应用笔记

Application Note

文档编号：**AN1128**

G32R501 双核 Debug 指导手册

版本：**V1.2**

1 引言

G32R5xx 双核微控制器（MCU）系列如表格 1 所述。本文所述的适用产品（在本文中称为 G32R5xx 微控制器）基于高性能 Arm® Cortex®-M52 32 位 RISC 内核。为了充分利用双核架构，G32R501 系列微控制器要求特定的开发方法。

注意：

（1） 本应用笔记**仅适用于 G32R501 双核封装（例如 Dx）**（见表格 1）。

（2） **G32R502 以及无 CPU1 的器件不适用于**本文的双核调试步骤。

（3） 从核启动 API、寄存器与 IPC 例程细节以 **AN1135** 及 “driverlib/g32r501/examples/eval/ipc/” 下源码为准。本文侧重三套工具链上的双核调试连接步骤。

本调试手册提供了在 G32R501 微控制器上调试自定义应用程序的指南，涵盖以下方面：

- G32R5xx 双核微控制器调试基本原理。
- 如何使用支持 Geehy-Link 调试的 EWARM、MDK-ARM、Eclipse 工具链调试双核设备。

有关 G32R501 微控制器的更多信息，请参考以下文档：

- G32R501 系列数据手册
- G32R501 系列用户手册
- AN1135 G32R501 IPC 应用笔记

本文适用的双核型号见表格 1。

表格 1 G32R5xx 双核型号

通用系列	具体支持产品型号
G32R5xx	G32R501DxCx7/G32R501DxYx7/G32R501DxYx8Q

注意：建议先打开 SDK 中 “G32R5xx_SDK\driverlib\g32r501\examples\eval\ipc\” 下的 IPC 双核工程，再对照本文理解相关设置。自行搭建的工程须完整落实第 4.2 / 5.2 / 6.2 节中的检查点。

目录

2 双核的基本机制

多核处理器由异构核（意味着不同内核）或同构（相同）内核组成。

G32R5xx 中的双核属于非对称体系结构，默认情况下，CPU0 被设置为 master，可以正常工作，而 CPU1 被设置为 slave，当芯片启动时，CPU1 被设置为 hold，它的时钟被禁用。要让从核工作，需要 CPU0 通过寄存器来使能其时钟，并设置其启动地址（例程中通常封装为“APP_bootCPU1”，内部调用“SysCtl_setCPU1BootAddress”与“SysCtl_enableBootCPU1”）。

2.1 功能特点

G32R501 系列微控制器能够提供全面且灵活的调试和性能分析支持。

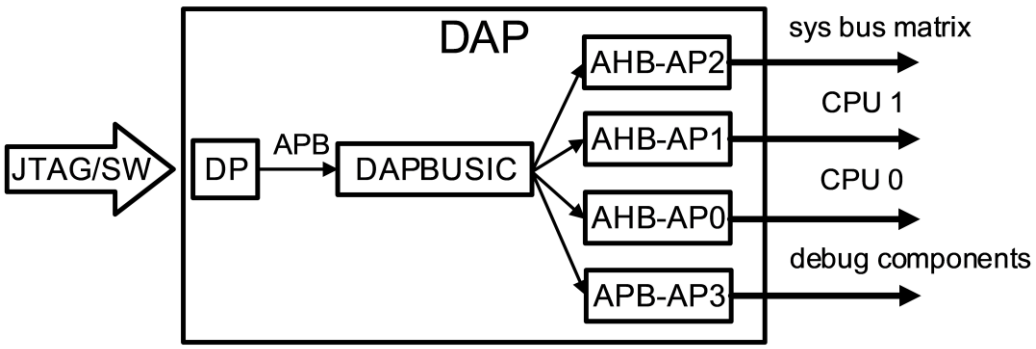
- **独立断点调试**：支持系统中每个 CPU 内核的独立断点调试，便于多核系统的精细调试。
- **代码执行跟踪**：支持对代码执行过程进行跟踪，有助于性能分析和调试。
- **JTAG 调试端口**：提供标准的 JTAG 调试接口，兼容广泛的调试工具。
- **串行线调试端口**：支持串行线调试端口（Serial Wire Debug Port），用于简化调试连接。

2.2 调试访问端口（DAP）

G32R501 微控制器包含四个连接到调试端口（DP）的访问端口（AP）：

- AHB-AP0：CPU0 访问端口（AHB-AP）通过连接到处理器的 AHBD 端口的 AHB-Lite 总线，提供对 CPU0 内核中集成的调试和跟踪功能的访问。
- AHB-AP1：CPU1 访问端口（AHB-AP）通过连接到处理器的 AHBD 端口的 AHB-Lite 总线，提供对 CPU1 内核中集成的调试和跟踪功能的访问。
- AHB-AP2：总线矩阵接入口。允许访问系统总线矩阵。
- APB-AP3：调试访问端口。允许访问外部调试组件。

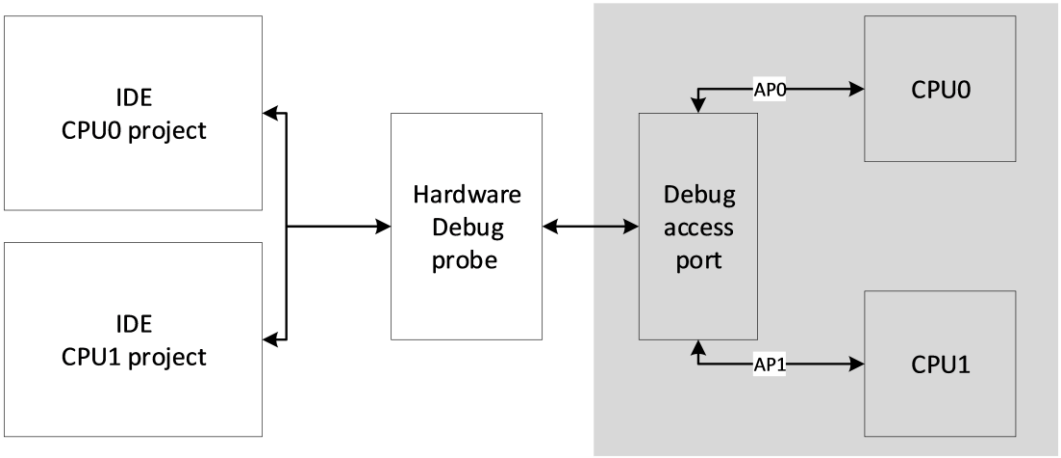
图 1 G32R5xx 调试访问端口（DAP）



3 调试支持

双核调试允许使用单个硬件调试探针同时调试两个内核。两个内核的调试信息可以在单个集成开发环境（IDE）图形用户界面（GUI）中显示，也可以为每个内核单独创建一个 IDE GUI 实例。分开的 IDE GUI 实例的表现如图 2 所示。

图 2 双核调试 IDE 示意



为了确保顺利进行双核调试，使用的调试器必须提供以下功能：

- 可选访问端口。
- 使用相同的调试探针同时连接多个内核的能力。
- 所有内核的可见性。
- 支持交叉触发 Arm®组件。
- 在同一调试会话中在不同领域之间切换访问端口的可能性，以可视化内存和外围设备的状态。

4 MDK-ARM 双核调试支持

最新版本的 MDK-ARM 可以从其官方网站下载。

4.1 在 MDK-ARM 上进行双核调试

如前文描述，G32R5xx 中的双核非对称系统需要特定的开发工具，在 MDK-ARM IDE v5.40 以上便支持对 G32R5xx 进行双核调试。

4.2 使用 GEEHY-LINK（WinUSB）进行双核调试的步骤

本节提供了使用 MDK-ARM v5.40 和 GEEHY-LINK（WinUSB）调试探针与 G32R5xx 微控制器配合工作的分步说明。

注意：

（1） Arm® Cortex®-M52 的双核调试在 MDK-ARM v5.40 及更高版本中得到支持。

（2） 在进行下面的操作前，请先安装 G32R5xx 芯片支持（**Geehy.G32R5xx_DFP.x.x.x.pack**）。

（3） 在此示例中，需要为每个内核创建一个项目。

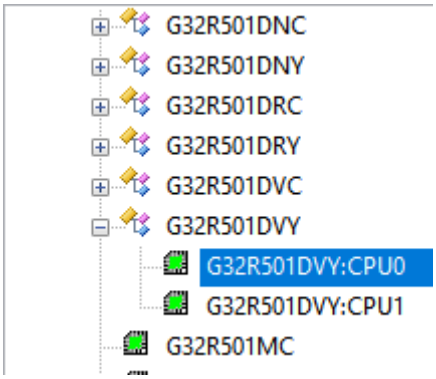
（4） 示例程序可参考 “G32R5xx_SDK\driverlib\g32r501\examples\eval\ipc\”。

4.2.1 新建工程，配置 CPU0 工程的调试设置

1. 打开 MDK-ARM 并新建工程。

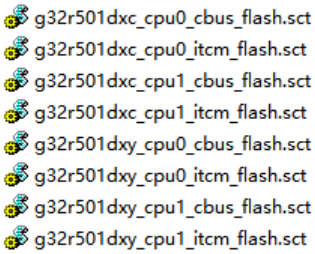
2. 选择正确的设备：Project→Options for Target→Device，选择双核系列的 G32R501（图 3），并选择后面带“CPU0”字样的型号。

图 3 G32R501 MDK 芯片选型（部分）



3. 配置 “.sct” 文件并选择**双核**配置（CPU0 须选用 “g32r501dxy_cpu0_cbus_flash.sct” 或匹配的 cpu0 双核模板，见图 4）。

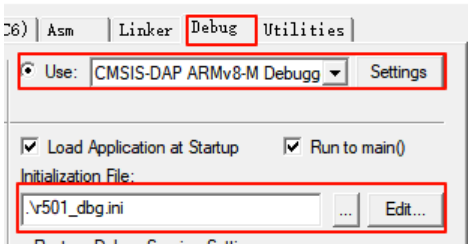
图 4 使用双核配置



4. 配置调试器以及调试脚本：Project→Options for Target→Debug。

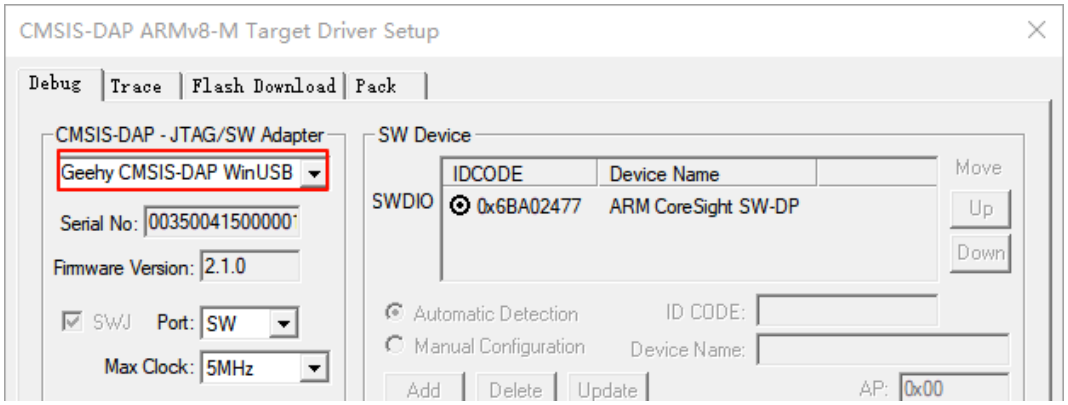
- 选择调试器为“CMSIS-DAP ARMv8-M Debugger”。
- 选择调试脚本为“r501_dbg.ini”。

图 5 配置调试器和调试脚本



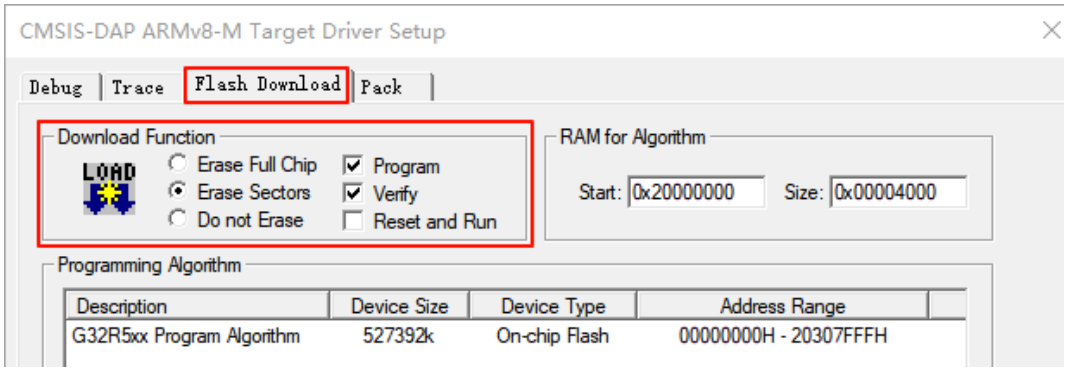
5. 配置调试器为 GEEHY-LINK。

图 6 选择调试器为 GEEHY-LINK



6. 配置程序下载方式。

图 7 CPU0 程序下载配置



7. **CPU0 应用程序必备项（与例程对齐；均为必需）：**

- 实现或移植**“APP_bootCPU1”**（或等价调用 “SysCtl_setCPU1BootAddress” + “SysCtl_enableBootCPU1”）。
- 预留 “cpu1_code” 段，并通过 “.incbin”（或等效链接方式）嵌入 CPU1 映像；详见下文 “使用 CPU0 工程下载 CPU1 的程序文件”。
- 双核调试断点须放在**“APP_bootCPU1(...)” 调用之后**（与 §6.2 Eclipse 一致）。
- 从核启动 API 细节见 **AN1135** 及 IPC 例程源码。

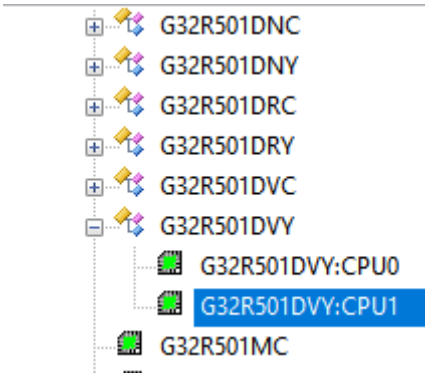
示例（与 “ipc_ex1_mailbox_interrupt” CPU0 源码同模式）：

```
void APP_bootCPU1(uint32_t cpu1Address)
{
    SysCtl_setCPU1BootAddress(cpu1Address);
    SysCtl_enableBootCPU1();
}
/* 在取得 cpu1_imageStartAddr 后： */
APP_bootCPU1(cpu1_imageStartAddr);
```

4.2.2 新建工程，配置 CPU1 工程的调试设置

1. 打开 MDK-ARM 并新建工程。
2. 选择正确的设备：Project→Options for Target→Device，选择双核系列的 G32R501（图 8），并选择后面带“CPU1”字样的型号。

图 8 CPU1 工程选择



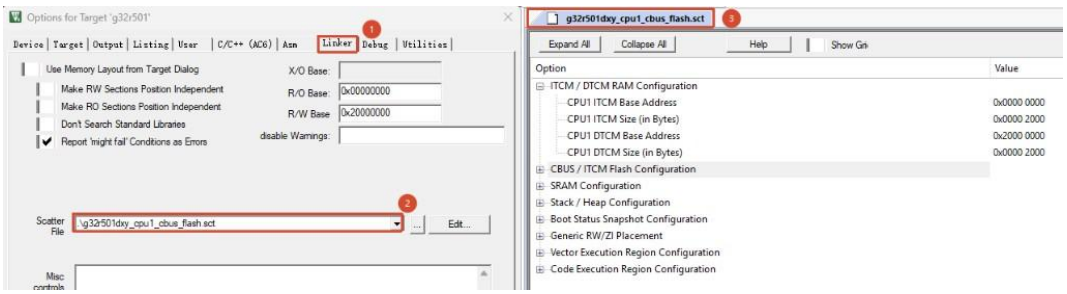
3. 配置“.sct”文件并**选用 CPU1 脚本**，例如“g32r501dxy_cpu1_cbus_flash.sct”（见图 9）。**勿**选用 CPU0 脚本“g32r501dxy_cpu0_cbus_flash.sct”。

CPU0 / CPU1 散列文件对应关系见表格 2。

表格 2 CPU0 / CPU1 散列文件对应

工程	散列文件	说明
CPU0	“g32r501dxy_cpu0_cbus_flash.sct”	双核 Flash 划分及“cpu1_code”段
CPU1	“g32r501dxy_cpu1_cbus_flash.sct”	CPU1 Flash 窗口内 ROM；**禁止**使用 cpu0 脚本

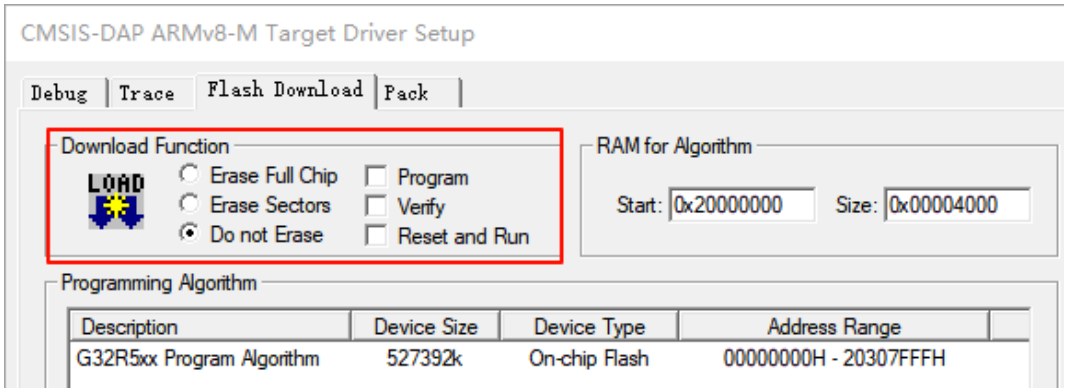
图 9 CPU1 .sct 文件配置



4. 调试器配置请与 CPU0 工程 Debug 页保持一致：选用同一调试器类型（如 CMSIS-DAP / GEEHY-LINK）与 Initialization File；Device 选为 CPU1，并按下文取消对本工程的 Flash 下载。

5. 配置程序无需进行下载。

图 10 CPU1 程序下载配置



4.2.3 使用 CPU0 工程下载 CPU1 的程序文件

由于下载过程中的 Flash 操作均由 CPU0 完成，CPU0 的工程在链接时须包含 CPU1 二进制映像。

须按以下顺序编译：

1. 先编译 **CPU1** 工程以生成 “cpu1_image.bin”。
2. 再编译并下载 **CPU0** 工程（其中已嵌入该 bin）。

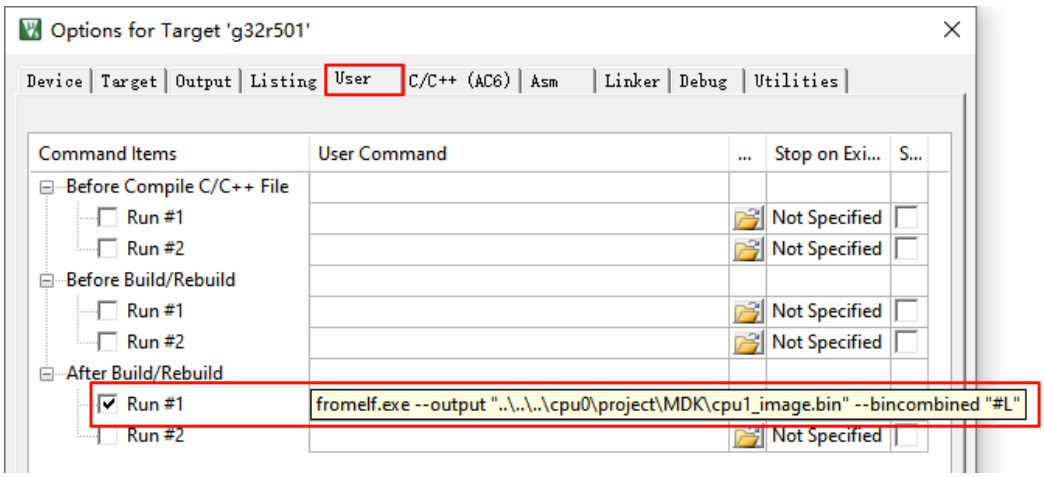
配置步骤如下：

1. 配置 CPU1 工程生成 bin 文件：在 Project→Options for Target→User→After Build/Rebuild 中配置生成 bin 的命令。示例（与现行 “ipc_ex1_mailbox_interrupt” 一致；相对路径随工程深度调整）：

```
fromelf --bin --output "..\..\..\cpu0\source\cpu1_image.bin" "#L"
```

使用命令时，注意工程的路径，须能写入 CPU0 侧 “source/cpu1_image.bin”（图 11）。

图 11 Project→Options for Target→User→After Build/Rebuild



2. 在 CPU0 工程中嵌入对应的 CPU1 bin 文件。示例程序如下（MDK 路径示意；须与本工程 User/incbin 一致）：

```
__attribute__((__used__, section("cpu1_code")))
void G32R501_incbin(void)
{
    __asm(".incbin \"../../source/cpu1_image.bin\"");
}
```

3. 若使用自定义 “.sct” 文件，须在 “.sct” 中指定 “cpu1_code” 段，且该段定义须与 CPU1 运行的程序空间一致。

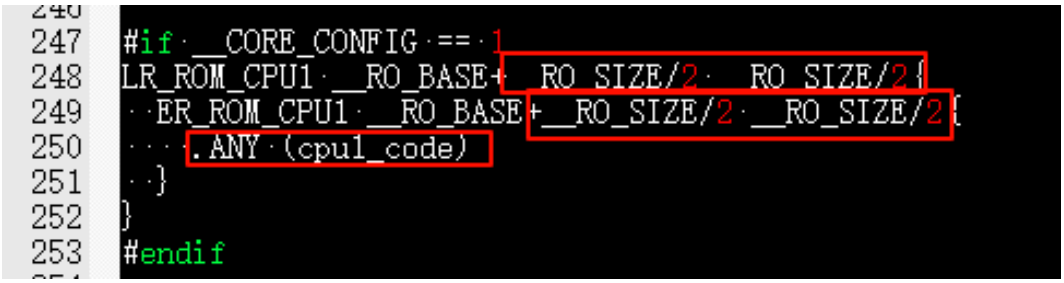
4.2.4 关于示例 .sct 文件

本章节使用的示例 “.sct” 位于 “SDK\device_support\g32r501\common\sct\” 中的 “g32r501dxy_cpu0_cbus_flash.sct” 与 “g32r501dxy_cpu1_cbus_flash.sct”。

说明：本章链接脚本截图对应 SDK “device_support/g32r501/common/sct/” 中的 “g32r501dxy_cpu0_cbus_flash.sct” / “g32r501dxy_cpu1_cbus_flash.sct” 现行文件。若本地界面仍显示旧文件名（如历史上的 “r501_cpu0_flash_link.sct”），请改选上述现行脚本。截图像素若与现行 Configuration Wizard 字段不一致，以 SDK 内脚本内容为准。

1. 在 “g32r501dxy_cpu0_cbus_flash.sct” 中，选择双核配置后，会将 G32R5xx 的 Flash 一分为 2，并声明 “cpu1_code” 段的位置（图 12）。Flash 在双核时的分配情况见表格 3。

图 12 CPU0 中.sct 对 CPU1 程序运行段设置



表格 3 CPU0/1 运行空间设置

Flash 起始地址	大小	使用内核
0x08000000	0x050000	CPU0
0x08050000	0x050000	CPU1

2. 在 “g32r501dxy_cpu1_cbus_flash.sct” 中，关于 Flash 的使用配置与 “g32r501dxy_cpu0_cbus_flash.sct” 的双核设置对应（图 13）。

图 13 CPU1 的.sct 程序运行段设置

```
LR_ROM __RO_BASE+__RO_SIZE/2 __RO_SIZE/2 {
  ER_ROM __RO_BASE+__RO_SIZE/2 __RO_SIZE/2 {
    *.o (RESET, +First)
    *(InRoot$$Sections)
    .ANY (+RO)
    .ANY (+XO)
  }
```

4.2.5 启动双核调试

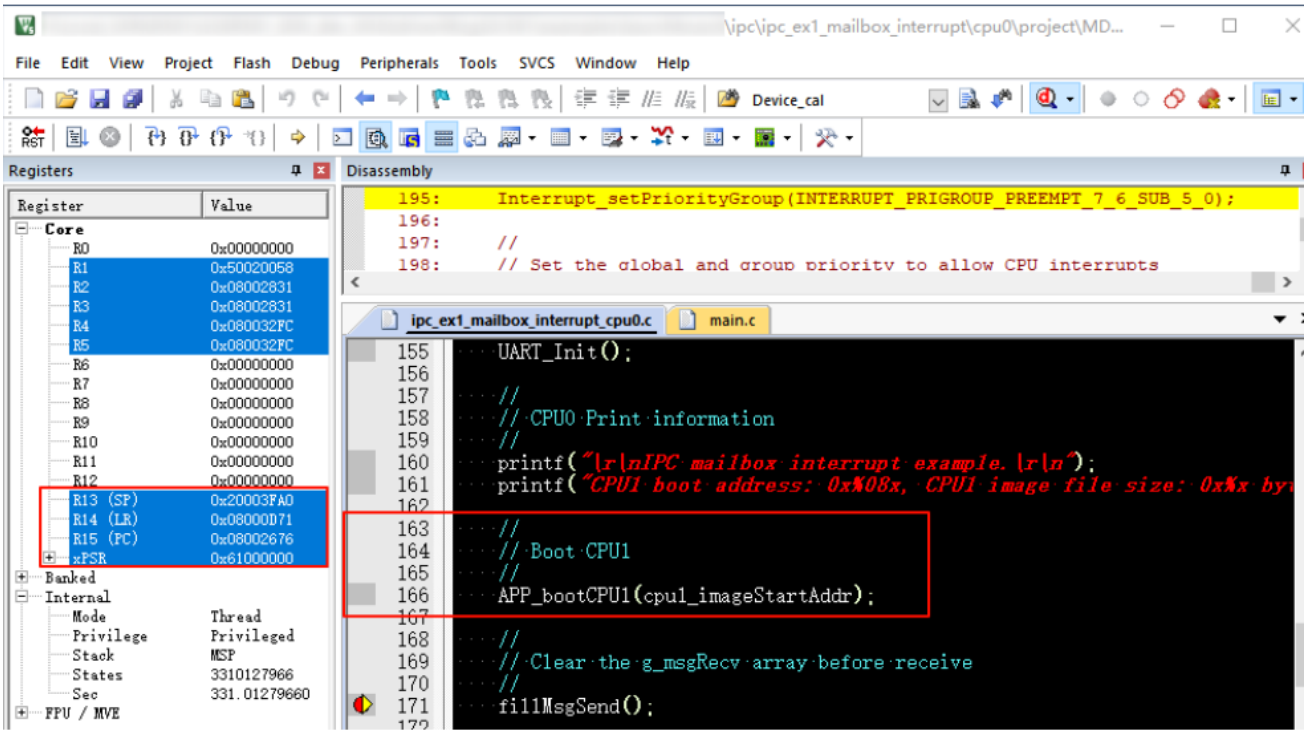
完成正常的调试器配置及程序编译无误后，即可开始双核调试配置。

1. 启动 CPU0 工程调试，并在**“APP_bootCPU1(cpu1_imageStartAddr);”调用之后**设置断点。以“ipc_ex1_mailbox_interrupt”为例：

```
//
// Boot CPU1
//
APP_bootCPU1(cpu1_imageStartAddr);
```

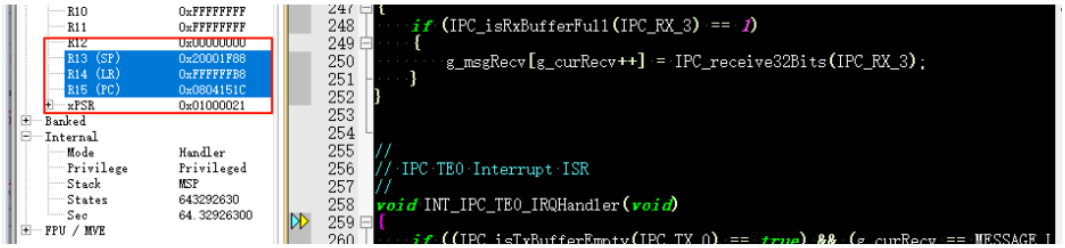
断点位置如图 14 所示。

图 14 CPU0 启动调试并启动 CPU1



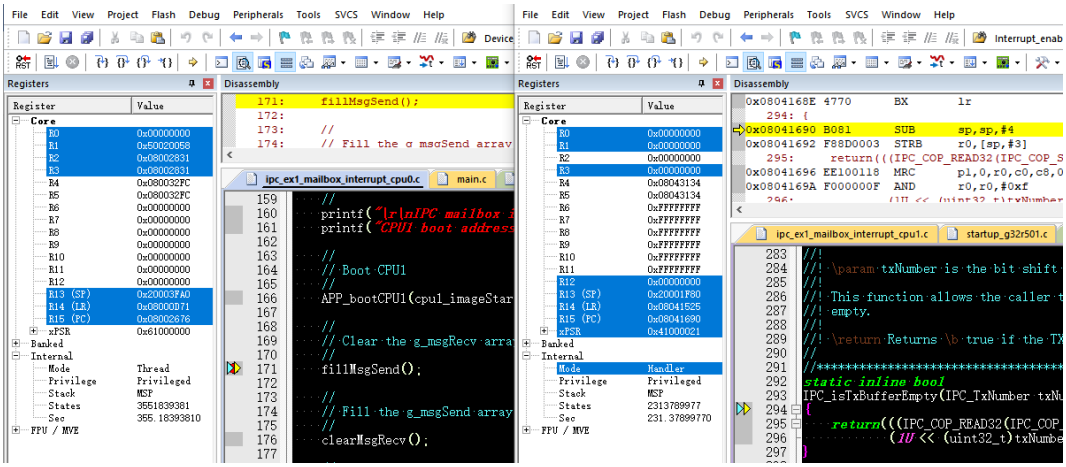
2. 启动 CPU1 工程调试，如图 15 所示。

图 15 CPU1 中断调试示意（IPC Handler）



3. 最终效果如图 16 所示。

图 16 CPU0/1 调试界面



5 IAR Embedded Workbench for Arm 双核调试支持

最新版本的 IAR Embedded Workbench for Arm 可以从 IAR 官方网站下载。

5.1 在 IAR Embedded Workbench for Arm 上进行双核调试

如前文描述，G32R5xx 中的双核非对称系统需要特定的开发工具，在 IAR Embedded Workbench for Arm 9.60.2 以上便支持对 G32R5xx 进行双核调试。

5.2 使用 GEEHY-LINK（WinUSB）进行双核调试的步骤

本节提供了使用 IAR Embedded Workbench for Arm 9.60.2 和 GEEHY-LINK（WinUSB）调试探针与 G32R5xx 微控制器配合工作的分步说明。

注意：

（1） Arm® Cortex®-M52 的双核调试在 IAR Embedded Workbench for Arm 9.60.2 及更高版本中得到支持。

（2） 在进行下面的操作前，请先安装 G32R5xx 芯片支持（**G32R5xx_AddOn_v1.0.0.exe**）。

（3） 在此示例中，需要为每个内核创建一个项目。

（4） 示例程序可参考 “G32R5xx_SDK\driverlib\g32r501\examples\eval\ipc\”。

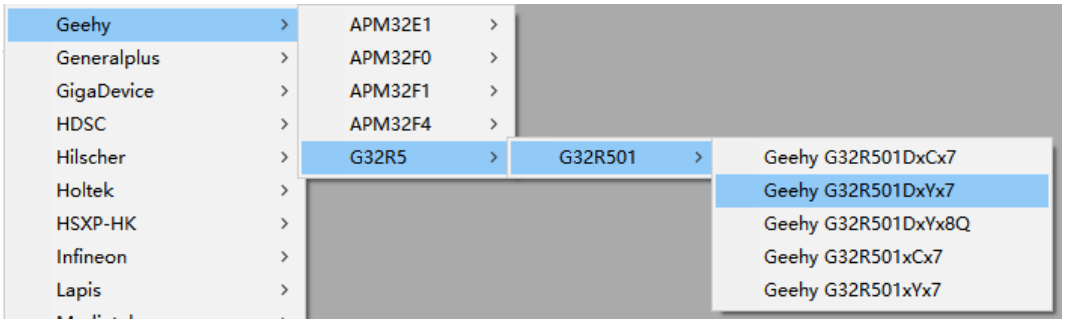
（5） IAR 器件选型不区分 CPU0/CPU1；选择双核器件，并以**“.icf”**区分内核。

5.2.1 配置 CPU0 / CPU1 工程的调试设置

1. 打开 IAR Embedded Workbench for Arm 并新建工程。

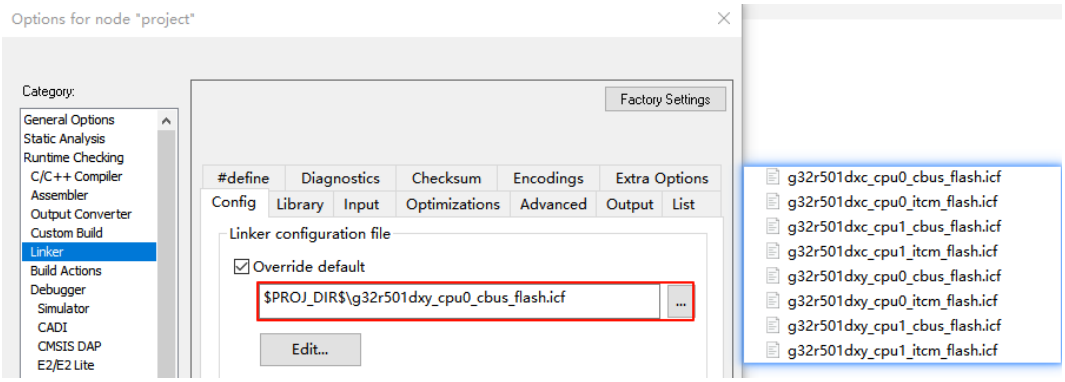
2. 选择正确的设备：General Options→Target→Device，选择双核系列的 G32R501（图 17）。

图 17 G32R501 IAR 芯片选型（部分）



3. 配置 “.icf” 文件，选择双核配置。不同的 CPU 工程请选择不同的 “.icf” 文件。评估板常用料号可选用 “g32r501dxyx7_cpu0_cbus_flash.icf”（CPU0）与 “g32r501dxyx7_cpu1_cbus_flash.icf”（CPU1）；“dxyx8q” 封装对应 “g32r501dxyx8q_cpu0_cbus_flash.icf” / “g32r501dxyx8q_cpu1_cbus_flash.icf”（图 18）。

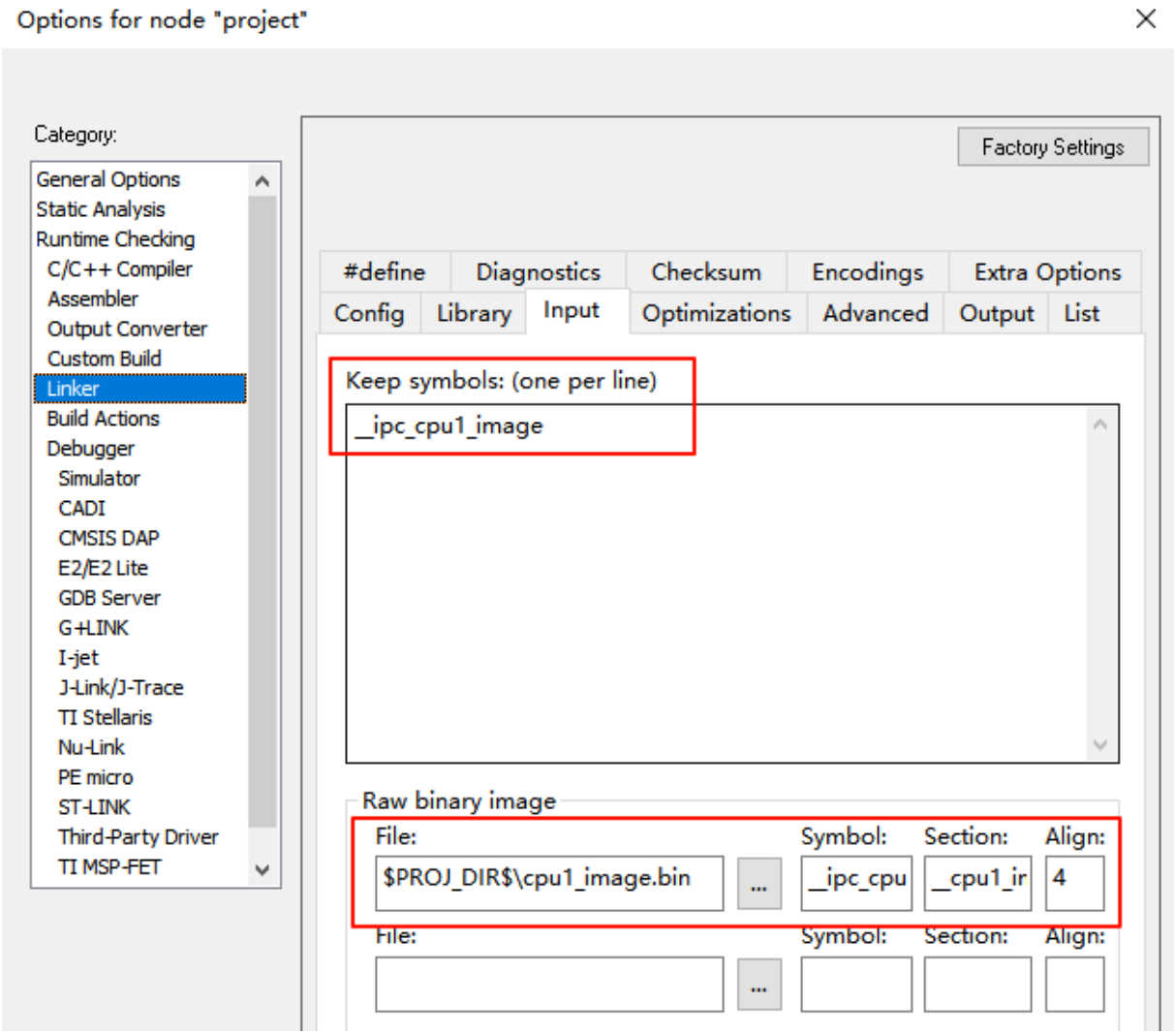
图 18 使用双核配置



4. 配置 CPU0 工程包含 CPU1 运行镜像：Linker→Input。

- 设置保持编译不优化 “ipc_cpu1_image”。
- 加载 bin 文件路径及相关配置（图 19）。

图 19 CPU0 Linker Input 嵌入 cpu1image.bin



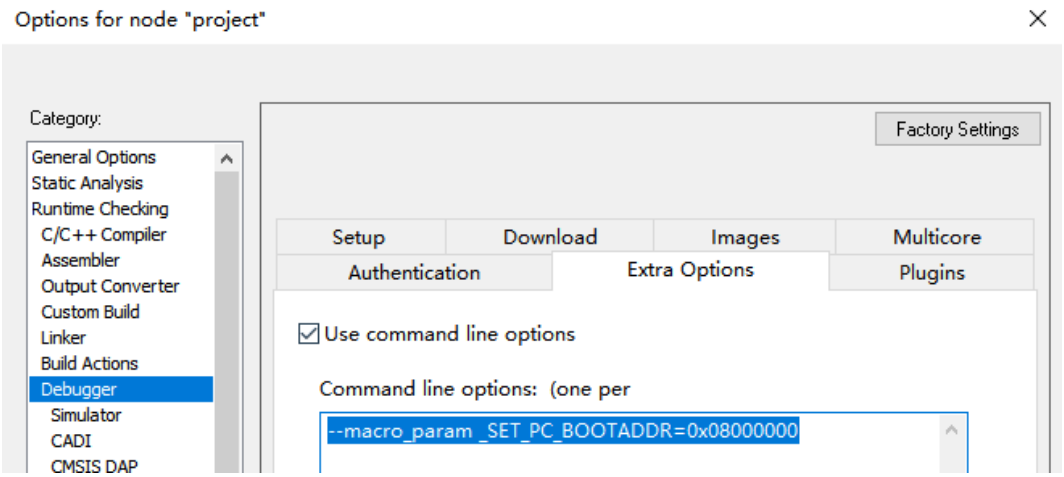
5. 配置调试器及启动地址：

- CPU0/CPU1 工程均需选择调试器为“CMSIS-DAP”：Debugger→Setup。
- CPU0 工程配置启动地址：Debugger→Extra Options→勾选“Use command line options”→文本框中输入：

“--macro_param _SET_PC_BOOTADDR=0x08000000”

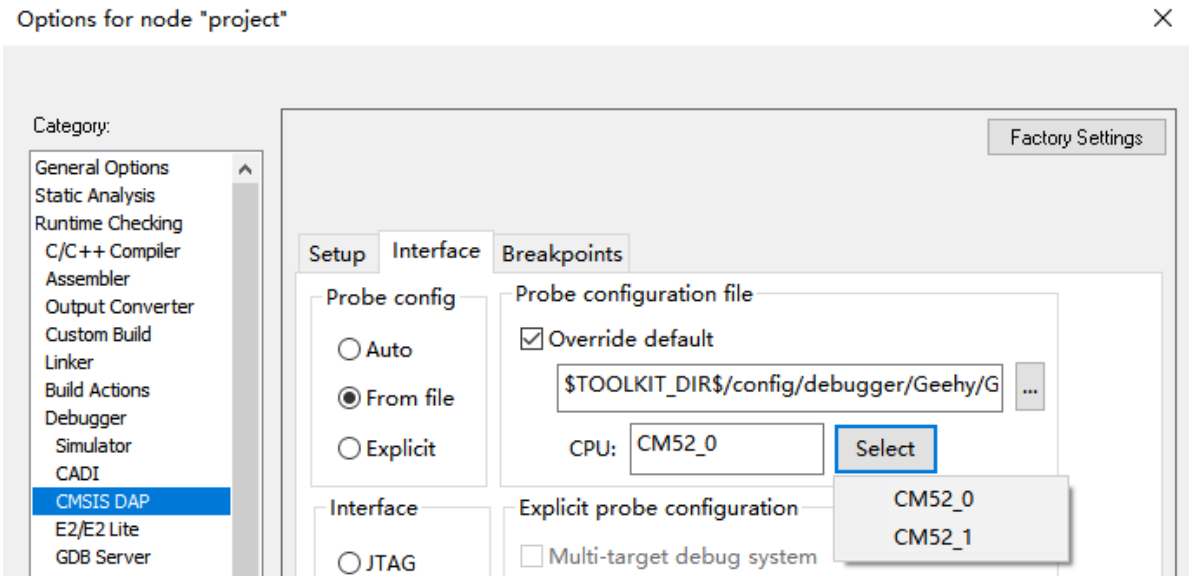
（地址须与 CPU0 实际启动地址相对应，见图 20）。

图 20 配置 CPU0 工程启动地址



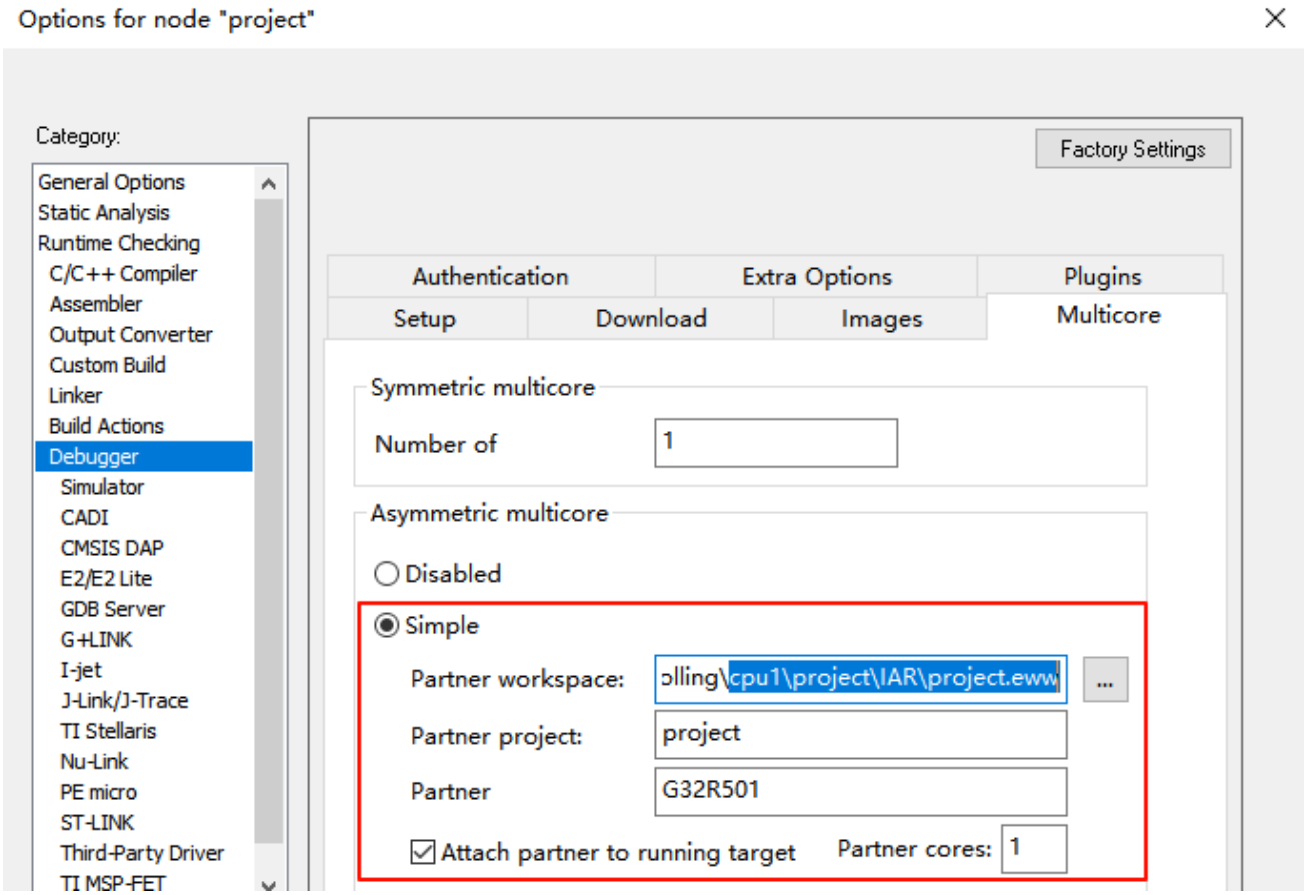
1. 配置 CPU0 工程 DebugAP 端口：CMSIS DAP→Interface→Probe config 选择“From file”→根据 CPU 选择“CM52_0”（图 21）。

图 21 配置 CPU0/CPU1 工程 DebugAP 端口



2. 配置 CPU0 工程链接 CPU1 工程：在各自工程完成基础配置后，使用 CPU0 链接 CPU1，以便启动 Debug 时可同时开启两个工程。配置过程：Debugger→Multicore→Asymmetric multicore 勾选“Simple”→Partner workspace 处选择 CPU1 工程文件→Partner project 处填写 CPU1 工程名称→Partner 处填写工程 Tag（图 22）。

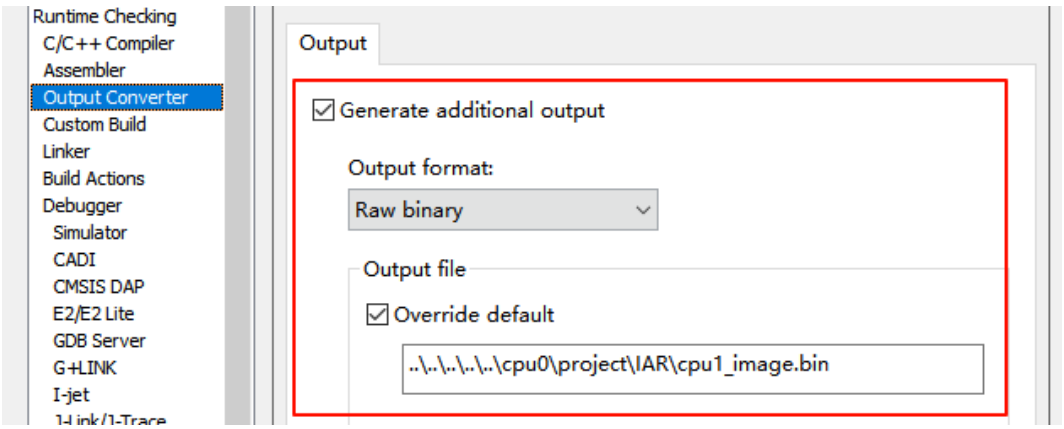
图 22 配置 CPU0 工程链接 CPU1 工程



5.2.2 CPU1 工程的额外设置

1. 参考前文完成正确的芯片选型、“.icf”文件设置以及 Debug 设置。
2. 配置输出 bin 文件至指定目录（须与 CPU0 包含 CPU1 运行镜像的路径一致，见图 23）。

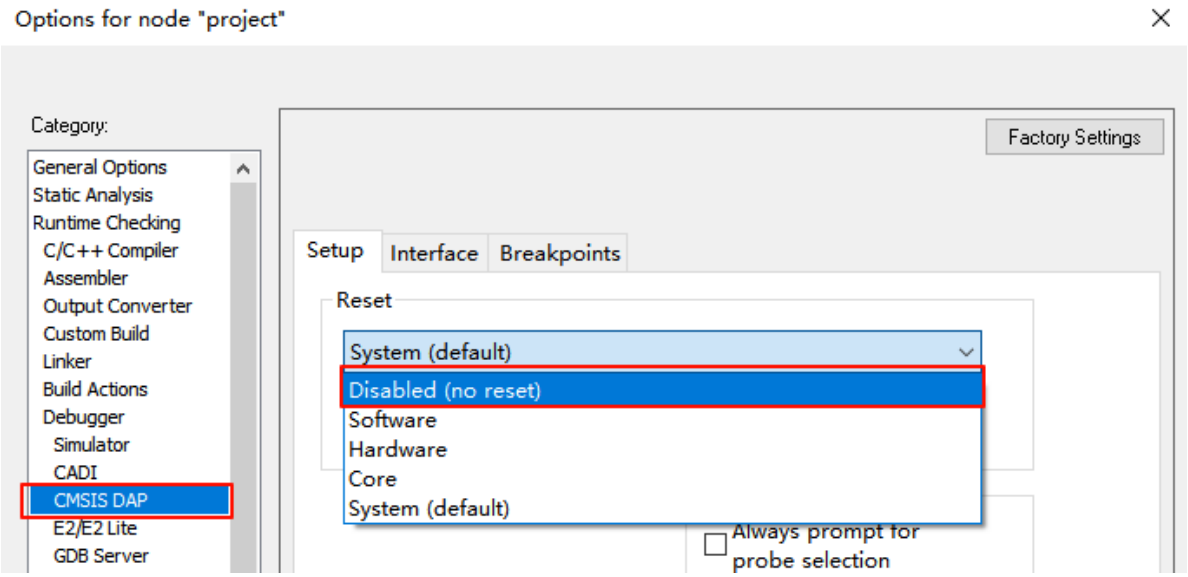
图 23 配置 CPU1 工程输出 bin 文件至指定目录



3. 配置 CPU1 工程 DebugAP 端口：CMSIS DAP→Interface→Probe config 选择“From file”→根据 CPU 选择“CM52_1”。

4. 配置 CPU1 在 Probe 连接时不复位 MCU：CMSIS DAP→Setup→Disabled（no reset）（图 24）。

图 24 配置 CPU1 连接时不复位 MCU

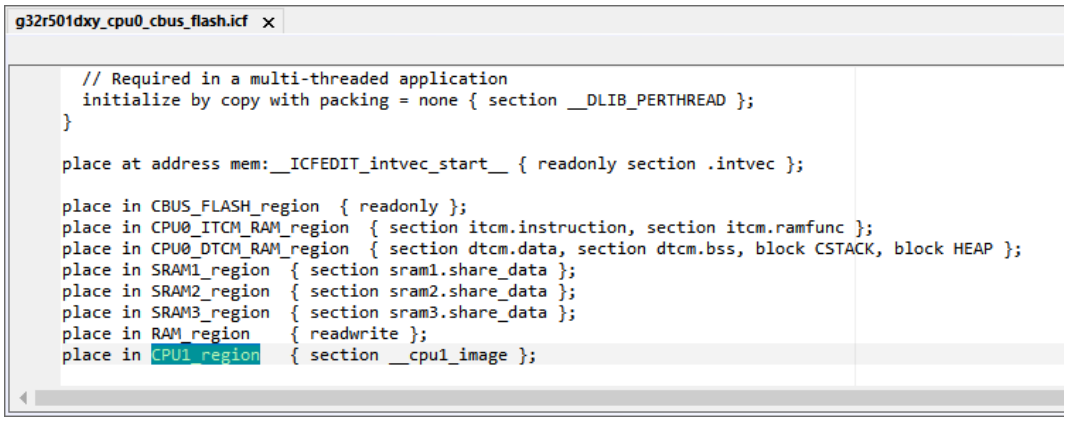


5.2.3 关于示例 .icf 文件

本章节使用的示例 “.icf” 位于 “SDK\device_support\g32r501\common\icf\”，例如 “g32r501dxyx7_cpu0_cbus_flash.icf” 与 “g32r501dxyx7_cpu1_cbus_flash.icf”（“dxyx8q” 同理换文件名）。

在 “g32r501dxyx7_cpu0_cbus_flash.icf” 中，选择双核配置后，会将 G32R501 的 Flash 一分为 2，并声明 “cpu1_image” 段的位置（图 25）。Flash 在双核时的分配情况见表格 3。

图 25 CPU0 中.icf 对 CPU1 程序运行段设置

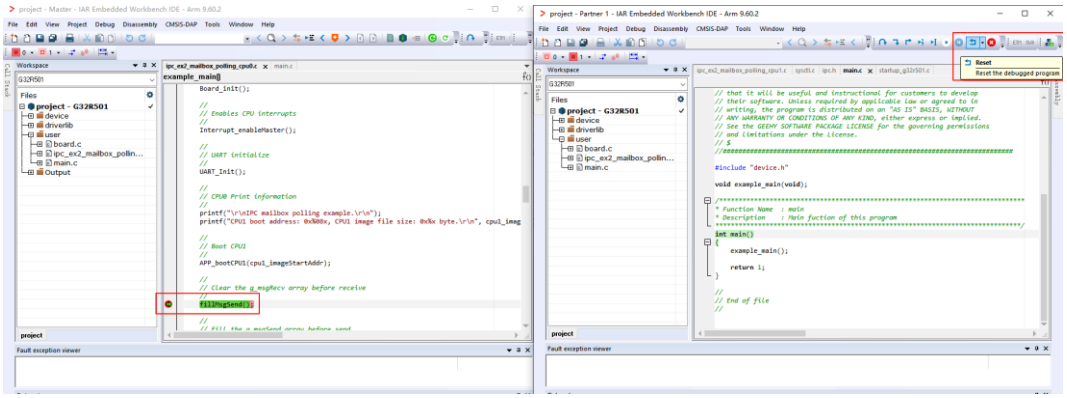


5.2.4 启动双核调试

完成调试器配置及程序编译无误后，即可开始双核调试：

1. 启动 CPU0 工程调试，会直接启动两个 CPU；启动调试窗口后可看到两个调试窗口。
2. 在 CPU0 工程中，在启动 CPU1 的程序（“APP_bootCPU1(...)”调用）之后打断点，并使程序运行至此断点；此时 CPU1 工程可正常连接 CPU1。
3. CPU1 工程正常连接后，请按下其工程的“Reset”，使 CPU1 回到起始地址。
4. 后续便可根据需要进行双核 Debug（图 26）。

图 26 IAR 双核 Debug 调试界面



6 Eclipse 双核调试支持

最新版本的 Eclipse 可以从 Eclipse 官方网站下载。

注意：本章的调试方式是使用 pyocd+GEEHY-LINK 进行调试。

6.1 在 Eclipse 上进行双核调试

请按 AN1126 第 7 章完成对 G32R501 系列芯片的 pyOCD 支持。

6.2 使用 GEEHY-LINK（WinUSB）进行双核调试的步骤

本节提供了使用 Eclipse 和 GEEHY-LINK（WinUSB）与 G32R5xx 微控制器配合工作的分步说明。

在此示例中，需要为每个内核创建一个项目。

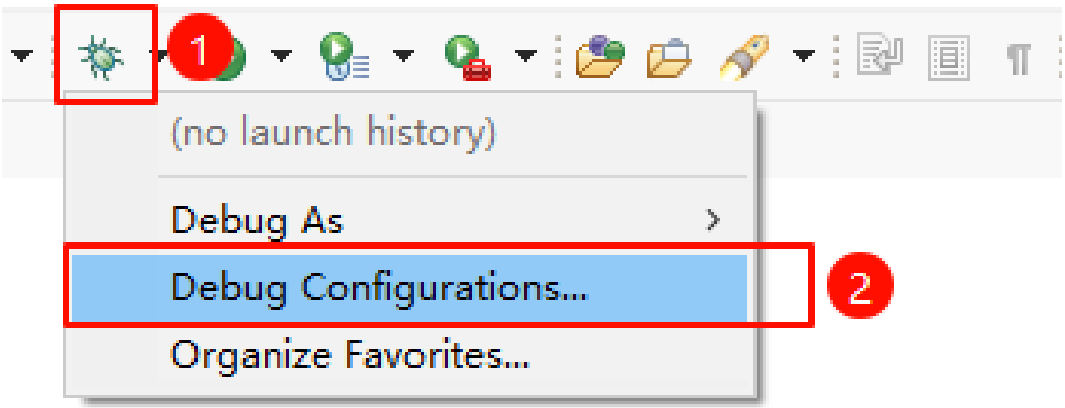
（示例程序可参考 G32R5xx_SDK\driverlib\g32r501\examples\eval\ipc\）

导入工程，确保编译无误后请按照如下步骤配置 Debug 选项卡。

6.2.1 新建 Debug 配置

1. 在 Debug 图标处左键，以显示 Debug 配置。
2. 选择显示出来的“Debug Configurations...”。
3. 在新窗口选择“GDB pyOCD Debugging”。
4. 选择“New Configuration”以新建 Debug 配置（图 27）。

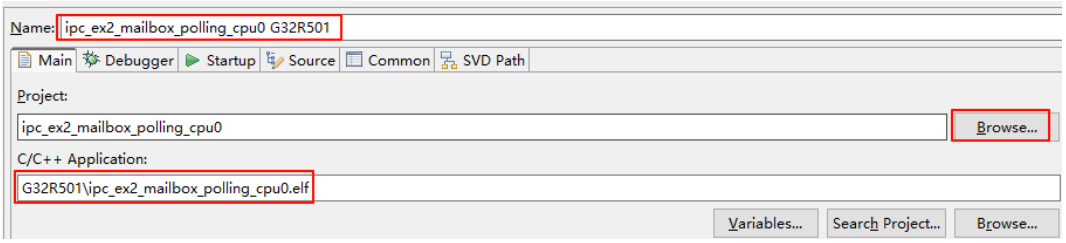
图 27 New Configuration



6.2.2 配置 Main 选项卡

1. 在最上端命名当前 Debug 配置名称。
2. 选择“Browse...”，选择当前 Debug 配置对应的工程。
3. 选择对应的 Debug ELF 文件，例如：“G32R501\ipc_ex2_mailbox_polling_cpu0.elf”。示例使用相对于工程文件的相对路径，亦可使用绝对路径（图 28）。

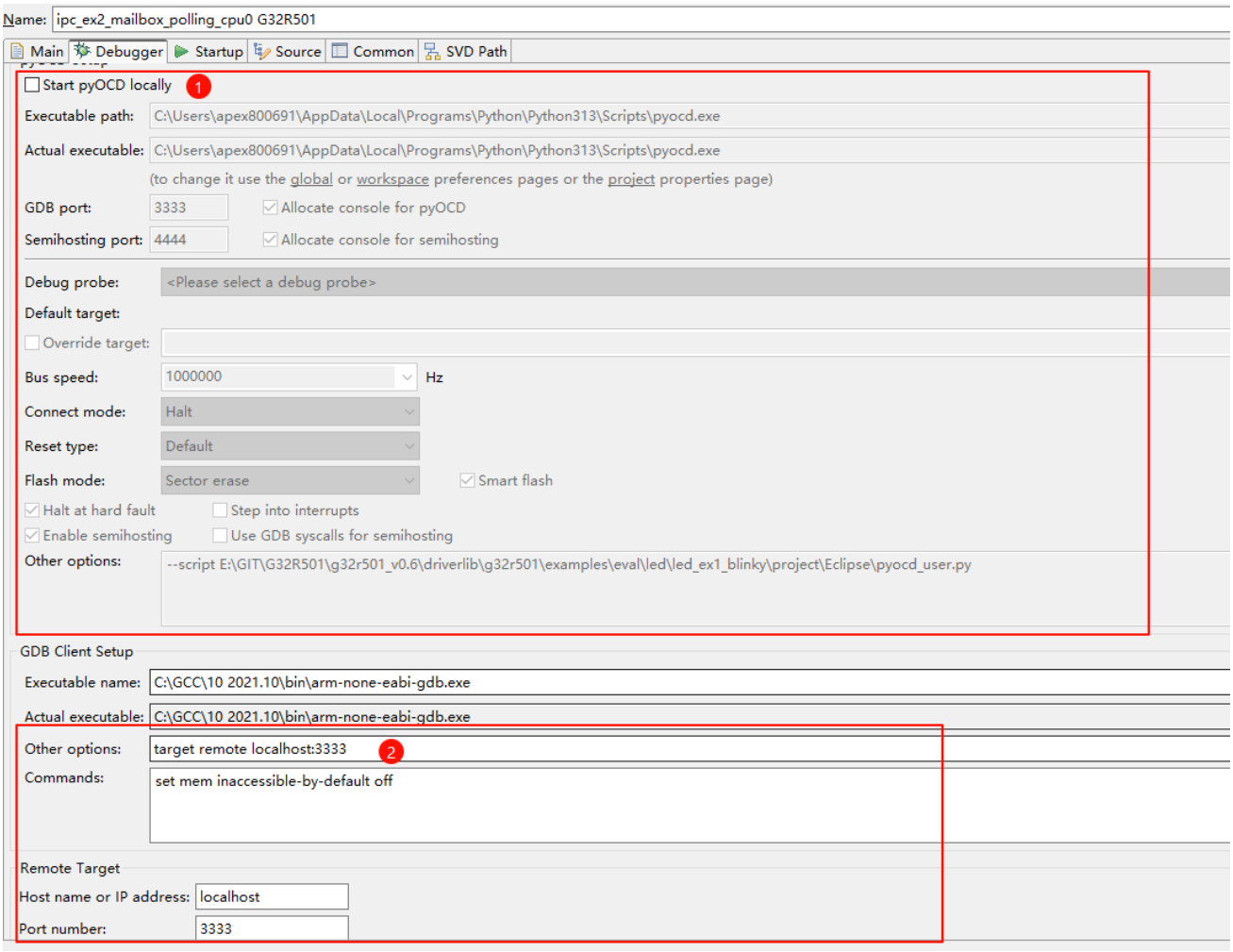
图 28 配置 Main



6.2.3 配置 Debugger 选项卡

1. 去除由 Eclipse 启动 pyOCD GDB Server 的设置。
2. 设置 GDB Client 连接至相应内核端口。一般默认端口为 core 0：3333、core 1：3334（图 29）。

图 29 双核 DebugDebugger 选项卡



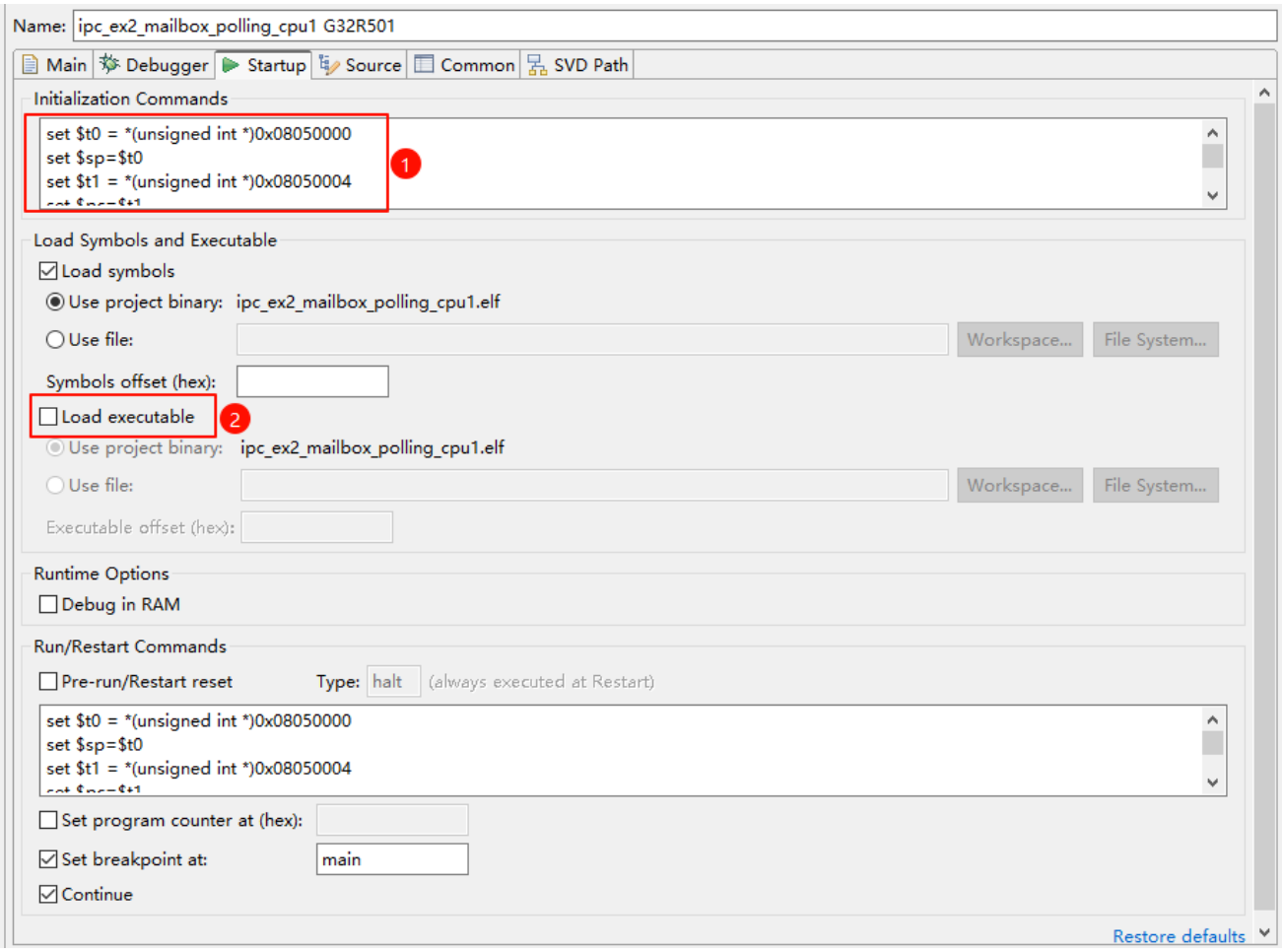
6.2.4 配置 Startup 选项卡

- cpu0：CPU0 Startup 选项卡参考单核配置，详见 AN1126 中单核 Debug 配置相关章节。
- cpu1：
  - Commands 选项卡去除解密序列，仅保留 SP/PC 设置序列。（注意：示例向量表基址为 “0x08050000”，须与本工程链接布局一致；基址是向量表起始地址，SP/PC 须读取该地址处内存内容，不可把基址本身赋给寄存器。）

```
set $sp = *(unsigned int*)0x08050000
set $pc = *(unsigned int*)0x08050004
set $xpsr = $xpsr | (1<<24)
```

  - 取消“Load executable”勾选（图 30）。

图 30 Startup 选项卡



6.2.5 启动 pyOCD gdbserver 与双核调试

从终端启动 pyOCD gdbserver，命令为：

```
pyocd gdbserver
```

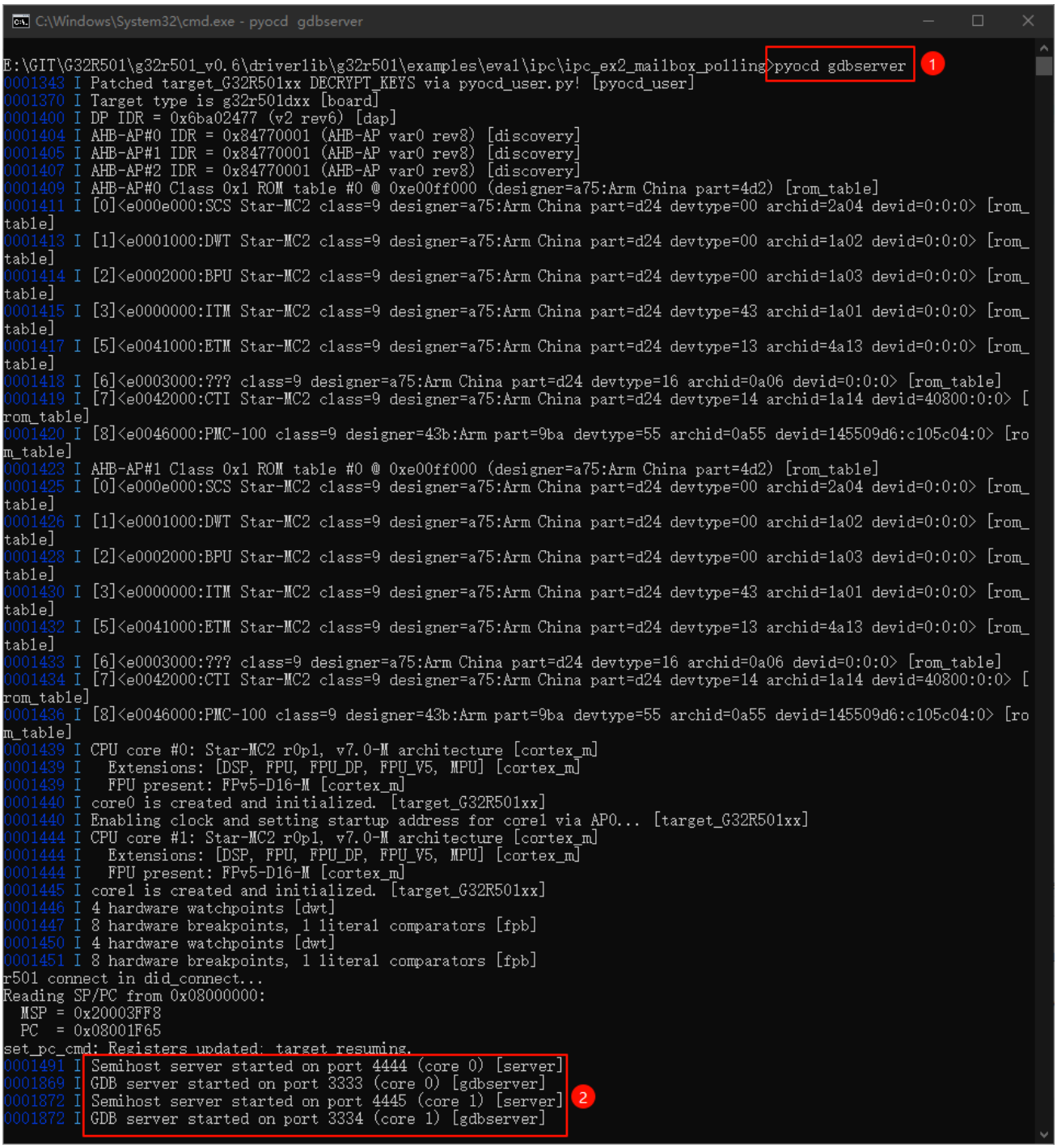
终端启动 pyOCD gdbserver 的路径下应当包含以下文件（参考：“SDK/device_support/g32r501/common/pyOCD/”）：

- “pyocd.yaml”（双核版本），例如：

```
target_override: g32r501dxx
frequency: 8000000  # Set 8 MHz SWD default for all probes
session:
  enable_multicore_debug: true
  persist: true
```

- “pyocd_user.py”（图 31）。

图 31 终端启动 pyocd gdbserver



在 Eclipse 分别启动两个 Debug 服务：

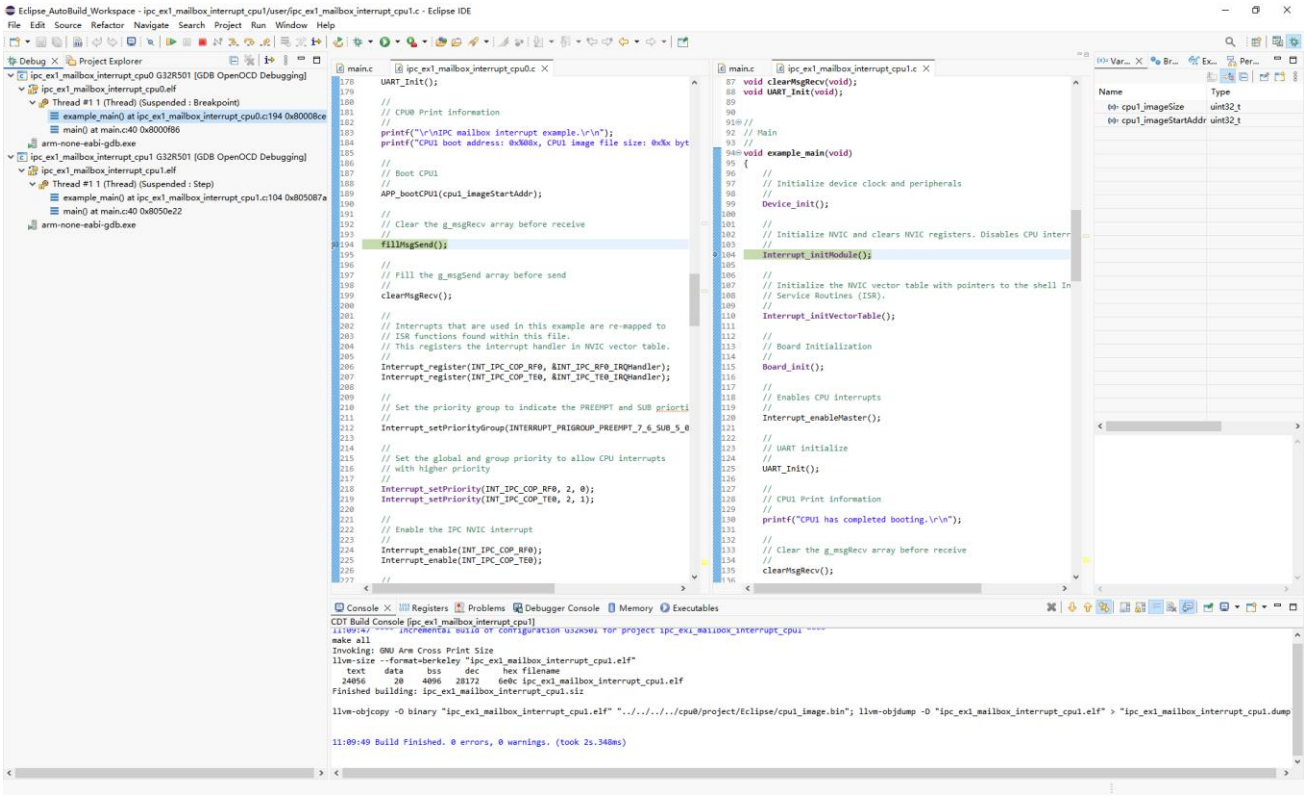
1. 启动 CPU0 工程调试，并在设置 CPU1 启动的语句后打断点。在示例中是在“APP_bootCPU1(cpu1_imageStartAddr);”语句后打断点。

2. 启动 CPU1 工程调试。

3. 根据需要进行双核 Debug：

- 点击 cpu0 对应的线程控制 cpu0 的 Debug。
- 点击 cpu1 对应的线程控制 cpu1 的 Debug（图 32）。

图 32 Eclipse 双核 Debug 调试界面



7 版本历史

表格 4 文件版本历史

日期	版本	变更历史
2025.1	1.0	● 新建
2025.4	1.1	● 新增 Eclipse 双核调试支持
2026.8	1.2	● 适用产品改为 G32R501Dx 系列双核产品 ● 链接脚本文件名对齐现行 g32r501dxy_ ● IAR 示例改为 g32r501dxyx7_/dxyx8q_.icf ● 修正 r501_dbg.ini 笔误 ● 按现行双核调试步骤更新相关截图 ● 修正 Eclipse cpu1 Startup 中 SP/PC 须按向量表解引用 ● 修正文档格式。

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，也不应被视为极海对第三方产品、服务或知识产权提供任何形式的保证，包括但不限于任何第三方知识产权的非侵权保证，除非极海在销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为

准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及/或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

极海产品并非设计、授权或担保适合用于军事、生命保障系统、污染控制或有害物质管理系统中的关键部件，亦非设计、授权或担保适合用于在产品失效或故障时可导致人员受伤、死亡、财产或环境损害的应用。

如果产品未标明“汽车级”，则表示不适用于汽车应用。如果用户对产品的应用超出极海提供的规格、应用领域、规范，极海不承担任何责任。

用户应该确保对产品的应用符合相应标准以及功能安全、信息安全、环境标准等要求。用户对极海产品的选择和使用负全部的责任。对于用户后续在针对极海产品进行设计、使用的过程中

所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册及产品的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册及产品而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失），这涵盖了可能导致的人身安全、财产或环境损害等情况，对于这些损害极海概不承担责任。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。