

1 Introduction

This application note describes how to use the FlexIO module to emulate Serial Synchronization Interface (SSI). SSI is a widely used serial interface between absolute position sensors and controllers. Currently, the i.MX RT1010 does not directly support SSI peripherals, but a good solution is to use FlexIO to emulate SSI to achieve performance similar to that of dedicated SSI peripherals.

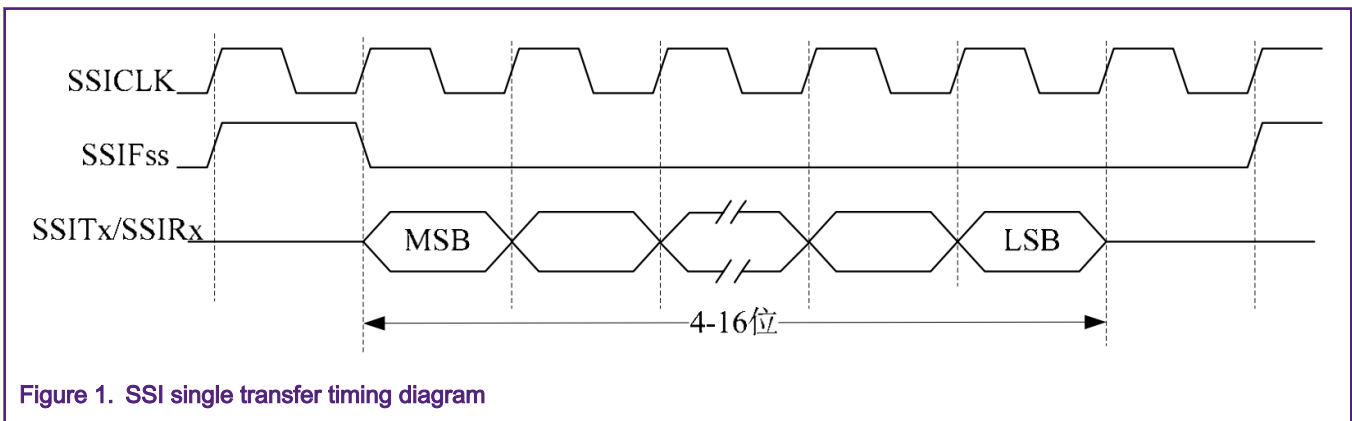
The i.MX RT1010 processor is based on the Arm® Cortex®-M7 platform. It provides high CPU performance and best real-time response, and contains rich peripheral devices. In order to verify the SSI peripherals emulated via FlexIO, a simple application was implemented on the RT1010 EVK boards.

Contents

- 1 Introduction..... 1
- 2 SSI overview..... 1
- 3 Development platform..... 1
- 4 SSI emulation..... 3
- 5 Conclusion..... 8
- 6 References..... 8

2 SSI overview

In this case, the SSI uses Texas Instruments synchronous serial frame format. Figure 1 shows the Texas Instruments synchronous serial frame format for one transmission. SSI and off-chip slave devices drive their respective output data on the rising edge of SSICLK, and latch data from another device on the falling edge.



3 Development platform

In order to emulate the communication between a master (such as a controller) and a slave (such as a sensor) of an SSI, two i.MXRT1010 EVK boards are used in this demo. One board is used to emulate the SSI master device, and the other board is used to emulate the SSI slave device.

Figure 2 describes the connection between SSI master board and SSI slave board.



Pin Name	Master Board		Pin Name	Slave Board
SSI_RX	J26-4	<----->	SSI_TX	J26-6
SSI_TX	J26-6	<----->	SSI_RX	J26-4
SSI_CLK	J26-8	<----->	SSI_CLK	J26-8
SSI_Fss	J56-10	<----->	SSI_Fss	J56-10
GND	J1-10	<----->	GND	J1-10

Figure 2. Connection between SSI master board and slave board

On the i.MX RT1010, FLEXIO has a total of 27 pins. In this case, four FlexIO pins are used to emulate SSI_Fss, SSI_RX, SSI_TX and SSI_CLK. Table 1 lists the FlexIO pins used in this case to emulate SSI.

Table 1. FlexIO Pins used

FlexIO pin	Pin location
flexio1.FLEXIO21	J26-4
flexio1.FLEXIO22	J26-6
flexio1.FLEXIO26	J26-8
flexio1.FLEXIO00	J56-10

Figure 3 shows the actual hardware platform. To make demo work, perform the following steps:

- Remove R792, and solder R800 with 0 Ω resistor on RT1010 EVK board.
- Change the ISP switch (SW8) to (0b0010).
- Connect Pin1 and Pin2 of J1 with shorting cap.
- Power on board with USB cable plugged to J41.



Figure 3. Hardware platform

4 SSI emulation

This chapter introduces how to use the FlexIO module to emulate the SSI single transfer timing diagram, as shown in [Figure 1](#). In addition, the configurations of SSI master mode and slave mode will be described in details.

4.1 FlexIO overview

The FlexIO is a highly configurable module supporting a wide range of protocols including, but not limited to UART, I²C, SPI, I²S. It provides a wide range of functionality, such as:

- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions.
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes.
- Interrupt, DMA or polled transmit/receive operation.
- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support.
- Array of 32-bit shift registers with transmit, receive and data match modes, double buffered shifter operation for continuous data transfer.

The FlexIO is a very flexible module. For a fixed timing emulation, its configuration method is not unique. The same effect can be achieved through multiple combinations of an unlimited number of Timers and Shifters and different configurations. This application note introduces one configuration method for SSI master and SSI slave respectively.

4.2 SSI master configuration

A total of two Timers, two Shifters are used to emulate the SSI master. Timer 0 is used to generate SSI_CLK signal. Timer 1 is used to generate SSI_Fss signal. Shifter 0 connects to SSI_TX pin and transmits the data on each rising edge of SSI_Clk. Shifter 2 connects to SSI_RX pin and receives the data on each falling edge of SSI_Clk. Figure 4 shows the FlexIO SSI master configuration diagram.

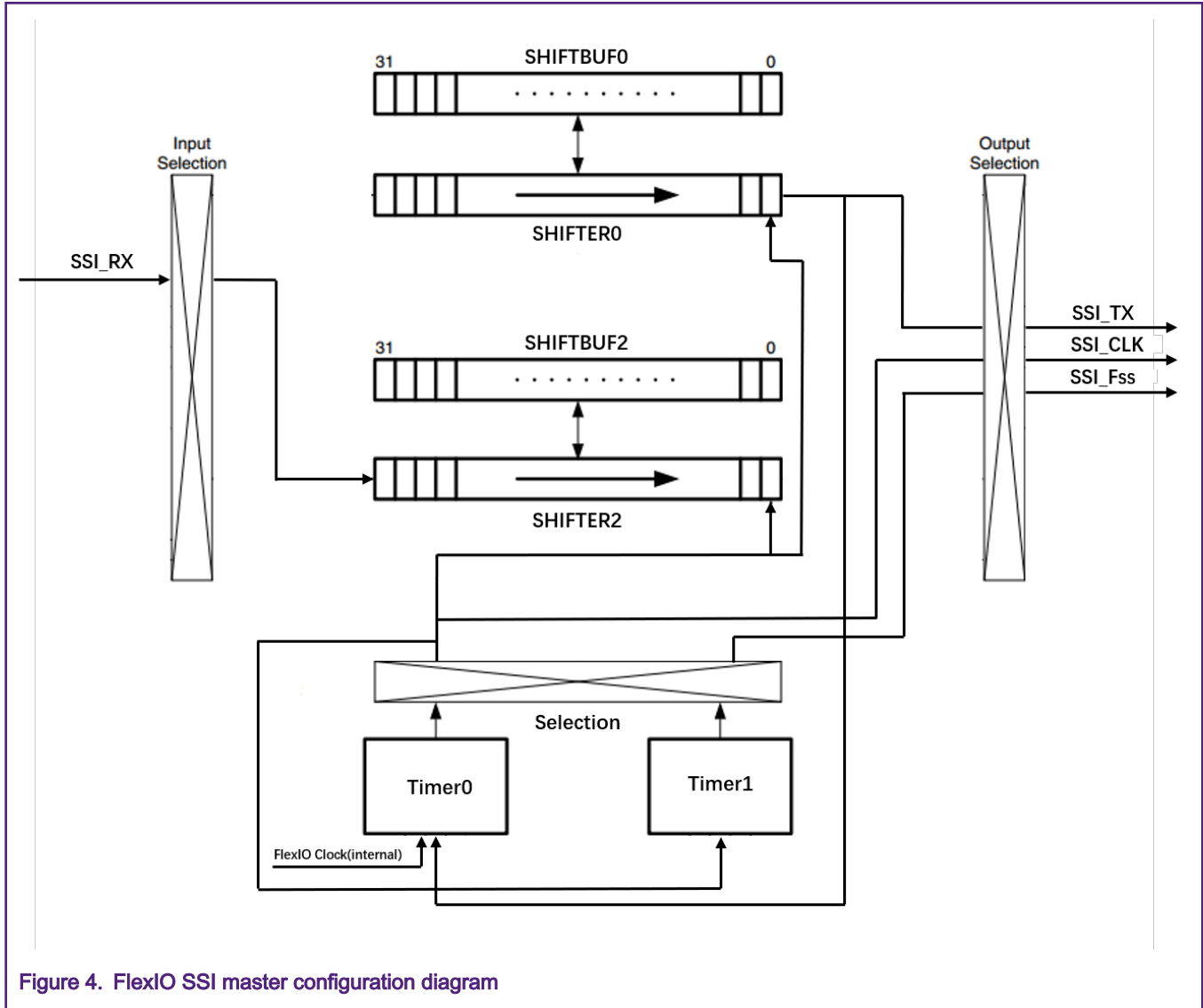


Figure 4. FlexIO SSI master configuration diagram

- **Timer 0** is configured to dual 8-bit counters, enabled on trigger high and disabled on compare event. The trigger source of **Timer 0** is connected to internal Shifter 0. The source of **Timer 0** decrement is FlexIO clock, shift clock equals Timer output. Timer 0 is triggered by shifter 0 status flag. In this application, the transfer baud rate is 200 k, so the compare value of **Timer 0** is set to 0xF1D. The start bit of **Timer 0** is enabled.
- **Timer 1** is configured to single 16-bit counter mode, triggered by **Timer 0**, enabled on **Timer 0** enable and disable on compare event. The source of **Timer 1** decrement is trigger input (both edges).
- **Shifter 0** is configured as transmit mode and shift on rising edge of shifter clock, shifter start bit enabled and is set to logic low level.
- **Shifter 2** is configured as receive mode and shift on falling edge of shifter clock, shifter start bit enabled and is set to logic low level.

The detailed register configurations are as follows:

- FIEXIO01.SHIFTCTL[0] = 0x00031502
- FIEXIO01.SHIFTCTL[2] = 0x00801601
- FIEXIO01.SHIFTCFG[2] = 0x00000002
- FIEXIO01.SHIFTCFG[2] = 0x00000002
- FIEXIO01.TIMCTL[0] = 0x01C31A01
- FIEXIO01.TIMCTL[1] = 0x03430003
- FIEXIO01.TIMCFG[0] = 0x00002222
- FIEXIO01.TIMCFG[1] = 0x00102100
- FIEXIO01.TIMCMP[0] = 0x00000F1D
- FIEXIO01.TIMCMP[1] = 0x00000002

When data has been stored into the SHIFTBUF register from the SHIFTER or when data has been loaded from the SHIFTBUF register into the SHIFTER, the SHIFTER status flag will be set and generate an enabled DMA request. Figure 5 shows the microarchitecture diagram of SHIFTER.

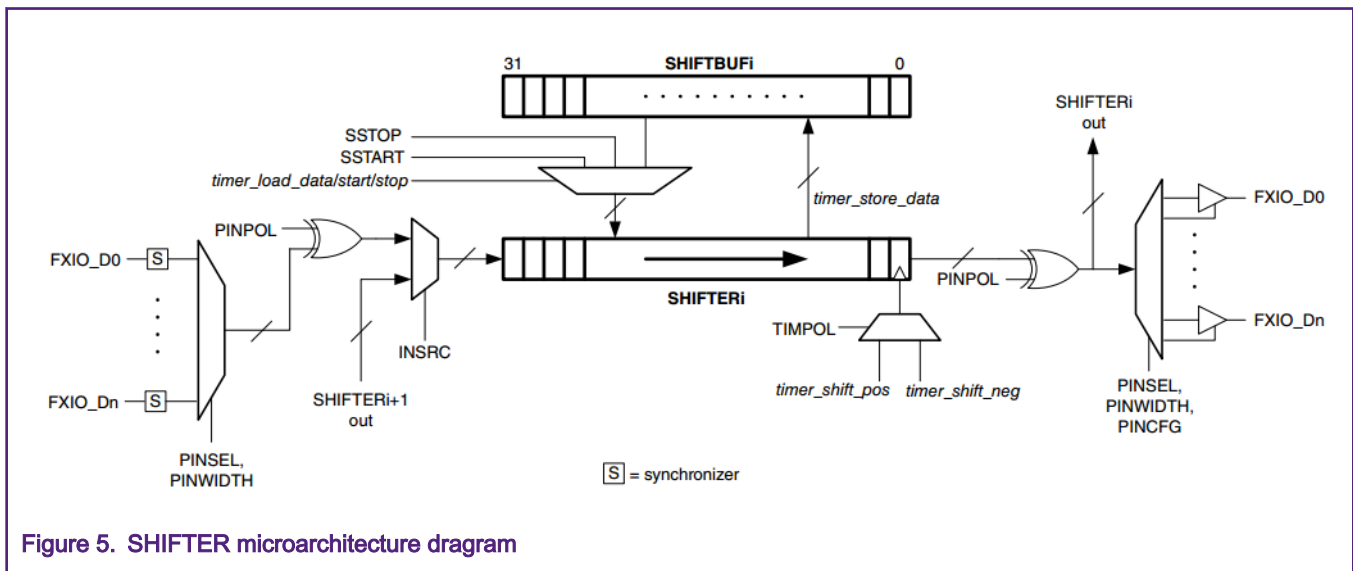


Figure 5. SHIFTER microarchitecture diagram

4.3 SSI slave configuration

It is similar as the SSI master, the emulation of the SSI slave uses two Shifters and two Timers. **Timer 0** is used to detect SSI_Fss signal. **Timer 1** is used to detect SSI_CLK signal. **Shifter 0** connects to SSI_TX pin, **Shifter 2** connects to SSI_RX pin. Figure 6 shows the FlexIO SSI slave configuration diagram.

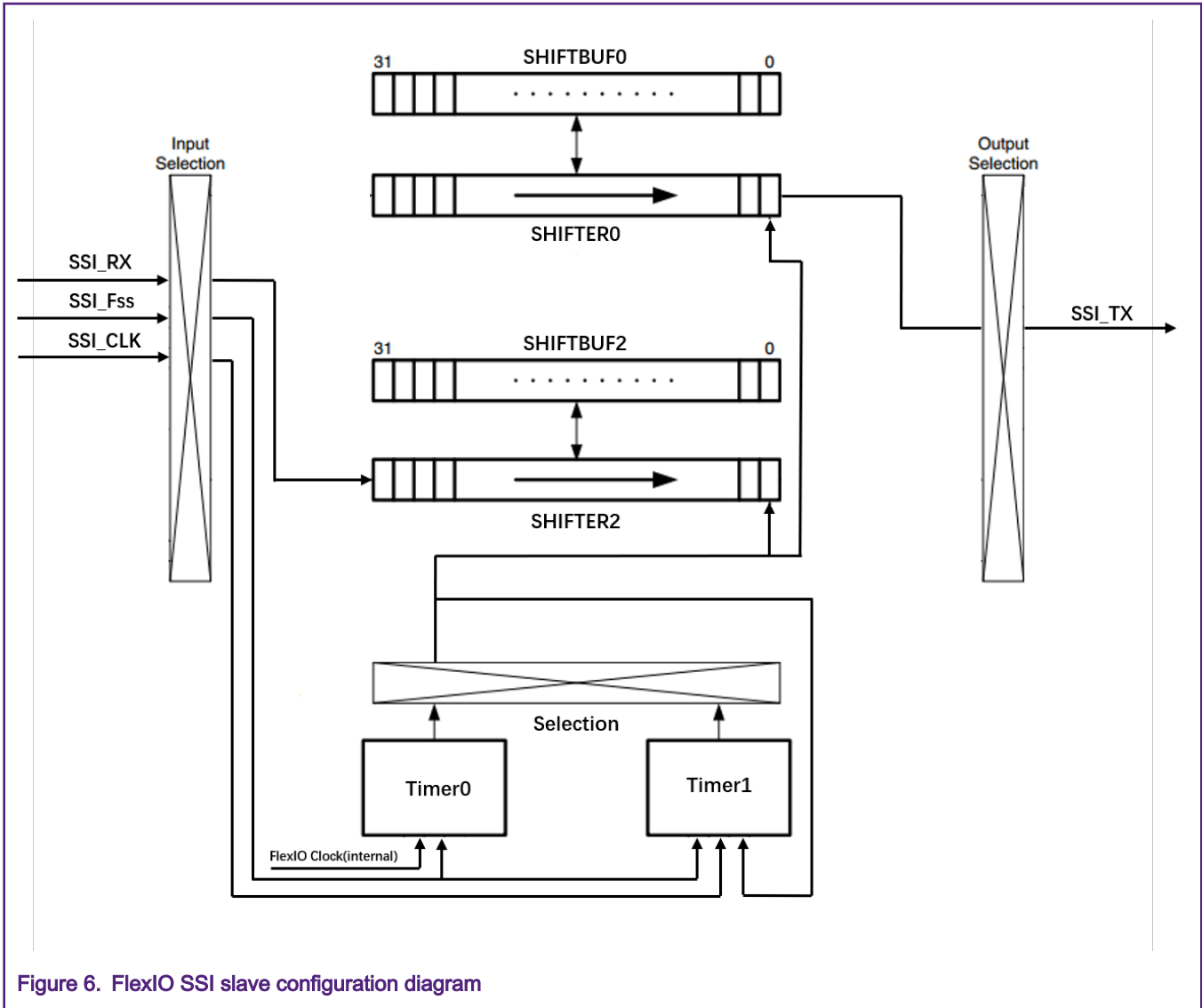


Figure 6. FlexIO SSI slave configuration diagram

- **Timer 0** is configured to dual 8-bit counters, enable on trigger high and disable on compare event. The trigger source of **Timer 0** is SSI_Fss pin input. The source of **Timer 0** decrement is FlexIO clock. The shift clock equals Timer output. As the transfer baud rate is 200 k, set the compare value of **Timer 0** to **0xF1D**. The start bit of **Timer 0** is enabled.
- **Timer 1** is configured to single 16-bit counter mode, triggered by SSI_Fss pin, enabled on trigger rising edge and disabled on **Timer 0** disable. The source of **Timer 1** decrement is SSI_CLK pin input (both edges).
- **Shifter 0** is configured as transmit mode and shift on rising edge of shifter clock, shifter start bit disable and transmitter loads data on first shift.
- **Shifter 2** is configured as receive mode and shift on falling edge of shifter clock, shifter start bit disable and transmitter loads data on first shift.

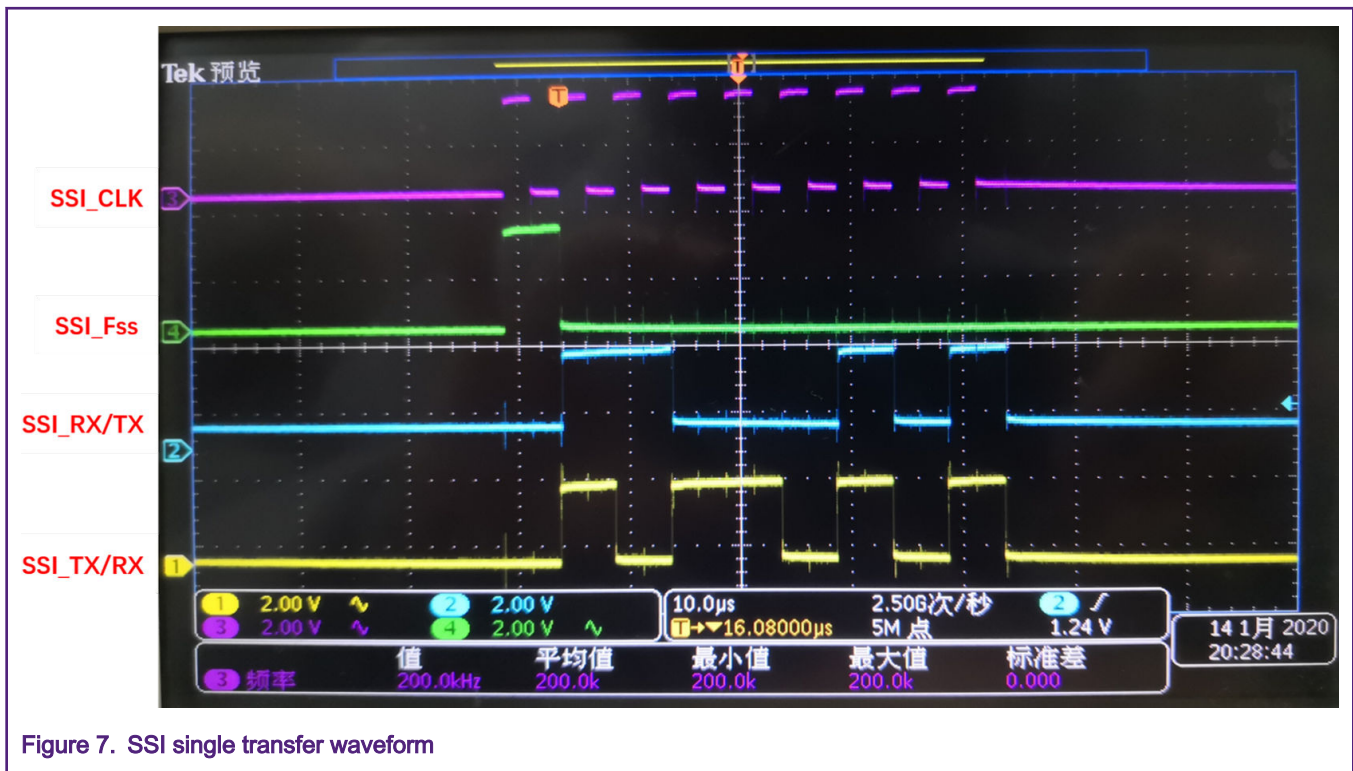
The detailed register configurations are as follows:

- FIEXIO01.SHIFTCTL[0] = 0x00031502
- FIEXIO01.SHIFTCTL[2] = 0x00801601
- FIEXIO01.SHIFTCFG[2] = 0x00000001
- FIEXIO01.SHIFTCFG[2] = 0x00000001
- FIEXIO01.TIMCTL[0] = 0x00400001

- FIEXIO01.TIMCTL[1] = 0x00401A03
- FIEXIO01.TIMCFG[0] = 0x00002402
- FIEXIO01.TIMCFG[1] = 0x01201600
- FIEXIO01.TIMCMP[0] = 0x00000F1D
- FIEXIO01.TIMCMP[1] = 0x0000000F

4.4 Running the demo

Taking the first single transmission between two boards as an example, the waveform is captured by an oscilloscope, as shown in Figure 7. The SSI master board sends 0xc5 to the SSI slave board and receives 0xb5 at the same time.



Download the SSI images into the i.MX RT1010EVK boards and power on. The communication process between the SSI master and slave is as shown in Figure 8.

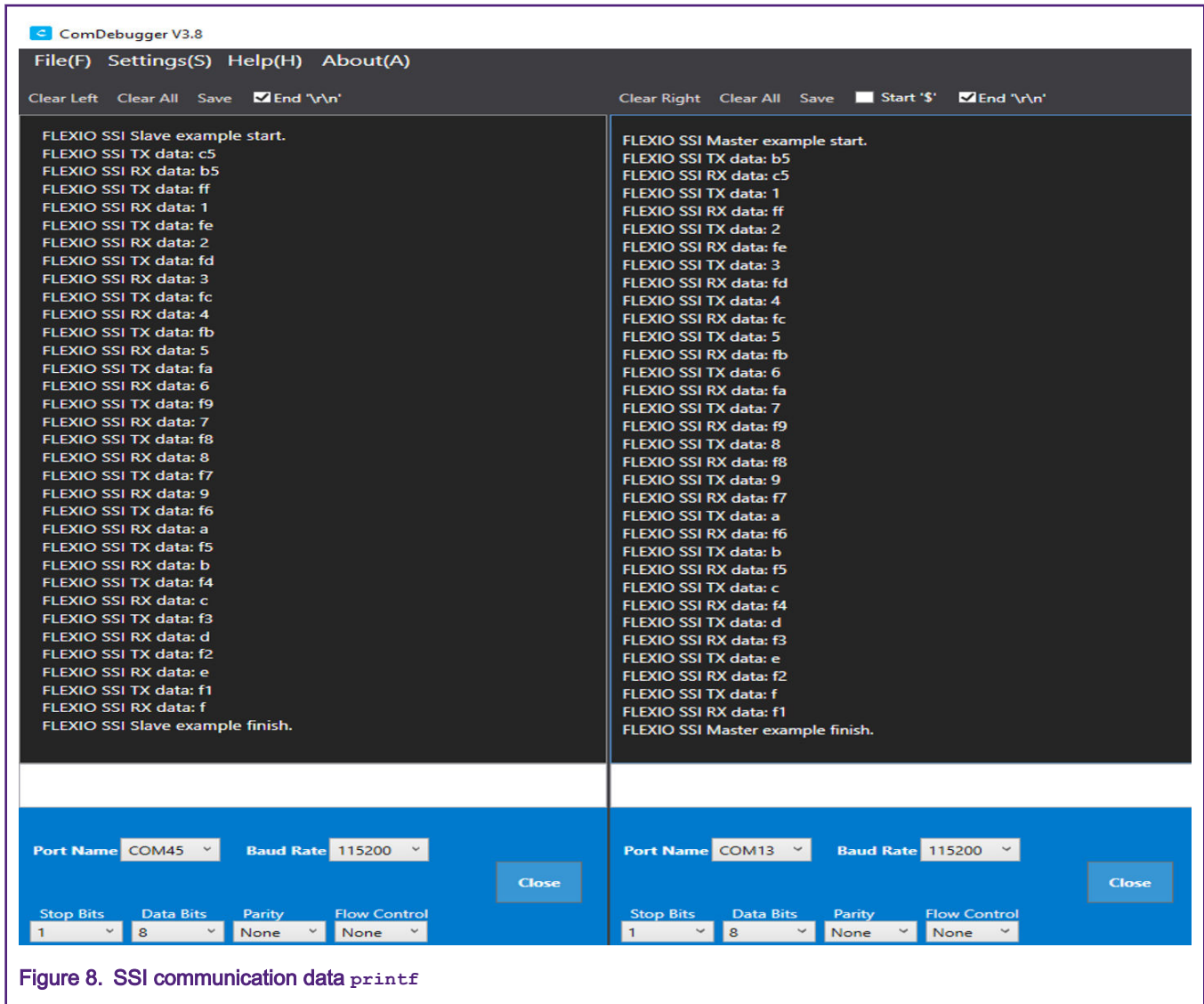


Figure 8. SSI communication data printf

5 Conclusion

This application note introduces an example of the SSI that can be implemented via the FlexIO module provided by the RT1010 MCU. If CPU resources are insufficient, it can emulate the SSI interface to communicate with the sensor device.

When using FlexIO module to emulate SSI interface:

- Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles and the maximum baud rate is divide by 4 of the FlexIO clock frequency.
- Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles and the maximum baud rate is divide by 6 of the FlexIO clock frequency.
- In addition to the configuration of Timer and SHIFTER described in this document, there can be other combinations.

6 References

- *i.MX RT1010 Processor Reference Manual (Rev. B, 07/2019)* (document [IMXRT1010RM](#))
- *Emulating SSI Using FlexIO* (document [AN5397](#))

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020.

All rights reserved.

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: March 31 2020

Document identifier: AN12759

