

## Лабораторная работа №5

### Прерывания. Внешние прерывания

**Цель работы:** изучение процессов прерываний и их видов, получение навыков работы с системой прерываний на примере микроконтроллера AVR ATMEGA328P.

#### Теоретическая часть

**Прерывание** – остановка выполнения текущей задачи процессора в результате возникновения некоторого события. **Событие** – это некоторая ситуация, на которое может реагировать процессорное ядро или периферия. Прерывания принято делить по источнику события на **внутренние** и **внешние**.

**Внешние (асинхронные) прерывания** – происходят при появлении сигналов, исходящих от периферийных (внешних) устройств, которым необходимо «внимание» со стороны центрального процессора. Такие сигналы формируют событие называемое IRQ (Interrupt ReQuest – запрос на прерывание). Внешние прерывания могут произойти в любой произвольный момент времени и поэтому называются асинхронными (например: нажатие кнопки, запрос на передачу данных).

**Внутренние прерывания** – происходят при возникновении события в самом процессоре. Примерами таких событий могут быть: деление на ноль, переполнение стека, обращение к недопустимым адресам памяти или недопустимый код операции. Частным случаем внутренних прерываний являются **программные прерывания** – событие возникает в результате исполнения специальной инструкции в коде программы.

Обработка прерывания представляет собой следующую последовательность шагов:

- возникновение события инициирующего прерывание;
- сохранение текущего состояния процессора (в AVR программист сам должен записать состояние рабочих регистров в стек);
- выполнение процессором специальной процедуры – обработчика прерывания;

- восстановление состояния процессора;
- продолжение выполнения основной программы.

**Обработчик прерывания** - специальная программа, выполняемая при возникновении прерывания. Её задача – обработка события, вызвавшего остановку выполнения основной программы. Обработчик разрабатывается программистом. Он обычно содержит минимальный набор команд необходимый для реакции на событие, несравнимый по объему и сложности с основной программой. У каждого прерывания может быть свой обработчик (очевидно, что разные события, произошедшие в системе, требуют различной реакции на них).

В начальных адресах программной памяти размещены **векторы прерываний** – указатели на начальный адрес процедуры обработчика прерывания. Область памяти, в которой размещены векторы, называется **таблицей векторов прерываний**.

### **Прерывания в ATmega328P**

Микроконтроллер ATmega328P имеет 26 обработчиков прерываний (см. таблицу 1), каждый из которых может быть вызван соответствующим событием, возникшем в системе. У каждого прерывания есть строго определенный приоритет. Приоритет прерывания зависит от его расположения в таблице векторов прерываний. Чем меньше номер вектора в таблице, тем выше приоритет, т.е. самый высокий приоритет имеет прерывание сброса (Reset interrupt), которое располагается первой в таблице, а соответственно и в памяти программ. Внешнее прерывание INT0 (INT – interrupt – прерывание), идущее следом за прерыванием Reset в “таблице векторов прерываний”, имеет приоритет меньше чем у Reset, но выше чем у всех остальных прерываний и т.д. Разрешение (или запрет) всех прерываний определяет бит I регистра SREG). Для того, чтобы микроконтроллер обрабатывал прерывания необходимо, чтобы этот бит был установлен в 1. Управление битом осуществляется следующими командами CLI – сброс бита в «0» (запрет всех прерываний), SFI – установка бита в «1» (разрешение всех прерываний). Однако, прерывание Reset, в отличие от всех остальных, нельзя запретить. Такие прерывания называют не маскируемыми Non-maskable interrupts.

Таблица 1 – Таблица векторов прерываний МК Atmega328P

№	Адрес	Источник	Описание	Вектор
1	0x0000	RESET	Вектор сброса	
2	0x0002	INT0	Внешнее прерывание 0	INT0_vect
3	0x0004	INT1	Внешнее прерывание 1	INT1_vect
4	0x0006	PCINT0	Прерывание по изменению состояния выводов группы 0	PCINT0_vect
5	0x0008	PCINT1	Прерывание по изменению состояния выводов группы 1	PCINT1_vect
6	0x000A	PCINT2	Прерывание по изменению состояния выводов группы 2	PCINT2_vect
7	0x000C	WDT	Таймаут сторожевого таймера	WDT_vect
8	0x000E	TIMER2_COMPA	Совпадение А таймера/счетчика T2	TIMER2_COMPA_vect
9	0x0010	TIMER2_COMPB	Совпадение В таймера/счетчика T2	TIMER2_COMPB_vect
10	0x0012	TIMER2_OVF	Переполнение таймера/счетчика T2	TIMER2_OVF_vect
11	0x0014	TIMER1_CAPT	Захват таймера/счетчика T1	TIMER1_CAPT_vect
12	0x0016	TIMER1_COMPA	Совпадение А таймера/счетчика T1	TIMER1_COMPA_vect
13	0x0018	TIMER1_COMPB	Совпадение В таймера/счетчика T1	TIMER1_COMPB_vect
14	0x001A	TIMER1_OVF	Переполнение таймера/счетчика T1	TIMER1_OVF_vect
15	0x001C	TIMER0_COMPA	Совпадение А таймера/счетчика T0	TIMER0_COMPA_vect
16	0x001E	TIMER0_COMPB	Совпадение В таймера/счетчика T0	TIMER0_COMPB_vect
17	0x0020	TIMER0_OVF	Переполнение таймера/счетчика T0	TIMER0_OVF_vect
18	0x0022	SPI STC	Передача по SPI завершена	SPI_STC_vect
19	0x0024	USART_RX	USART прием завершен	USART_RX_vect
20	0x0026	USART_UDRE	Регистр данных USART пуст	USART_UDRE_vect
21	0x0028	USART_TX	USART передача завершена	USART_TX_vect
22	0x002A	ADC	Преобразование АЦП завершено	ADC_vect
23	0x002C	EE READY	Готовность EEPROM	EE_READY_vect
24	0x002E	ANALOG COMP	Прерывание от аналогового компаратора	ANALOG_COMP_vect
25	0x0030	TWI	Прерывание от модуля TWI (I2C)	TWI_vect
26	0x0032	SPM READY	Готовность SPM (записи в память команд)	SPM_READY_vect

Адреса таблицы векторов прерываний зависят от битов IVCE, IVSEL регистра MCUCR (см. таблицу 2).

Таблица 2 – Варианты расположения таблицы векторов прерываний

BOOTRST	IVSEL	Адрес вектора сброса (Reset)	Адрес начала таблицы векторов маскируемых прерываний
1	0	0x0000	0x0002
1	1	0x0000	Начальный адрес секции загрузчика + 0x0002
0	0	Начальный адрес секции загрузчика	0x0002
0	1	Начальный адрес секции загрузчика	Начальный адрес секции загрузчика + 0x0002

**MCUCR** (MCU (microcontroller unit) control register) – регистр управления микроконтроллером

Bit	7	6	5	4	3	2	1	0
		BODS	BODSE	PUD			IVSEL	IVCE
Доступ		R/W	R/W	R/W			R/W	R/W
Начальное Состояние		0	0	0			0	0

Примечание: биты BODS и BODSE есть только на микроконтроллерах подсемейства *ricoPower* ATmega48PA/88PA/168PA/328P. Система BOD (Brown-Out Detection) контролирует напряжение питания и в случае если оно ниже нормы BOD сбрасывает контроллер во избежание ошибок. При восстановлении уровня напряжения сигнал сброса снимается и микроконтроллер возвращается к работе. Напряжение питания, при котором система сбрасывает микроконтроллер, устанавливается в одном из 4 \*Fuse регистров.

*\*Fuse регистры – конфигурационные регистры, которые отвечают за предварительную настройку микроконтроллера. Они расположены в отдельном адресном пространстве. В AVR биты fuse регистров определяют - внешний или внутренний генератор использовать; частоту тактирования (включение и отключение делителя частоты синхросигнала); частоту внутреннего генератора; запрет на чтение прошивки микроконтроллера; включение или выключение таймеров; переопределение вывода RESET в вывод порта данных; включение защиты EEPROM от стирания. Подробнее смотри описание fuse битов Atmega328P.*

**BODS:** BOD Sleep (запрет детектора некачественного напряжения питания во время режимов сна, sleep mode). BOD постоянно отслеживает напряжение питания во время режима сна. Чтобы экономить энергию, BOD можно запретить программно для некоторых режимов сна. Энергопотребление в режиме сна будет тогда такое же, как если бы BOD был глобально запрещен фьюзами. Если BOD запрещен программно, то функция BOD выключается сразу после входа в режим сна. После выхода из режима сна работа BOD автоматически разрешается. При этом требуется примерно 60  $\mu$ s, чтобы гарантировать корректную работу BOD до того, как MCU продолжит выполнение кода.

Установка бита BODS в «1» выключает систему в соответствующих режимах сна, а запись 0 восстанавливает её работу во всех режимах. По умолчанию BOD активен всегда, т. е. бит BODS сброшен в 0. Изменение бита BODS требует 3 машинных циклов (тактов синхроимпульса).

**BODSE** (BOD Sleep Enable) – разрешение отключения системы BOD (необходимо установить 1, затем в течении 4 машинных циклов (тактов синхросигнала) изменить бит BODS).

**PUD:** Pull-up Disable (запрет внутренних подтягивающих резисторов). Если бит PUD установлен в 1, резисторы pull-up на всех портах I/O (PORTB, PORTC, PORTD) отключены, даже если биты DDxn и PORTxn регистров DDRx и PORTx сконфигурированы для разрешения pull-up. См. л.р. "Программирование параллельного порта контроллеров AVR на языке Assembler".

**IVCE** (Interrupt Vector Change Enable) – разрешение. При установке IVCE автоматически запрещается обработка всех прерываний. Их обработка будет вновь разрешена после изменения бита IVSEL или по истечении 4 тактов.

**IVSEL** (Interrupt Vector Select) – выбор расположения вектора прерывания. По умолчанию сброшен, таблица векторов прерываний начинается с адреса 0x0002 (0x0000 включая вектор сброса). Чтобы переназначить её в секцию загрузчика (программы инициализации) необходимо запись в этот бит 1. Для этого сначала нужно разрешить его изменение, установив бит IVCE, затем в течение 4 тактов записать новое значение в IVSEL. Изменение IVSEL не

переносит физически таблицу векторов прерываний, однако IVSEL указывает микроконтроллеру, где он должен ее искать: в секции программ или в секции загрузчика. (Если загрузчик использует прерывания, то он должен иметь соответствующие обработчики и таблицу векторов. По завершении своей работы загрузчик сбрасывает IVSEL, чтобы прерывания обслуживались обработчиками основной программы.)

## Внешние прерывания в ATmega328P

### Прерывания INT

**INT** прерывания – разновидность внешних прерываний. Вызвать такое прерывание может любое устройство, подключенное к выводам, обозначенным INT<sub>n</sub>, где *n* – номер прерывания. В микроконтроллере ATmega328P таких прерываний 2 (INT0 – PortD2, INT1 – PortD3). Прерывания могут быть вызваны, как изменением сигнала на линии, так и по установленному состоянию. Управление INT прерываниями обеспечивается регистрами EICRA, EIMSK, EIFR. За обработку прерывания отвечают векторы с адресами (при нормальном режиме работы) 0x0002 – вектор INT0 и 0x0004 – вектор INT1. INT0 – самое высокоприоритетное маскируемое прерывание.

### Регистры INT прерываний

**EICRA** (External Interrupt Control Register A) – Регистр управления внешними прерываниями хранит биты, определяющие, по какому признаку будет сформирован запрос на прерывание.

Bit	7	6	5	4	3	2	1	0
					ISC11	ISC10	ISC01	ISC00
Доступ					R/W	R/W	R/W	R/W
Начальное состояние					0	0	0	0

Биты EICRA[1:0] (ISC01:ISC00) определяют признак, по которому будет сформирован запрос на прерывание от источника INT0.

Биты EICRA[3:2] (ISC11:ISC10) определяют признак, по которому будет сформирован запрос на прерывание от источника INT1.

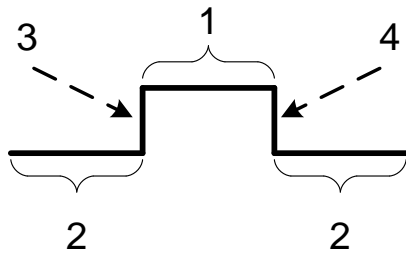
**ISC** (*Interrupt Sense control*) – управление «чувствительностью» прерывания – выбор уровня сигнала, при котором формируется запрос.

Используется 4 признака (см. таблицу 1, рисунок1):

Таблица 1 – Признаки формирования запросов на прерывание

$[ISCx1:ISCx0]^*$	Формирование запроса на прерывание
00	по низкому уровню сигнала на линии $INTx$
01	по любому изменению уровня сигнала на линии $INTx$
10	по спаду сигнала на линии $INTx$
11	по фронту уровня сигнала на линии $INTx$

\*  $x=0$  для прерывания  $int0$  и  $x=1$  для прерывания  $int1$



- 1 - высокий уровень сигнала, соответствует логической «1»
- 2 - низкий уровень сигнала, соответствует логическому «0»
- 3 – «фронт» сигнала, соответствует переходу от логического «0» к «1»
- 4 – «спад» сигнала, соответствует переходу от логической «1» к «0»

Рисунок 1 – Логические уровни

Для исключения формирования запросов на прерывания от шумов на линии значение на выводе  $INT1$  дискретизируется перед обнаружением переходов (спада или фронта). Если выбрано прерывание по изменению сигнала (коды 01 – 11 в таблице 1), импульсы продолжительностью более одного такта будут генерировать прерывание. Более короткие импульсы не гарантируют возникновение запросов на прерывания. Если выбрано прерывание низкого уровня, то низкий уровень должен удерживаться до завершения выполняемой в данный момент процессором инструкции для генерации прерывания.

**EIMSK** (*External Interrupt Mask Register*) – Регистр маскирования внешних прерываний. (в данном случае маскирование – разрешение прерывания)

Bit	7	6	5	4	3	2	1	0
							INT1	INT0
Доступ							R/W	R/W
Начальное состояние							0	0

Установка битов  $INT0$  и  $INT1$  в единицу разрешает соответствующее внешнее прерывание (при установленном в единицу бите  $I$  (глобальное разрешение прерываний) регистра состояния  $SREG$ ).

*Пример: Бит I регистра SREG установлен в «1», бит INT0 регистра EIMSK установлен в «1», биты INT[1:0] регистра EICRA принимают значение 10<sub>2</sub>. Это означает, что при изменении сигнала из 1 в 0 на выводе микроконтроллера отвечающего за внешнее прерывание INT0 будет выполнена остановка выполнения текущей задачи и запущена программа обработчика прерывания INT0, после завершения которой, процессор продолжит выполнение задачи, начиная с операции на которой остановился.*

**EIFR** (*External Interrupt Flag Register*) – Регистр флагов внешних прерываний.

Bit	7	6	5	4	3	2	1	0
							INTF1	INTF0
Доступ							R/W	R/W
Начальное состояние							0	0

Биты INTF1(*External INTerrapt Flag – флаг внешнего прерывания*) и INTF0 устанавливаются в 1 при формировании запроса на прерывание. Сброс флага осуществляется сразу после перехода к вектору обработчика прерывания. Флаг не устанавливается в 1, если биты регистра EICRA запрограммированы на формирование запроса по низкому уровню сигнала. Сбросить флаг можно вручную, записав в него «1».

### **PCINT – прерывания**

Помимо выводов PortD2 (INT0) и PortD3 (INT1), прерывания могут быть инициированы изменением на любом из информационных выводов микроконтроллера AVR ATmega328P. Однако, для всего набора выводов реализовано только 3 обработчика прерывания. Для этого выводы разбиты на группы: PCI0, PCI1, PCI2 подробно см. таблицу 3.



Таблица 3 – группы прерываний PCIn

Бит регистра	Номер вывода микроконтроллера	Назначение вывода микроконтроллера	Обозначение вывода (IDE Arduino)	PCINTx
<b>Регистр PCMSK0</b>				
0	14	PB0/CLKO/ICP1	D8	PCINT0
1	15	PB1/OC1A/PWM	D9	PCINT1
2	16	PB2/OC1B/PWM	D10	PCINT2
3	17	PB3/OC2A/PWM/MOSI	D11	PCINT3
4	18	PB4/MISO	D12	PCINT4
5	19	PB5/SCK	D13	PCINT5
6	9	PB6/OSC1/XTAL1	-	PCINT6
7	10	PB7/OSC2/XTAL2	-	PCINT7
<b>Регистр PCMSK1</b>				
0	23	PC0/ADC0	A0	PCINT8
1	24	PC1/ADC1	A1	PCINT9
2	25	PC2/ADC2	A2	PCINT10
3	26	PC3/ADC3	A3	PCINT11
4	27	PC4/ADC4/SDA	A4	PCINT12
5	28	PC5/ADC5/SCL	A5	PCINT13
6	1	PC6/RESET	-	PCINT14
7	-	-	-	-
<b>Регистр PCMSK2</b>				
0	2	PD0/RXD	D0	PCINT16
1	3	PD1/TXD	D1	PCINT17
2	4	PD2/INT0	D2	PCINT18
3	5	PD3/INT1/PWM/OC2B	D3	PCINT19
4	6	PD4/XCK/T0	D4	PCINT20
5	11	PD5/T1/PWM/OC0B	D5	PCINT21
6	12	PD6/PWM/OC0A	D6	PCINT22
7	13	PD7/AIN1	D7	PCINT23

*!Прерывания PCINT срабатывают и по фронту и по спаду (по изменению сигнала). Их нельзя настроить на срабатывание по уровню, а определение, того как изменился сигнал, фронт или спад, можно выполнить только программно в обработчике прерывания, при сравнении текущего значения вывода и заранее сохраненного предыдущего.*

## Регистры PCINT прерываний

**PCICR** (*Pin Change Interrupt Control Register*) – регистр управления прерываниями по изменению состояния входа микроконтроллера.

Bit	7	6	5	4	3	2	1	0
						PCIE2	PCIE1	PCIE0
Доступ						R/W	R/W	R/W
Начальное Состояние						0	0	0

PCIE0 – разрешение прерываний от группы контактов PCIO

PCIE1 – разрешение прерываний от группы контактов PCII

PCIE2 – разрешение прерываний от группы контактов PCI2

Для всех бит регистра: логическая «1» - прерывание разрешено, «0» - запрещено;

**PCIFR** (*Pin Change Interrupt Flag Register*) – регистр флагов прерываний по изменению состояния вывода

Bit	7	6	5	4	3	2	1	0
						PCIF2	PCIF1	PCIF0
Доступ						R/W	R/W	R/W
Начальное Состояние						0	0	0

PCIF0 – флаг изменения состояния входа контроллера из группы 0.

PCIF1 – флаг изменения состояния входа контроллера из группы 1.

PCIF2 – флаг изменения состояния входа контроллера из группы 2.

Флаг устанавливается в 1 при получении запроса о соответствующем прерывании, и сбрасывается при переходе к соответствующему вектору прерывания.

**PCMSK0** (*Pin Change Mask Register*) – регистр маскирования выводов

Bit	7	6	5	4	3	2	1	0
	PCINT 7	PCINT 6	PCINT 5	PCINT 4	PCINT 3	PCINT 2	PCINT 1	PCINT 0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное Состояние	0	0	0	0	0	0	0	0

PCINTn: Pin Change Enable Mask – включение прерывания по изменению состояния на конкретном выводе микроконтроллера. Каждый бит соответствует

своему выводу (см. таблицу 2). Логическая «1» - прерывание включено, логический «0» прерывание выключено.

### PCMSK1

Bit	7	6	5	4	3	2	1	0
		PCINT 14	PCINT 13	PCINT 12	PCINT 11	PCINT 10	PCINT 9	PCINT 8
Доступ		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное Состояние		0	0	0	0	0	0	0

Аналогично регистру PCMSK0, для группы контактов PC11.

### PCMSK2

Bit	7	6	5	4	3	2	1	0
	PCINT 23	PCINT 22	PCINT 21	PCINT 20	PCINT 19	PCINT 18	PCINT 17	PCINT 16
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное Состояние	0	0	0	0	0	0	0	0

Аналогично регистру PCMSK0, PCMSK1, для группы контактов PC12.

Пример: Бит 1 регистра SREG, бит PCIE1 регистра PCICR, бит PCINT12 регистра PCMSK1 установлены в «1». Это означает, что при любом изменении сигнала на выводе микроконтроллера №24 (PortC4) отвечающего за прерывание PCINT12 будет выполнена остановка выполнения текущей задачи и запущена программа обработчика прерывания PCINT1 (отвечающая за всю группу выводов), после завершения которой, процессор продолжит выполнение задачи, начиная с операции на которой остановился.

Примечание: для обработки 23 прерываний используется всего 3 вектора, т.е. 3 процедуры. Если требуется, чтобы для каждого из используемых прерываний выполнялась своя процедура обработки, то необходимо хранить состояние выводов до изменения сигнала, а после изменения анализировать, какой вывод вызвал прерывание и далее выполнять требуемые действия.

### Практическая часть

Пример 1. Рассмотрим пример программы, выполняющей настройку микроконтроллера на обработку внешних прерываний Int1 по низкому уровню

сигнала, вывод увеличивающегося на 1 с каждым циклом основной программы кода, и сброс этого кода к значению 0x00 при возникновении прерывания Int1.

1	<code>.include "m328Pdef.inc"</code>	14	<code>out EIMSK, r16</code>
2	<code>.org 0</code>	15	<code>cbi ddrd, 3</code>
3	<code>jmp Reset</code>	16	<code>sbi portd, 3</code>
4	<code>.org 0x004</code>	17	<code>ldi r16, 0xff</code>
5	<code>jmp Int_1</code>	18	<code>out ddrc, r16</code>
6	<code>Reset:</code>	19	<code>sei</code>
7	<code>ldi r16, High(RAMEND)</code>	20	<code>main: inc r17</code>
8	<code>out sph, r16</code>	21	<code>out portc, r17</code>
9	<code>ldi r16, Low(RAMEND)</code>	22	<code>jmp main</code>
10	<code>out spl, r16</code>	23	<code>Int_1:</code>
11	<code>ldi r16, 0x00</code>	24	<code>ldi r17, 0x7E</code>
12	<code>sts EICRA, r16</code>	25	<code>reti</code>
13	<code>ldi r16, 0x02</code>		

В первой строке программы выполняется подключение библиотеки микроконтроллера ATmega328P. В ней объявлены параметры контроллера необходимые компилятору, а также названия регистров ввода/вывода и их адреса. В строках 2-7 инициализируется таблица векторов прерываний. Директива `.org` указывает адрес сегмента кода в памяти команд. Например, строки 2-3 определяют, что по адресу 0x000 будет записана команда `jmp` (безусловный переход) к метке `reset`. Таким образом определяется вектор прерывания `reset` (немаскируемое прерывание, выполняется при включении контроллера или перезагрузке. Аналогично инициализируются вектор `Int1` (в соответствии с таблицей векторов прерываний контроллера (таблица 1) `Int1` – имеет адрес 0x004).

*Примечание: имена меток пользователь может выбрать самостоятельно.*

Начиная с метки `Reset` (строка 6) и до метки `Main` (строка 20) выполняется инициализации периферии микроконтроллера:

- 7-10 строки – инициализация указателя стека;
- 11-12 строки – запись значения 0x00 в регистр управления внешними прерываниями `EICRA`. Данная константа включает режим, при котором прерывание выполняется по низкому уровню сигнала на линиях (`INT0`, `INT1`).
- 13-14 строки – запись значения 0x02 в регистр маскирования (разрешения) прерываний `EIMSK`. Данная константа разрешает прерывания на линии `INT1`.

– 15-16 строки – подготовка битов порта D для работы с прерыванием INT1, порт настраивается на вывод данных

– 17-18 строки – инициализация порта C для вывода параллельного кода (начальное значение 0xff).

– 19 строка – глобальное разрешение прерываний запись 1 в бит I регистра SREG.

Строки 20-22 – основная программа. Программа выполняет инкремент регистра r17, и выводит его значение в порт C.

Строки 23-25 – Код прерывания INT1 – обнуляет регистр r17.

\*Примечание: изначально следует установить единицы во 2 (INT0) и 3(INT1) биты PIND. Если этого не сделать, то будет постоянно работать подпрограмма прерывания\*

Выполните компиляцию программы и запустите её в отладчике (Simulator). Проанализируйте выполнение программы. Для анализа регистров системы прерываний откройте Debug/Windows/IO, выберите строку External Interrupts (EXTINT). Прерывания в программе настроены по низкому уровню на линии INT1 (по логическому «0» на входе контроллера PORTD3 - (PD3)). Поэтому перед анализом работы следует установить единицу третий бит регистра PIND. Если этого не сделать, то будет постоянно работать подпрограмма прерывания. Чтобы вызвать прерывание нужно установить в регистре PIND3 логический «0».

*Примечание: При работе в IDE Atmel Studio 7.0 необходимо разрешить прерывания в режиме отладки. Для этого необходимо в главном меню выбрать **Tools/Options.../Tools/Tool settings/** в поле **Mask interrupts while stepping** установить значение **false**.*

Подключите устройства к учебному стенду согласно рисунку 2.

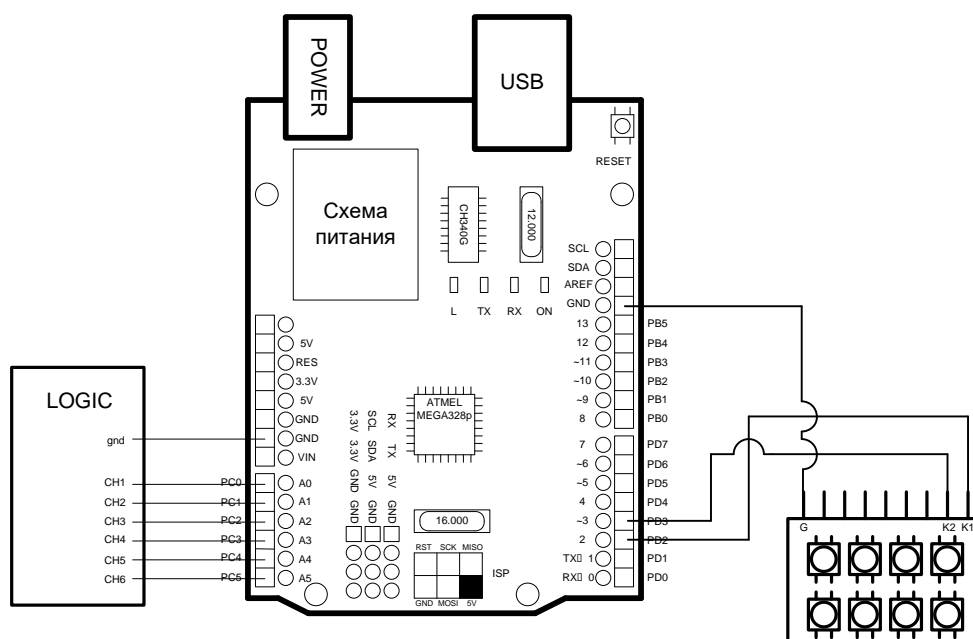


Рисунок 2 – подключение устройств для примера 1 и задания 1

Загрузите программу в контроллер и проанализируйте её работу. Для анализа изменения кода откройте программу Saleae Logic. Чтобы вызвать прерывание необходимо нажать на кнопочном модуле кнопку K1. Для того, чтобы увидеть момент выполнения процедуры обработчика прерывания необходимо в программе Saleae Logic установить время захвата 2 секунды. Нажать на кнопку Start и сразу нажать на кнопку K1 кнопочного модуля.

В отчёте зафиксируйте трассу программы (по анализу её работы с отладчиком), а также момент срабатывания прерывания на снимке экрана при работе Saleae Logic.

Задание 1 – Измените код из примера таким образом, чтобы система реагировала на прерывание Int0 переключением между двумя светодиодами (при нажатии на кнопку один диод загорается, второй гаснет, при следующем нажатии наоборот). Для работы используйте светодиоды подключенные к линиям PD0, PD1 (диоды Rx, Tx на учебной плате). В чем будет выражена разница работы микроконтроллера при настройке прерывания по уровню сигнала и по фронту? Подготовьте программы для работы в обоих режимах, проанализируйте их работу в отладчике, поочередно загрузите их в контроллер и проанализируйте работу.

Задание 2 – Измените программу таким образом, чтобы система реагировала на прерывания Int0 и Int1 по фронту импульса, изменяя состояние

светодиодов PD0 (в обработке Int0) и PD1(в обработке Int1) на противоположное. Запустите программу в отладчике. Вызовите прерывания int0 и int1 одновременно. Проанализируйте приоритет прерываний. Загрузите программу в контроллер и проанализируйте её работу.

### Прерывания PCINT

Пример 2. Рассмотрим пример программы обеспечивающей инициализацию прерываний PCINT0 и PCINT2, а также анализировать от какого бита получено прерывание.

1	<code>.include "m328Pdef.inc"</code>	17	<code>sts PCMSK2, r16</code>
2	<code>.org 0</code>	18	<code>sei</code>
3	<code>jmp Reset</code>	19	<code>m: inc r17</code>
4	<code>.org 0x006</code>	20	<code>jmp m</code>
5	<code>jmp mPCINT0</code>	21	<code>mPCINT0:</code>
6	<code>.org 0x00A</code>	22	<code>mov r18,r20</code>
7	<code>jmp mPCINT2</code>	23	<code>mov r22,r20</code>
8	<code>Reset:</code>	24	<code>in r20,PINB</code>
9	<code>ldi r16,low(RAMEND)</code>	25	<code>eor r22,r20</code>
10	<code>out spl, r16</code>	26	<code>reti</code>
11	<code>ldi r16,high(RAMEND)</code>	27	<code>mPCINT2:</code>
12	<code>out sph, r16</code>	28	<code>mov r19,r21</code>
13	<code>ldi r16, 0x07</code>	29	<code>mov r23,r21</code>
14	<code>sts PCICR, r16</code>	30	<code>in r21,PIND</code>
15	<code>ldi r16, 0xFF</code>	31	<code>eor r23,r21</code>
16	<code>sts PCMSK0, r16</code>	32	<code>reti</code>

В первой строке программы выполняется подключение библиотеки микроконтроллера ATmega328P. Далее, в строках 2-7 выполняется инициализация векторов прерываний, затем в строках 9-12 – инициализация стека.

В строках 13-18 – инициализация системы прерываний:

– 13-14 Загрузка значения "0x07" в регистр управления прерываниями по изменению состояния входа микроконтроллера PCICR. Разрешает прерывания от всех трёх групп контактов (PCI0, PCI1, PCI2);

– 15-17 строки – PCMSK0, PCMSK2 - регистры маскирования (разрешения) прерываний по изменению уровня на входе. Для обращения служит команда «sts». При загрузке значения "0xFF" разрешаются прерывания от всех выводов микроконтроллера;

– 18 строка – глобальное разрешение прерываний запись 1 в бит I регистра SREG.

В строках 19-20 – основная программа, выполняет инкремент регистра r17.

Строки 21-26 – код обработчика прерывания PCINT0:

– выполняет сравнение сигналов на входе порта В с сохраненными значениями сигнала до возникновения текущего прерывания. Таким образом можно установить какой именно вывод порта сформировал прерывание.

27-32 – код обработчика прерывания PCINT2 (аналогично с PCINT0, но для порта D)

Выполните компиляцию программы и запустите её в отладчике (Simulator). Проанализируйте выполнение программы.

Задание 3. Измените код программы в примере, оставив разрешенными прерывания только от указанных в таблице выводов контроллера.

Вариант	1	2	3	4	5	6	7	8	9	10
Задание	PB0 PD2	PB2 PD1	PB3 PD2	PB0 PD6	PB4 PD4	PB6 PD0	PB3 PD1	PB0 PD0	PB5 PD3	PB1 PD6

### Контрольные вопросы

1. Объясните понятие прерывание.
2. Что называют вектором прерывания? Что называют таблицей векторов прерываний.
3. Сколько векторов прерываний в таблице контроллера Atmega328P?
4. Опишите назначение регистров, отвечающих за прерывания INT0, INT1.
5. Опишите назначение регистров, отвечающих за прерывания PCINT0, PCINT2, PCINT3.
6. Что необходимо сделать, чтобы процессор прерывался по фронту сигнала на линии INT1?
7. Что необходимо сделать, чтобы процессор прерывался по фронту сигнала на линии PCINT6?
8. Каким образом, можно получить отдельные обработчики для прерываний группы PCINT0.