


# MT793X IoT SDK for Open Source Software Guide

Version: 1.3

Release date: 2022-11-21

# Version History

| Version | Date | Description |
|---|---|---|
| 1.0 | 2021-03-29 | Initial draft. Include the lwIP module. |
| 1.1 | 2021-06-04 | Update HTTPClient, SNTP and ZLMA decoder descriptions |
| 1.2 | 2022-01-26 | Update cJson, MQTT, Websocket descriptions. |
| 1.3 | 2022-11-21 | Update mbedtls and MQTT/XML CVE issue notification |

# Table of Contents

## List of Figures



## List of Tables

# 1 Overview

In the new IoT era, devices communicate with various Internet of Things (IoT) protocols to form a smart network. Many new protocols have been recently defined to meet the requirements of machine-to-machine communication on devices with constrained resources. Open standards enable IoT growth in the near future. MediaTek offers IoT SDK with mainstream open-source software to help you build your projects with globally available resources and a full control over the software. The adopted standards of IoT are still changing, but the SDK enables you to stay abreast with the updates and new specifications.

The SDK includes and supports all of the major protocols, such as the fundamental TCP/IP stack, the TLS library for secure data transport. When building these protocols, a minimal memory footprint is a major requirement. MediaTek provides competitive packages with a small memory footprint without compromising essential functions. These software packages are used to build applications and easily connect devices in the IoT world. The SDK requires RTOS as an underlying OS, more specifically FreeRTOS, the leading real-time operating system in the embedded systems market. All of the open-source software is distributed in the source release, therefore, you can make any of the necessary changes to satisfy your needs and redistribute the software under the rights permitted by the corresponding license.

This guide provides information about the open-source software bundled in the SDK. It provides information about the open-source software packages and guides you in designing, prototyping, and implementing IoT projects in a convenient environment.

As an open-source software package, the information is already available in various sources. This guide is an easy reference to module descriptions, including the official website and the hardware or software integration versions. You can access the latest source code from the official website and merge or replace the code in the package with a corresponding version. From the footprint statistics result, you can easily estimate the code size for setting up the flash memory layout. Before commencing your own IoT project or application development, look for example applications and their source code. Finally, the troubleshooting and limitations chapter provides more detailed information on reported issues that may be encountered during the application development.

## 1.1 Open Source Software Architecture of the SDK

The SDK integrates several widely adopted protocol implementations to reduce the development effort and enable device communication over the IP channel. The open source communication protocols and libraries included in the SDK are mbedtls (SSL/TLS) and lwIP (IPv4). They provide the communication layer between the applications and the FreeRTOS. The architecture layout of the SDK is shown in Figure 1. lwIP is a TCP/IP open-source protocol that provides the transport for sending and receiving data between the nodes connected through the Internet. SSL/TLS establishes secure data transport protection on TCP connections. Running HTTP over SSL/TLS enables secure data transfer.

In addition, the open-source FatFs is included to provide the FAT (File Allocation Table) based on file and directory operations. The Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and defines generic tool interfaces. The CMSIS enables consistent device support and simple software interfaces to the processor and the peripherals.

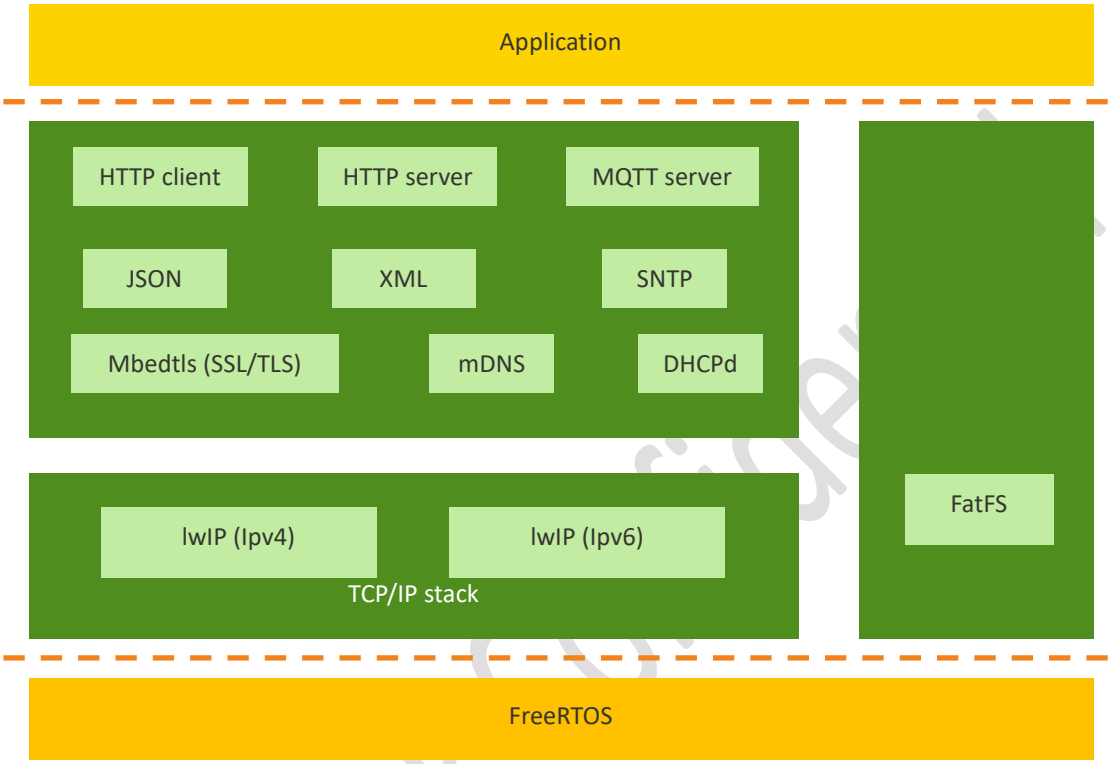



***Figure 1. Architecture layout of the SDK's open source software***

## 1.2 Open Source Software Resources for the SDK

The open-source software packages in the SDK are listed in Table 1 and for your reference only. You must acknowledge that such listed open-source software may be supplemented or amended by MediaTek from time to time. You must also comply with all licensing terms applicable to such open-source software. MediaTek makes the following disclaimers regarding the open-source software on behalf of itself, and the copyright holders, contributors, and licensors of the listed open-source software: TO THE FULLEST EXTENT PERMITTED UNDER APPLICABLE LAW, THE OPEN SOURCE SOFTWARE ARE PROVIDED BY THE COPYRIGHT HOLDERS, CONTRIBUTORS, LICENSORS, AND MEDIATEK "AS IS" AND ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER ORAL OR WRITTEN, WHETHER EXPRESS, IMPLIED, OR ARISING BY STATUTE, CUSTOM, COURSE OF DEALING, OR TRADE USAGE, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, ARE DISCLAIMED. IN NO EVENT WILL THE COPYRIGHT OWNER, CONTRIBUTORS, LICENSORS, OR MEDIATEK BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE OPEN SOURCE SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

***Table 1. Open source packages***

| Module | Open source software license | Comments |
|---|---|---|
| **RTOS** | FreeRTOS<br>License: MIT License<br>Please refer to license disclaimer in Kernel/rtos/FreeRTOS/SourceV10/include/FreeRTOS.h<br>Note: FreeRTOS V10 and later version are distributed under the MIT license.<br>http://www.freertos.org/a00114.html | Market leading, de-facto standard OS for embedded systems. |
| **TCP/IP** | LwIP<br>License: BSD License, MediaTek EULA<br>http://lwip.wikia.com/wiki/License | LwIP (lightweight IP) is a widely used open source TCP/IP stack designed for embedded systems. |
| **FatFs** | FatFs<br>License: BSD-style license<br>http://elm-chan.org/fsw/ff/pf/appnote.html | FatFs is a generic FAT file system module for small embedded systems. |
| **CMSIS** | CMSIS<br>License: BSD License<br>https://developer.mbed.org/blog/entry/CMSIS-Components-BSD-Licensed/ | CMSIS is a vendor-independent hardware abstraction layer for the Cortex-M processor series and defines generic tool interfaces. |
| **Micro-ecc** | micro-ecc<br>License: BSD 2-Clause "Simplified"<br>https://github.com/kmackay/micro-ecc | A small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors. |
| **HTTP client** | mbed HTTP client<br>License: MIT License, MediaTek License<br>https://developer.mbed.org/users/WiredHome/code/HTTPClient/file/3556275bebf3/HTTPClient.cpp | A small HTTP client supporting HTTP and HTTPS. The porting from .cpp to .c is implemented by MediaTek |
| **SNTP** | lwIP-contrib SNTP<br>License: Modified BSD License<br>http://lwip.wikia.com/wiki/License | Implementation of an SNTP client to retrieve the GMT time from servers. |
| **LZMA decoder** | LZMA<br>License: public domain<br>http://www.7-zip.org/sdk.html | LZMA decoder is extracted from the LZMA SDK for further use. |
| **JSON** | cJson<br>License: MIT License | An ultra-lightweight, portable single-file, simple-as-can-be ANSI-C compliant JSON parser. |

| | | |
|---|---|---|
| | https://github.com/DaveGamble/cJSON/blob/master/LICENSE | |
| **MQTT Client** | Eclipse Paho Embedded MQTT C/C++ client<br>License: (Dual license: EPL or EDL)<br>http://www.eclipse.org/org/documents/epl-v10.php | The Paho project provides an open-source client implementation of the MQTT messaging protocols. |
| **Websocket** | Librws<br>License: MIT License<br>https://github.com/OlehKulykov/librws | Librws is a tiny and cross platform WebSocket client C library. |
| **SSL/TLS** | mbed TLS<br>License: Apache 2.0 License<br>http://www.apache.org/licenses/LICENSE-2.0 | Easy to use SSL/TLS library including cryptographic and SSL/TLS capabilities with small footprint. |

The SDK open-source packages are implemented by the active open-source community with widely available online resources and forum support. The list of online resources is provided for each module in Table 2. The integrated versions are also included for you to merge hot bug fixes or new features directly from the official release links.

*Table 2. Online resources for each module in the SDK*

| Module | Official website | Integrated version | Online API reference |
|---|---|---|---|
| **RTOS** | http://www.freertos.org/RTOS.html | 10.4.3 | http://www.freertos.org/a00106.html |
| **TCP/IP** | http://savannah.nongnu.org/projects/lwip/<br>http://lwip.wikia.com/wiki/LwIP_Wiki | 2.1.2 | https://lwip.fandom.com/wiki/Application_API_layers |
| **SSL/TLS** | https://tls.mbed.org/ | 2.25.0 | https://tls.mbed.org/api/ |
| **FatFs** | http://elm-chan.org/fsw/ff/00index_e.html | R0.12(1) | http://elm-chan.org/fsw/ff/00index_e.html |
| **CMSIS** | http://www.keil.com/pack/doc/CMSIS/General/html/index.html | 5.7.0 | http://www.keil.com/pack/doc/CMSIS/Core/html/modules.html |
| **Micro-ecc** | https://github.com/kmackay/micro-ecc | V1.0 | https://github.com/kmackay/micro-ecc |
| **HTTP client** | https://os.mbed.com/users/WiredHome/code/HTTPClient/ | Revision: 34:3556275bebf3 | No online API.<br>Exported API with comments can be found at <sdk_root>/middleware/third_party/httpclient/inc/httpclient.h |
| **SNTP** | http://savannah.nongnu.org/projects/lwip/<br>http://lwip.wikia.com/wiki/LwIP_Wiki | Commit ID: 06dee8d | No online API.<br>Please find exported API at <sdk_root>/middleware/third_party/lwip/src/include/lwip/apps/sntp.h. |
| **LZMA decoder** | http://www.7-zip.org/sdk.html | 15.08 | http://www.7-zip.org/sdk.html |
| **JSON** | https://github.com/DaveGamble/cJSON | 1.7.12 | No online API.<br>Please find exported API at <sdk_root>/middleware/third_party/cjson/inc/cJSON.h. |
| **MQTT client** | https://www.eclipse.org/paho/clients/c/embedded/ | 1.1.0 | https://www.eclipse.org/paho/clients/c/embedded/ |
| **WebSocket** | https://github.com/OlehKulykov/librws | 1.2.3 | https://github.com/OlehKulykov/librws |

The following chapters give a more detailed description of each module.

# 2 RTOS: FreeRTOS

FreeRTOS is a real-time OS that manages a multitasking and multiprocessing system environment. It has a scheduler to manage user-created tasks. FreeRTOS provides APIs for you to control, synchronize, and communicate among various tasks.

## 2.1 Features

FreeRTOS supports the following five features to accomplish event/task scheduling and multitasking:

- Task and Scheduler – Each application consists of tasks or threads controlled by the operating system. The multitasking operation is implemented with a scheduler. The scheduler is in the kernel and manages the task execution at a specific time. The kernel can suspend and resume a task many times during lifecycle of the task execution.
- Queue – Queues are the primary forms of inter-task communications. In most cases they are used as thread safe first-in-first-out (FIFO) buffers.
- Semaphore – Threads use semaphores to control the access to shared resources.
- Software Timer – A software timer allows a function to be invoked at a predefined time. The timer callback functions are executed within the timer service task. It is essential to ensure the timer callback functions perform lightweight operations and return as quick as possible to avoid blocking the system resources.
- Event Group (enabled after FreeRTOS version 8.0.0) – An event group is a set of event bits. Individual event bits within an event group are referenced by a bit number. Event bits are used to indicate if an event has already occurred or not. Event groups can also be used to synchronize tasks.

For more information on FreeRTOS and its features, please refer to the official website.

## 2.2 Memory Usage

Brief details about memory usage can be found on FreeRTOS FAQ - Memory Usage, Boot Times & Context Switch Times website. Refer to the following sections:

- How much RAM does FreeRTOS use?
- How much ROM/Flash does FreeRTOS use?

## 2.3 Heap Service

heap_ext.c is used as the heap service on the SDK. The implementation is extended with more algorithms that are not supported by the official FreeRTOS. More details can be found in the header file `\kernel\rtos\FreeRTOS\SourceV10\include\portable.h`.
Table 3 describes some heap APIs.

*Table 3. Heap service APIs*

| Prototype | Description |
|---|---|
| `void *pvPortCalloc(size_t nmemb, size_t size)` | Provides the same functionality as the ISO C function `calloc()`.<br>Allocates memory for an array "`nmemb`" with elements in bytes and returns a pointer to the allocated memory.<br>The memory is set to zero.<br>This function is available in SDK v1 and later. |
| `void *pvPortRealloc(void *pv, size_t size)` | Provides the same functionality as the ISO C function `realloc()`.<br>Changes the size of the memory block pointed by "`pv`" to "`size`" in bytes.<br>The contents remains unchanged in the range from the start of the region up to the minimum of the old and new sizes. If the new size is larger than the old size, the added memory will not be initialized.<br>This function is available in SDK v1 and later. |
| `void *pvPortMallocNC(size_t xWantedSize)` | This function is similar to `pvPortMalloc()`.<br>Allocates "`xWantedSize`" bytes and returns a pointer to the allocated memory, except that the allocated memory is in non-cacheable region.<br>This function is available in SDK v1 and later. |
| `void vPortFreeNC(void *pv)` | This function is similar to `vPortFree()`.<br>Frees the memory space pointed by "`pv`". All cache related operations are directly implemented in the API and is transparent to users.<br>This function is available in SDK v1 and later. |

Heap is in cacheable region by default for performance critical operations, such as task stack. The functions `pvPortMallocNC()` and `vPortFreeNC()` are implemented for temporary non-cacheable memory requirements, such as when using DMAs.

For convenient migration of third-party software and libraries, the SDK also supports C standard library heap implementation, including `malloc()`, `calloc()`, `realloc()` and `free()`. However, these functions are wrapped in FreeRTOS heap service functions `pvPortMalloc()`, `pvPortCalloc()`, `pvPortRealloc()` and `vPortFree()`, and require building the project with the FreeRTOS module.

## 2.4 Examples

The source code and API documentation of the FreeRTOS can be found here.

## 2.5 Customization

You can provide custom configuration for the FreeRTOS in `FreeRTOSConfig.h` header file. The configurable parameters include OS tick frequency, maximum priorities, disabled functions, etc.
For more details, please refer to the online documentation.

## 2.6 Limitations

MediaTek does not impose any limitation on the integrated FreeRTOS. Please refer to the official online resources for more detailed information.

## 2.7 Developer Notes

- There is no software restriction on the number of tasks to create.
- The tasks can share the same priority.
- If the configuration `USE_PORT_OPTIMISED_TASK_SELECTION` is enabled, the maximum number of task priorities are 32 (0 to 31) in the ARM Cortex-M33 with floating point porting.
- Only API functions that end in "`FromISR`" can be called from within an interrupt service routine. Please refer to Open RTOS API documentation for more details.
- APIs that can potentially cause a context switch must not be called while the scheduler is suspended.
- APIs that can potentially cause a context switch must not be called from within a critical section.

# 3 TCP/IP: lwIP

TCP/IP (Transmission Control Protocol/Internet Protocol) is a communication internet protocol and can also be used in a private network, either an intranet or an extranet. It provides specifications on how data should be packetized, addressed, transmitted, routed, and received at the destination. The current IoT standard is moving towards IP communication and transporting data over various physical layers such as Wi-Fi, IEEE 802.15.4, and Bluetooth.

## 3.1 Features

lwIP is a widely used open-source TCP/IP stack designed for embedded systems. It includes the IP, ICMP, TCP, UDP, IGMP, ARP, AutoIP, DHCP, DNS, and SNMP protocols. The SDK provides the following supported features for these protocols.

- IPv4 (LWIP_IPV4).
- UDP (LWIP_UDP) – User Datagram Protocol, the widely adopted connectionless transmission protocol.
- TCP (LWIP_TCP) – Transmission Control Protocol, a widely used transport protocol providing reliable and in-order delivery.
- ARP (LWIP_ARP).
- ICMP (LWIP_ICMP).
- DHCP (LWIP_DHCP).
- DNS (LWIP_DNS).
- NETCONN (LWIP_NETCONN).
- Socket (LWIP_SOCKET).

## 3.2 Memory Usage

The MEM_SIZE parameter defines the size of the heap memory. PBUF_RAM stores the sent and received data. If the application requires more data to send, the value of the parameter must be set higher.

The values of different control blocks are configurable in lwIP. For example, MEMP_NUM_NETDB sets the concurrent Domain Name Resolution connections. Table 4 lists the current configuration values of these blocks in the SDK. These pools are configured and allocated from a buffer reserved only for the lwIP. The values are set to pass internal performance test and are expected to fulfill most common use cases. However, the number of configured control blocks could limit the maximum concurrent connections created by the network applications in the system. You can configure these values for a specific use case.

*Table 4. Number of control blocks and buffers in LwIP*

| Name | Current value |
|---|---|
| MEMP_NUM_UDP_PCB | 4 |
| MEMP_NUM_TCP_PCB | 8 |
| MEMP_NUM_TCP_PCB_LISTEN | 16 |
| MEMP_NUM_REASSDATA | 5 |
| MEMP_NUM_NETBUF | 2 |
| MEMP_NUM_NETCONN | 10 |
| MEMP_NUM_TCPIP_MSG_API | 8 |
| MEMP_NUM_TCPIP_MSG_INPKT | 8 |
| MEMP_NUM_SYS_TIMEOUT | 16 |
| MEMP_NUM_NETDB | 1 |
| MEMP_NUM_PBUF | 16 |
| PBUF_POOL_SIZE | 10 |

The required code size of lwIP is listed in Table 5. This information is gathered from an ARM Cortex M33 targeted configuration using the gcc -Os optimization. The feature set is IPv4, TCP, UDP, DHCP client, ICMP, RAW, NETCONN and Sockets and DNS client. The footprint is shown below.

*Table 5. Footprint of the LwIP*

| Static footprint | ROM (bytes) | RAM (bytes) |
|---|---|---|
| Debug release | 45541 | 51861 |

Note: If the option LWIP_DEBUG, LWIP_ERROR, LWIP_ASSERTS or LWIP_STATS is enabled, then the application has a significantly larger code footprint. Similarly, if the application is built with the compiler -O0 optimization flag on, the footprint is again significantly affected.

## 3.3 Examples

1) File: <sdk_root>/project/mt7933_hdk/apps/lwip_socket_demo/src/main.c.
2) Application Name:
    a) IPv4 UDP Client/Server test + IPv4 TCP Client/Server test,
    b) IPv6 UDP Client/Server test + IPv6 TCP Client/Server test by disabling MLD.
3) Application Overview. This is a reference application to demonstrate the usage of socket functions. The UDP can be used as a client or a server. When operating as a client, it can send a data packet to a remote UDP server. When operating as a server, it can receive data packets from remote UDP clients.
4) Similar tests can be carried out on TCP. When TCP operates as a client, it can connect to and communicate with a remote TCP server. When it operates as a server, it listens to and waits for

incoming connections from remote TCP clients. After a connection is established, it can communicate with the peer clients.

## 3.4 Customization

The custom settings and configuration can be applied in otp.h file located under <sdk_root>/middleware/third_party/lwip/src/include/lwip/. This file is fully commented and it is clear which options are defined, enabled, or disabled.

You can also customize the project settings in the lwipopts.h file located under <sdk_root>/project/mt7933_hdk/apps/<project>/include/lwipopts.h. For more information, please refer to introduction to lwIP on the Wiki website.

To enable or disable a feature, simply change the configuration parameters in lwipopts.h. For example, if you would like to disable DNS and enable DHCP, please modify or add the following lines in lwipopts.h header file.

//Disable DNS

```
#define LWIP_DNS 0
```

//Enable DHCP

```
#define LWIP  DHCP 1
```

Here are the major features in lwIP that can be customized (enabled or disabled):

LWIP_IPV4: Enables IPv4 protocol.

LWIP_IPV6: Enables IPv6 protocol. In this release, the SDK supports only IPv4. You can manually enable it. The basic functions work with IPv4/IPv6 dual stack support, but it cannot pass the IPv6 conformance test.

LWIP_UDP: Enables the UDP.

LWIP_TCP: Enables the TCP.

LWIP_RAW: The raw API is an event-driven API designed to be used without an operating system that implements zero-copy send and receive operation. However, the SDK suggests using BSD socket APIs (defined by LWIP_SOCKET) to make the socket application portable. If you are interested in the raw API, please refer to the lwIP native API documentation for more details.

LWIP_ARP: Enables the ARP functionality.

LWIP_ICMP: Enables the ICMP module in the IP stack.

LWIP_IGMP: Enables the IGMP module (multicast support).

LWIP_SNMP: Enables the lwIP SNMP agent. The UDP must be available for the SNMP transport.

LWIP_DHCP: Enables the DHCP client module.

LWIP_AUTOIP: Enables the AUTOIP module and the RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses.

LWIP_DNS: Enables the DNS module and requires UDP for DNS transport.

LWIP_NETCONN: Enables Netconn API; requires using api_lib.c source file.

LWIP_SOCKET: Enables Socket API; requires using sockets.c to include a set of APIs compatible with

POSIX-/BSD sockets.
LWIP_STATS: Enables statistics collection in lwip_stats.
LWIP_DEBUG: Enables debugging.
LWIP_ERROR: Enables error logging.
LWIP_NOASSERT: Disables LWIP_ASSERT checks.

## 3.5 Limitations

The list of feature limitations for lwIP is provided below.

- Only supports a few ICMP packet types, such as echo reply, destination unreachable, and time exceeded. Most of the other types that are not specific to the embedded systems are ignored.
- Does not support Network Address Translation (NAT) to map IP address space into another address for forwarding the IP packets.
- Few of TCP/IP options are supported in lwIP.

## 3.6 Developer Notes

Applications should invoke recv() API repeatedly until it returns EWOULDBLOCK indicating there is no pending data in the receiving buffer. Otherwise, more and more incoming data will exhaust the buffer.
lwIP supports both blocking and non-blocking methods in the SDK. By default, sockets are blocking, indicating that if a socket call is issued, it cannot be completed immediately. For example, in the TCP three-way handshaking process, the caller process is put to sleep while waiting for the condition to be true. With non-blocking socket operation, the socket function call can return immediately and the application can proceed with other operations. Non-blocking I/O is suitable for single-thread socket applications.

# 4 SSL/TLS: mbedTLS

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL) are cryptographic protocols designed to provide secure communication over the computer network. TLS and SSL use X.509 certificates and asymmetric cryptography to authenticate secure data communication and to negotiate the process with a symmetric session key that ensures message confidentiality.
mbed TLS offers libraries including SSL/TLS cryptographic communication capabilities for (embedded) devices and applications that provide end-to-end communication protection to the upper layer application protocols such as web browsing, email, instant messaging, and voice-over-IP (VoIP).
Starting from version 2.1.0, mbedTLS is released under Apache 2.0 License and enables you to use mbed TLS in both open source and closed source projects.
In MT793X IOT SDK 3.1, MTK upgraded mbedTLS to mbedTLS v3.2.1 for TLS1.3 support. For more detail, you can see mbed TLS 3.2.1.

## 4.1 Features

mbed TLS is an open-source and commercial SSL library licensed under Arm Limited. This library easily integrates with new and existing (embedded) devices and applications and provides the building blocks for secure communication, cryptography, and key management. Both the client-side and the server-side APIs support current SSL and TLS standards (i.e. SSL version 3.0, TLS version 1.0, TLS version 1.1 and TLS version 1.2).
The cryptographic algorithms enabled in the SDK include:

1) Symmetric encryption algorithms: AES, Triple-DES (3DES), DES, ARC4.
2) Modes of operations: Cipher Block Chaining Mode (CBC).
3) Hash algorithms: MD5, SHA-1, and SHA-256.
4) RSA/PKCS#1 v1.5.
5) Random number generation: CTR_DRBG.

## 4.2 Memory Usage

None.

## 4.3 Examples

mbed TLS is equipped with test cases. Developers can use them as a reference to develop and learn how to use the APIs. A few example applications can be found under <sdk_root>/project/mt7933_hdk/apps/mbedtls/ directory.

## 4.4 Customization

The default configuration file for mbed TLS is config-mtk-homekit.h in the SDK release. The file is located under <sdk_root>/middleware/third_party/mbedtls/configs/config-mtk-homekit.h. Developers can provide their

custom configuration file under <sdk_root>/middleware/third_party/mbedtls/configs and set the file name to the variable MTK_MBEDTLS_CONFIG_FILE in project's feature.mk makefile such as <sdk_root>/project/mt7933_hdk/apps/mbedtls/GCC/feature.mk.
Define MBEDTLS_DEBUG_C in the configuration file to enable debugging and error handling.
You can switch the feature options in the configuration file and define some of the values or parameters. The configuration file is well documented and you can refer to the comments in <sdk_root>/middleware/third_party/mbedtls/include/mbedtls/config.h.

## 4.5 Limitations

Due to the limitation of ARM Cortex-M33 with floating point computational power, the TLS handshake at server side takes more than 10 seconds with RSA 2048-bit public key length. Therefore, the SDK disables the server option in the suggested configuration file.

### 4.5.1 CVE ID: CVE-2021-43666

Description:
A Denial of Service vulnerability exists in mbed TLS 3.0.0 and earlier in the mbedtls_pkcs12_derivation function when an input password's length is 0.
This CVE is published on Mar 25, 2022.

### 4.5.2 CVE ID: CVE-2021-45451

Description:
In Mbed TLS before 3.1.0, psa_aead_generate_nonce allows policy bypass or oracle-based decryption when the output buffer is at memory locations accessible to an untrusted application.
This CVE is published on Dec 21, 2021.

### 4.5.3 Solution

MTK upgrades mbedtls from 2.28.0 to 3.2.1. mbedtls v3.2.1 has no CVE and supports TLS1.3.

## 4.6 Developer Notes

1) In order to keep mbed TLS thread safe, it is important to keep a few things in mind.
2) Most functions use an explicit context. As long as the context is not shared among the threads, the application is thread safe. However, sometimes a context is shared indirectly. For example, an SSL context can point to an RSA context (the private key).
3) The rule of thumb is that a context should only be used or accessed by a single thread at a time, unless:
   a) The function is documented explicitly that it is thread safe to access the shared context, or
   b) You have applied an explicit locking mechanism, such as a mutex, to protect a critical section through a wrapper function.
4) Client verifies the server's identity with the received certificate during the handshake operation. Typically the received server certificate is composed of three-level hierarchy. If the length is larger, the TLS client will require more heap space for verification process.

5) Complete the following procedure to configure the trusted root certificates.
   a) Since the SDK doesn't support File system, you can only keep the trusted certificate authentications (CAs) in memory and parse them by calling the mbedtls_x509_crt_parse() function.
   b) If there are several trusted CAs, invoke the mbedtls_x509_crt_parse() function several times to set them to the trusted CA chain one by one.
   c) Set authorization mode to NONE, OPTIONAL or REQUIRED, by calling the mbedtls_ssl_conf_authmode() function. If REQUIRED option is selected, the handshake operation will fail once the certificate verification fails.
   d) If the function mbedtls_ssl_set_hostname() is called, the client compares the hostname with the common name of the server certificate in the certification verification process.

# 5 File System: FatFs

FatFs is a generic FAT file system that manages the access to storage devices for small embedded systems. FatFs is written in compliance with ANSI C (C89) and is completely separated from the disk I/O layer.

## 5.1 Features

To facilitate the data (files or directories) management in the storage device, FatFs provides the following features:

- Windows OS compatible FAT file system.
- Platform independent, easy to port to any target platform.
- Small footprint for code and work area (file system objects, file objects, etc.).
- Long file name support in ANSI/OEM or Unicode.
- RTOS support for multi-tasking.
- Multiple sector size support up to 4 KB.
- Read-only, optional API, I/O buffer and other features.

More information on the FatFs and its features can be found here.

## 5.2 Memory Usage

The memory usage varies depending on the configuration options, as described in section 5.4, “Customization”.

The FatFs module provides several compilation options to disable unused sub-modules. Overall, according to the configuration in the SDK release, the ROM size of the current release is 12466 bytes, the RAM size is 518 bytes. The memory size may be different for different versions of the SDK.

## 5.3 Examples

The source code and API documentation of the FatFs can be found at official website.

## 5.4 Customization

You can customize the configuration for the FatFs in the `ffconf.h` header file. The configurable parameters include the volume to support, the sector size, etc.
More details can be found in the online documentation.

## 5.5 Limitations

The list of feature limitations for the FatFs is provided below.

- Multiple sector size support up to 4 KB.
- FAT sub-types - FAT12, FAT16 and FAT32.
- Number of open files - unlimited depending on the available memory.
- Number of volumes - up to 10.
- File size - up to 4 GB minus 1 byte. (FAT specifications.)
- Volume size - up to 2 TB at 512 bytes per sector. (FAT specifications.)
- Cluster size - up to 64 KB at 512 bytes/sector. (FAT specs.)
- Sector size - 512, 1024, 2048 and 4096 bytes. (FAT specs.)

## 5.6 Developer Notes

FatFs integrated in current SDK version only supports file system on the SD card (Secure Digital Memory Card) or eMMC (Embedded Multi Media Card) without flash.

- FatFs is not enabled by default in order to provide the flexibility to select whether to use this file system or a different file systems.

# 6 CMSIS

The CMSIS is a vendor-independent hardware abstraction layer for the Cortex-M processor series and defines generic tool interfaces. The CMSIS enables consistent device support and simple software interfaces to the processor and the peripherals, real-time operating systems and middleware components simplifying software re-use, reducing the learning curve for microcontroller developers and reducing the time to market for new devices.

The CMSIS is intended to enable the combination of software components from multiple middleware vendors.

## 6.1 Features

The CMSIS components include:

- CMSIS-CORE – Implements basic run-time APIs for a Cortex-M device and provides convenient access to the processor core and the device peripherals.
- CMSIS-DSP – This library provides a suite of common signal processing functions to apply on Cortex-M processor based devices.

MediaTek IoT development platform only includes CMSIS-CORE and CMSIS-DSP.

## 6.2 Memory Usage

CMSIS-CORE APIs are implemented in header files, so the memory usage is added to the source file. The library size of the CMSIS-DSP is up to 5.4 MB and the actual memory usage depends on which and how many APIs are in use.

## 6.3 Examples

CMSIS API examples can be found on official website.

## 6.4 Customization

No customization or configuration is required to use CMSIS-CORE and CMSIS-DSP. Apply them directly in your implementation.

## 6.5 Limitations

None.

## 6.6 Developer notes

MediaTek IoT Development Platform only includes CMSIS-CORE and CMSIS-DSP.

Visit the official Arm website to find more detailed information about CMSIS.

# 7 Micro-ecc

A small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors.

## 7.1 Features

- Resistant to known side-channel attacks.
- Written in C, with optional GCC inline assembly for AVR, ARM, and Thumb platforms.
- Support for 8-bit, 32-bit, and 64-bit architectures.
- Small code size.
- No dynamic memory allocation.
- Support for five standard curves: secp160r1, secp192r1, secp224r1, secp256r1, and secp256k1.
- BSD 2-clause license.

## 7.2 Memory Usage

None

## 7.3 Examples

None

## 7.4 Customization

None

## 7.5 Limitations

None

## 7.6 Developer Notes

None

# 8 HTTP (1.1) client: mbed HTTP Client

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is a structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

HTTP/1.1 is the most commonly used version of HTTP and was defined by RFC 2616 in 1999.

## 8.1 Features

HTTPClient implements the client-side of HTTP/1.1. It provides base interfaces to send HTTP requests and receive HTTP responses from a resource identified by a URI. It also supports HTTPS (HTTP over SSL/TLS) to provide secure communication.

## 8.2 Memory Usage

The static footprint statistics for HTTP client are shown in Table 6. Please note that the RAM size is 0 but it allocates the required buffer from system heap during application execution.

***Table 6. Footprint of the HTTP 1.1 client***

| | ROM (KB) | RAM, static analysis (KB) |
|---|---|---|
| Debugging disabled | 3.5 | 0 |
| Debugging enabled | 7.1 | 0 |

## 8.3 Examples

None.

## 8.4 Customization

Here are the configurable parameters for the HTTP Client:

Set the macro `HTTPCLIENT_DEBUG` to 1 in `<sdk_root>/middleware/third_party/httpclient/inc/httpclient.h` to enable debugging.

Switch `MTK_HTTPCLIENT_SSL_ENABLE` in the project configuration file (for example, in the GCC/feature.mk) to enable SSL to secure HTTP messages.

## 8.5 Limitations

Only high level APIs are currently provided. You must modify or use the internal functions in `<sdk_root>/middleware/third_party/httpclient/src/httpclient.c` to perform advanced configuration control over the HTTP link.

## 8.6 Developer Notes

The HTTP client supports concurrent requests. However, the number of concurrent connections is limited by the heap space, socket configuration and DNS slots in lwIP.

The peak heap usage for each HTTP over SSL connection process takes about 25 KB for the certificate verification process during handshake.

# 9 SNTP: lwIP-contrib SNTP

Simple Network Time Protocol (SNTP) is a less complex implementation of NTP (Network Time Protocol). It is used in embedded devices and in applications where high accuracy timing is not required. For more details about SNTP, please refer to RFC 4330.

## 9.1 Features

This SNTP implementation is derived from the `lwip-contrib` application projects, which can query the current time from a given server and set the acquired time to system clock. The SDK enables DNS server address, and multiple servers are supported in the release.

## 9.2 Memory Usage

SNTP is a small utility in the lwIP package. The memory footprint is shown in Table 7.

***Table 7. Footprint of SNTP***

| | ROM (KB) | RAM, static analysis (bytes) |
|---|---|---|
| Debugging enabled | 2 | 49 |
| Debugging disabled | 1 | 49 |

## 9.3 Examples

File: `<sdk_root>/project/mt7933_hdk/apps/lwip_socket_demo/src/main.c`.

1) Application Macro: __SNTP_DEMO__.
2) Application Overview: The application is a reference usage of the SNTP. It receives GMT time from an SNTP server and sets system time by calling `sntp_init()` function.

## 9.4 Customization

1) The source configuration file is located under `<sdk_root>/middleware/third_party/lwip/src/apps/sntp/sntp.c`. The header configuration file is located under `<sdk_root>/middleware/third_party/lwip/src/include/lwip/apps/sntp_opts.h`. The parameters that can be customized or configured are shown below.
2) Define SNTP_DEBUG; use LWIP_DBG_ON to enable debugging; use LWIP_DBG_OFF to turn off debugging.
3) SNTP_RECV_TIMEOUT means SNTP receives timeout and the default value is three seconds. The default value of SNTP_UPDATE_DELAY is one hour, which is the update interval.
4) SNTP_SET_SYSTEM_TIME can be defined by you to set the current system time.

5) SNTP_MAX_SERVERS defines the number of SNTP servers to connect to. The current default value is five to indicate the maximum number of available servers in this SDK release. This value can be more than one. The larger the number, the more RAM the application needs.

## 9.5 Limitations

This implementation supports only unicast mode and does not support broadcast and multicast modes.

## 9.6 Developer Notes

None.

# 10 LZMA: LZMA Decoder

The LZMA is an algorithm performing lossless data compression. The LZMA SDK provides a high compression ratio and fast decompression. The MediaTek IoT SDK only uses the LZMA decoder algorithm.

## 10.1 Features

The LZMA decoder provides functions to decompress the compressed data encoded by LZMA encoder. LZMA decoder has the following features:

- Small memory requirements: 8 to 32 KB + Dictionary Size.
- Small code size: 2 to 8 KB, depending on speed optimizations.

More information on the LZMA and its features can be found here.

## 10.2 Memory Usage

The memory usage is shown in Table 8.

***Table 8. LZMA decoder memory usage***

| | ROM (bytes) | RAM, static analysis (bytes) |
|---|---|---|
| Enable Debugging | 7182 | 40*1024<br>(suggested decoding buffer) |
| Disable Debugging | 6708 | 40*1024<br>(suggested decoding buffer) |

## 10.3 Examples

The LZMA decoder module is used to decode FOTA package file in bootloader only. Because Cortex-M33 binary size is usually too large to have enough buffer for the whole file data, use the wrapper function `lzma_decode2flash()` provided in `<sdk_dir>/middleware/third_parity/lzma_decoder/inc/lzma_decoder_interface.h` header file to decode data block by block, and then call HAL flash API to write the output data to the specified address on NOR flash.

## 10.4 Customization

None.

## 10.5 Limitations

None.

## 10.6 Developer Notes

The LZMA decoder integrated in current version of the SDK only supports bootloader decoding.

# 11 JSON: cJson

JSON (JavaScript Object Notation) is a lightweight data interchange format. It is convenient for humans to read and write and for the machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition. JSON is a text format that is completely language independent but uses conventions similar to C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

JSON is built in two structures:

1) A collection of name or value pairs. In various languages, this is implemented as an object, record, structure, dictionary, hash table, keyed list, or associative array.
2) An ordered list of values. In most languages, this is implemented as an array, vector, list, or sequence.

## 11.1 Features

cJSON is an ultra-lightweight, portable, single-file, simple-as-can-be ANSI-C compliant JSON parser under MIT license. It's a single C source file along with its header file.

## 11.2 Memory Usage

cJSON is a simple JSON parser with a small footprint. The operational buffers are allocated from the system heap. The details are shown in Table 9.

***Table 9. cJson memory usage***

| | ROM (bytes) | RAM, static analysis (bytes) | Heap Size |
|---|---|---|---|
| cJson | 6312 | 8 | It varies depending on the parsed content, ranging from 800 bytes to 2200 bytes. |

## 11.3 Examples

`File: <sdk_root>/project/mt7933_hdk/apps/cjson/src/main.c.`

1) Application Name: cJSON Test
2) Application Overview: The application is a string parser using cJSON's APIs. The parsed string is stored in a struct. Developers can refer to this simple application and reuse the functions in their own applications.

## 11.4 Customization

None.

## 11.5 Limitations

Standard C stdio.h is not supported in the SDK because ARM Cortex-M33 with floating point is a lightweight embedded system. The floating-point output is not as accurate as in a standard library. However, it is capable of handling common use cases such as representing longitude and latitude in a geographic coordinate system.

## 11.6 Developer Notes

1) -fsingle-precision-constant might cause an unexpected result when handling floating point data. The -fsingle-precision-constant compile option is used to improve performance due to less memory traffic by loading floating point constants in single precision format. This option also uses single precision constants in operations on double precision variables. For more information, please refer to semantics of floating point math in GCC Wiki.

   - For example, if the option is set while declaring a variable of type double, such as

     ```
     double number = 987654321.23;
     ```

     it is compiled into 987654336 for single floating point optimization.

# 12 MQTT Client: Paho Embedded MQTT C/C++ Client

MQTT is a lightweight publish/subscribe messaging protocol originally created sometime around 1998 by IBM and Arcom (later to become part of Eurotech). More information about the protocol can be found on the mqtt.org community website.

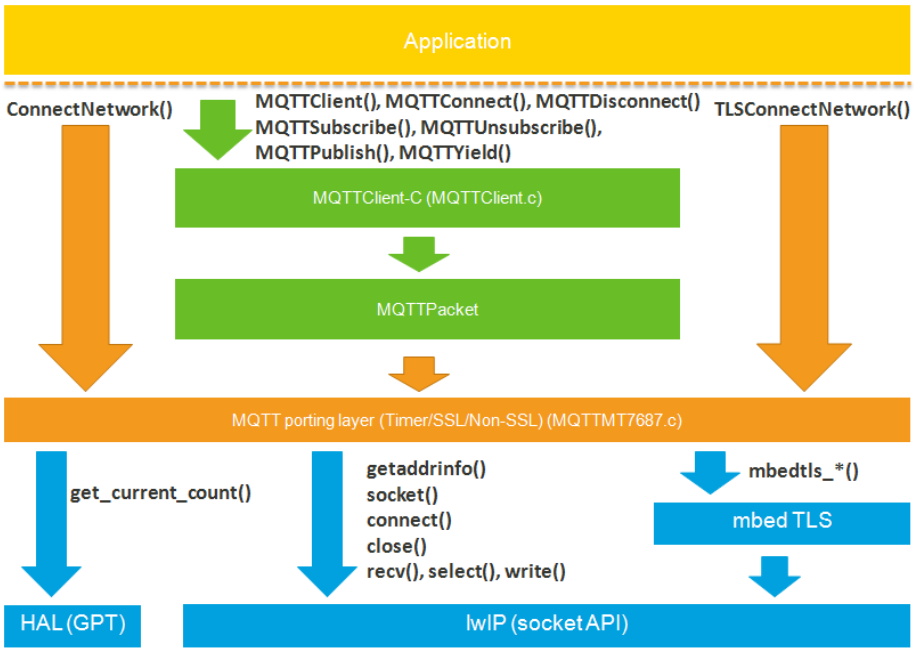
## 12.1 Features

The Paho project is created to provide scalable open-source implementations of open and standard messaging protocols aimed at new, existing and emerging applications for M2M and IoT devices and applications. Paho reflects the inherent physical and cost constraints of device connectivity. Its objectives include effective levels of decoupling between devices and applications, designed to keep markets open and encourage the rapid growth of scalable Web and Enterprise middleware and applications. Paho supports MQTT publish/subscribe client implementations on embedded platforms, along with corresponding server support as determined by the community.

In this release the Paho MQTT library is integrated with mbed TLS to widen the connectivity support of various types including:

- Unencrypted
- Encrypted, CA certificate required
- Encrypted, CA/Client certificate required

A new porting layer is designed to port the library to Mediatek IoT Development Platform, based on mbed TLS, lwIP and Mediatek Hardware Abstraction Layer (HAL) API. The porting architecture is shown in Figure 3. For each MQTT application a network connection is established using ConnectNetwork() or TLSConnectNetwork() APIs that are implemented in the platform-dependent porting layer. Then use MQTTClient() to initiate a MQTT client instance, followed by MQTTConnect(), MQTTSubscribe() or MQTTPublish() to interact with servers. The MQTT*() APIs implemented in the green blocks in the diagram are platform independent.



## 12.2 Memory Usage

MQTT was originally designed to be a lightweight transport protocol for M2M communication. Please refer to Table 10 for the required ROM/RAM size.

***Table 10. MQTT memory usage***

| | ROM (bytes) | RAM, static analysis (bytes) |
|---|---|---|
| Enable Debugging | 11805 | 20 |
| Disable Debugging | 8601 | 4 |

## 12.3 Examples

`File: <sdk_root>/project/mt7933_hdk/apps/mqtt_client/src/main.c.`

1) Application Name: mqtt client
2) Application Overview: The application is based on the MQTT functions implementing connectivity with an unencrypted or encrypted MQTT server. Developers can refer to this simple application and re-use the functions in their own applications.

## 12.4 Customization

Use MTK_MQTT_DEBUG to enable or disable the logs. Define it in `<sdk_root>/middleware/third_party/mqtt/MQTTClient-C/src/FreeRTOS/MQTTFreeRTOS.h.`

## 12.5 Limitations

### 12.5.1 CVE ID: CVE-2021-41036

Description:

In versions prior to 1.1 of the Eclipse Paho MQTT C Client, the client does not check rem_len size in readpacket.

### 12.5.2 Solution

MTK don’t support Eclipse License, so we do not have newer MQTT source codes to solve this CVE. Therefore, MTK will not offer the patch for this CVE. Users should use this software carefully to avoid the unpredicted results. We suggest users to upgrade MQTT to v1.1.0 for long term usage.

## 12.6 Developer Notes

None.

# 13 Websocket: librws

WebSocket protocol is a TCP-based protocol, which provides full-duplex communication channels for the server-client communication. Unlike HTTP, the WebSocket server can send content without being solicited by the client. The new URI schemes, ws and wss, are defined for WebSocket unencrypted and encrypted connections, separately.

Librws is a small cross-platform WebSocket client library in C, released under the MIT license. Mediatek provides an improved secure WebSocket based on mbed TLS.

## 13.1 Features

Librws provides the following features:

1) No additional dependencies
2) Single header library interface librws.h with public methods
3) Thread safe
4) Send/receive logic in background thread

More information on the librws and its features can be found here.

## 13.2 Memory Usage

The memory usage is shown in Table 11.

***Table 11. Librws memory usage***

| | ROM (bytes) | RAM, static analysis (bytes) |
|---|---|---|
| Disable Secure WebSocket | 8843 | 20 |
| Enable Secure WebSocket | 10822 | 20 |

## 13.3 Examples

`File: <sdk_root>/project/mt7933_hdk/apps/websocket_client/src/main.c.`

1) Application Name: websocket_client
2) Application Overview: The application is a demonstration of the WebSocket client. The developers can reuse the functions in their own applications.

## 13.4 Customization

- Mediatek provides an implementation to support the secure WebSocket based on mbed TLS. To enable secure WebSocket, mbed TLS module should be included and "MTK_WEBSOCKET_SSL_ENABLE = y" should be added into the GCC project makefile, such as feature.mk.
- The file config-mtk-websocket.h, is used for the secure WebSocket as the mbed TLS configuration file, by default. Developers can customize and use their own mbed TLS configuration files based on the specific requirements of the WebSocket servers.

## 13.5 Limitations

None.

## 13.6 Developer Notes

None.

# 14 XML: Mini-XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format, which is both human and machine-readable. It is defined by the W3C's XML 1.0 specification and by several other related specifications, all of which are free open standards. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

## 14.1 Features

The Mini-XML module is a small XML library that enables parsing the XML and XML-like data in the application without requiring large non-standard libraries. It supports reading of UTF-8 and UTF-16 and writing of UTF-8 encoded XML strings. Data is stored in a linked-list tree structure, preserving the XML data hierarchy and arbitrary element names, attributes and attribute values are supported with no preset limits, just available memory.

## 14.2 Memory Usage

The XML library of the SDK has two build types, full and basic with optimized and reduced memory footprint. The XML module provides several compilation options to disable unused sub-modules. Overall, the ROM size of most basic version is 13052 bytes. It is 21940 bytes if all of the sub-modules are enabled, see section 6.4, “Customization”.

## 14.3 Examples

1) File: <sdk_root>/project/mt7933_hdk/apps/xml/src/main.c.
2) Application Name: XML sample application.
3) Application Overview: The sample application is a reference to parse a typical XML file by means of this XML library. The developers can refer to this simple application and reuse the functions in their applications.

## 14.4 Customization

The XML module provides compile options to enable or disable sub-features. Table 9 shows the details.

| Compilation option | Function | ROM size (bytes) |
|---|---|---|
| MXML_SUPPORT_ENTITY | Convert the reserved characters in XML to corresponding character entities. | 4584 |
| MXML_SUPPORT_GET_FUNCTIONS | Get the attribute and content of an element | 892 |
| MXML_SUPPORT_SET_FUNCTIONS | Set the attribute and content of an element. | 1040 |
| MXML_SUPPORT_INDEX | Create the index with all elements, the elements are sorted by element name or attribute value. | 1056 |
| MXML_SUPPORT_SEARCH | Find the element by name. | 812 |

## 14.5 Limitations

For the platforms that don’t support file system, such as LinkIt 7687 development board, the file system related API sets of mxml*Fd() and mxml*File() are not accessible, and the XML objects can only be loaded and saved by mxml*String() APIs.

## 14.6 Developer Notes

None.

# Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.