

Project: Connect atmega16 to PC using visual studio C++

Hardware: Stk500, atmega16, serial cable

Software: WinAvr (programmers notepad), AVRStudio V.4.10 and Visual studio C++ V.6.0

It is not essential that you use WinAvr but the guys who made this work is genius and since they deliver it for free, I myself feel I should use it. But every compiler you feel comfortable with will do (must be c compiler).

My project will be simple because I intend to learn how to control both sides. I will not go into details about how to compile and use visual studio, I will start from where I have my skills, and that is a little about WinAvr and a little about visual studio ☺ .

The project will do this:

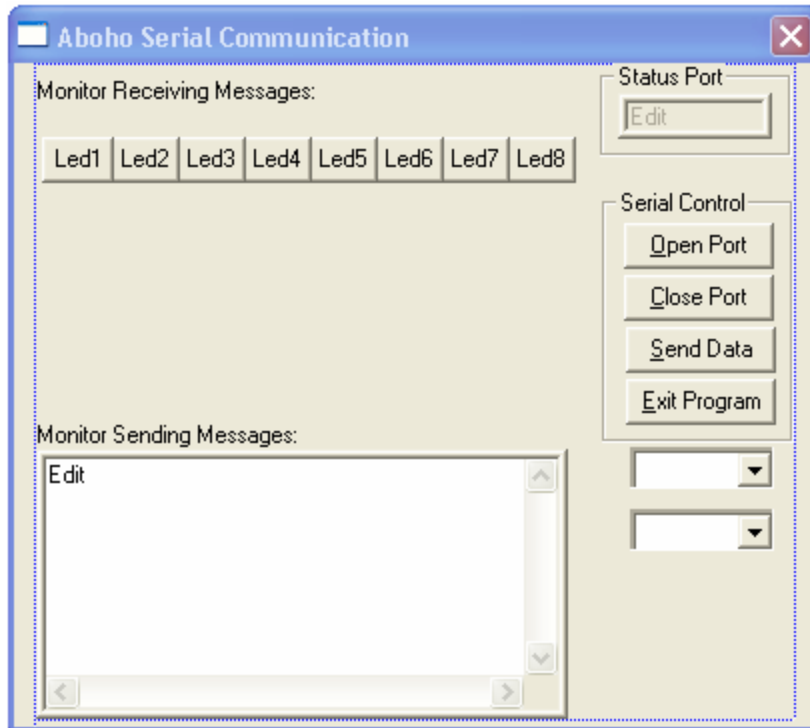
On PC side: There will be a program which is capable of switching between all the baudrate which is possible for the microcontroller. And there will be 8 buttons which controls each led on the STK. There will also be a window telling which button on the STK is pressed.

On STK side: A program which can handle the information send from the PC, and decide which led will be lightening. Also there will be a sending routine to tell which button is pressed.

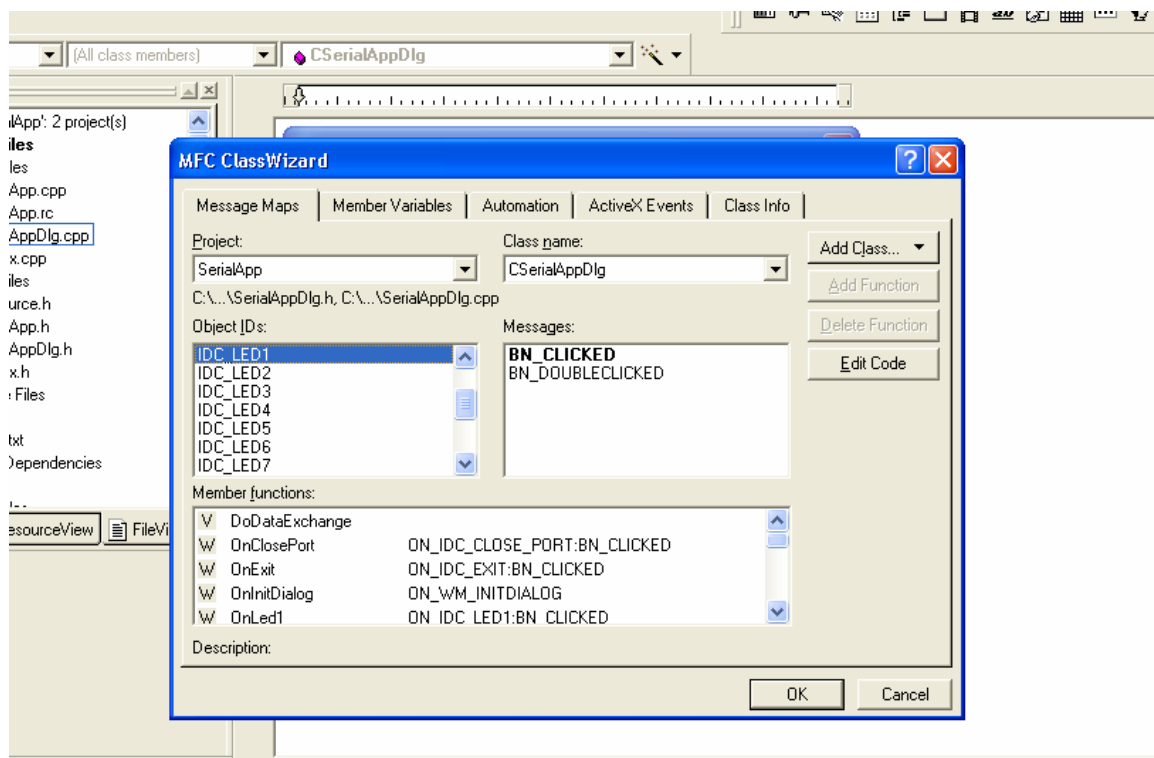
## VisualStudio

Ok in this case I have taken a little shortcut, because I don't want to develop the wheel ones more. And since there is some very genius guys out there who have done this in a project earlier I will use their knowledge. All the genius guys and girls (sorry) in visual studio have developed a nice homepage: [www.codeproject.com](http://www.codeproject.com), and it was here I picked up the original routine I will use: [http://www.codeproject.com/system/chaiyasit\\_t.asp](http://www.codeproject.com/system/chaiyasit_t.asp) . so lets start the visual studio and have fun.

The program will work just as it is, but we have to insert some buttons like this



After this is done we need to build some functions for the buttons to work, this done with class wizard, if you are inside the resource view (I think it will work any place) then you press w+ctrl and this window will pop up.



Here you press IDC\_LED1 and BN\_CLICKED and then add function, the visual studio will come up with a default name and this is as good as any so why not stick to it (i will

do it in this project). The fun thing now is that when you close the wizard and go into the class CSerialAppDlg then you will find all this functions, and it is here we will put in code so that we are able to switch on lights.

```
void CSerialAppDlg::OnLed1()
{
    // TODO: Add your control notification handler code here
    led = '1';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed2()
{
    // TODO: Add your control notification handler code here
    led = '2';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed3()
{
    // TODO: Add your control notification handler code here
    led = '3';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed4()
{
    // TODO: Add your control notification handler code here
    led = '4';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed5()
{
    // TODO: Add your control notification handler code here
    led = '5';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed6()
{
    // TODO: Add your control notification handler code here
    led = '6';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed7()
{
    // TODO: Add your control notification handler code here
    led = '7';
    UpdateData(TRUE);
    sendActivate = true;
}

void CSerialAppDlg::OnLed8()
{
    // TODO: Add your control notification handler code here
    led = '8';
    UpdateData(TRUE);
    sendActivate = true;
}
```

If we put in the code as above, we will in the first line set the number of the led to be switched on, the UpdateData(TRUE) (opening ports and stuff, but I will not go into that here) is a function call which will start the process, and sendActivate is variable which the sending function will check such that it knows it can do a send.

**Note : that the variable led have to be initialized in the header file: CSerialAppDlg.h**

All we do now is sending the led variable like this:

```
SerialThread::Run()
{
    // Check signal controlling and status to open serial communication.
    while(1)
    {

        // Start process of serial communication operation.
        while(ptrDlg->activeProccess == TRUE)
        {
            // enter if there is command of opening and port has be closed before.
            if ((SCC::serialCtl().getStatusPort() == FALSE) &&
                ptrDlg->openPortActivate)
            {
                // open port by calling api function of class serialCtl.
                if (SCC::serialCtl().openPort(ptrDlg->configSerial_,
                    ptrDlg->m_namePort) == TRUE)
                {
                    // Indicate message to status moditor that commnication connected already.
                    ptrDlg->SetDlgItemText(IDC_STATUS_PORT,"Connected");

                }
            }
            else
            {
                // Have problem since opening serial communication.
                ptrDlg->activeProccess = FALSE;
            }
        }
        else if (ptrDlg->openPortActivate)
        {
            char mess[MAX_MESSAGE];
            unsigned int lenBuff = MAX_MESSAGE;
            unsigned long lenMessage;
            static CString outPut;
            if (SCC::serialCtl().read_scc(mess,lenBuff,lenMessage) == TRUE)
            {
                {
                    if (lenMessage > 0)
                    {
                        outPut = '1';
                        ptrDlg->SetDlgItemText('1',outPut);
                    }
                }
            }
            else
            {
                ptrDlg->activeProccess = FALSE;
            }
        }
    }

    // Check signal controlling to send data.
    if (ptrDlg->sendActivate && (ptrDlg->led.GetLength() > 0))
    {

        unsigned long len;

        SCC::serialCtl().write_scc(ptrDlg->led ,
            ptrDlg->led.GetLength(),len);
        ptrDlg->sendActivate = false;
    }
}
```

```

    }

    // Check status and signal controlling to close serial communication.
    if (ptrDlg->closePortActivate)
    {
        if (SCC::serialCtl().closePort() == TRUE)
        {
            // Show message that close when performing of closing port okay.
            ptrDlg->SetDlgItemText(IDC_STATUS_PORT,"Closed");
            ptrDlg->closePortActivate = false;
        }
    }
}
}
return 0;

```

This routine is found in the SerialThread.cpp file, and the only thing I have fixed is this:

```

if (ptrDlg->sendActivate && (ptrDlg->led.GetLength() > 0))
{

    unsigned long len;

    SCC::serialCtl().write_scc(ptrDlg->led ,
        ptrDlg->led.GetLength(),len);

```

because we now need to check the variable led and not the m\_monitorSend. This little change will send the led variable instead. So now we are done with the VisualStudio work.

## WinAvr

Now we need to make the program for receiving, and since we all now how to build a serial routine for the USART in Atmega16 I will just show the code:

```

//-----
// Project:          Assignment6_1_4-6

// Version:          V1.01
// Date:             08 Mars 2003
// Author:           Roald, Kim & Hans inc.
// File:             ass_14.c
// Description:      this will catch the led variable from the PC and then decide
//                  what led will be switched on.
//-----

//*****
// include files
//*****

// registers and interrupts definitions
#include <avr/io.h>                                // include library    for
microcontroller
#include <avr/interrupt.h>                          // include library    for enable interrupt
#include <avr/signal.h>                              // include library    for
interrupt vector

//*****
// Definitions

```

```

//*****

#define key (PINA & 0x01)
#define key1 (PINA & 0x02)
#define key2 (PINA & 0x04)

//*****
// function prototypes
//*****

void delay (int ms); // declaration for delay
void sendposmsg (int y, int x, char*s); // declaration for sendposmsg

//*****
// constants
//*****

const unsigned int baudrate = 23; // baudrate = 9600

//*****
// Global structs
//*****

//*****
// Global variables
//*****

// int

unsigned int hypsend,test;
unsigned int sendhyp;

// double
double val,tall;

// arrays
/*
unsigned char header[] = {"this is the chef program"}; // message 1 to the pc
unsigned char msg0[] = {"wrong key pressed"}; // message to the pc
unsigned char msg1[] = {"you pressed key 0"}; // message 1 to the pc
unsigned char msg2[] = {"you pressed key 1"}; // message 1 to the pc
unsigned char msg3[] = {"you pressed key 7"}; // message 1 to the pc
unsigned int aray[5] = {0xfb, 0xf7, 0xef, 0xdf, 0xbf};
*/

unsigned char queue[50]; // character buffer

unsigned int hyper[4]; // buffer
for position codes for hyperterminal

// chars
unsigned char qcncr, sndcncr,ch;

//*****
// Interrupts
//*****

// interrupt for transmitt
//SIGNAL(SIG_UART_TRANS)
SIGNAL(SIG_UART_DATA)
{
// text string
if ( qcncr != sndcncr )
{
UDR = queue[sndcncr++]; // send next character
}
}

```

```

        //PORTB = sndcnt;
        // and increment index
    }
}

// interrupt for resive
SIGNAL(SIG_UART_RECV)
{
    if( UDR == '1' )                // test for key 7 is
pressed
    {
        PORTB = 0xfe;
    }
    if( UDR == '2' )                // test for key 7 is
pressed
    {
        PORTB = 0xfd;
    }
    if( UDR == '3' )                // test for key 7 is
pressed
    {
        PORTB = 0xfb;
    }
    if( UDR == '4' )                // test for key 7 is
pressed
    {
        PORTB = 0xf7;
    }
    if( UDR == '5' )                // test for key 7 is
pressed
    {
        PORTB = 0xef;
    }
    if( UDR == '6' )                // test for key 7 is
pressed
    {
        PORTB = 0xdf;
    }
    if( UDR == '7' )                // test for key 7 is
pressed
    {
        PORTB = 0xbf;
    }
    if( UDR == '8' )                // test for key 7 is
pressed
    {
        PORTB = 0x7f;
    }
}

//-----*****-----
// Initialization
//-----*****-----
// Delay routine
void initialize (void)
{
    // ports
    DDRA = 0x00;
}
// porta as in

```

```

DDRB = 0xFF;
PORTB = 0xFF;

// portd as out
// set default portb

// UART

/* Enable receiver and transmitter */
UCSRB = (1<<RXCIE)|(1<<TXEN)|(1<<RXEN)|(1<<UDRIE);

//UBRR = baudrate; // set baudrate =
9600
UBRRH = 0x00; // baudrate for atmega16
UBRRL = baudrate;

UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ0)|(1<<UCSZ1);

// Global enable interrupts
sei();
//testing

}

#####
// Functions
#####

// delay routine
void delay (int ms)
{
    int i,k,l;
    for (l=0; l<ms*5;l++)
    {
        for (i=0; i<199; i++)
            k++;
    }
}

// set position and send textstring to hyperterminal
void sendposmsg (int y, int x, char*s)
{
    sndcptr = 1;
    qcptr = 0;
    queue[qcptr++] = 0x1b;
    //queue[qcptr++] = 'Y'; // ESC // Y
    queue[qcptr++] = 32 + y; // [y+32]
    queue[qcptr++] = 32 + x; // [x+32]
    while (*s)
    {
        queue[qcptr++] = *s++;
    }
    UDR = queue[0];
}

#####
// Main
#####

void main(void)
{
    initialize(); // call initialize function

    //sendposmsg(0,0,header);

    while (1)

```

```
    {  
        //if(key == 0) sendposmsg(1,1,header);  
        //delay(500);  
    }  
}
```

Now we are good to go, good luck.....