Accessing SPI NOR flash registers in U-Boot console

About this document

Scope and purpose

This application note describes how to access configuration registers in Infineon SPI NOR flash devices in U-Boot console. It introduces source code snippets and usage examples of a simple SPI NOR flash access commands based on the U-Boot command line interface, sf.

Intended audience

This is intended for users who use Infineon SPI NOR flash devices in U-Boot console. It is assumed that users have knowledge and experience of building and customizing U-Boot.

Table of contents

About this document........................................................................................................................................ 1
Table of contents........................................................................................................................................... 1
1 Introduction ............................................................................................................................................. 2
2 U-Boot configuration ..................................................................................................................... 3
  2.1 U-Boot version........................................................................................................................................ 3
  2.2 Enable SPI Flash Support.......................................................................................................................... 3
  2.3 Enable sf .................................................................................................................................................. 4
3 Source code modification ............................................................................................................................ 5
  3.1 Add global scope functions for register access in MTD driver ................................................................. 5
  3.2 Add register access commands in sf........................................................................................................ 6
  3.3 All changes in unified diff format............................................................................................................ 8
4 Usage examples of sf with Infineon S25HS512T .....................................................................................11
5 Conclusion ...............................................................................................................................................12
Revision history.............................................................................................................................................13
1 Introduction

Infineon S25HL-T, S25HS-T, S25FL-L, and S25FS-S SPI NOR flash devices have separate non-volatile and volatile registers. During powerup, hardware reset, or software reset, the contents in the non-volatile registers are automatically loaded to the counterpart volatile registers. Non-volatile registers are used to apply default settings before system boot, while volatile registers are used to change settings at system runtime. This is because the non-volatile registers are based on flash memory cells which have limited update cycles, take a longer time to update as compared to volatile registers, and are intolerant to power interruption during update.

In general, non-volatile registers can be updated by flash programmers equipped in production facilities. On the other hand, engineers who develop and evaluate the systems may need a way to update non-volatile registers in their lab, especially a way of the in-system programming.

In U-Boot console, Memory Technology Device (MTD) drivers and a command line tool (sf) provide access to the flash memory array, but not to flash registers. This application note introduces a simple way to access flash registers, with small modification in MTD driver and sf. It describes how to enable SPI NOR flash support in U-Boot configuration and inspect the source code of MTD driver and sf.

See U-Boot documentation for the basics and usage of sf; See the corresponding device datasheets for information on SPI NOR flash registers.
Accessing SPI NOR flash registers in U-Boot console

U-Boot configuration

2  U-Boot configuration

2.1  U-Boot version

Non-volatile and volatile registers can be accessed by specific commands named Read Any Register and Write Any Register, followed by address mapped to each register. Read Any Register and Write Any Register are supported in **U-Boot v2021.10** or later, along with S25HL-T and S25HS-T support.

2.2  Enable SPI Flash Support

In menuconfig, enable the following options under Device Drivers > MTD Support > SPI Flash Support.

- SPI Flash Core Interface support (CONFIG_SPI_FLASH)
- SFDP table parsing support for SPI NOR flashes (CONFIG_SPI_FLASH_SFDP_SUPPORT)
- Spansion SPI flash support (CONFIG_SPI_FLASH_SUPPORT)

The settings for options other than above are depending on the platform and use case.

*Note:* SPI flash Bank/Extended address register support (CONFIG_SPI_FLASH_BAR) should be disabled to activate the SFDP option.

![Figure 1] SPI Flash Support configuration
U-Boot configuration

2.3 Enable sf

In menuconfig, enable sf option under Command line interface > Device access commands.

![Device access commands configuration](image)

Figure 2 Device access commands configuration
3 Source code modification

3.1 Add global scope functions for register access in MTD driver

In drivers/mtd/spi/spi-nor-core.c, there are file scope functions, spansion_read_any_reg() and spansion_write_any_reg(), which perform register access operations. Add global scope functions that wrap the local functions so that the command line tool can use them. Code Listing 1 and Code Listing 2 show the definitions and declarations of new functions. It is recommended that the code is added at the bottom of the source file.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Functions added in drivers/mtd/spi/spi-nor-core.c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>spi_nor_read_any_reg()</td>
<td>Wrapper for spansion_read_any_reg(), passing register address, number of dummy cycles, and buffer for register value.</td>
</tr>
<tr>
<td>spi_nor_write_any_reg()</td>
<td>Wrapper for spansion_write_any_reg(), passing register address and new register value. This function also calls write_enable() before spansion_write_any_reg().</td>
</tr>
</tbody>
</table>

Code Listing 1 drivers/mtd/spi/spi-nor-core.c

```
001   #ifdef CONFIG_SPI_FLASH_SPANSION
002   int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr,
003       u8 dummy, u8 *val)
004   {
005       return spansion_read_any_reg(nor, addr, dummy, val);
006   }
007   int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr,
008       u8 val)
009   {
010       int ret;
011       ret = write_enable(nor);
012       if (ret)
013           return ret;
014       return spansion_write_any_reg(nor, addr, val);
015   }
016   #endif
```

Code Listing 2 include/linux/mtd/spi-nor.h

```
001   #ifdef CONFIG_SPI_FLASH_SPANSION
002   int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr,
003       u8 dummy, u8 *val);
004   int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr,
005       u8 val);
006   #endif
```
### 3.2 Add register access commands in sf

The `cmd/sf.c` defines SPI flash access commands such as read, write, and erase that can be used from U-Boot console. Add new commands that provide register access. **Code Listing 3** shows the newly added functions. To call these functions, additional else-if blocks should be inserted in existing `do_spi_flash()` function (Code Listing 4).

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Functions added in cmd/sf.c</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><code>do_spi_rdar()</code></td>
<td>Performs register read in response to “rdar” command. Parses command line arguments as register address and number of dummy cycles. Prints the register value to the console.</td>
</tr>
<tr>
<td><code>do_spi_wrar()</code></td>
<td>Performs register write in response to “wrar” command. Parses command line arguments as register address and new register value to write.</td>
</tr>
</tbody>
</table>

**Code Listing 3** New functions in cmd/sf.c

```c
static int do_spi_rdar(int argc, char * const argv[]) {
    int ret = 0;
    loff_t ofs;
    ulong dummy;
    u8 val;
    if (argc != 3) {
        return -1;
    }
    if (!str2off(argv[1], &ofs)) {
        puts("register address is not a valid number
        return 1;
    }
    if (!str2long(argv[2], &dummy)) {
        puts("register address is not a valid number
        return 1;
    }
    ret = spi_nor_read_any_reg(flash, ofs, (u8)dummy, &val);
    if (ret == 0)
        printf("%02X
        return ret == 0 ? 0 : 1;
    }

static int do_spi_wrar(int argc, char * const argv[]) {
    int ret = 0;
    loff_t ofs;
    ulong val;
```
Accessing SPI NOR flash registers in U-Boot console

Source code modification

Code Listing 3   New functions in cmd/sf.c

```c
036 if (argc != 3)
037     return -1;
039
040 if (!str2off(argv[1], &ofs)) {
041     puts("register address is not a valid number\n");
042     return 1;
043 }
044
045 if (!str2long(argv[2], &val)) {
046     puts("val is not a valid number\n");
047     return 1;
048 }
049
050 ret = spi_nor_write_any_reg(flash, ofs, (u8)val);
051
052 return ret == 0 ? 0 : 1;
053 }
```

Code Listing 4   Additional else-if blocks inserted in do_spi_flash()

```c
001 else if (strcmp(cmd, "rdar") == 0)
002     ret = do_spi_rdar(argc, argv);
003 else if (strcmp(cmd, "wrar") == 0)
004     ret = do_spi_wrar(argc, argv);
```
Accessing SPI NOR flash registers in U-Boot console

3.3 All changes in unified diff format

Code Listing 5 shows all changes in unified diff format. Copy and paste it to a text file then apply these changes by 'patch' command.

Code Listing 5

```c
001  diff --git a/cmd/sf.c b/cmd/sf.c
002  index eac27ed2d7..1ffdaedae5 100644
003  --- a/cmd/sf.c
004  +++ b/cmd/sf.c
005  @@ -384,6 +384,60 @@ static int do_spi_protect(int argc, char
006         *const argv[])
007       return ret == 0 ? 0 : 1;
008
009 +static int do_spi_rdar(int argc, char * const argv[])
010 +{
011 +    int ret = 0;
012 +    loff_t ofs;
013 +    ulong dummy;
014 +    u8 val;
015 +    if (argc != 3)
016 +        return -1;
017 +    if (!str2off(argv[1], &ofs)) {
018 +        puts("register address is not a valid number\n");
019 +        return 1;
020 +    }
021 +    ret = spi_nor_read_any_reg(flash, ofs, (u8)dummy, &val);
022 +    if (ret == 0)
023 +        printf("%02X\n", val);
024 +    else
025 +        puts("failed to read register\n");
026 +    return ret == 0 ? 0 : 1;
027 +}
028 +
029 +static int do_spi_wrar(int argc, char * const argv[])
030 +{
031 +    int ret = 0;
032 +    loff_t ofs;
033 +    ulong val;
034 +    if (argc != 3)
035 +        return -1;
```
Accessing SPI NOR flash registers in U-Boot console

Source code modification

Code Listing 5

```c
+ 047  + if (!str2off(argv[1], &ofs)) {
+ 048  +     puts("register address is not a valid number\n");
+ 049  +     return 1;
+ 050  + }
+ 051  +
+ 052  + if (!str2long(argv[2], &val)) {
+ 053  +     puts("val is not a valid number\n");
+ 054  +     return 1;
+ 055  + }
+ 056  +
+ 057  + ret = spi_nor_write_any_reg(flash, ofs, (u8)val);
+ 058  +
+ 059  + return ret == 0 ? 0 : 1;
+ 060  +}
+ 061  +
+ 062  + #ifdef CONFIG_CMD_SF_TEST
+ 063  + enum {
+ 064  +     STAGE_ERASE,
+ 065  +     @@ -582,6 +636,10 @@ static int do_spi_flash(struct cmd_tbl *cmdtp, int flag, int argc,
+ 066  +     @@ -582,6 +636,10 @@ int do_spi_flash_erase(argc, argv);
+ 067  +     else if (strcmp(cmd, "protect") == 0)
+ 068  +         ret = do_spi_protect(argc, argv);
+ 069  +     else if (strcmp(cmd, "rdar") == 0)
+ 070  +     +     ret = do_spi_rdar(argc, argv);
+ 071  +     +     else if (strcmp(cmd, "wrar") == 0)
+ 072  +     +         ret = do_spi_wrar(argc, argv);
+ 073  +     #ifdef CONFIG_CMD_SF_TEST
+ 074  +     +     else if (!strcmp(cmd, "test"))
+ 075  +     +         ret = do_spi_flash_test(argc, argv);
+ 076  +     +     else if (!strcmp(cmd, "test"))
+ 077  +     +         ret = do_spi_flash_test(argc, argv);
+ 078  +     + }
+ 079  +
+ 080  + DIFF --git a/drivers/mtd/spi/spi-nor-core.c
+ 081  + b/drivers/mtd/spi/spi-nor-core.c
+ 082  + index f1b4e5ea8e..4a8726884f 100644
+ 083  + --- a/drivers/mtd/spi/spi-nor-core.c
+ 084  + +++ b/drivers/mtd/spi/spi-nor-core.c
+ 085  + @@ -3858,3 +3858,21 @@ int
+ 086  +     spi_flash_cmd_get_sw_write_prot(struct spi_nor *nor)
+ 087  +
```

Accessing SPI NOR flash registers in U-Boot console

Source code modification

Code Listing 5

```c
093     return (sr >> 2) & 7;
094 }
095 +
096 +#ifdef CONFIG_SPI_FLASH_SPANSION
097     +int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr, u8
dummy, u8 *val)
098     +{ return spansion_read_any_reg(nor, addr, dummy, val);
099     +}
100 +
101 +
102 +int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr, u8
val)
103 +{
104 +    int ret;
105 +
106 +    ret = write_enable(nor);
107 +    if (ret)
108 +        return ret;
109 +
110 +    return spansion_write_any_reg(nor, addr, val);
111 +}
112 +#endif
113
diff --git a/include/linux/mtd/spi-nor.h
   b/include/linux/mtd/spi-nor.h
114 index 4ceeae623d..83e576eb4d 100644
115 --- a/include/linux/mtd/spi-nor.h
116 +++ b/include/linux/mtd/spi-nor.h
117 @@ -611,4 +611,9 @@ static inline int spi_nor_remove(struct
118     int spi_nor_remove(struct spi_nor *nor);
119     +#endif
120 
121 +#ifdef CONFIG_SPI_FLASH_SPANSION
122     +int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr, u8
dummy, u8 *val);
123     +int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr, u8
val);
124     +#endif
125 +
#endif
```
Accessing SPI NOR flash registers in U-Boot console

Usage examples of sf with Infineon S25HS512T

4. **Usage examples of sf with Infineon S25HS512T**

1. Detect and initialize SPI NOR flash:
   
   ```
   > sf probe
   SF: Detected s25hs512t with page size 256 Bytes, erase size 256 KiB, total 64 MiB
   ```

2. Read Configuration Register 4 Non-volatile (CFR4N – address 000005h). In S25HS512T, reading Non-volatile register requires 8 dummy cycles by factory default. The factory default value of CFR4N is 08h.
   
   ```
   > sf rdar 000005 8
   08
   ```

   The CFR4N[7:5] controls the I/O driver output impedance (Table 3).

   **Table 3**

<table>
<thead>
<tr>
<th>CFR4N[7:5]</th>
<th>I/O driver output impedance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>45 Ω</td>
<td>Factory default</td>
</tr>
<tr>
<td>001</td>
<td>120 Ω</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>90 Ω</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>60 Ω</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>45 Ω</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>30 Ω</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>20 Ω</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>15 Ω</td>
<td></td>
</tr>
</tbody>
</table>

3. Write the CFR4N register to change the output impedance from the default values (45 Ω to 30 Ω).
   
   ```
   > sf wrar 000005 A8
   ```

4. Read Status Register 1 Volatile (STR1V – address 800000h) to check the completion of Write Any Register operation. In S25HS512T, reading volatile register require 0 dummy cycle by factory default. The value should be 00h if the Write Any Register operation is completed successfully.
   
   ```
   > sf rdar 800000 0
   00
   ```

5. Read Configuration Register 4 Volatile (CFR4V – address 800005h) to confirm. The volatile register is updated when the non-volatile register is written.
   
   ```
   > sf rdar 800005 0
   A8
   ```
5 Conclusion

This application note introduces how to configure the U-Boot, customize the source files to access SPI NOR flash registers in U-Boot console.
Accessing SPI NOR flash registers in U-Boot console

Revision history

<table>
<thead>
<tr>
<th>Document version</th>
<th>Date of release</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>2022-08-24</td>
<td>New application note</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.