

序説

Dallas 1-Wire[®]は装置との通信にGNDに加えて1線だけを必要とすることに於いて独特です。電源と通信が1つの接続を通して取り扱われます。Dallas 1-Wire装置との通信には汎用ピンが必要なだけです。この応用記述はソフトウェアのみ、またはU(S)ART単位部を利用してのどちらでもAtmel[®] AVR[®]で1-Wire主装置がどう実装できるかを示します。

要点

- 標準速Dallas 1-Wire[®]規約を支援
- 全AVR適合
- ポーリングまたは割り込み駆動の実装
- ポーリング実装は外部ハードウェア不要

目次

| | |
|----------------------------|----|
| 序説 | 1 |
| 要点 | 1 |
| 1. 動作の理屈 – Dallas 1-Wire規約 | 3 |
| 1.1. 基本バス信号 | 3 |
| 1.2. ROM機能命令 | 5 |
| 1.3. メモリ/機能命令 | 5 |
| 1.4. 一斉に適用 | 6 |
| 1.5. 巡回冗長検査 | 6 |
| 2. 実装 | 6 |
| 2.1. ホーリングドライバ | 7 |
| 2.1.1. ソフトウェアのみの実装 | 7 |
| 2.1.2. UARTホーリング実装 | 8 |
| 2.1.3. 上位関数 | 8 |
| 2.1.4. タイミングの考慮 | 10 |
| 2.1.5. 割り込み駆動UART実装 | 10 |
| 2.2. CRC計算 | 12 |
| 2.3. コード例 | 12 |
| 2.3.1. ホーリング例 | 12 |
| 2.3.2. 割り込み駆動例 | 12 |
| 3. 始める前に | 12 |
| 3.1. ソースコード | 12 |
| 3.1.1. ホーリングドライバ | 13 |
| 3.1.2. 割り込み駆動ドライバ | 13 |
| 4. 参考文献 | 13 |
| 5. 改訂履歴 | 14 |

1. 動作の理屈 – Dallas 1-Wire規約

1-Wireバスは信号と電力に関して1線だけを使います。通信は同期、半二重で、主従機構に厳密に従います。1つまたは多数の従装置をバスへ同時に接続することができます。1つの主装置だけがバスに接続されるべきです。

バスはアイドルがHighで、このためにプルアップ抵抗が存在しなければなりません。プルアップ抵抗の値を決めるには従装置のデータシートをご覧ください。バスに接続される全ての装置はバスをLowに駆動できなければなりません。装置がHi-Zにして置けないピンを通して接続される場合、オープンコレクタまたはオープンドレインの緩衝器が必要とされます。

1-Wireバス上の合図は60 μ sの時間幅(スロット)に分けられます。1つのデータビットはこの時間幅毎にバスへ送信されます。従装置は名目上の時間基準と違う意味の時間基準をもつことを許されます。けれども、これは異なる時間基準を持つ従装置との正しい通信を保証するために、非常に正確な主装置のタイミングが必要です。従って以降の項で記述される時間制限に従うことが非常に重要です。

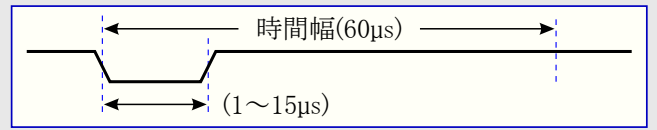
1.1. 基本バス信号

主装置はバス上に於いて完全にビット単位で毎回の通信を始めます。これは方向に拘らず、送信されるべきビット毎に主装置がビット送信を始めなければならないことを意味します。これは常にバスをLowに引っ張ることによって行われ、そしてこれは全ての単位部のタイミング論理回路を同期化します。1-Wireバス上の通信には、“1書き込み”、“0書き込み”、“読み込み”、“リセット”、“存在”の5つの基本命令があります。

1書き込み信号

“1書き込み”信号は右図で示されます。主装置は1~15 μ s間、バスをLowに引っ張ります。そして時間幅の残りの間、開放します。

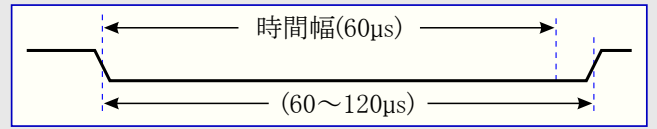
図1-1. “1書き込み”信号



0書き込み信号

“0書き込み”信号は右図で示されます。主装置は最低60 μ s、最大120 μ sの間、バスをLowに引っ張ります。

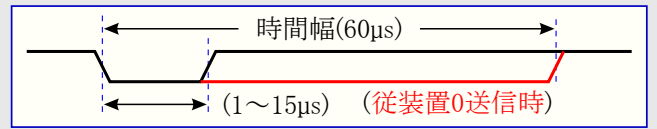
図1-2. “0書き込み”信号



“読み込み”信号

“読み込み”信号は右図で示されます。主装置は1~15 μ s間、バスをLowに引っ張ります。そして従装置は‘0’の送出手を望むなら、バスをLowに保ちます。‘1’の送出手を望むなら、単に線を開放します。バスがLowに引っ張られた後の15 μ s後にバスが採取されるべきです。主装置側から見ると、“読み込み”信号は本質的に“1書き込み”信号(と同じ)です。それが“1書き込み”または“読み込み”のどちらの信号を指示するかは、信号自体よりもむしろ従装置の内部状態です。

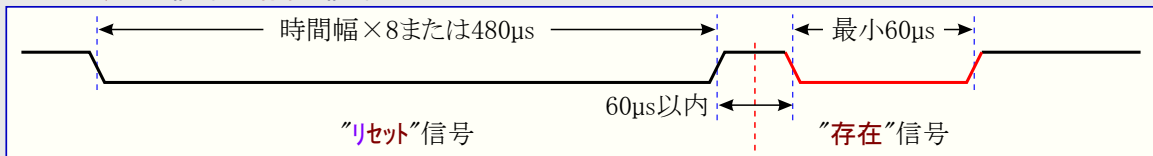
図1-3. “読み込み”信号



“リセット/存在”信号

“リセット”と“存在”の信号は下図で示されます。先の波形と時間尺度が異なることに注意してください。主装置は最低8時間幅、または480 μ s間、バスをLowに引っ張り、そして開放します。この長いLow区間が“リセット”信号と呼ばれます。存在する従装置があるなら、それはバスが主装置によって開放された後の60 μ s以内にバスをLowに引っ張り、最低60 μ s間、Lowを保つべきです。この応答が“存在”信号と呼ばれます。バス上に“存在”信号が全く出されなければ、主装置はバス上に装置が全く存在せず、更なる通信が不可能であると仮定しなければなりません。

図1-4. “リセット”信号と“存在”信号



ソフトウェアでの信号生成

ソフトウェアだけでAVRで1-Wire信号を生成するのは簡単です。単に汎用入出力ピンの方向と値を変更して必要な遅延を生成することで充分です。詳細な説明は「実装」章で与えられます。

UARTでの信号生成

基本的な1-Wire信号はUARTによっても生成することができます。これはバスへの接続にTXDとRXDの両方のピンが必要です。UART出力がHighの時に従装置がバスをLowに引くことを許すため、外部にオープンコレクタまたはオープントレインの緩衝器を必要とします。右図はNPNトランジスタを使う接続を示します。抵抗値は単に雰囲気値です。推奨プルアップ抵抗のより多くの情報については、従装置のデータシートをご覧ください。

1-Wire信号を生成する時に使うUARTデータ形式は8ビットデータ、パリティなし、1停止ビットです。1ビットまたは1つのリセット/存在手順の波形を生成するのに1つのUARTデータフレームが使われます。下表は波形を生成するためのUART構成方法と受信したデータの解釈方法を示します。対応するUARTビット様式は図1-6.~10.で示されます。

図1-5. オープンコレクタ緩衝器

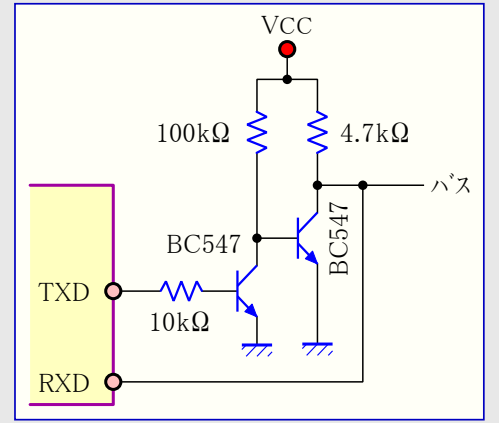


表1-1. UARTによる合図

| 信号 | 波特率 | 送信値 | 受信値 |
|---------|--------|------|----------------------------------|
| 1書き込み | 115200 | \$FF | \$FF |
| 0書き込み | 115200 | \$00 | \$00 |
| 読み込み | 115200 | \$FF | \$FFは'1'ビットと同等で、その他は'0'ビットと同等です。 |
| リセット/存在 | 9600 | \$F0 | \$F0は存在なしと同等で、その他は存在と同等です。 |

図1-6. "1書き込み"信号とUARTビット様式

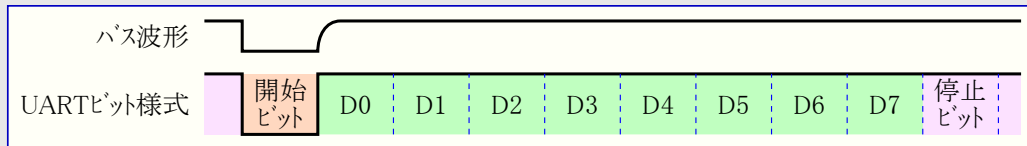


図1-7. "0書き込み"信号とUARTビット様式

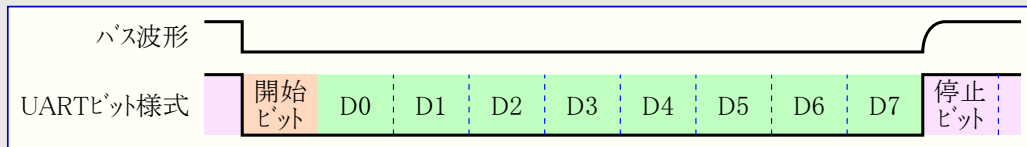


図1-8. "0読み込み"信号とUARTビット様式

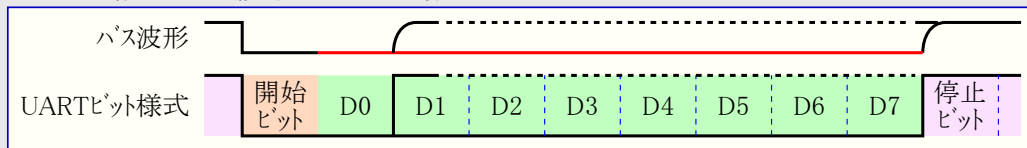


図1-9. "1読み込み"信号とUARTビット様式

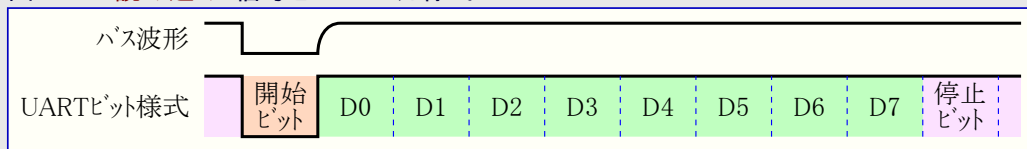
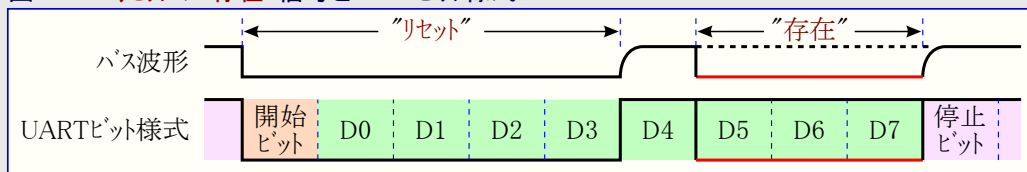


図1-10. "リセット"/"存在"信号とUARTビット様式



1.2. ROM機能命令

全ての1-Wire装置はROM内に格納された世界的に固有の64ビット識別番号を含みます。この番号はバスに於ける個別装置の識別やアドレス指定を容易にするのに使うことができます。識別子は8ビットの系統符号、48ビットの通番、先の56ビットから計算された8ビットのCRCの3つの部分から成ります。64ビット識別子に対して操作する小さな命令群が定義されています。これらはROM機能命令と呼ばれます。右方は6つの定義済みROM命令を一覧にします。

表1-2. ROM命令

| 命令 | 符号 | 使い方 |
|---------------------|------|----------------------|
| READ ROM | \$33 | 認証(ID読み込み(単一従装置用)) |
| SKIP ROM | \$CC | アドレス指定省略(基本的に単一従装置用) |
| MATCH ROM | \$55 | 従装置アドレス指定(次命令応答装置選択) |
| SEARCH ROM | \$F0 | バス上の全ての装置のIDを取得 |
| OVERDRIVE SKIP ROM | \$3C | SKIP ROMの過速(高速動作)版 |
| OVERDRIVE MATCH ROM | \$69 | MATCH ROMの過速(高速動作)版 |

READ ROM命令

“READ ROM”命令は単一従装置バス上で64ビットの固有識別子を読むのに使うことができます。バスに接続された従装置が多数ある場合、この命令の結果は従装置識別子のAND(負論理OR)の結果になるでしょう。通信は完全であると仮定され、多数の従装置の存在が誤ったCRCによって示されます。

SKIP ROM命令

“SKIP ROM”命令は特定の従装置を目標としない時に使うことができます。単一従装置バスではアドレス指定に関して“SKIP ROM”命令で充分です。複数従装置バスでは一度に全ての従装置をアドレス指定するのに使うことができます。これは従装置へ命令を送る、例えば、一度に多数の温度感知器の温度変換を開始する時にだけ有用です。複数従装置バス上の従装置から読む時に“SKIP ROM”命令を使うのは不可能です。

MATCH ROM命令

“MATCH ROM”命令はバス上の個別従装置をアドレス指定するのに使うことができます。“MATCH ROM”命令後、バス上に完全な64ビット識別子が送信されます。これが行われると、次のリセットパルスが受信されるまで、正確にこの識別子を持つ従装置だけが応答を許されます。

SEARCH ROM命令

“SEARCH ROM”命令は先立って全ての従装置の識別子を知らない時に使うことができます。これはバス上に存在する全ての従装置の識別子を発見することを可能にします。最初に“SEARCH ROM”命令がバス上に送信されます。そして主装置はバスから1ビットを読み取ります。各従装置は自身の識別子の先頭ビットをバス上に置きます。主装置はこれを全ての従装置の識別子の先頭ビットの論理AND(負論理OR)の結果として読み取ります。そして主装置はバスから更に1ビットを読み取ります。各従装置は自身の識別子の先頭ビットの補数(論理反転値)をバス上に置きます。主装置はこれを全ての従装置の識別子先頭ビットの補数の論理AND(負論理OR)の結果として読み取ります。全ての従装置が先頭ビットとして1を持つなら、主装置は‘10’を読むでしょう。同様に全ての従装置が先頭ビットとして0を持つなら、主装置は‘01’を読むでしょう。これらの場合では全アドレスの先頭ビットとしてこのビットを格納することができます。そして主装置はこのビット書き戻し、要するに送っている識別子のビットの維持を全ての従装置に告げます。バス上に識別子の先頭ビットとして0と1の両方の装置がある場合、主装置は‘00’を読むでしょう。この場合の主装置はこの時点で0か1のどちらかを持つアドレスでの継続を決めなければなりません。この選択がバス上に送信され、現実的に識別子のこの時点に於けるこのビットを持たない全ての従装置をアイドル状態にします。(訳補:即ち1ビットの確定に3ビットを要します。)

そして主装置は次ビットの読み込みを進め、64ビットを読むまでこの処理が繰り返されます。そして主装置は1つの完全な64ビット識別子を見つけたでしょう。もっと識別子を見出すには“SEARCH ROM”命令を再び動かすべきですが、この時に先刻に於いて不一致があったビット値に対して違う選択にされるべきです。各従装置に対するこの1度の繰り返しが全ての従装置を見出すでしょう。1つの検索が実行されてしまうと、1つ以外の全ての従装置がアイドル状態へ移行されていることに注意してください。“MATCH ROM”命令での特別なアドレス指定なしに活性の従装置と直ぐに通信することが可能です。

OVERDRIVE ~ ROM命令

“OVERDRIVE ~ ROM”命令は、過速(高速)動作がこの資料の範囲外のため、ここで網羅されず、標準速だけが網羅されます。

1.3. メモリ/機能命令

メモリ/機能命令は1つの装置または装置の分類部を指定する命令です。これらの命令は代表的に従装置内部のメモリとレジスタの読み書きを扱います。多数のメモリ機能命令が定義されていますが、全ての命令が全ての装置によって使われる訳ではありません。読み書きの命令は各装置特有で、一般的な仕様の部分ではありません。従ってメモリ機能命令はここで詳細に網羅されません。

1.4. 一斉に適用

全ての1-Wire装置は基本通信手順に従います。

1. 主装置が“リセット”パルスを送ります。
2. 従装置が“存在”パルスで応答します。
3. 主装置がROM機能命令を送ります。これが1つまたは多数の従装置を効率的にアドレス指定します。
4. 主装置がメモリ機能命令を送ります。

注: 各段階に至るには最終段階が完了されなければなりません。けれども手順全体を完了する必要はありません。例えば、ROM機能命令を終了した後で新しい通信を始めるために新規の“リセット”を送ることが可能です。

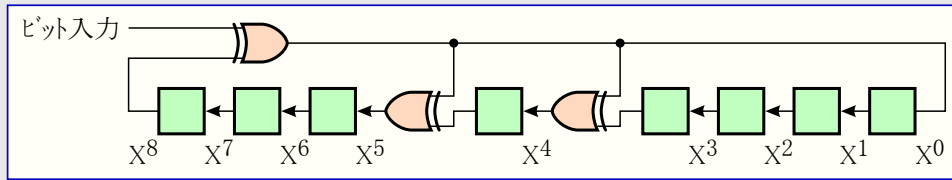
1.5. 巡回冗長検査

データの完全性を保証するために1-Wire装置によって巡回冗長検査(CRC)が使われます。CRCの背後にある原理はこの資料の範囲外で、更なる検討は行われません。CRCのより多くの情報については「参考文献」の2をご覧ください。

1-Wire装置に於いて2つの異なるCRCが主に見られます。1つは8ビットCRC(Dallas 1-Wire CRC, DOW-CRCまたは単にCRC8)で、もう1つは16ビットCRC(CRC16)です。CRC8は全ての装置のROM部で使われます。CRC8はいくつかの装置に於いてバス上に発行される命令のように他のデータを検証するのにも使われます。CRC16はより大きなデータ群の誤りを調べるためにいくつかの装置によって使われます。

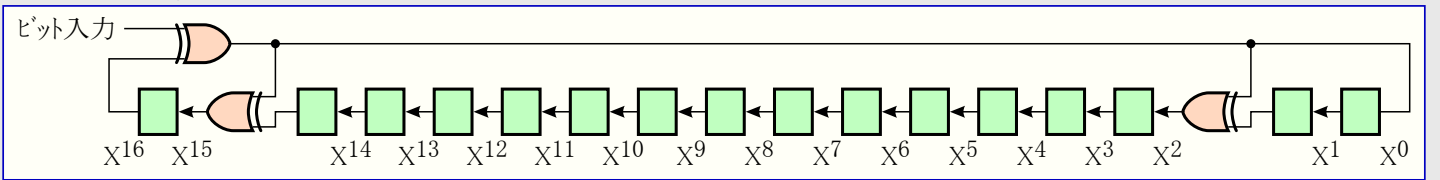
64ビット識別子で使われる8ビットCRCと等価なハードウェアが下図で示されます。(四角の)塊は8ビット移動レジスタ内の個別ビットを表します。等価なCRC多項式は $X^8+X^5+X^4+1$ です。

図1-11. 1-Wire装置で使われる8ビットCRCと等価なハードウェア



いくつかの1-Wire装置で使われるCRC16と等価なハードウェアが下図で示されます。(四角の)塊は16ビット移動レジスタ内の個別ビットを表します。等価なCRC多項式は $X^{16}+X^{15}+X^2+1$ です。

図1-12. 1-Wire装置で使われる16ビットCRCと等価なハードウェア



2. 実装

ここで、ソフトウェアのみ(ポーリング)、ポーリングUART、割り込み駆動UARTの3つ異なる1-Wire実装が検討されます。各々の短い説明が下で与えられます。ドライバの使い方についての詳細な情報はこの資料に含められていません。各種ドライバの使用法の詳細についてはこの応用記述に関するソースコードに含められた資料をご覧ください。

どんな特別なハードウェアを使うこともなくソフトウェアだけで1-Wire規約を実装することが可能です。この解決策は占有するハードウェアが1つの汎用入出力ピン(GPIO)だけの利点を持ちます。AVRの全てのGPIOが双方向で選択可能な内部プルアップ抵抗を持つので、AVRは外部の支援回路を全くなして1-Wireバスを制御することができます。内部プルアップ抵抗が従装置の電流構成設定に適合しない場合は、1つの外部抵抗が必要とされるだけです。良くない傾向について、この実装は“リセット/存在”とビットの信号中に忙しく待つことで信頼します。1-Wireバスの正しいタイミングを保证するため、ビットの送信中に割り込みが禁止されなければなりません。2つのビットの送信間に許される遅延(回復時間)は上限を持ちませんが、とはいえ、このため、毎ビット送信後に割り込みを処理の方が安全です。これは1-Wireバス活動が“リセット/存在”の実行時間と等価なので、最悪割り込み遅延を1ms未満にします。

ポーリングUARTドライバはビット単位で必要な波形を生成するのに多くのAVRで見られるUART単位部を使います。ドライバの残りはソフトウェアだけのドライバと同じです。ソフトウェアだけのドライバと比べてこのドライバでの主な利点はコード量と、UART単位部が独立して詳細なタイミングを処理するため、ビット信号中に割り込みをOFFにする必要がないことです。良くない傾向について、2つのGPIOといくつかの外部支援回路が必要です。

割り込み駆動UARTは波形を生成するのにポーリングUARTドライバと同じ様にUARTを使います。加えて255ビットまでのデータを自動で送信または受信するためにUART単位部で得られる割り込み能力の利点を利用します。

2.1. ホールリンクドライバ

ホールリンクドライバは2つの部分に分けられます。それはビット単位の波形生成と、バイトの送信やROM機能命令の実行のようなより高位の命令です。ビット単位処理だけが2つの版間で異なりますが、それらは共通インターフェースで実装され、どちらのドライバでの使用もより高位の命令に許します。

2.1.1. ソフトウェアのみの実装

この応用記述で提供されるソフトウェアのみの実装で、1つのAVRに接続された多数の1-Wireバスを持つことが可能です。けれども、全てのバスは同じ入出力ポートに接続されなければならない、そのポートはコンパイル時の任意選択です。これはバス数を8に制限しますが、ポート内のバスの配置は完全に構成設定可能です。1-Wireバスに使われない全てのピンは影響を及ぼされません。全ての1-Wireバスが同じポートに接続されているので、多数の操作が同時に1つまたはより多くのバス上で実行することができます。これは全ての関数へ渡されるピン引数を通して達成されます。この引数はその操作に対して使われるべきピンのビット遮蔽を含むでしょう。例えばピン引数として\$PFを渡すことによって同時に8つのバスへリセット信号を送ることが可能です。この同じ関数から返される値は、1つ以上の従装置が存在信号で応答した全てのバスのビット遮蔽値です。そしてこのビット遮蔽値はSKIP ROM命令を発行する関数へのピン引数として渡され、以下同様です。この実装内の全ての関数がピン選択を支援します。一般的な規則として、バスへ書く全ての関数は同時に多数のバスをアドレス指定することができます。バスからビットよりも多くを読む命令は何らかの方法で1つのバスだけをアドレス指定することができます。

初期化

ソフトウェアのみの1-Wireインターフェースに関する初期化処理は実に単純です。それは1-Wireピンを入力動作に設定するだけと、バスをアイドル動作にして置くことが必要とされる場合に内部プルアップを許可することから成ります。いくつかの装置がバス上のこの上昇端をリセット信号の終りとして反応し、存在信号で返答するでしょう。この信号がどの通信をも妨害しないことを保証するために、リセット回復時間に等しい長さの遅延が挿入されます。

ビット単位関数

ビット単位関数はDallas SemiconductorsからのAN126応用記述に従って実装されます。全てのタイミング項目はこの応用記述内の推奨値に適合します。10種の異なる遅延が必要とされます。これらは右表で一覧にされます。

注: G遅延は標準動作で0です。

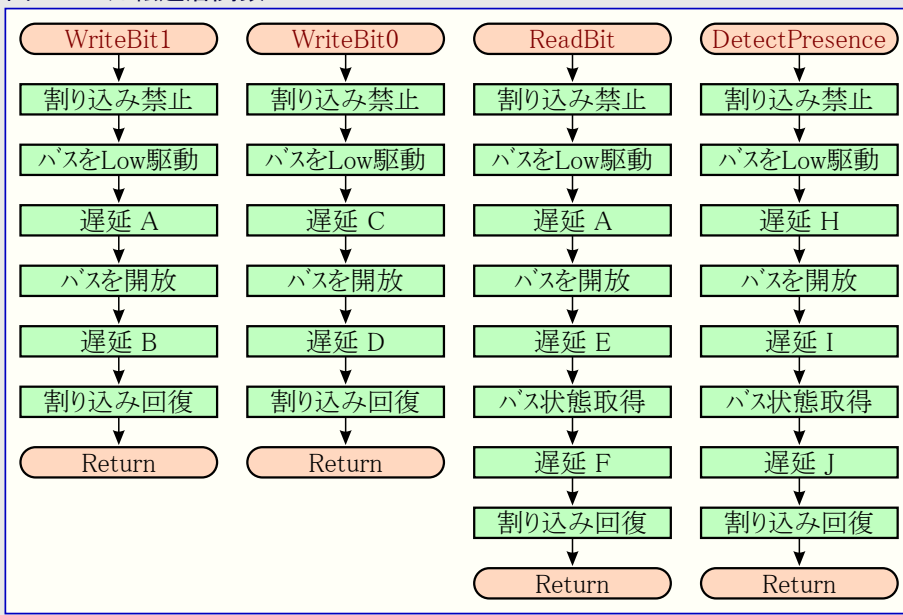
I/O操作がアセンブリ言語ではなく、C言語で実装されているため、コンパイラの最適化とその他の要因がタイミングに影響を及ぼし得ます。各ビット単位関数によって生成された波形をオシロスコープで監視し、必要ならば遅延を調整することが推奨されます。

ビット転送層関数は下図で示されるように実装されています。“DetectPresence”関数が“リセット”信号送出口と“存在”信号聴取の両方であることに注意してください。ビット転送層関数が同時に多数のバスをアドレス指定できることに留意してください。

表2-1. ビット転送層遅延

| 項目 | 推奨遅延 (μs) |
|----|-----------|
| A | 6 |
| B | 64 |
| C | 60 |
| D | 10 |
| E | 9 |
| F | 55 |
| G | 0 |
| H | 480 |
| I | 70 |
| J | 410 |

図2-1. ビット転送層関数



バスをLowに駆動するのとバスを開放するために2つのマクロが含まれています。これらが頻繁に起き、関数呼び出しによって引き起こされる副次的な付随処理が厳密なタイミングの必要条件のために望まれないので、これらはマクロとして実装されています。

2.1.2. UARTホーリング実装

この実装では詳細タイミングの全てがUART単位部で処理されます。ビットを送出するためにUARTホーレートが適切な値に設定され、UARTデータレジスタが「UARTでの信号生成」項で記述されるように望む波形を生成する値を格納されます。

初期化

ホーリングUARTドライバ用に1-Wireインターフェースを初期化するために、UART単位部は正しい値で初期化されなければなりません。送信と受信を許可し、データ形式を8ビット、パリティなし、1停止ビットに設定し、そしてホーレートを115200bpsに設定します。

これはTXDピンをUARTのアイドル状態にさせ、そしてこれは論理1(High)です。従装置はこの上昇端をリセット信号の終りとして解釈し、存在信号で応答します。

ビット単位関数

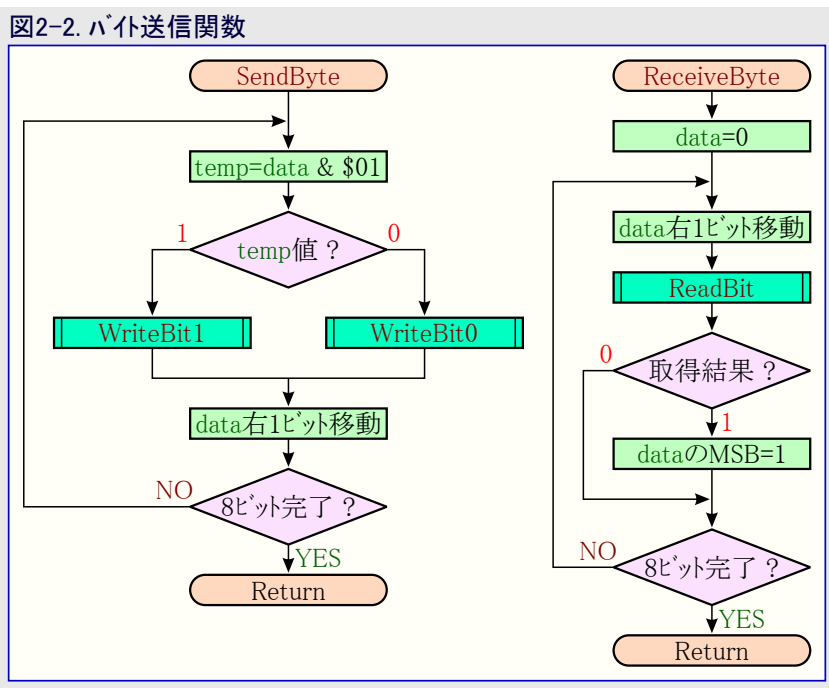
ホーリングUARTドライバに於ける全てのビット単位関数はOWI_TouchBitと呼ばれる1つの共通関数を通して実行されます。この関数は最初に入力引数をUART単位部へ出力し、UART受信が完了するまで待ち、そして受信値を返します。ビット単位関数の各々はバス上に正しい波形を生成する値と共にOWI_TouchBitを呼び出します。

これら関数へのインターフェースはソフトウェアのみの実装と同じです。けれども'pins'引数はホーリングUARTドライバに必要ありません。マクロ群がピン引数有りまたはなしでのこれらの関数呼び出しを可能にします。pins引数が含まれている場合はマクロによって取り除かれます。

2.1.3. 上位関数

この層内の多くの関数が符号なしchar型ポインタの引数を許容することに注意してください。このポインタは関数によって使うことができるメモリの8バイト配列を指し示します。これらの配列の割り当て時に初期化は呼び出し側によって行われなければなりません。関数を呼ぶ前にメモリが特別な方法で初期化されなければならない時をこの資料では明確に述べます。

2.1.3.1. バイト送信関数



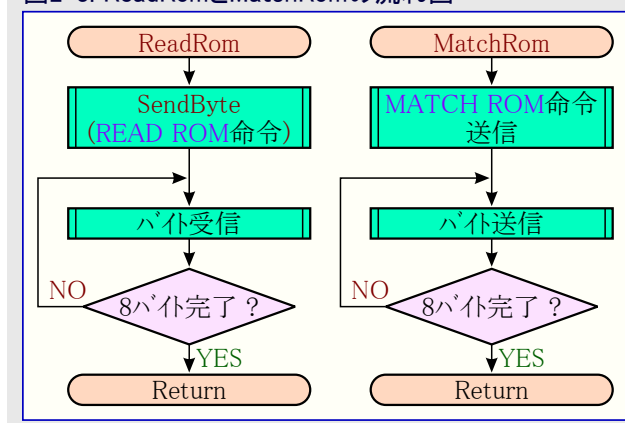
2.1.3.2. ROM機能命令

標準速通信に関する全ての標準ROM機能命令が実装されています。

最も簡単なROM機能命令はSKIP ROM命令です。これは単に引数としてSKIP ROM命令でSendByte関数を呼び出します。

READ ROMとMATCH ROMの命令に対する流れ図は右図で示されます。

図2-3. ReadRomとMatchRomの流れ図



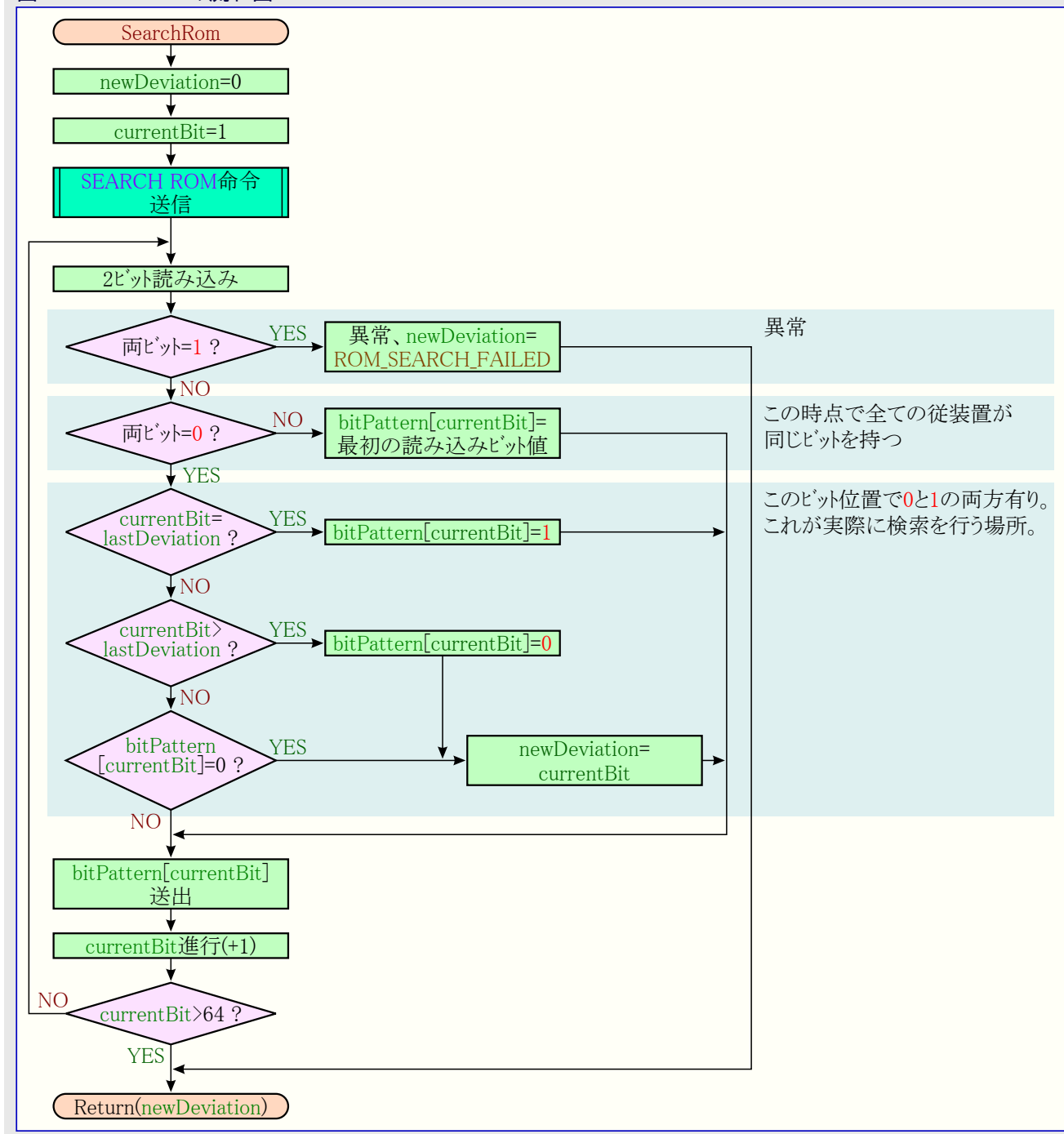
SEARCH ROM命令用の流れ図が下図で示されます。この関数はバス上で未発見の従装置が全なくなるまで、それが走行する時毎に1つの従装置を探します。それが走行する最終回にOWI_ROM_SERCH_FINISHEDを返します。どのバス上で検索を実行するのかわを選択する'pin'引数に加えて、この関数に'LastDeviation'と'bitPattern'の2つの引数が渡されなければなりません。これらの引数が従装置検索を制御します。全ての従装置に対して全検索を完了するのにこれらの関数をどう使うかを理解するには下表を参照してください。

表2-2. bitPatternとlastDeviationの使い方

| 走行種別 | bitPattern | lastDeviation |
|--------|---------------------------------------|-------------------------|
| 初回 | 0で満たした8バイト配列 | 0 |
| 後続する走行 | 直前の走行でbitPatternポインタを通して返された8バイト配列の複製 | 直前のOWI_SearchRomから返された値 |

関数は呼び出し側に最大の柔軟性を与えるためにこの方法で実装されています。全検索を実装するのにポーリングドライバ用のソフトウェア例を使うことができます。

図2-4. SearchRomの流れ図



2.1.4. タイミングの考慮

可能な限り正確に波形を生成できることが重要です。これを行うには正確な遅延が必要です。或る μs の数値に対する遅延に必要なクロック周期数がコンパイル時に計算されます。波形生成時に於いて、バスをLowに引っ張る時とバスを開放する時にいくつかのクロック周期が失われます。これらのクロック周期は遅延を生成するのに必要とするクロック周期数から引かれます。クロック周波数が低すぎる場合、これは負の遅延を生成し得ます。最短遅延を生成することができるには2.17MHzよりも高いクロック周波数が必要とされます。

2.1.5. 割り込み駆動UART実装

割り込み駆動UARTドライバはポーリングUARTドライバと同じハードウェア必要条件を持ちます。

この応用記述で提供される割り込み駆動実装の基本的な機能はバス上のより大きな塊のデータを自動で送受信することです。これは2つの割り込み処理ルーチン(ISR)で行われます。必要なパラメータの全てを構成するために1群の補助関数を呼び出すことができ、これらのISRが転送処理を自動的に完了します。**リセット/存在**の手順、または介在なしで1~255ビット間のデータを1方向でどこへでも転送を行うことが可能です。

ISRを可能な限り単純にするために、それらは送受信間で区別をしません。**UDRE** ISRはそれが動く時毎にデータ緩衝部から1ビットを単に送出します。**RXC** ISRはその同じビットを受信し、例えデータがどの方向に送られたとしても、データ緩衝部にそれを戻し置きます。送信中、送ったデータは受信したのと同じで、データ緩衝部は無変化に留まります。受信中、**'1書き込み'波形**が**'読み込み'波形**と同じなので、**'1'**だけが送信されるべきです。従装置によって書かれた(送信された)値を得るために信号が採取されます。そしてこの値がデータ緩衝部に置かれます。

3つの全体フラグが1-Wire[®]ドライバの状態を合図します。それは多忙、存在、異常です。多忙フラグは転送するデータがもっとある限り設定(1)されます。存在フラグは**リセット**信号送出時に**存在**信号が検出された場合に設定(1)されます。このフラグは**存在**信号が返らないバス上の**リセット**信号まで設定(1)に留まります。異常フラグはUART受信部がフレーミング誤りを検出した時に設定(1)されます。この場合、バス上に新規の**リセット**信号が送信されるべきです。これは**UDRE**と**RXC**のISRの内部状態だけでなく、バス上の全ての従装置をリセットします。

ISRは可能な限り素早く実行されるべきなので、**ROM機能命令**のようなもっと複雑な機能はISRに実装されていません。含まれているコード例はこのような動きが有限状態機構(FSM:Finite State Machine)でどう実装できるかを示します。

2.1.5.1. 割り込み処理ルーチン

ISR用の流れ図は次の2つの図で示されます。UARTデータレジスタ空(**UDRE**)ISRはUART送信緩衝部にデータ用の空きがある時毎に動きます。UART受信完了(**RXC**)ISRはデータが受信されてしまい、UART受信緩衝部内で準備が整っている時毎に動きます。補助関数の流れ図は**補助関数の図**で支援されます。

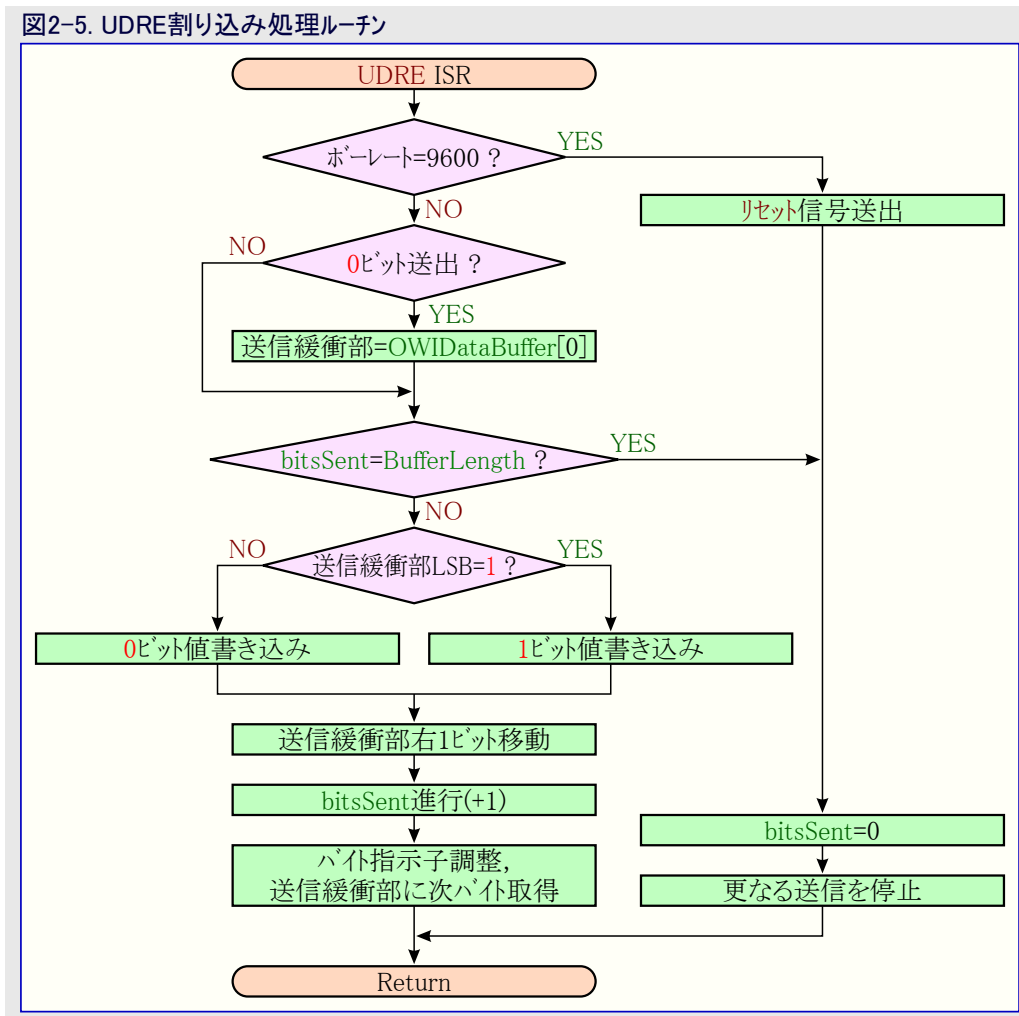
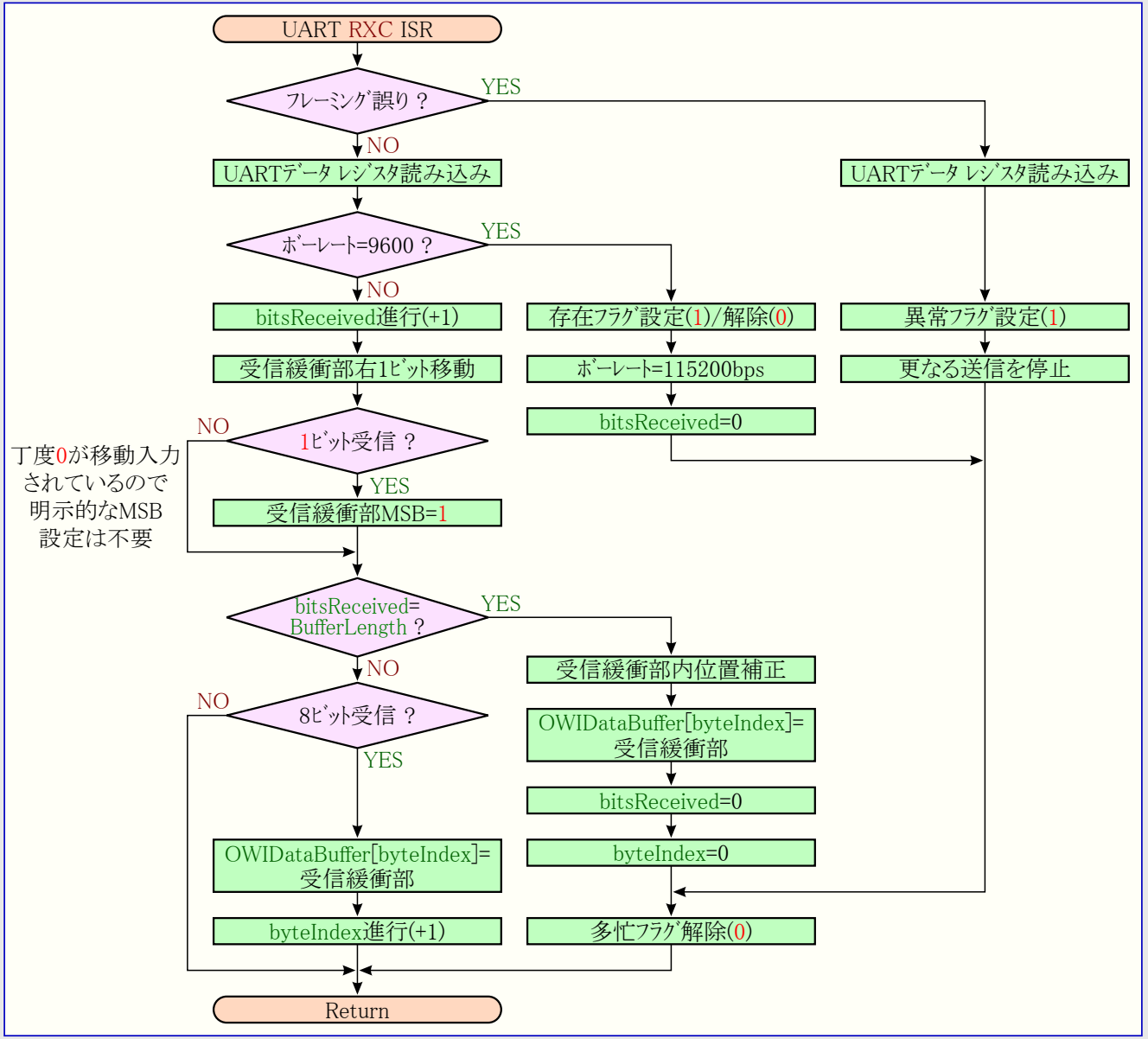


図2-6. RXC割り込み処理ルーチン



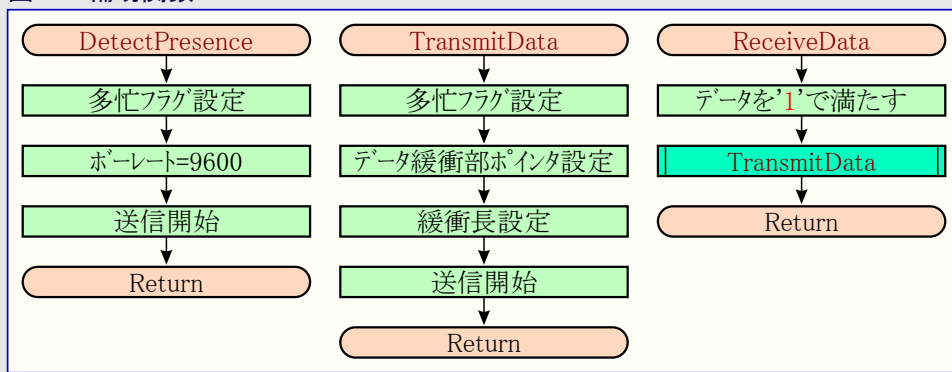
2.1.5.2. 補助関数

補助関数は後に来る自動化された割り込み駆動送信に必要ないくつかのパラメータを設定します。全ての必要なパラメータを設定した後でUDRE割り込みを許可することによって送信が開始されます。

補助関数用の流れ図は下図で示されます。

ReceiveData関数が実際にはデータ緩衝部を'1'で満たしてTransmitData関数を呼び出すことに注意してください。RXC ISRが信号を採取して従装置から読んだ値をデータ緩衝部に置きます。

図2-7. 補助関数



2.2. CRC計算

2つの異なるCRCを計算するのに使われる方法が下で記述されます。

CRC(関数内はseed)は0またはCRCの”種”のどちらかが設定されます。これは以下で説明されます。

1. CRCのLSBとデータのLSB間の”排他的論理和”を得ます。
2. この値が0なら、CRCを1ビット右移動します。
3. この値が1なら、
 - 3.1. CRCとCRC多項式の”排他的論理和”を取ることによって新しいCRC値を得ます。
 - 3.2. CRCを1ビット右移動します。
 - 3.3. CRCのMSBを1に設定します。
4. データを1ビット右移動します。
5. 完全に連続8回繰り返します。

この方法はCRC8とCRC16の両方を計算するのに使うことができます。違いはCRC移動レジスタの幅(CRC8用の8ビット、CRC16用の16ビット)と多項式の値だけです。この数値はハードウェアに於けるXORゲートの接続を偽装します。多項式の値はCRC8用が\$18で、CRC16用が\$4002です。

この方法は一度に1バイトのCRC値を得るように実装されていますが、CRCの”種”は引数としてCRCルーチンへ渡すことができます。このようにして1つのCRC操作の結果が次のバイトと共に次のそれへ渡すことができ、要するに任意のバイト数のCRCを計算します。

64ビット識別子のCRC検査がOWI_CheckRomCRCで実装されています。これは単に最初の56ビットのCRC8値を計算して識別子の最後の8ビットと比較します。

2.3. コード例

含められた2つのコード例は1-Wireドライバの各種実装を使う方法を示します。

2.3.1. ホールリング例

ホールリング用のコード例は装置に関して”BUSES”によって定義されたバスを検索します。装置はOWI_device型の配列に格納されます。OWI_deviceは装置がどのバスに接続されているかと64ビット識別子についての情報を含む構造体です。そしてドライバはポートD(PD0)上のDS1820温度感知器に関して利用可能な従装置を全体的に検索します。バス上にこの装置が見つかった場合、それは無限繰り返しで継続的にやり取りされます。各反復に於いて、DS1820の温度が監視され、例えばSTK[®]600開発基板のLEDで監視することができますように温度がポートBに出力されます。

このコード例はドライバの異なる部分がどう使われ得るかを示すことが意図されています。コードは非常に一般的で、客観的に最適化されていません。このため、コード例が4Kバイト未満のプログラムメモリを持つデバイスに適合しないことに注意してください。けれども、ドライバは1Kバイトのデバイスを含めて、全てのAVRマイクロコントローラで完全に適合します。

2.3.2. 割り込み駆動例

割り込み駆動例では有限状態機構(FSM)が実装されています。ドライバがバス上データを送信するので忙しくないなら、無限繰り返しからFSMが呼ばれます。ドライバが多忙のとき、他のどのコードにも走行を許すためにFSMが飛ばされます。FSM自身はバス上に単独のDS1820温度感知器が利用可能であると仮定します。それは現在温度を読んでそれが正しく読まれたことを保証するためにCRCを計算します。そして温度が全体変数に置かれます。ドライバが多忙の時は必ず無限繰り返しがこの温度をポートBに出力し、このために例えばSTK600開発基板のLEDで監視することができます。

3. 始める前に

本章はこの応用記述に含まれるコード例でどう始めるかの概要です。

3.1. ソースコード

例のコードはAtmel START(開始)用には書かれています。これはAtmel Studio 7とIAR™ IDEの両方用のAtmel STARTの入り口の”BROWSE EXAMPLES(例閲覧)”からダウンロードすることができます。

ダウンロードした.atzipファイルをダブルクリックしてください。プロジェクトがAtmel Studio 7にインポートされるでしょう。IAR™でプロジェクトをインポートするには、”Atmel START in IAR(IARでのAtmel開始)”を参照し、Atmel Start Output in External Tools⇒IARを選んでください。

3.1.1. ホーリングドライブ

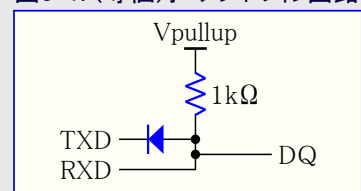
ホーリングドライブに於ける各ファイルの短い説明は下表で示されます。

表3-1. ホーリングドライブ ファイル

| ファイル名 | 内容 |
|-------------------------|---|
| main.c | ホーリングドライブ用コード例 |
| OWISWBitFunctions.c | ソフトウェアのみのビット単位関数の実装 |
| OWIUARTBitFunctions.c | UARTビット単位関数の実装 |
| OWIBitFunctions.h | OWISWBitFunctions.cとOWIUARTBitFunctions.c用の共通ヘッダ ファイル |
| OWIHighLevelFunctions.c | 上位関数 |
| OWIHighLevelFunctions.h | OWIHighLevelFunctions.c用の共通ヘッダ ファイル |
| OWIPolled.h | ホーリングドライブ用の構成設定ヘッダ ファイル |
| source.doc | このフォルダのソースコードの資料 |

- Atmel StudioプロジェクトまたはIARプロジェクトを開いてください。(Atmel STARTから.atzipをダウンロード後にAtmel StudioまたはIARにインポート)
- 編集するために"OWIPolled.h"ファイルを開き、"User defines(使用者定義)"と名付けられた項を突き止めてください。
- ファイルで記述されるように行の1つの注釈を外すことによって'software only'または'UART driver'を選んでください。
- 選んだドライブに対応する項へ下に移動してください。
- ファイルで記述されるようにハードウェア構成設定に従って項内の定義を調節してください。
- プロジェクトは今やコンパイルされる準備が整っています。
- ドライブの動作形態はOWIPolled.hファイルからOWI_SOFTWARE_DRIVERまたはOWI_UART_DRIVERとして選ぶことができます。OWI_UART_DRIVER動作については、右図で示されるようにTXDとRXDのピンが接続されるのに(等価)オープンドレイン回路が必要です。DQは1-Wire装置からの単線インターフェースです。

図3-1. (等価)オープンドレイン回路



3.1.2. 割り込み駆動ドライブ

割り込み駆動ドライブに於ける各ファイルの短い説明は下表で示されます。

表3-2. 割り込み駆動ドライブ ファイル

| ファイル名 | 内容 |
|----------------------|-----------------------------|
| main.c | 割り込み駆動ドライブ用コード例 |
| OWIInterruptDriven.h | 割り込み駆動ドライブ用の構成設定ヘッダ ファイル |
| OWIIntFunctions.c | 割り込み処理と補助関数の実装 |
| OWIIntFunctions.h | OWIIntFunctions.c用のヘッダ ファイル |
| source.doc | このフォルダのソースコードの資料 |

割り込み駆動ドライブで始めるには以下の段階に従ってください。

- Atmel StudioプロジェクトまたはIARプロジェクトを開いてください。(Atmel STARTから.atzipをダウンロード後にAtmel StudioまたはIARにインポート)
- 編集するために"OWIInterruptDriven.h"ファイルを開き、"User defines(使用者定義)"と名付けられた項を突き止めてください。
- ハードウェア構成設定を反映するように"User defines"項で定義を変更してください。
- プロジェクトは今やコンパイルされる準備が整っています。
- 「(等価)オープンドレイン回路」で示されるようにTXDとRXDのピンが接続されるのに(等価)オープンドレイン回路が必要です。

4. 参考文献

1. 応用記述126「ソフトウェアによる1-Wire通信」 - Dallas Semiconductors, 2004
2. iButton規約の本 - Dallas Semiconductors, 1997
3. 応用記述214「1-Wireバス主装置に対するUARTの使い方」 - Dallas Semiconductors, 2002

5. 改訂履歴

| 文書改訂 | 日付 | 注釈 |
|-------|----------|------------------|
| 2579A | 2004年9月 | 初版文書公開 |
| 2579B | 2016年10月 | Atmel STARTコード公開 |

Atmel®, Atmelロゴとそれらの組み合わせ、Enabling Unlimited Possibilities®, AVR®, megaAVR®, tinyAVR®, STK®とその他は米国及び他の国に於けるAtmel Corporationの登録商標または商標です。他の用語と製品名は一般的に他の商標です。

お断り: 本資料内の情報はAtmel製品と関連して提供されています。本資料またはAtmel製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。Atmelのウェブサイトに表示する販売の条件とAtmelの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、Atmelはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえAtmelがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益と損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してAtmelに責任がないでしょう。Atmelは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。Atmelはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、Atmel製品は車載応用に対して適当ではなく、使用されるべきではありません。Atmel製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

安全重視、軍用、車載応用のお断り: Atmel製品はAtmelが提供する特別に書かれた承諾を除き、そのような製品の機能不全が著しく人に危害を加えたり死に至らしめることがかなり予期されるどんな応用(“安全重視応用”)に対しても設計されず、またそれらとの接続にも使用されません。安全重視応用は限定なしで、生命維持装置とシステム、核施設と武器システムの操作の装置やシステムを含みます。Atmelによって軍用等級として特に明確に示される以外、Atmel製品は軍用や航空宇宙の応用や環境のために設計も意図もされていません。Atmelによって車載等級として特に明確に示される以外、Atmel製品は車載応用での使用のために設計も意図もされていません。

© HERO 2021.

本応用記述はAtmelのAVR318応用記述(Rev.2579B-10/2016)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。