David and Jacqui's Wiki

electronics:micromite_wiznet_library

Micromite Wiznet Library

Hardware Connections

Wiznet	Micromite	Notes
Ground	Pin 19	
+5V	+5V	
RST	Pin 23	Wiznet W5100 Reset
SS	Pin 15	SPI Slave Select
CLK	Pin 25	SPI Clock
MO	Pin 3	SPI MOSI - Master Out, Slave In
MI	Pin 14	SPI MISO - Master In, Slave Out
Gnd		not used
PoE+		not used
PoE-		not used

Required Code

Note: The wiznet module may not work when connected to a PoE switch - symptoms seen: gets Power LED but never establishes an ethernet connection to the Local Area Network (LAN).

The following code snippet (plus the library) is the minimum you require to get a ping response from the Micromite with Wiznet:

```
Option explicit
Option base 1

dim ip1%, ip2%, ip3%, ip4%

' Call with pin used for Wiznet Reset and SPI Slave Select of Wiznet:
WN.Init 23, 15

' Set static MAC address/IP address/subnet mask/gateway here
ip1%=192
ip2%=168
ip3%=1
ip4%=65
WN.SetMAC &H00,&HFE,ip1%,ip2%,ip3%,ip4%
WN.SetIP ip1%,ip2%,ip3%,ip4%
WN.SetMask 255,255,255,0
WN.SetGateway ip1%,ip2%,ip3%,1

WN.Start    ' Sets various internal registers for Wiznet
```

Notes:

1. If you change to “Option base 0” (which starts array numbering at 0, not 1), set “dim integer WN_OptionBase=0” in WN.Init or all SPI receives will fail.
2. You need to make sure the MAC address is unique to every host on your local network. This sample code sets the MAC address to &H00FEwwxxyyzz where ww.xx.yy.zz represent the IP Address. That way the code will work across multiple devices where you change only the IP address by setting ip1%, ip2%, ip3% and ip4%. If you produce a commercial product, you will need to apply for your own MAC address allocation.

When you can ping your Micromite

From here, up to four sockets (0, 1, 2, and 3) can be initialized and used for either TCP, UDP, IP-Raw or MAC-Raw (only TCP implemented in Micromite library at present) concurrently. In other words, you could have a webserver “Listening” on one TCP server socket on the Micromite that gives users access to a small web page showing analog pin values or temperature/humidity data. You could have another socket establishing a TCP client socket connection to a remote webserver data logging your temperature/humidity every 60 seconds.

The Micromite library allocates 2048 bytes to each send and 2048 bytes to each receive buffer on the Wiznet module (in WN.Start subroutine).

Sockets

Library Support	Protocol	Title	Notes
Yes	TCP	Transmission Control Protocol	Guaranteed delivery, eg: Telnet to a server; HTTP - accessing (client) or hosting (server) web sites; etc.
Yes	UDP	User Datagram Protocol	Very basic IP protocol - can send and receive packets but no guarantee they will go through or will arrive in order. Great if you want a simple one-packet reply quickly or for data logging applications where it doesn't matter if you miss a packet every so often.
No	IP-Raw	Raw TCP/IP Packet	Write your own IP sub-protocol! This is what you'd use to implement DHCP
No	MAC-Raw	Raw MAC level Packet	Write your own ethernet protocol! You'd need to come up with a routing protocol to get past your local network. Unlikely anybody would ever want to use this in a Micromite

When initialized, the Wiznet module sets all four sockets to “Closed”. From there, they can be “Initialized” to TCP or “Opened” to UDP, IP Raw or MAC Raw. When a socket is “Open”, data can be sent and received using the WN.SockWrite subroutine and WN.SockRead function respectively.

TCP server

```
'If socket closed, start listening for incoming connections
if (WN.SockClosed(0)=1) then
  if (WN.SockTCPInit(0, 23)=1) then
    print:print:print "Socket initialized on local port 23"
  else
    print:print:error "Failed to initialize socket"
```

```
  end if
```

WN.SockClosed(<socket number 0-4>) Function

Check Socket Closed Status.

Call with the socket number.

Returns 1 if socket is Closed or 0 if socket is Open.

WN.SockTCPInit(<socket number 0-4>, <local port number 0-65535>) Function

Initialize TCP Socket.

Call with the socket number and local port number to use. Port 23 is used for Telnet, port 80 for HTTP, etc. Here are some more:

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml [https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml]

You can use any port for testing, but typically keep them to below 1024 for TCP servers.

Returns 1 if socket inialized or 0 if socket is socket fails to initialize.

```
  if WN.SockTCPListen(0) then
    print "Socket listening for incoming connections - telnet to IP address"
  else
    error "Failed to listen on socket"
  end if
```

WN.SockTCPListen(<socket number 0-4>) Function

Attempt to listen on TCP Socket.

Call with the socket number.

Returns 1 if socket is Listening or 0 if socket fails (Note: if the Listen fails, you need to reinitialize the Socket with WN.SockTCPInit function before retrying the WN.SockTCPListen).

```
do while WN.SockData(0)
  print chr$(WN.SockRead(0));
loop

'Send some data to the client
if WN.SockEstablished(0) then
  txstring$="0123456789"
  WN.SockWrite 0, txstring$
end if
```

WN.SockData(<socket number 0-4>) Function

Check if there is data available in a socket buffer on the Wiznet module.

Call with the socket number.

Returns the number of bytes available in the receive buffer for the socket. There may be data remaining in a socket even after a TCP connection is “Established”.

WN.SockRead(<socket number 0-4>) Function

Read next byte from socket buffer on the Wiznet module.

Call with the socket number.

Returns the next data byte as an integer from the buffer for the socket. Do not call unless there is data waiting or corruption will occur - check sockData() function first.

WN.SockEstablished(<socket number 0-4>) Function

Checks if a TCP socket is established - if it is, data can be sent. This function also finalises a disconnection if the remote party has requested disconnection, so good practice dictates that it should be called in a loop while connections are established.

Call with the socket number.

Returns 1 if TCP socket established or 0 if socket is not established (in other words, 1 if there is a current connection to another device for this socket).

WN.SockWrite <socket number 0-4>, <string to send> Subroutine

Send data over an established socket.

Call with socket number and a string of characters to send.

WN.SockClose <socket number 0-4> Subroutine

Close an open socket.

Call with socket number.

TCP client

```
if WN.SockTCPConnect(0,192,168,1,14,80) then ' http://192.168.1.14
  print "Successful connection to webserver on port 80"
else
  error "Failed to connect to webserver on port 80"
end if
```

WN.SockTCPConnect(<socket number 0-4>, <Remote IP Octet 1>, <Remote IP Octet 2>, <Remote IP Octet 3>, <Remote IP Octet 4>, <Remote Port 0-65535>) Function

Attempt to establish a TCP connection to a remote host. Function may take up to 4 seconds to return as it waits for a connection.

Call with the socket number, the remote IP address octets and the remote port number to connect to (eg. port 80 for a standard webserver running HTTP). Note: If using client connection, initialize the socket with a high source port number, say between 63999 and 65536, eg. SockTCPInit(0, 64000).

Returns 1 if TCP client connection successful or 0 if TCP client connection fails.

```
do while WN.SockData(0)
  print chr$(WN.SockRead(0));
loop
```

Receiving data is the same as TCP server - refer to WN.SockData and WN.SockRead functions above.

```
    WN.SockWrite 0, "GET / HTTP/1.1"+chr$(&H0D)+CHR$(&H0A)
    WN.SockWrite 0, "Host: 192.168.1.14"+chr$(&H0D)+CHR$(&H0A)
    WN.SockWrite 0, "User-Agent: Micromite!"+chr$(&H0D)+CHR$(&H0A)
    WN.SockWrite 0, "Connection: close"+chr$(&H0D)+CHR$(&H0A)
    WN.SockWrite 0, chr$(&H0D)+CHR$(&H0A)
```

Similarly, sending data is the same as TCP server - refer to WN.SockEstablished function and WN.SockWrite subroutine above.

UDP

WN.SockUDPInit(socket%, srcport%) Function

Initialize socket for UDP traffic. Once this is done, you should be able to receive data using WN.SockRead function.

Note: An eight byte header is prepended to UDP packet data: Dest IP address (4 bytes), Dest port (2 bytes), Data size (2 bytes). It will be read using the WN.SockRead function before the actual packet data.

WN.SockUDPDest (<socket number 0-4>, <Remote IP Octet 1>, <Remote IP Octet 2>, <Remote IP Octet 3>, <Remote IP Octet 4>, <remote port number 0-65535>) Subroutine

Set destination IP address and port number for UDP traffic. Once this is done, you should be able to send data using WN.SockWrite function. Use a high source port number (eg. 64000) in WN.SockUDPInit to enable sending UDP packets.

Examples (library not included)

Webserver

```
Option explicit
Option base 1

dim ip1%, ip2%, ip3%, ip4%
Dim example$
Dim txstring$
Dim curLineBlank%, curFirstLine%
Dim char%
Dim firstLine$
Dim i%
Dim conversationStart%

' Call with pin used for Wiznet Reset and SPI Slave Select of Wiznet:
WN.Init 23, 15

' Set a pin as analog input for testing webserver
SetPin 26, ain

' Set up the IP address to use
ip1%=192
ip2%=168
ip3%=1
ip4%=65

' Set static MAC address/IP address/subnet mask/gateway

WN.SetMAC &H00,&HFE,ip1%,ip2%,ip3%,ip4%
WN.SetIP ip1%,ip2%,ip3%,ip4%
WN.SetMask 255,255,255,0
WN.SetGateway ip1%,ip2%,ip3%,1
```

```
WN.Start    ' Sets various internal registers for Wiznet

Do
  If WN.SockClosed(3) Then
    If WN.SockTCPInit(3, 80) Then
      Print "Socket initialized on local port 80"
    Else
      Error "Failed to initialize socket"
    End If
    If WN.SockTCPListen(3) Then
      Print "Server is at "+WN.GetIP$()
      curLineBlank%=1
      curFirstLine%=1
      firstLine$=""
      conversationStart%=0
    Else
      Error "Failed to listen on socket"
    End If
  End If

  Do While WN.SockEstablished(3)
    If conversationStart%=0 Then conversationStart%=Timer
    For i%=1 To WN.sockData(3)
      char%=WN.sockRead(3)
      Print Chr$(char%);
      If char%=&H0D Then
        If curLineBlank%=1 Then
          Print firstLine$
          WN.SockWrite 3,"HTTP/1.1 200 OK"+WN_CrLf$
          WN.SockWrite 3,"Content-Type: text/html"+WN_CrLf$
          WN.SockWrite 3,""+WN_CrLf$
          WN.SockWrite 3,"<!DOCTYPE HTML>"+WN_CrLf$
          WN.SockWrite 3,"<html>"+WN_CrLf$
          WN.SockWrite 3,"<h3>Micromite Webserver</h3>"+WN_CrLf$
          WN.SockWrite 3,"<p>Pin 26: "+Str$(Pin(26))+" volts</p>"+WN_CrLf$
          WN.SockWrite 3,"</html>"+WN_CrLf$
          WN.SockWrite 3,""+WN_CrLf$
          WN.SockClose 3
        End If
      End If
      If char%=&H0D Then
        curLineBlank%=1
        curFirstLine%=0
      Else If char%<>&H0A Then
        curLineBlank%=-1
        If curFirstLine%=1 Then
          firstLine$=firstLine$+Chr$(char%)
        End If
      End If
    Next i%
  Loop
  If conversationStart%>0 Then
    Print "Time: "+Str$((Timer-conversationStart%)/1000)+" seconds"
    conversationStart%=0
  End If
Loop

End

<Include Library here>
```

Connect to http://192.168.1.65 [http://192.168.1.65] in a web browser.

UDP Send and receive

```
Option explicit
Option base 1
```

```
dim ip1%, ip2%, ip3%, ip4%, dport%, looptimer%, i%, char%

' Call with pin used for Wiznet Reset and SPI Slave Select of Wiznet:
WN.Init 23, 15

' Set a pin as analog input for testing webserver
SetPin 26, ain

' Set up the IP address to use
ip1%=192
ip2%=168
ip3%=1
ip4%=65

' Set static MAC address/IP address/subnet mask/gateway

WN.SetMAC &H00,&HFE,ip1%,ip2%,ip3%,ip4%
WN.SetIP ip1%,ip2%,ip3%,ip4%
WN.SetMask 255,255,255,0
WN.SetGateway ip1%,ip2%,ip3%,1

WN.Start    ' Sets various internal registers for Wiznet

If WN.SockUDPInit(3, 64000) Then
  Print "Socket initialized on local port 64000"
Else
  Error "Failed to initialize socket"
End If

If WN.SockUDPInit(2, 124) Then
  Print "Socket initialized on local port 124"
Else
  Error "Failed to initialize socket"
End If

WN.SockUDPDest 3, 192, 168, 1, 14, 124

do
  WN.SockWrite 3,"Test UDP!!! "+str$(TIMER)+WN_CrLf$
  looptimer%=timer
  do
    For i%=1 To WN.sockData(2)
      char%=WN.sockRead(2)
      Print Chr$(char%);
    next i%
    if timer-looptimer%>5000 then exit DO
  loop
loop

End

<Include Library here>
```

Testing UDP transfers on Linux

Listen on UDP port 124 for incoming connections and display incoming data:

```
netcat -ul -p 124
```

Send test UDP packet to 192.168.1.65, port 124:

```
echo -n "UDP Back!!!" | nc -u -w1 192.168.1.65 124
```

Links

Discussion (The Back Shed Forum):

http://www.thebackshed.com/forum/forum_posts.asp?TID=7727&PN=1 [http://www.thebackshed.com/forum/forum_posts.asp?TID=7727&PN=1]

Datasheet for Wiznet module:

https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf [https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf]

eBay link for module:

http://www.ebay.com.au/itm/Durable-TOP-Mini-W5100-Ethernet-Shield-Network-Expansion-Module-for-Arduino-/121594616164 [http://www.ebay.com.au/itm/Durable-TOP-Mini-W5100-Ethernet-Shield-Network-Expansion-Module-for-Arduino-/121594616164]

To do

1. Possibly add string read function to read more than one byte at a time for better performance. Problem is the string variable only has 255 characters in the Micromite and the buffer may be much larger (up to 2048 bytes). One line at a time won't work either - a line may well be longer than 255 characters.

Library

```
'-----------------------------------------------------------------------
' WIZNET W5100 Subs and Functions version 2015-06-16

Sub WN.Init(resetPin%, slaveSelectPin%)
  dim integer WN_OptionBase=1       ' Set to match option base above (if changed)
  dim WN_CrLf$=Chr$(&H0D)+Chr$(&H0A)
  dim WN_SPISpeed%=1000000 '10000000 is max that works
  DIM WN_ResetPin%=resetPin%
  DIM WN_SSPin%=slaveSelectPin%

  ' Set the slave select pin HIGH and to output pin
  Pin(WN_SSPin%)=1
  SetPin WN_SSPin%, dout

  SetPin WN_ResetPin%, dout
  Pin(resetPin%)=0
  ' Wiznet docs: assert low for 2us to reset Wiznet
  Pause 10 ' we wait 10 milliseconds
  Pin(resetPin%)=1
End Sub

Sub WN.Close()
  PIN(WN_SSPin%)=0
  SETPIN WN_SSPin%,OFF
  PIN(WN_ResetPin%)=0
  SETPIN WN_ResetPin%,OFF
END SUB

Sub WN.Start()
  WN.Write(0, 0, &B00000000) ' Mode Register, enable ping
  WN.Write(0, &H16, &B00000000) ' Interrupt Mask Register
  WN.Write(0, &H17, &H0F) ' Retry time value high (400ms)
  WN.Write(0, &H18, &HA0) ' Retry time value low (400ms)
  WN.Write(0, &H19, &H08) ' retry count register
  WN.Write(0, &H1A, &B01010101) ' RX Memory Size (2KB/socket)
  WN.Write(0, &H1B, &B01010101) ' TX Memory Size (2KB/socket)
End Sub
```

```
Function WN.Read(addrHi%, addrLo%)
  Local junk%
  Pin(WN_SSPin%)=0
  SPI open WN_SPISpeed%, 0, 8
  junk%=SPI(&H0F)
  junk%=SPI(addrHi%)
  junk%=SPI(addrLo%)
  WN.Read=SPI(0)
  Pin(WN_SSPin%)=1
  SPI close
End Function

Sub WN.Write(addrHi%, addrLo%, writeData%)
  Local junk%
  Pin(WN_SSPin%)=0
  SPI open WN_SPISpeed%, 0, 8
  junk%=SPI(&HF0)
  junk%=SPI(addrHi%)
  junk%=SPI(addrLo%)
  junk%=SPI(writeData%)
  Pin(WN_SSPin%)=1
  SPI close
End Sub

Sub WN.SetGateway (oct1%, oct2%, oct3%, oct4%)
  WN.Write(&H00, &H01, oct1%)
  WN.Write(&H00, &H02, oct2%)
  WN.Write(&H00, &H03, oct3%)
  WN.Write(&H00, &H04, oct4%)
End Sub

Sub WN.SetMask (oct1%, oct2%, oct3%, oct4%)
  WN.Write(&H00, &H05, oct1%)
  WN.Write(&H00, &H06, oct2%)
  WN.Write(&H00, &H07, oct3%)
  WN.Write(&H00, &H08, oct4%)
End Sub

Sub WN.SetMAC (oct1%, oct2%, oct3%, oct4%, oct5%, oct6%)
  WN.Write(&H00, &H09, oct1%)
  WN.Write(&H00, &H0A, oct2%)
  WN.Write(&H00, &H0B, oct3%)
  WN.Write(&H00, &H0C, oct4%)
  WN.Write(&H00, &H0D, oct5%)
  WN.Write(&H00, &H0E, oct6%)
End Sub

Sub WN.SetIP (oct1%, oct2%, oct3%, oct4%)
  WN.Write(&H00, &H0F, oct1%)
  WN.Write(&H00, &H10, oct2%)
  WN.Write(&H00, &H11, oct3%)
  WN.Write(&H00, &H12, oct4%)
End Sub

Function WN.GetIP$()
  Local oct1%, oct2%, oct3%, oct4%
  oct1%=WN.Read(&H00, &H0F)
  oct2%=WN.Read(&H00, &H10)
  oct3%=WN.Read(&H00, &H11)
  oct4%=WN.Read(&H00, &H12)
  WN.GetIP$=Str$(oct1%)+"."+Str$(oct2%)+"."+Str$(oct3%)+"."+Str$(oct4%)
End Function

Function WN.SockTCPInit(socket%, srcport%)
  WN.Write(socket%+4, &H00, &H01) 'Sn_MR Mode Register
  WN.Write(socket%+4, &H04, srcport% >> 8) 'Sn_PORT Source Port Hi
  WN.Write(socket%+4, &H05, srcport% And &HFF) 'Sn_PORT Source Port Lo
  WN.Write(socket%+4, &H01, &H01) 'Sn_CR Command Register, Open
```

```
  'Sn_SR Status Register
  If WN.Read(socket%+4, &H03)<>&H13 Then
    'Failed to init socket
    WN.SockTCPInit=0
    WN.Write(socket%+4, &H01, &H10) 'Sn_CR Command Register, Close
  Else
    WN.SockTCPInit=1
  End If
End Function

Function WN.SockUDPInit(socket%, srcport%)
  WN.Write(socket%+4, &H00, &H02) 'Sn_MR Mode Register
  WN.Write(socket%+4, &H04, srcport% >> 8) 'Sn_PORT Source Port Hi
  WN.Write(socket%+4, &H05, srcport% And &HFF) 'Sn_PORT Source Port Lo
  WN.Write(socket%+4, &H01, &H01) 'Sn_CR Command Register, Open
  'Sn_SR Status Register
  If WN.Read(socket%+4, &H03)<>&H22 Then
    'Failed to init socket
    WN.SockUDPInit=0
    WN.Write(socket%+4, &H01, &H10) 'Sn_CR Command Register, Close
  Else
    WN.SockUDPInit=1
  End If
End Function

Function WN.SockTCPListen(socket%)
  WN.Write(socket%+4, &H01, &H02) 'Sn_CR Command Register, Listen
  'Sn_SR Status Register
  If WN.Read(socket%+4, &H03)<>&H14 Then
    'Failed to listen
    WN.SockTCPListen=0
    WN.Write(socket%+4, &H01, &H10) 'Sn_CR Command Register, Close
  Else
    WN.SockTCPListen=1
  End If
End Function

Sub WN.SockClose(socket%)
  WN.Write(socket%+4, &H01, &H08) 'Sn_CR Command Register, DISCON
End Sub

Function WN.SockEstablished(socket%)
  Local SnCR%
  SnCR%=WN.Read(socket%+4, &H03)
  If SnCR%=&H1C Then
    WN.Write(socket%+4, &H01, &H08) 'Sn_CR Command Register, DISCON
  End If
  If SnCR%=&H17 Then
    WN.SockEstablished=1
  Else
    WN.SockEstablished=0
  End If
End Function

Function WN.SockClosed(socket%)
  Local SnCR%
  SnCR%=WN.Read(socket%+4, &H03)
  If SnCR%=&H00 Then
    WN.SockClosed=1
  Else
    WN.SockClosed=0
  End If
End Function

Function WN.SockData(socket%)
  WN.SockData=(WN.Read(socket%+4, &H26)*256)+(WN.Read(socket%+4, &H27))
End Function

Function WN.SockRead(socket%)
```

```
  Local SnRXRead%, offset%, address%
  SnRXRead%=(WN.Read(socket%+4, &H28)*256)+(WN.Read(socket%+4, &H29))
  offset%=SnRXRead% And &H07FF
  address%=&H6000+(socket%*&H800)+offset%
  WN.SockRead=WN.Read(address% >> 8, address% And &HFF) 'readbyte
  SnRXRead%=SnRXRead%+1
  If SnRXRead%>65535 Then SnRXRead%=0
  WN.Write(socket%+4, &H28, SnRXRead% >> 8) 'increment counter
  WN.Write(socket%+4, &H29, SnRXRead% And &HFF)
  WN.Write(socket%+4, &H01, &H40) 'Sn_CR Command Register, RECV
End Function

SUB WN.SockUDPDest (socket%, doct1%, doct2%, doct3%, doct4%, dstport%)
  WN.Write(socket%+4, &H0C, doct1%)
  WN.Write(socket%+4, &H0D, doct2%)
  WN.Write(socket%+4, &H0E, doct3%)
  WN.Write(socket%+4, &H0F, doct4%)
  WN.Write(socket%+4, &H10, dstport% >> 8) 'Sn_PORT Source Port Hi
  WN.Write(socket%+4, &H11, dstport% And &HFF) 'Sn_PORT Source Port Lo
END SUB

Sub WN.SockWrite(socket%, sockTX$)
  Local TXchars%, SnTXFSR%, SnTXWR%, offset%, address%, i%
  TXchars%=Len(sockTX$)
waitforprocessing:
  SnTXFSR%=(WN.Read(socket%+4, &H20)*256)+(WN.Read(socket%+4, &H21))
  If SnTXFSR%<TXchars% Then GoTo waitforprocessing
  SnTXWR%=(WN.Read(socket%+4, &H24)*256)+(WN.Read(socket%+4, &H25))
  offset%=SnTXWR% And &H07FF
  address%=&H4000+(socket%*&H800)+offset%
  For i%=1 To TXchars%
    If address%>&H4000+(socket%*&H800)+&H7FF Then address%=address%-&H800
    WN.Write(address%>>8, address% And &HFF, Asc(Mid$(sockTX$, i%, 1)))
    address%=address%+1
  Next i%
  SnTXWR%=SnTXWR%+TXchars%
  If SnTXWR%>65535 Then SnTXWR%=SnTXWR%-65536
  WN.Write(socket%+4, &H24, SnTXWR% >> 8)
  WN.Write(socket%+4, &H25, SnTXWR% And &HFF)
  WN.Write(socket%+4, &H01, &H20) 'Sn_CR Command Register, SEND
End Sub

Function WN.SockTCPConnect(socket%, oct1%, oct2%, oct3%, oct4%, destport%)
  Local timerstart%, SnCR%
  WN.Write(socket%+4, &H0C, oct1%) 'Destination IP Address
  WN.Write(socket%+4, &H0D, oct2%)
  WN.Write(socket%+4, &H0E, oct3%)
  WN.Write(socket%+4, &H0F, oct4%)
  WN.Write(socket%+4, &H10, (destport% >> 8) 'Sn_DPORT Hi
  WN.Write(socket%+4, &H11, destport% And &HFF) ' Lo
  WN.Write(socket%+4, &H01, &H04) 'Sn_CR Command Register, Connect
  WN.SockTCPConnect=0
  timerstart% = Timer
  Do While (Timer - timerstart%)<4000 ' Wait at least 4 seconds for connection
    SnCR%=WN.Read(socket%+4, &H03)
    If SnCR%=&H17 Then ' Established
      WN.SockTCPConnect=1
      Exit Do
    Else If SnCR%=&H00 Then ' Closed (Timeout connecting)
      Exit Do
    End If
  Loop
End Function
```

electronics/micromite_wiznet_library.txt · Last modified: 2015/06/16 05:30 by david