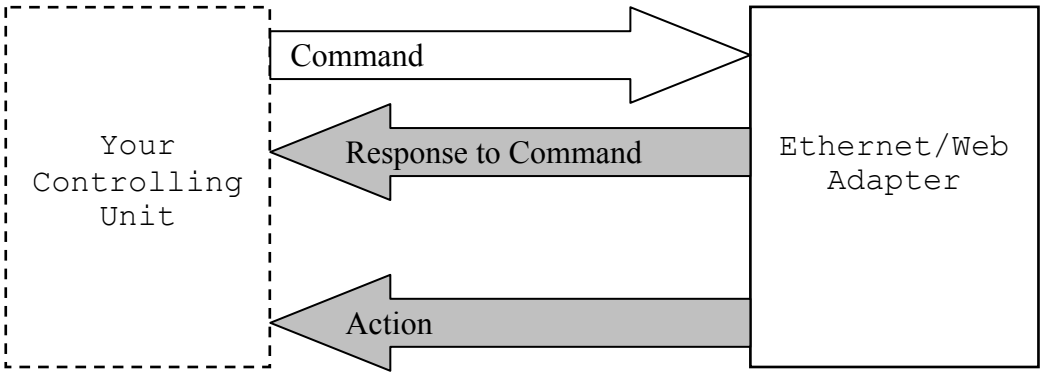
Ethernet/Web Adapter

Communication Protocol

Version 0.4

1. Communication with an Ethernet/Web Adapter 3
2. Byte Order 3
3. General Structure of Packet 3
4. Structure of Data Packet 3
4.1 Algorithm of the CRC Calculation implemented in C 4
5. Structure of Command, Response or Action 4
6. Description of particular commands 4
7. Socket Address Block (SA Block) 7
8. Examples 8
8.1 Data exchange for the “Set Local IP” command 8

1. Communication with an Ethernet/Web Adapter



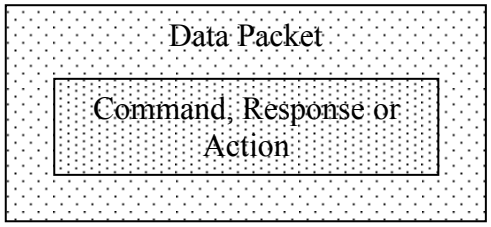
- A Command is always answered by Ethernet/Web Adapter.
- An Action should never be answered by your Controlling Unit. The decision to launch some subroutine on receiving an Action message or not is your decision.
- The detailed description of the serial channel or bus parameters may be found in the datasheet corresponding to the used module type. (The default settings for the EM01/EM02 Adapters using serial channel are: 38400, 8, N, 1; the baud rate can be reduced by means of an external switch.)

2. Byte Order

- All numbers (the fields with the specified size from 1 to 4 bytes) are transferred in High-Byte-First format.
- All data buffers (the fields with unspecified size and with the size that is bigger than 4 bytes) are transferred character by character from first to the last one.

3. General Structure of Packet

Command, Response to Command or Action packet is wrapped round by a Data Packet:



4. Structure of Data Packet

1 Byte	1 Byte	0...148 Byte	1 Byte	1 Byte
ID	Length	Data Bytes	CRC (Low Byte)	CRC (High Byte)

- ID – 80 (hex: 50),
- Length – size of Data Bytes only,
- Data Bytes – see “Structure of Command, Response or Action”,

- CRC – calculated using the following constants:
    1. Start multiplier – 133,
    2. Step – 6,
    3. CRC is additionally XOR-ed with the bit pattern: 5B AC (hex).

4.1 Algorithm of the CRC Calculation implemented in C

```
unsigned short CalcCrc(unsigned char* dataBuf, short dataSize)
{
 unsigned short index, crcStep;
 unsigned long crcMult, crcSum;
 crcMult = 133;  crcStep = 6  crcSum = 0;
 for( index = 0; index < dataSize; index++ )
   {
         crcSum += ((unsigned long)(dataBuf[index] * crcMult));
    if(crcMult <= (unsigned long)(0xffff - crcStep))
      crcMult += crcStep;
    else  crcMult = crcStep - (0x10000 - crcMult);
   }
 return( (unsigned short)crcSum ^ 0xac5b );
}
```

5. Structure of Command, Response or Action

1 Byte	1 Byte	1 Byte	0...128 Byte	1 Byte
Start Byte	Command ID	Terminator Byte	Parameters	End Byte

- Start Byte – '!' (hex: 21),
- Terminator Byte – ',' (hex: 2C),
- End Byte – ';' (hex: 3B),
- Command ID – also used in Response; see description of particular commands and actions below,
- Parameters – data fields separated by Terminator Byte; see description of particular commands and actions below.

6. Description of particular commands

- *Functionality* – the proper meaning of the command. The name in parenthesis is the corresponding Tiger Basic subroutine.
- *Subcommand ID* – one byte field eventually separated by Terminator (',') from further parameters (if the arguments are available). If the Subcommand ID is referred as "None" here, it must be merely left out together with the subsequent Terminator.
- *Arguments* – data fields separated by Terminator (',') from each other. The value in parenthesis is the argument length in bytes (the symbol '~' means that the length is not specified). The detailed description of the particular arguments can be found in the "Programming with Tiger Basic Sockets" chapter of the "Programming guide".

- *Response* – it has the same structure as Command, but it is sent by Ethernet/Web Adapter. The field Command ID of any Response contains the Command ID of the Command that is to be answered by this Response. The only parameter of the *Standard Response* is a success or an error code. The success code is always 0 (zero), the error codes are enumerated in the appropriate section of the "Programming Guide". If Command requests for some information or e.g. awaits an identifier as a result of the execution, the parameter list may be more complex and it is briefly described in the "Response" column of the table below. The value in parenthesis is the parameter length in bytes (the symbol '~' means that the length is not specified).

Functionality	Command ID	Subcommand ID	Arguments	Response
C O M M A N D S				
Set ISP *bSetupIsp*	1	2	1. Dial String (~) 2. User Name (~) 3. User Password (~)	Standard
Set Local IP *bSetupLocalIp*	1	4	1. IP Address (4) 2. Subnet Mask (4)	Standard
Enable DHCP *bSetupDhcp*	1	5	1. Flag (1) 2. Request Timeout (4)	Standard
Enable DNS *bSetupDns*	1	6	1. Flag (1) 2. DNS Server IP (4) [if DNS enabled]	Standard
Set PAP Secrets *bSetupPapSecret*	1	7	1. User Name (<=16) 2. User Passwd (<=16) 3. PAP Set Index (1)	Standard
Set Modem Baudrate *bSetModemBaudrate*	1	8	1. Baud Rate (4)	Standard
Set Default Gateway *bSetDefaultGateway*	1	9	1. Gateway IP (4)	Standard
Set Mac Address *bSetMacAddress*	1	10	1. MAC Address (6)	Standard
Set Tcp Window Size *bSetTcpWinSize*	1	11	1. Socket (4) 2. TCP Window Size (4)	Standard
Set Tcp Keep Alive *bSetTcpKeepAlive*	1	12	1. Socket (4) 2. TCP KA Ticks (4) 3. TCP KA Probes (1)	Standard
Get Local IP *lGetLocalIp*	2	1	1. Socket (4)	1. IP Address (4)
Get Local Port *wGetLocalPort*	2	2	1. Socket (4)	1. Port (2)
Get Remote IP *lGetRemoteIp*	2	3	1. Socket (4)	1. IP Address (4)
Get Remote Port	2	4	1. Socket (4)	1. Port (2)

wgetRemotePort				
Get CTS State *bGetCtsPinState*	2	5	None	1. Pin State (1)
Get Prog Version *bGetAdapterProgVers*	2	6	None	1. Version (4)
Get Mac Address *bGetMacAddress*	2	8	None	1. MAC Address (6)
Get Tcp Settings *lGetTcpSettings*	2	9	1. Socket (4)	1. TCP Window Size (4) 2. TCP KA Ticks (4) 3. TCP KA Probes (1)
Open Socket *lSocket*	3	None	1. Format (1) 2. Type (1)	1. Socket (4)
Bind *bBind*	4	None	1. Socket (4) 2. SA Block Size (2) 3. SA Block (16)	Standard
Connect *bConnect*	5	None	1. Socket (4) 2. Time Out (4) 3. SA Block Size (2) 4. SA Block (16)	Standard
Listen *bListen*	6	None	1. Socket (4) 2. Back Logs (4)	Standard
Close Socket *bCloseSocket*	7	None	1. Socket (4)	Standard
Send Data (TCP) *lSend*	8	None	1. Socket (4) 2. Flags (2) 3. Data Size (4) 4. Data (~)	Standard
Dial with Login *bDialIspWithLogin*	10	1	1. Time Out (4)	1. IP Address (4)
Dial only *bDialIsp*	10	2	1. Time Out (4)	1. IP Address (4)
Hang Up *bHangUp*	11	None	None	Standard
Send AT Command *bSendATCommand*	12	None	1. Time Out (4) 2. AT Command (~)	1. AT Reply (~)
Modem Send *bModemSend*	13	1		
Modem Receive *bModemReceive*	13	2	1. Time Out (4) 2. Buffer Size (4)	1. Data (~)
Modem Get Send Ready *wModemGetSendReady*	13	3	1. Time Out (4)	1. Buffer Size (2)
Modem Get Receive Ready *wModemGetRecvReady*	13	4	1. Time Out (4)	1. Buffer Size (2)
Modem Start Listen *bModemStartListen*	13	5	1. Min Data Size (4)	Standard

Modem Stop Listen *bModemStopListen*	13	6	None	Standard
Get IP by Name *lDnsGetIpByName*	14	None	1. Time Out (4) 2. Host Name (~)	1. IP Address (4)
Send Data (UDP) *lSendTo*	15	None	1. Socket (4) 2. SA Block Size (2) 3. SA Block (16) 4. Flags (2) 5. Data Size (4) 6. Data (~)	1. Number of Sent Bytes (4)
Get Time (SNTP) *lSntpGetTime*	16	None	1. IP Address (4) 2. Time Out (2)	1. Seconds (4) 2. Fraction (4)
A C T I O N S				
Accept Connection *OnAccept*	64	None	Socket (4) SA Block Size (2) SA Block (16)	Not used
Receive Data (TCP) *OnData*	65	None	Socket (4) Data Size (4) Data (~)	Not used
Remote Close *OnRemoteClose*	66	None	Socket (4)	Not used
PPP Run Down *OnPppDown*	68	None	None	Not used
DHCP Server Reply *OnDhcpUp*	69	None	IP Address (4) Subnet Mask (4) Gateway (4)	Not used
Receive Data From (UDP) *OnDataRecvFrom*	70	None	Socket (4) IP Address (4) Port (2) Flags (2) Data Size (4) Data (~)	Not used
Receive Modem Data *OnModemData*	71		Data Size (4) Data (~)	Not used

7. Socket Address Block (SA Block)

Various commands of the communication protocols use the string generally referenced as ‘Socket Address Block’. The data of different types and sizes are stored in the Socket Address Block string. The following offset and size values can be applied for storing and accessing the information in the SA Block:

Offset	Size	Description
0	1	Size of SA Block
1	1	Address Family

2	2	Port
4	4	IP Address

- The size of the SA Block is fix now (16), but may be modified in the future.
- The only *address family* currently supported is AF_INET (2).
- The *port* is a value of type 'word' (16-bit unsigned integer) standing for the connection port number.
- The *IP address* is of type 'long' (32-bit signed integer). Because the value of a numeric type can not directly represent an IP address in the more convenient 'dotted' notation, one should convert it into a 32-bit integer. For example the value 0C0A80102H (or 3232235778) is used to express the 192.168.1.2 IP address.
- The remained bytes must be filled with 0 (zeroes) up to size of the SA Block (16).
- The exact meaning of the *port* and *IP address* fields depends on the subroutine of the sockets family using the SA Block.

Please pay attention to the fact that the entry for the IP address enabling acceptance of any incoming connection by the **Bind** (*bBind*) command must be set to 0 (zeroes).

8. Examples

8.1 Data exchange for the "Set Local IP" command

The "SetLocalIp" command has two parameters: local IP address and local subnet mask.

The "SetLocalIp" command wrapped in the protocol envelope looks:
50 0F 21 01 2C 04 2C C0 A8 01 D0 2C FF FF FF 00 3B F6 C5

The particular bytes of the "SetLocalIp" command have the following meaning:
50 - ID of Data Packet
0F - length of Data Packet
  21 - start byte of Command
  01 - ID of Command
  2C - terminator byte
  04 - ID of SubCommand (Set Local IP)
  2C - terminator byte
  C0A801D0 - IP address (192.168.1.208)
  2C - terminator byte
  FFFFFF00 - subnet mask (255.255.255.0)
  3B - end byte of Command
F6 - CRC (low byte) of Data Packet
C5 - CRC (high byte) of Data Packet

The response to the above "SetLocalIp" command looks:
50 0C 21 01 2C 04 2C 00 2C 00 00 00 00 3B 86 3E

The particular bytes of the response to the "SetLocalIp" command have the following meaning:

50 - ID of Data Packet
0C - length of Data Packet
21 - start byte of Command
01 - ID of Command to response to
2C - terminator byte
04 - ID of SubCommand (Set Local IP)
2C - terminator byte
00 - error code (0 - no error)
2C - terminator byte
00000000 - return value (should be ignored in this case)
3B - end byte of Command
86 - CRC (low byte) of Data Packet
3E - CRC (high byte) of Data Packet