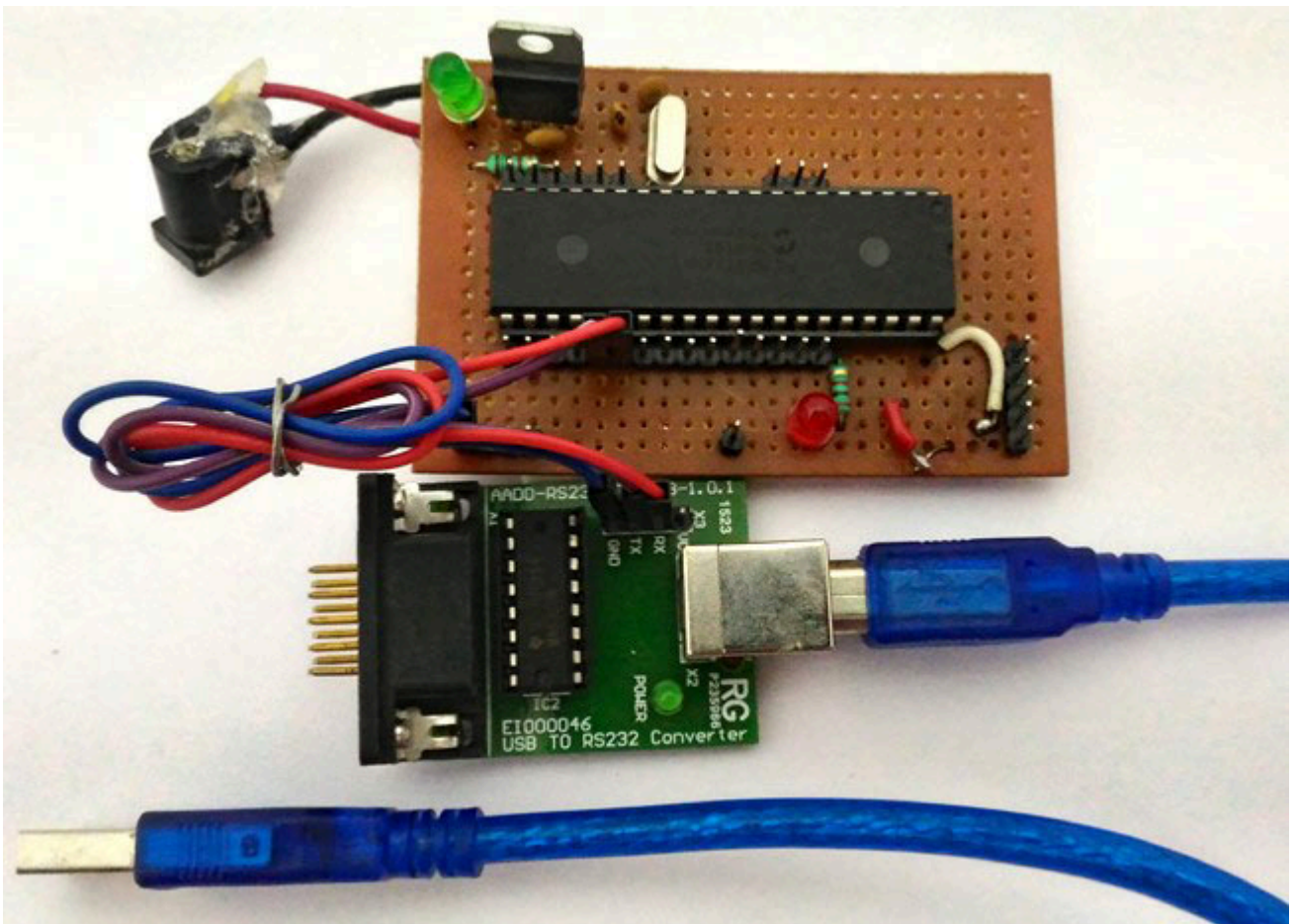


Последовательная связь в микроконтроллере PIC: руководство для начинающих

admin-new

В данной статье мы рассмотрим организацию последовательной связи (UART) между микроконтроллером PIC и персональным компьютером. Несмотря на то, что в современных персональных компьютерах последовательный (COM) порт уже в явном виде не используется, во встраиваемой электронике он продолжает широко применяться.



В данном проекте мы рассмотрим последовательную связь в микроконтроллере PIC16F877A, который имеет модуль под названием **USART** (Addressable Universal Synchronous Asynchronous Receiver and Transmitter – универсальный синхронно-асинхронный приемопередатчик). USART представляет собой двухпроводную

систему, в которой данные передаются последовательно. USART является полно дуплексным интерфейсом, что означает что данные в нем можно передавать и принимать одновременно.

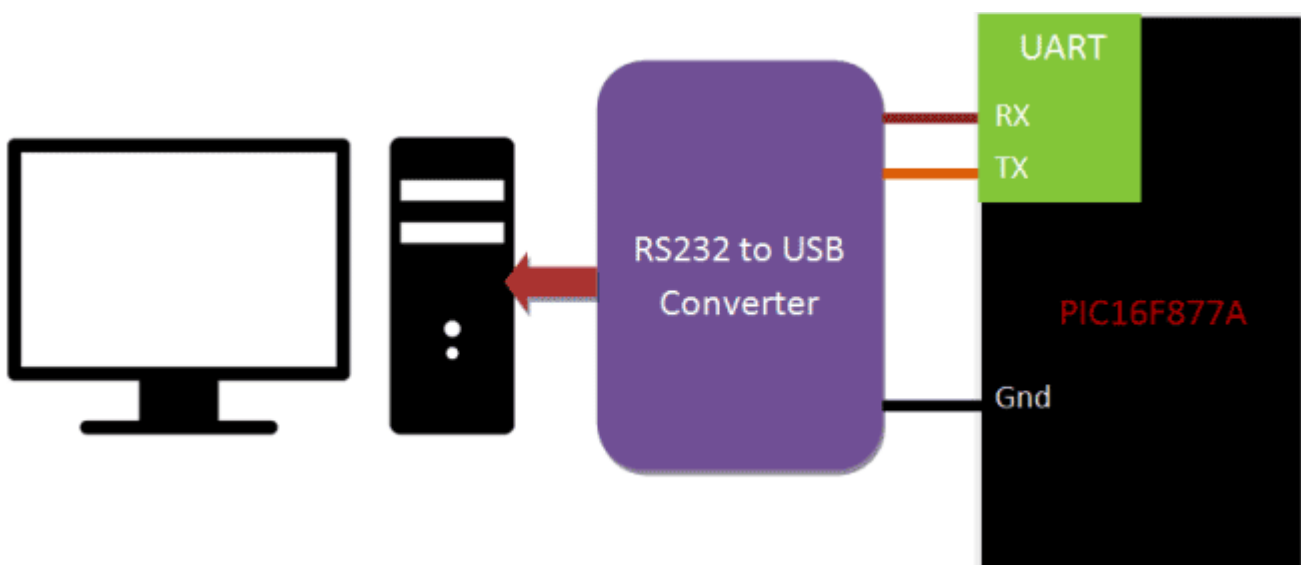
USART может быть сконфигурирован для работы в следующих режимах:

- асинхронный (полный дуплекс);
- синхронный ведущий (Master) (полудуплекс);
- синхронный ведомый (Slave) (полудуплекс).

Также в USART существует еще два различных режима, которые называются 8-битный и 9-битный. В данном проекте мы сконфигурируем работу USART в 8-битном асинхронном режиме, как наиболее часто применяемом в системах встраиваемой электроники. Поскольку режим асинхронный, то в нем нет необходимости передавать сигнал тактовой частоты.

UART (Universal Asynchronous Receiver/Transmitter – универсальный асинхронный приёмопередатчик) использует две линии – для передачи (Tx) и приема (Rx) данных. Данный тип связи не имеет общей линии синхронизации, поэтому наличие общего провода (земли) крайне необходимо для работы подобной системы.

Структурная схема организации последовательной связи между микроконтроллером PIC16F877A и компьютером в нашем проекте приведена на следующем рисунке.



С компьютера мы с помощью последовательной связи будем передавать сигналы, которые будут включать/выключать светодиод, подключенный к микроконтроллеру PIC. А с микроконтроллера PIC на компьютер по последовательной связи будет передаваться информация о состоянии светодиода. Тестирование проекта мы будем осуществлять с помощью программы Hyper Terminal, запущенной на компьютере. Более подробно работу проекта вы можете посмотреть на видео, приведенном в конце статьи.

Необходимые компоненты

Аппаратное обеспечение

1. Микроконтроллер PIC16F877A ([купить на AliExpress](#)).
2. Модуль конвертера RS232 в USB.
3. Программатор PICkit 3 ([купить на AliExpress](#)).
4. Держатель микросхем на 40 контактов ([купить на AliExpress](#)).
5. Кварцевый генератор 20 МГц ([купить на AliExpress](#)).
6. Конденсатор 33 пФ – 2 шт. ([купить на AliExpress](#)).
7. Светодиод ([купить на AliExpress](#)).
8. Компьютер.
9. Перфорированная плата.

Реклама: ООО "АЛИБАБА.КОМ (РУ)" ИНН: 7703380158

Программное обеспечение

1. MPLABX.
2. HyperTerminal.

Конвертер RS232 в USB необходим для преобразования последовательных данных, поступающих от микроконтроллера PIC, в форму, удобную для считывания компьютером. Данный конвертер можно как купить готовым, так и сделать самому. Внешний вид конвертера RS232 в USB представлен на следующем рисунке.



Примечание: каждый конвертер RS232 в USB требует установки специального драйвера. Для большинства подобных конвертеров подобный драйвер устанавливается автоматически как только вы подключите конвертер к компьютеру. Но не для всех, поэтому обратите на этот вопрос особое внимание.

Программирование микроконтроллера PIC для работы с UART

Для программирования работы микроконтроллера PIC с использованием последовательной связи необходимо инициализировать USART модель микроконтроллера. Мы будем настраивать его для работы в 8-битном режиме. После [настройки битов конфигурации](#) можно приступить к настройке модуля UART.

Инициализация модуля UART микроконтроллера PIC

Линии последовательной связи *Tx* и *Rx* микроконтроллера PIC расположены на его контактах RC6 и RC7. Запрограммируем контакт TX для работы в качестве цифрового выхода, а контакт RX – в качестве цифрового входа.

```
/**Setting I/O pins for UART***/
```

```
TRISC6 = 0; // TX Pin set as output
```

```
TRISC7 = 1; // RX Pin set as input
```

```
// _____ I/O pins set _____ //
```

После этого необходимо установить бодовую скорость – это скорость, с которой данные передаются по последовательному каналу связи. В нашем проекте мы будем использовать скорость передачи данных равную 9600 бод.

```
/**Initialize SPBRG register for required  
baud rate and set BRGH for fast baud_rate**/  
SPBRG = ((_XTAL_FREQ/16)/Baud_rate) - 1;  
BRGH = 1; // for high baud_rate  
// _____ End of baud_rate setting _____ //
```

Значение бодовой скорости устанавливается с помощью регистра *SPBRG*, также оно зависит от частоты используемого кварцевого генератора. Формула для вычисления бодовой скорости в микроконтроллере PIC выглядит следующим образом:

$$SPBRG = ((_XTAL_FREQ/16) / Baud_rate) - 1;$$

Бит *BRGH* необходимо устанавливать в high чтобы сделать доступными высокие значения бодовой скорости. В соответствии с даташитом на микроконтроллер PIC16F877A (страница 13) всегда желательно устанавливать данный бит, поскольку он также позволяет устранять ошибки во время передачи данных.

Как мы уже говорили ранее, мы будем работать в асинхронном режиме, поэтому бит *SYNC* необходимо установить в 0, а бит *SPEN* – в high чтобы сделать доступными контакты для последовательной связи (TRISC6 и TRISC5).

```
/**Enable Asynchronous serial port***/  
SYNC = 0; // Asynchronous  
SPEN = 1; // Enable serial port pins  
// _____ Asynchronous serial port enabled _____ //
```

В нашем проекте мы будем как передавать данные от микроконтроллера PIC к компьютеру, так и принимать их, поэтому нам необходимо установить оба бита *TXEN* и *CREN*.

```
/**Lets prepare for transmission & reception**//  
TXEN = 1; // enable transmission  
CREN = 1; // enable reception  
//__UART module up and ready for transmission and reception__//
```

Биты *TX9* и *RX9* необходимо установить в 0 чтобы работать в 8-битном режиме. Если возникнет острая необходимость в высокой стабильности связи, то тогда целесообразным выглядит использование 9-битного режима.

```
/**Select 8-bit mode**//  
TX9 = 0; // 8-bit reception selected  
RX9 = 0; // 8-bit reception mode selected  
//__8-bit mode selected__//
```

На этом настройка модуля UART у нас будет закончена и он будет готов к функционированию.

Передача данных с помощью UART

Для передачи данных с помощью модуля UART может быть использована следующая функция:

```
/**Function to send one byte of date to UART**//  
void UART_send_char(char bt)  
{  
    while(!TXIF); // hold the program till TX buffer is free  
    TXREG = bt; //Load the transmitter buffer with the received value  
}  
//_____End of function_____//
```

Когда модуль UART инициализирован для работы, то любое значение, которое будет загружено в регистр *TXREG*, будет передаваться через UART (последовательную связь), но передачи могут частично перекрываться. Чтобы этого не происходило мы должны всегда проверять флаг прерываний передачи *TXIF* (Transmission Interrupt flag). Только если этот флаг находится в состоянии low, мы можем начать передачу следующего бита, иначе мы должны подождать пока состояние этого флага не станет low.

Представленная выше функция может быть использована только для передачи одного байта данных, чтобы передать строку данных необходимо использовать следующую функцию:

```
/**Function to convert string to byte**/  
void UART_send_string(char* st_pt)  
{  
    while(*st_pt) //if there is a char  
        UART_send_char(*st_pt++); //process it as a byte data  
}  
//_____End of function_____//
```

Код этой функции может быть немного сложным для понимания поскольку он содержит указатели, но во многих случаях указатели значительно упрощают процесс написания программы – и у нас как раз именно этот случай.

Как вы можете видеть из кода представленной функции мы в ней функцию `UART_send_char()` вызываем многократно, внутри цикла. Фактически мы разделяем строку на отдельные символы, и каждый символ передаем отдельно, помещая его в регистр *TXREG*.

Прием данных с помощью UART

Следующая функция может быть использована для приема данных с помощью модуля UART.

```
/**Function to get one byte of data from UART**//
```

```

char UART_get_char()
{
    if(OERR) // check for Error
    {
        CREN = 0; //If error -> Reset
        CREN = 1; //If error -> Reset
    }
    while(!RCIF); // hold the program till RX buffer is free
    return RCREG; //receive the value and send it to main function
}
//_____End of function_____//

```

Когда модуль UART принимает данные он сохраняет их в регистре *RCREG*, после чего мы их можем передать в любую переменную и затем использовать по своему усмотрению. Но возможны частичные перекрытия в приеме данных и из этого ошибки приема в случае когда данные передаются непрерывно и мы не успеваем переписать их в переменную. В этом случае нам на помощь приходит флаг приема *RCIF*. Данный бит будет в состоянии low всегда, когда данные уже приняты, но еще не обработаны. Поэтому мы в цикле, проверяя значение данного флага, сможем создать задержку, необходимую для завершения обработки принятых данных.

Включение/выключение светодиода в зависимости от принятых по UART данных

В финальной части нашей программы, в функции `void main(void)` мы будем включать/выключать светодиод в зависимости от команд, передаваемых компьютером нашему микроконтроллеру PIC с помощью последовательной связи.

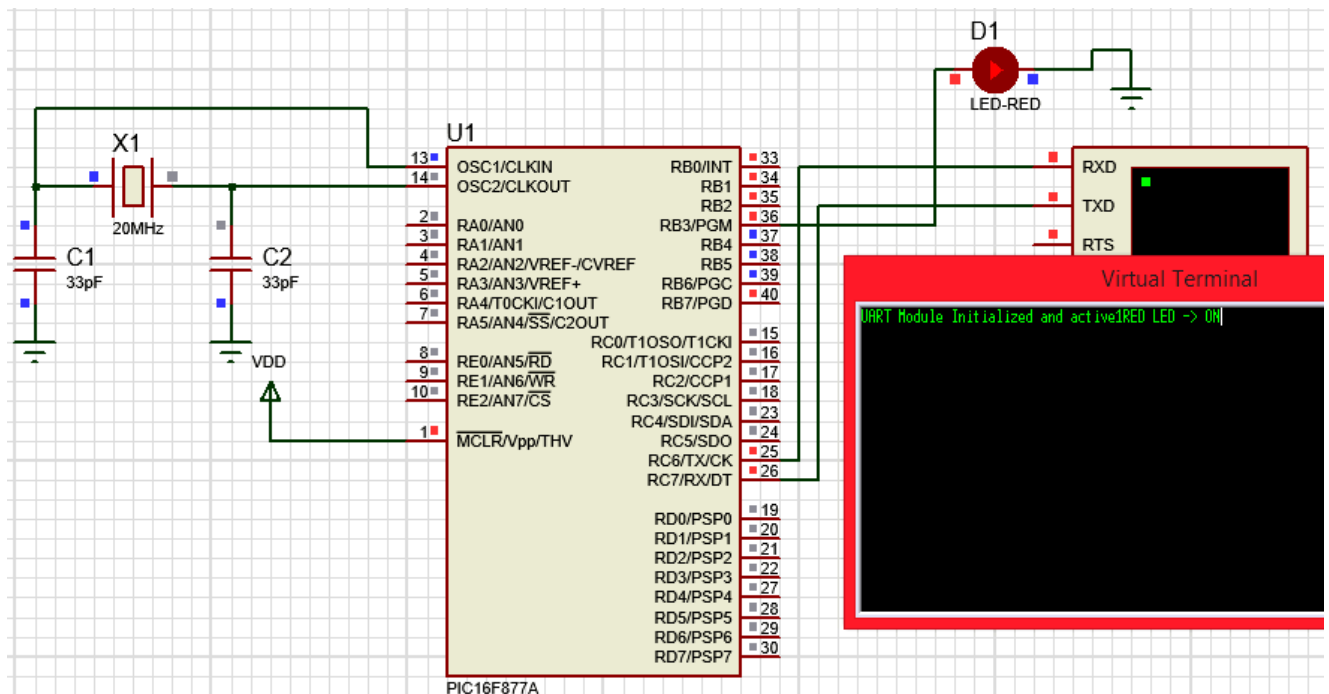
Когда с компьютера мы будем передавать символ "1" светодиод будет включаться и от микроконтроллера PIC компьютеру в обратную сторону будет передаваться сообщение "RED LED -> ON".

Аналогичным образом, когда с компьютера мы будем передавать символ "0" светодиод будет выключаться и от микроконтроллера PIC компьютеру в обратную сторону будет передаваться сообщение "RED LED -> OFF".

```
1  while(1) //Infinite loop
2      {
3          get_value = UART_get_char();
4          if (get_value == '1') //If the user sends "1"
5              {
6                  RB3=1; //Turn on LED
7                  UART_send_string("RED LED -> ON"); //Send notification to the computer
8                  UART_send_char(10); //ASCII value 10 is used for carriage return (to print in new
9  line)
10             }
11         if (get_value == '0') //If the user sends "0"
12             {
13                 RB3=0; //Turn off LED
14                 UART_send_string("RED -> OFF"); //Send notification to the computer
15                 UART_send_char(10); //ASCII value 10 is used for carriage return (to print in new
16  line)
17             }
18     }
```

Симуляция работы программы

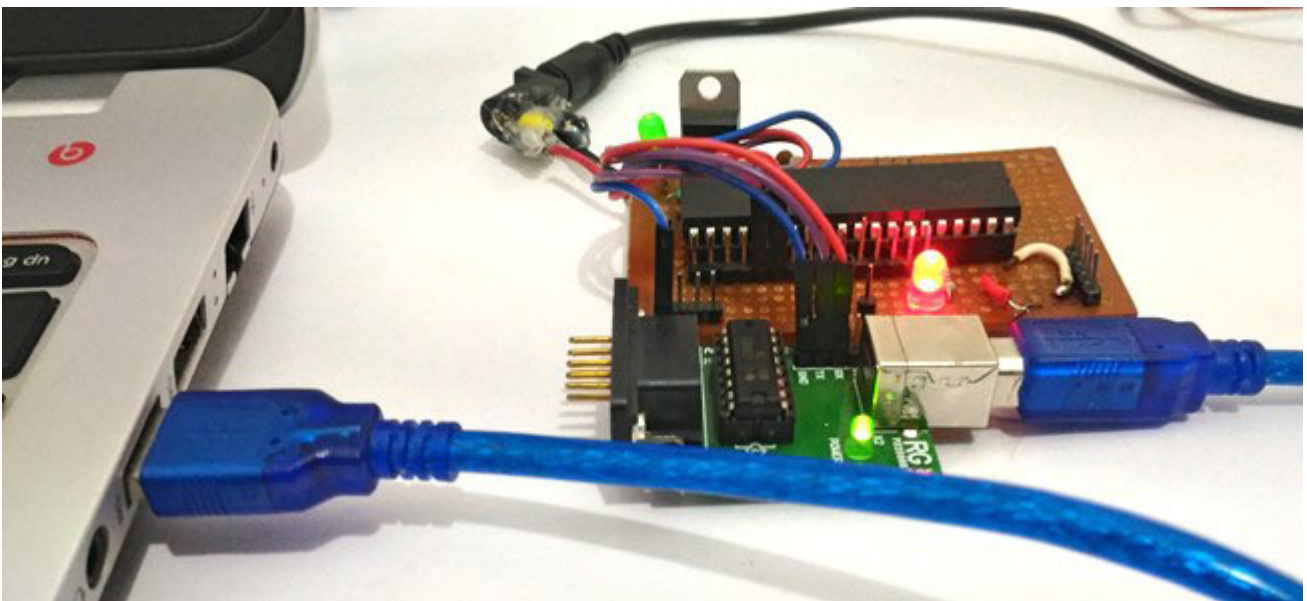
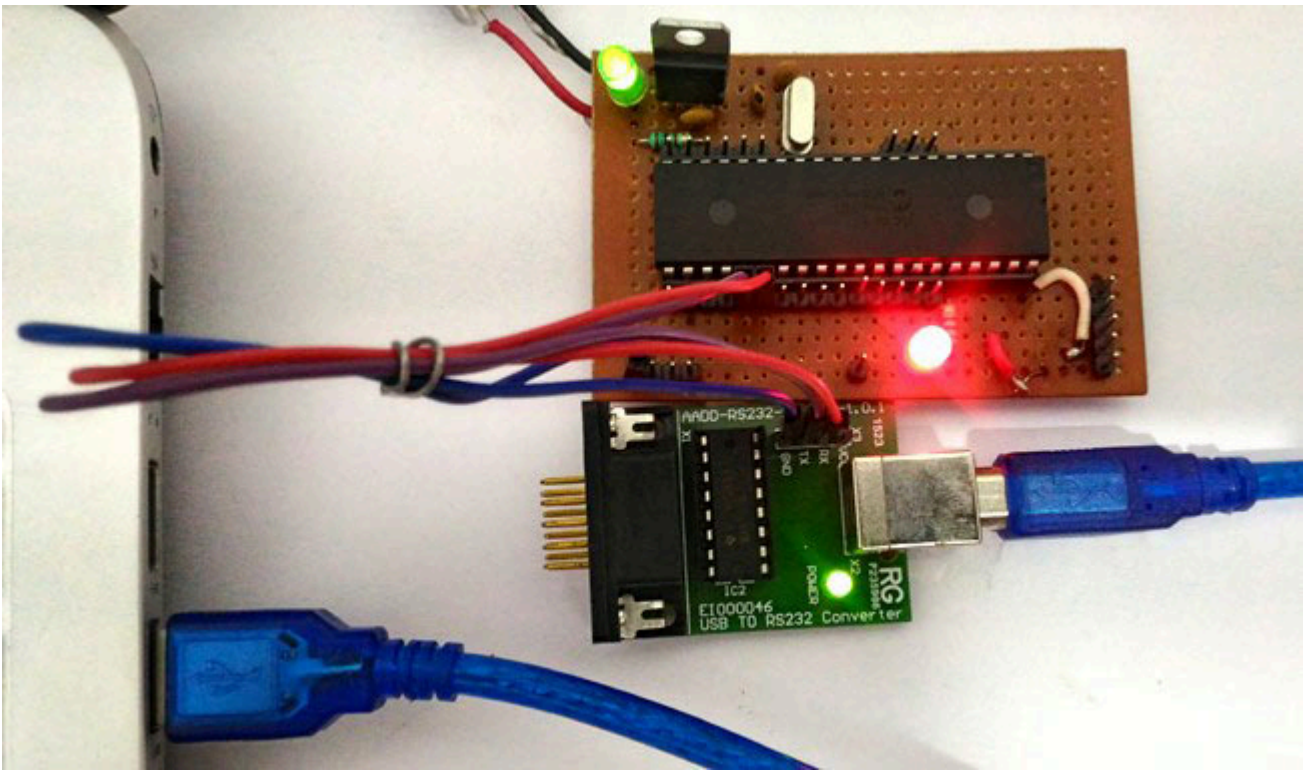
Мы произвели тестирование работы программы в симуляторе proteus. Внешний вид схемы нашего проекта в симуляторе proteus показан на следующем рисунке.



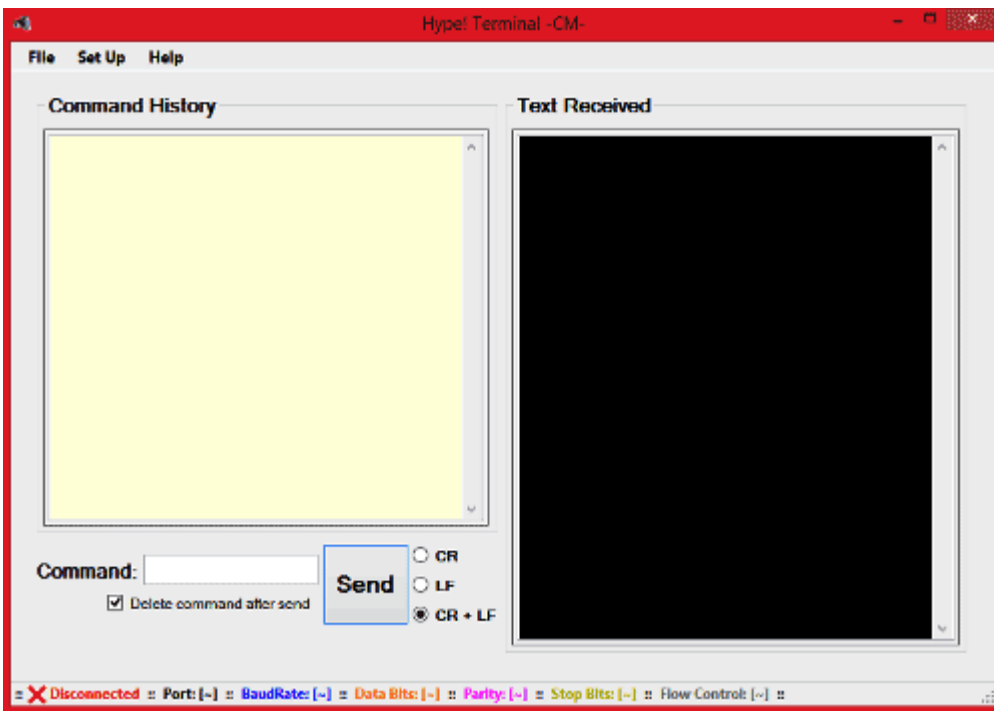
На представленном рисунке также показан виртуальный терминал, который показывает приветственное сообщение и состояние светодиода. Светодиод красного цвета подключен к контакту RB3. Более подробно работу проекта вы можете посмотреть на видео, приведенном в конце статьи.

Тестирование работы проекта

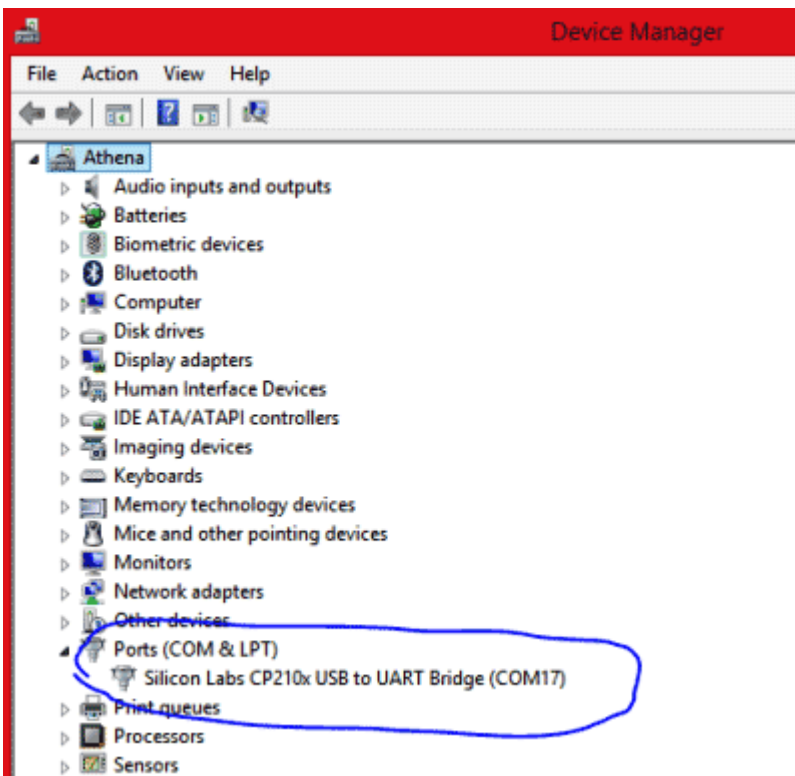
Для тестирования работы проекта мы подключили перфорированную плату с нашим микроконтроллером PIC тремя проводами к конвертеру RS232 в USB, а конвертер подключили к компьютеру с помощью USB кабеля как показано на следующих рисунках.



После этого мы установили на компьютер программу Hyper Terminal и запустили ее на выполнение.

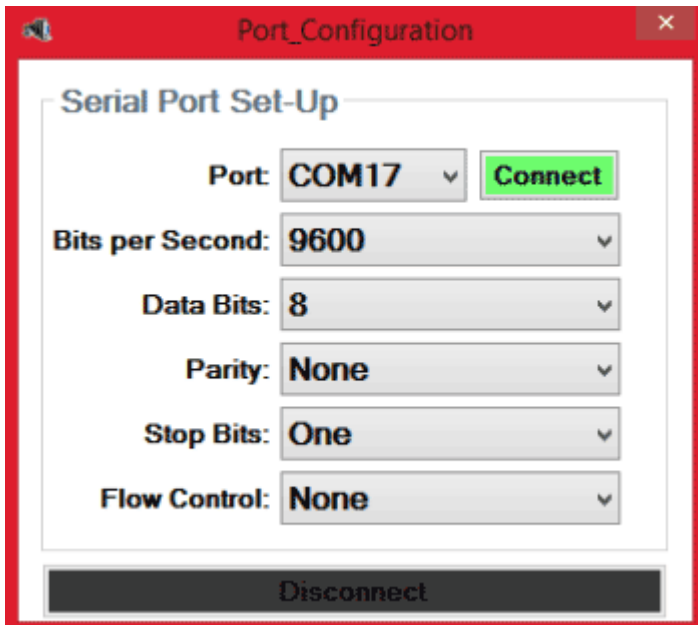


После этого откройте диспетчер устройств на своем компьютере и проверьте к какому COM порту подключился ваш конвертер, в нашем случае это оказался COM port 17 как показано на следующем рисунке. У вас это может быть другой номер порта.

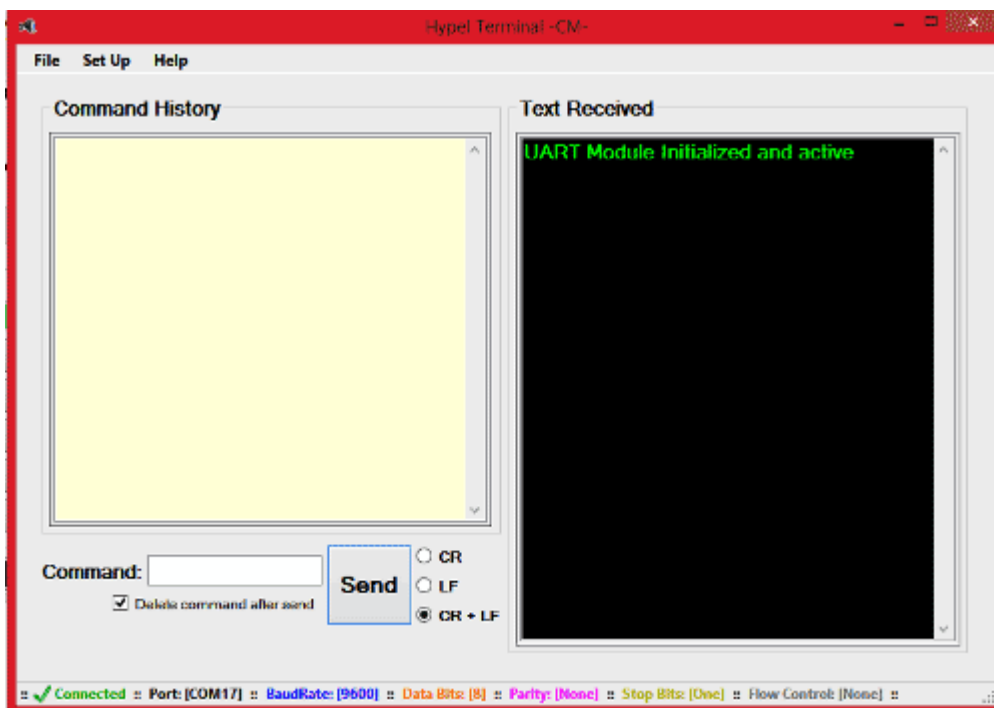


После этого вернитесь в программу Hyper Terminal и выберите в ней пункт меню Set Up -> Port Configuration (или нажмите Alt+C), в

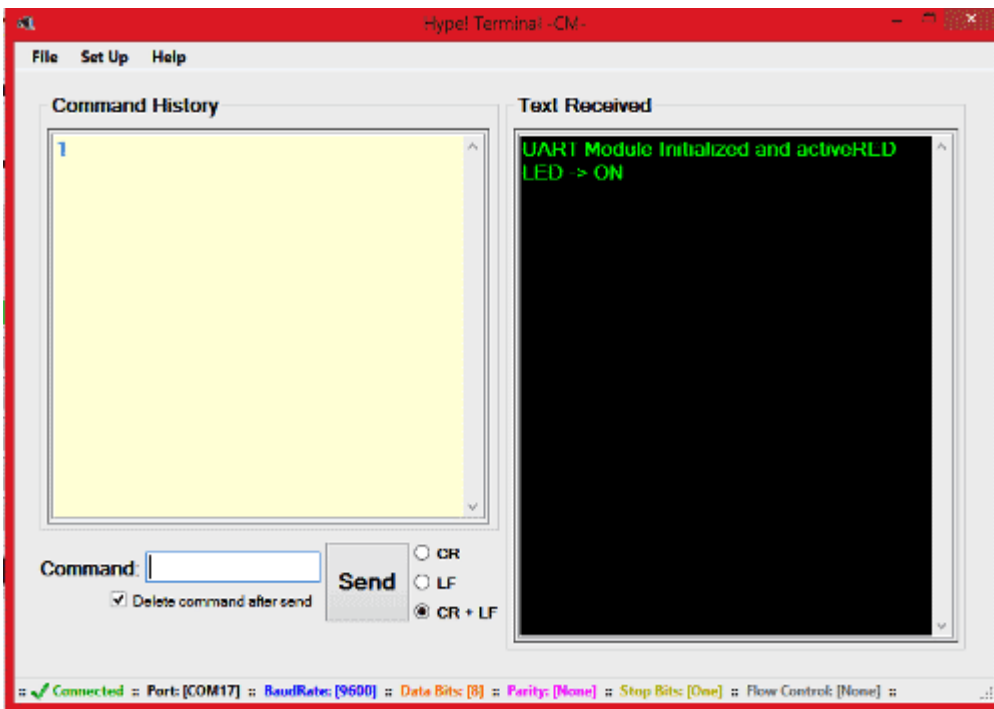
открывшемся всплывающем окне выберите необходимый COM порт (COM17 в нашем случае) и нажмите на кнопку connect.



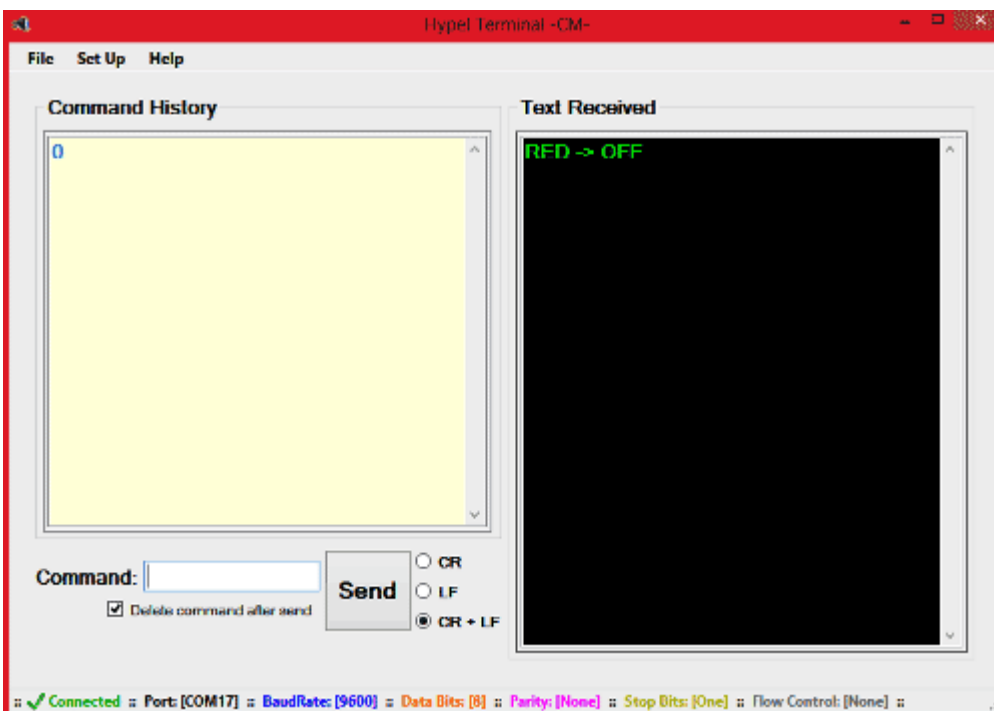
После того как соединение с вашим микроконтроллером PIC будет установлено вы на экране компьютера должны увидеть примерно следующую картину.



Поместите курсор в окно команд (Command Window), введите цифру 1 и затем нажмите на enter. Светодиод включится и на экране появится сообщение о том, что он включен.



Аналогичным образом, поместите курсор в окно команд (Command Window), введите цифру 0 и затем нажмите на enter. Светодиод выключится и на экране появится сообщение о том, что он выключен.



Исходный код программы

- 1 // CONFIG
- 2 #pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
- 3 #pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

```

4 #pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT enabled)
5 #pragma config BOREN = ON // Brown-out Reset Enable bit (BOR enabled)
6 #pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for
7 programming)
8 #pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)
9
10 #pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)
11 #pragma config CP = OFF // Flash Program Memory Code Protection bit (Code
12 protection off)
13 // End of configuration
14 #include <xc.h>
15 #define _XTAL_FREQ 20000000
16 #define Baud_rate 9600
17 /***/Initializing UART module for PIC16F877A***/i
18 void Initialize_UART(void)
19 {
20 ****Setting I/O pins for UART****/i
21 TRISC6 = 0; // TX Pin set as output
22 TRISC7 = 1; // RX Pin set as input
23 // _____ I/O pins set _____//
24 /**Initialize SPBRG register for required
25 baud rate and set BRGH for fast baud_rate**/
26 SPBRG = ((_XTAL_FREQ/16)/Baud_rate) - 1;
27 BRGH = 1; // для высоких скоростей передачи
28 // _____ End of baud_rate setting_____//
29 ****задействуем асинхронный последовательный порт*****/i
30 SYNC = 0; // Asynchronous
31 SPEN = 1; // Enable serial port pins
32 // _____ Asynchronous serial port enabled _____//
33 /**Lets prepare for transmission & reception**/
34 TXEN = 1; // enable transmission (разрешаем передачу)
35 CREN = 1; // enable reception (разрешаем прием)
36 // _UART module up and ready for transmission and reception_//

```

```

37  /**выбираем 8-битный режим**//
38  TX9  = 0;  // 8-bit reception selected
39  RX9  = 0;  // 8-bit reception mode selected
40  __8-bit mode selected__//
41  }
42  //_____UART module Initialized_____//
43  /**функция для передачи одного байта по UART**//
44  void UART_send_char(char bt)
45  {
46    while(!TXIF); // придерживаем выполнение программы до тех пор пока буфер
    передачи не освободится
47    TXREG = bt; //записываем в буфер передачи символ, который нужно передать
48  }
49  //_____End of function_____//
50  /**функция для приема одного символа из UART**//
51  char UART_get_char()
52  {
53    if(OERR) // проверка на ошибку
54    {
55      CREN = 0; //If error -> Reset
56      CREN = 1; //If error -> Reset
57    }
58    while(!RCIF); // придерживаем выполнение программы до тех пор пока буфер
    приема не освободится
59    return RCREG; //принимаем значение из буфера приема и передаем его в
    основную программу
60  }
61  //_____End of function_____//
62  /**функция для передачи строки символов**//
63  void UART_send_string(char* st_pt)
64  {
65    while(*st_pt) //if there is a char
66    {
67      UART_send_char(*st_pt++); //process it as a byte data
68    }
69  }

```

```

70 // _____End of function _____//
71 // *****START of Main Function*****//
72 void main(void)
73 {
74     int get_value;
75     TRISB = 0x00; //инициализируем PortB на вывод данных
76     Initialize_UART(); // инициализируем модуль UART
77     UART_send_string("UART Module Initialized and active"); // приветственное
сообщение
78     while(1) //бесконечный цикл
79     {
80         get_value = UART_get_char();
81         if (get_value == '1') //если пользователь передал "1"
82         {
83             RB3=1; // включаем светодиод
84             UART_send_string("RED LED -> ON"); //передаем уведомление на
85 компьютер
86             UART_send_char(10);//ASCII value 10 is used for carriage return (to print in
new line)
87         }
88         if (get_value == '0') // если пользователь передал "0"
89         {
90             RB3=0; // выключаем светодиод
91             UART_send_string("RED -> OFF"); // передаем уведомление на компьютер
92             UART_send_char(10);//ASCII value 10 is used for carriage return (to print in new
93 line)
94         }
95     }
96 }
97 // *****END of Main Function*****//
98
99
100
101
102

```

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

Видео, демонстрирующее работу проекта



Загрузка...

1 844 просмотров