LoRaWAN - GitHub Code

Quick Start Guide – Version 0.1

Document ID: 4000/40140/0143

IMST GmbH

Carl-Friedrich-Gauß-Str. 2-4

47475 KAMP-LINTFORT

GERMANY

Document Information

File name	LoRaWAN_GitHubCode_QuickStartGuide.docx
Created	2018-10-11
Total pages	18

Revision History

Version	Note
0.1	Created, Initial Version

Aim of this Document

This document contains an introduction to the LoRaWAN stack available in GitHub for the WiMOD LR radio module family (e.g. iM880B, iM881A and iM980A).

Table of Contents

1.Introduction

1.1 Overview

This document contains a guidance on the setup of the LoRaWAN™ [1] open source code available in GitHub running, including the installation and handling of a free GNU toolchain.

The next chapter provides the details to install and configure the required tools to develop an own LoRaWAN application under Windows 10 following the instructions in the *development-enviroment.md* file available in the GitHub documentation, where Visual Studio Code is chosen as source code editor and MinGW as Make utility needed to generate the executable files.

Furthermore, an example application for the WiMOD LR radio modules is shown.

[1] LoRa is a Trademark of Semtech Corporation / LoRaWAN is a Trademark of LoRa Alliance

2. Setup

Before starting any interaction with the LoRaWAN example code, the following components are required:

- **IMST Started Kit including Radio Module:** e.g. iM880B-L, iM881A-XL, iM980A
- **Programmer:** an ST Link V2 programmer with an adapter PCB (available from IMST) is recommended for downloading the hex file into the microcontroller or for debugging purposes.

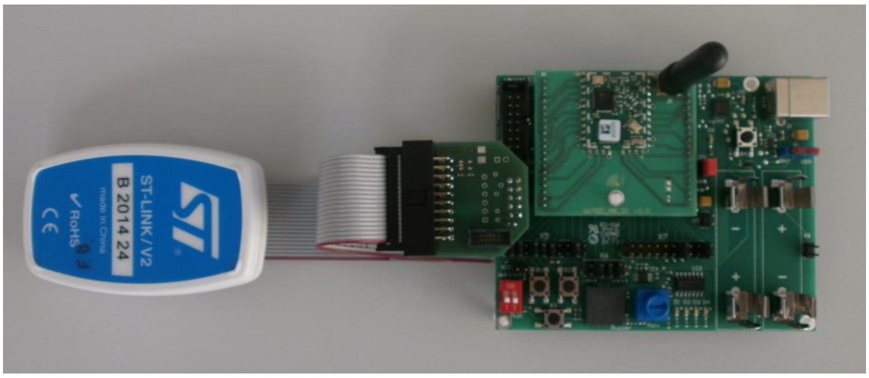

Fig. 2-1: ST Link V2 programmer

- **Integrated Development Environment (IDE):** for this document *Visual Studio Code* is used.

  A free version (e.g. *VSCode-win32-x64-1.28.0.zip*) is available under:

  https://code.visualstudio.com/Download
- **GitHub Source Code:** available under:

  https://github.com/Lora-net/LoRaMac-node

  For this, the *Git* application is required. E.g. *Git-2.19.1-64-bit.exe* under https://git-scm.com/

  The *TortoiseGit* interface (https://tortoisegit.org/) could be used to gain access easily to the latest source code.

And the following prerequisites should be met:

- **CMake:** version equal to or greater than 3.6 is required.

  This (e.g. cmake-3.12.3-win64-x64.msi) is available under:

  https://cmake.org/download/

  Note: Please use the latest full release.
- **GNU Arm Embedded Toolchain:** available under:

  https://developer.arm.com/open-source/gnu-toolchain/gnu-rm

  E.g. *gcc-arm-none-eabi-7-2018-q2-update-win32-sha1.exe*

- **Make utility:** for this document *MinGW* is used. This is available under: http://mingw.org

  E.g. here *mingw-get-setup*.exe is available under *downloads*.

  Thanks to this application, the required packages could be installed. These should be selected and then click on ***Apply Changes*** under ***Installation***.

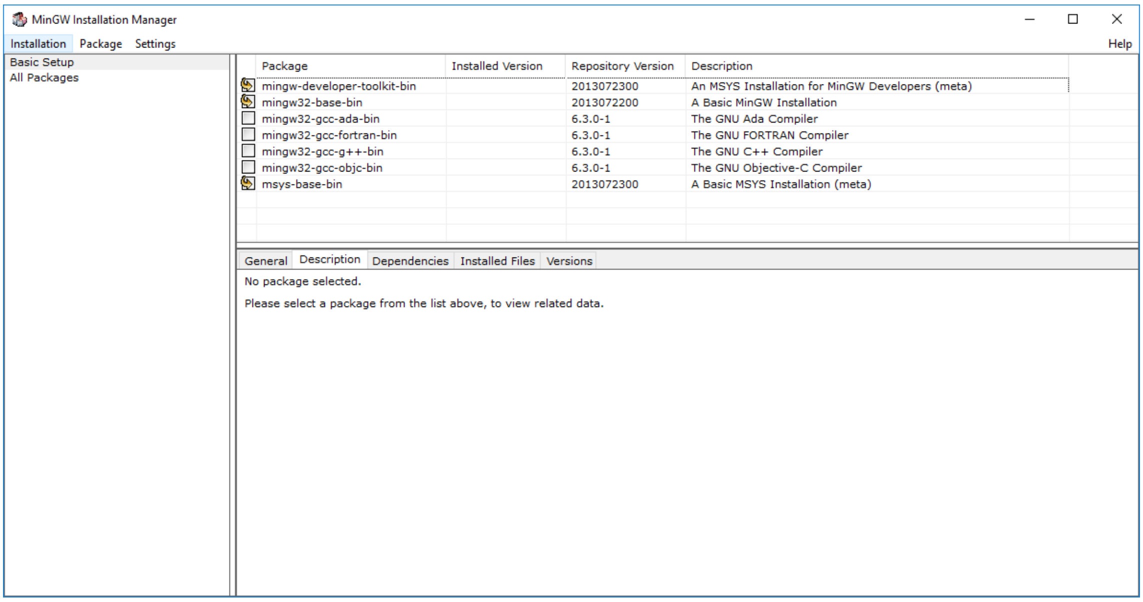

Fig. 2-2: MinGW – Example of installation

- **OpenOCD:** unofficial binary packages are available under http://openocd.org/getting-openocd/ for download.

  E.g. *gnu-mcu-eclipse-openocd-0.10.0-8-20180512-1921-win64.zip* under https://github.com/gnu-mcu-eclipse/openocd/releases. Unzip and copy the content of "*gnu-mcu-eclipse-openocd-0.10.0-8-20180512-1921-win64\GNU MCU Eclipse\OpenOCD\0.10.0-8-20180512-1921*" into "*C:/openocd*".

- Ensure that the following paths are added to the system *Path* variable:
  - CMake (e.g. *"C:\Program Files\CMake\bin"*)
  - MinGW (e.g. *"C:\msys64\mingw64\bin"*)
  - GNU Tools ARM Embedded (e.g. *"C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update\bin"*)

  For example, on Windows 10 followings steps could be followed:
  - Click on ***Settings*** (under the *Start Menu*) and then on ***System***:

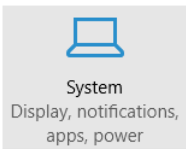

Fig. 2-3: Windows 10 - System

- Choose ***About*** and then ***System info***:

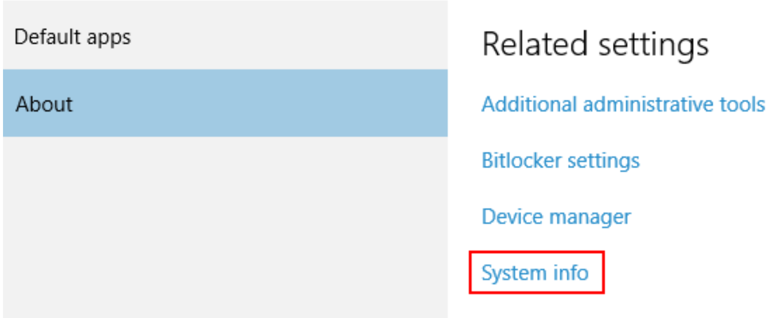

Fig. 2-4: Windows 10 – System info

Note: The same window could be open with the ***System*** option on the ***Control Panel***.

- Click on ***Advanced System Settings*** and select ***Environment Variables***:

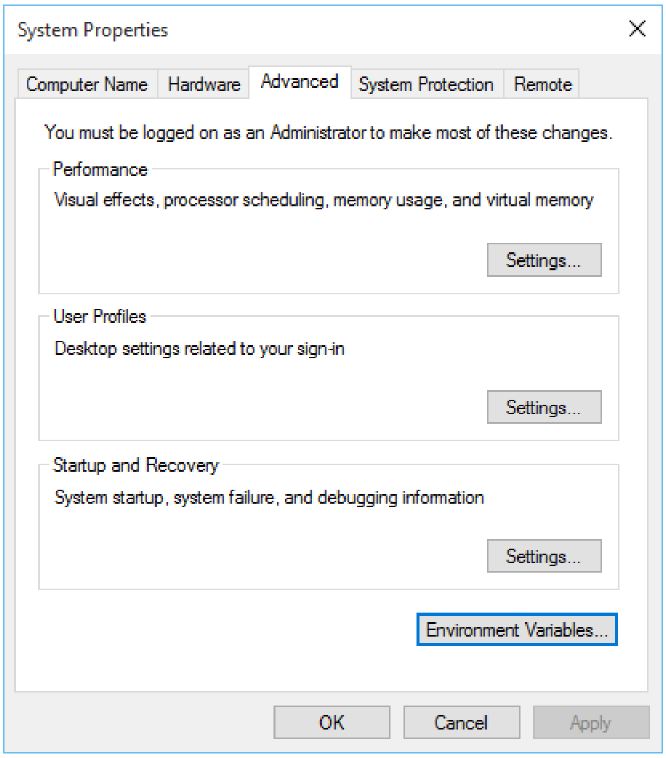

Fig. 2-5: Windows 10 – Environment Variables

- Add the required paths to the Path variable

Finally, the Visual Studio Code should be configured correctly:

- Install the required extensions. For this, open Visual Studio Code and search for the following extensions after clicking Ctrl+Shift+X:
    - C/C++

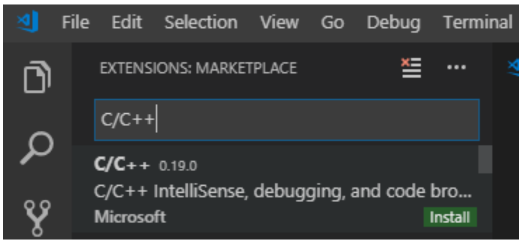

Fig. 2-6: Visual Studio Code – C/C++ extension

    - CMake and CMake Tools

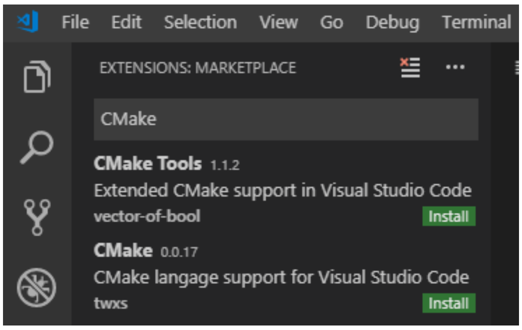

Fig. 2-7: Visual Studio Code – CMake and CMake Tools extensions

    - Native Debug

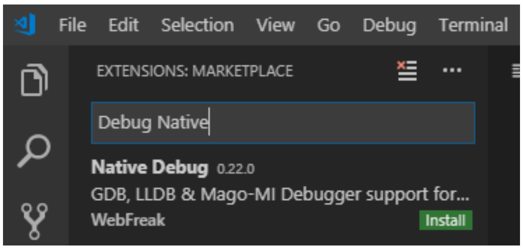

Fig. 2-8: Visual Studio Code – Native Debug extension

- Add CMake configuration to the user settings. For this, select ***File>Preferences>Settings*** and open *settings.json*.

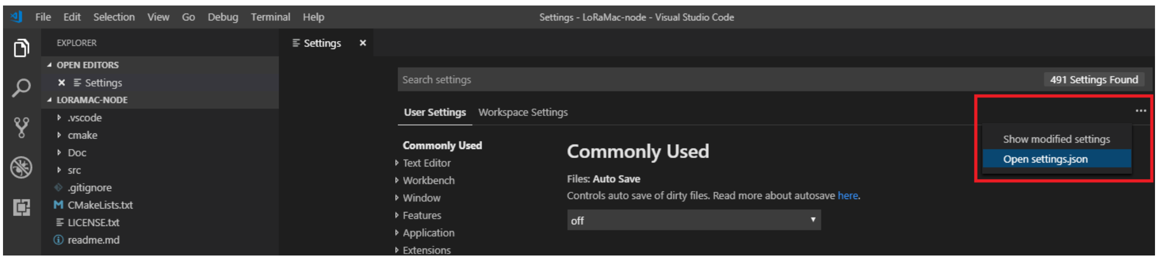

Fig. 2-9: Visual Studio Code – User Settings

Add the configuration needed for CMake as shown in the next picture:

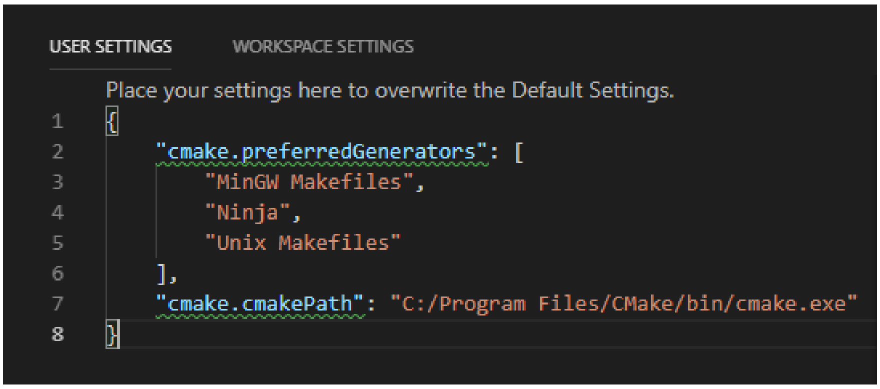

Fig. 2-10: Visual Studio Code – User Settings (CMake)

- Open the directory with the cloned repository selecting ***File>Open Folder*** and select e.g. *LoRaMac-node*.

  Here, the *CMake Tools* extension will automatically generate a **.cmaketools.json** file based on the *CMakeLists*.

- Modify the workspace settings as required (*settings.json* file under the *.vscode* folder). E.g:
  - "TOOLCHAIN_PREFIX":"C:/Program Files (x86)/GNU Tools Arm Embedded/7 2018-q2-update"
  - "OPENOCD_BIN":"C:/openocd/bin/openocd.exe"
  - "CMAKE_TOOLCHAIN_FILE":"cmake/toolchain-arm-none-eabi.cmake"

3. Demo Application

This chapter explains the steps needed to build the ClassA end-device example application using an iM880B radio module (available under *LoRaMac-node\src\apps\LoRaMac\classA\SKiM880B\main.c*).

Following steps are needed:

- Modify the workspace settings as required for the application (*settings.json* file under the *.vscode* folder). E.g:
    - APPLICATION: LoRaMac
    - CLASS: classA
    - ACTIVE_REGION: LORAMAC_REGION_EU868
    - BOARD: SKiM880B
    - REGION_EU868: ON

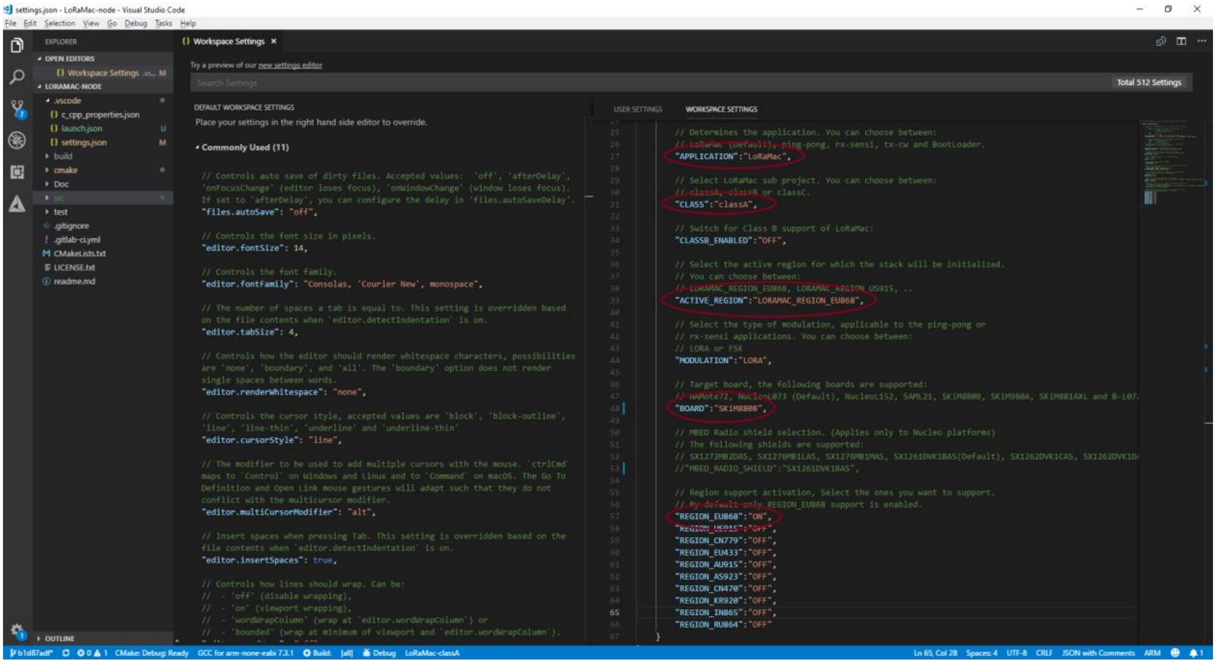

Fig. 3-1: Visual Studio Code – Workspace Settings

- Click on the blue status bar of *CMake Tools* to choose a build type (*Debug* or *Release*).

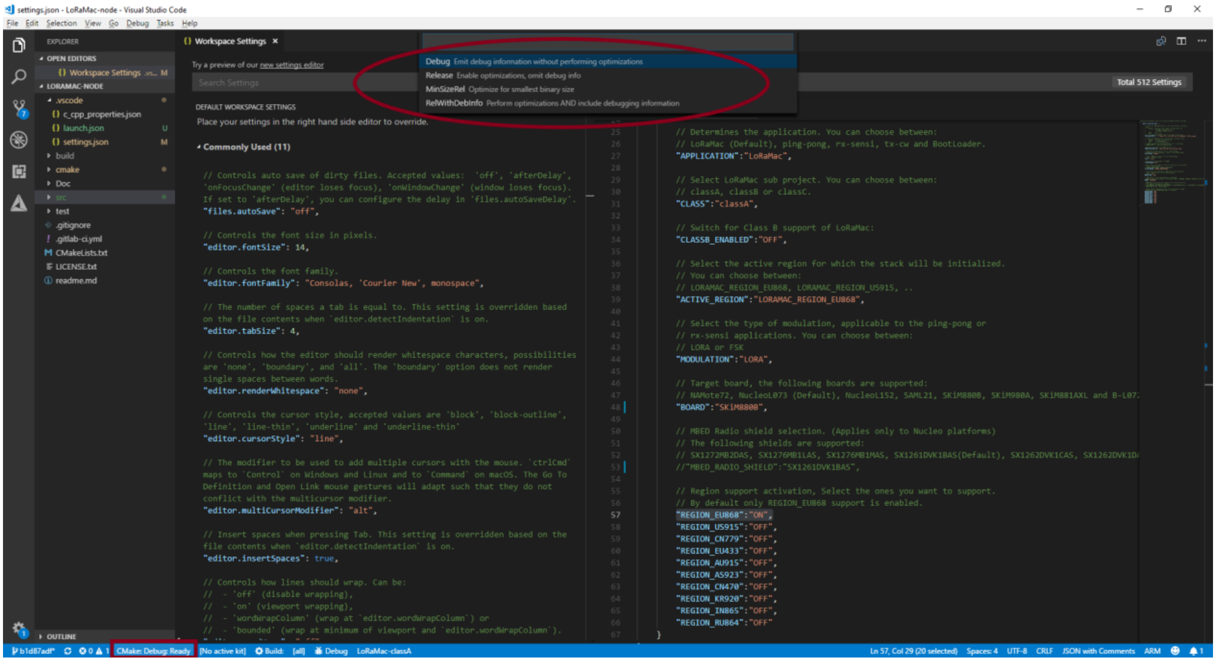

Fig. 3-2: Visual Studio Code – CMake Tools Settings

- Select the GCC compiler (e.g. GCC for arm-none-eabi 7.3.1) clicking on *[No active Kit]* on the blue status bar.

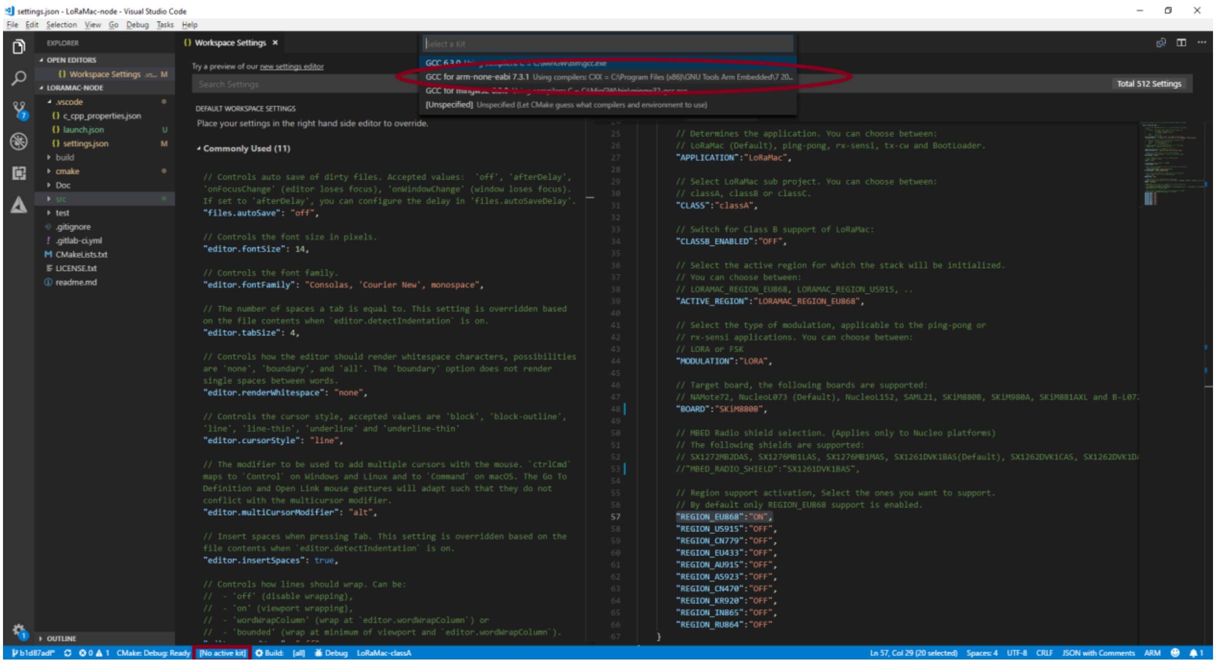

Fig. 3-3: Visual Studio Code – GCC Compiler Settings

- A *Build* option will be now available on the blue status bar. This will build the target.

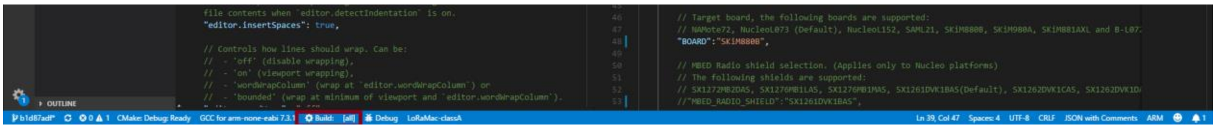

Fig. 3-4: Visual Studio Code – Build

The binary file (e.g. LoRaMac-classA.hex) will be available under the build directory (e.g. *build\src\apps\LoRaMac*) and could be programmed into the iM880B radio module, e.g. using an ST Link V2 programmer.

The CMake build system will automatically generate a **launch.json** file which setups the debugging process for the given board.

After this, the F5 key could be used to start a debug session. This will automatically start the GDB and OpenOCD processes.

Note: Please follow the next steps for a new build after any modification of the application/configuration:

- Change CMake options: Open the Command palette (Crtl+Shift+P) and type ***CMake: Edit the CMake Cache***

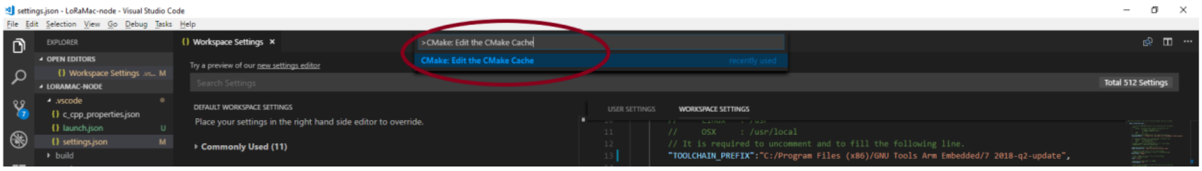

Fig. 3-5: Visual Studio Code – CMake Settings

- Execute a clean rebuild: Open the Command palette (Crtl+Shift+P) and type ***CMake: Clean rebuild***

Moreover, within this application the radio module provides details concerning its configuration and radio messages exchange via the USART serial communication (configured with 912600 baud rate). For this, a terminal emulation program for the serial interface (e.g. TeraTerm) could be used:

- Create a new connection to the serial port.

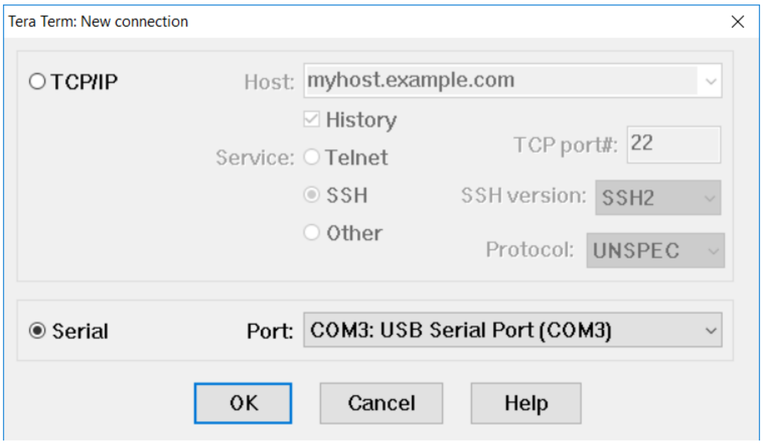

Fig. 3-6: Tera Term – New Connection

- Configure the baud rate under ***Setup>Serial port…***

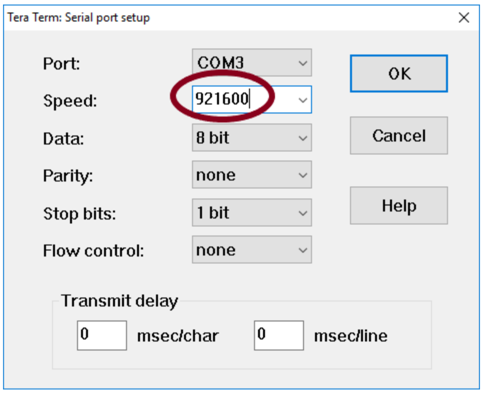

Fig. 3-7: Tera Term – Serial port setup

- Then the data sent by the radio module via the USART interface is shown as follows:

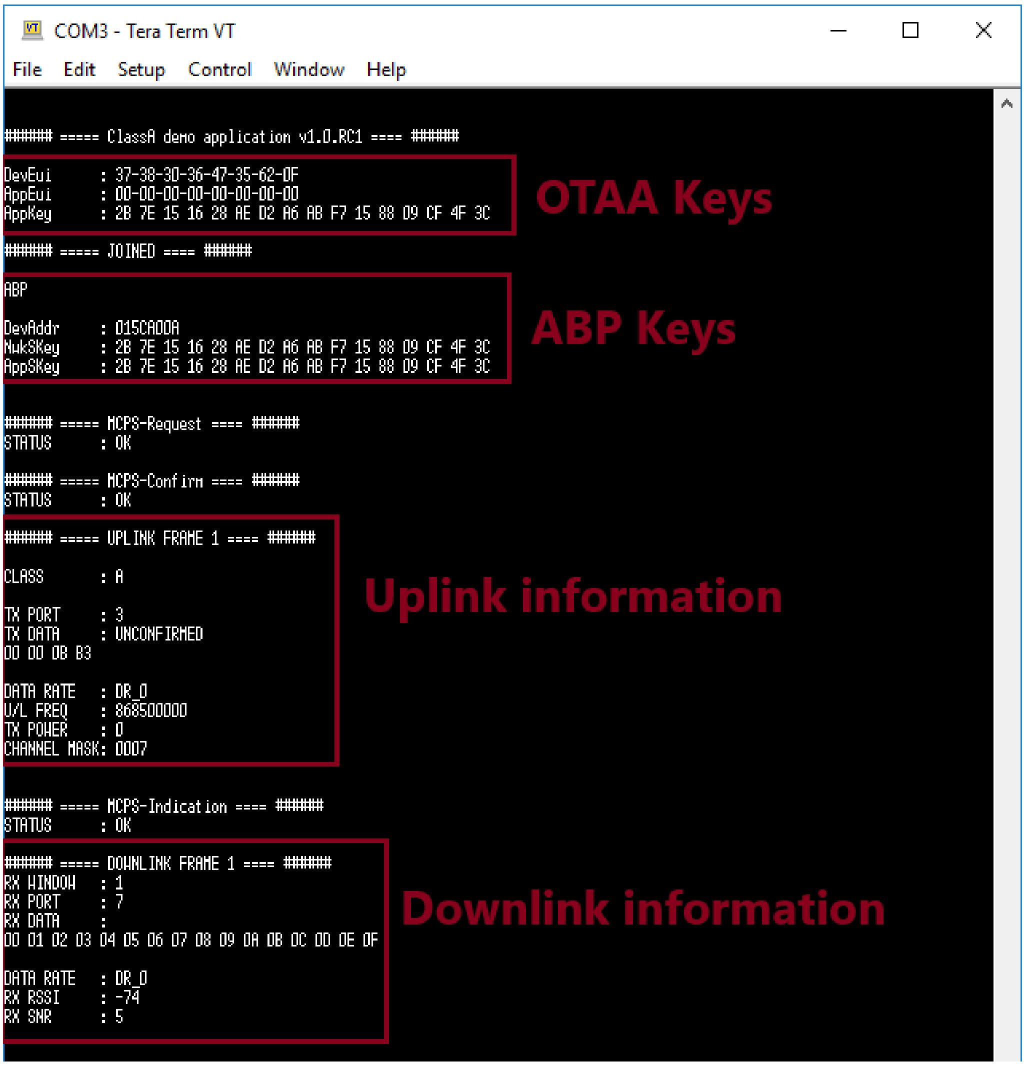

Fig. 3-8: Tera Term – Serial port communication

This example uses ABP activation. In *Commissioning.h* the activation mode and parameters could be modified.

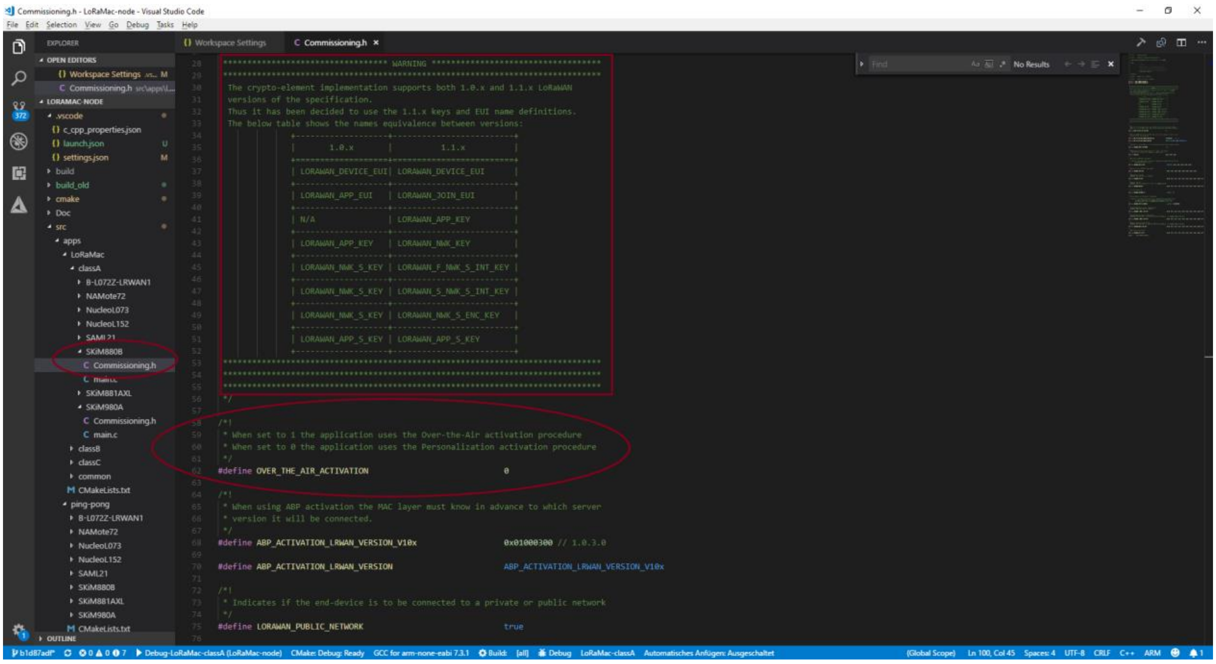

Fig. 3-9: Visual Studio Code – Commissioning

4. Appendix

4.1 List of Abbreviations

ABP	Activation By Personalization
API	Application Programming Interface
IDE	Integrated Development Environment
LoRaWAN	Long Range Wide Area Network
LR	Long Range
OTAA	Over-the-air Activation
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
WiMOD	Wireless Module

4.2 List of Figures

4.3 List of References

[1] LoRa WAN Specification.pdf.

5. Important Notice

5.1 Disclaimer

IMST GmbH points out that all information in this document is given on an “as is” basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication. IMST GmbH reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete. All products are sold and delivered subject to “General Terms and Conditions” of IMST GmbH, supplied at the time of order acknowledgment.

IMST GmbH assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer’s duty to bear responsibility for compliance of systems or units in which products from IMST GmbH are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications. The products are not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the products within such applications do so at their own risk.

Any reproduction of information in datasheets of IMST GmbH is permissible only if reproduction is without alteration and is accompanied by all given associated warranties, conditions, limitations, and notices. Any resale of IMST GmbH products or services with statements different from or beyond the parameters stated by IMST GmbH for that product/solution or service is not allowed and voids all express and any implied warranties. The limitations on liability in favor of IMST GmbH shall also affect its employees, executive personnel and bodies in the same way. IMST GmbH is not responsible or liable for any such wrong statements.

5.2 Contact Information

IMST GmbH

Carl-Friedrich-Gauss-Str. 2-4
47475 Kamp-Lintfort
Germany

T +49 2842 981 0
F +49 2842 981 299
E wimod@imst.de
I www.wireless-solutions.de