ANALOG DEVICES 超越一切可能™ | ADI AnalogDialogue

操纵MCU SPI接口以访问非标准SPI ADC

Steven Xie，产品应用工程师

问题：

能否用MCU访问非标准SPI接口？

回答：

可以，但可能需要做一些额外的努力。

简介

当前许多精密模数转换器(ADC)具有串行外设接口(SPI)或某种串行接口，用以与包括微控制器单元(MCU)、DSP和FPGA在内的控制器进行通信。控制器写入或读取ADC内部寄存器并读取转换码。SPI的印刷电路板(PCB)布线简单，并且有比并行接口更快的时钟速率，因而越来越受欢迎。而且，使用标准SPI很容易将ADC连接到控制器。

一些新型ADC具有SPI，但有些ADC具有非标准的3线或4线SPI作为从机，因为它们希望实现更快的吞吐速率。例如，AD7616、AD7606和AD7606B系列有两条或四条SDO线，在串行模式下可提供更快的吞吐速率。AD7768、AD7779和AD7134系列有多条SDO线，用作SPI主机。用户在设计微控制器SPI以配置ADC和读取代码时往往会遇到困难。

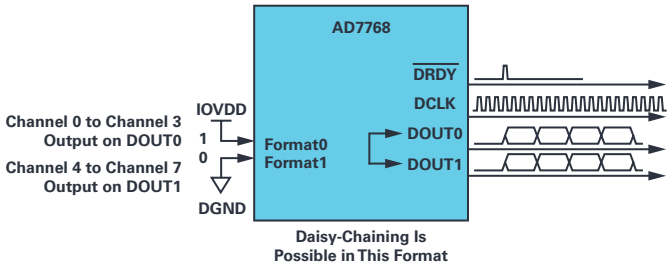

图1. AD7768用作串行主机，具有两个数据输出引脚(14001-193)。

与ADC的标准MCU SPI连接

SPI是一种同步、全双工、主从式接口。来自主机或从机的数据在时钟上升沿或下降沿同步。主机和从机可以同时传输数据。图2显示了典型的4线MCU SPI接口连接。

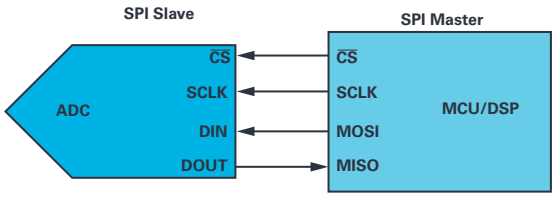

图2. 与ADC从机的标准MCU SPI连接。

要开始SPI通信，控制器必须发送时钟信号，并通过使能$\overline{CS}$信号（通常是低电平有效信号）来选择ADC。SPI是全双工接口，因此控制器和ADC可以分别通过MOSI/DIN和MISO/DOUT线同时输出数据。控制器SPI接口允许用户灵活选择时钟的上升沿或下降沿来采样和/或移位数据。为了在主机和从机之间进行可靠的通信，用户必须遵守微控制器和ADC芯片的数字接口时序规范。

如果微控制器SPI和ADC串行接口具有标准SPI时序模式，那么用户设计PCB布线和开发驱动器固件不成问题。但是，有些新型ADC的串行接口端口不是典型的SPI时序模式。MCU或DSP似乎不可能通过AD7768串行端口（一种非标准时序SPI端口）读取数据，如图4所示。

本文将介绍操纵标准微控制器SPI以便与具有非标准SPI端口的ADC接口的方法。

本文会给出四种通过串行接口读取ADC码的解决方案：

- 解决方案1：MCU作为SPI从机，通过一条DOUT线与作为SPI主机的ADC接口。
- 解决方案2：MCU作为SPI从机，通过两条DOUT线与作为SPI主机的ADC接口。
- 解决方案3：MCU作为SPI从机，通过DMA与作为SPI主机的ADC接口。
- 解决方案4：MCU作为SPI主机和SPI从机，通过两条DOUT线读取数据。

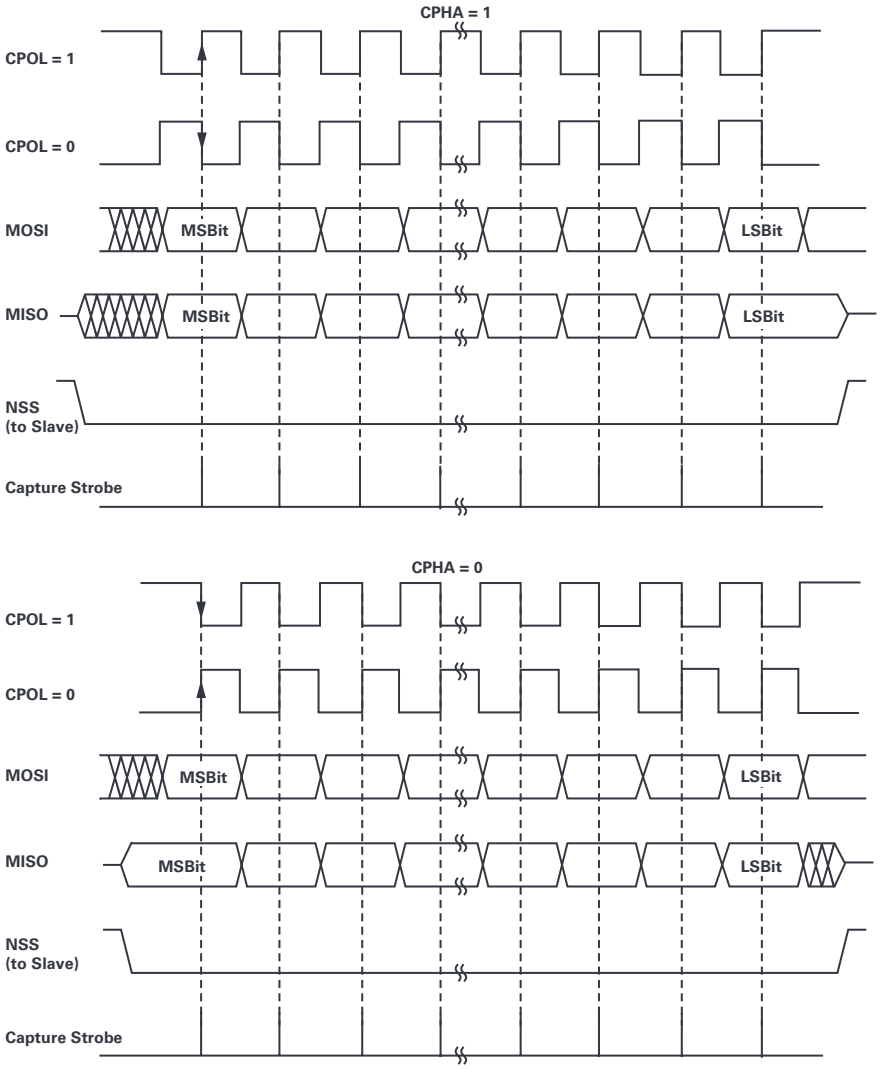

图3. SPI数据时钟时序图示例。

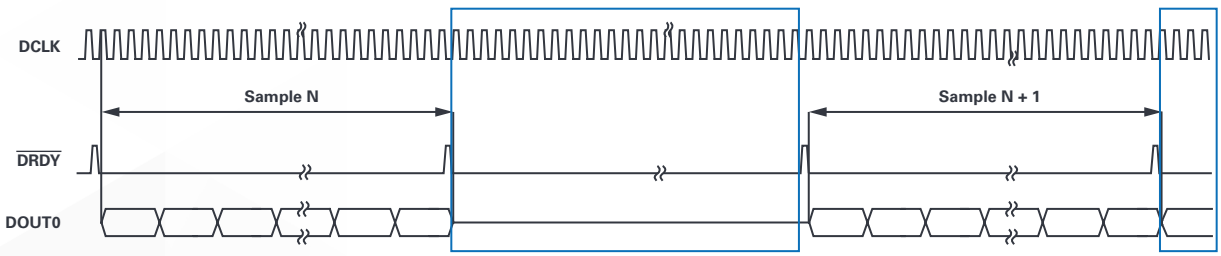

图4. AD7768 FORMATx = 1×时序图，仅通过DOUT0输出。

STM32F429微控制器SPI通过一条DOUT线读取AD7768代码

如图4所示，当FORMATx = 11或10时，通道0至通道7仅通过DOUT0输出数据。在标准工作模式下，AD7768/AD7768-4作为主机工作，数据流入MCU、DSP或FPGA。AD7768/AD7768-4向从机提供数据、数据时钟(DCLK)和下降沿帧使能信号(DRDY)。

STM32Fxxx系列微控制器广泛用于很多不同的应用中。该MCU有多个SPI端口，可以使用典型的SPI时序模式将其配置为SPI主机或从机。下文中介绍的方法也可应用于其他具有8位、16位或32位帧的微控制器。

AD7768/AD7768-4分别为8通道和4通道同步采样Σ-Δ型ADC，每通道均有Σ-Δ型调制器和数字滤波器，支持交流和直流信号的同步采样。这些器件在110.8 kHz的最大输入带宽下实现了108 dB动态范围，具备±2 ppm INL、±50 μV偏置误差和±30 ppm增益误差的典型性能。AD7768/AD7768-4用户可在输入带宽、输出数据速率和功耗之间进行权衡，并选择三种功耗模式之一以优化噪声目标和功耗。AD7768/AD7768-4的灵活性使其成为适合低功耗直流和高性能交流测量模块的可重复使用平台。遗憾的是，AD7768的串行接口不是典型SPI时序模式，而且AD7768充当串行接口主机。一般而言，用户必须使用FPGA/CPLD作为其控制器，

例如，使用32F429IDISCOVERY和AD7768评估板。变通SPI线的连接如图5所示。在这种设置下，AD7768的所有八通道数据仅通过DOUT0输出。

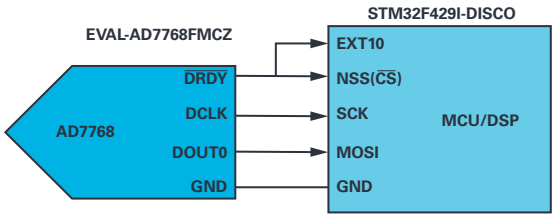

图5. AD7768通过DOUT0将数据输出到STM32F429 MCU SPI连接。

需要解决的问题：

- AD7768用作SPI主机，故必须将STM32F429I SPI配置为SPI从机。
- $\overline{DRDY}$ 高电平脉冲只持续一个DCLK周期，这不是典型的$\overline{CS}$。
- 完成所有通道数据位的输出之后，DCLK继续输出，$\overline{DRDY}$为低电平。

解决方案1：MCU SPI作为从机，通过一条DOUT线与SPI主机ADC接口

- 将STM32F429的一个SPI端口（如SPI4）配置为从机，以DCLK速率接收MOSI上的数据位。
- 将AD7768 $\overline{DRDY}$连接到STM32F429外部中断输入引脚EXTI0和NSS (SPI $\overline{CS}$)引脚。$\overline{DRDY}$的上升沿将触发EXTI0处理例程，以使SPI从机能够在$\overline{DRDY}$变为低电平之后的第一个DCLK下降沿开始接收数据位。时序设计在这里至关重要。
- 接收到通道0至通道7的所有数据后，应禁用SPI以防止读取额外的无效数据，因为$\overline{DRDY}$会使SPI从机$\overline{CS}$变为低电平，并且DCLK保持切换。

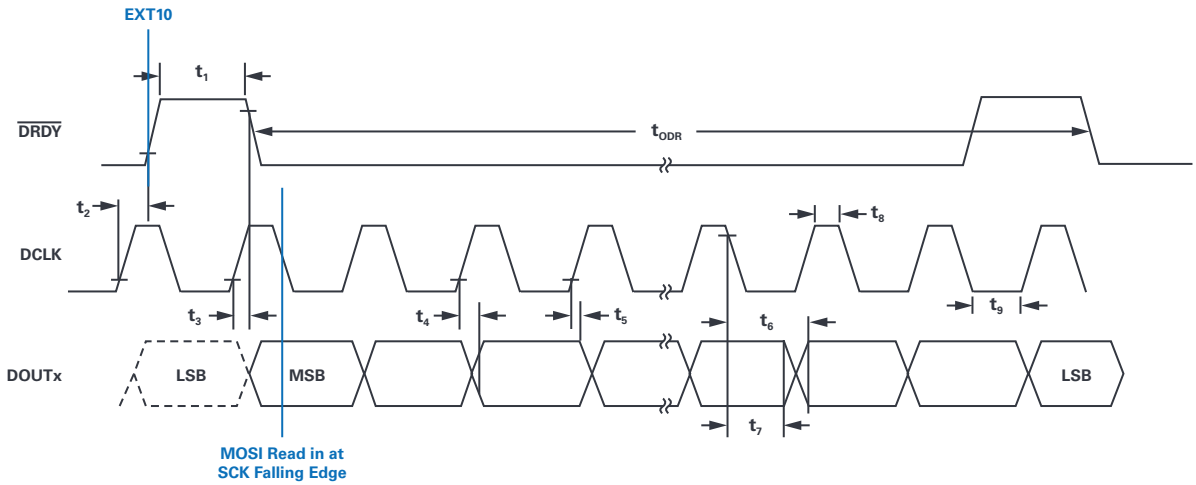

图6. 时序解决方案中的AD7768数据位读取。

MCU固件开发注意事项

当软件处于中断模式时，DCLK运行速率可以高达4 MHz，实现8 kSPS的ODR。软件应进入中断处理程序，在一个半DCLK周期(375 ns)内启动SPI。

为使软件更轻松地进入中断例程，MCU可以在DCLK上升沿读取数据，从而提供额外的半个DCLK周期时间。但是，t5 DCLK上升到DOUTx无效最小值为–3 ns（IOVDD = 1.8 V时为–4 ns），因此DOUTx上的传播延迟（>|t5| + MCU保持时间）应通过PCB布线或缓冲增加。

```
/*## Configure the SPI4 peripheral ###*/
Spi4Handle.Instance                      = SPI4;//use STM32F429 SPI4
Spi4Handle.Init.Direction                = SPI_DIRECTION_2LINES_RXONLY;
Spi4Handle.Init.CLKPhase                 = SPI_PHASE_1EDGE;//read at DCLK falling edge
Spi4Handle.Init.CLKPolarity              = SPI_POLARITY_HIGH;//read at DCLK falling edge
Spi4Handle.Init.DataSize                 = SPI_DATASIZE_8BIT;//or 16BIT
Spi4Handle.Init.NSS                      = SPI_NSS_HARD_INPUT;//make /CS low active
Spi4Handle.Init.Mode                     = SPI_MODE_SLAVE;//MCU SPI4 as SPI Slave
/*## Enable EXTI0 and SPI4 to Receive AD7768 Data bits ###*/
// clear EXTI0 IT flag prior to enable external interrupt 0 !!!
__HAL_GPIO_EXTI_CLEAR_IT(KEY_BUTTON_PIN);
HAL_NVIC_EnableIRQ(EXTI0_IRQn);
// wait for EXTI0 interrupt (/DRDY rising edge) to prepare for reading last conversion data
if (EXTI0_Flag == SET)
{
          EXTI0_Flag = RESET;//clear /DRDY rising edge flag variable
          // throw out the last byte/word captured in the previous ODR cycle !!!
          Rx_temp = *(__IO uint8_t *)&Spi4Handle.Instance->DR;
          __HAL_SPI_ENABLE(&Spi4Handle);
          // SPI4_CNVByteNum is the total data byte number to read in one conversion cycle
          while (SPI4_ByteCount < SPI4_CNVByteNum)
          {
                    // Check the RXNE flag
                    if (__HAL_SPI_GET_FLAG(&Spi4Handle, SPI_FLAG_RXNE))//
                    {
                              // transfer the received data from DR register to memory
                              SPI_RxBuffer[RxBuf_Idn] = *(__IO uint8_t *)&Spi4Handle.Instance->DR;
                              RxBuf_Idn++;
                              SPI4_ByteCount++;
                    }
          }
          // disable SPI4 to prevent read in extra data after all channel codes finished due to /DRDY
          is low active and DCLK continuously pulses
          __HAL_SPI_DISABLE(&Spi4Handle);
          SPI4_CNVCount++;
          RxBuf_Idn = SPI4_CNVCount * SPI4_CNVByteNum;
          SPI4_ByteCount = 0;
}//end of if (EXTI0_Flag == SET)
else
{//*** other software jobs ***//}
/*## handles External 0 interrupt request ###*/
// EXTI0 rising edge triggered to leave more response time for going into EXTI0_IRQHandler !!!
void EXTI0_IRQHandler(void)
{
          if(__HAL_GPIO_EXTI_GET_IT(EXTI0) != RESET)
          {
                    // enable SPI4 as soon as possible, and make sure before the first DCLK falling edge
                    after /DRDY falling !!!
                    __HAL_SPI_ENABLE(&Spi4Handle);
                    __HAL_GPIO_EXTI_CLEAR_IT(EXTI0);
                    EXTI0_Flag = SET;
          }
}
```

图7. 配置SPI4外设。

解决方案2：MCU SPI作为从机，通过两条DOUT线与SPI主机ADC接口

在第一种解决方案中，仅使用DOUT0来输出所有8通道数据。因此，数据读取将ADC吞吐速率限制为8 kSPS。如图1所示，在DOUT0上输出通道0至通道3，在DOUT1上输出通道4至通道7，可以减少数据传输时间。串行线的连接如图7所示。通过这种改进，在DCLK为4 MHz时，ODR可以轻松达到16 kSPS。

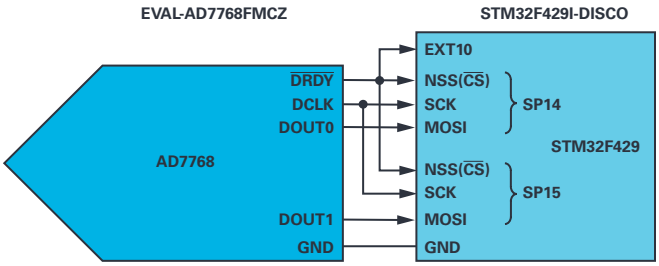

图8. AD7768通过DOUT0和DOUT1将数据输出到STM32F429 MCU SPI连接。

固件可以不使用中断模式，而使用轮询模式，以减少从$\overline{\text{DRDY}}$上升沿触发到使能SPI接收数据的时间延迟。这样可以在DCLK为8 MHz时实现32 kSPS的ODR。

解决方案3：MCU SPI作为从机，通过DMA与SPI主机ADC接口

直接存储器访问(DMA)用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。DMA可以迅速移动数据而不需要任何MCU操作，这样可以腾出MCU资源用于执行其他操作。下面是MCU SPI用作从机通过DMA接收数据的设计说明。

解决方案4：MCU SPI作为主机和从机，通过两条DOUT线读取数据

高吞吐量或多通道精密ADC为SPI端口提供两条、四条甚至八条SDO线，以在串行模式下更快地读取代码。对于具有两个或更多个SPI端口的微控制器，这些SPI端口可以同时运行以加快代码的读取。

```
/*## EXTI0 in Polling Mode and SPI4 & SPI5 to Receive AD7768 Data bits on DOUT0 and DOUT1 ###*/
// Polling for EXTI0 (/DRDY) rising edge to start MCU SPI ports
while (__HAL_GPIO_EXTI_GET_IT(EXTI0) != SET);
{
          __HAL_SPI_ENABLE(&Spi4Handle);
          __HAL_SPI_ENABLE(&Spi5Handle);
          __HAL_GPIO_EXTI_CLEAR_IT(EXTI0);
}
// throw out the last byte/word captured in the previous ODR cycle !!!
Rx_temp = *(__IO uint8_t *)&Spi4Handle.Instance->DR;
Rx_temp = *(__IO uint8_t *)&Spi5Handle.Instance->DR;
while (SPI4_ByteCount < SPI4_CNVByteNum)// total data byte number to read in one conversion cycle
{
          if (__HAL_SPI_GET_FLAG(&Spi5Handle, SPI_FLAG_RXNE))//
          {
                    SPI_RxBuffer[RxBuf_Idn] = *(__IO uint8_t *)&Spi4Handle.Instance->DR;
                    SPI_RxBuffer[RxBuf_Idn+1] = *(__IO uint8_t *)&Spi5Handle.Instance->DR;
                    RxBuf_Idn++;
                    SPI4_ByteCount += 2;
          }
}
__HAL_SPI_DISABLE(&Spi4Handle);
__HAL_SPI_DISABLE(&Spi5Handle);
```

图9. EXTI0处于轮询模式，SPI4和SPI5通过DOUT0和DOUT1接收AD7768数据位。

```
/*## EXTI0 in Polling Mode and SPI4 DMA to Receive AD7768 Data bits on DOUT0 ###*/
// Polling for EXTI0 (/DRDY) rising edge to start MCU SPI ports
while (EXTI0_Flag != SET);// wait for EXTI0 interrupt (/DRDY rising edge)
EXTI0_Flag = RESET;// clear flag variable
// throw out the last byte/word captured in the previous ODR cycle !!!
Rx_temp = *(__IO uint8_t *)&Spi4Handle.Instance->DR;
Spi4Handle.hdmarx->Instance->NDTR = SPI4_CNVByteNum;// set data number to read
Spi4Handle.hdmarx->Instance->PAR = (uint32_t)&(Spi4Handle.Instance->DR);// source address
Spi4Handle.hdmarx->Instance->M0AR = (uint32_t)(SPI_RxBuffer+RxBuf_Idn); // target address
//*** clear event flags corresponding to the stream in DMA_LISR or DMA_HISR register ***//
((DMA_Base_Registers *)Spi4Handle.hdmarx->StreamBaseAddress)->IFCR = 0x3FU << Spi4Handle.hdmarx->StreamIndex;
__HAL_DMA_ENABLE(Spi4Handle.hdmarx);
while ((Spi4Handle.hdmarx->Instance->CR & DMA_SxCR_EN) == SET) // hardware cleared
{;} // ADC data received in the target memory buffer
SPI4_CNVCount++;
RxBuf_Idn = SPI4_CNVCount * SPI4_CNVByteNum;
```

图10. EXTI0处于轮询模式，SPI4 DMA通过DOUT0接收AD7768数据位。

在以下使用案例中，32F429IDISCOVERY使用SPI4作为SPI主机，SPI5作为SPI从机，通过DOUTA和DOUTB接收EVAL-AD7606B-FMCZ数据，如图8所示。

AD7606B是一款16位同步采样模数转换数据采集系统(DAS)，具有八个通道，每个通道均包含模拟输入箝位保护、可编程增益放大器(PGA)、低通滤波器和16位逐次逼近寄存器(SAR)型ADC。AD7606B还内置灵活的数字滤波器、低漂移2.5 V精密基准电压源和基准电压缓冲器，可驱动ADC及灵活的并行和串行接口。AD7606B采用5 V单电源供电，支持±10 V、±5 V和±2.5 V真双极性输入范围，所有通道均能以800 kSPS的吞吐速率采样。

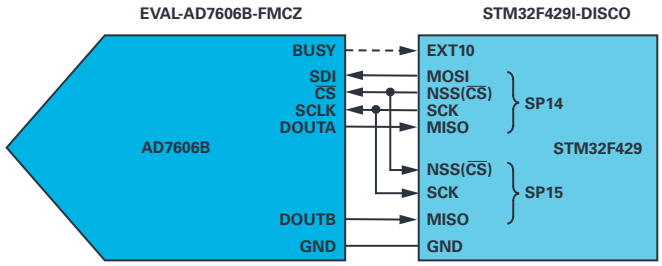

图11. 在主从模式下使用MCU SPI通过DOUTA和DOUTB接收数据。

```
/*## Configure the SPI4 as Master and SPI5 as Slave ###*/
Spi4Handle.Init.Direction      = SPI_DIRECTION_2LINES;
Spi4Handle.Init.CLKPhase       = SPI_PHASE_1EDGE;//read at DCLK falling edge
Spi4Handle.Init.CLKPolarity    = SPI_POLARITY_HIGH;//read at DCLK falling edge
Spi4Handle.Init.DataSize       = SPI_DATASIZE_16BIT;
Spi4Handle.Init.NSS            = SPI_NSS_SOFT;// NSS pin is configured as GPIO output for /CS
Spi4Handle.Init.Mode           = SPI_MODE_MASTER;// SPI4 as SPI Master
Spi5Handle.Init.Direction      = SPI_DIRECTION_2LINES_RXONLY;// only receive data
Spi5Handle.Init.NSS            = SPI_NSS_HARD_INPUT;
Spi5Handle.Init.Mode           = SPI_MODE_SLAVE;// SPI5 as SPI Slave
/*## Enable SPI4 as Master and SPI5 as Slave to Receive AD7606B Codes ###*/
__HAL_SPI_ENABLE(&Spi4Handle);
__HAL_SPI_ENABLE(&Spi5Handle);
while (SPI4_CNVCount < SPI4_CNVNum)
{
          CLR_CNV();
          SET_CNV();//AD7606B conversion start
          // wait for conversion finish, BUSY goes from high to low. Polling or interrupt mode
          while (BUSY == SET) {;}
          while (SPI4_WordCount < SPI4_CNVWordNum)// code number to read per conversion cycle
          {
                    CLR_CS();
                    *(__IO uint8_t *)&Spi4Handle.Instance->DR = 0;
                    while (__HAL_SPI_GET_FLAG(&Spi4Handle, SPI_FLAG_RXNE) != SET);
                    Delay_xus(1);// need half SCLK cycle delay for slow SCLK rate < 10MHz
                    SET_CS();
                    SPI_RxBuffer[RxBuf_Idn] = *(__IO uint16_t *)&Spi4Handle.Instance->DR;
                    SPI_RxBuffer[RxBuf_Idn+ADCSDO1_WordIdn] = *(__IO uint16_t \\
                    *)&Spi5Handle.Instance->DR;
                    RxBuf_Idn++;
                    SPI4_WordCount += 2;
          }
          SPI4_CNVCount++;
          RxBuf_Idn = SPI4_CNVCount * SPI4_CNVWordNum;
          SPI4_WordCount = 0;
}//while (SPI4_CNVCount < SPI4_CNVNum)
__HAL_SPI_DISABLE(&Spi4Handle);
__HAL_SPI_DISABLE(&Spi5Handle);
```

图12. SPI4配置为主机，SPI5配置为从机。

图13显示了AD7606B以240 kSPS运行时BUSY、SCLK、DOUTA和DOUB的数字接口截图。

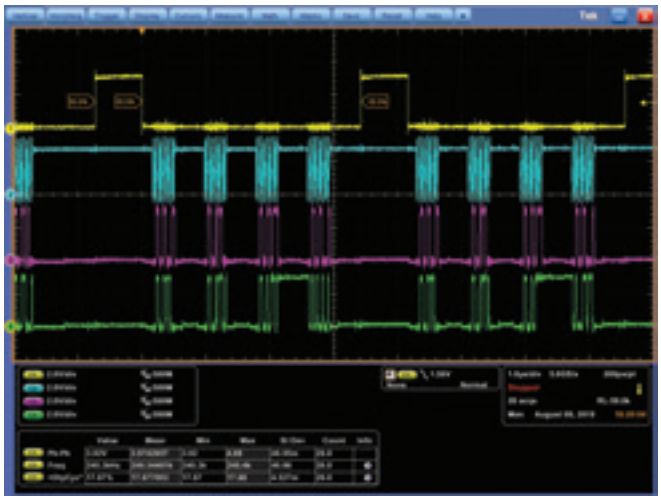

图13. AD7606B BUSY、SCLK以及DOUTA和DOUTB上的数据的示波器截图。

结论

本文讨论了使用微控制器SPI访问具有非标准SPI接口的ADC的方法。这些方法可以直接使用，也可以稍加调整即可控制ADC SPI；其可作为SPI主机使用，也可以与多条DOUT线配合使用以提高吞吐速率。

致谢

非常感谢应用工程师Mika Jiang和Yao Zhao，他们提供了有关STM32F429IDISCOVERY套件快速启动和固件调试工作的建议。

参考文献

Dhaker，Piyu。“SPI接口简介”。《模拟对话》，第52卷。2018年9月。

RM0090参考手册：STM32F405/415、STM32F407/417、STM32F427/437和STM32F429/439高级ARM® 32位MCU。STMicroelectronics，2019年2月。

STM32F427xx数据手册。STMicroelectronics，2018年1月。

UM1670用户手册：带STM32F429ZI MCU的Discovery套件。STMicroelectronics，2017年9月。

Usach, Miguel。AN-1248应用笔记：SPI接口。ADI公司，2015年9月。

作者简介

Steven Xie于2011年3月加入ADI北京分公司，担任ADI中国设计中心的产品应用工程师。他负责中国市场SAR型ADC产品的技术支持工作。在此之前，他曾在无线通信基站领域做过四年的硬件设计人员。2007年，Steven毕业于北京航空航天大学，并获得通信与信息系统硕士学位。联系方式：*steven.xie@analog.com*。