## Bootmode

SRC->SBMR2 BIT[25:24]

0x400f801c

## Boot device

0x400f8004

Low value : SRC->SBMR1 bit[5:4]

High value : SRC->SBMR1 bit[7:6]

switch(high\_value)

{

case 0:

switch(low\_value)

{

case 0:

flash\_device = FLASH\_TYPE\_FLEXSPI\_NOR; // FlexSPI NOR

break;

case 1:

flash\_device = FLASH\_TYPE\_SEMC\_NOR; // SEMC NOR

break;

default: // 2 or 3: SD

flash\_device = FLASH\_TYPE\_MMC\_SD;

break;

}

break;

case 1:

flash\_device = FLASH\_TYPE\_SEMC\_NAND; // SEMC NAND

break;

case 2:

flash\_device = FLASH\_TYPE\_MMC\_SD; // MMC/eMMC

break;

case 3:

flash\_device = FLASH\_TYPE\_FLEXSPI\_NAND; // FlexSPI NAND

break;

}

## FUSE

OCOTP\_BASE : 0x401f4000

Fuse 0x460 : 0x401f4460

## Persist register

SRC->gpr[9]

SRC base: 0x400F8000

0x400f8044

## NAND

### Search count

searchCount = 1 << (ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_VALUE());

#define ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_SHIFT ((UINT8)5)

#define ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_MASK (1U << ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_SHIFT)

#define ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_VALUE() \

((SRC->SBMR1&ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_MASK) >> ROM\_OCOTP\_BOOT\_SEARCH\_COUNT\_SHIFT)

1 << SBMR1 bit[5]得到search count （以block为单位）

ROM 会从0 block开始读FCB size的数据。。然后check数据是否为合法的FCB（tag，version，CRC）。

如果合法，则返回

非法，在search count范围内，每次增加search stride长度进行FCB的查找

一个page 是2048 byte，一个block 64 page

**Fuse里面有定义如下：**

**BOOT\_SEARCH\_STRIDE:**  
Search Stride for FCB and DBBT (in terms of pages)  
0 - 64  
1 - 128  
2 - 256  
3 – 32

找到FCB后，ROM会根据FCB重新初始化NAND的参数

FCB跟DBBT的查找操作都是在init函数里面实现的

### Search stride

#define ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_SHIFT ((UINT8)0)

#define ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_MASK (3U << ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_SHIFT)

#define ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_VALUE() \

((SRC->SBMR1&ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_MASK) >> ROM\_OCOTP\_BOOT\_SEARCH\_STRIDE\_SHIFT)

### 数据copy

根据FCB配合persist寄存器，persist寄存器的值是2， 则使用firmwareTable[2]里面定义的block。将指定block里面的前4096数据读取到0x20208000处，定义一个4byte长度的指针，指向IVT。

然后进入到认证function，

hab\_image\_entry = hab\_rvt.authenticate\_image(HAB\_CID\_ROM,

driver\_data->ivt\_offset,

(void\*\*)(&start),

&bytes,

&pu\_irom\_hwcnfg\_setup);

在这个函数里面会将数据根据IVT拷贝到指定的RAM，拷贝的函数如下

static hab\_status\_t pu\_irom\_hwcnfg\_setup(void \*\*start, size\_t \*bytes, const void \*boot\_data)

# Nor

如果nor中的bootdata指向的地址跟ROM里面配置的nor的地址不一致。

例如boot data指向的地址为0x20000000，则rom会从nor中将APP拷贝到0x20000000处然后执行。