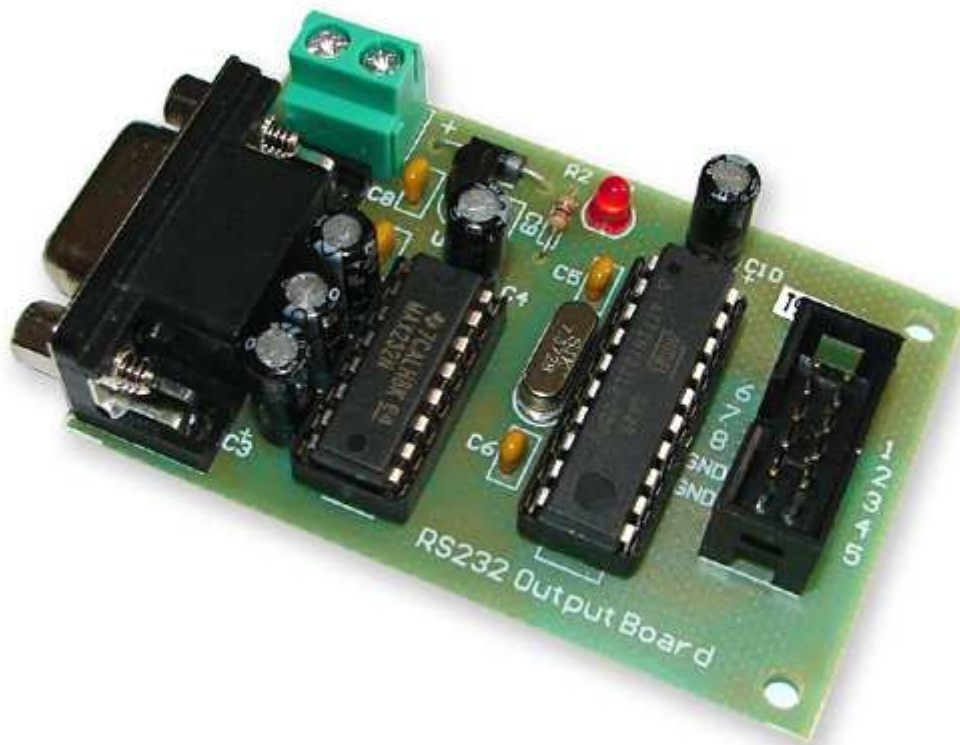


RS232 serial 8 Digital Channel Output Module



© 2010 KMtronic

Descriptions:

RS232 Output Board
8 Digital Outputs
Assembled and Tested

Specifications :

MCU : Atmel ATtiny2313
Interface: RS232
Power: 9Vdc - 15Vdc / 50mA
Baund : 9600
LED: Red led power indicator
Dimensions : 40 mm x 60 mm
Command Protocol : HEX
Temperature: 0° to 70°C (32° to 158°F) Commercial Temperature Range

Communication Parameters :

8 Data, 1 Stop, No Parity

Baud rate : 9600

FIRST chanel commands:

OFF command : FF 01 00 (HEX) or 255 1 0 (DEC)

ON command : FF 01 01 (HEX) or 255 1 1 (DEC)

...

...

...

EIGHT chanel commands:

OFF command : FF 08 00 (HEX) or 255 8 0 (DEC)

ON command : FF 08 01 (HEX) or 255 8 1 (DEC)

...

...

...

Command to turn ON ALL relays : FF 0A FF (HEX) or 255 10 255 (DEC)

Command to turn OFF ALL relays: FF 0A 00 (HEX) or 255 10 0 (DEC)

Each bit of last byte is state of outputs - FF 0A (0-FF)(HEX) or 255 10 (0-255) (DEC).

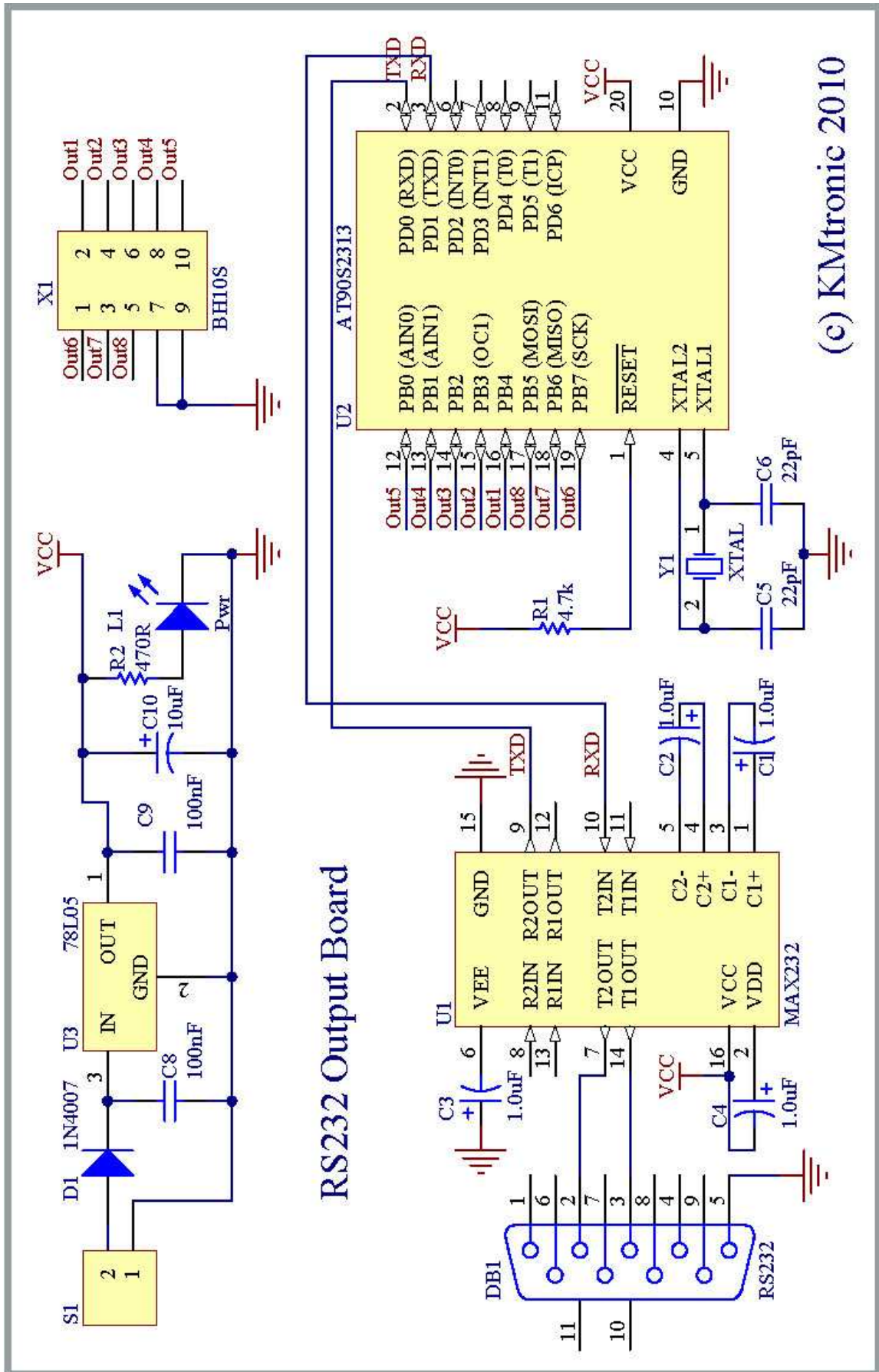
Example:

If you want to turn 3 and 5 relays ON you have to send
FF 0A 14 (HEX) or 255 10 20 (DEC).

If you want to turn on 1, 3, 5, 7 relays you have to send FF 0A 55 (HEX) or 255 10 85 (DEC).

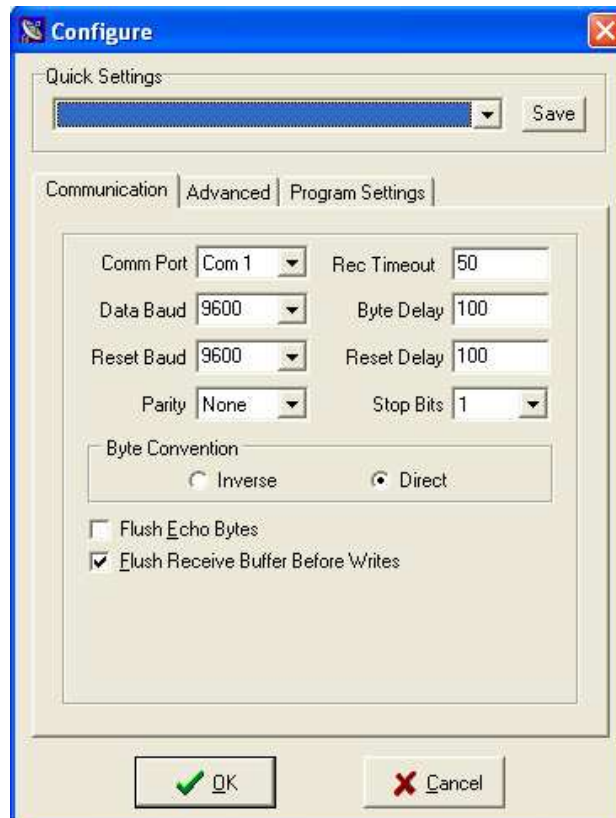
and so on..

Schematics:

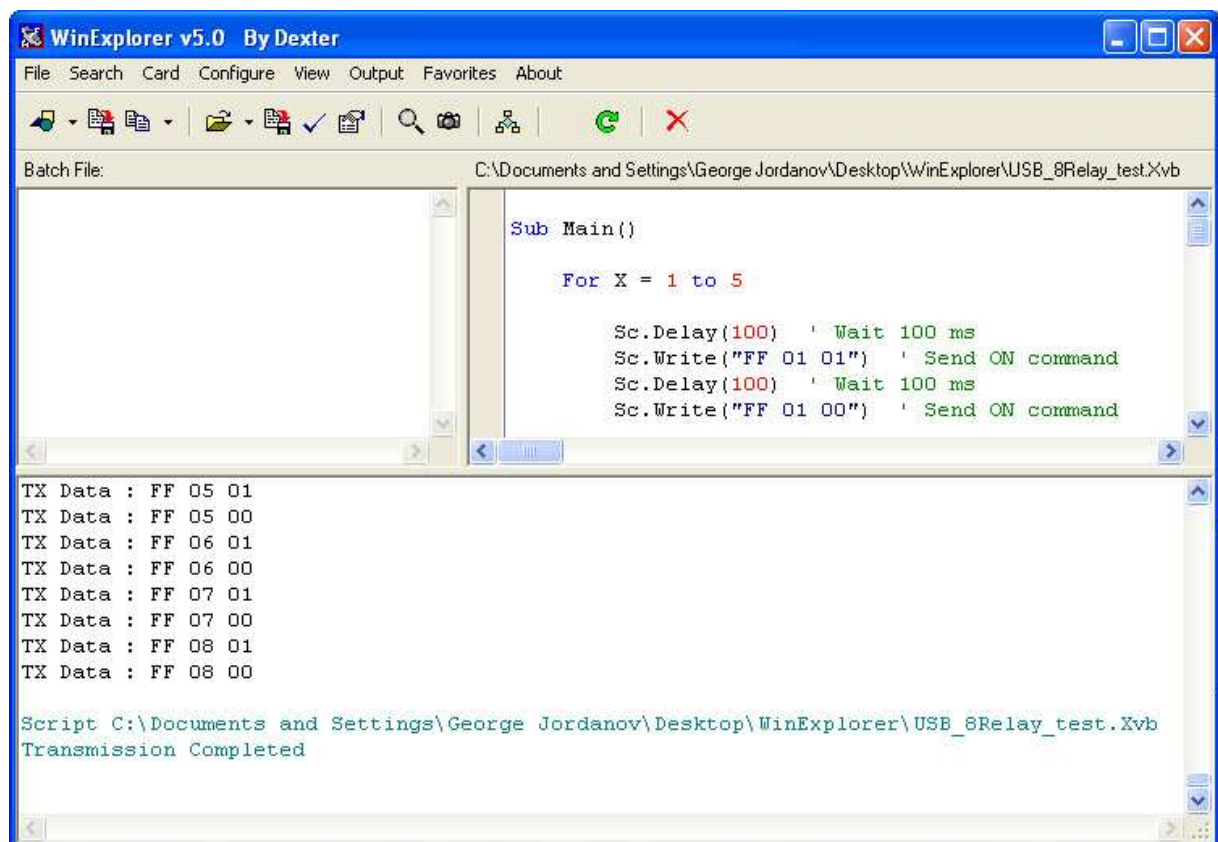


(c) KMtronic 2010

Software:
WinExplorer :



Click File -> Open -> Open Script and open USB_8Relay_test.Xvb



http://www.sigma-shop.com/software/usb_relay/WinExplorer.zip

Sample VB6 code :

```
Private Sub cmdOff_Click()  
    With MSComm1  
        'make sure the serial port is open  
        If .PortOpen = False Then .PortOpen = True  
        'send the data  
        .Output = Chr$(255)  
        .Output = Chr$(1)  
        .Output = Chr$(0)  
    End With 'MSComm1  
End Sub
```

```
Private Sub cmdOn_Click()  
    With MSComm1  
        'make sure the serial port is open  
        If .PortOpen = False Then .PortOpen = True  
        'send the data  
        .Output = Chr$(255)  
        .Output = Chr$(1)  
        .Output = Chr$(1)  
    End With 'MSComm1  
End Sub
```

Sample C# code :

```
private void button1_ON_Click(object sender, EventArgs e)  
{  
    serialPort1.Write(new byte[] { 0xFF, 0x01, 0x01 }, 0, 3);  
}  
  
private void button1_OFF_Click(object sender, EventArgs e)  
{  
    serialPort1.Write(new byte[] { 0xFF, 0x01, 0x00 }, 0, 3);  
}
```

Linux :

The USB-serial device is automatically detected and mapped to /dev/ttyUSB0 (or USB1 in case there is already a similar device).

My test script: (*Thanks Julian!*)

```
-----  
# cat relay.sh  
while true  
do  
echo -e "\xFF\x00\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x00\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x00\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x00\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x00\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x01\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x02\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x03\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x04\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x05\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x06\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x07\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x08\x01" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x01\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x02\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x03\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x04\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x05\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x06\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x07\x00" > /dev/ttyUSB0 ; sleep .1  
echo -e "\xFF\x08\x00" > /dev/ttyUSB0 ; sleep .1  
done  
-----
```