

Unisoc Confidential For hiar

SysDump简介

WWW.UNISOC.COM

紫光展锐科技

修改历史

版本号	日期	注释
V1.0	2020/6/1	初稿
V1.1	2020/9/6	1、调整文档结构 2、删除冗余信息，优化文档描述 3、操作方法步骤化 4、合并同类内容，修正文档错误 5、文档名称由 《UNISOC_sysdump_Brief_Introduction》修改 为《SysDump简介》
V1.2	2020/10/21	P18增加Logel_R9.20.1401_P1及之后版本上新增功能的描述。
V1.3	2020/12/29	1、更新P33 bt 命令执行后的示例图 2、更新P13 Crash工具的获取路径

关键字

SysDump、日志解析、Dump2PC、FullDump、Minidump

Unisoc Confidential For hiar

Unic Confidential For hiar

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

Unisoc Confidential For hiar

01 SysDump 简介



功能简介

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。

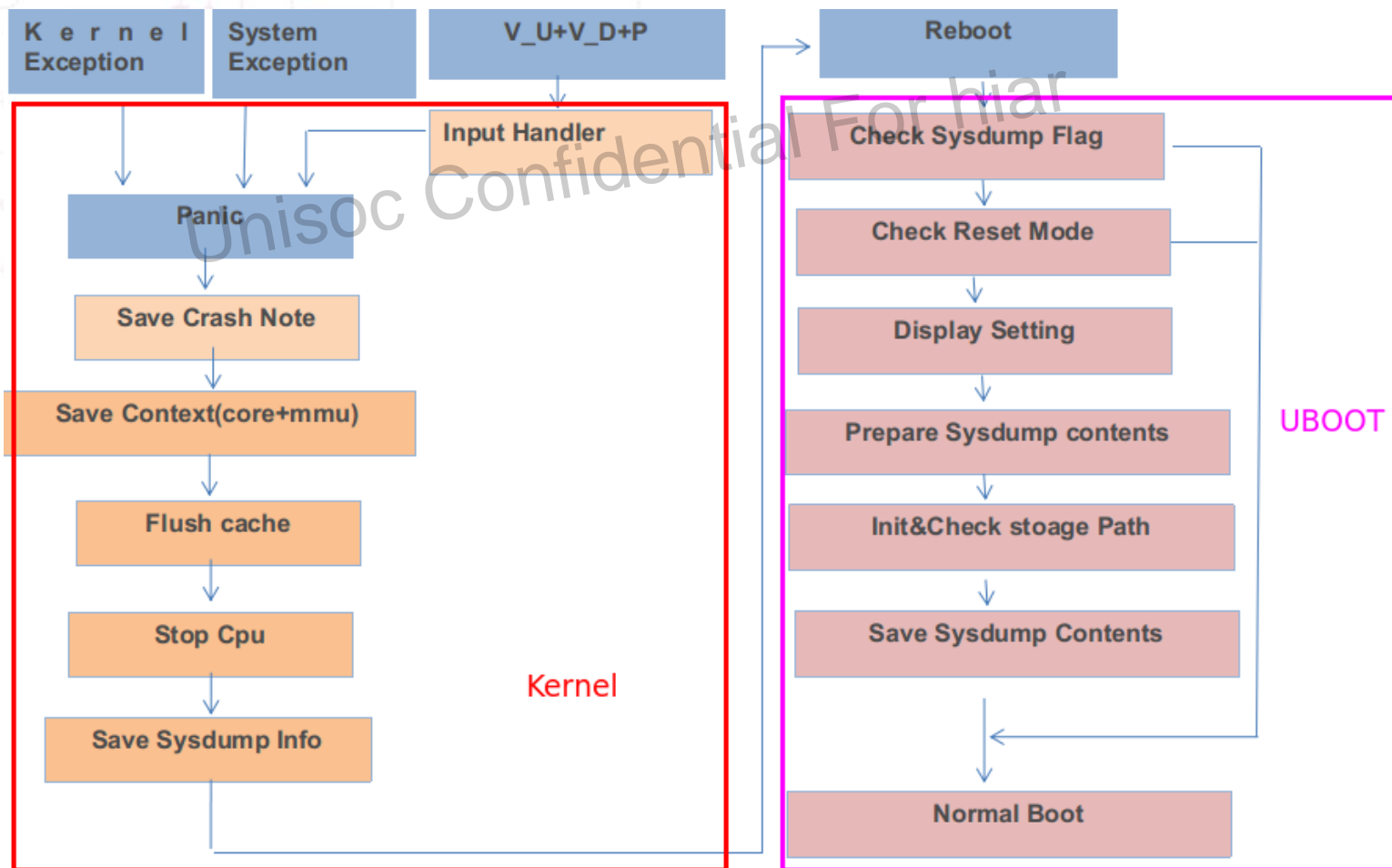
在系统发生诸如Kernel crash等异常时，在Kernel中完成flush cache等处理后，重启进入Uboot中完成所有数据的保存，保存过程会有相应屏幕提示，完成后根据屏幕提示重启手机，导出异常数据文件进行分析。

SysDump分为FullDump和MiniDump两个子功能，两个子功能是互不影响的。

- FullDump保存完整DDR信息，可以支持两种存储路径Dump2SD和Dump2PC。
- MiniDump保存少量信息到单独的SysDumpdb分区，然后再由native sevice 程序将分区保存的raw数据整理解析后放置到/sdcard/MiniDump路径下。

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。

设计简介



- 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（秒）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。

Unisoc Confidential For hiar

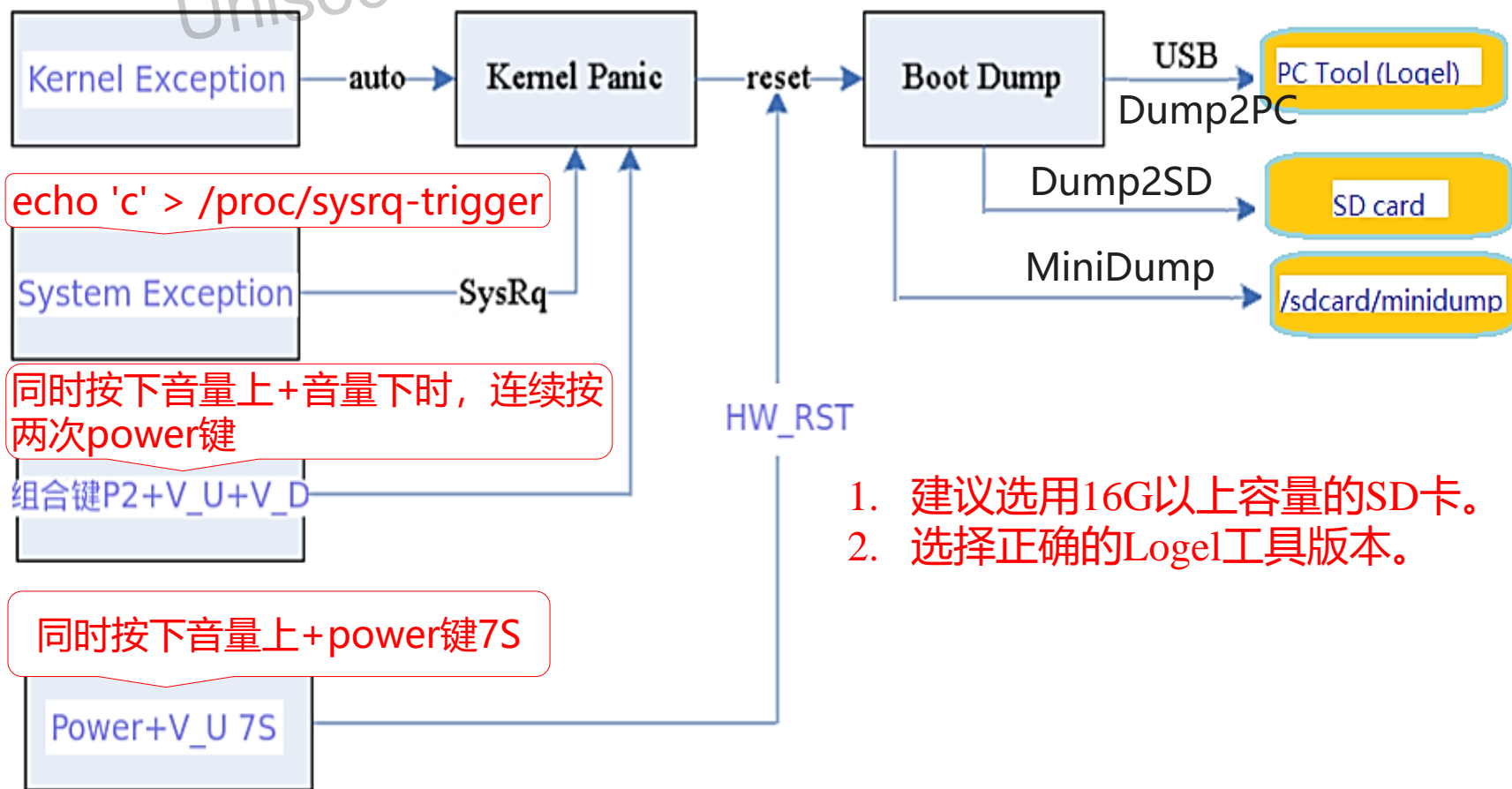
02 SysDump 配置



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。

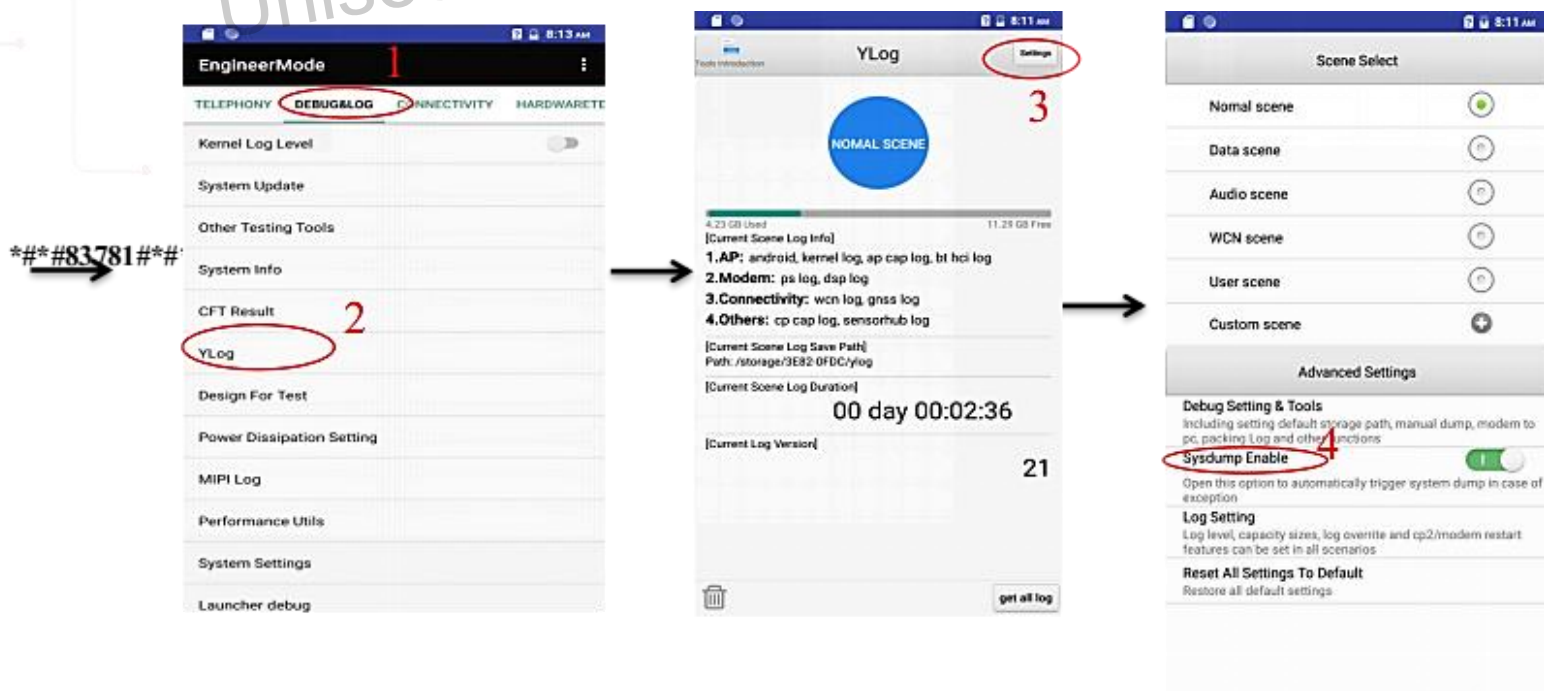
Panic函数中嵌入了sysdump的接口，以完成后续的sysdump流程。



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

功能配置

- FullDump功能在UserDebug版本上默认使能，在User版本上默认关闭。FullDump功能使能/禁止可以在“工程模式”中设置，具体如下：
(*##83781##*) --> Debug&LOG-->YLog-->Settings-->SysDump Enable。



- ✓ 使能/禁止FullDump操作后即可正常使用，不需重启设备。
- ✓ 重新开机后，FullDump的状态会保持上一次的使能/禁止状态。
- MiniDump功能在User 和 UserDebug版本默认都是使能。

Unisoc Confidential For hiar

03

FullDump 文件解析



FullDump文件解析(1/2)

- Dump2SD方式存储的FullDump一般在SD卡中，有SysDump文件夹，保存3份历史日志，其中文件夹1中为最新的。
- Dump2PC的文件可以通过Logel工具查看：File->Open Log Location。

SysDump文件主要有：

- 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*：log文件。

若无特殊说明，SysDump默认指的是FullDump。

dump_report.txt	2019/12/30 16:22	TXT 文件	3 KB
etbdata_uboot.bin	2019/12/30 16:26	BIN 文件	32 KB
sysdump.core.00	2019/12/30 16:22	00 文件	4 KB
sysdump.core.01_0x80000000-0x83fff...	2019/12/30 16:22	LST 文件	65,536 KB
sysdump.core.02_0x84000000-0x842f...	2019/12/30 16:22	LST 文件	3,072 KB
sysdump.core.03_0x84300000-0x844...	2019/12/30 16:22	LST 文件	1,388 KB
sysdump.core.04_0x8445b000-0x877f...	2019/12/30 16:22	LST 文件	52,884 KB
sysdump.core.05_0x87800000-0x87fff...	2019/12/30 16:22	LST 文件	8,192 KB
sysdump.core.06_0x88000000-0x895f...	2019/12/30 16:22	LST 文件	22,528 KB
sysdump.core.07_0x89600000-0x8ee...	2019/12/30 16:23	LST 文件	90,432 KB
sysdump.core.08_0x8ee50000-0x93fff...	2019/12/30 16:23	LST 文件	83,648 KB
sysdump.core.09_0x96000000-0xd5fff...	2019/12/30 16:24	LST 文件	1,048,576...
sysdump.core.10_0xd6000000-0xfd57...	2019/12/30 16:26	LST 文件	644,608 KB
sysdump.core.11_0x00800000-0x008...	2019/12/30 16:26	LST 文件	256 KB
sysdump-checksum.txt	2019/12/30 16:26	TXT 文件	1 KB
ylog_buf	2019/12/30 16:22	文件	1,024 KB

FullDump文件解析(2/2)

FullDump文件解析命令如下：

```
crash_arm -m phys_base=0x80000000 vmlinux vmcore
```

```
crash_arm64 -m phys_offset=0x80000000 vmlinux vmcore
```

其中：

- crash 工具及使用说明获取方法如下：

```
vendor/sprd/tools/crash/pycrash/bin/ //获取工具
```

```
vendor/sprd/tools/crash/Usage //获取工具使用说明
```

✓ crash_arm: 用于32bit ARM 平台

✓ crash_arm64:用于64bit ARM 平台

- vmlinux为编译时生成的最原始的内核文件，用于kernel debug，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
```


Unisoc Confidential For hiar

04

Dump2PC 功能介绍



Dump2PC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。

Dump2PC使用说明(1/6)

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。

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。

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

注：此处必须有USB插入动作才会被检测到。

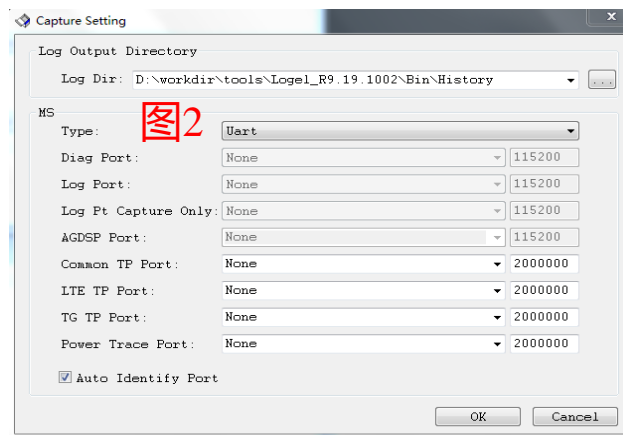
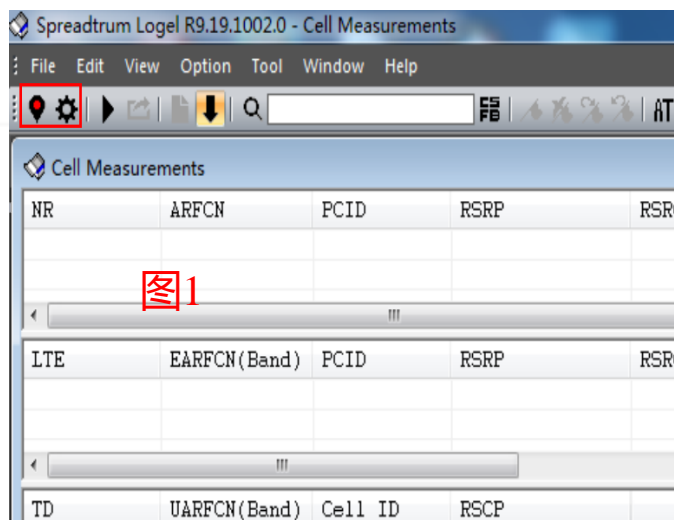
```
Sysdumping now, keep power on.

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+0x98/0xa8
[<000000005c7fd3e3>] gpio_keys_gpio_work_func+0x1c/0x40
[<00000000b045ff5b>] process_one_work+0x204/0x41c
[<00000000624cfa8d>] worker_thread+0x2ac/0x3b8
[<000000001aeb60b4>] 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 volum up pressed to abort
usb cable is inserted...
```

Dump2PC使用说明(2/6)

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。
 - 右侧按钮为“capture setting”，默认无需做任何设置，如图2所示。

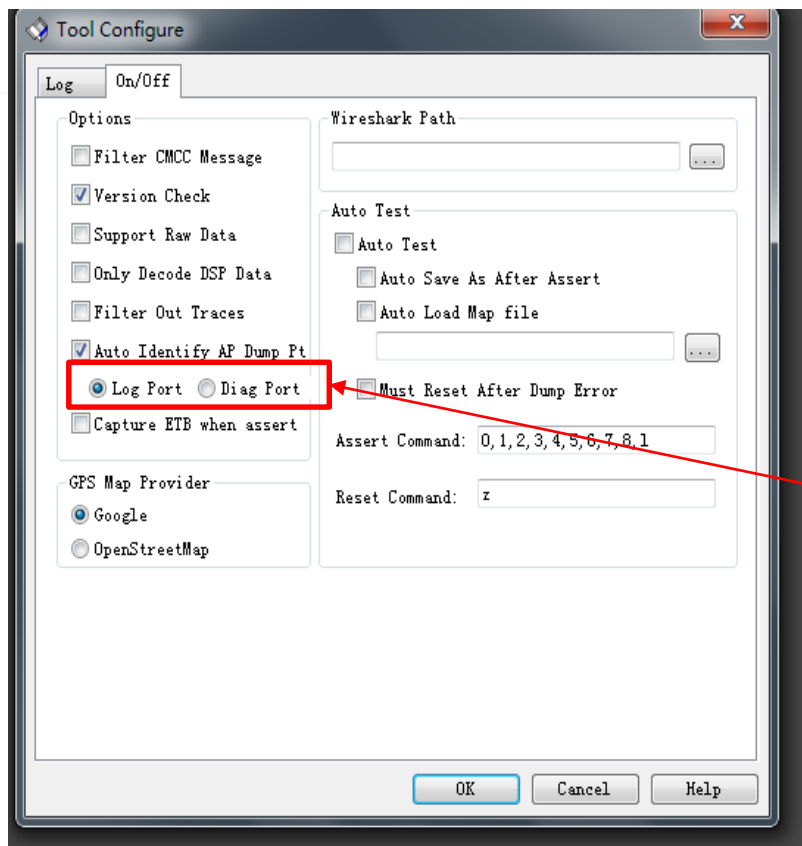


Dump2PC使用说明(3/6)

3. APDump端口自动识别设置

logel工具 (Logel_R9.19.1002_P1之后) 默认将APDump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时就会互相抢占这个端口, 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->ToolConfigure, 将AutoIdentityAPDumpPt勾上打开自动识别功能, 如下图所示。



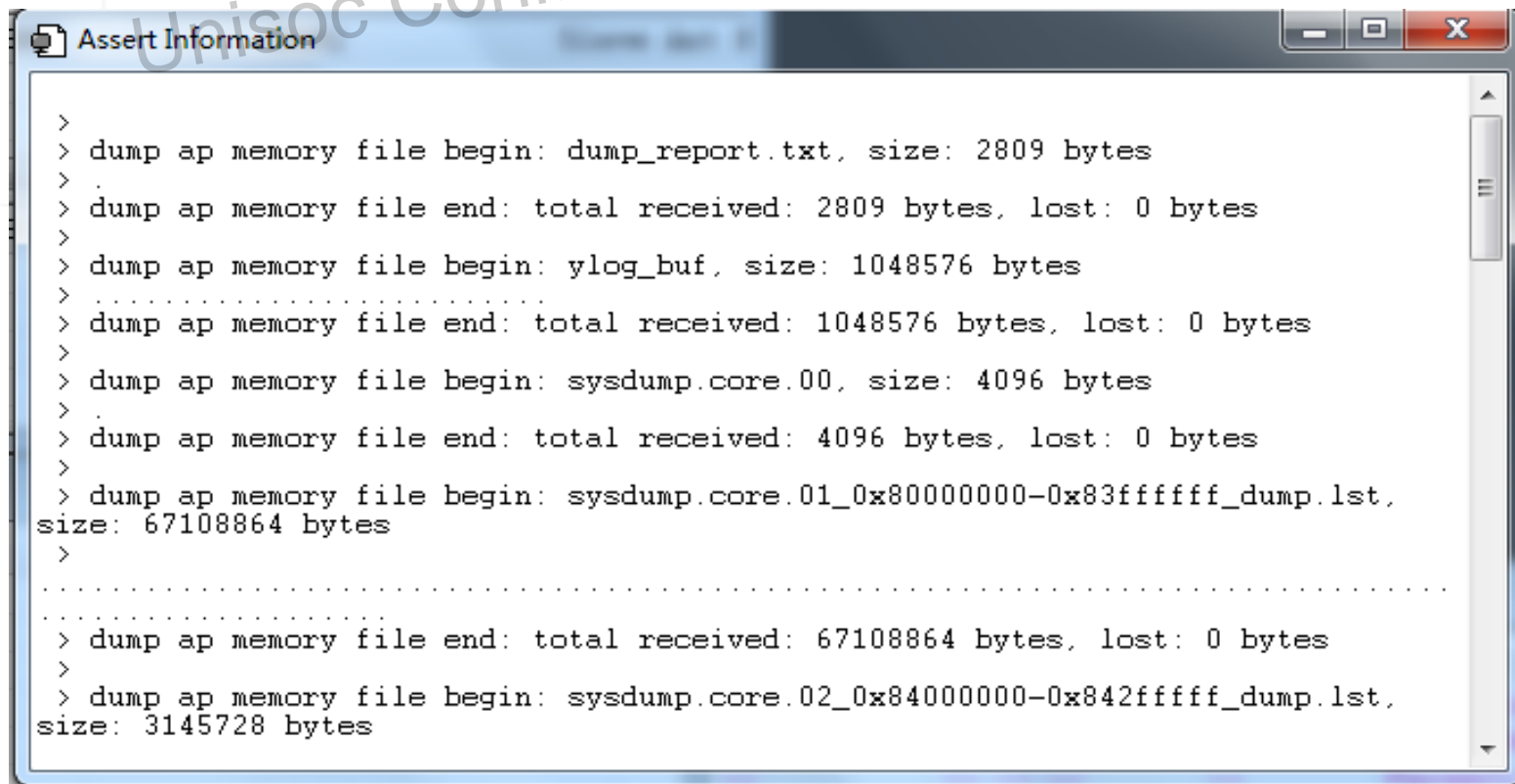
Logel_R9.20.1401_P1及之后版本上新增功能, 之前版本无此选项。

Dump2Pc时需要勾选Log Port。

Dump2PC使用说明(4/6)

4. Logel工具和手机握手成功，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“capture”按钮。

A screenshot of a Windows-style window titled "Assert Information". The window contains a text area with a monospaced font displaying log export progress. The text shows several "dump ap memory" commands being executed, with file names and sizes. The progress is shown in a series of lines, with some lines indicating the start and end of a dump, and others showing the total received and lost bytes. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
>
> 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
```

Dump2PC使用说明(5/6)

5. Dump2PC导出数据日志完成确认。

dump 完成后，手机提示“Press any key(Exp power key) to contiue...”，PC端也会显示“Total SysDump finished!”。此时按照设备屏幕提示按音量上键重启手机，整个dump过程结束。

```
Sysdumping now, keep power on.

Reset node: kernel crash
exception file info:
not-bugon
exception panic reason:
Crash Key:
exception stack info:
[<00000003af4d7f>] get_exception_stack_info+0xec/0x234
[<00000003c69d78>] prepare_exception_info+0x12d/0x170
[<0000000f08c2f185>] sysdump_enter+0x500/0x739
[<00000000944da842>] panic+0x140/0x259
[<00000000427c5235>] crash_note_save_cpu+0x0/0x170
[<00000000a95c619>] sysdump_event+0x3c/0x4c
[<000000004dea606c>] input_to_handler+0x100/0x118
[<000000007c6bd31d>] input_pass_values+0x6fc/0x130
[<000000005cd32c26>] input_handle_event+0x361/0x520
[<00000000f0925197>] input_event+0x70/0x90
[<00000000ac147692>] gpio_keys_gpio_report_event+0x38/0xa8
[<000000005e7fd3c3>] gpio_keys_gpio_work_func+0x1c/0x40
[<00000000b845f75b>] process_one_work+0x204/0x41c
[<00000000624cfa8d>] worker_thread+0x2ac/0x368
[<000000001ac60b4>] kthread+0x11c/0x12c
[<000000003cd604b>] ret_from_fork+0x10/0x18
[<000000002deca0c2>] 0xfffffffffffffff

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/volum up pressed to abort
usb cable is inserted...

handshake is ok from pc tools to sysdump!!!
Now start do memory dump
writing 0xa9f bytes to pc file dump_report.txt
writing 0x1000000 bytes to pc file ylog_buf
writing 0x1000000 bytes to pc file sysdump.core.01_0x00000000-0x03ffffff_dump.lst
writing 0x4000000 bytes to pc file sysdump.core.02_0x04000000-0x07ffffff_dump.lst
writing 0x3000000 bytes to pc file sysdump.core.03_0x08000000-0x0bffffff_dump.lst
writing 0x1500000 bytes to pc file sysdump.core.04_0xc0000000-0x0fffffff_dump.lst
writing 0x33a5000 bytes to pc file sysdump.core.05_0x00000000-0x07ffffff_dump.lst
writing 0x0000000 bytes to pc file sysdump.core.06_0x00000000-0x03ffffff_dump.lst
writing 0x1600000 bytes to pc file sysdump.core.07_0x09000000-0x0cffffff_dump.lst
writing 0x5850000 bytes to pc file sysdump.core.08_0x0c000000-0x0fffffff_dump.lst
writing 0x51b0000 bytes to pc file sysdump.core.09_0x0e000000-0x0ffffff_dump.lst
writing 0x4000000 bytes to pc file sysdump.core.10_0x0f000000-0x10ffffff_dump.lst
writing 0x2750000 bytes to pc file sysdump.core.11_0x00000000-0x00000000_dump.lst
writing 0x40000 bytes to pc file sysdump-checksum.txt
writing 0x3a3 bytes to pc file sysdump-checksum.txt
writing 0x0000 bytes to pc file etbdata_shoot.bin

Writing done.
Press any key (Exp power key) to continue...
```

```
Assert Information

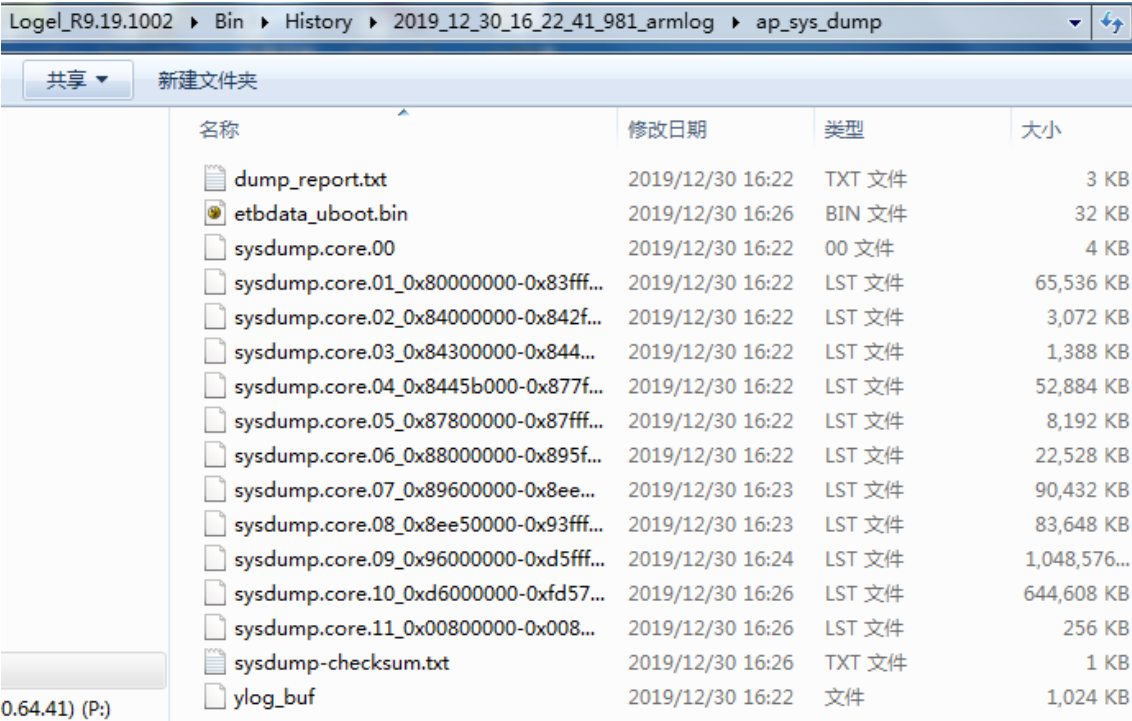
> dump ap memory file end: total received: 23068672 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.07_0x89600000-0x8ee4ffff_dump.lst,
size: 92602368 bytes
>
>
> dump ap memory file end: total received: 92602368 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.08_0x8ee50000-0x93ffffff_dump.lst,
size: 85655552 bytes
>
>
> dump ap memory file end: total received: 85655552 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.09_0x96000000-0xd5ffffff_dump.lst,
size: 1073741824 bytes
>
>
> dump ap memory file end: total received: 1073741824 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.10_0xd6000000-0xfd57ffff_dump.lst,
size: 660078592 bytes
>
>
> dump ap memory file end: total received: 660078592 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump.core.11_0x00800000-0x0083ffff_dump.lst,
size: 262144 bytes
>
>
> dump ap memory file end: total received: 262144 bytes, lost: 0 bytes
>
> dump ap memory file begin: sysdump-checksum.txt, size: 937 bytes
>
> dump ap memory file end: total received: 937 bytes, lost: 0 bytes
>
> dump ap memory file begin: etbdata_uboot.bin, size: 32768 bytes
>
> dump ap memory file end: total received: 32768 bytes, lost: 0 bytes
>
> Total sysdump finished!
>
```

Dump2PC使用说明(6/6)

6. SysDump日志文件检查。

SysDump日志文件存储在PC端工具的解压根目录/Bin/History/目录下以当前时间命名的文件夹XXX_armlog的ap_sys_dump文件夹中。

PC端显示界面如图：



名称	修改日期	类型	大小
dump_report.txt	2019/12/30 16:22	TXT 文件	3 KB
etbdata_uboot.bin	2019/12/30 16:26	BIN 文件	32 KB
sysdump.core.00	2019/12/30 16:22	00 文件	4 KB
sysdump.core.01_0x80000000-0x83fff...	2019/12/30 16:22	LST 文件	65,536 KB
sysdump.core.02_0x84000000-0x842f...	2019/12/30 16:22	LST 文件	3,072 KB
sysdump.core.03_0x84300000-0x844...	2019/12/30 16:22	LST 文件	1,388 KB
sysdump.core.04_0x8445b000-0x877f...	2019/12/30 16:22	LST 文件	52,884 KB
sysdump.core.05_0x87800000-0x87fff...	2019/12/30 16:22	LST 文件	8,192 KB
sysdump.core.06_0x88000000-0x895f...	2019/12/30 16:22	LST 文件	22,528 KB
sysdump.core.07_0x89600000-0x8ee...	2019/12/30 16:23	LST 文件	90,432 KB
sysdump.core.08_0x8ee50000-0x93ff...	2019/12/30 16:23	LST 文件	83,648 KB
sysdump.core.09_0x96000000-0xd5ff...	2019/12/30 16:24	LST 文件	1,048,576...
sysdump.core.10_0xd6000000-0xfd57...	2019/12/30 16:26	LST 文件	644,608 KB
sysdump.core.11_0x00800000-0x008...	2019/12/30 16:26	LST 文件	256 KB
sysdump-checksum.txt	2019/12/30 16:26	TXT 文件	1 KB
ylog_buf	2019/12/30 16:22	文件	1,024 KB

Unisoc Confidential For hiar

05

MiniDump 文件解析



MiniDump功能简介

- MiniDump功能常用于无SD卡或无法Dump2PC的场景，当前设计默认开启。
- MiniDump内容：
 - ✓ Android 11.0默认保存在/data/minidump目录下。
 - ✓ Android 10.0默认保存在/sdcard/minidump目录下。
- MiniDump日志使用TRACE32 ARM SIMULATOR（简称T32 sim）或crash等工具进行分析。
- 目前MiniDump功能支持保存5份历史数据，保存方式为：MiniDump文件夹下有名称为1、2、3、4、5的5个文件夹，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 pull命令直接导出，新版的log导出工具中也已经默认添加/data/minidump路径（Android 10.0 路径为/sdcard/minidump）。

MiniDump日志文件介绍

MiniDump生成的日志文件如下图所示：

minidump/1/

00_minidump_elfhdr	1
01_minidump_regs	2
02_minidump_regs_R1	
03_minidump_regs_R6	
04_minidump_regs_R9	
05_minidump_regs_R12	
06_minidump_regs_R13	
07_minidump_regs_R16	
08_minidump_regs_R19	
09_minidump_regs_R20	3
10_minidump_regs_R23	
11_minidump_regs_R25	
12_minidump_regs_R26	
13_minidump_regs_R28	
14_minidump_regs_R29	
15_minidump_regs_R30	
16_minidump_regs_R31	
17_minidump_regs_R32	
18_minidump_section_text	
19_minidump_section_data	
20_minidump_section_bss	
21_minidump_section_init	
22_minidump_section_inittext	4
23_minidump_section_rodata	
24_minidump_section_per_cpu	
25_minidump_section_log_buf	
26_minidump_section_ylog_buf	5
dump_report.txt	6
last_kmsg.log	
minidump	7

其中：

1. 00_minidump_elfhdr: elf文件头。
2. 01_minidump_regs: cpu regs。
3. 02~17: 是regs 前后256Bytes。
4. 18~25: 各段信息。
5. 26_minidump_section_ylog_buf: panic时Ylog buf。
6. dump_report: dump描述信息。
7. last_kmsg.log、minidump: 由脚本解析生成，保存系统重启之前的最后的内核log信息。

MiniDump日志解压

MiniDump输出的文件是压缩格式的，可以使用工具`unisoc_parse_dumplog.py`进行解压。

该工具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.

该工具主要功能有：

- 对导出的压缩文件解压。
- 解析生成文件`last_kmsg.log`。
- 合成MiniDump文件供T32 sim或crash工具分析使用。











该工具在linux环境或windows命令方式下使用方法如下：

- 使用指令：`python unisoc_parse_dumplog.py xxxx/minidump/1`
其中`xxxx/minidump/1`为MiniDump压缩文件所在目录。
- 将该工具直接复制到压缩文件所在一级目录，直接执行。

MiniDump日志分析_T32 sim(1/2)

MiniDump功能保存的MiniDump日志使用T32 sim进行分析。

T32 sim官网获取路径：<https://www.lauterbach.com/frames.html?Home.html>

Description	File	File/Date
Simulator for S08/HC08	 sim08.zip	12.01MB / 08-Jun-2018
Simulator for S12Z/S12X/S12/HC12	 sim12.zip	12.20MB / 08-Jun-2018
Simulator for 68HC16	 sim16.zip	12.04MB / 08-Jun-2018
Simulator for C166/XC2000/XC16x	 sim166.zip	12.32MB / 08-Jun-2018
Simulator for Intel 186	 sim186.zip	12.32MB / 08-Jun-2018
Simulator for 68K/ColdFire	 sim68k.zip	12.33MB / 08-Jun-2018
Simulator for 78K0R/RL78	 sim78k.zip	12.09MB / 08-Jun-2018
Simulator for ARM /CORTEX/XSCALE	 simarm.zip	14.08MB / 08-Jun-2018
Simulator for ARM 64	 simarm64.zip	14.45MB / 08-Jun-2018
Simulator for AVR32	 simavr32.zip	12.10MB / 08-Jun-2018

MiniDump日志分析_T32 sim(2/2)

所需工具及文件：

- unisoc_parse_dumplog.py
- simarm64或simarm
- unisoc_Debug.cmm脚本

分析方法：

1. 使用工具unisoc_parse_dumplog.py合成MiniDump文件。
2. 下载simarm.zip（arm64请使用simarm64.zip）并解压。
3. 将TRACE32脚本和1中合成的文件添加到2解压后的文件夹中。
4. 执行解压的文件夹中的t32marm.exe。

Windows直接双击执行，linux 需要安装wine，然后执行命令：wine t32marm.exe执行。

5. 启动到默认界面后，点击“ File” ->“ Run Script” ，选择unisoc_Debug.cmm脚本并执行。
6. 选择unisoc_Debug.cmm脚本执行结果进行分析。

MiniDump日志分析_crash

crash 工具 “minimal” 模式分析

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

arm时使用命令：

```
crash_arm minidump vmlinux --minimal
```

arm64时使用命令：

```
crash_arm64 -m kimage_voffset=0xffffffff7f88000000 minidump vmlinux --minimal
```

该模式中可以使用命令 “log” 完整的 解析出Kernel logbuf内容。

另外crash工具还支持其他如sym,rd等命令对MiniDump文件进行简易分析。

Unisoc Confidential For hiar

06 Crash工具 常见命令 介绍



help

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: fffffff800905f450 (1 of 4) [THREAD_INFO: fffffff800905f450]
CPU: 0
STATE: TASK_RUNNING (ACTIVE)
WARNING: panic task not found

crash_arm64> help

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

log

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
```

- 可以把 log 的输出内容重定向到一个文件，便于后续查看分析。

```
[ 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> █
```

ps

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	ffffffffc079178d00	RU	0.0	0	0	[swapper/1]
>	0	0	2	ffffffffc079179a00	RU	0.0	0	0	[swapper/2]
>	0	0	3	ffffffffc07917a700	RU	0.0	0	0	[swapper/3]
	1	0	2	ffffffffc079118000	IN	0.1	12732	2816	init
	2	0	2	ffffffffc079118d00	IN	0.0	0	0	[kthreadd]
	3	2	0	ffffffffc079119a00	IN	0.0	0	0	[ksoftirqd/0]
	4	2	0	ffffffffc07911a700	IN	0.0	0	0	[kworker/0:0]
	5	2	0	ffffffffc07911b400	IN	0.0	0	0	[kworker/0:0H]
	6	2	0	ffffffffc07911c100	IN	0.0	0	0	[kworker/u8:0]
	7	2	2	ffffffffc07911ce00	IN	0.0	0	0	[rcu_preempt]
	8	2	3	ffffffffc07911db00	IN	0.0	0	0	[rcu_sched]

bt

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>]
```


Unisoc Confidential For hiar

07

SysDump 常见异常 处理



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。

- FullDump 解析失败

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

SysDump 文件和vmlinux 不匹配，可通过如下两个命令查看获取的时间是否一致

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

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

Unisoc Confidential For hiar

谢谢



本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。