# BOOTROM

Small code block executed first

- Initializes the system
- Handles transition to application/customer code
- Immutable (some exceptions)
    - Manufacturer only “patch” section
    - Focused-ion-beam (FIB)

Security Implemented Here

- Secureboot
- OTP
- Debug/Readback protections

Notable bug examples

- iPhone bugs (limera1n/checkm8)

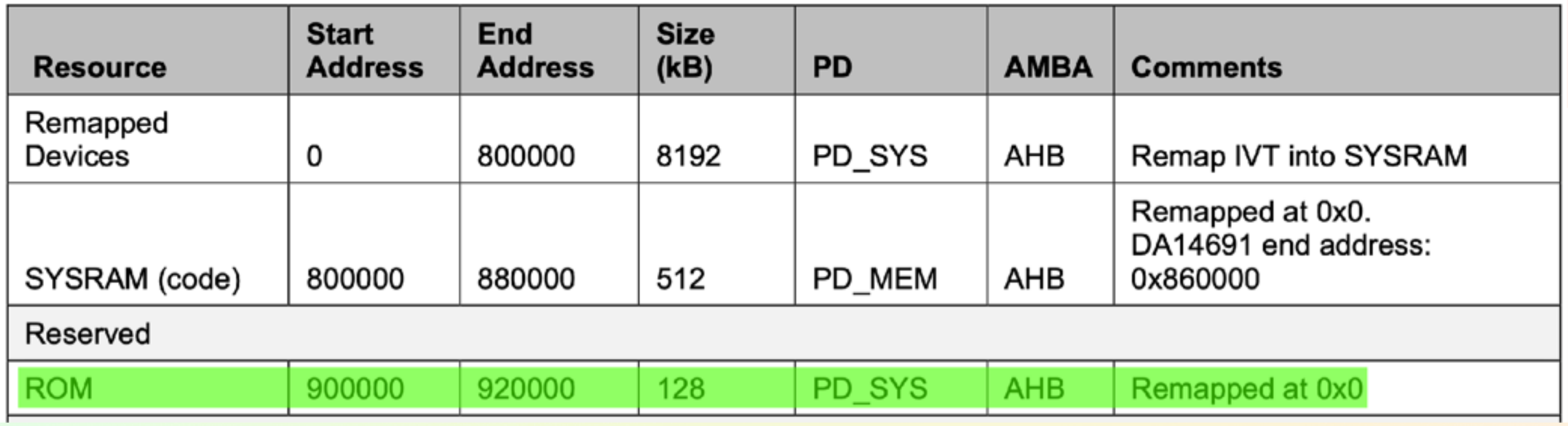
# EXTRACTION

**Datasheet provides the memory mappings**

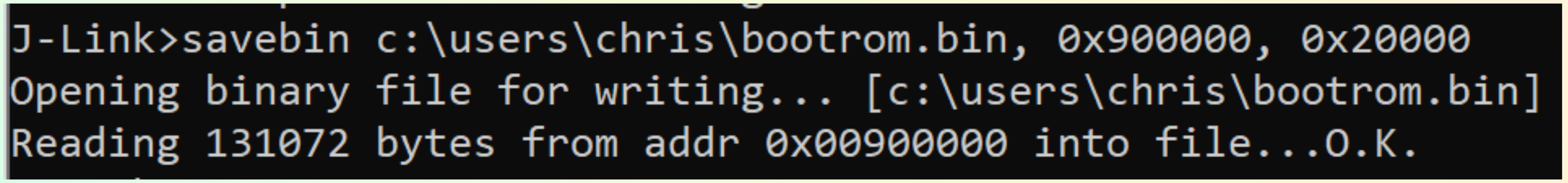
**Debugger/JLink to read to a file**

**Load into IDA**

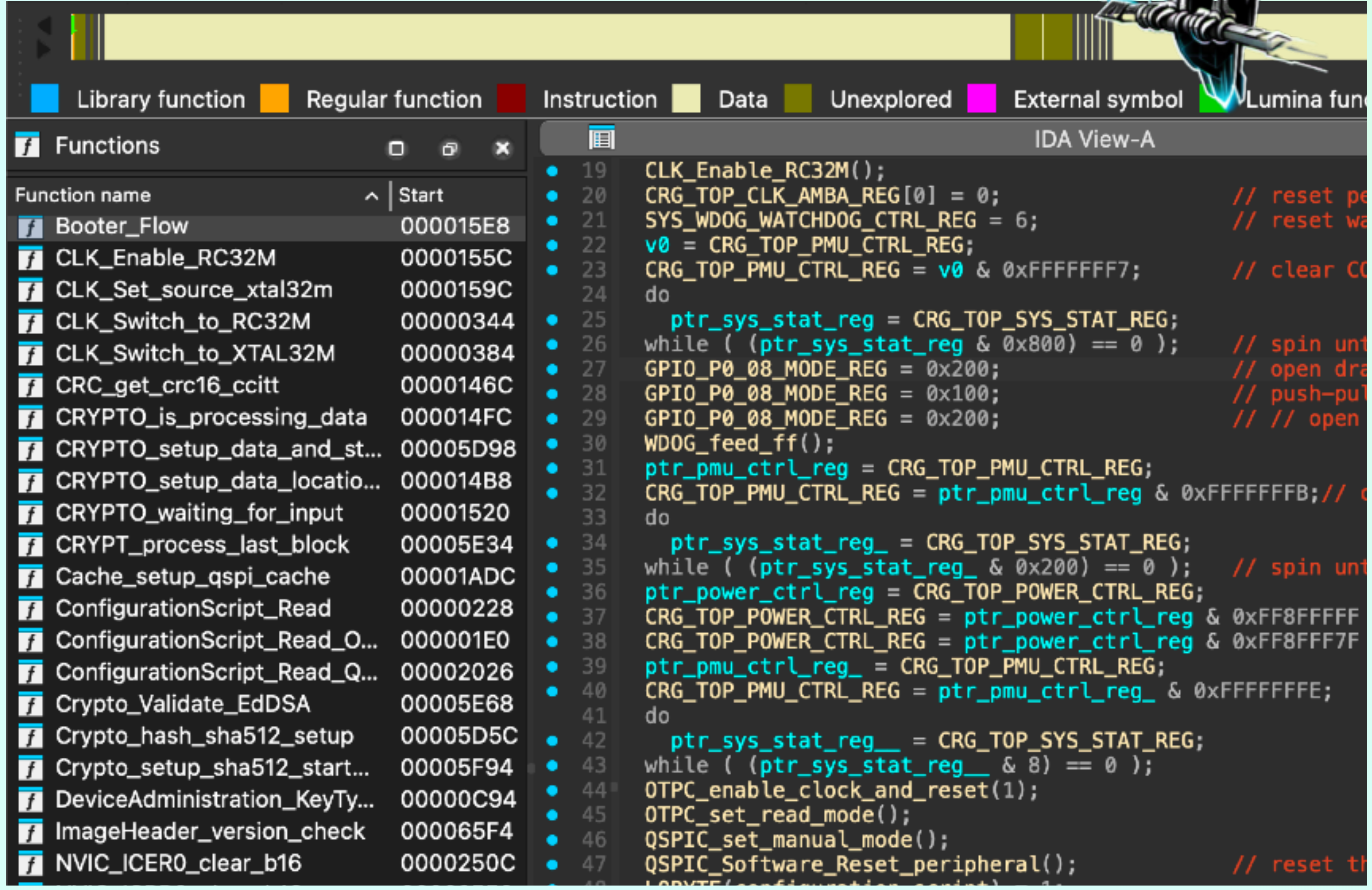
**Draw the rest of the owl**



| Resource | Start Address | End Address | Size (kB) | PD | AMBA | Comments |
|---|---|---|---|---|---|---|
| Remapped Devices | 0 | 800000 | 8192 | PD_SYS | AHB | Remap IVT into SYSRAM |
| SYSRAM (code) | 800000 | 880000 | 512 | PD_MEM | AHB | Remapped at 0x0.<br>DA14691 end address:<br>0x860000 |
| Reserved | | | | | | |
| ROM | 900000 | 920000 | 128 | PD_SYS | AHB | Remapped at 0x0 |

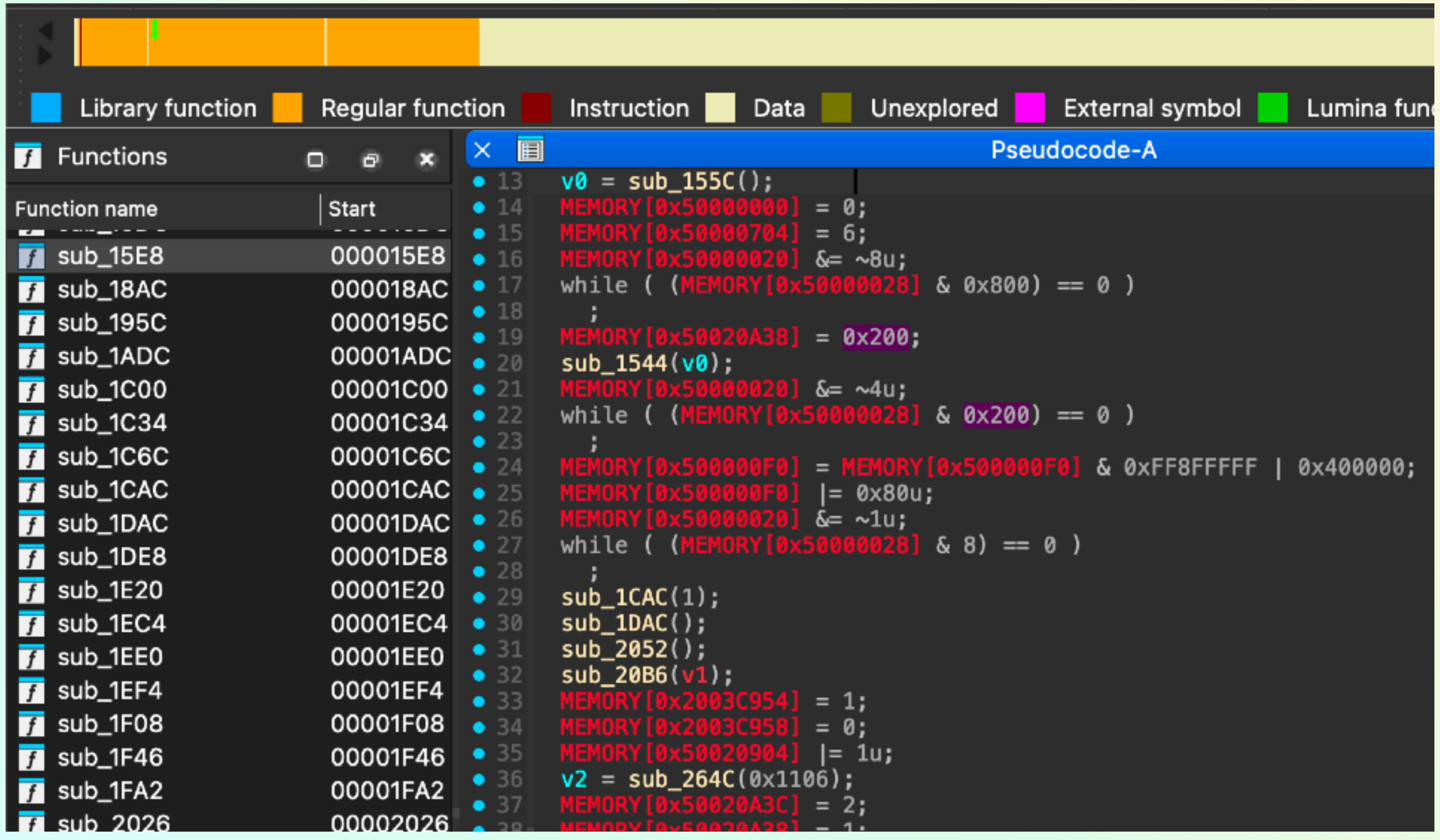


```
J-Link>savebin c:\users\chris\bootrom.bin, 0x900000, 0x20000
Opening binary file for writing... [c:\users\chris\bootrom.bin]
Reading 131072 bytes from addr 0x00900000 into file...O.K.
```



| Function name | Start |
|---|---|
| Booter_Flow | 000015E8 |
| CLK_Enable_RC32M | 0000155C |
| CLK_Set_source_xtal32m | 0000159C |
| CLK_Switch_to_RC32M | 00000344 |
| CLK_Switch_to_XTAL32M | 00000384 |
| CRC_get_crc16_ccitt | 0000146C |
| CRYPTO_is_processing_data | 000014FC |
| CRYPTO_setup_data_and_st... | 00005D98 |
| CRYPTO_setup_data_locatio... | 000014B8 |
| CRYPTO_waiting_for_input | 00001520 |
| CRYPT_process_last_block | 00005E34 |
| Cache_setup_qspi_cache | 00001ADC |
| ConfigurationScript_Read | 00000228 |
| ConfigurationScript_Read_O... | 000001E0 |
| ConfigurationScript_Read_Q... | 00002026 |
| Crypto_Validate_EdDSA | 00005E68 |
| Crypto_hash_sha512_setup | 00005D5C |
| Crypto_setup_sha512_start... | 00005F94 |
| DeviceAdministration_KeyTy... | 00000C94 |
| ImageHeader_version_check | 000065F4 |
| NVIC_ICER0_clear_b16 | 0000250C |

```
CLK_Enable_RC32M();
CRG_TOP_CLK_AMBA_REG[0] = 0;                        // reset pe
SYS_WDOG_WATCHDOG_CTRL_REG = 6;                     // reset wa
v0 = CRG_TOP_PMU_CTRL_REG;
CRG_TOP_PMU_CTRL_REG = v0 & 0xFFFFFFF7;             // clear CC
do
  ptr_sys_stat_reg = CRG_TOP_SYS_STAT_REG;
while ( (ptr_sys_stat_reg & 0x800) == 0 );          // spin unt
GPIO_P0_08_MODE_REG = 0x200;                        // open dra
GPIO_P0_08_MODE_REG = 0x100;                        // push-pul
GPIO_P0_08_MODE_REG = 0x200;                        // // open
WDOG_feed_ff();
ptr_pmu_ctrl_reg = CRG_TOP_PMU_CTRL_REG;
CRG_TOP_PMU_CTRL_REG = ptr_pmu_ctrl_reg & 0xFFFFFFFB;// c
do
  ptr_sys_stat_reg_ = CRG_TOP_SYS_STAT_REG;
while ( (ptr_sys_stat_reg_ & 0x200) == 0 );         // spin unt
ptr_power_ctrl_reg = CRG_TOP_POWER_CTRL_REG;
CRG_TOP_POWER_CTRL_REG = ptr_power_ctrl_reg & 0xFF8FFFFF
CRG_TOP_POWER_CTRL_REG = ptr_power_ctrl_reg & 0xFF8FFF7F
ptr_pmu_ctrl_reg_ = CRG_TOP_PMU_CTRL_REG;
CRG_TOP_PMU_CTRL_REG = ptr_pmu_ctrl_reg_ & 0xFFFFFFFE;
do
  ptr_sys_stat_reg__ = CRG_TOP_SYS_STAT_REG;
while ( (ptr_sys_stat_reg__ & 8) == 0 );
OTPC_enable_clock_and_reset(1);
OTPC_set_read_mode();
QSPIC_set_manual_mode();
QSPIC_Software_Reset_peripheral();                  // reset th
```



| Function name | Start |
|---|---|
| sub_15E8 | 000015E8 |
| sub_18AC | 000018AC |
| sub_195C | 0000195C |
| sub_1ADC | 00001ADC |
| sub_1C00 | 00001C00 |
| sub_1C34 | 00001C34 |
| sub_1C6C | 00001C6C |
| sub_1CAC | 00001CAC |
| sub_1DAC | 00001DAC |
| sub_1DE8 | 00001DE8 |
| sub_1E20 | 00001E20 |
| sub_1EC4 | 00001EC4 |
| sub_1EE0 | 00001EE0 |
| sub_1EF4 | 00001EF4 |
| sub_1F08 | 00001F08 |
| sub_1F46 | 00001F46 |
| sub_1FA2 | 00001FA2 |
| sub_2026 | 00002026 |

```
v0 = sub_155C();
MEMORY[0x50000000] = 0;
MEMORY[0x50000704] = 6;
MEMORY[0x50000020] &= ~8u;
while ( (MEMORY[0x50000028] & 0x800) == 0 )
  ;
MEMORY[0x50020A38] = 0x200;
sub_1544(v0);
MEMORY[0x50000020] &= ~4u;
while ( (MEMORY[0x50000028] & 0x200) == 0 )
  ;
MEMORY[0x500000F0] = MEMORY[0x500000F0] & 0xFF8FFFFF | 0x400000;
MEMORY[0x500000F0] |= 0x80u;
MEMORY[0x50000020] &= ~1u;
while ( (MEMORY[0x50000028] & 8) == 0 )
  ;
sub_1CAC(1);
sub_1DAC();
sub_2052();
sub_20B6(v1);
MEMORY[0x2003C954] = 1;
MEMORY[0x2003C958] = 0;
MEMORY[0x50020904] |= 1u;
v2 = sub_264C(0x1106);
MEMORY[0x50020A3C] = 2;
```