# RISCV RubySOC

陈弘 Bruce Chen
2020-11-17

# 版本

| 日期 | 版本 | 版本描述 |
| --- | --- | --- |
| 2020/11/17 | V1.0 | 初稿发布 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# 主要内容

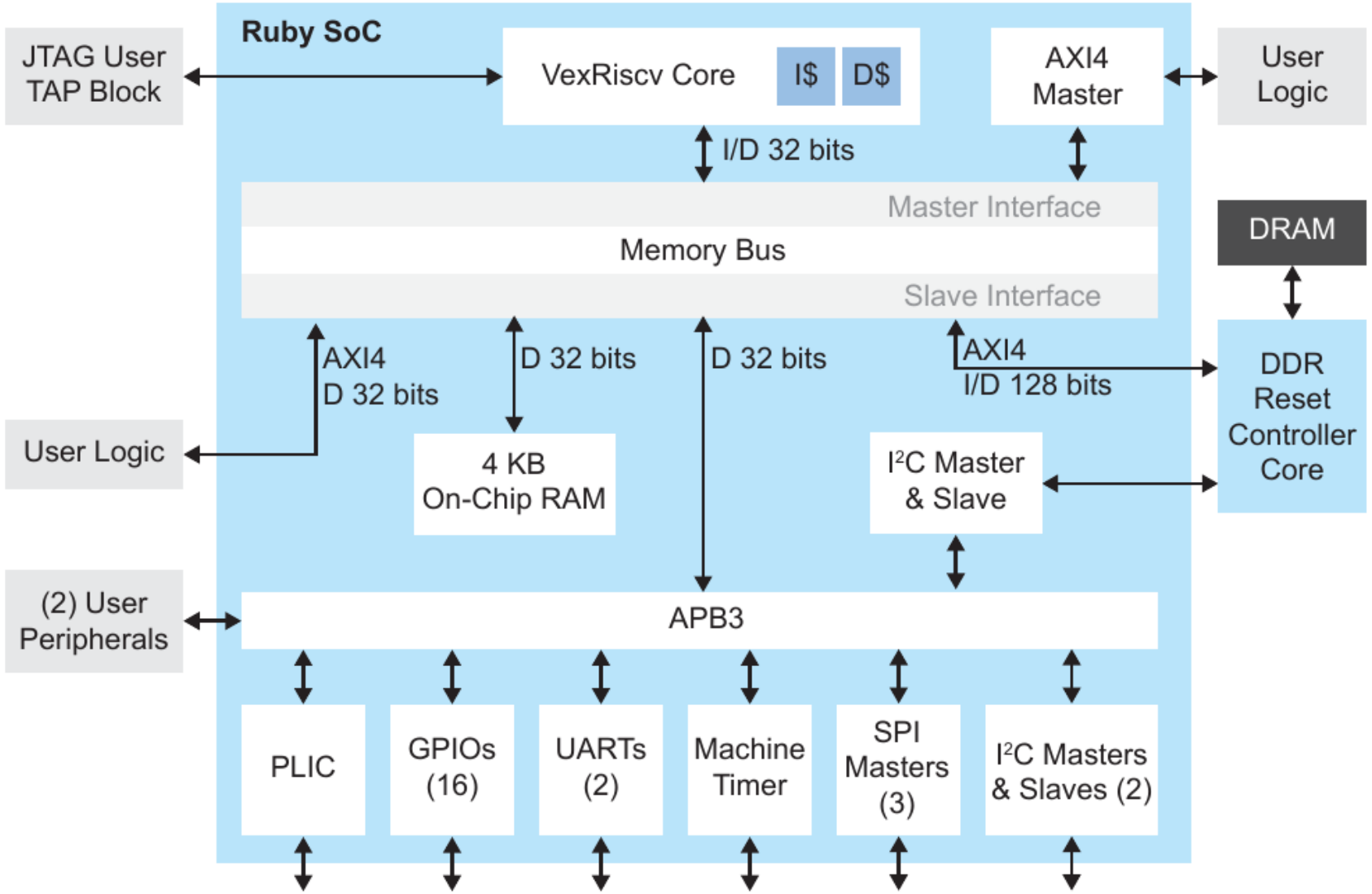
- RubySOC Demo硬件工程
- RubySOC Demo软件工程
- RubySOC 开发环境搭建
- RubySOC Demo
- APP 固化

# RISCV RubySOC

## ——硬件工程

陈弘 Bruce Chen
2020-12-13

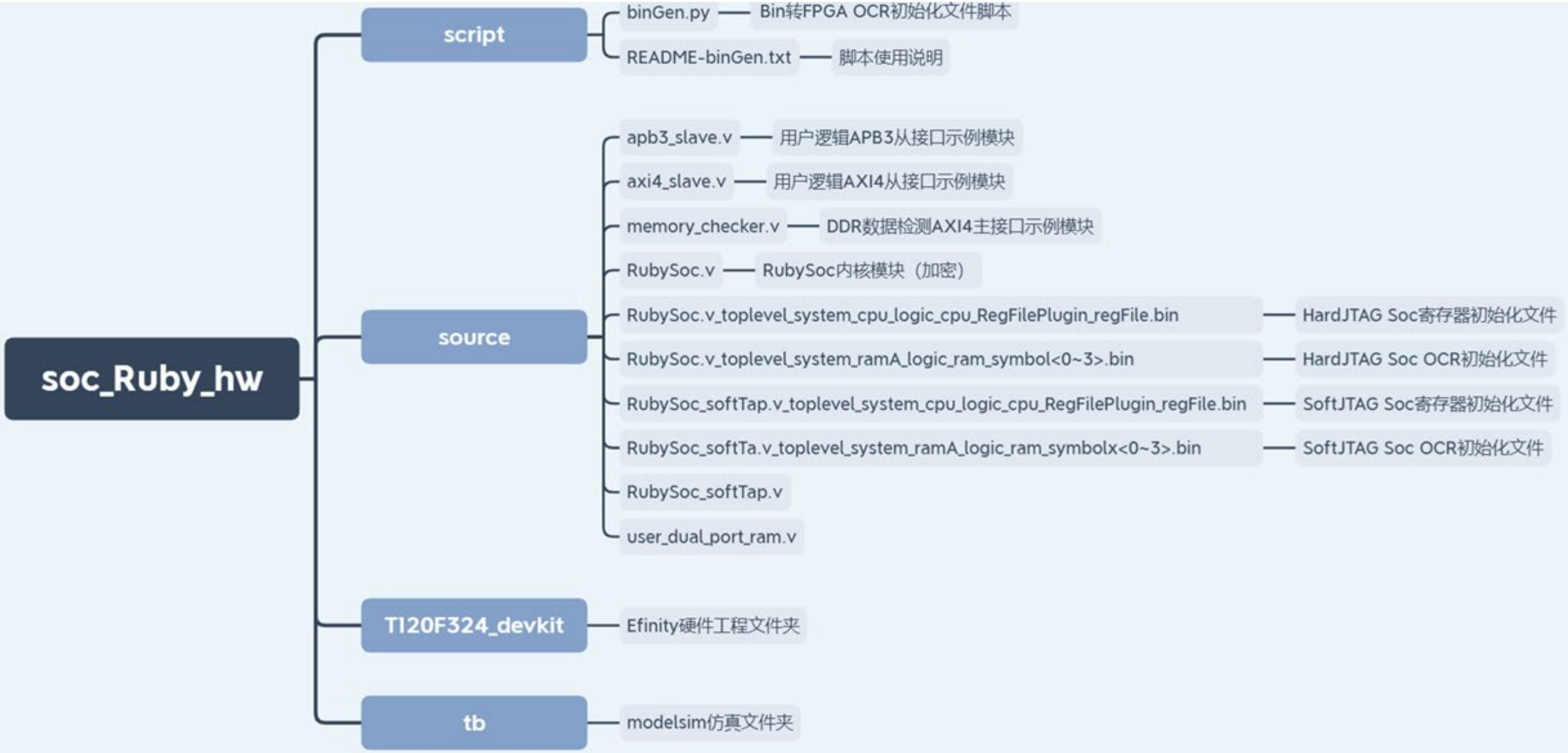
# RubySOC Core Block Diagram



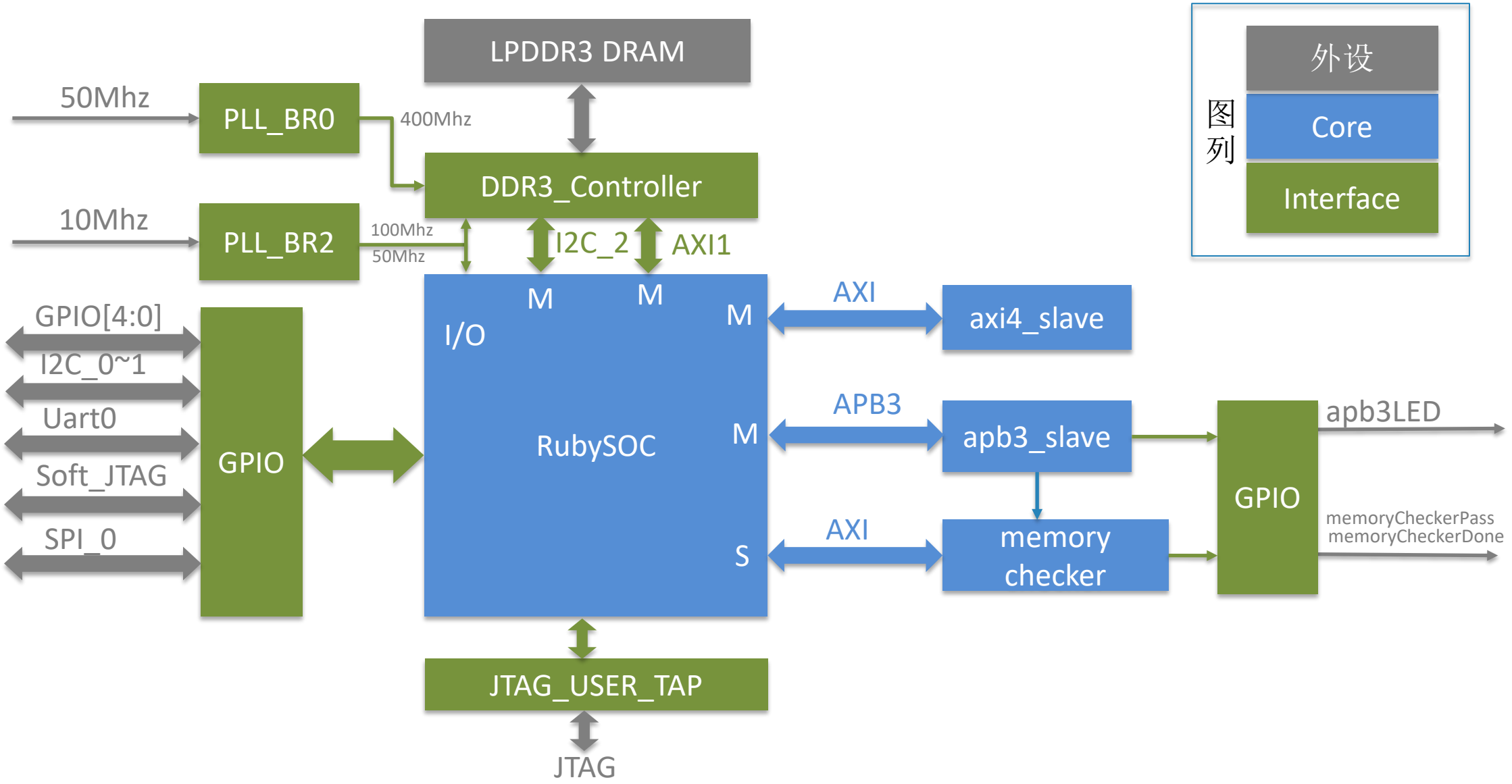
# RubySOC Standard Core Feature

- VexRiscv processor with 5 pipeline stages (fetch, decode, execute, memory, and write back), interrupts and exception handling with machine mode
  - 4 KB data cache
  - 4 KB instruction cache
- 50 MHz system clock frequency
  - 1.16 DMIPS/MHz
- 4 KB on-chip RAM with boot loader for SPI flash
- 1 AXI master channel to access the DDR memory
- 1 AXI master channel to user logic
- 1 AXI slave channel to user logic
- JTAG debug port
  - Hard JTAG though FPGA JTAG TAP
  - Soft JTAG though FPGA GPIO
- APB3 peripherals:
  - 16X GPIOs
  - 3x I2C masters and slaves
  - 1X 64bit Machine timer
  - PLIC
  - 3x SPI flash masters with a maximum clock frequency of 25 MHz
  - 2x UARTs
  - 2x slave user peripheral

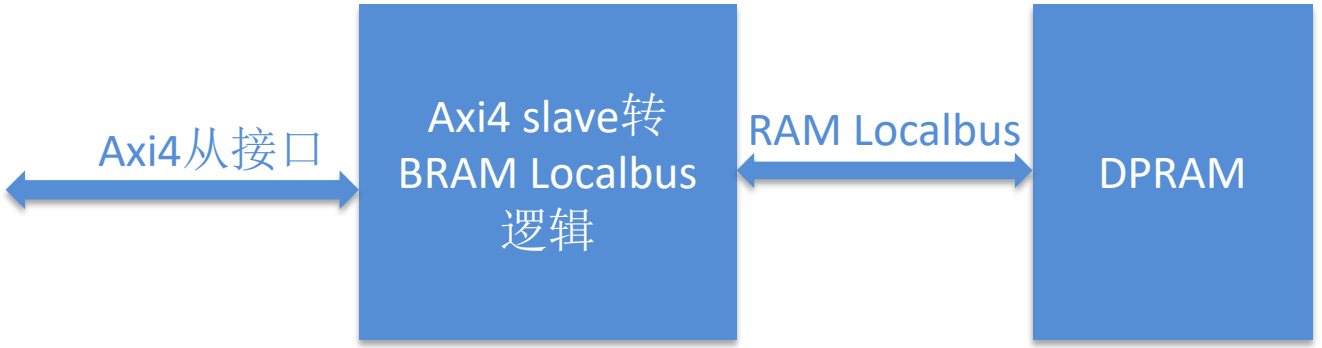
# RubySOC Demo 硬件工程开发包



# RubySOC Demo for T120F324 Devkit



# axi4_slave模块




提供RubySoc通过AXI访问用户逻辑的示范代码

# apb3_slave模块

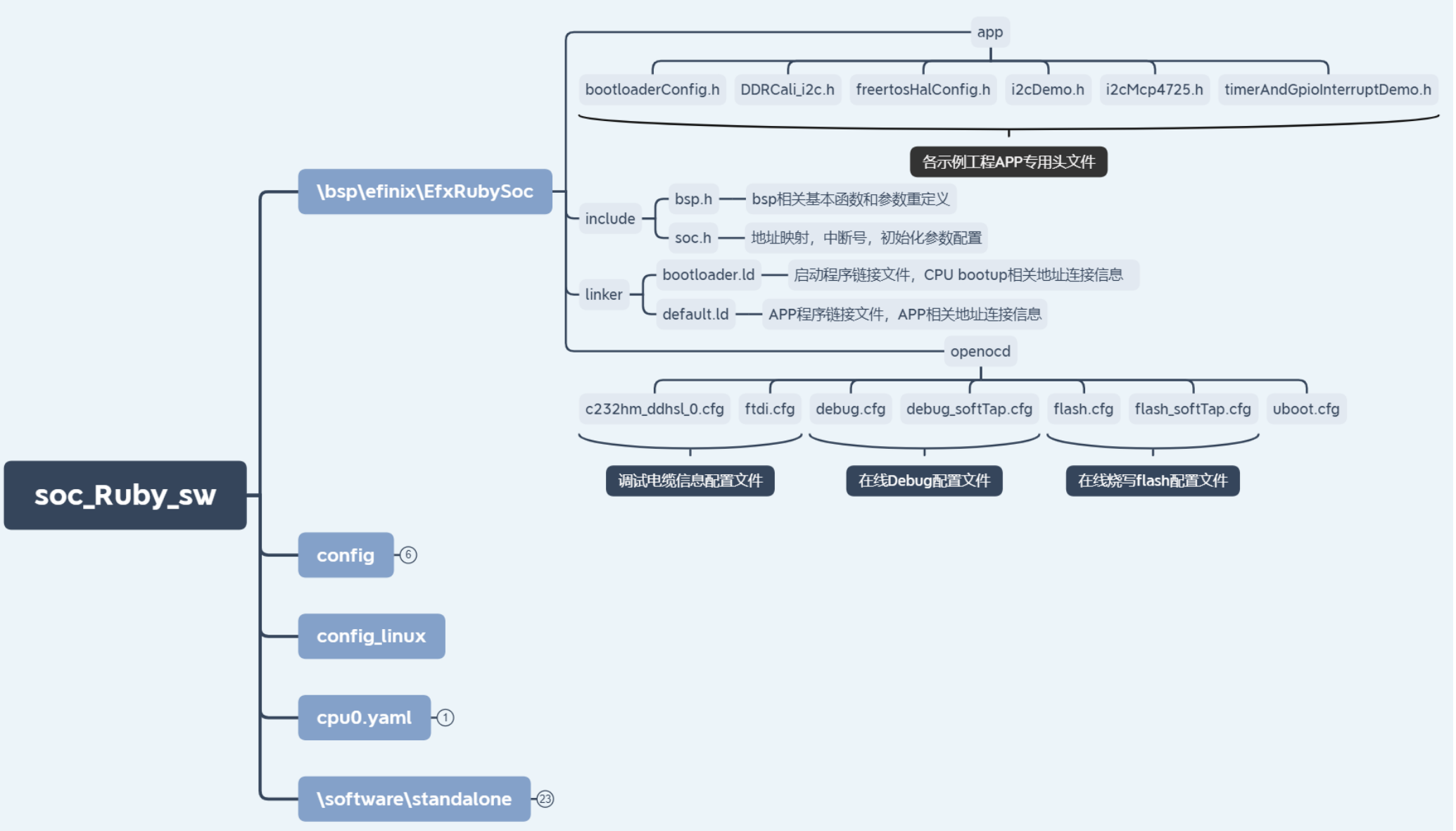

提供RubySOC通过APB3访问用户逻辑的示范代码

# RISCV RubySOC

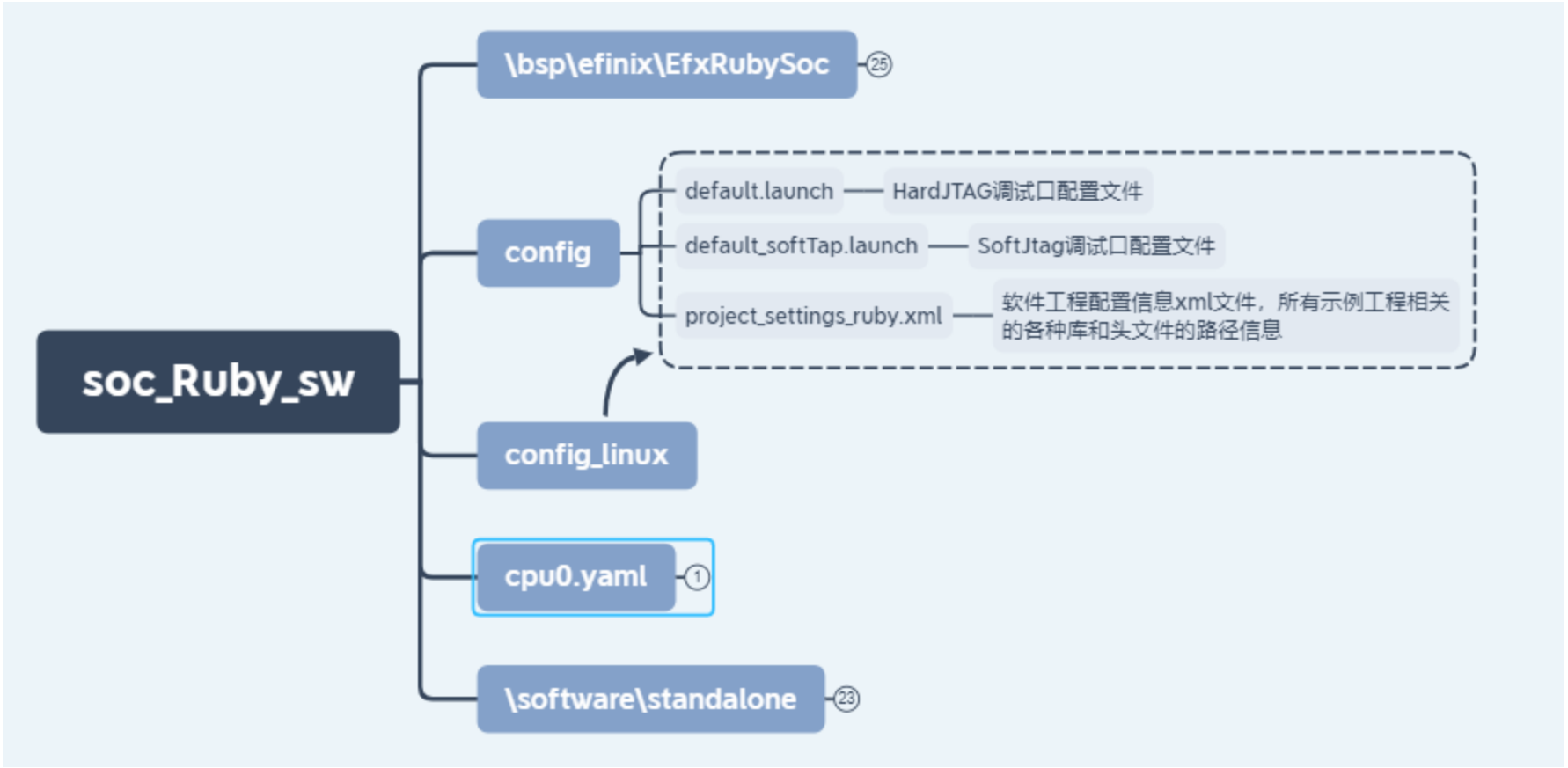
## ——软件工程

陈弘 Bruce Chen
2020-12-13

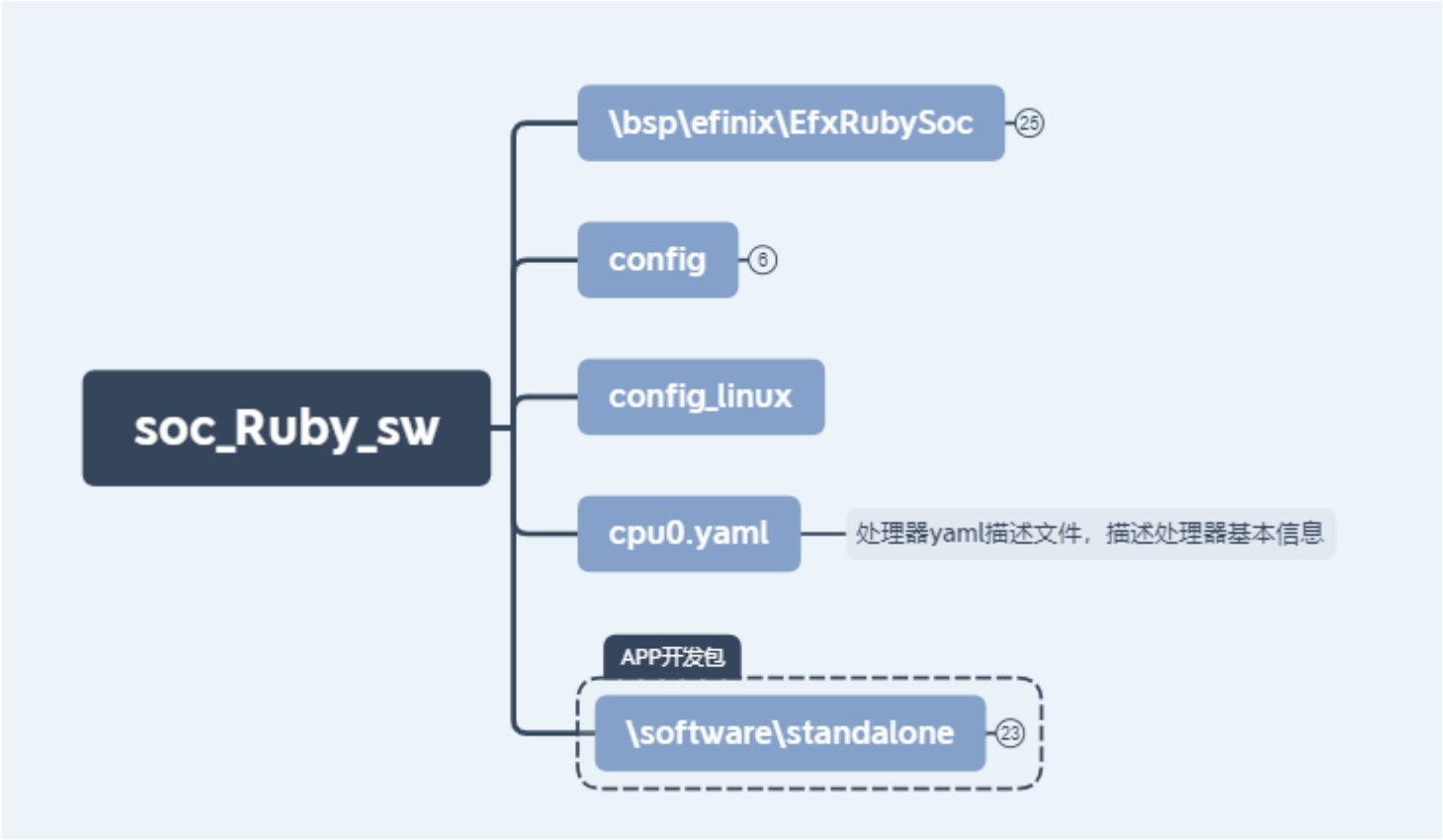
# BSP——Board Support Package



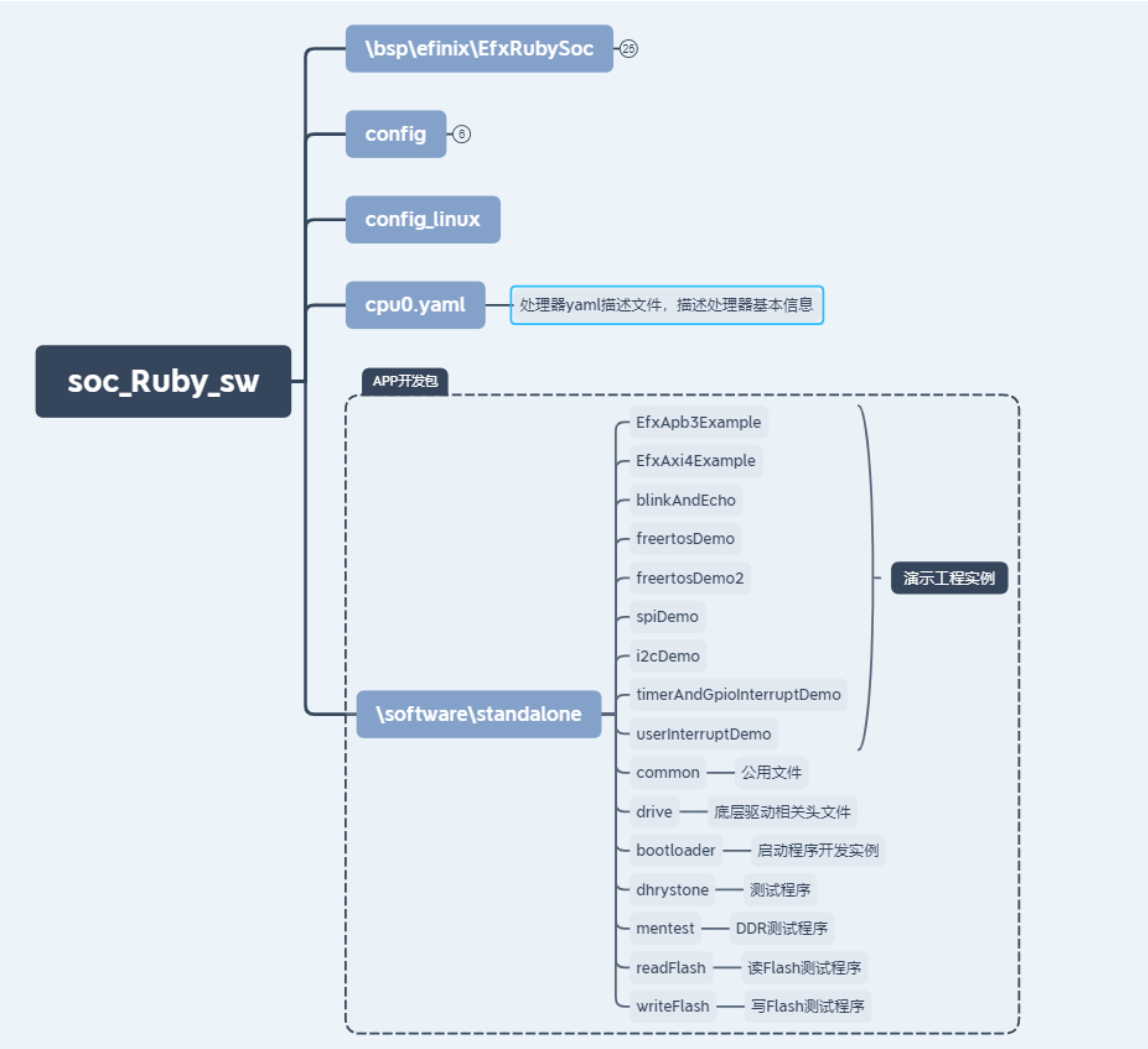
# config——Eclipse编译调试环境配置文件



# cpu0.yaml——处理器yaml 描述文件



# standalone——APP开发包



# RISCV RubySOC

## ——开发环境搭建

陈弘 Bruce Chen
2020-12-13

# RubySOC 开发环境搭建

- Eclipse 工作环境设置
- 新建工程
- 导入项目设置信息
- 设置debug编译开关
- 第一次编译
- 导入debug配置信息
- 调整debug配置信息

# 准备工作

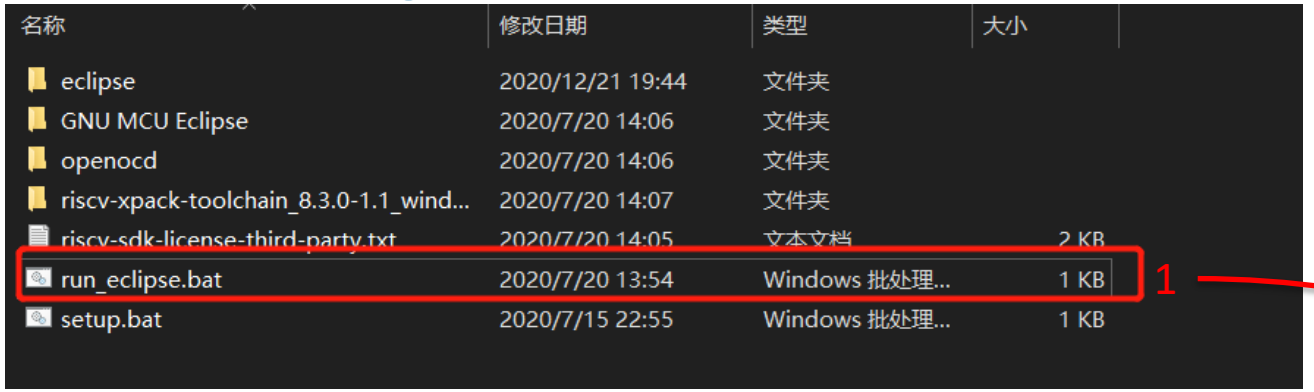
- IP core
  - efx_ruby_riscv_soc-v1.2.zip
- 软件开发套件SDK
  - riscv_sdk_windows-v1.1.zip
  - 开发套件和IP core版本相互独立不需要对应
- Jave运行环境
  - jre-8u251-windows-x64.exe

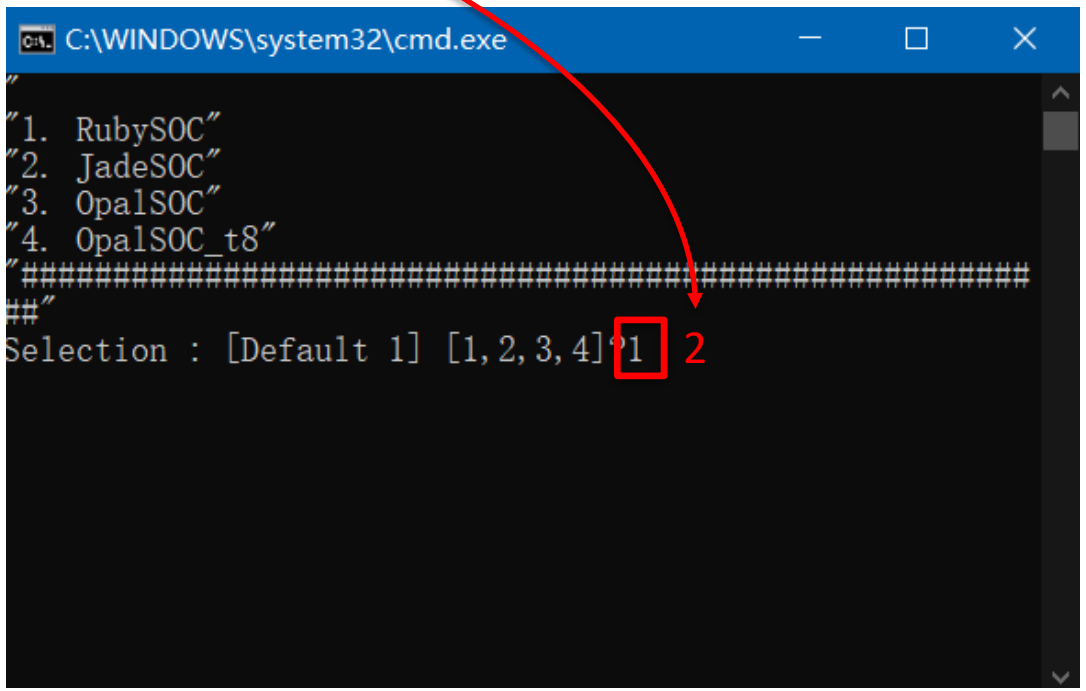
百度网盘链接：
https://pan.baidu.com/s/1vgfyciVWN1ICgnCFsfnb-g
百度网盘提取码：0ht3

# RISC-V SDK

- **Eclipse MCU**—Open-source Java-based development environment that uses plug-ins to extend and customize its functionality. The GNU MCU Eclipse plug-in lets you develop applications for ARM and RISC-V cores.
  Version: 2019.08.0
  Disk space required: 320 MB (Windows), 375 MB (Linux)
- **xPack GNU RISC-V Embedded GCC**—Open-source, prebuilt toolchain from the xPack Project.
  Version: 8.3.0-1.1
  Disk space required: 1.46 GB (Windows), 1.4 GB (Linux)
- **OpenOCD Debugger**—The open-source Open On-Chip Debugger (OpenOCD) software includes configuration files for many debug adapters, chips, and boards. Many versions of OpenOCD are available. The Elitestek RISC-V flow requires a custom version of OpenOCD that includes the VexRiscv 32-bit RISC-V processor.
  Version: 20200421
  Disk space required: 9.4 MB (Windows), 7.4 MB (Linux)
- **GNU MCU Eclipse Windows Build Tool (Windows Only)**—This open-source Windowsspecific package helps to manage build projects and includes GNU make.
  Version: 2.12-20190422-1053
  Disk space required: 3.8 MB

# 启动Eclipse

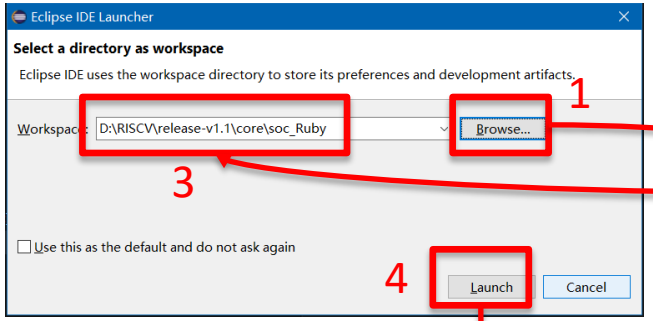


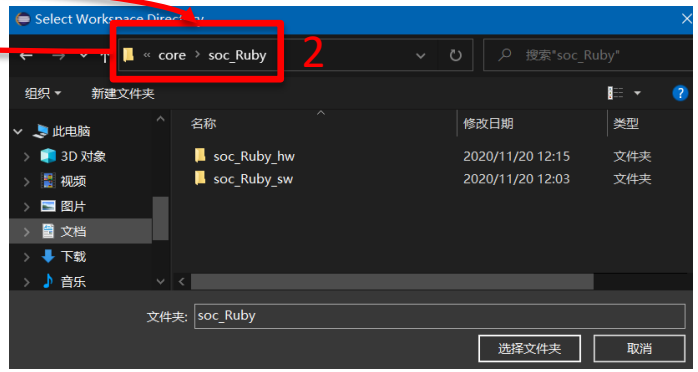



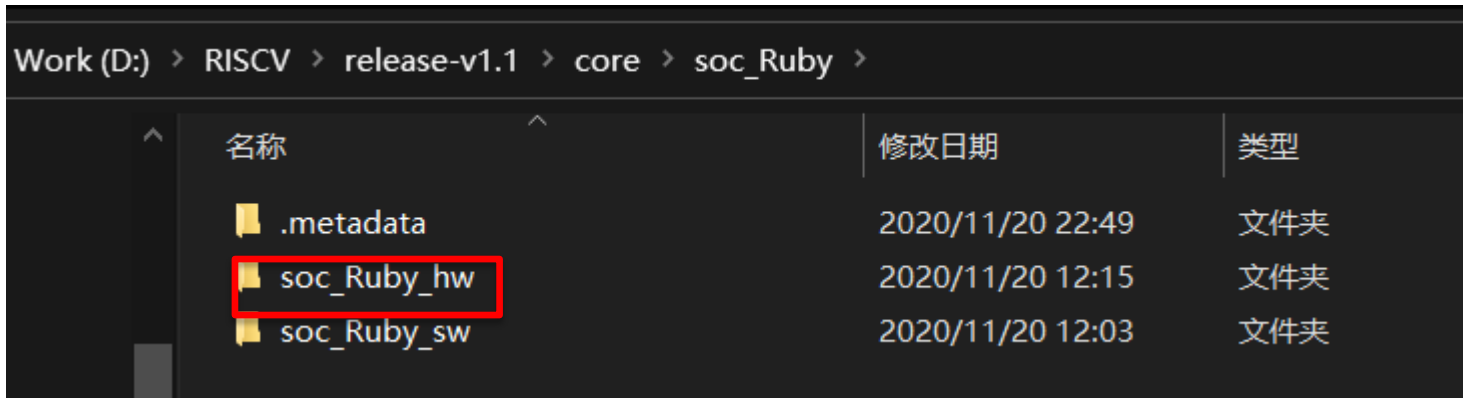

- 安装Jave运行环境
  - 运行jre-8u251-windows-x64.exe安装程序
- 安装软件开发套件SDK
  - 解压efx_ruby_riscv_soc-v1.2.zip到任意目录，不能含中文，空格和特殊字符

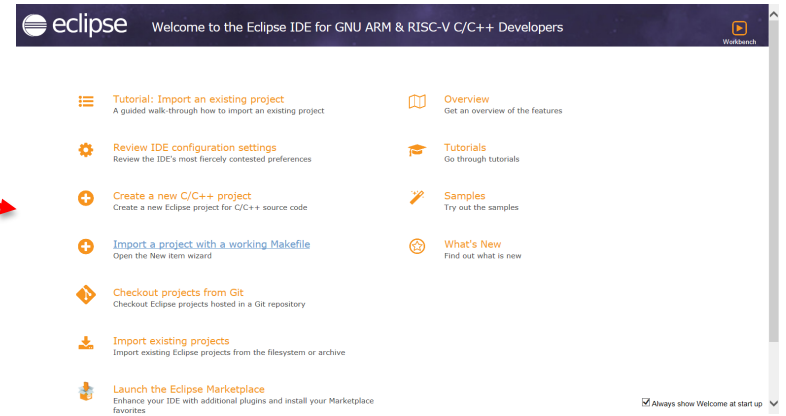
# Eclipse 工作环境设置

- Eclipse的Workspace一定要放在soc_Ruby目录下
- 否则相关所有设置的相对路径都会错
- 如果路径设置正确，打开文件浏览器，Eclipse工作环境.metadata文件夹会出现在soc_Ruby文件夹里，和软硬件工程文件夹并列

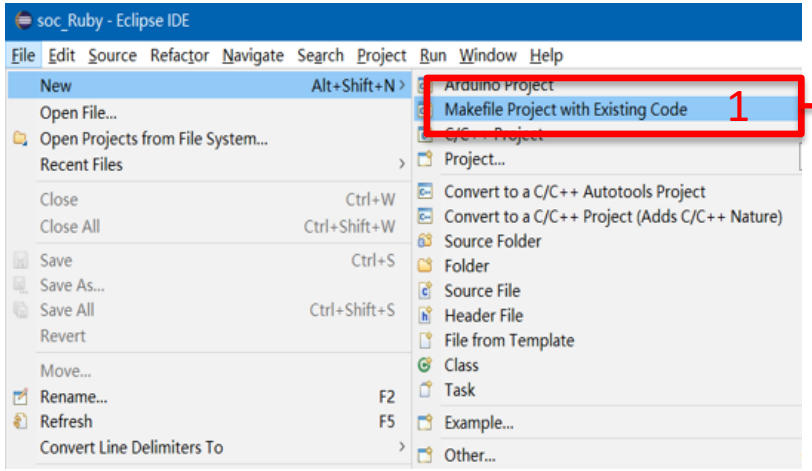


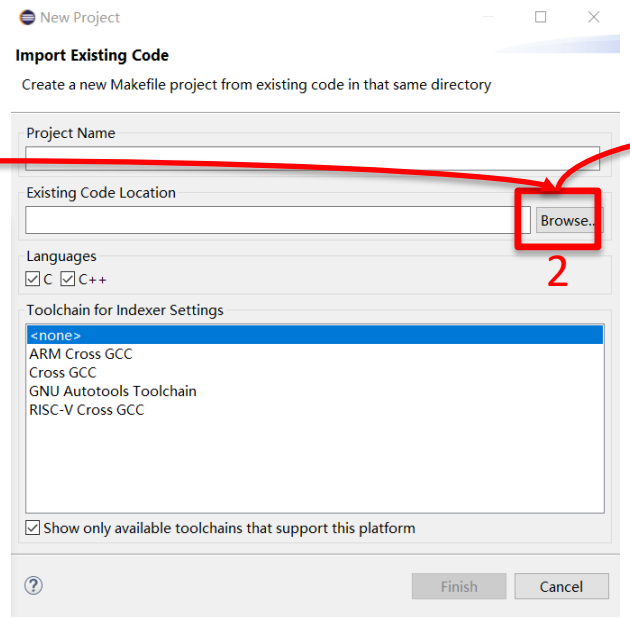



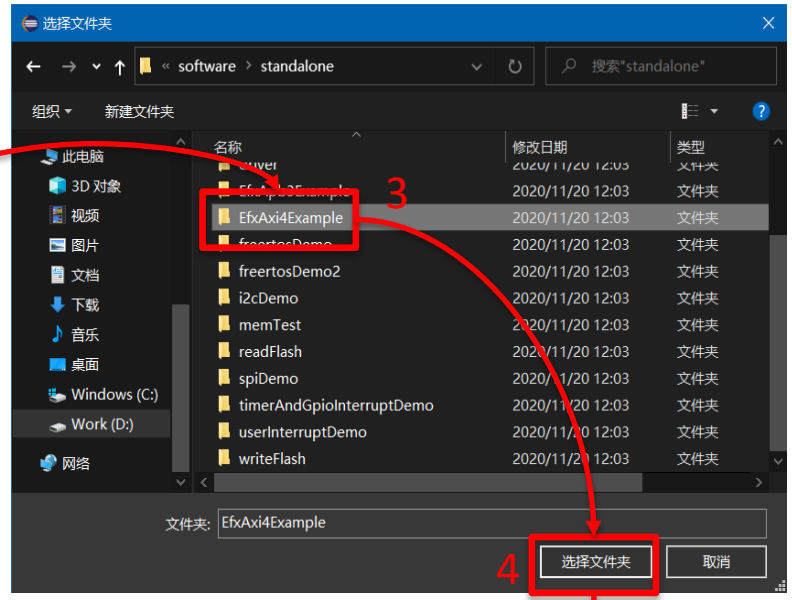



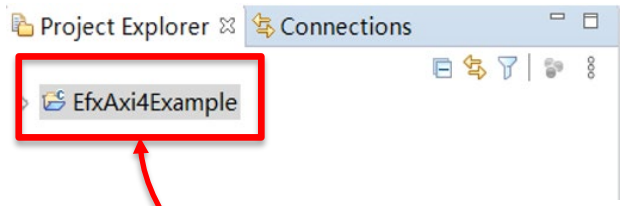



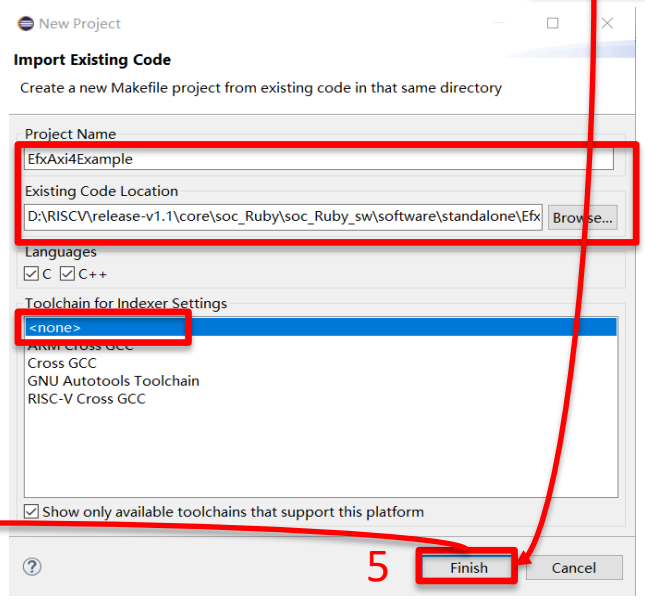
# 新建工程



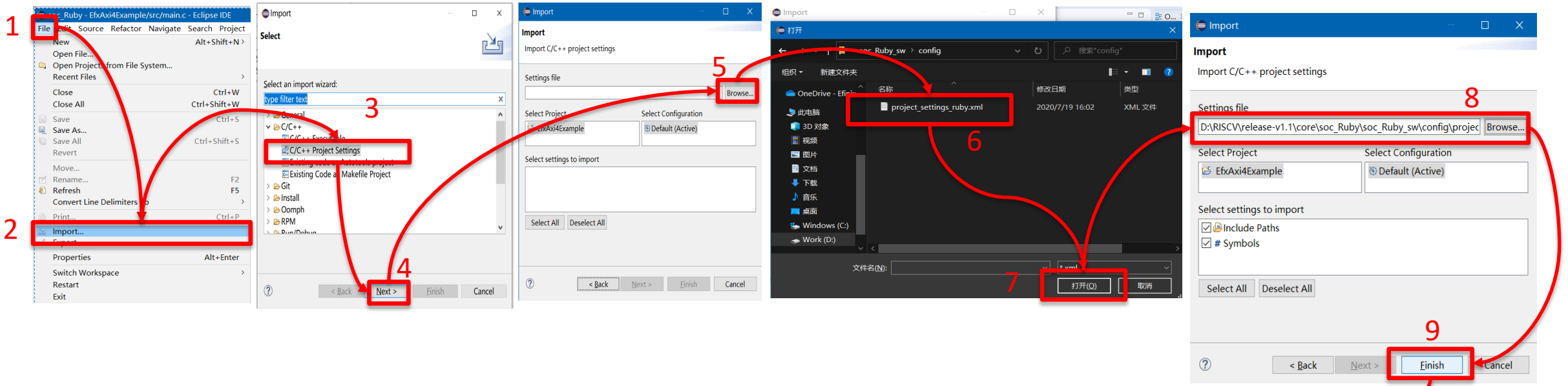



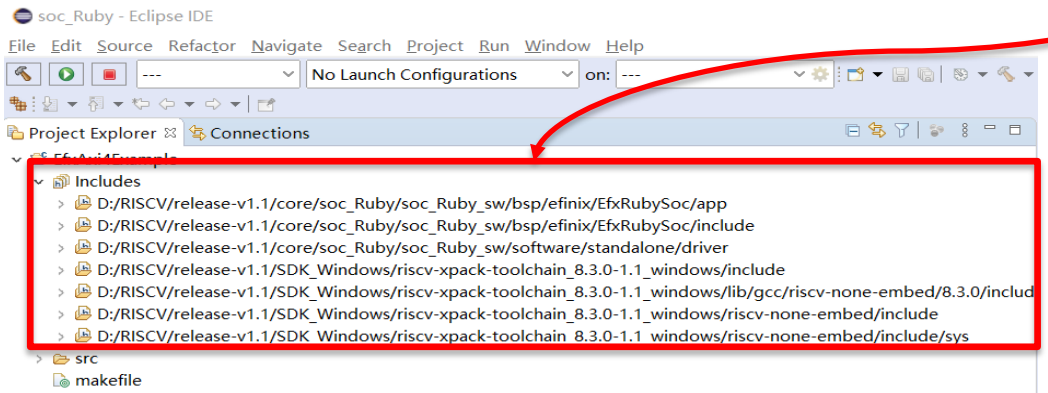




- 路径指定到standalone文件夹下的EfxAxi4Example子文件夹或者其它工程子文件夹上，不能再进入下一级。

# 导入项目设置信息



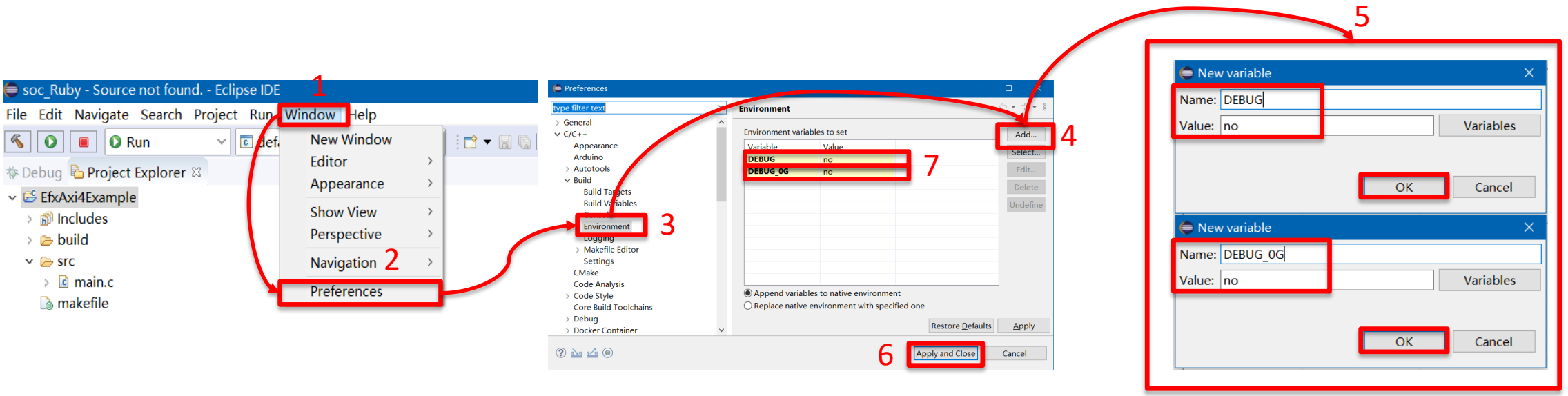

- 点击Finish以后，如果完全正确，当前工程Include下面不会有灰色的路径






易灵思

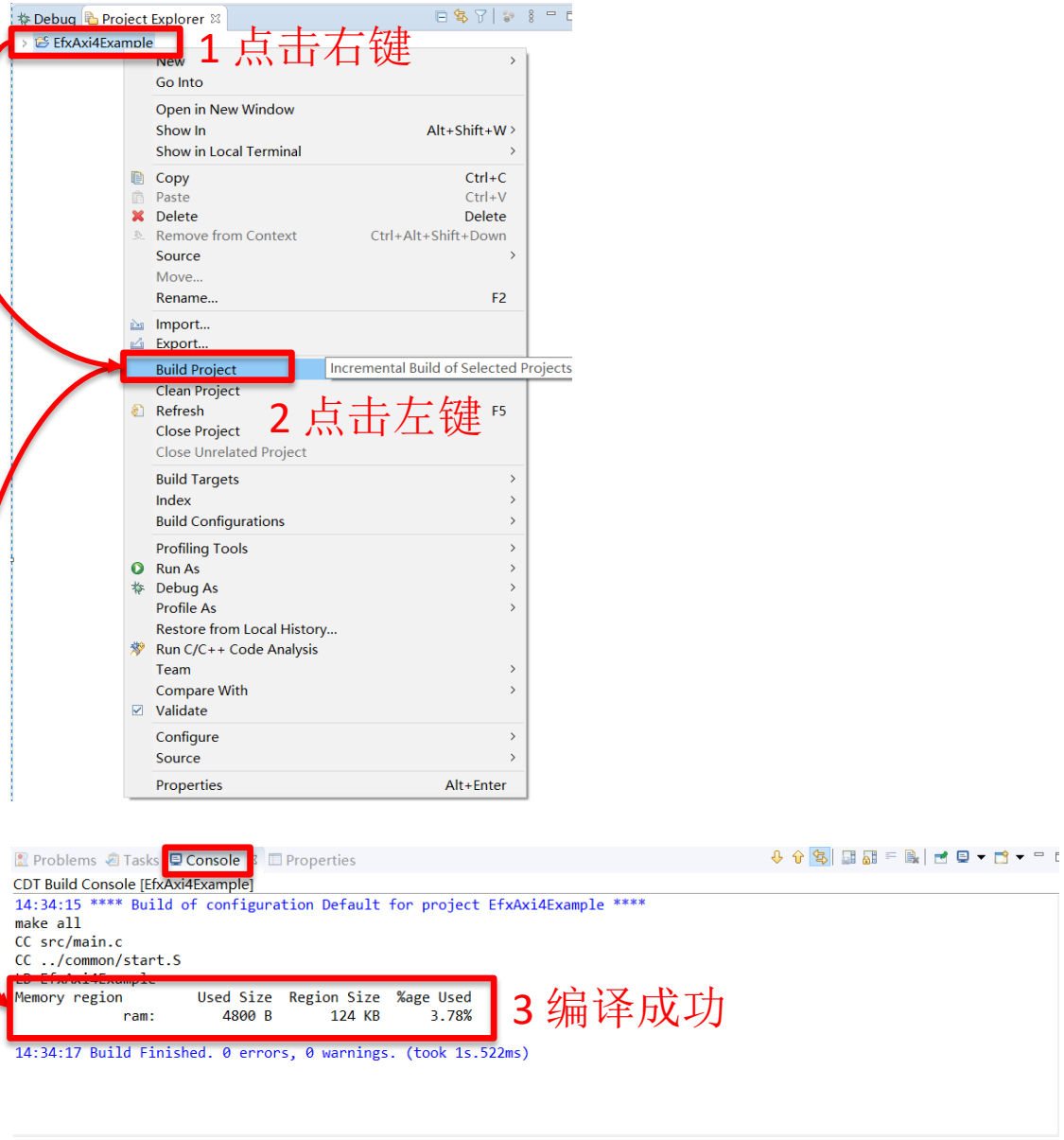
# 设置debug编译开关



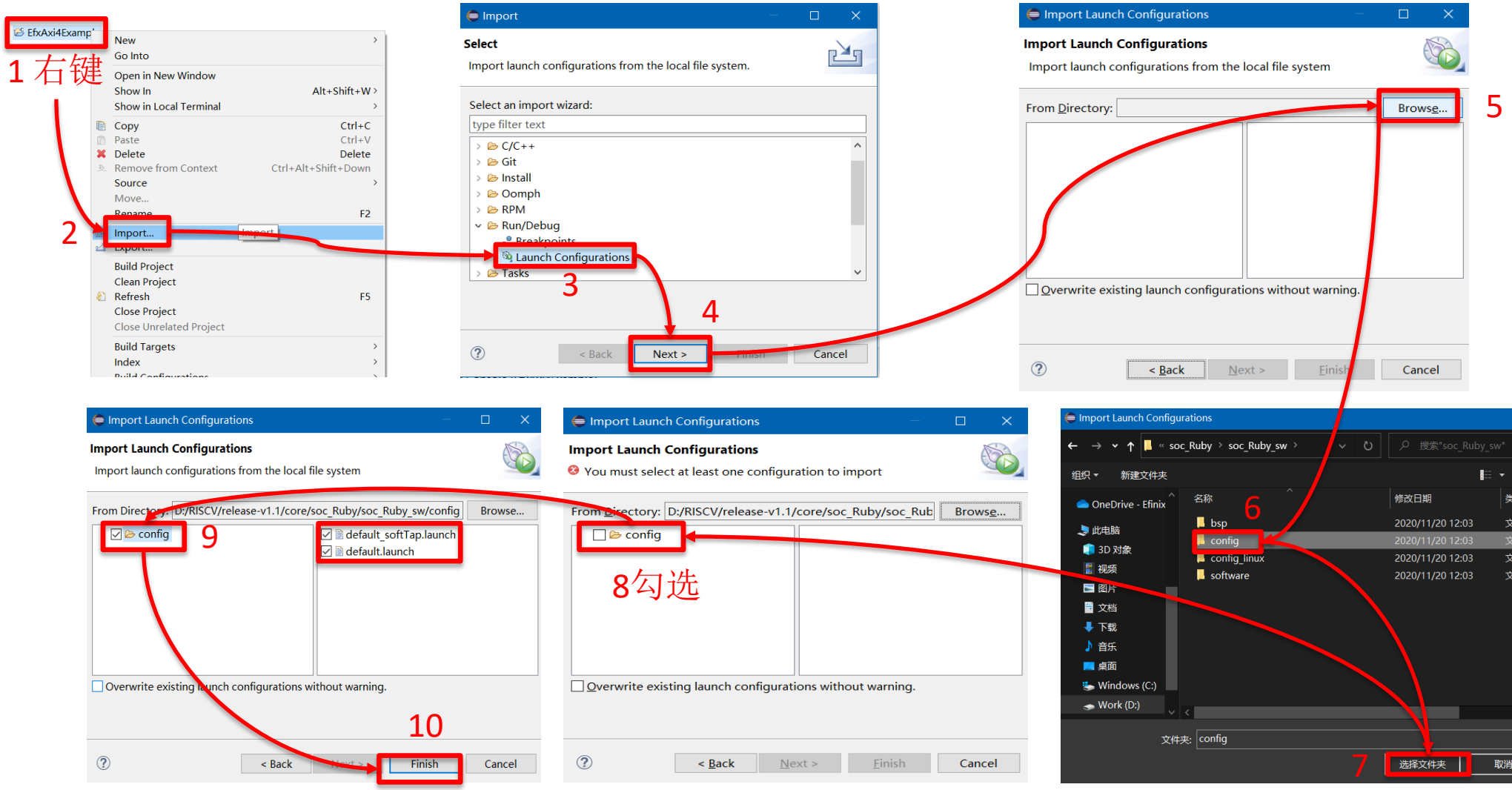

DEBUGOG,O是字母不是数字0

# 第一次编译

- 如果导入的工程编译过，先执行Clean Project一次再编译



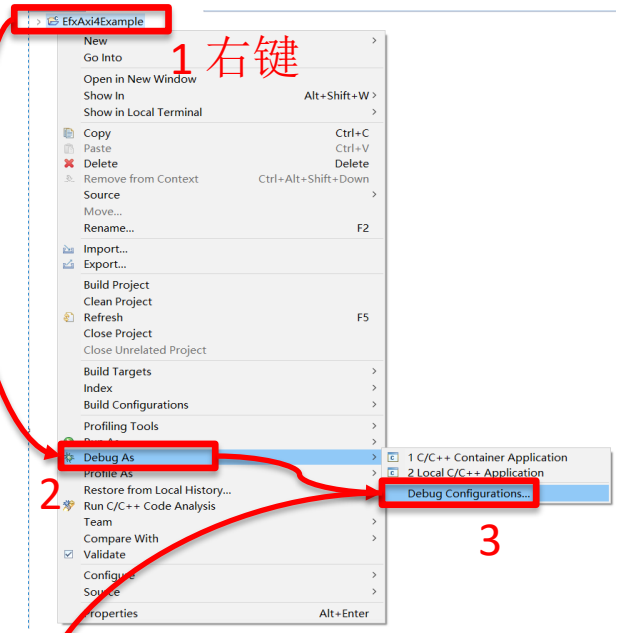
# 导入debug配置信息

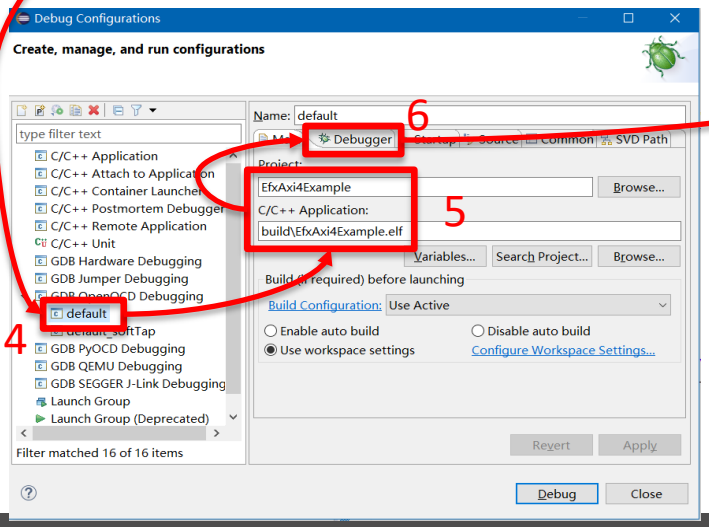


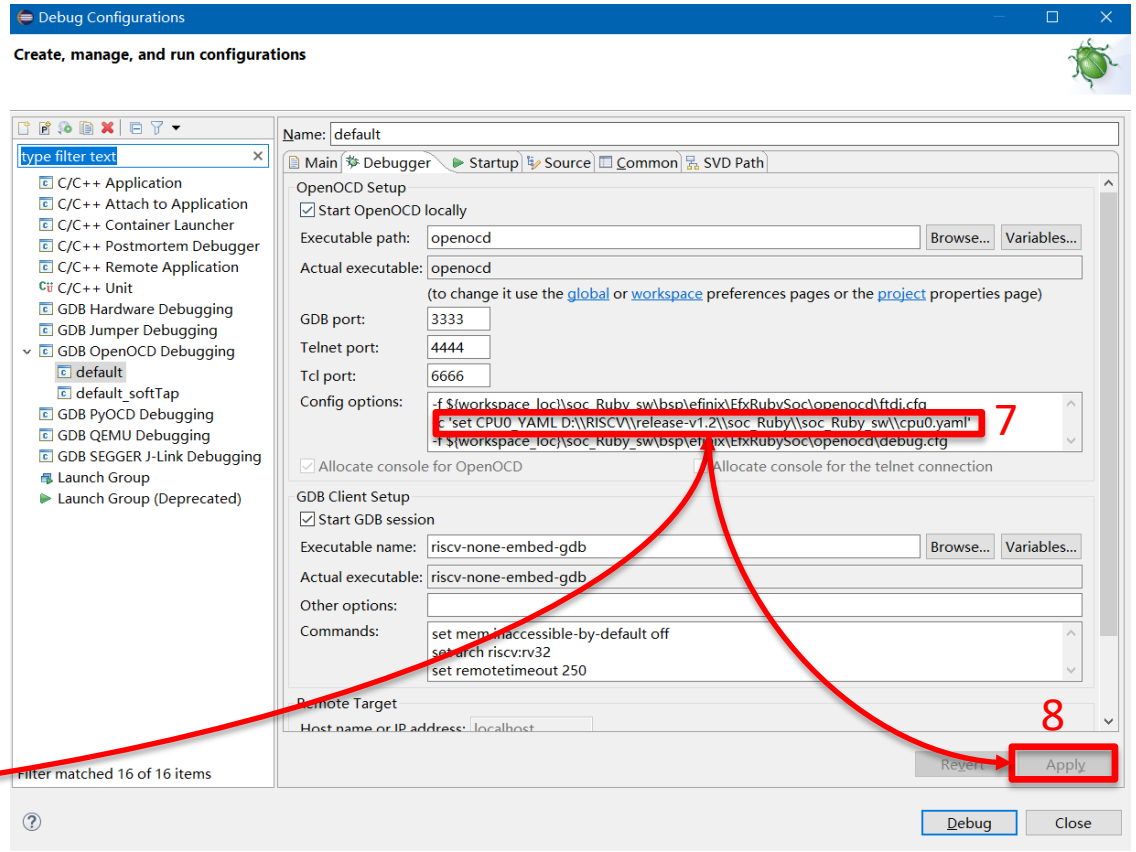



易灵思

# 调整debug配置信息




5 输入调试项目名和带正确路径的elf文件，也可以通过浏览器选择

7 用以"//"隔开的soc_Ruby文件夹完整路径替代原cpu0.yaml路径中的${workspace_loc}

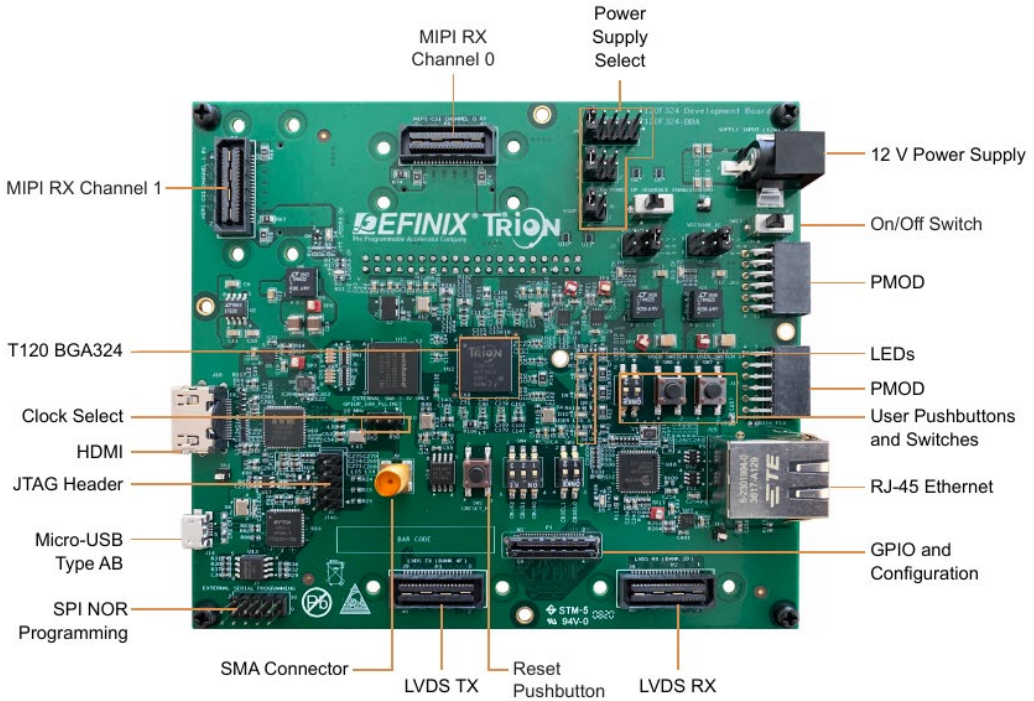
# RISCV RubySOC

## ——Demo

陈弘 Bruce Chen
2020-12-13

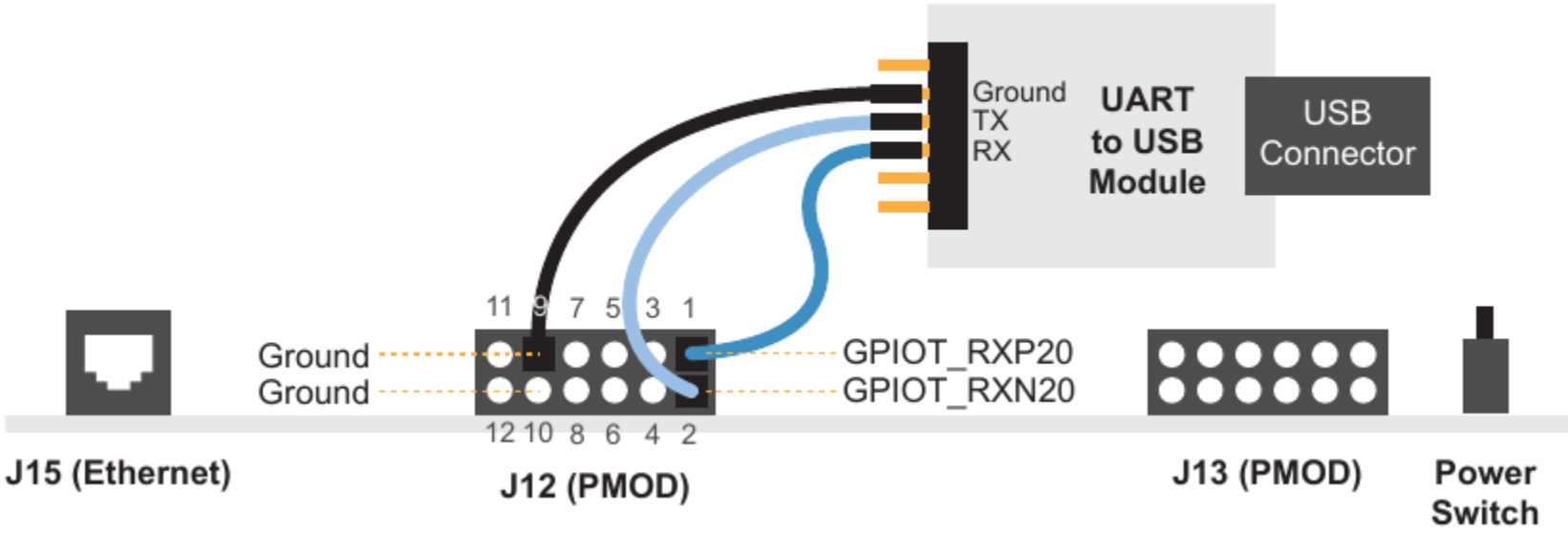
# 准备工作

- T120F324BBA Devkit一套
- USB串口电缆一根



# 串口连接

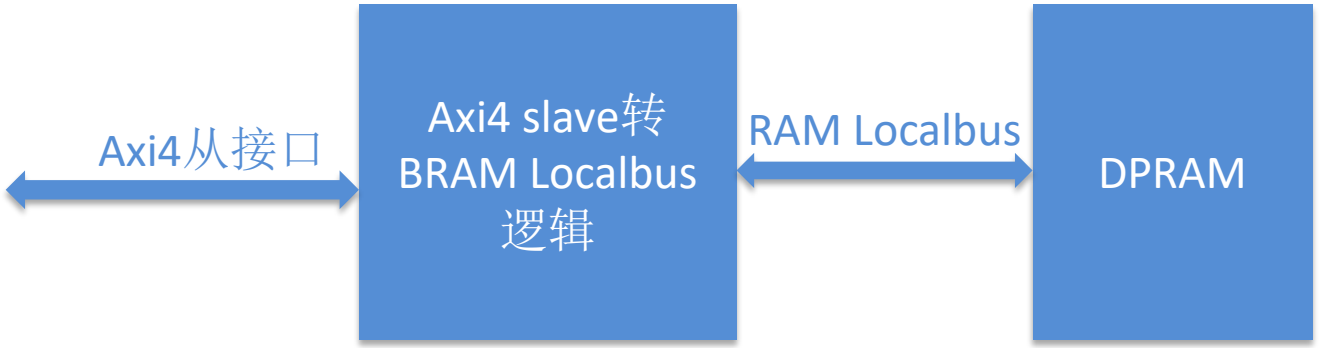
*Figure 5: Connect the UART Module to PMOD Connector J12*



# Demo

- Demo演示
  - EfxAxi4Example
  - EfxApb3Example
- 软件固化
  - OpenOCD
  - Efinity programmer

# EfxAxi4Example



# EfxAxi4Example

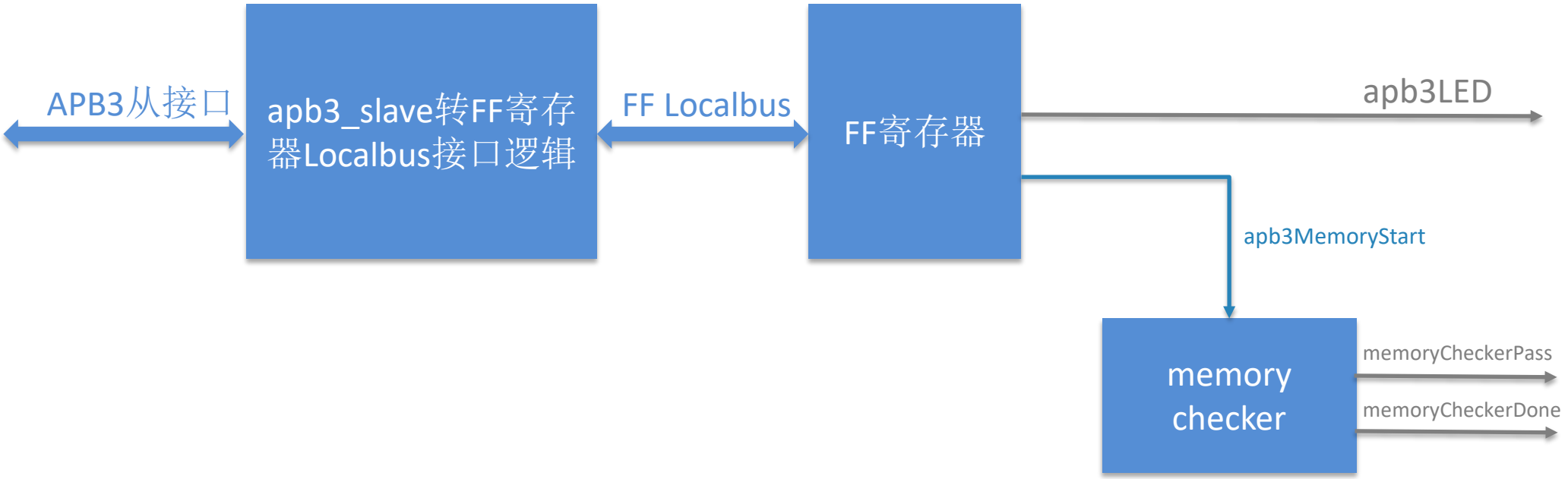
```
void main() {
    u32 data;
    bsp_init();

    uart_writeStr(BSP_UART_TERMINAL, "AXI4 Slave Example\n\r");
    gpio_setOutputEnable(BSP_LED_GPIO, BSP_LED_MASK);
    gpio_setOutput(BSP_LED_GPIO, 0x00000000);

    for (int i=0; i < 40 ; i = i +4 ){
    write_u32(i, SYSTEM_AXI_A_BMB + i);
    data = axi_slave_read32(SYSTEM_AXI_A_BMB + i);
    print_hex(data,8);
    bsp_uDelay(LOOP_UDELAY);
    }

    while(1){
        gpio_setOutput(BSP_LED_GPIO, gpio_getOutput(BSP_LED_GPIO) ^ BSP_LED_MASK);
        bsp_uDelay(LOOP_UDELAY);
    }
}
```

# EfxApb3Example



# EfxApb3Example

```
void main() {
    struct example_apb3_ctrl_reg cfg={0};
    bsp_init();

    gpio_setOutputEnable(BSP_LED_GPIO, BSP_LED_MASK);
    gpio_setOutput(BSP_LED_GPIO, 0x00000000);

    uart_writeStr(BSP_UART_TERMINAL, "Enabling Memory Checker\n\r");
    cfg.exampleLED = 1;
    cfg.exampleMemoryStart = 1;
    example_register_write(&cfg);

    while(1){
        gpio_setOutput(BSP_LED_GPIO, gpio_getOutput(BSP_LED_GPIO) ^ BSP_LED_MASK);
        bsp_uDelay(LOOP_UDELAY);
    }
}
```

# OpenOCD固化软件

## STEP 1

1. 打开CMD界面1
2. CD 进入SDK的安装目录
3. 运行setup.bat命令
4. CD 进入软件工程soc_Ruby_sw 目录
5. 运行 openocd.exe -f bsp\efinix\EfxRubySoc\openocd\ftdi.cfg -c “set CPU0_YAML cpu0.yaml” –f bsp\efinix\EfxRubySoc\openocd\flash.cfg

## STEP 2

1. 打开CMD界面2
2. 运行 telnet localhost 4444
3. CD进入需要烧写的BIN文件所在的目录
4. 运行 flash write_image erase unlock <filename>.bin 0x380000命令

## STEP 3

1. 在CMD界面2：运行exit
2. 在CMD界面1：CTRL+C

# Efinity programmer固化软件

STEP 1

1. 运行bin2hex.py
2. 按提示输入bin文件完整路径和名称
3. 按提示输入hex文件需要存放的路径和名称
4. 完成软件bin文件到FPGA bitstream格式转换

STEP 2

1. 运行Efinity Programmer
2. 在Start Flash Address输入软件APP的起始位置
3. 选择软件bin转换后的hex文件，作为bitstream
4. 开始烧写
5. 烧写完成以后Programmer会提示校验错误，这是因为转换以后的hex文件并不是真正的FPGA bitstream，可以忽略掉这个错误提示

# 谢 谢 ！