Министерство образования и науки Украины
Национальный технический университет
«Харьковский политехнический институт»

Методические указания
к лабораторной работе № 8

**«Связь микроконтроллера с компьютером
по каналу RS-232»**

по курсу «*Интерфейсы компьютерных систем управления*»

Харьков 2006

Министерство образования и науки Украины

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 8

«СВЯЗЬ МИКРОКОНТРОЛЛЕРА С КОМПЬЮТЕРОМ ПО КАНАЛУ RS-232»

ПО КУРСУ «ИНТЕРФЕЙСЫ КОМПЬЮТЕРНЫХ СИСТЕМ УПРАВЛЕНИЯ»

для студентов специальности 7.0910002

Утверждено
редакционно-издательским
советом университета,
протокол № 2 от 02.06.04

Харьков НТУ «ХПИ» 2006

Методические указания к лабораторной работе «Связь микроконтроллера с компьютером по каналу RS-232» по курсу «Интерфейсы компьютерных систем управления» для студентов специальности 7.0910002 / Сост. Д.А. Лунин, М.В. Гунбин. – Харьков: НТУ «ХПИ» 2006. – 13 с.

Составители: Д.А. Лунин, М.В. Гунбин

Рецензент А.В. Ивашко

Кафедра автоматики и управления в технических системах

Цель работы: изучить принципы разработки программ для микроконтроллера и компьютера, связь между которыми выполнена по каналу RS-232 в соответствии со стандартом RS-232.

Основные сведения.

Для поддержки связи по каналу RS-232 в используемом микроконтроллере предусмотрена аппаратная функция UART (Universal Asynchronous Receiver and Transmitter – универсальный асинхронный приемопередатчик). Компьютер для поддержки связи снабжен COM-портом. Поэтому для описания работы компьютера и микроконтроллера в режиме связи используется разная терминология (UART, RS-232, COM-порт).

1. Схема контроллера, обеспечивающая связь с COM-портом компьютера.

Канал RS-232 предназначен для обмена информацией между двумя устройствами как в синхронном, так и в асинхронном режиме.

В компьютере для организации COM-порта используется только асинхронный способ передачи и лишь часть сигналов и функций, предусмотренных стандартом RS-232.

В микроконтроллерах AVR с UART аппаратно-программными средствами поддерживаются только две линии: линия приема (RxD) и линия передачи данных (TxD).

Для связи компьютера с микроконтроллером мы используем только общий провод и две сигнальные линии. По одной из них данные передаются от микроконтроллера к компьютеру, по другой – от компьютера к микроконтроллеру.

Данные по каналу передаются последовательно, бит за битом. Информация передается группами битов. Для выделения группы перед ее началом передается стартовый бит, по окончании передачи группы – один или два стоповых бита, которым может предшествовать бит четности.

Описываемое здесь устройство передает стартовый бит, 8-битную группу (один байт), бит четности и один столовый бит.

И передающее, и принимающее устройства должны работать на одинаковой скорости, которая может быть выбрана из предусмотренного стандартом ряда.

Логическая единица передается по каналу RS-232 уровнем напряжения -3…-12 В относительно общего провода, логический ноль – уровнем напряжения +3...+12 В. Указанные уровни напряжений приведены ко входу приемника сигнала.

Уровни сигналов в диапазоне -3...+3 В попадают в зону нечувствительности и не воспринимаются приемником.

Компьютер снабжен необходимыми преобразователями уровней, и на контактах его COM-портов уровни сигналов соответствуют требованиям стандарта RS-232.

На линиях RxD и TxD микроконтроллера уровни сигналов соответствуют

уровням TTL-логики, поэтому для подключения микроконтроллера к каналу RS-232 требуется преобразователь уровней.

На Рис. 1 приведен фрагмент схемы контроллера, обеспечивающий подключение микроконтроллера к COM-порту компьютера.

Линия TxD микроконтроллера подключается к линии RxD компьютера через преобразователь уровня, а линия TxD компьютера – к линии RxD микроконтроллера также через преобразователь уровня.

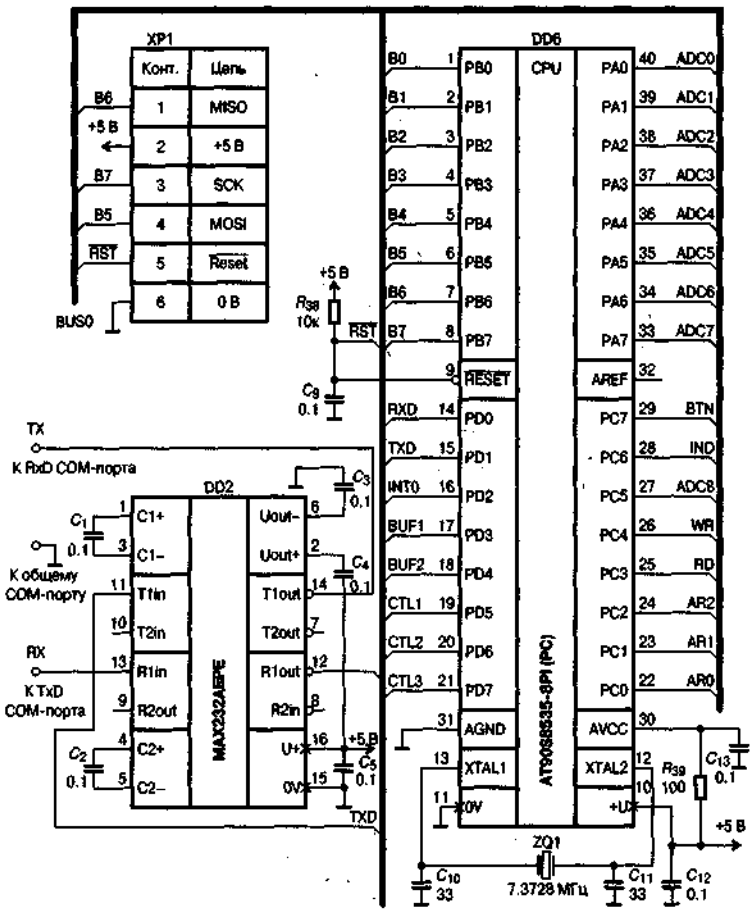

Рис. 1. Схема обеспечения связи микроконтроллера с компьютером

В компьютере для организации COM-порта используется либо 9-контактный (DB-9M), либо 25-контактный разъем (DB-25M). Поэтому для изготовления кабеля надо подобрать подходящий к разъему COM-порт вашего компьютера разъем DB-9F или DB-25F.

В табл. 1 приведены наименования и номера контактов этих разъемов использующихся для подключения нашей схемы к COM-порту компьютера.

Таблица 1 – Наименования и номера контактов разъемов DB-9F и DB-25F.

Наименование контакта	Разъем DB-9F	Разъем DB-25F
TxD	3	2
RxD	2	3
Общий	5	7

Частота кварцевого резонатора ZQ1 равена 7.3728 МГц. Использование

кварцевого резонатора с такой частотой позволяет точно устанавливать скорости передачи, рекомендованные стандартом RS-232.

2. Программное обеспечение связи по каналу RS-232.

Для того чтобы получить или передать данные по каналу RS-232, необходимо создать программы для обоих связываемых устройств. Значит, нужна программа для микроконтроллера и программа для компьютера.

Программы для компьютера написаны в Borland Delphi, а для обращения к COM-порту используется дополнительно установленный в Delphi компонент.

Прежде всего, надо решить, что, как и когда передавать.

2.1. Протокол обмена.

Обычно программное обеспечение для компьютера и для микроконтроллера разрабатывают разные специалисты или группы специалистов.

Протокол обмена должен содержать всю необходимую информацию для разработки обеих программ так, чтобы независимо созданные программы, удовлетворяющие всем требованиям протокола, могли быть в кратчайший срок стыкованы друг с другом.

Поэтому в протоколе должны быть учтены все тонкости совместной работы программ, а его положения должны быть четкими и не должны допускать возможности различного толкования. В то же время нет смысла перегружать протокол, например, переписывая из стандарта RS-232 требования к электрическим параметрам сигнала.

Протокол согласовывается всеми заинтересованными сторонами.

Ниже приводится пример протокола обмена, он содержит достаточно информации, чтобы понять задачу, которую предстоит решить.

2.2. Общие положения.

Обмен информацией выполняется асинхронными сериями в соответствии с требованиями интерфейса RS-232. Электрические характеристики сигнала также должны соответствовать требованиям интерфейса RS-232-C.

Данные передаются сообщениями. Сообщения состоят из 5... 134 байт и требуют передачи не чаще одного раза в секунду.

Каждый байт сообщения, состоящий из 8 информационных бит, передается в составе группы из одиннадцати бит. Группа начинается стартовым битом, затем передаются 8 информационных бит, за ними следуют бит четности и стоп-бит.

Параметры передаваемых групп:

- скорость передачи 115200 бит/с;
- 8 информационных бит;
- проверка на нечетность;
- стоп-бит.

2.3. Структура сообщения.

Сообщения имеют следующий вид: 5A CMD <байты данных> CSH CSL, где 5A — стартовый байт, CMD — байт команды, <байты данных> — их количество зависит от байта команды, CSH и CSL — 2 байта контрольной суммы.

Контрольная сумма получается суммированием всех байтов сообщения в сумматоре длиной в 2 байта. В зависимости от команды старший байт контрольной суммы CSH может не передаваться.

2.4. Передаваемые сообщения (команды компьютера и ответы контроллера).

В тексте сообщений неизменяемые байты записаны в шестнадцатеричном формате, изменяемые байты записаны как подчеркнутые трехсимвольные переменные.

1.Команда ТЕСТ RS-232: 5A 20 00 00 00 00 7A.

2.Ответ на команду ТЕСТ RS-232: 5A 20 CRH CRL CSL, где CRH и CRL – два байта контрольной суммы содержимого памяти программ контроллера, а CSL – младший байт контрольной суммы команды.

3.Команда ПЕРЕДАТЬ БЛОК 128 байт в контроллер: 5A 21 00 00 B00 ... BFF CSH CSL, где B00 ... BFF – 128 байт данных, передаваемых из компьютера, а CSH и CSL – два байта контрольной суммы команды.

4.Ответ на команду ПЕРЕДАТЬ БЛОК: 5A 21 00 00 7B.

5.Команда ЗАПРОС БЛОКА 128 байт из контроллера: 5A 22 00 00 00 00 7C.

6.Ответ на команду ЗАПРОС БЛОКА: 5A 22 00 00 B00 ... BFF CSH CSL, где B00 ... BFF – 128 байт данных, передаваемых из ОЗУ контроллера, а CSH и CSL – два байта контрольной суммы команды.

7.Команда ОЧИСТИТЬ ОЗУ: 5A 23 00 00 00 00 7D.

8.Ответ на команду ОЧИСТИТЬ ОЗУ: 5A 23 00 00 7D.

9. Команда ЗАПОЛНИТЬ ОЗУ: 5A 24 FIL 00 00 CSH CSL, где FIL -байт, которым заполняется ОЗУ.

10.Ответ на команду ОЧИСТИТЬ ОЗУ: 5A 24 00 00 7E.

11.Сообщение контроллера об ошибках: 5A OC ERR 00 CSL, где ERR – код ошибки (E0 — ошибка четности или первый байт команды не 5A, E1 – ошибка контрольной суммы команды, E2 – неизвестная команда), а CSL – младший байт контрольной суммы.

И микроконтроллер, и компьютер проверяют поступающие сообщения, в том числе и на их длину. Проще проверять длину сообщений, когда для одного источника она одинакова. В нашем устройстве команды компьютера имеют длину в 7 байт, а ответы контроллера – 5 байт. Исключения составляют сообщения, содержащие блоки данных. Их длина составляет 134 байта.

2.5. Программа для микроконтроллера

Как организовать прием команд, описанных в протоколе обмена? Предположим, все команды содержат одинаковое количество байтов, например семь. Тогда после приема первого байта надо принять еще шесть байт, а затем начать обработку принятой последовательности.

Что произойдет при такой организации, если прием первого байта был вызван помехой? Микроконтроллер при этом просто зависнет, ожидая следующих шести байт, а часть следующей команды будет воспринята как продолжение предыдущей. Если длина команд различна, то, приняв первый байт, надо продолжать прием в течение времени, достаточного для приема самой длинной команды, предусмотренной протоколом обмена, а по истечении этого времени проанализировать принятую последовательность.

Организуем программу для микроконтроллера следующим образом: после необходимой инициализации программа будет выполнять бесконечный цикл, ожидая прерывания после приема первого байта данных по каналу RS-232. Обработчик прерывания окончания приема символа UART (универсальным синхронным и асинхронным приемопередатчиком) запустит таймер TO и вызовет выполнение подпрограммы приема последовательности байтов. Завершение счета таймера TO прекратит выполнение подпрограммы приема и инициализирует выполнение ряда подпрограмм анализа принятой последовательности, распознавания команды, ее обработки и передачи ответа микроконтроллером по каналу RS-232.

Замечание. Прерывание окончания приема символа UART происходит после того, как принятый символ помещен в регистр UDR (UART Data Register – регистр данных UART). UART принимает символ длиной 9 бит, 8 из них представляют передаваемый байт, 9-й – это передаваемый бит четности.

2.6. Отладка работы UART в AVR Studio

После выполнения обычных действий по созданию проекта, набора Программы связи с компьютером по каналу RS-232 и ее ассемблирования перейдите к отладке.

Программа может использоваться в микроконтроллере ATraega8535, хотя написана она для AT90S8535. Поэтому следует выбрать AT90S8535 в качестве целевого устройства (Debug/Select Platform and Device.../AVR Simulator/AT90S8535). Этим же объясняется включение директивой .include файла 8535def.inc (а не m8535def.inc, связанного с микроконтроллером ATmega8535) а также использование наименования UART (для AT-mega8535 эта функция именуется USART, а число и наименования регистров несколько отличаются).

Найдите в окне программы метку *cycle:* и выполните часть программы до этой метки (Ctrl + F10). Выполните 5...7 шагов отладки (F11) – программа все время возвращается на метку *cycle:,* с которой начинается бесконечный цикл.

Откройте окно 10 (меню View/New IO View), разверните UART/Status Register и UART/Control Register так, чтобы видеть наименования разрядов. Щелкните правой клавишей мышки по строке Data Register и в открывшемся

окне выберите режим Check Box.

Имитируем окончание приема символа UART, установив в Status Register разряд RXC. Следующий шаг отладки вызовет переход на вектор прерывания окончания приема символа UART, по которому расположена команда *rjmp UART_RXC.* Ее выполнение переместит указатель отладчика на начало обработчика прерывания. Выполните часть обработчика до команды *in datrs, UDR.*

Имитируем прием первого байта команды $5A или в двоичном коде 01011010. Для этого установим разряды 6, 4, 3 и 1 в Data Register (окно IO/UART/Data Register). Выполним один шаг отладки, введенные данные должны оказаться в переменной datrs, проверьте ее состояние, вызвав окно Watches.

Так как в принятом байте четное число единиц, то бит четности должен равняться единице (при проверке на нечетность общее число единиц в 9-разрядном символе нечетно). Поэтому в Control Register установим разряд RXB8, хранящий последний бит символа.

Отлаживая программу далее, можно проследить вычисление бита четности, сравнение вычисленного и принятого битов, выполнение проверки значения первого байта команды.

Так как ошибки при приеме первого байта команды не обнаружены, обработчик прерывания выполняется полностью, а после возврата в бесконечный цикл программы будет вызвана подпрограмма *GetCmd* приема оставшихся байтов команды.

Для приема отдельных символов вызывается подпрограмма *RxCh.*

При ее отладке надо повторить действия, описанные для обработчика прерывания *UART_RXC:* выполнить несколько шагов отладки (это будет приводить к возврату на метку *mrxl:),* установить в Status Register разряд

RXC, ввести в Data Register значение очередного байта команды, а в Control Register изменить состояние разряда RXB8 в соответствии с тем, каким должен быть бит четности для записанного байта.

Вызовите окно Memory, чтобы увидеть отображение поступающей команды в ОЗУ микроконтроллера.

При необходимости отладить подпрограмму распознавания команд перед меткой *cycle:* установите команду *rjmp RcgCmd.*

При этом в соответствующие ячейки окна Memory введите байты команды. Напомню, что первый байт $5A не записывается в ОЗУ, а байты команды «Записать блок 128 байт в контроллер» разнесены: только один байт, представляющий код команды, записывается в ячейку по адресу *aC-MD,* остальные 132 байта помещаются в область ОЗУ, адрес которой *aNumBlk* (0x0O7E).

3. Канал RS-232: программное обеспечение для компьютера.

При разработке приложения в Delphi для управления COM-портом компьютера можно использовать обращение к портам ввода/вывода, вызывать функции операционной системы (Windows API) или использовать готовый компонент, который можно найти в сети Internet для среды разработки, в которой выполняется приложение.

Поскольку в Delphi не предусмотрены функции обращения к портам ввода/вывода компьютера, их придется писать самостоятельно на ассемблере. Программа получится громоздкой и неудобной. А при создании приложений под операционную систему Windows NT/2000/XP такой вариант потребует дополнительных мер для обхода запрета, налагаемого операционной системой на прямые обращения к портам ввода/вывода. Поэтому непосредственное обращение к портам ввода/вывода, без которого нельзя было обойтись на компьютерах с DOS, использовать в приложениях под операционные системы Windows 9x, 2000 или NT нецелесообразно.

При отсутствии компонентов можно пользоваться функциями Windows **API,** для этого необходимо наличие описания этих функций и их параметров, их можно разыскать в сети. Преимуществом функций Windows **API** является то, что обращение к ним остается одинаковым независимо от того, пишете вы приложение в Delphi или в C.

Наиболее удобным является использование готовых компонентов для работы с COM-портом, поэтому в рассматриваемых программах используется компонент VaComm из набора компонентов Varian Async32, предлагаемый компанией Varian Software. Этот компонент с файлами помощи и рекомендациями по установке для разных версий Delphi можно найти в сети.

4. Использование функций Windows API для обращения к COM-порту.

Компоненты, поддерживающие связь через COM-порт компьютера, как правило, обращаются к функциям Windows API. Поэтому полезно рассмотреть программу, использующую эти функции непосредственно.

Поскольку основная задача данного раздела лабораторной работы – ознакомление с функциями Windows API, программа будет максимально проста.

Разместите на форме компоненты Edit и Button со страницы Standard палитры компонентов Delphi.

В компонент Edit 1 будет вводиться команда, а после приема ответа микроконтроллера сюда же будет выводиться этот ответ.

Для компонента Edit 1 определите значение свойства /Text 5A 20 O O O O 7A/. Это команда /Тест COM/, которая появится в компоненте при запуске программы, и пользователю останется лишь нажать кнопку. Буквы «А» – латинские, между байтами команды должно быть только по одному пробелу.

Для компонента Button 1 определите обработчик события OnClick.

После добавлений, которые следует выполнить, программа должна принять следующий вид:

unit UAPI;

```
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
TForml = class(TForm)
Editl: TEdit;
Buttonl: TButton;
procedure ButtonlClick(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations.}
end;
var
Forml: TForml;
implementation
{$R *.DFM)
procedure TForml.ButtonlClick(Sender: TObject);
const arsz = 150;
var DCB:TDCB;
hndCOM:THandle;
A,B:array[0..arsz] of char;
Num:cardinal;
I,J:integer;
S,Sl,S2:string;
begin
hndCOM:= CreateFile(PChar(`COMl`) ,GENERIC_READ +
GENERIC_WRITE, 0, nil, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, 0);                    //1
DCB.BaudRate:= 115200;                        //2
DCB.ByteSize:= 8;                             //3
DCB.Parity:= 1;                               //4
DCB.StopBits:= 2;                             //5
SetCommState(hndCOM, DCB);                    //6
Sl:= Editl.Text;                              //7
S:= ` `;                                      //8
J: = 0;                                       //9
while SI[Length(S1)] = ' ' do SetLength(SI,Length(SI) - 1);       //10
while Length(S1)>0 do begin                   //11
S2:= '$'; I: = 1;                             //12
while (S1[I]<>' ') and (I< = Length(S1)) do begin
S2:= S2 + S1[I]; Inc(I); end;                 //13
Delete(S1.1,I);                               //14
A[J]:= Chr(StrToInt(S2));                     //15
```

```
Inc(J);                                          //16
end;                                             //17
WriteFile(hndCOM, A, J, Num, nil);               //18
ReadFile(hndCOM, B, 5, Num, nil);                //19
Sl:= ` `;                                        //20
for I:= 0 to Num - 1 do                          //21
Sl:= S1 + Format('%.2x',[Ord(B[I])]) + ' ';      //22
Editl.Text:= S1;                                 //23
Buttonl.Enabled:= false;                         //24
end;                                             //25
end.                                             //26
```

Оператор 1. Производится попытка создать файл `COM1`, если файл создан успешно, параметру *hnd.COM,* тип которого *THandle,* присваивается значение описателя файла. Это значение еще понадобится для обращений к функциям Windows API.

Операторы 2...5 записывают параметры COM-порта, соответствующие нашему протоколу обмена, в структуру BDE.

Оператор 6. Вызывается функция установки состояния COM-порта, определенного нами в структуре BDE.Операторы 7... 17 выполняют те же действия, что и рассмотренная ранее функция *VAString* (подраздел 5.4.2), с той лишь разницей, что значения подлежащих передаче через COM-порт символов помещаются не в строку, а в буфер (массив A типа char). Число подлежащих передаче элементов массива оказывается в переменной *J*, оно равно числу байтов в передаваемой команде.

Оператор 18. Вызывается функция записи содержимого буфера A в файл с описателем *hndCOM,* а это, как было определено, COM-порт; количество подлежащих записи элементов *J*; *Num.* – количество элементов, которые удастся записать.

Оператор 19. Вызывается функция считывания из файла с описателем *hndCOM* (из COM-порта) в буфер *B;* число считываемых элементов 5; *Num* – количество элементов, которые удастся считать.

Операторы 20...22 преобразуют принятые символы, хранящиеся в массиве *B,* в пригодный для просмотра вид (строка *S).*

Оператором 23 содержимое строки *S* отображается компонентом *Edit 1.*

Оператор 24 запрещает работу кнопки *Button 1.*

Запрещение работы кнопки вызвано тем, что порт уже открыт и инициализирован, а следующее нажатие кнопки вызвало бы новую попытку открыть порт, что привело бы к ошибке. Эти неприятности вы сможете легко устранить самостоятельно.

Основное неудобство при использовании функции *ReadFile* – это необходимость считывания заданного числа элементов. Если же задать заведомо большее число элементов, например не 5, а 150, то программа зависнет.

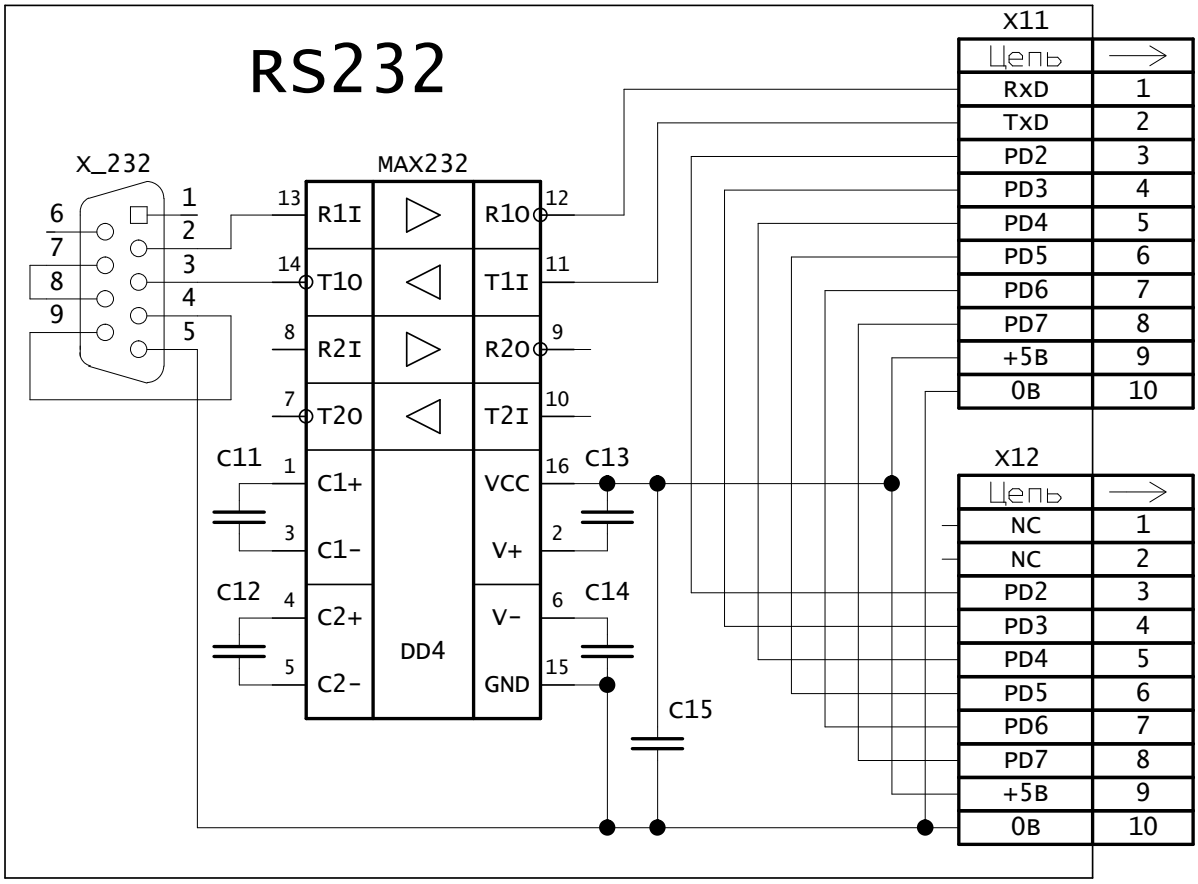

Рис.2 Схема электрическая принципиальная модуля драйвера последовательного интерфейса RS232

Задание на лабораторную работу.

1. На основании вышеприведенного теоретического материала разработать программу связи по каналу RS-232 микроконтроллера с компьютером. Для описания протокола обмена микроконтроллера использовать язык ассемблер. Для описания протокола обмена COM- порта в персональном компьютере, использовать среду разработки приложений Delphi.

2. Проверить наличие связи компьютера с контроллером посредством канала RS-232 в соответствии со стандартом RS-232.

3. Передать массив из компьютера в ОЗУ микроконтроллера, очистить ячейки ОЗУ, хранящие массив, заполнить эти ячейки значением, указанным в команде компьютера, и считать массив из ОЗУ микроконтроллера в компьютер. Все запросы должен генерировать компьютер, на каждый запрос микроконтроллер должен дать ответ.

Контрольные вопросы.

1. Для чего предназначен канал RS-232?
2. Какие режимы работы канала RS-232 бывают?
3. Электрические параметры канала RS-232.
4. Структура сообщения по каналу RS-232.
5. Требования интерфейса RS-232.