

QuecPython Technology and Resources Overview

LTE Standard Module Series

Version: 1.0.0

Date: 2020-11-11

Status: Preliminary



Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.

Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China

Tel: +86 21 5108 6236

Email: info@quectel.com

Or our local office. For more information, please visit:

<http://www.quectel.com/support/sales.htm>.

For technical support, or to report documentation errors, please visit:

<http://www.quectel.com/support/technical.htm>

Or email to support@quectel.com.

General Notes

Quectel offers the information as a service to its customers. The information provided is based upon customers' requirements. Quectel makes every effort to ensure the quality of the information it makes available. Quectel does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information. All information supplied herein is subject to change without prior notice.

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

The information contained here is proprietary technical information of Quectel Wireless Solutions Co., Ltd. Transmitting, reproducing, disseminating and editing this document as well as using the content without permission are forbidden. Offenders will be held liable for payment of damages. All rights are reserved in the event of a patent grant or registration of a utility model or design.

Copyright © Quectel Wireless Solutions Co., Ltd. 2020. All rights reserved.

About the Document

Revision History

| Version | Date | Author | Description |
|---------|------------|---------------|-------------|
| - | 2020-11-11 | Kinney/Kingka | Initial |
| 1.0.0 | 2020-11-11 | Kinney/Kingka | Initial |

Contents

| | |
|---|-----------|
| About the Document..... | 3 |
| Contents | 4 |
| Table Index..... | 5 |
| Figure Index | 6 |
| 1 Introduction | 7 |
| 2 Software & Hardware..... | 8 |
| 2.1. Hardware Framework | 8 |
| 2.2. Software System Architecture | 9 |
| 2.3. QuecPython System Characteristics | 10 |
| 2.3.1. Introduction..... | 10 |
| 2.3.2. QuecPython SDK | 12 |
| 3 Resources..... | 15 |
| 3.1. Hardware Resource | 15 |
| 3.1.1. Hardware Diagram | 15 |
| 3.1.2. Processor | 15 |
| 3.1.3. Air Interface Resources..... | 16 |
| 3.1.3.1. WAN Air Interface Resources | 16 |
| 3.1.4. Storage Resources | 16 |
| 3.1.4.1. Command..... | 18 |
| uos.statvfs 18 | |
| 3.1.5. External Resources..... | 18 |
| 3.1.6. Supported Examples..... | 19 |
| 3.2. Software Resource..... | 26 |
| 3.2.1. Power-on Process..... | 26 |
| 3.2.2. Python Script Running Process | 27 |
| 3.2.3. QuecPython Built-in Module | 27 |
| 3.2.4. QuecPython Extension Class Library | 28 |
| 4 Development Guide | 30 |
| 4.1. Related Documents | 30 |
| 4.2. Development Environment Setup | 30 |
| 4.2.1. System Version | 30 |
| 4.2.2. BearPi EVB | 30 |
| 4.2.3. Installing the Driver | 31 |
| 4.2.4. Firmware Download | 32 |
| 4.3. Running the First QuecPython Program..... | 33 |
| 4.3.1. Downloading hello_world.py Program to EVB..... | 33 |
| 4.3.2. Executing hello_world.py Program | 34 |
| 4.3.2.1. Executing Manually | 34 |
| 4.3.2.2. Running Automatically | 35 |
| 5 Appendix A References..... | 36 |

Table Index

| | |
|---|----|
| Table 1: WAN Air Interface Resources | 16 |
| Table 2: Flash Partition | 17 |
| Table 3: Corresponding Pin Number of Peripheral Resources | 18 |
| Table 4: Machine Hardware Related Interface | 19 |
| Table 5: Power-on Time | 27 |
| Table 6: QuecPython Built-in Module | 27 |
| Table 7: Extension Class Library | 28 |
| Table 8: Related Documents..... | 36 |
| Table 9: Terms and Abbreviations | 36 |

Figure Index

| | |
|--|----|
| Figure 1: QuecPython Hardware Structure..... | 8 |
| Figure 2: QuecPython Software System Architecture | 9 |
| Figure 3: USB CDC Serial Port and Its Property | 11 |
| Figure 4: QuecPython SDK Directory Structure | 14 |
| Figure 5: QuecPython Hardware Diagram..... | 15 |
| Figure 6: Power-on Process..... | 26 |
| Figure 7: Python Script Running Process | 27 |
| Figure 8: BearPi EVB | 30 |
| Figure 9: Serial Ports..... | 31 |
| Figure 10: Confirming the Driver is Installed Successfully | 32 |
| Figure 11: Confirming the Driver is Installed Successfully | 32 |
| Figure 12: Firmware Downloading Interface..... | 33 |
| Figure 13: Firmware Script Interface..... | 34 |

1 Introduction

Python is a programming language that lets you work quickly and integrate systems more effectively. You can see it wherever it is a crawler or a cluster, but Python almost has no chance of embedded development. What if Python can be employed in the Internet of Things?

Quectel can now use Python for development. Quectel has ported the MicroPython virtual machine based on modules to realize the development and application of Python in the Internet of Things.

This document takes EC100Y-CN as an example to introduce the software and hardware resources and technology of EC100Y-CN and EC600S-CN QuecPython.

This document is applicable to the following Quectel modules:

- EC100Y-CN
- EC600S-CN

2 Software & Hardware

2.1. Hardware Framework

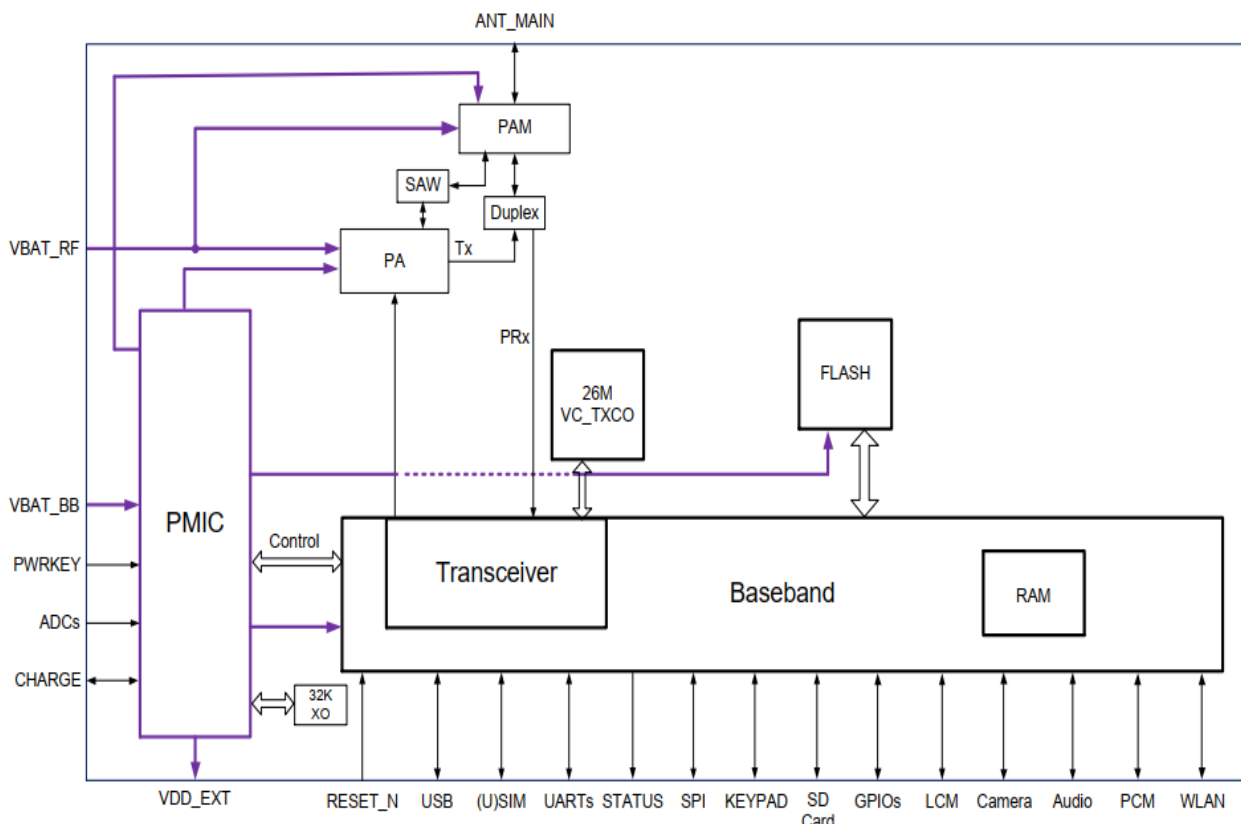


Figure 1: QuecPython Hardware Structure

2.2. Software System Architecture

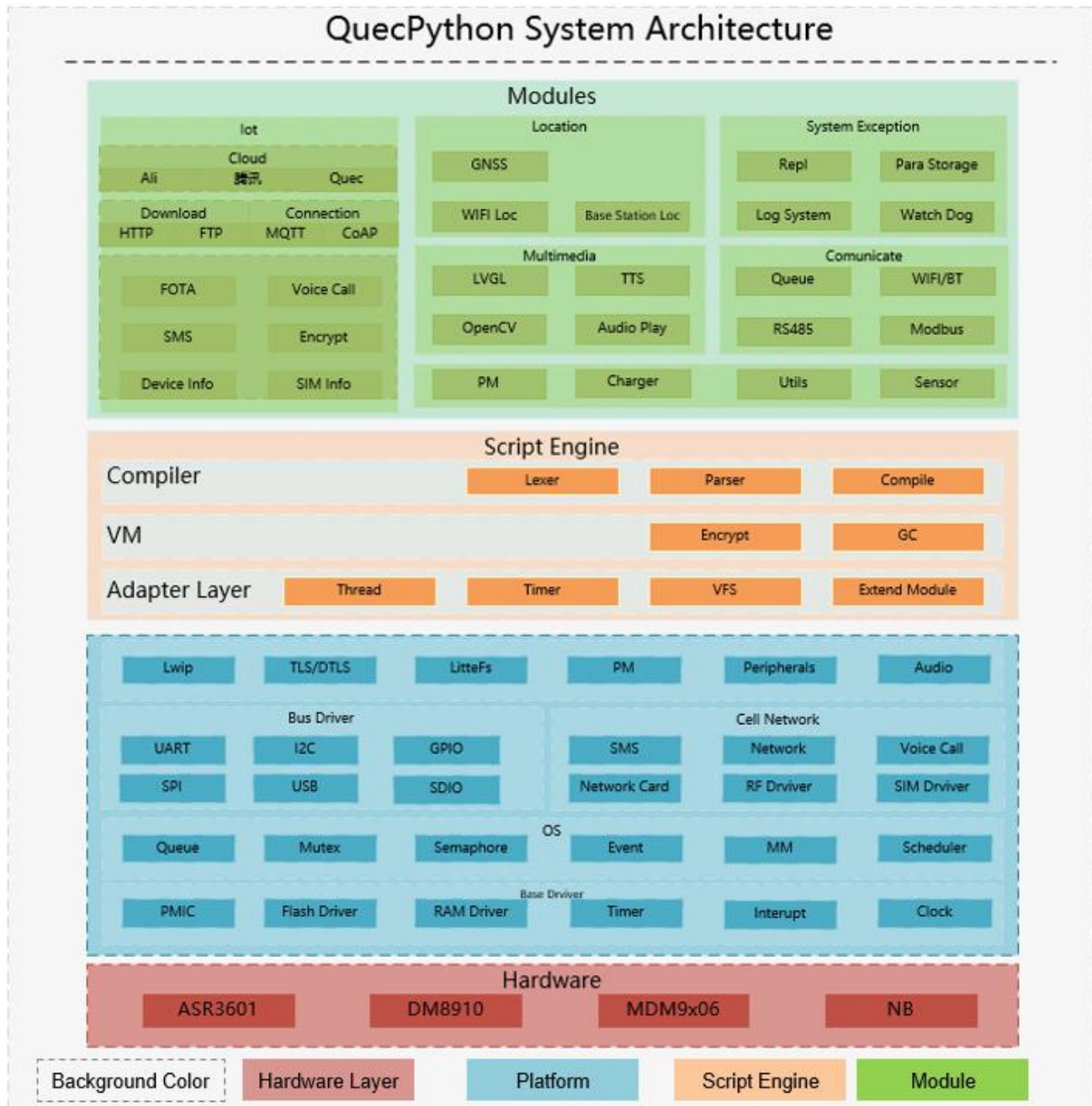


Figure 2: QuecPython Software System Architecture

NOTE

QuecPython does not currently support OpenCV, LVGL, Wi-Fi and Bluetooth functions.

2.3. QuecPython System Characteristics

2.3.1. Introduction

Quectel has transplanted MicroPython into the module. In addition to supporting the MicroPython function, it also introduces related functions of the communication module, such as data call, MQTT, HTTP, cloud, FOTA and other functions.

NOTE

MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments.

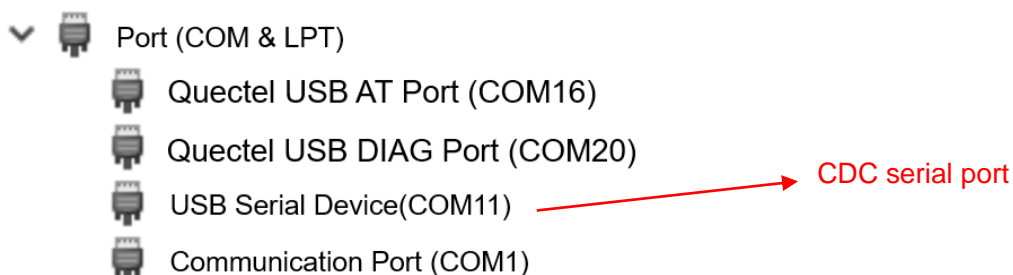
MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256 KB of code space and 16 KB of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

There are two ways to run QuecPython user code, that is REPL and script files:

- REP is an interactive way to talk to your computer in Python. After lexical and syntax analysis, compilation, the script code input by REPL generates bytecode, and the QuecPython virtual machine runs the script code.
- Script file: The code file is preprocessed into bytecode during the firmware generation stage and executed by the QuecPython virtual machine.

The above two methods can be developed and debugged through the USB CDC serial port:



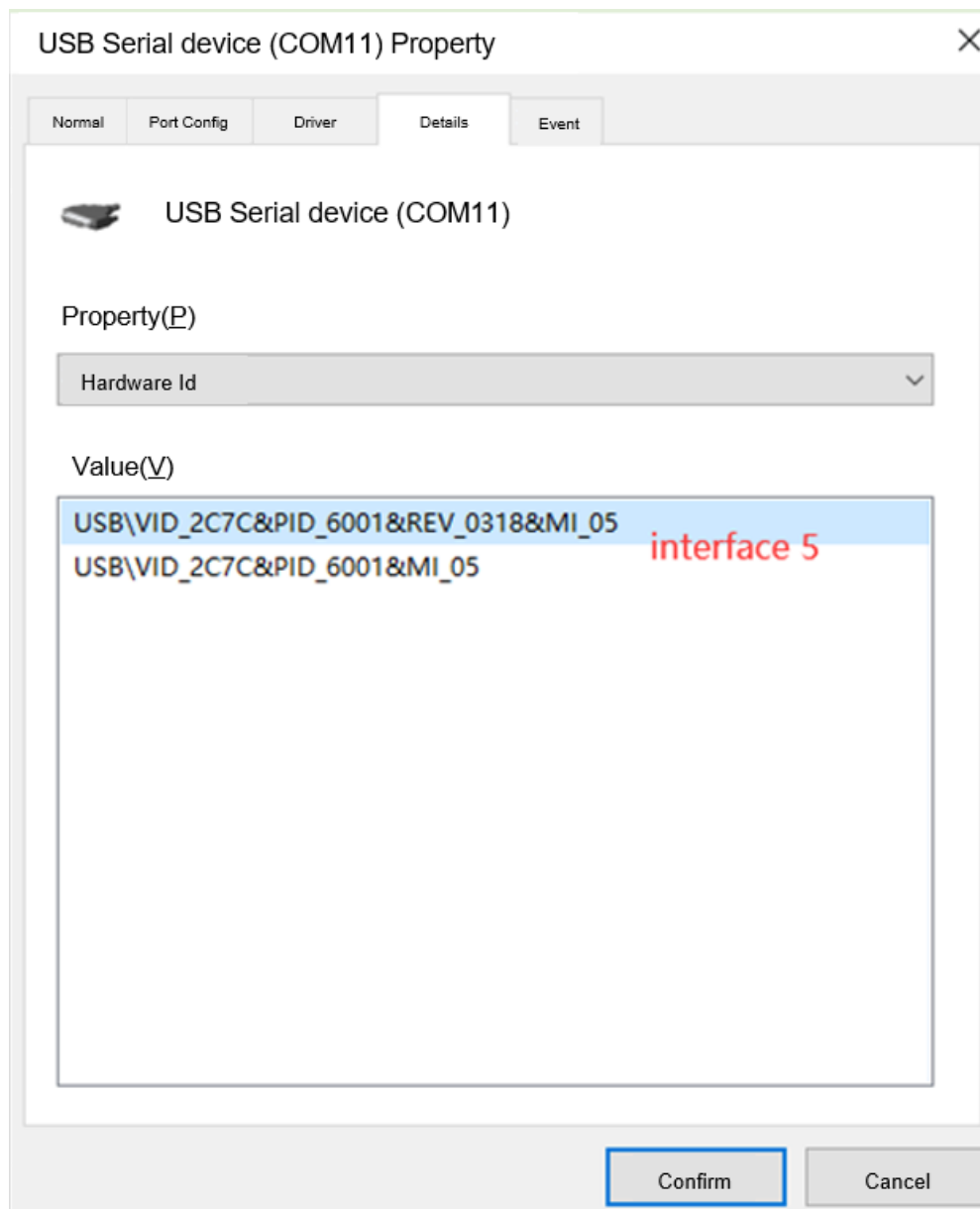


Figure 3: USB CDC Serial Port and Its Property

The serial port debugging interface is as follows:

```

QuecPython
Quecpython v1.12 on 2020-09-02: EC100Y with QUECTEL
Type "help()" for more information.
>>> uos.listdir()
['hello_world.py']
>>> import example
>>> example.exec('hello_world.py')
hello world!
>>> 

```

2.3.2. QuecPython SDK

QuecPython SDK includes platform-related documents, downloading tool, and various functional sample source codes. The directory structure is as follows:

```

README.md
boards
├── 小熊派-开发板
│   ├── Cat1_Core_RevA05.pdf
│   ├── Cat1_Core_RevB05_Gerber_生产文件.zip
│   └── Quectel_QuecPython_Cat1开发板使用说明_V1.3.pdf
cloud
├── ALiYun
│   └── aLiYun.py
├── QuecThing
└── TenCentYun
    └── TenCentYun.py
demo
├── README.md
├── aliyun
│   └── example_aliyun_file.py
├── common
│   ├── example_math_file.py
│   ├── example_random_file.py
│   ├── example_string_file.py
│   └── example_uos_file.py
├── dataCall
│   └── example_dataCall_callback_file.py
├── fota
│   └── example_fota_file.py
├── gpio
│   └── example_pin_file.py
├── http
│   ├── example_request_get_file.py
│   ├── example_request_post_file.py
│   └── example_request_ssl_file.py
├── I2C
│   └── example_i2c_file.py
├── json
│   └── example_json_file.py
├── log
│   ├── example_log_critical.py
│   ├── example_log_debug_file.py
│   ├── example_log_error_file.py
│   ├── example_log_info_file.py
│   └── example_log_warning_file.py
├── mqtt
│   └── example_mqtt_file.py
├── ntp
│   └── example_ntptime_file.py
├── pwm
│   └── example_pwm_file.py
├── socket
│   └── example_socket_file.py
├── TenCentyun
│   └── example_tencentyun_file.py
├── thread
│   └── example_thread_file.py
├── timer
│   └── example_timer_file.py
├── tts
│   └── example_tts_file.py
├── uart
│   └── example_uart_file.py
└── utime
    ├── example_utime_localtime_file.py
    ├── example_utime_mktime_file.py
    └── example_utime_sleep_file.py

```

```
├── document
│   ├── Quectel_QuecPython_类库API说明.md
│   ├── Quectel_QuecPython_QPYcom工具使用说明_V1.0.pdf
│   └── Quectel_QuecPython_Cat1开发板使用说明_V1.3.pdf
├── firmware
│   ├── EC100YCNAAR01A01M16_OCPU_PY.zip
│   └── README.md
└── tools
    ├── USB驱动 USB Driver
    │   └── Quectel_ASR_Series_UMTS&LTE_Windows_USB_Driver_Customer_V1.0.3.zip
    ├── 下载工具 Download Tool
    │   ├── QFlash_V4.20_CN.zip
    │   └── QLittlefs_tools.zip
    ├── 串口工具 Serial Tools
    │   └── QCOM_V1.6.zip
    └── 图形化工具 QPYcom GUI tool
        └── QPYcom_V1.0.1.zip
```

Figure 4: QuecPython SDK Directory Structure

For details of QuecPython SDK, see *Quectel_QuecPython_Class_Library_API_User_Guide*.

NOTE

QuecPython does not need to install any cross-compilation tool chain. Windows 10 only needs to install the Quectel USB driver (the driver is located in the *tools* directory) to start development. Windows 7 needs to install an additional USB CDC driver to solve the problems of installing the cross-compilation chain.

3 Resources

3.1. Hardware Resource

3.1.1. Hardware Diagram

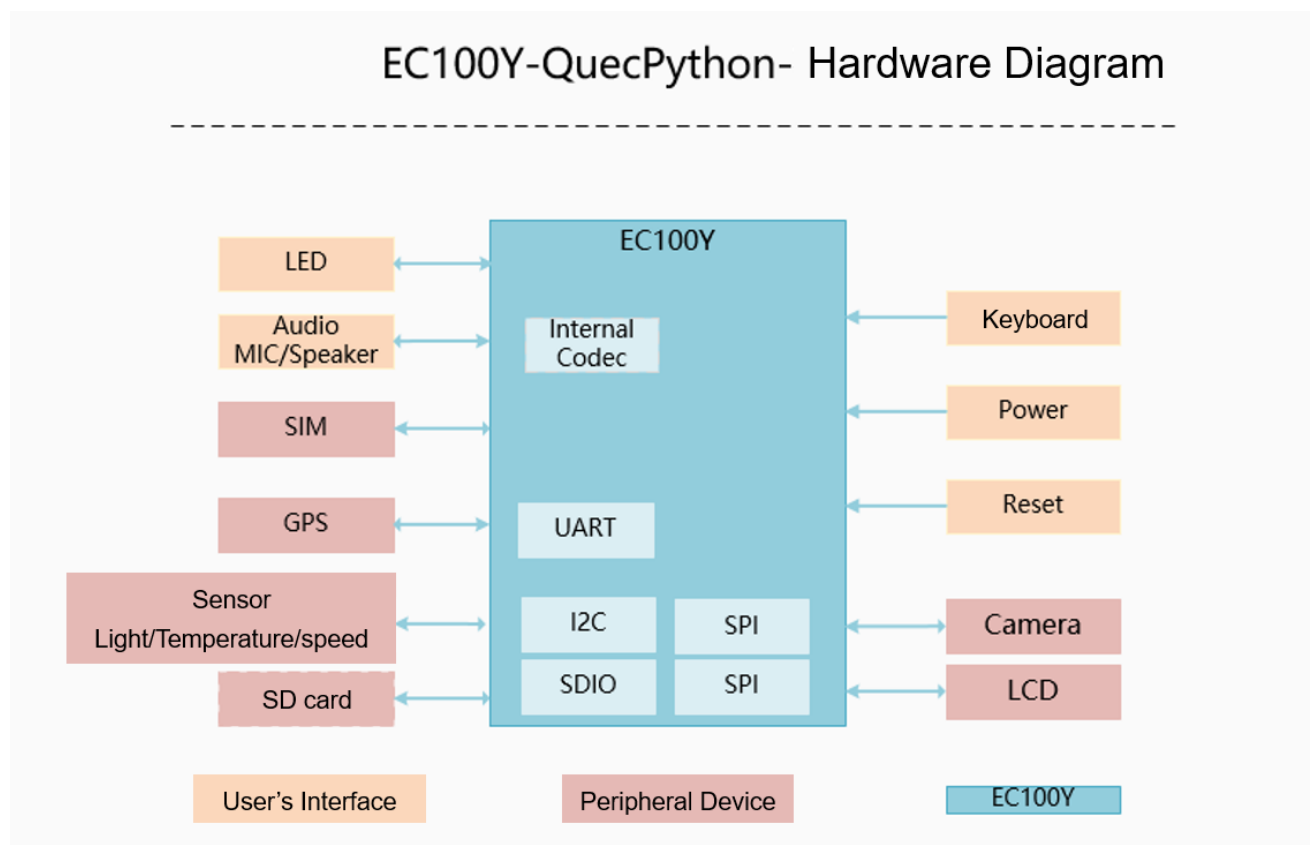


Figure 5: QuecPython Hardware Diagram

3.1.2. Processor

Application processor: ARM Cortex-R5 CPU 624 MHz with 32 KB L1 Instruction cache and 32KB L1 Data Cache

3.1.3. Air Interface Resources

3.1.3.1. WAN Air Interface Resources

Table 1: WAN Air Interface Resources

| Abbreviation | Description |
|--|---|
| <ul style="list-style-type: none">● GSM/GRPS/EDGE Class 12● CAT-M1 R13 DL 1Mbps/ UL 1Mbps | <ul style="list-style-type: none">● DL 10Mbps/ UL 5Mbps● LTE● VoLTE supported |

3.1.4. Storage Resources

With 16 MB Nor Flash and 16 MB RAM, the available space of EC100Y-CN is as follows:

- RAM: about 3 MB
- Flash: about 4 MB

The available flash space can be obtained through the following command:

```
>>> uos.statvfs("main.py")
(4096, 4096, 1280, 1274, 1274, 0, 0, 0, 0, 255)
>>>
>>> █
```

The remaining memory of Python can be obtained by the following command:

```
>>> gc.mem_free()
247904
>>> █
```

The remaining heap memory of the system can be obtained through AT commands.

NOTE

The available flash space, heap and GC space of the system are based on the actual return value.

See the table below for details of the flash partition. As the partition table may be adjusted, see the partition information in the firmware package.

Table 2: Flash Partition

| Partition | Size (K/M) | Description |
|---------------------|-------------|---|
| External | 16 M | / |
| Bootloader | 144 K | Start the firmware |
| system | About 8 M | All the system image |
| ptable | 4 K | Partition table information |
| fwcerts | 12 K | Network authentication |
| Rd | 64 K | / |
| Apn | 32 K | / |
| cp | About 5 M | Kernel firmware area; adaptive image size |
| Dsp | About 1.5 M | / |
| rabin | 20 K | / |
| logo | 200 K | / |
| reserved | / | Reserved; adaptive image size |
| customer_app | About 1.5 M | Customer application |
| customer_fs | 4 M | Customer file system firmware area |
| customer_backup_fs | 200 K | Backup file system |
| backup_restore_info | 4 K | / |
| fota_param | 12 K | FOTA upgrade parameter area |
| fota_pkg | 2.3 M | FOTA upgrade storage backup area |
| updater | 128 K | Upgrade firmware area; kernel |
| nvm | 512 K | / |
| erase_rd | 64 K | / |
| quec_cfg | 64 K | / |
| factory | 128 K | / |

3.1.4.1. Command

For the detailed usage of `uos.statvfs` and `gc.mem_free`, please refer to *Quectel_QuecPython_Class_Library_API_User_Guide*.

`uos.statvfs`

`uos.statvfs` obtains the status information of the file system. The return value format is: (`f_bsize`, `f_frsize`, `f_blocks`, `f_bfree`, `f_bavail`, `f_files`, `f_ffree`, `f_favail`, `f_flag`, `f_namemax`).

● Parameter

| Parameter Name | Description |
|------------------------|---|
| <code>f_bsize</code> | The size of the file system block. Unit: Byte |
| <code>f_frsize</code> | Sub-stack. Unit: Byte |
| <code>f_blocks</code> | Total number of file system data blocks |
| <code>f_bfree</code> | Available blocks |
| <code>f_bavai</code> | Number of blocks available to non-super users |
| <code>f_files</code> | Total number of file nodes |
| <code>f_ffree</code> | Available file nodes |
| <code>f_favail</code> | Number of blocks available to super users |
| <code>f_flag</code> | Mount flag |
| <code>f_namemax</code> | Maximum file length. Unit: Byte |

3.1.5. External Resources

For details on using peripheral resources, see *Quectel_EC100Y-CN_QuecOpen_Hardware_Design*.

Table 3: Corresponding Pin Number of Peripheral Resources

| Type | Array | Pin Number |
|------|-------|----------------|
| UIM | 1 | 57, 58, 59, 60 |

| | | |
|-------|--------------------------------|------------------------|
| USB | 1; USB 2.0 supports slave mode | 10, 11, 12 |
| SDIO | 1 | 29, 30, 31, 32, 33, 34 |
| UART | 1 | 20, 21 |
| SPI | 1 | 25, 26, 27, 28 |
| I2C | 1 | 55, 56 |
| GPIO | 5 | 22, 23, 178, 199, 204 |
| PWM | 1 | 16, 17, 18, 19 |
| ADC | 2 | 39, 81 |
| LCD | 1 | 21, 22, 23, 24, 25, 26 |
| Audio | 1 | 2, 3, 4, 41, 42, 82 |

3.1.6. Supported Examples

Table 4: Machine Hardware Related Interface

| Class | Functionality | Class Method | Example |
|--------------------|---------------|---|---|
| Hardware Interface | PIN | GPIO read and write operation <i>Pin.read()</i> <i>Pin.write(value)</i> | demo/gpio/example_pin_file.py |
| | UART | Data transmission of UART serial port <i>uart.any()</i> <i>uart.read(nbytes)</i> <i>uart.write(data)</i> <i>uart.close()</i> | demo/uart/example_uart_file.py |
| | Timer | Timer <i>timer.start(period, callback)</i> <i>timer.stop()</i> | <i>mode,</i> demo/timer/example_timer_file.py |
| | ExtInt | Configures I/O pins to interrupt when an external event occurs <i>extint.enable()</i> <i>extint.disable()</i> <i>extint.line()</i> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=extint |
| | RTC | Provides a method to get and set RTC time <i>rtc.datetime([year, month, day, week, hour, minute, second, microsecond])</i> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=rtc |

| | | | | |
|------------------|---------|---|--|---|
| | WDT | Watchdog (Software Watchdog) | <code>wdt.feed()</code> <code>wdt.stop()</code> | demo/wdt/example_wdt_file.py |
| | IIC | Two-wire protocol for communication between devices | <code>I2C.read(slaveaddress, addr, r_data, datalen, delay)</code> <code>I2C.write(slaveaddress, addr, data, datalen)</code> | demo/I2C/example_i2c_file.py |
| UIO | | Contains other types of stream (class file) objects and auxiliary functions | <code>fd = uio.open(name, mode='r', **kwargs)</code> <code>fd.close()</code> | demo/common/example_uos_file.py |
| Data Call | | Data call | <code>dataCall.start(profileIdx, ipType, apn, username, password, authType)</code> <code>dataCall.setApn(profileIdx, ipType, apn, username, password, authType)</code> <code>dataCall.setCallback(callback)</code> <code>dataCall.getInfo(profileIdx, ipType)</code> | demo/dataCall/example_dataCall_callback_file.py |
| Base positioning | station | Get coordinate information | <code>cellLocator.getLocation(serverAddr, port, token, timeout, profileID)</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=celllocator-基站定位 |
| Audio | | Audio playback, which supports TTS, mp3 and AMR file playback | <code>tts.close()</code> <code>tts.play(priority, breakin, mode, str)</code> <code>tts.getVolume()</code> <code>tts.setVolume(vol)</code> <code>tts.getSpeed()</code> <code>tts.setSpeed(speed)</code> <code>tts.getState()</code> <code>tts.stop()</code> <code>aud.play(priority, breakin, filename)</code> <code>aud.stop()</code> <code>aud.getState()</code> <code>aud.getVolume()</code> <code>aud.setVolume(vol)</code> | demo/tts/example_tts_file.py |
| SIM 卡 | | SIM card operations, | <code>sim.getImsi()</code> <code>sim.getIccid()</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=sim-si |

| | | | |
|------|---|---|---|
| | such as querying SIM card status, ICCID, IMSI, etc. | <code>sim.getPhoneNumber()</code> <code>sim.getStatus()</code> <code>sim.enablePin(pin)</code> <code>sim.disablePin(pin)</code> <code>sim.verifyPin(pin)</code> <code>sim.unblockPin(puk, newPin)</code> <code>sim.changePin(oldPin, newPin)</code> <code>sim.readPhonebook(storage, start, end, username)</code> <code>sim.writePhonebook(storage, index, username, number)</code> | <u>m 卡</u> |
| NET | Configure and query network mode information | <code>net.csqQueryPoll()</code> <code>net.getCellInfo()</code> <code>net.getConfig()</code> <code>net.getNetMode()</code> <code>net.getSignal()</code> <code>net.nitzTime()</code> <code>net.operatorName()</code> <code>net.getState()</code> <code>net.getCi()</code> <code>net.getMnc()</code> <code>net.getMcc()</code> <code>net.getLac()</code> <code>net.getModemFun()</code> <code>net.setModemFun(function, rst)</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=net-网络相关功能 |
| FOTA | Firmware upgrade | <code>fota.write(bytesData)</code> <code>fota.flush()</code> <code>fota.verify()</code> | <code>demo/fota/example_fota_file.py</code> |
| UOS | Contains file system access and mounting construction, and implements a subset of the corresponding modules of the CPython module | <code>uos.remove(path)</code> <code>uos.chdir(path)</code> <code>uos.getcwd()</code> <code>uos.listdir([dir])</code> <code>uos.mkdir(path)</code> <code>uos.rename(old_path, new_path)</code> <code>uos.rmdir(path)</code> <code>uos.ilistdir([dir])</code> <code>uos.stat(path)</code> <code>uos.statvfs(path)</code> <code>uos.uname()</code> <code>uos.urandom(n)</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=uos-基本系统服务 |
| gc | Implements the memory garbage collection | <code>gc.enable()</code> <code>gc.disable()</code> <code>gc.collect()</code> <code>gc.mem_alloc()</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=gc-内存碎片回收 |

| | | | |
|---------|--|--|------------------------------------|
| | mechanism, and a subset of the corresponding modules of the CPython module | <i>gc.mem_free()</i> | |
| socket | Provides access to the BSD socket interface | <i>usocket.getaddrinfo(host, port)</i> <i>socket.bind(address)</i> <i>socket.listen(backlog)</i> <i>socket.accept()</i> <i>socket.connect(address)</i> <i>socket.read([size])</i> <i>socket.readinto(buf, [, nbytes])</i> <i>socket.readline()</i> <i>socket.write(buf)</i> <i>socket.send(bytes)</i> <i>socket.sendall(bytes)</i> <i>socket.sendto(bytes, address)</i> <i>socket.recv(bufsize)</i> <i>socket.close()</i> <i>socket.setsockopt(level, optname, value)</i> <i>socket.setblocking(flag)</i> <i>socket.settimeout(value)</i> <i>socket.makefile(mode='rb')</i> | demo/socket/example_socket_file.py |
| thread | Provides methods to create a new thread, and mutex locks | <i>_thread.get_ident()</i> <i>_thread.stack_size()</i> <i>_thread.start_new_thread(function, n, args)</i> <i>_thread.allocate_lock()</i> <i>lock.acquire()</i> <i>lock.release()</i> <i>lock.locked()</i> | demo/thread/example_thread_file.py |
| urandom | Provides tools to generate random numbers | <i>urandom.choice(obj)</i> <i>urandom.getrandbits(k)</i> <i>urandom.randint(start, end)</i> <i>urandom.random()</i> <i>urandom.randrange(start, end, step)</i> <i>urandom.seed(sed)</i> <i>urandom.uniform(start, end)</i> | demo/common/example_random_file.py |
| math | Provides mathematical | <i>math.pow(x, y)</i> <i>math.acos(x)</i> | demo/common/example_math_file.py |

| | | | |
|--------|--|---|---|
| | operation functions | $math.asin(x)$ $math.atan(x)$ $math.atan2(x, y)$ $math.ceil(x)$ $math.copysign(x, y)$ $math.cos(x)$ $math.degrees(x)$ $math.e$ $math.exp(x)$ $math.fabs(x)$ $math.floor(x)$ $math.fmod(x, y)$ $math.modf(x)$ $math.frexp(x)$ $math.isfinite(x)$ $math.isinf(x)$ $math.isnan(x)$ $math.ldexp(x, exp)$ $math.log(x)$ $math.pi$ $math.radians(x)$ $math.sin(x)$ $math.sqrt(x)$ $math.tan(x)$ $math.trunc(x)$ | |
| utime | Gets current time and date, measurement interval and delay | $utime.localtime([secs])$ $utime.mktime(date)$ $utime.sleep(seconds)$ $utime.sleep_ms(ms)$ $utime.sleep_us(us)$ $utime.ticks_ms()$ $utime.ticks_us()$ $utime.ticks_cpu()$ $utime.ticks_diff(old, new)$ $utime.time()$ | demo/utime/example_utime_localtime_file.py demo/utime/example_utime_mktime_file.py demo/utime/example_utime_sleep_file.py |
| aLiYun | Alibaba Cloud IoT suite client function. Currently the product node only supports "device", and the device | $aLiYun(productKey, productSecret, DeviceName, DeviceSecret)$ $aLiYun.setMqtt(clientID, clean_session, keepAlive)$ $aLiYun.setCallback(sub_cb)$ $aLiYun.subscribe(topic, qos)$ $aLiYun.publish(topic, msg)$ $aLiYun.start()$ | demo/alijun/example_alijun_file.py |

| | | | |
|---------|---|--|--|
| | authentication method supports "unique-certific ate-per-device" and "unique-certific ate-per-product " | | |
| TXyun | Tencent Cloud IoT suite client function. Currently the product node type only supports "device", and the device authentication method supports "unique-certific ate-per-device" and "unique-certific ate-per-product " | TXyun(productID, devicename, devicePsk, ProductSecret) TXyun.setMqtt(clean_session, keepAlive) TXyun.setCallback(sub_cb) TXyun.subscribe(topic,qos) TXyun.publish(topic,msg) TXyun.start() | demo/TenCentyun/exam ple_tencentyun_file.py |
| HTTP 服务 | HTTP client related functions | request.get(url, data, json, headers) request.post(url, data, json, headers) request.put(url, data, json, headers) request.head(url, data, json, headers) request.patch(url, data, json, headers) request.delete(url, data, json, headers) response=request.get(url) response.content () response.text () response.json() response.close() | demo/http/example_requ est_get_file.py demo/http/example_requ est_post_file.py demo/http/example_requ est_ssl_file.py |

| | | | |
|------------|--|---|--|
| Log | System log, log level tool | <code>log.basicConfig(level)</code> <code>log.getLogger(name)</code> <code>log.debug(tag, msg)</code> <code>log.info(tag,msg)</code> <code>log.warning(tag,msg)</code> <code>log.error(tag,msg)</code> <code>log.critical(tag,msg)</code> | demo/log/example_log_critical_file.py demo/log/example_log_debug_file.py demo/log/example_log_error_file.py demo/log/example_log_info_file.py demo/log/example_log_warning_file.py |
| Umqtt 服务 | Provides the function of creating MQTT client to publish and subscribe | <code>MQTTClient(client_id, server, port=0, user=None, password=None, keepalive=0, ssl=False, ssl_params={})</code> <code>MQTTClient.set_callback(callback)</code> <code>MQTTClient.set_last_will(topic, msg, retain=False, qos=0)</code> <code>MQTTClient.connect(clean_session=True)</code> <code>MQTTClient.disconnect()</code> <code>MQTTClient.ping()</code> <code>MQTTClient.publish(topic, msg)</code> <code>MQTTClient.subscribe(topic, qos)</code> <code>MQTTClient.check_msg()</code> <code>MQTTClient.wait_msg()</code> | demo/mqtt/example_mqtt_file.py |
| NtpTime | Time synchronization | <code>ntptime.host</code> <code>ntptime.sethost(host)</code> <code>ntptime.settime()</code> | demo/ntp/example_ntptime_file.py |
| sys | Provides functions and variables related to the QuecPython operating environment | <code>sys.argv</code> <code>sys.byteorder</code> <code>sys.implementation</code> <code>sys.maxsize</code> <code>sys.modules</code> <code>sys.platform</code> <code>sys.stdin</code> <code>sys.stdout</code> <code>sys.version</code> <code>sys.version_info</code> <code>sys.exit(retval=0)</code> <code>sys.print_exception(exc, file=sys.stdout)</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=sys-系统相关功能 |
| ujson-Json | Realizes the function of | <code>ujson.dump(obj, stream)</code> <code>ujson.dumps(dict)</code> | demo/json/example_json_file.py |

| | | | | |
|--------------|--------------|---|--|---|
| | | converting between Python objects and JSON data formats | <code>ujson.load(stream)</code> <code>ujson.loads(str)</code> | |
| ustruct | | Compress and decompress raw data types | <code>ustruct.calcsize(fmt)</code> <code>ustruct.pack(fmt, v1, v2, ...)</code> <code>ustruct.unpack(fmt, data)</code> <code>ustruct.pack_info(fmt, buffer, offset, v1, v2, ...)</code> <code>ustruct.unpack_from(fmt, data, offset=0)</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=ustruct -打包和解压原始数据类型 |
| | Power module | Provides shutdown, software restart | <code>Power.powerDown()</code> <code>Power.powerRestart()</code> <code>Power.powerOnReason()</code> <code>Power.powerDownReason()</code> <code>Power.getVbatt()</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=power |
| misc | PWM module | PWM | <code>pwm = PWM(PWM.PWMn, highTime, cycleTime)</code> <code>pwm.open()</code> <code>pwm.close()</code> | demo/pwm/example_pwm_file.py |
| | ADC module | ADC | <code>adc.open()</code> <code>adc.read(ADCn)</code> <code>adc.close()</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=adc |
| Modem device | | Gets equipment information | <code>modem.getDevImei()</code> <code>modem.getDevModel()</code> <code>modem.getDevSN()</code> <code>modem.getDevFwVersion()</code> <code>modem.getDevProductId()</code> | http://qpy.quectel.com/wiki/#/zh-cn/api/?id=modem -设备相关 |

3.2. Software Resource

3.2.1. Power-on Process

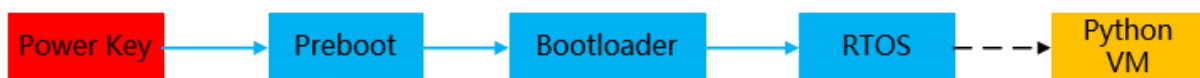


Figure 6: Power-on Process

Table 5: Power-on Time

| Process | Power-on Time | Main Functionality |
|------------|---------------|--|
| Preboot | < 0.1 second | Detect the power key Check whether to enter the download mode |
| Bootloader | < 4 second | Check whether to enter the OTA upgrade (not currently supported) |
| RTOS | < 6 second | Start system services Register to the network and start a data call |
| Python VM | < 7 second | Read and run Python scripts from the file system |

3.2.2. Python Script Running Process

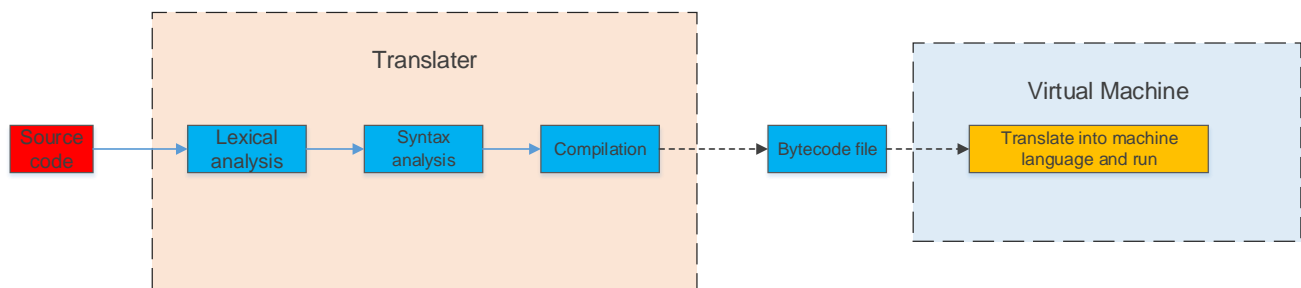


Figure 7: Python Script Running Process

After running the system, QuecPython runs as a background thread of RTOS, to read and execute Python scripts from the file system.

3.2.3. QuecPython Built-in Module

Table 6: QuecPython Built-in Module

| Built-in Module | Description |
|-----------------|--------------------------------|
| uos | Basic system service |
| gc | Memory fragment recovery |
| ubinasii | Binary and ASCII conversion |
| ucollections | Collection and container types |
| urandom | Generate random numbers |

| | |
|---------|--|
| math | Computation |
| usocket | Socket module |
| uio | Input and output stream |
| ustruct | Compress and decompress raw data types |
| ujson | JSON encoding and decoding |
| utime | Time-related functions |
| sys | System related functions |
| uzlib | unzip zlib |
| _thread | Multithreading |

3.2.4. QuecPython Extension Class Library

Table 7: Extension Class Library

| Module Name | Description |
|-------------|---|
| example | Run Python script |
| dataCall | Data call |
| cellLocator | Base station positioning |
| sim | SIM card related |
| net | Network-related functions |
| fota | Firmware upgrade |
| audio | TTS and audio file playback |
| misc | Restart, shutdown, PWM, ADC related functions |
| modem | Device related |
| machine | Hardware related functions |
| aLiYun | Alibaba Cloud Service |
| TenCentYun | Tencent Cloud Service |

| | |
|---------|--------------------------|
| request | HTTP |
| log | Log |
| umqtt | MQTT |
| ntptime | NTP time synchronization |
| pm | Low power consumption |
| ure | Regular |

4 Development Guide

4.1. Related Documents

- Hardware: *Quectel_EC100Y-CN_QuecOpen_Hardware_Design*
- Software: *Quectel_QuecPython_CAT1_EVB_User_Guide*

4.2. Development Environment Setup

4.2.1. System Version

The system version requires Windows 7 or higher.

4.2.2. BearPi EVB

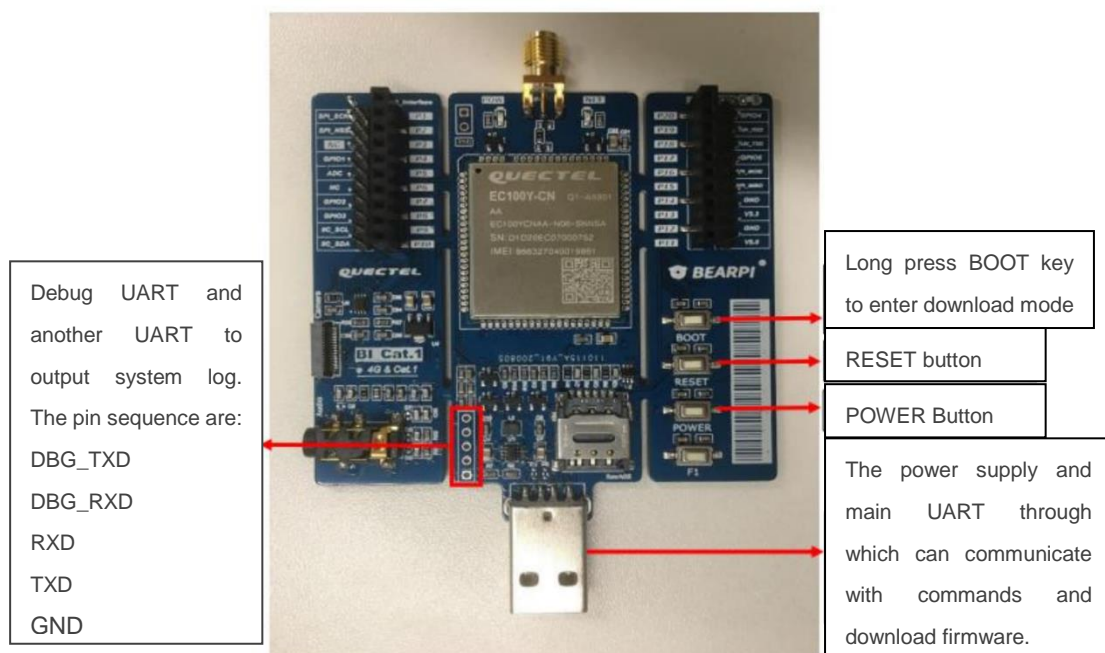



Figure 8: BearPi EVB

DBG_TXD and DBG_RXD are ports for outputting debug information. When using the ports, you need to connect DBG_TXD, DBG_RXD and GND to the USB to TTL converter, and connect to PC's serial port through the converter.

4.2.3. Installing the Driver

In the software tools, find the following driver installation package and decompress it. Double-click *setup.exe* to install the driver directly, and click **"Finish"** to complete the installation.

 Quectel_ASR_Series_UMTS<E_Windows_USB_Driver_Customer_V1.0.3.zip

After the driver is installed, connect the EVB to the PC, and then enter the device manager of the PC. After that, click **"Ports (COM and LPT)"** to view the following three ports (port number may be different, but the names are the same):

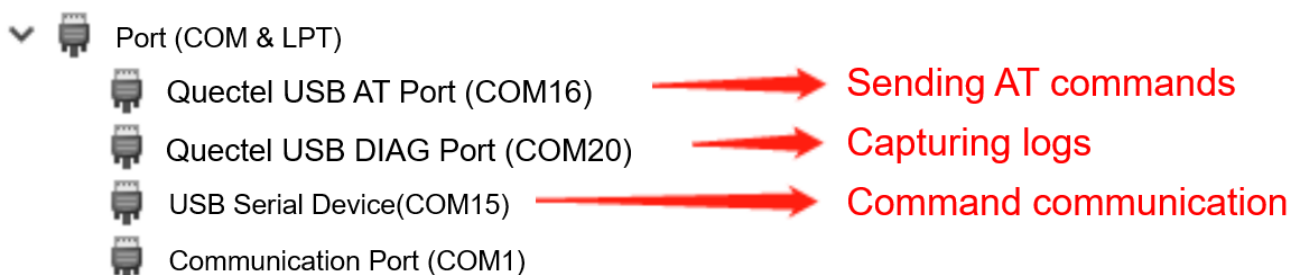


Figure 9: Serial Ports

Then, confirm whether the driver is installed successfully. Open the QCOM software, if you can send AT command and it can respond successfully, it means the driver is installed successfully. Note to select the port number corresponding to "Quectel USB AT Port" with corresponding baud rate of 115200, 1 bit of stop bit, no parity, 8 data bits, and no hardware control flow. The detailed information is shown in the figure below:

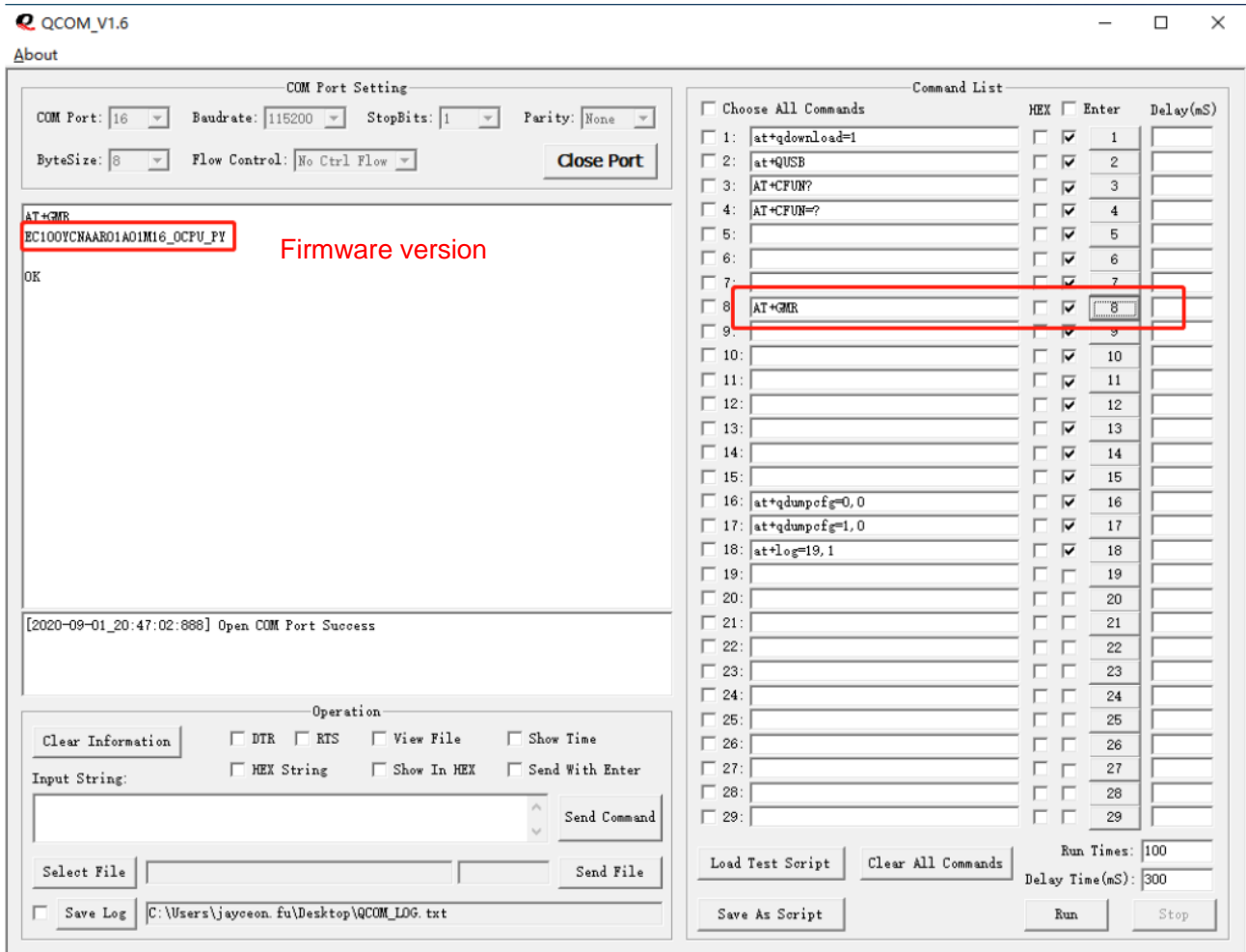


Figure 10: Confirming the Driver is Installed Successfully

4.2.4. Firmware Download

Send **AT+GMR** through the QCOM to view the firmware version with the return value:



Figure 11: Confirming the Driver is Installed Successfully

If the firmware version returned by **AT+GMR** ends with "PY", it means that the firmware is a python version, you can skip the firmware download process directly

If the firmware version returned by **AT+GMR** does not end with "PY", you need to download the firmware:

- Step 1:** In the software tools provided, find the compressed package of QPYcom to decompress it, and then double-click to run the software;
- Step 2:** Create a user project according to specific requirements (by clicking "**Creation**" button);
- Step 3:** Select the firmware package to be downloaded to the module;
- Step 4:** Click "▼" button to switch to "Download FW";
- Step 5:** Click "Download FW" button;
- Step 6:** When the progress bar reaches 100%, it indicates that the download is complete.

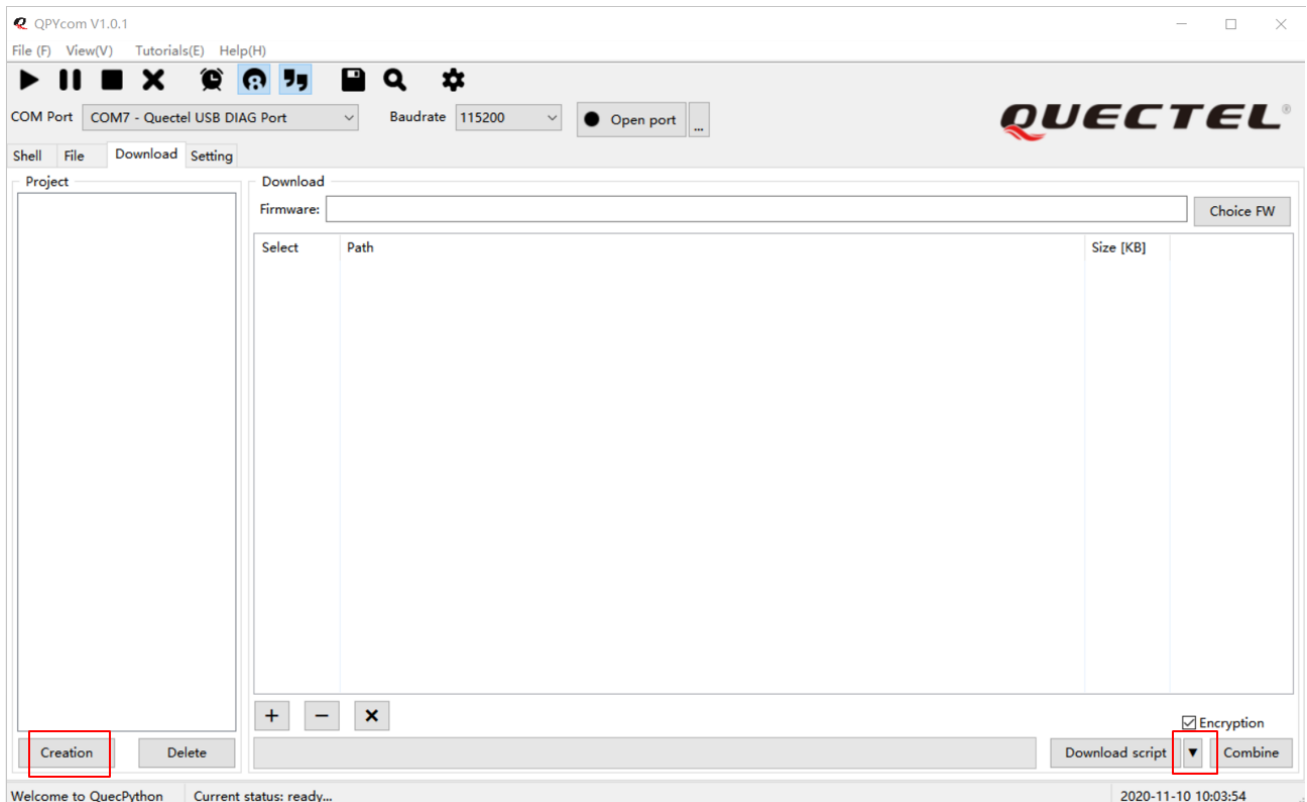



Figure 12: Firmware Downloading Interface

4.3. Running the First QuecPython Program

4.3.1. Downloading hello_world.py Program to EVB

Enter the *tools* directory in the SDK toolkit, find the QPYcom file package and decompress it. Then enter the decompressed QPYcom directory to find the *QPYcom.exe* tool, through which the python script file can be sent to EC100Y-CN.

- Step 1:** Create a user project according to specific requirements (by clicking "**Creation**" button);
- Step 2:** Select the *hello_world.py* script;

- Step 3:** Click “” button to switch to “Download script”;
- Step 4:** Click “Download script” button;
- Step 5:** When the progress bar reaches 100%, it indicates that the download is complete. Click “File” on the left to view the file details in the module.

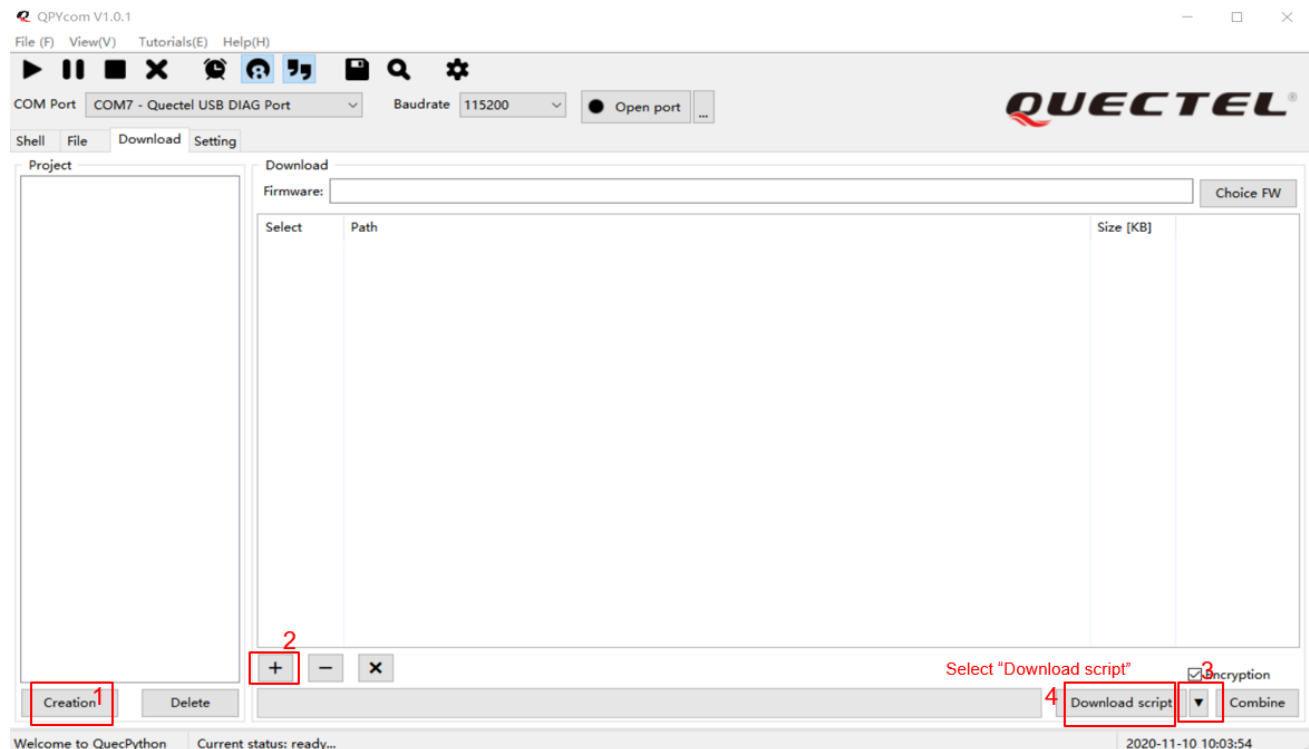


Figure 13: Firmware Script Interface

4.3.2. Executing hello_world.py Program

4.3.2.1. Executing Manually

In QPYcom, use main UART of EC100Y-CN to enter the communication interface. First confirm that user's program file is in the current directory through `uos.listdir()`, and then perform the following steps:

- Step 1:** Import the example module, which provides the `exec()` method to execute python script program;
- Step 2:** Execute hello_world.py script through the command `example.exec('hello_world.py')`.

The result is as follows:

```
QuecPython v1.12 on 2020-09-02; EC100Y with QUECTEL
Type "help()" for more information.
>>> uos.listdir()
['hello_world.py']
>>> import example
>>> example.exec('hello_world.py')
hello world!
>>> █
```

4.3.2.2. Running Automatically

EC100Y-CN supports automatic execution of user code. After the module powers on and runs, the system finds the program file named *main.py* and executes it automatically. Therefore, if you need to run the code automatically after module power-on, name the program by *main.py*. If *main.py* calls any other methods from the source file, you need to download the file into the module together. Here takes *hello_world.py* as an example.

hello_world.py file provides a method to print the "hello world!" string periodically in 2 seconds; the *main.py* file calls the methods in *hello_world.py*. Download both files to the module, and manually press the RESET button of the EVB. After the system starts, reconnect the main UART, press Enter on the keyboard, and enter the communication interface to see the automatic operation results.

```
Connecting to COM25...
Connected.

hello world!
hello world!
hello world!
hello world!
hello world!
█
```

NOTE

When the system restarts, you must disconnect the main UART, and then reconnect, otherwise you will not see the test result.

5 Appendix A References

Table 8: Related Documents

| SN | Document Name | Remark |
|-----|---|---|
| [1] | Quectel_QuecPython_Class_Library_API_User_Guide | QuecPython class library API user guide |
| [2] | Quectel_EC100Y-CN_QuecOpen_Hardware_Design | EC100Y-CN QuecOpen hardware design |
| [3] | Quectel_QuecPython_CAT1_EVB_User_Guide | QuecPython CAT1 EVB user guide |

Table 9: Terms and Abbreviations

| Abbreviation | Description |
|--------------|--|
| ADC | Analog-to-Digital Converter |
| API | Application Programming Interface |
| CPU | Central Processing Unit |
| DL | Downlink |
| EVB | Evaluation Board |
| FOTA | Firmware Upgrade Over-The-Air |
| HTTP | Hypertext Transfer Protocol |
| LTE | (Long-Term Evolution) a 4G mobile communications standard. |
| MQTT | Message Queuing Telemetry Transport |
| NTP | Network Time Protocol |
| OTA | Upgrade Over-The-Air |
| PWM | Pulse Width Modulation |

| | |
|-------|--|
| RAM | Random Access Memory |
| RTOS | Real-Time Operating System |
| SDK | Software Development Kit |
| SIM | Subscriber Identity Module |
| TTS | Text To Speech |
| UL | Uplink |
| USB | Universal Serial Bus |
| VoLTE | Voice (voice calls) over LTE. A standard high-speed wireless communication for mobile phones and data terminals, including Internet of things devices and wearables. |
| WAN | Wide Area Network |
| Wi-Fi | Wireless Fidelity |
