# QuecPython TTS Application Note

**LTE Standard Module Series**

Version: 1.0.0

Date: 2020-11-09

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

# About the Document

## Revision History

| Version | Date | Author | Description |
|---|---|---|---|
| - | 2020-11-09 | Rivern | Creation of the document |
| 1.0.0 | 2020-11-09 | Rivern | Preliminary |

# Contents

# 1 Introduction

This document takes Quectel EC100Y-CN module as an example to introduce how to use TTS function.

This document is applicable to the following Quectel modules:

- EC100Y-CN
- EC600S-CN

# 2 TTS Overview

## 2.1. Brief Introduction on TTS

TTS, Text To Speech, is a part of man-machine dialog, allowing the machine to speak. With the support of the embedded chip, it uses the neural network design to intelligently transform the text into a natural voice stream. TTS converts the text in real time with short time (in second). With the help of internal voice controller, TTS outputs the smooth voice without the unnatural feeling from the machine. Before using TTS feature on EVB board, see the figure below to know the audio interface of the module (taking EC100Y-CN as an example):

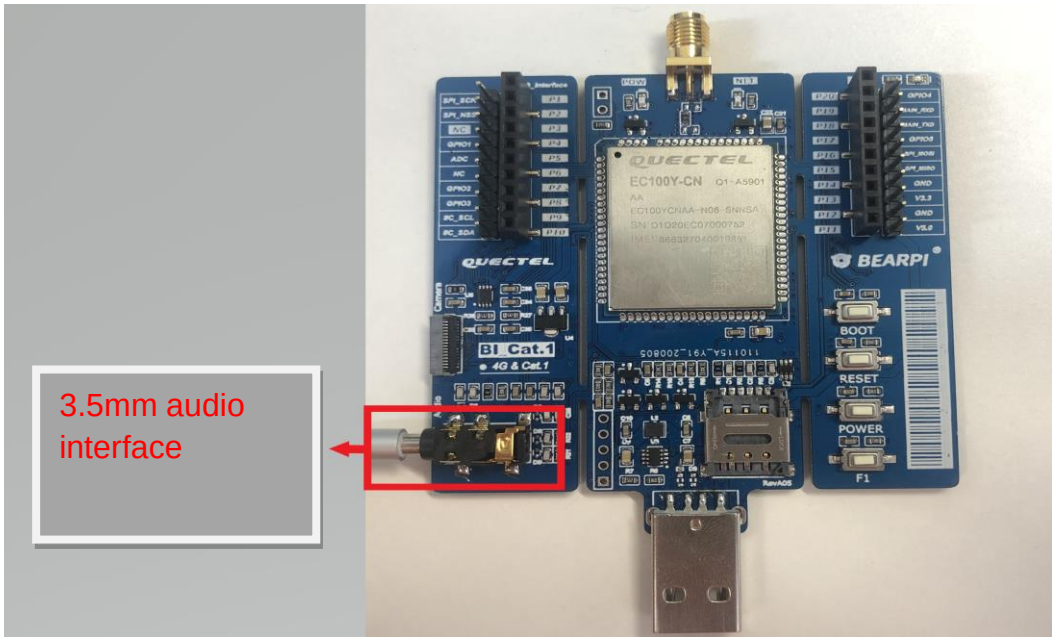



**Figure 1: TTS Module Audio Interface**

After connecting 3.5mm audio interface, see *Quectel-QuecPython-Cat1 EVB User Guide* for driver downloading and firmware installation.

# 3 TTS API Details

### 3.1.1. import audio tts = audio.TTS

This function imports audio library and creates a TTS object.

- **Prototype**

```
import audio tts = audio.TTS(device)
```

- **Parameter**

*device:*
The device type, and the values are as follows:
0 microphone
1 earphone
2 speakers

- **Return Value**

None.

### 3.1.2. tts.play

This function plays the voice.

- **Prototype**

```
tts.play(priority, breakin, mode, str)
```

- **Parameter**

*priority*:
Integer type. Play priority, the level is from 0 to 4. The larger the value, the higher the priority.

*breakin*:
Integer type. Whether to break in the voice playback.
0 Not allowed to be broken in
1 allowed to be broken in

*mode*:
Integer type. Encoding mode:

1 UNICODE16 (Size end conversion)
2 UTF-8
3 UNICODE16 (Don't convert)

*str*:
String type. String to be played.

- **Return Value**

0 the voice is played successfully.
-1 It fails to play voice
1 Voice could not play at once, the voice is added the queue to be played.
-2 voice could not play at once, the task in this queue has reached the upper limit and cannot be added to the playback queue.

**NOTE**

Each priority group can join up to 10 playback tasks at the same time. For the play queue policy, refer to the *Quectel QuecPython library API Introduction*.

### 3.1.3. tts.setSpeed

This function sets playback speed

- **Prototype**

```
tts.setSpeed(speed)
```

- **Parameter**

*speed*:
Integer type. Playback speed. Range: 0–9. the larger the value, the faster the speed.

- **Return Value**

The playback speed will be returned if the function is executed successfully, otherwise, -1 will be returned.

### 3.1.4. tts.setVolume

This function sets the playback volume.

- **Prototype**

```
tts.setVolume(vol)
```

- **Parameter**

*vol*:
Integer type. Playback volume. Range: 0–9. 0 means mute.

- **Return Value**

The playback volume will be returned if the function is executed successfully, otherwise, -1 will be returned.

### 3.1.5. tts.getSpeed

This function gets the current playback speed.

- **Prototype**

```
tts.getSpeed()
```

- **Parameter**

None.

- **Return Value**

The current playback speed will be returned if the function is executed successfully, otherwise, -1 will be returned.

### 3.1.6. tts.getVolume

This function gets the playback volume.

- **Prototype**

```
tts.getVolume()
```

- **Parameter**

None.

- **Return Value**

The current playback volume will be returned if the function is executed successfully, otherwise, -1 will be

returned.

### 3.1.7. tts.getState

This function gets the current playback status.

- **Prototype**

```
tts.getState()
```

- **Parameter**

None.

- **Return Value**

0 there is no TTS playing
1 TTS is currently playing

### 3.1.8. tts.stop

This function stops the playing.

- **Prototype**

```
tts.stop()
```

- **Parameter**

None.

- **Return Value**

0 The playing is stopped successfully.
-1 It fails to stop the playing.

### 3.1.9. tts.close

This function closes the TTS feature.

- **Prototype**

```
tts.close()
```

- **Parameter**

None.

- **Return Value**

0 TTS is closed successfully.
-1 It fails to close TTS.

# 4 TTS Usage Example

## 4.1. Execute Commands on the Command Line

In Xshell, connect the main serial port of the module, enter the communication interface, and then follow the steps below to use TTS feature:

**Step 1**: Import audio library and create a TTS object:

```
import audio tts = audio.TTS(device)
```


**Step 2**: Play voice

```
tts.play(priority, breakin, mode, str)
```

```
>>> for i in range(12):
...     tts.play(0,0,2,'222222222')
...
0
1
1
1
1
1
1
1
1
1
1
-2
>>>
```

**Step 3**: After the above step, you can hear the voice from the headset. At this time, you can set playback speed and volume through *tts.setSpeed(speed)* and *tts.setVolume(vol)* respectively.

```
>>> tts.setVolume(7)
0
>>> tts.setSpeed(8)
0
```

**Step 4**: Use *tts.getSpeed()* and *tts.getVolume()* to get the current playing speed and volume respectively. The current playback status can be obtained through *tts.getState()*.

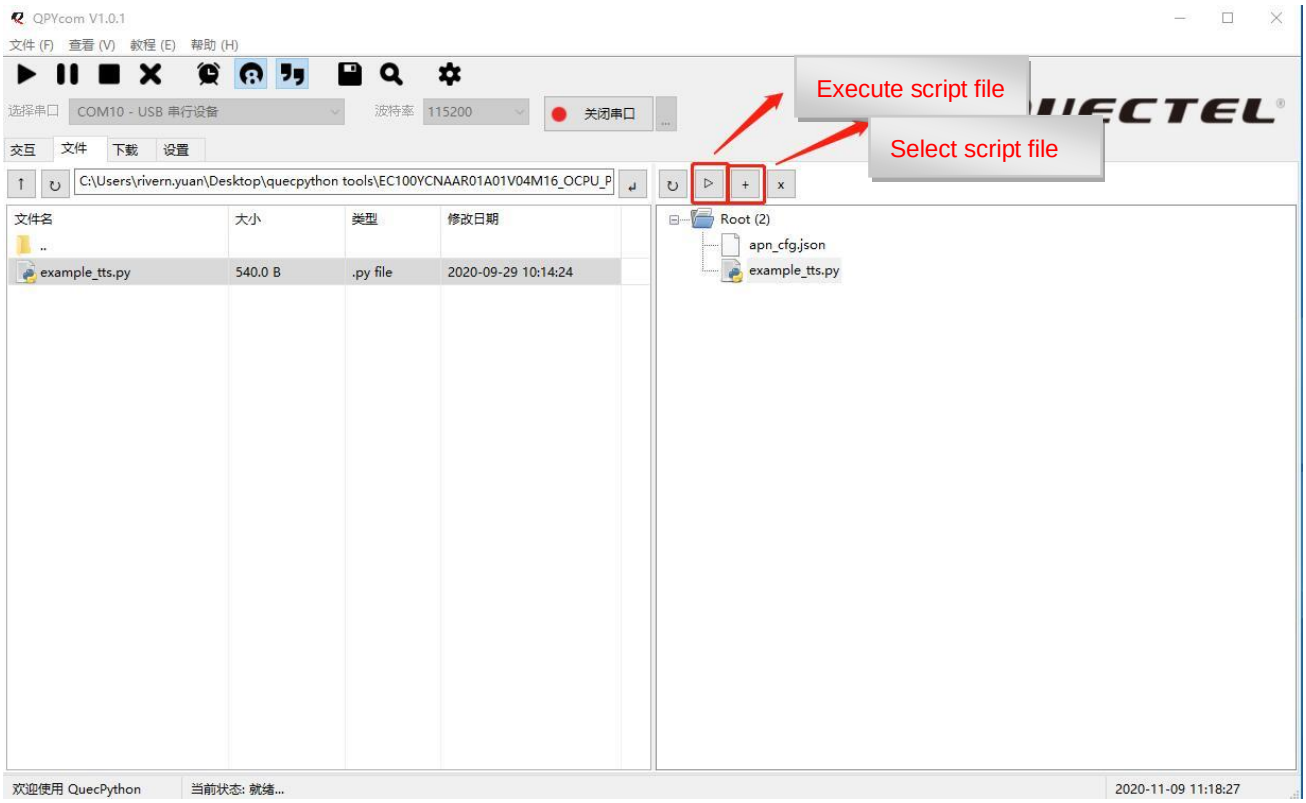
```
>>> tts.getSpeed()
8
>>> tts.getVolume()
7
>>> tts.getState()
0
```

**Step 5**: *tts.stop()* stops the playback. After the playback completes, *tts.close()* closes TTS feature.

```
>>> help(tts)
object tts  is of type TTS
  play -- <function>
  stop -- <function>
  close -- <function>
  setVolume -- <function>
  getVolume -- <function>
  setSpeed -- <function>
  getSpeed -- <function>
  getState -- <function>
>>> tts.stop()
>>> tts.close()
0
```

## 4.2. Execute py Files

**Step 1**: Enter the *modules* directory in SDK, find the TTS directory. Use the QPYcom tool to send the *example_tts.py* script file in the TTS directory to the module. Refer to *QPYcom User Guide* for details about sending and executing the script to the module.




**Step 2**: In Xshell, connect the main serial port of the module, enter the communication interface, use *uos.listdir()* to confirm whether the *example_tts.py* script file is in the current directory, and then follow the steps below:

```
>>> import uos
>>> uos.listdir()
['apn_cfg.json', example_tts.py', 'example_socket.py', 'example_thread.py']
>>> import example
>>> example.exec('example_tts.py')
>>>
```

Import example module by *import example*, which provides *exec()* to execute python scripts. Execute *example.tts.py* by *example.exec('example.tts.py')*, after that, you can hear the voice playing in the headset.

# 5 Appendix

**Table 1: Terms and Abbreviations**

| Abbreviation | Description |
|---|---|
| API | Application Programming Interface |
| LTE | Long Term Evolution |
| SDK | Software Development Kit |
| TTS | Text To Speech |
| UNICODE | Unicode |
| UTF | Universal Character Set/Unicode Transformation Format |