1 Migrating from CycloneTCP 1.6.0 to 1.6.4

1.1 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);
```


2 Migrating from CycloneTCP 1.5.0 to 1.6.0

2.1 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

2.2 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
```

2.3 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);
```

2.4 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
};
```


3 Migrating from CycloneTCP 1.4.4 to 1.5.0

3.1 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.

3.2 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

3.3 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`

4 Migrating from CycloneTCP 1.4.2 to 1.4.4

4.1 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);
```

5 Migrating from CycloneTCP 1.4.1 to 1.4.2

5.1 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);
```

6 Migrating from CycloneTCP 1.4.0 to 1.4.1

6.1 Project Settings

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

6.2 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);
```

6.3 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");
```

7 Migrating from CycloneTCP 1.3.8 to 1.4.0

7.1 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

7.2 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);

   //...
}
```

7.3 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");
   }

   //...
}
```

7.4 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);

//...
```

7.5 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);
```

7.6 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");
   }

   //...
}
```