

Application Note

Document No.: AN1161

APM32F402_F403_pyOCD Built-in Support

Version: V1.0

1 Introduction

When we need to add download/debug capabilities for a chip that has not been included in the built-in support list by the official pyOCD (or the community), the most common approach is to use the external support files of CMSIS-Pack or directly write a Target Python script in pyOCD.

This Application Note aims to introduce the second method in detail: how to deeply integrate the support for Geehy APM32F402/APM32F403 chips into pyOCD to make it a built-in recognizable target. In this way, users will no longer rely on external Pack files and can directly operate using simple command-line parameters (such as `-t apm32f402xb`), thereby simplifying the development process and facilitating team collaboration.

This application note applies to the Geehy APM32F402/APM32F403. The APM32F402 is used as an example throughout this document.

Contents

1	Introduction	1
2	Advantages and Ideas for Adding Built-in Support	3
3	Operation Steps	4
3.1	Early preparation	4
3.2	Parse the .FLM file and generate a Python algorithm script	4
3.3	Write target_APM32F402xx.py	4
3.4	Register the target model	6
4	Test and Verification	7
4.1	View the target list	7
4.2	Test erasure	7
4.3	Test download	7
5	Revision History	8

2 Advantages and Ideas for Adding Built-in Support

Adding support for the APM32F402/F403 directly to pyOCD mainly has the following advantages:

- Convenient team distribution and consistent environment: All team members use a customized pyOCD script, which can be stored in the version library for sharing, ensuring the unity and rapid deployment of the development environment.
- Support for more flexible customization and modification: Advanced debugging options such as sector size adjustment and configuration of protection bits can be flexibly adjusted in the script without being restricted by third-party Pack files.
- Deep learning of the internal mechanism of pyOCD: It helps to understand the working principle of pyOCD and lays a foundation for solving complex problems that may be encountered in the future.

The ideas for implementing this function are as follows:

- Extract the Flash programming algorithm file (.FLM) of the chip from the official SDK Pack.
- Use the generate_flash_algo.py script provided by pyOCD to parse the .FLM file into Python-format Flash algorithm data.
- Create a new Target definition file (e.g., target_APM32F402xx.py), integrate the generated algorithm data into it, and define the memory map of the chip (FlashRegion, RamRegion, etc.).
- Register the new target model in the builtin/__init__.py file of pyOCD so that it can be recognized by pyOCD.
- Finally, verify the success of the built-in support through actual erasing and downloading operations.

3 Operation Steps

3.1 Early preparation

- (1) Install Python and pyOCD

Ensure that a Python 3.7+ environment is installed locally, and install pyOCD via pip. It is recommended to use version 0.36.0 or higher.

```
pip install pyocd
```

- (2) Obtain the .FLM file of APM32F402

Find the Geehy.APM32F4xx_DFP.1.0.7.pack file in the official APM32F402_403_SDK_V1.0.1 from Geehy. After decompressing using a decompression tool (such as 7-Zip), search for the APM32F402_128.FLM file in the directory. This file is the core Flash algorithm file required for subsequent operations.

- (3) Prepare the generate_flash_algo.py script

This script is released along with the pyOCD source code. You can download the source code from the official GitHub repository of pyOCD (<https://github.com/pyocd/pyOCD>) and find the file in the scripts/ directory.

3.2 Parse the .FLM file and generate a Python algorithm script

Copy the APM32F402_128.FLM file to the directory where the generate_flash_algo.py script is located. Open the command line and navigate to this directory. Then execute the following command:

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

Note:

- (1) -o apm32f402_flash_algo.py: Specify the file name of the output Python script.
- (2) --ram-address=0x20000000: Specify the starting address of the MCU's RAM, which is used for the script to generate address-related code.
- (3) --stack-size=0x1000: The size of the stack space allocated for the Flash algorithm, which can be adjusted according to actual needs.
After the command is executed successfully, the apm32f402_flash_algo.py file will be generated, which contains information such as the instruction data and function entry points of the Flash algorithm.

3.3 Write target_APM32F402xx.py

Under the installation directory of pyOCD (for example, .../site-packages/pyocd/target/builtin), create a new Python file named target_APM32F402xx.py.

Copy the content of the FLASH_ALGO dictionary generated in the apm32f402_flash_algo.py file to the new file, and define the APM32F402xB class. An example is as follows:

```

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

# Content generated from generate_flash_algo.py
FLASH_ALGO = {
    'load_address': 0x20000000,
    'instructions': [
        0xe7fdba00, 0x4603b510,
        # ... (More hexadecimal instruction data is omitted here) ...
        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,=0x400,    # Usually programming granularity:
1KB
            is_boot_memory=True,
            algo=FLASH_ALGO
        ),
        RamRegion(
            start=0x20000000,
            Length=0x8000=0x8000      # For example, 32KB, or modify

```

```

according to the actual situation
    )
)

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

```

Note:

- (1) The start addresses and sizes of FlashRegion and RamRegion must be consistent with the actual specifications of the APM32F402 chip to avoid programming errors.
- (2) The VENDOR attribute should be set to "Geehy".

3.4 Register the target model

In order for pyOCD to recognize the newly added APM32F402xB class, you need to modify the pyocd/target/builtin/__init__.py file and register the new target model in the BUILTIN_TARGETS dictionary. Open the __init__.py file and add the following code:

```

# ... (Other existing imports)
from . import target_APM32F402xx

BUILTIN_TARGETS = {
    # ... (If there are other registered targets here, you can leave them
    unchanged)
    "apm32f402xb": target_APM32F402xx.APM32F402xB,
}

```

Note: First, import the target_APM32F402xx.py module we created.

- (1) Add a new key-value pair to the BUILTIN_TARGETS dictionary. The key "apm32f402xb" is the target name we use in the command line, and the value is the corresponding APM32F402xB class.

4 Test and Verification

After completing the above steps, you can test the built-in support.

4.1 View the target list

Execute the following command in the command line to check if APM32F402xb appears in the supported list:

```
pyocd list --targets
```

If the output list contains APM32F402xb, it indicates that the registration is successful.

4.2 Test erasure

Execute the following command to perform a full-chip erase on the chip:

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

If the command is executed smoothly without errors, it indicates that the customized Flash algorithm works properly.

```
C:\Program Files\Geehy>pyocd erase --chip -t apm32f402xb
0001836 I Erasing chip... [eraser]
0002016 I Chip erase complete [eraser]
```

4.3 Test download

Use a simple .hex or .elf firmware file (such as an LED blinking program) for the download test:

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

```
C:\Program Files\Geehy>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%
0002218 I Erased 2048 bytes (2 sectors), programmed 2048 bytes (2 pages), skipped 0 bytes (0 pages) at 3.30 kB/s [loader]
```

After successful programming, observe the operational behavior of the development board to verify whether the firmware has been correctly written and executed. At this point, the APM32F402 has been successfully integrated into pyOCD, and development and debugging can be carried out without the Pack file.

5 Revision History

Table 1 Revision History

Date	Version	Revision History
August, 2025	1.0	New

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party's products, services or intellectual property are involved in this manual, Geehy shall not be deemed to authorize users to use the aforesaid third party's products, services or intellectual property, nor shall it be deemed to provide any form of guarantee for

third-party products, services, or intellectual property, including but not limited to any non-infringement guarantee for third-party intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment are unavoidable. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall

agree and undertake to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy on an "as is" basis. To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy products are not designed, authorized, or guaranteed to be suitable for use as critical components in military, life support, pollution control, or hazardous substance management systems, nor are they designed, authorized, or guaranteed to be suitable for applications that may cause injury, death, property, or environmental damage in case of product failure or malfunction.

If the product is not labeled as "Automotive grade", it means it is not suitable for automotive applications. If the user's application of the product is beyond the specifications, application fields, and standards provided by Geehy, Geehy will assume no responsibility.

Users shall ensure that their application of the product complies with relevant standards, and the requirements of functional safety, information security, and environmental standards. Users are fully responsible for their selection and use of Geehy products. Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual and the products on an "as is" basis shall not be liable for damages, including any general or special direct, indirect or collateral damages arising from the use or no use of this manual and the products (including without limitation data loss or inaccuracy, or losses suffered by users or third parties), which cover damage to personal safety, property, or

environment, for which Geehy will not be responsible.

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.

©2025 Zhuhai Geehy Semiconductor Co., Ltd. All Rights Reserved