ANALOG DEVICES

AN-1427
アプリケーション・ノート

ADuCM3027/ADuCM3029 での SensorStrobe による 超低消費電力、時刻同期、センサー・データ・サンプリング

はじめに

正確なタイム・ベースに同期させたセンサーによる高精度なサンプリングは、社会インフラの構造物モニタリング、ウェアラブル・デバイス、環境検出などの多様なワイヤレス・センサー・ネットワークのアプリケーションで要求されます。センサー・データのサンプリングは、マイクロコントローラ・ユニット（MCU）によって指示されます。従来の方法では、MCU に搭載されたソフトウェアによって汎用の入出力（GPIO）パルスを生成し、特定の間隔でセンサーをトリガすることでセンサーのデータが収集されます。

この従来の方法には、2 つの問題があります。この方法は、ソフトウェアのオーバーヘッドが大きく、電流の消費量が増えます。パルスのトリガは、MCU のソフトウェアによって変化するので、時間が経過するにつれてドリフトすることがあります。

このアプリケーション・ノートでは、低消費電力のセンサーによる一貫した同期データ・アクイジションを認識するメカニズムである SensorStrobe™ 機構について説明します。

SensorStrobe 機構は、ADuCM3027/ADuCM3029 で使用できます。この機構を使用すると、ADuCM3027/ADuCM3029 MCU に接続されたセンサーによる時刻同期されたデータ・サンプリングが可能になります。

以下に、SensorStrobe が従来のソフトウェア方式に起因する問題に対処できる理由を示します。

- 休止モードで稼働し、消費電流を 1/10 未満に削減します。
- セットアップ後のソフトウェアによる介入は不要です。
- パルス・トリガ機構を使用すれば、ソフトウェアの実行中でも連続したトリガを生成できます（ドリフトなし）。

このアプリケーション・ノートでは、ADXL363 加速度センサーに接続された ADuCM3027/ADuCM3029 MCU で構成されるセットアップを例に挙げ、SensorStrobe 機構を使用することでサンプル・データを収集しながら消費電流を 1/10 未満に削減できることを実証します。SensorStrobe ソフトウェア以外の方法と比較すると、SensorStrobe 機構を使用した場合の削減量は明らかです。

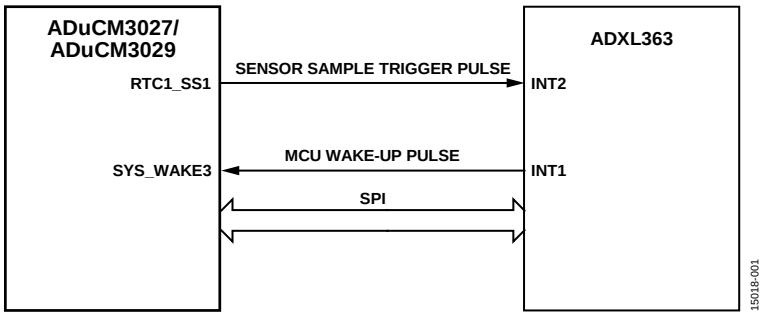

図 1. ADuCM3027/ADuCM3029 および ADXL363 の接続図

Rev. 0

アナログ・デバイセズ株式会社

本　　社／〒105-6891　東京都港区海岸 1-16-1　ニューピア竹芝サウスタワービル
電話 03（5402）8200
大阪営業所／〒532-0003　大阪府大阪市淀川区宮原 3-5-36　新大阪トラストタワー
電話 06（6350）6868

目次

はじめに 1
改訂履歴 2
SensorStrobe の概要 3
ADXL363 の機能 3
システムの説明 5
MCU と ADXL363 の間のインターフェース 5
データ転送シーケンス 6
ソフトウェアの概要 7
ソース・スニペット 8
ADXL363 の FIFO 読み出し 10
システム電力の分析 11
電力測定 11
まとめ 13
構造健全性モニタリング（SHM） 13
健全性モニタリング 13
環境検出 13

改訂履歴

3/2017−Revision 0: Initial Version

SENSORSTROBE の概要

SensorStrobe は、効率的かつ低消費電力で、本質的に同期された方法でセンサーによるサンプリングを実行するための機構です。ADuCM3027/ADuCM3029 では、この機構をサポートしています。SensorStrobe は、ADuCM3027/ADuCM3029 のアクティブ・モード、Flexi™ モード、休止モードで使用できます。

SensorStrobe 機構を使用すれば、一定の間隔でデータを収集しながら、ADuCM3027/ADuCM3029 を休止モード（750 nA）に移行できます。

SensorStrobe 機構と ADXL363 の外部トリガ機能を組み合わせることで、消費電力をできる限り抑えてセンサー・データを収集します。

SensorStrobe は、ADuCM3027/ADuCM3029 に搭載されたリアルタイム・クロック（RTC）のアラーム機能です。この機構では、ADuCM3027/ADuCM3029 が ADXL363 加速度センサーの外部トリガを提供します。このトリガは RTC1_SS1（RTC SensorStrobe）ピン上で発生し、ADuCM3027/ADuCM3029 に搭載された GPIO から駆動される、1 サイクル、高パルスの低周波数クロック源（32 kHz）です。このパルスは周期的に発生し、センサーによるサンプリング時間に変動がないので、周期を詳細に設定できます。

ADXL363 の機能

ADXL363 は、超低消費電力の 3 センサー・デバイスです。3 軸マイクロマシン・システム（MEMS）加速度センサー、温度センサー、そして外部センサーからの入力を A/D コンバータ（ADC）にて外部信号を同期変換します。

ADXL363 には、センサー・データを保存するため、サンプル 512 個分の先入れ先出し（FIFO）バッファが内蔵されています。FIFO の容量が大きいため、システム・レベルで電力を節約できます。ADXL363 が自発的に FIFO バッファにレコードを記録している間、MCU を休止モードに移行できます。

ADXL363 は、外部トリガ・モードに設定されています。ADuCM3027/ADuCM3029 は、RTC_SS ピンでこれらのトリガ・パルスを生成します。各トリガ発生時に、ADXL363 はデータを収集して FIFO（最大 512 個のサンプル、各 2 バイト）バッファに保存します。

ADXL363 は、FIFO バッファがサンプル数 480（各 2 バイト）のウォーターマークに到達すると、割込みを使用して MCU をウェイクアップさせます。ウォーターマーク機能を使用すると、FIFO バッファに空きスペースを確保しながら、MCU をウェイクアップさせ、FIFO バッファのドレインを開始します。

ADXL363 では、シリアル周辺機器インターフェース（SPI）経由でレジスタの読み出しと書き込みをサポートします。1 バイトまたは複数バイトでの読書きが可能です。FIFO バッファは、無限長の複数バイト読出しによって連続サンプルを中断せずに読み出すために実装されています。したがって、1 回の FIFO バッファ読み出し命令で、FIFO バッファのすべての内容をドレインできます。

他の加速度センサーでは、読み出し命令ごとに 1 個のサンプルしか取り出せません。さらに、ADXL363 FIFO バッファは、ADuCM3027/ADuCM3029 ダイレクト・メモリ・アクセス（DMA）コントローラを使用してドレインできます。

ADuCM3027/ADuCM3029 は、SPI インターフェースの読み出しコマンド・モードを使用して、ADXL363 と効率よく通信します。このモードでは、SPI プロトコルのオーバーヘッドを削減し、全体的なシステム消費電力を節約できます。

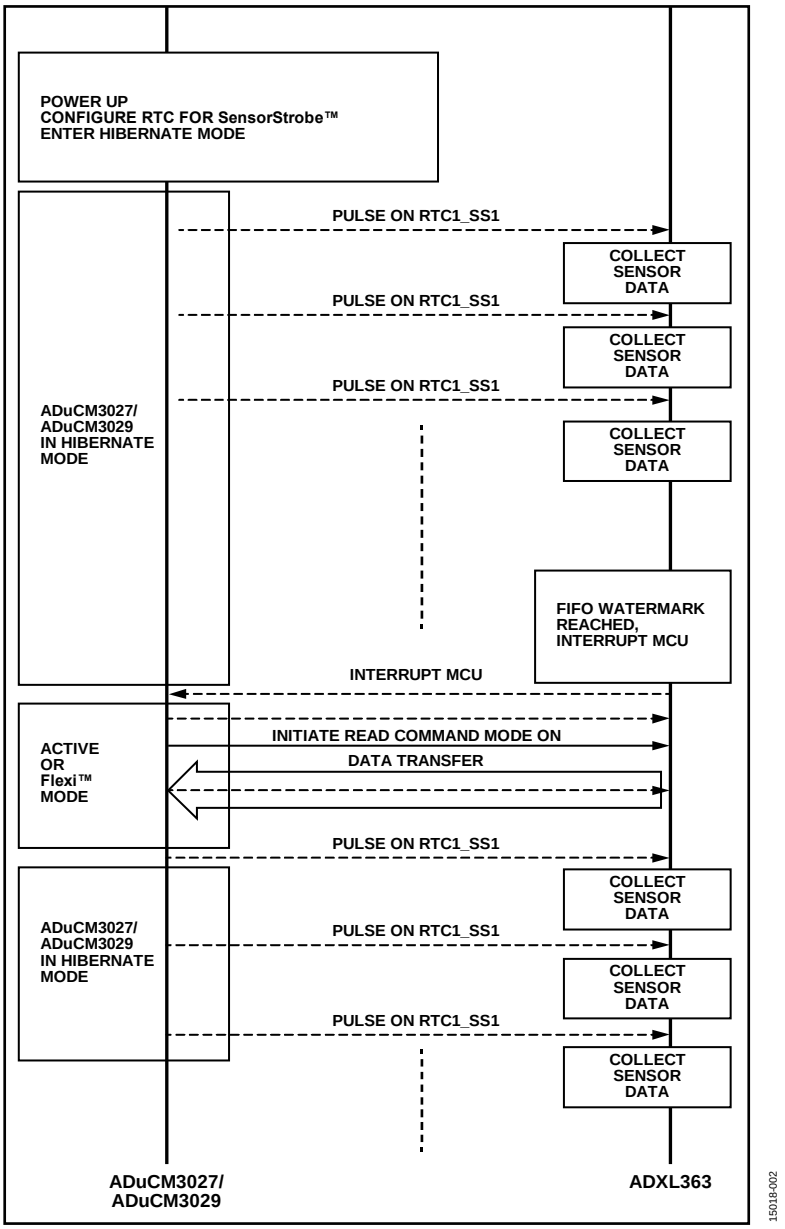

図 2. データ・シーケンス図

システムの説明

SensorStrobe の利点を実証するため、サンプル・システムを構築しました。このシステムは、EVAL-ADuCM3029 EZ-KIT マルチメータとソースメータで構成されています。システムの消費電流を測定するため、これらのシステム部品は直列に接続されます。

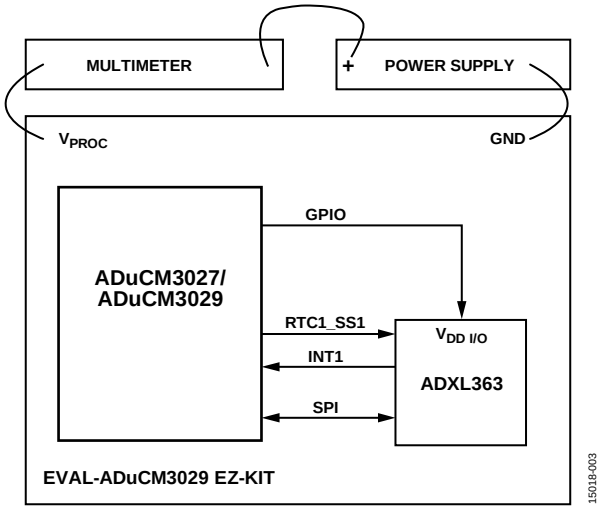

図 3. 電流測定のためのシステム接続

MCU と ADXL363 の間のインターフェース

表 1. ADuCM3027/ADuCM3029 MCU と ADXL363 の間のインターフェース接続

MCU	ADXL363	Description
SPIn_MOSI	MOSI	SPI data (from the MCU to the ADXL363)
SPIn_MISO	MISO	SPI data (from the ADXL363 to the MCU)
SPIn_CLK	SCLK	SPI clock
SPIn_CSm	$\overline{CS}$	SPI chip select
P2_11 (GPIO43)	INT2	RTC_SS−appears on GPIO43
P2_01 (SYS_WAKE3)	INT1	ADXL363 uses the INT1 pin to wake up the MCU from hibernate mode
P0_01	$V_{DD\ I/O}$	The ADXL363 is powered up by the MCU GPIO P0_01

ADXL363 は測定モードに設定され、FIFO がウォーターマークに到達すると、MCU に割込みを実行します。ADXL363 構成の詳細については、ソフトウェアの概要のセクションで説明します。

ADuCM3027/ADuCM3029 の SensorStrobe 機構はイネーブルで、ADuCM3027/ADuCM3029 は休止モードになります。トリガ・パルスは 128 Hz で生成されます。

各パルス発生時に、ADXL363 はサンプルを取得し、FIFO バッファに保存します。FIFO が上限のウォーターマークに到達すると、ADXL363 は SYS_WAKE3（P2_01）ピン経由で ADuCM3027/ADuCM3029 に割込みを実行します。

ADuCM3027/ADuCM3029 は、読み出しモード機能を使用して、1 回のコマンドで FIFO 全体を ドレインし、SPI プロトコルのオーバーヘッドを最低限に抑えます。DMA コントローラは、FIFO バッファをドレインすることで、MCU のアクティブ時間とシステム消費電力をさらに節約できます。

SensorStrobe は、ADuCM3027/ADuCM3029 をイネーブルにして、休止モードでも GPIO43 ピンでトリガ・パルスを生成します。RTC1 レジスタと GPIO ピンのマルチプレクスでパルス生成が設定されています。

Flexi モードでは、DMA が SPI データを転送して、システムの消費電力をさらに節約できます。

データ転送シーケンス

MCU によるセンサー・データの収集は、2 つのフェーズで発生します。図 4 と 図 5 に、これらのフェーズ中の信号の動作を示します。

まず、RTC1_SS1 ピンが FIFO バッファ内のサンプルを収集するため、ADXL363 の外部トリガとして動作します。次に、ADXL363 FIFO バッファが SPI 経由で内容を読み出します。

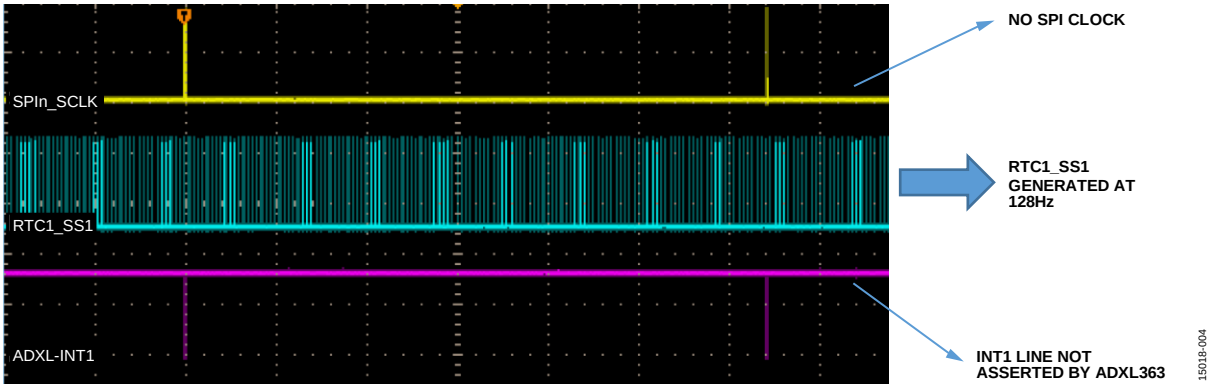

図 4. フェーズ 1 – データ・アクイジション・フェーズ: ADXL363 への RTC_SS トリガ

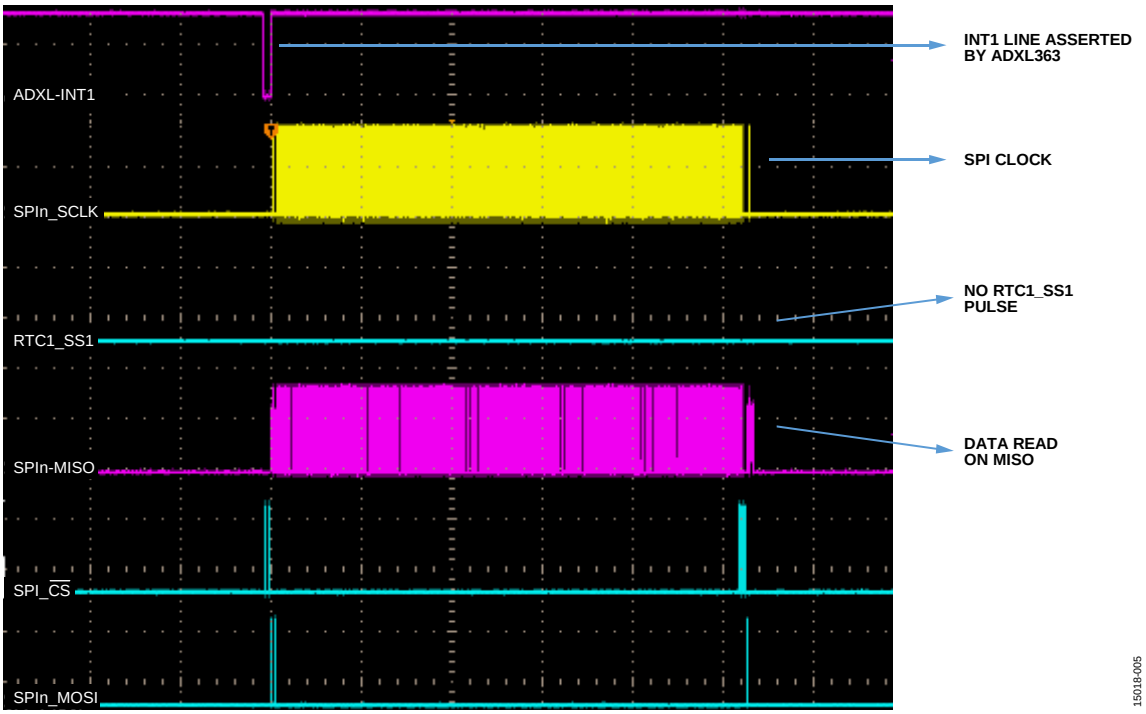

図 5. フェーズ 2 – MCU へのデータ転送:SPI 経由での ADXL363 FIFO の読み出し

ソフトウェアの概要

ここでは、サンプル・システムにおけるソフトウェア・フローを説明します。

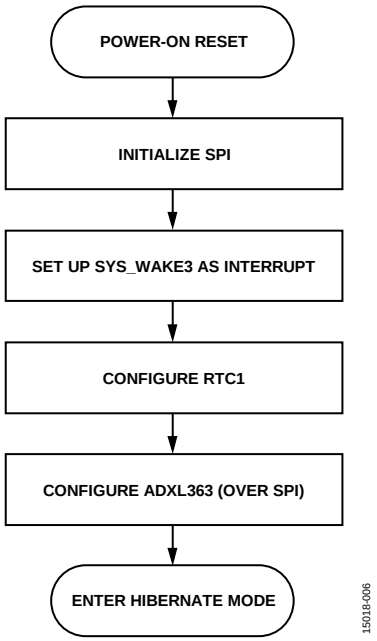

図 6. 初期化および構成の手順

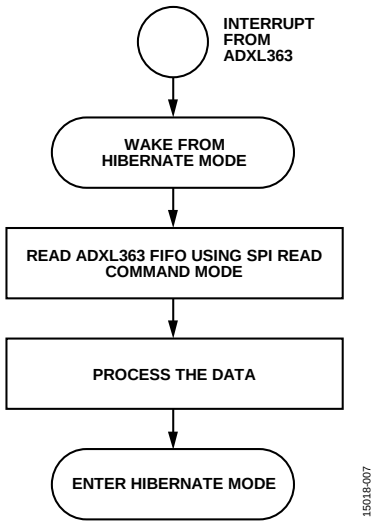

図 7. SPI 経由の FIFO 読み出し（読み出しコマンドを使用）: ADXL363 からのデータの読み出し

ソース・スニペット

ここでは、構成とデータの読み出しに使用するリファレンス・ソースのスニペットを提示します。

pREG_<モジュール>_<名前> は、レジスタを参照するボード・サポート・パッケージのメソッドです。ADuCM302x Ultra Low Power ARM Cortex-M3 MCU with Integrated Power Management Hardware Reference に記載されているのと同じレジスタが <モジュール>_<名前> の形式で参照されます。

SensorStrobe の RTC 構成

```
#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 の構成

初期化フェーズの間、ADXL363 は SPI 経由で ADuCM3027/ADuCM3029 によって設定されます。設定中の SPI トランザクションでは、ADuCM3027/ADuCM3029 から常に 2 バイトのデータが転送されます。最初のバイトは ADXL363 レジスタ・アドレスを示し、2 番目のバイトはレジスタに書出される値を示します。

この手順を次の adxl_write_reg 関数のコード・スニペットで示します。コードについては、adxl_write_reg のセクションを参照してください。

Adxl_configure 関数では、デモ・アプリケーションで使用したような ADXL363 の設定に必要なレジスタの書き込みが強調されています。コードについては、adxl_configure のセクションを参照してください。

レジスタと設定の詳細については、ADXL363 データシートを参照してください。

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 の FIFO 読み出し

ADXL363 が FIFO ウォーターマーク（960 バイト）に到達すると、
割込みがトリガされ、MCU がウェイクアップされ、SPI 経由で
ADXL363 FIFO がドレインされます。

```
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;
}
```

システム電力の分析

ADuCM3027/ADuCM3029 ホスト・プロセッサは以下の機能を実行します。

- 電力モード間の切替え（必要に応じてアクティブ・モードや休止モードなど）
- ADXL363 へのサンプル・トリガ・パルスを生成する RTC の設定
- ADXL363 との SPI 通信の制御
- ADXL363 からの生データの保存（結果データを処理または送信する機能は、ここで説明するシステムに実装されていません）

ADXL363 センサーは、以下の機能を実行します。

- ADuCM3027/ADuCM3029 によってトリガされた場合の生センサー・データのサンプリングと FIFO バッファへの保存
- ホスト・プロセッサからの SPI 通信への応答
- FIFO バッファがいっぱいになると、ホスト・プロセッサに割り込んでウェイクアップ

電力測定

次の手順では、システムの消費電流を監視する方法について説明します。

1. アプリケーション・コードを MCU にロードします。
2. ソースの正端子をマルチメータに接続します。
3. マルチメータの他端を EVAL-ADuCM3029 EZ-KIT の JP6 コネクタに接続します。
4. ソースの GND を EVAL-ADuCM3029 EZ-KIT の GND に接続します。
5. すべてのジャンパを取り除きます。
6. ボード上の RESET ボタンを押します。
7. ソースメータとマルチメータの消費電流を監視します。

表 2 に、このアプリケーション・ノートで使用する ADuCM3027/ADuCM3029 と ADXL363 の各電力モードにおける消費電流を示します。

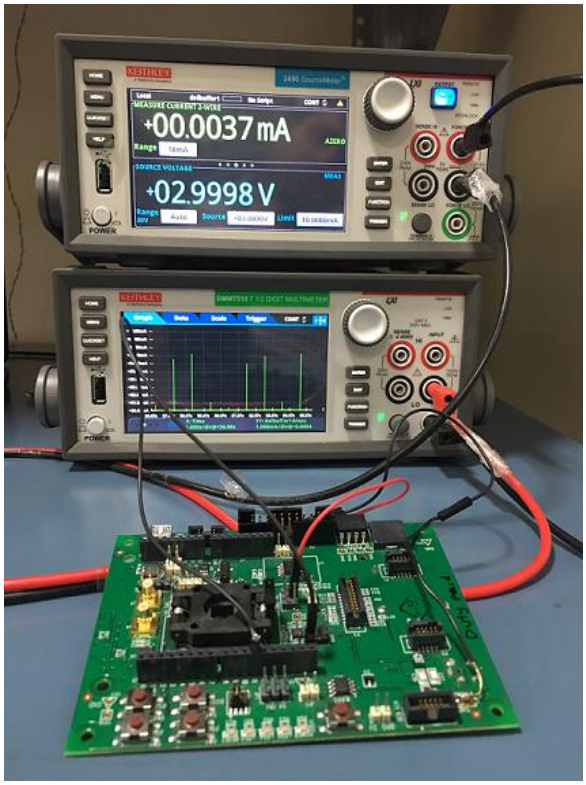

図 8. 電力測定のシステム接続図

図 9 と 図 10 に、SensorStrobe 機構のある場合とない場合におけるセンサー・データ収集中の消費電力の差を示します。すべての測定は、ADuCM3027/ADuCM3029 EVAL-ADuCM3029 EZ-KIT で実行されます。

表 2. ADuCM3027/ADuCM3029 および ADXL363 の消費電力

Device	Mode	Current Consumption	Description
ADuCM3027/ ADuCM3029	Hibernate mode	750 nA	Power of core and peripherals gated with 8 kB random access memory (RAM) is retained and the low frequency crystal enabled.
	Flexi mode	300 µA	The core is clock gated, but the remainder of the system is active. DMA transfers can continue between peripherals and memory.
	Active mode	30 µA/MHz	Full system is active.
ADXL363	Normal mode	1.8 µA	Measurement mode.

SensorStrobe がある場合の電力測定

図 9 に、SensorStrobe 方式を使用した場合のデモ・システムの電流プロファイルを示します。SensorStrobe を使用すれば、ホスト・プロセッサを休止モードに維持しながら、センサーを駆動してデータ・サンプルをキャプチャするトリガ・パルスを生成できます。平均電流の測定値は 4.2 µA です。

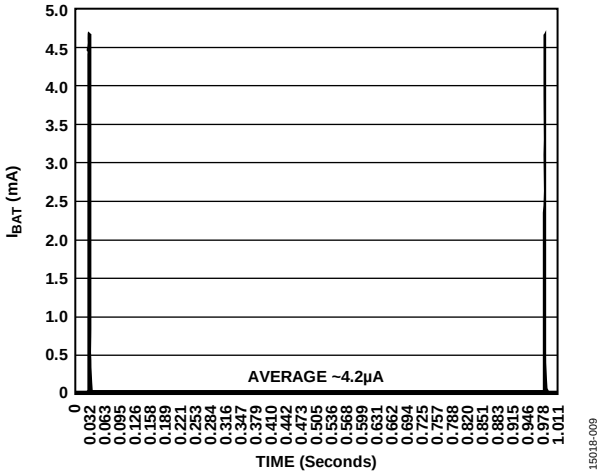

図 9. SensorStrobe がある場合の電力測定

SensorStrobe がない場合の電力測定

図 10 に、SensorStrobe を使用しない場合のサンプル・アクイジション・システムの電流プロファイルを示します。SensorStrobe 機構がない場合、ホスト・プロセッサは RTC 割込みでウェイクアップして GPIO ピンを切り替えることで、トリガ・パルスを生成します。平均電流の測定値は 75 µA です。

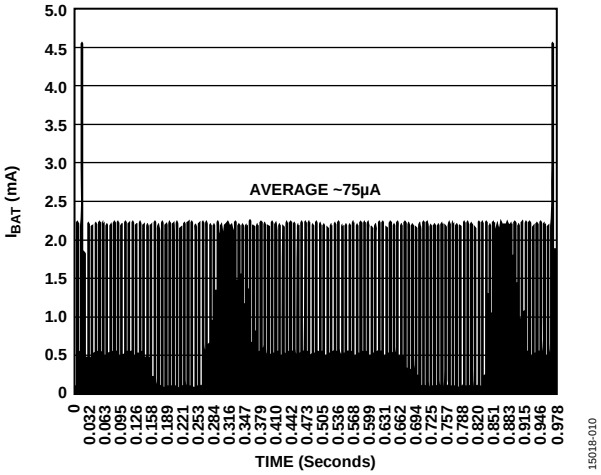

図 10. SensorStrobe がない場合の電力測定

まとめ

SensorStrobe 機構を使用すると、センサーのサンプル・アクイジション・システムの消費電流は大幅に減少します。このアプリケーション・ノートで説明しているデモ・システムでは、平均消費電流が 75 µA ～ 4.2 µA 減少しています。

設計者が ADuCM3027/ADuCM3029 の SensorStrobe 機構を利用すれば、ADXL363 の低消費電力性能を活かして消費電流を 1/10 未満に抑えることができます。

ADuCM3027/ADuCM3029 の SensorStrobe 機構を利用すれば、全体的な消費電流を削減することで、超低消費電力システムのバッテリ寿命が延びます。このメリットを活用できるアプリケーションについては、構造健全性モニタリング（SHM） セクション、健全性モニタリング セクション、環境検出 セクションで説明しています。

構造健全性モニタリング（SHM）

多くの場合、SHM のセンサー・ノードは、橋、塔、地理的に隔離された場所などの電源のない構造物で導入されています。定期的なバッテリの交換は故障の原因になり、生涯メンテナンスのコストが増えます。バッテリの寿命が長いセンサー・ノードでは、メンテナンス・コストが大幅に減ります。

健全性モニタリング

健全性モニタリング・アプリケーションには、身体の位置の測定、人物の位置、患者のバイタル・サインのモニタリングが含まれます。通常、これらのモニタリング・デバイスはウェアラブルまたはインプラントのデバイスです。システムの消費電流を減らすことで、デバイスに搭載するバッテリの健全性が向上します。

環境検出

空気汚染モニタリング、森林での火災検出、地滑り検出など、さまざまな環境検出アプリケーションでは、遠隔地にセンサー・ノードを配置する必要があります。多くの場合、これらの遠隔地では電力が供給されません。これらのノードでバッテリの寿命を延ばすと、メンテナンス・コストを節約できます。