

risc-v IDE调试

发布版本：1.0

作者邮箱：jason.zhu@rock-chips.com

日期：2018.12

文件密级：公开资料

前言

概述

本文参考《FreedomStudio IDE使用说明》，供risc-v调试使用。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

产品版本

修订记录

日期	版本	作者	修改说明
2018-12-27	V1.0	Jason Zhu	初始版本

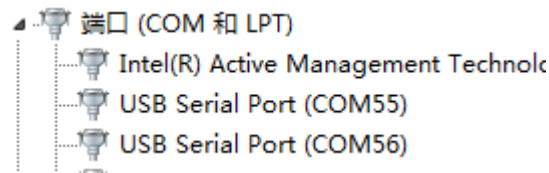
risc-v IDE调试

- Hifive1调试
 - 安装HIFIVE1上的FTDI驱动
 - 基于IDE的工程建立与调试
 - SCR1调试
 - 安装调试器驱动
 - 基于IDE的工程建立与调试
-

Hifive1调试

安装HIFIVE1上的FTDI驱动

HiFive插到电脑，windows默认把FTDI识别成两个UART，如下：



这会导致IDE无法连接设备进行调试，需要把其中一个Dial RS232-HS，解决方法如下：

1. 从<https://sourceforge.net/projects/libusbk/>下载libusbk
2. 安装libusbk
3. 在安装目录下找到libusbK-inf-wizard.exe，运行。
4. 选择安装的libusbk，下一步

USB Inf

InfWizard creates reusable USB installation packages customized for a specific to usb dev

To begin, Select the windows kernel driver th your USB applications will target: [Open Exter Package](#) | [Download](#)

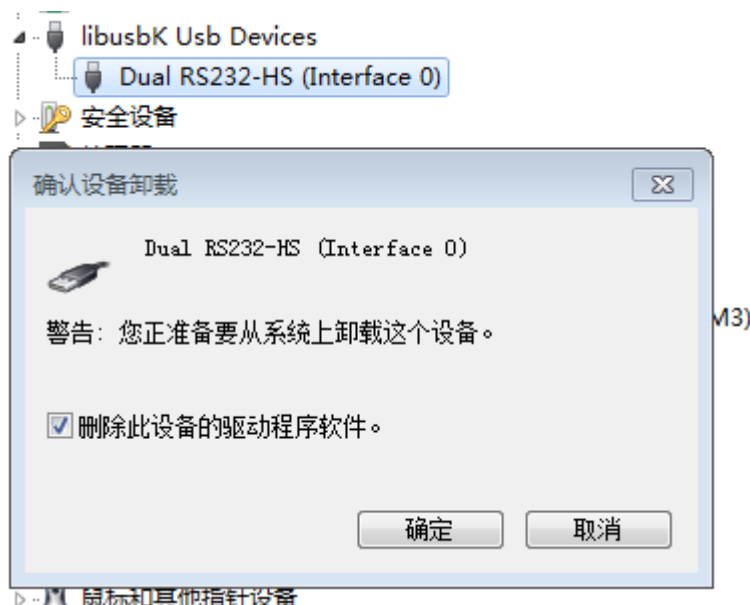
Kernel Driver Package(s)

WinUsb v6.1.7600.16385 (02/23/2013)

libusbK v3.0.7.0 (04/28/2014)

libusb0 v1.2.6.0 (02/23/2013)

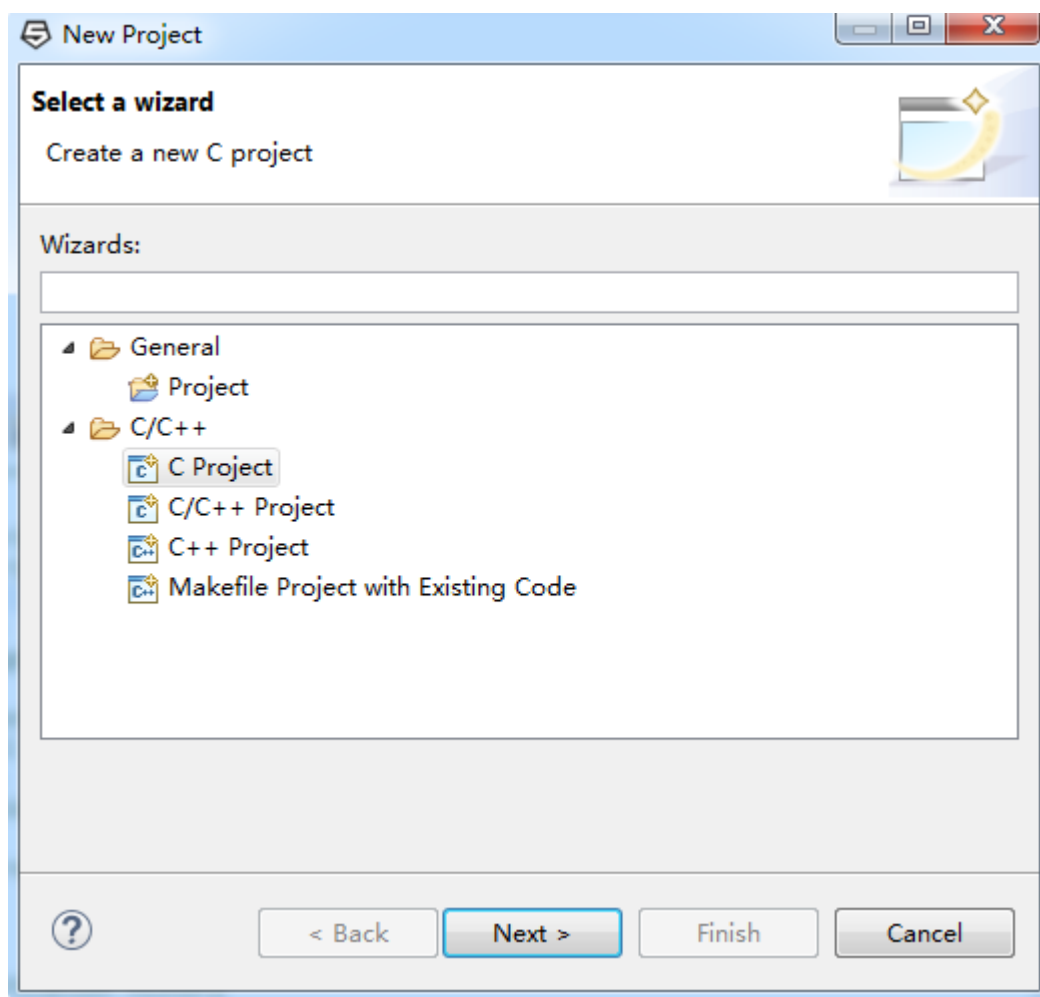
5. 选中Show All Device，可以看到三个Dual RS232-HS。第一个是父节点，第二和第三个是两个子节点。如果选中父节点，则生成的驱动会应用到父节点上。如果选中某个子节点，则生成的驱动只应用到相应的子节点上。HiFive1的第一个FTDI口用作TAG，第二个FTDI口作UART。所以生成驱动时，选择第一个子节点就可以。
6. 运行生成目录Dual_RS232-HS_Interface_0下的InstallDriver.exe
7. 最后可以在设备管理器上查找到对应的设备，如下：



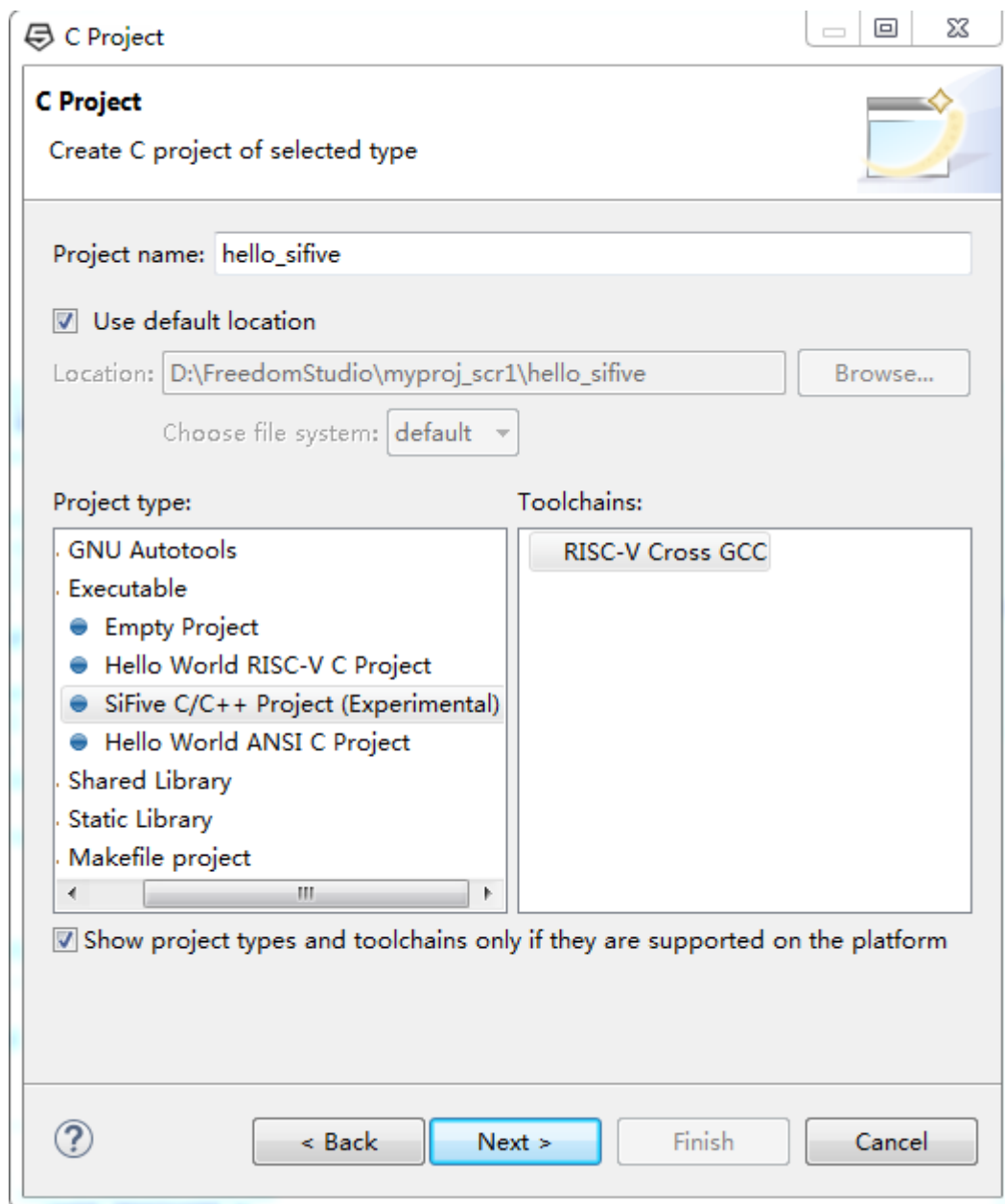
至此，驱动安装完成。

基于IDE的工程建立与调试

1. File->New->Project->C Project



2. 点开C Project，选择SiFive C/C++ Project，Project name输入工程名，如hello_fifive：



3.点击next

C Project

Project settings

Select the SiFive board and define project options.

Board: HiFive1

Content: Blinky (blink a led)

Use system calls: POSIX (system calls implemented by application code)

Trace output: UART0 (via FTDI)

Check some warnings ☒

Check most warnings ☐

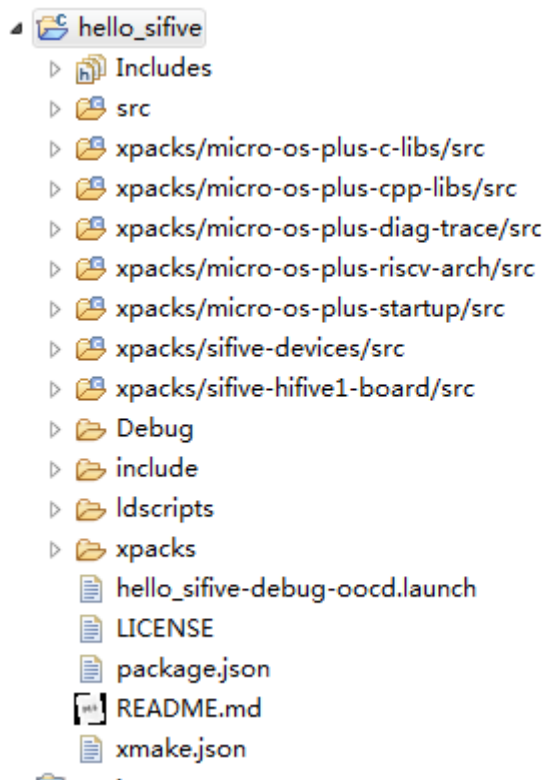
Enable -Werror ☐

Use -Og on debug ☐

Use newlib nano ☒

? < Back Next > Finish Cancel

4.后续都采用默认操作，点击next，最后点击finish。



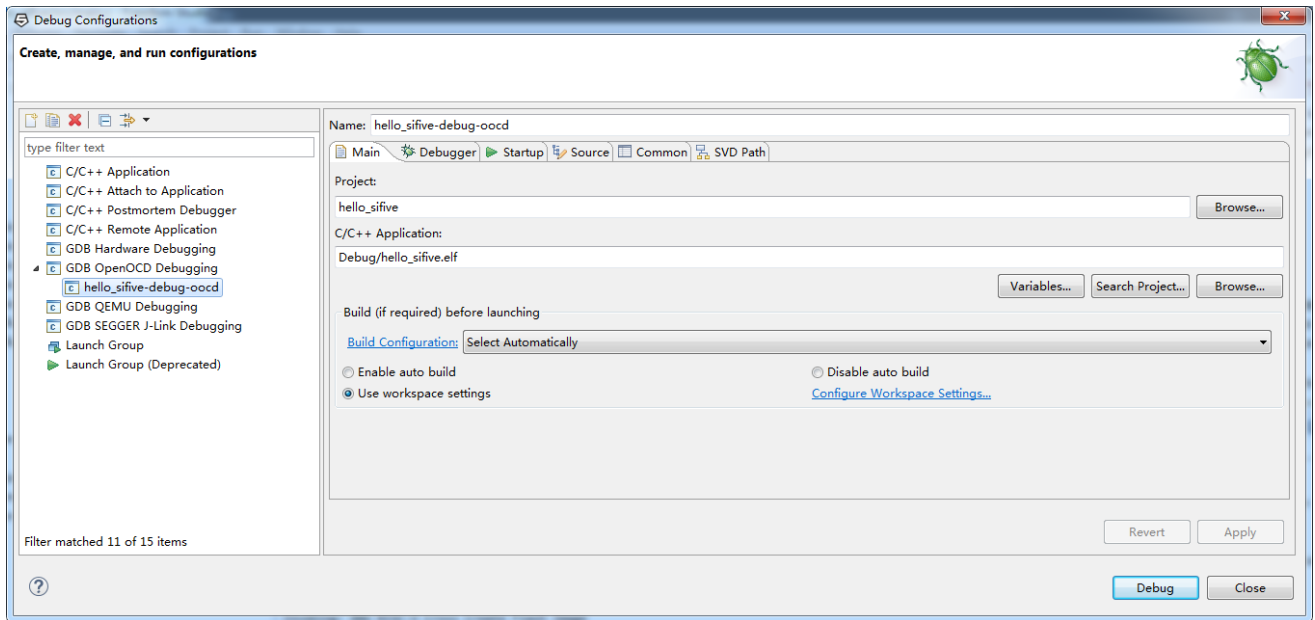
5. 点击hello_sifive，右键点击build project，编译工程。

```
CDT Build Console [hello_sifive]
./xpacks/sifive-devices/src/fe310/device-interrupts.o ./xpacks/sifive-devices/src/artty/e51/device-interrupts.o
./xpacks/sifive-devices/src/artty/e31/device-interrupts.o ./xpacks/sifive-devices/src/device-functions.o ./xpacks/sifive-devices/src/plic-functions.o
./xpacks/micro-os-plus-startup/src/startup.o ./xpacks/micro-os-plus-riscv-arch/src/arch-functions.o ./xpacks/micro-os-plus-riscv-arch/src/reset-entry.o
./xpacks/micro-os-plus-riscv-arch/src/trap-entry.o ./xpacks/micro-os-plus-riscv-arch/src/traps.o ./xpacks/micro-os-plus-diag-trace/src/trace.o
./xpacks/micro-os-plus-cpp-libs/src/cxx.o ./xpacks/micro-os-plus-c-libs/src/stdlib/assert.o ./xpacks/micro-os-plus-c-libs/src/stdlib/atexit.o
./xpacks/micro-os-plus-c-libs/src/stdlib/exit.o ./xpacks/micro-os-plus-c-libs/src/stdlib/init-fini.o ./xpacks/micro-os-plus-c-libs/src/_sbrk.o
./xpacks/micro-os-plus-c-libs/src/c-syscalls-empty.o ./src/initialize-hardware.o ./src/interrupts-handlers.o ./src/led.o ./src/main.o ./src/newlib-syscalls.o
./src/sysclock.o
Finished building target: hello_sifive.elf

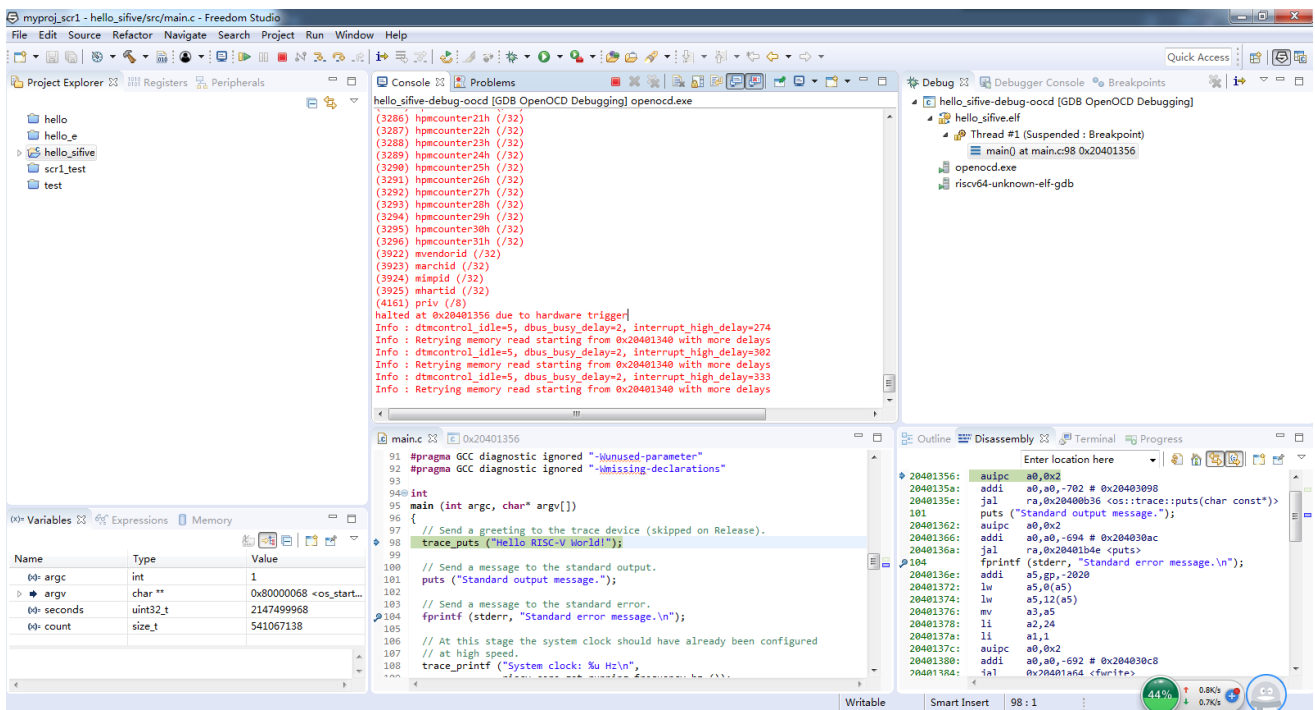
Invoking: GNU RISC-V Cross Create Flash Image
riscv64-unknown-elf-objcopy -O ihex "hello_sifive.elf" "hello_sifive.hex"
Finished building: hello_sifive.hex

Invoking: GNU RISC-V Cross Print Size
riscv64-unknown-elf-size --format=berkeley "hello_sifive.elf"
text data bss dec hex filename
12930 168 2160 15258 3b9a hello_sifive.elf
Finished building: hello_sifive.siz
```

6. 调试：连接板子，点击hello_sifive，右键点击Debug configurations



7.点击debug，调试工程。



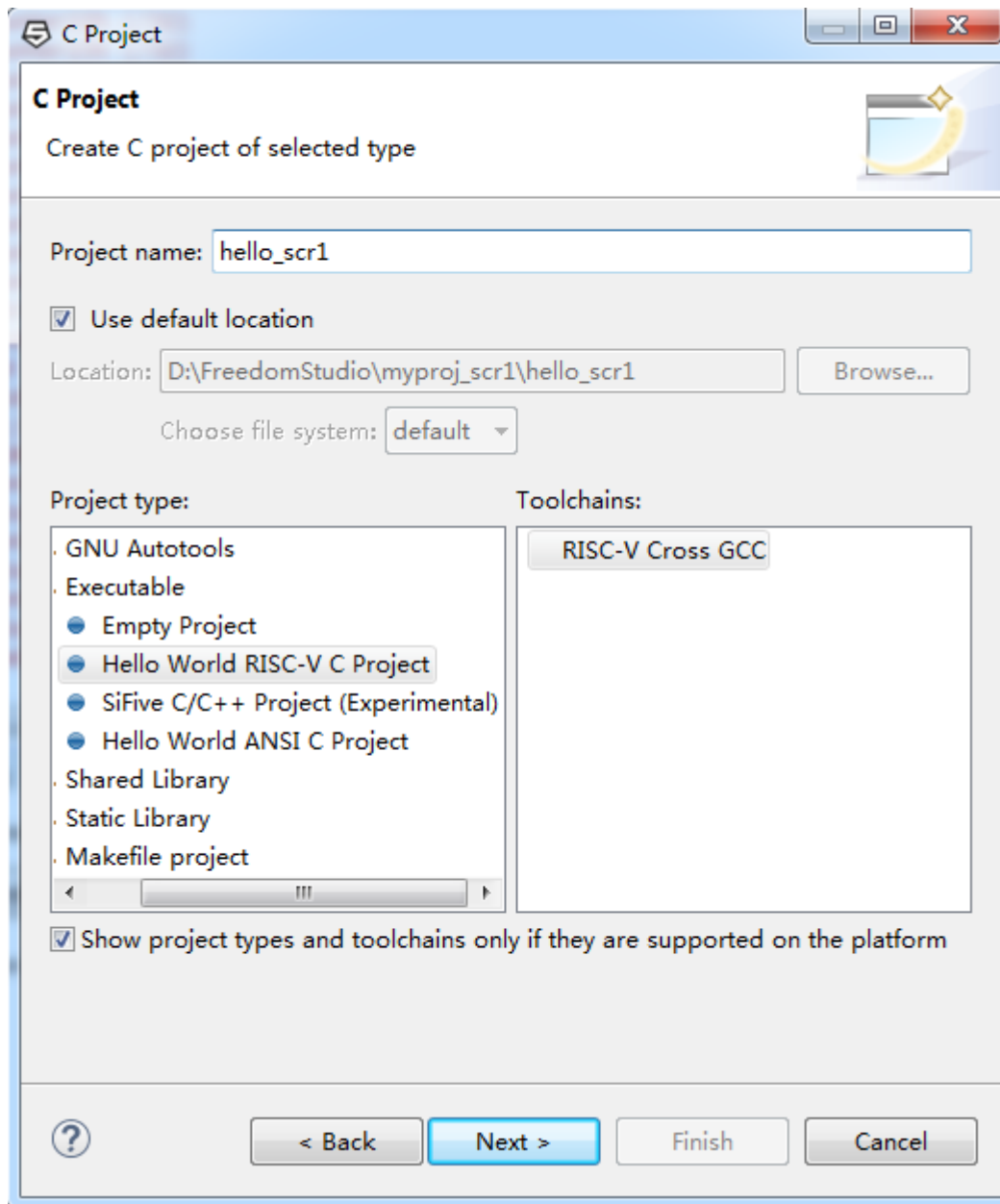
SCR1调试

安装调试器驱动

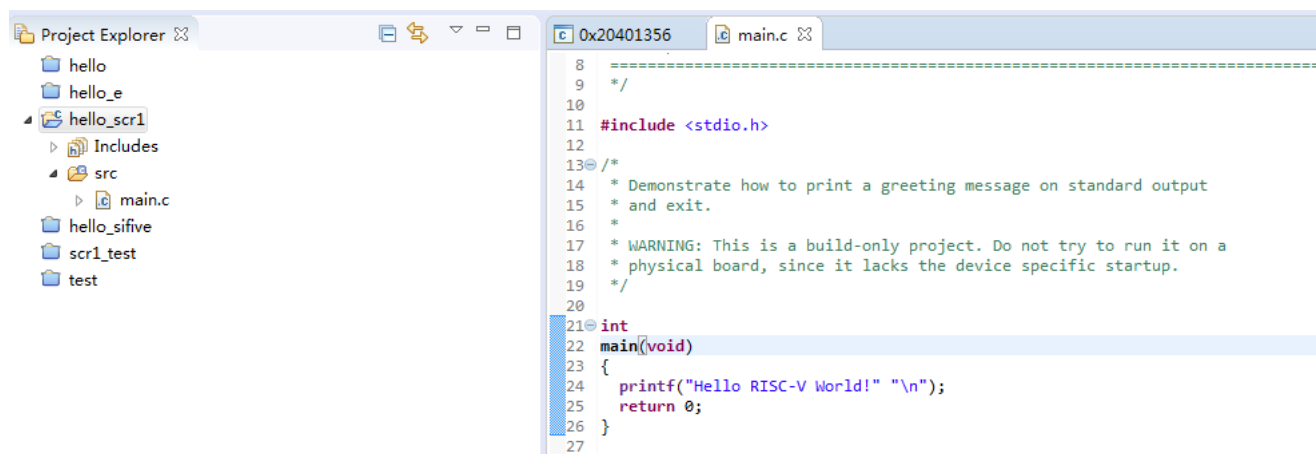
参考安装HIFIVE1上的FTDI驱动章节安装驱动。

基于IDE的工程建立与调试

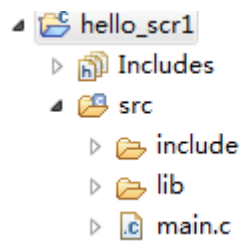
1.File->New->Project->C Project，Project name输入hello_scr1，选择Hello World RISC-V C Project



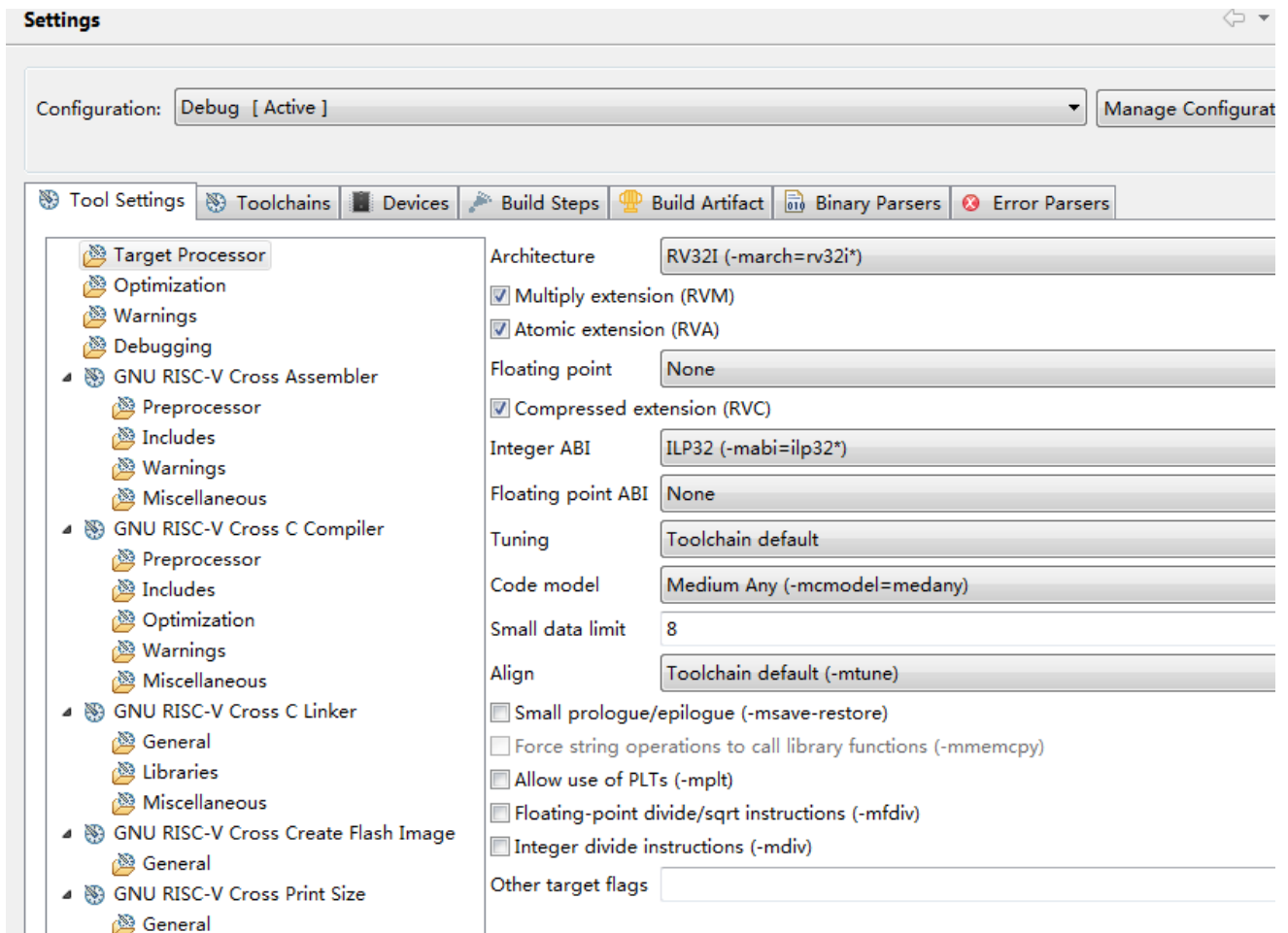
2.一直点击next , finish



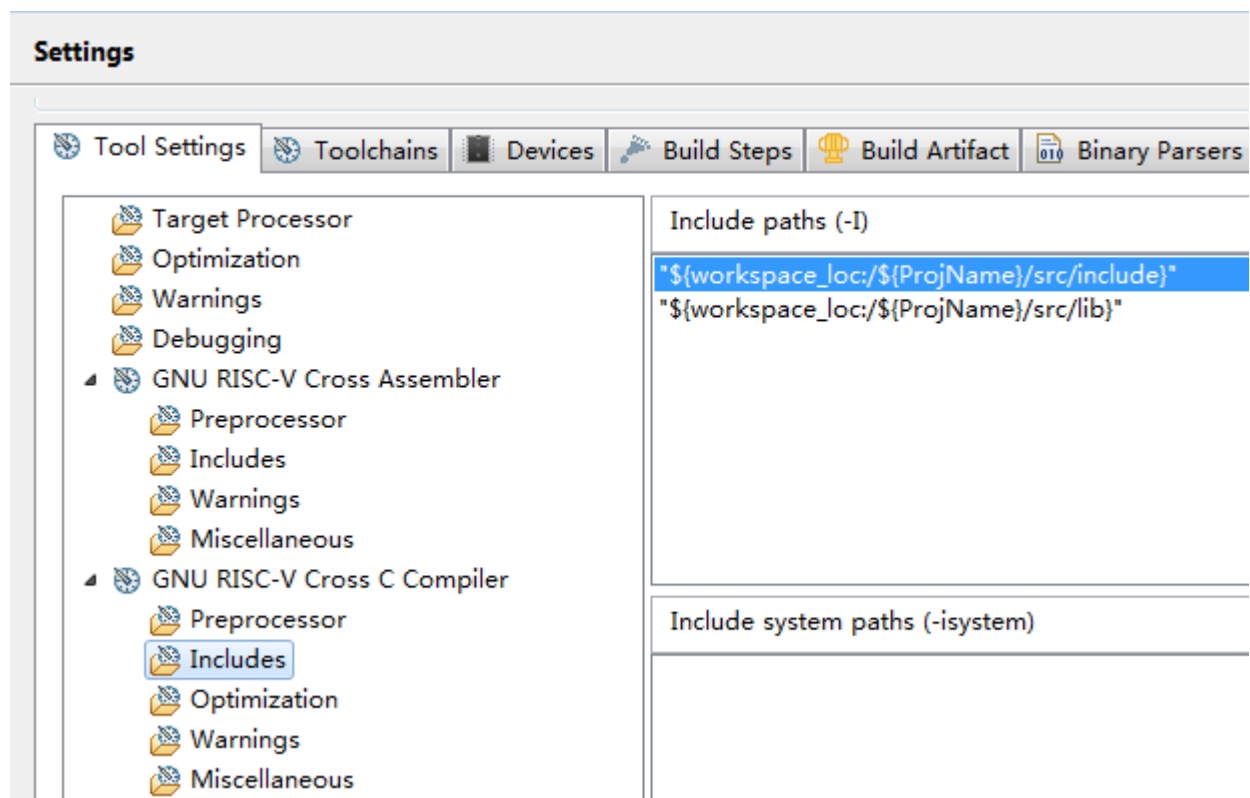
3.拷贝hello_ncore内的include与lib文件夹到工程目录src下，然后点击右键Refresh



4. 点击工程名右键Properties，选择C/C++ Build->Settings，如下配置Target Processor



5. 选择GNU RISC-V Cross C Compiler->Includes，配置如下



6.选择GNU RISC-V Cross C Linker->General , 配置如下

Settings

Configuration: Debug [Active]

Manage Configurations...

Tool Settings

Toolchains

Devices

Build Steps

Build Artifact

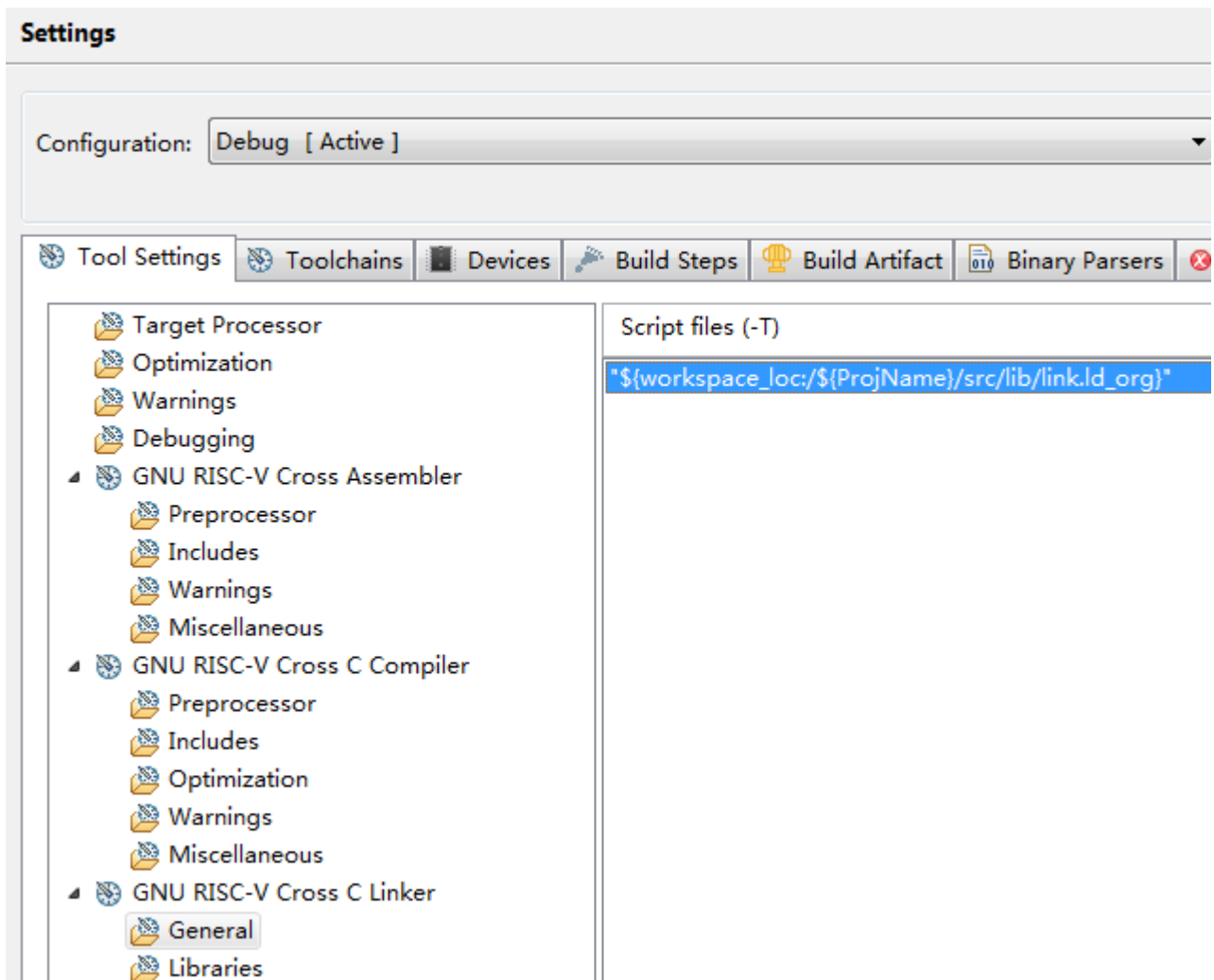
Binary Pa

- Target Processor
- Optimization
- Warnings
- Debugging
- GNU RISC-V Cross Assembler
 - Preprocessor
 - Includes
 - Warnings
 - Miscellaneous
- GNU RISC-V Cross C Compiler
 - Preprocessor
 - Includes
 - Optimization
 - Warnings
 - Miscellaneous
- GNU RISC-V Cross C Linker
 - General
 - Libraries
 - Miscellaneous
- GNU RISC-V Cross Create Flash Image
 - General
- GNU RISC-V Cross Print Size
 - General

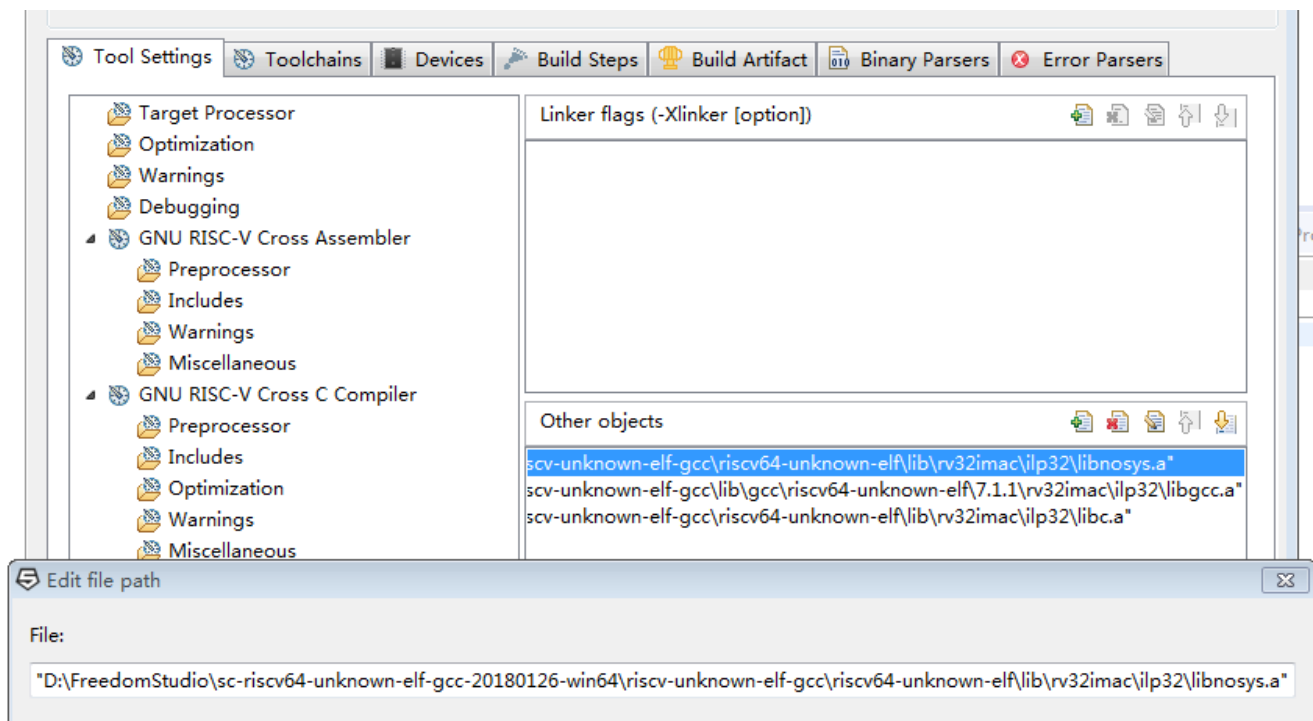
Script files (-T)



- ☒ Do not use standard start files (-nostartfiles)
- ☒ Do not use default libraries (-nodefaultlibs)
- ☐ No startup or default libs (-nostdlib)
- ☒ Remove unused sections (-Xlinker --gc-sections)

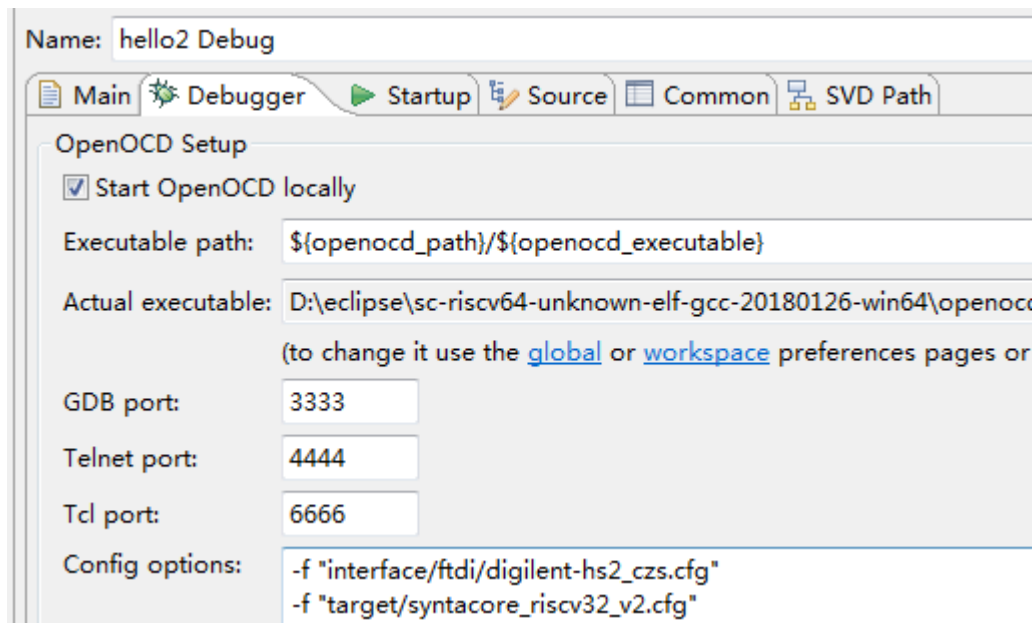


7.选择GNU RISC-V Cross C Linker->Miscellaneous

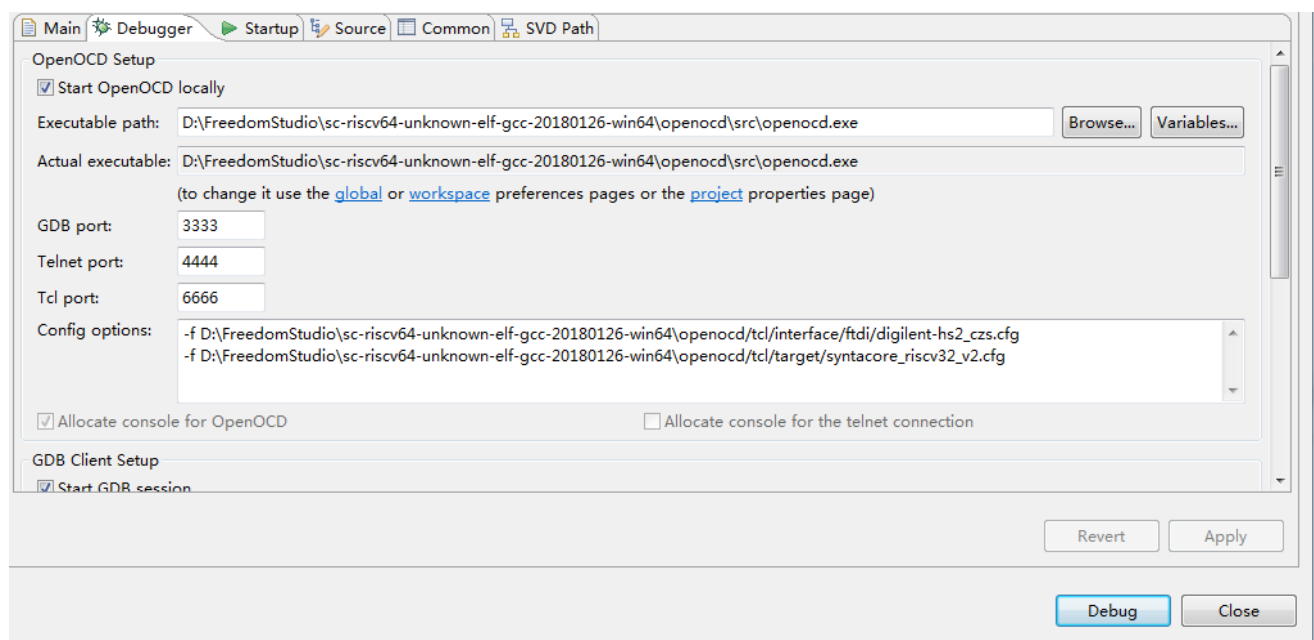


8.将main.c中的printf改为sc_printf, 同时添加#include "sc_print.h", 编译成功。

9.调试：连接板子，点击hello_scr1，右键点击Debug configurations



如果连接错误，可以尝试使用绝对路径，配置如下：



10.Debug