AN-1427
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com/cn

ADuCM3027/ADuCM3029中的SensorStrobe、超低功耗、时间同步传感器数据采样

简介

与一个精确时间同步的传感器精密采样，是建筑健康监控、可穿戴式设备、环境检测等各类无线传感器网络应用的要求。传感器数据采样由微控制器单元（MCU）支配。传统方法是利用MCU上的软件产生通用输入/输出（GPIO）脉冲，然后以特定间隔触发传感器收集数据。

传统方法有两个问题。一是涉及到相当大的软件开销，这会提高功耗。二是脉冲触发取决于MCU软件，因此可能随着时间推移而发生漂移。

本应用笔记介绍ADI公司的SensorStrobe™机制，利用它可实现低功耗、一致、同步的传感器数据采集。

ADuCM3027/ADuCM3029具备SensorStrobe机制。此机制支持与ADuCM3027/ADuCM3029 MCU相连的传感器实现时间同步的数据采样。

SensorStrobe解决了传统软件方法的问题，理由如下：

- 工作在休眠模式，功耗降低10倍以上。
- 设置之后无需软件干预。
- 脉冲触发机制独立于软件执行，即使在软件执行期间也能产生连续触发脉冲（且无漂移）。

本应用笔记使用一个示例设置，其中ADuCM3027/ADuCM3029 MCU连接到ADXL363加速度计，以证明利用SensorStrobe机制采集样本数据时功耗降低超过10倍。将SensorStrobe机制与非SensorStrobe的软件方法进行比较，这一降幅是很明显的。

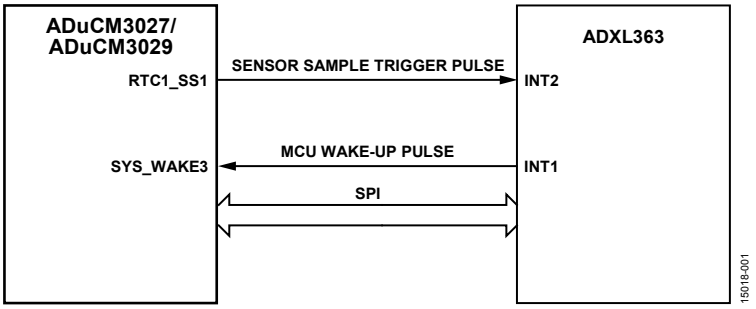

图1. ADuCM3027/ADuCM3029和ADXL363连接图

目录

修订历史

2017年3月—修订版0：初始版

SENSORSTROBE概述

SensorStrobe是一种以高效率、低功耗、内在同步的方式进行传感器采样的机制。ADuCM3027/ADuCM3029支持这种机制。SensorStrobe可以在ADuCM3027/ADuCM3029的活动、灵活（Flexi™）和休眠三种功耗模式下使用。

SensorStrobe机制允许ADuCM3027/ADuCM3029处于休眠模式（750 nA），同时传感器以固定间隔周期性收集数据。

SensorStrobe机制与ADXL363的外部触发特性相结合，以最低可能功耗收集传感器数据。

SensorStrobe是ADuCM3027/ADuCM3029中的实时时钟（RTC）的一种报警功能。通过此机制，ADuCM3027/ADuCM3029为ADXL363加速度计提供外部触发信号。触发信号位于RTC1_SS1（RTC SensorStrobe）引脚上，是通过ADuCM3027/ADuCM3029上的单一GPIO驱动出来的低频时钟源（32 kHz）的单周期、高电平脉冲。此脉冲是周期性的，确保传感器采样时间无变化，而其周期具有高度可配置性。

ADXL363特性

ADXL363是一款超低功耗、三传感器器件，集三轴微机电系统（MEMS）加速度计、温度传感器和模数转换器（ADC）输入于一体，用于同步采样外部信号。

ADXL363有一个512样本先进先出（FIFO）缓冲器用以存储传感器数据。这种大FIFO可节省系统功耗。在ADXL363将数据自主记录到FIFO缓冲器的同时，MCU可以处于休眠模式。

ADXL363配置为外部触发模式。ADuCM3027/ADuCM3029在RTC_SS引脚上产生这些触发脉冲。每个触发脉冲到来时，ADXL363便收集并存储数据到FIFO（最多512个样本，每样本两个字节）缓冲器中。

对ADXL363进行编程，当FIFO缓冲器达到480样本（每样本两个字节）的水印时，它便中断并唤醒MCU。使用水印特性可以让FIFO留下余地以供接收更多样本，与此同时，MCU唤醒并开始清空FIFO缓冲器。

ADXL363支持通过串行外设接口（SPI）进行寄存器读写访问。访问可以是单字节或多字节访问。实现FIFO缓冲器的目的是通过不限长度的多字节读取来连续读取连贯的样本。因此，一个FIFO缓冲器读指令便可清空FIFO缓冲器的全部内容。

而在其他加速度计中，每个读指令只能检索到一个样本。此外，ADXL363 FIFO缓冲器还可以利用ADuCM3027/ADuCM3029直接存储器访问（DMA）控制器清空。

利用SPI接口的读命令模式，ADuCM3027/ADuCM3029与ADXL363高效通信，通过减少SPI协议开销来降低系统整体功耗。

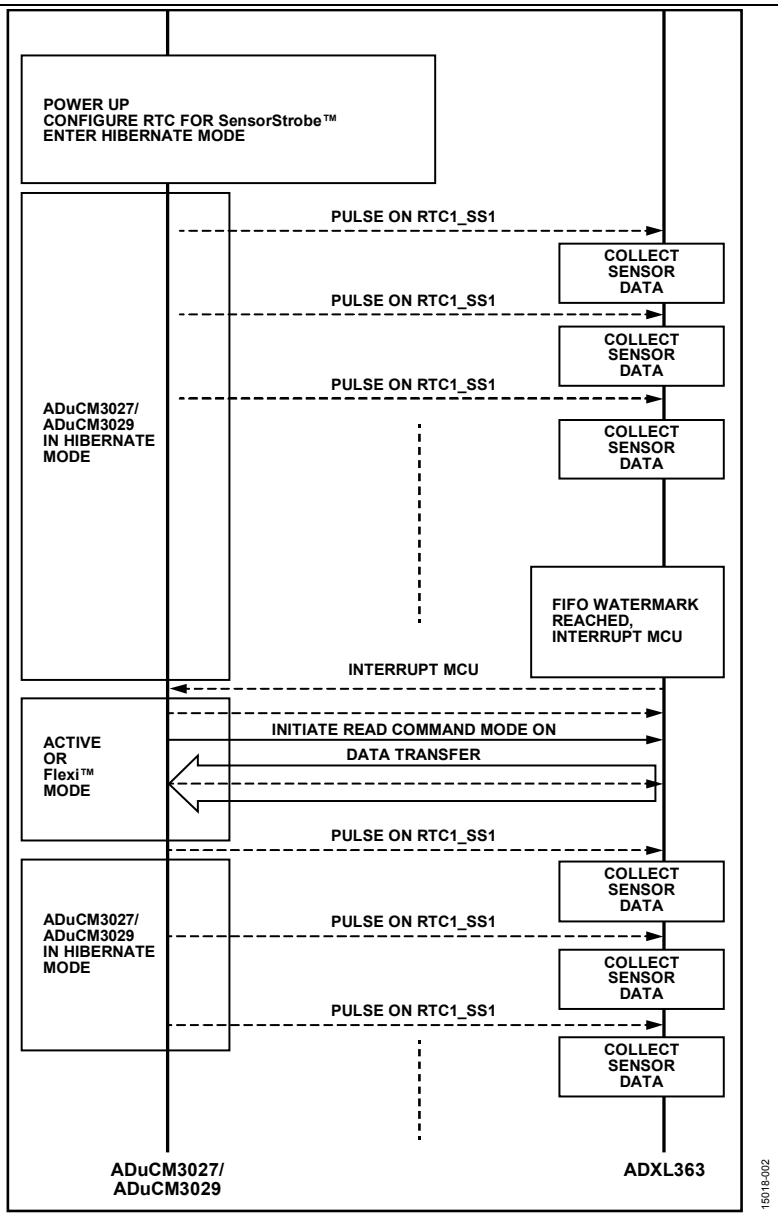
POWER UP
CONFIGURE RTC FOR SensorStrobe™
ENTER HIBERNATE MODE
PULSE ON RTC1_SS1
COLLECT SENSOR DATA
PULSE ON RTC1_SS1
COLLECT SENSOR DATA
PULSE ON RTC1_SS1
ADuCM3027/ ADuCM3029 IN HIBERNATE MODE
COLLECT SENSOR DATA
FIFO WATERMARK REACHED, INTERRUPT MCU
INTERRUPT MCU
ACTIVE OR Flexi™ MODE
INITIATE READ COMMAND MODE ON
DATA TRANSFER
PULSE ON RTC1_SS1
COLLECT SENSOR DATA
ADuCM3027/ ADuCM3029 IN HIBERNATE MODE
PULSE ON RTC1_SS1
COLLECT SENSOR DATA
PULSE ON RTC1_SS1
COLLECT SENSOR DATA
ADuCM3027/ ADuCM3029
ADXL363
15018-002

图2. 数据序列图

系统描述

我们构建了一个示例系统来说明使用SensorStrobe的优点。此系统包括一个EVAL-ADuCM3029 EZ-KIT万用表和电流源表。这些系统器件串联起来测量系统电流消耗。

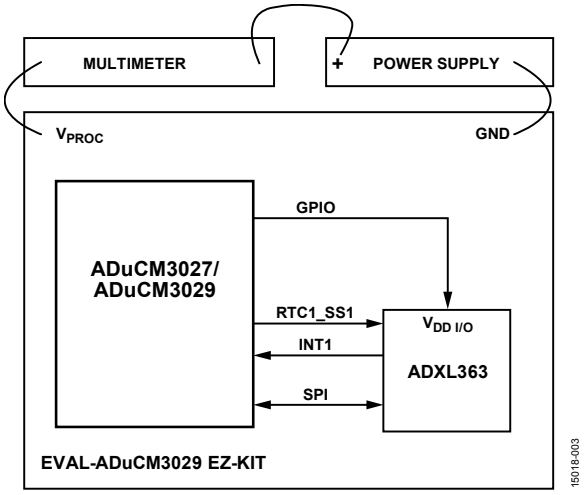

图3. 电流测量的系统连接

MCU和ADXL363之间的接口

表1. ADuCM3027/ADuCM3029 MCU和ADXL363之间的接口连接

MCU	ADXL363	描述
SPIn_MOSI	MOSI	SPI数据（从MCU到ADXL363）
SPIn_MISO	MISO	SPI数据（从ADXL363到MCU）
SPIn_CLK	SCLK	SPI时钟
SPIn_CSm	$\overline{CS}$	SPI片选
P2_11 (GPIO43)	INT2	RTC_SS—出现在GPIO43上
P2_01 (SYS_WAKE3)	INT1	ADXL363利用INT1引脚将MCU从休眠模式唤醒
P0_01	$V_{DD\ I/O}$	ADXL363由MCU GPIO P0_01供电

ADXL363配置为测量模式；当达到FIFO水印时，它会中断MCU。关于ADXL363配置，“软件概述”部分会做进一步说明。

使能ADuCM3027/ADuCM3029的SensorStrobe机制，并将ADuCM3027/ADuCM3029置于休眠模式。触发脉冲以128 Hz速率产生。

每收到一个脉冲，ADXL363便获取样本并将其存储在FIFO缓冲器中。当达到FIFO上水印时，ADXL363便通过SYS_WAKE3 (P2_01) 引脚中断ADuCM3027/ADuCM3029。

ADuCM3027/ADuCM3029利用读模式特性通过单个命令清空整个FIFO，使SPI协议开销最小。DMA控制器可以清空FIFO缓冲器，进一步降低MCU的工作时间和系统电流消耗。

通过SensorStrobe，ADuCM3027/ADuCM3029即便在休眠模式下也能在GPIO43引脚上产生触发脉冲。脉冲产生配置取决于RTC1寄存器和GPIO引脚复用。

在灵活模式下，DMA可以传输SPI数据，进一步降低系统功耗。

数据传输序列

MCU收集传感器数据分两个阶段进行。图4和图5显示了这些阶段中的信号活动情况。

首先，RTC1_SS1引脚充当外部触发信号，ADXL363收集样本并存储到FIFO缓冲器中。然后，ADXL363 FIFO缓冲器通过SPI读取内容。

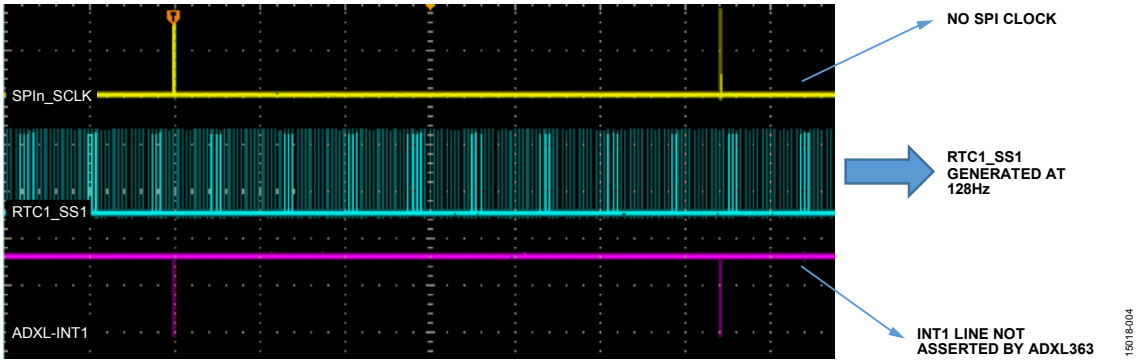

图4. 第一阶段——数据采集阶段：RTC_SS触发ADXL363

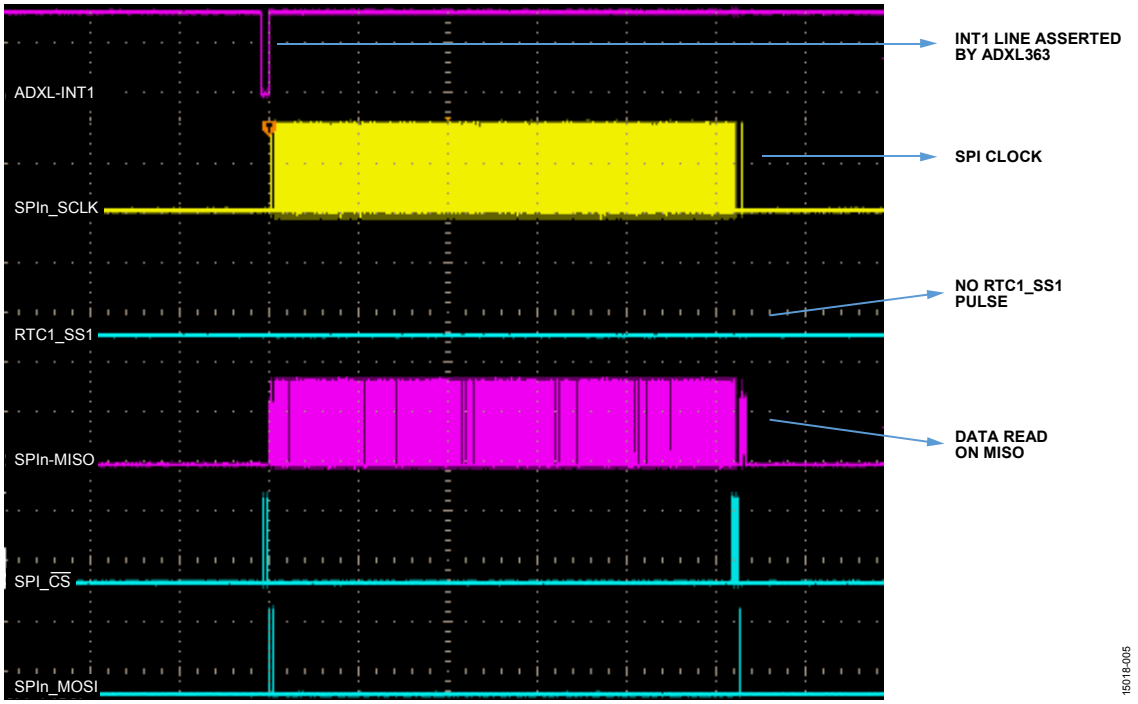

图5. 第二阶段——数据传输至MCU：通过SPI读取ADXL363 FIFO

软件概述

本部分介绍示例系统的软件流程。

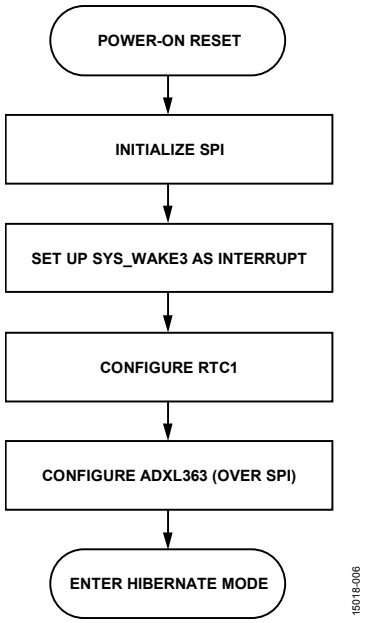

图6. 初始化和配置步骤

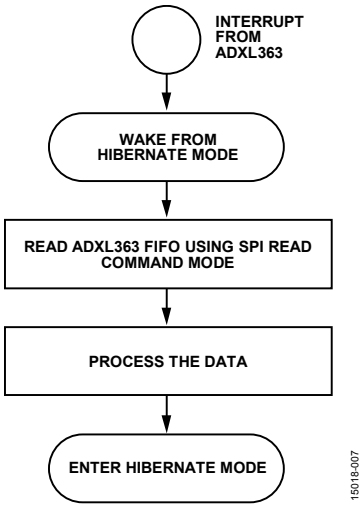

图7. 通过SPI读取FIFO（利用读命令）：从ADXL363读取数据

源代码片段

本部分提供关于配置和数据读取的参考源代码片段。

pREG_<module>_<name>是板支持包引用寄存器的方法。在*集成电源管理的ADuCM302x超低功耗ARM Cortex-M3 MCU硬件参考*中，同一寄存器称为<module>_<name>。

SensorStrobe的RTC配置

```
#define PRD_VAL 255
void SensorStrobe_Cfg()
{
     // SensorStrobe Pin Mux
     *pREG_GPIO2_CFG |= (0x3 << BITP_GPIO_CFG_PIN11);
     // Reset the RTC counter
     while(*pREG_RTC1_SR1 & 0x0180); //wait until sync
      *pREG_RTC1_SR0 = 0xFF;
     *pREG_RTC1_CR0 = 0;
     while(!(*pREG_RTC1_SR0 & 0x0080 )); //wait for sync

     // SensorStrobe configuration
     // Initial trigger and auto reload value = 255 RTC counts
     // RTC runs at 32KHz ideally, 1 RTC count = 30.7us
     while(*pREG_RTC1_SR5 != 0);
     *pREG_RTC1_SS1ARL = PRD_VAL;

     // SensorStrobe to be triggered after 255 RTC counts
     *pREG_RTC1_CR4SS = (1 << 9); //Enable Autoreload
     *pREG_RTC1_SS1 = PRD_VAL; // Initial Compare Value

     // Enable SensorStrobe
     *pREG_RTC1_CR3SS = 1;
     while(*pREG_RTC1_SR4 != 0x77FF);

     // Initialize the counter value to zero
     while(*pREG_RTC1_SR1 & 0x0600);
     *pREG_RTC1_CNT0 = 0;
     *pREG_RTC1_CNT1 = 0;
     while(!(*pREG_RTC1_SR0 & 0x0400));

     // Enable (start) the counter
     while(*pREG_RTC1_SR1 & 0x0180); //wait until sync
     *pREG_RTC1_SR0 = 0xFF;
     *pREG_RTC1_CR0 = 1;
     while(!(*pREG_RTC1_SR0 & 0x0080)); //wait for sync return;
}
```

ADXL363配置

在初始化阶段，ADXL363由ADuCM3027/ADuCM3029通过SPI配置。配置期间的每次SPI处理均包括来自ADuCM3027/ADuCM3029的2字节传输。第一个字节给出ADXL363寄存器地址，第二个字节给出要写入寄存器的值。

步骤显示在下面的adxl_write_reg函数源代码片段中。代码参见adxl_write_reg部分。

adxl_configure函数突出说明了配置ADXL363所需的寄存器写操作，示例应用中使用了这些操作。代码参见adxl_configure部分。

关于寄存器和设置的信息，参见ADXL363数据手册。

adxl_write_reg

```
void adxl_write_reg (unsigned char reg, unsigned char data)
{
    adxl_access(1);   //Enable chipselect
    spi_byte_tx(0x0A);       //Enable write
    spi_byte_tx(reg);        //Write register
    spi_byte_tx(data);       //Write data
    adxl_access(0);   //Disable chipselect
}
```

adxl_configure

```
void adxl_configure()
{
    // Softreset
    adxl_write_reg (0x1F,0x52);
    // Activity configuration
    adxl_write_reg (0x27,0x35);
    // FIFO configuration
    adxl_write_reg (0x28,0x00);
    adxl_write_reg (0x28,0xFF);
    // FIFO samples configuration
    adxl_write_reg (0x29,0xDF);
    // FIFO_watermark int
    adxl_write_reg (0x2A,0x84);
    // Filter control
    adxl_write_reg (0x2C,0x08);
    // Power control
    adxl_write_reg (0x2D,0x06);
}
```

ADXL363 FIFO读取

当ADXL363达到FIFO水印（960字节）时，会产生一个中断来唤醒MCU，然后通过SPI清空ADXL363 FIFO。

```
void ReadFifo_adxl()
{
    int j=0;
    // SPI2 configuration
    *pREG_SPI2_CTL = 0x0803;
    // Number of bytes to read
    *pREG_SPI2_CNT = 960;
    *pREG_SPI2_IEN = 7;
    // Enable Read command mode
    *pREG_SPI2_RD_CTL = 1;
    // Enable ADXl Chipselect adxl_access(1);
    // Flush out Rx FIFO
    while(*pREG_SPI2_FIFO_STAT & 0xF00)
          *pREG_SPI2_RX;
    // 0x0D written to kickstart the FIFO read from ADXL363
    *pREG_SPI2_TX = 0x0D;
    // Dummy read
    *pREG_SPI2_RX;
    // Number of samples
    while(j < 120)
    {if(*pREG_SPI2_STAT & BITM_SPI_STAT_RXIRQ)
      {*pREG_SPI2_STAT |= BITM_SPI_STAT_RXIRQ;
        i=0;
        while(i<8)
         {*(data_ref + (8 * j) + i) =
                              *pREG_SPI2_RX;
          i++ ; }
        j++;}
     }
    adxl_access(0);
    // Reset SPI
    *pREG_SPI2_RD_CTL = 0;
    *pREG_SPI2_IEN = 0;
    *pREG_SPI2_CTL = 0x0843;
}
```

系统功耗分析

ADuCM3027/ADuCM3029主机处理器负责以下操作：

- 在不同功耗模式（视需要而定，例如活动模式和休眠模式）之间切换。
- 配置RTC产生采样触发脉冲并发送给ADXL363。
- 控制与ADXL363的SPI通信。
- 存储来自ADXL363的原始数据（本应用笔记所述系统未实现结果数据的处理或发送）。

ADXL363传感器负责以下操作：

- 当ADuCM3027/ADuCM3029触发时，采样并存储原始传感器数据到FIFO缓冲器中。
- 响应主机处理器的SPI通信。
- 当FIFO缓冲器填满时，中断并唤醒主机处理器。

功耗测量

下述步骤说明如何监视系统的电流消耗：

1. 将应用程序代码载入MCU。
2. 将信号源的正端连接到万用表。
3. 将万用表的另一端连接到EVAL-ADuCM3029 EZ-KIT的JP6连接器。
4. 将信号源的GND连接到EVAL-ADuCM3029 EZ-KIT的GND。
5. 移除所有跳线。
6. 按电路板上的复位（RESET）按钮。
7. 监视源表和万用表上的电流消耗。

表2显示了ADuCM3027/ADuCM3029和ADXL363在本应用笔记所用不同功耗模式下的电流消耗。

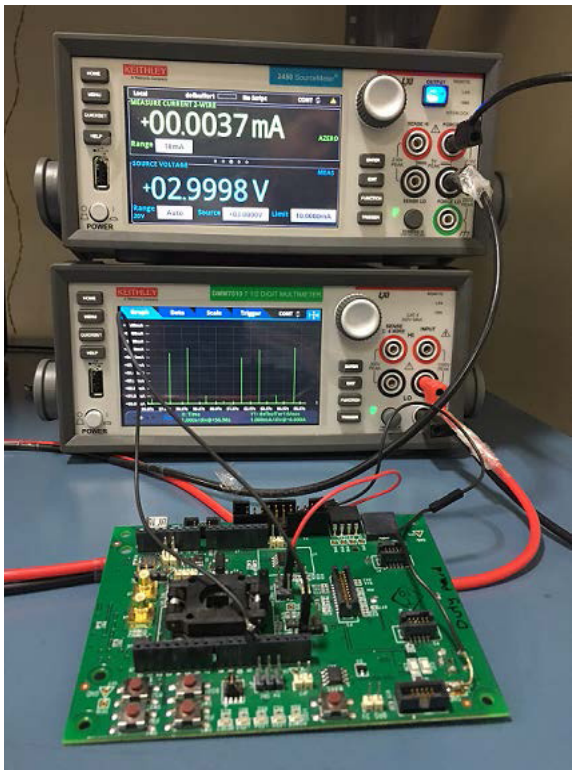

图8. 功耗测量的系统连接图

图9和图10分别显示了在传感器数据收集期间，使用和不使用SensorStrobe机制的功耗差异。所有测量都是在ADuCM3027/ADuCM3029 EVAL-ADuCM3029 EZ-KIT上进行的。

表2. ADuCM3027/ADuCM3029和ADXL363的功耗

器件	模式	电流消耗	描述
ADuCM3027/ADuCM3029	休眠模式	750 nA	保留通过8 kB随机存取存储器（RAM）选通的内核和外设的功耗，使能低频晶振。
	灵活模式	300 µA	内核通过时钟选通，但系统其余部分处于活跃状态。外设和存储器之间的DMA传输继续进行。
	活动模式	30 µA/MHz	全部系统功能都有效。
ADXL363	正常模式	1.8 µA	测量模式。

使用SensorStrobe的功耗测量

图9显示了使用SensorStrobe方法的演示系统的电流曲线。SensorStrobe支持主机处理器保持休眠模式，同时产生触发脉冲以驱动传感器捕捉和存储数据样本。测得的平均电流约为4.2 μA。

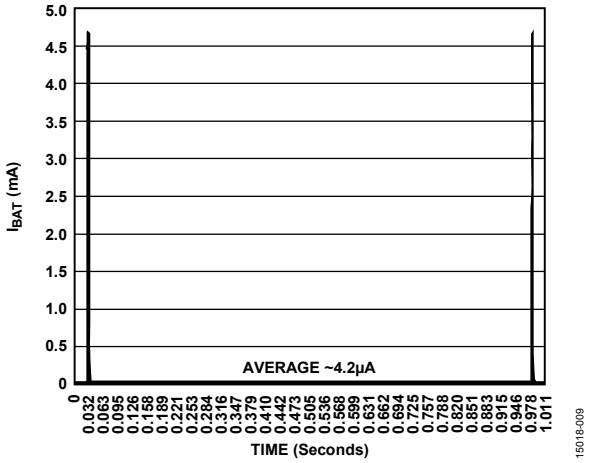

图9. 使用SensorStrobe的功耗测量

不使用SensorStrobe的功耗测量

图10显示了不使用SensorStrobe的样本采集系统的电流曲线。在没有SensorStrobe机制的情况下，主机处理器在RTC中断时唤醒并引起GPIO引脚跳变，从而产生触发脉冲。测得的平均电流约为75 μA。

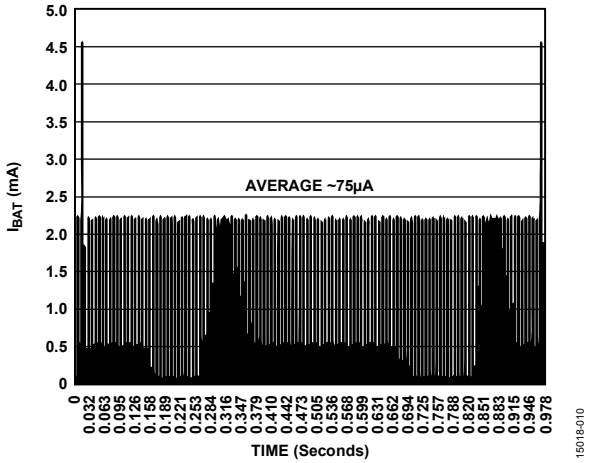

图10. 不使用SensorStrobe的功耗测量

结语

SensorStrobe机制使得传感器采样系统的电流消耗大幅降低。在本应用笔记所述的演示系统中，平均电流消耗从75 μA降至4.2 μA。

ADuCM3027/ADuCM3029的SensorStrobe机制可帮助设计人员利用ADXL363的低功耗特性，使功耗降低10倍以上。

ADuCM3027/ADuCM3029的SensorStrobe机制对超低功耗系统很有利，可降低整体电流消耗，延长系统的电池寿命。“结构健康监控（SHM)”部分、“医疗保健监护”部分和“环境检测”部分提及了一些可以获益于此的应用。

结构健康监控（SHM)

SHM中的传感器节点常常部署在无电源的结构中，例如桥梁、塔或偏远场所。定期更换电池是一个麻烦，会提高寿命周期的维护成本。电池寿命很长的传感器节点可显著降低维护成本。

医疗保健监护

医疗保健监护应用涉及到身体姿势、人员位置和病人生命体征的监测。此类监护设备通常是可穿戴式或植入式，降低系统电流消耗可提升设备的电池状况。

环境检测

各类环境检测应用，例如空气污染监测、森林防火检测、滑坡检测等，要求将传感器节点部署在偏远地区。这些偏远地区常常缺少供电线路。延长此类节点的电池寿命可降低维护成本。