Ingenic®

SPI NAND

ADD Parameter Description Document

Date: Feb 2023

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

Trademarks and Permissions

Disclaimer

All the deliverables and data in this folder serve only as a reference for customer development. Please read through this disclaimer carefully before you use the deliverables and data in this folder. You may use the deliverables in this folder or not. However, by using the deliverables and data in this folder, you agree to accept all the content in this disclaimer unconditional and irrevocable. If you do not find the content in this disclaimer reasonable, you shall not use the deliverables and data in this folder.

The deliverables and data in this folder are provided "AS IS" without representations, guarantees or warranties of any kind (either express or implied). To the maximum extent permitted by law, Ingenic Semiconductor Co., Ltd (Ingenic) provides the deliverables and data in this folder without implied representations, guarantees or warranties, including but not limited to implied representations, guarantees and warranties of merchantability, non-infringement, or fitness for a particular purpose. Deviation of the data provided in this folder may exist under different test environments.

Ingenic takes no liability or legal responsibility for any design and development error, incident, negligence, infringement, and loss (including but not limited to any direct, indirect, consequential, or incidental loss) caused by the use of data in this folder. Users shall be responsible for all risks and consequences caused by the use of data in this folder.

Ingenic Semiconductor Co., Ltd.
Add: Ingenic Headquarters, East Bldg. 14, Courtyard #10, Xibeiwang East Road, Haidian Dist., Beijing, China
Tel: **(86-10)56345000**
Fax: **(86-10)56345001**
http: //www.ingenic.cn

Directory

Release history

Date	Revision	Revision History
Feb.01,2023	2.0	Second released
Mar.21,2018	1.0	First released

Overview

To support new devices need to add SFC NAND parameters to SPL, UBOOT, and KERNEL.

Codes	Parameter Directory
SPL	u-boot/tools/ingenic-tools/nand_device/
UBOOT	1) X1000 series NAND parameter code path: u-boot/drivers/mtd/devices/jz_sfc/nand_device/ 2) X2000 series NAND parameter code path: u-boot/drivers/mtd/devices/jz_sfc_v2/nand_device/
KERNEL	1) X1000 series NAND parameter code path: kernel/drivers/mtd/devices/ingenic_sfc_v1/nand_device/ 1) X2000 series NAND parameter code path: kernel/module_drivers/drivers/mtd/devices/ingenic_sfc_v2/nand_device/

1. Add New Device to SPL

步骤	描述
1	If there is no corresponding vendor file in the parameter Directory, copy the supported vendor. c file and rename it as the new vendor name. For example: 1) cd cp tools/ingenic-tools/nand_device/ 2) cp foresee_nand.c new_nand.c 3) Replace the FS/fs/foresee vendor name field in the new_nand.c file.
2	Refer to the FLASH manual to modify parameters.

Use u-boot/tools/ingenic-tools/nand_device/foresee_nand.c as an example to explain parameters:

```
#include <stdio.h>
    #include "nand_common.h"

    #define FS_MID                      0xCD
```

➤ Explanation:

The manufacturer ID is 0xCD and the device ID is 0xA1.

```
#define FS_NAND_DEVICD_COUNT            1
```

➤ Explanation:

supported device ID counts

```
static unsigned char fs_xaw[] = {0x4, 0x7};
```

➤ Explanation:

ECC error status value definition

```
static struct device_struct device[] = {
        DEVICE_STRUCT(0xA1, 2048, 2, 4, 3, 2, fs_xaw),
```

➤ Explanation:

```
#define DEVICE_STRUCT(id, pagesize, addrlen, bit, bitcounts,eccstatcount, err) {\
    .device_id = id,                    \   FLASH device ID
    .page_size = pagesize,              \   FLASH page size
    .addr_len = addrlen,                \   single-line or four-line read address length
    .ecc_bit = bit,                     \   the lowest bit of ECC in the status register
    .bit_counts = bitcounts,            \   the number of ECC bits in the status register
    .eccstat_count = eccstatcount, \        number of ECC error status values
    .eccerrstatus = err,                \   ECC error status value memory address
}
```

```
};

static struct nand_desc fs_nand = {
        .id_manufactory = FS_MID,
        .device_counts = FS_NAND_DEVICD_COUNT,
        .device = device,
};

int foresee_nand_register_func(void) {
        return nand_register(&fs_nand);
}
```

2. Add New Device to UBOOT

Step	Description
1	If there is no corresponding vendor file in the parameter Directory, copy the supported vendor. c file and rename it as the new vendor name. Modify the Makefile in the parameter directory to add the new vendor name. o file. For example:

	1) cd u-boot/drivers/mtd/devices/jz_sfc_v2/nand_device/ 2) cp foresee_nand.c new_nand.c 3) Replace the FS/fs/foresee vendor name field in the new_nand.c file. 4) vi ../Makefile 5) Add COBJS-$(CONFIG_MTD_SFCNAND) +=nand_device/new_nand.o
2	Refer to the FLASH manual to modify parameters.

Use u-boot/drivers/mtd/devices/jz_sfc_v2/nand_device/foresee_nand.c as an example to explain parameters:

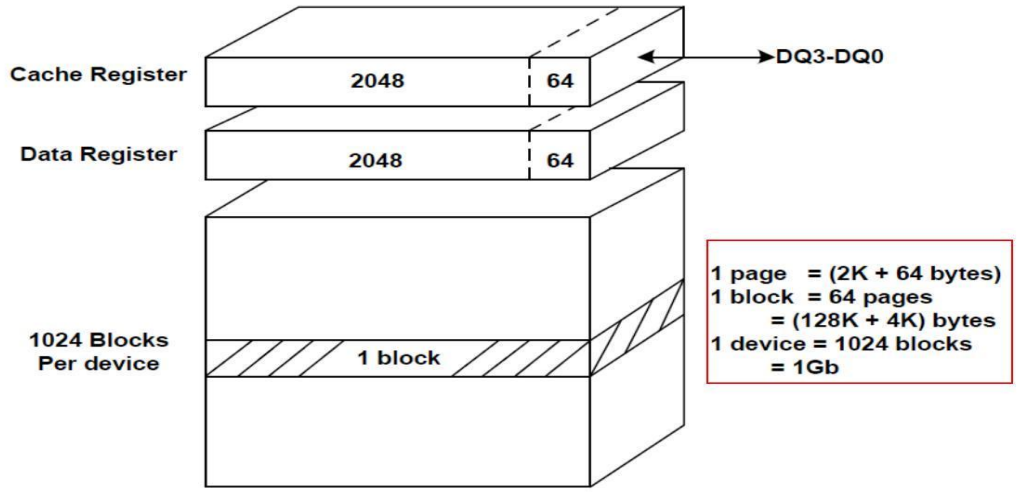
```
#include <errno.h>
    #include <malloc.h>
    #include <linux/mtd/partitions.h>
    #include "../jz_sfc_common.h"
    #include "nand_common.h"

    #define FS_DEVICES_NUM          1
```

➢ Explanation:

supported device ID count

```
    #define TSETUP          5
    #define THOLD       5
    #define TSHSL_R         20
    #define TSHSL_W         20

    #define TRD         180
    #define TPP     400
    #define TBE         3

    static struct jz_sfcnand_device *fs_nand;

    static struct jz_sfcnand_base_param fs_param[FS_DEVICES_NUM] = {

        [0] = {
            /*FS35ND01G*/
            .pagesize = 2 * 1024,
            .blocksize = 2 * 1024 * 64,
            .oobsize = 64,
            .flashsize = 2 * 1024 * 64 * 1024,
```

➢ Explanation:



pagesize	device page size without oob size
blocksize	device block size without oob size
oobsize	oob area size in each page
flashsize	chip size with oob size

.tSETUP = TSETUP,
.tHOLD = THOLD,
.tSHSL_R = TSHSL_R,
.tSHSL_W = TSHSL_W,

➤ Explanation:

片选时序参数:

tSLCH	CS# Active Setup Time	5			ns
tCHSH	CS# Active Hold Time	5			ns
tSHCH	CS# Not Active Setup Time	5			ns
tCHSL	CS# Not Active Hold Time	5			ns
tSHSL/tCS	CS# High Time	20			ns

tSETUP(tSLCH): CS# Active Setup Time
tHOLD (tCHSH): CS# Active Hold Time
tSHSL_R: CS #Deselect Time (for Array Read -> Array Read)
tSHSL_W: CS #Deselect Time (for Erase, Program or Read Status Registers ->Read Status Registers)

.tRD = TRD,
.tPP = TPP,
.tBE = TBE,

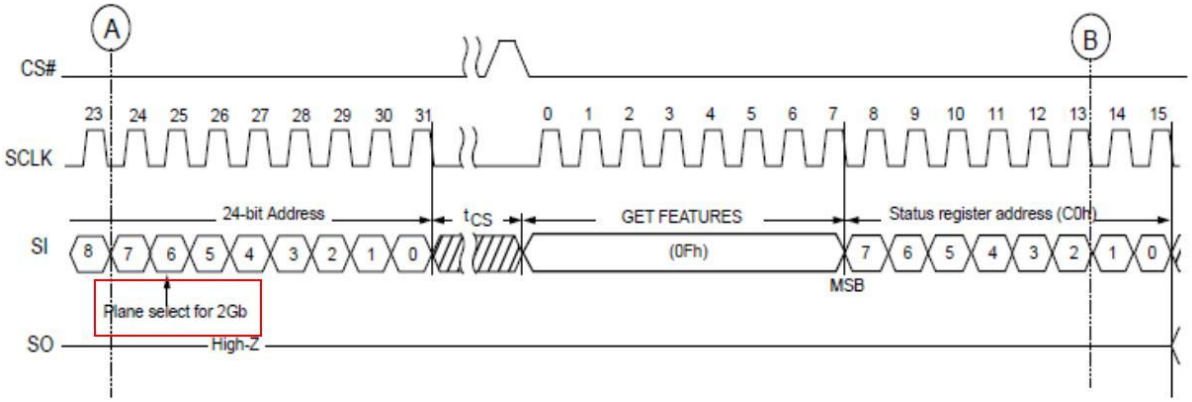
➤ Explanation:

SYMBOL	PARAMETER	SPEC			UNIT
		MIN	TYP	MAX	
tRST	CS# High to Next Command After Reset(FFh)			500	μs
tRD	Page Read From Array		120	450	μs
tPROG	Page Program		430	800	μs
tERS	Block Erase		2	10	ms
NOP	Number of partial page program			1	time

tRD	Maximum read time for a page containing OOB data
tPP	Maximum Write time for a page containing OOB data
tBE	Maximum Write time for a page containing OOB data

.plane_select = 0,

➤ Explanation:



plane_select = 1 when the read/write time series includes the chip selection signal, otherwise 0

Note: The X1000 platform does not have this parameter. This operation is not supported.

.ecc_max = 0x4,

➤ Explanation:

ECCS2 ECCS1 ECCS0	ECC Status	ECCS provides ECC Status as follows: 000b = No bit errors were detected during the previous read algorithm. 001b = Bit errors(=1) were detected and corrected. 010b = Bit errors(=2) were detected and corrected. 011b = Bit errors(=3) were detected and corrected. 100b = Bit errors(=4) were detected and corrected. 111b = Internal error was detected and the data not promised correctly. Others = Reserved. Bit errors might not be detected and corrected if their number exceedsthe tolerance. Therefore, block data should be refreshed when ECCStatus is equal to 100b. ECCS is set to 000b either following a RESET, or at the beginning ofthe READ. It is then updated after the device completes a valid READoperation. ECCS is invalid if internal ECC is disabled (via a SET FEATUREScommand to reset ECC_EN).

The maximum number of Bits that can be corrected in the ECC status

Note: some FLASH does not have internal ECC, which requires external controller support. We do not support it.

```
            .need_quad = 1,
```

➤ Explanation:

Set 1 when the four-wire function is activated, otherwise 0

```
        },

    };

    static struct device_id_struct device_id[FS_DEVICES_NUM] = {
        DEVICE_ID_STRUCT(0xA1, "FS35ND01G", &fs_param[0]),
```

➤ Explanation:

```
#define DEVICE_ID_STRUCT(id, name_string, parameter)        {\
    .id_device = id,                \      FLASH Device ID
    .name = name_string,            \       FLASH Device Name
    .param = parameter,             \}      Parameters Memory Address
    };
```

```
    static cdt_params_t *fs_get_cdt_params(struct sfc_flash *flash, uint8_t device_id)
    {
        CDT_PARAMS_INIT(fs_nand->cdt_params);

        switch(device_id) {
            case 0xA1:
                break;
```

➤ Explanation:

PAGE READ	**13h**	A23-A16	A15-A8	A7-A0		
READ FROM CACHE	**03h/0Bh**	A15-A8(2)	A7-A0	dummy	(D7-D0)	wrap
READ ID	**9Fh**	dummy	(MID) (8)	(DID) (8)		wrap
READ UID	**4Bh**	dummy	dummy	dummy	dummy	(D7-D0)
PROGRAM LOAD	**02h**	A15-A8(6)	A7-A0	D7-D0	Next byte	Byte N

Single-line and four-line read commands break when the time series address length is 2, and re-assign the number of address bytes when the time series address length is 3, as fol lows:

```
xx_nand->cdt_params.standard_r.addr_nbyte = 3;
xx_nand->cdt_params.quad_r.addr_nbyte = 3;
```

```
            default:
                pr_err("device_id err, please check your device id: device_id = 0x%02x\n", device_id);
                return NULL;
        }

        return &fs_nand->cdt_params;
```

```
}

static inline int deal_ecc_status(struct sfc_flash *flash, uint8_t device_id, uint8_t ecc_status)
{
    int ret = 0;

    switch(device_id) {
        case 0xA1:
            switch((ret = ((ecc_status >> 4) & 0x7))) {
                case 0x0 ... 0x4:
                    return 0x4;
                default:
                    return -EBADMSG;
            }
```

➤ Explanation:

case 0x0 ... 0x4:
When the ECC status value is between 0 and 4, the maximum recoverable ECC value is returned. When the UBI file system finds a bit flip error, the data is migrated to another location.

default:
Return -EBADMSG when other values are returned. The number of data errors on this page exceeds 4 and cannot be corrected.

```
            default:
                printf("device_id err, it maybe don`t support this device, check your device id: device_id = 0x%02x\n", device_id);
                ret = -EIO;     //notice!!!

        }
        return ret;

}

static int fs_nand_init(void) {

    fs_nand = kzalloc(sizeof(*fs_nand), GFP_KERNEL);
    if(!fs_nand) {
        pr_err("alloc fs_nand struct fail\n");
        return -ENOMEM;
    }

    fs_nand->id_manufactory = 0xCD;
```

➢ Explanation:
MID is Manufacture ID (CDh for FSNAND), DID is Device ID (A1h for current device)
The manufacturer ID is 0xCD and the device ID is 0xA1.

```
        fs_nand->id_device_list = device_id;
        fs_nand->id_device_count = FS_DEVICES_NUM;

        fs_nand->ops.get_cdt_params = fs_get_cdt_params;
```

➢ Explanation:
If the length of the single-line and four-line read time series address is less than 2, the value is re-assigned in the xx_get_cdt_params function in this file.

```
        fs_nand->ops.deal_ecc_status = deal_ecc_status;
```

➢ Explanation:
ECC status processing function address

```
        /* use private get feature interface, please define it in this document */
        fs_nand->ops.get_feature = NULL;

        return jz_sfcnand_register(fs_nand);
    }

SPINAND_MOUDLE_INIT(fs_nand_init);
```

Check whether the page size, OOB size, and number of blocks defined in the Uboot board-level header file are the same as those defined in the manual.

```
#define CONFIG_SPI_NAND_BPP                    (2048 +64)
#define CONFIG_SPI_NAND_PPB                    (64)
```

3. Add Device to KERNEL

Step	Description
1	If there is no corresponding vendor file in the parameter Directory, copy the new_nand.c added in UBOOT to the KERNEL parameter directory. For example: 1) cd module_drivers/drivers/mtd/devices/ingenic_sfc_v2/nand_device/ 2) cp ~/SDK/u-boot/drivers/mtd/devices/jz_sfc_v2/nand_devcie/new_nand.c . 3) vi Makefile 4) Add new_nand.o after obj-y += dosilicon_nand.o foresee_nand.o...
2	Modify new_nand.c For example: meld foresee_nand.c new_nand.c

4. ROOTFS Parameter configuration

When the page size is 2 KB, buildroot menuconfig configuration:
[*] ubifs root filesystem
(0x1f000) logical eraseblock size
(0x800) minimum I/O unit size

When the page size is 4 KB, buildroot menuconfig configuration:
[*] ubifs root filesystem
(0x3e000) logical eraseblock size
(0x1000) minimum I/O unit size