Application Note: Demonstrating CAN with nqBASIC

By Carl Barnes, Technological Arts, Inc.
May 21, 2011

Hardware used:
2 – NC12MAX Modules,
or 2 – NC12DX with 2 external user-added CAN transceivers
2 – School Board, Docking Module, or similar
(2 – LEDs and a 2K Ohm resistor => if not using Docking Module or School Board)

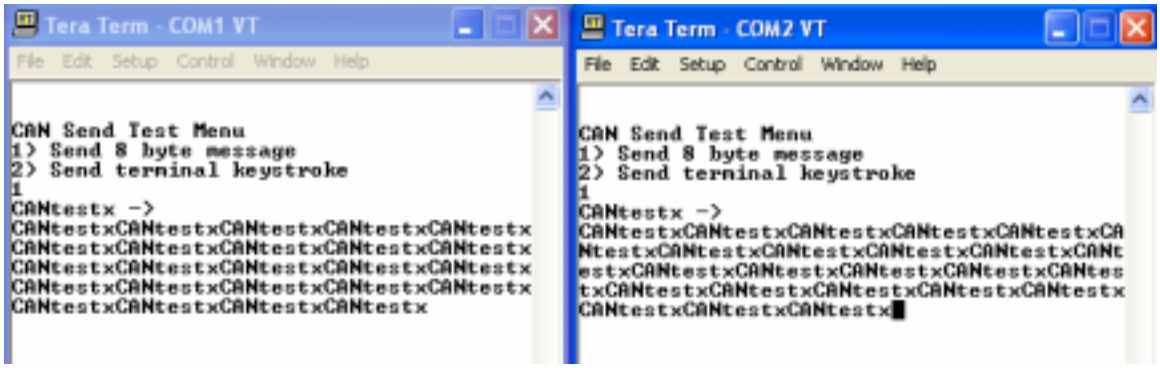
Wire up the circuit shown in the schematic (at the end of this document). Ensure proper power wiring; otherwise, damage may occur to the electronics.

This example shows how to setup the CAN and SCI functions. One module will send the message "CANTestx" repeatedly, and at the same time display a message on a terminal program to show that it was sent. The other module will receive the CAN messages and display them in a second terminal window to show that they were received.

Create the nqBASIC project using the code listed below, build it, and load it into each of the two NanoCore12 modules. Attach one of your NanoCore12 setups to each COMport, power them up, and run them. Don't forget to connect a cable between the CAN connectors on each setup. For the Docking Module, be sure to place JB2 and JB3 jumpers in the "1-2" position (i.e. towards the CAN connector, J3). For a short CAN cable, the termination resistors are not necessary, so JB4 jumpers are optional.

Launch a terminal program (HyperTerminal, TeraTermPro, or similar) on two nearby PCs. If you have two serial ports, you can use one PC by launching two instances of the terminal program-- one for each COMport. In any case, setup the COMport settings for 9600 baud, no parity, 8-bit data, and 1 stop bit (no handshaking).

Reset each hardware setup, and you'll see the menus displayed. In each terminal window, type the digit "1". Each setup will begin transmitting its message to the other, blinking the LED each time the message is sent. See screenshot, below:

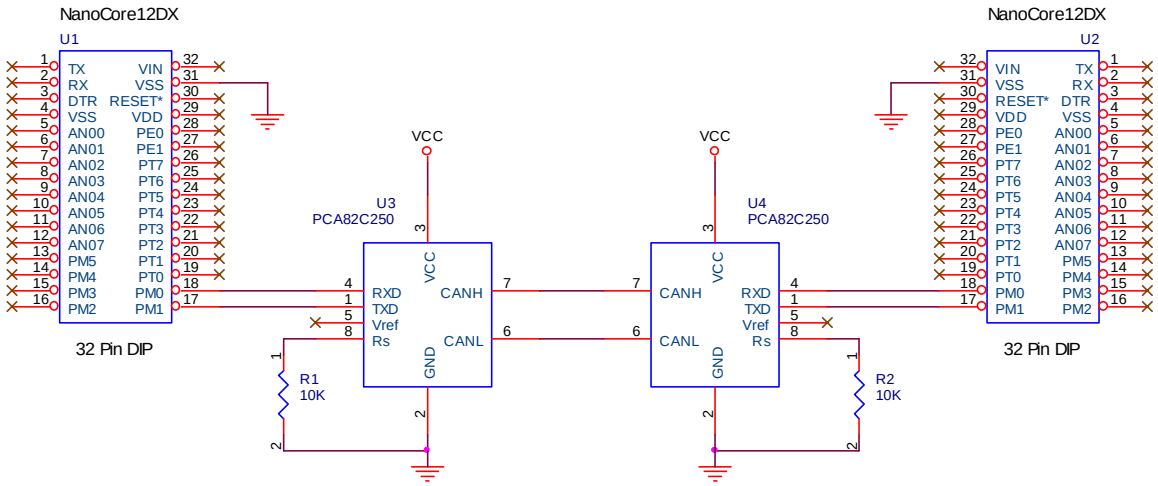


```
/*
CAN Communication Between Two NanoCore12 DX or MAX Modules
==========================================================
*/
/*
Use LED on PT0 or PT1 with School Board; PT2 or PT3 with Docking Module
*/
Const LED = PT1Const DEL_1MSEC = 1000

dim S as new SCI (RX_SCI, TX_SCI)    //SCI object
dim D as new DIO (LED)               //Discrete I/O object on LED on PT1-pin
dim C as new CAN (RXCAN, TXCAN)

//Misc variables
dim done as new BYTE
dim Char as new BYTE   //Variable to store received characters
dim Index as new byte

//Variables for CAN
dim canbuffer[16] as new BYTE
dim txbuffer[8] as new BYTE
dim rxbuffer[8] as new BYTE
dim length as new BYTE
dim extended as new BYTE
dim identlow as new WORD
dim identhigh as new WORD


//Note that max delay is MAX_WORD (65536) milliseconds (just over 1 minute).
sub DelayMsec (in word milliseconds)
       while (milliseconds > 0)
              System.Delay (DEL_1MSEC)      //Delay 1000 microsecond to make 1
millisecond
              milliseconds = milliseconds - 1
       end while
end sub


/*
**     Print the menu over the serial port.
*/
sub PrintMenu ()
       S.SER_Put_char ('\n')
       S.SER_Put_char ('\r')
       S.SER_Put_string ("1) Send 8 byte message")
       S.SER_Put_char ('\n')
       S.SER_Put_char ('\r')
       S.SER_Put_string ("2) Send terminal keystroke ")
end sub

/*
       Process the command received on the serial port
*/
sub ProcessCommand (in byte cmd)
       select cmd
              case '1':             //Send test message
                     //Init send data for Echo test
                     S.SER_Put_char ('\n')
              S.SER_Put_char ('\r')
                     S.SER_Put_string ("CANtestx -> ")
                     txbuffer[0] = 'C'
                     txbuffer[1] = 'A'
                     txbuffer[2] = 'N'
                     txbuffer[3] = 't'
                     txbuffer[4] = 'e'
                     txbuffer[5] = 's'
                     txbuffer[6] = 't'
                     txbuffer[7] = 'x'
                     done = 1
                     break
```

```
            case '2':          //Send single keystroke from terminal
                       S.SER_Get_char (1, Char)
                       S.SER_Put_char ('\n')
                   S.SER_Put_char ('\r')
                       S.SER_Put_char (Char)
                       S.SER_Put_string (" ")
                       txbuffer[0] = Char
                       txbuffer[1] = 0
                       txbuffer[2] = 0
                       txbuffer[3] = 0
                       txbuffer[4] = 0
                       txbuffer[5] = 0
                       txbuffer[6] = 0
                       txbuffer[7] = 0
                       done = 1
                       break

                   default:
                       done = 0
                       break
        end select
end sub


sub CANTest ()
        while (FOREVER)
                //Send txbuffer - node ID = 2
                C.CAN_Send (CAN_TX_BUFFER0, CAN_STANDARD, 0, 0x40, 0, 8, txbuffer)

                //Wait for echo
                C.CAN_Receive (canbuffer, length, extended)

                //Turn LED ON
                D.PIN_Out (LED, ON)

                //Get data out of CAN buffer
                CAN.CAN_Rec_data (canbuffer, rxbuffer, 8)

                //Echo received data to terminal
                Index = 0
                do
                        Char = rxbuffer[Index]
                        S.SER_Put_char (Char)
                        Index = Index + 1
                until (Index == length)

                //Turn LED OFF after 500 millisecond delay and repeat
                DelayMsec (500)
                D.PIN_Out (LED, OFF)
        end while
end sub

main
        System.INTS_On ()

        //Setup CAN receive filters (while in initialization mode) BEFORE calling
CAN_Setup!
        C.CAN_Filter (0, 0xFF, 0)
        C.CAN_Filter (1, 0x3F, 0x20)
        C.CAN_Filter (2, 0, 0)
        C.CAN_Filter (3, 0, 0)
        C.CAN_Filter (4, 0, 0)
        C.CAN_Filter (5, 0, 0)
        C.CAN_Filter (6, 0, 0)
        C.CAN_Filter (7, 0, 0)

        //Setup for 125kHz (with 8mHz crystal), and four 16-bit receive filters.
        C.CAN_Setup (0, CAN8MHZ_BAUD_125KHZ, CAN_FILTERS_4_16BIT)

    //Setup UART buffer size and baud rate
        S.SER_Setup (8, BAUD9600)
```

```
        //Send banner
        S.SER_Put_char ('\n')
        S.SER_Put_char ('\n')
        S.SER_Put_char ('\r')
        S.SER_Put_string ("CAN Send test")

        while (FOREVER)
               PrintMenu ()                          //Print menu every time
               S.SER_Get_char (1, Char)              //wait for character to be received
               S.SER_Put_char (Char)          //echo it back
               ProcessCommand (Char)          //check if it is a command
               if (done == 1)
                      CANTest ()
               End if
        end while
end main
```

Schematic Diagram for CAN Demo Hardware Setup
NanoCore12DXC32



Schematic Diagram for CAN Demo Hardware Setup
NanoCore12MAXC32/128