GNU MCU Eclipse OpenOCD

前言

概述

本文主要介绍 GNU MCU Eclipse OpenOCD调试方面的功能。

产品版本

芯片名称	内核版本
RK3588	
RK3568	
RK3566	
RK3399	
RK3288	
RK3368	
RK3326	
PX30	
RK3308	
RV1108	
RV1126/RV1109/RV1106	
RK2108	
RK2206	
RISCV	

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

目录

1. 说明

调试架构：Eclipse CDT+GNU MCU Eclipse OpenOCD+Eclipse+GDB+OpenOCD+ftdi/jlink+SOC

- Eclipse CDT (C/C++ Development Tooling) C/C++ 开发工具
- GNU MCU Eclipse OpenOCD 是一个开源插件，主要完成CDT和GDB、OpenOCD的交互
- Eclipse 是一个很强大的工具，可以集成各种插件，ARM DS-5也是基于它
- GDB GNU调试器
- OpenOCD 是一个开源的调试软件，可以适配各种SWD/JTAG适配器，支持ARM，RISCV等架构
- ftdi 采用ft232h，USB转SWD/JTAG芯片，可以作为SWD/JTAG适配器，速度快，稳定性高

2. 操作系统环境

2.1 Windows

2.1.1 eclipse软件包openocd_eclipse.zip

解压，进入主目录：

- "Eclipse_for_OpenOCD V1.0.exe" 执行该文件，可以打开eclipse软件
- RK目录：
    - eclipse-workspace 工作目录，eclipse已默认把工作目录设到该文件夹
    - example 相关的例子
    - tools 开源相关的工具，如GDB，JDK
    - 实战视频 快速上手
    - OpenOCD openocd相关文件
    - SVD（CMSIS System View Description format）主要用来查看芯片寄存器
    - doc 使用帮助文档

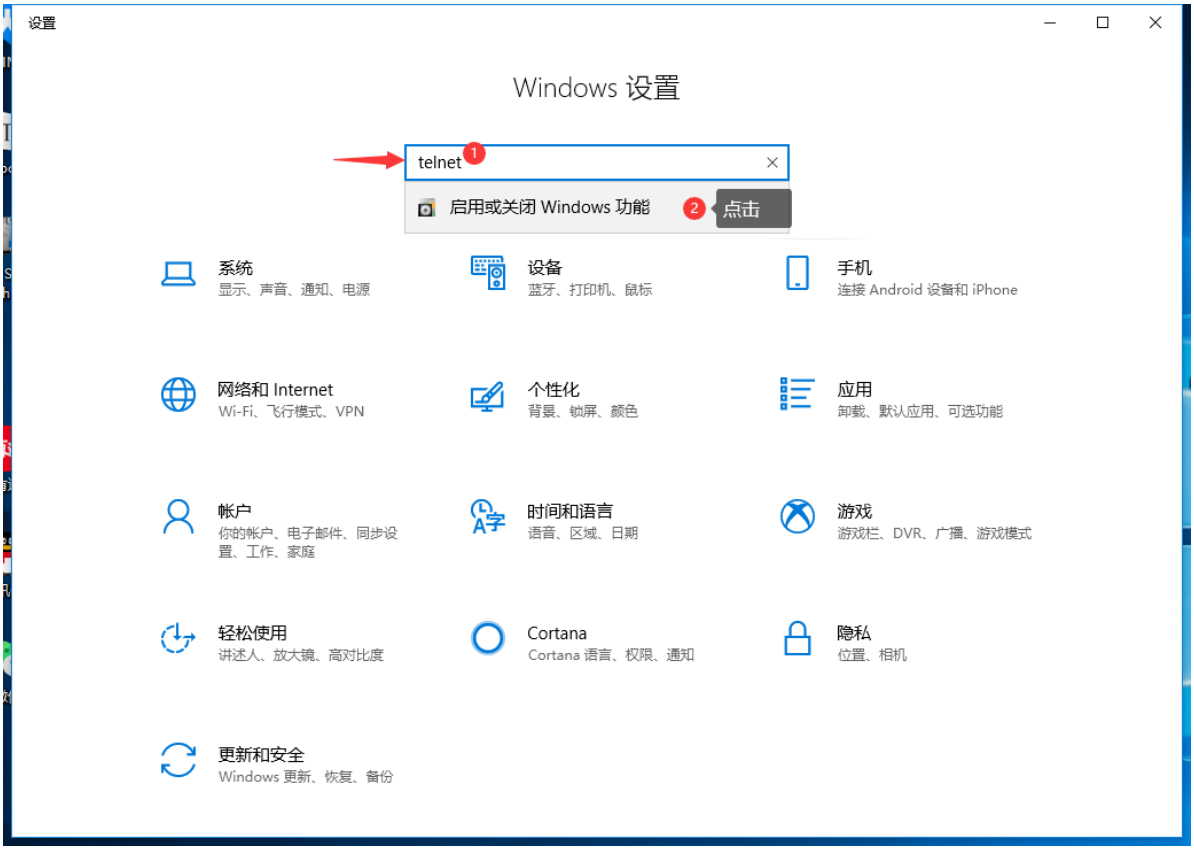
2.1.2 运行eclipse，需要安装JRE

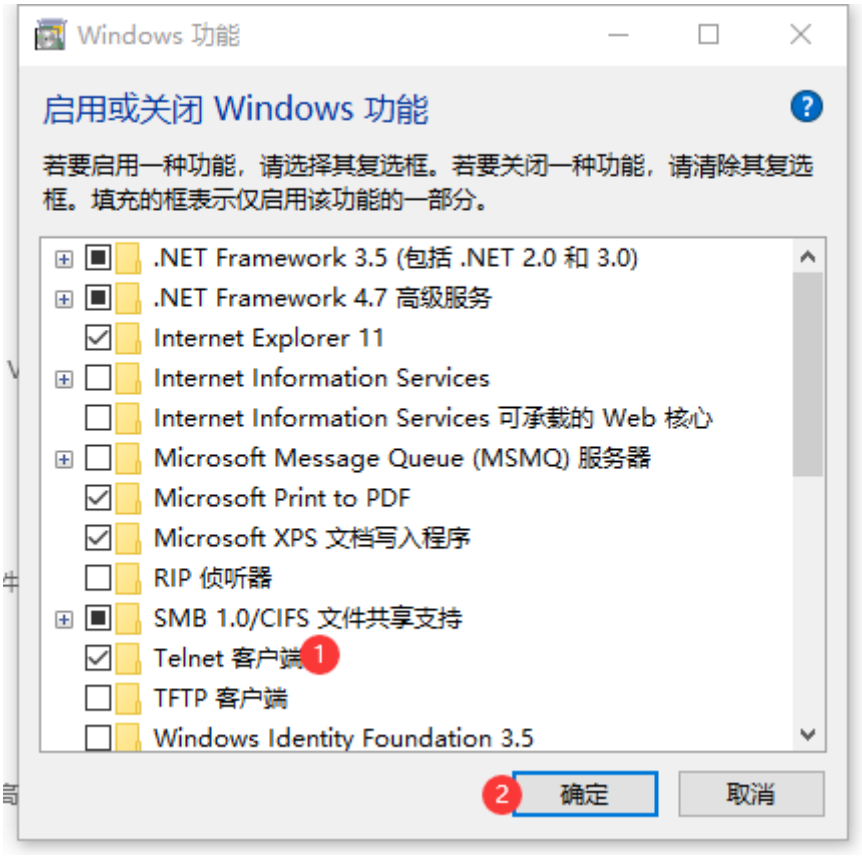
RK\tools\jdk_8.0.1310.11_64.exe

2.1.3 使能Windows telnet功能

telnet用来进入openocd的命令行模式。

不同的Windows版本，使能telnet的入口可能不一样，请自行百度。以下主要介绍windows10:





2.2 Ubuntu 64位

2.2.1 eclipse软件包openocd_eclipse.tar.gz

解压openocd_eclipse.tar.gz：

```
tar -xzvf openocd_eclipse.tar.gz
```

进入主目录：

- eclipse 执行该文件，打开eclipse软件
- RK目录：
  - eclipse-workspace 工作目录，eclipse已默认把工作目录设到该文件夹
  - example 相关的例子
  - OpenOCD openocd相关文件
  - SVD （CMSIS System View Description format）主要用来查看芯片寄存器
  - tools 开源相关的工具，如GDB
  - doc 使用帮助文档
  - 实战视频 快速上手

2.2.2 安装软件

- 运行eclipse，需要安装JRE。

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jre  这里不一定要8
```

- 运行openocd需要libusb。

```
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libftdi-dev
```

- 拷贝USB设备信息

```
sudo cp RK/OpenOCD/drivers/99-openocd.rules /etc/udev/rules.d/
sudo cp RK/OpenOCD/drivers/60-openocd.rules /etc/udev/rules.d/
```

- 安装开源的反汇编工具capstone。

```
sudo apt-get install libcapstone-dev
```

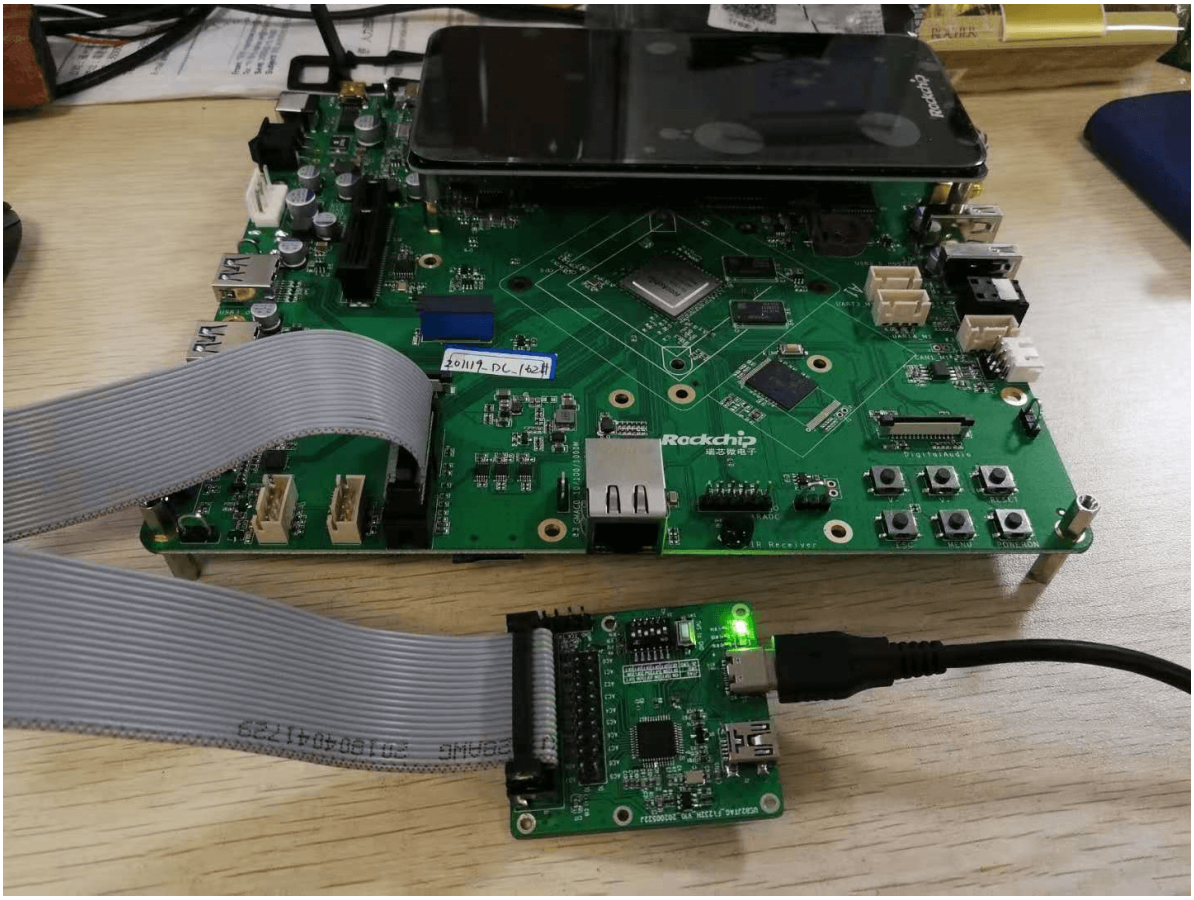
说明：在Ubuntu 16.04和Ubuntu 18.04测试正常

3. 快速上手

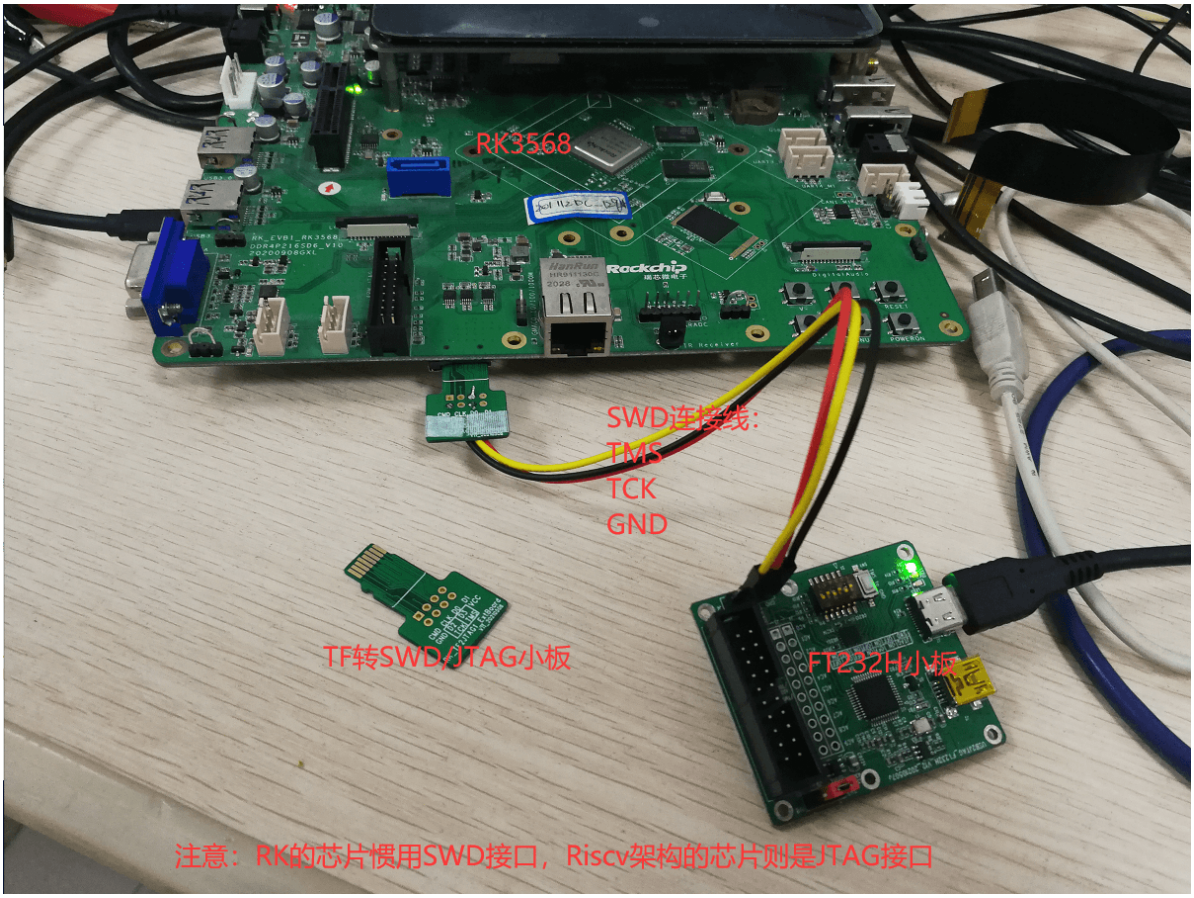
不管是Ubuntu还是Windows，UI界面基本一致。由于大部分人习惯Windows，本文介绍以Windows为主。

3.1 硬件连接

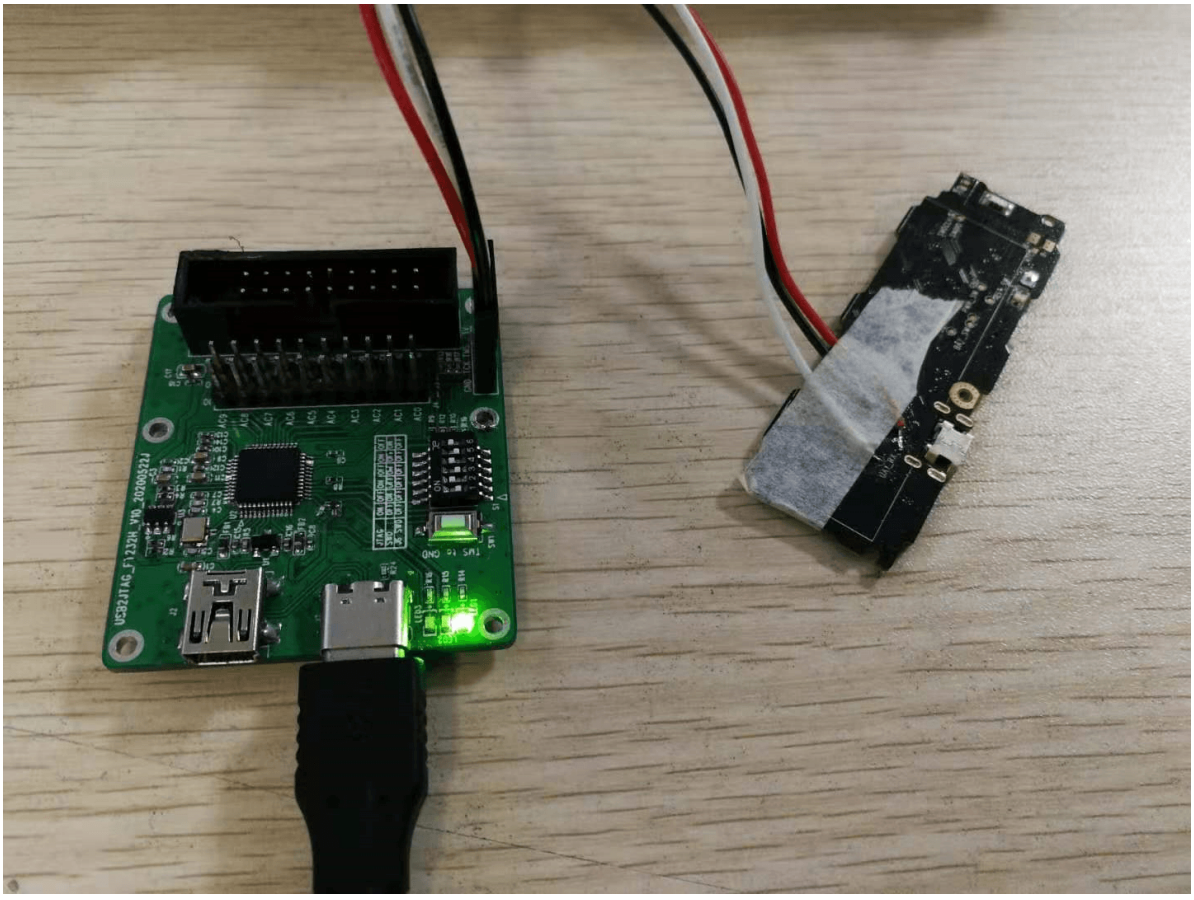
3.1.1 ARM 20PIN标准JTAG座子



3.1.2 TF转JTAG小板

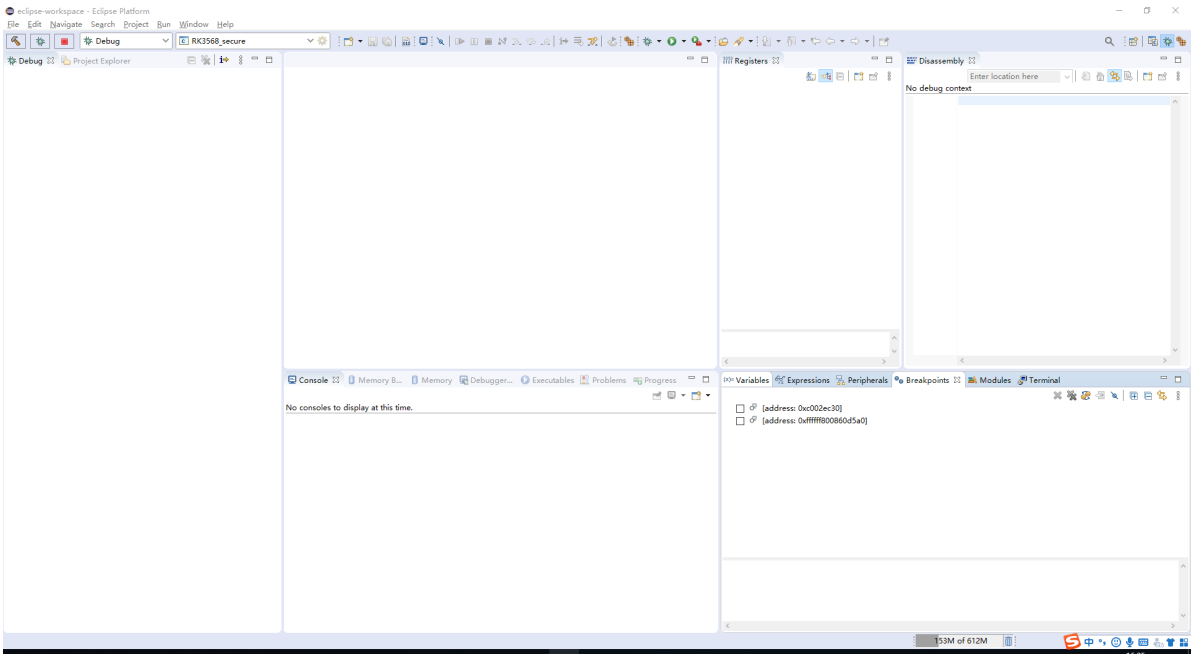


3.1.3 UART2与JTAG复用



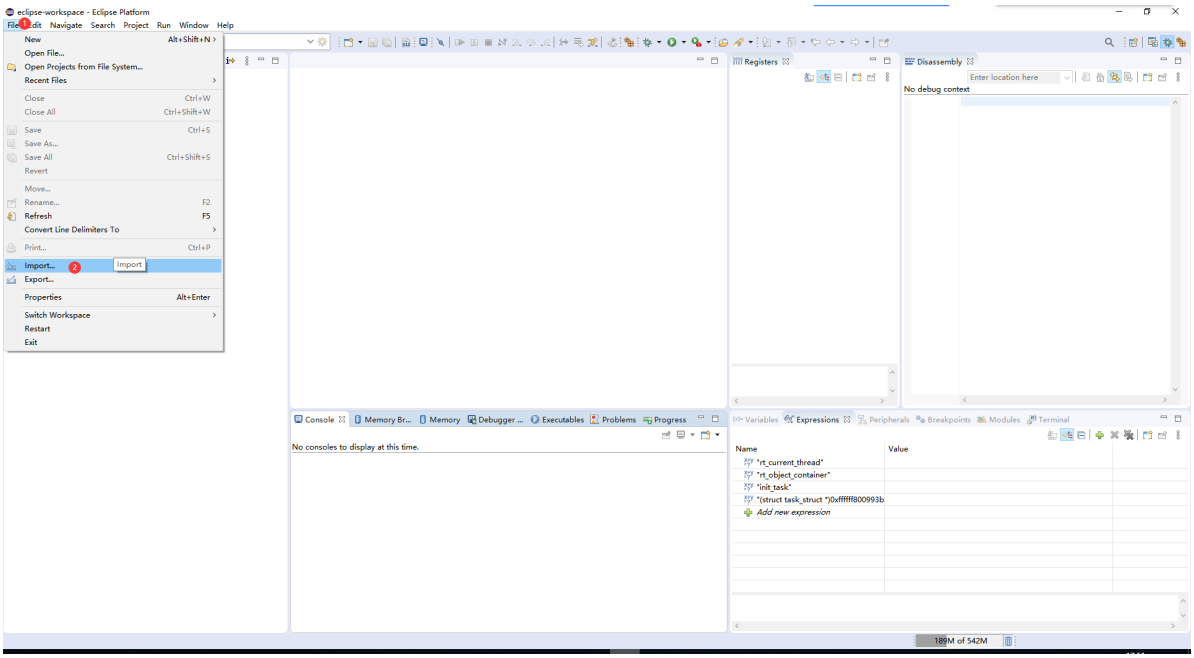
3.2 软件连接

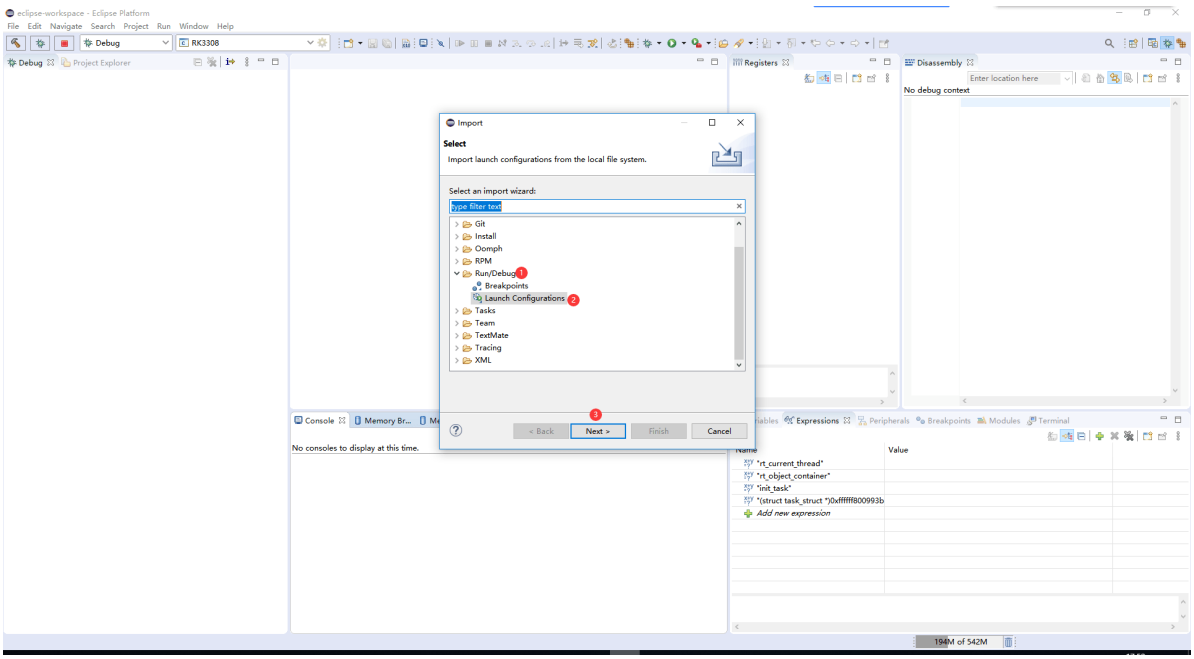
3.2.1 运行软件Eclipse_for_OpenOCD V1.0.exe

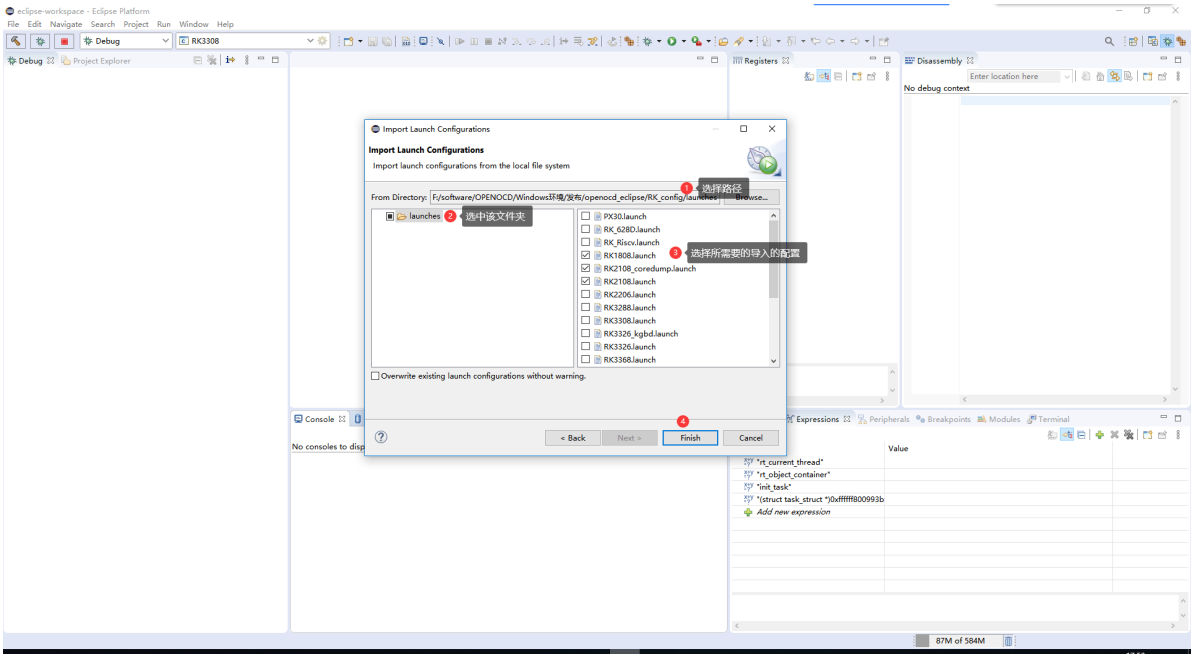


3.2.2 导入芯片配置（非必需）

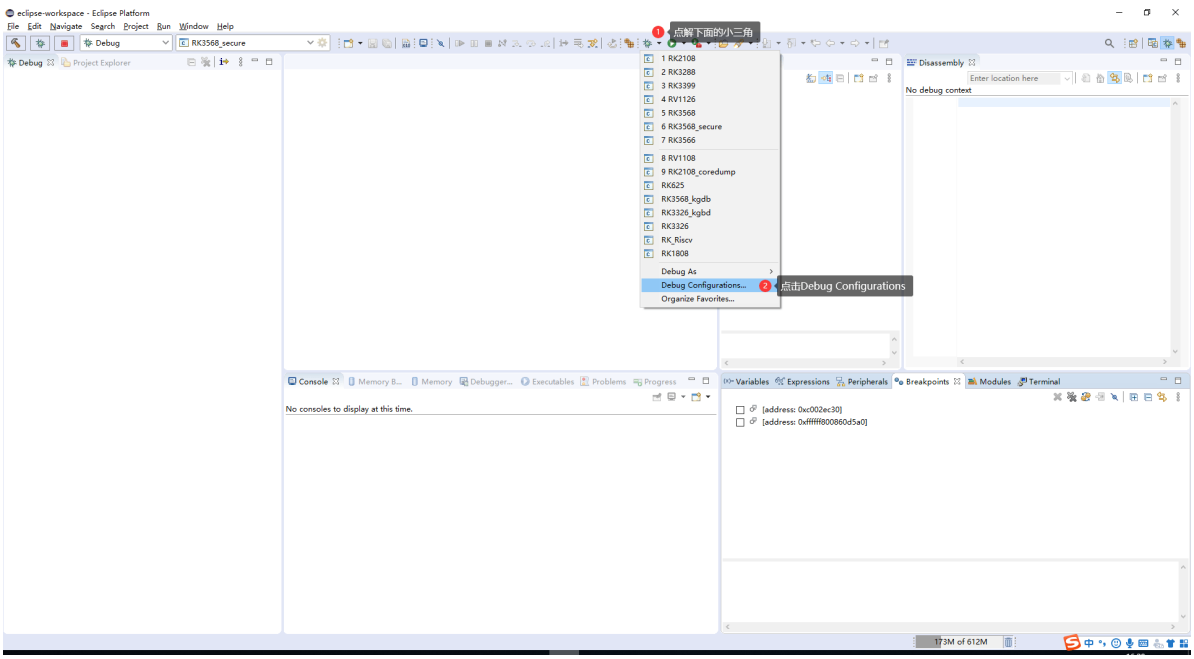
正常拿到的软件包，已经包含了芯片配置，但是芯片配置会持续更新，修改或增加芯片配置，这就需要导入芯片配置。

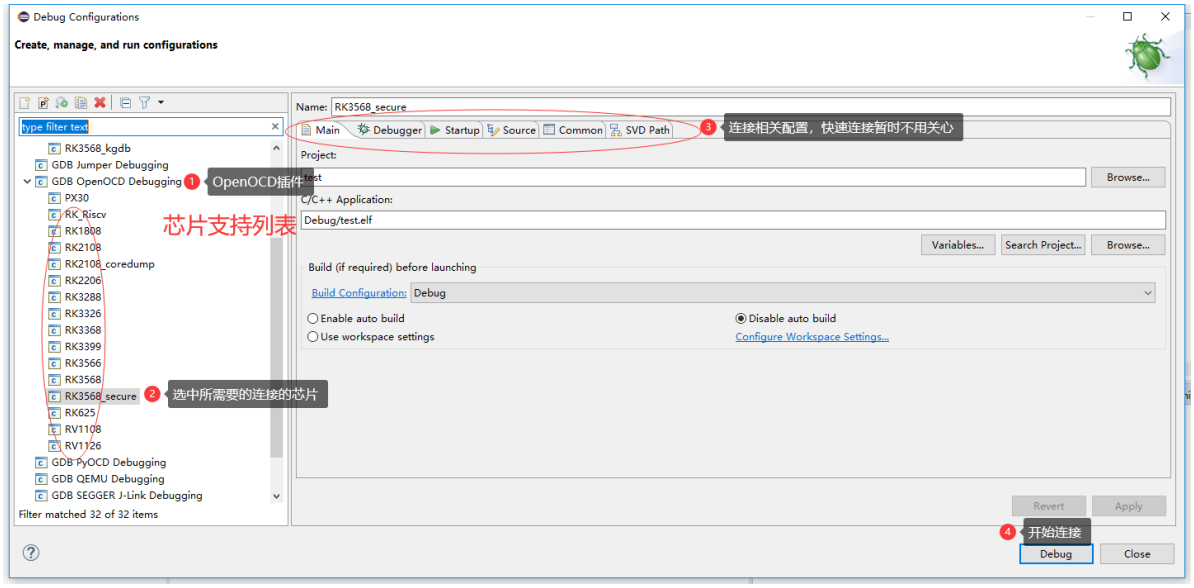






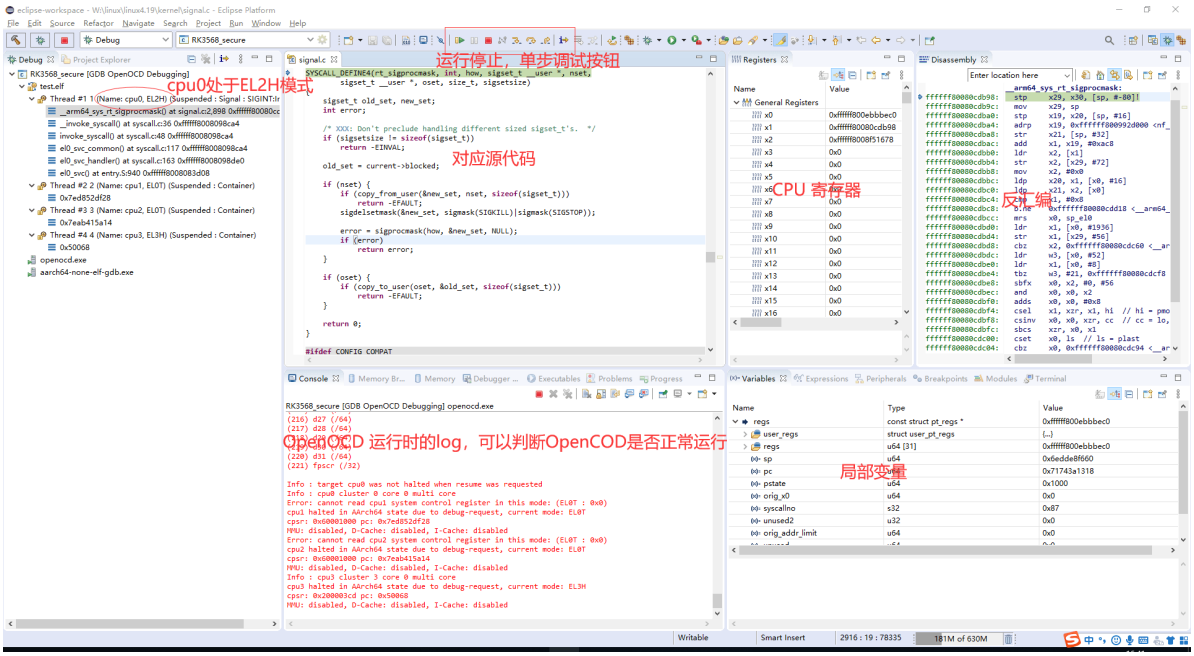
3.2.3 从芯片支持列表里找到所需要连接的芯片





3.2.4 连接成功

cpu stop后的效果如下：

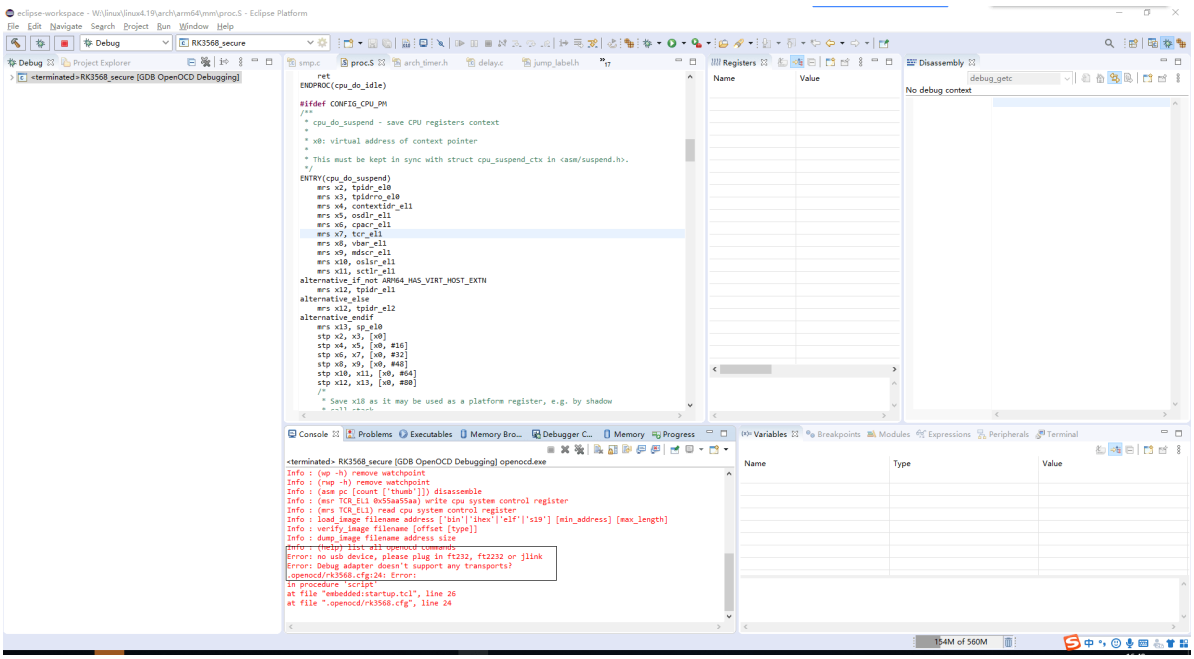


3.2.5 连接失败

3.2.5.1 JTAG适配器无法识别

需要确认适配器的驱动，或者适配器是否能正常工作。参照文档《Rockchip_Developer_Guide_FT232H_USB2JTAG.pdf》

说明：openocd默认可以自动识别ft232h ft2232h jlink三种适配器。



3.2.5.2 芯片无法识别

需要确认芯片引脚接触是否正常，或者芯片的JTAG IOMUX是否使能，或者降低JTAG的TCK速率。

- the connection is ok, but the cpu is in incorrect state

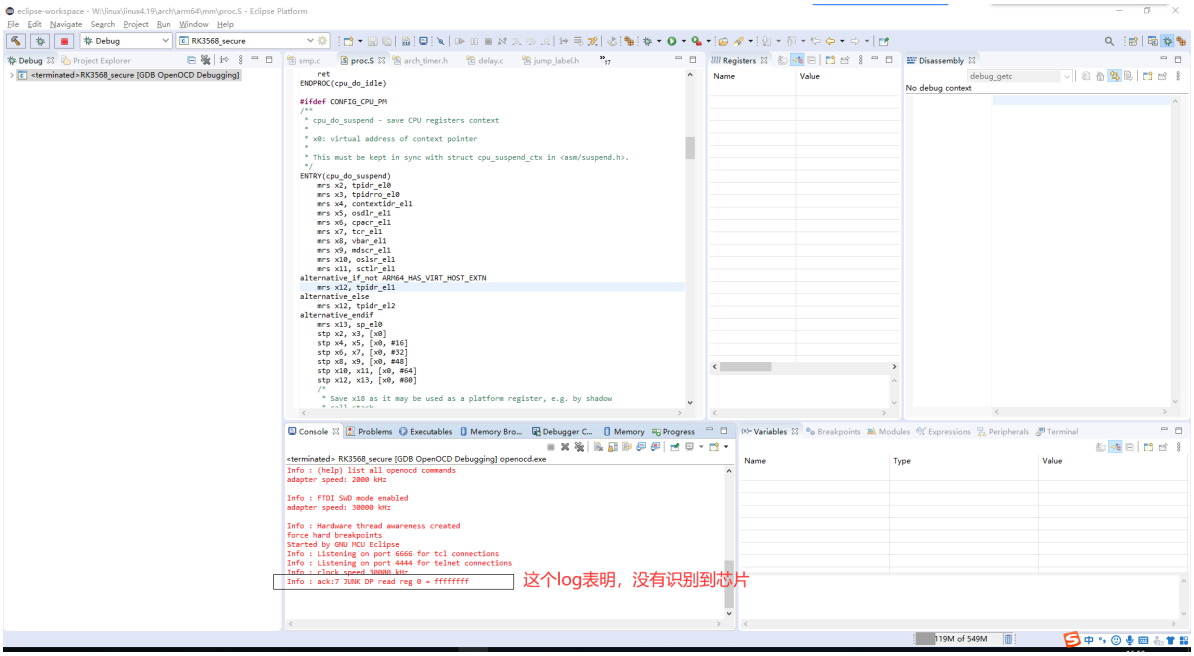
  这种情况表示硬件连接正常，JTAG也认到了，但是芯片状态异常，这种情况是无法继续debug的

- please check hardware, make sure swd/jtag iomux is correct, and the pins connection is ok
- please check hardware, the swdio(tms) pin is low level

上述2种情况是，硬件连接问题，可能是IOMUX没配好，或者pin脚接触不良

- Please，adapter speed 1000 set clk 1MHz to try again

遇到这个log，需要降低JTAG速率，再次尝试连接

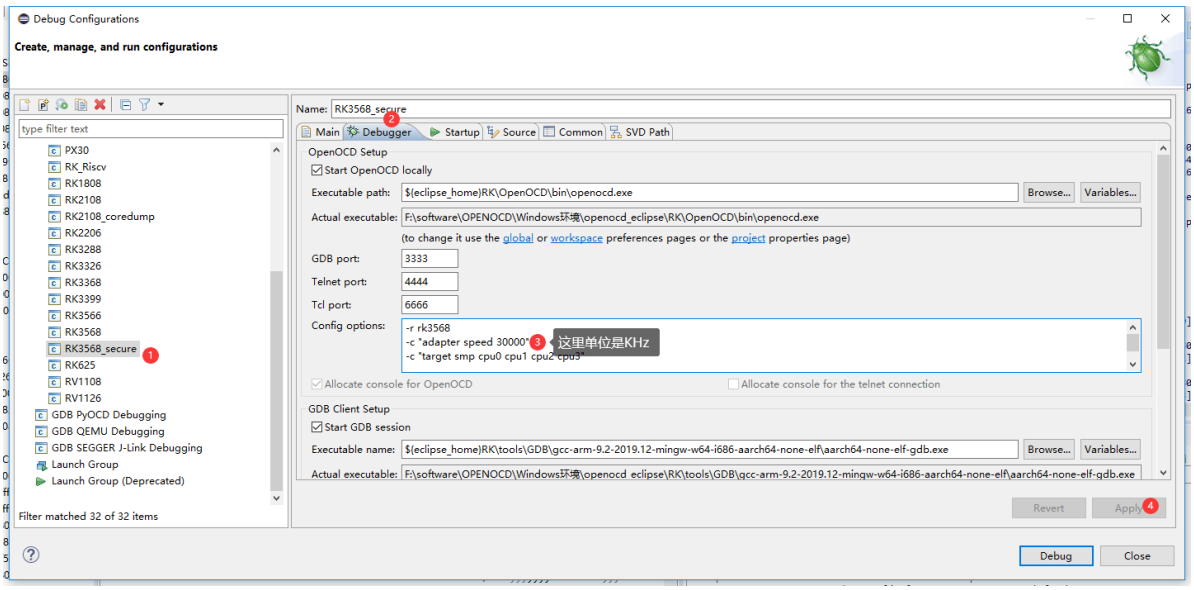


4. 基础调试功能

4.1 调整SWD/JTAG速率

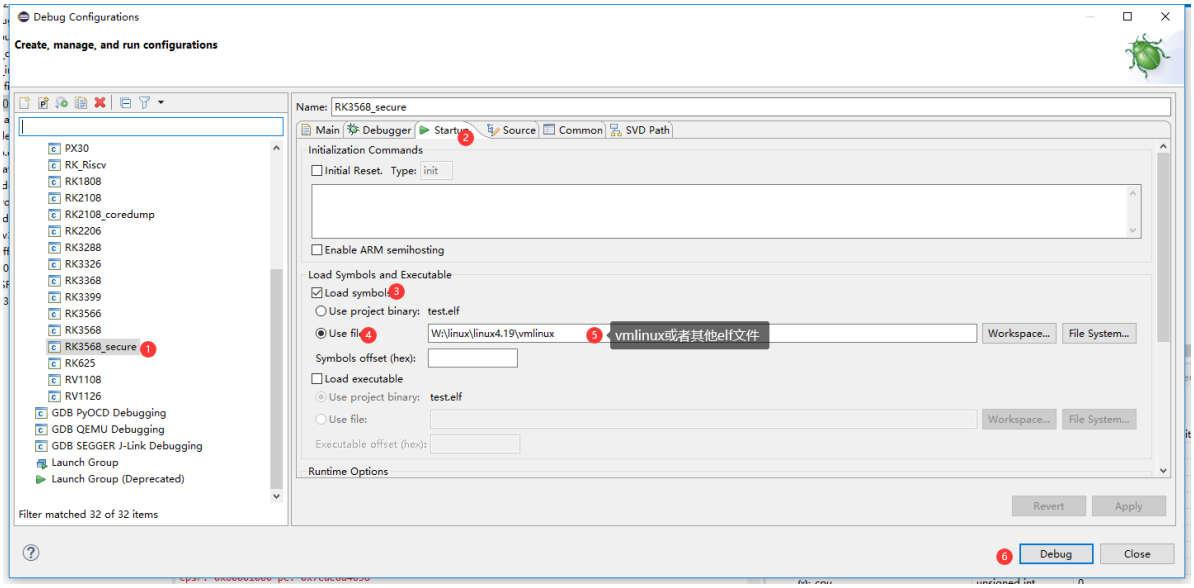
如果硬件限制导致SWD/JTAG通讯失败，需要降低TCK的速率，比如1000KHz。

最大到30000KHz，推荐使用15000KHz或者7500KHz。



4.2 静态加载符号表

静态加载符号表需要在连接前完成，而且只能加载一个符号表。



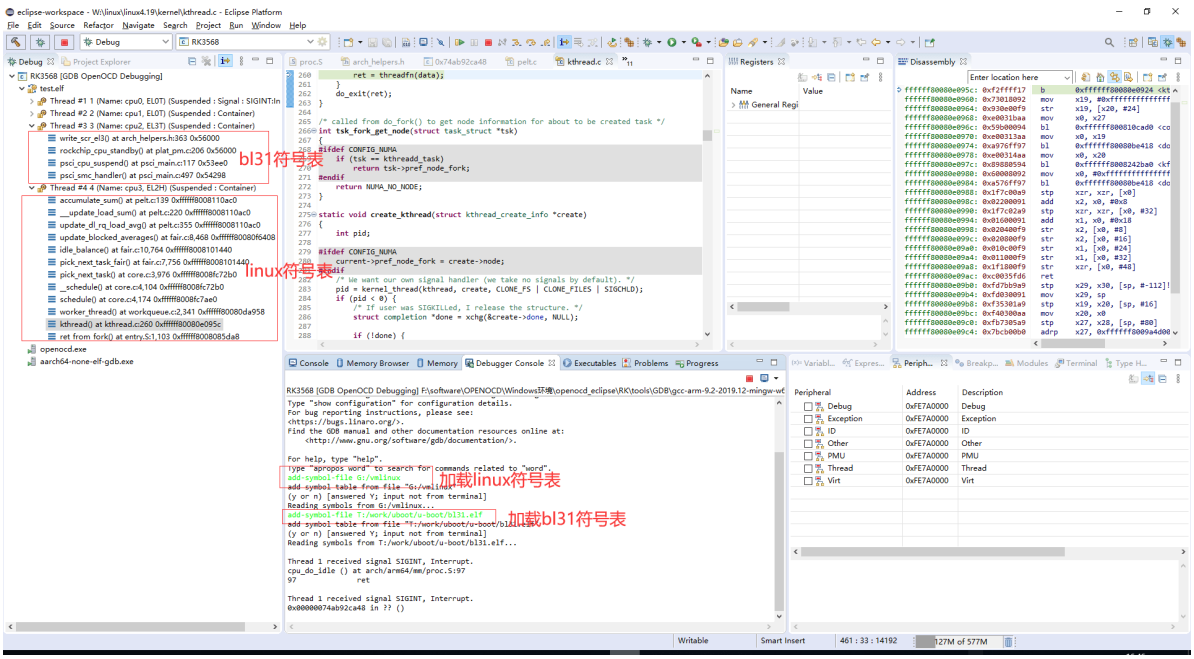
4.3 动态加载符号表

在Debugger Console窗口，这实际是执行GDB命令的窗口，执行add-symbol-file命令加载符号表，可同时加载多个符号表。

```
加载linux符号表：add-symbol-file G:/vmlinux
加载bl31符号表：add-symbol-file T:/work/uboot/u-boot/bl31.elf
加载tpl符号表：add-symbol-file T:/work/uboot/u-boot/tpl/u-boot-tpl
加载spl符号表：add-symbol-file T:/work/uboot/u-boot/spl/u-boot-spl
加载uboot符号表：add-symbol-file T:/work/uboot/u-boot/u-boot
```

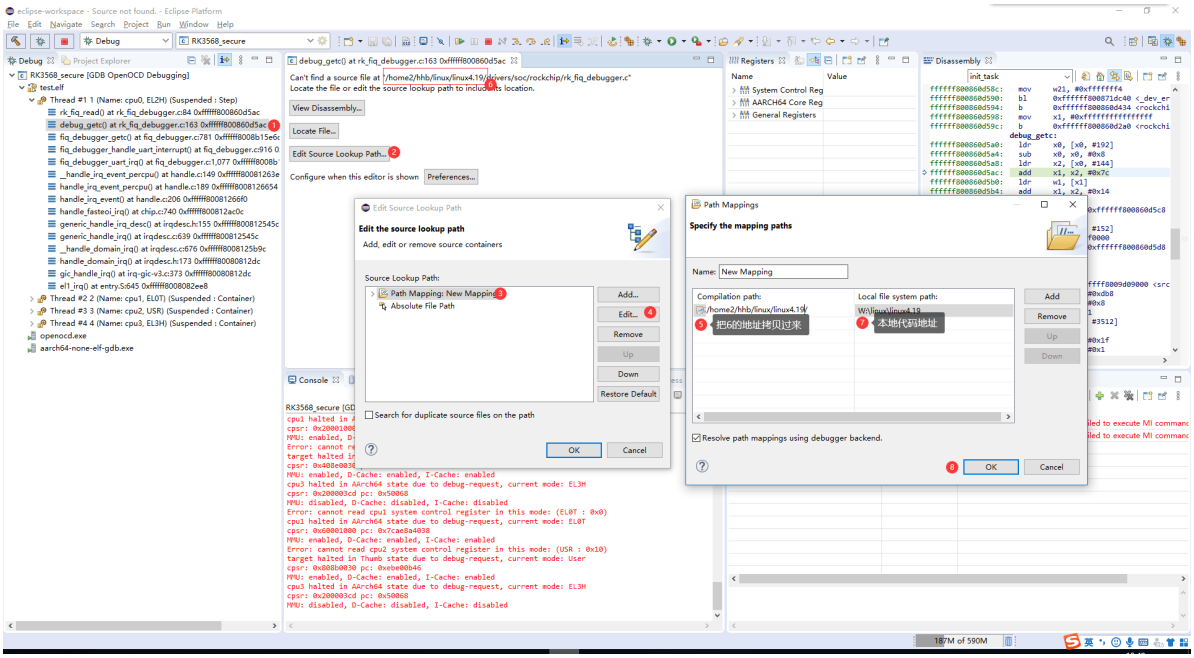
注意：在windows环境，需要将"\"改为"/"。

另外，执行完加载命令后，需要单步跑下，函数调用栈才会显示出来。

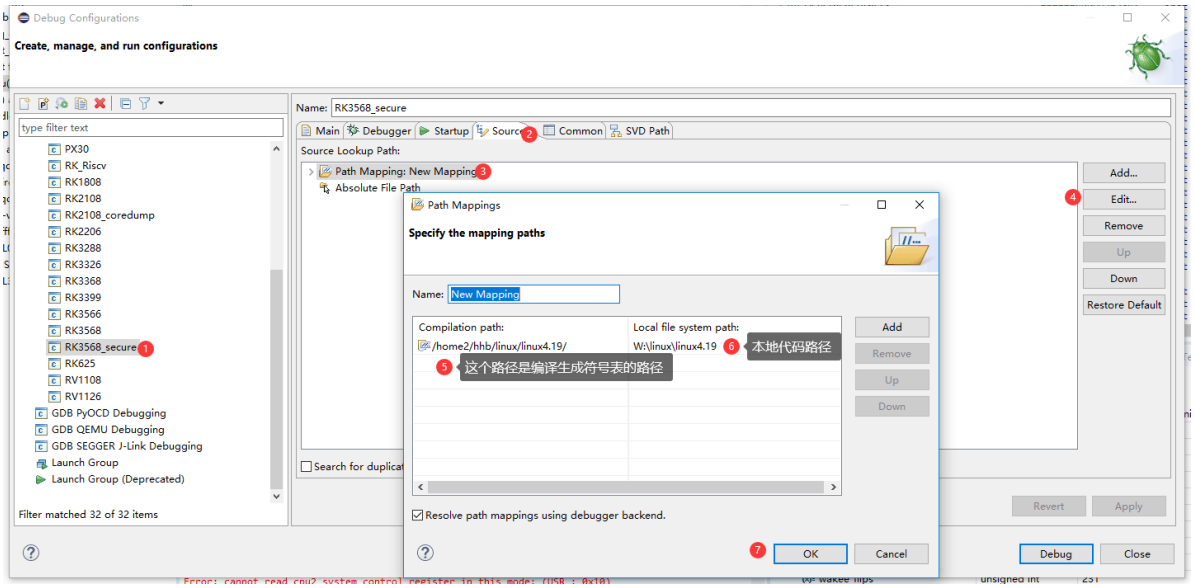


4.4 设置源代码路径

- 方法1，如下图

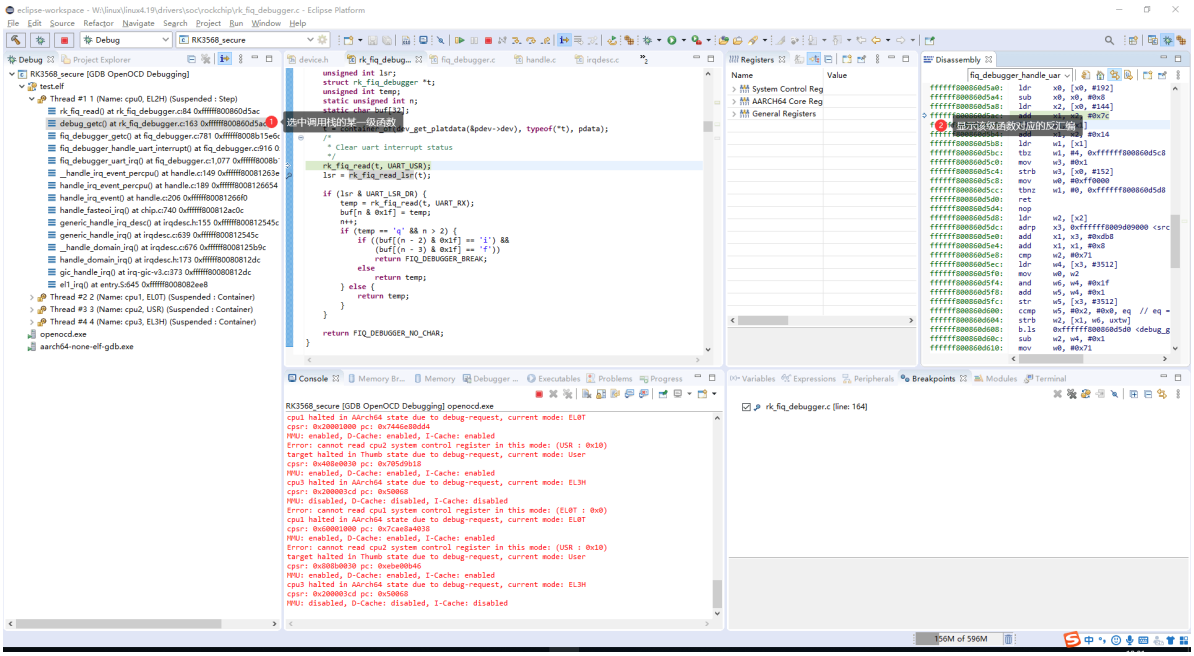


- 方法2，进入Debug Configurations界面设置

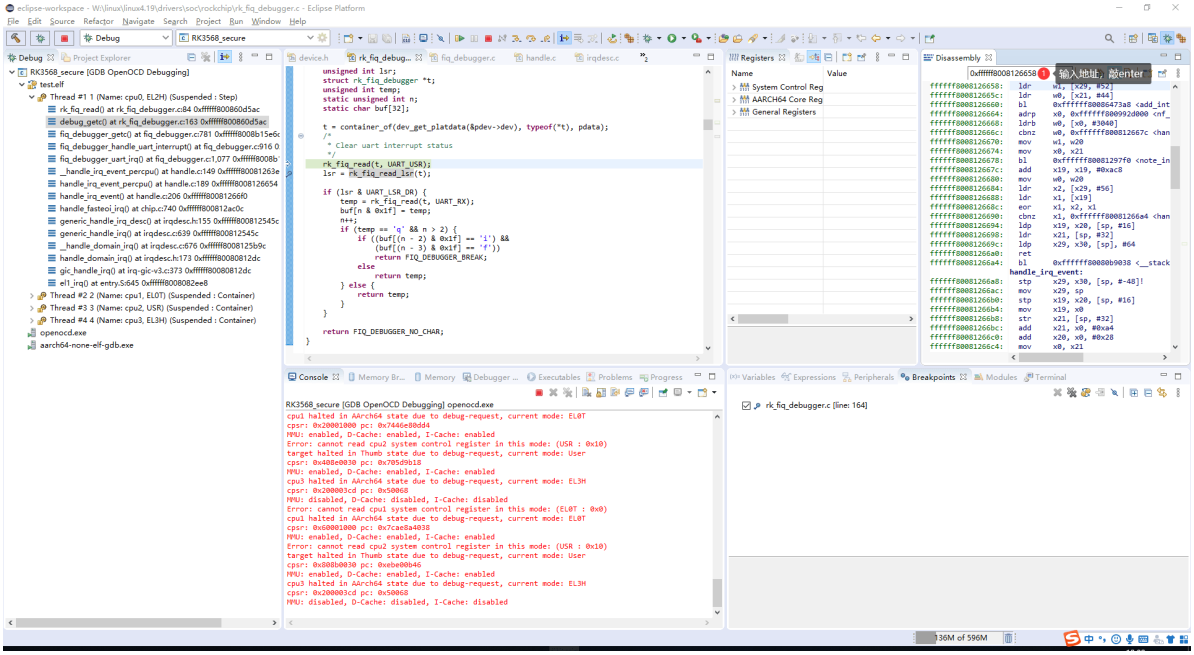


4.5 查看反汇编

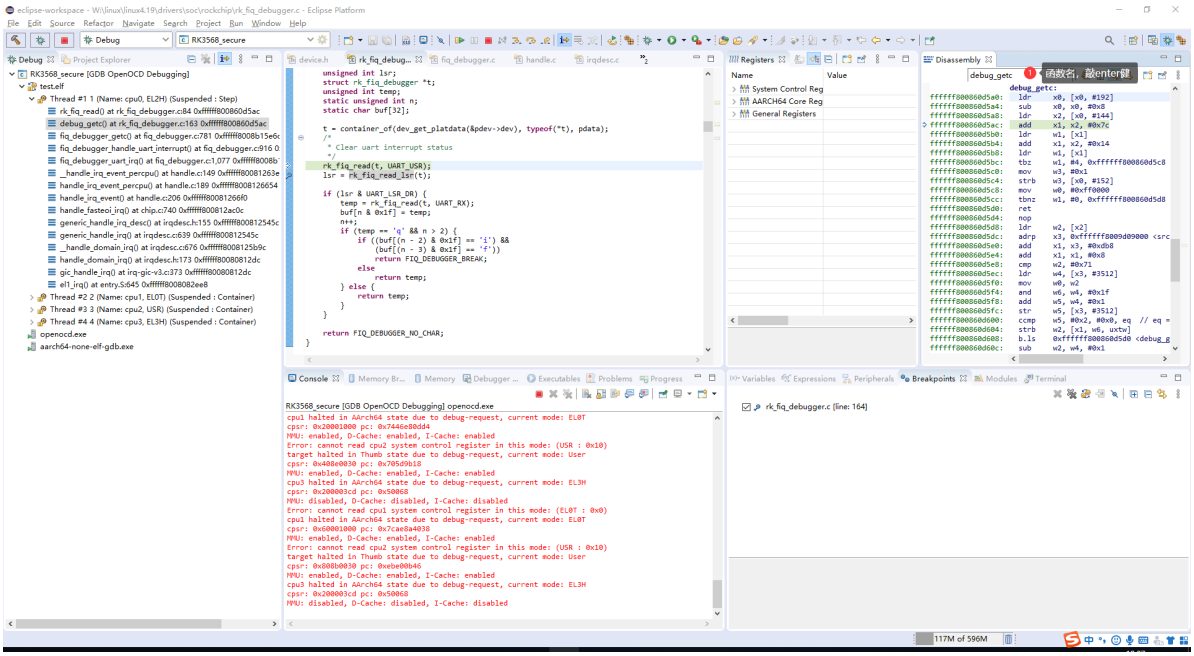
- 查看调用栈函数的反汇编



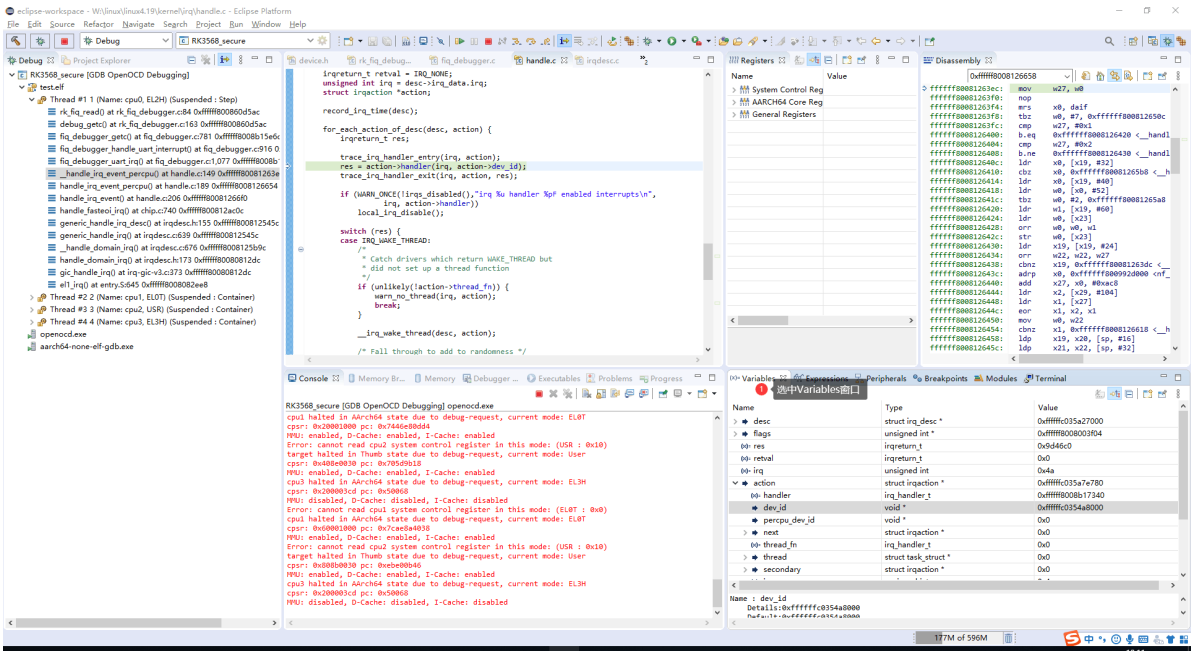
- 查看某个地址的反汇编



查看某个函数的反汇编，前提是有导入符号表

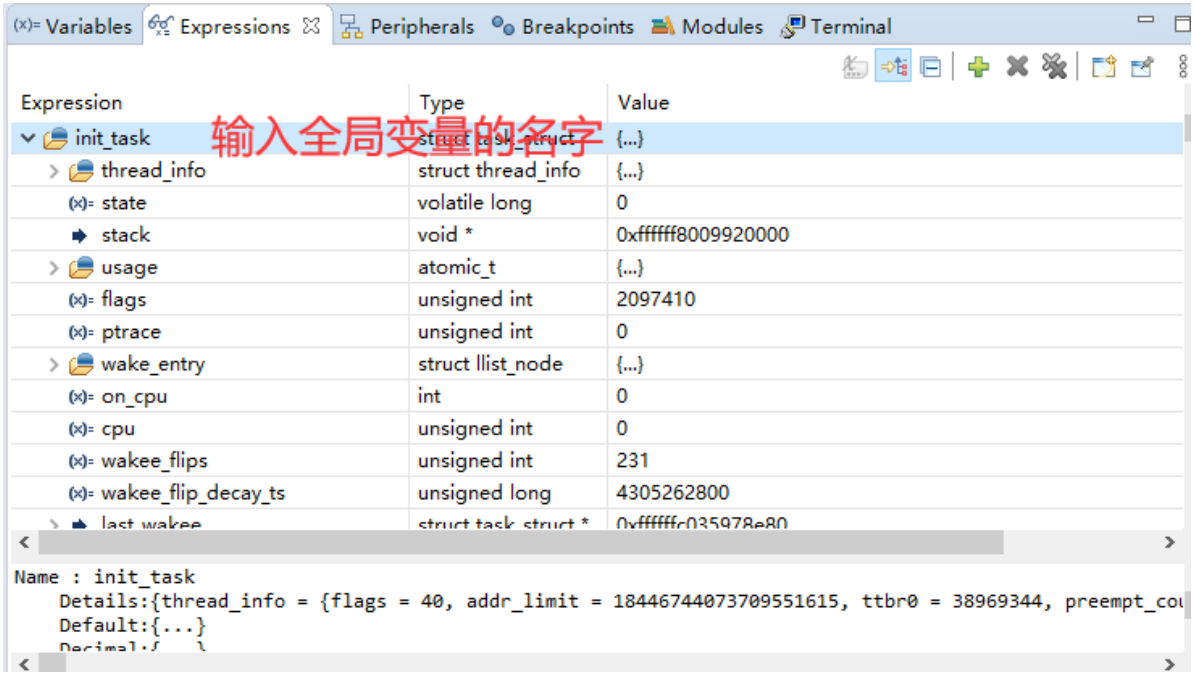


4.6 查看调用栈函数的局部变量

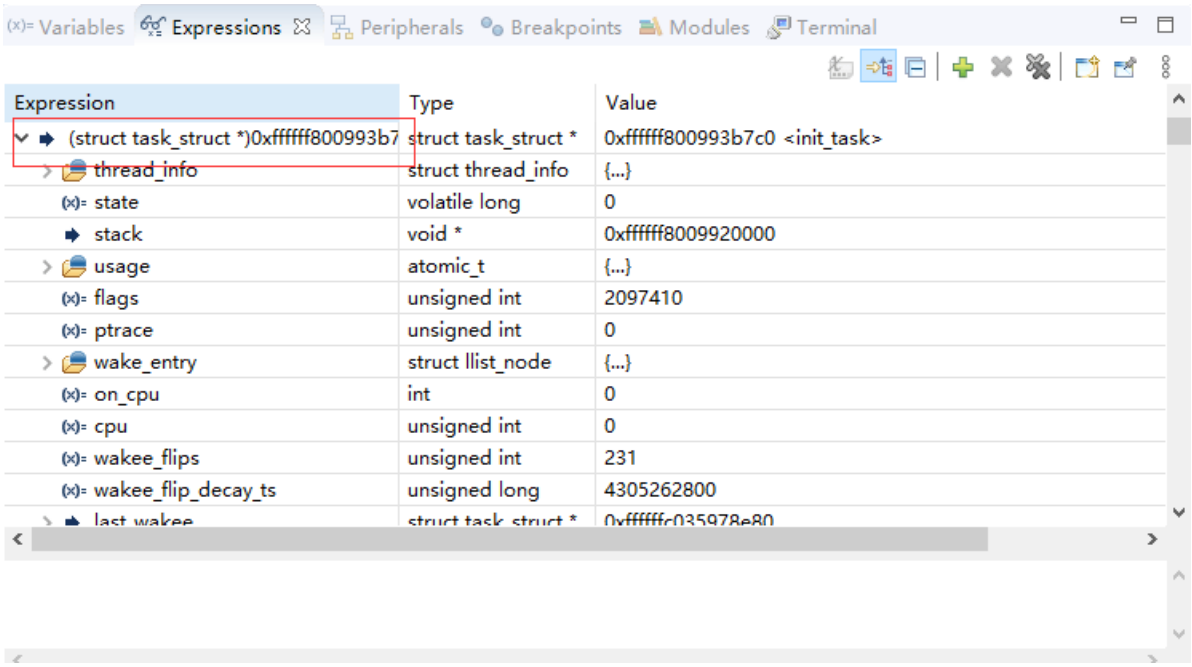


4.7 查看全局变量

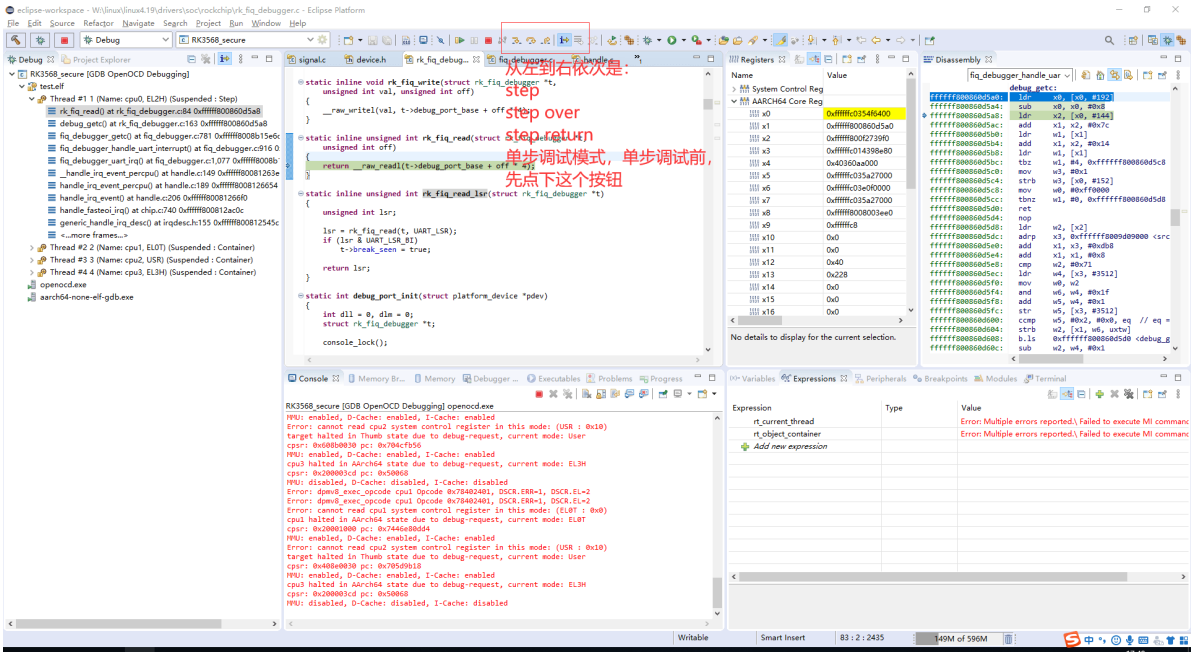
点击Expressions窗口的Add new expression，输入全局变量名字



点击Expressions窗口的Add new expression，输入指针形式的expression

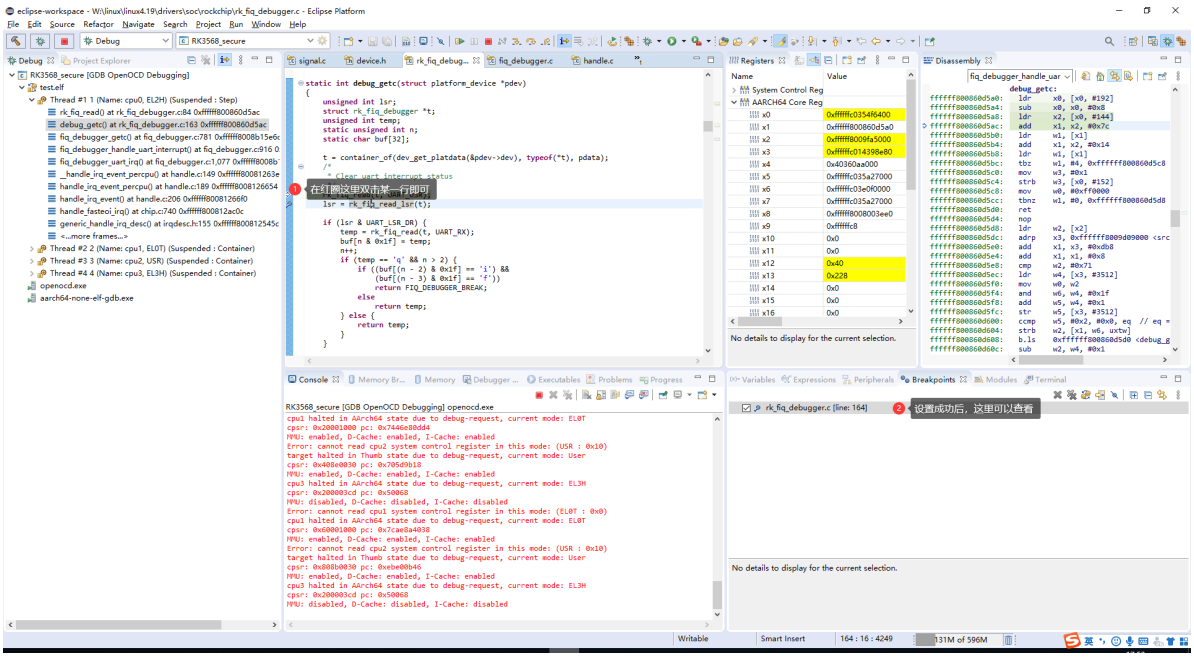


4.8 单步调试

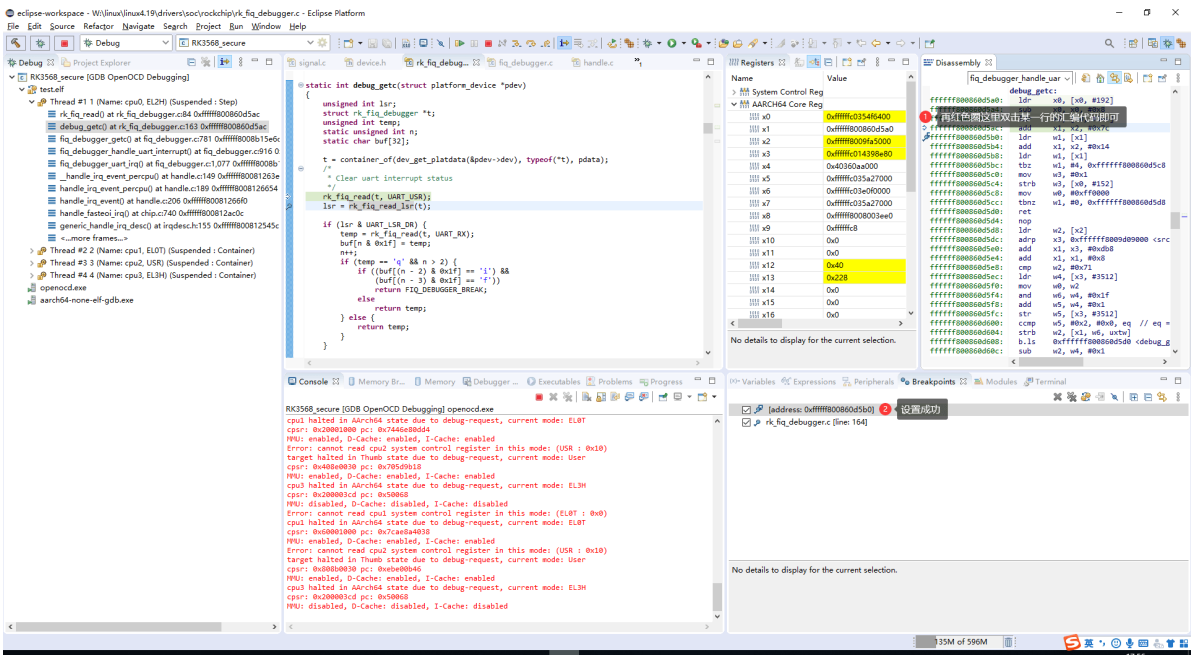


4.9 设置断点

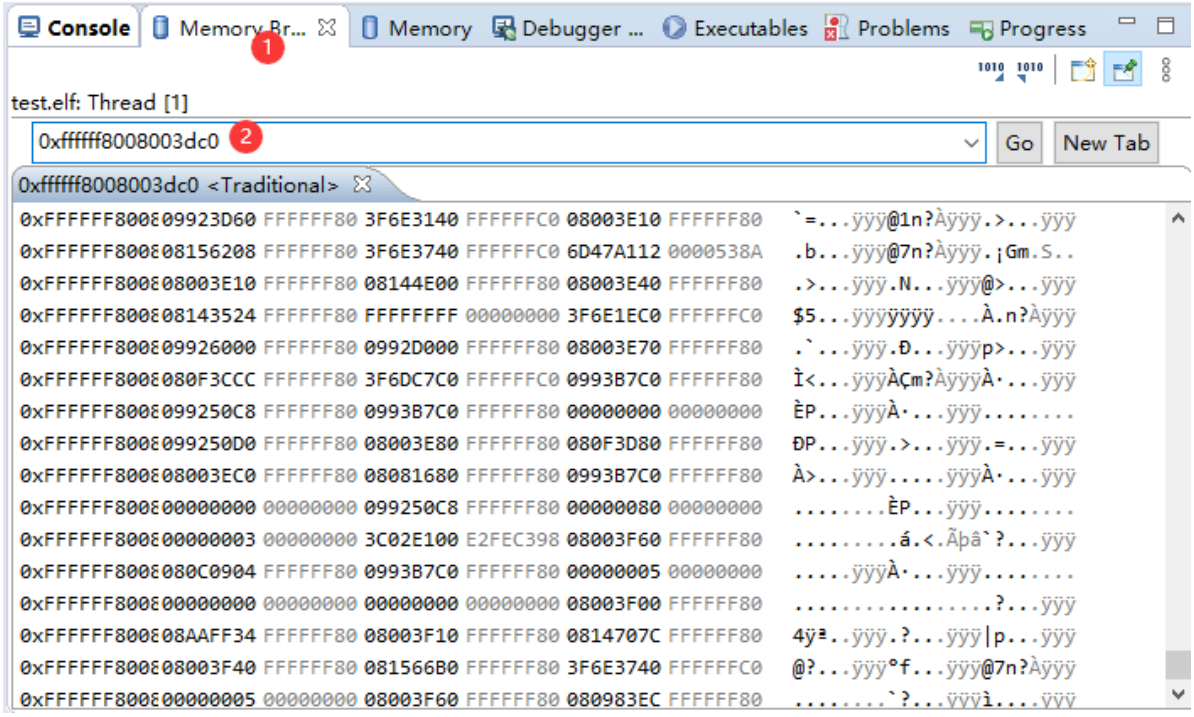
- 在从源代码窗口设置断点



- 从Disassembly窗口设置



4.10 查看内存数据



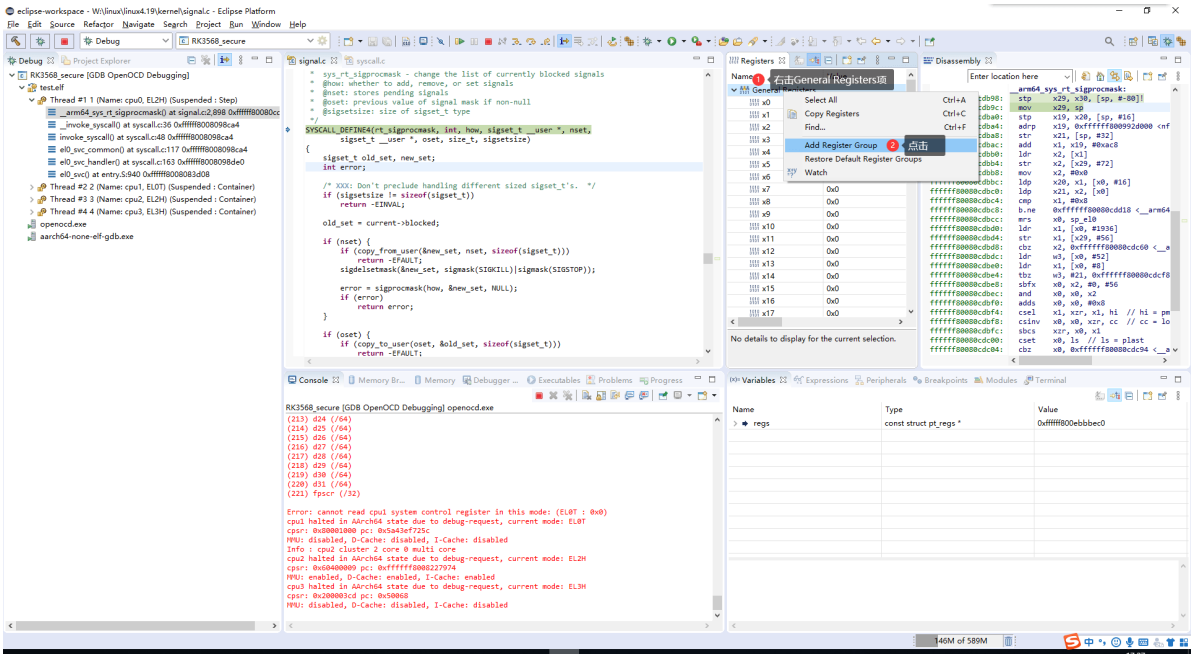
注意：Memory Browser窗口只能正常支持32位地址的访问，对于64位地址访问有问题，请进入OpenOCD命令行模式用mdw,mww,

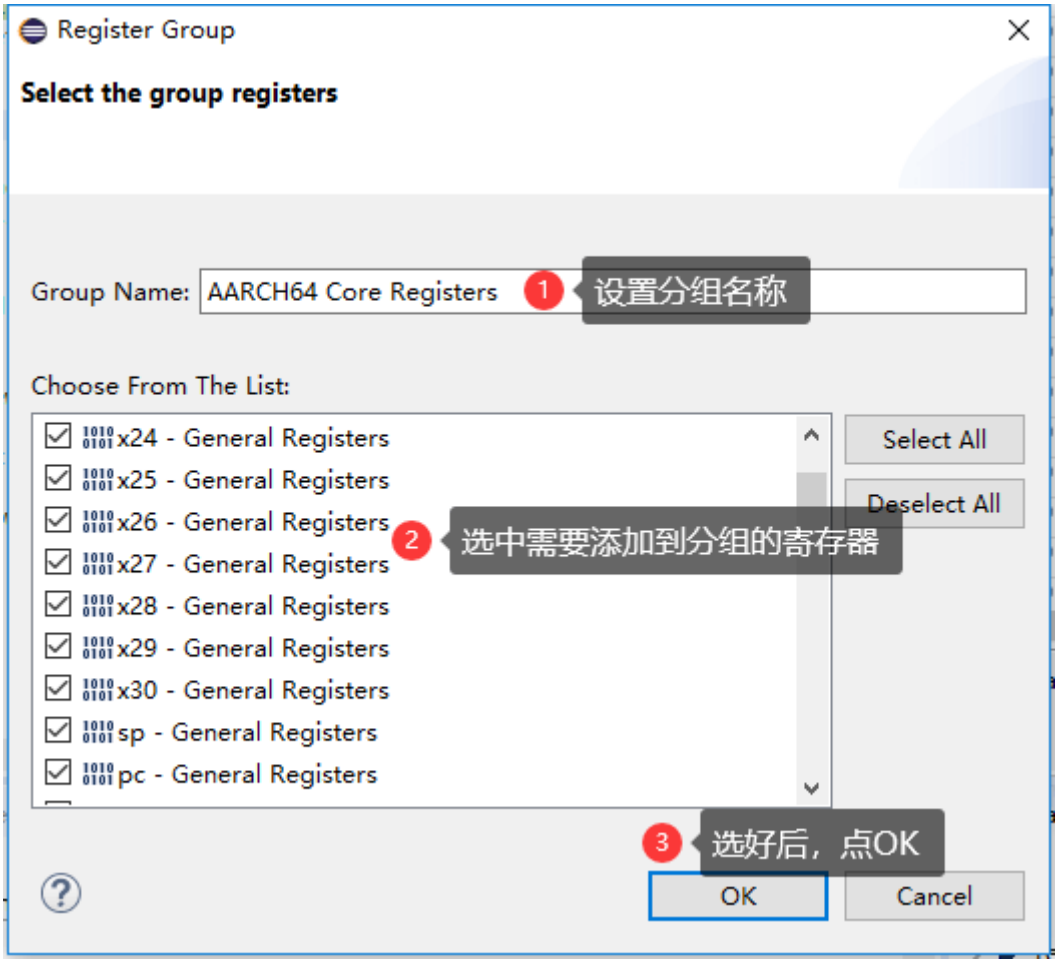
smdw,smww,io等操作内存。

5. 高级调试功能

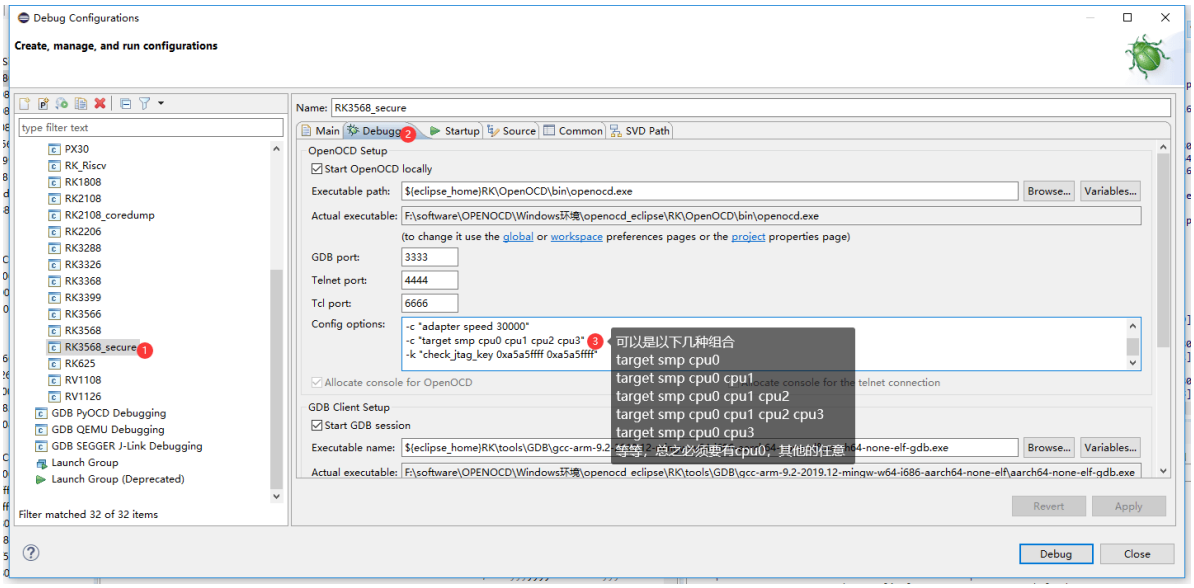
5.1 寄存器分组

由于寄存器很多，查看起来不方便，分组查看比较方便



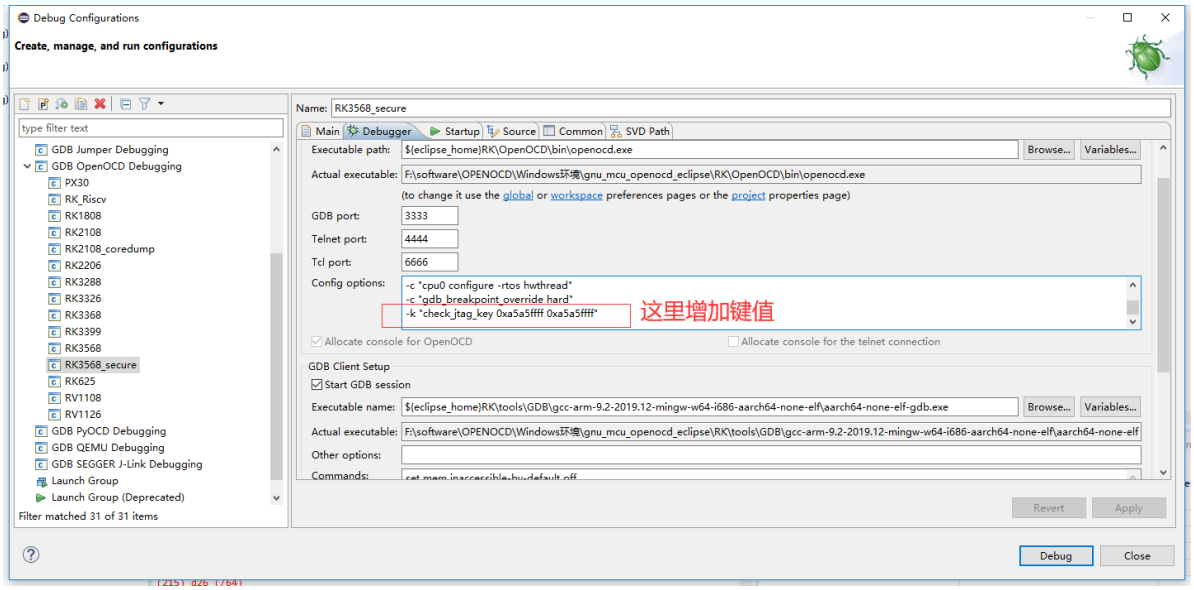


5.2 指定连接的CPU



5.3 安全调试

如果客户产品使能安全策略，那么JTAG需要输入key才能调试。



5.4 OpenOCD命令行模式（基于命令行终端）

在cmd窗口打开openocd，获取使用帮助

更多命令请参考：OpenOCD User’s Guide：

```
http://openocd.org/doc/html/General-Commands.html#General-Commands
```

openocd.exe -r 连接芯片

```
C:\Users\hhb>F:\software\OPENOCD\Windows环境\openocd_eclipse\RK\OpenOCD\bin\openocd.exe -r rk3568
Open On-Chip Debugger 0.10.0+dev-01526-g6d4a38b50-dirty (2021-06-10 15:47)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : Please follow steps below:
Info : step1: run (./openocd -r rk3399) to connect the target board
Info : step2: run (telnet localhost 4444) to enter command mode
Info : step3: telnet command mode, use the command below:
Info : (1) list the targets support
Info : (1 cpu0) set cpu0 as current target
Info : (pcsr) dump the pc of cpu when cpu is running
Info : (io) io -4 [-r(w)] [-l size] addr [write val]
Info : (halt) stop cpu
Info : (resume) resume cpu
Info : (step) step cpu
Info : (mdw) read memory in 4 bytes
Info : (mww) write memory in 4 bytes
Info : (smdw) secure read memory in 4 bytes
Info : (smww) secure write memory in 4 bytes
Info : (reg [x0]) display cpu reg
Info : (bp -h) add breakpoint
Info : (rbp -h) remove breakpoint
Info : (wp -h) remove watchpoint
Info : (rwp -h) remove watchpoint
Info : (asm pc [count ['thumb']]) disassemble
Info : (msr TCR_EL1 0x55aa55aa) write cpu system control register
Info : (mrs TCR_EL1) read cpu system control register
Info : load_image filename address ['bin'|'ihex'|'elf'|'s19'] [min_address] [max_length]
Info : verify_image filename [offset [type]]
Info : dump_image filename address size
Info : (help) list all openocd commands
adapter speed: 2000 kHz

Info : FTDI SWD mode enabled
Info : Hardware thread awareness created
force hard breakpoints
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 2000 kHz
Info : SWD DPIDR 0x2ba01477
Info : cpu0: hardware has 6 breakpoints, 4 watchpoints
Info : cpu1: hardware has 6 breakpoints, 4 watchpoints
Info : cpu2: hardware has 6 breakpoints, 4 watchpoints
Info : cpu3: hardware has 6 breakpoints, 4 watchpoints
Info : cpu0 cluster 0 core 0 multi core
Info : starting gdb server for cpu0 on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for cpu1 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for cpu2 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for cpu3 on 3336
Info : Listening on port 3336 for gdb connections
Info : accepting 'telnet' connection on tcp/4444
Info : accepting 'telnet' connection on tcp/4444
```

openocd路径

-r 芯片名字

连接步骤

常用命令

连接速率是2MHz

认到4个cpu

基于cmd窗口创建telnet连接，新打开一个cmd窗口，执行telnet localhost 4444

```
Telnet localhost
Open On-Chip Debugger
> 1
    TargetName         Type       Endian TapName            State
--  ------------------ ---------- ------ ------------------ ------------
 0  apb                mem_ap     little RK3568.cpu         running
 1* cpu0               aarch64    little RK3568.cpu         halted
 2  cpu1               aarch64    little RK3568.cpu         running
 3  cpu2               aarch64    little RK3568.cpu         running
 4  cpu3               aarch64    little RK3568.cpu         running

> mdw 0xffff6048 16
0xffff6048: 121e3529 340010e9 d0fee654 91154294 b940168a 3200014a 29022a89 b9430909
0xffff6068: 32010129 b9030909 b9401689 36000fa9 b940128a 7100055f 54000eab 11001155

> mdw phys 0xffff6048 16
0xffff6048: 121e3529 340010e9 d0fee654 91154294 b940168a 3200014a 29022a89 b9430909
0xffff6068: 32010129 b9030909 b9401689 36000fa9 b940128a 7100055f 54000eab 11001155

> io -4 -l 0x64 0xffff6048
0xffff6048: 121e3529 340010e9 d0fee654 91154294 b940168a 3200014a 29022a89 b9430909
0xffff6068: 32010129 b9030909 b9401689 36000fa9 b940128a 7100055f 54000eab 11001155
0xffff6088: 321e03f6 528019b7 528000b8 12828019 52801cfa b000001b 912f837b d0fee65c
0xffff60a8: 910c439c

> 1 cpu1
> 1
    TargetName         Type       Endian TapName            State
--  ------------------ ---------- ------ ------------------ ------------
 0  apb                mem_ap     little RK3568.cpu         running
 1  cpu0               aarch64    little RK3568.cpu         halted
 2* cpu1               aarch64    little RK3568.cpu         running
 3  cpu2               aarch64    little RK3568.cpu         running
 4  cpu3               aarch64    little RK3568.cpu         running

>
```

列出所有cpu

虚拟地址读

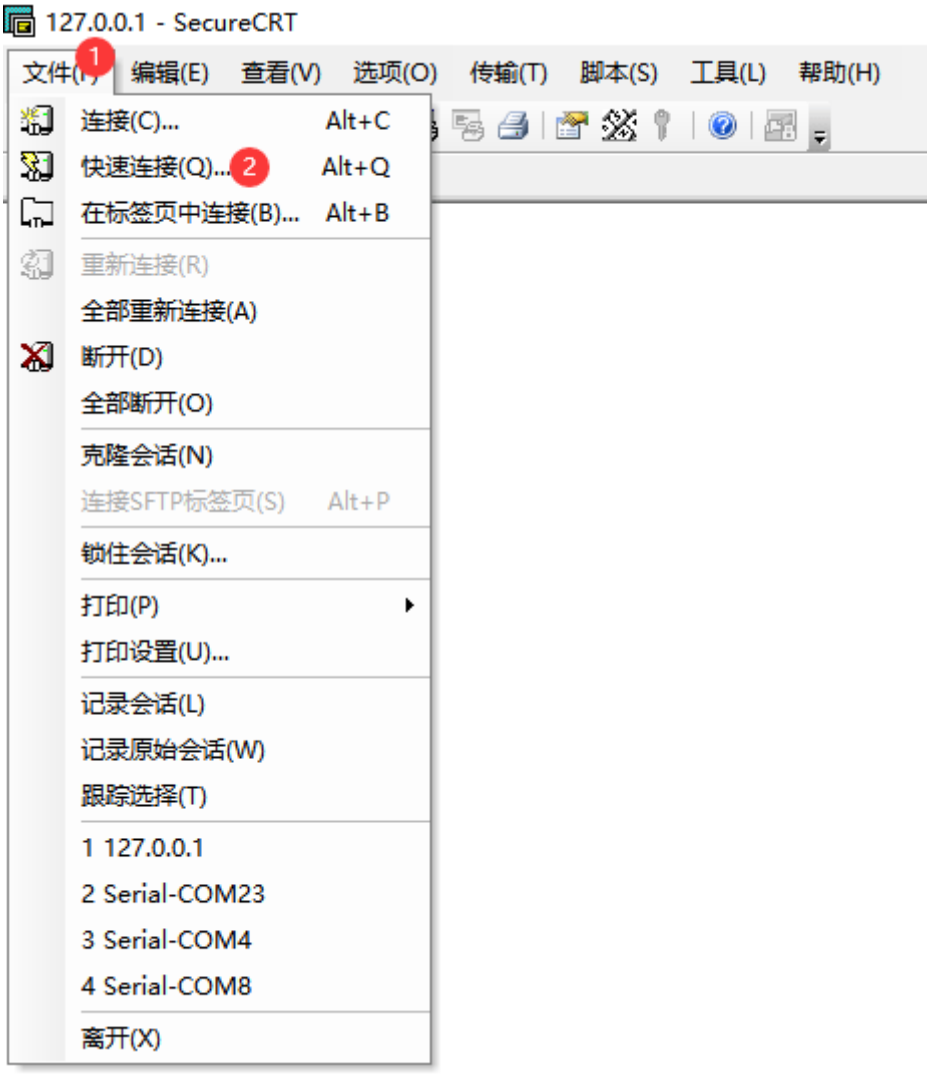
物理地址读

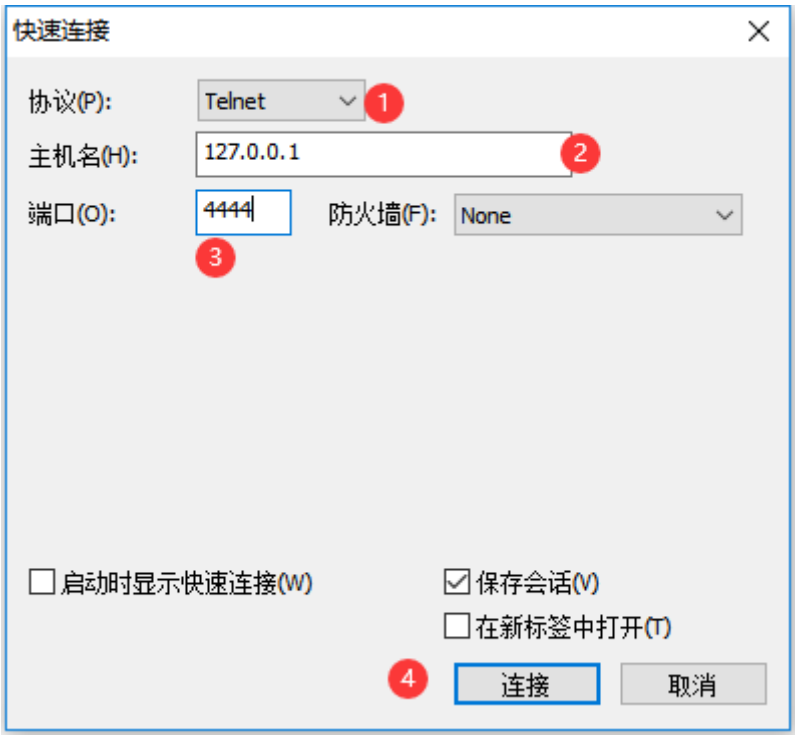
io命令物理地址读

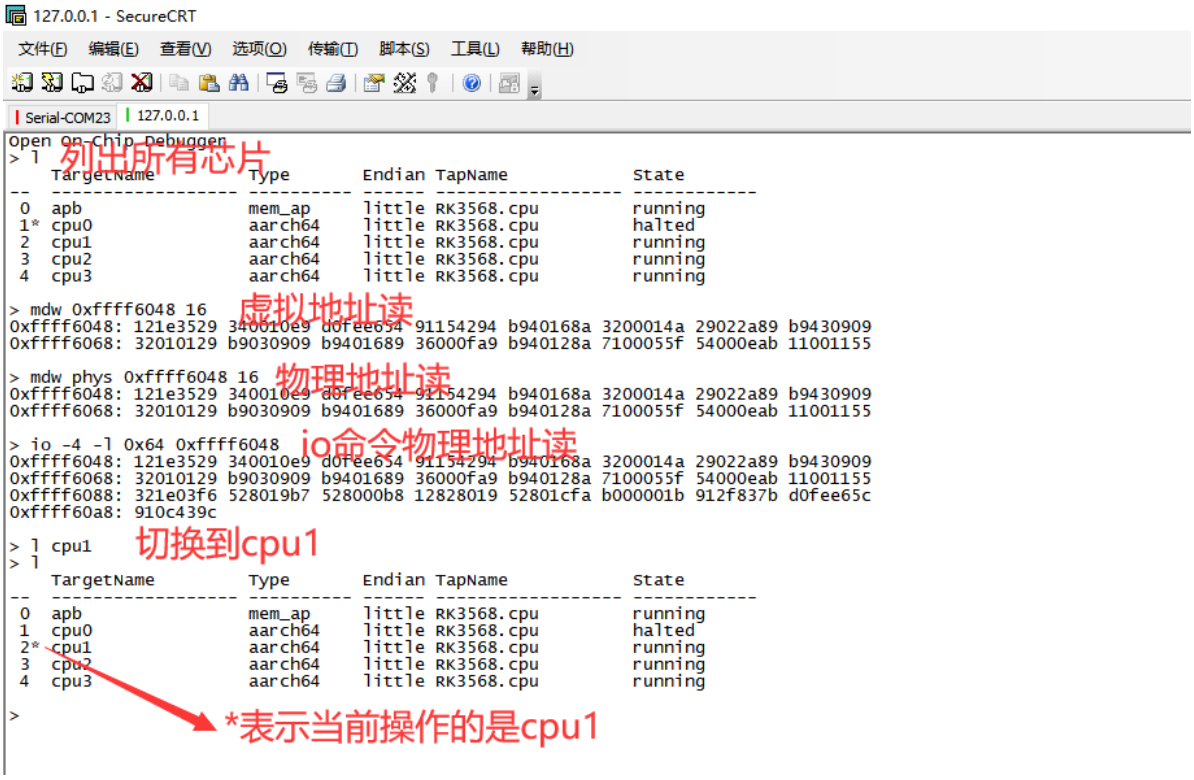
切换到cpu1

*表示当前操作的是cpu1

基于securecrt创建telnet连接



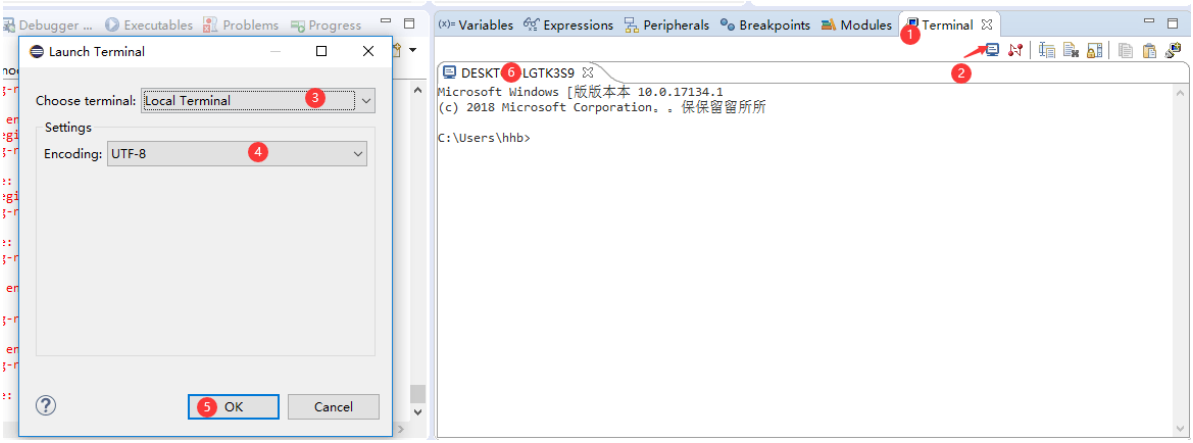




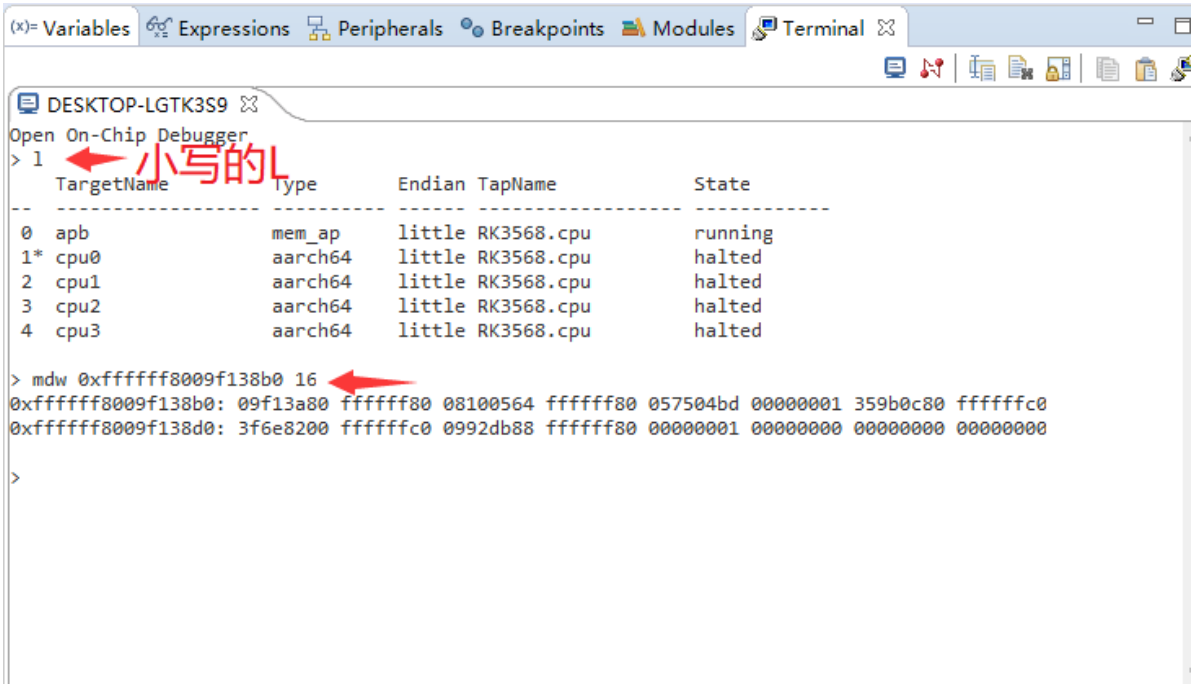
5.5 OpenOCD命令行模式（基于eclipse）

有时候有些信息UI无法很好的展示或者操作，这时就需要OpenOCD的命令行模式。

打开本地终端

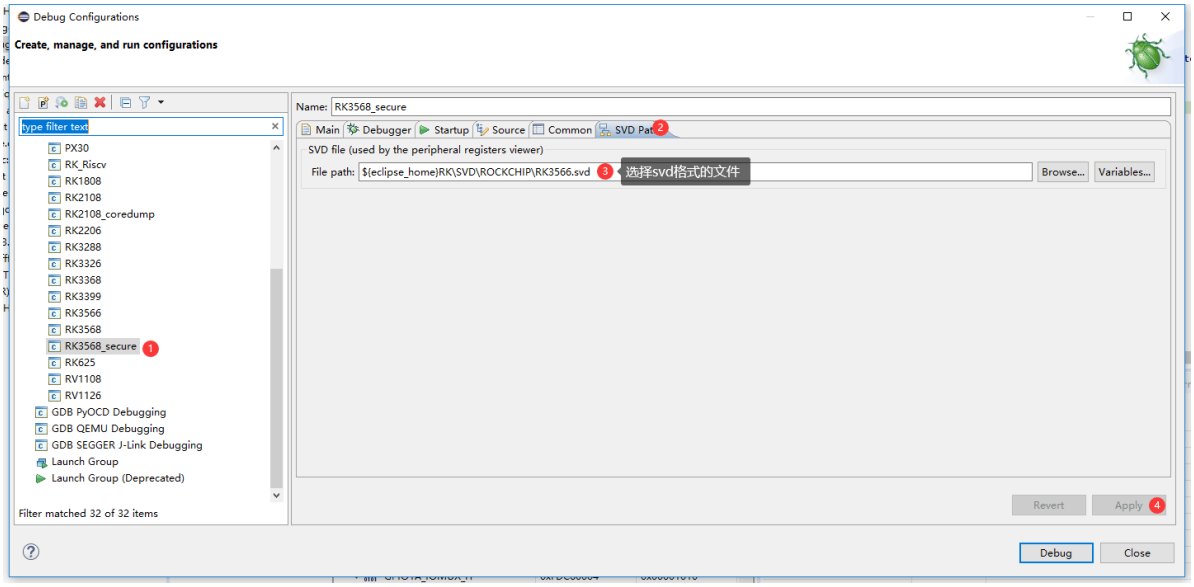


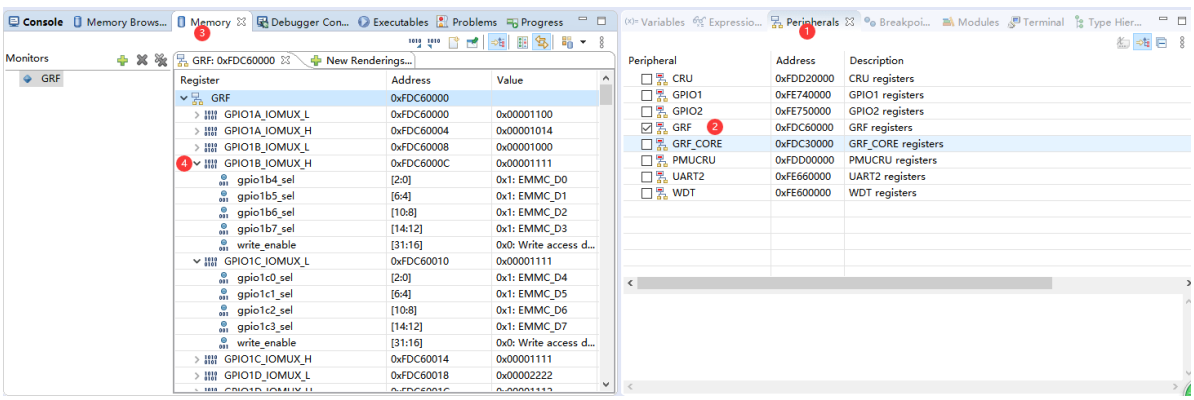
输入telnet localhost 4444进入命令行模式，如果提示不识别telnet命令，请参照《使能Windows telnet功能》章节。



5.6 查看可视化的外设寄存器

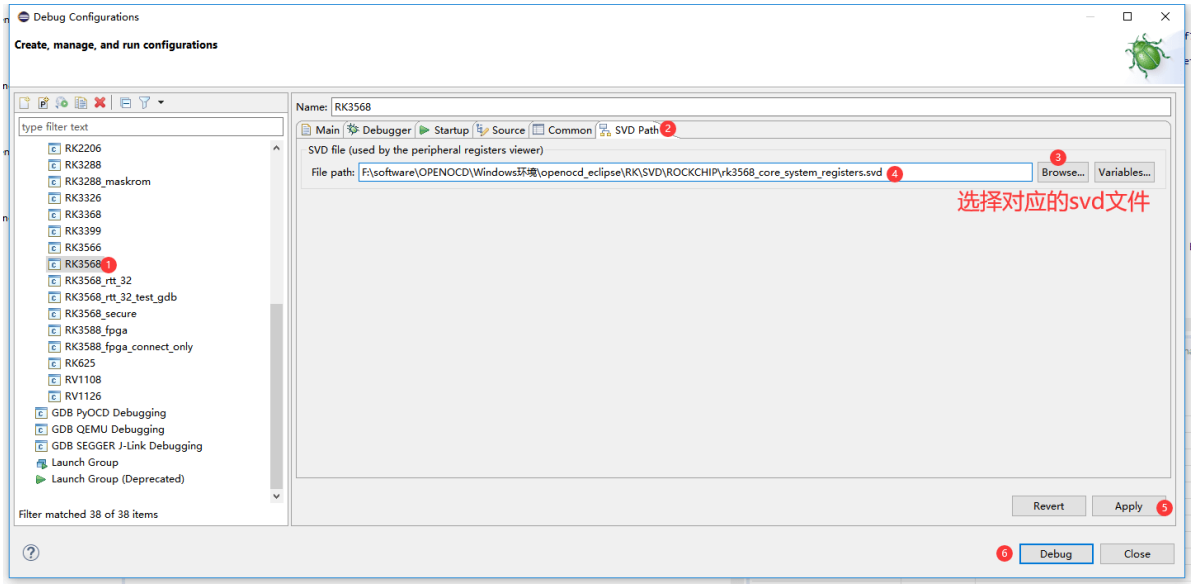
连接前，选择svd格式的文件

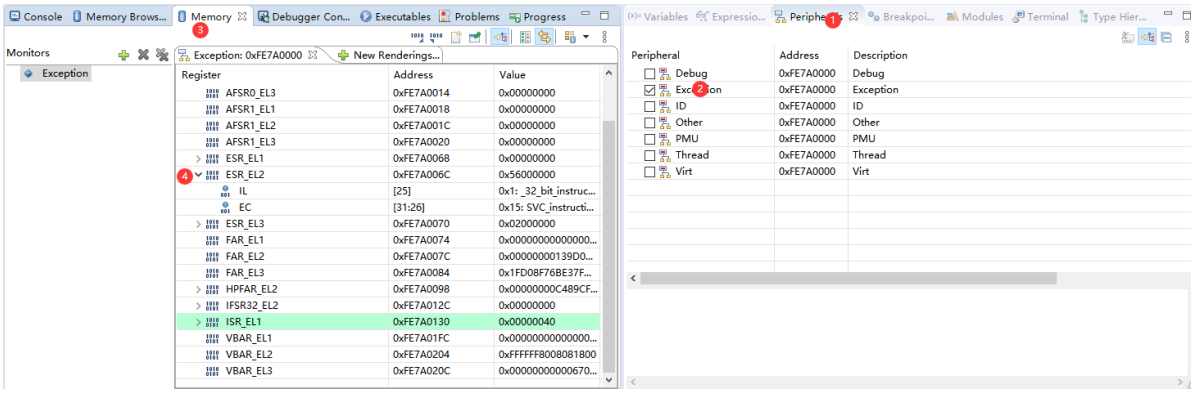




5.7 查看可视化的system control registers

连接前，选择svd格式的文件





5.8 从内存导出数据到文件

```
dump_image filename [p:]address size
虚拟地址访问:
> dump_image dd.bin 0xffffff8009a63c80 65536
dumped 65536 bytes in 0.113187s (565.436 KiB/s)
物理地址访问:
> dump_image dd.bin p:0x01c63c80 65536
dumped 65536 bytes in 0.094566s (676.776 KiB/s)
```

5.9 从文件导入数据到内存

```
load_image filename [p:]address ['bin'|'ihex'|'elf'|'s19'] [min_address]
[max_length]
虚拟地址访问:
> load_image dd.bin 0xffffff8009a63c80 bin
65536 bytes written at address 0xffffff8009a63c80
downloaded 65536 bytes in 0.055928s (1144.328 KiB/s)
物理地址访问:
> load_image dd.bin p:0x01c63c80 bin
65536 bytes written at address 0x01c63c80
downloaded 65536 bytes in 0.056400s (1134.752 KiB/s)
```

5.10 对比文件和内存里的数据是否一致

```
verify_image filename [offset [type]]
只支持虚拟地址访问:
> verify_image dd.bin 0xffffff8009a63c80
verified 65536 bytes in 0.066775s (958.443 KiB/s)
```

说明：上述3个命令都会涉及虚拟地址和物理地址。需要特别强调对于maskrom, tpl, spl, uboot, atf 等

可以认为虚拟地址跟物理地址是一样的，所以只需采用虚拟地址访问方式即可。但是对于linux，物理地址

和虚拟地址是不一样的，当需要访问物理地址时，需要在地址前面加p:。

同时filename是相对路径话，那么dd.bin就是保存在openocd运行的目录下。

6. 实际运用场景

6.1 调试CPU卡死问题

当出现画面卡死，或者黑屏，或者串口无法敲命令，或者内核报的相关错误，都表明CPU无法继续执行程序，可能卡死在某条或某几条指令上。

以调试RK3399开机死机为例子，如下死机时内核log，log基本停止，串口无法敲命令，显示画面不更新。

```
[    7.441882] type=1400 audit(1501952314.540:9): avc: denied { setattr } for
pid=1 comm="init" name="mmcblk1p13" dev="tmpfs" ino=15288 scontext=u:r:init:s0
tcontext=u:object_r:block_device:s0 tclass=lnk_file permissive=1
[    7.455950] init: Service 'exec 2 (/system/bin/vdc)' (pid 252) exited with
status 0
rk3399_all:/ $ [    7.589883] random: nonblocking pool is initialized
[    8.078474] init: Starting service 'bootanim'...
[    8.261358] read channel() error: -110
[   15.245727] BUG: spinlock lockup suspected on CPU#2, adbd/251
```

```
$ ./src/openocd.exe -r rk3399
Open On-Chip Debugger 0.10.0+dev-01555-g6ad367fa3-dirty (2021-08-04-17:44)
```

```
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
中间省掉很多log
Info : clock speed 2000 kHz
Info : SWD DPIDR 0x5ba02477
Info : cpu0: hardware has 6 breakpoints, 4 watchpoints
Info : cpu1: hardware has 6 breakpoints, 4 watchpoints
Info : cpu2: hardware has 6 breakpoints, 4 watchpoints
Info : cpu3: hardware has 6 breakpoints, 4 watchpoints
Info : cpu4: hardware has 6 breakpoints, 4 watchpoints
Info : cpu5: hardware has 6 breakpoints, 4 watchpoints
Info : starting gdb server for cpu0 on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for cpu1 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for cpu2 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for cpu3 on 3336
Info : Listening on port 3336 for gdb connections
Info : starting gdb server for cpu4 on 3337
Info : Listening on port 3337 for gdb connections
Info : starting gdb server for cpu5 on 3338
Info : Listening on port 3338 for gdb connections
```

另外再打开一个cmd窗口，输入telnet localhost 4444 进入命令行调试

```
Open On-Chip Debugger
> cpu_block     查看针对cpu卡死问题，有哪些命令可以用
l  查看cpu状态
dump_pc    多次打印pc指针
halt   停住cpu
if halt fail, try fhalt    强制cpu进入debug 状态
if fhalt success, try core_reg64 or core_reg32, but you can't access memory
> l
    TargetName         Type       Endian TapName            State
--  ------------------ ---------- ------ ------------------ ------------
 0  ahb                mem_ap     little RK3399.cpu         running
 1  apb                mem_ap     little RK3399.cpu         running
 2* cpu0               aarch64    little RK3399.cpu         running
 3  cpu1               aarch64    little RK3399.cpu         running
 4  cpu2               aarch64    little RK3399.cpu         running
 5  cpu3               aarch64    little RK3399.cpu         running
 6  cpu4               aarch64    little RK3399.cpu         running
 7  cpu5               aarch64    little RK3399.cpu         running
> dump_pc
1
cpu0 pc:0xffffff8008143a88
cpu1 pc:0xffffff8008143a5c
cpu2 pc:0xffffff8008081078
cpu3 pc:0xffffff8008081078
cpu4 pc:0xffffff80081133c8
cpu5 pc:0xffffff800808248c
2
cpu0 pc:0xffffff8008143a88
cpu1 pc:0xffffff8008143a5c
cpu2 pc:0xffffff8008081078
```

```
cpu3 pc:0xffffff8008081078
cpu4 pc:0xffffff80081133c8
cpu5 pc:0xffffff800808248c
3
cpu0 pc:0xffffff8008143a88
cpu1 pc:0xffffff8008143a5c
cpu2 pc:0xffffff8008081078
cpu3 pc:0xffffff8008081078
cpu4 pc:0xffffff80081133c8
cpu5 pc:0xffffff800808248c
4
cpu0 pc:0xffffff8008143a88
cpu1 pc:0xffffff8008143a5c
cpu2 pc:0xffffff8008081078
cpu3 pc:0xffffff8008081078
cpu4 pc:0xffffff80081133c8
cpu5 pc:0xffffff800808248c
5
cpu0 pc:0xffffff8008143a88
cpu1 pc:0xffffff8008143a5c
cpu2 pc:0xffffff8008081078
cpu3 pc:0xffffff8008081078
cpu4 pc:0xffffff80081133c8
cpu5 pc:0xffffff800808248c     经过5次打印，发现6个cpu的pc指针都没变，那么很可能是卡死了

> halt
Timeout waiting for target cpu0 halt    第一次尝试停cpu0失败

> fhalt
cpu0 fail to force entry to debug state; prsr:00000001  第二次尝试停cpu0失败，这时只能根据打印出来的pc指针来查问题了

> l cpu1

> halt
Timeout waiting for target cpu1 halt

> fhalt
cpu1 fail to force entry to debug state; prsr:00000001
> l cpu2
> halt
Timeout waiting for target cpu2 halt

> fhalt
cpu2 success to force entry to debug state; prsr:00000011   终于可以停住cpu2

> core_reg64      可以打印cpu通用寄存器，以便进一步分析
x0 (/64): 0xffffffc00a3d7e00
x1 (/64): 0xffffff8008081028
x2 (/64): 0x0000000000000000
x3 (/64): 0x00000040edbb6000
x4 (/64): 0x0100000000000000
x5 (/64): 0x000000000d8031bf
x6 (/64): 0x000000000000126a
x7 (/64): 0xffffff8008141ee0
x8 (/64): 0x0000000000000000
x9 (/64): 0x0000000000000000
x10 (/64): 0x00000000000012b0
```

```
x11 (/64): 0x000000000000000f
x12 (/64): 0x0000000100000000
x13 (/64): 0x0000000000000001
x14 (/64): 0x0000000000000000
x15 (/64): 0x0000000000000000
x16 (/64): 0xffffff80080b8b28
x17 (/64): 0x0000007cf3aeee00
x18 (/64): 0x0000000000000000
x19 (/64): 0x000000000000001e
x20 (/64): 0x0000008000000000
x21 (/64): 0xffffff80094e7990
x22 (/64): 0xffffff8009357ea0
x23 (/64): 0xffffffc00a3d7e00
x24 (/64): 0x0000000000000001
x25 (/64): 0xffffffc0f6ed4060
x26 (/64): 0xffffffc0f6ed8050
x27 (/64): 0x0000000000ed9234
x28 (/64): 0xffffffc00a3cad00
x29 (/64): 0xffffffc0f6ed8000
x30 (/64): 0xffffff8008083230
sp (/64): 0xffffffc0f6ed8000
pc (/64): 0xffffff800808107c

> mrs SCTLR_EL1   也可以读system control寄存器
Cannot reach EL 0, SPSR corrupted?
SCTLR_EL1:0x34d5d91d

> mrs SCTLR_EL2
Cannot reach EL 0, SPSR corrupted?
SCTLR_EL2:0x30c50830

> mrs SCTLR_EL3
Cannot reach EL 0, SPSR corrupted?
SCTLR_EL3:0x00cd383f
```

执行objdump.exe -d vmlinux | less，反汇编查看pc指针所处的函数和指令

```
ffffff8008081028 <gic_handle_irq>:
ffffff8008081028:       a9bc7bfd        stp     x29, x30, [sp, #-64]!
ffffff800808102c:       910003fd        mov     x29, sp
ffffff8008081030:       a9025bf5        stp     x21, x22, [sp, #32]
ffffff8008081034:       d00096b6        adrp    x22, ffffff8009357000
<event_hash+0x288>
ffffff8008081038:       a90363f7        stp     x23, x24, [sp, #48]
ffffff800808103c:       d000a335        adrp    x21, ffffff80094e7000
<event_class_drv_config+0x40>
ffffff8008081040:       aa0003f7        mov     x23, x0
ffffff8008081044:       913a82d6        add     x22, x22, #0xea0
ffffff8008081048:       912642b5        add     x21, x21, #0x990
ffffff800808104c:       a90153f3        stp     x19, x20, [sp, #16]
ffffff8008081050:       52800038        mov     w24, #0x1
// #1
ffffff8008081054:       d503201f        nop
ffffff8008081058:       14000008        b       ffffff8008081078
<gic_handle_irq+0x50>
ffffff800808105c:       71003e7f        cmp     w19, #0xf
```

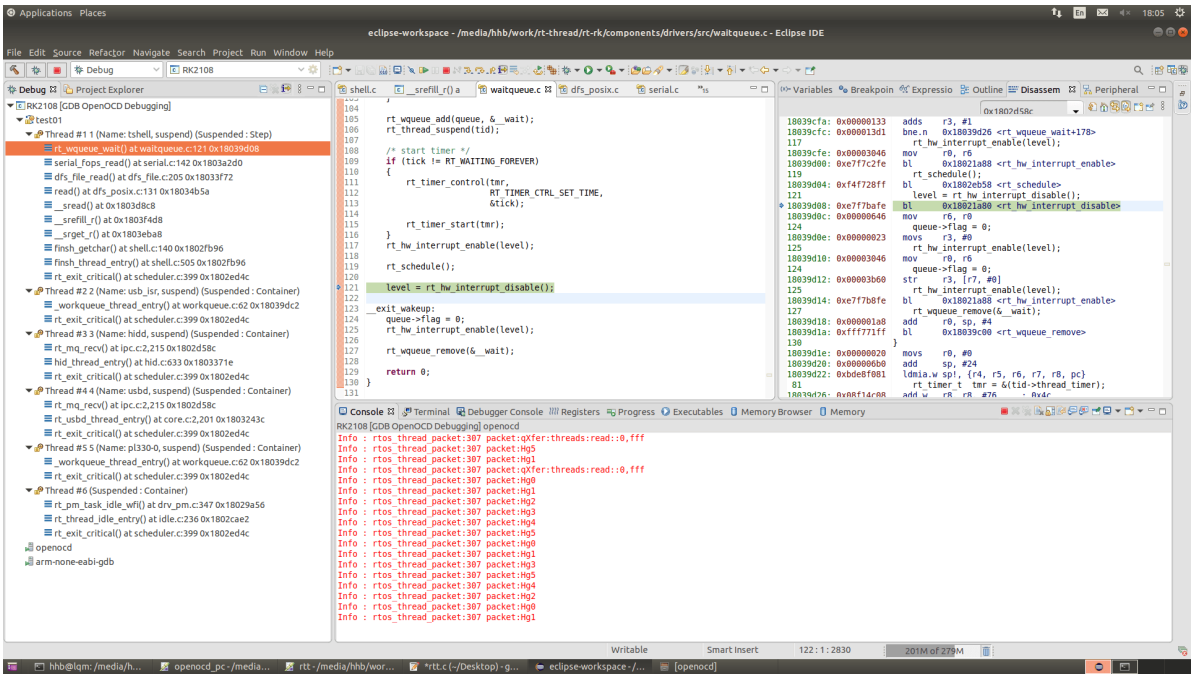
```
ffffff8008081060:       540007e8        b.hi    ffffff800808115c
<gic_handle_irq+0x134>  // b.pmore
ffffff8008081064:       92407e60        and     x0, x19, #0xffffffff
ffffff8008081068:       d518cc20        msr     s3_0_c12_c12_1, x0
ffffff800808106c:       d5033fdf        isb
ffffff8008081070:       d503201f        nop
ffffff8008081074:       14000035        b       ffffff8008081148
<gic_handle_irq+0x120>
ffffff8008081078:       d538cc13        mrs     x19, s3_0_c12_c12_0
*ffffff800808107c:       d5033f9f        dsb     sy      cpu2卡死在这个位置
ffffff8008081080:       51004260        sub     w0, w19, #0x10
ffffff8008081084:       2a1303f4        mov     w20, w19
ffffff8008081088:       710fac1f        cmp     w0, #0x3eb
ffffff800808108c:       54000089        b.ls    ffffff800808109c
<gic_handle_irq+0x74>  // b.plast
ffffff8008081090:       5283ffe0        mov     w0, #0x1fff
// #8191
ffffff8008081094:       6b00027f        cmp     w19, w0
ffffff8008081098:       54fffe29        b.ls    ffffff800808105c
<gic_handle_irq+0x34>  // b.plast
```

经过以上步骤，客户如果还是无法分析死机原因，可以将上述命令的log发给Rockchip的工程师分析。

6.2 RT-Thread调试说明

默认只显示单线程，添加以下命令后，可以显示多线程

```
-c "cpu0 configure -rtos RT_Thread"
```



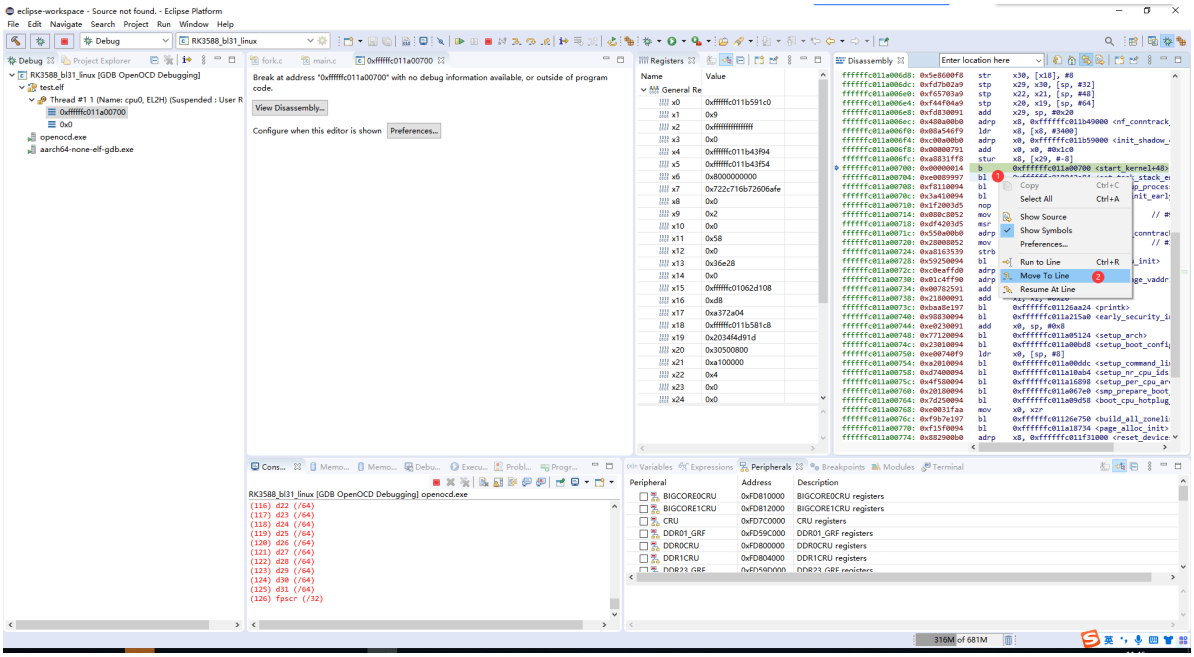
6.3 单步调试Linux

如果内核启动过程中，没任何内核的log输出，排除串口问题的话，那么比较可能是内核启动早期异常了。

6.3.1 内核单核阶段

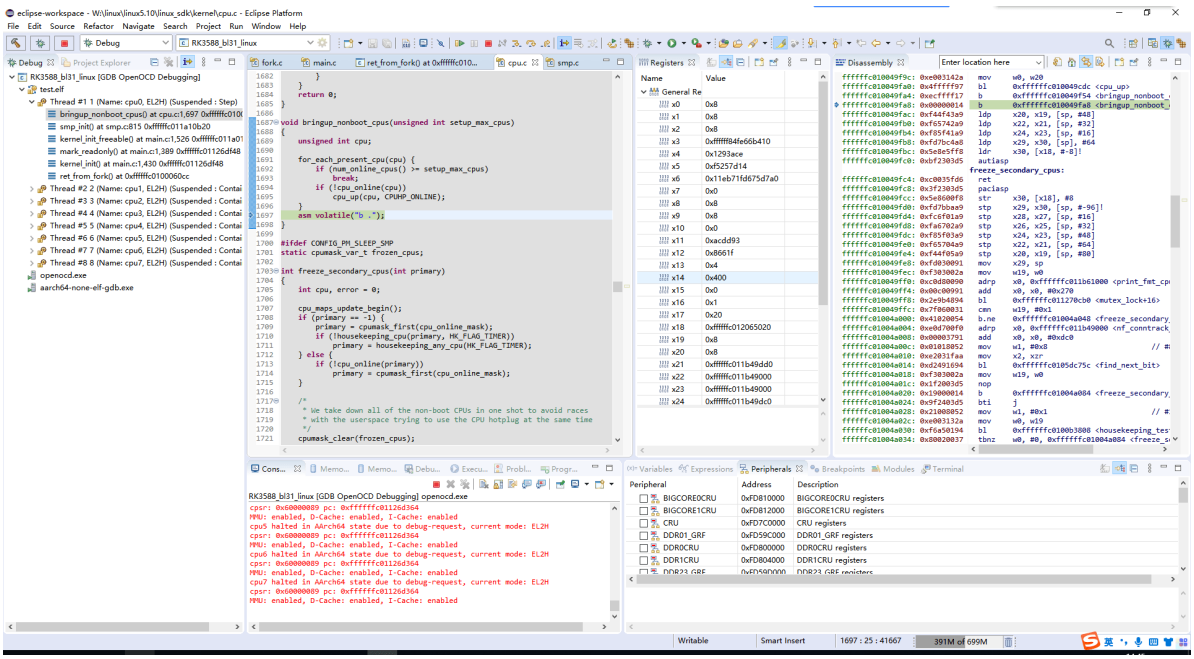
```
linux5.10/init/main.c
asmlinkage __visible void __init __no_sanitize_address start_kernel(void)
{
    char *command_line;
    char *after_dashes;
    asm volatile("b .");
 ...
 }
```

如下图，右击b . 指令的下一条指令，选择Move To line跳到下一条指令，进行单步调试



6.3.2 内核多核阶段

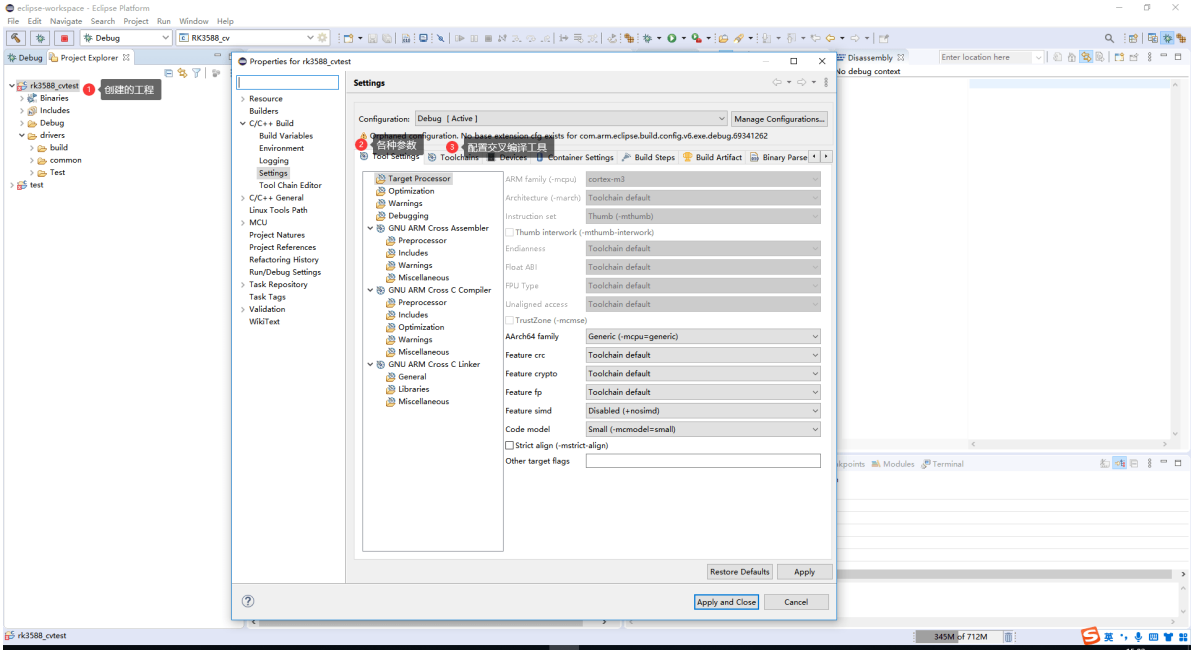
```
linux5.10/kernel/cpu.c
void bringup_nonboot_cpus(unsigned int setup_max_cpus)
{
    unsigned int cpu;

    for_each_present_cpu(cpu) {
        if (num_online_cpus() >= setup_max_cpus)
            break;
        if (!cpu_online(cpu))
            cpu_up(cpu, CPUHP_ONLINE);
    }
    asm volatile("b .");
}
```



6.4 裸机调试

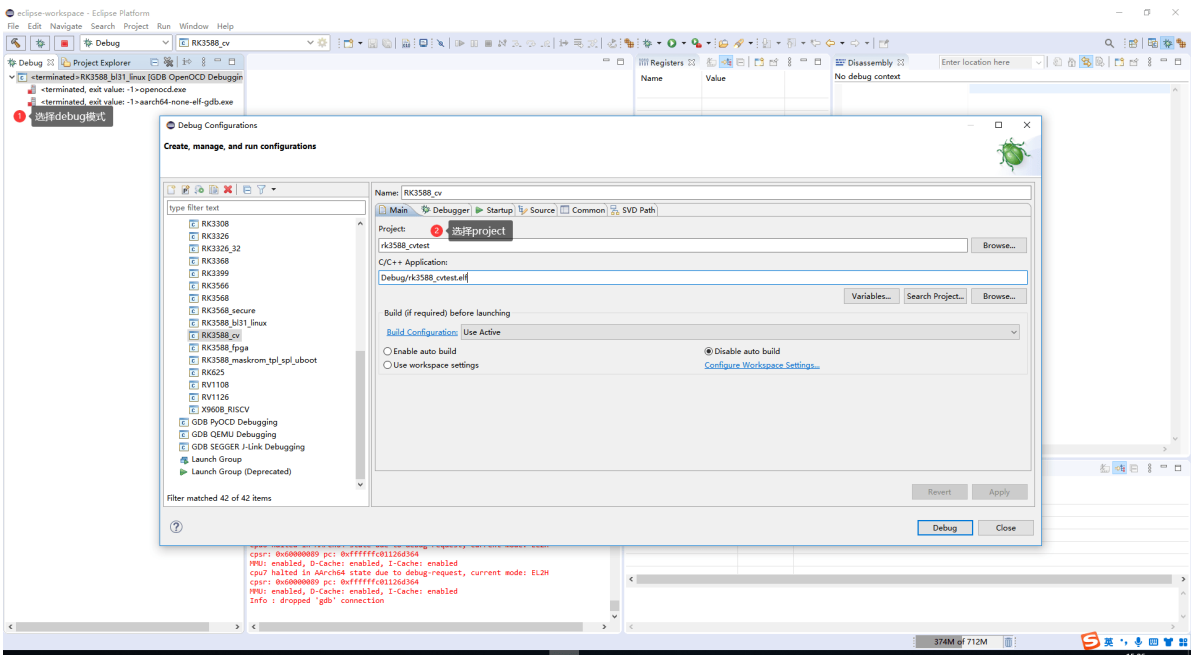
6.4.1 创建工程

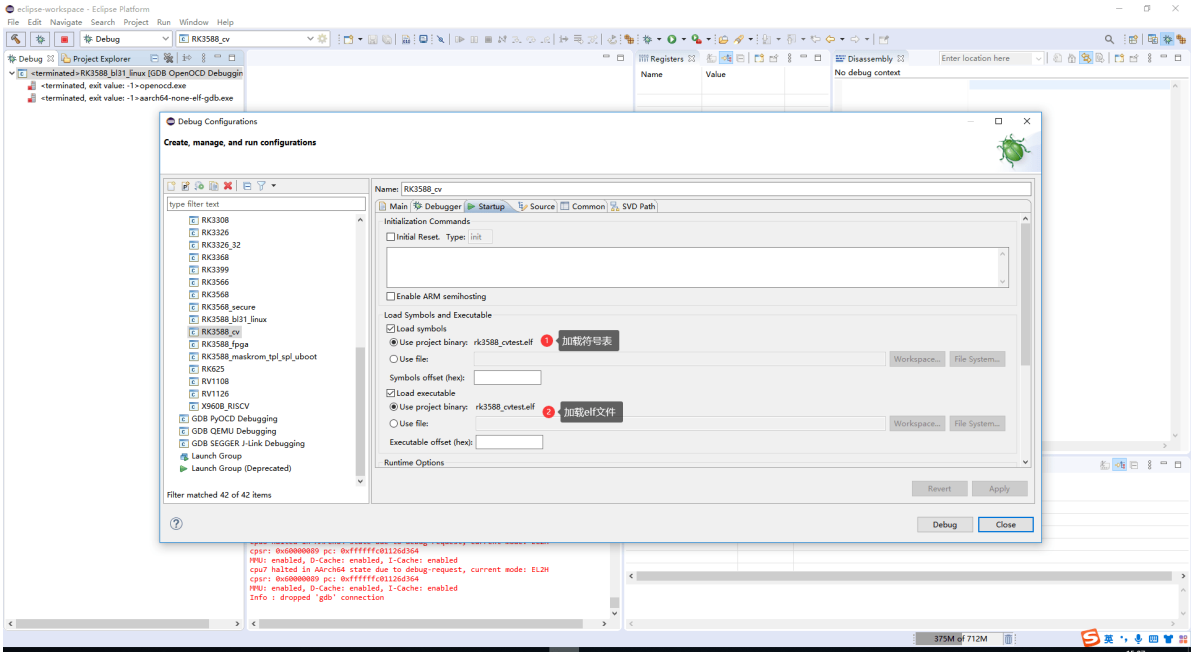
- 安装eclipse的标准流程创建工程能
- 配置编译器的各种参数
- 配置交叉编译器



6.4.2 加载并运行固件

注意：需要在哪个芯片上调试，就复制一份该芯片的配置，然后在这基础上修改





6.5 kgdb使用

6.5.1 kgdb用途

- 调试某个驱动
- 出现非法指针时，用来回溯整个调用栈，判断指针出错的点
- 采用串口通讯，而非JTAG接口，方便快捷

6.5.2 kgdb使能

- menuconfig配置

```
Device Drivers  --->
   Character devices  --->
      [*]   Virtual terminal
```

```
           [*]     Enable character translations in console (NEW)
           [*]     Support for console on virtual terminal (NEW)
-> Kernel hacking
  -> Generic Kernel Debugging Instruments
    [*] KGDB: kernel debugger  --->
      --- KGDB: kernel debugger
      [*]   KGDB: use kprobe blocklist to prohibit unsafe breakpoints (NEW)
      <*>   KGDB: use kgdb over the serial console (NEW)

Kernel hacking  --->
   Debug Oops, Lockups and Hangs  --->
     (0) panic timeout     这个要配成0，出现die时才会进入kgdb
 或者echo 0 > /sys/module/kernel/parameters/panic  将panic timeout时间设为0，即
不reboot
```

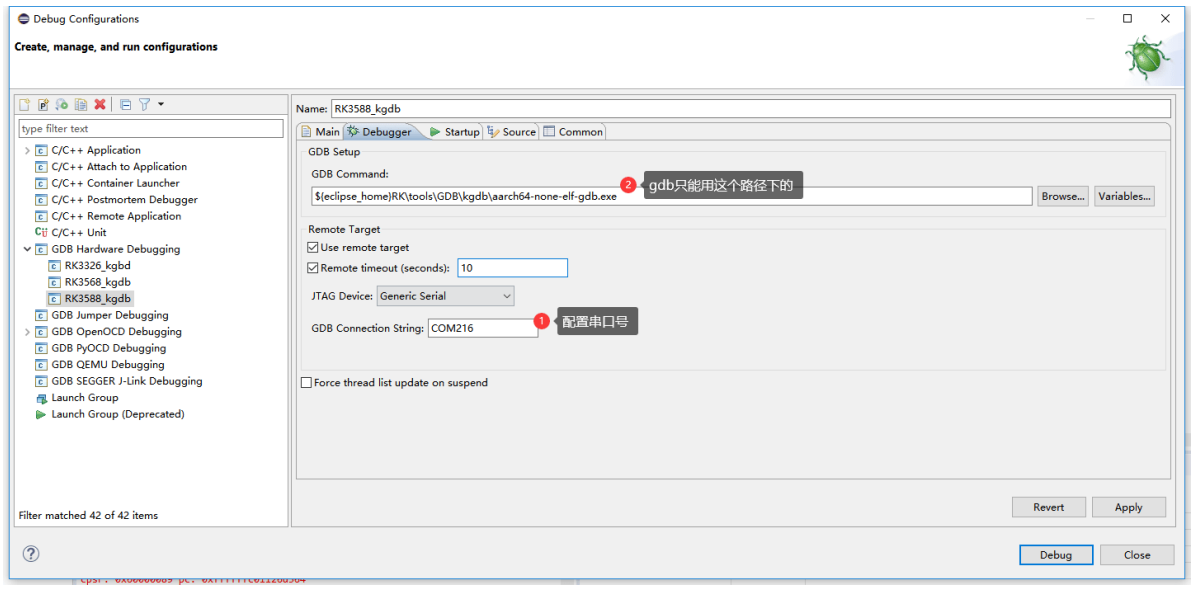
- 内核修改

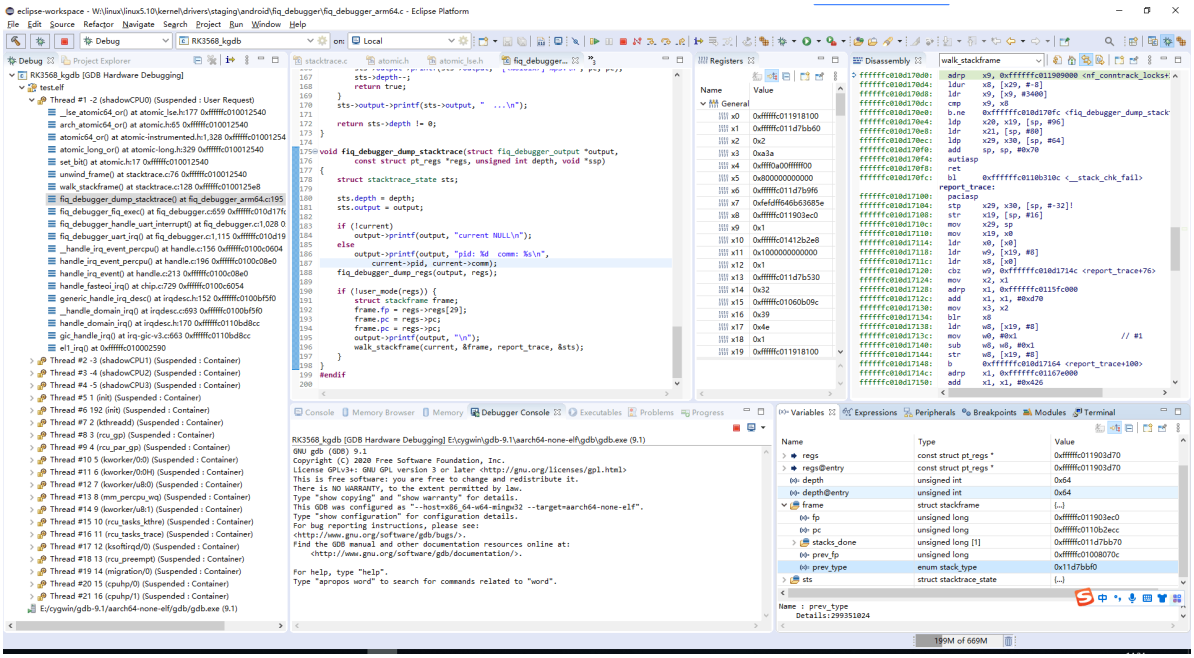
```
arch/arm64/boot/dts/rockchip/rk3588-android.dtsi
chosen {
    bootargs = "kgdboc_earlycon kgdboc=ttyFIQ0,1500000";
};
```

- 验证生效

```
echo g > /proc/sysrq-trigger

[    1.385017][    T0] kgdboc: No suitable earlycon yet, will try later
[    1.385069][    T0] earlycon: uart8250 at MMIO32 0x00000000feb50000
(options '')
[    1.390726][    T0] printk: bootconsole [uart8250] enabled
[    2.956193][    T0] kgdboc: Going to register kgdb with earlycon 'uart'
[    2.956196][    T0] KGDB: Registered I/O driver kgdboc_earlycon
[    3.092139][    T1] Unable to handle kernel NULL pointer dereference at
virtual address 000000000000007f
[    3.093010][    T1] Mem abort info:
[    3.093329][    T1]   ESR = 0x96000005
[    3.093672][    T1]   EC = 0x25: DABT (current EL), IL = 32 bits
[    3.094219][    T1]   SET = 0, FnV = 0
[    3.094561][    T1]   EA = 0, S1PTW = 0
[    3.094912][    T1] Data abort info:
[    3.095238][    T1]   ISV = 0, ISS = 0x00000005
[    3.095651][    T1]   CM = 0, WnR = 0
[    3.095987][    T1] [000000000000007f] user address but active_mm is
swapper
[    3.096633][    T1] Internal error: Oops: 96000005 [#1] PREEMPT SMP
[    3.097206][    T1] KGDB: Waiting for remote debugger
```

6.5.3 kgdb配置





6.6 aarch64 32位模式的调试

-c "cpu0 aarch64_32 "