Использование интерфейса USART микроконтроллеров AVR

Цель: Изучить интерфейс USART микроконтроллера ATMEGA8, реализовать прием и передачу в среде Proteus.

Микроконтроллеры **AVR** имеют в своем составе модуль полнодуплексного универсального асинхронного приемопередатчика **UART** (в семействе Mega универсальный синхронный/асинхронный приемопередатчик **USART**). Через него осуществляется прием и передача информации, представленной последовательным кодом, поэтому модуль **UART** часто называют также последовательным портом. С помощью этого модуля микроконтроллер может обмениваться данными с различными внешними устройствами.

Поток данных, передаваемых по каналу **UART**, представляет собой совокупность посылок или кадров. Каждый кадр содержит стартовый бит, восемь или девять битов данных и стоповый бит. Стартовый бит имеет уровень логического 0, стоповый - уровень логической 1. Скорость передачи данных может варьироваться в широких пределах, причем высокие скорости передачи могут быть достигнуты даже при относительно низкой тактовой частоте микроконтроллера.

Обмен данными по последовательному интерфейсу

Известно, что при передаче данных могут произойти различные сбои. Модуль **UART**, реализованный в микроконтроллерах, способен при приеме обнаруживать ошибку формата и переполнение. Для взаимодействия с программой в модуле предусмотрены прерывания при наступлении следующих событий: прием завершен с адресом вектора $009 в таблице

векторов прерываний, регистр данных передатчика пуст с адресом вектора $00A, передача завершена с адресом вектора $00B.

Выводы микроконтроллера, используемые модулем **UART**, являются линиями порта PD. В качестве входа приемника (RXD) используют вывод PD0, а в качестве выхода передатчика (TXD) - вывод PD1.

Принимаемые и передаваемые данные (восемь разрядов) хранятся в регистре **UDR**. Физически регистр **UDR** состоит из двух отдельных регистров, один из которых используется для передачи данных, другой - для приема. При чтении регистра **UDR** выполняется обращение к регистру приемника, при записи - к регистру передатчика.

Регистры управления модулями USART на примере микроконтроллера ATmega8

Буферные регистры приемника и передатчика располагаются по одному адресу пространства ввода/вывода и обозначаются как регистр данных **UDR (Universal Data Register)**. В этом регистре хранятся младшие 8 разрядов принимаемых и передаваемых данных. При чтении выполняется обращение к буферному регистру **UDR** приемника, при записи — к буферному регистру передатчика. 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.

Для управления модулями **USART** используются три регистра: **UCSRA, UCSRB и UCSRC.**

Описание разрядов регистра **UCSRA.**

Разряд	Название	Описание
7	RXC	Флаг завершения приема. Флаг устанавливается в «1» при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера (в UART — после прочтения регистра данных). Если разряд RXCIE регистра UCSRB установлен, то при

		установке флага генерируется запрос на прерывание «прием завершен»
6	TXC	Флаг завершения передачи. Флаг устанавливается в «1» после передачи всех разрядов посылки из сдвигового регистра передатчика, при условии, что в регистр данных UDR не было загружено нового значения. Если разряд TXCIE регистра UCSRB (UCSRnB) установлен, то при установке флага генерируется прерывание «передача завершена». Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, записью в него лог. 1
5	UDRE	Флаг опустошения регистра данных. Данный флаг устанавливается в «1» при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если разряд UDRIE регистра UCR (UCSRB) установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных
4	FE	Флаг ошибки кадрирования. Флаг устанавливается в «1» при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен «0». Флаг сбрасывается при приеме стоп-бита, равного «1»
3	DOR	Флаг переполнения. В USART флаг устанавливается в «1», если в момент обнаружения нового старт-бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (два значения). В UART флаг устанавливается в «1», если новый кадр будет помещен в сдвиговый регистр приемника до того, как из регистра данных будет считано предыдущее слово. Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер

2	PE	Флаг ошибки контроля четности. Флаг устанавливается в «1», если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот разряд постоянно сброшен в «0»
1	U2X	Удвоение скорости обмена. Если этот разряд установлен в «1», коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. В USART разряд U2X используется только при асинхронном режиме работы. В синхронном режиме он должен быть сброшен
0	MPCM	Режим мультипроцессорного обмена. Разряд MPCM используется в режиме мультипроцессорного обмена. Если он установлен в «1», ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются

Описание разрядов регистра **UCSRB.**

Разряд	**Название**	**Описание**
7	RXCIE	Разрешение прерывания по завершению приема. Если данный разряд установлен в «1», то при установке флага RXC регистра UCSRA генерируется прерывание «прием завершен» (если флаг I регистра SREG установлен в «1»)
6	TXCIE	Разрешение прерывания по завершению передачи. Если данный разряд установлен в «1», то при установке флага TXC регистра UCSRA генерируется прерывание

		«передача завершена» (если флаг I регистра SREG установлен в «1»)
5	UDRIE	Разрешение прерывания при очистке регистра данных UART. Если данный разряд установлен в «1», то при установке флага UDRE в регистра UCSRA генерируется прерывание «регистр данных пуст» (если флаг I регистра SREG установлен в «1»)
4	RXEN	Разрешение приема. При установке этого разряда в «1» разрешается работа приемника USART/UART и переопределяется функционирование вывода RXD (RXDn). При сбросе разряда RXEN работа приемника запрещается, а его буфер сбрасывается. Значения флагов TXC, DOR/OR и FE при этом становятся недействительными
3	TXEN	Разрешение передачи. При установке этого разряда в «1» разрешается работа передатчика UART и переопределяется функционирование вывода TXD. Если разряд сбрасывается в «0» во время передачи, выключение передатчика произойдет только после завершения передачи данных, находящихся в сдвиговом регистре и буфере передатчика
2	UCSZ2	Формат посылок. Этот разряд используется для задания размера слов данных, передаваемых по последовательному каналу. В модулях USART он используется совместно с разрядами UCSZ1:0 регистра UCSRC. В модулях UART,

		если разряд CHR9 установлен в «1», осуществляется передача и прием 9 разрядных данных, если сброшен — 8 разрядных
1	RXB8	8 й разряд принимаемых данных. При использовании 9 разрядных слов данных этот разряд содержит значение старшего разряда принятого слова. В случае USART содержимое этого разряда должно быть считано до прочтения регистра данных UDR
0	TXB8	8 й разряд передаваемых данных. При использовании 9 разрядных слов данных, содержимое этого разряда является старшим разрядом передаваемого слова. Требуемое значение должно быть занесено в этот разряд до загрузки байта данных в регистр UDR

Описание разрядов регистра **UCSRC.**

Разряд	**Название**	**Описание**
7	URSEL	Выбор регистра. Этот разряд определяет, в какой из регистров модуля производится запись. Если разряд установлен в «1», обращение производится к регистру UCSRC. Если же разряд сброшен в «0», обращение производится к регистру UBRRH.
6	UMSEL	Режим работы USART. Если разряд сброшен в «0», модуль USART работает в асинхронном режиме. Если разряд установлен в «1», то

		модуль USART работает в синхронном режиме
5	UPM1	Режим работы схемы контроля и формирования четности. Эти разряды определяют функционирование схем контроля и формирования четности
4	UPM0	
3	USBS	Количество стоп-битов. Этот разряд определяет количество стоп-битов, посылаемых передатчиком. Если разряд сброшен в «0», передатчик посылает 1 стоп-бит, если установлен в «1», то 2 стоп-бита. Для приемника содержимое этого разряда безразлично
2	UCSZ1	Формат посылок. Совместно с разрядом UCSZ2 эти разряды определяют количество разрядов данных в посылках (размер слова)
1	UCSZ0	
0	UCPOL	Полярность тактового сигнала. Значение этого разряда определяет момент выдачи и считывания данных на выводах модуля. Разряд используется только при работе в синхронном режиме. При работе в асинхронном режиме он должен быть сброшен в «0»

Скорость приема/передачи USART

В асинхронном режиме, а также в синхронном режиме при работе в качестве ведущего, скорость приема и передачи данных задается контроллером скорости передачи, функционирующим как делитель системного тактового сигнала с программируемым коэффициентом деления. Коэффициент определяется содержимым регистра контроллера **UBRR**. В блок приемника сформированный сигнал поступает сразу, а в блок передатчика — через дополнительный делитель, коэффициент деления которого (**2, 8 или 16**)

зависит от режима работы модуля **USART/UART**. Регистр **UBRR** является 12 разрядным и физически размещается в двух регистрах ввода/вывода **UBRRH** и **UBRRL.**

При работе в асинхронном режиме скорость обмена определяется не только содержимым регистра **UBRR**, но и состоянием разряда **U2X** регистра **UCSRA.** Если этот разряд установлен в «1», коэффициент деления предделителя уменьшается в два раза, а скорость обмена соответственно удваивается. При работе в синхронном режиме этот разряд должен быть сброшен.

Скорость обмена определяется следующими формулами, где **BAUD** — скорость передачи в бодах, **fCK** — тактовая частота микроконтроллера, **UBRR** — содержимое регистра контроллера скорости передачи (0…4095):

асинхронный режим (обычный, U2Xn = «0»)

BAUD = fCK/16(UBRR + 1);

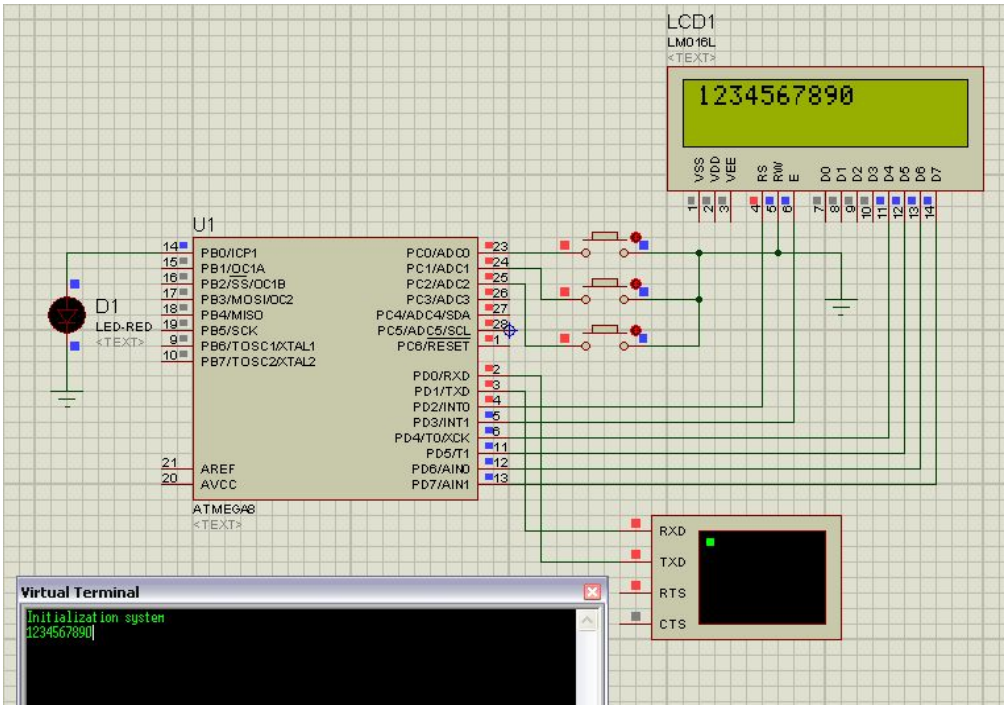
асинхронный режим (ускоренный, U2Xn = «1»)

BAUD = fCK/8(UBRR + 1);

синхронный режим ведущего

BAUD = fCK/2(UBRR + 1).

Ниже представлен пример тестовой программы для изучения протокола USART, где микроконтроллер Atmega8 обменивается информацией с терминалом, для наглядности к контроллеру подключен LCD 16x02 дисплей. При нажатии на кнопки 1-3 в терминале высвечиваются соответствующие строки, также если выводить в терминал символы "a" или "b", будет загораться или гаснуть светодиод, подключенный к порту PB0 контроллера. Любые символы, выводимые в терминал будут также высвечиваться на LCD дисплее. Для тестирования в "железе", микроконтроллер подключается к компьютеру через микросхему преобразователя уровней MAX232, для обмена данными используется стандартная программа Hyper Terminal от Microsoft.




```
//*** Пример работы с USART интерфейсом микроконтроллеров AVR ***
#include <avr/io.h>#include <avr/interrupt.h>
#define BAUDRATE 9600 // Скорость обмена данными
#define F_CPU 8000000UL // Рабочая частота контроллера

unsigned char NUM = 0;
unsigned char count = 0;
unsigned char byte_receive = 0;
unsigned char i = 1;

// Функция задержки в мкс
void _delay_us(unsigned char time_us)
{ register unsigned char i;

for(i = 0; i < time_us; i++)
{
asm volatile(" PUSH  R0 ");
asm volatile(" POP   R0 ");
}
}

// Функция задержки в мс
void _delay_ms(unsigned int time_ms)
{ register unsigned int i;
```

```
for(i = 0; i < time_ms; i++)
{
_delay_us(250);
_delay_us(250);
_delay_us(250);
_delay_us(250);
}
}

#define RS PD2
#define EN PD3

// Функция передачи команды
void lcd_com(unsigned char p)
{
PORTD &= ~(1 << RS); // RS = 0 (запись команд)
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}

// Функция передачи данных
void lcd_data(unsigned char p)
{
PORTD |= (1 << RS)|(1 << EN); // RS = 1 (запись данных), EN - 1 (начало записи
команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}

// Функция инициализации LCD
```

```
void lcd_init(void)
{
_delay_ms(50); // Ожидание готовности ЖК-модуля

// Конфигурирование четырехразрядного режима
PORTD |= (1 << PD5);
PORTD &= ~(1 << PD4);

// Активизация четырехразрядного режима
PORTD |= (1 << EN);
PORTD &= ~(1 << EN);
_delay_ms(5);

lcd_com(0x28); // шина 4 бит, LCD - 2 строки
lcd_com(0x08); // полное выключение дисплея
lcd_com(0x01); // очистка дисплея
_delay_us(100);
lcd_com(0x06); // сдвиг курсора вправо
lcd_com(0x0C); // включение дисплея, курсор не видим
}

// Функция вывода строки на LCD
void lcd_string(unsigned char command, char *string)
{
lcd_com(0x0C);
lcd_com(command);
while(*string != '\0')
{
lcd_data(*string);
string++;
}
}

// Функция передачи данных по USART
void uart_send(char data)
{
while(!( UCSRA & (1 << UDRE)));     // Ожидаем когда очистится буфер передачи
UDR = data; // Помещаем данные в буфер, начинаем передачу
}

// Функция передачи строки по USART
void str_uart_send(char *string)
{
while(*string != '\0')
```

```
{
uart_send(*string);
string++;
}
}

// Функция приема данных по USART
int uart_receive(void)
{
while(!(UCSRA & (1 << RXC))); // Ожидаем, когда данные будут получены
return UDR; // Читаем данные из буфера и возвращаем их при выходе из подпрограммы
}

// Функция инициализации USART
void uart_init(void)
{
// Параметры соединения: 8 бит данные, 1 стоповый бит, нет контроля четности
// USART Приемник: Включен
// USART Передатчик: Включен
// USART Режим: Асинхронный
// USART Скорость обмена: 9600

UBRRL = (F_CPU/BAUDRATE/16-1); // Вычисляем скорость обмена данными
UBRRH = (F_CPU/BAUDRATE/16-1) >> 8;

UCSRB |= (1 << RXCIE)| // Разрешаем прерывание по завершению приема данных
        (1 << RXEN)|(1 << TXEN); // Включаем приемник и передатчик

UCSRC |= (1 << URSEL)| // Для доступа к регистру UCSRC выставляем бит URSEL
       (1 << UCSZ1)|(1 << UCSZ0); // Размер посылки в кадре 8 бит
}

// Прерывание по окончанию приема данных по USART
ISR(USART_RXC_vect)
{
NUM = UDR; // Принимаем символ по USART
byte_receive = 1;
uart_send(NUM); // Посылаем символ по USART

if(NUM == 'a') // Если принят символ "a", включаем светодиод
PORTB |= (1 << PB0);
```

```
if(NUM == 'b') // Если принят символ "b", выключаем светодиод
PORTB &= ~(1 << PB0);
}

// Главная функция
int main(void)
{
DDRB |= (1 << PB0); // Светодиод
PORTB = 0x00;

DDRC  = 0x00;
PORTC = 0xFF;

DDRD  = 0b11111110;
PORTD = 0x00;

lcd_init(); // Инициализация LCD
uart_init(); // Инициализация USART

sei(); // Глобально разрешаем прерывания

str_uart_send("Initialization system\r\n"); // Передаем строку по USART
lcd_string(0x80, " AVR USART TEST "); // Выводим строку на LCD
_delay_ms(2500);
lcd_com(0x01); // Очищаем LCD

while(1)
{

if((PINC & (1 << PC0)) == 0) // Если нажата кнопка 1
{
while((PINC & (1 << PC0)) == 0){} // Ждем отпускания кнопки 1
str_uart_send("Button 1 TEST\r\n"); // Передаем строку по USART
lcd_string(0x80, "Button 1 TEST"); // Выводим строку на LCD
_delay_ms(1000);
lcd_com(0x01); // Очищаем LCD
}

if((PINC & (1 << PC1)) == 0) // Если нажата кнопка 2
{
while((PINC & (1 << PC1)) == 0){} // Ждем отпускания кнопки 2
str_uart_send("Button 2 TEST\r\n"); // Передаем строку по USART
lcd_string(0x80, "Button 2 TEST"); // Выводим строку на LCD
_delay_ms(1000);
lcd_com(0x01); // Очищаем LCD
```

```
}

if((PINC & (1 << PC2)) == 0) // Если нажата кнопка 3
{
while((PINC & (1 << PC2)) == 0){} // Ждем отпускания кнопки 3
str_uart_send("Button 3 TEST\r\n"); // Передаем строку по USART
lcd_string(0x80, "Button 3 TEST"); // Выводим строку на LCD
_delay_ms(1000);
lcd_com(0x01); // Очищаем LCD
}

if(byte_receive)
{
byte_receive = 0;
count++;
lcd_data(NUM); // Выводим символ на LCD
if(count > 16) // Если строка заполнена
{
count = 0;
lcd_com(0x01); // Очищаем LCD
}
}
}
}
```

Задания для выполнения:

1. Изучить интерфейс USART микроконтроллера ATMEGA8 и составить конспект.
2. Реализовать работу приема/передачи информации в среде Proteus.

Результат и архив с проектом отправить на e-mail: rasov@rambler.ru с темой USART_ФИО