QuecPython Multithreading User Guide

LTE Standard Module Series

Version: 1.0.0

Date: 2020-11-10

Status: Preliminary

www.quectel.com

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

About the Document

Revision History

Version	Date	Author	Description
-	2020-11-10	Kinney/Kingka	Initial
1.0.0	2020-11-10	Kinney/Kingka	Initial

Contents

Table Index

Figure Index

1 Introduction

This document takes EC100Y-CN as an example to show how develop multithreading project with QuecPython class library APIs.

This document is applicable to the following Quectel modules:

- EC100Y-CN
- EC600S-CN

2 Multithreading Introduction

2.1. Basic Concept

Python runs in the Python virtual machine, and the multiple threads created by the user are the virtual threads in the Python virtual machine, not real threads in the operating system. In other words, multithreading in Python is polled and scheduled by the Python virtual machine, not the operating system.

Multithreading is a type of execution model that allows multiple threads to exist within the context of a process such that they execute independently but share their process resources. Threads are different from processes in the execution process. In each independent thread, there is an entry for program operation, a sequential execution sequence, and an exit for the program. And the thread must be attached to a program, and the program controls the operation of multiple threads.

2.2. Basic Operation Process

Thread has 5 states, the process of state transition is as follows:

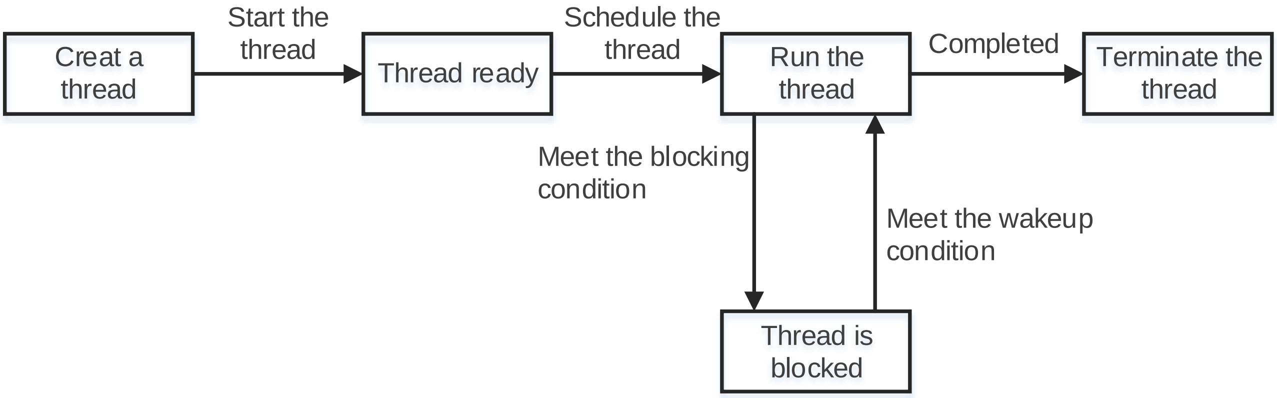


Figure 1: Five States of Threads

2.3. Main Difference between Threads and Processes

Threads and processes are the basic units of the operating system to control the program. The system achieves high concurrency for the program based on them. The main differences between threads and processes are as follows:

1. A program has at least one process; a process contains at least one thread.
2. The process has independent storage space in memory, and multiple threads share the storage space of the process it depends on.
3. Processes and threads have different ways of operating system resources management.

3 Alibaba Cloud related APIs

3.1. _thread.allocate_lock

This function allocates a mutex object.

- **Prototype**

```
_thread.allocate_lock()
```

- **Parameter**

None

- **Return Value**

Return the mutex object. The functions of the mutex object is detailed in ***Chapter 3.1.1***.

3.1.1. Mutex Object Function

3.1.1.1. lock.acquire

This function obtains the clock.

- **Prototype**

```
lock.acquire()
```

- **Parameter**

None

- **Return Value**

True	Succeed
False	Fail

3.1.1.2. lock.release

This function releases the clock.

- **Prototype**

```
lock.release()
```

- **Parameter**

None

- **Return Value**

None

3.1.1.3. lock.locked

This function returns the lock status.

- **Prototype**

```
lock.locked()
```

- **Parameter**

None

- **Return Value**

True	It is obtained by a certain thread.
False	It does not obtained by the thread.

3.2. _thread.get_ident

This function obtains the current thread ID.

- **Prototype**

```
_thread.get_ident()
```

- **Parameter**

None

- **Return Value**

Return current thread ID.

3.3. _thread.stack_size

This function sets the stack size for a new thread. Unit: byte.

- **Prototype**

```
_thread.stack_size(size)
```

- **Parameter**

size:
Stack size. Default: 8192.

- **Return Value**

Return current stack size.

3.4. _thread.start_new_thread

This function adds a new thread.

- **Prototype**

```
_thread.start_new_thread(function, args)
```

- **Parameter**

function:
The function

args:
The parameters of the function to be executed

- **Return Value**

None

4 Multithreading Example

Step 1: Connect the EVB to the PC. The operation method after connecting to the PC is detailed in *Quectel_QuecPython_Basic_Operation_Guide*.

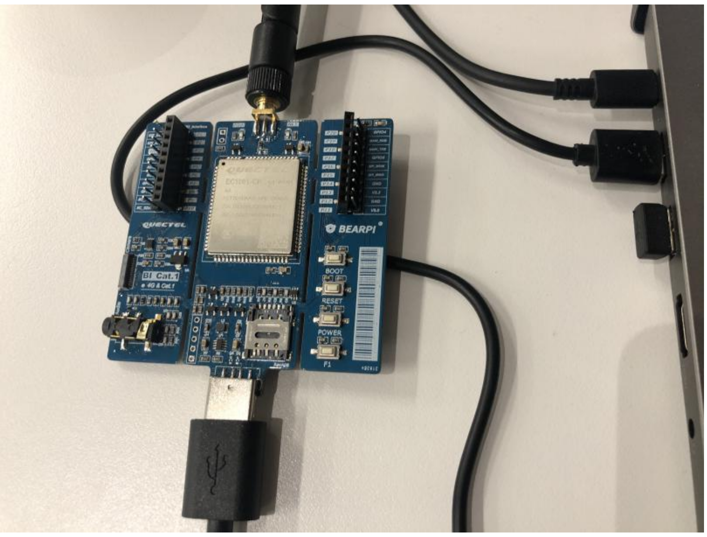


Figure 2: Connecting to the PC

Step 2: Create a *test.py* file. Import the _thread module in QuecPython into the file and write multithreading code.

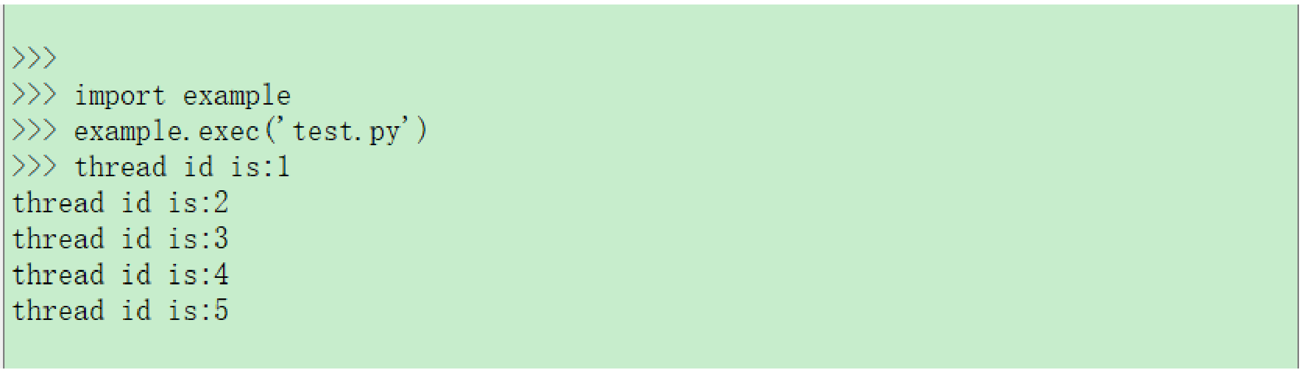
```
import _thread

def th_func(thread_id):
      Print("thread id is:%d" % thread_id)

for i in range(5):
    _thread.start_new_thread(th_func,(i+1,))
```

Step 3: Upload the *test.py* file to EVB. For the upload method, see *Quectel_QuecPython_Basic_Operation_Guide*.

Step 4: The result of the program is as shown in the figure:



```
>>>
>>> import example
>>> example.exec(’test.py’)
>>> thread id is:1
thread id is:2
thread id is:3
thread id is:4
thread id is:5
```

5 Appendix A References

Table 1: Related Documents

SN	Document Name	Remark
[1]	Quectel_QuecPython_Basic_Operation_Guide	QuecPython uploading and downloading fileintroduction

Table 2: Terms and Abbreviations

Abbreviation	Description
API	Application Programming Interface
EVB	Evaluation board
ID	Mostly refers to Identifier in terms of software
PC	Personal Computer