[第1章]

各インターフェースの用語や動作原理を理解する

# 1，2，3線シリアル・インターフェースの基本

最初に，2線式のI²C，3線式のSPI，1線式の1-Wireの3種類のシリアル通信インターフェースについて，基本的な概念や動作原理，用語などを説明します．

通信の用語で同期式/非同期式という言葉がよく使われますが，同期式とは同期用のクロック信号がデータ信号と一緒に伝えられるもので，本書ではI²CとSPIが相当します．この通信ではクロック信号を基準にして信号を送受信します．

非同期式とは同期用のクロック信号がないもので，パソコンなどで広く使われているRS-232Cレベルのシリアル・インターフェースはその代表的なものです．本書では1-Wireが相当します．この通信では信号のパルス幅などを基準として信号を送受信します．

## 1-1　2線でつなぐI²C(Inter-Integrated Circuit)

### ● I²Cとは

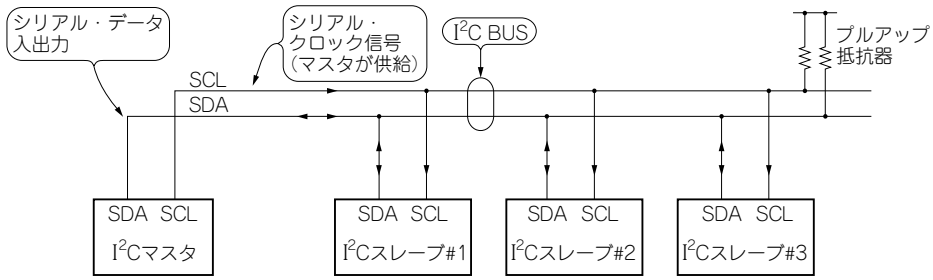
I²Cとは，オランダのフィリップス社が提唱する2線式の同期式シリアル通信インターフェースのことで，シリアル・データ信号SDAとシリアル・クロック信号SCLの2本の信号で通信します．シリアル・クロック信号SCLは，バス・マスタが生成します．シリアル・データ信号SDAは，送受信により入力と出力の向きが切り替わる双方向信号です．**図1-1**のようにマスタ一つに対して複数のスレーブ・デバイスをバス接続できます．また，マルチ・マスタ・モードではマスタを複数もつことも可能です．

スレーブには固有のアドレスが割り付けてあり，通常，マスタはそのアドレスを指定してスレーブ・デバイスを特定し通信します．スレーブ側でジェネラル・コール・アドレスという機能をサポートしていれば，複数のスレーブ・デバイスに一斉にデータを送信することもできます．不特定のスレーブに配信するということで，いわゆるブロードキャスト(放送)方式です．

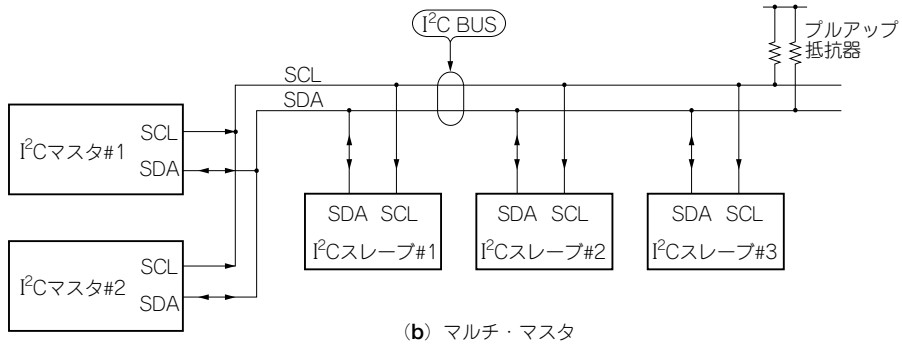
通信レートは標準モードで0～100kbps，ファースト・モードで最大400kbpsとなっていますが，後にハイ・スピード・モードの最大3.4Mbpsという高速な仕様も決められています．

### ● I²C通信の動作

通信の手順は，まずマスタがスタート・コンディションを発行し，コントロール・バイトをバス上のすべてのスレーブへ対して送信します．このコントロール・バイトにはスレーブのアドレスが入っていて，これを受信したスレーブは，そのアドレスが自分のものと一致するかどうかを調べます．




（**a**）シングル・マスタ
（一つのマスタに対して複数のスレーブが接続されている場合）




（**b**）マルチ・マスタ
（複数のマスタに対して複数のスレーブが接続されている場合）

**図1-1　I²Cバスの接続**
I²Cバスに複数のスレーブ・デバイスを接続する場合の構成図.

アドレスが自分宛てでなかったスレーブは，待機状態に戻ります．アドレスが自分宛てであったスレーブは，コントロール・バイトに含まれる送受信(R/W)ビットの値に応じてデータをマスタから受信するか，マスタへ送信します．データは8ビットですが，その後にACKデータが1ビットあります．ACKはマスタ，スレーブにかかわらず，データを受信したほうが出力します．

通信が終われば，マスタはストップ・コンディションを発行して，スレーブに通信の終わりを通知します．スレーブは，この通知を受け取り待機状態に戻ります．

転送されるデータは8ビット単位で，転送は最上位(MSB)から始まります．

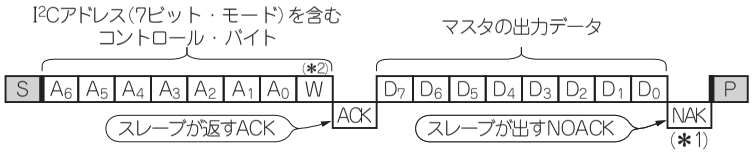
**図1-2**は，通信時の転送ビットを簡易的に示したものです．各ビットが上下2列で並んでいますが，上段はマスタが出力(スレーブが入力)するビットを，下段はマスタが入力(スレーブが出力)するビットを示しています．

**図1-2**(**a**)，(**b**)はI²Cアドレスが7ビットのときのもので，本書で扱うのはすべてこのモードです．**図1-2**(**c**)，(**d**)はI²Cアドレスが10ビットのもので，PICのMSSPはこの10ビット・モードもサポートしていて，半自動的に10ビット・モードを制御することもできます．

### ●スタート・コンディションとストップ・コンディション

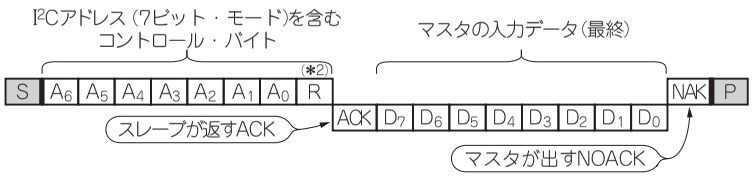
マスタは，通信を始める際にスレーブに対して通信シーケンスの始まりを示すためにスタート・コン

**用語**
S …… スタート・コンディション
RS …… リピート・スタート・コンディション
P …… ストップ・コンディション
NAK …… NOACK（ACK=“H”）

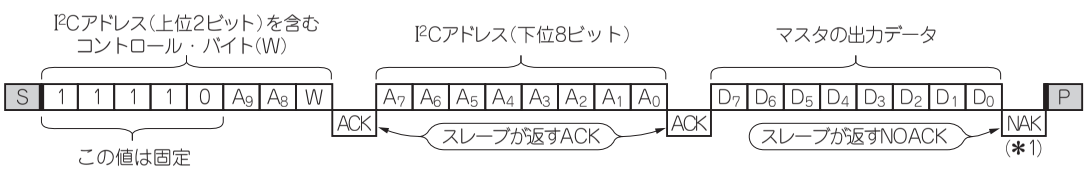



（**a**）7ビット・モード時のマスタのデータ出力（スレーブのデータ入力）

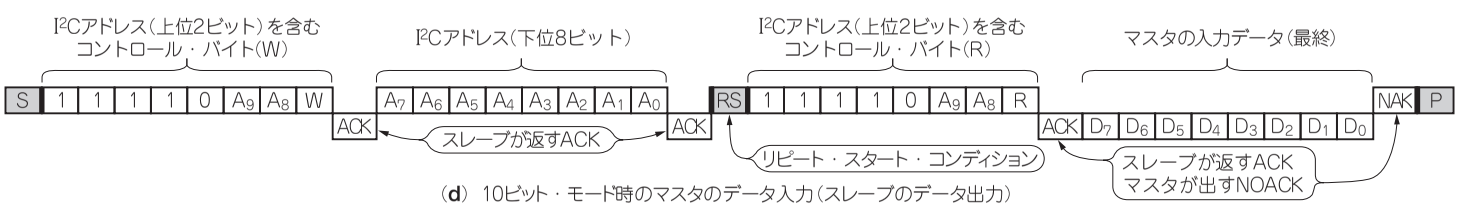
※上の列はバス・マスタの出力，下の列はバス・マスタの入力（スレーブの出力）を表す．
（*1）スレーブによってはACKを返す場合もある．
（*2）‘1’（Rと表記）のときにデータはスレーブからマスタへ，‘0’（Wと表記）のときにデータはマスタからスレーブに送信される．




（**b**）7ビット・モード時のマスタのデータ入力（スレーブのデータ出力）




（**c**）10ビット・モード時のマスタのデータ出力（スレーブのデータ入力）




（**d**）10ビット・モード時のマスタのデータ入力（スレーブのデータ出力）

図1-2　I²Cのデータ・フォーマット
この図は基本的な通信データのフォーマットを簡易的に表したもの．ACK/NOACKはデータを受信したほうが返す．10ビット・アドレス・モードではアドレスは2回に分けて送受信される．マスタが最終データを受信したときは，マスタはスレーブに対してNOACKを送信する．

ディションを発行します．これは，シリアル・クロックSCLが“H”レベル時にシリアル・データSDAを“H”レベルから“L”レベルに変化させることで成立します．

通信の終わりには，ストップ・コンディションを発行して通信シーケンスの終わりをスレーブに知らせます．これはSCLが“H”レベルのときに，スタート・コンディションのときとは逆にSDAを“L”レベルから“H”レベルに変化させることで成立します．

通常のデータ・ビットではSCLが“L”レベルのときにSDAのレベルを変化させるので，データ・ビットとスタート/ストップ・コンディションは区別できます．

EEPROMのアクセスの際などには，スタート・コンディションを発行した後，ストップ・コンディションを発行する前に再びスタート・コンディションを発行することがあります．これをリピート(またはコンティニアス)スタート・コンディションと言います．

これはI²CのEEPROMからデータを読み出す際など，一つの通信シーケンスの中で，送信と受信の向きが切り替わる場合に発生します．**図1-3**にEEPROMから1バイトのデータを読み出す際のバスの波形を示します．通信手順は次のようになります．

(1) スタート・コンディション発行 …… 通信シーケンスの開始
(2) コントロール・バイト(W)送信 …… I²Cアドレスと書き込みの指定
(3) ROMアドレス1送信 …… ROMアドレスの上位8ビットを出力
(4) ROMアドレス2送信 …… ROMアドレスの下位8ビットを出力
(5) リピート・スタート・コンディション発行 …… 通信シーケンスの再開
(6) コントロール・バイト(R)送信 …… I²Cアドレスと読み出しの指定
(7) ROMデータ受信 …… ROMデータを入力
(8) ストップ・コンディション発行 …… 通信シーケンスの終了

I²CアドレスとはI²Cのスレーブ・デバイスのアドレスのことで，ROMアドレス(EEPROM上のアドレス)とは区別してください．このような手順は，接続されるスレーブ・デバイスにより変わります．

### ● I²Cアドレス

スレーブ・デバイスの指定方法には，7ビット・モードと10ビット・モードの2種類があります．**図1-2**に，それぞれの通信フォーマットを簡略化した図を示しています．10ビット・モードの場合は1回のコントロール・バイトではアドレスを送信できないので，2回に分けて送信します．

マスタが10ビット・モードで送信する場合は，図(**c**)のようなフォーマットになります．第1アドレスが含まれるコントロール・バイトは上位5ビットが“11110”の固定値で，その後にスレーブ・アドレスの上位2ビットの$A_9$，$A_8$，最下位は送受信の向きを示すR/Wビットというような構成になっています．それに続けて第2アドレス($A_7$～$A_0$)の8ビットが続きます．この後に，データ・バイトが出力されます．これ以降は7ビット・モードのときと同じです．

マスタが10ビット・モードで受信する場合は，図(**d**)のように少し複雑なフォーマットになっています．第2アドレスを送信した後，リピート・スタート・コンディションを発行し，その後，第1アドレスを再び送信し，その次からデータを受信します．これ以降は7ビット・モードのときと同じです．このようにリピート・スタート・コンディションが入っているのは，I²Cの基本的なフォーマットに互換性をもたせたまま，第2アドレスのバイトを拡張したためだと思われます．リピート・スタート・コンディション以降は，図(**b**)と同じ形をしていることがわかります．

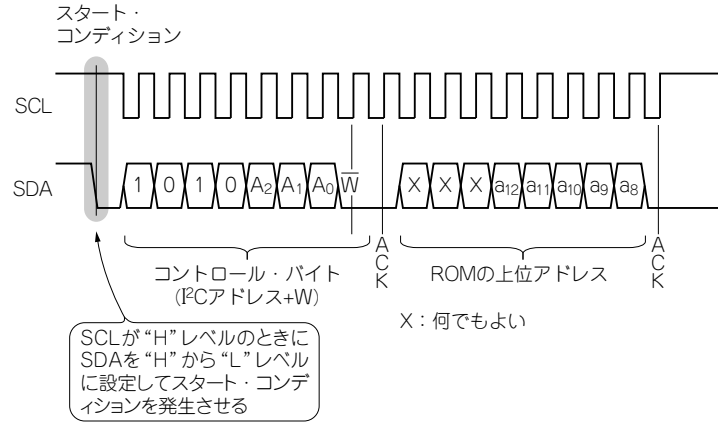



**図1-3**
**リピート・スタート・コンディションの例**
I²C EEPROMの24LC64で発生するリピート・スタート・コンディションの使用例を示したチャート．ROMアドレスを書き込んだ後，ROMデータを読み出す際にライトからリードへシーケンスを切り替える必要がある．この場合にスタート・コンディションを再発行する．なお，I²Cのデバイス・アドレスとROMのアドレスを混同しないように注意．

### ● クロック・ストレッチ

スレーブはマスタからデータを受信した後(またはマスタへデータを送信した後)に，内部処理に時間がかかるなどの理由でマスタに対してWAITをかけることができます．スレーブが強制的にクロック信号SCLを“L”レベルにすると，マスタはクロックが出せなくなるので，通信シーケンスを一時停止します．スレーブは，準備が整うとSCLの“L”レベル出力を解除します．これでマスタは通信シーケンスを再開します．

このように，マスタを待たせるためにスレーブがSCLを強制的に“L”レベルにすることをクロック・ストレッチと言います．

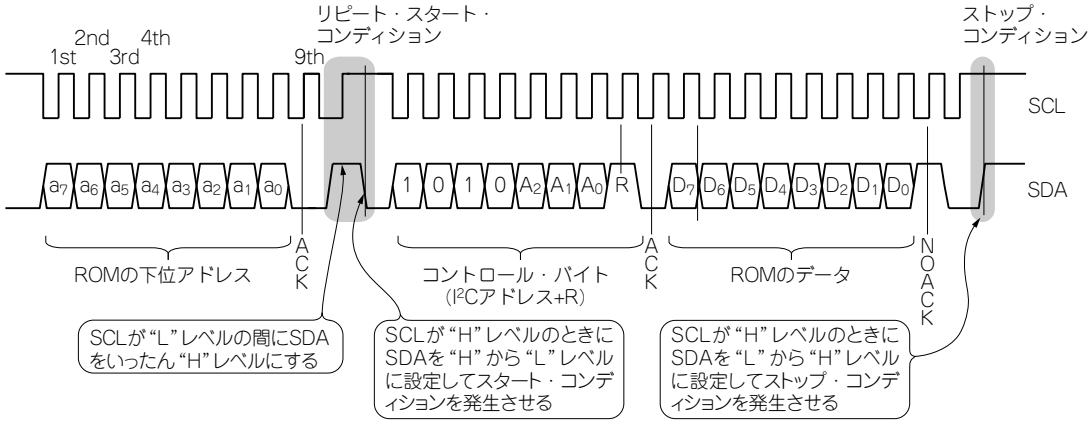
### ● ACKとNOACK

I²Cのデータ送受信の単位は8ビットですが，9ビット目にACKビットがあります．これは通信の正否を示すほかに，何らかの意味をもたせてある場合もあります．ACKビットはマスタ/スレーブにかかわらず，必ず受信したほうが出力します．たとえば，マスタが8ビットのデータをスレーブに出力し終わると，スレーブがACKを返します．逆にマスタが8ビットのデータを入力し終わると，マスタがスレーブにACKを返します．

I²Cの場合，ACKビットが“L”レベルのときが“ACK”，“H”レベルのときが“NOACK”と表現します．マスタは，通常はデータの受信後に“ACK”をスレーブへ返しますが，最終データの受信後は“NOACK”を返します．この“NOACK”により，スレーブはデータの転送が終わったと判断し，送信シーケンスをリセットします．

24LC64(I²CのEEPROM)などでは，マスタがシーケンシャル・リード(データの連続読み出し)の際に最終データを“NOACK”にしてEEPROMへシーケンシャル・リードの終了を通知するようなケースもあります．

スレーブは，最終データを受信したときでも“NOACK”を返さない場合があります．マスタ側の対応にもよりますが，スレーブ受信時の“NOACK”出力は必須ではありません．24LC64ではバイト・ライト(1バイト書き込み)時やページ・ライト時の最終バイトの送信時でも“ACK”を返してきます．そもそも，固定長のデータなら最終バイトかどうか判断できますが，可変長のデータをスレーブへ書き込む場




合は，スレーブ側は最終バイトかどうかを知る術がないわけですから“NOACK”の返しようがありません．本書でも，PICのスレーブ受信時の応答は“NOACK”ではなく“ACK”にしています．もし，マスタ側が最終バイトの応答に“NOACK”を期待している場合は，マスタかスレーブのどちらかのプログラムを修正する必要があります．

### ● マルチ・マスタ・モード

I²Cでは，同一のI²Cバス上に複数のマスタ・デバイスを置くことができます[**図1-1**(**b**)参照]．マスタはクロックを供給することと，自らの都合でデータを入出力することのため，複数のマスタが同時に動作するとクロックとデータ信号の衝突が起こります．この状態をバス・コリジョン(衝突)と言います．バス・コリジョンが発生すると，当然ながら正常な通信はできません．マルチ・マスタ・モードをサポートしているマイコンにはバス・コリジョンを検出する機能があり，衝突が起こると割り込みが発生するなどしてそのことがわかるようになっています．

実際は，マスタBが通信しようとしたときにすでにほかのマスタAが通信していると，そのことがマスタBで検出できるようになっているため，マスタBの通信シーケンスが失敗するようになっています．このような状態を調停(アービトレーション)負けといいます．

バス・コリジョン(調停負け)が発生したマスタBは，バスが解放されるのを待ってから通信をやり直さなければなりません．

PICの場合は，自分が出しているSDA信号のレベルと実際のバスのSDA信号の状態が異なることを検出すると，バス・コリジョン状態になります．

ところで，マスタはI²Cアドレスをもたないことからもわかるように，マスタ同士で通信することはできません．

マルチ・マスタの用途として一般的と思われるのは，ROMやA-Dコンバータなどのスレーブ・デバイスを複数のマスタで共有することです．RAMなどを共有メモリとして使えば，工夫しだいでマスタ間で通信させることもできるでしょう．

## 1-2　3線でつなぐSPI(Serial Peripheral Interface)

### ● SPIとは

SPIとは，米国モトローラ(現フリースケール)社が提唱する3線式の同期式シリアル通信インターフェースのことで，データ出力信号SDO，データ入力信号SDI，クロック信号SCKの3本の信号で通信します．

SPIマスタに複数のスレーブを接続する場合は，**図1-4**のような接続形態になります．SPI通信の最大の特徴は，送信と受信が同時に起こることです．マイコンとペリフェラル・デバイスなどの通信に使われます．ペリフェラル・デバイスにはA-Dコンバータ，D-Aコンバータ，EEPROMなどがあります．

SPIは通信レートやタイミングの厳格な規定はありません．

### ● SPI通信の動作

SPI通信は一言でいうと，マスタとスレーブのデータの内容を入れ替えるような動作をします．

マスタが生成するシフト・クロック(シリアル・クロック)は，マスタ内のシフト・レジスタをシフトさせると同時に，スレーブのシフト・レジスタもシフトさせます(**図1-5**参照)．

マスタの最上位ビット(MSB)のデータは，シフトの結果，スレーブの最下位ビット(LSB)に挿入されます．同時に，スレーブのMSBのデータもマスタのLSBに挿入されます．この動作を8回繰り返すと，マスタのシフト・レジスタとスレーブのシフト・レジスタの内容が入れ替わります．これがSPI通信の動作原理です．

1バイトのデータ転送は，最上位(MSB)から始まり最下位(LSB)で終わります．

### ● SS(スレーブ・セレクト)信号

SPI通信は，基本的には一つのマスタに対して一つのスレーブが接続されるという1：1の接続形態です

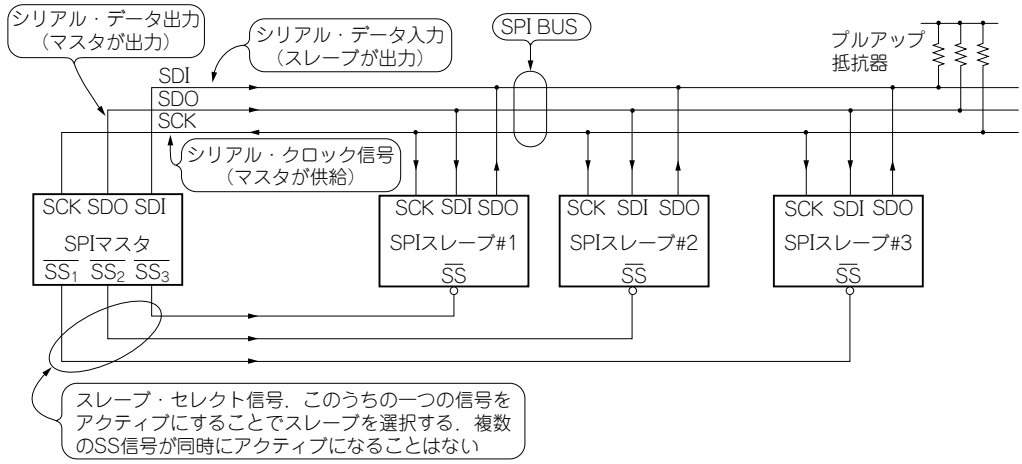
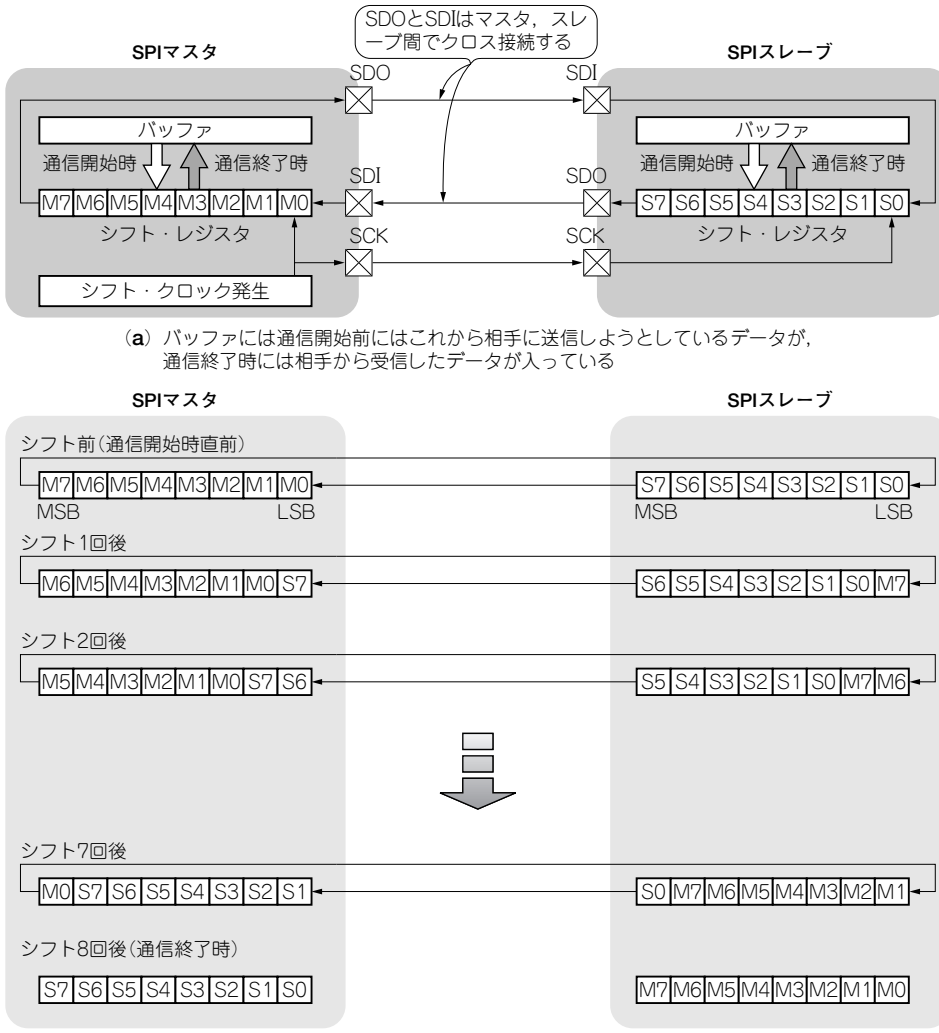



**図1-4　SPIバス接続**

SPIバスに複数のスレーブ・デバイスを接続する場合の構成図．マスタはSS信号により一つのスレーブを選択し，そのスレーブに対してリード/ライトする．

が，**図1-4**のように複数のスレーブをバス接続してSS(スレーブ・セレクト)信号でその中の一つのデバイスを選択し，個別に通信させることもできます．マスタは通信が必要なスレーブのSS信号をアクティブにすることでスレーブを特定し，そのスレーブとだけ通信します．

このSS信号は，通信シーケンスのリセットに使われることもあります．たとえば，8ビット未満の通信の最中にSS信号を非アクティブにすると，通信はリセットされ，次回SS信号をアクティブにしたときは，MSBから通信が始まるようになります．通信するビット数が8の倍数でない場合に，むだなビットを送信しないために通信を途中で打ち切るというような用途もあります．




(**a**) バッファには通信開始前にはこれから相手に送信しようとしているデータが，通信終了時には相手から受信したデータが入っている

(**b**) 8回シフトすると，マスタとスレーブの内容が入れ替わる

**図1-5　SPI通信の動作原理**

SPI通信を行う場合のデータ・ビットの動きを説明した図．マスタとスレーブを一つの16ビット・シフト・レジスタと考えると，8回シフト(ローテート)したときに上位8ビットと下位8ビットが入れ替わる．これをマスタとスレーブに置き換えれば理解しやすい．

## 1-3　1線でつなぐ1-Wire(Dallas One Wire)

### ● 1-Wireとは

1-Wireとは，米国ダラス・セミコンダクタ社(マキシム社に買収され，現在マキシムの子会社)が提唱する1線式の非同期式シリアル・インターフェースのことです．送信と受信を1本の線で交互に行います．また，信号線が1本で済むにもかかわらず，複数のスレーブ・デバイスをバス接続することができます．接続は**図1-6**のように非常にシンプルなものですが，その分制御が複雑になっています．

通信レートはスタンダード・モードで最大16kbps，オーバ・ドライブ・モードで最大142kbpsと規定されています．

発表当初，1-Wireバスの伝達距離は数mと，近距離のペリフェラル間の通信が対象とされてきましたが，近年ではアクティブ・プルアップやツイスト・ペア銅線を採用するなどで伝達距離を延ばし，数百m以上に及ぶ長距離の1-Wireネットワークも構築されています．

ペリフェラル・デバイスにはEEPROM，RAM，温度センサ，A-Dコンバータ，D-Aコンバータ，バッテリなどの電圧監視コントローラなど多種多様なものが用意されています．

1-Wireインターフェースを搭載したデバイスに“i-Button”というものがあります．これは，ステンレス製のCanパッケージ(ボタン電池のような形状)のデバイスで，小型，低コストの温度センサや認証システムなどに利用されています．

### ● 1-Wireの動作

まず，マスタがリセット・パルスを出力すると，バス上にあるスレーブ・デバイスがプレゼンス・パルスを返してきます[**図1-7**(**a**)]．複数のデバイスがある場合は，マスタは検索コマンドを出力して接続されているデバイスの数とそのROMコード(識別番号)を調べます．デバイスが一つしかないときやあらかじめROMコードがわかっている場合は，この検索は省略できます．

デバイスにアクセスする際は，このROMコードでデバイスを特定します．あとは，デバイスごとに決められたコマンドを決められた手順で送受信して目的の動作を実行させます．通信はスロットという単位で制御されます．1ビットのデータは1スロットで表します(詳細は後述)．

### ● ROMコード(識別番号)

ROMコードとは，スレーブ・デバイス同士で決して重複することのない64ビット長の固有の識別番号です．製造時にレーザでデバイス一つ一つにユニークなコード(同じ種類，型番のデバイスでも違うコー

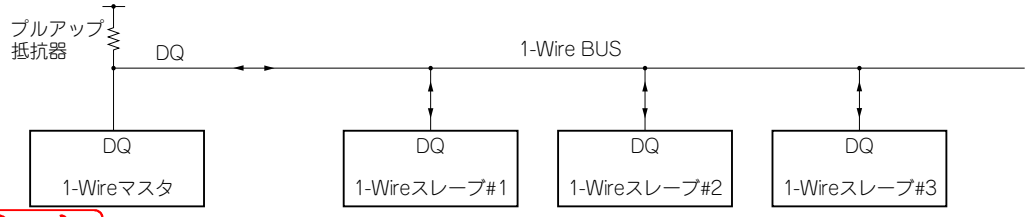
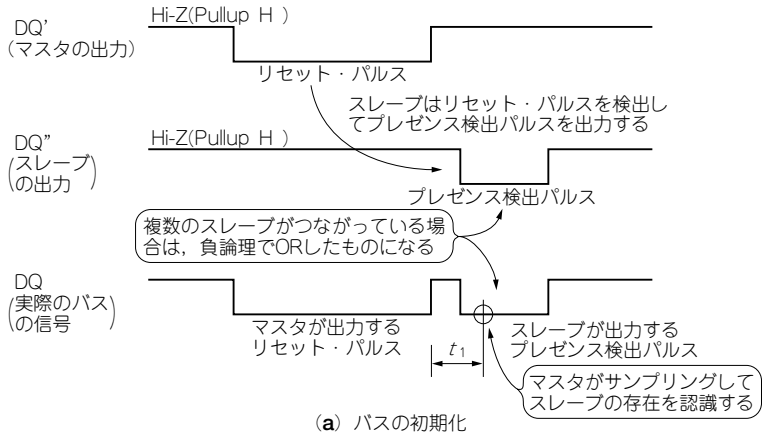
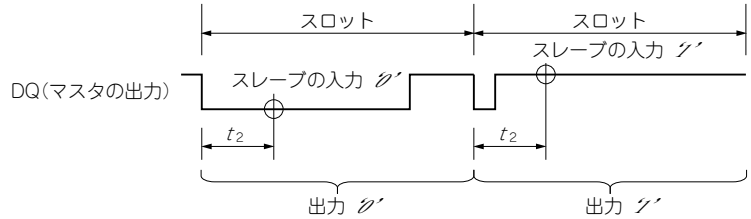



**図1-7　1-Wireバス接続**
1-Wireバスに複数のスレーブ・デバイスを接続する場合の構成図．



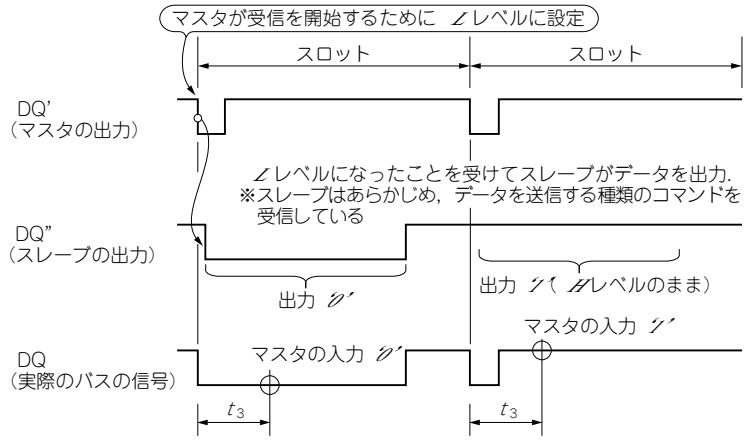

(**a**) バスの初期化

(マスタがリセット・パルスを出力すると，接続されている全スレーブがプレゼンス検出パルスを出力する．マスタはこの出力を読み取ってスレーブが存在することを認識する)




(**b**) マスタのデータ出力（スレーブのデータ入力）

(マスタはスロット単位でビット・データに応じた長さの "1" レベル・パルスを出力する．スレーブはスロットの始まりから$t_2$後の状態を読み取ってビット値を判断する)




(**c**) マスタのデータ入力（スレーブのデータ出力）

(マスタが "1" レベルを出力することで一つのリード・スロットが始まる．スレーブはスロットの始まりを検出してビット・データが "0" の場合は "1" レベルのパルスを出力する．マスタはスロットの始まりから$t_3$後の状態を読み込んで，スレーブが出力したビット値を判断する．なお，スレーブがデータを出力するのは，データを出力する種類のコマンドをマスタから受けているときに限られる．)



**図1-7　1-Wireの信号波形**

この図は1-Wire通信の出力波形を示したもの．通信はバスの初期化(マスタがリセット・パルスを発生させる)から始まる．

ドになる)が書き込まれているため,「64ビット・レーザ書き込みROM」とも言われます.

1-Wireのすべてのデバイスは,第13章の**図13-5**に示すような固有のROMコードを内蔵しています.この内容は,48ビットのシリアル番号(デバイス固有)とファミリ・コード(デバイスの種類により決まっている),それに8ビットのCRC値(ROMコードの内容が正しいかどうかを検証するもの)です.複数のデバイスが1-Wireバスに接続されている場合,マスタはこのROMコードを指定してデバイスを特定しなければなりません.

### ● 1-Wireバスの状態

1-Wireのバスには,“L”レベルとハイ・インピーダンス(以下Hi-Z)の二つの状態がありますが,バスは抵抗器により電源にプルアップされているため,Hi-Zのときは“H”レベルになります.わざわざHi-Zで“H”レベルにするのは,マスタと複数のスレーブをワイヤードOR接続にするためです.

本来,このような接続にはオープン・ドレイン出力のバッファを使わなければなりませんが,入出力ポートを入力に切り替えてハイ・インピーダンス状態にすることで代替するというううまい方法です.バス上のいずれかのデバイスが“L”レベルを出力するとバスは“L”レベルになりますが,複数のデバイスが同時に“L”レベルを出力しても,それを負論理のOR(論理和)条件で検出できます.

PICやH8などの汎用入出力ポートを1-Wireのバス駆動に使用する場合も,上記の原理を適用できます.ポートを出力に設定して“L”レベルを出力すればバスは“L”レベルになり,ポートを入力に設定すればバスはHi-Z状態となってプルアップ抵抗器で“H”レベルになります.このとき,必要ならバスの状態を読み込むこともできるわけです.

> 1-Wireのマニュアルなどにはバス接続をワイヤードANDとして記述してありますが,リセット・パルスは負論理(“L”レベルのときアクティブ)の信号なので,負論理のワイヤードORと解釈したほうがよいでしょう.他方,データ・ビットの場合は正論理と考えてワイヤードANDと考えたほうがよいと思います.ド・モルガンの定理により,負論理入力のORは正論理入力のAND(正確にはNAND)になります.

### ● リセット・パルスとプレゼンス・パルス

通信を始めるにあたって,バス上にスレーブ・デバイスが存在するかどうかを調べるために,マスタは一定時間の間バスを“L”レベルにします(リセット・パルス).これをバス・リセットといいます.スレーブ・デバイスは,バスが一定時間“L”レベルになっているのを検知して,自らもバスに“L”レベルを出力します(プレゼンス・パルス).

マスタは,規定時間後に“L”レベルの出力を停止した後にバスの状態を調べます.このときスレーブが“L”レベルを出力しているため,マスタはそれを検出してスレーブが存在していることを認識します.

**図1-7(a)**は,そのときの動作のようすをマスタとスレーブに個別に分けて描いたものですが,DQ'とDQ"は同じバス上でつながっているため,実際の波形はDQのようになります.

### ● マスタのデータ送信(スレーブのデータ受信)



データの送信は,スロットという単位で考えます.1ビットのデータは1スロットで伝達されるため,1バイト送信するのに8スロット必要です.1スロットの始まりはバスがHi-Z状態(プルアップされている

ため“H”レベル)から“L”レベルになる立ち下がりエッジです．この“L”レベルの幅で‘0’と‘1’を表します．簡単にいうと，“L”レベルの幅が長ければ‘0’，短ければ‘1’ということになります[**図1-7(b)**]．

スレーブ・デバイスは，スロットの始まりから規定時間後のバスの状態を読み込んで，この状態が“L”レベルのときが‘0’，“H”レベルのときが‘1’と判断します．データ・ビットの転送は最下位ビット(LSB)から始まり，最後が最上位ビット(MSB)になります．

### ● マスタのデータ受信(スレーブのデータ送信)

データの受信も，同様に1ビットが1スロットになります．やはり，スロットの始まりは“L”レベルの立ち下がりエッジです．

受信の際は，まずマスタがスロットの始まりとして，バスを“L”レベルにします．スレーブは，この立ち下がりを検出してデータ・ビットが‘0’の場合は“L”レベルの信号を一定時間出力します．データ・ビットが‘1’の場合は何も出力しません．マスタは，規定時間後に“L”レベルの出力をやめ，規定時間後にバスの状態を読み込みます．この状態が“L”レベルのときが‘0’，“H”レベルのときが‘1’となります[**図1-7(c)**]．データ・ビットの転送順序は送信時と同じくLSBから始まります．

なお，スレーブ・デバイスがデータを送信するのは，あらかじめマスタがスレーブにデータを要求するコマンドを出しているときに限ります(勝手にスレーブ・デバイスがデータを送信してくることはない)．

### ● 検索アルゴリズム

この部分が1-Wireデバイスを扱う上で一番厄介なところですが，複数のデバイスをバスにぶら下げる場合は避けることができないので，ここで簡単に説明しておきます．ただし，スレーブ・デバイスが一つだけの場合は検索は省略できます．

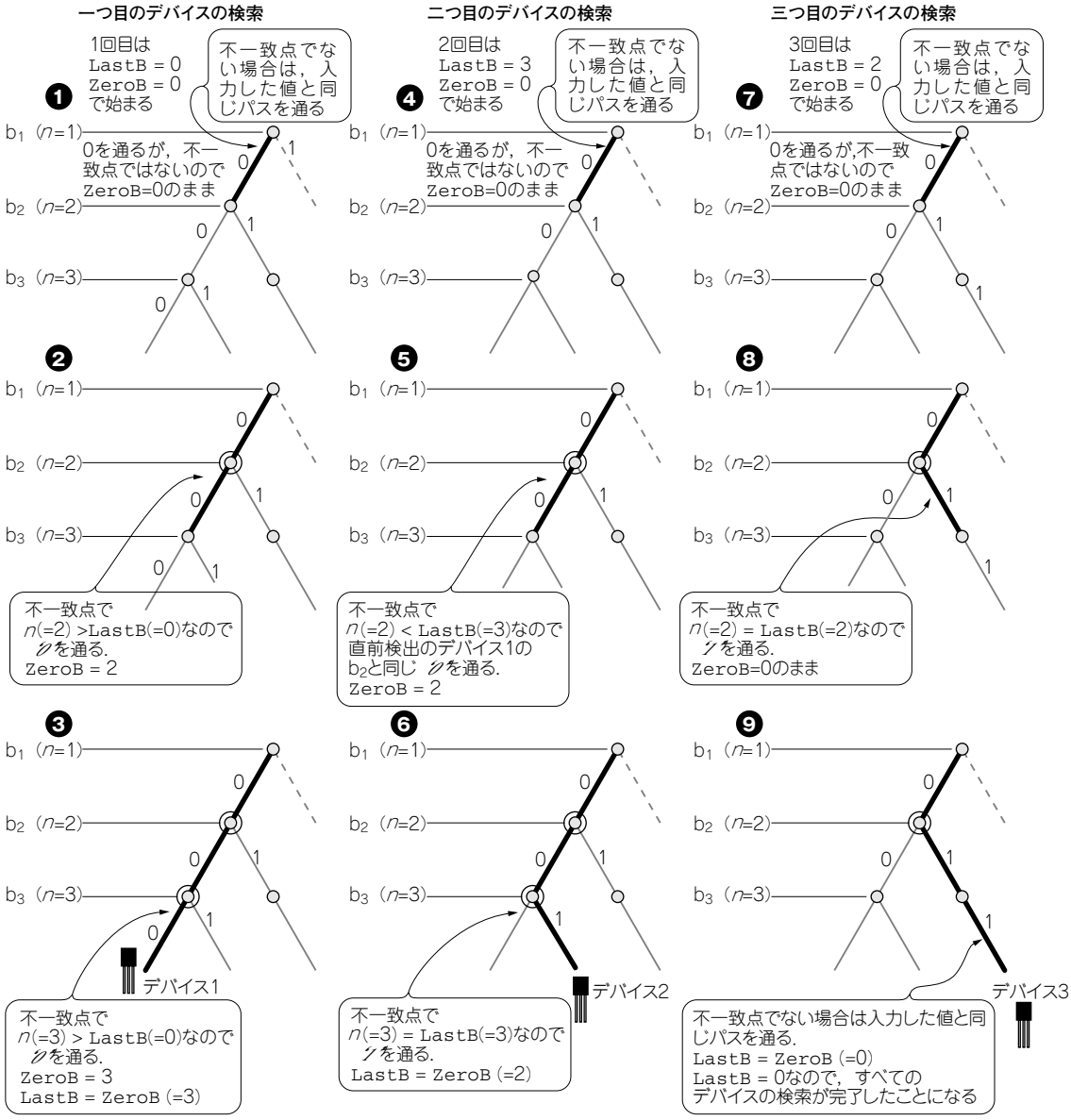
前述のように，1-Wireデバイスは同じタイプのデバイスでも同じROMコード(織別番号)のものは存在しません．そのため，同じ回路の装置を作っても装置ごとにスレーブ・デバイスのROMコードが異なるため，プログラムで固定のROMコードを使用するというわけにはいきません．セットごとにデバイスのROMコードを一つずつ調べて，個別にプログラムを変更すれば対応できないこともありませんが，手間がかかる上メインテナンス性も悪くなります．

そこで，マスタは1-Wireバスに接続されているデバイスのすべてのROMコードを調べるために検索コマンドを実行します．なお，バス上のデバイスが一つだけならROMコードを読み出してそれを直接使用することもできますが，この場合はROMコードを指定しないアクセス方法を使うことができます．

検索では，ツリー状にスレーブのROMコードのビットを調べていくバイナリ・ツリー検索という方法を使用します．**図1-8**は，検索を3ビットに簡略化して説明した検索のアルゴリズムを示すツリー図です．**図1-9**は，その検索の手順を説明したフローチャートです．

スレーブは検索コマンドを受けると，ROMコードの最下位ビットから順番に1ビットずつ，ビット値とそのビットの補数(反転したもの)をマスタへ送信します．この2ビットのデータは複数のスレーブが同時に出力するので，信号はワイヤードANDされてマスタに入力されます．この2ビットを**表1-1**のような規定ルールに基づいてスレーブ(参加デバイス)を探ります．

**表1-1**の(4)は，検索開始時にはあったデバイスが検索途中でとりはずされたか，電源が切られたなど




◎ は不一致点

ZeroB……不一致点で最後に ↙のパスを通ったときのビット番号

LastB……最終的なZeroBの値．次のデバイスの検索時に使用される

不一致点でのパスの判定ルール

| カレント・ビット番号n | 最後の不一致点のビット番号 LastB | 通るパス |
|---|---|---|
| n= LastB | | ↘のパスを通る |
| n< LastB | | 直前に検出したデバイスのROMコードのbnと同じパスを通る |
| n> LastB | | ↙のパスを通る |

**図1-8　1-Wireデバイスのツリー検索アルゴリズム**

ROMコードを3ビットに簡略化してバイナリ・ツリー検索を説明した図．番号順に進行する．この例では三つのデバイスがつながっているものとする．一つのデバイスを検索するたびにリセット・パルス出力→プレゼンス検出パルス検出→検索コマンド出力→ツリー検索のシーケンスを実行する．実際の検索ではCRCも含めた64ビットすべてのROMコードに対してツリー検索を行う．

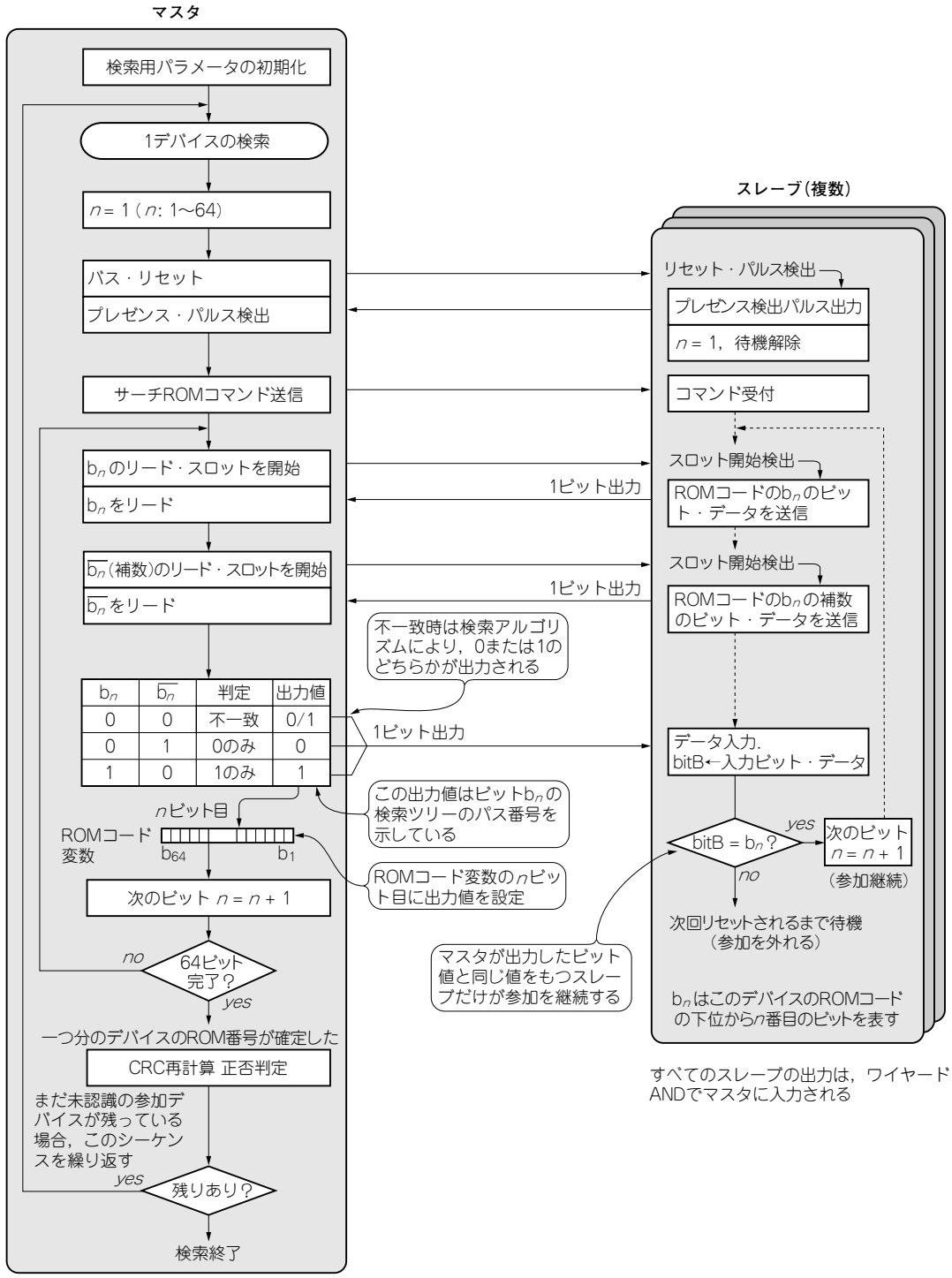



**図1-9　ツリー検索のフローチャート**

1-Wireバス上でマスタがバイナリ・ツリー検索を実行してスレーブ・デバイスを見つけるシーケンスを説明した図．ビット番号$n$は1から始まることに注意．

**表1-1　分岐点での判定**
バイナリ・ツリー検索で分岐点での判定条件の対応表．スレーブの応答により分岐するかどうかを判定する．

| | スレーブの応答 | | 判　定 | マスタの出力 |
|---|---|---|---|---|
| | $b_n$ | $b_n$の補数 | | |
| (1) | 0 | 0 | (a) 参加デバイスには $b_n$ が‘0’のものと‘1’のものがある（不一致） | 0/1 |
| (2) | 0 | 1 | (b) 参加デバイスには $b_n$ が‘0’のものしかない | 0 |
| (3) | 1 | 0 | (c) 参加デバイスには $b_n$ が‘1’のものしかない | 1 |
| (4) | 1 | 1 | (d) デバイスなし（検索の途中で取り外されたなど） | × |

の異常な状態です．

(2)と(3)は，どちらか一方に特定できるケースです．マスタの出力値がそのデバイスのROMコードのビット値ということになります．

(1)はROMコードの現在のビット位置で，ビット値が‘0’のものと‘1’のものの両方のデバイスがバス上につながっている状態です．これを「不一致」と呼びます．本書では「不一致点」と言う場合は不一致が起こっているビットのことを指します．

この不一致点では，ツリーは‘0’のパスと‘1’のパスの両方に分岐します．

パスの分岐に関しては，**図1-9**のフローチャートと**図1-8**のアルゴリズムを参照してください．図では最後に通過した不一致点を“LastB”，不一致点で最後に‘0’のパスを通過したビット位置を“ZeroB”という変数を使います．不一致点での分岐の判定は，**図1-8**の表のようになります．なお，新たなデバイスの検索を始めるごとに，マスタはバスをリセットします．“LastB”は，次回の検索に繰り越して使用します(前回どこで最後に分岐したかを憶えている)．また，これらの図では，処理の都合でビット番号は‘0’～‘63’ではなく‘1’～‘64’となっているので注意してください(最下位ビットを‘1’としている)．

パスを通るときに，マスタはそのビット値をすべてのスレーブに通知します．各スレーブは，自分のROMコードの同じビット位置に同じビット値があれば，検索への参加を継続しますが，その値と異なっている場合は検索対象から外れます．この場合は，次回リセット・パルスが入力されるまで待機状態になります．

このようなツリー検索を64ビットすべてのビットに対して繰り返すと，一つのデバイスが割り出せます．通ったパスを記憶しておけば，その値が検索したデバイスのROMコードになります．64ビットの値が確定したら，CRC値を計算してそのROMコードが有効かどうかを検証します．

64ビット分の検索が終わると，いったんバス・リセットをかけて，再びほかのデバイスの検索を開始します．すべてのデバイスを検索し終わるまで，この処理を繰り返します．

### ● CRC(巡回冗長検査)計算

CRCとは，送受信したデータやROMコードの内容が正しいかどうかを検証するための検査方法です．ビットをシフトして，入力データや特定のビットとXOR(排他的論理和)をとりながらビットを循環させて，結果を出力します．ソフトウェアで実現しやすいように描いたCRCジェネレータの原理図を**図1-10**に示します．

1-Wireデバイスで使用されるCRCは，8ビットです．このCRC値はスレーブ・デバイスのROMコードの上位8ビット，またはスクラッチ・パッド(温度センサDS18S20の場合)の最終バイトなどに保存されて