Практическая работа Тема 5.

Обработка прерываний. Внешние прерывания

Цель: изучить процессы прерываний и их виды, получить навыки работы с системой прерываний на примере микроконтроллера AVR ATmega328P.

Используемое оборудование:

Плата с МК Atmel AVR ATMega 328P	1 шт.
Блок питания 5В /3А	1 шт.
Программатор USBISP	1 шт.
Логический анализатор	1 шт.
Кнопочный модуль	1 шт.

Используемое ПО: Интегрированная среда разработки Atmel Studio 7.0 (или AVR Studio 4.19). Программа для загрузки программного кода в микроконтроллер AVRDUDEPROG

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Прерывания

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. Ее 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 размешены векторы, называется **таблицей векторов прерываний**.

Прерывания в ATmega328P

Микроконтроллер ATmega328P имеет 26 обработчиков прерываний (табл.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), `SLI` – 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.

Табл. 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 A таймера/счетчика T2	TIMER2_COMPA_vect
9	0x0010	TIMER2_COMPB	Совпадение B 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 A таймера/счетчика T1	TIMER1_COMPA_vect
13	0x0018	TIMER1_COMPB	Совпадение B таймера/счетчика T1	TIMER1_COMPB_vect
14	0x001A	TIMER1_OVF	Переполнение таймера/счетчика T1	TIMER1_OVF_vect
15	0x001C	TIMER0_COMPA	Совпадение A таймера/счетчика T0	TIMER0_COMPA_vect
16	0x001E	TIMER0_COMPB	Совпадение B 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 (табл. 2).

Табл. 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 picoPower 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 EEPROOM от стирания. Подробнее см.описание fuse битов ATmega328P.*

BODS (BOD Sleep) – запрет детектора низкого 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 μs, 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 ее 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 изменения вектора прерывания. 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 (или с адреса 0x0000, включая вектор сброса). Чтобы переназначить ее в секцию загрузчика (программы инициализации), необходимо записать 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 INTn, 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 IECRA[1:0] (ISC01:ISC00) определяют признак, по которому будет сформирован запрос на прерывание от источника INT0.

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

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

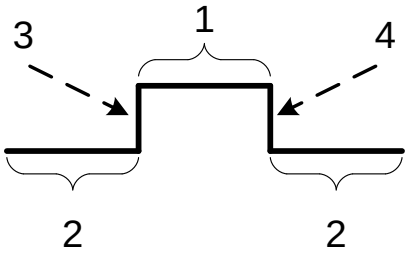
Используется 4 признака (табл. 3, рис. 5.1).

Табл. 3. 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 уровню 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»

Рис. 5.1. 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 – это 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_2. 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 в табл. 4).

Табл. 4. Группы прерываний PCInt

Бит регистра	**Номер вывода микроконтроллера**	**Назначение вывода микроконтроллера**	**Обозначение вывода (IDE Arduino)**	**PCINTx**
Регистр PCMSK0				
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
Регистр PCMSK1				
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	-	-	-	-
Регистр PCMSK2				
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

!Примечание: 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 PCI0;
PCIE1 – разрешение прерываний от группы контактов PCI1;
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. табл. 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 PCI1.

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 PCI2.

Пример: бит I 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.

Сводная таблица описанных регистров представлена ниже.

Адрес в ОЗУ	Адрес в I/O Reg	Имя регистра	Назначение битов регистра [7:0]							
			7	6	5	4	3	2	1	0
0x3C	0x1C	EIFR							INTF1	INTF0
0x3D	0x1D	EIMSK							INT1	INT0
0x69		EICRA					ISC11	ISC10	ISC01	ISC00
0x3B	0x1B	PCIFR						PCIF2	PCIF1	PCIF0
0x68		PCICR						PCIE2	PCIE1	PCIE0
0x6B		PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6C		PCMSK1		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D		PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x55	0x35	MCUCR		BODS	BODSE	PUD			IVSEL	IVCE

Для работы с регистрами EIFR, EIMSK, PCIFR, MCUCR используются команды `IN/OUT`, они расположены в адресном пространстве регистров ввода/вывода.

Для работы с регистрами EICRA, PCICR, PCMSK0, PCMSK0, PCMSK0 используются команды `sts, lds`, они расположены в адресном пространстве дополнительных регистров ввода/вывода.

Особенности программирования прерываний на микроконтроллерах AVR с использованием стандартных библиотек на языке Assembler

Чтобы запрограммировать микроконтроллер на обработку прерываний на Assembler, необходимо выполнить следующие действия:

- связать адрес, зарезервированный для прерывания, с подпрограммой, выполняющей его обработку;
- описать подпрограмму – обработчик прерывания;
- разрешить обработку интересующего прерывания;
- установить бит глобального разрешения прерываний в регистре SREG.

Ниже представлен фрагмент программы на языке Assembler, демонстрирующий работу с прерыванием INT1.

```
.include"m328Pdef.inc"      ;подключение библиотеки с идентификаторами (именами) регистров
.org 0                      ;директива, записывающая следующую команду по адресу 0x000 ПЗУ
                            ;к этому адресу контроллер переходит автоматически при включении и
                            ;перезагрузке контроллера
        jmp Reset           ;команда безусловного перехода к метке Reset
.org 0x004                  ;директива, записывающая следующую команду по адресу 0x004 ПЗУ
                            ;по этому адресу контроллер переходит автоматически при возникновении
                            ;соответствующего прерывания
        jmp Int_1           ;команда безусловного перехода к метке Int_1
Reset:                      ;метка Reset
        ldi r16, 0xC0       ;настройка прерывания INT1 запись константы 0xC0 в r16
        sts EICRA, r16      ;запись значения 0xC0 в регистр EICRA (прерывание INT1 по «фронту»)
                            ;с регистром EICRA используется команда sts
        ldi r16, 0x02       ;настройка прерывания INT1 запись константы 0x02 в r16
        out EIMSK, r16      ;запись значения 0x02 в регистр EIMSK (разрешение прерывания INT1)
                            ;с регистром EIMSK используется команда out
        (команды)           ;прочие действия (инициализация и настройка)
        sei                 ;глобальное разрешение прерываний
main:                       ;метка основной программы
        (команды)           ;действия основной программы
        jmp main            ;возврат к метке основной программы (бесконечный цикл)
Int_1:                      ;метка основной программы
        (команды)           ;действия подпрограммы обработчика прерываний
        reti                ;выход из прерывание, продолжение выполнения основной программы
```

Особенности программирования прерываний на микроконтроллерах AVR с использованием стандартных библиотек на языке С

Чтобы запрограммировать микроконтроллер на обработку прерываний на С необходимо выполнить следующие действия:

- подключить библиотеку для работы с прерываниями;
- описать подпрограмму – обработчик прерывания, подпрограмма должна быть описана следующим образом `ISR(ИмяВектора)`. ISR – имя для всех подпрограмм обработчиков прерываний. Параметрами подпрограммы является имя вектора из таблицы 1;
- разрешить все прерывания командой `sei();`
- разрешить обработку интересующего прерывания.

Ниже представлен фрагмент программы на языке С, демонстрирующий работу с прерываниями PCINT2.

```
#include <avr/io.h>                 //подключение библиотеки с идентификаторами (именами) регистров
#include <avr/interrupt.h>          //подключение библиотеки для работы с прерываниями
ISR(PCINT2_vect)                    //объявление подпрограммы обработчика прерываний PCINT2
{                                   //
   (действия)                       //действия подпрограммы обработчика прерываний
}                                   //
int main(void)                      //начало основной программы
{                                   //
   PCMSK2 |= 0xAA;                  //разрешение PCINT прерываний от выводов PD1, PD3, PD5, PD7
   PCICR  |= (1<<PCIE2);            //разрешение прерываний PCINT2
   (действия)                       //прочие действия (инициализация и настройка)
   sei();                           //глобальное разрешение прерываний
     while (1)                      //бесконечный цикл, основная программа
     {                              //
         (действия)                 //действия основной программы
 }}                                 //
```

ПРАКТИЧЕСКАЯ ЧАСТЬ

Прерывания INT

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

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 (табл. 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 23-26 – код прерывания INT1 – записывает значение 0x00 в регистр r17 и выводит это значение в порт C.

!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 ее 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 в 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 slipping*** *установить значение* ***false***.

Подключите устройства к учебному стенду согласно рисунку 5.2.

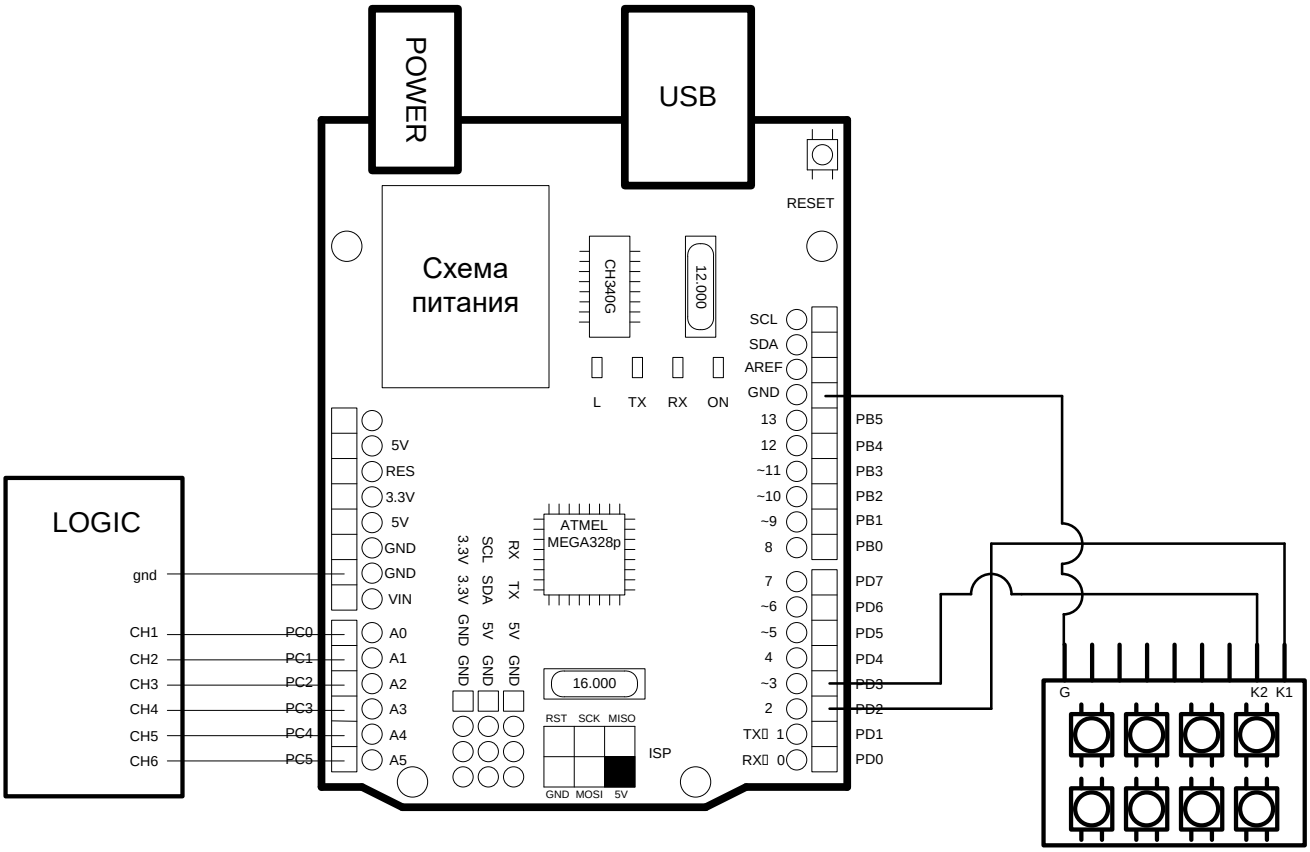


Рис.5.2. Подключение устройств для примера 1 и задания 1

Загрузите программу в контроллер и проанализируйте ее 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, затем нажать на кнопку Start и сразу нажать на кнопку K1 кнопочного модуля.

В отчете зафиксируйте трассу программы (по анализу ее работы в отладчике), а также момент срабатывания прерывания на снимке экрана при работе Saleae Logic.

Рассмотрим реализацию описанной выше программы на языке С.

```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  int x;
5  ISR(INT1_vect)
6  {
7        PORTC = 0x00;
8  }
9  int main(void)
10 {
11       x=0;
12       EICRA = 0x00;
13       EIMSK = 0x02;
14       DDRD = 0x00;
15       PORTD = 0x08;
16       DDRC = 0xFF;
17       sei();
18    while (1)
19    {
20             PORTC = x;
21             x++;
22    }
23 }
```

Первые две строки – подключение библиотек. Для использования прерываний необходимо подключение файла <avr/interrupt.h>. В нем описаны функции для работы с прерываниями и константы векторов прерываний. В заголовочном файле <avr/io.h> описаны основные константы для работы с микроконтроллером и его периферией.

Для описания процедуры обработчика прерывания необходимо инициализировать процедуру ISR (int vect), где vect – вектор прерывания, для которого описывается обработчик. Имена векторов прерываний можно посмотреть в файле <avr/interrupt.h>, также они перечислены в таблице 3. Для глобального разрешения прерывания необходимо в основной программе выполнить функцию `sei()`. Для глобального запрета прерываний используется функция `cli()`.

Задание 1. Измените код из примера таким образом, чтобы система реагировала на прерывание INT0 переключением между двумя светодиодами (при нажатии на кнопку один диод зажигается, второй гаснет, при следующем нажатии наоборот). Для работы используйте светодиоды RX – PD0, TX – PD1. Подготовьте программы для работы в режиме срабатывания прерывания по уровню и по фронту. Проанализируйте их работу в отладчике и отметьте разницу в отчете. Переключите кнопку из PD3 в PD2 (используйте рис. 2). Поочередно загрузите программы 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 AT1 (в обработчике *INT0*) и AT2 (в обработчике INT1) на противоположное. Запустите программу в отладчике. Вызовите прерывания INT0 и INT1) одновременно. Проанализируйте приоритет прерываний. Загрузите программу в контроллер и проанализируйте ее работу.

Прерывания PCINT

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

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

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 строки – загрузка значения "0x07" в регистр управления прерываниями по изменению состояния входа микроконтроллера PCICR. Разрешает прерывания от всех трех 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 B с сохраненными значениями сигнала до возникновения текущего прерывания. Таким образом можно установить какой именно вывод порта сформировал прерывание.

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

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

Ниже представлена реализация описанной программы на языке C. Проанализируйте разницу в коде. Выполните сборку программы и проанализируйте ее работу в отладчике.

```c
#include <avr/io.h>
#include <avr/interrupt.h>
char x,y,temp, count;
ISR(PCINT0_vect)
{
        x = PINB;
        y = x ^ temp;
        temp = x;
}
ISR(PCINT2_vect)
{
        x = PIND;
        y = x ^ temp;
        temp = x;
}
int main(void)
{
        x=0;
        PCICR0 = 0x07;
        PCMSK0 = 0xFF;
        PCMSK2 = 0xFF;
        sei();
    while (1)
    {
                PORTC = count;
                count++;
    }
}
```

Задание 3. Измените код программы из примера 2 (на языке С), оставив разрешенными прерывания только от указанных в таблице выводов контроллера. Обработчики прерываний измените таким образом, чтобы на вывод PORTB5 (светодиод L) выводилось текущее значение старшего бита счетчика count и сохранялось до следующего прерывания.

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

Загрузите программу в контроллер. Проанализируйте результаты работы. Зафиксируйте реакцию системы в отчете.

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.