ANALOG DEVICES

AN-1479 アプリケーション・ノート

ADuCM4050 の SPI フロー制御モード

はじめに

シリアル・ペリフェラル・インターフェース（SPI）は業界標準の同期シリアル・リンクで、A/D コンバータ（ADC）、D/A コンバータ（DAC）、デジタル・ポテンショメータ、不揮発性メモリ（NVM）、センサー、マイクロコントローラ・ユニット（MCU）などの他の SPI 互換デバイスへの全二重動作が可能です。これらのデバイスの中には、高速化、イベント通知、コマンド応答メカニズムの実現など、特定目的のための特別な機能を備えているものがあります。

ADuCM4050 の SPI 動作モードは強化されており、半二重動作とフロー制御オプションを柔軟に使用できます。これにより、SPI ブロックは、ペリフェラルに固有のこれらの特性のほとんどをハードウェアで自動的に使用できます。このため、ユーザ・コードが簡略化され、電力量効率を改善できます。

このアプリケーション・ノートでは、ADuCM4050 MCU で使用可能な SPI フロー制御モードについて説明し、使用例をいくつか示します。

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 10F
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 10F
電話 06（6350）6868
名古屋営業所／〒451-6038　愛知県名古屋市西区牛島町 6-1　名古屋ルーセントタワー 38F
電話 052（569）6300

目次

改訂履歴

5/2019—Revision 0: Initial Version

背景

フロー制御は、マスタとスレーブ間のデータ・フローの同期に必要です。ADuCM4050 MCU は、差別化機能であるフロー制御を SPI で提供します。読出しコマンド・モードでは、フロー制御を使用して複数のデータ・バイトを受信できます。

フロー制御では、SPIマスタとスレーブ間のデータ転送は、定期的なデータ読出しまたはオンデマンドのデータ読出しに関するアプリケーション条件に基づいて制御されます。

ADuCM4050 の SPI マスタは、次のモードのフロー制御をサポートします。

- ピンベースのフロー制御、SPI スレーブで制御。
  - MISO ピンを使用。
  - RDY ピンを使用。
- タイマーベースのフロー制御、SPI マスタで制御。

フロー制御モードについては、次のセクションで詳細に説明します。SPI フロー制御レジスタ（SPI_FLOW_CTL）のモード・フィールドは、フロー制御モードを 3 つのモードのいずれかに構成します。SPI_FLOW_CTL レジスタを図 1 に示します。

フロー制御機構は、ADuCM4050 が SPI マスタとして構成されている場合に限り使用できます。

ピンベースのフロー制御

専用の RDY ピンの使用

一部の SPI スレーブには、SPI マスタ（この場合は、ADuCM4050）の SPI_RDY ピンに接続する専用の RDY ピンがあります。SPI_RDY ピンは、各 SPI インスタンスに専用のピン（汎用入出力（GPIO）の代替機能）です。

例えば、ON Semiconductor®の CAT64LC40 シリアル・フラッシュは、専用の RDY ピンを使用して、SPI マスタにデータが使用できるかどうかを信号で通知します。

スレーブが専用の RDY ピンをサポートしていない場合、ADuCM4050 の RDY ピンは SPI スレーブの割込みピンに配線できます。スレーブは、RDY ピンを使用して、アクイジションとデータ処理が完了したことを示します。マスタは、このピンのアクティブなレベルが表示されるまで SPI クロックを提供しません。

RDY ピンがアサートされたときに読出しを実行するバイト数を構成できます。この構成を実行するには、SPI フロー制御レジスタ（SPI_FLOW_CTL）の RDBURSTSZ フィールドを設定します。MISO でこのバイトのバーストを受信した後、SPI マスタは、次の RDY ピンのアサーションを待機し、次のバイトのセットを受信します。このプロセスは、SPI カウント・レジスタ（SPI_CNT）のセットのすべてのバイトが受信されるまで繰り返されます。

読出しコマンド・モードを使用すると、最大 16 バイトを転送できます。この転送は、SPI 読出し制御レジスタ（SPI_RD_CTL）の TXBYTES フィールドを使用して構成されます。フロー制御を使用した場合に 1 回のバーストで受信されたバイト数は、SPI 読出し制御レジスタ（SPI_FLOW_CTL）の RDBURSTSZ フィールドで設定されます。ただし、受信バイトの合計数に上限は課せられません。

MISO ピンの使用

一部の SPI スレーブには専用の RDY ピンがありませんが、MISO ピンを再利用して MISO でデータを送信する準備が整ったことを SPI マスタに通知する手法をとっています。これは、AD7798 などの一部の ADC では一般的です。

ADuCM4050 SPI マスタは、MISO ラインのアクティブ・レベルの遷移まで待機し、これが検出されると、RDBURSTSZ + 1 個のバイトの読出しを実行して、次のアクティブ・レベルが MISO で検出されるまで待機状態に戻ります。

MISO/RDY ピンの極性は、SPI 読出し制御レジスタの RDYPOL フィールド（SPI_FLOW_CTL）を使用して構成できます。

タイマーベースのフロー制御

専用ピンがないスレーブから、マスタにデータが使用できるかどうかを通知するには、MCU で 16 ビット・タイマーを使用してデータの読出し中に待機状態を導入します。タイマーがトリガすると、マスタはバイトのバースト（RDBURSTSZ + 1）の読出しを実行し、タイマーを再起動します。タイマーは SPI クロック・レート（SCK）で出力され、タイマーのトリガを待機する SCK サイクルの数は、SPI_WAIT_TMR レジスタを使用して設定できます。例については、図 2 を参照してください。

このスキームを、フロー制御で SCK の待機と駆動に使用する場合は、最後の SCK エッジをサンプリング・エッジにする必要があります。待ち時間が終わったら、SCK 駆動エッジが次のデータ転送を実行します。

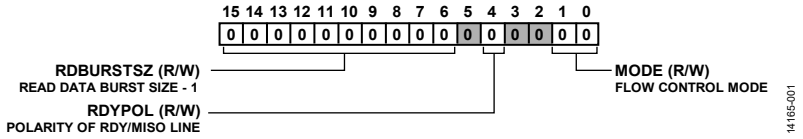

図 1. SPI_FLOW_CTL レジスタ

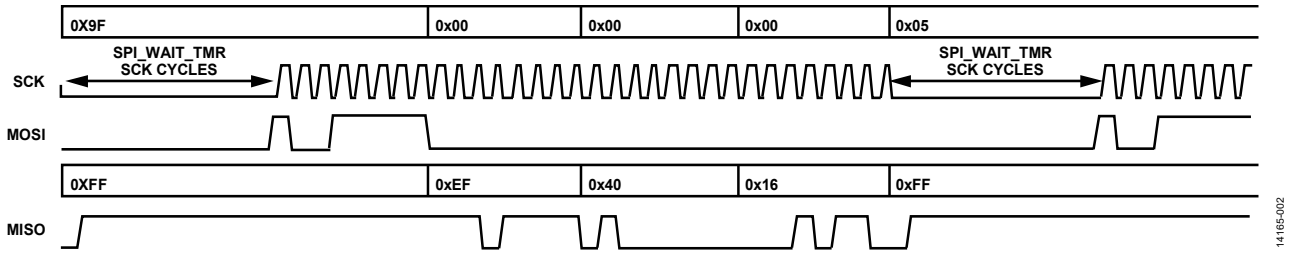

図 2. ソフトウェア・フロー制御とタイマー

ソース・コード

このセクションでは、ハードウェア・フロー制御モードを使用して、フロー制御機能によりユーザ・コードとシステム内の電力節約を簡素化する方法を説明します。

このセクションでは 2 つの例を示します。1 つはタイマーベースの概念を示し、もう 1 つはピンベースの例を示します。

タイマーベースのフロー制御の例

SPI フロー・オプションを介してデバイスを自動的に制御する第 1 の方法はタイマーです。タイマーベースのフロー制御のセクションで説明したように、SPI ブロックは連続する転送間に設定可能な遅延を追加します。この方法は、固定周波数のサンプリング・レートとデータ・レートでデータ・ストリームを提供するような ADC やセンサーでは一般的です。

SPI フローの方法は SPI_FLOW_CTL レジスタで示されます。この場合、SPI_FLOW_CTL.MODE でタイマー・オプションは 1 に設定します。

遅延は、PCLK 周波数と SPI_WAIT_TMR レジスタに依存します。このレジスタの値は、新しい SPI 動作の実行を待機する SPI_CLK サイクル数（PCLK/SPI_DIV）を指定します。

この他に考慮すべきオプションは次のとおりです。

- SPI_CTL.CONTINUOUS ビット・フィールド：すべてのデータが転送されるまで転送シーケンスを継続する。
- SPI_CNT レジスタ：転送するバイト数。
- SPI_FLOW_CTL.RDBURSTSZ ビット・フィールド：転送ごとに読出しバイト数から 1 を引いた数。

これらのフィールドの効果を図 5 に示します。

この例では、プログラムは固定数のワードを読み出し、タスクが完了するとプログラムが終了します。コードは、コード例 1 のセクションに示しています。

受信データは割込みハンドラで読み出されます。この場合、読出しバッファが一杯になり、values 変数が増加されます（コード例 2 を参照）。

結果を図 3 と図 4 に示します。これらの図は、あるバーストと次のバーストの間の遅延を制御する WAIT_TMR レジスタ値を変更することによって作成されています。

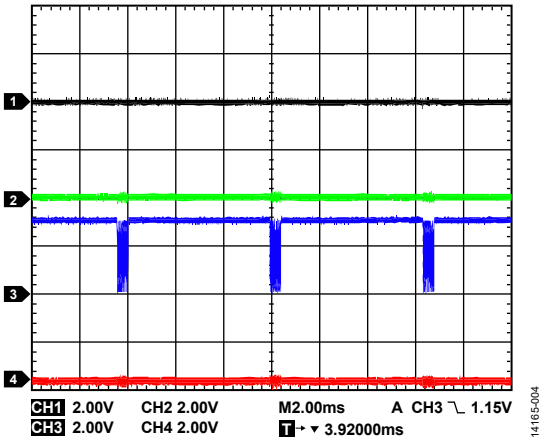

図 3. WAIT_TMR = 200

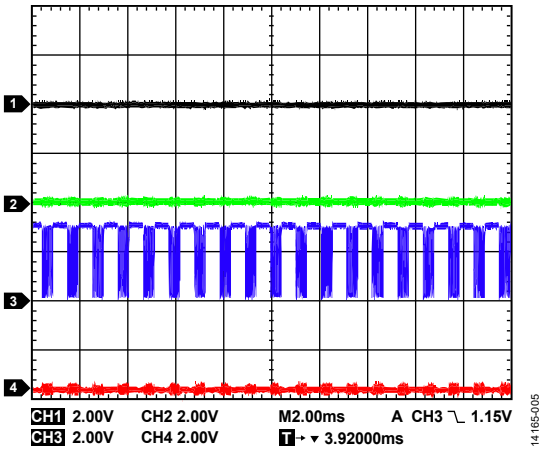

図 4. WAIT_TMR = 20

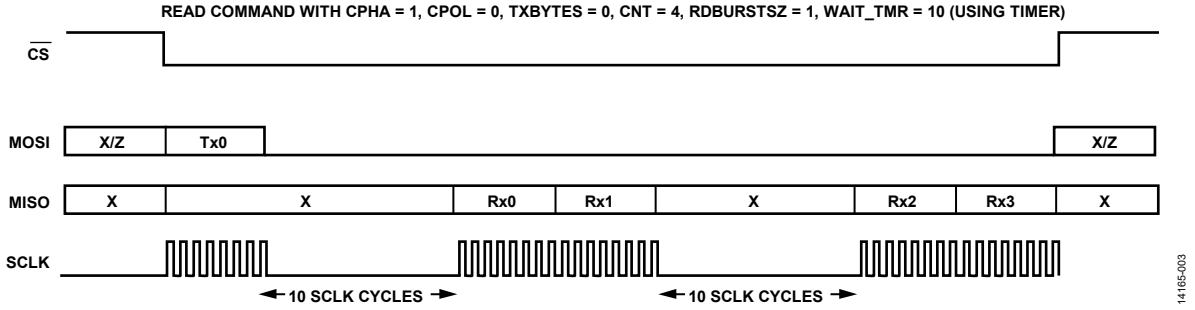

図 5. ADuCM4050 ハードウェア・リファレンス・マニュアルでの SPI フローの例

コード例 1

```
  #define NUM_VALUES      50
  volatile uint16_t values = 0;
  uint16_t buffer[NUM_VALUES] = {0};
[…]
  NVIC_EnableIRQ(SPI0_EVT_IRQn);

  pADI_SPI0->CS_OVERRIDE = 0;
  pADI_SPI0->CTL =  (1 << BITP_SPI_CTL_CON) |           //連続
                    (1 << BITP_SPI_CTL_CPOL) |          //極性
                    (1 << BITP_SPI_CTL_CPHA) |          //フェーズ
                    (1 << BITP_SPI_CTL_MASEN) |         //マスタ
                    (1 << BITP_SPI_CTL_SPIEN);          //イネーブル
  pADI_SPI0->IEN =  (1 << BITP_SPI_IEN_IRQMODE);        //2 バイトごとに tx 割込み
  pADI_SPI0->CNT =  NUM_VALUES * 2;                     //転送バイト数（サンプルあたり 2 バイト）
  pADI_SPI0->RD_CTL = (0 << BITP_SPI_RD_CTL_TXBYTES) |  //読出しコマンドでは、tx -1
                      (1 << BITP_SPI_RD_CTL_CMDEN);     //コマンド・モード
  pADI_SPI0->FLOW_CTL = (1 << BITP_SPI_FLOW_CTL_RDBURSTSZ) |    //バースト -1
                        (1 << BITP_SPI_FLOW_CTL_MODE);  //WAIT_TMR ベースのフロー制御
  pADI_SPI0->WAIT_TMR = 20;                             //待機するサイクル数

  tmp = pADI_SPI0->RX;                                  //転送を開始するためのダミー読出し

  while(values < NUM_VALUES);                           //すべてのサンプルまで待機
```

コード例 2

```
void SPI0_Int_Handler(){
  uint16_t aux = 0;

  if((pADI_SPI0->STAT & 0x20) == 0x20){         //TX 終了
    pADI_SPI0->STAT = 0x22;
  }else if((pADI_SPI0->STAT & 0x40) == 0x40){   //RX

    pADI_SPI0->STAT = 0xFFFF;                  //割込みをクリア

    aux = (pADI_SPI0->RX << 8);
    aux |= pADI_SPI0->RX;

    buffer[values++] = aux;
  }
}
```

ピンベースのフロー制御の例

SPI フローの第 2 の方法は、新しい転送を信号で通知する方法ですこの例では、MISO ラインがレディ信号として使用されています。

この例では、AD7798 ADC へのインターフェースを実装しています。この ADC は、サンプルを自動的に送信する自律モードで動作するように設定できます。新しいサンプルが準備できると、MISO ラインにより通知されます。連続モードを選択する設定は、0x5C の値を書き込むことで行います。これは、連続モードで次に読み出す対象がデータ・レジスタであることを COM レジスタで示します。

連続モードは、AD7798 からサンプルを連続的に読み出す手段です。各サンプルは 16 ビット・ワードで、ADC によって MISO ラインがグラウンドに接続された後でのみ使用できます。ワークフローを図 7 に示します。

SPI_FLOW_CTL.CTL_MODE ビット・フィールドに 3 を書き込むことにより、SPI フロー・オプションとして ADuCM4050 の SPI を MISO モードに設定する必要があります。この方法でも、同じレジスタ内の RDYPOL ビット・フィールドが考慮されます。この場合、ローの値がトリガとなるため、1 を書き込む必要があります。

各転送は 16 ビット幅ワードであるため、SPI_CTL.IRQMODE に 1 の値を設定する必要があります。これは、RX バッファに 2 バイトが入ったときに割込みを通知するのに使用されます。更に、SPI_FLOW_CTL.RDBUSTSZ が 1 になることは、転送ごとに 16 ビットクロック・サイクルが完了したことを示します。

最後に、デバイスを正しく設定するために、AD7798 に設定ワード（0x5C）を書き込む必要があります。

コードは、コード例 3 のセクションに示しています。

読出しは、割込みハンドラの関数でも実行できます（コード例 4 のセクションを参照）。

この結果、図 6 に示すようなデータ・ストリームになります。AD7798 ADC は新しいサンプルが使用できることを通知し、このデータを MISO ラインに出力します。各読出しで、ADuCM4050 はクロック（青色）を生成し、転送全体が終了するまでチップ・セレクト・ラインをローに保ちます（ピンク色）。MOSI ラインは初期設定後は必要ありません（緑色）。

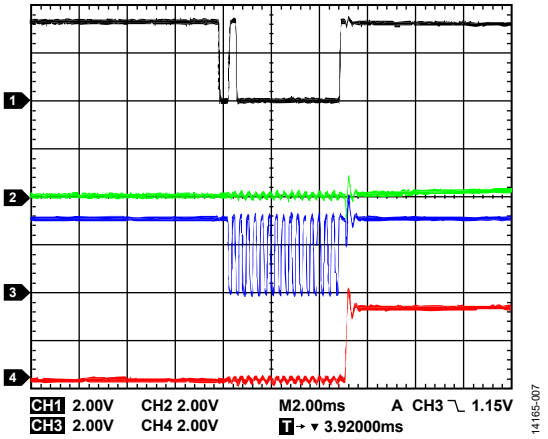

図 6. 連続モードでの AD7798 への SPI 転送

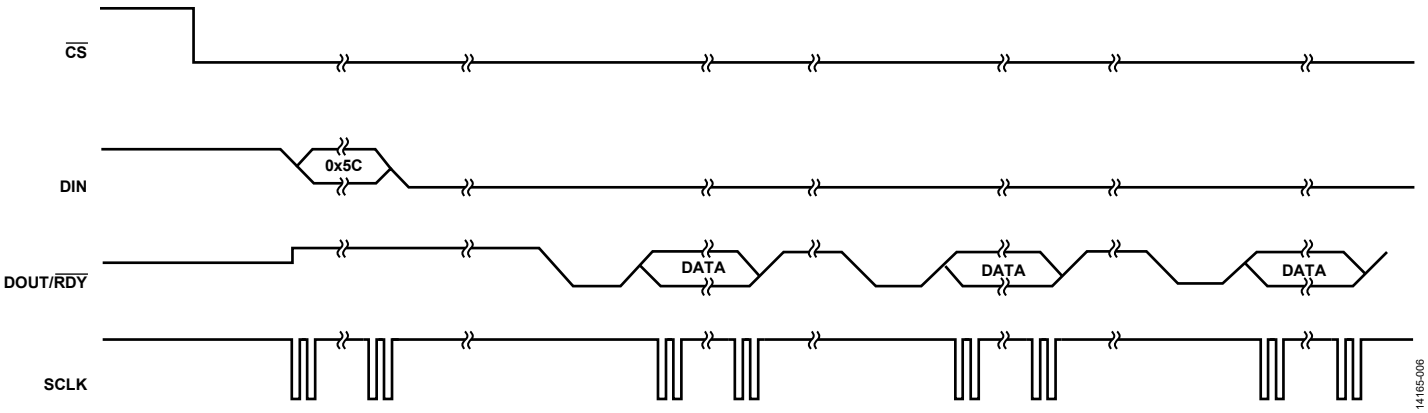

図 7. AD7798、連続モード

コード例 3

```
  NVIC_EnableIRQ(SPI0_EVT_IRQn);

  pADI_SPI0->CS_OVERRIDE = 0;
  pADI_SPI0->CTL =    (1 << BITP_SPI_CTL_CON) |                 //連続
                      (1 << BITP_SPI_CTL_CPOL) |                //極性
                      (1 << BITP_SPI_CTL_CPHA) |                //フェーズ
                      (1 << BITP_SPI_CTL_MASEN) |               //マスタ
                      (1 << BITP_SPI_CTL_SPIEN);                //イネーブル
  pADI_SPI0->IEN =    (1 << BITP_SPI_IEN_IRQMODE);              //2 バイトごとに tx 割込み
  pADI_SPI0->CNT =   NUM_VALUES * 2;                            //転送バイト数
  pADI_SPI0->RD_CTL = (0 << BITP_SPI_RD_CTL_TXBYTES) |          //読出しコマンドでは tx -1
                       (1 << BITP_SPI_RD_CTL_CMDEN);            //コマンド・モード
  pADI_SPI0->FLOW_CTL = (1 << BITP_SPI_FLOW_CTL_RDBURSTSZ) |    //バースト -1
                        (1 << BITP_SPI_FLOW_CTL_RDYPOL) |       //レディ信号の極性：ロー
                        (3 << BITP_SPI_FLOW_CTL_MODE);          //MISO ベースのフロー制御

  pADI_SPI0->TX = 0x5C;                                         //AD7798 を連続モードに設定
  tmp = pADI_SPI0->RX;                                          //転送を開始するためのダミー読出し

  while(values < NUM_VALUES);
```

コード例 4

```
void SPI0_Int_Handler(){
  uint16_t aux = 0;

  if((pADI_SPI0->STAT & 0x20) == 0x20){          //TX 終了
    pADI_SPI0->STAT = 0x22;
  }else if((pADI_SPI0->STAT & 0x40) == 0x40){    //RX

    pADI_SPI0->STAT = 0xFFFF;                    //割込みをクリア

    aux = (pADI_SPI0->RX << 8);
    aux |= pADI_SPI0->RX;

    buffer[values++] = aux;
  }
}
```

まとめ

読出しコマンド・モードやフロー制御などの ADuCM4050 SPI の種々の機能により、このデバイスは、SPI ペリフェラルで MCU の負担を軽減するバッテリ駆動システムに最適で、データ収集に個別に使用できます。

これらの特性をハードウェアで実装することで、遅延やライン操作などの一部の動作が設定不要になるため、ユーザ・コードを簡略化できます。また、MCU が不要になるため、ADuCM4050 の特性により電力量消費を高効率化できます。

ADuCM4050 の特性を、ダイレクト・メモリ・アクセス（DMA）およびフレキシ・モードと組み合わせて実装することにより、電力量効率と生産性を大幅に向上させることができます。