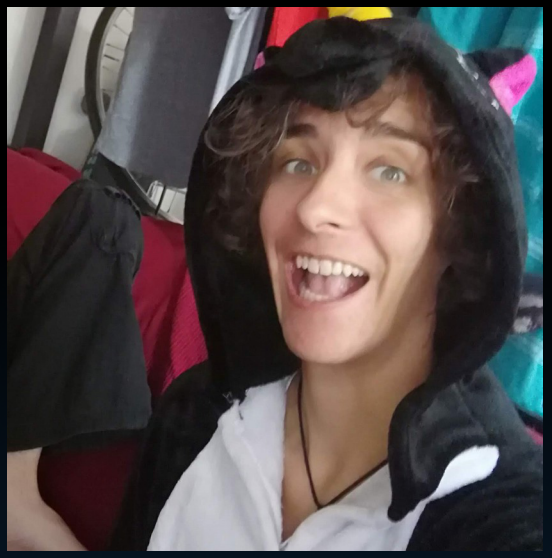
On-chip debugging with OpenOCD -for N00bs-



C00kie
@ko97551819

BSIDES MANCR3ST3R 2018

DC11331 PARIS

$ whoami

- Teacher for 10 years
- Climber
- 42born to code
- Threat intel analyst
- I bake gluten free cookies

OpenOCD

- Open On-Chip Debugger
- Open source and free
- Interfaces with the JTAG port, using a transfer protocol (Telnet, GDB)
- A lot of targets/interfaces .cfg files directly included
- Regularly maintained

JTAG ?
Joint Test Action Group

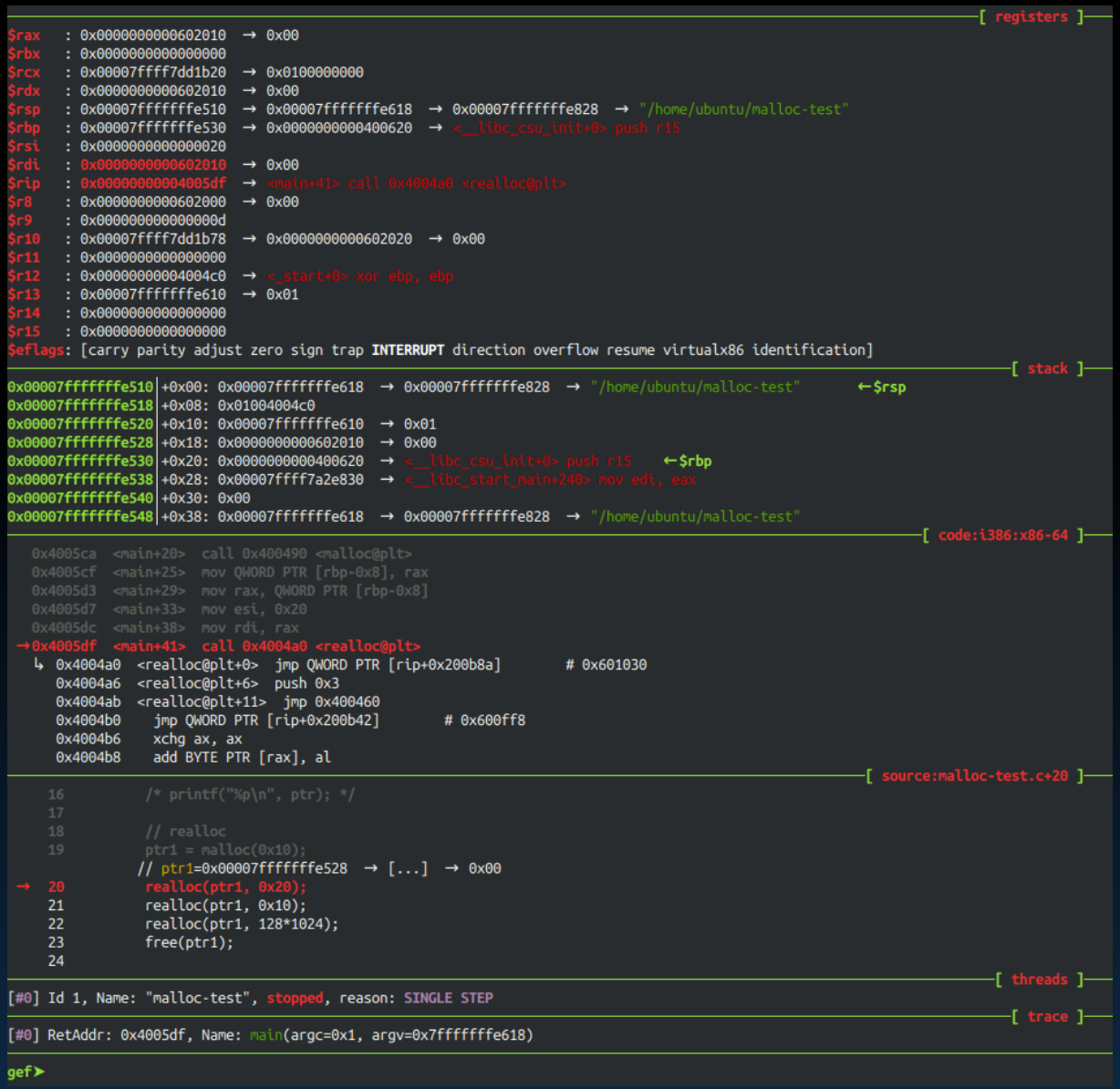
GDB + GEF

https://www.gnu.org/software/gdb/
GNU project debugger

GEF Ehanced features for GDB
https://github.com/hugsy/gef

Python API for using dynamic analysis

GDB/GEF


[registers]
$rax : 0x0000000000602010 → 0x00
$rbx : 0x0000000000000000
$rcx : 0x00007ffff7dd1b20 → 0x0100000000
$rdx : 0x0000000000602010 → 0x00
$rsp : 0x00007fffffffe510 → 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test"
$rbp : 0x00007fffffffe530 → 0x0000000000400620 → <__libc_csu_init+0> push r15
$rsi : 0x0000000000000020
$rdi : 0x0000000000602010 → 0x00
$rip : 0x00000000004005df → <main+41> call 0x4004a0 <realloc@plt>
$r8 : 0x0000000000602000 → 0x00
$r9 : 0x000000000000000d
$r10 : 0x00007ffff7dd1b78 → 0x0000000000602020 → 0x00
$r11 : 0x0000000000000000
$r12 : 0x00000000004004c0 → <_start+0> xor ebp, ebp
$r13 : 0x00007fffffffe610 → 0x01
$r14 : 0x0000000000000000
$r15 : 0x0000000000000000
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow resume virtualx86 identification]
[stack]
0x00007fffffffe510 +0x00: 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test" ←$rsp
0x00007fffffffe518 +0x08: 0x01004004c0
0x00007fffffffe520 +0x10: 0x00007fffffffe610 → 0x01
0x00007fffffffe528 +0x18: 0x0000000000602010 → 0x00
0x00007fffffffe530 +0x20: 0x0000000000400620 → <__libc_csu_init+0> push r15 ←$rbp
0x00007fffffffe538 +0x28: 0x00007ffff7a2e830 → <__libc_start_main+240> mov edi, eax
0x00007fffffffe540 +0x30: 0x00
0x00007fffffffe548 +0x38: 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test"
[code:i386:x86-64]
0x4005ca <main+20> call 0x400490 <malloc@plt>
0x4005cf <main+25> mov QWORD PTR [rbp-0x8], rax
0x4005d3 <main+29> mov rax, QWORD PTR [rbp-0x8]
0x4005d7 <main+33> mov esi, 0x20
0x4005dc <main+38> mov rdi, rax
→0x4005df <main+41> call 0x4004a0 <realloc@plt>
↳ 0x4004a0 <realloc@plt+0> jmp QWORD PTR [rip+0x200b8a] # 0x601030
0x4004a6 <realloc@plt+6> push 0x3
0x4004ab <realloc@plt+11> jmp 0x400460
0x4004b0 jmp QWORD PTR [rip+0x200b42] # 0x600ff8
0x4004b6 xchg ax, ax
0x4004b8 add BYTE PTR [rax], al
[source:malloc-test.c+20]
16 /* printf("%p\n", ptr); */
17
18 // realloc
19 ptr1 = malloc(0x10);
// ptr1=0x00007fffffffe528 → [...] → 0x00
→ 20 realloc(ptr1, 0x20);
21 realloc(ptr1, 0x10);
22 realloc(ptr1, 128*1024);
23 free(ptr1);
24
[threads]
[#0] Id 1, Name: "malloc-test", stopped, reason: SINGLE STEP
[trace]
[#0] RetAddr: 0x4005df, Name: main(argc=0x1, argv=0x7fffffffe618)
gef➤

GDB

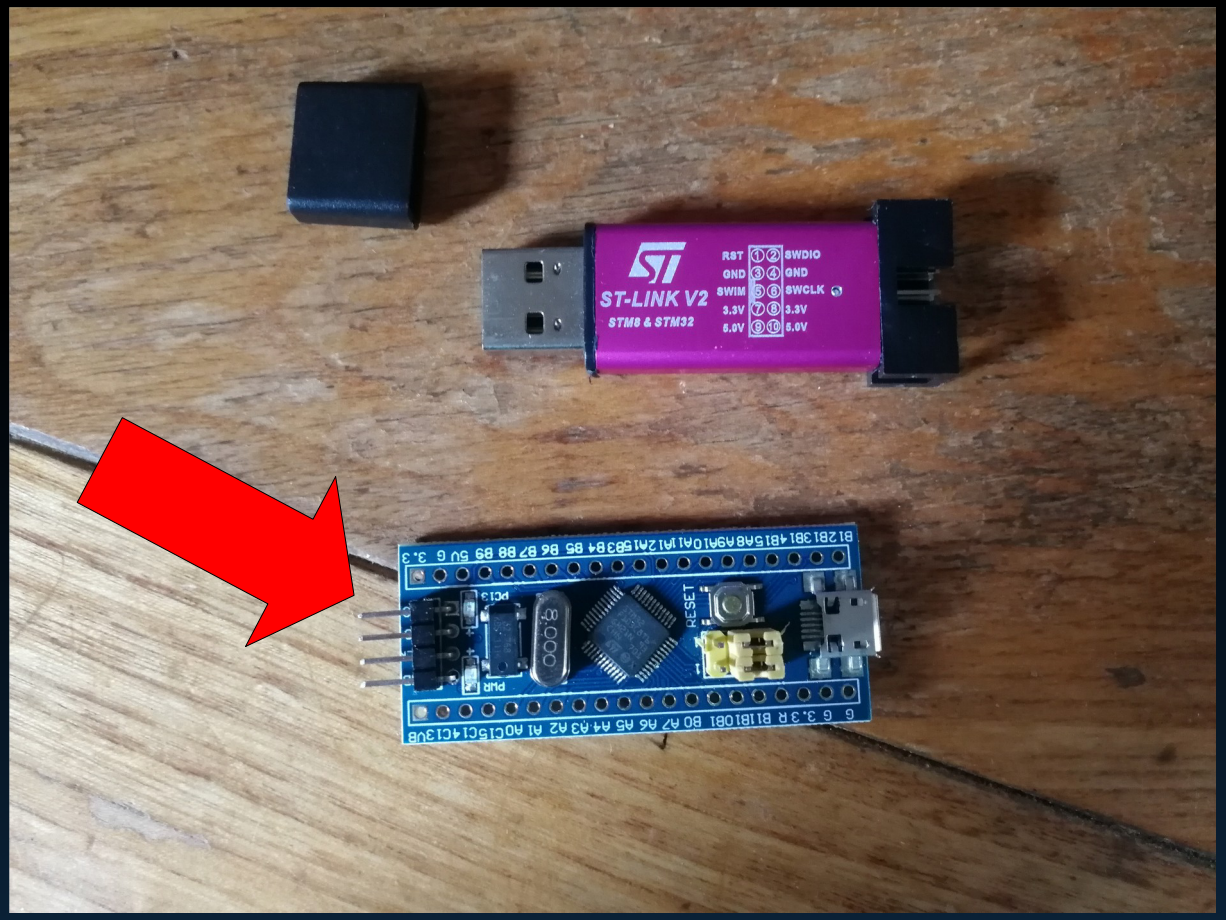
- Run code
- Disassemble
- Set breakpoints
- Inspect memory content
- Inspect registers content
- Set variables...

Getting started:

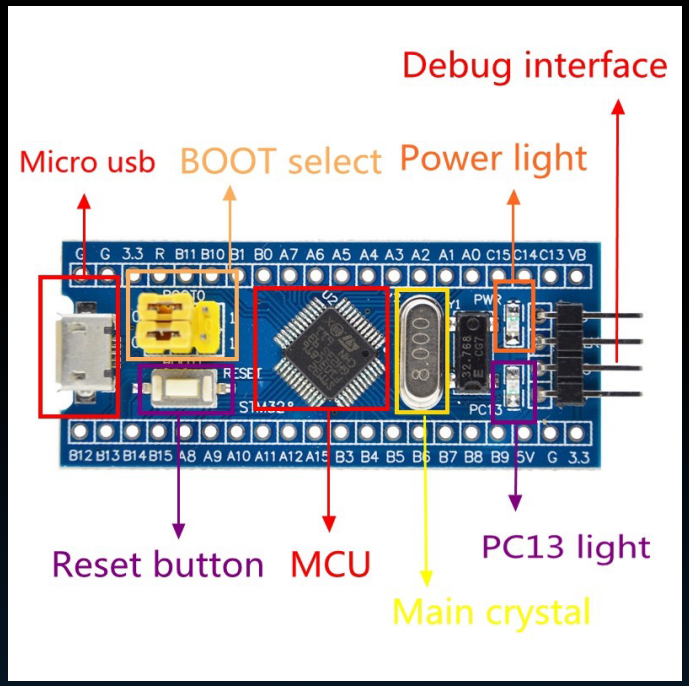
OpenOCD requires

- A debug adapter (ST-LINK USB dongle)
- A debug host (your machine) =-)
- A target (SMT32 microchip)

ST-Link v2/STM32F103

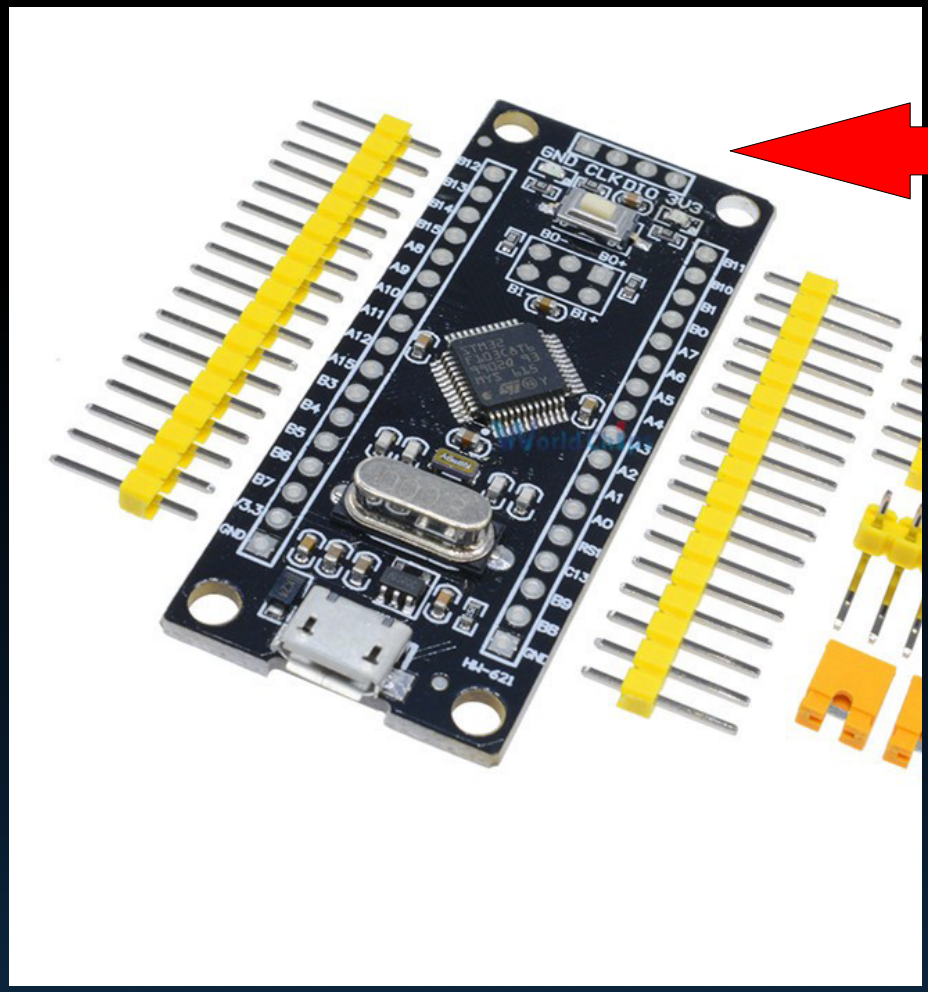


How to plug it to debug interface ?

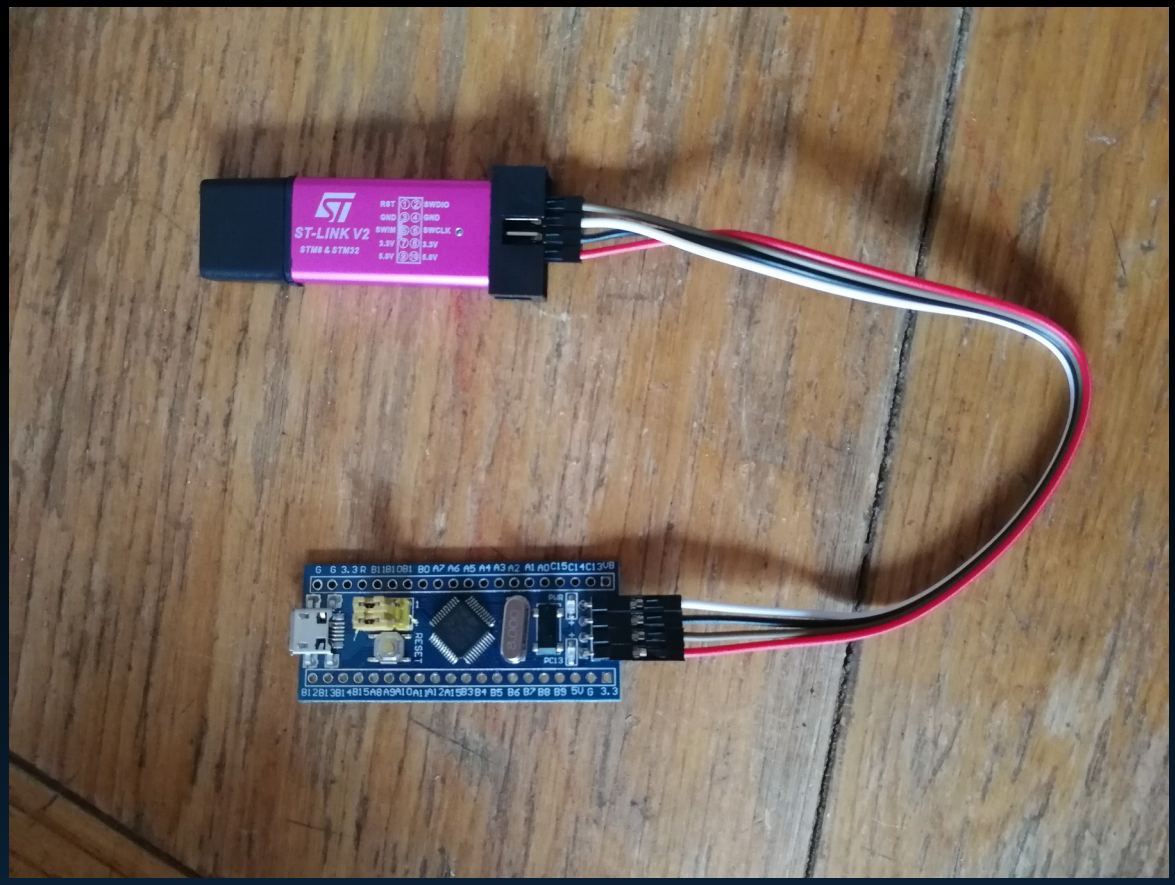


https://www.aliexpress.com/store/product/Free-Shipping-STM32F103C8T6-ARM-STM32-Minimum-System-Development-Board-Module-Forarduino/...

Pins ?



Ok !


ST-LINK V2
STM8 & STM32
RST
GND
SWIM
3.3V
5.0V
SWDIO
GND
SWCLK
3.3V
5.0V
RESET
PC13

*Configuration files *.cfg*

- In the main folder run sudo ./openocd
- Basic commands
  - $ sudo openocd -h
  - $ sudo openocd -s
  - $ sudo openocd -f [interface] -f[target]

Running

- sudo openocd -f interface/stlink-v2.cfg -f target/stm32f1x.cfg
- remote target localhost:3333
- c (continue)
- ^c (ctrl + c to break)
- i r (display about information registers)
- i f (display about information functions)
- s 4 (step 4 instructions)

demo

contact

- Cookie @ko97551819
- pokiji42@gmail.com

Thanks!