AN13633

Using Enhanced Configurable SPI (ECSPI) as Slave in Linux for i.MX 8M Series

Rev. 2 — 13 September 2023 Application note

Document Information

Information	Content
Keywords	i.MX 8M; ECSPI; SPI
Abstract	This application note describes ECSPI as slave in Linux and provides some optimizations and test results.

1 Introduction

The i.MX 8M family of applications processors based on Arm Cortex-A53 and Cortex-M cores provide industry-leading audio, voice, and video processing for applications. It scales from consumer home audio to industrial building automation and mobile computers.

Enhanced Configurable SPI (ECSPI) is an SPI IP module that is widely used in i.MX series SoCs, for example, i.MX 6, i.MX 7, and i.MX 8M.

More i.MX users are intended to use ECSPI as slave in Linux to receive and transmit data. This application note guides them to deploy applications under Linux with acceptable speed and error rate on NXP i.MX 8M series SoCs.

This application note describes ECSPI as slave in Linux and provides some optimizations and test results.

However, as Linux is not a real-time operating system, it is not a good idea to use Linux to act as SPI slave. But surely it is the easiest way, for example, all drivers and test applications are ready there.

The target audiences of the document are those who:

1. Want to use ECSPI as slave in Linux.
2. Want to know the maximum transfer speed of using ECSPI as slave in Linux.
3. Get troubled by transmission speed of ECSPI as slave in Linux.
4. Get troubled by the error rate of ECSPI as slave in Linux.
5. Want to get familiar with ECSPI on i.MX 8M.
6. Want to know more about ECSPI.

Note: *The data presented in this application note is based on empirical measurements taken in a small sample size. The presented results are not guaranteed.*

2 Definitions, acronyms, and abbreviations

Table 1. Definitions, acronyms, and abbreviations

Acronyms	Meanings
ECSPI	Enhanced Configurable SPI
DMA	Direct Memory Access
SDMA	Smart Direct Memory Access Controller
PIO	Programming Input/Output Model
DTS	Device Tree Source
DTB	Device Tree Blob
RTOS	Real-Time Operating System
SoC	System on Chip
MISO	Master In Slave Out
MOSI	Master Out Slave In
SCLK	Serial Clock
CS/SS	Chip Select

3 Overview of i.MX 8M ECSPI

The ECSPI is a full-duplex, synchronous, four-wire serial communication block.

Key features of the ECSPI include:

- Full-duplex synchronous serial interface.
- Master/Slave configurable.
- One Chip Select (SS) signal.
- Transfer continuation function allows unlimited length data transfers.
- 32-bit wide by 64-entry FIFO for both transmit and receive data.
- Polarity and phase of the SS and SPI Clock (SCLK) are configurable.
- Direct Memory Access (DMA) support.

Figure 1 shows the block diagram of ECSPI.

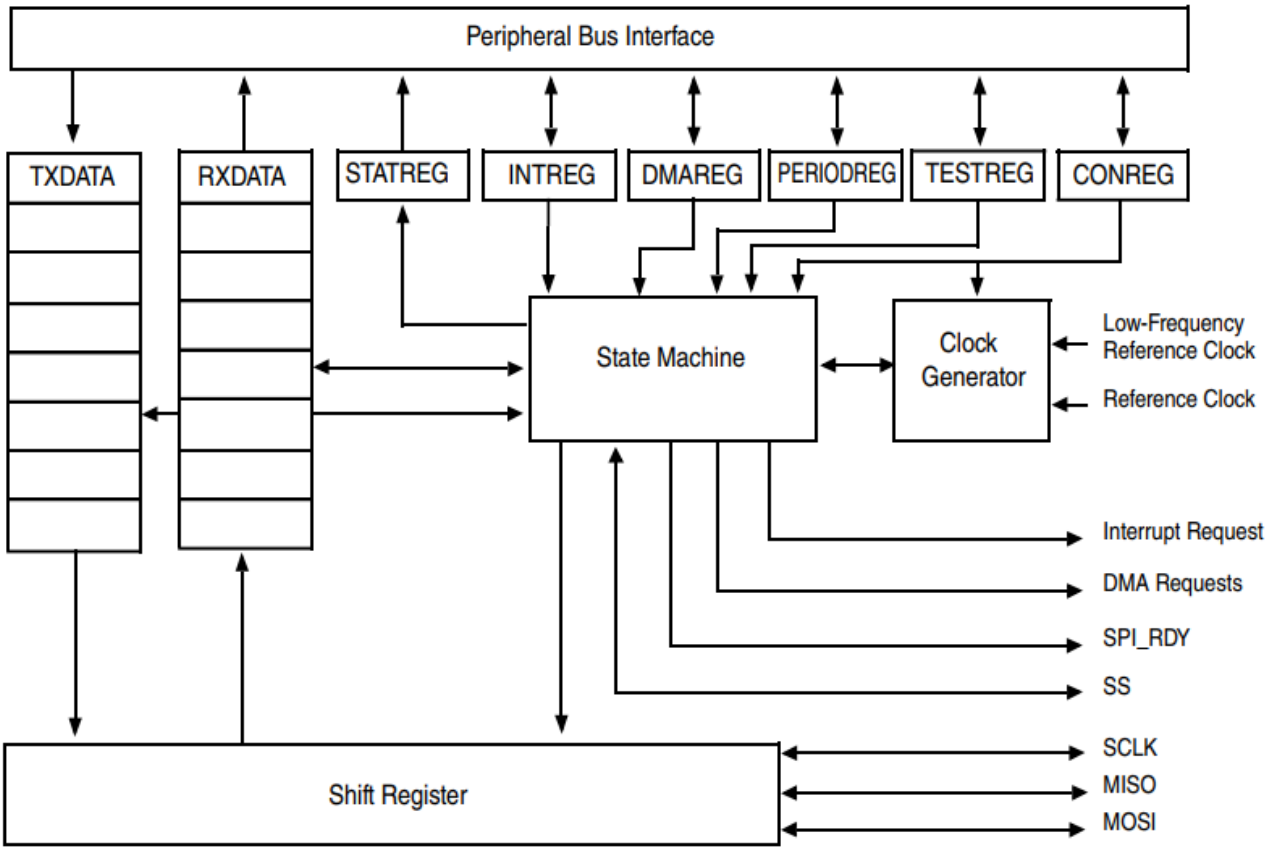

Figure 1. Block diagram of ECSPI

3.1 ECSPI driver in Linux

In Linux, the ECSPI driver is at *drivers/spi/spi-imx.c*. The ECSPI DTS node can configure the driver.

The ECSPI driver in Linux supports:

- Support ECSPI on i.MX 2, i.MX 3, i.MX 6, i.MX 7, and i.MX 8M.
- Configurable Master/Slave mode.

- Configurable DMA/PIO mode in Master mode.
- PIO mode only in Slave mode.
- Full-duplex data transfer with synchronous serial interface in Master/Slave mode.
- Configurable SS control mode. SS pin can be controlled as a GPIO by software or by ECSPI itself (master only).
- Configurable number of SS pins (master only).

3.2 ECSPI limitations as slave

The ECSPI has two slave-related hardware limitations, TXFIFO and burst size.

3.2.1 TXFIFO issue in slave mode

The description of the limitation is that:

When working in Slave mode, after 64 words are written to TX FIFO, even TXFIFO becomes empty, `ECSPI_TXDATA` keeps shift out the last word data.

Generally, a TXFIFO issue is that TXDATA outputs invalid data when TXFIFO is empty.

The workaround in ECSPI driver is that the ECSPI controller is disabled after every data transfer.

Code:

In function `spi_imx_pio_transfer_slave()` in *spi-imx.c*:

```
static int spi_imx_pio_transfer_slave(struct spi_device *spi,
                      struct spi_transfer *transfer)
{
    … …
    /* ecspi has a HW issue when works in Slave mode,
     * after 64 words writtern to TXFIFO, even TXFIFO becomes empty,
     * ECSPI_TXDATA keeps shift out the last word data,
     * so we have to disable ECSPI when in slave mode after the
     * transfer completes
     */
    if (spi_imx->devtype_data->disable)
        spi_imx->devtype_data->disable(spi_imx);
    … …
}
```

In driver and user space, this limitation has the following effects:

- When acting as slave, the TXFIFO must not be empty.
- As the controller is disabled and then enabled, to ensure that the slave is ready for receiving data, an interval is needed in master side between two transfers.

3.2.2 Set SPI burst size to transfer size in slave mode

In *i.MX 8M Dual/8M QuadLite/8M Quad Applications Processors Reference Manual* (document IMX8MDQLQRM), `ECSPI_CONFIGREG[SS_CTL]` has the following description:

11–8 SS_CTL	SPI SS Wave Form Select. In master mode, this field controls the output wave form of the Chip Select (SS) signal when the SMC (Start Mode Control) bit is cleared. The SS_CTL bits are ignored if the SMC bit is set. SS CTL[3] is reserved. SS CTL[2] is reserved. SS CTL[1] is reserved. SS CTL[0] is for SPI channel 0. In slave mode, this bit controls when the SPI burst is completed. An SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data. 0 In master mode - only one SPI burst will be transmitted. 1 In master mode - Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty. 0 In slave mode - an SPI burst is completed when the number of bits received in the shift register is equal to (BURST LENGTH + 1). Only the n least-significant bits (n = BURST LENGTH[4:0] + 1) of the first received word are valid. All bits subsequent to the first received word in RXFIFO are valid. 1 Reserved

From the description, when acting as slave, the burst size must be set exactly to the size of the transfer. As the maximum burst length is 2^12 bits, it limits SPI transaction size to maximum 2^12.

In code, we have the following macro in *spi-imx.c*:

```
/* The maximum bytes that IMX53_ECSPI can transfer in slave mode.*/
#define MX53_MAX_TRANSFER_BYTES 512
static int spi_imx_pio_transfer_slave(struct spi_device *spi,
                      struct spi_transfer *transfer)
{
    … …
    if ((is_imx51_ecspi(spi_imx) || is_imx53_ecspi(spi_imx)) &&
        transfer->len > MX53_MAX_TRANSFER_BYTES) {
        dev_err(&spi->dev, "Transaction too big, max size is %d bytes\n",
            MX53_MAX_TRANSFER_BYTES);
        return -EMSGSIZE;
    }
    … …
}
```

This limitation has the following effects on driver:

1. In slave mode, DMA can't be used. Instead, only PIO mode can be used. It is based on the following considerations:
   a. To use DMA mode, the burst size must be 4 byte aligned.
   b. The ECSPI driver must support unaligned transfer size.
2. The burst size must be set to transfer size in PIO mode. But maximum burst size is 2^12 = 512 bytes. So ECSPI driver can transfer maximum 512 bytes in a PIO transfer.

Note: *From testing, it seems no such transfer length restriction must be added in DMA mode. So only PIO mode has this restriction.*

3.2.3 External master restrictions

As a result of hardware limitations in ECSPI, there come some restrictions on master.

3.2.3.1 Insert intervals between data transfers at master side

As described in Section 3.2.1, an interval is needed at master side between two transfers to ensure that the slave is ready for receiving data.

3.2.3.2 SS signal control methods

Generally, there are two methods on SS signal control from master:

1. SS signal kept active during whole data transfer.

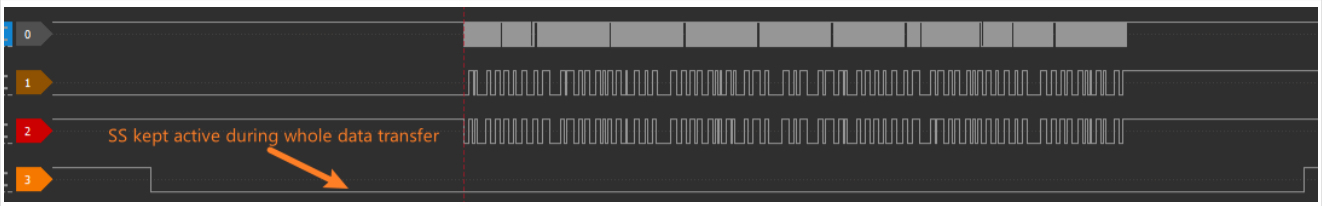

Figure 2. SS kept active waveform

2. SS signal gets deasserted between transfer words.

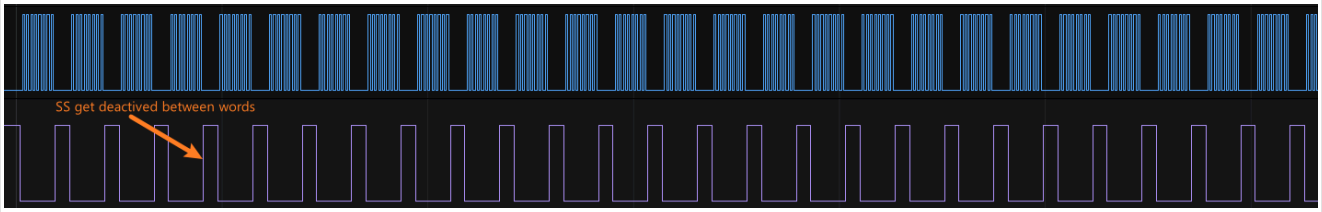

Figure 3. SS gets deasserted between 8-bit data words

We fully support method 1 and partly support method 2 (only 32-bit).

Note: *In ECSPI driver, The SS control method is controlled by defining* `cs-gpios` *in* `dts` *node.*

```
&ecspi2 {
  … …
  cs-gpios = <&gpio5 13 GPIO_ACTIVE_LOW>;
  … …
};
```

When `cs-gpios` property is defined, it means that the SS pin is muxed as a GPIO port, and the driver code controls its signal level. In this way, SS signal is kept active during data transfer. This solution is default provided in our BSP.

Otherwise, if `cs-gpios` property is not defined, SS signal gets deasserted between words.

4 ECSPI slave in default release

4.1 DTS changes to support ECSPI slave

Before making DTS changes to support ECSPI slave, make sure that the slave DTS does not exist.

Some i.MX 8M platforms provide ECSPI slave DTS file in release, for example, `imx8mn-evk-ecspi-slave.dts` or `imx8mm-evk-ecspi-slave.dts`.

4.1.1 DTS node for ECSPI slave

To add an ECSPI slave node, see below as reference:

imx8mm-evk-ecspi-slave.dts:

```
#include "imx8mm-evk.dts"

/delete-node/&spidev0;

&ecspi2 {
        #address-cells = <0>;
        /delete-property/cs-gpios;
        spi-slave;
};

&pinctrl_ecspi2_cs {
        fsl,pins = <
                MX8MM_IOMUXC_ECSPI2_SS0_ECSPI2_SS0              0x82
        >;
};
```

Note:

Beside the DTS change, to see `/dev/spidevX.X` device, user must perform:

echo spidev > /sys/class/spi_slave/spi1/slave (before 5.15 kernel) or echo dh2228fv > /sys/class/spi_slave/spi1/slave (after 5.15 kernel).

Note:

When acting as slave, the SS pin can't be configured as GPIO and can only be configured as ECSPI SS pin.

4.1.2 Performance of ECSPI slave in default release

For the test results of slave performance, see Section 6.2.

The performance of slave is very poor.

In default BSP release, the recommended SCLK for SPI slave mode is <= 1 M and the transfer interval in master is 5 ms.

5 How to improve ECSPI slave performance

This chapter describes methods to improve ECSPI slave performance.

5.1 Increase transfer speed

An obvious way to improve performance is increasing transfer speed, such as, increasing transfer clock (SCLK).

But when SCL is increased, issues come out.

5.1.1 Issues in high transfer speed

Generally, we can see two issues in high transfer speed.

- Bit shift.
- Data lost.

5.1.1.1 Bit shift

Bit shift means that bits within 1 byte are shifted.

For example:

```
Master Send:   FF FF FF 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
Slave Receive: 04 0A 18 38 80 21 06 0E 78 01 4C A0 51 C2 E3 03
```

In the example, first three `0xFF` are missed, `0x05` left shifts 1 bit to `0x0A`, `0x06` left shifts 1 bit to `0x38`, and so on.

5.1.1.2 Data lost by byte

Data lost means that the data bytes sent by master are lost.

For example:

```
Master Send:   FF FF FF 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
Slave Receive: 04 0A 18 38 80 21 06 0E 78 01 4C A0 51 C2 E3 03
```

In the example, first three `0xFF` are lost.

5.1.2 Solutions for issues in high transfer speed

5.1.2.1 Use good and short cables

When using higher SCLK, good and short cables are required.

Long cables bring in interferences on CLK and MOSI when SCLK increases and result in bit-shift issue.

We recommend using cables within 8 cm, as short as possible, with good quality.

5.1.2.2 Use DMA instead of PIO

DMA control provides another method to utilize the FIFOs in the ECSPI. By using DMA request and acknowledge signals, larger amounts of data can be transferred and reduce interrupts and host processor loading.

In PIO mode, data is shifted out of shift register bit by bit. But in DMA mode, data is shifted out word by word. So, using DMA mode largely improves performance and avoids bit shift issue.

For details, see Section 5.2.

5.1.2.3 Increase ECSPI root clock

Theoretically, the sampling frequency must be greater than twice the bandwidth of the signal being sampled.

When trying to increase SCLK, pay attention to ECSPI working clock and make it greater than twice of SCLK.

And vice versa, increasing ECSPI root clock can have benefits on sampled data.

5.2 Use DMA instead of PIO

As a result of ECSPI hardware limitations (see Section 3.2.2), the kernel driver of ECSPI slave does not use DMA.

To ensure data stable and higher SCLK, enable DMA in slave mode.

The ECSPI slave DMA patch is pushed in latest i.MX releases. If it is not, find the patch in AN13633SW.

5.2.1 Enable DMA in ECSPI slave

In current i.MX release, the DMA node is added in *<soc>.dtsi*.

For example, in *arch/arm64/boot/dts/freescale/imx8mm.dtsi*:

```
ecspi1: spi@30820000 {
                compatible = "fsl,imx8mm-ecspi", "fsl,imx51-ecspi";
                #address-cells = <1>;
                #size-cells = <0>;
                reg = <0x30820000 0x10000>;
                interrupts = <GIC_SPI 31 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clk IMX8MM_CLK_ECSPI1_ROOT>,
                     <&clk IMX8MM_CLK_ECSPI1_ROOT>;
                clock-names = "ipg", "per";
                dmas = <&sdma1 0 7 1>, <&sdma1 1 7 2>;
                dma-names = "rx", "tx";
                status = "disabled";
            };
```

So, users must add a slave node. See Section 4.1.

5.2.2 ECSPI Slave DMA patch limitations

As described in Section 3.2.2, the result that DMA cannot be used in ECSPI slave mode is that the driver must support unaligned transfer size, **this DMA patch only support 4 bytes aligned transfer size**.

For unaligned transfer size, PIO is still used.

Take care of the transfer size and make it 4 bytes aligned.

5.2.3 Waveform when using DMA

In ECSPI, there is a CLK IDLE interval between the transfer of 2 bytes.

In PIO mode, MISO is kept LOW when CLK is in IDLE. When DMA is enabled in transmission, the MISO might stay in HIGH when CLK is in IDLE.

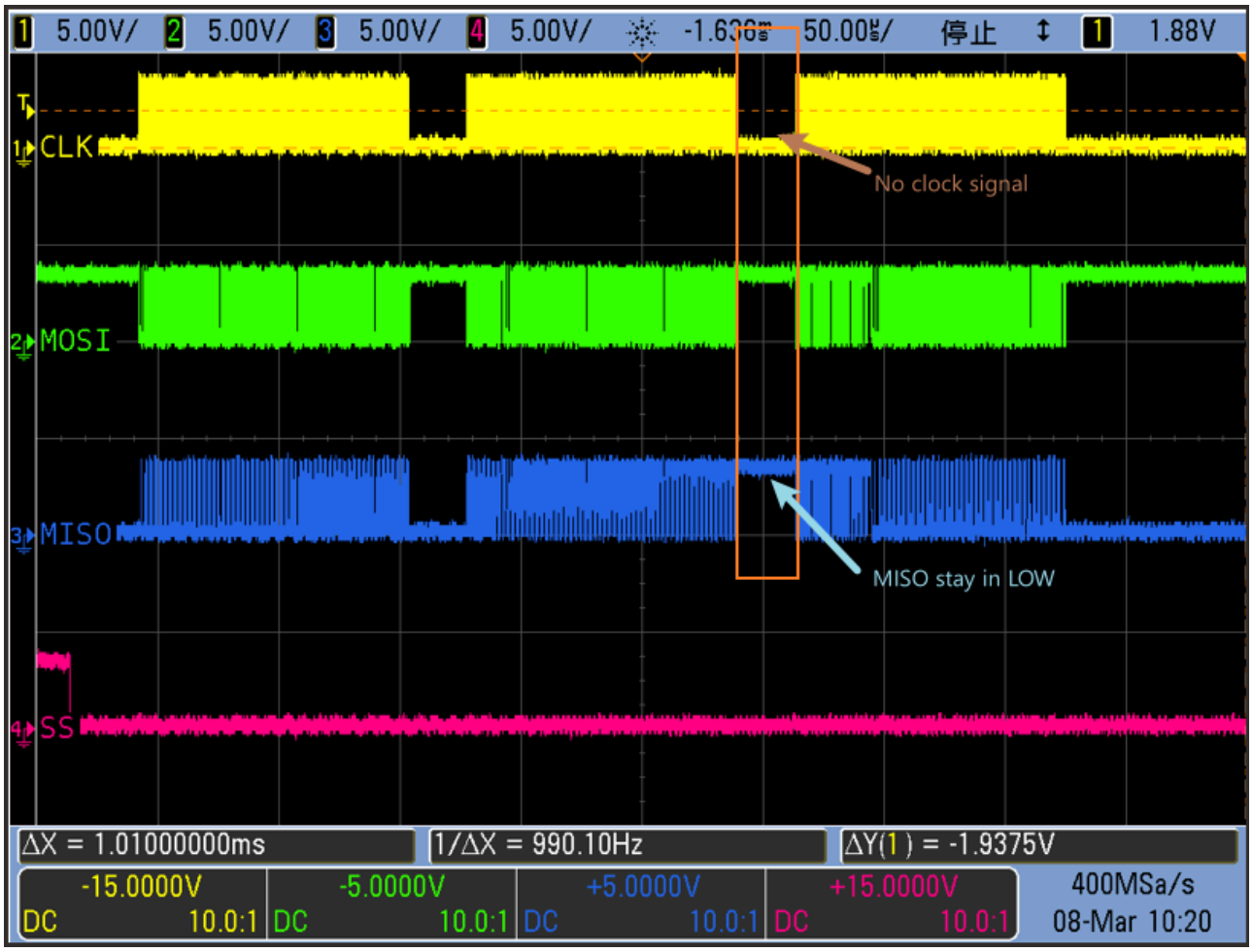

Figure 4. MISO

From description of ECSPI in data sheet, when SS is active, there is no requirement on MISO signal level.

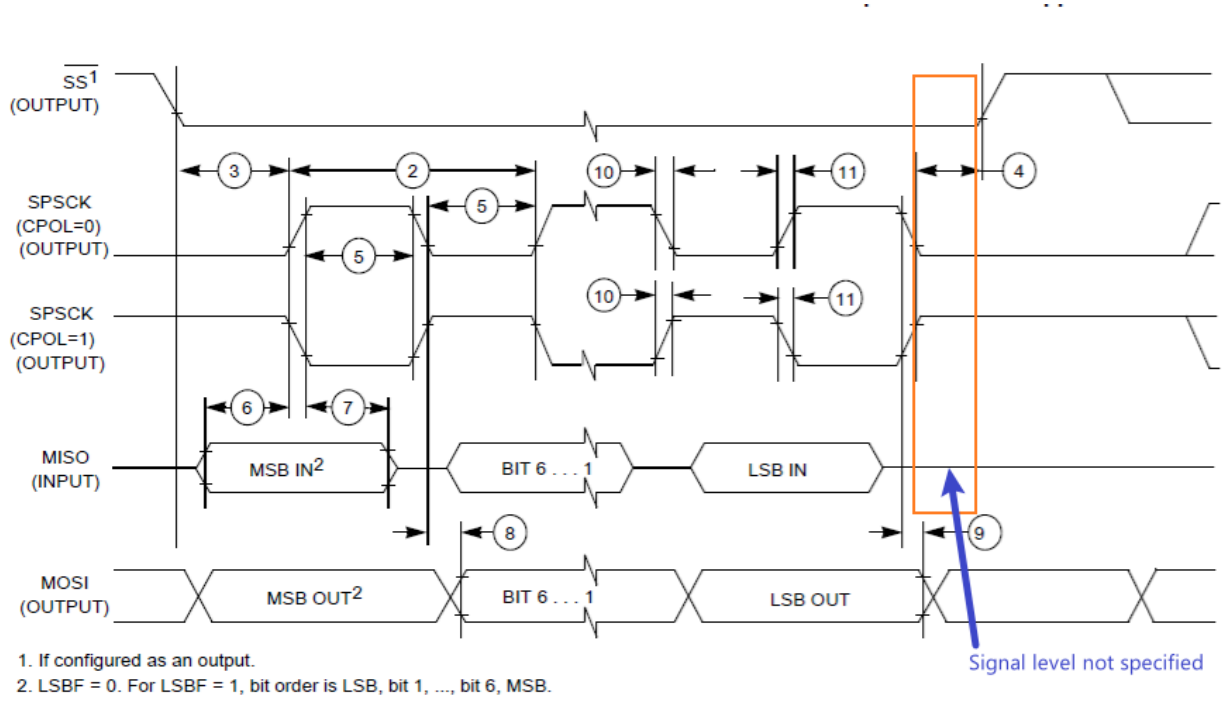

Figure 5. SPI master mode timing (CPHA = 0)

So this issue does not exist here. ECSPI is able to guarantee that the data is sampled correctly.

5.2.4 Byte order for DMA copy

Note:

From i.MX 8M Dual/8M QuadLite/8M Quad Applications Processors Reference Manual (document IMX8MDQLQRM):

The ECSPI does not support byte reordering in hardware.

As ECSPI does not support byte reordering in hardware, when using DMA, driver code reorders bytes when data width is 8 or 16.

See function `spi_imx_dma_transfer_convert_8` and `spi_imx_dma_transfer_convert_16` in `spi-imx.c`.

So 32-bit data width, as it does not need any byte reordering, has a better performance.

5.2.5 Choose appropriate transfer size for DMA copy

When DMA is enabled in slave, the total transfer size can affect the speed.

The reason is that:

1. As DMA can only move data word by word. The data length copied by DMA each time must be 4 bytes aligned, such as, 4, 8, 12, 128. The more data can be copied, the better performance it is.
2. As a result of 1, the total transfer size should be taken care of. If it is (4 * 32 = 128) bytes aligned, DMA copy can reach best performance. For example, the total transfer size is 512. Or, if it is only 4 bytes aligned, DMA can only copy 4 bytes each time. The DMA performance is the worst. For example, total transfer is 73 * 4 = 292.

This logic is inside `spi-imx.c` driver code, function `spi_imx_dma_transfer()`:

```
static int spi_imx_dma_transfer(struct spi_imx_data *spi_imx,
                struct spi_transfer *transfer)
{
    … …
    /* Get the right burst length from the last sg to ensure no tail data */
    bytes_per_word = spi_imx_bytes_per_word(spi_imx->bits_per_word);
    for (i = spi_imx->devtype_data->fifo_size / 2; i > 0; i--) {
        if (!(sg_dma_len(last_sg) % (i * bytes_per_word)))
            break;
    }
    … …
}
```

In the code snippet, `sg_dma_len(last_sg)` is total transfer size, `bytes_per_word` is 4, and `i` presents number of words that DMA can copy each time.

In our tests, to ensure 8 MHz SCLK, make sure that `i` is larger than 2. It means that DMA copies at least 8 bytes each time.

Generally, the DMA mode is the most recommended way to improve the performance.

5.3 Reduce interval between two master transfer

From description of Section 3.1, when acting as slave, the ECSPI controller must be disabled and re-enabled between two transfers.

The recommendation interval from RD is 5 ms, which is too long.

So how the master TX interval is comprised and how we reduce this interval.

5.3.1 How the master TX interval is comprised

The interval makes sure that slave is ready to receive data.

Generally, it is composed of two durations:

1. The duration of resetting ECSPI. Disabling ECSPI => enabling ECSPI => Ready to receive data in slave device.
2. The duration of sending data. Data copied from user space => kernel space => sent by driver

5.3.2 How to estimate the minimum interval

There are two ways to estimate the interval:

- Use test program to do stress tests on an interval and check the error rate.
- Force master to send data continuously and use an oscilloscope to observe the duration of sending data. Then Add the duration with 100 µs as minimum interval.

In our testing, ~1 ms interval is recommended. From the feedback of customer, 800 µs is also OK.

5.4 Bind ECSPI interrupt to another core

ECSPI driver uses interrupt to TX/RX data. By default, kernel binds all interrupts to core0. For high loading context, if the ECSPI interrupts are not processed in time, it causes data loss.

In this situation, bind ECSPI interrupt to other cores.

Steps:

1. Check interrupts and find ECSPI interrupt number. cat /proc/interrupts | grep -e “spi”

```
cat /proc/interrupts | grep -e "spi"
                    CPU0       CPU1       CPU2       CPU3
                    34:          0          0          0          0     GICv3
  64 Level     30830000.spi
                    46:          6          0          0          0     GICv3
 139 Level     30bb0000.spi
```

   Here the IRQ numbers of SPI are 34 and 46.
2. Bind ECSPI interrupt to other core. See https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-cpu-irq

Note:

This method is optional. It is mainly for high loading system with high frequency interrupts. It depends on the context in your system.

In our test, this method did not show any improvement.

5.5 Apply PREEMPY-RT patches

As Linux is not a real-time OS, the real-time performance can’t be guaranteed.

For SPI slave, the real-time performance is a key in data transfer. So we can consider applying PREEMPY-RT patches, which intend to make Linux a real-time OS.

But in our test, real-time Linux does not show any improvement here.

For details about PREEMPY-RT patches, see https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/preemptrt_setup.

6 Performance test result

This chapter lists the test results.

6.1 Test application

In the test, we used two applications.

1. Default `spidev_test`.
   The default `spidev_test` program is provided in kernel, at *tools/spi/*.
2. Modified `spidev_test`.
   Besides 1, we provide another `spidev_test` in AN13633SW.
   This `spidev_test` program provides:
   a. To perform the stress test, add the `-t` parameter.
   b. To add an interval in us between transfers (master only), add the `-V` parameter.
   c. To perform the loop transferring on data (master and slave), use the `-I` parameter.
   d. To define the amount of data that is sent/received each time (master and slave, used to test transfer size), add the `-A` parameter.
   e. Do validation on received data.

Example:

- Master sends: *./spidev_test -D /dev/spidev1.0 -b 16 -s 8000000 -t -V 1000 -I 100000 -A 336*

- Slave receives: *./spidev_test -D /dev/spidev1.0 -b 16 -t -I 100000 -A 336*

Note: *Master and slave must use the same* `-t`, `-b`, `-I`, *and* `-A` *parameters.*

6.2 Slave PIO performance

Table 2. Slave PIO performance

SPI CLK (Hz)	Error rate @ different interval time		
	1500 µs	800 µs	400 µs
500 K	0.008 %	1.744 %	Cannot run
1 M	0.000 %	5.984 %	Cannot run
4 M	10.124 %	1.317 %	0.166 %
8 M	7.510 %	0.431 %	Cannot run

6.3 Slave DMA performance

Table 3. Slave DMA performance

SPI CLK (Hz)	Error rate @ different interval time				
	1500 µs	1000 µs	800 µs	400 µs	200 µs
8 M	0.0 %	0.001 %	0.001 %	2 %	42.016 %

Note:

1. *From the feedback of customer, the result of 10 MHz SCLK is similar to 8 MHz. Higher frequency (> 10 MHz) is not recommended.*
2. *Most tests are on 8 MHz SCLK. Frequencies under 8 MHz have similar results.*

7 Receive more data to improve performance

As described above, the speed and stability can be improved a lot on ECSPI slave.

But what else, is there any other way to improve the performance?

Yes, the answer is yes. But it also depends on your requirements.

7.1 Check RX requirements in slave

As discussed in Section 3.2.1, there is a limitation in TXFIFO when acting as slave. So we must disable ECSPI controller after every transfer.

Considering the following conditions:

1. Re-enabling ECSPI controller takes some time.
2. In kernel, data copying from/to user space to/from kernel driver also takes some time.
3. In our testing, 1+2 takes a ~1 ms interval between each data transfer.
4. The maximum transfer length in a transfer is maximum burst length, which is 512 bytes. It means when transferring large amount of data, to avoid TXFIFO issue, we can only transfer maximum 512 bytes each time.

If ignoring the TXFIFO issue, we can transfer more data (> 512 bytes) each time. When this way is accepted, we can also reduce transfer interval, and therefore improve the performance.

In this way, slave works in a half-duplex way.

7.2 Receive more data (> 512 bytes) each time

When receiving more data (> 512 bytes) each time, TX from slave contains unnecessary data when TXFIFO is empty. See Figure 6 .

Figure 6. Receiving more data

But master should be able to transfer more data now.

In this case, we recommend performing TX/RX separately, which seems like half-duplex.

Note:

In `spidev` driver code, `drivers/spi/spidev.c`, the maximum data size that driver can hold is 4096 bytes. It is hardcoded in code:

```
static unsigned bufsiz = 4096;
```

To receive more data from master at one time, enlarge this `const` value.

7.3 Test result on transfer size

Table 4. Test result

8 M SCLK, Master transfers at 1 ms interval		
Bytes sent from master	**Result**	**Comments**
<= 5000 Bytes	PASS	Tested 1800 times, all passed.
>5000 Bytes and < 8000 Bytes	FAIL	Not failed each time.

Table 4. Test result*...continued*

8 M SCLK, Master transfers at 1 ms interval		
Bytes sent from master	**Result**	**Comments**
>= 8000 Bytes	FAIL	Failed each time.

8 Software package description

A software package is attached with this application note. Table 5 describes the files in the package.

Table 5. Software package

Filename	Description	Comments
0001-Add-dma-support-to-ecspi-slave-for-5.4.70-kernel.patch	ECSPI spi-imx slave DMA patch.	Based on kernel version: imx_5.4.70_2.3.0
spi-imx.c	ECSPI driver file with DMA slave code.	Based on kernel version: imx_5.4.70_2.3.0
spidev_test.c	Modified spidev_test.c.	Based on kernel version: imx_5.4.70_2.3.0
imx8mn-evk-ecspi-slave.dts	ECSPI slave dts file.	Based on kernel version: imx_5.4.70_2.3.0

9 Reference

- *i.MX 8M Dual/8M QuadLite/8M Quad Applications Processors Reference Manual* (document IMX8MDQLQRM)

10 Note About the Source Code in the Document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

11 Revision history

Table 6 summarizes the revisions to this document.

Table 6. Revision history

Revision number	Release date	Description
2	13 September 2023	Updated Section 4.1.1
1	22 May 2023	Updated Section 4.1.1
0	07 May 2022	Initial public release

12 Legal information

12.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

12.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

12.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 13 September 2023
Document identifier: AN13633