Application Note

SPI Enablement and Validation on TDA4 Family

ABSTRACT

The application report demonstrates the SPI enablement on TDA4 family for both master and slave mode using Linux®. It also provides an example device tree node for one of the SPI instance to demonstrate the master mode. The slave mode mandates DMA support and needs additional patches to enable the PSIL threads required for UDMA functionality. Testing is demonstrated using standard Linux approaches.

Table of Contents

Trademarks

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.
All trademarks are the property of their respective owners.

1 SPI: Serial Peripheral Interface

SPI is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems.

Salient features of SPI protocol are listed below:

- Serial interface
- Synchronous
- Master-slave configuration
- Data Exchange - DMA/PIO

The SPI bus specifies four logic signals:

- SCLK: Serial Clock (output from master)
- MOSI: Master Out Slave In (data output from master)
- MISO: Master In Slave Out (data output from slave)
- CS /SS: Chip/Slave Select (often active low, output from master to indicate that data is being sent)

MOSI on a master connects to MOSI on a slave. MISO on a master connects to MISO on a slave.

Slave Select has the same functionality as chip select and is used instead of an addressing concept.

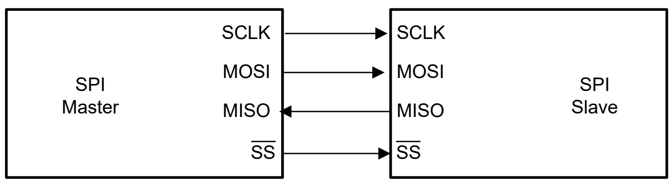

Figure 1-1. Missing Title

2 J7200/J721e MCSPI Support

MCSPI stands for Multichannel Serial Peripheral Interface (MCSPI). The MCSPI module is a multichannel transmit/receive, master/slave synchronous serial bus. There are eleven MCSPI modules in the device (see Table 2-1).

Table 2-1. MCSPI Overviews

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

For more information, see the *J7200 DRA821 Processor Silicon Revision 1.0 Technical Reference Manual*.

missing table

MCSPI4 is directly connected as a slave to MCU_MCSPI2 by default at power-up. MCSPI4 and

MCU_MCSPI2 are not pinned out externally.

2.1 MCSPI Features

The MCSPI modules include the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of MCSPI word lengths, ranging from 4 to 32 bits
- Up to four master channels, or single channel in slave mode
- Single interrupt line for multiple interrupt source events
- Enable the addition of a programmable start-bit for MCSPI transfer per channel (start-bit mode)
- Supports start-bit write command
- Supports start-bit pause and break sequence
- Programmable shift operations (1-32 bits)
- Programmable timing control between chip select and external clock generation
- Built-in FIFO available for a single channel.
- Master multichannel mode:
  – Full duplex/half duplex
  – Transmit-only/receive-only/transmit-and-receive modes
  – Flexible input/output (I/O) port controls per channel
  – Programmable clock granularity
  – MCSPI configuration per channel. This means, clock definition, polarity enabling and word width

3 SPI: Master Mode Enabling and Validation on Linux

3.1 Enable SPI Instances of J721e/TDA4VM

For example lets us take main domain SPI6 instance. To enable SPI6, add the device tree node and the corresponding pinmux node.

```
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>;
+ };
+
```

CONFIG_SPI_OMAP24XX=y is already set in arch/arm64/configs/tisdk_j7-evm_defconfig. The SPI master driver is drivers/spi/spi-omap2-mcspi.c.

3.2 Enable SPIDEV on TD4VM SDK

Add a spidev node inside the spi6 node like below in arch/arm64/boot/dts/ti/k3-j7200-common-proc-board.dts:

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

Enable CONFIG_SPI_SPIDEV=y explicitly in arch/arm64/configs/tisdk_j7-evm_defconfig.

Once you boot Linux one should entries like below:

```
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 Exercise SPI From User Space on TI J7/TDA4x Using Standard Linux spidev_test Tool

The Linux kernel provides spidev_test tool. We need just build & use that. Follow the instructions here:

```
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
```

The above should build spidev_test binary in the tools/spi folder and copy that to rootfs of your target filesystem.

Basic test on TDA4VM Linux command prompt:

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

Output:

```
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 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
__ |..........|
```

Since there is no Slave the RX we always see 0xFF.

4 SPI: Slave Mode Enabling and Validation on Linux

To test slave mode, a master is also needed. Advantage of the hardware capability of J7200 is taken to demonstrate SPI slave support.

For more information, see the MCSPI Overviews chapter in the *J7200 DRA821 Processor Silicon Revision 1.0 Technical Reference Manual*.

MCSPI4 is directly connected as a slave to MCU_MCSPI2 by default at power-up. MCSPI4 and MCU_MCSPI2 are not pinned out externally.

4.1 Enable SPI Instances of J7200

Add the MCU_MCSPI2 node that is the SPI master and MAIN_MCPSI4 node, which is the SPI slave node.

```
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 Enable DMA for MCSPI4 Slave Node

By default the SDK 8.1 does not have the UDMA PSIL threads needed for MCSPI instances. Add them and add the DMA properties needed for SPI slave functionality.

```
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 Enable SPIDEV and SPI_SLAVE Configs

Two configs need to be enabled in arch/arm64/configs/tisdk_j7200-evm_defconfig:

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

4.4 Test SPI Slave Functionality From User Space on TI J7200 Using Standard Linux spidev_test Tool

Make use of the spidev_test compiled in the above step 3.3 to demonstrate slave mode functionality as well.

Initiate communication between MCPSI4 Slave & MCU_MCSPI2 using the spidev_test utility.

Logs of the above test:

```
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|
```

The master MCU_MCSPI2 transmitted TX message master-hello-to-slave and received the RX message slave-hello-to-master from the MCPSI4 slave can be seen and vice versa.

4.5 SPI Slave Testing Using spi-slave-time

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

Logs:

```
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|
```

In the above test case, the MCSPI4 slave node sends the time in 64 bits format to MCU_MCSPI2 master node.

4.6 Linux SPI Slave Challenges

- SPI Slave needs to be Real Time:
  - SPI slave has to be ready to send/receive data when master starts clock. But, Master has no way of knowing if slave is ready
  - Slave has to receive all arbitrary length data that master sends Else, buffer overflows leading to data loss
  - Slave has to be ready with data to send to master in next clock cycle Else, 0s are shifted (corruption if in the middle of valid data flow)
- No flow control of any sort whatsoever. Therefore, no way to stop transaction in the middle for Slave to provision additional resources (buffers to receive cmd/data to be sent to master) Therefore there may be RX underrun and TX data loss, if master overwhelms slave.

4.7 Linux SPI Slave Mode General Limitations

- SPI slave controller processes one request at a time. No batch processing of requests
- Linux slave cannot respond in real time
- No flow control or protocol negotiation support
- No support for multiple outstanding DMA request in SPI slave(and master) framework
- Userspace stack running on master and slave needs to implement and agree on certain protocol
  - Slave RX: Slave needs to know in advance how much data to receive and when
  - Slave TX: Slave needs to be ready with data and queue DMA before Master starts clock

Kernel does not take care of meeting above requirements, its up to protocol drivers at users pace to make sure above conditions are met.

4.8 McSPI SPI Slave Mode Limitations

- McSPI controller has to use DMA as CPU writes/reads cannot meet the hard deadline of putting data into FIFO before master sends next clock.
- McSPI can transmit or receive 64K-1 bytes per transfer, therefore, SPI slave driver needs to make sure to break larger read/write transactions into 64K-1 chunks and master must make sure slave is ready to send/receive data before starting the clock.
- More information on limitations can be found in this presentation: Linux as an SPI Slave/Adding SPI slave support to Linux.
- SPI slave instance fails without DMA. So adding DMA RX and TX properties is compulsory for the functionality.

5 References

- *J7200 DRA821 Processor Silicon Revision 1.0 Technical Reference Manual*