# TELIT'S APPZONE USER GUIDE

1vv0301082 Rev.4 2015-05-07

# APPLICABILITY TABLE

## PRODUCTS

| | | |
|---|---|---|
| ■ | ■ | GE910-QUAD |
| ■ | ■ | GE910-GNSS |
| ■ | ■ | UE910 SERIES |
| ■ | ■ | HE910 SERIES |

# SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

## LEGAL NOTICE

These Specifications are general guidelines pertaining to product selection and application and may not be appropriate for your particular project. Telit (which hereinafter shall include, its agents, licensors and affiliated companies) makes no representation as to the particular products identified in this document and makes no endorsement of any product. Telit disclaims any warranties, expressed or implied, relating to these specifications, including without limitation, warranties or merchantability, fitness for a particular purpose or satisfactory quality. Without limitation, Telit reserves the right to make changes to any products described herein and to remove any product, without notice.

It is possible that this document may contain references to, or information about Telit products, services and programs, that are not available in your region. Such references or information must not be construed to mean that Telit intends to make available such products, services and programs in your area.

## USE AND INTELLECTUAL PROPERTY RIGHTS

These Specifications (and the products and services contained herein) are proprietary to Telit and its licensors and constitute the intellectual property of Telit (and its licensors). All title and intellectual property rights in and to the Specifications (and the products and services contained herein) is owned exclusively by Telit and its licensors. Other than as expressly set forth herein, no license or other rights in or to the Specifications and intellectual property rights related thereto are granted to you. Nothing in these Specifications shall, or shall be deemed to, convey license or any other right under Telit's patents, copyright, mask work or other intellectual property rights or the rights of others.

You may not, without the express written permission of Telit: (i) copy, reproduce, create derivative works of, reverse engineer, disassemble, decompile, distribute, merge or modify in any manner these Specifications or the products and components described herein; (ii) separate any component part of the products described herein, or separately use any component part thereof on any equipment, machinery, hardware or system; (iii) remove or destroy any proprietary marking or legends placed upon or contained within the products or their components or these Specifications; (iv) develop methods to enable unauthorized parties to use the products or their components; and (v) attempt to reconstruct or discover any source code, underlying ideas, algorithms, file formats or programming or interoperability interfaces of the products or their components by any means whatsoever. No part of these Specifications or any products or components described herein may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without the prior express written permission of Telit.

## HIGH RISK MATERIALS

Components, units, or third-party products contained or used with the products described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (“High Risk Activities"). Telit, its licensors and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

## TRADEMARKS

You may not and may not allow others to use Telit or its third party licensors’ trademarks. To the extent that any portion of the products, components and any accompanying documents contain proprietary and confidential notices or legends, you will not remove such notices or legends.

## THIRD PARTY RIGHTS

The software may include Third Party Right software. In this case you agree to comply with all terms and conditions imposed on you in respect of such separate software. In addition to Third Party Terms, the disclaimer of warranty and limitation of liability provisions in this License shall apply to the Third Party Right software.

TELIT HEREBY DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED FROM ANY THIRD PARTIES REGARDING ANY SEPARATE FILES, ANY THIRD PARTY MATERIALS INCLUDED IN THE SOFTWARE, ANY THIRD PARTY MATERIALS FROM WHICH THE SOFTWARE IS DERIVED (COLLECTIVELY “OTHER CODE”), AND THE USE OF ANY OR ALL THE OTHER CODE IN CONNECTION WITH THE SOFTWARE, INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF SATISFACTORY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE.

NO THIRD PARTY LICENSORS OF OTHER CODE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND WHETHER MADE UNDER CONTRACT, TORT OR OTHER LEGAL THEORY, ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE OTHER CODE OR THE EXERCISE OF ANY RIGHTS GRANTED UNDER EITHER OR BOTH THIS LICENSE AND THE LEGAL TERMS APPLICABLE TO ANY SEPARATE FILES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

# CONTENTS

# FIGURES

# TABLES

# 1 INTRODUCTION

## 1.1 Scope

The present document provides the guidelines to install and use the Telit's AppZone SDK (called also Telit Application Development Environment (ADE)) to create a M2M application. In addition, it describes the +M2M AT commands used to configure the module and manage the M2M applications uploaded into the module. Appendix 8.1 collects the set of all M2M AT commands and describes their syntax. Chapter 5 describes the Telit AT Controller, it is a console integrated inside the ADE tool. Its objective is to improve the interface toward the module.

## 1.2 Audience

This User Guide is intended for those users who need to develop a custom application, and run it on the Telit's module as an embedded application that uses the GPRS/HSPA services provided by the module itself.

## 1.3 Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at:

TS-EMEA@telit.com<br>
TS-AMERICAS@telit.com<br>
TS-APAC@telit.com

Alternatively, use:

http://www.telit.com/support

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

## 1.4 Text Conventions

STOP

Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.

⚠

Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.

ℹ

Tip or Information – Provides advice and suggestions that may be useful when integrating the module.

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

## 1.5 Related Documents

[1] UE910 Hardware User Guide, 1vv0301012
[2] AppZone APIs User Guide, 1vv0301130
[3] HE910 Hardware User Guide, 1vv03700925
[4] GE910 Hardware User Guide, 1vv0300962
[5] File System Tool User Guide, 1vv0301192

ℹ

In the present guide is used the notation [x]/[y] to refer to documents of different families of modules. You have to refer to the document in accordance with the module you are using.

# 2 APPZONE OVERVIEW

The AppZone is a software layer that provides a set of APIs to access the modem features of major operational interest. By means of the AppZone, the user M2M application runs on the module CPU; this solution does not require an external application processor.

The user can develop M2M applications that can address a wide range of different applications such as remote monitoring and control, security and surveillance, telemetry, location services, billing, fleet management, etc. The generic user application can access the following modem resources:

| | |
|---|---|
| • Operating System: | Signals, Semaphores, Timers, Dynamic Memory Management, etc. |
| • HW/SW: | GPIO, I2C, UART, SPI, Keypad, File-System, RTC, etc. |
| • GSM/GPRS: | Communication services. |
| • Networking: | BSD socket support, SSL capabilities. |

Refer to documents [1]/[3]/[4] for information on module hardware resources.

Fig. 1 shows the high-level architecture of a module provided with the AppZone software layer. The user to develop the custom M2M applications, in accordance with the AppZone layer, should use the Telit Application Development Environment (ADE) - refer to chapter 3 - that provides a set of files called skeleton.

The skeleton factory default configuration includes only one M2M_procX.c template file, named M2M_proc1.c and containing the M2M_msgProc1() callback function, see chapter 3.2.2. The user may write new M2M_procX.c files, new M2M_msgProcX() callbacks, and create new AppZone Tasks. AppZone layer can support up to 32 tasks in total. In general, each AppZone Task calls its callback function, for more information refer to document [2]:

- AppZone Task_1 calls M2M_msgProc1 (default)
- AppZone Task_2 calls M2M_msgProc2 (created by the user)
- AppZone Task_3 calls M2M_msgProc3 (created by the user)
- AppZone Task_... calls M2M_msgProc… (created by the user)

The programmer uses the M2M_msgProcX() callback functions to develop the M2M applications. Tasks communicate with each other exchanging messages using the suitable API.

Use the C programming language to develop the M2M applications.

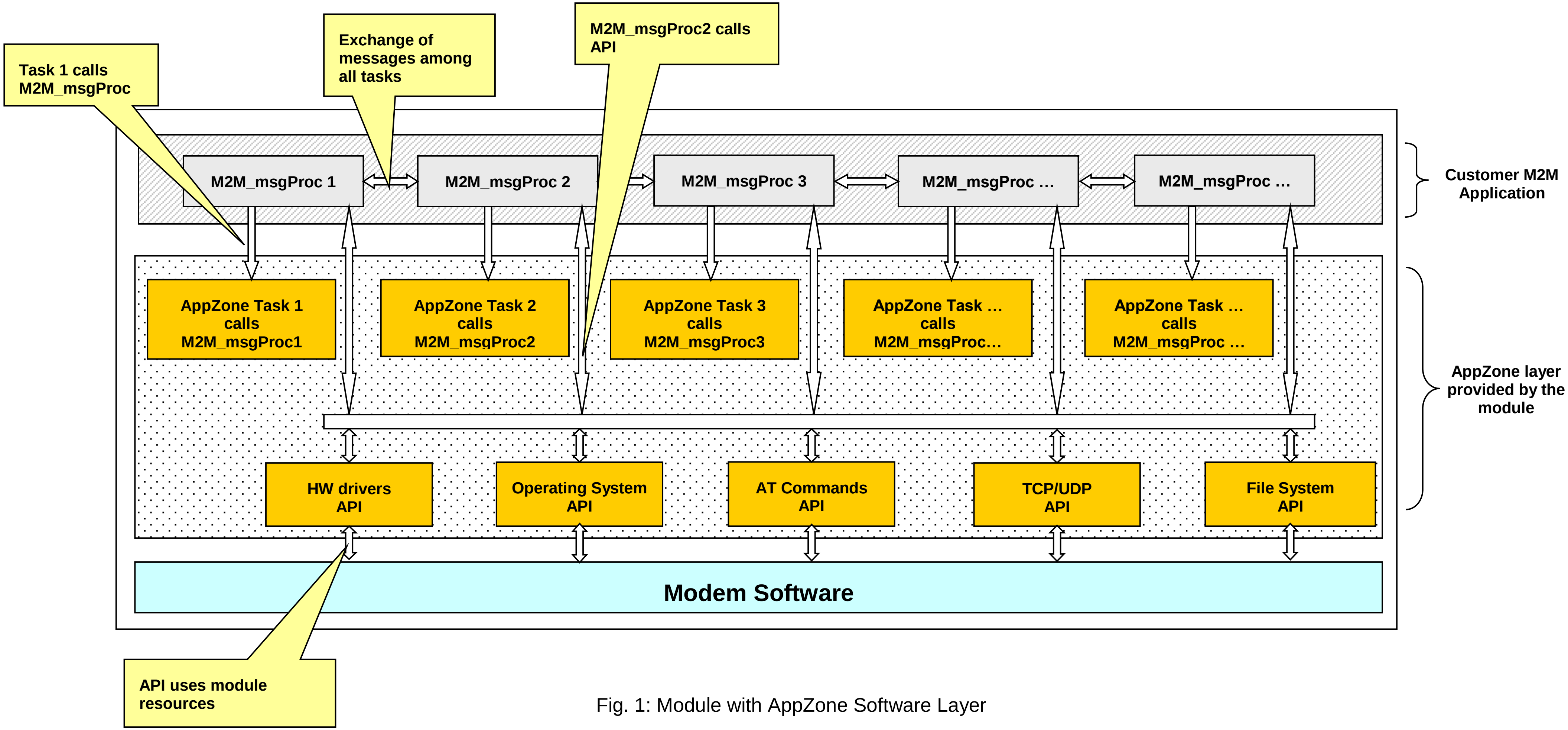

Task 1 calls M2M_msgProc
Exchange of messages among all tasks
M2M_msgProc2 calls API
M2M_msgProc 1
M2M_msgProc 2
M2M_msgProc 3
M2M_msgProc …
M2M_msgProc …
Customer M2M Application
AppZone Task 1 calls M2M_msgProc1
AppZone Task 2 calls M2M_msgProc2
AppZone Task 3 calls M2M_msgProc3
AppZone Task … calls M2M_msgProc…
AppZone Task … calls M2M_msgProc …
AppZone layer provided by the module
HW drivers API
Operating System API
AT Commands API
TCP/UDP API
File System API
Modem Software
API uses module resources

Fig. 1: Module with AppZone Software Layer

## 2.1 AppZone vs. Application Processor

The figures below show two generic examples of hardware/software configurations that point out the differences between the use of a module providing the AppZone Software Layer and another not providing that feature.

Fig. 2 shows a configuration where the user application is running on an external application processor. The user application uses the GPRS/HSPA services and User Device #2 by means of the “A” interface, e.g. a serial port.

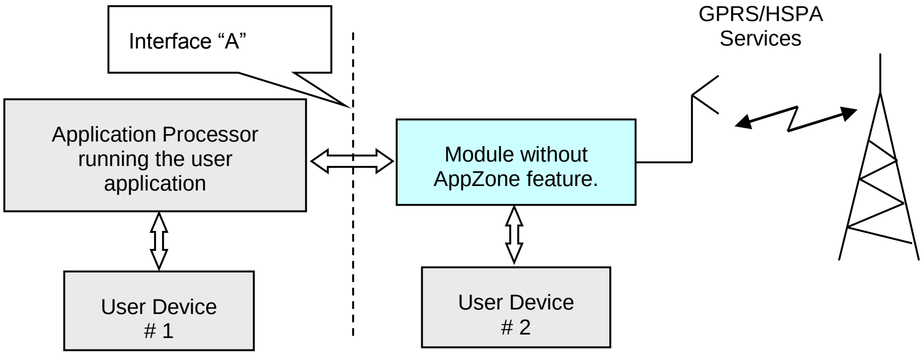

Interface “A”
GPRS/HSPA Services
Application Processor running the user application
Module without AppZone feature.
User Device # 1
User Device # 2

Fig. 2: Module without AppZone Layer

Fig. 3 shows a configuration where the module provides the AppZone layer. In this scenario, the user application is running on the module CPU. The user application uses the GPRS/HSPA services, User Device #1, and #2 by means of the set of AppZone layer interfaces (APIs). The external user application processor is not required!

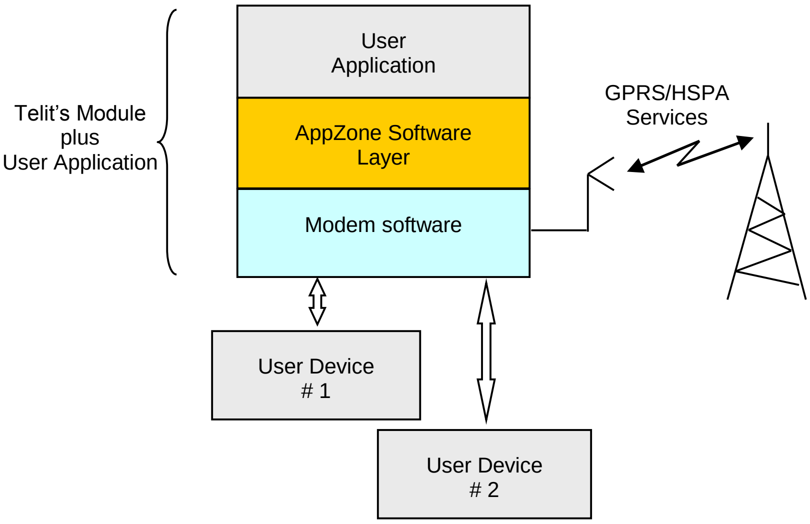


Fig. 3: Module with AppZone Layer

# 3 APPZONE SDK

The AppZone SDK is an integrated development tool based on Eclipse IDE[1] and running on Windows. This environment provides a set of template files that form the skeleton of a future user M2M application. The user writes the code within the callback functions contained in the .c files of the skeleton.

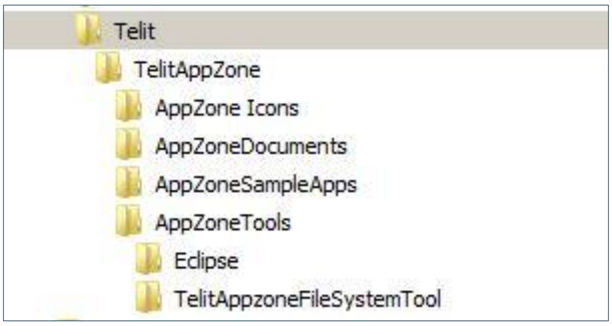
AppZone SDK during installation announces itself as Telit Application Development Environment. This term will be used throughout the document.

## 3.1 How to install the AppZone SDK

To install AppZone package, administrator permissions are needed. The installation requires:

- 128 MB RAM
- 250 MB free disk space
- Windows XP, Windows Vista, Windows 7 (32bit or 64bit), Windows 8

Run TelitAppZone Installer. The figure below shows the folders that appear on the computer after the installation. The upper folders of Telit folder and the Telit folder itself depend on your configuration during the installation. Now, on your desktop is available the shortcut to start the AppZone SDK.



```
Telit
  TelitAppZone
    AppZone Icons
    AppZoneDocuments
    AppZoneSampleApps
    AppZoneTools
      Eclipse
      TelitAppzoneFileSystemTool
```

Fig. 4: Main Folder of AppZone SDK

- Install the C language compiler, selecting between:
  - ARM Compiler: see Appendix 8.3 to install the RVCT 3.0 SP1 ARM compiler and license. It is customer responsibility to order the ARM compiler and related license.
  - GCC Compiler: see Appendix 8.3 to install the GCC compiler, distributed as Free Software. See Terms and Conditions in Appendix 8.4.
- Start the AppZone SDK.
  - Enter the name of the workspace folder, and follow the steps described in the next chapters.

[1] Eclipse is a multi-language development environment (IDE). Developer(s): Eclipse Foundation.

### 3.1.1 Create a M2M Project

Refer to Fig. 5 to create a new M2M project: Select File→New→C Project, and click on C Project

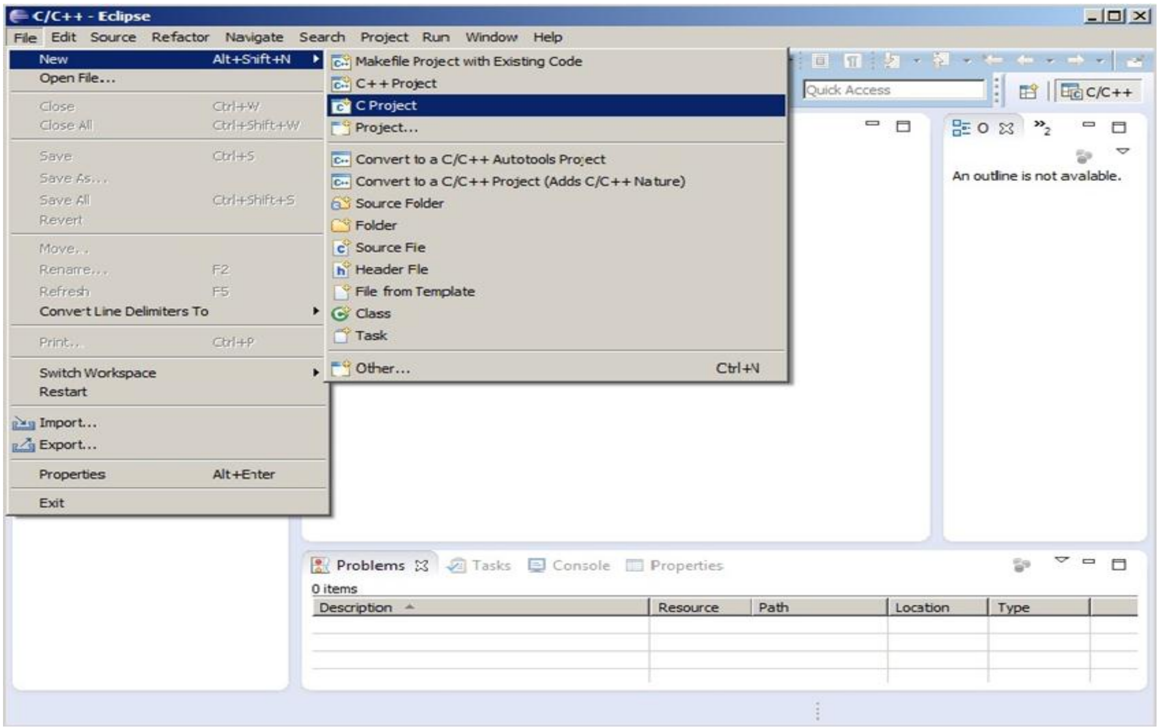
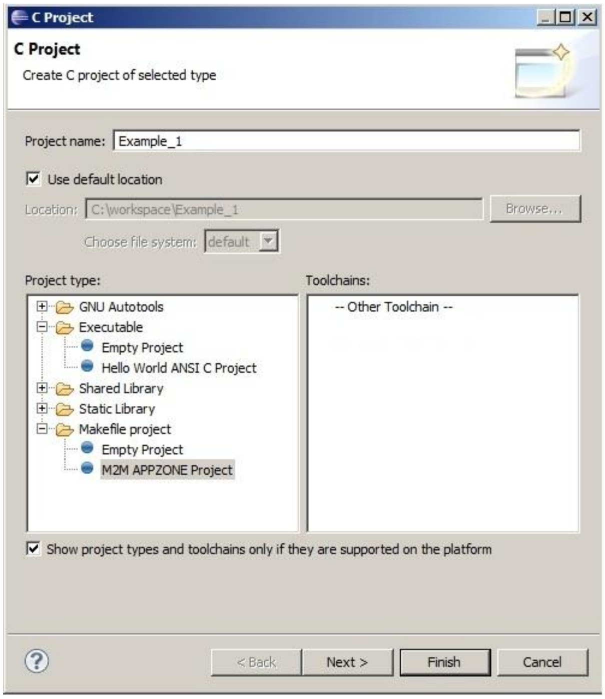


Fig. 5: Eclipse: Create a new M2M Project



Project name: Example_1

☑ Use default location

Location: C:\workspace\Example_1

Choose file system: default

☑ Show project types and toolchains only if they are supported on the platform

Fig. 6: Eclipse: Create the Skeleton

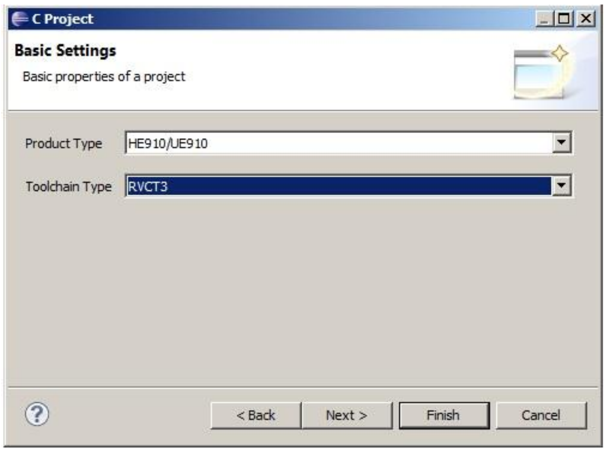
Use the dialog box shown on the right side to enter the name of the project, for example Example_1. Open the “Makefile project” folder and select M2M APPZONE Project. **NOTICE**:

- Telit header files are automatically included in the project.
- To include the RVCT 3.0 SP1 ARM compiler header files you must set the RVCT30INC variable in accordance with the compiler storage location. Telit_AppZone\bin\setenv.bat file contains the RVCT30INC variable.
- To use GCC compiler, refer to Appendix 8.3.

Hit the "Next" button.

Check or select the Product Type and Toolchain Type on the right dialog box. These variables are persistent variables. It means that if you create a new project their values do not change.

Hit the "Finish" button and the system shows the next window.



Basic Settings

Basic properties of a project

Product Type: HE910/UE910

Toolchain Type: RVCT3

Fig. 7: Eclipse: Basic Settings

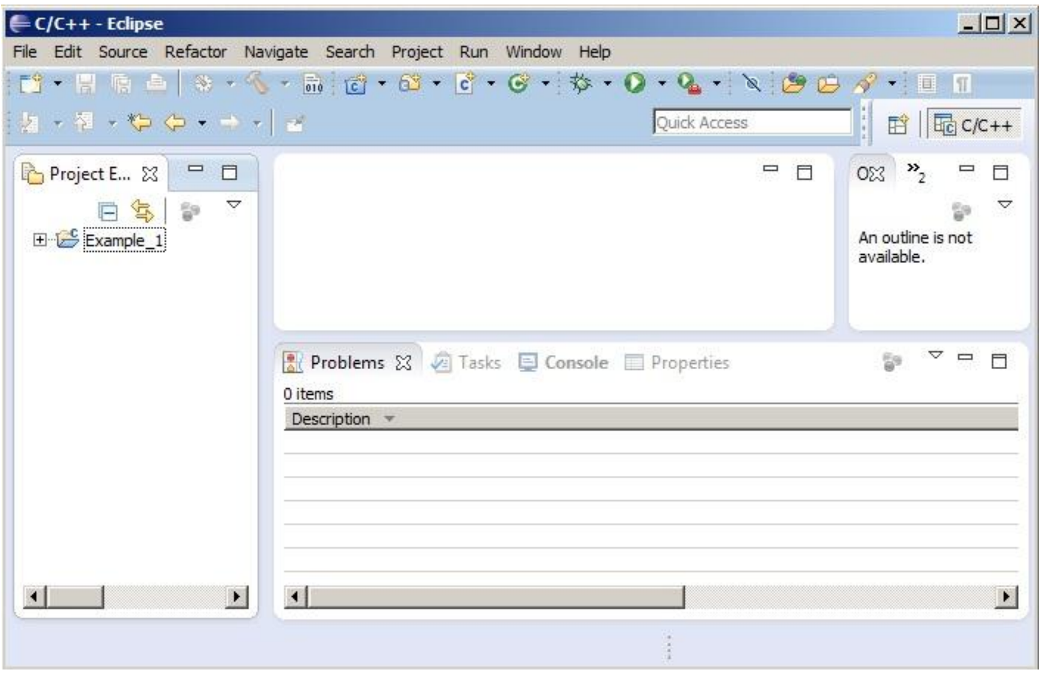
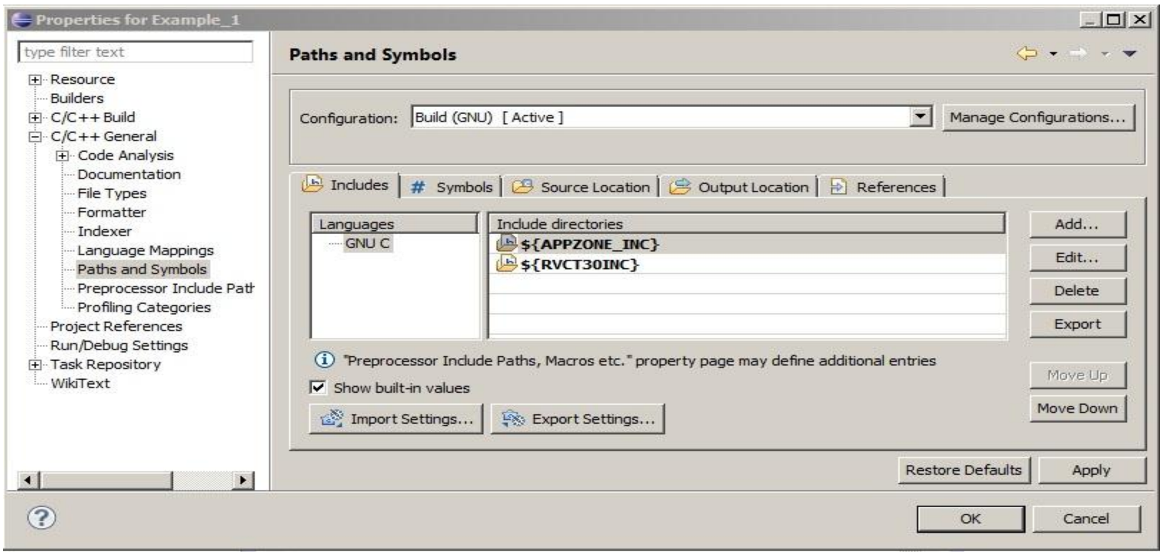


Fig. 8: Eclipse: Example_1 is the new M2M Project

Now, you can see the path variables of Telit's header files and compiler's header files:

- Right Click on "Example_1", the name of the project, refer to Fig. 8.
- Select Properties→C/C++ General→Paths and Symbols. You see the following variables: ${APPZONE_INC}, ${RVCT30INC}, refer to Fig. 9.



Configuration: Build (GNU) [ Active ]

Include directories:
${APPZONE_INC}
${RVCT30INC}

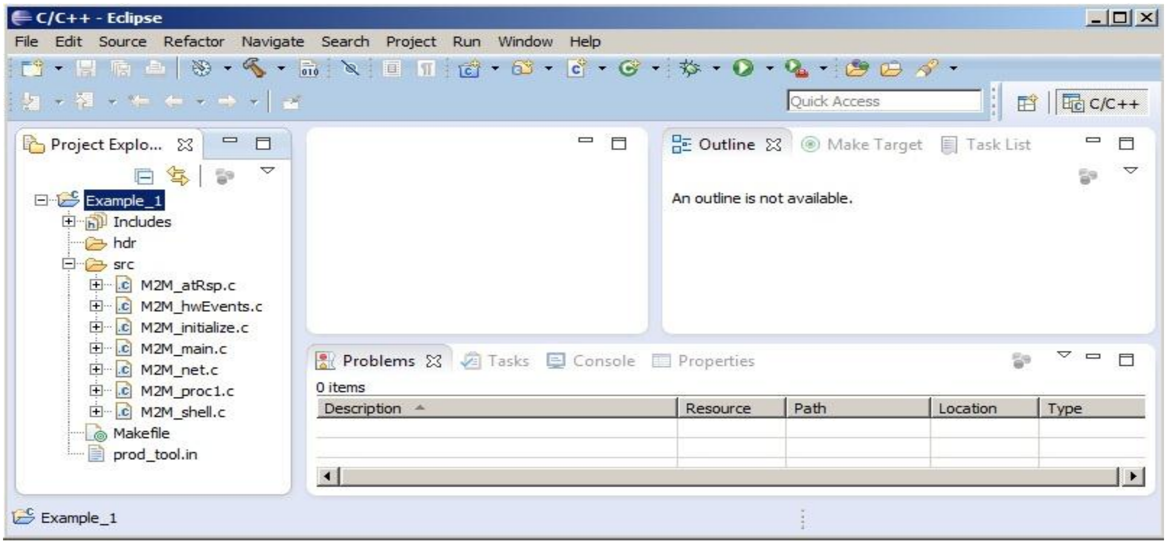
"Preprocessor Include Paths, Macros etc." property page may define additional entries

☑ Show built-in values

Fig. 9: Eclipse: Paths and Symbols

### 3.1.2 M2M Application Skeleton

The figure below shows the files of the skeleton and the "Includes" folder; the developer fills them with code, in accordance with his requirements, to develop the M2M application. The prod_tool.ini file contains information on products and toolchain.



An outline is not available.

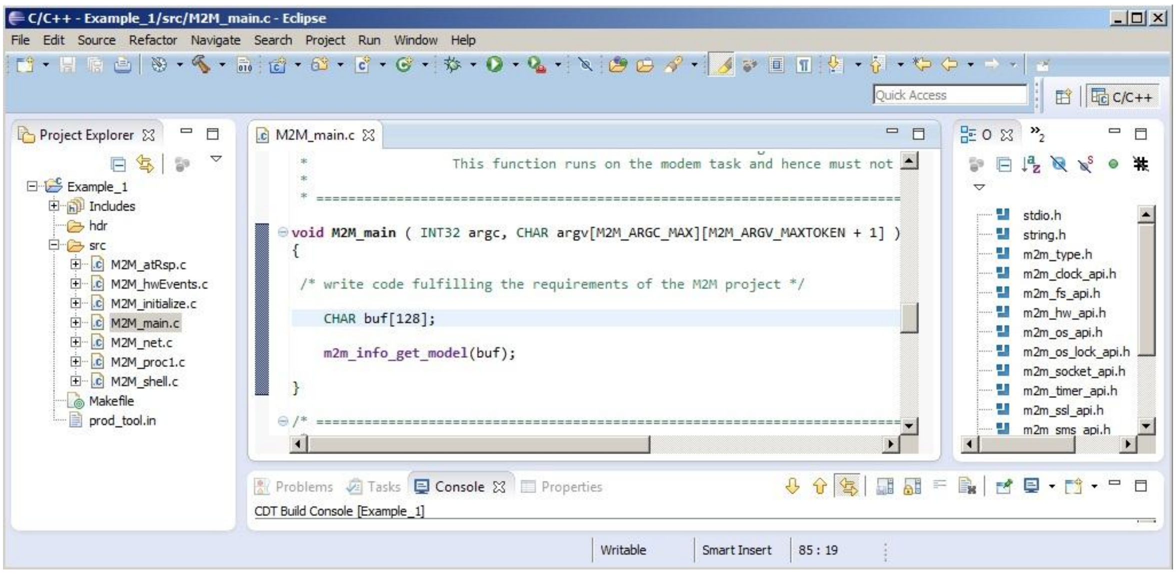
0 items

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|

Fig. 10: Eclipse: the Skeleton with the References to the header files (Includes folders)

### 3.1.3 Compile the M2M Project

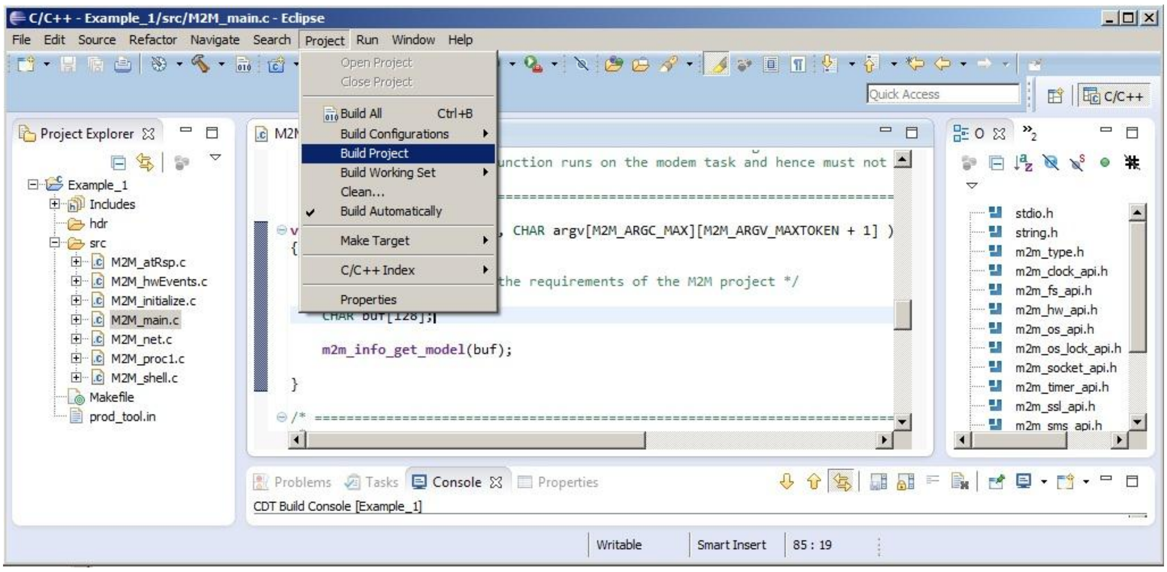
Before starting the compiling procedure, you must write in the M2M_main.c file some code to avoid warning messages. For example, write the function m2m_info_get _model(buf).



```
 *          This function runs on the modem task and hence must not
 *
 * ===========================================================================

void M2M_main ( INT32 argc, CHAR argv[M2M_ARGC_MAX][M2M_ARGV_MAXTOKEN + 1] )
{

  /* write code fulfilling the requirements of the M2M project */

    CHAR buf[128];

    m2m_info_get_model(buf);

}

/* ===========================================================================
```

Fig. 11: Write Code in M2M_main.c file

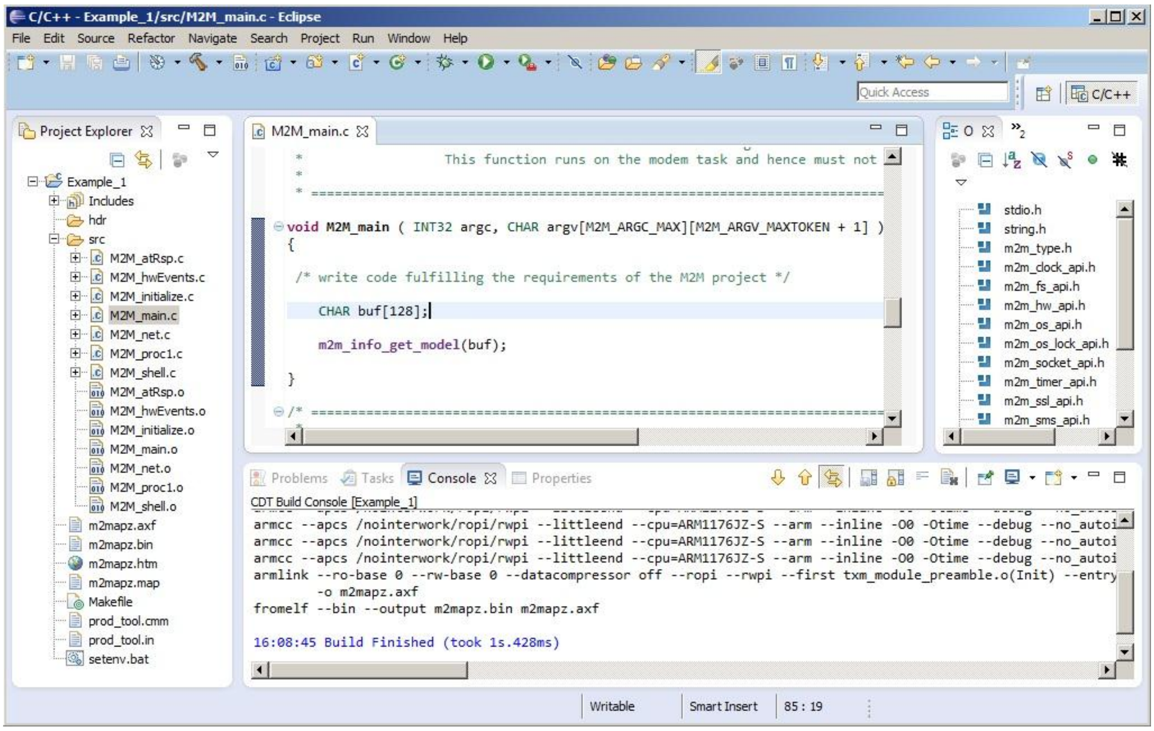
Now, compile the project. Select Project →Build Project and click on it



```
unction runs on the modem task and hence must not
, CHAR argv[M2M_ARGC_MAX][M2M_ARGV_MAXTOKEN + 1] )
{
the requirements of the M2M project */
    CHAR buf[128];
    m2m_info_get_model(buf);
}
/* ===========================================================================
```

Fig. 12: Eclipse: Compile the Example_1 Project

The figure below shows the compilation results. The file named m2mapz.bin contains the user M2M application that will be uploaded into the module as described in the next chapters.



```
 *                  This function runs on the modem task and hence must not
 *
 * ===========================================================================

void M2M_main ( INT32 argc, CHAR argv[M2M_ARGC_MAX][M2M_ARGV_MAXTOKEN + 1] )
{

 /* write code fulfilling the requirements of the M2M project */

    CHAR buf[128];

    m2m_info_get_model(buf);

}

/* ===========================================================================
```

```
armcc --apcs /nointerwork/ropi/rwpi --littleend --cpu=ARM1176JZ-S --arm --inline -O0 -Otime --debug --no_autoi
armcc --apcs /nointerwork/ropi/rwpi --littleend --cpu=ARM1176JZ-S --arm --inline -O0 -Otime --debug --no_autoi
armcc --apcs /nointerwork/ropi/rwpi --littleend --cpu=ARM1176JZ-S --arm --inline -O0 -Otime --debug --no_autoi
armlink --ro-base 0 --rw-base 0 --datacompressor off --ropi --rwpi --first txm_module_preamble.o(Init) --entry
        -o m2mapz.axf
fromelf --bin --output m2mapz.bin m2mapz.axf

16:08:45 Build Finished (took 1s.428ms)
```

Fig. 13: Eclipse: The m2mapz.bin Application is Ready

The m2mapz.bin application to run successfully on the module must be developed with a version of AppZone SDK tool aligned with the software version installed on the used module.

## 3.2 How to Use the Skeleton Files

The following sub-chapters describe how the developer should use the functions included in the .c files of the skeleton. Refer to document [2] to have more information on the number of AppZone tasks and M2M_msgProcX callback functions.

### 3.2.1 M2M_initialize.c File

The M2M_initialize.c file contains the following callback functions:

| Callback | Event starting the callback: |
|---|---|
| InitUserInterface(…) | is executed on the startup of the user M2M application. The user may configure the application modifying this function. |
| M2M_onAppUpgradeAvailable(…) | for future use. |
| M2M_initGPIO(…) | for future use. |

### 3.2.2 M2M_proc1.c (default) M2M_procX.c Files

The M2M_proc1.c file (default) contains the following callback functions:

| Callback | Event starting the callback: |
|---|---|
| M2M_msgProc1(...) | is the default callback. It is called by Task_1, see chapter 2 |
| M2M_msgProcCompl(...) | is called by the control when the generic (default or user) M2M_msgProcX(...) callback has finished its execution. |

As stated in chapter 2, the user can write custom M2M_procX.c files that may contain one or more M2M_msgProX(...) callback functions that are called by the connected AppZone Tasks.

The AppZone tasks, calling the M2M_msgProcX(...) callback, execute the code written by the user. The user may dedicate each task to a well-defined group of activities in accordance with the M2M application requirements and the needed execution priority.

Task_1 has the higher priority and Task_32 has the lower priority of the 32 tasks that can be provided by the AppZone Software Layer. The AppZone tasks have lower priorities than the modem tasks.

Example:

- Use the m2m_timer_create(…) API and its callback function to manage the timer expiration. Start the timer with the m2m_timer_start(…) API;
- When the timer is expired, the control executes the callback function to manage the timer expiration. Its code, written by the developer, in accordance with the timer identifier sends a message, for example, to Task_1 using the m2m_os_send_message_to_task(…) API;
- Task_1 detects the received message in its queue and calls the M2M_msgProc1(…) callback function that executes the code written by the developer;
- M2M_msgProc1(…) callback function, for example, sends a message to Task_5. In turn, it detects the message and calls the M2M_msgProc5(…) callback function.

### 3.2.3 M2M_main.c File

The M2M_main.c file contains the following callback functions:

| Callback | Event starting the callback: |
|---|---|
| M2M_main (…) | is the first function that the control executes when the M2M application is started. Refer to document [2] to have information on “argc” and “argv” parameters. |
| M2M_suspend (void) | will be used for future internal debugging activity. |
| M2M_resume (void) | will be used for future internal debugging activity. |

### 3.2.4 M2M_arRsp.c File

The M2M_arRsp.c file contains the following callback function.

| Callback | Event starting the callback: |
|---|---|
| M2M_onReceiveResultCmd(…) | When an AT command is entered into the module, it is executed by the AT parser. The callback function captures the AT command response generated by the AT parser. The code, written in the callback by the user, in accordance with the AT response performs the programmed action. The callback works either in Command and Data mode. |

M2M_onReceiveResultCmd(…) example:

- Send an AT command to AT parser by means of the m2m_os_iat_at_command(…) API ;
- AT parser executes the received AT command and the M2M_on_ReceiveResultCmd(…) callback is called;
- M2M_onReceiveResultCmd(…) callback executes the code written by the developer in accordance with the AT command result.

### 3.2.5 M2M_hwEvents.c File

The M2M_hwEvents.c file contains the following callback functions:

| Callback | Event starting the callback: |
|---|---|
| M2M_onWakeup(…) | expiration of the alarm time |
| M2M_onInterrupt(…) | interrupt created by a GPIO |
| M2M_onHWTimer(…) | expiration of the hardware timer |
| M2M_onUSbCableEvent(...) | plugging in/unplugging USB cable |
| M2M_onI2CEvent(…) | for future use |
| M2M_onKeyEvent(…) | actions on the “ON” key |

M2M_onWakeup(…) example:

- Use the m2m_rtc_set_date(…) and m2m_rtc_set_time(…) APIs to set the date and time of the module. Use the m2m_rtc_set_alarm(…) API to set the date and time of the alarm;
- When the date/time alarm is reached the M2M_onWakeup(…) callback is executed.

M2M_onInterrupt(…) example:

- Connect GPIO X with GPIO Y using a wire;
- Use the m2m_hw_gpio_write(…) API to force to 0 the GPIO X. Use the m2m_hw_gpio _int_enable(…) to configure the GPIO Y as a source of interrupt. Force to 1 the GPIO X via the m2m_hw_gpio_write(…) API;
- When the transition on the GPIO Y is detected, the M2M_onInterrupt(…) callback is executed.

M2M_onHWTimer(…) example:
When one or more hardware timers are expired, the control calls the M2M_onHWTimer(…) callback. The developer writes in the callback the code that will be executed in accordance with the identifier of the expired timer.

- Use the m2m_hw_timer_start(…) API to start one or more hardware timers;
- When an hardware timer expires the M2M_onHWTimer(…) callback is activated;
- M2M_onHWTimer(…) callback executes the code written by the developer in accordance with the identifier of the expired timer.

M2M_onUSbCableEvent(UINT32 usb_cable_event) example:
The function checks the plugging in/unplugging of the USB cable.

- When the USB cable is plugged in, the control calls the callback with the "usb_cable_event" parameter equal to 1.
- When the USB cable is unplugged, the control calls the callback with the "usb_cable_event" parameter equal to 0.

M2M_onKeyEvent(INT32 val1, INT32 val2) example:

The function checks the “ON” key that turns on/off the module.

- When the “ON” key is pushed, the control calls the callback with the "val1" parameter equal to 1 (key is pressed) and the "val2" parameter equal to 12 (key code of the "ON" key).
- When the “ON” key is released, the control calls the callback with the "val1" parameter equal to 0 (key is released) and the "val2" parameter equal to 12 (key code of the "ON" key).
- If the “ON” key is pushed down for a long time the control calls the callback with the "val1" parameter equal to 2 (key is held down) and the "val2" parameter equal to 12 (key code of the "ON" key). The callback is continuously called. Use the UART API to print out key status and code.

### 3.2.6 M2M_net.c File

The M2M_net.c file contains the following callback functions:

| Callback | Event starting the callback: |
|---|---|
| M2M_onNetEvent(…) | PDP context activation/deactivation, SOCKET closed /error, etc. |
| M2M_onRegStatusEvent(…) | Cell change |
| M2M_onGprsRegStatusEvent(...) | GPRS registration update |
| M2M_onMsgIndEvent(…) | reception of a SMS |
| M2M_onIP6RawEvent(…) | reception of an ip6 raw packet |

M2M_onNetEvent(...) example:

- Use the m2m_timer_create(…) API and write its callback function to manage the timer expiration. Start the timer through  the m2m_timer_start(…) API;
- When the timer is expired, the control calls the function to manage the timer expiration. Its code, written by the developer, in accordance with the timer identifier sends a message to the Task_1 using the m2m_os_send_message_to_task(…) API;
- Task_1 detects the received message in its queue and calls the M2M_msgProc1(…) callback function that executes the code written by the developer;
- M2M_msgProc1(…) callback, using the m2m_pdp_activate(…) API, create a PDP context. To wait for the module registration and the PDP context activation the M2M_msgProc1(…) starts a timer. When the timer expires, the control sends again a message to Task_1 to check the PDP status.
- When the PDP is active, the M2M_onNetEvent callback is executed. In general, this callback is activated by several network events. It is responsibility of the developer to wait for the desired event.

M2M_onRegStatusEvent(…) example:

- Enable the notification of the location registration by means of the m2m_network_enable_ registration _location_unsolicited() API;
- When cell_id or LAC changes, the M2M_onRegStatusEvent(…) callback is executed.

M2M_onGprsRegStatusEvent(...) example:

- Enable the notification of the GPRS registration by means of the m2m_network_enable_gprs _registration _location_unsolicited() API;
- When GPRS registration change, the M2M_onGprsRegStatusEvent(...) callback is executed.

M2M_onMsgIndEvent(…) example:

- Enable the message indication by means of the m2m_sms_enable_new_message_ indication(…) API;
- Send a SMS to the module;
- When the module receives the SMS message, the M2M_onMsgIndEvent (…) callback is executed.

M2M_onIP6RawEvent(…) example:

- Enable the ip6 raw mode by means of the m2m_ip6_raw(…) API;
- When the module receives an ip6 packet in raw mode, the M2M_onIP6RawEvent(…) callback is executed.

### 3.2.7 M2M_shell.c File

The M2M_shell.c file contains the following callback function:

| Callback | Event starting the callback: |
|---|---|
| M2M_cmdShell(…) | for future use |

# 4 +M2M AT COMMANDS

Assume that the developer has created a user M2M application contained in the m2mapz.bin file (default name). In addition, suppose that the developer has renamed the m2mapz.bin file into user_m2mapz.bin and uploaded it into the file system of the module.

The modules with AppZone layer provide several M2M AT commands to manage the user M2M applications and to configure the start mode of the selected application on next module reboot.

The AT commands examples described in the next chapters refer to Fig. 14. The DTE is based on a Windows-PC running a Terminal tool that sends AT commands and files to the module, and receives AT commands results from it. To have more information on AT commands syntax refer to Appendix 8.1.

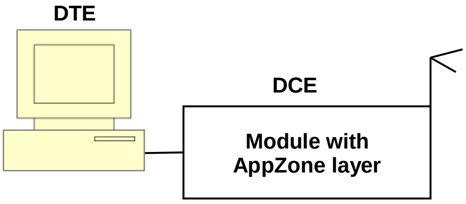


Fig. 14: DTE Connected to the Module

The chapter 5 ADE Console describes the console integrated into the Telit Application Development Environment (ADE). The console enables the user to enter AT Commands and upload/download files into/from module. It simplifies and speeds up the activities performed with the DTE. In addition, the console substitutes also the tool described in chapter 8.2 File System Tool.

# 4.1 AT+M2M=0

AT+M2M=0 disables the execution of both user and default M2M applications.

## 4.1.1 Example

This example introduces the default M2M application provided by the AppZone layer.

Assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and enter the following command to check the current +M2M value defining the application start mode on next reboot:

```
AT+M2M?
+M2M: 0,10,0   → the execution of any application is disabled on next reboot (default)
OK
```

Enter the command below to reboot the module.

```
AT+M2M=1
OK
```

After rebooting, the module starts its default M2M application because no user M2M application is available. The default application displays on DTE the following message:

```
Telit Communications S.p.A - AppZone M2M Default Application...
OK
```

The AT parser can still receive AT commands from the operator on the serial line. Enter the following command to reboot again the module.

```
AT+M2M=0
OK
```

After rebooting, the module does not execute the default M2M application; the operator may continue to enter AT commands into the module:

```
AT+M2M?
+M2M: 0,10,0   → the execution of any application is disabled on next reboot (default)
OK
```

## 4.2 AT+M2M=1

AT+M2M=1 starts the user M2M application having the RUN permission set through AT#M2MRUN command. If no user M2M application has set the RUN permission, AT+M2M=1 command starts the default M2M application.

### 4.2.1 Example

Assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and check the current drive:

```
AT#M2MCHDRIVE?
#M2MCHDRIVE: 0                → the module provides only drive 0
OK
```

Check the current directory:

```
AT#M2MCHDIR?
#M2MCHDIR: “/”                → factory setting
OK
```

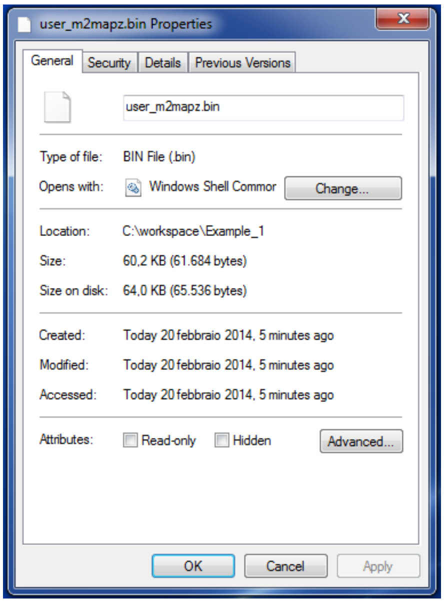
Enter the following command to verify the free space of file systems:

```
AT#M2MLIST
#M2MLIST: <MOD>
#M2MLIST: “m2m_config”,134
#M2MLIST: free bytes: 6160384 → free bytes
OK
```

The size of the file system depends on the module type under test.

Now, assume that the user has created a M2M application as show in the previous chapters, and renamed the m2mapz.bin file in user_m2mapz.bin.

Here are the steps to upload the user M2M application into the module.



user_m2mapz.bin

Type of file: BIN File (.bin)
Opens with: Windows Shell Commor
Location: C:\workspace\Example_1
Size: 60,2 KB (61.684 bytes)
Size on disk: 64,0 KB (65.536 bytes)
Created: Today 20 febbraio 2014, 5 minutes ago
Modified: Today 20 febbraio 2014, 5 minutes ago
Accessed: Today 20 febbraio 2014, 5 minutes ago
Attributes: □ Read-only □ Hidden

Before uploading the file into the file system of the module, you need to know the size of the file expressed in bytes. Use the Properties dialog box to get this information. The figure on the left side shows 61684 bytes, use this value with the AT#M2MWRITE command. Telit provides a tool to upload the user M2M application, refer to Appendix 8.2

Enter the command below. After the command has responded with the “>>>”prompt, you are allowed to send the M2M application using the Send File option provided by the Terminal tool (use RAW ASCII protocol). On success, the module returns the OK message.

```
AT#M2MWRITE=”user_m2mapz.bin”,61684,1
>>>
OK
```

If a file is already present in the module with the same name of the file just uploaded, the new one without notification overwrites the old one. The new uploaded file assumes the run permission of the old one.

If the third parameter is equal to 1, the AT#M2MWRITE command uploads automatically the .bin file into drive 0 and directory /MOD. To verify it enter the following commands.

Change the directory:

```
AT#M2MCHDIR=”MOD”
OK
```

Check the current directory:

```
AT#M2MCHDIR?
#M2MCHDIR: “/MOD”
OK
```

Check if the uploaded .bin file has been stored in the /MOD directory.

```
AT#M2MLIST
#M2MLIST: <.>
#M2MLIST: <..>
#M2MLIST: "user_m2mapz.bin",61684
#M2MLIST: free bytes: 6097920
OK
```

Enter the next command to check which M2M application has RUN permission. The command works on executable binary files and compressed files stored in the /MOD directory (drive 0).

```
AT#M2MRUN?
OK                 → no user applications have RUN permission
```

Check the current +M2M value defining the application start mode on next reboot.

```
AT+M2M?
+M2M: 0,10,0   → M2M application execution on next reboot is disabled (default)
OK
```

Enter the next command to reboot the module.

```
AT+M2M=1
OK
```

After rebooting, the module starts the default M2M application because the just uploaded user application has not RUN permission. The default M2M application displays on DTE the following message:

```
Telit Communications S.p.A - AppZone M2M Default Application...
OK
```

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

```
AT+M2M?
+M2M: 1,10,0   → M2M application execution on next reboot is disabled (default)
OK
```

Use the following command to set the RUN permission for the user_m2mapz.bin application, and reset RUN permission for all other applications included the default M2M application. The command below works on the executable binary files and compressed files stored in the /MOD directory of drive 0.

```
AT#M2MRUN=2,"user_m2mapz.bin"
OK
```

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

```
AT#M2MRUN?
#M2MRUN: “user_m2mapz.bin"            →user application has RUN permission
OK
```

Enter the next command to reboot the module.

```
AT+M2M=1
OK
```

After rebooting, the module starts the user M2M application enabled via the previous AT#M2MRUN command. For example, the user M2M application displays on DTE the following message:

```
Hello World
```

The AppZone layer has executed the user application. The AT parser cannot receive AT commands from the operator because the user application has permanently captured the serial line. If the module is turned off/on, the user application runs again; to exit from this scenario the module must be re-flashed.

## 4.3 AT+M2M=2

AT+M2M=2 starts only the default M2M application, regardless if a user M2M application has the RUN permission set through AT#M2MRUN command.

### 4.3.1 Example

Assume that the module has factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Enter the following command to upload the user_m2mapz.bin application. Use the procedure shown in previous examples.

```
AT#M2MWRITE=”user_m2mapz.bin”,61684,1
>>>
OK
```

Enter /MOD directory, and verify if the just uploaded .bin file has been stored in it.

```
AT#M2MLIST
#M2MLIST: <.>
#M2MLIST: <..>
#M2MLIST: "user_m2mapz.bin",61684
#M2MLIST: free bytes: 6097920
OK
```

Enter the next command to check which M2M application has RUN permission. The command works on executable binary files and compressed files stored in the /MOD directory (drive 0).

```
AT#M2MRUN?
OK                                  → no user applications have RUN permission
```

Use the following command to set the RUN permission for user_m2mapz.bin application and reset RUN permission for all other applications included the default M2M application.

```
AT#M2MRUN=2,"user_m2mapz.bin"
OK
```

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

```
AT#M2MRUN?
#M2MRUN: “user_m2mapz.bin"          → user application has RUN permission
OK
```

Check the current +M2M value defining the application start mode on next reboot.

```
AT+M2M?
+M2M: 0,10,0                        →the execution of any application is disabled on next reboot
(default)
OK
```

Enter the next command to reboot the module.

```
AT+M2M=2
OK
```

After rebooting, the default M2M application is executed even if the user M2M application has the RUN permission. The default M2M application displays on DTE the following message:

```
Telit Communications S.p.A - AppZone M2M Default Application...
OK
```

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

```
AT+M2M?
+M2M: 2,10,0
OK
```

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

```
AT#M2MRUN?
#M2MRUN: “user_m2mapz.bin"          → user application has RUN permission
OK
```

Now, turn off/on the module. The module reboots and starts the default application even if the user M2M application has the RUN permission. The default M2M application displays on DTE the following message:

```
Telit Communications S.p.A - AppZone M2M Default Application...
OK
```

The AT parser can still receive AT commands from the operator on the serial line.

## 4.4 AT+M2M=3

AT+M2M=3 starts the user M2M application execution in accordance with the level of DTR control line. To perform the example described in the next chapter on DTE - refer to Fig. 14 - is installed the Telit AT Controller application because it provides the feature to change manually the DTR line. DTR is an output control line of the DTE, when it is low (active) informs the module that the DTE is ready to establish a communication.

### 4.4.1 Example

Assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and upload the user M2M application into /MOD directory (driver 0).

```
AT#M2MWRITE=”user_m2mapz.bin”,61684,1
>>>
OK
```

Check the current +M2M value defining the application start mode on next reboot. AT+M2M?

```
+M2M: 0,10,0    → the execution of any application is disabled on next reboot (default)
OK
```

Force the DTR control line to low using the proper button provided by the Telit AT Controller application (click twice on the green DTR check marker, it becomes red). Enter the command below to reboot the module. DTR is low, the module starts the default M2M application because the just uploaded user application does not have the RUN permission, refer to Tab. 1.

```
AT+M2M=3
OK
```

After rebooting, the default M2M application is executed and displays on DTE the following message:

```
Telit Communications S.p.A - AppZone M2M Default Application...
OK
```

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

```
AT+M2M?
+M2M: 3,10,0
OK
```

Force the DTR control line to high (click twice on the read DTR check marker, it becomes green), and power off/on the module. After rebooting, no M2M application is started (DTR is high), the operator may continue to enter AT commands into the module, refer to Tab. 1.

Check the current +M2M value stored in NVM after the previous rebooting.

```
AT+M2M?
+M2M: 3,10,0
OK
```

Enter the next command to check which user M2M application has RUN permission.

```
AT#M2MRUN?
OK                                  → no user M2M application has RUN permission
```

Use the following command to set the RUN permission for user_m2mapz.bin application and reset RUN permission for all other applications included the default M2M application.

```
AT#M2MRUN=2,"user_m2mapz.bin"
OK
```

Check which M2M application has RUN permission.

```
AT#M2MRUN?
#M2MRUN: "user_m2mapz.bin"
OK
```

Force the DTR control line to low (click twice on the green DTR check marker, it becomes red), and power off/on the module, refer to Tab. 1. After rebooting, the user M2M application is started and displays on DTE, for example, the following message:

```
Hello World
```

The operator may not enter more AT commands into the module. The user M2M application has permanently captured the serial line. Follow the steps below to exit this scenario:

1. Force the DTR control line to high (click twice on the read DTR check marker, it becomes green);
2. Turn off/on the module.

After rebooting, no M2M applications have been started, the AT parser accepts again AT commands, refer to Tab. 1. Check the current +M2M value stored in NVM after the previous rebooting.

```
AT+M2M?
+M2M: 3,10,0
OK
```

The table below summarizes the four cases above described.

| DTR | Run Permission | Enter | Start |
|---|---|---|---|
| Low | No | AT+M2M=3 | Default M2M Appl. |
| High | No | OFF/ON | No M2M Appl. |
| Low | Yes | OFF/ON | User M2M Appl. |
| High | Yes | OFF/ON | No M2M Appl. |

Tab. 1: DTR & +M2M=3

## 4.5 AT+M2M=4,30

AT+M2M=4,xx starts the user M2M application having the RUN permission set, if xx seconds are expired. If the user types an AT command before the expiration of the xx seconds, the control does not executes the M2M application. The RUN permission is set by means of the AT#M2MRUN command.

### 4.5.1 Example

Assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and upload the user M2M application into /MOD directory (drive 0). For details, refer to chapter 4.2.1.

```
AT#M2MWRITE=”user_m2mapz.bin”,61684,1
>>>
OK
```

Change the directory:

```
AT#M2MCHDIR=”MOD”
OK
```

Check the current directory:

```
AT#M2MCHDIR?
#M2MCHDIR: “/MOD”
OK
```

Verify, after the uploading, the list of files stored in the current directory (/MOD).

```
AT#M2MLIST
#M2MLIST: <.>
#M2MLIST: <..>
#M2MLIST: "user_m2mapz.bin",61684              → the user M2M application
#M2MLIST: free bytes: 6097920
OK
```

Use the following command to set the RUN permission for user_m2mapz.bin application and reset RUN permission for all other application included the default M2M application. The next command works on the executable binary files and compressed files stored in the /MOD directory.

```
AT#M2MRUN=2,"user_m2mapz.bin"
OK
```

Enter the next command to check which M2M application has RUN permission.

```
AT#M2MRUN?
#M2MRUN: “user_m2mapz.bin"                     → user M2M application has RUN permission
OK
```

Check the current +M2M value defining the application start mode on next reboot.

```
AT+M2M?
+M2M: 0,10,0          → the execution of any application is disabled on next reboot (default)
OK
```

Use the AT+M2M command with two parameters: the first one (4) sets the start mode, the second one (30, as example) defines a time interval expressed in sec.

```
AT+M2M=4,30
OK
```

After entering this command, the module reboots, and the following two choices are available:

1. If the operator enters an AT command (different from AT<CR>), before the expiration of the time interval, the control does not start the user M2M application, the user may enter AT commands. Enter the next command to verify the +M2M values:

   ```
   AT+M2M?
   +M2M: 4,30          → the +M2M values are memorized in NVM
   OK
   ```

After entering again the AT+M2M=4,30 command the module reboots, and again the two choices are available.

2. If the time interval expires and the operator has not entered AT commands, the control starts the user M2M application. If no user application has RUN permission, the default application is executed.

If the module is powered OFF/ON, it uses the AT+M2M=4,30 configuration memorized in NVM and one of the scenarios above described may be used again.

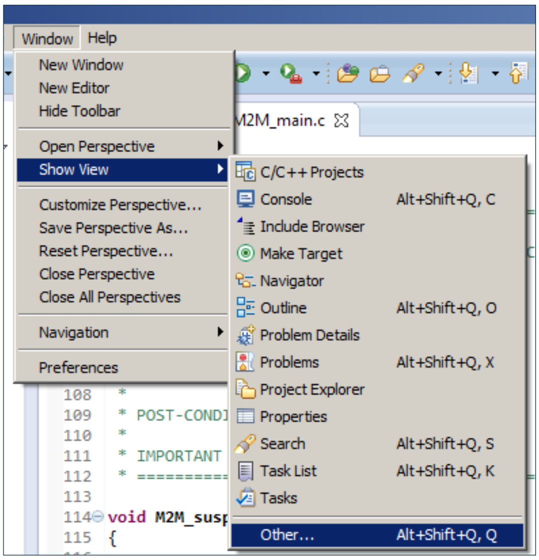
# 5 ADE CONSOLE

Telit Application Development Environment (ADE) provides an integrated console to simplify and speed up the exchange of AT Commands/Results and files with the module. The list below summarizes the features provided by this console called Telit AT Controller:

- transfer a .bin file into the module file system (uploading)
- read a file stored on the module file system and store it on PC (downloading)
- list files, check current directory, change directories on the module file system
- set COM ports, Baud Rate, etc.
- set the selected file as AppZone Application ( refer to AT+M2MRUN command)
- enter AT commands

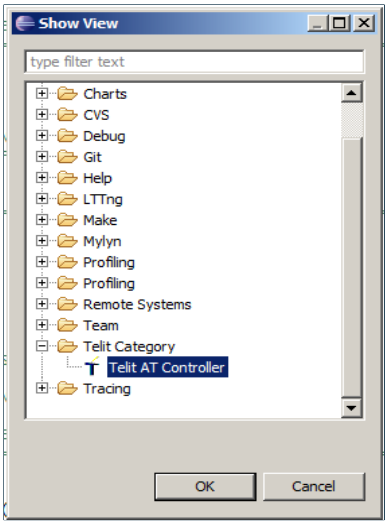
To start the Telit AT Controller see the following steps and the screenshots on the right side.

Window → Show View → other.



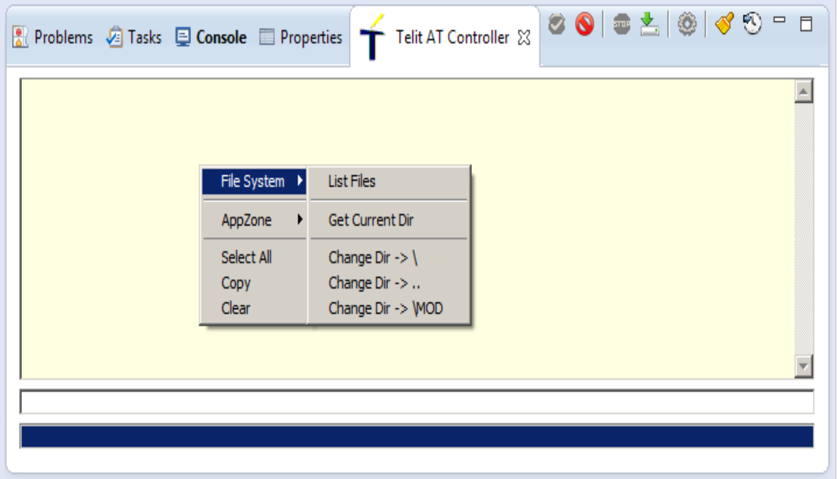
```
 *
 * POST-CONDI
 *
 * IMPORTANT
 * ==========

void M2M_susp
{
```

Select, on the Show View dialog box, Telit AT Controller and click on the OK button.



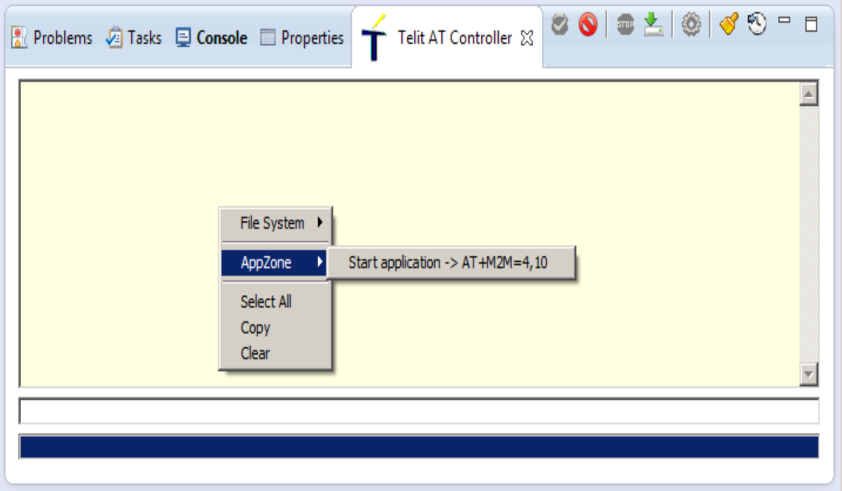
On the bottom of the ADE main window, the Telit AT Controller TAB (console) is displayed. See the screenshot below.

In the yellow area of the console are displayed the messages coming from the module, for example the AT Commands results, the open COM result, etc. The first dialog box under the yellow area is used to enter AT commands. The second one shows the progress of a file downloading/uploading that will be described in detail later.



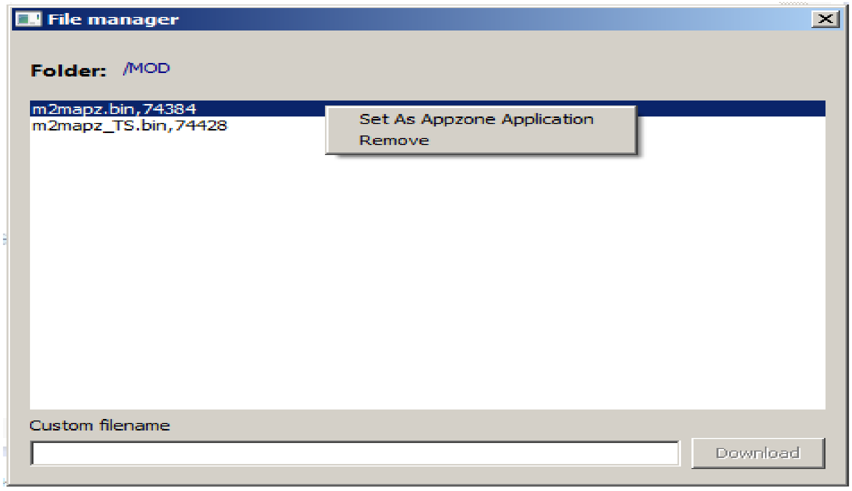
Right click on the yellow area of the console to display the gray box. The list items are self-explanatory.

Before clicking on one of the list items, click on ✓ icon to open the COM, otherwise a warning message will be displayed on the console.



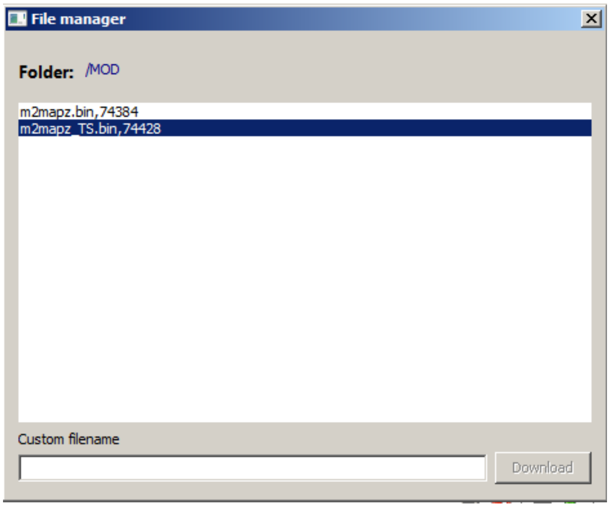
Click on CHANGE DIR → \MOD to enter into \MOD directory. Now, click on icon to open the File Manager dialog box, see the right screenshot.

Select the desired file and right click on it, the gray box is displayed. Now, you can set it as AppZone Application or remove it.



Use the same File Manager box to download (save on PC) a file stored on the module file system. Select the desired file and double click on it. The Save As dialog box appears to manage the storing of the file on a selected folder of the PC. When the downloading is finished, a message is displayed on the console.

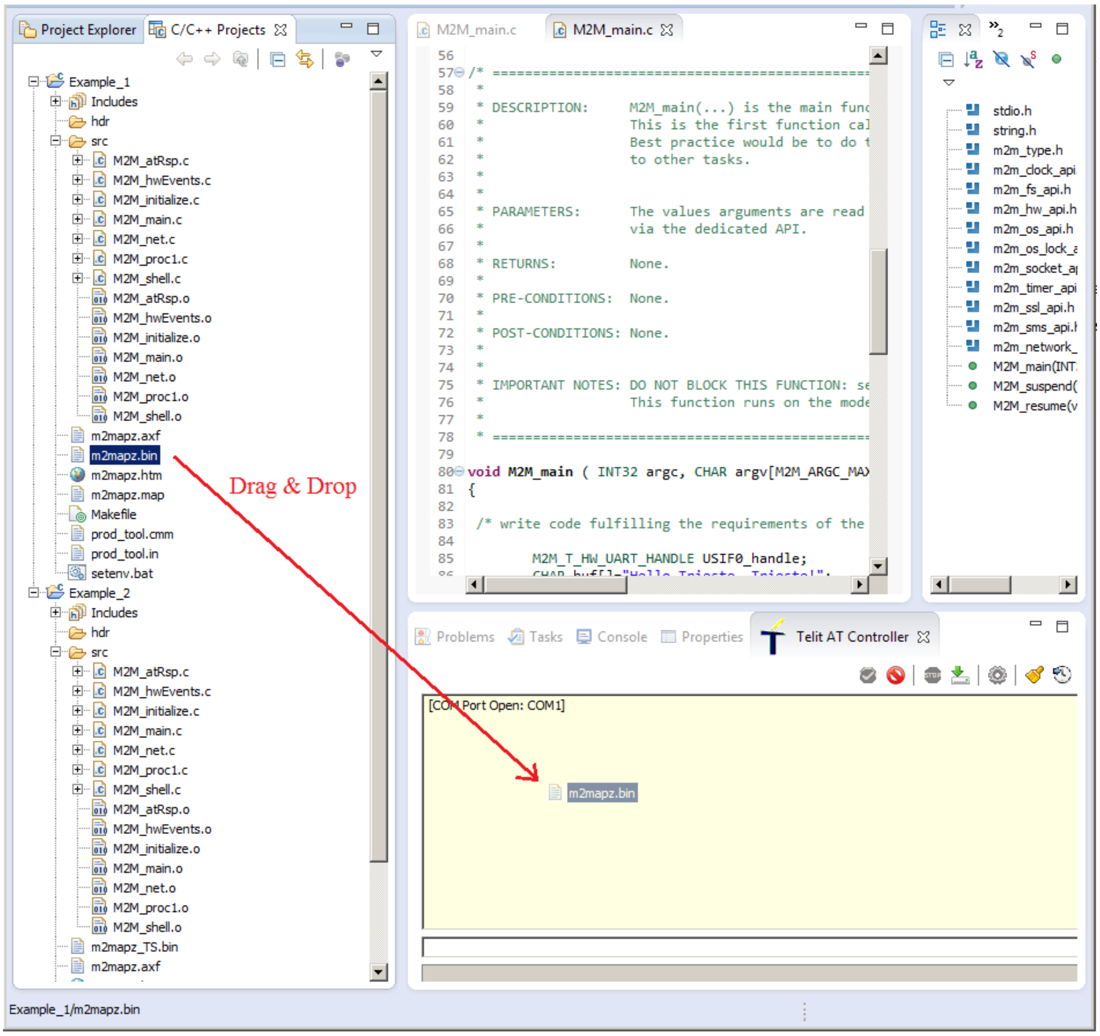
The Custom filename field and the related Download button are provided for future uses.



m2mapz.bin,74384
m2mapz_TS.bin,74428

The screenshots below pointing out the integration between the development environment and the Telit AT Controller (console). The user by mean of a unique tool can perform the following activities:

- creates the project
- compiles the project and obtains the m2maps.bin file
- clicks on ✓ icon to open the COM
- drags and drops the file to be uploaded from the C/C++ Projects TAB to the yellow area of the console, see the screenshot below.



```
/* ===============================================
 *
 * DESCRIPTION:     M2M_main(...) is the main func
 *                  This is the first function cal
 *                  Best practice would be to do t
 *                  to other tasks.
 *
 *
 * PARAMETERS:      The values arguments are read
 *                  via the dedicated API.
 *
 * RETURNS:         None.
 *
 * PRE-CONDITIONS:  None.
 *
 * POST-CONDITIONS: None.
 *
 *
 * IMPORTANT NOTES: DO NOT BLOCK THIS FUNCTION: se
 *                  This function runs on the mode
 *
 * ===============================================

void M2M_main ( INT32 argc, CHAR argv[M2M_ARGC_MA)
{

 /* write code fulfilling the requirements of the

        M2M_T_HW_UART_HANDLE USIF0_handle;
```

```
[COM Port Open: COM1]
```

Here are the icons meaning:

Open COM port

Close COM port

Stop transfer

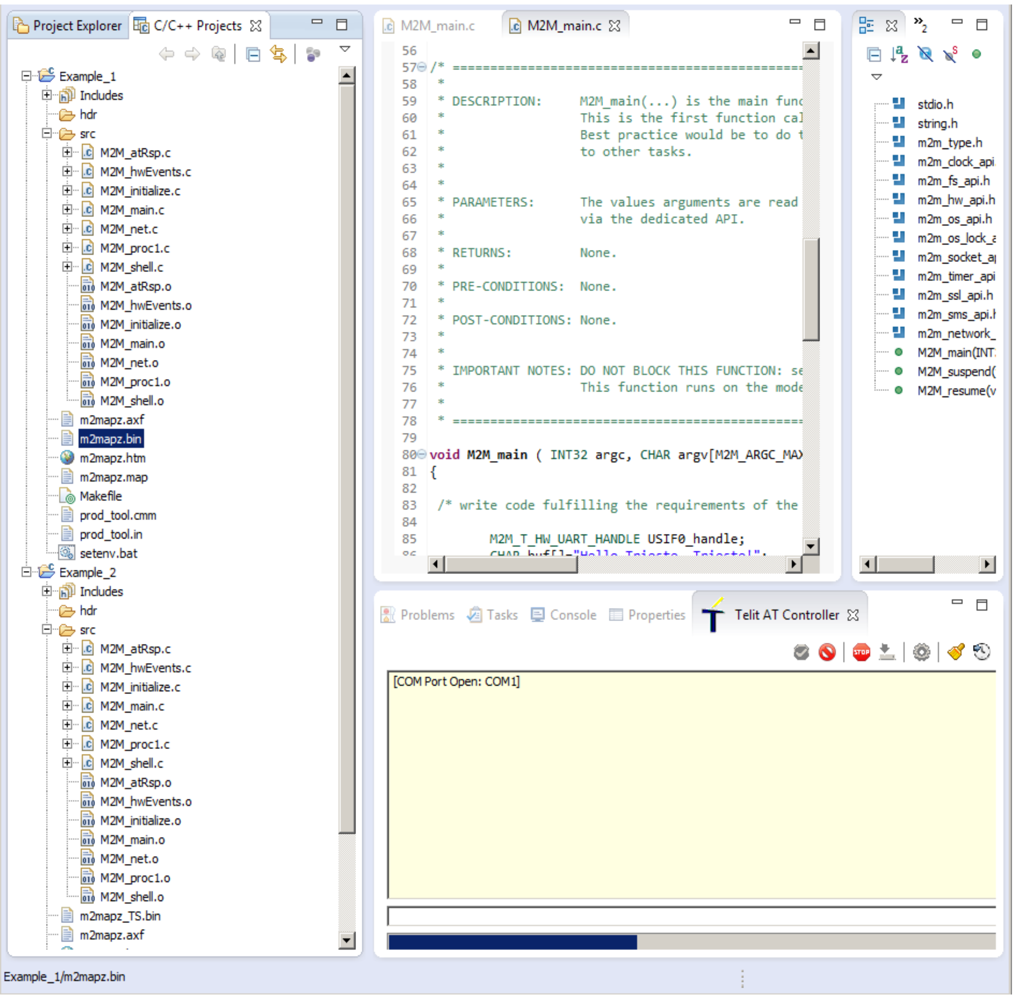
File Manager

Settings

Clear Log

Enable Timer Log

When the user releases the right button, the uploading of the file start as shown by the right screenshot.

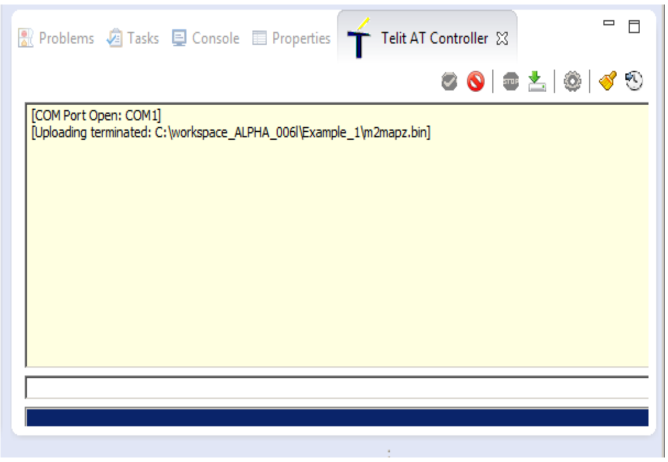


```
/* ================================================
 *
 * DESCRIPTION:     M2M_main(...) is the main func
 *                  This is the first function cal
 *                  Best practice would be to do t
 *                  to other tasks.
 *
 *
 * PARAMETERS:      The values arguments are read
 *                  via the dedicated API.
 *
 * RETURNS:         None.
 *
 * PRE-CONDITIONS:  None.
 *
 * POST-CONDITIONS: None.
 *
 *
 * IMPORTANT NOTES: DO NOT BLOCK THIS FUNCTION: se
 *                  This function runs on the mode
 *
 * ================================================

void M2M_main ( INT32 argc, CHAR argv[M2M_ARGC_MAX
{

 /* write code fulfilling the requirements of the

        M2M_T_HW_UART_HANDLE USIF0_handle;
```

```
[COM Port Open: COM1]
```

Example_1/m2mapz.bin

When the uploading is finished, the user is notified by the message shown on the console.



```
[COM Port Open: COM1]
[Uploading terminated: C:\workspace_ALPHA_006\Example_1\m2mapz.bin]
```

# 6 ACRONYMS & ABBREVIATIONS

| | |
|---|---|
| ADE | Application Development Environment |
| API | Application Programming Interface |
| DTE | Data Terminal Equipment |
| GCC | GNU Compiler Collection |
| GPIO | General Purpose Input/Output |
| I2C | Inter-Integrated Circuit |
| NVM | Non-Volatile Memory |
| PDP | Packet Data Protocol |
| SDK | Software Development Kit |
| SMS | Short Message Service |
| SPI | Serial Peripheral Interface |
| SSL | Secure Socket Layer |
| UART | Universal Asynchronous Receiver Transmitter |

# 7 MODULES & SW VER. TABLES

**SOFTWARE VER. TABLE**

| | SW Versions |
|---|---|
| **HE910 Family** | |
| HE910 [2] | **12.00.xx6-Bxxx** |
| **UE910 Family (Embedded)** | |
| UE910-EUR / UE910-EUD | **12.00.xx6-Bxxx** |
| UE910-NAR / UE910-NAD | **12.00.xx6-Bxxx** |
| **GE910 Family (Embedded)** | |
| GE910-QUAD | **13.00.xx7-Bxxx** |
| GE910-GNSS | **13.00.xx7-Bxxx** |

**SERVICES COEXISTENCE TABLE**

| | Services | | | |
|---|---|---|---|---|
| | Embedded GPS | External GPS | Python | AppZone |
| **HE910 Family** | | | **Python and AppZone are mutually exclusive** | |
| HE910 | ✓ | | ✓ | ✓* |
| **UE910 Family (Embedded)** | | | | |
| UE910-EUR / UE910-EUD | | ✓ | ✓ | ✓* |
| UE910-NAR / UE910-NAD | | ✓ | ✓ | ✓* |
| **GE910 Family (Embedded)** | | | | |
| GE910-QUAD | | ✓ | ✓ | ✓* |
| GE910-GNSS | ✓ | | ✓ | ✓* |

The table above summarizes the Services provided by the modules when they are equipped with the suitable software version, and shows the Services coexistence. Embedded/External GPS and Python Services are beyond the scope of this guide.

(*): AppZone available on demand on specific part numbers.

[2] HE910 is the “type name” of the products marketed as HE910-G & HE910-DG

# 8 APPENDIXES

## 8.1 AT Syntax

The previous chapters describe the use of some M2M AT Commands. This Appendix describes the syntax of all AT Commands dedicated to manage the M2M applications.

### 8.1.1 AT+M2M

| **+M2M – Enable/disable M2M Application execution** | **SELINT 2** |
|---|---|
| **AT+M2M=<start_mode>[,<br><start_to>,<start_shell>]** | Set command sets the M2M Application execution start mode<br><br>Parameters:<br>**<start_mode>**<br>0 – disable the M2M Application execution at the next startup (default).<br>1 – enable the M2M Application execution at the next startup.<br>2 – enable the default M2M Application execution at the next startup.<br>3 – enable the M2M Application execution only if, at the next startup, the DTR line is found Low (that is: COM is not open on a PC).<br>4 – enable the M2M Application execution only if, at the next startup, the user does not send any AT command on the serial port for time interval specified in. **<start_to>** parameter. The DTR line is not tested.<br><br>**<start_to>** – M2M Application execution start time-out;<br>10..60 - time interval in seconds; this parameter is used only if parameter **<start_mode>** is set to 4; it is the waiting time for an AT command on the serial port to disable active script execution start. If the user does not send any AT command on the serial port for the time specified in this parameter the M2M Application will not be executed (default is 10).<br><br>**<start_shell>** – Reserved for future use.<br><br>Note: after issuing the AT command, the module will automatically restart.<br>Note: an optional way to disable the M2M application execution (when **<start_mode>**=1 or 2) consists to flash an appropriate the stream. |
| **AT+M2M?** | Read command reports the M2M Application execution start mode, start time-out, and start shell in the format:<br>**+M2M: <start_mode>,< start_to>,<start_shell>** |
| **AT+M2M=?** | Test command returns the range of available values for parameters<br>**< start_mode>, < start_to>, and <start_shell>**. |

### 8.1.2 AT#M2MWRITE

| **#M2MWRITE – M2M File System File Write** | **SELINT 2** |
|---|---|
| **AT#M2MWRITE=<file_name>,<size>[,<permission>]** | Execution command causes the module to store a generic file, for example M2M Application binary file, in the M2M file system in the current working directory, naming it **<file_name>**<br><br>The file should be sent using RAW ASCII file transfer. It is important to set properly the port settings. In particular: Flow control: hardware.<br><br>Parameters:<br>**<file_name>** – file name in M2M file system, quoted string type (up to max 16 chars depending on current working directory, case sensitive).<br><br>**<size>** – file size in bytes<br><br>**<permission>** – file permission (optional); sum of integers each representing a file permission; default value if not present is 0;<br>0 - nothing<br>1 - executable binary (.bin)<br>16 - compressed (.gz)<br><br>If the parameter **<permission>** is present, it is not 0, its value matches file name extension (e.g.: 1 - .bin; 16 - .gz, 17 - .bin.gz) the file will be stored in the M2M file system in drive 0 in the directory \MOD setting the requested file permission. If the parameter **<permission>** is not present the file will be stored in the M2M file system in the current working directory without setting any file permission.<br><br>The device shall prompt a five character sequence<br>**<CR><LF><greater_than><greater_than><greater_than>**<br>**(IRA 13, 10, 62, 62, 62)**<br>after command line is terminated with **<CR>**; after that a file can be entered from TE, sized **<size>** bytes.<br><br>The operations complete when all bytes are received.<br><br>If writing ends successfully, the response is **OK**; otherwise an error code is reported.<br><br>Note: the file name should be passed between quotes; file names are case sensitive. |
| **AT#M2MWRITE=?** | Test command returns **OK** result code. |
| Example | AT#M2MWRITE="M2MAPZ.bin",58044<br>>>> *here receive the prompt; then type or send the file, sized 58044 bytes*<br>OK<br>File has been stored |

### 8.1.3 AT#M2MLIST

| **#M2MLIST – M2M File System List** | **SELINT 2** |
|---|---|
| **AT#M2MLIST** | Execution command reports the list of directories names and file names of directories and files currently stored in the M2M file system in the current working directory. In the end reports the available free memory in the current drive. The report is in the format:<br><br>**[<CR><LF>#M2MLIST: <.>**<br>**<CR><LF>#M2MLIST: <..>]**<br>**[<CR><LF>#M2MLIST: <dir_name1>…**<br>**[<CR><LF>#M2MLIST: <dir_namen>]]**<br>**[<CR><LF>#M2MLIST: <file_name1>,<size1>…**<br>**[<CR><LF>#M2MLIST: <file_namen>,<sizen>]]**<br>**<CR><LF>#M2MLIST: free bytes: <free_mem>**<br><br>where:<br>**<.>** current directory<br>**<..>** upper directory<br>**<dir_namen>** – directory name, string type delimited by < and > (max 16 chars, case sensitive)<br>**<file_namen>** – file name, quoted string type (max 16 chars, case sensitive)<br>**<sizen>** – size of file in bytes<br>**<free_mem>** – size of available free memory in the current drive in bytes |
| **AT#M2MLIST=?** | Test command returns **OK** result code. |
| Example | AT#M2MLIST<br>#M2MLIST: <.><br>#M2MLIST: <..><br>#M2MLIST: <dir1><br>#M2MLIST: "M2MAPZ.bin",58044<br>#M2MLIST: free bytes: 458752<br><br>OK |

### 8.1.4 AT#M2MDEL

| **#M2MDEL – M2M File System File Delete** | **SELINT 2** |
|---|---|
| **AT#M2MDEL=[<file_name>]** | Execution command deletes the file from the M2M file system in the current working directory.<br><br>Parameter:<br>**<file_name>** – file name to delete, quoted string type (max 16 chars, case sensitive)<br><br>Note: the file name should be passed between quotes; file names are case sensitive.<br>Note: If the file **<file_name>** is not present in the current working directory, an error code is reported. |
| **AT#M2MDEL=?** | Test command returns **OK** result code. |
| Example | AT#M2MDEL="M2MAPZ.bin"<br>OK |

## 8.1.5 AT#M2MREAD

| **#M2MREAD – M2M File System File Read** | **SELINT 2** |
|---|---|
| **AT#M2MREAD=[<file_name>]** | Execution command reports the content of the file <file_name> stored in the M2M file system in the current working directory.<br>Parameter:<br>**<file_name>** – file name, quoted string type (max 16 chars, case sensitive)<br>The device shall prompt a five character sequence<br>**<CR><LF><less_than><less_than><less_than>**<br>**(IRA 13, 10, 60, 60, 60)**<br>followed by the file content.<br>Note: the file name should be passed between quotes; file names are case sensitive.<br>Note: If the file **<file_name>** is not present in the current working directory, an error code is reported. |
| AT#M2MREAD=? | Test command returns **OK** result code. |
| Example | AT#M2MREAD="config.txt "<br><<< *here receive the prompt; then the file is displayed, immediately after the prompt*<br>OK |

### 8.1.6 AT#M2MRUN

| **#M2MRUN – M2M Set Run File Permission** | | **SELINT 2** |
|---|---|---|
| **AT#M2MRUN=<mode>[,<file_name>]** | Set command sets and resets the RUN file permission for the executable binary files and compressed files stored in the drive 0 in the directory \MOD in the M2M file system.<br>Parameters:<br>**<mode>** - integer type, set/reset mode value:<br>0 – resets RUN file permission for all the files stored in the drive 0, directory \MOD.<br>1 – sets RUN file permission for the file **<file_name>** stored in the drive 0, directory \MOD. Reserved for future use.<br>2 – sets RUN file permission for the file **<file_name>** stored in the drive 0, directory \MOD, and resets RUN file permission for all the other files stored in the drive 0, directory \MOD.<br>**<file_name>** - file name to set RUN file permission, quoted string type (max 16 chars, case sensitive). File name extension must be either .bin or .gz or .bin.gz. **<file_name>** parameter must not be present if **<mode>** is 0.<br>Note: the file name should be passed between quotes; file names are case sensitive. | |
| **AT#M2MRUN?** | Read command reports the files with the RUN file permission between those in the drive 1 in the directory \MOD in the M2M file system in the format:<br>**[<CR><LF>#M2MRUN: <file_name1>…**<br>**[<CR><LF>#M2MRUN: <file_namen>]]**<br>where:<br>**<file_namen>** – file name, quoted string type (max 16 chars, case sensitive) | |
| **AT#M2MRUN=?** | Test command returns the allowed values for parameter **<mode>**. | |
| Example | AT#M2MRUN =2,"M2MAPZ.bin"<br>OK<br><br>AT#M2MRUN?<br>#M2MRUN: "M2MAPZ.bin"<br>OK | |

### 8.1.7 AT#M2MCHDRIVE

| **#M2MCHDRIVE – M2M File System Change Current Drive** | | **SELINT 2** |
|---|---|---|
| **AT#M2MCHDRIVE=<drive>** | Set command sets the current drive in the M2M file system.<br>Parameter:<br>**<drive>** – integer type, current drive integer value.<br>Note: the only available drive value in the M2M file system is 0. | |
| **AT#M2MCHDRIVE?** | Read command reports the current drive in the M2M file system in the format:<br>**#M2MCHDRIVE: <drive>** | |
| **AT#M2MCHDRIVE=?** | Test command returns the allowed values for parameter **<drive>**. | |
| Example | AT#M2MCHDRIVE?<br>#M2MCHDRIVE: 0<br>OK | |

### 8.1.8 AT#M2MCHDIR

| **#M2MCHDIR – M2M File System Change Current Directory** | | **SELINT 2** |
|---|---|---|
| **AT#M2MCHDIR=<path_name>** | Set command sets the current working directory in the current drive in the M2M file system.<br>Parameter:<br>**<path_name>** – directory name, quoted string type (up to max 16 chars depending on current working directory, case sensitive) or relative path name, quoted string type (up to max 124 chars depending on current working directory, case sensitive) or absolute path name, quoted string type (max 124 chars, case sensitive)<br>Note: the directory name, relative path name or absolute path name should be passed between quotes; directory and path names are case sensitive.<br>Note: path separator can be either \ or /.<br>Directory name begins with a character different from path separator and is relative to the current working directory.<br>Relative path name begins with a character different from path separator and is relative to the current working directory.<br>Absolute path name begins with path separator.<br>System max path name length (current directory name length + file name length) is 128.<br>System reserves 2 characters for internal use.<br>Note: if the directory name, relative path name or absolute path name **<path_name>** is not present an error code is reported.<br>Note: the current directory in the drive 0 in the M2M file system at every power on is \. | |
| **AT#M2MCHDIR?** | Read command reports the current working directory in the current drive in the M2M file system in the format:<br>**#M2MCHDIR: <path_name>**<br>Where:<br>**<path_name>** – absolute path name, quoted string type (max 124 chars, case sensitive)<br>Note: path separator used in this report is \. | |
| **AT#M2MCHDIR=?** | Test command returns **OK** result code. | |
| Example | AT#M2MCHDIR?<br>#M2MCHDIR: "\MOD"<br>OK<br><br>AT#M2MCHDIR="dir1"<br>OK<br><br>AT#M2MCHDIR?<br>#M2MCHDIR: "\MOD\dir1"<br>OK | |

### 8.1.9 AT#M2MMKDIR

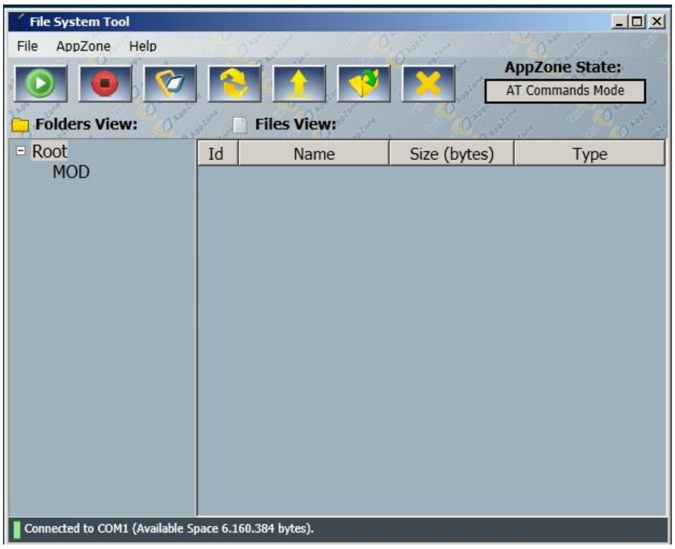
| **#M2MMKDIR – M2M File System Make Directory** | | **SELINT 2** |
|---|---|---|
| AT#M2MMKDIR=<dir_name> | Set command makes a new directory in the current working directory in the M2M file system.<br>Parameter:<br>**<dir_name>** – directory name, quoted string type (up to max 16 chars depending on current working directory, case sensitive)<br>Note: the directory name should be passed between quotes; directory names are case sensitive. | |
| **AT#M2MMKDIR=?** | Test command returns **OK** result code. | |
| Example | AT#M2MMKDIR="dir1"<br>OK | |

### 8.1.10 AT#M2MRMDIR

| **#M2MRMDIR – M2M File System Remove Directory** | | **SELINT 2** |
|---|---|---|
| **AT#M2MRMDIR=<dir_name>** | Set command removes the directory from the current working directory in the M2M file system.<br>Parameter:<br>**<dir_name>** – directory name, quoted string type (max 16 chars, case sensitive)<br>Note: the directory name should be passed between quotes; directory names are case sensitive.<br>Note: if the directory **<dir_name>** is not present in the current working directory, an error code is reported.<br>Note: if the directory **<dir_name>** is not empty, it is not  possible to remove it and an error code is reported. | |
| **AT#M2MRMDIR=?** | Test command returns **OK** result code. | |
| Example | AT#M2MRMDIR="dir1"<br>OK | |

## 8.2 File System Tool

Telit provides a tool to inspect the file system of the module and download the user M2M application. To have information on the tool refer to document [5].



AT Commands Mode

Connected to COM1 (Available Space 6.160.384 bytes).

Fig. 15: File System Tool

## 8.3 Compilers Installation

To setup AppZone environment you need all the subsequent items:

- TelitAppZone Installation Package
- Module with AppZone Layer
- Compiler & licenses. You can use one of the two solutions:

Solution A

- RVCT compiler

- License file (license.dat or license.lic file usually downloaded from ARM after registration)

Solution B

- GCC compiler

- Licenses, refer to Appendix: 8.4 Third Party Licenses

Here are the main steps:

- Install the TelitAppZone from the Installation Package, refer to chapter 3.1
- Follow the steps described in the next pages.

### 8.3.1 How to Install the RVCT Compiler

TelitAppZone Package does not provide the RVCT 3.0 SP1 compiler, so you have to install it. Usually, you have to get the compiler from ARM Web site (after registration).

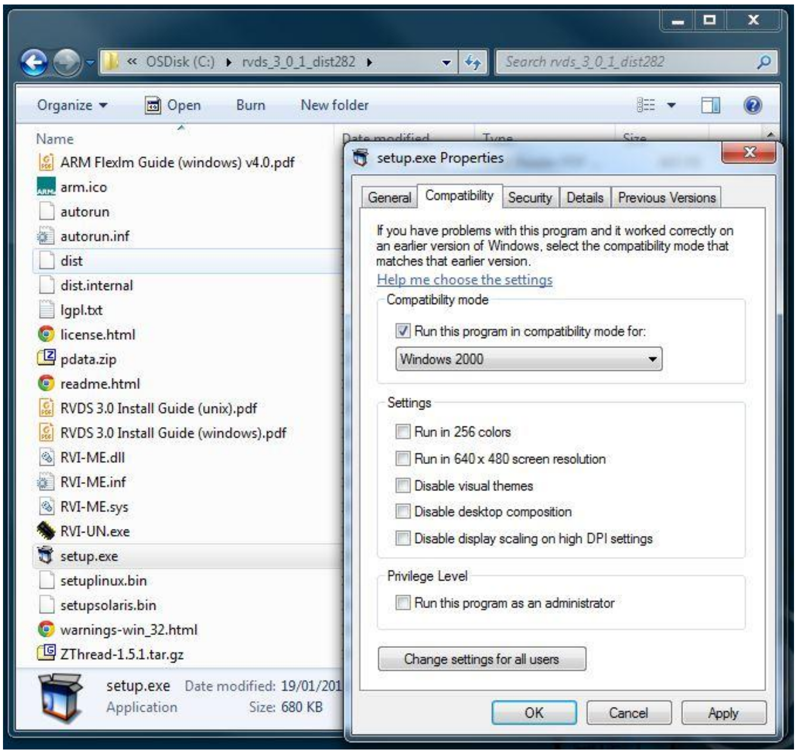
a) Download the RVCT 3.0 SP1 compiler from ARM Web site following the next instructions:

- Register to the following link: https://login.arm.com/register.php
- Send a e-mail to license.support@arm.com asking for a RVCT3.0 evaluation S/N
- Wait for the answer e-mail containing the S/N.
- Download from https://silver.arm.com/browse/RVS30 the "RVDS 3.0 SP1 Binary File".

b) After extracting the downloaded files in a folder on PC (suggested on C:\), and before running the extracted setup.exe utility, follow these steps in accordance with the used Operating System.

*Windows Vista and Windows 7*

Right click on the setup.exe file → Properties → Compatibility then, choose to run in compatibility mode as “Windows 2000”.



setup.exe Properties

If you have problems with this program and it worked correctly on an earlier version of Windows, select the compatibility mode that matches that earlier version.

Compatibility mode

☑ Run this program in compatibility mode for: Windows 2000

Settings

□ Run in 256 colors

□ Run in 640 x 480 screen resolution

□ Disable visual themes

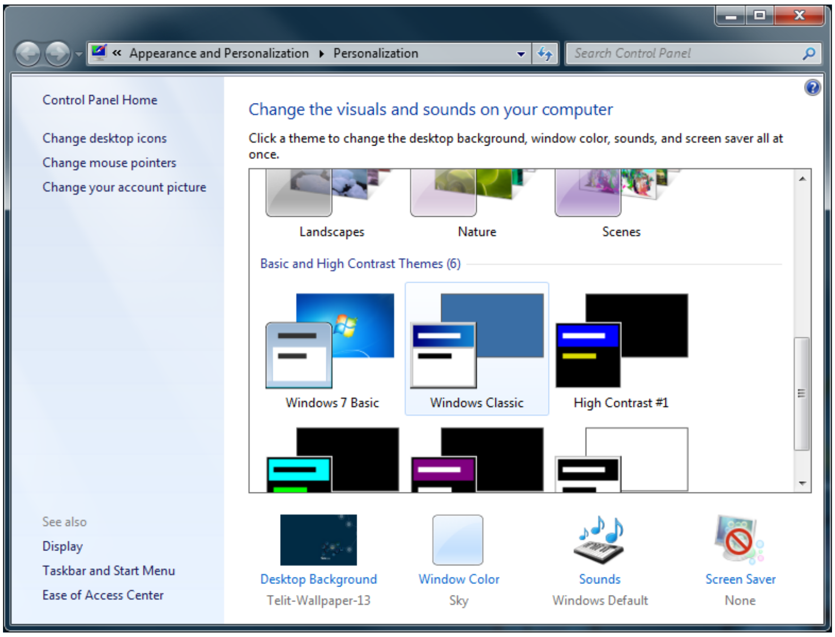
□ Disable desktop composition

□ Disable display scaling on high DPI settings

Privilege Level

□ Run this program as an administrator

If it is not possible, then right click on desktop → Personalization, and choose “Windows Classic”. Do not change this setting until the setup is completed!



Change the visuals and sounds on your computer

Click a theme to change the desktop background, window color, sounds, and screen saver all at once.

Desktop Background: Telit-Wallpaper-13

Window Color: Sky

Sounds: Windows Default

Screen Saver: None

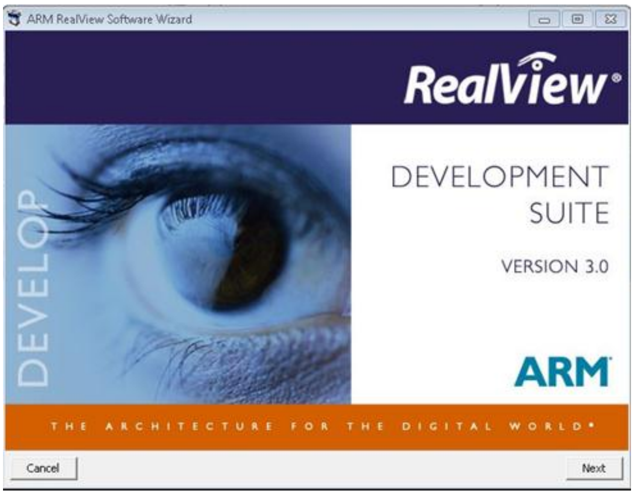
The length of PATH environment variable must be less than 1K, otherwise there could be an error and environment ARM variables will not be written causing compiler errors.

*Windows 8*

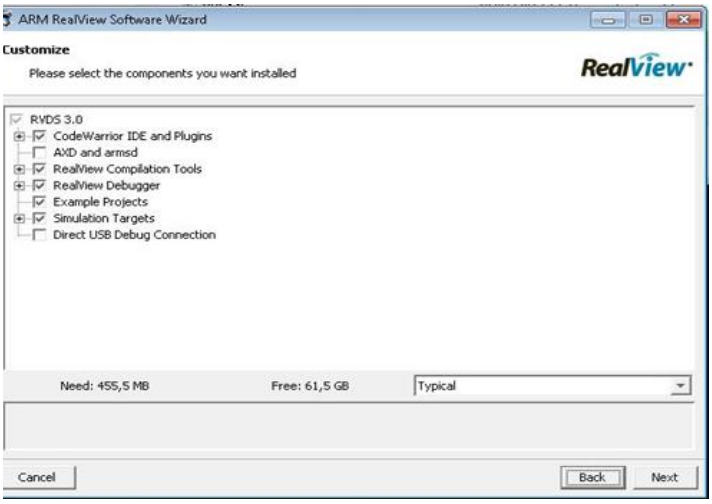
Use the following instructions (Installation from command line):

- Open command prompt window (run it as administrator)
- Go to the directory:
  "Utilities\Installer\<Installer Version>\<Build Number>\win_32-pentium"
- Execute the following command:
  **wh.exe install -p RVDS –source <source folder path> –target <target folder path> –env SYSTEM**
  **for example, wh.exe install -p RVDS –source C:\rvds_3_0_1_dist82 –target C:\ARM –env SYSTEM**
- Accept license agreement when required
- When the compiler installation is ended, run the “License Wizard” as administrator, and follow steps e) to j) shown below.

c) Now, you can execute setup.exe utility. ARM RealView Software Wizard will open, click on “Next” button, and accept the required license agreement.


ARM RealView Software Wizard
RealView
DEVELOP
DEVELOPMENT SUITE
VERSION 3.0
ARM
THE ARCHITECTURE FOR THE DIGITAL WORLD
Cancel
Next

d) Choose “Typical” installation type, then click “Next” button to start the installation.


ARM RealView Software Wizard

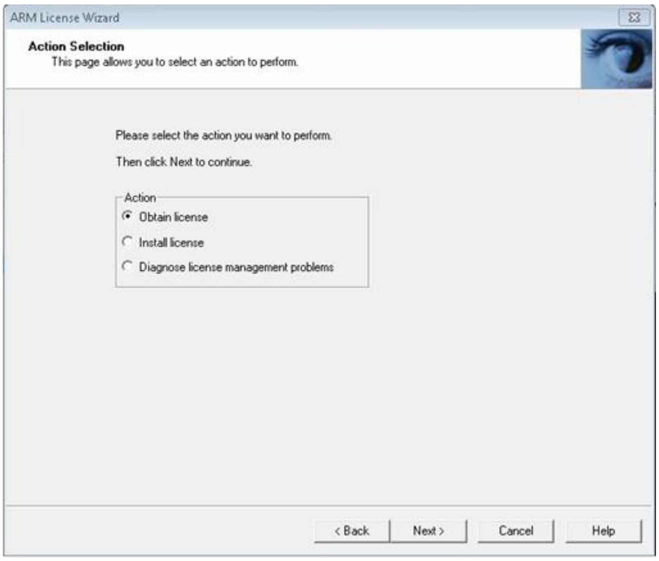
**Customize**

Please select the components you want installed

☑ RVDS 3.0
- ☑ CodeWarrior IDE and Plugins
- □ AXD and armsd
- ☑ RealView Compilation Tools
- ☑ RealView Debugger
- ☑ Example Projects
- ☑ Simulation Targets
- □ Direct USB Debug Connection

Need: 455,5 MB Free: 61,5 GB Typical

e) Choose “Obtain License” instead of “Install License”


ARM License Wizard

**Action Selection**

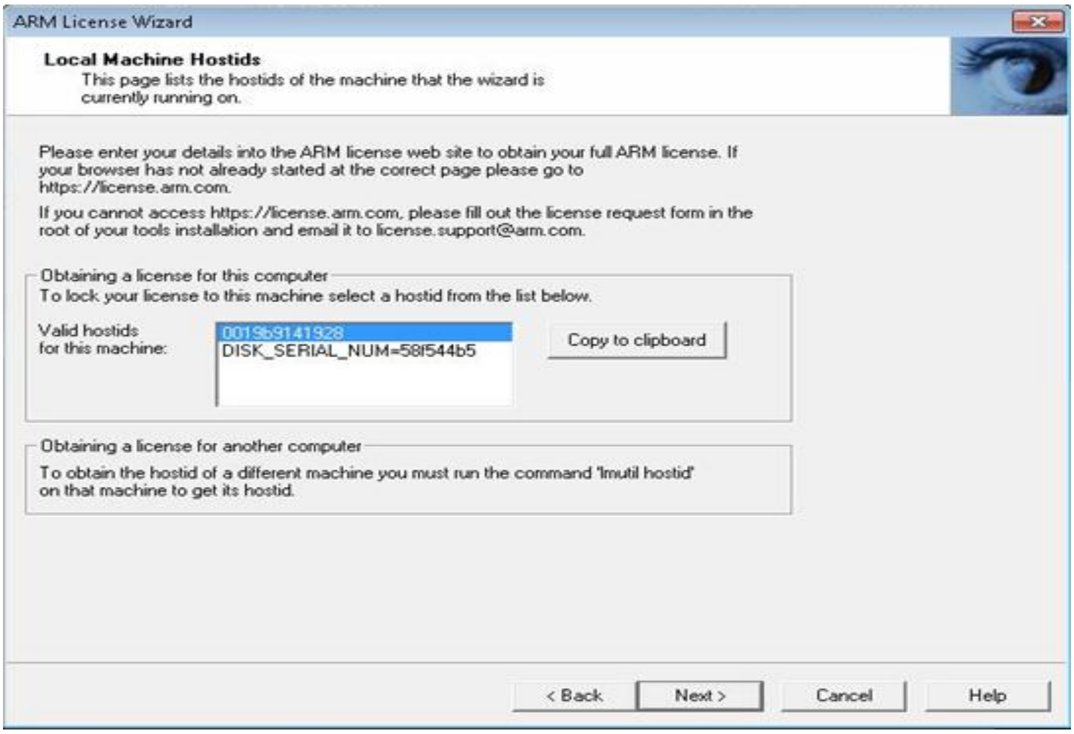
This page allows you to select an action to perform.

Please select the action you want to perform.

Then click Next to continue.

Action
- ● Obtain license
- ○ Install license
- ○ Diagnose license management problems

f) When prompted to choose disk or network interface to be associate to the license, pay attention to choose the primary network interface address (not any virtual one) or the hard disk serial number, then click on “Copy to Clipboard” button.



**Local Machine Hostids**

This page lists the hostids of the machine that the wizard is currently running on.

Please enter your details into the ARM license web site to obtain your full ARM license. If your browser has not already started at the correct page please go to https://license.arm.com.

If you cannot access https://license.arm.com, please fill out the license request form in the root of your tools installation and email it to license.support@arm.com.

Obtaining a license for this computer

To lock your license to this machine select a hostid from the list below.

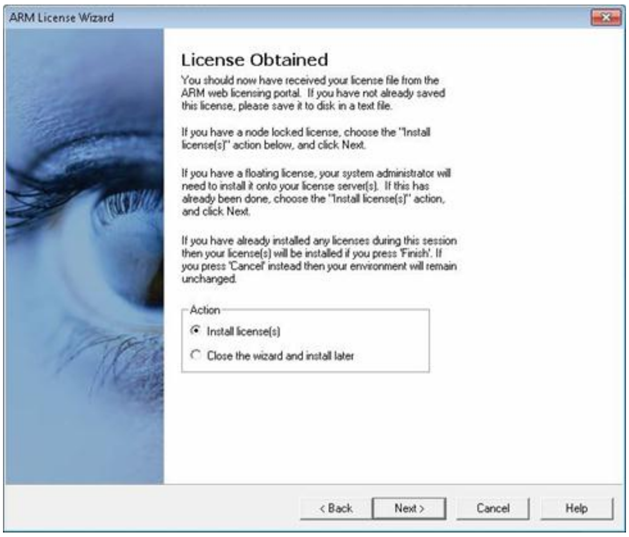
Valid hostids for this machine:

0019b9141928
DISK_SERIAL_NUM=58f544b5

Obtaining a license for another computer

To obtain the hostid of a different machine you must run the command 'lmutil hostid' on that machine to get its hostid.

g) Choose “Install license”, then click "Next" button.



**License Obtained**

You should now have received your license file from the ARM web licensing portal. If you have not already saved this license, please save it to disk in a text file.

If you have a node locked license, choose the "Install license(s)" action below, and click Next.

If you have a floating license, your system administrator will need to install it onto your license server(s). If this has already been done, choose the "Install license(s)" action, and click Next.

If you have already installed any licenses during this session then your license(s) will be installed if you press 'Finish'. If you press 'Cancel' instead then your environment will remain unchanged.

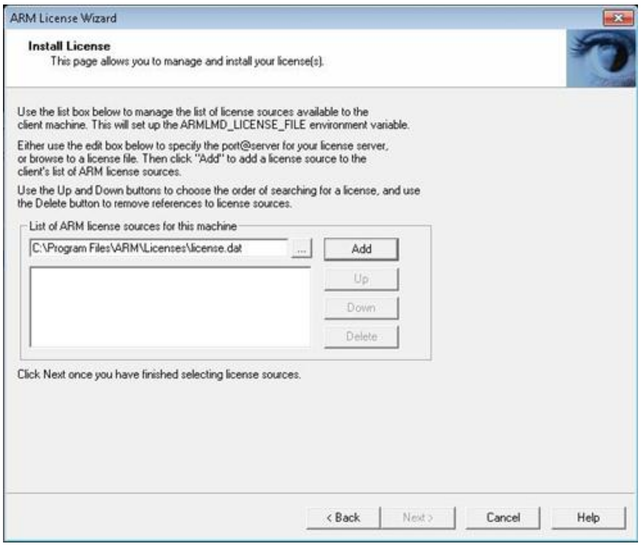
Action

● Install license(s)

○ Close the wizard and install later

h) Go to https://silver.arm.com/licensing/generate.tm execute the login, refer to step a), paste the MAC address, previously copied, and follow the instructions to obtain the file “License.dat” containing the license.

i) Set the License.dat file with its path (refer to previous step). Click on “Add” button, "Next" button, and then on "Finish" button.



ARM License Wizard

**Install License**

This page allows you to manage and install your license(s).

Use the list box below to manage the list of license sources available to the client machine. This will set up the ARMLMD_LICENSE_FILE environment variable.

Either use the edit box below to specify the port@server for your license server, or browse to a license file. Then click "Add" to add a license source to the client's list of ARM license sources.

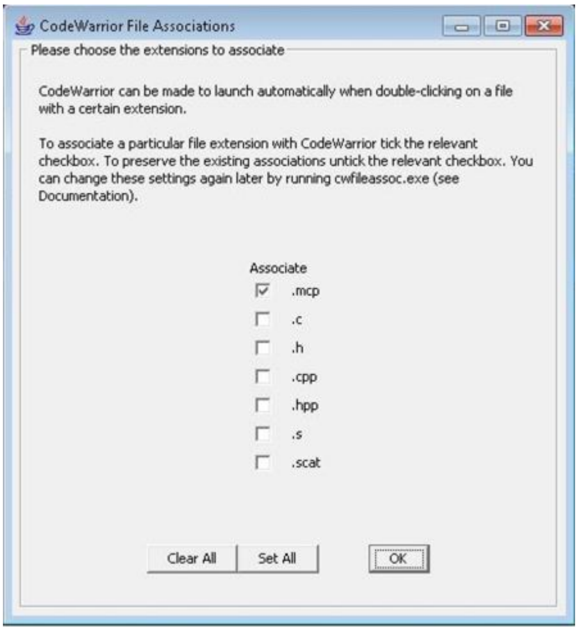
Use the Up and Down buttons to choose the order of searching for a license, and use the Delete button to remove references to license sources.

List of ARM license sources for this machine

C:\Program Files\ARM\Licenses\license.dat

Click Next once you have finished selecting license sources.

j) On “Code Warrior” window, you can choose “Clear All” button, and then click “OK”. On the next window, click “Finish” buttons to complete installation.



CodeWarrior File Associations

Please choose the extensions to associate

CodeWarrior can be made to launch automatically when double-clicking on a file with a certain extension.

To associate a particular file extension with CodeWarrior tick the relevant checkbox. To preserve the existing associations untick the relevant checkbox. You can change these settings again later by running cwfileassoc.exe (see Documentation).

Associate

☑ .mcp

□ .c

□ .h

□ .cpp

□ .hpp

□ .s

□ .scat

### 8.3.2 How to Install the GCC Compiler

TelitAppZone Package without GCC compiler.

a) Download “gcc-arm-none-eabi-4_9-2014q4-20141203-win32.zip(md5)” file the Web site: http://launchpad.net/gcc-arm-embedded/4.9/4.9-2014-q4-major

b) Unzip the file inside ...Telit\TelitAppZone\AppZoneTools\Eclipse\arm_gcc493 folder of the SDK; refer to chapter 3.1. After unzipping, on arm_gcc493 folder you have to find the following folders:
\arm-none-eabi
\bin
\lib
\share (optional)

Now, you are ready to compile the project.

## 8.4 Third Party Licenses

Telit AppZone Development Environment (ADE) contains the following Third Parties Software:

| Telit product | Third Party Software | Description | Owner and license |
|---|---|---|---|
| Telit ADE | Eclipse framework | Eclipse development environment | http://www.eclipse.org |
| Telit ADE | GCC compiler | Gnu compiler collection | http://gcc.gnu.org |

**NOTICE**: The SDK contains libraries libc_3G.ar and libc_2G.ar. All of the source code of libc_3G.ar e libc_2G.ar is under licenses which are both free and open source. The two libraries have been generated recompiling some subdirectories of the *newlib* subdirectory that is a collection of software from several sources. The *newlib* source code is available here: https://launchpad.net/gcc-arm-embedded/4.9/4.9-2014-q4-major @ gcc-arm-none-eabi-4_9-2014q4-20141203-src.tar.bz2 (md5). Refer to the license.txt file, Part 9), in https://launchpadlibrarian.net/192227637/license.txt, for terms and conditions.

LIBC_3G.AR AND LIBC_2G.AR ARE PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR 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 THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 9 DOCUMENT HISTORY

| Revision | Date | Changes |
|---|---|---|
| 0 | 2013-08-01 | First issue |
| 1 | 2013-09-26 | Warning: the current revision is mainly addressed for 13.00.xx5–B003 software version.<br>The Applicability Table has been updated.<br>The figures shown in chapters 3.1, 3.1.2, 3.1.3 have been updated |
| 2 | 2014-06-20 | The Applicability Table has been updated with HE910 module |
| 3 | 2015-02-16 | Updated:<br>- All the figures.<br>- Applicability Table:<br>HE910 / 12.00.xx6<br>UE910-XXX / 12.00.xx6<br>GE910-XXX / 13.00.xx7<br><br>Added:<br>- Services Coexistence Table<br>- Appendixes 7, 8, 9, 10 |
| 4 | 2015-05-07 | Updated:<br>- Template of the User Guide<br>- Appendixes Structure<br><br>Added:<br>- Chapter 5: ADE AT Controller<br>- M2M_onUSbCableEvent(...), chapter 3.2.5<br>- M2M_onGprsRegStatusEvent(...), chapter 3.2.6<br>- M2M_onIP6RawEvent(...), chapter 3.2.6 |

Design Review
120°
180°
Process Validation
RF/EMC Pre-Certification
240°
Module Selection
60°
GE864-QUAD V2
Telit
300°
Approvals Support
360°
Value Added Services & Connectivity
Enterprise Integration &
Application Enablement
m2mair