1 Migrating from CycloneTCP 1.8.6 to 1.9.0

1.1 Background

This upgrade notice only concerns users that make use of FTP client (major changes), FTP server (minor change) and HTTP server (minor change) modules.

1.2 FTP Client

The architecture of the FTP client has been reworked to allow RTOS-less operation. The FTP client module now depends on the following files:

- cyclone_tcp/ftp/ftp_client.c
- cyclone_tcp/ftp/ftp_client.h
- cyclone_tcp/ftp/ftp_client_transport.c (new file)
- cyclone_tcp/ftp/ftp_client_transport.h (new file)
- cyclone_tcp/ftp/ftp_client_misc.c (new file)
- cyclone_tcp/ftp/ftp_client_misc.h (new file)

The FTP client has been reworked to match the API model of other CycloneTCP modules. The following functions are deprecated. It is highly recommended to switch to the new API instead. The legacy API is still available to maintain compatibility with existing projects:

Deprecated API	New API
-	ftpClientInit
ftpRegisterTlsInitCallback	ftpClientRegisterTlsInitCallback
-	ftpClientSetTimeout
-	ftpClientBindToInterface(FtpClientContext *context, NetInterface *interface);
error_t ftpConnect(FtpClientContext *context, NetInterface *interface, const IpAddr *serverIpAddr, uint16_t serverPort, uint_t mode)	ftpClientConnect(FtpClientContext *context, const IpAddr *serverIpAddr, uint16_t serverPort, uint_t mode)
ftpLogin(FtpClientContext *context, const char_t *username, const char_t *password, const char_t *account)	ftpClientLogin(FtpClientContext *context, const char_t *username, const char_t *password)
ftpGetWorkingDir	ftpClientChangeWorkingDir
ftpChangeWorkingDir	ftpClientChangeToParentDir
-	ftpClientOpenDir
-	ftpClientReadDir
-	ftpClientCloseDir
ftpMakeDir	ftpClientMakeDir
ftpRemoveDir	ftpClientRemoveDir
ftpMakeDir	ftpClientMakeDir
ftpRemoveDir	ftpClientRemoveDir
ftpOpenFile	ftpClientOpenFile
ftpWriteFile(FtpClientContext *context, const void *data, size_t length, uint_t flags)	ftpClientWriteFile(FtpClientContext *context, const void *data, size_t length, size_t *written, uint_t flags)
ftpReadFile	ftpClientReadFile
ftpCloseFile	ftpClientCloseFile

Deprecated API (continued)	New API (continued)
ftpRenameFile	ftpClientRenameFile
ftpDeleteFile	ftpClientDeleteFile
ftpReadFile	ftpClientReadFile
ftpCloseFile	ftpClientCloseFile
ftpRenameFile	ftpClientRenameFile
ftpDeleteFile	ftpClientDeleteFile
-	ftpClientDisconnect
ftpClose	ftpClientClose
-	ftpClientDeinit

Remarks:

- The *ftpClientInit* API must be called before calling any other FTP-related functions.
- The deprecated *ftpConnect* API allows the user to specify a given network interface. With the new API, you must call *ftpClientBindToInterface* before connecting (using the *ftpClientConnect* API).
- The deprecated *ftpLogin* API allowed the user to specify an account name. The new *ftpLogin* API drops support for account name. If an account name is required by the remote FTP server, then you may consider the *ftpClientLoginEx* API instead.
- The new "written" parameter of the *ftpClientWrite* API is optional. NULL parameter can be passed to the function.
- The *ftpClientDisconnect* API may be called to gracefully disconnect from the FTP server. If you want to abort the FTP connection, you may call *ftpClientClose* instead.
- The *ftpClientDeinit* API must be called before releasing the FTP client context. If the FTP connection has not been closed previously (using either *ftpClientDisconnect* or *ftpClientClose*), then then FTP connection is automatically aborted.

1.3 FTP Server

The FTP_SERVER_PASV_HOOK macro is deprecated. If the server is behind a NAT router, then you may specify the external IP address by using the new publicIpv4Addr setting:

```
//Get default settings
ftpServerGetDefaultSettings(&ftpServerSettings);
//Bind FTP server to the desired interface
ftpServerSettings.interface = &netInterface[0];
//Listen to port 21
ftpServerSettings.port = FTP_PORT;
//Root directory
strcpy(ftpServerSettings.rootDir, "/");
//User verification callback function
ftpServerSettings.checkUserCallback = ftpServerCheckUserCallback;
//Password verification callback function
ftpServerSettings.checkPasswordCallback = ftpServerCheckPasswordCallback;
//Callback used to retrieve file permissions
ftpServerSettings.getFilePermCallback = ftpServerGetFilePermCallback;
//Set external IP address
ipv4StringToAddr("1.2.3.4", &ftpServerSettings.publicIpv4Addr);

//FTP server initialization
error = ftpServerInit(&ftpServerContext, &ftpServerSettings);
//Failed to initialize FTP server?
if(error)
{
   ...
}
```

1.4 HTTP Server

Support for gzip-compressed resources is disabled by default. If you want to enable this feature, then you must add the following line to your net_config.h configuration file:

```
//Gzip content type support
#define HTTP_SERVER_GZIP_TYPE_SUPPORT ENABLED
```

2 Migrating from CycloneTCP 1.8.2 to 1.8.6

No changes required. Version 1.8.6 can be used as drop-in replacement

3 Migrating from CycloneTCP 1.8.0 to 1.8.2

No changes required. Version 1.8.2 can be used as drop-in replacement

4 Migrating from CycloneTCP 1.7.8 to 1.8.0

4.1 Drivers

In CycloneTCP 1.8.0, the "cyclone_tcp/drivers" folder has been reorganized. Drivers are now sorted by type:

- cyclone_tcp/drivers/mac: Ethernet MAC drivers
- cyclone_tcp/drivers/phy: Ethernet PHY drivers
- cyclone_tcp/drivers/eth: External Ethernet controllers
- cyclone_tcp/drivers/wifi: Wi-Fi wrappers

You may keep your existing driver in the cyclone_tcp/drivers directory since 1.7.8 and 1.8.0 versions share compatible drivers. But it is recommended to switch to the new file structure. Note that all drivers now end with the "_driver" suffix:

Old filenames (version 1.7.8)	New filenames (version 1.8.0)
cyclone_tcp/drivers/stm32f2x7_eth.c	cyclone_tcp/drivers/mac/stm32f2x7_eth_driver.c
cyclone_tcp/drivers/stm32f4x7_eth.c	cyclone_tcp/drivers/mac/stm32f4x7_eth_driver.c
cyclone_tcp/drivers/stm32f4x9_eth.c	cyclone_tcp/drivers/mac/stm32f4x9_eth_driver.c
cyclone_tcp/drivers/stm32f7xx_eth.c	cyclone_tcp/drivers/mac/stm32f4xx_eth_driver.c
…	…
cyclone_tcp/drivers/sam3x_eth.c	cyclone_tcp/drivers/mac/sam3x_eth_driver.c
cyclone_tcp/drivers/sam4e_eth.c	cyclone_tcp/drivers/mac/sam4e_eth_driver.c
cyclone_tcp/drivers/same70_eth.c	cyclone_tcp/drivers/mac/same70_eth_driver.c
…	…
cyclone_tcp/drivers/pic32mx_eth.c	cyclone_tcp/drivers/mac/pic32mx_eth_driver.c
cyclone_tcp/drivers/pic32mz_eth.c	cyclone_tcp/drivers/mac/pic32mz_eth_driver.c
…	…
drivers/dp83848.c	cyclone_tcp/drivers/phy/dp83848_driver.c
…	…
drivers/ksz8051.c	cyclone_tcp/drivers/phy/ksz8051_driver.c
drivers/ksz8081.c	cyclone_tcp/drivers/phy/ksz8081_driver.c
…	…
drivers/lan8720.c	cyclone_tcp/drivers/phy/lan8720_driver.c
drivers/lan8740.c	cyclone_tcp/drivers/phy/lan8740_driver.c
drivers/lan8742.c	cyclone_tcp/drivers/phy/lan8742_driver.c
…	…
drivers/wilc1000_driver.c	cyclone_tcp/drivers/wifi/wilc1000_driver.c
drivers/winc1500_driver.c	cyclone_tcp/drivers/wifi/winc1500_driver.c

It will impact the corresponding #include directives in your code. For example:

```
#include "drivers/mac/stm32f2x7_eth_driver.h"
#include "drivers/phy/lan8742_driver.h"
```

4.2 SNMP Agent

The SNMP agent supports remote management of users (USM) and access rights (VACM). As a consequence, the structure of the module has changed. The SNMP agent now depends on the following files:

- cyclone_tcp/snmp/snmp_agent.c
- cyclone_tcp/snmp/snmp_agent_dispatch.c
- cyclone_tcp/snmp/snmp_agent_message.c (new file)
- cyclone_tcp/snmp/snmp_agent_pdu.c
- cyclone_tcp/snmp/snmp_agent_object. c (new file)
- cyclone_tcp/snmp/snmp_agent_trap. c (new file)
- cyclone_tcp/snmp/snmp_agent_inform. c (new file)
- cyclone_tcp/snmp/snmp_agent_usm. c (new file)
- cyclone_tcp/snmp/snmp_agent_vacm. c (new file)
- cyclone_tcp/snmp/snmp_agent_misc.c

The following file are deprecated and must be removed from your project:

- cyclone_tcp/snmp/snmp_common.c (to be removed)
- cyclone_tcp/snmp/snmp_usm. c (to be removed)
- cyclone_tcp/snmp/snmp_usm.h (to be removed)

4.3 Crypto library

The “cyclone_crypto” folder has been reorganized. If your application makes use of cryptographic algorithms (SNMP agent, WebSockets, HTTP with Basic/Digest authentication, etc), then you must modify your project or makefile in order to reflect the following changes:

Old filenames (version 1.7.8)	New filenames (version 1.8.0)
cyclone_crypto/crypto.h	cyclone_crypto/core/crypto.h
cyclone_crypto/des.c cyclone_crypto/des.h	cyclone_crypto/cipher/des.c cyclone_crypto/cipher/des.h
cyclone_crypto/aes.c cyclone_crypto/aes.h	cyclone_crypto/cipher/aes.c cyclone_crypto/cipher/aes.h
cyclone_crypto/cipher_mode_cbc.c cyclone_crypto/cipher_mode_cbc.h	cyclone_crypto/cipher_mode/cbc.c cyclone_crypto/cipher_mode/cbc.h
cyclone_crypto/cipher_mode_cfb.c cyclone_crypto/cipher_mode_cfb.h	cyclone_crypto/cipher_mode/cfb.c cyclone_crypto/cipher_mode/cfb.h
cyclone_crypto/md5.c cyclone_crypto/md5.h	cyclone_crypto/hash/md5.c cyclone_crypto/hash/md5.h
cyclone_crypto/sha1.c cyclone_crypto/sha1.h	cyclone_crypto/hash/sha1.c cyclone_crypto/hash/sha1.h
cyclone_crypto/hmac.c cyclone_crypto/hmac.h	cyclone_crypto/mac/hmac.c cyclone_crypto/mac/hmac.h
cyclone_crypto/base64.c cyclone_crypto/base64.h	cyclone_crypto/encoding/base64.c cyclone_crypto/encoding/base64.h
cyclone_crypto/asn1.c cyclone_crypto/asn1.h	cyclone_crypto/encoding/asn1.c cyclone_crypto/encoding/asn1.h
cyclone_crypto/oid.c cyclone_crypto/oid.h	cyclone_crypto/encoding/oid.c cyclone_crypto/encoding/oid.h

5 Migrating from CycloneTCP 1.7.6 to 1.7.8

5.1 Background

The following changes only concern **TFTP server** and **SNMP agent.**

5.2 TFTP Server

The structure of the TFTP server has changed. The TFTP server now depends on the following files:

- cyclone_tcp/tftp/tftp_server.c
- cyclone_tcp/tftp/tftp_server.h
- cyclone_tcp/tftp/tftp_server_misc.c (new file)
- cyclone_tcp/tftp/tftp_server_misc.h (new file)
- cyclone_tcp/tftp/tftp_common.h

5.3 SNMP Agent

New MIBs have been implemented in the mibs/ directory:

- cyclone_tcp/mibs/if_mib_module.c
- cyclone_tcp/mibs/if_mib_module.h
- cyclone_tcp/mibs/if_mib_impl.c
- cyclone_tcp/mibs/if_mib_impl.h
- cyclone_tcp/mibs/ip_mib_module.c
- cyclone_tcp/mibs/ip_mib_module.h
- cyclone_tcp/mibs/ip_mib_impl.c
- cyclone_tcp/mibs/ip_mib_impl.h
- cyclone_tcp/mibs/tcp_mib_module.c
- cyclone_tcp/mibs/tcp_mib_module.h
- cyclone_tcp/mibs/tcp_mib_impl.c
- cyclone_tcp/mibs/tcp_mib_impl.h
- cyclone_tcp/mibs/udp_mib_module.c
- cyclone_tcp/mibs/udp_mib_module.h
- cyclone_tcp/mibs/udp_mib_impl.c
- cyclone_tcp/mibs/udp_mib_impl.h
- cyclone_tcp/mibs/snmp_mib_module.c
- cyclone_tcp/mibs/snmp_mib_module.h
- cyclone_tcp/mibs/snmp_mib_impl.c
- cyclone_tcp/mibs/snmp_mib_impl.h
- cyclone_tcp/mibs/snmp_usm_mib_module.c
- cyclone_tcp/mibs/snmp_usm_mib_module.h
- cyclone_tcp/mibs/snmp_usm_mib_impl.c
- cyclone_tcp/mibs/snmp_usm_mib_impl.h

Callback functions responsible for setting objects now have an extra *commit* parameter. Write operations are always performed in 2 passes:

- In the first pass the commit parameter is set to FALSE. The function only verify that the write request is acceptable. It must not update the database with the new value.
- If the whole SNMP set request is acceptable, then the function is called again (second pass). The *commit* parameter is set to TRUE and the value can be safely written to the database.

```
error_t privateMibSetLedEntry(const MibObject *object,
   const uint8_t *oid, size_t oidLen, const MibVariant *value,
   size_t valueLen, bool_t commit);
```

New fields have been added to the MibModule structure:

- name: Short name describing the MIB
- oid: Identifier of the MIB
- oidLen: Length of the MIB identifier
- load: Callback function invoked when the MIB is loaded by the SNMP agent. Can be NULL if not implemented.
- unload: Callback function invoked when the MIB is unloaded by the SNMP agent. Can be NULL if not implemented.

```
/**
 * @brief Private MIB module
 **/

const MibModule privateMibModule =
{
   "Private MIB",             //MIB short name
   {0},                       //MIB object identifier
   1,                         //Length of the MIB object identifier
   privateMibObjects,
   arraysize(privateMibObjects),
   privateMibInit,
   NULL,                      //MIB load callback
   NULL,                      //MIB unload callback
   privateMibLock,
   privateMibUnlock
};
```

6 Migrating from CycloneTCP 1.7.4 to 1.7.6

6.1 File name conflict with some GCC toolchains

It has been reported that the header file endian.h conflicts with some GCC toolchains. The following files must be removed from the project:

- common/endian.c
- common/endian.h

The following files must be added to the project in replacement of endian.c and .h:

- common/cpu_endian.c
- common/cpu_endian.h

For big-endian MCUs (AVR32, Coldfire V2), the preprocessor macro _BIG_ENDIAN must be removed from C project properties. _CPU_BIG_ENDIAN must be used as replacement in C project properties. This modification does not concern little-endian MCUs (Cortex-M3/4/7, Cortex-A5/8/9, ARM7, ARM9, PIC32, RX600)

6.2 HTTP Server

This change only concerns users that make use of HTTP digest authentication feature. A callback function is now used to generate random data:

```
   //Pseudo-random number generator
   //httpServerSettings.prngAlgo = YARROW_PRNG_ALGO;  <= Deprecated
   //httpServerSettings.prngContext = &yarrowContext; <= Deprecated

   //Callback function
   httpServerSettings.randCallback = httpServerRandCallback;
```

Here is an example of implementation for this callback function :

```
/**
 * @brief Random data generation callback function
 * @param[out] data Buffer where to store the random data
 * @param[in] lenght Number of random bytes to be generated
 * @return Error code
 **/

error_t httpServerRandCallback(uint8_t *data, size_t length)
{
   //Generate some random data
   return yarrowRead(&yarrowContext, data, length);
}
```

6.3 WebSockets

The following files must be added to the project:

- cyclone_tcp/web_socket/ web_socket_frame.c
- cyclone_tcp/web_socket/ web_socket_frame.h
- cyclone_tcp/web_socket/ web_socket_transport.c
- cyclone_tcp/web_socket/ web_socket_transport.h

6.4 MQTT Client

The MQTT client now allows RTOS-less operation. The file structure of the MQTT client has changed. MQTT depends on the following files:

- cyclone_tcp/mqtt/mqtt_client.c
- cyclone_tcp/mqtt/mqtt_client.h
- cyclone_tcp/mqtt/mqtt_client_misc.c
- cyclone_tcp/mqtt/mqtt_client_misc.h
- cyclone_tcp/mqtt/mqtt_client_packet.c <= New file
- cyclone_tcp/mqtt/mqtt_client_packet.c <= New file
- cyclone_tcp/mqtt/mqtt_client_transport.c
- cyclone_tcp/mqtt/mqtt_client_transport.h
- cyclone_tcp/mqtt/mqtt_common.h

The following file must be removed from the project:

- cyclone_tcp/mqtt/mqtt_common.c <= Removed file

Connecting to a remote MQTT server only requires a call to mqttClientConnect(). The function mqttClientOpen() is deprecated. It is not necessary to call this function anymore.

Note the new prototypes of the following functions:

```
error_t mqttClientConnect(MqttClientContext *context,
   const IpAddr *serverIpAddr, uint16_t serverPort, bool_t cleanSession);

error_t mqttClientSubscribe(MqttClientContext *context,
   const char_t *topic, MqttQosLevel qos, uint16_t *packetId);

error_t mqttClientUnsubscribe(MqttClientContext *context,
   const char_t *topic, uint16_t *packetId)

error_t mqttClientPublish(MqttClientContext *context,
   const char_t *topic, const void *message, size_t length,
   MqttQosLevel qos, bool_t retain, uint16_t *packetId);
```

When an RTOS is used, the additional "packetId" parameter can be set to NULL for mqttClientSubscribe(), mqttClientUnsubscribe() or mqttClientPublish().

Note the new prototype of the publish callback function:

```
void MqttClientPublishCallback(MqttClientContext *context,
   const char_t *topic, const uint8_t *message, size_t length,
   bool_t dup, MqttQosLevel qos, bool_t retain, uint16_t packetId)
{
   ...
}
```

Callback functions (such as publish or TLS initialization) must be registered using the mqttClientRegisterCallbacks() function:

```
   //Initialize MQTT client context
   mqttClientInit(&mqttClientContext);
   ...

   //Initialize MQTT client callbacks
   mqttClientInitCallbacks(&mqttClientCallbacks);

   //Attach application-specific callback functions
   mqttClientCallbacks.publishCallback = mqttPublishCallback;
   mqttClientCallbacks.tlsInitCallback = mqttTlsInitCallback;
   ...

   //Register MQTT client callbacks
   mqttClientRegisterCallbacks(&mqttClientContext, &mqttClientCallbacks);
```

7 Migrating from CycloneTCP 1.7.2 to 1.7.4

The following changes only concern users that make use of the **Kinetis K60/K64/K70 MAC Ethernet driver.**

7.1 Kinetis K60/64 MAC Ethernet driver

CycloneTCP now supports Kinetis K65 and K66 series. All K6x devices (K60, K64, K65 and K65) now share a common driver implementation (mk6x_eth_driver.c)

Older driver files (mk60_eth_driver.c and mk70_eth_driver.c) must be replaced by the new mk6x_eth_driver.c driver file.

All references to mk60EthDriver and mk64EthDriver must be changed to mk6xEthDriver:

```
//Dependencies
#include "drivers/mk6x_eth.h"

...
//Select the relevant MAC Ethernet driver
netSetDriver(interface, &mk6xEthDriver);
...
```

7.2 Kinetis K70 MAC Ethernet driver

For coherency purposes, mk70_eth_driver.c has been deprecated and replaced by by the mk7x_eth_driver.c driver file.

All references to mk70EthDriver must be changed to mk7xEthDriver:

```
//Dependencies
#include "drivers/mk7x_eth.h"

...
//Select the relevant MAC Ethernet driver
netSetDriver(interface, &mk7xEthDriver);
...
```

8 Migrating from CycloneTCP 1.7.0 to 1.7.2

8.1 Background

The following changes only concern users that make use of **SNMP agent.**

8.2 Project

The TCP/IP has been modified to support version 3 of SNMP. Moreover, the descriptions of the MIB bases have moved from cyclone_tcp/snmp to cyclone_tcp/mibs. As a consequence, the following files must be deleted from the project tree:

- cyclone_tcp/snmp/mib_common.h
- cyclone_tcp/snmp/mib2_module.c
- cyclone_tcp/snmp/mib2_module.h
- cyclone_tcp/snmp/mib2_impl.c
- cyclone_tcp/snmp/mib2_impl.h

The following files must be added to the project tree:

- cyclone_tcp/mibs/mib_common.c (new file)
- cyclone_tcp/ mibs /mib_common.h
- cyclone_tcp/ mibs /mib2_module.c
- cyclone_tcp/ mibs /mib2_module.h
- cyclone_tcp/ mibs /mib2_impl.c
- cyclone_tcp/ mibs /mib2_impl.h
- cyclone_tcp/snmp/snmp_dispatch.c (new file)
- cyclone_tcp/snmp/snmp_dispatch.h (new file)
- cyclone_tcp/snmp/snmp_usm.c (new file)
- cyclone_tcp/snmp/snmp_usm.h (new file)

If only version v1 and/or v2c of SNMP are needed, there is no need to add any other files. If SNMPv3 needs to be implemented, then it necessary to add the following crypto algorithms to the project:

- cyclone_crypto/md5.c
- cyclone_crypto/md5.h
- cyclone_crypto/sha1.c
- cyclone_crypto/sha1.h
- cyclone_crypto/hmac.c
- cyclone_crypto/hmac.h
- cyclone_crypto/des.c
- cyclone_crypto/des.h
- cyclone_crypto/aes.c
- cyclone_crypto/aes.c
- cyclone_crypto/cipher_mode_cbc.c
- cyclone_crypto/cipher_mode_cbc.h
- cyclone_crypto/cipher_mode_cfb.c
- cyclone_crypto/cipher_mode_cfb.h

8.3 SNMP Agent initialization

- SNMPv1, SNMPv2c and SNMPv3 can now coexist. The fields *versionMin* and *versionMax* of the *SnmpAgentSettings* structure define the minimum and the maximum acceptable versions for this SNMP agent instance.
- The enterprise OID can be dynamically adjusted using a specific function named *snmpAgentSetEnterpriseOid*. The *enterpriseOid* field of the *SnmpAgentSettings* structure is deprecated.
- Communities can be dynamically created/deleted using *snmpAgentCreateCommunity* and *snmpAgentDeleteCommunity* functions. The *readOnlyCommunity* and *readWriteCommunity* fields of the structure *SnmpAgentSettings* are deprecated.

```
#include "snmp/snmp_agent.h"
#include "mibs/mib2_module.h"
#include "mibs/mib2_impl.h"
#include "oid.h"

   //Get default settings
   snmpAgentGetDefaultSettings(&snmpAgentSettings);
   //Bind SNMP agent to the desired network interface
   snmpAgentSettings.interface = interface;
   //Minimum version accepted by the SNMP agent
   snmpAgentSettings.versionMin = SNMP_VERSION_1;
   //Maximum version accepted by the SNMP agent
   snmpAgentSettings.versionMax = SNMP_VERSION_2C;

   //SNMP agent initialization
   error = snmpAgentInit(&snmpAgentContext, &snmpAgentSettings);
   //Failed to initialize SNMP agent?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to initialize SNMP agent!\r\n");
   }

   //Convert enterprise OID from string representation
   oidFromString("1.3.6.1.4.1.8072.9999.9999", oid, sizeof(oid), &oidLen);
   //Set enterprise OID
   snmpAgentSetEnterpriseOid(&snmpAgentContext, oid, oidLen);

   //Set read-only community string
   error = snmpAgentCreateCommunity(&snmpAgentContext, "public",
      SNMP_ACCESS_READ_ONLY);

   //Set read-write community string
   snmpAgentCreateCommunity(&snmpAgentContext, "private",
      SNMP_ACCESS_READ_WRITE);

   //Load standard MIB-II
   snmpAgentLoadMib(&snmpAgentContext, &mib2Module);
```

8.4 Sending SNMP trap messages

In order to send SNMP traps, the *snmpAgentSendTrap* function must be called. This new function enables users to specify:

- The destination IP address
- The SNMP version to be used
- The community string (or the user name for SNMPv3) to be used
- Generic trap type
- Specific trap code
- The list of objects to be included in the SNMP trap message

```
#include "snmp/snmp_agent.h"
#include "oid.h"

   error_t error;
   IpAddr destIpAddr;
   SnmpTrapObject trapObjects[2];

   //Destination IP address
   ipStringToAddr'"192.168.0.100", &destIpAddr);

   //Add the ifDescr.1 object to the variable binding list
   oidFromString("1.3.6.1.2.1.2.2.1.2.1", trapObjects[0].oid,
      SNMP_MAX_OID_SIZE, &trapObjects[0].oidLen);

   //Add the ifPhysAddress.1 object to the variable binding list
   oidFromString("1.3.6.1.2.1.2.2.1.6.1", trapObjects[1].oid,
      SNMP_MAX_OID_SIZE, &trapObjects[1].oidLen);

   //Send a SNMP trap
   error = snmpAgentSendTrap(&snmpAgentContext, &destIpAddr,
      SNMP_VERSION_2C, "public", SNMP_TRAP_LINK_UP, 0, trapObjects, 2);

   //Failed to send trap message?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to send SNMP trap message!\r\n");
   }
```

The *snmpSendTrap* function is deprecated and cannot be used anymore.

9 Migrating from CycloneTCP 1.6.4 to 1.7.0

9.1 Background

New IPv6 features have been added to comply with TAHI Phase-2 Conformance Test Suite (Core Protocols):

- Support for multiple global IPv6 addresses
- Support for multiple prefixes
- Support for multiple Default Routers
- Support for Path MTU Discovery
- Support for ICMPv6 Redirect messages
- SLAAC is now fully compliant with RFCs

Previous CycloneTCP releases make use of 2 tasks to manage TCP/IP events. The task model of the TCP/IP task has been reworked to allow a single task. Main advantages are:

- Less RAM memory usage
- Simpler task model that eases software certification in safety environments

mDNS and DNS-SD modules now comply with BCT (Bonjour Conformance Test)

9.2 Task model

Since the task model has changed, the following configuration properties are now deprecated. You will expect compiler warnings if these properties are still present in the net_config.h header :

```
NET_TICK_STACK_SIZE
NET_TICK_PRIORITY
NET_RX_STACK_SIZE
NET_RX_PRIORITY
```

It is recommended that the new properties that defines the size of the stack and the priority for the unique task should be taken from NET_RX_STACK_SIZE and NET_RX_PRIORITY.

The name for the new properties are :

```
NET_TASK_STACK_SIZE
NET_TASK_PRIORITY
```

9.3 mDNS and DNS-SD API

mDNS and DNS-SD are now separate services that are disabled by default. To enable the mDNS responder, add the following line in net_config.h:

```
//mDNS responder support
#define MDNS_RESPONDER_SUPPORT ENABLED
```

Then, the mDNS responder must be initialized as follows:

```
//Global variable
MdnsResponderContext mdnsResponderContext;

   //Local variable
   MdnsResponderSettings mdnsResponderSettings;

   //Get default settings
   mdnsResponderGetDefaultSettings(&mdnsResponderSettings);
   //Underlying network interface
   mdnsResponderSettings.interface = &netInterface[0];

   //mDNS responder initialization
   error = mdnsResponderInit(&mdnsResponderContext,
      &mdnsResponderSettings);
   //Failed to initialize mDNS responder?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to initialize mDNS responder!\r\n");
   }

   //Set mDNS hostname
   error = mdnsResponderSetHostname(&mdnsResponderContext,
      "TestHostName");
   //Any error to report?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to set hostname!\r\n");
   }

   //Start mDNS responder
   error = mdnsResponderStart(&mdnsResponderContext);
   //Failed to start mDNS responder?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to start mDNS responder!\r\n");
   }
```

To enabled DNS-SD, add the following line in net_config.h:

```
//DNS-SD support
#define DNS_SD_SUPPORT ENABLED
```

Then, DNS-SD must be initialized as follows:

```
//Global variable
DnsSdContext dnsSdContext;

   //Local variable
   DnsSdSettings dnsSdSettings;

   //Get default settings
   dnsSdGetDefaultSettings(&dnsSdSettings);
   //Underlying network interface
   dnsSdSettings.interface = &netInterface[0];

   //DNS-SD initialization
   error = dnsSdInit(&dnsSdContext, &dnsSdSettings);
   //Failed to initialize DNS-SD?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to initialize DNS-SD!\r\n");
   }

   //Set service instance name
   error = dnsSdSetInstanceName(&dnsSdContext, "Test Service Name");
   //Any error to report?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to set instance name!\r\n");
   }

   //Register DNS-SD service
   error = dnsSdRegisterService(&dnsSdContext,
      "_http._tcp", 0, 0, 80,
      "txtvers=1;path=/");

   //Any error to report?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to register service!\r\n");
   }

   //Start DNS-SD
   error = dnsSdStart(&dnsSdContext);
   //Failed to start DNS-SD?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to start DNS-SD!\r\n");
   }
```

9.4 Project (for IPv6 users only)

New files must be added to the project. These changes only apply to application where IPv6 is currently used:

- cyclone_tcp/ipv6/ipv6_misc.c
- cyclone_tcp/ipv6/ipv6_misc.h
- cyclone_tcp/ipv6/ipv6_pmtu.c
- cyclone_tcp/ipv6/ipv6_pmtu.h
- cyclone_tcp/ipv6/ndp_cache.c
- cyclone_tcp/ipv6/ndp_cache.h
- cyclone_tcp/ipv6/ndp_misc.c
- cyclone_tcp/ipv6/ndp_misc.c

Please note that a new file (cyclone_tcp/core/net_legacy.h) has been introduced. This file contains deprecated properties and functions.

9.5 IPv6 API (for IPv6 users only)

In order to support multiple global IPv6 addresses, IPv6 prefixes and default routers, the IPv6 related API has been slightly modified to feature a "index" parameter. This parameter selects an item in the list of global addresses, IPv6 prefixes and default routers :

```
error_t ipv6SetGlobalAddr(NetInterface *interface,
   uint_t index, const Ipv6Addr *addr);

error_t ipv6GetGlobalAddr(NetInterface *interface,
   uint_t index, Ipv6Addr *addr);

error_t ipv6SetPrefix(NetInterface *interface,
   uint_t index, const Ipv6Addr *prefix, uint_t length);

error_t ipv6GetPrefix(NetInterface *interface,
   uint_t index, Ipv6Addr *prefix, uint_t *length);

error_t ipv6SetDefaultRouter(NetInterface *interface,
   uint_t index, const Ipv6Addr *addr);
error_t ipv6GetDefaultRouter(NetInterface *interface,
   uint_t index, Ipv6Addr *addr);
```

By default the IPv6 stack supports 3 IPv6 addresses (1 link-local address + 2 global addresses), 2 IPv6 prefixes and 2 default routers. This is the minimum configuration to comply with TAHI conformance test suite. Nevertheless, the net_config.h configuration file can be tuned to override theses parameters:

```
//Size of the IPv6 address list
#define IPV6_ADDR_LIST_SIZE 3
//Size of the prefix list
#define IPV6_PREFIX_LIST_SIZE 2
//Maximum number number of default routers
#define IPV6_ROUTER_LIST_SIZE 2
```

9.6 Miscellaneous changes (for both IPv4 and IPv6 users)

Some configuration properties has been renamed for coherency purposes across modules but still keep the same definition :

Deprecated names	New names
IPV4_MAX_DNS_SERVERS	IPV4_DNS_SERVER_LIST_SIZE
IPV6_MAX_DNS_SERVERS	IPV6_DNS_SERVER_LIST_SIZE
MAC_FILTER_MAX_SIZE	MAC_MULTICAST_FILTER_SIZE
IPV4_FILTER_MAX_SIZE	IPV4_MULTICAST_FILTER_SIZE
IPV6_FILTER_MAX_SIZE	IPV6_MULTICAST_FILTER_SIZE

Warnings may be reported during compilation if the deprecated properties are still used in net_config.h (but the source code will still compile).

9.7 Device drivers

The structure of the Network Interface Controller (NIC) has been slightly modified to simply the interaction between MAC and PHY drivers.

Old NicDriver structure	New NicDriver structure
typedef struct { NicType type; size_t mtu; NicInit init; NicTick tick; NicEnableIrq enableIrq; NicDisableIrq disableIrq; NicEventHandler eventHandler; NicSetMacFilter setMacFilter; NicSendPacket sendPacket; NicWritePhyReg writePhyReg; NicReadPhyReg readPhyReg; bool_t autoPadding; bool_t autoCrcGen; bool_t autoCrcCheck; } NicDriver;	typedef struct { NicType type; size_t mtu; NicInit init; NicTick tick; NicEnableIrq enableIrq; NicDisableIrq disableIrq; NicEventHandler eventHandler; NicSendPacket sendPacket; NicSetMulticastFilter setMulticastFilter; NicUpdateMacConfig updateMacConfig; NicWritePhyReg writePhyReg; NicReadPhyReg readPhyReg; bool_t autoPadding; bool_t autoCrcGen; bool_t autoCrcCheck; } NicDriver;

- The position of the function that set the multicast MAC filter has changed and its name has been modified for coherency purpose.
- A new function has been introduced (updateMacConfig) to allow a PHY driver to update dynamically the configuration of a MAC driver. For all-integrated Ethernet controller or 6LoWPAN drivers, this new field can be NULL.

Because of the new stack model, a common event object is used to notify the core of the TCP/IP stack of driver events.

- All occurrences of osSetEvent(&interface->nicRxEvent) shall be replace by the following sequence:

```
interface->nicEvent = TRUE;
osSetEvent(&netEvent);
```

- All occurrences of osSetEventFromIsr(&interface->nicRxEvent) shall be replace by the following sequence:

```
interface->nicEvent = TRUE;
osSetEventFromIsr(&netEvent);
```


10 Migrating from CycloneTCP 1.6.0 to 1.6.4

10.1 Ping utility

The prototype of the ping function has changed. It is now possible to specify the size of the data and the TTL value :

```
//New function prototype
error_t ping(NetInterface *interface, const IpAddr *ipAddr,
   size_t size, uint8_t ttl, systime_t timeout, systime_t *rtt);
```


11 Migrating from CycloneTCP 1.5.0 to 1.6.0

11.1 Project

The tcpIpStack* namespace has been replaced to net*. As a consequence:

- You MUST rename you configuration file from "tcp_ip_stack_config.h" to "net_config.h"
- You MUST remove "core/tcp_ip_stack.c" and "core/tcp_ip_stack_mem.c" from your project and link "core/net.c" and "core/net_mem.c" instead
- All references to "core/tcp_ip_stack.h" MUST be replaced with "core/net.h" in your source code

```
//Dependencies
#include "core/net.h"
```

- The following functions are deprecated. It is highly recommended (but not mandatory) to switch to the new API instead. The legacy API is still available to maintain compatibility with existing projects:

Deprecated functions	New functions
tcpIpStackInit	netInit
tcpIpStackConfigInterface	netConfigInterface
tcpIpStackGetLinkState	netGetLinkState
tcpIpStackGetDefaultInterface	netGetDefaultInterface
tcpIpStackInitRand	netInitRand
tcpIpStackSetHostname	netSetHostname
tcpIpStackSetDriver	netSetDriver
tcpIpStackSetPhyDriver	netSetPhyDriver
tcpIpStackSetSpiDriver	netSetSpiDriver
tcpIpStackSetExtIntDriver	netSetExtIntDriver
tcpIpStackSetUartDriver	netSetUartDriver
tcpIpStackSetMacAddr	netSetMacAddr
tcpIpStackSetProxy	netSetProxy

- The following properties are deprecated. It is highly recommended (but not mandatory) to use the new property names instead:

Deprecated property names	New property names
TCP_IP_TICK_STACK_SIZE	NET_TICK_STACK_SIZE
TCP_IP_TICK_PRIORITY	NET_TICK_PRIORITY
TCP_IP_RX_STACK_SIZE	NET_RX_STACK_SIZE
TCP_IP_RX_PRIORITY	NET_RX_PRIORITY
TCP_IP_TICK_INTERVAL	NET_TICK_INTERVAL
TCP_IP_STATIC_OS_RESOURCES	NET_STATIC_OS_RESOURCES
MEM_POOL_SUPPORT	NET_MEM_POOL_SUPPORT
MEM_POOL_BUFFER_COUNT	NET_MEM_POOL_BUFFER_COUNT
MEM_POOL_BUFFER_SIZE	NET_MEM_POOL_BUFFER_SIZE
TCP_SYN_QUEUE_SIZE	TCP_DEFAULT_SYN_QUEUE_SIZE

11.2 HTTP Server

The HTTP server now consists of 3 files:

- http_server.c
- http_server_auth.c
- http_server_misc.c

The HTTP server does not longer create/delete tasks dynamically. All the necessary resources (including client tasks) are created statically at startup (in httpServerInit and httpServerStart functions). Your Web server initialization routine shall be modified as follows to allocate these resources statically:

```
//Dependencies
#include "net.h"
#include "http_server.h"

//Number of simultaneous client connections
#define APP_HTTP_MAX_CONNECTIONS 4

//Global variables
HttpServerContext httpServerContext;
HttpConnection httpConnections[APP_HTTP_MAX_CONNECTIONS];

   //Local variables
   HttpServerSettings httpServerSettings;

   //Get default settings
   httpServerGetDefaultSettings(&httpServerSettings);
   //Bind HTTP server to the desired interface
   httpServerSettings.interface = &netInterface[0];
   //Listen to port 80
   httpServerSettings.port = HTTP_PORT;
   //Client connections
   httpServerSettings.maxConnections = APP_HTTP_MAX_CONNECTIONS;
   httpServerSettings.connections = httpConnections;
   //Specify the server's root directory
   strcpy(httpServerSettings.rootDirectory, "/www/");
   //Set default home page
   strcpy(httpServerSettings.defaultDocument, "index.htm");
   //Callback functions
   //...

   //HTTP server initialization
   error = httpServerInit(&httpServerContext, &httpServerSettings);

   //Start HTTP server
   error = httpServerStart(&httpServerContext);
```

The HTTP_SERVER_MAX_CONNECTIONS property does not longer exist and should be removed from your "net_config.h" configuration file.

The HTTP connections are made non-persistent by default in version 1.6.0, contrary to version 1.5.0. If you still want to use persistent HTTP connections (for instance for HTTPS servers, in order to process multiple requests with a single key establishment), please modify your net_config.h configuration file as follows :

```
//Force HTTP server to use persistent connections
#define HTTP_SERVER_PERSISTENT_CONN_SUPPORT ENABLED
```

11.3 Socket Interface

The socketListen now takes an extra parameter (backlog) to set the maximum length of the pending connection queue. If this parameter is zero, then the default backlog value is used instead.

```
error = socketListen(socket, 0);
```

11.4 Network Interface Drivers

All the device drivers includes a new field that identify the type of device :

- NIC_TYPE_ETHERNET for Ethernet drivers
- NIC_TYPE_PPP for PPP drivers
- NIC_TYPE_6LOWPAN for 6LoWPAN drivers

All driver structures also feature the link MTU (1500 bytes for Ethernet)

```
/**
 * @brief STM32F407/417/427/437 Ethernet MAC driver
 **/

const NicDriver stm32f4x7EthDriver =
{
   NIC_TYPE_ETHERNET,
   ETH_MTU,
   stm32f4x7EthInit,
   stm32f4x7EthTick,
   stm32f4x7EthEnableIrq,
   stm32f4x7EthDisableIrq,
   stm32f4x7EthEventHandler,
   stm32f4x7EthSetMacFilter,
   stm32f4x7EthSendPacket,
   stm32f4x7EthWritePhyReg,
   stm32f4x7EthReadPhyReg,
   TRUE,
   TRUE,
   TRUE
};
```

12 Migrating from CycloneTCP 1.4.4 to 1.5.0

12.1 Project Settings

The subdirectories in cyclone_tcp (cyclone_tcp/core, cyclone_tcp/ipv4, cyclone_tcp/dhcp, etc) shall not be included anymore in the "include path" of your C compiler. For sake of simplicity, only the following directories are necessary:

- common
- cyclone_tcp
- cyclone_ssl (if applicable)
- cyclone_crypto (if applicable)

As a consequence, you may have to change the include directives in your source code files. For example, replace the following include directives:

```
//Dependencies
#include "tcp_ip_stack.h"
#include "stm32f4x7_eth.h"
#include "dp83848.h"
#include "dhcp_client.h"
#include "slaac.h"
#include "http_server.h"
#include "mime.h"
```

…by the new ones:

```
//Dependencies
#include "core/tcp_ip_stack.h"
#include "drivers/stm32f4x7_eth.h"
#include "drivers/dp83848.h"
#include "dhcp/dhcp_client.h"
#include "ipv6/slaac.h"
#include "http/http_server.h"
#include "http/mime.h"
```

Note that a new file (compiler_port.h) has been added in the common/ directory. You can add this file to your project.

12.2 RTOS Abstraction Layer

Because of name collision with CMSIS-RTOS, the entire API has been renamed. This brand new RTOS abstraction layer also makes the process of porting of the TCP/IP stack to new operating systems easier.

Normally, the RTOS abstraction layer is only used internally. Thus, this modification shall not impact users. However, if you make use of some of the functions of the RTOS abstraction layer, you have to switch to the new functions :

Deprecated functions	New functions
osInit	osInitKernel
osStart	osStartKernel
osTaskCreate	osCreateTask
osTaskDelete	osDeleteTask
osDelay	osDelayTask
osTaskSwitch	osSwitchTask
osTaskSuspendAll	osSuspendAllTasks
osTaskResumeAll	osResumeAllTasks
osEventCreate	osCreateEvent
osEventClose	osDeleteEvent
osEventSet	osSetEvent
osEventReset	osResetEvent
osEventWait	osWaitForEvent
osEventSetFromIrq	osSetEventFromIsr
osSemaphoreCreate	osCreateSemaphore
osSemaphoreClose	osDeleteSemaphore
osSemaphoreWait	osWaitForSemaphore
osSemaphoreRelease	osReleaseSemaphore
osMutexCreate	osCreateMutex
osMutexClose	osDeleteMutex
osMutexAcquire	osAcquireMutex
osMutexRelease	osReleaseMutex
osGetTickCount	osGetSystemTime
osMemAlloc	osAllocMem
osMemFree	osFreeMem
osQueueCreate	No longer used
osQueueClose	No longer used
osQueueSend	No longer used
osQueueReceive	No longer used
osQueuePeek	No longer used
osQueueSendFromIrq	No longer used
osQueueReceiveFromIrq	No longer used

12.3 Cortex-M3/M4 Specific Changes

To make the TCP/IP stack portable among various RTOS environments, the Ethernet drivers now configure all the priority bits of the Cortex-M core to pre-emption priority (no bits for subpriority). You can change this default behavior, by overriding some properties in your tcp_ip_stack_config.h configuration file.

For instance, if you use the STM32F4x7 Ethernet MAC driver, you can override the following properties with the desired settings:

- `STM32F4X7_ETH_IRQ_PRIORITY_GROUPING`
- `STM32F4X7_ETH_IRQ_GROUP_PRIORITY`
- `STM32F4X7_ETH_IRQ_SUB_PRIORITY`

13 Migrating from CycloneTCP 1.4.2 to 1.4.4

13.1 HTTP Server

The prototypes of the callback routines have been modified. If you make use of callbacks replace the deprecated prototypes…

```
//HTTP server callback routines
error_t httpServerTlsInitCallback(HttpConnection *connection);

HttpAccessStatus httpServerBasicAuthCallback(HttpConnection *connection);

error_t httpServerCgiCallback(HttpConnection *connection,
   const char_t *param);

error_t httpServerUriNotFoundCallback(HttpConnection *connection);
```

…by the new ones:

```
//HTTP server callback routines
error_t httpServerTlsInitCallback(HttpConnection *connection,
   TlsContext *tlsContext);

HttpAccessStatus httpServerAuthCallback(HttpConnection *connection,
   const char_t *user, const char_t *uri);

error_t httpServerCgiCallback(HttpConnection *connection,
   const char_t *param);

error_t httpServerUriNotFoundCallback(HttpConnection *connection,
   const char_t *uri);
```

14 Migrating from CycloneTCP 1.4.1 to 1.4.2

14.1 FTP Server

The prototypes of the callback routines have been modified. If you make use of callbacks to check usernames, passwords and right accesses, replace the deprecated prototypes…

```
//FTP server callback routines
uint_t ftpServerCheckUserCallback(const char_t *user);

uint_t ftpServerCheckPasswordCallback(const char_t *user,
   const char_t *password);

uint_t ftpServerGetFilePermCallback(char_t *user,
   const char_t *path);
```

…by the new ones:

```
//FTP server callback routines
uint_t ftpServerCheckUserCallback(FtpClientConnection *connection,
   const char_t *user);

uint_t ftpServerCheckPasswordCallback(FtpClientConnection *connection,
   const char_t *user, const char_t *password);

uint_t ftpServerGetFilePermCallback(FtpClientConnection *connection,
   const char_t *user, const char_t *path);
```

15 Migrating from CycloneTCP 1.4.0 to 1.4.1

15.1 Project Settings

- Add the following directories to your include path:
  - cyclone_tcp/dns
  - cyclone_tcp/mdns

15.2 DNS Client

- Remove the following files from your project:
  - cyclone_tcp/core/dnc_client.c
  - cyclone_tcp/core/dnc_client.h

- Add the following files to your project:
  - cyclone_tcp/dns/dns_cache.c
  - cyclone_tcp/dns/dns_cache.h
  - cyclone_tcp/dns/dns_client.c
  - cyclone_tcp/dns/dns_client.h
  - cyclone_tcp/dns/dns_common.c
  - cyclone_tcp/dns/dns_common.h
  - cyclone_tcp/dns/dns_debug.c
  - cyclone_tcp/dns/dns_debug.h

- getHostByName prototype has been simplified. The 4th and the 5th parameters have been removed. Please read the user manual to get more details about this function.

```
IpAddr ipAddr;

//Resolve host name
error = getHostByName(NULL, "www.example.com", &ipAddr, 0);
```

15.3 mDNS Client

- Add the following files to your project:
  - cyclone_tcp/dns/mdns_client.c
  - cyclone_tcp/dns/mdns_client.h
  - cyclone_tcp/dns/mdns_responder.c
  - cyclone_tcp/dns/mdns_responder.h
  - cyclone_tcp/dns/mdns_common.c
  - cyclone_tcp/dns/mdns_common.h

- mDNS client and responder features can enable or disabled using MDNS_CLIENT_SUPPORT and MDNS_RESPONDER_SUPPORT compilation flags

- If mDNS responder is being used, the hostname is configured as follows:

```
//Configure the first Ethernet interface
interface = &netInterface[0];
//Set host name
tcpIpStackSetHostname(interface, "FTPServerDemo");
```

16 Migrating from CycloneTCP 1.3.8 to 1.4.0

16.1 FreeRTOS Port

- As CycloneTCP supports multiple RTOS, os.c and os.h files are now deprecated. For FreeRTOS users, os_port_freertos.c shall be used instead of os.c. Also consider to include os_port.h and os_port_freertos.h in you code instead of os.h.

- Remove USE_FREERTOS from compiler settings (preprocessor macro)

- Add a new file to your project os_port_config.h. This file shall contain

```
#define USE_FREERTOS
```

- At the end of your FreeRTOSConfig.h file, remove the following defines

```
#define pvPortMalloc osMemAlloc
#define vPortFree    osMemFree
```

- Add one of the following FreeRTOS files to your project in order to manage memory allocation. The files can be copied from demo\common\freertos\portable\memmang

File	Description
heap_1.c	The simplest possible implementation of pvPortMalloc(). Note that this implementation does not allow allocated memory to be freed again
heap_2.c	A simple implementation of pvPortMalloc() and vPortFree() that permits allocated blocks to be freed, but does not combine adjacent free blocks into a single larger block (and so will fragment memory)
heap_3.c	Implementation of pvPortMalloc() and vPortFree() that relies on the compiler own malloc() and free() implementations
heap_4.c	A sample implementation of pvPortMalloc() and vPortFree() that combines (coalescences) adjacent memory blocks as they are freed, and in so doing limits memory fragmentation.

Using heap_3.c maintains compatibility with existing projects (linker files are not required to be changed using this file).

If you consider using heap_4.c, please update FreeRTOS configuration header by settings the desired heap size (configTOTAL_HEAP_SIZE compilation flag)

```
//Define your heap size here (in bytes)
#define configTOTAL_HEAP_SIZE 32768
```

- Add common/date_time.c file to your project

16.2 Network Interface Initialization

It is recommended to replace the deprecated initialization sequence…

```
void main(void)
{
   //...

   //Configure the first Ethernet interface
   interface = &netInterface[0];

   //Select the relevant network adapter
   interface->nicDriver = &stm32f2x7EthDriver;
   interface->phyDriver = &dp83848PhyDriver;
   //Interface name
   strcpy(interface->name, "eth0");
   //Set host MAC address
   macStringToAddr("00-AB-CD-EF-02-07", &interface->macAddr);

   //Initialize network interface
   error = tcpIpStackConfigInterface(interface);

   //...
}
```

…by the following one

```
void main(void)
{
   MacAddr macAddr;

   //...

   //Configure the first Ethernet interface
   interface = &netInterface[0];

   //Set interface name
   tcpIpStackSetInterfaceName(interface, "eth0");
   //Select the relevant network adapter
   tcpIpStackSetDriver(interface, &stm32f2x7EthDriver);
   tcpIpStackSetPhyDriver(interface, &dp83848PhyDriver);
   //Set host MAC address
   macStringToAddr("00-AB-CD-EF-02-07", &macAddr);
   tcpIpStackSetMacAddr(interface, &macAddr);

   //Initialize network interface
   error = tcpIpStackConfigInterface(interface);

   //...
}
```

16.3 DHCP Client

DHCP client API has evolved since DHCP client has now start/stop feature.

- It is recommended to call dhcpClientGetDefaultSettings to retrieve default settings before customizing some of theses settings
- dhcpClientInit must be called to initialize the DHCP client. When the code returns from dhcpClientInit, the DHCP client is not yet running. For that purpose your must call the dhcpClientStart function

Here is the correct sequence to initialize the DHCP client:

```
//DHCP client context shall be global (or static)
DhcpClientCtx dhcpClientContext;

void main(void)
{
   //Settings structure can be a local variable
   DhcpClientSettings dhcpClientSettings;

   //...

   //Get default settings
   dhcpClientGetDefaultSettings(&dhcpClientSettings);
   //Set the network interface to be configured by DHCP
   dhcpClientSettings.interface = &netInterface[0;
   //Disable rapid commit option
   dhcpClientSettings.rapidCommit = FALSE;

   //DHCP client initialization
   error = dhcpClientInit(&dhcpClientContext, &dhcpClientSettings);
   //Failed to initialize DHCP client?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to initialize DHCP client!\r\n");
   }

   //Start DHCP client
   error = dhcpClientStart(&dhcpClientContext);
   //Failed to start DHCP client?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to start DHCP client!\r\n");
   }

   //...
}
```

16.4 IPv4 Manual Configuration

It is recommended to replace the deprecated IPv4 manual configuration sequence…

```
//...

//Manual configuration
interface = &netInterface[0];

//IPv4 address
ipv4StringToAddr("192.168.0.20", &interface->ipv4Config.addr);
//Subnet mask
ipv4StringToAddr("255.255.255.0", &interface->ipv4Config.subnetMask);
//Default gateway
ipv4StringToAddr("192.168.0.254", &interface >ipv4Config.defaultGateway);

//Primary and secondary DNS servers
interface->ipv4Config.dnsServerCount = 2;
ipv4StringToAddr("212.27.40.240", &interface->ipv4Config.dnsServer[0]);
ipv4StringToAddr ("212.27.40.241", &interface->ipv4Config.dnsServer[1]);

//...
```

…by the following one

```
Ipv4Addr ipv4Addr;

//...

//Set IPv4 host address
ipv4StringToAddr("192.168.0.20", &ipv4Addr);
ipv4SetHostAddr(interface, ipv4Addr);

//Set subnet mask
ipv4StringToAddr("255.255.255.0", &ipv4Addr);
ipv4SetSubnetMask(interface, ipv4Addr);

//Set default gateway
ipv4StringToAddr("192.168.0.254", &ipv4Addr);
ipv4SetDefaultGateway(interface, ipv4Addr);

//Set primary and secondary DNS servers
ipv4StringToAddr("212.27.40.240", &ipv4Addr);
ipv4SetDnsServer(interface, 0, ipv4Addr);
ipv4StringToAddr ("212.27.40.241", &ipv4Addr);
ipv4SetDnsServer(interface, 1, ipv4Addr);

//...
```

16.5 IPv6 Manual Configuration

- The IPv6 Link-Local address cannot be accessed directly. Instead you must use ipv6SetLinkLocalAddr to properly configure the Link-Local address. This function **must not be called prior to network interface initialization**. It shall be called after tcpIpStackConfigInterface.

- ipv6ComputeSolicitedNodeAddr and ipv6JoinMulticastGroup functions **do not need to be called anymore**. The process of joining the relevant multicast group is done automatically when calling ipv6SetLinkLocalAddr or ipv6SetGlobalAddr.

You must replace the deprecated IPv6 manual configuration sequence…

```
//Select the relevant interface
interface = &netInterface[0];

//Set link-local IPv6 address
ipv6StringToAddr("fe80::abcd", &interface->ipv6Config.linkLocalAddr);

//...

//Initialize network interface
error = tcpIpStackConfigInterface(interface);

//...

//Prefix
interface->ipv6Config.prefixLength = 64;
ipv6StringToAddr("1122:3344:5566:7788::", &interface->ipv6Config.prefix);

//Global address
ipv6StringToAddr("1122:3344:5566:7788::abcd",
   &interface->ipv6Config.globalAddr);
//Router
ipv6StringToAddr("fe80::eeff", &interface->ipv6Config.router);

//Primary and secondary DNS servers
interface->ipv6Config.dnsServerCount = 2;
ipv6StringToAddr("2a01:e00::1", &interface->ipv6Config.dnsServer[0]);
ipv6StringToAddr("2a01:e00::2", &interface->ipv6Config.dnsServer[1]);

//A host is required to join a Solicited-Node multicast group for each of
//its configured unicast address
ipv6ComputeSolicitedNodeAddr(&interface->ipv6Config.globalAddr,
&solicitedNodeAddr);
ipv6JoinMulticastGroup(interface, &solicitedNodeAddr);
```

…by the following one

```
Ipv6Addr ipv6Addr;

//Select the relevant interface
interface = &netInterface[0];

//Initialize network interface
error = tcpIpStackConfigInterface(interface);

//...

//Set link-local address
ipv6StringToAddr("fe80::abcd", &ipv6Addr);
ipv6SetLinkLocalAddr(interface, &ipv6Addr);

//Set IPv6 prefix
ipv6StringToAddr("1122:3344:5566:7788::", &ipv6Addr);
ipv6SetPrefix(interface, &ipv6Addr, 64);

//Set global address
ipv6StringToAddr("1122:3344:5566:7788::abcd", &ipv6Addr);
ipv6SetGlobalAddr(interface, &ipv6Addr);

//Set router
ipv6StringToAddr("fe80::eeff", &ipv6Addr);
ipv6SetRouter(interface, &ipv6Addr);

//Set primary and secondary DNS servers
ipv6StringToAddr("2a01:e00::1", &ipv6Addr);
ipv6SetDnsServer(interface, 0, &ipv6Addr);
ipv6StringToAddr ("2a01:e00::2", &ipv6Addr);
ipv6SetDnsServer(interface, 1, &ipv6Addr);
```

16.6 HTTP Server

The following initialization sequence shall be used to initialize the HTTP server

```
//HTTP server context shall be global (or static)
HttpServerContext httpServerContext;

void main(void)
{
   //Settings structure can be a local variable
   HttpServerSettings httpServerSettings;

   //...

   //Get default settings
   httpServerGetDefaultSettings(&httpServerSettings);
   //Bind HTTP server to the desired interface
   httpServerSettings.interface = &netInterface[0];
   //Listen to port 80
   httpServerSettings.port = HTTP_PORT;
   //Specify the server's root directory
   strcpy(httpServerSettings.rootDirectory, "/www/");
   //Set default home page
   strcpy(httpServerSettings.defaultDocument, "index.shtm");
   //Callback functions
   httpServerSettings.cgiCallback = httpServerCgiCallback;
   httpServerSettings.uriNotFoundCallback = httpServerUriNotFoundCallback;

   //HTTP server initialization
   error = httpServerInit(&httpServerContext, &httpServerSettings);
   //Failed to initialize HTTP server?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to initialize HTTP server!\r\n");
   }

   //Start HTTP server
   error = httpServerStart(&httpServerContext);
   //Failed to start HTTP server?
   if(error)
   {
      //Debug message
      TRACE_ERROR("Failed to start HTTP server!\r\n");
   }

   //...
}
```