AN13358

FlexIO 仿真 SPI 从机在具有动态帧大小的连续模式下工作

版本 0 — 2021 年 8 月 23 日

应用笔记

内容

1 介绍

本应用笔记介绍了如何实现在 RT1010 上使用 FlexIO 模拟 SPI Slave 设备，在连续模式下接收动态大小数据帧的方法。

应用笔记涉及的硬件环境是 RT1010 EVK Rev C, 软件版本为 SDK 2.9.1。工程是基于 edma_lpspi_transfer 中 slave 项目进行开发，具体路径如下：

```
boards\evkmimxrt1010\driver_examples\flexio\spi\edma_lpspi_transfer\slave
```

FlexIO 是 i.MX RT 系列 MCU 支持的外设，它具有高度的灵活性和可被配置性，可以模拟常见的 UART、I2C、I2S 接口等。

2 FlexIO 概述

RT1010 的 FlexIO 模块支持以下功能：

- 多组 32bit 移位寄存器（shifter），支持发送、接收和数据匹配三种工作模式。
- 具有双重缓存区的移位寄存器支持连续的数据传输。
- 生成并插入起始位和停止位。
- 支持中断，DMA 或者轮询的方式进行数据的收发。
- 可配置波特率，支持 Stop 模式下的异步操作。
- 高度灵活的 16 位定时器，支持各种内部或外部触发器、复位、使能与失能方式。
- 可编程的逻辑模式，通过集成外部数字逻辑功能芯片或结合引脚/移位器/定时器功能，以产生复杂的输出。
- 支持当 CPU 进入低功耗状态后，各项系统控制功能模式下的可编程状态机，支持最多 8 个状态，8 个输出和每个状态 3 个可选择的输入。

图 1 为 FLEXIO 模块的模块功能框图。

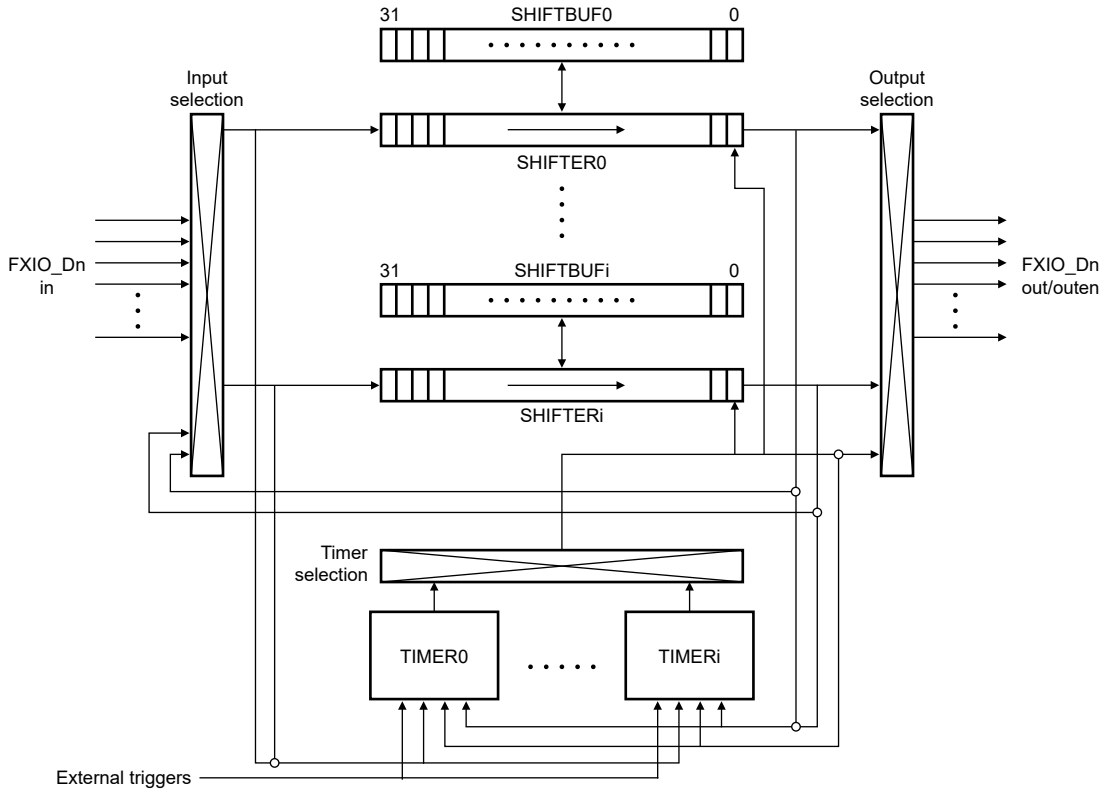

图 1. FlexIO 模块功能框图

i.MX RT 系列 MCU 的 FlexIO 模块所包含的资源不尽相同，用户可根据 FLEXIO_PARAM 寄存器读取所用芯片 FlexIO 模块包含的移位器、定时器、引脚及触发器数量。本例程中所使用的 i.MX RT1010 芯片中包含 8 个寄存器，8 个定时器，32 个引脚以及两个外部触发器（由于芯片设计引脚的限制，RT1010 的实际使用的 FlexIO 引脚只有 27 个）。

3 模拟一个 SPI 从机设备

3.1 非连续模式下 SPI 从机设置

在开始阅读下面内容之前，建议学习 AN12780 作为基础。实现非连续模式下 SPI 从机，需要用到以下资源：

- 一个定时器（Timer）：用于控制两个移位器的数据读取、写入和移位操作。
- 两个移位器（Shiter）：分别用于数据的发送与接收。
- 四个引脚（Pin）：分别用于连接 SPI 总线的 SPI_CS、SPI_SCK、SPI_MOSI 和 SPI_MISO。

图 2 为使用 FlexIO 模拟 SPI 从机时的结构框图。

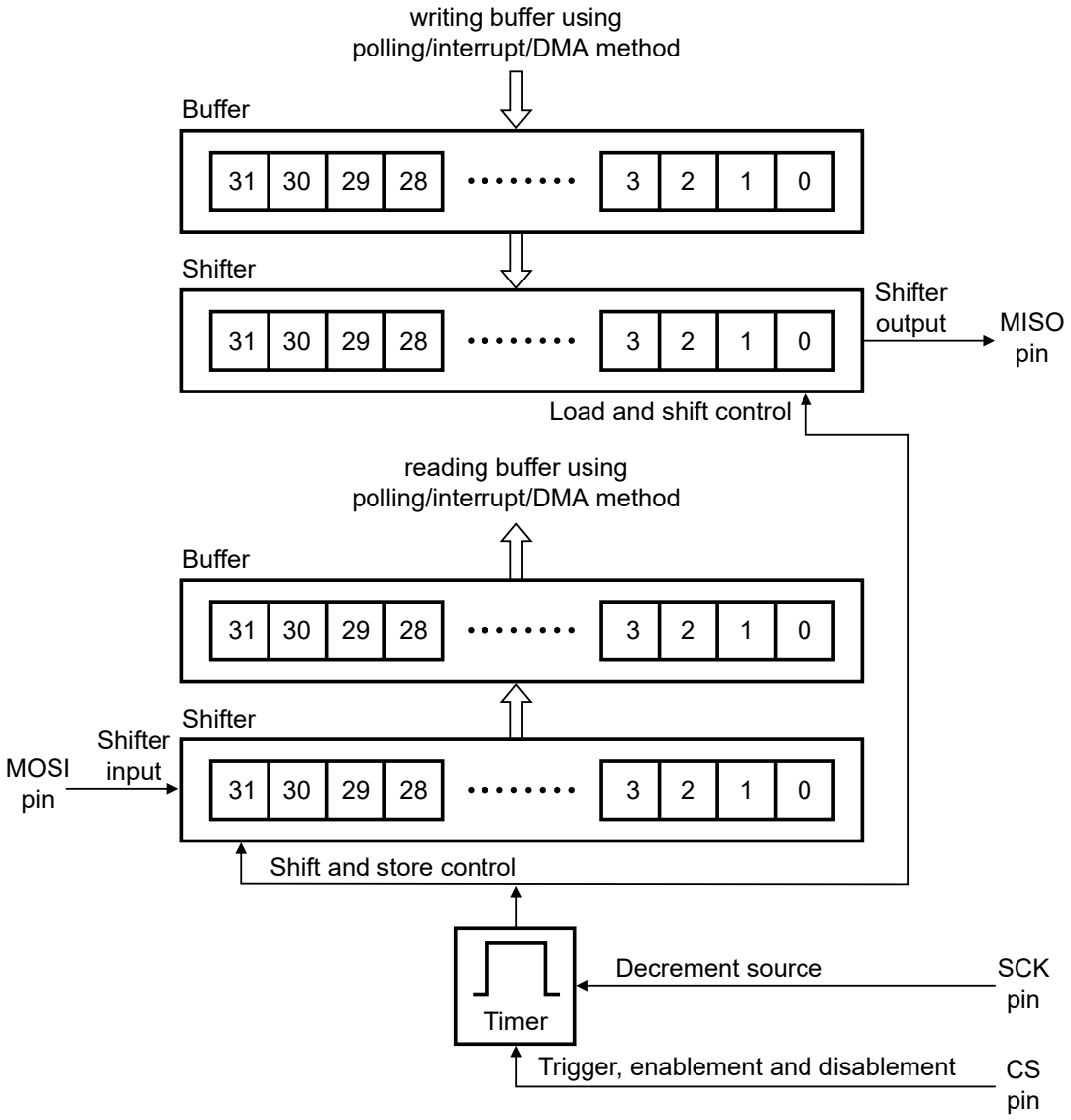

图 2. 使用 FlexIO 模拟 SPI 从机时的结构框图

在从机状态下，Timer0 连接到 FlexIO_D26 Pin 用于接收从 SPI 主机发送出的 CLK 信号，并根据这个信号来进行计数控制两个移位器的数据读取，存储和移位。CLK 和 CS 信号被配置为输入状态，FlexIO_D0 作为 CS 信号的输入引脚，用于触发 Timer 0。FlexIO_D21 pin 连接到 Shifter 0，用于输出从机发送的数据。FlexIO_D22 pin 连接到 Shifter1 用于接收来自主机的数据。

3.2 连续模式下 SPI 从机设置

为了实现连续模式，Timer0 需要在非连续模式设置下进行一些改动。首先 Timer0 需要在 CS pin 拉高时关闭，即 CS pin 的上升沿时关闭。其次，Timer0 在传输数据帧传输过程中需要一直开启。此时的数据帧不是特指 1 个 byte 的数据，而是广泛的指一个或多个 byte 组成的数据帧。然而，因为 Timer0 无法在一帧数据的最后关闭——CS 信号拉高总是晚于最后一个数据的收发——这意味着，总会在最后一个字符的最后一位传输后发生一次额外的加载。请注意图 3 中用绿色标记的地方。额外加载的原因在于当 shifter 工作在发送模式时，Timer 计满之后将会从 SHIFBUF 加载新的数据到 TX shifter。这意味着在下一轮数据传输时，一个无效的数字将会被传输。

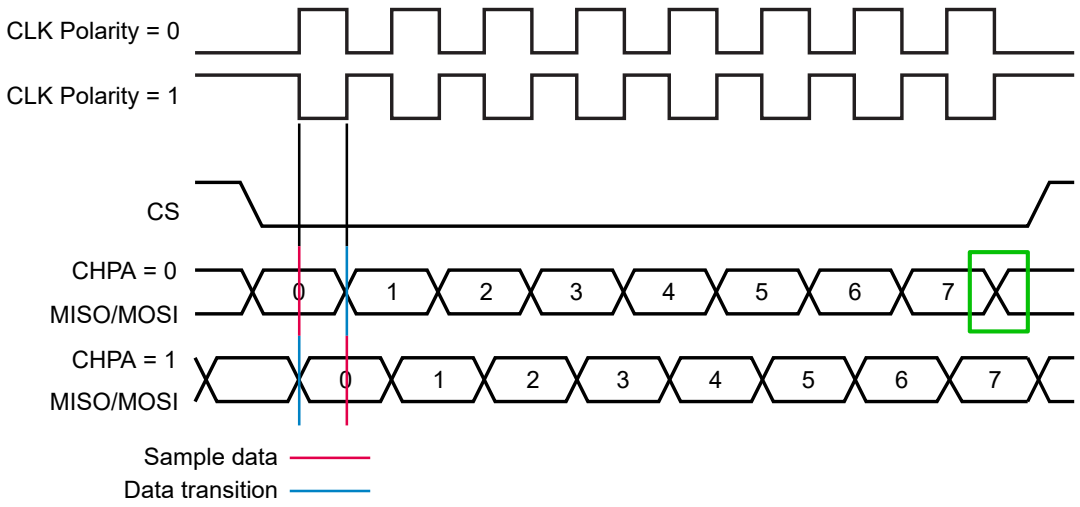

图 3. FlexIO SPI 从模块框图

对于 RX shifter, 它将会在 CS 拉高的时候触发一次额外的存储事件。因为当 CS 拉高的时候，Timer 会被关闭，此时 RX shifter 将会把 shifter 内的数据存储到 SHIFBUF 中，随后马上发起了新的一次 DMA 搬运。这意味着一个无效的数据被存储并且该数据通常是 0。

为了避免这些问题，清空 shifter 内的数据即可解决问题。因为 FlexIO 模块没有相关的 bit 可以清空 shifter 寄存器的功能，所以可以使用下面的方法实现清空 shifter 的功能。

TX shifter: 关闭 shifter 后再次配置为 TX 模式，对应的 shifter 内数据即可清空。

RX shifter: 读一下 shifter 对应的 buffer 寄存器，shifter 内的数据即可清空。

在例程代码中，shifter0 是 TX 移位器 shifter1 是 RX 移位器。因此在完成一帧数据的传输后，调用下面的接口函数即可实现共两个 shifter 数据的清空。

```
void FLEXIO_SPI_FlushShifters(FLEXIO_SPI_Type *base)
{
    volatile uint32_t tmp;
    base->flexioBase->SHIFTCTL[base->shifterIndex[0]] &= ~FLEXIO_SHIFTCTL_SMOD_MASK;
    base->flexioBase->SHIFTCTL[base->shifterIndex[0]] |=
FLEXIO_SHIFTCTL_SMOD(kFLEXIO_ShifterModeTransmit);
    tmp = base->flexioBase->SHIFTBUF[base->shifterIndex[1]];
    __DSB();
}
```

实际上，这个附加的加载事件在数据帧的每一个字符传输过程中都会发生，但是因为工作在连续模式，所以不会有异常发生。

表 1 描述了 Timer0 的具体设置。用红色标记的地方是区别于非连续模式的设置，更改这里后即可实现从机工作在连续模式。

表 1. Timer 0 配置

项目	配置
Trigger Select	Trigger from FlexIO_D0 input
Trigger Polarity	Active low
Trigger Source	Internal trigger
Pin Config	Output disable

下页继续...

表 1. Timer 0 配置 (continued)

项目	配置
Pin Select	FlexIO_D26
Pin Polarity	Active high
Timer mode	Single 16-bit counter mode
Timer Output	Timer output is logic zero when enabled and is not affected by timer reset
Timer Decrement	Decrement counter on pin input, shift clock equals pin input
Timer Reset	Timer never reset
Timer Disable	Timer disabled on trigger falling[1]
Timer Enable	Timer enabled on trigger rising edge[1]
Timer Stop Bit	Disable
Timer Start Bit	Disable
Timer Compare	15 (bitCountPerChar * 2 - 1)

1. 因为 trigger Polarity 设置为 active low, 所以时序极性发生改变，因此使能信号为上升沿（真实的输入使能信号为下降沿）。

表 2 描述了用于 TX 功能的 shifter0 具体设置。

表 2. Shifter 0 TX 模式配置

项目	配置
Timer Select	Timer 0
Timer Polarity	Shift on falling edge of shift clock
Pin Config	Shifter pin output
Pin Select	FlexIO_D21
Pin Polarity	Active high
Shifter Mode	Transmit mode
Input Source	Input from pin
Shifter Stop Bit	Disable
Shifter Start Bit	Disable, transmitter loads data on enable

表 3 描述了用于 RX 功能的 shifter1 具体设置。

表 3. Shifter 1 RX 模式配置

项目	配置
Timer Select	Timer 0
Timer Polarity	Shift on rising of shift clock
Pin Config	Output disable
Pin Select	FlexIO_D22
Pin Polarity	Active high
Shifter Mode	Receive mode
Input Source	Input from pin
Shifter Stop Bit	Disable
Shifter Start Bit	Disable, transmitter loads data on enable

3.3 连续模式下 SPI 从机接收动态帧大小的设置

SDK 默认例程中，DMA 被用来搬运数据以提升效率。但是 DMA 中断通常是在接收到一定数量的数据后产生。但在实际的应用中，每帧传输的数据的大小可能不是固定的，从机也不知道在一帧数据中有多少数据。在这种未知一帧数据大小的情况下，使用 DMA 接收到固定数量的数据后产生中断的方法不能满足实际需求。在 SPI 传输过程中，CS 引脚的下降沿表示传输开始，CS 引脚的上升沿表示传输结束。因此，从机可以通过检测 CS 引脚的上升沿和下降沿来知道一帧数据已经完成传输。

为了实现这个功能，可以增加一个定时器（Timer）来检测 CS 引脚的状态。基本方法是让定时器在 16 位计数器模式下工作，计数值为 0。这意味着一旦定时器超时，它会产生一个比较事件，并会产生一个 FLEXIO 中断。从机再处理此中断并且解析接收数据的数量。现在，有两个问题。第一个是定时器何时启用（timer enables source）以及定时器递减源（timer decrement source）是什么，第二个是从机如何知道它使用 DMA 接收了多少数据。

对于第一个问题，定时器可以通过引脚的下降沿或触发信号得下降沿使能。关于定时器递减源，引脚输入递减或触发输入递减均可使用。定时器可以通过 CS 引脚的下降沿使能，然后当 CS 引脚上升时，比较事件（compare event）发生，同时产生 FlexIO 中断。除此之外，定时器还需要在定时器比较事件时关闭。

表 4 描述了使用 pin 下降沿作为定时器使能递减源的具体配置。表 5 描述了使用 trigger 下降沿作为定时器使能递减源的具体配置。

表 4. Timer 1 使用 pin 下降沿的配置

项目	配置
Trigger Select	NA
Trigger Polarity	NA
Trigger Source	NA
Pin Config	Output disable
Pin Select	FlexIO_D00

下页继续...

表 4. Timer 1 使用 pin 下降沿的配置 (continued)

项目	配置
Pin Polarity	Active low
Timer mode	Single 16-bit counter mode
Timer Output	Timer output is logic one when enabled and is not affected by timer reset
Timer Decrement	Decrement counter on Pin input (both edges) Shift clock equals Pin input
Timer Reset	Timer never reset
Timer Disable	Timer disabled on Timer compare
Timer Enable	Timer enabled on Pin rising edge[1]
Timer Stop Bit	Disable
Timer Start Bit	Disable
Timer Compare	0

1. 因为 pin Polarity 设置为 active low, 所以时序极性发生改变，因此使能信号为上升沿（真实的输入使能信号为下降沿）。

表 5. Timer 1 使用 trigger 下降沿的配置

项目	配置
Trigger Select	Trigger from FlexIO_D0 input
Trigger Polarity	Active low
Trigger Source	Internal trigger
Pin Config	Output disable
Pin Select	NA
Pin Polarity	NA
Timer mode	Single 16-bit counter mode
Timer Output	Timer output is logic one when enabled and is not affected by timer reset
Timer Decrement	Decrement counter on trigger input (both edges) Shift clock equals timer output
Timer Reset	Timer never reset
Timer Disable	Timer disabled on Timer compare
Timer Enable	Timer enabled on trigger rising edge[1]

下页继续...

表 5. Timer 1 使用 trigger 下降沿的配置 (continued)

项目	配置
Timer Stop Bit	Disable
Timer Start Bit	Disable
Timer Compare	0

1. 因为 trigger Polarity 设置为 active low, 所以时序极性发生改变，因此使能信号为上升沿（真实的输入使能信号为下降沿）。

第二个问题是从机如何知道它使用 DMA 接收了多少数据。下面得这个接口函数 API 可以用来实现这个功能：

```
static inline status_t FLEXIO_SPI_SlaveTransferGetCountEDMA(FLEXIO_SPI_Type *base,
                                                            flexio_spi_slave_edma_handle_t *handle,
                                                            size_t *count)
```

对于这个 API 函数，有两点需要注意。第一点是这个 API 得注释描述不是很准确（将会再 SDK 2.11.0 版本进行更新）。这个 API 的功能是：获取当前通过 FlexIO SPI DMA 的方法，接收到的数据数量。第二点是这个 API 返回的结果（count，API 中第三个参数）不能直接使用，正如上文所述，因为在 CS 拉高的时候会触发额外的一次 DMA 搬运。 因此，接收到的数据大小应该是：

```
size = count - 1; //The count is the third parameter on API
```

在获取数据帧大小后，需要使用下面的 API 终止 FlexIO SPI DMA 的传输。但是现在，这个 API 实现的功能是暂停传输而不是终止传输。这意味着这个 API 需要进行一些改动才能满足要求。这个问题将会在 SDK 2.11.0 进行更新。

```
void FLEXIO_SPI_SlaveTransferAbortEDMA(FLEXIO_SPI_Type *base, flexio_spi_slave_edma_handle_t *handle)
```

在 fsl_flexio_spi_edma.c 的第 409 和 410 行使用下面的代码替代即可。

```
EDMA_AbortTransfer(handle->txHandle);
EDMA_AbortTransfer(handle->rxHandle);
```

解决这两个问题后，从机就可以接收数据大小变化的数据帧了。但是在这种情况下仍然有一个限制，即帧的大小应小于或等于 DMA 可接收的数量。

4 例程代码

在运行代码之前，EVK 板需要进行一定的改动：

- 去掉电阻 R90，使用 0Ω 电阻焊接到 R800。
- 按照下面的设置将 SPI 的主机和从机连接起来。

Pin Name	Master (LPSPI1)		Pin Name	Slave (FlexIO SPI)
SOUT	J57-8	<---->	SIN	J26-6
SIN	J57-10	<---->	SOUT	J26-4
SCK	J57-12	<---->	SCK	J26-8
CS	J57-6	<---->	CS	J56-10

具体实现代码可以在应用笔记的附件中找到，打开 edma_lpspi_transfer slave 工程，使用附件中的文件替换。具体路径为：boards\evkmimxrt1010\driver_examples\flexio\spi\edma_lpspi_transfer\slave

在默认设置下，主机将向从机发送 16 个字节，从机将产生一个 FlexIO 中断。从机可以接收的最大数量为 64 字节。用户可以更改以下参数来配置主机发送数据帧的大小。

```
masterXfer.dataSize = TRANSFER_SIZE;
```

图 4 为连续模式下 SPI 数据传输的波形图。在一帧数据完成发送/接收后，将会进行一次简单的校验。如果校验通过，在串口终端输入任意一个字符即可开启新的一轮数据传输。

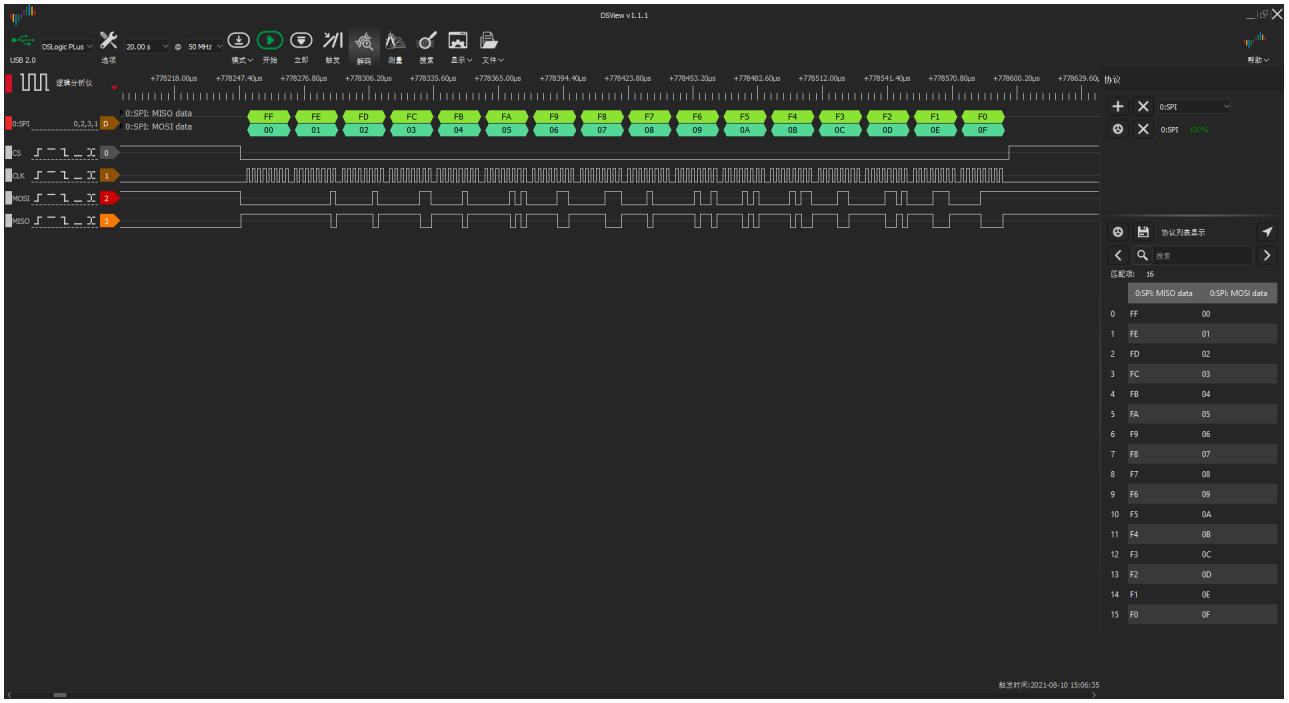

图 4. 连续模式下 SPI 数据传输的波形图

5 修订记录

版本号	日期	说明
0	08/2021	初始版本

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — 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.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. 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.

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

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

Date of release: 2021 年 8 月 23 日

Document identifier: AN13358

arm