Cellular:Cellular X-CUBE-CELLULAR

Click here for Cellular overview

Contents

1 Introduction

This wiki article contains information to help the user navigate ST Cellular software.

2 General description

The X-CUBE-CELLULAR Expansion Package only provides software components running on the host STM32 MCU. Cellular modem firmware is not within the scope of this document. The following integrated development environments are supported:

- STMicroelectronics integrated development environment for STM32 products (STM32CubeIDE)
- IAR Systems® IAR Embedded Workbench® for Arm® (EWARM)
- Keil® Microcontroller Development Kit (MDK-ARM)

3 X-CUBE-CELLULAR versions

Since the initial version **V1.0.0** in mid-2018, the software content and hardware support have increased continuously. Versions 1.0.0 to 5.2.0 progressively encompassed new features, applications, and hardware. The cellular middleware however was part of X-CUBE, which is not sufficiently independent.
Therefore, **V6.0.0** is reworked to provide standalone cellular middleware that is easier to integrate into existing applications. Therefore the boot menu has been removed but command line calls are still available. The cellular middleware API is also redesigned to ease/clarify its usage by application. Finally, V6.0.0 is the first delivery that introduces low power modem features for some modems (BG96 and Murata).
The current version, **V6.0.0**, supports the following **hardware configurations**:

- P-L496G-CELL02[1] (pack with a screenless 32L496GDISCOVERY associated with an STMod+ BG96 Quectel modem)
- B-L4S5I-IOT01A[2] (STM32L4+ Discovery kit IoT node)
- 32L496GDISCOVERY[3] (Discovery kit with STM32L496AG MCU)
- STEVAL-STWINKT1B[4] (STWIN SensorTile Wireless Industrial Node development kit and reference design for industrial IoT applications)
- B-L462E-CELL1[5] (Discovery board with T1SE Murata module (STM32 L462, ST4SIM, ALT1250 modem))
- X-NUCLEO-STMODA1[6] (STMod+ connector expansion board for ARDUINO® connectors)
- STEVAL-STMODLTE[7] (LTE connectivity expansion board for STMod+ connector compatible evaluation boards, based on the Quectel BG96 modem, the same modem board in the P-L496G-CELL02 pack described above)
- GM01Q-STMOD[8] or GMS01Q-STMOD[9] (Monarch from Sequans, GMS01Q is a GM01Q that also contains an internal SIM with a prepaid data plan).

P-L496G-CELL02 and B-L462E-CELL1 are standalone setups. For other STM32 boards (B-L4S5I-IOT01A, 32L496GDISCOVERY, STEVAL-STWINKT1B), just plug in the STEVAL-STMODLTE or GM01Q-STMOD modem boards.
Note: for B-L4S5I-IOT01A, an X-NUCLEO-STMODA1 (ARDUINO® / STMod+ adapter) must be used.
For the complete description of deliveries (software and supported hardware), please refer to the release notes included in the X-CUBE package.

4 Modem socket versus LwIP

Either a modem socket or LwIP can be used for the IP stack:

- Modem socket: the IP stack runs in modem firmware
- LwIP: the LwIP stack runs on the STM32 side

This option is selected in software through a #define used during the compilation process.

Note:Only IPV4 is supported.

If LwIP is selected, the communication between host and modem is done through the PPP layer. There is a PPP client on the host side, and a PPP server on the modem side. PPPoSIF adapts the LwIP stack to a serial interface, while LwIP usually uses Ethernet interfacing.

5 What is X-CUBE-CELLULAR?

X-CUBE-CELLULAR (or XCC) allows applications to send and receive data on the cellular network using BSD API sockets.

Remember that the official web page on X-CUBE-CELLULAR[10] on www.st.com provides access to multiple resources (software, documents, and so on).

X-CUBE-CELLULAR consists of a set of libraries and application examples for STM32L4 Series MCUs acting as hosts for cellular connectivity applications.

SW Architecture diagram of Cellular stack

The colors highlight the different parts, **light green** for applications, **light blue** for X-CUBE-CELLULAR specific components, **dark blue** for components from STM32Cube Firmware (aka HAL), **dark green** for ecosystem.

L4 family STM32 MCUs are currently supported, and more families are to be added in the future. However it is easy to port the cellular middleware to other STM32 MCU families thanks to STM32Cube Firmware HAL and using STM32CubeMX

6 Software architecture

6.1 API description

The X-CUBE-CELLULAR API of the cellular package is defined by two components:

- **Cellular management (Ctrl):** Initialization/starting of cellular features. It also represents the control plane.
- **Com:** Data communication API based on the BSD API. It represents the data plane.

6.2 Component description

- **Cellular App:** Implements an echo client application by sending data to a remote echo server and waiting for the return of the same packet. It provides a simple example using the TCP or UDP protocol socket. It also implements a ping service and provides the throughput at application level. Two instances can be started simultaneously to send and receive data.
- **PPP client task / PPPoSIF:** Optional component. It is only present when LwIP is used, and is in charge of establishing the PPP link with the modem.
- **LwIP / Net IF:** The LwIP component and its adaptation to PPP.
- **Cellular Service:**
  - **Cellular Service task:** Controls modem power-on and initialization, instructs the modem to perform network registration, activates the PDN (PDP context), and enters data transfer mode. It uses the cellular service OS library to send AT commands to the modem.
  - **Cellular Service OS:** A library that offers a collection of functions to the low-level cellular service. The library serializes the access to the single AT channel interface that is used to communicate with the modem. The functions are called by the COM service and the cellular service task.
  - **Cellular Service library:** Offers a collection of function calls to perform transactions with a modem. Each function translates the transaction request to service requested and calls the service provided by the AT service in order to send the sequence of AT commands related to the service requested. The function returns only when all responses are received, or an error or timeout has expired from the AT service.
- **AT Service:** Provides a function to translate the cellular service requested into one or several transactions of AT commands to the modem. It sends the AT command either using a generic 3GPP command, or by using the modem-specific AT command. It is also in charge of processing AT command responses and URCs from the modem, and forwards them to the cellular service. AT is split into two parts, a **generic** part, "Core" (AT framework and manage standard AT commands) and a **specific** custom part (implements specific modem behavior and AT commands).
- **Modem system control:** Supports modem hardware system control signaling (power on/off, reset). It is split into a generic and a specific part. The generic part exposes the generic API to the application (cellular service) while the specific part controls the GPIO dedicated to the modem.
- **IPC:** Manages circular RX FIFO buffer to receive data from the modem using the physical interface. When a complete frame or character is received the IPC calls a callback function provided by the upper layer for processing. The upper layer can then call a function to read the data from the RX FIFO buffer. It supports two channels: character mode (used to send AT commands) and stream mode (used for data transfer (PPP)). It abstracts the physical interface for communication with the Modem, and provides a collection of functions that initialize, send, and receive data from a modem.
- **Data_Cache:** This component allows the sharing of data between components. A particular component can be notified via an event callback when an element of the Data_Cache has been modified by another component.
- **FreeRTOS™:** Provides RTOS services to create the resources and scheduler needed by the software to run, such as threads, dynamic memory allocation, mutexes, and semaphores. A default task (freertos.c) is in charge of system initialization, and creation of all application tasks.
- **RTOSAL:** Provides the CMSIS V1/V2 abstraction. The application uses the RTOSAL API; according to a flag, CMSIS V1 or CMSIS V2 is used.

7 Applications provided

A single application, Cellular App, is provided with examples of all the supported hardware configurations. It allows the cellular middleware to be tested, and contains the following features:

- ECHO to provide an example of connection and data exchanges using the TCP or UDP (connected or not-connected mode) socket protocols. By default one instance of ECHO is started but a second one can be started to test multiple-socket configurations
- PING to test the access to a remote machine
- Performance to test throughput.

At boot, after hardware initialization, Cellular App is started and this then starts the cellular middleware.
X-CUBE-CELLULAR also contains a command tool CLI (to interact with the firmware through a terminal on a connected PC)

The following hardware setup (B-L462E-CELL1) embeds more functionalities than the others. It also includes BSP for sensors and display, and a complete IoT application that: displays information on the screen, reads humidity, pressure, temperature sensors, and uses cellular middleware to exchange with the network (Echo, Ping).

8 Run-time configuration

During firmware execution, the command-line interface (CLI) is entered by pressing the [return] key. For example, the help command lists all available components.

Command	Object
help	Help command
cellularapp	Cellular App commands
echoclient	Echo client commands
ping	Ping commands
trace	Trace management
csp	Power management
comlib	Com library commands
cst	Cellular service task management
atcmd	Send an AT command
modem	Modem configuration management

Command examples:

- cst info: provides information about the cellular network
- trace off : stops the trace
- echo off: stops the Echo application

The CLI and traces are displayed on the same terminal.

9 Board support package

The X-CUBE-CELLULAR targets 2 goals, the first is to demonstrate the cellular middleware based on multiple hardware setups. The second is to demonstrate the B-L462E-CELL1.
In the X-CUBE-CELLULAR the BSP is only available for B-L462E-CELL1 boards.
For users who need to develop an application that uses sensors on supported hardware setups other than B-L462E-CELL1, please refer to the Cube firmware deliveries to copy/paste/adapt for cellular usage.
The B-L462E-CELL1 BSP is composed of 2 parts, the board part (Drivers/BSP/B-L462E-CELL1) and the component part (Drivers/Components/).
All the required component drivers are listed below:

- hts221 (capacitive digital sensor for relative humidity and temperature) from STMicroelectronics
- lps22hh (260-1260 hPa absolute digital output barometer) from STMicroelectronics
- lsm303agr (Ultra-low-power 3D accelerometer and 3D magnetometer) from STMicroelectronics
- ssd1315 (display driver for 0.96-inch 128 x 64 OLED / SPI Ref=9OL9935701000)
- M24128 (128-Kbit serial I²C bus EEPROM) from STMicroelectronics
- w25q80ew (8 Mbit-bit hence 1 MByte serial Flash memory) from Winbond
- mt25qu512 (MT25Q Serial NOR Flash, 512 Mbit hence 64 MByte) from Micron.

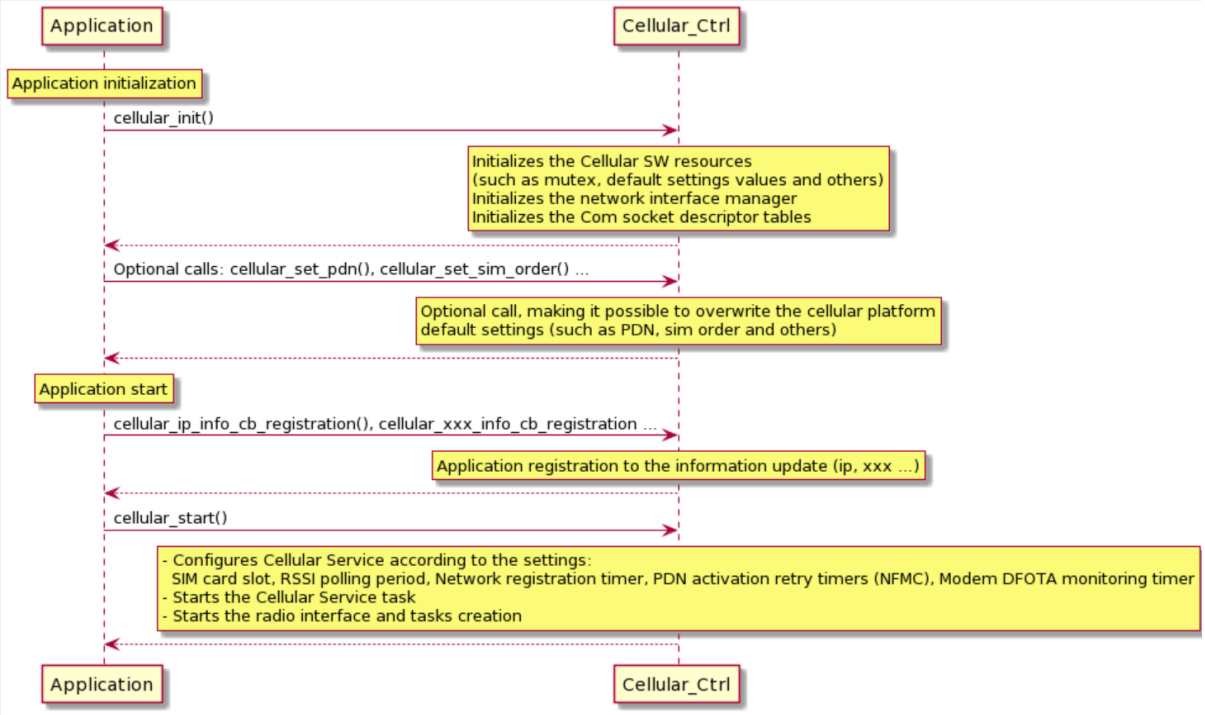
Notes:

- An EEPROM driver is not needed because the board BSP uses the HAL API directly
- The w25q80ew QSPI is internal Flash memory located in the Murata module
- The mt25qu512 QSPI is an external Flash memory soldered on the board outside the module
- The 2 QSPIs are exclusive and the default is the internal one. To use external QSPI soldering is needed, information about this is available in the board user manual.

QSPI drivers are to be provided into next delivery, X-CUBE-CELLULAR V6.1.0.

10 X-CUBE-CELLULAR dynamics

The following sequence diagram shows the cellular middleware setup from an application, more sequence diagrams are to be provided soon.



11 Folder structure

- STM32CubeExpansion_CELLULAR_V6.0.0
  - Drivers\
    - BSP\
      - B-L462E-CELL1\
      - Components\
        - Common\
        - hts221\
        - lps22hh\
        - ...
      - X_STMOD_PLUS_MODEMS\
        - BG96\
        - MONARCH\
        - TYPE1SC\
    - CMSIS\
    - STM32L4xx_HAL_Driver\
  - Middlewares\ **(Internal and external middleware)**
    - ST\ **(STMicroelectronics (internal) middleware)**
      - STM32_Cellular\
    - Third_Party\ **(Third-party (external middleware)**
      - FreeRTOS\
      - LwIP\
  - Projects\ **(Projects, one directory per board supported)**
    - 32L496GDISCOVERY\ **(One of the supported board)**
    - B-L462E-CELL1\ **(One of the supported board)**
      - Demonstrations\
        - Cellular\
          - Binaries\
          - Core\
          - IDE\ **(Supported IDEs)**
            - EWARM\
            - MDK-ARM\
            - STM32CubeIDE\
          - STM32_Cellular\
            - Config\
            - Target\
        - CellularIoT\
    - B-L4S5I-IOT01A\ **(One of the supported board)**
    - Misc\
      - Cmd\
      - RTOS\

- Samples\
  - Cellular\
    - CellularIoT\
- STWINKT1\ **(One of the supported board)**

- Utilities\
  - Fonts\
  - LCD\
  - Modem_FW\ **(material to allow modem FW update)**

12 Real network or simulator

Users must ensure that network coverage with the target technology is available at their location since 2G, LTE Cat M and NB-IoT are not available worldwide. If the live network is not available for the target technology, the user can use a network simulator such as Amarisoft's. To run the example, it is assumed that a network connection is possible with the device.

13 Miscellaneous information

To support low power, the application must close the socket before entering low power mode.

14 References

- ↑ P-L496G-CELL02
- ↑ B-L4S5I-IOT01A
- ↑ 32L496GDISCOVERY
- ↑ STEVAL-STWINKT1B
- ↑ B-L462E-CELL1
- ↑ X-NUCLEO-STMODA1
- ↑ STEVAL-STMODLTE
- ↑ GM01Q-STMOD
- ↑ GMS01Q-STMOD
- ↑ X-CUBE-CELLULAR