Application Note

AM6x 和 TDA4VEN 上的 OSPI NOR 闪存调试支持

Biao Li, Kevin Peng

摘要

eMMC、NAND 或 NOR 等外部闪存对于 SOC 系统的启动至关重要，这使得外部闪存成为嵌入式系统中的关键组件。所有 TI MPU 器件（包括 AM6x 和 TDA4x）都支持 OSPI (八线串行外设接口) NOR 闪存，这是一种多功能存储芯片，具有片上执行、读取速度快、无坏块和高稳定性等特点。这种类型的 NOR 闪存广泛用于 TI MPU 应用中的启动代码存储，尤其是在需要早期 CAN 响应和高稳定性的汽车应用场景中。

OSPI 模块支持对外部闪存设备进行单线、双线、四线或八线读写访问，但由于涉及多种协议和功能，给调试带来了复杂性。该模块支持直接内存映射模式（用于直接从外部闪存执行代码）和间接模式（通过内部 SRAM 进行数据传输，并通过中断或状态寄存器发出信号）。这种灵活性虽然对性能有益，但使调试工作流变得更加复杂。

本应用手册重点介绍一种使用 TI MPU 器件读取 NOR 闪存中状态和配置寄存器的方法，解决 OSPI 多协议架构带来的挑战。

内容

商标

1 简介

在嵌入式系统中，存储器是必不可少的组件。Sitara™ 用于存储 ROM 代码的内部 ROM 有限。客户或用户无法访问代码。需要外部闪存来存储客户代码和用户代码，例如，SBL/SPL、uboot 等。除 AM62A 外，几乎所有 TI Sitara MPU EVM 均连接一个 NOR 闪存。由于 NOR 闪存驱动程序的复杂性，这些与 NOR 闪存相关的问题可能难以调试。本应用手册提供一些读取 NOR 闪存内部寄存器的示例方法，以帮助客户识别问题。

本文档描述如何在 U-Boot 控制台中访问 NOR 闪存器件中的配置寄存器。该文档介绍基于 U-Boot 命令行界面的简单 SPI NOR 闪存访问命令的源代码片段和使用示例。本文档介绍如何在 MCU PLUS SDK 中添加寄存器打印或更改寄存器。假设用户具备构建和定制 U-Boot 和 TI MCU PLUS SDK 的知识和经验。

本应用手册以 Cypress NOR 闪存(连接在 EVM 上)为例。如果使用任何其他供应商的 NOR 闪存，请按照本指南，在 TI MCU PLUS SDK 中添加支持。

2 访问 SPI NOR 闪存寄存器

通常，SPI NOR 闪存器件具有单独的非易失性和易失性寄存器。在上电，硬件复位或软件复位期间，非易失性寄存器中的内容将自动加载到对应的易失性寄存器中。非易失性寄存器用于在系统引导前应用默认设置，而易失性寄存器用于在系统运行时更改设置。这是因为，非易失性寄存器基于更新周期有限的闪存存储单元，与易失性寄存器相比，更新时间更长，且在更新期间不能中断电源。

NOR 闪存包括两个状态寄存器名称：状态寄存器 1 和状态寄存器 2。状态寄存器 1 包含状态位和控制位。状态寄存器 2 提供器件运行状态。状态寄存器中的几乎所有位都是只读的，因此只需要读取易失性寄存器。共有五个配置寄存器。这些配置寄存器可以读取和写入。通常，这些寄存器必须由产品线上的闪存编程器进行更新。客户开发和评估系统时，可以在实验室中更新非易失性寄存器，尤其是在系统编程中。TI 不建议客户随意频繁地更改配置寄存器。当对配置寄存器进行任何更改时，产品线必须联系闪存供应商或 TI 支持窗口。

表 2-1. 表 1 NOR 闪存状态和配置寄存器映射

寄存器类型	寄存器名称	易失性地址（十六进制）	非易失性地址（十六进制）
状态寄存器 1	STR1N[7:0],STR1V[7:0]	0x00800000	0x00000000
状态寄存器 2	STR2V[7:0]	0x00800001	不适用
配置寄存器 1	CFR1N[7:0],CFR1V[7:0]	0x00800002	0x00000002
配置寄存器 2	CFR2N[7:0],CFR2V[7:0]	0x00800003	0x00000003
配置寄存器 3	CFR3N[7:0],CFR3V[7:0]	0x00800004	0x00000004
配置寄存器 4	CFR4N[7:0],CFR4V[7:0]	0x00800005	0x00000005
配置寄存器 5	CFR5N[7:0],CFR5V[7:0]	0x00800006	0x00000006

2.1 修改 U-Boot 源代码，以在 U-Boot 控制台中访问 NOR 闪存寄存器

在 MTD 驱动程序中，添加用于寄存器访问的全局范围函数。在 linux sdk 的 /board-support/ti-u-boot/drivers/mtd/spi/spi/spi-nor-core.c 和 /board-support/ti-u-boot/include/linux/mtd/spi-nor.h 文件中，有两个文件作用域的函数 spansion_read_any_reg() 和 spansion_write_any_reg()，可以执行寄存器访问操作。添加用于包装局部函数的全局作用域函数，以便命令行工具可以使用这些函数。TI 建议将代码添加到源文件的底部。补丁如下代码所示。

```
diff --git a/drivers/mtd/spi/spi-nor-core.c
b/drivers/mtd/spi/spi-nor-core.c
index 232a0ac3a9..b532ac6a08 100644
--- a/drivers/mtd/spi/spi-nor-core.c
+++ b/drivers/mtd/spi/spi-nor-core.c
@@ -4178,3 +4195,24 @@ int spi_flash_cmd_get_sw_write_prot(struct spi_nor *nor)

        return (sr >> 2) & 7;
 }
+
+
+#ifdef CONFIG_SPI_FLASH_SPANSION
+ int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr,
+ u8 dummy, u8 *val)
+ {
+ return spansion_read_any_reg(nor, addr, dummy, val);
+ }
+
+ int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr,
+ u8 val)
+ {
+ int ret;
+
+ ret = write_enable(nor);
+ if (ret)
+ return ret;
+
+ return spansion_write_any_reg(nor, addr, val);
+ }
+ #endif
diff --git a/include/linux/mtd/spi-nor.h
b/include/linux/mtd/spi-nor.h
index 2861b73edb..01a651ebeb 100644
```

```
--- a/include/linux/mtd/spi-nor.h
+++ b/include/linux/mtd/spi-nor.h
@@ -652,3 +652,10 @@ int spi_nor_remove(struct spi_nor *nor);
 #endif
 #endif
+
+#ifdef CONFIG_SPI_FLASH_SPANSION
+ int spi_nor_read_any_reg(struct spi_nor *nor, u32 addr,
+ u8 dummy, u8 *val);
+ int spi_nor_write_any_reg(struct spi_nor *nor, u32 addr,
+ u8 val);
+ #endif
\ No newline at end of file
```

使用以下代码，在 U-Boot 控制台中，添加提供寄存器访问的新命令。 ~/board-support/ti-u-boot/cmd/sf.c 定义可从 U-Boot 使用的 SPI 闪存访问命令，如读取、写入和擦除。do_spi_rdar() 函数响应 *rdar* 命令执行寄存器读取操作。将命令行参数解析为寄存器地址和虚拟周期数，然后将寄存器值打印到控制台。do_spi_wrar () 执行寄存器写入以响应 *wrar* 命令。将命令行参数解析为寄存器地址和要写入的新寄存器值。

若要调用这些函数，必须在现有的 do_spi_flash() 函数控制台中插入其他 else-if 块。

```
diff --git a/cmd/sf.c
b/cmd/sf.c
index 11b9c25896..465bf962ec 100644
--- a/cmd/sf.c
+++ b/cmd/sf.c
@@ -160,6 +160,59 @@ static int do_spi_flash_probe(int argc, char *const argv[])
         return 0;
 }
+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\n");
+                return 1;
+               }
+               if (!str2long(argv[2], &dummy)) {
+                puts("register address is not a valid number\n");
+               return 1;
+                }
+                ret = spi_nor_read_any_reg(flash, ofs, (u8)dummy, &val);
+                if (ret == 0)
+                printf("%02X\n", val);
+                else
+                puts("failed to read register\n");
+                return ret == 0 ? 0 : 1;
+        }
+       static int do_spi_wrar(int argc, char * const argv[])
+       {
+               int ret = 0;
+               loff_t ofs;
+                ulong val;
+               if (argc != 3)
+                return -1;
+               if (!str2off(argv[1], &ofs)) {
+               puts("register address is not a valid number\n");
+               return 1;
+                }
+               if (!str2long(argv[2], &val)) {
+               puts("val is not a valid number\n");
+               return 1;
+                }
+                ret = spi_nor_write_any_reg(flash, ofs, (u8)val);
+
+               return ret == 0 ? 0 : 1;
+       }
 /**
  * Write a block of data to SPI flash, first checking if it is different from
@@ -603,6 +656,10 @@ static int do_spi_flash(struct cmd_tbl *cmdtp, int flag, int argc,
                ret = do_spi_protect(argc, argv);
```

```
        else if (IS_ENABLED(CONFIG_CMD_SF_TEST) && !strcmp(cmd, "test"))
                ret = do_spi_flash_test(argc, argv);
+       else if (strcmp(cmd, "rdar") == 0)
+  ret = do_spi_rdar(argc, argv);
+ else if (strcmp(cmd, "wrar") == 0)
+ ret = do_spi_wrar(argc, argv);
        else
                ret = CMD_RET_USAGE;
```

此外，要添加一些要打印到控制台的调试日志，请使用以下代码作为参考来打印 opcode、nbytes、buswidth、dtr，dummy.nbytes 参数，以实现更高效的调试。

```
diff --git a/drivers/mtd/spi/spi-nor-core.c
b/drivers/mtd/spi/spi-nor-core.c
index 232a0ac3a9..b532ac6a08 100644
--- a/drivers/mtd/spi/spi-nor-core.c
+++ b/drivers/mtd/spi/spi-nor-core.c
@@ -335,6 +335,17 @@ static int spansion_read_any_reg(struct spi_nor *nor, u32 addr, u8 dummy,
                           SPI_MEM_OP_DUMMY(dummy / 8, 1),
                           SPI_MEM_OP_DATA_IN(1, NULL, 1));

+ printf("*** [%s] [%d] op.cmd.opcode= [%d]\n",__func__,__LINE__, op.cmd.opcode);
+        printf("*** [%s] [%d] op.cmd.nbytes= [%d]\n",__func__,__LINE__, op.cmd.nbytes);
+         printf("*** [%s] [%d] op.cmd.buswidth= [%d]\n",__func__,__LINE__, op.cmd.buswidth);
+          printf("*** [%s] [%d] op.cmd.dtr= [%d]\n",__func__,__LINE__, op.cmd.dtr);
+
+
+//        printf("*** [%s] [%d] op.addr.val= [%d]\n",__func__,__LINE__, op.addr.val);
+        printf("*** [%s] [%d] op.dummy.nbytes= [%d]\n",__func__,__LINE__, op.dummy.nbytes);
+
+
+
        return spi_nor_read_write_reg(nor, &op, val);
 }

@@ -414,6 +425,12 @@ static ssize_t spi_nor_read_data(struct spi_nor *nor, loff_t from, size_t len,
                op.data.buf.in += op.data.nbytes;
        }

+      printf("*** [%s] [%d] op.cmd.opcode= [%d]\n",__func__,__LINE__, op.cmd.opcode);
+          printf("*** [%s] [%d] op.cmd.nbytes= [%d]\n",__func__,__LINE__, op.cmd.nbytes);
+           printf("*** [%s] [%d] op.cmd.buswidth= [%d]\n",__func__,__LINE__, op.cmd.buswidth);
+            printf("*** [%s] [%d] op.cmd.dtr= [%d]\n",__func__,__LINE__, op.cmd.dtr);
+            printf("*** [%s] [%d] op.dummy.nbytes= [%d]\n",__func__,__LINE__, op.dummy.nbytes);
+
        return len;
 }
```

若要将模式更改为 1 行模式，请将闪存保持在尽可能低的设置值。可以应用以下代码；此项更改可选。以下代码是基于 AM64 SDK 的示例。

```
diff --git a/arch/arm/dts/k3-am642-evm.dts
b/arch/arm/dts/k3-am642-evm.dts
index 7f82730736..76d660a561 100644
--- a/arch/arm/dts/k3-am642-evm.dts
+++ b/arch/arm/dts/k3-am642-evm.dts
@@ -463,8 +463,8 @@
         flash@0 {
                 compatible = "jedec,spi-nor";
                 reg = <0x0>;
-                spi-tx-bus-width = <8>;
-                spi-rx-bus-width = <8>;
+                spi-tx-bus-width = <1>;
+                spi-rx-bus-width = <1>;
                 spi-max-frequency = <25000000>;
                 cdns,tshsl-ns = <60>;
                 cdns,tsd2d-ns = <60>;
diff --git a/arch/arm/dts/k3-am642-r5-evm.dts
b/arch/arm/dts/k3-am642-r5-evm.dts
index 6b32be1c4c..482f1ef639 100644
--- a/arch/arm/dts/k3-am642-r5-evm.dts
+++ b/arch/arm/dts/k3-am642-r5-evm.dts
@@ -308,8 +308,8 @@
         flash@0{
                 compatible = "jedec,spi-nor";
                 reg = <0x0>;
-                spi-tx-bus-width = <8>;
-                spi-rx-bus-width = <8>;
+                spi-tx-bus-width = <1>;
+                spi-rx-bus-width = <1>;
                 spi-max-frequency = <25000000>;
                 cdns,tshsl-ns = <60>;
                 cdns,tsd2d-ns = <60>;
diff --git a/arch/arm/dts/k3-am642-r5-sk.dts
b/arch/arm/dts/k3-am642-r5-sk.dts
index 6701e754bb..815dd5cb9b 100644
--- a/arch/arm/dts/k3-am642-r5-sk.dts
+++ b/arch/arm/dts/k3-am642-r5-sk.dts
@@ -290,8 +290,8 @@
         flash@0{
                 compatible = "jedec,spi-nor";
                 reg = <0x0>;
-                spi-tx-bus-width = <8>;
-                spi-rx-bus-width = <8>;
+                spi-tx-bus-width = <1>;
+                spi-rx-bus-width = <1>;
                 spi-max-frequency = <25000000>;
                 cdns,tshsl-ns = <60>;
                 cdns,tsd2d-ns = <60>;
diff --git a/arch/arm/dts/k3-am642-sk.dts
b/arch/arm/dts/k3-am642-sk.dts
index cbb10a4964..b2ee7e6965 100644
--- a/arch/arm/dts/k3-am642-sk.dts
+++ b/arch/arm/dts/k3-am642-sk.dts
@@ -482,8 +482,8 @@
         flash@0 {
                 compatible = "jedec,spi-nor";
                 reg = <0x0>;
-                spi-tx-bus-width = <8>;
-                spi-rx-bus-width = <8>;
+                spi-tx-bus-width = <1>;
+                spi-rx-bus-width = <1>;
                 spi-max-frequency = <25000000>;
                 cdns,tshsl-ns = <60>;
                 cdns,tsd2d-ns = <60>;
```

2.2 修改 MCU Plus 和 RTOS SDK 源代码，以访问 NOR 闪存寄存器

在 MCU Plus SDK 中，某些闪存需要额外的配置才能完全正常工作。混合扇区配置就是例子。这是一个添加此类特性的挂钩，在闪存驱动程序中打开函数的末尾调用此处提到的函数。在 MCU plus SDK 中，该默认函数位于 *~/ /source/board/flash/ospi/flash_nor_ospi.c*，名称为 *Flash_quirkSpansionUNHYSADisable()*。使用此函数添加一些调试信息，以帮助解决与 NOR 闪存相关的问题。使用以下示例，添加寄存器打印，并在驱动器中将 NOR 闪存模式从统一模式更改为混合顶部模式。

注意以下步骤：

1. 使用易失性寄存器地址进行读取；使用非易失性寄存器地址进行写入。
2. 请勿随意修改（写入）NOR 闪存寄存器。在更改配置寄存器的值之前，请联系 NOR 闪存供应商，并提交 E2E 工单，以确认更改。
3. 使用以下模板（首先读出，不相等时再写入）写入寄存器，以避免多次写入。使用以下代码，强制 NOR FLASH 进入统一模式。

```
void loop_forever()
{
    volatile uint32_t loop = 1;
    while(loop)
        ;
}
int32_t Flash_quirkSpansionUNHYSADisable(Flash_Config *config)
{
    int32_t status = SystemP_SUCCESS;
    uint8_t regData = 0x00;
uint32_t write = 0;
status |= Flash_norOspiRegRead(config, 0x65, 0x00800002, &regData);
DebugP_log("Value of Configuration Register 1: %u \r\n", regData);
regData = 0x00;
status |= Flash_norOspiRegRead(config, 0x65, 0x00800004, &regData);
DebugP_log("Value of Configuration Register 3: %u \r\n", regData);
    /* Hybrid Sector Disable */
    status = Flash_norOspiRegRead(config, 0x65, 0x00800004, &regData);
        if((regData & ((uint8_t)(1 << 3))) == 0)
        {
            /* Set UNHYSA bit */
            regData |= (1 << 3);
            write = 1U;
        }
        else
        {
            /* No action */
        }
    if(write)
    {
        status = Flash_norOspiRegWrite(config, 0x71, 0x04, regData);
}
regData = 0x00;
status = Flash_norOspiRegRead(config, 0x65, 0x00800000, &regData);
    if(regData!=0)
    {
DebugP_log("ERROR!!!!!! Need Reset board, State Register1 !=0. Value of State Register 1: %u \r\n",
regData);
loop_forever();
}
return status;
}
```

打开闪存驱动程序后，将打印调试信息。

3 调试 NOR 闪存的使用示例

在 U-boot 和 MCU PLUS SDK 中进行上述更改后，本节将介绍如何使用代码调试 NOR 闪存问题。

3.1 在 U-Boot 控制台中读取和写入 NOR Flash 寄存器

首先，使用 *sf probe*，检测和初始化 SPI NOR 闪存：

```
> sf probe
SF: Detected s25hs512t with page size 256 Bytes, erase size 256 KiB,
total 64 MiB
```

使用 *rdar* 读取 NOR 闪存中的寄存器。请注意，大多数情况下，读取时只读取易失性寄存器。易失性寄存器地址为 80000xh。例如，在读取配置寄存器 4（易失性，CFR4V - 地址 800005h）时，读取易失性地址后，用户也可以获得非易失性（CFR4N - 地址 000005h）的值，在复位后，非易失性值会加载到易失性寄存器中。在 NOR 闪存中，读取易失性寄存器出厂默认需要 0 个虚拟周期。读取非易失性寄存器出厂默认需要八个虚拟周期。使用以下代码读取。

```
> sf rdar 000005 8
08
> sf rdar 800005 0
08
```

使用 *wrar* 进行写入，但请注意，用户必须写入非易失性寄存器，写入易失性寄存器的值在断电后会丢失。写入完成后，需要读取易失性读状态寄存器 1（STR1V - 地址 800000h）以检查“写入任意寄存器”操作的完成情况。如果写入任何寄存器操作成功完成，则该值必须为 00h。一旦非易失性存储器被恢复出厂默认设置，请使用以下代码将新值写入配置寄存器 4：

```
> sf wrar 000005 A8
> sf rdar 800000 0
00
> sf rdar 800005 0
A8
```

3.2 在 MCU Plus SDK 中读写 NOR Flash 寄存器

由于在 NOR 闪存驱动程序中更改了代码，因此，一旦调用该驱动程序,就会执行在 *Flash_quirkSpansionUNHYSADisable()* 中添加的代码。可通过以下方式调用代码：

Board_driversOpen()-> Board_flashOpen() -> Flash_open()-> Flash_quirkSpansionUNHYSADisable()

若要获取寄存器在代码更改后的值，需要编译 SBL（假设客户使用 NOR 闪存启动模式）或使用 NOR 闪存驱动的应用程序。打开 NOR 闪存驱动程序后，可以查看日志打印内容。

下面的代码显示了日志输出。这会打印在 SBL1 中，因为 NOR 闪存驱动程序未打开，当 MCU 应用程序映像开始运行时，由于 NOR 闪存驱动程序再次打开，代码会再次打印。

```
Value of Configuration Register 1: 4
Value of Configuration Register 3: 0
SYSFW Firmware Version 9.2.7--v09.02.07 (Kool Koala)
SYSFW Firmware revision 0x9
SYSFW ABI revision 3.1
[BOOTLOADER_PROFILE] Boot Media : FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 200.000MHz
[BOOTLOADER_PROFILE] Boot Image Size : 138 KB
[BOOTLOADER_PROFILE] Cores present :
r5f0-0
[BOOTLOADER PROFILE] System_init : 23864us
[BOOTLOADER PROFILE] Board_init : 0us
[BOOTLOADER PROFILE] Drivers_open : 192us
[BOOTLOADER PROFILE] Board_driversOpen : 69896us
[BOOTLOADER PROFILE] Sciclient Get Version : 10164us
[BOOTLOADER PROFILE] App_waitForMcuPbist : 5489us
[BOOTLOADER PROFILE] App_loadSelfcoreImage_A : 4592us
[BOOTLOADER_PROFILE] SBL Total Time Taken : 114201us
Image loading done, switching to application ...
```

```
Starting MCU-r5f and 2nd stage bootloader
Value of Configuration Register 1: 4
Value of Configuration Register 3: 0
SYSFW Firmware Version 9.2.7--v09.02.07 (Kool Koala)
SYSFW Firmware revision 0x9
SYSFW ABI revision 3.1
```

4 总结

本应用手册总结如何使用 TI Sitara 和 Jacinto™ MPU 读取和写入 NOR 闪存寄存器。客户可以轻松使用 U-Boot 控制台命令。简化复杂时序：U-Boot 封装了闪存命令序列，从而简化操作的复杂性。使用 TI 本机 MCU PLUS SDK 中提供的自定义功能接口，可以方便地打印相应的 NOR 闪存寄存器，并修改对应的寄存器。本文希望通过提供上述方法，简化客户调试问题的难度，并缩短解决问题的时间。

5 参考资料

1. 德州仪器 (TI)，*AM62x Sitara™ 处理器*，数据表。
2. 德州仪器 (TI)，*AM62A3：NorFlash S28HS512T 扇区擦除失败*，E2E 论坛。
3. 英飞凌科技，*在 U-Boot 控制台中访问 SPI NOR 闪存寄存器*
4. 德州仪器 (TI)，*AM62x MCU+ SDK：添加对定制闪存器件的支持*，论坛。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。 严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265