

Unisoc Confidential For pax

SysDump简介

WWW.UNISOC.COM

版本号	日期	注释
V2.1	2023/08/03	1. 增加Android 14平台相关操作说明 2. 删除T32 sim工具分析MiniDump日志的方法
V2.0	2022/12/30	1、增加MiniDump2.0介绍 2、增加Yocto PWS平台MiniDump配置
V1.9	2022/08/15	增加P13：不支持adb命令的平台SysDump功能配置方法
V1.8	2022/06/27	1、增加Android 13平台相关操作说明 2、增加P11：Android平台通过指令配置SysDump方式
V1.7	2021/10/11	P27增加unisoc_parse_dumplog.py脚本获取方法
V1.6	2021/09/14	增加P12：非Android平台SysDump开启、关闭方法
V1.5	2021/08/04	P6增加Dump2SD及MiniDump功能的依赖项说明
V1.4	2021/06/21	增加Android 9相关说明
V1.3	2020/12/29	1、更新P39 bt命令执行后的示例图 2、更新P17 Crash工具的获取路径
V1.2	2020/10/21	P22增加Logel_R9.20.1401_P1及之后版本上新增功能的描述
V1.1	2020/09/06	1、调整文档结构，优化文档描述 2、文档名称由《UNISOC_sysdump_Brief_Introduction》修改为《SysDump简介》
V1.0	2020/06/01	初稿

关键字

SysDump、日志解析、Dump2PC、FullDump、MiniDump

Unisoc Confidential For pax

Unisoc Confidential For pax

目录



- 01 SysDump简介

- 02 SysDump配置

- 03 FullDump文件解析

- 04 Dump2PC功能介绍

- 05 MiniDump文件解析

- 06 Crash工具常见命令介绍

- 07 SysDump常见异常处理

01

SysDump简介

SysDump即Dump system memory，是一种转存储机制，是将发生异常时的内存信息、寄存器信息等有效信息转存为文件，以便于借助分析工具分析问题现场。

在系统发生诸如Kernel crash等异常时，在Kernel中完成flush cache等处理后，重启进入Uboot中完成所有数据的保存，保存过程会有相应屏幕提示，完成后根据屏幕提示重启设备，导出异常数据文件进行分析。

SysDump分为FullDump和MiniDump两个子功能，两者互不影响。

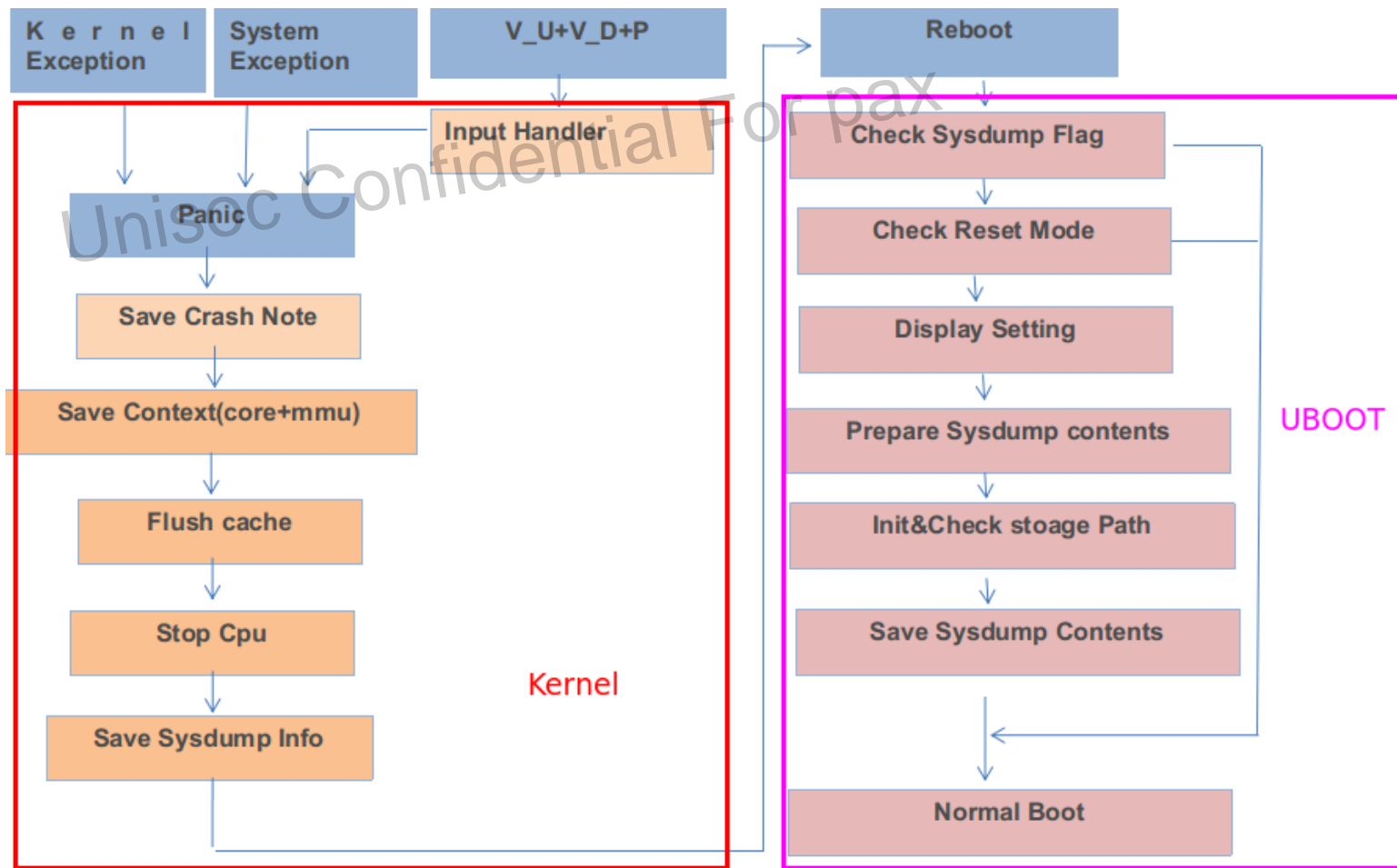
- FullDump保存完整的DDR信息，可以支持两种存储方式：Dump2SD和Dump2PC。

Dump2SD功能依赖于SD卡，即设备需支持插入大于4GB的SD卡。

- MiniDump保存少量信息到单独的sysdumpdb分区，然后再由Native Service程序将分区保存的RAM数据整理解析后保存到指定路径。

仅Android平台设备及Yocto SL8541E PWS设备支持MiniDump功能。

在资源允许的条件下，优先选择保存分析FullDump日志，因为其保存了完整的现场信息快照，更有利于深入分析问题。



- 系统异常和组合键主动触发系统异常都会走到Kernel的处理流程，但长按7s的操作不会进入Kernel处理流程。
- MiniDump在Kernel阶段完成初始化和异常处理时的数据保存。
- Uboot中完成数据的存储操作，包括FullDump和MiniDump。

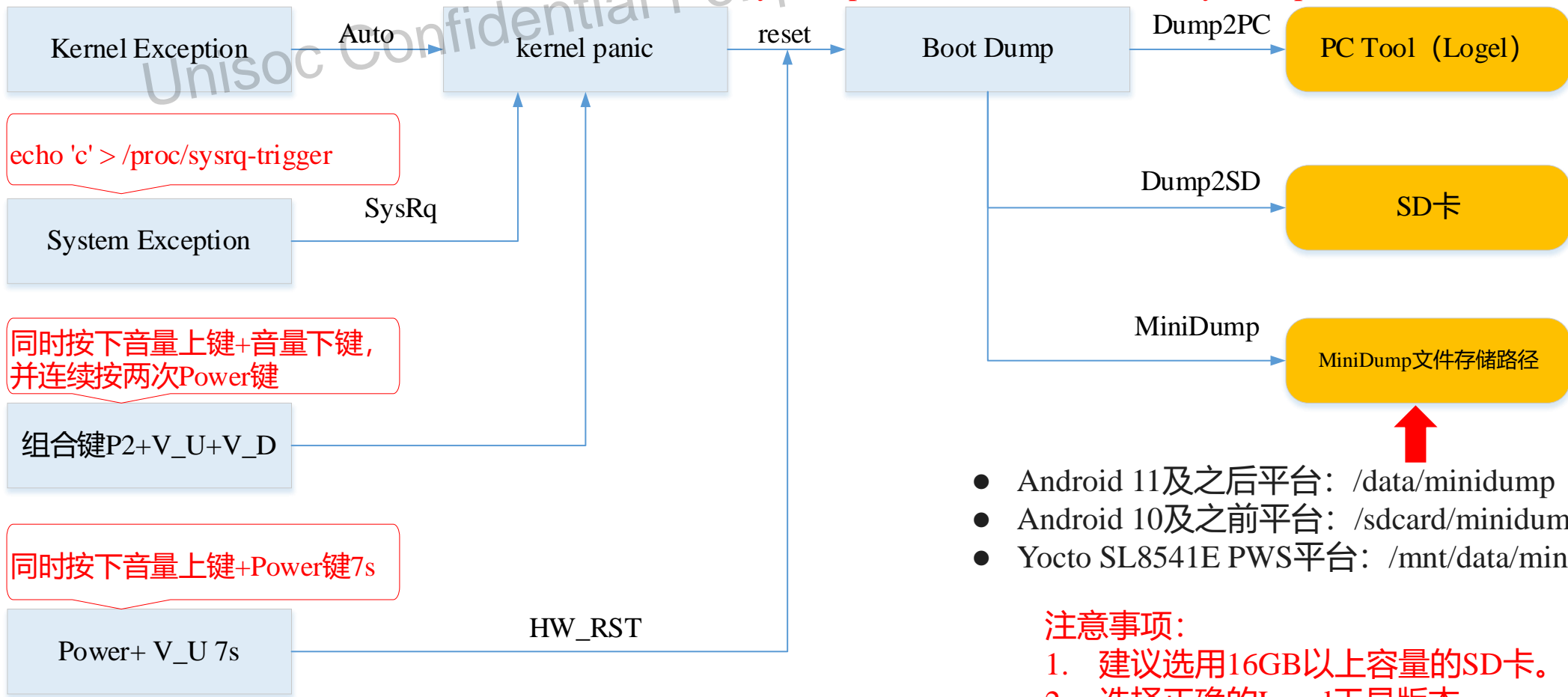
02

SysDump配置

触发方式配置

硬件配置不同，SysDump触发方式也有所差异，具体如下。

Panic函数中嵌入了SysDump的接口，以完成后续的SysDump流程。



- Android 11及之后平台： /data/minidump
- Android 10及之前平台： /sdcard/minidump
- Yocto SL8541E PWS平台： /mnt/data/minidump

注意事项：

1. 建议选用16GB以上容量的SD卡。
2. 选择正确的Logel工具版本。

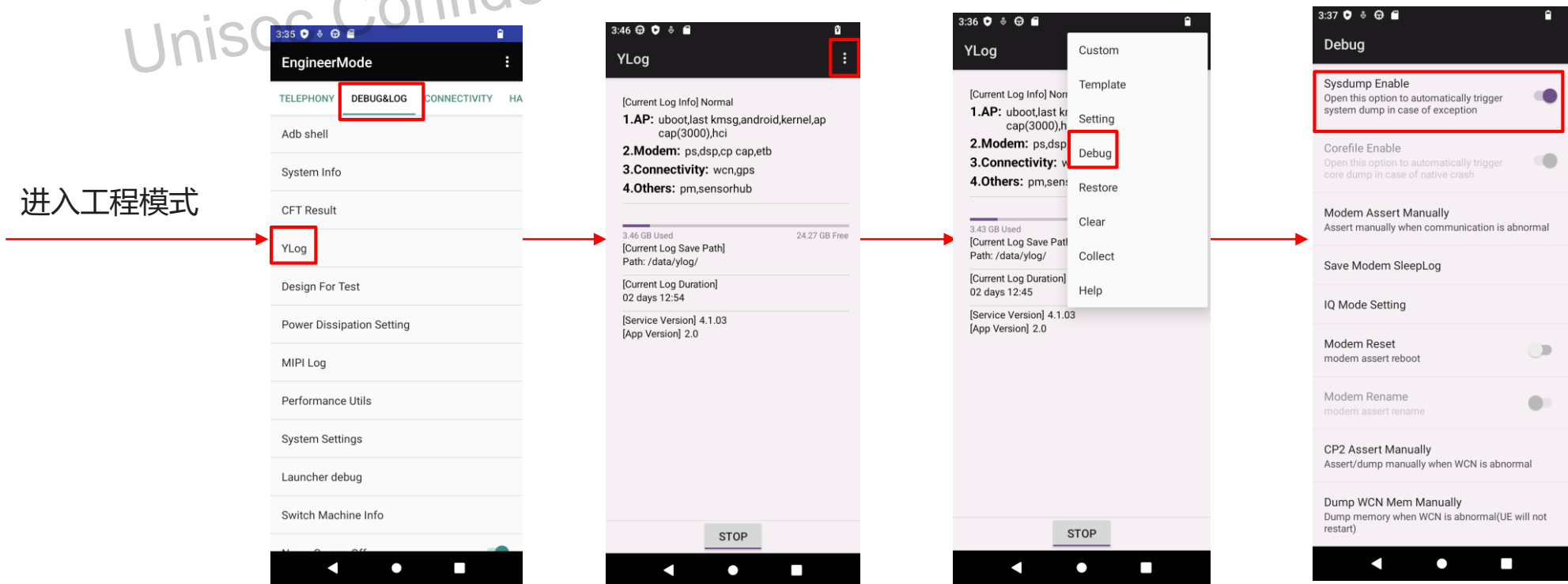
- UIS6780 & UIS6850 & UIS6880组合键方式为：长按遥控器菜单键5s以上。
- UIS6780 & UIS6850 & UIS6880 7s长按方式更改为：长按“按键小板rst键+home键” 7s后，松开rst键。

Android平台功能配置 (1/2)

FullDump功能在UserDebug版本上默认使能，在User版本上默认禁用。MiniDump功能在User和UserDebug版本默认使能。

FullDump功能使能/禁用可以在工程模式中设置，也可以通过指令设置。

- 通过工程模式设置方法：进入工程模式> DEBUG&LOG > Ylog >  > Debug > Sysdump Enable



Android版本不同，版本界面略有差异，具体以实际界面为准。

- ✓ 使能/禁止FullDump操作后立即生效，无需重启设备。
- ✓ 重新开机后，FullDump的状态会保持上一次的使能/禁用状态。

Android平台功能配置 (2/2)

- 通过指令设置

- ✓ 通过fastboot指令设置

- Ubuntu

sudo chmod 777 ./fastboot #为fastboot工具增加执行权限
adb reboot bootloader #进入fastboot模式
./fastboot getdump status #查看SysDump状态
./fastboot setdump full-enable #使能FullDump
./fastboot setdump full-disable #禁用FullDump
./fastboot setdump mini-enable #使能MiniDump
./fastboot setdump mini-disable #禁用MiniDump

- Windows

Windows下使用fastboot.exe工具修改SysDump状态，无需增加权限，方式同fastboot，示例如下。

fastboot.exe getdump status #查看SysDump状态
fastboot.exe setdump full-enable #使能FullDump
fastboot.exe setdump full-disable #禁用FullDump
fastboot.exe setdump mini-enable #使能MiniDump
fastboot.exe setdump mini-disable #禁用MiniDump

如需获取fastboot.exe，请提交CQ。

- ✓ 通过adb指令设置

adb shell setprop debug.vendor.sysdump.enabled true #使能SysDump
adb shell setprop debug.vendor.sysdump.enabled false #禁用SysDump

请根据产品功能支持情况选择SysDump配置方式。

非Android平台功能配置 (1/3)

可使用命令使能或禁用SysDump，在使能或禁用SysDump功能前需先查找是否存在sysdumpdb分区。

注意：非Android平台的设备如无特殊说明，SysDump功能均指FullDump。

检查方法：在分区文件中检查是否有sysdumpdb分区定义，如有类似<Partition id=“**sysdumpdb**” size=“10”/>的定义，表示sysdumpdb分区存在。

如平台支持adb命令，可通过如下方法配置SysDump功能。

- 存在sysdumpdb分区时，使用如下命令：

```
adb reboot bootloader      #进入fastboot模式
./fastboot getdump status   #查看SysDump状态
./fastboot setdump full-enable #使能FullDump
./fastboot setdump full-disable #禁用FullDump
```

- 不存在sysdumpdb分区，使用如下命令：

```
adb shell
echo on > /proc/sprd_sysdump #使能SysDump
echo off > /proc/sprd_sysdump #禁用SysDump
```

User版本7s重启进入SysDump功能生效条件：在使能SysDump的前提下，还需要设置重启模式为软重启，方法如下：

```
cd /hard mode路径 #不同平台的hard mode路径可能不同，可使用find命令搜索hard_mode路径。
echo 0 > hard_mode #将重启模式置为软重启（重启后恢复默认状态）。
```

非Android平台功能配置 (2/3)

如平台不支持adb命令，可通过如下方法配置SysDump功能。

- 存在sysdumpdb分区时，可通过脚本控制SysDump功能，方法如下：
 - 获取脚本文件SYSDUMP_ENABLE_FLASH（可提CQ获取）。
 - 打开ResearchDownload工具：
 - 加载User版本Pac包后，单击Settings。
 - MainPage界面先勾选“Select All Files”项后再去勾选。
 - 切换到Flash Operations页签。
 - 勾选“Active Write Flash”，Base处输入sysdumpdb，File处选择脚本路径。
 - 单击“OK”按钮。

- 不存在sysdumpdb分区，使用如下命令配置：

```
ssh root@192.168.42.1
```

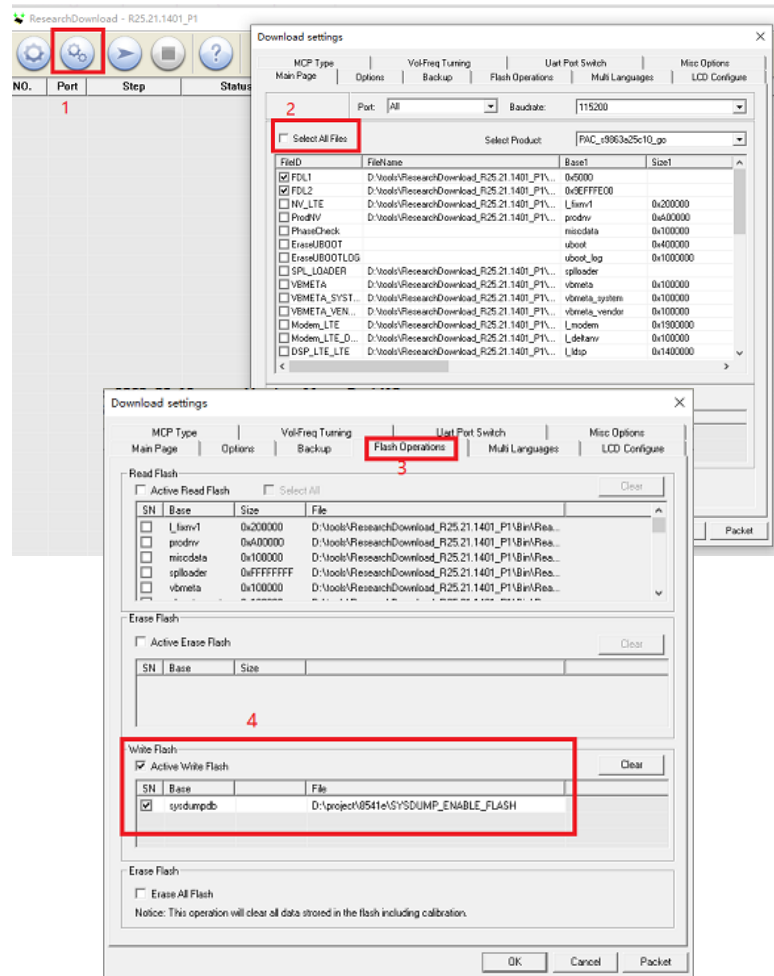
```
echo on > /proc/sprd_sysdump #使能SysDump
```

```
echo off > /proc/sprd_sysdump #禁用SysDump
```

User版本7s重启进入SysDump功能生效条件：在使能SysDump的前提下，还需要设置重启模式为软重启，方法如下：

```
cd /hard mode路径
```

```
echo 0 > hard_mode
```



【特殊说明：有sysdumpdb分区的情况下】

除上述方式外，SL8541E PWS项目还可通过如下命令配置FullDump功能。

```
ssh root@192.168.42.1
```

```
/usr/bin/systemdebuggerd get_dump_status //查看SysDump使能状态
```

```
/usr/bin/systemdebuggerd full-enable //使能FullDump
```

```
/usr/bin/systemdebuggerd full-disable //禁用FullDump
```

SL8541E PWS项目支持MiniDump功能，可通过如下命令配置MiniDump功能。

```
ssh root@192.168.42.1
```

```
/usr/bin/systemdebuggerd get_dump_status //查看SysDump使能状态
```

```
/usr/bin/systemdebuggerd mini-enable //使能MiniDump
```

```
/usr/bin/systemdebuggerd mini-disable //禁用MiniDump
```


03

FullDump文件 解析

FullDump文件解析 (1/2)

- Dump2SD方式存储的文件在SD卡中，有SysDump文件夹，保存3份历史日志，其中文件夹1为最新的。
- Dump2PC方式存储的文件可通过Logel工具查看，方法：File > Open Log Location。

SysDump文件夹主要信息如下：

- dump_report.txt: SysDump信息文件，记录了SysDump文件的个数，重启原因等。
- ylog_buf文件：记录ylog buffer中信息。
- sysdump.core*: memory dump文件。
- extend_*文件：其他debug信息，如log_buf、per_cpu段等。
- bootloader_*_log: bootloader log, last为最新的，pre为上一次的。

不同的项目，不同的操作系统的SysDump文件夹内容略有差异，具体以实际展示为准。

bootloader_last_log	2022/4/11 16:32	文件	512 KB
bootloader_pre_log	2022/4/11 16:32	文件	256 KB
dump_report	2022/4/11 16:30	文本文档	2 KB
etbdata_uboot.bin	2022/4/11 16:32	BIN 文件	32 KB
extend_cpustack1	2022/4/11 16:32	文件	8 KB
extend_etb_data	2022/4/11 16:32	文件	32 KB
extend_interruptes	2022/4/11 16:32	文件	12 KB
extend_io	2022/4/11 16:32	文件	36 KB
extend_kernel_pt	2022/4/11 16:32	文件	16 KB
extend_log_buf	2022/4/11 16:32	文件	1,024 KB
extend_memstat	2022/4/11 16:32	文件	12 KB
extend_nhang	2022/4/11 16:32	文件	1,024 KB
extend_per_cpu	2022/4/11 16:32	文件	672 KB
extend_runqueue	2022/4/11 16:32	文件	16 KB
extend_sprd_log_buf0	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf1	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf2	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf3	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf4	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf5	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf6	2022/4/11 16:32	文件	32 KB
extend_sprd_log_buf7	2022/4/11 16:32	文件	32 KB
extend_stack_regs	2022/4/11 16:32	文件	16 KB
extend_task_stats	2022/4/11 16:32	文件	470 KB
extend_vmcoreinfo	2022/4/11 16:32	文件	8 KB
extend_ylog_buf	2022/4/11 16:32	文件	1,024 KB
sysdump.core.01_0x80000000-0xbffff...	2022/4/11 16:31	LST 文件	1,048,576...
sysdump.core.02_0xc0000000-0xbffff...	2022/4/11 16:32	LST 文件	1,048,320...

FullDump文件解析命令如下:

- 32bit Arm®平台
 - ✓ Android 13、Android 14: `./crash_arm -m phys_base=0x80000000 vmlinux vmcore@0x80000000`
 - ✓ 其他: `./crash_arm -m phys_base=0x80000000 vmlinux vmcore`
- 64bit Arm®平台
 - ✓ Android 14: `./crash_arm64 -m vabits_actual=39 -m phys_offset=0x80000000 -m kimage_voffset=0xffffffffbf88000000 vmcore@0x80000000 vmlinux --kaslr 0x80000`
 - ✓ Android 13: `./crash_arm64 -m vabits_actual=39 -m phys_offset=0x80000000 -m kimage_voffset=0xffffffffbf90000000 vmcore@0x80000000 vmlinux`
 - ✓ Android 12: `./crash_arm64 -m vabits_actual=39 -m phys_offset=0x80000000 -m kimage_voffset=0xffffffffbf90000000 vmcore vmlinux`
 - ✓ 其他: `./crash_arm64 -m phys_offset=0x80000000 vmlinux vmcore`

其中:

- Android 12及以上平台Crash工具需使用7.2.8及以上版本, Android 13和Android 14平台推荐使用crash8.0.0版本
Crash工具请在uni-support工具管理模块下获取, 可搜索关键字crash_utility。
- vmlinux为编译时生成的最原始的内核文件, 用于Kernel调试, 获取路径为: `out/target/product/xxxx/obj/Kernel/vmlinux`
- vmcore为通过SysDump收集到的系统的core dump信息, 由SysDump文件sysdump.core*合成, 合成命令如下:
`cat sysdump.core.* > vmcore`

04 Dump2PC功能 介绍

功能简介

Dump2PC功能是为了适应无SD卡场景时，将SysDump产生的日志借助Logel工具导出到PC上。

该功能默认：

- 有SD卡时将log存储到SD卡。
- 无SD卡或者Dump2SD失败时，自动切换到Dump2PC功能，提示连接PC进行Dump2PC操作。

Dump2PC工具和SPRD U2S DIAG端口驱动建议使用最新版本。

1. 支持Dump2PC功能的设备发生SysDump, 在Dump2SD失败后会自动尝试Dump2PC, 屏幕会给出打印信息, 等待连接PC。

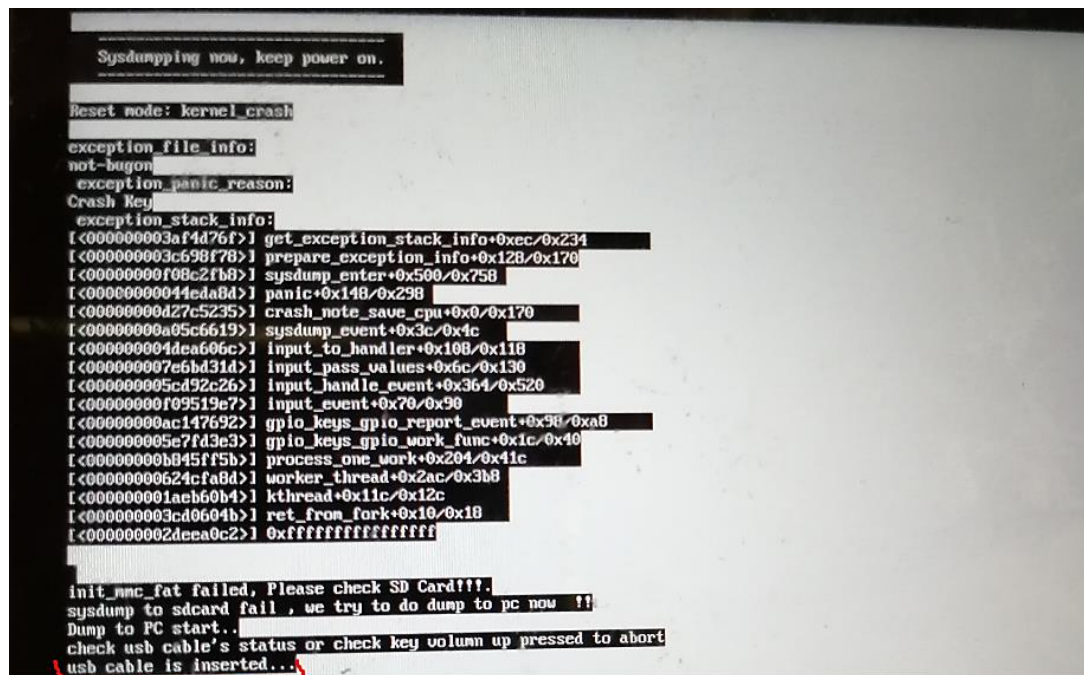
此时并未开始dump, 只有连接成功后设备才会和PC开始握手。

注意: Android 13及之后平台, Dump2SD卡失败后, 屏幕无 “sysdump to sdcard fail” 提示。

使用USB线连接设备和PC, 此时如果设备可以正常检测到USB线插入, 则会在屏幕上有 “usb cable is inserted...” 的打印提示。

测试前如未连接充电器, 此处会一直等待直到电量耗尽; 如连接充电器, 此状态可以充电, 保证电量不会耗尽。

注意: 此处必须有USB插入动作才会被检测到, 如测试前已插入USB, 则需重新进行拔插操作。



```
Sysdumping now, keep power on.

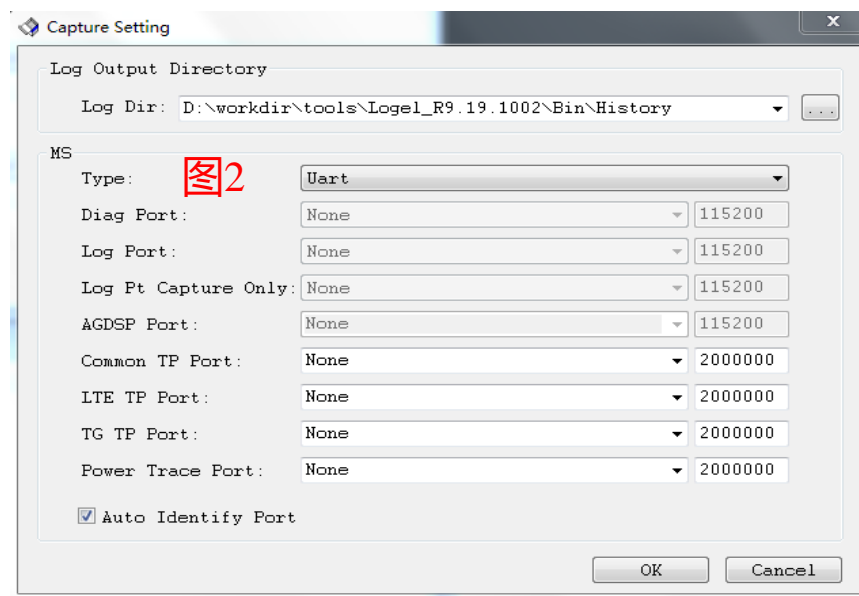
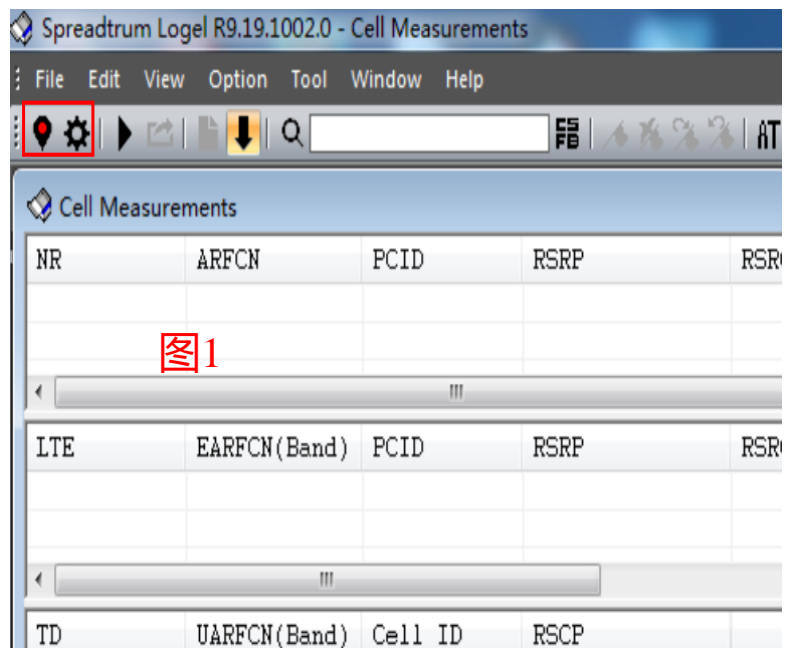
Reset mode: kernel_crash
exception file info:
not-bugon
exception_panic_reason:
Crash Key
exception_stack_info:
[<000000003af4d76f>] get_exception_stack_info+0xec/0x234
[<000000003c698f78>] prepare_exception_info+0x128/0x170
[<00000000f08c2fb8>] sysdump_enter+0x500/0x758
[<00000000044eda8d>] panic+0x148/0x298
[<00000000d27c5235>] crash_note_save_cpu+0x0/0x170
[<00000000a05c6619>] sysdump_event+0x3c/0x4c
[<000000004dea606c>] input_to_handler+0x108/0x118
[<000000007e6bd31d>] input_pass_values+0x6c/0x130
[<000000005cd92c26>] input_handle_event+0x364/0x520
[<00000000f09519e7>] input_event+0x70/0x90
[<00000000ac147692>] gpio_keys_gpio_report_event+0x38/0xa8
[<000000005e7fd3e3>] gpio_keys_gpio_work_func+0x1c/0x40
[<00000000b845ff5b>] process_one_work+0x204/0x41c
[<00000000624cfa8d>] worker_thread+0x2ac/0x3b8
[<000000001acb60b4>] kthread+0x11c/0x12c
[<000000003cd0604b>] ret_from_fork+0x10/0x18
[<000000002deea0c2>] 0xffffffffffffffff

init_mmc_fat failed, Please check SD Card!!!
sysdump to sdcard fail , we try to do dump to pc now !!
Dump to PC start..
check usb cable's status or check key volumn up pressed to abort
usb cable is inserted...
```


2. 打开PC端工具Logel (版本必须高于R8.18.1702_P2), Logel工具界面如图1所示。

其中, 红框内:

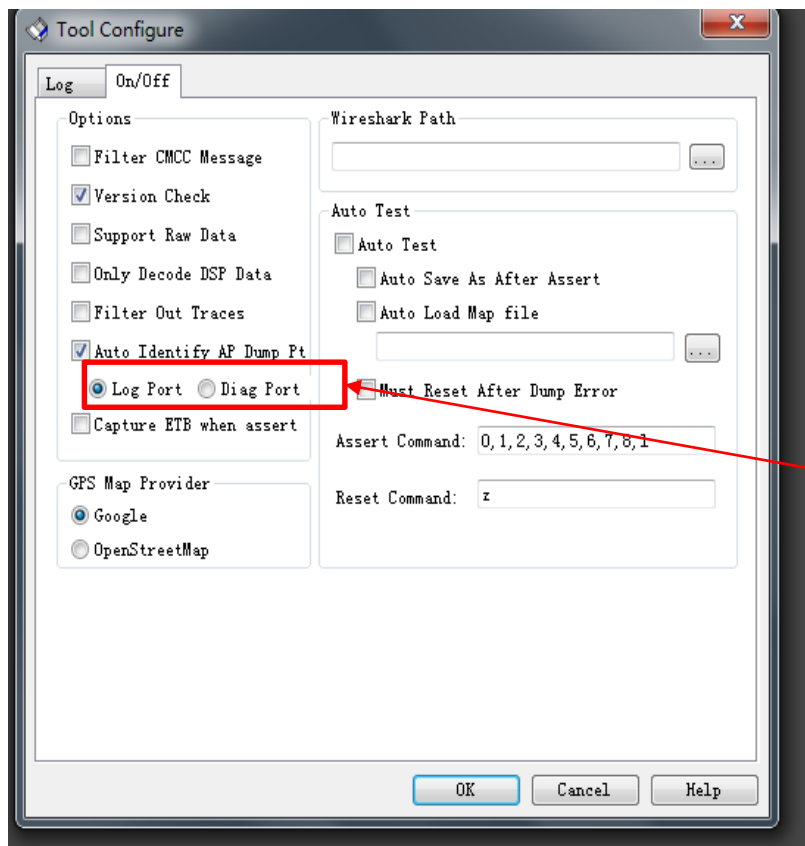
- 左侧为capture按钮, 初始状态为红色, 抓取状态为绿色。
- 右侧为capture setting按钮, 默认无需做任何设置, 如图2所示。



3. AP Dump端口自动识别设置

Logel工具 (Logel_R9.19.1002_P1之后) 默认将AP Dump端口自动识别功能关闭，因为此端口与下载端口/校准端口名称一样，当同一台PC上同时使用Pandora/Simba/ResearchDownload时就会互相抢占此端口，所以工具默认关闭了自动识别的功能。

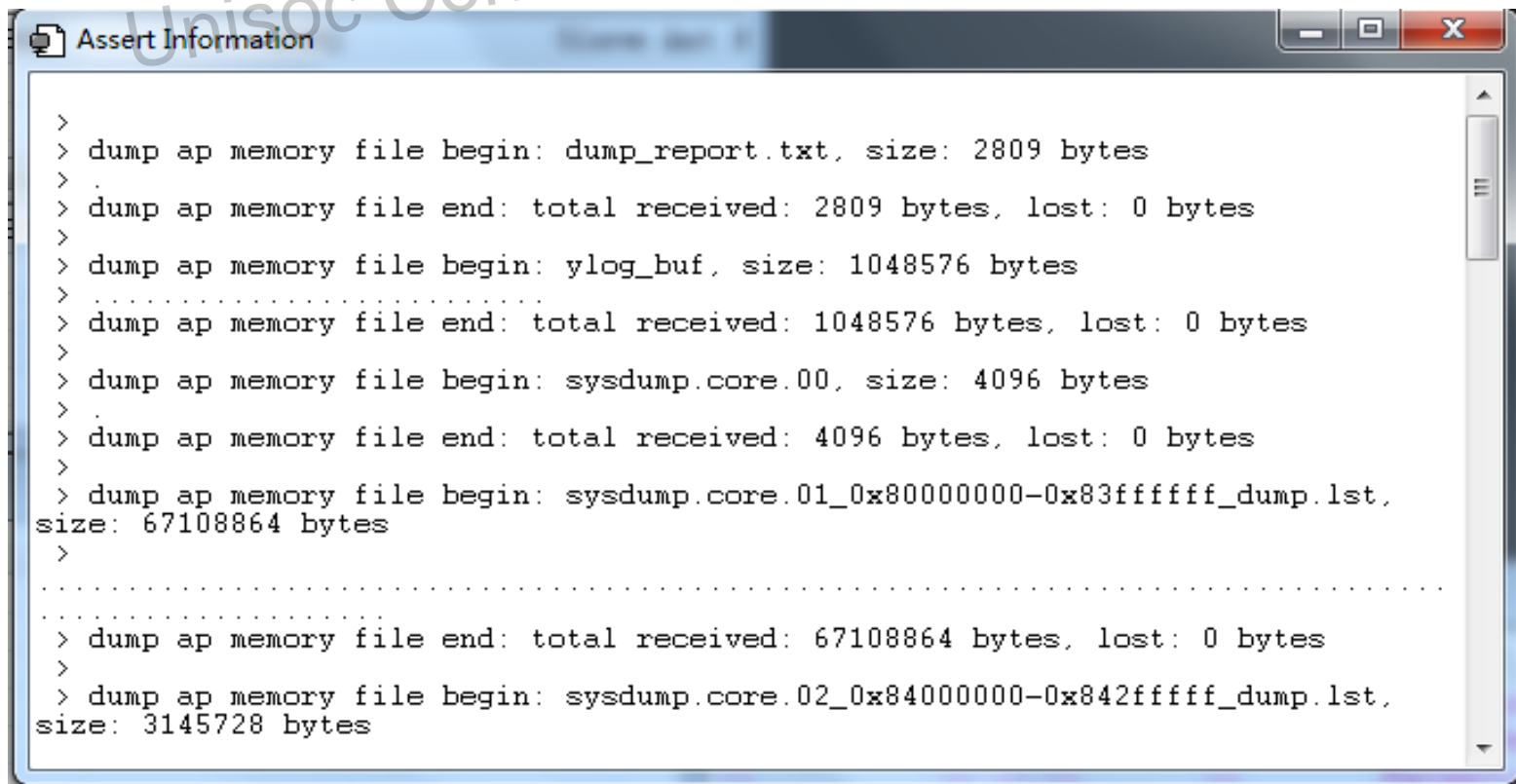
可通过点击工具的菜单Option > Tool Configure，将Auto Identity AP Dump Pt勾上打开自动识别功能，如下图所示。



Logel_R9.20.1401_P1及之后版本上新增功能，之前版本无此选项。
Dump2PC时需要勾选Log Port。

4. Logel工具和设备握手成功，PC端工具会自动弹出数据框，并开始导出日志，如下图。

注：自动弹出是因为之前使用过相同端口导出过，若未使用过可能需要单击capture按钮。



The screenshot shows a window titled "Assert Information" with a text area containing the following log output:

```
>
> dump ap memory file begin: dump_report.txt, size: 2809 bytes
> .
> dump ap memory file end: total received: 2809 bytes, lost: 0 bytes
>
> dump ap memory file begin: ylog_buf, size: 1048576 bytes
> .....
> dump ap memory file end: total received: 1048576 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.00, size: 4096 bytes
> .
> dump ap memory file end: total received: 4096 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.01_0x80000000-0x83ffffff_dump.lst,
size: 67108864 bytes
>
> .....
> dump ap memory file end: total received: 67108864 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.02_0x84000000-0x842ffffff_dump.lst,
size: 3145728 bytes
```

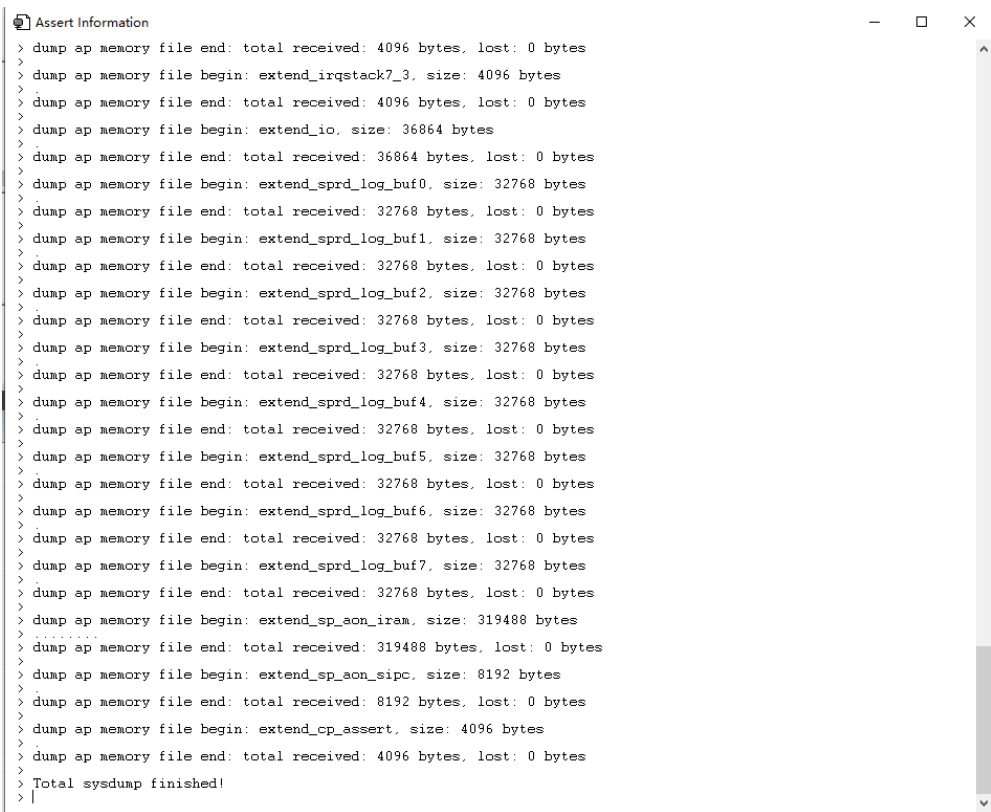
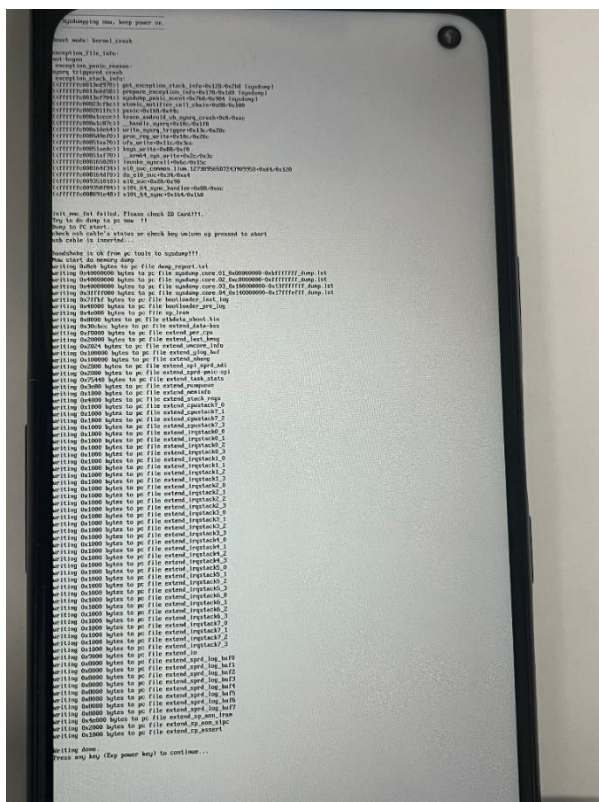
使用说明 (5/6)

5. Dump2PC导出数据日志完成确认。

dump完成后，设备提示“Press any key(Exp power key) to continue...”，PC端也会显示“Total sysdump finished!”。

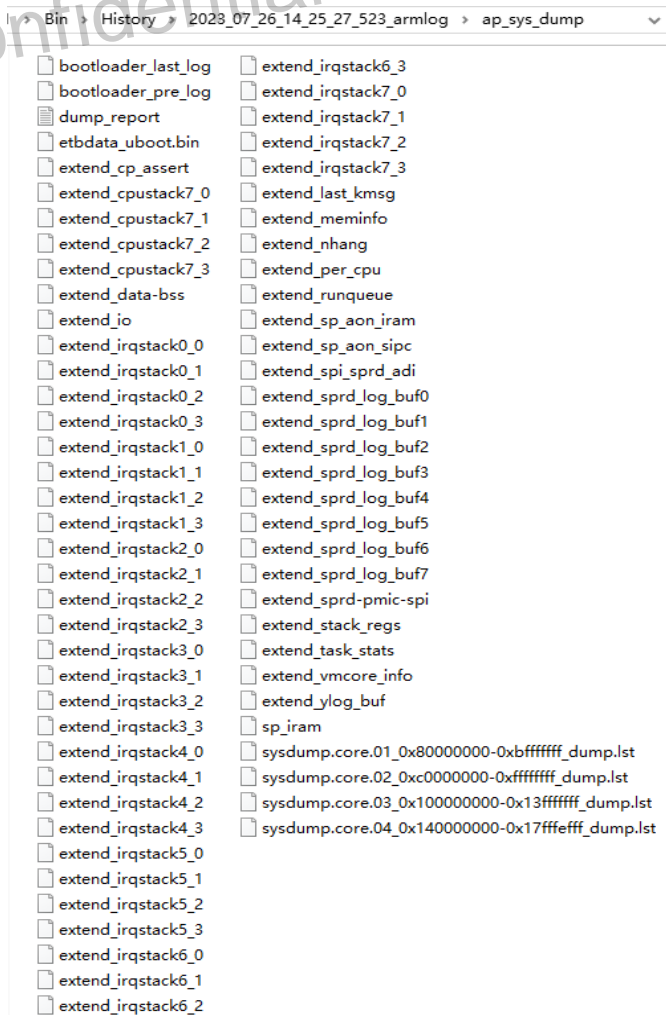
此时按照设备屏幕提示单击任意键（如音量上键）重启设备，整个dump过程结束。

注意：Dump2PC完成后，UIS6780、UIS6880、UIS6850平台会自动重启，无需单击按键。



6. SysDump日志文件检查。

SysDump日志文件存储路径：*Logel工具的解压根目录*/Bin/History/*当前时间*_armlog/ap_sys_dump，示例详见下图。



05 MiniDump文件 解析

MiniDump功能常用于无SD卡或无法Dump2PC的场景，平台默认使能。

- MiniDump功能支持保存5份历史数据文件，对应的文件夹名称分别为1、2、3、4、5，用于存储MiniDump的压缩数据。文件夹1永远为最新生成的数据，再次发生MiniDump时，将文件夹1命名为2，并重新创建文件夹1，超过5个文件夹时，原有的文件夹5丢弃，以此类推。
- MiniDump日志可以使用adb或ssh命令直接导出，MiniDump日志默认保存路径如下：
 - ✓ Android 11及之后平台：默认保存在/data/minidump目录下。
 - ✓ Android 10及之前平台：默认保存在/sdcard/minidump目录下。
 - ✓ Yocto SL8541E PWS平台：默认保存在/mnt/data/minidump目录下。

若设备无法开机，可以使用下载工具回读sysdumpdb分区数据，再使用sysdumpdb_parse解析工具将裸分区数据解析成Minidump文件。Android13及之后平台升级过minidump2.0的版本，该工具已随代码发布。

工具路径：vendor/sprd/modules/minidump/scripts

解析命令：./sysdumpdb_parse mini *回读文件名* //Linux环境下使用

- MiniDump日志使用Crash工具进行分析。

目前UIS6850 & UIS6880 & UIS6780平台以及搭载Android 13及之后操作系统的平台进一步优化了MiniDump功能，升级为MiniDump 2.0，MiniDump 2.0将保存的内容分类成了四个部分：bootloader、kernel、modem、trusty。MiniDump2.0覆盖的场景更广，保存的内容更多，解压解析更方便。同时，Minidump2.0实现了底层和上层的解耦，增强了MiniDump的可扩展性。

MiniDump1.0日志文件介绍

MiniDump1.0生成的日志文件如下图所示：

其中：

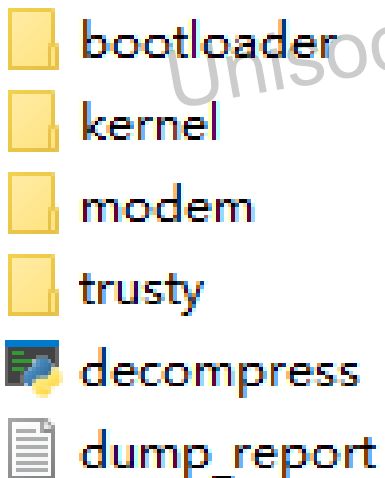
00_minidump_elfhdr
01_minidump_regs
02_minidump_regs_R0
03_minidump_regs_R1
04_minidump_regs_R2
05_minidump_regs_R3
06_minidump_regs_R4
07_minidump_regs_R5
08_minidump_regs_R6
09_minidump_regs_R7
10_minidump_regs_R8
11_minidump_regs_R9
12_minidump_regs_R10
13_minidump_regs_R11
14_minidump_regs_R12
15_minidump_regs_R13
16_minidump_regs_R14
17_minidump_regs_R15
18_minidump_section_data
19_minidump_section_bss
20_minidump_section_init
21_minidump_section_inittext
22_minidump_section_rodata
23_minidump_section_extend_log_buf
24_minidump_section_extend_per_cpu
25_minidump_section_extend_etb_data
26_minidump_section_extend_vmcoreinfo
27_minidump_section_extend_spr_log_buf0
28_minidump_section_extend_spr_log_buf1
29_minidump_section_extend_spr_log_buf2
30_minidump_section_extend_spr_log_buf3
31_minidump_section_extend_spr_log_buf4
32_minidump_section_extend_spr_log_buf5
33_minidump_section_extend_spr_log_buf6
34_minidump_section_extend_spr_log_buf7
35_minidump_section_extend_vlog_buf
36_minidump_section_extend_kernel_pt
37_minidump_section_extend_nhang
38_minidump_section_extend_task_stats
39_minidump_section_extend_runqueue
40_minidump_section_extend_stack_regs
41_minidump_section_extend_memstat
42_minidump_section_extend_interrupts
43_minidump_section_extend_cpustack1
44_minidump_section_extend_io
45_minidump_section_mini_3_bootloader_last_log
46_minidump_section_mini_3_bootloader_pre_log
47_minidump_section_mini_5_mini_mem_info
compressed_info
dump_report
last_kmsg

- 00_minidump_elfhdr：elf文件头（Android 13和Android 14由脚本生成）。
- 01_minidump_regs：CPU寄存器。
- 02 ~ 17：寄存器前后各一页信息。
- 18 ~ 22：各段信息。
- 23 ~ 44：其他debug信息。如：irqstack, cpustack, io, sprd_log_buf, mini_mem_info, log_buf, per_cpu段等。
- 45 ~ 46：bootloader log, last为最新的, pre为前一次的。
- 47：生成elf头文件的脚本所需的数据。
- compressed_info：压缩信息，bootloader为LK时特有，使用脚本对MiniDump文件解压缩时使用。
- dump_report：dump描述信息。
- last_kmsg：系统重启之前最后的内核log信息，由脚本解析生成。

不同的项目、不同的操作系统，MiniDump文件略有差异，具体以实际展示为准。

MiniDump2.0日志文件介绍 (1/4)





MiniDump2.0日志保存内容包括6部分，其中：



- bootloader：保存bootloader阶段添加到MiniDump的section。
- kernel：保存kernel阶段添加到MiniDump的section。
- modem：保存CM4的IRAM。
- trusty：保存安全相关的memory，如tos_mem、sml_mem。
- decompress：解压脚本，发生异常后，该脚本会同步保存到MiniDump日志中，获取MiniDump日志文件后，可直接双击该脚本解压MiniDump日志文件。
- dump_report：dump描述信息，包括重启原因、异常原因、调用栈等。

MiniDump2.0日志文件介绍 (2/4)

bootloader文件夹内容

 `bootloader_last_log`
 `bootloader_mem`
 `compress_record_file`
 `etb_data`

- `bootloader_last_log`: bootloader log。
- `bootloader_mem`: 发生bootloader panic时保存的bootloader memory。
- `compress_record_file`: 记录压缩信息相关的文件，解压缩时使用。
- `etb_data`: SoC dump保存的etb数据。

MiniDump2.0日志文件介绍 (3/4)

kernel文件夹内容

001_minidump_regs
002_minidump_regs_memory_4
003_minidump_regs_memory_5
004_minidump_regs_memory_8
005_minidump_regs_memory_11
006_minidump_regs_memory_15
007_minidump_regs_memory_17
008_minidump_regs_memory_19
009_minidump_regs_memory_20
010_minidump_regs_memory_23
011_minidump_regs_memory_24
012_minidump_regs_memory_25
013_minidump_regs_memory_27
014_minidump_extend_data-bss
015_minidump_extend_per_cpu
016_minidump_extend_last_kmsg
017_minidump_extend_vmcore_info
018_minidump_extend_ylog_buf
019_minidump_extend_task_stats
020_minidump_extend_runqueue
021_minidump_extend_meminfo
022_minidump_extend_stack_regs
023_minidump_extend_cpustack0_0
024_minidump_extend_cpustack0_1
025_minidump_extend_cpustack0_2
026_minidump_extend_cpustack0_3
027_minidump_extend_cpustack1_0
028_minidump_extend_cpustack1_1
029_minidump_extend_cpustack1_2
030_minidump_extend_cpustack1_3
031_minidump_extend_cpustack2_0
032_minidump_extend_cpustack2_1
033_minidump_extend_cpustack2_2
034_minidump_extend_cpustack2_3
035_minidump_extend_cpustack3_0
036_minidump_extend_cpustack3_1
037_minidump_extend_cpustack3_2
038_minidump_extend_cpustack3_3
039_minidump_extend_cpustack4_0
040_minidump_extend_cpustack4_1
041_minidump_extend_cpustack4_2
042_minidump_extend_cpustack4_3
043_minidump_extend_cpustack5_0
044_minidump_extend_cpustack5_1
045_minidump_extend_cpustack5_2
046_minidump_extend_cpustack5_3
047_minidump_extend_cpustack7_0
048_minidump_extend_cpustack7_1
049_minidump_extend_cpustack7_2
050_minidump_extend_cpustack7_3
051_minidump_extend_irqstack0_0
052_minidump_extend_irqstack0_1
053_minidump_extend_irqstack0_2
054_minidump_extend_irqstack0_3
055_minidump_extend_irqstack1_0
056_minidump_extend_irqstack1_1
057_minidump_extend_irqstack1_2
058_minidump_extend_irqstack1_3
059_minidump_extend_irqstack2_0
060_minidump_extend_irqstack2_1
061_minidump_extend_irqstack2_2
062_minidump_extend_irqstack2_3
063_minidump_extend_irqstack3_0
064_minidump_extend_irqstack3_1
065_minidump_extend_irqstack3_2
066_minidump_extend_irqstack3_3
067_minidump_extend_irqstack4_0
068_minidump_extend_irqstack4_1
069_minidump_extend_irqstack4_2
070_minidump_extend_irqstack4_3
071_minidump_extend_irqstack5_0
072_minidump_extend_irqstack5_1
073_minidump_extend_irqstack5_2
074_minidump_extend_irqstack5_3
075_minidump_extend_irqstack6_0
076_minidump_extend_irqstack6_1
077_minidump_extend_irqstack6_2
078_minidump_extend_irqstack6_3
079_minidump_extend_irqstack7_0
080_minidump_extend_irqstack7_1
081_minidump_extend_irqstack7_2
082_minidump_extend_irqstack7_3
083_minidump_extend_nhang
084_minidump_extend_io
085_minidump_extend_sprd_log_buf0
086_minidump_extend_sprd_log_buf1
087_minidump_extend_sprd_log_buf2
088_minidump_extend_sprd_log_buf3
089_minidump_extend_sprd_log_buf4
090_minidump_extend_sprd_log_buf5
091_minidump_extend_sprd_log_buf6
092_minidump_extend_sprd_log_buf7
093_minidump_extend_ch_aon_s
094_minidump_extend_ch_iram
095_minidump_extend_sp_aon_s
096_minidump_extend_sp_iram
097_minidump_extend_ps_ch_aon_s
098_minidump_kernel_mem_info

- 001_minidump_regs: CPU寄存器列表。
- 002 ~ 013: regs_memory, 寄存器前后各一页信息。
- 014 ~ 015: data、bss、per_cpu的段信息。
- 016: last_kmsg, 发生kernel panic时的kernel log。
- 017: vmcore信息。
- 018: ylog_buf, Android log。
- 019 ~ 082: 内核中的task、runqueue、mem、stack、regs、irqstack, cpustack等信息。
- 083: 发生native_hang时的system_server等进程信息。
- 084: IO信息。
- 085 ~ 092: 发生AP_CPU_HANG时的现场信息。
- 093 ~ 097: CP dump保存的debug信息。
- 098: kernel_mem_info, 生成elf文件时使用的信息。

MiniDump2.0日志文件介绍 (4/4)

modem文件夹内容

 cm4_mem

 compress_record_file

- cm4_mem: cm4的IRAM

- compress_record_file: 记录压缩信息相关的文件, 解压缩时使用。

trusty文件夹中内容

 sml_mem

 tos_mem

- sml_mem: sml的memory, 发生sml panic时保存。

- tos_mem: tos的memory, 发生tos panic时保存。

MiniDump输出文件格式

- MiniDump2.0: zlib格式的压缩文件, 解压脚本 (decompress.py) 会保存到MiniDump日志中, 双击解压脚本实现日志的解压缩, 执行完脚本后, kernel文件夹中会生成一个MiniDump文件, 该文件支持使用Crash工具解析。
- Minidump1.0:
 - ✓ 使用的bootloader为Uboot: gzip格式的压缩文件, 可以使用脚本unisoc_parse_dumplog.py解压文件、合成MiniDump文件、解析生成last_kmsg.log文件。
 - ✓ 使用的bootloader为LK: zlib格式的压缩文件, 可以使用脚本unisoc_parse_dumplog_A13.py进行解压、合成MiniDump文件, 使用脚本generate_elfhdr_for_mini_v1.0.py生成elf头文件, 使用脚本parse_log_buf_k54.py解析生成last_kmsg.log。

实际使用过程中先将三个脚本放入MiniDump文件夹中, 然后执行unisoc_parse_dumplog_A13.py脚本即可, 因为脚本unisoc_parse_dumplog_A13.py执行过程中会自动调用其他两个脚本。

Minidump1.0如需获取上述脚本, 请提交CQ。

脚本支持的测试环境如下:

- Python 2.7.6 and 3.7.1 test pass on Ubuntu
- Python 2.7.1 and 3.7.1 test pass on Windows

Crash工具 “minimal” 模式分析

目前导出数据合成的MiniDump文件支持使用Crash工具 “minimal” 模式分析，命令如下：

- 32bit Arm®平台：
`./crash_arm minidump vmlinux -minimal`
- 64bit Arm®平台：
 - ✓ Android 14: `./crash_arm64 -m vabits_actual=39 -m kimage_voffset=0xffffffffbf88000000 minidump ../vmlinux --kaslr 0x80000 -minimal`
 - ✓ Android 12 、 Android 13: `./crash_arm64 -m vabits_actual=39 -m kimage_voffset=0xffffffffbf90000000 minidump vmlinux --minimal`
 - ✓ 其他: `./crash_arm64 -m kimage_voffset=0xffffffff7f88000000 minidump vmlinux -minimal`

minimal模式下，Crash工具支持使用log、sym、rd等命令对MiniDump文件进行简易分析。

06

Crash工具 常见命令介绍

进入crash之后，使用“help”指令查看Crash工具支持的所有命令；使用“help *cmd*”查看单独每个命令的使用方法，如“help ps”。

```
MEMORY: 2 GB
PANIC: ""
PID: 0
COMMAND: "swapper/0"
TASK: ffffffff800905f450 (1 of 4) [THREAD_INFO: ffffffff800905f450]
CPU: 0
STATE: TASK_RUNNING (ACTIVE)
WARNING: panic task not found

crash_arm64> help

*
alias      extend  log          rd          task
ascii      files   mach        repeat     timer
ascii      foreach  mod         runq       tree
bt         fuser   mount       search     union
bt         gdb     net         set        vm
compare    help    p           sig        vtop
dev        ipcs    ps          struct     waitq
dis        irq    pte        swap       whatis
eval       kmem   ptob       sym        wr
exit       list   ptov       sys        q

crash_arm64 version: 7.1.7++  gdb version: 7.6
For help on any command above, enter "help <command>".
For help on input options, enter "help input".
```

使用 “log” 命令：

- 可以将_log_buf中的Kernel log内容dump出来。

```
crash_arm64> log
[ 4.238861] c3 trusty: 0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[ 4.238866] c3 trusty: 0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[ 4.238870] c3 trusty: 0030 00 00 00 00 00 00 00 00 00
[ 4.238875] c3 trusty: trusty_kernelbootcp: 125: TA:update version flag = 0
[ 4.238879] c3 trusty: enter SEC_KBC_START_CP
[ 4.238883] c3 trusty: kbc_start_cp() enter
[ 4.238888] c3 trusty: reg_addr = 0xfffffffffe25fc048
[ 4.238892] c3 trusty: before reg = 2010101
[ 4.238896] c3 trusty: after reg = 10101
[ 4.238900] c3 trusty: reg_addr = 0xfffffffffe25fc0cc
```

- 可以将log的输出内容重定向到一个文件，便于检查异常。

例如将log重定向到kernel文件中，示例如下。

```
[ 4.296438] c1 cproc_proc_write: start!
[ 4.296445] c1 sprd_cproc: native start type = 0x0
[ 4.296450] c1 sprd_cproc_native_arm_start: test start, type = 0x0, status = 0x1
crash_arm64>
crash_arm64>
crash_arm64> log>kernel.txt
crash_arm64> █
```

使用 “ps” 命令可以列出所有线程及其状态等信息。

```
crash_arm64> ps
```

	PID	PPID	CPU	TASK	ST	%MEM	VSZ	RSS	COMM
>	0	0	0	ffffffff800905f450	RU	0.0	0	0	[swapper/0]
>	0	0	1	ffffffffffc079178d00	RU	0.0	0	0	[swapper/1]
>	0	0	2	ffffffffffc079179a00	RU	0.0	0	0	[swapper/2]
>	0	0	3	ffffffffffc07917a700	RU	0.0	0	0	[swapper/3]
	1	0	2	ffffffffffc079118000	IN	0.1	12732	2816	init
	2	0	2	ffffffffffc079118d00	IN	0.0	0	0	[kthreadd]
	3	2	0	ffffffffffc079119a00	IN	0.0	0	0	[ksoftirqd/0]
	4	2	0	ffffffffffc07911a700	IN	0.0	0	0	[kworker/0:0]
	5	2	0	ffffffffffc07911b400	IN	0.0	0	0	[kworker/0:0H]
	6	2	0	ffffffffffc07911c100	IN	0.0	0	0	[kworker/u8:0]
	7	2	2	ffffffffffc07911ce00	IN	0.0	0	0	[rcu_preempt]
	8	2	3	ffffffffffc07911db00	IN	0.0	0	0	[rcu_sched]

使用“bt”命令可以查看Kernel stack的back trace。

```
crash_arm> bt
PID: 490    TASK: eac78800  CPU: 7   COMMAND: "kworker/7:2"
#0 [<c05b9a28>] (sysdump_panic_event) from [<c01566dc>]
#1 [<c01566dc>] (notifier_call_chain) from [<c0156750>]
#2 [<c0156750>] (atomic_notifier_call_chain) from [<c012ff3c>]
#3 [<c012ff3c>] (panic) from [<c05b721c>]
```

07

SysDump 常见异常处理

- SysDump过程失败

SysDump过程均有屏幕显示，并伴有提示信息。

SysDump多数异常为SD卡出现异常，可尝试更换SD卡或使用Dump2PC方式解决。

- FullDump解析失败

屏幕打印信息显示：“crash : vmlinux and vmcore do not match!”

表示vmcore和vmlinux不匹配，可通过如下两个命令查看vmcore和vmlinux的版本是否一致。

```
strings vmcore |grep "Linux version"
```

```
strings vmlinux |grep "Linux version"
```

vmcore和vmlinux版本一致表示匹配。

注：SysDump日志文件的Pac包与vmlinux需要在同一次编译中生成才会匹配。

谢谢



文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

紫光展锐、UNISOC、展讯、Spreadtrum、SPRD、锐迪科、RDA及其他紫光展锐的商标均为紫光展锐（上海）科技有限公司及/或其子公司、关联公司所有。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

本文档可能包含第三方内容，包括但不限于第三方信息、软件、组件、数据等。紫光展锐不控制且不对第三方内容承担任何责任，包括但不限于准确性、兼容性、可靠性、可用性、合法性、适当性、性能、不侵权、更新状态等，除非本文档另有明确说明。在本文档中提及或引用任何第三方内容不代表紫光展锐对第三方内容的认可、承诺或保证。

用户有义务结合自身情况，检查上述第三方内容的可用性。若需要第三方许可，应通过合法途径获取第三方许可，除非本文档另有明确说明。