应用笔记

Application Note

文档编号：**AN1161**

APM32F402_F403_pyOCD 内置支持

版本：**V1.0**

1 引言

当我们需要为一款尚未被 pyOCD 官方（或社区）列入内置支持清单的芯片添加下载/调试能力时，最常见的做法是利用 CMSIS-Pack 的外部支持文件，或直接在 pyOCD 中编写一个 Target Python 脚本。

本应用笔记旨在详细介绍第二种方法：如何将极海 APM32F402/APM32F403 芯片的支持深度集成到 pyOCD 内部，使其成为一个内置可识别的目标。通过这种方式，用户将不再依赖外部 Pack 文件，可以使用简单的命令行参数（如 -t apm32f402xb）直接进行操作，从而简化开发流程、方便团队协作。

本应用笔记适用于极海 APM32F402/F403，以下正文以 APM32F402 为例。

目录

1 引言 1

2 添加内置支持的优势与思路 3

3 操作步骤 4

3.1 前期准备 4

3.2 解析 .FLM 文件并生成 Python 算法脚本 4

3.3 编写 target_APM32F402xx.py 4

3.4 注册目标型号 6

4 测试验证 7

4.1 查看目标列表 7

4.2 测试擦除 7

4.3 测试下载 7

5 版本历史 8

2 添加内置支持的优势与思路

将 APM32F402/F403 的支持直接添加到 pyOCD 中，主要有以下优势：

- 方便团队分发与环境一致：所有团队成员使用一份自定义的 pyOCD 脚本，可存放在版本库中共享，确保开发环境的统一性和快速部署。
- 支持更灵活的定制修改：可在脚本中灵活调整扇区大小、配置保护位等高级调试选项，不受第三方 Pack 文件限制。
- 深入学习 pyOCD 内部机制：有助于理解 pyOCD 的工作原理，为解决未来可能遇到的复杂问题打下基础。

实现该功能的思路如下：

- 从官方 SDK Pack 中提取芯片的 Flash 编程算法文件（.FLM）。
- 使用 pyOCD 提供的 generate_flash_algo.py 脚本，将 .FLM 文件解析为 Python 格式的 Flash 算法数据。
- 创建一个新的 Target 定义文件（例如 target_APM32F402xx.py），在其中整合生成的算法数据，并定义芯片的内存映射（FlashRegion、RamRegion 等）。
- 在 pyOCD 的 builtin/__init__.py 文件中注册新的目标型号，使其能够被 pyOCD 识别。
- 最终通过实际的擦除和下载操作，验证内置支持是否成功。

3 操作步骤

3.1 前期准备

（1） 安装 Python 和 pyOCD

确保本地已安装 Python 3.7+ 环境，并通过 pip 安装 pyOCD。建议使用 0.36.0 或更高版本。

```
pip install pyocd
```

（2） 获取 APM32F402 的 .FLM 文件

从极海官方的 APM32F402_403_SDK_V1.0.1 中找到 Geehy.APM32F4xx_DFP.1.0.7.pack 文件。使用解压工具（如 7-Zip）解压后，在目录中搜索到 APM32F402_128.FLM 文件，此文件即为后续操作所需的核心 Flash 算法文件。

（3） 准备 generate_flash_algo.py 脚本

此脚本随 pyOCD 源码一同发布。您可以从 pyOCD 的官方 GitHub 仓库 (https://github.com/pyocd/pyOCD) 下载源码，并在 scripts/ 目录下找到该文件。

3.2 解析 .FLM 文件并生成 Python 算法脚本

将 APM32F402_128.FLM 文件复制到 generate_flash_algo.py 脚本所在的目录。打开命令行并进入该目录，执行以下命令：

```
python generate_flash_algo.py -o apm32f402_flash_algo.py APM32F402_128.FLM
--ram-address=0x20000000 --stack-size=0x1000
```

注意：

（1） -o apm32f402_flash_algo.py: 指定输出的 Python 脚本文件名。

（2） --ram-address=0x20000000: 指定 MCU 的 RAM 起始地址，用于脚本生成地址相关的代码。

（3） --stack-size=0x1000: 为 Flash 算法分配的栈空间大小，可根据实际需求调整。

命令执行成功后，将生成 apm32f402_flash_algo.py 文件，其中包含 Flash 算法的指令数据和函数入口点等信息。

3.3 编写 target_APM32F402xx.py

在 pyOCD 的安装目录(例如 .../site-packages/pyocd/target/builtin)下，创建一个新的 Python 文件，命名为 target_APM32F402xx.py。

将 apm32f402_flash_algo.py 文件中生成的 FLASH_ALGO 字典内容复制到新文件中，并定义

APM32F402xB 类，示例如下：

```
from ...coresight.coresight_target import CoreSightTarget
from ...core.memory_map import (FlashRegion, RamRegion, MemoryMap)

# 从 generate_flash_algo.py 生成的内容
FLASH_ALGO = {
    'load_address': 0x20000000,
    'instructions': [
        0xe7fdbe00, 0x4603b510,
        # ... (此处省略更多十六进制指令数据) ...
        0x00000000
    ],
    'pc_init': 0x20000005,
    'pc_unInit': 0x20000035,
    'pc_program_page': 0x200000c3,
    'pc_erase_sector': 0x20000083,
    'pc_eraseAll': 0x20000047,
    'static_base': 0x2000013c,
    'begin_stack': 0x20001940,
    'end_stack': 0x20000940,
    'begin_data': 0x20001000,
    'page_size': 0x400,
    'analyzer_supported': False,
    'analyzer_address': 0x00000000,
}

class APM32F402xB(CoreSightTarget):
    VENDOR = "Geehy"
    MEMORY_MAP = MemoryMap(
        FlashRegion(
            start=0x08000000,
            length=0x20000,          # 128KB
            blocksize=0x400,         # 通常编程粒度：1KB
            is_boot_memory=True,
            algo=FLASH_ALGO
        ),
        RamRegion(
            start=0x20000000,
            length=0x8000            # 例如 32KB，或按实际修改
        )
    )
```

```
    def __init__(self, session):
        super().__init__(session, self.MEMORY_MAP)
```

注意：

（1） FlashRegion 和 RamRegion 的起始地址和大小必须与 APM32F402 芯片的实际规格一致，以避免编程错误。

（2） VENDOR 属性应设置为 "Geehy"。

3.4 注册目标型号

为了让 pyOCD 能够识别我们新添加的 APM32F402xB 类，需要修改 pyocd/target/builtin/__init__.py 文件，将新的目标型号注册到 BUILTIN_TARGETS 字典中。 打开 __init__.py 文件，添加以下代码：

```
# ... (其他已有的导入)
from . import target_APM32F402xx

BUILTIN_TARGETS = {
    # ... (若这里已存在其它已注册的目标，可保持不动)
    "apm32f402xb": target_APM32F402xx.APM32F402xB,

}
```

注意：首先导入我们创建的 target_APM32F402xx.py 模块。

（1） 在 BUILTIN_TARGETS 字典中添加一个新的键值对，键 "apm32f402xb" 是我们在命令行中使用的目标名称，值是对应的 APM32F402xB 类。

4 测试验证

完成以上步骤后，即可对内置支持进行测试。

4.1 查看目标列表

在命令行中执行以下命令，检查 APM32F402xb 是否出现在支持列表中：

```
pyocd list --targets
```

如果输出列表中包含 APM32F402xb，说明注册成功。

4.2 测试擦除

执行以下命令对芯片进行全片擦除：

```
pyocd erase --chip -t apm32f402xb
```

如果命令顺利执行且无报错，表明我们定制的 Flash 算法工作正常。

```
>pyocd erase --chip -t apm32f402xb
0001836 I Erasing chip... [eraser]
0002016 I Chip erase complete [eraser]
```

4.3 测试下载

使用一个简单的 .hex 或 .elf 固件文件（如 LED 闪烁程序）进行下载测试：

```
pyocd flash -t apm32f402xb path/to/your/firmware.hex
```

```
>pyocd flash -t apm32f402xb D:\Desktop\402\402_pyocd\402_pyocd_source\APM32F402_GPIO_Toggle.hex
0001574 I Loading D:\Desktop\402\402_pyocd\402_pyocd_source\APM32F402_GPIO_Toggle.hex [load_cmd]
[==================================================] 100%
0002210 I Erased 2048 bytes (2 sectors), programmed 2048 bytes (2 pages), skipped 0 bytes (0 pages) at 3.30 kB/s [loader
]
```

烧录成功后，观察开发板的运行现象，以确认固件是否正确写入并执行。至此，APM32F402 已成功集成到 pyOCD 中，可以脱离 Pack 文件进行开发和调试。

5 版本历史

表格 1 文件版本历史

日期	版本	变更历史
2025.08	1.0	新建

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，也不应被视为极海对第三方产品、服务或知识产权提供任何形式的保证，包括但不限于任何第三方知识产权的非侵权保证，除非极海在销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及/或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

极海产品并非设计、授权或担保适合用于军事、生命保障系统、污染控制或有害物质管理系统中的关键部件，亦非设计、授权或担保适合用于在产品失效或故障时可导致人员受伤、死亡、财产或环境损害的应用。

如果产品未标明“汽车级”，则表示不适用于汽车应用。如果用户对产品的应用超出极海提供的规格、应用领域、规范，极海不承担任何责任。

用户应该确保对产品的应用符合相应标准以及功能安全、信息安全、环境标准等要求。用户对极海产品的选择和使用负全部的责任。对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册及产品的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册及产品而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失），这涵盖了可能导致的人身安全、财产或环境损害等情况，对于这些损害极海概不承担责任。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。