ADuC702x MicroConverter®のI²C®互換インターフェース

著者：Michael Looney

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ADuC702xファミリー高精度マイクロコントローラを使用して、I²C（inter-integrated circuit）互換インターフェース・ハードウェアのマスターとスレーブを実装する方法について説明します。マスターとスレーブがI²Cインターフェースを介してどのように通信するかを示すコード例も紹介します（「シリアルEEPROMプロトコルの実行」を参照）。

I²Cバスの主な特長は、以下のとおりです。

- 必要なバス・ラインは、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つのみです。いずれも双方向であるため、マスターとスレーブの両方がトランスミッタまたはレシーバとして動作できます。
- I²Cマスター1個で複数のスレーブ・デバイスと通信できます。各スレーブ・デバイスには一意の7ビット・アドレスがあるため、複数のスレーブを使用する環境でも1つのマスター／スレーブ関係が存在できます。
- マスターとスレーブは、最大400kbpsでデータを送受信できます。
- オンチップのフィルタリング機能により、SDAとSCLの各ライン上で発生する50ns未満のスパイクを除去して、データ・インテグリティを維持します。

I²Cインターフェースの代表的なブロック図を図1に示します。

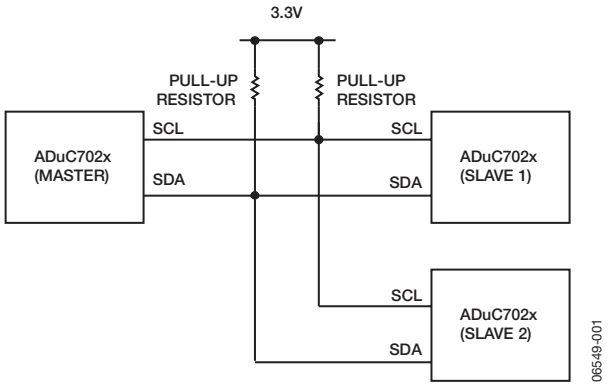

図1. 1個のマスターと複数のスレーブで構成するI²Cインターフェースのブロック図

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MTビル2号
電話06（6350）6868

目次

I²Cインターフェースの概要

I²Cは、フィリップス社が開発した2線式シリアル通信システムであり、2線（SCLとSDA）を介して複数のマスターと複数のスレーブの接続ができます。I²Cインターフェースでは、少なくとも1個のマスターと1個のスレーブが必要です。

SCL信号が、マスターとスレーブ間のデータ転送を制御します。常にマスターがスレーブにSCL信号を送信します。ただし、スレーブが次の送信を開始する準備ができていない場合は、この信号ラインをローレベルにすることができます。これをクロック・ストレッチングと呼びます。データビットの送信のたびに、クロック・パルスを1つ発生する必要があります。

SDA信号は、データの送信または受信に使用します。SCLがハイレベルの間は、SDA入力を安定した状態にしなければなりません。SCLがハイレベルのときにSDAラインのロジック・レベルが遷移すると、スタートまたはストップ条件が発生したと判断されます（図2と図3を参照）。

I²Cの基本

スタート条件

I²Cインターフェースの典型的なデータ転送シーケンスは、スタート条件から始まります。SCLラインがハイレベルのときにSDAラインがハイレベルからローレベルに遷移するのが、スタート条件です（図2を参照）。スタート条件を発生するのは、常にマスターです。スタート（およびストップ）条件のみが、SCLラインがハイレベルの間にSDAラインのロジック・レベルの遷移が必要になる場合です。通常のデータ転送（スレーブのアドレッシングを含む）時には、SCLラインがハイレベルの間はSDAライン上のデータを安定した状態にする必要があります。

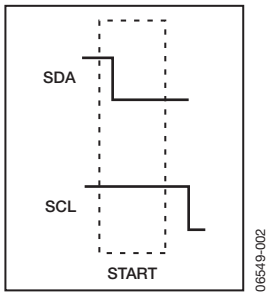

図2. I²Cのスタート条件

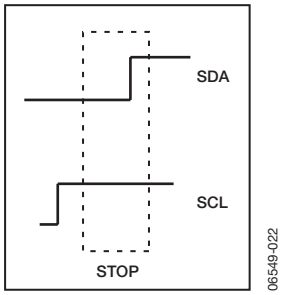

図3. I²Cのストップ条件

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パルスとともに1バイトを最上位ビット（MSB）ファーストでSDAラインから送信します。このバイトの最初の7ビットが、7ビット・スレーブ・アドレスです。7ビットの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1つ）と一致する場合のみ、スレーブがマスターに応答します。8番目のビットの最下位ビット（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決定します。このビットをクリアすると、マスターが選択されたスレーブに対してデータを書き込みます。このビットをセットすると、マスターはスレーブからのデータ受信を待ちます。いずれの場合も、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、つまりマスターから送信された7個のMSBがI2C0ADRメモリマップ・レジスタ（MMR）の7個のMSBと一致すれば、スレーブは有効なACKを返し、SCLラインをローレベルにして、I2C0STAのフラグをセットします。

前述したように、スレーブはI²Cスレーブ・アドレッシングの処理をすべてハードウェアで自動的に実行しますが、スレーブ・アドレスを正しく出力することはマスターの責任です。

アクノレッジ（ACK）／ノー・アクノレッジ（NACK）

マスターから送信されたアドレスとスレーブ・アドレスが一致すれば、スレーブはアクノレッジ（ACK）を自動的に送信します。アドレスが一致しなければ、ノー・アクノレッジ（NACK）を送信します。9番目のクロック・パルス時にSDAラインがローレベルに遷移すると、ACKになります。9番目のクロック・パルス時にSDAラインがハイレベルに遷移すると、NACKになります（図4を参照）。

データ転送時には、レシーバが必ずACKまたはNACKを返します。ただし、ACKに必要なクロック・パルスは必ずマスターが発生します。ACKクロック・パルスの間、トランスミッタはSDAライン（ハイレベル）を解放する必要があります。有効なACKを返すためには、レシーバがSDAラインをローレベルにする必要があるためです。

ADuC702x MicroConverterでは、ACKもNACKも受信中の各バイトの終わりでハードウェアにより自動的に発生されます。

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・アドレスまたは送信されたデータに対してスレーブが応答しなかった場合）、マスターはストップ条件を発生してデータ転送を停止します（「データ転送」を参照）。

マスター・レシーバは、スレーブから送信された最後のバイトの後にノー・アクノレッジ（NACK）を返し、スレーブ・トランスミッタにデータ・シーケンスの終了を通知しなければなりません。スレーブはNACKを受信すると、ただちにSDAラインを解放し、マスターがストップ条件を発生できるようにします。

データ転送

I²Cの割込みサービス・ルーチン（ISR）、すなわちポーリング方式では、マスターから送信されたR/$\overline{W}$ビットのステータスから、スレーブは送信または受信のいずれを行うかどうかを判断します。次に、スレーブはマスターからクロックが送信されるたびに1ビットを送信もしくは受信します。マスターの方では、スレーブがマスターとデータの送受信ができるように9個のクロック（データ用の8個とACK用の1個）を出力する必要があります。スレーブが有効なデータバイトを送信または受信するたびに、I²C割込みビットがセットされます。

繰り返しになりますが、スレーブがトランスミッタ、マスターがレシーバとなるシステムでは、マスターはスレーブから送信された最後のバイトの後にNACKを送信することで、スレーブに対してデータ・シーケンスの終了を通知する必要があります。スレーブはNACKを受信した場合、ただちにSDAラインを解放して、マスターがストップ条件を発生できるようにする必要があります。

マスターがデータ転送を停止するか、またはバス上の別のマスターに対してデータ転送を中断させたい場合は、スタート条件に続いてストップ条件を送信することによってこれを実行できます。

ストップ条件

データ転送シーケンスは、ストップ条件によって終了します。ストップ条件は、SCLがハイレベルの間にSDAラインがローレベルからハイレベルに遷移することです（図3を参照）。

ストップ条件を発生するのは、常にマスターです。マスターがデータ・シーケンスが正常に終了したと判断したとき、またはスレーブ・デバイスからNACKを受信した場合にストップ条件を送信します。スレーブ・デバイスはストップ条件を受信するとリセットされ、再びスレーブ・アドレスが送信されてくるのを待ちます。

ADuC702xデバイスのI²Cインターフェースでは、ストップ条件で割込みを行うように設定できます。この機能は、I2CxCFGMMRのビット14を使ってイネーブルします。

代表的な転送シーケンスを図5に示します。

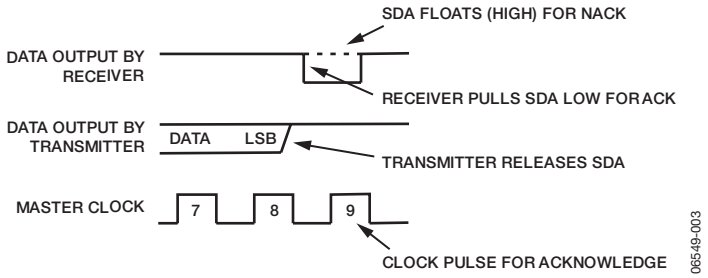

図4. I²Cバス上のアクノレッジ（ACK）およびノー・アクノレッジ（NACK）

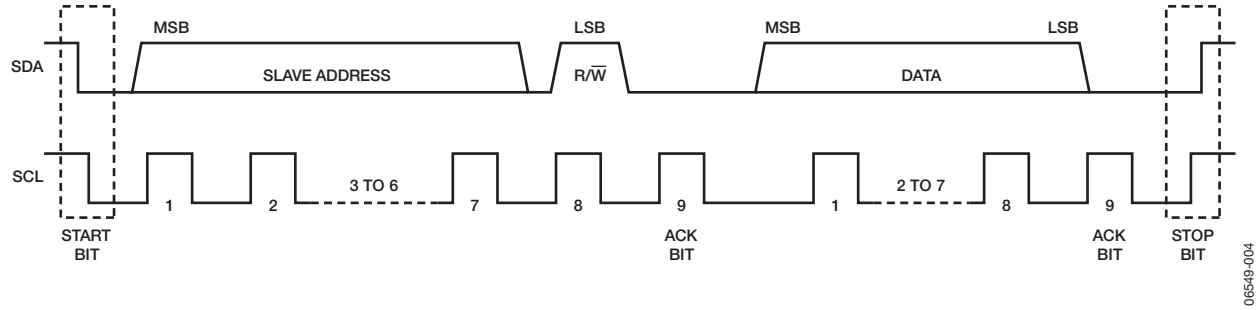

図5. 代表的なI²C転送シーケンス

繰返しスタート条件

ストップ条件が送信されずに2番目のスタート条件がスレーブに送信されると、繰返しスタート条件となります。これにより、マスターはバスの制御を放棄することなく、R/$\overline{W}$ビットを変更して転送方向を反転できるようになります。

転送シーケンスの一例を図6に示します。この方式は一般に、デバイスに送信される最初のデータに基づいて読出し対象となるレジスタ・アドレスが設定される場合に採用されています。

繰返しスタート条件＋スレーブ・アドレスを受信すると、割込みが発生します。この割込みは、I2CxSSTA MMRのステータスビットを使用することで、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。

ADuC702xデバイスは、マスター・モードのときにI²C繰返しスタートのシーケンスを直接発生させることができないので注意してください。

ユーザがI²Cバス全体を完全に制御できる場合は、任意の2番目のスタート条件の送信前にストップ条件をディスエーブルするという方法でこの問題を解決することができます。これは、次の手順で行います。

1. 値の小さいプルアップ抵抗をSDAラインに接続します。たとえば、4.7kΩの抵抗を介してSDAをV_{CC}に接続し、20kΩの抵抗を介してSCLをV_{CC}に接続します。
2. 以下のコード・シーケンスを追加します。

```
I2C0CCNT = 0x0;       // Sets Start/Stop condition counter value to 0 - minimum value.
I2C0ADR = 0xA0;       // Write sequence
I2C0MTX = 0x7;        // Load the Tx FIFO
while ((I2C0FSTA & 0x30) != 0x00) {} // Wait for the Tx FIFO to empty
I2C0CNT = 0x1;        // Read 2 bytes from the slave
I2C0ADR = 0xA1;       // Send out the Read condition
I2C0CCNT = 0x80;      // Set the Start/Stop counter to a nonzero value to re-enable the Stop
                      Condition
```

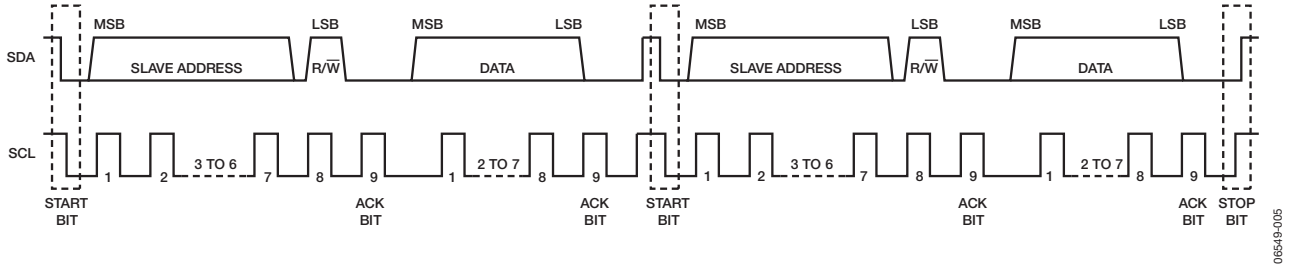

図6. I²Cの繰返しスタート・シーケンス

クロック・ストレッチング

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。RS232と異なり、I²Cバスでは、予め定められたボーレートに対してマスターとスレーブが正確に同期することを不要にする明確なクロック信号を使用しています。

ただし、場合によってはマスターが設定したクロック速度にI²Cスレーブが対応できず、速度を多少下げなければならないことがあります。これは、クロック・ストレッチングと呼ばれるメカニズムによって行います。

I²Cスレーブは、バス速度を低下させる必要がある場合、クロック信号をローレベルに維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レベルの状態に設定して解放した後でこのクロック信号を読込んで、クロック信号ラインが実際にハイレベルになるまで待ちます。

I2CxCFG MMRのビット11を使用して、クロック・ストレッチングを設定できます。

シリアルEEPROMプロトコル

ATMEL AT24Cシリーズのシリアル EEPROMは、以下の5つのコマンドに対応しています。

- ランダム書込み
- シーケンシャル書込み
- カレント・アドレス読出し
- ランダム読出し
- シーケンシャル読出し

これらの5つのコマンドを図7～11に示します。

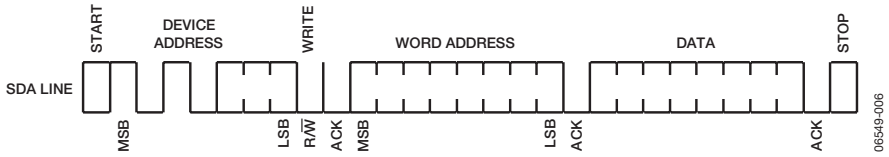

図7. ランダム書込み

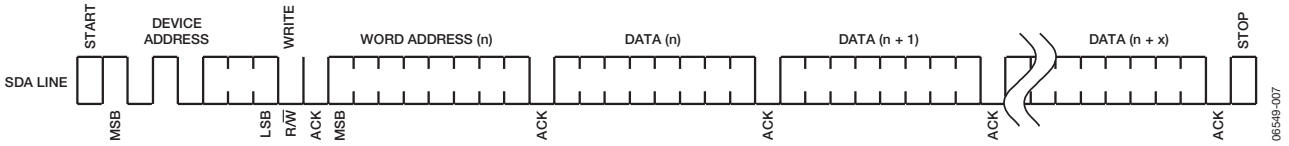

図8. シーケンシャル書込み

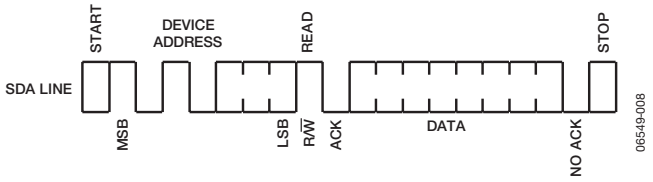

図9. カレント読出し

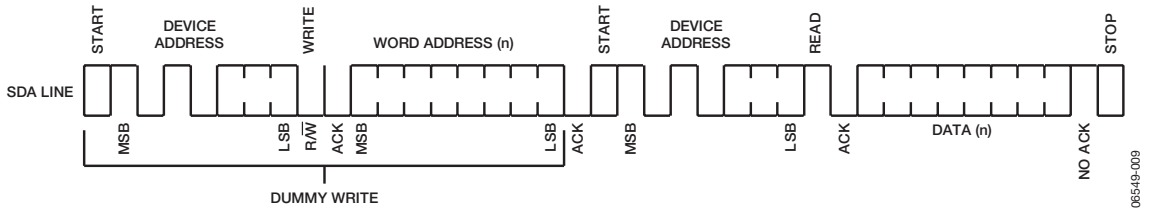

図10. ランダム読出し

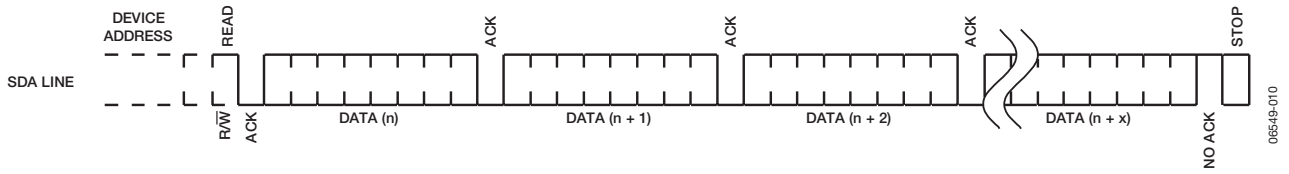

図11. シーケンシャル読出し

ADuC702xシリーズMicroConverterでのI²Cの実装

ADuC702xシリーズのデバイスには、ハードウェアの全機能装備マスターおよびスレーブI²Cポートが2つあります。これらのポートはそれぞれ最大4つのアドレスをサポートし、シリアルEEPROMコマンドの実行を可能にする割込みを設定できます。

基本的には、I²Cハードウェア・インターフェースは標準のUARTと同じように動作します。受信バッファと送信バッファがあり、それぞれ2バイトのFIFOから構成されています。ここでは、4つのタイプの通信（マスター／スレーブの送受信）とFIFOの使用法について詳しく説明します。

FIFOの使用法

各I²Cブロックごとに、次の4個の2バイトFIFOがあります。

- マスター受信
- マスター送信
- スレーブ受信
- スレーブ送信

以下、送信FIFOと受信FIFOについて説明します。

表1. I2CxFSTA MMRのビットの説明

ビット番号	説明
31～10	予備
9	マスター送信FIFOのフラッシュ ユーザが設定して、マスターTx FIFOをフラッシュします。このビットでスレーブ受信FIFOもフラッシュされます。 マスターTx FIFOがフラッシュされると、自動的にクリアされます。
8	スレーブ送信FIFOのフラッシュ ユーザが設定して、スレーブTx FIFOをフラッシュします。 スレーブTx FIFOがフラッシュされると、自動的にクリアされます。
7～6	マスターRx FIFOのステータスビット 00 FIFOが空 01 FIFOにバイトが書き込まれる 10 FIFOに1バイト 11 FIFOが満杯
5～4	マスターTx FIFOのステータスビット 00 FIFOが空 01 FIFOにバイトが書き込まれる 10 FIFOに1バイト 11 FIFOが満杯
3～2	スレーブRx FIFOのステータスビット 00 FIFOが空 01 FIFOにバイトが書き込まれる 10 FIFOに1バイト 11 FIFOが満杯
1～0	スレーブTx FIFOのステータスビット 00 FIFOが空 01 FIFOにバイトが書き込まれる 10 FIFOに1バイト 11 FIFOが満杯

送信FIFO

データを送信するには、I2C0STX/I2C0MTXレジスタにデータをロードする必要があります。Txレジスタに1バイトを書き込むことは、FIFOのバイト1への書込みと同じです（図12を参照）。

- バイト0が空のとき、バイト1のバイトデータがバイト0に自動的に移動します。これは、ステート・マシーンを使って説明してあります（図13を参照）。I2C0FSTAレジスタではユーザがステートを確認できます。
- バイト0が満杯の場合は、バイトデータはバイト1に留まります。Txに再び書込みを行うと、バイト1が上書きされます。

I2C0FSTAレジスタで送信FIFOフラッシュビットをセットすると、このFIFOが空になります。

送信が実行されると、バイト0が送信され、バイト1がバイト0にシフトされ、FIFOはステート2に遷移します。

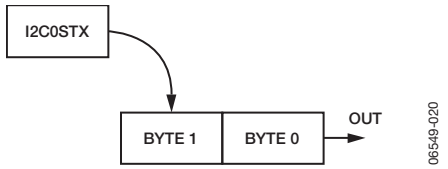

図12. 送信FIFO

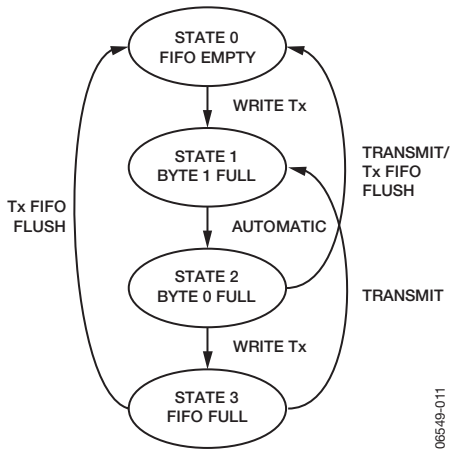

図13. 送信FIFOのステート・マシーン

受信FIFO

データの受信時は、バイト0にデータが入力されます。

- バイト1が空の場合は、バイト0がバイト1に自動的にシフトされます。
- バイト1が満杯の場合は、I2C0SRXからデータが読み出される（すなわちバイト1から読み出される）まで、バイト0にデータが保持されます。
- FIFOが満杯のときに他のデータが到着すると、スレーブはこのデータに対してNACKを返して、I2C0SSTAのビット4をセットします。

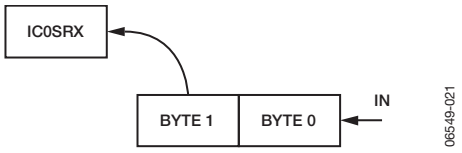

図14. 受信FIFO

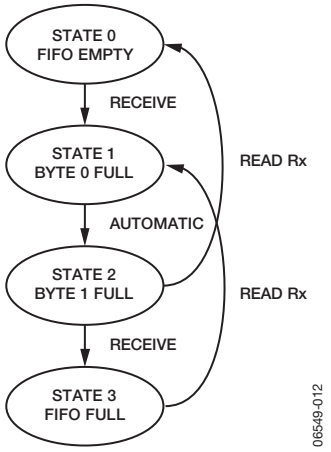

図15. 受信FIFOのステート・マシーン

マスター送信

バイトを送信するには、まずデータを送信FIFOにロードする必要があります。スレーブのアドレスをI2C0ADRレジスタで指定します。データを書き込むときは、アドレス・レジスタの書込み（$\overline{W}$）ビットをゼロに設定してください。I2C0ADRレジスタに書込みを行うと、自動的にスタート条件になります。

送信する各バイトの最初のクロックで、I^2C割込みが発生します。I2C0MSTAのビット2とビット1がともにセットされることにより、マスターが1バイトの送信を完了し、かつFIFOがアンダーフローの状態であることが表示されます。この表示によって、ユーザはFIFOに1バイトを追加することができます。

転送を開始するときにFIFOに1バイトしかない場合は、送信するアドレスの最初のクロックで最初のI^2C割込みが発生します。FIFOに2バイトある場合は、送信する最初のバイトの最初のクロックで割込みが発生します。

FIFOが空になると、送信が終了します。ストップ条件も自動的に発生されます。これは、最後のバイトの送信後5.1µsで発生します。

この動作の簡単な例を図16のフローチャートに示します。

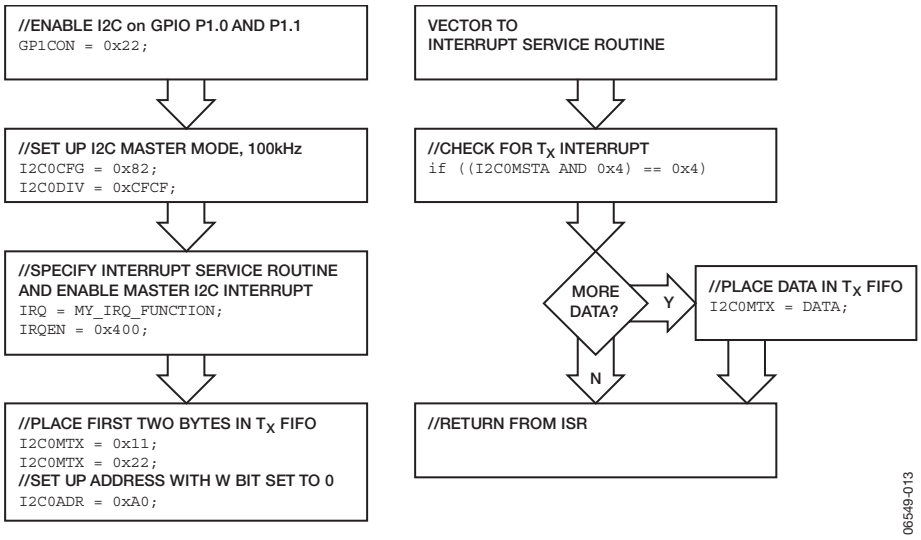

図16. マスター送信のフローチャート

スレーブ受信

I²Cスレーブがデータを受信する際に、各データ・バイトが受信FIFOに格納されるときに割込みが発生します。これは、各バイトの9番目のクロックが受信された後になります。3番目のバイトを受信するまでにFIFOのデータが読み出されない場合は、最後に送信されたデータに対してインターフェースが自動的にNACKを返し、I2CSSTAレジスタのビット4をセットして、受信FIFOがオーバーフロー状態であることを表示します。

FIFOのデータを読み出すには、I2C0RXレジスタを使用します。I2C0SSTAレジスタのビット3は、スレーブがデータの受信を完了したことを示します。I2C0SRXを読み出すだけで、このビットはクリアされます。FIFOをフラッシュしても、ビット3はクリアできません。

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。

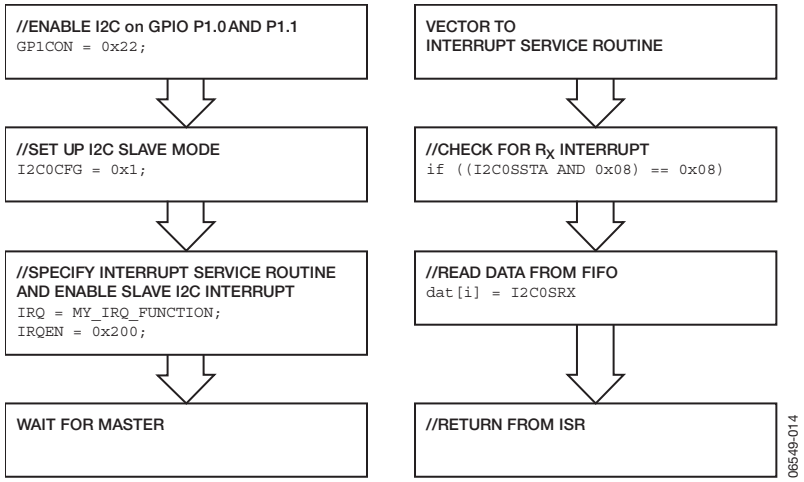

図17.　スレーブ受信のフローチャート

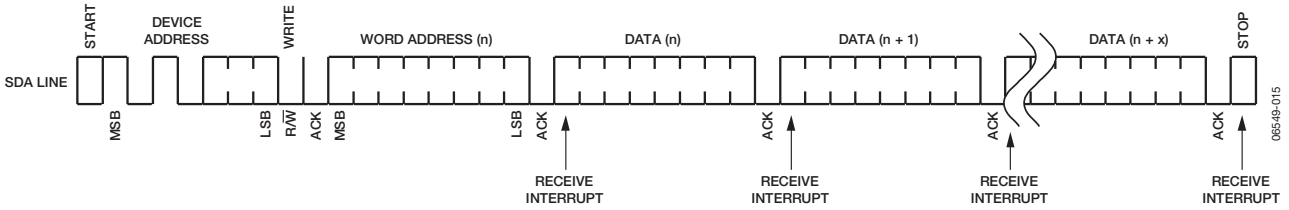

図18.　スレーブ受信の例

マスター受信

マスター・モードでスレーブからデータを読み出すときも、同様の手法を使用します。まず読み出すバイト数をI2C0CNTレジスタに設定します。これは、スレーブから読み出されるバイト数に1を加えた値になります。0～7の値を設定できますが、コードの実行中にリセットしてさらに多くのデータを読み出すこともできます。

データの受信を開始するには、I2C0ADRレジスタで読出し（R）ビットをセットします。これによって、アドレスとI2C0ADRレジスタに設定したR/$\overline{W}$ビットを使って生成したスタート条件により転送が開始されます。各バイトの受信後（9番目のクロック、ACKまたはNACKの後）、割込みが発生します。I2C0MSTAのビット3がセットされて、1バイトの受信が完了したことが表示されます。I2C0MRXを読み出すだけで、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は、最後に受信されたバイトに対して自動的にNACKを返します。これにより、スレーブはバイトの送信を終了するように指示され、続いてマスターがストップ条件を発生できるようになります。

受信されたデータがすみやかに読み出されず、FIFOが満杯になった場合は、マスターが溢れたデータに対してNACKを返します。

スレーブから4バイトを受信する場合のフローチャートを図19に示します。

I2C0CNTは、ユーザからアクセスできない内部カウンタへのロード・レジスタです。これは3ビットのカウンタであるため、マスターは一度に8バイトまで送信できることを意味します。ただし、送信シーケンスの実行中にI2C0ADRに書込みを行って、内部カウンタにデータを再ロードすることができます。

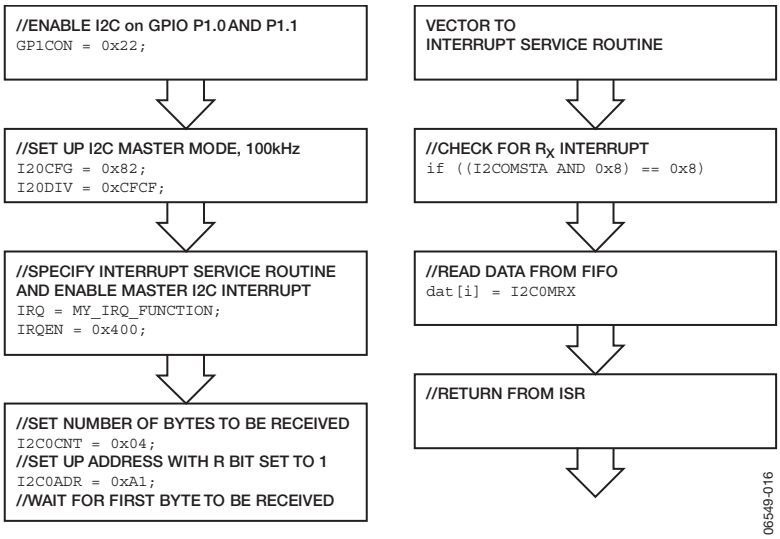

図19. マスター受信のフローチャート

スレーブ送信

スレーブはデータ送信の要求が出されるたびに割込みを発生しますが、最初の割込みはアドレスのACKの後、つまりFIFOのバイト0が送信される間に発生します。スレーブのTx FIFOにデータを事前にロードしておく必要があります。これをしておかないと、マスターから最初の読出し要求が出されるときにNACKが発生します。事前に2セットのデータがFIFOにロードされている場合は、アドレスのACKの後で1つの割込みが発生し、送信された各バイトのACKの後で次の割込みが発生します。

FIFOに事前にロードされているデータが1セットのみの場合は、アドレスのACKの後で2つの割込みが発生し、最初のデータが送信された後でFIFOが空になります。

1バイトが送信された後、マスターがデータを要求している限り割込みが発生します。マスターに1バイトが送信されるたびに、I2C0SSTAのビット2がセットされます。

スレーブがマスターからのデータ要求に応答する例を図21に示します。

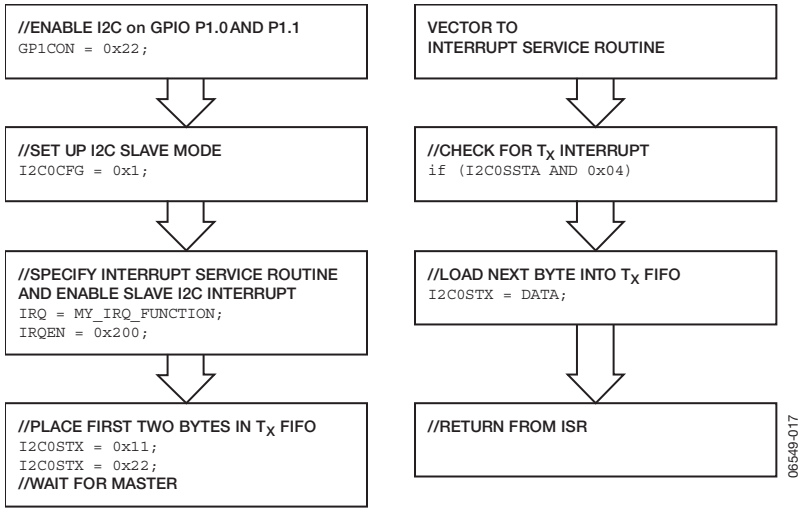

図20. スレーブ送信のフローチャート

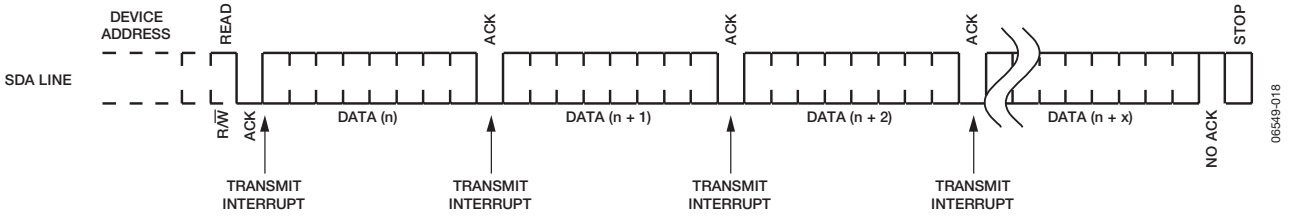

図21. スレーブ送信の例

I²Cレジスタの定義

I²Cペリフェラル・インターフェースは、合計15個のレジスタで構成されています。

- 送信／受信のマスター／スレーブMMRの4個のレジスタ（I2CxSRX、I2CxSTX、I2CxMRX、I2CxMTX）
- マスター／スレーブFIFOの3個のステータス・レジスタ（I2CxMSTA、I2CxSSTA、I2CxFSTA）
- 4つのスレーブ・アドレス、1つのマスター・アドレスバイト、1つのマスター・クロック分周器、1つのマスター受信データ・カウント、1つのI²Cコンフィギュレーション・レジスタの8個のコンフィギュレーション・レジスタ（I2CxID0、I2CxID1、I2CxID2、I2CxID3、I2CxADR、I2CxDIV、I2CxCNT、I2CxCFG）
- ジェネラルコール用の上記以外のレジスタについては、ここでは解説しません。

以下では、これらのレジスタのうち次の4個のレジスタについて詳しく説明します。

- I2CxDIV、クロック分周器レジスタ
- I2CxMSTA、マスター・ステータス・レジスタ、表2を参照
- I2CxCFG、I²Cコンフィギュレーション・レジスタ、表3を参照
- I2CxSSTA、スレーブ・ステータス・レジスタ、表4を参照

I2CxMSTA：マスター・ステータス・レジスタ

表2. I2CxMSTA MMRのビットの説明

ビット番号	説明
7	マスター送信FIFOのフラッシュ ユーザが設定して、マスターTx FIFOをフラッシュします。このビットでスレーブRx FIFOもフラッシュされます。 マスターTx FIFOがフラッシュされると、自動的にクリアされます。
6	マスター・ビジー マスターがビジーの場合に自動的にセットされます。 自動的にクリアされます。
5	アービトレーション・ロス マルチマスター・モードで別のマスターがバスを使用している場合にセットされます。 バスが利用できるようになるとクリアされます。
4	ノーACK スレーブ・デバイスがアドレスのアクノレッジを返さないと、自動的にセットされます。 I2C0MSTAレジスタの読出しによって自動的にクリアされます。
3	マスター受信IRQ データの受信後にセットされます。 I2C0MRXレジスタの読出しによって自動的にクリアされます。
2	マスター送信IRQ 送信の終了時にセットされます。 I2C0MTXレジスタへの書込みによって自動的にクリアされます。
1	マスター送信FIFOアンダーフロー マスター送信FIFOがアンダーフロー状態の場合に自動的にセットされます。 I2C0MTXレジスタへの書込みによって自動的にクリアされます。
0	マスターTx FIFOエンプティ マスター送信FIFOが空の場合に自動的にセットされます。 I2C0MTXレジスタへの書込みによって自動的にクリアされます。

I2CxDIV、クロック分周器レジスタ

DIVHとDIVLの2個の8ビット値を含む16ビットのレジスタです。このレジスタの値で、I²Cバスの速度を設定します。以下の式により設定します。

$$f_{serialclock} = \frac{f_{UCLK}}{(2+DIVH)+(2+DIVL)}$$

ここで、

f_{UCLK}はクロック分周器に送られる前のクロックです。
*DIVH*はクロックのハイレベル期間です。
*DIVL*はクロックのローレベル期間です。

ここから、100kHz動作の場合は次式が成立します。

DIVH＝*DIVL*＝0xCF（I2C0DIV＝0xCFCF）

400kHz動作の場合は次式が成立します。

DIVH＝0x28 *DIVL*＝0x3C（I2C0DIV＝0x283C）

I2CxCFG：I²Cコンフィギュレーション・レジスタ

表3. I2CxCFG MMRのビットの説明

ビット番号	説明
31～15	予備。これらのビットには0を書き込んでください。
14	ストップ割込みイネーブル ユーザが設定して、有効なスタート・アドレスと一致アドレスを受信した後ストップ条件を受信したときに割込みの発生ができるようにします。 ユーザがクリアして、ストップ条件を受信したときに割込みの発生をディスエーブルします。
13～12	予備。これらのビットには0を書き込んでください。
11	SCLストレッチ・イネーブル（SCLをローレベルに保持） ユーザがセットして、SCLラインのストレッチングをイネーブルします。SCLがすでにローレベルになっている場合または次にローレベルになる場合に、このビットがI²Cインターフェースに対しSCLをローレベルに保持するよう指示します。 ユーザがクリアして、SCLラインのストレッチングをディスエーブルします。
10	予備。このビットには0を書き込んでください。
9	スレーブTx FIFO要求割込みイネーブル ユーザがセットして、スレーブTx FIFOの要求割込みをディスエーブルします。 ユーザがこのビットをクリアすると、R/$\overline{W}$ビットに対応するクロックの立下がりエッジ直後に割込み要求が発生します。これによって、スレーブTx FIFOが空であれば、ユーザはスレーブTx FIFOにデータを書込むことができます。400ksps時およびコア・クロックの動作速度が41.78MHzのときに、ユーザは割込み遅延を考慮した適切な処理を行うため、45クロック・サイクルを使用することができます。
8	ジェネラルコール・ステータスビット・クリア ユーザがセットして、ジェネラルコール・ステータスビットをクリアします。 ジェネラルコール・ステータス・ビットがクリアされた後、ハードウェアによって自動的にクリアされます。
7	マスター・シリアル・クロック・イネーブルビット ユーザがセットして、マスター・モードでシリアル・クロックの発生をイネーブルします。 ユーザがクリアして、マスター・モードでシリアル・クロックの発生をディスエーブルします。
6	ループバック・イネーブルビット ユーザがセットして、内部で遷移を受信先に接続し、ユーザ・ソフトウェアをテストします。 ユーザがクリアして、ノーマル・モードの動作にします。
5	バックオフ・スタート・ディセーブルビット マルチマスター・モード時にユーザがセットします。アービトレーションが失われると、ただちにマスターは送信を再開しようとします。 ユーザがこのビットをクリアすると、バックオフ・スタートが有効になります。アービトレーションが失われた後、マスターは待機してから送信を再開します。
4	ハードウェア・ジェネラルコール・イネーブル このビットとジェネラルコール・イネーブルビットをセットして、ジェネラルコール（アドレス0x00）と1つのデータバイトを受信すると、デバイスは受信レジスタとI2C0ALTのデータ内容を比較してチェックします。これらのデータが一致すれば、デバイスはそのハードウェア・ジェネラルコールを受信します。デバイスがマスター・デバイスのアテンションを緊急に必要とし、かつ相手のマスター・デバイスを知らない場合にこの機能を使用します。ADuC702xデバイスはこれらのアドレスを待ちます。アテンションを要求しているデバイスは、そのアドレスをメッセージの中に書込みます。全マスターがこれをリスニングし、そのデバイスに対する処理を知っているマスターがスレーブにコンタクトして適切に処理します。I²Cバス規格バージョン2.1（2000年1月発行）に従って、I2C0ALTレジスタのLSBには必ず1を書き込んでください。
3	ジェネラルコール・イネーブルビット このビットをセットすると、スレーブ・デバイスはアドレス0x00（書込み）のI²Cジェネラルコールに対するACKを送信できるようになります。そうすると、デバイスはデータビットを認識します。I²Cインターフェースは、データバイトとして0x06、すなわち「ハードウェアからリセットおよび書込み可能なスレーブ・アドレス部分」を受信すると、I²Cバス規格に従ってリセットを行います。このコマンドは、I²Cシステム全体をリセットするときに使うことができます。ジェネラルコールが発生するときは、必ずジェネラルコール割込みステータスビットがセットされます。リセット後にI²Cインターフェースを設定して適切に対応することは、ユーザの責任で行う必要があります。データバイトに0x04、すなわち「ハードウェアから書込み可能なスレーブ・アドレス部分」を受信すると、ジェネラルコールが発生するときに必ずジェネラルコール割込みステータスビットがセットされます。ユーザの責任でデバイス・アドレスを再設定して適切に対応する必要があります。
2	予備
1	マスター・イネーブルビット ユーザがセットして、マスターのI²Cチャンネルをイネーブルします。 ユーザがクリアして、マスターのI²Cチャンネルをディスエーブルします。
0	スレーブ・イネーブルビット ユーザがセットして、スレーブのI²Cチャンネルをイネーブルします。スレーブ転送シーケンスを監視して、I2C0ID0、I2C0ID1、I2C0ID2、I2C0ID3にデバイス・アドレスがないか調べます。デバイス・アドレスを確認すると、デバイスはそのスレーブ転送シーケンスに参加します。 ユーザがクリアして、スレーブのI²Cチャンネルをディスエーブルします。

I2CxSSTA：スレーブ・ステータス・レジスタ

注：ステータス・レジスタを読み出すと、その内容が変更されます。読出しは1回限りにし、ISR時にその値を変数として保存してください。

表4. I2CxSSTA MMRのビットの説明

ビット番号	説明
31～15	予備。これらのビットには0を書き込んでください。
14	スタート・デコードビット 有効なスタート・アドレスと一致アドレスをデバイスが受信した場合に、ハードウェアによってこのビットがセットされます。 I²Cストップ条件またはI²Cジェネラルコール・リセットのいずれかによってクリアされます。
13	繰返しスタート・デコードビット 有効な繰返しスタート・アドレスと一致アドレスをデバイスが受信した場合に、ハードウェアによってこのビットがセットされます。 I²Cストップ条件、I2CxSSTAレジスタの読出し、またはI²Cジェネラルコール・リセットのいずれかによってクリアされます。
12～11	IDデコードビット 00　受信したアドレスがIDレジスタ0と一致 01　受信したアドレスがIDレジスタ1と一致 10　受信したアドレスがIDレジスタ2と一致 11　受信したアドレスがIDレジスタ3と一致
10	スタート後のストップおよび一致アドレス割込み 前のI²Cスタート条件および一致アドレスの後、I²Cストップ条件をスレーブ・デバイスが受信した場合にハードウェアによってこのビットがセットされます。 I2CxSSTAレジスタの読出しによってクリアされます。
9～8	ジェネラルコールID 00　ジェネラルコールなし 01　ジェネラルコールのリセットおよびプログラム・アドレス 10　ジェネラルコールのプログラム・アドレス 11　ジェネラルコールの代替一致ID
7	ジェネラルコール割込み
6	スレーブ・ビジー スレーブがビジーの場合に自動的にセットされます。 自動的にクリアされます。
5	ノーACK マスターがデータを要求し、利用できるデータがない場合にセットされます。 自動的にクリアされます。
4	スレーブ受信FIFOオーバーフロー スレーブ受信FIFOがオーバーフローの場合に、自動的にセットされます。 I2C0SRXの読出しによって自動的にクリアされます。
3	スレーブ受信IRQ データの受信後にセットされます。 I2C0SRXレジスタの読出しによって自動的にクリアされます。
2	スレーブ送信IRQ 送信の終了時にセットされます。 I2C0STXレジスタへの書込みによって自動的にクリアされます。
1	スレーブ送信FIFOアンダーフロー スレーブ送信FIFOがアンダーフローの状態の場合に自動的にセットされます。 I2C0STXレジスタへの書込みによって自動的にクリアされます。
0	スレーブ送信FIFOエンプティ スレーブ送信FIFOが空の場合に自動的にセットされます。 I2C0STXレジスタへの書込みによって自動的にクリアされます。

シリアルEEPROMプロトコルの実行

ここでは、I²Cアドレスが1つのみの場合にシリアルEEPROM仕様で提供する次の5つのコマンドの実行について説明します。

- カレント・アドレス読出し
- ランダム読出し
- ランダム書込み
- シーケンシャル読出し
- シーケンシャル書込み

割込みサービス・ルーチンでは、ステータス・レジスタの読出しを1回行い、その値を保存する必要があります。割込みサービス・ルーチンのフローチャートを図22に示します。

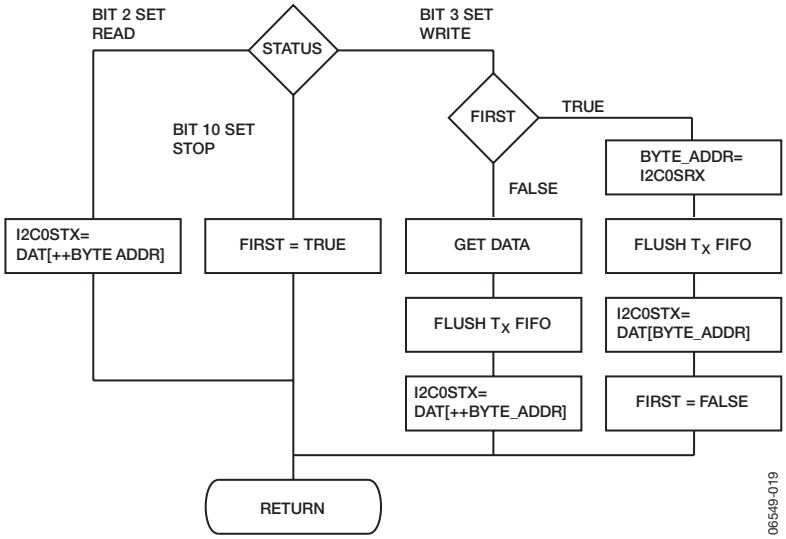

図22. スレーブ割込みサービス・ルーチン

コンパニオン・コードについては、「AN-895 Companion Code.zip」を参照してください。

コマンド・コード

I²Cの設定

```
I2CCFG = 0x4001; // Enable slave, enable STOP detect
I2C0ID0 = 0xA0; // Slave ID
I2C0STX = dat[0]; // Set initial data
```

変数の初期化

```
Byte_addr = 0
First = 1
```

I²Cスレーブ割込みのイネーブル

```
IRQEN = 0x200; // I2C0 Slave Interrupt
while (1) // Wait for interrupt
```


AN06549-0-1/07(0)-J