Vol 55, No 4—November 2021

ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™
AnalogDialogue

I²Cプライマ、SMBus、PMBusの仕様を学ぶ

著者：Mary Grace Legaspi、ファームウェア・エンジニア
Eric Peña、ファームウェア・エンジニアリング・マネージャ

概要

I²C（Inter-Integrated Circuit）はシリアル通信プロトコルの一種です。デバイス間、特に2つ以上の異なる回路間で通信を実現するためによく使用されます。最も広く普及しているのはI²Cプライマですが、それを基にした派生プロトコルも使われています。本稿では、まず、I²Cプライマの基本的な機能と仕様について説明します。特に、通信機能を利用する際の適切な実装方法について詳しく解説します。その上で、I²Cの派生サブセットであるSMBus（System Management Bus）とPMBus（Power Management Bus）を取り上げ、両者の違いとそれぞれの有用性について説明します。これら3つのプロトコルは、様々なユーザの要求に対応することを目的とし、それぞれに固有の機能を備えています。

なぜI²Cは重要なのか？

I²Cには1つの大きな特徴があります。それは、システム内の数多くのノードの間で、シンプルかつ柔軟性の高い双方向の通信が行えるというものです。I²Cでは、わずか2本のワイヤを使用するだけで、双方向で情報をやり取りすることができます。つまり、複雑さを回避することが可能です。また、複数のシステムICをメインとノードに振り分け、それらの間で通信が行えるように構成することもできます。I²Cは、システムや電力を管理したい場合にも有用です。つまり、短期間で品質の高い製品を開発するために活用できます。

「コミュニケーションは、コミュニケーションをとるための努力をする人に対して有効に働きます。」
- John Powell氏

通信プロトコルは、デバイス間で通信を実現する上で大きな役割を担います。システムの要件に応じて様々に設計され、円滑な通信を実現するための具体的な規則が定められます。

読者の中には、LEDを使用したディスプレイ、センサー、加速度センサー・モジュールなどから成るシステムを構築した経験を持つ方もいらっしゃるでしょう。その場合、おそらくはI²Cも利用したのではないでしょうか。I²Cは、単一のメインのICに複数のノードのICを、あるいは複数のメインのICに複数のノードのICを接続する機能をサポートしています。この機能を利用すれば、1つのマイクロコントローラによって、1つのメモリ・カードにデータを記録したり、1つの液晶ディスプレイにテキストを表示したりすることが可能になります。

最も広範に普及しているのはI²Cプライマですが、残り2つのバリエーションであるSMBusとPMBusも広く使われています。それぞれ、システム・アプリケーション、電源アプリケーションで使用されることを想定して仕様が策定されています。

I²Cは、もともとICとICの間で通信を行うためのものとして定義されました。複数のメイン、複数のノードに対応し、シリアル・バスを使用した同期通信を実現するためのハードウェア通信用のプロトコルです。ここで言う同期通信では、データをやり取りする2つ（あるいはそれ以上）のデバイスがクロック・ラインを共有します。I²Cは、低速の周辺ICをプロセッサやマイクロコントローラに接続するために広く使用されています。I²Cのバスは、同一基板上のコンポーネントの間で簡単に通信が行えるようにPhilipsによって設計されました。

I²Cプライマの詳細

まずは、I²Cプライマの詳細について説明します。

インターフェース

I²Cプライマの通信では、SDA（Serial Data）ライン、SCL（Serial Clock）ライン、共通のグラウンドを使用します。このような最小限の配線で、あらゆる通信を実現できるようになっています。

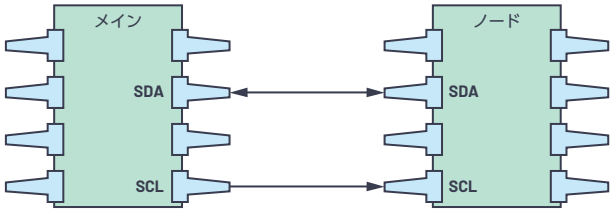

図1. I²Cプライマによる通信。
ICとICが相互に直接通信を行います。

I²Cに対応する各デバイスは、以下の2本のワイヤを備えています。

- SDA：メインとノードがデータを送受信するために使用するラインです。
- SCL：クロック信号の伝送に使用するラインです。SCLは、常にI²Cのメインによって生成されます。I²Cプライマの仕様では、クロック信号のローとハイの位相の最小期間が定められています。

つまり、I²Cでは、各ICが備えるSDA、SCLという双方向のラインを使用するだけで、シンプルな通信が実現されます。

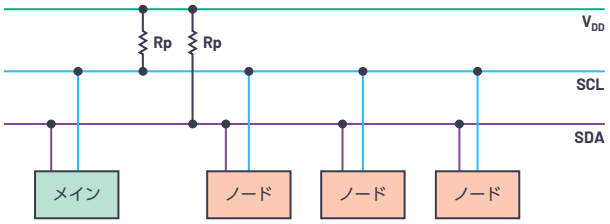

図2. I²Cにおける
プルアップ抵抗の接続

I²Cプライマで最も重要なハードウェア部品は、SDA/SCLのラインで使用するプルアップ抵抗です。I²Cに対応するデバイスは、オープン・コレクタまたはオープン・ドレインのピンによってバスに接続され、ラインの電圧をローに引き下げます。データが転送されていない期間は、I²Cのバスのラインはハイになります。これはアイドル状態に相当します。各ラインは、受動的にハイに引き上げられることになります。データの転送は、ラインをトグルする、つまりローに引き下げてからハイに戻すことによって行います。各ビットは、クロックの立ち下がりエッジによって捉えられます。

オープン・ドレイン出力に接続されたプルアップ抵抗（図2のRp）は、適切にハイを出力するために使用されます。同抵抗は、ハイの状態を実現するために必要な電圧（図2のV_{DD}）と出力ピンの間に接続されます。

V_{CC}やV_{DD}が標準的な値（5V）である場合、プルアップ抵抗の値としては4.7kΩが最もよく使用されます。

ちなみに、シールドされた2 AWGのツイスト・ペア・ケーブルでは、静電容量が100pF/m～240pF/mとなります。そのため、I²Cの通信リンクにおけるバス長は、100kボーの場合には最長で約1m、10kボーの場合には最長で約10mとなります。通常、シールドのないケーブルでは、静電容量ははるかに低い値になります。ただ、その種のケーブルは、シールドの施された筐体内でのみ使用するべきです。

表1は、I²Cプライマの主な仕様についてまとめたものです。

表1. I²Cの主な仕様

項目	仕様
ワイヤの数	2
最高速度	標準モード：100kbps ファスト・モード：400kbps ハイ・スピード・モード：3.4Mbps ウルトラファスト・ モード：5Mbps
同期か、非同期か？	同期
シリアルか、パラレルか？	シリアル
メインの最大数	無制限
ノードの最大数	1008

理論的には、アドレス指定モードにおける最大ノード数は2^7または2^{10}です。但し、16個のアドレスは予約済みとなっています。

I²Cは同期型の通信プロトコルです。各ビットの出力については、メインとノードの間でクロック信号を共有し、ビットのサンプリングを行うことで同期がとられます。クロック信号は、常にメインによって制御されます。

ノードの予約済みアドレス

上述したように、I²Cでは16個の予約済みアドレスが定められています。それらのアドレスは、0000 XXXまたは1111 XXXのいずれかのパターンに対応しています。表2に、予約済みアドレスの概要を示しました。

表2. 予約済みのアドレス

ノードのアドレス	R/W ビット	説明
0000 000	0	ゼネラル・コール・アドレス
0000 000	1	スタート・バイト
0000 001	X	CBUSアドレス
0000 010	X	異なるバス・フォーマット用に予約済み
0000 011	X	将来生じ得る目的のために予約済み
0000 1XX	X	ハイ・スピード・モードのメイン・コード
1111 1XX	X	将来生じ得る目的のために予約済み
1111 0XX	X	10ビットのノード・アドレス

I²Cの動作

I²Cにおいて、データはメッセージとして転送されます。メッセージはデータ・フレームに分割されます。読み出しと書き込みのプロトコルには、ノードのバイナリ・アドレスを備えた**アドレス・フレーム**、転送するデータを含む**データ・フレーム**、**スタート／ストップ・コンディション**、**リピート・スタート・ビット**、**読み出し／書き込みビット**、各データ・フレーム間の**ACK/NACK（アクノリッジ／ノット・アクノリッジ）ビット**が含まれます。

タイミングの仕様

I²Cを採用する場合、バスの要件に適合したICを設計するためには、タイミングの仕様を遵守する必要があります。また、遵守すべきタイミング仕様は、使用するデータ転送レートに応じて異なります。データを正しく転送するには、メインとノードがその仕様を遵守していなければなりません。

表3に、タイミングの仕様を定める際に使われるシンボルとパラメータを示しました。

表3. タイミング仕様の表現に使われるシンボルとパラメータ

シンボル	パラメータ	単位
f_{SCL}	SCLのクロック周波数	kHz
$t_{HD(STA)}$	（リピート）スタート・コンディションのホールド時間	マイクロ秒
t_{LOW}	SCLにおけるローの期間	マイクロ秒
t_{HIGH}	SCLにおけるハイの期間	マイクロ秒
$t_{SU(STA)}$	リピート・スタート・コンディションのセットアップ時間	マイクロ秒
$t_{HD(DAT)}$	データのホールド時間	マイクロ秒
$t_{SU(DAT)}$	データのセットアップ時間	ナノ秒
t_r	SDAの信号の立ち上がり時間	ナノ秒
t_f	SDAの信号の立ち下がり時間	ナノ秒
$t_{SU(STO)}$	ストップ・コンディションのセットアップ時間	マイクロ秒

転送に使用するサブプロトコル

バスを介した転送では、読み出し、書き込みのうちいずれかの処理が行われます。読み出し／書き込みのプロトコルは、一連のサブプロトコルに基づいて構築されています。スタート／ストップ・コンディション、リピート・スタート・ビット、アドレス・バイト、データ転送ビット、ACK/NACKビットなどのサブプロトコルです。以下、それぞれについて説明します。なお、以下に示す図の白い部分はノードを、青い部分はメインを表します。

スタート・コンディション

スタート・コンディションは、その名のとおり、常に転送の開始時に発生します。メインによって処理が開始されると、バス上にあるアイドル状態のノードがウェイクアップします。SCLのラインの電圧レベルがハイからローに切り替わる前に、SDAのラインはハイからローに切り替わります（図4）。

図3. メッセージの構成

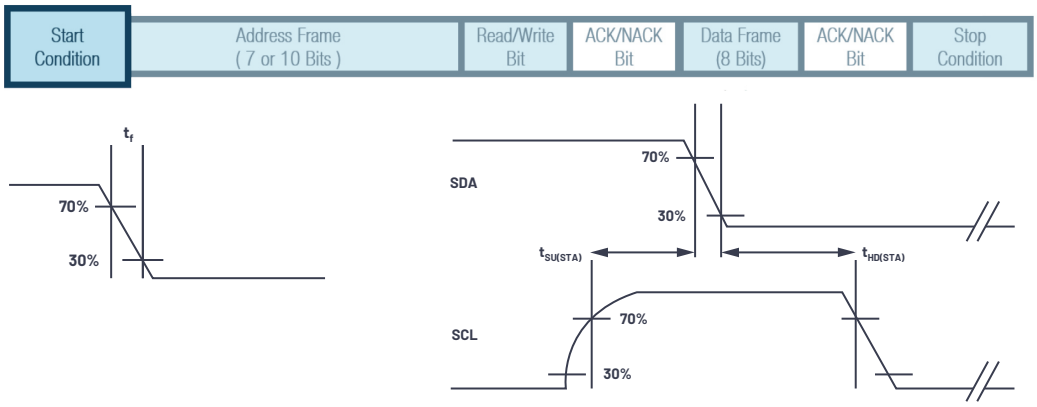

図4. スタート・コンディション

リピート・スタート・コンディション

データの転送中は、ストップ・コンディションを発行することなく、スタート・コンディションを繰り返すことができます。これは、リピート・スタートと呼ばれる特別な動作です。データの転送方向の変更、転送の試行の繰り返し、複数のICの同期、シリアル・メモリの制御などに使用されます（図5）。

アドレス・フレーム

アドレス・フレームには、利用可能性に応じて7ビットまたは10ビットのシーケンスが含まれます（図6）。詳細についてはデータシートをご覧ください。

I²Cには、SPI（Serial Peripheral Interface）で使われるようなノード・セレクト・ラインは存在しません。そのため、データ送信の対象となるノードに、データを送信することを知らせるための方法が別途必要です。その方法としてアドレスの指定が使われます。新しいメッセージのスタート・ビットの直後に位置するフレームは、常にアドレス・フレームになります。

メインは、接続されているすべてのノードに対し、通信の対象となるノードのアドレスを送信します。それを受けた各ノードは、メインから送信されたアドレスと自身のアドレスを比較します。アドレスが一致している場合、電圧がローのACKビットをメインに送り返します。アドレスが一致していない場合には、ノードは何も行わず、SDAラインはハイのままになります。

読み出し／書き込みビット

アドレス・フレームの最後の1ビットは、読み出し／書き込みビットです。これは、メインがそのノードにデータを書き込もうとしているのか、そのノードからデータを読み出そうとしているのかを知らせるためのものです。メインがノードにデータを送信しようとしている場合、読み出し／書き込みビットの電圧レベルはローになります。メインがノードにデータを要求している場合には、このビットの電圧レベルはハイになります（図7）。

ACK／NACKビット

各フレームの次のビットは、ACK/NACKビットとなります。アドレス・フレームまたはデータ・フレームが正常に受信されたら、受信側のデバイスから送信側のデバイスにACKビットが返されます（図8）。

図5. リピート・スタート・コンディション

図6. アドレス・フレーム

図7. 読み出し／書き込みビット

図8. ACK/NACKビット

データ・フレーム

メインがノードからのACKビットを検出したら、最初のデータ・フレームを送信する準備が整ったことになります。データ・フレームのビット長は常に8ビットで、最上位ビットから順に送信されます。各データ・フレームの直後には、そのフレームが正常に受信されたことを確認するためのACK／NACKビットが続きます。次のデータ・フレームは、メインまたはノードのいずれかで（どちらがデータを送信しているかによる）ACKビットが受信されてから送信する必要があります（図9）。

ストップ・コンディション

すべてのデータ・フレームが送信されたら、メインはノードに対してストップ・コンディションを送信します。それにより、転送が停止します。ストップ・コンディションは、SCLラインがローからハイに遷移した後に、SDAラインの電圧がローからハイに遷移することによって表されます。そのとき、SCLラインはハイのままになっている必要があります（図10）。

書き込み時のステップ

単一のロケーションに対してデータの書き込みを行う場合、図11のような形で転送処理が行われます。以下、各ステップについて説明します。

【ステップ1】

メインは、接続されているすべてのノードに対してスタート・コンディションを送信します。具体的には、SCLラインの電圧レベルをハイからローに切り替える前に、SDAラインをハイからローに切り替えます。

【ステップ2】

メインは、各ノードに対し、通信の対象となるノードに対応する7ビット／10ビットのアドレスに書き込みビットを付加して送信します。例えば、7ビットのアドレスが0x2Dであるとします。書き込みビットに相当する0を追加すると、0x5Aという8ビットのデータになります。

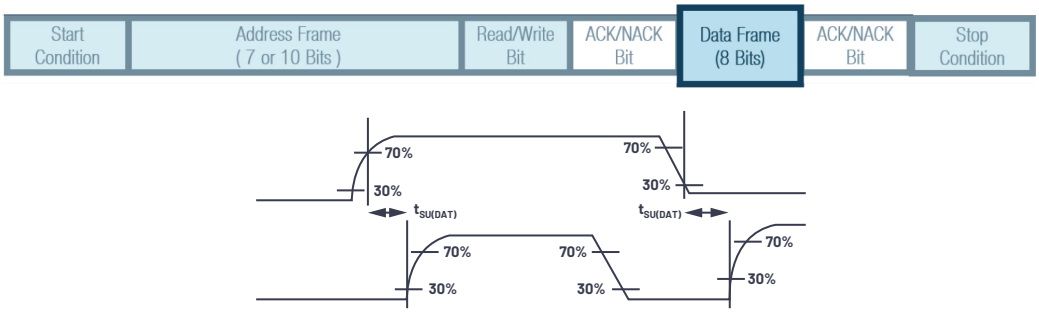

図9. データ・フレーム

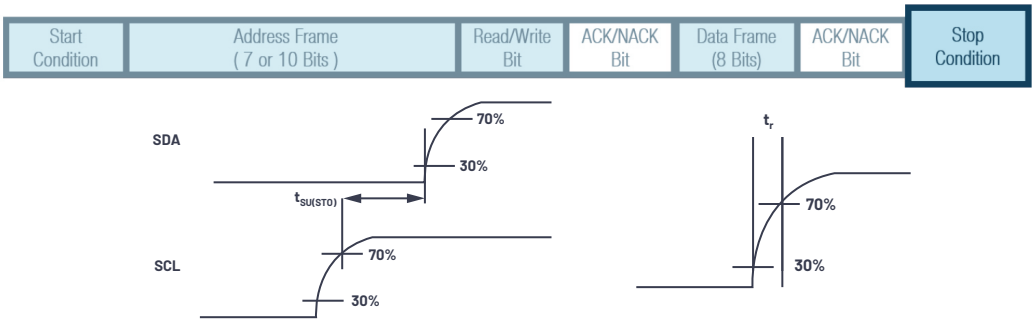

図10. ストップ・コンディション

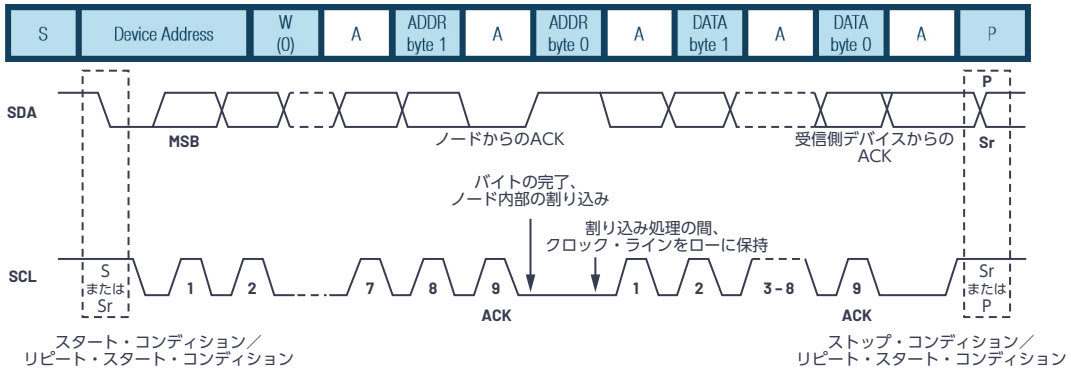

図11. 単一のロケーションに対する書き込み処理。データシートに記載されている例です。

【ステップ3】
各ノードは、メインから送信されたアドレスを自身のアドレスと比較します。アドレスが一致している場合、そのノードはSDAラインを1ビット分ローに引き下げることによってACKビットを返します。アドレスが一致していない各ノードは、SDAラインをハイのまま維持します。ACKの場合には、SCLの9番目のパルスの間にSDAラインをローに引き下げます。NACKの場合にはハイのまま維持します。

【ステップ4】
メインはデータ・フレームを送信／受信します。

【ステップ5】
各データ・フレームの転送が完了したら、受信側のデバイスは送信側のデバイスに対してACKビットをもう1つ返します。それにより、フレームを正常に受信したことが通知されたことになります。

【ステップ6】
データの転送を停止するために、メインはノードにストップ・コンディションを送信します。具体的には、SDAをハイに切り替える前にSCLをハイに切り替えます。

読み出し時のステップ

読み出しを行う際には、図12のような形で転送処理が行われます。以下、各ステップについて説明します。

【ステップ1】
メインは、接続されているすべてのノードに対してスタート・コンディションを送信します。具体的には、SCLラインの電圧レベルをハイからローに切り替える前に、SDAラインをハイからローに切り替えます。

【ステップ2】
メインは、各ノードに対し、通信の対象となるノードに対応する7ビット／10ビットのアドレスに書き込みビットを付加して送信します。例えば、7ビットのアドレスが0x2Dであるとします。書き込みビットに相当する0を追加すると、0x5Aという8ビットのデータになります。

【ステップ3】
各ノードは、メインから送信されたアドレスを自身のアドレスと比較します。アドレスが一致している場合、そのノードはSDAラインを1ビット分ローに引き下げることによってACKビットを返します。アドレスが一致していない各ノードは、SDAラインをハイのまま維持します。

【ステップ4】
冒頭のスタート、アドレスの指定、ACKの処理が終わった時点で、メインはノードと指定すべきアドレスを既に把握しています。そのため、デバイスによっては、トランザクションをクリーンアップするために、リピート・スタート・コンディションを受け取ることがあります。これは、読み出しの場合にだけ当てはまることに注意してください。

【ステップ5】
メインは、各ノードに対し、通信の対象となるノードに対応する7ビット／10ビットのアドレスに読み出しビットを付加して送信します。例えば、7ビットのアドレスが0x2Dであるとします。読み出しビットに相当する1を追加すると、0x5Bという8ビットのデータになります。

【ステップ6】
各ノードは、メインから送信されたアドレスを自身のアドレスと比較します。アドレスが一致している場合、そのノードはSDAラインを1ビット分ローに引き下げることによってACKビットを返します。アドレスが一致していない各ノードは、SDAラインをハイのまま維持します。

【ステップ7】
ACKビットの処理後、メインはノードからデータ・フレームを受信します。

【ステップ8】
各データ・フレームの転送が完了したら、メインは送信側のデバイスに対してACKビットをもう1つ返します。それにより、フレームを正常に受信したことが通知されたことになります。なお、読み出しのリクエストが既に完了している場合にはNACKビットを返します。

【ステップ9】
データの転送を停止するために、メインはノードにストップ・コンディションを送信します。具体的には、SDAをハイに切り替える前にSCLをハイに切り替えます。

単一のメイン、複数のノード

I^2Cプライマでは、ノードをアドレスで指定する手法を採用しています。そのため、単一のメインによって複数のノードを制御することが可能です。7ビットのアドレスでは、128（2^7）種のアドレスを使用することができます。10ビットのアドレスが使われることはあまりありませんが、1024（2^{10}）種のアドレスを使用できることになります。単一のメインに複数のノードを接続する際には、4.7kΩのプルアップ抵抗を使ってSDAラインとSCLラインをV_{CC}に接続します。

図12. 単一のロケーションからの読み出し処理。
データシートに記載されている例です。

複数のメイン、複数のノード

I²Cプライマでは、複数のメインに単一のノードまたは複数のノードを接続することも可能です。但し、1つのシステム内に複数のメインが存在する場合には、SDAラインを介して2つのメインが同時にデータを送受信するという状況が生じる可能性があります。この問題を回避するために、各メインはメッセージを送信する前に、SDAラインがローであるかハイであるかを検出する必要があります。

SDAラインがローである場合、いずれかのメインがバスを制御していることになります。したがって、それ以外のメインはメッセージの送信を控える必要があります。SDAラインがハイである場合には、メッセージを転送して構いません。複数のメインに複数のノードを接続するためには、図13のような回路構成を採用し、4.7kΩのプルアップ抵抗によってSDAラインとSCLラインをV_{CC}に接続します。

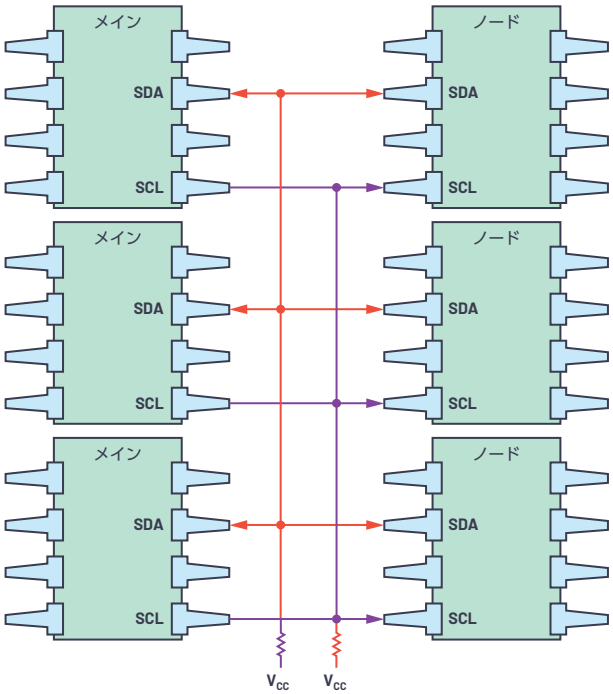

図13. 複数のメインに
複数のノードを接続する場合の構成

アービトレーション

上述したように、I²Cプライマでは、複数のメインを1つのバスに接続して同時に動作させることができます。その場合、SDAとSCLのスタート・コンディションやストップ・コンディションを常に監視することで、バスがアイドル状態にあるか否かを判定する必要があります。バスがビジーの状態である場合、メインは、ストップ・コンディションによってバスが再度フリーになったことが判明するまで、転送処理を保留の状態に保ちます。

しかし、2つのメインが同時に転送を開始するということもあり得ます。転送を行っている間、メインは常にSDAとSCLを監視しています。1つのメインが、SDAがハイであるべきときにローであることを検出したとします。その場合、そのメインはもう1つのメインがアクティブであると見なし、即座に転送を停止します。このプロセスをアービトレーション（調停）と呼びます。

どちらのメインもスタート・ビットを生成し、それぞれの転送処理を開始します。各メインが偶然同じロジック・レベルを選択した場合には、何も生じません。各メインが異なるロジック・レベルに設定しようとした場合には、信号をローにしようとしているメインの設定が優先されます。このメインを勝者と呼ぶことにしましょう。それ以外のメイン（敗者）はロジックの不一致を検出して、転送を中止することになります。

以下、このアービトレーションのシンプルさと有効性についてまとめておきます。

- 勝者は中断することなく転送を続けます。データの破損やドライバの競合が生じることはなく、トランザクションを再起動する必要もありません。
- 理論的には、敗者はアービトレーションのプロセスの間、ノードのアドレスを監視することができます。その敗者からアドレスを指定されたノードであれば、適切な応答を返すことが可能です。
- 競合するメインがどちらも同じノードからのデータを要求している場合には、アービトレーションのプロセスは、どちらかのトランザクションを無駄に中断することはありません。不一致がないことが検出されたら、ノードは複数のメインによって受信できるようにデータをバスに出力します。

クロック・ストレッチ

クロック・ストレッチは、クロック同期とも呼ばれる機能です。I²Cプライマの通信プロトコルでは、クロックの速度と信号は、常にメインによって設定／生成されることになっています。メインによって生成された信号により、メインとノードの接続において同期がとられます。ここで、ノードあるいはサブノートの準備が十分に整っておらず、メインから生成されたクロックを受信する前に速度を落とす必要があったとしましょう。このような処理を実現する仕組みがクロック・ストレッチです。

クロック・ストレッチの期間中、ノードはバスの速度を低下させるためにクロックをローに保持するということが行えます。一方、メインは、クロックがハイになった後にクロック信号を再読み出しする必要があります。つまり、ラインがハイになるまでメインは待機しなければなりません。

なお、I²Cプライマの仕様では、クロック・ストレッチにおけるタイムアウトについては規定されていません。つまり、いずれのデバイスが必要に応じてSCLをローに保持して構わないということになります。この点には注意してください。

帯域幅に及ぶ影響

クロック・ストレッチは一般的な手法です。但し、帯域幅に影響が及ぶという欠点を抱えています。クロック・ストレッチを使用すると、共有バスの総帯域幅が大幅に低下する可能性があるのです。クロック・ストレッチを使用する場合でも、バスは高速で信頼性が高くなければなりません。特に複数のデバイスでI²Cのバスを共有する場合には、クロック・ストレッチを使用した際に生じる影響を想定して対処しておく必要があります。

クロック・ストレッチにより、ノードはメインを強制的に待機状態にすることができます。ノードは、データを管理するための時間が足りない場合にクロック・ストレッチを実行します。例えば、受信したデータを格納したり、残る1バイトのデータを送信する準備を行ったりするといった場合です。通常、これはノードが1バイトのデータを受信してACKの処理を行った後に生じます。

クロック・ストレッチを必要とするノード

クロック・ストレッチが必要か否かは、ノードの機能に依存します。以下に2つの例を示しておきます。

- マイクロプロセッサやマイクロコントローラなどのICでは、割り込み処理、データの受信／管理、適切な機能の実行といった処理のためにより多くの時間が必要になることがあります。
- EEPROMのようなシンプルなデバイスでは、内部でデータを処理することはありません。その種のICが備える本質的な機能を実行するために、クロック・ストレッチが必要になることはありません。

データシートにおけるI²C関連の記載例

データシートは、メーカーごとに様々なアプローチで作成されます。図14に、マイクロコントローラのデータシートのサンプルを示しました。また、図15にはマイクロコントローラのメモリ・マップの例を示しておきます。

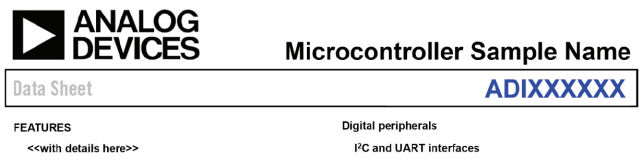

図14. マイクロコントローラのデータシートのサンプル

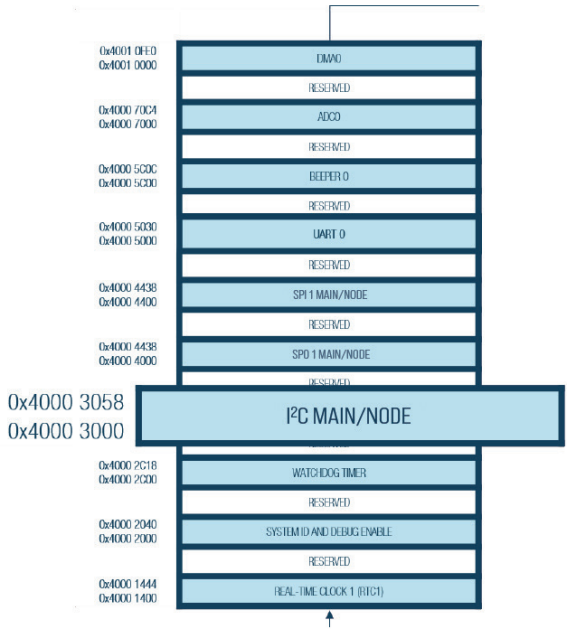

図15. マイクロコントローラのメモリ・マップ

表4に示したのは、I²Cのレジスタに関する記載例です。最も一般的な例だと言えますが、使用される名称や説明はデータシートによって異なるかもしれません。ただ、機能や使用法は一般的なものだと言うことができます。

表4. I²Cのレジスタに関する説明

名称	説明
I2C_ADDR1	メインのアドレス（バイト1）
I2C_ADDR2	メインのアドレス（バイト2）
I2C_BYT	スタート・バイト
I2C_ID	ノードのアドレス（デバイスのID）
I2C_MCTL	メインの制御
I2C_MRX	メインの受信データ
I2C_SCTL	ノードの制御
I2C_SRX	ノードの受信
I2C_STAT	メインとノードのFIFOの状態

I²Cの構成については使用法によって異なる部分があります。表5に、I²C用の基本的なドライバに対応するAPI（Application Programming Interface）の要件についてまとめました。

表5. I²C用のドライバに関する記述

メイン	ノード
初期化	
Txハンドラ	Txハンドラ
Rxハンドラ	Rxハンドラ
イベントの割り込み	
エラーの割り込み	

システム管理に使われるSMBus

SMBusは、I²Cプライマから派生したプロトコルです。重要なパラメータの監視を必要とするアプリケーションでよく使用されています。代表的な用途は、コンピュータのマザーボードや組み込みシステムのアプリケーションです。温度、供給電圧、ファン、制御用ICなどの監視に適した形で仕様が定義されています。

先述したように、I²CはPhilipsによって1980年に開発されました。SMBusは、それと同様の2線式のバス規格です。2つの信号線としてはクロック・ラインのSMBCLKとデータ・ラインのSMBDATを使用します（図16）。I²CプライマとSMBusは互いに互換性を持ちますが、以下に示すような大きな相違点があります。

- SMBus のロジック・レベルの閾値は固定であり、デバイスの電源電圧には比例しません。そのため、様々な電源電圧のデバイスを同じ基盤の上で動作させられることになります。例えば、SMBus に対応する 1 つのバスによって、電源電圧が 1.8V、3.3V、5V のデバイスに対応するといったことが可能です。
- I²C プライマも SMBus も、最高 100kHz で動作します。但し、I²C プライマには 400kHz と 2MHz のバージョンも用意されています。
- SMBus では、最も低速のクロックを供給し、1 回のトランザクションでクロック・ストレッチが生じる回数を削減できるようになっています。タイムアウトの制限に反する状態が発生したら、バスを再起動できるようにするために、SMBus に対応する全デバイスは I/O のロジックをリセットします。このような動作により、バスの堅牢性を高めています。
- I²C プライマにはタイムアウトについての規定はありません。それに対し、SMBus にはタイムアウト機能が定義されています。クロック速度が最も遅い 10kHz という条件では、35 ミリ秒でタイムアウトが発生します。
- パケット・エラー・チェック（PEC：Packet Error Checking）の起源は、SMBus にあります。各トランザクションの最後に、パケット・エラー・コード・バイトが追加されます。
- I²C プライマと SMBus には、転送タイプ、アラート・ライン、サスペンド・ライン、パワー・ダウン／パワー・アップなどの面でも違いがあります。

SMBusに対応するデバイス（SMBusデバイス）は、デバイスがどのような状態にあるかに関わらず、自身のアドレスを受信するたびにACKビットを送信しなければなりません。それにより、メインは、どのデバイスがバス上でアクティブになっているのかを正確に判断することができます。

SMBusのすべてのトランザクションは、SMBusの仕様で規定されたプロトコルのうち1つに従って実行されます。

SMBusには、SMBALERT#というオプションの信号が用意されています。これを使用することで、ノードは、障害が発生していることの報告など、メイン向けの情報が存在することを、メインあるいはシステムのホストに対して迅速に通知することができます。

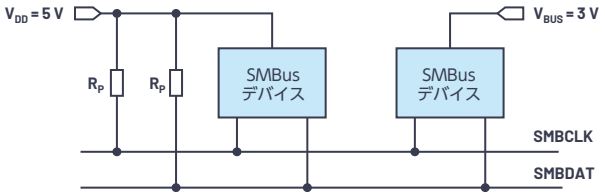

図16. SMBusのトポロジ

SMBusのプルアップ回路

SMBusにも、プルアップ用の回路が必要です。図17にその概念図を示します。

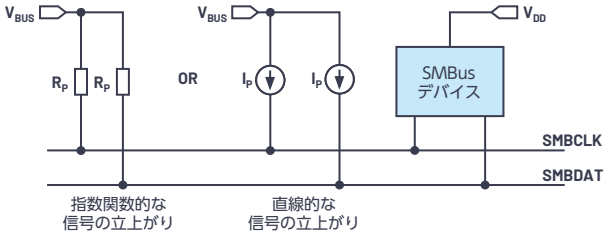

図17. SMBusで使用するプルアップ回路

SMBusのアドレス

SMBusのアドレスは、7ビットのデータで表されます（図18）。慣例として、「0001 110b」といったように、4ビットのデータの後に3ビットのデータを記述し、その後にbという文字を付加して表記されます。アドレスは、バス上の8ビットのフィールドの上位7ビットを占めます。そのフィールドの下位のビットは、SMBusのアドレスの一部ではありません。別の意味を持つ情報が記述されます。

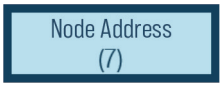

図18. ノードのアドレス

7ビットのアドレスは、メインからバス上の1つ以上のデバイス（ゼネラル・コール・アドレスを備えているものなど）に向けて送信されます。

なお、スタート・コンディションとストップ・コンディションはビットではなく遷移によって表現／判別されます。そのため、シンボルの上にビット数は示されません。トランザクションのダイアグラムを示す場合、リピート・スタートもビットではなく遷移で表現／判別されます。そのため、シンボルの上にビット数は示されません。

図19. SMBusのメッセージ

SMBusのタイミング

SMBusの各種タイミングについては、表6、図20のような形で示されます。

表6. SMBusのタイミング関連のパラメータ

シンボル	パラメータ	単位
f_{SMB}	SMBusの動作周波数	kHz
t_{BUF}	ストップ・コンディションとスタート・コンディションの間でバスがフリーになる時間	マイクロ秒
T_{HD-STA}	（リピート）スタート・コンディション後のホールド時間	マイクロ秒
T_{SU-STA}	リピート・スタート・コンディションのセットアップ時間	マイクロ秒
$t_{SU(STO)}$	ストップ・コンディションのセットアップ時間	マイクロ秒
$t_{HD(DAT)}$	データのホールド時間	ナノ秒
$t_{SU(DAT)}$	データのセットアップ時間	ナノ秒
$t_{TIMEOUT}$	クロックのタイムアウト（ローが継続）の検出	ミリ秒
t_{LOW}	クロックのローの期間	マイクロ秒
t_{HIGH}	クロックのハイの期間	マイクロ秒

パワー・マネージメント向けに定義されたPMBus

SMBusと同様に、PMBusはI^2Cプライマから派生したものです。パワー・マネージメントの用途に向けたオープン・スタンダードのプロトコルとして規定されました。柔軟性と汎用性が高く、アナログ技術とデジタル技術の両方をベースとしたデバイス間の通信が行えます。それにより、真の相互運用性が実現されます。結果として、電源システムの設計の複雑さを軽減し、市場投入までの時間を短縮することが可能になります。

PMBusは、電力の制御用コンポーネントと管理用コンポーネントを備える電源で使用されます。デジタル管理を行えるようにするために、管理の要件に対応可能なコマンドと構造を備えています。I^2Cプライマとの間には、電気的要件とコマンドのセマンティクスの面で互換性があります。つまり、I^2CプライマとPMBusは相互運用が可能だということです。

電力管理においては、過電圧の監視が非常に重要です。PMBusは、これに関するパラメータの設定を行い、結果を取得するためのコマンドを提供しています。また、PMBusには、I^2CプライマとSMBusの機能のうち利用可能なものを踏襲しています。そのため、既存の規格、特にSMBusを基盤とするプロトコル層として機能させることが可能です。

I^2Cプライマの仕様では、2線式バスの物理層、タイミング、フロー制御に関連することだけが定められています。同仕様では、（SMBusのプロトコルとは異なり）メッセージのフォーマットについては定義していません。また、メッセージの内容についても特に規定はありません。

PMBusの仕様には、電力管理を行うためのプロトコル一式が用意されています。あるデバイスから別のデバイスにビット／バイト単位のデータを転送する方法が定義されているだけではありません。それらのデータに意味を与えるためのコマンド／言語についても記載されています。

電源のアドレス

冗長システムに電源を実装する場合、そのアドレス位置を指定するために最多で3つの信号を使用することができます。Address2、Address1、Address0の3つです。非冗長システムの場合、電源のアドレス位置としてはB0hを指定する必要があります（表7）。

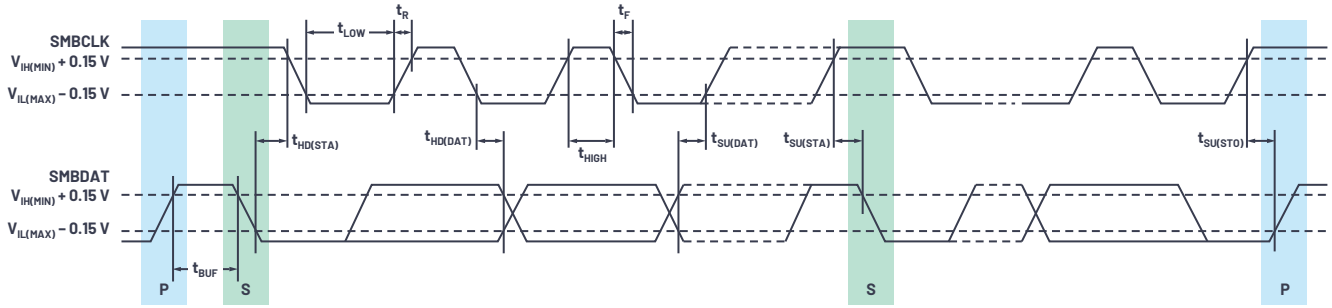

図20. SMBusのタイミング仕様

表7. PMBusのアドレス

使用されるアドレス	アドレスの指定用に2本のピンを備える大半のサーバ電源で使用されるメインのアドレス				電源にアドレスの指定用のピンが3本ある場合の追加アドレス			
システムのアドレス Address2/Address1/Address0	0/0/0	0/0/1	0/1/1	0/1/1	1/0/0	1/0/1	1/1/0	1/1/1
PMBus対応のデバイスの読み出しアドレス	B0h/B1h	B2h/B3h	B4h/B5h	B6h/B7h	B8h/B9h	BAh/BBh	BCh/BDh	BEh/BFh

電源内部のハードウェア

電源内で使われるデバイスについては、I^2CのV_{DD}をベースとする電源とドライブ（V_{DD}が3.3Vの場合）は、ハイ・パワー仕様であるSMBus 2.0に準拠している必要があります。このバスは3.3Vで動作させなければなりません。

電力の調達

電源内部の回路は、スタンバイ出力から電力を得ることになります。また、冗長構成の電源の場合、デバイスはORをとっているデバイスのシステム側から電力を得ます。システム内の電源または冗長構成の並列電源がAC電源から電力を得ている場合、PMBusに対応するデバイスは、常にオンになっていなければなりません。

プルアップの方法

電源内部のSCLラインまたはSDAラインには、ウィーク・プルアップ抵抗だけを接続します。メインのプルアップ抵抗はシステムから提供され、3.3Vまたは5Vに接続されます。システム設計において、メインのプルアップ抵抗は電源の外部に配置し、スタンバイ・レールから電力を得るようにしなければなりません。

データの速度

電源内で使用されているPMBus対応のデバイスは、100kbpsというSMBusの最高速度で動作します。クロック・ストレッチはバスの速度を低下させる可能性があるので、できるだけ使用を避けなければなりません。

3種のプロトコルの仕様

表8をご覧ください。これは、I^2Cプライマ、SMBus（ハイ・パワー、ロー・パワー）、PMBusの仕様についてまとめたものです。信号の伝送方法、タイミング、電気的特性など、各仕様の概要をご確認ください。

I^2Cプライマ、SMBus、PMBusの関係

SMBusは、もともとバッテリ管理システムを容易に実現するために策定されました。I^2Cプライマのハードウェア仕様を踏襲していますが、ソフトウェア仕様については様々な追加が行われています。その結果、システムを再起動することなくデバイスをホット・スワップできるようになっています。PMBusは、SMBusを拡張したものだと言えます。パワー・コンバータに関連するデバイスを管理するための一連のコマンドが定義されており、電圧、電流、温度の測定値といったデバイスの属性を取得することが可能になっています。一般に、I^2Cプライマ、SMBus、PMBusに対応するデバイスは、大きな問題を起こすことなくバスを共有することができます。

表8. I^2Cプライマ、SMBus、PMBusの仕様

仕様		I^2Cプライマ	SMBus		PMBus
			ハイ・パワー	ロー・パワー	
信号の伝送	PEC（オプション）	—	—	—	—
	SMBALERT（オプション）	—	—	—	—
	ブロック・サイズの制限	—	32バイト	32バイト	255バイト
タイミング	データ・レート	100kbps	100kbps	100kbps	100kbps
	・標準モード	400kbps	—	—	400kbps
	・ファスト・モード	1Mbps	—	—	—
	・ファスト・モード・プラス	3.4Mbps	—	—	—
	・ハイ・スピード・モード	0Hz～3.4MHz	10kHz～100kHz	10kHz～100kHz	10kHz～400kHz
	クロック速度		25ミリ秒～35ミリ秒	25ミリ秒～35ミリ秒	25ミリ秒～35ミリ秒
	バスのタイムアウト	—	50マイクロ秒	50マイクロ秒	50マイクロ秒
	バスにおけるメインのリクエストの遅延（最小）	—	—	—	—
	SCLのホールド時間（最大）	—	2ミリ秒	2ミリ秒	2ミリ秒
	データのホールド時間（最小）	—	300ナノ秒	300ナノ秒	300ナノ秒
電気的特性	バスのセグメントあたりの容量性負荷（最大）	400pF	400pF	—	400pF
	立上がり時間（最大）	1マイクロ秒（100kHz）、300ナノ秒（400kHz）	1マイクロ秒	1マイクロ秒	1マイクロ秒（100kHz）、300ナノ秒（400kHz）
	0.4Vにおけるプルアップ電流（最大）	3mA（標準モード、ファスト・モード）	4mA	350µA	4mA
	デバイスあたりのリーク電流（最大）	±10µA	±10µA	±5µA	±10µA
	VIL（ロー・レベルの入力閾値、最大）	0.3×V_{DD}または1.5V	0.8V	0.8V	0.8V
	VIH（ハイ・レベルの入力閾値、最小）	0.7×V_{CC}または3V	2.1V	2.1V	2.1V
	VOL（ロー・レベルの出力閾値、最大）	0.4 V	2.4V	0.4V	0.4V

I²C、SMBus、PMBusの長所

3つのプロトコルの長所としては、以下のような事柄が挙げられます。

- ワイヤは 2 本しか必要ない
- ACK/NACK ビットに対応する
- いずれのプロトコルも、よく知られている
- 複数のメインと複数のノードをサポート
- ハードウェアは UART（Universal Asynchronous Receiver/Transmitter）より簡素
- 広く使用されている

I²C、SMBus、PMBusの短所

3つのプロトコルには、以下のような短所があります。

- SPI よりもデータ転送レートが遅い
- データ・フレームのサイズが 8 ビットに制限されている
- SPI よりも複雑なハードウェアを実装しなければならない

ユース・ケース

最後に、3つのプロトコルの代表的なユース・ケースを列挙しておきます。

- センサーからのデータの読み取り
- センサーに対するデータの書き込み
- EEPROM、温度センサー、タッチ・スクリーン、近接センサー
- ユーザが指定した操作の送信、制御
- 複数のマイクロコントローラとの通信
- 民生用の電子機器
- システム管理
- パワー・マネージメント
- デバッグ

参考資料

「Advantages and Limitations of I²C Communication（I²Cによる通信のメリットと限界）」Total Phase、2016年8月

Sal Afzal「I²C Primer: What Is I²C? (Part 1)（I²Cプライマ：I²Cとは何か？（Part 1））」Analog Devices

Sal Afzal「I²C Timing: Definition and Specification Guide (Part 2)（I²Cのタイミング：定義と仕様に関するガイド（Part 2））」Analog Devices

Scott Campbell「Basics of the I²C Communication Protocol（I²C通信プロトコルの基礎）」Circuit Basics

「I²C Quick Guide（I²Cのクイック・ガイド）」、Analog Devices

「I²C－What's That?（I²Cとは何なのか？）」I²C Bus

著者について

Mary Grace Legaspi（mary.legaspi@analog.com）は、アナログ・デバイセズのファームウェア・エンジニアです。2018年9月に入社（フィリピンのカビテ支社）しました。現在はコンシューマ・ソフトウェア・エンジニアリング・グループの設計／レイアウト・チームに所属しつつ、フィリピン大学で経営学の修士号の取得を目指しています。タルラク州立大学で電子工学の学士号を取得しました。

著者について

Eric Peña（eric.pena@analog.com）は、アナログ・デバイセズのファームウェア・エンジニアリング・マネージャです。2019年4月に入社（フィリピンのカビテ支社）しました。現在は産業用プラットフォーム／ネットワーキング・グループの設計／レイアウト・チームに所属しています。それ以前はTechnology Enabler Designerにファームウェア・エンジニアとして、Fujitsu Ten Solutionsにシステム・エンジニアとして勤務していました。マニラのアダムソン大学でコンピュータ工学の学士号を取得しています。

ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™

アナログ・デバイセズ株式会社

VISIT ANALOG.COM/JP

お住いの地域の本社、販売代理店などの情報は、analog.com/jp/contact をご覧ください。

オンラインサポートコミュニティEngineerZoneでは、アナログ・デバイセズのエキスパートへの質問、FAQの閲覧ができます。

AD5511-0-11/21