Minimum Port Wiki

The purpose of this document is to explain how to alter Cleanflight's Makefile to compile a simple example program for the HiFive1 Rev. B. Board.

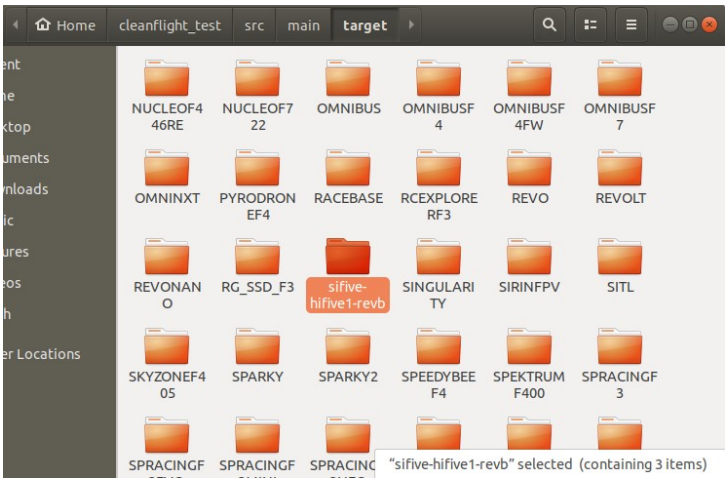
You will need to have installed the Freedom-E-SDK (available using the command: *git clone --recursive https://github.com/sifive/freedom-e-sdk.git*) and GNU's RISC-V Toolchain (available using the command: *git clone --recursive https://github.com/riscv/riscv-gnu-toolchain).* See the 'hifive1b getting started' guide for more information.

You will also need to have already compiled one of SiFive's example programs as a standalone project (such as the 'hello' program). To do this, run the following command inside the freedom-e-sdk directory: *make standalone BSP=metal PROGRAM=hello TARGET=sifive-hifive1-revb STANDALONE_DEST=desired-standalone-dir*.

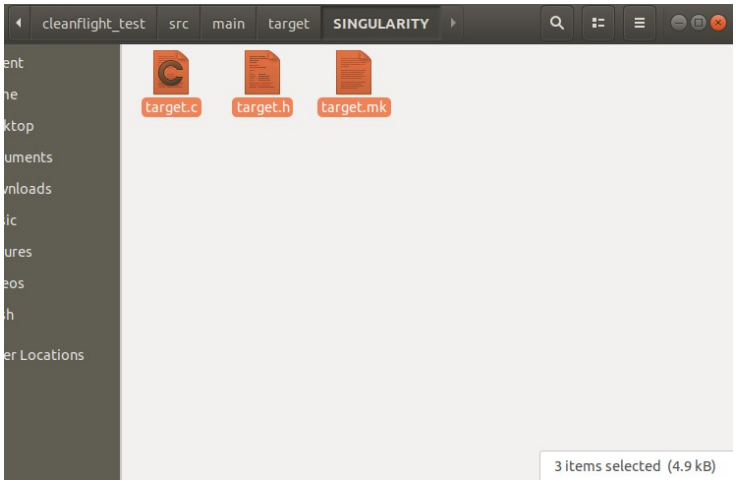
Steps:

1. First, we need to add the HiFive1 Rev. B board to the VALID_TARGETS variable found in targets.mk. To do this, follow the steps below:

i. Create a folder in /src/main/target called: sifive-hifive1-revb.

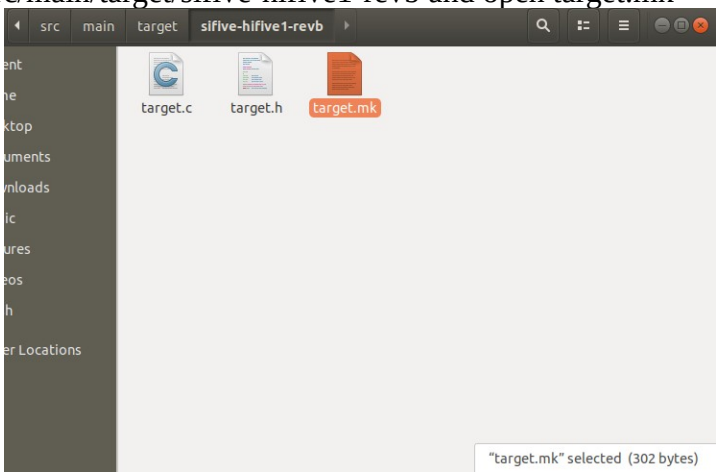


ii. Copy the target.c, target.h, and target.mk from another target's folder in /src/main/target. (I suggest using files from the SINGULARITY folder because it contains a very simple target.mk).

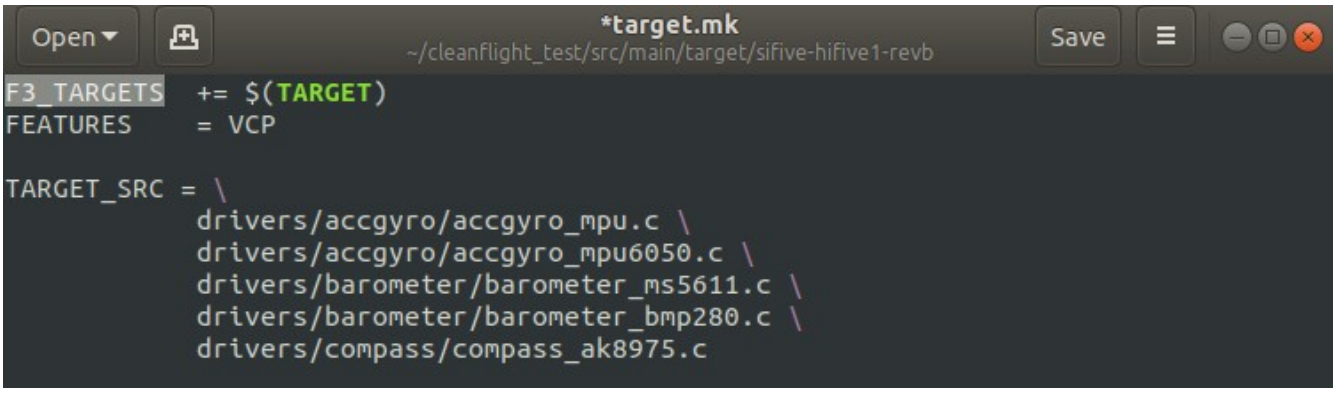


2. Next, we need to make a target group specifically for the HiFive1-Rev. B. To do this, follow the steps below:

i. Navigate to /src/main/target/sifive-hifive1-revb and open target.mk



ii. Inside, at the top, there will be a line that says: "F3_TARGETS += $(TARGET)"
Change the left side of this assignment to: SIFIVE_TARGETS.



iii. For consistency, also add SIFIVE_TARGETS to the ifeq statement at line 82 of targets.mk, and add 'SIFIVE' to the error message. It should look like:

```
ifeq ($(filter $(TARGET),$(F1_TARGETS) $(F3_TARGETS) $(F4_TARGETS) $(F7_TARGETS) $
(SITL_TARGETS) $(SIFIVE_TARGETS)),)
$(error Target '$(TARGET)' has not specified a valid STM group, must be one of F1,
F3, F405, F411, F7x5, or SIFIVE. Have you prepared a valid target.mk?)
endif
```

3. Next, we need to assign the TARGET_MCU variable in targets.mk appropriately based on the HiFive1 Rev. B target. To do this, open targets.mk located in ./make/, then follow the steps below:

i. Add the following else ifeg statement directly below line 100:

else ifeq ($(TARGET),$(filter $(TARGET), $(SIFIVE_TARGETS)))
TARGET_MCU := SIFIVE1REVB

It should look like:

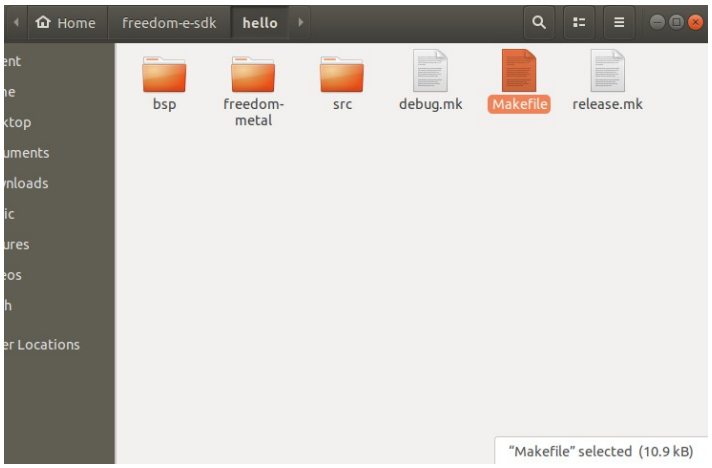
```
else ifeq ($(TARGET),$(filter $(TARGET), $(F1_TARGETS)))
TARGET_MCU := STM32F1

else ifeq ($(TARGET),$(filter $(TARGET), $(SIFIVE_TARGETS)))
TARGET_MCU := SIFIVE1REVB

else
$(error Unknown target MCU specified.)
endif
```

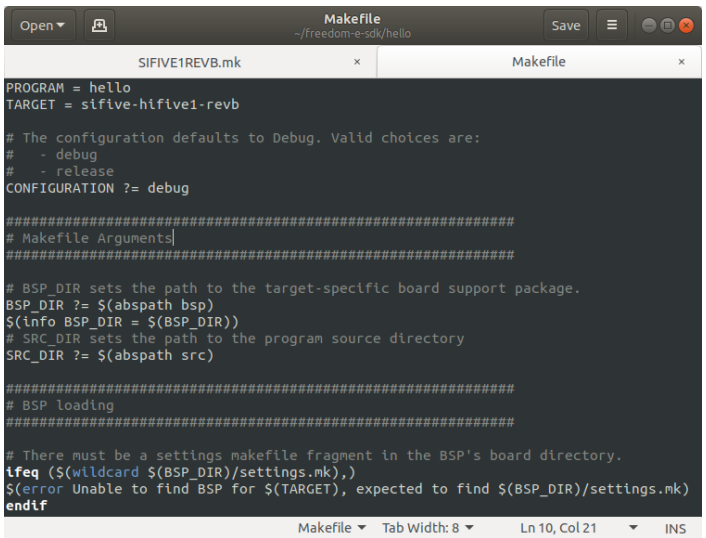
4. Next, we're going to create an MCU Makefile specifically for our HiFive1 Rev. B target. (You can see several MCU Makefiles in ./make/mcu/ already exist for STM32 microcontrollers that support Cleanflight).

Note: The name of this MCU Makefile *must* be equal to value of TARGET_MCU, shown above. So we will name it: SIFIVE1REVB.mk.

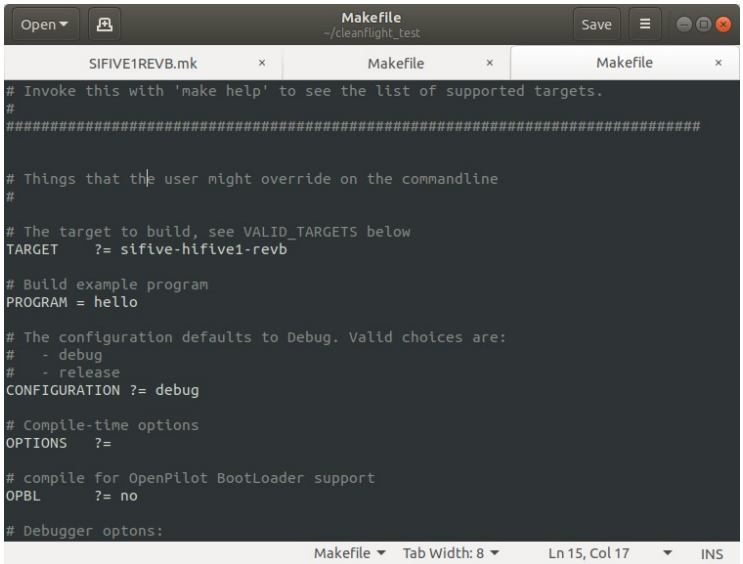
To create and populate SIFIVE1REVB.mk follow the steps below:

i. Open the Makefile of your previously compiled SiFive example program. (In this case, the 'hello' program). From here on, this Makefile will be referred to as the SiFive Makefile.



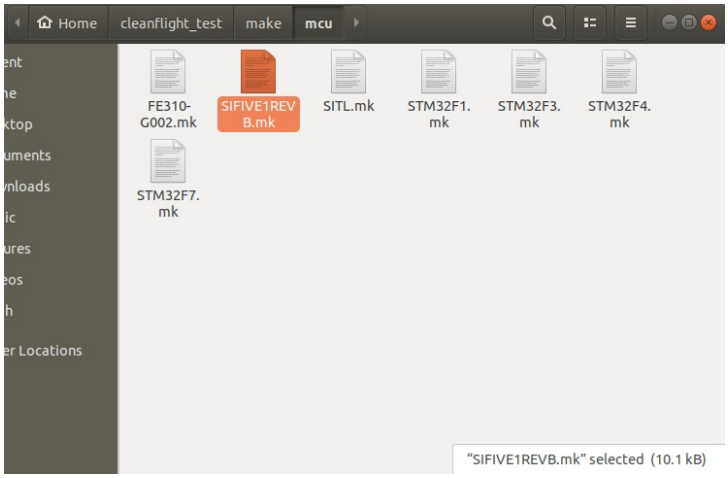


ii. Copy/paste the variables PROGRAM, TARGET, CONFIGURATION from the SiFive Makefile into Cleanflight's Makefile, as seen below:



Note: Code from Cleanflight's Makefile *and* SiFive's Makefile both use the variable TARGET. For SiFive's Makefile code to work properly, TARGET *must* equal: sifive-hifive1-revb. Cleanflight's Makefile will work with this assignment *as long as* the HiFive board's target file in ./src/main/target is also called: sifive-hifive1-revb.

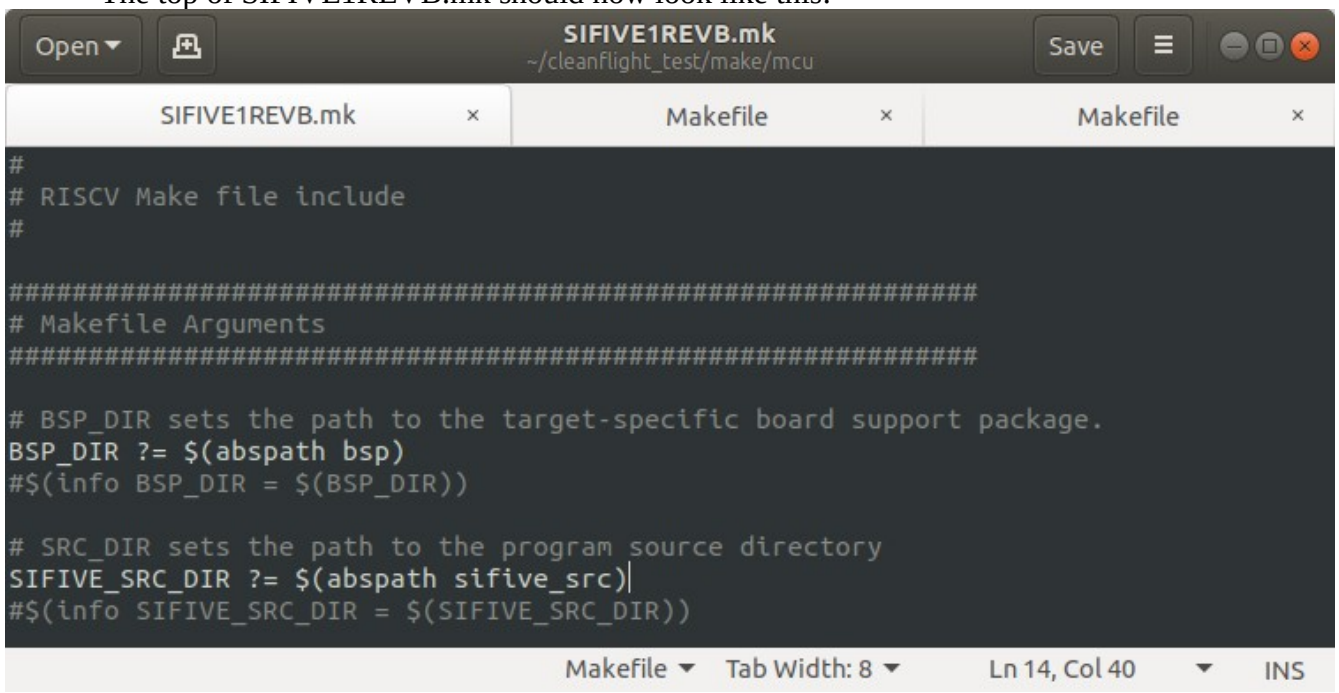
iii. Copy an existing STM32 MCU Makefile from ./make/mcu/ and save it in the same location. Rename it: SIFIVE1REVB.mk.



iv. Copy "Makefile Arguments" section from SiFive's Makefile, and paste it into SIFIVE1REVB.mk.

Note: Since Cleanflight's Makefile already uses the variable SRC_DIR, the SRC_DIR variable in the "Makefile Arguments" section must be renamed something different. Let's call it: SIFIVE_SRC_DIR. We must also change the value of SIFIVE_SRC_DIR, as you'll see next step. Let's call it: sifive_src.

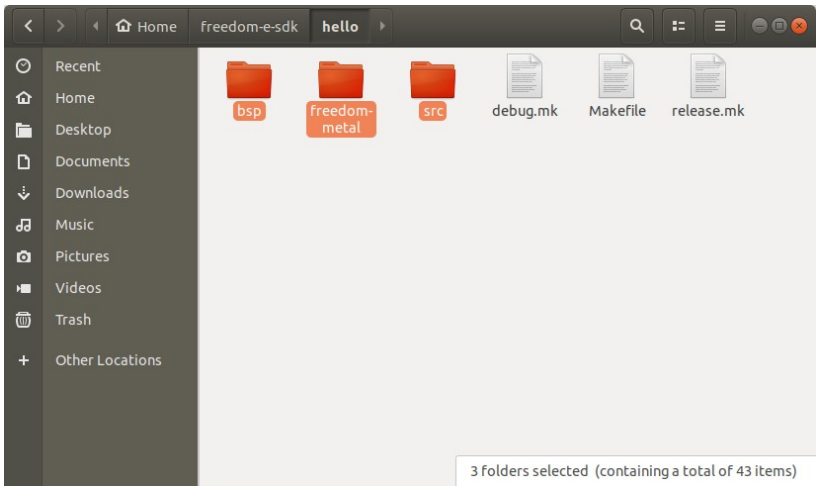
The top of SIFIVE1REVB.mk should now look like this:

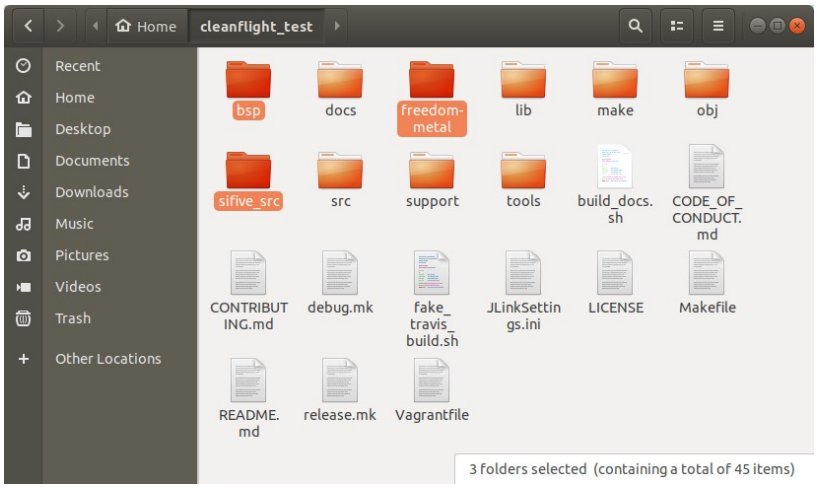


v. Next, copy the bsp, freedom-metal, and src folders of your SiFive example program (located at ./freedom-e-sdk/hello), and paste them into the top level of your Cleanflight directory.

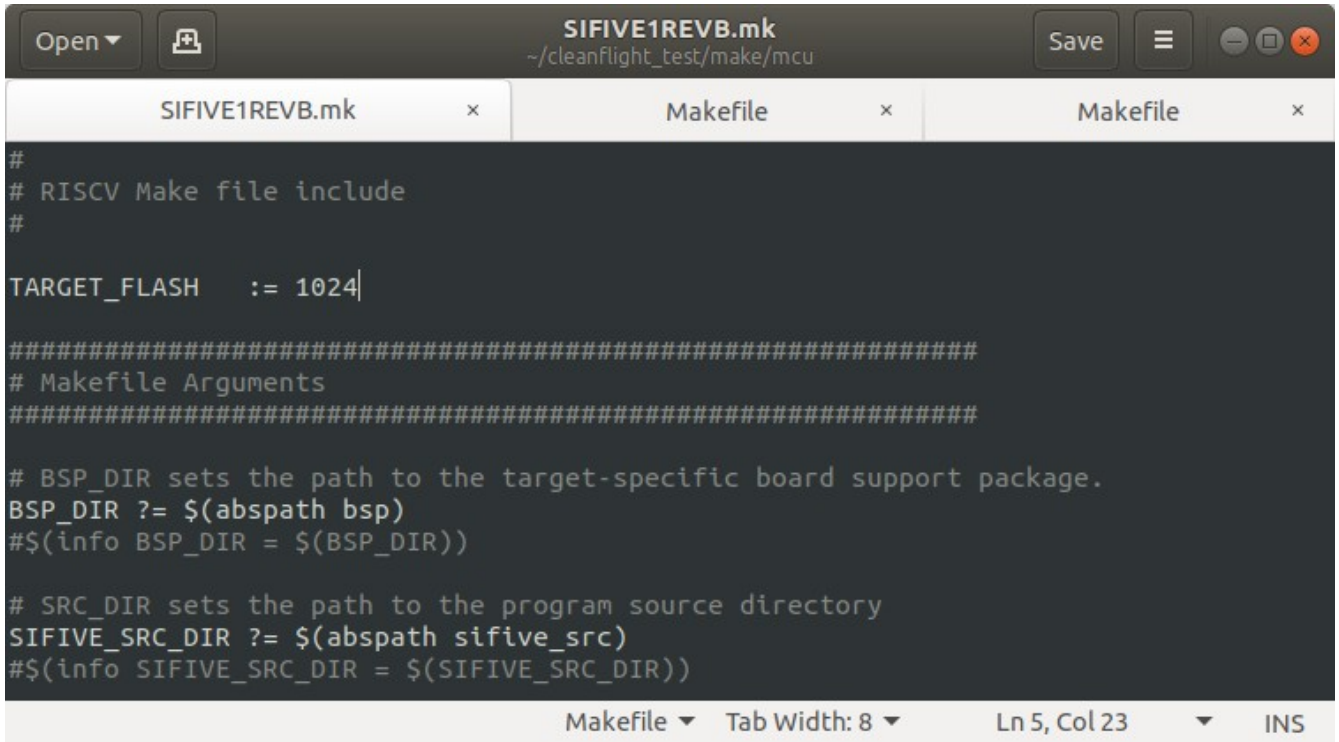
Note: Since a “src” folder already exists in the top-level Cleanflight directory, you will be prompted to rename the src folder (this is why we changed the value of SIFIVE_SRC_DIR in the previous step). Change the name of the incoming src folder to: sifive_src.

See the screenshots below:

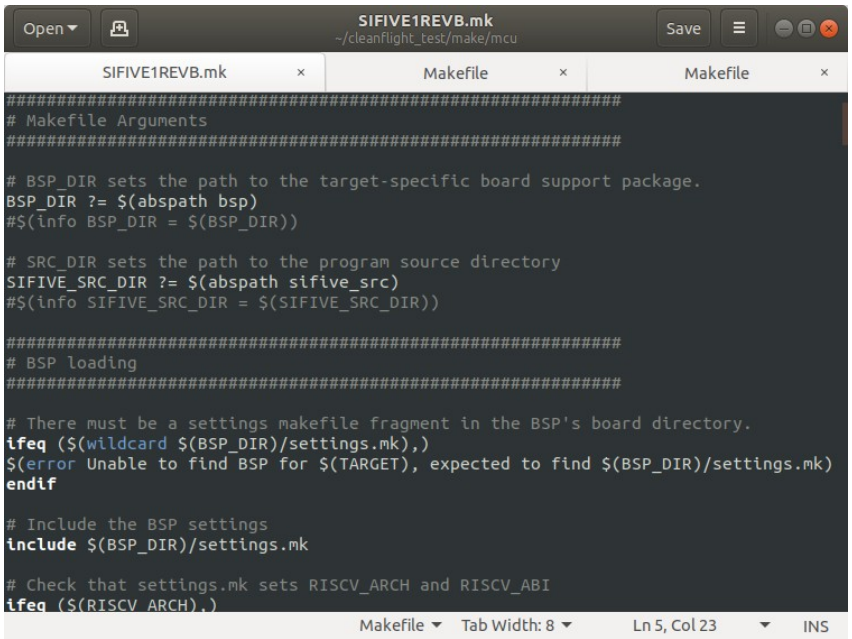




vi. Next, above “Makefile Arguments” section, specify the HiFive’s program memory size by entering the line: TARGET_FLASH := 1024.



vii. Next, copy the “BSP Loading” and “Software Flags” sections from the SiFive Makefile, and paste them below the “Makefile Arguments” in SIFIVE1REVB.mk. See the screenshots below:



```
##############################################################
# Software Flags
##############################################################

# Set the arch, ABI, and code model
RISCV_CCASFLAGS += -march=$(RISCV_ARCH) -mabi=$(RISCV_ABI) -mcmodel=$(RISCV_CMODEL)
RISCV_CFLAGS    += -march=$(RISCV_ARCH) -mabi=$(RISCV_ABI) -mcmodel=$(RISCV_CMODEL)
RISCV_CXXFLAGS  += -march=$(RISCV_ARCH) -mabi=$(RISCV_ABI) -mcmodel=$(RISCV_CMODEL)
RISCV_ASFLAGS   += -march=$(RISCV_ARCH) -mabi=$(RISCV_ABI) -mcmodel=$(RISCV_CMODEL)

# Prune unused functions and data
RISCV_CFLAGS    += -ffunction-sections -fdata-sections
RISCV_CXXFLAGS  += -ffunction-sections -fdata-sections

# Include the Metal headers
RISCV_CCASFLAGS += -I$(abspath $(BSP_DIR)/install/include/)
RISCV_CFLAGS    += -I$(abspath $(BSP_DIR)/install/include/)
RISCV_CXXFLAGS  += -I$(abspath $(BSP_DIR)/install/include/)

# Use newlib-nano
RISCV_CCASFLAGS += --specs=nano.specs
RISCV_CFLAGS    += --specs=nano.specs
RISCV_CXXFLAGS  += --specs=nano.specs

# Turn on garbage collection for unused sections
```

viii. Finally, you can delete the original Cleanflight code at the bottom of the file, or block it with a conditional statement, as seen below.

Note: For the purposes of the Minimum Port, this original Cleanflight code is not needed.

```
##############################################################
# Original Cleanflight code below
##############################################################

ifneq ($(TARGET_MCU), SIFIVE1REVB)
STDPERIPH_DIR   = $(ROOT)/lib/main/STM32F7/Drivers/STM32F7xx_HAL_Driver
STDPERIPH_SRC   = $(notdir $(wildcard $(STDPERIPH_DIR)/Src/*.c))
EXCLUDES        = stm32f7xx_hal_can.c \
                  stm32f7xx_hal_cec.c \
                  stm32f7xx_hal_crc.c \
                  stm32f7xx_hal_crc_ex.c \
                  stm32f7xx_hal_cryp.c \
                  stm32f7xx_hal_cryp_ex.c \
                  stm32f7xx_hal_dcmi.c \
                  stm32f7xx_hal_dcmi_ex.c \
                  stm32f7xx_hal_dfsdm.c \
                  stm32f7xx_hal_dma2d.c \
                  stm32f7xx_hal_dsi.c \
                  stm32f7xx_hal_eth.c \
                  stm32f7xx_hal_hash.c \
                  stm32f7xx_hal_hash_ex.c \
                  stm32f7xx_hal_hcd.c \
                  stm32f7xx_hal_i2s.c \
                  stm32f7xx_hal_irda.c \
                  stm32f7xx_hal_iwdg.c \
```

5. Next, we are going to migrate the RISC-V toolchain names and directories over to the top-level Cleanflight Makefile.

i. Copy and paste the “Toolchain” section of the SiFive Makefile into the top-level Cleanflight Makefile. (I recommend pasting it somewhere around Cleanflight’s toolchain section for organizational purposes, but the location doesn’t matter).

```
#


ifeq ($(TARGET_MCU), SIFIVE1REVB)

##############################################################
# RISC-V Toolchain
##############################################################

# Allow users to select a different cross compiler.
CROSS_COMPILE ?= riscv64-unknown-elf

# If users don't specify RISCV_PATH then assume that the tools will just be in
# their path.
ifeq ($(RISCV_PATH),)
RISCV_GCC     := $(CROSS_COMPILE)-gcc
RISCV_GXX     := $(CROSS_COMPILE)-g++
RISCV_OBJDUMP := $(CROSS_COMPILE)-objdump
RISCV_OBJCOPY := $(CROSS_COMPILE)-objcopy
RISCV_GDB     := $(CROSS_COMPILE)-gdb
RISCV_AR      := $(CROSS_COMPILE)-ar
RISCV_SIZE    := $(CROSS_COMPILE)-size
else
RISCV_GCC     := $(abspath $(RISCV_PATH)/bin/$(CROSS_COMPILE)-gcc)
RISCV_GXX     := $(abspath $(RISCV_PATH)/bin/$(CROSS_COMPILE)-g++)
RISCV_OBJDUMP := $(abspath $(RISCV_PATH)/bin/$(CROSS_COMPILE)-objdump)
```

6. Next, we are going to migrate the SiFive Makefile's rules over to the top-level Cleanflight Makefile. These rules will tell the make utility how to accomplish certain tasks (such as building an executable file).

i. Copy and paste the following sections of the SiFive Makefile into the top-level Cleanflight Makefile: "Software," "elf2hex," "Compiles an instance of Metal targeted at $(TARGET)". (I recommend pasting them directly above Cleanflight's rules). See screenshots below:

```
##############################################################
# Software
##############################################################

PROGRAM_ELF ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).elf
PROGRAM_HEX ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).hex
PROGRAM_LST ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).lst

.PHONY: all
all: software

.PHONY: software
software: $(PROGRAM_ELF)

software: $(PROGRAM_HEX)
```

Software section

```
Open ▾   Makefile   ~/freedom-e-sdk/hello   Save
##############################################################
# elf2hex
##############################################################
scripts/elf2hex/build/Makefile: scripts/elf2hex/configure
        @rm -rf $(dir $@)
        @mkdir -p $(dir $@)
        cd $(dir $@); \
                $(abspath $<) \
                --prefix=$(abspath $(dir $<))/install \
                --target=$(CROSS_COMPILE)

scripts/elf2hex/install/bin/$(CROSS_COMPILE)-elf2hex: scripts/elf2hex/build/
Makefile
        $(MAKE) -C $(dir $<) install
        touch -c $@

.PHONY: clean-elf2hex
Makefile ▾   Tab Width: 8 ▾   Ln 281, Col 26   INS
```

elf2hex section

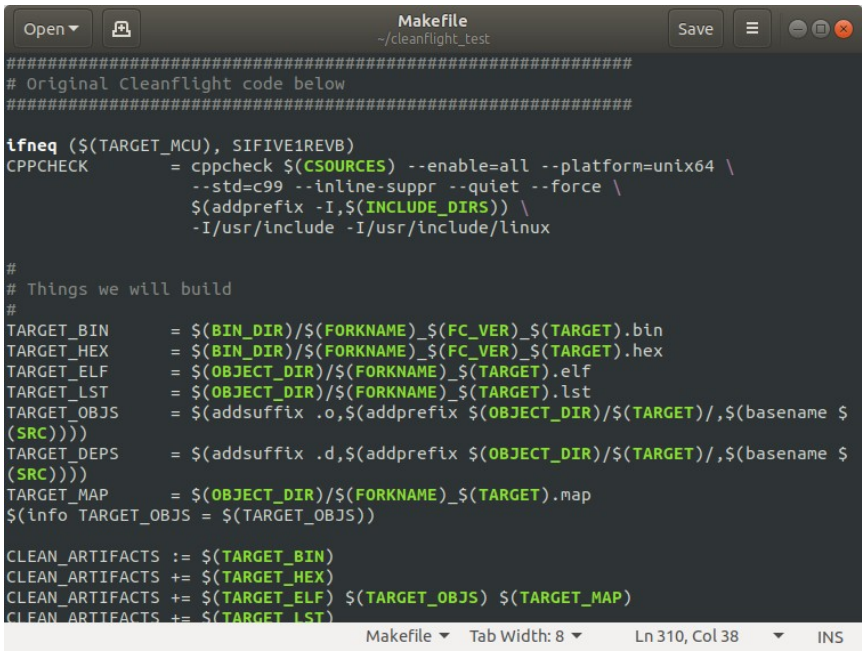
```
Open ▾   Makefile   ~/freedom-e-sdk/hello   Save

##############################################################
# Compiles an instance of Metal targeted at $(TARGET)
##############################################################
METAL_SOURCE_PATH ?= freedom-metal
METAL_LDSCRIPT     = $(BSP_DIR)/metal.$(LINK_TARGET).lds
METAL_HEADER       = $(BSP_DIR)/metal.h
METAL_INLINE       = $(BSP_DIR)/metal-inline.h
PLATFORM_HEADER    = $(BSP_DIR)/metal-platform.h

METAL_PREFIX       = $(abspath $(BSP_DIR)/install)
METAL_BUILD_DIR    = $(abspath $(BSP_DIR)/build/$(CONFIGURATION))
METAL_LIB_DIR      = $(abspath $(BSP_DIR)/install/lib/$(CONFIGURATION))

.PHONY: metal
metal: $(METAL_LIB_DIR)/stamp
Makefile ▾   Tab Width: 8 ▾   Ln 281, Col 26   INS
```

Metal section

ii. We only want to see these rules if the HiFive board is the target, so we will enclose the above sections in one big conditional statement: ifeq ($(TARGET_MCU), SIFIVE1REVB), as seen below:

```
Open ▾   Makefile   ~/cleanflight_test   Save

ifeq ($(TARGET_MCU), SIFIVE1REVB)

##############################################################
# Software
##############################################################

PROGRAM_ELF ?= $(SIFIVE_SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).elf
PROGRAM_HEX ?= $(SIFIVE_SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).hex
PROGRAM_LST ?= $(SIFIVE_SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).lst

.PHONY: all
all: software

.PHONY: software
software: $(PROGRAM_ELF)

software: $(PROGRAM_HEX)

PROGRAM_SRCS = $(wildcard $(SIFIVE_SRC_DIR)/*.c) $(wildcard $(SIFIVE_SRC_DIR)/*.h)
$(wildcard $(SIFIVE_SRC_DIR)/*.S)

$(PROGRAM_ELF): \
                $(PROGRAM_SRCS) \
                $(BSP_DIR)/install/lib/$(CONFIGURATION)/libmetal.a \
                $(BSP_DIR)/install/lib/$(CONFIGURATION)/libmetal-gloss.a \
                $(BSP_DIR)/metal.$(LINK_TARGET).lds
        mkdir -p $(dir $@)
Makefile ▾   Tab Width: 8 ▾   Ln 310, Col 38   INS
```

iii. If the HiFive board is the target, we also *don't* want to see the rules for STM32 targets. So, we will enclose Cleanflight's original rules in a ifneq statement: ifneq ($(TARGET_MCU), SIFIVE1REVB), as shown below:




```
##############################################################
# Original Cleanflight code below
##############################################################

ifneq ($(TARGET_MCU), SIFIVE1REVB)
CPPCHECK        = cppcheck $(CSOURCES) --enable=all --platform=unix64 \
                  --std=c99 --inline-suppr --quiet --force \
                  $(addprefix -I,$(INCLUDE_DIRS)) \
                  -I/usr/include -I/usr/include/linux

#
# Things we will build
#
TARGET_BIN      = $(BIN_DIR)/$(FORKNAME)_$(FC_VER)_$(TARGET).bin
TARGET_HEX      = $(BIN_DIR)/$(FORKNAME)_$(FC_VER)_$(TARGET).hex
TARGET_ELF      = $(OBJECT_DIR)/$(FORKNAME)_$(TARGET).elf
TARGET_LST      = $(OBJECT_DIR)/$(FORKNAME)_$(TARGET).lst
TARGET_OBJS     = $(addsuffix .o,$(addprefix $(OBJECT_DIR)/$(TARGET)/,$(basename $
(SRC))))
TARGET_DEPS     = $(addsuffix .d,$(addprefix $(OBJECT_DIR)/$(TARGET)/,$(basename $
(SRC))))
TARGET_MAP      = $(OBJECT_DIR)/$(FORKNAME)_$(TARGET).map
$(info TARGET_OBJS = $(TARGET_OBJS))

CLEAN_ARTIFACTS := $(TARGET_BIN)
CLEAN_ARTIFACTS += $(TARGET_HEX)
CLEAN_ARTIFACTS += $(TARGET_ELF) $(TARGET_OBJS) $(TARGET_MAP)
CLEAN_ARTIFACTS += $(TARGET_LST)
```

iv. Now, a few lines of transferred code in the "Software" section will contain directory addresses that need to be updated (since we changed SiFive's SRC_DIR variable to SIFIVE_SRC_DIR). Refer to the two screenshots below for which addresses need to be updated with the new SIFIVE_SRC_DIR variable:

Open ▾ Makefile ~/freedom-e-sdk/hello Save

```
##############################################################
# Software
##############################################################

PROGRAM_ELF ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).elf
PROGRAM_HEX ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).hex
PROGRAM_LST ?= $(SRC_DIR)/$(CONFIGURATION)/$(PROGRAM).lst

.PHONY: all
all: software

.PHONY: software
software: $(PROGRAM_ELF)

software: $(PROGRAM_HEX)

PROGRAM_SRCS = $(wildcard $(SRC_DIR)/*.c) $(wildcard $(SRC_DIR)/*.h) $(wildcard $(SRC_DIR)/*.S)

$(PROGRAM_ELF): \
                $(PROGRAM_SRCS) \
                $(BSP_DIR)/install/lib/$(CONFIGURATION)/libmetal.a \
                $(BSP_DIR)/install/lib/$(CONFIGURATION)/libmetal-gloss.a \
                $(BSP_DIR)/metal.$(LINK_TARGET).lds
        mkdir -p $(dir $@)
        $(MAKE) -C $(SRC_DIR) $(basename $(notdir $@)) \
                PORT_DIR=$(PORT_DIR) \
                AR=$(RISCV_AR) \
                CC=$(RISCV_GCC) \
                CXX=$(RISCV_GXX) \
                ASFLAGS="$(RISCV_ASFLAGS)" \
                CCASFLAGS="$(RISCV_CCASFLAGS)" \
                CFLAGS="$(RISCV_CFLAGS)" \
                CXXFLAGS="$(RISCV_CXXFLAGS)" \
                XCFLAGS="$(RISCV_XCFLAGS)" \
                LDFLAGS="$(RISCV_LDFLAGS)" \
                LDLIBS="$(RISCV_LDLIBS)"
        mv $(SRC_DIR)/$(basename $(notdir $@)).map $(dir $@)
        mv $(SRC_DIR)/$(basename $(notdir $@)) $@
        touch -c $@
        $(RISCV_OBJDUMP) --source --all-headers --demangle --line-numbers --wide $@ > $(PROGRAM_LST)
        $(RISCV_SIZE) $@

# Use elf2hex if we're creating a hex file for RTL simulation
ifneq ($(filter rtl,$(TARGET_TAGS)),)
.PHONY: software
$(PROGRAM_HEX): \
                scripts/elf2hex/install/bin/$(CROSS_COMPILE)-elf2hex \
                $(PROGRAM_ELF)
        $< --output $@ --input $(PROGRAM_ELF) --bit-width $(COREIP_MEM_WIDTH)
else
$(PROGRAM_HEX): \
```

Makefile ▾ Tab Width: 8 ▾ Ln 207, Col 34 ▾ INS

```
$(PROGRAM_HEX): \
                scripts/elf2hex/install/bin/$(CROSS_COMPILE)-elf2hex \
                $(PROGRAM_ELF)
        $< --output $@ --input $(PROGRAM_ELF) --bit-width $(COREIP_MEM_WIDTH)
else
$(PROGRAM_HEX): \
                $(PROGRAM_ELF)
        $(RISCV_OBJCOPY) -O ihex $(PROGRAM_ELF) $@
endif


.PHONY: clean-software
clean-software:
        $(MAKE) -C $(SRC_DIR) PORT_DIR=$(PORT_DIR) clean
        rm -rf $(SRC_DIR)/$(CONFIGURATION)
.PHONY: clean
clean: clean-software
```

7. Finally, we have a few cleanup tasks to ensure Cleanflight's Makefile executes correctly with the HiFive1 as the target.

i. Move the 'include tools.mk' instruction directly below the 'include targets.mk' instruction.

ii. Put the following ifneq statement around the 'include tools.mk' instruction: ifneq ($ (TARGET_MCU), SIFIVE1REVB). Step i and ii will now prevent the 'include tools.mk' instruction from executing when the target is the HiFive1 board.

```
FEATURES        =

include $(ROOT)/make/targets.mk

ifneq ($(TARGET_MCU), SIFIVE1REVB)
# include the tools makefile
include $(ROOT)/make/tools.mk
endif
```

iii. Finally, enclose the two .DEFAULT_GOAL rules in Cleanflight's Makefile in the ifneq statement: ifneq ($ (TARGET_MCU), SIFIVE1REVB). The Makefile's execution will snag if these rules are present when the target is not an STM32.

Open ▾ Makefile ~/cleanflight_test Save

```
ifneq ($(TARGET_MCU), SIFIVE1REVB)
ifeq ($(OPBL),yes)
TARGET_FLAGS := -DOPBL $(TARGET_FLAGS)
.DEFAULT_GOAL := binary
else
.DEFAULT_GOAL := hex
endif
endif
```

Makefile ▾ Tab Width: 8 ▾ Ln 100, Col 1 ▾ INS