



MediaTek LinkIt™ Development Platform for RTOS Get Started Guide

Version: 1.1

Release date: 20 June 2018

© 2015 - 2018 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc. ("MediaTek") and/or its licensor(s). MediaTek cannot grant you permission for any material that is owned by third parties. You may only use or reproduce this document if you have agreed to and been bound by the applicable license agreement with MediaTek ("License Agreement") and been granted explicit permission within the License Agreement ("Permitted User"). If you are not a Permitted User, please cease any access or use of this document immediately. Any unauthorized use, reproduction or disclosure of this document in whole or in part is strictly prohibited. THIS DOCUMENT IS PROVIDED ON AN "AS-IS" BASIS ONLY. MEDIATEK EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF ANY KIND AND SHALL IN NO EVENT BE LIABLE FOR ANY CLAIMS RELATING TO OR ARISING OUT OF THIS DOCUMENT OR ANY USE OR INABILITY TO USE THEREOF. Specifications contained herein are subject to change without notice.

Document Revision History

Revision	Date	Description
1.0	10 Jan 2018	Initial release
1.1	20 Jun 2018	Upgrade GCC compiler version

Table of Contents

1.	Overview	1
1.1.	Architecture of the platform	1
1.2.	Supported key components	2
1.2.1.	NB-IoT Modem	2
1.2.2.	Network	3
1.2.3.	Sensor subsystem	4
1.2.4.	GNSS	4
1.2.5.	FOTA	4
1.2.6.	Peripheral drivers	5
1.2.7.	Advanced features and components	6
1.3.	Folder structure	7
1.4.	Project source structure	10
2.	Getting Started Using GCC	11
2.1.	Environment	11
2.2.	Developing on MT2625 platform	11
2.2.1.	Installing IoT Flash Tool for MT2625 platform	11
2.2.2.	Flashing the image to MT2625 platform	11
2.2.3.	Genie Tool for logging on MT2625 platform	12
2.2.4.	Debugging with MT2625 platform from Microsoft Windows	15
2.3.	Building the project using the SDK	18
2.3.1.	Installing the SDK build environment on Linux	18
2.3.2.	Installing the SDK build environment on Microsoft Windows	19
2.3.3.	Methods to build a project	24
2.4.	Create your own project	27
2.4.1.	Using an existing project	27
2.4.2.	Removing a module	28
2.4.3.	Add the source and header files	28
3.	Appendix A: Acronyms and Abbreviations	30
4.	Appendix B: Disabling Automatic Driver Installation on Windows OS	31

Lists of Tables and Figures

Table 1. NB-IoT modem main features	2
Table 2. Supported network protocols	3
Table 3. Sensor subsystem features	4
Table 4. GNSS features	4
Table 5. FOTA features	4
Table 6. Supported peripheral drivers	5
Table 7. Advanced features and components	6
Table 8. Recommended build environment	18
Table 9. Acronyms and Abbreviations	30
Figure 1. Architecture layout of the platform	1
Figure 2. Folder structure	8
Figure 3. Project folder structure	10
Figure 4. Download the firmware to MT2625 platform by use USB connection	12
Figure 5. Genie configuration	13
Figure 6. Running genie	14
Figure 7. Quick Genie log package creation for sharing with others	14
Figure 8. Dump information in TXT viewer tool	15
Figure 9. Folder structure snapshot for extracting dump file	15
Figure 10. Output folder structure snapshot	15
Figure 11. Call stack in Trace32 simulator	16
Figure 12. Error dialog for elf and load mismatching	16
Figure 13. Mini dump assertion information	17
Figure 14. Context switch information snapshot	17
Figure 15. MinGW Installation Manager Setup Tool	19
Figure 16. Keep default installation preferences	20
Figure 17. Download and set up MinGW Installation Manager	21
Figure 18. Basic setup on MinGW Installation Manager	21
Figure 19. A basic MinGW installation	22
Figure 20. Schedule of pending actions	22
Figure 21. Applying scheduled changes	22
Figure 22. MinGW folder structure	23
Figure 23. tools\gcc folder structure	24
Figure 24. Modify the Makefile under the GCC folder of my_project	27
Figure 25. Project source and header files under the project folder	29
Figure 26. Disabling automatic driver updates on Windows 7 OS	31
Figure 27. Disabling automatic driver updates on Windows 8 and 10	32

1. Overview

MediaTek MT2625 platform provides the software and tools for your application development on MT2625 chipset. The SDK includes drivers for hardware abstraction layer, peripherals, connectivity, such as NB-IoT modem, GNSS, sensor subsystem, lightweight IP (lwIP) and other third party features. It also provides NVDM, Firmware update Over-The-Air (FOTA) and FreeRTOS.

This get started guide provides quick steps on how to use the SDK and its supported features on GCC environment.

1.1. Architecture of the platform

The three-layer architecture of the platform including **BSP**, **Middleware** and **Application** with underlying components is shown in Figure 1.

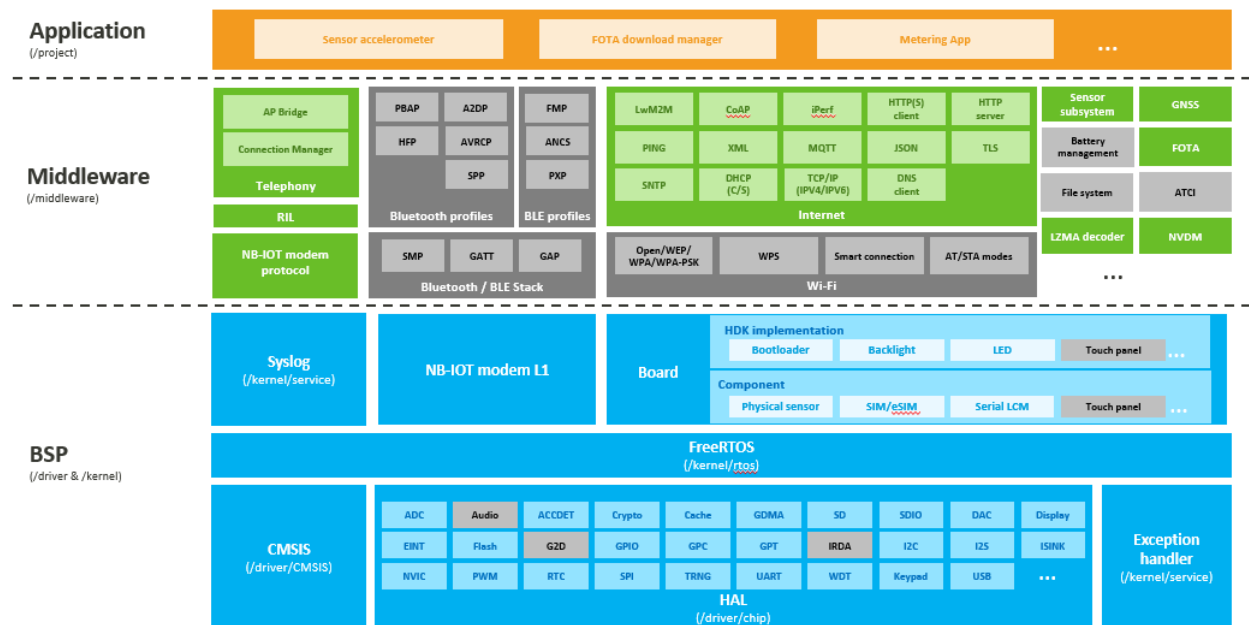


Figure 1. Architecture layout of the platform

A brief description of the layers is provided below:

- **BSP**
 - Hardware drivers. Provide peripheral drivers for the platform, such as ADC, I2S, I2C, SPI, RTC, GPIO, UART, Flash, Security Engine, TRNG, GDMA, PWM, WDT and IRDA TX/RX.
 - Hardware Abstraction Layer (HAL). Provides the driver Application Programming Interface (API) encapsulating the low-level functions of peripheral drivers for the operating system (OS), **Middleware** features and **Application**.
 - The hardware components located at <sdk_root>\driver\board\component are used by the HW (<sdk_root>\driver\board\mtxxxx). For example, the LCM drivers of ST7789H2 and ST7301 are located under <sdk_root>\driver\board\component\lcm folder. The display driver is located under <sdk_root>\driver\board\mt2625_hdk\lcd, the example code uses ST7789H2 LCM on MT2625 platform.
 - [FreeRTOS](#). An OS with the open source software for **Middleware** components and **Application**.

- Syslog. This module implements system logging for development and debugging.
- **Middleware**
 - NB-IoT modem. Provides NB-IoT network access capability, including both R13 and R14 features.
 - Network. Provides OS dependent features, such as IPv4, Hyper-Text Transfer Protocol (HTTP) client and the Simple Network Time Protocol (SNTP).
 - Sensor subsystem. Software framework to interact with sensor drivers and fusion algorithms, including buffer and flow control.
 - FOTA. Provides a mechanism to update the firmware.
 - GNSS. Provides APIs to control the onboard GNSS system.
 - Other features. Non-Volatile Data Management (NVDM), Extensible Markup Language (XML), JavaScript Object Notation (JSON) and other features that are dependent on **HAL** and **FreeRTOS**.
- **Application**
 - Pre-configured projects using **Middleware** components, such as meter application with sensor data reporting demonstration.

The application layer enables running the projects that are based on **Middleware**, **FreeRTOS** and **HAL** layers. These layers provide rich features for application development, such as **Middleware** provides the network features, NB-IoT modem, and the OS provides the underlying real-time operating system.

To use the HAL features, enable the compile options of the corresponding modules.

- 1) Open the header file `hal_feature_config.h`, located under `inc` folder of each example project.
- 2) Edit and define the compile options as needed.
- 3) Include the corresponding module header files, located at `<sdk_root>\driver\chip\inc`, in the project source files.

1.2. Supported key components

The platform offers rich connectivity options, such as NB-IoT, network, GNSS, peripheral drivers and other advanced components. This section introduces each of these components.

1.2.1. NB-IoT Modem

To include NB-IoT modem, include `<sdk_root>\prebuilt\middleware\MTK\nbiot\module.mk` in project makefile for 2625 projects.

Table 1. NB-IoT modem main features

Item	Features
NB-IoT modem	<ul style="list-style-type: none"> ● PSM/eDRX ● Coverage level selection ● Paging ● 20 dB coverage gain ● IP/non-IP ● Single tone ● Multi-cast (SC-PTM, R14) ● Data over user-plane

Item	Features
	<ul style="list-style-type: none"> Control plane EPS ClOT optimization Attach without PDN NAS security RoHC RRC connection re-establishment RRC connection re-configuration UICC, USIM User plane IoT optimization R14 enhancement

1.2.2. Network

The internet middleware APIs can be found in the Internet Middleware API Reference Manual and LinkIt for RTOS Open Source Components Guide under `<sdk_root>\doc`. Supported network features of the platform are listed in Table 2. Learn how to include each supported protocol module from `<sdk_root>\middleware\third_party\xxx\readme.txt`.

Table 2. Supported network protocols

Item	Features
IP Stack	<ul style="list-style-type: none"> IPv4 (LWIP) TCP, UDP ICMP DHCP Client/Server DNS Client NETCONN SOCKET
SNTP	<ul style="list-style-type: none"> Simple Network Time Protocol RFC4330 Support SNTP receive timeout Support SNTP update delay Support SNTP max server
HTTP	<ul style="list-style-type: none"> HTTP 1.1 Client (POST/GET)
HTTPS	<ul style="list-style-type: none"> HTTP 1.1 Client (POST/GET)
SSL/TLS	<ul style="list-style-type: none"> mbed TLS Client, Server (not tested) SSL3.0, TLS1.0, 1.1, 1.2 AES, 3DES, DES, ARC4 MD5, SHA-1, SHA-256 RSA/PKCS#1 v1.5

1.2.3. Sensor subsystem

Supported sensor subsystem features of the SDK are listed in Table 3. More information on the sensor subsystem SDK APIs can be found in sensor subsystem section of the 2625 API Reference Manual under <sdk_root>\doc. In addition, find more details on how to include the sensor subsystem module in <sdk_root>\middleware\MTK\sensor_subsys\readme.txt.

Table 3. Sensor subsystem features

Item	Features
Physical sensor	<ul style="list-style-type: none"> Accelerometer Biosensors (PPG, EKG)
Sensor fusion	<ul style="list-style-type: none"> Heart rate Blood pressure

1.2.4. GNSS

The detailed list of GNSS features is provided in Table 4. The API and module descriptions can be found in API Reference Manual and LinkIt for RTOS GNSS Developer's Guide under <sdk_root>\doc. In addition, find more details on how to use this module in <sdk_root>\middleware\MTK\gnss\readme.txt.

Table 4. GNSS features

Item	Features
GNSS	<ul style="list-style-type: none"> GPS, BeiDou, GLONASS Low Power Mode Periodic mode GLP mode Time Aiding, Location Aiding
Extended Prediction Orbit (EPO)	<ul style="list-style-type: none"> EPO host aiding EPO download using NB-IoT network

1.2.5. FOTA

The detailed list of FOTA features is provided in Table 5. The API and module descriptions can be found in LinkIt SDK API Reference Manual and LinkIt for RTOS Firmware Update Developer's Guide under <sdk_root>\doc. In addition, find more information on how to include this module in <sdk_root>\middleware\MTK\fota\readme.txt.

Table 5. FOTA features

Item	Features
FOTA	<ul style="list-style-type: none"> Full binary update mechanism Package data compression Integrity check Power loss protection FOTA packaging tool

1.2.6. Peripheral drivers

The detailed list of peripheral drivers is provided in Table 6. The APIs for the drivers can be found in the LinkIt SDK API Reference Manual under <sdk_root>\doc. To include HAL module, include <sdk_root>\driver\chip\mt2625\module.mk in project makefile for 2625 projects.

Table 6. Supported peripheral drivers

Item	Features
ACCDDET	<ul style="list-style-type: none"> Accessory Detector. Detects plug-in/out of earphone based on the suggested circuit.
ADC	<ul style="list-style-type: none"> ADC module. DAC module.
CACHE	<ul style="list-style-type: none"> The maximum size of the cache is 32kB.
EINT	<ul style="list-style-type: none"> External interrupt controller. Processes the interrupt request from an external source or a peripheral device.
Flash	<ul style="list-style-type: none"> Supports execute in place (XIP) and programming flash by software. Default 4MB system in package (SiP) flash on MT2625 platform. Additional 1MB secured OTP on system in package (SiP) flash on MT2625 platform.
GPIO	<ul style="list-style-type: none"> GPIO mode (in or out) Set Pull Up/Down for GPIO IN mode
GPT	<ul style="list-style-type: none"> General Purpose Timer. Supports 32kHz and 1MHz clock sources, repeat and one-shot modes for timing events and delays in μs or ms.
PWM	<ul style="list-style-type: none"> Range is 256 duty cycles 32kHz, 2MHz, XTAL clock for PWM frequency reference
UART	<ul style="list-style-type: none"> Two full set (TX/RX) UART support. Four UART ports, two of them featuring hardware flow control. Baud rate of up to 921600.
I2C Master	<ul style="list-style-type: none"> Two I2C interfaces Supports 50/100/200/400kHz transmission rate
I2S Master	<ul style="list-style-type: none"> Supports 16-bit or 24-bit addressing I2S master is capable of servicing an external codec component. Supports 8/11.025/12/16/22.05/24/32/44.1/48 kHz sampling rates in mono or stereo mode.
ISINK	<ul style="list-style-type: none"> Current sink

Item	Features
	<ul style="list-style-type: none"> Adjustable backlight current.
MPU	<ul style="list-style-type: none"> Memory Protection Unit
MSDC/SDIO/eMMC	<ul style="list-style-type: none"> SD memory card specification version 2.0 eMMC specification version 4.41 or higher SDIO card specification version 2.0. 1 bit/4 bit data transfer mode @ max 52Mhz clock
IrDA	<ul style="list-style-type: none"> TX (NEC, RC5, RC6, pulse width) RX (RC5, pulse width)
GPC	<ul style="list-style-type: none"> General Purpose Counter Supports 1MHz pulse detection
WDT	<ul style="list-style-type: none"> Supports hardware, software watchdog Supports system reset
I2S-Slave	<ul style="list-style-type: none"> Supports 16-bit or 24-bit addressing Supports 8/11.025/12/16/22.05/24/32/44.1/48 kHz sampling rates in mono or stereo mode.
SPI-Master	<ul style="list-style-type: none"> Serial Peripheral Interface
RTC	<ul style="list-style-type: none"> Real-Time Clock
GDMA	<ul style="list-style-type: none"> General Purpose DMA
Security	<ul style="list-style-type: none"> SHA1, SHA2 (256, 384, 512), MD5, AES, 3DES
TRNG	<ul style="list-style-type: none"> Truly Random Number Generator Generates 32bit random number
Charger	<ul style="list-style-type: none"> Supports single-cell Li-Ion battery charging.
Keypad	<ul style="list-style-type: none"> Keypad scanner Supports 3x3 single/double key mode

1.2.7. Advanced features and components

The advanced features and components included in the platform are listed in Table 7.

Table 7. Advanced features and components

Item	Features
XML	<ul style="list-style-type: none"> Mini-XML Supports <ul style="list-style-type: none"> Entity GET/SET Index Search
JSON	<ul style="list-style-type: none"> cJSON JSON string parser
Connection manager	<ul style="list-style-type: none"> Manages the activation / deactivation of the PDP

Item	Features
	connections on MT2625 platform.
RIL	<ul style="list-style-type: none"> Provides more intuitive and convenient way to communicate with modem.
AP bridge	<ul style="list-style-type: none"> Enable AT commands and user data exchange between external application chip and application domain on MT2625 platform.
NVDM	<ul style="list-style-type: none"> Provides memory mechanism that retains its contents when the system power is turned off.

1.3. Folder structure

The SDK is delivered as a single package organized in a folder structure, as shown in Figure 2.

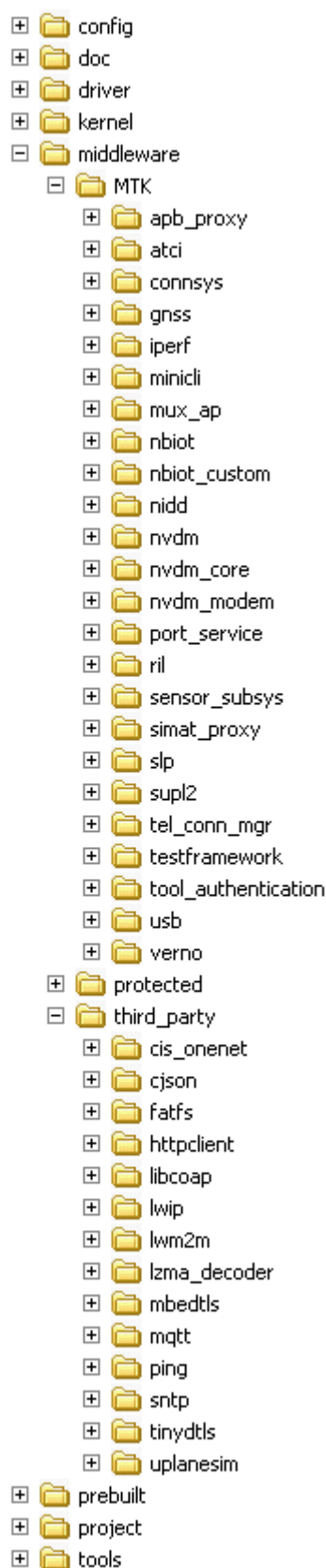


Figure 2. Folder structure

This package contains the source and library files of the major components, build configuration, related tools and documentation. A brief description on the layout of these files is provided below:

- **config.** Includes make and compile configuration files for compiling a binary project.
- **doc.** Includes SDK related documentation, such as developer and SDK API reference guides.
- **driver.** Includes common driver files, such as board drivers, peripheral and CMSIS-CORE interface drivers.
- **kernel.** Includes the underlying RTOS and system services for exception handling and error logging.
- **middleware.** Includes software features for HAL and OS, such as network and advanced features.
- **project.** Includes pre-configured example and demo projects using NB-IoT modem, HTTP, HAL, and more.
- **tools.** Includes tools to compile, download and debug projects using the SDK.

The main components that belong to middleware are in the **middleware** folder:

- **MTK**
 - **minicli.** A Command Line Interface (CLI) that provides a framework for the upper layer to register a function executed by an input command. The input command and output message streaming is communicated through the UART.
 - **nvdmm.** NVDM is a type of memory mechanism that retains its contents when the system power is turned off.
 - **atci.** Provides the interface for a target communication using AT commands through UART.
 - **nbiot.**
 - **nbiot_custom.**
 - **fota.** FOTA provides firmware update functionality.
 - **gnss.** Provides APIs to receive GNSS data and control the on-board GNSS module.
 - **sensor_subsys.** Provides sensor drivers and fusion algorithms.
- **third_party**
 - **cjson.** JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. cJSON is a single file implementation in C.
 - **dhcpcd.** DHCP daemon (DHCPD) is a program that operates as a daemon on a server to provide Dynamic Host Configuration Protocol (DHCP) service to a network. Devices in a network with a DHCP server can retrieve network parameter configuration from the server.
 - **httpclient.** The HTTP client is the client implementation for requesting data from HTTP servers.
 - **lwip.** A widely used open source TCP/IP stack designed for embedded systems. The focus of the lwIP TCP/IP implementation is to reduce resource usage while still having a full-scale TCP.
 - **mbedtls.** Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are cryptographic protocols designed to provide communications security over a computer network. mbed TLS is an open source implementation for developers to include cryptographic and SSL/TLS capabilities in embedded products with a minimal coding footprint.
 - **sntp.** SNTP is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks.
 - **xml.** XML is a markup language defined by the W3C's XML 1.0 Specification that defines a set of rules for encoding documents in a human-readable and machine-readable format.

1.4. Project source structure

The SDK provides a set of reference applications. For example, projects with a single function showing how to use drivers or other module features and others with complex functionality demonstrating how to use the middleware components.

Example applications are located in the `<sdk_root>\project\mt2625_evb\apps` and they all have the same folder structure, as shown in Figure 3.

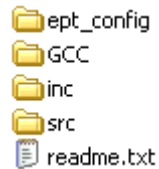


Figure 3. Project folder structure

- 1) `ept_config`. Stores the EPT tool configuration file with “.ews” as postfix.
- 2) `GCC`. GCC related project configuration files, such as a makefile.
- 3) `inc`. Project header files.
- 4) `src`. Project source files.
- 5) `Readme.txt`. A brief introduction about project behavior and the required environment.

You can apply the relevant reference applications to further your development.

2. Getting Started Using GCC

This section provides a guide to getting started with the LinkIt development platform for RTOS and covers the following items:

- Supported environments for development.
- Building the project using the SDK.
- Downloading and running the project from Microsoft Windows.
- Debugging the project from Microsoft Windows.
- Creating your own project.

2.1. Environment

The SDK can be used on any edition of Microsoft Windows XP, Vista, 7 and 8, and on Linux. A GCC compiler is required to build the project.

- Download and extract the content of the SDK package on your local PC.
- Follow the instructions in the `readme.txt` file to download and extract the SDK toolchain package.
 - Copy the GCC compiler ("`gcc`" folder) to `<sdk_root>\tools\`. The compiler settings are in the `<sdk_root>\.config` configuration file.

The default GCC toolchain is supported for the following versions of the Linux 32 or 64 bit hosts.

- Ubuntu 8.x or later (tarball).
- Ubuntu LTS 10.04 or later (PPA).
- RHEL 4/5/6 (tarball).

2.2. Developing on MT2625 platform

2.2.1. Installing IoT Flash Tool for MT2625 platform

IoT Flash Tool is a flexible device flashing tool for application development on MT2625 platform.

Please download IoT Flash Tool (folder name MT2625_IoT_Flash_Tool v2.6.2.1) from [here](#). The tool is a setup free package, and the `FlashTool.exe` inside the folder can be executed directly.

2.2.2. Flashing the image to MT2625 platform

Before using the IoT Flash Tool, it's required to have a pre-built project configure file (`.cfg`) or build your own project to get one (see 2.3, "Building the project using the SDK").

The following steps will show how to download the firmware into MT2625 target, by **USB** interface:

- 1) Plug in Power Jack.
- 2) Launch IoT Flash Tool, and click **Download** on the left panel of the main GUI.
- 3) Select **USB** from the **COM Port** drop down menu.
- 4) If you don't have the adapter or battery, check the **Enable Download without Battery** option.
- 5) Check Settings-> System-> **Disable Long-Press Power Key Setting** to avoid platform power off if long press power key.

5) Click **Open** to choose the configuration file, which is usually named as `flash_download.cfg` and is generated after build process. If it loads successfully, **Download Information** will be displayed, including **Name**, **Length** and **File Path** of the firmware binary, as shown in Figure 4.

6) Click **Start** to start downloading.

7) Plug in the USB cable and **long press Power Key** (until first progress bar complete) on 2625 platform. Then the process will start automatically.

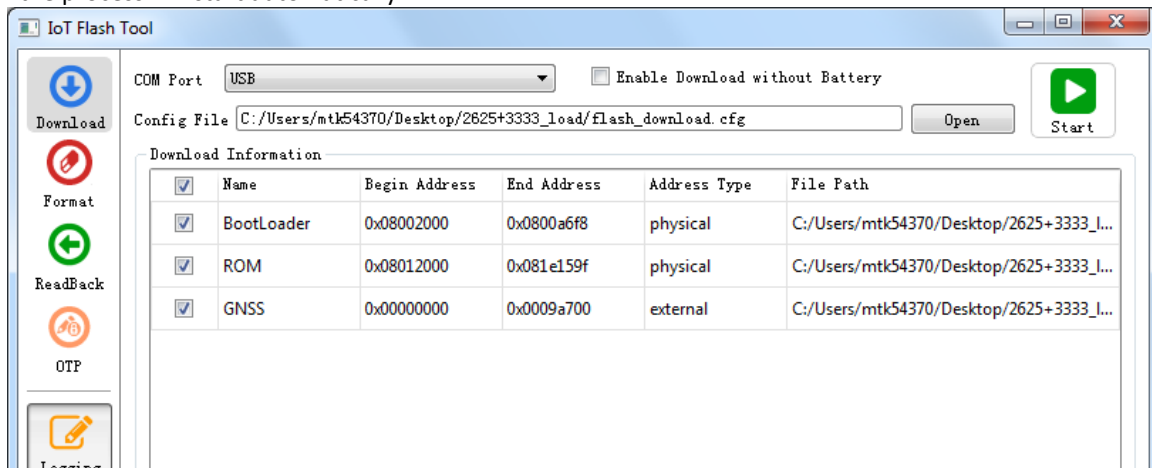


Figure 4. Download the firmware to MT2625 platform by use USB connection

2.2.3. Genie Tool for logging on MT2625 platform

Genie is a Windows application which provides logging for the nbioT project for two streams of data:

- GKI signals:
 - Modem inter-task communication.
 - APPs traces via the SIG_TEST_FILE_OUT signal.
- HSL traces: traces from any part of the system.

The two streams are independent of each other

- On PC, two separate serial communication ports are used (UART based or virtual if using USB).
- Both streams can be captured at the same time.
- It is possible to capture GKI data only, by disabling the HSL (High Speed Logging) stream.

2.2.3.1. Installation for stand-alone operations

- 1) Copy and unzip to a local drive (Genie logging tool provided by MTK) .
- 2) IMPORTANT: Do not use on a network drive as TeraHsl is a C# application for which Windows network drive restrictions can prevent its execution.

2.2.3.2. Genie Configuration

- 1) Open genie.exe. Path: nbioT_tools_customer_release_xxx\nbioT\tools\core\genie.

- 2) The Edit Configuration dialog box allows the user to configure the parameters for a Genie test. Dialog can be accessed by selecting the Edit Configuration option. Click on a tab below to view one of the GKI, Enable HSL logging, Baud Rate, Integrate RRC and ASN message decoder and Database. See Figure 5.

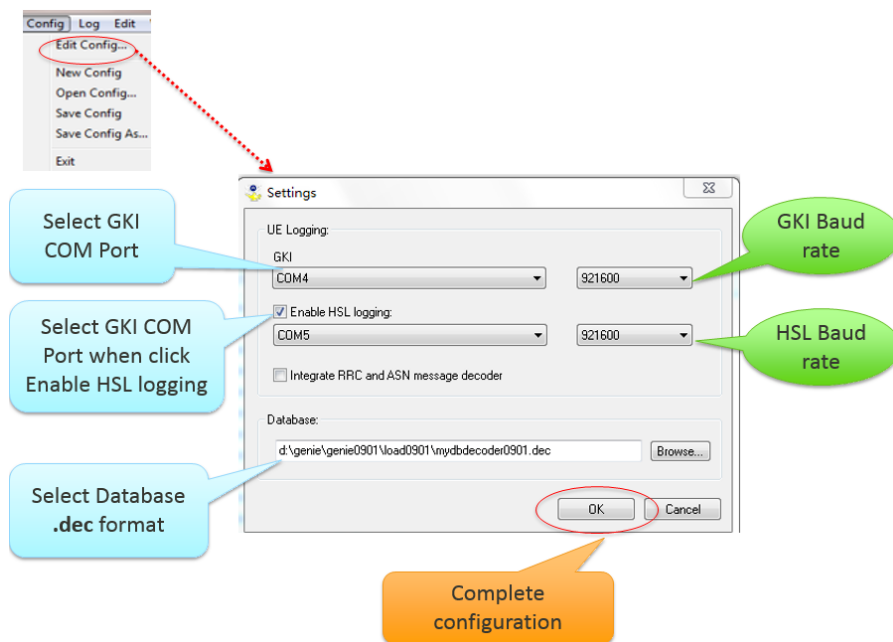



Figure 5. Genie configuration

Note: UART logging: GKI COM Port: UART1, HSL COM Port: UART2.

USB Logging: GKI COM Port: USB Modem Port, HSL COM Port: USB Debug Port.

You can change USB to UART port/UART to USB port by AT Command, please refer to UART USB Modem logging config SOP.pdf document in the genie tool package for more details.

2.2.3.3. Running Genie – capturing a log

After configurations are completed (and saved), the logging starts by clicking on  or key presses Ctrl-F9. Pressing again will stop logging.

If the UE is running and ready to connect, GKI/HSL will be captured and displayed. See Figure 6.

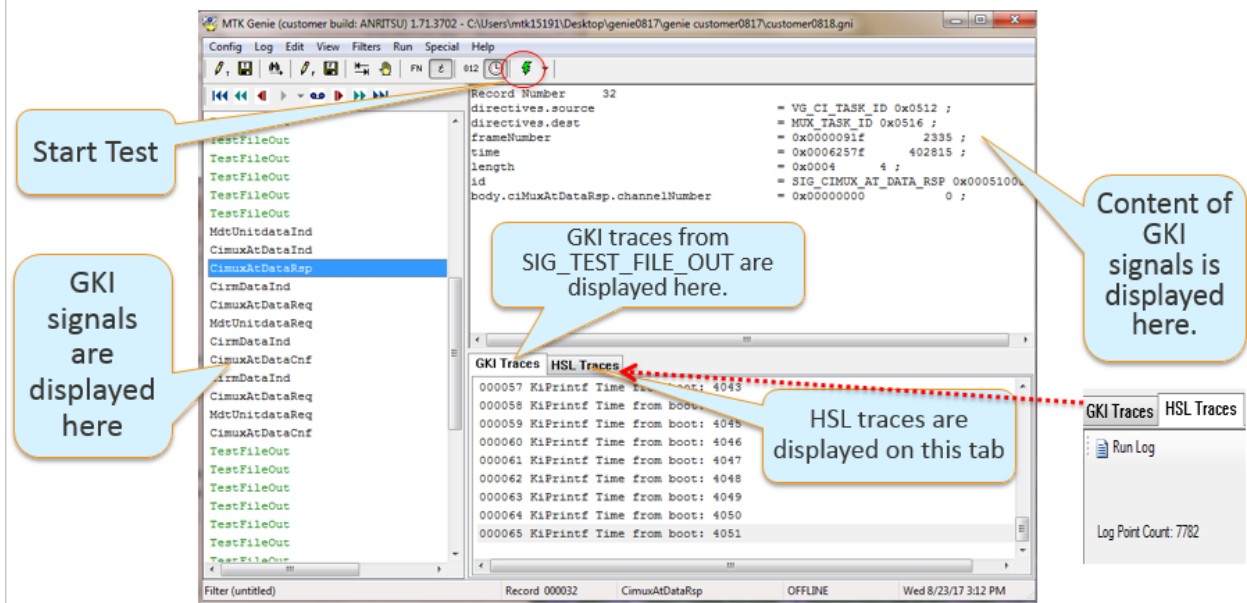


Figure 6. Running genie

2.2.3.4. Quick Genie log package creation for sharing with others

Follow these steps to save a log package which will put into one Genie *.glp all the necessary GKI/HSL logs/decoders applicable. See Figure 7.

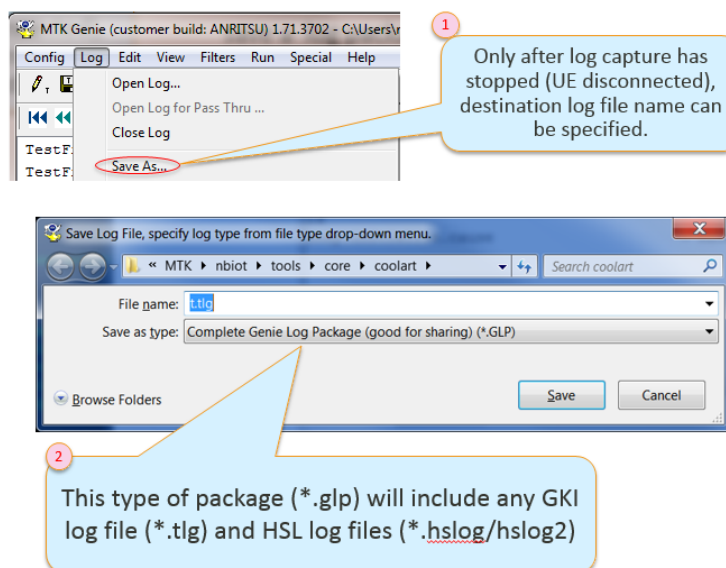


Figure 7. Quick Genie log package creation for sharing with others

- Share this single *.glp with other users.
- By creating a Windows application association with *.glp, it is then possible to double-click on these files to open them. Alternatively, use the Log/Open menu option.

This guide only covers the basics of Genie as a quick guide. Please refer to Genie documentation Genie Logging Tool.pdf and Genie_Logging_Tool_Users_Guide.pdf in the genie tool package for more details

2.2.4. Debugging with MT2625 platform from Microsoft Windows

This section provides detailed guideline on how to use dump log to analyze issue. There are two types of dump file. When genie is connected, we can get a core dump log (about 2Mb ~ 5Mb) via genie tool. (Please refer to section 2.2.3, “Genie Tool for logging on MT2625 platform”, for how to capture core dump). When no genie tool connected, no core dump is available, a small log (8k) called mini dump is saved in NVDM. It can be fetched out via NVDM Editor after next boot.

2.2.4.1. Deal with Core dump

We can open core dump file with TXT viewer tools. Then we can get the assert info in Figure 8:

```




?CORE DUMPING: Press 'D' to download log...?
?
?>>> dump syslog buffer
?[T: 135246 M: common C: info F: nvdm_modem_port_log_notice L: 141]: minidump_init:dump_num=5,dump_peb_num=2,curr_idx=5?
?PhyTimerCounterValue0 (previous latched value): 0x00000000f74fb30
?PhyTimerCounterValue1 (now latched value 1 ): 0x00000000f75e64d
?PhyTimerCounterValue2 (now latched value 2 ): 0x00000000f75e653
?assert failed: ( portNVIC_INT_CTRL_REG & portVECTACTIVE_MASK ) == 0, file: ../../../../../../kernel/rtos/FreeRTOS/Source/portable/GCC/mc2625/ARM_CM4F/port.c, line: 441
?
?In Hard Fault Handler
?SCB->HFSR = 0x40000000
?Forced Hard Fault
?SCB->CFSR = 0x01000082
?Usage fault: Unaligned access
  
```

Figure 8. Dump information in TXT viewer tool

Hard fault type is used as assert function. It means software find some error and do assert itself. It is the mostly assert type and we could find assert file and assert line in core dump file.

If we want see more detail about assert info, we need use some tools to analyze the dump file. Necessary tool: restore_dump_V1.4.6.2.exe; Trace32 simulation tool;


Step 1, put core dump log, elf file and restore_dump_V1.4.6.2.exe in the same folder as shown in Figure 9:


 CoreDump_0117-12-26_01-24-10.txt	2017/12/26 1:24	TXT 文件	2,702 KB	core dump file
 nbio_m2m_demo.elf	2017/12/25 16:56	ELF 文件	30,549 KB	elf file
 restore_dump_V1.4.6.2.exe	2018/1/3 17:33	应用程序	1,443 KB	

before drag

Figure 9. Folder structure snapshot for extracting dump file

Step 2, drag core dump log to restore_dump_v1.4.6.2.exe, then a folder with name of current time will be generated. Core dump and elf will be move into YYYY_MM_DD_HH_MM_SS/dumpfile/ as shown in Figure 10:

2018_1_4_15_36_53	2018/1/4 15:36	文件夹	
 dumpfile			
memdump			
minidump			

2018_1_4_15_36_53	2018/1/4 15:36	文件夹	
 restore_dump_V1.4.6.2.exe	2018/1/3 17:33	应用程序	1,443 KB

After drag

Figure 10. Output folder structure snapshot

Step 3, open t32marm.exe: File-->RunBatchFile... choose YYYY_MM_DD_MM_SS/load_dump.cmm. Then it will show the latest call stack as shown in Figure 11:

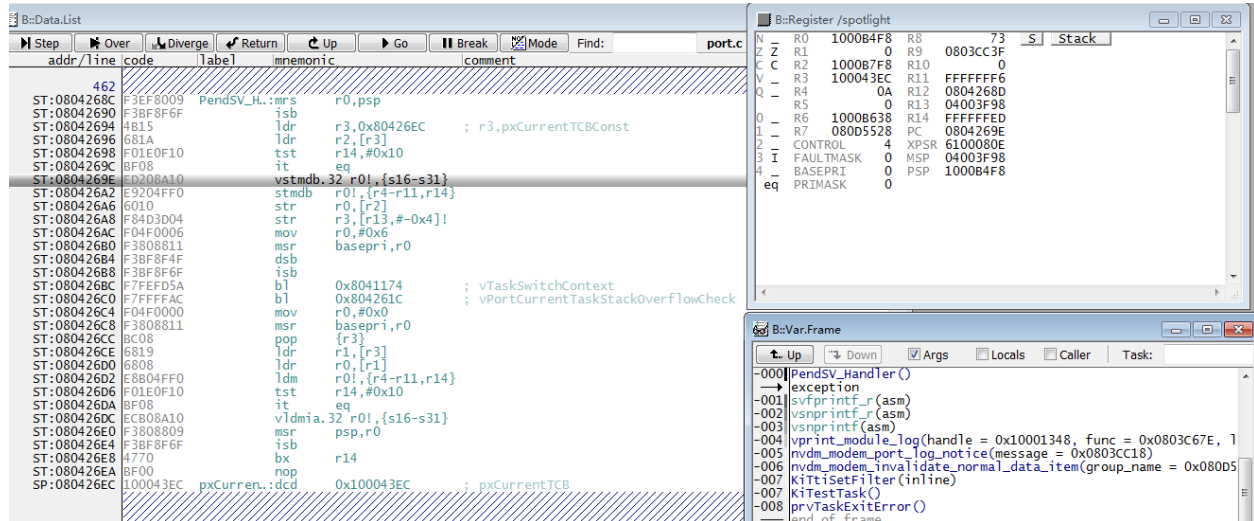


Figure 11. Call stack in Trace32 simulator

Also, we can use View->Symbols->Browse variable to check global variable

And also we can use View->Dump to check memory context.

Some errors:

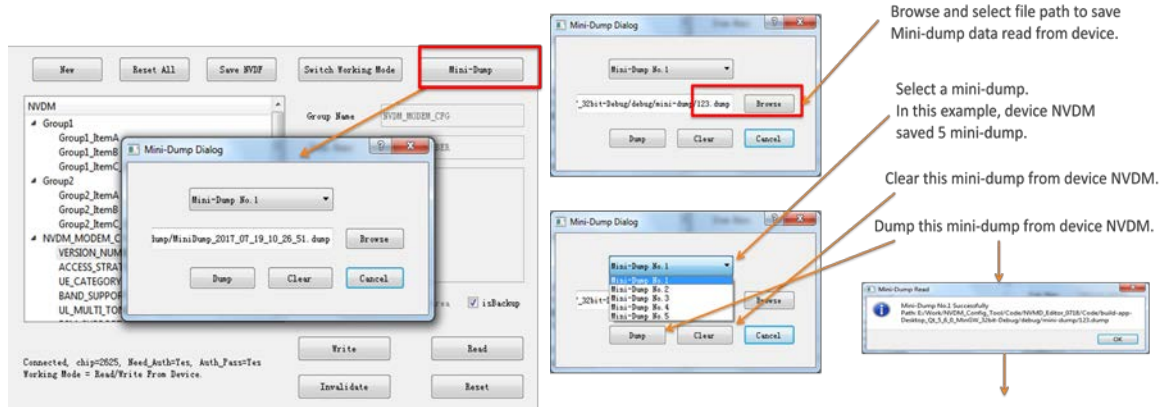
When ELF and core dump log is mismatch, it means the elf and the load occurs assert is mismatch (as shown in Figure 12), please check the timestamp and get the correct elf file.



Figure 12. Error dialog for elf and load mismatching

2.2.4.2. Deal with Mini dump

When core dump is not available, we can only use mini dump for debugging. We can use NVDMEditor tool get mini dump after next boot up. Please check NVDMEditor_Tool_User_Guide.pptx for detail usage.



Mini dump is a file of binary format. We need use restore_dump_v1.4.6.2.exe and trace32 simulation tool to analyze. The process flow is just like core dump.

Step 1, put mini dump log, elf file and restore_dump_v1.4.6.2.exe in the same folder

Step 2, drag mini dump log to restore_dump_v1.4.6.2.exe, then a folder with name of current time will be generated. Mini dump log and elf will be move into YYYY_MM_DD_HH_MM_SS/dumpfile/

Step 3, open t32marm.exe: File-->RunBatchFile... choose YYYY_MM_DD_MM_SS/load_dump.cmm. Then it will show the latest call stack

Note: the global variable via this way can't be trusted because the memory is not record in mini dump.

After step 3, we could get a readable log: YYYY_MM_DD_MM_SS/minidump/log/minidump.log

It includes assert info, modem signal send history, modem task state, modem unprocessed signal and context switch history.

Assert info and context switch is useful for debugging issue. Modem related info is used for MTK Software RD to debug issues.

Assert info is showed as Figure 13:

```

3 assert failed: (NULL), file: RVGNUT, line: 319
4 Current Task Name:VG_CI_TASK_ID
5
6 Assert section
7   Assert type:
8     SYSTEM_ASSERT_TYPE
9
10  Assert message:
11    Assert failed:(NULL),file:RVGNUT,line:319,arg1:0,arg2:0,arg3:0
12    Running thread = VG_CI_TASK_ID

```

Figure 13. Mini dump assertion information

Context switch info includes all tasks and ISRs switch information, it could help to debug issue, the top line is the nearest switch to assert. (Such as the latest switch is to VG_CI_TASK_ID, as shown in Figure 14)

```

223 Context switch history
224 001 Time = 02125(17107.021ms)    VG_CI_TASK_ID/START_PS_NAS_ALIASES
225 002 Time = 02066(17106.962ms)    EMMI_LOW_PRI_TASK_ID/GKI_COMMS_TASK_ID
226 003 Time = 01942(17106.838ms)    EMMI_HIGH_PRI_TASK_ID
227 004 Time = 01910(17106.806ms)    EMMI_LOW_PRI_TASK_ID/GKI_COMMS_TASK_ID
228 005 Time = 01721(17106.617ms)    VG_CI_TASK_ID/START_PS_NAS_ALIASES
229 006 Time = 01667(17106.563ms)    EMMI_LOW_PRI_TASK_ID/GKI_COMMS_TASK_ID
230 007 Time = 01535(17106.431ms)    EMMI_HIGH_PRI_TASK_ID
231 008 Time = 01503(17106.399ms)    EMMI_LOW_PRI_TASK_ID/GKI_COMMS_TASK_ID
232 009 Time = 01171(17106.067ms)    EMMI_HIGH_PRI_TASK_ID
233 010 Time = 01131(17106.027ms)    EMMI_LOW_PRI_TASK_ID/GKI_COMMS_TASK_ID
234 011 Time = 01099(17105.995ms)    EMMI_HIGH_PRI_TASK_ID

```

Figure 14. Context switch information snapshot

2.3. Building the project using the SDK

This section provides detailed guideline on how to set up the SDK build environment with default GCC on Linux OS and on Microsoft Windows using the [MinGW](#) cross-compilation tool and describes two methods on how to build a project.

2.3.1. Installing the SDK build environment on Linux

The default GCC compiler provided in the SDK is required to run on Linux OS. The following description is based on the Ubuntu 14.04 LTS environment.



Note: The LinkIt SDK can be used on any edition of Linux OS. The default GCC compiler provided in the SDK is based on the 32-bit architecture.

Before building the project, verify that you've installed the required toolchain for your build environment, as shown in Table 8.

Table 8. Recommended build environment

Item	Description
OS	Linux OS
make	GNU make 3.81
Compiler	Linaro GCC Toolchain for ARM Embedded Processors 7.2.1

The following command downloads and installs the basic building tools on Ubuntu.

```
sudo apt-get install build-essential
```

Note, a compilation error occurs when building the LinkIt SDK with the default GCC cross compiler on a 64-bit system without installing the package to support the 32-bit executable binary, as shown below.

```
/bin/sh: 4: tools/gcc/gcc-arm-none-eabi/bin/arm-none-eabi-gcc: not found
```



The commands to install the basic build tools and the package for supporting 32-bit binary executable on the Ubuntu 14.04 are shown below.

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6-i386
```

Install the SDK package according to the instructions at `<sdk_root>\readme.txt`. The default installation path of the GCC compiler is `<sdk_root>\tools\gcc`, and the compiler settings are in the `<sdk_root>\.config` configuration file.

Setup the BINPATH in the `.config` file, as shown below.

```
BINPATH = $(SOURCE_DIR)/tools/gcc/gcc-arm-none-eabi/bin
```

2.3.2. Installing the SDK build environment on Microsoft Windows

To build the project on Windows OS, install MinGW cross-compiler and integrate ARM GCC Toolchain for Windows with already installed LinkIt SDK.

- 1) Download mingw-get-setup.exe from [here](#).
- 2) Launch the installer, and click **Install** (see Figure 15).

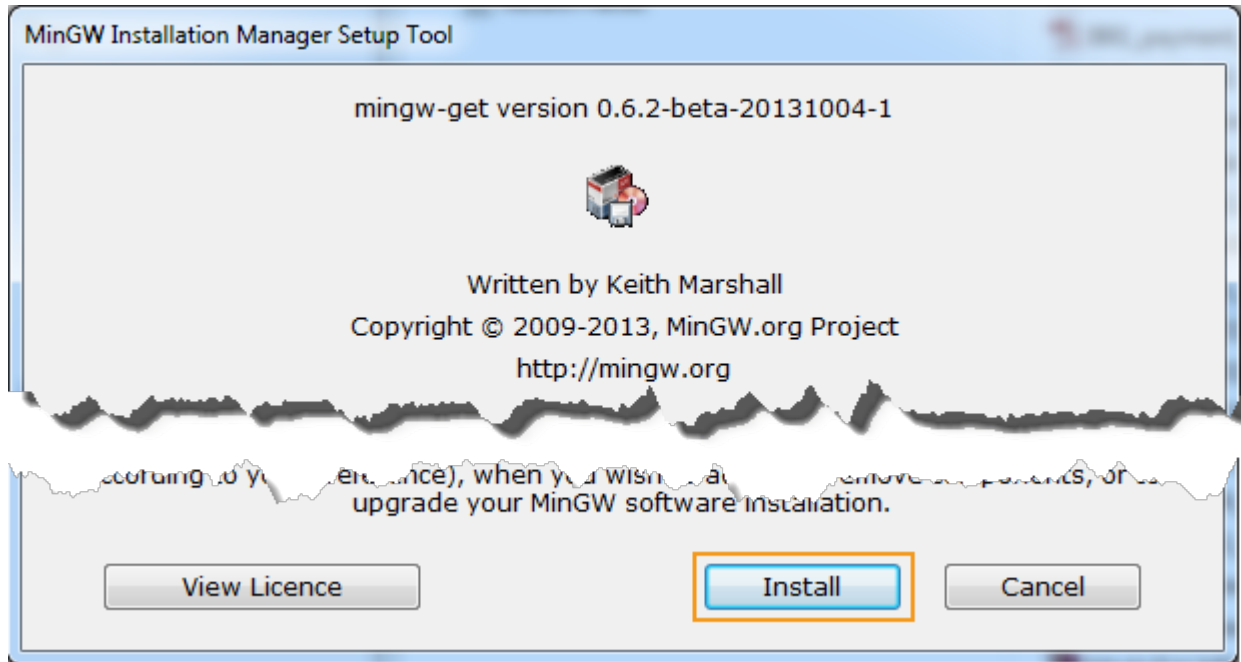


Figure 15. MinGW Installation Manager Setup Tool

- 3) Follow the on screen instructions and keep the default settings, then click **Continue** to download the tool to C:\MinGW installation directory (see Figure 16).

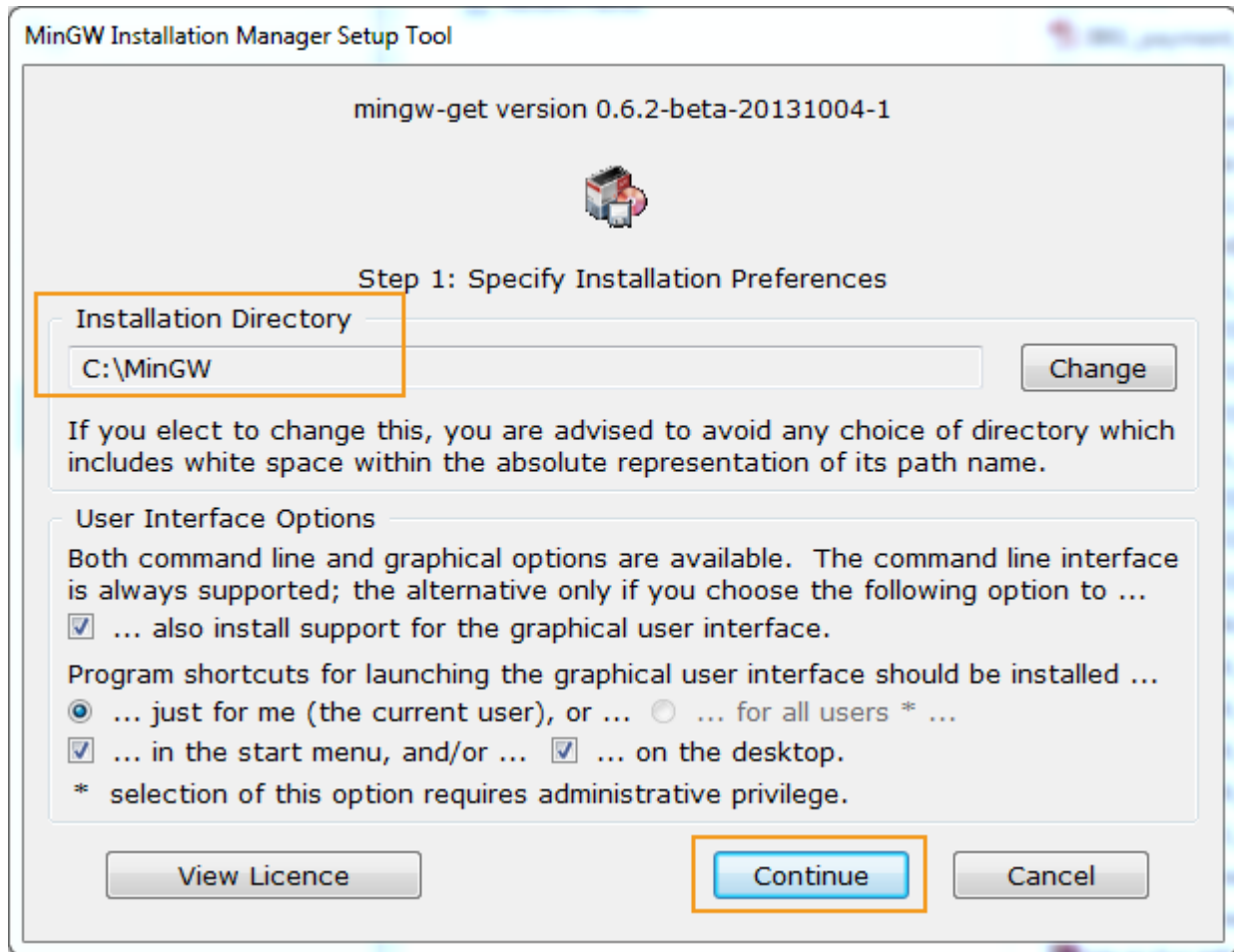


Figure 16. Keep default installation preferences

- 4) Click **Continue** on the **MinGW Installation Manager Setup Tool**, after the download is complete (see Figure 17).

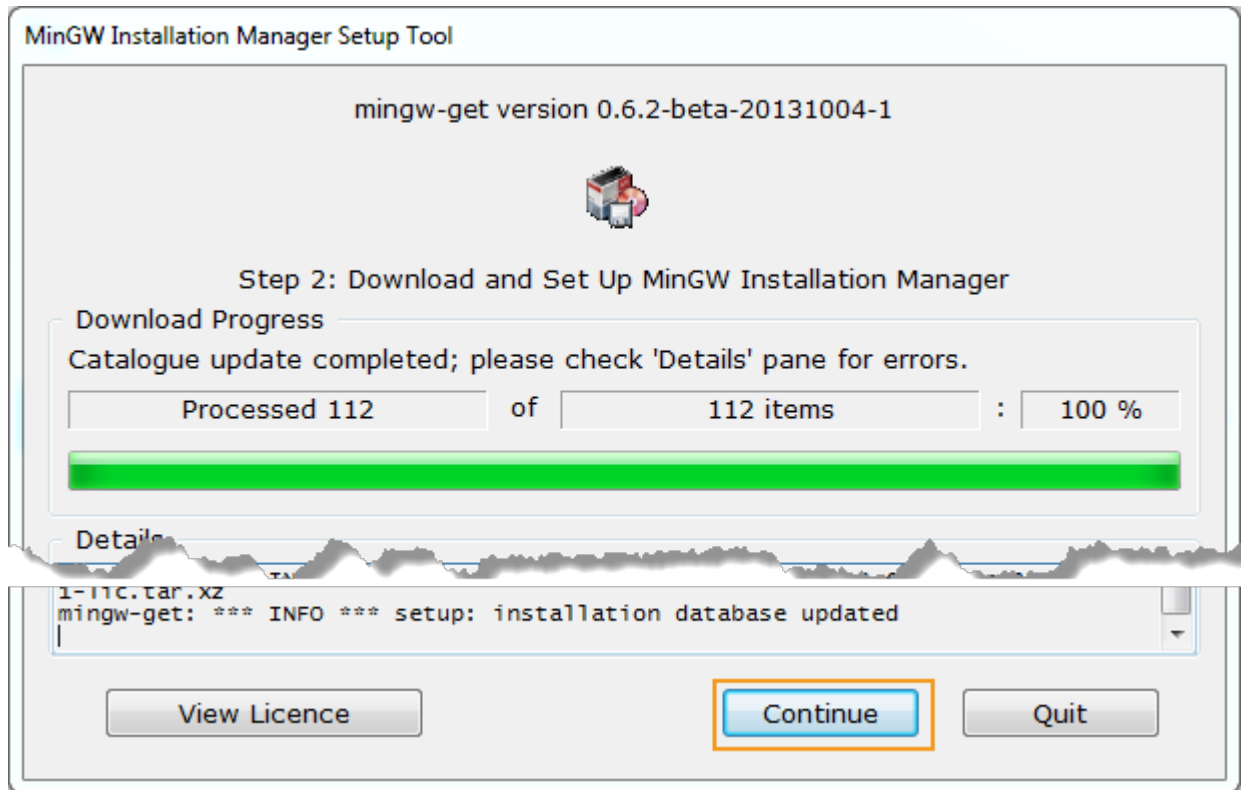


Figure 17. Download and set up MinGW Installation Manager

- 5) Select **msys-base** and **mingw32-base** from **Basic Setup** package list, and right click to bring up the menu options. Click **Mark for Installation** from the menu (see Figure 18).

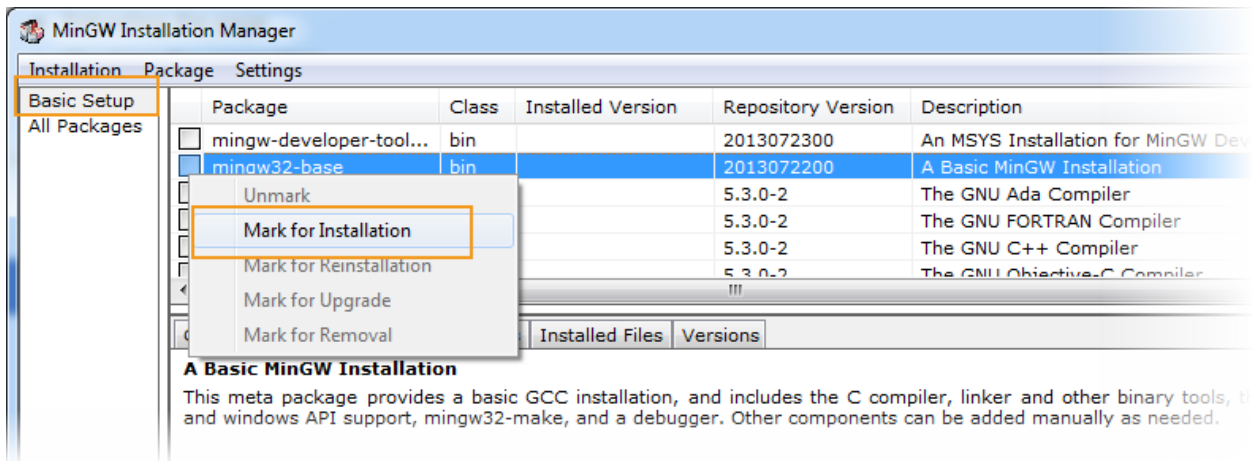


Figure 18. Basic setup on MinGW Installation Manager

- 6) Click **Apply Changes** from the **Installation** menu (see Figure 19).

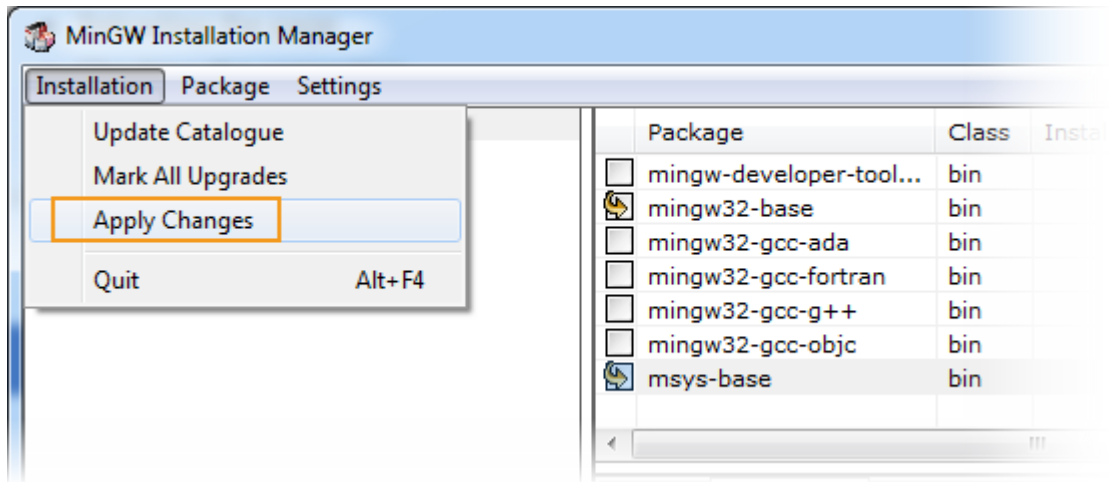


Figure 19. A basic MinGW installation

- 7) Click **Apply** on the pop-up dialog window (see Figure 20).

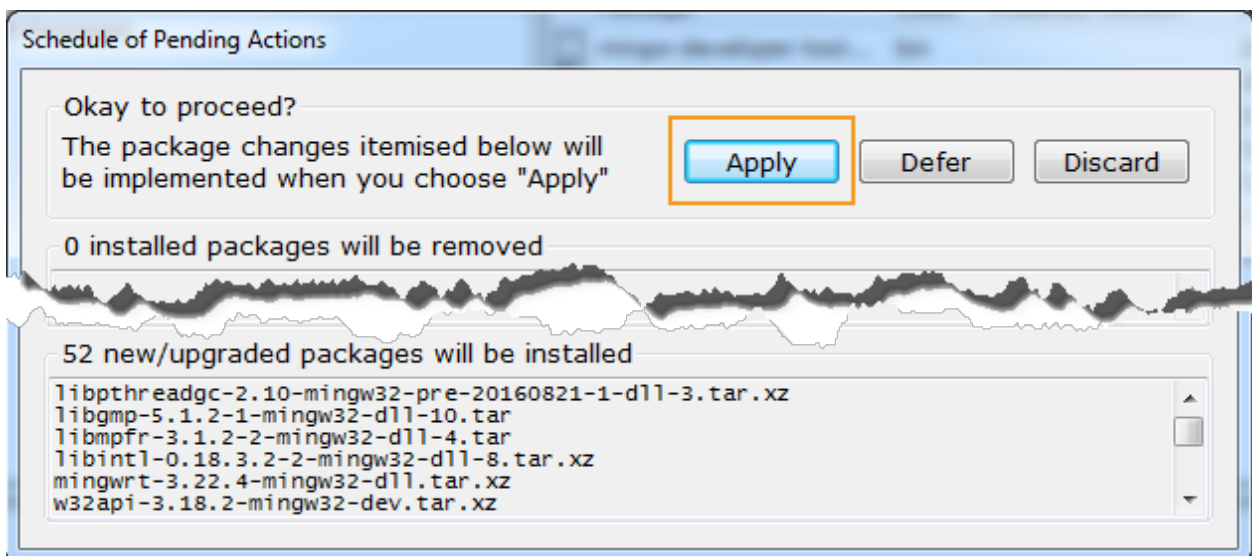


Figure 20. Schedule of pending actions

- 8) Click **Close** to close the dialog window once the operation is complete (see Figure 21).

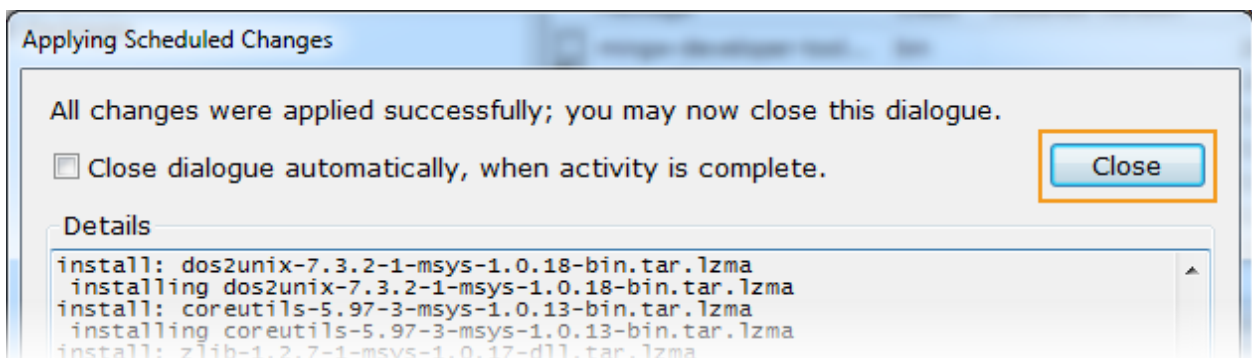


Figure 21. Applying scheduled changes

- 9) Navigate to C:\MinGW\msys\1.0 folder and launch the MinGW terminal by running `msys.bat` to create `home/<user_name>` folder.
- 10) Copy the SDK to MinGW `home\<user_name>` folder, as shown in Figure 22.

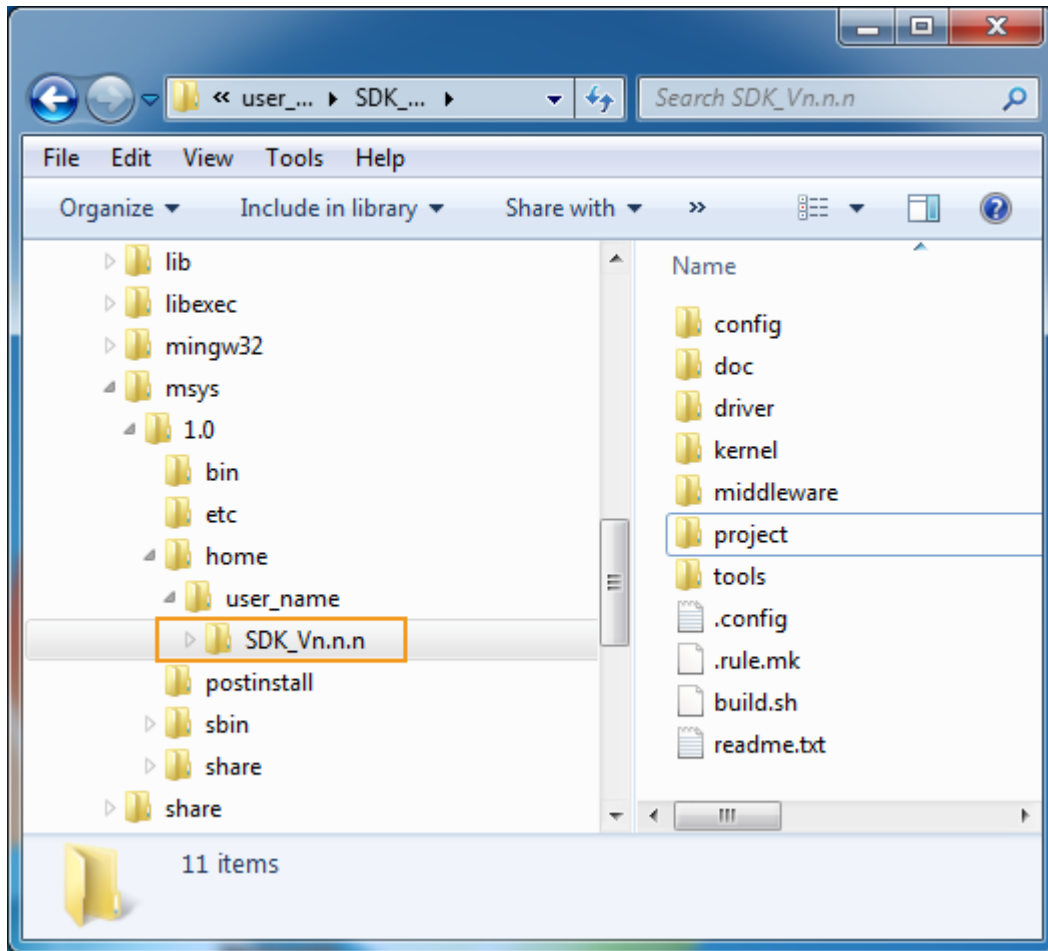


Figure 22. MinGW folder structure

- 11) Download ARM-GCC-win32 from [here](#).
 - a) Create a new folder named `win` under `<sdk_root>\tools\gcc`.
 - b) Unzip the content of `gcc-arm-none-eabi-4_8-2014q3-20140805-win32.zip` to `<sdk_root>\tools\gcc\win\` folder.
 - c) Rename the unzipped `gcc-arm-none-eabi-4_8-2014q3-20140805-win32` folder to `gcc-arm-none-eabi`, as shown in Figure 23.

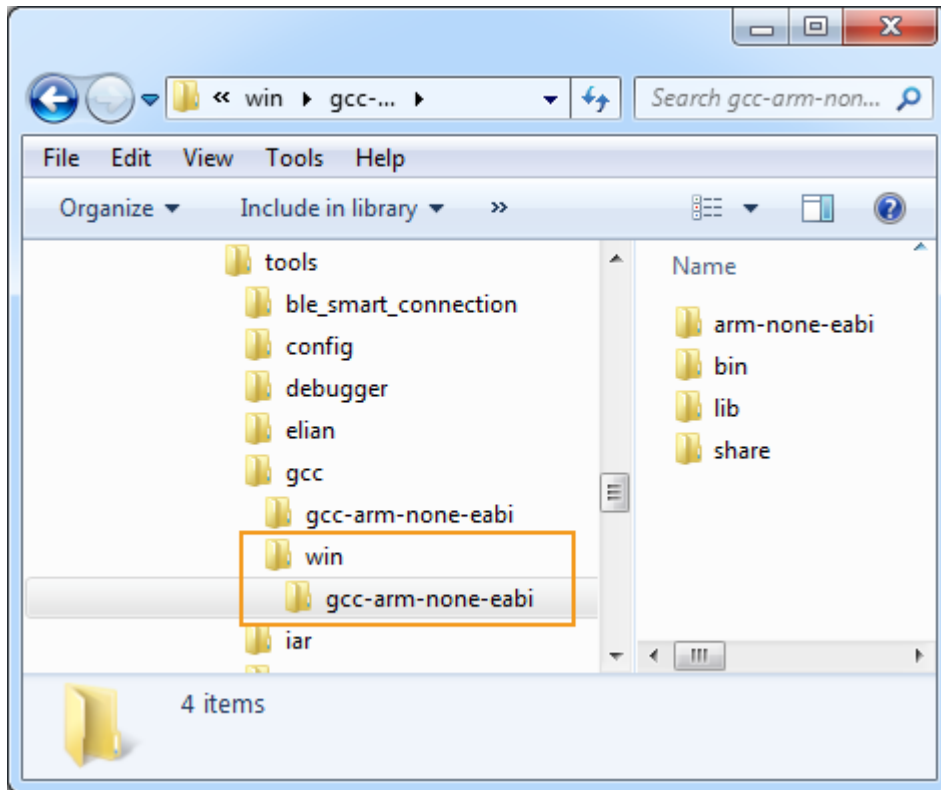


Figure 23. tools\gcc folder structure

- 12) Use `build.sh` to compile the project in MinGW32 console, as described in section 2.3.3, “Methods to build a project”.

2.3.3. Methods to build a project

This section describes two methods on how to build a project:

- Build a project from the SDK root directory.
- Build a project from the project GCC configuration directory.

2.3.3.1. Building the project from the SDK root directory

Build the project using the script at `<sdk_root>\build.sh`. To find out more about the script, navigate to the SDK’s root directory and execute the following command:

```
cd <sdk_root>
./build.sh
```

The outcome is:

```
=====
Build Project
=====
Usage: ./build.sh <board> <project> [bl|clean] <argument>

Example:
./build.sh mt2625_evb nbiot_m2m_demo
./build.sh mt2625_evb nbiot_m2m_demo bl      (build with bootloader)
./build.sh clean                               (clean folder: out)
./build.sh mt2625_evb clean                     (clean folder: out/mt2625_evb)
./build.sh mt2625_evb nbiot_m2m_demo clean    (clean folder:
```

```
out/mt2625_evb/nbiot_m2m_demo)
```

Argument:

```
-f=<feature makefile> or --feature=<feature makefile>
    Replace feature.mk with other makefile. For example,
    the feature_example.mk is under project folder, -
f=feature_example.mk
    will replace feature.mk with feature_example.mk.

-o=<make option> or --option=<make option>
    Assign additional make options. For example,
    to compile module sequentially, use -o=-j1;
    to turn on specific feature in feature makefile, use -
o=<feature_name>=y;
    to assign more than one options, use -o=<option_1> -o=<option_2>.
```

```
=====
List Available Example Projects
=====
Usage: ./build.sh list
```

- List all available boards and projects.

Run the command to show all available boards and projects:

```
./build.sh list
```

The available boards and projects are listed based on the related configuration files under `<sdk_root>/config/project/<board>/<project>` folder. The console output is shown below.

```
=====
Available Build Projects:
=====
mt2625_evb
  bootloader
  fota_da
  nbiot_m2m_demo
  nbiot_meter_demo
  sensor_subsys_accelerometer
  freertos_create_thread
...
mt2625_bc26
  bootloader
  nbiot_sdk
mt2625_g1000
  bootloader
  nbiot_sdk
```

- Build the project.

To build a specific project, simply run the following command.

```
./build.sh <board> <project>
```

The output files will be placed under `<sdk_root>\out\<board>\<project>` folder.

For example, to build a project on the MT2625 platform, run the following build command.

```
./build.sh mt2625_evb nbiot_m2m_demo
```

The standard output in the terminal window:

```
$ ./build.sh mt2625_evb nbiot_m2m_demo
UE build board:mt2625_evb
```

```
UE build project:nbiot_m2m_demo
platform=linux
FEATURE=feature.mk
make: Entering directory
`/usr/home/user_name/SDK_vn.n.n/project/mt2625_evb/apps/
nbio_t_m2m_demo/GCC'
...
```

The output files will be placed under `<sdk_root>\out\mt2625_evb\nbio_t_m2m_demo\` folder.

- Build the project with the "b1" option.

By default, the pre-built bootloader image file is copied to the `<sdk_root>\out\<board>\<project>\` folder after the project is built. The main purpose for the bootloader image is to download the Flash Tool.

Apply the "b1" option to rebuild the bootloader and use the generated bootloader image file instead of the pre-built one, as shown below.

```
./build.sh <board> <project> b1
```

To build the project on the MT2625 platform:

```
./build.sh mt2625_evb nbio_t_m2m_demo b1
```

The output image file of the project and the bootloader will be placed under `<sdk_root>\out\mt2625_evb\nbio_t_m2m_demo` folder.

- Clean the out folder

The build script `<sdk_root>\build.sh` provides options to remove the generated output files as follows.

- 1) Clean the `<sdk_root>\out` folder.

```
./build.sh clean
```

- 2) Clean the `<sdk_root>\out\<board>` folder

```
./build.sh <board> clean
```

- 3) Clean the `<sdk_root>\out\<board>\<project>` folder.

```
./build.sh <board> <project> clean
```

2.3.3.2. Building the project from the project GCC configuration directory

For this method, please follow the steps below:

- 1) Change the current directory to project source directory where the SDK is located.
- 2) There are makefiles provided for the project build configuration. For example, the `nbio_t_m2m_demo` is built by the project makefile under `<sdk_root>\project\mt2625_evb\apps\nbio_t_m2m_demo\GCC`.
 - i) Navigate to the example project's location.

```
cd <sdk_root>/project/mt2625_evb/apps/nbio_t_m2m_demo/GCC
```

- ii) Run the make command.

```
make
```

The output folder is defined under variable `BUILD_DIR` in the Makefile located at `<sdk_root>\project\mt2625_evb\apps\nbio_t_m2m_demo\GCC`:

```
BUILD_DIR = $(PWD)/Build
PROJ_NAME = mt2625_nbio_t_m2m_demo
```

A project image `mt2625_nbiot_m2m_demo.bin` is generated under
`<sdk_root>\project\mt2625_evb\apps\nbiot_m2m_demo\GCC\Build`.

2.4. Create your own project

This section provides details on how to use an existing project and create your own project named `my_project` on MT2625 platform using `nbiot_m2m_demo` project as a reference.

2.4.1. Using an existing project

Apply an existing project as a reference design for your own project development.

- 1) Copy the folder `<sdk_root>\project\mt2625_evb\apps\nbiot_m2m_demo` to a new directory `<sdk_root>\project\mt2625_evb\apps\my_project`.

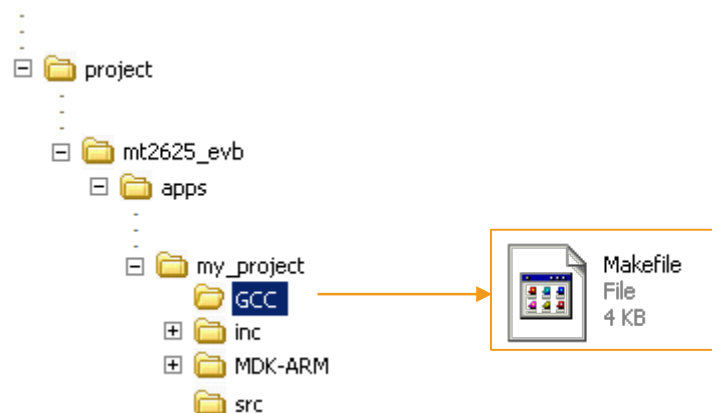


Figure 24. Modify the Makefile under the GCC folder of my_project

- 2) Modify the following settings defined in the `<sdk_root>\project\mt2625_evb\apps\my_project\GCC\Makefile` under the GCC folder (see Figure 24) `my_project`, as shown below:
 - `SOURCE_DIR`: the path to `<sdk_root>`.
 - `PROJ_NAME`: project name.
 - `APP_PATH`: project path.

```
...
SOURCE_DIR = ../../../../..
BINPATH    = ~/gcc-arm-none-eabi/bin
PWD        = $(shell pwd)
DATIME     = $(shell date --iso=seconds)
V          ?= 0
...
# Project name
PROJ_NAME  = mt2625_my_project
PROJ_PATH  = $(PWD)
OUTPATH    = $(PWD)/Build

APP_PATH   = project/mt2625_evb/apps/my_project
APP_PATH_SRC = $(APP_PATH)/src
...
```

2.4.2. Removing a module

The copied project has modules that could be removed in order to have a clean start for your project development. After the previous steps, a project with the same features has been created. It can be built to generate image file as the original project.

To remove a module:

- 1) Open the project Makefile from
`<sdk_root>/project/mt2625_evb/apps/my_project/GCC/Makefile.`
- 2) Locate the module include list of the project and remove any unwanted module by removing or commenting out the corresponding include statement.

```
...
#####
####
#
# SDK source files
#
#####
####
#include cJSON
include $(SOURCE_DIR)/middleware/third_party/cjson/module.mk

#include xml
include $(SOURCE_DIR)/middleware/third_party/xml/module.mk
...
```

2.4.3. Add the source and header files

User defined project source and header files should be placed under the `src` and the `inc` folder respectively, as shown in Figure 25.

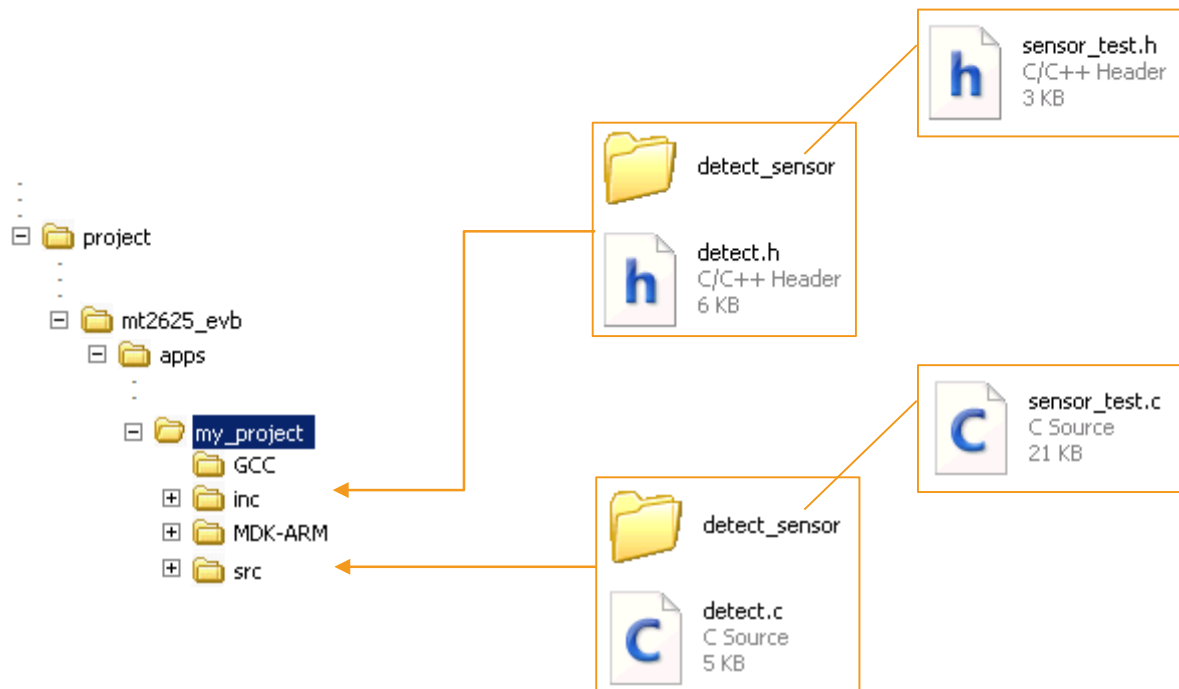


Figure 25. Project source and header files under the project folder

To compile the added source code, simply add the .c source files to variable "C_FILES" and the header search path to variable "CFLAGS" in the project Makefile, as shown below. The corresponding variables to support compiling the source files (.cpp) of the module are CXX_FILES and CXXFLAGS).

<sdk_root>/project/mt2625_evb/apps/my_project/GCC/Makefile

```
...
C_FILES += $(APP_PATH_SRC)/main.c \
            $(APP_PATH)/GCC/syscalls.c \
            $(APP_PATH_SRC)/sys_init.c \
            $(APP_PATH_SRC)/md_init.c

CXX_FILES +=

...
CFLAGS += -I$(SOURCE_DIR)/kernel/service/inc
CFLAGS += -I$(SOURCE_DIR)/$(APP_PATH)/inc

CXXFLAGS +=

...
```

3. Appendix A: Acronyms and Abbreviations

The acronyms and abbreviations used in this get started guide are listed in Table 9.

Table 9. Acronyms and Abbreviations

Abbreviation/Term	Expansion/Definition
ATCI	AT command interface (ATCI), usually AT commands are used to control modems to do their specified functions, they are also used to production line test.
ANSI	The American National Standards Institute
DBCS	A double-byte character set (DBCS) is a character encoding in which all characters (including control characters) are encoded in two bytes.
FOTA	Firmware over the air, a way to update firmware using wireless communication
GNSS	Global Navigation Satellite System
NEC	A kind of Infrared Transmission Protocol, The NEC IR transmission protocol uses pulse distance encoding of the message bits.
NVDM	Non-Volatile Data Management
OEM	Original Equipment Manufacturer (OEM) is a company that makes a part or subsystem that is used in another company's end product.
RC5	A type of infrared transmission protocol. The RC-5 protocol was developed by Philips in the late 1980s as a semi-proprietary consumer IR (infrared) remote control communication protocol for consumer electronics.
RC6	A type of infrared transmission protocol. RC-6 is, as may be expected, the successor of the RC-5 protocol. Like RC-5 the new RC-6 protocol was also defined by Philips. It is a very versatile and well-defined protocol.
TLS	Transport Layer Security (TLS) is cryptographic protocols that provide communications security over a computer network.

4. Appendix B: Disabling Automatic Driver Installation on Windows OS

The automatic download and installation of device drivers can prevent proper installation of the USB COM port driver on Windows 7, 8 and 10 machines. If you've already disabled the automatic installation of device drivers, you can skip this step, otherwise:

- 1) Open **Control Panel**, search for and open **Change device installation settings**.
 - 2) In **Device Installation Settings** select **No, let me choose what to do**, then click **Never install driver software from Windows Update**, as shown below.
- On Windows 7:

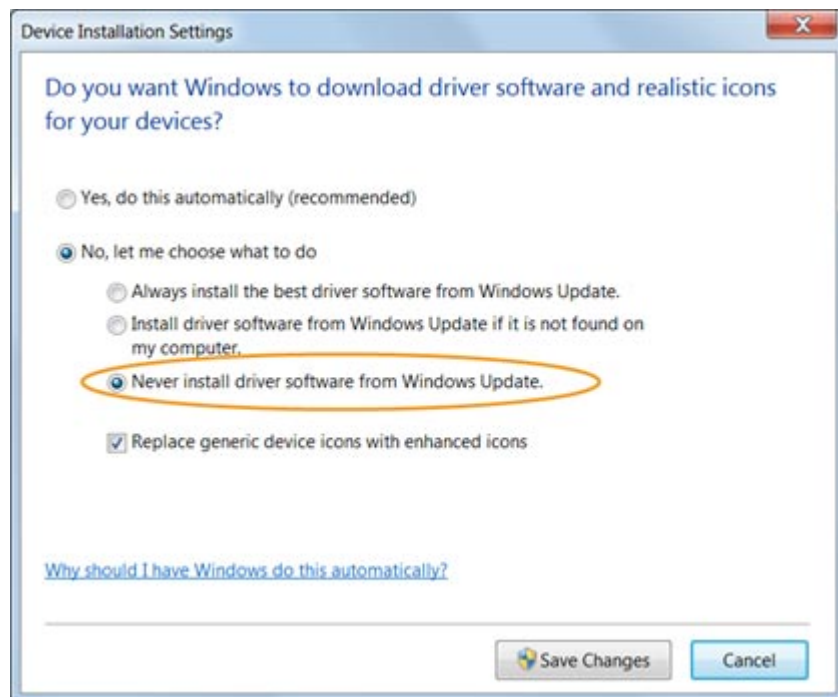


Figure 26. Disabling automatic driver updates on Windows 7 OS

- On Windows 8 and 10:

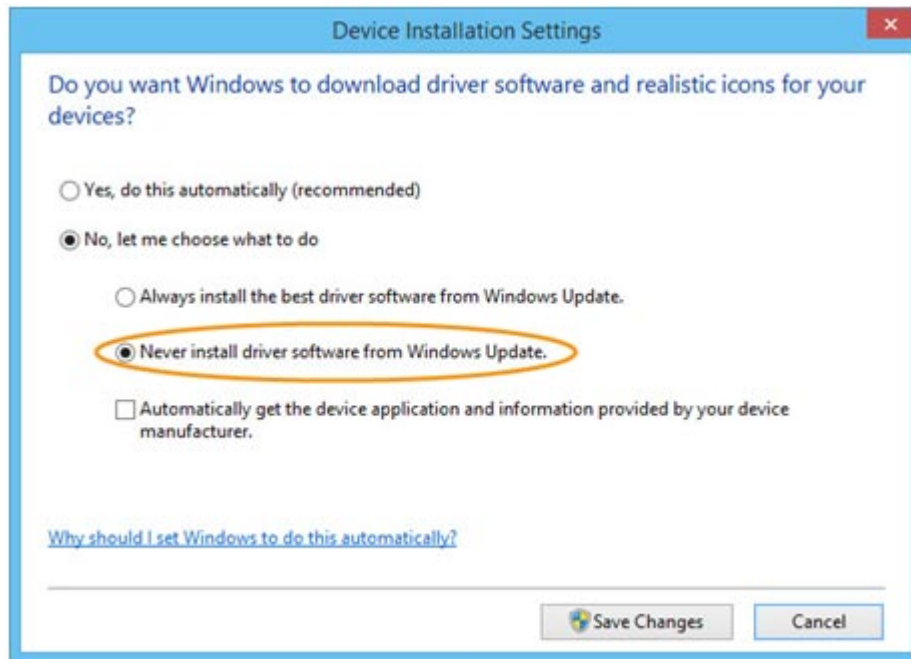


Figure 27. Disabling automatic driver updates on Windows 8 and 10

Also make sure to uncheck **Automatically get the device application and information provided by your device manufacturer.**

- 3) Click **Save** Changes

You can now install the dedicated USB Driver for your device.