Beken JTAG/SWD Debug Handbook

BEKEN WIFI GROUP

个人用户：baili_beken@sina.com
博通集成电路上海有限公司 | 张东路 1387 号

目录

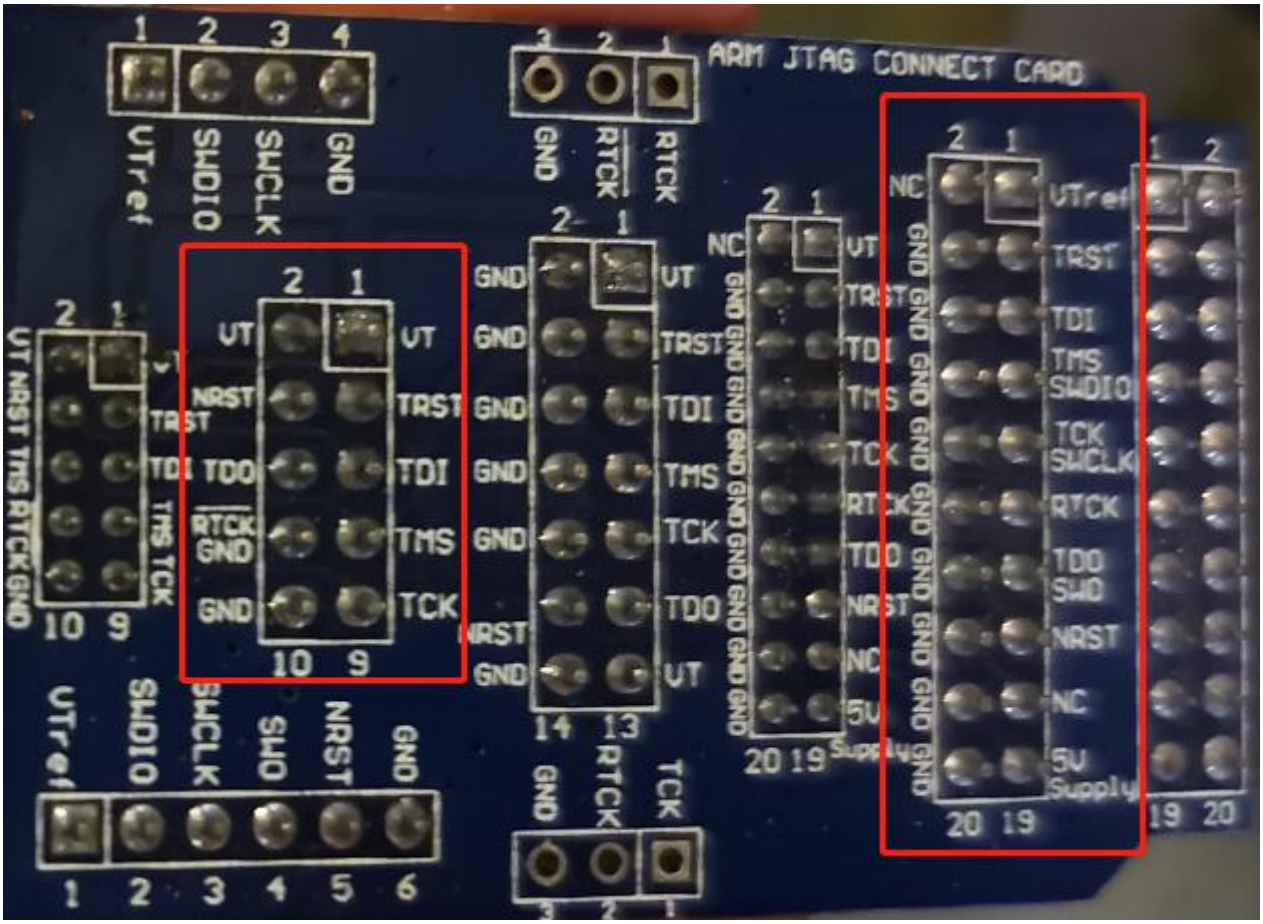
JTAG 原理 6
JTAG 接口 6
Arm 20-pin Connector 7
SWD 接口 7
IAR + STM32U585 + GNU Tools ARM Embedded + ELF + ST-LINK 9
安装 IAR 9
安装 STM32 工具 9
调试配置 10
选项 10
映像文件替换 12
只加载符号表的调试 12
指定符号表路径 13
正常调试 13
TRACE32 + ARM9 15
前序 15
Personal Computer 15
Trace32 软件 16
Trace32 硬件 16
EVB 固件 17
EVB 硬件 17
安装 17
调试 17
连接 17
启动 19
脚本 19
代码显示窗口 22
单步执行、运行、暂停 24
设置断点 25
观察变量 27
观察内存 28
查看调用栈 28
Freertos 特性 29
性能分析 35
断点进阶设置 38
常见问题 58
Debug port fail 58
Fatal error #FF, contact technical support (support@lauterbach.com) 58
TRACE32 device already used by other GUI 59
IAR + ARM9 + GNU Tools ARM Embedded + ELF + JLINK 59
IAR 简介 59
IAR 安装以及破解 60
台式机配置 60
IAR 版本 61
安装 61
破解 64
JLINK 驱动安装以及破解 67
Jlink 版本信息 67
安装 67
破解 67
驱动信息 68
调试搭建 69
建立一个基于 Arm968ejs 的 IAR 简易工程 69
ELF 文件替换 71
调试 72
Debug 集锦 73
Attach to Running Target 73

ARM9 cpu 速率疑问 74
备注 75
DWARF 75
KEIL + ARM9 + GNU Tools ARM Embedded + ELF + JLINK 79
安装以及破解 79
调试 79
Eclipse + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB 79
bk72xx-eclipse-compileAndDebug guide 79
JDK--java development kit 80
Eclipse IDE 81
llg plug-in 82
GNU ARM Embedded Toolchain 83
J-link driver install 84
Compile setting 86
J-link debugging setting 87
Micoder + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB + OPENOCD 92
bk72xx-Micoder-compileAndDebug guide 92
JRE 92
MiCoder IDE 93
J-Link 94
Compiler Setting 95
Debugging Setting 95
Debug without Downloading 98
Alios 使用 Micoder + JTAG 调试的补充说明 99
使用 MiCoder Jtag 驱动安装方法 100
配置细节 101
bk72xx-gdb-command.txt 108
bk72xx.cfg 109
jlink.cfg 111
comments 111
references 112
http://gnuarmeclipse.sourceforge.net/ 112
https://eclipse.org 112
UM08001_JLink.pdf 112
How to install the GNU MCU Eclipse plug-ins? 112
Micoder + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB 130
更新 JLink 调试插件 130
Attach 调试功能配置 131
Debug without Downloading 调试配置 133
Debug with Downloading 调试配置 135
备注 136
Device Name 136
TRACE32 + BA22 138
安装 138
软件安装 138
目录地址：\\192.168.0.30\日常软件\软件开发\TRACE32_201302 138
安装步骤 138
补丁安装 143
补丁 1 143
补丁 2 143
使用 145
Attention 1---Erase flash 145
Attention 2---sleep and jtag pins 145
Attention 3---flto 145
Attention 4---help document 146
Attention 5---cmm script 146
Attention 6---start 147
Attention 7---change SR's value 148

Attention 8---write breakpoint at SPR 148
脚本 149
Attention 1---reset duration 154
Attention 2---bus busy issue 154
BACKUP 1---little endian 154
BACKUP 2---jtag protocol 154
BACKUP 3---memory access module 154
沟通纪要 155
现存问题 157
有时候第一次上电，运行脚本无法下载 Image 成功 157
正常运行 CMM 脚本，偶现擦除 FLASH 完成之后 Bus Error Generated By CPU 157
调试方法集 157
第一次使用，如何开始？ 157
点击菜单 BA22 里面子菜单项（如下图），怎么没有实现？看不到寄存器的值？ 157
为什么 LA3500 Trace32 停下来，发现有 undef 指令？ 158
为什么运行起来不能打断点？ 158
Trace32 SOC 运行起来，想找某个函数符号表，然后设置断点，通过菜单项 Var—>Show Function，出现异常 access timeout，target running 158
为什么 BK3260 不能使用 Beyond Trace32？ 158
想知道某个全局变量在哪里被改写了？在哪里第 n 次被改写？ 159
Break 之后单步调试，timer 中断太多，怎么关掉中断？ 159
SOC 已经运行一段时间，可能出错，也可能正常，怎样将 TRACE32 attach 上去看看现场？ 159
下载过 elf 文件，如何在运行之后 dump 出 Flash 里面的内容？ 160
如何设置条件断点 160
浅析异常现场 160
Alignment Exception 异常之后会进入异常处理例程，LA3500 能在出错的地方停下来吗？ 162
技术支持 162
德国 162
中国 162
私有文档声明 162

JTAG 原理

JTAG 接口



Arm 20-pin Connector

VREF	p	1	2	nc	-
TRST_N	i	3	4	p	GND
TDI	i	5	6	p	GND
TMS	i	7	8	p	GND
TCK	i	9	10	p	GND
-	nc	11	12	p	GND
TDO	o	13	14	p	GND
SRST_N	od	15	16	p	GND
-	nc	17	18	p	GND
-	nc	19	20	p	GND

Table 8: ARM 20-pin connector (target board

SWD 接口

- Serial Wire Debug Port
- GND，SWDIO，SWCLK

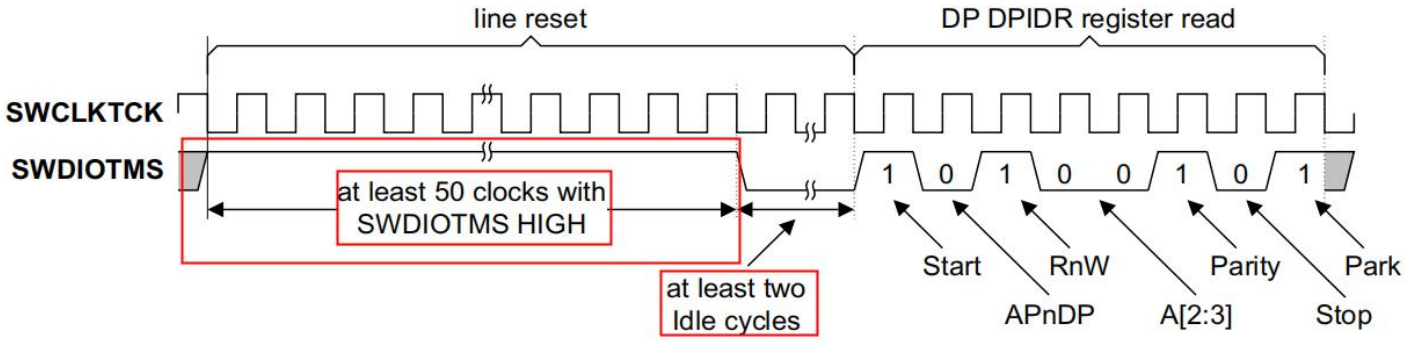


Figure B4-8 Line reset sequence followed by a DP DPIDR register read

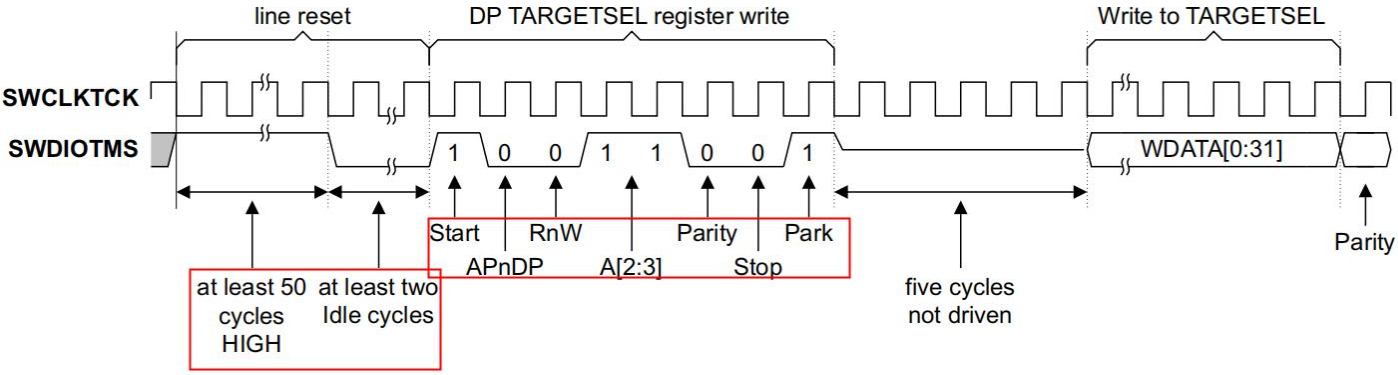


Figure B4-9 Line reset sequence followed by a DP TARGETSEL write

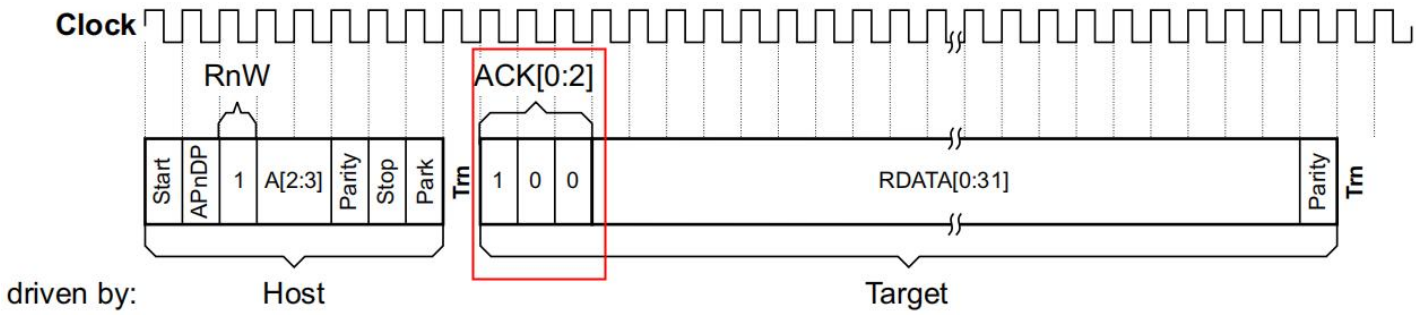


Figure B4-2 SWD successful read operation

IAR + STM32U585 + GNU Tools ARM Embedded + ELF + ST-LINK

安装 IAR

软件下载路径：

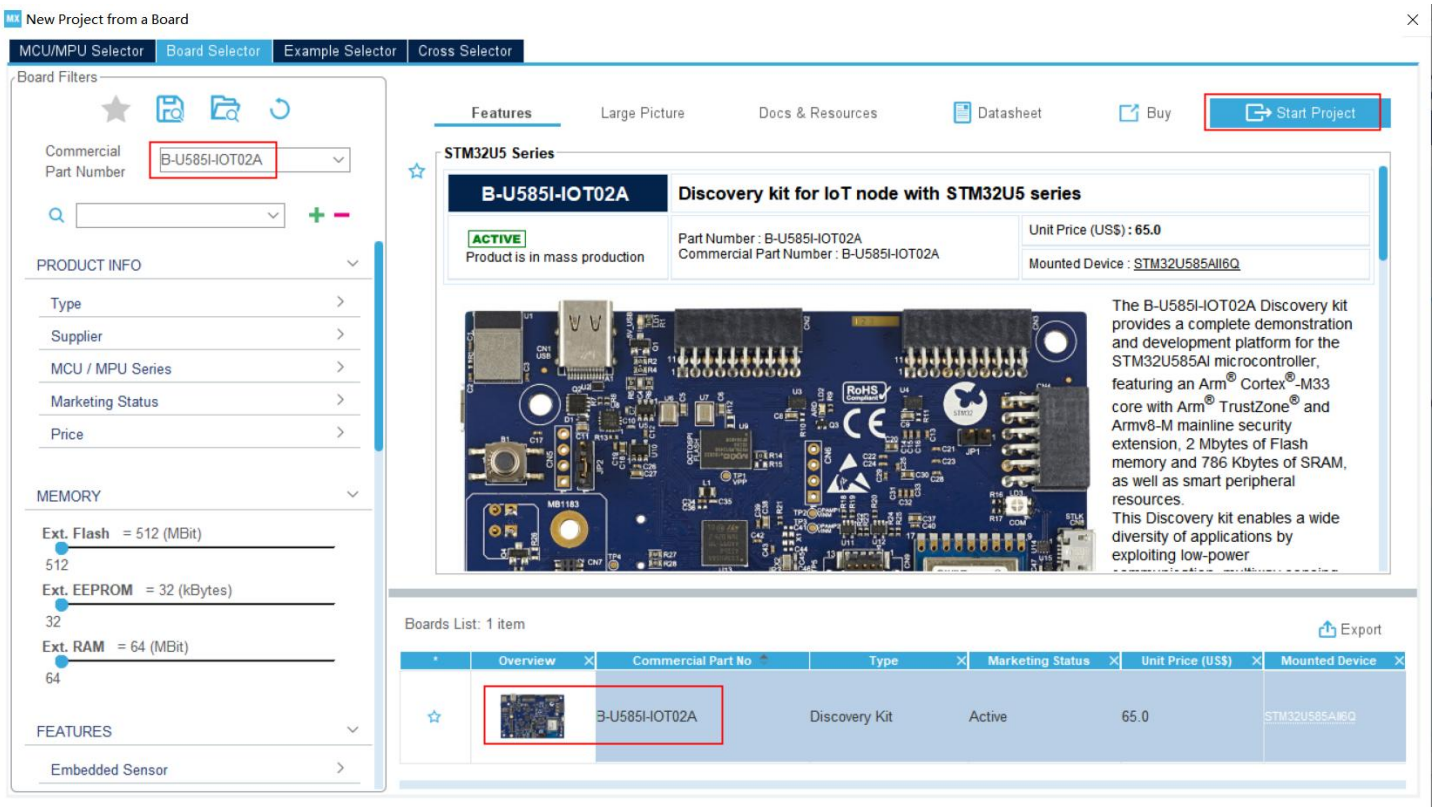
- [IAR Embedded Workbench for Arm 9.30 版本] https://pan.baidu.com/s/1urZO2Q2E3Kb7MdWscbp8rA?pwd=ivvr 提取码：ivvr
- \\192.168.0.30\exchange\wangzhilei\tools\IAR Embedded Workbench for Arm version 9.30.1

步骤：

- 安装 EWARM-9301-50054.exe
- 打补丁.\Crack\Patch\licpatcher64a.exe，注意阅读同级目录的 **readme**
- 破解.\Crack\Keygen\License Generator.exe，注意阅读同级目录的 **readme**

安装 STM32 工具

1. 安装 STM32CubeMX，用于生成 STM32U585 的 demo **工程**；



2. 安装 STM32CubeProgrammer，用于**下载** STM32U585 的映像文件；

笔者调试的 STM32u585 之 TF-M(platform\ext\target\stm**b_u585i_iot02a**)，下载的时候主要安装如下顺序运行 shell 脚本：

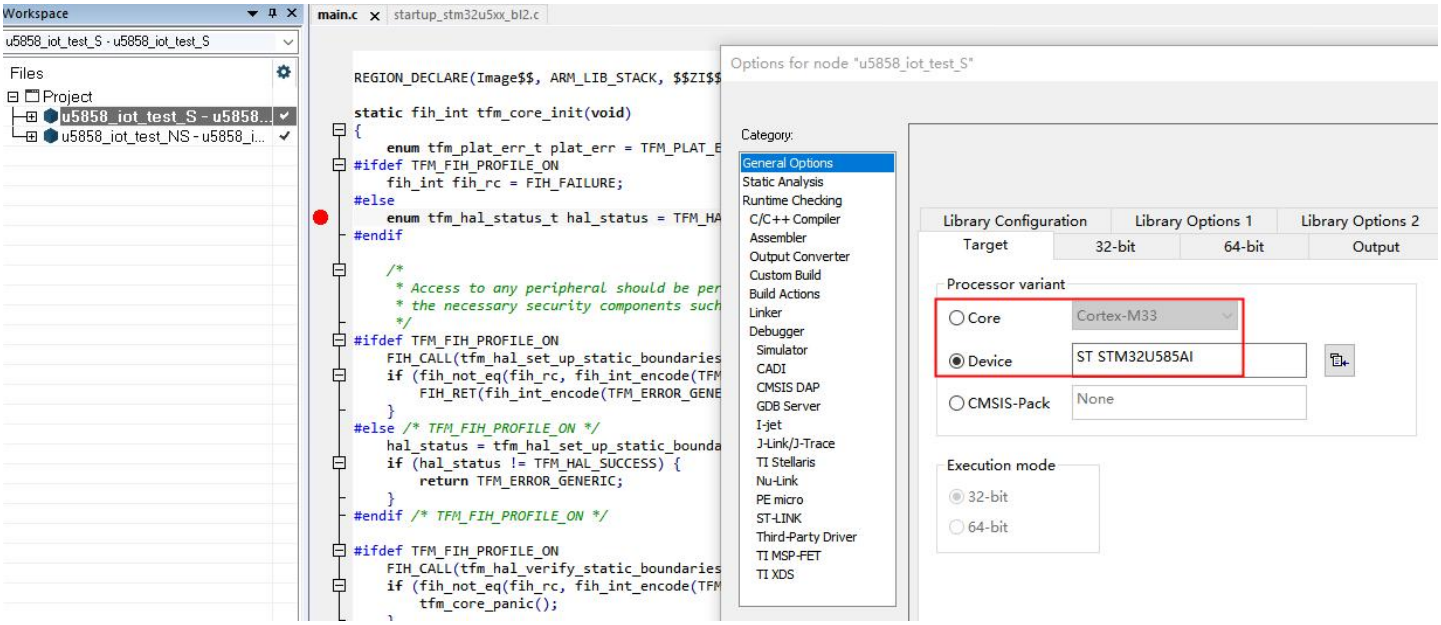
1. postbuild.sh

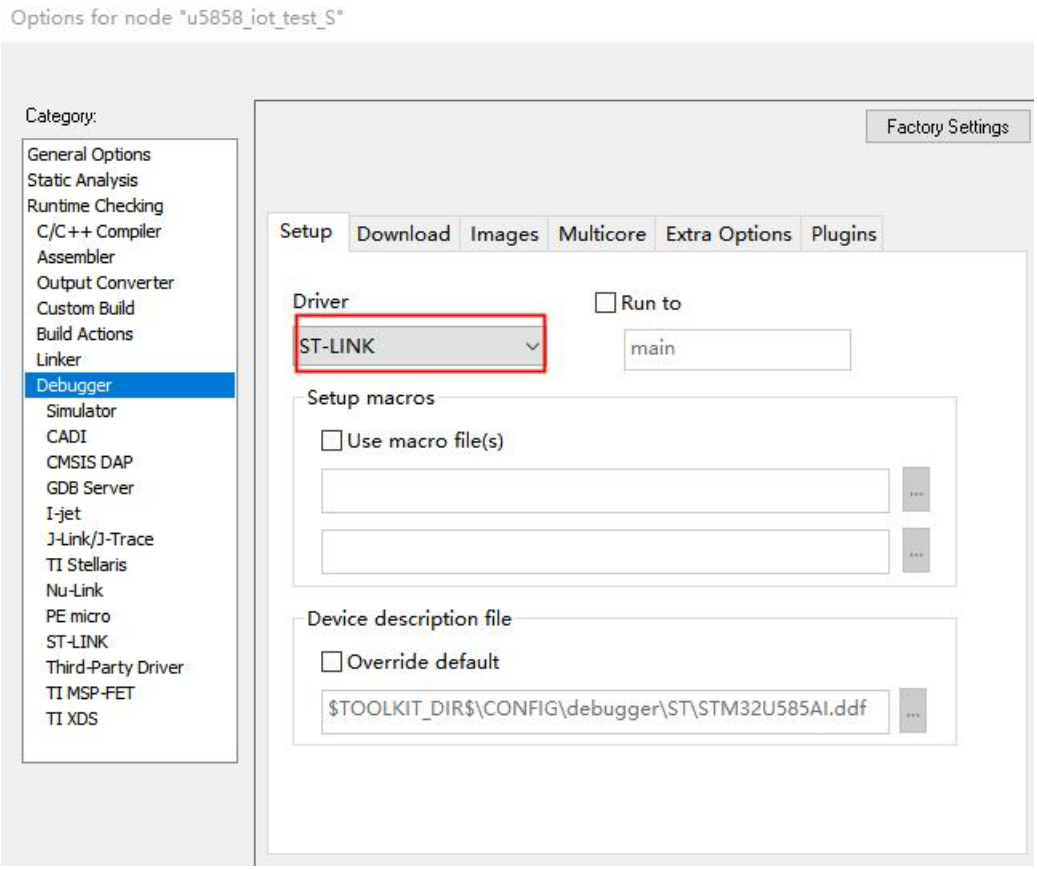
2. regression.sh
3. TFM_UPDATE.sh

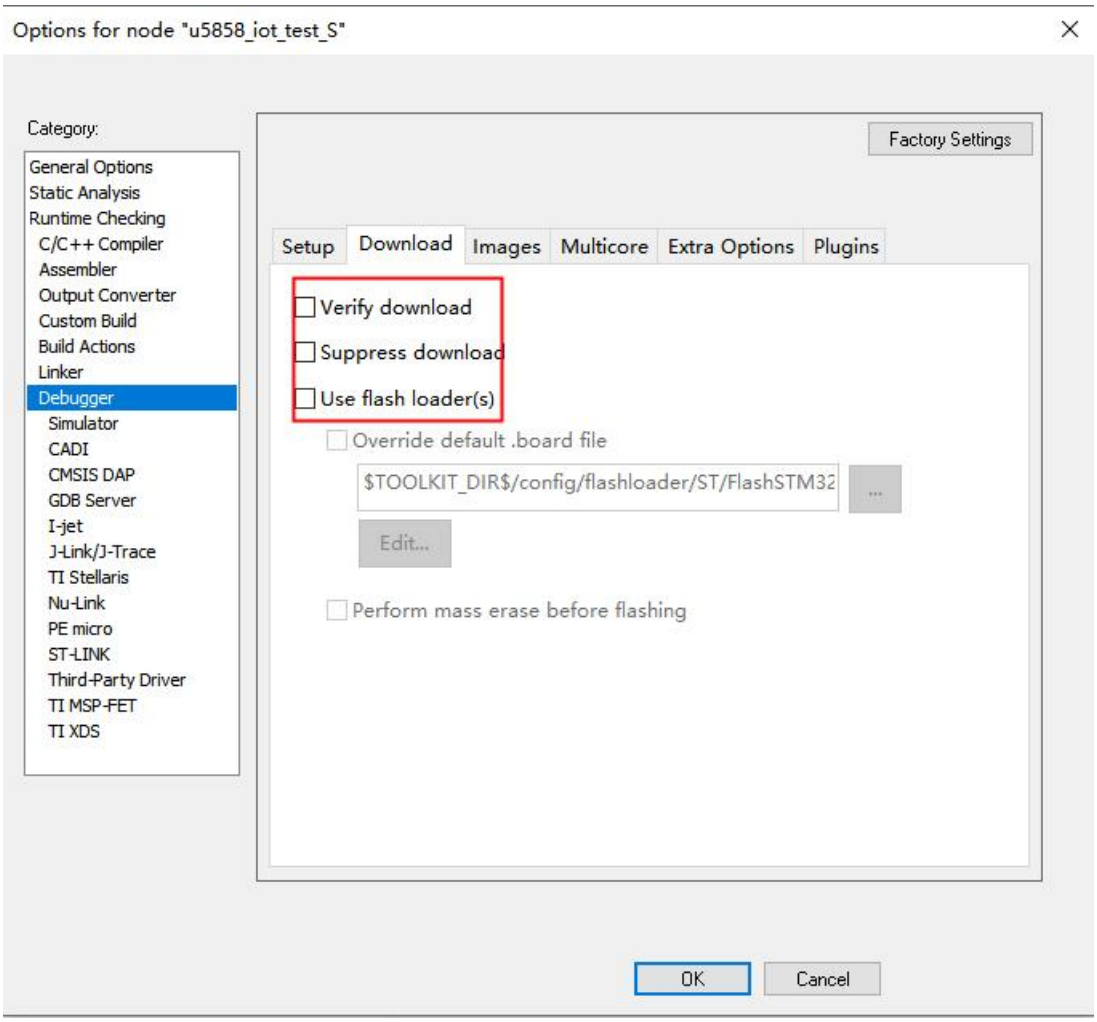
注意需要在脚本中指定下载工具(STM32_Programmer_CLI)的路径;

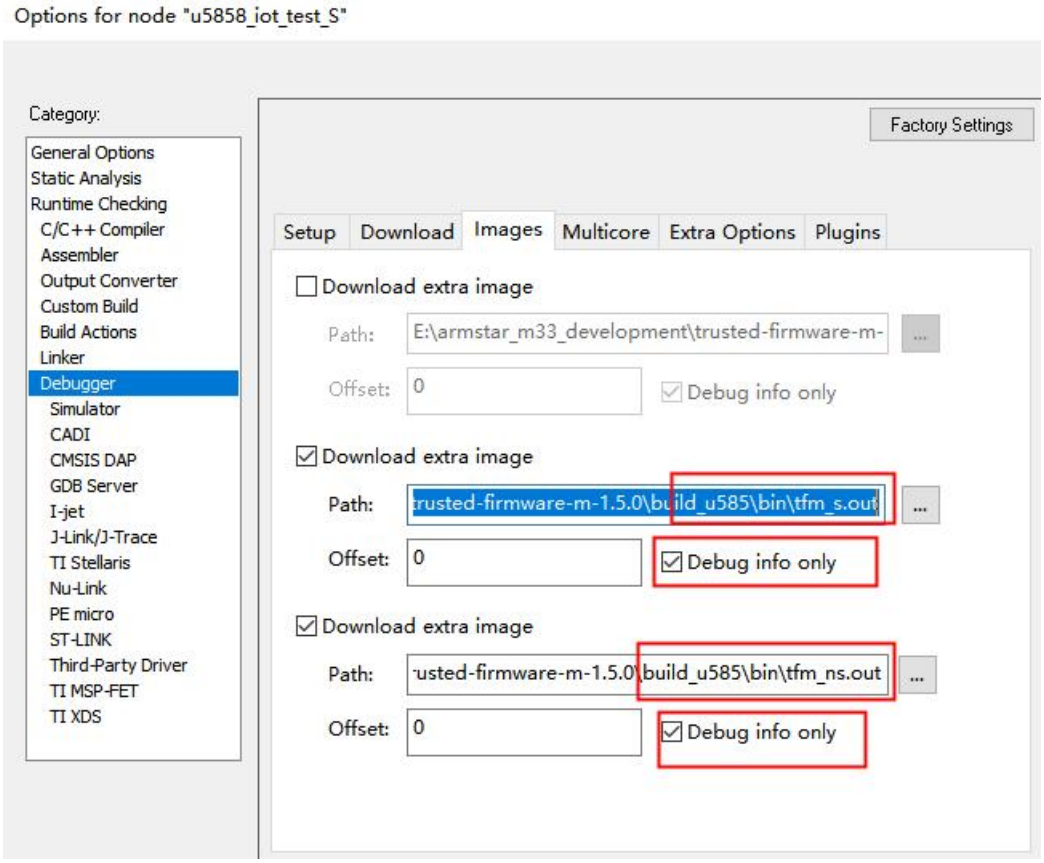
调试配置

选项

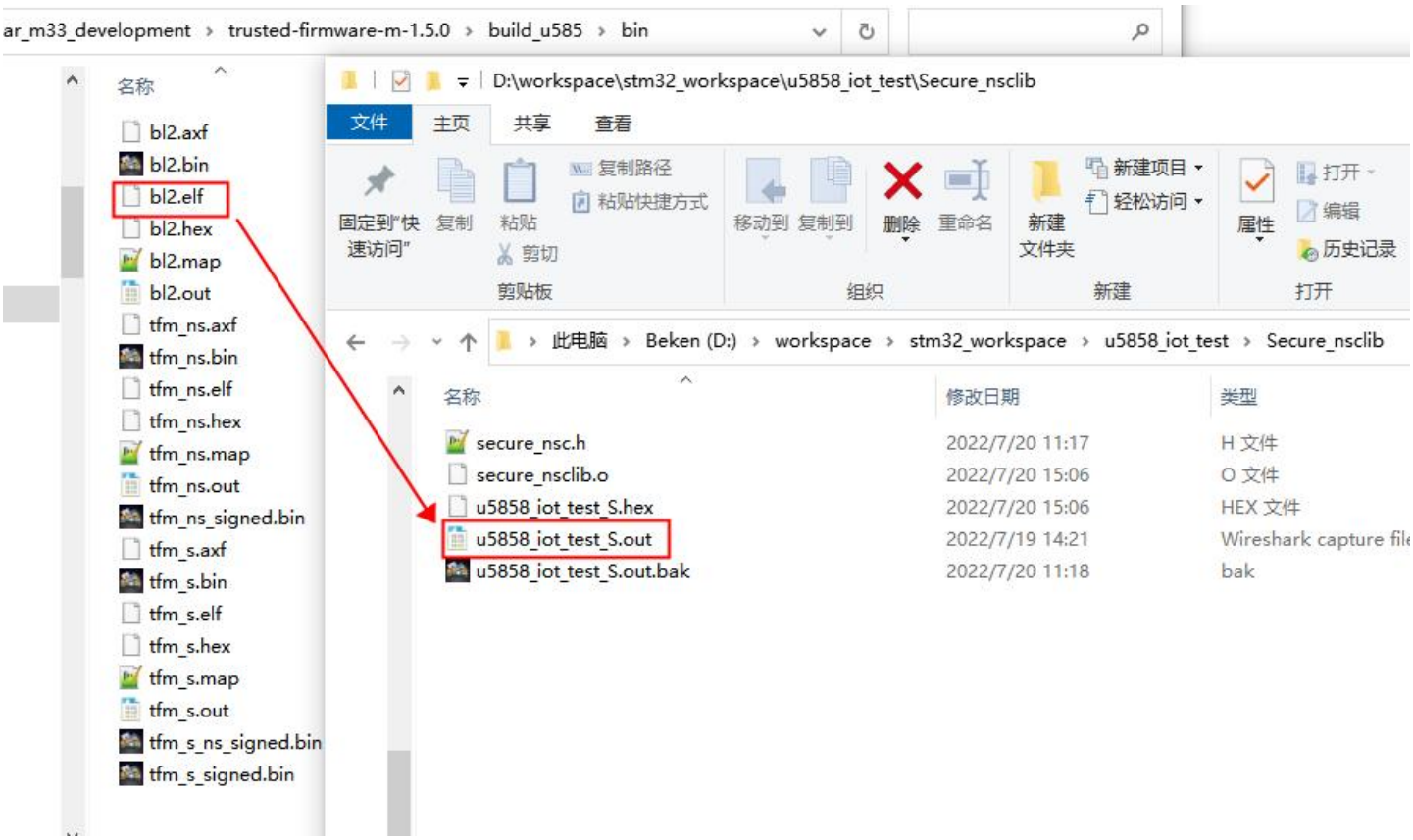





Options for node "u5858_iot_test_S"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Setup
Download
Images
Multicore
Extra Options
Plugins
Verify download
Suppress download
Use flash loader(s)
Override default .board file
$TOOLKIT_DIR$/config/flashloader/ST/FlashSTM32
Edit...
Perform mass erase before flashing
OK
Cancel


Options for node "u5858_iot_test_S"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Setup
Download
Images
Multicore
Extra Options
Plugins
Download extra image
Path:
E:\armstar_m33_development\trusted-firmware-m-
Offset:
0
Debug info only
Download extra image
Path:
trusted-firmware-m-1.5.0\build_u585\bin\tfm_s.out
Offset:
0
Debug info only
Download extra image
Path:
usted-firmware-m-1.5.0\build_u585\bin\tfm_ns.out
Offset:
0
Debug info only

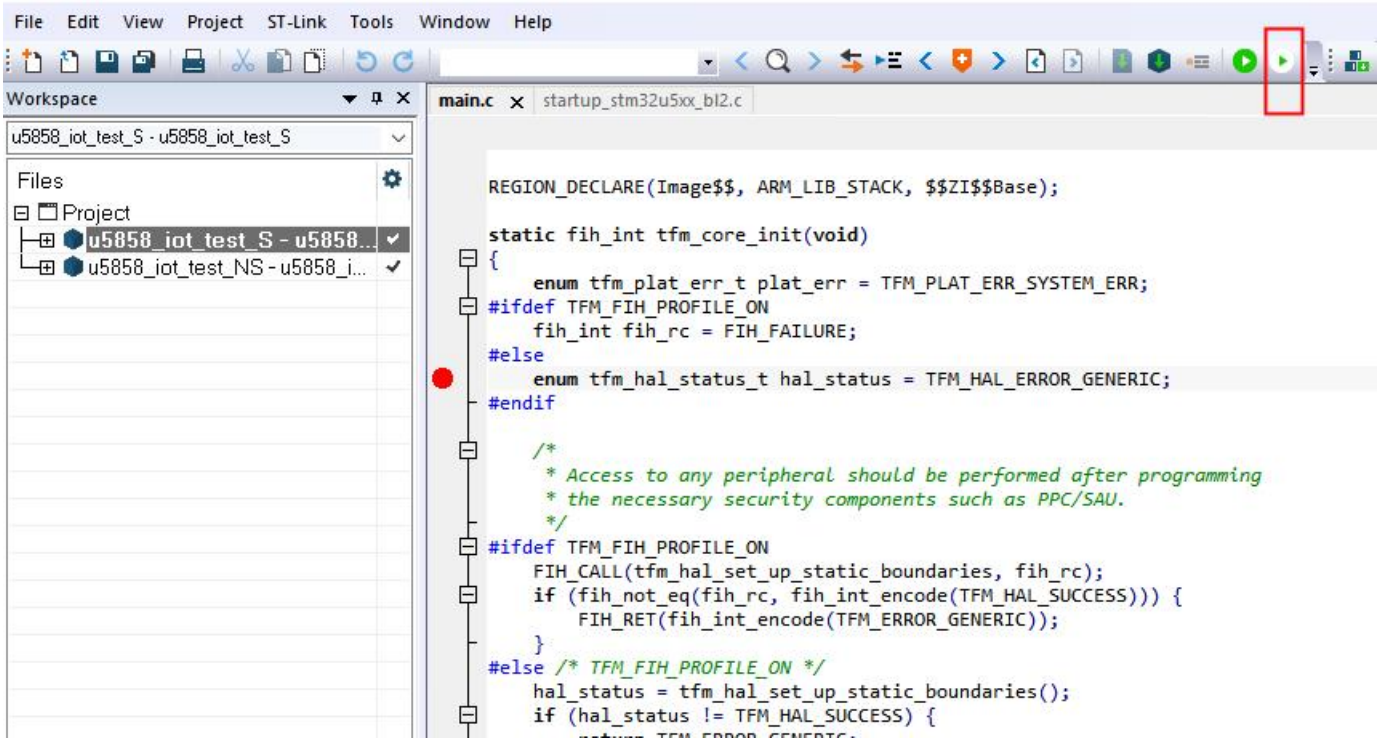
映像文件替换



注意：

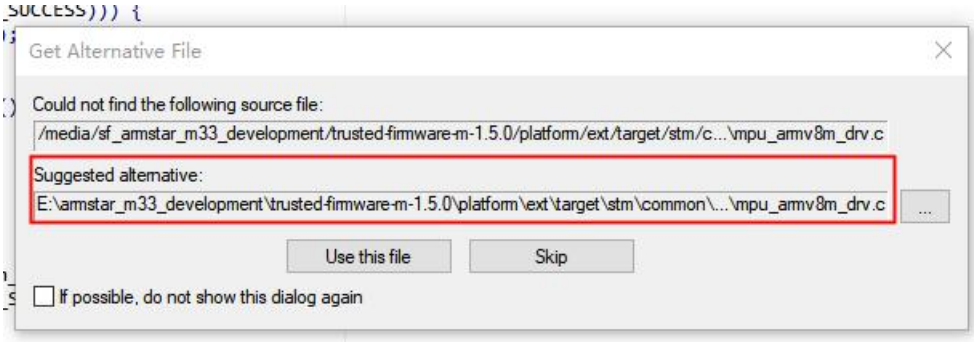
- 如果需要加载多个映像文件，Options->Debugger->Images 选择 Download extra image，勾选 Debug info only.
- 用原映像文件 xxx.elf 改名为 iar 工程生成的 out 文件，这个工程 out 会默认加载调试；

只加载符号表的调试

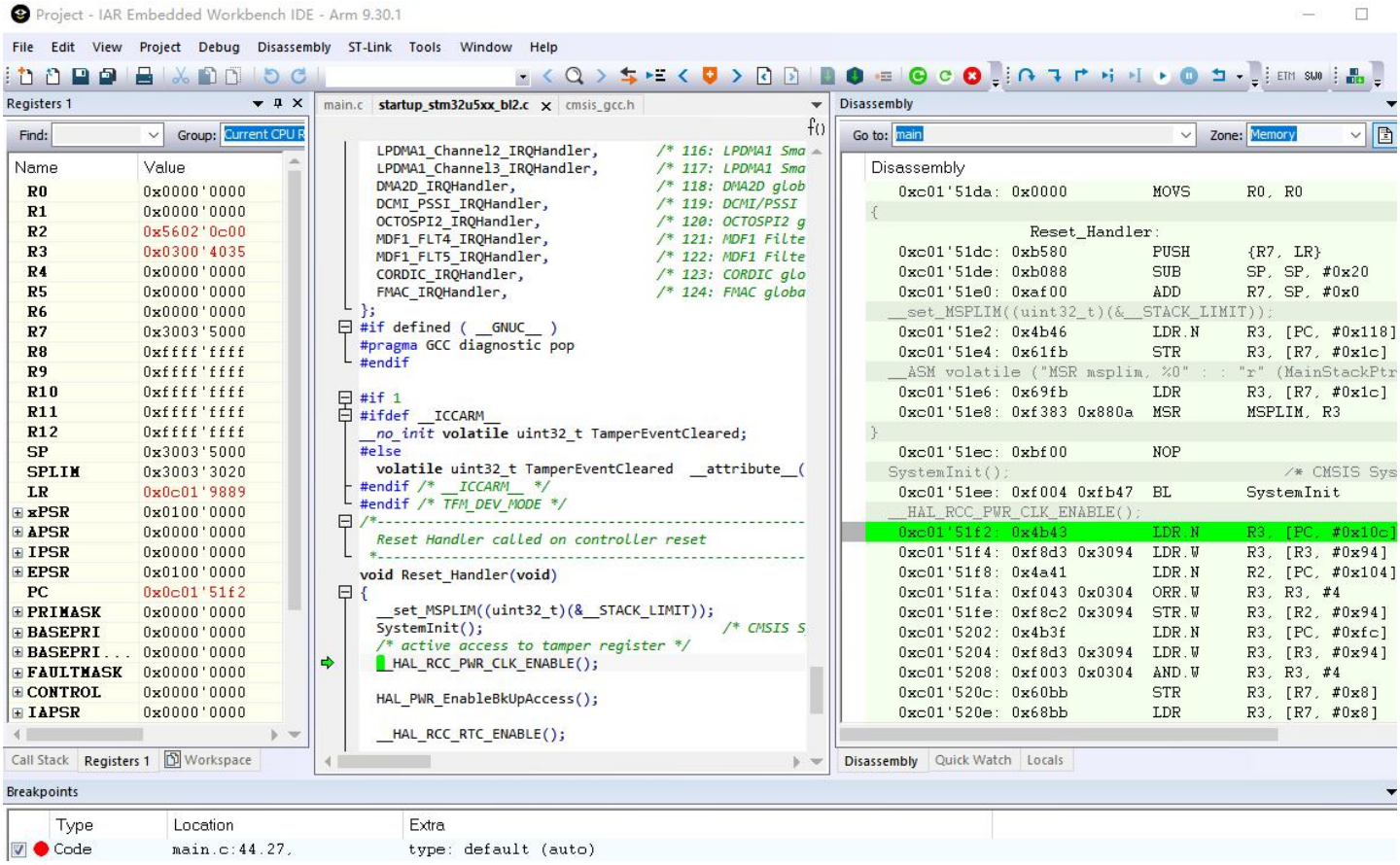


指定符号表路径

编译生成的符号表目录结构跟调试的目录不一致;



正常调试



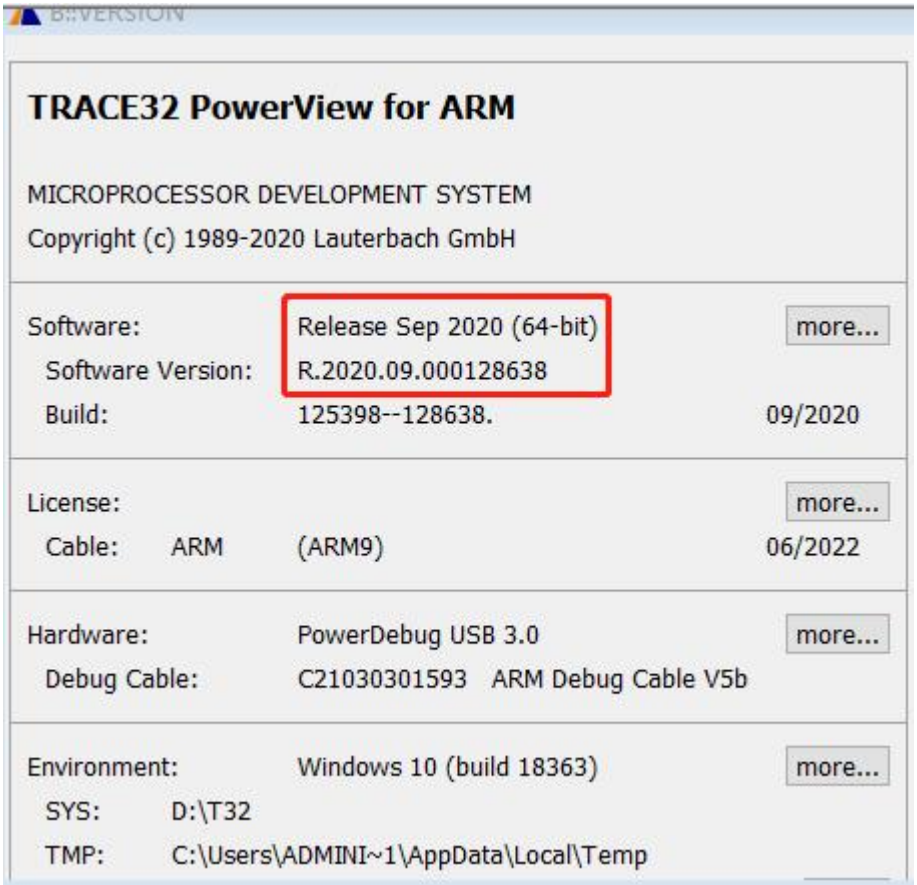
TRACE32 + ARM9

前序

本章后续章节基于如下硬件软件配置，请反复体会/玩味之后阅读。仅做参考。

Personal Computer

Trace32 软件

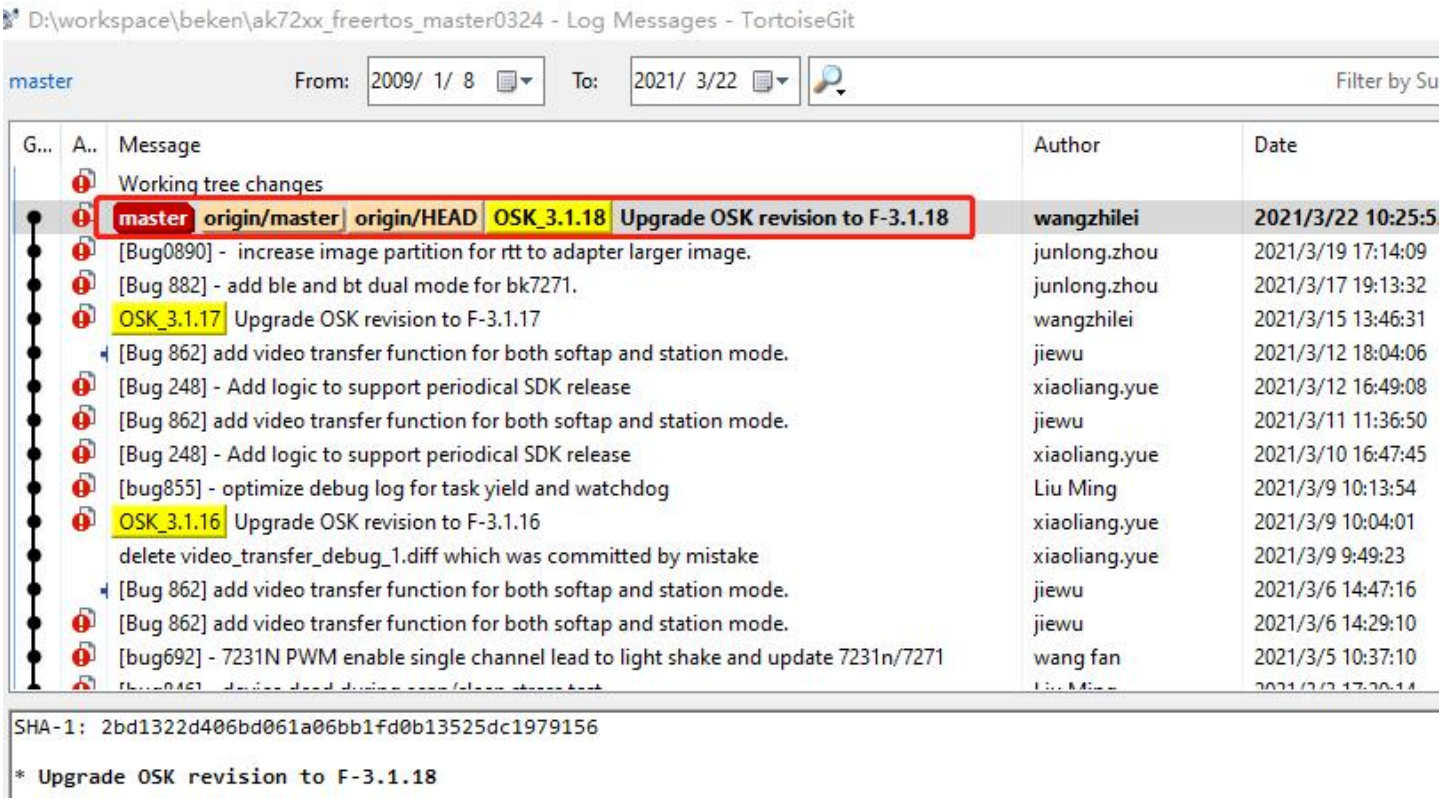


Trace32 硬件

LA-3500 DEBUG-USB3
LA-7742 JTAG-ARM9

Pos.	Qty.	Unit	Order-No.	Text		UnitPrice	TotalPrice
1	1	PCS	LA-3500	PowerDebug Module USB 3.0 Universal debug controller with USB 3.0 for Windows/Linux/MacOSX applicable for all TRACE32 debug cables requires TRACE32 software DVD 2013/02 or newer	RMB	18,950.00	18,950.00
2	1	pcs	LA-7742	JTAG Debugger for ARM9 supports ARM9 (0.4 V - 5 V) supports 5-pin standard JTAG, cJTAG and Serial Wire Debug Port includes software for Windows, Linux and MacOSX requires Power Debug Module cJTAG and SWD require Power Debug Interface USB 2.0/USB 3.0, Power Debug Ethernet, PowerTrace, Power Debug II or PowerDebug PRO	RMB	21,940.00	21,940.00

EVB 固件



EVB 硬件

BK7251_V3 MP OFN68 Oct 15 2018

安装

文件位置：\\192.168.0.30\Documentation\software\wangzhilei\trace32
步骤中请选择：**ICD ARM 32-bit (ARMv4-ARMv7, ARMv8M, Cortex-A/R/M, XScale)**

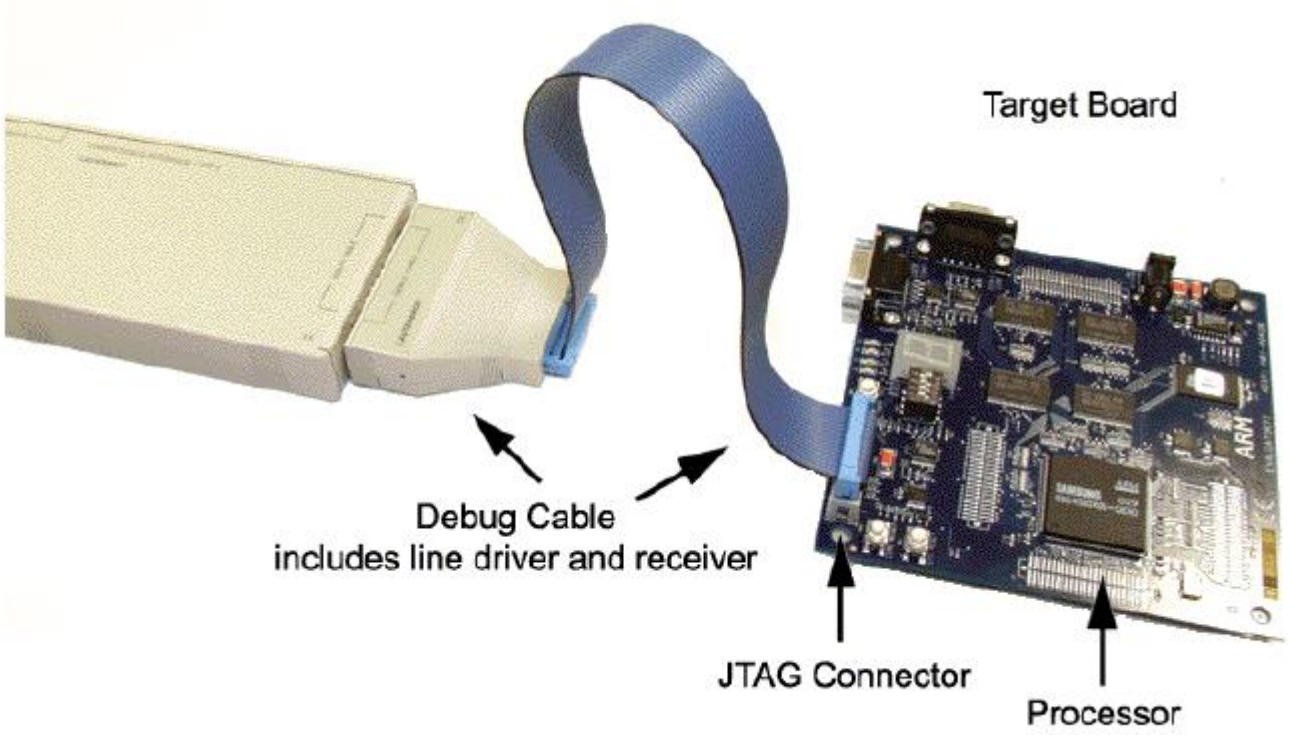
调试

连接

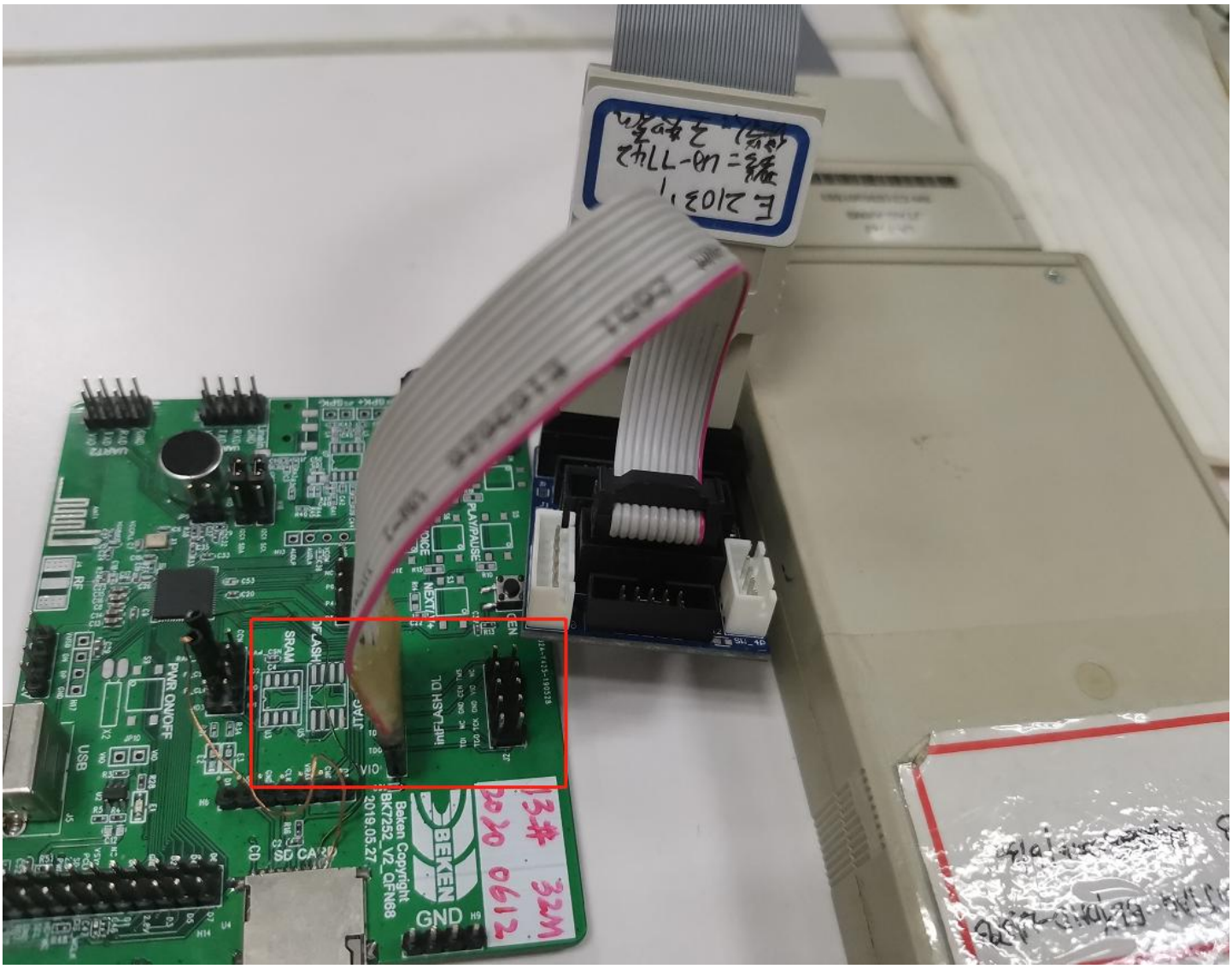
请按照图示连接硬件，**请特别注意 jtag VCC 脚**。
Attention：
禁止在 target/soc 开发板上电情况下拔插 debugger，建议上电顺序：
上电时，debugger→target；
断电时，target→debugger；

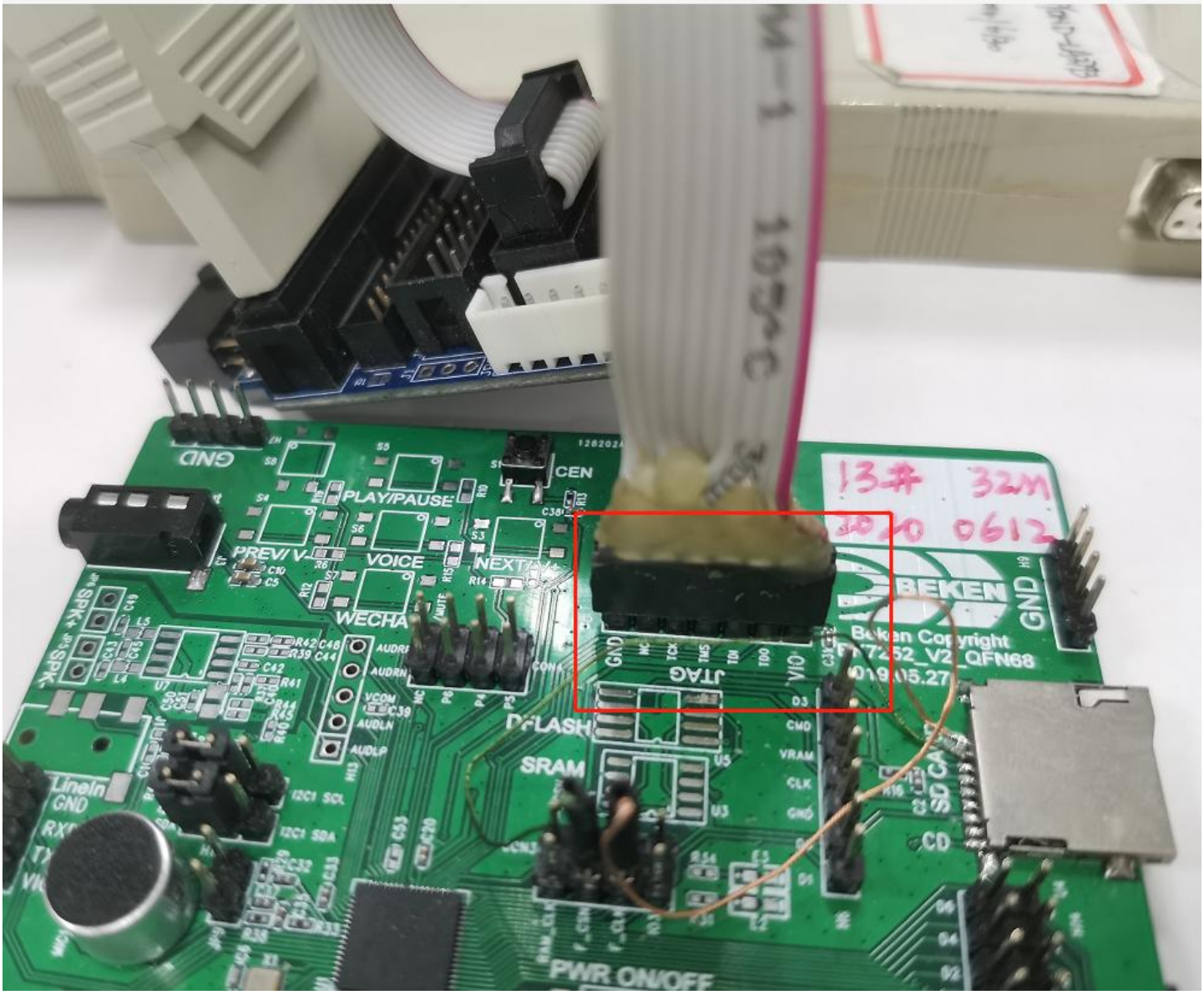
为了保证连接成功，请关闭低功耗，尽量不要切时钟，也不要配置跟 jtag 相关的 gpio 等等。Jtag 引脚 TMS/ TCLK/ TDI/ TDO 一般都和 gpio 复用，如果配置作为 gpio 使用，会影响 jtag 的通讯。

通用连接图如下：


Target Board
Debug Cable
includes line driver and receiver
JTAG Connector
Processor

实际连接图如下:


SRAM
FLASH
JTAG
PWR ON/OFF
USB
SD CARD
BEKEN
GND
Beken Copyright
BK7252_V2_QFN68
2019.03.27



启动

点击开始菜单→TRACE32→**TRACE32 ICD ARM32 USB**

脚本

备注：脚本中文件的相对路径根据 cmm 脚本与对应文件的实际情况配置。

Up Script

```
WinCLEAR

; Refresh when printing
SCREEN.ON

; Debugger Reset
WinPAGE.RESet
AREA.RESet
WinPOS 0. 24. 75. 8.
AREA.view

PRINT "resetting..."
sYmbol.RESet
SYStem.RESet
TASK.RESet
Break.Delete

; select CPU type
SYStem.CPU ARM968E

; set endianism according to the selection
SYStem.Option BigEndian OFF

; Disable interrupts while single stepping
SYStem.Option IMASKASM ON

; Disable interrupts while HLL single stepping
SYStem.Option IMASKHLL ON
SYStem.JtagClock CTCK 1MHz

; start debugger
SYStem.Up

// load selected example program
Data.LOAD.ELF "../out/beken7231_bsp.elf" /RelPATH /PATH ../ /NoCODE

Register.Set PC 0
ENDDO
```

Attach Script

```
WinCLEAR

; Refresh when printing
SCREEN.ON

; Debugger Reset
WinPAGE.RESet
AREA.RESet
WinPOS 0. 24. 75. 8.
AREA.view

PRINT "resetting..."
```

```
sYmbol.RESet
SYStem.RESet
TASK.RESet
Break.Delete

; select CPU type
SYStem.CPU ARM968E

; set endianism according to the selection
SYStem.Option BigEndian OFF

; Disable interrupts while single stepping
SYStem.Option IMASKASM ON

; Disable interrupts while HLL single stepping
SYStem.Option IMASKHLL ON
SYStem.JtagClock CTCK 1MHz

; attach CPU
SYStem.Mode Attach

; load selected example program
Data.LOAD.ELF "../out/beken7231_bsp.elf" /RelPATH /PATH ../ /NoCODE

ENDDO
```

Freertos Support

```
task.CONFIG ../debug/freertos/freertos.t32
MENU.ReProgram    ../debug/freertos/freertos.men
HELP.FILTER.Add rtosfreertos
TASK.STacK.PATtern 0x5A
```

CPU Type

当前 TRACE32 ARM 型号为 LA-7742 JTAG-ARM9。BEKEN 芯片支持的 arm 型号如下图:

CPU	ICE	FIRE	ICD DEBUG	ICD MONITOR	ICD TRACE	POWER INTEGRATOR	INSTRUCTION SIMULATOR
ARM1156T2F-S			YES	YES	YES		YES
ARM1176JZ-S			YES	YES	YES		YES
ARM1176JZF-S			YES	YES	YES		YES
ARM11MPCORE			YES	YES	YES		YES
ARM710T			YES	YES	YES		YES
ARM710T-AMBA		YES	YES	YES	YES		YES
ARM720T			YES	YES	YES		YES
ARM720T-AMBA		YES	YES	YES	YES		YES
ARM740T			YES	YES	YES		YES
ARM740T-AMBA		YES	YES	YES	YES		YES
ARM7DI			YES	YES	YES		YES
ARM7EJ-S			YES	YES	YES		YES
ARM7TDMI		YES	YES	YES	YES		YES
ARM7TDMI-AMBA		YES	YES	YES	YES		YES
ARM7TDMI-S			YES	YES	YES		YES
ARM915T			YES	YES	YES		YES
ARM920T			YES	YES	YES		YES
ARM922T			YES	YES	YES		YES
ARM926EJ-S			YES	YES	YES		YES
ARM940T			YES	YES	YES		YES
ARM946E-S			YES	YES	YES		YES
ARM966E-S			YES	YES	YES		YES
ARM968E-S			YES	YES	YES		YES
ARM9E-S			YES	YES	YES		YES
ARM9EJ-S			YES	YES	YES		YES
ARM9TDMI			YES	YES	YES		YES
AT91RM9200			YES	YES	YES		YES
AT91SAM3A4C			YES	YES	YES		YES
AT91SAM3A8C			YES	YES	YES		YES
AT91SAM3N00A			YES	YES	YES		YES
AT91SAM3N00B			YES	YES	YES		YES
AT91SAM3N0A			YES	YES	YES		YES

代码显示窗口

如果想看符号表，可以 **view 菜单→symbols** 去看。或者输入 **command line** 见下图:

B::sYmbol.Browse.sYmbol

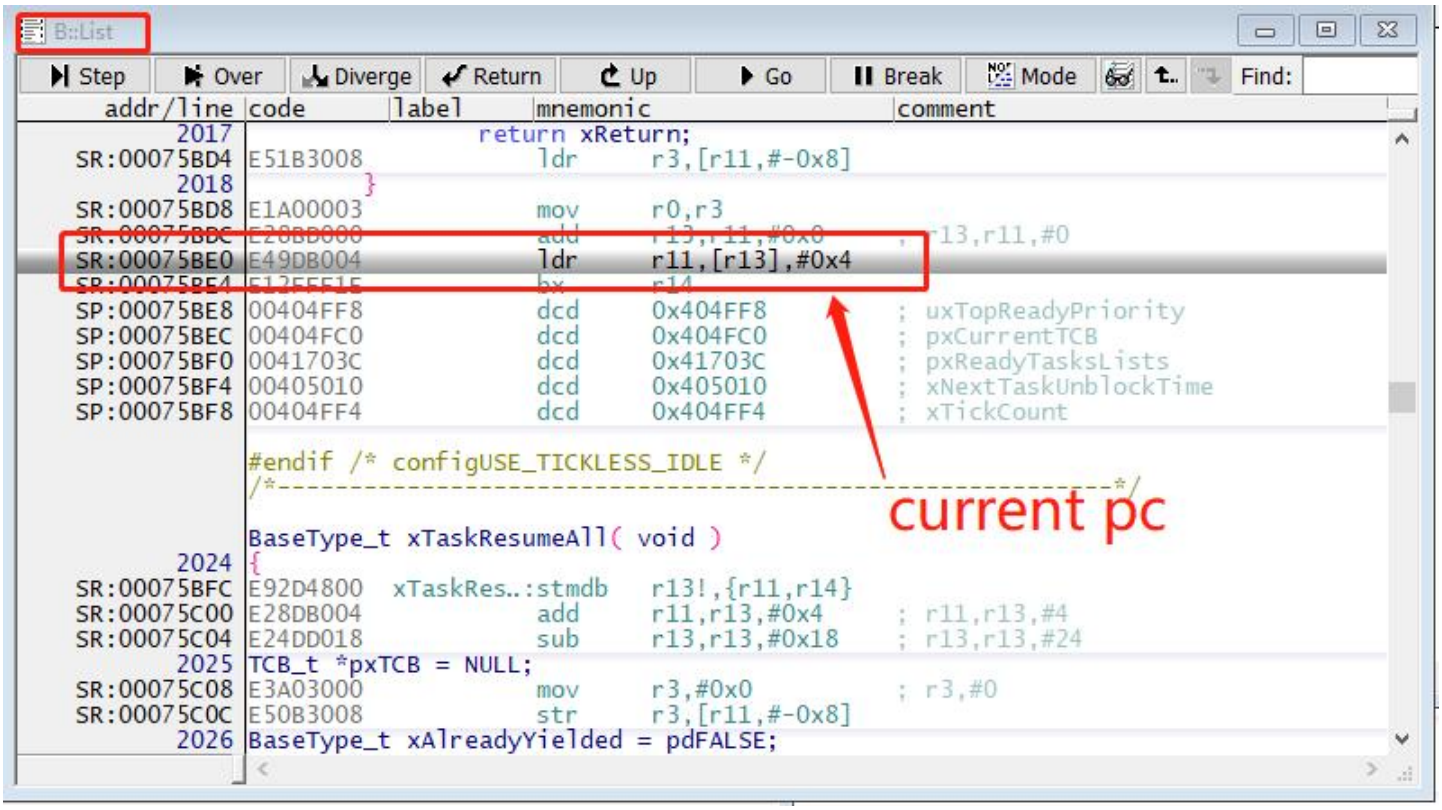
Filter: *** Type: Funcs Vars Sta

symbol	type	address
em_rd		P:000DB4B0--000DB4EB
em_rd		P:000DD644--000DD67F
em_rd		P:000FADAC--000FADE7
em_read_16	(uint16_t ())	P:000D8A46--000D8A77
em_read_8	(uint16_t ())	P:000D8A18--000D8A45
em_wr	(void ())	P:000D8668--000D86A3
em_wr	(void ())	P:000DB4EC--000DB527
em_wr	(void ())	P:00101BB4--00101BEF
em_wr		P:000D8668--000D86A3
em_wr		P:000DB4EC--000DB527
em_wr		P:00101BB4--00101BEF
_empty_ram	(unsigned char)	D:00427C40--00427C40
entry_main	(void ())	P:0001F52C--0001F559
err_tcp	(void ())	P:0003EF4C--0003F0D7
err_to_errno	(int ())	P:0004077C--000407C3
err_to_errno_table	(const int [17] cons..	D:001230D4--00123117
errno	(int)	D:00427C3C--00427C3F
eTaskGetState	(eTaskState ())	P:000758F8--00075A33
etharp_add_static_entry	(err_t ())	P:00047260--00047297
etharp_cached_entry	(u8_t)	D:00404AD8--00404AD8
etharp_cleanup_netif	(void ())	P:0004730C--0004738F
etharp_find_entry	(s8_t ())	P:00046DC4--000470C3
etharp_free_entry	(void ())	P:00046B80--00046C3F
etharp_input	(void ())	P:00047390--00047517

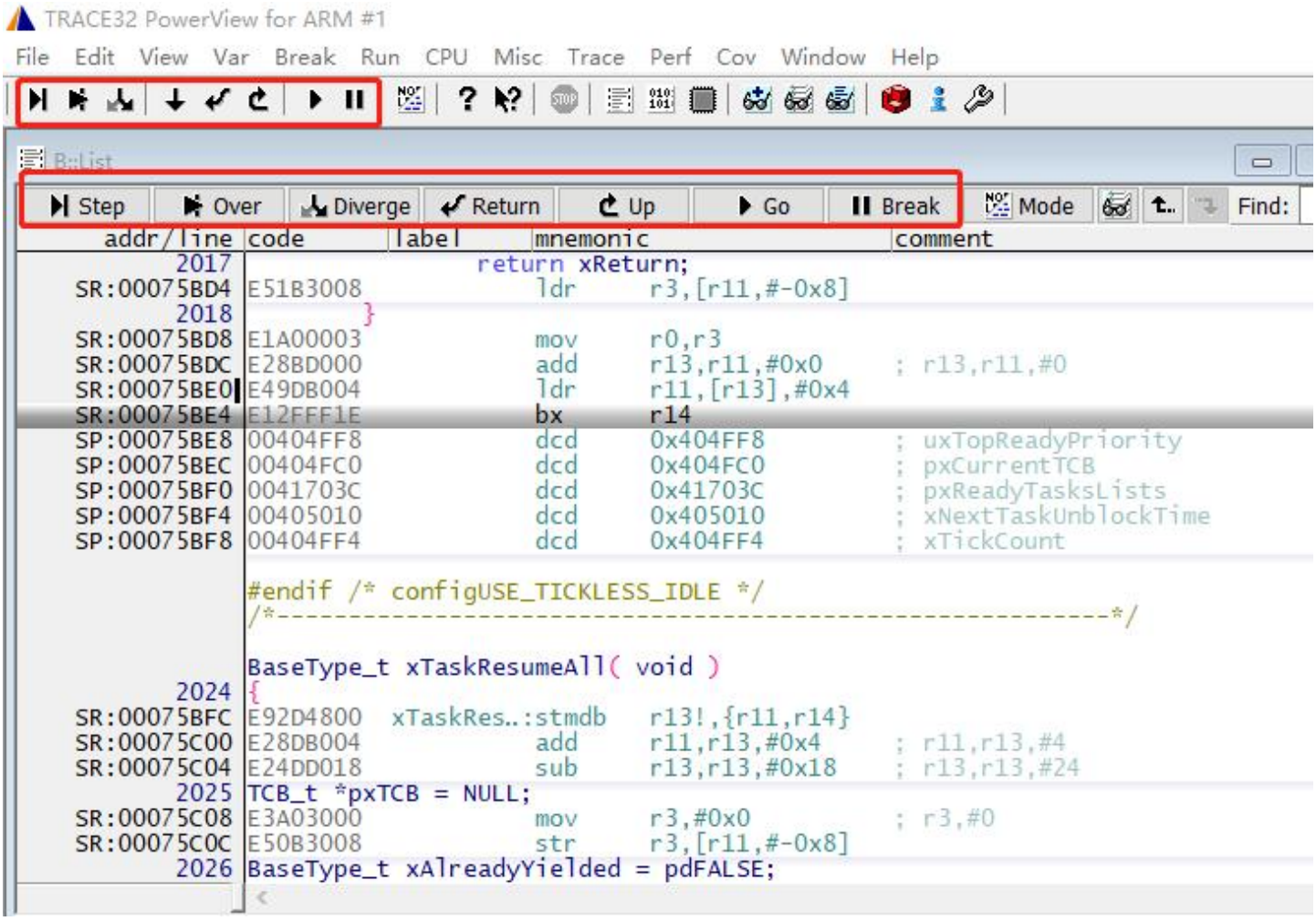
B::sYmbol.Browse.Type

Structs Unions Classes Enums

type	info
enum wps_event	enum (8 bits, unsigned, WPS_EV_M2D =
union wps_event_data	union (52 bytes, struct wps_event_m2d
struct wps_event_er_ap	struct (52 bytes, const u8 * uuid, co
struct wps_event_er_ap_settin..	struct (8 bytes, const u8 * uuid, con
struct wps_event_er_enrollee	struct (40 bytes, const u8 * uuid, co
struct wps_event_er_set_selec..	struct (16 bytes, const u8 * uuid, in
struct wps_event_fail	struct (16 bytes, int msg, u16 config
struct wps_event_m2d	struct (52 bytes, u16 config_methods,
struct wps_event_pwd_auth_fail	struct (16 bytes, int enrollee, int p
struct wps_event_success	struct (6 bytes, u8 [6] peer_macaddr)
enum wps_mode	enum (8 bits, unsigned, WPS_MODE_NONE
enum wps_msg_type	enum (8 bits, unsigned, WPS_Beacon =
struct wps_registrar	struct (0 bytes,)
enum wps_state	enum (8 bits, unsigned, WPS_STATE_NOT
write_req_t	typedef (struct)
struct xLIST	struct (20 bytes, UBaseType_t uxNumbe
struct xLIST_ITEM	struct (20 bytes, TickType_t xItemVal
struct xMEMORY_REGION	struct (12 bytes, void * pvBaseAddres
struct xMINI_LIST_ITEM	struct (12 bytes, TickType_t xItemVal
xQUEUE	typedef (struct QueueDefinition)
struct xtal_cali_st	struct (6 bytes, INT8 xtal_c_delta, I
XTAL_CALI_ST	typedef (struct xtal_cali_st)
struct xTASK_STATUS	struct (36 bytes, TaskHandle_t xHandl
struct xTIME_OUT	struct (8 bytes, BaseType_t xOverflow
xTIMER	typedef (struct tmrTimerControl)


B::List
Step
Over
Diverge
Return
Up
Go
Break
Mode
Find:
addr/line
code
label
mnemonic
comment
2017 return xReturn;
SR:00075BD4 E51B3008 ldr r3,[r11,#-0x8]
2018 }
SR:00075BD8 E1A00003 mov r0,r3
SR:00075BDC E28BD000 add r13,r11,#0x0 ; r13,r11,#0
SR:00075BE0 E49DB004 ldr r11,[r13],#0x4
SR:00075BE4 E12FFF1E bx r14
SP:00075BE8 00404FF8 dcd 0x404FF8 ; uxTopReadyPriority
SP:00075BEC 00404FC0 dcd 0x404FC0 ; pxCurrentTCB
SP:00075BF0 0041703C dcd 0x41703C ; pxReadyTasksLists
SP:00075BF4 00405010 dcd 0x405010 ; xNextTaskUnblockTime
SP:00075BF8 00404FF4 dcd 0x404FF4 ; xTickCount
#endif /* configUSE_TICKLESS_IDLE */
/*----------*/
current pc
BaseType_t xTaskResumeAll(void)
2024 {
SR:00075BFC E92D4800 xTaskRes..:stmdb r13!,{r11,r14}
SR:00075C00 E28DB004 add r11,r13,#0x4 ; r11,r13,#4
SR:00075C04 E24DD018 sub r13,r13,#0x18 ; r13,r13,#24
2025 TCB_t *pxTCB = NULL;
SR:00075C08 E3A03000 mov r3,#0x0 ; r3,#0
SR:00075C0C E50B3008 str r3,[r11,#-0x8]
2026 BaseType_t xAlreadyYielded = pdFALSE;

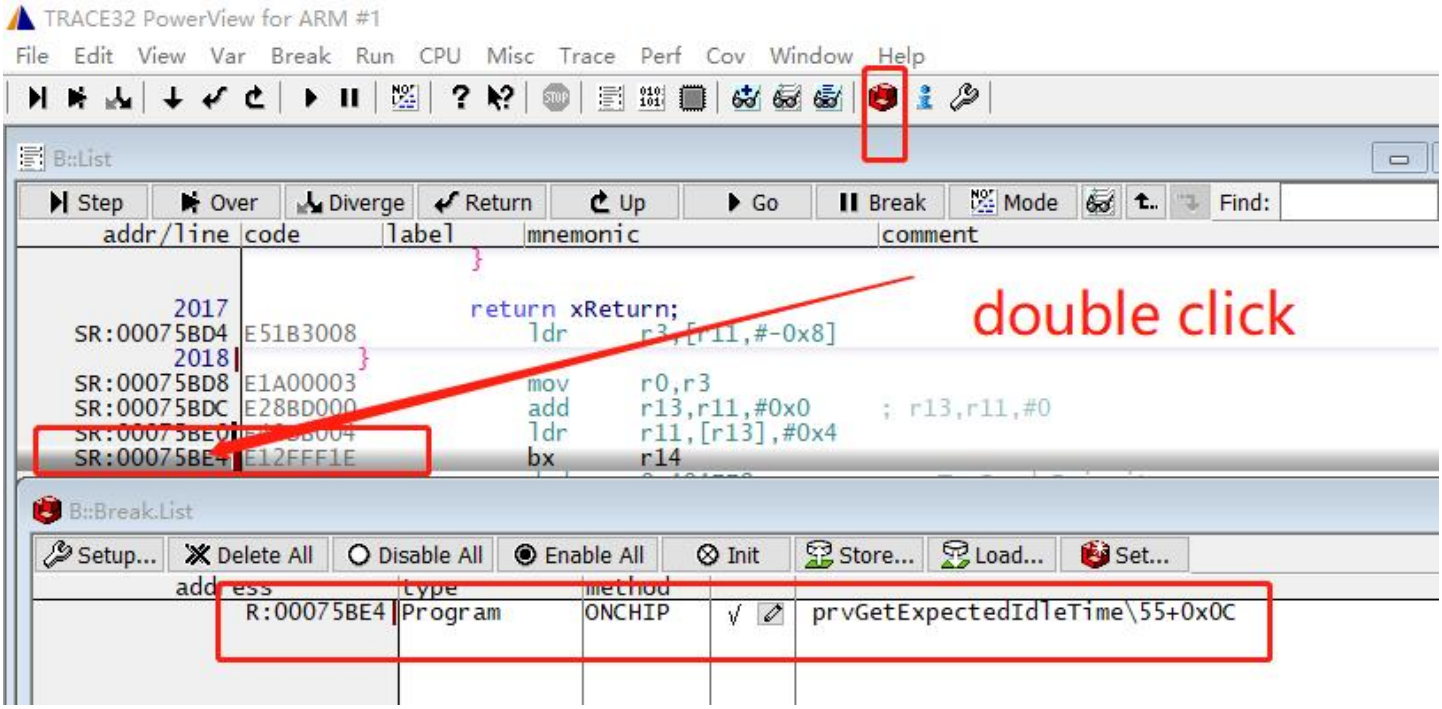
单步执行、运行、暂停


TRACE32 PowerView for ARM #1
File Edit View Var Break Run CPU Misc Trace Perf Cov Window Help
B::List
Step
Over
Diverge
Return
Up
Go
Break
Mode
Find:
addr/line
code
label
mnemonic
comment
2017 return xReturn;
SR:00075BD4 E51B3008 ldr r3,[r11,#-0x8]
2018 }
SR:00075BD8 E1A00003 mov r0,r3
SR:00075BDC E28BD000 add r13,r11,#0x0 ; r13,r11,#0
SR:00075BE0 E49DB004 ldr r11,[r13],#0x4
SR:00075BE4 E12FFF1E bx r14
SP:00075BE8 00404FF8 dcd 0x404FF8 ; uxTopReadyPriority
SP:00075BEC 00404FC0 dcd 0x404FC0 ; pxCurrentTCB
SP:00075BF0 0041703C dcd 0x41703C ; pxReadyTasksLists
SP:00075BF4 00405010 dcd 0x405010 ; xNextTaskUnblockTime
SP:00075BF8 00404FF4 dcd 0x404FF4 ; xTickCount
#endif /* configUSE_TICKLESS_IDLE */
/*----------*/
BaseType_t xTaskResumeAll(void)
2024 {
SR:00075BFC E92D4800 xTaskRes..:stmdb r13!,{r11,r14}
SR:00075C00 E28DB004 add r11,r13,#0x4 ; r11,r13,#4
SR:00075C04 E24DD018 sub r13,r13,#0x18 ; r13,r13,#24
2025 TCB_t *pxTCB = NULL;
SR:00075C08 E3A03000 mov r3,#0x0 ; r3,#0
SR:00075C0C E50B3008 str r3,[r11,#-0x8]
2026 BaseType_t xAlreadyYielded = pdFALSE;

设置断点

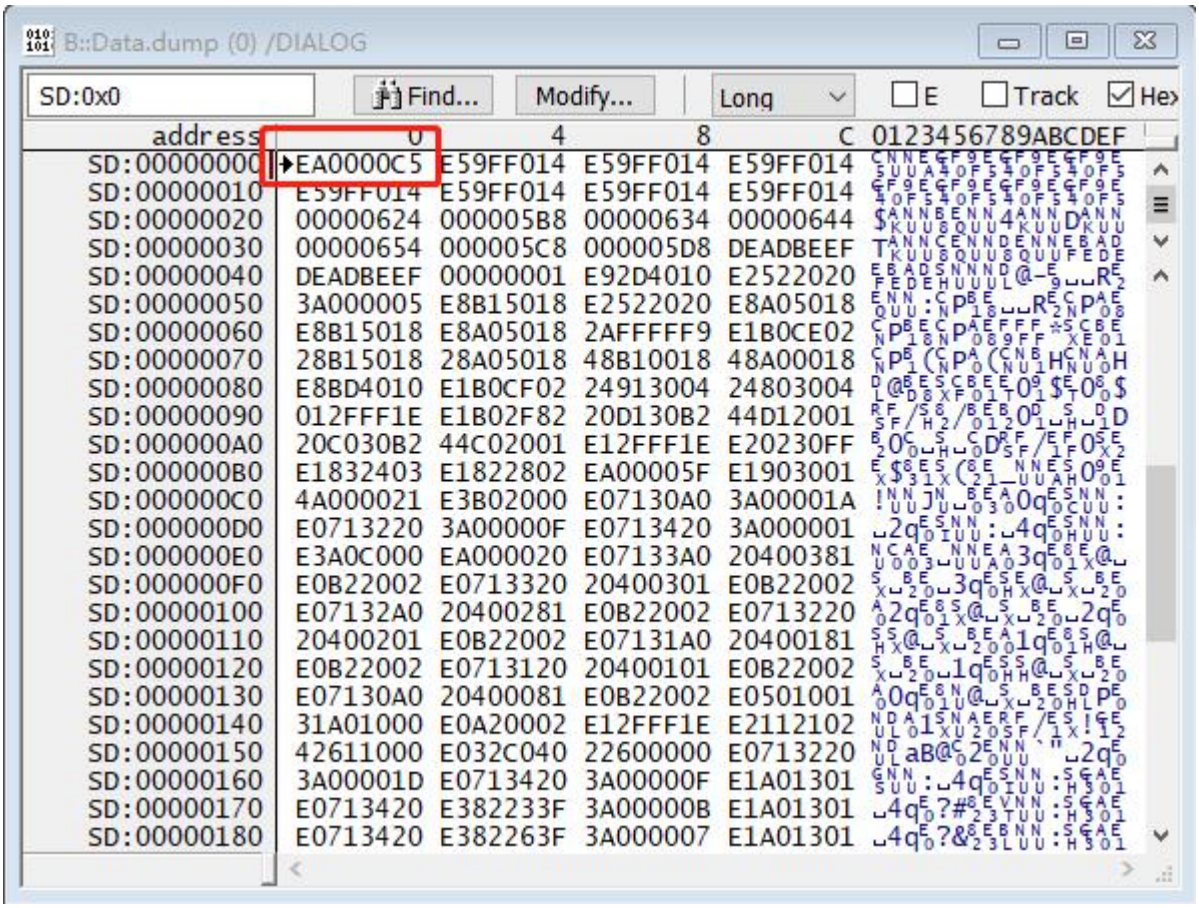
双击代码显示框指定行最左侧

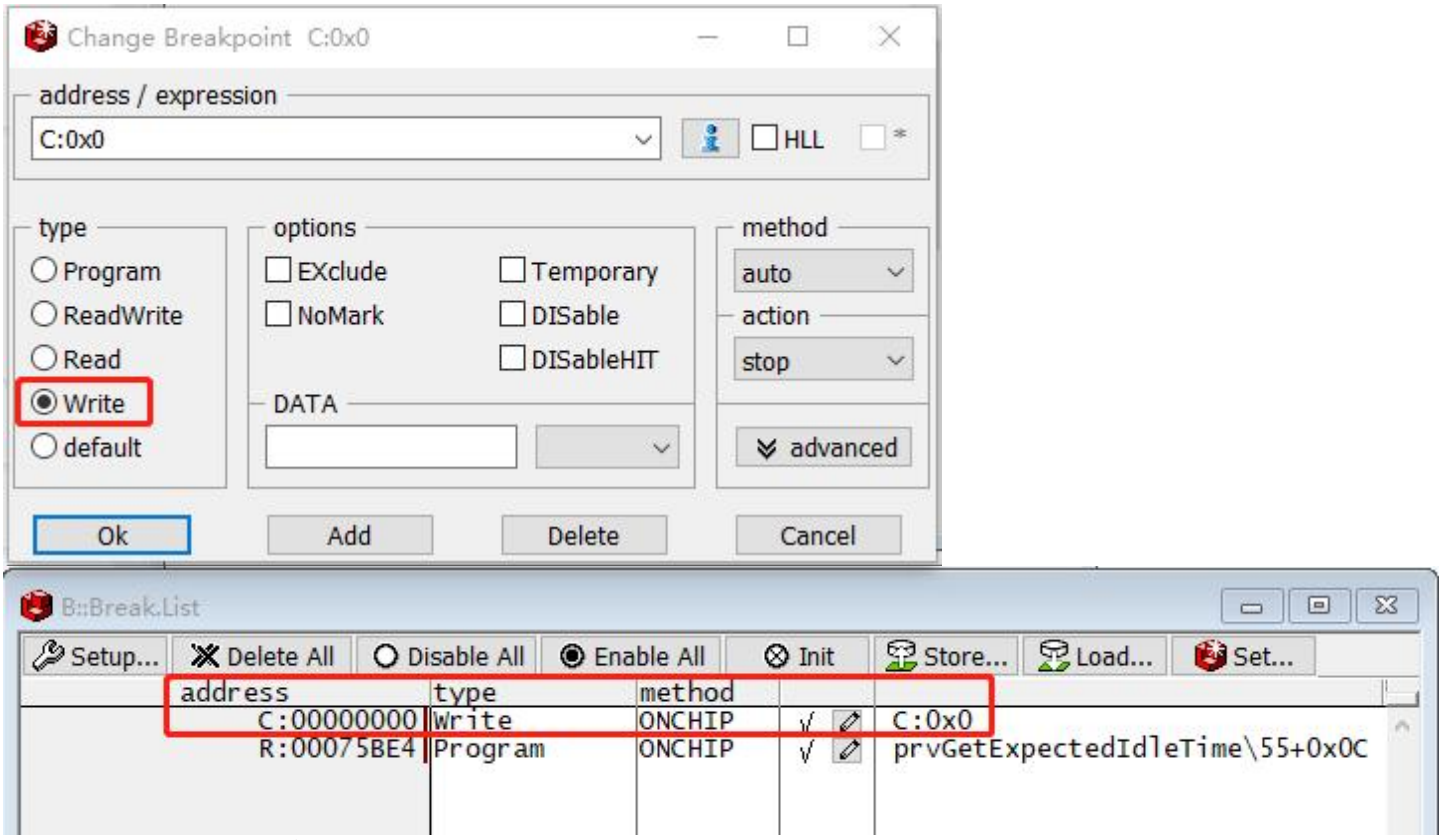
详细可以参考后续章节



打开 Data.dump 选中对应地址右键

详细可以参考后续章节

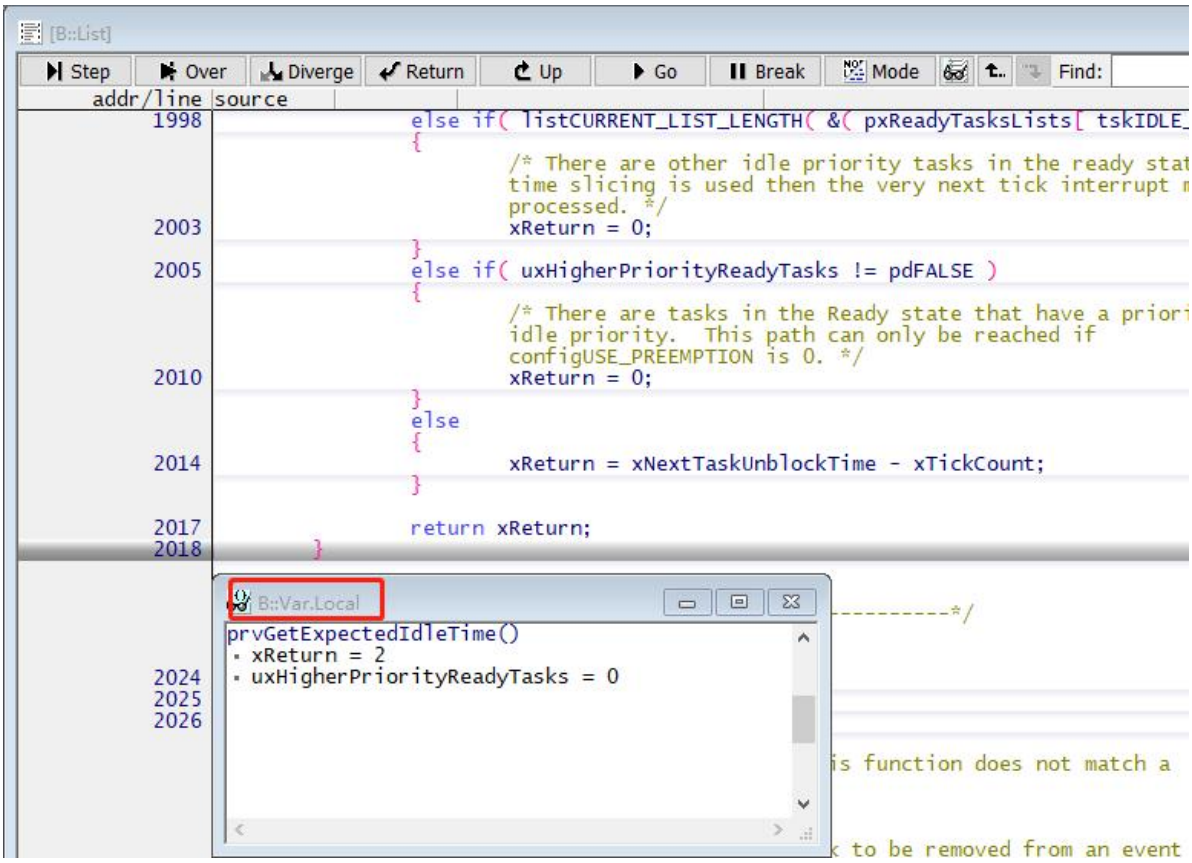




（上图为 0 地址写断点实例）

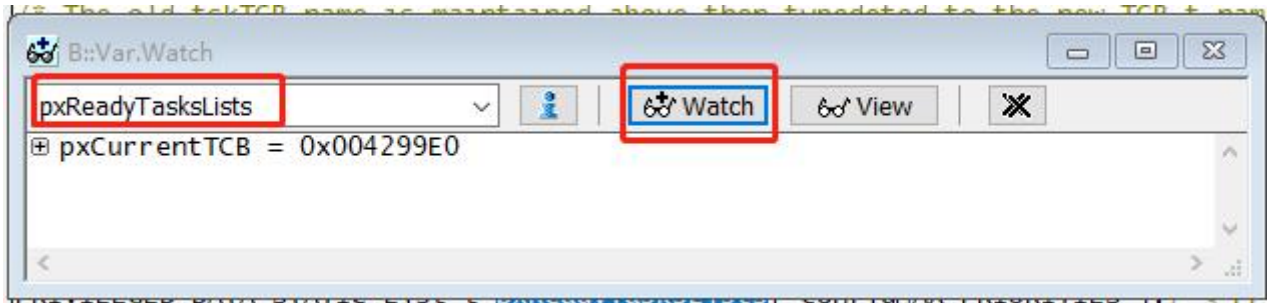
观察变量

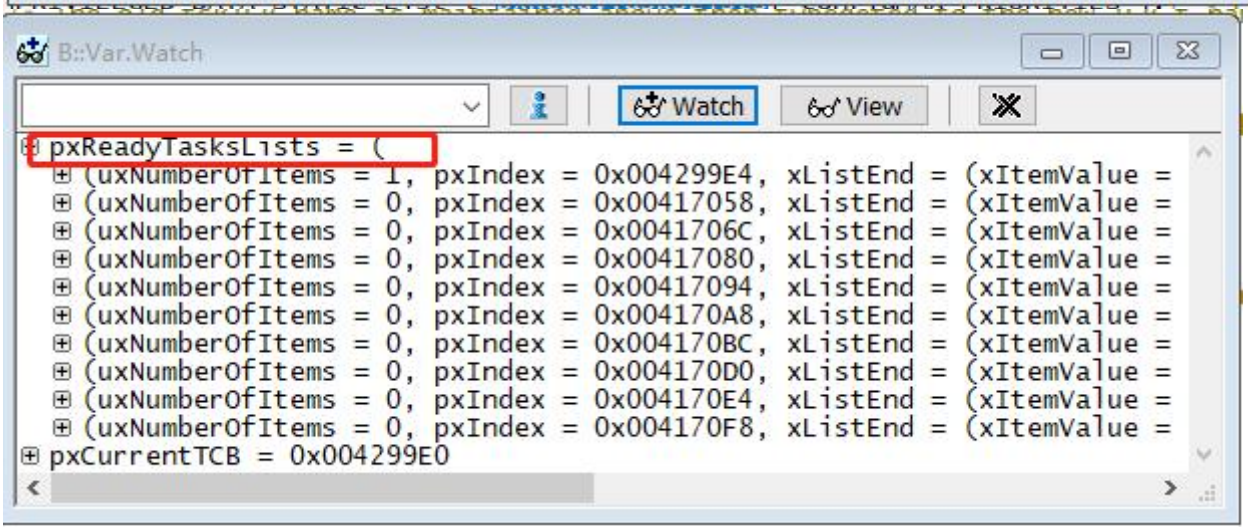
局部变量查看



全局变量查看

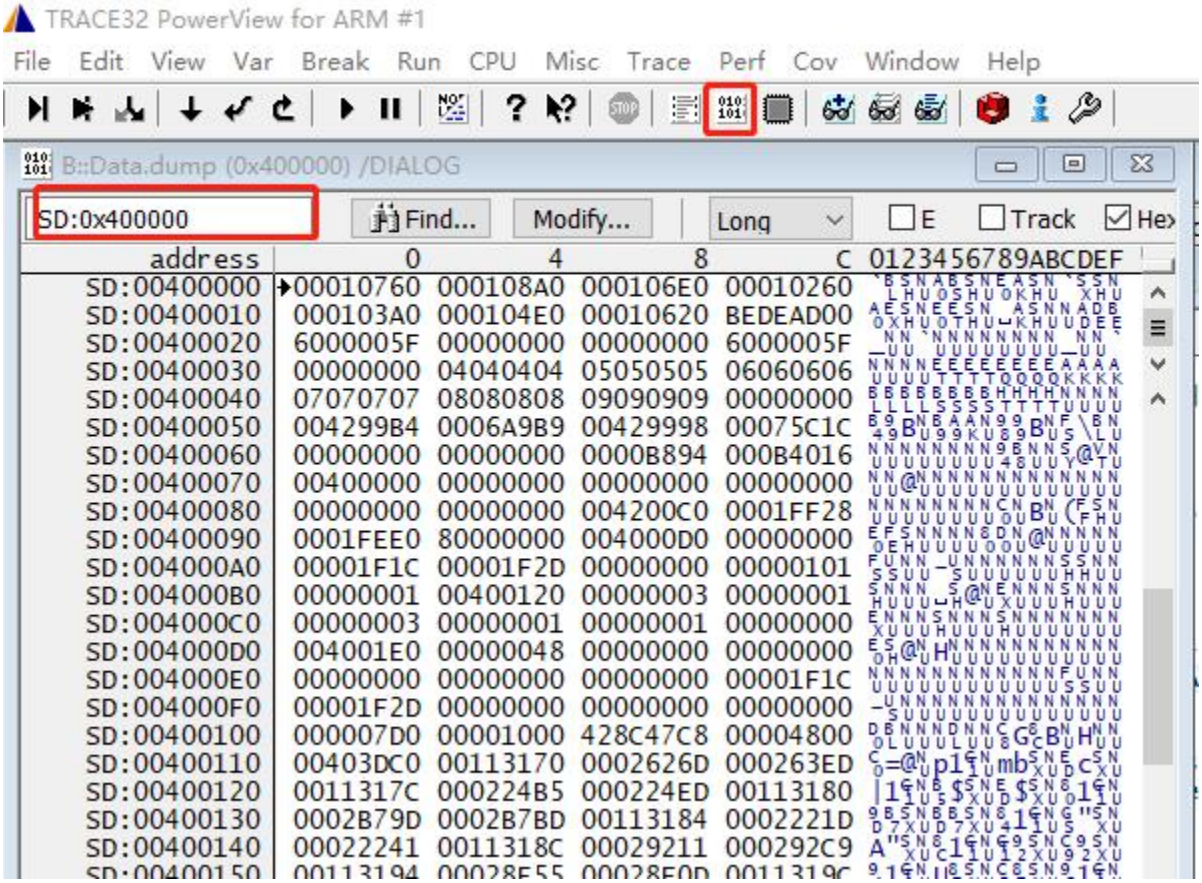
拷贝粘贴对应的全局变量，然后查看





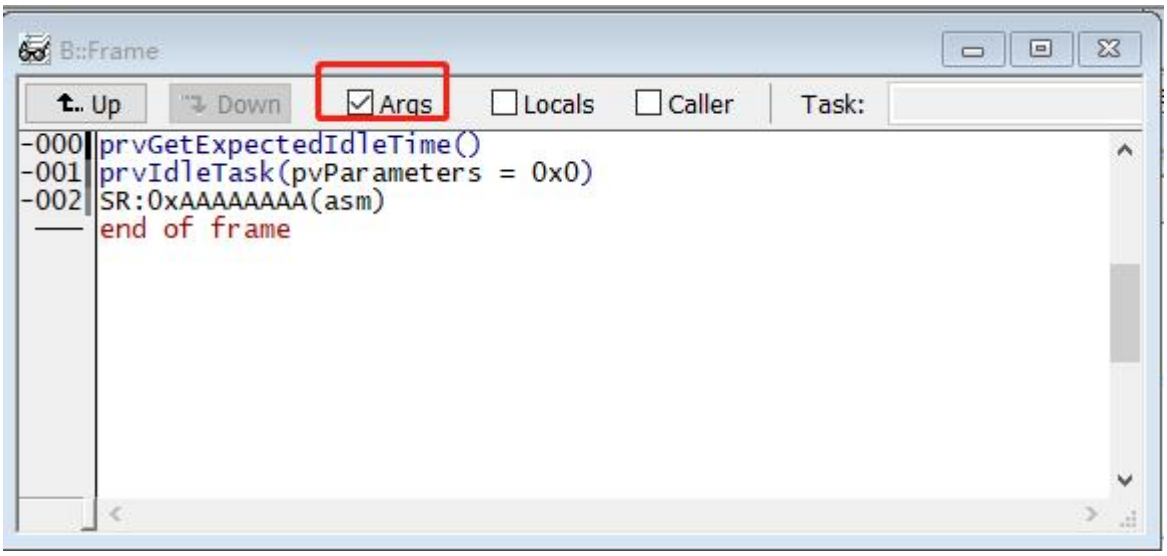
观察内存

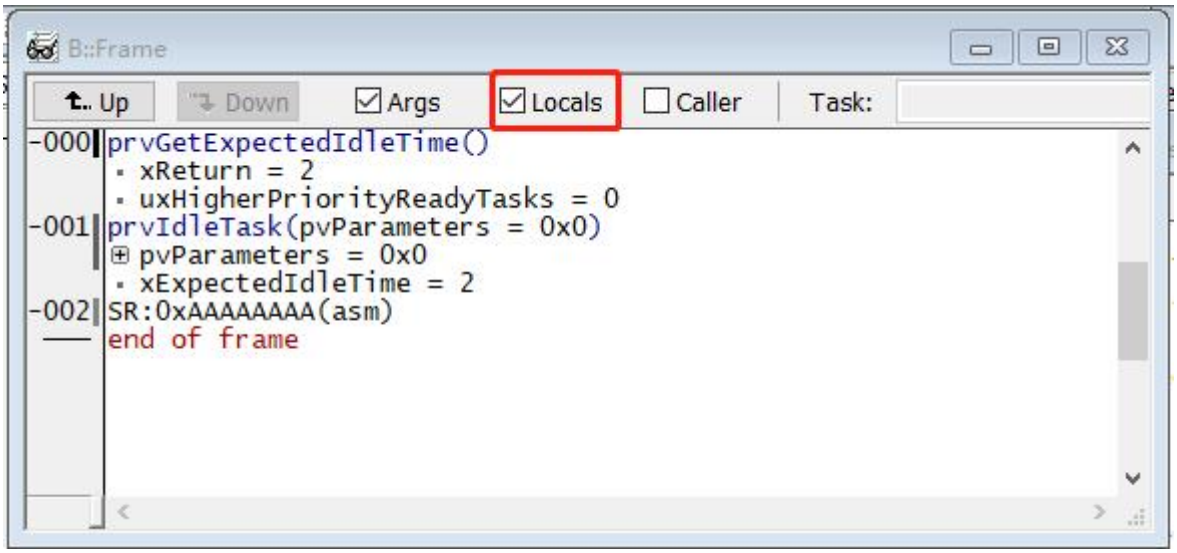
输入 **Data.dump** 0xyyyyyyyy 或者点击如下图:

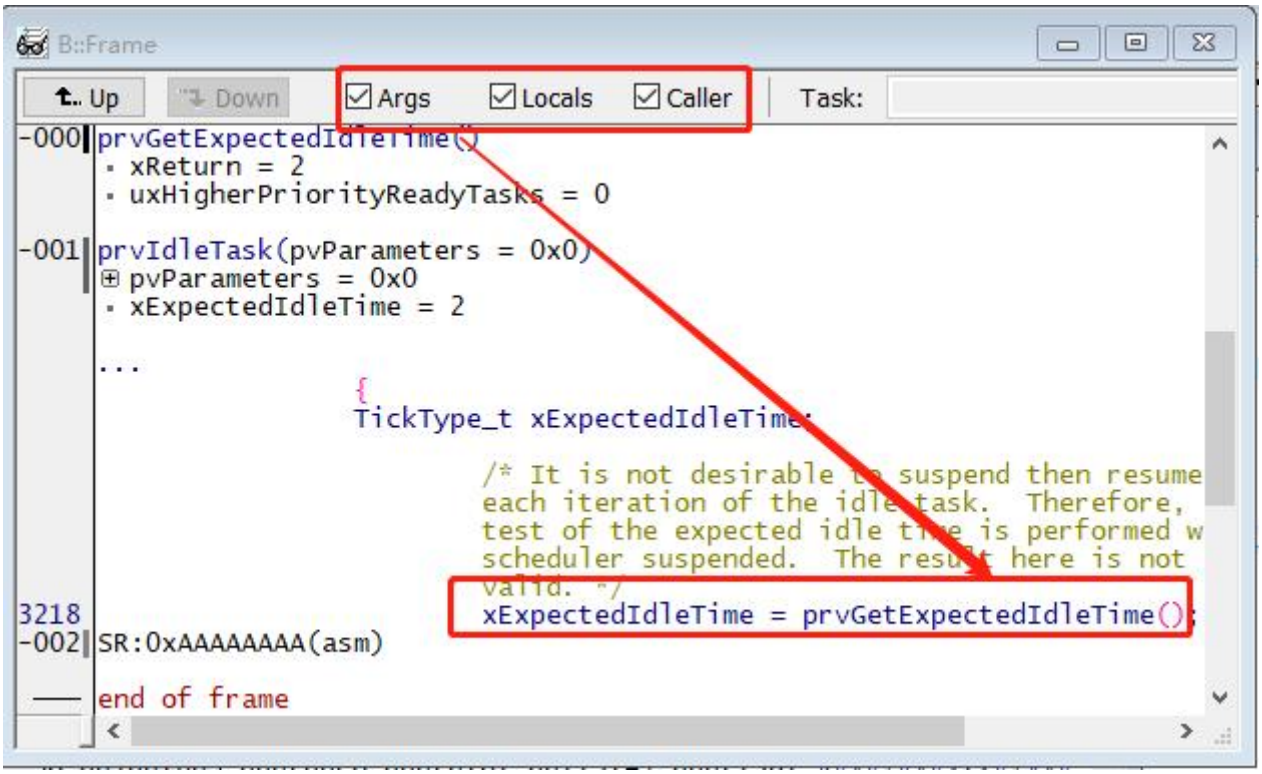


查看调用栈

输入命令 Frame 或者菜单里面点击 View→Stackframe







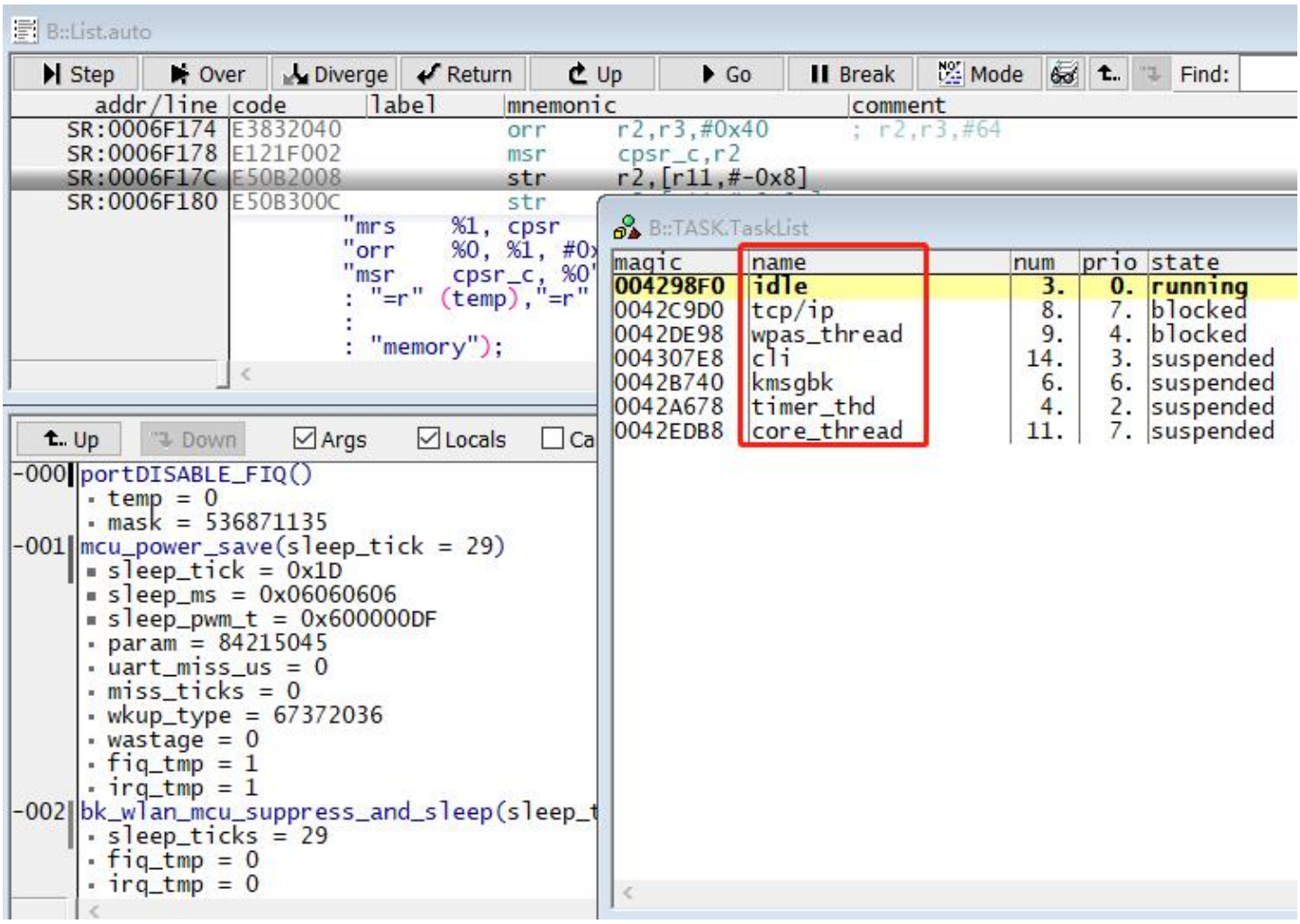
Freertos 特性

脚本

参见章节

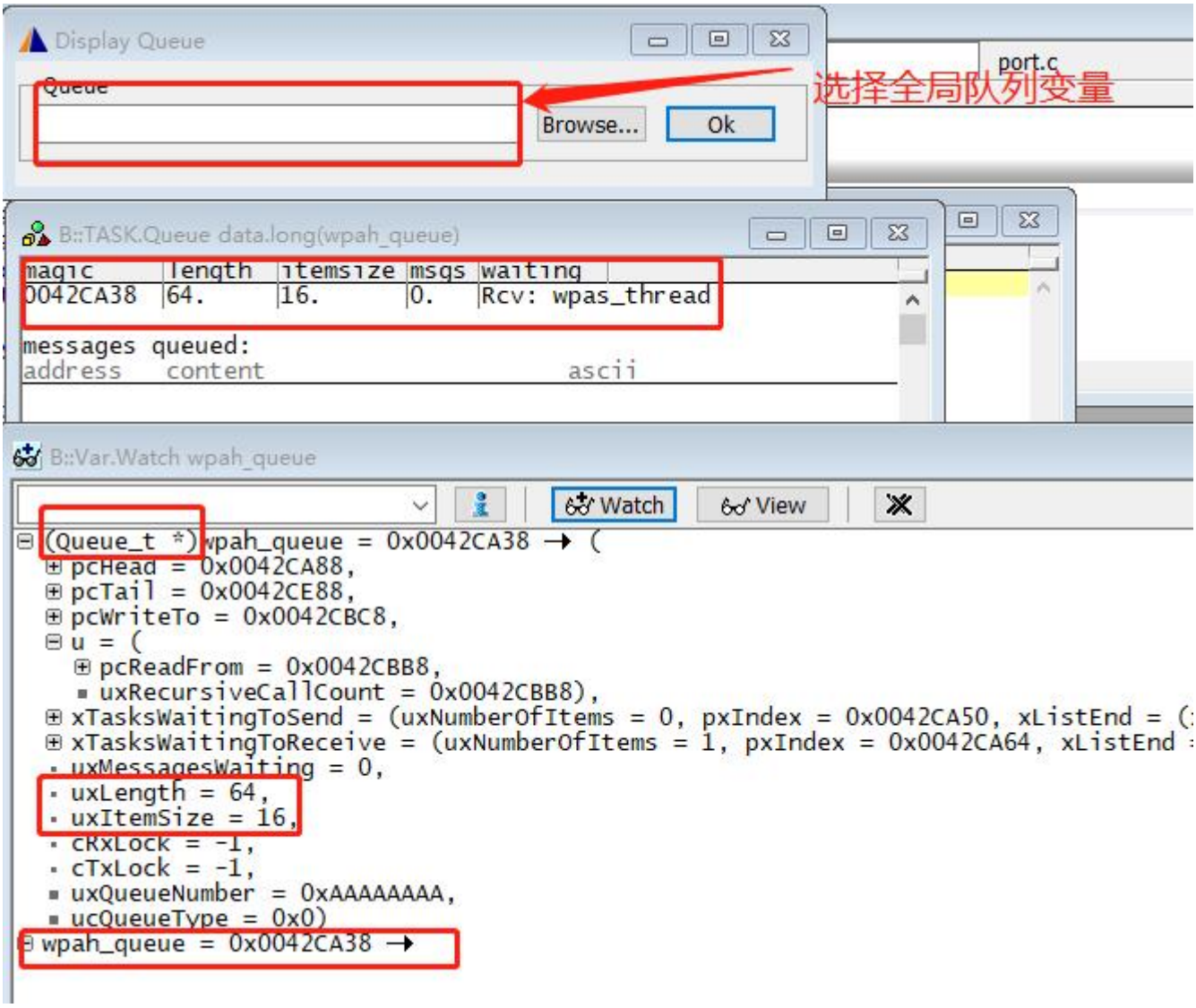
查看任务列表

TASK.TaskList



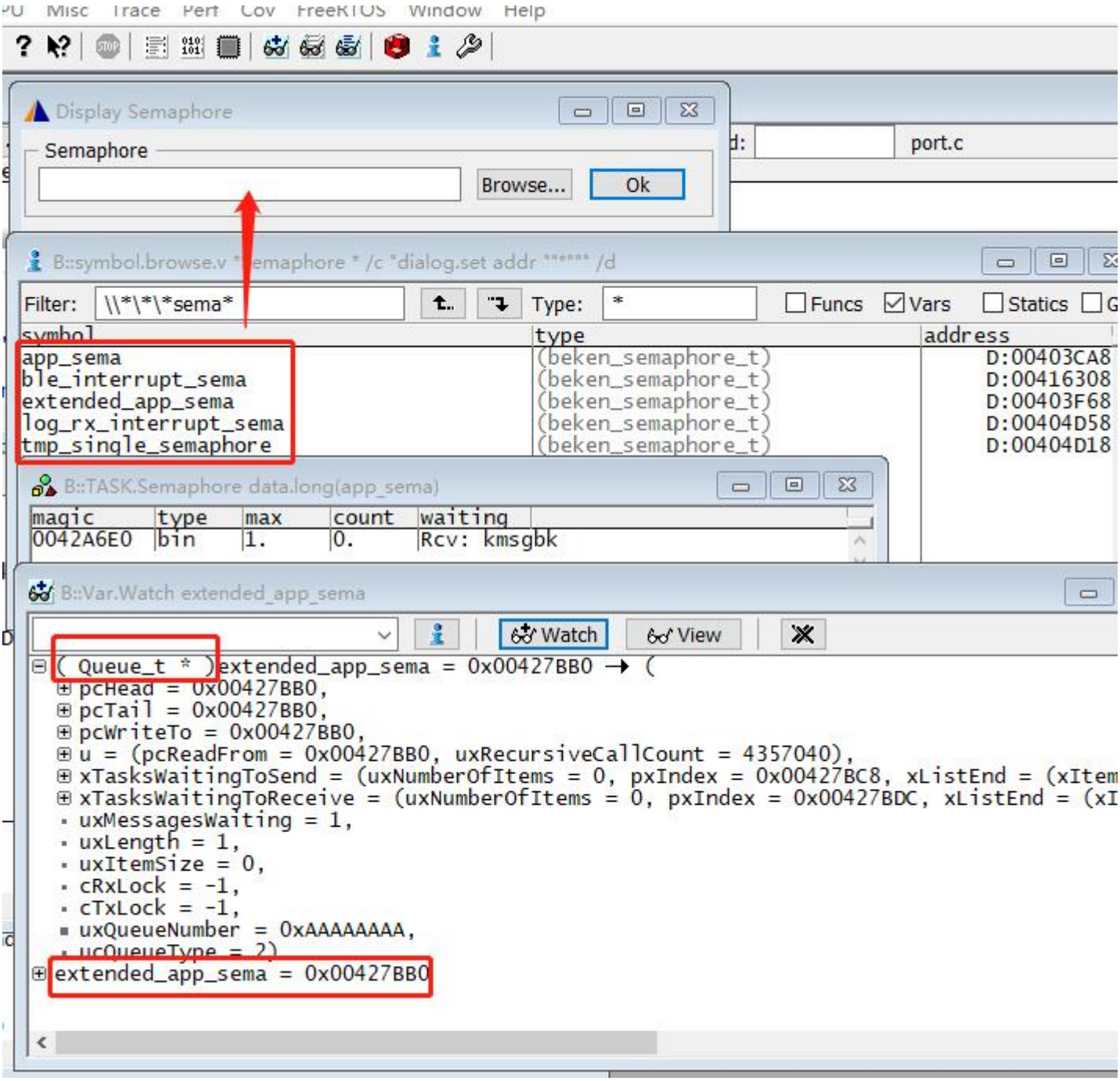
查看队列信息

TASK.Queue



查看信号量

TASK.Semaphore



其他

如果想一次性列出所有的 mutex 或者 queue 或者 semaphore，需要添加调试代码。

- 配置宏定义 **configQUEUE_REGISTRY_SIZE；**
- vQueueDelete（）的时候调用 **vQueueUnregisterQueue**（）；
- queue 等创建的时候调用 **vQueueAddToRegistry**（）；
  创建的时候在应用层去注册，信息或名称会更加准确。示例如下图：

```
/* Memory */
#define configDYNAMIC_HEAP_SIZE                    1
#define configTOTAL_HEAP_SIZE                      ( ( size_t )

/* Queue & Semaphore & Mutex */
#define configQUEUE_REGISTRY_SIZE                  64
#define configUSE_COUNTING_SEMAPHORES              1
#define configUSE_MUTEXES                          1

/* Utilities */
#define configUSE_TRACE_FACILITY                   1
#define configUSE_STATS_FORMATTING_FUNCTIONS       1
#define configUSE_ALTERNATIVE_API                  0
#define configCHECK_FOR_STACK_OVERFLOW             2
#define configGENERATE_RUN_TIME_STATS              0
#define configUSE_IDLE_SLEEP_HOOK                  ( 1 )
```

```
void vQueueDelete( QueueHandle_t xQueue )
{
Queue_t * const pxQueue = ( Queue_t * ) xQueue;

    configASSERT( pxQueue );
    traceQUEUE_DELETE( pxQueue );

    #if ( configQUEUE_REGISTRY_SIZE > 0 )
    {
        vQueueUnregisterQueue( pxQueue );
    }
    #endif

    #if( ( configSUPPORT_DYNAMIC_ALLOCATION == 1 ) && ( configSUPPORT_STA
    {
        /* The queue can only have been allocated dynamically - free it
        again. */
        vPortFree( pxQueue );
    }
```

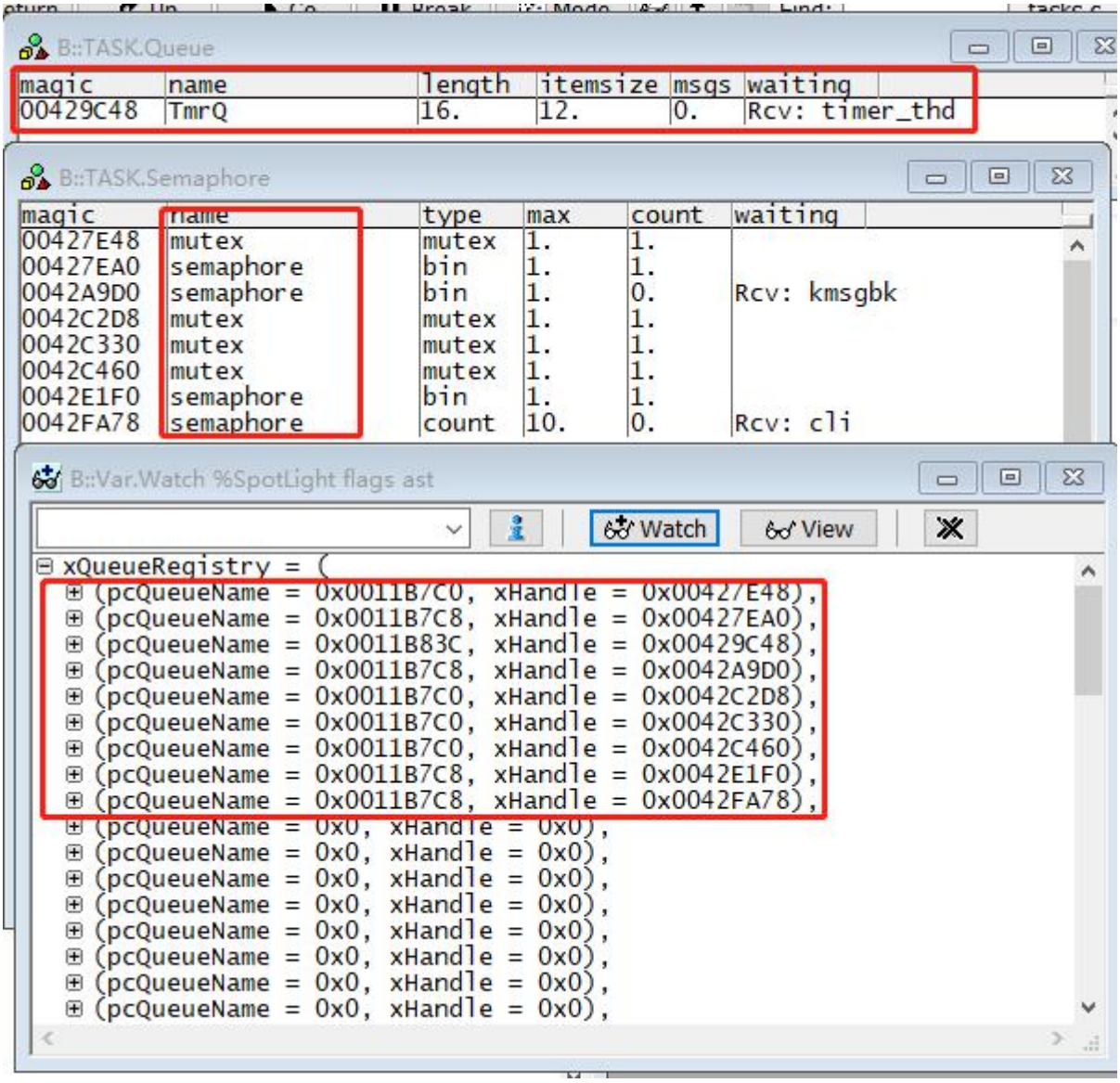
```
taskEXIT_CRITICAL
if( configSUPPORT_STATIC_ALLOCATION == 1 )
    xQueueGenericCreateStatic
endif
if( configSUPPORT_DYNAMIC_ALLOCATION == 1
    xQueueGenericCreate
endif
prvInitialiseNewQueue
if( configUSE_MUTEXES == 1 )
    prvInitialiseMutex
endif
if( ( configUSE_MUTEXES == 1 ) && ( configSUP
    xQueueCreateMutex
endif
```

```
#if( ( configUSE_MUTEXES == 1 ) && ( configSUPPORT_DYNAMIC_ALLOCATION == 1 ) )

    QueueHandle_t xQueueCreateMutex( const uint8_t ucQueueType )
    {
    Queue_t *pxNewQueue;
    const UBaseType_t uxMutexLength = ( UBaseType_t ) 1, uxMutexSize = ( UBaseType_t

        pxNewQueue = ( Queue_t * ) xQueueGenericCreate( uxMutexLength, uxMutexSize,
        prvInitialiseMutex( pxNewQueue );

#if ( configQUEUE_REGISTRY_SIZE > 0 )
        vQueueAddToRegistry( pxNewQueue, "mutex" );
#endif
        return pxNewQueue;
    }

#endif /* configUSE_MUTEXES */
/*-----------------------------------------------------------*/
```

最佳添加位置，如下图：

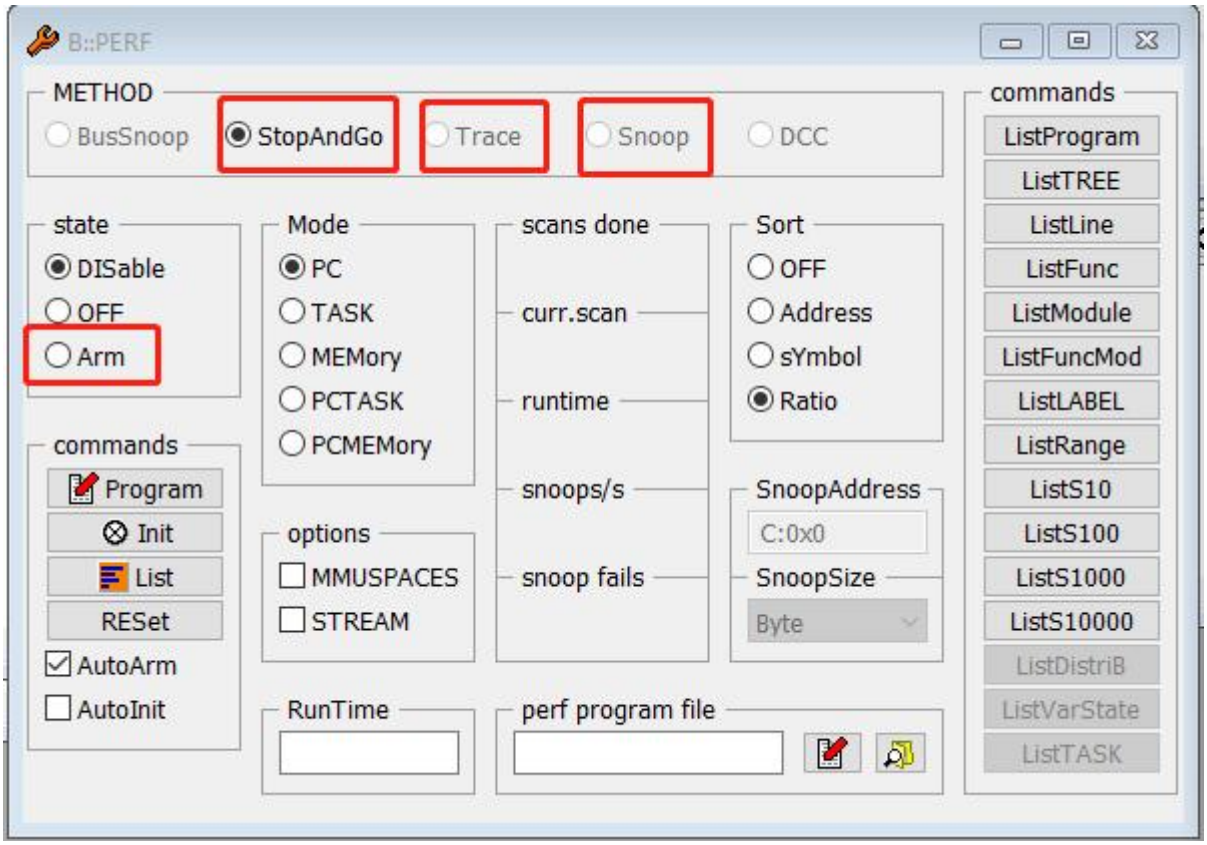
```c
BaseType_t xNetworkBuffersInitialise( void )
{
BaseType_t xReturn, x;

    /* Only initialise the buffers and their associated kernel objects if they
    have not been initialised before. */
    if( xNetworkBufferSemaphore == NULL )
    {
        xNetworkBufferSemaphore = xSemaphoreCreateCounting( ipconfigNUM_NETWORK_BUFFER_DE
        configASSERT( xNetworkBufferSemaphore );

        if( xNetworkBufferSemaphore != NULL )
        {
            #if ( configQUEUE_REGISTRY_SIZE > 0 )
            {
                vQueueAddToRegistry( xNetworkBufferSemaphore, "NetBufSem" );
            }
            #endif /* configQUEUE_REGISTRY_SIZE */

            /* If the trace recorder code is included name the semaphore for viewing
            in FreeRTOS+Trace.  */
            #if( ipconfigINCLUDE_EXAMPLE_FREERTOS_PLUS_TRACE_CALLS == 1 )
            {
                extern QueueHandle_t xNetworkEventQueue;
                vTraceSetQueueName( xNetworkEventQueue, "IPStackEvent" );
                vTraceSetQueueName( xNetworkBufferSemaphore, "NetworkBufferCount" );
            }
            #endif /*  ipconfigINCLUDE_EXAMPLE_FREERTOS_PLUS_TRACE_CALLS == 1 */
```

```c
#if( ( configUSE_COUNTING_SEMAPHORES == 1 ) && ( configSUPPORT_DYNAMIC_ALLOCATION == 1 )

    QueueHandle_t xQueueCreateCountingSemaphore( const UBaseType_t uxMaxCount, co
    {
    QueueHandle_t xHandle;

        configASSERT( uxMaxCount != 0 );
        configASSERT( uxInitialCount <= uxMaxCount );

        xHandle = xQueueGenericCreate( uxMaxCount, queueSEMAPHORE_QUEUE_ITEM_LENGTH, queu

        if( xHandle != NULL )
        {
            ( ( Queue_t * ) xHandle )->uxMessagesWaiting = uxInitialCount;

#if ( configQUEUE_REGISTRY_SIZE > 0 )
            vQueueAddToRegistry( xHandle, "semaphore" );
#endif
            traceCREATE_COUNTING_SEMAPHORE();
        }
        else
        {
            traceCREATE_COUNTING_SEMAPHORE_FAILED();
        }

        return xHandle;
    } « end xQueueCreateCountingSemaphore »

#endif /* ( ( configUSE_COUNTING_SEMAPHORES == 1 ) && ( configSUPPORT_DYNAMIC_ALLOCATION
```

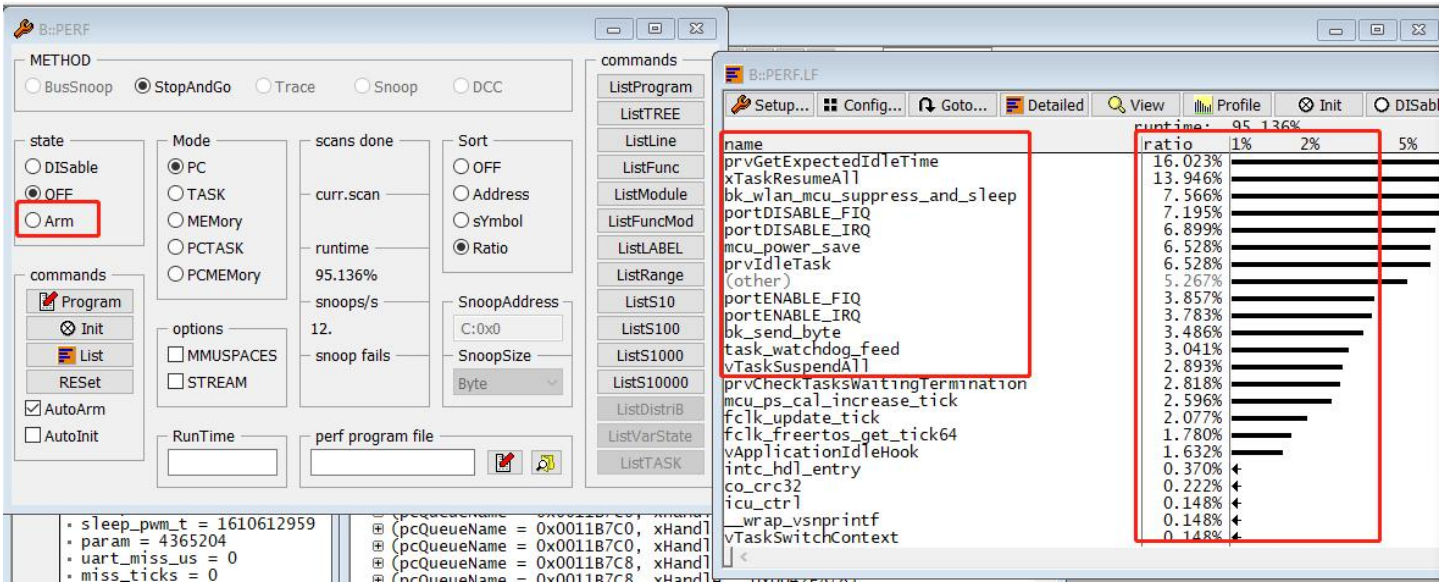


性能分析

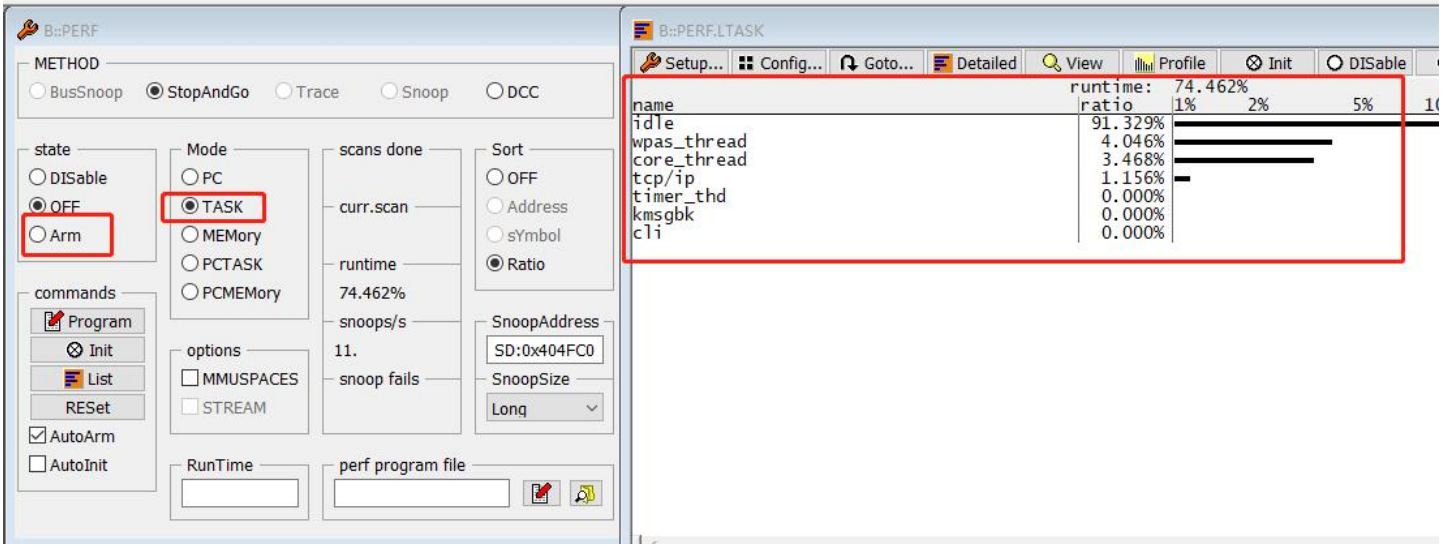
TRACE32 性能分析可以了解函数的运行时间、任务的运行时间以及变量的相关情况等。性能分析方式有以下几种：Snoop，Trace，StopAndGo，DCC，BusSnoop。Snoop 方式需要 CPU 支持，Trace 虽然是软件功能，但是是个增值服务。一般选择 StopAndGo，它会影响 cpu 的正常运行。


B::PERF
METHOD
BusSnoop
StopAndGo
Trace
Snoop
DCC
commands
ListProgram
ListTREE
ListLine
ListFunc
ListModule
ListFuncMod
ListLABEL
ListRange
ListS10
ListS100
ListS1000
ListS10000
ListDistriB
ListVarState
ListTASK
state
DISable
OFF
Arm
Mode
PC
TASK
MEMory
PCTASK
PCMEMory
scans done
curr.scan
runtime
snoops/s
snoop fails
Sort
OFF
Address
sYmbol
Ratio
commands
Program
Init
List
RESet
AutoArm
AutoInit
options
MMUSPACES
STREAM
SnoopAddress
C:0x0
SnoopSize
Byte
RunTime
perf program file

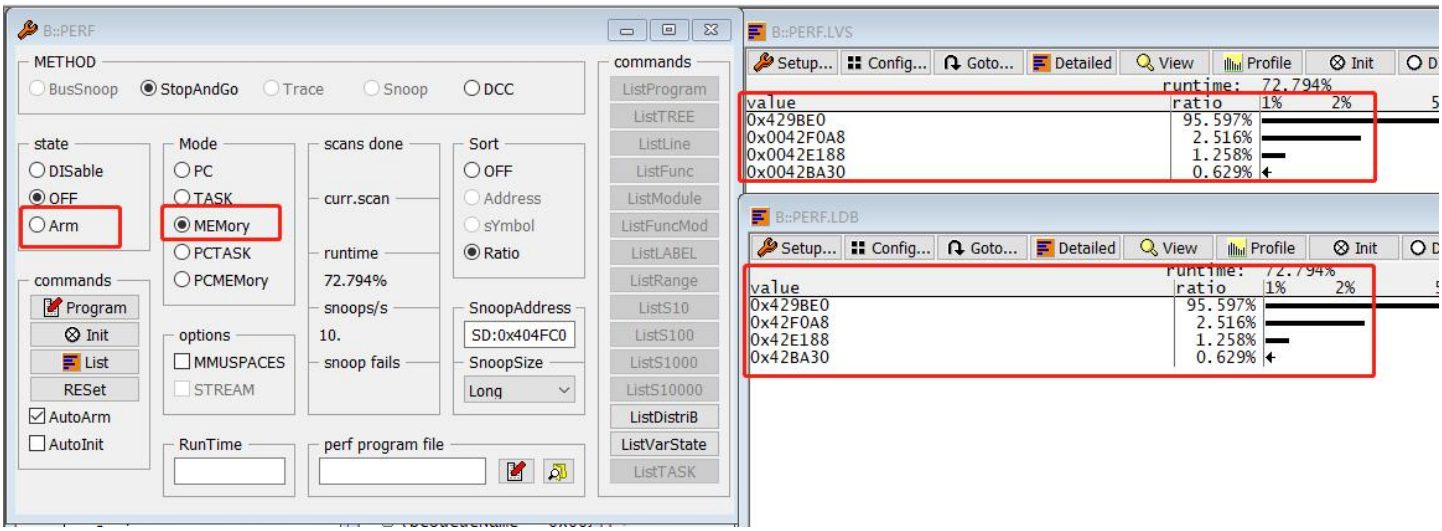
PC Mode


B::PERF
METHOD
BusSnoop
StopAndGo
Trace
Snoop
DCC
commands
ListProgram
ListTREE
ListLine
ListFunc
ListModule
ListFuncMod
ListLABEL
ListRange
ListS10
ListS100
ListS1000
ListS10000
ListDistriB
ListVarState
ListTASK
state
DISable
OFF
Arm
Mode
PC
TASK
MEMory
PCTASK
PCMEMory
scans done
curr.scan
runtime
95.136%
snoops/s
12.
snoop fails
Sort
OFF
Address
sYmbol
Ratio
commands
Program
Init
List
RESet
AutoArm
AutoInit
options
MMUSPACES
STREAM
SnoopAddress
C:0x0
SnoopSize
Byte
RunTime
perf program file
B::PERF.LF
Setup...
Config...
Goto...
Detailed
View
Profile
Init
runtime: 95.136%
name
ratio
1%
2%
5%
prvGetExpectedIdleTime 16.023%
xTaskResumeAll 13.946%
bk_wlan_mcu_suppress_and_sleep 7.566%
portDISABLE_FIQ 7.195%
portDISABLE_IRQ 6.899%
mcu_power_save 6.528%
prvIdleTask 6.528%
(other) 5.267%
portENABLE_FIQ 3.857%
portENABLE_IRQ 3.783%
bk_send_byte 3.486%
task_watchdog_feed 3.041%
vTaskSuspendAll 2.893%
prvCheckTasksWaitingTermination 2.818%
mcu_ps_cal_increase_tick 2.596%
fclk_update_tick 2.077%
fclk_freertos_get_tick64 1.780%
vApplicationIdleHook 1.632%
intc_hdl_entry 0.370%
co_crc32 0.222%
icu_ctrl 0.148%
__wrap_vsnprintf 0.148%
vTaskSwitchContext 0.148%
sleep_pwm_t = 1610612959
param = 4365204
uart_miss_us = 0
miss_ticks = 0

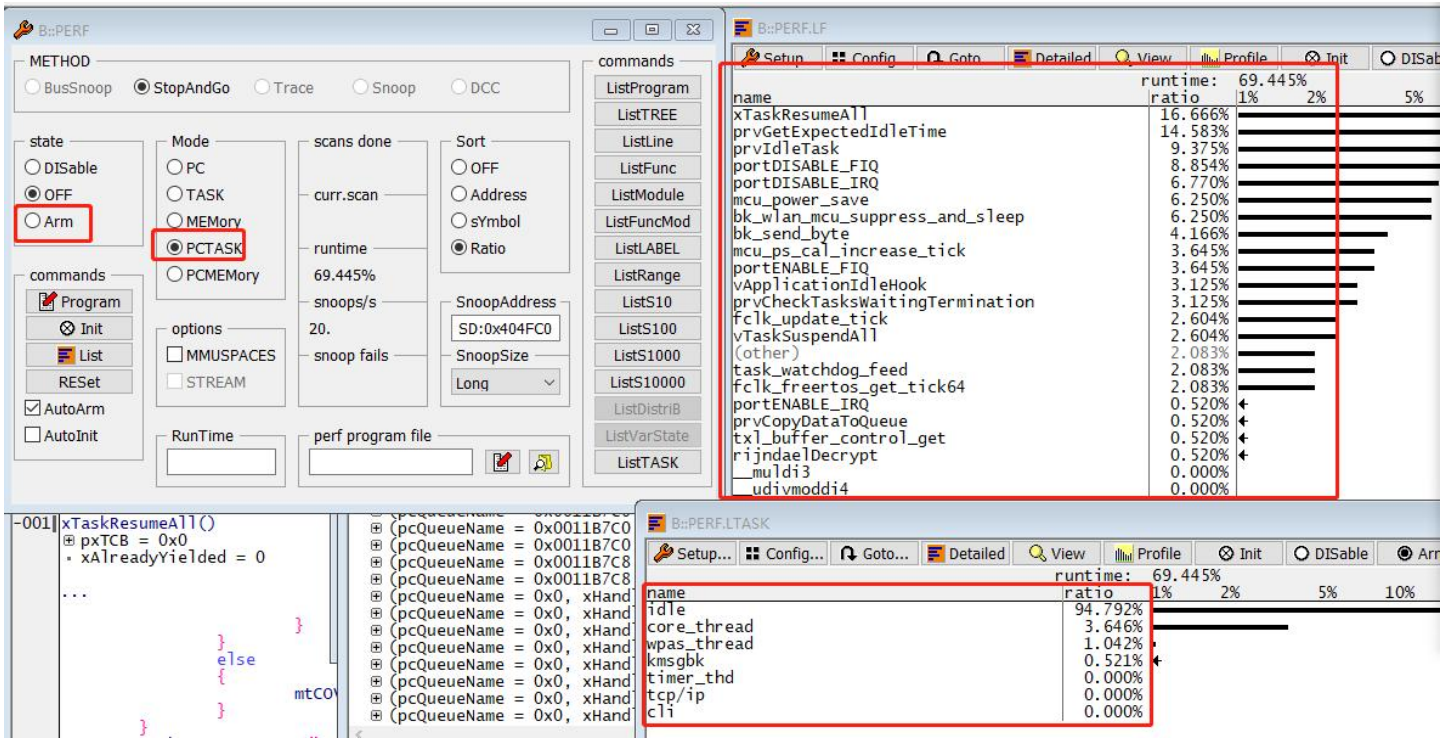
TASK Mode



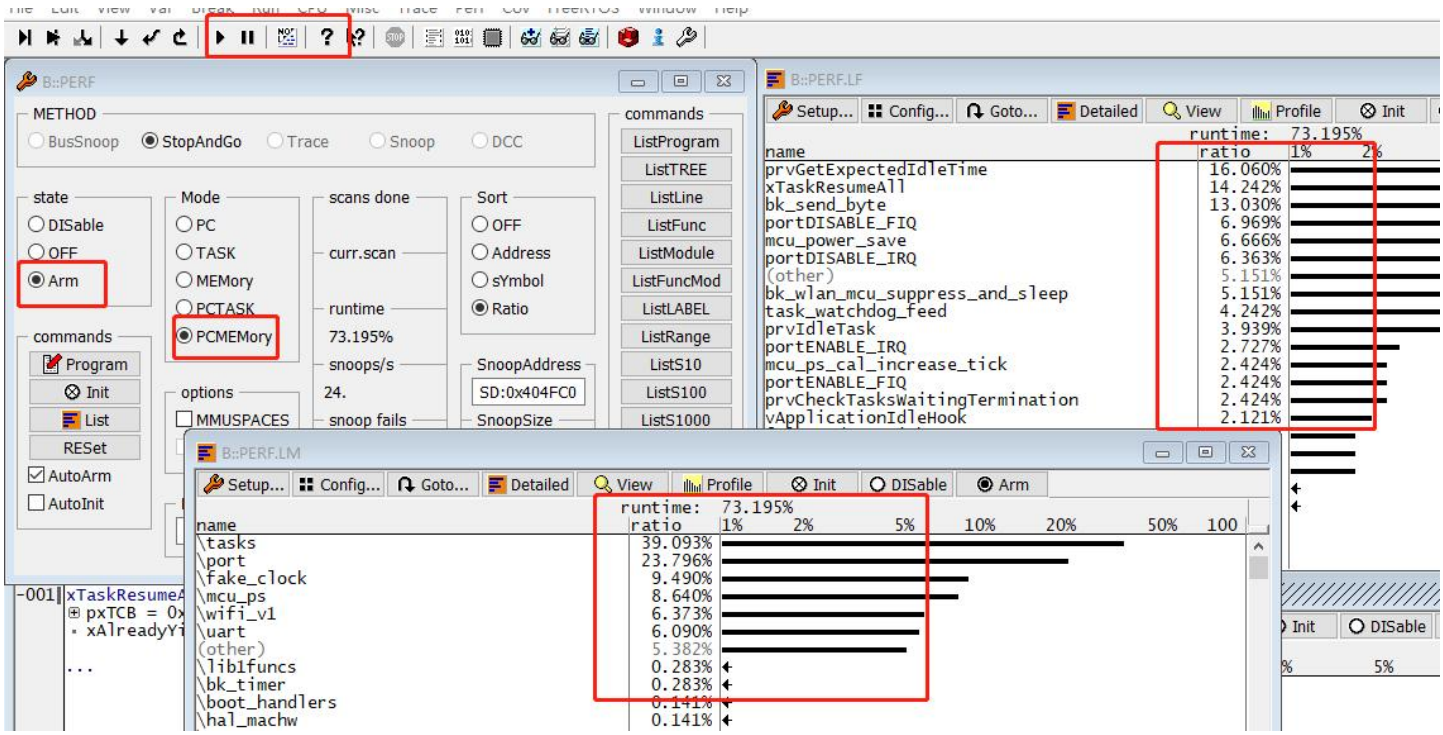
MEMory Mode



PCTASK Mode



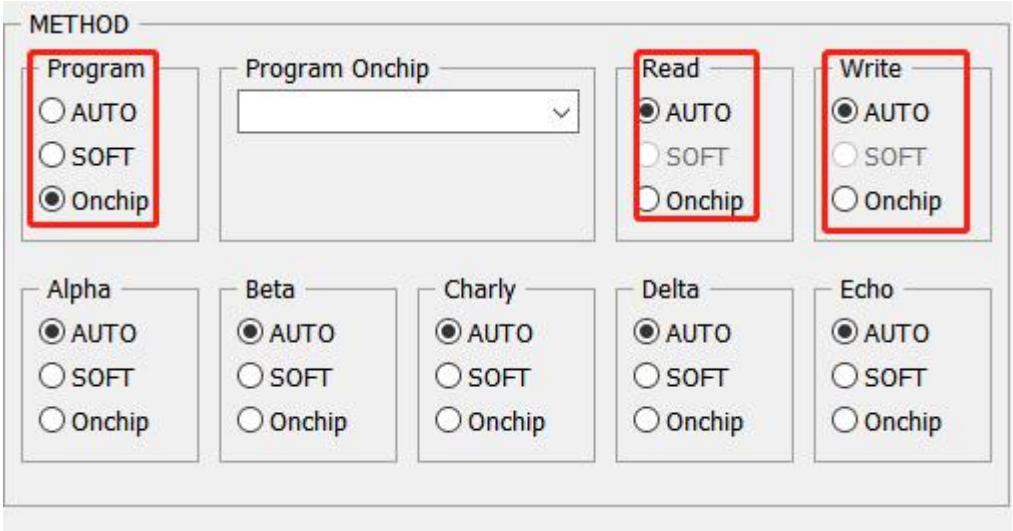
PCMEMory Mode



断点进阶设置

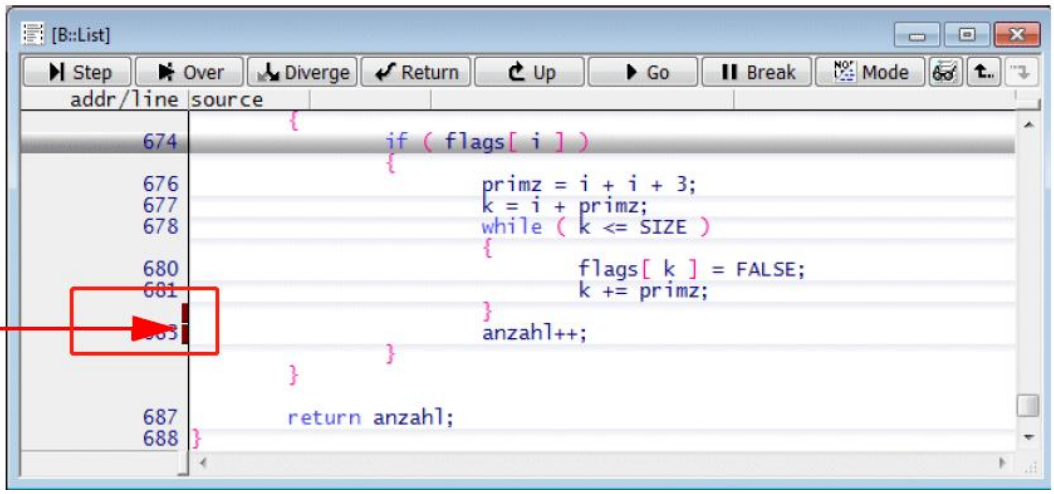
断点类型

程序断点和读写断点，这两种断点实现是否是软断点或者硬断点，看配置。

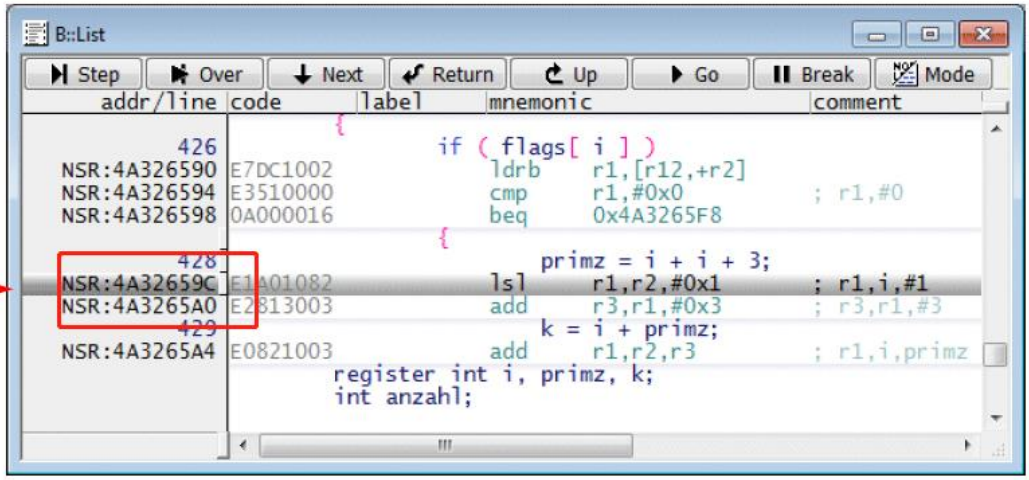


程序断点

参考相关章节，如下图：



Set a Program breakpoint by a left mouse double-click to the instruction



Disable the Program breakpoint by a left mouse double-click to the red program breakpoint indicator. The program breakpoint indicator becomes grey.

Tips

如果有足够多的断点，或者说 bk72xx 支持软件断点的话，可以通过这样去设置程序断点：

Break.SetHll 0x10000—0x0xffff　　// 对程序**空间地址范围**设置断点

Break.SetHll /dhcp　　// 对**模块或者文件**设置范围断点

对于**单个函数**，可以通过命令行形式直接设置断点：

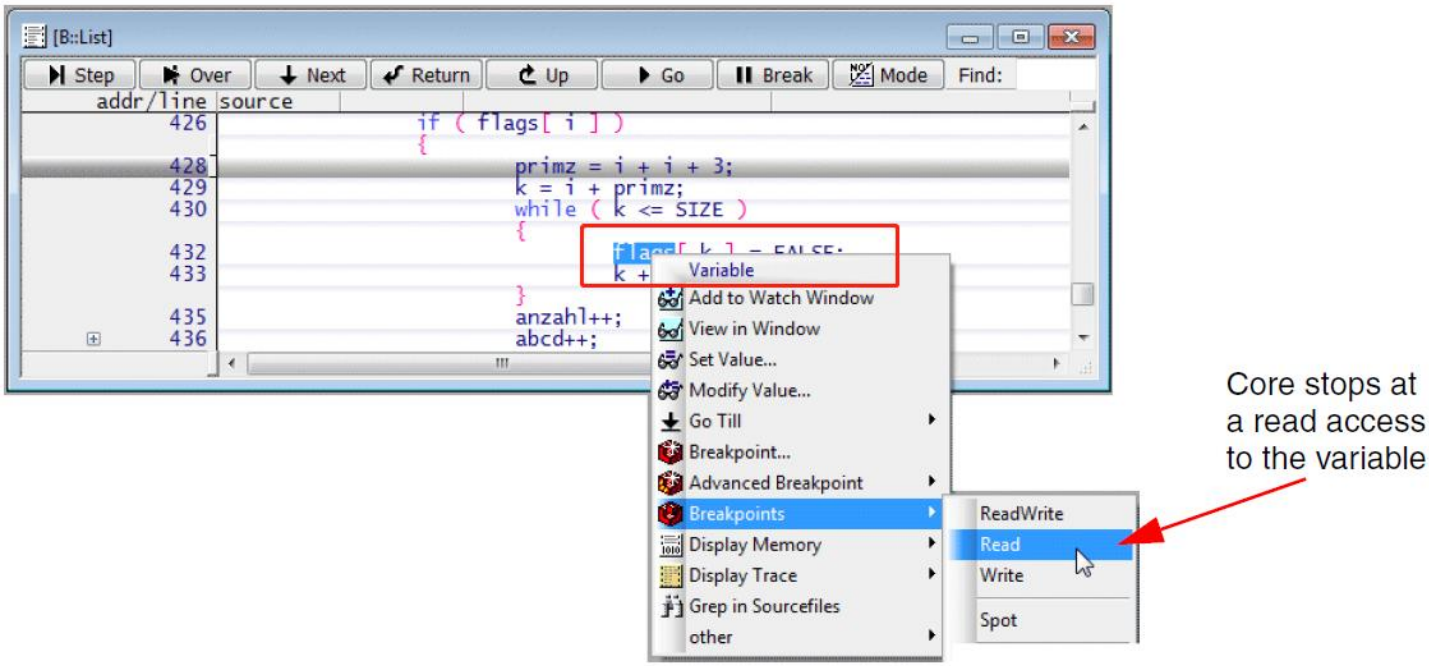
Break.SetHll entry_main

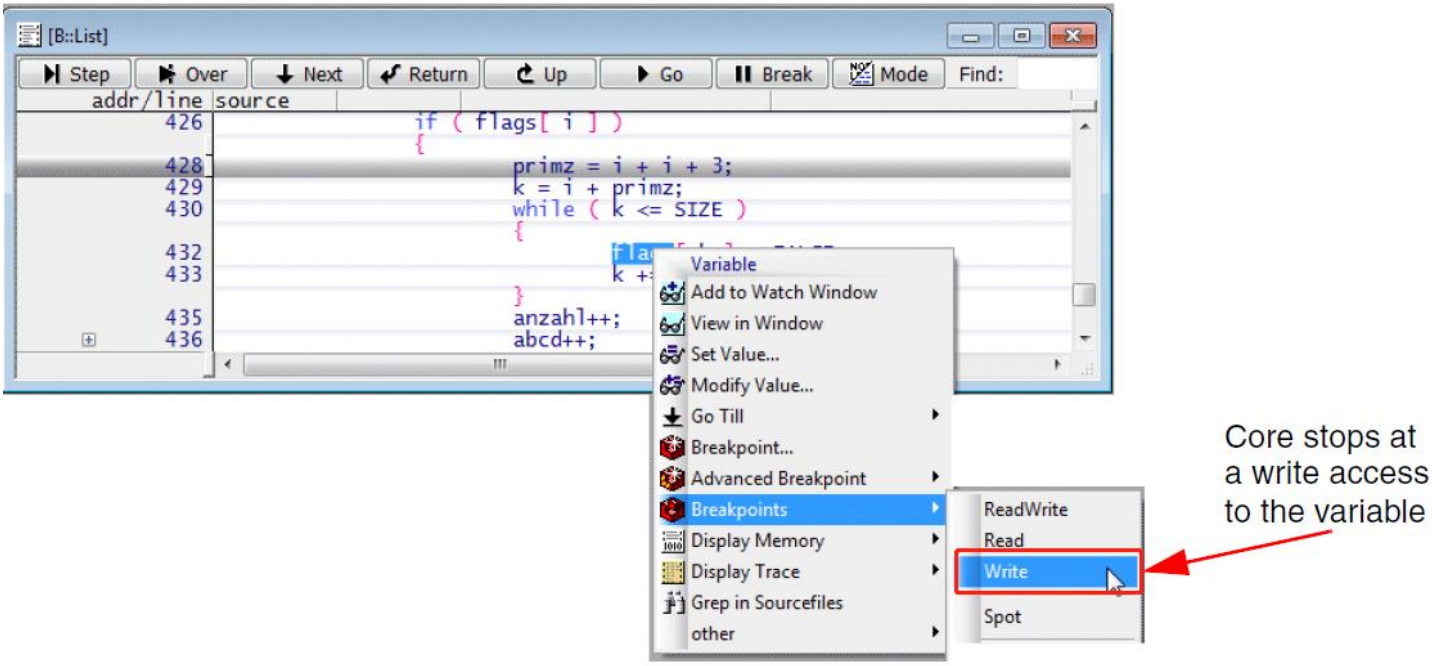
对于某一行或多行设置断点：

Break.SetLine，对于这个功能，我们的芯片不适用

读写断点

参考相关章节，如下图：





HARD 断点

由 TRACE32 提供，只有 TRACE32-FIRE 支持

SOFT 断点

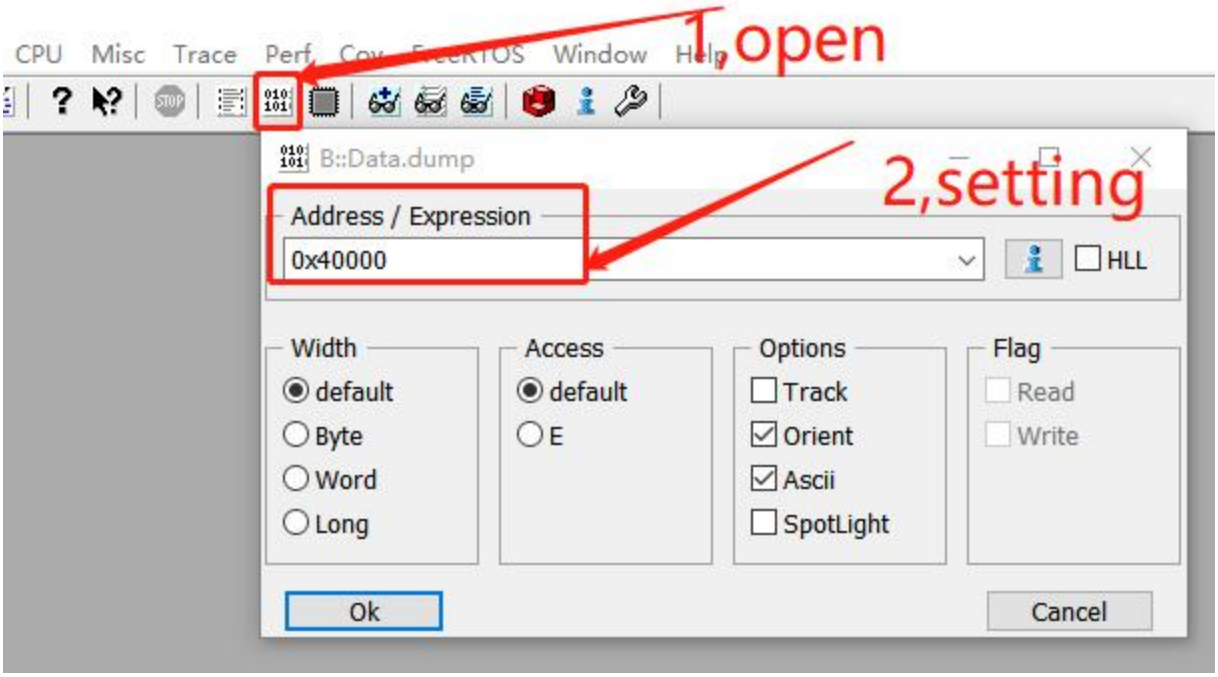
Breakpoint 指令替换完成

ONCHIP 断点

芯片提供

内存断点设置

单一断点的设置



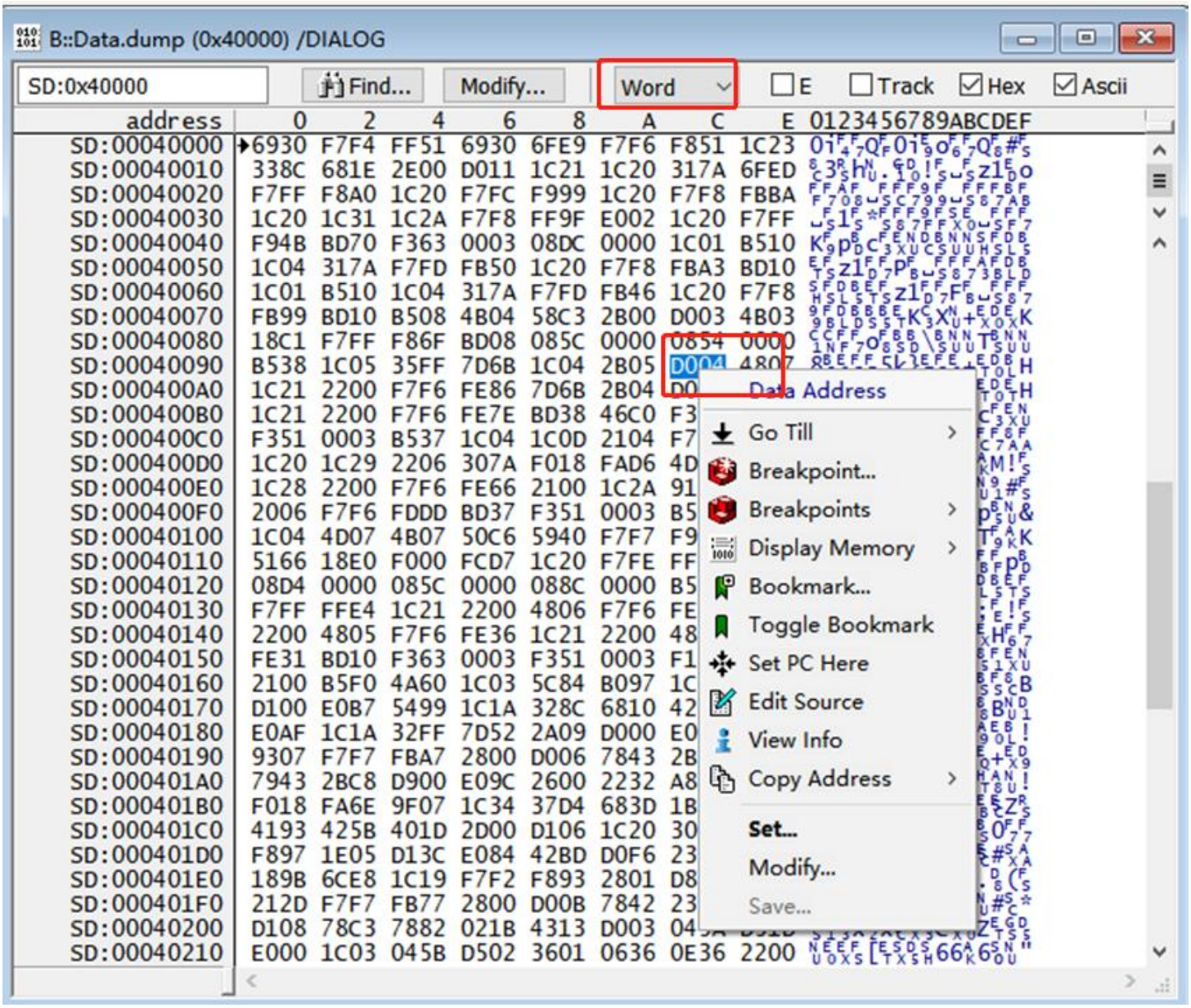
选择单一数据内容，右键配置

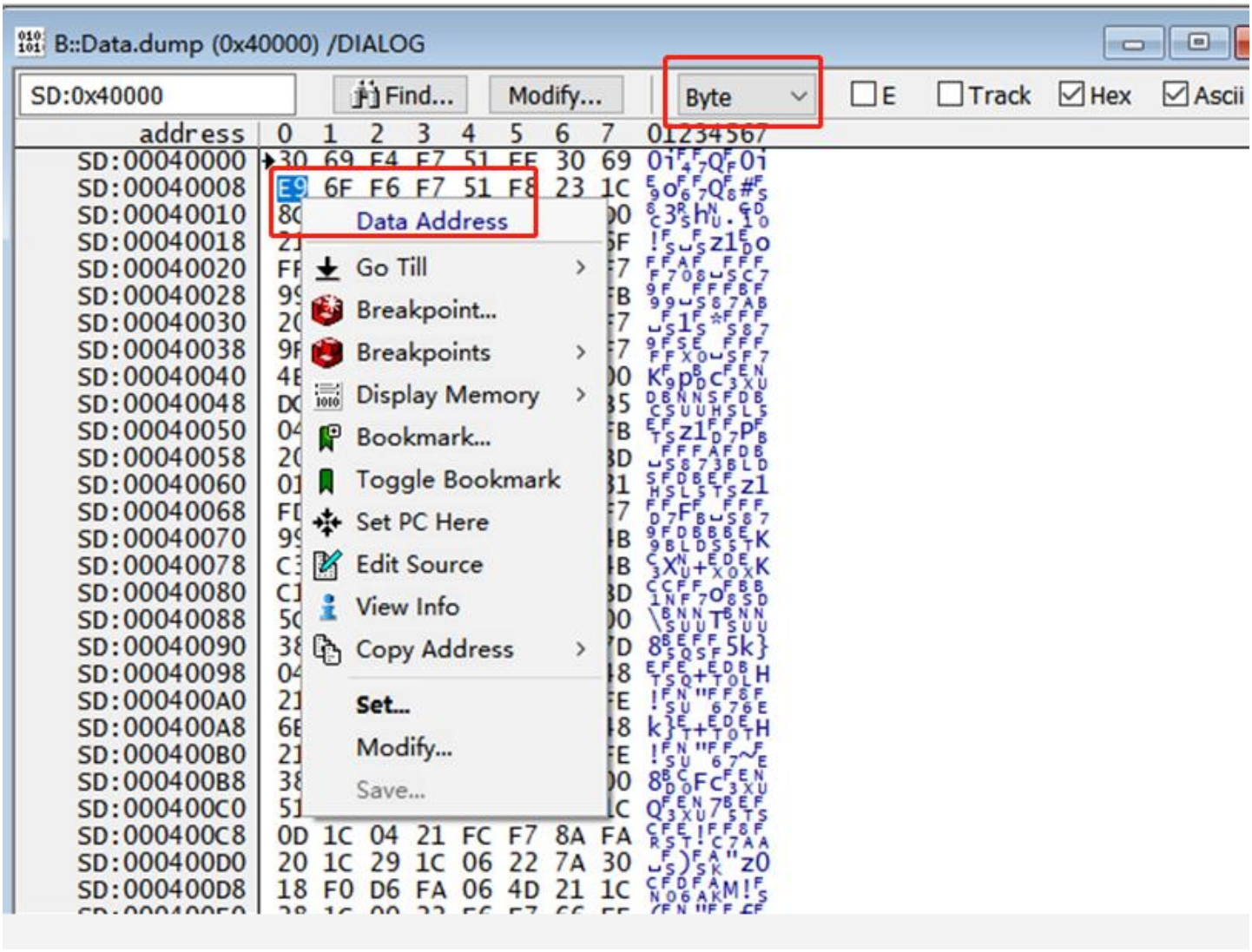
B::Data.dump (0x40000) /DIALOG

SD:0x40000 | Find... | Modify... | Long | E | Track | Hex | Ascii

address	0	4	8	C	0123456789ABCDEF
SD:00040000	F7F46930	6930FF51	F7F66FE9	1C23F851	
SD:00040010	681E338C	D0112E00	1C201C21	6FED317A	
SD:00040020	F8A0F7FF	F7FC1C20	1C20F000		
SD:00040030	1C311C20	F7F81C2A	E0		
SD:00040040	BD70F94B	0003F363	00		
SD:00040050	317A1C04	FB50F7FD	F7		
SD:00040060	B5101C01	317A1C04	FB		
SD:00040070	BD10FB99	4B04B508	2B		
SD:00040080	F7FF18C1	BD08F86F	00		
SD:00040090	1C05B538	7D6B35FF	2B		
SD:000400A0	22001C21	FE86F7F6	2B		
SD:000400B0	22001C21	FE7EF7F6	46		
SD:000400C0	0003F351	1C04B537	21		
SD:000400D0	1C291C20	307A2206	FA		
SD:000400E0	22001C28	FE66F7F6	1C		
SD:000400F0	F7F62006	BD37FDDD	00		
SD:00040100	4D071C04	50C64B07	F7		
SD:00040110	18E05166	FCD7F000	F7		
SD:00040120	000008D4	0000085C	00		
SD:00040130	FFE4F7FF	22001C21	F7		
SD:00040140	48052200	FE36F7F6	22		
SD:00040150	BD10FE31	0003F363	00		
SD:00040160	B5F02100	1C034A60	B0		
SD:00040170	E0B7D100	1C1A5499	68		
SD:00040180	1C1AE0AF	7D5232FF	D0		
SD:00040190	F7F79307	2800FBA7	784		
SD:000401A0	2BC87943	E09CD900	22322600	2100A809	
SD:000401B0	FA6EF018	1C349F07	683D37D4	1E5A1BEB	
SD:000401C0	425B4193	2D00401D	1C20D106	F7F73008	
SD:000401D0	1E05F897	E084D13C	D0F642BD	AA02231B	
SD:000401E0	6CE8189B	F7F21C19	2801F893	1C28D82E	
SD:000401F0	F7F7212D	2800FB77	7842D00B	2A1A2300	
SD:00040200	78C3D108	021B7882	D0034313	D51D045A	
SD:00040210	1C03E000	D502045B	06363601	22000E36	

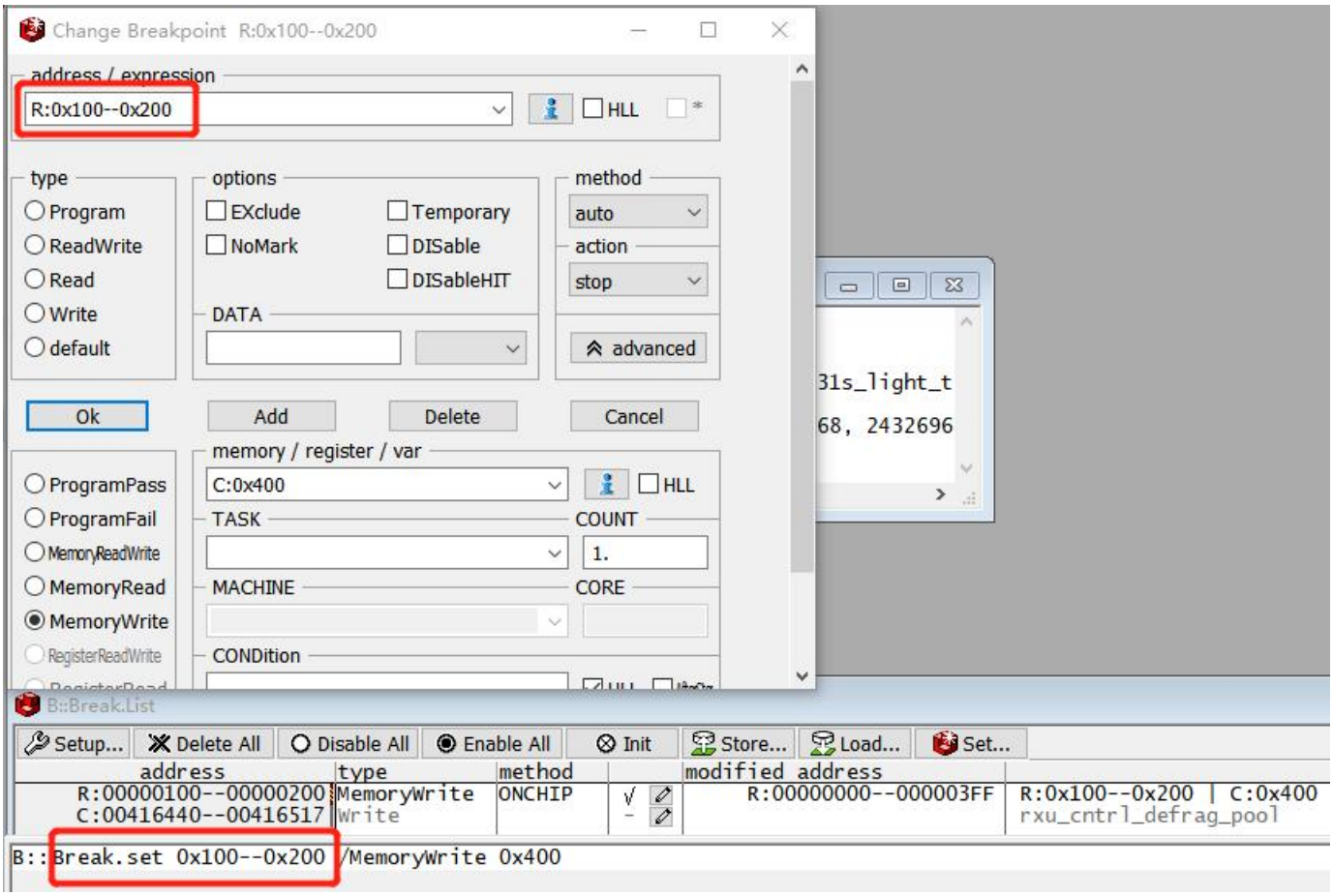
- Data Address
- Go Till
- Breakpoint...
- Breakpoints
- Display Memory
- Bookmark...
- Toggle Bookmark
- Set PC Here
- Edit Source
- View Info
- Copy Address
- **Set...**
- Modify...
- Save...


B::Data.dump (0x40000) /DIALOG
SD:0x40000
Find...
Modify...
Word
E
Track
Hex
Ascii
address
0 2 4 6 8 A C E 0123456789ABCDEF
SD:00040000 6930 F7F4 FF51 6930 6FE9 F7F6 F851 1C23
SD:00040010 338C 681E 2E00 D011 1C21 1C20 317A 6FED
SD:00040020 F7FF F8A0 1C20 F7FC F999 1C20 F7F8 FBBA
SD:00040030 1C20 1C31 1C2A F7F8 FF9F E002 1C20 F7FF
SD:00040040 F94B BD70 F363 0003 08DC 0000 1C01 B510
SD:00040050 1C04 317A F7FD FB50 1C20 F7F8 FBA3 BD10
SD:00040060 1C01 B510 1C04 317A F7FD FB46 1C20 F7F8
SD:00040070 FB99 BD10 B508 4B04 58C3 2B00 D003 4B03
SD:00040080 18C1 F7FF F86F BD08 085C 0000 0854 0000
SD:00040090 B538 1C05 35FF 7D6B 1C04 2B05 D004 4807
SD:000400A0 1C21 2200 F7F6 FE86 7D6B 2B04 D0
SD:000400B0 1C21 2200 F7F6 FE7E BD38 46C0 F3
SD:000400C0 F351 0003 B537 1C04 1C0D 2104 F7
SD:000400D0 1C20 1C29 2206 307A F018 FAD6 4D
SD:000400E0 1C28 2200 F7F6 FE66 2100 1C2A 91
SD:000400F0 2006 F7F6 FDDD BD37 F351 0003 B5
SD:00040100 1C04 4D07 4B07 50C6 5940 F7F7 F9
SD:00040110 5166 18E0 F000 FCD7 1C20 F7FE FF
SD:00040120 08D4 0000 085C 0000 088C 0000 B5
SD:00040130 F7FF FFE4 1C21 2200 4806 F7F6 FE
SD:00040140 2200 4805 F7F6 FE36 1C21 2200 48
SD:00040150 FE31 BD10 F363 0003 F351 0003 F1
SD:00040160 2100 B5F0 4A60 1C03 5C84 B097 1C
SD:00040170 D100 E0B7 5499 1C1A 328C 6810 42
SD:00040180 E0AF 1C1A 32FF 7D52 2A09 D000 E0
SD:00040190 9307 F7F7 FBA7 2800 D006 7843 2B
SD:000401A0 7943 2BC8 D900 E09C 2600 2232 A8
SD:000401B0 F018 FA6E 9F07 1C34 37D4 683D 1B
SD:000401C0 4193 425B 401D 2D00 D106 1C20 30
SD:000401D0 F897 1E05 D13C E084 42BD D0F6 23
SD:000401E0 189B 6CE8 1C19 F7F2 F893 2801 D8
SD:000401F0 212D F7F7 FB77 2800 D00B 7842 23
SD:00040200 D108 78C3 7882 021B 4313 D003 04
SD:00040210 E000 1C03 045B D502 3601 0636 0E36 2200
Data Address
Go Till
Breakpoint...
Breakpoints
Display Memory
Bookmark...
Toggle Bookmark
Set PC Here
Edit Source
View Info
Copy Address
Set...
Modify...
Save...

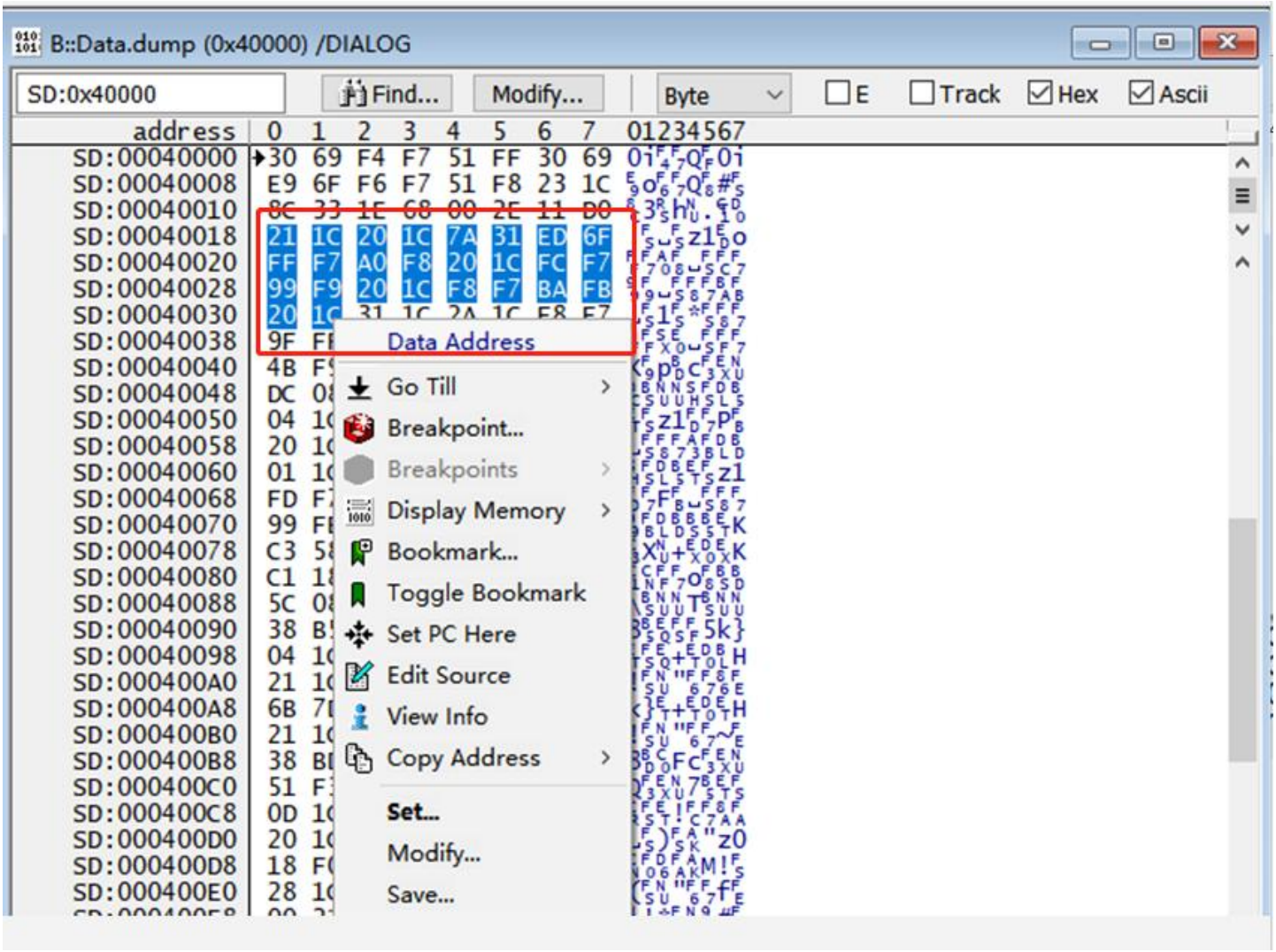


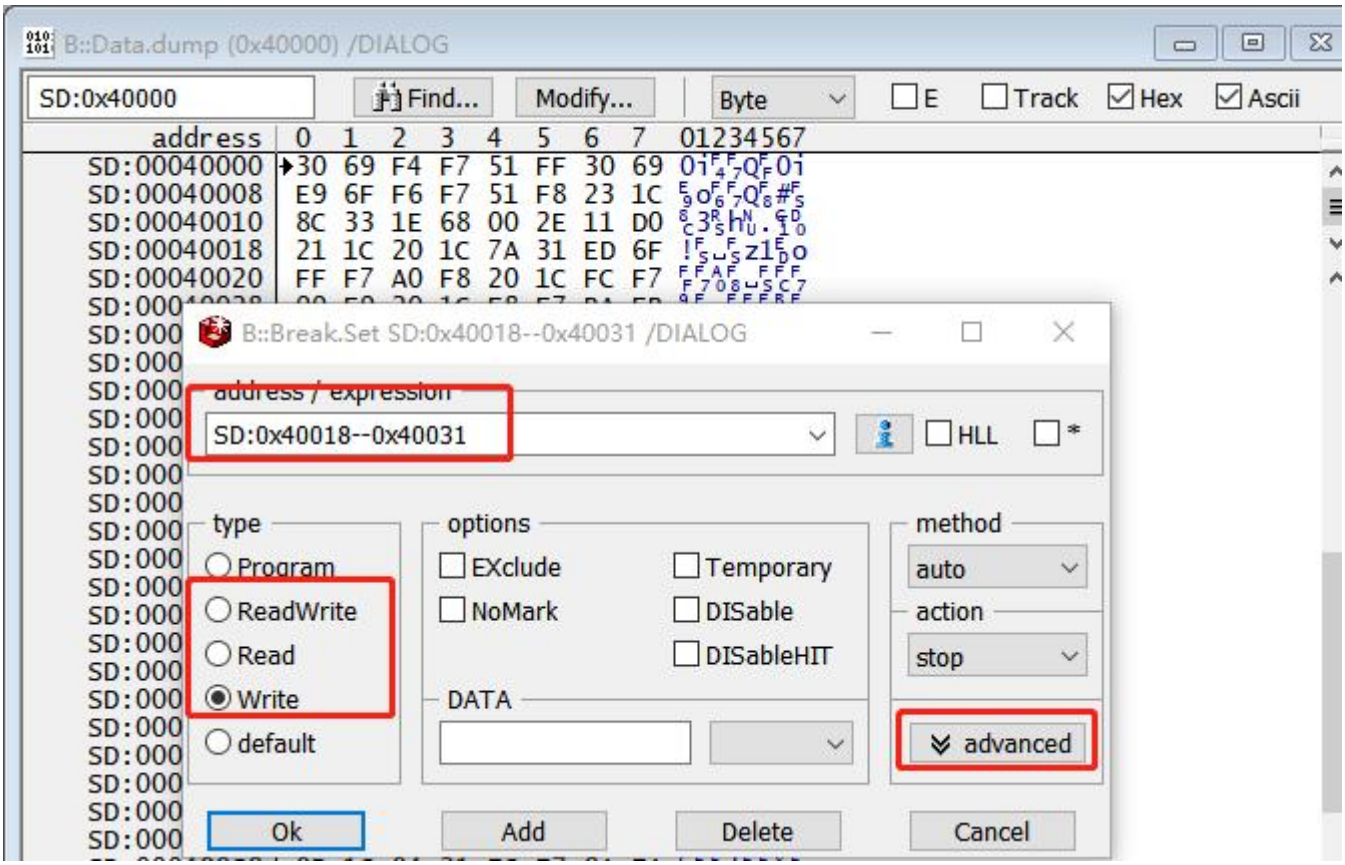
范围断点的设置

命令设置


Change Breakpoint R:0x100--0x200
address / expression
R:0x100--0x200
HLL
type
Program
ReadWrite
Read
Write
default
options
EXclude
Temporary
NoMark
DISable
DISableHIT
DATA
method
auto
action
stop
advanced
Ok
Add
Delete
Cancel
memory / register / var
C:0x400
HLL
ProgramPass
ProgramFail
MemoryReadWrite
MemoryRead
MemoryWrite
RegisterReadWrite
TASK
COUNT
1.
MACHINE
CORE
CONDition
B::Break.List
Setup...
Delete All
Disable All
Enable All
Init
Store...
Load...
Set...
address
type
method
modified address
R:00000100--00000200
MemoryWrite
ONCHIP
R:00000000--000003FF
R:0x100--0x200 | C:0x400
C:00416440--00416517
Write
rxu_cntrl_defrag_pool
B::Break.set 0x100--0x200 /MemoryWrite 0x400

界面设置

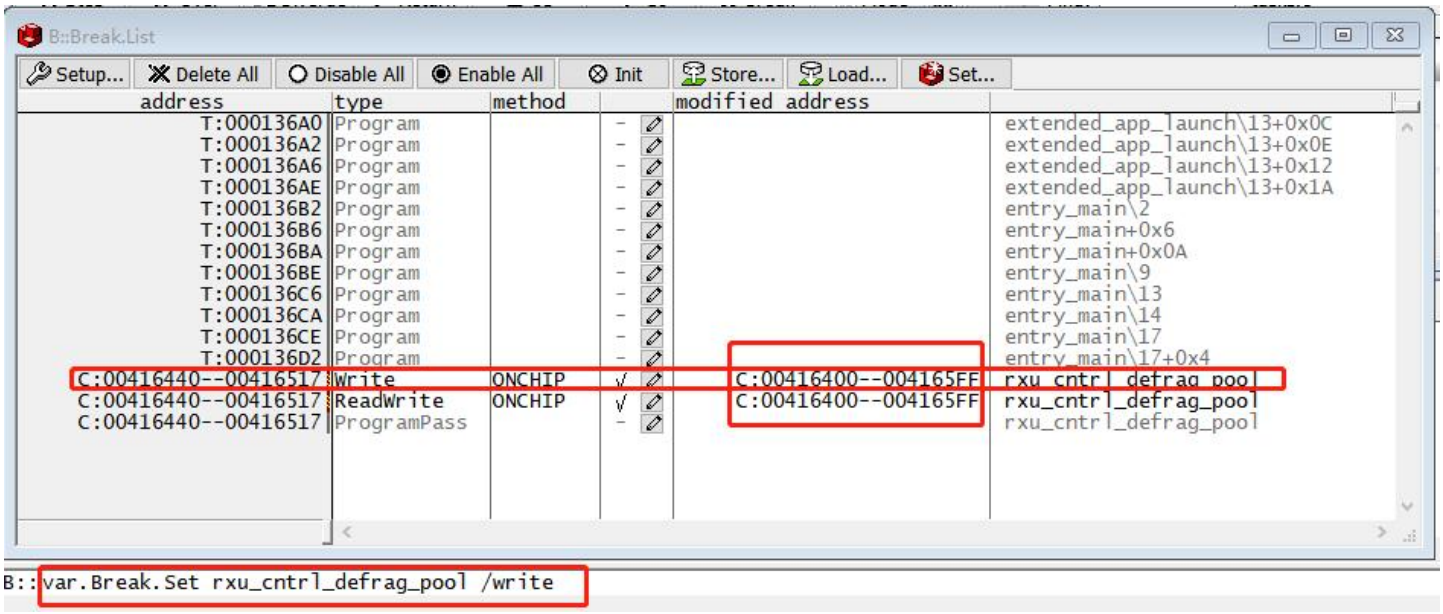

B::Data.dump (0x40000) /DIALOG
SD:0x40000
Find...
Modify...
Byte
E
Track
Hex
Ascii
address
0 1 2 3 4 5 6 7
01234567
SD:00040000
SD:00040008
SD:00040010
SD:00040018
SD:00040020
SD:00040028
SD:00040030
SD:00040038
SD:00040040
SD:00040048
SD:00040050
SD:00040058
SD:00040060
SD:00040068
SD:00040070
SD:00040078
SD:00040080
SD:00040088
SD:00040090
SD:00040098
SD:000400A0
SD:000400A8
SD:000400B0
SD:000400B8
SD:000400C0
SD:000400C8
SD:000400D0
SD:000400D8
SD:000400E0
Data Address
Go Till
Breakpoint...
Breakpoints
Display Memory
Bookmark...
Toggle Bookmark
Set PC Here
Edit Source
View Info
Copy Address
Set...
Modify...
Save...



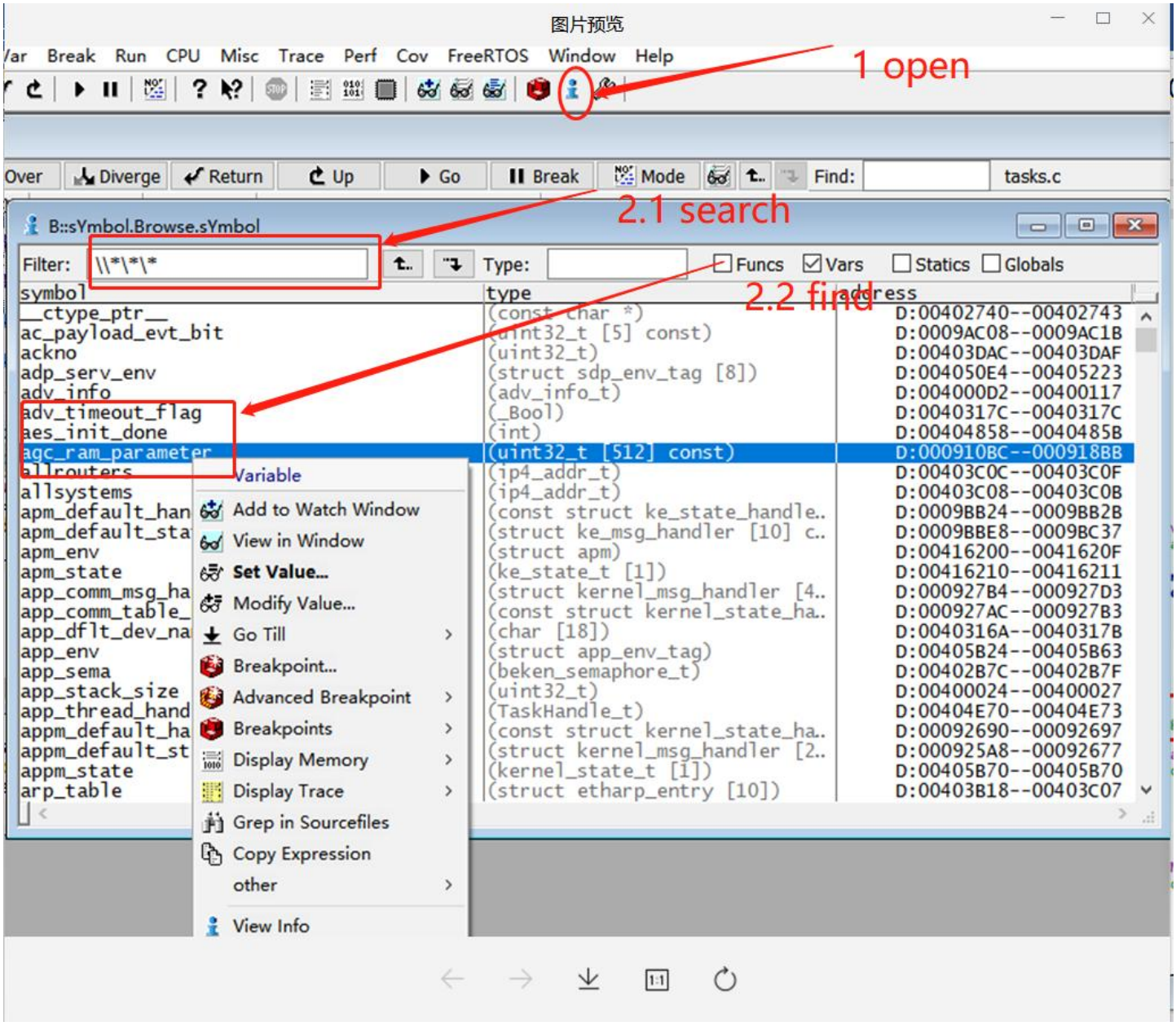
变量断点设置

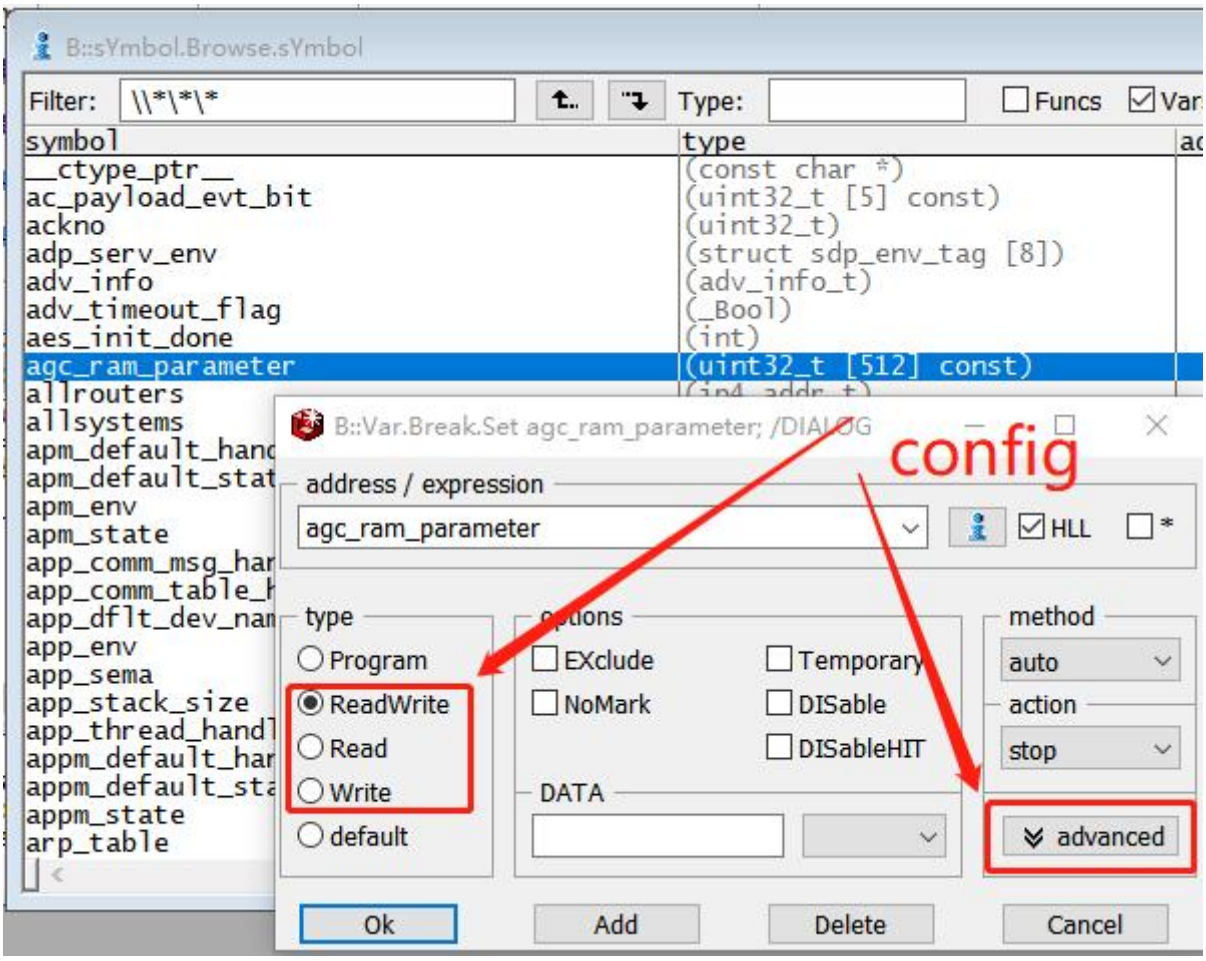
命令行设置

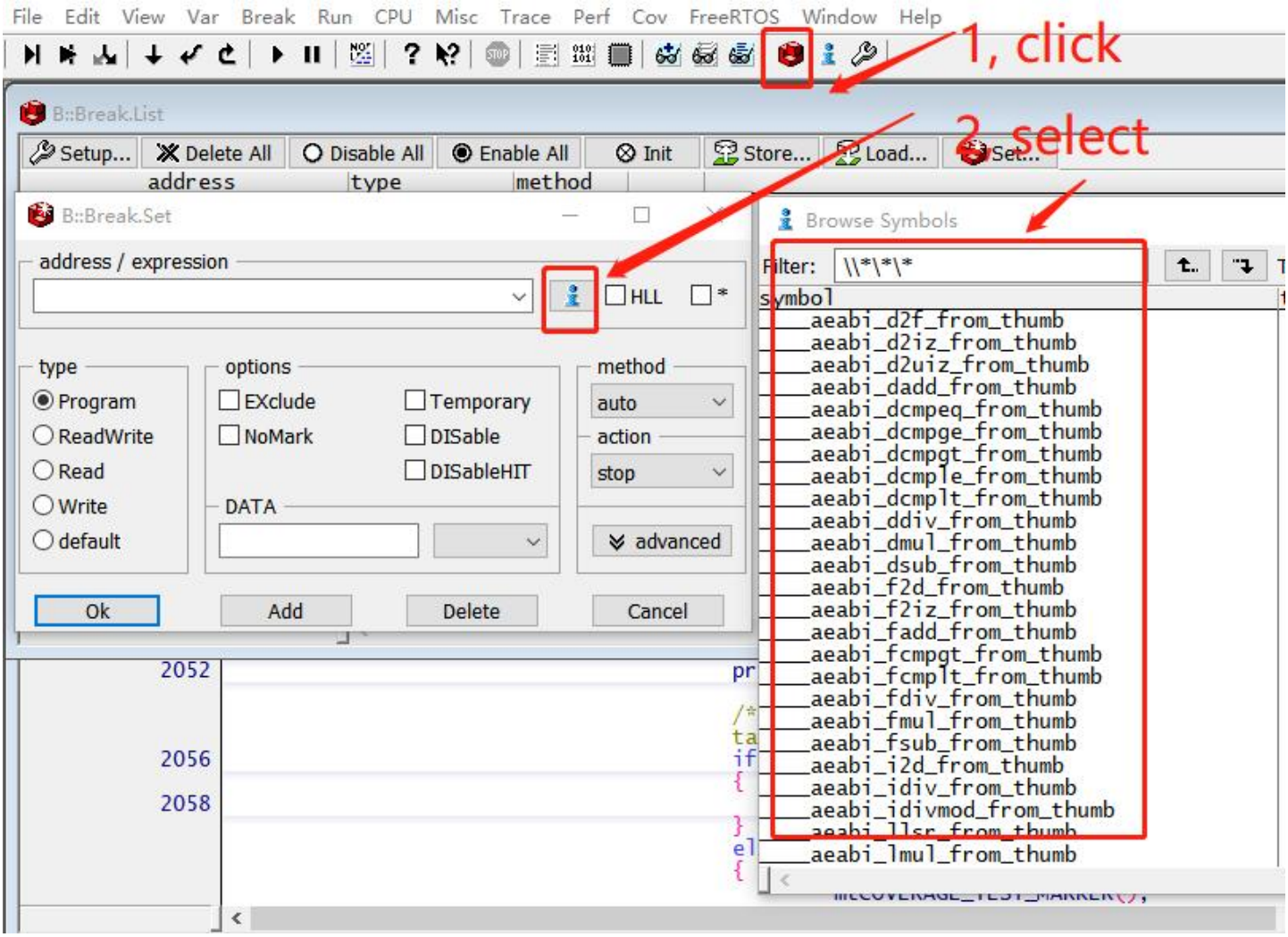
Var.Break.Set


B::Break.List
Setup... Delete All Disable All Enable All Init Store... Load... Set...
address type method modified address
T:000136A0 Program extended_app_launch\13+0x0C
T:000136A2 Program extended_app_launch\13+0x0E
T:000136A6 Program extended_app_launch\13+0x12
T:000136AE Program extended_app_launch\13+0x1A
T:000136B2 Program entry_main\2
T:000136B6 Program entry_main+0x6
T:000136BA Program entry_main+0x0A
T:000136BE Program entry_main\9
T:000136C6 Program entry_main\13
T:000136CA Program entry_main\14
T:000136CE Program entry_main\17
T:000136D2 Program entry_main\17+0x4
C:00416440--00416517 Write ONCHIP C:00416400--004165FF rxu_cntrl_defrag_pool
C:00416440--00416517 ReadWrite ONCHIP C:00416400--004165FF rxu_cntrl_defrag_pool
C:00416440--00416517 ProgramPass rxu_cntrl_defrag_pool
B::var.Break.Set rxu_cntrl_defrag_pool /write

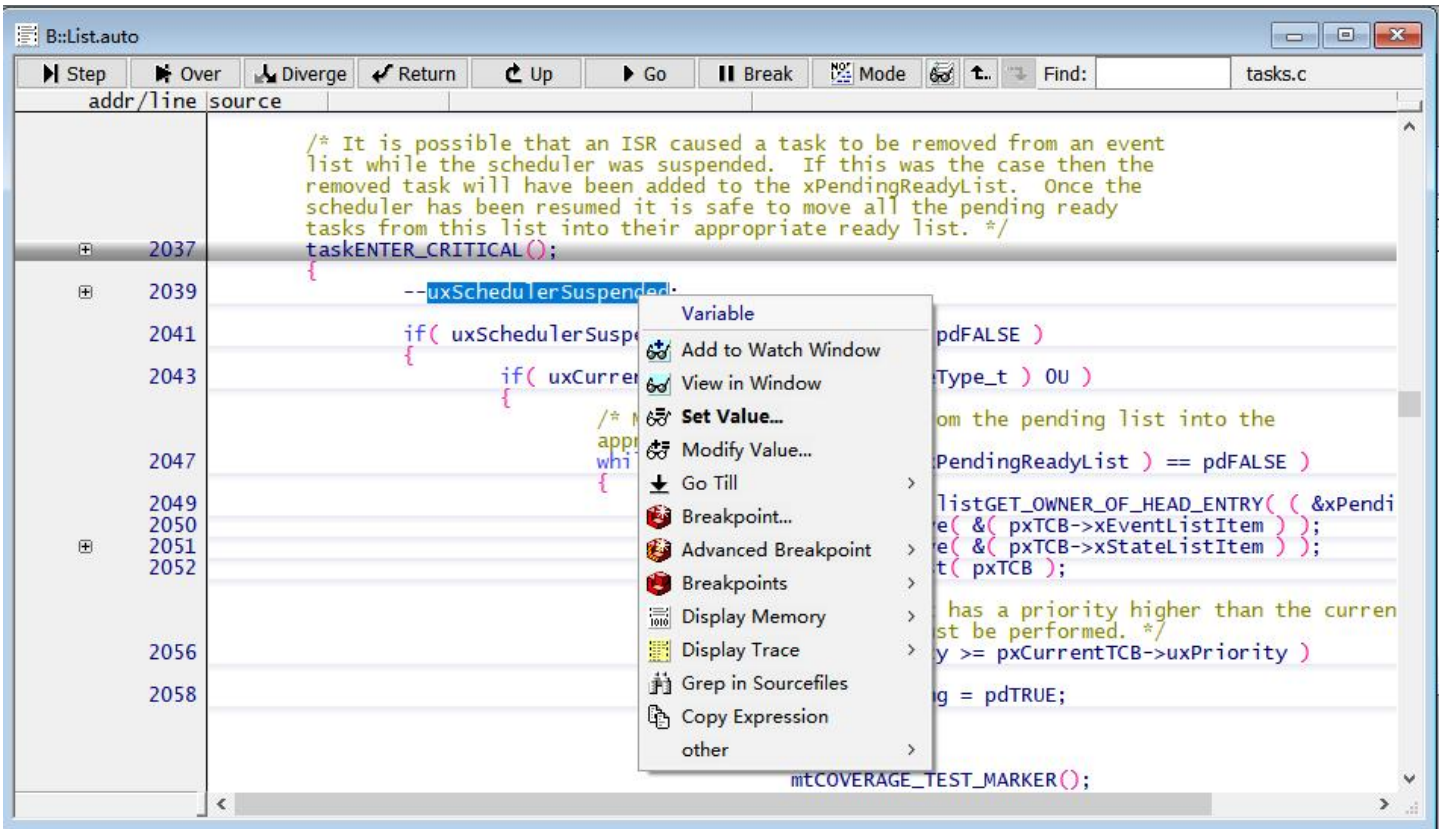
符号表设置


图片预览
Var Break Run CPU Misc Trace Perf Cov FreeRTOS Window Help
1 open
Over Diverge Return Up Go Break Mode Find: tasks.c
2.1 search
B::sYmbol.Browse.sYmbol
Filter: *** Type: Funcs Vars Statics Globals
2.2 find
symbol type address
__ctype_ptr__ (const char *) D:00402740--00402743
ac_payload_evt_bit (uint32_t [5] const) D:0009AC08--0009AC1B
ackno (uint32_t) D:00403DAC--00403DAF
adp_serv_env (struct sdp_env_tag [8]) D:004050E4--00405223
adv_info (adv_info_t) D:004000D2--00400117
adv_timeout_flag (_Bool) D:0040317C--0040317C
aes_init_done (int) D:00404858--0040485B
agc_ram_parameter (uint32_t [512] const) D:000910BC--000918BB
allrouters (ip4_addr_t) D:00403C0C--00403C0F
allsystems (ip4_addr_t) D:00403C08--00403C0B
apm_default_han (const struct ke_state_handle.. D:0009BB24--0009BB2B
apm_default_sta (struct ke_msg_handler [10] c.. D:0009BBE8--0009BC37
apm_env (struct apm) D:00416200--0041620F
apm_state (ke_state_t [1]) D:00416210--00416211
app_comm_msg_ha (struct kernel_msg_handler [4.. D:000927B4--000927D3
app_comm_table_ (const struct kernel_state_ha.. D:000927AC--000927B3
app_dflt_dev_na (char [18]) D:0040316A--0040317B
app_env (struct app_env_tag) D:00405B24--00405B63
app_sema (beken_semaphore_t) D:00402B7C--00402B7F
app_stack_size (uint32_t) D:00400024--00400027
app_thread_hand (TaskHandle_t) D:00404E70--00404E73
appm_default_ha (const struct kernel_state_ha.. D:00092690--00092697
appm_default_st (struct kernel_msg_handler [2.. D:000925A8--00092677
appm_state (kernel_state_t [1]) D:00405B70--00405B70
arp_table (struct etharp_entry [10]) D:00403B18--00403C07
Variable
Add to Watch Window
View in Window
Set Value...
Modify Value...
Go Till
Breakpoint...
Advanced Breakpoint
Breakpoints
Display Memory
Display Trace
Grep in Sourcefiles
Copy Expression
other
View Info


B::sYmbol.Browse.sYmbol
Filter: ***
Type:
Funcs
Var
symbol
type
__ctype_ptr__
(const char *)
ac_payload_evt_bit
(uint32_t [5] const)
ackno
(uint32_t)
adp_serv_env
(struct sdp_env_tag [8])
adv_info
(adv_info_t)
adv_timeout_flag
(_Bool)
aes_init_done
(int)
agc_ram_parameter
(uint32_t [512] const)
allrouters
allsystems
apm_default_hanc
apm_default_stat
apm_env
apm_state
app_comm_msg_har
app_comm_table_h
app_dflt_dev_nam
app_env
app_sema
app_stack_size
app_thread_handl
appm_default_har
appm_default_sta
appm_state
arp_table
B::Var.Break.Set agc_ram_parameter; /DIALOG
config
address / expression
agc_ram_parameter
HLL
*
type
Program
ReadWrite
Read
Write
default
options
EXclude
Temporary
NoMark
DISable
DISableHIT
DATA
method
auto
action
stop
advanced
Ok
Add
Delete
Cancel


File Edit View Var Break Run CPU Misc Trace Perf Cov FreeRTOS Window Help
1, click
B::Break.List
Setup... Delete All Disable All Enable All Init Store... Load... Set...
address type method
2, select
B::Break.Set
address / expression
HLL
*
type
Program
ReadWrite
Read
Write
default
options
EXclude
Temporary
NoMark
DISable
DISableHIT
DATA
method
auto
action
stop
advanced
Ok
Add
Delete
Cancel
Browse Symbols
Filter: ***
symbol
__aeabi_d2f_from_thumb
__aeabi_d2iz_from_thumb
__aeabi_d2uiz_from_thumb
__aeabi_dadd_from_thumb
__aeabi_dcmpeq_from_thumb
__aeabi_dcmpge_from_thumb
__aeabi_dcmpgt_from_thumb
__aeabi_dcmple_from_thumb
__aeabi_dcmplt_from_thumb
__aeabi_ddiv_from_thumb
__aeabi_dmul_from_thumb
__aeabi_dsub_from_thumb
__aeabi_f2d_from_thumb
__aeabi_f2iz_from_thumb
__aeabi_fadd_from_thumb
__aeabi_fcmpgt_from_thumb
__aeabi_fcmplt_from_thumb
__aeabi_fdiv_from_thumb
__aeabi_fmul_from_thumb
__aeabi_fsub_from_thumb
__aeabi_i2d_from_thumb
__aeabi_idiv_from_thumb
__aeabi_idivmod_from_thumb
__aeabi_llsr_from_thumb
__aeabi_lmul_from_thumb
2052
2056
2058

HLL 设置



Task 相关断点设置

如果是 TRACE32 支持的 RTOS，设置断点的时候，**可以指定 task**。换句话说只有当前 task 是指定的 task 的时候，才可能 break。如下图所示：

B::TASK.TaskList

magic	name	num	prio	state
0041ECC8	idle	3.	0.	running
004223B8	tcp/ip	9.	7.	blocked
0041FAC8	timer_thd	4.	6.	blocked
00421180	kmsgbk	7.	6.	blocked
00423880	wpas_thread	10.	4.	blocked
004247E0	core_thread	12.	7.	blocked
00426240	cli	13.	3.	blocked
004200B8	temp_detct	6.	3.	blocked

[B::List \printf\1]

Step | Over | Diverge | Return | Up | Go | Brea

```
addr/line  source
           * if a size of zero was specified, whi
           */
           if (str == NULL && size != 0)
250                size = 0;

252        while (ch != '\0') {
253                switch (state) {
                   case PRINT_S_DEFAULT:
```

Change Breakpoint T:0x1862E

address / expression: __wrap_vsnprintf\34 — HLL

type: Program, ReadWrite, Read, Write, default

options: EXclude, Temporary, NoMark, DISable, DISableHIT

DATA

method: auto

action: stop

advanced

Ok | Add | Delete | Cancel

ProgramPass, ProgramFail, MemoryReadWrite, MemoryRead, MemoryWrite, RegisterReadWrite, RegisterRead, RegisterWrite

memory / register / var — HLL

TASK: "temp_detct"

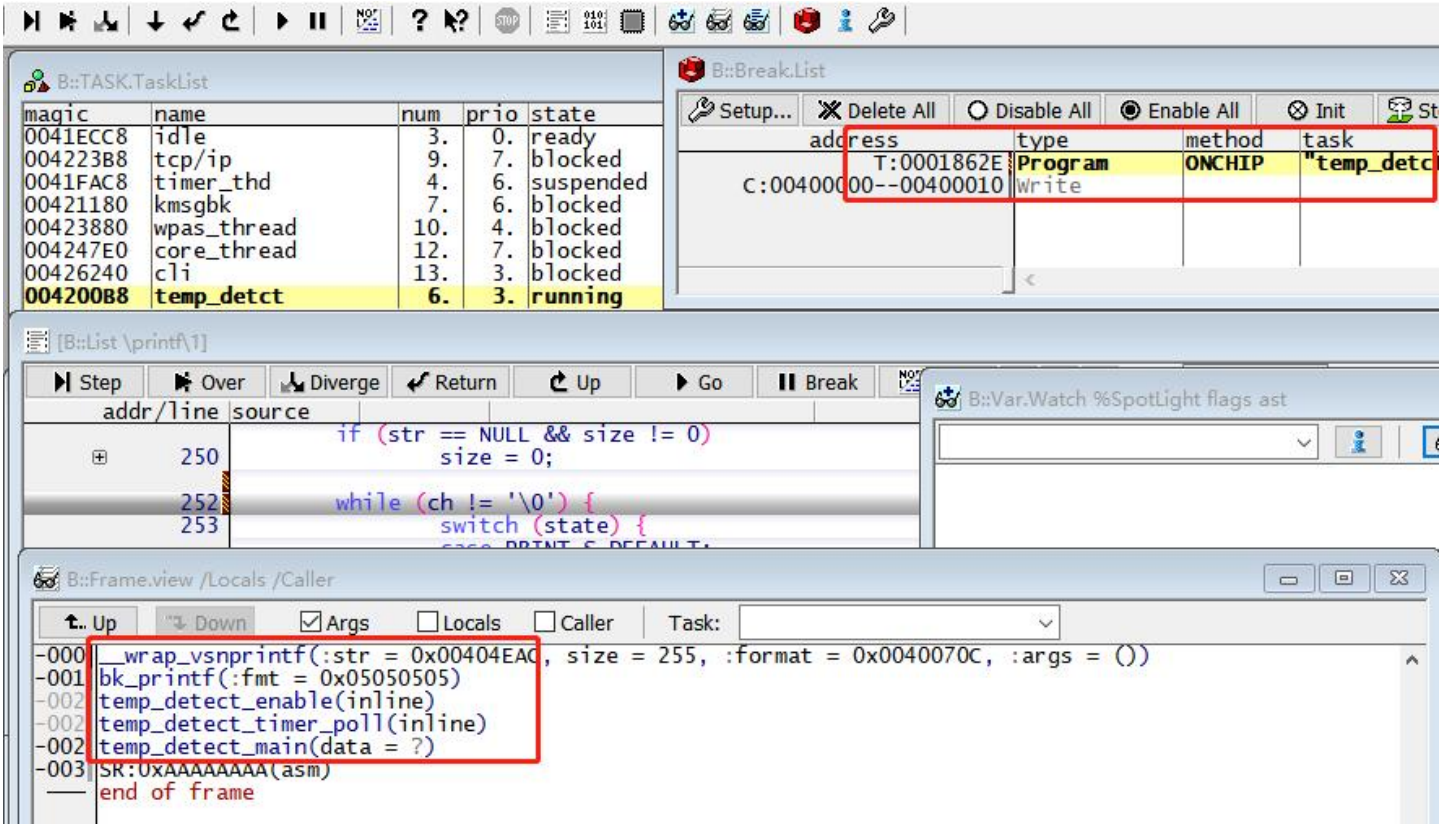
COUNT: 1.

MACHINE

CORE

CONDition — HLL, AfterStep

CMD — RESUME

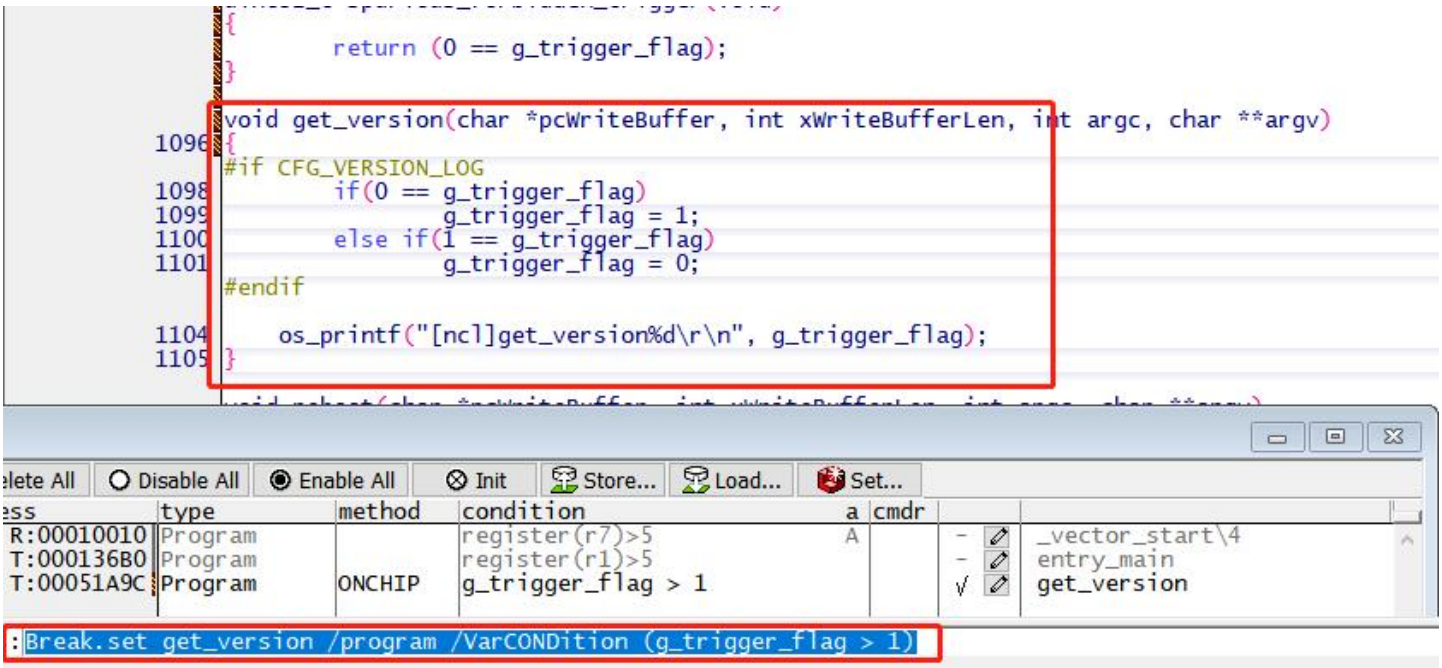


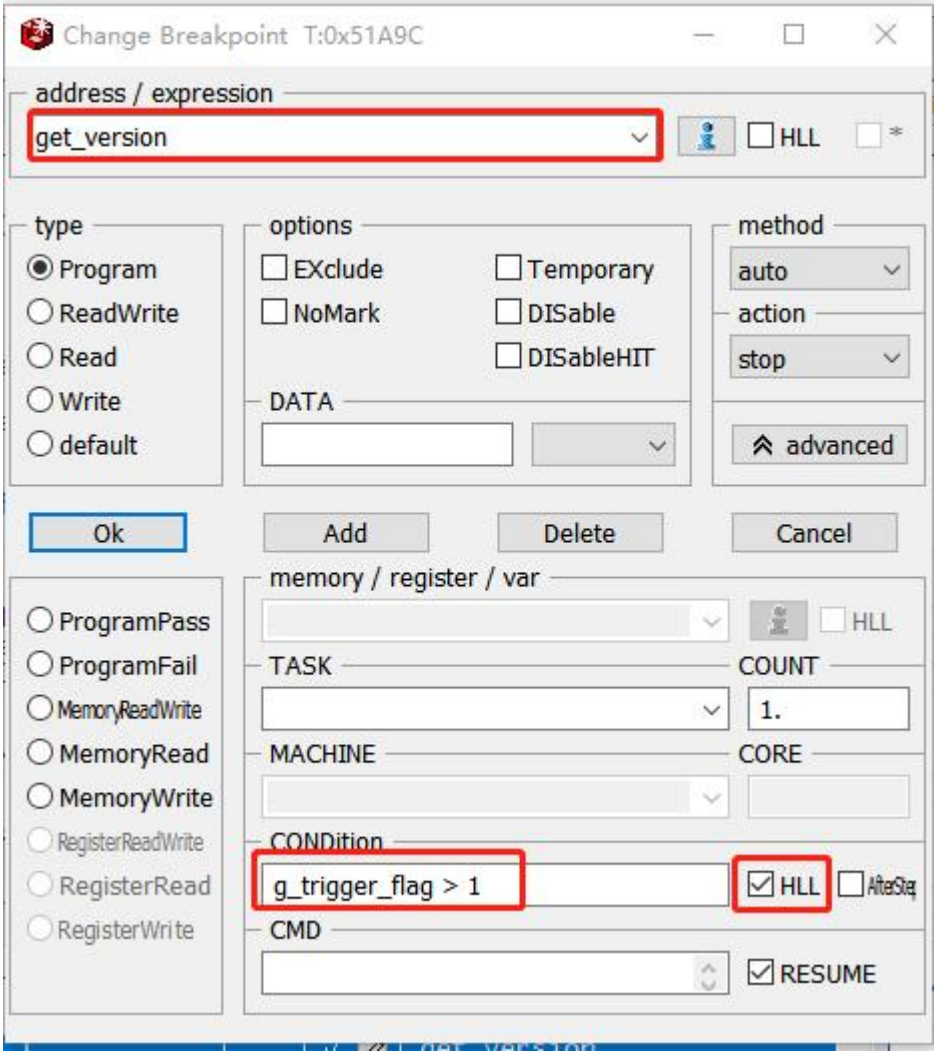
条件断点设置

HLL 条件断点

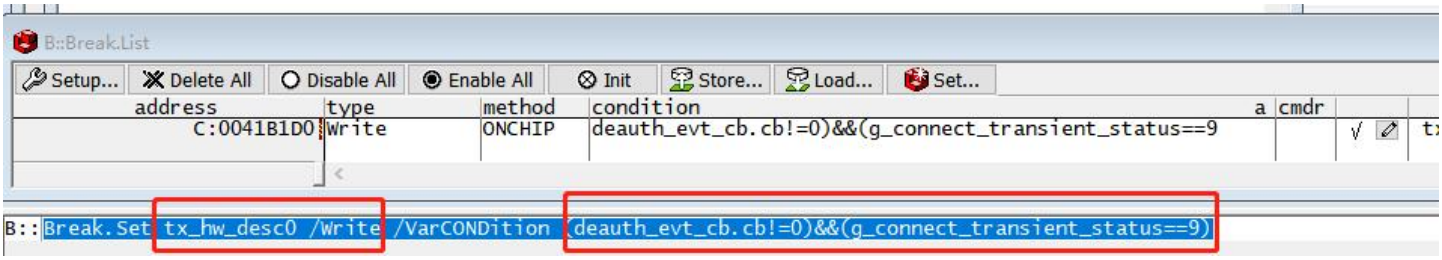
Hll 条件就是高级语言的**条件表达式**

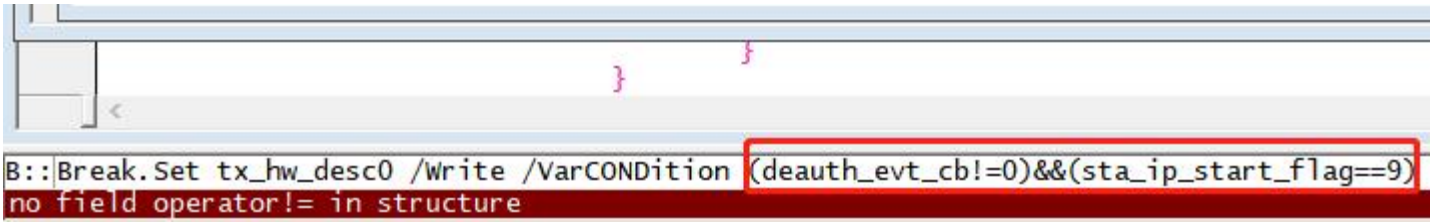
Break.set get_version /program /VarCONDition (g_trigger_flag > 1)





Break.Set tx_hw_desc0 /Write /VarCONDition (deauth_evt_cb.cb!=0)&&(g_connect_transient_status==9)

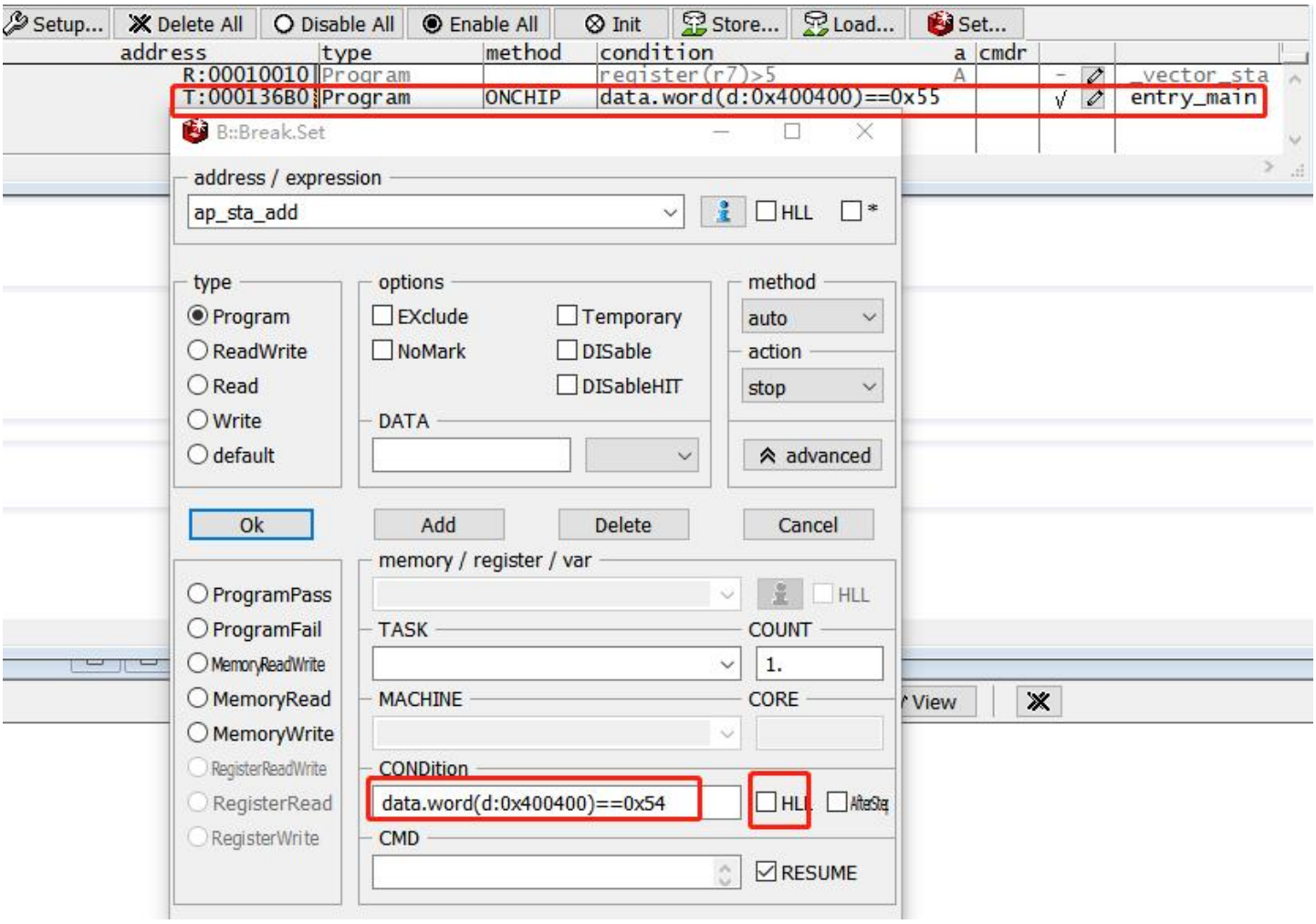




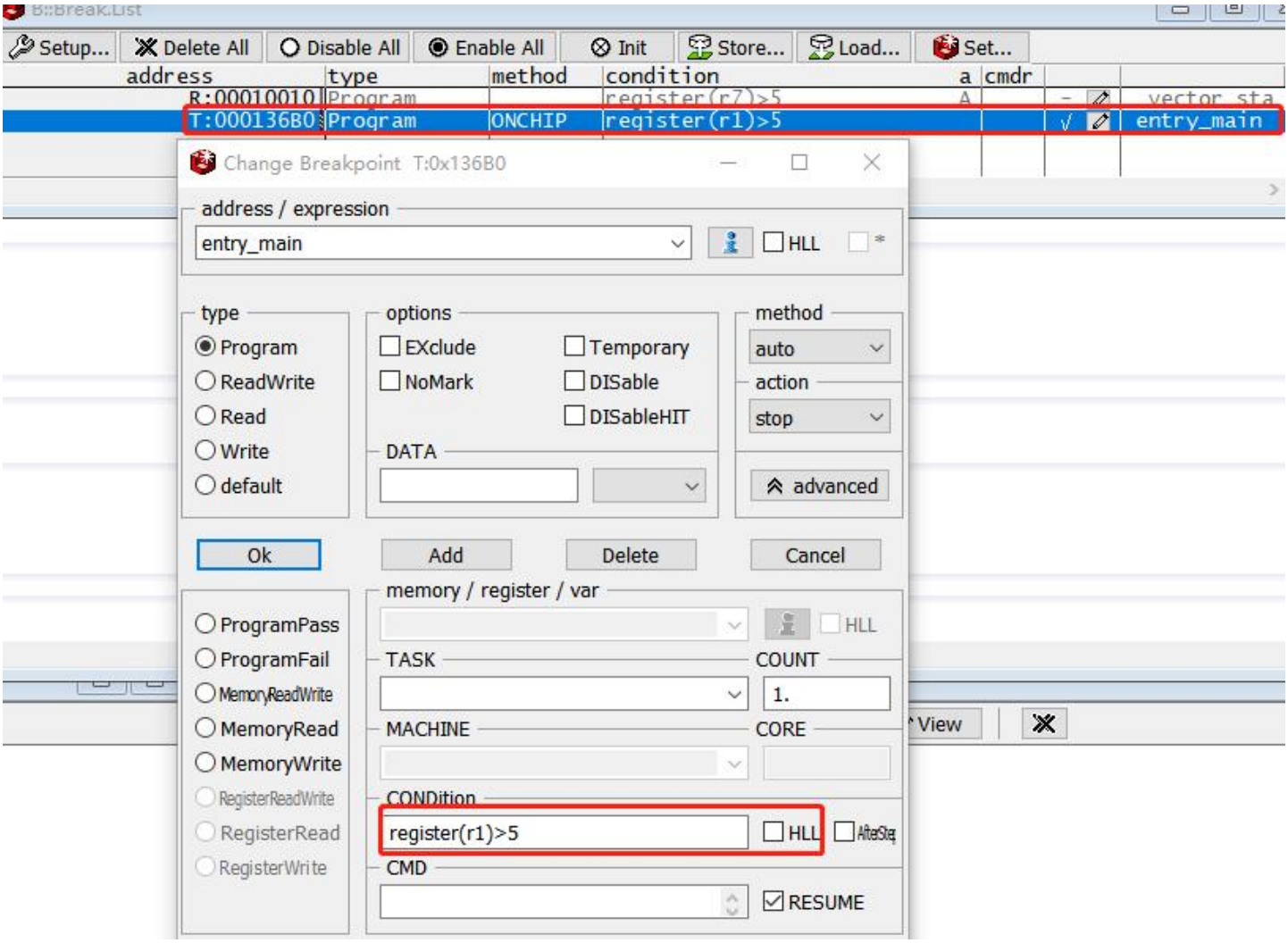
TRACE32 脚本条件断点

当不勾选 HLL 或者使用 Condition 参数的时候，可以使用 TRACE32 脚本作为条件表达式

Break.set entry_main /program /CONDition data.word(d:0x400400)==0x54



Break.set entry_main /program /CONDition Register(R1)>5



Tips： Data.Long/ Data.Word/Data.Byte

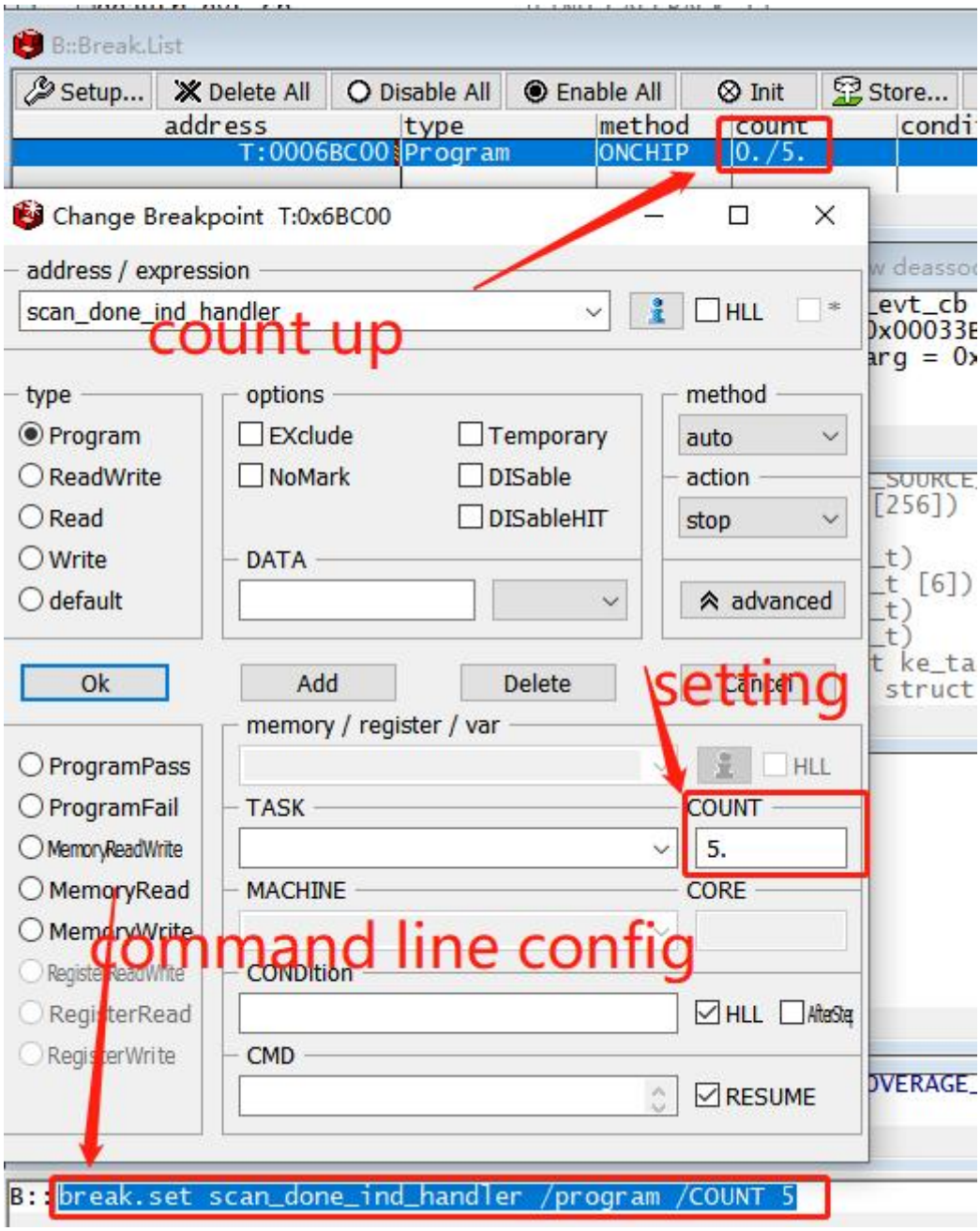
Data.Long/ Data.Word/Data.Byte 返回 32bits/16bits/8bits 的内存内容，括号内表示地址。D：Data Space

FCode	Description
P	Program Space
D	Data Space
C	Access by the CPU
E	Direct access to emulation memory
EP	Direct access to program memory
ED	Direct access to data memory
A	Absolute (Physical addressing)

断点计数器设置

break.set scan_done_ind_handler /program /COUNT 5

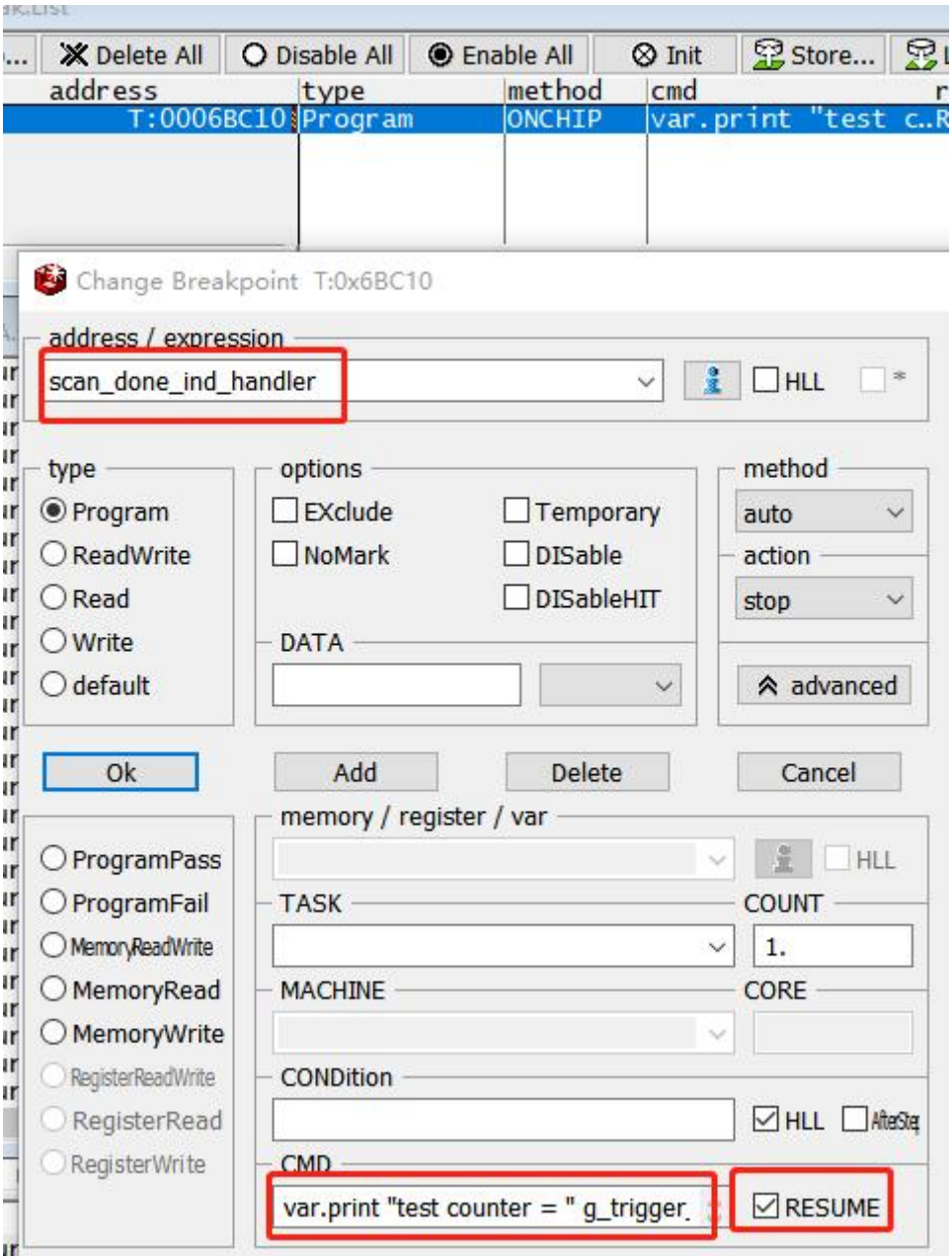
COUNT 指的是命中断点几次之后，程序会停止执行。次数设置如下图：



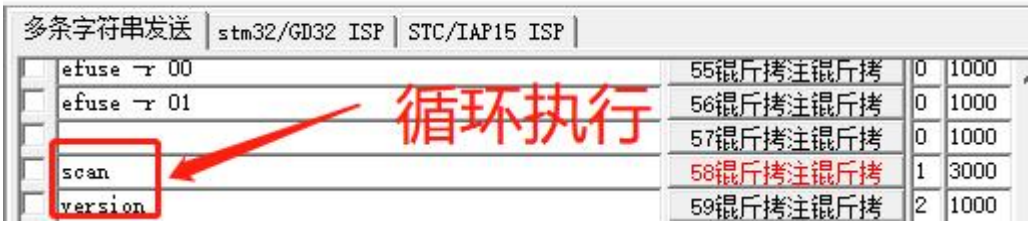
Cmd 相关断点设置

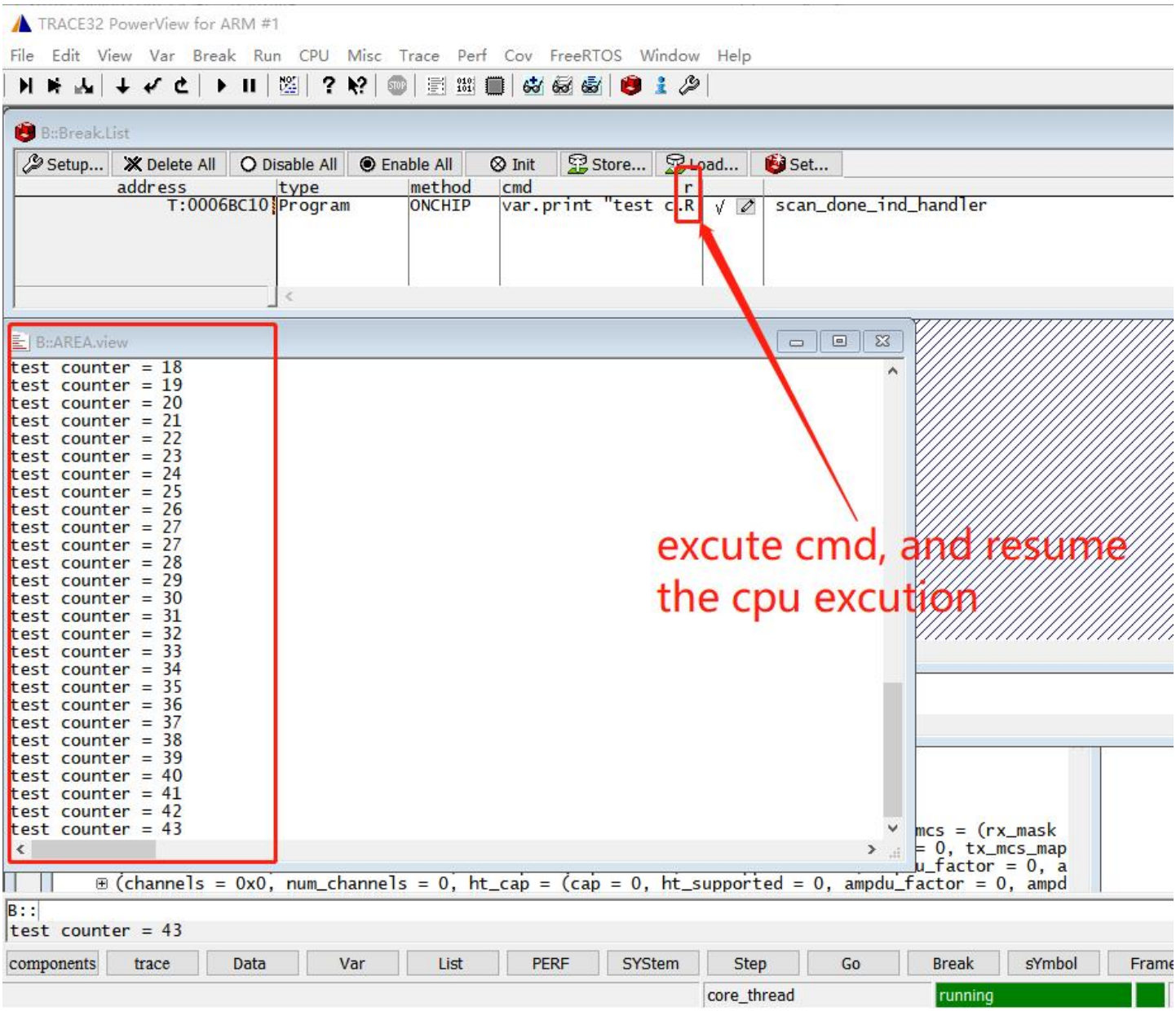
break.Set scan_done_ind_handler /Program /CMD "var.print ""test counter = "" g_trigger_flag" /RESUME

CMD 值得是 **TRACE32 命令字符串**，如果不加 RESUME，当断点断下来之后执行完 CMD 就结束了。如果 RESUME，CPU 会继续执行。注意：**执行 CMD 命令至少要耗费 200ms 时间**。

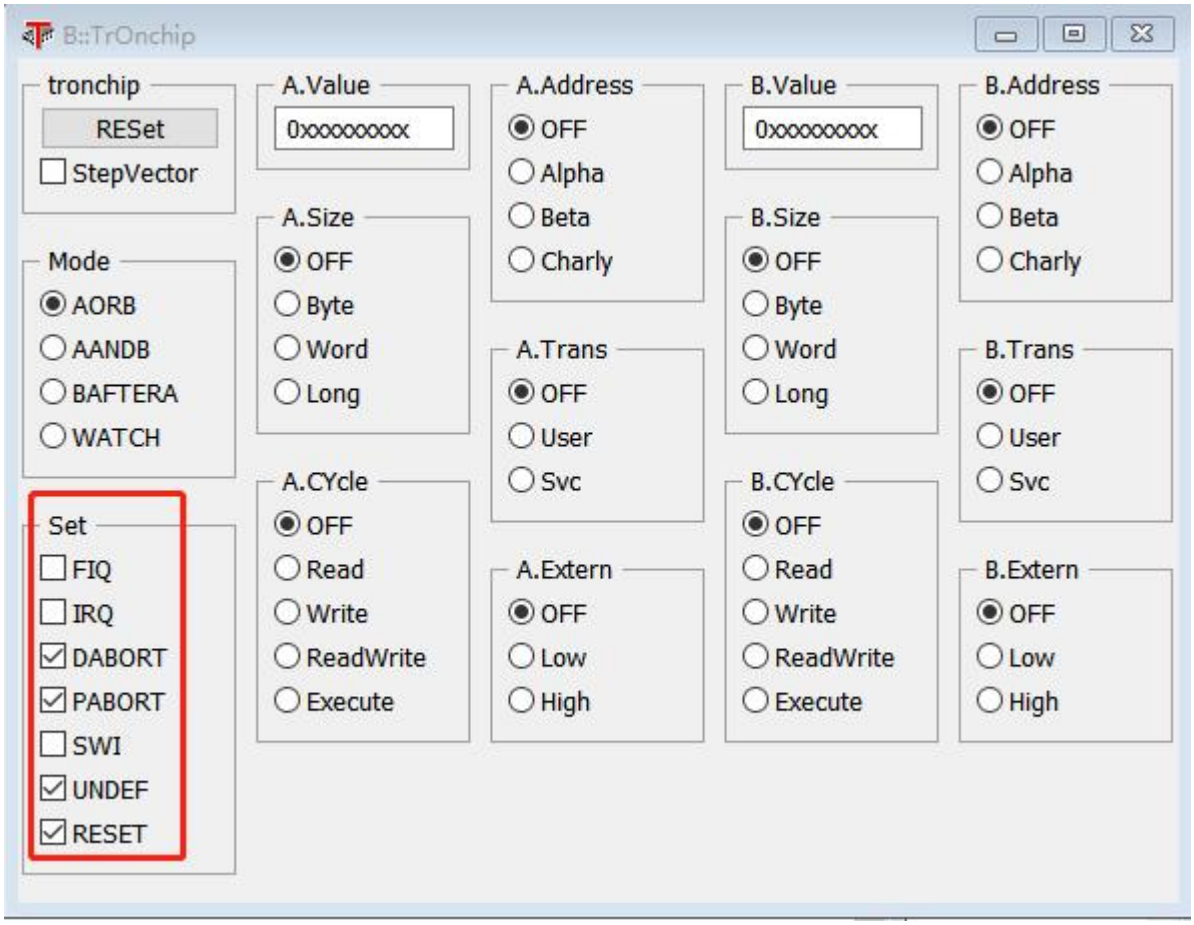



```
094:
095: void get_version(char *pcWriteBuffer, int xWriteBuffe
096: {
097: #if CFG_VERSION_LOG
098:     if(0 == g_trigger_flag)
099:         g_trigger_flag = 1;
100:     else if(1 == g_trigger_flag)
101:         g_trigger_flag = 0;
102: #endif
103:     g_trigger_flag ++;
104:
105:     if(g_trigger_flag == 2048)
106:         g_trigger_flag = 0;
107:
108:     os_printf("[ncl]get_version%d\r\n", g_trigger_flag);
109: }
```




TRACE32 PowerView for ARM #1
File Edit View Var Break Run CPU Misc Trace Perf Cov FreeRTOS Window Help
B::Break.List
Setup... Delete All Disable All Enable All Init Store... Load... Set...
address type method cmd r
T:0006BC10 Program ONCHIP var.print "test c.R scan_done_ind_handler
B::AREA.view
test counter = 18
test counter = 19
test counter = 20
test counter = 21
test counter = 22
test counter = 23
test counter = 24
test counter = 25
test counter = 26
test counter = 27
test counter = 27
test counter = 28
test counter = 29
test counter = 30
test counter = 31
test counter = 32
test counter = 33
test counter = 34
test counter = 35
test counter = 36
test counter = 37
test counter = 38
test counter = 39
test counter = 40
test counter = 41
test counter = 42
test counter = 43
excute cmd, and resume
the cpu excution
B::
test counter = 43
components trace Data Var List PERF SYStem Step Go Break sYmbol Frame
core_thread running

ARM 异常模式断点设置



常见问题

Debug port fail

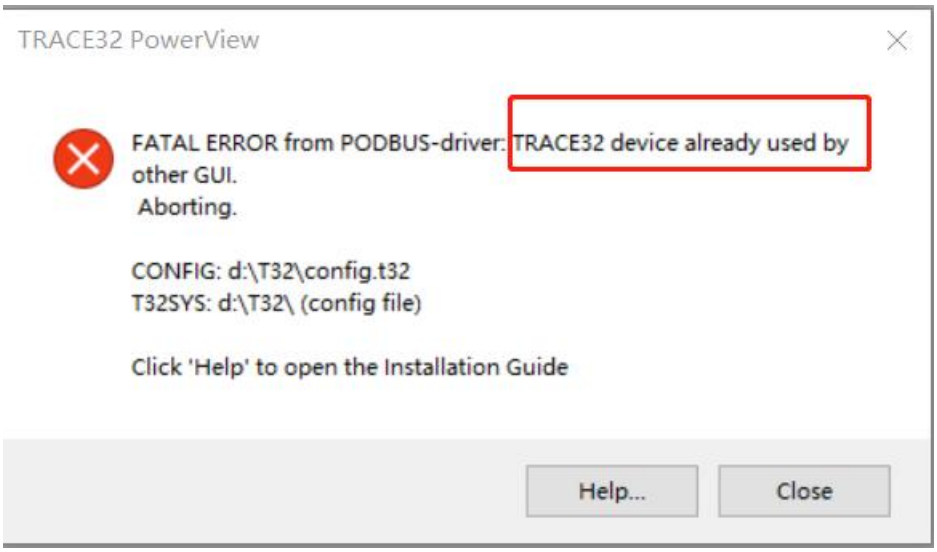
确认**开发板是否上电**，连接端口是否异常

Fatal error #FF, contact technical support (support@lauterbach.com)

调试器下电了，软件界面异常。关闭软件，重启 TRACE32 硬件。

```
resetting...
fatal error #FF, contact technical support (support@lauterbach.com)
function not implemented
fatal error #FF, contact technical support (support@lauterbach.com)
function not implemented
fatal error #FF, contact technical support (support@lauterbach.com)
fatal error #FF, contact technical support (support@lauterbach.com)
fatal error #FF, contact technical support (support@lauterbach.com)
Warning: timebase correlation fail (factor = -2323.725196)
         host time: 469159875717 usec, device time: 44376208 usec, delta time: 46911
fatal error #FF, contact technical support (support@lauterbach.com)
fatal error #FF, contact technical support (support@lauterbach.com)
fatal error #FF, contact technical support (support@lauterbach.com)
function not implemented
Warning: Initial DAP discovery failed!
fatal error #FF, contact technical support (support@lauterbach.com)
function not implemented
Warning: DAP component not available!
Warning: timebase correlation fail (factor = 2324.455020)
         host time: 246193631 usec, device time: 469361715709 usec, delta time: -469
```

TRACE32 device already used by other GUI



原因是 TRACE32 没有掉电的情况下，换了 usb port。

IAR + ARM9 + GNU Tools ARM Embedded + ELF + JLINK

IAR 简介

IAR for ARM 是一款业内顶级的集成开发环境，全称为“**IAR Embedded Workbench for ARM**”，同时也是由世界知名的瑞典 IAR Systems 公司为微处理器开发的一个集成开发环境，该公司是嵌入式系统开发工具和服务的供应商，公司成立于 **1983** 年，公司总部在位于**瑞典乌普萨拉市**，于 **1999 年 1 月 4 日**在纳斯达克 OMX 斯德哥尔摩上市。因此，使

得 IAR for ARM 软件在业内享有极高的权威性，它支持 **ARM，AVR，MSP430** 等芯片内核平台，其中包含一个全软件的模拟程序(simulator)。用户不需要任何硬件支持就可以模拟各种 ARM 内核、外部设备甚至中断的软件运行环境。从中可以了解和评估 IAR EWARM 的功能和使用方法。尤其是 IAR 嵌入式工作台与所有部门提供了一个完整的 IDE 开发人员需要在一个视图中。包括 IAR C/c++编译器提供领先技术建造的 IAR 编译器专家系统 C 语言开发工具，添加符合最新标准的 ISO/IEC 9899:2011 以及最新的 c++标准 ISO/IEC。
强大的扩展和集成，包括易于使用的调试和跟踪调查以及集成的静态分析工具和运行时分析，添加额外的功能。完全支持开发人员在他们的特定需求，IAR for ARM 系统提供了世界各地的广泛的技术支持服务和培训课程，能让开发人员的程序更加符合设计标准，更容易在不同的处理器中进行运行，提高了用户开发设计程序的可能性，对于开发人员来说，这款开发环境还非常的小巧玲珑，但是各类功能却一定也不少，还有针对 STM8 单片机的各种优化，更是让开发人员在开发微处理器程序的时候能够更加轻松。
除此之外，新版的 IAR for ARM 还提供了一个工具 IDE 的重大更新，包括功能的改进,如改善了窗口管理和对接使更高效的工作流程。
– 支持 C11 语言标准
编译器和库增加了对最新 C 语言标准 ISO / IEC 9899：2011 的支持。
注意：新的 C 库二进制对象接口与早期的 EWARM 版本不兼容，有关更多详细信息，请参阅编译器发行说明。

– 支持 C ++ 14 语言标准
编译器和库现在支持最新的 C ++标准 ISO / IEC 14882：2014。
注意：新的 C ++库二进制对象接口与早期的 EWARM 版本不兼容，有关更多详细信息，请参阅编译器发行说明。

– 更新的 IDE 外观
有新的艺术品，增强窗口管理和对接，修订信息中心和新的教程。

– 一般支持 Unicode
整个工具链现在支持 Unicode，包括编译器，IDE，编辑器和调试器。

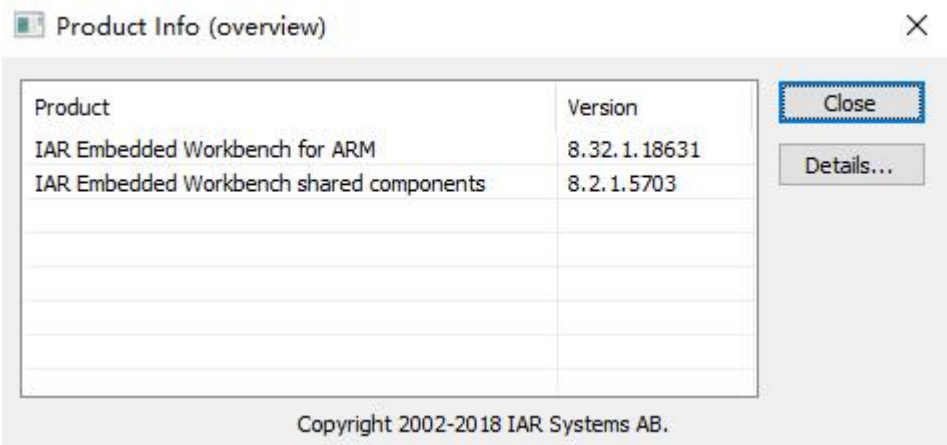
– CMSIS-Pack 支持
项目经理已扩展到使用 ARM CMSIS-Pack 软件组件进行读取和操作。

– 跟踪过滤和导航功能
在大型跟踪数据采集中的导航相当容易，可以在循环，函数，中断和语句边界上向前/向后导航。

IAR 安装以及破解

台式机配置

IAR 版本

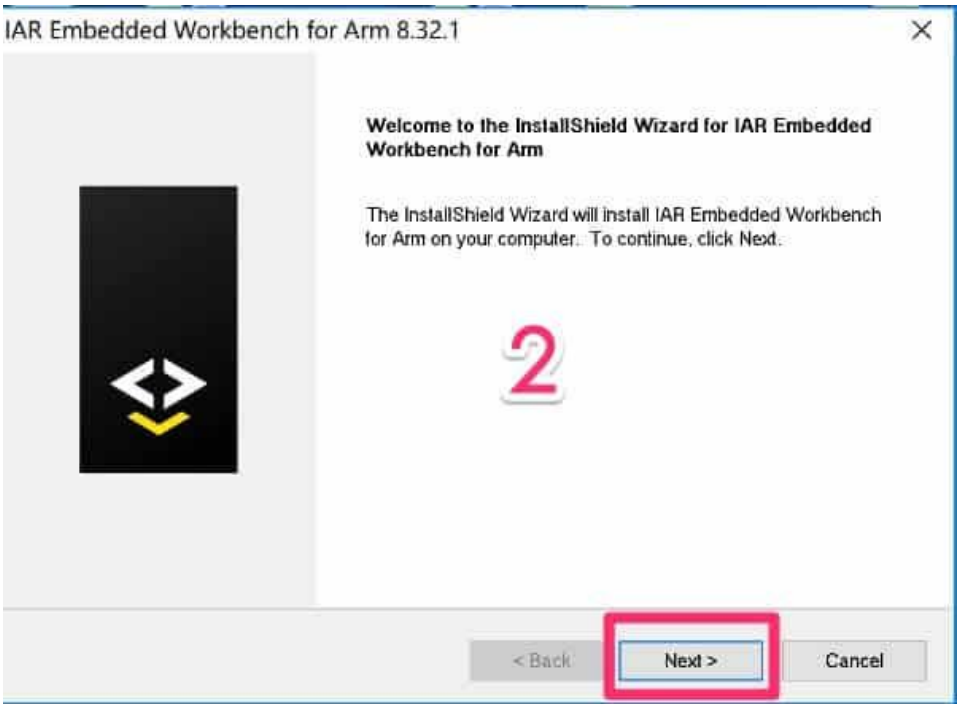


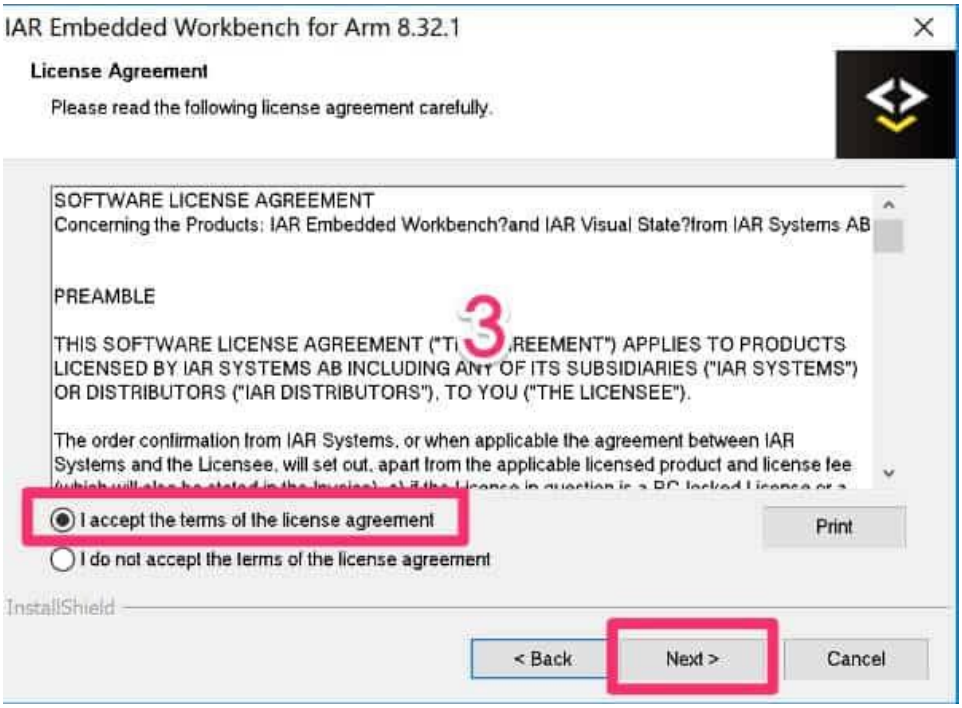
安装

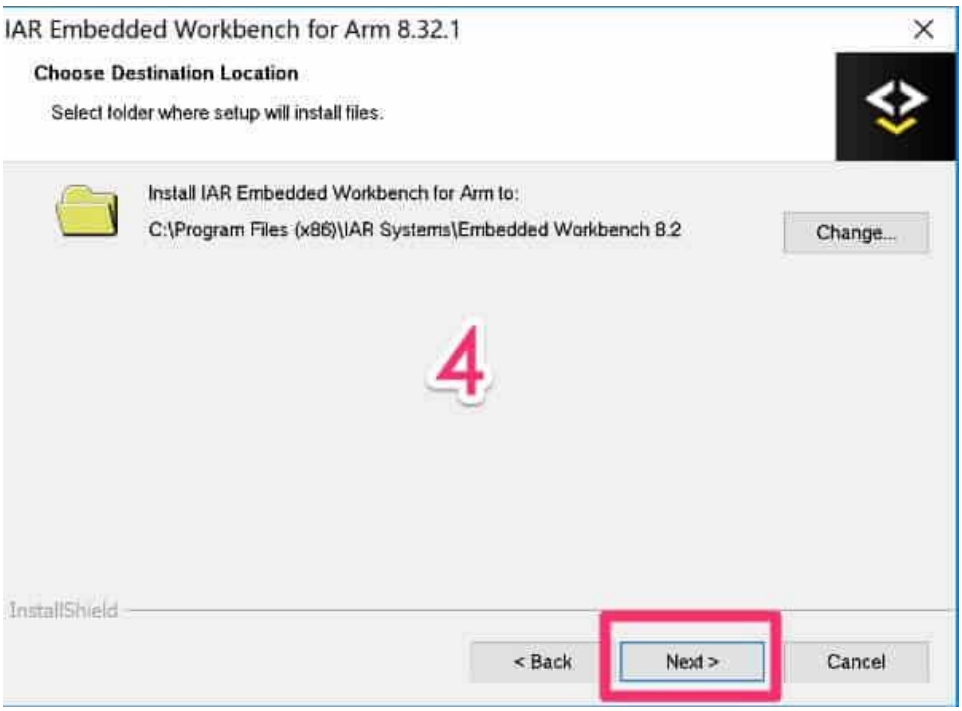
TIPS：step by step，安装常规下一步就可以；

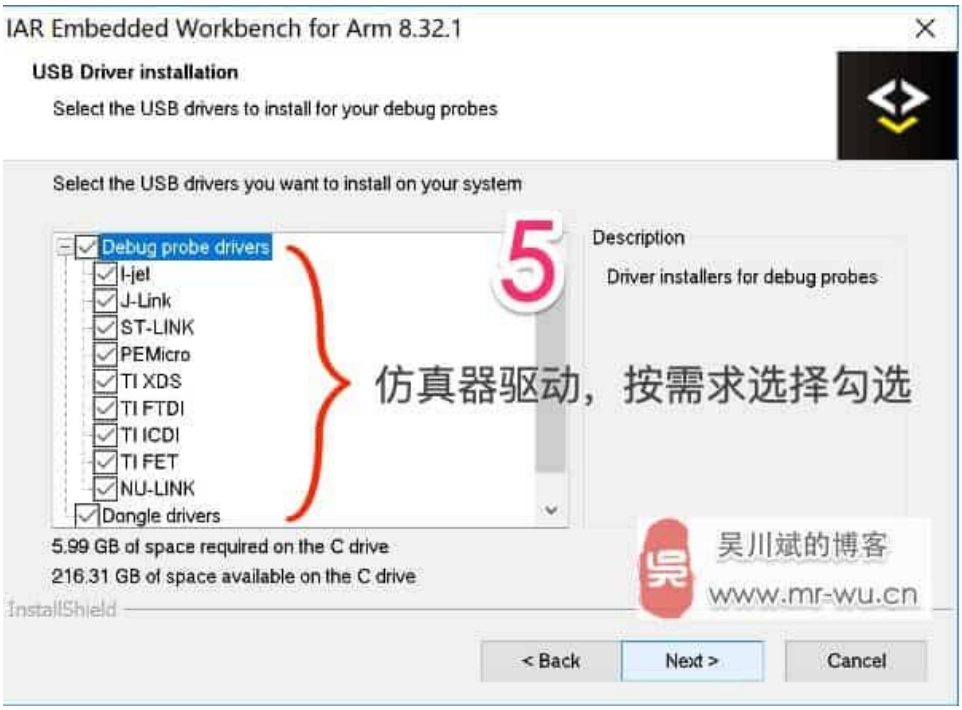
下载内网地址：\\192.168.0.30\exchange\wangzhilei\tools\IAR Embedded Workbench for ARM 8.32.1

细节如下：

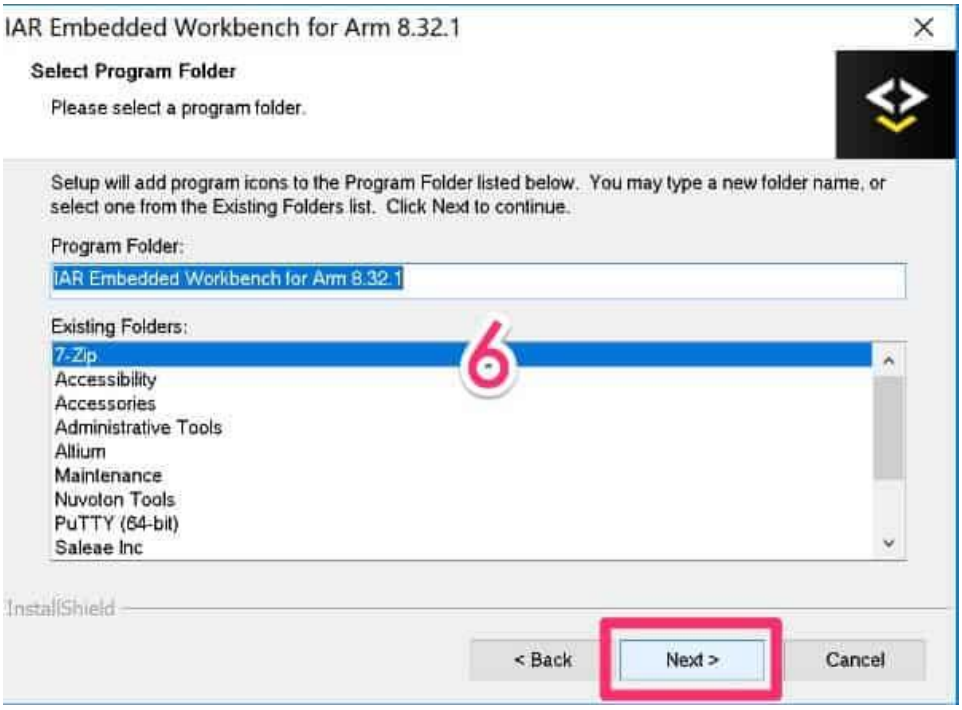

IAR Embedded Workbench for Arm 8.32.1
Welcome to the InstallShield Wizard for IAR Embedded Workbench for Arm
The InstallShield Wizard will install IAR Embedded Workbench for Arm on your computer. To continue, click Next.
2
< Back
Next >
Cancel

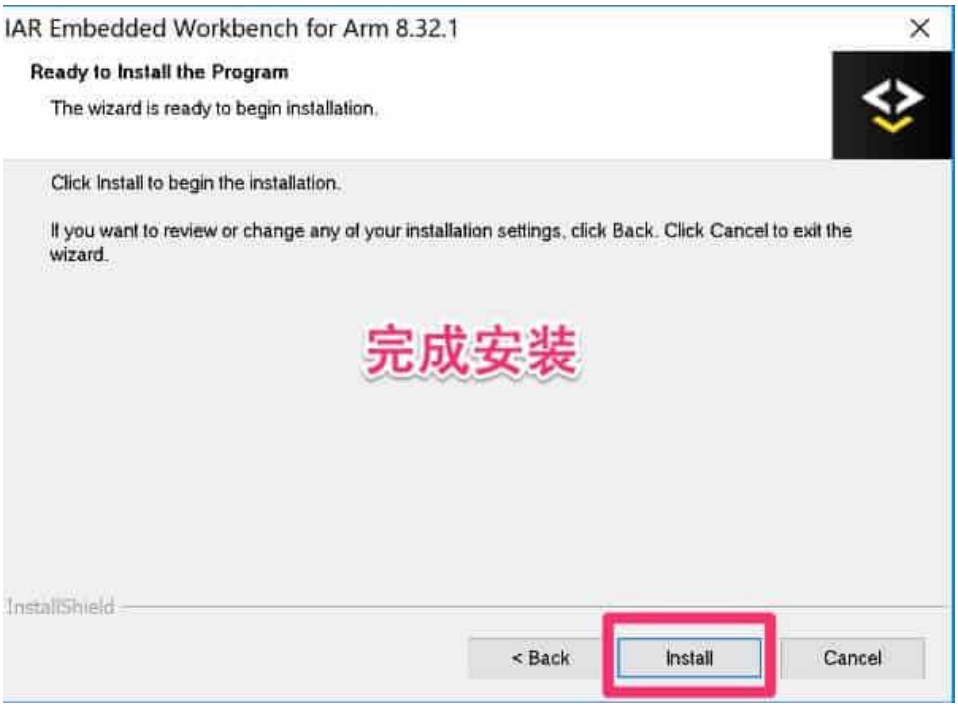

IAR Embedded Workbench for Arm 8.32.1
License Agreement
Please read the following license agreement carefully.
SOFTWARE LICENSE AGREEMENT
Concerning the Products: IAR Embedded Workbench?and IAR Visual State?from IAR Systems AB
PREAMBLE
THIS SOFTWARE LICENSE AGREEMENT ("THIS AGREEMENT") APPLIES TO PRODUCTS LICENSED BY IAR SYSTEMS AB INCLUDING ANY OF ITS SUBSIDIARIES ("IAR SYSTEMS") OR DISTRIBUTORS ("IAR DISTRIBUTORS"), TO YOU ("THE LICENSEE").
The order confirmation from IAR Systems, or when applicable the agreement between IAR Systems and the Licensee, will set out, apart from the applicable licensed product and license fee
3
I accept the terms of the license agreement
I do not accept the terms of the license agreement
Print
InstallShield
< Back
Next >
Cancel


IAR Embedded Workbench for Arm 8.32.1
Choose Destination Location
Select folder where setup will install files.
Install IAR Embedded Workbench for Arm to:
C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2
Change...
4
InstallShield
< Back
Next >
Cancel



吴川斌的博客
www.mr-wu.cn

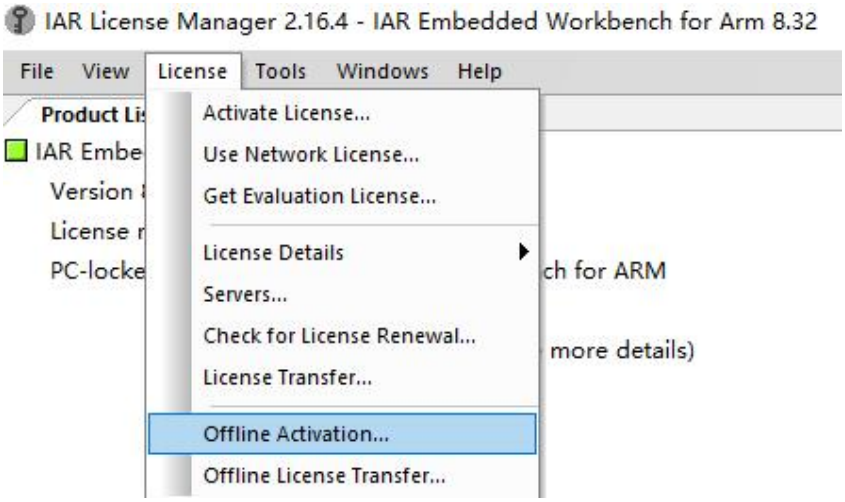




破解

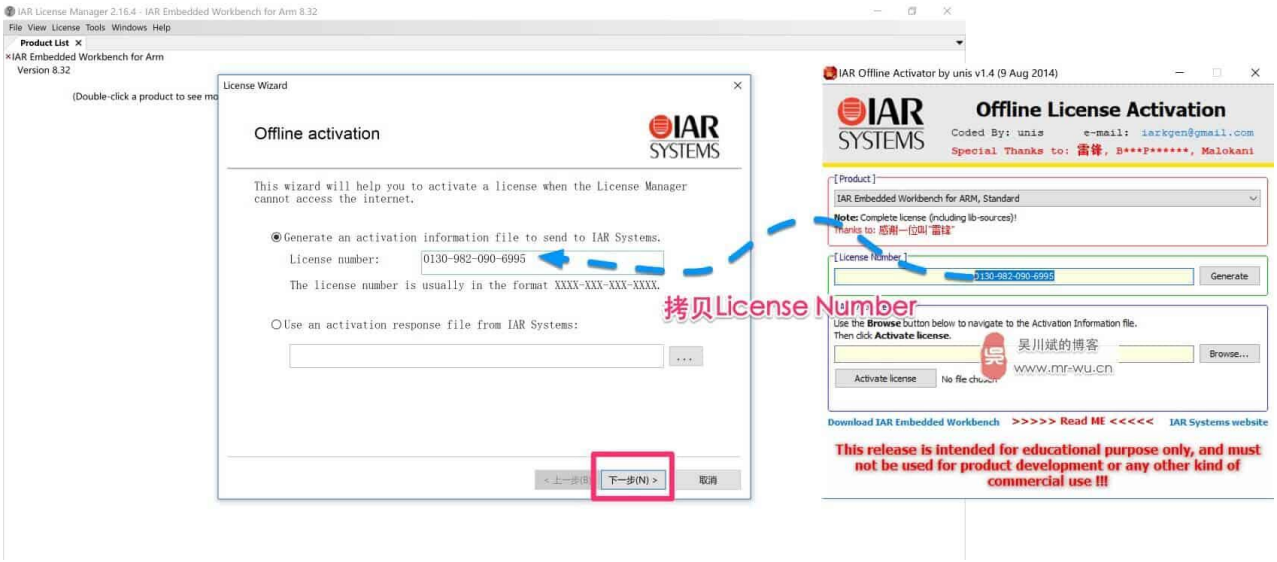
打开 License Manager

IAR Help→License Manager

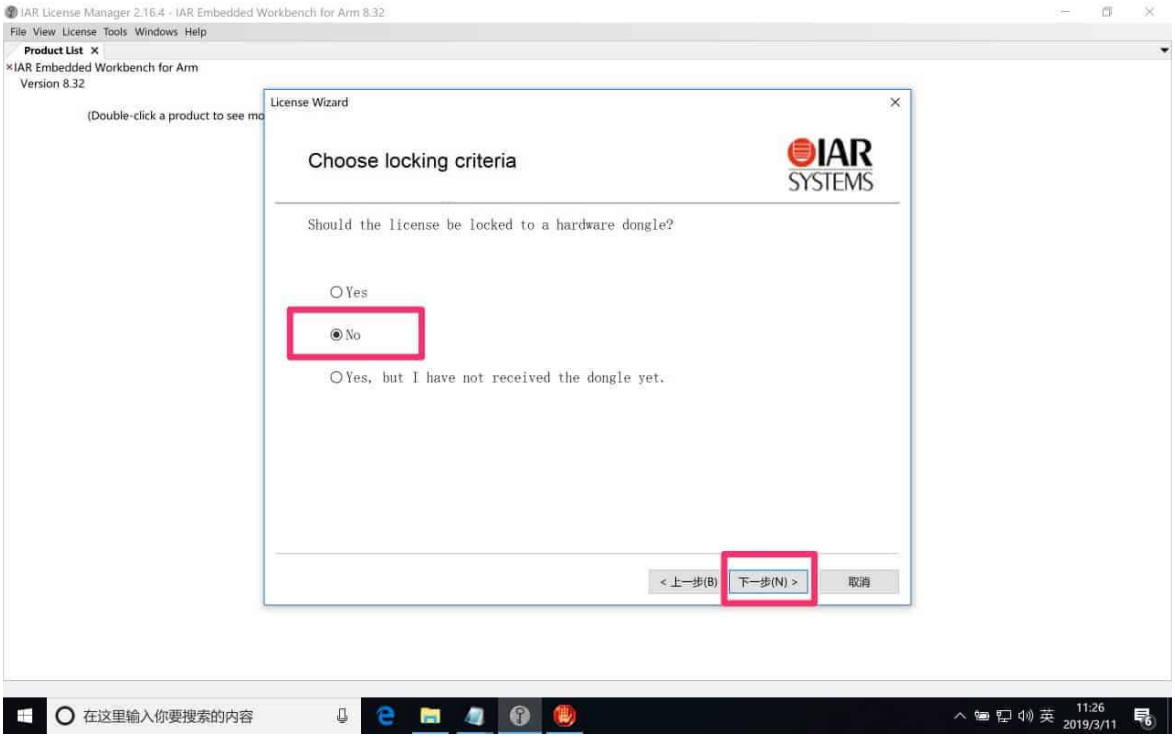


打开注册机程序 License Generator.exe

License Number 拷贝

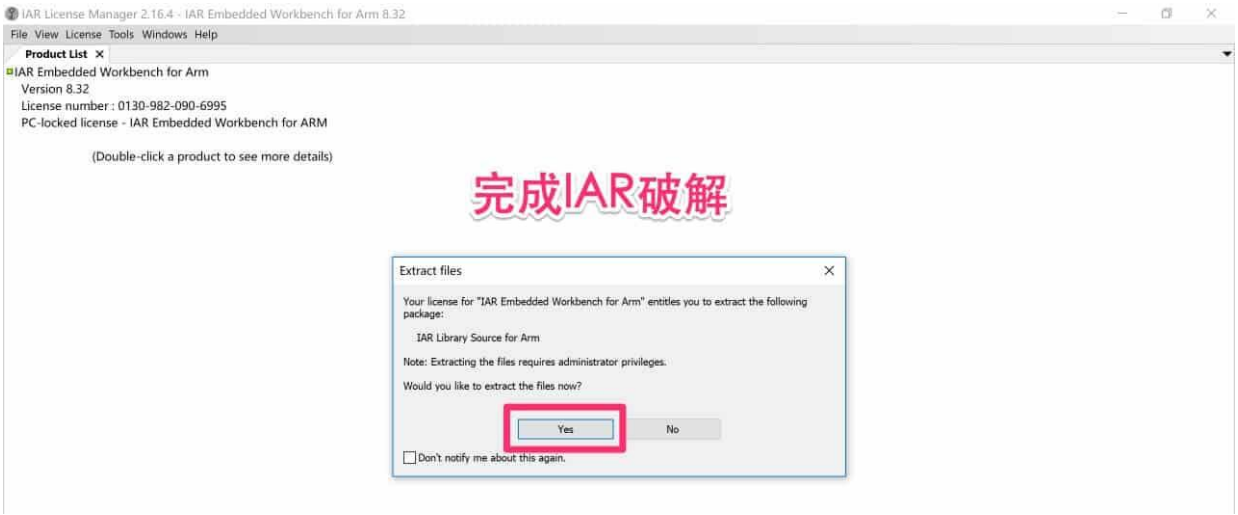


选择锁策略 Choose locking criteria



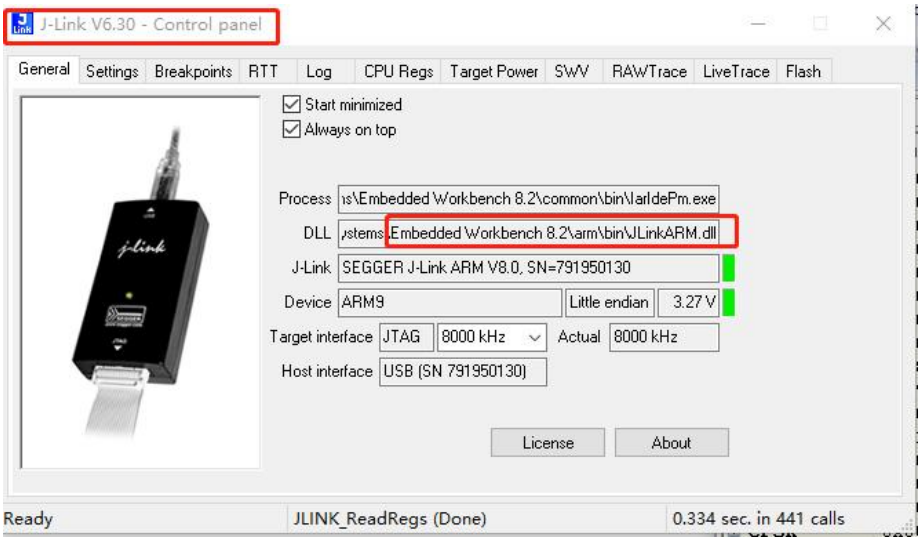
ActivationInfo.txt 和 ActivationResponse.txt

IAR 破解成功



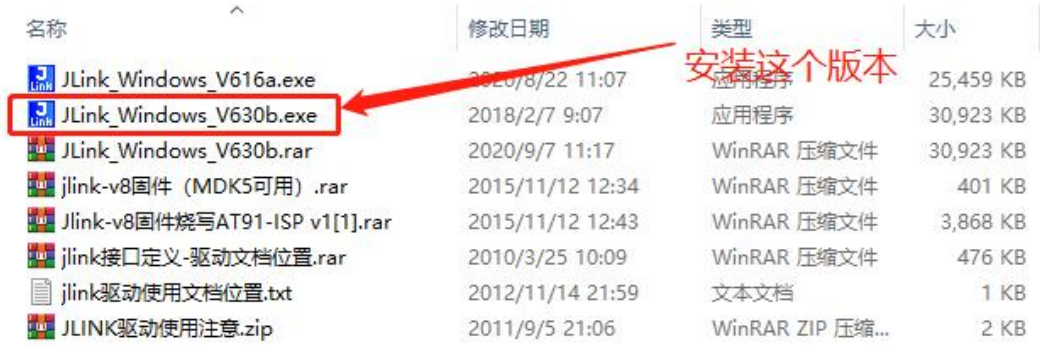
JLINK 驱动安装以及破解

Jlink 版本信息



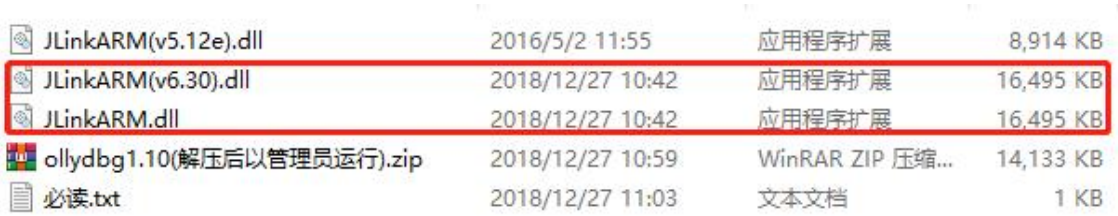
安装

请安装 JLink_Windows_V630b.exe，常规一步步就可以。



破解

请用破解过的 JLinkARM.dll 替换 JLink arm 动态链接库文件。
源文件如下：



目标文件如下：

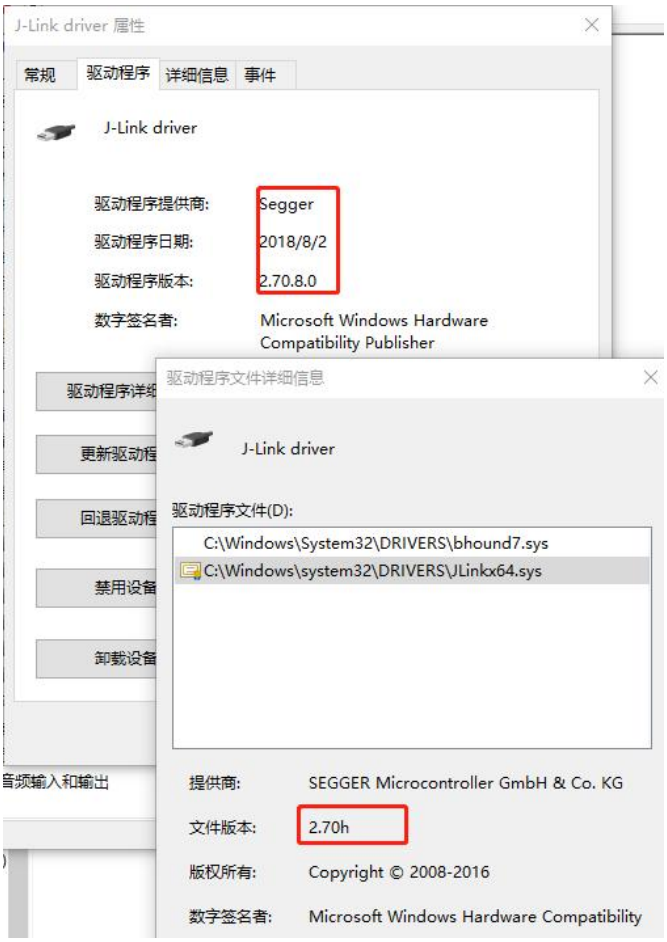
Program Files (x86) › IAR Systems › Embedded Workbench 8.2 › arm › bin

名称	修改日期	类型	大小
JLink x64.dll	2018/2/3 1:39	应用程序扩展	17,298 KB
jlinkarm.dll	2018/12/27 10:42	应用程序扩展	16,495 KB
JLinkARM_Backup00.dll	2015/5/12 18:30	应用程序扩展	7,727 KB
JLinkARM_org.dll	2015/12/9 19:52	应用程序扩展	8,394 KB
JLinkConfig.exe	2018/2/3 1:38	应用程序	181 KB
JLinkDevices.ref	2017/2/19 14:34	REF 文件	1 KB
JLinkGDBServer.exe	2018/2/3 1:38	应用程序	702 KB
JLinkGDBServerCL.exe	2018/2/3 1:38	应用程序	672 KB
JLinkRDI.dll	2018/2/3 1:39	应用程序扩展	305 KB
JLinkRemoteServer.exe	2017/6/6 18:45	应用程序	461 KB
JLinkSTM32.exe	2017/6/6 18:45	应用程序	86 KB
JLinkSTR91x.exe	2018/10/10 20:21	应用程序	146 KB

驱动信息

具体信息如下：

- Jungo Connectivity
- USB Video Class Device
- 安全设备
- 处理器
- 磁盘驱动器
- 存储控制器
- 打印队列
- 打印机
- 端口 (COM 和 LPT)
- 固件
- 计算机
- 监视器
- 键盘
- 人体学输入设备
- 软件设备
- 软件组件
- 声音、视频和游戏控制器
- 鼠标和其他指针设备
- 通用串行总线控制器
  - Alcor Micro USB 2.0 Card Reader
  - Intel(R) USB 3.0 可扩展主机控制器 - 1.0 (Microsoft)
  - J-Link driver
  - USB Composite Device
  - USB 根集线器(USB 3.0)
  - 通用 USB 集线器
- 网络适配器
- 系统设备
- 显示适配器
- 音频输入和输出

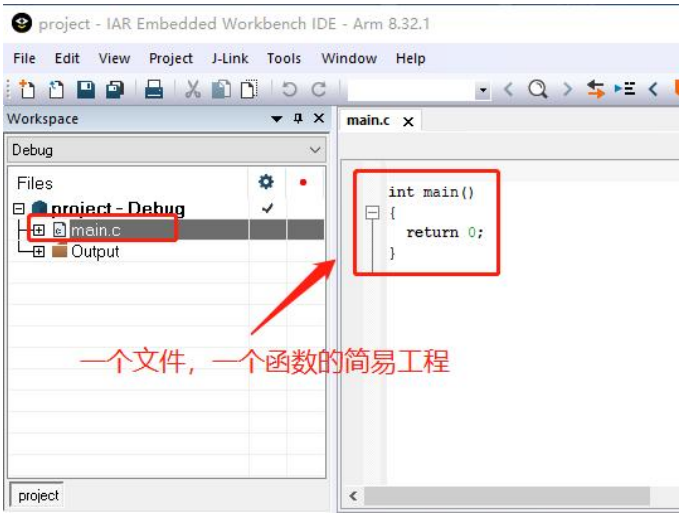


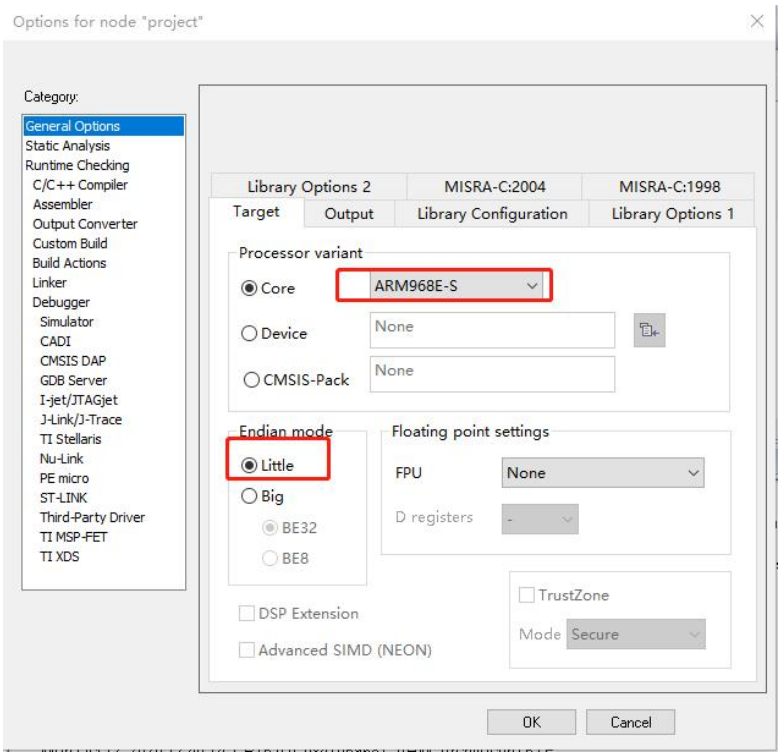
调试搭建

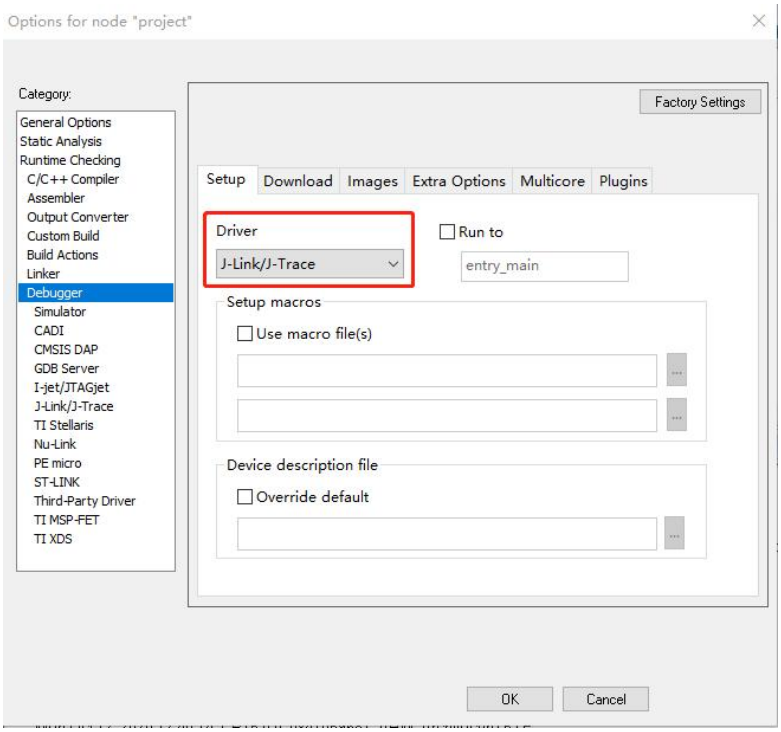
Tips：ELF 生成的时候，建议编译选项加上“**-O0 -gdwarf-2**”。调试的时候汇编和符号表对应会更直接。

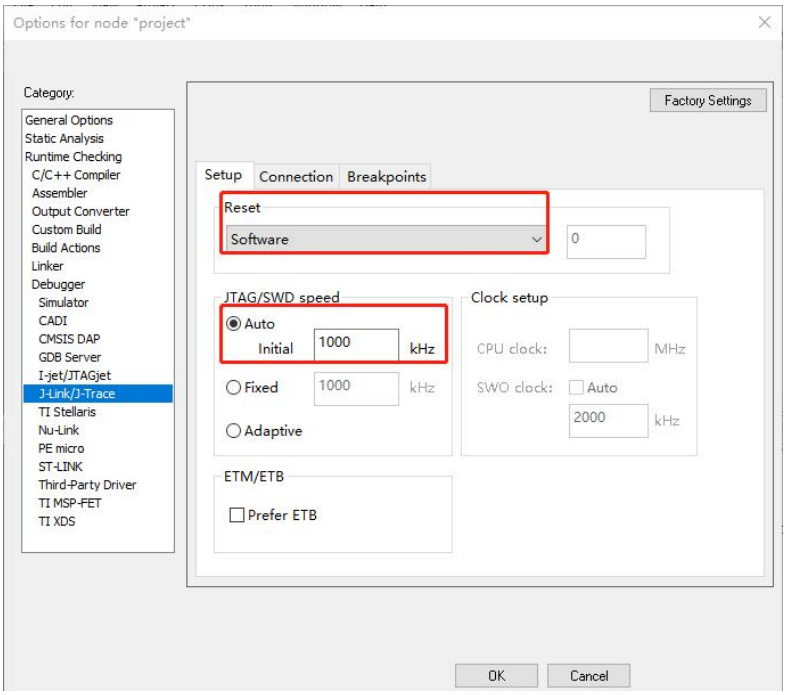
在编译的时候，编译器会从源文件中收集大量的信息，例如变量名、变量类型、变量所在行号、函数名、函数参数、函数的地址范围、行号和地址的对应关系等等，然后按照一种特定的格式写入到编译后的文件中。调试的时候，调试器便从文件中读取并解析这些信息，以产生可读性比较强的信息。**Dwarf2、dwarf3 就是一种调试信息存放格式**。

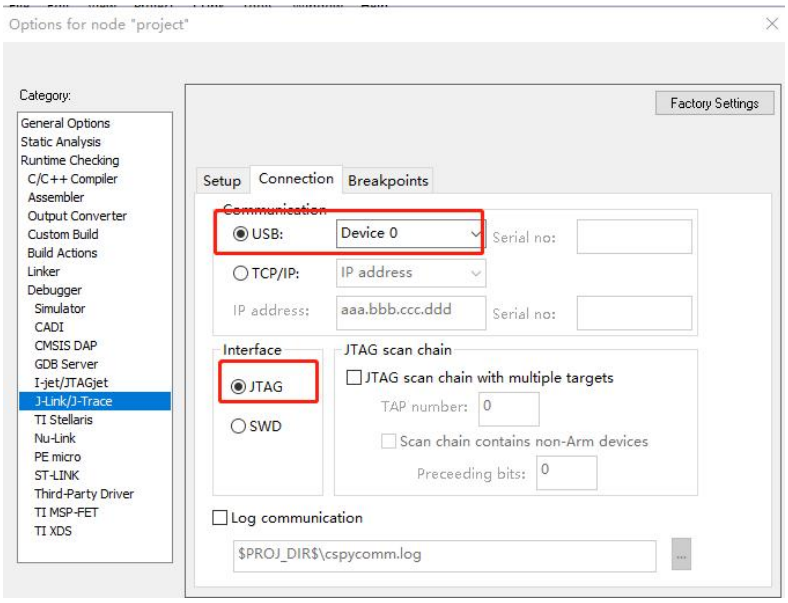
建立一个基于 Arm968ejs 的 IAR 简易工程




Options for node "project"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Library Options 2
MISRA-C:2004
MISRA-C:1998
Target
Output
Library Configuration
Library Options 1
Processor variant
Core
ARM968E-S
Device
None
CMSIS-Pack
None
Endian mode
Little
Big
BE32
BE8
Floating point settings
FPU
None
D registers
DSP Extension
Advanced SIMD (NEON)
TrustZone
Mode
Secure
OK
Cancel

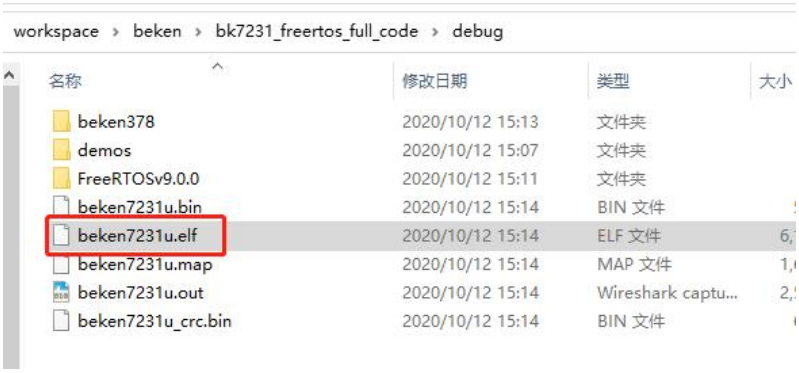

Options for node "project"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Setup
Download
Images
Extra Options
Multicore
Plugins
Driver
J-Link/J-Trace
Run to
entry_main
Setup macros
Use macro file(s)
Device description file
Override default
OK
Cancel


Options for node "project"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Setup
Connection
Breakpoints
Reset
Software
0
JTAG/SWD speed
Auto
Initial
1000
kHz
Fixed
1000
kHz
Adaptive
Clock setup
CPU clock:
MHz
SWO clock:
Auto
2000
kHz
ETM/ETB
Prefer ETB
OK
Cancel


Options for node "project"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Factory Settings
Setup
Connection
Breakpoints
Communication
USB:
Device 0
Serial no:
TCP/IP:
IP address
IP address:
aaa.bbb.ccc.ddd
Serial no:
Interface
JTAG
SWD
JTAG scan chain
JTAG scan chain with multiple targets
TAP number:
0
Scan chain contains non-Arm devices
Preceeding bits:
0
Log communication
$PROJ_DIR$\cspycomm.log

ELF 文件替换

workspace › beken › jlink_iar_debug › Debug › Exe
名称
修改日期
类型
大小
project.out
2020/10/12 11:23
Wireshark captu...

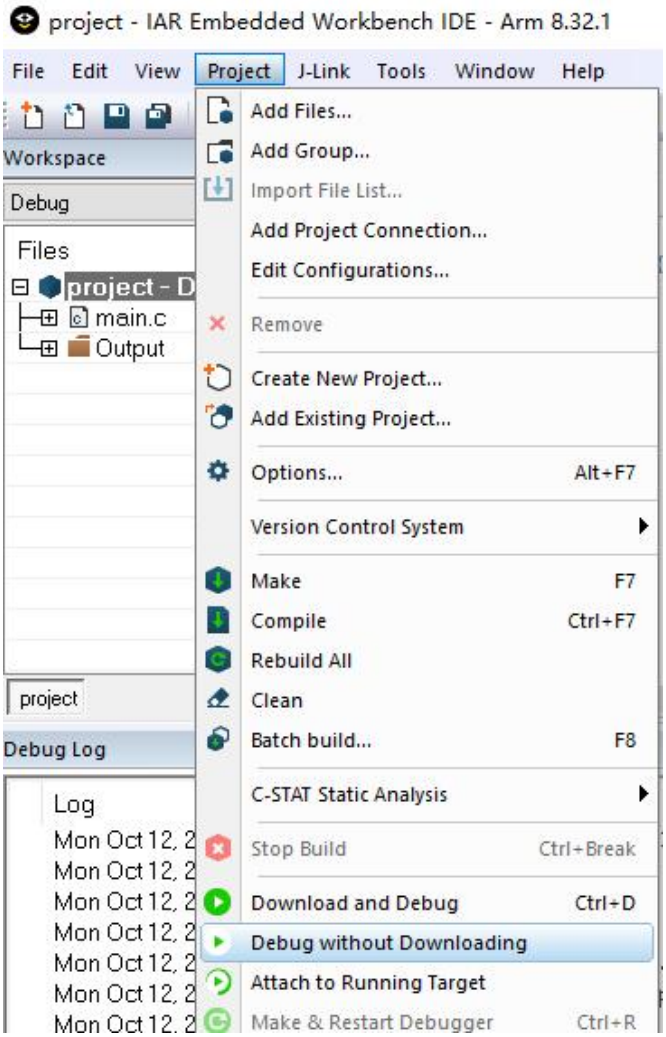

ertos_full_code\debug
workspace › beken › bk7231_freertos_full_code › debug
名称
修改日期
类型
大小
beken378
2020/10/12 15:13
文件夹
demos
2020/10/12 15:07
文件夹
FreeRTOSv9.0.0
2020/10/12 15:11
文件夹
beken7231u.bin
2020/10/12 15:14
BIN 文件
beken7231u.elf
2020/10/12 15:14
ELF 文件
beken7231u.map
2020/10/12 15:14
MAP 文件
beken7231u.out
2020/10/12 15:14
Wireshark captu...
beken7231u_crc.bin
2020/10/12 15:14
BIN 文件

- beken7231u.elf 文件**拷贝**到 IAR 工程目录的 Exe 目录，例如：jlink_iar_debug\Debug\Exe；
- **删除**源文件 project.out；
- 将 beken7231u.elf 文件**改名**为 project.out；

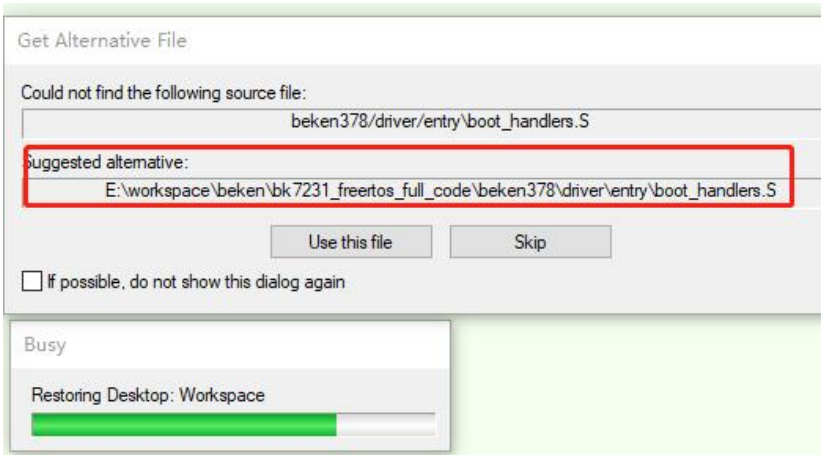
调试

针对 bk7231u 描述。如果您使用的开发板不是 bk7231u，请发挥主观能动性自由发挥之。

1. 给开发板供电，查收 JLink 仿真器；
   请使用 **V8 版本的 JLink 仿真器**，使用 V9 版本的仿真器现象诡异。请以怀疑或进取的态度阅读此段文字。
2. 使用 spi **下载**工具，或者 com 下载工具更新开发板固件；
   目的是更新运行的固件，**注意这个固件需要跟 elf 有个对应关系**。否则后面 JLink 调试的时候，会非常奇怪。
3. 点击 Debug without Downloading；



4. 选择源文件或源码路径



5. Jlink 信息

细节如下：

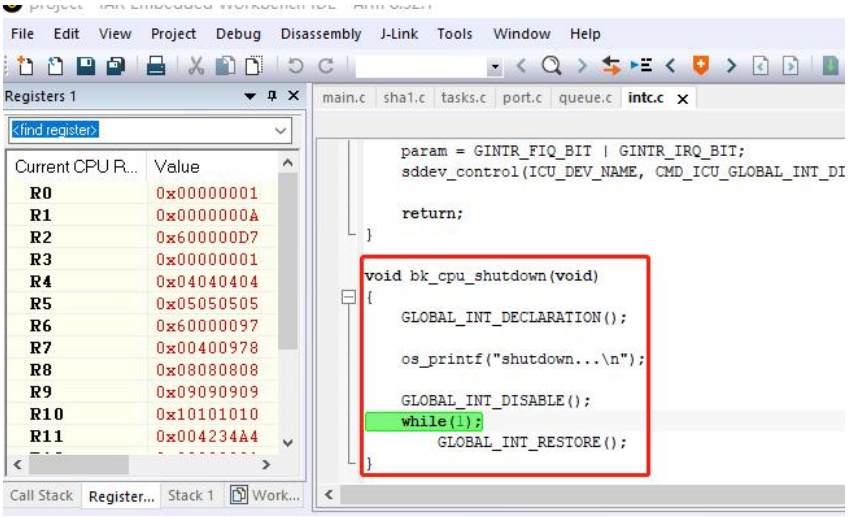
```
Mon Oct 12, 2020 19:34:34: IAR Embedded Workbench 8.32.1 (D:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)
Mon Oct 12, 2020 19:34:34: Device "ARM9" selected.
Mon Oct 12, 2020 19:34:34: JLINK command: ProjectFile = D:\workspace\beken\jlink_iar_debug\settings\project_Debug.jlink, return = 0
Mon Oct 12, 2020 19:34:34: Device "ARM9" selected.
Mon Oct 12, 2020 19:34:34: DLL version: V6.30 , compiled Jan 30 2018 18:44:37
Mon Oct 12, 2020 19:34:34: Firmware: J-Link ARM V8 compiled Nov 28 2014 13:44:46
Mon Oct 12, 2020 19:34:34: JTAG speed is initially set to: 1000 kHz
Mon Oct 12, 2020 19:34:34: Initial reset was performed
Mon Oct 12, 2020 19:34:34: TotalIRLen = 4, IRPrint = 0x01
Mon Oct 12, 2020 19:34:34: JTAG chain detection found 1 devices:
Mon Oct 12, 2020 19:34:34:  #0 Id: 0x15968001, IRLen: 04, ARM968E-S Core
Mon Oct 12, 2020 19:34:34: CP15.0.0: 0x41059681: ARM, Architecure 5TE
Mon Oct 12, 2020 19:34:34: J-Link: ARM9, 968 core
Mon Oct 12, 2020 19:34:34: Found 1 JTAG device, Total IRLen = 4:
Mon Oct 12, 2020 19:34:34:  #0 Id: 0x15968001, IRLen:  4, IRPrint: 0x1 ARM968E-S Core
Mon Oct 12, 2020 19:34:34: J-Link found 1 JTAG device(s), ARM core Id: 15968001 ARM9
Mon Oct 12, 2020 19:34:34: Adaptive clocking not supported for selected CPU core. Only supported for -S cores.
Mon Oct 12, 2020 19:34:34: Auto JTAG speed: 8000 kHz
Mon Oct 12, 2020 19:34:42: 0 bytes with download suppressed
Mon Oct 12, 2020 19:34:42: Loaded debugee: D:\workspace\beken\jlink_iar_debug\Debug\Exe\project.out
Mon Oct 12, 2020 19:34:42: Software reset was performed
Mon Oct 12, 2020 19:34:42: Target reset
```

Debug 集锦

写在调试之前，“尽信书不如无书”，IAR Debug 有时候也出现不准确的 callstack 和函数入参。请配合 micoder 调试来定位问题。

Attach to Running Target

TODO



```
pretetch abort
Current regs:
r00:0x00000000 r01:0x00423514 r02:0x00405924 r03:0x00000000
r04:0x04040404 r05:0x05050505 r06:0x06060606 r07:0x004234a8
r08:0x08080808 r09:0x09090909 r10:0x10101010
fp :0x004234a4 ip :0x12121212
sp :0x00400998 lr :0x0000000c pc :0x0000000c
SPSR:0x60000093
CPSR:0x60000097

separate regs:
SYS:cpsr r8-r14
0x6000009f
0x08080808
0x09090909
0x10101010
0x004234a4
0x12121212
0x00423460
0x0008c1a0
IRQ:cpsr spsr r8-r14
0x60000092
0x6000001f
0x08080808
0x09090909
0x10101010
0x004234a4
0x12121212
0x004021d0
0x0008afd8
FIR:cpsr spsr r8-r14
```

ARM9 cpu 速率疑问

软件版本：http://192.168.0.6/wifi/bk7231_freertos_full_code f2033aa015bc456c3255a816698fd3585dbba3cc
编译选择添加：-O0 -gdwarf-2
现象：不连接 JLink，串口输入 sta 113ch11 113ch11123 或者 sta wifi-team stm32f215，都能成功。Log 如下图：

```
sizeof(wpa_supplicant)=760
mm_add_if_req_handler:0
hapd_intf_add_vif,type:2, s:0, id:0
wpa_dInit
wpa_supplicant_req_scan
Setting scan request: 0.100000 sec
MANUAL_SCAN_REQ
enter low level!
mac c8:47:8c:42: 0:48
leave low level!
[net]addvif_idx:0
wpa_supplicant_scan
wpa_supplicant_scan 863
wpa_drv_scan
wpa_send_scan_req
no ht in scan
scan_start_req_handler
scan_cancel_req_handler
scan_send_cancel_cfm
wpa_driver_scan_cb
wpa_get_scan_rst:1
cipher2security 2 2 16 16
PSKC: end
wpa_supplicant_connect
Cancelling scan request
wpa_driver_associate
sm_auth_send:1
sm_auth_handler
sm_assoc_rsp_handler
rc_init: station_id=0 format_mod=0 pre_type=0 short_gi=0 max_bw=0
rc_init: nss_max=0 mcs_max=255 r_idx_min=0 r_idx_max=11 no_samples=10
mm_set_vif_state_req_handler
chan_bcn_detect_start
--------SM_CONNECT_IND_ok
wpa_driver_assoc_cb
Cancelling scan request
WPA: TK 7c66ef3cae13ac31cea29b918f276bb1
hapd_intf_add_key CCMP
add sta_mgmt_get_sta
sta:0, vif:0, key:0
sta_mgmt_add_key
add hw key idx:24
WPA: GTK 109b6dd19f117e49afa963e76a228dfc
hapd_intf_add_key CCMP
add is_broadcast_ether_addr
sta:255, vif:0, key:1
add hw key idx:1
ctrl_port_hdl:1
me_set_ps_disable:847 0 0 0 0 1
dis set ps 1!!
sta_ip_start

configuring interface mlan (with DHCP client)
dhcp_check_status_init_timer:20000
new dtim period:2

new ie: 0 : 0 0 0 0 0 0 0 0 0
new ie: 1 : 82 84 b 16 24 30 48 6c
new ie: 3 : 1
new ie: 30 : 1 0 0 f ac 4 1 0 0 f ac 4 1 0 0 f ac 2 c 0

ip_addr: 1b2ca8c0
```

连接 Jlink，串口发生同样命令 sta 113ch11 113ch11123 或者 sta wifi-team stm32f215，偶尔出现连接 router 或 ap 失败。Log 如下图：

```
sta_Command
ssid:wifi-team key:stm32f215

# normal_connect
 1663: [sa_sta]MM_RESET_REQ
[bk]tx_txdesc_flush
[sa_sta]ME_CONFIG_REQ
[sa_sta]ME_CHAN_CONFIG_REQ
[sa_sta]MM_START_REQ
[THD]pskc:[tcb]41eb88 [stack]41e780-41eb80:1024:4
PSKC: ssid wifi-team, passphrase stm32f215
sizeof(wpa_supplicant)=760
mm_add_if_req_handler:0
hapd_intf_add_vif,type:2, s:0, id:0
wpa_dInit
wpa_supplicant_req_scan
Setting scan request: 0.100000 sec
MANUAL_SCAN_REQ
wpa_supplicant_scan
wpa_supplicant_scan 863
wpa_drv_scan
wpa_send_scan_req
no ht in scan
scan_start_req_handler
scan_cancel_req_handler
scan_send_cancel_cfm
wpa_driver_scan_cb
wpa_get_scan_rst:1
cipher2security 2 2 16 16
enter low level!
mac c8:47:8c:42: 0:48
leave low level!
[netladdvif idx:0
PSKC: end
wpa_supplicant_connect
```

这种情况下串口命令还可以工作，上下文切换正常。怀疑算 pmk 太长时间导致异常；
根本原因：

BSS: Remove id 0 BSSID 4c:d1:a1:c5:38:e4 SSID '113ch11' due to wpa_bss_flush_by_age

没有发起 scan。细节待定

备注

DWARF

DWARF 的全称是"**Debugging With Attributed Record Formats**"，遵从 GNU FDL 授权。现在已经有 dwarf1，dwarf2，dwarf3 三个版本。

Dwarf 最初被贝尔实验室设计用来供 Unix System V 的 sdb 调试器使用，并且在 1989 年被 Unix 国际化部门的 PLSIG (Programming Languages Special Interest Group)标准化成为 dwarf1.0。但是 dwarf1 有着很多明显的缺点，于是 PLSIG 继续开发，改正了缺点，并加入了对 C++等语言的支持，并在 1990 年正式公布了 dwarf2 的标准草案。但是稍后由于一些原因，PLSIG 被解散，dwarf 的开发陷入到多个并不合作的组织中间，造成 dwarf2 的一些实现细节要取决于特定的编译器。这种情况一直持续到 1999 年，开发工作受到了来自实现对 HP/Inter IA-64 架构提供较好支持的推动，成立了 dwarf 委员会，dwarf 的原作者担任负责人，

开始了 dwarf3 的开发，并于 2006 年 1 月份推出 dwarf3.0，同时为了解决分歧，dwarf 委员会加入了自由标准组织，在自由标准组织与来自 Linux 基金会的 OSDL(Open Source Development Labs)合并后，dwarf 重返独立状态并创建了自己的网站：**dwarfstd.org**。

这三个版本中，**dwarf2 对 dwarf1 的改变很大，dwarf3 大多是对 dwarf2 的扩充**。

现在 dwarf 已经是一种独立的标准，可以支持 C、C++、JAVA、Fortran 等语言。

在了解了 dwarf 的历史之后，来看一下如何查看 dwarf 所包含的调试信息内容。查看内容的工具常用的有四种：

<!--[if !supportLists]-->1. <!--[endif]-->readelf

GNU 提供的二进制工具，功能很多，并不限于读 dwarf 信息

<!--[if !supportLists]-->2. gdb

这个就不用多说了吧，^_^

<!--[if !supportLists]-->3. drawfdump

是一个被打包在 libdwarf 内的程序

<!--[if !supportLists]-->4. <!--[endif]-->libdwarf

是一个封装好的 C 库 API，用来读取 dwarf 信息

在这里我们主要使用 readelf 工具。

先写一个简单的 C 程序，如下：

```
int add(int, int);

int main()
{
    int i, j;

    for(i = 0; i < 100; i += 5, j = i * 5)
        add(i, j);

    return 0;
}

int add(int a, int b)
{
    return a + b;
}
```

然后使用 gcc –g hello.c –o hello 编译。生成 hello 文件。

Hello 文件是 elf 格式的，elf 一般由多个节(section)组成，不熟悉的可以看前面两篇关于 elf 文件格式的文章。调试信息被包含在某几个节中，**如果是用 dwarf2 格式编译的，这些节的名字一般是以.debug 开头，如.debug_info，.debug_line，.debug_frame 等**，如果是用 dwarf1 格式编译的，这些节的名字一般是.debug，.line 等。现在的编译器默认大多数是 dwarf2 格式编译，当然可以通过 gcc 的编译选项改变。

现在来看 hello 文件都包含了哪些调试信息。

首先来看都包含了哪些调试节，使用 readelf –S hello 命令，产生如下输出(已删一些无关内容)：

```
[Nr] Name              Type            Addr     Off    Size
 [ 0]                  NULL            00000000 000000 000000
 [ 1] .text            PROGBITS        00008000 008000 0006c4
 [ 2] .ARM.exidx       ARM_EXIDX       000086c4 0086c4 000008
 [ 3] .data            PROGBITS        000086d0 0086d0 000520
 [ 4] .bss             NOBITS          00008bf0 008bf0 000020
 [ 5] .debug_aranges   PROGBITS        00000000 008bf0 000020
 [ 6] .debug_pubnames  PROGBITS        00000000 008c10 000023
 [ 7] .debug_info      PROGBITS        00000000 008c33 0000cc
 [ 8] .debug_abbrev    PROGBITS        00000000 008cff 00006b
 [ 9] .debug_line      PROGBITS        00000000 008d6a 00003e
 [10] .debug_frame     PROGBITS        00000000 008da8 000188
 [11] .debug_loc       PROGBITS        00000000 008f30 000054
 [12] .ARM.attributes  ARM_ATTRIBUTES  00000000 008f84 000010
 [13] .comment         PROGBITS        00000000 008f94 000032
 [14] .shstrtab        STRTAB          00000000 008fc6 0000ad
 [15] .symtab          SYMTAB          00000000 00931c 000780
 [16] .strtab          STRTAB          00000000 009a9c 000416
```

可见一共包含了 17 个节，其中 7 个调试信息的节。

在来看一下各个调试信息节包含的内容，使用 readelf –w* hello 命令，*是调试节名的第一个字母，如-wi 就是查看.debug_info 节的内容，-wl 就是查看.debug_line 节的内容。

对于一个调试文件，.debug_info 和.debug_line 节是必须有的，其他的不见得。同时也可以自己写链接脚本实现对所有节(不局限于调试节)的控制，如指定每个节的基址等。

.debug_info 基本包含了一个源文件内部的大部分信息，如函数、参数、变量、类型等等，我们看一下它的输出：

The section .debug_info contains:

```
Compilation Unit @ offset 0x0:
  Length:        200
```

```
  Version:       2
  Abbrev Offset: 0
  Pointer Size: 4
<0><b>: Abbrev Number: 1 (DW_TAG_compile_unit)
    DW_AT_stmt_list    : 0
    DW_AT_high_pc      : 0x8248
    DW_AT_low_pc       : 0x81ac
    DW_AT_producer     : GNU C 4.1.1
    DW_AT_language     : 1     (ANSI C)
    DW_AT_name         : hello.c
    DW_AT_comp_dir     : C:\Program Files\CodeSourcery\Sourcery G++\bin
<1><5c>: Abbrev Number: 2 (DW_TAG_subprogram)
    DW_AT_sibling      : <92>
    DW_AT_external     : 1
    DW_AT_name         : main
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 5
    DW_AT_type         : <92>
    DW_AT_low_pc       : 0x81ac
    DW_AT_high_pc      : 0x8214
    DW_AT_frame_base : 0     (location list)
<2><79>: Abbrev Number: 3 (DW_TAG_variable)
    DW_AT_name         : i
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 6
    DW_AT_type         : <92>
    DW_AT_location     : 2 byte block: 91 68        (DW_OP_fbreg: -24)
<2><85>: Abbrev Number: 3 (DW_TAG_variable)
    DW_AT_name         : j
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 6
    DW_AT_type         : <92>
    DW_AT_location     : 2 byte block: 91 6c        (DW_OP_fbreg: -20)
<1><92>: Abbrev Number: 4 (DW_TAG_base_type)
    DW_AT_name         : int
    DW_AT_byte_size    : 4
    DW_AT_encoding     : 5     (signed)
<1><99>: Abbrev Number: 5 (DW_TAG_subprogram)
    DW_AT_external     : 1
    DW_AT_name         : add
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 15
    DW_AT_prototyped : 1
    DW_AT_type         : <92>
    DW_AT_low_pc       : 0x8214
    DW_AT_high_pc      : 0x8248
    DW_AT_frame_base : 0x2a       (location list)
<2><b2>: Abbrev Number: 6 (DW_TAG_formal_parameter)
    DW_AT_name         : a
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 14
    DW_AT_type         : <92>
    DW_AT_location     : 2 byte block: 91 6c        (DW_OP_fbreg: -20)
<2><be>: Abbrev Number: 6 (DW_TAG_formal_parameter)
    DW_AT_name         : b
    DW_AT_decl_file    : 1
    DW_AT_decl_line    : 14
    DW_AT_type         : <92>
```

```
    DW_AT_location     : 2 byte block: 91 68        (DW_OP_fbreg: -24)
```

.debug_line 包含了所有地址和源文件行的对应信息，内容如下：

```
Dump of debug contents of section .debug_line:

 Length:                      58
 DWARF Version:               2
 Prologue Length:             30
 Minimum Instruction Length:    2
 Initial value of 'is_stmt':        1
 Line Base:                   -5
 Line Range:                  14
 Opcode Base:                 13

 Opcodes:
 Opcode 1 has 0 args
 Opcode 2 has 1 args
 Opcode 3 has 1 args
 Opcode 4 has 1 args
 Opcode 5 has 1 args
 Opcode 6 has 0 args
 Opcode 7 has 0 args
 Opcode 8 has 0 args
 Opcode 9 has 1 args
 Opcode 10 has 0 args
 Opcode 11 has 0 args
 Opcode 12 has 1 args

 The Directory Table is empty.

 The File Name Table:
 Entry    Dir    Time        Size Name
 1 0      0      0      hello.c

 Line Number Statements:
 Extended opcode 2: set Address to 0x81ac
 Special opcode 9: advance Address by 0 to 0x81ac and Line by 4 to 5
 Special opcode 120: advance Address by 16 to 0x81bc and Line by 3 to 8
 Special opcode 90: advance Address by 12 to 0x81c8 and Line by 1 to 9
 Special opcode 88: advance Address by 12 to 0x81d4 and Line by -1 to 8
 Advance PC by constant 34 to 0x81f6
 Special opcode 78: advance Address by 10 to 0x8200 and Line by 3 to 11
 Special opcode 34: advance Address by 4 to 0x8204 and Line by 1 to 12
 Special opcode 120: advance Address by 16 to 0x8214 and Line by 3 to 15
 Special opcode 174: advance Address by 24 to 0x822c and Line by 1 to 16
 Special opcode 90: advance Address by 12 to 0x8238 and Line by 1 to 17
 Advance PC by 16 to 0x8248
 Extended opcode 1: End of Sequence
```

其他的信息内容不在一一列举，有兴趣的可以自己查看。

KEIL + ARM9 + GNU Tools ARM Embedded + ELF + JLINK

安装以及破解

调试

Eclipse + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB

bk72xx-eclipse-compileAndDebug guide

bk7231 是一颗 2.4GHz 802.11b/g/n 数据传输 SoC，芯片集成了 802.11b/g/n 从射频到 MAC 层的所有的软件和硬件功能，其 CPU 是 ARM968E-S。

【备注：bk7231s/bk7231t/bk7231n/bk7231u/bk7221u/bk7251 等 soc，都可以使用此文档来配置 jtag 调试方法】

随着 bk7231 客户推广和二次开发需求，软件项目的编译链由 armcc 改变为 arm-none-eabi-gcc，开发 IDE 由 mdk keil 改为 eclipse。对于编译、调试等细节描述如以下章节。注意：以下章节基于笔者的硬软件环境和此次使用实践。仅作参考，建议**读者理解之后使用**。

电脑硬软件信息：

查看有关计算机的基本信息

Windows 版本

Windows 7 旗舰版

版权所有 © 2009 Microsoft Corporation。保留所有权利。

Service Pack 1

系统

制造商:	Windows Microsoft®
分级:	4.6 Windows 体验指数
处理器:	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.30 GHz
安装内存(RAM):	8.00 GB (7.86 GB 可用)
系统类型:	64 位操作系统
笔和触摸:	没有可用于此显示器的笔或触控输入

笔者电脑系统是 **windows7，64bit。**

JDK--java development kit

下载地址

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u141 checksum

Java SE Development Kit 8u141

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.88 MB	jdk-8u141-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u141-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.66 MB	jdk-8u141-linux-i586.rpm
Linux x86	179.4 MB	jdk-8u141-linux-i586.tar.gz
Linux x64	162.11 MB	jdk-8u141-linux-x64.rpm
Linux x64	176.92 MB	jdk-8u141-linux-x64.tar.gz
Mac OS X	226.6 MB	jdk-8u141-macosx-x64.dmg
Solaris SPARC 64-bit	139.84 MB	jdk-8u141-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.17 MB	jdk-8u141-solaris-sparcv9.tar.gz
Solaris x64	140.59 MB	jdk-8u141-solaris-x64.tar.Z
Solaris x64	97.01 MB	jdk-8u141-solaris-x64.tar.gz
Windows x86	190.95 MB	jdk-8u141-windows-i586.exe
Windows x64	197.78 MB	jdk-8u141-windows-x64.exe

安装步骤

一步步安装，next→next→…→finish

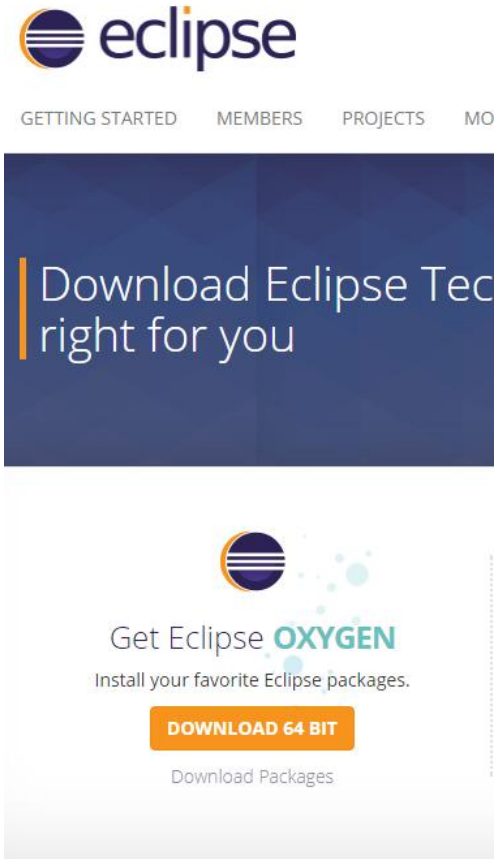
注意点

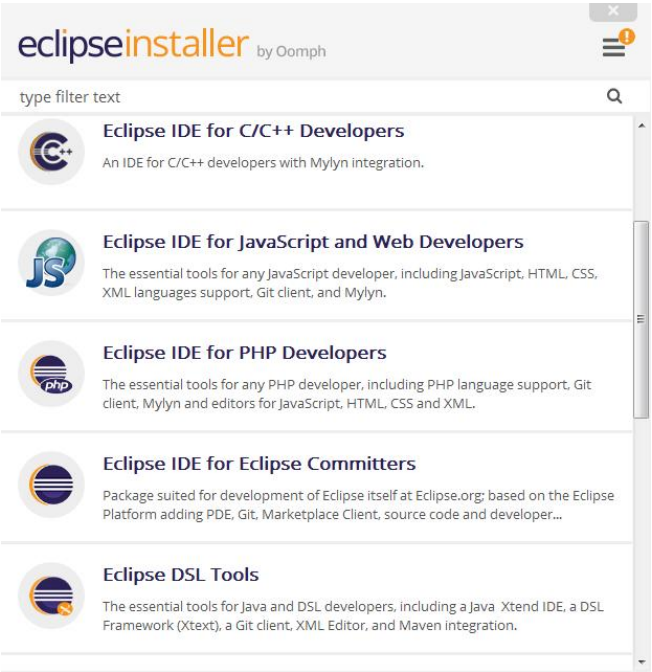
下载的 jdk 文件依赖使用者电脑系统。笔者电脑是 win7 64bit，所以下载文件为：jdk-8u141-windows-x64.exe

Eclipse IDE

下载地址

http://www.eclipse.org/downloads/





请选择：**Eclipse IDE for C/C++ Developers**

安装步骤

一步步安装，next→next→…→finish

注意点

下载的 eclipse 文件依赖使用者电脑系统。笔者电脑是 win7 64bit，所以下载文件为 eclipse-inst-win64.exe

llg plug-in

【备注：

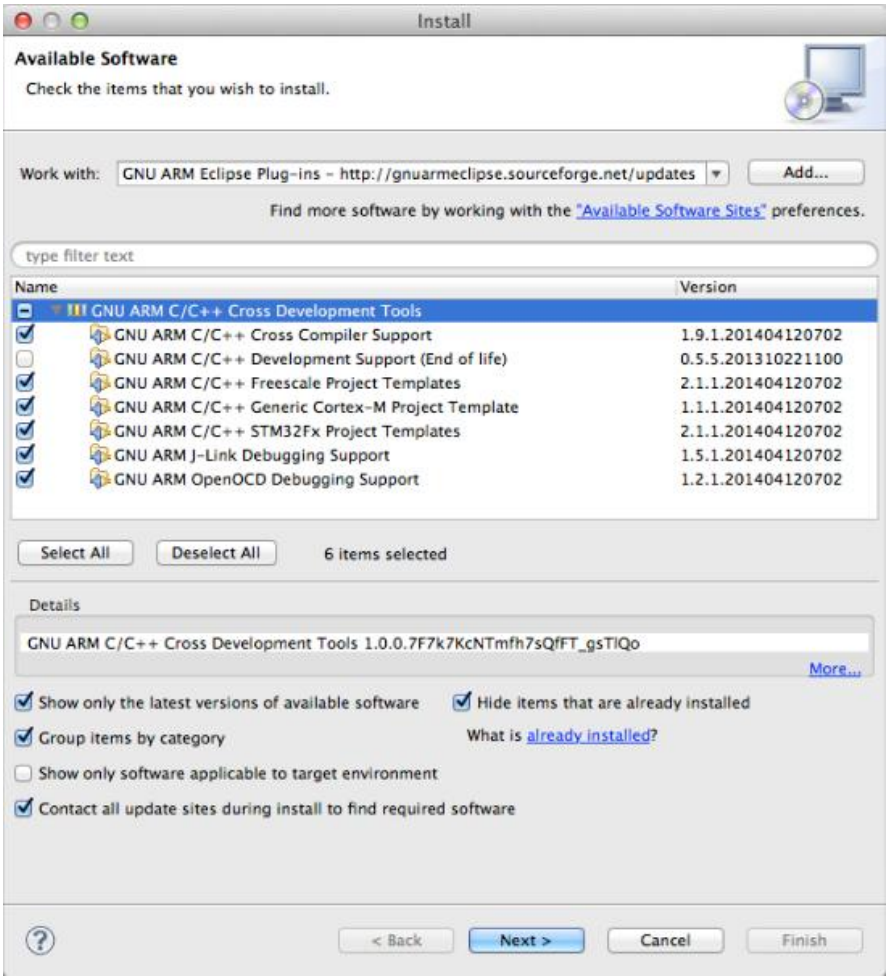
These plug-ins provide build and debug extensions for Eclipse CDT (C/C++ Development Tools) for 32/64-bit GNU ARM toolchains like GNU Tools for Embedded, Linaro, etc, ready to run STM32Fx project templates and full integration for advanced J-Link JTAG/SWD probes, including SWO tracing console】

下载地址

安装步骤

在线安装

http://gnuarmeclipse.sourceforge.net/updates



离线安装

https://sourceforge.net/projects/gnuarmeclipse/files/Current%20Releases/3.x/

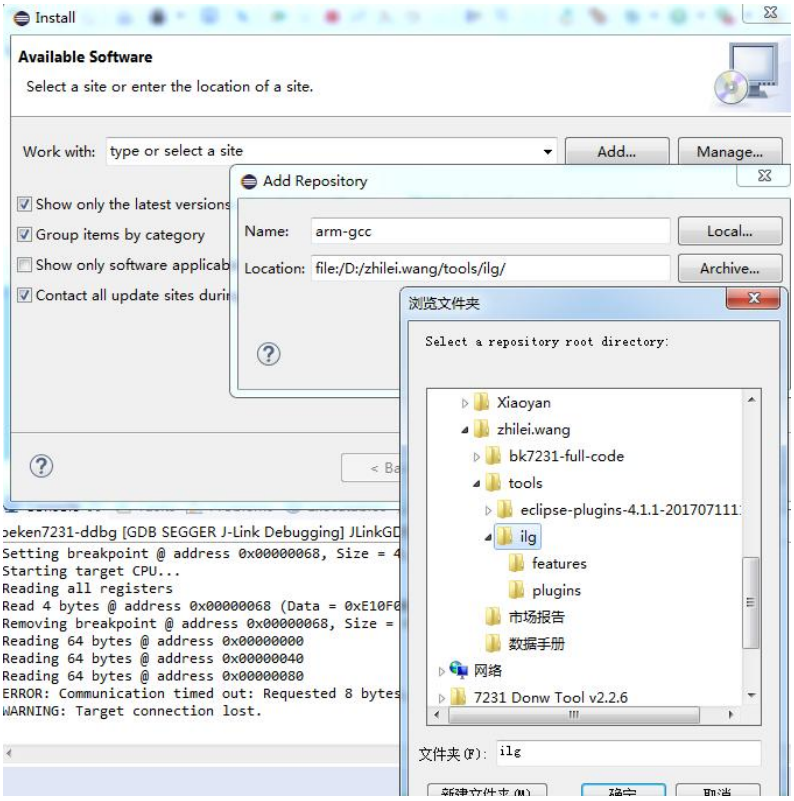
https://github.com/gnu-mcu-eclipse/eclipse-plugins/releases

1》 打开 eclipse；

2》 点击菜单 Help—》Install New Software;

3》 点击 Add—》Local；选择 ilg 所在目录，填充 Name 文本框；

4》 全部选中，Next 至 Finish。（注意：建议你这个时候**断开网络**，否则可能遇到很慢很慢。因为这里 IDE 会试图连接网络）



注意点

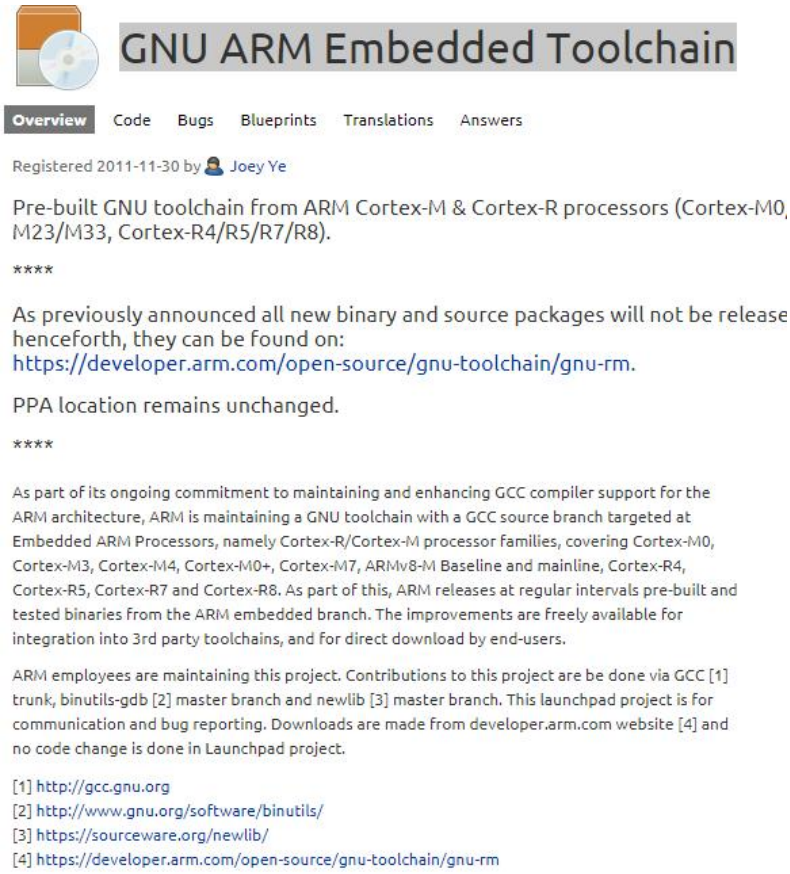
参考：https://gnu-mcu-eclipse.github.io/plugins/install/#plug-ins-installupdate

离线安装 ilg 的时候，建议断开电脑网络。[zhoujunlong: 实在不行，你联网试试]。

GNU ARM Embedded Toolchain

下载地址

https://launchpad.net/gcc-arm-embedded



GNU ARM Embedded Toolchain

Overview Code Bugs Blueprints Translations Answers

Registered 2011-11-30 by Joey Ye

Pre-built GNU toolchain from ARM Cortex-M & Cortex-R processors (Cortex-M0, M23/M33, Cortex-R4/R5/R7/R8).

****

As previously announced all new binary and source packages will not be release henceforth, they can be found on:
https://developer.arm.com/open-source/gnu-toolchain/gnu-rm.

PPA location remains unchanged.

****

As part of its ongoing commitment to maintaining and enhancing GCC compiler support for the ARM architecture, ARM is maintaining a GNU toolchain with a GCC source branch targeted at Embedded ARM Processors, namely Cortex-R/Cortex-M processor families, covering Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M0+, Cortex-M7, ARMv8-M Baseline and mainline, Cortex-R4, Cortex-R5, Cortex-R7 and Cortex-R8. As part of this, ARM releases at regular intervals pre-built and tested binaries from the ARM embedded branch. The improvements are freely available for integration into 3rd party toolchains, and for direct download by end-users.

ARM employees are maintaining this project. Contributions to this project are be done via GCC [1] trunk, binutils-gdb [2] master branch and newlib [3] master branch. This launchpad project is for communication and bug reporting. Downloads are made from developer.arm.com website [4] and no code change is done in Launchpad project.

[1] http://gcc.gnu.org
[2] http://www.gnu.org/software/binutils/
[3] https://sourceware.org/newlib/
[4] https://developer.arm.com/open-source/gnu-toolchain/gnu-rm

安装步骤

一步步安装，next→next→…→finish

注意点

安装目录**尽量不要选择有空格的路径**。笔者下载的文件：gcc-arm-none-eabi-5_4-2016q3-20160926-win32.exe

J-link driver install

下载地址

https://www.segger.com/downloads/jlink

安装步骤

笔者选择 Setup_JLink_V512.zip，其他的驱动你可以自己试试。看这个驱动行不行，运行 JLinkGDBServer.exe，看能不能正常工作；

一步步安装，next→next→…→finish。

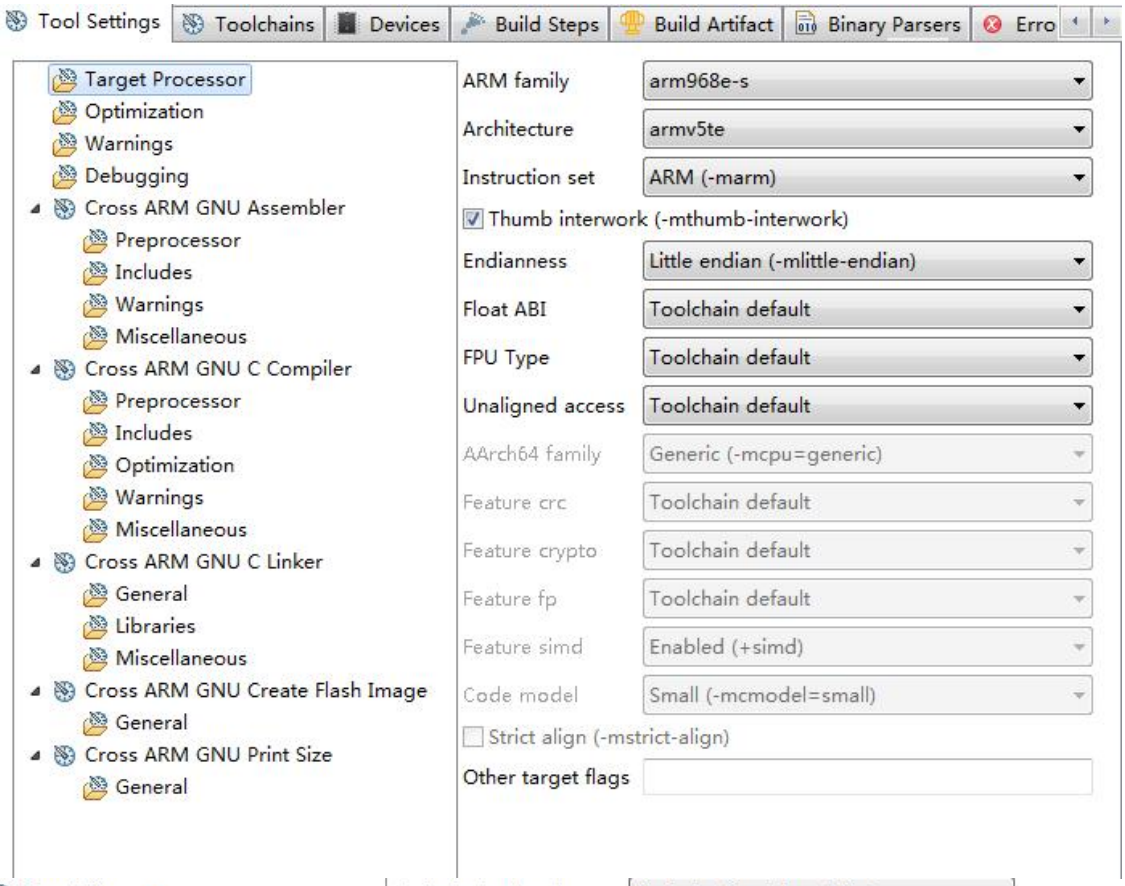
注意点

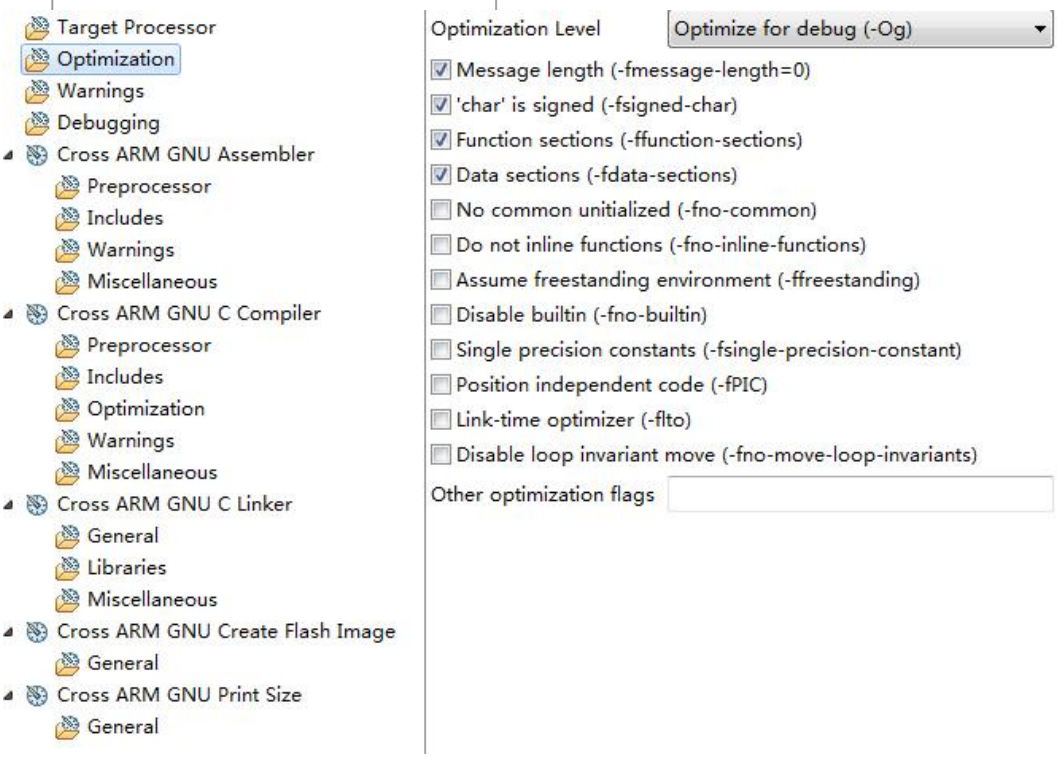
1，j-link 仿真器，JLinkGDBServer.exe 可能会提示它是 clone 的。建议推荐你或者你的公司买正版。正版光荣；

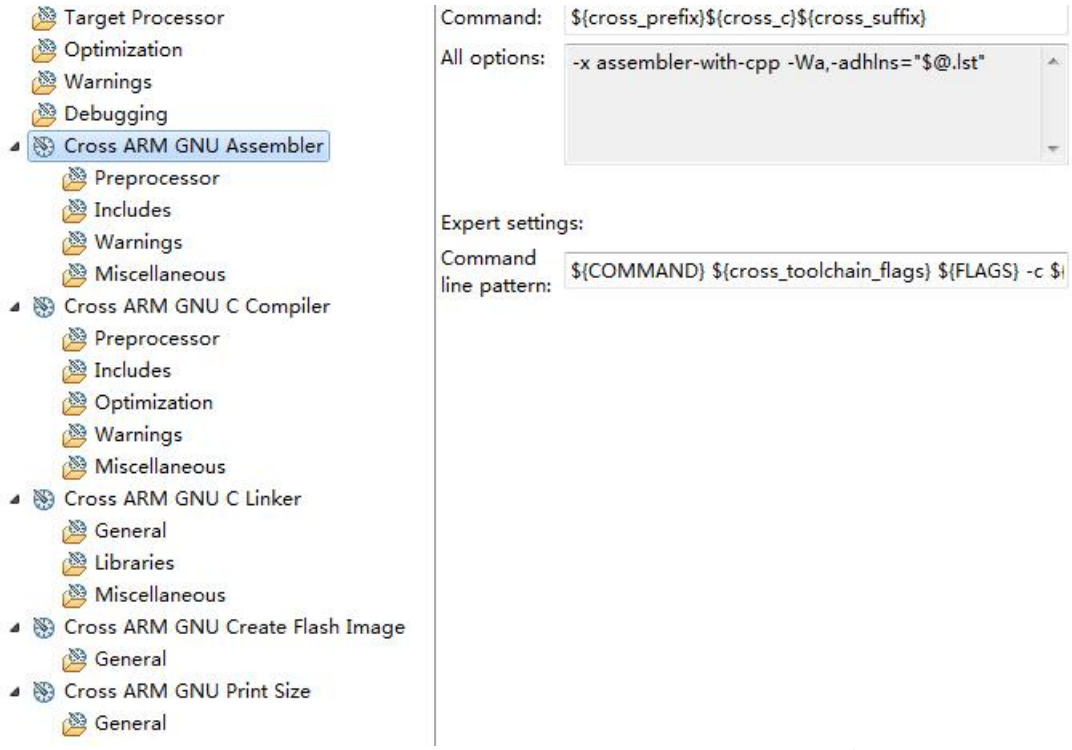
2，如果提示 j-link 是 clone 的，可以降低你的 j-link 版本，或者更新 j-link 固件；

3，Setup_JLink_V512.zip 笔者验证过，可以使用。

Compile setting







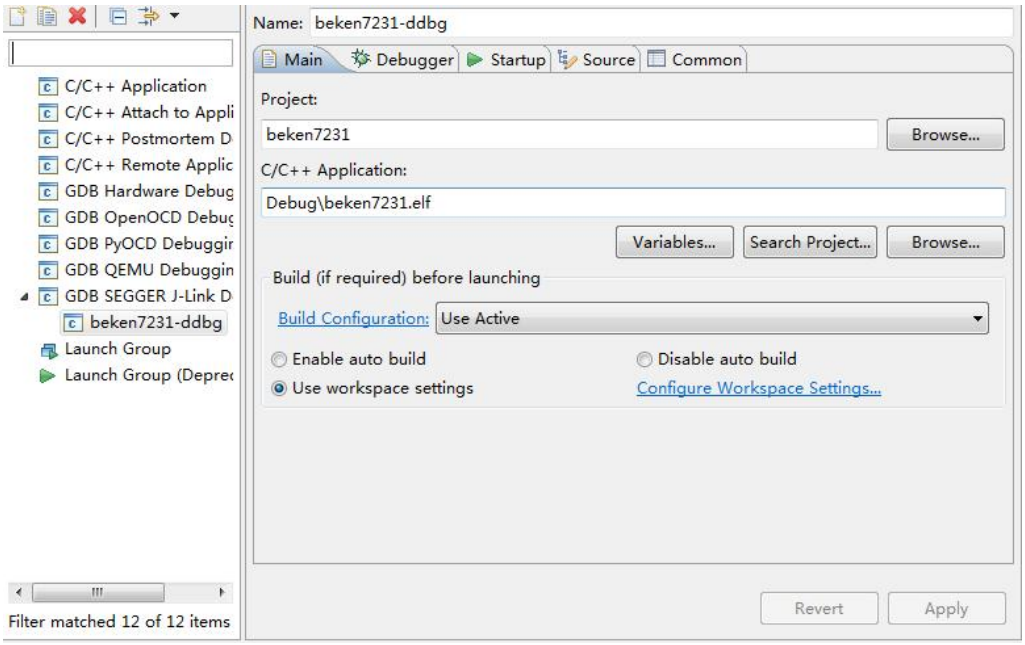
备注：编译配置建议在 bk7231 的工程配置（.cproject/.cproject）基础上更改或者优化。

J-link debugging setting

创建调试配置

点击 eclipse 菜单 Run—》Debug Configurations，双击 GDB SEGGER J-Link Debugging

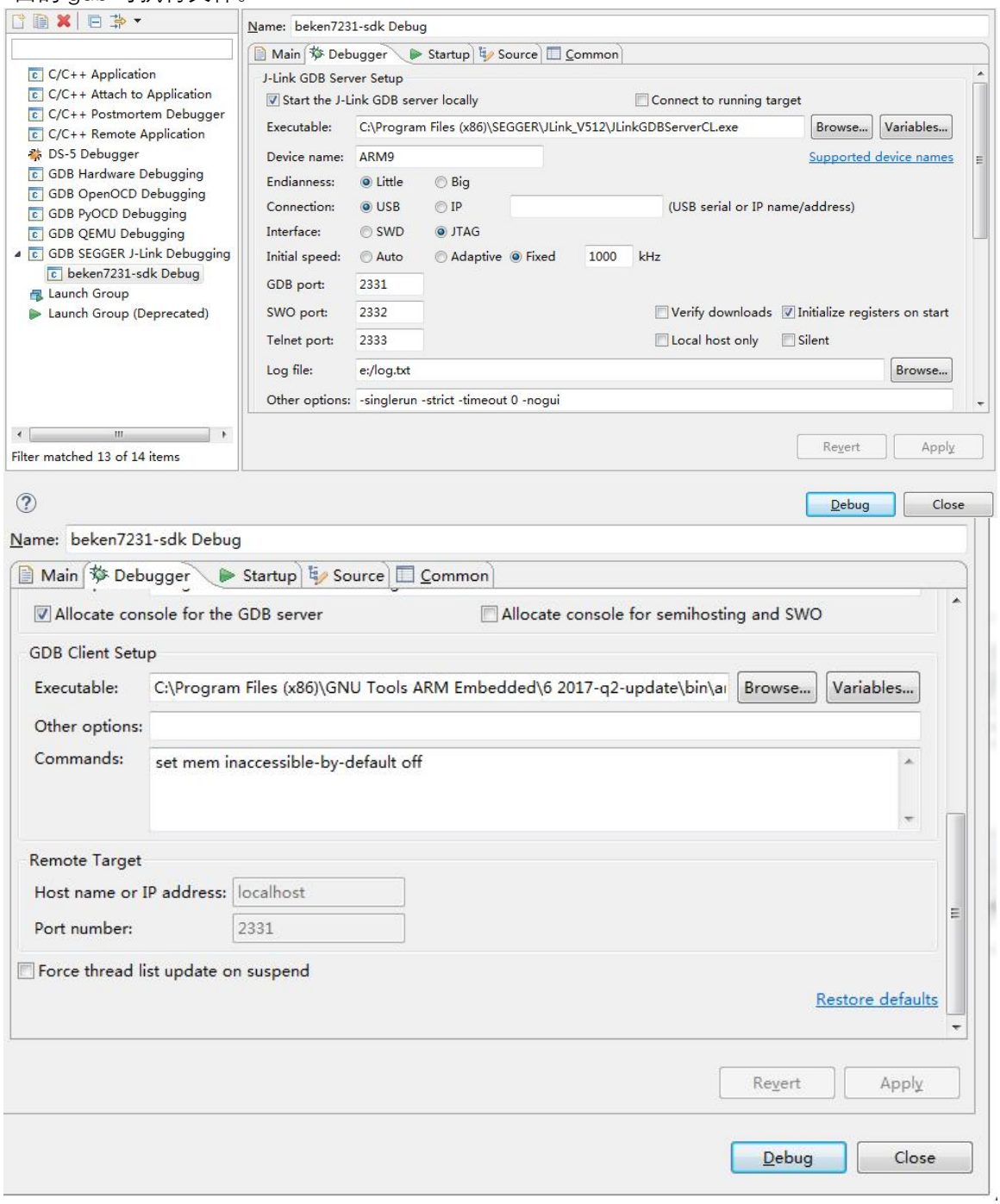
配置之 Main 属性页



配置之 Debugger 属性页

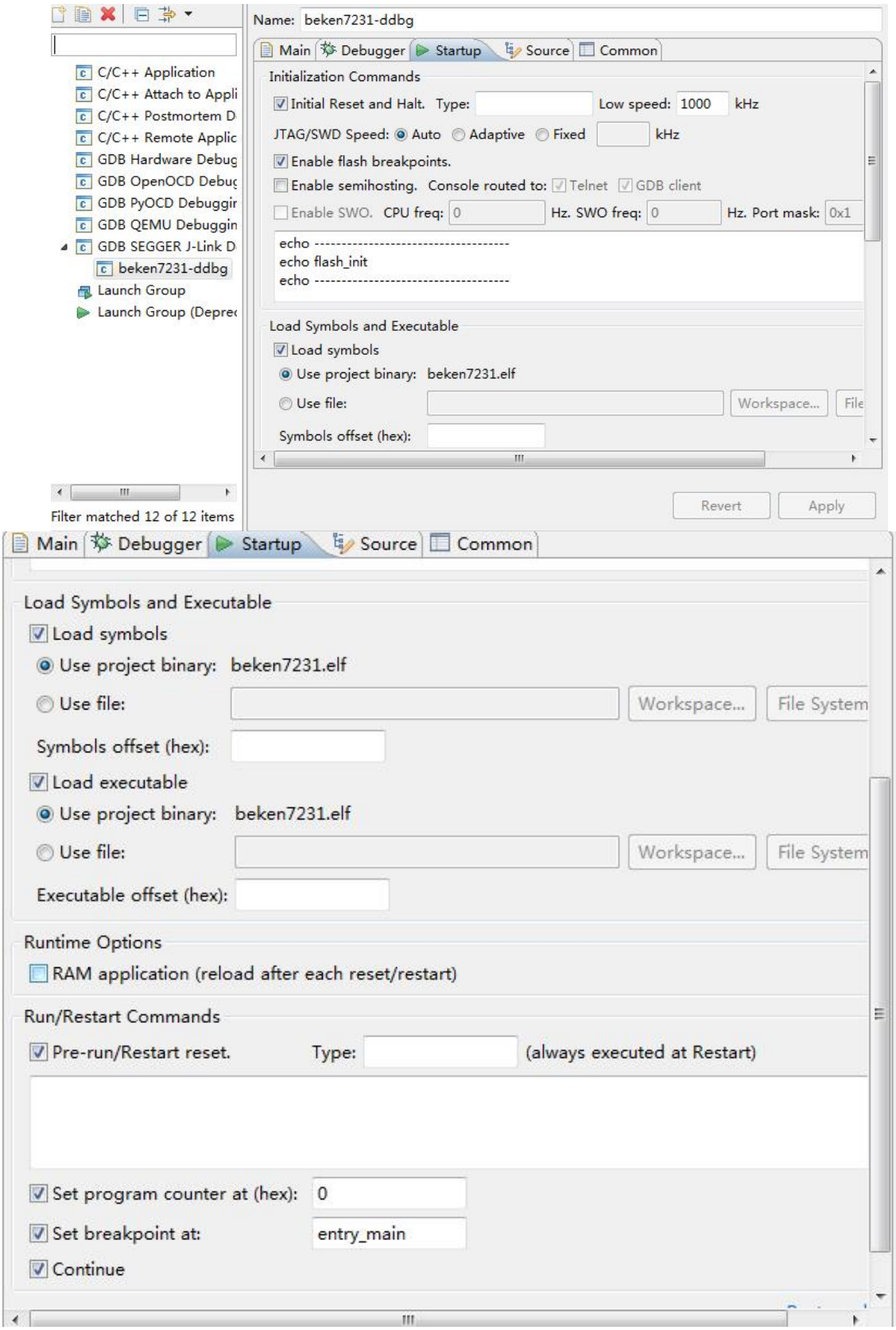
J-Link GDB Server Setup，对于 Executable 选择 JLinkGDBServerCL.exe。JLinkGDBServerCL.exe 是 gdb server 的命令行版本。注意这个目录。

GDB Client Setup，对于 Executable 选择 arm-none-eabi-gdb.exe。arm-none-eabi-gdb.exe 是编译链里面的 gdb 可执行文件。



备注：**不要选择 Verify Download**。

配置之 Startup 属性页



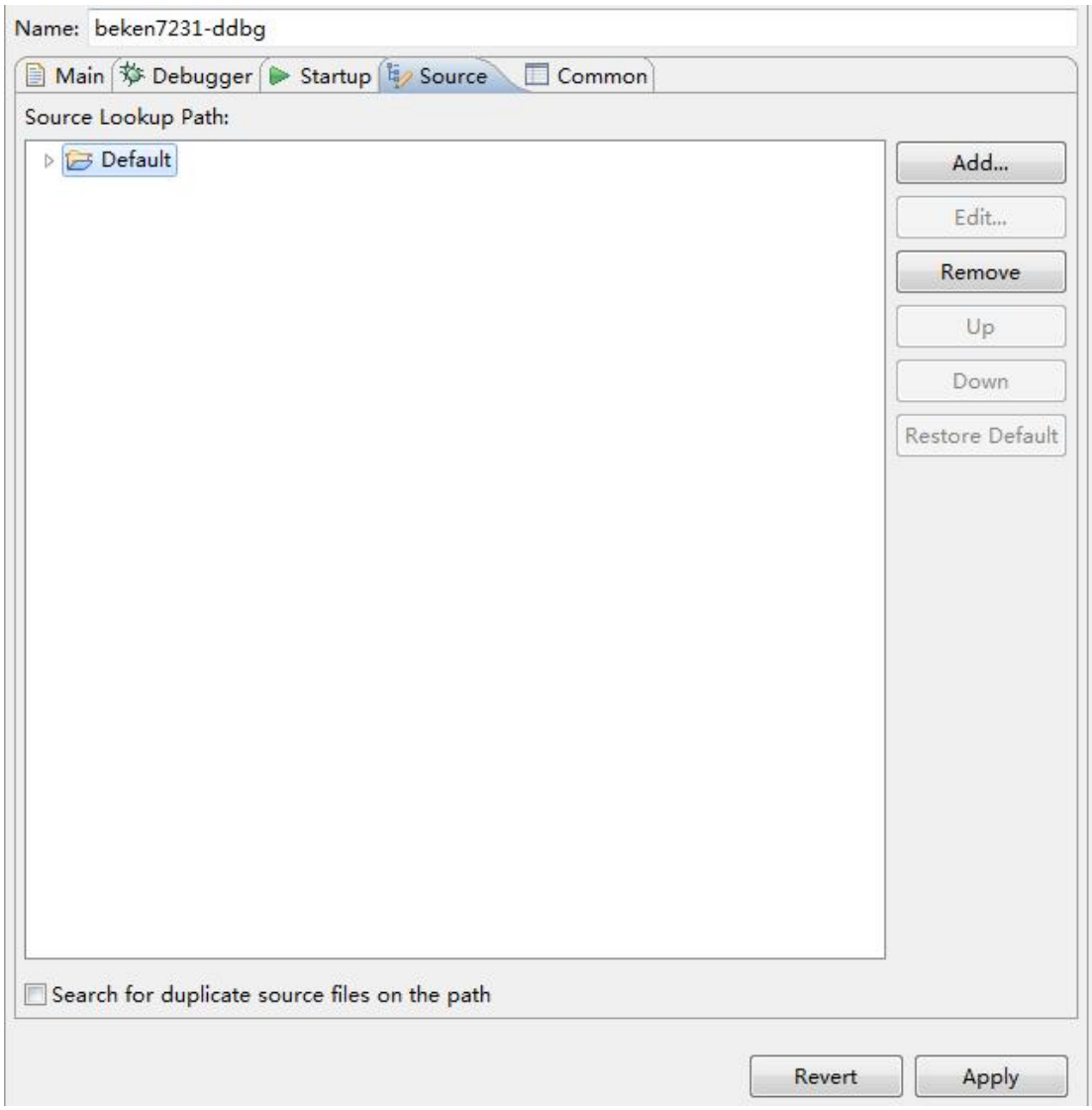
配置之 initialization commands

初始化脚步的作用初始化和擦除 flash。

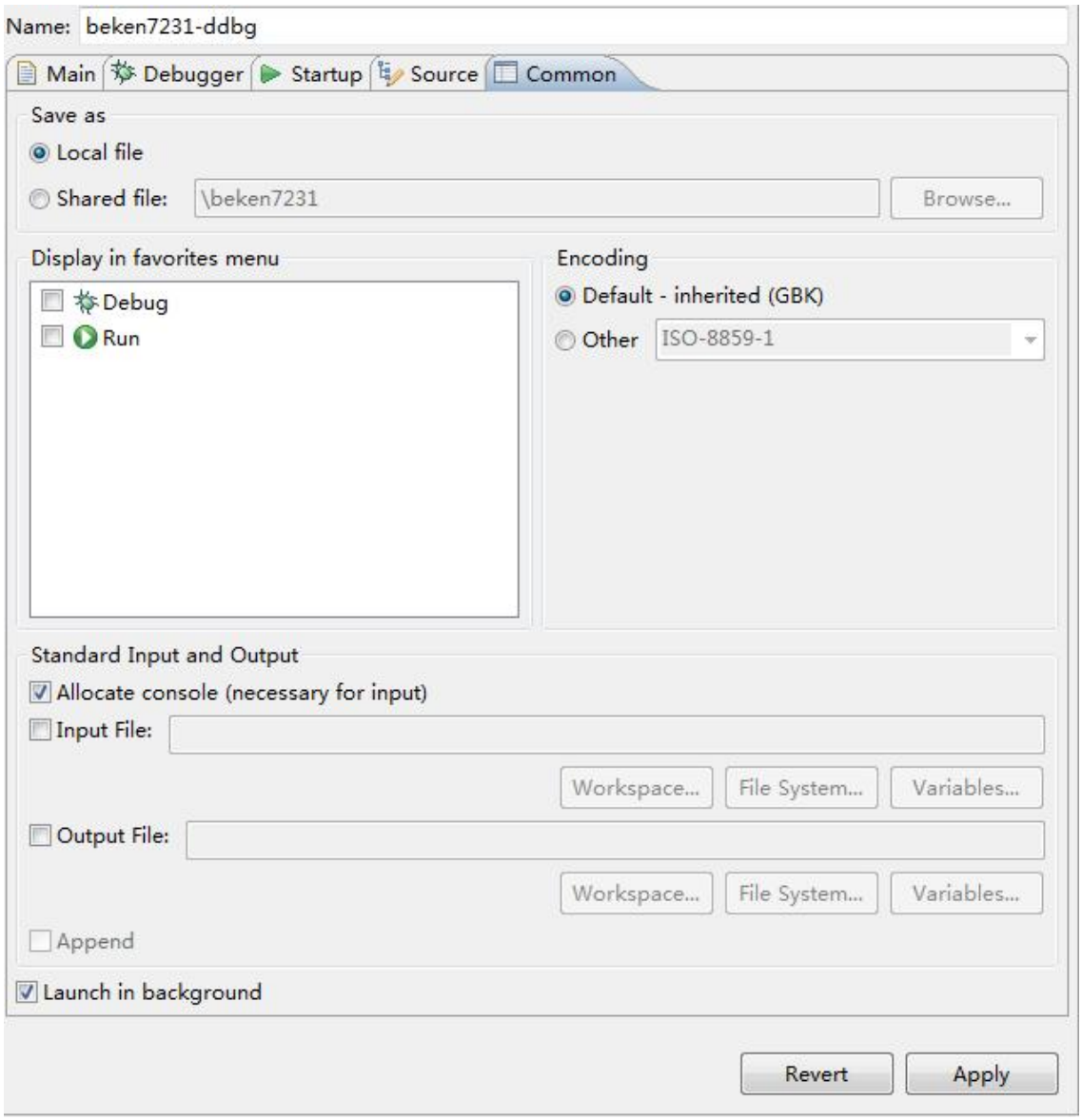
```
echo ------------------------------------
echo initialization command
echo Copyright (C) Beken Corp 2017-2026
echo ------------------------------------

echo setting up the environment for debugging.\n
echo author:zhilei.wang@bekencorp.com.\n
echo date:  2017/07/18                .\n

monitor reset
monitor halt
monitor speed 1000
monitor endian little
monitor device ARM9

echo define flash_init

echo read register of 0x0080301c
monitor memU32 0x0080301c

echo flash data to cpu crc disable
monitor memU32 0x0080301c = 0x0000000f
monitor memU32 0x0080301c

echo flash wr enable
monitor memU32 0x0080301c = 0x0000020f
monitor memU32 0x0080301c

echo flash operation command: WRSR2
monitor memU32 0x00803000
monitor memU32 0x00803000 = 0x67000000
monitor sleep 1000

echo flash operation command: CE
monitor memU32 0x00803000
monitor memU32 0x00803000 = 0x70000000
monitor sleep 10000

echo end

break boot_reset
```

备注：**如果你的 flash 是 2M Bytes 的，请将上述脚本的 sleep 10000 改为 sleep 20000 或者 sleep 30000**。因为擦除的时间需要更久了。

配置之 Source 属性页



配置之 Common 属性页



Micoder + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB + OPENOCD

bk72xx-Micoder-compileAndDebug guide

JRE

JRE 为 **Java Runtime Environment** 的简称，Java Runtime Environment（包括 Java Plug-in）是 Sun 的产品，包括两部分：Java Runtime Environment 和 Java Plug-in。JavaRuntimeEnvironment(JRE)是可以在其上运行、测试和传输应用程序的 Java 平台。***它包括 Java 虚拟机（jvm）、Java 核心类库和支持文件。它不包含开发工具(JDK)--***

编译器、调试器和其它工具。JRE 需要辅助软件--Java Plug-in--以便在浏览器中运行 applet。

Java Plug-in 软件允许 Java Applet 和 JavaBean 组件在使用 Sun 的 Java Runtime Environment(JRE)的浏览器中运行，而不是在使用缺省的 Java 运行环境的浏览器中运行。Java Plug-in 可用于 Netscape Navigator 和 Microsoft Internet Explorer。

J2RE 是 Java2 Runtime Environment，即 Java 运行环境，只是强调其匹配 Java2 平台，有时简称 JRE。***如果你只需要运行 Java 程序或 Applet，下载并安装它即可。如果你要自行开发 Java 软件，请下载 JDK(JRE 和 JDK 的区别)。***在 JDK 中附带有 JRE。

下载地址

MiCoder IDE 运行依赖 Java 运行环境: JRE 1.8，地址：
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

安装步骤

Step by step

注意点

笔者下载的文件：jre-8u121-windows-x64.exe

MiCoder IDE

下载地址

http://ok0noocp1.bkt.clouddn.com/MiCoder_IDE_1_2_1_Win32_x64.exe

http://developer.mico.io/handbooks/9#在 windows 和 macos 下安装

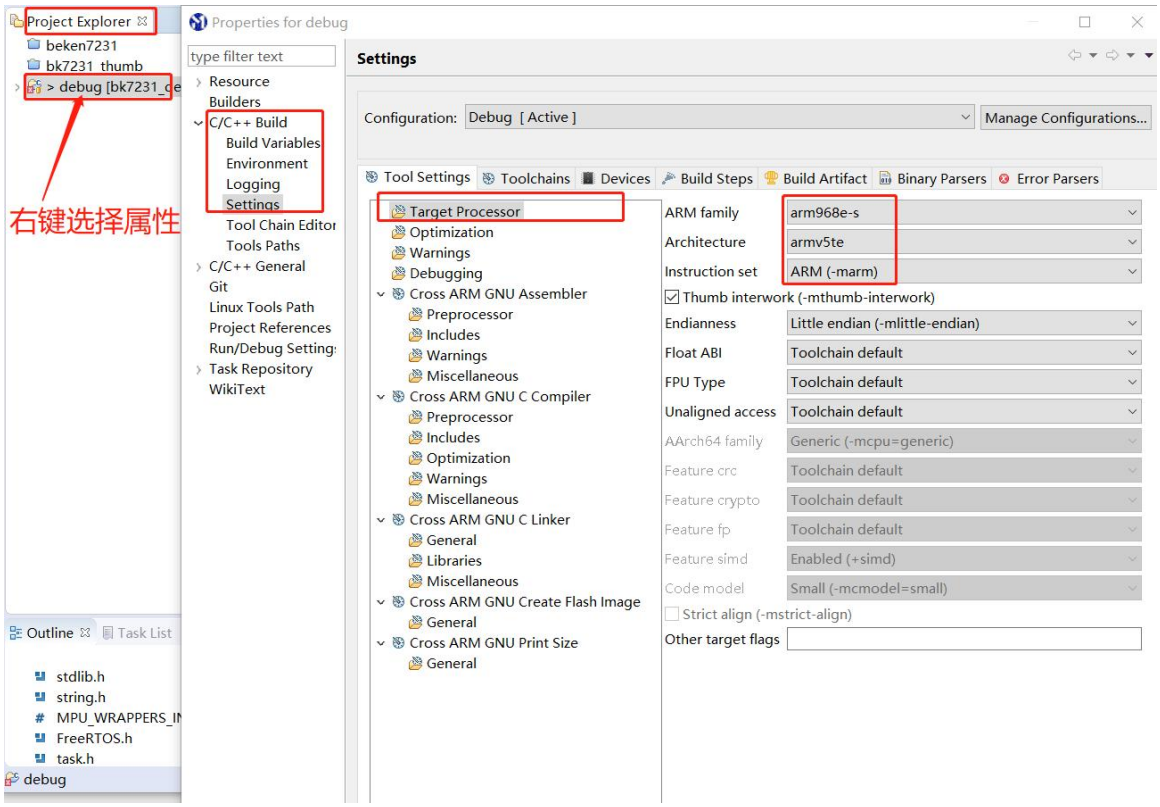
安装步骤

Step by step

检查是否能使用

如果能在 Tool Settings 里看到 Target Processor 等相关信息（见下图），并且能够编译基本工程就表示安装成功。

如果不能使用怎么办？关闭 Micoder IDE，然后请从可以使用的同事电脑上压缩拷贝一份安装目录文件，覆盖在您的安装目录。前提条件是 Windows 系统一样。



注意点

MiCoder_IDE_1_2_1_Win32_x64.exe

如果您安装的 micoder 不能使用，请**从可以使用的同事电脑上压缩拷贝一份**。

J-Link

下载地址

https://www.segger.com/downloads/jlink

安装步骤

Step by step

注意点

目前由 MiCO 支持的所有硬件平台均支持 JLink 调试器 （V8 或 V9）。在 macOS，Linux 平台上将 Jlink 连接上 PC 的 USB 接口后即可使用。在 Windows 下的安装步骤如下：

1. 下载和安装 JLink 驱动程序：Setup_JLink_V600i.zip.
2. 下载 Jlink 的 libusb 驱动：Jlink_Driver_for_MiCO.zip，解压缩后运行 **zadig.exe 或 zadig_xp.exe**
3. 选择菜单： Options->List All Devices，下拉列表选择 J-Link，Driver 选择“libusb-win32（v1.2.6.0）"，点击“Replace Driver"，返回 "successful” 后即可关闭。在设备管理器中会出现“libusb-win32 devices。

注意：由于不同厂家的 JLink 固件版本不同，导致部分 JLink 驱动更换为 libusb-win32（v1.2.6.0）后可能无法正常工作。这种情况下可尝试使用“libusbK (v3.0.7.0)"

笔者使用的就是：libusbK (v3.0.7.0)

Compiler Setting

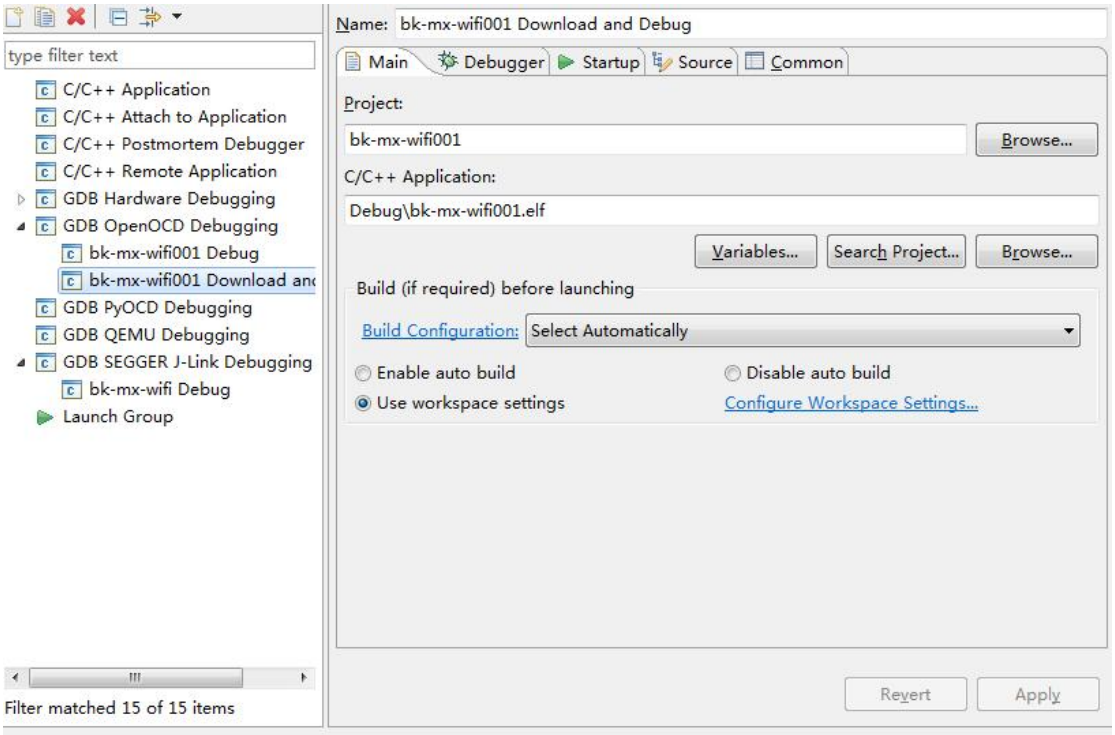
同前面章节

Debugging Setting

创建调试配置

点击 eclipse 菜单 Run—》Debug Configurations，双击 GDB OpenOCD Debugging

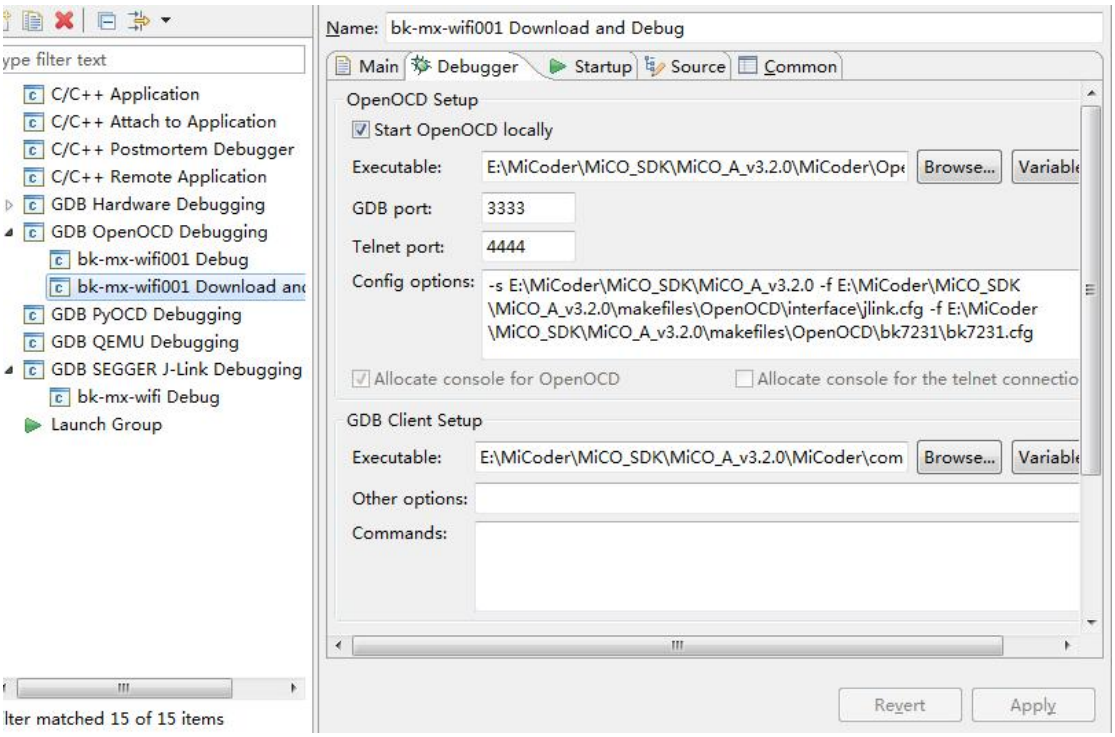
配置之 Main 属性页



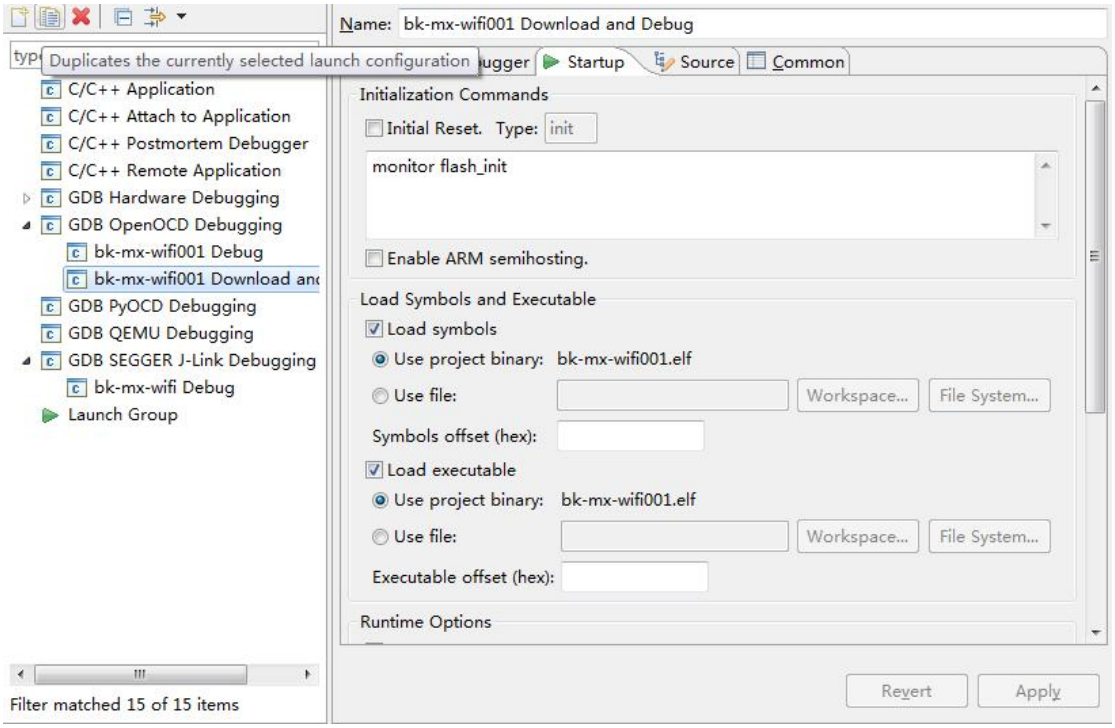
配置之 Debugger 属性页

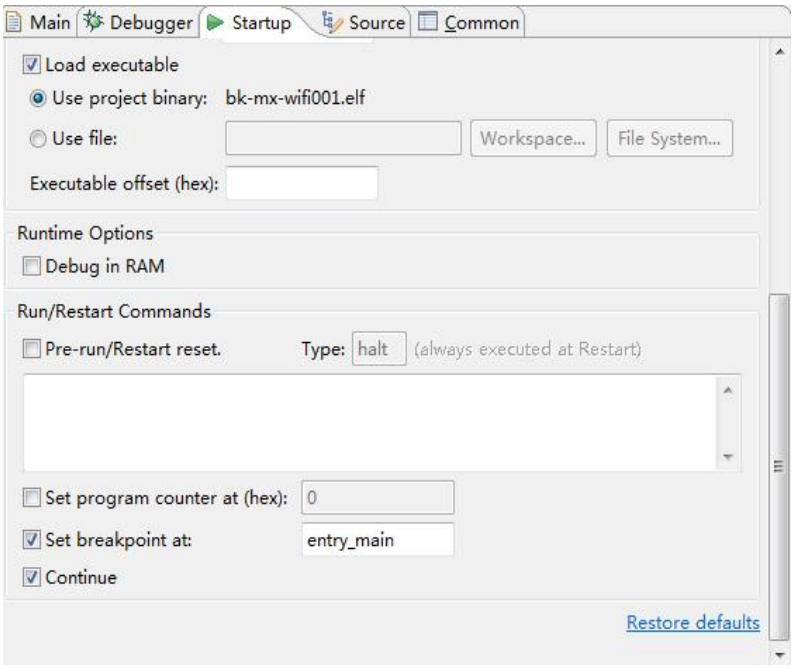
J-Link GDB Server Setup，对于 Executable 选择 openocd_mico.exe (MiCoder\OpenOCD\Win32\openocd_mico.exe)。

GDB Client Setup，对于 Executable 选择 arm-none-eabi-gdb.exe。arm-none-eabi-gdb.exe 是编译链里面的 gdb 可执行文件。



配置之 Startup 属性页



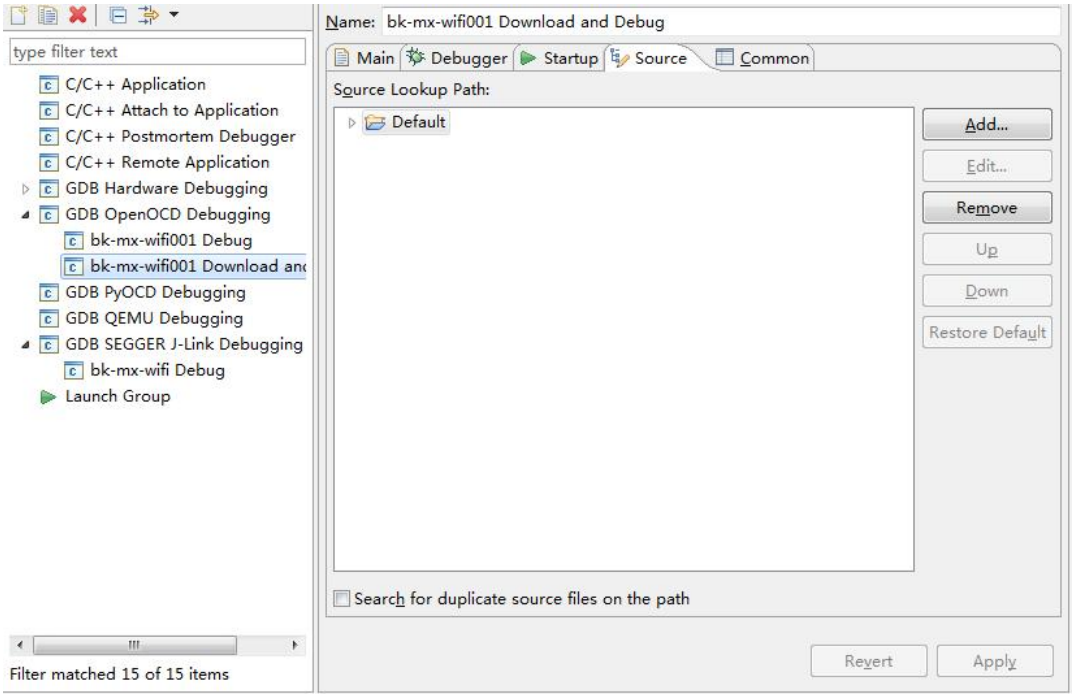


备注：如果只加载符号表，不下载调试，只需要去掉 monitor flash_init 和去掉 Load executable 前面的勾选。

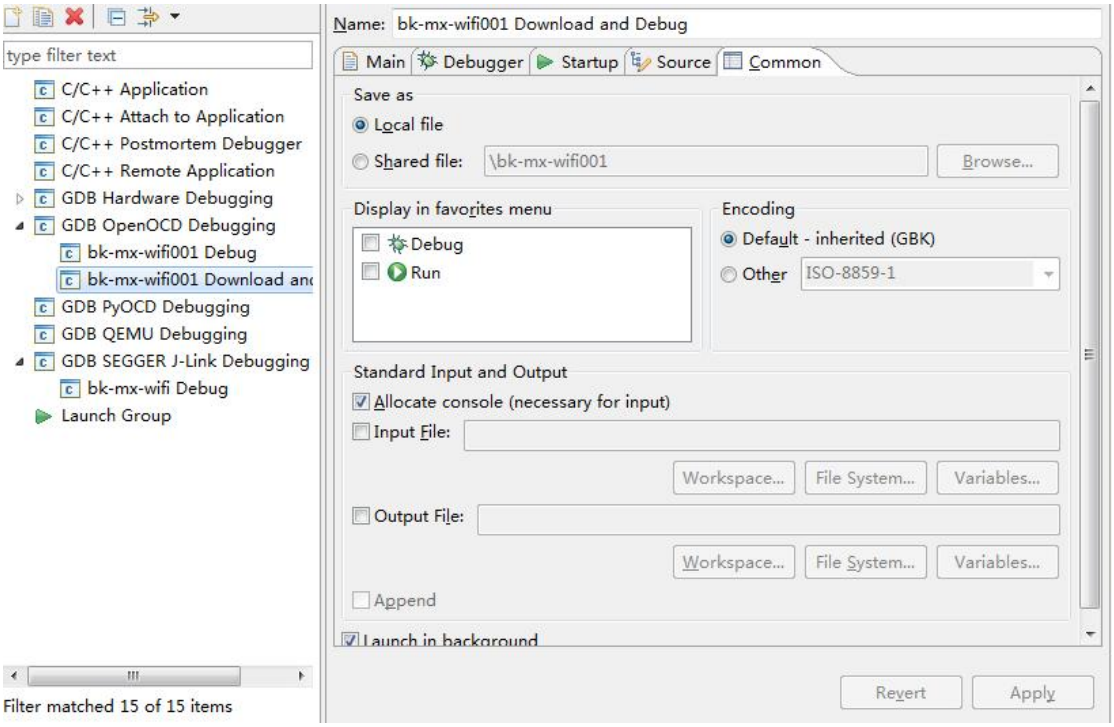
配置之 initialization commands

初始化脚步的作用初始化和擦除 flash。

配置之 Source 属性页



配置之 Common 属性页

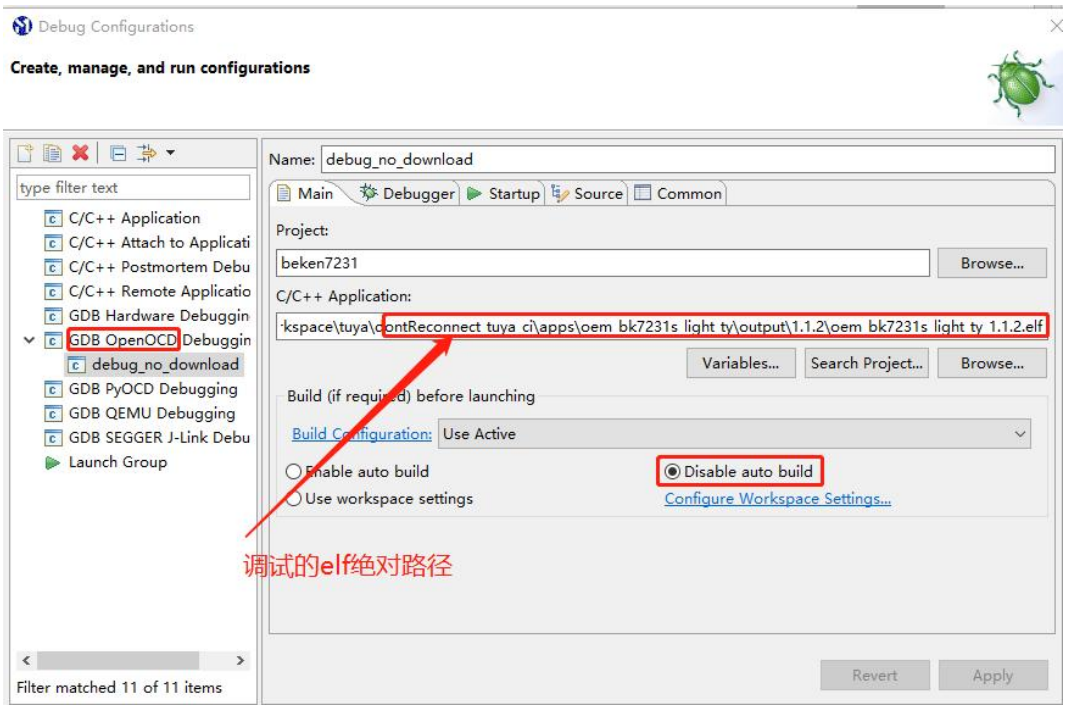


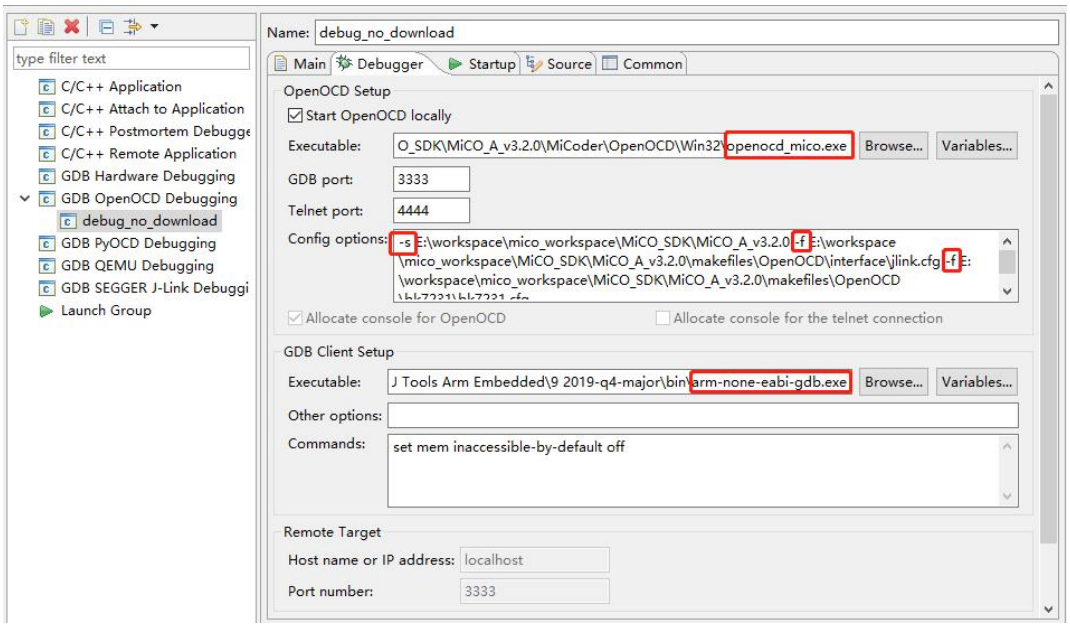
Debug without Downloading

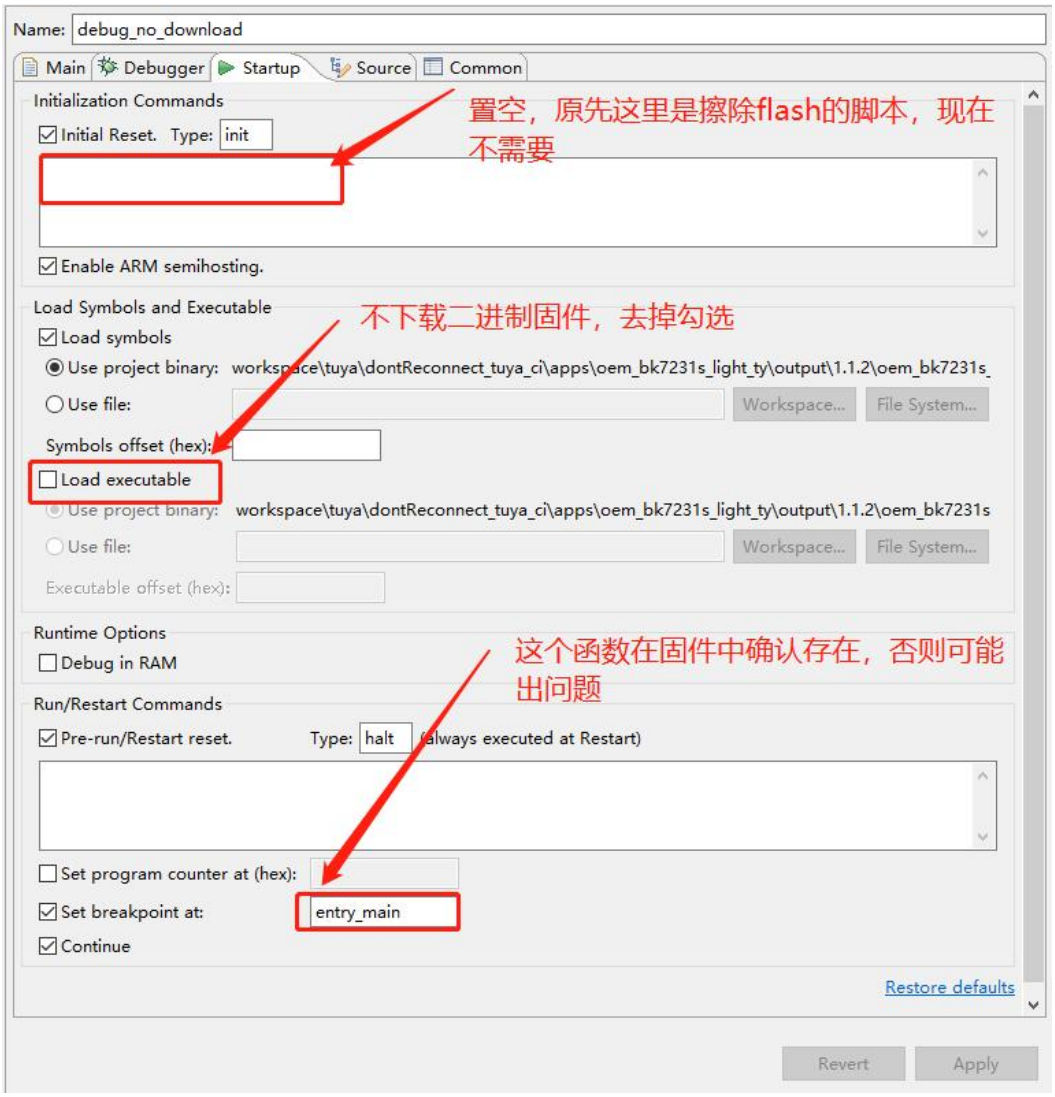
不下载直接调试，关键在于：

1. **Initialization command 属性框中不要加 flash_init;**
2. **去掉 Load exclutable 勾选；**

细节如下：







Alios 使用 Micoder + JTAG 调试的补充说明

对于 alios，需要加如下配置，可以编译生成带 debug 信息的 elf 文件
文件路径\build\aos_toolchain_arm-none-eabi.mk

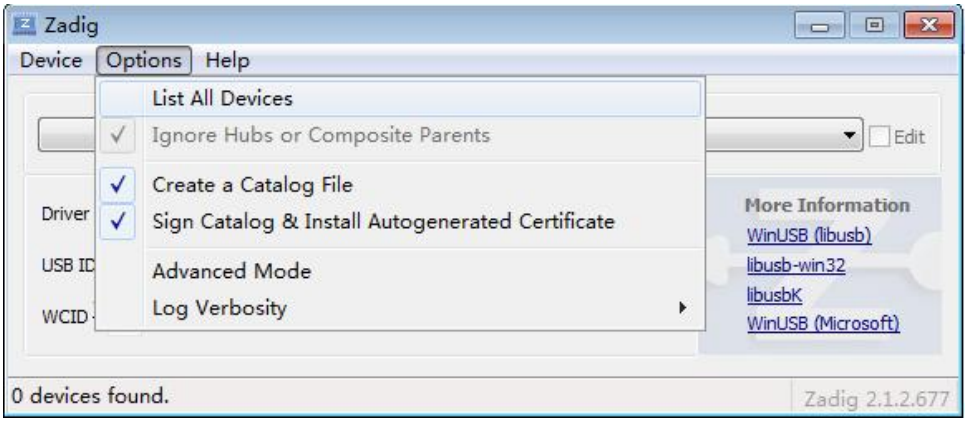
```
ifeq ($(HOST_ARCH),ARM968E-S)
CPU_BASE_FLAGS      := -mcpu=arm968e-s -march=armv5te -marm -mlittle-endian -mthumb-interwork
CPU_CFLAGS          := $(CPU_BASE_FLAGS) -O0 -gdwarf-2
```

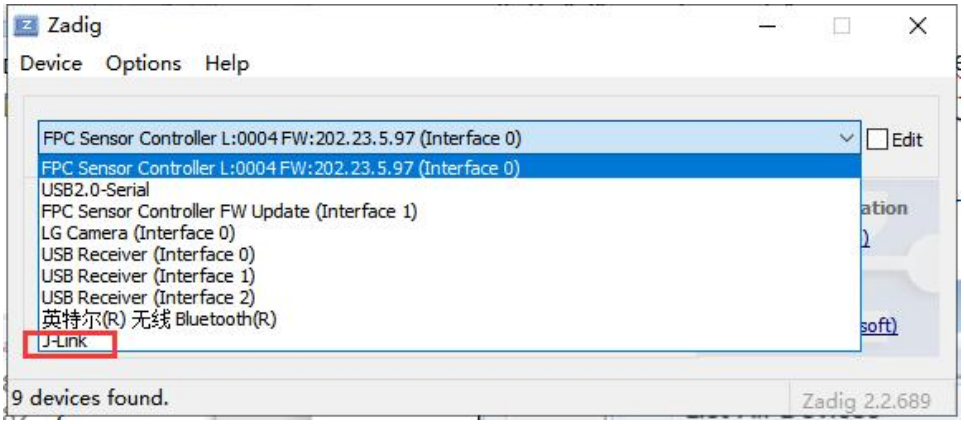
```
CPU_CXXFLAGS          := $(CPU_BASE_FLAGS)
CPU_ASMFLAGS          := $(CPU_BASE_FLAGS)
CPU_LDFLAGS           := $(CPU_BASE_FLAGS)
Endif
```

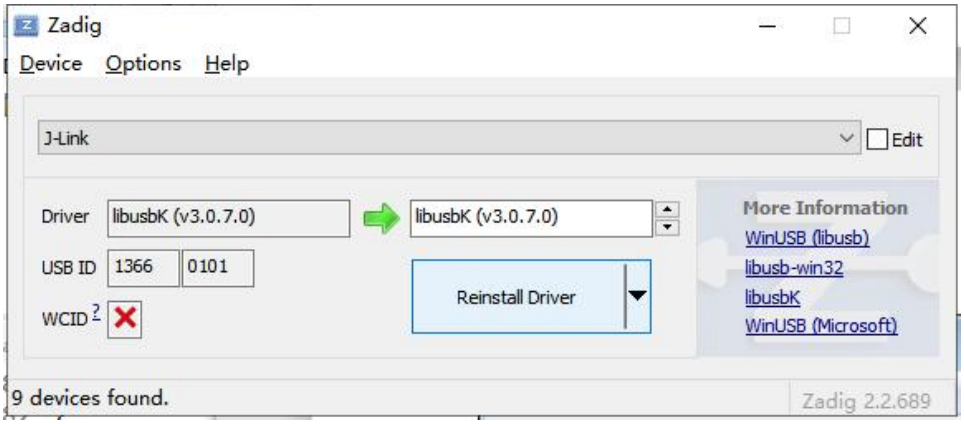
使用 MiCoder Jtag 驱动安装方法

1. 安装 Jave 依赖环境：jre-8u181-windows-x64.exe 到默认路径
2. 安装 Micoder 开发环境：MiCoder_IDE_1_2_1_Win32_x64.exe 到默认路径
3. 安装 JLink 驱动：Setup_JLink_V512 到默认路径
4. 安装 Jlink for Micoder 驱动：Jlink_Driver_for_MiCO 中的 **zadig.exe**

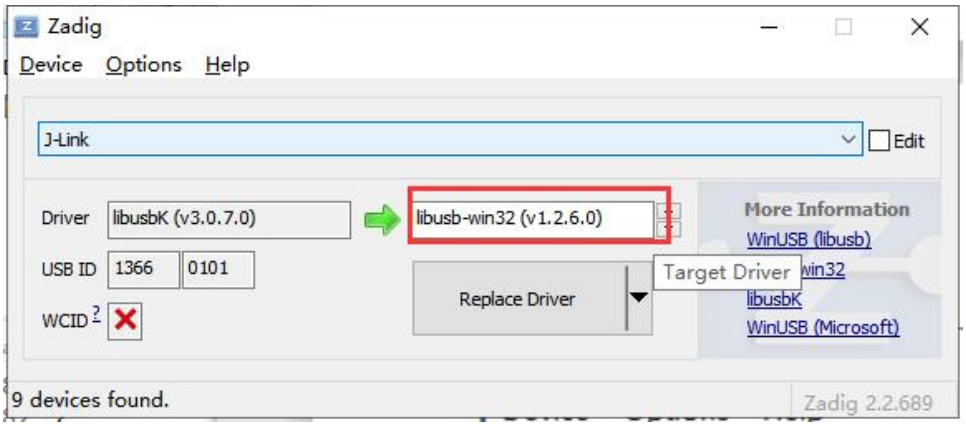
安装步骤







如果不能调试，请尝试

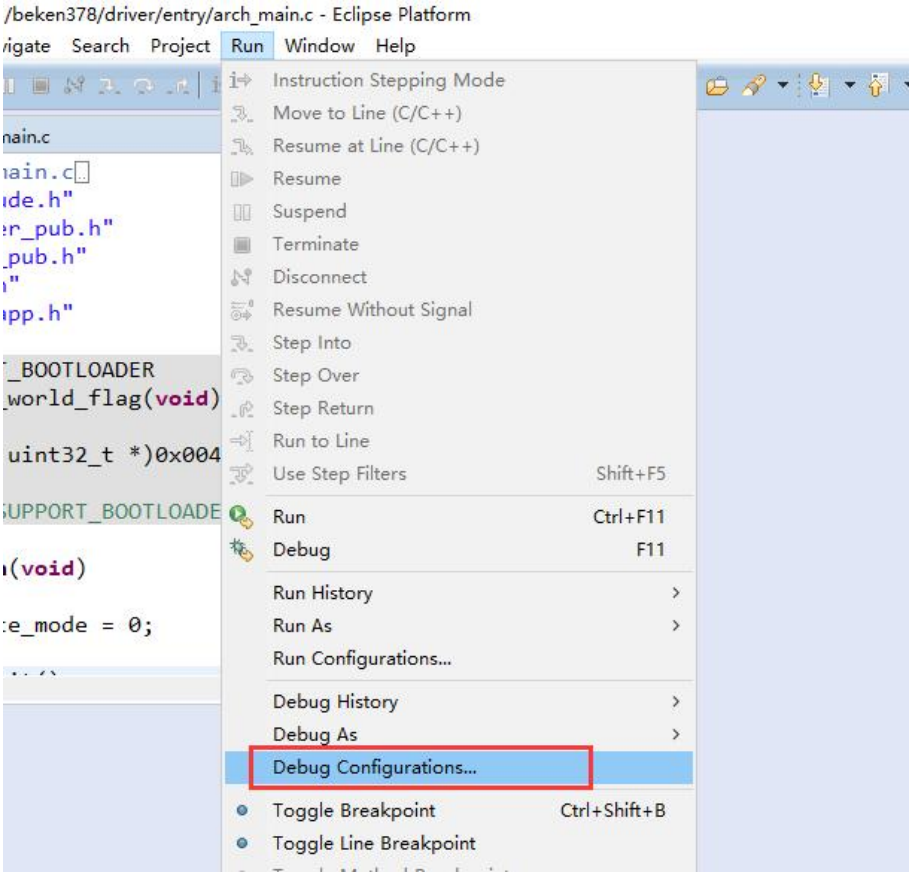


完成后，在设备管理器中看到设备如下图

配置细节

1、将 MiCO_SDK.rar 解压缩放在 C 盘根目录

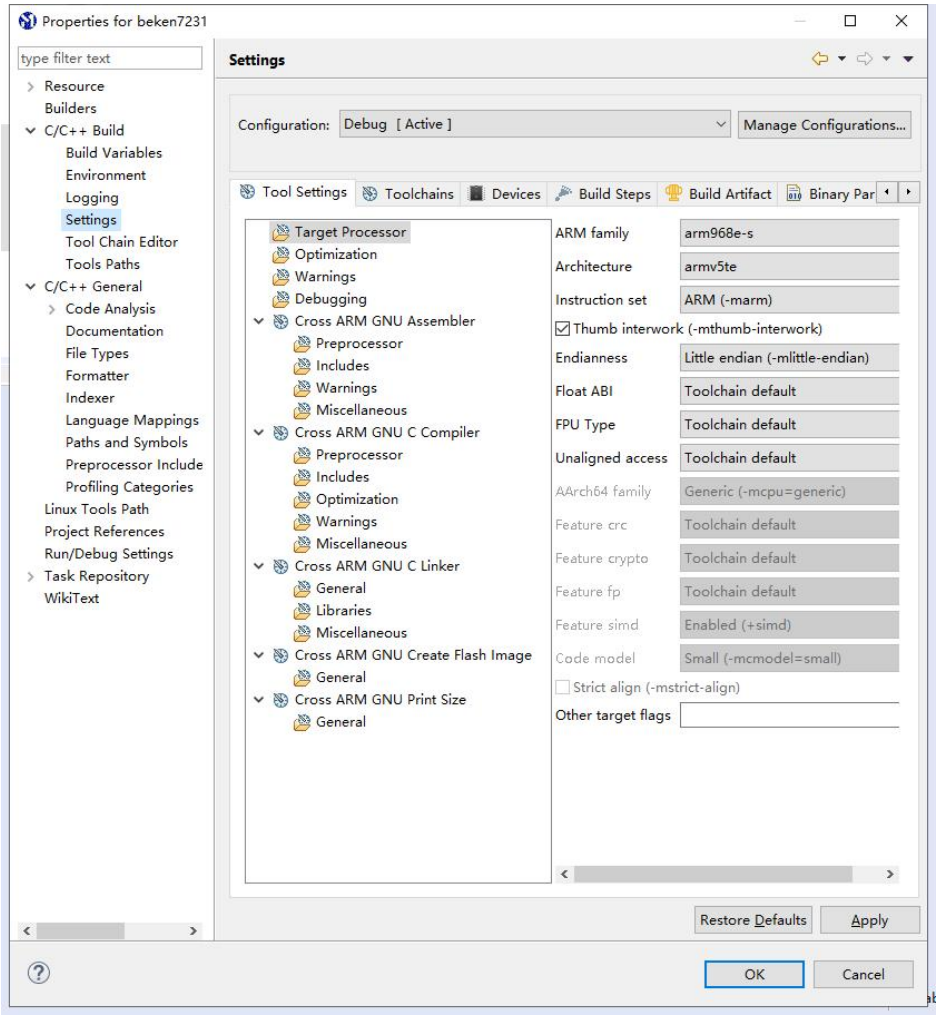
2、打开 MiCoder，开始配置，重点是 Debug Configurations

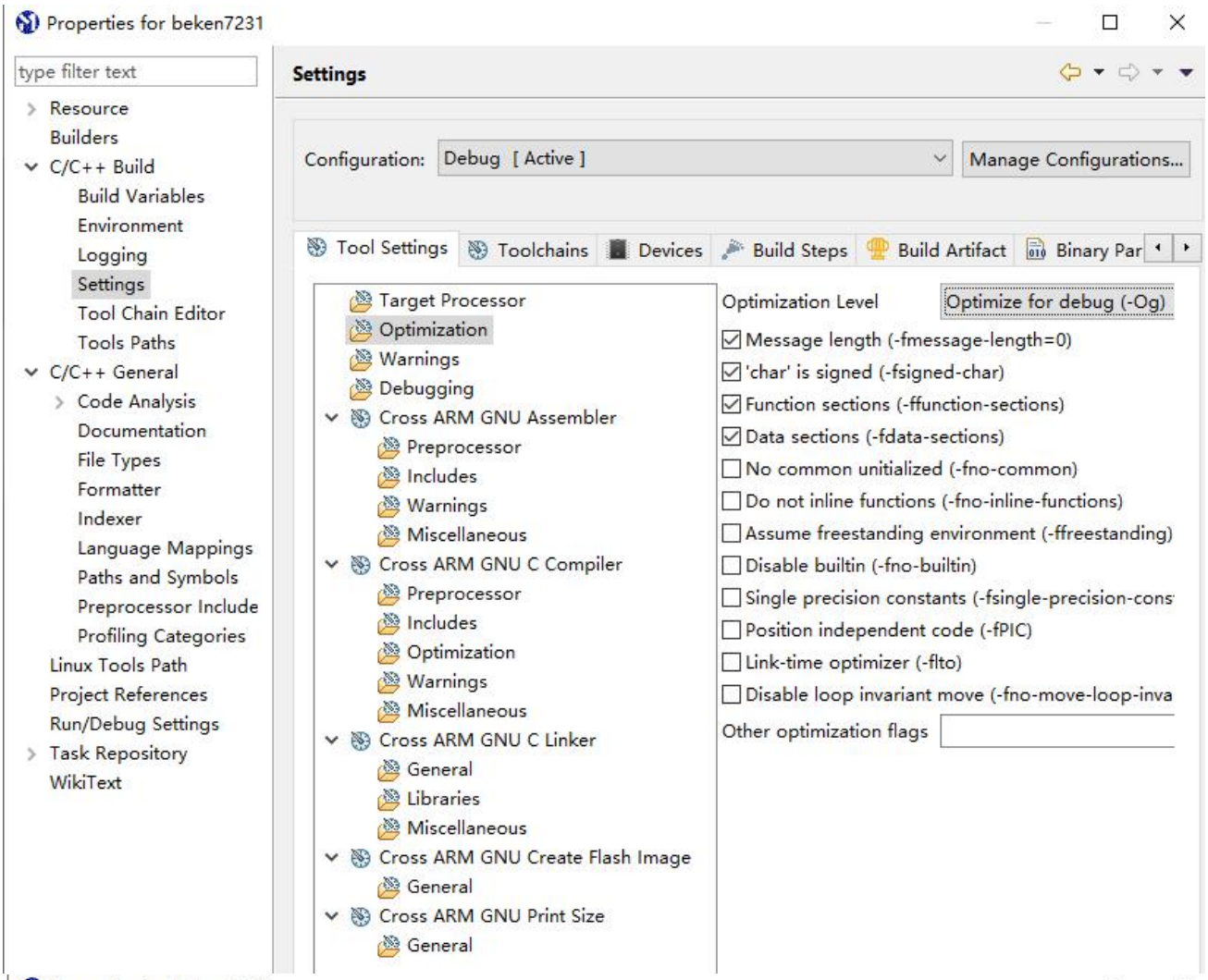


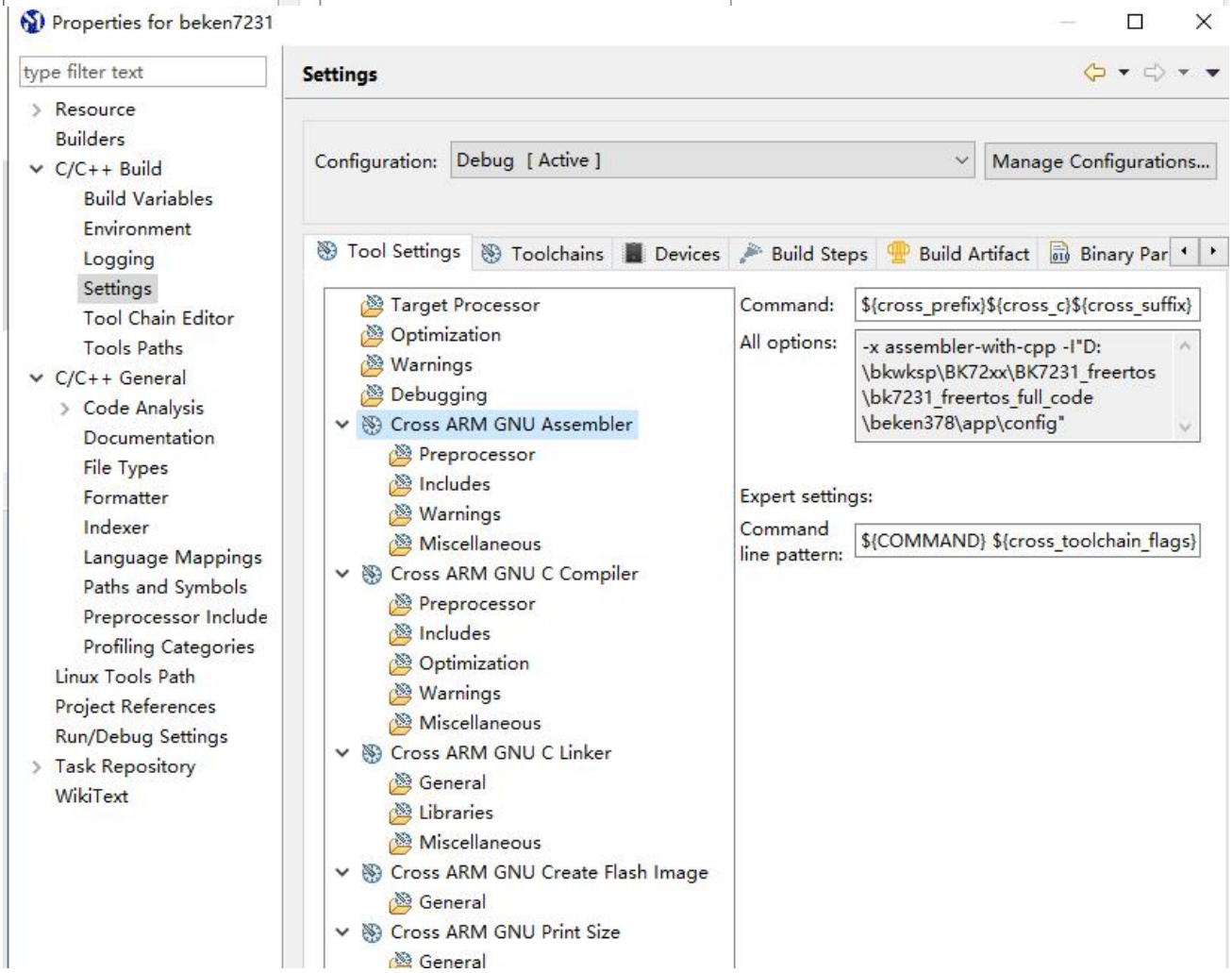
3、建议先 Import 一个之前编译过的工程，这样可以自动导入大部分 Properties 与 Debug 的配置

可用的工程如之前解压缩到 C 盘根目录下的 MiCO_SDK 或 bk7231_freertos_full_code 导入后，大致检查一下相关页面是否已经把配置导入

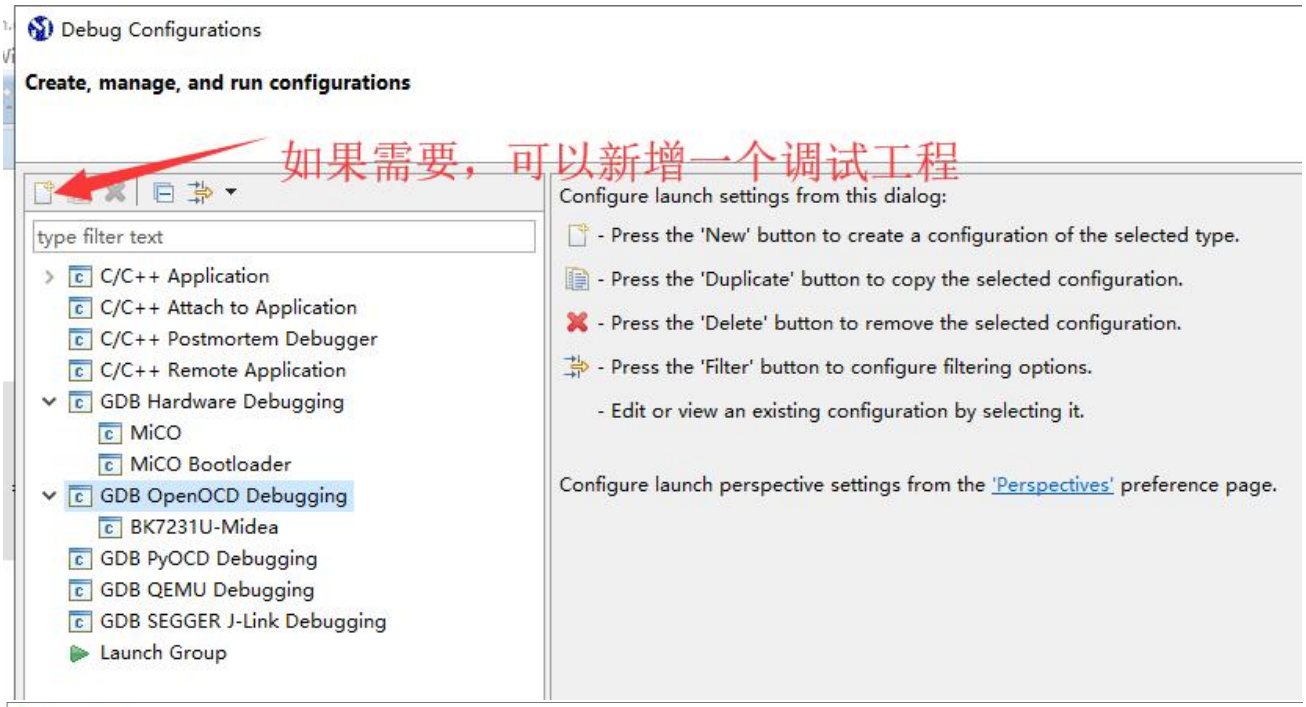
备注：编译配置建议在 bk7231 的工程配置（.cproject/.cproject）基础上更改或者优化。

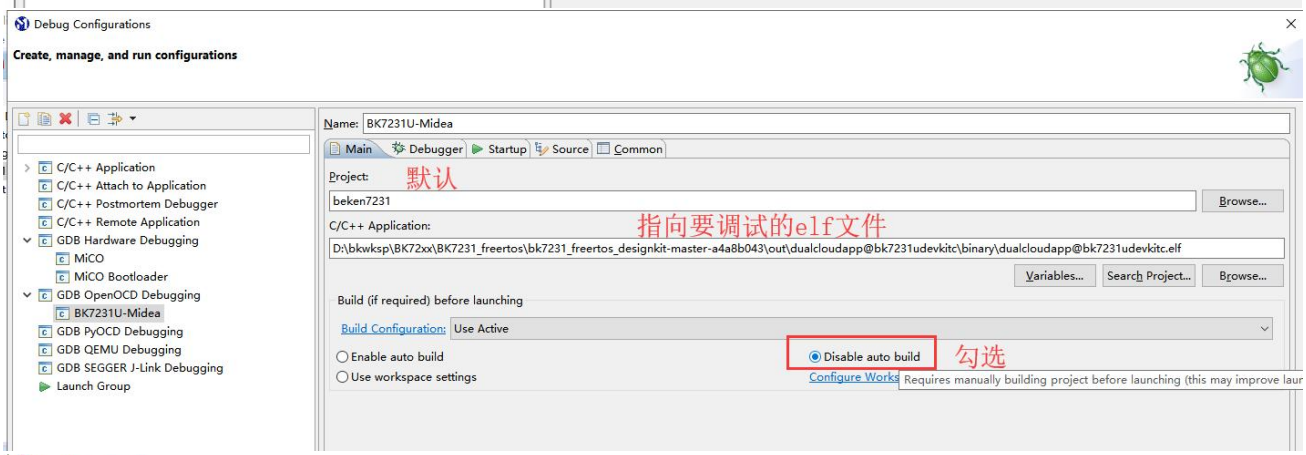


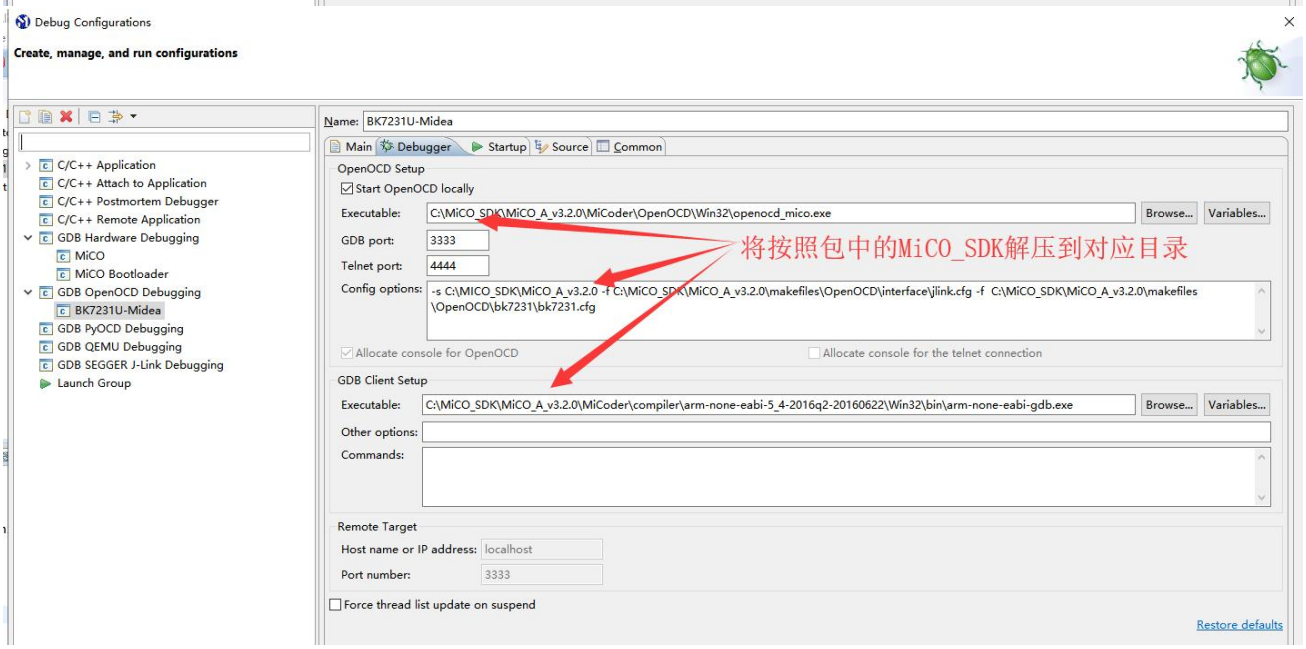




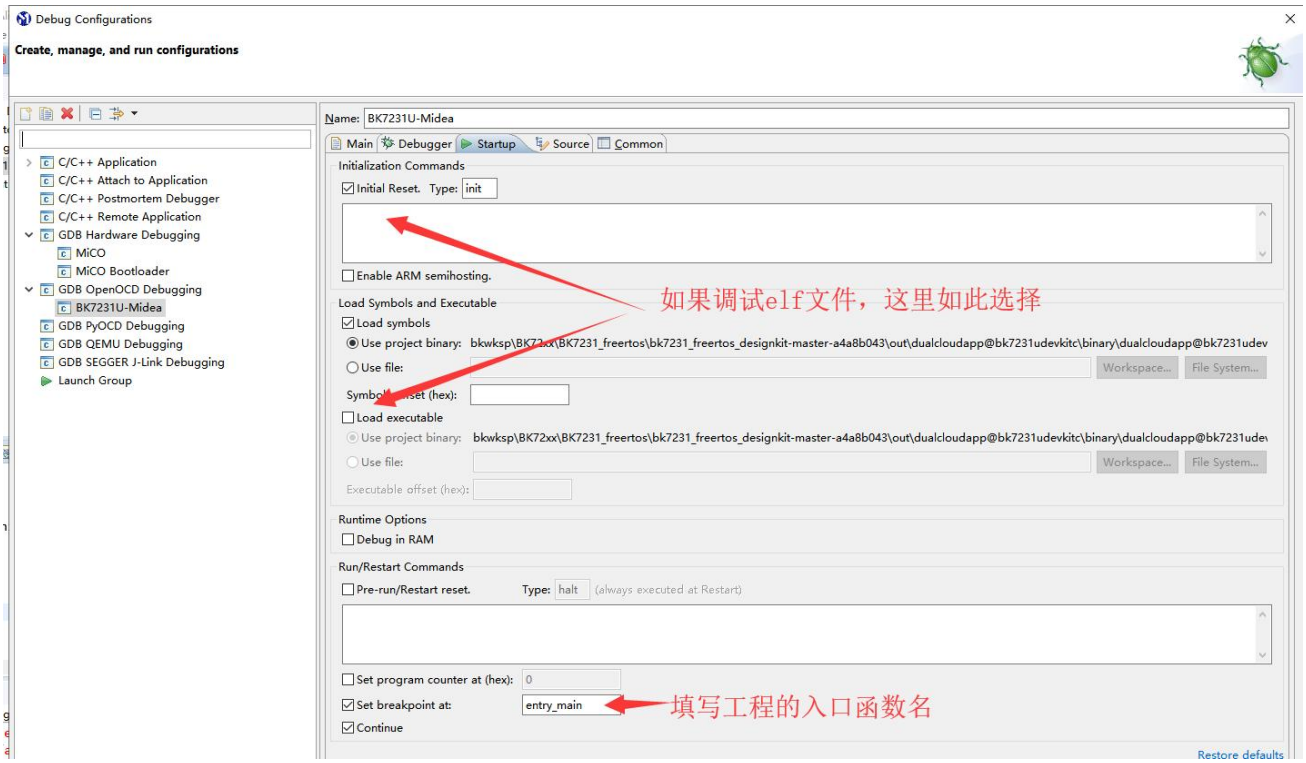
4、再具体检查 Debug Configurtions 的配置



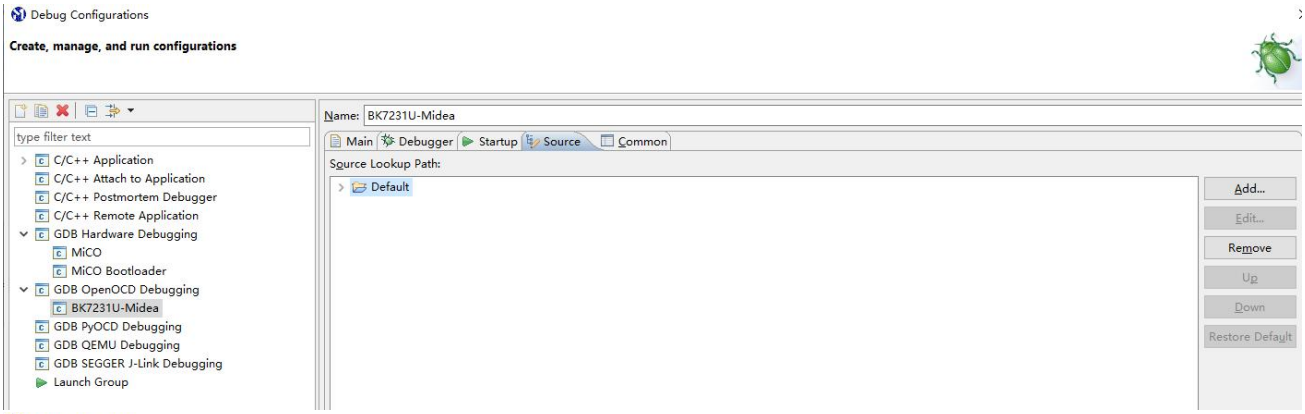


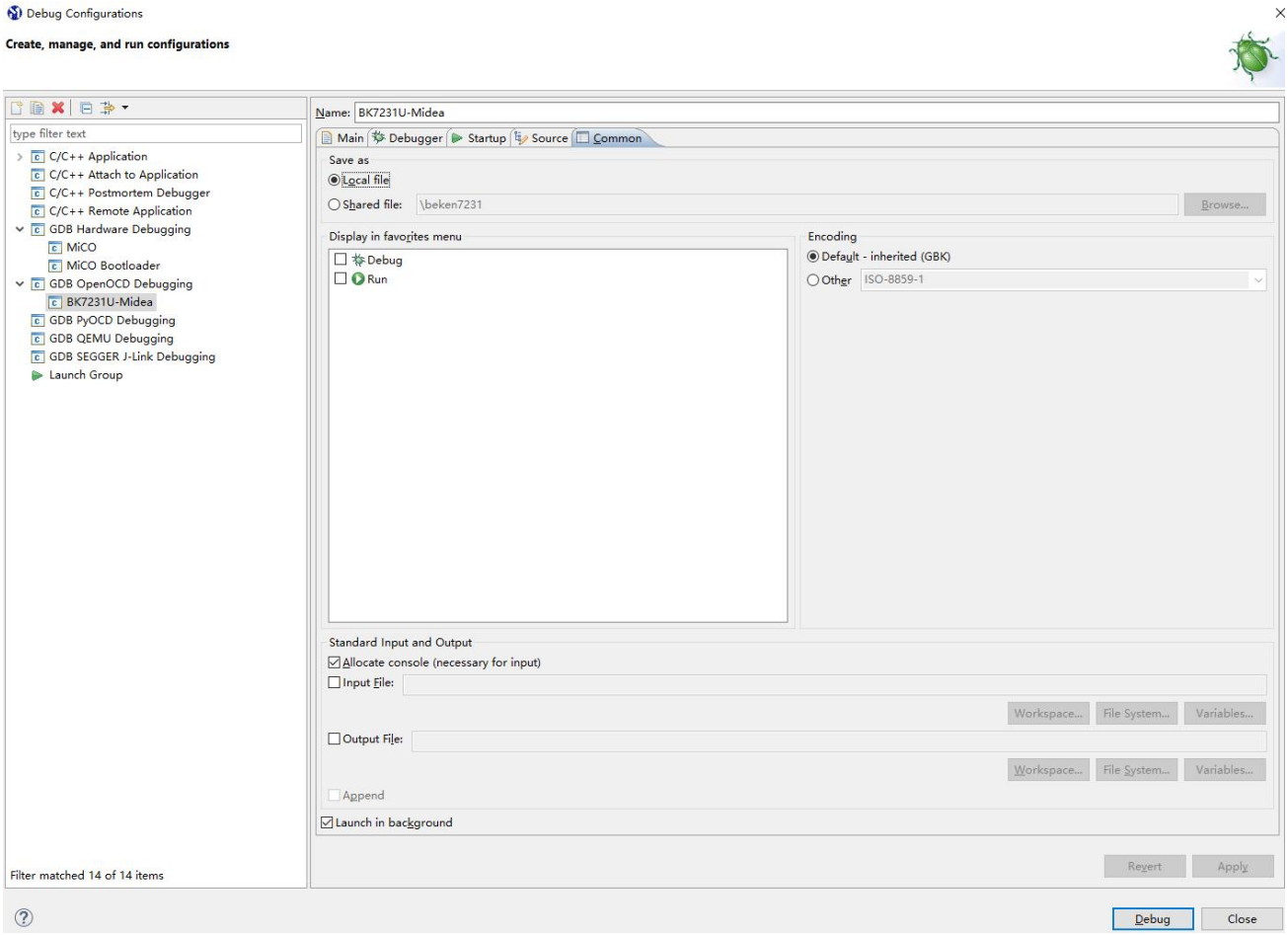


-s C:\MICO_SDK\MiCO_A_v3.2.0 -f C:\MiCO_SDK\MiCO_A_v3.2.0\makefiles\OpenOCD\interface\jlink.cfg -f C:\MiCO_SDK\MiCO_A_v3.2.0\makefiles\OpenOCD\bk7231\bk7231.cfg



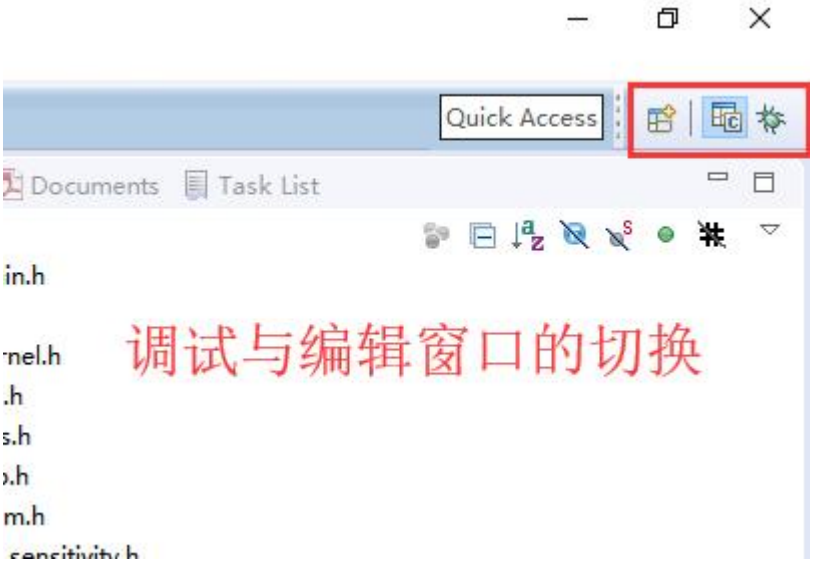
如果调试前需要下载代码，则勾选 Load executable





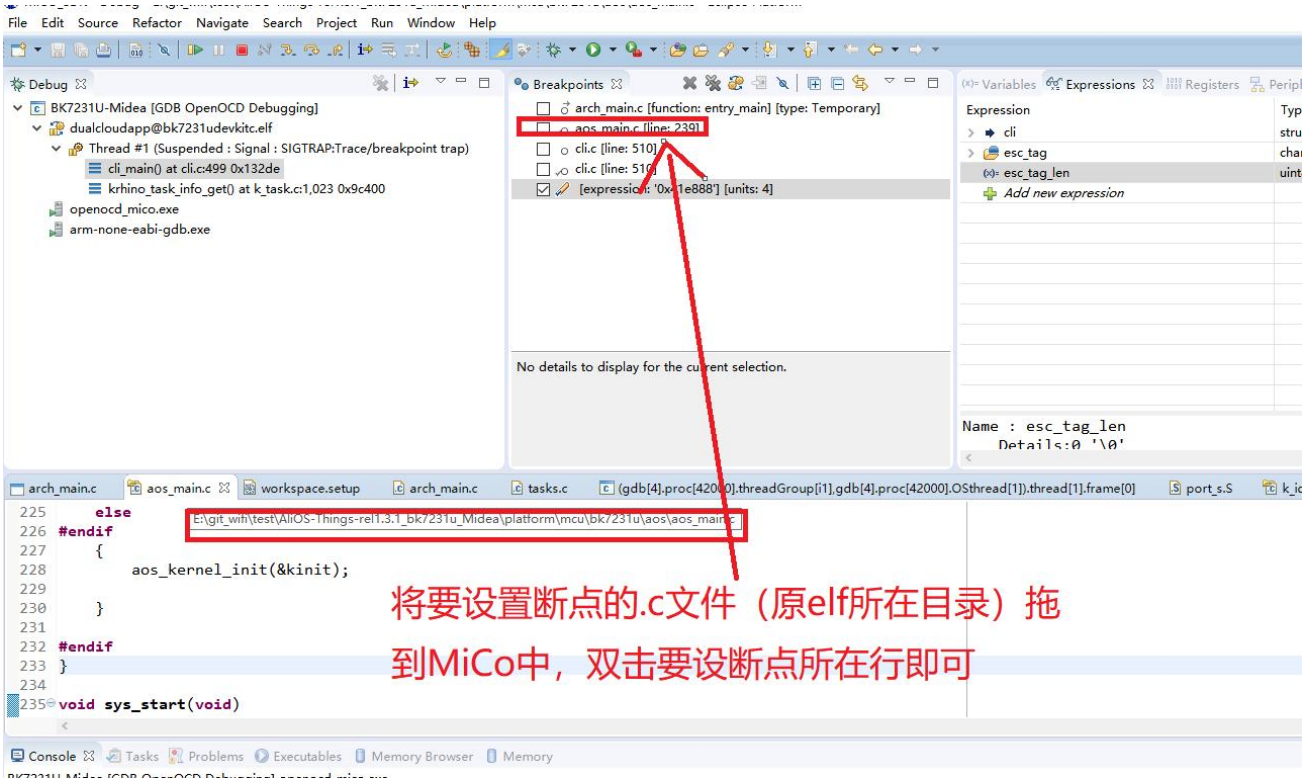
5、所有按照如上配置完成后，每页面点击 Apply，然后 Debug 开始调试

10、调试窗口与编辑窗口（资源管理器）的切换

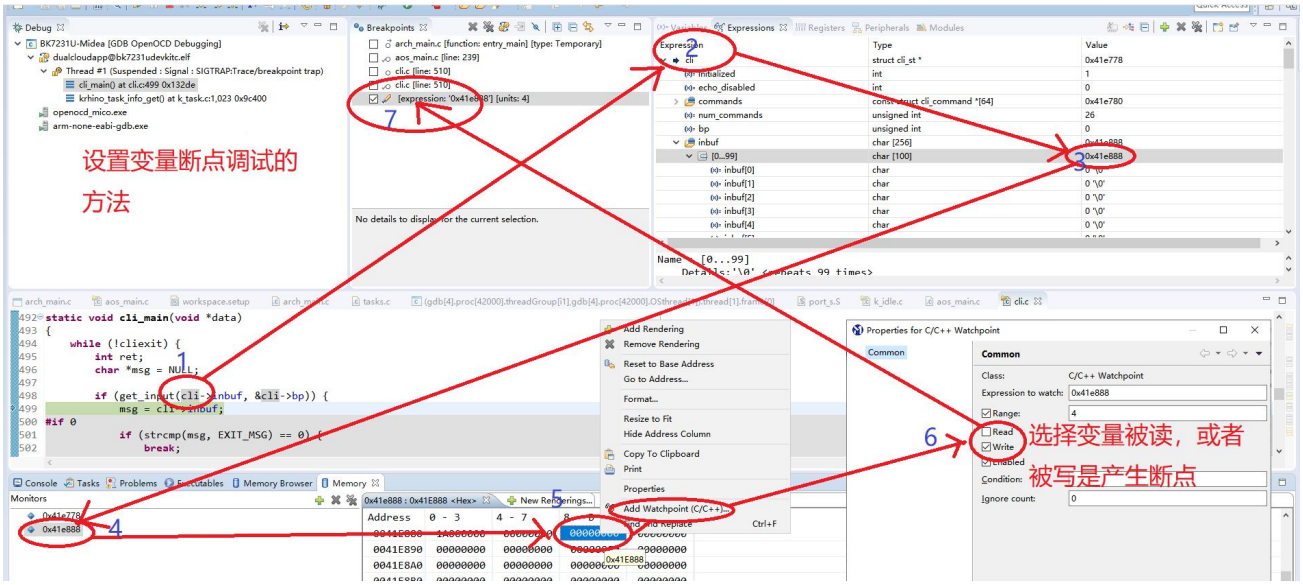


11、设置断点的方法

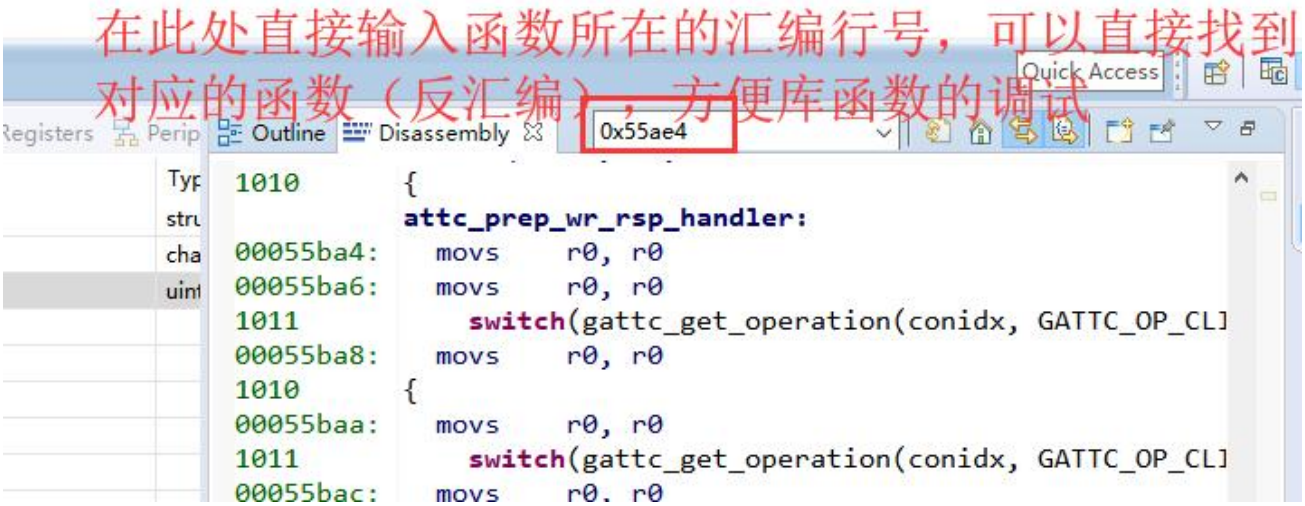
设置程序断点调试的方法：



设置变量断点调试的方法：



库函数的断点调试



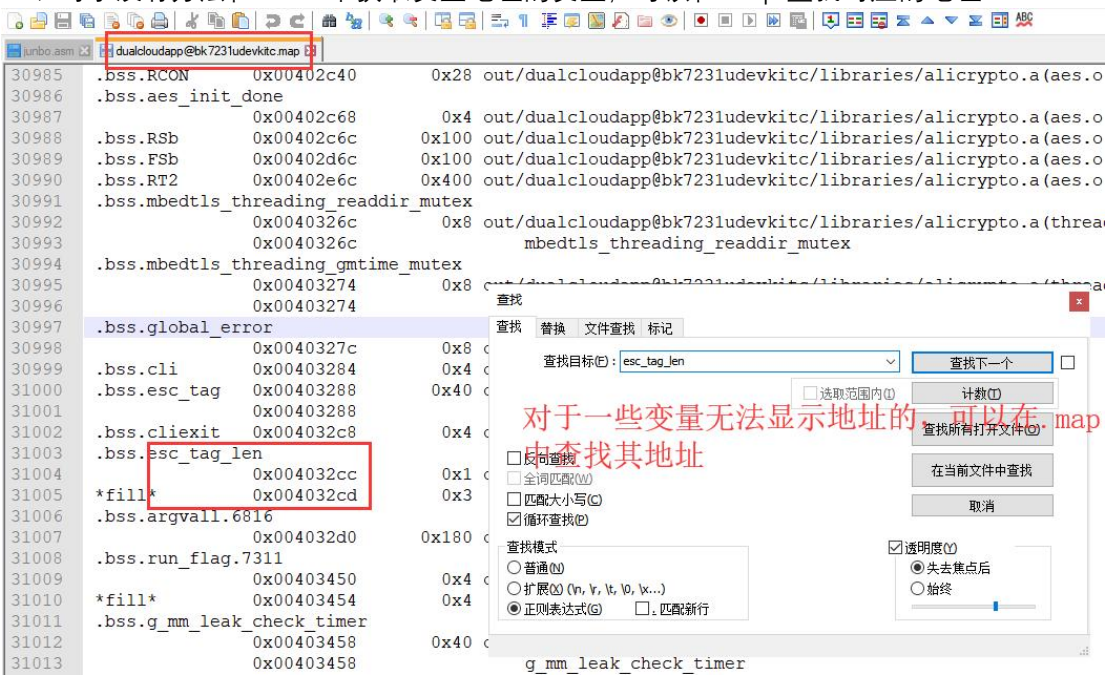
12、由.elf 文件生成.asm 文件的方法：

arm-none-eabi-objdump.exe -d dualcloudapp@bk7231udevkitc.elf > dualcloudapp.asm

在.asm 文件中由汇编文件对应的行号，函数名等信息，用于库文件的追踪，断点调试

```
0000a864 <ali_aes_finish>:
    a864:   b5f0        push    {r4, r5, r6, r7, lr}
    a866:   b091        sub sp, #68 ; 0x44
    a868:   001f        movs    r7, r3
    a86a:   ab16        add r3, sp, #88 ; 0x58
    a86c:   781b        ldrb    r3, [r3, #0]
    a86e:   9004        str r0, [sp, #16]
```

13、对于没有办法在 mico 中获取变量地址的变量，可以在.map 查找对应的地址



注意事项：

1、初次使用，可以先导入 bk7231_freertos_designkit-master-a4a8b043，先用 BK7231 跑起来，排除一些未知问题。

2、第二步，如果要调试其他工程的 elf，可以将其他工程根目录下的全部文件，拷贝到 bk7231_freertos_designkit-master-a4a8b043 目录下，如需覆盖，则覆盖。

3、以下警告不用理会！！！

bk72xx-gdb-command.txt

```
echo ------------------------------------
echo initialization command
echo Copyright (C) Beken Corp 2017-2026
echo ------------------------------------

echo setting up the environment for debugging.\n
echo author:zhilei.wang@bekencorp.com.\n
echo date:    2017/07/18                        .\n

monitor reset
monitor halt
monitor speed 1000
monitor endian little
monitor device ARM9

echo define flash_init

echo read register of 0x0080301c
monitor memU32 0x0080301c

echo flash data to cpu crc disable
monitor memU32 0x0080301c = 0x0000000f
monitor memU32 0x0080301c

echo flash wr enable
monitor memU32 0x0080301c = 0x0000020f
monitor memU32 0x0080301c

echo flash operation command: WRSR2
monitor memU32 0x00803000
monitor memU32 0x00803000 = 0x67000000
monitor sleep 1000

echo flash operation command: CE
monitor memU32 0x00803000
monitor memU32 0x00803000 = 0x70000000
```

```
monitor sleep 10000

echo end

break boot_reset

echo -----------------------------------
echo author:zhilei.wang@bekencorp.com
echo date:    2017/07/18
echo -----------------------------------
```

bk72xx.cfg

```
source [find makefiles/OpenOCD/interface/swj-dp.tcl]

set _CHIPNAME bk7231
set _CPUTAPID 0x15968001
set _TARGETNAME $_CHIPNAME.cpu

adapter_khz 3000

# TRST and SRST both exist, and can be controlled independently
#reset_config srst_only
reset_config none

# Add the ARM9 core debug tap
jtag newtap $_CHIPNAME cpu -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_CPUTAPID

# Create the ".cpu" target
target create $_TARGETNAME arm966e -endian little -chain-position $_TARGETNAME

#$_TARGETNAME configure -work-area-phys 0x400000 -work-area-size 0x8000 -work-area-backup 0

arm7_9 dbgrq enable
arm7_9 dcc_downloads enable

gdb_breakpoint_override enable

# Reset target when gdb attaches
$_TARGETNAME configure -event gdb-attach {
	soft_reset_halt
}

# Shutdown OpenOCD daemon when gdb detaches
$_TARGETNAME configure -event gdb-detach {
```

```
	resume
	shutdown
}

$_TARGETNAME configure -event gdb-flash-write-start {
	puts ">>>>>>>> gdb-flash-write-start"
}

proc flash_init { } {
	# Flash data to CPU CRC disable
	puts ">>>>>>>> Flash data to CPU CRC disable"
	mem2array tmp_arr 32 0x0080301c 1
	set reg_val [expr ($tmp_arr(0))&(~(1<<26))]
	mww 0x0080301c $reg_val

	# Flash status register data to be written
	puts ">>>>>>>> Flash status register data to be written"
	mem2array tmp_arr 32 0x0080301c 1
	set reg_val [expr ($tmp_arr(0))&(~(0xffff<<10))]
	mww 0x0080301c $reg_val

	# CPU data writting enable
	puts ">>>>>>>> CPU data writting enable"
	mem2array tmp_arr 32 0x0080301c 1
	set reg_val [expr ($tmp_arr(0))|(1<<9)]
	mww 0x0080301c $reg_val

	# Flash operation command : WRSR2
	puts ">>>>>>>> Flash operation command : WRSR2"
	mem2array tmp_arr 32 0x00803000 1
	set reg_val $tmp_arr(0)
	# clr op_type_sw
	set reg_val [expr $reg_val&(~(0x1f<<24))]
	# op_sw, wp_value
	set reg_val [expr $reg_val|((0x07<<24)|(1<<29)|(1<<30))]
	mww 0x00803000 $reg_val
	sleep 1000
	set ready 1
	puts ">>>>>>>> Waitting for operation completed..."
	while { $ready } {
		mem2array tmp_arr 32 0x00803000 1
		set reg_val $tmp_arr(0)
		set ready [expr $reg_val&(0x01<<31)]
	}
	puts ">>>>>>>> Operation completed!"

	# Flash operation command : CE
```

```
        puts ">>>>>>>> Flash operation command : CE"
        mem2array tmp_arr 32 0x00803000 1
        set reg_val $tmp_arr(0)
        # clr op_type_sw
        set reg_val [expr $reg_val&(~(0x1f<<24))]
        # op_sw, wp_value
        set reg_val [expr $reg_val|((0x10<<24)|(1<<29)|(1<<30))]
        mww 0x00803000 $reg_val
        sleep 1000
        set ready 1
        puts ">>>>>>>> Waitting for operation completed..."
        while { $ready } {
            mem2array tmp_arr 32 0x00803000 1
            set reg_val $tmp_arr(0)
            set ready [expr $reg_val&(0x01<<31)]
        }
        puts ">>>>>>>> Operation completed!"
}
```

jlink.cfg

```
#
#   UNPUBLISHED PROPRIETARY SOURCE CODE
#   Copyright (c) 2016 MXCHIP Inc.
#
#   The contents of this file may not be disclosed to third parties, copied or
#   duplicated in any form, in whole or in part, without the prior written
#   permission of MXCHIP Corporation.
#
# Segger J-Link
#
# http://www.segger.com/jlink.html
#

interface jlink
```

comments

bk7231-gdb-command.txt

bk7231.cfg jlink.cfg

references

http://gnuarmeclipse.sourceforge.net/

https://eclipse.org

UM08001_JLink.pdf

How to install the GNU **MCU Eclipse plug-ins?**

Contents

- 1_Quicklinks
- 2_Prerequisites
  - 2.1_Win XP - a word of caution
  - 2.2_Win XP 64
  - 2.3_Java
  - 2.4_Java Cryptography Extension (JCE)
  - 2.5_Eclipse & CDT
  - 2.6_CDT
  - 2.7_Compatibility issues
- 3_Plug-ins install/update
  - 3.1_The Eclipse Marketplace way
  - 3.2_The Eclipse update site way
  - 3.3_The local archive way
- 4_Off-line install
- 5_Check/set the global tools paths
- 6_Troubleshooting
  - 6.1_Java security problems
  - 6.2_SourceForge problems
  - 6.3_Incomplete Eclipse
  - 6.4_Ubuntu GTK issue
  - 6.5_macOS com.apple.quarantine
- 7_Plug-ins versions
- 8_Set workspace preferences
- 9_Download CMSIS Packs
- 10_Toolchain and debugging support
- 11_Test project
- 12_Support

Quicklinks

If you know what this is all about, you can do the following:

- use the **GNU MCU Eclipse IDE for C/C++ Developers** packages, which packs Eclipse with the plug-ins; for this go to GitHub Releases and download the binary for your platform;
- go to the *Eclipse* menu → **Help** → **Eclipse Marketplace…**, find **GNU MCU Eclipse** and install

If, for any reason, this does not work and you have to do it manually, you only need the update site details:

- name: **GNU MCU Eclipse Plug-ins**
- URL: `http://gnu-mcu-eclipse.netlify.com/v4-neon-updates/`

Note: as the name implies, this URL is only valid for Eclipse **Neon** or later; the old URL `http://gnuarmeclipse.sourceforge.net/updates/` is now deprecated, and its content will point to v3.x, that can be used to install the older versions of the plug-ins on older Eclipses.

It is recommended that you install the plug-ins after installing the ARM toolchain and/or the RISC-V toolchain, the build tools (on Windows), and the debugging binaries.

Important notes:

- *if you use the Oracle JDK, starting with mid January 2017, attempts to install from SourceForge fail with handshake_error;*
- *if you are behind a firewall, older Java virtual machines fail to connect to SourceForge.*

Prerequisites

Since these are Eclipse plug-ins, they obviously require a functional Eclipse CDT (Neon or later), which requires Java.

Win XP - a word of caution

For the old-timers, a word of caution about Windows XP: recent Java distributions (from 7 up) are no longer certified for XP, so, even if they might seem to work, you are on thin ice. If possible, avoid using XP at all; otherwise use only the 32-bit version, with 32-bit Java and Eclipse.

Win XP 64

The warning is even more important for Win XP 64, which, in certain configurations, has difficulties to start 32-bit applications (like the JLink GDB Server or OpenOCD). Although generally Win XP 64 is not recommended at all, if you really need to use it, preferably experiment with a 32-bit Java and a 32-bit Eclipse.

However please note that no support requests referring to Win XP 64 will be processed.

Java

The recommended package is the latest version from the official Oracle **OpenJDK** page. The **OpenJDK Java** used in Ubuntu is also fine. The minimum is JDK 1.8, or even JRE 1.8 (the Java Runtime Environment), but, as said before, better use the latest JDK (currently 1.8.151). On macOS the last Apple Java implementation is 1.6, so it is required to use the Oracle version.

Important note: Java may be distributed in separate 32/64-bit packages. Unless you have older applications requiring 32-bit Java, on 64-bit systems it is recommended to install the 64-bit JDK. Anyway, Oracle JDK 9 might not even be available for 32-bit.

On some distributions, for example on Ubuntu 16.04 LTS, Java seems to be already installed in the standard distribution:

```
$ java -version
openjdk version "1.8.0_121"
OpenJDK Runtime Environment (build 1.8.0_121-8u121-b13-0ubuntu1.16.04.2-b13)
OpenJDK 64-Bit Server VM (build 25.121-b13, mixed mode)
```

On other distributions, for example on Ubuntu 14.04 LTS, you can install the OpenJDK run-time and test if it was properly installed with the following commands:

```
$ sudo apt-get -y install default-jdk
$ java -version
java version "1.7.0_85"
OpenJDK Runtime Environment (IcedTea 2.6.1) (7u85-2.6.1-5ubuntu0.14.04.1)
OpenJDK 64-bit Server VM (build 24.85-b03, mixed mode)
```

If you are behind a firewall, older Java virtual machines fail to connect to SourceForge. Update your Java JDK to **8u101 or later** and retry to install the plug-ins.

Java Cryptography Extension (JCE)

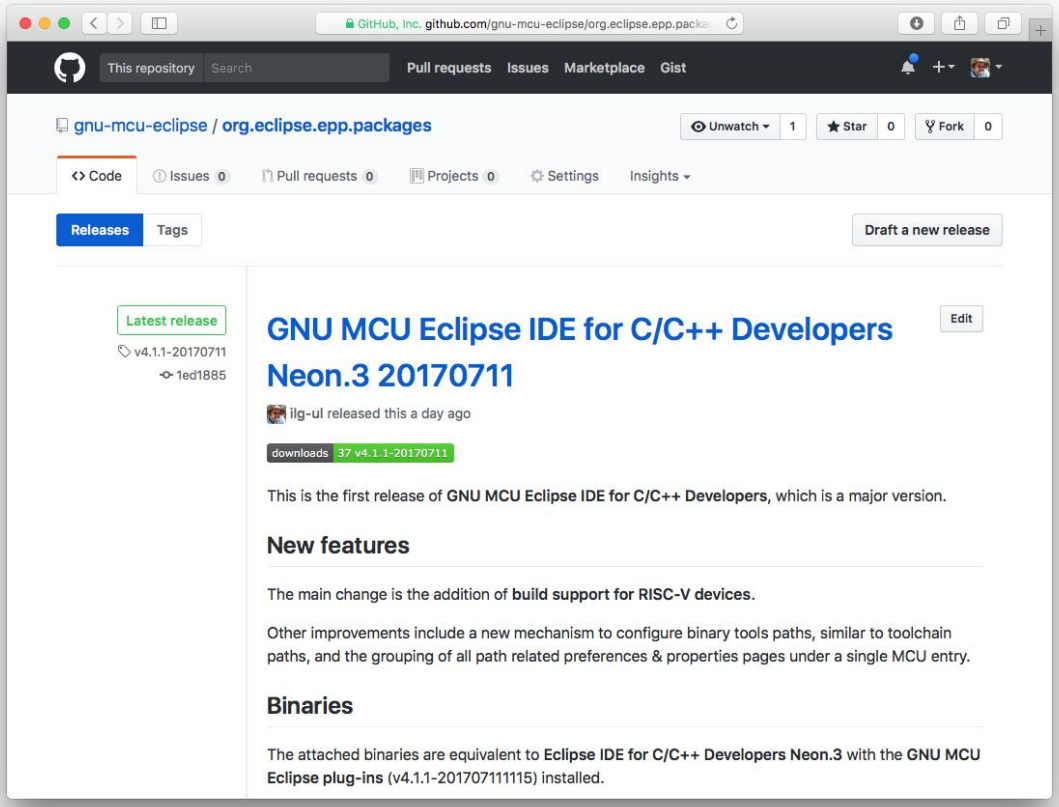
If you use the Oracle JDK, starting with mid January 2017, attempts to install via HTTPS from SourceForge might fail with handshake_error. Install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files and the installer should be able to reach the update site.

Eclipse & CDT

Starting with version 4.x, the oldest Eclipse supported by the plug-ins is Eclipse 4.6 Neon.3 (CDT 9.2), and the recommended version is **4.7 Oxygen**. Do not try to install them on Mars, Luna, Kepler, Juno, Indigo, or older versions, since the install will fail.

The package way

The simple way is to download **GNU MCU Eclipse IDE for C/C++ Developers** from GitHub Releases and you get at once both Eclipse CDT and the MCU plug-ins.



Important note: there are different binaries for 32-bit (`_x86`) and 64-bit (`x86-64`). Be sure the Eclipse binary word size matches the Java word size, otherwise the obscure `Java was started but returned exit code=13` message is issued.

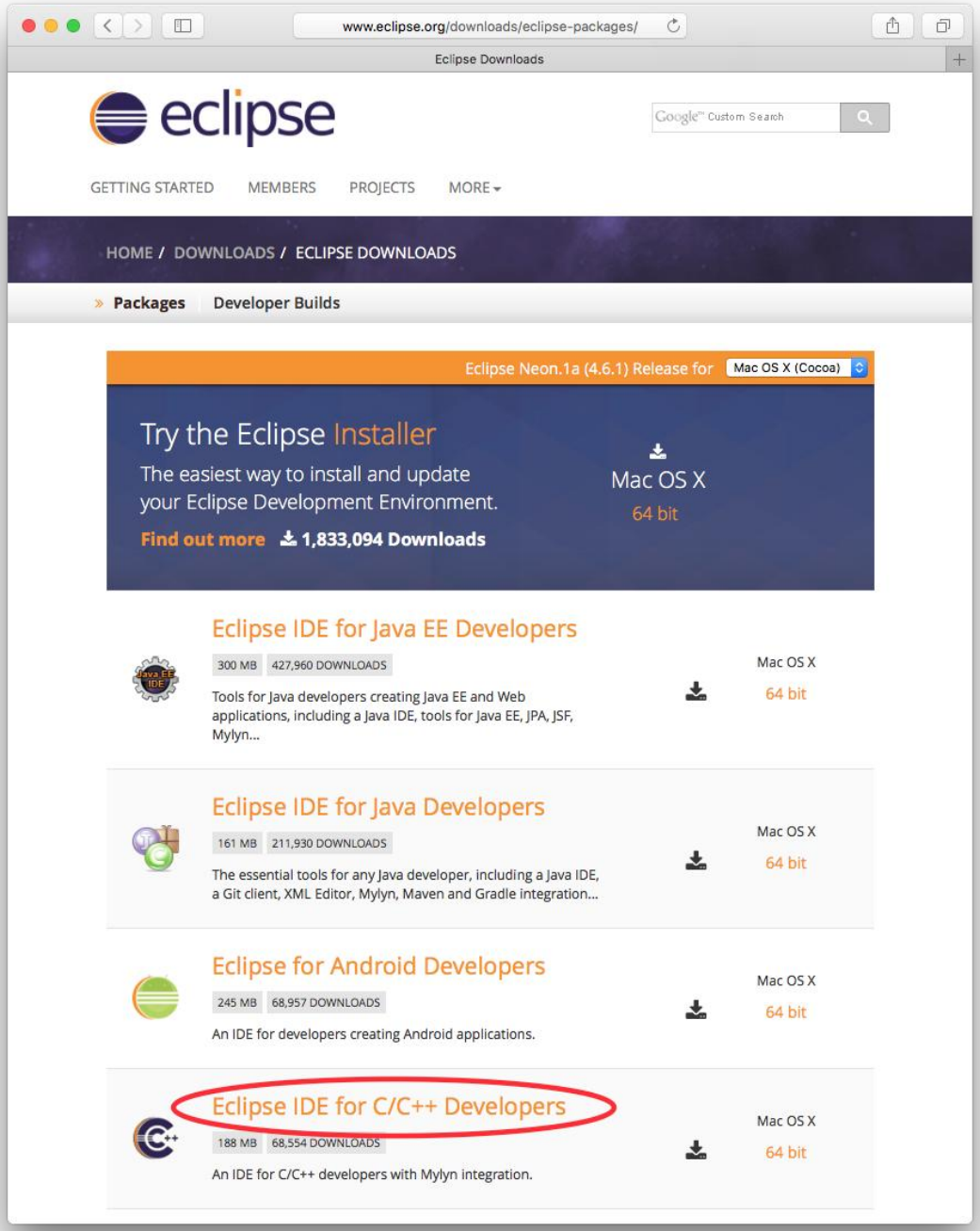
On macOS, after un-archiving, remove the `com.apple.quarantine` extended attribute, otherwise the application will run from a randomised path and you won't be able to store persistent changes.

```
$ xattr -d com.apple.quarantine Eclipse.app
```

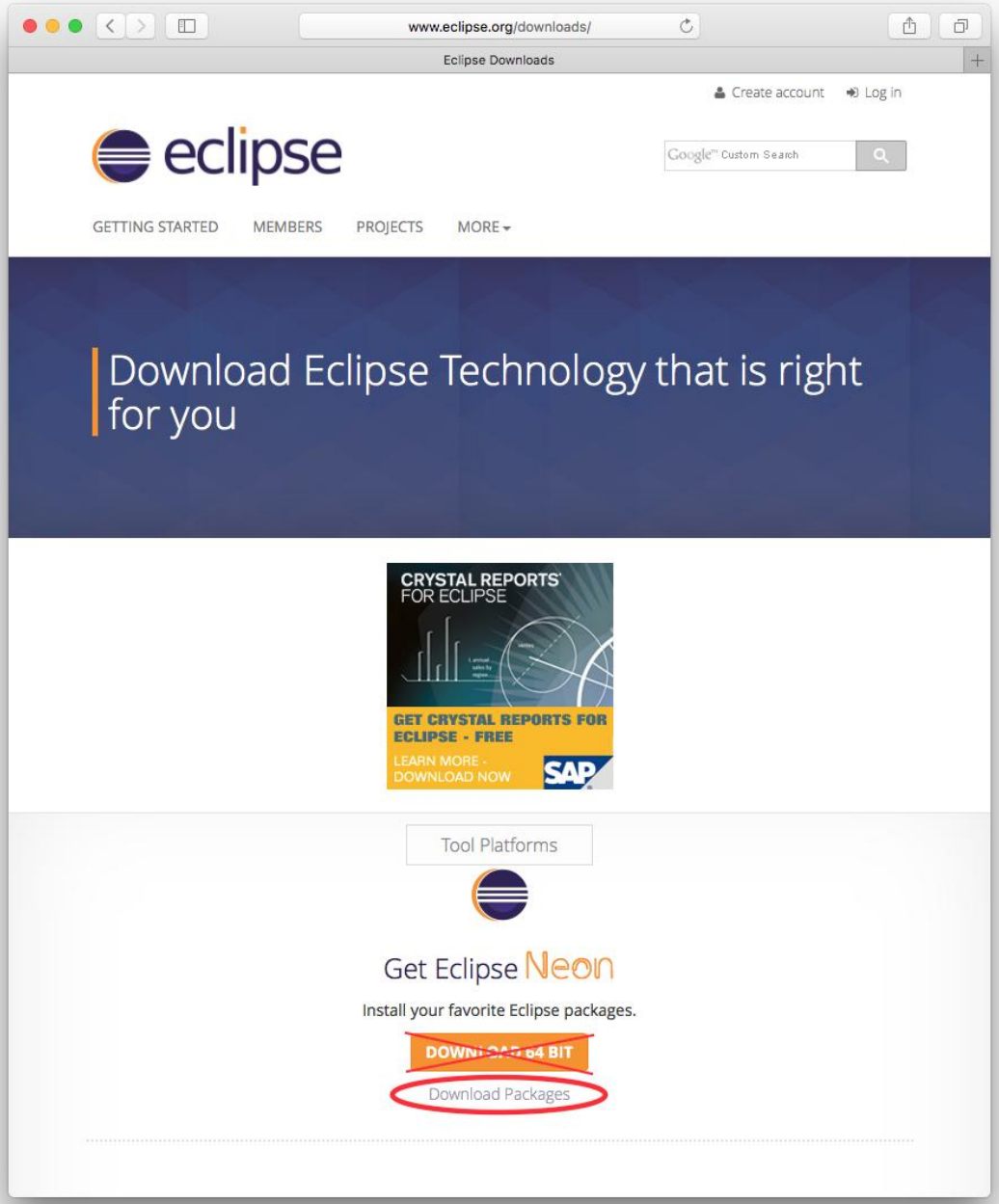
The plug-ins install way

If you prefer to do this in more steps, go to the Eclipse packages and get the **Eclipse IDE for C/C++ Developers** archive appropriate for your platform, preferably the 64-bit version.

Links to older versions are available in the **More Downloads** section, at the bottom.



In case you reached the main Eclipse downloads page, avoid the recommended download and follow the link to the Eclipse packages.



Traditionally Eclipse does not need an installer, Eclipse is distributed as a plain archive. To install Eclipse, simply unpack the archive at a place of your choice and start using it.

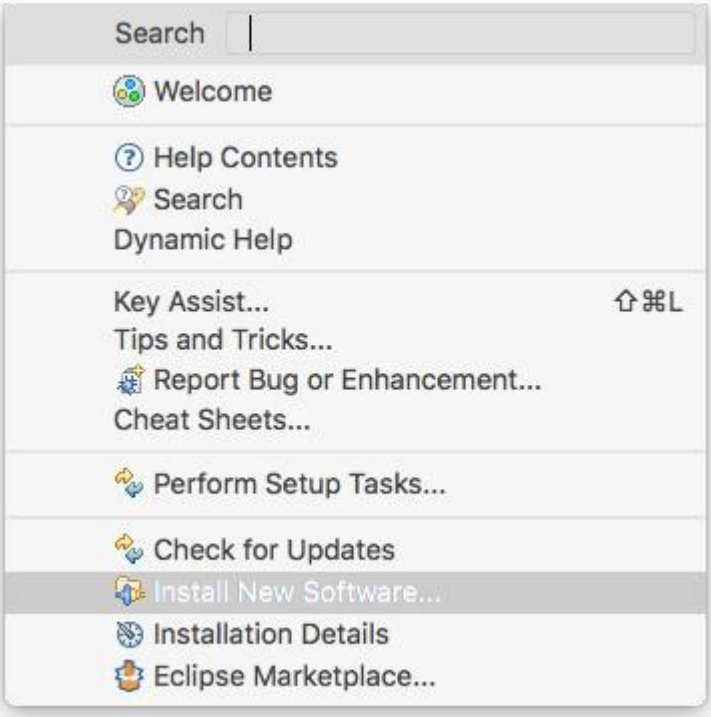
*Note 1: on GNU/Linux be sure you manually install the Eclipse; **DO NOT** try to install it via the **Synaptics** package manager, or similar, since usually you get an older version and the CDT plug-ins are not included.*

Note 2: Contrary to other tools, Eclipse does not need administrative rights, does need not be installed in a system location, and does not need not be a single instance on a system. Actually you are encouraged to install multiple instances of Eclipse, possibly one for each project type, to avoid mixing plug-ins from different sources in case different project types.

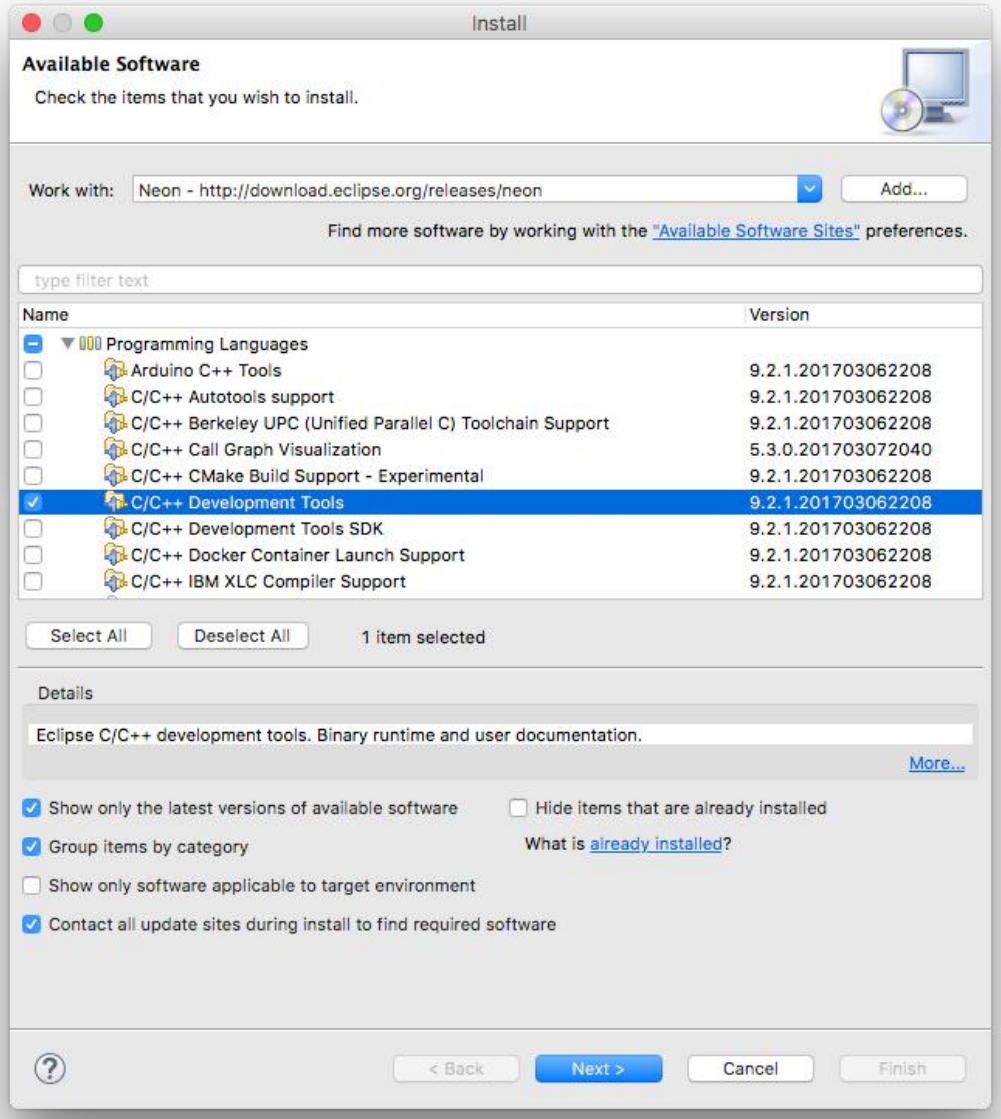
CDT

The minimum CDT version is 9.2.1, that came with Eclipse Neon. Do not try to use earlier versions, since either the install will fail (with something like *… requires 'bundle org.eclipse.cdt 9.2.1' but it could not be found*), or it will not run properly.

As mentioned before, the recommended way is to use a fresh **Eclipse IDE for C/C++ Developers** for the cross ARM development projects. Even if you did so, but especially if you did not do so, it is a good idea to check if you really have the latest version available. For this, enter the *Eclipse* menu and go to **Help → Install New Software**



- select *Work with:* **Neon** (or more recent)
- if the *Group items by category* is enabled, expand the **Programming Languages** group
- select the **C/C++ Development Tools** feature
- click the **Next** button and follow the usual installation procedure



Compatibility issues

Please note that starting with 4.x, compatibility with Eclipses previous than 4.6 Neon was no longer possible. If, for any reason, you need a solution for Eclipse Luna or Mars, you can try the GNU MCU Eclipse Plug-in version 3.*, but please keep in mind that this version is no longer maintained.

Similarly:

- starting with 3.1.x, compatibility with Eclipses previous than 4.4 Luna was no longer possible. If, for any reason, you need a solution for Eclipse Kepler, you can try the GNU MCU Eclipse Plug-in version 2.12, but please keep in mind that this version is no longer maintained.
- starting with 1.1.x, compatibility with Eclipses previous than 4.3 Kepler was no longer possible. If, for any reasons, you need a solution for older Eclipse versions, you can try the GNU MCU Eclipse Plug-in version 0.5.5, but please keep in mind that this old version is no longer maintained.

Plug-ins install/update

The following samples are from a slightly older Eclipse 4.3 Kepler on macOS. Other platforms may behave slightly different, but the idea is the same.

Warning: Be sure your Eclipse is 4.6 Neon.3 or later, otherwise the plug-ins will refuse to install, claiming **Missing requirements** and **Cannot satisfy dependency**.

The Eclipse Marketplace way

The easiest way to install/update the plug-ins is to use the Eclipse Marketplace:

- go to the *Eclipse* menu → **Help** → **Eclipse Marketplace…**
- in the Find field, enter **GNU MCU Eclipse**
- click the **Go** button

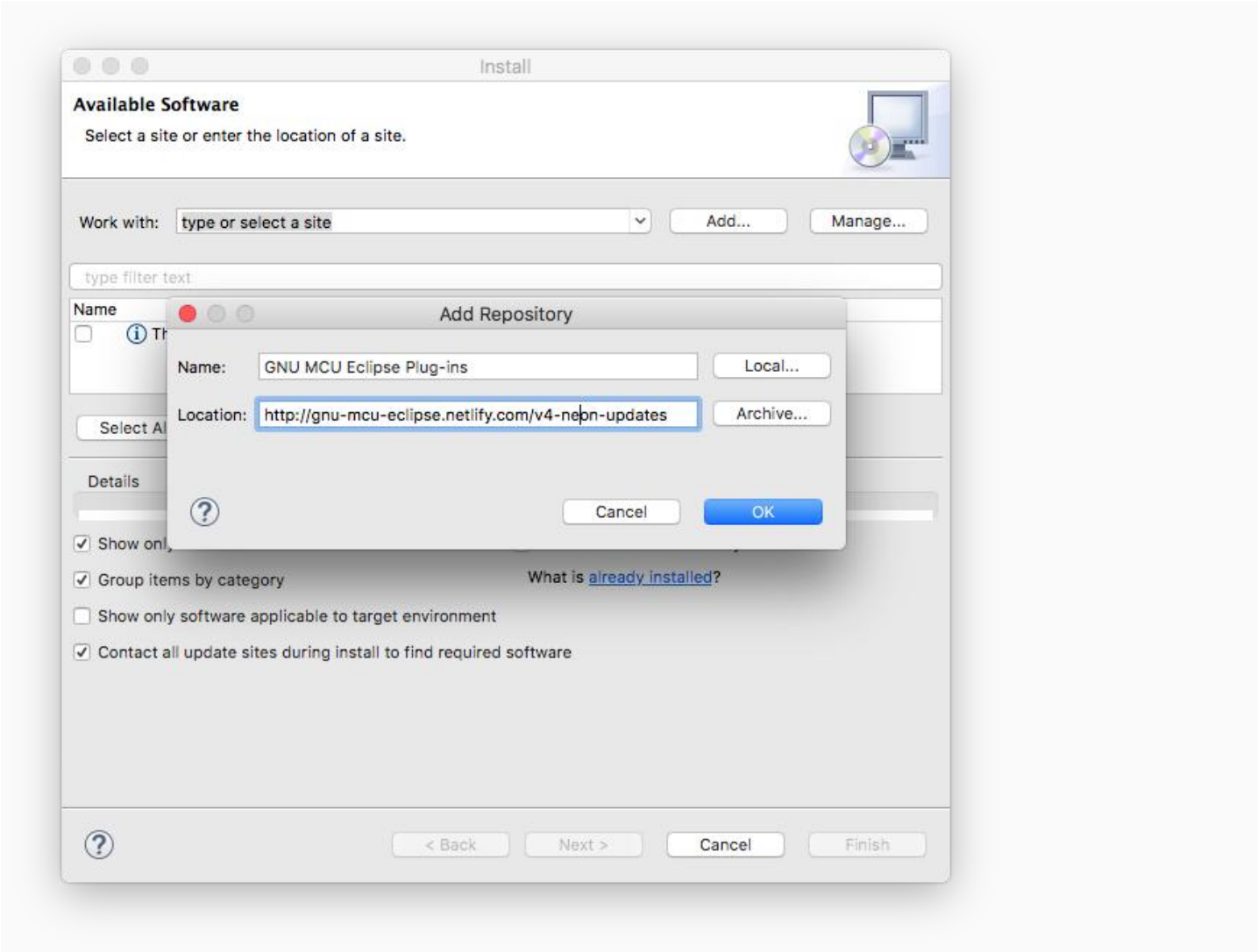
This should identify GNU MCU Eclipse and offer to Install/Update/Uninstall

assets

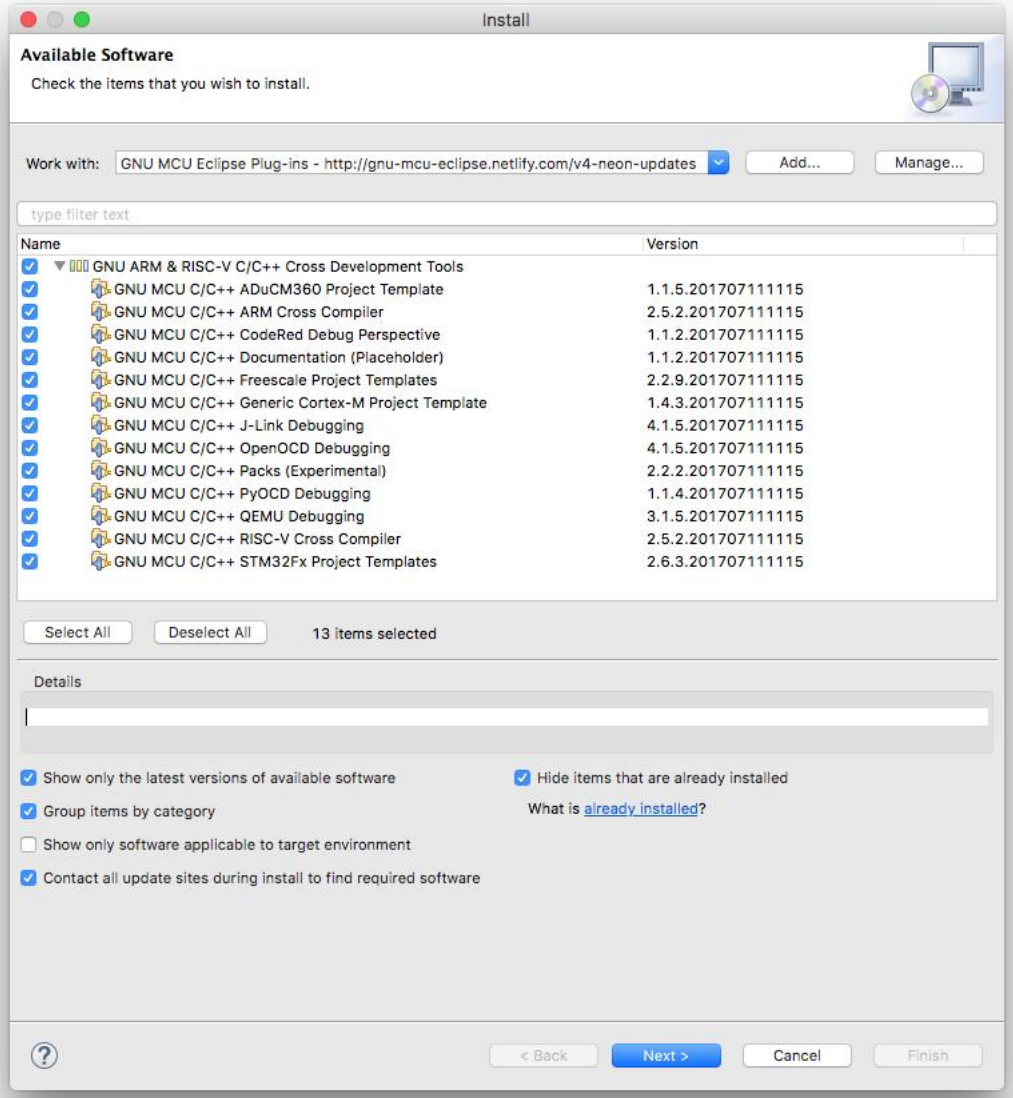
The Eclipse update site way

The classical way to install the GNU MCU Eclipse plug-ins is to use the Eclipse standard install/update mechanism: In the *Eclipse* menu → **Help** → **Install New Software…**

- in the *Install* window, click the **Add…** button (on future updates, select the URL in the **Work with:** combo)
- fill in *Name:* with **GNU MCU Eclipse Plug-ins**
- fill in *Location:* with **http://gnu-mcu-eclipse.netlify.com/v4-neon-updates**
- click the **OK** button



- normally the main window should list a group named **GNU ARM & RISC-V Cross Development Tools**; expand it
- (in case the main window will list *There are no categorized items*, you are probably using a very old version; disable the Group items by category option)
- select all the plug-ins (the one marked *End of life* is needed only for compatibility with previous version, normally can be safely skipped)
- click the **Next** button and follow the usual installation procedure

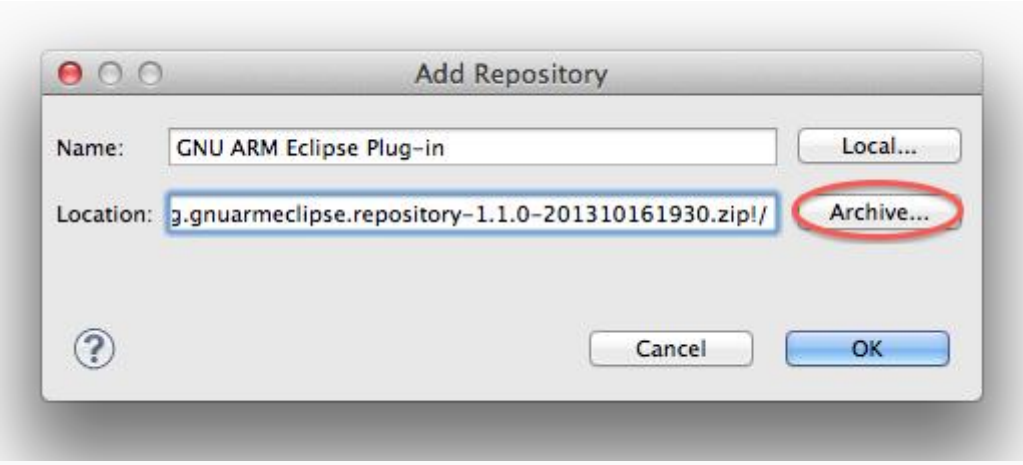


Once you define the update site URL, further updates are greatly simplified (**Help** → **Check For Updates**).

The local archive way

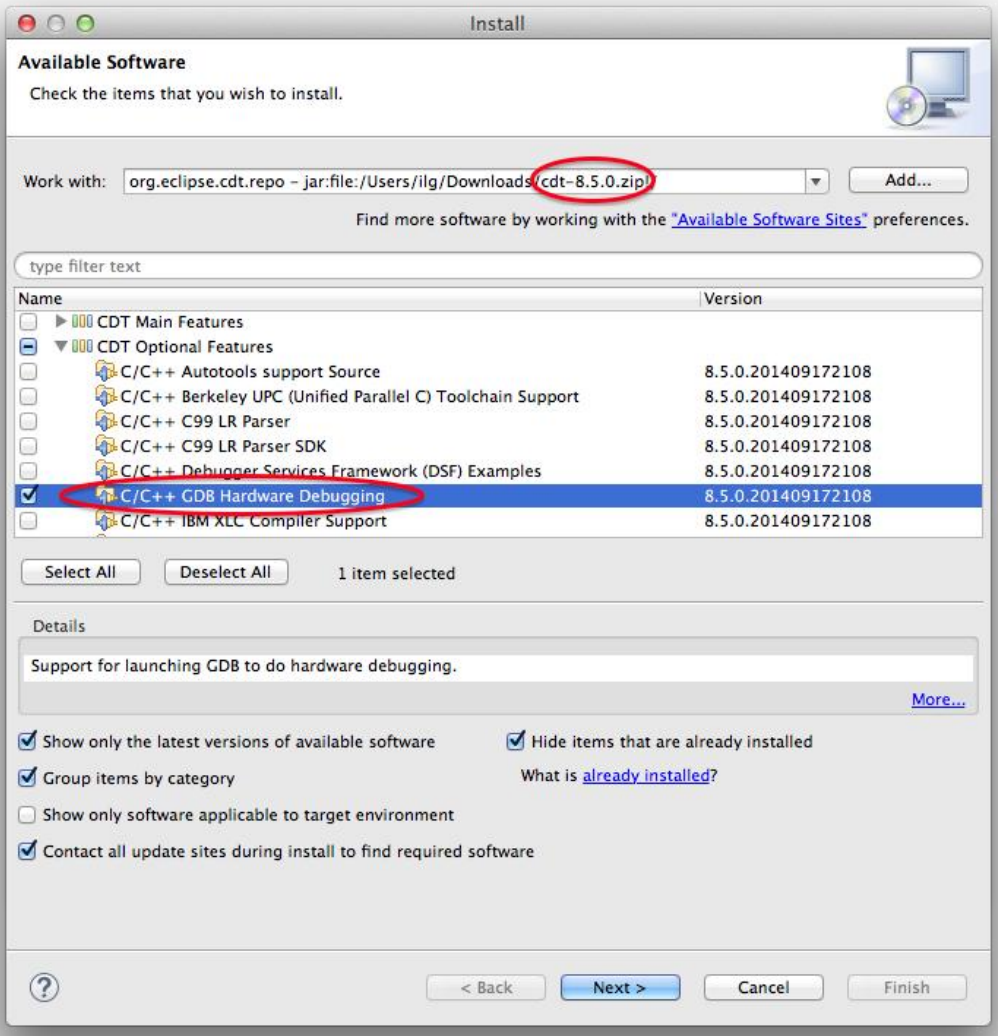
If, for any reason, you need to install a specific version of the plug-in, the solution is to download the archive from GitHub Releases, or, for older releases, from SourceForge Files and to point the Eclipse update mechanism to the local archive: In the* Eclipse* menu → **Help** → **Install New Software…**

- in the *Install* window, click the **Add…** button
- in the *Add Repository* window, click the **Archive…** button * in the *Repository archive* window, select the downloaded .zip archive * click the **OK** button
- click the OK button
- follow the usual installation procedure



Off-line install

If, for any reasons, you need to install the GNU MCU Eclipse plug-ins on a system without permanent Internet connection, and decide for the alternate way presented above, please be aware that the debug plug-ins require the presence of the **C/C++ GDB Hardware Debugging** plug-in. Usually this plug-in is not included in the standard package, but is present in the **CDT Optional Features** category, packed with the CDT archives available from the [CDT download site](). Download the desired archive, add it to your update sites, select the **C/C++ GDB Hardware Debugging** plug-in, restart as usual, and then install the GNU MCU Eclipse plug-ins from the local archive, as above.

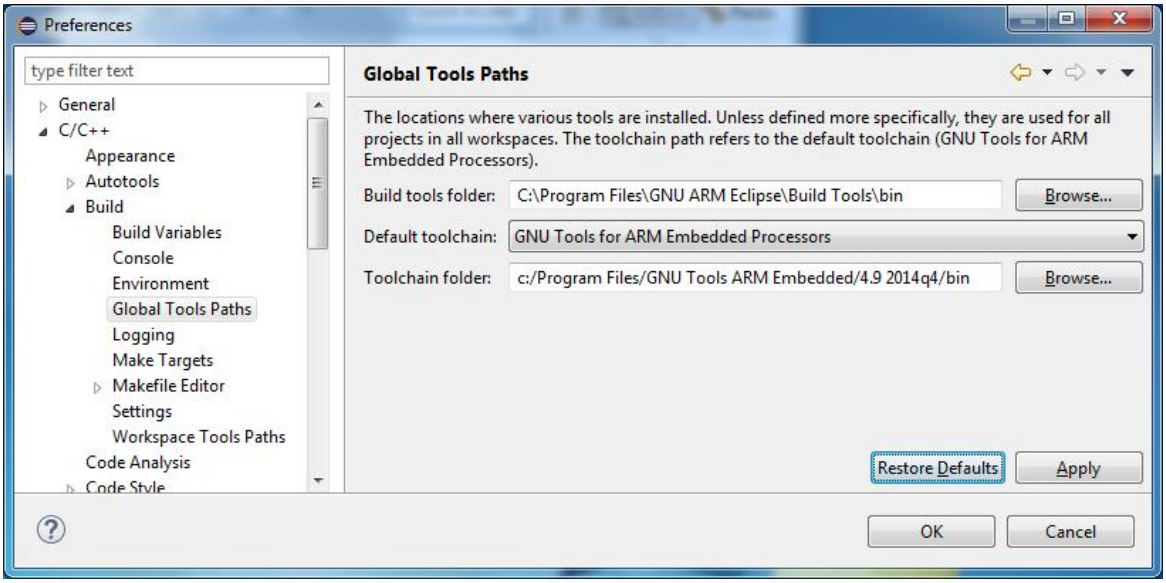


Note: Attempts to install the GNU MCU Eclipse plug-ins off-line without having the **C/C++ GDB Hardware Debugging** installed fails with an error related to installing the `ilg.gnumcueclipse.debug.gdbjtag.jlink.feature.group` and other debugging features.

On-line install do not have this problem since the Eclipse automatically downloads the C/C++ GDB Hardware Debugging plug-in from the CDT update site.

Check/set the global tools paths

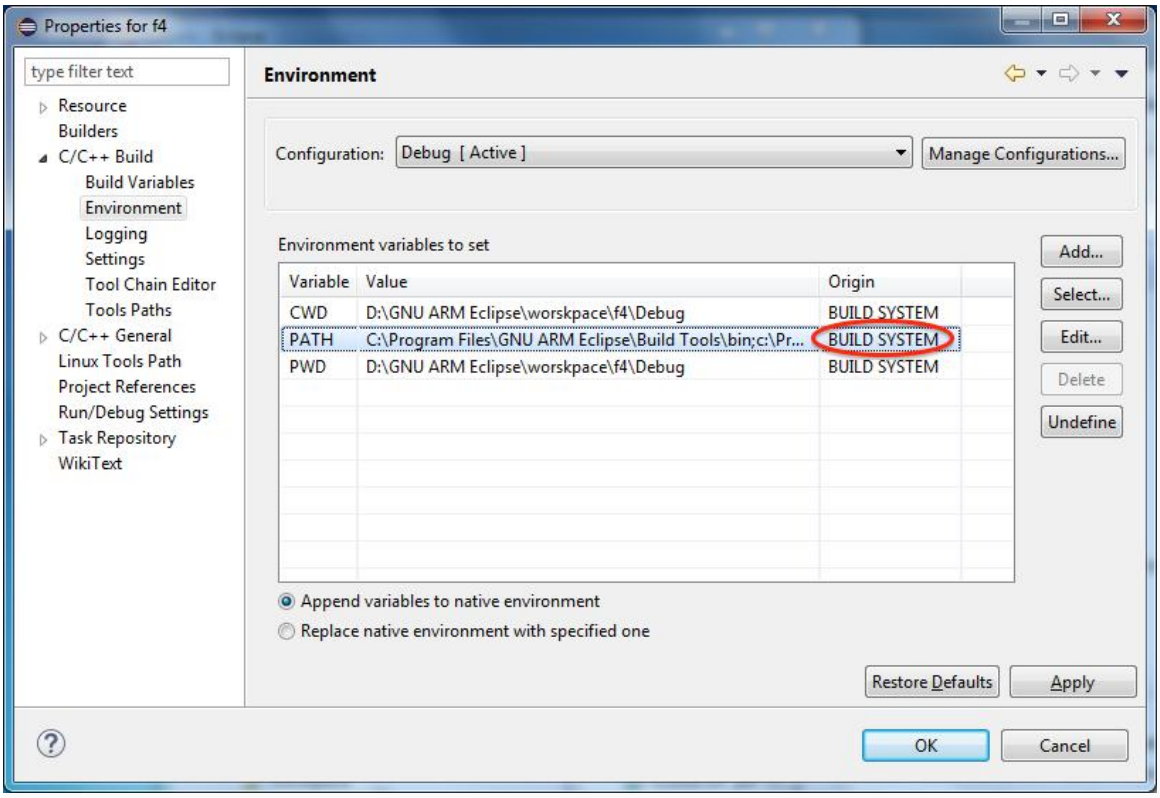
If you already installed the toolchain (and, on Windows, the build tools) in the default locations, as suggested in the above steps, the plug-ins might have automatically discovered them.



The above definition will make the toolchain and build tools accessible to all projects in all workspaces.

If needed, you can define different paths per workspace (**Workspace Tools Paths**) or even per project (**Tools Paths** in the project properties).

To check if the paths definitions are effective, go to the project properties page and identify the `PATH` variable. Be sure the **Origin** column reads `BUILD SYSTEM`; if you manually edit it, the Origin will change and will read `USER`, but this is totally not recommended, since manually editing the path disables further automated updates of the path.



Troubleshooting

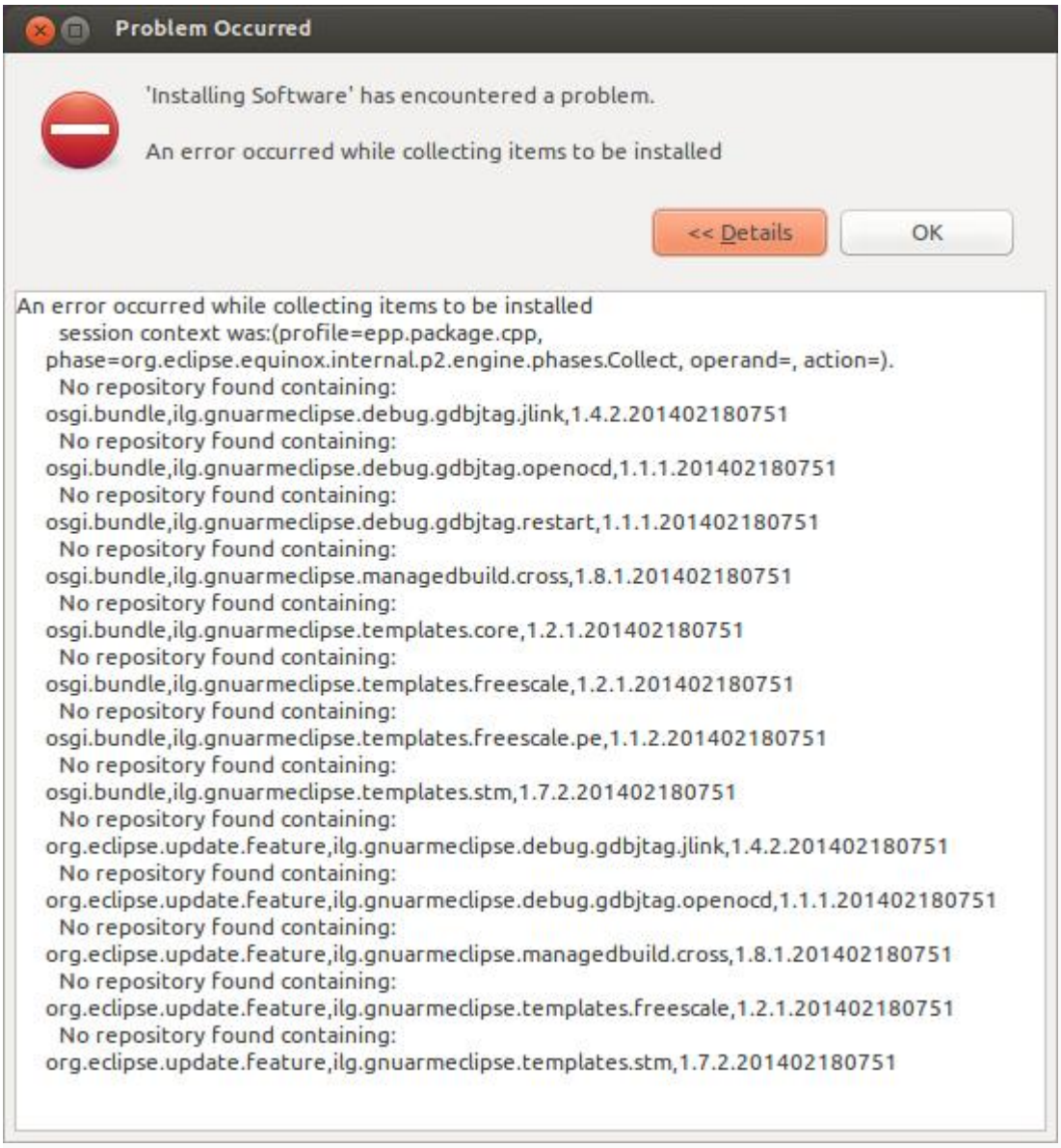
Java security problems

- if you use the Oracle JDK, starting with mid January, attempts to install from SourceForge fail with handshake_error. Install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files and retry to install the plug-ins.
- if you are behind a firewall, older Java virtual machines fail to connect to SourceForge. Install the latest JDK and retry to install the plug-ins.

SourceForge problems

Note: starting with version v4.1.1, the SourceForge server is no longer actively used, so this problem should no longer be actual.

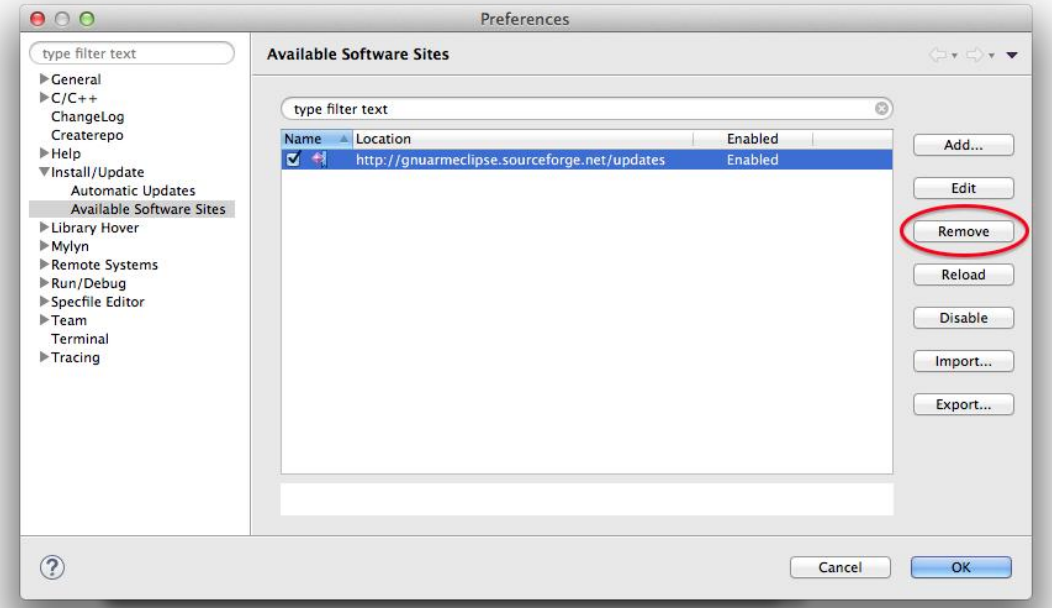
There are cases when the on-line access to the SourceForge mirror servers might not be reliable. In these cases the Eclipse update system may fail. For example:



If this happens, Eclipse remains in a confused state and rarely can recover itself.

The workaround is to remove the GNU MCU Eclipse update site from the available sites, restart Eclipse and add it again.

For this go to *Eclipse* menu → **(Window →) Preferences** → **Install/Update** → **Available Software Sites**:



If this still fails, the alternative method is to manually download the latest version archive and to perform the install from it, as presented below.

Incomplete Eclipse

If Eclipse complains that some required items could not be found, most probably you are trying to install the plug-ins on an older Eclipse, or on another Eclipse package than the recommended **Eclipse IDE for C/C++ Developers**.

```
Cannot complete the install because one or more required items could not be found.
  Software being installed: GNU ARM C/C++ J-Link Debugging 4.1.1.201606210758
(ilg.gnuarmeclipse.debug.gdbjtag.jlink.feature.group 4.1.1.201606210758)
  Missing requirement: GNU ARM C/C++ Core 3.1.1.201606210758 (ilg.gnuarmeclipse.core
3.1.1.201606210758) requires 'bundle org.eclipse.cdt 8.6.0' but it could not be found
  Cannot satisfy dependency:
    From: GNU ARM C/C++ J-Link Debugging 4.1.1.201606210758
(ilg.gnuarmeclipse.debug.gdbjtag.jlink 4.1.1.201606210758)
    To: bundle ilg.gnuarmeclipse.core 0.0.0
  Cannot satisfy dependency:
    From: GNU ARM C/C++ J-Link Debugging 4.1.1.201606210758
(ilg.gnuarmeclipse.debug.gdbjtag.jlink.feature.group 4.1.1.201606210758)
    To: ilg.gnuarmeclipse.debug.gdbjtag.jlink [4.1.1.201606210758]
```

Reinstall the correct Eclipse package, or add CDT to the existing Eclipse (**Programming Languages** → **C/C++ Development Tools**).

Ubuntu GTK issue

Eclipse is incompatible with the GTK version 3 distributed with Ubuntu (confirmed even with Ubuntu 16LTS). To overcome this, disable the use of GTK 3, by adding the following line to your environment:

```
$ export SWT_GTK3=0
```

If, for any reason, a global setting is not possible, the GTK version can be set in `eclipse.ini`:

```
--launcher.GTK_version
2
```

You might need to place this definition before:

```
--launcher.appendVmargs
```

This requirement is anecdotal and appears on some forums; the Eclipse documentation does not mention it.

macOS com.apple.quarantine

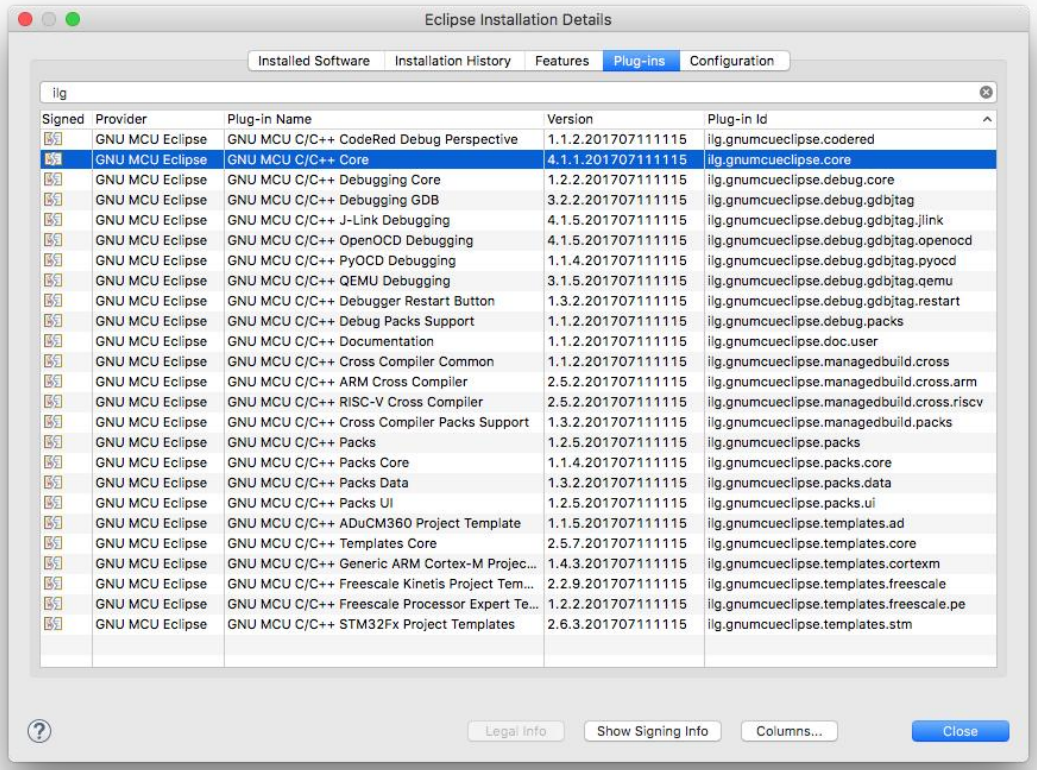
On macOS Sierra and later, unsigned applications are marked as *quarantined* and subject to a path randomization, which prevents Eclipse to maintain persistent preferences (for more details see [blog post]). To disable this, remove the `com.apple.quarantine` extended attribute from `Eclipse.app`:

```
$ xattr Eclipse.app
com.apple.quarantine
$ xattr -d com.apple.quarantine Eclipse.app
```

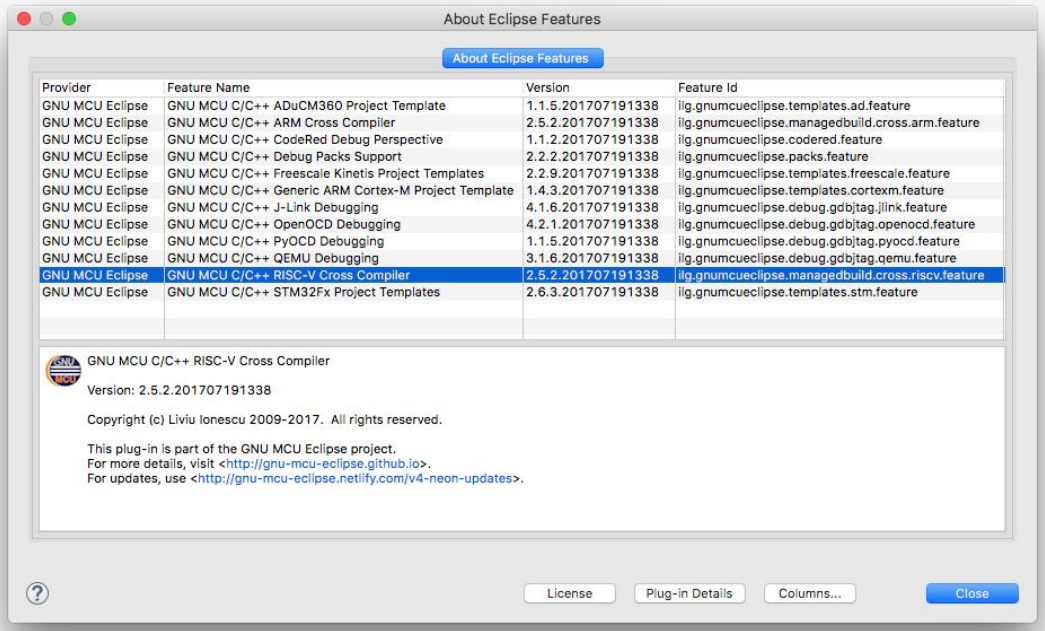
Plug-ins versions

One of the confusing details of the GNU MCU Eclipse plug-ins versioning system is matching the version from the release announcement (for example **Version 2.6.1-201502281154 released**) with existing plug-ins/features.

The short answer is that the announced version refers to the update site, also packed as an archive, and available from SourceForge. The same version is also used for the `ilg.gnuarmeclipse.core` plug-in:



Although the core plug-in is referred by all features, this version number cannot be found in the list of the features:



However, the common point for all these magic numbers is the release date, *201707191338* in this case, which is identical for all plug-ins and features.

Thus, the recommended method to **identify plug-ins and features** is **by date, not by version**, which differs from object to object.

Set workspace preferences

After completing the installation of the plug-ins it is recommended to set the workspace preference.

Download CMSIS Packs

Although support for CMSIS Packs is still experimental, the content of the packs can be used populate the Peripheral registers views during debug sessions, so it is recommended to install the packs for the processors in use.

For this follow the steps in the Packs manager page.

Toolchain and debugging support

By completing the above steps, the Eclipse environment is ready, and you can start creating projects right away. However, for being able to build and debug these projects, you also need to install:

- the toolchain
- the optional build tools (if you are running on Windows)
- the debugging tools

It is also highly recommended to install the Packs plug-in, to switch to the Packs perspective and using the Packs manager to install the packages related to the devices in use. Even if Packs support is not complete yet, you still can benefit from the existing CMSIS Packs, for example by using the peripherals registers view in debug, the documentation view in the C/C++ perspective, etc.

Test project

Follow the above steps and when everything is ready, proceed to create and build a test project.

Support

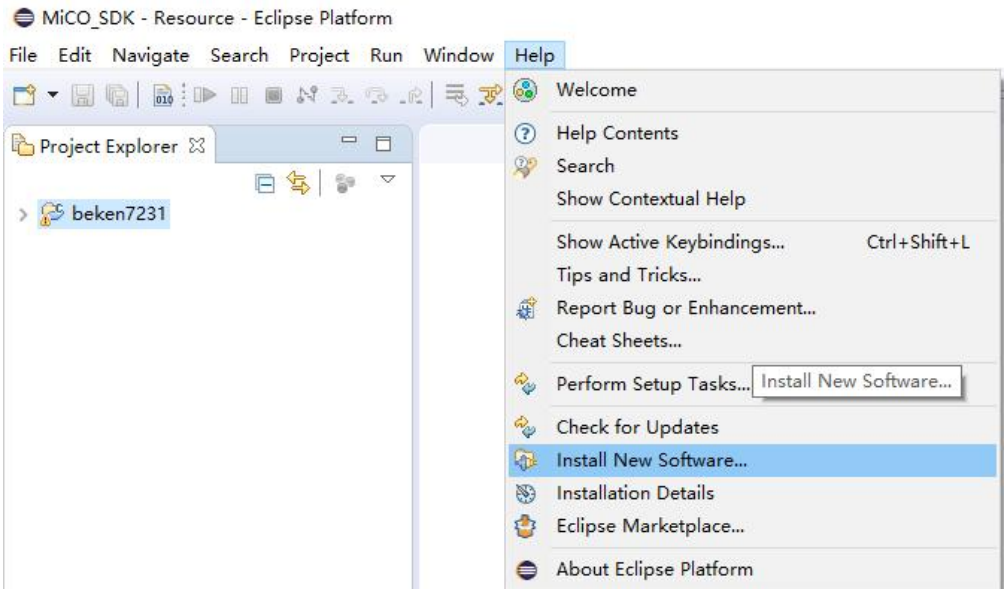
If you encountered any problems when installing the plug-ins, pleas refer to the Support page and **do not** send private email.

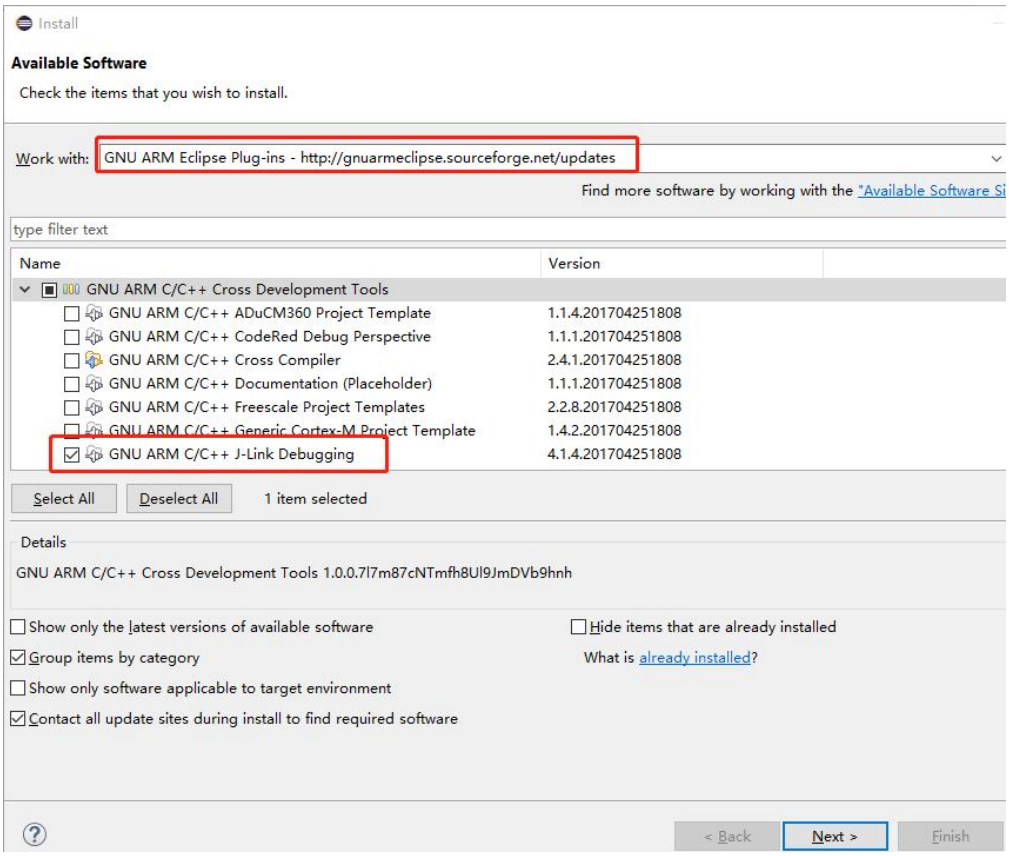
Last modified on Fri Dec 27 09:26:49 2019 UTC.

Micoder + ARM9 + GNU Tools ARM Embedded + ELF + JLINK + GDB

本章节重点配置调试的 Attach to Running Target 功能。前提 JTAG 硬件接口正常的情况下，**当异常发生之后，JLink 随后接入调试**。

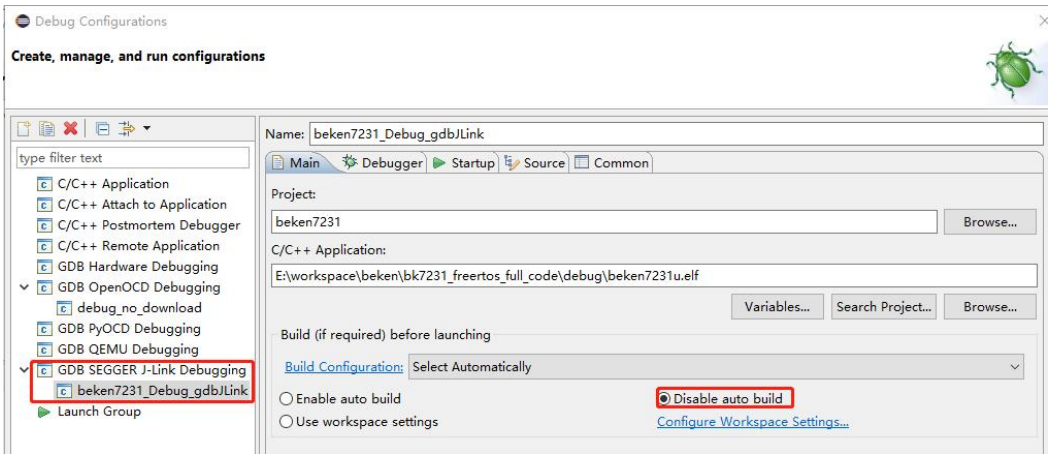
更新 JLink 调试插件





然后选择 Next 或 Yes，完成安装

Attach 调试功能配置



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - debug_no_download
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
  - beken7231_Debug_gdbJLink
- Launch Group

Name: beken7231_Debug_gdbJLink

Attach

Main | Debugger | Startup | Source | Common

J-Link GDB Server Setup

- [x] Start the J-Link GDB server locally
- [x] Connect to running target

Executable: D:\Program Files (x86)\SEGGER\JLink_V630b\JLinkGDBServerCL.exe Browse... Variables...

Device name: LPC2917 Supported device names

Endianness: (•) Little ( ) Big

Connection: (•) USB ( ) IP (USB serial or IP name/address)

Interface: ( ) SWD (•) JTAG

Initial speed: ( ) Auto ( ) Adaptive (•) Fixed 1000 kHz

GDB port: 2331

SWO port: 2332

Telnet port: 2333

- [ ] Verify downloads
- [x] Initialize registers on start
- [x] Local host only
- [ ] Silent

Log file: Browse...

Other options: -singlerun -strict -timeout 0 -nogui

- [x] Allocate console for the GDB server
- [x] Allocate console for semihosting and SWO

GDB Client Setup

Executable: mpiler\arm-none-eabi-5_4-2016q2-20160622\Win32\bin\arm-none-eabi-gdb.exe Browse... Variables...

Other options:

Commands: set mem inaccessible-by-default off

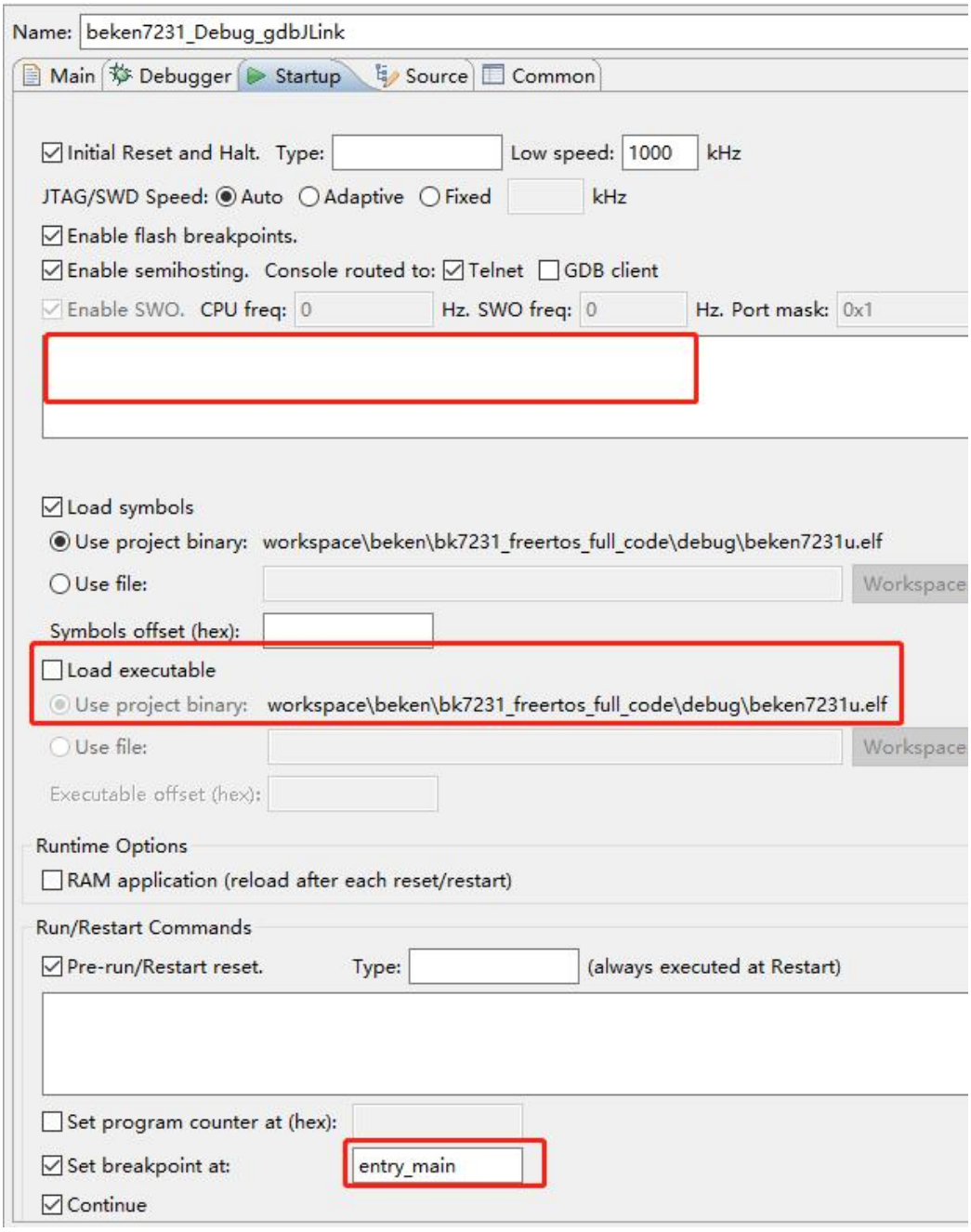
Remote Target

Host name or IP address: localhost

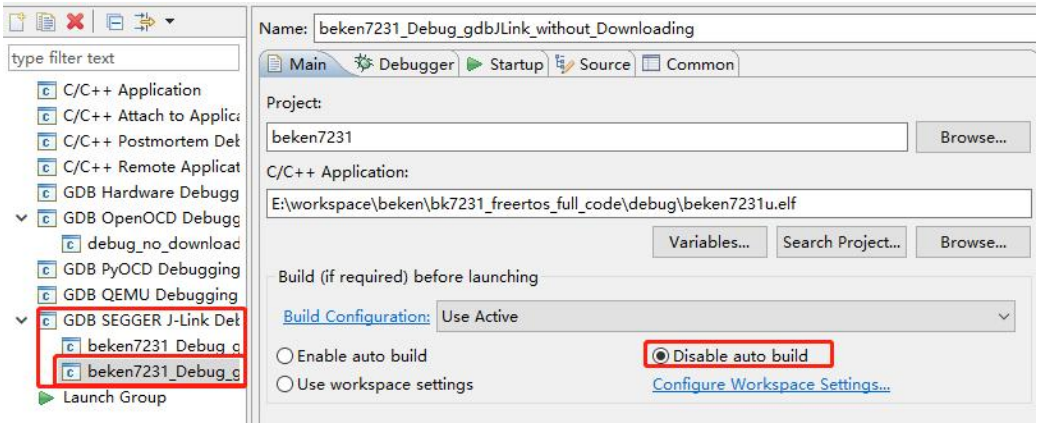
Port number: 2331

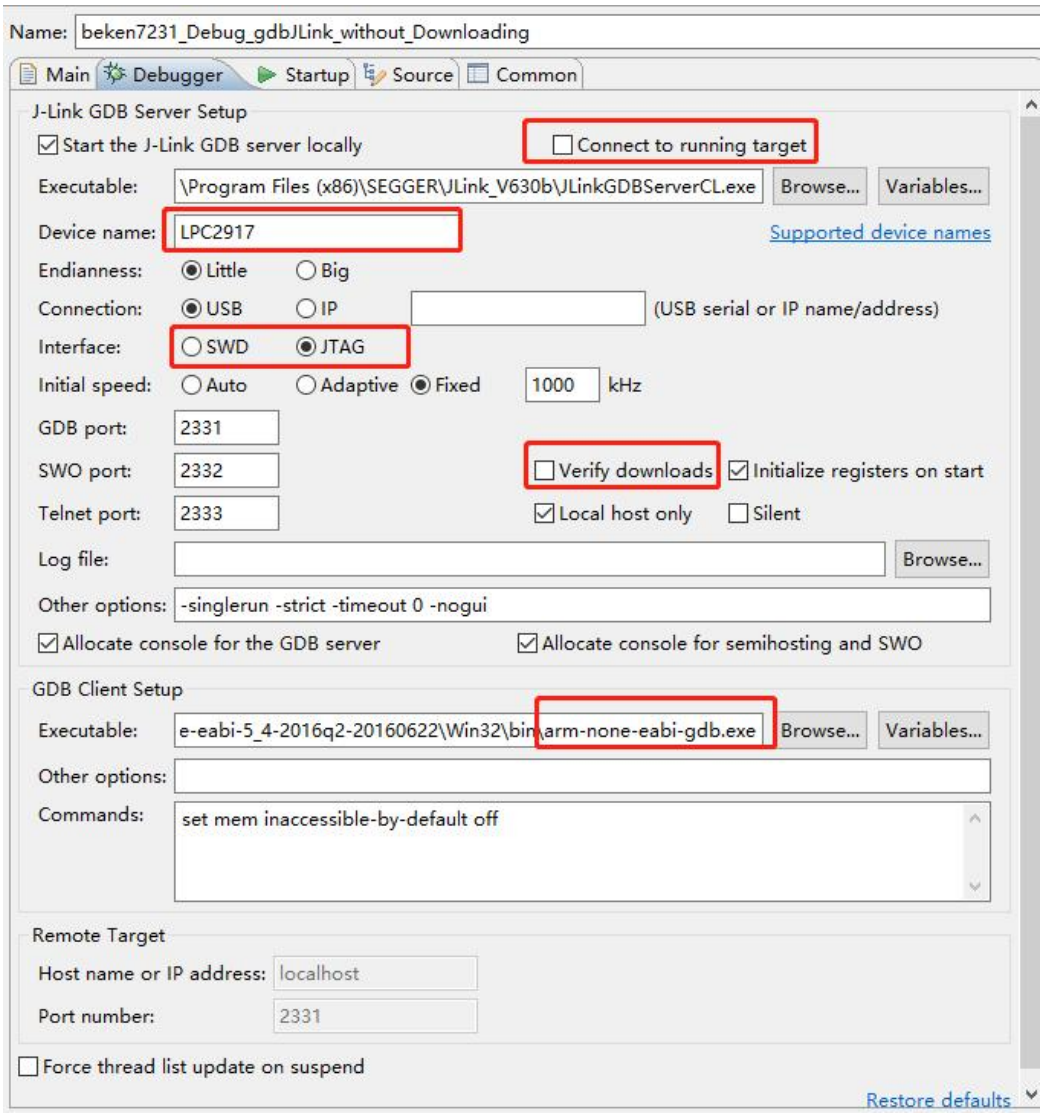
- [ ] Force thread list update on suspend

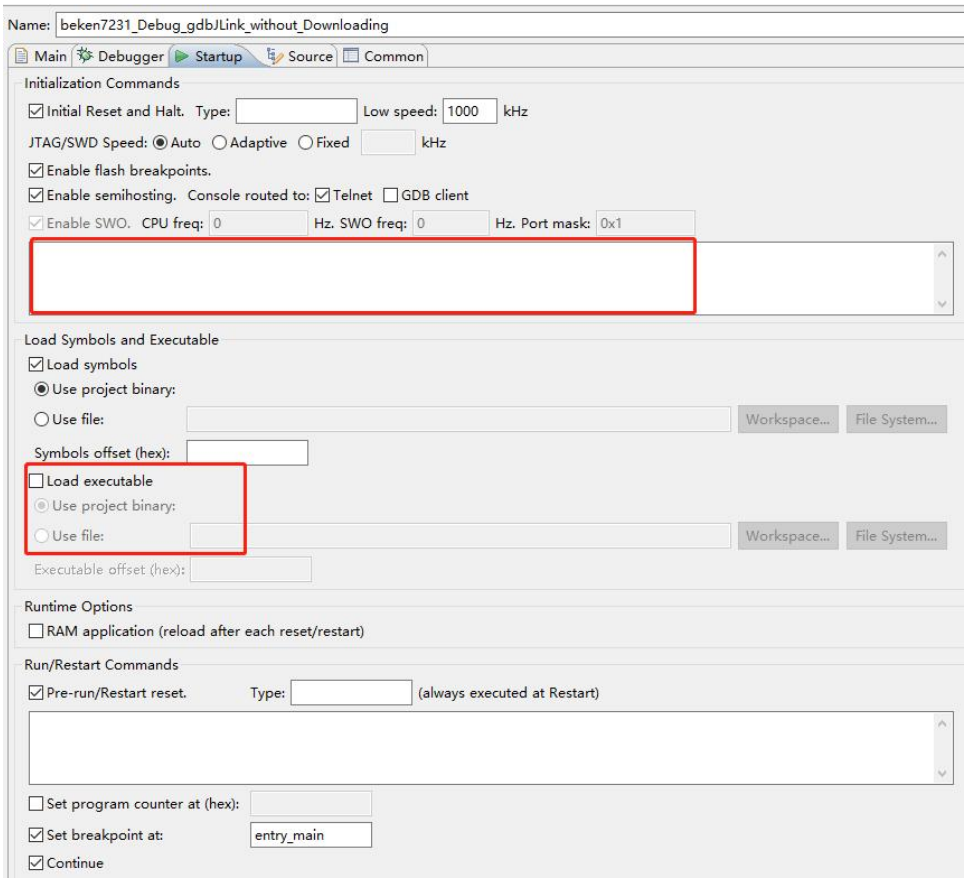
Restore defaults



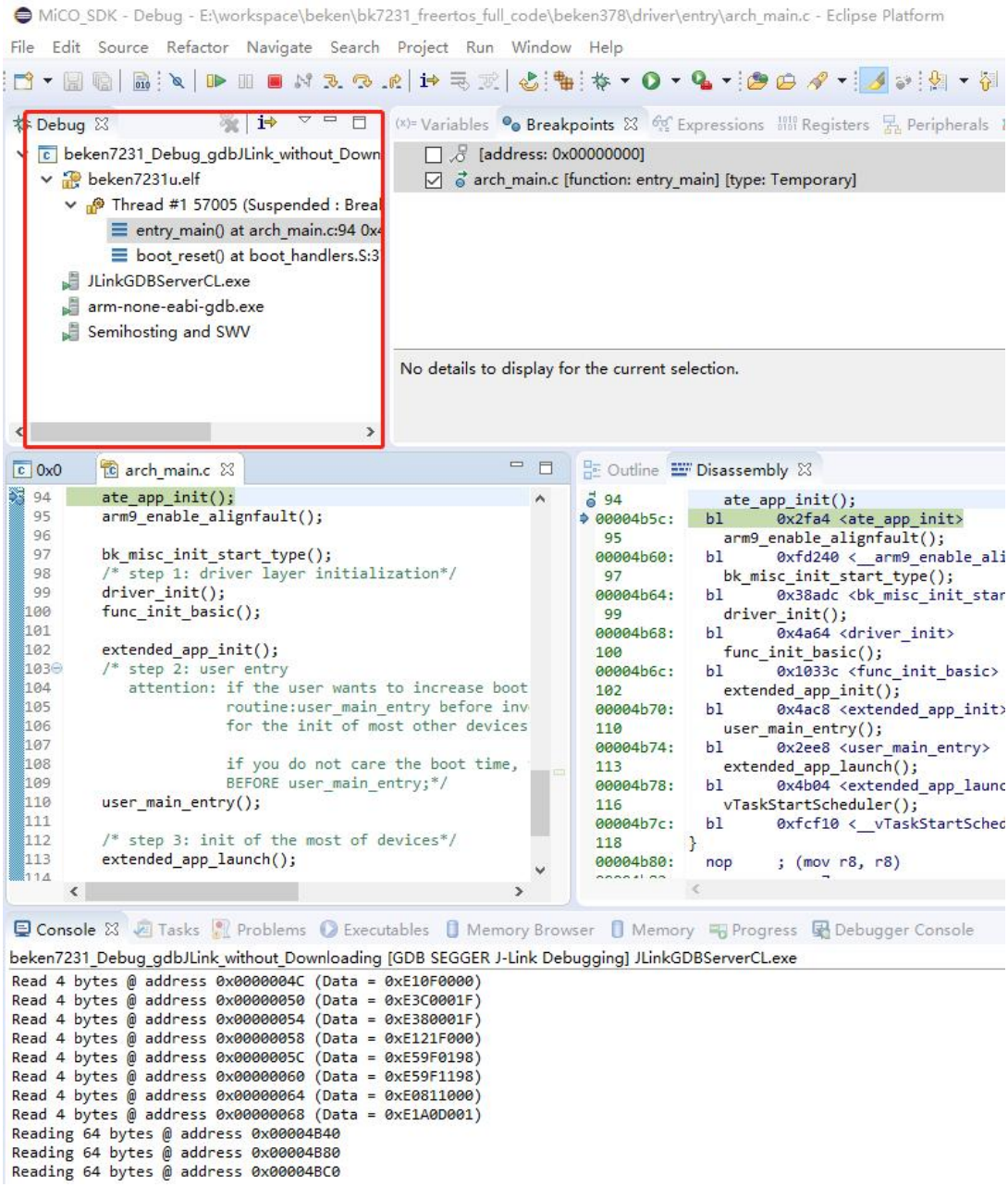
Debug without Downloading 调试配置




Name: beken7231_Debug_gdbJLink_without_Downloading
Main
Debugger
Startup
Source
Common
J-Link GDB Server Setup
Start the J-Link GDB server locally
Connect to running target
Executable:
\Program Files (x86)\SEGGER\JLink_V630b\JLinkGDBServerCL.exe
Browse...
Variables...
Device name:
LPC2917
Supported device names
Endianness:
Little
Big
Connection:
USB
IP
(USB serial or IP name/address)
Interface:
SWD
JTAG
Initial speed:
Auto
Adaptive
Fixed
1000
kHz
GDB port:
2331
SWO port:
2332
Verify downloads
Initialize registers on start
Telnet port:
2333
Local host only
Silent
Log file:
Browse...
Other options:
-singlerun -strict -timeout 0 -nogui
Allocate console for the GDB server
Allocate console for semihosting and SWO
GDB Client Setup
Executable:
e-eabi-5_4-2016q2-20160622\Win32\bin\arm-none-eabi-gdb.exe
Browse...
Variables...
Other options:
Commands:
set mem inaccessible-by-default off
Remote Target
Host name or IP address:
localhost
Port number:
2331
Force thread list update on suspend
Restore defaults


Name: beken7231_Debug_gdbJLink_without_Downloading
Main
Debugger
Startup
Source
Common
Initialization Commands
Initial Reset and Halt.
Type:
Low speed:
1000
kHz
JTAG/SWD Speed:
Auto
Adaptive
Fixed
kHz
Enable flash breakpoints.
Enable semihosting.
Console routed to:
Telnet
GDB client
Enable SWO.
CPU freq:
0
Hz.
SWO freq:
0
Hz.
Port mask:
0x1
Load Symbols and Executable
Load symbols
Use project binary:
Use file:
Workspace...
File System...
Symbols offset (hex):
Load executable
Use project binary:
Use file:
Workspace...
File System...
Executable offset (hex):
Runtime Options
RAM application (reload after each reset/restart)
Run/Restart Commands
Pre-run/Restart reset.
Type:
(always executed at Restart)
Set program counter at (hex):
Set breakpoint at:
entry_main
Continue

请查看成功的喜悦：

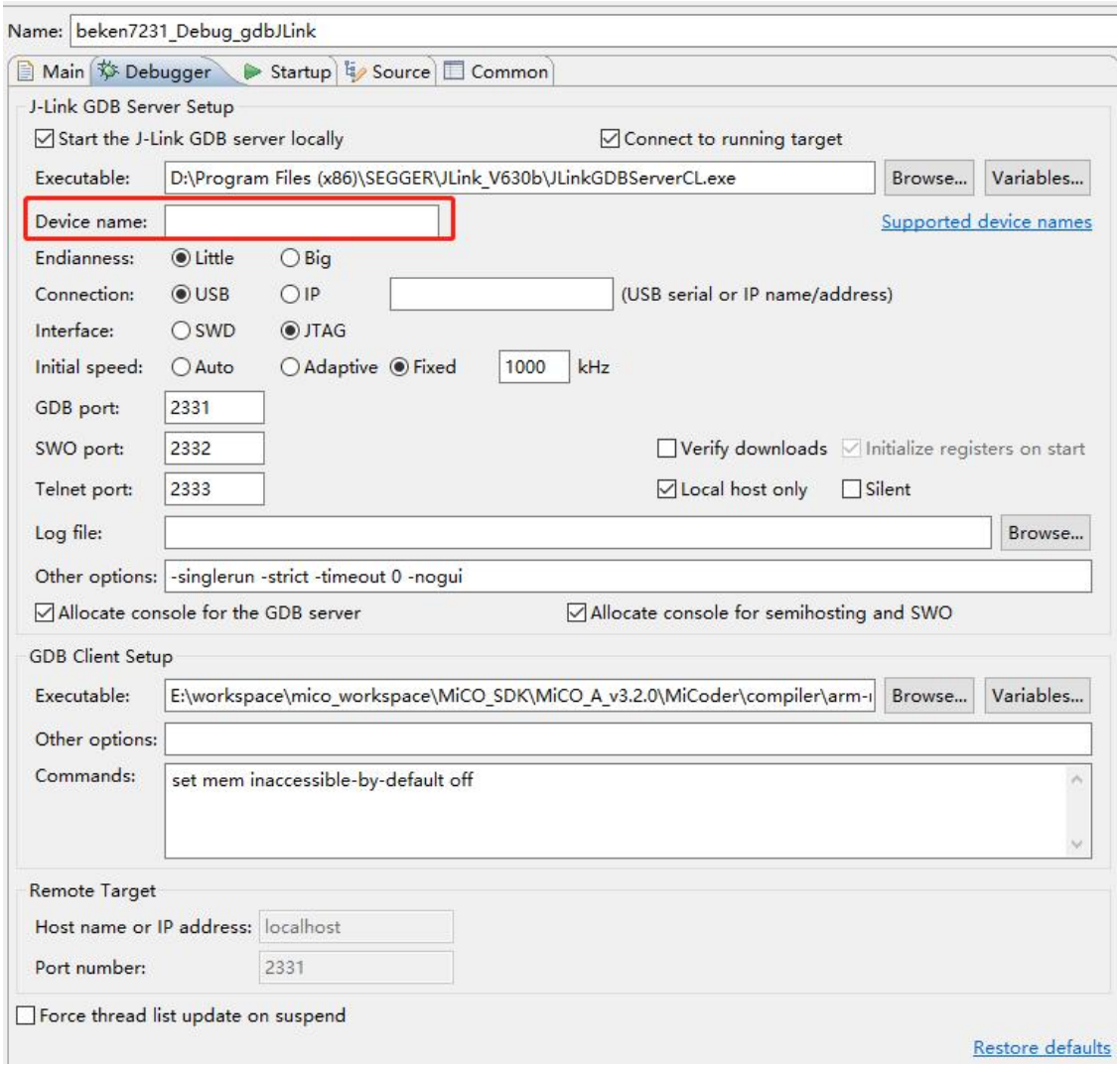


Debug with Downloading 调试配置

参见相关章节，请自由发挥，谢谢

备注

Device Name



当 Device Name 选为 **DUMMY 或者 Unspecified**，不能工作。

```
D:\Program Files (x86)\SEGGER\JLink_V630b>JLinkGDBServerCL.exe
SEGGER J-Link GDB Server V6.30b Command Line Version

JLinkARM.dll V6.30 (DLL compiled Jan 30 2018 18:44:37)

-----GDB Server start settings-----
GDBInit file:                      none
GDB Server Listening port:         2331
SWO raw output listening port:     2332
Terminal I/O port:                 2333
Accept remote connection:          localhost only
Generate logfile:                  off
Verify download:                   off
Init regs on start:                off
Silent mode:                       off
Single run mode:                   off
Target connection timeout:         0 ms
------J-Link related settings------
J-Link Host interface:             USB
J-Link script:                     none
J-Link settings file:              none
------Target related settings------
Target device:                     Unspecified
Target interface:                  JTAG
Target interface speed:            4000kHz
Target endian:                     little

Connecting to J-Link...
J-Link is connected.
Firmware: J-Link ARM V8 compiled Nov 28 2014 13:44:46
Hardware: V8.00
S/N: 791950130
Feature(s): RDI,FlashDL,FlashBP,JFlash,GDBFull
Checking target voltage...
Target voltage: 3.28 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 4
JTAG ID: 0x15968001 (ARM9)
Connected to target
```

请查询 Device Name 列表链接：https://www.segger.com/downloads/supported-devices.php。基于 bk72xx 处理器型号，支持的部分设备名如下图：

LPC2917	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2919	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2921	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2923	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2925	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2926	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2927	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2929	ARM968E-S	Parallel CFI NOR flash Internal flash
LPC2930	ARM9	Parallel CFI NOR flash
LPC2939	ARM968E-S	Parallel CFI NOR flash Internal flash

NS9360	ARM9	Parallel CFI NOR flash
		DSPGroup
DA56KLF	ARM968E-S	Parallel CFI NOR flash Internal flash
		Epson
S1S65010	ARM7	Parallel CFI NOR flash
		Eta Compute
ECM3531	Cortex-M3	Parallel CFI NOR flash Program flash
ECM3532	Cortex-M3	Parallel CFI NOR flash Program flash

TRACE32 + BA22

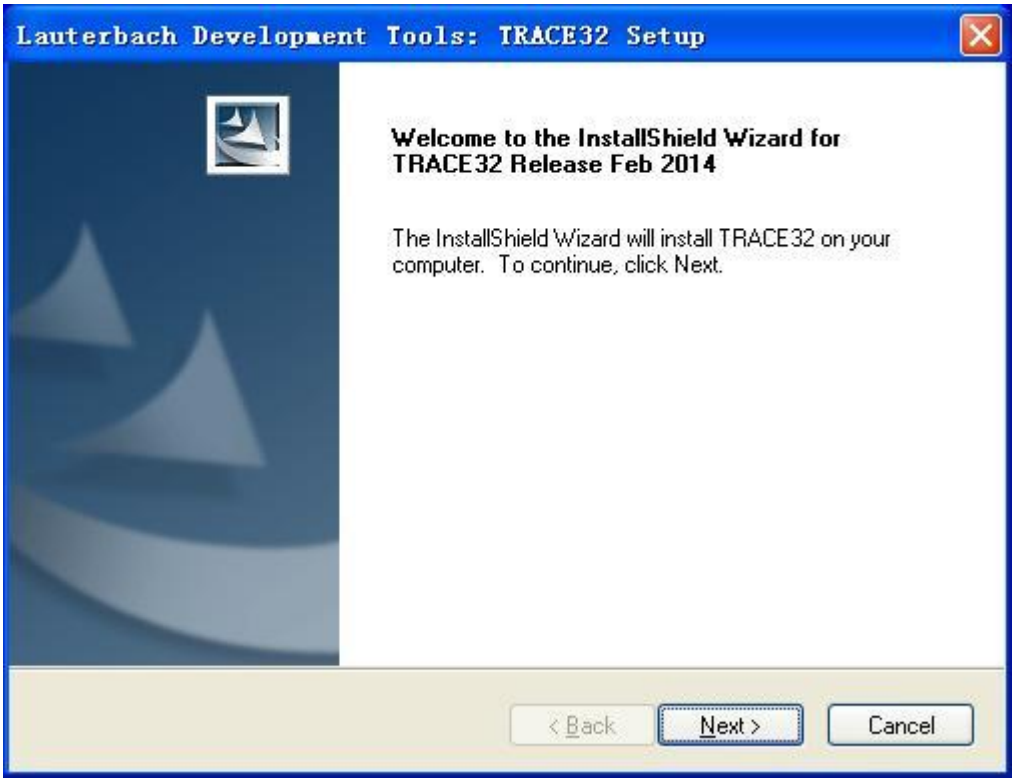
安装

软件安装

目录地址：\\192.168.0.30\日常软件\软件开发\TRACE32_201302

安装步骤

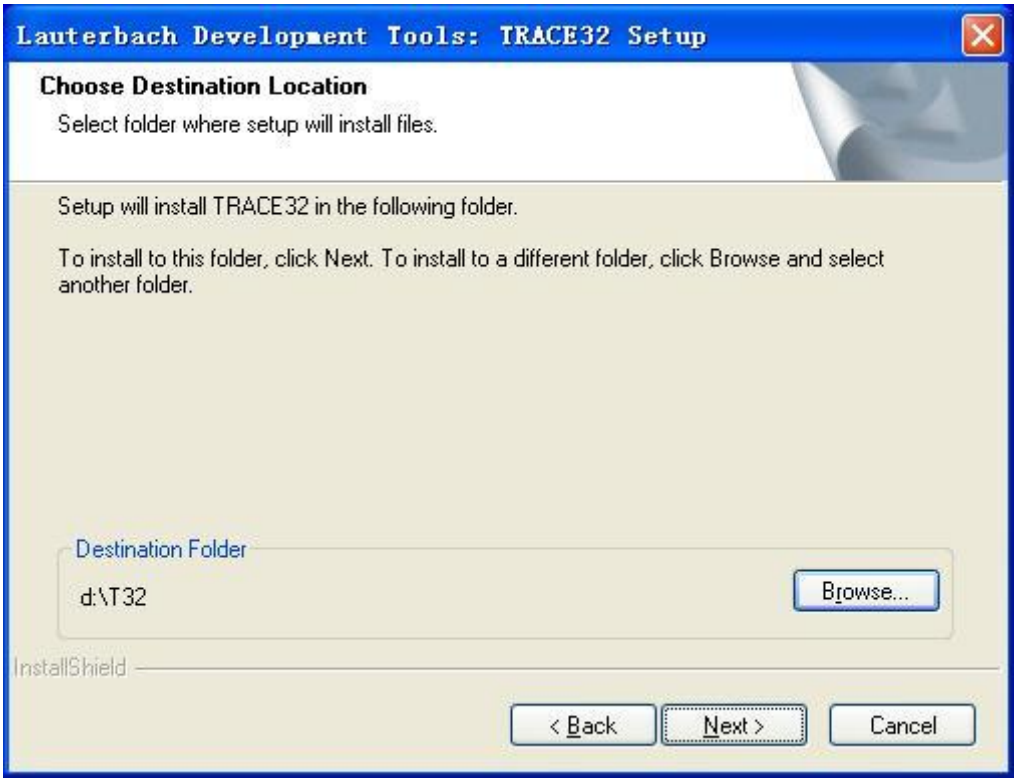
1→



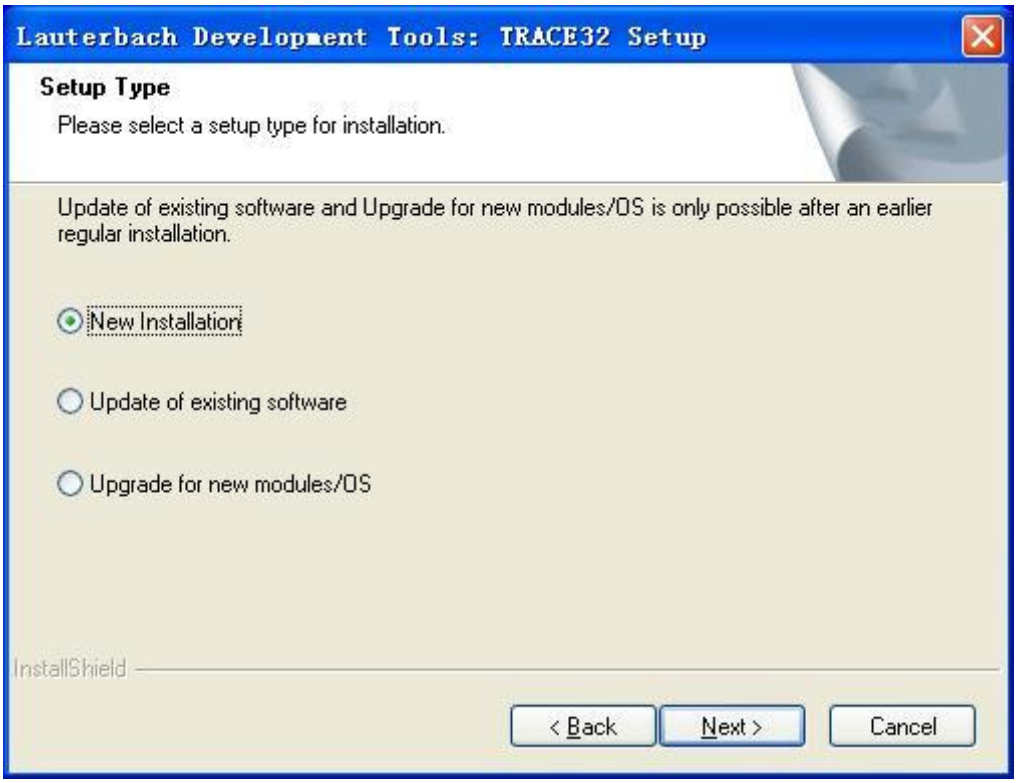
2→

Software License Agreement
Please read the following License Agreement. Use the scroll bar to view the agreement.
End User Licence Agreement for Lauterbach TRACE32 software
This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a legal entity) and Lauterbach GmbH for the TRACE32 software ("SOFTWARE PRODUCT").
By installing, copying, or otherwise using the software, you agree to be bound by the terms of this EULA. If you do not agree to these terms, do not install or use the SOFTWARE PRODUCT.
The SOFTWARE PRODUCT is owned by Lauterbach GmbH. It is licensed, not sold. The software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.
Do you accept all the terms of the preceding License Agreement? If you choose Decline, Setup will close. To install TRACE32 software, you must accept this agreement.
Print
<Back
Accept >
< Decline>

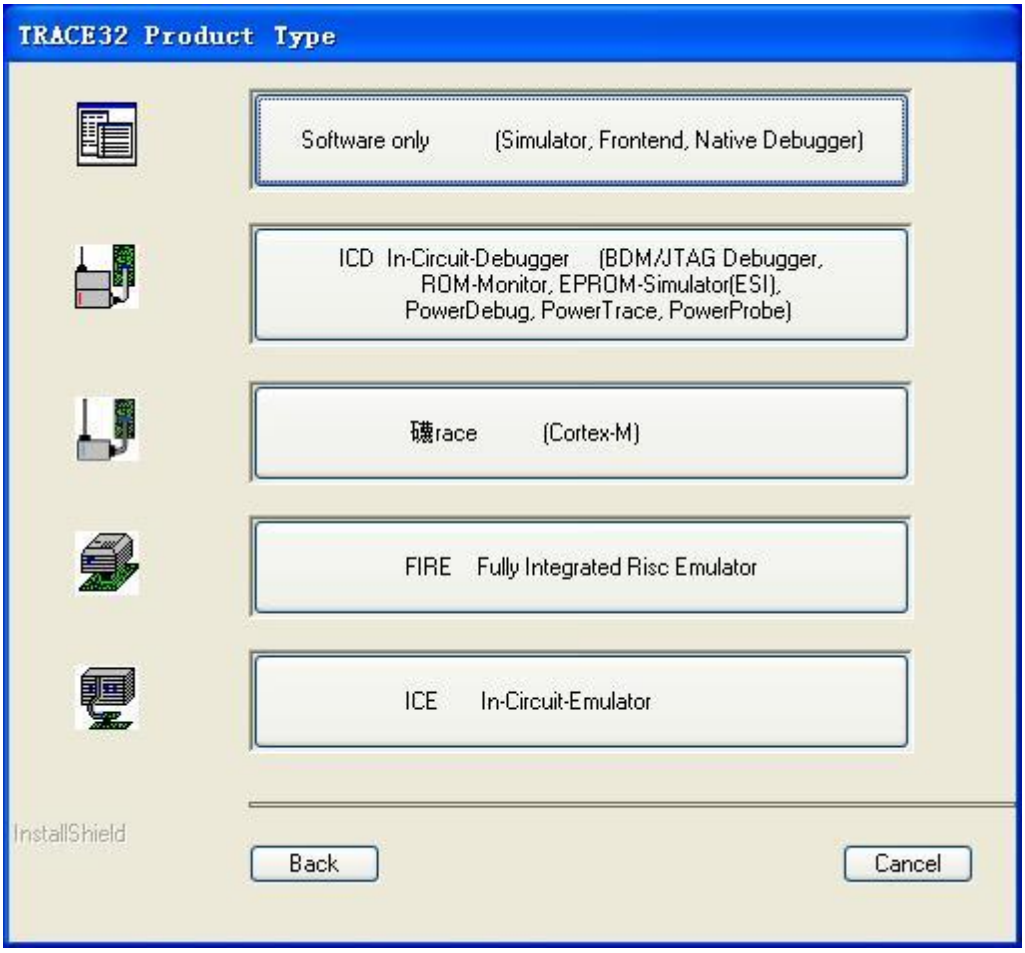
3→


Lauterbach Development Tools: TRACE32 Setup
Choose Destination Location
Select folder where setup will install files.
Setup will install TRACE32 in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
d:\T32
Browse...
InstallShield
< Back
Next >
Cancel

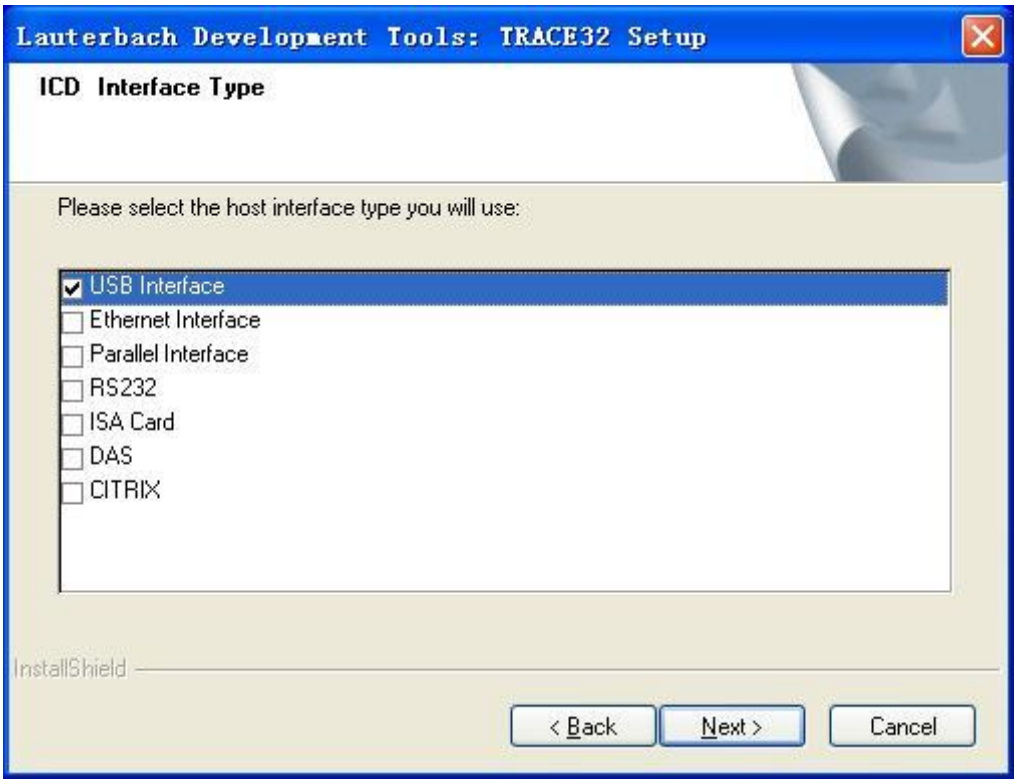
4→



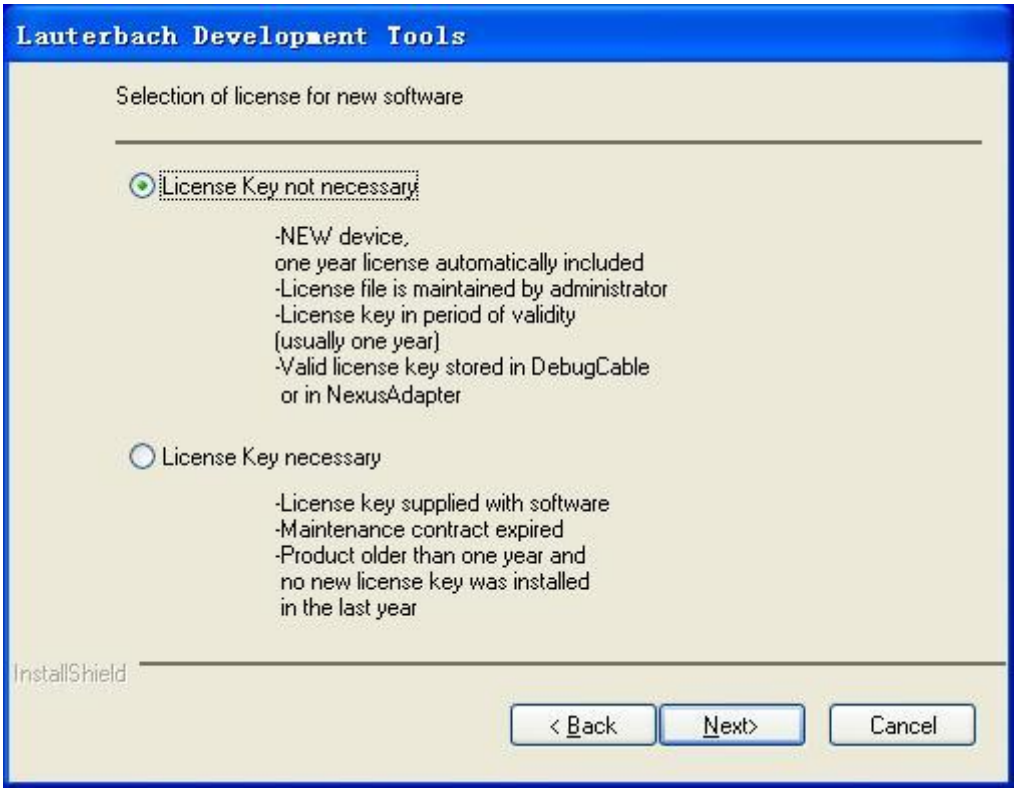
5→（**选择 ICD，第二个框**）



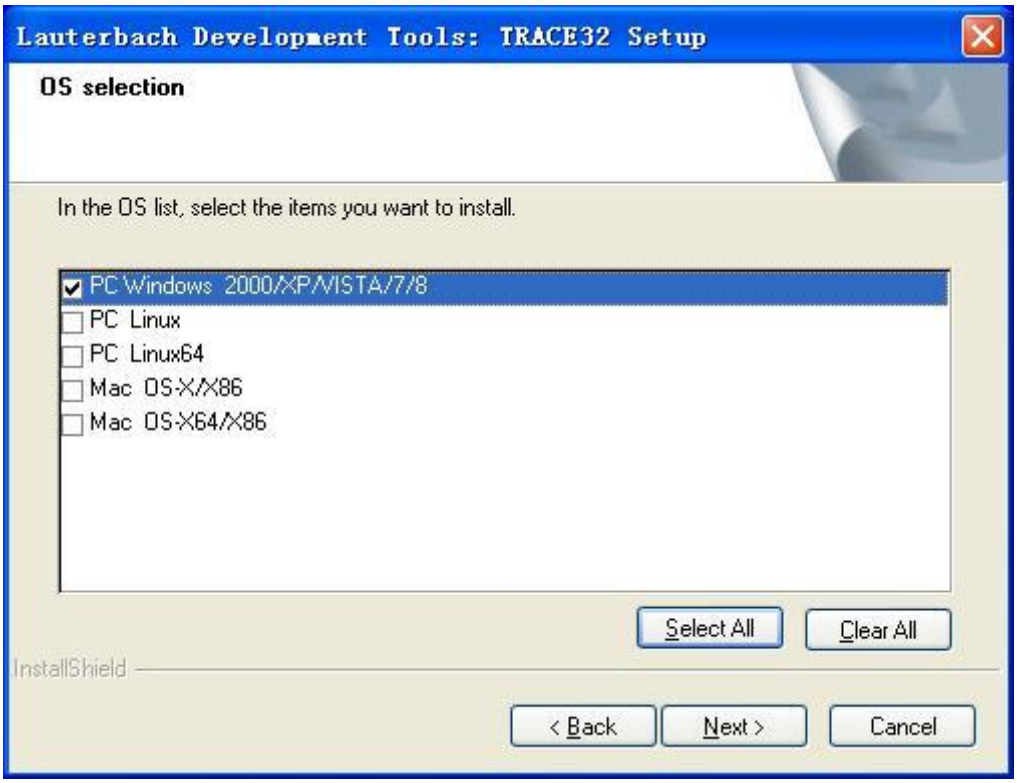
6→（注意是 USB 接口）


Lauterbach Development Tools: TRACE32 Setup
ICD Interface Type
Please select the host interface type you will use:
USB Interface
Ethernet Interface
Parallel Interface
RS232
ISA Card
DAS
CITRIX
InstallShield
< Back
Next >
Cancel

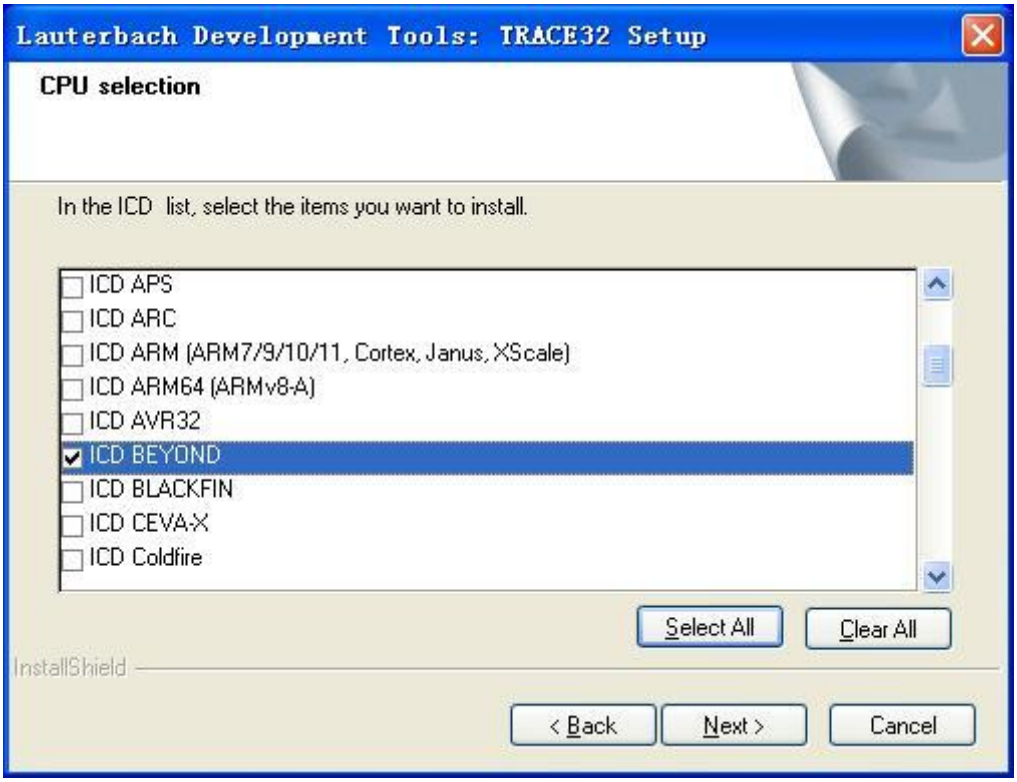
7→


Lauterbach Development Tools
Selection of license for new software
License Key not necessary
-NEW device,
one year license automatically included
-License file is maintained by administrator
-License key in period of validity
(usually one year)
-Valid license key stored in DebugCable
or in NexusAdapter
License Key necessary
-License key supplied with software
-Maintenance contract expired
-Product older than one year and
no new license key was installed
in the last year
InstallShield
< Back
Next>
Cancel

8→



9→（**Beken 处理器版本是 BA22，注意这里选择 ICD BEYOND**）



10→

11→

12→

稍等，**别着急，可能要等好几分钟。后面几步都可以点击 NEXT，**即完成安装

补丁安装

补丁 1

目录地址：\\192.168.0.30\Exchange\wangzhilei\TRACE32_201402\40950120_26-08-14_20140826beken.zip 或 \\192.168.0.30\Documentation\software\wangzhilei\trace32\20140826beken

安装步骤

或解压，或拷贝到用户的安装目录（例如 D:/T32，或 E:/T32），它会覆盖一些文件。补丁的内容包括：\\192.168.0.30\DOCUMENTATION\SOFTWARE\WANGZHILEI\TRACE32\20140826BEKEN

补丁 2

目录地址：\\192.168.0.30\Documentation\software\wangzhilei\trace32\trace32_beyond_r_2018_02_000096069_win64.zip

安装步骤

或解压，或拷贝到用户的安装目录（例如 D:/T32，或 E:/T32），它会覆盖一些文件。

使用

Attention 1---Erase flash

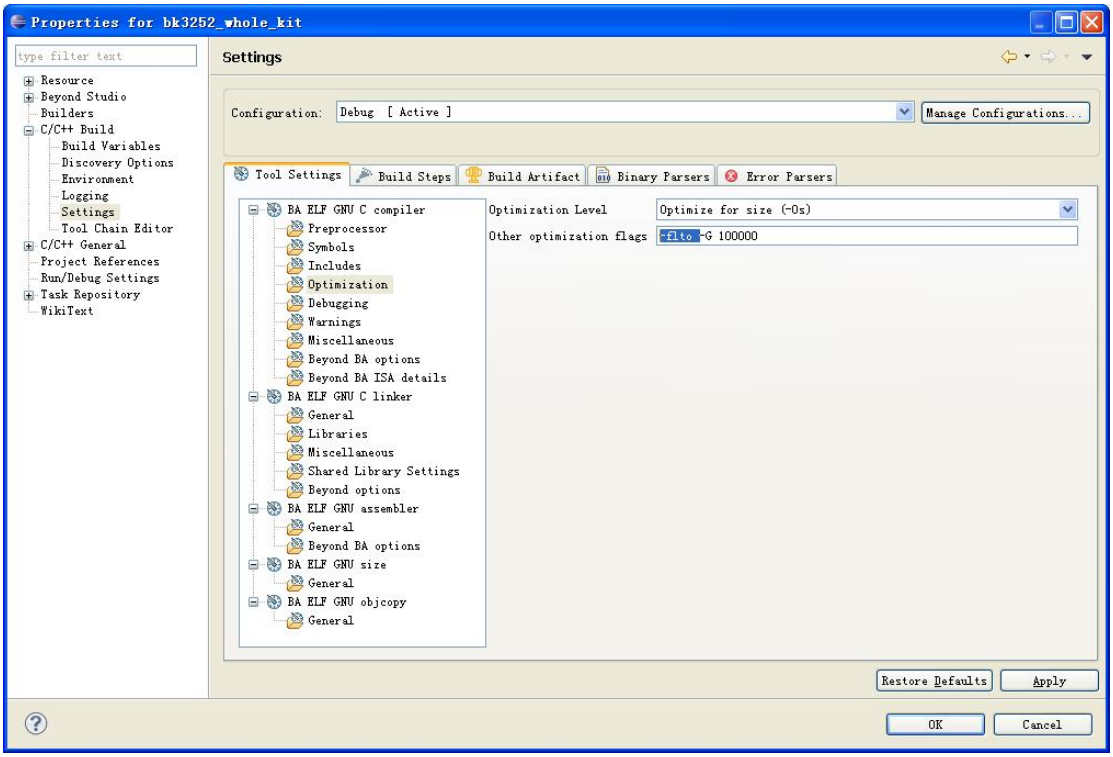
请用户将 BOARD 采用 JTAG CPLD 小板把 Flash 全擦除，或者下载一个肯定不会影响 TRACE32 的映像文件。
原因：BOARD 中残留的固件可能会改变 JTAG Pin 的配置，导致 Trace32，或 JTAG 无法正常连接；
惨痛教训：无法连接时，采用示波器测试 CLK 引脚，发现其幅度只有 100mV 左右。正常 IO 电压应该在 2.8V 左右。这个异常就是因为残留映像文件运行导致 JTAG Pin 脚配置异常。
【pujie yang [pujie@bekencorp.com]补充】
该问题在我们自己的 cpld 下载板中也曾经遇到，后来解决了。该问题的解决办法为：
1，连接前让 trace32 发出一个大于 80ms 的 reset 信号，然后将这个 reset 信号连接到 ba22 板子的 reset 上；
2，Reset 信号发出后让 trace32 循环多次去连接 ba22，直到连接成功。
原理： 1》，芯片 reset 复位后,默认 jtag 信号有效；
2》，芯片复位后到再次运行软件配置 jtag 为 GPIO 中间有一段时间。在这一段时间间隙让 trace32 去连接 ba22 就能成功；
3》，Reset 信号撤销后，ba22 还需要一小段时间才能进入正常工作状态。所以 reset 信号撤销后，trace32 不能一次连接成功，而需要多次尝试。

Attention 2---sleep and jtag pins

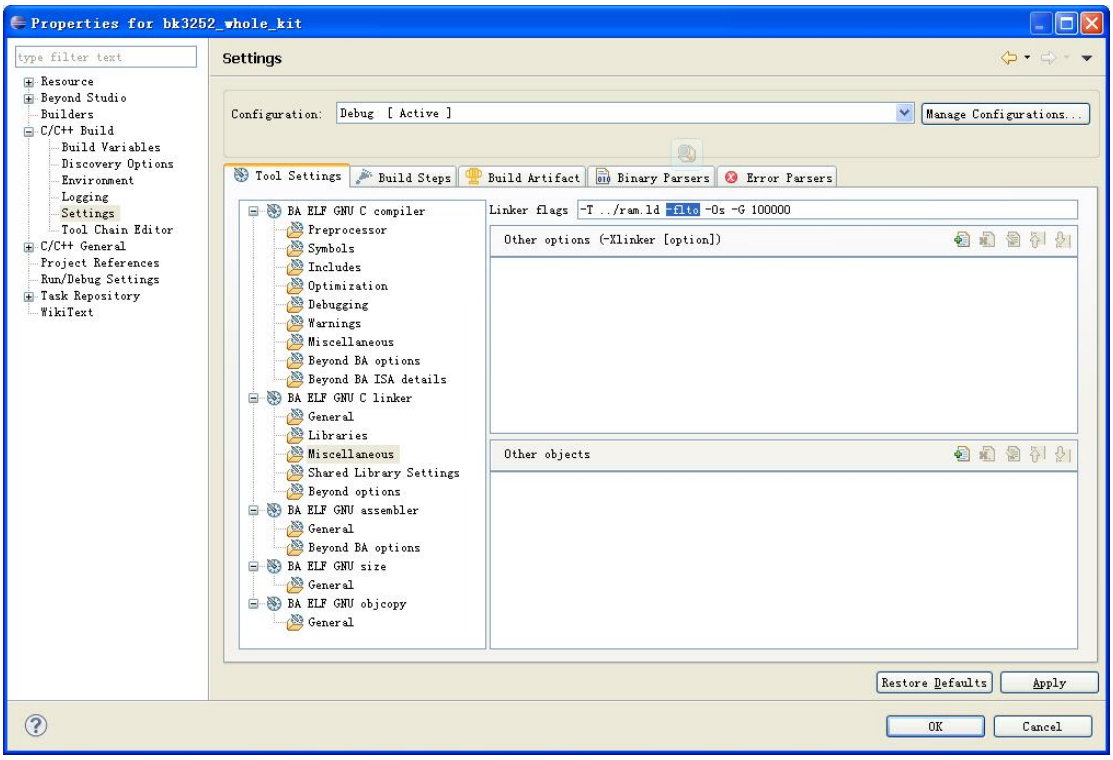
在即将运行，或调试的工程，屏蔽掉睡眠和对 JTAG GPIO 脚的配置。
BK3252：在 svn313 版本以上打开宏---- CONFIG_TRACE32_ENABLE（config.h）

Attention 3---flto

编译选项中去掉-flto，如果加上这个选择，在使用 Trace32 的时候应该看不到符号表
例图 1/2：

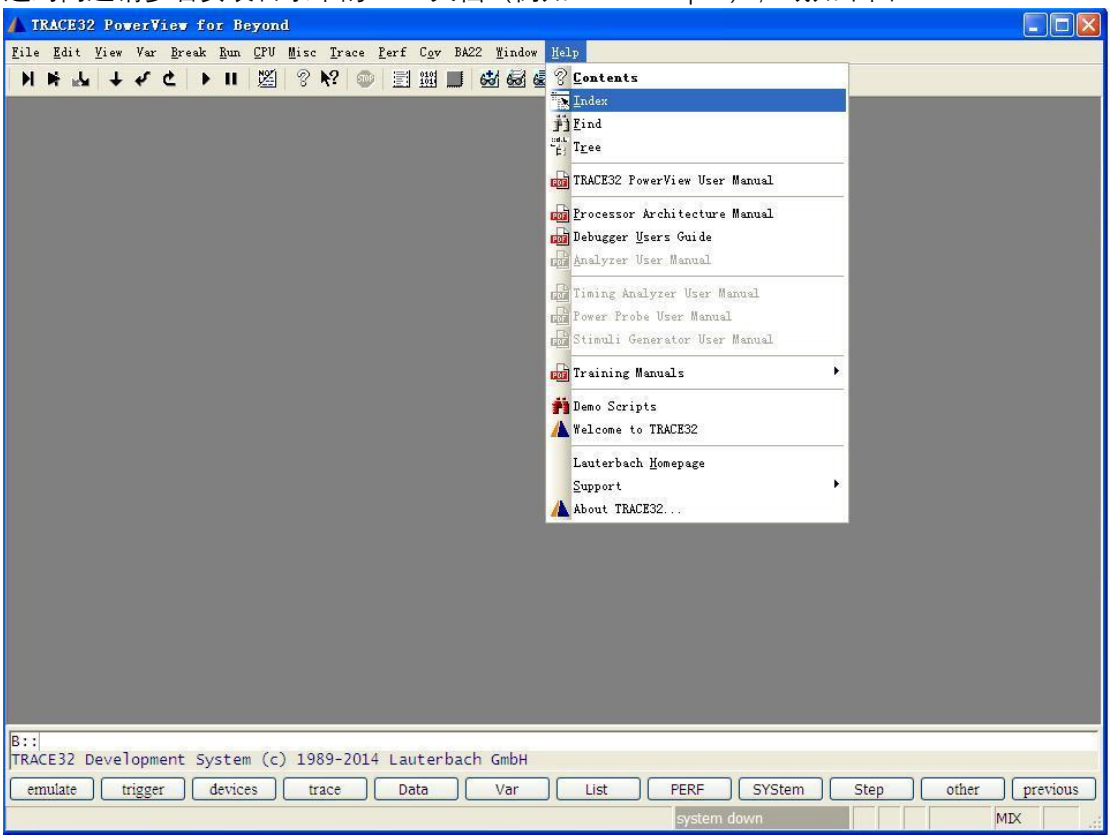


例图 2/2：



Attention 4---help document

遇到问题请多看安装目录中的 PDF 文档（例如：D:/T32/pdf），或如下图：



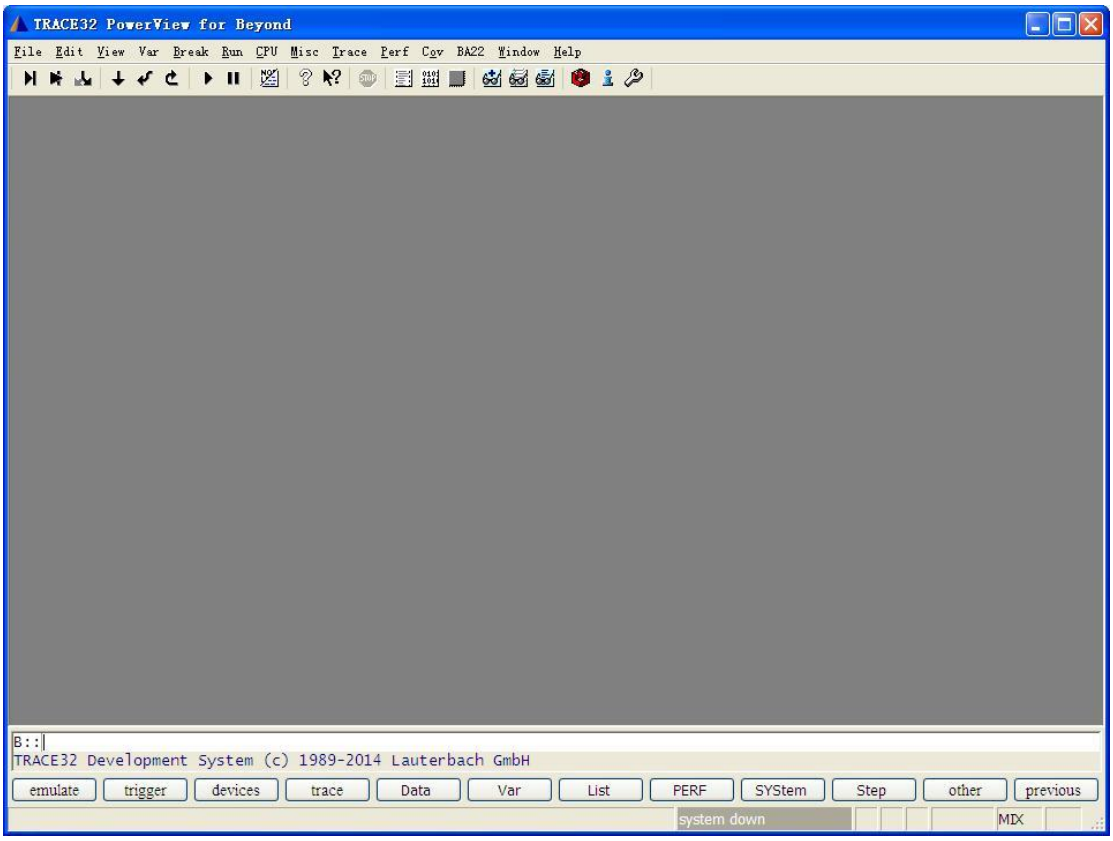
Attention 5---cmm script

注意调试 CMM 脚本，它不是对所有的 BA22 工程都通用。每个 BA22 项目的芯片可能都有一些特殊的配置需要在脚本中实现。譬如 workaround 擦除 Flash 的 bug，关闭 watchdog 等等

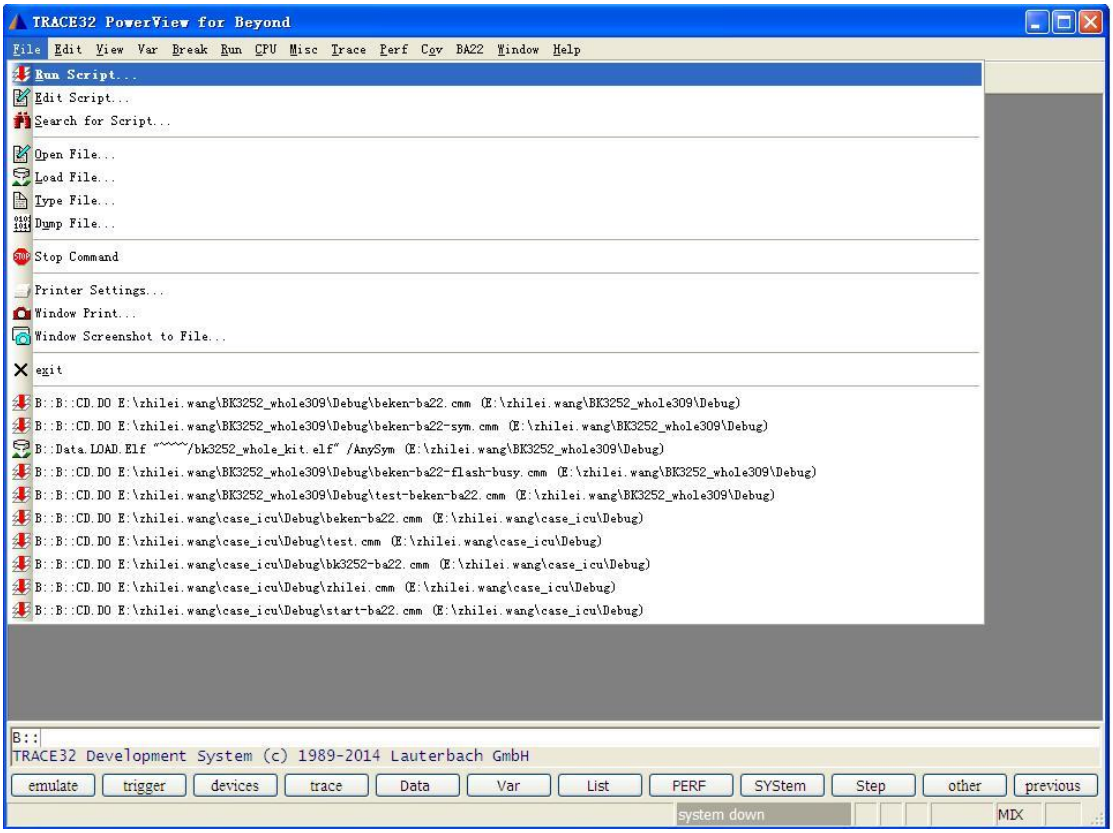
Attention 6---start

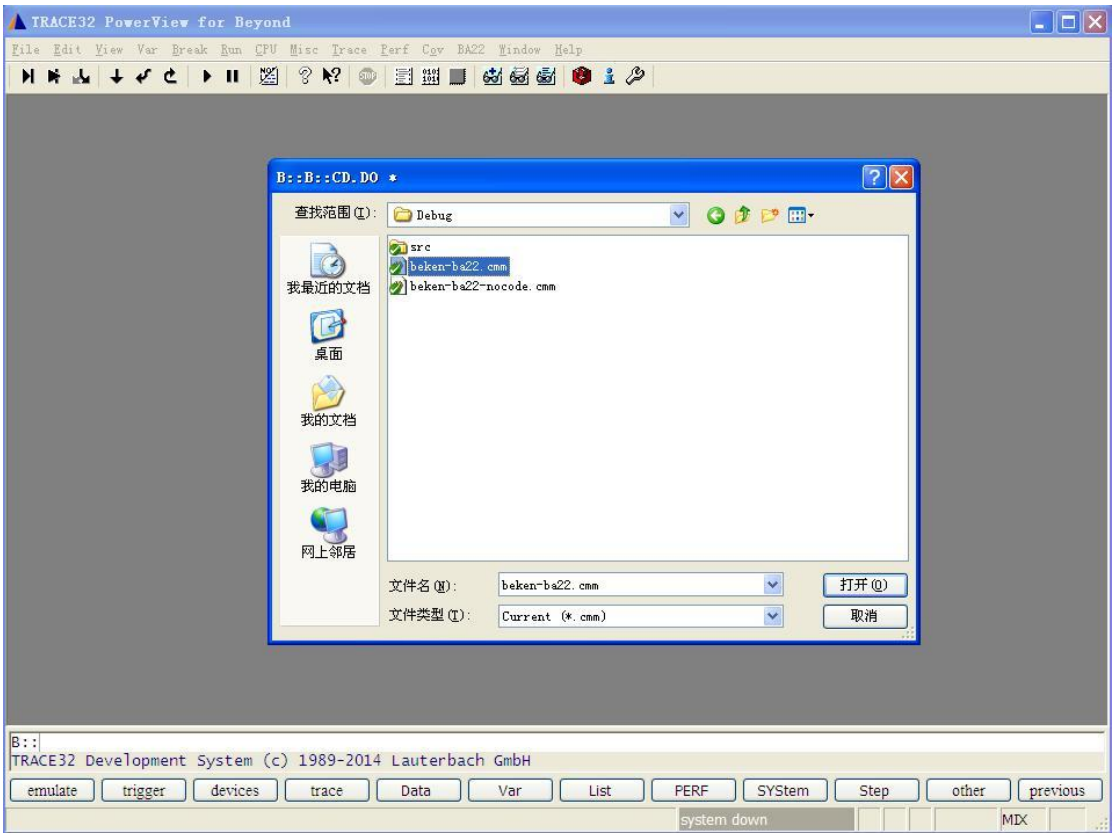
使用 CMM（Trace32）步骤:

打开界面：开始菜单中点击 Trace32 ICD BEYOND USB



加载 CMM 脚本:





Attention 7---change SR’s value

Register.Set SR <value>

如果想在界面上修改 SPR_SR 寄存器的值，在 SPR_SR 值区域右键选择 set 或者 modify（“PER.Set SPR:0x11 %Long 0x821D”），暂时没有办法完成。如果想修改，试试这个命令：Register.Set SR <value>。

Attention 8---write breakpoint at SPR

不支持

官方回复如下：Such a feature is not part of the BA22 specification. If you did extensions to the Core to trigger accesses on the SPR's we can add these for you into TRACE32。

脚本

BK3252—TRACE32—CMM，参见目录 http://192.168.0.8:8080/svn/bk3252/branches/BK3252_whole/Debug/*.CMM
脚本细节如下：

```
; ------------------------------------------------------------------------------
; @Title: Configuration for setting up a BA22 custom core.
; @Copyright: (C) 2013-2114 BEKEN CORP, licensed for use with TRACE32(R) only
; ------------------------------------------------------------------------------

   print ""
   print "------------------------------------------------------------------------------"
   print "Configuration for setting up a BA22 custom core"
   print "------------------------------------------------------------------------------"

   area
   print "Step1: Reset                    "
   RESet
   SYStem.RESet
   SYStem.CPU BA22

   print "Step2: select JTAG Protocol                "
; select JTAG Protocol 34bit FAST or 80bit ORIGINAL
   SYStem.CONFIG.DebugProtocol ORIGINAL

   print "Step3: select Memory Access Module                "
; select Memory Access Module
; SPR mapped CPU module or WISHBONE
   SYStem.CONFIG.MemAccessModule CPU

   SYStem.memaccess DENIED
   SYStem.cpuaccess Enable

   print "Step4: Set Little Endian                "
; Big or Little Endian
   SYStem.Option LittleEnd ON
   SYStem.Option IMASKASM ON
   SYStem.Option IMASKHLL ON

   print "Step5: Set Jtag Clock                "
   SYStem.JtagClock                                    200khz

   print "Step6: Configure Exception Vectors of the Design                    "
; Configure Exception Vectors of the Design
   SYStem.CONFIG.ReSeTException                    0x008
   SYStem.CONFIG.BUSErrorException             0x010
   SYStem.CONFIG.AlignmentException            0x018
   SYStem.CONFIG.IllegalInstrException      0x020
   SYStem.CONFIG.TRAPException                     0x028
   SYStem.CONFIG.TickTimerException            0x030
   SYStem.CONFIG.INTerruptException            0x038
   SYStem.CONFIG.SystemCallException         0x040
   SYStem.CONFIG.FloatingPointException    0x048
   SYStem.CONFIG.InstrPageFaultException 0x060
   SYStem.CONFIG.DataPageFaultException    0x068
```

```
    SYStem.CONFIG.ItlbMissException             0x070
    SYStem.CONFIG.DtlbMissException             0x078

; RESET and connect to BA2x core
    print "Step7: Configure System Reset Duration                    "
    SYStem.Option ResetDURation 85ms
    SYStem.Option WaitReset 20ms /Poll

    print "Step8: System Up                  "
    SYStem.Up

    print "STEP9: Inform the debugger about read only address ranges(0x0, 0x400000)."
    MAP.BOnchip 0x00000000++0x00400000

    print "Step10: Invoke InitializationBK3252 Routine                "
    gosub InitializationBK3252

    print "                                                "
    print "Step11: Load Elf File                  "
    SYStem.JtagClock                                  1MHz
    y.Spath ..\src
    Data.LOAD.Elf "~~~~/bk3252_ott.elf" /AnySym

    ;SYStem.JtagClock                                  200khz
    print "Step12: Over                  "

    ;==Open                                                          some                                                                basic
windows==========================================================================
    print "WINPAGE.RESET                                            "
    TOOLBAR ON
    STATUSBAR ON
    ;FRAMEPOS 17.857,10.417,,,Maximized
    ;WINPAGE.RESET

    ;WINCLEAR
    print "Breakpoint listing                                        "
    WINPOS 99.429 44.083 78. 6. 23. 1. W002
    WINTABS 13. 16. 0. 0. 0. 0. 0.
    Break.List

    ;WINPOS 99.714 56.833 77. 7. 0. 0. W003
    ;AREA.RESet
    ;AREA

    print "CPU registers and stack view                                          "
    WINPOS 0.0 51.75 94. 12. 0. 0. W005
    Register

    print "Data list window with disassembly and/or C/C++ code                                    "
    WINPOS 0.0 0.0 94. 45. 11. 1. W000
    WINTABS 10. 10. 25. 62.
    Data.List

    print "Dump on SRAM data                                        "
    WINPOS 99.0 0.0 78. 11. 13. 1. W004
    Data.dump 0x0 /SpotLight

    ENDDO
```

```
; ------------------------------------------------------------------------------
;                                    BK3252 Initialization
; ------------------------------------------------------------------------------
InitializationBK3252:

   LOCAL &reg &busyFlag

   ;watchdog needs be deactivated
   print "       1, Disable Watchdog"
   D.S AD:0x0F48000 %LE %Long 0x5A4000
   PRINT Data.Long(AD:0x0F48000)

   D.S AD:0x0F48000 %LE %Long 0xA54000
   PRINT Data.Long(AD:0x0F48000)

   print "       2, Init BK3252 Analog Register"
   D.S AD:0x00f40058 %LE %Long 0xf8000000
   wait 5.ms
   D.S AD:0x00f4005c %LE %Long 0xf8000000
   wait 5.ms
   D.S AD:0x00f40060 %LE %Long 0xf8000000
   wait 5.ms
   D.S AD:0x00f40064 %LE %Long 0x0000ffff
   wait 5.ms
   D.S AD:0x00f40068 %LE %Long 0x0000ffff
   wait 5.ms
   D.S AD:0x00f4006c %LE %Long 0xf8000000
   wait 5.ms
   D.S AD:0x00f40070 %LE %Long 0xf8000000
   wait 5.ms
   D.S AD:0x00f40074 %LE %Long 0xf8000000
   wait 5.ms

   D.S AD:0x00f40058 %LE %Long 0x90881FC4
   wait 5.ms
   D.S AD:0x00f4005c %LE %Long 0x38B6BE9F
   wait 5.ms
   D.S AD:0x00f40060 %LE %Long 0x10001039
   wait 5.ms

   PRINT Data.Long(AD:0x00f40070)
   &reg=Data.Long(AD:0x00f40070)
   &reg=&reg+0xC00000
   D.S AD:0x00f40070 %LE %Long &reg
   wait 5.ms
   PRINT Data.Long(AD:0x00f40070)

   PRINT Data.Long(AD:0x00f40064)
   &reg=Data.Long(AD:0x00f40064)
   &reg=&reg&(~0x0000C004)
   D.S AD:0x00f40064 %LE %Long &reg
   wait 5.ms
   PRINT Data.Long(AD:0x00f40064)

   ; BK3252_Ana_PLL_enable
   &reg=Data.Long(AD:0x00f40068)
```

```
&reg=&reg&(~0x00001a)
D.S AD:0x00f40068 %LE %Long &reg
wait 5.ms

&reg=Data.Long(AD:0x00f40058)
&reg=&reg+(0x02000000)
D.S AD:0x00f40058 %LE %Long &reg
wait 5.ms

&reg=Data.Long(AD:0x00f40058)
&reg=&reg&(~0x02000000)
D.S AD:0x00f40058 %LE %Long &reg
wait 5.ms

&reg=Data.Long(AD:0x00f40070)
&reg=&reg&(~0x001C)
D.S AD:0x00f40070 %LE %Long &reg
wait 5.ms

print "       3, Set Low Speed of Flash"
D.S AD:0x0080100c %LE %Long 0x00000000
PRINT Data.Long(AD:0x0008)
PRINT Data.Long(AD:0x0200)
PRINT Data.Long(AD:0x0080100c)

print "       4, Select 26MHz of System"
D.S AD:0x00800000 %LE %Long 0x00000100
PRINT Data.Long(AD:0x0080)
PRINT Data.Long(AD:0x0300)
PRINT Data.Long(AD:0x00800000)

print "       5, Select Clock of Flash"
D.S AD:0x80101c %LE %Long 0x000000F
PRINT Data.Long(AD:0x0010)
PRINT Data.Long(AD:0x02A0)
PRINT Data.Long(AD:0x80101c)

print "       6, Write 0 into Status Register of Flash"
D.S AD:0x801010 %LE %Long 0x0001
PRINT Data.Long(AD:0x801010)

D.S AD:0x801018 %LE %Long 0x0002
PRINT Data.Long(AD:0x801018)

D.S AD:0x801014 %LE %Long 0x0000
PRINT Data.Long(AD:0x801014)

D.S AD:0x801004 %LE %Long 0x0004
PRINT Data.Long(AD:0x801004)

D.S AD:0x801008 %LE %Long 0x0001
PRINT Data.Long(AD:0x801008)

print "       7, Erase Flash"
D.S AD:0x801004 %LE %Long 0x0010
PRINT Data.Long(AD:0x801004)

D.S AD:0x801008 %LE %Long 0x0001
```

```
   PRINT Data.Long(AD:0x801008)

   PRINT "Waiting for erasing flash..."

   &busyFlag=1
   while &busyFlag!=0
   (
      ; use blocking or non-blocking method
      ;GOSUB FlashIsBusyBK3252_Blocking
      GOSUB FlashIsBusyBK3252_NonBlocking
      ENTRY &busyFlag
      &busyFlag=&busyFlag&0x1
      if (&busyFlag==0x1)
      (
         PRINT "waiting for erasing flash..."
      )
   )

   PRINT "Erase flash successfully"

   RETURN

; ---------------------------------------------------------------------------------
;                                    BK3252 Check Flash Busy Bit
; ---------------------------------------------------------------------------------
FlashIsBusyBK3252_Blocking:
   LOCAL &temp
   &temp=0x1
   ON ERROR GOTO _retry
_retry:
   WAIT 0.1s
   PRINT "This is a dummy read to check if flash is ready"
   &temp=Data.Long(AD:0x801020)
   ON ERROR inherit
   RETURN &temp

FlashIsBusyBK3252_NonBlocking:
   LOCAL &temp
   &temp=0x1
   ON ERROR CONTinue
   PRINT "This is a dummy read to check if flash is ready"
   &temp=Data.Long(AD:0x801020)
   ON ERROR inherit
   RETURN &temp

; ---------------------------------------------------------------------------------
;                                    BK3256 Initialization
; ---------------------------------------------------------------------------------
InitializationBK3256:
      ;TODO
   RETURN

; ---------------------------------------------------------------------------------
;                                    BK3260 Initialization
; ---------------------------------------------------------------------------------
InitializationBK3260:
      ;TODO
   RETURN
```

Attention 1---reset duration

一般芯片 JTAG 接口有两个 Reset Pin：system reset，test reset。通常 system reset 只有 5—10ms 左右。在我们 BK3252 上面为了降低成本，减少 Pin 脚，舍弃了 TRST Pin 脚，只有 SRST，并且要求 SRST 信号宽带必须大于 80ms。安装软件中的补丁就是为了满足这个条件，德国工程师重新编译个特殊版本给 Beken Corp。

Attention 2---bus busy issue

在启动 Flash 擦除之后必须要等一段时间，Trace32 才能 Data.Load.ELF 文件。在 BK3252 硬件里面不停的查询 Flash 的状态寄存器，AHB 总线处于繁忙状态。这个时候，Trace32 JTAG 接口也处于繁忙状态。如果在 Trace32 侧读取 Flash 地址，或寄存器，就会出现异常：Bus Error Generated By CPU。在文档 Beyond_JTAG_TAP_Processor_Debug_Operation_Manual.pdf page10—11 实际有 READ DATA BUS 的命令来查询 JTAG 总线是否繁忙。在现阶段 CMM 脚本中通过读取寄存器，直到它正常来确认总线闲置。

BACKUP 1---little endian

BA22 可以实现大端，或小端。在 BA22 的文档中很多软件仿真都是采用大端，不过在 Beken 的现实中都是采用小端。

BACKUP 2---jtag protocol

JTAG 协议采用的是 80bit 的 ORIGINAL 方式，即低速方式。细节可以咨询 Beken 硬件工程师：pujie yang [pujie@bekencorp.com]

BACKUP 3---memory access module

BK3252 之 BA22 版本为 5.2，存储器访问模块是 CPU 方式，其访问效率比较低。一般在 40%左右，如果设置 JTAG 时钟为 10MHz，其访问速率只有 500KBps。下一代 BA2X 可能会采用 wishbone 总线访问方式，效率会提高 94%。

沟通纪要

1． 我们使用的 CPU core 版本为：BA22 V5.2.2，这个可能是 BA22 目前发布的最新版本了，请确认下 TRACE32 对该型号的支持情况。
支持 BA22，TRACE32 不区分内核版本。

2． 那个 USB3.0 的 debugger 对 USB2.0 的电脑兼容情况怎么样？ 我们的调试电脑都是 USB2.0 接口的，winxp 系统。
没问题，可以兼容。

3．Trace 功能不可以通过 jtag 实现，需要被测 CPU 芯片引出专用的 trace 接口？是这样吗？
我的理解如下：
Jtag 接口：只需要下图中左边一列的信号：TCK，TMS，TDI，TDO，TRST
Trace 接口：需要下图中的所有信号。也就是说不但需要 jtag 接口的信号，还需要右边的 DATA[0] ~DATA[7]

使用 trace 功能，trace 接口必须引出。如果 trace 接口没引出，买带 trace 功能的调试器也没意义，买简单的版本就行。

4. trace32 simulation 仿真问题
编译链接时加参数 -flto 后，trace32 simulation 时不能显示出相应的 C 语言，只能显示汇编。 如果去掉参数-flto 后，C 语言和汇编可以正常混合显示出来，然而 code_size 会增大一倍。 我们的芯片大多是量身定制的，code ram 余量不多。
去掉-flto 参数后，code_size 会变大很多，我们的芯片 RAM 容量可能就放不下代码了，没办法调试，然而如果添加-flto 参数，trace32 又不能显示出相应的 C 语言，这个问题可以解决吗？

GNU GCC 的问题，需寻使用者找其他优化方法。Debugger 工具厂商对该问题的支持有限。

5.通常的芯片会有两个 reset 信号，一个是 SYS_RST(system rest),另一个是 jtag 接口上的 jtag_rst.为了减少芯片封装的 pin,我们去掉了 jtag_rst，只保留了 SYS_RST。使用我们以前的工具是可以正常下载和调试 BA22 的。 请问 trace32 是否能支持这种情况？

可以支持，两个 RST 信号对 debugger 来说是可选项。

6.我们的芯片 rest 信号要求持续时间长度大于 80ms(80 毫秒),请问 trace32 可以支持自定义 reset 信号的时间长度吗？我们这样设计的目的是为了提高芯片的稳定度。毛刺，静电，上电电压不稳等都不会触发误复位。其他公司可能不这么用。

可以支持，reset 信号长度可配置。

7.我们的芯片可以外挂指令 spi_flash,当通过 jtag 接口对外挂指令 spi_flash 下载代码时，由于 flash 自己的特点，写速度会比较慢。每写一个 word 后，jtag 都需要插入一些等待时钟（或者 debugger 通过 jtag 读 ba22 的 busy 状态来实现。在写 flash 的过程中 AHB_ready 会被拉低。）trace32 可以支持这种情形吧？
待确认
有可能引起 debugger timeout，发一个写 flash 的 AHB 时序图进一步确认。
时序图请参考附录。
【Beken：Beken Corp 设计了自己 Flash Controller，在 AHB 角度看可以把 SPI Flash 理解为 RAM】

8.我们做的是低功耗芯片，芯片主时钟频率范围是 32Khz~120MHz. jtag 时钟频率最高可以支持到 30MHz.请问在芯片主时钟为 32KHz 或 120MHz 时，trace32 是否都能通过 jtag 正常下载？我们以前的工具是在 BA22 芯片主时钟为 10MHz~120MHz 时工作比较稳定。当 BA22 芯片主时钟为 32KHz~100Kz 时不好下载，后来 beyond 帮我们更新了下载工具才可以在这种情形下实现下载调试。请问 trace32 对主芯片的运行时钟频率有要求吗？

A:Debugger 支持发出 reset 信号命令。
B: Debugger 支持配置芯片寄存器命令。
C: Debugger 支持动态配置 jtag 速率命令。
使用以上 ABC 三个命令的配合应该可以解决这个问题。

现存问题

有时候第一次上电，运行脚本无法下载 Image 成功

原因需要继续跟踪

正常运行 CMM 脚本，偶现擦除 FLASH 完成之后 Bus Error Generated By CPU

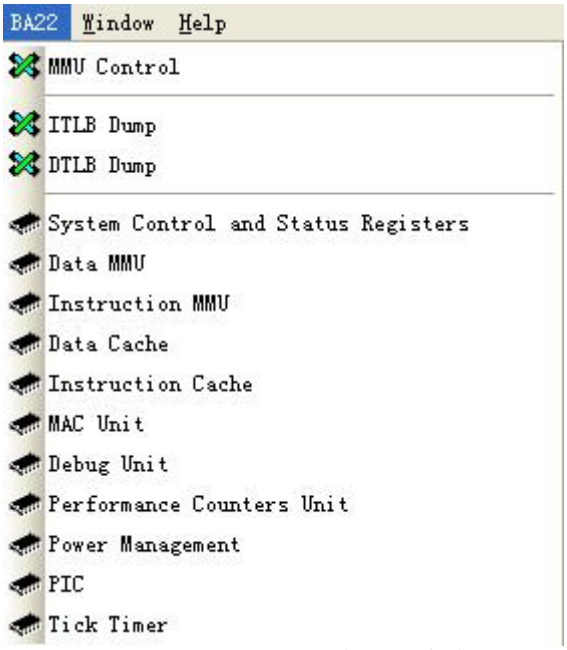
原因需要继续追踪

调试方法集

第一次使用，如何开始？

A，PC 软件准备：安装 Trace32 Beyond 版本，打 patch；
B，调试工程准备：打开工程中对应的宏，参见 Attention 2---sleep and jtag pins；去掉一些优化选项利于调试，参见 Attention 3---flto；参见使用；
C，硬件准备：连接 Trace32 USB 线、电源线、JTAG 接口；
D，点击 Trace32 菜单 File→Run Script，参见 Attention 6---start，选择正确的 CMM 脚本文件；

点击菜单 BA22 里面子菜单项（如下图），怎么没有实现？看不到寄存器的值？



A，Beyond Trace32 自带光盘中没有实现这个部分，需要更新文件 perba22.per，文件所在位置：\\192.168.0.30\Exchange\wangzhilei\TRACE32_201402；
B，拷贝对应的正确的 perba22.per，覆盖在安装目录下，例如 E:\T32；

为什么 LA3500 Trace32 停下来，发现有 undef 指令?

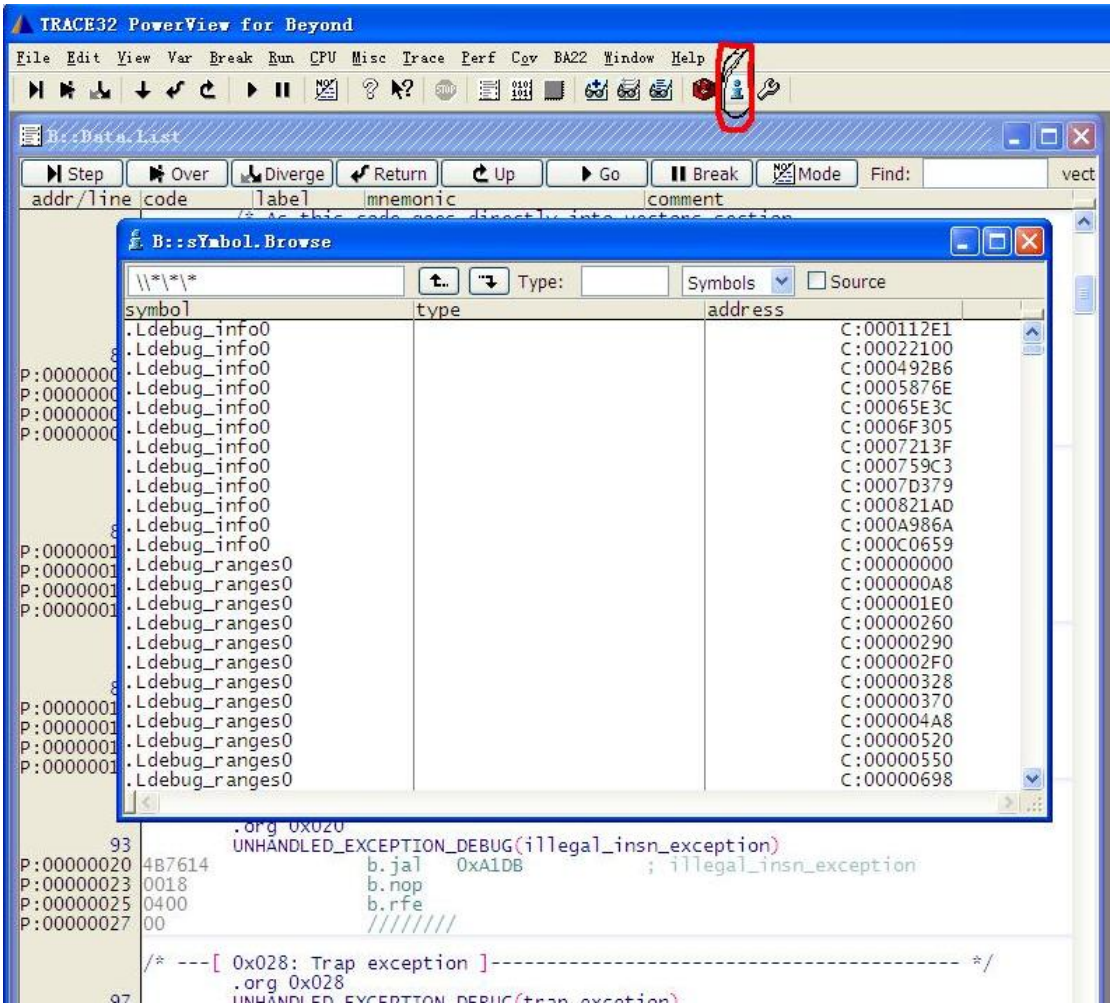
A，先观察当前的 bitcode，机器语言码，是否和 DUMP 文件中同一地址的内容一致；BA22 是变长指令，Trace32 在显示上有时候会有一些问题；
B，观察代码附近是否有“.org”，Trace32 没有关注 org 信息，它会按照顺序去解析指令，org 会重新指定指令位置；
C，可能确实有 undef 的异常，具体问题具体分析吧；

为什么运行起来不能打断点?

需要在你运行的 CMM 脚本文件添加或更改成如下一句：
SYStem.CpuAccess Enable

Trace32 SOC 运行起来，想找某个函数符号表，然后设置断点，通过菜单项 Var—>Show Function，出现异常 access timeout，target running

A，第一次需要先点击 View→Symbols→Browse，或者点击下图中的蓝色 i 型图标



C，然后可以随意使用 Var→Show Function；
D，随意在函数内打断点吧；

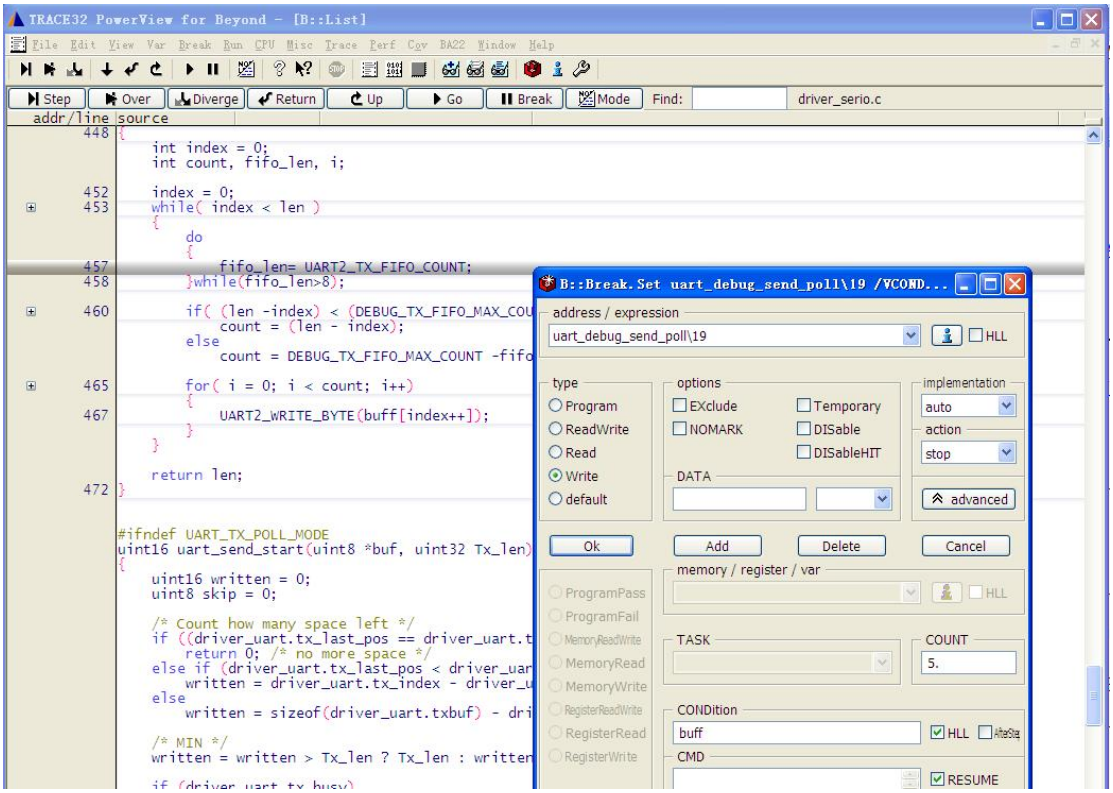
为什么 BK3260 不能使用 Beyond Trace32?

请关注 CMM 脚本细节，需要更改 CMM 脚本。笔者最初完成 BK3252 CMM 脚本的时候参考了：

a， 安装目录的 sample 脚本文件，例如 E:\T32\demo\beyond\hardware\ba22\custom\start-ba22.cmm；
b， svn 目录的 BK3252 的初始化流程，地址在：
http://192.168.0.8:8080/svn/temp/branches/yujun/JTAG_DOWNLOAD/JTAG_SOC_BK3252
c， 跟德国工程师沟通；

如果要 BK3260 可以使用 Beyond Trace32，只需要更改 http://192.168.0.8:8080/svn/bk3252/branches/BK3252_whole/Debug/beken-ba22.cmm 脚本文件的最后一个初始化例程。

想知道某个全局变量在哪里被改写了？在哪里第 n 次被改写？



例如上图
A， 鼠标选中变量，右键选中 advanced breakpoint；
B， 出现上图界面，选择 Write 类型；这种就是读写断点；
C， 设置 COUNT 编辑框，写上 n，例如上图 5。这就表示变量 buff 被写 5 次之后 Break；

Break 之后单步调试，timer 中断太多，怎么关掉中断？

第一反应是关掉中断使能，如果操作 BA22 VIC mask 寄存器，可惜大多数时候不好用，原因未知；有效的处理是在脚本中添加：
SYStem.Option IMASKASM ON
SYStem.Option IMASKHLL ON
如果使能 SYStem.Option IMASKASM 选项，在汇编语言单步调试的时候 LA3500 会把全局中断使能位清除；如果使能 SYStem.Option IMASKHLL 选项，在高级语言单步调试的时候 LA3500 会把全局中断使能位清除；

SOC 已经运行一段时间，可能出错，也可能正常，怎样将 TRACE32 attach 上去看看现场？

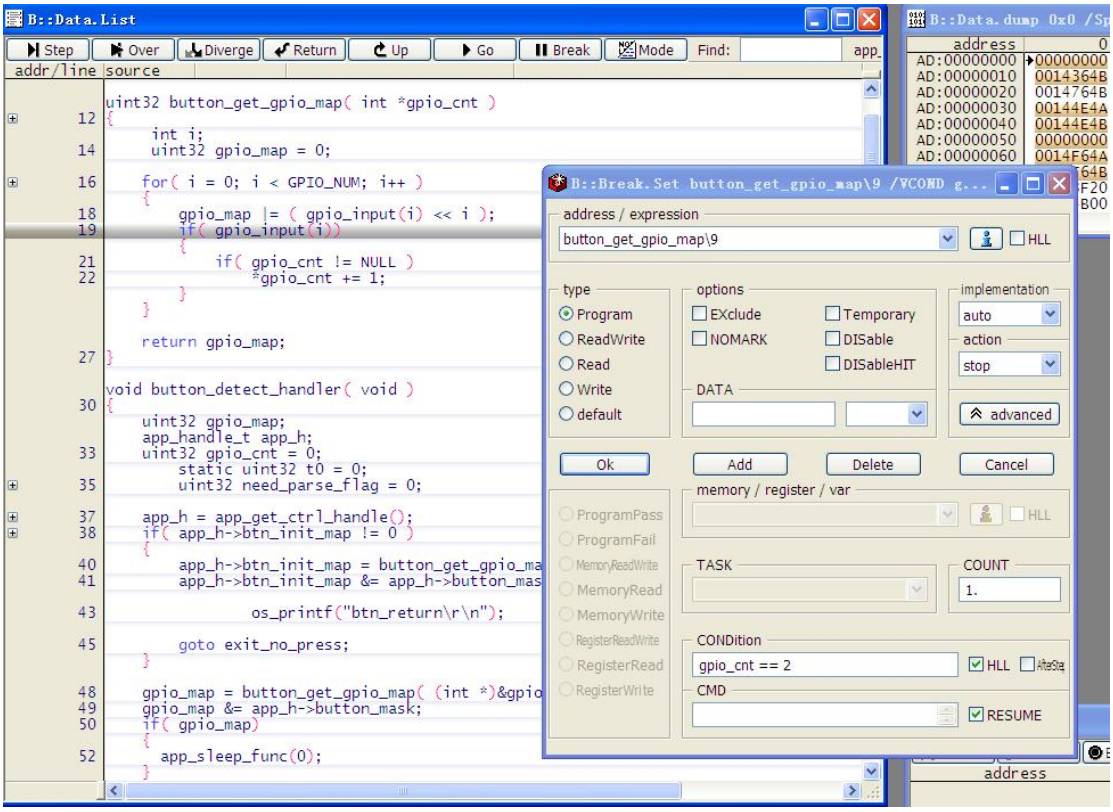
A， 配置 JTAG Protocol：SYStem.CONFIG.DebugProtocol ORIGINAL；
B， 选择 Memory 接口：SYStem.CONFIG.MemAccessModule CPU；
C， Attach：System.Mode.Attach；

D， Break；
E， 恭喜你，可以分析现场了；

下载过 elf 文件，如何在运行之后 dump 出 Flash 里面的内容？

Data.Save.Binary filename.bin start_address—end_addres；文件名可以指定路径，默认在 CMM 所在路径；起始地址中是两个中划线；

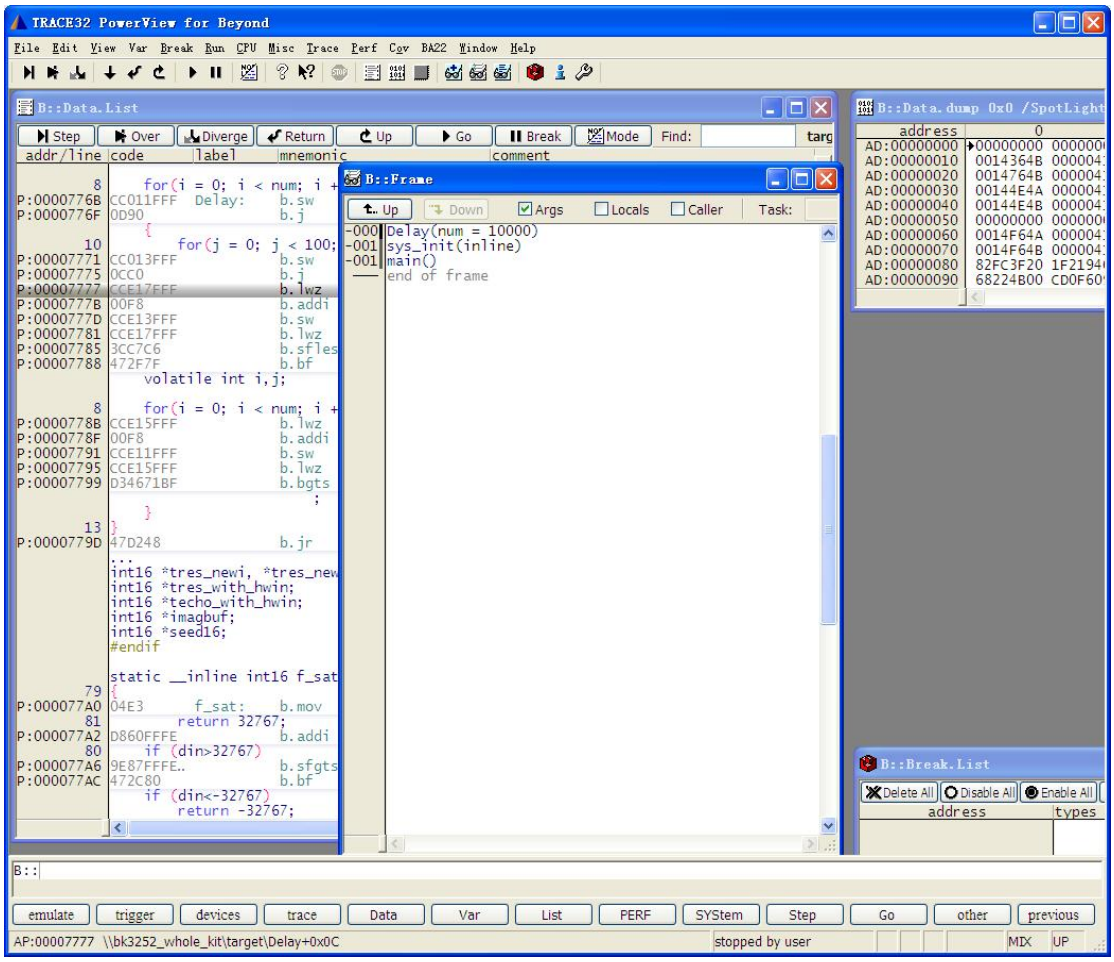
如何设置条件断点



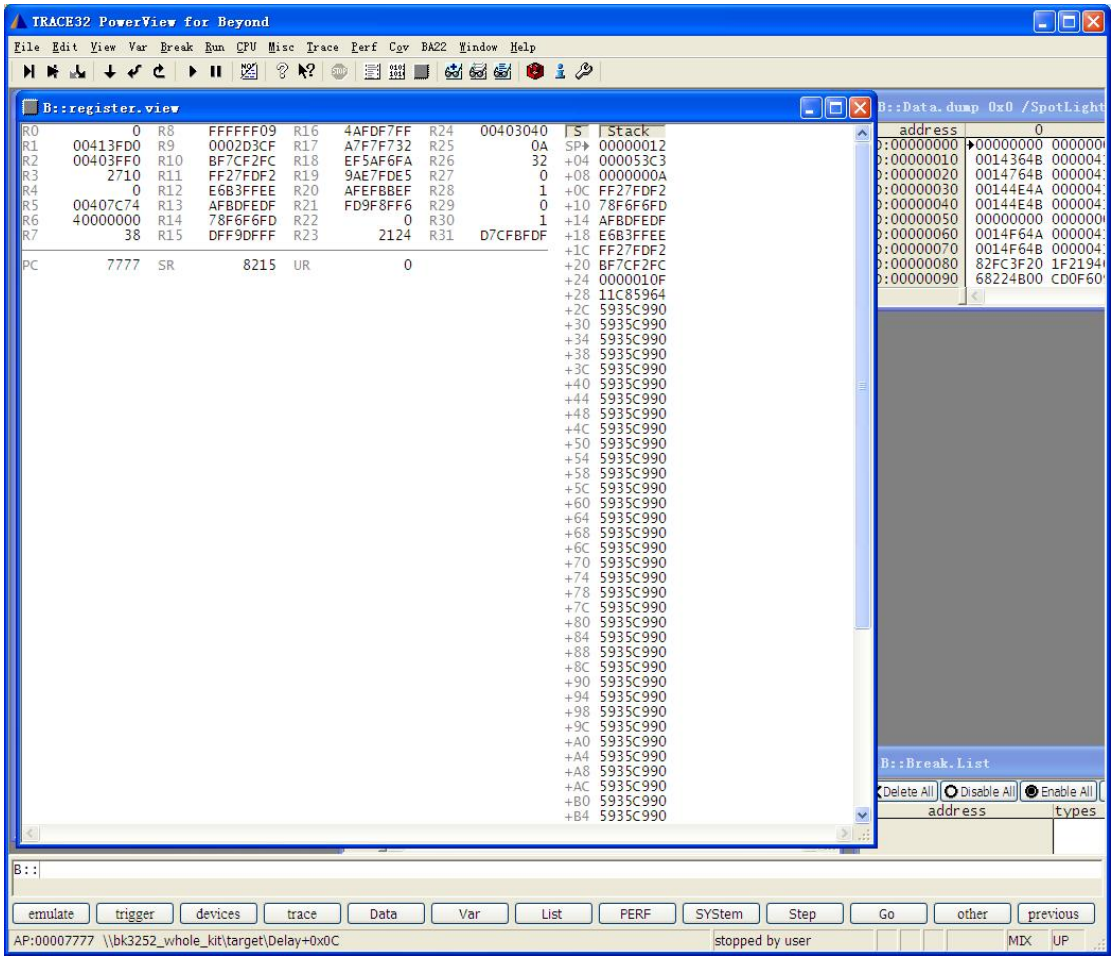
条件关键在于条件表达式的设置，如果选择 hll 选项，表示可以用 high level language 去完成表达式；如果不选择这个 hll 选项，你就能用绝对地址或者寄存器去完成你的表达式；
A， 光标选择函数中的变量，有时候无法选中，可能这个变量已经被优化掉了；
B， 编辑 CONDtion 编辑框，例如上图：希望 gpio_cnt == 2 的时候 Break；
C， 确定就好；

浅析异常现场

A， View→StackFrame，查看函数调用栈；例如下图：



B， Register.View，查看当前寄存器内容，关注 PC、R9、Stack 等，如下图：



C，根据上述两个部分信息，结合 ELF 文件对应的 dump 文件去分析异常的发生原因；

Alignment Exception 异常之后会进入异常处理例程，LA3500 能在出错的地方停下来吗?

当然可以，这样需要你在跑程序之前运行：TrOnChip.Set AE On;这里的 AE 就是 Alignment Exception，它还支持 Reset Exception 等，Trap Exception 就没有了，因为 Core 会在这个异常处主动停下来，它走不下去了。参见 T32/pdf

技术支持

德国

工程师邮件地址: alexander.merkle@lauterbach.com
公司地址：Lauterbach GmbH, Altlaufstraße 40 85635 Hoehenkirchen-Siegertsbrunn Germany
网站地址：http://www.lauterbach.com
电话：+49 8102 9876-147
传真：+49 8102 9876-8147

中国

工程师邮件地址: frank.xing@lauterbach.com, able.zhou@lauterbach.com
公司地址：劳特巴赫(苏州)技术有限公司，星海街 188 号，恒宇广场 709 室，中国苏州
电话：0512-62658030-105，13706215074

私有文档声明

仅限博通集成电路（上海）股份有限公司同事参阅，有问题请联系：zhilei.wang@bekencorp.com。谢谢在整理文档过程中帮助过的同仁们，再次感谢