ANALOG DEVICES

AN-1159 アプリケーション・ノート

Cortex-M3 ベースの高精度アナログ・マイクロコントローラ（ADuCxxx ファミリー) の I²C 互換インターフェース

著者: Bridget Dowling

はじめに

このアプリケーション・ノートでは、アナログ・デバイセズの Cortex-M3 ベースの高精度マイクロコントローラ・ファミリー（ADuCxxx ファミリー）を使用した I²C（Inter-Integrated Circuit) 互換インターフェースのハードウェア・マスターとハードウェア・スレーブの実装について説明します。

このアプリケーション・ノートには、I²C インターフェースを使ってマスターとスレーブが相互に通信する方法を示すコード例も記載してあります。次の例が含まれています。

- マスターの送信と受信
- スレーブの送信と受信
- スレーブ・モードでの DMA 転送（送信と受信）
- マスター・モードでの DMA 転送（送信と受信）

コンパニオン・コードについては、http://www.analog.com/jp/の AN-1159 コンパニオン・コード zip ファイルをご覧ください。

I²C バスの主な特長は、

- シリアル・データ・ライン（SDA）とシリアル・クロック・ライン（SCL）の 2 本のバス・ラインのみが必要です。これらのラインは双方向であるので、マスターとスレーブはトランスミッタまたはレシーバとして動作可能です。
- I²C マスターは複数のスレーブ・デバイスと通信することができます。各スレーブ・デバイスは固有のアドレスを持つため、マルチスレーブ環境においても常にマスターと各スレーブとの間で固有の関係が成立します。
- 調停機能を使うと、同じ I²C バス上に複数のマスターが存在できます。
- マスターとスレーブは、最大 400 kbps で送受信できます。
- 内蔵フィルタ機能により SDA ラインと SCL ラインで 50 ns 以下のスパイクを除去し、データ・インテグリティを維持します。

複数デバイスの代表的な I²C 接続を図 1 に示します。

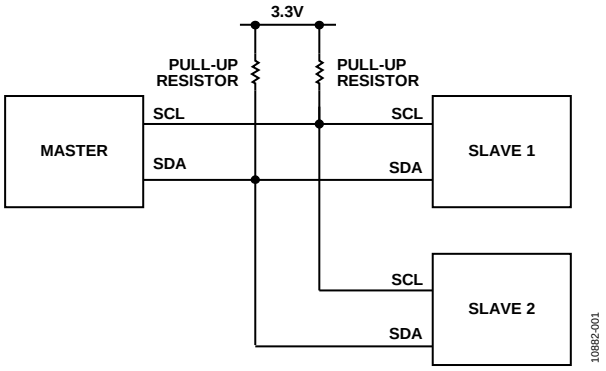

図 1.シングル・マスター・マルチ・スレーブ I²C のブロック図

Rev. A

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

改訂履歴

12/13—Rev. 0 to Rev. A

Changes to Figure 16 13
Changes to Figure 19 15

9/12—Revision 0: Initial Version

I²C の基礎

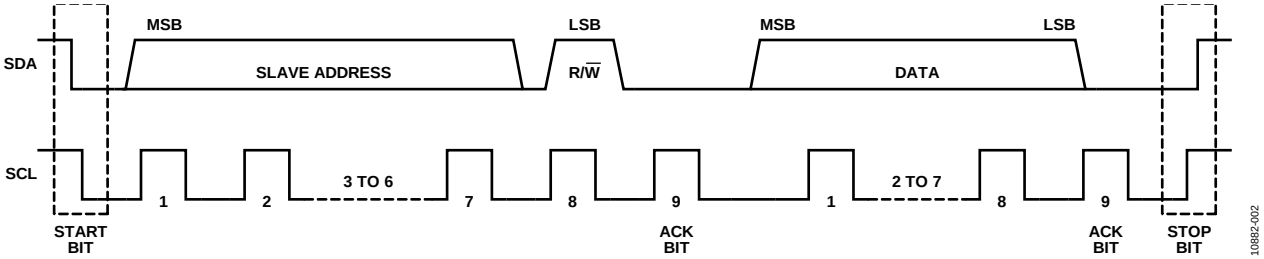

図 2.代表的な I²C 転送シーケンス

I²C インターフェースの概要

I²C は、Philips Semiconductors 社（現在の NXP Semiconductors 社）が開発した 2 線式シリアル通信システムで、複数のマスターと複数のスレーブを 2 本の線（SCL と SDA）で接続することができます。I²C インターフェースでは、少なくとも 1 つのマスターと 1 つのスレーブが存在する必要があります。

SCL 信号は、マスターとスレーブの間のデータ転送を制御します。転送される各データ・ビットに対して 1 個のクロック・パルスを発生します。SCL 信号は常にマスターからスレーブへ送信します。ただし、スレーブは次の送信を開始できない場合、このラインをロー・レベルにすることができます。これはクロック・ストレッチと呼ばれます。

SDA 信号は、データの送信または受信に使います。SDA 入力は、SCL がハイの間安定している必要があります。SCL がハイの間に SDA ラインが変化すると、スタート条件またはストップ条件と見なされます。代表的な転送シーケンスを図 2 に示します。

スタート条件

I²C インターフェースの代表的なデータ転送シーケンスは、スタート条件で開始されます。スタート条件は、単に SCL をハイに保ったまま SDA ラインをハイからローへ変化させます （図 3 参照）。スタート条件は常にマスターが発行します。スタート（およびストップ） 条件は、SCL ラインがハイの間に SDA ラインを変化させてもよい唯一のときです。通常のデータ転送では（スレーブ・アドレシングも含む）、SDA ライン上のデータは、SCL ラインがハイの状態で安定している必要があります。

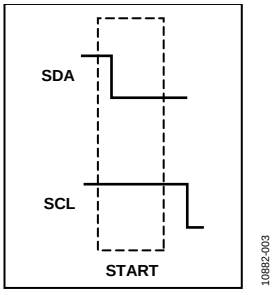

図 3. I²C のスタート条件

ストップ条件

データ転送シーケンスは、ストップ条件で停止します。ストップ条件は、SCL がハイの間に SDA ラインがローからハイへ変化することであると定義されます （図 4 参照）。

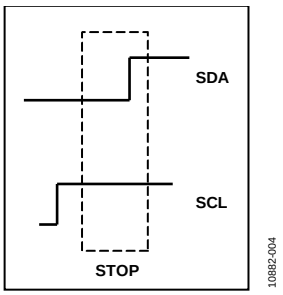

図 4.I²C のストップ条件

ストップ条件は常にマスターが生成します。データ・シーケンスが終了した場合、またはスレーブ・デバイスからマスターが NACK を受信した場合に、マスターはストップ条件を送信します。ストップ条件を受信すると、スレーブ・デバイスがリセットされ、スレーブ・アドレス待ちの状態に戻ります。

ストップ条件で割込みを発生するように I²C インターフェースを設定することができます。

スレーブ・アドレス

スタート条件の後、マスターは 8 個の SCL パルスと一緒に、最上位ビット（MSB）を先頭にして SDA ラインに 1 バイトを送信します。このバイトの最初の 7 ビットは、7 ビットのスレーブ・アドレスです。この 7 ビット・アドレスがスレーブ・デバイスのアドレス （すなわち、4 つのスレーブ・アドレスのうちの 1 つ）に一致した場合にのみ、スレーブはマスターに応答します。8 番目のビットすなわち最下位ビット （LSB）は、R/$\overline{W}$ ステータス・ビットです。R/$\overline{W}$ ステータス・ビットはメッセージの方向を指定します。このビットがローの場合、マスターは選択したスレーブにデータを書込みます。このビットがセットされていると、マスターはスレーブからのデータ受信を期待します。いずれのケースもマスターがクロックを発生します。

スレーブが正しいアドレスを受信すると、すなわちマスターからの 7 ビットの MSB が、I2CADR0 メモリ・マップド・レジスタ（MMR）の 7 ビットの MSB と一致すると、スレーブは有効な ACK を返すため SCL ラインをローにして、I2CSSTA のフラグをセットします。

スレーブはすべての I²C スレーブ・アドレシングの操作を自動的にハードウェアで行いますが、スレーブ・アドレスを正しく出力するのはマスターの役割です。

アクノリッジ（ACK/ナック（NACK）

スレーブ・アドレスがマスターから送られたアドレスに一致すると、スレーブは自動的にアクノリッジ（ACK）を送信します。そうでない場合は、ナック（NACK）を送信します。ACK は、9 番目のクロック・パルスのときの SDA ラインのロー・レベルとして検出されます。NACK は、9 番目のクロック・パルスのときの SDA ラインのハイ・レベルとして検出されます（図 5 参照）。

データ転送時、ACK または NACK は常にレシーバが生成しますが、ACK に必要とされるクロック・パルスは常にマスターが生成することに注意してください。トランスミッタは ACK クロック・パルスの間、SDA ラインを解放する（ハイにする） 必要があります。有効な ACK の場合、レシーバは SDA ラインをローにする必要があります。

受信時、ACK と NACK は各バイトの最後に、ハードウェアにより自動的に生成されます。

マスターがスレーブ・レシーバから NACK を受信した場合（スレーブがスレーブ・アドレスまたは送信されたデータに応答しない場合）、マスターはストップ条件を生成し、転送を停止する必要があります （データ転送のセクション参照）。

マスター・レシーバは、スレーブから送信された最終バイトの後にナック （NACK） を生成して、データ・シーケンスの終わりをスレーブ・トランスミッタに通知する必要があります。スレーブが NACK を受信すると、スレーブは SDA ラインを解放して、マスターがストップ条件を生成できるようにします。

NACK を強制するように、スレーブを設定することもできます。

データ転送

I^2C 割込みサービス・ルーチン （ISR）またはポーリング方式では、スレーブはマスターから送信された R/$\overline{W}$ ビットの状態に応じて送信または受信を決定します。次にスレーブは、マスターから送信された各クロックでビットを送信または受信します。マスターとの間でスレーブがデータを送信/受信するために 9 個のクロック （データに 8 個と ACK に 1 個）を供給するのはマスターの役割です。スレーブによって有効なデータ・バイトが送信または受信されるごとに I^2C 割込みビットがセットされます。

ここでも、スレーブ・トランスミッタ/マスター・レシーバ・システムでは、スレーブから送信された最後のバイトの後でマスターが NACK を送信して、マスターがデータ・シーケンスの終わりをスレーブへ通知する必要があることに注意してください。スレーブが NACK を受信すると、スレーブは SDA ラインを解放して、マスターがストップ条件を生成できるようにします。

マスターがデータ転送をアボートする場合、またはバス上の別のマスターのデータ転送に割込む場合、マスターはスタート条件を送信し、その後にストップ条件を送信して、これを行うことができます。

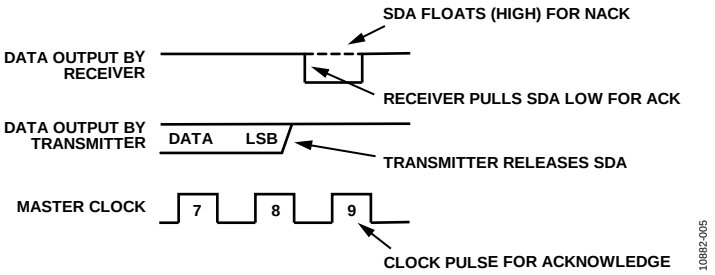

図 5.I^2C バス上のアクノリッジ（ACK）とナック（NACK）

繰り返しスタート条件

繰り返しスタート条件は、ストップ条件が送信されることなく 2 つ目のスタート条件がスレーブへ送信されたときに発生します。これにより、マスターはバス制御を放棄することなく R/$\overline{W}$ ビットを変更することにより、転送方向を変更できるようになります。

転送シーケンスの例を図 6 に示します。これは一般に、デバイスへ送られた最初のデータが読出し対象のレジスタ・アドレスを設定する場合に使用されます。“繰り返しスタート + スレーブ・アドレス” が受信されると、割込みが発生します。I2CxSSTA MMR のステータス・ビットを使用することにより、これを“スタート + スレーブ・アドレス” から区別することができます。

図 6 に示すシーケンスを使用することにより、ADuCxxx ファミリーはマスター・モードで直接 I²C 繰り返しスタート・シーケンスを発生することができます。

クロック・ストレッチ

I²C 通信では、マスター・デバイスがクロック速度を決定します。RS-232 とは異なり、I²C バスではマスターとスレーブが既定のボー・レートに正確に同期しなくて済むように、専用のクロック信号を使用しています。

ただし、マスターによって与えられるクロック速度で I²C スレーブが動作できないために、少し速度を下げる必要がある場合があります。これは、クロック・ストレッチと呼ばれるメカニズムで実行されます。

I²C スレーブがバス速度を下げる必要がある場合、クロックをロー・レベルに保持することができます。これに対して、マスターはクロック信号をハイ状態に解放した後に、クロック信号をリードバックして、ラインが実際にハイになるまで待つ必要があります。

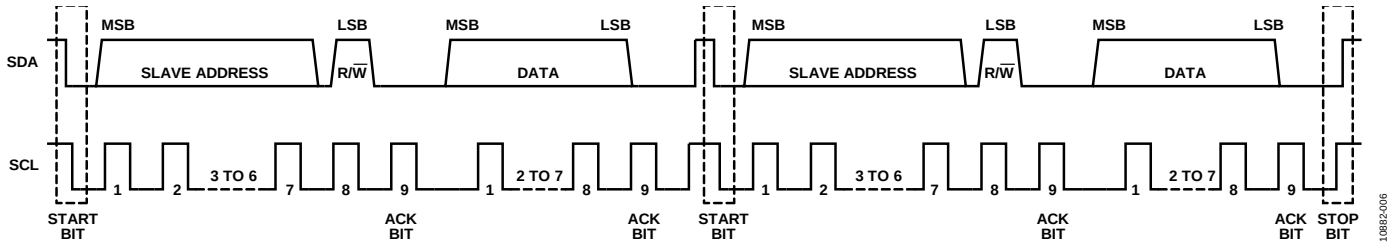

図 6. I²C 繰り返しスタート・シーケンス

```
I2cFifoFlush(MASTER, ENABLE);
I2cFifoFlush(MASTER, DISABLE);
NVIC_DisableIRQ(I2CM_IRQn);
I2cMWrCfg(0xA0);      // configure to send slave address
I2cTx(MASTER, ucStartAddress);  // send slave address
while ((I2cSta(MASTER)& I2CMSTA_TXFSTA_EMPTY) != I2CMSTA_TXFSTA_EMPTY){} {}
                      // wait for the Tx FIFO to empty
I2cMRdCfg(0xA0, ucLength , DISABLE);
                      // sends out the read condition, this function sets the read bit
NVIC_EnableIRQ(I2CM_IRQn);
```

代表的な I²C タイミング図

表 1 と表 2 に、Cortex-M3 ベースの高精度アナログ・マイクロコンバータに実装されている I²C バスの代表的なタイミングを示します。各 I²C バス・ラインの容量負荷 Cb は、I²C バス仕様によると最大 400 pF です。

SCL と SDA の内部プルアップは、ソフトウェアからディスエーブルする必要があることに注意してください。

表 1. 高速モード（400 kHz）での I²C タイミング

Parameter	Description	Min	Max	Unit
t_L	Clock low pulse width	1300		ns
t_H	Clock high pulse width	600		ns
t_{SHD}	Start condition hold time	600		ns
t_{DSU}	Data setup time	100		ns
t_{DHD}	Data hold time	0		ns
t_{RSU}	Setup time for repeated start	600		ns
t_{PSU}	Stop condition setup time	600		ns
t_{BUF}	Bus-free time between a stop condition and a start condition	1.3		µs
t_R	Rise time for both clock and data	20 + 0.1 Cb	300	ns
t_F	Fall time for both clock and data	20 + 0.1 Cb	300	ns
t_{SUP}	Pulse width of spike suppressed	0	50	ns

表 2. 標準モード（100 kHz）での I²C タイミング

Parameter	Description	Min	Max	Unit
t_L	Clock low pulse width	4.7		µs
t_H	Clock high pulse width	4.0		µs
t_{SHD}	Start condition hold time	4.7		µs
t_{DSU}	Data setup time	250		ns
t_{DHD}	Data hold time	0		µs
t_{RSU}	Setup time for repeated start	4.0		µs
t_{PSU}	Stop condition setup time	4.0		µs
t_{BUF}	Bus-free time between a stop condition and a start condition	4.7		µs
t_R	Rise time for both clock and data		1	µs
t_F	Fall time for both clock and data		300	ns

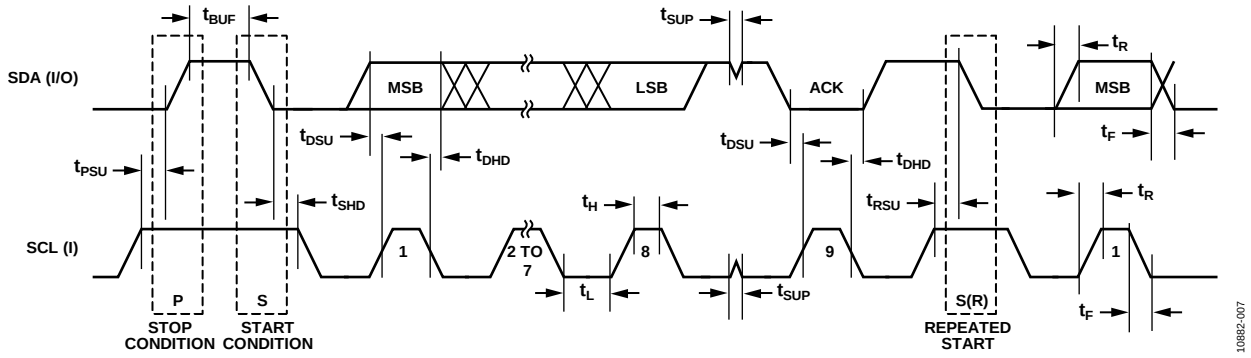

図 7. I²C 互換インターフェースのタイミング

Cortex-M3 ベースの MicroConverter の I²C の実装

Cortex-M3 ベースの ADuCxxx ファミリーは、フル・ハードウェアのマスター I²C ポートとスレーブ I²C ポートを内蔵しています。

I²C ペリフェラル・インターフェースは、合計 19 個のレジスタから構成されています（マスター用に 9 個、スレーブ用に 9 個、マスターとスレーブの共用に 1 個）。これらの一覧を表 3 に示します。

表 3.I²C レジスタ

Registers	Master	Slave	Shared
Receive	I2CMRX	I2CSRX	I2CFSTA
Transmit	I2CMTX	I2CSTX	I2CFSTA
Status	I2CMSTA	I2CSSTA	
Control	I2CMCON	I2CSCON	
Address/ID	I2CADR0/ I2CADR1	I2CID0/I2CID1/ I2CID2/I2CID3	
Clock	I2CDIV		
Other	I2CMRXCNT I2XMCRXCNT	I2CALT	

すべてのレジスタとビットの説明は、デバイス・ユーザー・ガイドに記載されています。

幾つかの特定機能は、このアプリケーション・ノートで詳しく説明しています。

通信速度の設定

I2CDIV は 16 ビット・レジスタで、2 つの 8 ビット値 HIGH と LOW を格納します。このレジスタ値が I²C バスの速度を次式に従って設定します。

$$f_{I2CSCL} = f_{PERIPH}/(LOW + HIGH + 3)$$

ここで、
f_{PERIPH} は I²C ペリフェラル・クロック。
HIGH = I2CDIV[15:8]。I²C バス・クロックのハイ・レベル区間は、(HIGH + 2) ÷ I²C ペリフェラル・クロックで決定されます。
LOW = I2CDIV[7:0]。I²C バス・クロックのロー・レベル区間は、(LOW + 1) ÷ I²C ペリフェラル・クロックで決定されます。

したがって、100 kHz 動作の場合、I²C ペリフェラル・クロック = 16 MHz で、LOW = 0x4F、HIGH = 0x4E。400 kHz の場合、LOW = 0x13、HIGH = 0x12。

ADuCM360 固有の部分

ADuCM360 では、I²C ペリフェラル・クロックは次式で与えられます。

$$f_{PERIPH} = f_{UCLK} \div (CLKSYSDIV \times I2CCLK)$$

ここで、
UCLK はシステム・クロックであり、16 MHz。
CLKSYSDIV は、CLKSYSDIV[0] ビットの設定に応じて 1 または 2。
I2CCD はクロック分周値であり、CLKCON1[8:6] ビットにより 1 ～ 7 の値に設定されます。

I²C ペリフェラル・クロックはデフォルトでディスエーブルされており、CLKDIS[2]で最初にイネーブルする必要があります。これは、デバイスの消費電力を最適化するためのものです。詳細については、AN-1111 アプリケーション・ノート「*Options for Minimizing Power Consumption When Using the ADuCM360/ADuCM361*」を参照してください。

次のフローチャートでは、I²C ペリフェラル・クロックがイネーブルされ、設定済みであると想定しています。

FIFO の使用

I²C ハードウェア・インターフェースは、I²C 機能あたり 4 個の 2 バイト FIFO を内蔵しています。

- マスター受信
- マスター送信
- スレーブ受信
- スレーブ送信

これらの各々は、受信/送信するバイトの残りのビットを保持するシフトレジスタを内蔵しています。

送信 FIFO

データを送信するときは、I2CSTX/I2CMTX レジスタへロードする必要があります。

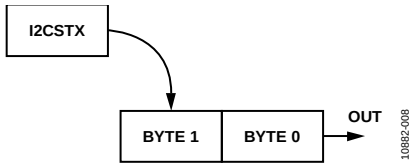

図 8. 送信 FIFO

Tx レジスタへのバイト書込みは、 FIFO のバイト 1 への書込みと等価です （図 8 参照）。

- バイト 0 がエンプティの場合、バイト 1 内のバイトが自動的にバイト 0 へプッシュされます。これはユーザーから見えません。I2CFSTA レジスタは、FIFO に 1 バイトが存在するとき表示します。
- バイト 0 が既にフルの場合、バイトはバイト 1 に留まります。Tx に再度書込みを行うと、バイト 1 が上書きされます。

バイト 0 のシフトレジスタ （OUT）への移動もユーザーから見えませんが、スレーブ側で有効な通信が必要です。
FIFO はフルでない場合、TXREQ ビット （I2CMSTA[2]/I2CSSTA[2]）をセットします。これらのビットがイネーブルされていると、割込みが発生します。

I2CFSTA レジスタの送信 FIFO フラッシュ・ビットをセットすると、FIFO がエンプティになります。

受信 FIFO

データを受信するときは、データはバイト 0 に到着します。

- バイト 1 がエンプティの場合、バイト 0 が自動的にバイト 1 へシフトされます。
- バイト 1 が既にフルの場合、I2CSRX が読出されるまで（バイト 1 の読出しと等価 ）、バイト 0 はそこに留まります。
- FIFO がフルのとき他のデータが到着すると、スレーブはそのデータに対して NACK を送信し、I2CSSTA[4]（マスターの場合は I2CMSTA[9] ）をセットします。

FIFO がエンプティでない場合、RXREQ ビット（I2CMSTA[3]/I2CSSTA[3]）をセットします。これらのビットがイネーブルされていると、割込みが発生します。

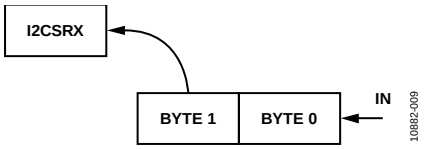

図 9. 受信 FIFO

DMA チャンネルの使用

DMA コントローラの次の 4 チャンネルが I²C インターフェースに割当てられています。

- マスター受信
- マスター送信
- スレーブ受信
- スレーブ送信

送信の場合は送信 FIFO に空きがあるとき、受信の場合は受信 FIFO にバイトが存在するとき、I²C ペリフェラルは DMA 要求を発生します。

メモリからすべてのバイトが送信 FIFO へ転送されたとき、または DMA チャンネルに割り当てられたメモリがフルのとき、DMA 転送が終了して、割込みが発生します。

I²C ステータス・ビットの TXREQ と RXREQ は DMA モードでセットされた場合、割込みは生成しませんが、トランザクション終了を表す TCOMP ビットは割込みを発生させることができるので、 DMA モードで使用することができます。

I²C 低レベル関数

コード開発を簡単化するため、低レベル関数のセットが I2cLib に用意されています。これらの関数を表 4 に示します。これらは、CD/DVD のドキュメント・フォルダで詳しく説明しています。

各シナリオのフローチャートは、これらの低レベル関数を使用しています。

表 4.I²C 低レベル関数

Master Configuration	**Description**
int I2cMCfg(int iDMACfg, int iIntSources, int iConfig);	Configure I²C master
int I2cBaud(int iHighPeriod, int iLowPeriod);	Configures I²C baud rate
int I2cMWrCfg(unsigned int uiDevAddr);	Configures slave address
int I2cMRdCfg(unsigned int uiDevAddr, int iNumBytes, int iExt);	Configures slave address, number of bytes to read
int I2cMRdCnt(void);	Read the counter of bytes received by the master
Slave Configuration	**Description**
int I2cSCfg(int iDMACfg, int iIntSources, int iConfig);	Configures I²C slave
int I2cSIDCfg(int iSlaveID0, int iSlaveID1, int iSlaveID2, int iSlaveID3);	Configure the slave addresses
int I2cSGCallCfg(int iHWGCallAddr);	Setup the hardware general call
Common Functions	**Description**
int I2cRx(int iMode);	Reads the Rx register of the slave or master
int I2cTx(int iMode, int iTx);	Writes in the Tx register of the slave or master
int I2cStr(int iMode, int iStretch);	Configures clock stretching
int I2cFifoFlush(int iMode, int iFlush);	Flush slave or master Tx FIFO
int I2cSta(int iMode);	Reads the status of the slave or master

マスター送信

バイトを送信するには、データを先に送信 FIFO へロードします。スレーブのアドレスは、I2CADR0 レジスタで指定します。データ書込みの場合は、アドレス・レジスタの書込み（$\overline{W}$）ビットをゼロに設定します。I2CADR0 レジスタに書込みを行うと、スタート条件が自動的に発生します。

送信 FIFO がエンプティであるか、またはフルではなく、かつ I2CMSTA のビット 2 がセットされた場合、送信されたバイトの先頭クロックで I²C 割込みが発生し、マスターがバイトを送信したことを示します。これにより、ユーザーはバイトを FIFO に追加することができます。FIFO のステータスは、I2CMSTA[1:0]または I2CFSTA レジスタでチェックすることができます。

転送を開始するとき FIFO に 1 バイトしか存在しない場合、最初の I²C 割込みは送信されたアドレスの先頭クロックで発生します。FIFO に 2 バイト存在する場合は、送信された最初のバイトの先頭クロックで割込みが発生します。

送信 FIFO のフルが維持されない場合、送信割込みは各クロック・エッジで発生します。最後のバイトの最終ビットが送信された後に、TXUR ビットがセットされます。TXUR がセットされてから½ クロック・サイクル以内に、すなわちスレーブ・アクノリッジ・ビットの間にデータが FIFO にロードされないと、マスターは自動的に送信を終了します。

ストップ条件は、最終バイト送信の 5.1 µs 後に自動的に発生します。

I2CMCON MMR の TXREQ 割込みイネーブル・ビット（I2CMCON[5]）をクリアする必要があります。そうしないと、ストップ条件が発生するまで、送信割込みが連続して発生します。

割込みがイネーブルされている場合 （I2CMCON[8]）、ストップ条件が発生すると、TCOMP ビットも割込みを発生します。これにより、例えば低消費電力モードになる前に、I²C ペリフェラルを安全にターンオフすることができます。

スレーブからのデータ要求にマスターが応答する例を図 10 に示します。

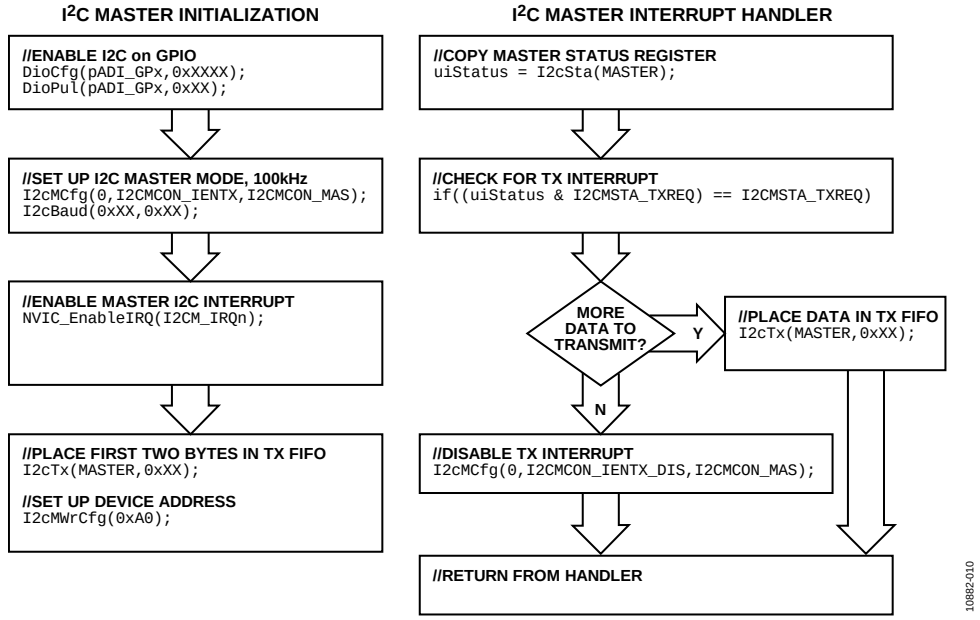

図 10.マスター送信のフローチャート

スレーブ受信

I²C スレーブがデータを受信するとき、各バイトの 8 番目のクロックが受信された後で割込みが発生します。3 番目のバイトが受信される前に FIFO が読出されていない場合、RXOF、I2CSSTA[4]がセットされて、受信 FIFO のオーバーフローを表示します。このとき、FIFO を読出すことができます。9 番目のクロックの立上がりエッジの前に FIFO が読出されない場合、スレーブ・インターフェースは自動的に NACK を出力します。

FIFO からデータを読出すときは、I2CRX レジスタを使います。RXREQ、I2CSSTA[3]は、スレーブがデータを受信したことを表示します。I2CSRX を読出すだけでこのビットはクリアされます。割込みがイネーブルされていて RXREQ がセットされると、I²C 割込みが連続して発生します。

マスターは最後のデータを送信した後、自動的にストップ条件を送信します。スレーブはストップ条件を検出し I2CSSTA[10]をセットします。IENSTOP、I2CSCON[8]がセットされている場合、このビットは割込みを発生することができます。

スレーブがマスターからバイトを受信するフローチャートを図 11 に示します。図 12 に、ステータス・ビットがセットされるタイミングと割込みが発生するタイミングを示します。

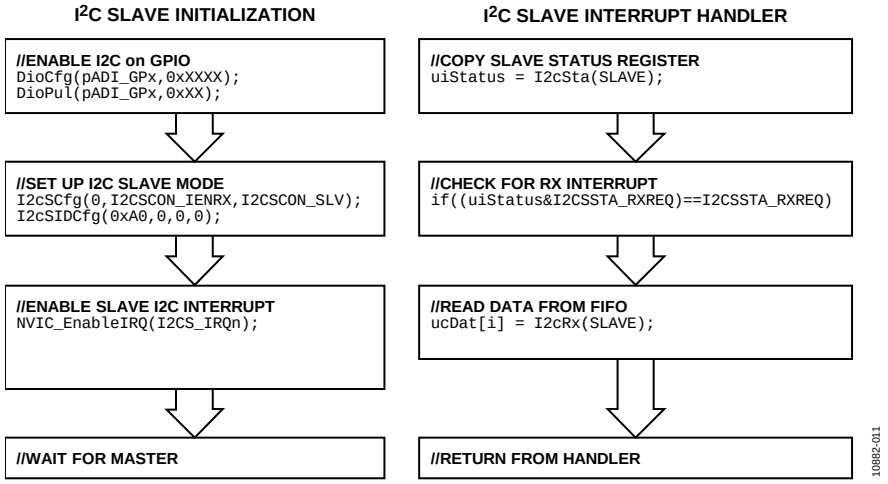

図 11.スレーブ受信のフローチャート

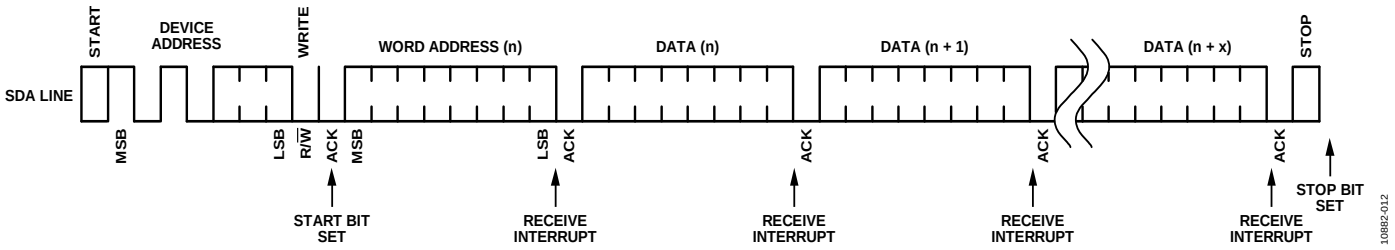

図 12.スレーブ受信の例

マスター受信

マスター・モードでは、スレーブからのデータ読出しに同様の方法を使います。まず、I2CMRXCNT レジスタを使って読出すバイト数を設定します。これは、スレーブから読出すバイト数 + 1 を示します。これは 8 ビット・レジスタであるため、一度に 256 バイトを受信することができます。大きな転送には EXTEND オプションを使用することができます。I2CMCRXCNT には、マスターが受信した現在のバイト数が表示されます。

データの受信を開始するためには、 I2CADR0 レジスタの読出し（R）ビットをセットします。これにより、I2CADR0 レジスタで設定したアドレスと R/$\overline{W}$ ビットの条件の下で発生したスタート条件から転送が開始されます。各バイトが受信された後（9 番目のクロックすなわち ACK または NACK の後）、割込みが発生します。RXREQ、I2CMSTA[3]がセットされて、1 バイトが受信されたことを示します。I2CMRX を読出すだけでこのビットはクリアされます。

マスターがこれ以上データを受信する必要がない場合、受信した最後のバイトに対して NACK を自動的に発生します。これにより、スレーブにバイト送信の停止を通知して、マスターがストップ条件を発生できるようにします。

受信されたデータが時間内に読出されず、FIFO がフルになると、マスターは受信された余分なデータに対して NACK を送信します。

マスターがスレーブからバイトを受信するフローチャートを図 13 に示します。

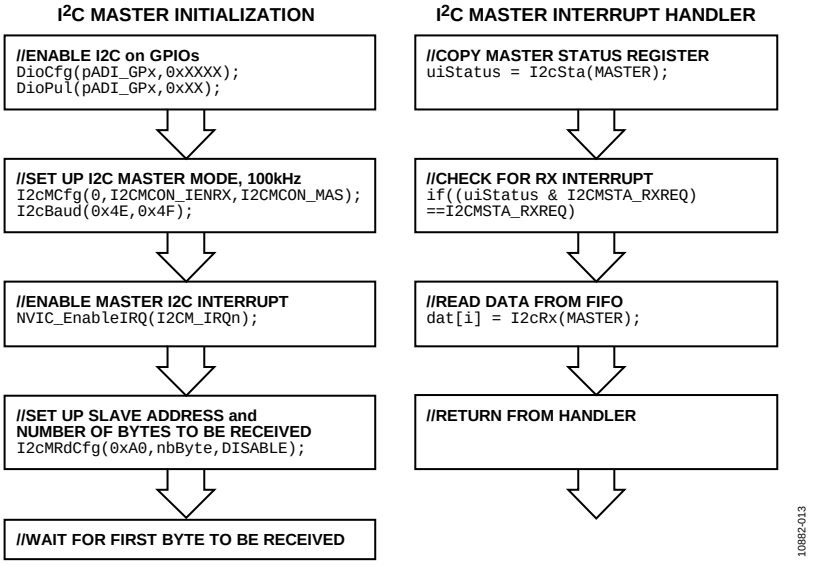

図 13.マスター受信のフローチャート

スレーブ送信

スレーブはデータ送信が要求されるたびに割込みを発生します。最初はアドレスの ACK 後に発生、すなわち FIFO のバイト 0 の送信中に発生します。データはスレーブ送信 FIFO に予めロードされている必要があります。そうしないと、マスターからの最初の読出し要求で NACK が発生します。FIFO に予め 2 セットのデータがロードされている場合、アドレスの ACK の後に 1 つの割込みが発生し、その後送信された各バイトの ACK の後に発生します。FIFO に予めロードされているのが 1 セットのデータのみである場合、アドレスの ACK の後に 2 つの割込みが発生し、FIFO は最初のデータを送信した後にエンプティになります。

バイトが送信された後、マスターがデータの要求を続ける限り割込みが発生します。

送信 FIFO に空きがある場合、またはバイトがマスターへ送信されるごとに、TXREQ、I2CSSTA[2]がセットされます。送信するデータがなくなった場合、送信割込みをディスエーブルすることが可能で、ストップ条件を検出したとき （I2CSSTA[10]）再度イネーブルすることができます。IENSTOP、I2CSCON[8]がセットされていると、STOP 検出は割込みを発生することができます。

スレーブがマスターからのバイト要求に応答するフローチャートの例を図 14 に示します。図 15 に、ステータス・ビットがセットされるタイミングと割込みが発生するタイミングを示します。

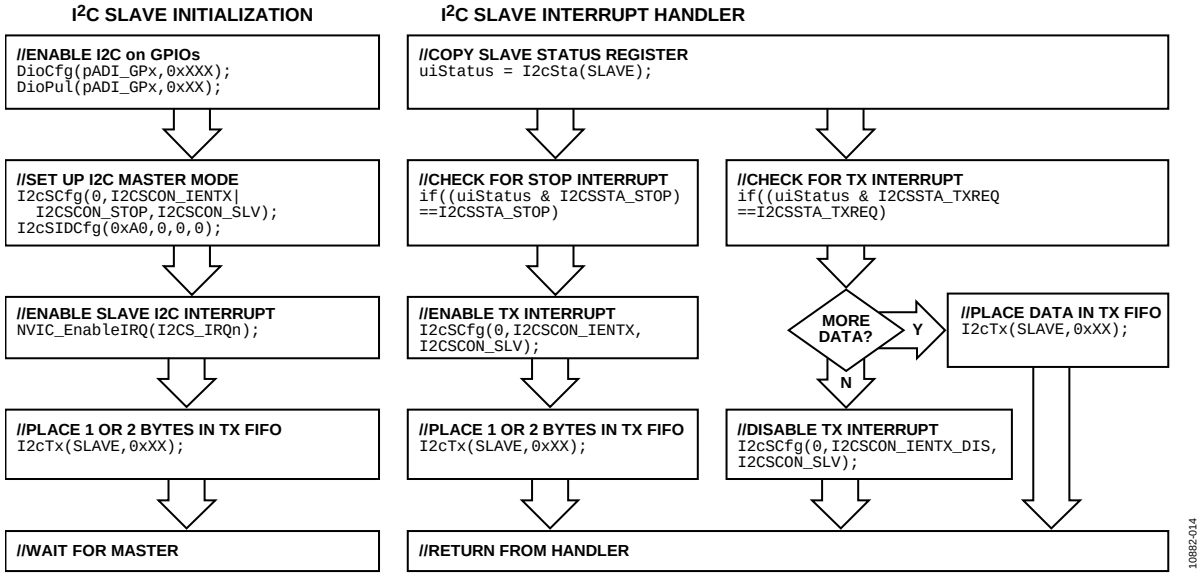

図 14.スレーブ送信のフローチャート

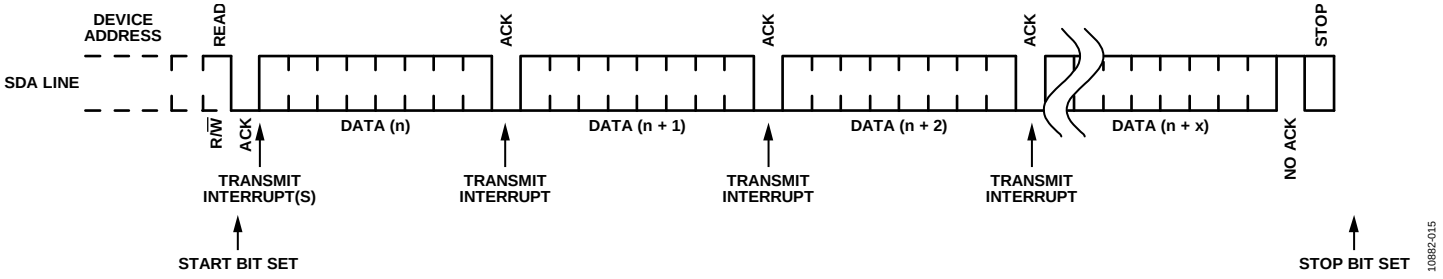

図 15.スレーブ送信の例

DMA モード、マスター送信

I²C 転送は、マスターからのスレーブ・アドレスの送信で開始されます。I2CMCON MMR の TXDMA をセットして、DMA 送信モードが設定されると、I²C ペリフェラルがすべて設定されたとき、すなわちボー・レート、スレーブ・アドレス、DMA 送信要求が設定され、さらに DMA コントローラが設定されてイネーブルされたとき、転送が開始されます。転送の開始前に、I²C マスター送信 DMA チャンネルも NVIC でイネーブルする必要があります。

メモリから FIFO へすべてのバイトが転送されたとき DMA 転送は完了します。これは、最後の 2 バイトが FIFO に残り、終わりから 3 番目のバイトの送信中を意味します（図 17 参照）。ここで、DMA コントローラの中で、DMA チャンネルが自動的にディスエーブルされますが、FIFO がフルでない場合、I²C ペリフェラルは DMA 要求を DMA コントローラに送信して割込みを発生させます。このため、DMA コントローラで I²C 要求をマスクして重複割込みを避ける必要があります(DMARMSKSET）。

新しい転送を開始するときは、DMA コントローラを再設定／再イネーブルする必要があります。

I²C マスター送信 DMA 転送のフローチャートを図 16 に示します。

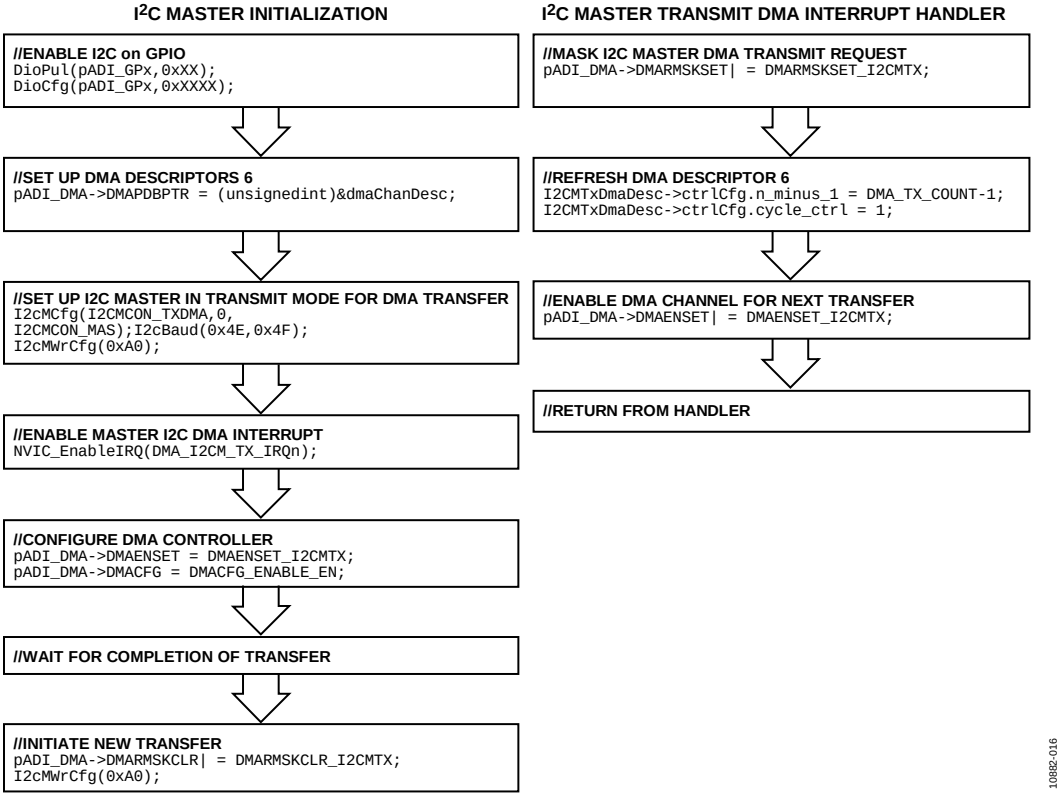

図 16.マスター送信 DMA 転送のフローチャート

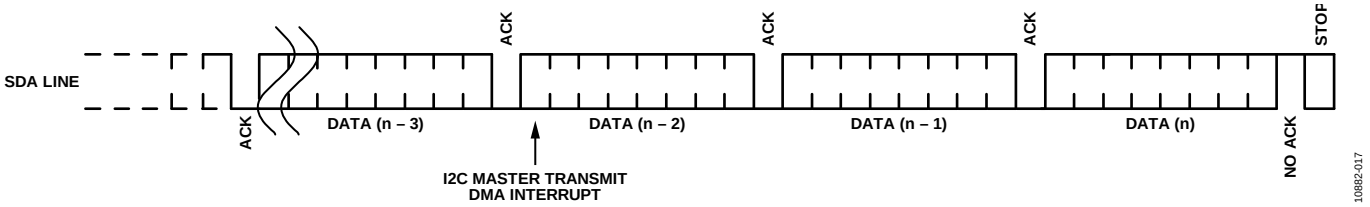

図 17.マスター送信 DMA 割込み

DMA モード、マスター受信

I2CMCON MMR の RXDMA をセットすると、I²C ペリフェラルで DMA 転送がイネーブルされます。DMA コントローラが設定されイネーブルされ、かつ I²C ペリフェラルがフルに設定されたとき、すなわちボー・レート、スレーブ・アドレス、DMA 送信要求が設定されたとき、転送が開始されます。転送の開始前に、I²C マスター送信 DMA チャンネルも NVIC でイネーブルする必要があります。

DMA コントローラで予定したすべてのバイトが受信されたとき、DMA 転送が完了します。I²C マスター I2CMRXCNT MMR も同じバイト数に設定される必要があります。

DMA 転送が完了すると、 DMA コントローラで対応するチャンネルが自動的にディスエーブルされます。新しい転送を開始するには、DMA コントローラを再設定/再イネーブルする必要があります。I2CADR0 MMR にスレーブ・アドレスを再度書込むと、新しい転送が開始されます。

マスターがスレーブからデータを受信する DMA 転送のフローチャートを図 18 に示します。

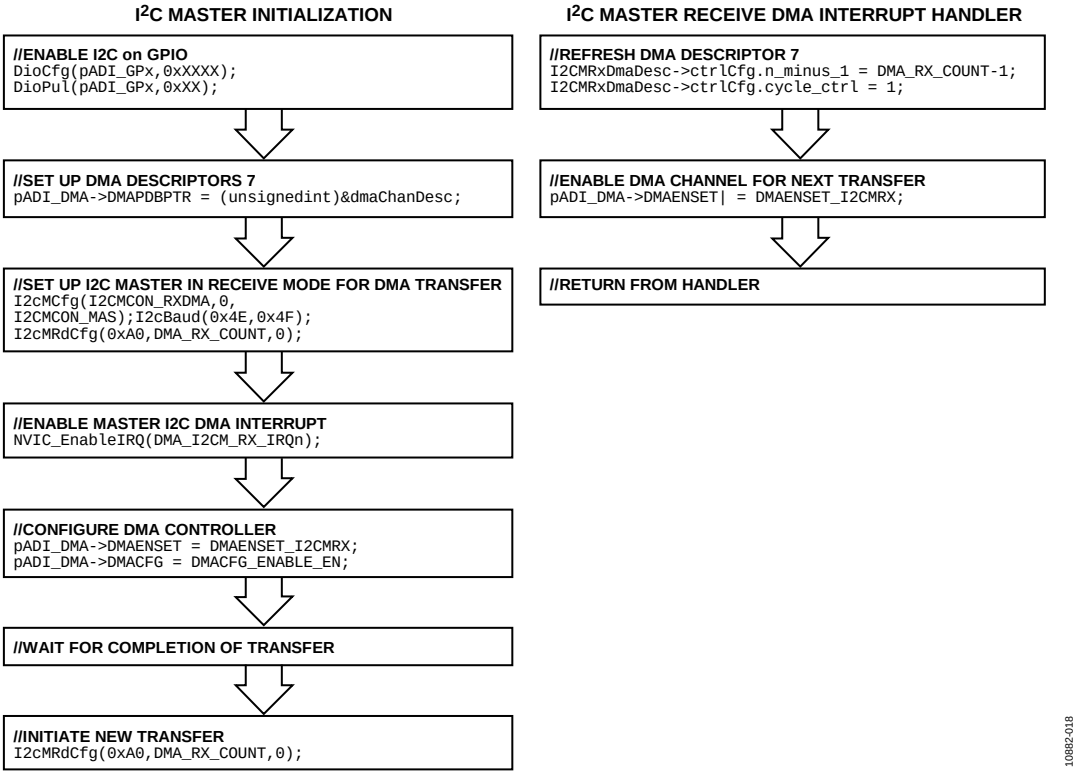

図 18.マスター受信 DMA 転送のフローチャート

DMA モード、スレーブ送受信

I2CMCON MMR の TXDMA および/または RXDMA をセットすると、I²C ペリフェラルで DMA 転送が可能になります。I²C ペリフェラルをスレーブ・モードに設定し、I²C 割込みをディスエーブルします。NVIC と DMA コントローラを設定します。正しい I²C アドレスを受信すると、DMA 転送が開始され、データ・バイトのみがメモリへ転送されます。DMA 転送が完了すると、DMA コントローラで対応するチャンネルが自動的にディスエーブルされます。新しい転送を開始するときは、DMA コントローラのみを再設定する必要があります。図 19 に、スレーブが送信および受信する DMA 転送のフローチャートを示します。

送信モードでの DMA 転送は、最終バイトから 3 番目の送信中に終了することに注意してください。これはマスター送信の場合と同様です。

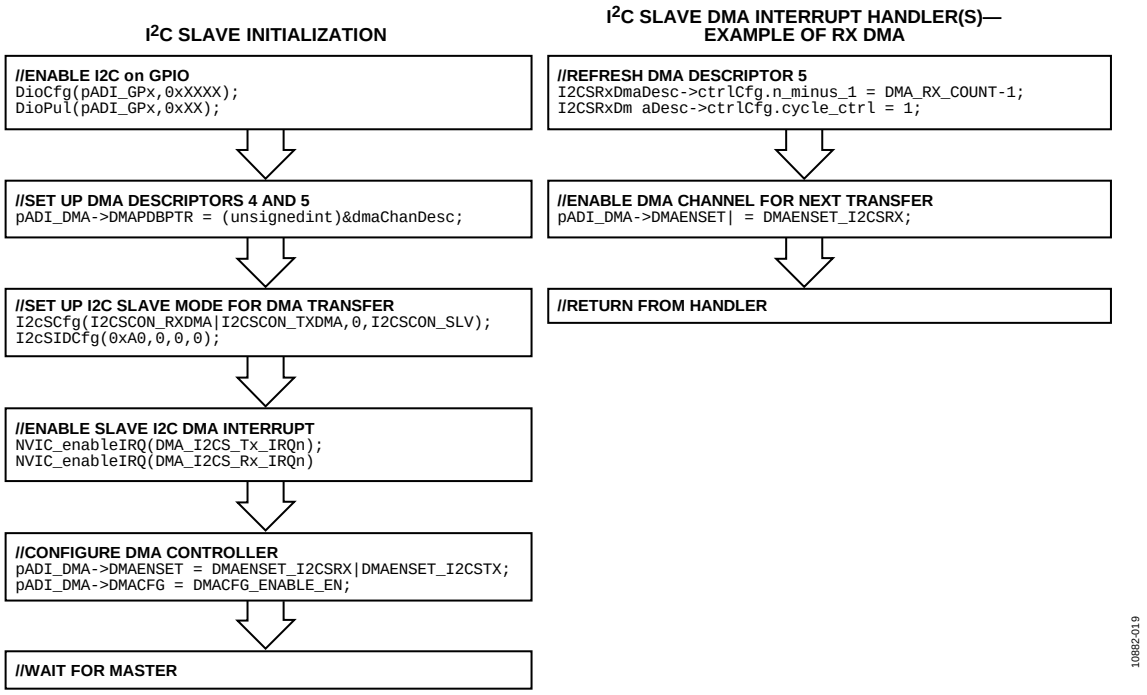

図 19.スレーブ DMA 転送のフローチャート

コンパニオン・コード

コンパニオン・コードのリストと説明を表 5 に示します。

表 5.コンパニオン・コード

Mode/Flowchart	Code Example/Tools
Master Transmit	I2Cmaster.c
Slave Receive	I2Cslave.c
Master Receive	I2Cmaster.c
Slave Transmit	I2Cslave.c
DMA Mode, Master Transmit	I2CmasterDMA.c
DMA Mode, Master Receive	I2CmasterDMA.c
DMA Mode, Slave Receive and Transmit	I2CslaveDMA.c

I²C は、Philips Semiconductors 社（現在の NXP Semiconductors 社）が制定した通信プロトコルです。