Release Notes
V2.11.00

Micriμm

For the Way Engineers Work

Revision History

Version	Date	Description
V2.11.00	2011 Mar	Secure sockets New features, bug fixes, and improvements
V2.10.04	2011 Jan	Bug fixes and improvements
V2.10.03	2010 Dec	Bug fixes and improvements
V2.10.02	2010 Oct	New features, bug fixes, and improvements
V2.10.01	2010 Aug	Corrected µC/TCP-IP manual and book
V2.10	2010 Apr	Updated µC/TCP-IP manual and book New features, bug fixes, and improvements
V2.06	2009 Dec	New µC/OS-III port New features, bug fixes, and improvements
V2.05.02	2009 Oct	Bug fixes and improvements
V2.05.01	2009 Aug	New µC/FS V4 ports for µC/TCP-IP applications
V2.05	2009 Jul	IP/IGMP multicasting New features, bug fixes, and improvements
V2.04	2009 Apr	New features, bug fixes, and improvements
V2.03	2009 Mar	New features, bug fixes, and improvements
V2.02	2009 Feb	New features, bug fixes, and improvements
V2.01	2008 Nov	New features, bug fixes, and improvements
V2.00	2008 Jun	Multiple devices/interfaces/addresses Bug fixes and improvements
V1.92	2008 Jul	New features, bug fixes, and improvements
V1.91	2007 Oct	New features, bug fixes, and improvements
V1.90	2007 May	New features and improvements
V1.89	2007 Mar	New features and improvements
V1.88	2006 Nov	New features, bug fixes, and improvements
V1.87	2006 Sep	Bug fixes and improvements
V1.86	2006 Aug	Bug fixes and improvements
V1.85	2006 Jun	Bug fixes and improvements
V1.84	2006 Apr	New features, bug fixes, and improvements
V1.83	2005 Dec	Bug fixes and improvements

Version	Date	Description
V1.82	2005 Dec	Bug fixes and improvements
V1.81	2005 Oct	New features and improvements
V1.80	2005 Oct	Network communication performance improvements Bug fixes and improvements
V1.73	2005 Aug	TCP transmit round-trip time and retransmission controls Bug fixes and improvements
V1.72	2005 Jul	Bug fixes and improvements
V1.71	2005 May	Bug fixes and improvements
V1.70	2005 Apr	TCP transmit congestion and window controls Bug fixes and 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 and improvements
V1.54	2004 Dec	Bug fixes and improvements
V1.52	2004 Nov	Bug fixes and 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		

Required Modules

Version 2.11.00

µC/CPU version 1.27
µC/LIB version 1.34
µC/SSL version 1.08.00
(optional)

Version 2.10.04

µC/CPU version 1.27
µC/LIB version 1.34

Version 2.10.03

µC/CPU version 1.27
µC/LIB version 1.34

Version 2.10.02

µC/CPU version 1.27
µC/LIB version 1.30

Version 2.10.01

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.10

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.06

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.05.02

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.05.01

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.05

µC/CPU version 1.22
µC/LIB version 1.30

Version 2.04

µC/CPU version 1.22
µC/LIB version 1.27

Version 2.03

µC/CPU version 1.22
µC/LIB version 1.27

Version 2.02

µC/CPU version 1.20
µC/LIB version 1.27

Version 2.01

µC/CPU version 1.19
µC/LIB version 1.25

Version 2.00

µC/CPU version 1.18
µC/LIB version 1.25

Version 1.92

µC/CPU version 1.18
µC/LIB version 1.25

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 2.11.00

V2.11.00-001

Added support for secure sockets:

V2.11.00-001a

Added network security manager configuration in `net_cfg.h` to configure secure sockets feature:

`NET_SECURE_CFG_EN`	enable/disable security manager
`NET_SECURE_CFG_FS_EN`	enable/disable file system availability to install keying material
`NET_SECURE_CFG_VER`	configure secure protocol version
`NET_SECURE_CFG_WORD_SIZE`	configure optimized word size for security port (if applicable)
`NET_SECURE_CFG_CLIENT_DOWNGRADE_EN`	enable/disable client downgrade feature
`NET_SECURE_CFG_SERVER_DOWNGRADE_EN`	enable/disable server downgrade feature
`NET_SECURE_CFG_MAX_NBR_SOCK`	configure maximum number of sockets that can be secured
`NET_SECURE_CFG_MAX_NBR_CA`	configure maximum number of certificate authorities to install
`NET_SECURE_CFG_MAX_KEY_LEN`	configure maximum length (in octets) of keying material
`NET_SECURE_CFG_MAX_ISSUER_CN_LEN`	configure maximum length (in octets) of common name
`NET_SECURE_CFG_MAX_PUBLIC_KEY_LEN`	configure maximum length (in octets) of public key

V2.11.00-001b1

Added the following network security manager files:

```
\uC-TCPIP-V2\Secure\net_secure_mgr.h
\uC-TCPIP-V2\Secure\net_secure_mgr.c
```

V2.11.00-001b2

Added the following µC/SSL security port files:

```
\uC-TCPIP-V2\Secure\uC-SSL\net_secure.h
\uC-TCPIP-V2\Secure\uC-SSL\net_secure.c

\uC-TCPIP-V2\Secure\uC-SSL\OS\uCOS-II\net_secure_mgr.h
\uC-TCPIP-V2\Secure\uC-SSL\OS\uCOS-II\net_secure_mgr.c
\uC-TCPIP-V2\Secure\uC-SSL\OS\uCOS-III\net_secure_mgr.h
\uC-TCPIP-V2\Secure\uC-SSL\OS\uCOS-III\net_secure_mgr.c
```

V2.11.00-001c1

Added network security manager functions:

`NetSecureMgr_InstallBuf()`	installs a certificate authority, certificate, or private key from a buffer
`NetSecureMgr_InstallFile()`	installs a certificate authority, certificate, or private key from a file

V2.11.00-001c2

Added new secure socket configuration function:

`NetSock_CfgSecure()`	configures a socket's security mode

Version 2.10.04

V2.10.04-001

Added new IP function to get a configured host address:

`NetIP_GetAddrHostCfgd()`	returns the first configured IP address on the same local network as a specified remote host address

Version 2.10.03

N/A

Version 2.10.02

V2.10.02-001

Added new network interface function:

`NetIF_LinkStateWaitUntilUp()`	waits for a network interface's link state to be UP

V2.10.02-002

Added new socket configuration function:

`NetSock_CfgBlock()`	configures a socket for blocking or non-blocking operations

See also 'Changes V2.10.02-004'.

Version 2.10.01

N/A

Version 2.10

V2.10-001

Added new network interface functions:

`NetIF_GetRxDataAlignPtr()`	returns the first address in an application data buffer that is aligned with network buffer data areas in order to improve socket receive performance
`NetIF_GetTxDataAlignPtr()`	returns the first address in an application data buffer that is aligned with network buffer data areas in order to improve socket transmit performance

V2.10-002

Added new TCP configuration function:

`NetTCP_ConnCfgTxNagleEn()`	configures a TCP connection to enable/disable transmitting immediate acknowledgements for any received and pushed TCP segments

Version 2.06

V2.06-001a

Added `NET_CFG_BUILD_LIB_EN` in `net_cfg.h` to enable or disable building µC/TCP-IP as a linkable library. See your compiler's/linker's documentation for information on how to build a project into a linkable library.

V2.06-001b

Added `\uC-TCPIP-V2\Ports\IAR\dbg_net.c` IAR Embedded Workbench port file to declare µC/TCP-IP global variables and include their debug information when building µC/TCP-IP as a linkable library.

V2.06-002

Added new network debug information constants:

`NetBuf_Size`	size of network buffers (does not include size of network buffer data areas)
`NetBuf_PoolsSize`	size of network buffer pools (does not include size of network buffers or network buffer data areas)
`NetIP_IF_CfgSize`	size of IP/network interface address configuration management (see also 'Changes 2.06-001')

V2.06-003

Added `net_os.c` (µC/OS-III port) to support µC/OS-III V3.01.0 (and later versions).

Version 2.05.02

N/A

Version 2.05.01

V2.05.01-001

Added µC/FS V4 ports for µC/TCP-IP applications.

Version 2.05

V2.05-001

Added support for IP and IGMP multicasting:

V2.05-001a

Added NET_IP_CFG_MULTICAST_SEL in net_cfg.h to configure IP multicast feature:

NET_IP_MULTICAST_SEL_NONE	no multicasting functionality enabled
NET_IP_MULTICAST_SEL_TX	transmit† multicasting only enabled
NET_IP_MULTICAST_SEL_TX_RX	transmit† and receive multicasting enabled, as well as IGMP management

† Note that all IP multicast packets are transmitted with a default IP Time-To-Live (TTL) of 1. This transmit TTL value is not yet configurable globally or per-socket, but may be changed by modifying global NET_IP_TTL_MULTICAST_DFLT in net_ip.h. See also 'Changes V2.05-002'.

V2.05-001b

Added NET_IGMP_CFG_MAX_NBR_HOST_GRP in net_cfg.h to configure the maximum number of IGMP host groups to support. Required only if NET_IP_CFG_MULTICAST_SEL is configured to NET_IP_MULTICAST_SEL_TX_RX.

V2.05-001c

Added new network debug information constants:

NetIP_CfgMulticastSel	indicates which level of IP multicasting is configured—none, transmit only, transmit and receive (see 'New Features V2.05-001a')
NetIGMP_CfgMaxNbrHostGrp	indicates the maximum number of IGMP host groups addresses that can be joined to all interfaces

V2.05-001d

Added new IGMP functions to join/leave multicast host groups:

NetIGMP_HostGrpJoin()	join a multicast host group address on an interface
NetIGMP_HostGrpLeave()	leave a multicast host group address on an interface

V2.05-001e

Added new IP status functions:

NetIP_IsAddrClassD()	indicates whether an address is a Class-D IP address
NetIP_IsAddrMulticast()	indicates whether an address is a multicast IP address

See also 'New Features V1.90-003b and V1.92-001'.

Version 2.04

V2.04-001a

Added new application socket handling example functions:

NetApp_SockOpen()
NetApp_SockClose()
NetApp_SockBind()
NetApp_SockConn()
NetApp_SockListen()
NetApp_SockAccept()
NetApp_SockRx()
NetApp_SockTx()

V2.04-001b

Added new application time delay function:

NetApp_TimeDly_ms()

V2.04-001c

Added NET_APP_CFG_API_EN in net_cfg.h to enable or disable new application handler functions.

Version 2.03

V2.03-001

Added configuration to swap data octets read from or written to an Ethernet device [i.e., swap data words' high-order octet(s) with data words' low-order octet(s), and vice-versa] if required by device-to-CPU data bus wiring and/or CPU endian word order:

DEF_NO	Device driver data octets not swapped (default)
DEF_YES	Device driver data octets swapped

See device configuration template file (\uC-TCPIP-V2\Cfg\Template\net_dev_cfg.c).

Version 2.02

V2.02-001

Refactored network socket status function:

NetSock_IsConn()	indicates whether a network socket is connected

V2.02-002

Added new socket connection function:

NetSock_GetConnTransportID()	gets a socket's TCP transport layer connection handle identifier (ID)

Version 2.01

V2.01-001

Added new network debug information constants:

`Net_CfgOptimizeAsmEn`	indicates whether assembly optimization functions are enabled or disabled (see also 'Improvements V2.01-001')
`NetIF_CfgMaxNbrIF`	indicates the maximum number of network interfaces that can be added at run-time
`NetIF_CfgAddrFltrEn`	indicates whether network interface address filtering is enabled or disabled (see also 'Changes V2.01-001c')
`NetIF_CfgLoopbackEn`	indicates whether network loopback interface is enabled or disabled
`NetIF_CfgEtherEn`	indicates whether Ethernet interfaces are enabled or disabled
`NetIP_CfgIF_MaxNbrAddr`	indicates the maximum number of IP addresses that can be added at run-time to a network interface

See also 'Changes V2.01-001c'.

Version 2.00

V2.00-001

Added support for multiple network devices, interfaces, and addresses:

V2.00-001a

Added new network interface configuration:

`NetIF_Add()`	configures a network device and interface
`NetIF_Start()`	starts a network device and interface
`NetIF_Stop()`	stops a network device and interface

V2.00-001b

Added new loopback interface with configuration.

V2.00-001c

Added new network interface IP address configuration:

`NetIP_CfgAddrAdd()`	configures a network interface IP address
`NetIP_CfgAddrRemove()`	removes a network interface IP address

Version 1.92

V1.92-001

Added new IP status functions:

`NetIP_IsAddrHost()`	indicates whether an address is a host IP address
`NetIP_IsAddrHostCfgd()`	indicates whether an address is a configured host IP address

See also 'New Features V1.90-003b and V2.05-001e'.

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 for this host 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

See also 'New Features V1.92-001 and V2.05-001e'.

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 and 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 and 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 and Delayed Acknowledgements
- Receive Silly Window Syndrome Avoidance
- Receive Window Configuration / Control

V1.60-002

TCP Transmit Segment Aggregation

- Data
- FIN-Close
- PUSH

Improvements

Version 2.11.00

V2.11.00-001

Updated µC/TCP-IP's CERT-C and MISRA-C compliance:

V2.11.00-001a

Removed `NetBuf_Free()`'s recursive call to free IP options buffer, when available.

Version 2.10.04

V2.10.04-001

Improved overall communication throughput performance by replacing many `Mem_Copy()` and memory get macros, `NET_UTIL_VAL_GET_???()`, with calls to memory copy macros, `NET_UTIL_VAL_COPY_???()`.

V2.10.04-002

Modified socket `connect()` and datagram socket `send()` functions to connect or transmit on the same local network as the specified remote host address, whenever possible. See also 'New Features V2.10.04-001'.

Version 2.10.03

V2.10.03-001

Updated µC/TCP-IP's CERT-C and MISRA-C compliance:

V2.10.03-001a1

Removed 'u' qualifier from certain integer constants. This reverts a previously implemented improvement only for certain integer constants that may be used in both signed and unsigned expressions. See also 'Improvements V2.05.02-001a1'.

V2.10.03-001a2

Added 'L' qualifier to certain long integer constants. This reverts a previously incorrect assumption about certain integer data type and constant promotions. See also 'Improvements V2.05.02-001a2'.

V2.10.03-002

Modified `NET_SOCK_RTN_CODE`'s data type definition from `CPU_INT16S` to `CPU_INT32S` to handle returning all transport layers' 16-bit maximum positive return values. See also 'Improvements V2.05.02-002'.

Version 2.10.02

V2.10.02-001

Refactored `NetIF_Add()` to not increment `NetIF_NbrNext` on any error(s), which was required prior to µC/TCP-IP V2.10 in order to synchronize network interface numbers between added network interfaces and their device and/or BSP function handlers.

Version 2.10.01

N/A

Version 2.10

V2.10-001

Updated µC/TCP-IP's CERT-C and MISRA-C compliance:

V2.10-001a

Added argument names to function pointer data types.

V2.10-001b

Encapsulated all macros defined as code blocks within `do..while(0)` conditions.

V2.10-002

Refactored network interface layer to reduce code and increase consistency between functions/modules/layers.

V2.10-003

Removed loopback receive task and queue.

V2.10-004a

Added `Net_InitDone` validation to all network protocol suite API functions that may be called directly by application function(s).

V2.10-004b

Removed `Net_InitDone` validation from all internal network protocol suite functions that are not called directly by application function(s).

V2.10-005

Refactored network buffer unlink functions to be defined as `NET_BUF_FNCT` function type [i.e., `(void) (NET_BUF *)`]. This complies with ISO IEC 9899-1999 ANSI-C, Sections 6.3.2.3.1 and 6.3.2.3.7 and avoids potentially incorrect pointer conversion behavior between `void` pointer parameters and network buffer pointers. See also 'Changes V1.86-002'.

V2.10-006

Moved global variable declarations from `net_os.c` to `net_os.h` (µC/OS-II and µC/OS-III ports) in order to include their debug information when building µC/TCP-IP as a linkable library. See also 'Changes V2.10-008' & 'New Features V2.06-001'.

Version 2.06

V2.06-001

Updated µC/TCP-IP's CERT-C and MISRA-C compliance:

V2.06-001a1

Replaced generic `void` pointer function casts with appropriate function-specific pointer casts in network interface, device, and physical layers.

V2.06-001a2

Added appropriate casts to/from `void` pointers to specific network interface/device/physical layer configuration, data, or API structures.

V2.06-002

Added `net_os.c` (µC/OS-III port) to support µC/OS-III V3.01.0 (and later versions) [see also 'New Features V2.06-003'].

Version 2.05.02

V2.05.02-001

Updated µC/TCP-IP's CERT-C and MISRA-C compliance:

V2.05.02-001a1

Appended unsigned 'u' qualifier to all unsigned integer constants.

V2.05.02-001a2

Removed redundant 'L' qualifier from all long integer constants.

V2.05.02-001b

Replaced all instances of '???' comments with '&&&' (to avoid possible usage of C trigraphs).

V2.05.02-001c

Refactored the following functions to copy any function arguments into local variables before modifying:

`NetApp_SockRx()`

V2.05.02-001d

Replaced all calls to unbounded µC/LIB string library functions [e.g., `Str_Copy()`] in `net_os.c` (µC/OS-II port) with calls to bounded functions [e.g., `Str_Copy_N()`].

V2.05.02-002

Modified `NET_SOCK_ADDR_LEN`'s data type definition from `CPU_INT16S` to `CPU_INT32S` to conform with BSD `socklen_t` data type, which is required to be a signed integer of at least 32 bits. See also 'Improvements V2.10.03-002'.

V2.05.02-003a

Refactored `NET_CTR_INC()` and `NET_CTR_INC_LARGE()` to call new network counter functions `NetCtr_Inc()` and `NetCtr_IncLarge()`, respectively.

V2.05.02-003b

Re-added critical section locks when accessing network counter variables. Required for API functions that call network counter macros but do not acquire the global network lock. See also 'Improvements V1.92-002b'.

V2.05.02-004

Removed all unused/deprecated `NET_TMR_TIME_???` values.

Version 2.05.01

N/A

Version 2.05

V2.05-001

Added new network buffers' protocol header type parameters:

`ProtocolHdrTypeIF_Sub`	Network interface sub-protocol layer (e.g., ARP) protocol header type
`ProtocolHdrTypeNetSub`	Network sub-protocol layer (e.g., ICMP) protocol header type

See also 'Improvements V2.04-001'.

V2.05-002

Updated `net_os.c` (µC/OS-II port) to support µC/OS-II V2.88 (and later versions).

Version 2.04

V2.04-001

Added new network buffers' protocol header type parameters:

`ProtocolHdrTypeIF`	Network interface layer (e.g., Ethernet) protocol header type
`ProtocolHdrTypeNet`	Network layer (e.g., IP) protocol header type
`ProtocolHdrTypeTransport`	Transport layer (e.g., TCP) protocol header type

See also 'Improvements V2.05-001'.

Version 2.03

V2.03-001

Improved `NetIF_Ether_IF_Add()` by validating all device and physical layer configuration parameters.

V2.03-002

Replaced all device driver register data type definitions with µC/CPU's new volatile `CPU_REGxx` data types.

V2.03-003

Replaced all `cpu_sr` local variable declarations with µC/CPU's new `CPU_SR_ALLOC()` macro. See also 'Corrections V2.04-002'.

Version 2.02

V2.02-001

Refactored network buffer configuration to allow zero (0) number of small transmit buffers OR zero (0) number of large transmit buffers to be configured. However, network buffer configuration still does not allow both zero (0) number of small AND zero (0) number of large transmit buffers to be configured. See also 'Changes V2.02-001'.

V2.02-002

Refactored `NetBuf_Get()` to allocate a large transmit buffer whenever no small transmit buffers are available.

Version 2.01

V2.01-001

Added `NET_CFG_OPTIMIZE_ASM_EN` in `net_cfg.h` to enable or disable assembly optimization functions. See also 'Changes V2.01-001a'.

V2.01-002

Added names to TCP and socket connection signals in `net_os.c` (µC/OS-II port).

Version 2.00

N/A

Version 1.92

V1.92-001

Refactored configuration functions to fail if desired configuration parameters were invalid rather than configure values to default values.

V1.92-002a

Reorganized all network counter variables into two structures – `Net_StatCtrs` for network statistics counters and `Net_ErrCtrs` for network error counters.

V1.92-002b

Removed critical section locks when accessing network counter variables. See also 'Improvements V2.05.02-003b'.

V1.92-003

Refactored `NetSock_BindHandler()` to allow binding to a localhost IP address. See also 'Corrections V1.92-002'.

V1.92-004

Improved Network Connection Management to reduce connection search times.

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 and OS tick overflow detection.

Version 1.88

V1.88-001

Refactored `NetTCP_TxConnClose()` to remove logically 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 and 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 callback functions to be defined as `CPU_FNCT_PTR` function type [i.e., `(void) (void *)`] and 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 and 6.3.2.3.7 and avoids potentially incorrect pointer conversion behavior between `void` pointer parameters and callback functions' object pointers. See also 'Improvements V2.10-005'.

Version 1.85

V1.85-001

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

Version 1.84

V1.84-001

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

V1.84-002

Modified `NetTCP_ConnClose()` 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 and included in the project build:

DEF_DISABLED	BSD 4.x API functions disabled and excluded from build
DEF_ENABLED	BSD 4.x API functions enabled and 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 and configuration.

Version 1.80

V1.80-001

Network Transmit/Receive Load Balancing

- Handle network transmit and receive packets in an approximately balanced ratio.

V1.80-002

Improved NetIP_TxPktDatagramNextHopSel()'s IP address host, network, and 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 macros 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 2.11.00

V2.11.00-001

Modified the following network socket handler functions to be defined as global functions since they may now be called via network security manager port files:

`NetSock_RxDataHandlerStream()`
`NetSock_TxDataHandlerStream()`

Version 2.10.04

N/A

Version 2.10.03

V2.10.03-001

Moved `NetOS_TimeDly()` function prototypes from `net_os.h` into `net.h`.

Version 2.10.02

V2.10.02-001a

For consistency with other configuration functions, modified the following functions to return `DEF_OK` / `DEF_FAIL` :

`NetSock_CfgTimeoutRxQ_Dflt()`
`NetSock_CfgTimeoutRxQ_Set()`
`NetSock_CfgTimeoutTxQ_Dflt()`
`NetSock_CfgTimeoutTxQ_Set()`
`NetSock_CfgTimeoutConnReqDflt()`
`NetSock_CfgTimeoutConnReqSet()`
`NetSock_CfgTimeoutConnAcceptDflt()`
`NetSock_CfgTimeoutConnAcceptSet()`
`NetSock_CfgTimeoutConnCloseDflt()`
`NetSock_CfgTimeoutConnCloseSet()`

V2.10.02-001b

For consistency with other complex configuration functions [e.g., `NetSock_CfgTimeout???()` or `NetIP_CfgAddr???()`], modified the following functions to return error codes via `perr` argument:

`NetTCP_ConnCfgMaxSegSizeLocal()`
`NetTCP_ConnCfgRxWinSize()`
`NetTCP_ConnCfgReTxMaxTh()`
`NetTCP_ConnCfgReTxMaxTimeout()`
`NetTCP_ConnCfgTxNagleEn()`
`NetTCP_ConnCfgTxAckImmedRxdPushEn()`

V2.10.02-002

Renamed network socket object flags from NET_SOCK_FLAG_??? to NET_SOCK_FLAG_SOCK_??? to avoid confusion with network socket API flags, which were also named NET_SOCK_FLAG_???.

V2.10.02-003

Renamed network socket flag NET_SOCK_FLAG_SOCK_BLOCK to NET_SOCK_FLAG_NO_BLOCK to change network sockets' blocking flag default (i.e., when bit-flag is clear) from indicating non-blocking to blocking. See also 'New Features V2.10.02-002'.

V2.10.02-004

Modified NET_SOCK_CFG_BLOCK_SEL from globally configuring all sockets' blocking behavior at compile-time to configuring each socket's initial default blocking behavior. See also 'New Features V2.10.02-002'.

V2.10.02-005a

Changed NET_TCP_RE_TX_TH_MIN value from 6 to 5.

V2.10.02-005b

Changed NET_TCP_RE_TX_TH_DFLT value from 12 to 11.

Version 2.10.01

N/A

Version 2.10

V2.10-001

Modified device/interface configuration structures. See template device configuration file (\uC-TCPIP-V2\Cfg\Template\net_dev_cfg.c) for examples and notes on the changes to configuration structures (NET_DEV_CFG_ETHER and NET_IF_CFG_LOOPBACK):

V2.10-001a

Added network buffer data offsets for devices or drivers that require additional octets be reserved prior to actual receive or transmit packets.

V2.10-001b

Added (optional) device configuration bit flags:

NET_DEV_CFG_FLAG_NONE	No device configuration flags selected
NET_DEV_CFG_FLAG_SWAP_OCTETS	Swap data octets [i.e., swap data words' high-order octet(s) with data words' low -order octet(s), and vice-versa] if required by device-to-CPU data bus wiring and/or CPU endian word order (see also 'Changes V2.10-001c')

V2.10-001c

Deprecated NET_DEV_CFG_ETHER's DataOctetSwap configuration flag and replaced with NET_DEV_CFG_FLAG_SWAP_OCTETS flag (see 'Changes V2.10-001b').

V2.10-002a

Deprecated global device BSP functions and replaced with device-specific BSP functions:

```
void  NetDev_CfgClk<DeviceNumber>     (NET_IF   *pif,
                                       NET_ERR  *perr);
void  NetDev_CfgIntCtrl<DeviceNumber>(NET_IF   *pif,
                                       NET_ERR  *perr);
void  NetDev_CfgGPIO_<DeviceNumber>   (NET_IF   *pif,
                                       NET_ERR  *perr);

CPU_INT32U  NetDev_ClkFreqGet<DeviceNumber>(NET_IF   *pif,
                                            NET_ERR  *perr);
```

where <DeviceNumber> is the (optional) device number for each specific instance of a device on a development board.

V2.10-002b

Added device BSP interface structures to configure each device's BSP functions. See BSP template file (\uC-TCPIP-V2\BSP\Template\net_bsp.c) for examples and notes on how to configure an Ethernet device's BSP interface structure (NET_DEV_BSP_ETHER).

V2.10-003

Added dev_bsp argument to NetIF_Add() to configure devices' BSP functions:

```
NET_IF_NBR  NetIF_Add(void     *if_api,
                      void     *dev_api,
                      void     *dev_bsp,
                      void     *dev_cfg,
                      void     *phy_api,
                      void     *phy_cfg,
                      NET_ERR  *perr);
```

See also 'Changes V2.10-002b'.

V2.10-004

Deprecated asynchronous physical link state updates via NetIF_LinkStateSet() for periodically polled link state updates via NetIF_PhyLinkStateHandler().

V2.10-005

Renamed network interface receive buffer index from NET_IF_RX_IX to NET_IF_IX_RX.

V2.10-006

Added `pix_offset` argument to the following functions to return the interface's configured network buffer receive/transmit index offsets:

```
NET_BUF     *NetBuf_Get        (NET_IF_NBR        if_nbr,
                                NET_TRANSACTION   transaction,
                                NET_BUF_SIZE      size,
                                NET_BUF_SIZE      ix,
                                NET_BUF_SIZE     *pix_offset,
                                CPU_INT16U        flags,
                                NET_ERR          *perr);

CPU_INT08U  *NetBuf_GetDataPtr(NET_IF            *pif,
                                NET_TRANSACTION   transaction,
                                NET_BUF_SIZE      size,
                                NET_BUF_SIZE      ix,
                                NET_BUF_SIZE     *pix_offset,
                                NET_BUF_SIZE     *p_data_size,
                                NET_TYPE         *ptype,
                                NET_ERR          *perr);
```

See also 'Changes V2.10-001a'.

V2.10-007

Added `NET_DBG_STATUS_FAULT_INIT` / `NET_DBG_SF_INIT` return status codes to `NetDbg_ChkStatus???()` / `NetDbg_GetStatus???()` functions.

V2.10-008

Moved global variable declarations from `net_os.c` to `net_os.h` (µC/OS-II and µC/OS-III ports) in order to include their debug information when building µC/TCP-IP as a linkable library. See also 'Improvements V2.10-006' & 'New Features V2.06-001'.

V2.10-009

Replaced all instances of global configuration constant `OS_CFG_TICK_RATE_HZ` in all µC/OS-III ports (e.g., `net_os.c` and `net_bsp.c`) with either run-time variable `OSCfg_TickRate_Hz` or an appropriate #define constant since application configuration constants (such as `OS_CFG_TICK_RATE_HZ`) are not guaranteed to be available during later compilation when µC/TCP-IP is provided as a linkable library. See also 'New Features V2.06-001'.

Version 2.06

V2.06-001

Renamed the following `net_dbg.h` debug information constant:

`NetIP_InfoTblSize` to `NetIP_IF_CfgTblSize`

See also 'New Features V2.06-002'.

V2.06-002a

Reordered Ethernet device API structure (NET_DEV_API_ETHER) to group the multicast address API functions [AddrMulticast???()] prior to the ISR and I/O API functions to more closely parallel the order of API functions in the network interface API structure (NET_IF_API). See Ethernet device template files (\uC-TCPIP-V2\Dev\Ether\Template\net_dev_ether_template_???.c) for examples of the changes to the device API structure.

V2.06-002b

Reordered Ethernet physical layer API structure (NET_PHY_API_ETHER) to group the enable/disable API function [EnDis()] immediately following the initialization API function [Init()] to more closely parallel the order of the start/stop API functions in the network interface API structure (NET_IF_API). See generic Ethernet physical layer file (\uC-TCPIP-V2\Dev\Ether\Phy\Generic\net_phy.c) for examples on the specific changes to the physical layer API structure.

V2.06-003

Modified NetIF_PerfMonHandler() to no longer clear but to latch a network interface's performance calculations when a network interface is disabled. See also 'Corrections V2.06-004'.

V2.06-004

Modified NetSock_ConnHandlerStream() to wait for connections-in-progress to connect using the socket's connection request timeout even on socket connect retries. See also 'Corrections V2.06-001'.

V2.06-005a

Reordered NetTmr_Get()'s callback function and object arguments.

V2.06-005b

Modified NetTmr_Get() to accept NULL objects for callback functions that do not require or use objects in order to execute.

V2.06-006

Removed calls to NetBSP_Dly_ms() from device drivers. Replaced with calls to NetOS_TimeDly_ms().

Version 2.05.02

V2.05.02-001a

Modified µC/TCP-IP version number to include sub-minor version numbers:

Vx.yy.zz

where:

V	denotes 'Version' label
x	denotes major software version revision number
yy	denotes minor software version revision number
zz	denotes sub-minor software version revision number

V2.05.02-001b

Modified NET_VERSION to include sub-minor version numbers, formatted as follows:

```
version = x.yyzz * 100 * 100
```

where:

version	denotes software version number scaled as an integer value
x.yyzz	denotes software version number, where the unscaled integer value denotes the major version number and the unscaled fractional value denotes the minor version numbers

V2.05.02-002

Modified NET_SOCK_ADDR_LEN's data type definition from CPU_INT16S to CPU_INT32S. See also 'Improvements V2.05.02-002'.

Version 2.05.01

N/A

Version 2.05

V2.05-001a

Renamed the following net_cfg.h configuration constant:

NET_ICMP_CFG_TX_SRC_QUENCH_SIZE to NET_ICMP_CFG_TX_SRC_QUENCH_NBR

V2.05-001b

Renamed the following net_dbg.h debug information constant:

NetICMP_CfgTxSrcQuenchSize to NetICMP_CfgTxSrcQuenchNbr

V2.05-002

Renamed the following IP transmit Time-To-Live (TTL) values:

NET_IP_HDR_TTL_NONE to NET_IP_TTL_NONE
NET_IP_HDR_TTL_MIN to NET_IP_TTL_MIN
NET_IP_HDR_TTL_MAX to NET_IP_TTL_MAX
NET_IP_HDR_TTL_DFLT to NET_IP_TTL_DFLT

V2.05-003

Deprecated OS_SEM_OVF error codes and replaced with newer OS_ERR_SEM_OVF error code in net_os.c (µC/OS-II port).

Version 2.04

V2.04-001

Renamed the following net_err.h error codes to comply with ANSI 31-character symbol length uniqueness:

```
NET_ARP_ERR_INVALID_PROTOCOL_ADDR_LEN to NET_ARP_ERR_INVALID_PROTOCOL_LEN
NET_MGR_ERR_INVALID_PROTOCOL_ADDR_LEN to NET_MGR_ERR_INVALID_PROTOCOL_LEN
```

Version 2.03

V2.03-001

Renamed Ethernet device configuration structure's (NET_DEV_CFG_ETHER) data bus size parameter from BusDataSizeNbrBits to DataBusSizeNbrBits.

Version 2.02

V2.02-001

Reordered Ethernet device configuration structures (NET_DEV_CFG_ETHER) to group dedicated memory configuration prior to all Ethernet-specific configuration. See device configuration template file (\uC-TCPIP-V2\Cfg\Template\net_dev_cfg.c) for examples on the specific changes. See also 'Changes V2.10-001'.

V2.02-002

Added pif argument to all network interface API functions.

V2.02-003

Added ptype argument to NetBuf_GetDataPtr() to return allocated network buffer type:

```
CPU_INT08U  *NetBuf_GetDataPtr(NET_IF           *pif,
                               NET_TRANSACTION   transaction,
                               NET_BUF_SIZE      size,
                               NET_BUF_SIZE      ix_start,
                               NET_BUF_SIZE     *p_data_size,
                               NET_TYPE         *ptype,
                               NET_ERR          *perr);
```

Version 2.01

V2.01-001a

Replaced uC_CFG_OPTIMIZE_ASM_EN from app_cfg.h with NET_CFG_OPTIMIZE_ASM_EN in net_cfg.h. See also 'Improvements V2.01-001'.

V2.01-001b

Renamed the following net_cfg.h configuration constants:

NET_IF_CFG_NBR_IF to NET_IF_CFG_MAX_NBR_IF

NET_CFG_TX_SUSPEND_TIMEOUT_MS to NET_IF_CFG_TX_SUSPEND_TIMEOUT_MS
(see also 'Changes V1.80-002')

V2.01-001c

Renamed the following net_dbg.h debug information constant:

NetARP_CfgAddrFilterEn to NetARP_CfgAddrFltrEn

See also 'New Features V2.01-001'.

V2.01-002

Renamed network socket IPv4 wildcard address from NET_SOCK_ADDR_IP_WILD_CARD to NET_SOCK_ADDR_IP_WILDCARD (i.e., removed the spurious underscore from WILD_CARD).

V2.01-003

Renamed the following net_err.h error codes:

NET_ASCII_ERR_INVALID_LEN to NET_ASCII_ERR_INVALID_STR_LEN

NET_ARP_ERR_INVALID_HW_LEN to NET_ARP_ERR_INVALID_HW_ADDR_LEN
NET_ARP_ERR_INVALID_PROTOCOL_LEN to NET_ARP_ERR_INVALID_PROTOCOL_ADDR_LEN

See also 'Changes V2.04-001'.

V2.01-004

Replaced NetTCP_TxConnRTT_GetTS_ms() with NetUtil_TS_Get_ms() in net_bsp.c:

```
NET_TS_MS  NetUtil_TS_Get_ms(void);
```

See also 'Corrections V2.01-003'.

V2.01-005a

Changed `net_dbg.c` network resource low configuration functions to configure low thresholds and hystereses in units of percentage:

```
CPU_BOOLEAN  NetDbg_CfgRsrcTmrThLo       (CPU_INT08U  th_pct,
                                          CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcConnThLo      (CPU_INT08U  th_pct,
                                          CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcARP_CacheThLo(CPU_INT08U  th_pct,
                                          CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcTCP_ConnThLo (CPU_INT08U  th_pct,
                                          CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcSockThLo      (CPU_INT08U  th_pct,
                                          CPU_INT08U  hyst_pct);
```

V2.01-005b

Added `net_dbg.c` network buffer resource low configuration functions:

```
CPU_BOOLEAN  NetDbg_CfgRsrcBufThLo        (CPU_INT08U  th_pct,
                                           CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcBufRxLargeThLo(CPU_INT08U  th_pct,
                                           CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcBufTxSmallThLo(CPU_INT08U  th_pct,
                                           CPU_INT08U  hyst_pct);
CPU_BOOLEAN  NetDbg_CfgRsrcBufTxLargeThLo(CPU_INT08U  th_pct,
                                           CPU_INT08U  hyst_pct);
```

Version 2.00

N/A

Version 1.92

V1.92-001a

Added `addr_src` argument to `NetOS_ICMP_TxMsgReq()`.

V1.92-001b

Reordered IP address arguments for `NetOS_ICMP_TxMsgReq()`:

```
void  NetOS_ICMP_TxMsgReq(CPU_INT08U    type,
                          CPU_INT08U    code,
                          NET_IP_ADDR   addr_src,
                          NET_IP_ADDR   addr_dest,
                          NET_IP_TOS    TOS,
                          NET_IP_TTL    TTL,
                          CPU_INT16U    flags,
                          void         *popts,
                          void         *p_data,
                          CPU_INT16U    data_len,
                          NET_ERR      *perr);
```

V1.92-002a

Reorganized all network counter variables into two structures – `Net_StatCtrs` for network statistics counters and `Net_ErrCtrs` for network error counters.

V1.92-002b

Removed critical section locks when accessing network counter variables. See also 'Improvements V2.05.02-003b'.

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` and `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 and transmit functions' application data length parameters – `data_buf_len` and `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        *p_data,
                         _size_t       data_len,
                          int          flags,
                  struct  sockaddr    *paddr_remote,
                          socklen_t    addr_len);

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

V1.91-004b

Changed network socket receive and transmit functions' application data length parameters – `data_buf_len` and `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 and 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 and 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 and 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 and 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 and 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 and 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

Renamed network debug information and status names to remove redundant 'DBG' token. Examples:

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

Modified `NetTCP_TxConnAck()` to transmit immediate acknowledgments for any received and 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. See also 'Changes V2.01-001b'.

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

Renamed 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 2.11.00

V2.11.00-001

`NetSock_Close()` failed to return error codes from `NetSock_CloseHandlerStream()`. Fixed by returning `NetSock_CloseHandlerStream()` error codes.

Version 2.10.04

V2.10.04-001

`NetIF_GetDataAlignPtr()` failed to invalidate network buffer alignments configured to zero (0) octets which prevents determining whether application data buffer and network buffer data area alignment is possible. Fixed by not returning an aligned application data buffer address for zero-sized network buffer alignments.

V2.10.04-002

The following functions failed to convert received packet IP options from network order to host order using CPU word-aligned addresses (for processors that require data word alignment):

```
NetIP_RxPktValidateOptRoute()
NetIP_RxPktValidateOptTS()
```

Fixed by converting IP options using appropriate word-aligned memory macros.

Version 2.10.03

V2.10.03-001

Starting in V2.05.02, the following socket layer functions failed to correctly convert `NET_SOCK_ADDR_LEN` arguments into `NET_CONN_ADDR_LEN` parameters for big-endian processors/architectures [since `NET_SOCK_ADDR_LEN`'s base data type was modified (see 'Improvements V2.05.02-002' & 'Changes V2.05.02-002')]:

```
NetSock_Accept()
NetSock_BindHandler()
NetSock_ConnStreamHandler()
NetSock_ConnHandlerAddrRemoteValidate()
NetSock_RxDataHandlerStream()
NetSock_TxDataHandlerDatagram()
NetSock_FreeAddr()
```

Fixed by changing `addr_len` local variables to `NET_CONN_ADDR_LEN` data type to correctly convert from `NET_SOCK_ADDR_LEN`.

V2.10.03-002

`NetTCP_RxPktValidate()` failed to transmit a TCP reset for received segments with any invalid TCP option fields. Fixed by transmitting a TCP reset.

V2.10.03-003

NetTCP_TxConnTxQ() incorrectly assumed that if the TCP Nagle algorithm is disabled, the transmit Silly Window Syndrome avoidance algorithm must still be checked in order to transmit small TCP segments. However, this directly contradicts the algorithm provided in RFC #1122, Section 4.2.3.4.(2) which allows TCP to transmit a small segment if the TCP connection's useable transmit window is large enough and the segment's Push (PSH) bit is set. Fixed by transmitting a small TCP segment if the TCP connection's useable transmit window is large enough, the segment's Push bit is set, and the Nagle algorithm is disabled on the TCP connection—without further checking for Silly Window Syndrome avoidance or any other congestion controls.

V2.10.03-004

Starting in V2.05.02, NetTCP_TxConnRTT_RTO_Calc() incorrectly calculated a TCP connection's RTO values since signed RTT/RTO constants (in net_tcp.h) were incorrectly modified to append the unsigned 'u' qualifier (see 'Improvements V2.05.02-001a1'). Fixed by removing the unsigned 'u' qualifier from all signed RTT/RTO constants.

V2.10.03-005

NetTCP_TxConnRTT_RTO_Calc() failed to update a TCP connection's scaled RTO timeout control (TxRTT_RTO_ms_scaled) when performing RTO back-off calculations. (Since the TCP connection's scaled RTO timeout control is unnecessary for RTO back-off, this bug was merely an artifact where the scaled RTO timeout was inconsistent with the TCP connection's other RTO controls.) Fixed by updating all TCP connection RTO controls when performing all RTO calculations.

V2.10.03-006

NetTCP_TxConnWinSizeZeroWinHandler() failed to acquire and configure a network timer in order for a TCP connection to transmit zero window probes. Fixed by correctly checking if a TCP connection needs to acquire and configure its zero window timer in order to periodically transmit zero window probes.

V2.10.03-007

NetIP_RxPktValidateOpt() failed to allocate a network buffer data area to copy the received packet's IP options. Fixed by allocating a network buffer data area.

Version 2.10.02

V2.10.02-001

NetTCP_TxConnReTxQ() failed to translate NetIP_ReTx()'s return error NET_IP_ERR_NONE. Fixed by translating to NET_TCP_ERR_NONE.

V2.10.02-002

NetTCP_ConnCloseTimeout() failed to properly clear the TCP connection's network timer pointer. Fixed by clearing the network timer pointer. See also 'Corrections V2.01-002'.

V2.10.02-003

Passive TCP connections established in NetTCP_RxPktConnHandlerListen() failed to correctly initialize the configured RTO timeouts copied from the listen TCP connection. Fixed by calling NetTCP_ConnCfg() from NetTCP_RxPktConnHandlerListen() [via NetTCP_ConnCopy()] to initialize configured RTO timeout controls.

V2.10.02-004

NetUDP_TxAppData() failed to return NetOS_Lock()'s error code via perr argument. Fixed by returning NetOS_Lock()'s error code when necessary.

V2.10.02-005

NET_IF_HDR_SIZE_MAX incorrectly not configured if Ethernet interface disabled but network loopback interface enabled (i.e., NET_IF_CFG_ETHER_EN configured to DEF_DISABLED and NET_IF_CFG_LOOPBACK_EN configured to DEF_ENABLED). Fixed by always configuring NET_IF_HDR_SIZE_MAX even when only network loopback interface is enabled.

V2.10.02-006

NetIF_Loopback_Rx() incorrectly incremented Net_ErrCtrs.NetIF_Ether_ErrRxPktDiscardedCtr error counter. Fixed by incrementing Net_ErrCtrs.NetIF_Loopback_ErrRxPktDiscardedCtr.

V2.10.02-007

The following net_os.c (µC/OS-II & µC/OS-III ports) functions incorrectly returned NET_TCP_ERR_NONE:

```
NetOS_Sock_RxQ_TimeoutGet_ms()
NetOS_Sock_ConnReqTimeoutGet_ms()
NetOS_Sock_ConnAcceptQ_TimeoutGet_ms()
NetOS_Sock_ConnCloseTimeoutGet_ms()
```

Fixed by returning NET_SOCK_ERR_NONE.

Version 2.10.01

N/A

Version 2.10

V2.10-001

NetTCP_RxAppData() failed to return that a TCP connection's receive queue was closed if the only remaining packet in the TCP connection's receive queue is a TCP close segment with no data. Fixed by correctly returning receive queue closed if no more TCP data is available and a TCP close segment has been received. See also 'Corrections V1.80-001'.

V2.10-002

TCP connections failed to properly and/or completely active close whenever socket blocking was disabled since the following functions did not appropriately handle return errors codes from NetTCP_RxPktConnHandlerSignalClose() that indicated that the socket connection was already closed:

```
NetTCP_RxPktConnHandlerFinWait1()
NetTCP_RxPktConnHandlerClosing()
```

Fixed by ignoring socket connection closed errors and properly handling all TCP connection closing state transitions.

V2.10-003

NetIP_TxPktValidate() failed to transmit IP datagrams from the localhost interface if the source address was not an IP localhost address. Fixed by validating any configured IP address on any enabled interface as a source address to transmit IP datagrams from the localhost interface.

V2.10-004

The following functions incorrectly freed network buffers already freed by previous network layers for certain error codes:

```
NetIP_Tx()
NetIP_ReTx()
NetICMP_Rx()
NetICMP_TxMsgErr()
NetICMP_TxMsgReq()
NetICMP_TxMsgReply()
NetIGMP_TxMsg()
NetUDP_Tx()
NetTCP_RxPktConnHandler()
NetTCP_TxPktHandler()
```

Fixed by not re-freeing network buffers already freed by previous network layers.

V2.10-005

`NetBuf_Init()` failed to initialize `NetBuf_ID_Ctr`. Fixed by initializing the counter.

V2.10-006

The following functions failed to acquire the global network lock:

```
NetARP_CacheCalcStat()
NetARP_CachePoolStatResetMaxUsed()
NetConn_PoolStatResetMaxUsed()
NetSock_PoolStatResetMaxUsed()
NetTCP_ConnPoolStatResetMaxUsed()
NetTmr_PoolStatResetMaxUsed()
```

Fixed by acquiring the global network lock. See also 'Corrections V1.92-009'.

V2.10-007

Network task stacks (µC/OS-II port) incorrectly declared as `CPU_STK_SIZE`. Fixed by declaring as `CPU_STK`.

Version 2.06

V2.06-001

`NetApp_SockConn()` failed to retry TCP socket connections using the specified timeout since `NetSock_ConnHandlerStream()` did not block or wait for connections-in-progress. Fixed by updating `NetSock_ConnHandlerStream()` to wait for TCP sockets to connect using the socket's connection request timeout even on socket connect retries. See also 'Changes V2.06-004'.

V2.06-002a

`NetSock_Sel()` incorrectly returned multiple sockets ready even for a single readied socket if that socket was included in multiple (socket) file descriptors. Fixed by returning only one socket ready for each readied socket regardless if that socket was included in multiple (socket) file descriptors. See also 'Corrections V2.05.02-007'.

V2.06-002b

NetOS_Sock_SelWait() [µC/OS-II port] failed to initialize sock_events_nbr_rdy to zero (0) if OSEventPendMulti() returned zero (0) OS events ready for errors OS_ERR_NONE or OS_ERR_ABORT. Fixed by always initializing sock_events_nbr_rdy to zero (0) when OSEventPendMulti() returns either OS_ERR_NONE or OS_ERR_ABORT errors.

V2.06-003

NetTCP_RxPktConnHandlerSeg() incorrectly omitted the default case when handling NetTCP_TxConnWinSizeHandlerCongCtrl()'s return error. Fixed by adding the default case for all error handling.

V2.06-004

NetIF_PerfMonHandler() incorrectly handled disabled network interfaces by using their invalid network interface number (NetIF_Tbl[].Nbr) to index into an network interface array and corrupt memory. Fixed by indexing into network interface arrays using valid network interface numbers only. See also 'Changes V2.06-003'.

Version 2.05.02

V2.05.02-001a

NetApp_SockAccept() failed to save and restore the initial remote socket address length required to correctly handle possible socket accept retries. Fixed by saving and restoring the initial remote socket address length for each socket accept attempt.

V2.05.02-001b

NetApp_SockRx() incorrectly restored a remote socket address length without checking the remote socket address length pointer for NULL. Fixed by checking the remote socket address length pointer for NULL before restoring the initial remote socket address length.

V2.05.02-002

NetTCP_TxConnReset() incorrectly closed TCP connections in the LISTEN state when transmitting a TCP reset packet. Fixed by transmitting a TCP reset packet from the LISTEN state without closing the TCP connection.

V2.05.02-003

TCP connection (NET_TCP_CONN) data member TxWinZeroWinTimeout_ms incorrectly declared as NET_TCP_TIMEOUT_SEC. Fixed by declaring as NET_TCP_TIMEOUT_MS.

V2.05.02-004

NetIP_IsValidAddrHost() incorrectly invalidated Class-D multicast IP addresses by comparing to NET_IP_ADDR_CLASS_C_MASK. Fixed by invalidating multicast IP addresses by comparing to NET_IP_ADDR_CLASS_D_MASK.

V2.05.02-005

NetIP_TxPktValidate() failed to set tx_remote_host for transmits to the local subnet. Fixed by setting tx_remote_host to indicate a remote host transmit on the local subnet.

V2.05.02-006

The following functions' prototype versus definition parameters were not identical:

```
NetIF_AddrHW_GetHandler()
NetIF_LinkStateSet()
NetIF_Loopback_AddrMulticastAdd()
NetIF_Loopback_AddrMulticastRemove()
NetIF_Ether_AddrMulticastAdd()
NetIF_Ether_AddrMulticastRemove()
```

Fixed by updating all function prototypes and definitions to be identical.

V2.05.02-007

`NetOS_Sock_SelWait()` [µC/OS-II port] incorrectly re-signaled a consumed OS event twice if pending on both the event AND the event's error condition. Fixed by re-signaling the consumed OS event exactly once regardless if pending on either the event OR the event's error condition. See also 'Corrections V2.06-002'.

V2.05.02-008a

The following `net_os.c` (µC/OS-II port) functions were not updated to call µC/OS-II V2.87 (and later versions) functions using `INT32U` timeouts:

```
NetOS_Tmr_Task()
NetOS_IF_TxSuspendWait()
NetOS_TCP_RxQ_Wait()
NetOS_TCP_TxQ_Wait()
NetOS_Sock_RxQ_Wait()
NetOS_Sock_ConnReqWait()
NetOS_Sock_ConnAcceptQ_Wait()
NetOS_Sock_ConnCloseWait()
NetOS_Sock_SelWait()
```

Fixed by using `INT32U` timeout values for µC/OS-II V2.87 functions (and later versions).

V2.05.02-008b

The following `net_os.c` (µC/OS-II port) functions were not updated to correctly call µC/LIB's V1.27 string format functions `Str_FmtNbr_Int32U()` and `Str_FmtNbr_Int32S()`:

```
NetOS_Dev_Init()
NetOS_IF_Init()
NetOS_TCP_Init()
NetOS_Sock_Init()
```

Fixed by updating the calls to `Str_FmtNbr_Int32U()` and `Str_FmtNbr_Int32S()`.

Version 2.05.01

N/A

Version 2.05

V2.05-001

The following receive/transmit validation functions incorrectly omitted the default case when validating the receive/transmit network buffer type:

```
NetIF_Ether_RxPktValidateBuf()
NetARP_RxPktValidateBuf()
NetIP_RxPktValidateBuf()
NetICMP_RxPktValidateBuf()
NetUDP_RxPktValidateBuf()
NetTCP_RxPktValidateBuf()
NetSock_RxPktValidateBuf()
NetIF_TxPktValidate()
NetIF_Loopback_TxPktValidate()
NetIF_Ether_TxPktValidate()
NetIP_TxPktValidate()
NetUDP_TxPktValidate()
NetTCP_TxPktValidate()
```

Fixed by adding the default case for receive/transmit network buffer type validation.

V2.05-002

`NetIP_RxPktValidateOpt()` incorrectly invalidated and discarded IP packets with unknown IP options. Fixed by silently ignoring any unknown IP options received without discarding the IP packet.

Version 2.04

V2.04-001a

TCP connections occasionally failed to properly and/or completely close, while also occasionally transmitting a spurious TCP reset (RST) packet, because TCP connection state transitions in `NetTCP_TxConnReqClose()` were not updated prior to transmitting the TCP close (FIN) segment. Fixed by modifying `NetTCP_TxConnReqClose()` to update TCP connection states prior to transmitting the TCP close segment. See also 'Corrections V1.92-004'.

V2.04-001b

TCP connections occasionally failed to properly and/or completely close, while also occasionally transmitting a spurious TCP reset (RST) packet, because TCP connection state transitions to suspend the transmit queue in `NetTCP_TxConnTxQ()` did not appropriately handle all transmit queue closing states. Fixed by adding closing suspend states to properly handle all TCP connection transmit queue closing state transitions.

V2.04-002

The following functions failed to compile if `NULL` statements were macro-substituted in `CPU_SR_ALLOC()` if not declared after all other local variables are declared:

```
NetARP_CacheCalcStat()
NetOS_Lock()      (µC/OS-II port)
```

Fixed by requiring all functions to declare `CPU_SR_ALLOC()` only after all other local variables have been declared.

Version 2.03

V2.03-001

NetDbg_ChkStatusHandlerRsrcLost() incorrectly checked for lost network buffers even when NET_CFG_CTR_ERR_EN is disabled. Fixed by checking for lost network buffers only when NET_CFG_CTR_ERR_EN is enabled.

V2.03-002

Removed superfluous '+1' offset in NET_OS_TIMEOUT_MAX_uS's preprocessor conditional in net_os.h (µC/OS-II port).

Version 2.02

V2.02-001

Removed NetIF_MTU_Get()'s superfluous mtu parameter.

V2.02-002

NetTCP_TxConnReTxQ() incorrectly re-transmitted, as is, TCP segments with data that had been moved. Fixed by re-transmitting, as is, only TCP segments whose controls AND data have not been modified.

V2.02-003

Starting in V2.00, NetIP_TxPktValidate() and NetIP_TxPktDatagramRouteSel() failed to transmit IP datagrams to localhost or hosts on the local network from any interface with NO default gateway configured. Fixed by always transmitting IP datagrams to localhost or local-network hosts even if no default gateway is configured for the interface.

V2.02-004

NetIP_Tx() failed to indicate when an IP packet was not prepared due to invalid IP or network parameters. This allowed potentially unprepared IP packets to be queued onto TCP connections' re-transmit queues. Fixed by returning an IP transmit packet error (NET_IP_ERR_TX_PKT) whenever an IP packet is not properly prepared.

V2.02-005

NetIP_TxPktValidate() failed to invalidate IP transmit TTL (Time-To-Live) for value of zero. Fixed by invalidating IP transmit TTL value of zero.

V2.02-006

Starting in V2.00, NetBuf_Get() incorrectly eliminated setting buffers' data area cleared flag (NET_BUF_FLAG_CLR_MEM) when NET_DBG_CFG_MEM_CLR_EN is enabled. Fixed by restoring the setting of buffers' data area cleared flag when necessary.

Version 2.01

V2.01-001

NetTCP_TxConnReTxQ_Timeout() incorrectly invalidated TCP connections' transmit queue state when in any TCP connection synchronization states (SYN-SENT or SYN-RECEIVED), thereby preventing packets to be retransmitted. Fixed by correctly validating TCP connections' transmit queue state when in any synchronization state thereby permitting packets to be retransmitted.

V2.01-002

The following network timer callback functions failed to properly clear a network timer pointer for certain TCP connection states:

```
NetTCP_TxConnWinSizeZeroWinTimeout()
NetTCP_TxConnAckDlyTimeout()
NetTCP_TxConnTxQ_TimeoutIdle()
NetTCP_TxConnTxQ_TimeoutSillyWin()
NetTCP_TxConnReTxQ_Timeout()
```

Fixed by clearing the network timer pointer even for invalid TCP connection states. See also 'Corrections V2.10.02-002'.

V2.01-003

Former `NetTCP_TxConnRTT_GetTS_ms()` [see 'Changes V2.01-004'] incorrectly calculated millisecond timestamps in `net_bsp.c` (µC/OS-II port). Fixed by designing and implementing new timestamp calculation.

V2.01-004

`NetIP_IsValidAddrHostCfgd()` did not correctly validate IP host addresses for invalid 'This Host' or broadcast subnetted addresses. Fixed by invalidating 'This Host' or broadcast subnet addresses.

Version 2.00

N/A

Version 1.92

V1.92-001

Configuration API functions failed to exclusively access configuration variables. Fixed by exclusively accessing configuration variables in critical sections or by acquiring the global network lock.

V1.92-002

`NetSock_BindHandler()` failed to allow binding to a random port number when the local address argument's port number is '0'. Fixed by allowing binding to random port numbers. See also 'Improvements V1.92-003'.

V1.92-003

`NetSock_BindHandler()` failed to free possible random port numbers from a previous connection address when re-binding (UDP) sockets to a new connection address. Fixed by freeing possible random port numbers when re-binding (UDP) sockets.

V1.92-004

TCP connections requested to and from the same target failed to connect because TCP connection state transitions in `NetTCP_TxConnReq()` and `NetTCP_RxPktConnHandlerListen()` were not updated prior to completing the TCP three-way connection handshake. Fixed by modifying `NetTCP_TxConnReq()` and `NetTCP_RxPktConnHandlerListen()` to update their TCP connection states prior to completing the TCP three-way connection handshake. See also 'Corrections V2.04-001a'.

V1.92-005

`NetUDP_RxAppData()` incorrectly returned an invalid application receive data buffer size error (`NET_UDP_ERR_INVALID_DATA_SIZE`) even if the application receive data buffer was exactly large enough to receive all available data. Fixed by returning an invalid application receive data buffer size error only if the application receive data buffer is not large enough to receive all available data.

V1.92-006

`NetIP_RxPktValidate()` and `NetIP_TxPktValidate()` incorrectly invalidated all IP datagrams with localhost source addresses. Fixed by allowing localhost source addresses for IP datagrams received and transmitted entirely internal to the host.

V1.92-007

`NetBuf_FreeBufQ_SecList()` failed to clear network buffers' secondary list pointers when the appropriate unlink function is freed. Fixed by clearing network buffers' secondary list pointers.

V1.92-008

`NetDbg_ChkStatusHandlerConns()` failed to initialize or use local variable `err_conn`. Fixed by removing `err_conn`.

V1.92-009

The following functions failed to acquire the global network lock:

```
NetARP_IsAddrProtocolConflict()
NetARP_CacheGetAddrHW()
NetARP_ProbeAddrOnNet()
NetARP_TxReqGratuitous()
NetICMP_TxMsgReq()
NetUDP_TxAppData()
```

Fixed by acquiring the global network lock. See also 'Corrections V2.10-006'.

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 connections' close segments with TCP data by occasionally discarding a single octet of the received TCP data. Fixed by correctly handling all received TCP data and 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 and 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 and 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()` and `NetConn_CloseFromTransport()` failed to check for previously-closed network connections following any other network connection close operations. Fixed by checking for and 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 and 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 adjust the remote receive window update by the TCP connection's recently transmitted sequences. Fixed by adjusting 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 and 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 and 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 and 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 V2.10-001 and 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.

V1.71-006a

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

V1.71-006b

`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 2.11.00

Version 2.10.04

Version 2.10.03

Version 2.10.02

Version 2.10.01

Version 2.10

Version 2.06

Version 2.05.02

Version 2.05.01

Version 2.05

Version 2.04

Version 2.03

Version 2.02

V2.01-001 (Unresolved)
V1.73-001 (Unresolved)
V1.60-002 (Unresolved)

Version 2.01

V2.01-001
Currently only the default network interface can be configured with a dynamic IP address.

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

Version 2.00

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

Version 1.92

Version 1.91

Version 1.90

Version 1.89

Version 1.88

Version 1.87

Version 1.86

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

Version 1.82

Version 1.81

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

Version 1.71

Version 1.70

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 and fault connection closes, close correctly (see also 'Improvements V1.84-001')

V1.60-002

IP route and timestamp options incorrectly implemented.

Limitations

001

Following IP features NOT supported:

- IP forwarding/routing
- IP transmit fragmentation
- IP Security options
- ICMP Address Mask Agent/Server

002

Following TCP features NOT supported:

- TCP Urgent Data
- TCP Security and Precedence
- TCP Multihoming

003

Following socket features NOT supported:

- Socket `shutdown()`
- Multiple sockets bound to same socket address or socket pair

Contacts

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326
USA

Phone: +1 954 217 2036
Fax: +1 954 217 2037
E-mail: Licensing@Micrium.com
Web: www.Micrium.com