МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИССЛЕДОВАНИЕ ТЕХНОЛОГИЙ ПОСЛЕДОВАТЕЛЬНОЙ ПЕРЕДАЧИ ДАННЫХ СРЕДСТВАМИ МИКРОКОНТРОЛЛЕРОВ AVR

Методические указания к выполнению лабораторных работ
по дисциплине «Микроконтроллеры в системах управления»
для студентов направления подготовки
27.03.04 – Управление в технических системах

Севастополь
2020

УДК 681.5

Исследование технологий последовательной передачи данных средствами микроконтроллеров AVR: Методические указания к выполнению лабораторных работ по дисциплине «Микроконтроллеры в системах управления» для студентов направления подготовки 27.03.01 – «Управление в технических системах» / Разраб. А.А. Кабанов – Севастополь: Изд-во СевГУ, 2020. – 22 с.

Целью методических указаний является оказание помощи студентам при выполнении лабораторных работ, целью которых является приобретение навыков программирования микроконтроллеров AVR.

Методические указания предназначены для студентов по направлению подготовки 27.03.04 – «Управление в технических системах».

Методические указания рассмотрены и утверждены на заседании кафедры «Информатика и управление в технических системах», протокол № ___ от _____ 2020.

Допущено учебно-методическим комиссией Института информационных технологий и управления в технических системах СевГУ в качестве методических указаний.

Рецензент:
В.А. Карапетьян, канд. техн. наук, доцент.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является исследование принципов работы последовательной передачи данных средствами микроконтроллеров, получение навыков настройки модуля USART.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Сведения об интерфейсе RS-232

Аббревиатура *USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter)* означает «универсальный синхронный и асинхронный последовательный приемник и передатчик». *USART* – внутреннее устройство микроконтроллера (МК), предназначенное для обмена информацией (байтами данных) с другими устройствами. Так называют устройство, которое может работать с интерфейсом *RS-232* (и его аналогами).

RS-232 (*Recommended Standard 232*) – в телекоммуникациях, стандарт последовательной синхронной и асинхронной передачи двоичных данных между терминалом и коммуникационным устройством. Чаще всего используется в промышленном и узкоспециальном оборудовании, встраиваемых устройствах. Иногда присутствует на современных персональных компьютерах (рис. 1). В персональных компьютерах этот интерфейс используется COM-портом (*Communication port*).

Рисунок 1 – Разъем DB-9 для передачи по RS-232.

Последовательный порт персонального компьютера

Последовательный порт (serial port), серийный порт или COM-порт (произносится «ком-порт», от англ. Communication port) – двунаправленный последовательный интерфейс, предназначенный для обмена битовой информацией.

Последовательным данный порт называется потому, что информация через него передаётся по одному биту, бит за битом последовательно (в отличие от параллельного порта). Хотя некоторые другие интерфейсы компьютера – такие как Ethernet, FireWire и USB – также используют последовательный способ обмена, название «последовательный порт» закрепилось за портом, имеющим стандарт RS-232C, и предназначенным изначально для обмена информацией с модемом. Ранее последовательный порт использовался для подключения терминала, позже для модема или мыши. Сейчас он используется для соединения с источниками бесперебойного питания, для связи с аппаратными средствами разработки встраиваемых

вычислительных систем, спутниковыми ресиверами, а также с приборами систем безопасности объектов.

Наиболее часто для последовательного порта персональных компьютеров используется стандарт RS-232C. Напряжениями высокого и низкого уровней в интерфейсе RS-232C являются +12 В (низкий) и -12 В (высокий). Состояние 0 В не используется при передачи данных и существует только при отключении питания. Интерфейс RS-232C имеет 9 выводов, их назначение приведено в таблице 2.1.

Таблица 2.1 – Выводы последовательного порта компьютера

Номер вывода	Наименование сигнала	Описание
1	DCD	Обнаружение устройства
2	RXD	Прием данных
3	TXD	Посылка данных
4	DTR	Готовность терминала к передаче данных
5	GND	Общий провод (земля)
6	DSR	Data Set Ready. Готовность внешнего устройства к передаче данных
7	RTS	Запрос на передачу
8	CTS	Разрешение передачи данных терминалу
9	RI	Ring Indicator (специализированная сигнальная линия управления)

Оба провода передачи данных работают одинаково. При этом контакт RXD одного устройства соединяется с контактом TXD другого, и наоборот, контакт TXD первого устройства подсоединяется к контакту RXD второго устройства.

При передаче в асинхронном режиме для связи используются 3 вывода:

RxD (Receive), TxD (Transmit) и GND (Ground).

При передаче в синхронном режиме для связи используются 9 выводов:

DCD, RxD, TxD, DTR, GND, DSR, RTS, CTS и RI.

COM-порт компьютера поддерживает следующие скорости обмена данными через интерфейс RS-232C (bps – бит/сек):

• 1200 bps;• 2400 bps;• 4800 bps;• 9600 bps;• 14400 bps;• 19200 bps;

• 28800 bps;• 38400 bps;• 57600 bps;• 76800 bps;• 115200 bps.

Наиболее часто используют скорость 9600 бит/с. Для того, чтобы получить требуемое значение скорости с высокой точностью, следует использовать кварцевые резонаторы, кратные скорости передачи данных, например 11,0592 МГц. Чем выше частота резонатора, тем выше максимальная скорость передачи. Для микроконтроллера Atmega324p с резонатором на 20 МГц возможна скорость до 2 Мбит/с. Однако, данная скорость – это скорость между битами в кадре, а в кадрах есть еще биты управления и биты четности. В результате реальная скорость передачи всегда ниже на 10–35 %. При использовании других резонаторов (4 МГц, 8 МГц и тому подобное) будет возникать погрешность для первых 11 скоростей, тогда как скорости в 250 кбит/с и кратные будут без погрешностей. Чем больше погрешность, тем ниже устойчивость к помехам и выше вероятность ошибок.

Принцип передачи данных по интерфейсу RS-232

Передача данных по интерфейсу RS-232 может проходить в двух режимах:

- асинхронная передача;
- синхронная передача.

По структуре RS-232 это обычный асинхронный последовательный протокол, то есть передающая сторона по очереди выдает в линию 0 и 1, а принимающая отслеживает их и запоминает.

Изначально на линии передачи данных установлен высокий логический уровень (-12 В). Передача данных осуществляется кадрами по 5-8 бит (рис.2). Чтобы обозначить начало и конец кадра, используют «старт-бит» и «стоп-бит». Приемник и передатчик заранее должны быть настроены на работу с одинаковыми параметрами для гарантированной передачи данных. Когда нужно передать данные, передатчик меняет состояние с логической единицы на логический ноль (+12 В). Это стартовый бит. После него начинается передача информационных битов. Этот момент (появление стартового бита) фиксируется приемником, и он переходит в режим получения данных, начиная отсчитывать временные интервалы. Биты передаются через одинаковые фиксированные моменты времени с определенной скоростью, называемой битрейтом (baudrate). Например, пусть этот интервал составляет t = 100 мкс. Передатчик держит стартовый бит (всегда логический ноль) в течение 100 мкс, а затем выставляет на линии первый информационный бит. Еще через 100 мкс он устанавливает второй бит и так далее. Приемник в свою очередь, обнаружив стартовый бит, ждет 100 мкс, затем ждет половину этого времени, т.е. 50 мкс (чтобы попасть как раз в середину первого бита), после чего начинает запоминать состояние линии каждые 100 мкс.

Рисунок 2 – Структура кадра данных USART

За информационными битами следуют бит четности и стоповые биты. Стоповые биты всегда имеют уровень -12 В (логическая единица) и служат одновременно паузой между кадрами данных. Если стоповый бит один, то пауза между кадрами меньше и скорость передачи данных выше. Если их два, то пауза будет дольше. Бит четности (P) служит для контроля правильности передачи данных. Если выставлен параметр четности как Odd (Нечетный), то бит четности должен быть таким, чтобы сумма всех отправленных информационных битов вместе с ним была обязательно

нечетной. Если приемник, приняв кадр, установил, что сумма нечетная, то все верно, если она оказалась четной, значит, была ошибка в одном бите (или в 3-м, 5-м, 7-м, 9-м). Этот способ достаточно примитивен и не обнаружит ошибки во 2-м, 4-м, 6-м или 8-м. Но наиболее вероятной является ошибка именно в 1 бит, а не в 2 и больше. Если четность установлена как Even (Четная), то сумма всех отправленных битов будет всегда четной.

Временные интервалы, по которым происходит передача и прием данных, синхронизируются по стартовому биту. Если они идут синхронно, тогда передача проходит корректно, если же нет, то момент считывания будет смещаться от середины бита в ту или иную сторону.

2.2. Модуль USART микроконтроллеров AVR семейства Mega

2.2.1. Структура модуля USART

Упрощенная структурная схема одного модуля USART/UART приведена на рис. 3 [1]. За работу модуля USART отвечают несколько специальных регистров.

Регистр UDR:

Регистр UDR (USART Data Register) используется для хранения принятых/отправляемых данных и состоит из трех отдельных регистров. При записи значения в UDR происходит запись в тот, который изображен на рис. 3 верхним (UDR (transmit) или TXB). Это значение незамедлительно отправляется другому устройству, и регистр очищается. Когда в контроллер приходит байт информации, он сохраняется в другом UDR-регистре (UDR (receive) или RXB), причем в верхнем RXB-регистре. Как только байт полностью принят, он автоматически копируется в нижний RXB-регистр, откуда его можно уже прочитать, а в верхний при этом принимается следующий байт (чтобы избежать пауз в работе и потери данных). Итак, при чтении данных из UDR берется принятый байт, а при записи в него данные сразу же отправляются через интерфейс приемнику. Если после приема двух байтов в оба RXB-регистра не произошло считывания из нижнего, то третий байт не будет принят и модуль USART выставит флаг ошибки Data OverRun (DOR).

В состав модуля USART также входят регистры управления и статуса – UCSRA, UCSRB и UCSRC.

Регистр UCSRA:

- Бит 7 – RXC (Receive Comlete) – это бит завершения приема. Автоматически устанавливается, если принимаются данные. При чтении принятых данных сбрасывается.
- Бит 6* – TXC (Transmit Comlete) – это бит завершения передачи. Автоматически устанавливается после отправки кадра данных при отсутствии следующих данных для отправки.
- Бит 5 – UDRE (UDR Empty) – это бит отсутствия данных для отправки. Если он установлен, то данные для отправки отсутствуют.
- Бит 4 – FE (Frame Error) – это бит ошибки кадра. Равен единице, если была ошибка в формате принятого кадра.

- Бит 3 – DOR (Data OverRun) – это бит ошибки переполнения буфера. Равен единице, если нет места для приема нового кадра данных.
- Бит 2 – PE (Parity Error) – это бит ошибки чётности. Равен единице, если была ошибка чётности.
- Бит 1* – U2X (Double speed) – это бит двойной скорости USARTa. Если он установлен, то скорость работы и битрейт удваиваются.
- Бит 0* – MPCM (Multi-Processor Communication Mode) – это бит работы в многопроцессорном режиме. Если он равен единице, то включается многопроцессорный режим.

Звездочкой (*) отмечены биты, доступные для записи. Остальные биты доступны только для чтения. Биты 7, 6, 5 могут использоваться для генерации прерываний. Многопроцессорный режим используется тогда, когда к одному порту подключаются сразу несколько микроконтроллеров.

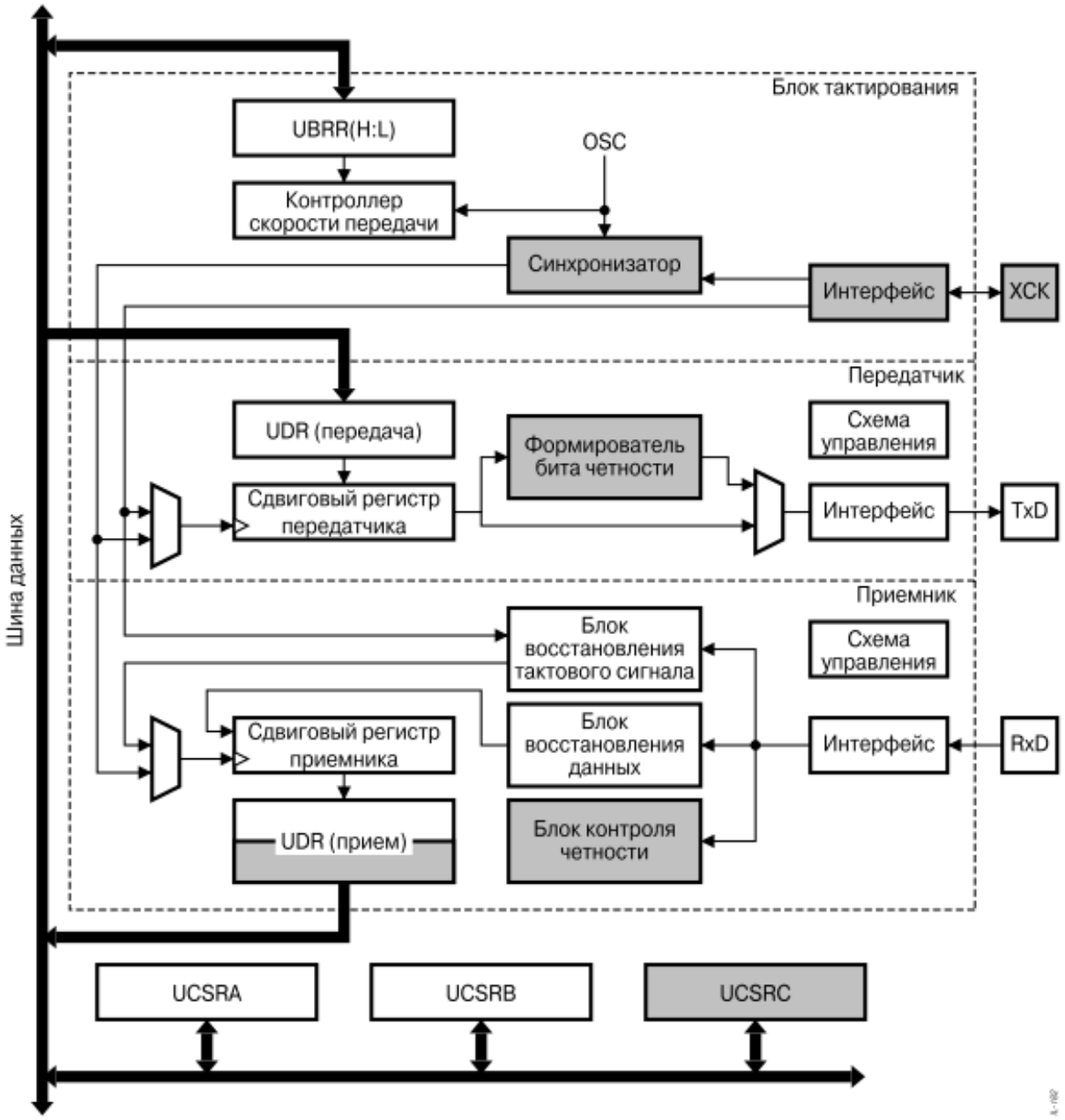

Рисунок 3 – Упрощенная структурная схема одного модуля USART/UART.

Регистр UCSRB:

- Бит 7 – RXCIE (Receive Comlete Interrupt Enable) – бит генерации прерывания при приеме данных. Если установлен, происходит прерывание для немедленной обработки принятых данных.
- Бит 6 – TXCIE (Transmit Comlete Interrupt Enable) – бит генерации прерывания при завершении передачи. Если он установлен, то происходит прерывание после отправки данных.
- Бит 5 – UDRIE (UDR Empty Interrupt Enable) – бит генерации прерывания при отсутствии данных для передачи.
- Бит 4 – RXEN (Receiver Enable) – бит наличия приемника данных. Если он установлен, то МК будет принимать данные через RS-232C.
- Бит 3 – TXEN (Transmitter Enable). Бит наличия передатчика данных. Если установлен, МК будет передавать данные через RS-232C.
- Бит 2 – UCSZ2 (USART Character SiZe 2) – определяет количество бит в кадре. Работает вместе с битами UCSZ1 и UCSZ0 в регистре UCSRC.
- Бит 1* – RXB8 (Receive Data Bit 8). Девятый бит принятых данных (восьмой при нумерации с 0), использующийся в многопроцессорном режиме.
- Бит 0 – TXB8 (Transmit Data Bit 8) – девятый бит отправляемых данных, использующийся в многопроцессорном режиме.

Звездочкой (*) отмечен бит, доступный только для чтения. Все остальные доступны для чтения/записи.

Регистр UCSRC (для моделей кроме ATmega48/88/168):

- Бит 7 – URSEL (USART Register SELect) – бит выбора регистра. 1 – UCSRC, 0 – UBRRH.
- Бит 6 – UMSEL (USART Mode SELect) – бит выбора режима работы: 1 – синхронный, 0 – асинхронный.
- Бит 5 – UPM1 (USART Parity Mode 0) – установка режима четности/нечетности. Если оба сброшены – нет контроля.
- Бит 4 – UPM0 (USART Parity Mode 0) – оба установлены – нечетная сумма. UPM0 = 0 и UPM1 = 1 – четная сумма.
- Бит 3 – USBS (USART Stop Bit Select) – количество стоп-битов. Если он сброшен, то – 1 стоп-бит, если нет, то 2 стоп-бита.
- Бит 2 – UCSZ1 (USART Character Size 1) – это размер кадра данных (вместе с UCSZ2 из UCSRB).
- Бит 1 – UCSZ0 (USART Character Size 0) – возможные его значения приведены в приложении А (см. табл. А.1).
- Бит 0 – UCPOL (USART Clock POLarity) – используется в синхронном режиме. Если равен нулю, то тактирование отправляемых данных осуществляется по переднему фронту импульса, принимаемых – по заднему. Если установлен, то наоборот.

В контроллерах моделей ATmega48/88/168 регистр UCSRC имеет несколько иную структуру, отличающуюся в двух старших битах:

➢ Биты 7,6 –UMSEL1, UMSEL0 – (USART Mode SELect) – биты выбора режима работы модуля. Конфигурация модуля в зависимости от установки этих битов дана в таблице А.2 (приложение А).

Особенностью регистра UCSRC является то, что он имеет тот же адрес, что и верхний регистр битрейта (0x80). Это означает, что процессор не видит разницы между этими регистрами. Для того чтобы их можно было отличать, введен седьмой бит URSEL. Если он равен единице, процессор работает с регистром UCSRC, если нет, то он работает с регистром UBRRH.

2.2.2. Задание скорости передачи данных по модулю USART

Для установки битрейта используются регистры UBRRH и UBRRL. В регистре UBRRH доступны биты 0–3 (если в UCSRC URSEL = 0). Регистр UBRRL доступен весь для сохранения восьми младших битов числа. В этих двух регистрах может быть сохранено 12-битное число (от 0 до 4095). Значение этого числа вычисляется по следующей формуле:

$$UBRR = \frac{f_{OSC}}{X \cdot BAUD} - 1,$$

где X = 16 при режиме с нормальной скоростью (бит U2X = 0) и 8 – в режиме удвоенной скорости (бит U2X = 1), $BAUD$ – нужный битрейт (бит в секунду), f_{OSC} – частота кварцевого резонатора контроллера. По этой формуле легко вычислить значение регистра самостоятельно, но существует также специальная таблица наиболее типичных битрейтов и соответствующих значений UBRR (см. табл. А.3, приложение А).

2.2.3. Работа с модулем USART

Инициализация модуля

На отладочной плате лабораторного макета нужно соединить выводы модуля моста USB-USART RX и TX с выводами микроконтроллера RXD и TXD. В качестве моста USB-USART в работе может использоваться программатор Pololu AVR USB, либо можно организовать передачу данных с другим аналогичным контроллером.

Для микроконтроллера ATmega необходимо соединить вывод RX программатора с выводом PD1 (TXD) микроконтроллера и TX с PD0 (RXD). Вывод GND подключать не обязательно, так как модуль USB-USART и микроконтроллер имеют общую шину питания. К выводу PD1 микроконтроллера подключен красный светодиод для сигнализации высокого уровня напряжения на нем.

Перед началом работы с модулем USART его нужно сконфигурировать в соответствии требуемыми параметрами передачи данных (битрейт, размер пакета данных и т.п.). По умолчанию после перезагрузки контроллера значения во всех регистрах настройки USART сброшены в низкое логическое состояние, то есть все биты имеют значение 0. Поэтому для настройки модуля USART нужно устанавли-

вать только те значения, которые должны иметь высокий уровень (логическую «1»). Для инициализации модуля USART необходимо создать специальную функцию:

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
     {...
     USART_Init(MYUBRR);
     ...
     }

void USART_Init( unsigned int ubrr)
{
     /*Set baud rate */
     UBRR0H = (unsigned char)(ubrr>>8);
     UBRR0L = (unsigned char)ubrr;

     /*Enable receiver and transmitter */
     UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);

     /* Set frame format: 8 data, 2 stop bit */
     UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

Функция `USART_Init` устанавливает величину битрейта (значение заносится в регистр UBRR0), включается приемник и передатчик USART и разрешается прерывание по приему данных. Количество информационных битов в кадре устанавливается равное 8, стоп-бита – 2.

Передача данных

Передача инициируется записью передаваемых данных в буферный регистр передатчика — регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика. Одновременно, если используются 9-разрядные данные, значение разряда TXB8 (TXB8n) регистра UCSRB (UCSRnB) копируется в 9-й разряд сдвигового регистра.

При этом возможны два варианта:

• запись в регистр UDR осуществляется в тот момент, когда передатчик находится в состоянии ожидания (предыдущие данные уже переданы). В этом случае данные пересылаются в сдвиговый регистр сразу же после записи в регистр UDR;

• запись в регистр UDR осуществляется во время передачи. В этом случае данные пересылаются в сдвиговый регистр после передачи последнего стоп-бита текущего кадра.

После пересылки слова данных в сдвиговый регистр, флаг UDRE (UDREn) регистра UCSRA (UCSRnA) устанавливается в «1», что означает готовность передатчика к получению нового слова данных. В этом состоянии флаг остается до следующей записи в буфер. Одновременно с пересылкой в регистре формируется

служебная информация — старт-бит, возможный бит четности (только в USART), а также один или два стоп-бита. После загрузки сдвигового регистра его содержимое начинает сдвигаться вправо и поступать на вывод TXD (TXDn). Скорость сдвига определяется настройками контроллера тактовых сигналов. При работе в синхронном режиме (только USART) изменение состояния вывода TXD (TXDn) происходит по одному из фронтов сигнала XCK (XCKn). Если разряд UCPOL (UCPOLn) регистра UCSRC (UCSRnC) сброшен в «0», изменение состояния вывода происходит по нарастающему фронту сигнала XCK (XCKn), если же установлен в «1» — по спадающему фронту сигнала.

Если во время передачи в регистр UDR было записано новое слово данных, то после передачи последнего стоп-бита оно пересылается в сдвиговый регистр. Если же к моменту окончания передачи кадра такой записи выполнено не было, устанавливается флаг прерывания «Передача завершена» TXC (TXCn) регистра UCSRA (UCSRnA). Сброс флага осуществляется аппаратно при входе в подпрограмму обработки соответствующего прерывания или программно, записью в этот разряд лог. 1.

Выключение передатчика осуществляется сбросом разряда TXEN (TXENn) регистра UCSRB (UCSRnB). Если в момент выполнения этой команды осуществлялась передача, сброс разряда произойдет только после завершения текущей и отложенной передач, т. е. после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD (TXDn) может использоваться как контакт ввода/вывода общего назначения. Пример кода функции передачи данных с кадром размером в 9 бит показан ниже.

```
void USART_Transmit( unsigned int data )
{
/* Wait for empty transmit buffer */
while ( !( UCSR0A & (1<<UDRE0))) );

/* Copy 9th bit to TXB8 */
UCSR0B &= ~(1<<TXB8);
if ( data & 0x0100 )
     UCSRnB |= (1<<TXB8);

/* Put data into buffer, sends the data */
UDR0 = data;
}
```

Прием данных

Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый разряд содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи или тактовым сигналом XCK (XCKn). Считанные разряды данных последовательно помещаются в сдвиговый регистр приемника до обнаружения первого стоп-бита кадра. После этого содержимое сдвигового регистра пересылается в буфер приемника, из которого принятое значение может быть получено путем чтения регистра данных модуля. При использовании 9-разрядных слов данных значение старшего разряда может

быть определено по состоянию флага RX8 (RX8n) регистра UCSRB (UCSRnB). Причем в модулях USART содержимое старшего разряда данных должно быть считано до обращения к регистру данных. Это связано с тем, что флаг RX8 (RX8n) отображает значение старшего разряда слова данных кадра, находящегося на верхнем уровне буфера приемника, состояние которого при чтении регистра данных изменится.

Если во время приема кадра была включена схема контроля четности (только USART), она вычисляет бит четности для всех разрядов принятого слова данных и сравнивает его с принятым битом четности. Результат проверки запоминается в буфере приемника вместе с принятым словом данных и стоп-битами. Наличие или отсутствие ошибки контроля четности может быть затем определено по состоянию флага UPE (UPEn). Этот флаг устанавливается в «1», если следующее слово, которое может быть прочитано из буфера, имеет ошибку контроля четности. При выключенном контроле четности флаг UPE (UPEn) всегда читается как «0».

Блок приемника модулей USART/UART имеет еще два флага, показывающих состояние обмена: флаг ошибки кадрирования FE (FEn) и флаг переполнения DOR (DORn)/OR (ORn). Флаг FE (FEn) устанавливается в «1», если значение первого стоп-бита принятого кадра не соответствует требуемому, т. е. равно «0».

Флаги DOR (DORn) в USART и OR (ORn) в UART индицируют потерю данных из-за переполнения буфера приемника. В UART флаг устанавливается в «1», если к моменту окончания приема кадра (заполнения сдвигового регистра приемника) данные предыдущего кадра не были считаны из регистра данных. В USART флаг устанавливается в «1» в случае приема старт-бита нового кадра при заполненных буфере и сдвиговом регистре приемника. Установленный флаг DOR (DORn)/OR (ORn) означает, что между прошлым байтом, считанным из регистра UDR, и байтом, считанным в данный момент, произошла потеря одного или нескольких кадров.

В модулях USART все флаги ошибок буферизуются вместе со словом данных, т. е. соответствующие разряды регистра UCSRA (UCSRnA) относятся к кадру, слово данных которого будет прочитано при следующем обращении к регистру данных UDR (UDRn). Поэтому состояние этих флагов должно быть считано перед обращением к регистру данных. Кроме того, для совместимости с будущими устройствами рекомендуется при записи в регистр UCSRA (UCSRnA) сбрасывать соответствующие этим флагам разряды записываемого значения в «0».

Для индикации состояния приемника в модулях USART/UART используется флаг прерывания «Прием завершен» RXC (RXCn) регистра UCSRA (UCSRnA). Этот флаг устанавливается в «1» при наличии в буфере приемника непрочитанных данных. В модулях UART этот флаг сбрасывается после прочтения регистра данных, а в модулях USART — при опустошении буфера (после считывания всех находящихся в нем данных).

Выключение приемника осуществляется сбросом разряда RXEN (RXENn) регистра UCSRB (UCSRnB). В отличие от передатчика приемник выключается сразу же после сброса разряда, т. е. кадр, принимаемый в этот момент, теряется. В модулях USART, кроме того, при выключении приемника очищается его буфер, т. е. теряются также все непрочитанные данные. При выключенном приемнике вывод RXD (RXDn) может использоваться как контакт ввода/вывода общего назначения.

Пример подпрограммы приема по интерфейсу USART приведен ниже. Как и в предыдущем примере, здесь используется опрос флага прерывания.

```
unsigned int USART_Receive( void )
{
unsigned char status, resh, resl;

/* Wait for data to be received */
while ( !(UCSRnA & (1<<RXCn)) );

/* Get status and 9th bit, then data  from buffer */
status = UCSRnA;
resh = UCSRnB;
resl = UDRn;

/* If error, return -1 */
if ( status & (1<<FEn)|(1<<DORn)|(1<<UPEn) )
     return -1;

/* Filter the 9th bit, then return */
resh = (resh >> 1) & 0x01;
return ((resh << 8) | resl);
}
```

Обработка прерываний

Могут возникать случаи, когда по наступлению одного из событий «прием завершен», «передача завершена» или «опустошение регистра данных» контроллер должен выполнить некоторое действие. В такой ситуации необходимо программировать специальные обработчики прерываний.

Рассмотрим пример, когда требуется реализовать устройство, которое будет посылать эхо-ответ на персональный компьютер от микроконтроллера по интерфейсу USART, т.е. при получении пакета данных по модулю USART контроллер должен переслать полученный пакет данных обратно.

Для реализации приема данных по интерфейсу USART на микроконтроллере необходимо прописать обработчик прерывания по событию «прием завершен». В теле обработчика нужно ввести временную переменную `temp` для хранения принятых данных. Далее данные из регистра UDR нужно передать в temp. В завершение данные требуется отправить обратно, записывая значение `temp` обратно в URD:

```
ISR(SIG_USART_RECV)
{
 unsigned char temp = 0;
 temp = UDR;

 UDR =temp;
 }
```

3. ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА

Лабораторный стенд состоит из персонального компьютера и трех модулей: модуля «Atmega32», модуль «Датчики», модуль «Исполнительные механизмы». В состав стенда также входит набор перемычек и соединительных проводов для подключения модулей к персональному компьютеру. Общая вид модулей стенда показана на рисунке 8.

3.1. Описание модуля «Atmega32»

Общий вид модуля представлен на рис.9.

Состав модуля:

- Высокопроизводительный RISC-процессор.
- Жидкокристаллический двухстрочного-цифро-буквенный дисплей.
- Семисегментный индикатор 4шт.
- Светодиоды для индикации логических уровней (8 шт.)
- Потенциометр для генерации аналоговых сигналов.
- Генератор импульсов на ровно 50 Гц.
- Фильтры нижних частот.
- 10-канальный генератор логических уровней.
- Кнопка сброса целевого микроконтроллера.
- Программатор целевого микроконтроллера.

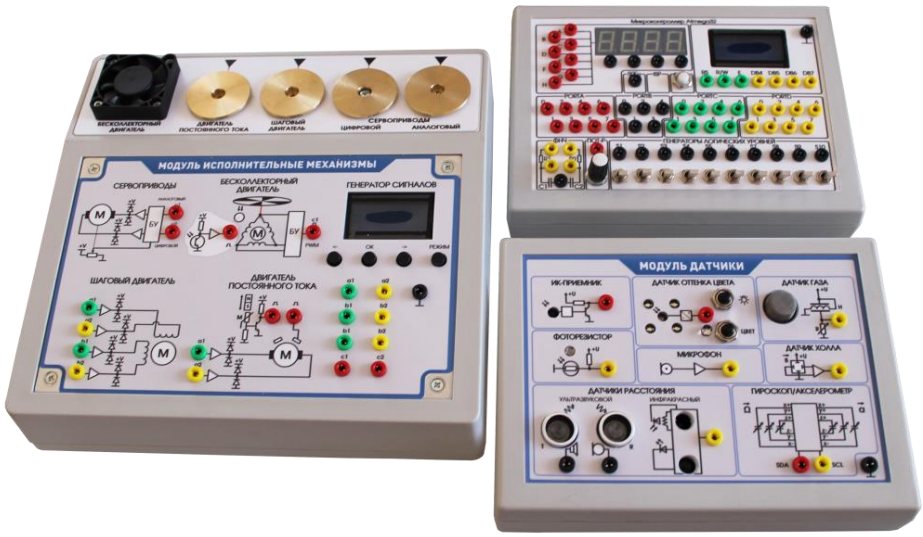

Рисунок 13 – Общая вид модулей лабораторного стенда

Спецификация МК Atmega32:

- Тактовая частота: 16 МГц
- Размер программной памяти (Flash) (Программная память): 32 кБ
- Память данных (EEPROM): 1024 Б
- Память ОЗУ (SRAM) (RAM): 2048 Б
- Кол-во таймеров (Таймеров): 3
- Кварцевый резонатор 32кГц для RTC (RTC 32кГц): Да
- Разрешение и количество каналов аналого-цифрового преобразователя (АЦП) (АЦП): 8x10-bit

- Аналоговый компаратор напряжения (кол-во) (Компаратор): 1
- Количество линий ввода-вывода (I/O): 32
- Типы последовательных интерфейсов и кол-во каналов (Последовательные интерфейсы): UART; SPI; I2C
- Встроенный температурный датчик (Датчик температуры): Нет
- Напряжение питания (min) (Uпит (min)): 4.5 В
- Напряжение питания (max) (Uпит (max)): 5.5 В
- Минимальная рабочая температура (tmin): -40 °C
- Максимальная рабочая температура (tmax): 85 °C

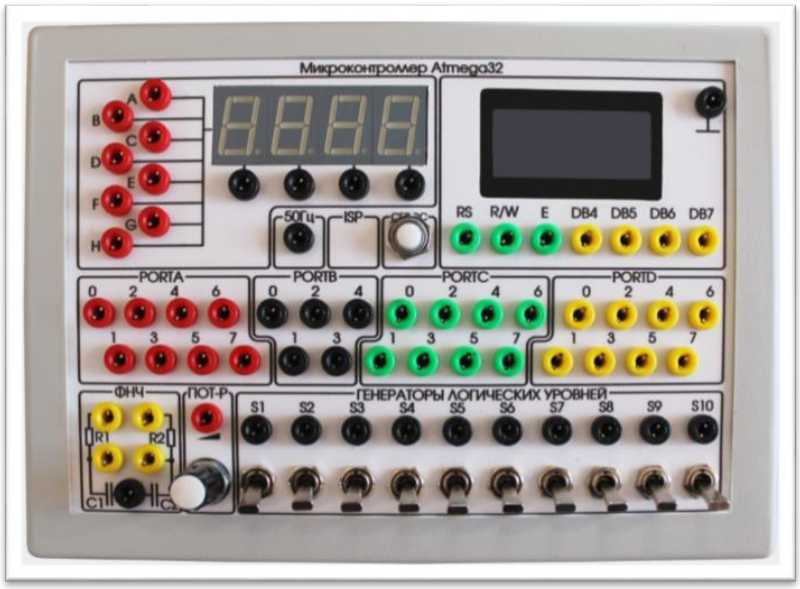

Рисунок 14 – Общий вид модуля «Микроконтроллер Atmega32

3.2. Описание программатора Pololu USB AVR Programmer

В качестве моста USB-USART в работе может использоваться программатор Pololu AVR USB (рис.7). Программатор эмулирует AVRISP v2 на виртуальном последовательном порту, делая его совместимым со стандартным программным обеспечением AVR. Кроме того, плата программатора имеет два дополнительных средства, которые позволяют упростить создание и процесс отладки проектов: последовательный порт TTL для основной связи и SLO-scope для мониторинга сигналов и уровней напряжения.

Программатор подключается к USB-порту компьютера и взаимодействует с имеющимся программным обеспечением, например, AVR Studio, через виртуальный COM-порт с помощью AVRISPV2/STK500 протокола. К целевому устройству программатор подключается посредством, входящего в комплект, 6-контактного ISP кабеля программирования.

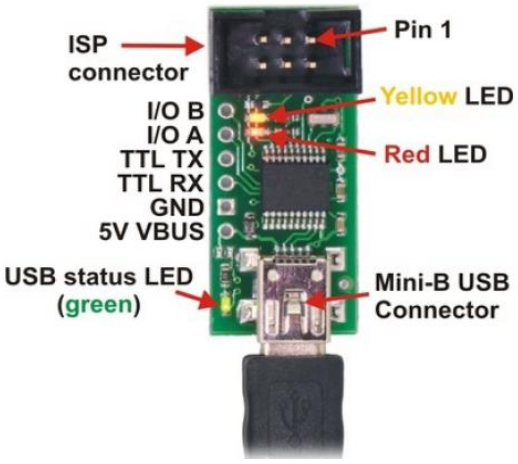

Рисунок 7 – Вид программатора Pololu USB AVR Programmer

3.3. Описание программы Pololu Serial Transmitter

Pololu Serial Transmitter – это простая утилита для Windows, которая позволяет передавать последовательности байтов по выбранному COM-порту. Версия 1.3 программы позволяет указывать скорость передачи данных, отправлять и получать любые последовательности данных (рис.8). Установленная по умолчанию программа указывается в меню «Пуск» под именем Pololu Serial Transmitter в каталоге Pololu.

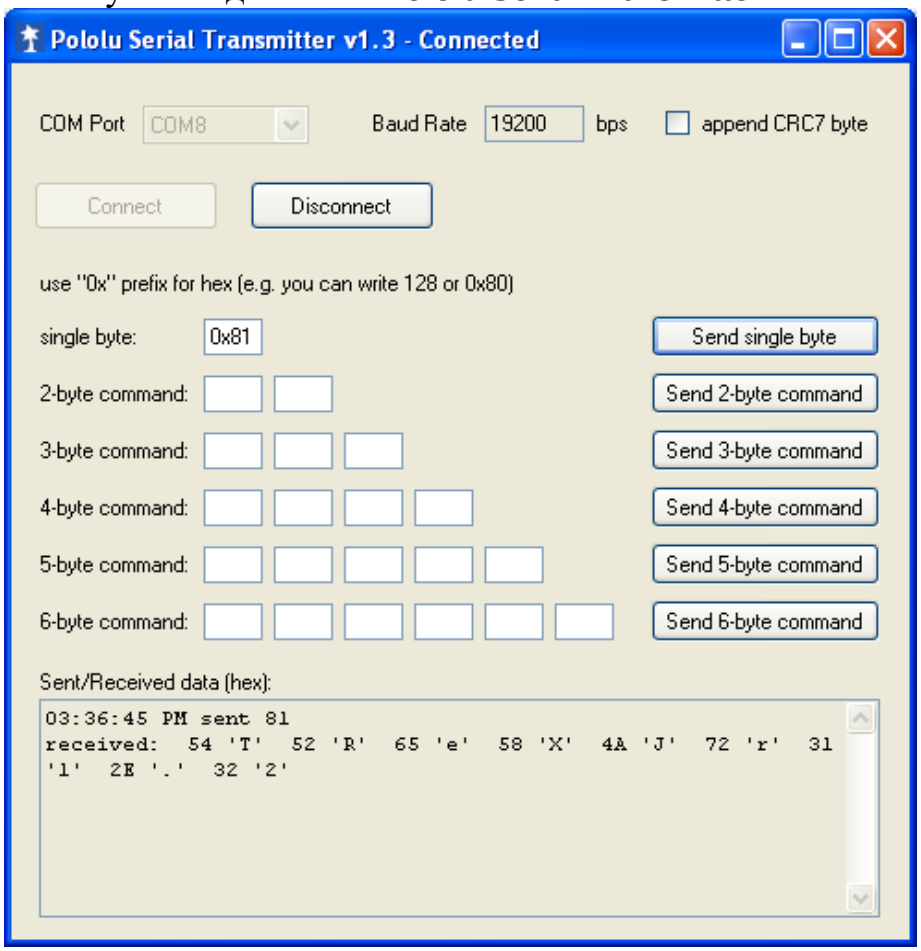

Рисунок 8 – Вид рабочего окна приложения Pololu Serial Transmitter

Программа позволяет отправить один байт (нажатием кнопки *Send single byte)* или целые пакеты до шести байт в длину (нажатием кнопок *Send 2-byte command* и т.д.). Пересылаемые с ПК данные вводятся в соответствующие поля. Значения этих данных интерпретируются как десятичные (основание-10) целые числа. Если значениям данных предшествуют префикс 0x, то в этом случае они интерпретируются в шестнадцатеричном формате. Например, чтобы отправить байт, который имеет значение 128, вы можете ввести 128 или 0x80. Журнал в нижней части рабочего окна ведет учет пакетов, которые были отправлены и получены. Этот журнал отображает каждый посланный байт как шестнадцатеричное двузначное число и каждого принятый байт как шестнадцатеричное двузначное число с указанием его символьного представления в одинарны кавычках согласно таблицы ASCII.

3.4. Средства эмуляции для выполнения работы в дистанционном формате

Проверить работу программы для микроконтроллера можно с помощью средства разработки и отладки в режиме реального времени электронных схем Proteus. Пример схемы подключения микроконтроллера и периферийных устройств в Proteus показан на рис.9.

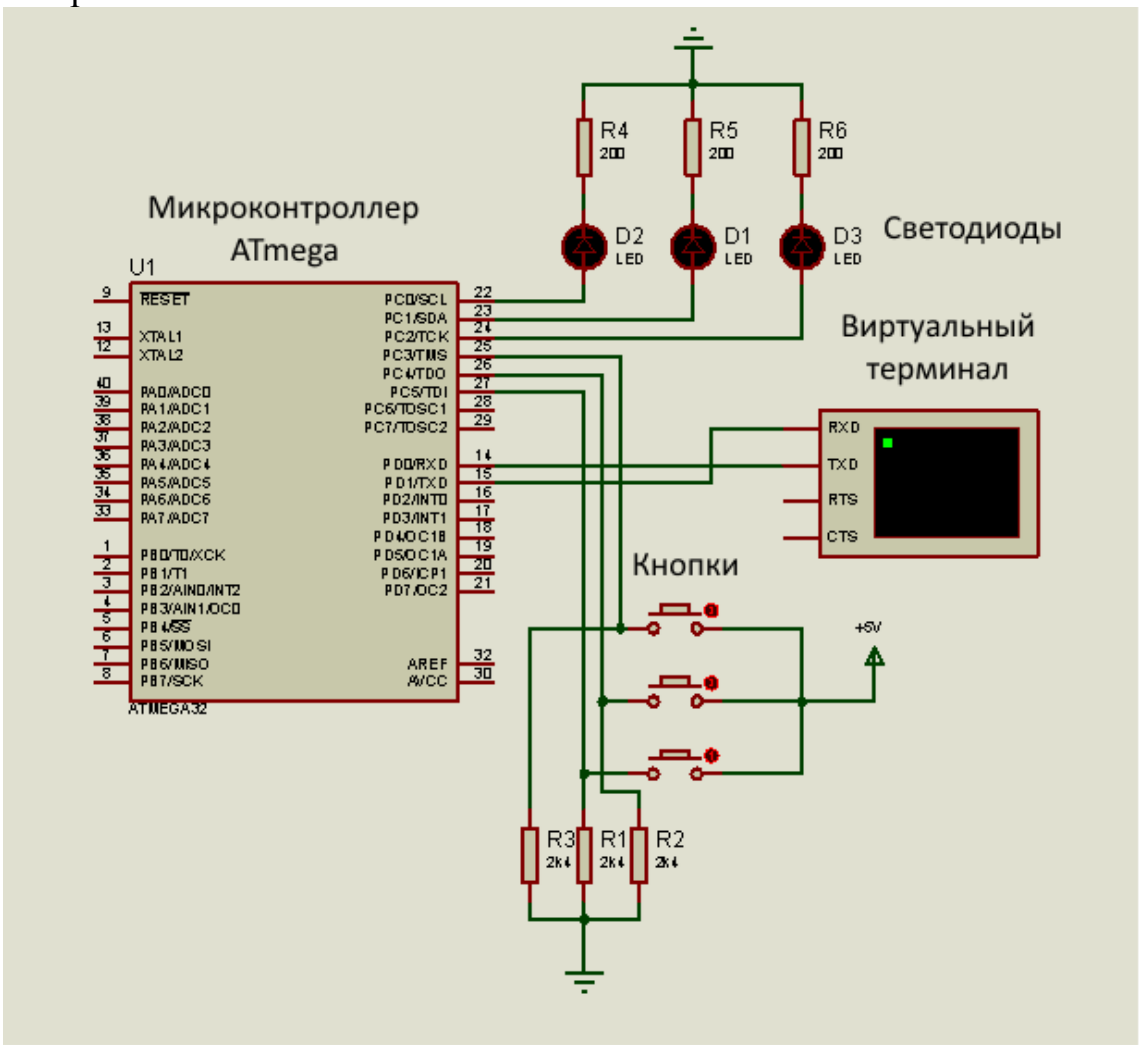

Рисунок 9 – Схема подключения контроллера и виртуального терминала в Proteus

Важно заметить, что виртуальный терминал должен иметь соответствующие режиму передачи настройки, в частности, битрейт, формат кадра передачи данных. Окно параметров терминала показано на рис.10.

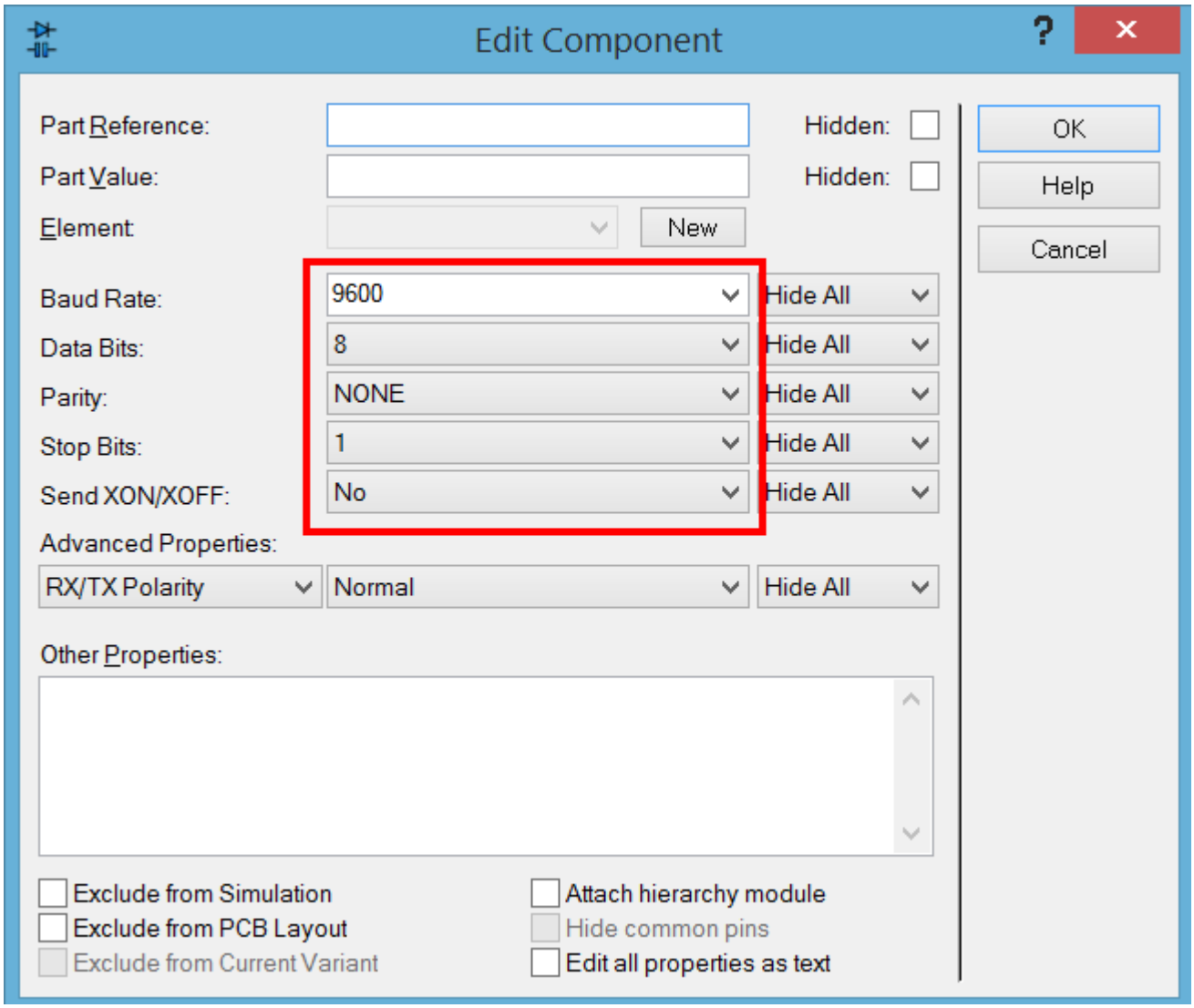

Рисунок 10 – Окно параметров виртуального терминала

Данный пример передает по последовательному интерфейсу номер одной из трех нажатых кнопок и включает соответствующий светодиод. Номер кнопки передается в текстовом сообщении «Button N», где вместо N указывается номер 1, 2 или 3.

4. ЗАДАНИЕ НА РАБОТУ

1) Изучите основные принципы программирования модуля USART микроконтроллеров семейства AVR.

2) Спроектируйте схему лабораторного стенда, в которой реализуется подключение модуля USART последовательной передачи данных микроконтроллера ATMega32.

3) Для собранного варианта лабораторного стенда необходимо разработать программу, передающую на ПК по интерфейсу USART номер нажатой на стенде

кнопки. Настройки модуля USART выбираются в соответствии с вариантом задания (табл.3).

4) Проведите отладку программы в режиме эмуляции Proteus;

5) Загрузите программу в контроллер и проверьте результаты ее работы на реальном устройстве.

6) Проанализируйте полученные результаты.

7) Подготовьте отчет о проделанной работе.

Таблица 3 – Варианты задания

Номер варианта	Тактовая частота, МГц	Скорость передачи, bps	Бит чётности	Количество стоп-битов
1	1	9600	None	1
2	2	14400	Odd	2
3	4	19200	Even	1
4	8	28800	None	2
5	1	38400	Odd	1
6	2	56000	Even	2
7	4	2400	None	1
8	8	4800	Odd	2
9	1	76800	Even	1
10	2	115200	None	2
11	4	9600	Odd	1
12	8	14400	Even	2
13	1	19200	None	1
14	2	28800	Odd	2
15	4	38400	Even	1
16	8	56000	None	2
17	1	2400	Odd	1
18	2	4800	Even	2
19	4	76800	None	1
20	8	115200	Odd	2
21	1	14400	Even	1
22	2	19200	None	2
23	4	28800	Odd	1
24	8	38400	Even	2
25	1	56000	None	1
26	2	2400	Odd	2
27	4	4800	None	1
28	8	76800	Odd	2

5. СОДЕРЖАНИЕ ОТЧЕТА И ПОРЯДОК ЗАЩИТЫ РАБОТЫ

Выполнение и защита лабораторной работы производится каждым студентом индивидуально. Защита результатов лабораторной работы осуществляется при наличии работающей программы и полностью оформленного отчета.

Отчет должен включать в себя следующие разделы

- титульный лист;
- цель работы,
- постановка задачи.
- схема подключения внешних устройств к контроллеру, схема программы;
- текст программы;
- результаты работы программы;
- осциллограммы сигналов с линий RX и TX;
- выводы.

Защита работы состоит в следующем:

- предъявление работающей программы на симуляторе и стенде;
- предъявление отчета, оформленного в соответствии с требованиями;
- ответы на вопросы по теоретической и практической части работы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое модуль USART?
2. Чем отличается синхронный режим работы от асинхронного?
3. Какие основные линии необходимы для работы в асинхронном режиме?
4. Для чего используются старт- и стоп-биты?
5. Сколько информационных битов может содержать один кадр USART?
6. Что такое бит четности?
7. Какие регистры используются для настройки модуля USART в микроконтроллере ATmega?
8. Что такое битрейт?
9. Как настроить битрейт в микроконтроллере ATmega?
10. По каким событиям модуль USART микроконтроллера ATmega может генерировать прерывания?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL, 5-е изд., стер. — М.: Издательский дом «Додэка-XXI», 2008. — 560 с.

2. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров. - Изд. 2-е, перераб. и доп. - Киев: МК-Пресс, 2012. - 534 с.

3. Мортон. Дж.Микроконтроллеры AVR. Вводный курс. / Пер. с англ. – М.: Издательский дом «Додэка-XXI», 2006. – 272 с.

4. http://atmel.ru/ [интернет ресурс]

ПРИЛОЖЕНИЕ А

Справочные данные о модуле USART

Таблица А.1 – Определение размера слова данных

UCSZ2 (UCSZn2)	UCSZ1 (UCSZn1)	UCSZ0 (UCSZn0)	Размер слова данных
0	0	0	5 разрядов
0	0	1	6 разрядов
0	1	0	7 разрядов
0	1	1	8 разрядов
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Зарезервировано
1	1	1	9 разрядов

Примечание: n = 0, 1 или 2.

Таблица А.2 – Конфигурация модуля USART разрядами UMSEL1:0

UMSEL1	UMSEL0	Режим работы
0	0	Асинхронный
0	1	Синхронный
1	0	зарезервировано
1	1	Ведущий SPI

Таблица А.3 – Формулы определения скорости передачи данных

Режим работы	Битрейт	Значение регистра UBRR0
Асинхронный обычный (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{(UBRR+1)16}$	$UBRR = \frac{f_{OSC}}{16 \cdot BAUD} - 1$
Асинхронный с удвоенной скоростью (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{(UBRR+1)8}$	$UBRR = \frac{f_{OSC}}{8 \cdot BAUD} - 1$
Синхронный режим ведущего	$BAUD = \frac{f_{OSC}}{(UBRR+1)2}$	$UBRR = \frac{f_{OSC}}{2 \cdot BAUD} - 1$

Таблица А.4 – Значения UBRR для ряда частот тактовых резонаторов

Baud rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max.(1)	0.5Mbps		1Mbps		691.2Kbps		1.3824Mbps		921.6Kbps		1.8432Mbps	

Baud rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

Заказ № _____ от «___» _____ 2020г.
Изд-во СевГУ