# Contents

## 1. Objectives

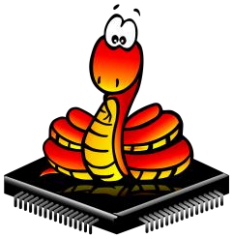
- Configure environment for Zephyr project
- Clone the MicroPython from git repo and configure environment
- Build and Flash Zephyr a demo using Terminal
- Build and Flash MicroPython application using Terminal
- Build and Flash Zephyr project using Eclipse

## 2. Hardware

- VEGAboard (rv32m1_vega)
- 2 Micro-USB cables
- Debug probe (Guide uses J-Link EDU mini)
- Internet access

## 3. Software

- Toolchain (Prebuilt GCC and OpenOCD) for Linux - https://open-isa.org/downloads
- Zephyr project - https://www.zephyrproject.org
- MicroPython repo - https://github.com/micropython
- GNU MCU Eclipse IDE - https://github.com/gnu-mcu-eclipse
- Segger J-Link software for Linux - https://www.segger.com

## 4. Prepare Linux environment for VEGAboard SDK

The instructions from this guide are executed on a Linux Terminal (Ubuntu). Refer to https://open-isa.org for Windows environment instructions.

Install some required Linux software packages:

```
sudo apt install curl openjdk-11-jre snapd
```

Note: You may need to update your software database ('apt-get update').

### 4.1 Download SDK and Toolchain

```
curl -L https://github.com/open-isa-org/open-isa.org/releases/download/1.0.0/rv32m1_sdk_riscv_installer.sh > \
$HOME/rv32m1_sdk_riscv_installer.sh

curl -L https://github.com/open-isa-org/open-isa.org/releases/download/1.0.0/Toolchain_Linux.tar.gz > \
$HOME/Toolchain_Linux.tar.gz
```

### 4.2 Extract SDK

Extract and install SDK.

```
cd $HOME
chmod +x rv32m1_sdk_riscv_installer.sh
./rv32m1_sdk_riscv_installer.sh
```

### Accept license ###

After the license was accepted, create a /vega folder and extract them there

```
mkdir vega && cd vega
tar xf ../rv32m1_sdk_riscv.tar.gz
```

### 4.3 Extract toolchain

Extract toolchain compressed file into the /vega folder

```
cd $HOME/vega
mkdir toolchain && cd toolchain
tar xf ../../Toolchain_Linux.tar.gz
tar xf riscv32-unknown-elf-gcc.tar.gz
rm riscv32-unknown-elf-gcc.tar.gz
tar xf openocd.tar.gz
rm openocd.tar.gz
```

### 4.4 Set environment variables

```
export RV32M1_SDK_DIR=$HOME/vega/rv32m1_sdk_riscv
export PATH=$PATH:$HOME/vega/toolchain
export RISCV32GCC_DIR=$HOME/vega/toolchain/riscv32-unknown-elf-gcc
export PATH=$PATH:$RISCV32GCC_DIR/bin
```

### 4.5 Download and extract GNU-MCU Eclipse IDE

```
cd $HOME
curl -L https://github.com/gnu-mcu-eclipse/org.eclipse.epp.packages/releases/download/v4.6.1-20190923-2019\
-09/20190923-1700-gnumcueclipse-4.6.1-2019-09-R-linux.gtk.x86_64.tar.gz > gnumcueclipse-4.6.1_x86_64.tar.gz
tar xf gnumcueclipse-4.6.1_x86_64.tar.gz
rm gnumcueclipse-4.6.1_x86_64.tar.gz
```

### 4.6 Configure toolchain and OpenOCD in Eclipse

```
mkdir -p $HOME/eclipse/configuration/.settings/

echo "eclipse.preferences.version=1" > \
$HOME/eclipse/configuration/.settings/ilg.gnumcueclipse.debug.gdbjtag.openocd.
prefs

echo "install.folder=$HOME/vega/toolchain" >> \
$HOME/eclipse/configuration/.settings/ilg.gnumcueclipse.debug.gdbjtag.openocd.
prefs

echo "eclipse.preferences.version=1" > \
$HOME/eclipse/configuration/.settings/ilg.gnumcueclipse.managedbuild.cross.ris
cv.prefs

echo "toolchain.path.512258282=$HOME/vega/toolchain/riscv32-unknown-elf-
gcc/bin" >> \
$HOME/eclipse/configuration/.settings/ilg.gnumcueclipse.managedbuild.cross.ris
cv.prefs
```

### 4.7 Download J-Link Software for Linux

Go to https://www.segger.com/downloads/jlink#J-LinkSoftwareAndDocumentationPack
Download the latest J-Link Software and Documentation pack for your system (DEB installer in this case)
Install the J-Link pack:

```
sudo dpkg -i $HOME/Downloads/JLink_Linux_Vxxx_x86_64.deb
```

## 5. Prepare Linux environment for Zephyr

The instructions from this guide will be executed on a Linux Terminal (Ubuntu) and it will use the cross-compiler from open-isa.org instead of the one from zephyr-sdk, hence previous chapter is required prior running these instructions.
See https://docs.zephyrproject.org/latest/getting_started/index.html for additional details.

### 5.1 Configure PATH environment variable

```
export PATH=$PATH:$HOME/.local/bin
```

### 5.2 Install required software packages

```
sudo apt install --no-install-recommends git ninja-build gperf \
  ccache dfu-util device-tree-compiler wget \
  python3-pip python3-setuptools python3-tk python3-wheel xz-utils file \
  make gcc gcc-multilib
```

### 5.3 Install CMAKE (CMake 3.13.1 or higher is required)

```
sudo snap install cmake --classic
```

Note: You may need to uninstall any previous cmake package and restart the terminal.
Verify your current cmake with: '**cmake --version**'

### 5.4 Update Device Tree Compiler (dtc_1.4.6 or higher is required)

```
curl -L http://mirrors.kernel.org/ubuntu/pool/main/d/device-tree-compiler/device-\
tree-compiler_1.4.7-1_amd64.deb > device-tree-compiler_1.4.7-1_amd64.deb
sudo dpkg -i device-tree-compiler_1.4.7-1_amd64.deb
rm device-tree-compiler_1.4.7-1_amd64.deb
```

### 5.5 Install west tool

```
pip3 install --user -U west
```

### 5.6 Initialize west and update it

```
cd $HOME
west init zephyrproject
cd zephyrproject
west update
```

### 5.7 Install additional applications from zephyr/scripts/requirements.txt

```
cd $HOME/zephyrproject/zephyr
pip3 install --user -r scripts/requirements.txt
```

### 5.8 Set environment variables for cross-compile toolchain from open-isa.org

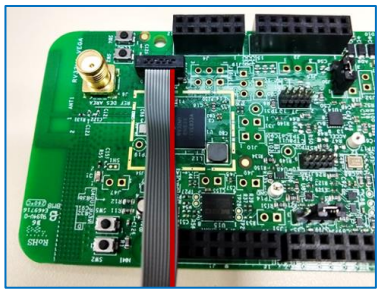
```
export ZEPHYR_TOOLCHAIN_VARIANT=cross-compile
export CROSS_COMPILE=$HOME/vega/toolchain/riscv32-unknown-elf-gcc\
/bin/riscv32-unknown-elf-
```

### 5.9 Run script from Zephyr to set some environment variables

```
source zephyr-env.sh
```

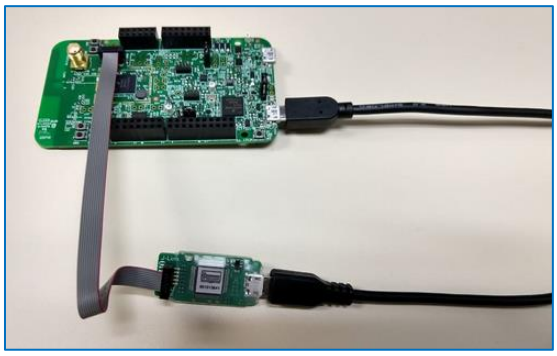
## 6. Connect your board and debugger to computer

Connect the J-Link debug probe to J55 header from VEGAboard as shown below.



Connect an USB cable to J12 port from VEGAboard and then to computer as shown below.

Connect the J-Link debug probe to the computer.
This is how the setup should look like, both USB cables connected to the PC.



## 7. Build & Flash Zephyr application using riscv32-unknown-elf-gcc

### 7.1 Build the blinky example project

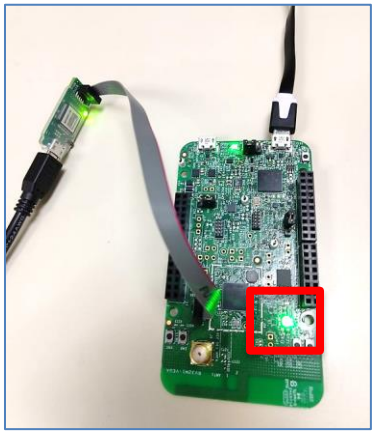
```
cd $ZEPHYR_BASE
cmake -B blinkyBuild -GNinja -DBOARD=rv32m1_vega_ri5cy -\
DCMAKE_REQUIRED_FLAGS=-Wl,-dT=/dev/null samples/basic/blinky
```

Note: If something fails, delete the $ZEPHYR_BASE/*blinkyBuild* folder, double check the environment variables, DTC version, CMAKE version and run command again.

### 7.2 Download the application to the board using west (make sure the J-link is connected)

```
cd blinkyBuild
west flash --openocd=$HOME/vega/toolchain/openocd
```

Disconnect and connect the board, you should see the **Green LED blinking**



# 8. MicroPython

## 8.1 Get Micropython

```
cd $HOME/zephyrproject
git clone https://github.com/micropython/micropython.git
cd $HOME/zephyrproject/micropython/ports/zephyr
```

## 8.2 Modify Makefile

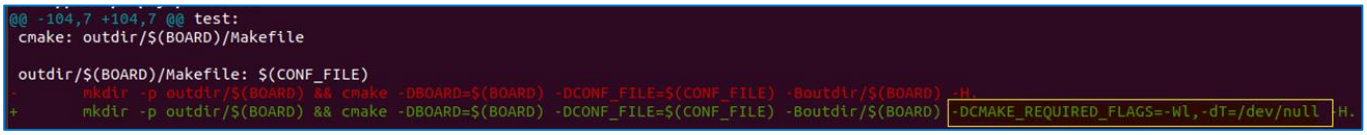
Open Makefile file

```
vi Makefile
```

Locate line 107 and insert the following:

**-DCMAKE_REQUIRED_FLAGS=-Wl,-dT=/dev/null** in mkdir command, after -Boutdir/$(BOARD), it should look like this:

mkdir -p outdir/$(BOARD) && cmake -DBOARD=$(BOARD) **-DCONF_FILE=$(CONF_FILE) -Boutdir/$(BOARD) -DCMAKE_REQUIRED_FLAGS=-Wl,-dT=/dev/null** -H.




## 8.3 Build the application

Build the application indicating the board to be used.

```
make BOARD=rv32m1_vega_ri5cy
```

After the build, the generated image (zephyr.elf) is stored in:

"…/outdir/rv32m1_vega_ri5cy/zephyr".

## 8.4 Program application into flash

Go to folder where the build application resides and flash it to your board.

```
cd $HOME/zephyrproject/micropython/ports/zephyr/outdir/rv32m1_vega_ri5cy/
west flash --openocd=$HOME/toolchain/openocd
```

Disconnect and connect the board; then press the reset button.

## 8.5 Verify serial output

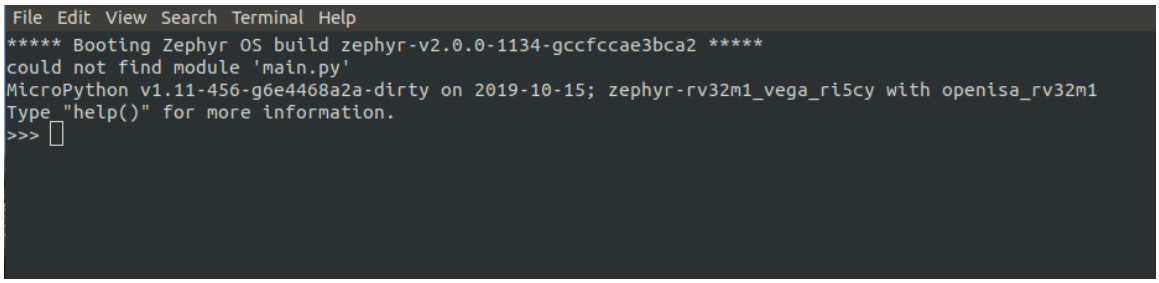
Open a Serial Terminal to verify output. This guide will use "Screen" application.
Terminal settings: Baud-rate: 115200, Data: 8bits, Parity: None, Flow Control: None.
Note: Locate your interface under: /dev/**tty*xxx***:

```
ls /dev/tty*
```

Make sure you select the port corresponding to your setup, in this case: /dev/**ttyACM0**

```
sudo apt install screen
sudo screen /dev/ttyACM0 115200
```

After the terminal is ready, **press the reset button** on the VEGAboard.



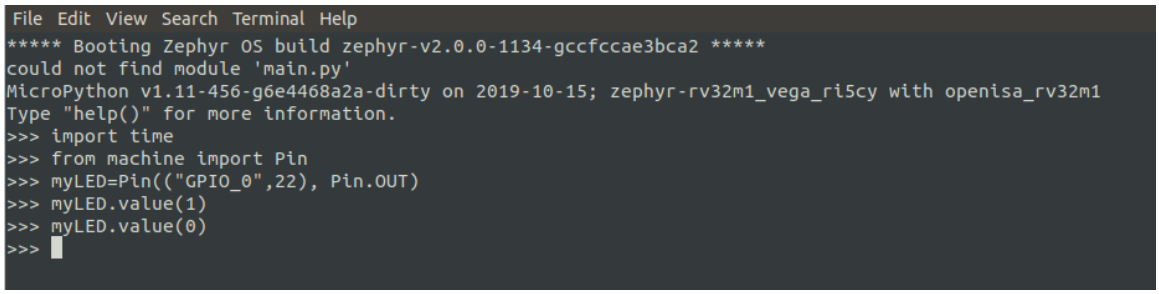

To end the session in screen, press '**Ctrl+a**', type '**:quit**' and press '**Enter**'.

### 8.6 Test MicroPython

Use the Serial Terminal to test MicroPython. Use the following python scripts for test.

**Turn on the blue LED (PTA22) test:**

```
import time
from machine import Pin
myLED=Pin(("GPIO_0",22), Pin.OUT)
myLED.value(1)
myLED.value(0)
```




**Flash the blue LED (PTA22) test:**

```
import time
from machine import Pin
LED=Pin(("GPIO_0",22), Pin.OUT)
while True:
  LED.value(1)
  time.sleep(0.5)
  LED.value(0)
  time.sleep(0.5)
```

## 9. Build & Flash Zephyr demo from Eclipse

### 9.1 Make sure GNU-MCU Eclipse IDE is installed and configured

Refer to sections 4.5 and 4.6.

## 9.2 Generate Eclipse project files for Hello_World example from Zephyr

The files will be created in Zephyr folder.

```
cd $ZEPHYR_BASE
```

Create a folder for project

```
mkdir myProject && cd myProject
```

Compile project and generate eclipse files

```
cmake -G"Eclipse CDT4 - Ninja" -DBOARD=rv32m1_vega_ri5cy -DCMAKE_REQUIRED_FLAGS=-Wl,-dT=/dev/null \
$ZEPHYR_BASE/samples/hello_world/
```

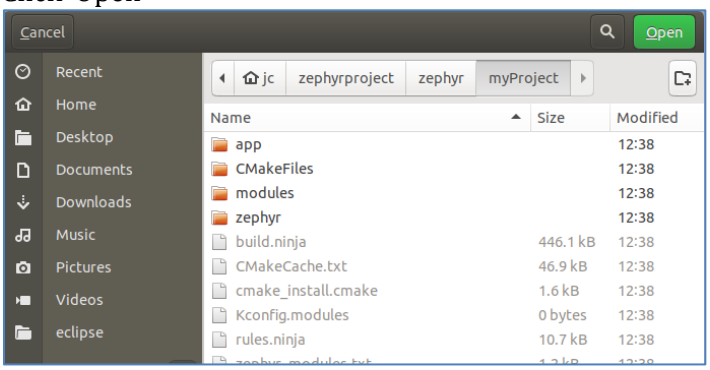
## 9.3 Open eclipse

```
cd $HOME/eclipse
./eclipse
```

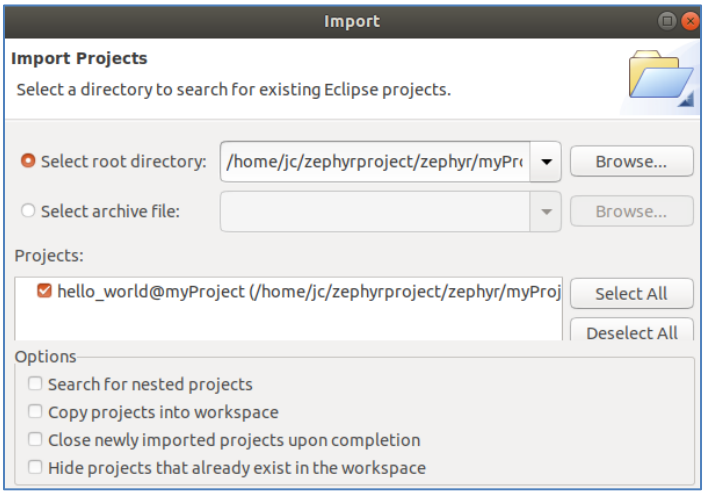
Select any workspace.

## 9.4 Import files from 'myProject' folder

1. Click on File -> Import 
2. Select General -> Existing projects into workspace
3. Click on Browse and navigate to the previously generated **myProject** folder
4. Click 'Open'




5. Make sure "hello_world@myProject" project is selected
6. Click 'OK' and 'Finish'

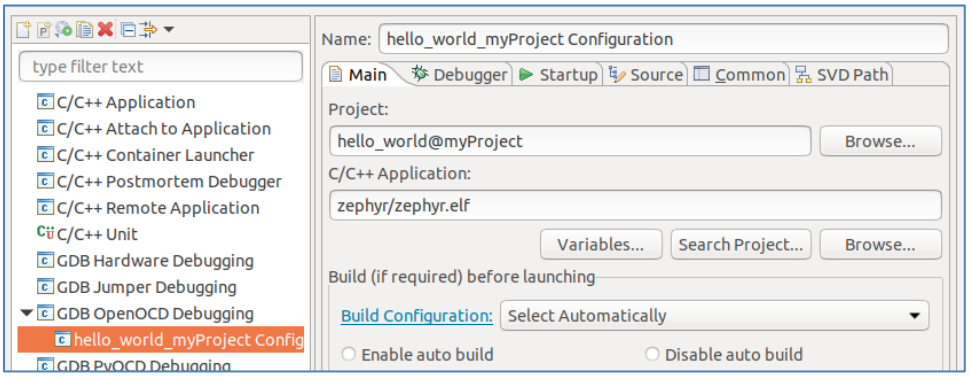


## 9.5 Build the project

Click the “Hammer” to build your application 

## 9.6 Create your Debug Configuration

1. Click on Run -> Debug Configurations
2. Double click on “GDB OpenOCD Debugging” to create a debug configuration
3. Main Tab
    - Project: hello_world@myProject
    - C/C++ Application: Click on “Search Project” and select zephyr.elf

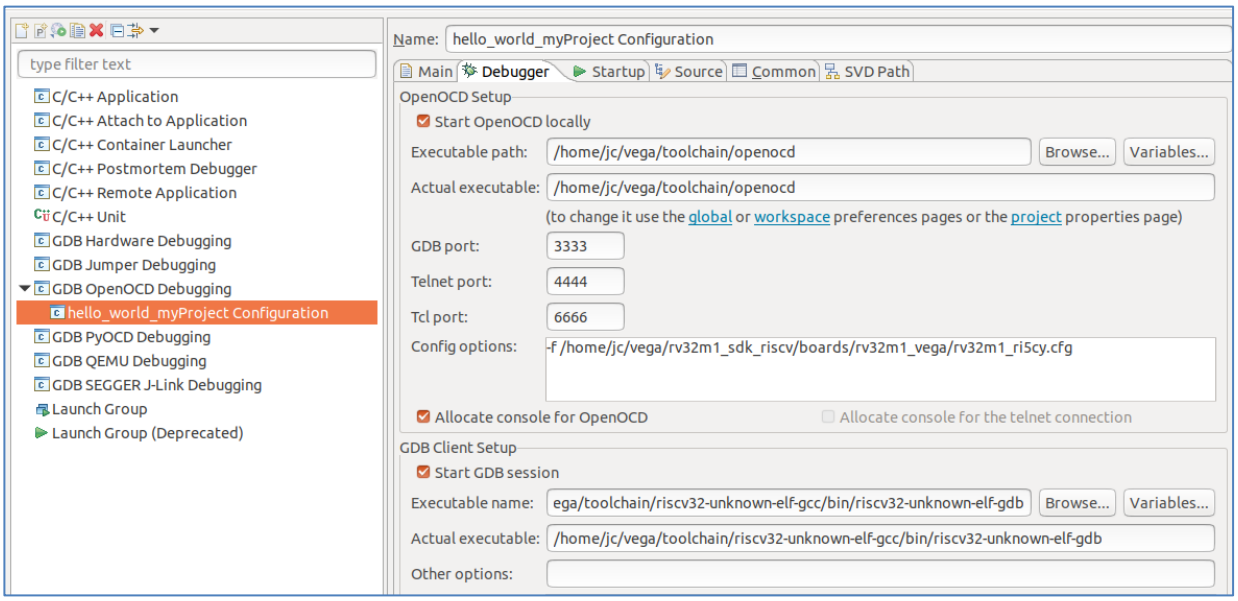



4. Debugger Tab

    OpenOCD Setup:
    - Executable path: Click on Browse and locate the OpenOCD executable under $HOME/vega/toolchain
    - Actual executable: Filled automatically based on previous parameter
    - GDB port: 3333
    - Telnet port: 4444
    - Tcl port: 6666
    - Config options: -f <$HOME>/vega/rv32m1_sdk_riscv/boards/rv32m1_vega/rv32m1_ri5cy.cfg

    GDB Client Setup:
    - Executable name: Click on Browse and locate the riscv32-unknown-elf-gdb executable under $HOME/vega/toolchain/riscv32-unknown-elf-gcc/bin
    - Actual executable: Filled automatically based on previous parameter

5. Click on ‘Apply’




## 9.7 Flash the application to your board using Eclipse

6. Make sure your board is connected and click on 'Debug' to program your board
7. Reset your board and verify the application was programmed
8. Open a Serial Terminal to verify output. See section 8.5 for instructions.