Objective by the Sea
version 8.0





What's at the Bottom of the Sea, One Baseband?



Lukas Arnold – 16th October 2025







Objective by the Sea
version 8.0

What's at the Bottom of the C1 Baseband?

Lukas Arnold – 16th October 2025



Apple’s Baseband Families



Infineon → intel. →

S- / X-Gold → ICE → C4000

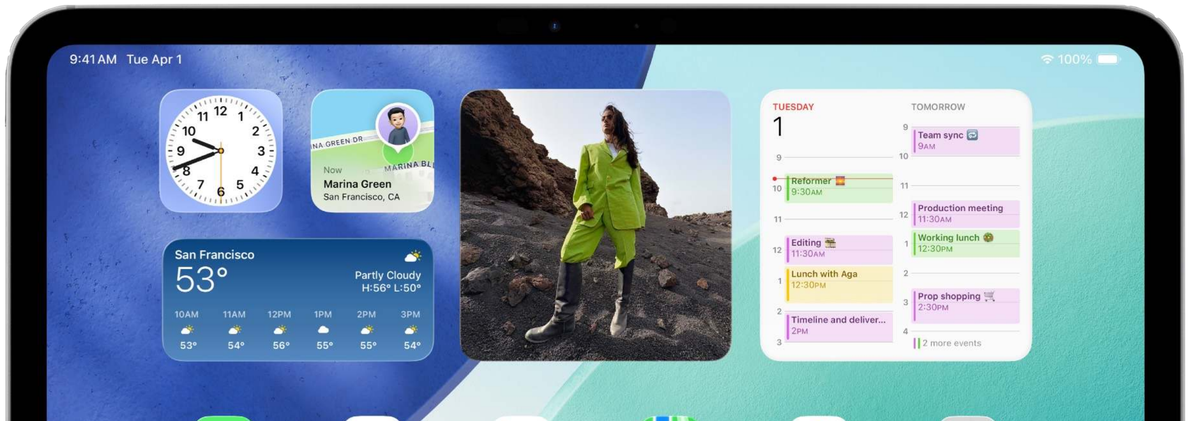
MEDIATEK

Dale

Qualcomm

Maverick





Apple’s Baseband Families



Infineon → intel. →

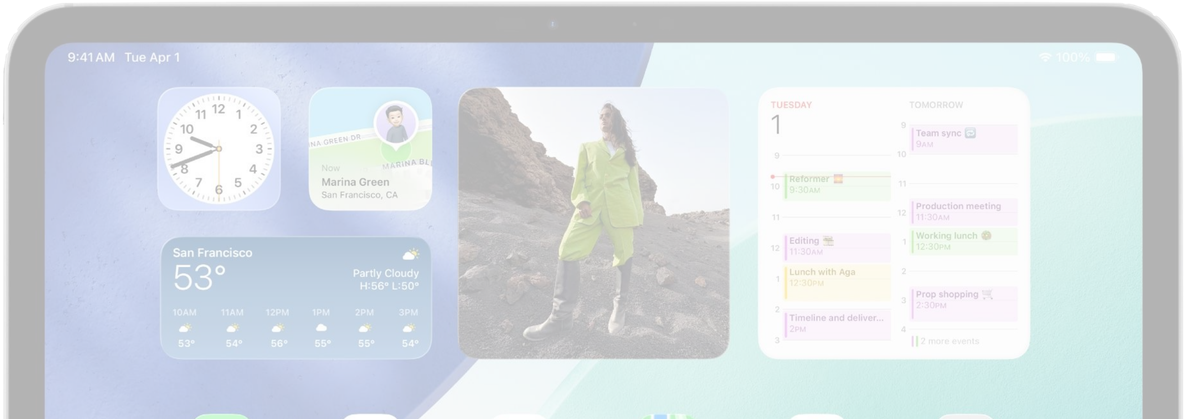
S- / X-Gold → ICE → C4000

MEDIATEK

Qualcomm

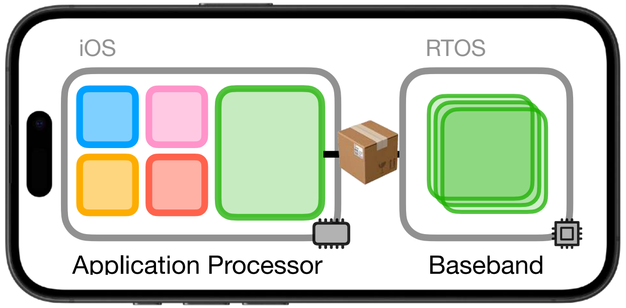
Maverick





iOS-Baseband Protocols

Timeline of their Usage



Qualcomm MSM Interface (QMI)

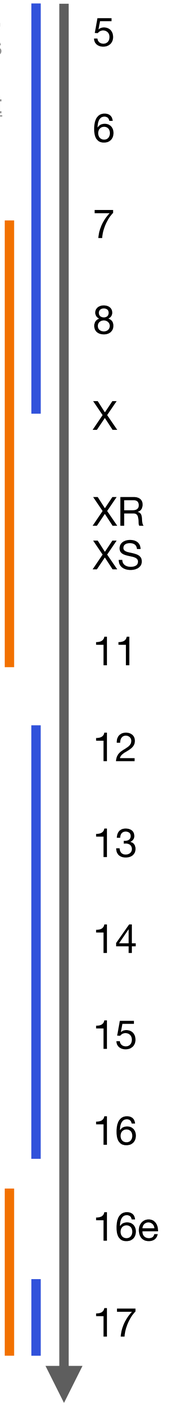
- Used in iPhones, Android smartphones, USB modems
- Open-source libraries available, e.g. libqmi
- iOS: Mostly proprietary service and message types
- BaseTrace

Apple Remote Invocation (ARI)

- Only seen in combination with Intel and Apple basebands
- New protocol reverse engineered by Tobias Kröll
- Automatic recovery of structure information from libARI.dylib
- ARIstoteles

Without iPhone SE, iPads, Apple Watches

See full list



Adaption of Existing ARI Tooling

From Intel to Apple

CellGuard

New ARI structures for 5G and satellite communication

ARIstoteles

2 files changed +5368 -414 lines changed

Search within code

CHANGELOG.md +1

```
@@ -9,6 +9,7 @@ This project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.htm
### Features

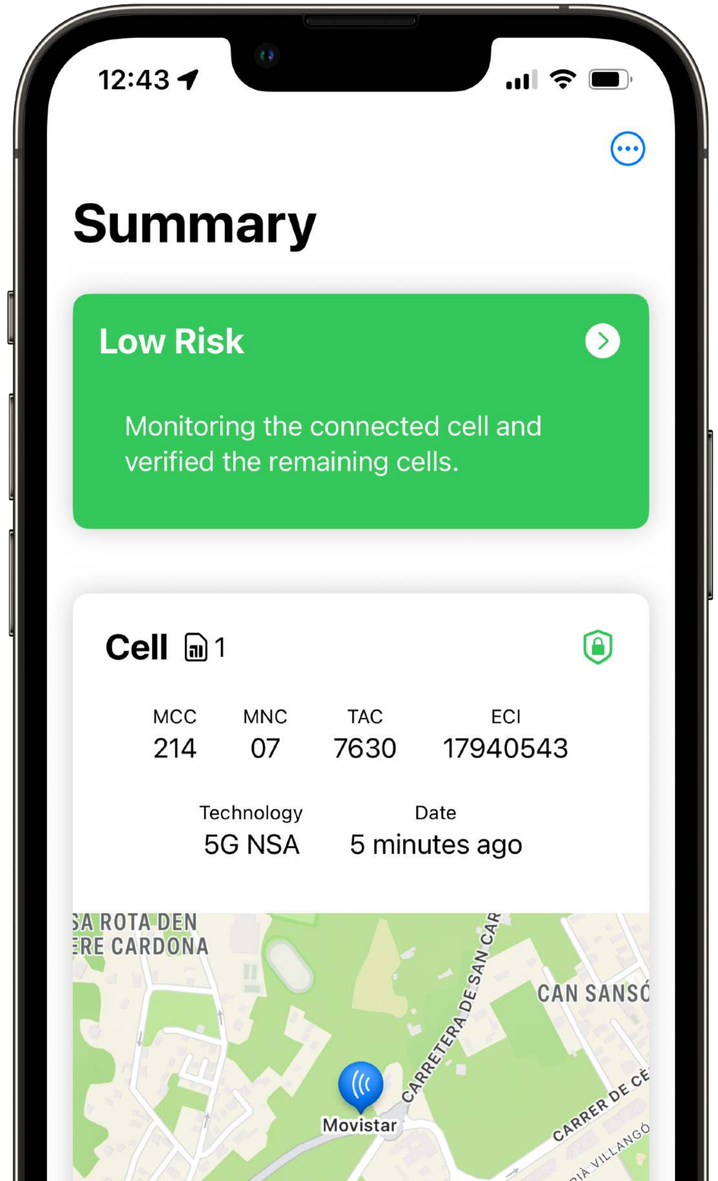
- Added TLV parser for `IBILteCellInfoT`
+ - Added the new ARI structures from iOS 18.3.1

### Changes
```

types/structure/libari_dylib.lua +5.367 -414

```
@@ -1,4 +1,4 @@
- --- AUTO GENERATED OUTPUT from Ghidra script "ari-structure-extractor.py" on 2025-02-24_12:04:55
+ --- AUTO GENERATED OUTPUT from Ghidra script "ari-structure-extractor.py" on 2025-03-08_15:59:48
  return {
      [1] = {
          ["name"] = "_ARIMSGDEF_GROUP01_bsp",

@@ -3411,6 +3411,13 @@ return {
```



What’s interesting about C1?

It’s the future baseband for all iPhones

- iPhone 16e & Air use C1(X) baseband
  - C1X is more efficient (chip_revision_b0)
  - Identical firmware for both
  - Drawbacks: No 5G mmWave
- iPhone 17 (Pro) still use Qualcomm BBs
- Rumors that iPhone 17e & full iPhone 18 line-up will switch to C2





iPhone 16e
February 2025



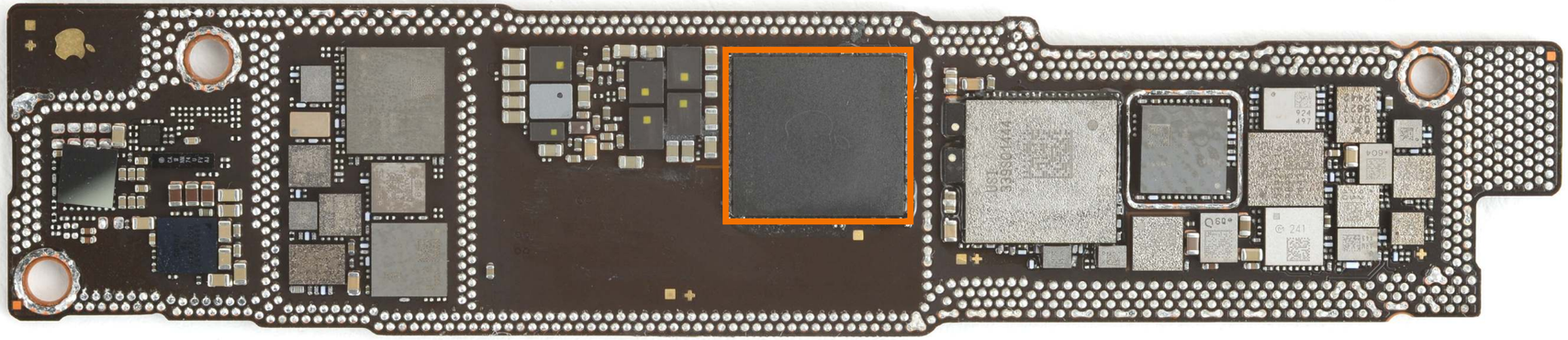


iPhone Air
September 2025

Hardware Architecture ⚡

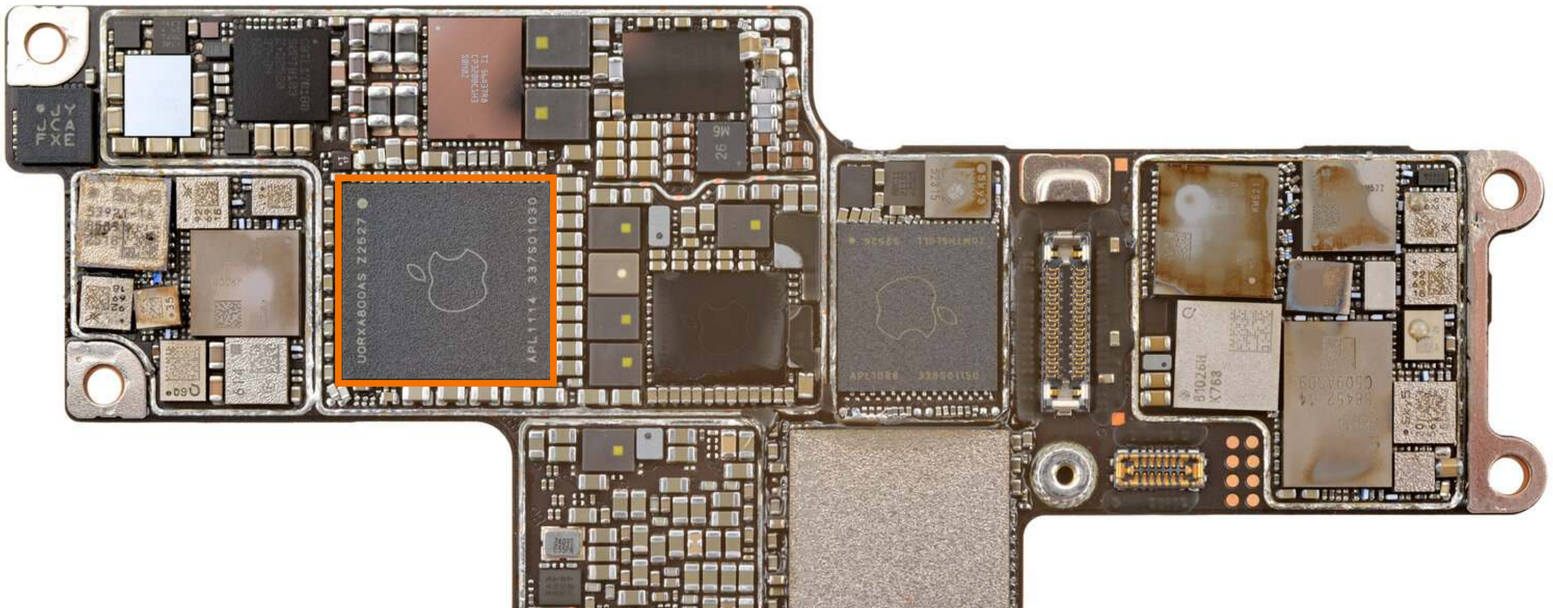
C1 in iPhone Teardowns

Part number: APL1114 - 337S01030



iPhone 16e

RF bottom board that is at the top of the logic board



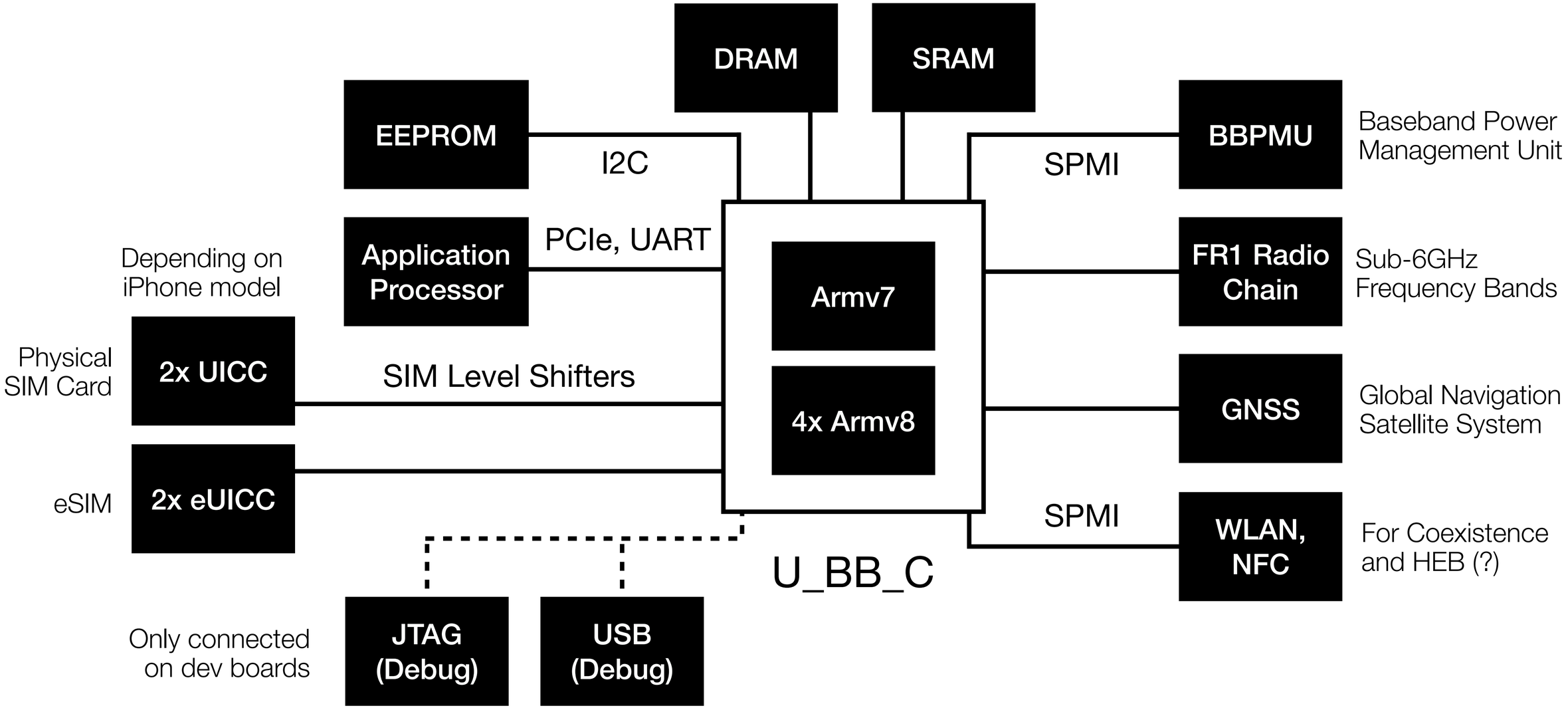
iPhone Air

Outer side of the main board

Source: iFixit (iPhone 16e, iPhone Air)

Abstract Schematics of C1

Focus on U_BB_C with identifier TMRP15C0-C1



C1’s IO Connections

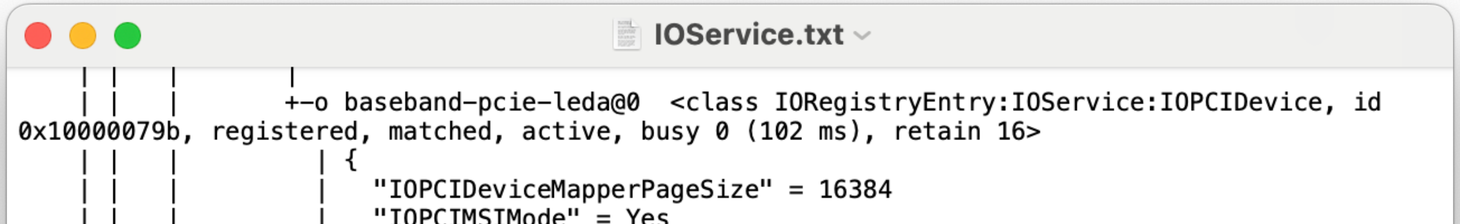
What do IO*.txt files in sysdiagnoses tell us?

Enables inclusion of baseband crash logs into sysdiagnoses

- Shared disk: /private/var/wireless/baseband_data
- AppleARMIODevice: baseband (Interrupt Controller)
- AppleSMCInterface: smc-pmu (Power Management Unit)
- AppleARMSPMIService: baseband-heb, baseband-idc (SPMI)
- IOPCIDevice: baseband-pcie-leda (PCIe)
- IOPlatformDevice: baseband (IO Service Interface)

IDC = Inter-Device Communication for Coexistence?

HEB = Human Ether Blockage?



IOPlatformDevice: baseband

Creates AppleBasebandINT1 (Provider for IO Services)

- Processes that interact with the baseband create an AppleBasebandUserClient

nfcd, CommCenter, abm-helper, locationd, WirelessRadioMan

- AppleConvergedIPCLedaBBControl defines receivers for RTKit IPC calls
- Each receiver creates IOAppleConvergedIPCRTIDevice with interface definitions

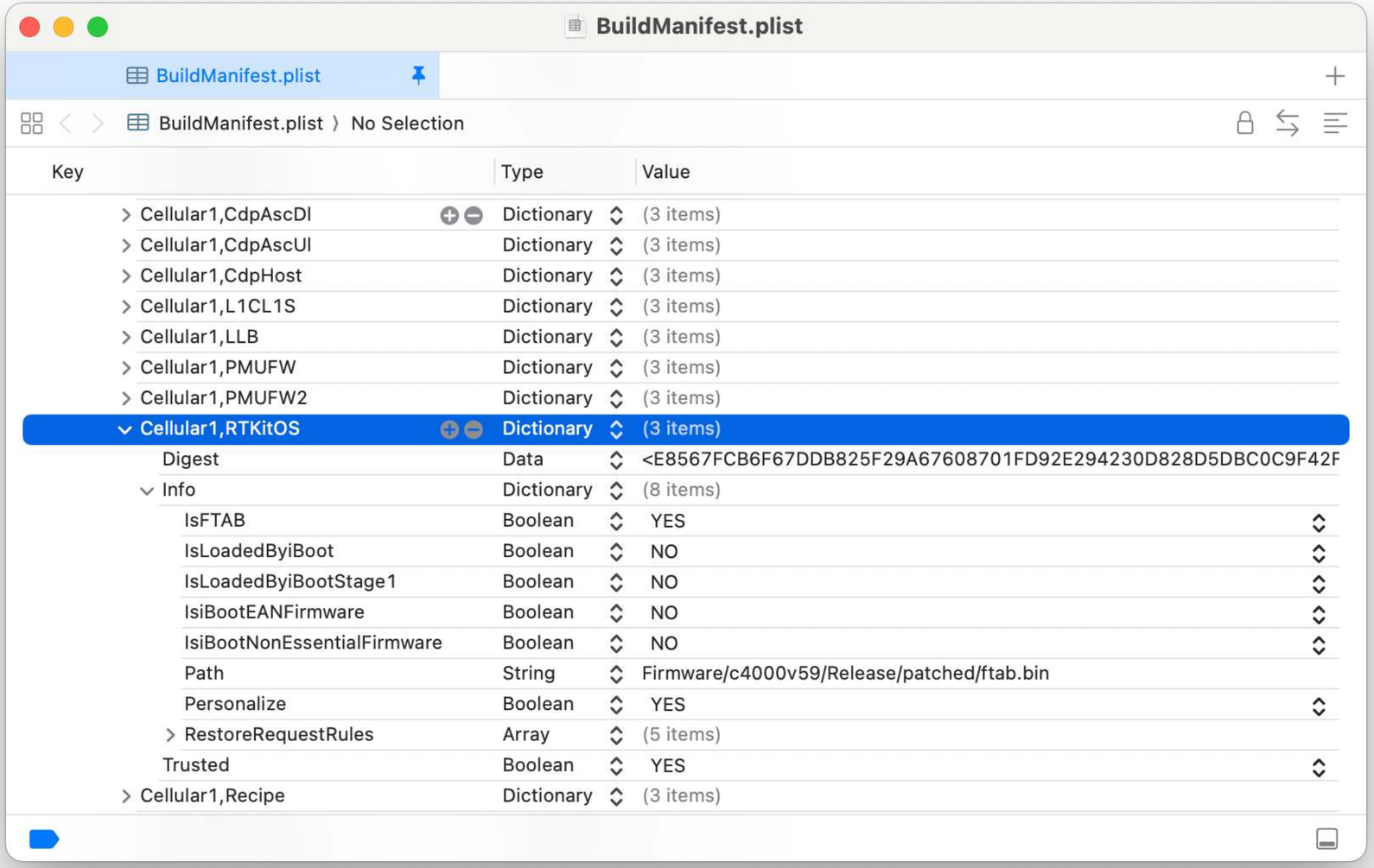
ari (CommCenter)	loopback	iqtrace
gnss (gpsd)	nvfs	hsfilter
vcom	user-pdp	mmapdebug
trace (abm-helper)	mediactl	kernel-datapacketpool (kernel)
at	rctth	kernel-dataplane (kernel)

Interface (process) - Most interfaces are not continuously connected

Firmware Architecture 

IPSW Build Manifest

References some baseband firmware binaries



Extracting C1 Firmware

Codenames: C4000, Sinope, Leda, INITIUM1

- Download IPSW or OTA for iPhone 16e or iPhone Air
- Extract file /Firmware/c4000v59/Release/patched/ftab.bin
  - Encoded with FTAB file format (Apple-custom)
  - Some FTAB segments are compressed with lzfse

```
struct FtabHeader {
    unk0: u32, // depending on device
    unk1: u32, // 0xFFFFFFFF
    unk2: u32, // 0
    unk3: u32, // 0
    ticket_offset: u32,
    ticket_length: u32,
    unk4: u32, // 0
    unk5: u32, // 0
    magic: [u8; 8], // "rkosftab"
    segment_count: u32,
    unk6: u32, // 0
}

struct FtabSegmentHeader {
    tag: [u8; 4], // 4-byte string
    offset: u32, // start of file
    length: u32,
    unk0: u32, // 0
}
```

```
$ ipsw download appledb --os iOS --latest --device "iPhone17,5" --release

$ ipsw extract --pattern "c4000v59" ./iPhone17,5_26.0.1_23A355_Restore.ipsw

$ ipsw fw c1 23A355__iPhone17,5/Firmware/c4000v59/Release/patched/ftab.bin
```

Firmware Architecture

Cores: 4x ARMv8, ARMv7, ARCv2

- **rkos:** Operating system & cellular protocol stack
- **cdpd:** Cellular data processing for downlink
- **cdpu:** Cellular data processing for uplink
- **l1cs:** Lower levels of cellular protocol stack
- **gns1:** Global navigation satellite system (ARCv2)
- **cdph:** PCIe host & power management (ARMv7)

File	Type	Size
CRnn	Data (lzfse, fwsg)	5.7 MB
RPnn	Data (lzfse, fwsg)	3.9 MB
R1nn	Data (lzfse, fwsg)	12 MB
R203	Data (lzfse, fwsg)	4.1 MB
rcpi	Recipe File Format	5 KB
bver	Version String	81 B
ibdt	iBootData	272 B
l1c2	Zeros	6 KB
ARC2	"empty"	7 KB
CAR2	0xFEEDCAFE	487 KB
CAR3	0xFEEDCAFE	487 KB
GNS1	ARCv2	480 KB
apmu	ARMv7	7 KB
pmfw	ARMv7	10 KB
illb	ARMv8	555 KB
cdph	ARMv7 (fwsg)	917 KB
cdpd	ARMv8 (fwsg)	15 MB
cdpu	ARMv8 (fwsg)	18 MB
rkos	ARMv8 (fwsg)	75 MB
l1cs	ARMv8 (fwsg)	68 MB

Loading C1 firmware

For Reverse Engineering

```
struct SegmentTableHeader {
    magic: [u8; 4],
    flag: u32,
    table_offset: u32,
    nr_entries: u32,
}
```

```
struct SegmentTableEntry {
    virtual_address: u32,
    space: u32,
    file_offset: u32,
    file_size: u32,
    memory_size: u32,
    flag: u32,
    name_bytes: [u8; 8],
}
```

- Extract firmware segments (fwsg) from header at EOF
  - Existing loaders for IDA from nlitsme and pwnlambda
    based on contributions by npupyshev
  - For some file_size != memory_size
- Use 32-bit memory virtual address space (even for Arm64)
- Set executable permissions based on flag (0x4 = code, 0x0 = data)
- C1 can also dynamically load segments and map them into multiple cores
- Segment names can appear multiple times, expand with data from binaries

Segments of C1 Firmware

Kernel, executable, and data segments

rkos #1

m.__TEXT
m.__DATA
m.__DATA_CONST
m.APPCONFIG
m.OSC_01
m.OSC_02
m.OSC_03
m.OSC_04
m.OSC_05
m.OSC_06
m.OSK_01
m.OSK_02
m.OSK_03
m.OSK_04
m.OSK_05
m.OSK_06
m.OSCSHR
m.OSCSH2
m.__PW_PERF_SEG
m.__MHOST_RNG_SEG
m.__RF_INIT_SEG
m.__IOSM_CSHM_SEG
m.__GTI_SHM_SEG
m.__CPS_DP_SEG
m.__CPS_FPDB_SEG
m.__EXPRO_SEG
m.__RT_SENSOR_SEG
m.__SENSOR_SEG
m.__SAH_CPS_SEG
m.__UE_CAP_SEG

rkos #2

m.__RO_DATA_SEG
m.__DRAM_NC_SEG
m.__DRAM_SHM_SEG
m.__CPS_ICC_SEG
m.__SHMS_NVM_SEG
m.__SHMC_NVM_SEG
m.__PBM_SHM_SEG
m.__MSG_CPS_SEG
m.__MSG2_CPS_SEG
m.__PS_FW_DB_SEG
m.__L1MEM_TRC_SEG
m.__TAGGR_MEM_SEG
m.__MC_SHM_SEG
m.__2G_DATA_SEG
m.__2G_CODE_SEG
m.__RO_NVM_SEG
m.__L2DPU_SEG
m.__NC_L2DPU_SEG
m.__RW_CPS
m.__3G_CODE_SEG
m.__3G_DATA_SEG
m.__OVERLAY_SEG
m.__OPBM_SHM_SEG
m.__CPSICC_S_SEG
m.__SRAM_TEXT_SEG
r.__TEXT
r.__DATA
KERNEL__TEXT
KERNEL__DATA

cdpu

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_UL_SEG
__MSG2_UL_SEG
__CDPULICC_SEG
__MCDPULICC_SEG
__OCDPULICC_SEG
__DRAM_L2FE_SEG
__MX_UL_DL_SEG
__DRAMUPTM_SEG
__UL_RSM_SEG
__CDPUL1_SEG
__CPMS_SHM_SEG
__DRAMUGDCI_SEG
__L2UL_L2PB_SEG
__ULPCM_S_SEG
__SRAM_L2FE_SEG
__ULICC_S_SEG
__SRAMUPTM_SEG
__SRAMUGDCI_SEG

cdpd

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_DL_SEG
__CDPDL3_SEG
__CDPDLICC_SEG
__MCDPDLICC_SEG
__OCDPDLICC_SEG
__L2_PDCP_SEG
__PDCPSN_SEG
__CDPDL2_SEG
__DL_PDCP_SEG
__L2_RLC_SEG
__L3DL_PTM_SEG
__RSM_SEG
__GDCI_SEG
__GDCISRAM_SEG
__GDUCSRAM_SEG
__DLICC_S_SEG

l1cs

__TEXT
__DATA
__DATA_CONST
__UC_TXT
__FW3G_SEG
__DRAM_NC_SEG
__RF_FR1_SEG
__RF_FR2_SEG
__GNSS_FW_SEG
__L1C_DP_SEG
__L1C_GNSS_SEG
__L1C_CPS_SEG
__L1C_3_SEG
__L1C_IPC_SEG
__STW_PCM_SEG
__MSG_L1C2_SEG
__MSG2_L1C_SEG
__L1C_ICC_SEG
__ML1C_ICC_SEG
__OL1C_ICC_SEG

cdph

__TEXT
__SRAM_TEXT_SEG
__SRAM_SEG
__SRAM_NC_SEG
__CHICC_S_SEG
__SRAM_PCM_SEG
__SRAM_CH_SEG
__DATA
__DRAM_T_SEG
__DRAM_D_SEG
__MSG2_CH_SEG
__L3UL_DRB_SEG
__CDPH_ICC_SEG
__DRAM_NC_SEG
__DRAM_PCM_SEG
__MSG_CHOST_SEG
__OCDPH_ICC_SEG
__MCDPH_ICC_SEG
__CDPH_1_SEG

Segments of C1 Firmware (OSCKit)

Kernel, executable, and data segments

rkos #1

m.__TEXT
m.__DATA
m.__DATA_CONST
m.APPCONFIG
m.OSC_01
m.OSC_02
m.OSC_03
m.OSC_04
m.OSC_05
m.OSC_06
m.OSK_01
m.OSK_02
m.OSK_03
m.OSK_04
m.OSK_05
m.OSK_06
m.OSCSHR
m.OSCSH2
m.__PW_PERF_SEG
m.__MHOST_RNG_SEG
m.__RF_INIT_SEG
m.__IOSM_CSHM_SEG
m.__GTI_SHM_SEG
m.__CPS_DP_SEG
m.__CPS_FPDB_SEG
m.__EXPRO_SEG
m.__RT_SENSOR_SEG
m.__SENSOR_SEG
m.__SAH_CPS_SEG
m.__UE_CAP_SEG

rkos #2

m.__RO_DATA_SEG
m.__DRAM_NC_SEG
m.__DRAM_SHM_SEG
m.__CPS_ICC_SEG
m.__SHMS_NVM_SEG
m.__SHMC_NVM_SEG
m.__PBM_SHM_SEG
m.__MSG_CPS_SEG
m.__MSG2_CPS_SEG
m.__PS_FW_DB_SEG
m.__L1MEM_TRC_SEG
m.__TAGGR_MEM_SEG
m.__MC_SHM_SEG
m.__2G_DATA_SEG
m.__2G_CODE_SEG
m.__RO_NVM_SEG
m.__L2DPU_SEG
m.__NC_L2DPU_SEG
m.__RW_CPS
m.__3G_CODE_SEG
m.__3G_DATA_SEG
m.__OVERLAY_SEG
m.__OPBM_SHM_SEG
m.__CPSICC_S_SEG
m.__SRAM_TEXT_SEG
r.__TEXT
r.__DATA
KERNEL__TEXT
KERNEL__DATA

cdpu

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_UL_SEG
__MSG2_UL_SEG
__CDPULICC_SEG
__MCDPULICC_SEG
__OCDPULICC_SEG
__DRAM_L2FE_SEG
__MX_UL_DL_SEG
__DRAMUPTM_SEG
__UL_RSM_SEG
__CDPUL1_SEG
__CPMS_SHM_SEG
__DRAMUGDCI_SEG
__L2UL_L2PB_SEG
__ULPCM_S_SEG
__SRAM_L2FE_SEG
__ULICC_S_SEG
__SRAMUPTM_SEG
__SRAMUGDCI_SEG

cdpd

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_DL_SEG
__CDPDL3_SEG
__CDPDLICC_SEG
__MCDPDLICC_SEG
__OCDPDLICC_SEG
__L2_PDCP_SEG
__PDCPSN_SEG
__CDPDL2_SEG
__DL_PDCP_SEG
__L2_RLC_SEG
__L3DL_PTM_SEG
__RSM_SEG
__GDCI_SEG
__GDCISRAM_SEG
__GDUCSRAM_SEG
__DLICC_S_SEG

l1cs

__TEXT
__DATA
__DATA_CONST
__UC_TXT
__FW3G_SEG
__DRAM_NC_SEG
__RF_FR1_SEG
__RF_FR2_SEG
__GNSS_FW_SEG
__L1C_DP_SEG
__L1C_GNSS_SEG
__L1C_CPS_SEG
__L1C_3_SEG
__L1C_IPC_SEG
__STW_PCM_SEG
__MSG_L1C2_SEG
__MSG2_L1C_SEG
__L1C_ICC_SEG
__ML1C_ICC_SEG
__OL1C_ICC_SEG

cdph

__TEXT
__SRAM_TEXT_SEG
__SRAM_SEG
__SRAM_NC_SEG
__CHICC_S_SEG
__SRAM_PCM_SEG
__SRAM_CH_SEG
__DATA
__DRAM_T_SEG
__DRAM_D_SEG
__MSG2_CH_SEG
__L3UL_DRB_SEG
__CDPH_ICC_SEG
__DRAM_NC_SEG
__DRAM_PCM_SEG
__MSG_CHOST_SEG
__OCDPH_ICC_SEG
__MCDPH_ICC_SEG
__CDPH_1_SEG

Segments of C1 Firmware (Cellular)

Kernel, executable, and data segments

rkos #1

m.__TEXT
m.__DATA
m.__DATA_CONST
m.APPCONFIG
m.OSC_01
m.OSC_02
m.OSC_03
m.OSC_04
m.OSC_05
m.OSC_06
m.OSK_01
m.OSK_02
m.OSK_03
m.OSK_04
m.OSK_05
m.OSK_06
m.OSCSHR
m.OSCSH2
m.__PW_PERF_SEG
m.__MHOST_RNG_SEG
m.__RF_INIT_SEG
m.__IOSM_CSHM_SEG
m.__GTI_SHM_SEG
m.__CPS_DP_SEG
m.__CPS_FPDB_SEG
m.__EXPRO_SEG
m.__RT_SENSOR_SEG
m.__SENSOR_SEG
m.__SAH_CPS_SEG
m.__UE_CAP_SEG

rkos #2

m.__RO_DATA_SEG
m.__DRAM_NC_SEG
m.__DRAM_SHM_SEG
m.__CPS_ICC_SEG
m.__SHMS_NVM_SEG
m.__SHMC_NVM_SEG
m.__PBM_SHM_SEG
m.__MSG_CPS_SEG
m.__MSG2_CPS_SEG
m.__PS_FW_DB_SEG
m.__L1MEM_TRC_SEG
m.__TAGGR_MEM_SEG
m.__MC_SHM_SEG
m.__2G_DATA_SEG
m.__2G_CODE_SEG
m.__RO_NVM_SEG
m.__L2DPU_SEG
m.__NC_L2DPU_SEG
m.__RW_CPS
m.__3G_CODE_SEG
m.__3G_DATA_SEG
m.__OVERLAY_SEG
m.__OPBM_SHM_SEG
m.__CPSICC_S_SEG
m.__SRAM_TEXT_SEG
r.__TEXT
r.__DATA
KERNEL__TEXT
KERNEL__DATA

cdpu

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_UL_SEG
__MSG2_UL_SEG
__CDPULICC_SEG
__MCDPULICC_SEG
__OCDPULICC_SEG
__DRAM_L2FE_SEG
__MX_UL_DL_SEG
__DRAMUPTM_SEG
__UL_RSM_SEG
__CDPUL1_SEG
__CPMS_SHM_SEG
__DRAMUGDCI_SEG
__L2UL_L2PB_SEG
__ULPCM_S_SEG
__SRAM_L2FE_SEG
__ULICC_S_SEG
__SRAMUPTM_SEG
__SRAMUGDCI_SEG

cdpd

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_DL_SEG
__CDPDL3_SEG
__CDPDLICC_SEG
__MCDPDLICC_SEG
__OCDPDLICC_SEG
__L2_PDCP_SEG
__PDCPSN_SEG
__CDPDL2_SEG
__DL_PDCP_SEG
__L2_RLC_SEG
__L3DL_PTM_SEG
__RSM_SEG
__GDCI_SEG
__GDCISRAM_SEG
__GDUCSRAM_SEG
__DLICC_S_SEG

l1cs

__TEXT
__DATA
__DATA_CONST
__UC_TXT
__FW3G_SEG
__DRAM_NC_SEG
__RF_FR1_SEG
__RF_FR2_SEG
__GNSS_FW_SEG
__L1C_DP_SEG
__L1C_GNSS_SEG
__L1C_CPS_SEG
__L1C_3_SEG
__L1C_IPC_SEG
__STW_PCM_SEG
__MSG_L1C2_SEG
__MSG2_L1C_SEG
__L1C_ICC_SEG
__ML1C_ICC_SEG
__OL1C_ICC_SEG

cdph

__TEXT
__SRAM_TEXT_SEG
__SRAM_SEG
__SRAM_NC_SEG
__CHICC_S_SEG
__SRAM_PCM_SEG
__SRAM_CH_SEG
__DATA
__DRAM_T_SEG
__DRAM_D_SEG
__MSG2_CH_SEG
__L3UL_DRB_SEG
__CDPH_ICC_SEG
__DRAM_NC_SEG
__DRAM_PCM_SEG
__MSG_CHOST_SEG
__OCDPH_ICC_SEG
__MCDPH_ICC_SEG
__CDPH_1_SEG



No NR FR2 (mmWave) radio chain in schematics.
Is Apple already testing mmWave internally?

Segments of C1 Firmware (Memory)

Kernel, executable, and data segments

rkos #1

m.__TEXT
m.__DATA
m.__DATA_CONST
m.APPCONFIG
m.OSC_01
m.OSC_02
m.OSC_03
m.OSC_04
m.OSC_05
m.OSC_06
m.OSK_01
m.OSK_02
m.OSK_03
m.OSK_04
m.OSK_05
m.OSK_06
m.OSCSHR
m.OSCSH2
m.__PW_PERF_SEG
m.__MHOST_RNG_SEG
m.__RF_INIT_SEG
m.__IOSM_CSHM_SEG
m.__GTI_SHM_SEG
m.__CPS_DP_SEG
m.__CPS_FPDB_SEG
m.__EXPRO_SEG
m.__RT_SENSOR_SEG
m.__SENSOR_SEG
m.__SAH_CPS_SEG
m.__UE_CAP_SEG

rkos #2

m.__RO_DATA_SEG
m.__DRAM_NC_SEG
m.__DRAM_SHM_SEG
m.__CPS_ICC_SEG
m.__SHMS_NVM_SEG
m.__SHMC_NVM_SEG
m.__PBM_SHM_SEG
m.__MSG_CPS_SEG
m.__MSG2_CPS_SEG
m.__PS_FW_DB_SEG
m.__L1MEM_TRC_SEG
m.__TAGGR_MEM_SEG
m.__MC_SHM_SEG
m.__2G_DATA_SEG
m.__2G_CODE_SEG
m.__RO_NVM_SEG
m.__L2DPU_SEG
m.__NC_L2DPU_SEG
m.__RW_CPS
m.__3G_CODE_SEG
m.__3G_DATA_SEG
m.__OVERLAY_SEG
m.__OPBM_SHM_SEG
m.__CPSICC_S_SEG
m.__SRAM_TEXT_SEG
r.__TEXT
r.__DATA
KERNEL__TEXT
KERNEL__DATA

cdpu

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_UL_SEG
__MSG2_UL_SEG
__CDPULICC_SEG
__MCDPULICC_SEG
__OCDPULICC_SEG
__DRAM_L2FE_SEG
__MX_UL_DL_SEG
__DRAMUPTM_SEG
__UL_RSM_SEG
__CDPUL1_SEG
__CPMS_SHM_SEG
__DRAMUGDCI_SEG
__L2UL_L2PB_SEG
__ULPCM_S_SEG
__SRAM_L2FE_SEG
__ULICC_S_SEG
__SRAMUPTM_SEG
__SRAMUGDCI_SEG

cdpd

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_DL_SEG
__CDPDL3_SEG
__CDPDLICC_SEG
__MCDPDLICC_SEG
__OCDPDLICC_SEG
__L2_PDCP_SEG
__PDCPSN_SEG
__CDPDL2_SEG
__DL_PDCP_SEG
__L2_RLC_SEG
__L3DL_PTM_SEG
__RSM_SEG
__GDCI_SEG
__GDCISRAM_SEG
__GDUCSRAM_SEG
__DLICC_S_SEG

l1cs

__TEXT
__DATA
__DATA_CONST
__UC_TXT
__FW3G_SEG
__DRAM_NC_SEG
__RF_FR1_SEG
__RF_FR2_SEG
__GNSS_FW_SEG
__L1C_DP_SEG
__L1C_GNSS_SEG
__L1C_CPS_SEG
__L1C_3_SEG
__L1C_IPC_SEG
__STW_PCM_SEG
__MSG_L1C2_SEG
__MSG2_L1C_SEG
__L1C_ICC_SEG
__ML1C_ICC_SEG
__OL1C_ICC_SEG

cdph

__TEXT
__SRAM_TEXT_SEG
__SRAM_SEG
__SRAM_NC_SEG
__CHICC_S_SEG
__SRAM_PCM_SEG
__SRAM_CH_SEG
__DATA
__DRAM_T_SEG
__DRAM_D_SEG
__MSG2_CH_SEG
__L3UL_DRB_SEG
__CDPH_ICC_SEG
__DRAM_NC_SEG
__DRAM_PCM_SEG
__MSG_CHOST_SEG
__OCDPH_ICC_SEG
__MCDPH_ICC_SEG
__CDPH_1_SEG

As seen before SRAM, DRAM, NVM (EEPROM)

Segments of C1 Firmware (Cross-Core)

Kernel, executable, and data segments

rkos #1

m.__TEXT
m.__DATA
m.__DATA_CONST
m.APPCONFIG
m.OSC_01
m.OSC_02
m.OSC_03
m.OSC_04
m.OSC_05
m.OSC_06
m.OSK_01
m.OSK_02
m.OSK_03
m.OSK_04
m.OSK_05
m.OSK_06
m.OSCSHR
m.OSCSH2
m.__PW_PERF_SEG
m.__MHOST_RNG_SEG
m.__RF_INIT_SEG
m.__IOSM_CSHM_SEG
m.__GTI_SHM_SEG
m.__CPS_DP_SEG
m.__CPS_FPDB_SEG
m.__EXPRO_SEG
m.__RT_SENSOR_SEG
m.__SENSOR_SEG
m.__SAH_CPS_SEG
m.__UE_CAP_SEG

rkos #2

m.__RO_DATA_SEG
m.__DRAM_NC_SEG
m.__DRAM_SHM_SEG
m.__CPS_ICC_SEG
m.__SHMS_NVM_SEG
m.__SHMC_NVM_SEG
m.__PBM_SHM_SEG
m.__MSG_CPS_SEG
m.__MSG2_CPS_SEG
m.__PS_FW_DB_SEG
m.__L1MEM_TRC_SEG
m.__TAGGR_MEM_SEG
m.__MC_SHM_SEG
m.__2G_DATA_SEG
m.__2G_CODE_SEG
m.__RO_NVM_SEG
m.__L2DPU_SEG
m.__NC_L2DPU_SEG
m.__RW_CPS
m.__3G_CODE_SEG
m.__3G_DATA_SEG
m.__OVERLAY_SEG
m.__OPBM_SHM_SEG
m.__CPSICC_S_SEG
m.__SRAM_TEXT_SEG
r.__TEXT
r.__DATA
KERNEL__TEXT
KERNEL__DATA

cdpu

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_UL_SEG
__MSG2_UL_SEG
__CDPULICC_SEG
__MCDPULICC_SEG
__OCDPULICC_SEG
__DRAM_L2FE_SEG
__MX_UL_DL_SEG
__DRAMUPTM_SEG
__UL_RSM_SEG
__CDPUL1_SEG
__CPMS_SHM_SEG
__DRAMUGDCI_SEG
__L2UL_L2PB_SEG
__ULPCM_S_SEG
__SRAM_L2FE_SEG
__ULICC_S_SEG
__SRAMUPTM_SEG
__SRAMUGDCI_SEG

cdpd

__TEXT
__DATA
__DATA_CONST
__DRAM_NC_SEG
__MSG_DL_SEG
__CDPDL3_SEG
__CDPDLICC_SEG
__MCDPDLICC_SEG
__OCDPDLICC_SEG
__L2_PDCP_SEG
__PDCPSN_SEG
__CDPDL2_SEG
__DL_PDCP_SEG
__L2_RLC_SEG
__L3DL_PTM_SEG
__RSM_SEG
__GDCI_SEG
__GDCISRAM_SEG
__GDUCSRAM_SEG
__DLICC_S_SEG

l1cs

__TEXT
__DATA
__DATA_CONST
__UC_TXT
__FW3G_SEG
__DRAM_NC_SEG
__RF_FR1_SEG
__RF_FR2_SEG
__GNSS_FW_SEG
__L1C_DP_SEG
__L1C_GNSS_SEG
__L1C_CPS_SEG
__L1C_3_SEG
__L1C_IPC_SEG
__STW_PCM_SEG
__MSG_L1C2_SEG
__MSG2_L1C_SEG
__L1C_ICC_SEG
__ML1C_ICC_SEG
__OL1C_ICC_SEG

cdph

__TEXT
__SRAM_TEXT_SEG
__SRAM_SEG
__SRAM_NC_SEG
__CHICC_S_SEG
__SRAM_PCM_SEG
__SRAM_CH_SEG
__DATA
__DRAM_T_SEG
__DRAM_D_SEG
__MSG2_CH_SEG
__L3UL_DRB_SEG
__CDPH_ICC_SEG
__DRAM_NC_SEG
__DRAM_PCM_SEG
__MSG_CHOST_SEG
__OCDPH_ICC_SEG
__MCDPH_ICC_SEG
__CDPH_1_SEG

Segments that may be loaded by rkos

SHM (Shared Memory), ICC (Interrupts), MSG (Messages)

Significant Changes since Release

Contents of bver.bin

First C1 firmware shipped with iOS 18.3.1 -> iPhone 16e with C1

```
223.225608003000000.2019|22.3.0.0|BBFW:1.01.13|date:2025 2 17
```



- High amount of changes in binaries
- File sizes increasing with each update
- Overarching concepts remain the same

- Split of m.OSC in m.OSC & m.OSK segments
- New segments for 3G stack, …

Latest C1 firmware shipped in iOS 26.0.1 -> iPhone 16e (C1) & iPhone Air (C1X)

```
230.267500007600000.10003|23.0.0.0|BBFW:2.00.09|date:2025 9 25|
chip_revision_b0
```



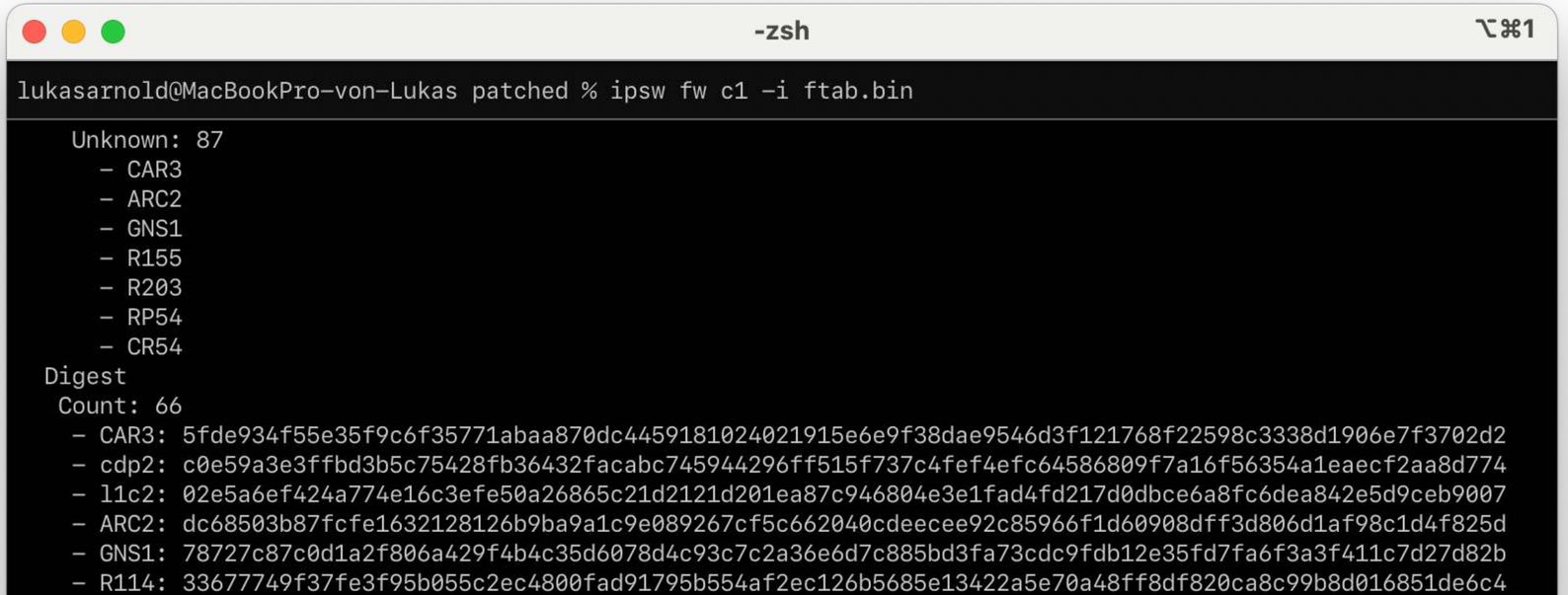
Different revisions of C1 chip in iPhones,
one firmware supports them all

Security Model & Mitigations 🔐

iBoot Boot Chain

Low Level Bootloader

- Perform extraction of files from ftab.bin
- Checks hashes of firmware files with rcpi.bin
- Assembles GNSS configuration (RXXX, RPXX, CRXX)



cL4 Kernel

__TEXT and __DATA segments

rkos
cdpu,cdpd,l1cs
chph

- Apple’s custom implementation of a L4-based microkernel with capability-based system calls
- Executes in EL1 and locks VA upper (TTBR1_EL1) for its use
- Locks two text regions to read-only (CTRR_A & CTRR_B)
- cL4 also has been seen in Apple Exclaves

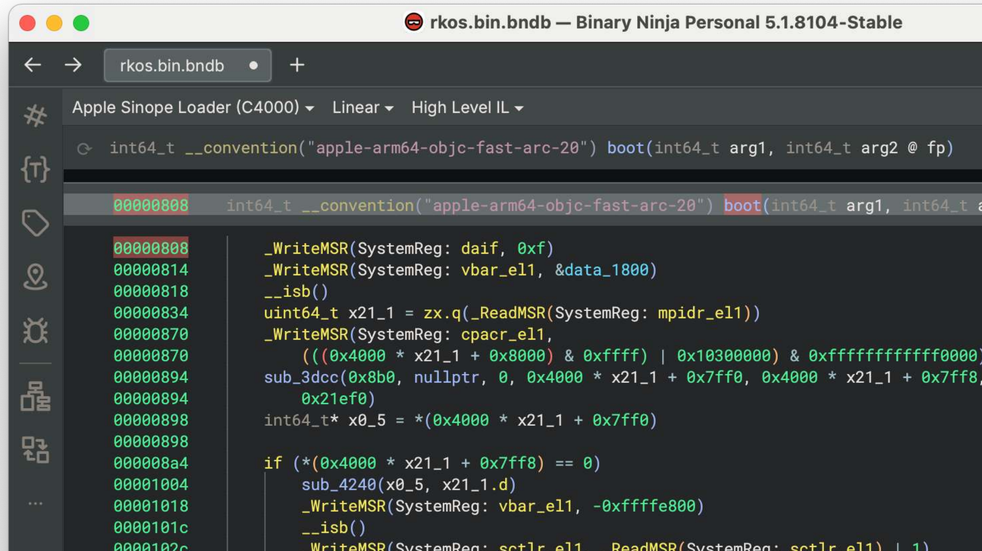
29 Apr 2024

Apple joins the seL4 Foundation

The seL4 Foundation is pleased to welcome **Apple** as our latest **member**.

We are excited to see their interest in seL4 and look forward to seeing their work with seL4.





Roottask

r.__TEXT and r.__DATA segments

rkos
cdpu,cdpd,l1cs
chph

- OSCKit = Operating System (Component | Compartment) Kit
- Processes boot arguments and prepares global memory Including PAC & ASLR
- Loads OSCKit segments `m.__TEXT, m.__DATA, m.__DATA_CONST, m. APPCONFIG, m.OS*`
- Creates processes for three compartments, maps IRQs, and launches them





Compartments

rkos
cdpu,cdpd,l1cs
chph

m.APPCONFIG segment defines 3 compartments



- Platform (PID 1) - Manages hardware and provides OS functions
- OverTheAir (PID 2) - Processes and generates over the air packets
- Middleware (PID 3) - Limited hardware access for some cellular features
- Communication between compartments via well-defined OSCKit RPC
- Each compartment has a set of startup tasks and then initializes long running tasks via the boot & mode manager (bootman & modeman)
- Some functions behave differently based on caller / enforce correct compartment

Common ARM64 Mitigations

PAC and ASLR are applied on all ARM64 cores

rkos
cdpu,cdpd,l1cs
chph

prepare memory, then enter user space

- Other cores (cdpu cdpd, l1cs) boot in simple RTKit manner
- Ensure pointer integrity with Apple-custom PAC implementation
  - 10k of 29k functions pointers in rkos binary have PAC diversity data
- Address Space Layout Randomization is active
  - Offset (slide) is unique for each core
  - Slides of other cores are provided via patchbays
- No indication of enhanced memory tagging extensions (yet?)



TLV structures (4B name, 4B length) in firmware, populated with data before booting.

SLID (rkos), RSL1 (l1cs),
RSCU (cdpu), RSCD (cdpd)

Assertion Statements

Ensure preconditions of are met

- Check expected state and if not met trigger a crash
- Baseband will enter spinlock, iOS watchdog will detect unresponsiveness, i.e. crash, and restart baseband
- Intermediate assertion functions for various subsystems
- Stack guards are a special type of assertion failure
- Useful for RE: Often contain message of fault reason and name of file or even full file path

/Library/Caches/com.apple.xbs/Sources/INITIUM1BaseBandFW_osc

```
├── cellular_protocol_stack
│   ├── 3p_src
│   ├── 3p_tools
│   ├── as
│   ├── cps_api
│   ├── dp
│   ├── nas
│   ├── services
│   └── xps
├── common
│   ├── platform_sw
│   ├── protocol_stack
│   └── system-build
├── firmware_bb
│   ├── l1c
│   └── rfsw
├── modem
│   └── msw_platform_services
├── output
│   └── INITIUM1
├── platform_sw
│   ├── analytics
│   ├── audio
│   ├── boot_and_mode_manager
│   ├── core
│   ├── external
│   ├── factory
│   ├── gti
│   ├── hwid
│   ├── ipc
│   ├── l1c_l1cs
│   ├── mccr
│   ├── memory
│   ├── mri
│   ├── os_frameworks
│   ├── power
│   ├── sah.v3
│   ├── security
│   ├── sim
│   └── trace
├── roottask
│   └── src
├── src
│   └── libc
├── stewie_stack
│   ├── l1c
│   └── protocol_stack
└── vinyl
    ├── sim_arb
    └── src
```

How is the security model holding up?

Only one 8.1 CVSS CVE so far due to a logic bug

Baseband

Available for: iPhone 16e

Impact: An attacker in a privileged network position may be able to intercept network traffic

Description: This issue was addressed through improved state management.

CVE-2025-31214: 秦若涵, 崔志伟, and 崔宝江

Ruohan Qin, Zhiwei Cui, and Baojiang Cui
Beijing University of Posts and Telecommunications

Fixed in iOS 18.5 (https://support.apple.com/en-us/122404)

Apple puts Money behind their Claims

https://security.apple.com/blog/apple-security-bounty-evolved/

> We believe the **architectural improvements and enhanced security in devices with C1(X) modems make them the most secure in the industry**, making proximity-based attacks more challenging to execute than ever.
>
> We've never observed a real-world, zero-click attack executed purely through wireless proximity.

Wireless proximity attack: Attack requiring physical proximity to device

$250K ⟹ $1M

Limitations of Hacking with C1

Approaches besides static analysis

```
{{~~  /-----\
{{~~ /       \
{{~~|         |
{{~~| S T O P |
{{~~|         |
{{~~ \       /
{{~~  \-----/
Copyright(C) 2001
Apple Computer, Inc.
```

From C1's iBoot

- Ways to interact with the baseband on an iPhone
  - Extract baseband data from sysdiagnoses (often empty)
  - Extract baseband (ARI) packets from syslog
  - Send over-the-air packets and hope for a crash
- No jailbreaks for iPhone 16e & Air available – none expected in the future
- No SRD with C1 baseband – fingers crossed for next year
- Best approach for dynamic analysis: emulation and native rehosting



Thanks!

We’re continuing our work

SECURE MOBILE NETWORKING

Research by

Lukas Arnold[1], Lucas Becker[1], Nils Rollshausen[1], Dyon Goos[2], Jiska Classen[3], Marius Muench[4]

1 TU Darmstadt - **2** VU Amsterdam - **3** Hasso-Plattner-Institut - **4** University of Birmingham

Contact Lukas

📪 larnold@seemoo.tu-darmstadt.de
🦣 @lukasarnld@mastodon.social
🌐 lukasarnold.de

Open-Source Projects



CellGuard



BaseTrace