参考資料

Application Note

AM6x と TDA4VEN の OSPI NOR フラッシュ デバッグをサポート

Biao Li, Kevin Peng

概要

SOC システムの起動には eMMC、NAND、NOR などの外部フラッシュが不可欠であり、組込みシステムでは外部フラッシュが重要なコンポーネントです。AM6x や TDA4x を含むすべての TI MPU デバイスは、OSPI (オクタル シリアル ペリフェラル インターフェイス) NOR フラッシュをサポートしています。NOR フラッシュは、オンチップ実行、高速な読み取り速度、不適切なブロックなし、高い安定性を特徴とする多用途メモリ チップです。このタイプの NOR フラッシュは、TI の MPU アプリケーションのブート コード ストレージに広く使用されており、特に早期の CAN 応答と高い安定性が要求される車載コンテキストで必要です。

OSPI モジュールにより、外部フラッシュ デバイスへのシングル、デュアル、クワッド、またはオクタルの読み取り/ 書き込みアクセスが可能になるため、多様なプロトコルと機能によりデバッグが複雑になります。このモジュールは、ダイレクト メモリ マップ モード (外部フラッシュからコードを直接実行するため) と間接モード (データ転送は内部 SRAM を通じて行われ、割り込みまたはステータス レジスタによって通知されます) の両方をサポートしています。この柔軟性は、パフォーマンスには有益ですが、デバッグ ワークフローが複雑になります。

このアプリケーション ノートでは、TI の MPU デバイスを使用して NOR フラッシュでステータス レジスタと構成レジスタを読み取る方法を紹介し、OSPI のマルチプロトコル アーキテクチャがもたらす課題に対処します。

目次

商標

1 はじめに

組込みシステムでは、メモリは不可欠なコンポーネントです。Sitara™ は、ROM コードを格納するために使用される内部 ROM が制限されています。顧客またはユーザーがコードにアクセスすることはできません。SBL/SPL や Uboot などの顧客コードやユーザー コードを保存するには、外部フラッシュが必要です。TI Sitara MPU EVM のほぼすべてにおいて、AM62A を除く NOR フラッシュを 1 つ取り付けています。NOR フラッシュ ドライバは複雑なため、NOR フラッシュに関連するこれらの問題のデバッグが困難な場合があります。このアプリケーション ノートでは、お客様が問題を識別するのに役立つ、NOR フラッシュ内部レジスタを読み取るためのいくつかのサンプル方法を説明します。

このドキュメントでは、U-Boot コンソールの NOR フラッシュ デバイスの構成レジスタにアクセスする方法について説明します。このドキュメントでは、ソース コード スニペットと、U-Boot コマンド ライン インターフェイスをベースとするシンプルな SPI NOR フラッシュ アクセス コマンドの使用例を紹介します。このドキュメントでは、マイコン + SDK でレジスタ表示または変更レジスタを追加する方法を示します。ユーザーが、U-Boot と TI マイコン + SDK の構築とカスタマイズに関する知識と経験を持っていると仮定します。

このアプリケーション ノートでは、例として Cypress NOR フラッシュ (評価基板に取り付け) を使用しています。他のベンダの NOR フラッシュを使用する場合は、このガイドに従って、TI の マイコン + SDK のサポートを追加します。

2 SPI NOR フラッシュ レジスタへのアクセス

通常、SPI NOR フラッシュ デバイスには、個別の不揮発性レジスタと揮発性レジスタがあります。電源投入時、ハードウェア リセット時、またはソフトウェア リセット時に、不揮発性レジスタの内容がカウンタ用の揮発性レジスタに自動的にロードされます。不揮発性レジスタはシステム ブート前にデフォルト設定を適用するために使用され、揮発性レジスタはシステム実行時に設定を変更するために使用されます。これは、不揮発性レジスタがフラッシュ メモリ セルをベースとしているため、更新サイクルが制限されており、揮発性レジスタに比べて更新に時間がかかり、更新中の電源切断に耐性がないためです。

NOR フラッシュには、次の 2 つのステータス レジスタ名があります。ステータス レジスタ 1 とステータス レジスタ 2。ステータス レジスタ 1 には、ステータス ビットと制御ビットが両方含まれています。ステータス レジスタ 2 は、動作に関するデバイス ステータスを示します。ステータス レジスタのほぼすべてのビットは読み取り専用であるため、揮発性レジスタのみを読み取る必要があります。構成レジスタは 5 つあります。これらの構成レジスタは、読み取りと書き出しの両方が可能です。一般に、これらのレジスタは製品ラインのフラッシュ プログラマによって更新される必要があります。お客様がシステムを開発および評価する際は、ラボで不揮発性レジスタを更新する方法、特にインシステム プログラミングを行う方法があります。TI では、お客様が構成レジスタを随意かつ頻繁に変更することは推奨していません。構成レジスタに何らかの変更を加えた場合、製品ラインは、フラッシュ ベンダまたは TI サポート ウィンドウに連絡する必要があります。

表 2-1. 表 1 NOR フラッシュ ステータスと構成レジスタ マップ

レジスタ タイプ	レジスタ名	揮発性アドレス (16 進)	不揮発性アドレス (16 進)
ステータス レジスタ 1	STR1N[7:0]、STR1V[7:0]	0x00800000	0x00000000
ステータス レジスタ 2	STR2V[7:0]	0x00800001	該当なし
設定レジスタ 1	CFR1N[7:0]、CFR1V[7:0]	0x00800002	0x00000002
設定レジスタ 2	CFR2N[7:0]、CFR2V[7:0]	0x00800003	0x00000003
設定レジスタ 3	CFR3N[7:0]、CFR3V[7:0]	0x00800004	0x00000004
設定レジスタ 4	CFR4N[7:0]、CFR4V[7:0]	0x00800005	0x00000005
設定レジスタ 5	CFR5N[7:0]、CFR5V[7:0]	0x00800006	0x00000006

2.1 U-Boot コンソールの NOR フラッシュ レジスタにアクセスするための U-Boot ソース コードの変更

MTD ドライバのレジスタ アクセスのグローバル スコープ関数を追加。Linux SDK の /board-support/ti-u-boot/drivers/mtd/spi/spi/spi-nor-core.c および/ board-support/ti-u-boot/include/linux/mtd/spi-nor.h には、ファイル スコープ関数 spansion_read_any_reg() および spansion_write_any_reg() があり、こレジスタ アクセス操作を実行できます。ローカル関数をラップするグローバル スコープ関数を追加して、コマンド ライン ツールで使用できるようにします。TI では、このコードをソース ファイルの下部に追加することを推奨しています。パッチは次のコードに示されています。

```
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
```

以下のコードを使用して、U-Boot コンソールでレジスタ アクセスを提供する新しいコマンドを追加します。~/board-support/ti-u-boot/cmd/sf.c には、U-Boot からの使用が可能な読み取り、書き込み、消去などの SPI フラッシュ アクセス コマンドが定義されています。do_spi_rdar() は *rdar* コマンドに応答してレジスタ読み取りを実行します。コマンド ライン引数をレジスタ アドレスとダミー サイクル数として解析し、そのレジスタ値をコンソールに出力します。do_spi_wrar() は *wrar* コマンドに応答してレジスタ書き込みを実行します。コマンド ライン引数をレジスタ アドレスおよび書き込み用の新しいレジスタ値として解析します。

これらの関数を呼び出すには、既存の do_spi_flash() 関数コンソールに追加の else-if ブロックを挿入する必要があります。

```
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;
```

さらに、コンソールに出力するデバッグ ログを追加するには、より効率的なデバッグのために opcode、nbytes、buswidth、dtr、dummy.nbytes パラメータを出力するための参照として次のコードを使用します。

```
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;
 }
```

モードを 1 ライン モードに変更するには、フラッシュを可能な限り低い設定に維持します。以下のコードを適用できます。これはオプションの変更です。以下のコードは、AM64 SDK をベースとする例です。

```
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 NOR フラッシュ レジスタにアクセスするための マイコン + SDK と RTOS SDK ソース コードの変更

マイコン + SDK では、特定のフラッシュでフラッシュが完全に動作するように追加の構成を必要とします。たとえば、ハイブリッド セクター構成です。これは、そのようなコードを追加するためのフックであり、ここで説明した関数はフラッシュ ドライバの open 関数の最後に呼び出されます。マイコン + SDK では、このデフォルト関数は*~//source/board/flash/ospi/flash_nor_ospi.c* の *Flash_quirkSpansionUNHYSADisable()* に配置されています。この関数を使用して、NOR フラッシュに関連する問題を解決できるように、いくつかのデバッグ情報を追加します。次の例を使用して、レジスタ プリントを追加し、ドライバの NOR フラッシュ モードを均一モードからハイブリッド トップ モードに変更します。

次の手順に注意してください:

1. 揮発性レジスタ アドレスを使用して読み取り、不揮発性レジスタ アドレスを使用して書き込みます。
2. NOR フラッシュ レジスタを自由に変更 (書き込み) しないでください。構成レジスタの値を変更する前に、NOR フラッシュのベンダに問い合わせ、E2E チケットを送信して変更を確認してください。
3. 複数の書き込みを回避するには、レジスタの書き込みには以下のテンプレートを使用します (最初に読み取り、同じでなければ書き出し)。NOR フラッシュを強制的に均一モードにするには、次のコードを使用します。

```
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;
}
```

フラッシュ ドライバを開くと、デバッグ情報が表示されます。

3 NOR フラッシュをデバッグする使用例

このセクションでは、U-boot と マイコン + SDK で上記の変更を加えた後で、NOR フラッシュの問題をデバッグするために、コードを使用する方法を紹介します。

3.1 U-Boot コンソールでの NOR フラッシュ レジスタの読み取り / 書き込み

まず、*sf プローブ*を使用して SPI NOR フラッシュを検出および初期化します:

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

NOR フラッシュのレジスタを読み取るには、*rdar* を使用します。ほとんどのシナリオでは、読み取り時に揮発性レジスタのみを読み取ることに注意してください。揮発性レジスタ アドレスは 80000xh です。たとえば、構成レジスタ 4 の揮発性レジスタ (CFR4V – アドレス 800005h) を読み取ると、不揮発性値がリセット後に揮発性レジスタにロードされるため、揮発性アドレスを読み取り、ユーザーは不揮発性レジスタの値 (CFR4N – アドレス 000005h) の値も取得できます。NOR フラッシュでは、揮発性レジスタを読み取るには、工場出荷時デフォルトで 0 ダミー サイクルが必要です。不揮発性レジスタの読み取りには、工場出荷時デフォルトで 8 ダミー サイクルが必要です。次のコードを使用して読み取ります。

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

書き込みには *wrar* を使用しますが、ユーザーは不揮発性レジスタに書き込みを行う必要があり、電源オフの後に揮発性への値の書き込みはカバーされます。書き込み後にステータス レジスタ 1 の揮発性読み取り (STR1V – アドレス 800000h) を読み取って、任意のレジスタの書き込み動作の完了を確認してください。任意のレジスタの書き込み動作が正常に完了すると、値は 00h となります。不揮発性が工場出荷時のデフォルトで変更された場合、次のコードを使用して新しい値を構成レジスタ 4 に書き込みます。

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

3.2 MCU Plus SDK の NOR フラッシュ レジスタの読み取り / 書き込み

コードは NOR フラッシュ ドライバで変更されるため、ドライバが呼び出されると、*Flash_quirkSpansionUNHYSADisable()* に追加されたコードが実行されます。コードは、次の方法で起動されます。

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

レジスタの値を取得するには、コード変更後に、SBL (お客様が NOR フラッシュ ブート モードを使用していると仮定) または NOR フラッシュ ドライバを使用しているアプリケーションをコンパイルします。NOR フラッシュ ドライバを開いたら、ログ プリントを表示できます。

次のコードはログ出力を示しています。これは、NOR フラッシュ ドライバが開いているため、SBL1 で表示され、NOR フラッシュ ドライバが再び開いたために MCU アプリケーション イメージの実行を開始したときにコードが再度表示されます。

```
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 まとめ

このアプリケーション ノートでは、TI Sitara と Jacinto™ MPU を使用して NOR フラッシュ レジスタの読み書きを行う方法をまとめています。U-Boot コンソール コマンドは、お客様が簡単に使用できます。複雑なタイミングの簡素化：U-Boot はフラッシュ コマンド シーケンスをカプセル化するため、動作の複雑さを簡素化できます。TI のネイティブ マイコン + SDK で提供されているカスタム機能インターフェイスを使用すると、対応する NOR フラッシュ レジスタを表示し、対応するレジスタを利便性の高い方法で変更することができます。この記事を通じて、上記の方法を提供することで、お客様の問題をデバッグする難しさを簡単にし、問題解決までの時間を短縮できれば幸いです。

5 参考資料

1. テキサス インスツルメンツ、『*AM62x Sitara™ プロセッサ*』データシート
2. テキサス・インスツルメンツ、『*AM62A3：NorFlash S28HS512T セクタ消去の失敗*』、E2E フォーラム。
3. Infineon Technologies、『*U-Boot コンソールの SPI NOR フラッシュ レジスタへのアクセス*』
4. テキサス インスツルメンツ、『*AM62x MCU+ SDK：カスタム フラッシュ デバイスへのサポートの追加*』、フォーラム。

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月