HCCHIP SDK添加flash及修改开机读取flash速度说明-对外发布

1. 文档履历

版本号	日期	制/修订人	制/修订记录
1.0	2023.10.13	邱浩佳	新增说明
2.0	2024.01.25	邱浩佳	将hcbootloader/hcrtos和hclinux的文档合并在一起

2. 概述

2.1 编写目的

介绍如何在hcchip sdk上新增一款flash。

2.2 读者对象

hcchip sdk的开发工程师和FAE工程师。

3. HCBOOTLOADER\HCRTOS

hcbootloader和hcrtos共用一个flash driver，所以添加只需要加一处，但编译时需要编译bootloader和hcrtos的kernel-rebuild同时编译。

3.1 如何增加一款新flash的基础信息

下面以博雅BY25Q128AS为例，完整介绍如何添加一款新的flash。

3.1.1 第一步：查看对应的JEDEC ID

flash的识别主要依靠flash的JEDEC ID，该id记录着flash的厂家和容量等信息，可以通过flash datasheet来查找。

6. **Device Identification**

Three legacy Instructions are supported to access device identification that can indicate the manufacturer, device type, and capacity (density). The returned data bytes provide the information as shown in the below table.

Table 7. BY25Q128AS ID Definition table

Operation Code	M7-M0	ID15-ID8	ID7-ID0
9FH	68	40	18
90H/92H/94H	68		17
ABH			17

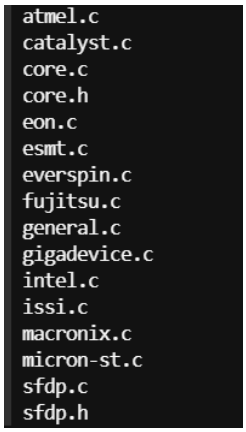
如图所示为BY25Q128AS的datasheet中描述的JEDEC ID，为：0x684018；

3.1.2 第二步：查看支持的read wire命令

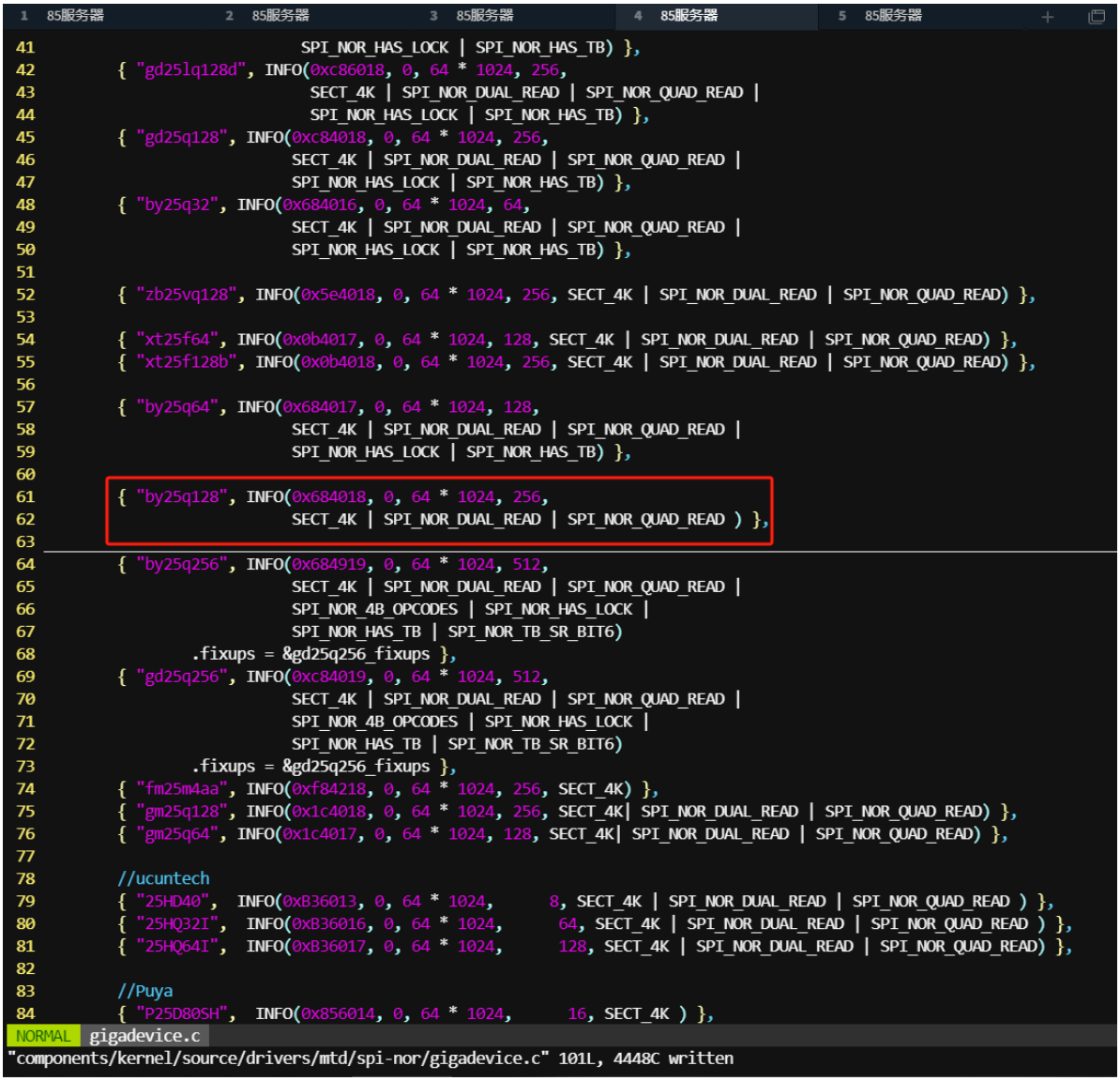
如图所示：read cmd支持single/dual/qual wire读。即属性：SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ。

3.1.3 第三步：在sdk中添加flash

在sdk的路径：components/kernel/source/drivers/mtd/spi-nor/下存放着各家厂商的flash id table。



在对应厂家中添加需要新增的flash，如果没有则可以添加在general.c中。



这里需要完成的基本属性有name：

```
{ "by25q128", INFO(0x684018, 0, 64 * 1024, 256,
                   SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ) },
```

下面是一些不同大小的基本配置，替换所使用的flash id以及是否支持二线或者四线模式，供参考：

```
// zbit
{ "25VQ16B",  INFO(0x5E4015, 0, 64 * 1024, 32, SECT_4K | SPI_NOR_QUAD_READ | SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//2M
{ "25VQ32B",  INFO(0x5E4016, 0, 64 * 1024, 64, SECT_4K | SPI_NOR_QUAD_READ | SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//4M
{ "25VQ64B",  INFO(0x5E4017, 0, 64 * 1024, 128, SECT_4K | SPI_NOR_QUAD_READ | SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//8M
{ "25VQ128B", INFO(0x5E4018, 0, 64 * 1024, 256, SECT_4K | SPI_NOR_QUAD_READ | SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//16M
{ "25VQ256B", INFO(0x5E4019, 0, 64 * 1024, 512, SECT_4K | SPI_NOR_QUAD_READ | SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//32M
```

3.1.4 第四步：编译命令

make O=output-bl/ kernel-rebuild apps-bootloader-rebuild && make kernel-rebuild all

3.2 sdk支持unknown flash 功能

该功能支持不在flash id table支持中的flash型号。需要打开宏：CONFIG_SUPPORT_UNKNOW_FLASH；开启该功能仅仅支持一线的读写。

```
There is no help available for this option.
Symbol: CONFIG_SUPPORT_UNKNOW_FLASH [=n]
Type  : bool
Prompt: support unknow flash id
  Location:
    -> Components
      -> kernel (BR2_PACKAGE_KERNEL [=y])
        -> Drivers
          -> Memory Technology Device (MTD) Support (CONFIG_MTD [=y])
            -> SPI NOR device support (CONFIG_MTD_SPI_NOR [=y])
  Defined at spi-nor:70
  Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_MTD [=y] && CONFIG_MTD_SPI_NOR [=y]
```

编译命令：make O=output-bl/ kernel-rebuild apps-bootloader-rebuild && make kernel-rebuild all。

3.2.1 通过unkonwn flash功能获取flash id

当出现找不到flash datasheet时，可以通过支持unknown flash 功能进入系统。再通过CONFIG_CMDS_SPI命令获取到flash 的id，再将其添加到对应代码位置。

```
There is no help available for this option.
Prompt: Test spi controller
  Location:
    -> Components
      -> Cmds (BR2_PACKAGE_CMDS [=y])
        -> spi operations (CONFIG_CMDS_SPI [=y])
  Defined at source:62
  Depends on: BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_SPI [=y]
  Selected by [m]:
  - BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_SPI [=y] && m
```

打开对应cmd后，执行，make cmds-rebuild all，在串口命令行执行：spi；会出现如下打印

```
jedec_id=0x684018
```

则表示该id为flash的jedec id。

3.3 flash速度的修改

打开对应的dts设备树文件，通过修改对应的属性来获取不同的速度。

```
sfspi {
        pinmux-active = <PINPAD_T15 1 PINPAD_T16 1 PINPAD_T17 1 PINPAD_T18 1>;
        sclk = <27000000>;
        status = "okay";

        spi_nor_flash {
                spi-tx-bus-width = <1>;
                spi-rx-bus-width = <1>;
                reg = <0>;
                status = "okay";
                partitions {
                        part-num = <4>;

                        /* part0 is for entire norflash by default */

                        part1-label = "boot";
                        part1-reg = <0x0 0x64000>;
                        part1-filename = "bootloader.bin";

                        part2-label = "eromfs";
                        part2-reg = <0x64000 0x20000>;
                        part2-filename = "romfs.img";

                        part3-label = "firmware";
                        part3-reg = <0x84000 0x35c000>;
                        part3-filename = "hcdemo.uImage";

                        part4-label = "persistentmem";
                        part4-reg = <0x3e0000 0x20000>;
                        part4-filename = "persistentmem.bin";
                };
        };

        spidev@0 {
                devpath = "/dev/spidev0";
                reg = <0>;
                spi-max-frequency = <50000000>;
                status = "okay";
        };
};
```

spi-rx-bus-width = <1>;：flash的读使用的wire cmd，如为4时，发送读的命令为4线。4线需要flash 型号支持和硬件支持，硬件支持需要flash的WP和HOLD引脚接到芯片对应的引脚上。

spi-tx-bus-width = <1>;：同上，但flash不存在2 wire write命令。

flash使用不同的bus-width需要flash支持表中支持对应的属性： SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ。

sclk：表示读取flash的clk频率，可支持的flash频率参见下表。

flash clk 支持表（15xx系列最高支持54M，16xx系列可配置到98M）	
9bH(27-24)	频率（MHz）
0	54
1	27
2	18
3	13.5
4	10.8
5	9
6	7.7
7	6.75
8	6
9	5.4
a	4.9
b	4.5
c	4.1
d	3.8
e	3.6
f	3.3

15xx系列支持一线和二线传输模式；16xx系列支持一线、二线和四线传输模式。传输速度四线》二线》一线；四线传输需要flash和硬件设计支持。

传输模式的选择：代码中做了最优速度处理，即会根据配置选择最优传输模式，15xx的传输模式如表所示；16xx的传输模式默认使用DMA。

3.4 flash的测试

最新master（23.10.y）分支及以后的sdk带有flash测试命令。

```
There is no help available for this option.
Symbol: CONFIG_CMDS_FLASH_TORTURETEST [=y]
Type  : bool
Prompt: flash torture test operations
  Location:
    -> Components
      -> Cmds (BR2_PACKAGE_CMDS [=y])
        -> spi operations (CONFIG_CMDS_SPI [=y])
          -> flash test cmd (CONFIG_CMDS_FLASH_TEST [=y])
  Defined at spi:7
  Depends on: BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_SPI [=y] && CONFIG_CMDS_FLASH_TEST [=y]
```

该命令会对一个分区进行随机数据的擦写读校验，并重复多次。注意使用该命令会破坏flash上原有的数据甚至会损坏flash。

```
("************************************\n");
("toruretest test cmds help\n");
("\tfor example : toruretest -d2 -c100\n");
("\t'd'    2 means mtdblock2\n");
("\t'c'    100 means test 100 times, defualt 10000\n");
("************************************\n");
```

编译命令：make cmds-rebuild all。

4. HCLINUX

hclinux sdk下添加一款flash除了在linux内核中添加，还要在hcbootloader和hcrtos下也添加，即hclinux/SOURCE/avp/components/kernel/source/drivers/mtd/spi-nor目录下，具体可以参考HCBOOTLOADER/HCRTOS

4.1 如何增加一款新flash的基础信息

下面以博雅BY25Q128AS为例，完整介绍如何添加一款新的flash。

4.1.1 第一步：查看对应的JEDEC ID

flash的识别主要依靠flash的JEDEC ID，该id记录着flash的厂家和容量等信息，可以通过flash datasheet来查找。

6. **Device Identification**

Three legacy Instructions are supported to access device identification that can indicate the manufacturer, device type, and capacity (density). The returned data bytes provide the information as shown in the below table.

Table 7. BY25Q128AS ID Definition table

Operation Code	M7-M0	ID15-ID8	ID7-ID0
9FH	68	40	18
90H/92H/94H	68		17
ABH			17

如图所示为BY25Q128AS的datasheet中描述的JEDEC ID，为：0x684018；

4.1.2 第二步：查看支持的read wire命令

如图所示：read cmd支持single/dual/qual wire读。即属性：SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ。

4.1.3 第三步：在sdk中添加flash

4.1.3.1 Linux4.14内核

在sdk的路径：hclinux/buildroot/output/d3100_v20/build/linux-4.4.186/drivers/mtd/spi-nor/spi-nor.c中存放着不同厂家支持的flash id。

```
666 static const struct flash_info spi_nor_ids[] = {
667         /* Atmel -- some are (confusingly) marketed as "DataFlash" */
668         { "at25fs010",  INFO(0x1f6601, 0, 32 * 1024,   4, SECT_4K) },
669         { "at25fs040",  INFO(0x1f6604, 0, 64 * 1024,   8, SECT_4K) },
670
671         { "at25df041a", INFO(0x1f4401, 0, 64 * 1024,   8, SECT_4K) },
672         { "at25df321a", INFO(0x1f4701, 0, 64 * 1024,  64, SECT_4K) },
673         { "at25df641",  INFO(0x1f4800, 0, 64 * 1024, 128, SECT_4K) },
674
675         { "at26f004",   INFO(0x1f0400, 0, 64 * 1024,  8, SECT_4K) },
676         { "at26df081a", INFO(0x1f4501, 0, 64 * 1024, 16, SECT_4K) },
677         { "at26df161a", INFO(0x1f4601, 0, 64 * 1024, 32, SECT_4K) },
678         { "at26df321",  INFO(0x1f4700, 0, 64 * 1024, 64, SECT_4K) },
679
680         { "at45db081d", INFO(0x1f2500, 0, 64 * 1024, 16, SECT_4K) },
681
682         /* EON -- en25xxx */
683         { "en25f32",    INFO(0x1c3116, 0, 64 * 1024,   64, SECT_4K) },
684         { "en25p32",    INFO(0x1c2016, 0, 64 * 1024,   64, 0) },
685         { "en25q32b",   INFO(0x1c3016, 0, 64 * 1024,   64, 0) },
686         { "en25p64",    INFO(0x1c2017, 0, 64 * 1024,  128, 0) },
687         { "en25q64",    INFO(0x1c3017, 0, 64 * 1024,  128, SECT_4K) },
688         { "en25qh128",  INFO(0x1c7018, 0, 64 * 1024,  256, 0) },
689         { "en25qh256",  INFO(0x1c7019, 0, 64 * 1024,  512, 0) },
690         { "en25s64",    INFO(0x1c3817, 0, 64 * 1024,  128, SECT_4K) },
691         { "en25qx256",  INFO(0x1c7119, 0, 64 * 1024,  512, SECT_4K) },
692
NORMAL  spi-nor.c
```

将对应的flash属性添加进spi_nor_ids中。

这里需要完成的基本属性有name：

```
1 { "by25q128", INFO(0x684018, 0, 64 * 1024, 256,
2                     SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ) },
```

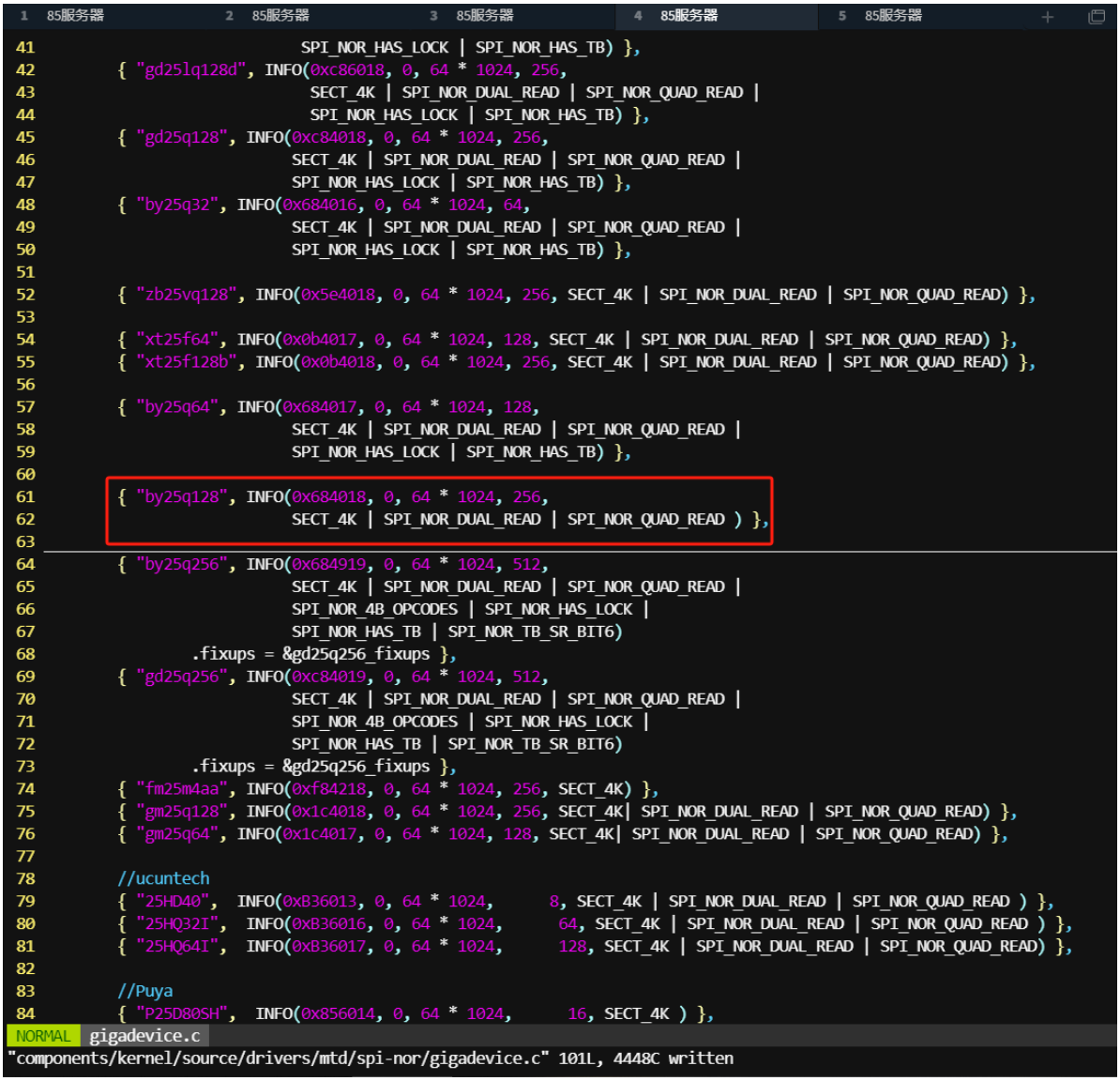
```
        /* GigaDevice */
        { "gd25q32", INFO(0xc84016, 0, 64 * 1024,  64, SECT_4K) },
        { "gd25q64", INFO(0xc84017, 0, 64 * 1024, 128, SECT_4K) },
        { "gd25q128", INFO(0xc84018, 0, 64 * 1024, 256, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ) },
        { "by25q128", INFO(0x684018, 0, 64 * 1024, 256, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ) },
        { "by25q256", INFO(0x684919, 0, 64 * 1024, 512, SECT_4K) },
        { "xt25p1288", INFO(0x0b4018, 0, 64 * 1024, 256, SECT_4K) },
        { "zb25vq128", INFO(0x5e4018, 0, 64 * 1024, 256, SECT_4K) },
        { "zb25vq16B",  INFO(0x5E4015, 0, 64 * 1024,  32, SECT_4K ) },
        { "fm25m4aa", INFO(0xf84218, 0, 64 * 1024, 256,SECT_4K) },
        { "gm25q128", INFO(0x1c4018, 0, 64 * 1024, 256, SECT_4K) },
        { "gm25q64", INFO(0x1c4017, 0, 64 * 1024, 128, SECT_4K) },

```

```
/* zbit */
{ "25VQ16B",  INFO(0x5E4015, 0, 64 * 1024,   32, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ ) },//2M
{ "25VQ32B",  INFO(0x5E4016, 0, 64 * 1024,   64, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ ) },//4M
{ "25VQ64B",  INFO(0x5E4017, 0, 64 * 1024,  128, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ ) },//8M
{ "25VQ128B", INFO(0x5E4018, 0, 64 * 1024,  256, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ ) },//16M
{ "25VQ256B", INFO(0x5E4019, 0, 64 * 1024,  512, SECT_4K | SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ ) },//32M
```

4.1.3.2 Linux5.12内核

在sdk的路径：hclinux/buildroot/output/build/linux-5.12.4/drivers/mtd/spi-nor下存放着各家厂商的flash id table。

在对应厂家中添加需要新增的flash，如果没有则可以添加在general.c中。



这里需要完成的基本属性有：

```
{ "by25q128", INFO(0x684018, 0, 64 * 1024, 256, SECT_4K | SPI_NOR_DUAL_READ |
SPI_NOR_QUAD_READ | SPI_NOR_SKIP_SFDP) },
```

下面是一些不同大小的基本配置，替换所使用的flash id以及是否支持二线或者四线模式，供参考：

```
// zbit
{ "25VQ16B",  INFO(0x5E4015, 0, 64 * 1024,  32, SECT_4K | SPI_NOR_QUAD_READ |
SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//2M
{ "25VQ32B",  INFO(0x5E4016, 0, 64 * 1024,  64, SECT_4K | SPI_NOR_QUAD_READ |
SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//4M
{ "25VQ64B",  INFO(0x5E4017, 0, 64 * 1024, 128, SECT_4K | SPI_NOR_QUAD_READ |
SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//8M
{ "25VQ128B", INFO(0x5E4018, 0, 64 * 1024, 256, SECT_4K | SPI_NOR_QUAD_READ |
SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//16M
{ "25VQ256B", INFO(0x5E4019, 0, 64 * 1024, 512, SECT_4K | SPI_NOR_QUAD_READ |
SPI_NOR_DUAL_READ | SPI_NOR_SKIP_SFDP) },//32M
```

SPI_NOR_SKIP_SFDP建议加上，虽然一些flash的datasheet表示有支持SFDP表，但实际使用下来发现，读出来的可能是错误值，加上该flag会跳过SFDP的读取。

4.1.4 第四步：编译命令

make hcboot-kernel-rebuild hcboot-rebuild hcboot-all avp-kernel-rebuild avp-rebuild avp-all linux-rebuild all。

4.2 sdk支持unknown flash 功能

该功能支持不在flash id table支持中的flash型号。需要打开宏：CONFIG_SUPPORT_UNKNOW_FLASH；开启该功能flash仅仅支持一线的读写。

```
There is no help available for this option.
Symbol: SUPPORT_UNKNOW_FLASH [=n]
Type  : bool
Defined at drivers/mtd/spi-nor/Kconfig:27
  Prompt: support unknow flash id
  Depends on: MTD [=y] && MTD_SPI_NOR [=y]
  Location:
    -> Device Drivers
      -> Memory Technology Device (MTD) support (MTD [=y])
        -> SPI NOR device support (MTD_SPI_NOR [=y])
```

编译命令：make linux-rebuild all。

4.2.1 通过unkonwn flash功能获取flash id

当出现找不到flash datasheet时，可以通过支持unknown flash 功能进入系统。再通过CONFIG_CMDS_SPI命令获取到flash 的id，再将其添加到对应代码位置。

```
There is no help available for this option.
Prompt: Test spi controller
  Location:
    -> Components
      -> Cmds (BR2_PACKAGE_CMDS [=y])
        -> spi operations (CONFIG_CMDS_SPI [=y])
  Defined at source:62
  Depends on: BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_SPI [=y]
  Selected by [m]:
  - BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_SPI [=y] && m
```

开启support unknown flash id后，系统起来后，会出现如下打印：

```
usbcore: registered new interface driver usb-storage
i2c /dev entries driver
IR NEC protocol handler initialized
Synopsys Designware Multimedia Card Interface Driver
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
virtuart init ok
kshmdev init ok
WARNNING!!! This flash id:0x544018 is no in flash id table,trying use general id !
m25p80 spi32766.0: found Unknow16M, expected m25p80
m25p80 spi32766.0: Unknow16M (16384 Kbytes)
8 ofpart partitions found on MTD device spi32766.0
Creating 8 MTD partitions on "spi32766.0":
0x000000000000-0x000001000000 : "nor"
0x000000000000-0x000000080000 : "boot"
0x000000080000-0x000000090000 : "dtb"
0x000000090000-0x000000230000 : "avp"
0x000000230000-0x000000530000 : "linux"
0x000000530000-0x000000fb0000 : "rootfs"
0x000000fb0000-0x000000fe0000 : "eromfs"
0x000000fe0000-0x000001000000 : "persistentmem"
fb node name: fb0!
fb0 enable!
```

会将该flash的id打印出来，并使用一个通用flash id来使系统跑起来。

4.3 flash速度的修改

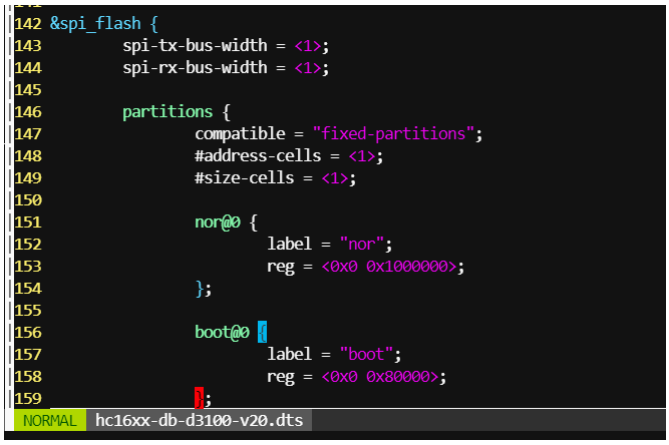
打开对应的board/hichip/hc16xx/common/dts/hc16xx-common.dtsi设备树文件，通过修改对应的属性来获取不同的速度。

```
spi0: spi@1882e000 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "hichip,hc16xx-spi-sf";
        reg = <0x1882e000 0xe0>, <0x1f000000 0x1000>, <0x1f000000 0x1000>;
        num-cs = <2>;
        interrupts = <35>;
        sclk = <50000000>;
        dma-mode = <1>;
        pinctrl-names = "active";
        pinctrl-0 = <&pctl_sf>;

        spi_flash: spi_flash@0 {
                #address-cells = <1>;
                #size-cells = <1>;
                compatible = "m25p80";
                reg = <0>;
                spi-max-frequency = <50000000>;
        };

        spidev@0 {
                compatible = "rohm,dh2228fv";
                reg = <1>;
                spi-max-frequency = <50000000>;
        };
};
```

sclk：表示读取flash的clk频率，可支持的flash频率参见下表。

flash clk 支持表（16xx系列最高支持98M，设备树直接设置即可）	
9bH(27-24)	频率（MHz）（主频 108MHz）
0	54
1	27
2	18
3	13.5
4	10.8
5	9
6	7.7
7	6.75
8	6
9	5.4
a	4.9
b	4.5
c	4.1
d	3.8
e	3.6
f	3.3

打开对应的board/hichip/hc16xx/common/dts/hc16xx-db-d3100-v20.dts设备树文件，通过修改对应的属性来获取不同的速度。



spi-rx-bus-width = <1>;：flash的读使用的wire cmd，如为4时，发送读的命令为4线。4线需要flash 型号支持和硬件支持。

spi-tx-bus-width = <1>;：同上，但flash不存在2 wire write命令。

flash使用不同的bus-width需要flash支持表中支持对应的属性： SPI_NOR_DUAL_READ | SPI_NOR_QUAD_READ。

15xx系列：27M以下均一线或者两线，在pio+dma或者单独pio模式，都可以正常启动。

传输模式的选择：代码中做了最优速度处理，即会根据配置选择最优传输模式，15xx的传输模式如表所示；16xx的传输模式默认使用DMA。

4.4 flash的测试

通过使用Linux自带mtd test命令，可以实现对flash的测试。通过宏MTD_TESTS打开。

```
CONFIG_MTD_TESTS:

This option includes various MTD tests into compilation. The tests
should normally be compiled as kernel modules. The modules perform
various checks and verifications when loaded.

WARNING: some of the tests will ERASE entire MTD device which they
test. Do not use these tests unless you really know what you do.

Symbol: MTD_TESTS [=n]
Type  : tristate
Prompt: MTD tests support (DANGEROUS)
  Location:
    -> Device Drivers
      -> Memory Technology Device (MTD) support (MTD [=y])
  Defined at drivers/mtd/Kconfig:15
  Depends on: MTD [=y] && m && MODULES [=y]
```

该测试驱动只能编译成模块，编译后位置在：

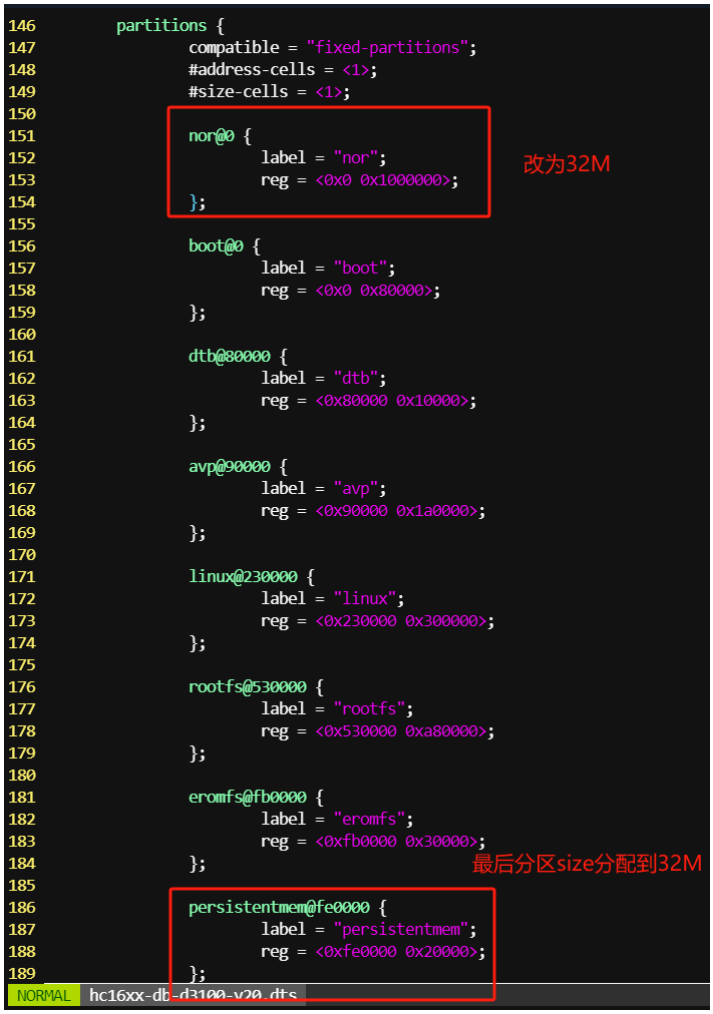
```
~/linux/07.y.hclinux/hclinux/buildroot/output/a3300_v10/build/linux-4.4.186/drivers/mtd/tests (master) $ ls *.ko
mtd_nandbiterrs.ko  mtd_nandecctest.ko  mtd_oobtest.ko  mtd_pagetest.ko  mtd_readtest.ko  mtd_speedtest.ko  mtd_stresstest.ko  mtd_subpagetest.ko  mtd_torturetest.ko
~/linux/07.y.hclinux/hclinux/buildroot/output/a3300_v10/build/linux-4.4.186/drivers/mtd/tests (master) $
```

当flash是spi nor时，仅支持部分命令，命令的使用参考：使用linux的MTD tests support测试flash性能_mtdtest-CSDN博客。

4.5 常见问题

Q：为什么使用的flash是32M byte，但在Linux内核中只有16M byte?

A：请检测dts设备树文件中flash的分区配置是否为32M byte。


partitions {
compatible = "fixed-partitions";
#address-cells = <1>;
#size-cells = <1>;
nor@0 {
label = "nor";
reg = <0x0 0x1000000>;
};
boot@0 {
label = "boot";
reg = <0x0 0x80000>;
};
dtb@80000 {
label = "dtb";
reg = <0x80000 0x10000>;
};
avp@90000 {
label = "avp";
reg = <0x90000 0x1a0000>;
};
linux@230000 {
label = "linux";
reg = <0x230000 0x300000>;
};
rootfs@530000 {
label = "rootfs";
reg = <0x530000 0xa80000>;
};
eromfs@fb0000 {
label = "eromfs";
reg = <0xfb0000 0x30000>;
};
persistentmem@fe0000 {
label = "persistentmem";
reg = <0xfe0000 0x20000>;
};
改为32M
最后分区size分配到32M
NORMAL