Micrium

Empowering Embedded Systems

µC/TCP-IP

V1.91

Release Notes

www.Micrium.com

Revision History

Version	Date	Description
V1.91	2007 Oct	New features, bug fixes, & improvements
V1.90	2007 May	New features & improvements
V1.89	2007 Mar	New features & improvements
V1.88	2006 Nov	New features, bug fixes, & improvements
V1.87	2006 Sep	Bug fixes & improvements
V1.86	2006 Aug	Bug fixes & improvements
V1.85	2006 Jun	Bug fixes & improvements
V1.84	2006 Apr	New features, bug fixes, & improvements
V1.83	2005 Dec	Bug fixes & improvements
V1.82	2005 Dec	Bug fixes & improvements
V1.81	2005 Oct	New features & improvements
V1.80	2005 Oct	Network communication performance improvements Bug fixes & improvements
V1.73	2005 Aug	TCP transmit round-trip time & retransmission controls Bug fixes & improvements
V1.72	2005 Jul	Bug fixes & improvements
V1.71	2005 May	Bug fixes & improvements
V1.70	2005 Apr	TCP transmit congestion and window controls Bug fixes & improvements
V1.61	2005 Feb	Bug fixes
V1.60	2005 Feb	TCP receive congestion and window controls First version with release history
V1.56	2004 Dec	Bug fixes & improvements
V1.54	2004 Dec	Bug fixes & improvements
V1.52	2004 Nov	Bug fixes & improvements
V1.50	2004 Oct	First TCP/IP version released
V1.00	2004 Jun	First Beta version released
V0.60	2004 Feb	First UDP/IP version released
V0.50		

Requires the following versions of needed Modules

Version 1.91

µC/CPU Version 1.18
µC/LIB Version 1.24

Version 1.90

µC/CPU Version 1.17
µC/LIB Version 1.24

Version 1.89

µC/CPU Version 1.16
µC/LIB Version 1.23

Version 1.88

µC/CPU Version 1.15
µC/LIB Version 1.22

Version 1.87

µC/CPU Version 1.15
µC/LIB Version 1.22

Version 1.86

µC/CPU Version 1.14
µC/LIB Version 1.21

Version 1.85

µC/CPU Version 1.14
µC/LIB Version 1.20

Version 1.84

µC/CPU Version 1.13
µC/LIB Version 1.18

Version 1.83

µC/CPU Version 1.13
µC/LIB Version 1.18

Version 1.82

µC/CPU	Version 1.13
µC/LIB	Version 1.18

Version 1.81

µC/CPU	Version 1.13
µC/LIB	Version 1.18

Version 1.80

µC/CPU	Version 1.13
µC/LIB	Version 1.18

Version 1.73

µC/CPU	Version 1.12
µC/LIB	Version 1.17

Version 1.72

µC/CPU	Version 1.12
µC/LIB	Version 1.17

Version 1.71

µC/CPU	Version 1.11
µC/LIB	Version 1.15

Version 1.70

µC/CPU	Version 1.11
µC/LIB	Version 1.14

Version 1.61

µC/CPU	Version 1.10
µC/LIB	Version 1.13

Version 1.60

µC/CPU	Version 1.10
µC/LIB	Version 1.13

New Features

Version 1.91

V1.91-001
Added BSD `select()` network socket functionality :

`select()`	
`NetSock_Sel()`	indicates which sockets are ready with available resources or operations.

Version 1.90

V1.90-001
Added new ARP status functions :

`NetARP_TxReqGratuitous()`	transmits an ARP request for the host's (IP) protocol address onto the local network.
`NetARP_IsAddrProtocolConflict()`	indicates whether a (IP) protocol address conflict exists with another host on the local network.

See also 'New Features V1.88-001'.

V1.90-002
Improved IP address handling to include link-local addresses.

See also 'Improvements V1.90-001'.

V1.90-003a
Added new IP status functions :

`NetIP_GetAddrSubnetMask()`	returns the currently configured IP address subnet mask for this host.
`NetIP_GetAddrStrSubnetMask()`	returns the currently configured IP address subnet mask, as an ASCII string.

See also 'New Features V1.84-001'.

V1.90-003b
Added new IP status functions :

`NetIP_IsAddrClassA()`	indicates whether an address is a Class-A IP address.
`NetIP_IsAddrClassB()`	indicates whether an address is a Class-B IP address.
`NetIP_IsAddrClassC()`	indicates whether an address is a Class-C IP address.
`NetIP_IsAddrThisHost()`	indicates whether an address is a 'This Host' initialization IP address.
`NetIP_IsAddrLocalHost()`	indicates whether an address is a 'localhost' IP address.
`NetIP_IsAddrLocalLink()`	indicates whether an address is a link-local IP address.
`NetIP_IsAddrBroadcast()`	indicates whether an address is a limited broadcast IP address.

Version 1.89

V1.89-001

Added new socket timeout configuration functions :

```
void  NetSock_CfgTimeoutRxQ_Dflt(NET_SOCK_ID  sock_id,
                                 NET_ERR     *perr);

void  NetSock_CfgTimeoutTxQ_Dflt(NET_SOCK_ID  sock_id,
                                 NET_ERR     *perr);

void  NetSock_CfgTimeoutConnReqDflt(NET_SOCK_ID  sock_id,
                                    NET_ERR     *perr);

void  NetSock_CfgTimeoutConnAcceptDflt(NET_SOCK_ID  sock_id,
                                       NET_ERR     *perr);

void  NetSock_CfgTimeoutConnCloseDflt(NET_SOCK_ID  sock_id,
                                      NET_ERR     *perr);
```

See also 'Changes V1.89-001'.

Version 1.88

V1.88-001

Added new ARP status functions :

`NetARP_ProbeAddrOnNet()`	probes the local network for a specific (IP) protocol address.
`NetARP_CacheAddrGetHW()`	returns the (MAC) hardware address for a specific (IP) protocol address, if cached.

See also 'New Features V1.90-001'.

Version 1.87

N/A

Version 1.86

N/A

Version 1.85

N/A

Version 1.84

V1.84-001

Added new IP status functions :

`NetIP_GetAddrThisHost()`	returns the currently configured IP address for this host.
`NetIP_GetAddrDfltGateway()`	returns the currently configured IP address of this host's default gateway.
`NetIP_GetAddrStrThisHost()`	returns the currently configured IP address for this host, as an ASCII string.
`NetIP_GetAddrStrDfltGateway()`	returns the currently configured IP address of this host's default gateway, as an ASCII string.

See also 'New Features V1.90-003a'.

Version 1.83

N/A

Version 1.82

N/A

Version 1.81

V1.81-001

Added new TCP configuration function :

`NetTCP_ConnCfgTxAckImmedRxdPushEn()`	configures TCP connection to enable/disable transmitting immediate acknowledgements for any received & pushed TCP segments.

Version 1.80

V1.80-001

Added new Network Debug Status functions :

`NetDbg_ChkStatusTCP()`	returns TCP layer status.
`NetDbg_ChkStatusBufs()`	returns network buffer(s) status.

Version 1.73

V1.73-001

TCP Transmit Congestion Controls
- Round-Trip Time (RTT)
- Retransmission Timeout (RTO)

Version 1.72

N/A

Version 1.71

V1.71-001

Added new Network Debug Status functions :

`NetDbg_ChkStatus()`	returns network status errors & faults.
`NetDbg_ChkStatusRsrcLost()`	returns network resource(s) lost status.
`NetDbg_ChkStatusRsrcLo()`	returns network resource(s) low status.
`NetDbg_ChkStatusConns()`	returns network connection(s) status.

Version 1.70

V1.70-001

TCP Transmit Congestion Controls
- Slow Start / Congestion Avoidance
- Fast Re-transmit / Fast Recovery
- Nagle Algorithm
- Transmit Silly Window Syndrome Avoidance
- Transmit Window Configuration / Control
- Idle Connection / Zero-Window Probe Persist Timers

Version 1.61

N/A

Version 1.60

V1.60-001

TCP Receive Congestion Controls

- Immediate & Delayed Acknowledgements
- Receive Silly Window Syndrome Avoidance
- Receive Window Configuration / Control

V1.60-002

TCP Transmit Segment Aggregation

- Data
- FIN-Close
- PUSH

Improvements

Version 1.91

V1.91-001
Modified `NetASCII_Str_to_MAC()` to allow a MAC address string's hexadecimal values to be separated by either hyphen characters or colon characters.

See also 'Changes V1.91-001'.

Version 1.90

V1.90-001
Improved IP address handling to include link-local addresses.

See also 'New Features V1.90-001'.

Version 1.89

V1.89-001
Added functionality to abort application or network tasks waiting on any network resources that have been closed, disconnected, or aborted.

V1.89-002
Refactored `NetOS_TimeoutCalc_OS_tick()` to improve timeout & OS tick overflow detection.

Version 1.88

V1.88-001
Refactored `NetTCP_TxConnClose()` to remove logical dead code.

Version 1.87

V1.87-001
Refactored TCP to ignore incoming connection requests when the listen socket's connection accept queue exceeds the maximum configured number of connections.

V1.87-002
Refactored `NetSock_RxDataHandlerDatagram()` to call `NetUDP_RxAppData()` to deframe received application data.

Version 1.86

V1.86-001

Refactored network packet reads/writes to & from network buffers to be internally independent of any CPU word-alignment requirements. This offers NIC drivers the capability to transfer/copy network packets directly to/from network buffers, especially for NICs/CPUs that require NIC-to-CPU word-aligned transfers (e.g. Direct Memory Access (DMA) transfers from NIC packets to/from CPU memory).

V1.86-002

Refactored network timer expiration functions to be defined as '`CPU_FNCT_PTR`' function type [i.e. '`(void) (void *)`'] & to explicitly cast '`void`' pointer arguments to specific network object pointers. This complies with ISO IEC 9899-1999 ANSI-C, Sections 6.3.2.3.1 & 6.3.2.3.7 & avoids potentially incorrect pointer conversion behavior between '`void`' pointer parameters & expiration functions' object pointers.

Version 1.85

V1.85-001

Improved ARM assembly port files to be compatible for both ARM & Thumb modes.

Version 1.84

V1.84-001

Improved TCP/Socket/Connection Close Management to reduce/eliminate connection leaks, corruption, faults.

V1.84-002

`NetTCP_ConnClose()` modified to transmit immediate TCP reset packet on any abnormal or fault TCP connection close.

Version 1.83

V1.83-001

Added `NET_BSD_CFG_API_EN` to indicate whether BSD 4.x Layer API functionality should be enabled & included in the project build :

`DEF_DISABLED`	BSD 4.x API functions disabled & excluded from build
`DEF_ENABLED`	BSD 4.x API functions enabled & included in build

Version 1.82

V1.82-001

Refactored network buffer free functionality into appropriate `NetBuf_FreeBuf()` functions :

`NetBuf_FreeBuf()`	free a network buffer.
`NetBuf_FreeBufList()`	free a network buffer list.
`NetBuf_FreeBufQ_PrimList()`	free a network buffer queue, organized by primary list.
`NetBuf_FreeBufQ_SecList()`	free a network buffer queue, organized by secondary list.

V1.82-002

Improved `NetDbg_ChkStatus()` functions by returning more specific debug status codes.

Version 1.81

V1.81-001

Improved `NetTCP_ConnCfg()` functions' validation & configuration.

Version 1.80

V1.80-001

Network Transmit/Receive Load Balancing
- Handle network transmit & receive packets in an approximately balanced ratio.

V1.80-002

Improved `NetIP_TxPktDatagramNextHopSel()`'s IP address host, network, & default gateway validation.

Version 1.73

V1.73-001

Added `NET_NIC_CFG_TX_PKT_PREPARE_EN` to allow the NIC to prepare transmit packet(s) while simultaneously transmitting previously-prepared packets.

If configured as `DEF_ENABLED`, the NIC driver **MUST** implement an appropriate `NetNIC_TxPktPrepare()` function to prepare transmit packet(s) separate from transmitting them in `NetNIC_TxPkt()`.

Version 1.72

V1.72-001

Added `NET_NIC_CFG_RD_WR_SEL` to indicate how the NIC's read/write functionality should be implemented :

`NET_NIC_RD_WR_SEL_MACRO`	implement with macro's to improve read/ write performance
`NET_NIC_RD_WR_SEL_FNCT`	implement with functions to handle more complex read/write functionality

Version 1.71

V1.71-001

Added fatal network socket error code, `NET_SOCK_ERR_FAULT`. When this or the following error codes are returned by a socket function, that socket must be immediately `close()`'d with no further access :

```
NET_SOCK_ERR_NOT_USED
NET_SOCK_ERR_INVALID_FAMILY
NET_SOCK_ERR_INVALID_PROTOCOL
NET_SOCK_ERR_INVALID_TYPE
NET_SOCK_ERR_INVALID_STATE
NET_SOCK_ERR_FAULT
```

Version 1.70

N/A

Version 1.61

N/A

Version 1.60

V1.60-001

Modified String Conversion Functions Argument Lists :

> ASCII MAC address to MAC address: `NetASCII_Str_to_MAC()`
> MAC address to ASCII MAC address: `NetASCII_MAC_to_Str()`
>
> NOTE: New arguments were added to these functions.

V1.60-002

Improved IP address String Conversion Functions :

> `NetASCII_Str_to_IP()` now allows leading zeros in IP address string.
> `NetASCII_IP_to_Str()` now allows ASCII IP address to have leading zeros.

Changes

Version 1.91

V1.91-001

Added `hex_colon_sep` argument to configure whether a MAC address string's hexadecimal values is separated by either hyphen characters or colon characters.

```
void  NetASCII_MAC_to_Str(CPU_INT08U   *paddr_mac,
                          CPU_CHAR     *paddr_mac_ascii,
                          CPU_BOOLEAN   hex_lower_case,
                          CPU_BOOLEAN   hex_colon_sep,
                          NET_ERR      *perr);
```

See also 'Improvements V1.91-001'.

V1.91-002

Renamed network socket address structures' – `NET_SOCK_ADDR` & `NET_SOCK_ADDR_IP` – address family member from `Family` to `AddrFamily`. However, address family constant, `NET_SOCK_ADDR_FAMILY_IP_V4` (equivalent to BSD address family constant, `AF_INET`), remains unchanged.

V1.91-003

Modified network socket open parameters for consistency with other socket data types :

```
NET_SOCK_ID  NetSock_Open(NET_SOCK_PROTOCOL_FAMILY   protocol_family,
                          NET_SOCK_TYPE              sock_type,
                          NET_SOCK_PROTOCOL          protocol,
                          NET_ERR                   *perr);
```

V1.91-004a

Modified BSD socket receive & transmit functions' application data length parameters – `data_buf_len` & `data_len` – from a signed integer to an unsigned integer as well as ensuring that all other parameter data types comply with IEEE Std 1003.1, 2004 Edition, Section '`sys/socket.h` : DESCRIPTION' :

```
ssize_t  recvfrom(        int         sock_id,
                          void       *pdata_buf,
                         _size_t      data_buf_len,
                          int         flags,
                  struct  sockaddr   *paddr_remote,
                          socklen_t  *paddr_len);

ssize_t  recv    (        int         sock_id,
                          void       *pdata_buf,
                         _size_t      data_buf_len,
                          int         flags);
```

```
ssize_t  sendto  (        int          sock_id,
                          void        *pdata,
                         _size_t       data_len,
                          int          flags,
                   struct sockaddr    *paddr_remote,
                          socklen_t    addr_len);

ssize_t  send    (        int          sock_id,
                          void        *pdata,
                         _size_t       data_len,
                          int          flags);
```

V1.91-004b
Changed network socket receive & transmit functions' application data length parameters – `data_buf_len` & `data_len` – from a signed integer to an unsigned integer to comply with IEEE Std 1003.1, 2004 Edition, Section '`sys/socket.h` : DESCRIPTION' :

```
NET_SOCK_RTN_CODE  NetSock_RxDataFrom(NET_SOCK_ID         sock_id,
                                      void               *pdata_buf,
                                      CPU_INT16U          data_buf_len,
                                      CPU_INT16S          flags,
                                      NET_SOCK_ADDR      *paddr_remote,
                                      NET_SOCK_ADDR_LEN  *paddr_len,
                                      void               *pip_opts_buf,
                                      CPU_INT08U          ip_opts_buf_len,
                                      CPU_INT08U         *pip_opts_len,
                                      NET_ERR            *perr);

NET_SOCK_RTN_CODE  NetSock_RxData    (NET_SOCK_ID         sock_id,
                                      void               *pdata_buf,
                                      CPU_INT16U          data_buf_len,
                                      CPU_INT16S          flags,
                                      NET_ERR            *perr);


NET_SOCK_RTN_CODE  NetSock_TxDataTo  (NET_SOCK_ID         sock_id,
                                      void               *p_data,
                                      CPU_INT16U          data_len,
                                      CPU_INT16S          flags,
                                      NET_SOCK_ADDR      *paddr_remote,
                                      NET_SOCK_ADDR_LEN   addr_len,
                                      NET_ERR            *perr);

NET_SOCK_RTN_CODE  NetSock_TxData    (NET_SOCK_ID         sock_id,
                                      void               *p_data,
                                      CPU_INT16U          data_len,
                                      CPU_INT16S          flags,
                                      NET_ERR            *perr);
```

V1.91-005
Changed the following `net_cfg.h` TCP & socket default timeout values to the infinite (i.e. no-timeout) timeout value, `NET_TMR_TIME_INFINITE` :

```
NET_TCP_CFG_TIMEOUT_CONN_RX_Q_MS
NET_TCP_CFG_TIMEOUT_CONN_TX_Q_MS
NET_SOCK_CFG_TIMEOUT_RX_Q_MS
NET_SOCK_CFG_TIMEOUT_CONN_REQ_MS
NET_SOCK_CFG_TIMEOUT_CONN_ACCEPT_MS
```

See also 'Changes V1.89-002'.

Version 1.90
N/A

Version 1.89

V1.89-001a

Changed `net_os.c` TCP & socket timeout configuration functions to `net_sock.c` socket configuration functions :

```
void  NetSock_CfgTimeoutRxQ_Set(NET_SOCK_ID  sock_id,
                                CPU_INT32U   timeout_ms,
                                NET_ERR     *perr);

void  NetSock_CfgTimeoutTxQ_Set(NET_SOCK_ID  sock_id,
                                CPU_INT32U   timeout_ms,
                                NET_ERR     *perr);

void  NetSock_CfgTimeoutConnReqSet(NET_SOCK_ID  sock_id,
                                   CPU_INT32U   timeout_ms,
                                   NET_ERR     *perr);

void  NetSock_CfgTimeoutConnAcceptSet(NET_SOCK_ID  sock_id,
                                      CPU_INT32U   timeout_ms,
                                      NET_ERR     *perr);

void  NetSock_CfgTimeoutConnCloseSet(NET_SOCK_ID  sock_id,
                                     CPU_INT32U   timeout_ms,
                                     NET_ERR     *perr);


CPU_INT32U  NetSock_CfgTimeoutRxQ_Get_ms(NET_SOCK_ID  sock_id,
                                         NET_ERR     *perr);

CPU_INT32U  NetSock_CfgTimeoutTxQ_Get_ms(NET_SOCK_ID  sock_id,
                                         NET_ERR     *perr);

CPU_INT32U  NetSock_CfgTimeoutConnReqGet_ms(NET_SOCK_ID  sock_id,
                                            NET_ERR     *perr);

CPU_INT32U  NetSock_CfgTimeoutConnAcceptGet_ms(NET_SOCK_ID sock_id,
                                               NET_ERR     *perr);

CPU_INT32U  NetSock_CfgTimeoutConnCloseGet_ms(NET_SOCK_ID  sock_id,
                                              NET_ERR     *perr);
```

See also 'New Features V1.89-001'.

Applications should use these new socket configuration API functions in place of the internal network RTOS-port configuration functions. The following lists the appropriate functions to replace the previously-used RTOS-port configuration function :

```
RTOS FUNCTION                          REPLACE WITH SOCKET API FUNCTION
-------------                          --------------------------------

NetOS_TCP_RxQ_TimeoutSet()             NetSock_CfgTimeoutRxQ_Set()
NetOS_TCP_TxQ_TimeoutSet()             NetSock_CfgTimeoutTxQ_Set()

NetOS_Sock_RxQ_TimeoutSet()            NetSock_CfgTimeoutRxQ_Set()
NetOS_Sock_ConnReqTimeoutSet()         NetSock_CfgTimeoutConnReqSet()
NetOS_Sock_ConnAcceptQ_TimeoutSet()    NetSock_CfgTimeoutConnAcceptSet()
NetOS_Sock_ConnCloseTimeoutSet()       NetSock_CfgTimeoutConnCloseSet()


NetOS_TCP_RxQ_TimeoutGet_ms()          NetSock_CfgTimeoutRxQ_Get_ms()
NetOS_TCP_TxQ_TimeoutGet_ms()          NetSock_CfgTimeoutTxQ_Get_ms()

NetOS_Sock_RxQ_TimeoutGet_ms()         NetSock_CfgTimeoutRxQ_Get_ms()
NetOS_Sock_ConnReqTimeoutGet_ms()      NetSock_CfgTimeoutConnReqGet_ms()
NetOS_Sock_ConnAcceptQ_TimeoutGet_ms() NetSock_CfgTimeoutConnAcceptGet_ms()
NetOS_Sock_ConnCloseTimeoutGet_ms()    NetSock_CfgTimeoutConnCloseGet_ms()
```

Note that since the API (i.e. parameter lists & return values) is the same for both sets of configuration functions, applications only need to replace the name of the RTOS configuration function with the name of the socket configuration API function.

V1.89-001b
Added socket timeout configuration constants :

```
NET_SOCK_TIMEOUT_MIN_MS
NET_SOCK_TIMEOUT_MAX_MS
NET_SOCK_TIMEOUT_MIN_SEC
NET_SOCK_TIMEOUT_MAX_SEC
```

V1.89-002
Changed `net_cfg.h` TCP & socket timeout configurations from previously configuring timeouts in units of seconds to configuring in units of milliseconds.

Examples :

```
NET_TCP_CFG_TIMEOUT_CONN_RX_Q_MS
NET_SOCK_CFG_TIMEOUT_RX_Q_MS
NET_SOCK_CFG_TIMEOUT_CONN_CLOSE_MS
```

See also 'Changes V1.80-001 & V1.91-005'.

Version 1.88

N/A

Version 1.87

V1.87-001

Changed UDP application receive API to include flags :

```
CPU_INT16U  NetUDP_RxAppData(NET_BUF     *pbuf,
                             void        *pdata_buf,
                             CPU_INT16U   data_buf_len,
                             CPU_INT16U   flags,
                             void        *pip_opts_buf,
                             CPU_INT08U   ip_opts_buf_len,
                             CPU_INT08U  *pip_opts_len,
                             NET_ERR     *perr);
```

`NET_UDP_FLAG_RX_DATA_PEEK`	Receive UDP application data without consuming the data; i.e. do NOT free any UDP receive packet buffer(s).

Version 1.86

N/A

Version 1.85

V1.85-001

Added `NET_UDP_MTU_ACTUAL` & `NET_TCP_MTU_ACTUAL` to adjust the transport layer Maximum Transmission Units (MTUs) based on the configured values for network buffer sizes.

Version 1.84

V1.84-001

Changed network debug information & status names to remove redundant '`DBG`' token. E.g. '`NET_DBG_CFG_DBG_INFO_EN`' to '`NET_DBG_CFG_INFO_EN`' & '`NET_DBG_CFG_DBG_STATUS_EN`' to '`NET_DBG_CFG_STATUS_EN`'.

Version 1.83

N/A

Version 1.82

N/A

Version 1.81

V1.81-001

`NetTCP_TxConnAck()` modified to transmit immediate acknowledgments for any received & pushed TCP segments based on the configuration of a TCP connection enable (see also 'New Features V1.81-001').

Version 1.80

V1.80-001-a

Changed all `net_cfg.h` timeout configurations from previously configuring timeouts in units of network ticks to configuring timeouts in units of seconds or milliseconds.

Examples :

```
NET_TCP_CFG_TIMEOUT_CONN_MAX_SEG_SEC
NET_TCP_CFG_TIMEOUT_CONN_RX_Q_SEC
NET_SOCK_CFG_TIMEOUT_CONN_CLOSE_SEC
```

See also 'Changes V1.89-002'.

V1.80-001-b

Changed `net_os.c` timeout configuration functions from configuring timeouts in units of network ticks to configuring timeouts in units of milliseconds.

Examples :

```
void  NetOS_TxSuspendTimeoutSet(CPU_INT32U   timeout_ms,
                                NET_ERR     *perr);

void  NetOS_Sock_RxQ_TimeoutSet(NET_SOCK_ID   sock_id,
                                CPU_INT32U    timeout_ms,
                                NET_ERR      *perr);
```

V1.80-001-c

Changed all `net_os.c` timeout configuration functions from returning timeout values in units of network ticks to returning timeout values in units of milliseconds.

Examples :

```
CPU_INT32U  NetOS_TxSuspendTimeoutGet_ms(NET_ERR  *perr);

CPU_INT32U  NetOS_Sock_RxQ_TimeoutGet_ms(NET_SOCK_ID   sock_id,
                                         NET_ERR      *perr);
```

V1.80-002

Added `NET_CFG_TX_SUSPEND_TIMEOUT_MS` to `net_cfg.h` to configure the network transmit/receive load balancing timeout.

Version 1.73
N/A

Version 1.72
N/A

Version 1.71
V1.71-001-a
Moved Network Status Monitor Task from ICMP Module to Network Debug Module.

V1.71-001-b
Moved ICMP Monitor Low Resource configuration to Network Debug Module:

```
NetDbg_CfgRsrcBufSmallThLo()
NetDbg_CfgRsrcBufLargeThLo()
NetDbg_CfgRsrcTmrThLo()
NetDbg_CfgRsrcConnThLo()
NetDbg_CfgRsrcARP_CacheThLo()
NetDbg_CfgRsrcTCP_ConnThLo()
NetDbg_CfgRsrcSockThLo()
```

V1.71-001-c
Changed ICMP Monitor Task configuration to ICMP Transmit Source Quench configuration.
I.e. '`NET_ICMP_CFG_MON_TASK_EN`' to '`NET_ICMP_CFG_TX_SRC_QUENCH_EN`'.

Version 1.70
N/A

Version 1.61
N/A

Version 1.60
N/A

Corrections

Version 1.91

V1.91-001

`NetTCP_TxConnReTxQ()` incorrectly closed a TCP connection when the number of TCP re-transmits was (greater than or) equal to the excessive retransmission threshold. Fixed by closing a TCP connection only when the number of TCP re-transmits was greater than the excessive retransmission threshold.

Version 1.90

N/A

Version 1.89

N/A

Version 1.88

V1.88-001

`NetSock_RxDataHandlerDatagram()` failed to check a network buffer pointer for `NULL` before accessing the pointer. Fixed by checking for `NULL` prior to accessing.

V1.88-002

`NetTCP_RxPktConnHandlerRxQ_Conn()` failed to transmit an immediate TCP acknowledgement segment when a received segment initially arrived out-of-order. Fixed by transmitting an immediate TCP acknowledgement whenever a received segment arrives out-of-order.

V1.88-003

Receive IP fragmentation reassembly algorithms failed to validate fragmented datagrams with the maximum IP total datagram length. Fixed by validating fragmented datagrams with the maximum IP total datagram length.

Version 1.87

V1.87-001

NetTCP_RxAppData() incorrectly handled received TCP close segments with TCP data by occasionally discarding a single octet of the received TCP data. Fixed by correctly handling all received TCP data & NOT discarding any TCP data octets. See also 'Corrections V1.80-001'.

V1.87-002a

NetSock_RxDataHandler()'s incorrectly returned '0' when no receive data was available instead of returning '-1' and appropriate receive error(s). Corrected by returning '-1' when no receive data is available & returning '0' only if the socket connection close()'s.

V1.87-002b

NetSock_TxDataHandler()'s incorrectly returned '0' when no transmit data was transmitted instead of returning '-1' and appropriate transmit error(s). Corrected by returning '-1' when no transmit data is transmitted & returning '0' only if the socket connection close()'s.

Version 1.86

V1.86-001

'NET_ARP_REQ_RETRY' defines incorrectly incremented the base retry values by one. Fixed by not incrementing the base retry values.

V1.86-002

NetDbg_ChkStatusHandlerTmrs() failed to initialize local variable 'tmr_nbr_used'. Fixed by initializing 'tmr_nbr_used'.

Version 1.85

V1.85-001a

`NetConn_CloseFromApp()` & `NetConn_CloseFromTransport()` failed to check for previously-closed network connections following any other network connection close operations. Fixed by checking for & skipping any previously-closed network connections.

V1.85-001b

`NetConn_CloseAllConnListConns()` failed to properly check for asynchronously-freed network connections while closing all network connections from a network connection list. Fixed by checking for & advancing past any asynchronously-freed network connections in the network connection list.

V1.85-002

`NetDbg_ChkStatus()` status functions failed to acquire the global network lock for asynchronous access by applications. Fixed by acquiring the global network lock.

Version 1.84

V1.84-001a

`NetTCP_RxPktConnHandlerTxWinRemote()` incorrectly allowed received duplicate acknowledgement segments to update the remote receive window. Fixed by preventing duplicate acknowledgement segments from updating the remote receive window.

V1.84-001b

`NetTCP_TxConnWinSizeHandlerCongCtrl()` failed to compensate the remote receive window update by the TCP connection's recently transmitted sequences. Fixed by compensating the remote receive window update by the number of recently transmitted sequences.

V1.84-002

`NetTCP_TxConnTxAck()` incorrectly invalidated received synchronization (SYN) or close (FIN) segments with no data as acknowledgement-only segments & therefore failed to transmit a TCP acknowledgement segment in reply. Fixed by validating received synchronization or close segments as valid TCP control segments that require acknowledgement.

Version 1.83

V1.83-001a

`NetTCP_RxPktConnHandlerRxQ_Conn()` failed to properly trim duplicate sequences prior to the next expected receive sequence for received segments whenever the TCP connection's transport receive queue was initially empty. Fixed by handling all received segments similarly, thereby properly trimming all duplicate sequences regardless of whether the transport receive queue is initially empty or non-empty.

V1.83-001b

`NetTCP_RxPktConnHandlerRxQ_Conn()` incorrectly cast received segments' buffer sequence numbers to 16-bit instead of 32-bit. Fixed by casting to 32-bit.

V1.83-001c

`NetTCP_RxPktConnHandlerRxQ_Conn()` incorrectly decremented received duplicate data segments' '`TCP_SegLen`' only without also decrementing '`TCP_SegLenData`'. Fixed by also decrementing '`TCP_SegLenData`'.

Version 1.82

V1.82-001

`NetTCP_TxConnReTxQ()` failed to properly prepare & re-transmit certain partially acknowledged segments in the re-transmit queue whose total transmit packet length was smaller than the minimum network interface packet size. Fixed by properly preparing all TCP re-transmit segments for the minimum network interface packet size.

V1.82-002

`NetTmr_TaskHandler()` failed to properly check for asynchronously-freed timers while handling timers on the Timer Task List. Fixed by checking for & advancing past any asynchronously-freed timers in the Timer Task List.

Version 1.81

N/A

Version 1.80

V1.80-001

`NetTCP_RxAppData()` incorrectly closed TCP connections that received TCP close segments with no TCP data. Fixed by correctly handling all received TCP close segments, with and without TCP data. See also 'Corrections V1.87-001'.

V1.80-002

`NetTCP_RxPktConnIsValidSeq()` incorrectly invalidated duplicate received TCP close segments. Fixed by appropriately handling received TCP close segments prior to validating the TCP close segment sequence numbers.

V1.80-003

`NetTCP_RxPktConnHandlerSeg()` did not appropriately call `NetTCP_RxPktConnHandlerReTxQ()` for received duplicate acknowledgements which also contained a TCP close flag. This prevented TCP connections in certain states from closing. Fixed by correctly handling all received TCP close segments, regardless of duplicate acknowledgements.

V1.80-004

`NetTCP_RxPktConnHandlerRxQ_AppData()` incorrectly discarded certain received TCP synchronization segments with TCP data. Fixed by appropriately handling all received TCP synchronization segments, with or without TCP data.

V1.80-005

`NetTCP_RxPktConnHandlerReTxQ()` incorrectly validated partially acknowledged segments in the re-transmit queue. Partially acknowledged segments are acknowledged to an aligned number of sequences to avoid data alignment errors (see 'Corrections V1.72-002'), but a partial acknowledgement delta must be used to validate new acknowledgement segments. Fixed by correctly validating partially acknowledged segments using the partial acknowledgement delta.

V1.80-006

`NetTCP_TxPktHandler()` incorrectly returned transitory transmit errors for TCP segments that were discarded. Fixed by returning fatal transmit errors.

V1.80-007

`NetTCP_TxConnTxQ()` incorrectly used only the amount of queued TCP transmit data octets in validating the next transmit segment for the Nagle algorithm. The algorithm should used the minimum of the queued TCP transmit data amount and the actual length of the next TCP segment to transmit. Fixed by correctly comparing and using the minimum of the queued TCP transmit data and the next TCP transmit segment length to validate the Nagle algorithm.

Version 1.73

V1.73-001

`inet_addr()` incorrectly returned `in_addr` structure. Fixed by returning `in_addr_t` IP address.

V1.73-002

`NetTCP_RxPktConnHandlerReTxQ()` failed to check the TCP re-transmit queue's updated head buffer as non-null before handling. Fixed by checking the re-transmit queue's new head buffer for non-null.

V1.73-003

Subnetted broadcast or specified host IP addresses incorrectly validated or handled. Fixed by correctly validating or handling these subnetted IP addresses.

Version 1.72

V1.72-001

`NetARP_CacheHandler()` incorrectly incremented transmit buffers' reference counter when queuing transmit buffers to a pending ARP cache. Fixed by not incrementing transmit buffers' reference counters.

V1.72-002

`NetTCP_RxPktConnHandlerReTxQ()` incorrectly updated partially acknowledged segments in the re-transmit queue. Advancing the segments by a non-aligned number of sequences created data alignment errors. Fixed by correctly advancing partially acknowledged segments by an aligned number of sequences.

Version 1.71

V1.71-001

`NetTCP_RxPktConnHandlerReTxQ()` incorrectly updated the TCP re-transmit queue's initial head buffer instead of the new head buffer when updating partial segments. Fixed by updating the re-transmit queue's new head buffer.

V1.71-002

`NetTCP_RxPktConnHandlerRxQ_Conn()` used incorrect TCP connection receive window variable '`RxWinSizeActual`' instead of '`RxWinSizeCfgdActual`' to sequence received segments. Fixed by using the correct TCP receive window variable.

V1.71-003

`NetTCP_RxConnWinSizeHandler()` always transmitted an immediate TCP acknowledgement whenever a TCP connection's local receive window updated, even if the acknowledgement should have been delayed. Fixed by delaying the acknowledgement, when applicable.

V1.71-004

`NetTCP_TxConnTxQ()` did not clear a pointer when moving segments from TCP transmit queue to TCP re-transmit queue. Fixed by clearing the pointer.

V1.71-005

`NetTCP_TxConnTxQ()` always requested a transmit silly window timer, even if the TCP connection had already or previously requested a timer. Fixed by not getting a new transmit silly window timer if the TCP connection previously requested a timer.

V1.71-006-a

`NetIP_RxPktValidate()` did not invalidate received IP datagrams with 'This Host' or specified host destination addresses. Fixed by invalidating these destination addresses.

V1.71-006-b

`NetIP_TxPktValidate()` did not validate transmit IP datagrams with a broadcast destination address. Fixed by validating the broadcast address as a destination address.

V1.71-007

`NetBuf_GetMaxSize()` always returned the maximum buffer data size for large buffers, even if the current buffer was a small buffer. Fixed by returning the maximum buffer data size for the current buffer, when applicable.

Version 1.70

V1.70-001

Duplicate acknowledgement segments incorrectly updated TCP connections' last unacknowledged sequence number. Fixed by updating TCP connections' last unacknowledged sequence number only with segments that acknowledge new data.

Version 1.61

V1.61-001

Data pointers advanced by packet lengths incorrectly cast the packet length increment to 8-bit, regardless of packet length size. Fixed by removing incorrect 8-bit cast.

V1.61-002

Random port numbers NOT freed back to random port number queue for stream socket `close()`. Fixed by freeing random port numbers for all socket `close()`.

Version 1.60

V1.60-001

`NetTCP_ConnReqClose()` failed to set TCP timeouts on transition to TCP `close()`. Fixed by adding/updating TCP timeouts on transition to TCP `close()`.

Known Problems

Version 1.91
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.90
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.89
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.88
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.87
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.86
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.85
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.84
V1.60-001 (Redesigned; see also 'Improvements V1.84-001')

V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 1.83

V1.73-001 (Unresolved)
V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.82

V1.73-001 (Unresolved)
V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.81

V1.73-001 (Unresolved)
V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.80

V1.73-001 (Unresolved)
V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.73

V1.73-001
Received IP broadcasts not demultiplexed to appropriate, non-wildcard-address socket(s).

V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.72

V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.71

V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.70

V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.61

V1.60-001 (Verification Ongoing)
V1.60-002 (Unresolved)

Version 1.60

V1.60-001
TCP/Socket/Connection Close Updates:
Verify that all normal & fault connection closes, close correctly.
(See also 'Improvements V1.84-001')

V1.60-002
IP route & timestamp options incorrectly implemented.

Limitations

001
Only supports a single :
(a) network interface (IF/NIC)
(b) host IP address
(c) default gateway

002
Following IP features NOT supported :
(a) IP forwarding/routing
(b) IP multicasting
(c) IP transmit fragmentation
(d) IP Security options
(e) ICMP Address Mask Agent/Server

003
Following TCP features NOT supported :
(a) TCP Urgent Data
(b) TCP Security & Precedence
(c) TCP Multihoming

004
Following socket features NOT supported :
(a) Socket shutdown()
(b) Multiple sockets bound to same socket
address or socket pair

Contacts

Micriμm
949 Crestview Circle
Weston, FL 33327
USA
+1 954 217 2036
+1 954 217 2037 (FAX)
e-mail: Licensing@Micrium.com
WEB: www.Micrium.com