



Coolwatcher 用户手册

Version1.1.b

This Document contains proprietary information which is the property of Coolsand Technologies, Inc. and is strictly confidential and shall not be disclosed to others in whole or in part, reproduced, copies, or used as basis for design, manufacturing or sale of apparatus with not the written permission of Coolsand Technologies, Inc..

目录

1 概述	4
2 环境配置	4
3 安装与设置	4
3.1 启动 coolwatcher	5
3.2 linux 下的使用 (Ubuntu16.04)	7
4 功能说明	8
4.1 烧写 Flash	8
4.2 查看 buffer 信息	9
4.3 寄存器查看	10
4.4 查看 trace	10
4.4.1 启动 tracer	10
4.4.2. Tracer 菜单项	11
4.4.3. Tracer 应用步骤	12
4.5 GDB Launcher	14
4.5.1 启动 GDB	14
4.5.2 GDB 常用命令	16
4.6 Buffer Profile	16
4.6.1 抓取 prf 文件	16
4.6.2 分析 prf 文件	18
4.7 离线分析	20
4.7.1 生成 *.core(*.elf)	20
4.7.2 建立离线分析环境	20
4.7.3 离线分析 gdb	22
4.7.4 Elf Data Check	23
4.8 Heap Report	25

4.9 Register Viewer	27
4.10 芯片控制	28
4.10.1 关闭芯片	28
4.10.2 重新启动芯片	28
4.11 命令行操作	28
4.11.1 端口操作	29
4.11.2 Flash 编程	29
4.11.3 读 Flash	29
4.11.4 写 Flash	30
4.11.5 其他命令	30
4.12 其他功能	31
4.12.1 Kill 当前运行的程序	31
4.12.2 Kill 所有运行的程序	31
4.12.3 清除脚本输出信息	31
4.13 Linux 串口配置	31
5 疑难解析	31
6 附录	31

1 概述

本文详细介绍了 Coolwatcher 的基本功能与使用说明，包括硬件配置、软件配置、使用步骤、故障处理等。

2 环境配置

本节详细描述了在使用本工具进行手机应用开发中必须的软件环境和硬件设备。

1) 可支持的 PC 系统版本

Windows XP/7/10

3 安装与设置

双击打开 coolwatcher.exe 的配置界面，如图 1 所示。左侧为模块/手机型号选择区，右侧为配置项。

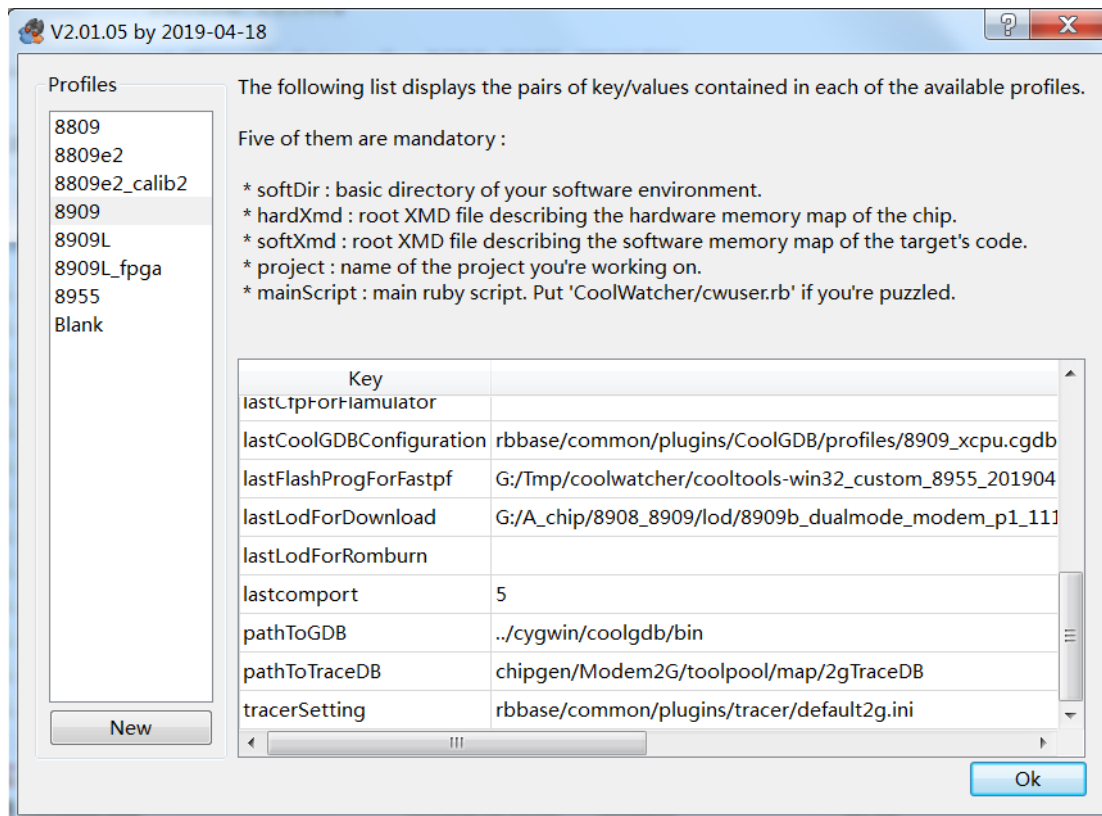


图 1 coolwatcher 配置界面

3.1 启动 coolwatcher

1. 选择手机/模块类型；
2. 修改必要的配置项：如端口号；

单机“OK”键，“coolhost”配置界面将会被打开，界面如图 2 所示。选择相应的 COM 口，之后单击“connect”。

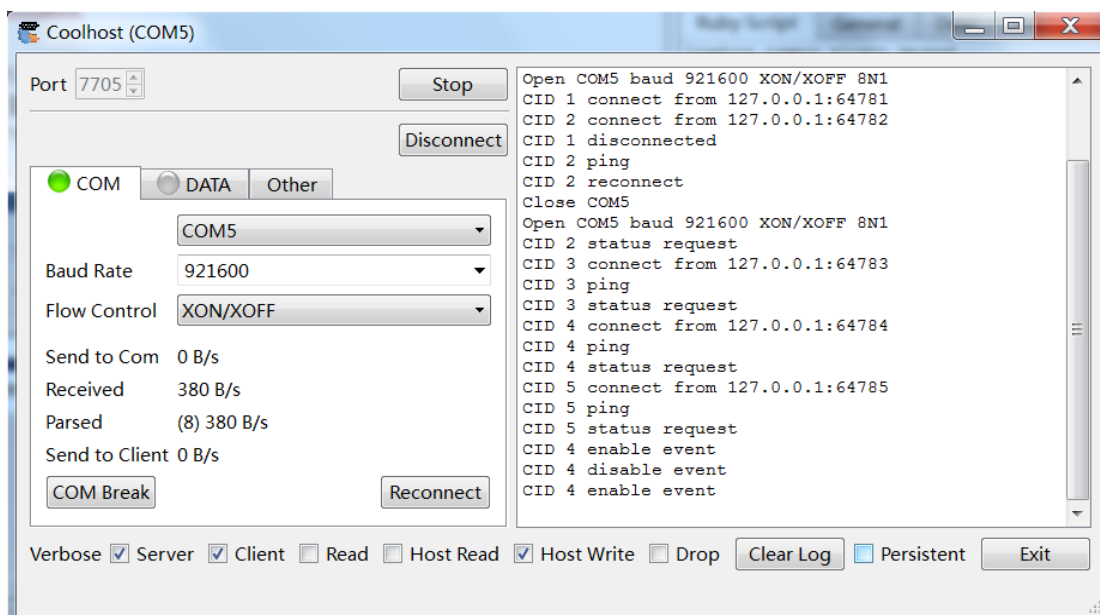


图 2 coolhost 界面

端口号可通过“计算机->管理->设备管理器->端口”进行查看。

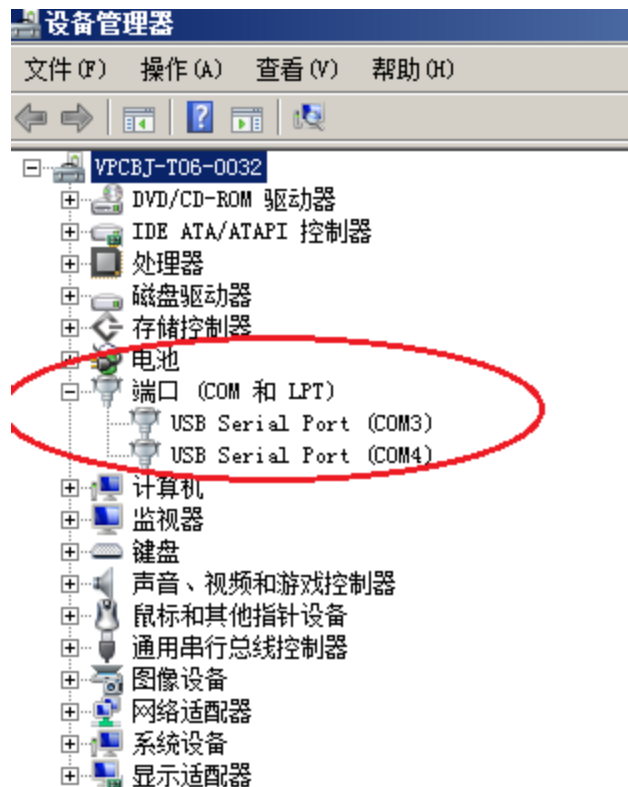


图 3 查看端口

3. Coolwatcher 主界面

Coolwatcher 主界面如图 4 所示。

主界面包括：菜单栏、工具栏、log 打印区、HW Library 显示树、SW Library 显示树，寄存器查看区，Buffer 读取区等功能区域。

Coolwatcher 工具可实现烧录 flash、检查 buffer 信息、读取寄存器、查看 trace 以及 dump 数据等功能。

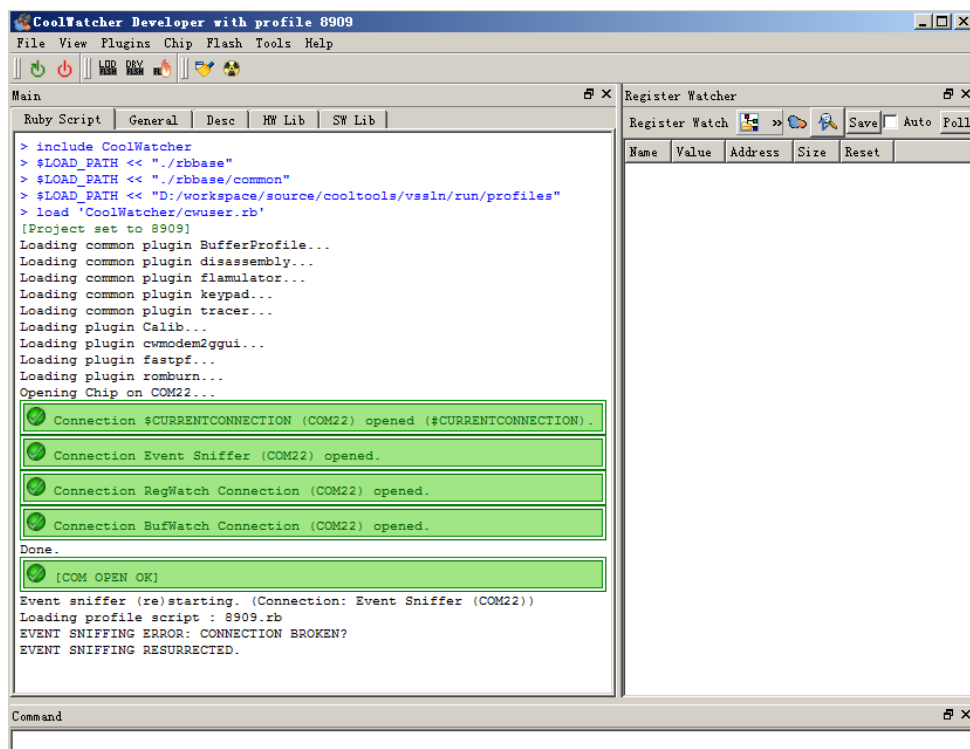


图 4 coolwatcher 主界面

3.2 linux 下的使用（Ubuntu16.04）

1. 依赖包安装

用 `sudo apt install` 命令逐个安装 `build-essential`、`libqt4-qt3support`、`itcl3`、`itk3`、`iwidgets4`

2. 99-coolsand-dongle.rules

把 `99-coolsand-dongle.rules` 文件放到 `/etc/udev/rules.d/`路径中

3. 端口配置

在应用程序 `coolwatcher.exe` 所在文件夹中建立子文件夹 `comport`。

为 `/dev/ttyUSB0`、`/dev/ttyUSB1` 创建符号连接文件

如 `ln -s /dev/ttyUSB0 comport/COM1`

`ln -s /dev/ttyUSB1 comport/COM2`

访问端口。


注意 `COM1`、`COM2` 字母需大写。

4 功能说明

4.1 烧写 Flash

烧写 Flash 需要以下几个步骤：

1. 将手机或开发板通过 dongle 正确连接到 PC 的 USB 端口上。
2. 选择 lod 文件

点菜单项“Flash->Choose Lod file”或工具条按钮，选定要下载到 Flash 内的 Lod 文件（如图 5、6）。

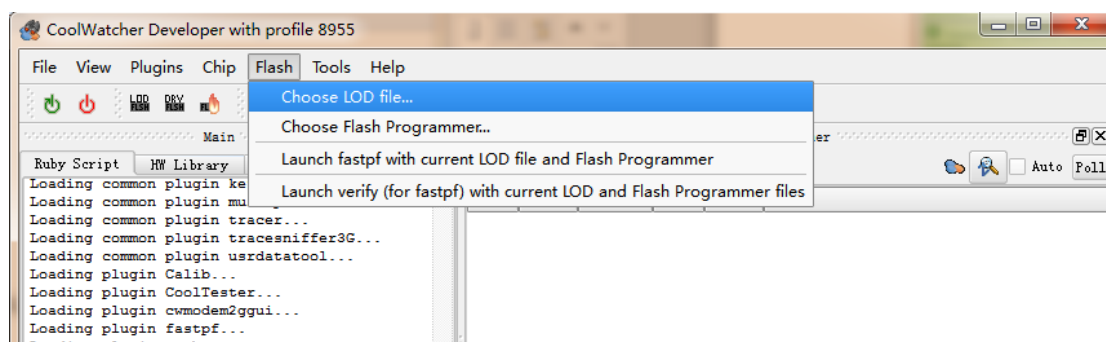


图 5 选择 lod 文件

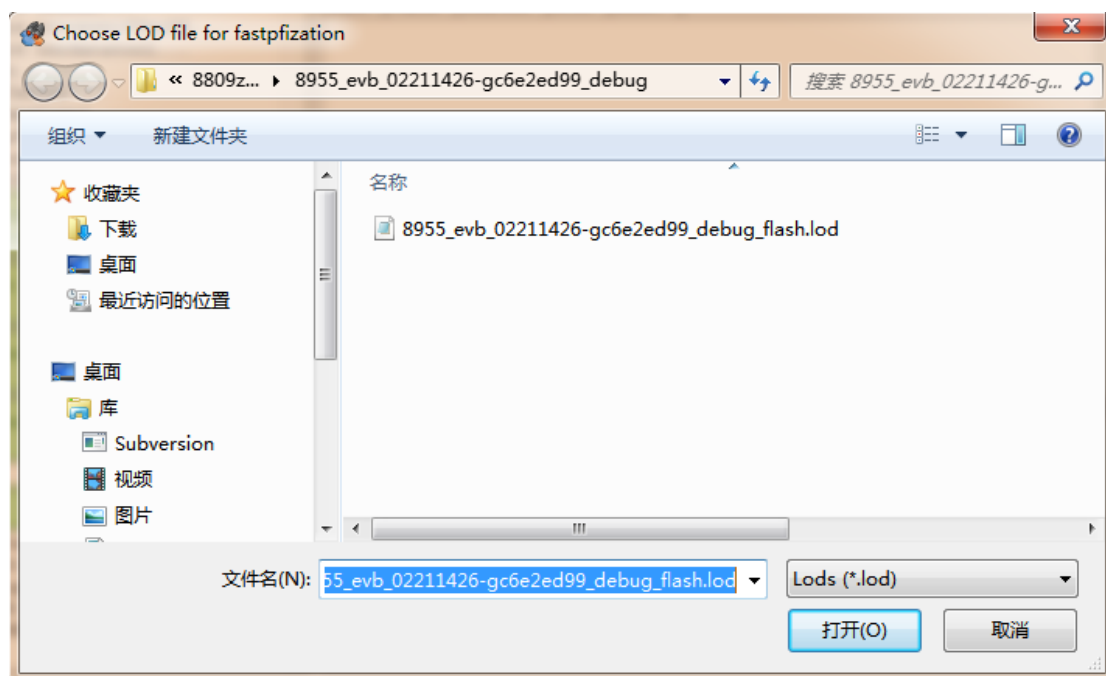



图 6 选择 lod 文件

3. 选择 ramrun 文件

点菜单项“Flash->Choose Flash programmer...”或工具条按钮，为该手机所使用的Flash 类型选定正确的 Flash programmer (*_ramrun.lod) 文件（如图 7）。

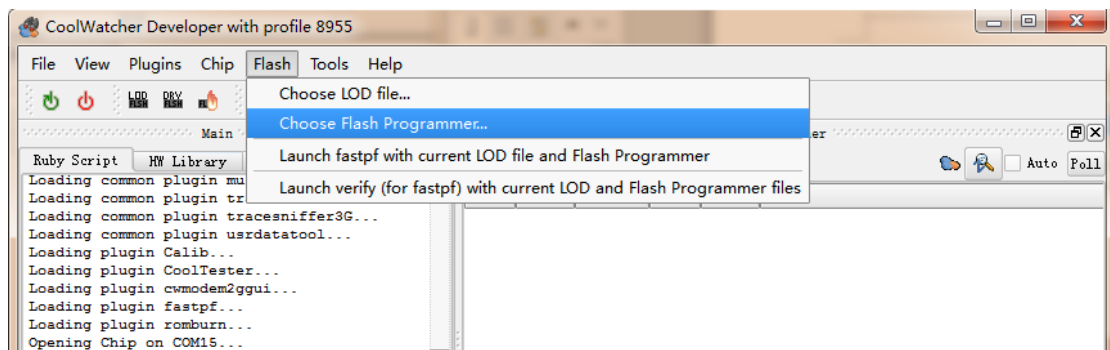


图 7 选择 ramrun 文件

4. 开始烧录


单项“Flash->Launch fastpf with current LOD file and flash programmer”或工具条按钮，开始下载 lod file。此时程序右下角出现一个进度条显示下载进度（如图 8），下载完成后进度条显示 100%，并且“Ruby Script”区显示完成信息。



图 8 烧录过程中进度条显示

4.2 查看 buffer 信息

在 Buffer Watcher 中的 Address 填写地址，Size 中填写大小，点击 Poll 按钮，可以从手机/模块读取相关数据。

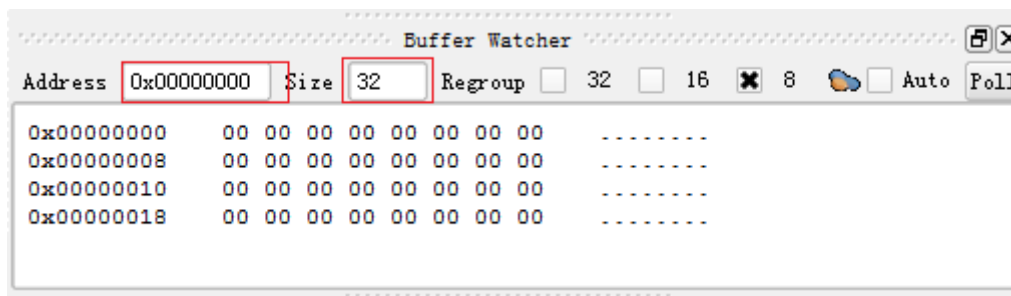


图 9 Buffer watcher 窗口

4.3 寄存器查看

把 HW Library 或 SW Library 中的某个节点拖入 Register Watcher 中，点击 pull 按钮，可以读取并显示相关内容。

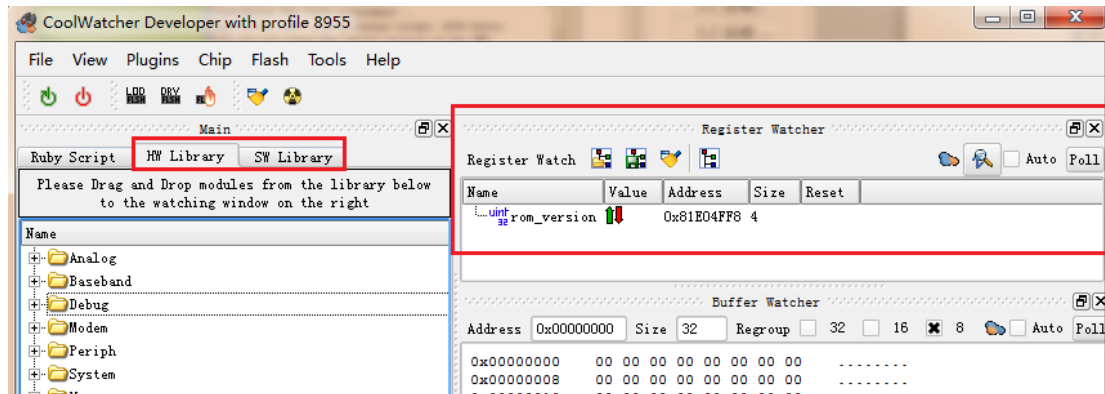


图 10 查看寄存器

4.4 查看 trace

4.4.1 启动 tracer

点击菜单 Plugins->Activate Tracer，如图 11，启动 Trace 插件，Trace 主界面如图 12

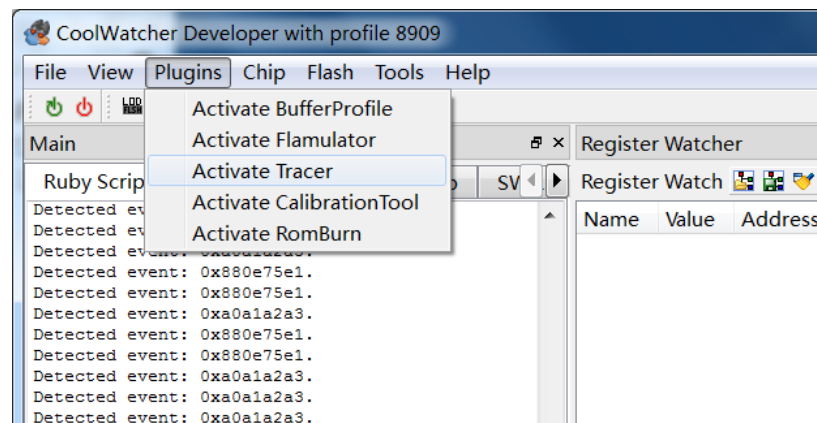


图 11 启动 tracer

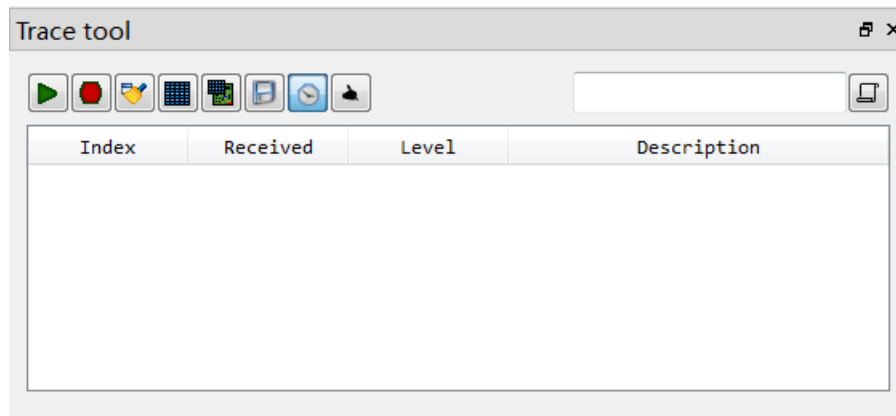


图 12 Tracer 主操作界面

工具栏各按钮分别对应: 开始 Trace、停止、清空、设置 TraceLevels、Reapply trace levels、保存、启动/关闭 Received 列、启动/关闭 comment。

各列分别对应: PC 接收 Trace 时间、手机/模块发出 Trace 的时间、Level、描述等。

4.4.2. Tracer 菜单项

插件启动后，主菜单增加 Tracer 项，如图 13 所示:

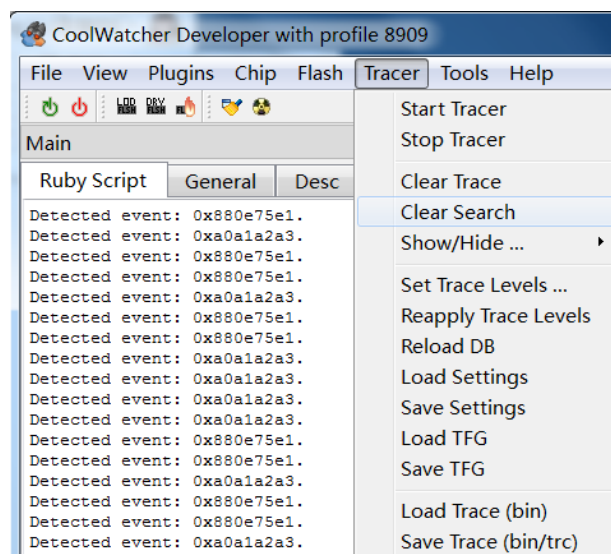


图 13 tracer 菜单项

菜单项

Start Tracer: 开始;

Stop Tracer: 停止;

Clean Trace: 清空 Trace 表格;


Clean Search: 清空 Trace Search 表格;

Toggle Raw Data: 启动/关闭 Raw 表格;

Set Trace Levels: 设置 Trace Levels;
Load DB: 加载 DB 文件;
LoadSettings: 加载 Levels 配置;
SaveSettings: 保存 Levels 配置;
Save TFG: 选择保存 T 卡 Trace 文件;
Trace Filter: 启动 Trace Filter 配置框;
Load bin: 加载二进制 Trace;
Save Trace (bin/trc): 保存 trace。

4.4.3. Tracer 应用步骤

1. 设置 TraceLevels

点击  按钮, 设置 tracelevel, 显示如图 14。需在左侧表格中选择关注的 levels;

AutoSave

- check 是否自动保存 trace, 状态为 checked, 则自动保存。
- bin、trc: 为 trace 文件类型, 前者为二进制文件, 后者为文本文件;
- Split Size: trace 文件大小。当文件 size 超过该值时, 自动切分文件;

DB file name

DB 文件。

RowLimit

图 12 的 Trace 表格的最大行数;

Auto reapply trace levels on reset

重启前的 TraceLevels 配置信息是否用于重启后。

Save TimeStamp in Trc

是否保存时间戳。

Save ReceivedTime in Trc

是否保存工具解析 trace 的时间

Tick in flow ID 0x80

该配置项需与 lod 保持一致, 如果 lod 中有时间戳, 则选中本项, 否则不选。

ReceiveEvent

是否接收 Event.

SavePcap

是否解析保存 pcap 信息到*.pcap 文件.

左下角

Save 按钮: 保存 Levels 配置;
Load 按钮: 加载 Levels 配置;

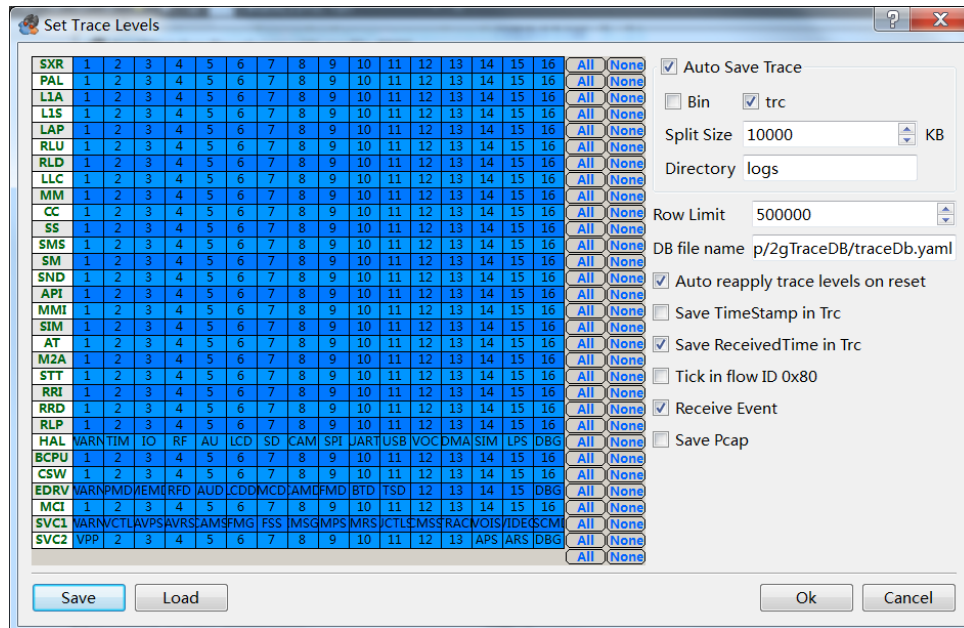



图 14 trace level 界面设置

2. 点击图 15 中的  按钮，开始 Trace 功能，Trace 信息会显示到表格中，如图 15 所示：

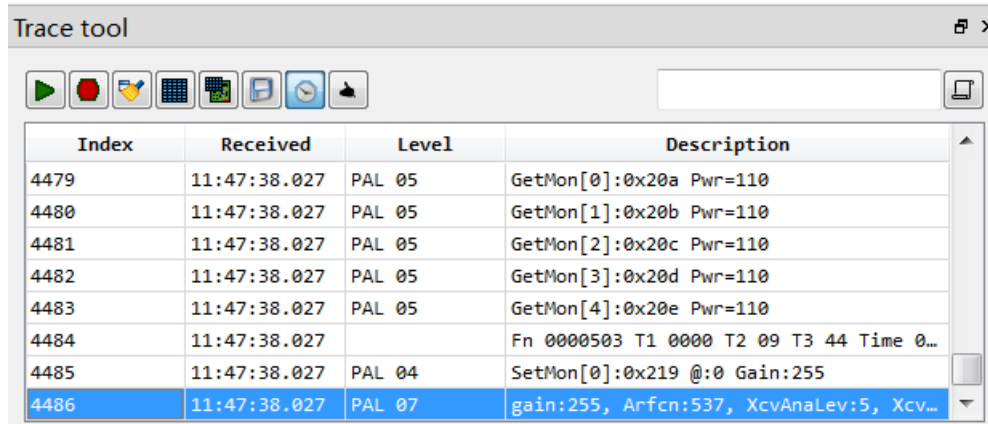



图 15 tracer 显示界面

3. 点击图 15 中的  按钮，结束 Trace 功能；

注意：Tracer 的配置信息，默认为 rbase\common\plugins\tracer\ 文件夹中的文件。

4.5 GDB Launcher

GDB 是分析死机、跟踪问题的一种重要方法。

4.5.1 启动 GDB

点击菜单项 Tools->GDB Launcher，启动如下配置框：

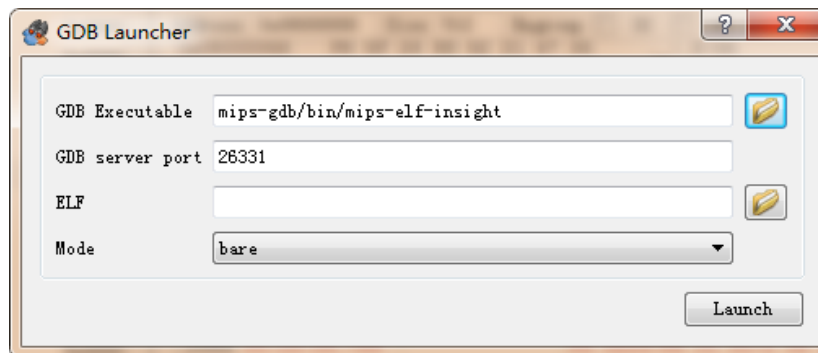


图 16 启动 GDB

注意：

- GDB exe 文件：选择 mips-gdb\bin 中的 mips-elf-insight.exe 文件；
- elf 文件：elf 文件对应的 lod 需要与手机/模块中已烧录的 lod 保持一致，一般是与 lod 一起发布的 elf 文件；
- Mode 模式：一般选择 bare
五种模式 “bare”、“sx(with REDUCED_REGS)”、“sx(without REDUCED_REGS)”、xcpu_rom、live；“sx”相比“bare”启动的 gdb，多了两个命令，thread info 和 thread；“xcpu_rom”一般用于系统启动之前的死机；“live”模式一般用于严重的死机分析。
注意：8910AP 和 8910CP 不能用于 8909、8955、8809e2、8908 等模块。

点击 Launch 按钮后，会弹出如下界面：

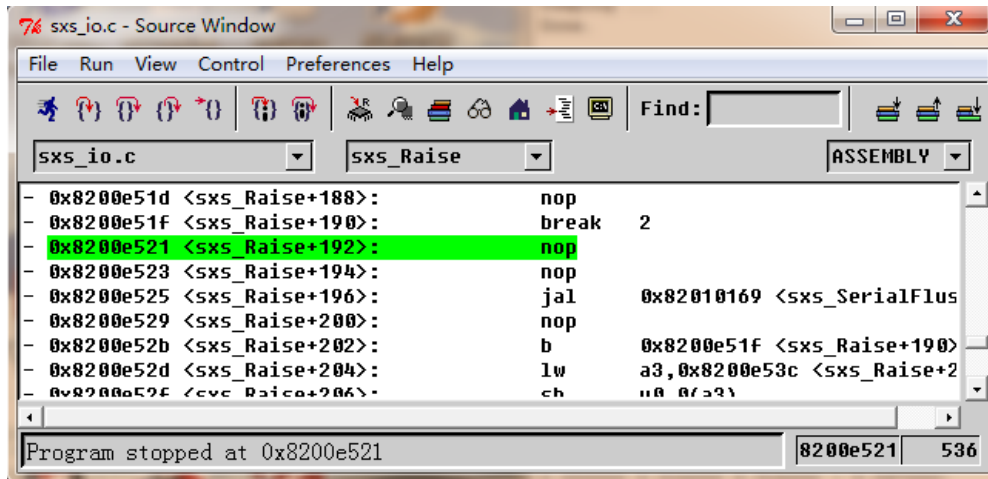


图 17 GDB 源码窗口

点击图 17 工具栏上的 Console 按钮 , 启动 gdb 命令窗口:

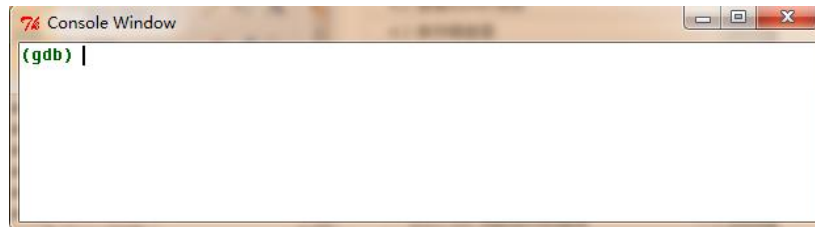


图 18 GDB 命令行

输入 gdb 命令, 即可分析代码。

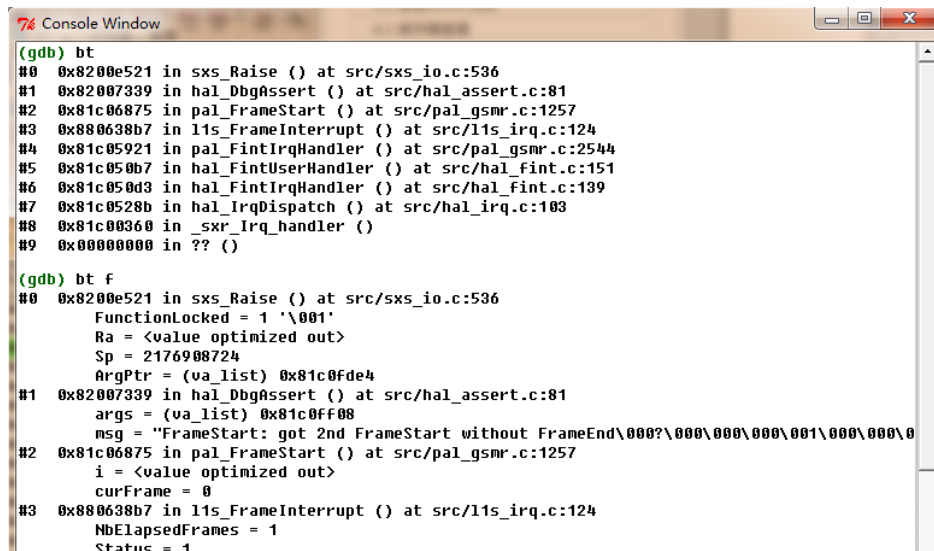


图 19 GDB 分析结果显示

4.5.2 GDB 常用命令

ID	GDB 命令	注释	
1	p<sth>(print <sth>)	打印 sth 的值，sth 可以是一个表达式、变量、指针等。	
2	display <sth>	和“p sth”相同，不过 display 会在你每次输入命令后显示 sth 的值。	
3	bt or bt f (backtrace or backtrace full)	Backtraces (backtraces full) the current executed code. You get the call stack, parameters passed to each function, & so on. By using "full" you will also get the display of all local variables for these functions (EXTREMELY useful).	
4	up	Goes up into the call stack. To be used in conjunction with bt.	
5	down	Goes down into the call stack. To be used in conjunction with bt.	

4.6 Buffer Profile

Profile 是分析问题的一种重要方法。

4.6.1 抓取 prf 文件

点击 coolwatcher 主界面中的 Plugins->Activate BufferProfile 菜单项，如图 20，启动 Buffer Profile 插件，主菜单上会出现 BufferProfile 菜单，工具栏上会有相关按钮。

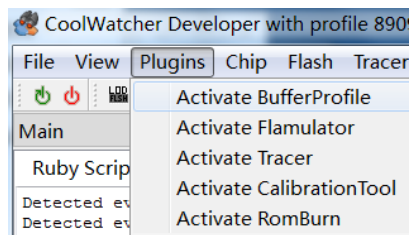


图 20 启动 buffer profile

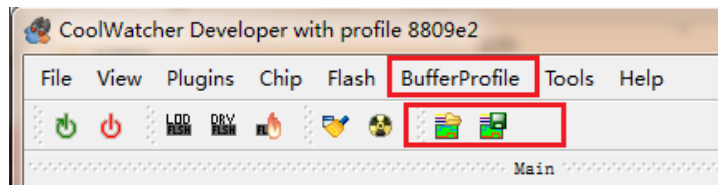


图 21 工具栏新增的 bufferProfile 按钮

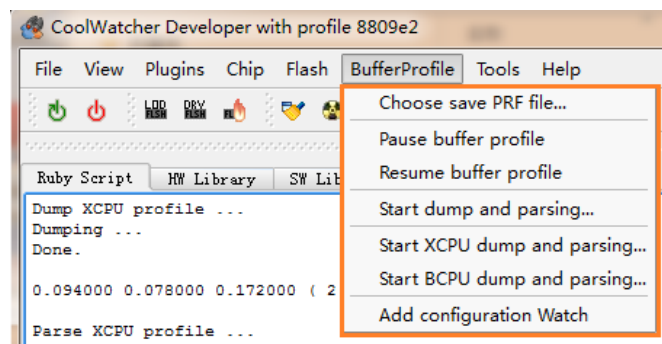



图 22 buffer profile 状态栏

手机死机后，可以通过该插件抓取有关信息。使用方法如下：

1. 手机通过 COM 口与 PC 机相连，保证可以与 coolwatcher 正常通信。
2. 点击图 22 中 Choose save PRF file 菜单项，或工具栏上的  按钮，输入要保存的 *.prf 文件，文件的默认路径与 coolwatcher.exe 路径一致。
3. 点击图 22 中 Start XCPU dump and parsing 菜单，开始保存 XCPU 相关信息。类似信息如下图：

```
> parseProfilerGo(PARSE_ALL_PROFILE)
XCPU Buffer Address = 0x824747cc
XCPU Record Position = 0x22ed
XCPU Record Number = 0x3000
Dump XCPU profile ...
Dumping ...
Done.

0.094000 0.078000 0.172000 ( 2.919000)

Parse XCPU profile ...
1 - File Size = 49243
1 - Buffer Size = 49152
1 - Buffer Position = 35764
1 - Buffer freq = 16384
Begin parsing memory profile ...
count = 49152
Parse done: F:/cooltools-win32_20170224/test_r630.prf
```

图 23 保存 XCPU 相关信息

4. 如点击图 22 中 Start BCPU dump and parsing 菜单，则保存 BCPU 相关信息。类似信息如下图：

```
BCPU buffer profile exists!
BCPU Buffer Address = 0xa1983ab0
BCPU Record Position = 0x32
BCPU Record Number = 0x40
Dump BCPU profile ...
Dumping ...
Done.

0.000000 0.015000 0.015000 ( 0.037000)

Parse BCPU profile ...
1 - File Size = 343
1 - Buffer Size = 256
1 - Buffer Position = 200
1 - Buffer freq = 16384
Begin parsing memory profile ...
count = 256
Parse done: F:/cooltools-win32_20170224/test_r630_bcpu.prf
```

图 24 保存 BCPU 相关信息

5. 如点击图 22 中 Start dump and parsing 菜单，则保存 BCPU 和 XCPU 两者的信息。Coolwatcher 输出如下图：

```
Combine and parse both XCPU and BCPU profile ...
1 - File Size = 49243
1 - Buffer Size = 49152
1 - Buffer Position = 35764
1 - Buffer freq = 16384
2 - File Size = 343
2 - Buffer Size = 256
2 - Buffer Position = 200
Begin parsing memory profile ...
count = 49408
Parse done: F:/cooltools-win32_20170224/test_r630_all.prf
```

图 25 保存 XCPU 与 BCPU 信息

注意

如果手机没有死机，则抓取*.prf 之前，最好点击菜单项中的 Pause buffer profile 项，文件保存完成后，点击 Resume buffer profile 菜单项，恢复状态。

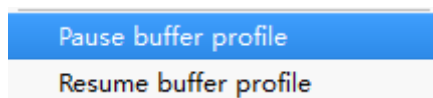


图 26 手机未死机时保存 profile 文件

4.6.2 分析 prf 文件

打开 coolprofile.exe 工具，如图 27 所示。

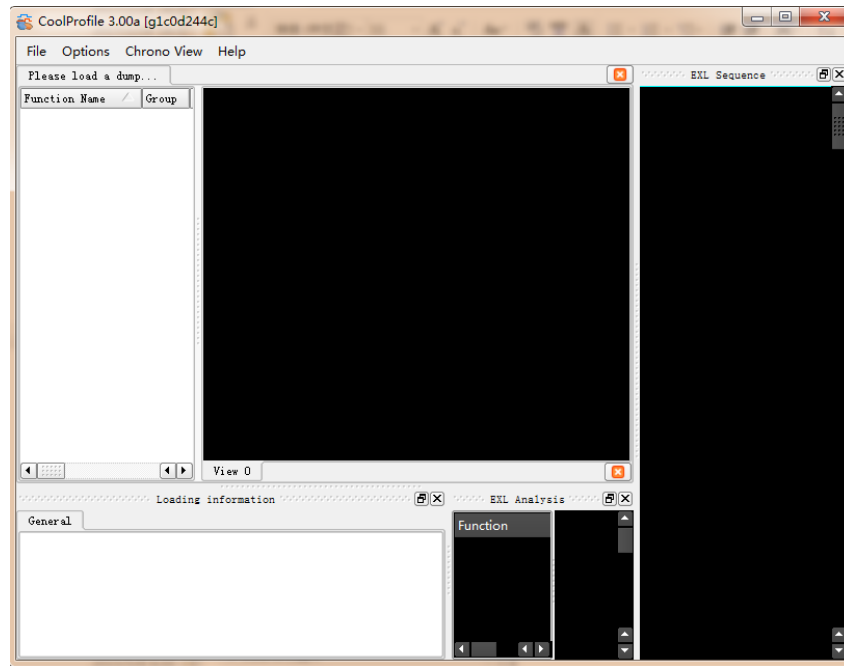


图 27 coolprofile 工具主界面

点击 file->open 按钮，通过下对话框，选择 prf 文件。

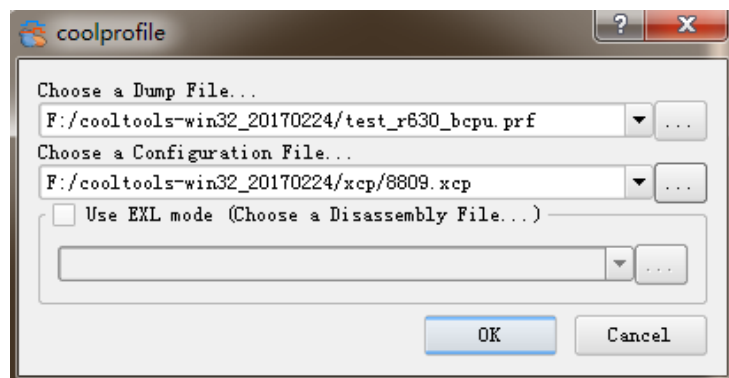


图 28 选择 profile 文件

打开效果如下：

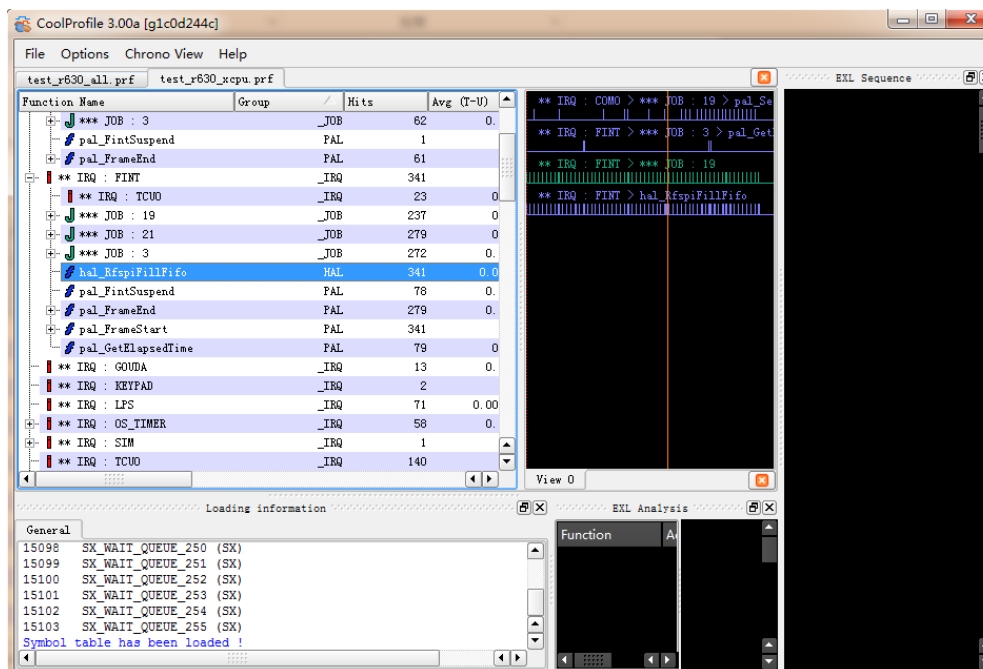


图 29 加载 profile 文件

即可分析 prf 文件。

4.7 离线分析

4.7.1 生成*.core(*.elf)

手机死机后，可以利用 elfdump 命令可以生成*.core(*.elf)文件， 抓取*.core(*.elf)文件的命令见章节 4.10.5 ， 举例如下：

```
> elfdump "r630.elf"
building elf for 8809e2...
Reading page reg @0x81a0c000...
done
Reading xcpu reg @0x81a2b000...
done
Reading bcpu reg @0x8190a000...
done
```

图 30 生成 elf 文件

4.7.2 建立离线分析环境

步骤如下：

1. 加载要分析的*.core (或*.elf)文件。

Coolwatcher 关闭的情况下，启动 coolhost.exe 程序，主界面如下：

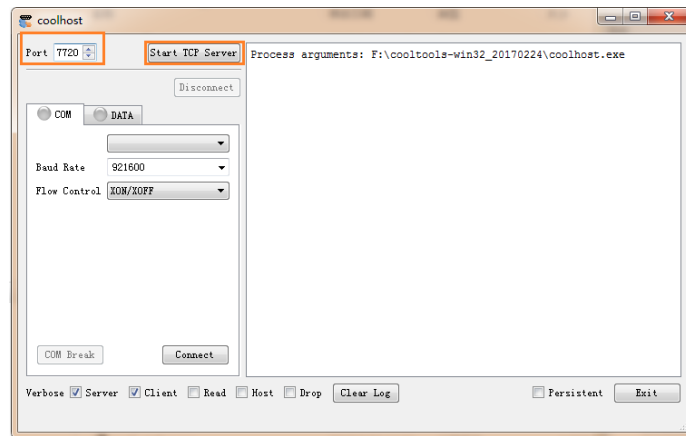


图 31 coolhost 主界面

- Port 端口号，可以随意定；
- 点击 Start TCP Server 按钮；
- 点开 DATA tab 页，右键点击 Add 菜单项，如下图，选择*.core (*.elf) 文件和 lod 文件，如图 32 所示。注意，先选择*.core (*.elf) 文件。

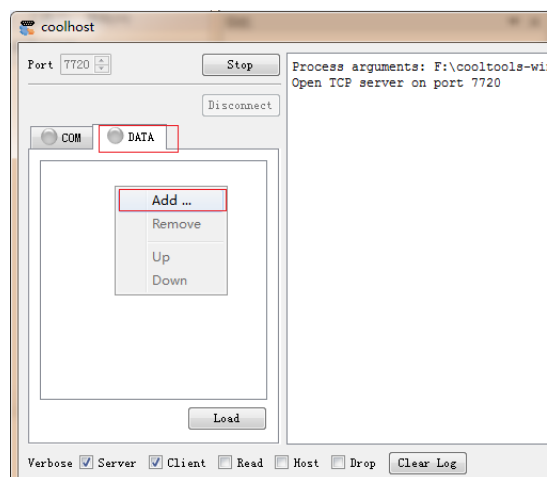


图 32 加载 elf 文件

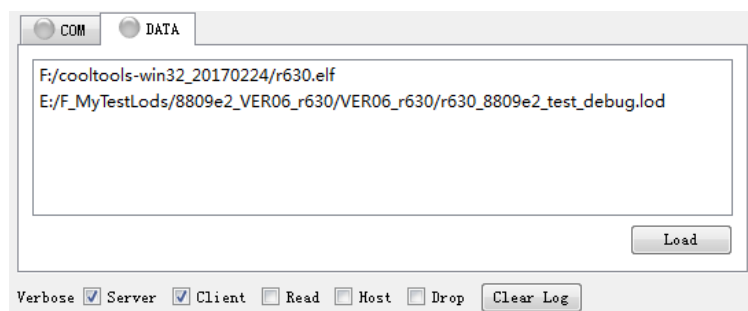
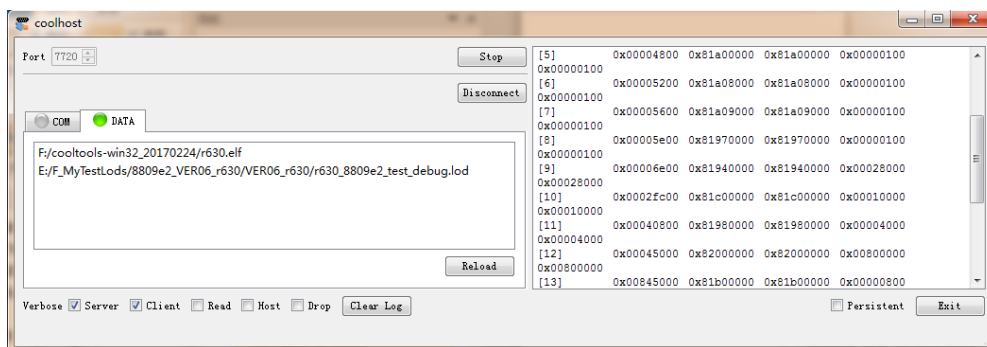


图 33 选择 elf 文件路径

- 加载文件：点击图 33 的 load 按钮，加载文件，效果如下图：



图

图 34 加载数据后界面显示

注意：该*.core (*.elf) 文件用 elfdump 命令生成，不是和 lod 一起发布的 elf 文件。

2. 启动离线分析环境

启动 coolwatcher，下图中的 lastcomport 项的数值要与图 32 中 Port 7720 的后两位保持一致，如 Port 选择 7720，lastcomport 项的值则为 20。

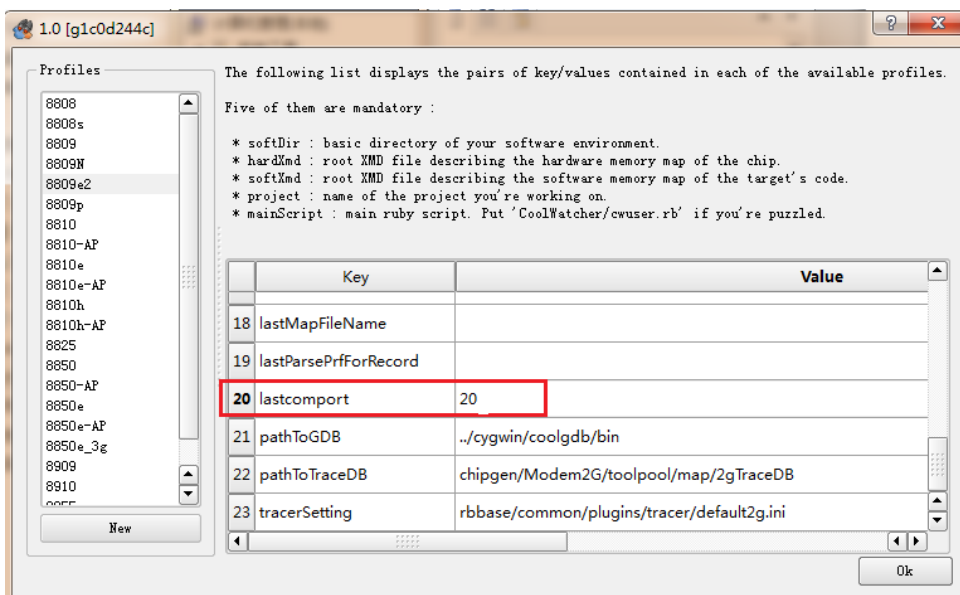


图 35 设置 lastcomport

启动 coolwatcher 后，即可进行离线分析。

4.7.3 离线分析 gdb

保持 4.7.2 环境不变，按照步骤 4.7.2 即可离线分析*.elf 文件。

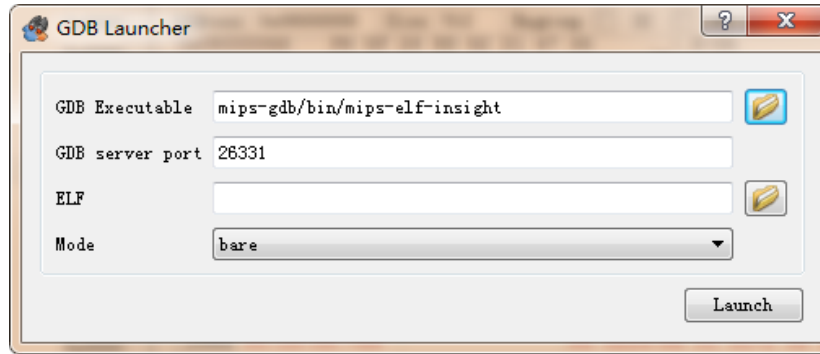


图 36 选择 elf 文件路径

注意：该配置框中选择的*.elf 文件，是与 lod 一起发布的*.elf 文件。

4.7.4 Elf Data Check

用于比较 dump 和原始 elf，检查代码是否有被改写。

启动过程

利用 4.7.2 章节介绍搭建离线环境后，点击 Tools->Elf Data Check 菜单，启动主界面图 38:

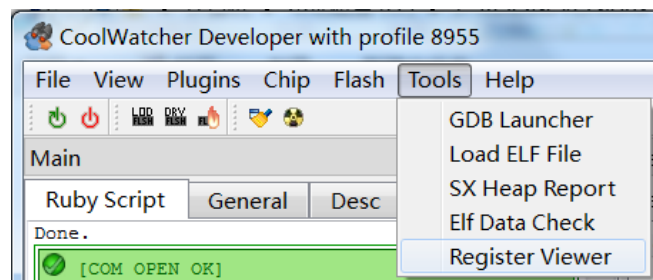


图 37 选择 Elf Data Check

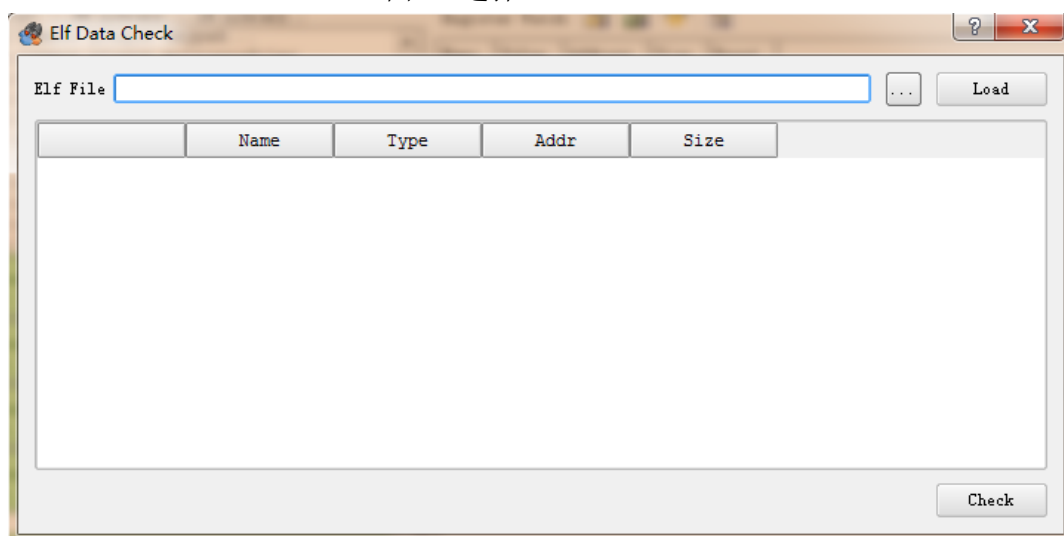


图 38 Elf Data Check 主界面

操作步骤:

1. 点击上图中的“load”按钮，选择与 lod 一起发布的*.elf 文件，点击“Check”按钮后如下图:

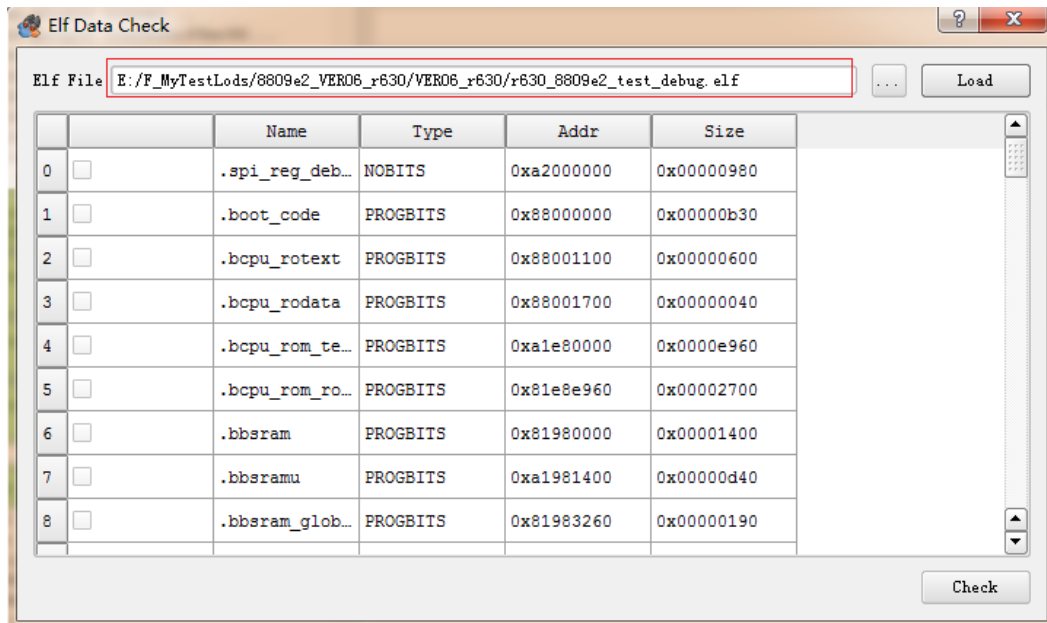


图 39 load elf 文件后界面显示

2. 选择待比较的项

通过操作首列 check box，选择待比较的项:

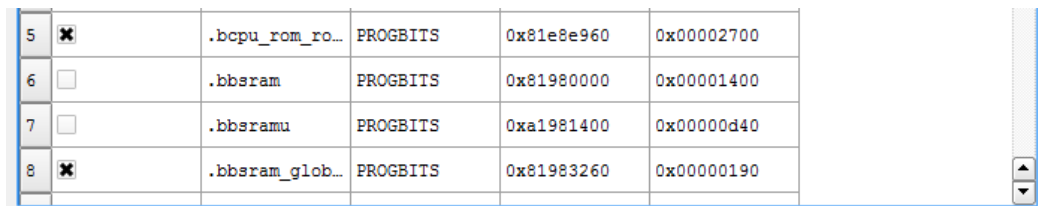


图 40 选择比较项

3. 比较和保存结果

点击图 39 中的 check 按钮，如果有不同，则会弹出下对话框:

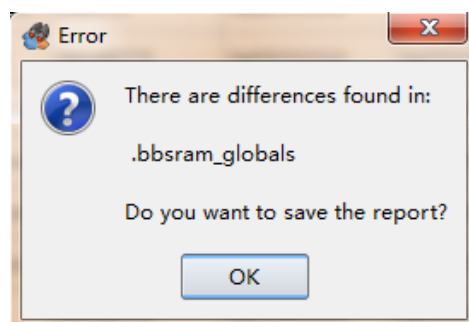


图 41 错误提示

点击 OK 按钮，选择/填写要保存的文件:

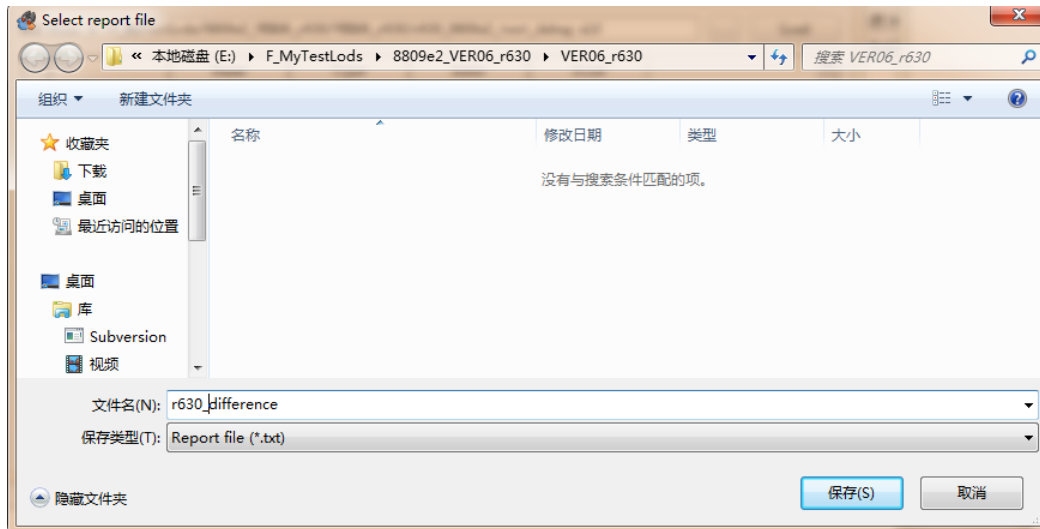


图 41 保存对比报告

不同处将会保存到文件中，效果如下图：

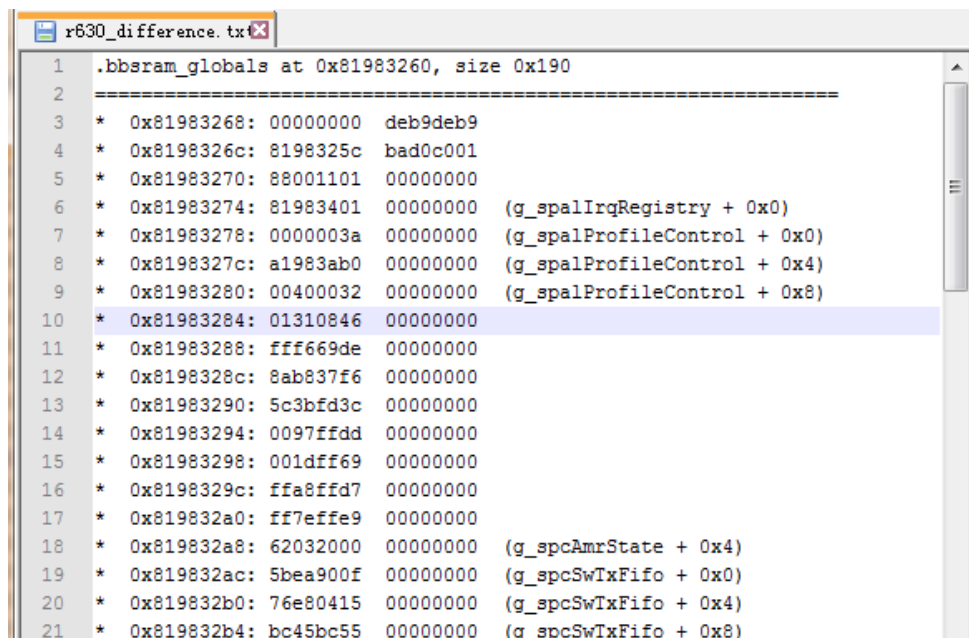


图 42 对比报告中不同处对比

4.8 Heap Report

不从 map 中读消息，而是直接从 ELF 中读消息。应可兼容 8809 没有添加 sx_access 的软件。

点击主菜单 Tools->Heap Report 项，如下图，启动 Heap Report 设置对话框，如图 44。

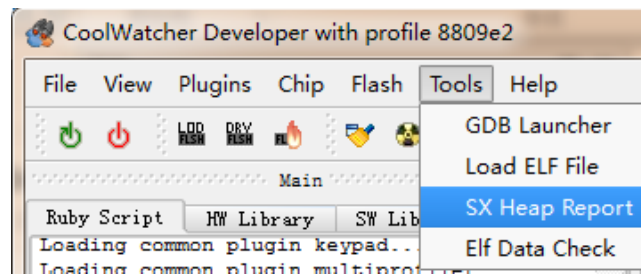


图 43 启动 Heap Report

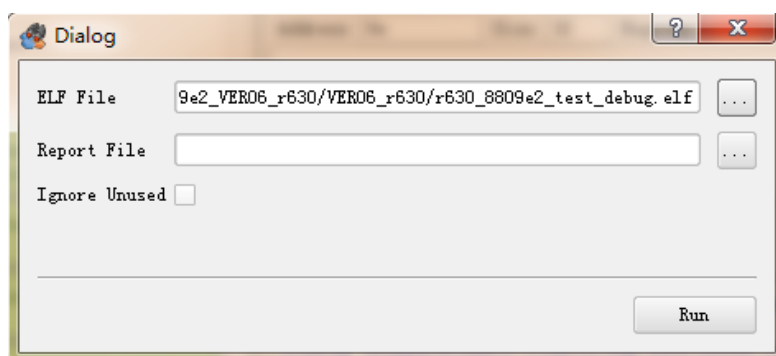


图 44 Heap Report 对话框

Elf 文件：与 lod 一起发布的 elf 文件；

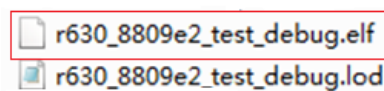
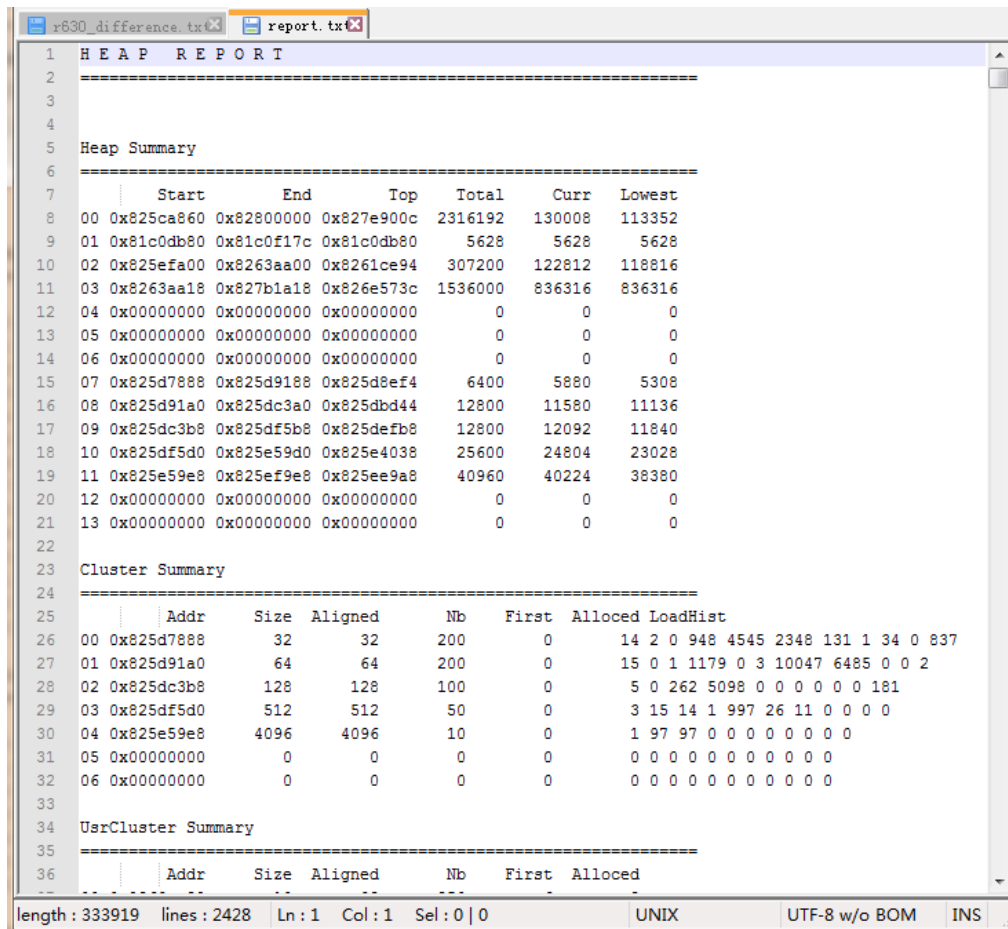


图 45 选择 elf 文件

Report File：要生成的文件。

选择或填写相关信息，点击 Run 按钮，生成 Report File，示例如下：



HEAP REPORT						
Heap Summary						
	Start	End	Top	Total	Curr	Lowest
00	0x825ca860	0x82800000	0x827e900c	2316192	130008	113352
01	0x81c0db80	0x81c0f17c	0x81c0db80	5628	5628	5628
02	0x825efa00	0x8263aa00	0x8261ce94	307200	122812	118816
03	0x8263aa18	0x827b1a18	0x826e573c	1536000	836316	836316
04	0x00000000	0x00000000	0x00000000	0	0	0
05	0x00000000	0x00000000	0x00000000	0	0	0
06	0x00000000	0x00000000	0x00000000	0	0	0
07	0x825d7888	0x825d9188	0x825d8ef4	6400	5880	5308
08	0x825d91a0	0x825dc3a0	0x825dbd44	12800	11580	11136
09	0x825dc3b8	0x825df5b8	0x825defb8	12800	12092	11840
10	0x825df5d0	0x825e59d0	0x825e4038	25600	24804	23028
11	0x825e59e8	0x825ef9e8	0x825ee9a8	40960	40224	38380
12	0x00000000	0x00000000	0x00000000	0	0	0
13	0x00000000	0x00000000	0x00000000	0	0	0
Cluster Summary						
	Addr	Size	Aligned	Nb	First	Alloced LoadHist
00	0x825d7888	32	32	200	0	14 2 0 948 4545 2348 131 1 34 0 837
01	0x825d91a0	64	64	200	0	15 0 1 1179 0 3 10047 6485 0 0 2
02	0x825dc3b8	128	128	100	0	5 0 262 5098 0 0 0 0 0 0 181
03	0x825df5d0	512	512	50	0	3 15 14 1 997 26 11 0 0 0 0
04	0x825e59e8	4096	4096	10	0	1 97 97 0 0 0 0 0 0 0 0
05	0x00000000	0	0	0	0	0 0 0 0 0 0 0 0 0 0 0
06	0x00000000	0	0	0	0	0 0 0 0 0 0 0 0 0 0 0
UserCluster Summary						
	Addr	Size	Aligned	Nb	First	Alloced

图 46 生成 Report File

4.9 Register Viewer

RegisterViewer 工具可以按 bit 位读取、编辑寄存器，同时具有查找功能。点击主菜单 Tools->Register Viewer 项，如下图，启动 Register Viewer 页面，如图 48。

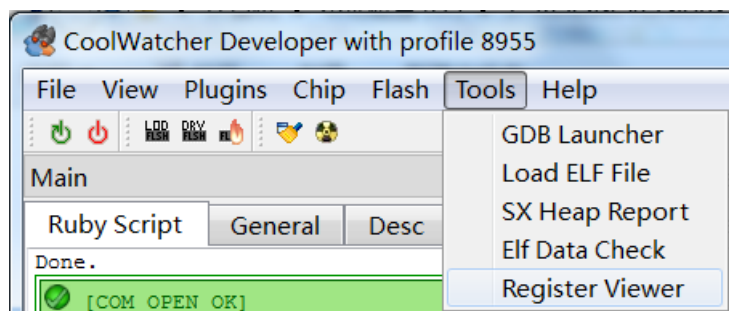


图 47 启动 Register Viewer

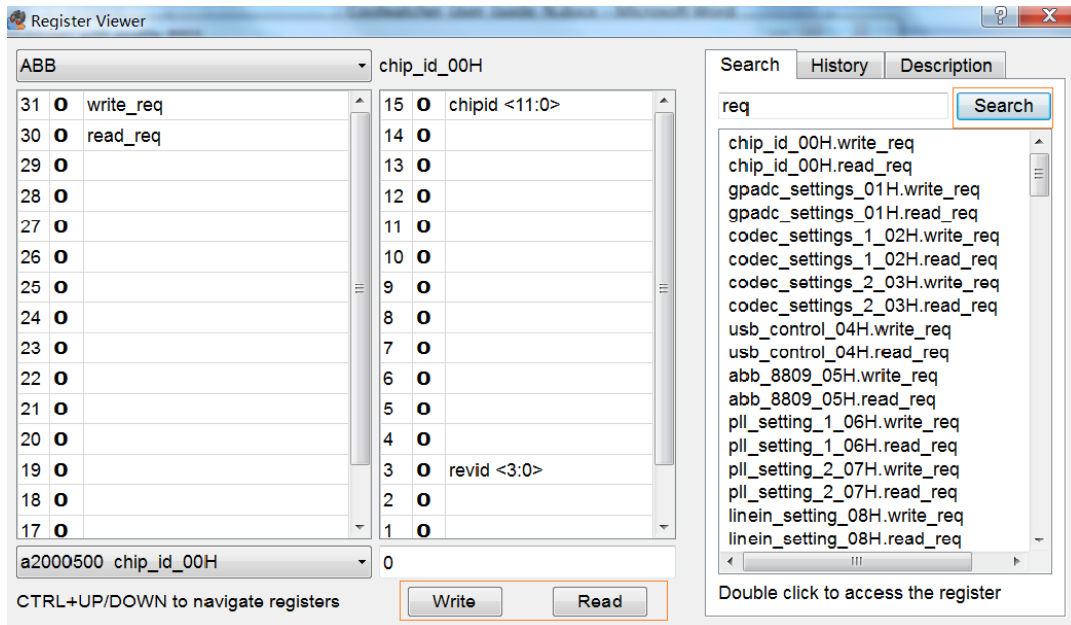



图 48 Register Viewer 主界面


4.10 芯片控制

工具提供了下列常用的对芯片进行控制的操作功能，大大提高了用户调试过程中对手机硬件控制的能力。

4.10.1 关闭芯片

点击菜单项“Chip->Turn off the chip”或工具条按钮，可以控制手机或开发板关机。

4.10.2 重新启动芯片

点击菜单项“Chip->Restart the chip”或工具条按钮，可以控制手机或开发板重新启动，便于调试用户程序。

4.11 命令行操作

本工具提供了一些开发中常用的操作命令，用户可以很方便的在命令行输入框输入这些命令，完成相应的操作，并将操作结果显示在“Ruby script”框内。

4.11.1 端口操作

常用的端口操作命令如下列表:

命令	参数	示例	备注
copen	(NUM,BR=BR_AUTOMATIC)	copen(2,115200)	打开 COM2
reop		reop	重新打开当前端口

4.11.2 Flash 编程

常用的 Flash 编程命令如下列表:

命令	参数	示例	备注
fastpf	(flash_programmer_filename, lod_filename, disable_event_sniffer = true)	fastpf("c:\xxxx_ramrun.lod", "c:\\lod")	烧写 Flash
fastSectorEraser	(flash_programmer_filename, sector_list, disable_event_sniffer = true)	fastSectorEraser ("c:\xxxx_ramrun.lod", [0x01000000,0x01200000] ")	擦除扇区

4.11.3 读 Flash

常用的读 Flash 操作命令如下列表

命令	参数	示例	备注
r	(addr)	r x01004000	32 位写
r32	(addr)	r32 0x01004000	32 位写
r16	(addr)	r16 0x01004000	16 位写
r8	(addr)	r8 0x01004000	8 位写

4.11.4 写 Flash

常用的写 Flash 操作命令如下列表


命令	参数	示例	备注
w	(addr,val)	w(0x01004000,0xffffffff)	32 位读
w32	(addr,val)	W32(0x01004000,0xffffffff)	32 位读
w16	(addr,val)	W16(0x01004000,0xffff)	16 位读
w8	(addr,val)	W8(0x01004000,0xff)	8 位读

4.11.5 其他命令

命令	参数	示例
elfdump	elf 文件 [,xml 文件名] xml 文件名可以不写, 如果写需要给出绝对路径	elfdump "test.elf" 或者 elfdump "test.elf", "D:/8809.xml"
dump	(str_filename,address,nbwords) ,读取一段连续的 flash 区域到指定文件。	Dump("c:\\xxxx.lod",0x01004,0x100)
chipID	无参数, 直接返回芯片 ID	000,0x1000)
flashReadStatus	无参数	
flashSectorErase	flashSectorErase(addr)	
flashBlockErase	flashBlockErase(addr)	
flashBlock32kErase	flashBlock32kErase(addr)	
flashChipErase	flashChipErase()	
xcvRead	xcvRead(addr)	
xcvWrite	xcvWrite(addr,data)	

4.12 其他功能

4.12.1 Kill 当前运行的程序

点击菜单项“File->Kill command thread”或工具条按钮, 可以强制终止当前正在运行的程序（一般为用户在命令行输入的命令线程）。

4.12.2 Kill 所有运行的程序

点击菜单项“File->Kill all thread”, 可以强制终止正在运行的全部（一般为用户在命令行输入的命令线程）。

4.12.3 清除脚本输出信息

点击菜单项“View->Clear Screen”或工具栏按钮, 可以清空左下方的“Ruby script”区域的输出信息。

4.13 Linux 串口配置

在应用程序 coolwatcher.exe 所在文件夹中建立子文件夹 comport。

为/dev/ttyUSB0、/dev/ttyUSB1 创建符号连接文件

如 `ln -s /dev/ttyUSB0 comport/COM1`

`ln -s /dev/ttyUSB1 comport/COM2`

访问端口。

5 疑难解析

6 附录