HT MICRON SEMICONDUTORES S.A.
Av. Unisinos, 1550 | 93022-750 | São Leopoldo – RS | Brasil
www.htmicron.com.br

HT MICRON semicondutores

iMCP HTNB32L-XXX SDK User Manual

SDK User Manual for iMCP HTNB32L-XXX System-in-Package

Classification: CONFIDENTIAL

Doc. Type: USER MANUAL

Revision: v.01

Date: 08/08/2023

Code: HTNB32L-XXX-UM0004

SUMMARY

DOCUMENT INFO

This document provides technical information about the iMCP HTNB32L-XXX SDK content and how to use it. It is intended to contribute only the necessary information to understand and use iMCP HTNB32L-XXX SDK. To obtain access to the SDK, request access to the HTNB32L-XXX SDK and after having received the proper authorization, the content will be available in a private GitHub repository iMCP HTNB32L-XXX GitHub Repository.

1. GENERAL DESCRIPTION

The iMCP HTNB32L-XXX is a highly compact and low-power wireless communication MCO/SiP featuring Qualcomm QCX-212 LTE IoT Modem supporting single-mode 3GPP Release 14 Cat. NB2 IoT connectivity. Its SDK (Software Development Kit) provides OpenCPU solutions based on a FreeRTOS system, where users can embed their own IoT application, as well as AT Commands, used in a master-slave model.

2. SDK OVERVIEW

The Software Development Kit contains brief documentation, firmware, and demonstrations of using the iMCP HTNB32L-XXX. Figure 2.1 illustrates the SDK directory with contents and subdirectories.

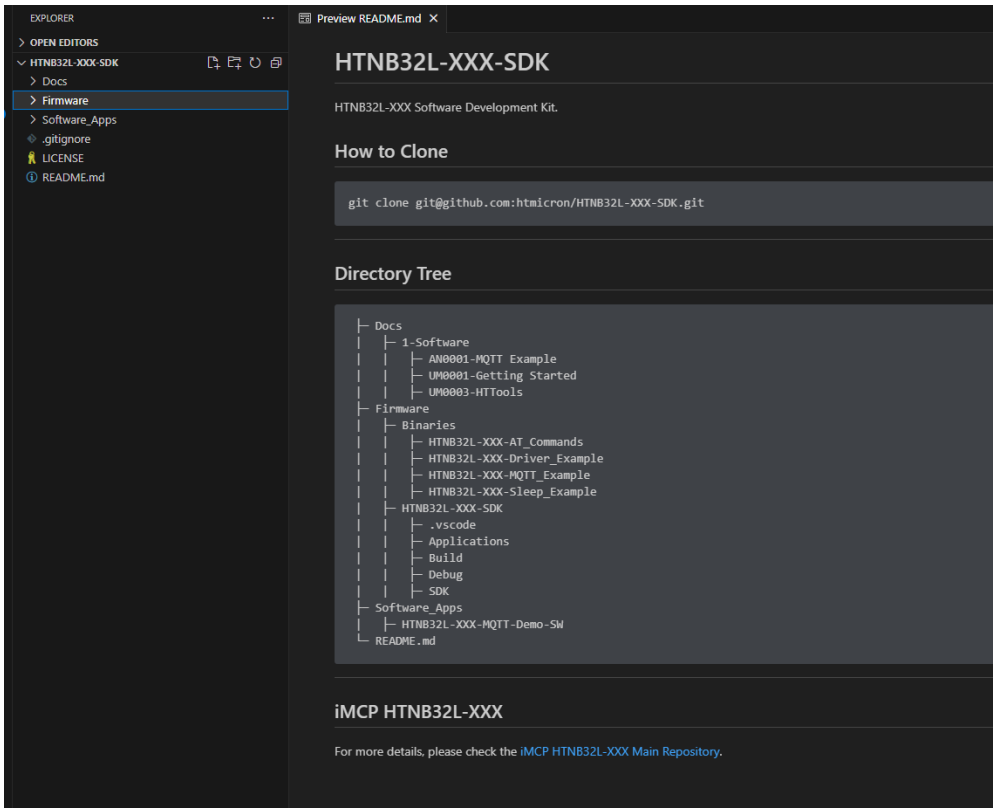


Figure 2.1 - SDK Overview.

2.1. HTNB32L-XXX-SDK/

The root directory of the SDK contains a README.md explaining how to clone and giving an overview of the environment's contents. It also contains the Apache License v2.0, more details about the license HERE. There are also *Docs*, *Firmware* and *Software_Apps* subdirectories.

2.1.1. Docs/

In the *Docs* directory you can find User Manuals that describe how to use the SDK. It also describes how to compile, write, and debug Firmware for the HTNB32L-XXX and develop your application on the SDK. The *Application Notes* bring details about the sample applications developed and how to use them.

2.1.2. Firmware/

The Firmware directory contains the working environment for the HTNB32L-XXX. In this directory you will find the *Binaries* subdirectory, containing examples of applications already compiled and ready to be flashed in the HTNB32L-XXX.

In the HTNB32L-XXX-SDK subdirectory you will find the Software Development Kit, with code and application examples. In the HTNB32L-XXX-SDK subdirectory there is the '*.vscode*' directory with the description of the

compilation, recording and debugging tasks used by VS Code. In the *Applications* directory there are examples of applications already developed by HT Micron, which can be used as a starting point for your application. The binaries and object files generated from the firmware compilation will be stored in the *Build* directory. This directory is created after compiling a SDK application. In the Debug directory are the scripts used in the recording and debugging process. In the *HT_Prebuild* and *SDK* directories are the precompiled libraries and their header files.

2.1.3. Software_Apps/

In the *Software_Apps* directory you will find specific software for some sample firmware applications.

For more information on how to setup the environment, how to compile, record and debug the environment see the Getting Started documentation at iMCP HTNB32L-XXX GitHub Repository.

2.2. MAKEFILE STRUCTURE

The SDK has a Makefiles structure to compile the applications and precompiled libraries needed for your firmware. This structure is composed of a main Makefile, your application's Makefile, Makefile.rules and Makefile.vars and several Makefile.inc according to the precompiled libraries. Figure 2.2 illustrates the SDK's Makefile structure.

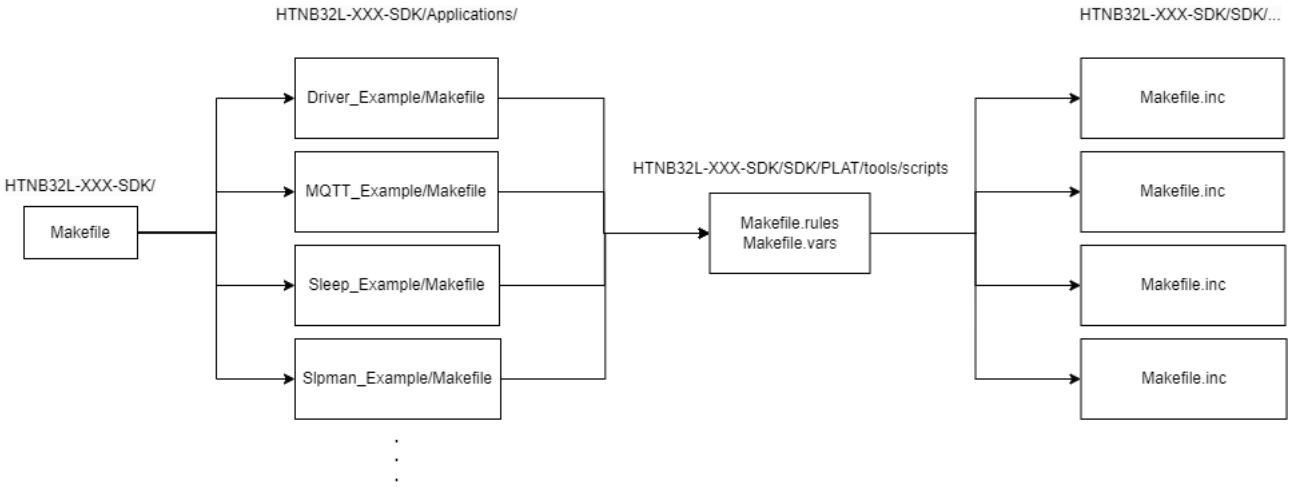


Figure 2.2 - Makefile Structure.

2.2.1. Main Makefile

The main Makefile is located at *HTNB32L-XXX-SDK/Firmware/HTNB32L-XXX-SDK/* directory. This is the Makefile responsible for calling the application that is currently being compiled. It is also responsible for cleaning up the build environment. See this Makefile in Figure 2.3:

Makefile

Firmware > HTNB32L-XXX-SDK > Makefile

```
.PHONY:gccall
gccall:
    (cd Applications/$(PROJECT) && $(MAKE) V=$(V) -f Makefile all)

.PHONY:clean_all
clean_all:
ifneq ("$(wildcard Build/)","")
    @$(RM) -r Build/
endif

.PHONY:clean-gccall
clean-gccall:
    (cd Applications/$(PROJECT) && $(MAKE) -f Makefile cleanall)

.PHONY:clean-gcc
clean-gcc:
    (cd Applications/$(PROJECT) && $(MAKE) -f Makefile clean)
.PHONY:size
size:
    (cd Applications/$(PROJECT) && $(MAKE) -f Makefile size)

.PHONY:jflash
jflash:
    (JLink -commanderscript Applications/$(PROJECT)/Launch/jflash)

```

Figure 2.3 - Main Makefile.

2.2.2. Application Makefile

Each application at *HTNB32L-XXX-SDK/Firmware/HTNB32L-XXX-SDK/Applications* directory has its own Makefile with specific application information. More details about this Makefile can be found in Section 3.1.1. Figure 2.4 presents the Makefile related to the Sleep_Example as an example:

Makefile

Firmware > HTNB32L-XXX-SDK > Applications > Sleep_Example > Makefile

```

AVAILABLE_TARGETS = qcx212_0h00
TOOLCHAIN         = GCC
BINNAME           = HTNB32L-XXX-Sleep_Example

TOP := ../..

BUILD_AT                = n
BUILD_AT_DEBUG          = n
THIRDPARTY_ROHC_ENABLE  = n
HT_USART_API_ENABLE := y
DRIVER_USART_ENABLE       = y
HT_DEFAULT_LINKER_FILE = y
HT_SPI_API_ENABLE := n
HT_I2C_API_ENABLE := n

HT_LIBRARY_SLEEP_ENABLE := y
UART_UNILOG_ENABLE = y

CFLAGS_INC        +=  -I Inc

obj-y             += Src/main.o \
                     Src/HT_BSP_Custom.o \
                     Src/HT_Sleep_Example.o

include $(TOP)/SDK/PLAT/tools/scripts/Makefile.rules

CFLAGS += -DSLEEP_EXAMPLE_TEST

```

Figure 2.4 - Sleep_Example Makefile.

2.2.3. Makefile.rules and Makefile.vars

Makefile.rules and Makefile.vars located at *HTNB32L-XXX-SDK/Firmware/HTNB32L-XXX-SDK/SDK/PLAT/tools/scripts/* are the Makefiles that effectively do the linking process with the precompiled libraries and binary generation. Makefile.rules includes Makefile.inc from precompiled libraries and builds the binary according to the targets defined by Makefile.vars. Figure 2.5 presents these Makefiles:

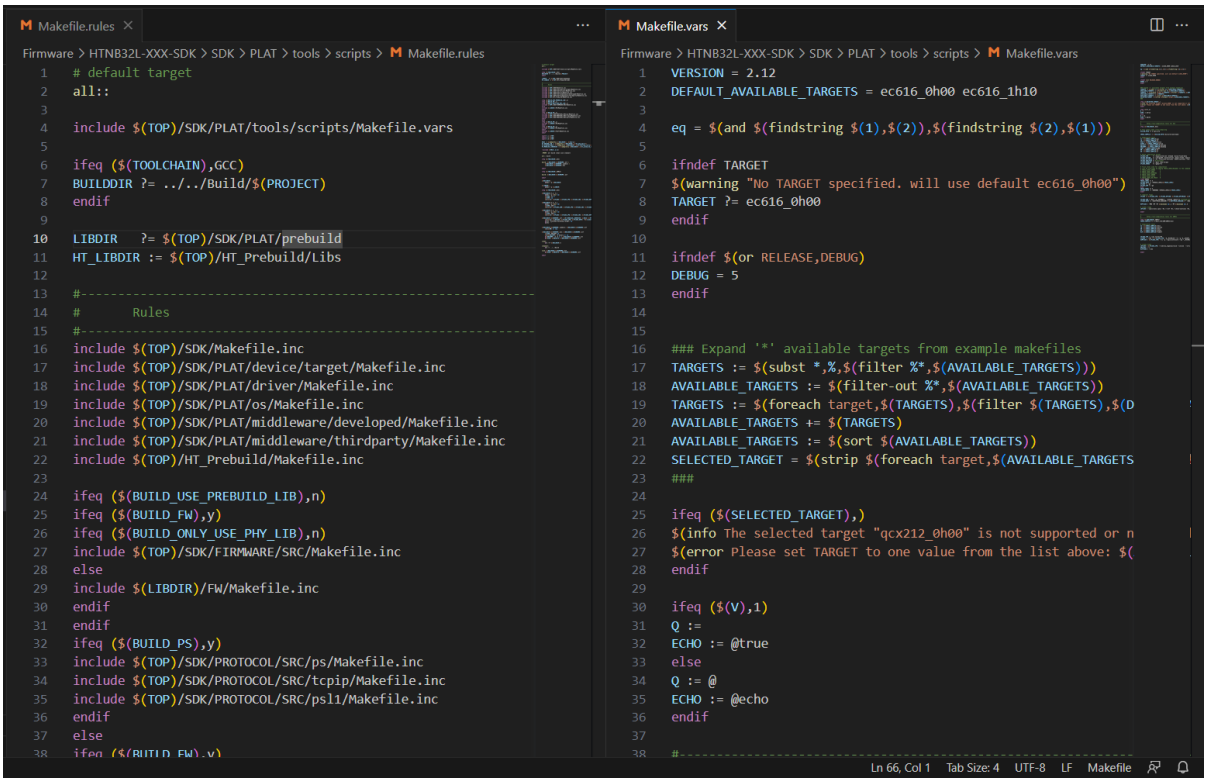


Figure 2.5 - Makefile.rules and Makefile.vars.

2.2.4. Makefile.inc

Are the Makefiles that define which precompiled libraries will be included in the application's binary. The Makefile.inc are responsible for defining which libraries will be added to the binary from the Macros defined in Application Makefile 2.2.2. Figure 2.6 demonstrate an example of Makefile.inc, which is available in *HTNB32L-XXX-SDK/HT_Prebuild/Makefile.inc*.

Makefile.inc

Firmware > HTNB32L-XXX-SDK > HT_Prebuild > Makefile.inc

```
    ht_prebuild_libraries :=

ht_prebuild_libraries += $(HT_LIBDIR)/libdriver_private_ht.a \
                         $(HT_LIBDIR)/libdriver.a \
                         $(HT_LIBDIR)/liblfs.a

ifeq ($(HT_LIBRARY_MQTT_ENABLE),y)
CFLAGS_INC +=   -I $(TOP)/HT_Prebuild/MQTT/MQTTClient/Inc \
                -I $(TOP)/HT_Prebuild/MQTT/FreeRTOS/Inc \
                -I $(TOP)/HT_Prebuild/MQTT/MQTTPacket/Inc

ht_prebuild_libraries += $(HT_LIBDIR)/libfreertos.a \
                         $(HT_LIBDIR)/libHTmqtt.a \
                         $(HT_LIBDIR)/libiperf.a \
                         $(HT_LIBDIR)/liblwip.a \
                         $(HT_LIBDIR)/libmbedtls.a \
                         $(HT_LIBDIR)/libmiddleware_ec.a \
                         $(HT_LIBDIR)/libping.a \
                         $(HT_LIBDIR)/libpsnv.a \
                         $(HT_LIBDIR)/libsntp.a

endif

ifeq ($(HT_LIBRARY_CJSON_ENABLE),y)
CFLAGS_INC += -I $(TOP)/HT_Prebuild/CJSON/Inc
ht_prebuild_libraries += $(HT_LIBDIR)/libHTcjson.a
endif

ifeq ($(HT_LIBRARY_SLPMAN_ENABLE),y)

ht_prebuild_libraries += $(HT_LIBDIR)/libfreertos.a \
                         $(HT_LIBDIR)/libiperf.a \
                         $(HT_LIBDIR)/liblwip.a \
                         $(HT_LIBDIR)/libmiddleware_ec.a \
                         $(HT_LIBDIR)/libping.a \
                         $(HT_LIBDIR)/libpsnv.a \
                         $(HT_LIBDIR)/libsntp.a
```

Figure 2.6 - Makefile.inc.

3. APPLICATION DESCRIPTION

To develop your own application, it is recommended to start from one of the examples available in *Firmware/HTNB32L-XXX-SDK/Applications/*. In this directory you will find four examples of applications to use in your firmware: *Driver_Example, MQTT_Example, Sleep_Example and Slpman_Example*. For more details on these applications, consult the Application Notes in the *Docs* directory. If you don't want to use an example, you can create a directory at *Firmware/HTNB32L-XXX-SDK/Applications/* containing your source code.

NOTE

Open *Firmware/HTNB32L-XXX-SDK/* in VS Code to build and debug tasks present in *Firmware/HTNB32L-XXX-SDK/.vscode* (for more details, see Getting Started User Manual at HTNB32L-XXX GitHub)

3.1. APPLICATION STRUCTURE

The application directory where users are supposed to create their application is *Firmware/HTNB32L-XXX-SDK/Applications/*. Figure 3.1 demonstrates a directory structure example that is used in the MQTT_Example:

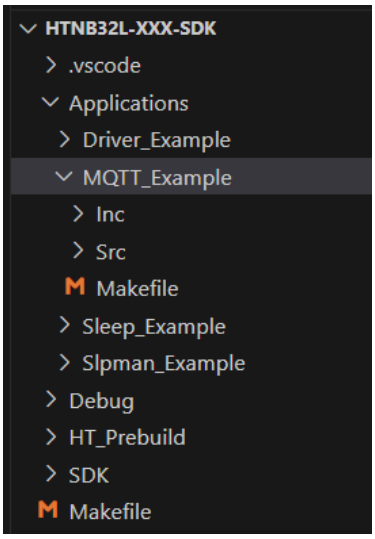


Figure 3.1- MQTT_Example directory structure.

3.1.1. Makefile

It is mandatory to have a Makefile to compile your customized application and the expected libraries. Figure 3.2 illustrates a Makefile located at *HTNB32L-XXX-SDK/Firmware/HTNB32L-XXX-SDK/Applications/MQTT_Example* example with the necessary information for building your own script:

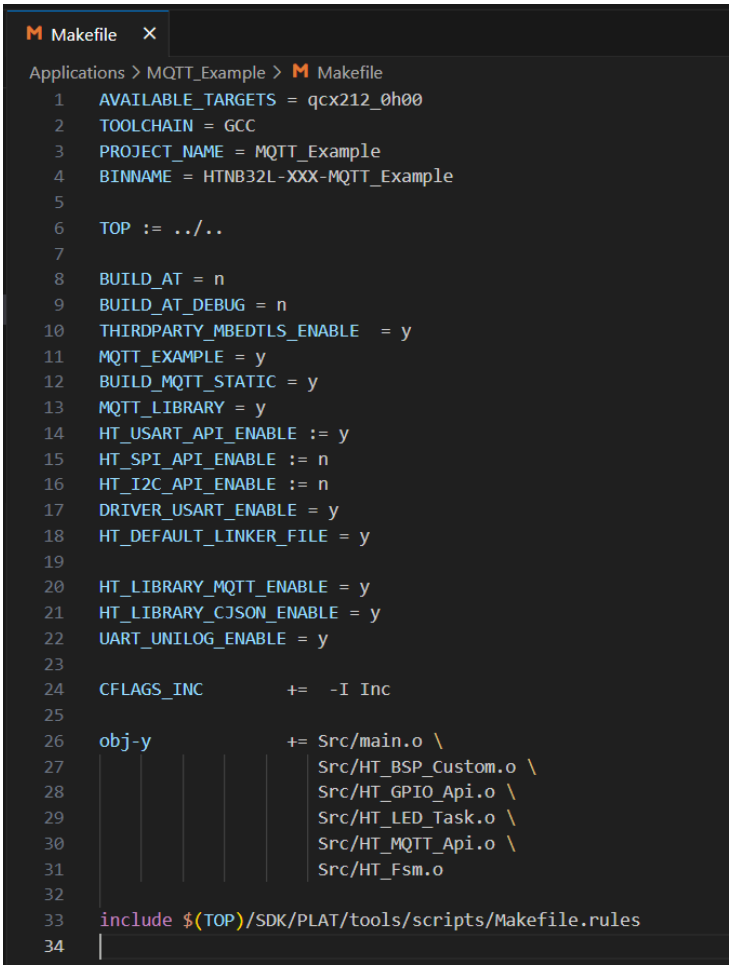


Figure 3.2 – Makefile.

This Makefile contains important definitions for compiling the application, such as the project name, libraries to be included and source files that will be compiled and compilation rules. At the top of the Makefile is the header with some important Macros demonstrated in the Table 3.1:

Table 3.1- Makefile Macros

Macro	Description
AVAILABLE_TARGETS	Must be "qcx212_0h00"
TOOLCHAIN	Must be "GCC"
PROJECT_NAME	Your Project name
BINNAME	Binary file name
TOP	HTNB32L-XXX-SDK directory

Figure 3.3 indicates in the Makefile the libraries that will be included in the assembly of the binary file. In this section of the file, users must include all the libraries necessary for its application. To include a library, write its name followed by "= y", so the compilation script will include the desired library in your application binary.

```
BUILD_AT = n
BUILD_AT_DEBUG = n
THIRDPARTY_MBEDTLS_ENABLE  = y
MQTT_EXAMPLE = y
BUILD_MQTT_STATIC = y
MQTT_LIBRARY = y
HT_USART_API_ENABLE := y
HT_SPI_API_ENABLE := n
HT_I2C_API_ENABLE := n
DRIVER_USART_ENABLE = y
HT_DEFAULT_LINKER_FILE = y

HT_LIBRARY_MQTT_ENABLE = y
HT_LIBRARY_CJSON_ENABLE = y
UART_UNILOG_ENABLE = y
```

Figure 3.3 – Makefile Libraries.

Users also must indicate the directory of its application's header files through the macro "CFLAGS_INC += -I Inc". Then, indicate which objects (.o files) will be compiled. Since these objects are generated from the source files, they must have the same name as their respective source file. This indication should be given through the macro "obj-y += Src/your_source.o". Finally, the Makefile.rules must be included at the end of the customized Makefile. This whole process is illustrated in Figure 3.4:

```
CFLAGS_INC        +=  -I Inc

obj-y             += Src/main.o \
                     Src/HT_BSP_Custom.o \
                     Src/HT_GPIO_Api.o \
                     Src/HT_LED_Task.o \
                     Src/HT_MQTT_Api.o \
                     Src/HT_Fsm.o
```

Figure 3.4- Makefile Inc and Src.

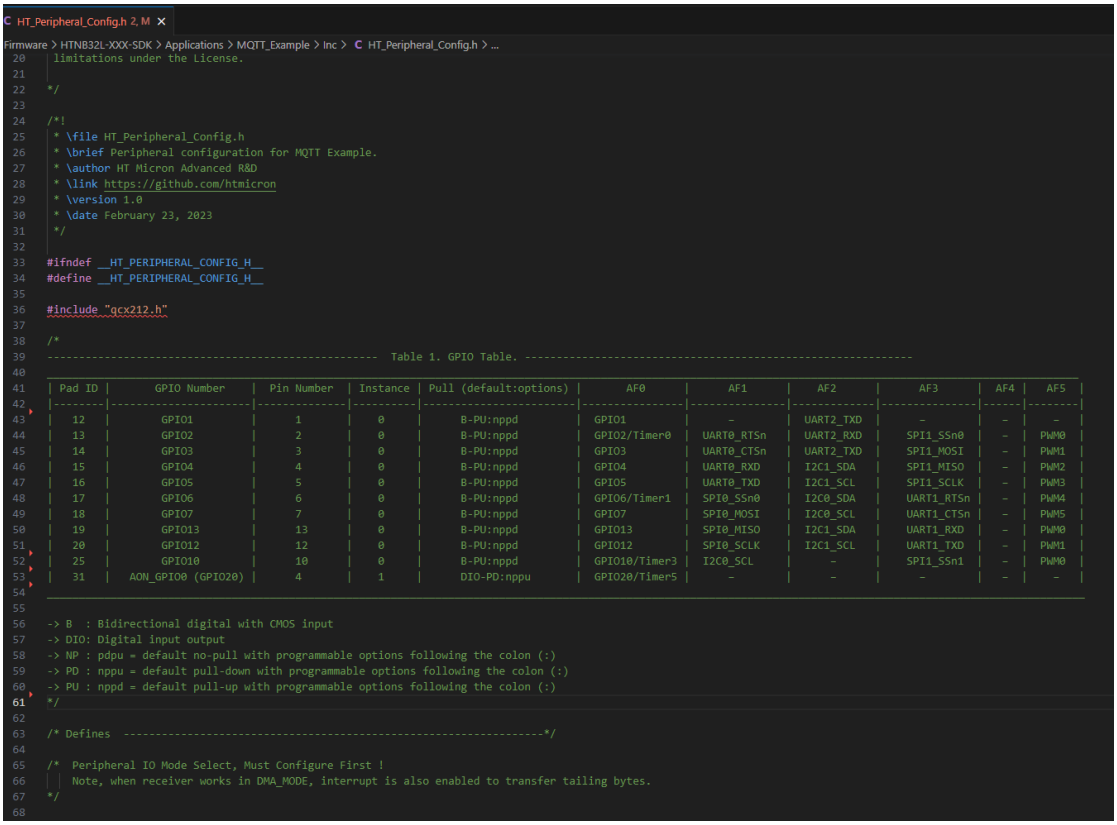
3.1.2. Src

The *Src* directory should contain the application's source files. Here, users are supposed to create the source files according to the needs of their application. It is mandatory to include these files in the Makefile of the respective application, in order to build them during the compiling process.

3.1.3. Inc

3.1.3.1. HT_Peripheral_Config.h

The Inc directory should contain all the application's header files, with prototype functions that are used externally. This directory must contain the "HT_Peripheral_Config.h" file, responsible for defining the functions of the HTNB32L-XXX pins, as it is illustrated in Figure 3.5:



HT_Peripheral_Config.h 2, M

Firmware > HTNB32L-XXX-SDK > Applications > MQTT_Example > Inc > C HT_Peripheral_Config.h > ...

```
 20   limitations under the License.
 21
 22  */
 23
 24  /*!
 25   * \file HT_Peripheral_Config.h
 26   * \brief Peripheral configuration for MQTT Example.
 27   * \author HT Micron Advanced R&D
 28   * \link https://github.com/htmicron
 29   * \version 1.0
 30   * \date February 23, 2023
 31   */
 32
 33  #ifndef __HT_PERIPHERAL_CONFIG_H__
 34  #define __HT_PERIPHERAL_CONFIG_H__
 35
 36  #include "qcx212.h"
 37
 38  /*
 39  ------------------------------------------------  Table 1. GPIO Table. ------------------------------------------------------------
 40
 41  | Pad ID |       GPIO Number     | Pin Number | Instance | Pull (default:options) |       AF0       |      AF1      |     AF2     |      AF3      | AF4 |   AF5   |
 42  |--------|-----------------------|------------|----------|------------------------|-----------------|---------------|-------------|---------------|-----|---------|
 43  |   12   |        GPIO1          |      1     |    0     |      B-PU:nppd         |  GPIO1          |       -       |  UART2_TXD  |       -       |  -  |    -    |
 44  |   13   |        GPIO2          |      2     |    0     |      B-PU:nppd         |  GPIO2/Timer0   |  UART0_RTSn   |  UART2_RXD  |    SPI1_SSn0  |  -  |  PWM0   |
 45  |   14   |        GPIO3          |      3     |    0     |      B-PU:nppd         |  GPIO3          |  UART0_CTSn   |  UART2_TXD  |    SPI1_MOSI  |  -  |  PWM1   |
 46  |   15   |        GPIO4          |      4     |    0     |      B-PU:nppd         |  GPIO4          |  UART0_RXD    |  I2C1_SDA   |    SPI1_MISO  |  -  |  PWM2   |
 47  |   16   |        GPIO5          |      5     |    0     |      B-PU:nppd         |  GPIO5          |  UART0_TXD    |  I2C1_SCL   |    SPI1_SCLK  |  -  |  PWM3   |
 48  |   17   |        GPIO6          |      6     |    0     |      B-PU:nppd         |  GPIO6/Timer1   |  SPI0_SSn0    |  I2C0_SDA   |    UART1_RTSn |  -  |  PWM4   |
 49  |   18   |        GPIO7          |      7     |    0     |      B-PU:nppd         |  GPIO7          |  SPI0_MOSI    |  I2C0_SCL   |    UART1_CTSn |  -  |  PWM5   |
 50  |   19   |        GPIO13         |     13     |    0     |      B-PU:nppd         |  GPIO13         |  SPI0_MISO    |  I2C1_SDA   |    UART1_RXD  |  -  |  PWM0   |
 51  |   20   |        GPIO12         |     12     |    0     |      B-PU:nppd         |  GPIO12         |  SPI0_SCLK    |  I2C1_SCL   |    UART1_TXD  |  -  |  PWM1   |
 52  |   25   |        GPIO10         |     10     |    0     |      B-PU:nppd         |  GPIO10/Timer3  |  I2C0_SCL     |      -      |    SPI1_SSn1  |  -  |  PWM0   |
 53  |   31   |   AON_GPIO0 (GPIO20)  |      4     |    1     |      DIO-PD:nppu       |  GPIO20/Timer5  |       -       |      -      |       -       |  -  |    -    |
 54
 55
 56  -> B  : Bidirectional digital with CMOS input
 57  -> DIO: Digital input output
 58  -> NP : pdpu = default no-pull with programmable options following the colon (:)
 59  -> PD : nppu = default pull-down with programmable options following the colon (:)
 60  -> PU : nppd = default pull-up with programmable options following the colon (:)
 61  */
 62
 63  /* Defines  ------------------------------------------------------------------*/
 64
 65  /*  Peripheral IO Mode Select, Must Configure First !
 66      Note, when receiver works in DMA_MODE, interrupt is also enabled to transfer tailing bytes.
 67  */
 68
```

Figure 3.5 - HT_Peripheral_Config.h.

This header file contains a table with the GPIO pins available on the HTNB32L-XXX and their alternative functions. This information will be used to initialize the pins in your application.

3.1.3.2. Peripheral Initialization

This file is where users must specify the serial peripherals that are supposed to be compiled in their application. It is also responsible for configuring the pins of these peripherals, as well as their respective operating mode (*POLLING_MODE*, *DMA_MODE*, *IRQ_MODE* or *UNILOG_MODE*). Figure 3.6 demonstrates an example of how these operating modes can be configured:

```
69   #define POLLING_MODE            0x1
70   #define DMA_MODE                0x2
71   #define IRQ_MODE                0x3
72   #define UNILOG_MODE             0x4
73
74   #define RTE_UART0_TX_IO_MODE    UNILOG_MODE
75   #define RTE_UART0_RX_IO_MODE    DMA_MODE
76   #define USART0_RX_TRIG_LVL      (30)
77
78   #define RTE_UART1_TX_IO_MODE    POLLING_MODE
79   #define RTE_UART1_RX_IO_MODE    DMA_MODE
80
81   #define RTE_UART2_TX_IO_MODE    POLLING_MODE
82   #define RTE_UART2_RX_IO_MODE    DMA_MODE
83
84   #define RTE_SPI0_IO_MODE          POLLING_MODE
85
86   #define RTE_SPI1_IO_MODE          POLLING_MODE
87
88   #define I2C0_INIT_MODE          POLLING_MODE
89   #define I2C1_INIT_MODE          POLLING_MODE
```

Figure 3.6 - Peripheral Initialization.

After selecting the operating mode of the serial interfaces used in the customized application, users can enable the interfaces that they will utilize. To this end, they can just assign the value 1 in the macros "RTE_interfacex". An example can be found at line 94, which is illustrated in Figure 3.7, where it is possible to enable the I2C0 interface by assigning the value 1. Similar procedure shall be followed to disable any serial interface, by assigning the value 0 to their respective macros.

```
92   // I2C0 (Inter-integrated Circuit Interface) [Driver_I2C0]
93   // Configuration settings for Driver_I2C0 in component ::Drivers:I2C
94   #define RTE_I2C0                    1
```

Figure 3.7 - Peripheral enable.

3.1.3.3. Peripheral Pinout

Here is how to select the output pins of the interface. On lines 98 to 102 of Figure 3.8 are the macros that define the PAD_ID of the I2C0 interface and the Alternative Function of this PAD_ID. Users must assign in this macro the PAD_ID and AF values presented in the GPIO Table of the HT_Peripheral_Config.h file. This process is the same for all peripheral available in HTNB32L-XXX.

```
97
98   // { PAD_PIN18},  // 0 : gpio7  / 2 : I2C0 SCL
99   // { PAD_PIN17},  // 0 : gpio6  / 2 : I2C0 SDA
100  #define RTE_I2C0_SCL_PAD_ID              18
101  #define RTE_I2C0_SCL_FUNC              PAD_MuxAlt2
102
103  #define RTE_I2C0_SDA_PAD_ID              17
104  #define RTE_I2C0_SDA_FUNC              PAD_MuxAlt2
105
```

Figure 3.8 - Peripheral Pinout

3.2. Libraries for Applications

3.2.1. Precompiled libraries

It is possible to include several precompiled libraries to enable firmware functions that can extend application functionality. In the *HTNB32L-XXX-SDK/Firmware/HTNB32L-XXX-SDK/HT_Prebuild* directory, are the compiled libraries for example applications (see section 2.1.2). This section describes how to add some peripheral libraries to firmware.

Libraries are added from a macro defined in the application's Makefile. In MQTT_Example, two macros are used to add libraries: HT_LIBRARY_MQTT_ENABLE and HT_LIBRARY_CJSON_ENABLE, see Figure 3.9.

```
20    HT_LIBRARY_MQTT_ENABLE = y
21    HT_LIBRARY_CJSON_ENABLE = y
```

Figure 3.9 - MQTT Libraries Enable

By enabling these libraries in the application's Makefile, it is possible to add the precompiled libraries illustrated in Figure 3.10. These libraries are responsible for basic firmware functions, such as real-time system, peripherals and NB-IoT protocol. It also includes some application-specific libraries as an MQTT package.

```
7   ifeq ($(HT_LIBRARY_MQTT_ENABLE),y)
8   CFLAGS_INC +=   -I $(TOP)/HT_Prebuild/MQTT/MQTTClient/Inc \
9                   -I $(TOP)/HT_Prebuild/MQTT/FreeRTOS/Inc \
10                  -I $(TOP)/HT_Prebuild/MQTT/MQTTPacket/Inc
11
12  ht_prebuild_libraries += $(HT_LIBDIR)/libfreertos.a \
13                           $(HT_LIBDIR)/libHTmqtt.a \
14                           $(HT_LIBDIR)/libiperf.a \
15                           $(HT_LIBDIR)/liblwip.a \
16                           $(HT_LIBDIR)/libmbedtls.a \
17                           $(HT_LIBDIR)/libmiddleware_ec.a \
18                           $(HT_LIBDIR)/libping.a \
19                           $(HT_LIBDIR)/libpsnv.a \
20                           $(HT_LIBDIR)/libsntp.a
21
22  endif
23
24  ifeq ($(HT_LIBRARY_CJSON_ENABLE),y)
25  CFLAGS_INC += -I $(TOP)/HT_Prebuild/CJSON/Inc
26  ht_prebuild_libraries += $(HT_LIBDIR)/libHTcjson.a
27  endif
```

Figure 3.10 - HT_Prebuild/Makeile.inc libraries

3.2.2. Including ADC

To add Analog to Digital Converter to your application you can follow the steps below:

- Add the compilation macro “DRIVER_ADC_ENABLE = y” in the application Makefile (Figure 3.11).

```
24    DRIVER_ADC_ENABLE = y
```

Figure 3.11 - DRIVER_ADC_ENABLE

- Add the ADC headers to your application: #include “hal_adc.h” and #include “adc_qcx212.h” as shown in Figure 3.12.

```
49    #include "hal_adc.h"
50    #include "adc_qcx212.h"
```

Figure 3.12 - ADC Headers Include

With these steps you were able to add the Analog to Digital Converter to your application. It is now possible to call the functions described in the headings “hal_adc.h” and “adc_qcx212.h” to use the ADC.

ABBREVIATIONS

Table 0.1: Abbreviations

Acronym	Description
LTE	Long Term Evolution
IoT	Internet of Things
GPIO	General Purpose Input Output
SDK	Software Development Kit
VSCODE	Visual Studio Code
MQTT	Message Queuing Telemetry Transport
I2C	Inter-Integrated Circuit
NB-IoT	Narrow Band – Internet of Things
ADC	Analog to Digital Converter

LIST OF FIGURES

LIST OF TABLES

REVISION HISTORY

Version	Date	Changes	Authors
00	29/08/2023	- Initial draft	GAA
01	26/09/2023	- Review for SDK V0.2	MSZ

CONTACT

HT MICRON SEMICONDUTORES S.A.
Av. Unisinos, 1550 | 93022-750 | São Leopoldo | RS | Brasil
www.htmicron.com.br

DOCUMENT INFORMATION

Document Title:	iMCP HTNB32L-XXX SDK User Manual
Document Subtitle:	SDK User Manual for iMCP HTNB32L-XXX System-in-Package
Classification:	CONFIDENTIAL
Doc. Type:	USER MANUAL
Revision:	v.01
Date:	08/08/2023
Code:	HTNB32L-XXX-UM0004

DISCLAIMER