Flashing does not work in sleep modes, connect under reset should be used while resetting manually by grounding NRST.

This can be easily achieved by running what is in indicated in the make debug command source file:

Running openocd

Running make debug

Loading the binary in the make debug command line:

load ./Release/ReleaseStm32G4Static/CortexM4FApp.elf

continue

Then run make debug again to either debug or load

**INNER WORKINGS:**

Check FreertosConfig systick sources in m4 programming manual and explain how systick works in stm32

**INNER WORKINGS:**

[Lecture 9: Interrupts - YouTube](https://www.youtube.com/watch?v=uFBNf7F3l60&list=PLRJhV4hUhIymmp5CCeIFPyxbknsdcXCc8&index=9)

[Lecture 10: Interrupt Enable and Interrupt Priority - YouTube](https://www.youtube.com/watch?v=K0vmH2YGbOY&list=PLRJhV4hUhIymmp5CCeIFPyxbknsdcXCc8&index=10)

[Lecture 11: External interrupts (EXTI) - YouTube](https://www.youtube.com/watch?v=uKwD3JuRWeA&list=PLRJhV4hUhIymmp5CCeIFPyxbknsdcXCc8&index=11)

[Lecture 15: Booting Process - YouTube](https://www.youtube.com/watch?v=3brOzLJmeek&list=PLRJhV4hUhIymmp5CCeIFPyxbknsdcXCc8&index=15)

Booting from RAM is done through modifying linker file to load code into RAM and issuing a reset.

startup\_stm32g431xx.s incorporates different sections:

.text.Reset\_Handler

.text.Default\_Handdler

.isr\_vector

The text linker STM32G431CBTX\_FLASH.ld puts sections either on RAM or flash:

.isr\_vector:{….}>FLASH

.text:{}>FLASH

.rodata:{….}>FLASH

.data:{}>RAM AT>FLASH

.bss:{}>RAM

**MAKEFILE**

Compiling:

Set architecture:

Cpu type

Floating type

Linking against nano

The makefile needs to have a bash compatible interface (command line tool capable of executing linux commands) which has added to PATH:

1. Compiler & related tools: GNU Tools ARM Embedded
2. Flasher: stlink, stm32cubeProgrammer
3. Debugger: openocd

The makefile need gdb-multiarch installed

Directories:

OpenOCD: contains stlink.cfg stm32g4x.cfg as interface and targets for openocd, swj-dp.tcl for SWDIO support and mem\_helper.tcl for openocd to work

How to program?

Via openocd, stm32cubeprogrammer, st-flash

**2.- HOW TO DEBUG**

**Openocd** is a command line tool found at: [Open On-Chip Debugger (openocd.org)](https://openocd.org/)

To debug we need to call openocd **binary** with an **interface** and a **target** while selecting a **communication** **protocol** like swd. Interfaces and targets are textfiles contained in the scripts folder in windows installation directory.

Example:

openocd -f D:/Program\_Files/xpack-openocd-0.10.0-15/scripts/interface/stlink-v2.cfg -c "transport select hla\_swd" -f D:/Program\_Files/xpack-openocd-0.10.0-15/scripts/target/stm32g4x.cfg

xPack OpenOCD, x86\_64 Open On-Chip Debugger 0.10.0+dev (2020-10-13-17:29)

Licensed under GNU GPL v2

For bug reports, read

http://openocd.org/doc/doxygen/bugs.html

WARNING: interface/stlink-v2.cfg is deprecated, please switch to interface/stlink.cfg

hla\_swd

Info : The selected transport took over low-level target control. The results might differ compared to plain JTAG/SWD

Info : Listening on port 6666 for tcl connections

Info : Listening on port 4444 for telnet connections

Info : clock speed 2000 kHz

Info : STLINK V2J37S7 (API v2) VID:PID 0483:3748

Info : Target voltage: 3.241434

Info : stm32g4x.cpu: hardware has 6 breakpoints, 4 watchpoints

Info : starting gdb server for stm32g4x.cpu on 3333

Info : Listening on port 3333 for gdb connections

After that a server running on localhost:3333 will be waiting for gdb connections.

**2.2 Debugging process**:

1. For debugging the application position at the top directory **“EncoderBoard-G431”** and execute:

openocd -f ./OpenOCD/stlink.cfg -c "transport select hla\_swd" -f ./OpenOCD/stm32g4x.cfg

1. Then call make debug in another terminal

Use hexdump to see hex file

Semihosting intialise monitor handles() Serial wire viewer

No syscalls with –specs=nosys.specs Runtine library –specs=nano.specs

-specs nano

How to compile?

CMSIS, HAL, libopencm3

[microcontroller - Can I use ST-Link programmer for non-ST chips? - Electrical Engineering Stack Exchange](https://electronics.stackexchange.com/questions/250664/can-i-use-st-link-programmer-for-non-st-chips)

[Getting Started With STM32 and Nucleo Part 3: FreeRTOS - How To Run Multiple Threads w/ CMSIS-RTOS - YouTube](https://www.youtube.com/watch?v=OPrcpbKNSjU)

Elf format

https://www.youtube.com/watch?v=nC1U1LJQ

Debugging topics

[How to debug a HardFault on an ARM Cortex-M MCU | Interrupt (memfault.com)](https://interrupt.memfault.com/blog/cortex-m-hardfault-debug)

Masters can initiate communication

**Order of includes:**

Separated by sections:

Freertos first

Clib last

**3.- STM32CUBE\_IDE**

1.- Set filepath for firmware installation repository (CMSIS, HAL, BSP, OpenBootloader, USB, FatFS, FreeRTOS)

Library:

DMA memory addresses cannot and should not be hard coded because command sequences exists which are saved in different arrays which make changing from memory address necessary. Periphereal address should stay stable.

RTOS, timer task priority should be the lowest so when sending to a queue it should find receiving task queues empty and so, it will not block the daemon task