以下是关于 Rawdump 相关组件的总结及工作时序图。

1. Rawdump 核心组件与作用总结（基于源码）

组件名称	运行位置	核心源码文件 (Evidence)	核心作用与机制
`memorydump`	QNX Host	`ramdump.c` `ss_minidump_main.c`	**总控与登记处** 1. **下发模式**：接收配置指令，调用 `set_wdog_dload_cookie` 向 **Boot MISC 寄存器** 写入 `MINI_RAWDUMP` 标志，告知固件崩溃后执行 Rawdump。 2. **Host 登记**：提供 `md_ss_register_region` 接口，允许 QNX 侧驱动（如 Audio, Camera）将自身内存注册到 SMEM 的全局目录表（Global TOC）中。

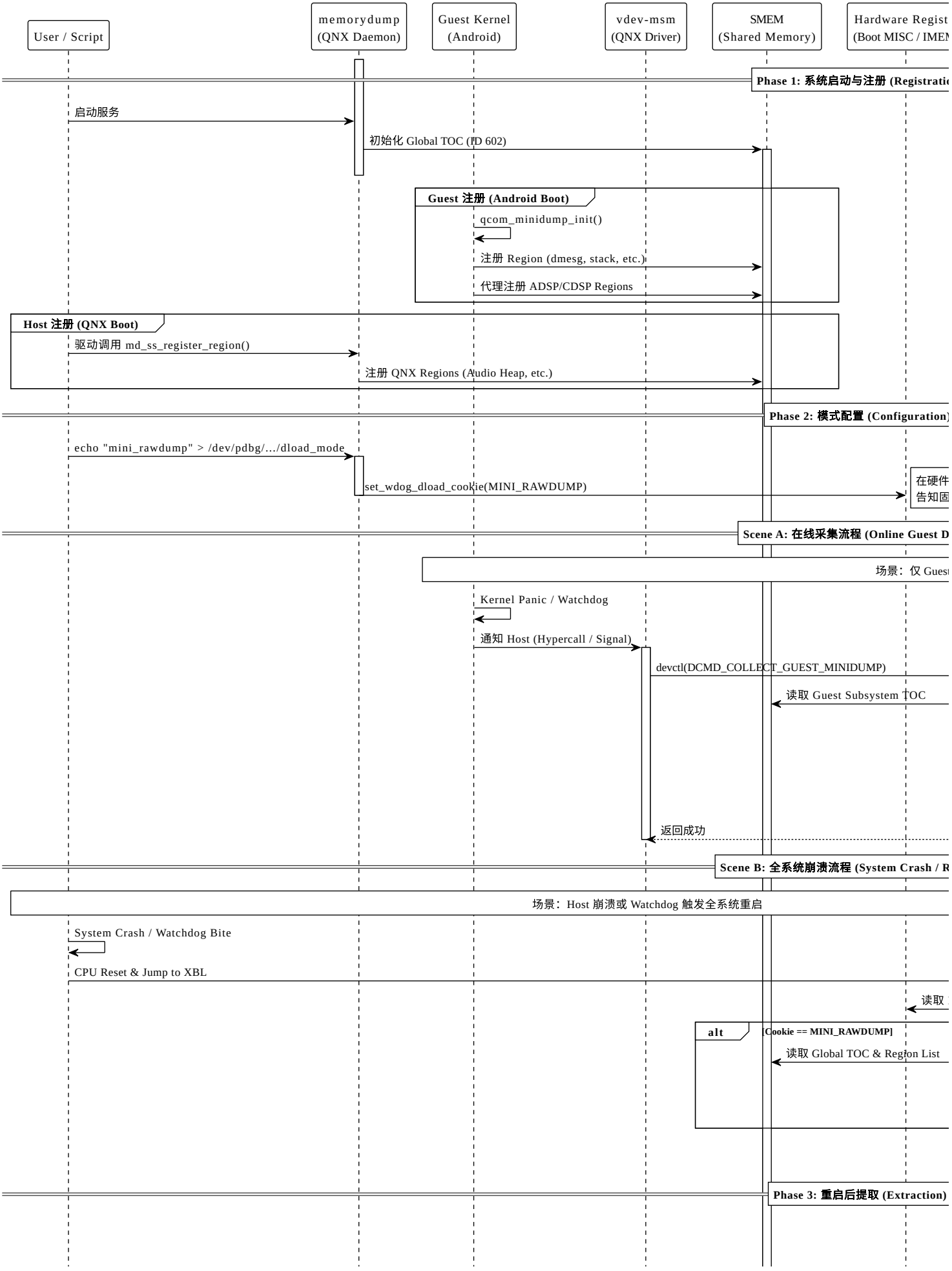
`log_collector`	QNX Host	`main.c`
		`vm_collect_minidump.c`
		`log_collect_resmgr.c`
			搬运工与解析者
			1. **离线提取**：系统重启后，读取 `/dev/disk/rawdump`，校验 `Raw_Dmp!` 签名，将分区数据提取为文件并清空分区头。
			2. **在线采集**：作为资源管理器响应 `devctl`，通过 SMEM 获取 Guest VM 内存布局，利用 `mmap` (MAP_PHYS) 在运行时提取 VM 内存。

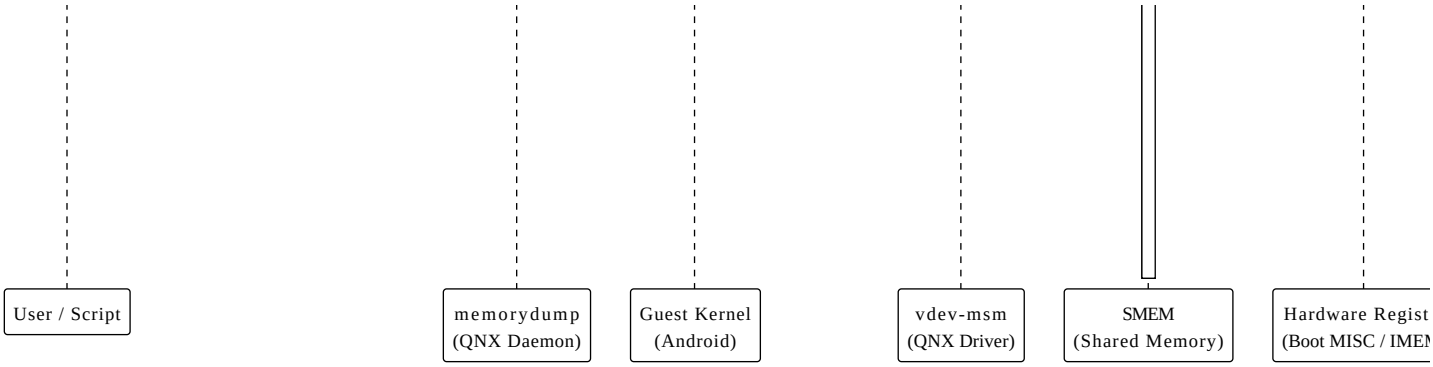
`vdev-msm`	QNX Host	`vdev-msm.c` (引用自搜索结果)	**吹哨人** 监听 Guest VM 的状态。当 Guest 发生崩溃但系统未重启时，它调用 `devctl(..., DCMD_COLLECT_GUEST_MINIDUMP, ...)` 通知 `log_collector` 进行在线采集。
`qcom_minidump`	Guest Kernel	`drivers/remoteproc/*.c` (引用自搜索结果)	**Guest 代理人** 1. **自身注册**：在 Android 启动时，将内核日志 (`dmesg`)、堆栈等注册到 SMEM。 2. **代理注册**：协助 ADSP/CDSP 等无法直接访问 SMEM 的子系统注册 Minidump 区域。
Firmware (XBL)	Bootloader	(逻辑推断自 `log_collector` 的解析代码)	**执行者** 在系统崩溃且通过 Boot MISC 寄存器确认进入 Rawdump 模式后，直接读取 SMEM 中的 TOC 清单，将物理内存写入 Rawdump 分区。
SMEM	Hardware	`smem_type.h` `ss_minidump_main.h`	**共享账本** 存储 `SMEM_MINIDUMP_ID (602)`，包含全局目录表（Global TOC）和各子系统区域描述符（Region Descriptors），是 Host、Guest 和固件交互的唯一数据中心。

2. 组件协同工作时序图

该图包含两个核心流程：

1. **System Crash Flow (Rawdump)**: 涉及 `memorydump` 配置、固件写盘、 `log_collector` 提取。
2. **Online Guest Dump Flow**: 涉及 `vdev-msm` 触发、 `log_collector` 在线 `mmap` 。

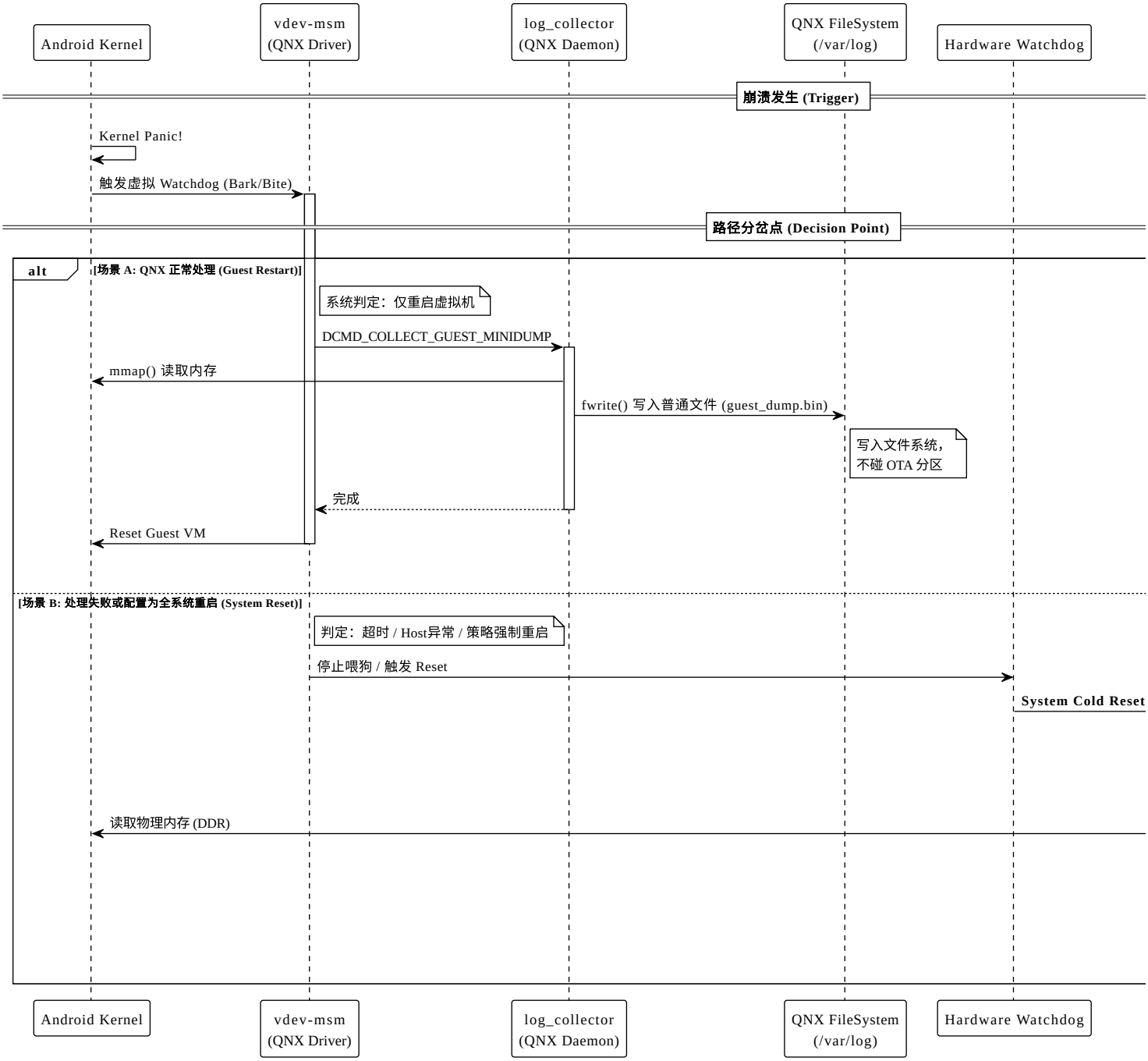

User / Script
memorydump (QNX Daemon)
Guest Kernel (Android)
vdev-msm (QNX Driver)
SMEM (Shared Memory)
Hardware Regist
(Boot MISC / IMEM
Phase 1: 系统启动与注册 (Registrati
启动服务
初始化 Global TOC (ID 602)
Guest 注册 (Android Boot)
qcom_minidump_init()
注册 Region (dmesg, stack, etc.)
代理注册 ADSP/CDSP Regions
Host 注册 (QNX Boot)
驱动调用 md_ss_register_region()
注册 QNX Regions (Audio Heap, etc.)
Phase 2: 模式配置 (Configuration)
echo "mini_rawdump" > /dev/pdbg/.../dload_mode
set_wdog_dload_cookie(MINI_RAWDUMP)
在硬件
告知固
Scene A: 在线采集流程 (Online Guest D
场景：仅 Guest
Kernel Panic / Watchdog
通知 Host (Hypercall / Signal)
devctl(DCMD_COLLECT_GUEST_MINIDUMP)
读取 Guest Subsystem TOC
返回成功
Scene B: 全系统崩溃流程 (System Crash / R
场景：Host 崩溃或 Watchdog 触发全系统重启
System Crash / Watchdog Bite
CPU Reset & Jump to XBL
读取
alt
[Cookie == MINI_RAWDUMP]
读取 Global TOC & Region List
Phase 3: 重启后提取 (Extraction)



3. 图解说明

1. **注册阶段（数据源头）**：
   Rawdump 里的数据是由 Guest (`qcom_minidump`) 和 Host (`memorydump`) 在**系统正常启动时**主动去 SMEM 里“占座”登记的。固件只负责按名单抓人。
2. **崩溃契约（Hardware Register）**：
   `memorydump` 进程的关键作用是**写硬件寄存器**（Cookie）。没有这一步，固件在崩溃重启后会直接进入正常引导，而不会去执行“写 Rawdump 分区”这个耗时操作。
3. **双模工作**：

- **在线模式**：不经过固件，由 `log_collector` 直接通过物理内存映射（ `mmap` ）去“偷”虚拟机的内存。
- **离线模式**：经过固件，数据先暂存在 `Rawdump Partition` ，重启后由 `log_collector` 搬运成文件。


Android Kernel
vdev-msm (QNX Driver)
log_collector (QNX Daemon)
QNX FileSystem (/var/log)
Hardware Watchdog
崩溃发生 (Trigger)
Kernel Panic!
触发虚拟 Watchdog (Bark/Bite)
路径分岔点 (Decision Point)
alt
[场景 A: QNX 正常处理 (Guest Restart)]
系统判定：仅重启虚拟机
DCMD_COLLECT_GUEST_MINIDUMP
mmap() 读取内存
fwrite() 写入普通文件 (guest_dump.bin)
写入文件系统，
不碰 OTA 分区
完成
Reset Guest VM
[场景 B: 处理失败或配置为全系统重启 (System Reset)]
判定：超时 / Host异常 / 策略强制重启
停止喂狗 / 触发 Reset
System Cold Reset
读取物理内存 (DDR)
Android Kernel
vdev-msm (QNX Driver)
log_collector (QNX Daemon)
QNX FileSystem (/var/log)
Hardware Watchdog