Лабораторная работа 6. Последовательный интерфейс UART

Цель работы: изучить принцип работы последовательного интерфейса UART, изучить порядок настройки и работы с интерфейсом на микроконтроллере Atmega328P.

Теоретическая часть

UART (Universal Asynchroanous Receiver/Transmitter) – последовательный асинхронный интерфейс передачи данных. Для передачи используются два проводника RX (Receiver - приемник) и TX (Transmitter – передатчик), которые отвечают за передачу информации в противоположных направлениях. Обычно в передаче также используется линия GND. Схема соединения двух устройств по UART представлена на рисунке 1.

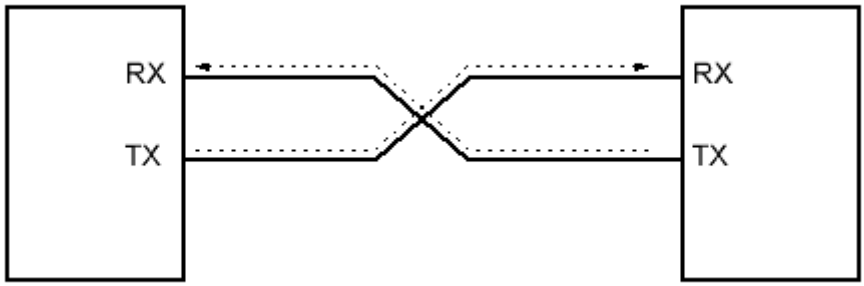


Рисунок 1 – Подключение двух устройств

Передача данных в UART выполняется следующим образом. В момент, когда линия свободна, на ней установлена 1. Первый бит пакета – стартовый, равен 0. Далее передаётся информационная часть, обычно вначале идут младшие биты. Информационная часть может содержать от 5 до 9 бит (в некоторых случаях даже до 12). После, если потребуется, передаётся бит чётности для контроля целостности данных (Если признак чётности вычисляемый приёмником не совпадет с передаваемым битом паритета, приёмник будет понимать, что данные переданы не корректно). Передача завершается стоповыми битами, которых может быть один или два. Протокол передачи представлен на рисунке 2.

Примечание: На рисунке в скобках обозначены необязательные биты, их использование указывается в настройках работы интерфейса

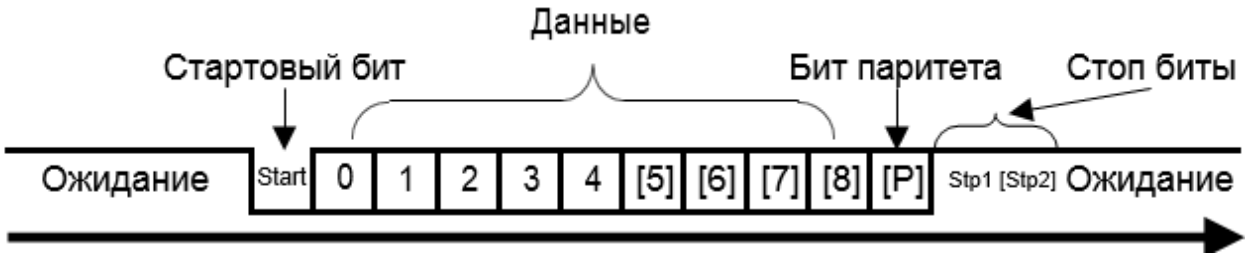


Рисунок 2 – Передача данных в UART

Скорость передачи определяется количеством бит, передаваемых в секунду (Количество бит в секунду также называют бодами). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бит/с.

Для управления потоком данных UART используется программный или аппаратный метод.

В случае **программного метода**, информация о готовности устройства принимать данные или о необходимости остановить передачу передаётся по тем же каналам, что и данные. Принимающая сторона программно разделяет данные и управляющие сигналы в соответствии с принятым протоколом.

Аппаратное управление может использоваться некоторыми медленными устройствами или устройствами с простой схемной реализацией, однако оно потребует двух дополнительных линий для подключения устройства. При использовании аппаратного метода интерфейс UART предусматривает возможность использования дополнительных сигналов **CTS** и **RTS.** Аппаратное управление изображено на рисунке 3.

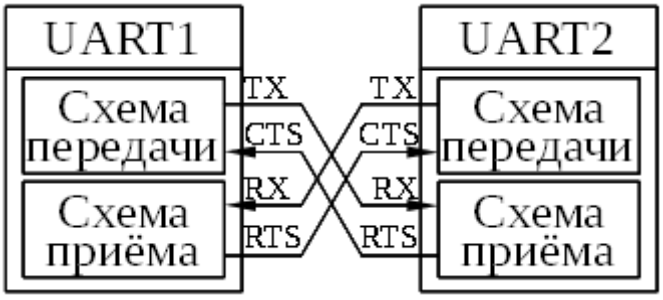


Рисунок 3 – Аппаратное управление

CTS (Clear To Send) – устанавливает принимающая сторона при готовности к приёму данных, активный уровень 0.

RTS (Request To Send) – запрос на отправку данных от передающей стороны для управления потоком данных.

Перед отправкой данных передатчик устанавливает сигнал **RTS** в 0. Если на **CTS** будет также установлен низкий уровень, передача происходит, иначе - нет. Если сигнал **CTS** будет установлен во время передачи информации, текущая передача всё равно будет завершена перед остановкой.

UART в AVR Atmel 328P

Практически для всех методов отладки программы необходимо физическое соединение с компьютером. Во многих платах Arduino имеется USB-разъём и специальная микросхема, которая преобразует интерфейс UART в USB. Построена такая микросхема на базе чипа CH340 или ATmega, или др. Шина USB этих чипов подключена к порту USB, а шина UART к аппаратным выводам TX и RX контроллера платы Arduino.

Плата Arduino UNO имеет один порт UART, сигналы которого подключены к выводам 0 (сигнал RX) и 1 (сигнал TX). Через эти выводы можно подключить к плате другое устройство имеющее интерфейс UART.

Портов может быть много, поэтому они обозначаются как COM1, COM2 и т. д. по порядку обнаружения драйверов соответствующих устройств.

Регистры UART

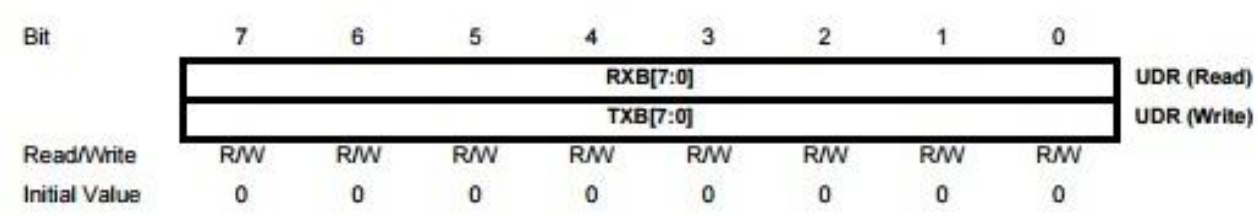
UDR0 (UART Data Register) – представляет собой два разных регистра, имеющих один адрес. Регистр буфера передачи данных (TXB) служит для записи данных, записанных UD в регистр UDR0. Для приема данных используется буфер данных приема (RXB).

Поскольку передача данных идет довольно медленно, то помещать данные в регистр **UDR** нужно только после окончания передачи предыдущего байта. О том, что **UDR** пуст и готов к приему нового байта сигнализирует бит **UDRE**, он же вызовет аппаратное прерывание по

опустошению буфера.

В модулях USART буфер приемника является двухуровневым (FIFO-буфер), изменение состояния которого происходит при любом обращении к регистру UDR. В связи с этим не следует использовать регистр UDR в качестве операндов команд типа «чтение/модификация/запись» (SBI и CBI). Кроме того, следует быть очень аккуратными при использовании команд проверки SBIC и SBIS, поскольку они также изменяют состояние буфера приемника.

Примечание: Для 5-, 6- или 7-битных пакетов старшая часть слова будет игнорироваться передатчиком и устанавливаться в ноль в получателе.



UCSR0A, UCSR0B, UCSR0C (UART Control Status Register) являются регистрами управления UART’ом.

UCSR0A – Регистр состояния

Бит 7 RXC0 (Receive Complete) — флаг окончания приема данных. 1 – есть непрочитанные данные. 0 – регистр опустошен.

Бит 6 TXC0 (Transmit Complete) — флаг окончания передачи данных. 1 – передача завершена и в UDR0 не было загружено нового значения.

Бит 5 UDRE0 (Data Register Empty) — флаг, означающий готовность регистра UDR получать новые данные. Если бит равен 1, то регистр UDR пуст и готов к приему новых данных.

Бит 4 FE0 (Frame Error) — флаг ошибки кадрирования (фрейма). При обнаружении ошибки кадрирования (первый стоп-бит равен 0) устанавливается в 1, сбрасывается в 0 - при приеме стоп-бита равного 1.

Бит 3 DOR0 (Data OverRun) — флаг переполнения регистра данных. 1 если в момент обнаружения нового старт-бита в сдвиговом регистре

находится последнее принятое слово, а буфер приемника полон.

Бит 2 UPE0 (Parity Error) — флаг ошибки четности. 1 при ошибке четности.

Бит 1 U2X0 — удвоение скорости обмена, если бит равен 1 *(только в асинхронном режиме. в синхронном следует установить этот бит в 0).*

Бит 0 MPCM0 — мультипроцессорный режим коммуникации. Если установлен в 1, режим включен.

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

UCSR0B – регистр управления прерываниями

Бит 7 RXCIE0 (RX Complete Interrupt Enable) - разрешение прерывания при завершении **приема** если установлен в 1.

Бит 6 TXCIE0 (TX Complete Interrupt Enable) - разрешение прерывания при завершении **передачи** если установлен в 1.

Бит 5 UDRIE0 (USART Data Register Empty Interrupt Enable)- разрешение прерывания при очистке регистра данных если установлен в 1.

Бит 4 RXEN0 (Receiver Enable)- разрешение **приема** данных, если установлен в 1.

Бит 3 TXEN0 (Transmitter Enable) - разрешение **передачи** если установлен в 1.

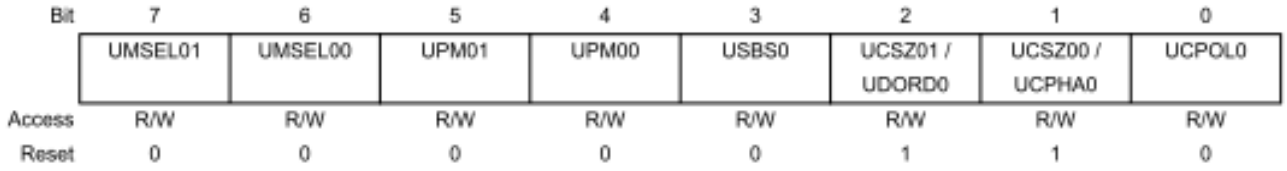
Бит 2 UCSZ02 (Character Size) - формат посылки данных (используется совместно с битами UCSZ01 и UCSZ00 регистра **UCSR0C**).

Бит 1 RXB80 (Receive Data Bit 8) - 8-й разряд **принимаемых** данных при использовании 9-разрядных слов.

Бит 0 TXB80 (Transmit Data Bit 8) - 8-й разряд **передаваемых** данных при использовании 9-разрядных слов.

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

UCSR0C – многофункциональный регистр. Отвечает за режим работы UART, систему управления и длину передаваемых пакетов.



Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Таблица 1 – Режим работы UART

UMSEL01	UMSEL00	Режим работы
0	0	Асинхронный режим
0	1	Синхронный режим
1	0	Резерв
1	1	Ведущий SPI

Таблица 2 – Система контроля и формирование четности в UART

UMP01	UMP00	Режим работы системы контроля и формирования четности USART0
0	0	Выключена система контроля
0	1	Резерв
1	0	Проверка четности
1	1	Проверка нечетности

Бит 3 USBS0 - устанавливает количество стоп битов (1 стоп-бит если сброшен в 0. 2 стоп-бита если установлен в 1).

Таблица 3 - Количество бит данных в пакете

UCSR0B	UCSR0C		Количество бит данных в пакете
UCSZ02	UCSZ01	UCSZ00	
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	0	0	Резерв
1	0	1	Резерв
1	1	0	Резерв
1	1	1	9 бит

Бит 0 UCPOL0 устанавливает полярность тактовых сигналов: 0 - передача по спаду, прием по нарастанию; 1 - передача по нарастанию, прием по спаду (для режима USART).

Регистр UBRR0

Является 12-разрядным регистром, состоящий из двух частей: UBBR0H (UART Baud Rate Register Hight), где хранятся 4 наиболее значимых бита, и UBBR0L (UART Baud Rate Register Low), где хранятся 8 младших. Записывается определенное значение в зависимости от тактовой частоты и скорости передачи. Значение вычисляется по формуле:

$\text{BAUD} = \frac{\text{CLK}}{16*(\text{UBRR0}+1)}$, где CLK - тактовая частота микроконтроллера, а BAUD – требуемая частота в бодах, UBRR0-значение регистра.

Пример: предположим, что нам требуется установить скорость передачи, равную 2400 бод. Воспользуемся формулой:

$$\text{BAUD} = \frac{\text{CLK}}{16*(\text{UBRR0}+1)} \quad (1)$$

Выразим из (1) UBRR0. Получаем формулу (2).

$$\text{UBRR0} = \frac{\text{CLK}}{16*\text{BAUD}} - 1 \qquad (2)$$

*BAUD = 2400 бод, CLK = 16 * 10^6 (Частота процессора 16 МГц). Подставляем эти значения в 3 формулу. Получаем UBRR0 = $415_{<10>}$.*

Практическая часть

Рассмотрим пример отправки определённого символа при нажатии на кнопку.

```
1  .include "m328Pdef.inc"
2  .org 0
3        jmp Reset
4  .org 0x002
5        jmp knopka
6  Reset:
7        ldi r16,low(RAMEND)
8        out spl,r16
9        ldi r16,high(RAMEND)
10       out sph,r16
11       ldi r16, 0x03
12       sts EICRA, r16
13       ldi r16, 0x01
14       out EIMSK, r16
15       cbi ddrd, 2
16       cbi ddrd, 3
17       ldi r16, 0xff
18       out ddrc, r16
19 .equ CLK=16000000
20 .equ BAUD=9600
```

```
21  .equ UBRR0_value = (CLK/(BAUD*16)) - 1
22        ldi r16, high(UBRR0_value)
23        sts UBRR0H, r16
24        ldi r16, low(UBRR0_value)
25        sts UBRR0L, r16
26        ldi r16,(1<<TXEN0)
27        sts UCSR0B,R16
28        ldi r16,(1<< UCSZ00)|(1<< UCSZ01)
29        sts UCSR0C,R16
30        ldi r24,0
31        out DDRD,r24
32        ldi r24,0x0C
33        out PORTD,r24
34        sei
35  main: jmp main
36  knopka:
37        ldi r16, 0x3E
38        sts   UDR0,r16
39        Reti
```

1 строка – подключение библиотеки для ATmega328P

2-5 строки – инициализация прерываний. В рамках этого примера используется прерывание INT0

7-10 строки – инициализация стека

11-18 строки – настройка прерываний

19-25 строки – задаем скорость передачи - бит/с (бод). *ВАЖНО: если скорости приёмника и передатчика не будут совпадать, то передача данных будет некорректной*. 19 строка – тактовая частота генератора платы. 20 строка – требуемая скорость. 21 строка – вычисляем значение, которое нужно занести в UBBR0 (см. в описании лабораторной работы). Так как регистр UBBR0L является 12 разрядным, то занесение числа должно проходить в 2 этапа – сначала заносим старшую часть, после – младшую (22-25 строки).

26-29 строки – настройки UART. 25 и 26 – настраиваем плату как передатчик. 28 и 29 - количество бит данных в пакете

30 – 33 строки – указываем, на какие биты PORTD будем посылать сигнал с кнопок (в нашем случае это 2 и 3).

34 строка – разрешаем прерывания.

35 строка – основная программа.

36-39 – подпрограмма прерывания INT0. Она посылает символ «>» (В кодировке ASCII = 3E)

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Рисунок 4 – Таблица кодировки ASCII

Монитор порта на ARDUINO IDE

Теперь нам нужно понять, какие данные плата будет посылать по UART'у. Для этого воспользуемся приложением ARDUINO IDE - это программная среда разработки, использующая C++ и предназначенная для программирования всех плат Arduino.

Подключаем плату через USB и в диспетчере устройств (ПКМ по «Мой компьютер» -> диспетчер устройств / через «Пуск») проверяем, что она определилась как COM порт.

Далее открываем ARDUINO IDE Выберите во вкладке «Инструменты» плату Arduino Uno, как это сделано на рисунке 5.

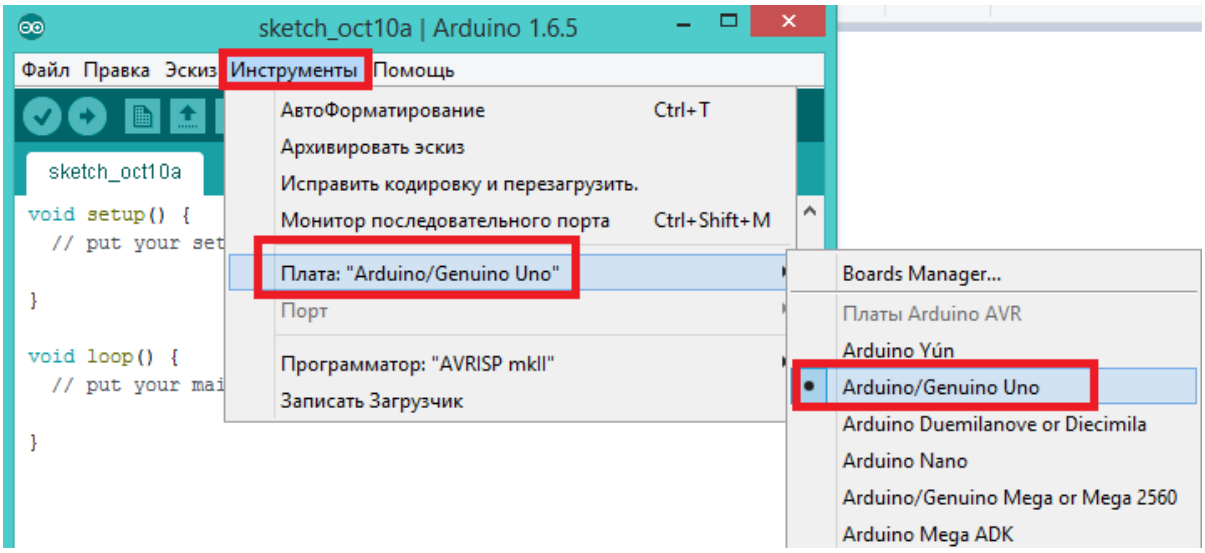


Рисунок 5 – Выбор платы в ARDUINO IDE

Далее выбираем в разделе соответствующий COM порт как на рисунке 6 (в примере COM4, имя порта может отличаться).

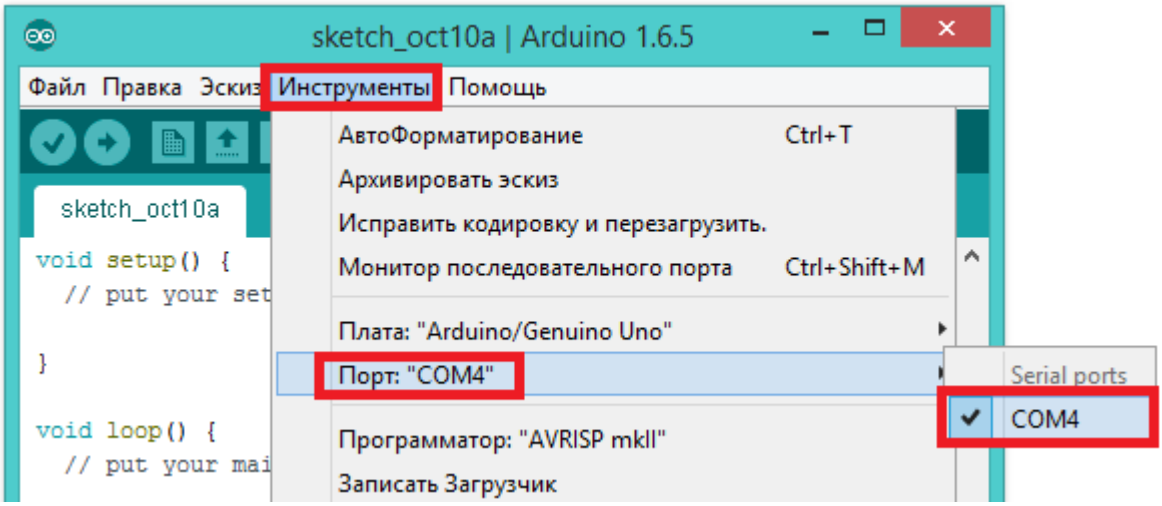


Рисунок 6 – Выбор COM порта

Во вкладке «Инструменты» выбираем «Монитор последовательного порта». Теперь, нажимая на кнопку, подключенному ко 2 биту PORTD будет выводится символ.

Задание 1: Измените выводимый символ в соответствии с таблицей. Результат зафиксируйте в виде скриншота Монитора последовательного порта в ARDUINO IDE. Выполните сборку программы. Загрузите код в контроллер. Проанализируйте работу контроллера. Передачу данных на контроллер выполните с помощью монитора порта Arduino IDE.

№ варианта	1	2	3	4	5	6	7	8	9	10
Символ	=	A	v	R	/	#	w	@	i	T

Задание 2: Добавьте подпрограмму прерывания INT1, которая выводит один любой символ (на ваш выбор). Результат зафиксируйте в виде скриншота Монитора последовательного порта в ARDUINO IDE. Выполните сборку программы. Загрузите код в контроллер. Проанализируйте работу контроллера. Передачу данных на контроллер выполните с помощью монитора порта Arduino IDE.

Задание 3. Приём информации: исправьте код примера таким образом, чтобы плата реагировала на принятый байт (см. Таблицу) следующим

образом - если получен символ согласно номеру варианта, светодиод L (PB5) должен зажечься, в противоположном случае погаснуть. Учитывая, что скорость работы контроллера выше скорости обмена данными по UART, анализ регистра UDR выполняйте только по нажатию на клавишу (INT0).

№ варианта	1	2	3	4	5	6	7	8	9	10
Символ	A	T	m	e	g	a	3	2	8	P

Для того, чтобы контроллер UART принимал данные, необходимо разрешить данную операцию (см. описание регистра UCSR0B). Выполните сборку программы. Загрузите код в контроллер. Проанализируйте работу контроллера. Передачу данных на контроллер выполните с помощью монитора порта Arduino IDE.

Задание 4: Приём информации по прерыванию UART: В предыдущем примере виден большой недостаток, связанный с разными частотами работы процессора и контроллера UART. Наиболее удобным способом приёма информации от внешних устройств является использование прерывания по получению данных. Адрес вектора прерывания по завершению приёма информации контроллером UART в таблице векторов см. в лабораторной работе 5. Для разрешения прерывания необходимо установить соответствующий бит в регистре UCSR0B. Выполните сборку программы. Загрузите код в контроллер. Проанализируйте работу контроллера. Передачу данных на контроллер выполните с помощью монитора порта Arduino IDE.