CH552 Multi-Protocol Programmer Getting Started

Release 0.0.1

Cesar Bautista

May 21, 2025

CONTENTS

Note: This documentation is a work in progress.

The **CH552 USB Multi-Protocol Programmer** is engineered for high-precision embedded system applications, supporting a variety of architectures including AVR, ARM (CMSIS-DAP), and CPLD (MAX II). It facilitates rigorous in-system programming, debugging, and boundary-scan testing, ensuring robust and reliable operation in complex development scenarios. Compatible with industry-standard microcontrollers such as STM32, RP2040, and PY32, this device integrates advanced hardware interfaces to support rapid prototyping and diagnostic processes.

This documentation provides comprehensive technical guidance for effectively deploying and interfacing with the **CH552 USB Multi-Protocol Programmer** in diverse embedded system environments.

CHAPTER

ONE

GENERAL INFORMATION

The **CH552 USB Multi-Protocol Programmer** is a compact and versatile development tool designed for high-precision embedded system applications. It supports a broad range of protocols and device architectures, including **AVR**, **ARM (CMSIS-DAP)**, and **CPLD (MAX II)**. Its USB connectivity enables direct interfacing with standard development environments, enabling:

- In-system programming (ISP)
- Step-through debugging
- Boundary-scan testing (JTAG)
- Flash memory operations

Compatible with popular platforms such as **STM32**, **RP2040**, and **PY32**, this programmer integrates configurable GPIO lines, multi-protocol headers, and streamlined power delivery—making it ideal for rapid prototyping and field diagnostics.

1.1 Supported Architectures

- **AVR** — via ISP and SPI
- **ARM Cortex-M** — via CMSIS-DAP and SWD
- **CPLD/FPGA (MAX II)** — via JTAG
- **RP2040** — via SWD and UF2 bootloader
- **PY32** — CMSIS-DAP, DFU, or UART

All protocols are exposed via labeled headers or JST connectors, allowing fast, solderless prototyping.

Note: GPIO numbers refer to the CH552 internal ports. Ensure correct firmware pin mapping before connecting external devices.

This device connects to a host system via USB and allows the user to program and debug various microcontrollers and programmable logic devices.

1.2 Supported Interfaces

- **JTAG**, for full-chip debugging and boundary scan
- **SWD**, for ARM Cortex-M series
- **SPI**, for flash and peripheral programming
- **UART**, for serial bootloaders and communication
- **GPIO**, for bit-banging or peripheral testing

1.3 GPIO Pins

> **Protocol SPI**

> **Protocol JTAG**

> **Protocol SWD**

1.4 Schematic Diagram

CHAPTER

TWO

OPENOCD AND PYOCD OVERVIEW

2.1 Introduction

2.1.1 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.

2.1.2 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.

2.2 ARM Cortex-M Debug Capabilities

2.2.1 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.

2.2.2 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.

2.3 CMSIS-DAP: A Standardized Debug Adapter

2.3.1 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.

2.4 SWD Communication Protocols

2.4.1 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.

2.4.2 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.

CHAPTER

THREE

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 0x10000000" \
        -c "reset run" -c "shutdown"
```

3.1 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.

3.1.1 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`

CHAPTER FOUR

PYOCD

```
pyocd erase -t py32f003x6 --chip --config ./Misc/pyocd.yaml
```

CHAPTER

FIVE

STM32 MICROCONTROLLERS

STMicontrollers 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.

5.1 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.

CHAPTER

SIX

HOW TO GENERATE AN ERROR REPORT

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

6.1 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.

6.2 Review and Follow-Up

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

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search