Ethernet library version V3.5
with TCP/IP Stack
MCU family PIC18F97J60
and ENC28J60
MikroBasic

Available symbols:

- **IpHeader** as byte[4]
- **MacHeader** as byte[6]

Available Functions and Procedures :

(AddrPacket is address of received packet, TXSTART is start address of transmitted packet)

- Sub Procedure Eth_SetParameters
' Set your IP, Mac, Mask, Gateway ... etc... here.
' Str2Ip('192.168.1.14',eth_ip_addr)
' Str2Ip('85.120.221.251',eth_ntp_addr)
' Str2Ip('192.168.1.1',eth_gateway)
' Str2Ip('255.255.255.0',eth_mask)
' Str2Mac('0004A300809A',eth_mac)

- Sub Procedure Eth_Init
' Init Ethernet part from MCU.

- Sub Procedure Eth_DoPacket
' Process incoming packets without TCP/IP Stack

- Sub Procedure Do_EthernetProc
' Process all Ethernet processes including TCP/IP Stack

- Sub Procedure Eth_UserProcess
' List of user processes, put you desired Ethernet processes here.
' this procedure is called by library, must be placed in **unit eth_lib_user**
' see examples for more detail

- Sub Procedure Eth_UserTCP(dim byref dest_ip_addr_T as IpHeader, dim byref source_port_T, dest_port_T, len_T as word)
' This Sub Function is called by library. Put your TCP response for TCP processes here.

- Sub Function Eth_UserUDP(dim byref dest_ip_addr_U as IpHeader, dim byref dest_port_U, source_port_U, len_U as word) as word
' This Sub Function is called by library. Put your UDP response here. See example.(ECHO example)

- Sub Procedure Eth_PutByte(dim value as byte)
‘ Put one byte in ETH memory.

- Sub Function Eth_GetByte as byte
‘ Get one byte from ETH memory.

- Sub Function CopyFlashToEthMem_CP(dim start_Flash_address as longint) as word
‘ Copy Const from flash to Eth Memory and return length of Const data.
‘ Const data must be defined as STRING. (and must be zero terminated)
‘ Ex. len = CopyFlashToEthMem_CP(@httpHeader)

- Sub Function CopyRamToEthMem_CP(dim byref s as string[4096]) as word
‘ Copy var data to Eth Memory and return length of var data.
‘ Var data must be defined as ARRAY. (and must be zero terminated)
‘ Ex. len = CopyRamToEthMem_CP(“data data data”)

- Sub Function Eth_SendARP(dim byref ip_dest as IpHeader) as Boolean
‘ Get ARP request. Return True if Mac exist.
‘ Return dest. mac address in var dest_mac : MacHeader.
‘ Return dest. ip address in var dest_ip_addr : IpHeader.
‘ Ex. bol = Eth_SendARP(user_ip_addr)
‘ First Sub Function search in ArpCache (size of 17) ...

- Sub Procedure SendUDP(dim dest_port_S, source_port_S, len_data as word,
Dim byref data_udp as byte[1472])
‘ Send UDP mesage. Max 1472 bytes.
‘ Ex. If Eth_SendARP(user_ip_addr) then SendUDP(dest_port, eth_port, 10, “Test Test “)

- Sub Procedure ErasePingCache
‘ Erase Ping cache table.

- Sub Procedure Eth_SendPing(dim SlotState as Boolean, dim PingSlot as Byte)
‘ Have 8 Slots available 0 .. 7 and Ping Cache size of 8.
‘ Ex. Eth_SendPing(Eth_SendARP(ip_addr),0) Send Ping at ip_addr and put result in Slot 0.
‘ PingCache format :
‘ PingCache[Slot].IpP as IpHeader
‘ PingCache[Slot].MacP as MacHeader
‘ PingCache[Slot].Time as word
‘ PingCache[Slot].TTL as byte.

- Sub Procedure CopyEthMemToRam(dim start_eth_address, dest_ram_address,
length_w as word)
‘ Ex. CopyEthMemToRam(AddrPacket+6,@dest_mac_addr,6)

- Sub Procedure CopyRamToEthMem(dim start_ram_address, dest_eth_address,
length_w as word)
‘ Ex. CopyRamToEthMem(@eth_mac,TXSTART+22,6)

- Sub Procedure CopyFlashToEthMem(dim start_Flash_address as longint,
dim dest_eth_address, length_w as word)
‘ Ex. CopyFlashToEthMem(@httpHeader,TXSTART+54,30)

- Sub Procedure CopyEthMemToEthMem(dim start_eth_address, dest_eth_address,
length_w as word, dim where as byte)
‘ where = 0 copy from Eth RxBuf to Eth TxBuf
‘ where = 1 copy from Eth TxBuf to Eth TxBuf
‘ Ex. CopyEthMemToEthMem(AddrPacket+38,TXSTART+28,4,0)

- Sub Procedure WriteToEthMem(dim dest_eth_address as word, dim value as byte)
‘ Ex. WriteToEthMem(TXSTART+12,$08)

- Sub Function ReadFromEthMem(dim start_eth_address as word) as byte
‘ Ex. data = ReadFromEthMem(AddrPacket+38)

- Sub Function EthMemCompareWithRam(dim start_eth_address, start_ram_address,
length_w as word) as boolean
‘ Ex. bol = EthMemCompareWithRam(AddrPacket+30,@eth_ip_addr,4)

- Sub Function EthMemCompareWithFlash(dim start_eth_address as word,
dim start_Flash_address as longint,
dim length_w as word) as boolean
‘ Ex. bol = EthMemCompareWithFlash(AddrPacket+54, @httpHeader, 30)

- Sub Function Eth_Cksum(dim start_eth_address, length_w as word) as word
‘ Ex. cksum_ip = Eth_Cksum(TXSTART+14,20)

- Sub Procedure Eth_WritePHYReg(dim register_address as byte, dim data as word)
‘ Write to PHY registers

- Sub Procedure Eth_SetLedConfig(dim NewConfig as word)
‘ Set Eth Led configuration. See datasheet for more detail.

- Sub Function Eth_ReadPacket as word
‘ Read packet and return TYPE OF SERVICE.

- Sub Procedure Eth_Send(dim length_w as word)
‘ Send packet from Tx buffer.

- Sub Function Send_Ping(dim byref ip_address as IpHeader) as word
‘ Send ping at specified ip_address and return response time.
‘ Ex. PingTimeResponse = Send_Ping(user_ip_address)

- Sub Function Send_UDP(dim byref ip_address as IpHeader, dim dest_port_S, source_port_S,
len_data as word, dim byref data_udp as byte[1472]) as boolean
‘ Send UDP mesage, max 1472 bytes, at specified ip_address and return true if success.
‘ Ex. Success = Send_UDP(user_ip_addr, 10001, 9999, 10, “data data “)

- Sub Function Send_ARP(dim byref ip_address as IpHeader,
dim byref mac_address as MacHeader) as boolean
‘ Get MAC for specified ip_address, put result in mac_address variable and return true if success.
‘ Ex. Success = Send_ARP(user_ip_addr,reply_mac_addr)

- Sub Procedure Firewall(dim ICMP, TCP, UDP as boolean)
‘ Default all false, allow all type of packets.
‘ If ICMP = true, ignore ICMP request. Ex. Firewall(true, false, false)
‘ If TCP = true, ignore TCP packets. Ex. Firewall(false, true, false)
‘ If UDP = true, ignore UDP packets. Ex. Firewall(false, false, true)

- Sub Procedure EraseARPCache
‘ Erase ARP cache table.

- Sub Procedure CounterTask ‘ Must be called in interrupt routine at 1ms
‘ Increments all counters used by library in ARP, ICMP, TCP, UDP,NTP routine.

- Sub Procedure CopyEthMemToRam_Inv(dim start_eth_address, dest_ram_address,
length_w as word)
‘ Ex. CopyEthMemToRam(AddrPacket+6,@data_dWord,4)

- Sub Procedure CopyRamToEthMem_Inv(dim start_ram_address, dest_eth_address,
length_w as word)
‘ Ex. CopyRamToEthMem(@data_dWord,TXSTART+22,4)

- Function Ntp_query as Boolean
‘ Syncronize time.
‘ Ntp address must me stored in eth_ntp_addr (see Eth_SetParameters)
‘ If syncronization was successfully Ntp_sync flag will be true.

- Procedure Get_Time
‘ Transform NTP time in day/month/year ...
‘ Data will be stored in :
‘ TTime.Rfc = Time in RFC format
‘ TTime.Unix = Time in UNIX format
‘ TTime.Year = Year (ex. 2008)
‘ TTime.Month = Month (ex. 07)
‘ TTime.Day = Day (ex. 24)
‘ TTime.Hour = Hour(ex. 14)
‘ TTime.Min = Min (ex. 25)
‘ TTime.Sec = Sec (ex. 00)
‘ TTime.Str = String format max. 32 (“2008-07-24 14:25:00”)

20-Aug-2008

Added **hw_cksum** flag, default state **false** (Ethernet checksum in software)
User can modify this flag on the fly, (can chose between **software checksum** and **hardware checksum**)

Regarding ERRATA, hardware checksum is not recommended!

10-Nov-2008

Added new **Firewall** rules.
Default **ALL Ports** are blocked, both **TCP** and **UDP**.
Have **10** sockets for opened **TCP** Ports and **10** for **UDP**.

Implemented:

- Sub Procedure UDP_Open_Port(port_u as word)
‘ Open UDP Port number, must be <> 0.

- Sub Procedure UDP_Close_Port(port_u as word)
‘ Close UDP Port number, must be <> 0.

- Sub Procedure TCP_Open_Port(port_u as word)
‘ Open TCP Port number, must be <> 0.

- Sub Procedure TCP_Close_Port(port_u as word)
‘ Close TCP Port number, must be <> 0.

- Sub Procedure Eth_Policy(poli as byte)
‘ **poli** can be **Drop** or **Reject** .
‘ if is **Drop**, library will ignore incoming packets addressed to closed ports.
‘ if is **Reject**, library will reply to Host with ICMP packet **Destination Host Unreachable**
‘ default value is **Drop.**

TCP/IP Stack

The **Structure** for one socket, **10** are available:

```
Structure TSocket
   Dim  State_S as byte          ' State of Socket, 0–disconnect, 1-waiting for connection
                                 ' 2-connected
   Dim source_port_S as word     ' Ethernet module Source Port
   Dim Dest_Ip as IpHeader       ' Destination IP Address
   Dim Dest_Mac as MacHeader     ' Destination MAC Address
   Dim dest_port_S as word       ' Destination Port Address
   Dim Seq_No_S as dword         ' Sequence Number
   Dim Ack_No_S as dword         ' Acknowledgment Number
   Dim Wait_ACK as byte          ' Wait_ACK flag
   Dim Exp_Time as byte          ' Expiration Time (sec)
   Dim Start_addr as dword       ' Pointer Start Address for transmitted data
   Dim Stop_addr as dword        ' Pointer Stop Address for transmitted data
   Dim RAM_ROM as byte           ' Pointer location, 0-RAM, 1-ROM
   Dim Keep_Alive as boolean     ' Keep Alive connections, max 60 sec without activity if true
End Structure
```

- Sub Function Open_TCP_Connection(dim byref dest_ip_addr as IpHeader, dim dest_port_T, source_port_T as word) as boolean

‘ Open one TCP connection , see exaples
‘ Return True if done successfully.

- Sub Procedure Close_TCP_Connection(dim byref dest_ip_addr as IpHeader, dim dest_port_T as word)

‘ Close TCP connection

- Sub Function Send_TCP(dim byref dest_ip_addr as IpHeader, dim dest_port_T, len_data as word, dim byref data_tcp as byte[1400])as boolean

‘ Send data over TCP, from RAM, max 1400 bytes, return true if successfully

Author: Florin Andrei Medrea, Software Version V3.5
Copyrights (c) 2008 - YO2LIO, All Rights Reserved