

DESIGN SHOWCASE

マイクロコントローラ環境における 1-Wire温度センサとの ソフトウェアインタフェースの使用

1-Wire[®]デバイスをマイクロコントローラにインタフェースするにはDS18B20、DS18S20またはDS1822のようないくつかの方法があります。これらの方法には、簡単なソフトウェアソリューションから、ダラスセミコンダクタ社のVHDL 1-WireマスタコントローラをカスタムASICに組み込んだDS2480などの直列インタフェースチップの使用まで広範にわたっています。本文ではマイクロコントローラとDS18x20またはDS1822温度センサを、個数に制限なく、つなぐ際の基本的な1-Wire通信の最も簡単なソフトウェアソリューションを提供しています。

DS18B20、DS18S20及びDS1822に関するタイミング及び動作詳細情報は、マキシム社/ダラス社のウェブサイトwww.maxim-ic.comでご利用いただけるそれぞれのデータシートをご覧ください。

ハードウェアコンフィギュレーション

図1のブロックダイアグラムでは、複数の1-Wire温度センサを使った場合、ハードウェアコンフィギュレーションが簡素化できることを示しています。単線バスはすべてのデバイスに通信アクセスと電源の両方を供給します。バスへの電力は3V~5.5Vの電源レールから4.7kプルアップ抵抗を介して提供されます。それぞれのデバイスがユニークな64ビットROM識別コードをもつので、ほぼ無制限数の1-Wireデバイスがバスに接続できます。

インタフェースタイミング

DS18x20/DS1822とのコミュニケーションは時間スロットの使用によって行われ、1-Wireバスを介してデータ送信ができます。すべての通信サイクルは

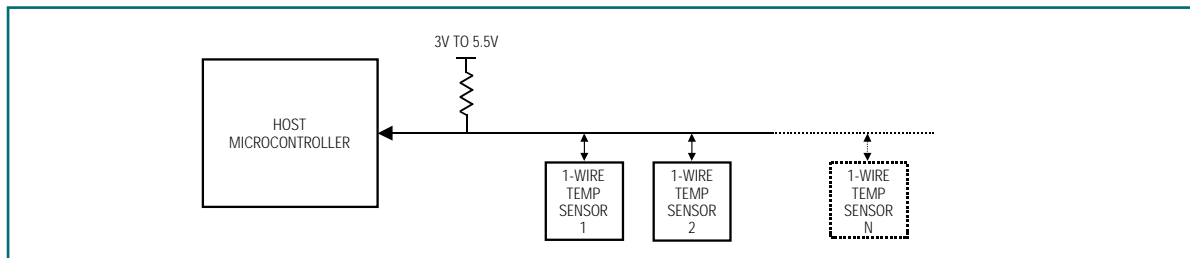


図1. 複数の1-Wire温度センサが単線バスに接続可能。

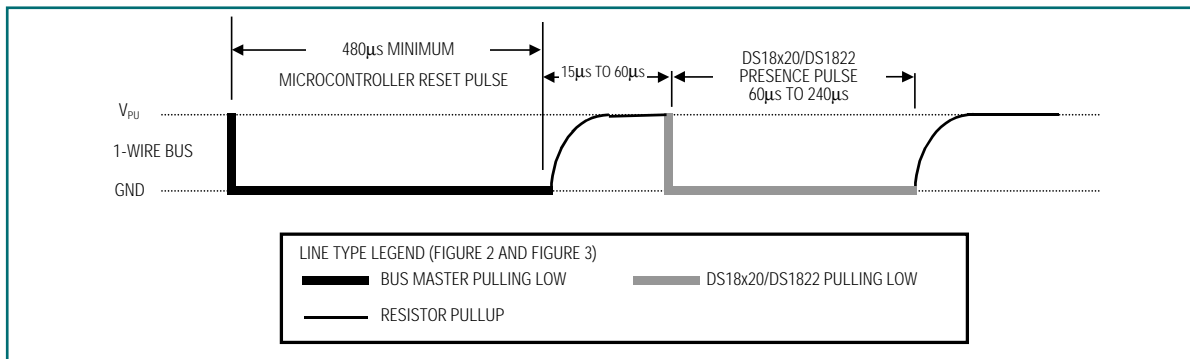


図2. すべての通信サイクルはマイクロコントローラからのリセットパルスで始まり、次にDS18x20/DS1822からの存在パルスが続きます。

1-WireはDallas Semiconductorの登録商標です。

図2に示されているようにマイクロコントローラからのリセットパルスで始まり、DS18x20/DS1822からの存在パルスがその後続きます。

書込み時間スロットはバスマスタが1-Wireバスをロジックハイ(インアクティブ)からロジックローまでプルする時に開始されます。すべての書込み時間スロットは、60 μ sから120 μ sの時間で、サイクル間のリカバリ時間が最小1 μ sでなければなりません。書込み0と書込み1の時間スロットは図3に示されています。書込み0の時間スロット中、ホストマイクロコントローラは、ラインローを時間スロットの長さプルします。しかしながら、書込み1の時間スロット中、マイクロコントローラはラインローにプルし、時間スロット開始後15 μ s以内に開放します。

読み時間スロットは、マイクロコントローラが1 μ sバスをローにプルし、それに続いてDS18x20/DS1822がラインの制御及び有効な(ハイまたはローの)データをだせるように、バスを開放します。全ての読み時間スロットは60 μ sから120 μ sの時間で、サイクル間のリカバリ時間が最小1 μ sでなければなりません(図3)。

ソフトウェア制御

1-Wireインタフェースの特別なタイミング条件を正確にコントロールするためには、ある重要機能がまず確立されなければなりません。最初に確立される機能はディレー機能で、すべての読み及び書込みコントロールに不可欠なものです。この機能は完全にマイクロコントローラのスPEEDに依存しています。このデザインショーケースでは、11.059MHz動作のDS5000(8051コンパチブル)マイクロコントローラが使われています。図4の例は、タイミングディレーを生成するプロトタイプCの機能が示されています。

各通信サイクルがマイクロコントローラからのリセットで開始しなければなりませんから、リセット機能は次に最も重要な実施されるべき機能です。リセット時間スロットは480 μ sです。ディレーを3に設定し次に25に設定することによって(図5)、リセットパルスが必要な時間継続します。リセット後、マイクロコントローラは、DS18x20/DS1822がラインをローにプルすることによって、その存在を示すことができるように、バスを開放しなければ

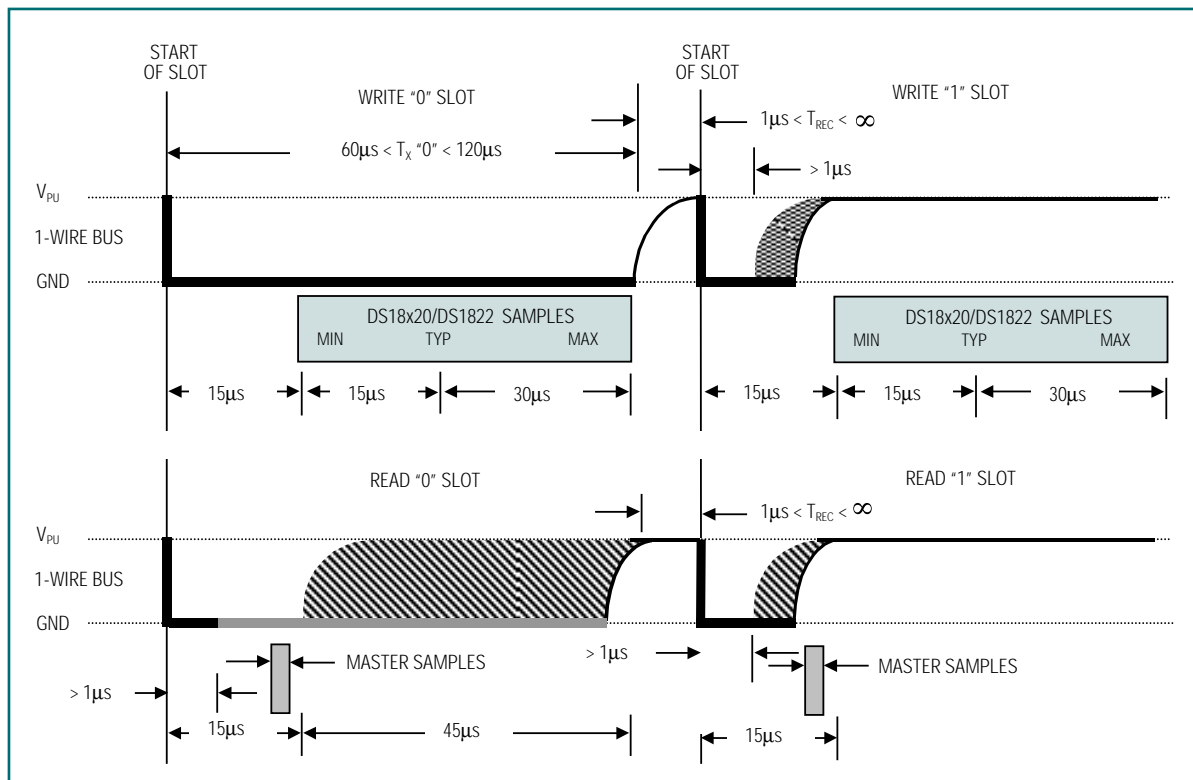


図3. タイミングディレーを生成するプロトタイプCの機能

なりません。複数の温度センサがバス上にあるなら、それらは存在パルスですべて同時に応答します。

図6、7、8、及び9に示されている読み込み及び書き込み機能の例は、すべてのデータビットとデータバイトの読み込み及び書き込み動作に必要な基本構造を示しています。

```
// DELAY - with an 11.059MHz crystal
// Calling the routine takes about 24µs, and then
// each count takes another 16µs
//
void delay (int µs)
{
    int s;
    for (s = 0; s < µs; s++);
}
```

図4. デレイの例

```
unsigned char ow_reset(void)
{
    unsigned char presence;

    DQ = 0;           //pull DQ line low
    delay(29);        // leave it low for 480µs
    DQ = 1;           // allow line to return high
    delay(3);         // wait for presence
    presence = DQ;    // get presence signal
    delay(25);        // wait for end of timeslot
    return(presence); // presence signal returned
}                    // presence = 0, no part = 1
```

図5. リセットの例

```
unsigned char read_bit(void)
{
    unsigned char i;

    DQ = 0; // pull DQ low to start timeslot
    DQ = 1; // then return high
    for (i = 0; i < 3; i++); // delay 15µs from
    // start of timeslot
    return(DQ); // return value of DQ line
}
```

図6. 読み込みビットの例

```
void write_bit(char bitval)
{
    DQ = 0; // pull DQ low to start timeslot
    if(bitval==1) DQ=1; // return DQ high if write 1
    delay(5); // hold value for remainder of timeslot
    DQ = 1;

} // Delay provides 16µs per loop, plus 24µs
// Therefore, delay(5) = 104µs
```

図7. 書き込みビットの例

```
unsigned char read_byte(void)
{
    unsigned char i;
    unsigned char value = 0;

    for (i = 0; i < 8; i++)
    {
        if(read_bit()) value|= 0 x 01<<i;
        // reads byte in, one byte at a time and then
        // shifts it left
        delay(6); // wait for rest of timeslot
    }
    return(value);
}
```

図8. 読み込みバイトの例

```
void write_byte(char val)
{
    unsigned char i;
    unsigned char temp;

    for (i = 0; i < 8; i++) // writes byte, one bit at a time
    {
        temp = val>>i; // shifts val right 'i' spaces
        temp &= 0x01; // copy that bit to temp
        write_bit(temp); // write bit in temp into
    }
    delay(5)
}
```

図9. 書き込みバイトの例