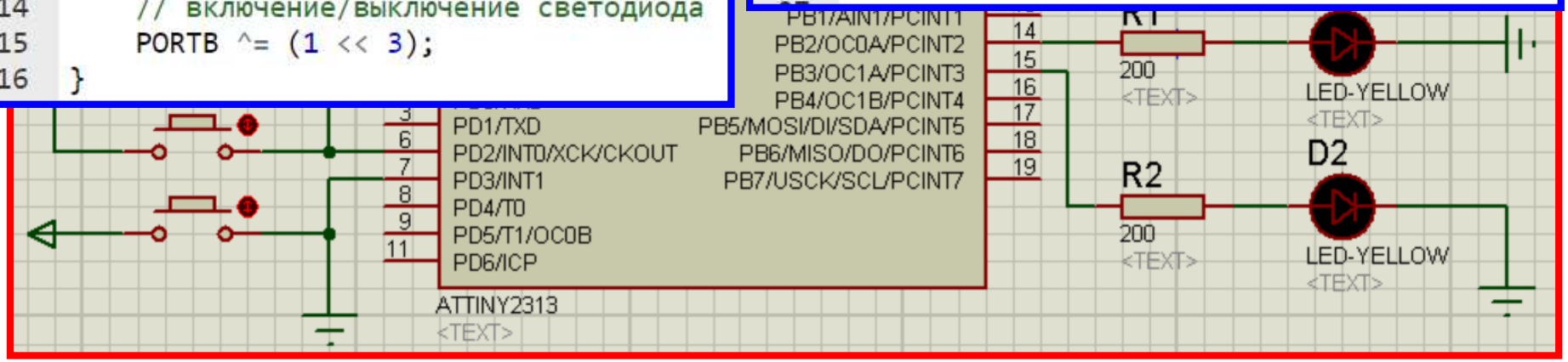
Пример программы, демонстрирующей работу ВП INT1: включение/выключение светодиода (вывод микроконтроллера PB2) нажатием кнопки (вывод микроконтроллера INT1).



Пример программы, демонстрирующей работу ВП INT0 и INT1: вкл./выкл. светодиода (вывод PB2) нажатием кнопки (вывод INT0) и вкл./выкл. светодиода (вывод PB3) нажатием кнопки (вывод INT1) .

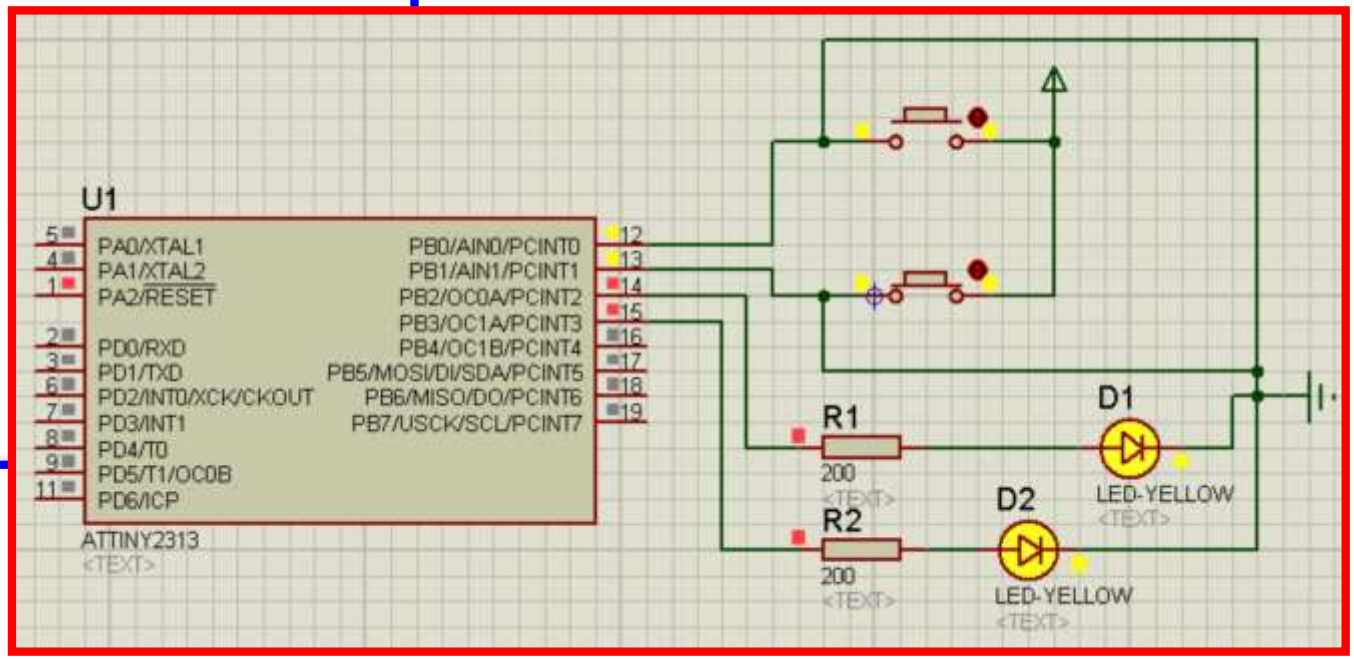
```
#include <avr/io.h>
#include <avr/interrupt.h> //библиотека

// подпрограмма обработки ВП INT0
ISR(INT0_vect)
{
    // включение/выключение светодиода
    PORTB = PORTB ^ (1 << 2);
}

// подпрограмма обработки ВП INT1
ISR(INT1_vect)
{
    // включение/выключение светодиода
    PORTB ^= (1 << 3);
}
```

```
int main(void)
{
    // настройка вывода PB2 на вывод данных
    DDRB = (1 << DDB2) | (1 << DDB3);

    //настройка ВП INT0 (передний фронт)
    MCUCR = (1 << ISC00) | (1 << ISC01);
    //настройка ВП INT1 (передний фронт)
    MCUCR |= (1 << ISC10) | (1 << ISC11);

    //разрешение прерываний INT0 и INT1
    GIMSK = (1 << INT0) | (1 << INT1);

    // глобальное разрешение прерываний
    sei();

    while(1)
    { }
}
```



```c
#include <avr/io.h>
#include <avr/interrupt.h>

// подпрограмма обработки
//не настраиваемого прерывания
ISR(PCINT_vect)
{
    // включение (передний фронт) и
    // выключение (задний фронт) светодиодов
    PORTB = PORTB ^ ((1 << PB2) | (1 << PB3));
}

int main(void)
{
    // настройка выводов PB2 и PB3
    DDRB = (1 << DDB2) | (1 << DDB3);

    // разрешение ВП на выводах
    // PCINT0 и PCINT1
    PCMSK = 0b11;

    //разрешение
    //не настраиваемого
    //прерывания
    GIMSK = (1 << PCIE);

    sei();

    while(1)
    {  }
}
```

Пример программы, демонстрирующей работу не настраиваемого ВП: вкл. и выкл. светодиодов (вывода PB2 и PB3) нажатием кнопок (вывода PCINT0 и PCINT1) .



```c
#include <avr/io.h>
#include <avr/interrupt.h>

// подпрограмма обработки
//не настраиваемого прерывания
ISR(PCINT_vect)
{
   if (PINB & (1 << PINB0))
       PORTB ^= (1 << PB2);

  if (PINB & (1 << PINB1))
       PORTB ^= (1 << PB3);
}

int main(void)
{
    DDRD = 0xFF;

    // настройка выводов PB2 и PB3
    DDRB = (1 << DDB2) | (1 << DDB3);

    // разрешение ВП на выводах
    // PCINT0 и PCINT1
    PCMSK = (1 << PCINT0) | (1 << PCINT1);

    //разрешение не настраиваемого прерывания
    GIMSK = (1 << PCIE);

    // глобальное разрешение прерываний
    sei();

    while(1)
    {  }
}
```

Пример программы, демонстрирующей работу не настраиваемого ВП: вкл./выкл. светодиода (вывод PB2) нажатием кнопки (вывод PCINT0) и вкл./выкл. светодиода (вывод PB3) нажатием кнопки (вывод PCINT1) .