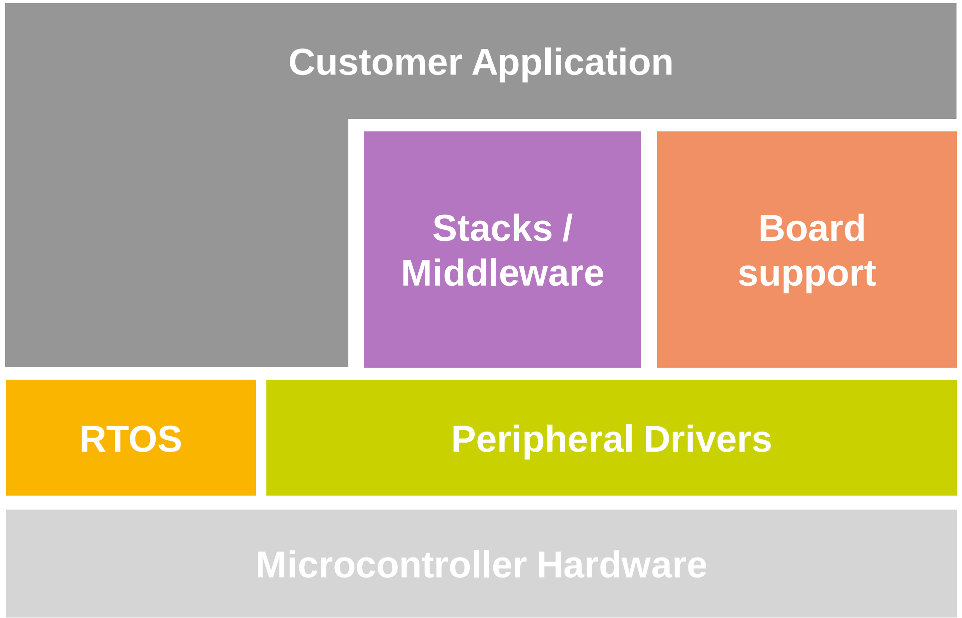
VEGABOARD SDK RISCV

QUICK GUIDE

PUBLIC

SECURE CONNECTIONS
FOR A SMARTER WORLD

VEGAboard SDK



Features

Architecture:

- Single driver for each peripheral
- Transactional APIs w/ optional DMA support for communication peripherals

Integrated RTOS:

- FreeRTOS v9
- RTOS-native driver wrappers

Integrated Stacks and Middleware

- USB Host, Device and OTG
- BLE stack
- Amazon Web Service IoT
- QCA WiFi Stacks
- lwIP, FatFS
- Crypto acceleration plus wolfSSL
- SD and eMMC card support

Reference Software:

- Peripheral driver usage examples
- Application demos
- FreeRTOS usage demos

License:

- BSD 3-clause for startup, drivers, USB stack

Toolchains:

- Eclipse IDE
- GCC w/ Cmake

Quality

- Production-grade software
- MISRA 2004 compliance
- Checked with Coverity® static analysis tools

Get Software & Tools

- https://open-isa.org/downloads/

- Linux/Mac SDK:
  - rv32m1_sdk_riscv_installer.sh

- Toolchain (Prebuilt GCC and OpenOCD for Linux)
  - Toolchain_Linux.tar.gz

rv32m1_SDK folder

```
<sdk root directory>
├ devices    SOC peripheral driver source code and toolchain support code
├ boards    Demo source code and project files
├ rtos    FreeRTOS support package
├ middleware    Third Party middleware source code
├ tools    CMake supporting files
├ RISCV    RISCV supporting files
├ Getting Started with RV32M1 SDK RISCV.pdf    Getting Started Tutorial
├ SW-Content-Register.txt    Software Content Register File
└ LA_OPT_NXP_Software_License.htm
  LA_OPT_WOLFSSL_EVAL.htm    License Files
```

rv32m1_SDK demo applications

```
boards
└ rv32m1_vega
  ├ demo_apps            out of box demos
  ├ driver_examples      baremater examples for demo driver of various on-chip/on-board peripherals
  ├ rtos_examples        demos in freertos context
  ├ multicore_examples   rpmsg-lite based multicore examples
  ├ usb_examples         various usb examples
  ├ wireless_examples
  │ └ bluetooth          bluetooth examples
  ├ rv32m1_ri5cy.cfg     ri5cy OpenOCD config file
  └ rv32m1_zero_riscy.cfg   zero_riscy openOCD config file
```

STEP BY STEP USING TERMINAL

Get software and tools (already in VM image)

 # 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

 # Extract SDK

- cd $HOME
- chmod +x rv32m1_sdk_riscv_installer.sh
- ./rv32m1_sdk_riscv_installer.sh
- # Accept license
- mkdir vega && cd vega
- tar xf ../rv32m1_sdk_riscv.tar.gz

 # Extract toolchain

- cd $HOME
- 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

Set environment variables

 # Set environment variables

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

Build & Run: From Terminal

 # Go to the demo application folder. Ie. hello_world:

- cd $RV32M1_SDK_DIR/boards/rv32m1_vega/demo_apps/hello_world/ri5cy/riscvgcc

 # Execute the script to build the application

- ./build_debug.sh

 # Flash the application using OpenOCD + GDB (Make sure the board is connected to PC and J-Link)

- openocd -f $HOME/vega/rv32m1_sdk_riscv/boards/rv32m1_vega/rv32m1_ri5cy.cfg

  # Open another terminal session (don’t forget to configure the env variables) or Press Ctrl+z and ‘bg’

- cd $RV32M1_SDK_DIR/boards/rv32m1_vega/demo_apps/hello_world/ri5cy/riscvgcc/debug
- riscv32-unknown-elf-gdb hello_world.elf

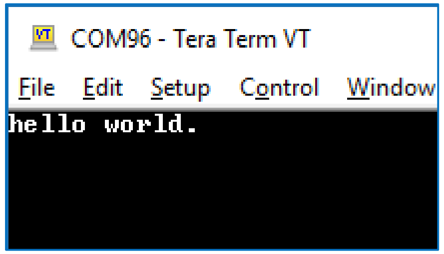
  (gdb) target remote localhost:3333

  (gdb) load

  (gdb) monitor reset

  (gdb) quit

 # Open a Serial Terminal to verify output.
Settings: Baud-rate: 115200, Data: 8bits, Parity: None, Flow Control: None.



Build & Run: From Eclipse

- Your VM image should already have Eclipse installed and configured, if you don't have it, please refer to "Getting Started with RV32M1 SDK", Chapter 4.
  - Summary:
    - Make sure GNU MCU Eclipse plug-in is installed with RISCV C/C++ Cross Tools selected
    - Configure Global OpenOCD Path - /home/user/toolchain
    - Configure Global RISC-V Toolchains Paths - /home/user/toolchain/riscv32-unknown-elf-gcc/bin

 # Open eclipse

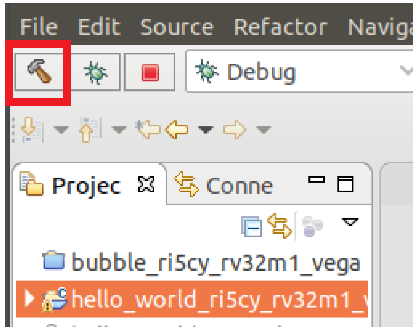
- cd $HOME/eclipse
- ./eclipse

 # Import an existing project. Ie, the hello_world path:

$HOME/vega/rv32m1_sdk_riscv/boards/rv32m1_vega/demo_apps/hello_world/ri5cy/riscveclipse

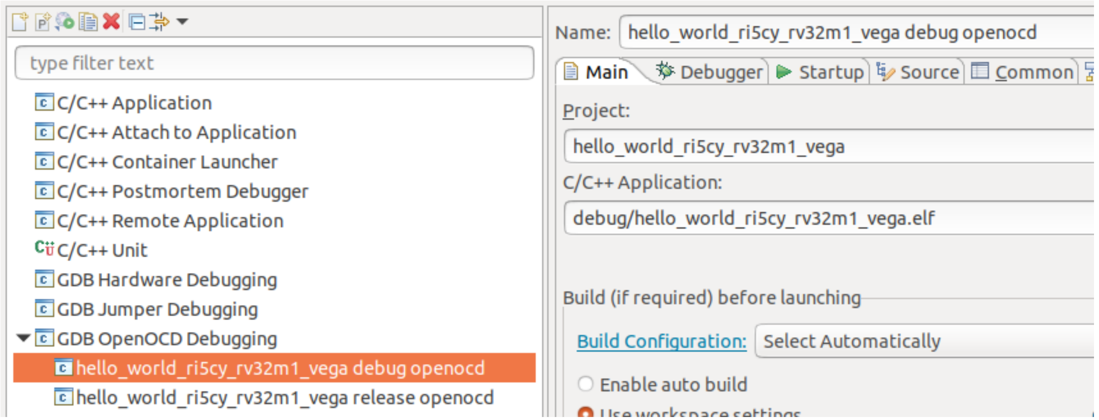
 # Click 'OK' and 'Finish'

 # Click the "Hammer" to build your application



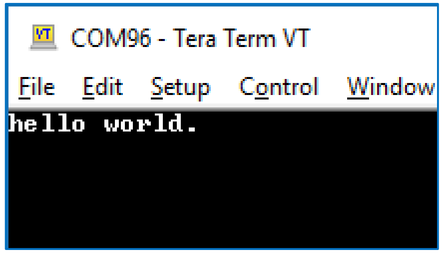
Build & Run: From Eclipse (2)

- Go to Run -> Debug Configurations
- Select a debug configuration from ‘GDB OpenOCD Debugging’



- Click on ‘Debug’
- Click on ‘Resume’ or stop the debugger

- Open a serial terminal and verify the output



Now it’s your turn!

- Download NXP’s IoT Toolbox application to your smartphone
  The application is Available for Android and iOS.

- Load the Bluetooth Low Energy **Heart Rate Sensor** application located at:

…/rv32m1_sdk_riscv/boards/rv32m1_vega/wireless_examples/bluetooth/heart_rate_sensor/freertos/ri5cy/

- It’s recommended to use the Eclipse-based scenario but feel free to try any setup.
  - Start the Heart Rate Sensor application, you should see the red LED blinking.
  - Open the IoT Toolbox and select the Heart Rate app
  - You should see your device being advertised
  - Select your device to start a connection

Advertising

Connected

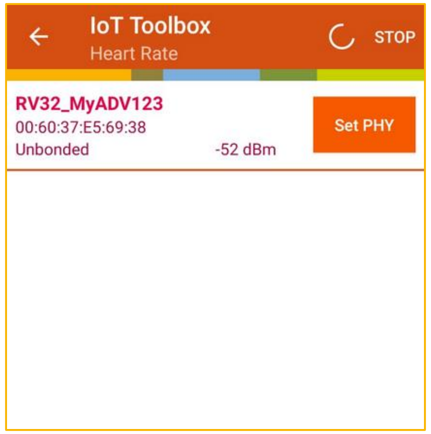
Which one is my board?

- You may see many Advertisements from different boards around you, let’s change the ADV NAME of your device to make sure you are connecting to it.
- Open the file “wireless_examples\bluetooth\heart_rate_sensor\freertos\app_config.c” in the heart_rate_sensor demo.
- In line 75, you will find the Advertising name **.aData**, modify this to identify your board.

  Note: the length cannot be larger than 14 including the ending character (\0).
- Don’t forget to adapt the **.length** variable.

app_config.c

```
70        .aData = (uint8_t *)adData1
71     },
72     {
73        .adType = gAdShortenedLocalName_c,
74  //       .length = 9,
75  //       .aData = (uint8_t*)"RV32_HRS"
76        .length = 14,
77        .aData = (uint8_t*)"RV32_MyADV123" //Max length: 14 characters including end character '\0'
78     }
79  };
80
```



Reference

- Open-ISA.org
  - [Getting Started with RV32M1 SDK (RISCV)]

SECURE CONNECTIONS
FOR A SMARTER WORLD