



STM32 Getting Started

Release 0.0.1

Cesar Bautista

Mar 18, 2025

CONTENTS

1	OpenOCD and PyOCD Overview	3
1.1	Introduction	3
1.2	OpenOCD (Open On-Chip Debugger)	3
1.3	PyOCD: A Python Interface to OpenOCD	3
1.4	ARM Cortex-M Debug Capabilities	3
1.5	Debugging Interface of ARM Cortex-M	3
1.6	Connection Through Debug Pins	4
1.7	CMSIS-DAP: A Standardized Debug Adapter	4
1.8	Overview of CMSIS-DAP	4
1.9	SWD Communication Protocols	4
1.10	Signals and Operations	4
1.11	Advantages of SWD Over JTAG	4
2	Using OpenOCD	5
2.1	Supported Microcontrollers	5
2.2	Selecting Your Microcontroller Target	5
3	STM32 microcontrollers	7
3.1	Getting Started with STM32	7
4	How to Generate an Error Report	9
4.1	Steps to Create an Error Report	9
4.2	Review and Follow-Up	9
5	Indices and tables	11

This documentation provides a comprehensive guide to getting started with ARM Cortex-M microcontrollers. It includes information on setting up the development environment, installing the necessary software, and writing code for ARM Cortex-M microcontrollers.

Note: This documentation is a work in progress.

OPENOCD AND PYOCD OVERVIEW

1.1 Introduction

1.2 OpenOCD (Open On-Chip Debugger)

OpenOCD is an open-source software tool that facilitates debugging, in-system programming, and boundary-scan testing of embedded devices. It supports a diverse range of microcontrollers including STM32, RP2040, and PY32. Through interfaces such as JTAG or SWD, OpenOCD provides a command-line interface for effective interaction with the target devices.

1.3 PyOCD: A Python Interface to OpenOCD

PyOCD simplifies the use of OpenOCD by providing a Python-based high-level interface. It offers a Python API that abstracts the complexities of the OpenOCD command-line interface, making it easier to program and debug microcontrollers. PyOCD is user-friendly, flexible, and designed to be extensible for various microcontroller interactions.

Tool	Functionality
OpenOCD	Provides a command-line interface for debugging and programming microcontrollers.
PyOCD	Offers a Python API for seamless interaction with microcontrollers via OpenOCD.

1.4 ARM Cortex-M Debug Capabilities

1.5 Debugging Interface of ARM Cortex-M

The ARM Cortex-M architecture includes a comprehensive debug interface that allows external tools to access the microcontroller core for debugging and programming tasks. This interface includes a suite of registers and commands that empower debuggers to halt the core, access memory, and manage the execution of programs.

1.6 Connection Through Debug Pins

Access to the debug interface is facilitated through specific pins on the microcontroller, such as SWDIO, SWCLK, and nRESET. These are linked to a debug adapter like the ST-Link or CMSIS-DAP, which forms the physical bridge between the host computer and the target device.

1.7 CMSIS-DAP: A Standardized Debug Adapter

1.8 Overview of CMSIS-DAP

CMSIS-DAP (Cortex Microcontroller Software Interface Standard - Debug Access Port) serves as a standardized debug adapter for ARM Cortex-M devices. It is endorsed by OpenOCD and provides a cost-effective solution compared to proprietary debug adapters.

1.9 SWD Communication Protocols

1.10 Signals and Operations

The SWDIO (Serial Wire Debug Input/Output) and SWCLK (Serial Wire Clock) signals are integral for communication over the SWD interface. SWDIO is a bidirectional line used for transmitting debug commands and data, while SWCLK is a clock signal that ensures synchronized data transfer between the host and the target.

1.11 Advantages of SWD Over JTAG

Utilizing a two-wire connection, the SWD interface minimizes the pin count required compared to the traditional JTAG interface. This reduction is particularly beneficial for microcontrollers where pin availability is limited.

USING OPENOCD

To program your microcontroller using OpenOCD, use the following command:

```
openocd -f interface/cmsis-dap.cfg -f target/rp2040.cfg \  
-c "init" -c "reset init" \  
-c "flash write_image erase blink.bin 0x100000000" \  
-c "reset run" -c "shutdown"
```

2.1 Supported Microcontrollers

This guide supports the following microcontrollers:

Microcontroller	Description
RP2040	Low-cost microcontroller with a dual-core ARM Cortex-M0+ processor.
STM32F103C8T6	Budget-friendly microcontroller with an ARM Cortex-M3 processor.
STM32F411	Cost-effective microcontroller with an ARM Cortex-M4 processor.

2.2 Selecting Your Microcontroller Target

Each microcontroller requires a specific configuration file. This file is a script that instructs OpenOCD on how to communicate with the microcontroller. The configuration file is tailored to the microcontroller and the debugger interface.

Microcontroller	Configuration File
RP2040	rp2040.cfg
STM32F103C8T6	stm32f103c8t6.cfg
STM32F411	stm32f411.cfg

STM32 MICROCONTROLLERS

STMicrontrollers is a family of 32-bit microcontrollers designed and manufactured by STMicroelectronics. They are widely used in embedded systems and IoT applications due to their low power consumption, high performance, and rich peripheral set. STM32 microcontrollers are based on the ARM Cortex-M architecture and are available in a wide range of series, each tailored to specific applications and requirements.

3.1 Getting Started with STM32

This documentation provides a comprehensive guide to getting started with STM32 microcontrollers. It includes information on setting up the development environment, installing the necessary software, and writing code for STM32 microcontrollers.

This technical documentation is a work in progress and is an adaptation of documentation from the STM32CubeIDE and STM32CubeMX software tools. It uses the ARM-GCC compiler and visual studio code as the IDE.

HOW TO GENERATE AN ERROR REPORT

This guide explains how to generate an error report using GitHub repositories.

4.1 Steps to Create an Error Report

1. Access the GitHub Repository

Navigate to the [GitHub repository](#) where the project is hosted.

2. Open the Issues Tab

Click on the “Issues” tab located in the repository menu.

3. Create a New Issue

- Click the “New Issue” button.
- Provide a clear and concise title for the issue.
- Add a detailed description, including relevant information such as:
 - Steps to reproduce the error.
 - Expected and actual results.
 - Any related logs, screenshots, or files.

4. Submit the Issue

Once the form is complete, click the “Submit” button.

4.2 Review and Follow-Up

The development team or maintainers will review the issue and take appropriate action to address it.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`