Лабораторная работа 5

РАБОТА С ВНЕШНИМИ ИНТЕРФЕЙСАМИ: UART И I2C

Цель работы

Изучение структуры данных, пересылаемых через последовательный порт. Получение практических навыков программирования встроенного в микроконтроллер UART модуля. Получение практических навыков по работе с интерфейсом I^2C и принципами работы памяти EEPROM.

Теоретические сведения

Внешние интерфейсы, последовательная передача данных

Устройства на базе микроконтроллеров часто используются в качестве периферийных устройств сбора и первичной обработки информации, работающих в составе более сложной информационной системы, включающей в себя большую ЭВМ, предназначенную для решения более сложных задач анализа и хранения собранной информации.

Существует несколько способов подключения устройств к ЭВМ, в зависимости от требуемой скорости передачи. Подключение к IBM PC устройств с относительно низким быстродействием осуществляется через последовательный порт передачи данных (COM-порт).

Главное различие последовательной и параллельной передачи данных заключается в том, что при последовательной передаче информации в единичный интервал времени передаётся только один бит. Для параллельной передачи данных требуется наличие проводников в количестве, равном количеству одновременно передаваемых бит.

Достоинство последовательной передачи данных заключается в малом количестве проводов, необходимых для организации передачи, а недостаток – это относительно низкая скорость передачи информации.

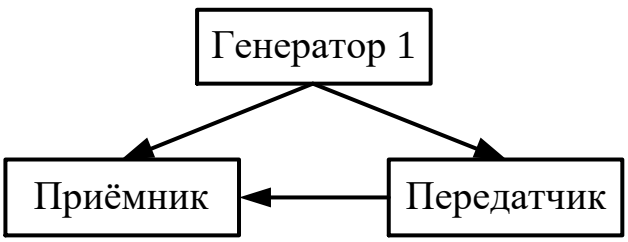

Рис. 1. Схема соединения при синхронной передаче

Существует две разновидности последовательного способа передачи данных – синхронный и асинхронный.

Необходимое условие синхронной передачи данных – наличие одного синхрогенератора для приёмника и передатчика.

Достоинства: высокая скорость передачи.

Недостаток: необходимость специального синхропровода для передачи синхросигнала (для синхронной работы приёмника и передатчика).

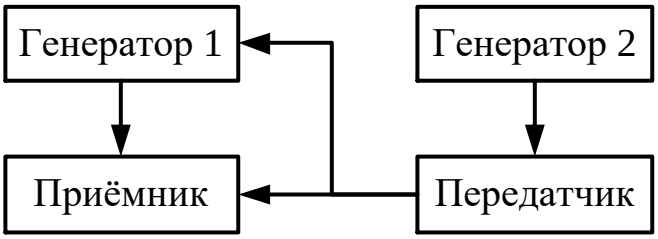

Рис. 2. Схема соединения при асинхронной передаче

Генератор приёмника периодически подстраивается под генератор передатчика по специальным сигналам синхронизации. Для подстройки используют специальную структуру передаваемых сигналов. Обычно синхросигнал предшествует началу передачи блока данных и называется стартовым битом.

Данные в последовательный порт передаются пакетами. Структура пакета данных изображена на рисунке.

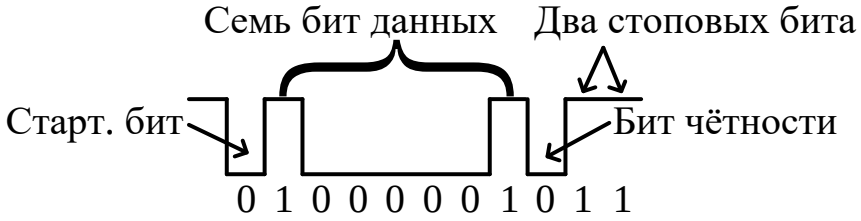

Рис. 3. Структура пакета

Из рисунка видно, что исходное состояние линии последовательной передачи данных – это уровень логической единицы. Стартовый бит служит для формирования перехода из единицы в ноль, который означает начало передачи данных. Далее передаются биты данных (от 5 до 8 в зависимости от выбранного формата данных), далее возможно наличие бита проверки на чётность (на рисунке отсутствует), а затем один или два стоповых бита, завершающих передачу пакета и устанавливающих уровень линии в единицу до прихода следующего стартового бита (следующего пакета данных). Структура пакета, формируемого передатчиком, должна

быть известна приёмнику, иначе он не сможет правильно организовать приём.

Другая важная характеристика – это скорость передачи данных. Она должна быть одинаковой для передатчика и приёмника. Скорость последовательной передачи измеряется в бодах. Боды – это количество бит (как информационных так и вспомогательных), передаваемых за секунду.

Последовательный порт персонального компьютера поддерживает следующие скорости передачи данных: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бод.

Главная трудность передачи данных через последовательный порт связана с эффектом накапливания погрешности рассинхронизации с течением времени. Он иллюстрируется рисунком, где изображены передаваемые контроллером данные. Жирными точками отмечены моменты возможного изменения сигнала.

Рис. 4. Проблема рассинхронизации

Сигналы синхронизации последовательного порта должны совпадать с серединой каждого бита информации. Пунктиром показаны идеальные синхроимпульсы.

Видно, что в середине стартового бита включается генератор синхронизирующих сигналов приёмника (COM-порта). На каждом синхросигнале считывается значение с входа порта. Если длительность передачи контроллером одного бита пакета отличается от периода синхросигналов, появляется погрешность. С каждым следующим передаваемым битом погрешность накапливается и возможна ошибка. Например, на рисунке 5-й синхроимпульс приходится на 6-й бит пакета.

Аппаратный модуль UART

В состав большинства контроллеров AVR входит модуль UART, который позволяет принимать и получать байты данных автоматически. Управление модулями осуществляется через специальные регистры.

Основным регистром для общения с модулем UART является *UDR* (UART Data Register). На самом деле это два разных регистра, имеющих один адрес. При записи данные попадает в один регистр (регистр передатчика), а при чтении данные вычитываются из другого регистра (регистр приёмника).

Когда байт пришёл в регистр UDR, может сработать прерывание по завершению приёма (которое перед этим нужно разрешить), которое вызывается сразу же, как только приёмник получил все биты данных.

Поскольку передача во времени происходит не мгновенно, то перед записью очередного байта нужно дождаться окончания передачи предыдущего. О том, что *UDR* пуст и готов к приёму нового байта, сигнализирует бит *UDRE*, он же вызовет аппаратное прерывание по опустошению буфера.

Таким образом, всё управление UART можно осуществлять либо при помощи прерываний (что предпочтительнее), либо в цикле проверять значения управляющих регистров.

Настройка модуля UART

Все настройки приёмопередатчика хранятся в регистрах конфигурации. Это *UCSRA, UCSRB* и *UCSRC*. Скорость задаётся в паре регистров *UBBRH:UBBRL*.

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
R	R/W	R	R	R	R	R/W	R/W

Рис. 5. Регистр UCSRA (USART Control and Status Register A)

Регистр *UCSRA* содержит биты *RXC* и *TXC*, это флаги завершения приёма и передачи соответственно. RXC устанавливается, когда непрочитанный байт находится в регистре UDR (заполнение регистра приёмника), а TXC устанавливается, когда последний стоп-бит прошёл, а новое значение в UDR не поступило (опустошение регистра передатчика). Также одновременно с этими флагами вызывается прерывание (если оно было разрешено). Сбрасываются биты RXC и TXC аппаратно – принимающий после чтения из регистра UDR, передающий при переходе на прерывание, либо программно (чтобы сбросить флаг программно, в него надо записать единицу).

Бит *UDRE* сигнализирует о том, что регистр UDR передатчика пуст и в него можно записать новый байт. Сбрасывается он аппаратно после засылки в UDR какого-либо байта. При опустошении регистра передатчика может генерироваться прерывание.

Бит *U2X* – бит удвоения скорости при работе в асинхронном режиме. Его надо учитывать при расчёте значения в *UBBRH:UBBRL*

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

Рис. 6. Регистр UCSRB (USART Control and Status Register B)

Регистр *UCSRB* содержит биты *RXEN* и *TXEN* – разрешение приёма и передачи. Для работы UART их нужно установить, т.к. при их сбросе выводы UART работают как обычные пины ввода-вывода.

Биты *RXCIE, TXCIE, UDRIE* разрешают прерывания по завершению приёма, передачи и опустошению буфера передачи UDR.

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Рис. 7. Регистр UCSRC (USART Control and Status Register C)

Регистр *UCSRC* в разных моделях AVR может быть организован по-разному. В общем случае регистр UCSRC задаёт количество стоп-битов, наличие бита чётности и способ его проверки. Если оставить все по умолчанию, то получится стандартный режим – 1 стоп-бит, без бита чётности. Надо только задать количество бит в посылке (от 5 до 9). Делается это *битами UCSZ0, UCSZ1, UCSZ2* (бит UCSZ2 находится в регистре UCSRB). Для стандартного формата пакета в 8 бит нужно установить комбинацию UCSZ2 UCSZ1 UCSZ0 в значение 011.

Скорость передачи и приёма задаётся в регистровой паре *UBBRx*. Вычисляется требуемое значение этой регистровой пары по формуле:

UBBR=XTAL/(16*baudrate)-1 для U2X=0

UBBR=XTAL/(8*baudrate)-1 для U2X=1

Интерфейс I2C

I_2C – двухпроводной интерфейс, разработанный и запатентованный корпорацией Philips. Последняя версия стандарта представлена в 2001 году. С 2006 года за использование интерфейса не требуется платить патентные отчисления, что, вкупе с простотой реализации делает протокол популярным для связи с периферийными устройствами.

Основные термины:

– *Передатчик* – устройство, передающее данные по шине;

– *Приёмник* – устройство, получающее данные с шины;

– *Master (ведущий)* – устройство, которое инициирует передачу и формирует тактовый сигнал;

– *Slave (ведомый)* – устройство, к которому обращается «Master»;

– *Multi-Master* – режим работы шины I2C с более чем одним «Master»;

– *Арбитраж* – процедура, гарантирующая, что только один «Master» управляет шиной;

– *Синхронизация* – процедура синхронизации тактового сигнала от двух или более устройств.

Интерфейс I2C. Физический уровень.

Данные передаются по двум проводам – линия данных (SDA) и линия синхронизации (SCL). Всего на одной двухпроводной шине может быть до 127 устройств. Такты генерирует master.

Передача сигналов осуществляется установкой линии в ноль, в единицу линия возвращается сама. Линии подключены к Vcc через подтягивающие резисторы (рекомендуется использовать резисторы номиналом в 10 кОм). Чем больше сопротивление резистора, тем дольше линия восстанавливается в единицу и тем сильней заваливаются фронты импульсов, и падает скорость передачи.

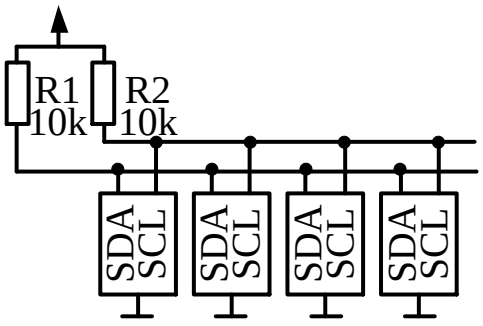

Рис. 8. Схема подключения устройств

Интерфейс I2C. Начало передачи

Процедура обмена начинается с того, что ведущий формирует состояние СТАРТ – ведущий генерирует переход сигнала линии SDA из единицы в ноль при единице на линии SCL. Этот переход воспринимается всеми устройствами, подключёнными к шине как признак начала процедуры обмена.

Генерация синхросигнала – это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

Процедура обмена завершается тем, что ведущий формирует состояние СТОП – переход состояния линии SDA из нуля в единицу при единице линии SCL.

Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.

При передаче посылок по шине I_2C каждый ведущий генерирует свой синхросигнал на линии SCL. После формирования состояния СТАРТ ведущий переводит состояние линии SCL в ноль и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено.

Спецификация шины I_2C разрешает изменения на линии SDA только при нуле на линии SCL. Данные действительны и должны оставаться стабильными только во время состояния единицы на линии SCL.

Для подтверждения приёма байта от ведущего – передатчика ведомым – приёмником в спецификации протокола обмена по шине I_2C вводится специальный бит подтверждения, выставляемый на шину SDA после приёма 8 бита данных.

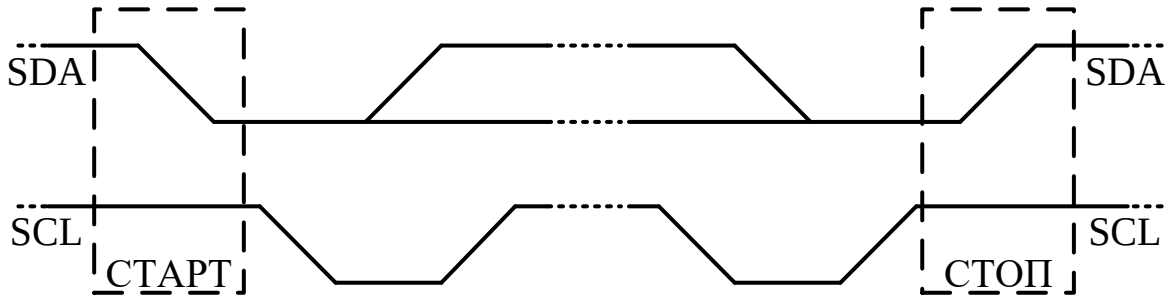

Рис. 9. Сигналы СТАРТ и СТОП

Интерфейс I2C. Подтверждение (ACK)

Передача 8 бит данных от передатчика к приёмнику завершается дополнительным циклом (формированием 9-го тактового импульса линии SCL), при котором приёмник выставляет нулевой уровень сигнала на линии SDA, как признак успешного приёма байта.

Подтверждение при передаче данных обязательно. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает линию SDA (переводит в единицу) на время синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение единичного уровня синхроимпульса подтверждения в стабильном нулевом уровне состоянии.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), на линии данных должна быть выставлена единица. После этого ведущий может выдать сигнал СТОП для прерывания пересылки данных.

Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путём не подтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал СТОП или повторить сигнал СТАРТ.

Адресация в шине I_2C

Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу. **Для микросхемы внешней EEPROM на плате easyAVR используется адрес 0x51**.

Для выбора приёмника сообщения ведущий использует уникальную адресную компоненту в формате посылки. При использовании однотипных устройств, ИС часто имеют дополнительный селектор адреса, который может быть реализован как в виде дополнительных цифровых входов селектора адреса, так и в виде аналогового входа. При этом адреса таких однотипных устройств оказываются разнесены в адресном пространстве устройств, подключённых к шине. В обычном режиме используется 7-битная адресация.

Процедура адресации на шине I_2C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый адресуется ведущим для

проведения цикла обмена. Исключение составляет адрес «Общего вызова», который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако устройства, которые могут обрабатывать «общий вызов», на практике встречаются редко.

Первые семь битов первого байта образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных. «Ноль» означает, что ведущий будет записывать информацию в выбранного ведомого. «Единица» означает, что ведущий будет считывать информацию из ведомого.

После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Комментированный листинг программы на языке ассемблера.
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Укажите назначение конфигурационных регистров модуля UART?
2. Какими способами можно осуществлять последовательную передачу данных в микроконтроллерах AVR?
3. Какие события генерируются модулем UART?
4. Каков размер адресного пространства шины I2C?
5. В каком режиме осуществляется связь по шине I2C, каковы могут быть роли устройств на шине?

Варианты заданий

Номер варианта	Задание
1	Настройки UART-интерфейса: скорость 4800 бод, контроль чётности – even, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия:

<table>
<tr><td></td><td colspan="2">1. хранить в ОЗУ МК (в переменных типа uint8) переменные {a, b, c, d, e}. После запуска МК до ввода новых значений в качестве значений переменных устанавливается ноль;
2. обеспечивать постоянное отображение значения одной из переменных на четырёх семисегментных индикаторах в виде «<имя_переменной>.<значение>», например, «a.123»;
3. Выполнять обработку внешних прерываний:
3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: a→b→c→d→e→a→…) переменной на семисегментном индикаторе;
3.2. при срабатывании прерывания INT1 выполнять передачу значения переменной, отображаемой в данный момент на семисегментном индикаторе, в терминал ПК (по UART-интерфейсу) в виде «<имя_переменной>=<значение>», например, «a=123».
4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC):
4.1. ввод новых значений для переменных. Формат ввода: «<имя_переменной>=<значение>», например, «a=123»;</td></tr>
<tr><td></td><td>а</td><td>4.2. выполнение арифметических операций {+, -, *} с переменными {a, b, c, d, e}. В ответ МК отправляет результат операции. Формат ввода:
«=<имя_переменной><оператор><имя_переменной>», например, «=a+b»;</td></tr>
<tr><td></td><td>б</td><td>4.3. выполнение сдвиговых и логических операций {>>, <<, |, &} с переменными {a, b, c, d, e}. В ответ МК отправляет результат операции. Формат ввода:
«=<имя_переменной><оператор><имя_переменной>», например, «=a<<c»;</td></tr>
<tr><td></td><td>в</td><td>3.3. выполнять передачу значений всех переменных в терминал ПК (по UART-интерфейсу) при срабатывании прерывания INT0 и нажатой кнопке PD0 в виде «<имя_переменной>=<значение>,…», например, «a=123,b=456,c=789,d=159».</td></tr>
<tr><td>2</td><td colspan="2">Настройки UART-интерфейса: скорость 9600 бод, контроль чётности – odd, один стоповый бит, 8 значащих битов в пакете.</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">Программа для МК должна выполнять следующие действия:
1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI):
1.1. ввод строковых переменных {a, b, c, d}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале:
«имя_строковой_переменной=значение»;
1.2. ввод 8-разрядных неотрицательных целочисленных переменных {x, y}. Формат запроса в терминале:
«имя_целочисленной_переменной=значение»;
2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК:
2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале:
«=имя_переменной»;
2.2. вывод результата операции со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «=название_команды(параметры)».
3. выполнение операции со строковыми переменными, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале:
«имя_переменной=название_команды(параметры)».
В качестве параметров могут выступать только целочисленные и строковые переменные;</td></tr>
<tr><td></td><td>а</td><td>Реализуемые операции:
- x=strcmp(s1,s2) – сравнение строк s1 и s2, результат 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2.
- s0=left(s1,x) – копирование первых x элементов строки s1 в переменную s0</td></tr>
<tr><td></td><td>б</td><td>Реализуемые операции:</td></tr>
</table>

		- x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=xor(s1,x) – запись в переменную s0 результата поразрядного исключающего ИЛИ всех байтов строки s1 с числом x;
	в	Реализуемые *операции*: - x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число); - s0=substr(s1,x,y) – копирование фрагмента строки s1, начинающегося с позиции x и имеющего длину y, в переменную s0;
3	Настройки UART-интерфейса: скорость 28800 бод, контроль чётности выключен, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 16-разрядных (от -2^{15} до 2^{15}) целочисленных переменных $\{a, b, c\}$ в шестнадцатеричной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;	
	а	1.2. ввод логических выражений $\{@, \#, \$\}$, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов $\{+, -, *\}$. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;
	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти

<table>
<tr><td></td><td></td><td>непосредственно перед вычислением). Формат запроса в терминале «=имя_переменной-выражения».</td></tr>
<tr><td>4</td><td colspan="2">Настройки UART-интерфейса: скорость 38400 бод, контроль чётности – even, один стоповый бит, 8 значащих битов в пакете.
Программа для МК должна выполнять следующие действия:
1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI):
1.1. ввод строковых переменных {@, #, $}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале:
«имя_строковой_переменной=значение»;
1.2. ввод 8-разрядных неотрицательных целочисленных переменных {a, b, c}. Формат запроса в терминале:
«имя_целочисленной_переменной=значение»;
2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК:
2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале:
«=имя_переменной»;
2.2. вывод результата операции со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «=название_команды(параметры)».
3. выполнение операции со строковыми переменными, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале:
«имя_переменной=название_команды(параметры)».
В качестве параметров могут выступать только целочисленные и строковые переменные;</td></tr>
<tr><td></td><td>а</td><td>Реализуемые операции:
- x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число);</td></tr>
</table>

		- s0=replace(s1,s2,s3) – копирование строки s1 в переменную s0, с заменой всех вхождений подстроки s2 на подстроку s3. Если при замене получается строка длиннее 20 символов, то остаются только первые 20 символов;
	б	Реализуемые *операции*: - x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=inv(s1) – копирование инвертированной строки s1 в переменную s0. Пример: inv(“123abc”) = “cba321”;
	в	Реализуемые *операции*: - x=strcmp(s1,s2) – сравнение строк s1 и s2 – 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2; - s0=trim(s1) – копирование строки s1 в переменную s0, предварительно удаляя все пробелы (код 0x32) в начале и конце строки;
5	Настройки UART-интерфейса: скорость 57600 бод, контроль чётности – odd, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 8-разрядных (от -2^7 до 2^7) целочисленных переменных {*x, y, z*} в шестнадцатеричной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;	
	а	1.2. ввод логических выражений {*A, B, C, D*}, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов {& – *побитовое AND*, | – *побитовое OR*, ^ – *побитовое XOR*, ~ – *инверсия*}. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;

	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти непосредственно перед вычислением). Формат запроса в терминале «*=имя_переменной-выражения*».
6	Настройки UART-интерфейса: скорость 4800 бод, контроль чётности выключен, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменных типа unsigned char *) строк {*x, y, z*}, состоящих из комбинаций следующих символов {<пробел>,0,1,2,3,4,5,6,7,8,9, a,b,c,d,e,f}, длины строк не должны превышать 20 символов. После запуска МК до ввода значений в качестве значения {*x, y, z*} устанавливаются пустые строки; 2. обеспечивать постоянное отображение одной из переменных {*x, y, z*} в виде бегущей строки на четырёх семисегментных индикаторах. При отображении переменной *x* должен светиться светодиод PD7, при отображении переменной *y* – PD6, *z* – PD5; 3. Выполнять обработку внешних прерываний: 3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: *x*→*y*→*z*→*x*→…) переменной на семисегментном индикаторе. 3.2. при срабатывании прерывания INT1 выполнять передачу значения переменной, отображаемой в данный момент на семисегментном индикаторе, в терминал ПК (по UART-интерфейсу) в виде «*<имя_переменной>=<значение>*», например, «*x*=123». 4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 4.1. считывать новое значение переменных {*x, y, z*} по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC) в виде «*<имя_переменной>=<строка>*», например, «x=12 34 56 78 90 ab cd ef».	

	а	4.2. считывать строку *a*, которая будет дописана в конец текущей строки *s*: *s = strcat(s, a)*. Если при объединении получается строка длиннее 20 символов, то в переменную *s* сохраняются только первые 20 символов. Формат ввода: «+<*дописываемая строка*>», например, «+extra_string»;
	б	3.3. при срабатывании прерывания INT0 и зажатой кнопке PD0 изменить направление движения бегущей строки
	в	3.4. при срабатывании прерывания INT1 и зажатой кнопке PD0 очистить содержимое текущей строки
7	Настройки UART-интерфейса: скорость 9600 бод, контроль чётности – even, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменных типа uint16) переменные {*a, b, c*}. После запуска МК до ввода новых значений в качестве значений переменных устанавливается ноль; 2. обеспечивать постоянное отображение значения одной из переменных в шестнадцатеричной системе счисления на четырёх семисегментных индикаторах в виде: «<*имя_переменной*>.<*значение*>», например, «a. F2»; 3. Выполнять обработку внешних прерываний: 3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: *a→b→c→a→…*) переменной на семисегментном индикаторе; 3.2. при срабатывании прерывания INT1 сбросить значение всех переменных в ноль. 4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 4.1. ввод новых значений для переменных. Формат ввода: «<*имя_переменной*>=<*значение*>», например, «a=123»;	
	а	3.3. выполнять передачу значений всех переменных в терминал ПК (по UART-интерфейсу) при срабатывании прерывания INT0 и зажатой кнопке PD0 в виде «<*имя_переменной*>=<*значение*>,…», например, «a=123,b=456,c=789,d=159».

	б	4.2. выполнение арифметических операций {+, -} с переменными {*a, b, c*}. В ответ МК отправляет результат операции. Формат ввода: «=<*имя_переменной*><*оператор*><*имя_переменной*>», например, «=a+b»;
	в	4.3. выполнение сдвиговых и логических операций {|, &, ^} с переменными {*a, b, c, d, e*}. В ответ МК отправляет результат операции. Формат ввода: «=<*имя_переменной*><*оператор*><*имя_переменной*>», например, «=a<<c»;
8	Настройки UART-интерфейса: скорость 28800 бод, контроль чётности – odd, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменной типа unsigned char *) строку *s*, состоящую из печатных ASCII-символов (коды символов должны быть больше 31), длина строки не должна превышать 100 символов. После запуска МК до ввода новой строки в качестве значения *s* устанавливается пустая строка; 2. Выполнять обработку внешних прерываний: 2.1. при срабатывании прерывания INT0 выводить строку *s* в терминал ПК (по UART-интерфейсу); 2.2. при срабатывании прерывания INT1 выполнять очистку строки *s*; 3. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 3.1. считывать новое значение строки *s*. Формат ввода в терминал: «*<*новая строка*>», например, «*new_string»;	
	а	3.2. считывать целое беззнаковое число *y*, для выполнения поразрядного исключающего ИЛИ с каждым байтом строки *s* с сохранением результата в переменную *s*. Формат ввода: «^*number*», например, «^26»; 3.3. считывать строку *x*, которая будет дописана в конец текущей строки *s*: *s = strcat(s, x)*. Если при объединении получается строка длиннее 100 символов, то в переменную *s* сохраняются только первые 100 символов. Формат ввода: «+<*дописываемая строка*>», например, «+extra_string»;

	б	3.4. считывать строку *x*, для который будет выполнен поиск всех вхождений подстроки *x* в строке *s* с удалением подстроки и схлопыванием строки *s*. Формат ввода: «-<*удаляемая_строка*>», например, «-delete_me»; 3.5. считывать целое беззнаковое число *y*, показывающее номер символа строки *s*, значение которого необходимо вернуть в терминал в виде цифр в шестнадцатеричном представлении – *0xXX*. Формат ввода: «*?number*», например, «?26»;
	в	3.6. считывать целое беззнаковое число *y*, показывающее количество символов, соответствующее новой длине переменной *s*, все символы за пределами обновляемой строки заменяются нулём, все символы, добавляемые к строке, заполняются числом 0x32 (пробелом). Формат ввода: «*/number*», например, «/26»; 3.7. считывать строку *x*, которая будет использоваться в качестве подстроки при поиске в переменной *s*. Если подстрока *x* найдена, то в терминал возвращается номер первого символа первого вхождения подстроки в строке, если не найдена – число *0xFF*. Формат ввода: «*#<подстрока>*», например, «#find_substring»;
9		Настройки UART-интерфейса: скорость 38400 бод, контроль чётности выключен, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 16-разрядных (от -2^{15} до 2^{15}) целочисленных переменных $\{a, b, c, d, e\}$ в десятичной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;
	а	1.2. ввод логических выражений $\{X, Y, Z\}$, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов {lr – логический сдвиг

		вправо, ll – логический сдвиг влево, rr – циклический сдвиг вправо, rl – циклический сдвиг влево, & – *AND*, | – *OR* }. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;
	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти непосредственно перед вычислением). Формат запроса в терминале «*=имя_переменной-выражения*».
10		Настройки UART-интерфейса: скорость 57600 бод, контроль чётности – even, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод строковых переменных {*x*, *y*, *z*}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале: «*имя_строковой_переменной=значение*»; 1.2. ввод 8-разрядных неотрицательных целочисленных переменных {*A*, *B*, *C*}. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале: «*=имя_переменной*»; 2.2. вывод результата *операции* со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «*=название_команды(параметры)*». 3. выполнение *операции* со строковыми переменными, значения задействованных в операции целочисленных и строковых

	переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале: «*имя_переменной=название_команды(параметры)*». В качестве параметров могут выступать только целочисленные и строковые переменные;	
	а	Реализуемые *операции*: - x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=xor(s1,x) – запись в переменную s0 результата поразрядного исключающего ИЛИ всех байтов строки s1 с числом x;
	б	Реализуемые *операции*: - x=strcmp(s1,s2) – сравнение строк s1 и s2 – 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2; - s0=strcat(s1,s2) – объединение строк s1 и s2 и помещение результата в переменную s0. Если при объединении получается строка длиннее 20 символов, то остаются только первые 20 символов;
	в	Реализуемые *операции*: - x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число); - s0=strcpy(s1) – копирование строки s1 в переменную s0;