

织女星开发板快速入门指南

1. 基本介绍

本指南描述了在 Windows 环境下编译和运行织女星开发板 SDK 示例的详细步骤。织女星开发板（RV32-VEGA-Lite）是一款体积小，功耗低，高性价比的 RISC-V 开发板。在此开发板上，用户可以快速建立一个使用 RV32M1 芯片的 RISC-V 应用和演示系统。

RV32M1 是一颗真正的 RISC-V 芯片，其目的是为了扩大和推动 RISC-V 生态系统的发展，让广大的 MCU 嵌入式开发工程师有真正的 RISC-V 芯片可用。目前该芯片只供应给 OPEN-ISA 社区生产评估开发板使用。

RV32M1 芯片集成了四个核，其中有两个 RISC-V 核：一个 RI5CY 核，一个 ZERO_RISCY 核。另外两个是 ARM Cortex-M4 核和 ARM Cortex-M0+核。有关 RV32M1 设备的详细信息，请参阅 RV32M1 参考手册文档。从 www.open-isa.cn 下载支持所有核的 SDK 软件开发包。

在织女星开发板上编译和运行一个示例需要准备以下四个硬件工具：

- 个人电脑
- J-Link 仿真器
- Micro-USB 数据线
- 织女星开发板

目录

1. 基本介绍	1
2. 准备工作	2
2.1 建立 RISC-V 开发环境.....	2
2.2 下载织女星开发板的 SDK.....	2
2.3 开发板简介	2
3. 启动配置	3
4. 通过 Eclipse 运行示例	5
4.1 编译示例应用程序.....	5
4.2 运行示例应用程序.....	7
5. 通过命令行运行示例.....	10
5.1 编译示例应用程序.....	10
5.2 通过 OpenOCD + telnet 运行示例应用程序	11
5.3 通过 OpenOCD + GDB 运行示例应用程序	12
6. 参考文献	15
7. 修订版本	15



2. 准备工作

本节详细介绍了在织女星开发板上运行 SDK 自带的例子程序所需的准备工作及其操作步骤。

2.1 建立RISC-V开发环境

搭建 RISC-V 开发环境的详细过程请参考《搭建织女星开发板的 RISC-V 开发环境》文档，该文档可以从 www.open-isa.cn 中下载。在 Windows 下建立 RISC-V 开发环境需要安装如下软件和工具：

- Telnet 远程登陆客户端应用（Windows Telnet Client）
- GNU Eclipse IDE for C/C++ Developers
- GNU MCU Eclipse Windows Build Tools
- RV32M1 GNU GCC 工具链
- OpenOCD调试软件和J-Link驱动

如果习惯使用命令行来进行开发，则需要额外安装以下软件工具：

- CMake
- MinGW

2.2 下载织女星开发板的SDK

从 www.open-isa.cn 网站下载最新的支持 RISC-V 核的 SDK 压缩包，并且在电脑上适合的位置进行解压缩。

2.3 开发板简介

织女星开发板可以支持丰富多样的参考设计，提供尽可能多的 I/O 口连接到 RV32M1 主芯片。织女星开发板可以作为独立的板子应用，也可以连接其它有 Freedom 接口的板子来搭建更多功能的应用。此开发板的硬件用户指南文档可以从 www.open-isa.cn 网站下载。

图 1 展示了织女星开发板的主要部件，包括 RISC-V 的 JTAG 接口(J17), 复位键（SW1）和 USB 接口（J11）。

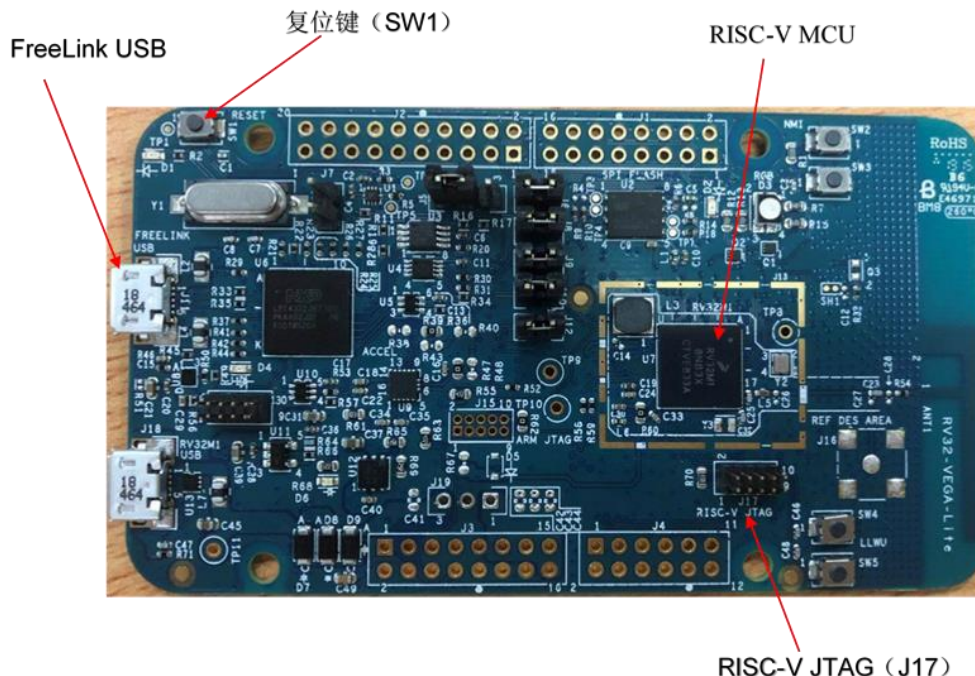


图 1 织女星开发板组件配置

3. 启动配置

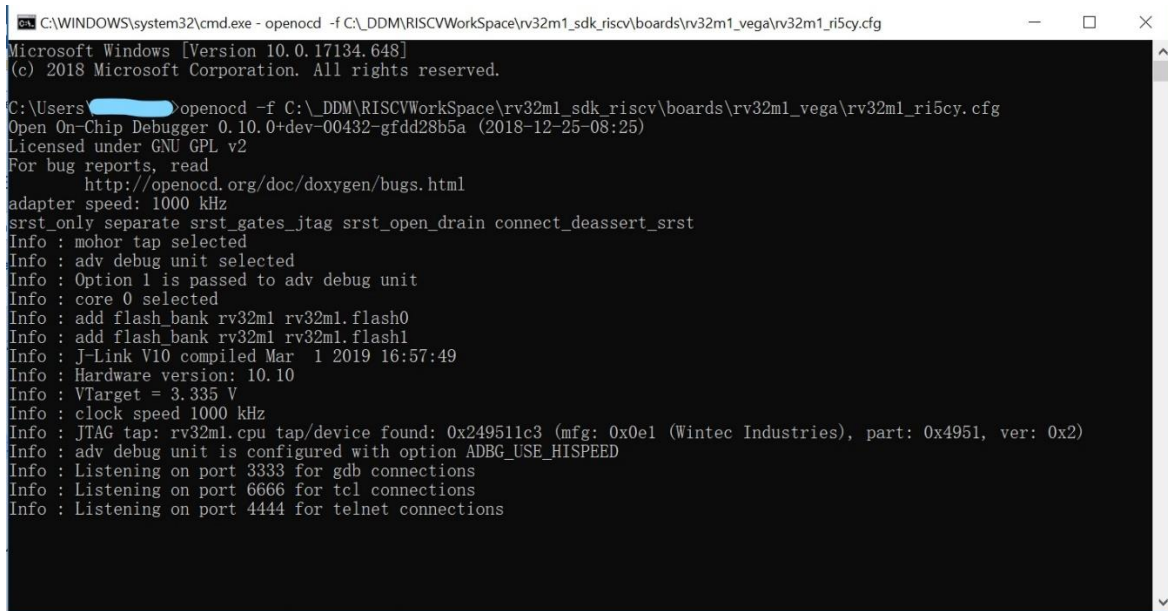
RV32M1 是一个异构四核 MCU，可以配置为上电后从某个核先启动，默认情况下是从 RISC-V RI5CY 核启动。更改启动模式可以通过配置 FOPT 设置来实现，有关 FOPT 设置的详细信息，请参阅 RV32M1 参考手册。如果开发板的启动模式配置被改变了，若还想恢复从 RI5CY 核启动，可以按照以下步骤来更改启动模式的配置。

注意：更改启动配置的操作，将擦写整个 Flash。

1. 通过 RISC-V JTAG 接口 (J17) 将 J-Link 调试器连接到目标板。
2. 将 Micro-USB 数据线插入 J11。
3. 打开 `cmd.exe`，输入如下的命令，通过调试软件 OpenOCD 连接目标开发板。

```
Openocd -f <install_dir>\boards\rv32m1_vega\rv32m1_ri5cy.cfg
```

其中，`<install_dir>` 是 RV32M1 SDK 的安装路径，图 2 显示了通过 OpenOCD 连接目标开发板。



```
C:\WINDOWS\system32\cmd.exe - openocd -f C:\_DDM\RISCVWorkspace\rv32m1_sdk_riscv\boards\rv32m1_vega\rv32m1_ri5cy.cfg
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\>openocd -f C:\_DDM\RISCVWorkspace\rv32m1_sdk_riscv\boards\rv32m1_vega\rv32m1_ri5cy.cfg
Open On-Chip Debugger 0.10.0+dev-00432-gfdd28b5a (2018-12-25-08:25)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
adapter speed: 1000 kHz
srst_only separate srst_gates_jtag srst_open_drain connect_deassert_srst
Info : mohor tap selected
Info : adv debug unit selected
Info : Option 1 is passed to adv debug unit
Info : core 0 selected
Info : add flash_bank rv32m1 rv32m1.flash0
Info : add flash_bank rv32m1 rv32m1.flash1
Info : J-Link V10 compiled Mar 1 2019 16:57:49
Info : Hardware version: 10.10
Info : VTarget = 3.335 V
Info : clock speed 1000 kHz
Info : JTAG tap: rv32m1.cpu tap/device found: 0x249511c3 (mfg: 0x0e1 (Wintec Industries), part: 0x4951, ver: 0x2)
Info : adv debug unit is configured with option ADBG_USE_HISPEED
Info : Listening on port 3333 for gdb connections
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
```

图 2 通过 OpenOCD 连接目标开发板

- 再打开一个新的 *cmd.exe* 界面，输入以下命令运行 **telnet** 程序。

```
telnet localhost 4444
```

- 持续按住复位键（SW1）的同时，在 **telnet** 窗口中输入以下命令并回车。

```
ri5cy_boot
```

- 完成后，释放复位按钮。
- 在步骤 6 中，也可以使用其他命令，如 *zero_boot*、*cm4_boot* 和 *cm0_boot* 来配置不同的启动模式。

4. 通过 Eclipse 运行示例

本节描述了通过 Eclipse 集成开发环境编译和运行一个 RV32M1 SDK 中提供的示例所需要的步骤。以 hello_word 为例，这些步骤也可以应用到其他的示例中去。同样的，所有的这些步骤也可以用来运行 RISC-V ZERO-RISCV 核中的例子。

4.1 编译示例应用程序

本章节中都是以 hello_world 作为例子来说明如何编译和运行，从下面路径的文件夹中可以找到 RISCV 核的 hello_world 例子的 Eclipse 项目：

```
<install_dir> \boards\rv32m1_vega\demo_apps\hello_world\ri5cy\riscveclipse
```

1. 将项目导入到Eclipse工作区。点击Eclipse菜单 “**File -> Import**”，在导入窗口中，选择 “**General ->Existing Project into Workspace**”，然后单击 “**Next**”，如图 3 所示。

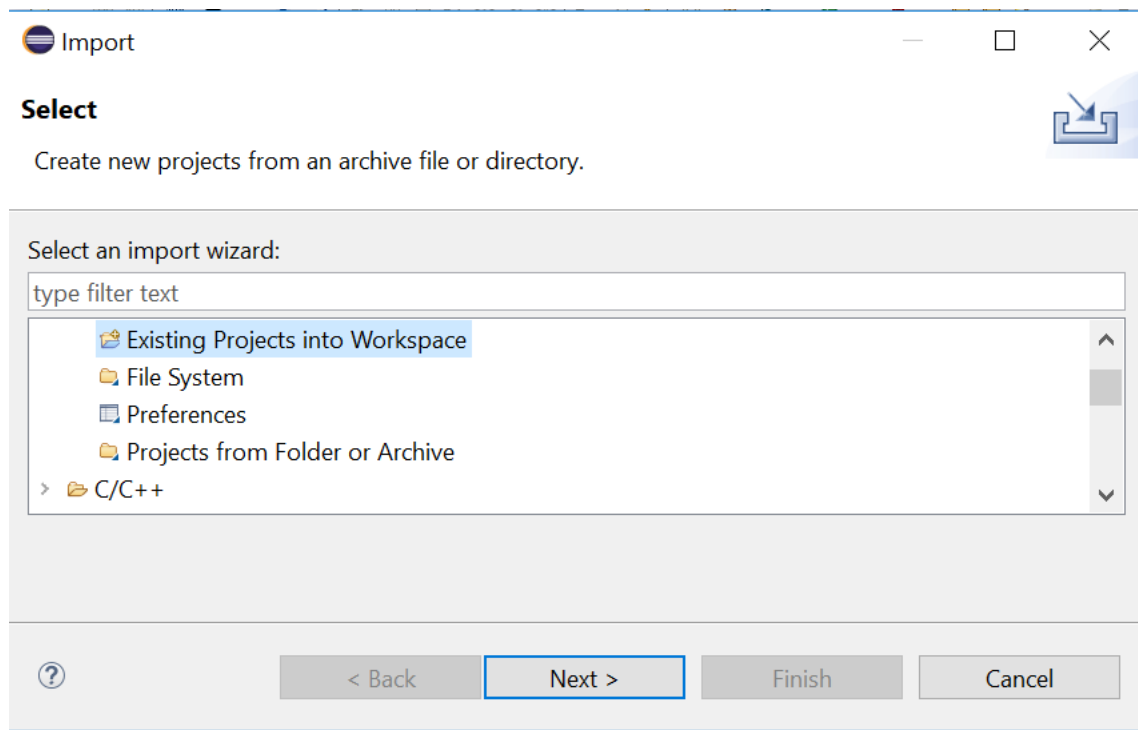


图 3 导入时的选择导入类型

2. 定位到 **hello_world** 的文件夹，选择要导入的项目，单击 “**Finish**”，如图 4 所示。

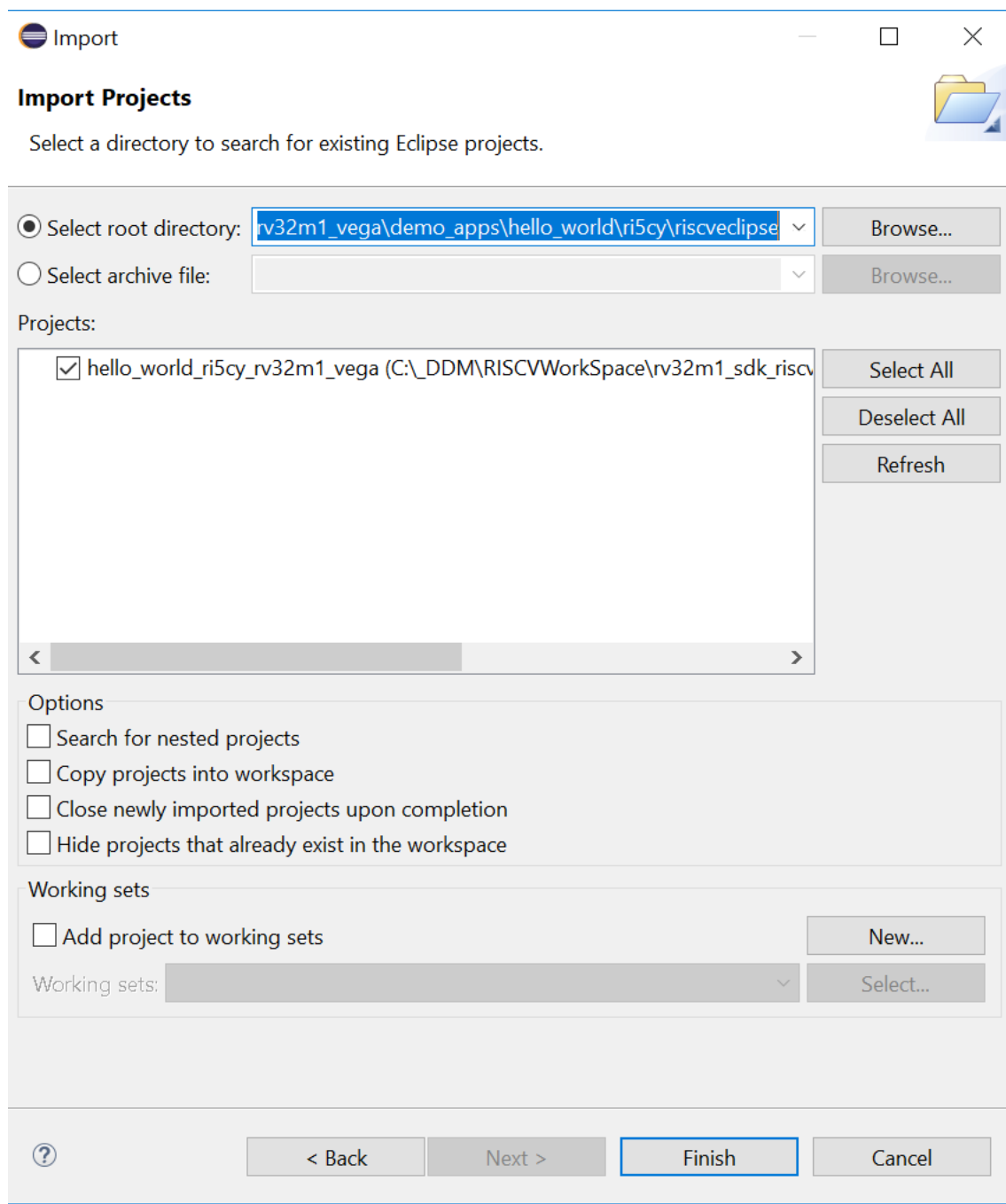


图 4 导入项目选择 hello_world 项目

3. 点击 “**Project -> Build project**” 来编译hello-world例子项目。

4.2 运行示例应用程序

1. 示例应用程序编译完成后，通过 RISC-V 的 JTAG（J17）接口，将 J-Link 调试器连接到目标开发板。
2. 将 Micro-USB 数据线插入 J11，另一端连接到电脑上。
3. 点击菜单 “Run -> Debug Configurations”，如图 5 所示，在左侧窗口选择要调试的目标对象，然后点击 “Run”。

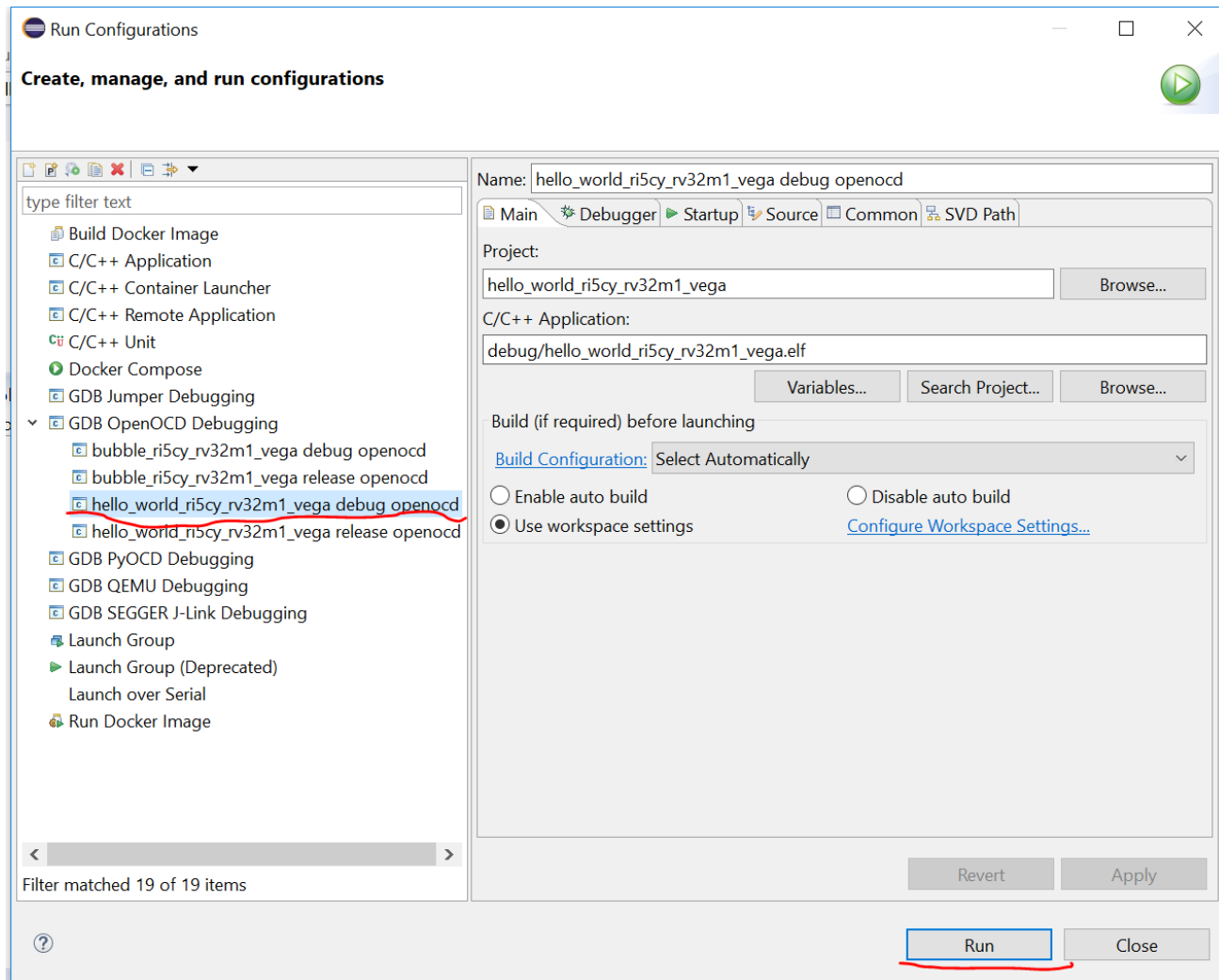


图 5 选择要调试的目标对象

4. 从 windows 设备管理器找到由 FreeLink 提供的虚拟串口号，运行串口调试工具程序，将其波特率设置为 115200，并打开对应串口设备。参考界面如图 6 和图 7 所示。

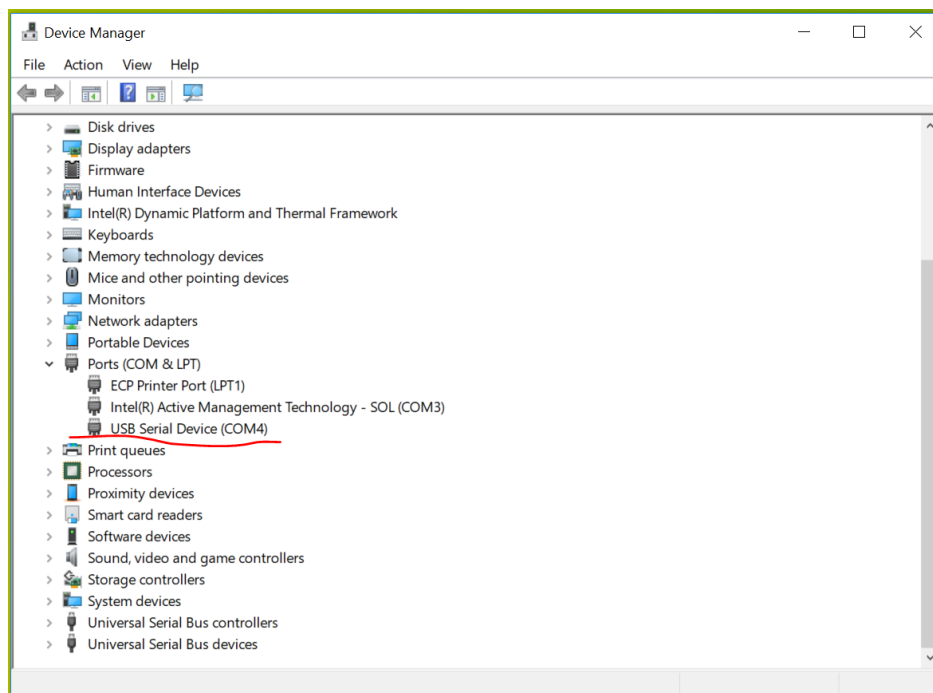


图 6 Windows 设备管理器中查看虚拟串口编号

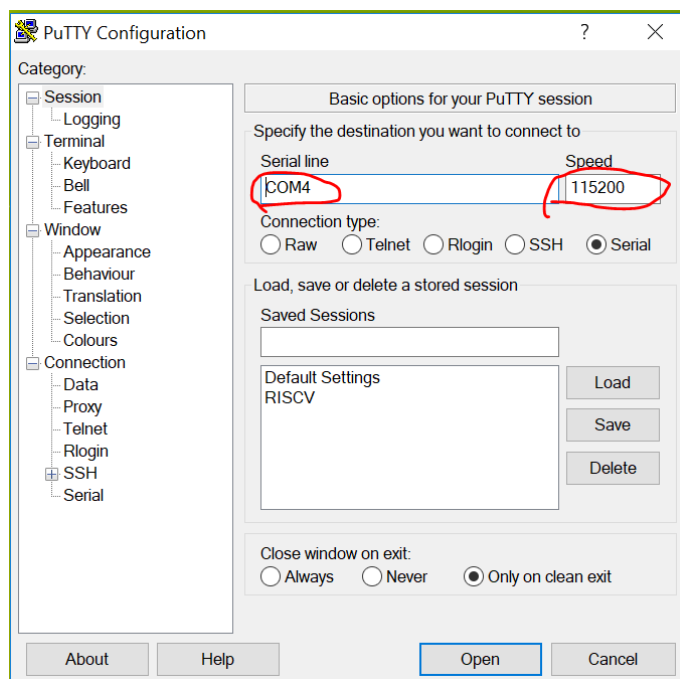


图 7 设置串口工具的串口号和波特率

5. 点击菜单 “**Run->Resume**” 来运行程序，如图 8 所示。也可以使用其它的命令来调试和控制应用程序。如果操作正确，这时候可以在串口工具上看到输出 **hello world**，如图 9 所示。

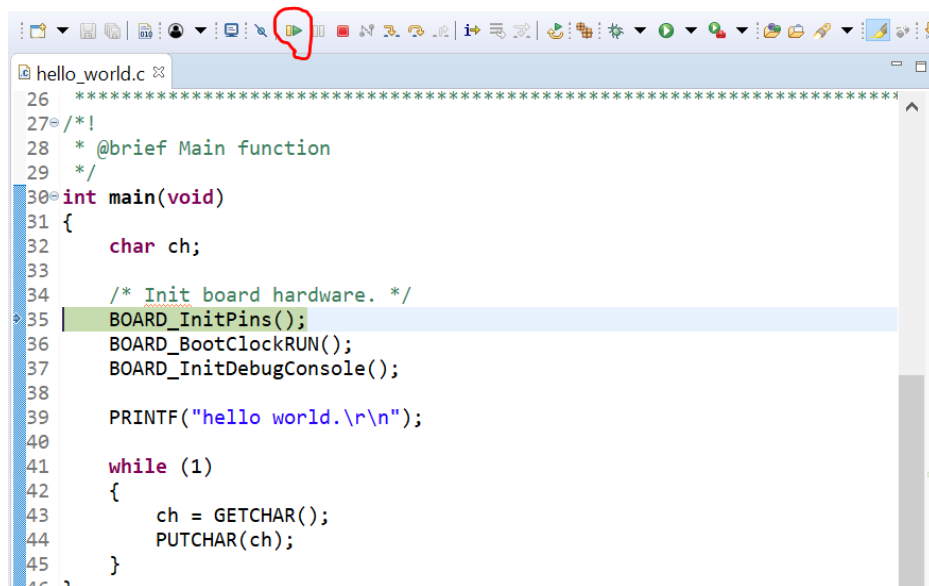


图 8 点击 resume 按钮运行

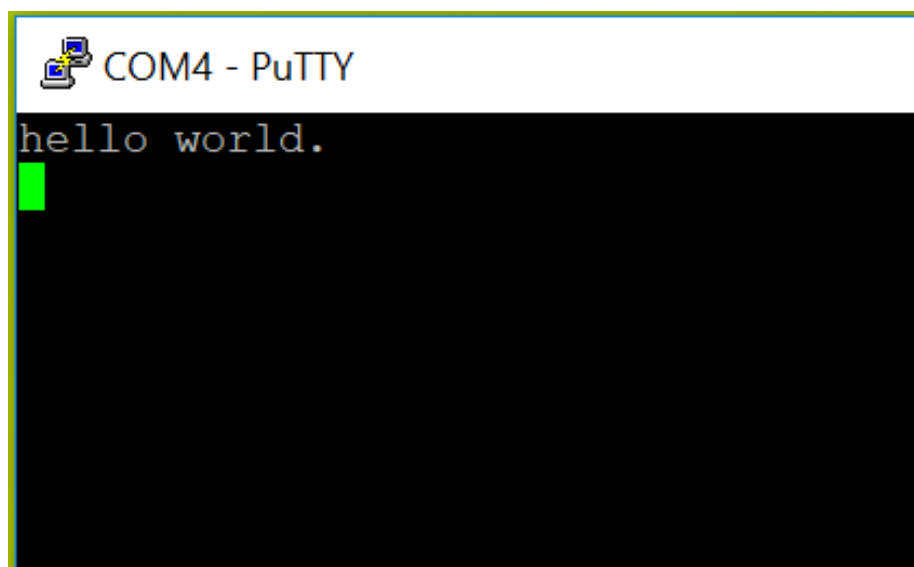


图 9 串口显示 hello world

5. 通过命令行运行示例

本节描述了通过命令行编译和运行一个 RV32M1 SDK 中提供的示例所需要的步骤。以 hello_word 为例，这些步骤也可以应用到其他的示例中去。同样的，所有的这些步骤也可以用来运行 RISC-V ZERO-RISCVY 核中的例子。

5.1 编译示例应用程序

- 1. 运行 *cmd.exe* 打开 Windows 命令提示符，改变当前目录为应用程序项目所在的目录。

```
cd <install_dir>\boards\rv32m1_vega\demo_apps\hello_world\ri5cy\riscvgcc
```

- 2. 输入 *build_debug.bat* 命令编译 debug 版本的例子。如图 10 显示已经编译成功了。也可以在 Windows 资源管理器中双击 build_debug.bat 文件来编译，如图 11 所示。结果在 debug 文件夹中将编译出 hello_world.elf 的可执行文件，如图 12 所示。

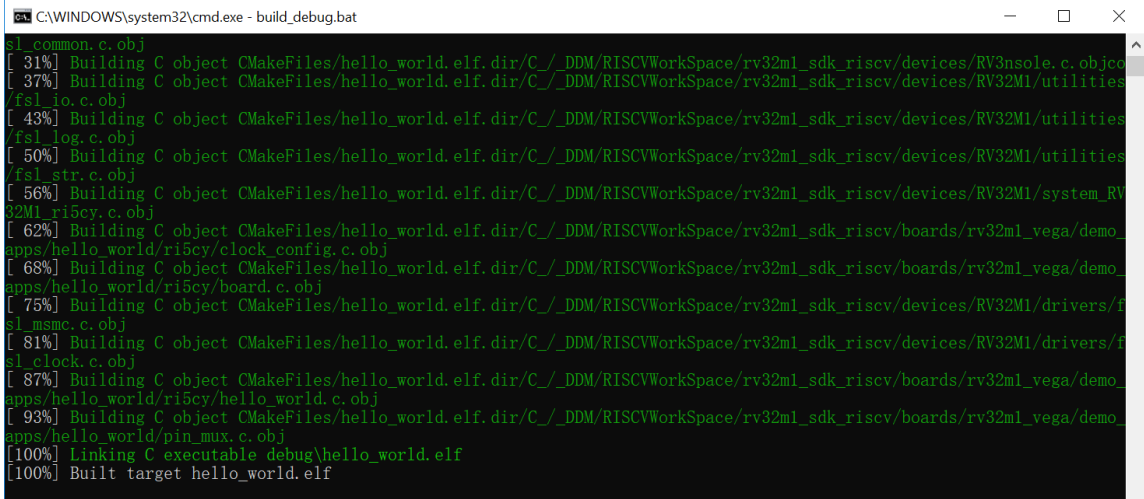


图 10 成功编译 hello_world.elf

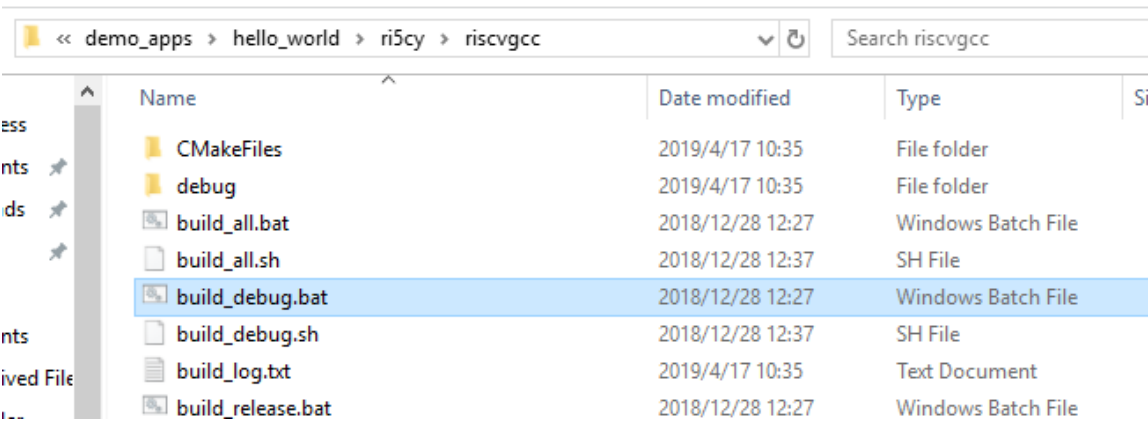


图 11 双击文件夹中的 build_debug.bat 文件

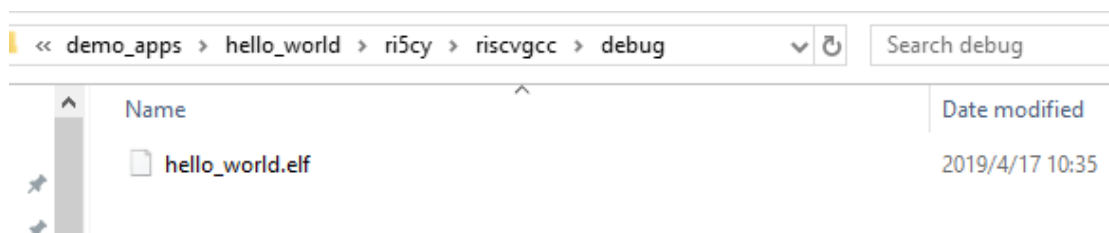


图 12 可执行文件编译成功

5.2 通过OpenOCD + telnet运行示例应用程序

1. 示例应用程序编译完成后，通过 RISC-V 的 JTAG 调试接口（J17）将 J-Link 调试器连接到目标板。
2. 将 Micro-USB 数据线插入 J11。
3. 打开 `cmd.exe`，输入如下命令将当前目录更改为前面编译生成的可执行文件所在的目录。

```
cd <install_dir>\boards\rv32m1_vega\demo_apps\hello_world\ri5cy\riscvgcc\debug
```

4. 再输入如下命令，通过 OpenOCD 连接到目标开发板，如图 13 所示。

注意：如果当前编译运行的是 ZERO-RISCV 核的应用，则需要使用 `rv32m1_zero_riscy.cfg` 文件。

```
Openocd -f <install_dir>\boards\rv32m1_vega \rv32m1_ri5cy.cfg
```

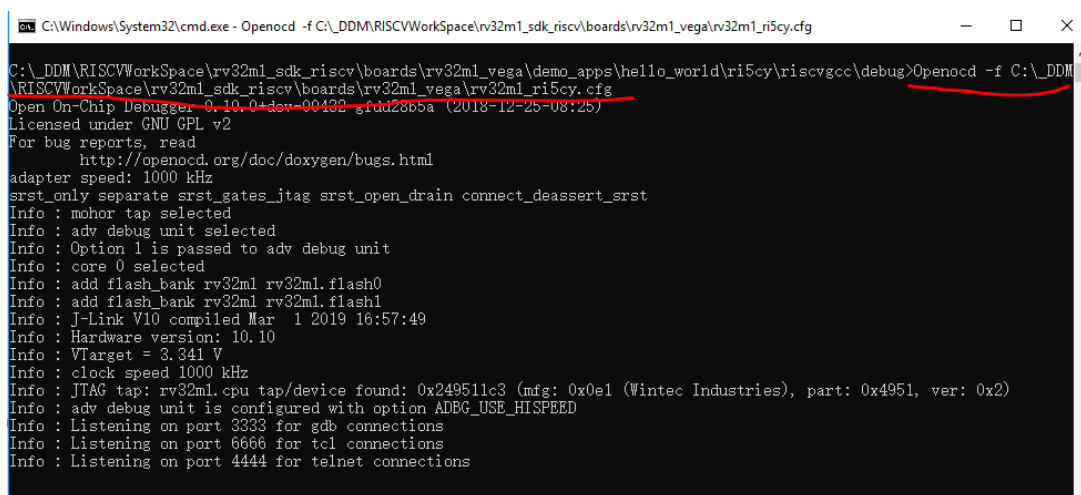


图 13 通过 OpenOCD 连接到目标板

5. 重新打开一个新的 `cmd.exe`，输入如下指令来打开 telnet 窗口。

```
telnet localhost 4444
```

- 在 **telnet** 窗口输入如下命令将可执行文件下载到 RV32M1 的片内 flash。如图 14 所示，下载完成后，终端上显示 **programming Finished**。

```
program hello_world.elf
```

```
Telnet localhost
Open On-Chip Debugger
> program hello_world.elf
JTAG tap: rv32m1.cpu tap/device found: 0x249511c3 (mfg: 0x0e1 (Wintec Industries), part: 0x4951, ver: 0x2)
** Programming Started **
auto erase enabled
Flash write discontinued at 0x0000310a, next section at 0x000fff00
Adding extra erase range, 0x000fff00 to 0x000ffeff
wrote 16640 bytes from file hello_world.elf in 0.468732s (34.668 KiB/s)
** Programming Finished **
>
```

图 14 编程完成

- 从 windows 设备管理器找到由 FreeLink 提供的虚拟串口号，运行串口调试工具程序，将其波特率设置为 115200，并打开对应串口设备。参考界面如图 6 和图 7 所示。
- 在 telnet 窗口中输入 **reset** 命令，MCU 将重启并且运行，串口工具中将会打印出 **hello world**。

除此之外，还可以通过下面这些命令来调试程序：

- reset halt** : 复位 MCU，并且 PC 停在程序的第一条指令处。
- halt** : 暂停 CPU 核的运行。
- resume** : 重新开始运行。
- step** : 单步运行。
- reg [reg_name]** : 读取寄存器的值。**reg_name** 是有效的寄存器名，包括: **ra**、**sp**、**gp**、**tp**、**t0-t6**、**s0-s11**、**s0-s7**、**npc**（下一个 PC 值）、**ppc**（前一个 PC 值）、**ustatus**、**utvec**、**uhartid**、**uepc**、**ucause**、**mstatus**、**mtvec**、**mepc**、**mcause**、**mhartid** 和 **privlv**。只能在 CPU 核停止运行的状态下使用这个指令来读取寄存器的值。
- reg reg_name value** : 设置寄存器值。通常只能在 CPU 核停止运行时使用。
- mw<w/h/b> addr value** : 将字/半字/字节的值写入给定的地址。
- md<w/h/b> addr [count]** : 从给定的地址开始读取一个（或者 **count** 个）字/半字/字节的值。

5.3 通过OpenOCD + GDB运行示例应用程序

- 示例应用程序编译完成后，通过 RISC-V 的 JTAG 调试接口（**J17**）将 J-Link 调试器连接到目标板。
- 将 Micro-USB 数据线插入 **J11**。
- 打开 **cmd.exe**，输入如下指令通过 OpenOCD 连接到目标板。打开一个新的 **cmd.exe** 输入如下指令将目录更改为之前编译生成的可执行文件的路径。

```
Openocd -f <install_dir>\boards\rv32m1_vega\rv32m1_ri5cy.cfg
```

4. 打开一个新的 `cmd.exe` 输入如下指令将目录更改为之前编译生成的可执行文件的路径。

```
cd <install_dir>\boards\rv32m1_vega\demo_apps\hello_world\ri5cy\riscvgcc\debug
```

5. 输入如下指令来运行 GDB 调试 `hello_world` 程序。如图 15 所示。

```
riscv32-unknown-elf-gdb hello_world.elf
```

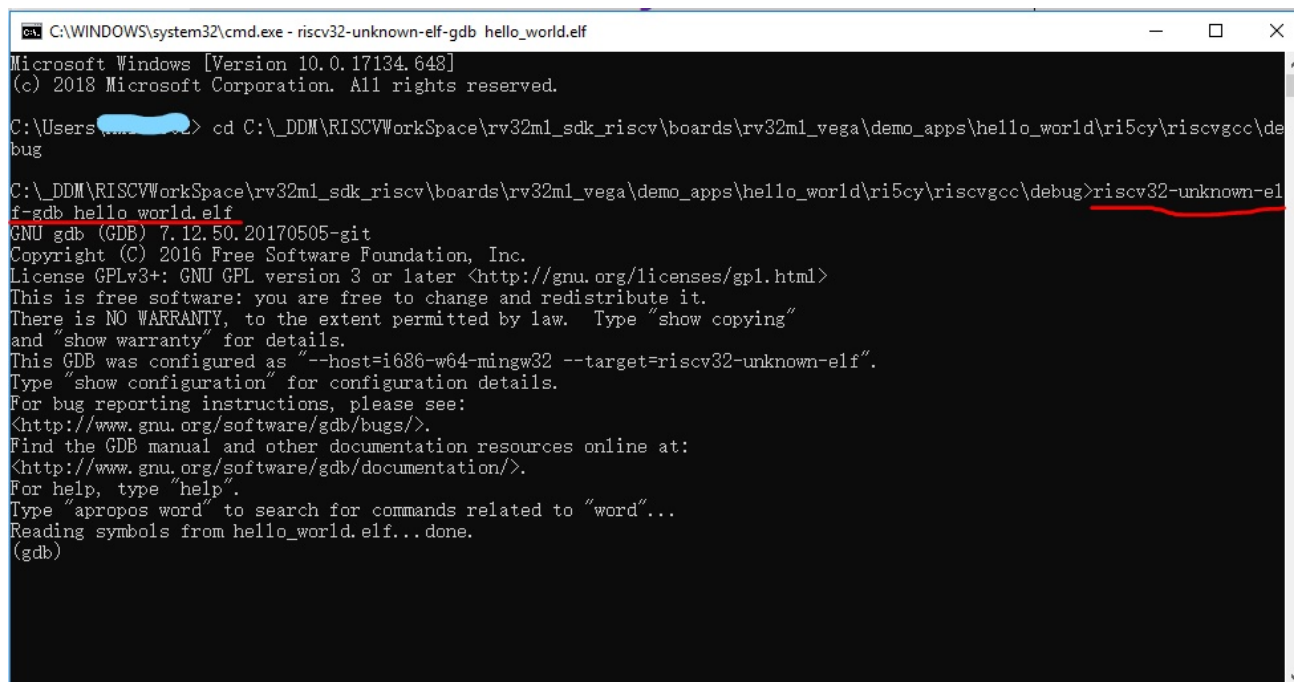


图 15 开启 GDB

6. 输入如下指令来连接 GDB 到 OpenOCD。

```
target remote localhost:3333
```

7. 输入 `load` 命令将二进制文件加载到 flash 当中去。
8. 从 windows 设备管理器找到由 FreeLink 提供的虚拟串口号，运行串口调试工具程序将其波特率设置为 115200，并打开对应串口设备。参考界面如图 6 和图 7 所示。
9. 输入 `monitor reset` 重启一下，如图 16 所示，然后就能在串口调试工具中看到打印出的 `hello world` 了。

```

C:\WINDOWS\system32\cmd.exe - riscv32-unknown-elf-gdb hello_world.elf
C:\_DDM\RISCVWorkspace\rv32m1_sdk_riscv\boards\rv32m1_vega\demo_apps\hello_world\ri5cy\riscvgcc\debug\riscv32-unknown-elf-gdb hello_world.elf
GNU gdb (GDB) 7.12.50.20170505-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=riscv32-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello_world.elf...done.
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
__VECTOR_TABLE () at C:\_DDM\RISCVWorkspace\rv32m1_sdk_riscv\devices\RV32M1\gcc\startup_RV32M1_ri5cy.S:67
67      jal x0, Reset_Handler
(gdb) load
Loading section .text, size 0x30f2 lma 0x0
Loading section .data, size 0x18 lma 0x30f2
Loading section .vectors, size 0x90 lma 0xffff00
Start address 0x0, load size 12698
Transfer rate: 15 KB/sec, 4232 bytes/write.
(gdb) monitor reset
JTAG tap: rv32m1.cpu tap/device found: 0x249511c3 (mfg: 0x0e1 (Wintec Industries), part: 0x4951, ver: 0x2)

```

图 16 通过 OpenOCD + GDB 运行完成

除此之外，还可以通过以下指令来进行调试：

- **monitor resume** : 重新开始运行 CPU 核。
- **monitor halt** : 暂停 CPU 核。

6. 参考文献

以下参考文献均可从 www.open-isa.cn 网站获取：

- VEGA-Lite-SCH: 原理图
- VEGA-Lite-LAYOUT: PCB 元器件分布图
- 织女星开发板硬件用户指南.pdf
- RV32M1RM: 芯片参考手册
- RV32M1DS: 芯片数据手册
- 织女星开发板嵌入式开发环境搭建.pdf

7. 修订版本

版本号	日期	修改内容
0	04/2019	第一版

VEGA*