QuePython

WiKi 编写规范

版本：1.0.0

日期：2023-01-12

状态：临时文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登陆网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。
2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。
3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。
4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2023-01-12	Chavis CHEN	文档创建
1.0.0	2023-01-12	Chavis CHEN	临时版本

目录

图片索引

1 前言

本文档为移远通信 QuecPython 团队的对外 WiKi 编写规范，从样式、配色、内容等个方面进行阐述，旨在指引相关人员正确编写文档。

2 WiKi 编写规范

2.1. WiKi 中需编写的内容

2.1.1. 文档引言

引言部分用于概述文档的主要内容。

2.1.2. 整体内容

WiKi 中需描述的内容整体包含两个方面：模块与类。

2.1.2.1. 模块描述

Python 中一个内建库或一个 py 文件即为一个模块，对于 microPython 而言，uos、machine 等均为模块。

模块描述需包含以下内容：

- 模块的作用：用于描述模块功能、使用时的注意事项、整体用法演示等
- 模块对外的对象描述
- 模块对外的函数描述
- 模块对外的常量描述
- 模块对外的类的列举

2.1.2.2. 类描述

类一定在某一个模块中被定义。

类的描述需包含以下内容：

- 类的作用：用于描述类功能、使用时的注意事项、整体用法演示等
- 类的构造函数描述
- 类对外的方法描述
- 类对外的属性描述
- 类对外的常量描述

2.1.3. 内容细节

2.1.3.1. 函数描述内容

本章节所述函数包括模块中的函数、类的构造函数、类方法等任何具有函数性质的对象。

函数描述基本结构为：

- 函数名作为标题
- 函数原型定义作为正文的代码块
- 函数原型后编写函数功能描述
- 函数参数描述
- 函数返回值描述
- 函数调用时的注意事项
- 使用范例

如下图所示：

• **function_name**

```
function_name(arg1, arg2)
```

该函数用于xxx功能，能够xxxxx。

参数描述：

- arg1 - 参数1作用描述；取值范围1-8。
- arg2 - 参数2作用描述；取值范围见下表枚举：

arg2	值	含义
ARG2_ENUM_1	1	含义描述
ARG2_ENUM_2	5	含义描述
ARG2_ENUM_3	7	含义描述

返回值描述：

返回xxx对象/返回xxx执行的结果，取值范围如下：

返回值	含义
RES_ENUM_1	含义描述
RES_ENUM_2	含义描述

> 注意事项描述

示例：

```
res = function_name(arg1, arg2)
print(res)
```

图 1：函数描述的基本结构

备注

1. 对于模块中的函数，函数原型应写作 module.func(arg1, arg2)。
2. 对于类构造函数，函数原型应写作 class module.ClassName(arg1, arg2)。
3. 对于类方法，函数原型应写作 ClassName.func(arg1, arg2)。
4. 对于无参数的函数，省略包括标题和内容的整个参数描述项。
5. 对于无返回值的函数，省略包括标题和内容的整个返回值描述项。
6. 对于类构造函数，由于其返回的是类的对象，因此无需描述返回值。
7. 注意事项依实际情况而定，无则不写。
8. 对于具有一定应用复杂度或必须关联一定的上下文逻辑方能正确应用的函数，**必须**编写示例，其他情况可不编写。
9. 若示例代码演示的功能较复杂，请放置链接到文档中心或对外的 github 仓库的跳转链接，并编写内容为“点此查看使用示例”。

2.1.3.2. 对象、属性和常量描述内容

模块中对外的对象、类对外的属性及模块和类中对外的常量具有相似的原型，因此共同遵守如下规范：

- 原型仅作为标题，不在正文中将其再次写作代码块
- 功能描述
- 使用时的注意事项
- 使用范例

对象、属性和常量描述基本结构如下图所示：

图 2：对象、属性和常量描述的基本结构

2.1.3.3. 注意事项描述内容

1. 注意事项必须以 **markdown 的引用**方式突出显示。

2. 注意事项**必须**体现如下内容：

1) 若使用某功能时必须满足一定的前置条件，否则该功不能正常使用：在注意事项中，需描述无前置条件时为何不能正常使用以及会出现何种异常。
2) 若某功能仅支持部分硬件接口、部分场景或部分型号：在注意事项中，若内容篇幅较短，应进行详尽描述，必要时可辅以表格；若内容篇幅较长，需在文档中心添加模块功能说明版块描述此方面内容，并在当前注意事项中以链接的形式跳转至文档中心对应的版块。
3) 某些功能不可同时使用：在注意事项中，需清晰描述哪些功能不可同时使用、为何不能同时使用、若同时使用会导致当前功能与其互斥的功能出现何种异常。若内容篇幅较短，应进行详尽描述，必要时可辅以表格；若内容篇幅较长，需在文档中心添加模块功能说明版块描述此方面内容，并在当前注意事项中以链接的形式跳转至文档中心对应的版块。
4) 对于各型号模块 QuecPython GPIO 编号与模块引脚映射关系及其他类似内容的说明：由于篇幅太长，需在文档中心添加模块功能说明版块描述此方面内容。可按照模块型号分别进行说明，每个型号对应一个表格。在说明中，如遇某几个 GPIO 不能同时使用或其他需要特殊说明的部分，需在描述中进行即时备注，示例如下：

GPIO编号	引脚编号	备注
Pin.GPIO1	13	不可与引脚17同时作为GPIO使用
Pin.GPIO2	14	
Pin.GPIO3	17	不可与引脚13同时作为GPIO使用

图 3：特殊说明备注

5) 其他任何需要特别注意的地方。

2.1.3.4. 模块对外的类的列举

一个模块中经常包含一个或多个类的定义，这些类的描述应单独放在一个文档中，而在模块的描述文档中，仅列举出对外的类即可。这些类的列出项需带有超链接，点击该超链接后能够跳转到对应类的描述文档。

模块对外的类列举的基本结构如下：

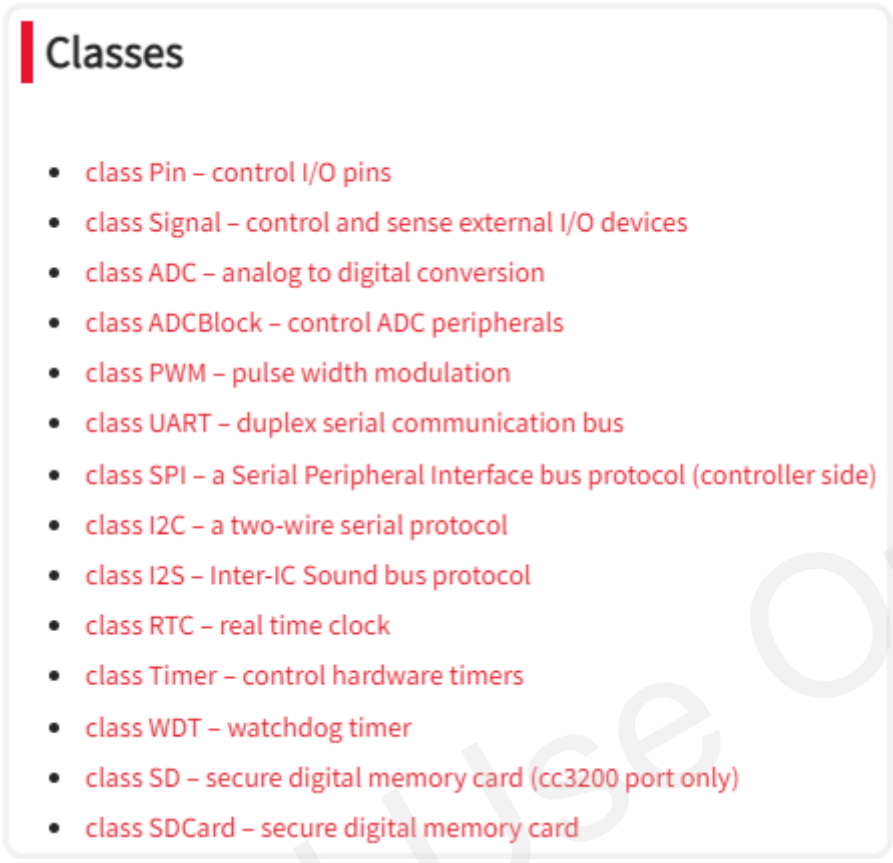

图 4：模块对外的类列举的基本结构

2.1.3.5. 其他

WiKi 编写人员在编写文档的过程中应注意接口或功能实现是否跨平台。例如，net.getApn(args)在部分平台上的参数为 1 个位置参数和 1 个可变参数，而在其它平台上为 2 个位置参数，这种接口实现在不同平台上不一致，失去了跨平台性。此时应向研发提出整改要求，整改通过后该部分内容方能编写在 WiKi 中。

以上描述了 WiKi 中需要编写的主要内容，下文将从具体的排版布局、样式等方面提供指引方法。

2.2. CSS 样式引入

在 markdown 源码的开头，复制并修改下面的代码，以引入 CSS 样式：

```
<link rel="stylesheet" type="text/css" href="path/to/file.css">
```

- *path/to/file.css* 表示实际的 CSS 文件路径。
- 后台会自动加载此 CSS 样式，本地编写的文档在提交时，需删除上面 CSS 样式引入的代码。

2.3. 文档引言

1. **必须**出现在渲染后文档的开头部分。
2. **必须**使用 Tab 键缩进，形成灰色引用效果。
3. 引言需简明扼要地描述本文的目的，帮助读者快速了解文章的整体内容。

渲染效果如下：

> 本文阐述了QuecPython的machine模块的用法，描述了machine模块最新版本的特性。

图 5：引言渲染效果

2.4. 标题层级

若无特殊需求，文档标题层级**建议不超过**三级。

2.5. 语言描述

2.5.1. 标题

标题仅用于简述功能或直接体现接口名称，**禁止**带入注意事项或其他与标题无关的内容。

2.5.2. 内容

1. 切合实际，不夸大、不遗漏、不偏离主题。
2. 使用**专业术语**。
3. 思维逻辑清晰，简洁明了，**禁止**口语化。
4. **禁止**暴露用户无需关心的细节。
5. 同一接口在不同外设、不同场景、不同平台中使用时，如果存在用法差异，**必须**进行详尽说明。

2.5.3. 表格

表格**必须**添加表头。

表头	表头	表头
正文	正文	正文

图 6：表格格式

2.5.4. 代码

1. 代码性质的文本**必须**标记为代码块。
1) 嵌入至文本中的代码（一般比较短，一个变量或接口等）使用一对“`”进行标记；
2) 单独一行或多行的代码，使用一对“```”进行标记，并且在开头处标记编程语言的类型，示例如下：

```python
import uos
uos.listdir('/usr')
```

图 7：单独一行或多行代码标记

2. 参考代码必要处**必须**添加注释，注释内容也**必须**遵守本章节规范。

3. 参考代码的编写风格：
1) 若需演示 REPL 相关的内容，**必须**按照如下示例编写参考代码：

```
>>> import uos
>>> uos.listdir('/usr')
['system.config', 'user.config']
```

2) 若演示的功能需要编写在脚本文件中，**必须**按照如下示例编写参考代码：

```
import uos
uos.listdir('/usr')
```

2.6. API 编写说明

2.6.1. 整体规范

1. 在 CSS 样式引入、文档引言、标题层级、添加重点标记内容等方面**必须**严格遵循前述内容。
2. 一个模块**必须**单独形成一篇文档，可参考 microPython-machine 的描述。
3. 一个类**必须**单独形成一篇文档，可参考 microPython-machine.Pin 的描述。
4. 函数的参数使用●开头的列表形式书写，若参数取值范围不连续且候选值较多，可使用表格描述该参数的取值范围，否则可不使用表格。
5. 函数的返回值如果具有容器性质（列表、字典、元组等），可使用表格描述每个元组，否则可不使用表格。
6. 此规范是基本约束，文档编写人员可在任意恰当的位置增加引用说明（markdown 中的>格式）或重点标记的内容。

备注

1. 无论是模块内的方法、以类名作为函数名的构造函数，还是类方法，只要具备函数性质的接口，对该接口的描述应使用相同的格式，包括函数原型定义、功能描述、参数描述及返回值描述。需重点关注的内容，**必须**通过 markdown 的引用或重点标记（Tips、Note、Warning）等进行描述。文中使用“必须”、“禁止”、“建议”等对编写规范进行限定约束，文档编写人员必须严格遵守。
2. 涉及的方法，有些无参数，有些无返回值或返回众所周知的结果（比如类的构造函数必定返回对应的对象），即可省略对应的描述。
3. 文档编写人员应综合考虑以上因素，根据实际情况，灵活调整。

2.6.2. 模块 API 说明编写规范

2.6.2.1. 一级标题

一级标题编写示例：

```
# `machine` - 硬件相关功能
```

渲染效果：

machine - 硬件相关功能

图 8：模块 API 一级标题渲染效果

备注

使用以下格式可插入一个带灰色圆角边框的图片，仅需修改 src 后的图片路径即可：
<img src="./media/module-API-under-h2.png" style="border-style: solid; border-radius: 10px; color: #f1f1f1;" border=3 alt="">

2.6.2.2. 一级标题后内容

该部分内容属于综述，用于描述模块功能、使用时的注意事项、整体用法演示等。

一级标题后内容编写示例：

```
# `machine` - 硬件相关功能

`machine`模块包含硬件相关的特定功能。大多数功能允许直接并且非严格地访问和控制系统硬件。如果使用不正确，可能会导致板级功能异常，甚至死机；极端条件下，可能会引起硬件损坏。

> machine 模块内的函数或类方法的回调函数，应当被看做在中断上下文中执行，这在物理设备或虚拟设备中均适用。

**示例：**

```python
import machine
from micropython import const

GPIOA = const(0x48000000)
GPIO_BSRR = const(0x18)
GPIO_IDR = const(0x10)

# set PA2 high
machine.mem32[GPIOA + GPIO_BSRR] = 1 << 2

# read PA3
value = (machine.mem32[GPIOA + GPIO_IDR] >> 3) & 1
```
```

渲染效果：

图 9：模块 API 一级标题后内容渲染效果

2.6.2.3. 二级标题

由于模块中存在对外的功能方法、常量或类等元素，**必须**按照顺序直接以 XXX1 功能、XXX2 功能、常量、Classes 作为二级标题。

二级标题编写示例：

```
## 内存访问功能

## 复位相关功能

## 中断相关功能

## 常量
```

```
## Classes```
```

渲染效果：

图 10：模块 API 二级标题渲染效果

2.6.2.4. 二级标题后内容

该部分内容用于模块内对外的功能方法、常量或类的描述。

- 对外功能内的具体方法**必须**作为三级标题。
- 该三级标题**必须**使用高亮效果显示。
- 功能方法或常量的说明**必须**作为三级标题后的段落内容。
- 该段落中，涉及到代码相关的单词或接口调用，**必须**使用高亮效果显示。
- 在 Classes 二级标题下，只需按照如下格式列出模块内的所有的类即可，无需使用三级标题，示例如下：

```
## Classes
- [class Pin – control I/O pins](./machine.Pin.md)
- [class Signal – control and sense external I/O devices](./machine.Signal.md)
```

备注

1. markdown 语法：*[display_text](path)*用于显示一个带链接的文本，其中 *path* 可以是本地的任何文件或任何 URL。点击该文本会打开链接到的本地文件或网站。
2. Classes 标题下类的枚举，需链接到该类对应的 API 说明文档。

二级标题后内容编写示例：

```
## 内存访问功能

本模块提供 3 个对象，用于原始内存访问。

### `machine.mem8`

读写 8 位内存。

### `machine.mem16`

读写 16 位内存。

### `machine.mem32`

读写 32 位内存。

<br>
使用脚标[...]来索引这些对象的内存。请注意，无论被访问的内存有多大，地址是按照字节访问的。

**示例：**

```python
import machine
from micropython import const

GPIOA = const(0x48000000)
GPIO_BSRR = const(0x18)
GPIO_IDR = const(0x10)

# set PA2 high
machine.mem32[GPIOA + GPIO_BSRR] = 1 << 2

# read PA3
value = (machine.mem32[GPIOA + GPIO_IDR] >> 3) & 1
```

复位相关功能

`machine.reset`

```python
machine.reset()
```
```

```

以类似于按下外部复位按键的方式复位设备。

### `machine.soft_reset`

```python
machine.soft_reset()
```

复位虚拟机解释器，删除所有的 Python 对象并且复位 Python 的堆内存。

中断相关功能

以下函数允许控制中断。某些系统需要中断正常运行，因此长时间禁用它们可能会破坏核心功能，例如看门狗定时器可能会意外触发。中断只应在最短时间内禁用，然后重新启用，恢复到以前的状态。例如：

```python
import machine

# Disable interrupts
state = machine.disable_irq()

# Do a small amount of time-critical work here

# Enable interrupts
machine.enable_irq(state)
```

`machine.disable_irq`

```python
machine.disable_irq()
```

禁用中断请求。返回此前的中断状态。此返回值应传递给`enable_irq()`函数，以还原调用`disable_irq()`之前的中断状态。

`machine.enable_irq`

```python
machine.enable_irq(state)
```
```

启用中断请求。`state`参数应该是最近一次调用`disable_irq()`函数时的返回值。

常量

`machine.IDLE`

此处添加描述。

`machine.SLEEP`

此处添加描述。

`machine.DEEPSLEEP`

此处添加描述。

Classes

- [class Pin – 控制 I/O 引脚](./machine.Pin.md)
- [class Signal – 控制和感知外部 I/O 设备](./machine.Signal.md)
- [class ADC – 模数转换](./machine.ADC.md)
- [class ADCBlock – 控制 ADC 外设](./machine.ADCBlock.md)
- [class PWM – 脉冲宽度调制](./PWM.md)
- [class UART – 串口通信](./machine.UART.md)
- [class SPI – SPI 通信 (主机功能)](./machine.SPI.md)
- [class I2C – I2C 通信](./machine.I2C.md)
- [class I2S – I2S 音频协议](./machine.I2S.md)
- [class RTC – 实时时钟](./machine.RTC.md)
- [class Timer – 硬件定时器](./machine.Timer.md)
- [class WDT – 看门狗定时器](./machine.WDT.md)
- [class SD – SD 卡 (仅支持 cc3200)](./machine.SD.md)
- [class SDCard – SD 卡](./machine.SDCard.md)

渲染效果：

内存访问功能

本模块提供3个对象，用于原始内存访问。

- **machine.mem8**

读写8位内存。

- **machine.mem16**

读写16位内存。

- **machine.mem32**

读写32位内存。

使用脚标[...]来索引这些对象的内存。请注意，无论被访问的内存有多大，地址是按照字节访问的。

示例：

```
import machine
from micropython import const

GPIOA = const(0x48000000)
GPIO_BSRR = const(0x18)
GPIO_IDR = const(0x10)

# set PA2 high
machine.mem32[GPIOA + GPIO_BSRR] = 1 << 2

# read PA3
value = (machine.mem32[GPIOA + GPIO_IDR] >> 3) & 1
```

复位相关功能

- **machine.reset**

```
machine.reset()
```

以类似于按下外部复位按键的方式复位设备。

- **machine.soft_reset**

```
machine.soft_reset()
```

复位虚拟机解释器，删除所有的Python对象并且复位Python的堆内存。

中断相关功能

以下函数允许控制中断。某些系统需要中断正常运行，因此长时间禁用它们可能会破坏核心功能，例如看门狗定时器可能会受外触发。中断只应在最短时间内禁用，然后重新启用，恢复到以前的状态。例如：

```
import machine

# Disable interrupts
state = machine.disable_irq()

# Do a small amount of time-critical work here

# Enable interrupts
machine.enable_irq(state)
```

- **machine.disable_irq**

```
machine.disable_irq()
```

禁用中断请求。返回此前的中断状态。此返回值应传递给enable_irq()函数，以还原调用disable_irq()之前的中断状态。

- **machine.enable_irq**

```
machine.enable_irq(state)
```

启用中断请求。state参数应该是最近一次调用disable_irq()函数时的返回值。

常量

- **machine.IDLE**

此处添加描述。

- **machine.SLEEP**

此处添加描述。

- **machine.DEEPSLEEP**

此处添加描述。

Classes

- class Pin - 控制I/O引脚
- class Signal - 控制和感知外部I/O设备
- class ADC - 模数转换
- class ADCBlock - 控制ADC外设
- class PWM - 脉冲宽度调制
- class UART - 串口通信
- class SPI - SPI通信(主机功能)
- class I2C - I2C通信
- class I2S - I2S音频协议
- class RTC - 实时时钟
- class Timer - 硬件定时器
- class WDT - 看门狗定时器
- class SD - SD卡 (仅支持cc3200)
- class SDCard - SD卡

图 11：模块 API 二级标题后内容渲染效果

2.6.3. 类 API 说明编写规范

2.6.3.1. 一级标题

一级标题编写示例：

```
# `class Pin` - I/O 引脚控制
```

渲染效果：

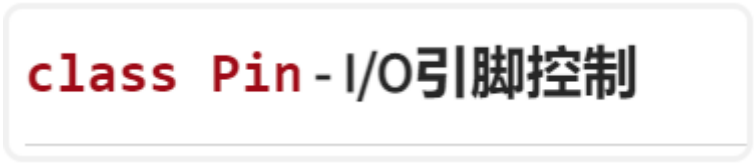

图 12：类 API 一级标题渲染效果

2.6.3.2. 一级标题后内容

该部分内容属于综述，用于描述类功能、使用时的注意事项、整体用法演示等。

一级标题后内容编写示例：

```
# `class Pin` - I/O 引脚控制

Pin 对象用来控制 I/O 引脚（以 GPIO 为大家所熟知）。Pin 对象一般与一个物理引脚关联，可以驱动输出电平或读取电平。Pin 类提供方法来设置引脚的模式、设置或获取引脚逻辑电平。对模拟引脚的控制，参考[ADC](http://docs.micropython.org/en/latest/esp32/quickref.html#ADC)类。

Pin 对象是通过制定具体 I/O 引脚的标识符来构造的。这个标识符可能是与端口或引脚编号相关的整数、字符串或元组。

**示例：**

```python
from machine import Pin

# create an output pin on pin #0
p0 = Pin(0, Pin.OUT)

# set the value low then high
p0.value(0)
```
```

```
p0.value(1)

# create an input pin on pin #2, with a pull up resistor
p2 = Pin(2, Pin.IN, Pin.PULL_UP)

# read and print the pin value
print(p2.value())

# reconfigure pin #0 in input mode with a pull down resistor
p0.init(p0.IN, p0.PULL_DOWN)

# configure an irq callback
p0.irq(lambda p:print(p))
```

渲染效果：

class Pin - I/O引脚控制

Pin对象用来控制I/O引脚（以GPIO为大家所熟知）。Pin对象一般与一个物理引脚关联，可以驱动输出电平或读取电平。Pin类提供方法来设置引脚的模式、设置或获取引脚逻辑电平。对模拟引脚的控制，参考ADC类。

Pin对象是通过制定具体I/O引脚的标识符来构造的。这个标识符可能是与端口或引脚编号相关的整数、字符串或元组。

示例：

```
from machine import Pin

# create an output pin on pin #0
p0 = Pin(0, Pin.OUT)

# set the value low then high
p0.value(0)
p0.value(1)

# create an input pin on pin #2, with a pull up resistor
p2 = Pin(2, Pin.IN, Pin.PULL_UP)

# read and print the pin value
print(p2.value())

# reconfigure pin #0 in input mode with a pull down resistor
p0.init(p0.IN, p0.PULL_DOWN)

# configure an irq callback
p0.irq(lambda p:print(p))
```

图 13：类 API 一级标题后内容渲染效果

2.6.3.3. 二级标题

由于类中会存在构造函数（与类同名）、对外的方法、属性和常量等元素，**必须**按照顺序直接以构造函数、方法、属性，常量作为二级标题。

二级标题编写：

```
## 构造函数

## 方法

## 属性

## 常量
```

渲染效果：

图 14：类 API 二级标题渲染效果

2.6.3.4. 二级标题后内容

该部分内容用于类内对外的方法、属性或常量的描述。

- 对外的方法、属性或常量名**必须**作为三级标题。
- 该三级标题**必须**使用高亮效果显示。
- 对方法、属性或常量的说明**必须**作为三级标题后的段落内容。
- 该段落中涉及到代码相关的单词或接口调用，**必须**使用高亮效果显示。

2.6.3.4.1. 构造函数说明

1. 三级标题**必须**以 module.Class 的形式命名。
2. 构造函数原型定义**必须**以 class module.Class(args)形式命名。

构造函数说明编写示例：

```
## 构造函数

### `machine.Pin`

```python
class machine.Pin(id, mode=- 1, pull=- 1, *, value=None, drive=0, alt=- 1)
```

创建 Pin 对象，以访问关联给定`id`的引脚外设。若额外的参数被传入构造函数，则用于初始化该引脚。任何没有指定的参数，将保持之前的状态。

参数描述：

- `id` - 引脚标识，int 型表示引脚编号，str 型表示引脚名称，tuple 型表示([port, pin],)；点此查看引脚编号与物理引脚的映射关系。
- `mode` - 引脚输入输出模式，`Pin.IN`：输入模式，`Pin.OUT`：输出模式。

    引脚编号与物理引脚的映射关系：

引脚编号	引脚名称	物理引脚脚序
`Pin.GPIO0`	`"gpio0"`	`10`
`Pin.GPIO0`	`"gpio0"`	`10`
`Pin.GPIO0`	`"gpio0"`	`10`
```

渲染效果：

构造函数

• machine.Pin

```
class machine.Pin(id, mode=- 1, pull=- 1, *, value=None, drive=0, alt=- 1)
```

创建Pin对象，以访问关联给定id的引脚外设。若额外的参数被传入构造函数，则用于初始化该引脚。任何没有指定的参数，将保持之前的状态。

参数描述：

- id - 引脚标识，int型表示引脚编号，str型表示引脚名称，tuple型表示([port, pin],)； 点此查看 引脚编号与物理引脚的映射关系。
- mode - 引脚输入输出模式，Pin.IN：输入模式，Pin.OUT：输出模式。

引脚编号与物理引脚的映射关系：

引脚编号	引脚名称	物理引脚脚序
Pin.GPIO0	"gpio0"	10
Pin.GPIO0	"gpio0"	10
Pin.GPIO0	"gpio0"	10

图 15：构造函数渲染效果

2.6.3.4.2. 方法说明

1. 三级标题必须以 Class.method 的形式命名。
2. 构造函数原型定义必须以 Class.method(args)形式命名。

方法说明编写示例：

```
## 方法

### Pin.value

```python
Pin.value([x])
```

该方法用于设置或读取引脚电平，取决于参数`x`是否提供。

参数描述：
```

- `x` - 提供该参数，用于设置引脚电平；`1` 表示设置引脚电平为高，`0`表示设置引脚电平为低。

返回值描述：

当参数`x`没有提供时，返回引脚电平；`1` 表示获取到的引脚电平为高，`0`表示获取到的引脚电平为低。

渲染效果：

图 16：方法说明渲染效果

2.6.3.4.3. 属性和常量说明

三级标题必须以 Class.attr 或 Class.constants 的形式命名。

属性和常量说明示例：

```
## 属性

### Pin.attr1

此处添加属性功能描述。

### Pin.attr2

此处添加属性功能描述。

## 常量
```

```
### Pin.IN

此处添加常量功能描述。

### Pin.OUT

此处添加常量功能描述。
```

渲染效果：

图 17：属性和常量说明渲染效果

3 案例演示

1. machine 模块：参考 module-machine.md 文件
2. machine.Pin 类：参考 machine.Pin.md 文件
3. 英文文档效果：参考 demo1.md 文件和 demo2.md 文件