



[Handbook\(/handbook/\)](/handbook/)[Cookbook\(/cookbook/\)](/cookbook/)[Platforms\(/platforms/\)](/platforms/)[Components\(/components/\)](/components/)[Code\(/code/\)](/code/)[Questions\(/questions/\)](/questions/)[Forum\(/forum/\)](/forum/)[Dashboard\(/dashboard/\)](/dashboard/)[Compiler\(/compiler/\)](/compiler/)

# mbed(/)

Hi,

[sgstreet\(/users/sgstreet/\)](/users/sgstreet/)[Logout\(/account/logout/\)](/account/logout/)[Blog\(/blog/\)](/blog/) » [Debugging from GDB using pyOCD!](#)

## [Debugging from GDB using pyOCD!](/blog/entry/Debugging-from-GDB-using-pyOCD/) [\(/blog/entry/Debugging-from-GDB-using-pyOCD/\)](/blog/entry/Debugging-from-GDB-using-pyOCD/)

Posted 05 Apr 2013(05 Apr 2013), by  [\(/users/samux/\)](/users/samux/) **Samuel Mokrani(/users/samux/)**. [5 replies](#)  
[\(/blog/entry/Debugging-from-GDB-using-pyOCD/#commentform\)](/blog/entry/Debugging-from-GDB-using-pyOCD/#commentform)  [CMSIS-DAP\(/search/?q=CMSIS-DAP&type=Blog\)](/search/?q=CMSIS-DAP&type=Blog), [mbed\(/search/?q=mbed&type=Blog\)](/search/?q=mbed&type=Blog), [python\(/search/?q=python&type=Blog\)](/search/?q=python&type=Blog), [USB\(/search/?q=USB&type=Blog\)](/search/?q=USB&type=Blog)

We are pleased to release a python library which allows to drive the Debug Access Port of Cortex-M microcontrollers over CMSIS-DAP!

[Back to blog\(/blog/\)](/blog/)

About mbed

mbed is a tool for Rapid Prototyping with microcontrollers.

[Take the tour...\(/tour/\)](/tour/)

## What can be achieved with pyOCD?

- Debugging using GDB, as a gdbserver is integrated on the library
- Writing python applications that can communicate with the CMSIS-DAP and coresight debug interface:
  - read/write memory
  - read/write core registers
  - set breakpoints
  - flash new binary
  - run/stop/step the execution
- Act as a great reference to show how the CMSIS-DAP protocol works

Currently, the library works on Windows (using pyWinUSB as backend) and on Linux (using pyUSB as backend).

## Quick overview

### Use python to control your mbed platform

```

1 from pyOCD.board import MbedBoard
2
3 board = MbedBoard.chooseBoard()
4
5 target = board.target
6 flash = board.flash
7
8 target.resume()
9 target.halt()
10 print "pc: 0x%X" % target.readCoreRegister("pc")
11     pc: 0xA64
12
13 target.step()
14 print "pc: 0x%X" % target.readCoreRegister("pc")
15     pc: 0xA30
16
17 flash.flashBinary("binaries/l1_lpc1768.bin")
18
19 print "pc: 0x%X" % target.readCoreRegister("pc")
20     pc: 0x10000000
21
22 target.reset()
23 target.halt()
24 print "pc: 0x%X" % target.readCoreRegister("pc")
25     pc: 0xAAC
26
27 board.uninit()

```

## Use GDB to debug your mbed projects

Before using GDB, a .elf file has to be generated with a GCC toolchain.

- Python code to start a GDB server on port 3333

```

1 from pyOCD.gdbserver import GDBServer
2 from pyOCD.board import MbedBoard
3
4 board = MbedBoard.chooseBoard()
5
6 # start gdbserver on port 3333
7 gdb = GDBServer(board, 3333)

```

- Debug the target from GDB:

```

1 arm-none-eabi-gdb l1_lpc1768.elf
2
3 <gdb> target remote localhost:3333
4 <gdb> load
5 <gdb> continue

```

## Get Started

All the source code is available on our **git**

**repository**([https://github.com/mbedmicro/mbed/tree/master/workspace\\_tools/debugger](https://github.com/mbedmicro/mbed/tree/master/workspace_tools/debugger))

under **workspace\_tools/debugger**

You can quickly get started with pyOCD by reading the README. It provides all the information that you need to know concerning the dependencies, installation and how to use the library. There are even some sample programs to get started even quicker!

# Conclusion

pyOCD provides a simple and efficient solution to debug mbed platforms over CMSIS-DAP.

We expect quite soon the support of all the mbed platforms in OpenOCD as well. There is even a fork of OpenOCD adding CMSIS-DAP support: **[cmsis-dap support in OpenOCD\(https://github.com/TheShed/OpenOCD-CMSIS-DAP/tree/cmsis-dap\)](https://github.com/TheShed/OpenOCD-CMSIS-DAP/tree/cmsis-dap)**

Have fun with pyOCD!


g+1

2

Like


5


## 5 comments on Debugging from GDB using pyOCD!:

 (/users/ytsuboi/) Yoshihiro TSUBOI(/users/ytsuboi/)  
#(/comments/cr/27/281/#c5807) 10 Apr 2013(10 Apr 2013)

@Samuel Thanks for amazing works!! Many people asked me about CMSIS-DAP with gdb at Maker Faire Shenzhen. If I knew this, I could say "YES!!".


Why not supprting OS X? Is this because technical difficulties of HID handling on OS X?


 quote

 (/users/monaka/) Masaki Muranaka(/users/monaka/)  
#(/comments/cr/27/281/#c5856) 16 Apr 2013(16 Apr 2013)

@Samuel, thanks for your good product.


@ytsuboi, as you know, it's hard to support OSX because of technical issues around HID. I've not tried but probably it can support also OSX by preparing the another transport based on HIDAPI. I'll hack it on first of May if no one try.


 quote

 (/users/andyk1/) andy sellers(/users/andyk1/)  
#(/comments/cr/27/281/#c5897) 18 Apr 2013(18 Apr 2013)

Brilliant ! Thanks You very much for building this.

Are you planning on doing a video of the mbed with the DDD debugger ? If it works i think it could raise a lot of interest in the embedded community.

 quote

 (/users/timothyteh/) Timothy Teh(/users/timothyteh/)  
#(/comments/cr/27/281/#c6045) 04 May 2013(04 May 2013)

@Samuel: While I was trying out pyOCD, I found a bug in pywinusb\_backend.py. If there are more than 1 usb device (apart from the mbed) connected to the host computer, then inside the function getAllConnectedInterface, not all the non-mbed usb devices will be

removed from the *all\_devices* list because when device is removed, the index of items change and the iteration skips the next device it wants to remove. Hence, this will not remove a non-mbed device:

```
#keep devices with good vid/pid
for d in all_devices:
    if (d.vendor_id != vid) or (d.product_id != pid):
        all_devices.remove(d) #this will cause the next item's index to decrement and gets skipped
```

I fixed this by using a temporary list *temp\_devices* to remove the devices from it while iterating over *all\_devices*, and then *pointing all\_devices* to *temp\_devices* after removal:


```
#keep devices with good vid/pid
for d in all_devices:
    if (d.vendor_id != vid) or (d.product_id != pid):
        temp_devices.remove(d)


if not all_devices:
    logging.debug("No Mbed device connected")
    return

all_devices = list(temp_devices)
```


Other than that, pyOCD works great!

Cheers,  
Timothy Teh

 [quote](#)

 (/users/floha/) Flo rian(/users/floha/)  
[#\(/comments/cr/27/281/#c7307\)](#) 10 Sep 2013(10 Sep 2013)

I've used the python PyOCD gdb proxy. After uploading a file the KL25Z didn't react any more. Nothing, not even the MBED drive appears. I've tried to reflash the mbed\_if\_v2.0\_frdm\_kl25z.s19 File from windows: still dead. Unfortunately that happened to two (!) KL25Z of mine. So how can I get it alive again?

 [quote](#)

## Post a new comment

[Insert images or files](#)



▼ [Editing tips](#)

**Post comment**

**Show preview**

[\(<http://www.mbed.org>\)](http://www.mbed.org)

© mbed | [blog\(/blog\)](/blog/) | [get an mbed\(/order\)](/order/) | [about mbed\(/handbook/About\)](/handbook/About/) | [we're hiring!\(/handbook/Jobs\)](/handbook/Jobs/) | [support\(/handbook/Help\)](/handbook/Help/) | [service status\(/handbook/Service-status\)](/handbook/Service-status/) | [privacy policy\(/privacy\)](/privacy/) | [terms and conditions\(/terms\)](/terms/) | Language: [en\(/setlang?lang=en\)](/setlang?lang=en) [ja\(/setlang?lang=ja\)](/setlang?lang=ja) [es\(/setlang?lang=es\)](/setlang?lang=es) [de\(/setlang?lang=de\)](/setlang?lang=de)



mbed, the fastest way to create devices with ARM-based microcontrollers.

[^ back to top](#)