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](#_Toc502150749)

[2 软、硬件需求 4](#_Toc502150750)

[3 安装与设置 4](#_Toc502150751)

[4 功能说明 4](#_Toc502150752)

[4.1 烧写 Flash 7](#_Toc502150753)

[4.2 查看buffer信息 8](#_Toc502150754)

[4.3 寄存器查看 8](#_Toc502150755)

[4.4 查看trace 9](#_Toc502150756)

[4.4.1 查看trace 9](#_Toc502150757)

[4.5 GDB Launcher 12](#_Toc502150758)

[4.5.1 启动GDB 12](#_Toc502150759)

[4.5.2 GDB常用命令 13](#_Toc502150760)

[4.6 Buffer Profile 14](#_Toc502150761)

[4.6.1 抓取prf文件 14](#_Toc502150762)

[4.6.1 分析prf文件 15](#_Toc502150763)

[4.7离线分析 17](#_Toc502150764)

[4.7.1 生成\*.core(\*.elf) 17](#_Toc502150765)

[4.7.2 建立离线分析环境 17](#_Toc502150766)

[4.7.3 离线分析gdb 19](#_Toc502150767)

[4.7.4 Elf Data Check 19](#_Toc502150768)

[4.8 Heap Report 22](#_Toc502150769)

[4.9 RegisterViewer 23](#_Toc502150770)

[4.10 芯片控制 24](#_Toc502150771)

[4.10.1 关闭芯片 24](#_Toc502150772)

[4.10.2 重新启动芯片 24](#_Toc502150773)

[4.11命令行操作 24](#_Toc502150774)

[4.11.1 端口操作 25](#_Toc502150775)

[4.11.2 Flash 编程 25](#_Toc502150776)

[4.11.3 读 Flash 25](#_Toc502150777)

[4.11.3 写 Flash 26](#_Toc502150778)

[4.11.5 其他命令 26](#_Toc502150779)

[4.12 其他功能 27](#_Toc502150780)

[4.12.1 Kill 当前运行的程序 27](#_Toc502150781)

[4.12.2 Kill 所有运行的程序 27](#_Toc502150782)

[4.12.3 清除脚本输出信息 27](#_Toc502150783)

[4.13 linux 版本识别端口 27](#_Toc502150784)

[5 疑难解析 27](#_Toc502150785)

[6 附录 27](#_Toc502150786)

# 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 软、硬件需求

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。

**软件需求**

PC 机安装 Windows XP/7/10、CSDTK。

# 3 安装与设置

## linux 下的使用

1. 依赖包安装

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

1. 把99-coolsand-dongle.rules文件

把该文件放到 /etc/udev/rules.d/路径中

1. 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

点击可执行程序coolwatcher.exe ,启动配置界面，如下图：

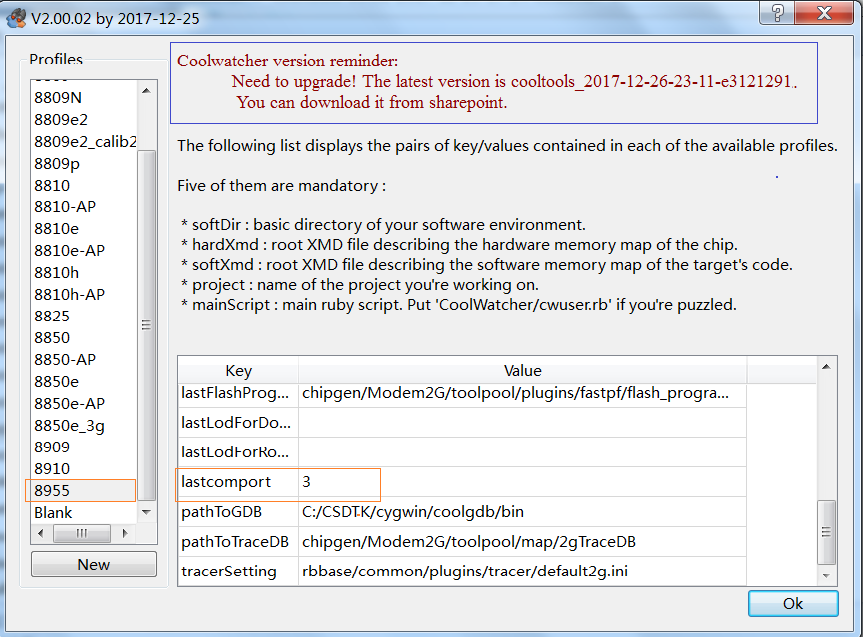


图1

左侧列表profiles为手机/模块类型，右侧表格为配置项。

步骤：

1. 选择手机/模块类型；
2. 修改必要的配置项：如端口号；

端口号可通过计算机->管理->设备管理器->端口 查看

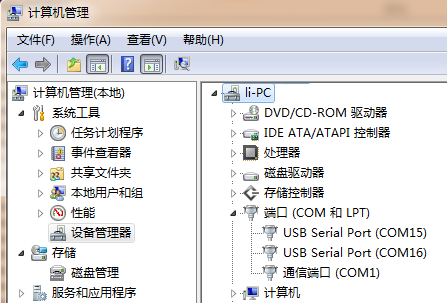


图2

1. 点击OK按钮，启动coolwatcher主操作界面。

主操作界面如下：

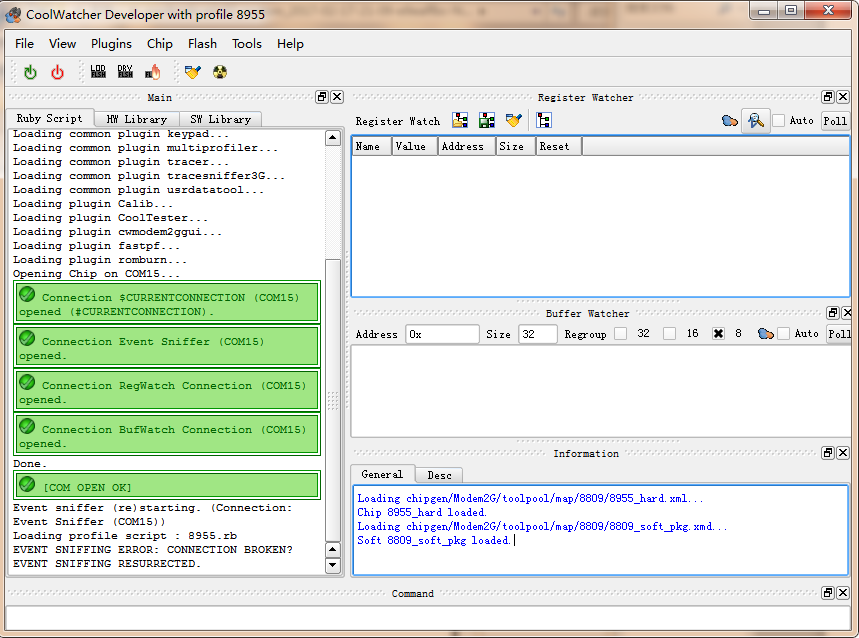


图 3

具有菜单栏、工具栏，信息打印区，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。

具有烧写flash、查看buffer信息、读取寄存器、查看trace、dump 数据等功能。

**版本提醒功能**

配置页面有最新版本提醒功能，如果当前启动的版本不是最新版本，提示见图4-1； 如果当前启动的版本是最新版本，提示见图4-2。

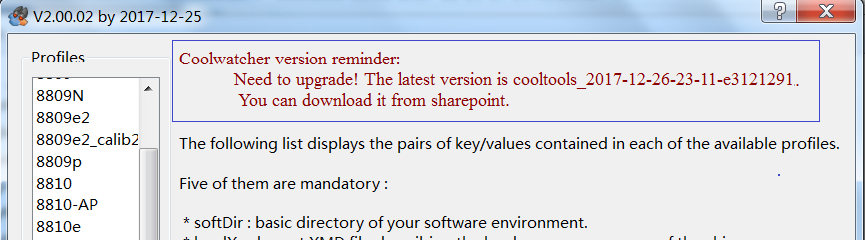


图 4-1

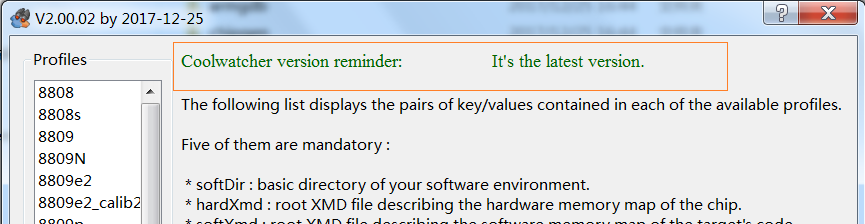


图 4-2

## 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． 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）。

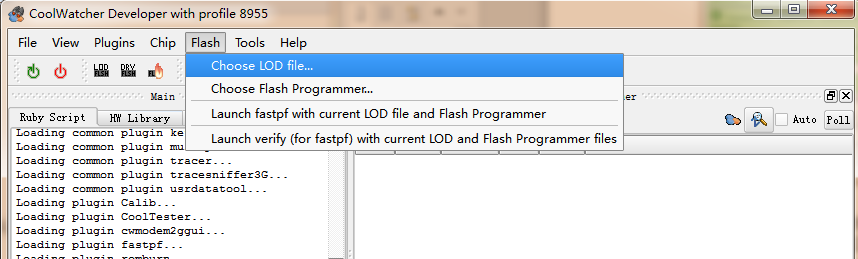


图 5

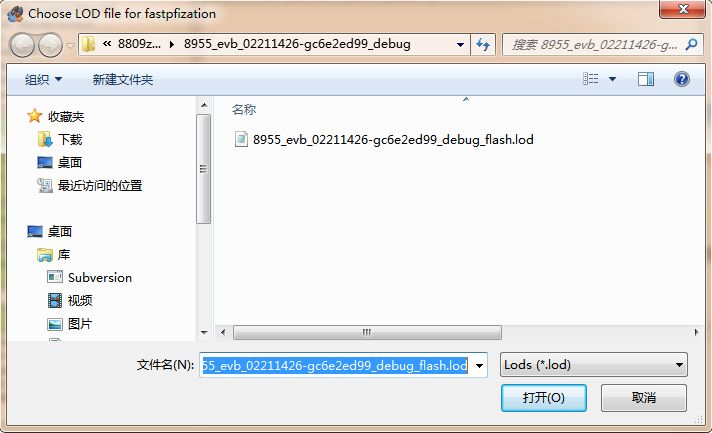


图6

3． 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）。

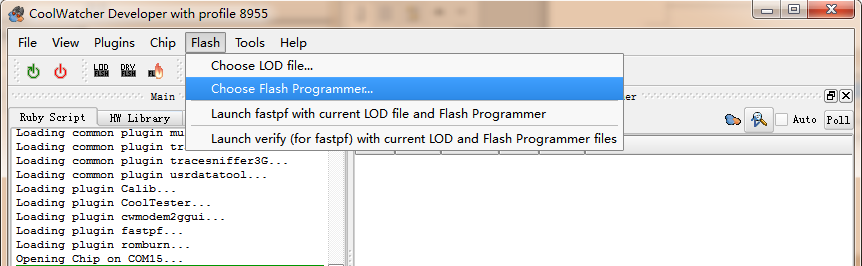


图7

4． 点菜单项“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

## 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。

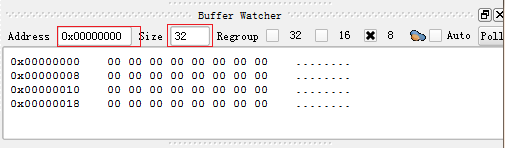
****

图9

## 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。

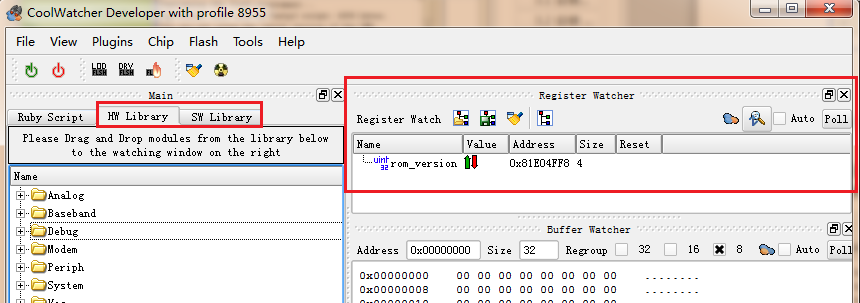
****

图10

## 4.4 查看trace

### 4.4.1 查看trace

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

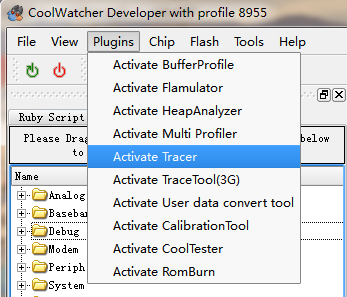
****

图11

**Tracer主操作界面**

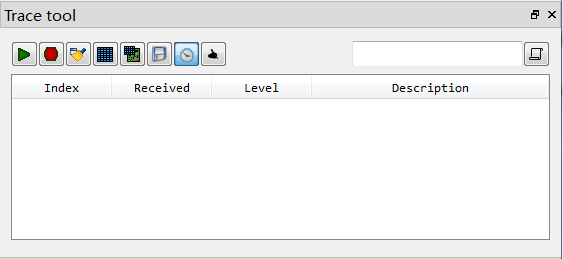
****

图12

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。

**Tracer菜单项**

插件启动后，主菜单增加Tracer项，如下图:

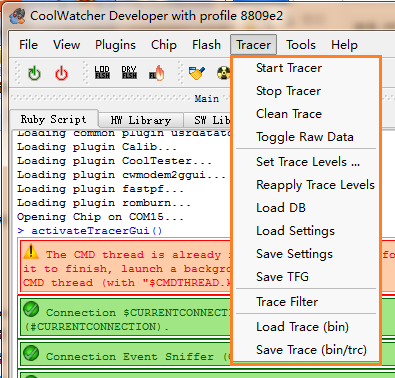


图13

菜单项

Start Tracer：开始；

Stop Tracer：停止；

Clean Trace：清空Trace表格；

Toggle Raw Data: 启动/关闭Raw 表格；

Set Trace Levels：设置Trace Levels；

Load DB：加载图13的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。

**应用步骤如下：**

1. 点击按钮，设置TraceLevels

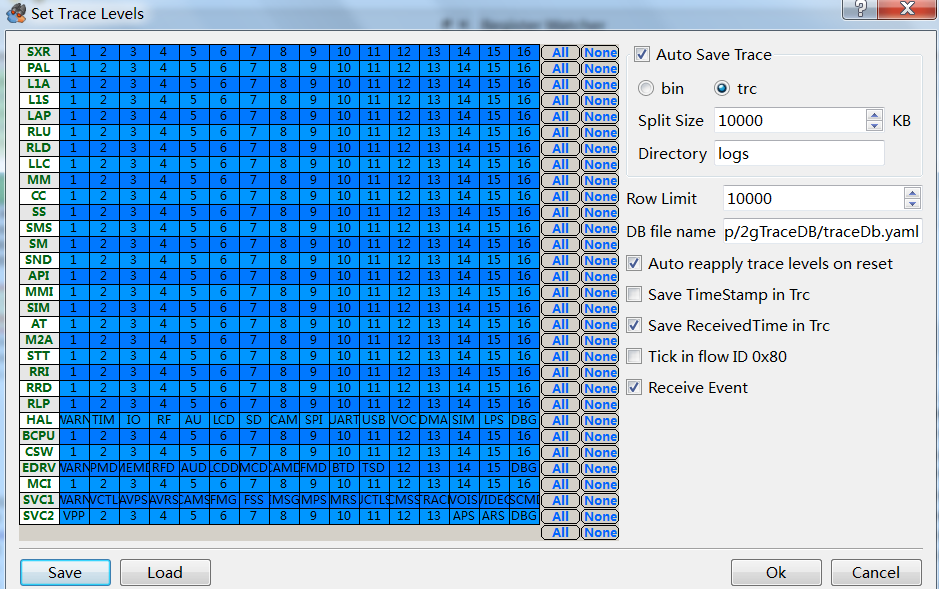


图14

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.

左下角

Save按钮：保存Levels 配置；

Load 按钮：加载Levels配置；

1. 点击图15中的 按钮，开始Trace功能，Trace信息会显示到表格中，如下图：

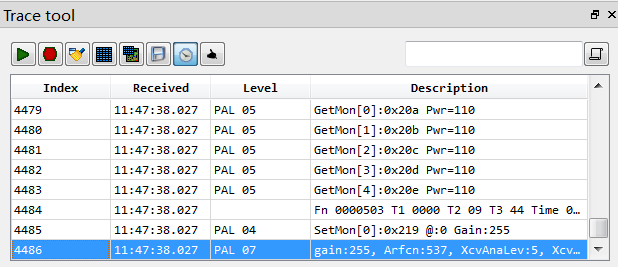


图15

1. 点击图15中的按钮，结束Trace功能；

注意：Tracer的配置信息，默认为rbbase\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：

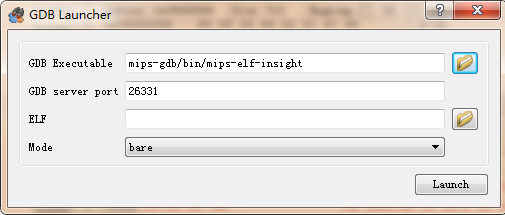


图16

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。

点击Launch按钮后，会弹出如下界面：

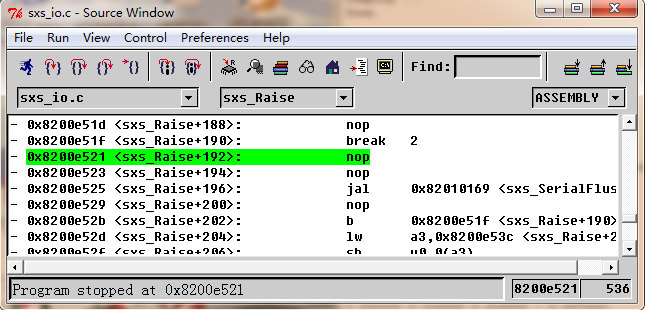


图17

点击图17工具栏上的Console按钮，启动gdb命令窗口：

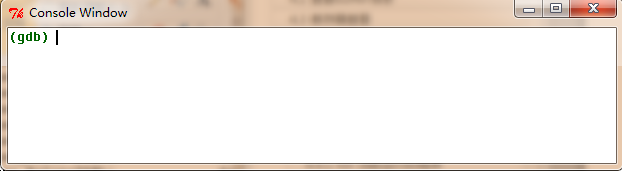


图18

输入gdb命令，即可分析代码。

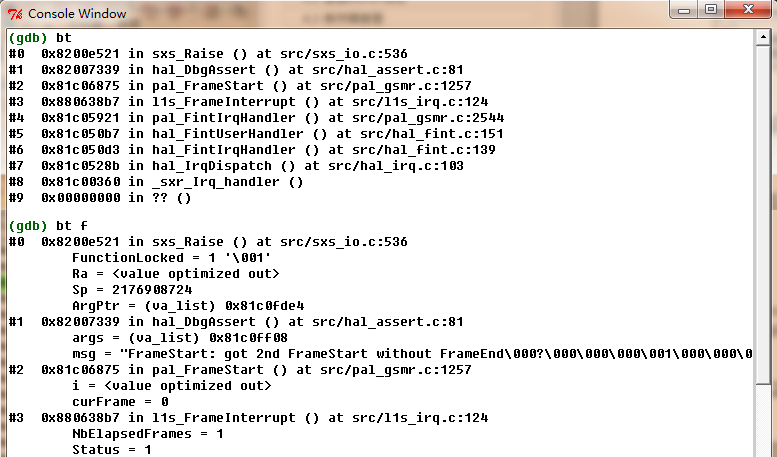


图19

### 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) | Aacktraces (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 |  |
|  |  | conjonction with bt. |  |
| 5 | down | Goes down into the call stack. To be used in |  |
|  |  | conjonction 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。

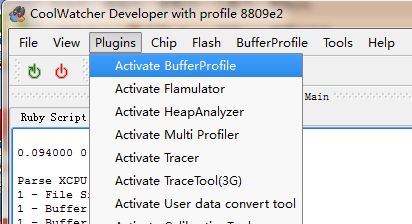


图20

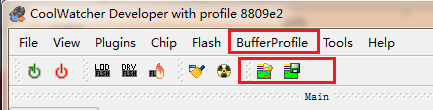


图21

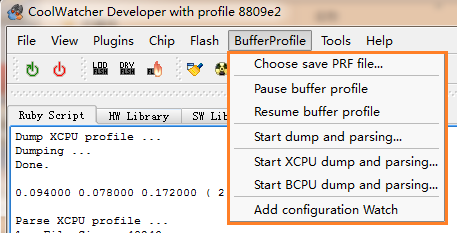
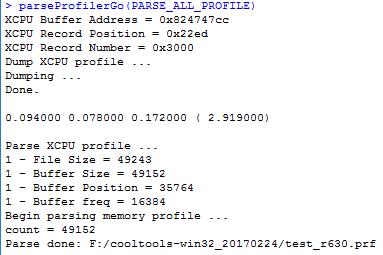


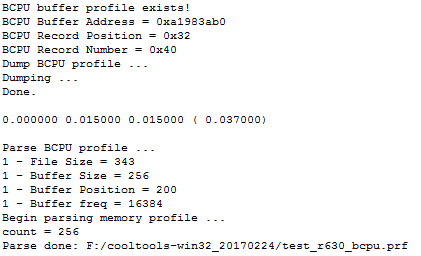
图22

手机死机后，可以通过该插件抓取有关信息。使用方法如下：

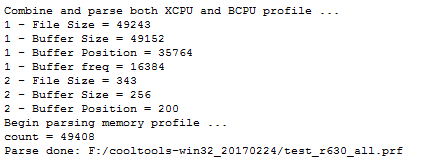
1. 手机通过COM口与PC机相连，保证可以与coolwatcher正常通信。
2. 点击图22 Choose save PRF file菜单项，或工具栏上的按钮，输入要保存的\*.prf文件，文件的默认路径与coolwacher.exe路径一致。
3. 点击图22 Start XCPU dump and parsing菜单，开始保存XCPU相关信息。类似信息如下图：



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



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



**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。



### 4.6.1 分析prf文件

打开coolprofile.exe工具，如下图

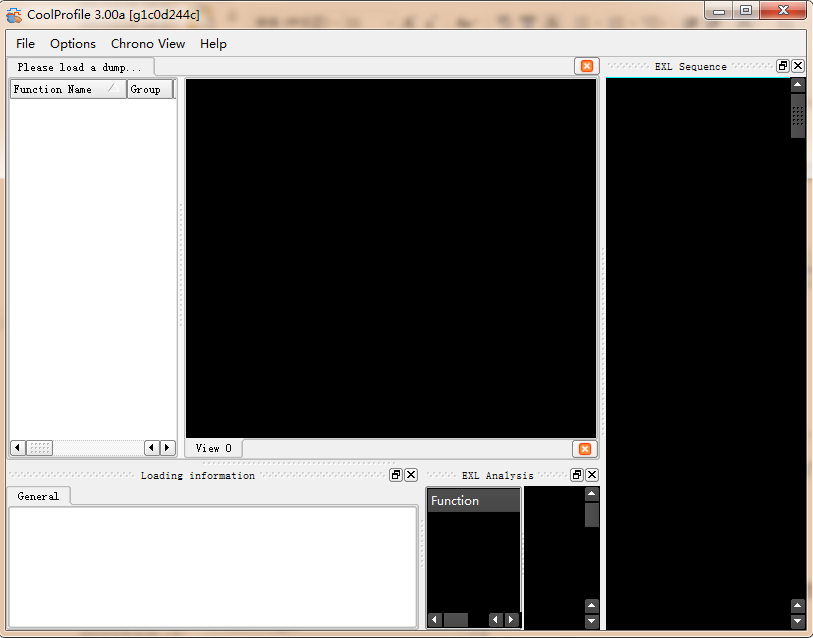


图23

点击file->open按钮，通过下对话框，选择prf文件。

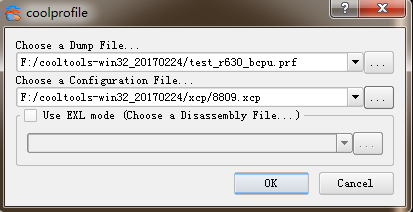


图24

打开效果如下：

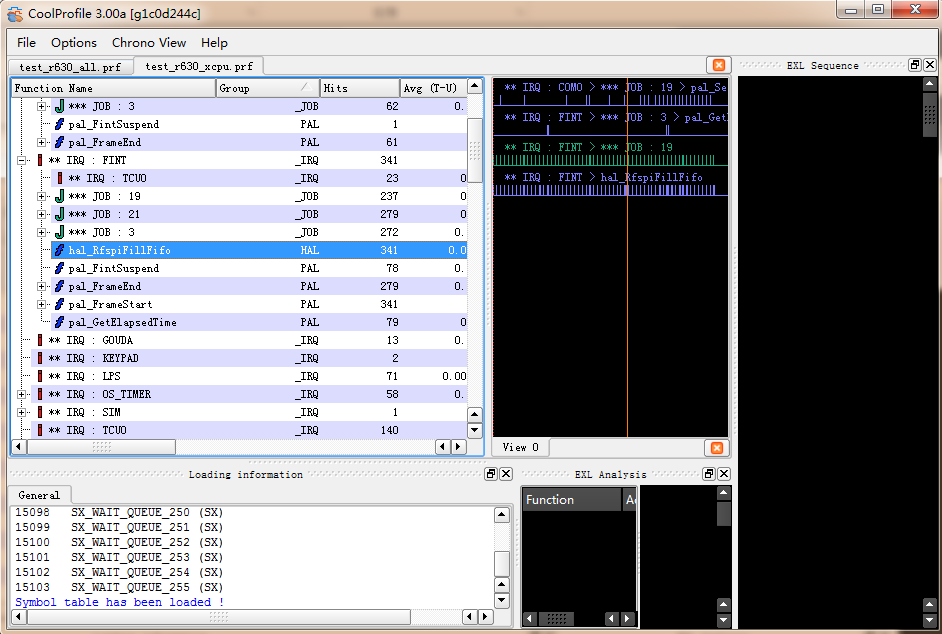


图25

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:

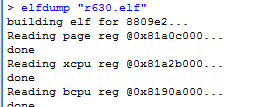


图26

### 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：

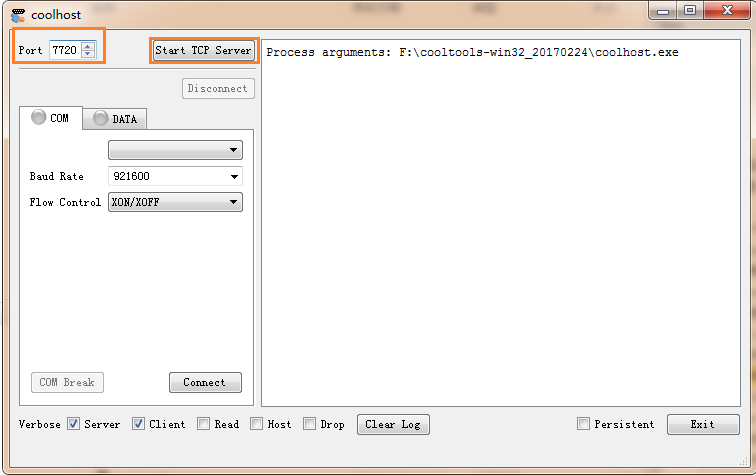


图27

* 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28所示。注意，先选择\*.core（\*.elf）文件。

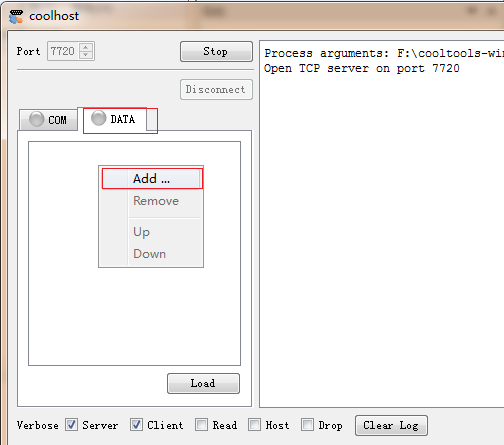


图28

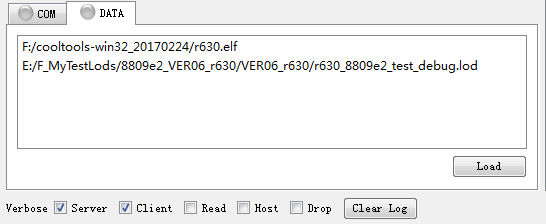


图29

* 加载文件：点击图29的load按钮，加载文件，效果如下图：

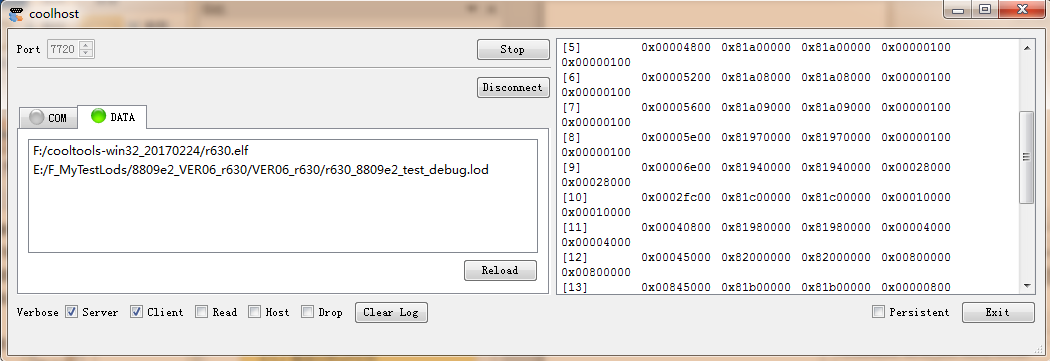
图

图30

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。

1. 启动离线分析环境

启动coolwatcher，下图中的lastcomport项的数值要与图28 Port 的后两位保持一致，如Port选择7720，lastcomport项的值则为20。

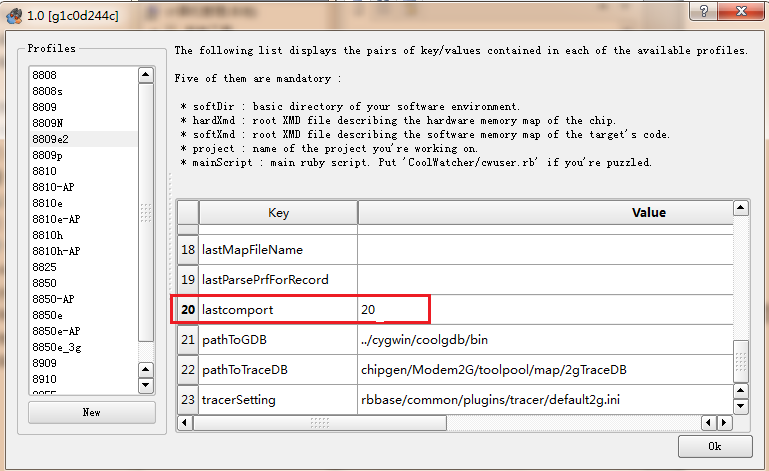


图31

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，安照步骤4.7.2即可离线分析\*.elf文件。

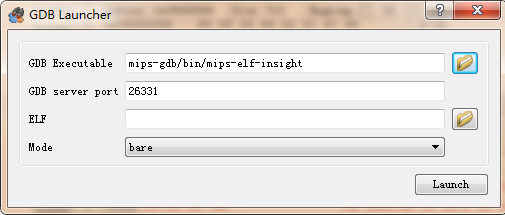


图32

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34：

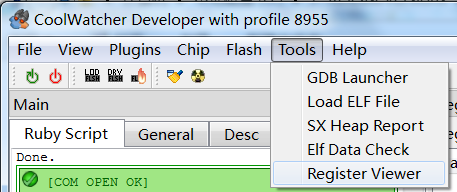


图33

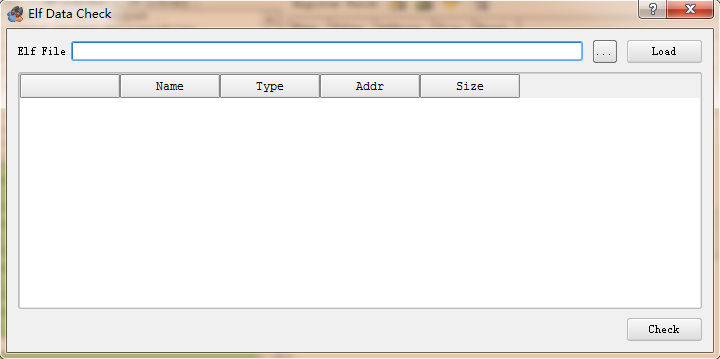


图34

**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:

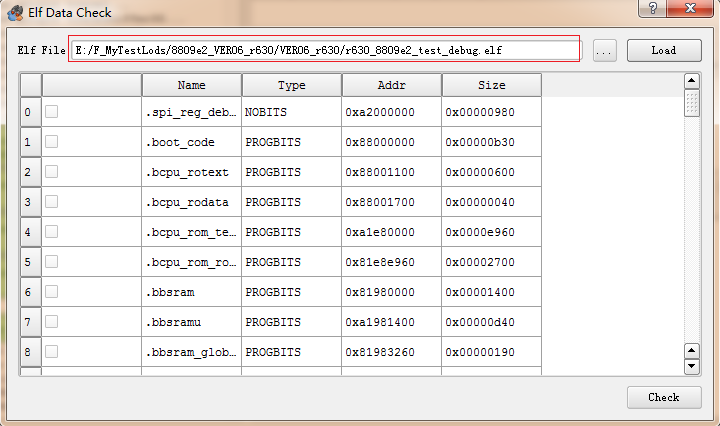


图35

1. 选择待比较的项

通过操作首列check box，选择待比较的项：

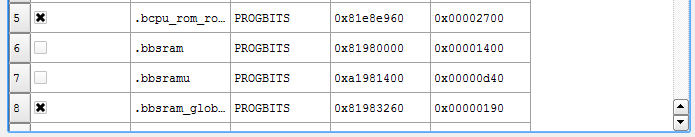


图36

1. 比较和保存结果

点击图35中的check按钮，如果有不同，则会弹出下对话框：

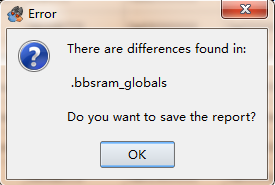


图37

点击OK按钮，选择/填写要保存的文件：

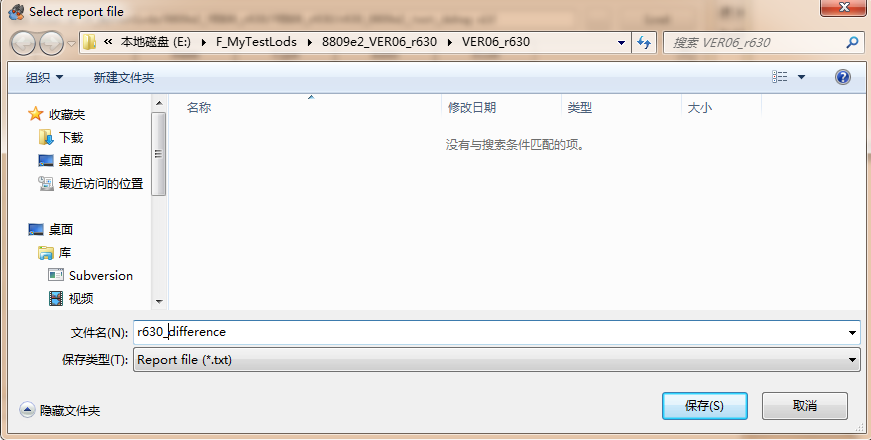


图38

不同处将会保存到文件中，效果如下图：

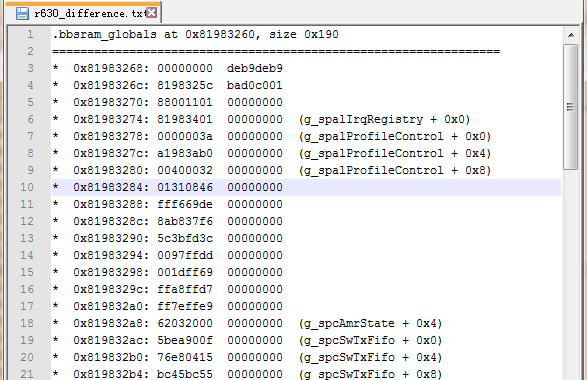


图39

## 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41。

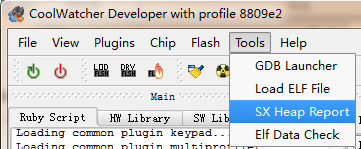


图40

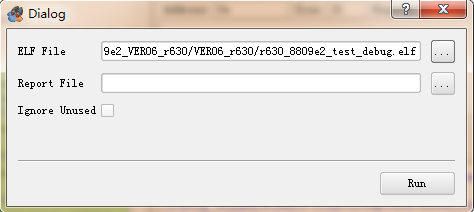


图41

Elf文件：与lod一起发布的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：

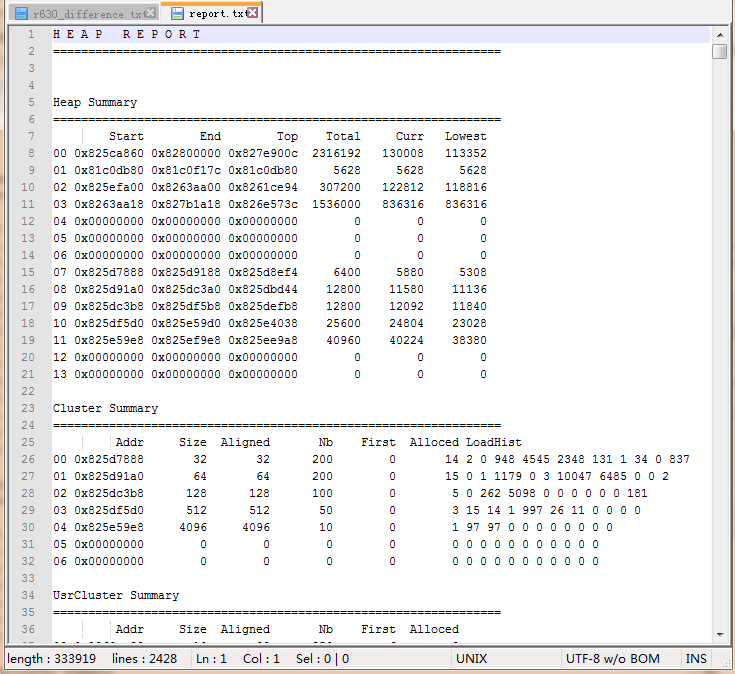


图42

## 4.9 RegisterViewer

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44

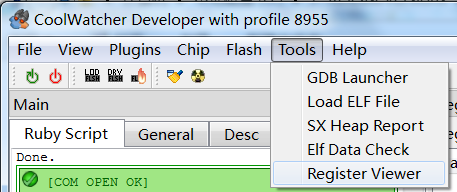


图43

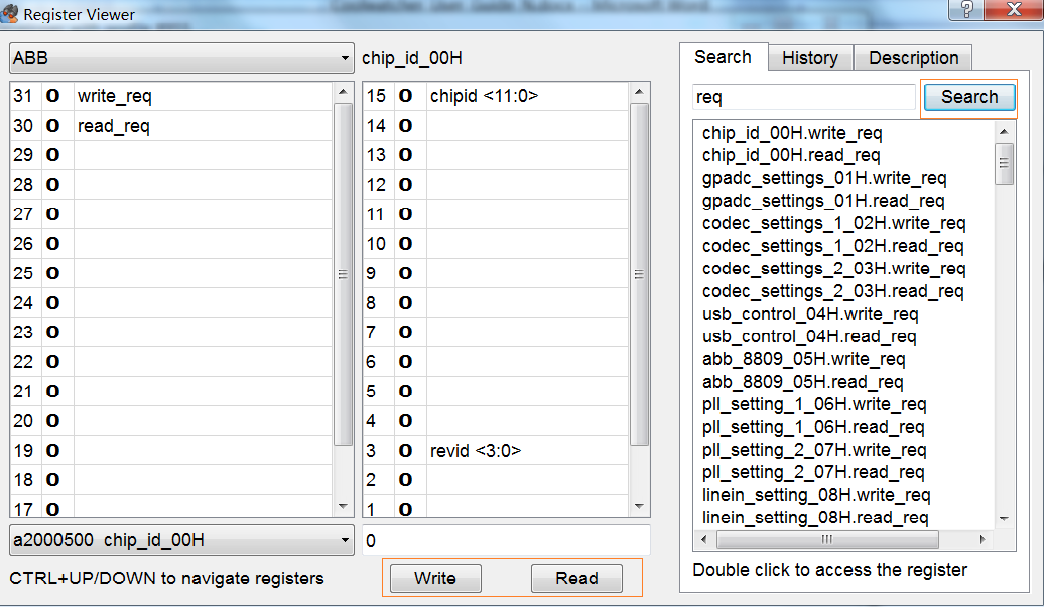


图44

## 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, | fastpf(“c:\xxxx\_ramrun.lod”,” | 烧写 |  |
|  |  | ”c:\\lod”) | Flash |  |
|  | lod\_filename, |  |  |  |
|  | disable\_event\_sniffer = true) |  |  |  |
| fastSectorEraser | (flash\_programmer\_filename, | fastSectorEraser | 擦除扇区 |  |
|  | sector\_list, | (“c:\xxxx\_ramrun.lod”,” |  |  |
|  | [0x01000000,0x01200000] |  |  |
|  | disable\_event\_sniffer = true) |  |  |
|  |  |  |  |
|  |  | ”) |  |  |

### 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 |



### 4.11.3 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 threasd”，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。

## 5 疑难解析

## 6 附录