

Project Overview:

Goal: Port **Contiki-NG** to the **EFR32ZG28B312F1024IM48** (Silicon Labs Wireless Gecko Series 2) board and verify hardware functionality (UART, LEDs, etc.).

- ✓ Contiki-NG does not directly support the EFR32ZG28 board. But we can make it work by writing our own driver files for the board and adding them to Contiki-NG.
- ✓ This means creating code to control things like UART, timers, GPIO pins, and also using startup and system files from the Gecko SDK.
- ✓ After we set up these files and configure them properly, we can compile Contiki-NG for the EFR32ZG28. We also need to make a platform Makefile, use the correct linker script, and set the right memory addresses for the chip.
- ✓ If we write all the needed drivers and set up the build environment correctly, the board should be able to run Contiki-NG.

step 1: To set the ubuntu to install the contiki-ng os .

- `sudo apt update`
- `sudo apt install git build-essential python3 python3-pip python3-setuptools python3-wheel srecord doxygen`
- `sudo apt install gcc-arm-none-eabi binutils-arm-none-eabi gdb-multiarch //GNU Arm Embedded Toolchain`

Step2: Install the contiki os

- `git clone https://github.com/contiki-ng/contiki-ng.git`
- `cd contiki-ng`

- `git submodule update --init --recursive` // It helps to Fetch submodules like `OS, examples, tools`
- `arch/` → CPU & platform architecture support
- `examples/` → Sample apps (hello-world, coap, etc.)
- `os/` → Contiki-NG core (kernel, networking)
- `tools/` → Helper scripts & utilities

step 3: To write the board support files like a

`contiki-ng/arch/platform/gecko/efr32zg28b312f1024im48/` // This location to write my board support files .

- To write a CPU file to manually to write in the contiki-ng os for the `efr32zg28b312f1024im48` board support

step 4: To get the board support files from Simplicity Studio

- Startup & System Files
`platform/Device/SiliconLabs/EFR32ZG28/Source`
- `startup_efr32zg28.S` → Assembly file that sets up the stack pointer, calls `main()`
- `system_efr32zg28.c` → Configures system clock, oscillators, and power modes

Device Header Files

`platform/Device/SiliconLabs/EFR32ZG28/Include/`
`em_device.h` → MCU register definitions

- `efr32zg28b312f1024im48.h` → Chip-specific defines (RAM, Flash sizes, IRQ numbers)
- `core_cm33.h` → ARM Cortex-M33 core register defines

Linker Script:

`platform/Device/SiliconLabs/EFR32ZG28/Source/GCC/`

`efr32zg28.ld` → Defines Flash start address, RAM size, vector table location

If not present, you can take the linker script from a Simplicity Studio project generated for your board.

The Flash address start like a

FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 1024K

RAM (rwx) : ORIGIN = 0x20000000, LENGTH = 256K

How to Extract Files from Simplicity Studio

- create a new simplicity studio project
- Select:
 - **Board:** EFR32ZG28B312F1024IM48
 - **SDK:** Gecko SDK v3.x.x
 - **Toolchain:** GCC
- Choose “**Empty C Project**” or “Blink LED” (simplest).
- Generate code

To add a peripheral, open the `.slcp` file and click on **Software Components**, then search for whatever you need (e.g., UART) and install it from there.

- Find generated project files
- `autogen/` → Auto-generated pin/board configs

- `config/` → Peripheral initialization configs
- `src/` → Your code
- `.slcp` → Project configuration file

Copy needed files to Contiki-NG

➤ `cp`

`~/SimplicityStudio/SDKs/gecko_sdk_3/platform/Device/SiliconLabs/EF
R32ZG28/Source/startup_efr32zg28.S`

`~/Documents/contiki-ng/arch/platforms/efr32zg28/`

➤ `cp`

`~/SimplicityStudio/SDKs/gecko_sdk_3/platform/Device/SiliconLabs/EF
R32ZG28/Source/system_efr32zg28.c`

`~/Documents/contiki-ng/arch/platforms/efr32zg28/`

➤ `cp ~/SimplicityStudio/SDKs/gecko_sdk_3/platform/emlib/src/em_gpio.c`

`~/Documents/contiki-ng/arch/platforms/efr32zg28/`

➤ `cp ~/SimplicityStudio/SDKs/gecko_sdk_3/platform/emlib/src/em_uart.c`

`~/Documents/contiki-ng/arch/platforms/efr32zg28/`

`cd ~/Documents/phytec_week2_tasks/contiki-ng/examples/uart-write-test`

this directory contains the main.c file and make file

Step 5: to build the project

`make TARGET=gecko BOARD=efr32zg28b312f1024im48-board`

If the build file like a

`~/Documents/phytec_week2_tasks/contiki-ng/examples/uart-test-write/`

`build/gecko/efr32zg28b312f1024im48-board$ ls`

`obj uart-test-write.gecko`

It contains the .gecko build file now to flash to need like a .hex or .bin file to convert a .hex file

step 6: the contiki-ng use the ARM GCC toolchain

```
arm-none-eabi-objcopy -O ihex uart-write-test.gecko uart-write-test.hex
```

now to get the .hex file

step 7: To flash the file using

➤ Simplicity Commander

[illegible]

->

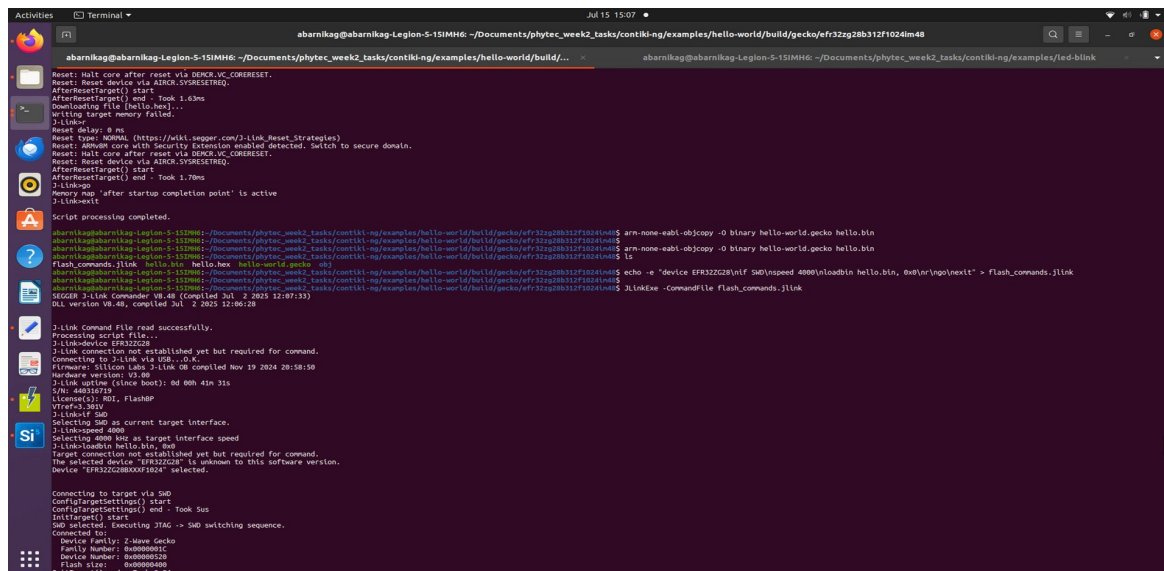
**Downloads/SimplicityStudio-5/SimplicityStudio_v5/developer/adapte
r_packs/commander/commander flash**

```
~/Documents/phytec_week2_tasks/contiki-ng/examples/uart-write-  
test/build/gecko/efr32zg28b312f1024im48-board/uart-write-test.hex
```

```
--device EFR32ZG28B312F1024IM48
```

To use this command to flash

➤ J-Link Commander



```
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/...
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/...
Reset: Halt core after reset via DENDR_V0_CORERESET.
Reset: Reset device via AFRCR_SYSAESCTREQ.
AfterResetTarget() start
AfterResetTarget() end - Took 1.43ms
Downloading file [hello.hex]...
Writing target memory failed.
J-LinkError
Reset delay: 5 ms
Reset type: NORMAL (https://wiki.segger.com/J-Link_Reset_Strategies)
Reset: Halt core after reset via DENDR_V0_CORERESET.
Reset: Reset device via AFRCR_SYSAESCTREQ.
AfterResetTarget() start
AfterResetTarget() end - Took 1.70ms
J-LinkInfo
Memory map "after startup completion point" is active
J-LinkError
Script processing completed.
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$ arm-none-eabi-objcopy -O binary hello-world.peco hello.bin
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$ arm-none-eabi-objcopy -O binary hello-world.peco hello.bin
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$
Flash_commands.jlink hello.bin hello.hex hello-world.gecko obj
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$ echo -e "device EFR32G28B312F1024im48" > flash_commands.jlink
abarnikag@abarnikag-Legion-5-15IMH6: ~/Documents/pyhtec_week2_tasks/contiki-ng/examples/hello-world/build/gecko/efr32g28b312f1024im48$
SEGGER J-Link Commander V8.48 (Compiled Jul 2 2025 12:07:33)
DLL version V8.48, compiled Jul 2 2025 12:06:28

J-Link Command File read successfully.
Processing script file...
J-Link-device EFR32G28B
J-Link connection not established yet but required for command.
Connecting to J-Link via USB...O.K.
Firmware: Silicon Labs J-Link OB compiled Nov 19 2024 20:58:50
Hardware version: V9.00
J-Link uptime (since boot): 0J 00h 41m 31s
S/N: 440310719
License(s): RTU, FlashBP
VTarget: 3.30V
J-LinkSWD
J-LinkSWD
Selecting SWD as current target interface.
J-LinkSpeed 4000
Selecting 4000 kHz as target interface speed
J-LinkLoadbin hello.bin, not
Target connection not established yet but required for command.
The selected device "EFR32G28B312F1024im48" is unknown to this software version.
Device "EFR32G28B312F1024im48" selected.

Connecting to target via SWD
ConfigTargetSettings() start
InitTarget() start
SWD selected, executing JTAG -> SWD switching sequence.
Connected to:
Device family: Z-Wave Gecko
Family number: 0x0000001C
Device number: 0x00000020
Flash size: 0x00000040
InitTarget() end - Took 3.54ms
```

I am using the commander to flash .

Step 8: After flash i want to try to get output in the minicom
`minicom -D /dev/ttyACM0 -b 115200`

I successfully flashed the firmware, but it doesn't seem to actually flash on the device. I don't know what the error is because no error message is shown.

I tried to print a character using loopback through a UART TTL converter, but it did not work.