Overview

The MCB167-NET prototype board offers a 10Base-T (RJ45) Ethernet connection. This standard interface can be used for fast data transmissions in a local area network (LAN) to PC's or other microcontroller. In case the LAN is connected to the internet via a router, data can be exchanged worldwide. The software to handle the Ethernet interface is much more complex than for a serial interface. In most cases, the TCP/IP protocol is used for data transmission because it provides a fast and reliable connection to other members of the LAN.
This small TCP/IP stack and webserver was published in the extra issue 'Embedded Internet' of the magazine Design&Elektronik. For a complete German description of this software the extra issue can be ordered at www.elektroniknet.de/extraheft.
This TCP/IP stack is able to handle the protocols ARP, ICMP, IP and TCP. It is optimized for low resource consumption, not for performance. However, increasing the buffer sizes can improve the performance a lot. Because of the low resource consumption, the TCP/IP stack has quite some restrictions:
- Only one active TCP session at the same time
- No reassembling of fragmented incoming IP frames
- No buffering of TCP segments that are received in wrong order
- No checksum verification of received data
- no support for 'type of service (TOS)' and security options
- received TCP options are ignored

The webserver has some restrictions, too:
- Supports only one html page
- Since there is no virtual file system, no pictures (gif or jpeg) can be used
- No POSTing of Data supported. No data can be entered on the web page

This TCP/IP stack offers an easy-to-use application programming interface (API). There is no need to know the different protocols in detail. The API consists of a set of subroutines for sending and receiving data and a flag register that indicates the current status of stack. As an application example, a dynamic HTTP server is implemented.

CONTENTS

Application Programming Interface (API)

Global Variables and Macro's

The following global variables and macros can be found in the TCPIP.H and CS8900.H. Please check all these settings before compiling and running the software.

MYIP_1 … MYIP_4

Specifies the internet protocol (IP) address of the board. In most cases, an unused intranet IP should be specified. Intranet IP addresses have the address range of 192.168.x.x or 10.x.x.x.

SUBMASK_1 … SUBMASK_4

Specifies the subnet mask. With this mask the TCP/IP stack checks if the remote host is in the same LAN or if it must be addressed over the standard gateway. A typical subnet mask is 255.255.255.0 which specifies that a remote hosts is in the same LAN if the first three bytes of the IP address is identical to the own IP address.

MYMAC_1 ... MYMAC_6

These values specify the 48-bit ethernet MAC address which must be unique in the LAN. The broadcast address FF FF FF FF FF FF is not allowed as an individual address.

GWIP_1 … GWIP_4

Specifies the IP address of the standard gateway. This gateway is addressed if the remote IP is not part of the subnet.

RETRY_TIMEOUT

Specifies the number of timer overflows before a TCP frame is retransmitted when no ACK has been received. The default value is 8 which means 8 x 262ms (about 2 sec.)

FIN_TIMEOUT

Specifies the number of timer overflows before the TCP state machine is closed when no ACK or FIN has been received. The default value is 2 which means 2 x 262ms (about 0.5 s)

MAX_RETRYS

Specifies the number of unsuccessful retransmissions before TCP connection is reset. The default number is 4. The total number of transmissions is MAX_RETRYS + 1.

MAX_TCP_TX_DATA_SIZE

Specifies the size of the transmit buffer and the maximum size of sent datagrams. The default value is 512. This number must be even! Increasing this number leads to better network performance but allocates more memory in the target system.

MAX_TCP_RX_DATA_SIZE

Specifies the size of the receive buffer and the maximum size of datagrams that can be received. The default value is 256. This number must be even! Increasing this number leads to a much better network performance but allocates more memory in the target system.

MAX_ETH_TX_DATA_SIZE

Specifies the size of an additional buffer which is used to handle the ARP, ICMP and TCP protocol. The default value is 60 (must be even!) which is enough for a 32 byte ping (ICMP echo).

DEFAULT_TTL

Specifies the 'Time To Live' value which is used in transmitted TCP packets. The default value is 64 which means that the maximum lifetime of a datagram in internet is 64 seconds. When the time expires, the datagram is destroyed.

TCPLocalPort

Specifies the port which is listened to when 'TCPPassiveOpen()' is used (e.g. 80 for HTTP). When 'TCPActiveOpen()' is used, a random value can be specified which must be bigger than 1024.

RemoteIP

Specifies the internet protocol (IP) address of the remote host. It is only used if 'TCPActiveOpen()' is used.

TCPRemotePort

Specifies the port of the remote host. It must only be set if 'TCPActiveOpen()' is used.

TCPTxDataCount

Specifies the number of bytes to be transmitted. This variable must be set before the function 'TCPTransmitTxBuffer' is called.

TCPRxDataCount

This variable is set by the TCP stack. It specifies the number of bytes that have been received. These number of bytes can be read out of the receive buffer ('TCP_RX_BUF').

TCP_RX_BUF

Address of the receive buffer. When the status flag 'SOCK_DATA_AVAILABLE' is set, 'TCPRxDataCount' bytes can be read out of this buffer. After reading all bytes, the function 'TCPReleaseRxBuffer()' must be called to send an acknowledge to the remote host and to use the receive buffer for the next data frame.

TCP_TX_BUF

Address of the transmit buffer. When the status flag 'SOCK_TX_BUF_RELEASED' is set, the transmit buffer can be filled. The maximum number of bytes (MAX_TCP_TX_DATA_SIZE) must not be exceeded when the transmit buffer is filled. The actual number of bytes to be transmitted must be set in the variable 'TXPTxDataCount' before the function 'TCPTransmitTxBuffer' is called.

SocketStatus

This byte variable contains 8 status and error flags and can only be read. Details are listed below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit 1	Bit 0
SOCK_ ERROR	SOCK_ ERROR	SOCK_ ERROR	SOCK_ ERROR	SOCK_TX_BUF _RELEASED	SOCK_DATA AVAILABLE	SOCK_ CONNECTED	SOCK_ ACTIVE

SOCK_ACTIVE (bit 0)

This flag is set when the TCP/IP stack is busy with setting up a connection to a remote host. This happens when 'TCPPassiveOpen' or 'TCPActiveOpen' was called. As long as this flag is set, none of these two function may be called. In case a connection cannot be established or is closed, this flag is cleared automatically.

SOCK_CONNECTED (bit 1)

This flag indicates that a TCP connection is 'established'. As long as this flag is set, data can be sent and transmitted. This flag is automatically cleared when the connection is aborted or closed.

SOCK_DATA_AVAILABLE (bit 2)

This flag specifies that a new data frame has been received. If it is set, the application can read the receive buffer at the address of 'TCP_RX_BUF'. The length is stored in

'TCPRxDataCount'. After reading and processing the received data, the receive buffer must be released with the function 'TXPReleaseRxBuffer'. If the receive buffer is not released for too long, the TCP connection may be aborted.

SOCK_TX_BUF_RELEASED (bit 3)

This flag specifies whether the application may write into the transmit buffer and the variable 'TCPTxDataCount'. Only when a sent data frame has been acknowledged by the remote host, the next data frame may be written into the transmit buffer.

ERROR_CODE (bit 4-7)

The high nibble of SocketStatus indicates an error condition. All errors except SOCK_ERR_OK cause an immediate termination of the TCP connection. SOCK_ERROR_MASK can be used to mask the error bits. Here are the error conditions in detail:

Error Code	Reason
SOCK_ERR_OK	No error.
SOCK_ERR_ARP_TIMEOUT	The remote host did not respond to the ARP-request. The MAC address of the remote host was not determined. The remote host is either not connected to the network or is not able to answer.
SOCK_ERR_TCP_TIMEOUT	Even after several resends, the remote host did not acknowledge the sent data segment. This may happen if the connection is very unreliable or if the remote host is not able to anwer.
SOCK_ERR_CONN_RESET	A connection that is about to be established or an existing connection has been reset by the remote host. The remote host either does not accept a connection to the specified port or the remote application has reset the connection (STOP button of a internet browser).
SOCK_ERR_REMOTE	Because of a severe mistake in the remote TCP stack a corrupted TCP segment was received.
SOCK_ERR_ETHERNET	The TCP stack was not able was not able to send data over the ethernet. This happens when network cable is not connected.

Functions

The following functions are declared in the TCPIP.H header file.

void DoNetworkStuff(void)

This function polls the status of the ethernet controller, decodes the received frames, checks TCP timeouts, answers ARP requests and updates the SocketStatus variable. This TCP/IP stack polls the status of the ethernet controller and does not use interrupts for sending and receiving data. The function 'DoNetworkStuff' has to be called as often as possible get good network performance.

void TCPLowLevelInit(void)

This function initializes the ethernet contoller, the timer and the status flags. It must be called before any other TCP function is called.

void TCPPassiveOpen(void)

This function sets the Stack into 'LISTEN' mode and the status flag 'SOCK_ACTIVE' is set. In this mode, the stack waits for a connection. Before this function is called, the global variable 'TCPLocalPort' must be set to the port number the socket should listen to. When a connection is established, the status flag 'SOCK_CONNECTED' is set. This function does not block the program execution until a connection is established.

void TCPActiveOpen(void)

This function tries to establish a connection with a remote host. The status flag 'SOCK_ACTIVE' is set and a ARP-request is sent to determine the Ethernet MAC address of the remote host. Before this function is called, the variables 'RemoteIP', 'TCPRemotePort' and 'TCPLocalPort' must be set to the right values. In case the RemoteIP is not located within the local subnet, the standard gateway is addressed instead. This function does not block the program execution until a connection is established. After connecting to the remote system successfully, the status flag 'SOCK_CONNECTED' is set and the status of the TCP stack changes to 'ESTABLISHED'. If the remote host cannot be reached, an error status is set in the variable 'SocketStatus'.

void TCPClose(void);

This function closes the current TCP connection to the remote host. Pending data transmissions are finished before the connection is actually closed. After the connection is closed, the stack can be reconfigured to a different IP address and port before a new connection is opened.

void TCPReleaseRxBuffer(void)

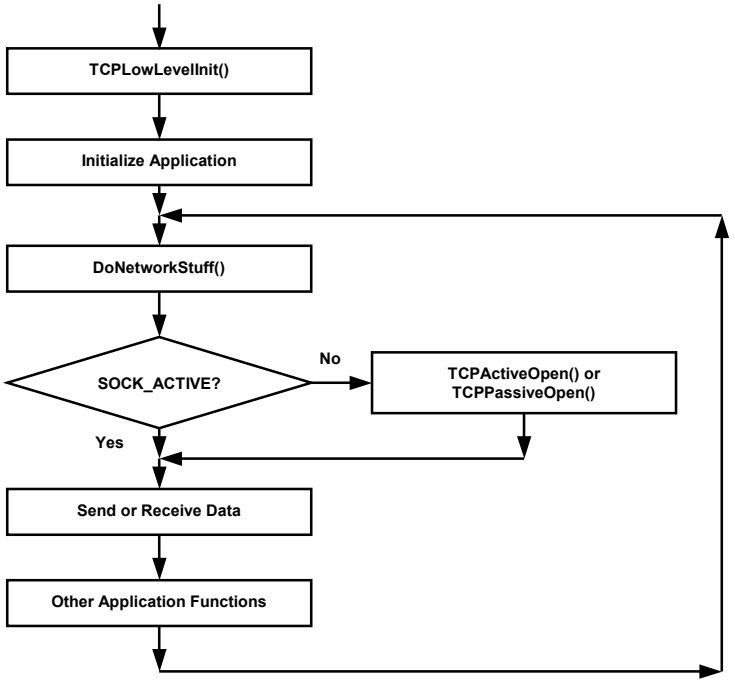
This function tells the stack that the data in the receive buffer are read and are not needed any more. The stack sends an acknowledge to the remote host and uses the receive buffer for the next data frame. The status flag 'SOCK_DATA_AVAILABLE' indicates whether a new data frame has been received or not.

void TCPTransmitTxBuffer(void)

This function transmits the content of the transmit buffer to the remote host. Before the transmit buffer can be filled, the status flag SOCK_TX_BUF_RELEASED must be checked. Only if this flag is set, data can be written into the transmit buffer. The pointer TXP_TX_BUF points to the beginning of the transmit buffer. The maximum number of bytes (MAX_TCP_TX_DATA_SIZE) must not be exceeded when the transmit buffer is filled. The actual number of bytes to be transmitted must be set in the variable TXPTxDataCount before the function 'TCPTransmitTxBuffer' is called.

Using the Application Program Interface (API)

This section describes how to use the API functions. The following figure shows in principle how to open a TCP connection and to transfer data.



Opening a Connection

A connection can be opened in two different modes, the passive or the active mode. This is done by calling the API function 'TCPPassiveOpen()' or 'TCPActiveOpen()'. When one of these functions is called, the status flag SOCK_ACTIVE is set. When the connection with a remote host is established, the flag SOCK_CONNECTED is set. This only happens after one or more calls to 'DoNetworkStuff()'.

Passive Mode

The function 'TCPPassiveOpen()' puts the stack into 'Listen' or 'Server' mode and waits for a connection from a remote host. Before this function is called, the 'TCPLocalPort' variable must be set to the port number the stack should listen to.

Active Mode

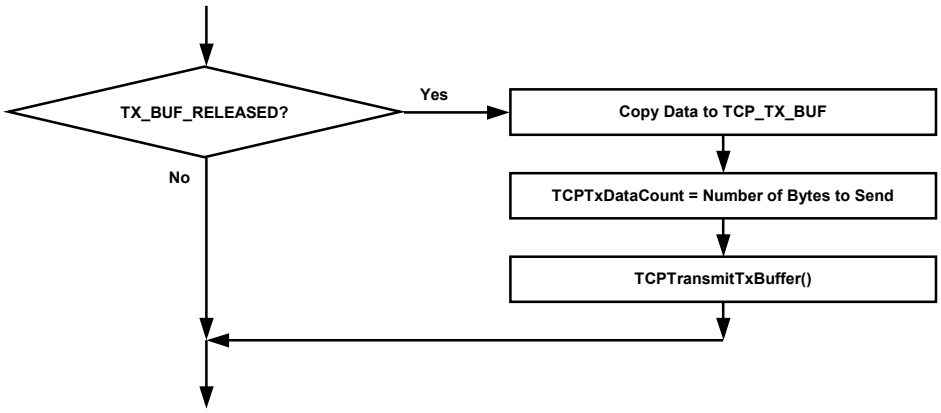
This function 'TCPActiveOpen()' tries to establish a connection with a remote host. An ARP-request is sent to determine the Ethernet MAC address of the remote host. Before this function is called, the variables 'RemoteIP', 'TCPRemotePort' and 'TCPLocalPort' must be set to the right values. In case the RemoteIP is not located within the local subnet, the standard gateway is addressed instead.

Data Transfer

Once a connection is established, the data transfer can be started. The following sections show how to send and receive data.

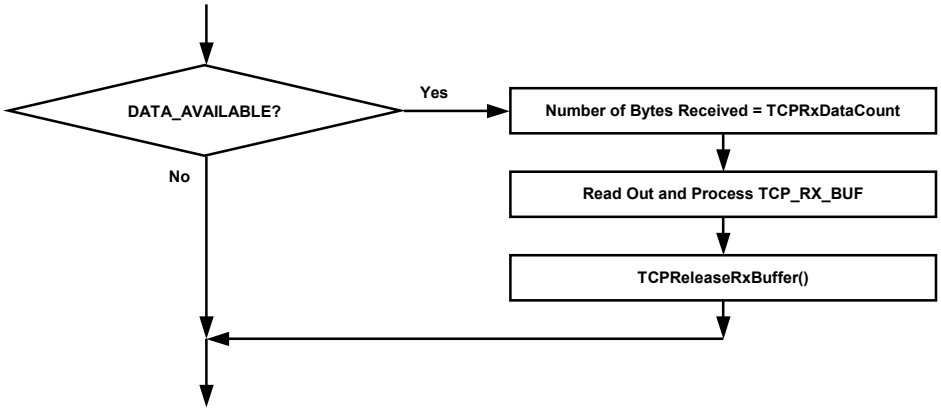
Sending Data

First, the flag SOCK_TX_BUF_RELEASED it must be checked if the transmit buffer is empty. If this is the case data can be copied into the transmit buffer. The API variable TCPTxDataCount must then be set to the number of bytes to be send. Finally, calling the function TCPTransmitTxBuffer() sends to data to the remote host and clears the flag SOCK_TX_BUF_RELEASED. This flag is set again when the remote host has acknowledged the datagram.



Receiving Data

The flag SOCK_DATA_AVAILABLE indicates whether a new datagram has been received. If this is the case, the API variable TCPRxDataCount specifies the number of bytes received. After copying the data out of the receive buffer the function TCPReleaseBuffer() must be called to free the receive buffer for new data.



Closing a Connection

A TCP connection can be closed because of different events. Usually this is done either locally by calling the API function 'TCPClose()' or remotely by the remote host. On the other hand, a connection is also closed because of an error condition like an overflow of the retransmission counter or a received RST (reset) flag. Error conditions are indicated by the 'SocketStatus' variable.

Web Server Example Application

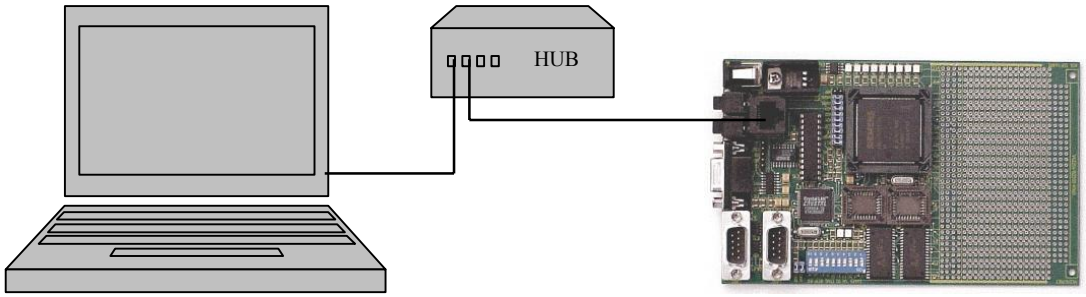
This very simple web server uses the TCP/IP stack described above. It can only handle one web page but it already can insert dynamic values. The web page that comes with this example shows the input voltage of analog input 0 and 1 as a bar graph on the screen (see below). The values of the A/D converter are inserted instead special strings (e.g. 'ADA%') in this case.

µVision2 Project Settings

The RAM is mapped into the address range 0x00000 – 0xFFFFF (1MB), the ethernet controller is mapped into the address range 0x100000 – 0x100FFF (4KB). The timer 2 is used as a system timer for the TCP stack.

Connecting the MCB167-NET to an existing LAN

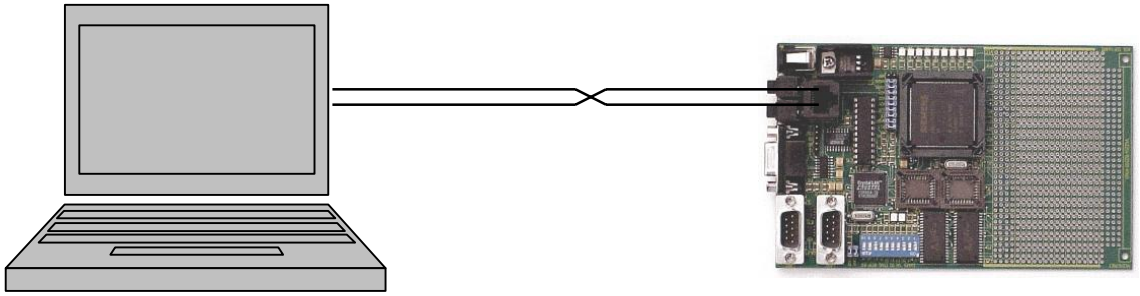
If the MCB167-NET should be connected to a LAN a standard shielded (STP) or unshielded (UTP) patch cable is used. The IP address, the MAC address and the gateway IP address needs to be adapted to the network. Please ask your network administrator if you are not sure about the addresses to use.



Connecting the MCB167-NET to a single PC

If one MCB167-NET should be connected to single PC, a crossover patch cable can be used instead of a hub or switch. The network interface card (NIC) of the PC must be configured to a

fixed IP address in this case. Usually the intranet IP range 192.168.x.x or 10.x.x.x with a subnet mask of 255.255.255.0 is used. The IP address of the MCB167-NET must be within the same subnet.



Starting the easyWEB application

Perform the following steps to run the easyWEB application on the MCB167-NET board.

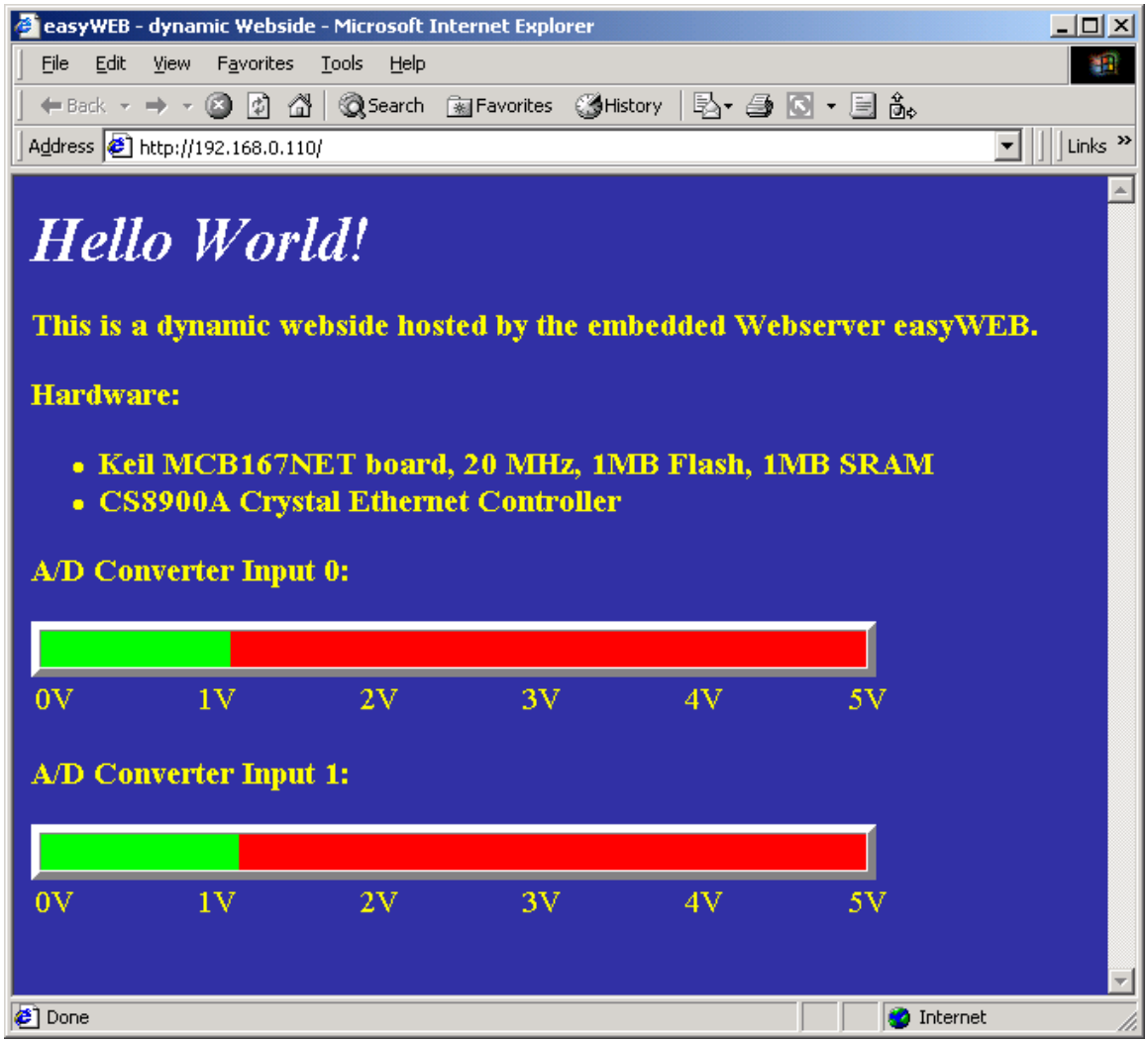
- Connect the serial interface, network and power supply to the board.
- Start µVision2 with the easyWEB.uv2 project file.
- Modify the IP address, gateway address and MAC address in the module TCPIP.H according to your network requirements.
- Compile and link the project with 'Project' -> 'Rebuild all target files'. Please note, that this application is already too big to be compiled with a free evaluation version.
- Start the µVision debugger with 'Debug' -> 'Start Debug Session'
- Start the application with 'Debug' -> 'Go'
- Check if the MCB167-NET responds to a ping. Open a command prompt window and type:
  ping <boards IP address>
  An output similar to the one below should be displayed:

```
C:\>ping 192.168.0.110
Pinging 192.168.0.110 with 32 bytes of data:
Reply from 192.168.0.110: bytes=32 time<10ms TTL=64
Reply from 192.168.0.110: bytes=32 time<10ms TTL=64
Reply from 192.168.0.110: bytes=32 time=10ms TTL=64
Reply from 192.168.0.110: bytes=32 time<10ms TTL=64

Ping statistics for 192.168.0.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  10ms, Average =  2ms
```

When only timeouts are reported, the application does not work and you have to check all network setting again.

- Open your web-browser and enter the board's IP address in the address line.
  Example: http://192.168.0.110
  Now, you should see a screen similar to the one below.



Conclusion

This application note shows how TCP/IP and ethernet can be used even in small microcontroller environments. It can be extended in many ways. The performance could be improved by using the memory interface instead of the IO interface of ethernet controller. Using the interrupt output of the ethernet controller instead of 'DoNetworkStuff' would yield better network response times. The web server could support more than one page and could support the 'post' method to enter data in the web page. Of course, all these features are covered by other commercial TCP/IP stacks.

Here are some useful links to get more information:

MCB167-NET manual: www.keil.com/mcb167net

Crystal ethernet controller CS8900: www.crystal.com

RFC's: www.faqs.org/rfcs/index.html

In the USA:
Keil Software, Inc.
1501 10th Street, Suite 110
Plano, TC 75074
USA

Sales: 800-348-8051
Phone: 972-312-1107
FAX: 972-312-1159

E-mail: sales.us@keil.com
support.us@keil.com

Internet: http://www.keil.com/

In Europe:
Keil Elektronik GmbH
Bretonischer Ring 15
D-85630 Grasbrunn b. Munchen
Germany

Phone: (49) (089) 45 60 40 - 0
FAX: (49) (089) 46 81 62

E-mail: sales.intl@keil.com
support.intl@keil.com