Application Note

TDA4 系列的 SPI 启用和验证

Vignesh R and Keerthy J

摘要

本应用报告演示了如何使用 Linux® 为 TDA4 系列启用 SPI 的主模式和从模式，还为其中一个 SPI 实例提供了示例器件树节点，用于对主模式进行演示。从模式要求 DMA 支持，并需要额外的补丁来启用 UDMA 功能所需的 PSIL 线程。测试是使用标准的 Linux 方法演示的。

内容

商标

1 SPI：串行外设接口

SPI 是一种用于短距离通信的同步串行通信接口规范，主要用于嵌入式系统。

下面列出了 SPI 协议的显著特性：

- 串行接口
- 同步
- 主从配置
- 数据交换 - DMA/PIO

SPI 总线指定四个逻辑信号：

- SCLK：串行时钟（从主器件输出）
- MOSI：主器件出/从器件入（从主器件输出数据）
- MISO：主器件入/从器件出（从从器件输出数据）
- CS/SS：芯片/从器件选择（通常低电平有效，从主器件输出，指示正在发送数据）

主器件上的 MOSI 连接到从器件上的 MOSI。主器件上的 MISO 连接到从器件上的 MISO。

从器件选择与芯片选择具有相同的功能，并用于代替寻址概念。

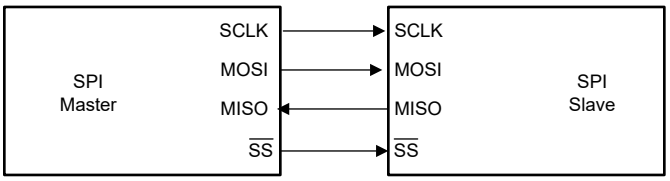

图 1-1. 标题缺失

2 J7200/J721e MCSPI 支持

MCSPI 表示多通道串行外设接口（MCSPI）。MCSPI 模块是一个多通道发送/接收、主/从同步串行总线。器件中有十一个 MCSPI 模块（请参阅表 2-1）。

表 2-1. MCSPI 概述

实例	域		
	WKUP	MCU	MAIN
MCU_MCSPI0	-	✓	-
MCU_MCSPI1	-	✓	-
MCU_MCSPI2	-	✓	-
MCSPI0	-	-	✓
MCSPI1	-	-	✓
MCSPI2	-	-	✓
MCSPI3	-	-	✓
MCSPI4	-	-	✓
MCSPI5	-	-	✓
MCSPI6	-	-	✓
MCSPI7	v	-	✓

有关更多信息，请参阅 *J7200 DRA821 处理器器件修订版 1.0 技术参考手册*。

表格缺失

默认情况下，MCSPI4 在上电时作为从器件直接连接到 MCU_MCSPI2。MCSPI4 和

MCU_MCSPI2 在外部没有引脚输出。

2.1 MCSPI 特性

MCSPI 模块包括以下主要特性：

- 具有可编程频率、极性和相位的串行时钟用于每个通道
- MCSPI 字长范围宽，从 4 位到 32 位
- 在主模式下多达四个通道工作，或在从模式下单个通道工作
- 针对多个中断源事件的单一中断线路
- 支持为每个通道的 MCSPI 传输添加可编程起始位（起始位模式）
- 支持起始位写入命令
- 支持起始位暂停和间断序列
- 可编程移位操作（1-32 位）
- 芯片选择与外部时钟生成之间的时序控制可编程
- 内置 FIFO 可用于单个通道。
- 主器件多通道模式：
  - 全双工/半双工
  - 仅发送/仅接收/发送和接收模式
  - 每个通道具有灵活的输入/输出（I/O）端口控制
  - 可编程时钟粒度
  - 支持每通道的 MCSPI 配置，即时钟定义、极性启用和字宽

3 SPI：Linux 上的主模式启用和验证

3.1 启用 J721e/TDA4VM 的 SPI 实例

我们以主域 SPI6 实例为例。要启用 SPI6，请添加器件树节点和相应的引脚多路复用节点。

```
diff --git a/arch/arm64/boot/dts/ti/k3-j721e-common-proc-board.dts b/arch/arm64/boot/dts/ti/k3-
j721e-common-proc-board.dts
index 6788a3611..77b845354 100644
--- a/arch/arm64/boot/dts/ti/k3-j721e-common-proc-board.dts
+++ b/arch/arm64/boot/dts/ti/k3-j721e-common-proc-board.dts
@@ -170,6 +170,18 @@
>;
};

+ spi6_pins_default: spi6_pins_default {
+ pinctrl-single,pins = <
+ J721E_IOPAD(0x9c, PIN_INPUT, 4) /* (AC22) PRG1_PRU1_GPO17.SPI6_CLK */
+ J721E_IOPAD(0x74, PIN_INPUT, 4) /* (AC21) PRG1_PRU1_GPO7.SPI6_CS0 */
+ J721E_IOPAD(0x28, PIN_INPUT, 4) /* (AG20) PRG1_PRU0_GPO9.SPI6_CS1 */
+ J721E_IOPAD(0x2c, PIN_INPUT, 4) /* (AD21) PRG1_PRU0_GPO10.SPI6_CS2 */
+ J721E_IOPAD(0x7c, PIN_INPUT, 4) /* (AF21) PRG1_PRU1_GPO9.SPI6_CS3 */
+ J721E_IOPAD(0xa0, PIN_INPUT, 4) /* (AJ22) PRG1_PRU1_GPO18.SPI6_D0 */
+ J721E_IOPAD(0xa4, PIN_INPUT, 4) /* (AH22) PRG1_PRU1_GPO19.SPI6_D1 */
+ >;
+ };
+

diff --git a/arch/arm64/boot/dts/ti/k3-j721e-main.dtsi b/arch/arm64/boot/dts/ti/k3-j721e-main.dtsi
index c036df124..edc42720f 100644
--- a/arch/arm64/boot/dts/ti/k3-j721e-main.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j721e-main.dtsi
@@ -74,6 +74,16 @@
};
};

+ main_spi6: spi@2160000 {
+ compatible = "ti,am654-mcspi","ti,omap4-mcspi";
+ reg = <0x0 0x2160000 0x0 0x400>;
+ interrupts = <GIC_SPI 190 IRQ_TYPE_LEVEL_HIGH>;
+ clocks = <&k3_clks 272 1>;
+ power-domains = <&k3_pds 272 TI_SCI_PD_EXCLUSIVE>;
+ #address-cells = <1>;
+ #size-cells = <0>;
+ };
+
```

已在 arch/arm64/configs/tisdk_j7-evm_defconfig 中设置了 CONFIG_SPI_OMAP24XX=y。SPI 主器件驱动程序为 drivers/spi/spi-omap2-mcspi.c。

3.2 在 TD4VM SDK 上启用 SPIDEV

在 arch/arm64/boot/dts/ti/k3-j7200-common-proc-board.dts 中，在 spi6 节点内添加一个 spidev 节点，如下所示：

```
+&main_spi6 {
+ pinctrl-names = "default";
+ pinctrl-0 = <&spi6_pins_default>;
+ status="okay";
+
+ spidev@0 {
+
+ spi-max-frequency = <24000000>;
+ reg = <0>;
+ compatible = "linux,spidev";
+};
+};
```

在 arch/arm64/configs/tisdk_j7-evm_defconfig 中显式启用 CONFIG_SPI_SPIDEV=y。

引导 Linux 后，应输入以下条目：

```
ls -l /sys/class/spi*
/sys/class/spi_master: total 0 lrwxrwxrwx 1 root root 0 Jun 17 14:17 spi6 -> ../../devices/platform/
interconnect@100000/2160000.spi/spi_master/spi6
/sys/class/spidev: total 0 lrwxrwxrwx 1 root root 0 Jun 17 14:17 spidev6.0 -> ../../devices/
platform/interconnect@100000/2160000.spi/spi_master/spi6/spi6.0/spidev/spidev6.0
```

3.3 使用标准 Linux spidev_test 工具在 TI J7/TDA4x 上的用户空间中练习 SPI

Linux 内核提供了 spidev_test 工具。我们只需要编译并使用它。请按照此处的指示操作：

```
cd ti-processor-sdk-linux-automotive-j7-evm-*/board-support/linux-*/tools/spi
make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu-
cp spidev_test /media/$user/rootfs/home/root
```

以上内容应在 tools/spi 文件夹中编译 spidev_test 二进制文件，并将其复制到目标文件系统的 rootfs。

TDA4VM Linux 命令提示符上的基本测试：

```
cd /home/root
./spidev_test -v -D /dev/spidev6.0 -p "HELLOWORLD"
```

输出：

```
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | 48 45 4C 4C 4F 57 4F 52 4C 44 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |HELLOWORLD|
RX | FF FF FF FF FF FF FF FF FF FF __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |..........|
```

由于没有从器件，对于 RX 我们总是看到 0xFF。

4 SPI：Linux 上的从模式启用和验证

要测试从模式，也需要主器件。利用 J7200 的硬件功能演示 SPI 从器件支持。

有关更多信息，请参阅 *J7200 DRA821 处理器器件修订版 1.0 技术参考手册* 中的“MCSPI 概述”一章。

默认情况下，MCSPI4 在上电时作为从器件直接连接到 MCU_MCSPI2。MCSPI4 和 MCU_MCSPI2 在外部没有引脚输出。

4.1 启用 J7200 的 SPI 实例

添加作为 SPI 主节点的 MCU_MCSPI2 节点和作为 SPI 从节点的 MAIN_MCPSI4 节点。

```
diff --git a/arch/arm64/boot/dts/ti/k3-j7200-common-proc-board.dts b/arch/arm64/boot/dts/ti/k3-
j7200-common-proc-board.dts
index 68e9369b6..bf37cc98f 100644
--- a/arch/arm64/boot/dts/ti/k3-j7200-common-proc-board.dts
+++ b/arch/arm64/boot/dts/ti/k3-j7200-common-proc-board.dts
@@ -256,6 +256,25 @@
status = "disabled";
};

+&mcu_spi2 {
+ status="okay";
+ spidev@0 {
+ spi-max-frequency = <24000000>;
+ reg = <0>;
+ compatible = "linux,spidev";
+ };
+};
+
+&main_spi4 {
+ status="okay";
+ spi-slave;
+ slave@0 {
+ spi-max-frequency = <24000000>;
+ reg = <0>;
+ compatible = "linux,spidev";
+ };
+};
+
&wkup_gpio1 {
status = "disabled";
};
diff --git a/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi b/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
index 79749f250..70a028481 100644
--- a/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
@@ -426,6 +426,16 @@
clock-names = "fclk";
};

+ main_spi4: spi@2140000 {
+ compatible = "ti,am654-mcspi","ti,omap4-mcspi";
+ reg = <0x0 0x2140000 0x0 0x400>;
+ interrupts = <GIC_SPI 188 IRQ_TYPE_LEVEL_HIGH>;
+ clocks = <&k3_clks 270 1>;
+ power-domains = <&k3_pds 270 TI_SCI_PD_EXCLUSIVE>;
+ #address-cells = <1>;
+ #size-cells = <0>;
+ };
+
main_i2c0: i2c@2000000 {
compatible = "ti,j721e-i2c", "ti,omap4-i2c";
reg = <0x00 0x2000000 0x00 0x100>;
diff --git a/arch/arm64/boot/dts/ti/k3-j7200-mcu-wakeup.dtsi b/arch/arm64/boot/dts/ti/k3-j7200-mcu-
wakeup.dtsi
index be334bcfe..35ec3b0c3 100644
--- a/arch/arm64/boot/dts/ti/k3-j7200-mcu-wakeup.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j7200-mcu-wakeup.dtsi
@@ -70,6 +70,16 @@
#size-cells = <1>;
};

+ mcu_spi2: mcu-spi2@40320000 {
+ compatible = "ti,am654-mcspi","ti,omap4-mcspi";
```

```
+ reg = <0x0 0x40320000 0x0 0x400>;
+ interrupts = <GIC_SPI 850 IRQ_TYPE_LEVEL_HIGH>;
+ clocks = <&k3_clks 276 1>;
+ power-domains = <&k3_pds 276 TI_SCI_PD_EXCLUSIVE>;
+ #address-cells = <1>;
+ #size-cells = <0>;
+ };
+
wkup_uart0: serial@42300000 {
compatible = "ti,j721e-uart", "ti,am654-uart";
reg = <0x00 0x42300000 0x00 0x100>;
diff --git a/arch/arm64/boot/dts/ti/k3-j7200.dtsi b/arch/arm64/boot/dts/ti/k3-j7200.dtsi
index b7005b803..2b77308ae 100644
--- a/arch/arm64/boot/dts/ti/k3-j7200.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j7200.dtsi
@@ -152,6 +152,7 @@
#size-cells = <2>;
ranges = <0x00 0x28380000 0x00 0x28380000 0x00 0x03880000>, /* MCU NAVSS*/
<0x00 0x40200000 0x00 0x40200000 0x00 0x00998400>, /* First peripheral window */
+ <0x00 0x40320000 0x00 0x40320000 0x00 0x00000400>, /* MCU SPI2 */
<0x00 0x40f00000 0x00 0x40f00000 0x00 0x00020000>, /* CTRL_MMR0 */
<0x00 0x41000000 0x00 0x41000000 0x00 0x00020000>, /* MCU R5F Core0 */
<0x00 0x41400000 0x00 0x41400000 0x00 0x00020000>, /* MCU R5F Core1 */
--
```

4.2 为 MCSPI4 从节点启用 DMA

默认情况下，SDK 8.1 并不具有 MCSPI 实例所需的 UDMA PSIL 线程。添加这些例程，并添加 SPI 从模式功能所需的 DMA 属性。

```
diff --git a/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi b/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
index 70a028481..4af897173 100644
--- a/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j7200-main.dtsi
@@ -434,6 +434,8 @@
power-domains = <&k3_pds 270 TI_SCI_PD_EXCLUSIVE>;
#address-cells = <1>;
#size-cells = <0>;
+ dmas = <&main_udmap 0xc610>, <&main_udmap 0x4610>;
+ dma-names = "tx0", "rx0";
};

main_i2c0: i2c@2000000 {
diff --git a/drivers/dma/ti/k3-psil-j7200.c b/drivers/dma/ti/k3-psil-j7200.c
index 5ea63ea74..06ae1c1a2 100644
--- a/drivers/dma/ti/k3-psil-j7200.c
+++ b/drivers/dma/ti/k3-psil-j7200.c
@@ -164,6 +164,11 @@ static struct psil_ep j7200_dst_ep_map[] = {
/* SA2UL */
PSIL_SA2UL(0xf500, 1),
PSIL_SA2UL(0xf501, 1),
+ /* PDMA_SPI_G1 - SPI4 */
+ PSIL_PDMA_XY_PKT(0xc610),
+ PSIL_PDMA_XY_PKT(0xc611),
+ PSIL_PDMA_XY_PKT(0xc612),
+ PSIL_PDMA_XY_PKT(0xc613),
};
```

4.3 启用 SPIDEV 和 SPI_SLAVE 配置

需要在 arch/arm64/configs/tisdk_j7200-evm_defconfig 中启用两个配置：

- Change #CONFIG_SPI_SPIDEV is not set ---> CONFIG_SPI_SPIDEV=y
- Change #CONFIG_SPI_SLAVE is not set ---> CONFIG_SPI_SLAVE=y

4.4 使用标准 Linux spidev_test 工具在 TI J7200 上的用户空间中测试 SPI 从模式功能

还利用在上述步骤 3.3 中编译的 spidev_test 来演示从模式功能。

使用 spidev_test 实用程序启动 MCPSI4 从器件和 MCU_MCSPI2 之间的通信。

上述测试的日志：

```
root@j7200-evm:~# echo spidev > /sys/class/spi_slave/spi2/slave
root@j7200-evm:~# ./spidev_test -v -D /dev/spidev2.0 -p slave-hello-to-master &
[1] 1186
root@j7200-evm:~# ./spidev_test -v -D /dev/spidev0.0 -p master-hello-to-slave
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | 6D 61 73 74 65 72 2D 68 65 6C 6C 6F 2D 74 6F 2D 73 6C 61 76 65 __ __ __ __ __ __ __ __ __ __
__ |master-hello-to-slave|
RX | 73 6C 61 76 65 2D 68 65 6C 6C 6F 2D 74 6F 2D 6D 61 73 74 65 72 __ __ __ __ __ __ __ __ __ __
__ |slave-hello-to-master|
TX | 73 6C 61 76 65 2D 68 65 6C 6C 6F 2D 74 6F 2D 6D 61 73 74 65 72 __ __ __ __ __ __ __ __ __ __
__ |slave-hello-to-master|
RX | 6D 61 73 74 65 72 2D 68 65 6C 6C 6F 2D 74 6F 2D 73 6C 61 76 65 __ __ __ __ __ __ __ __ __ __
__ |master-hello-to-slave|
```

可以看到主器件 MCU_MCSPI2 发送了 TX 消息 master-hello-to-slave，并从 MCPSI4 从器件接收了 RX 消息 slave-hello-to-master，反之亦然。

4.5 使用 spi-slave-time 进行 SPI 从器件测试

```
echo spi-slave-time > /sys/class/spi_slave/spi2/slave
modprobe spi-slave-time
./spidev_test -v -D /dev/spidev0.0 -p "dummy-8B"
```

日志：

```
root@j7200-evm:~# ./spidev_test -v -D /dev/spidev0.0 -p "dummy-8B"
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | 64 75 6D 6D 79 2D 38 42 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |dummy-8B|
RX | 00 00 00 65 00 07 82 B3 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |...e....|
root@j7200-evm:~# ./spidev_test -v -D /dev/spidev0.0 -p "dummy-8B"
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | 64 75 6D 6D 79 2D 38 42 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |dummy-8B|
RX | 00 00 00 66 00 04 44 F4 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |...f..D.|
root@j7200-evm:~# ./spidev_test -v -D /dev/spidev0.0 -p "dummy-8B"
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | 64 75 6D 6D 79 2D 38 42 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |dummy-8B|
RX | 00 00 00 67 00 00 DB 4A __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |...g...J|
```

在上述测试案例中，MCSPI4 从节点以 64 位格式向 MCU_MCSPI2 主节点发送时间。

4.6 Linux SPI 从器件的挑战

- SPI 从器件需要实时运行：
  - 当主器件启动时钟时，SPI 从器件必须准备好发送/接收数据。但是，主器件无法知道从器件是否准备就绪
  - 从器件必须接收主器件发送的所有任意长度的数据，否则缓冲区溢出，同时导致数据丢失
  - 从器件必须准备好数据，以便在下一个时钟周期内发送到主器件，否则 0 将移位（如果在有效数据流的中间，则会损坏）
- 没有任何类型的流控制。因此，从器件无法在中间停止事务以提供额外的资源（用于接收发送给主器件的 cmd/数据的缓冲区）。由此，如果从器件无法承受主器件的数据量，则可能会出现 RX 欠运转和 TX 数据丢失。

4.7 Linux SPI 从模式一般性限制

- SPI 从控制器一次处理一个请求。不对请求进行批处理
- Linux 从器件无法实时响应
- 无流控制或协议协商支持
- 在 SPI 从（和主）框架中不支持多个未完成的 DMA 请求
- 在主器件和从器件上运行的用户空间栈需要实现并同意某种协议
  - 从器件 RX：从器件需要提前知道要接收多少数据以及何时接收
  - 从器件 TX：在主器件启动时钟之前，从器件需要准备好数据和队列 DMA

内核并不负责满足上述要求，而由协议驱动程序以用户的速度确保满足上述条件。

4.8 McSPI SPI 从模式限制

- McSPI 控制器必须使用 DMA，因为 CPU 写入/读取无法满足在主器件发送下一个时钟之前将数据放入 FIFO 的严格最后期限。
- McSPI 每次传输可以发送或接收 64K-1 字节，因此，SPI 从器件驱动程序需要确保将较大的读取/写入事务分解为若干 64K-1 块，而主器件必须确保在启动时钟之前从器件准备好发送/接收数据。
- 有关限制的更多信息，请参阅本演示文稿：Linux 作为 SPI 从器件/向 Linux 添加 SPI 从器件支持。
- SPI 从实例在没有 DMA 的情况下将失败。因此，添加 DMA RX 和 TX 属性对于此功能来说是必需的。

5 参考文献

- *J7200 DRA821 处理器器件修订版 1.0 技术参考手册*

重要声明和免责声明

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 产品的应用。严禁对这些资源进行其他复制或展示。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