一、 基础 完整的生成链条

📘 Jupyter Notebook：从 CSV 到 HTML 的完整流水线
🔷 Step 0. 环境初始化

安装依赖（只需运行一次）

!pip install pandas pyyaml sphinx sphinx_rtd_theme

🔷 Step 1. 准备 CSV 文件

📄 commands.csv 文件内容示例：

命令,命令标题,命令类型,命令格式,示例命令,示例响应,功能描述,备注,参数JSON "AT+CREG","查询网络注册状态","执行;查询;测试","AT+CREG=[]|AT+CREG?|AT+CREG=?","AT+CREG=1|AT+CREG?|AT+CREG=?","OK|+CREG: 0,1 OK|+CREG: (0-2) OK","查询或控制模块的网络注册状态","AT+CREG=5 返回ERROR（参数超出范围）","[{""name"": """", ""desc"": ""控制结果代码输出方式"", ""values"": {""0"": ""禁用上报"", ""1"": ""启用上报"", ""2"": ""上报并包含LAC/CI信息""}}, {""name"": """", ""desc"": ""网络注册状态"", ""values"": {""0"": ""未注册"", ""1"": ""已注册本地网络"", ""2"": ""正在搜索"", ""3"": ""注册被拒绝"", ""4"": ""未知"", ""5"": ""已注册漫游""}}, {""name"": """", ""desc"": ""接入技术"", ""values"": {""0"": ""GSM"", ""2"": ""UTRAN"", ""7"": ""E-UTRAN""}}]"

🔷 Step 2. CSV → YAML import pandas as pd, json, yaml, os

df = pd.read_csv('commands.csv') os.makedirs('data', exist_ok=True) cmd_objects = []

for _, row in df.iterrows(): param_json = json.loads(row['参数JSON']) cmd_data = { 'command': row['命令'], 'title': row['命令标题'], 'type': [t.strip() for t in row['命令类型'].split(';')], 'formats': [f.strip() for f in row['命令格式'].split('|')], 'parameters': param_json, 'examples': [{'cmd': c.strip(), 'resp': r.strip()} for c, r in zip(row['示例命令'].split('|'), row['示例响应'].split('|'))], 'description': row['功能描述'], 'notes': row['备注'] } cmd_objects.append(cmd_data) with open(f"data/{row['命令']}.json", 'w', encoding='utf-8') as f: json.dump(cmd_data, f, ensure_ascii=False, indent=2)

with open('all_commands.yaml', 'w', encoding='utf-8') as f: yaml.safe_dump({'commands': cmd_objects}, f, allow_unicode=True, sort_keys=False)

print('✅ 已生成 data/*.json 和 all_commands.yaml')

🔷 Step 3. YAML → RST import yaml, os

with open('all_commands.yaml', 'r', encoding='utf-8') as f: data = yaml.safe_load(f) commands = data['commands']

def generate_rst(cmd): rst = [] rst.append(f"{cmd['command']}\n{'=' * len(cmd['command'])}\n") rst.append(f"**命令标题**：{cmd['title']}\n") rst.append(f"**命令类型**：{', '.join(cmd['type'])}\n\n") rst.append("命令格式::\n") for fmt in cmd['formats']: rst.append(f" {fmt}") rst.append("") rst.append("参数说明\n--------\n") rst.append(".. list-table::\n :header-rows: 1\n :widths: 15 30 40\n") rst.append(" * - 参数名\n - 描述\n - 取值\n") for p in cmd.get('parameters', []): values = '\n'.join([f"{k}: {v}" for k, v in p['values'].items()])

rst.append(f" * - {p['name']}\n - {p['desc']}\n - {values}") rst.append("") rst.append("示例\n----\n") for ex in cmd.get('examples', []): rst.append(f".. code-block:: none\n\n {ex['cmd']}\n {ex['resp']}\n") rst.append("") rst.append(f"**功能描述**：{cmd.get('description','')}\n\n") if cmd.get('notes'): rst.append(f"**注意**：{cmd['notes']}\n") return '\n'.join(rst)

os.makedirs('rst_output', exist_ok=True) for cmd in commands: with open(f"rst_output/{cmd['command']}.rst", 'w', encoding='utf-8') as f: f.write(generate_rst(cmd))

index = ['AT 命令手册\n==================\n', '.. toctree::\n :maxdepth: 1\n'] for cmd in commands: index.append(f" {cmd['command']}") with open('rst_output/index.rst', 'w', encoding='utf-8') as f: f.write('\n'.join(index))

print('✅ 已生成 rst_output/*.rst 与 index.rst')

🔷 Step 4. 初始化 Sphinx 项目 !sphinx-quickstart docs --sep --project "AT Command Manual" --author "Your Team" --release "1.0" -q !pip install sphinx_rtd_theme

修改 docs/source/conf.py 文件：

import os, sys sys.path.insert(0, os.path.abspath('../..')) html_theme = 'sphinx_rtd_theme'

🔷 Step 5. 复制 RST 文件到 Sphinx 项目中 !cp -r rst_output/* docs/source/

🔷 Step 6. 构建 HTML / PDF

构建 HTML

!make -C docs html

（可选）安装 latexpdf 支持

!sudo apt-get install texlive-xetex -y !make -C docs latexpdf

✅ 最终成果 docs/ ├── build/html/index.html ← 打开查看网页版文档 ├── build/latex/*.pdf ← PDF 格式手册（可选） └── source/.rst ← 自动生成的结构化源文档

二、调整 生成目标格式的html内容

调整Sphinx配置生成ESP-AT命令手册格式HTML文档指南

一、主题选择与基础配置

1.1 核心主题选型

ESP-AT命令手册推荐使用乐鑫定制的`sphinx_idf_theme`主题，该主题内置命令分类、参数表格等专用模板，直接适配AT指令文档需求[1]。若需自定义，可基于Sphinx内置`classic`主题扩展，通过`html_theme_options`调

整侧边栏位置、关系栏颜色等视觉元素[10][17]。

1.2 基础配置项设置

在`conf.py`中需配置以下核心参数：

```
# 项目元数据
project = "ESP-AT Command Guide"
copyright = "2025, Espressif Systems"
author = "Espressif IoT Team"
version = "2.4.0"
release = "2.4.0.0"

# 主题设置
html_theme = "sphinx_idf_theme"
html_theme_path = ["path/to/sphinx_idf_theme"]  # 若使用自定义主题路径[10]

# 扩展配置
extensions = [
    "sphinx.ext.autodoc",           # 自动提取代码注释
    "sphinx_copybutton",            # 代码块复制按钮
    "sphinxcontrib.wavedrom",       # 时序图支持[3]
]
```

二、目录结构设计

2.1 推荐目录结构

采用以下层级结构组织文档源文件，确保与ESP-AT手册的模块化设计一致[4][6]：

```
docs/
├── source/
│   ├── conf.py                 # Sphinx配置文件
│   ├── index.rst               # 首页入口
│   ├── _static/                # 静态资源（图片/CSS/JS）
│   ├── _templates/             # 自定义模板
│   ├── at_commands/            # AT命令文档主目录
│   │   ├── index.rst           # 命令总索引
│   │   ├── general.rst         # 通用命令（AT+RST等）
│   │   ├── wifi.rst            # Wi-Fi相关命令
│   │   └── ble.rst             # 蓝牙相关命令
│   └── api_guides/             # 辅助指南文档
│       ├── getting_started.rst
│       └── troubleshooting.rst
└── build/                      # 构建输出目录（自动生成）
```

2.2 首页与目录树配置

`index.rst`需通过`toctree`指令定义文档层级：

```
Welcome to ESP-AT Command Documentation
=======================================

.. toctree::
   :maxdepth: 2
   :caption: Command Reference:

   at_commands/general
   at_commands/wifi
   at_commands/ble

.. toctree::
   :maxdepth: 1
   :caption: Guides:

   api_guides/getting_started
   api_guides/troubleshooting
```

设置`:maxdepth: 2`确保目录树显示两级标题，`:caption:`参数用于分组显示[14]。

三、命令模板与格式化

3.1 命令描述模板设计

使用reStructuredText的`list-table`指令格式化AT命令参数，示例：

```
AT+CWJAP - Join Wi-Fi Network
-----------------------------

.. list-table::
   :header-rows: 1
   :widths: 15 10 20 55

   * - Parameter
     - Type
     - Range
     - Description
   * - ssid
     - String
     - 1-32 bytes
     - Wi-Fi network name[2]
   * - pwd
     - String
     - 0-64 bytes
     - Password (WPA2-PSK required)
```

该格式与ESP-AT手册的参数表格样式一致，支持响应式布局[1][26]。

3.2 代码块与示例格式化

使用`code-block`指令添加AT命令示例，配合`sphinx_copybutton`扩展实现复制功能：

```
.. code-block:: bash
   :caption: Join Wi-Fi example

   AT+CWJAP="MyWiFi","MyPassword"    # 连接AP
   OK                                # 成功响应[1]
```

四、侧边栏与导航配置

4.1 侧边栏结构定义

通过`html_sidebars`配置侧边栏组件，典型配置包含：

```
html_sidebars = {
    '**': [
        'globaltoc.html',         # 全局目录树
        'relations.html',         # 上/下一篇导航
        'searchbox.html',         # 搜索框
        'sourcelink.html',        # 源码链接[17]
    ]
}
```

设置`globaltoc_collapse = True`可实现章节折叠功能[10]，优化长文档导航体验。

4.2 固定侧边栏实现

如需ESP-AT手册式固定侧边栏，可通过自定义CSS实现：

1. 在`_static/custom.css`中添加：

```
.wy-nav-side {
    position: fixed;
    height: 100%;
    overflow-y: auto;
}
```

2. 在`conf.py`中引用自定义样式：

```
html_css_files = ['custom.css']  # 添加自定义CSS[26]
```

五、模板定制与高级功能

5.1 页面布局定制

通过重写Jinja2模板修改页面结构：

1. 在`_templates/layout.html`中扩展基础模板：

```
{% extends "!layout.html" %}
{% block extrahead %}
    <meta name="description" content="ESP-AT Command Reference">
    <link rel="stylesheet" href="{{ pathto('_static/at_custom.css', 1)
}}">
{% endblock %}
```

2. 自定义命令卡片组件，使用`sphinx-design`扩展的`card`指令[16]：

```
.. card:: AT+RST
   :class-card: at-command-card
   :footer: *Restart module*

   Reset the ESP32 module to factory settings.
```

5.2 多版本与搜索配置

1. 版本切换功能：配置`versions_url`指向版本控制JS文件[3]：

```
html_context = {
    'versions_url': '_static/versions.js',  # 版本选择器数据
}
```

2. 中文搜索支持：安装`sphinxcontrib-jieba`扩展并配置：

```
extensions += ['sphinxcontrib.jieba']
html_search_language = 'zh'
```

六、构建与部署流程

6.1 本地构建命令

执行以下命令生成HTML文档：

```
# 清理旧构建
make clean
# 生成HTML输出
make html
```

构建结果位于`build/html`目录，可直接通过浏览器打开`index.html`查看[5][14]。

6.2 CI/CD集成

配置GitLab CI/CD流水线实现自动构建：

```
stages:
  - build_docs

build_docs:
  image: python:3.9
  script:
    - pip install -r requirements.txt
    - make html
  artifacts:
    paths:
      - build/html
```

构建结果可自动推送到飞书知识库或文档门户，形成"Docs-as-Code"闭环[6]。

七、关键配置对比表

配置项	ESP-AT手册推荐值	标准Sphinx默认值
`html_theme`	`sphinx_idf_theme`[1]	`alabaster`[10]
`extensions`	包含`sphinx_copybutton`[3]	仅基础扩展
`html_sidebars`	固定全局目录树+搜索框	动态生成导航
`source_suffix`	`.rst`	`.rst`
`templates_path`	`_templates`（含命令模板）	空列表

通过以上配置，可使Sphinx生成的文档在结构、样式和交互体验上接近ESP-AT命令手册的专业水准。实际应用中建议结合`sphinx_idf_theme`源码和ESP-Docs用户指南进一步优化细节[3][26]。