Лабораторная работа №15:

Тема: Связь. Использование UART для последовательной связи.

Цель: Изучить принципы работы UART на микроконтроллерах AtMega.

Лабораторное оборудование:

Компьютер с установленным программным обеспечением:

- Atmel Studio - для программирования микроконтроллеров AtMega.
- Proteus - для моделирования работы микроконтроллеров AtMega.

Теоретическая часть.

UART — *от англ. Universal Asynchronous Receiver-Transmitter* — Универсальный Асинхронный Приёмопередатчик. В простейшем случае имеет линии: RX (receiver — приём данных), TX (transmitter — передача данных) и общий провод.

USART — *от англ. Universal Synchronous/Asynchronous Receiver-Transmitter* — Универсальный Синхронный/Асинхронный Приёмопередатчик. Кроме линий передачи данных, может иметь отдельную линию для сигнала синхронизации. Главное отличие USART в том, что он может работать как в синхронном, так и в асинхронном режимах.

В синхронном режиме, для синхронизации приёмного и передающего устройств, с передающей стороны могут посылаться сигналы синхронизации по линии данных, либо может использоваться отдельная линия синхронизации. Синхронный режим обычно используется в специфических случаях (например, если необходимо обеспечить высокую скорость передачи данных).

В асинхронном режиме — синхронизация осуществляется только по стартовым битам, без каких-либо дополнительных сигналов синхронизации, поэтому, для успешной передачи данных, приёмник и передатчик заранее должны быть настроены на одинаковые скорость обмена и формат пакетов. Т.к. реализация асинхронного режима проще, то в подавляющем большинстве случаев, для простых задач, используют именно его.

Протокол обмена USART

Пакет данных USART состоит из:

- стартового бита;
- 5, 6, 7, 8 или 9 бит данных;
- бита контроля чётности (опционально);
- 1 или 2 стоп-бит.

Формат пакета USART

Здесь:

St — стартовый бит (логический 0);

0-8 — биты данных;

P — бит контроля чётности — может быть проверка на чётность (Even), нечётность (Odd), или отсутствовать;

Sp — стоп биты (логическая 1);

IDLE — нет передаваемых данных (логическая 1).

Формат передачи данных устанавливается конфигурационными битами в регистрах UCSRB и UCSRC.

Для приема и передачи данных USART использует следующие сигналы:

- Прием данных RxD (ножка PDO Atmega8)
- Передача данных TxD (ножка PD1 Atmega8)
- Синхронизация XCK (ножка PD4 Atmega8)

Описание регистров

Рассматривать регистры конфигурации и контроля состояния USART будем на примере МК Atmega8. Во всех остальных МК AVR — всё очень похоже, только названия и расположение бит и регистров может незначительно отличаться.

Для управлением работы с USART используются следующие регистры:

UCSRA — содержит флаги состояния приема/передачи данных.

UCSRB — определяет какие прерывания генерировать при наступление определенных событий, разрешает/запрещает передачу/прием.

UCSRC — задает режим работы синхронный/асинхронный, определяет режим работы контроля данных (проверка на четность/не честность или отключено), количество стоп битов (1 или 2).

UBRR — определяет скорость приема/передачи данных

Регистр UCSRA:

Биты	7	6	5	4	3	2	1	0	
	RXC	**TXC**	**UDRE**	**FE**	**DOR**	**PE**	**U2X**	**MPCM**	**UCSRA**
Запись/Чтение	R	R/W	R	R	R	R	R/W	R/W	
Нач. значение	0	0	1	0	0	0	0	0	

RXC — флаг **завершения приема**, устанавливается в 1 при наличие непрочитанных данных в буфере приемника **UDR**;

TXC — флаг **завершения передачи**, устанавливается в 1 при передачи всех разрядов из передатчика **UDR**;

UDRE — флаг **опустошения регистра передатчика**, устанавливается в 1 при пустом буфере передатчика **UDR** после передачи;

FE — флаг **ошибки кадрирования**, устанавливается в 1 при обнаружение неправильного кадра, когда стоп бит равен 0.

DOR — флаг **переполнения регистра приемника**, устанавливается в 1, когда байт данных принят, а предыдущий еще не прочитан из **UDR;**

PE — флаг **ошибки контроля четности**, устанавливается в 1 при обнаружение ошибки контроля четности (если включена проверка);

U2X — бит установки удвоенной скорости обмена, если установлена 1, то скорость передачи удваивается, данный бит используется только при **асинхронном режиме работы**;

MPCM — бит **мультипроцессорного обмена**, если установлена 1, то контроллер аппаратно не принимает информацию, а только посылки с адресами, далее устанавливается бит завершения приема (или прерывание) и программа обрабатывает адрес, её ли это адрес. Отличие информации от адреса определяется с помощью 9-ого бита в режиме 9-и битового обмена.

Регистр UCSRB:

Биты	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Запись/чтение	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Нач. значение	0	0	0	0	0	0	0	0	

RXCIE — бит разрешения прерывания по **завершению приема**, если установлена 1, то при установке флага **RXC** регистра **UCSRA** произойдет прерывание прием завершен;

TXCIE — бит разрешения прерывания по **завершению передачи**, если установлена 1, то при установке флага **TXC** регистра **UCSRA** произойдет прерывание передача завершена;

UDRIE — бит разрешения прерывания по **опустошению регистра передатчика**, если установлена 1, то при установке флага **UDRE** регистра **UCSRA** произойдет прерывание регистр данных пуст;

RXEN — бит **разрешения приема**, при установки 1 разрешается работа приемника **USART** и включается вывод RXD;

TXEN — бит **разрешения передачи**, при установки 1 разрешается работа передатчика **USART** и включается вывод **TXD**;

UCSZ2 — бит **формат посылок**, данный бит совместно с битами **UCSZ1** и **UCSZ0** регистра **UCSRC** определяют количество бит данных в посылке.

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	1	1	9-bit

RXB8 — **9 разряд принимаемых данных** при использование данного бита , данные необходимо считывать до считывание регистра **UDR**;

TXB8 — **9 разряд передаваемых данных** при использование данного бита, данные необходимо записывать до записи в регистр **UDR**.

Регистр UCSRC:

Бит	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Чтение/запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Нач. значение	1	0	0	0	0	1	1	0	

URSEL — данный бит отвечает за выбор работа с регистром **UCSRC**или **UBRR**, при установке 1 мы работаем с регистром **UCSRC**, при 0 мы работаем с регистром **UBRR;**

UMSEL — данным битом выбираем режим работы: асинхронный или синхронный, **1** — режим синхронный, 0 — режим асинхронный;

UPM1, UPM0 — биты выбора режима проверки на четность/нечетность;

UPM1	UPM0	Режим проверки на четность
0	0	Выключен
1	0	Включен, проверка на четность
1	1	Включен, проверка на нечетность

USBS — данный бит отвечающий за количество стоп-битов в посылке: 1 — два стоп-бита, 0 — один стоп-бит;

UCSZ1, UCSZ0 , UCSZ2(расположен в регистре UCSRB) — определяют количество бит данных в посылке;

UCPOL — данный бит определяет полярность тактового сигнала, определяет по какому фронту принимать/передавать данные – по спадающему или по нарастающему.

Регистр UBRR отвечает за скорость обмена, и состоит из двух 8-и битных регистров — UBRRH и UBRRL

Биты	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Чтение/запись	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начал. значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Скорость передачи данных исчисляются в бодах (количество передаваемых бит в секунду, причем в количество бит входят как данные так и стоп биты и другая информация).

Вычисляется следующим образом:

UBRR = (**fCK** /(**BAUD** * **16**)) — 1 — если значение бита **U2X = 0.**

UBRR = (**fCK** /(**BAUD** * **8**)) — 1 — если значение бита **U2X =1.**

где:

fCK — тактовая частота микроконтроллера в герцах;

BAUD — требуемая скорость в бодах;

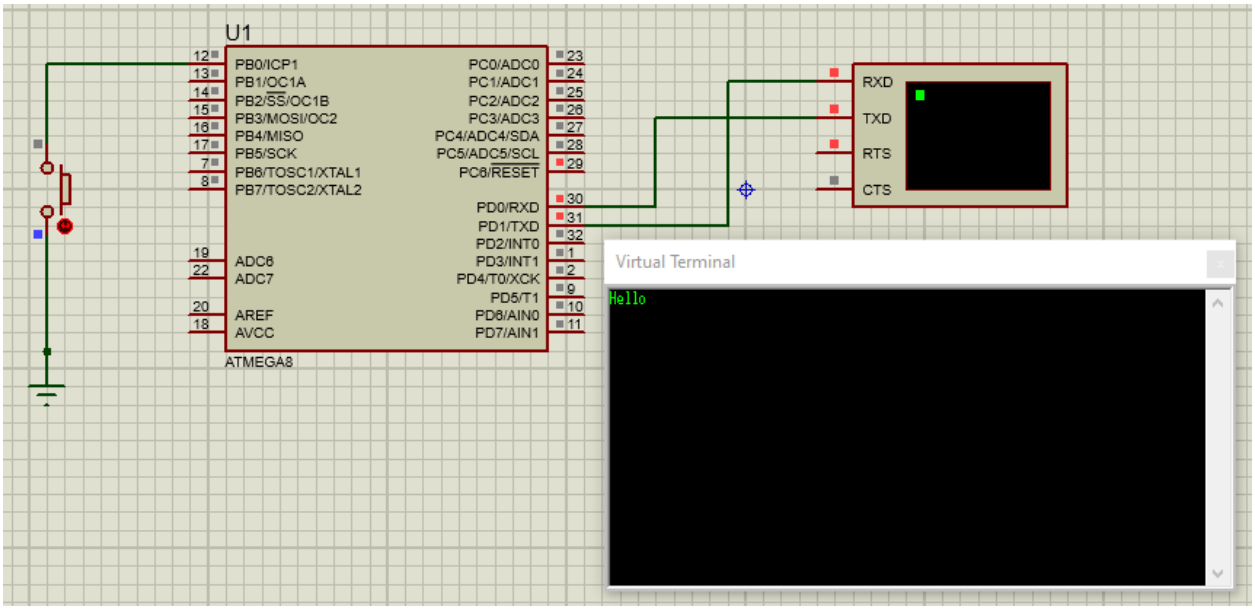
UBRR — содержимое регистра **UBRR.**

В рамках данной работы будем использовать следующие настройки: асинхронный режим, скорость 9600 бод, 8 бит данных, 1 стоп-бит, без проверки четности.

Пример использования UART:

```
#define F_CPU 8000000 // Рабочая частота контроллера
#define UBRRL_value (F_CPU/(9600L*16))-1 //подсчитываем значение
регистра UBRR (скорость 9600).

#include <avr/io.h> // определение регистров ввода-вывода
#include <util/delay.h> // функции задержки

void init_UART() {
	UBRRL = UBRRL_value; // младшие 8 бит значения UBRR (регистра
настройки скорости передачи данных)
	UBRRH = UBRRL_value >> 8; // Старшие 8 бит значения UBRR
	UCSRB |=(1<<TXEN); //Устанавливает бит TXEN в регистре UCSRB,
разрешая передачу данных через UART.
	UCSRC |=(1<< URSEL)|(1<< UCSZ0)|(1<< UCSZ1); // Настраиваем
формат кадра на 8 бит данных, используя биты URSEL, UCSZ0, и UCSZ1 в
регистре UCSRC
}

void send_UART(char value) {
	while(!( UCSRA & (1 << UDRE))); // Ожидаем когда очистится буфер
передачи. Бит UDRE в регистре UCSRA показывает готовность буфера.
	UDR = value; // Помещаем символ value в регистр данных UDR,
начиная его передачу.
}

int main(void)
{
	init_UART(); //инициализация USART
	send_UART('H'); //посылаем ASCII код знака 'H'
	send_UART('e'); //посылаем ASCII код знака 'e'
	send_UART('l'); //посылаем ASCII код знака 'l'
	send_UART('l'); //посылаем ASCII код знака 'l'
	send_UART('o'); //посылаем ASCII код знака 'o'
	send_UART('\r');//переход в начало строки
	send_UART('\n');//переход на новую строку

	while(1)
	{
		// Бесконечный цикл
	}
}
```

Работа программы в Proteus:



Задание:

1. В Atmel Studio написать программу на языке C, которая будет:
    - Инициализировать UART для передачи данных.
    - При включении передавать по UART строку "Hello".
    - При нажатии кнопки передавать строку с вашей фамилией или именем
2. В Proteus:
    - Собрать схему, включающую микросхему AtMega8 и виртуальный терминал.
    - Загрузить в микросхему AtMega8 программу, написанную в п. 1.
    - Проверить работоспособность программы.

Отчет должен содержать:

- листинг программы на языке C.
- Скриншоты с результатами работы программы в Proteus.
- Выводы