お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

お客様各位

資料中の「三菱電機」、「三菱XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って株式会社日立製作所及び三菱電機株式会社のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。
従いまして、本資料中には「三菱電機」、「三菱電機株式会社」、「三菱半導体」、「三菱XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

注:「高周波・光素子事業、パワーデバイス事業については三菱電機にて引き続き事業運営を行います。」

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

M16C/62 グループ
簡易 I²C バス編

1.0 要約

「M16C/62 グループ アプリケーションノート＜簡易 I²C バスモード＞」は、三菱 CMOS16 ビットマイクロコンピュータ M16C/62 グループに内蔵されている簡易 I²C バスモードを制御していただくための参考資料です。なお当資料は I²C バスの通信動作を保証するものではありませんので、ご使用の際には十分に評価を行ってください。M16C/62 グループの命令体系については、「M16C/60 シリーズ ソフトウェアマニュアル」を合わせてご利用ください。M16C/62 グループのハードウェアにつきましてはご使用品種のユーザーズマニュアルを、開発サポートツールにつきましては各ツールの操作説明書をご利用ください。

2.0 はじめに

この資料で説明する応用例は、次のマイコンでの利用に適用されます。
マイコン： M16C/62 グループ （M30622M4A-xxxFP）
発振周波数： 10MHz

3.0 応用例の紹介

M16C/62 グループは、シリアル I/O 回路（UART2）に簡易 I²C バス回路が搭載されています。簡易 I²C バス回路をソフトウエアと組み合わせて使用することにより、I²C バスインターフェースの制御が可能となります。本アプリケーションノートでは、I²C バス仕様の概要を説明し、簡易 I²C バス機能で I²C バスインターフェースを実現するための各機能の使い方およびプログラムをご紹介致します。
本書を使用するにあたって、電気回路、論理回路、およびマイクロコンピュータの基本的な知識が必要です。
以下に目的に応じた参照先を示します。

- I²C バスプロトコルを理解したい
  → 3.1 I²C バス仕様概要
- M16C/62 のシリアル I/O の構成を知りたい
  → 3.2 UART2 の機能 3.2.1 UART2 の有する機能
- M16C/62 簡易 I²C バスのブロック図およびレジスタ構成を知りたい
  → 3.2 UART2 の機能 3.2.2 ～3.2.4
- M16C/62 の簡易 I²C バスの各機能の使い方を理解したい
  → 3.3 簡易 I²C バスの各機能
- 簡易 I²C バスモードを使用する際に注意することを知りたい
  → 3.4 簡易 I²C バスモードの注意事項
- M16C/62 を使用した I²C バスインターフェース部のプログラムを参考にしたい
  → 4.0 参考プログラム

3.1 I²C バス仕様概要

I²C バスとは、PHILIPS 社が開発した双方向のマルチ・マスタ・バスの通信制御プロトコルであり、現在数多くの IC に採用されています。I²C バスの詳細な仕様は、PHILIPS 社の I²C バス仕様書をご覧下さい。
はじめに本章では、I²C バスの仕様の概要について説明します。

3.1.1 I²C バスの特徴

I²C バスは、IC 間相互の制御を効率よく行うことを目的とした、2 本のワイヤからなる構造の簡単な双方向バスです。I²C バスの特徴を、いくつかご紹介します。

- 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 本のバス・ラインのみで構成されます。

- バスに接続されている各装置はそれぞれ固有のアドレスを持ち、装置間にはマスタ（*1 ）とスレーブ（*2 ）という単純な関係が常に成り立ちます。マスタはマスタ送信装置またはマスタ受信装置として機能し、またスレーブはスレーブ送信装置またはスレーブ受信装置として機能します。

- 複数のマスタが同時にデータ転送を開始しようとした場合でも、データ破壊を防ぐための衝突検出機能（*3 ）を含む通信調整手順（*4 ）を備えているため、マルチ・マスタ 動作（*5 ）が可能です。

- 高速モードにおける双方向シリアルデータ転送は、最高 400kbit/s を実現します。

(*1)データ転送を開始し、クロック信号を生成し、データ転送を終了する装置

(*2)マスタからアドレス指定される装置

(*3)複数のマスタがデータを送出したときに、自分と異なるレベルのデータが送出されたことを検出する機能

(*4)複数のマスタが同時にバスを制御しようとしたときに、ひとつのマスタだけがバスを制御できるようにし、さらに、メッセージが失われたり、内容が変更されないようにする手順

(*5)メッセージを失うことなく、複数のマスタが同時にバスを制御しようとすること

3.1.2 I²C バスの概念

I²C バスは、２本のワイヤ(SDA および SCL)によって、バスに接続されている装置間でのデータ転送がおこなわれます。各装置は固有のアドレスによって認識され、装置の機能に応じて送信装置または受信装置として動作します。これらの装置は、データ転送を行う時クロック信号を送出する装置がマスタとなり、マスタによってアドレス指定される装置（１つまたは複数）はスレーブとなります。マスタがデータ転送を行う場合は、マスタがデータ転送のタイミングをとるためのクロック信号を送出して、スレーブのアドレスを指定（転送）します。その後送信側が受信側に対してデータを送信し、マスタによってデータ転送が終了されます。

I²C バスでは、常にマスタによってクロック信号が生成されます。従って、バス上でデータのやりとりが行われる場合には、各マスタがそれぞれ独自のクロック信号を生成することになります。マスタの生成するクロック信号は、SCL を Low に保持する低速のスレーブ装置（3.1.4 「SCL 同期を使用した協調手順」参照）と、通信調整手順中の他のマスタ（3.1.4 「SCL 同期」参照）によって変更されることがあるのみです。

SDA と SCL はどちらも双方向ラインであり、並列抵抗を通じて正の電源電圧に接続されています。バスが解放されているときには、どちらのラインも High の状態になります。バスに接続されている装置の出力段には、AND 接続機能を実行するためにオープンドレインまたはオープンコレクタが必要となります(*)。

(*)M16C/62 の SDA、SCL 出力端子はＮチャネルオープンドレイン出力となります。

3.1.3 データ転送

I²C バスのデータ転送のフォーマットは、次のように定義されています。

データの有効性

SCL が High の間は、SDA の状態は一定でなければいけません。SDA のレベルを変更できるのは、SCL が Low の時に限られます。

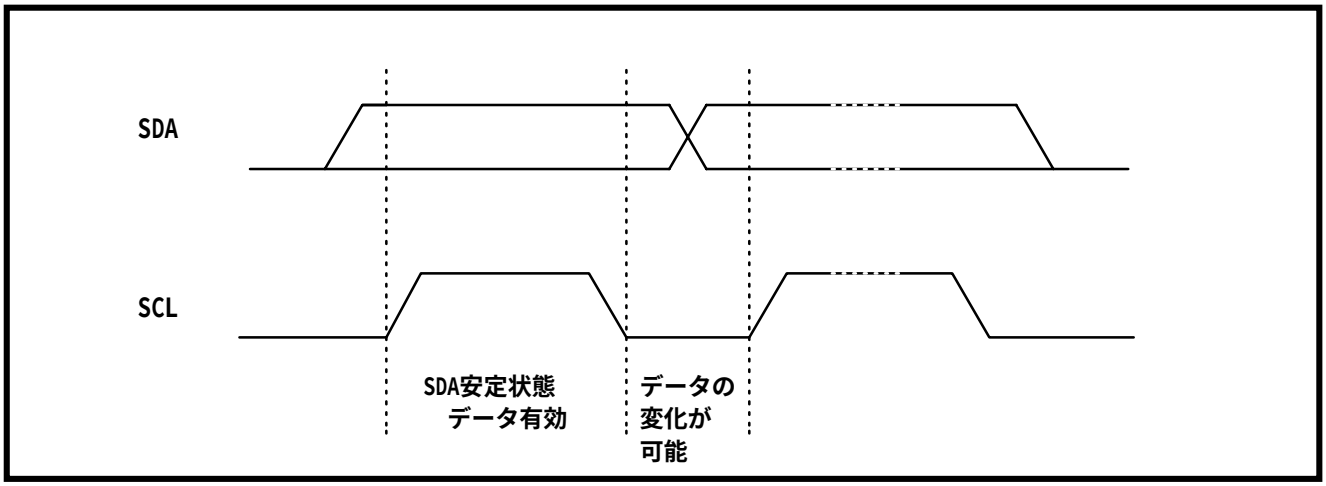


図 3.1.1 **データの有効性**

開始条件と停止条件

I²C バスの通信手順では、マスタによる開始条件の送出でデータ転送が始まり、停止条件の送出でデータ転送が終了します。SCL が High のときに、SDA が High から Low に変化する状況は、開始条件（スタートコンディション）と呼ばれます(*1)。また SCL が High のときに、SDA が Low から High に変化する状況は、停止条件（ストップコンディション）と呼ばれます(*2)。

開始条件、および停止条件以外に、SCL が High のときに SDA のレベルが変化することはありません。開始条件と停止条件は常にマスタによって生成されます。開始条件が発生した後は、バスはビジー状態になります。停止条件が生成されると、その後、バスはフリー状態となります(*3)。

(*1)M16C/62 の簡易 I²C バスモードは、開始条件を検出する「スタートコンディション検出割込」をもちます。（3.3.2「スタートコンディション／ストップコンディション検出」参照）

(*2)M16C/62 の簡易 I²C バスモードは、停止条件を検出する「ストップコンディション検出割込」をもちます。（3.3.2「スタートコンディション／ストップコンディション検出」参照）

(*3)M16C/62 の簡易 I²C バスモードは、バスの状態を示す「バスビジーフラグ」を持ちます。（3.3.2「バスビジー検出」参照）

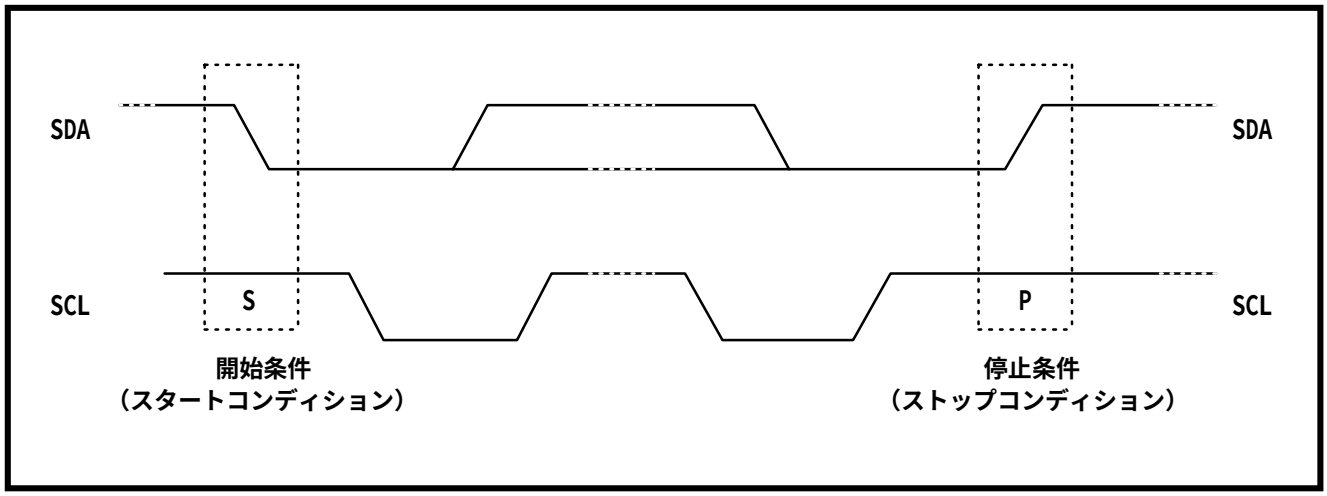


図 3.1.2　**開始条件と停止条件**

バイトのフォーマット

SDA に出力されるデータの各バイトの長さは、必ず 8 ビットとなります。1 回の転送（開始条件の生成時から停止条件の生成まで）で送出できるバイト数には制限がなく、何バイトでも連続して送ることができます。データは、最上位ビット(MSB)から順に送信され、各バイトの後（9 ビット目）には確認応答ビット（ACKnowledge bit ）が付加されます。

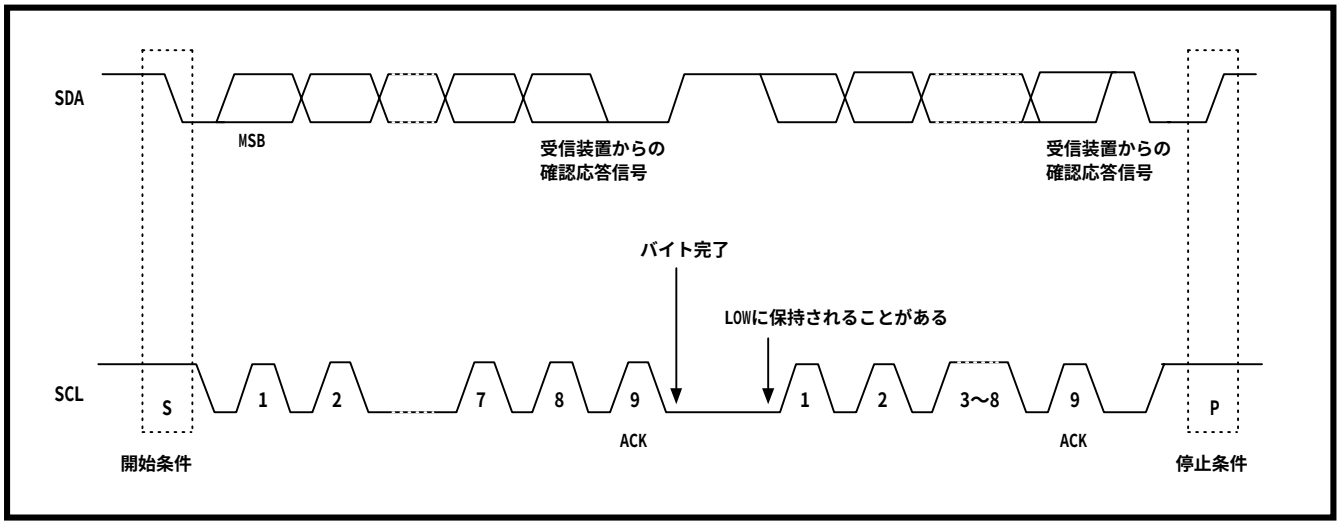


図 3.1.3　**バイトのフォーマット**

確認応答

データ転送では、確認応答が必要です。確認応答用のクロック信号は、マスタによって SCL 上に生成されます。確認応答用クロック信号（9 ビット目）が生成されると、送信装置側は SDA を解放します（High の状態）。受信装置側は、確認応答クロック信号の立ち上がり時に SDA を Low 状態で安定することによって、確認応答を生成します(*1)。

通常、アドレス指定された受信装置は、各バイトの完了を受信するたびに確認応答を生成しなければいけません。スレーブ受信装置がアドレス確認を行えない場合、そのスレーブは確認応答クロック信号の立ち上がり時 SDA を High の状態に保持して確認応答を生成しません(*2)。この時、マスタは停止条件を送出してデータ転送をやめることができます。またスレーブ受信装置がアドレスを確認した場合でも、転送途中でデータを受信できなくなった時には、確認応答を生成しないことによってそのことを示します。このとき、スレーブは SDA を High の状態に保持し、マスタが停止条件を生成します。

マスタが受信装置となる場合、スレーブから送信された最後のデータ・バイトに対して確認応答を生成しないことによって、マスタはスレーブ送信装置にデータの終わりを知らせます。このとき、スレーブ送信装置は SDA を解放し、マスタが停止条件を生成できるようにします。

(*1)M16C/62 の簡易 I²C バスモードは、確認応答の生成を検出する「アクノリッジ検出割込」をもちます。（3.3.3「アクノリッジ検出」参照）

(*2)M16C/62 の簡易 I²C バスモードは、確認応答が生成されていないことを検出する「アクノリッジ未検出割込」をもちます。（3.3.3「アクノリッジ未検出」参照）

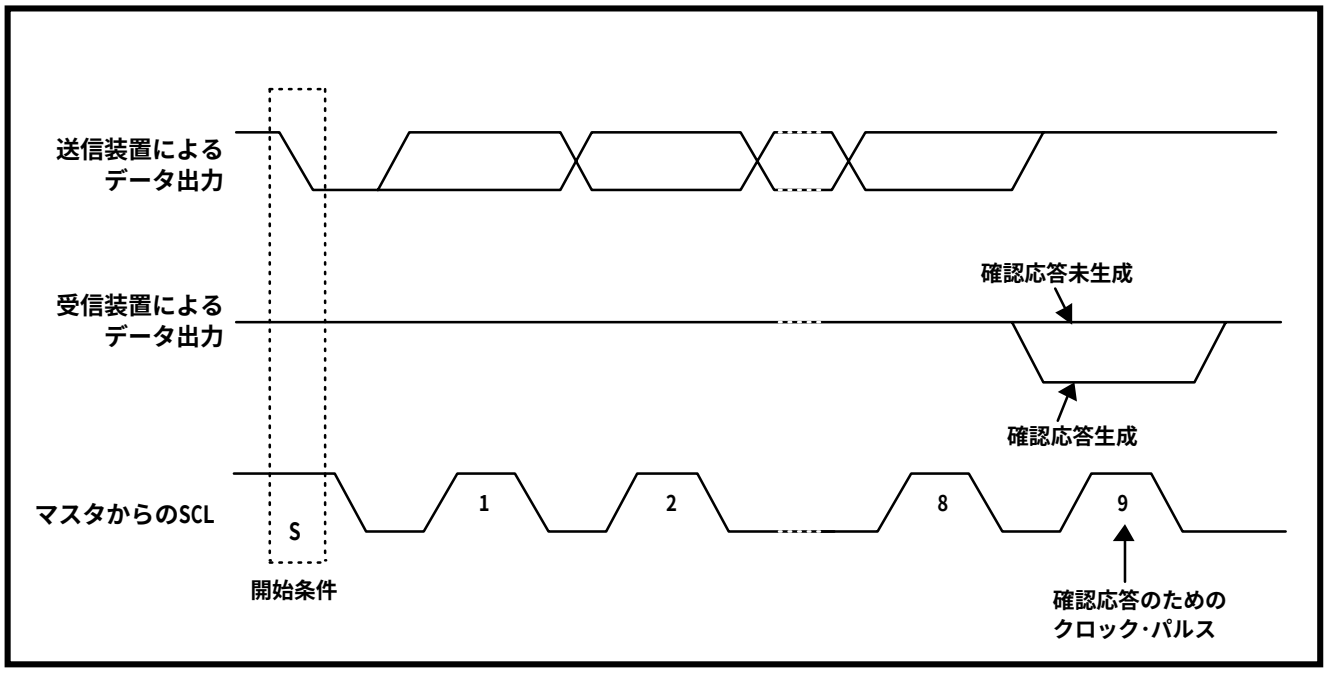


図 3.1.4　**確認応答**

3.1.4 通信調整

ひとつの I²C バス上に接続されている複数のデバイス間では、次のような通信調整手順に従って通信が行われます。

通信許可手順

マスタは、バスが解放状態の時のみデータ転送を開始することができます。しかし I²C はマルチ・マスタ・バスなので、複数のマスタが全く同時に「開始」条件を生成する場合があります。その時転送データの混乱を回避するために、通信許可手順が存在します。これは、SDA と各装置のオープンドレイン出力、またはオープンコレクタ出力端子との AND 特性を利用することによって行います。

SCL が High レベルにあるとき、SDA に High レベルを送信しているマスタが、（他のマスタが SDA に Low レベルを送信しているために）SDA のレベルが Low であることを検出した場合（アービトレーション・ロスト発生時）、通信不許可が決定したとしてデータの出力段をオフします(*1)。このようにして、複数のマスタがそれぞれ異なるデータを送信しようとする場合に、SDA 上で通信許可手順が行われ、データを送信できるマスタがひとつに限定されます。また、通信不許可になったマスタは、スレーブ機能を持っている場合はすぐにスレーブ受信モードに切り替わりますが、通信不許可が決定されたバイトの終わりまでクロック信号を生成することができます。

下図に、２つのマスタ間での通信許可手順を示します。DATA1 を生成するマスタの内部データ・レベルと SDA の実際のレベルが異なっている時、このマスタのデータ出力はオフになります。これにより通信許可を得たマスタによって開始されたデータ転送には、何の影響も与えません。これらの手続きは、多くのマスタが同時にスタートする時のみ使われるものであり、優先マスタや優先順位等を設定するものではありません。

(*1)M16C/62 の簡易 I²C バスモードは、SCL の立ち上がりのタイミングで内部データ・レベルと SDA のレベルの不一致を検出する機能 (3.3.5「アービトレーション・ロスト検出機能」 参照)、および出力段の出力をオフする機能 (3.3.5「アービトレーション・ロスト発生時の SDA 出力禁止機能」参照)を持ちます。

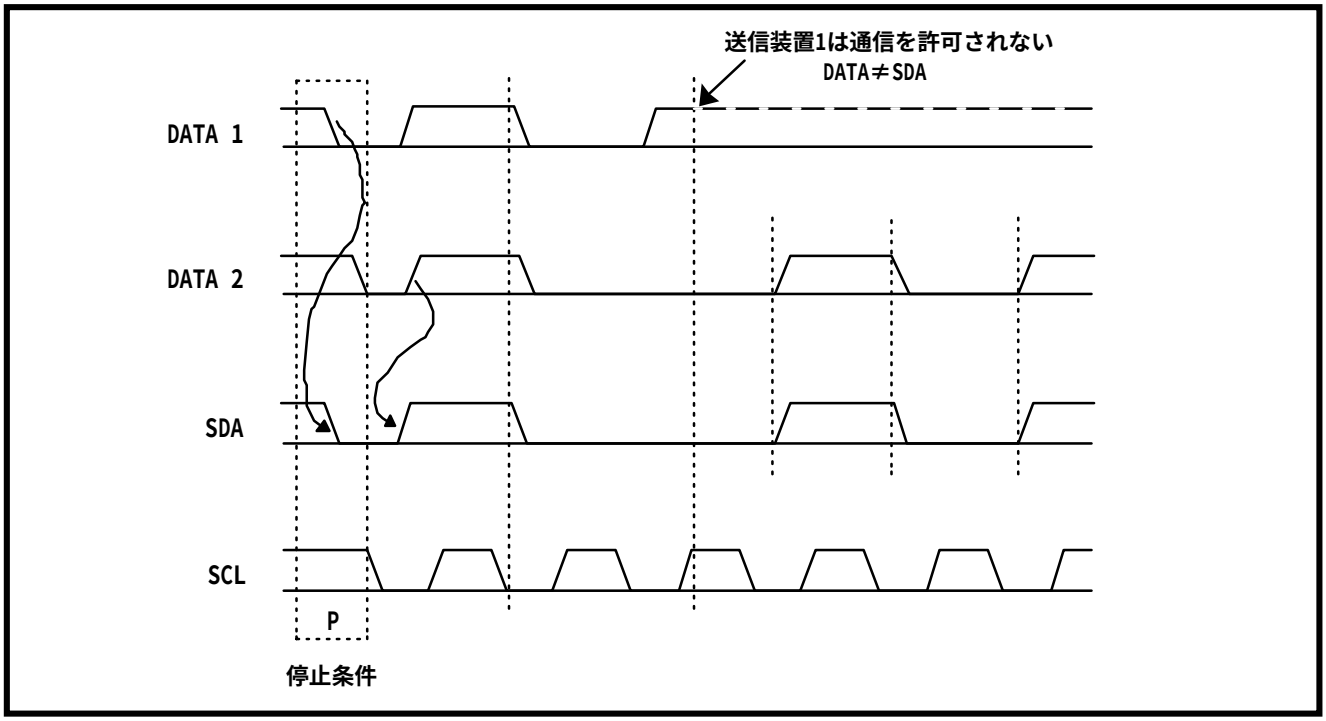


図 3.1.5 **通信許可手順**

SCL 同期

複数のマスタが同時にデータ転送を開始しようとした場合の通信許可手順は、SCL においても存在します。アービトレーション・ロスト発生時、通信不許可が決定されたマスタはそのバイトの終わりまでクロック信号を出力しますが、各マスタがクロック信号を SCL 上で独自に生成する中で、バス上には一つの限定されたクロック信号のみが発生することが必要です。

異なるクロック信号を同期させるには、SCL と各装置のオープンドレイン出力、またはオープンコレクタ出力端子との AND 特性を利用することによって実現できます。ある装置のクロック信号が Low になると、SCL を Low に保持したまま Low 期間のカウントを開始します。この装置の Low 期間が終了しクロック信号が Low から High に変化しても、他の装置のクロック信号がまだ Low 期間内にある場合、SCL は変化しません。従って SCL の Low 期間は、Low 期間の最も長い装置によって決定されることになります。この間、Low 期間の短い装置は、クロック信号を High にしたまま（ハイインピーダンス状態のまま）待ち状態となります。

全ての装置が Low 期間を終了すると、クロック信号は解放され、High 状態になります。これで、各装置の送出するクロック信号と SCL が同じ状態となり、各装置は High 期間のカウントを開始します。SCL は、High 期間を最初に終了した装置によって再び Low 状態にされます。

このように、Low 期間の最も長い装置によって Low 期間が、また High 期間の最も短い装置によって High 期間がそれぞれ決定され、SCL の同期がとられることになります。

(*)M16C/62 の簡易 I²C バスモードは、SCL の同期をとるための「SCL 同期化機能」をもちます。
（3.3.5「SCL 同期化機能」 参照）

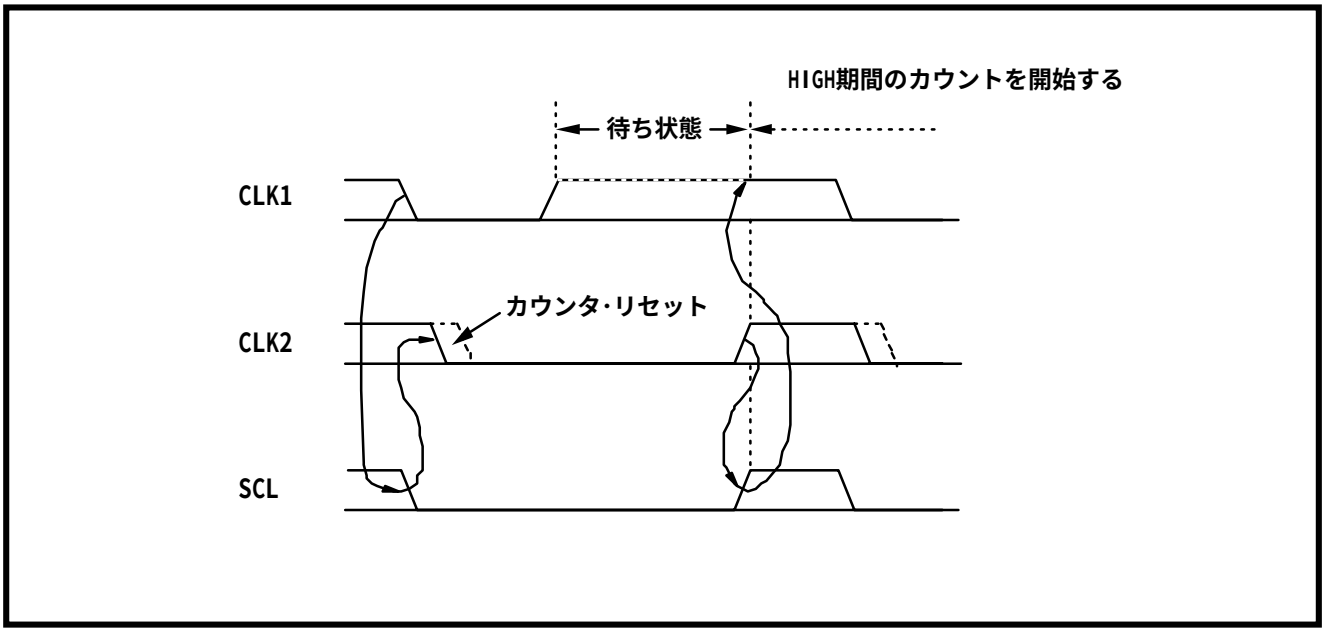


図 3.1.6 **SCL 同期**

SCL 同期を利用した協調手順

SCL 同期メカニズムは、複数のマスタが同時にスタートした時に使われるだけでなく、1 つのマスタが異なる処理スピードのデバイス間で伝送を行う場合にも利用されます。仮にマスタが、スレーブの準備が整う前に第 2 のバイトを続けようとした時、スレーブは準備が整うまで SCL を Low の状態に保持します。そのマスタは、自動的にクロックの High 期間の始めで待機状態に入り、マスタとスレーブは各バイト毎に同期をとります。またビット・レベルにおいても、各クロックの Low 期間の延長をすることにより、バス・クロックの速度を遅くすることができます。

このようにして、マスタの速度をスレーブの内部動作速度に適合させることが可能となります。

(*) M16C/62 の簡易 I²C バスモードは、スレーブ側からバイト単位、またはビット単位で SCL を Low に保持された場合でも、「SCL 同期化機能」によりスレーブ側の内部動作速度に適合できます。
（3.3.5「SCL 同期化機能」参照）

3.1.5 第１バイトの定義

I²C バスにおいて、開始条件後に送出される第一バイトは、スレーブを指定する重要なデータです。
次に、第一バイトの定義を示します。

７ビットアドレスのフォーマット

I²C バスにおいて、マスタに選択されたスレーブのアドレスは、通常、開始条件に続く最初のバイトで判断できます。ここでは、7 ビットアドレスのフォーマットについて説明します。

下図に、データ転送のフォーマットを示します。開始条件（S）の後、スレーブのアドレスが送信されます。このアドレスは 7 ビットから構成され、8 ビット目にはデータ方向ビット（R/W）が続きます。このデータ方向ビットが"0"ならスレーブに対しての送信（書き込み）、"1"ならスレーブに対してのデータ要求（読み込み）を示します。データ転送は、必ずマスタが生成する停止条件（P）によって終了します。しかし、マスタがまだバス上での通信を続けたい場合には、先に停止条件を生成することなく再送開始条件（Sr）を生成して、次の 1 バイトで別のアドレスを指定することができます。その場合も、再送開始条件に続く 7 ビットでスレーブのアドレスが送信され、8 ビット目にはデータ方向ビット（R/W）が続きます。

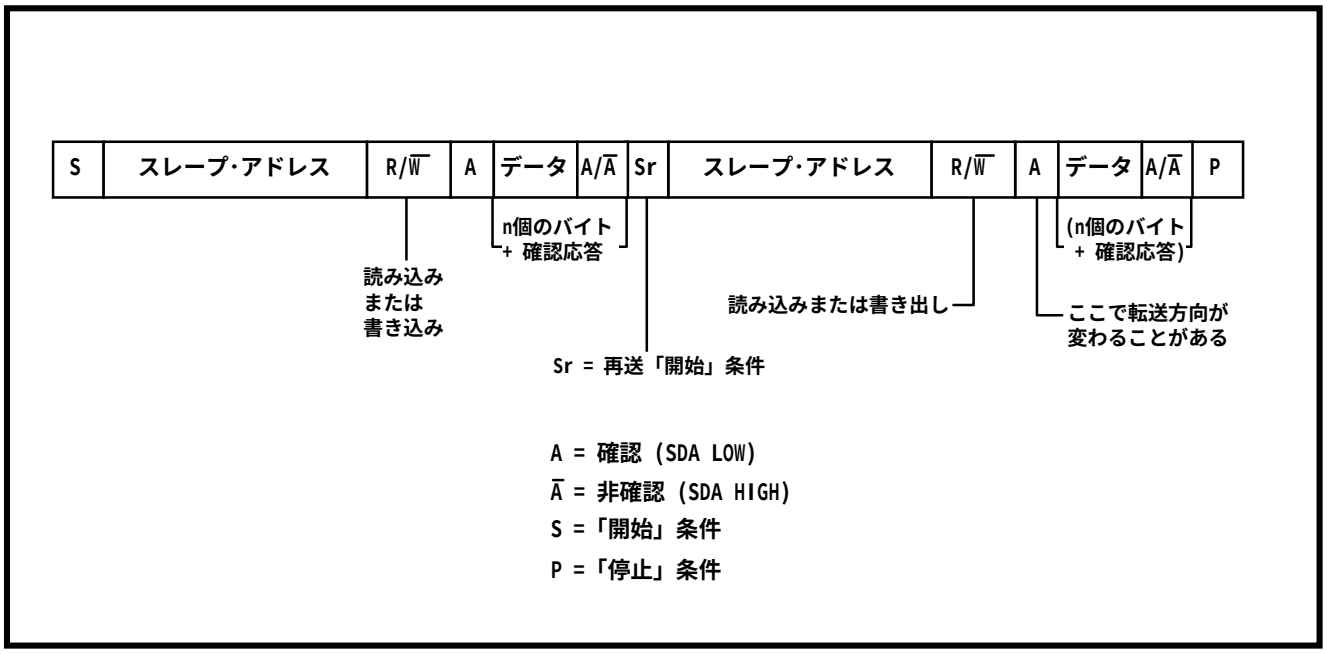


図 3.1.7 **７ビットアドレスのフォーマット**

第 1 バイトの各ビットの定義

開始条件後、第 1 バイトの最初の 7 ビットはスレーブ・アドレスを示します。8 番目のビットはデータ方向ビット（R/W）で、メッセージの方向を決定するためのビットです。アドレスが送信されると、システム内の各装置は開始条件後の最初の 7 ビットをそれぞれ各自のアドレスと比較します。アドレスが一致した場合、その装置は、スレーブとしてマスタからアドレス指定されたと判断し、9 ビット目のクロック・パルスのタイミングで SDA ラインを Low にしてアクノリッジを返し、データを受信（データ方向ビット"W"）または送信（データ方向ビット"R"）します。

(*) M16C/62 の簡易 I²C バスモードは、アドレス判断処理をしてアクノリッジを送信できるまでマスタを待機状態にするための機能を持ちます。（3.3.4「SCL 端子 L 出力機能」参照）

3.1.6 標準モードと高速モード

I²C バスの「標準モード」では、データ転送レートが最大 100k ビット／秒で、7 ビットアドレス指定が可能です。「標準モード」を拡張した「高速モード」では、データ転送レートが最大 400k ビット／秒で、7 ビットアドレス指定および 10 ビットアドレス指定が可能となります。I²C バス・インターフェースをもつ新しい装置には、全て高速モードが用意されています。

(*) M16C/62 の簡易 I²C バスモードにおいても、一部の制限事項を除いて高速モードはサポートされています（3.4「簡易 I²C バスモード注意事項」参照)。ただし、アドレスの送受信はソフトウエアで行います(3.3.4「自己アドレス判定の一例」参照)。

3.1.7 I²C バスの SDA および SCL バス・ラインの特性

最後に、I²C バスの標準モードおよび高速モード時の電気的特性および SDA・SCL バス・ラインの定義を示します。

表 3.1.1 **電気的特性**

パラメータ	記号	標準モード		高速モード		単位
		Min.	Max.	Min.	Max.	
LOWレベル入力電圧 ：	V_{IL}					V
入力レベルが一定の場合		−0.5	1.5	−0.5	1.5	
入力レベルがV_{DD}に応じて変化する場合		−0.5	$0.3V_{DD}$	−0.5	$0.3V_{DD}$	
HIGHレベル入力電圧 ：	V_{IH}					V
入力レベルが一定の場合		3.0	*1)	3.0	*1)	
入力レベルがV_{DD}に応じて変化する場合		$0.7V_{DD}$	*1)	$0.7V_{DD}$	*1)	
シュミット(Schmitt)トリガ入力のヒステリシス ：	V_{hys}					V
入力レベルが一定の場合		n/a	n/a	0.2	−	
入力レベルがV_{DD}に応じて変化する場合		n/a	n/a	$0.05V_{DD}$	−	
入力フィルタによって抑制されるスパイクのパルス幅	t_{SP}	n/a	n/a	0	50	ns
LOWレベル出力電圧（オープン･ドレインまたはオープン･コレクタ）：						V
シンク電流3mA時	V_{OL1}	0	0.4	0	0.4	
シンク電流6mA時	V_{OL2}	n/a	n/a	0	0.6	
バスのキャパシタンスが10pF～400pF(V_{OL2}の並列抵抗を通して最大6mAまで)の場合の$V_{IHmin.}$から$V_{ILmax.}$への出力立ち下がり時間 ：	t_{bF}					ns
V_{OL1}での最大シンク電流3mA		−	250 2)	20 + 0.1C_b 2)	250	
V_{OL2}での最大シンク電流6mA		n/a	n/a	20 + 0.1C_b 2)	250 3)	
入力電圧0.4V～0.9$V_{DDmax.}$時の各I/Oピンの入力電流	I_i	−10	10	−10 3)	10 3)	μA
各I/Oピンのキャパシタンス	C_i	−	10	−	10	pF

n/a = 該当せず

1) 最大$V_{IH} = V_{DD\ max.} + 0.5V$

2) C_b = 1つのバス･ラインのキャパシタンス(単位pF)。SDAおよびSCLバス･ラインの最大t_F(300ns)は出力段での最大t_{bF}(250ns)より大きくなります。
これによって、最大規定t_Fを超えることなくSDA/SCLピンとSDA/SCLバス･ラインの間に直列保護抵抗(Rs)を接続することが可能となります。

3) V_{DD}の供給が切れたときに、I/OピンがSDAおよびSCLラインを妨害しないようにする必要があります。

注)M16C/62 の電気的特性は I²C **バス規格とは異なります。（3.4.1「電気的特性」参照）**

タイミング定義

表 3.1.2 タイミング定義

パラメータ	記号	標準モードI²Cバス		高速モードI²Cバス		単位
		Min.	Max.	Min.	Max.	
SCLクロック周波数	f_{SCL}	0	100	0	400	KHz
「停止」条件と「開始」条件の間のバス・フリー・タイム	t_{BUF}	4.7	—	1.3	—	μs
ホールド・タイム(再送)「開始」条件。この期間の後、最初のクロック・パルスが生成されます。	$t_{HD:STA}$	4.0	—	0.6	—	μs
SCLクロックのLOW状態ホールド・タイム	t_{LOW}	4.7	—	1.3	—	μs
SCLクロックのHIGH状態ホールド・タイム	t_{HIGH}	4.0	—	0.6	—	μs
再送「開始」条件のセットアップ時間	$t_{SU:STA}$	4.7	—	0.6	—	μs
データ・ホールド・タイム ：	$t_{HD:DAT}$					
CBUS互換マスタの場合(9.1.3の「注意」参照)		5.0	—	—	—	μs
I²Cバスの場合		0 1)	—	0 1)	0.9 2)	μs
データセットアップ時間	$t_{SU:DAT}$	250	—	100 3)	—	ns
SDAおよびSCL信号の立ち上がり時間	t_R	—	1000	20 + 0.1C_b 4)	300	ns
SDAおよびSCL信号の立ち下がり時間	t_F	—	300	20 + 0.1C_b 4)	300	ns
「停止」条件のセットアップ時間	$t_{SU:STD}$	4.0	—	0.6	—	μs
各バス・ラインの容量性負荷	C_b	—	400	—	400	pF

上記の数値はすべて$V_{IH\ min.}$および$V_{IL\ max.}$レベルに対応した値です。

1) 装置は、SCLの立ち下がり端の未定義領域を埋めるために、
(SCL信号の$V_{IH\ min.}$での)SDA信号用に最低300nsのホールド時間を内部的に提供する必要があります。

2) 装置がSCL信号のLOWホールド・タイム(t_{LOW})を延長しない場合は、 最大データ・ホールド・タイム$t_{HD:DAT}$のみを満たすことが必要となります。

3) 高速モードI²Cバスは、標準モードI²Cバス・システム内で利用することが可能ですが、この場合には$t_{SU:DAT} \geqq 250ns$を満足することが条件となります。
装置がSCL信号のLOW状態ホールド・タイムを延長しない場合には、この条件を満たすことが無条件に求められます。
装置がSCL信号のLOW状態ホールド・タイムを延長する場合には、SCLラインが解放される$t_{R\ max.} + t_{SU:DAT} = 1000 + 250 = 1250ns$
(標準モードI²Cバスの仕様による)前に次のデータ・ビットをSDAラインに送出することが必要です。

4) C_b = 1つのバス・ラインの合計キャパシタンス(単位 pF)

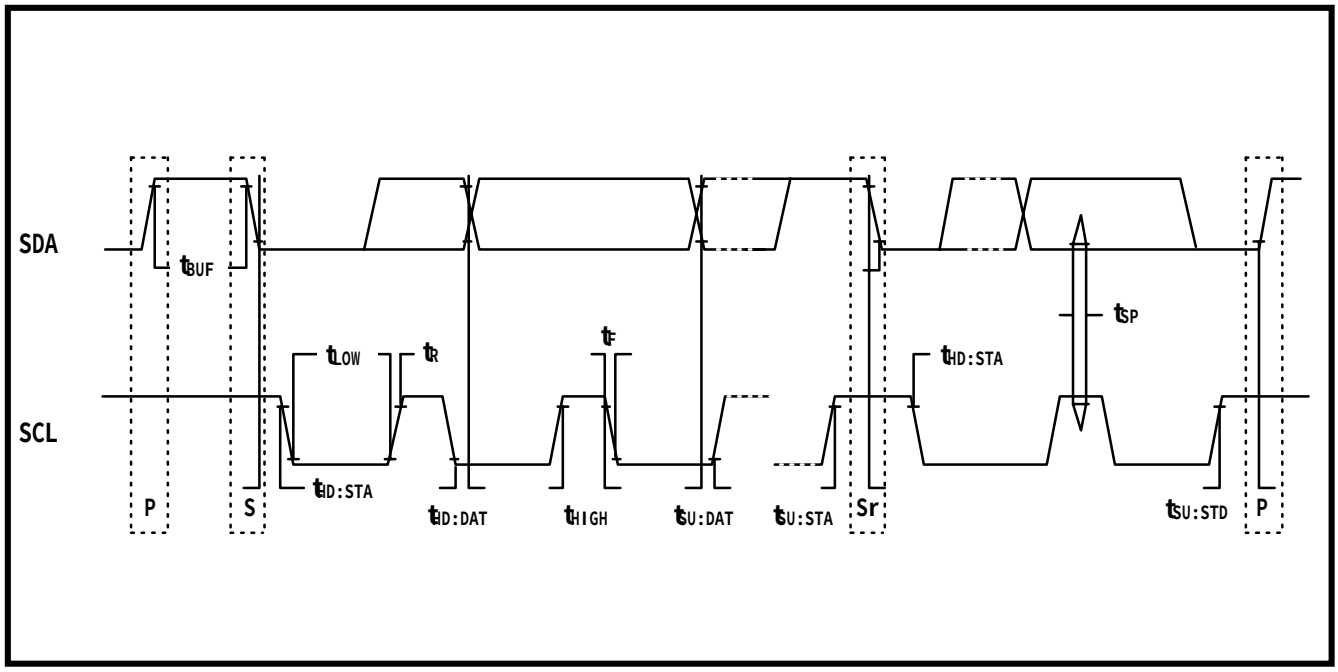


図 3.1.8　**タイミング定義**

(注)M16C/62 のタイミングは I²C バス規格とは異なります。（3.4.1「電気的特性」参照）

3.2 M16C/62 UART2 の機能

M16C/62 のシリアル I/O は、UART0、UART1、UART2、および SI/O3、SI/O4 の 5 チャネルで構成されています。これらはそれぞれ専用の転送クロック発生用タイマを持ち、独立して動作します。本章では、この中の UART2 について、また UART2 のもつ機能の一つである簡易 I²C バスモードの設定について詳しく説明します。

3.2.1 UART2 の有する機能

UART0～UART2 は、一部の機能が異なることを除いてほぼ同一の機能を持ちます。その中で特に UART2 は、SIM インターフェース、IE バスインターフェース、および I²C バスインターフェースに対応するモードが使用できます。

UART0～UART2 は、クロック同期形シリアル I/O モード、またはクロック非同期形シリアル I/O モードのいずれかを選択して使用します。

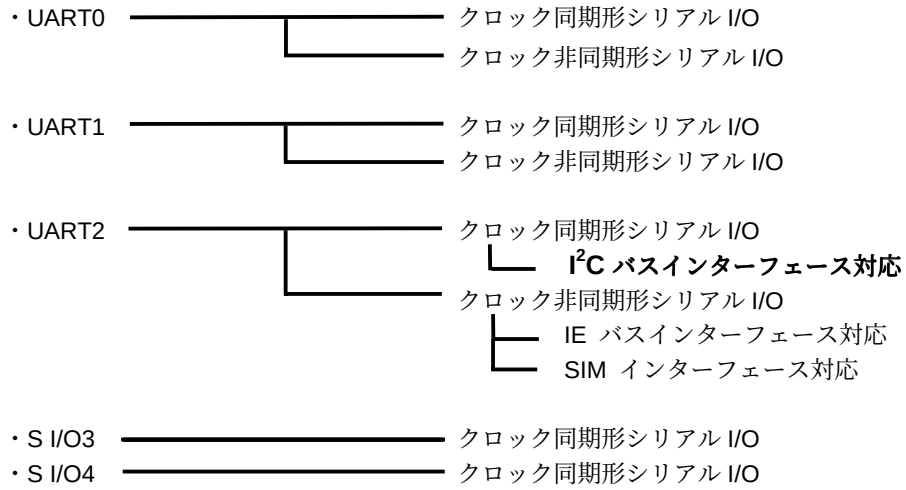
特に UART2 のクロック非同期形シリアル I/O モードでは、IE バスインターフェースおよび SIM インターフェース対応が可能です。M16C/62 では IE バスインターフェースおよび SIM インターフェース実現のために、シリアルデータ論理切り替え機能やバス衝突検出機能を持ちます。この機能の詳細は、M16C/62 のデータシートをご覧下さい。

また UART2 のクロック同期形シリアル I/O モードでは、I²C バスインターフェース対応が可能です。

本章では M16C/62 の簡易 I²C バスブロック図や各種関連レジスタを紹介し、次章では M16C/62 で I²C バスインターフェースを実現するための各種機能を説明します。

IE バスインターフェース、SIM インターフェースおよび I²C バスインターフェースは、UART2 に限定した機能となります。

M16C/62 シリアル I/O の構成



3.2.2 簡易 I²C バスモードブロック図

I²C バスインターフェースを実現するには、M16C/62 の簡易 I²C バスモードを使用します。簡易 I²C バスモードは、I²C モード選択ビット[IICM]を"1"に設定することで I²C バスインターフェースを実現するための回路を有効にします。以下に簡易 I²C バスモードのブロック図を示します。

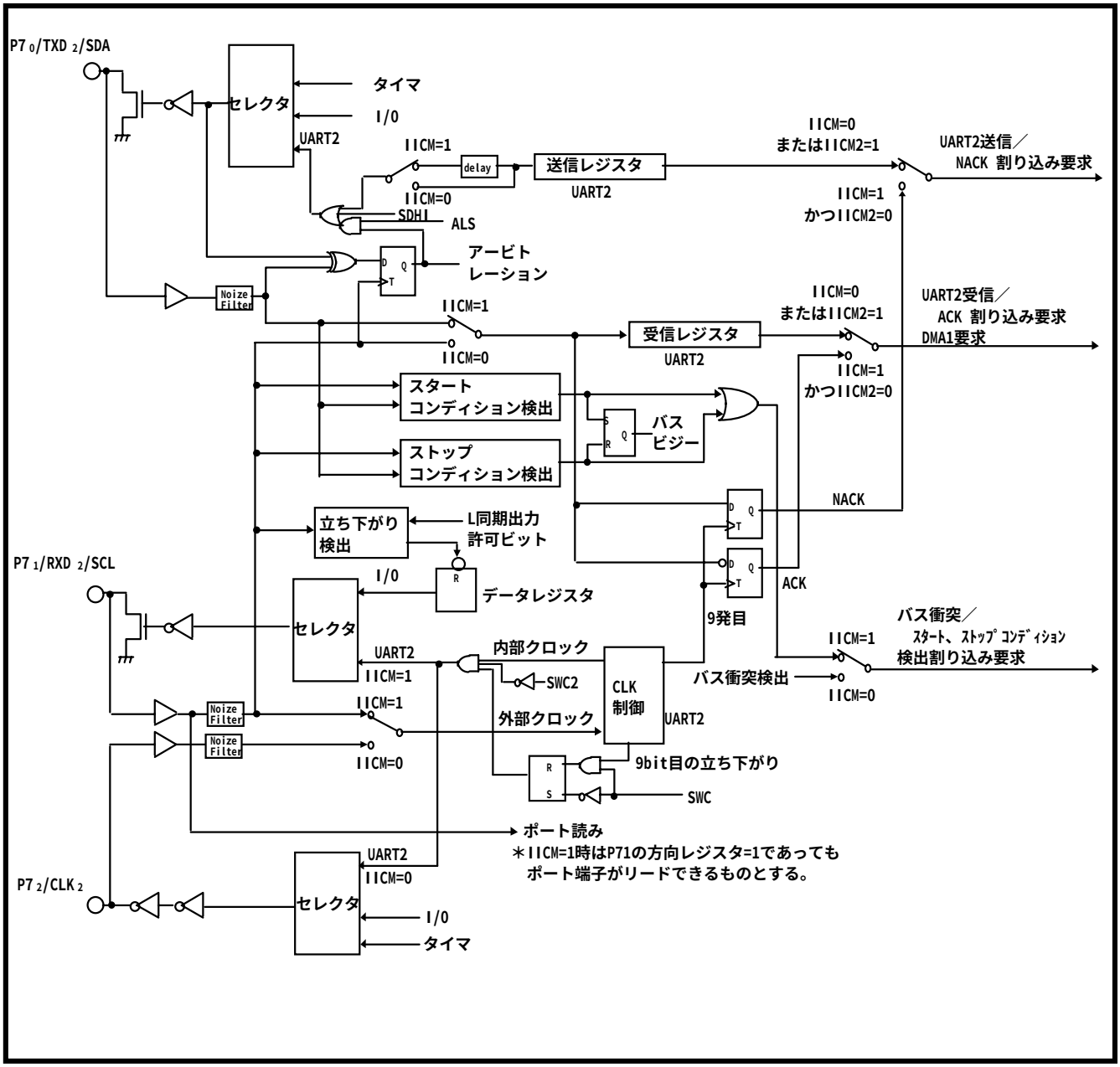


図 3.2.1 I²C **バスモードブロック図**

3.2.3 I²C バスモードで変化する端子の機能および割り込み要因

I²C モード選択ビット[IICM]の設定により、M16C/62 の簡易 I²C バスモードは有効になります。簡易 I²C バスモード選択時の各機能を示します。

表 3.2.1 **端子の機能**

	[IICM] =1(簡易I²Cバスモード)	[IICM] =0(通常SI/Oモード)
P7_0 端子機能	SDA （入出力）	TxD2 （出力）
P7_0 出力の初期値	シリアルI/O 無効時、P7_0 に設定した値	H レベル
P7_1 端子機能	SCL （入出力）	RxD2 （入力）
P7_1 端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ設定に従う
P7_2 端子機能	ポート P7_2	CLK2

表 3.2.2 **割り込み要因**

	[IICM] =1 (簡易I²Cバスモード)		[IICM] =0 (通常SI/O モード)
	[IICM2] =0	[IICM2] =1	
割込番号10 の要因	ｽﾀｰﾄ･ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ検出		バス衝突検出
割込番号15 の要因	ｱｸﾉﾘｯｼﾞ未検出	UART2 送信	UART2 送信
割込番号16 の要因	ｱｸﾉﾘｯｼﾞ検出	UART2 受信	UART2 受信
DMA 要求要因選択ビット="1101"時の DMA1 要因	ｱｸﾉﾘｯｼﾞ検出	UART2 受信	UART2 受信

ポートに対するビット処理命令の注意事項

入出力ポートのデータレジスタ（ポートラッチ）をビット処理命令を用いて書き替える場合、指定していないビットの値が変化することがあります。

（理由）ビット処理命令はリード・モディファイ・ライト形式の命令で、バイト単位で読み出し及び書き込みを行います。従って、入出力ポートのデータレジスタのあるビットに対してこの命令を実行した場合、そのデータレジスタの全ビットに対して以下の処理が行われます。

・入力に設定されているビット：端子の値が CPU に読み込まれ、ビット処理後、このビットに書き込まれる。
・出力に設定されているビット：データレジスタのビットの値が CPU に読み込まれ、ビット処理後、このビットに書き込まれる。

(*)ポート P7 に対するリード・モディファイ・ライト命令を行った場合、SCL、SDA の出力値を変化させてしまう場合がありますのでご注意ください。

3.2.4 簡易 I²C バスモード時のレジスタ設定

M16C/62 の UART2 を簡易 I²C バスモードで使用する場合の、関連するレジスタの構成を以下に示します。

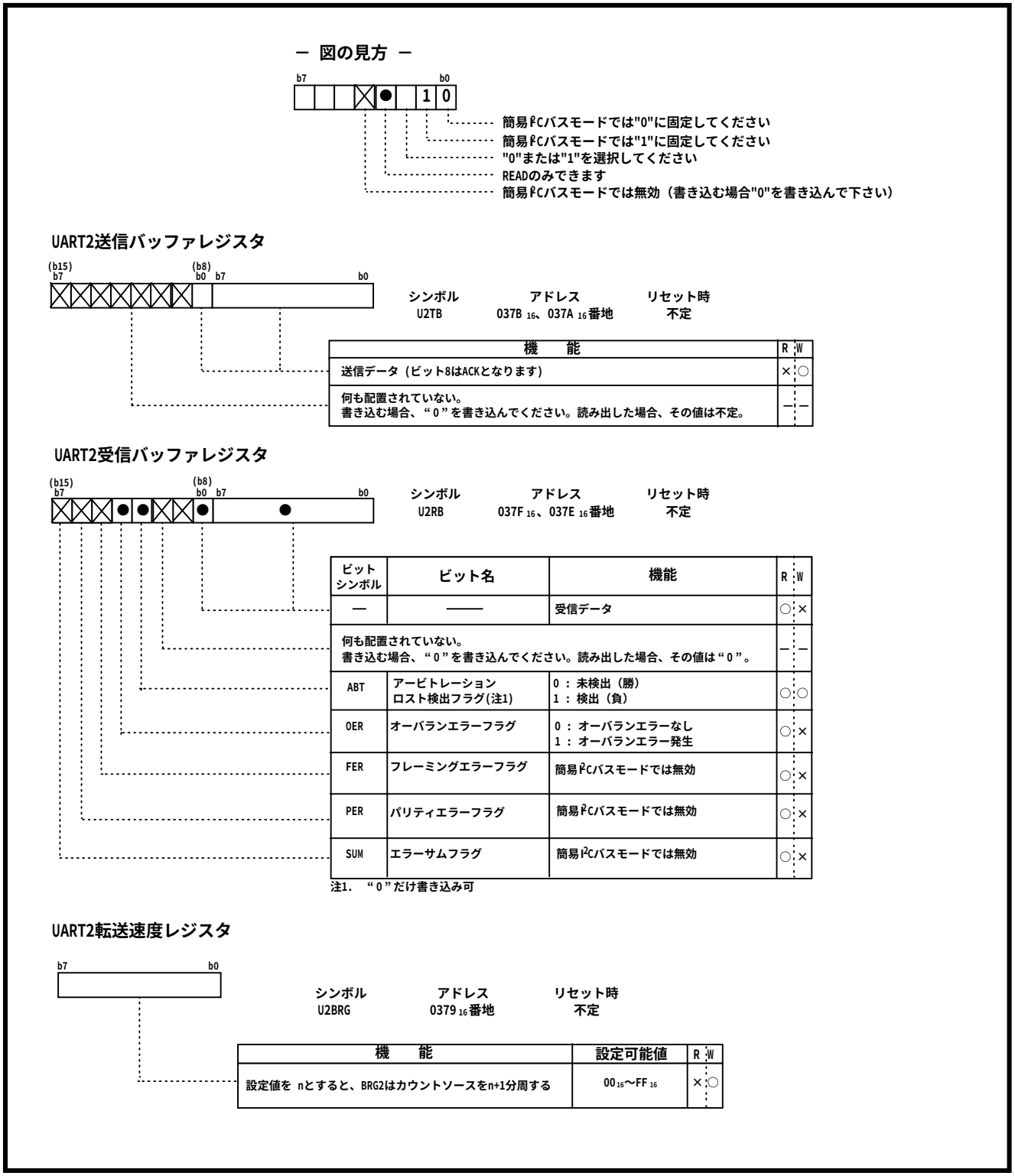


図 3.2.2 UART2 **関連レジスタ（1）**

UART2送受信モードレジスタ

b7	b6	b5	b4	b3	b2	b1	b0
0	X	X	X	X	0	1	0

シンボル U2MR　アドレス 0378 16番地　リセット時 00 16

ビットシンボル	ビット名	機能	R	W
SMD0	シリアルI/Oモード選択ビット	b2 b1 b0 0 0 0 : シリアルI/O無効（ポート制御） 0 1 0 : 簡易 I²C バスモード 簡易I²Cバス使用時は“010”に固定してください	○	○
SMD1			○	○
SMD2			○	○
CKDIR	内/外部クロック選択ビット	0 : 内部クロック（マスタ時"0"に設定） 1 : 外部クロック（スレーブ時"1"に設定）	○	○
STPS	ストップビット長選択ビット	簡易I²Cバスモードでは無効	○	○
PRY	パリティ奇/偶選択ビット	簡易I²Cバスモードでは無効	○	○
PRYE	パリティ許可ビット	簡易I²Cバスモードでは無効	○	○
IOPOL	TxD,RxD入出力極性切り替えビット	0 : 反転あり 1 : 反転なし 簡易I²Cバスモードでは“0”に固定してください	○	○

UART2 送受信制御レジスタ0

b7	b6	b5	b4	b3	b2	b1	b0
1	0	X	1	●	X		

シンボル U2C0　アドレス 037C 16番地　リセット時 08 16

ビットシンボル	ビット名	機能	R	W
CLK0	BRGカウントソース選択ビット	b1 b0 0 0 : f1を選択 0 1 : f8を選択 1 0 : f32を選択 1 1 : 使用禁止	○	○
CLK1			○	○
CRS	$\overline{CTS}/\overline{RTS}$機能選択ビット	ビット4が"1"のとき無効	○	○
TXEPT	送信レジスタ空フラグ	0 : 送信レジスタにデータあり（送信中） 1 : 送信レジスタにデータなし（送信完了）	○	×
CRD	$\overline{CTS}/\overline{RTS}$禁止ビット	0 : $\overline{CTS}/\overline{RTS}$機能許可 1 : $\overline{CTS}/\overline{RTS}$機能禁止 簡易I²Cバスモードでは"1"に固定してください	○	○
	何も配置されていない。 書き込む場合、“0”を書き込んでください。読み出した場合、その値は“0”。		—	—
CKPOL	CLK極性選択ビット	0 : 転送ｸﾛｯｸの立ち下がりで送信ﾃﾞｰﾀ出力、立ち上がりで受信ﾃﾞｰﾀ出力 1 : 転送ｸﾛｯｸの立ち上がりで送信ﾃﾞｰﾀ出力、立ち下がりで受信データ入力 簡易I²Cバスモードでは"0"に固定してください	○	○
UFORM	転送フォーマット選択ビット	0 : LSBファースト 1 : MSBファースト	○	○

図 3.2.3　UART2 **関連レジスタ（2）**

UART2送受信制御レジスタ1

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0		●		●	

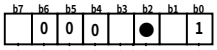
シンボル U2C1　　アドレス 037D 16番地　　リセット時 02 16

ビットシンボル	ビット名	機能	R	W
TE	送信許可ビット	0 : 送信禁止 1 : 送信許可	○	○
TI	送信バッファ空フラグ	0 : 送信バッファレジスタにデータあり 1 : 送信バッファレジスタにデータなし	○	×
RE	受信許可ビット	0 : 受信禁止 1 : 受信許可	○	○
RI	受信完了フラグ	0 : 受信バッファレジスタにデータなし 1 : 受信バッファレジスタにデータあり	○	×
U2IRS	UART2送信割り込み要因選択ビット（注1）	0 : 送信バッファ空（TI=1） 1 : 送信完了(TXEPT=1)	○	○
U2RRM	UART2連続受信モード許可ビット	0 : 連続受信モード禁止 1 : 連続受信モード許可 簡易IICバスモードでは“0”に固定してください	○	○
U2LCH	データ論理選択ビット	0 : 反転なし 1 : 反転あり 簡易IICバスモードでは“0”に固定してください	○	○
U2ERE	エラー信号出力許可ビット	簡易IICバスモードでは“0”に固定してください	○	○

注1. [IICM]=1 かつ [IICM2]=0 のときは [U2IRS] は 無効となります。

図 3.2.4　UART2 **関連レジスタ（3）**

UART2特殊モードレジスタ

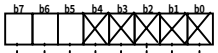


シンボル　U2SMR　　アドレス　0377 16番地　　リセット時　00 16

ビットシンボル	ビット名	機能	R	W	各ビットの説明章
IICM	IICモード選択ビット(注1)	0 : 通常モード 1 : I²Cモード	○	○	3.2.2章
ABC	アービトレーション ロスト検出フラグ制御	0 : ビット毎に更新 1 : バイト毎に更新	○	○	3.3.5章
BBS	バスビジーフラグ	0 : ストップコンディション検出 1 : スタートコンディション検出	○	○ (注1)	3.3.2章
LSYN	SCLL同期出力 許可ビット	0 : 禁止 1 : 許可	○	○	3.3.6章
ABSCS	バス衝突検出サンプリング クロック選択ビット	"0"に固定してください	○	○	—
ACSE	送信許可ビット自動クリア 機能選択ビット	"0"に固定してください	○	○	—
SSS	送信開始条件選択ビット	"0"に固定してください	○	○	—
SDDS	SDAディジタル遅延 選択ビット (注3,注4)	0：アナログディレイ出力選択 1：ディジタルディレイ出力選択 (I²Cモード時以外は"0"固定)	○	○	3.3.6章

注1.　I²Cモードの各機能を使用する時はシリアルI/Oモード選択ビット(SMD0～SMD2)を"010"にしてください。
注2.　"0"だけ書き込み可。
注3.　本機能はI²Cモード時以外は"1"を書き込まないでください。通常モード時は"0"に固定してください。本ビットが"0"の場合はUART2特殊モードレジスタ3(U2SMR3／0375 16番地)のビット7～ビット5(DL2～DL0＝SDAディジタル遅延値設定ビット)が初期化され"000"となり、アナログ遅延回路が選択されます。また、SDDS＝"0"の場合にはU2SMR3の読み出し、書き込みはできません。
注4.　アナログ遅延選択時はアナログ遅延値のみ、ディジタル遅延選択時はディジタル遅延値のみの遅延となります。

UART2特殊モードレジスタ3



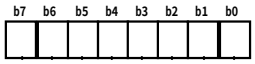
シンボル　U2SMR3　　アドレス　0375 16番地　　リセット時　不定
(ただし、SDDS= "1"の場合の初期値"00

ビットシンボル	ビット名	機能	R	W	各ビットの説明章
		何も配置されていない。 書き込む場合、"0"を書き込んでください。読み出した場合、その値は不定。 ただし、SDDS＝1とした場合は"0"が読み出される(注1)。	—	—	—
DL0	SDAディジタル 遅延値設定ビット (注1、注2、注3、注4)	b7 b6 b5 0 0 0：アナログ遅延選択 0 0 1：1/f(XN)の2サイクル(ディジタル遅延) 0 1 0：1/f(XN)の3サイクル(ディジタル遅延) 0 1 1：1/f(XN)の4サイクル(ディジタル遅延) 1 0 0：1/f(XN)の5サイクル(ディジタル遅延) 1 0 1：1/f(XN)の6サイクル(ディジタル遅延) 1 1 0：1/f(XN)の7サイクル(ディジタル遅延) 1 1 1：1/f(XN)の8サイクル(ディジタル遅延)	○	○	3.3.6章
DL1			○	○	
DL2			○	○	

注1.　本ビットはUART2特殊モードレジスタ（U2SMR／0377 16番地）のビット7(SDDS＝SDAディジタル遅延選択ビット)が"1"のときのみ読み出し、書き込み可能です。SDDS＝"1"にしてUART2特殊モードレジスタ3(U2SMR3)の初期値を読み出した場合、その値は"00 16"です。
SDDS＝"1"にしてUART2特殊モードレジスタ3(U2SMR3)に書き込む場合、ビット0～ビット4には"0"を書き込んでください。
SDDS＝"0"のときは書き込み不可、読み出した場合、その値は不定です。
注2.　本ビットはSDDS＝"0"のときには初期化され"000"となり、アナログ遅延回路が選択されます。本ビットはリセット解除時は"000"となり、アナログ遅延回路が選択されます。ただし、SDDS＝"1"とした場合のみ読み出し可能のため、SDDS＝"0"の場合、読み出した値は不定です。
注3.　アナログ遅延選択時はアナログ遅延値のみ、ディジタル遅延選択時はディジタル遅延値のみの遅延となります。
注4.　遅延量はSCL端子、SDA端子の負荷により変化します。また、外部クロックを使用した場合には、100ns程度、遅延が大きくなりますので、評価の上、使用してください。

図 3.2.5　UART2 **関連レジスタ（4）**

UART2特殊モードレジスタ2



シンボル　U2SMR2　　アドレス　0376 16番地　　リセット時　00 16

ビットシンボル	ビット名	機能	R	W	各ビットの説明章
IICM2	I²Cモード選択ビット2	表1参照	○	○	3.2.3章
CSC	クロック同期化ビット	0 : 禁止 1 : 許可	○	○	3.3.5章
SWC	SCLウエイト出力ビット	0 : 禁止 1 : 許可	○	○	3.3.4章
ALS	SDA出力停止ビット	0 : 禁止 1 : 許可	○	○	3.3.5章
STAC	UART2初期化ビット	0 : 禁止 1 : 許可	○	○	3.3.6章
SWC2	SCLウエイト出力ビット2	0 : UART2クロック 1 : 0出力	○	○	3.3.2章
SDHI	SDA出力禁止ビット	0 : 許可 1 : 禁止（ハイインピーダンス）	○	○	3.3.6章
SHTC	スタート/ストップコンディション条件制御ビット	1 : I²Cモード時、必ず“1”に設定してください。（表2参照）	○	○	—

表1　IICM=1

	機能	IICM2 = 0	IICM2 = 1
1	割り込み番号15の要因	アクノリッジ未検出（NACK）	UART2送信　（送信クロックの最終ビットの立ち上がり）
2	割り込み番号16の要因	アクノリッジ検出（ACK）	UART2受信　（受信クロックの最終ビットの立ち下がり）
3	DMA要求要因選択ビット= "1101"時のDMA1要因	アクノリッジ検出（ACK）	UART2受信　（受信クロックの最終ビットの立ち下がり）
4	UART2受信シフトレジスタから受信バッファへのデータ転送タイミング	受信クロックの最終ビットの立ち上がり	受信クロックの最終ビットの立ち下がり
5	UART2受信／アクノリッジ検出割り込み要求発生タイミング	受信クロックの最終ビットの立ち上がり（アクノリッジ検出）	受信クロックの最終ビットの立ち下がり（UART2受信）

表2

スタート／ストップコンディション検出タイミング特性（注1）

3～6サイクル ＜ セットアップ時間　（注2）
3～6サイクル ＜ ホールド時間　（注2）

（注1）　スタート／ストップコンディション条件制御ビット SHTC=1 の場合。
（注2）　サイクル数はメインクロック入力発振周波数 f(XIN)のサイクル数を示します。

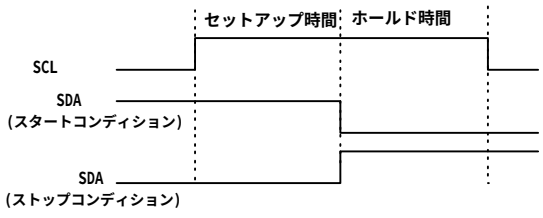


図 3.2.6　UART2 **関連レジスタ（5）**

3.3 簡易 I²C バスモードの各機能

本章では、I²C バスインターフェースを実現するために M16C/62 を簡易 I²C バスモードで使用する場合の、各ハードウエア機能の使用方法について説明します。

3.3.1 バイトデータの送信／受信の方法

M16C/62 をマスタとして使用し簡易 I²C バスモードの SCL を送出する設定方法、また 1 バイトのデータを送信、または受信する場合の設定方法を説明します。

SCL 生成方法（マスタ時）

M16C/62 をマスタとして使用する場合、送出クロック(SCL)の速度を設定する必要があります。それらは通常のシリアル I/O 送信と同様に、以下のレジスタで設定します。SCL は、送信バッファにデータを書き込み後、SCL の 1.5 サイクル以内に送出されます。

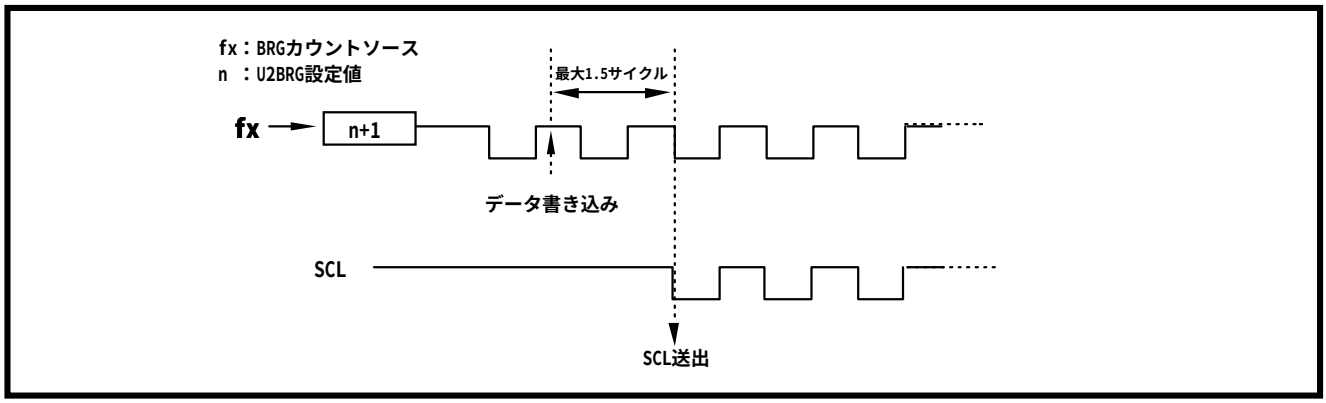


図 3.3.1　SCL **送出タイミング**

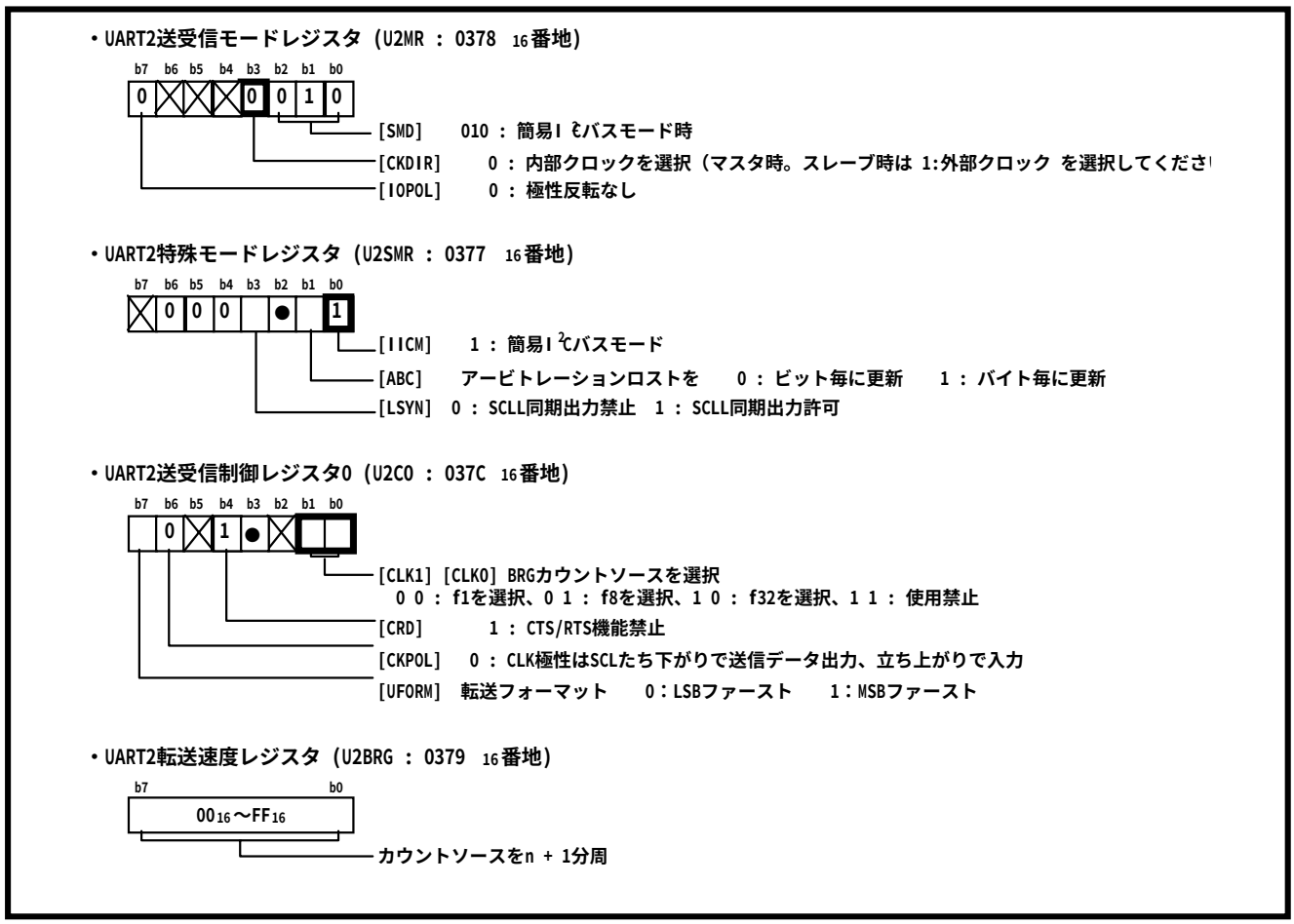


図 3.3.2　**関連レジスタ**

【設定例】
源発振 10MHz で使用している場合、送信速度を 100kbps に設定する場合は、
- U2MR =00000010b （IIC モード、内部クロック選択）
- U2C0 =10010000b （BRG カウントソースにｆ１を選択）
- U2BRG =49

【スレーブ時の設定】
スレーブとして使用する際には、UART2 送受信モードレジスタ(U2MR)のビット 3 [CKDIR]を"1"に設定し、外部クロックを選択してください。その時、BRG カウントソース選択ビット[CLK0] [CLK1] および UART2 転送速度レジスタ(U2BRG)の設定は無効となります。

バイトデータの送信方法

M16C/62 を送信装置として使用する場合、SDA 端子から 8 ビットの送信データを送出しますが、送信クロックの 9 ビット目ではアクノリッジを受信するために M16C/62 の SDA 端子を解放する必要があります。これは、送信バッファに設定するデータで操作できます。

送信バッファには、9 ビットのデータを設定します。I²C バスでは、MSB ファーストでデータを送信します。M16C/62 では、転送フォーマットを MSB ファーストにして 9 ビットで設定したデータは b7→ b6 → ･･･→ b0 → b8 の順で送出されます。従って最上位ビットの送出時がアクノリッジを受信するタイミングとなるので、このときに SDA 端子を解放するためには最上位ビットに"1"を設定して M16C/62 の SDA をハイインピーダンス状態にします。この様に、バイトデータの送信を行います。

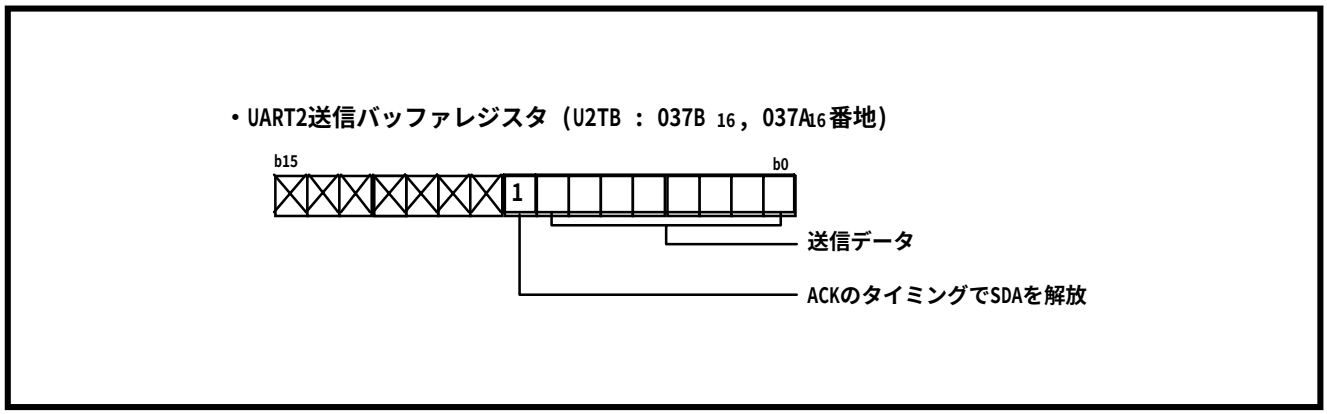


図 3.3.3　**関連レジスタ**

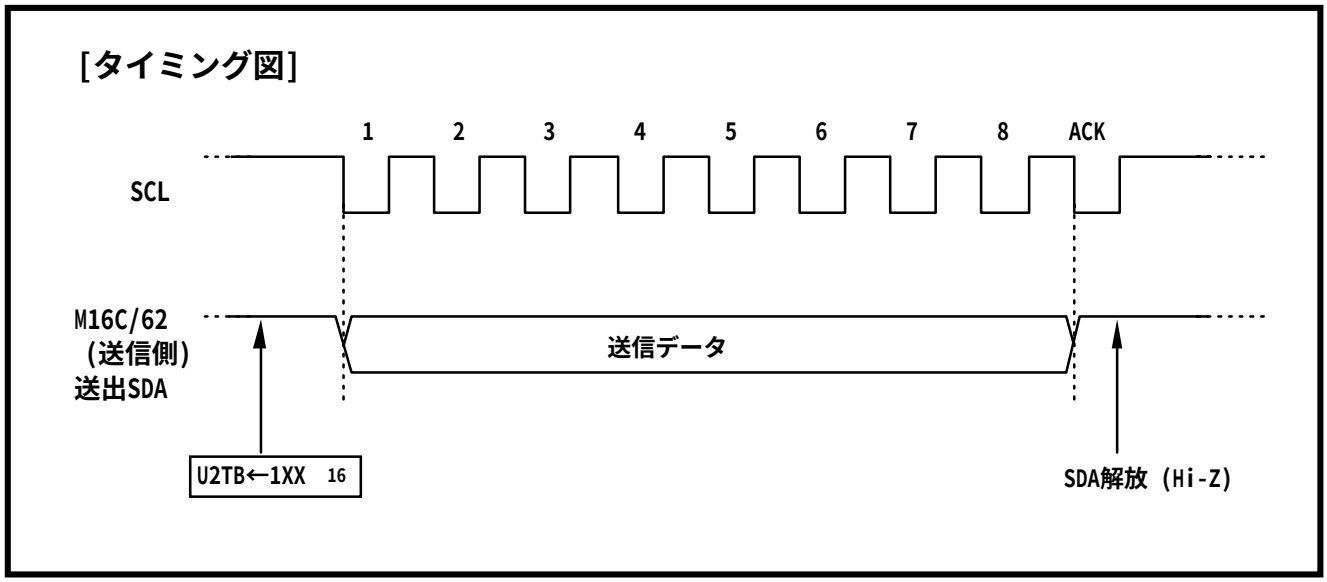


図 3.3.4　**タイミング図**

バイトデータの受信方法

M16C/62 を受信装置として使用する場合､SDA 端子から 8 ビットのデータを受信する間は M16C/62 の SDA 端子を解放する必要があります。またクロックの 9 ビット目では、SDA 端子を"L"にしてアクノリッジを生成する必要があります。これは、すでにマスタ側が指定した受信装置のアドレス判定が終了し、自装置に対して送信が行われていることが確定している場合には、送信バッファに書き込む値でアクノリッジの送出を簡単に操作できます。

M16C/62 ではデータ受信時も、送信バッファに 9 ビットのデータをダミーデータとして設定します。下位 8 ビットを送出している間 SDA 端子を解放するためには、下位 8 ビットに"1"を設定します。また、アクノリッジ生成のためには､最後に送出するビット(b8)に"0"を設定します｡この様に､バイトデータの受信を行います。

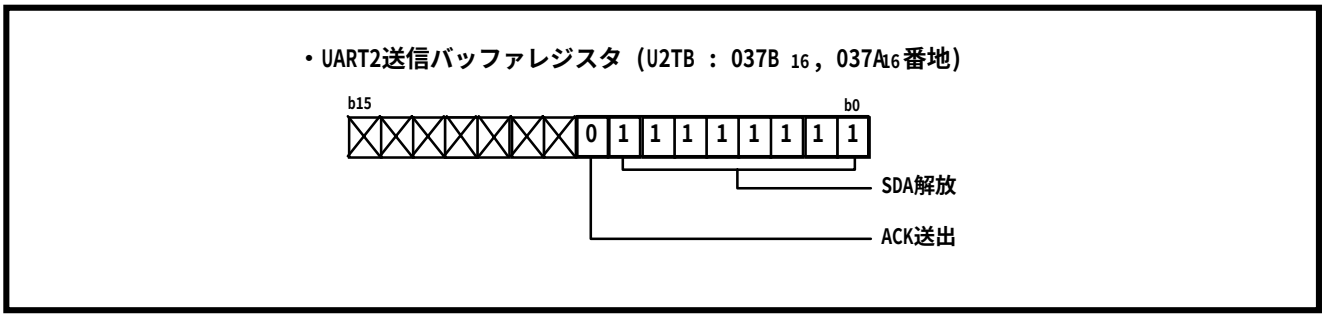


図 3.3.5　**関連レジスタ**

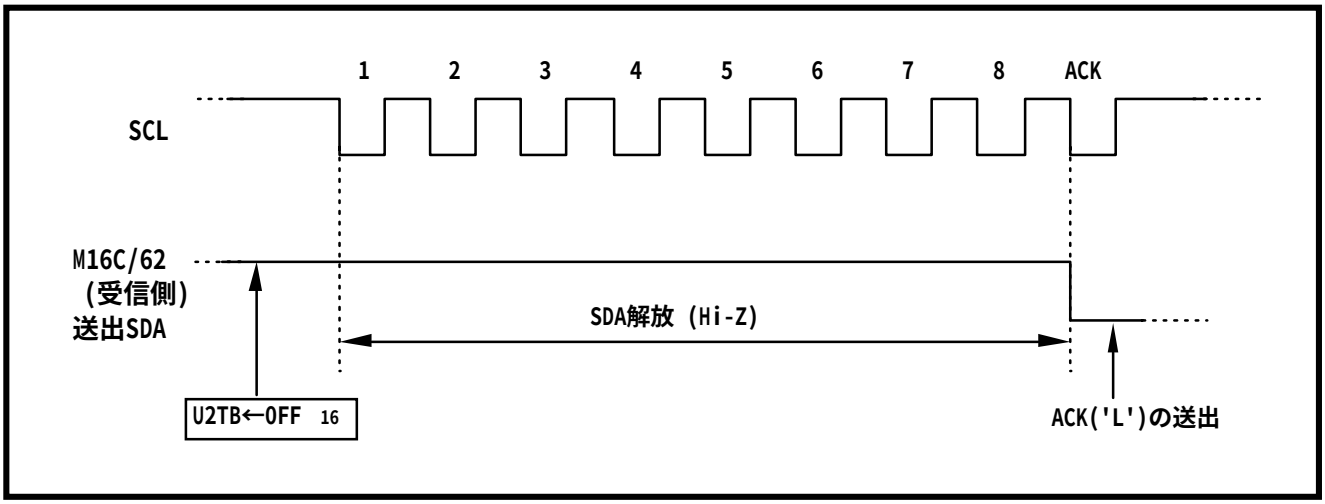


図 3.3.6　**タイミング図**

送信割り込み／受信割り込み

M16C/62 を送信装置として使用する場合、データ送出の完了は「UART2 送信割込」で検出できます。また、M16C/62 を受信装置として使用する場合、データ受信の完了は「UART2 受信割込」で検出できます。これらの割込は、割込番号 15 および割込番号 16 のレジスタに割り当てられており、I²C モード選択ビット 2[IICM2] を "1"に設定することで割込要因がそれぞれ UART2 送信、および UART2 受信になります。この場合の送信割込発生タイミングは、UART2 送信割り込み要因選択ビット[U2IRS]が"0"のときは送信クロックの開始ビットの立ち下がりとなり、[U2IRS]が"1"のときは最終ビットの立ち上がりとなります。また受信割込発生タイミングは、受信クロックの最終ビットの立ち下がりとなります。(3.2.4 章「簡易 I²C バスモード時のレジスタ設定」UART2 関連レジスタ(4)表 1 参照)

なお、受信クロックの最終ビットの立ち上がり前（簡易 I²C バスモードの受信完了割込中等）に受信バッファを読み出すと、受信データはビットの位置が変化した状態で読み出されますのでご注意ください。

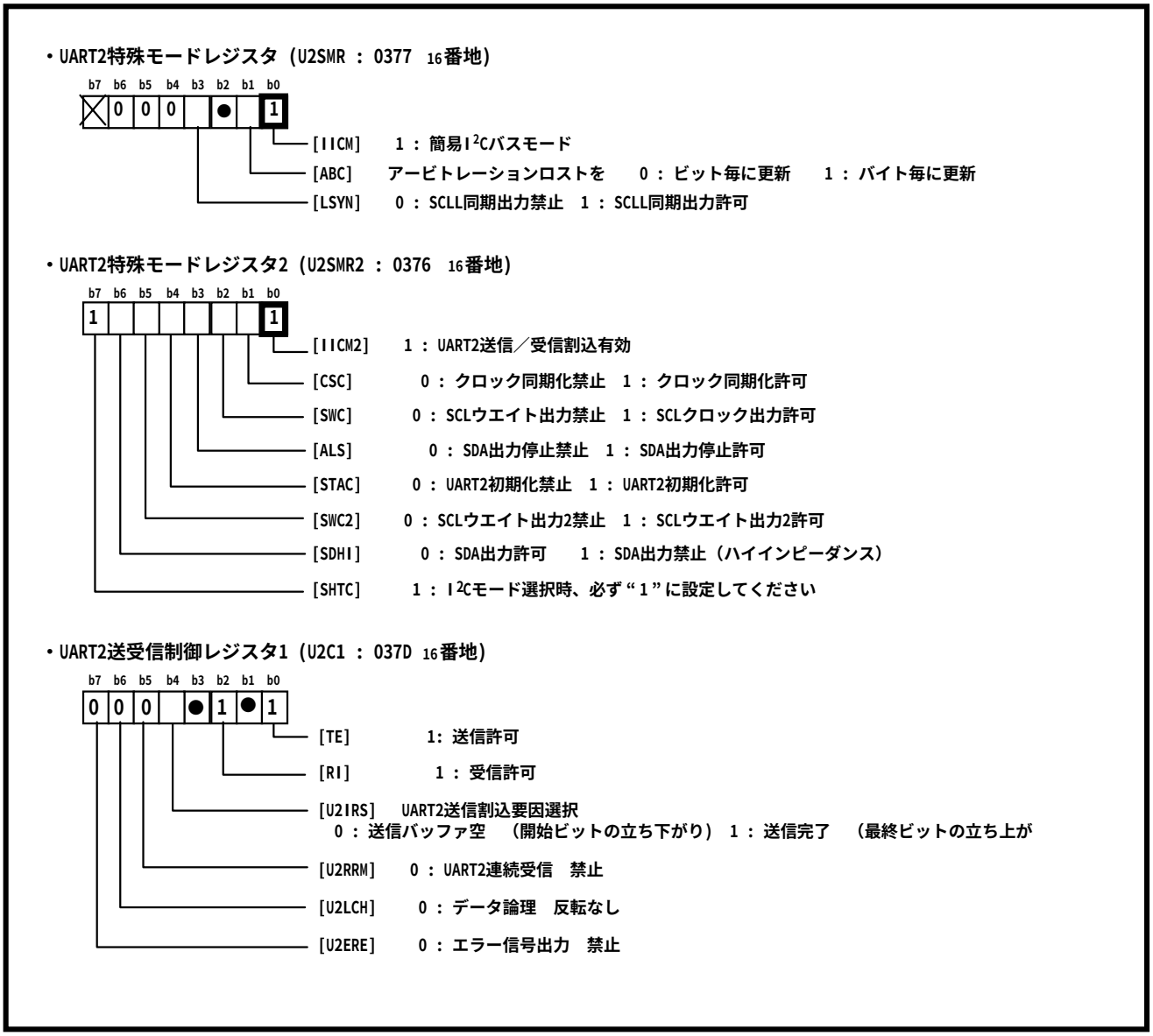
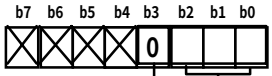


図 3.3.7 **関連レジスタ**

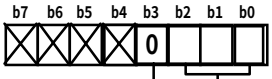
送信割り込み／受信割り込み（つづき）

・UART2送信割り込み制御レジスタ0（S2TIC ： 004F 16番地）－送信割込使用時－



[ILVL] 割り込み優先レベル
1～7 ： 割り込みレベル設定
（割り込みを使用しないときは"0"(禁止)にしてください）
[IR]
0 ： 割り込み要求発生時"1"にセットされます

・UART2受信割り込み制御レジスタ0（S2RIC ： 0050 16番地）－受信割込使用時－



[ILVL] 割り込み優先レベル
1～7 ： 割り込みレベル設定
（割り込みを使用しないときは"0"(禁止)にしてください）
[IR]
0 ： 割り込み要求発生時"1"にセットされます

・Iフラグ = 1 (割り込み要求により割り込みを発生させる場合)

図 3.3.8　**関連レジスタ**

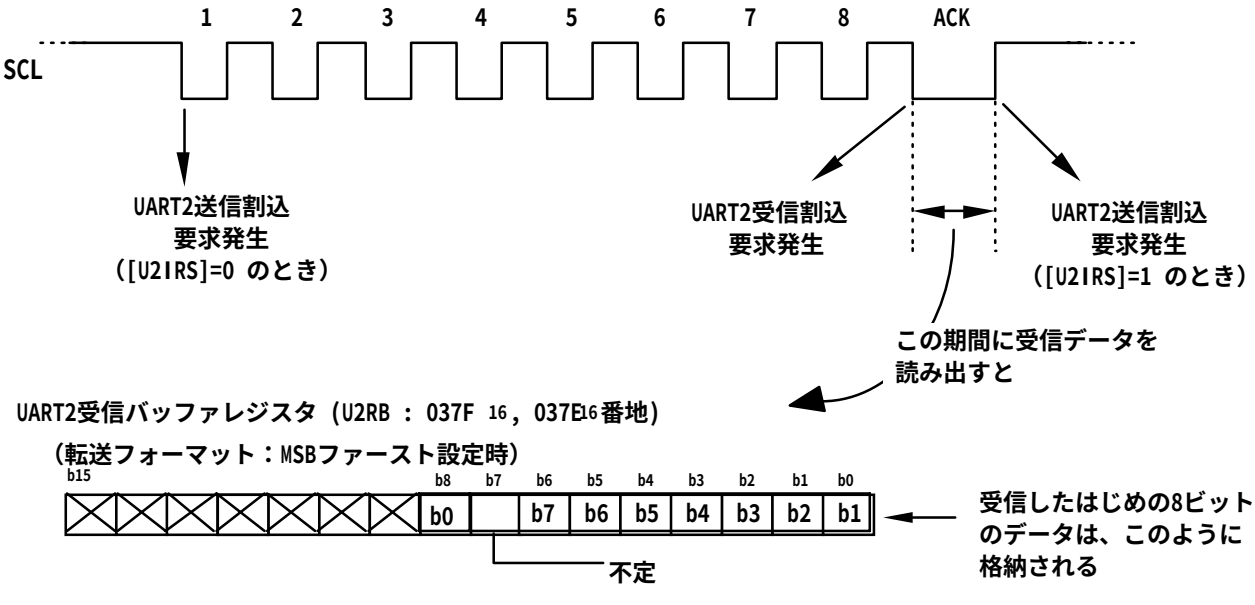


図 3.3.9　**タイミング図**

3.3.2 スタートコンディション／ストップコンディション

データ送受信の開始、およびデータ送受信の終了時には、開始条件（スタートコンディション）、および停止条件（ストップコンディション）が発生します。

M16C/62 をスレーブとして使用する場合、マスタの生成するスタートコンディションおよびストップコンディションを検出するために、「スタートコンディション／ストップコンディション検出割込」機能をハードウエアで備えています。

M16C/62 をマスタとする場合は、スタートコンディションおよびストップコンディションをソフトウエアで生成し送出します。スタートコンディション作成時にバスの使用状態を検知するための機能として、「バスビジー検出」機能を、またスタートコンディション送出後に他デバイスからクロック送出を禁止するための機能として、SCL から強制的に"L"を出力する「SCL 端子 L 出力機能 2」機能をハードウエアで備えています。

スタートコンディション／ストップコンデション検出

SCL が High のときに SDA が High から Low に変化するスタートコンディション、および SCL が High のときに SDA が Low から High に変化するストップコンディションは、M16C/62 では「スタートコンディション／ストップコンディション検出割込」で検出することができます。本割込は「ソフトウエア割込番号 10」に割り当てられ、I²C モード選択時（[IICM]="1"）は割り込み番号 10 の割込要因が「スタートコンディション／ストップコンディション検出割込」に変化します。この割込を検出した場合は、バスビジーフラグ(BBS)の状態でスタートコンディションまたはストップコンディションのどちらが発生したのかを判断してください。

なお、スタートコンディションおよびストップコンディション検出のセットアップタイム、ホールドタイムは、I²C バス規格と異なる場合がありますのでご注意ください（3.4.1「スタート／ストップコンディションのセットアップタイム・ホールドタイム」参照）。

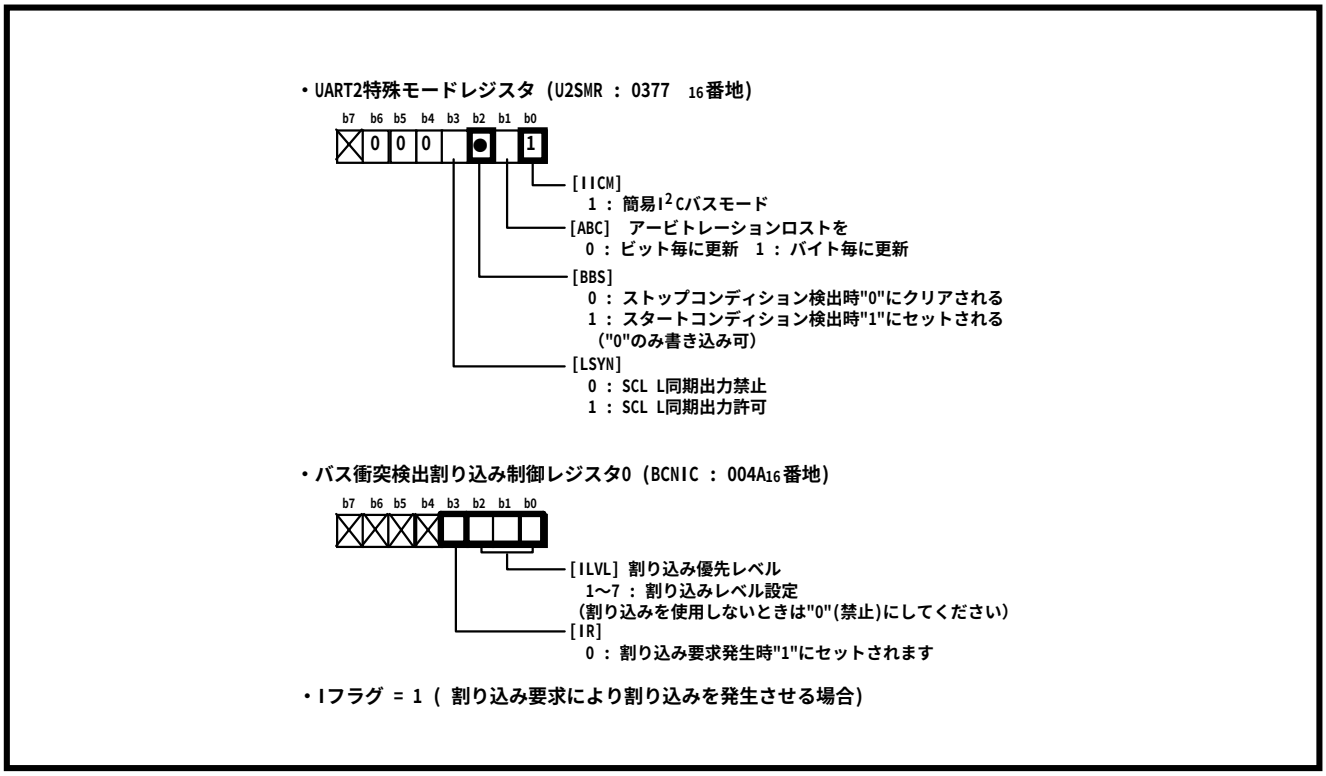


図 3.3.10　**関連レジスタ**

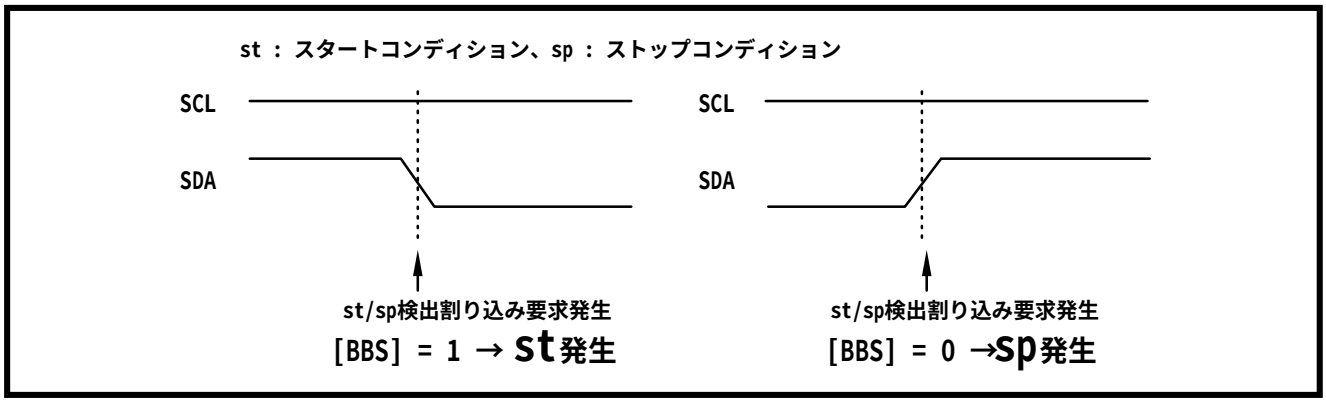


図 3.3.11　**タイミング図**

スタートコンディション送出

M16C/62 をマスタとして使用する場合のスタートコンディションの生成は、ポート制御で行います。M16C/62 の簡易 I²C バスモードでの SDA 送信出力の初期値は、シリアル I/O 制御禁止時（ポート制御時）にポート P7_0 に設定した値となりますので、この機能を使用します。
スタートコンディションを生成する制御の一例をご紹介します。

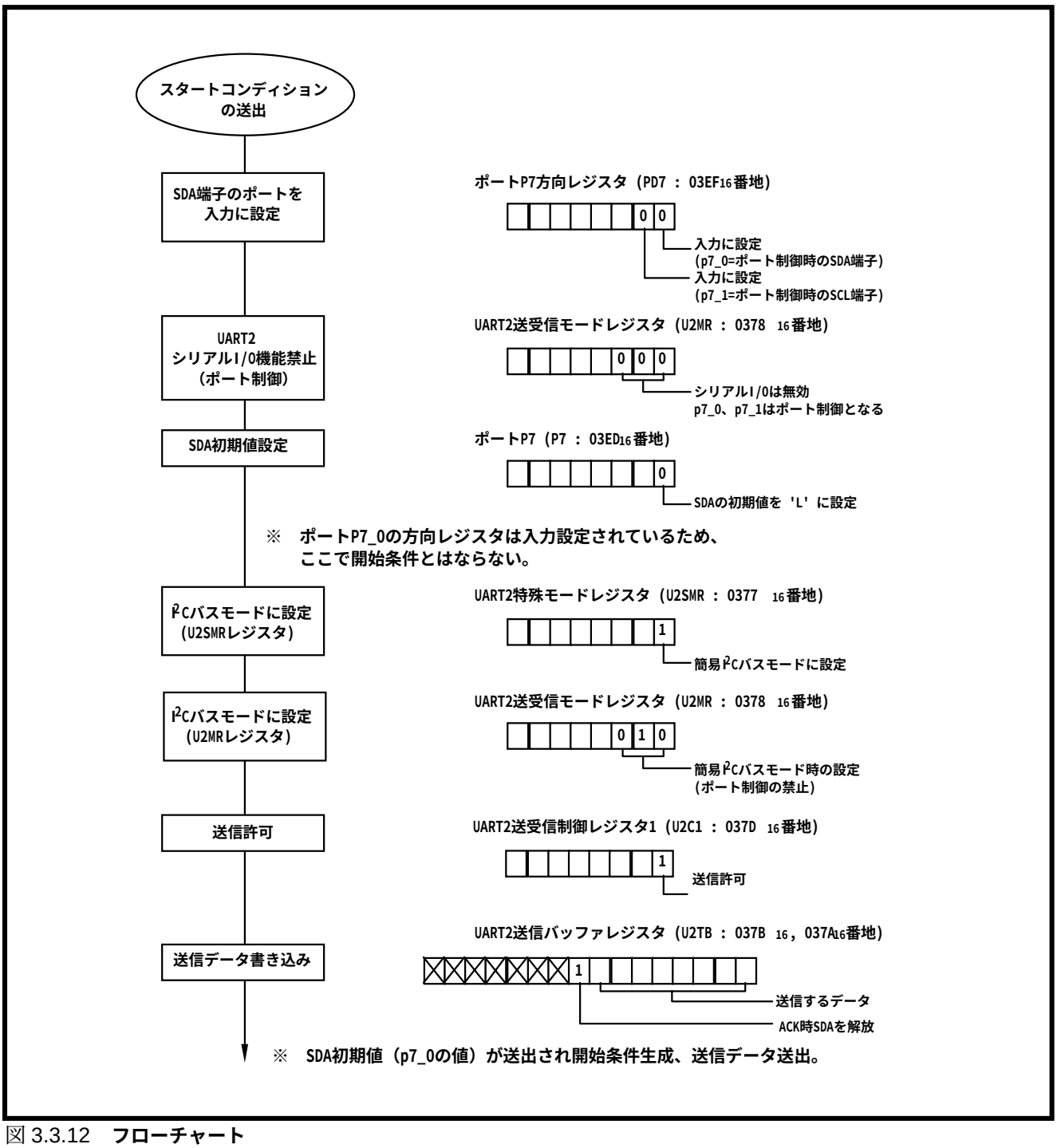


図 3.3.12　**フローチャート**

バスビジー検出

スタートコンディションを送出する前には、バスが解放されていることを確認する必要があります。M16C/62 の簡易 I^2C バスモードでは、バスの状態はバスビジーフラグ[BBS]で検出できます。

スタートコンディションを検出したとき[BBS]は"1"にセットされ、ストップコンディションを検出したとき[BBS]は"0"にクリアされます。従ってスタートコンディションを送出しようとするときに[BBS]="1"であったときバスは使用状態ですので、[BBS]="0"にクリアされるまで待ってから送出を開始します。

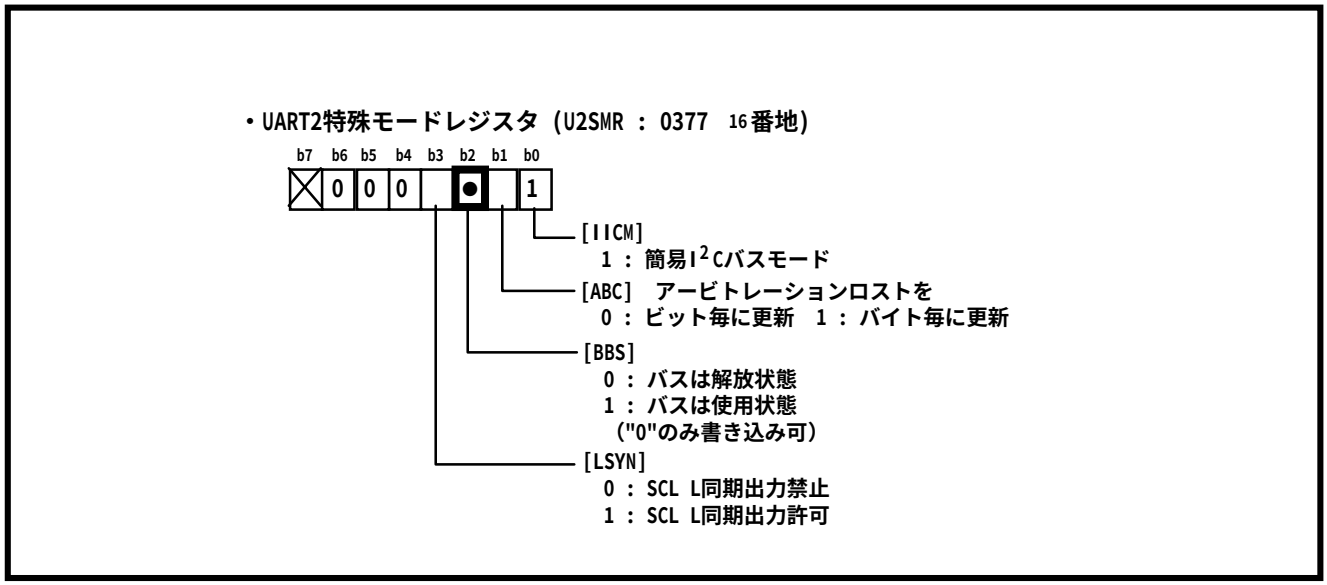


図 3.3.13 **関連レジスタ**

SCL 端子出力機能 2

M16C/62 のシリアル I/O は、送信データを送信バッファに書き込んでから転送クロック(簡易 I²C バスモードでは SCL)送出までに、最大で転送クロック(SCL)の 1.5 サイクルの期間を要します。また、M16C/62 の SCL 同期化機能（3.3.5 通信調整 参照）は 1 ビット目の SCL 送出時から有効になりますので、スタートコンディション発生後クロックライン(SCL)同期化機能が有効になるまでの期間内に他デバイスが 1 ビット目の送出を行った場合、ビットずれを起こす可能性があります（タイミング図上図）。このため、M16C/62 ではスタートコンディション送出後に他デバイスからのクロック送出を禁止するための SCL 端子 L 出力機能を持っています。この機能を使用することにより、送信バッファにデータを書き込むと同時に SCL 端子から"L"を出力し、他デバイスを待ち状態することができます（タイミング図下図）。本機能はウエイト出力ビット 2[SWC2 を"1"にすることで動作が許可され（SCL から'L'出力）、'0'にすることで解除されます。

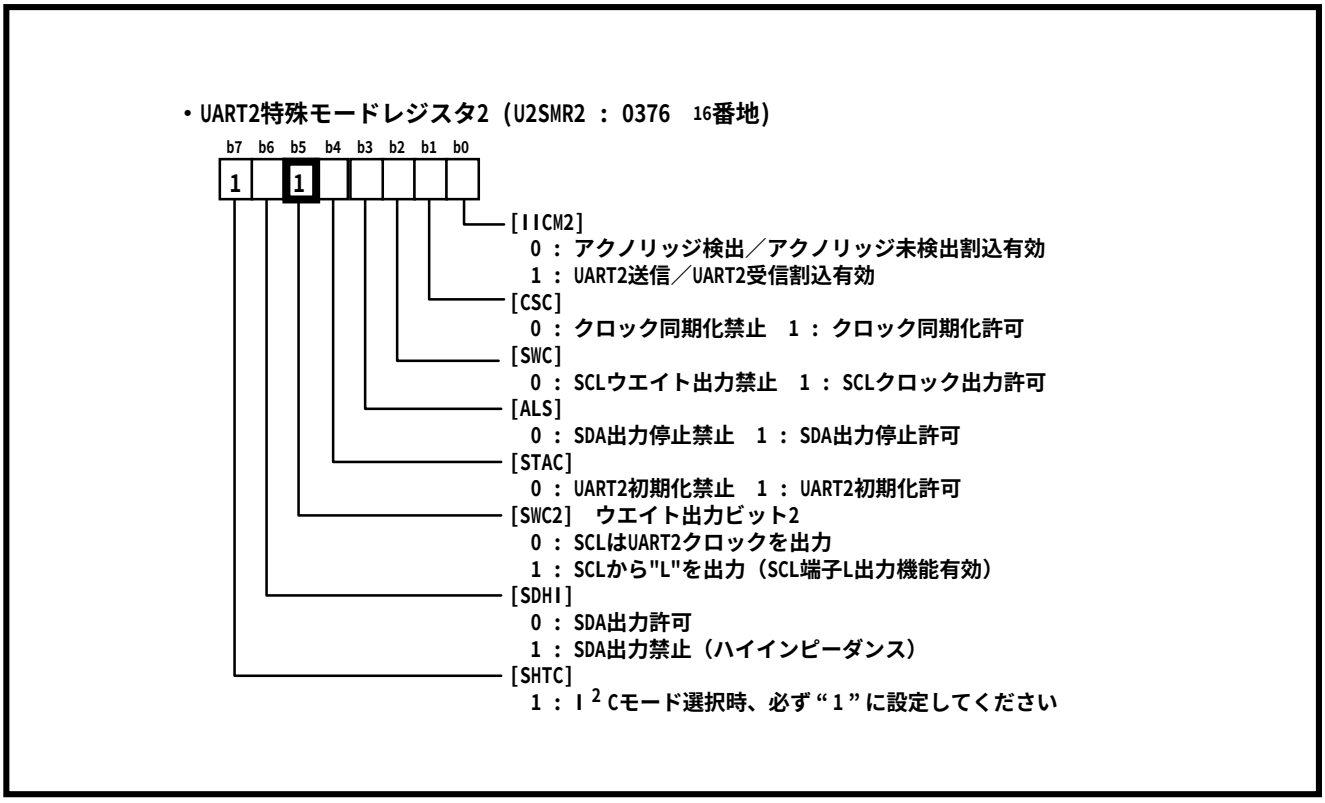
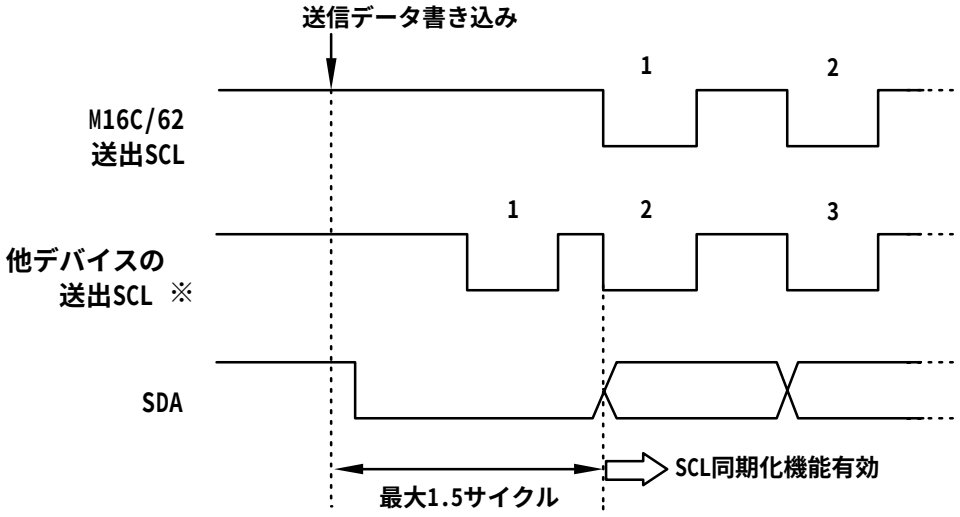


図 3.3.14 **関連レジスタ**

SCL 端子 L 出力機能 2 （つづき）

SCL端子L出力機能を使用しない場合



※　通常、ビットずれが生じる。

SCL端子L出力機能を使用する場合

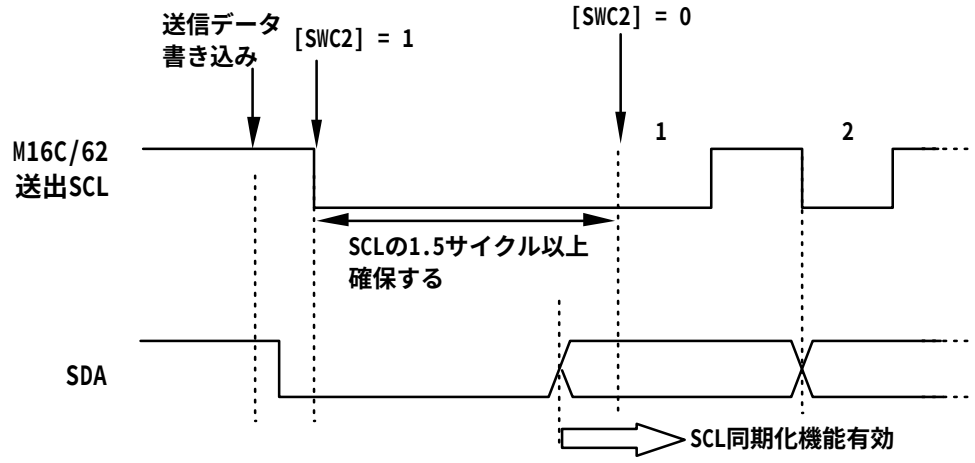


図 3.3.15　**タイミング図**

3.3.3 アクノリッジ

データの送受信には、1 バイト毎に確認応答（アクノリッジ）が付加されます。

M16C/62 を送信装置として使用する場合は、1 バイト毎に受信装置からのアクノリッジの有無を検出する必要があります。そのための機能として、「アクノリッジ検出割込」および「アクノリッジ未検出割込」をハードウエアで備えています。また M16C/62 を受信装置として使用し、一対一通信を行う場合は、送信データの 9 ビット目に'0'を設定することにより容易にアクノリッジの生成ができます。(3.3.1 「バイトデータの受信方法」参照)

「アクノリッジ検出割込」および「アクノリッジ未検出割込」を使用する場合には、I²C モード選択ビット 2(IICM2)を"0"に設定することが必要です。この設定により、割り込み番号 15 および割り込み番号 16 の要因がそれぞれ「アクノリッジ未検出割込」、「アクノリッジ検出割込」となります。その場合、UART2 受信レジスタから受信バッファレジスタへのデータ転送タイミングは、受信クロックの最終ビットの立ち上がりとなります。(3.2.4 「簡易 I²C バスモード時のレジスタ設定」UART2 関連レジスタ(4)表 1 参照)

アクノリッジ検出

送信クロックの９ビット目の立ち上がり時に、送信装置側では解放している SDA のラインのレベルが"L"になっている場合、受信装置側がアクノリッジを生成したと判断できます。M16C/62 の場合、これは「アクノリッジ検出割込」機能で検出できます。本割込は「ソフトウエア割込番号 16」に割り当てられ、I²C モード選択時（[IICM]="1"）で、かつ I²C モード選択ビット 2[IICM2]="0"の時のみ割り込み番号 16 の割込要因が「アクノリッジ検出割込」となります。

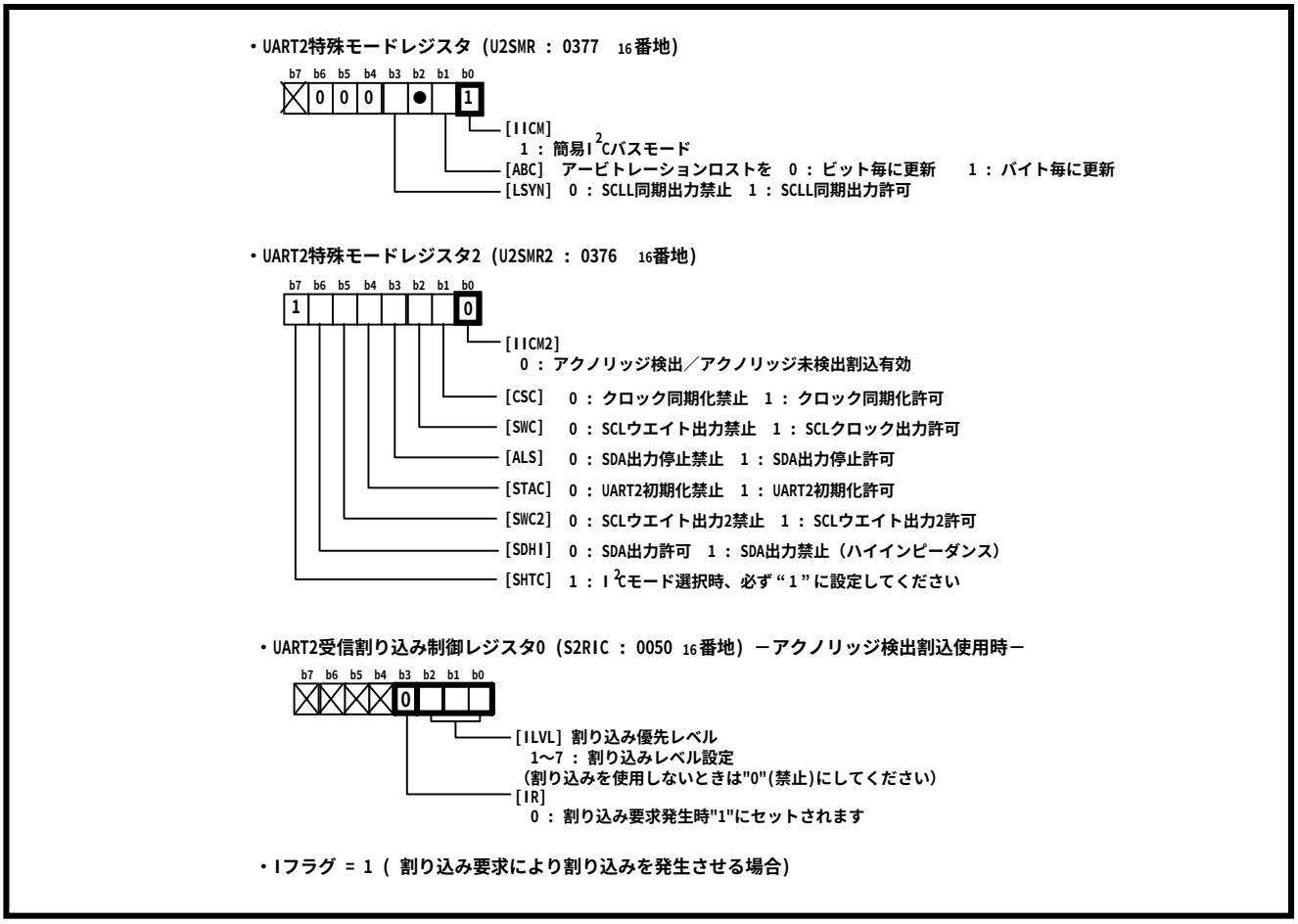


図 3.3.16　**関連レジスタ**

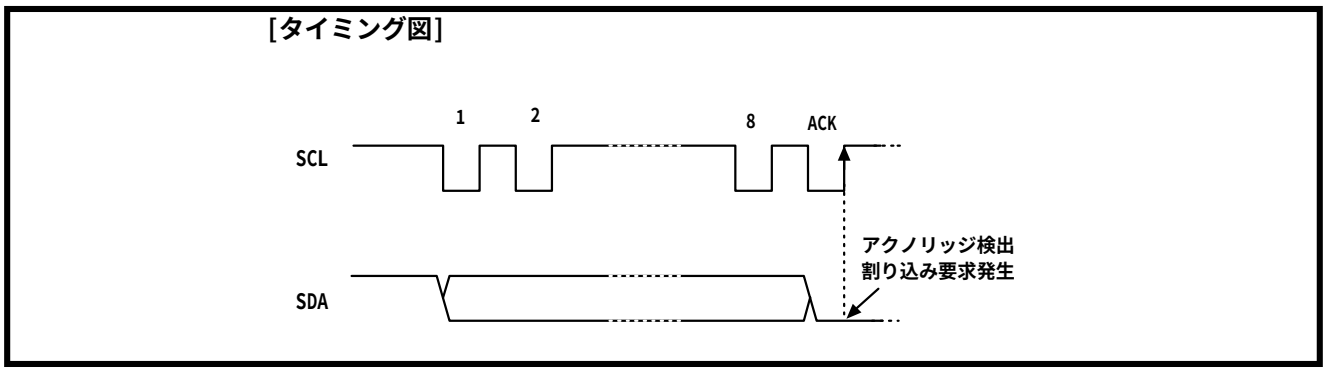


図 3.3.17　**タイミング図**

アクノリッジ未検出

送信クロックの９ビット目の立ち上がり時に、送信装置側で解放している SDA のラインのレベルが"H"になっている場合、受信装置側がアクノリッジを生成していないと判断します。M16C/62 の場合、これは「アクノリッジ未検出割込」機能で検出できます。本割込は「ソフトウエア割込番号 15」に割り当てられ、I²C モード選択時（[IICM]="1"）で、かつ I²C モード選択ビット[IICM2]="0"の時のみ割り込み番号 15 の割込要因が「アクノリッジ未検出割込」となります。

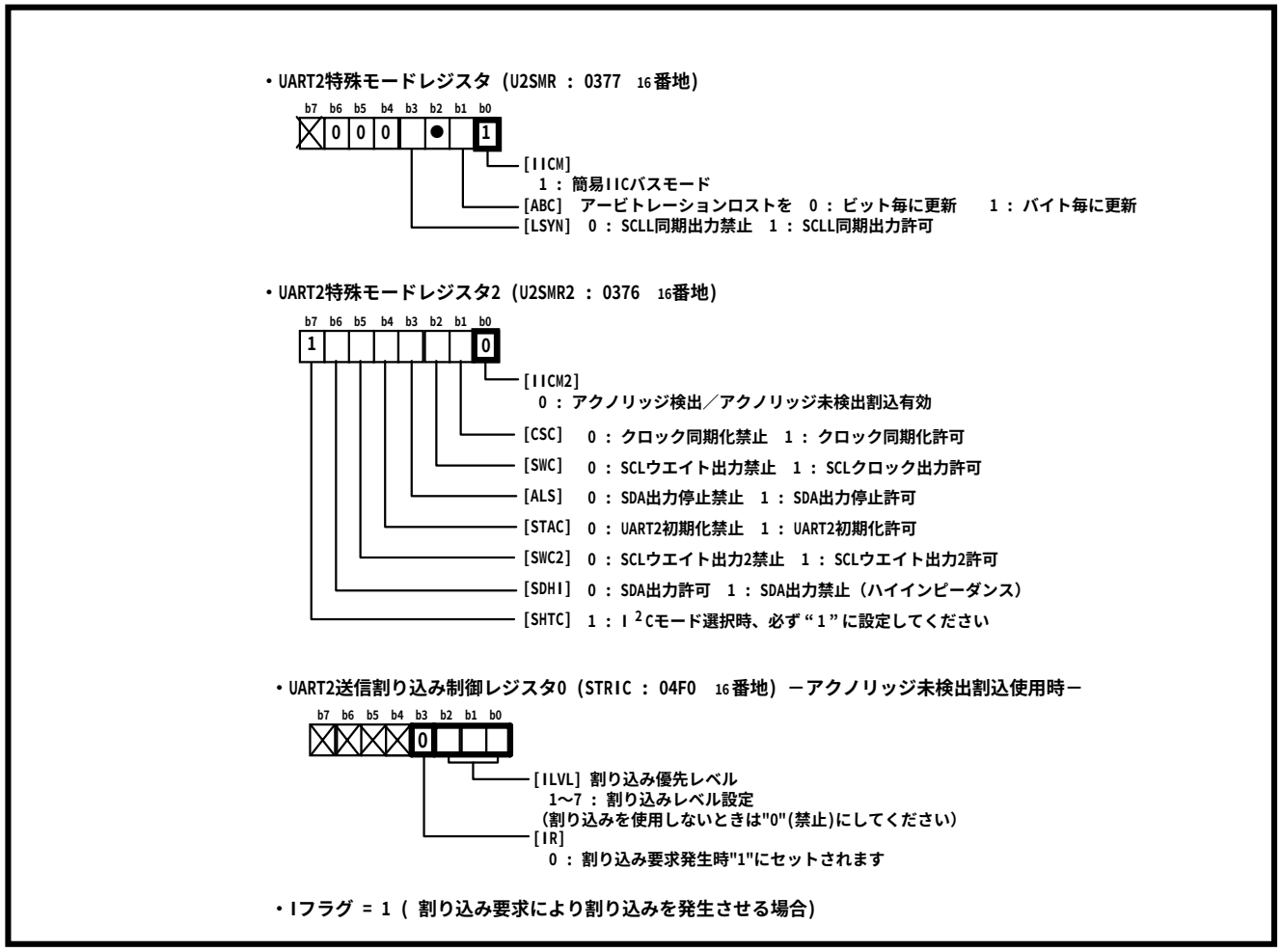


図 3.3.18　**関連レジスタ**

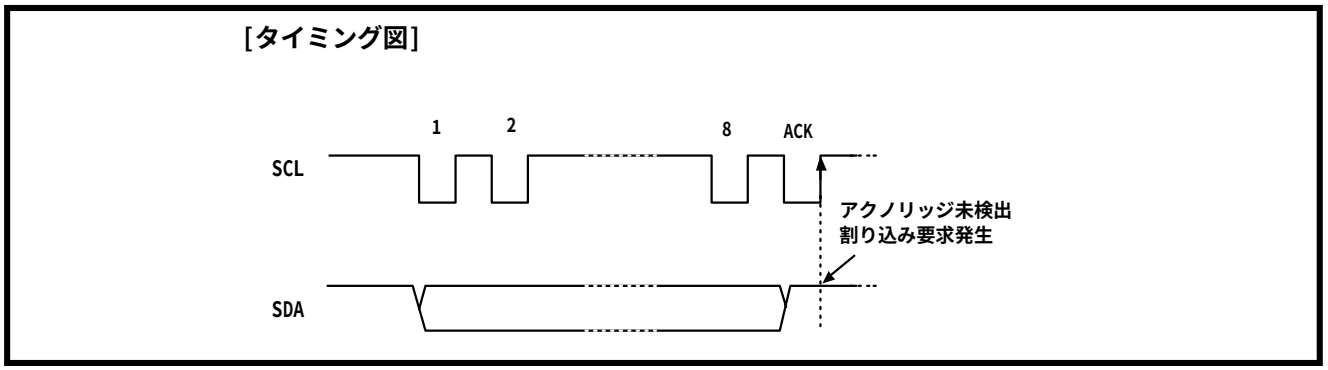


図 3.3.19　**タイミング図**

3.3.4 自己アドレス指定の判定

M16C/62 をスレーブとして使用している場合、マスタから送信されたアドレスが自己のアドレスと一致するか比較し、一致したときスレーブ(M16C/62)はマスタに対しアクノリッジを送出します。M16C/62 の簡易 I^2C バスモードでは、自己アドレスとの比較およびアクノリッジの送出ソフトウエアで行いますが、その間 SCL 端子を L ホールドしマスタを待ち状態にする「SCL 端子 L 出力機能」をハードウエアで備えています。

SCL 端子 L 出力機能

I²C バスでは、スタートコンディション検出後の第一バイトに、指定されるスレーブのアドレスが送出されます（7 ビットアドレスモード時）。スレーブは、第一バイトに他のマスタから送出されるクロックのはじめの 7 ビットの受信データを自己アドレスと比較し、クロックの 9 ビット目に同期してアクノリッジを生成（または未生成）する処理が必要です。本処理のための機能として M16C/62 では「SCL 端子 L 出力機能」を持っています。この機能を用いると、初めの 8 ビットのデータを受信後、9 ビット目の SCL が"L"になるタイミングで M16C/62 の SCL 端子に"L"を出力して、強制的にマスタを待ち状態にします。そしてソフトウエアでのアドレス比較処理が終了後に、ポート制御でアクノリッジの生成／未生成を行うことができます。（一対一通信で使用している場合等は「3.3.1 バイトデータの受信方法」で説明した方法で、アドレスの受信、アクノリッジの生成が行えます）

本機能は、ウエイト出力ビット[SWC]を"1"にすることにより動作が許可され、"0"にすることにより禁止されます。また、本機能で SCL 端子が"0"になったときは、[SWC]を"0"にすることで解除できます。

なお、本機能によりアドレス比較処理を行う場合、最終ビットのクロックの立ち上がり前に受信バッファレジスタの内容を読み出すことになりますので、読み出した受信データはビットの位置が変化していることにご注意ください。（3.3.1「送信割込／受信割込」のタイミング図参照）

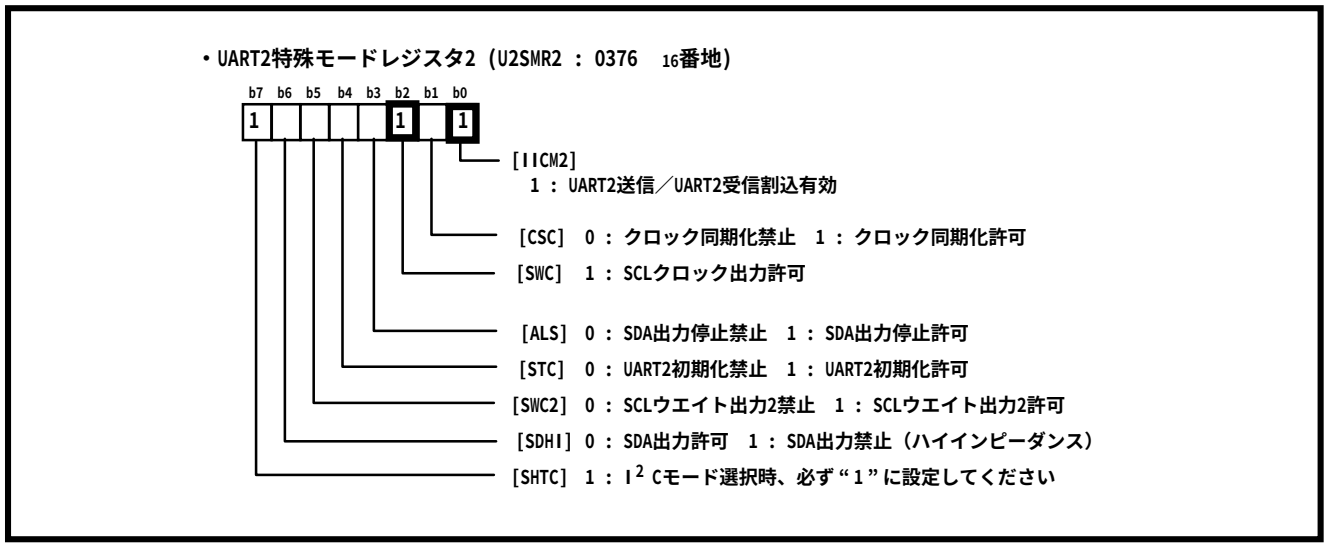


図 3.3.20　**関連レジスタ**

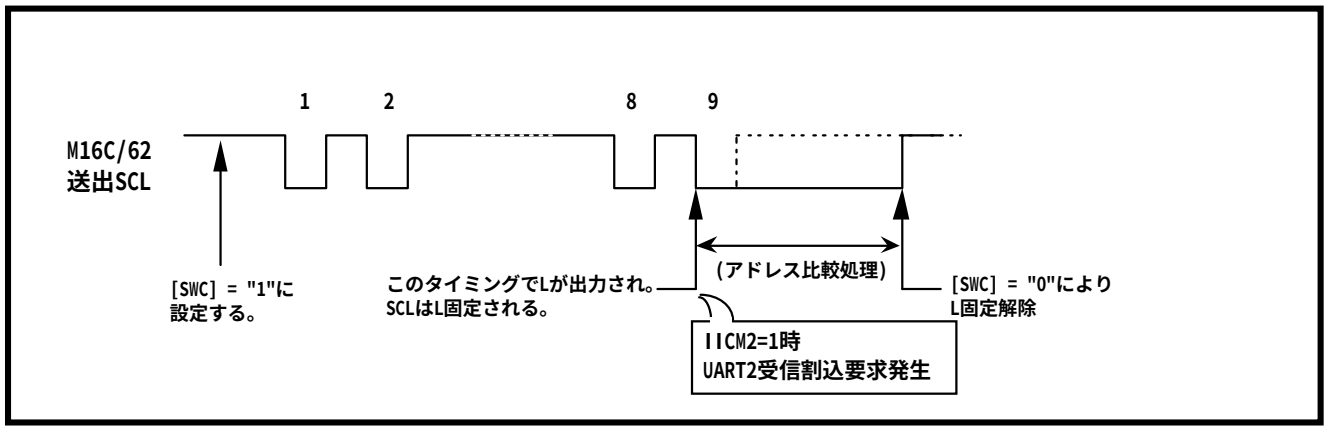


図 3.3.21　**タイミング図**

自己アドレス判定の一例

アドレス指定のフォーマットは、7 ビットアドレスと 10 ビットアドレスの 2 通りあります。ここでは、7 ビットアドレスの判定の一例をご紹介しますが、10 ビットアドレスのフォーマットでも同様にして制御できます。ここでは、スタートビット受信後、第一バイトの受信割り込みが発生したときの制御の一例をご紹介します。ここでは、第一バイト受信前に[SWC]="1"（SCL 端子 L 出力機能許可）をすでに設定しているものとし、受信割り込みルーチンの一部のみ記述します。

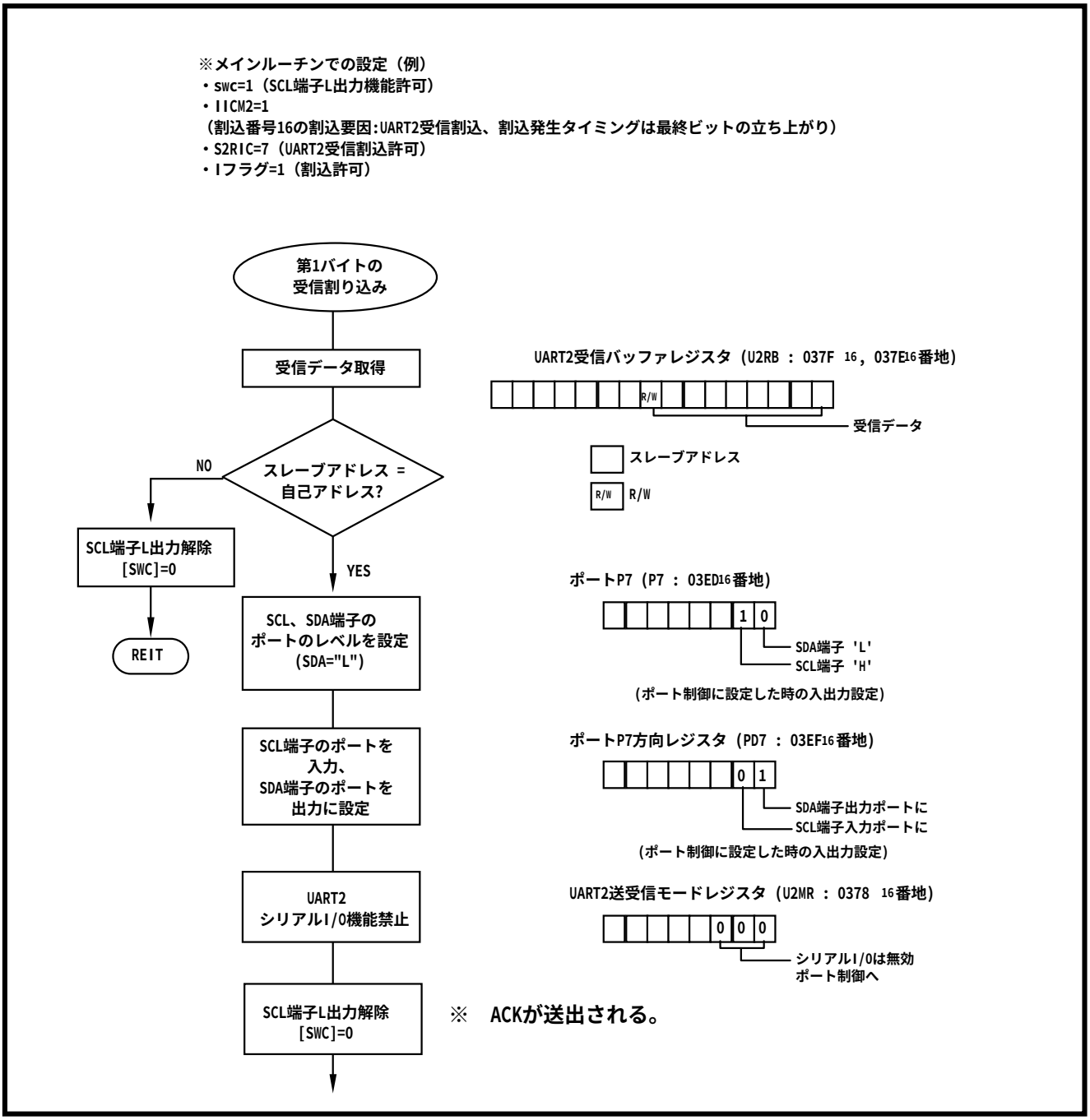


図 3.3.22 **フローチャート**

3.3.5 通信調整

I²C バスをマルチマスタで使用する場合は、複数のマスタが同時にスタートコンディションを生成してデータ送信をスタートしようとすることが考えられます（アービトレーション発生）。この場合、I²C バスのシステムでは複数のマスタ間で通信調整が行われます。M16C/62 の簡易 I²C バスモードでは、アービトレーション・ロスト発生時に通信を回復するための機能として、「アービトレーション・ロスト検出機能」、「アービトレーション・ロスト発生時の SDA 出力禁止機能」をハードウエアで備えています。またアービトレーション・ロスト以外でも、クロック同期を利用した通信調整のひとつとして「SCL 同期化機能」も備えています。

アービトレーション・ロスト検出機能

M16C/62 の簡易 I²C バスモードは、アービトレーション・ロスト検出フラグ[ABT]を持っています。この検出フラグは UART2 受信バッファレジスタのビット 3 に配置されており、SCL の立ち上がりのタイミングで内部データ・レベルと SDA のレベルの不一致を検出するとアービトレーションロスト検出フラグ[ABT]は"1"にセットされます。アービトレーションロスト検出フラグの更新は、アービトレーションロスト検出フラグ制御[ABC]により、ビット毎（"0"）／バイト毎（"1"）の選択が行えます。I²C モード選択ビット[IICM]、および I²C モード選択ビット 2[IICM2] が共に"1"のときは、[ABC] =0 に固定してください。

アクノリッジ受信のタイミングでは出力"H"、入力"L"となりアービトレーションロスト検出フラグがセットされてしまいますので、次の送信時にはアービトレーションロスト検出フラグを"0"にクリアしてから送信を行ってください。

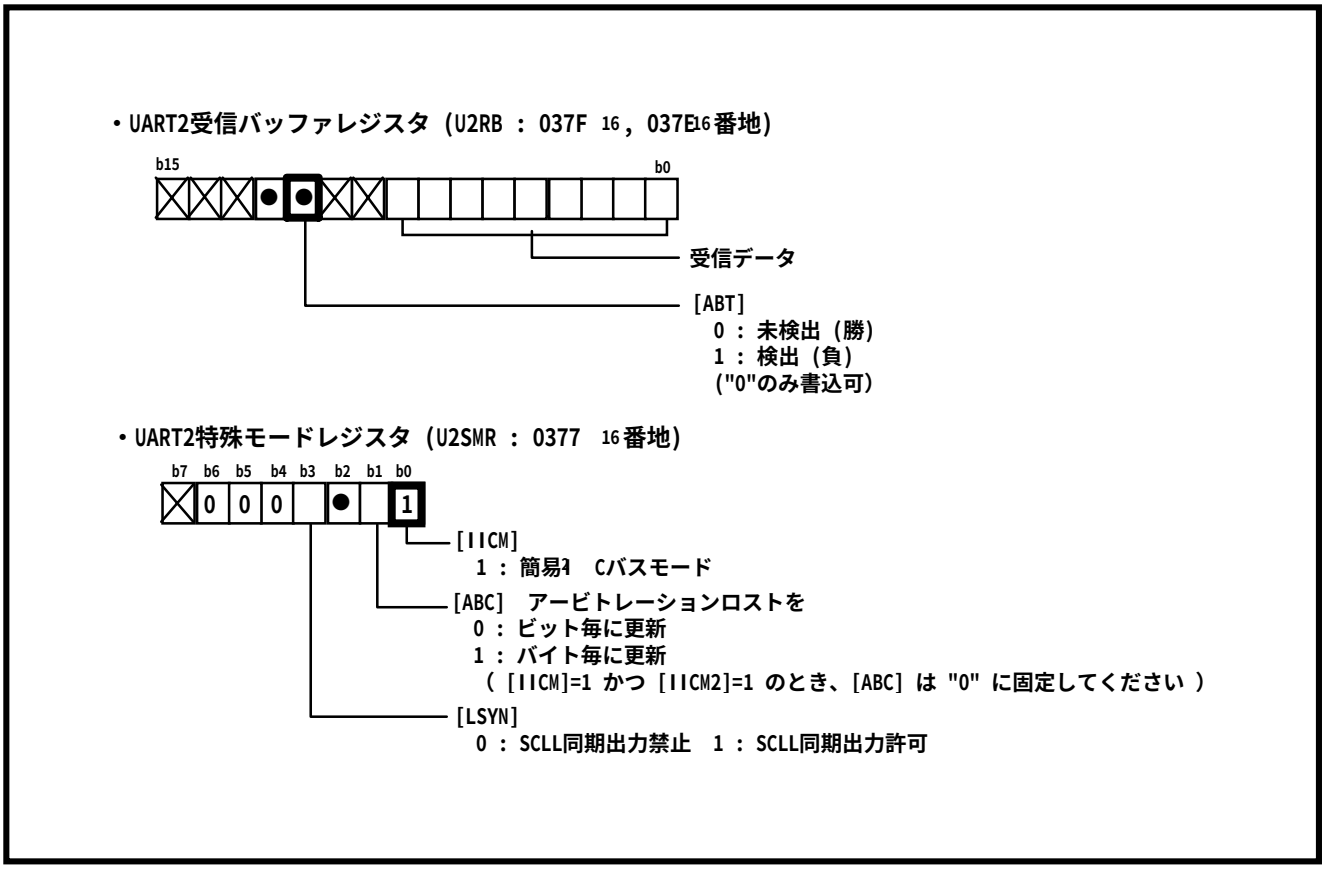


図 3.3.23 **関連レジスタ**

アービトレーション・ロスト発生時の SDA 出力禁止機能

アービトレーション・ロストの発生を検知したマスタは、その時点で SDA の出力をオフしなくてはなりません。M16C/62 の簡易 I²C バスモードは、アービトレーション・ロスト発生時にハードウエアで自動的に SDA 出力をオフする機能を選択できます。本機能は、SDA 出力停止ビット[ALS]に"1"を設定することで許可され、"0"を設定することで禁止されます。本機能により SDA 出力がオフされた場合は、SDA 出力停止ビット[ALS]またはアービトレーションロスト検出フラグ[ABT]をクリアすることにより解除されます。ただし本機能を許可している場合はアクノリッジのタイミングでもアービトレーション・ロスト発生と判断して出力をオフしてしまいますので、次のバイトデータの送出はアービトレーション検出ビット[ABT]をクリアしてから行ってください。また、アービトレーション検出フラグ制御[ABC]は"0"に固定してください。

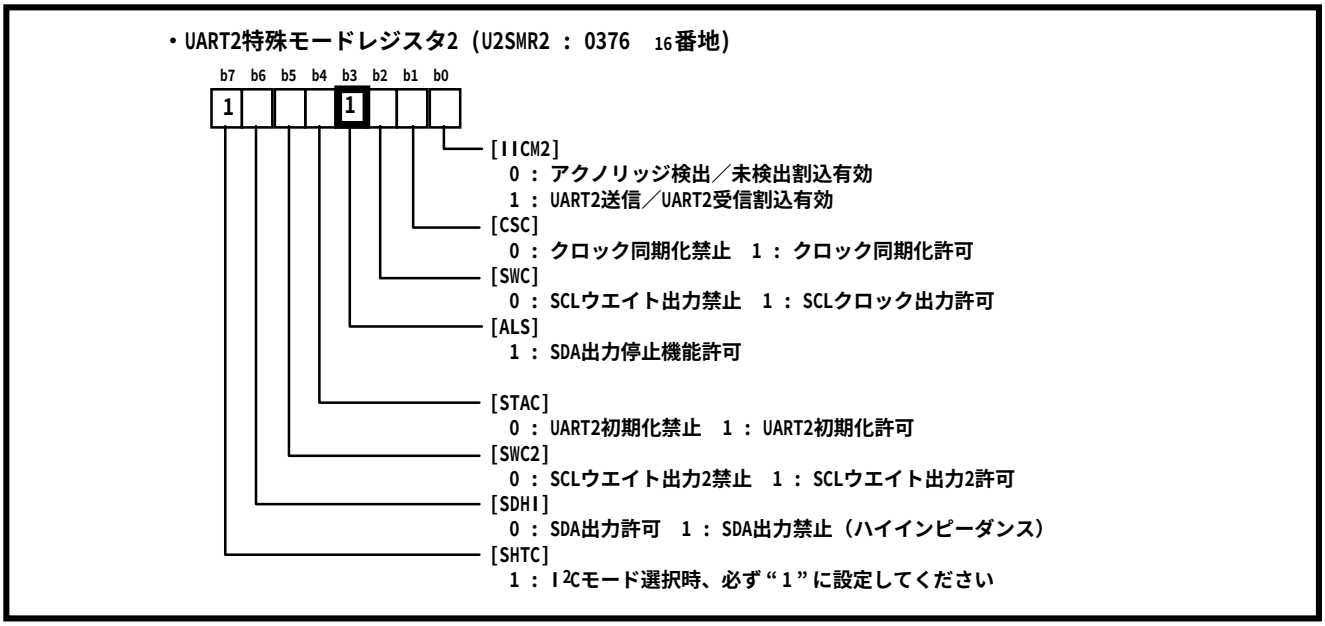


図 3.3.24　**関連レジスタ**

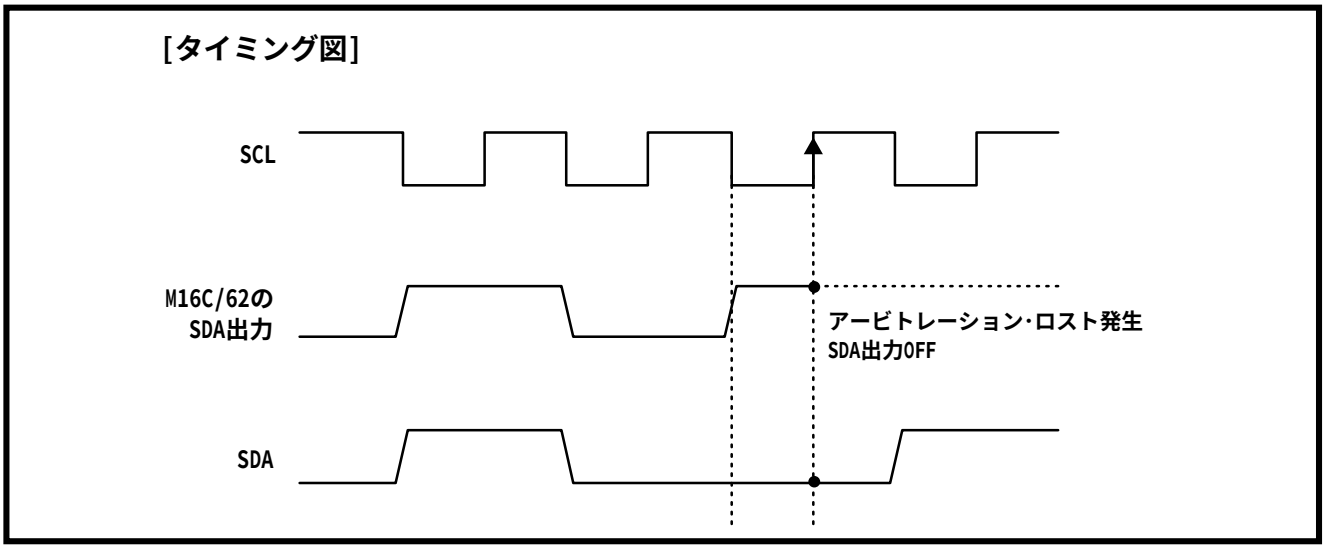


図 3.3.25　**タイミング図**

SCL 同期化機能

M16C/62 を処理速度の遅いデバイスと接続している場合、他のデバイスが SCL に対し L ホールドを行い、マスタから送出するクロックを強制的に待ち状態にすることがあります。M16C/62 の簡易 I²C バスモードでは、他デバイスからの L ホールドに対し自動的に待ち状態に入り、L ホールドの解除により待ち状態も解除する SCL 同期化機能を持っています。本機能は、クロック同期化ビット[CSC]を"1"に設定することにより動作が許可され、"0"を設定することにより禁止されます。本機能は M16C/62 をマスタとして使用した設定（内部クロックモード）でのみご使用ください。

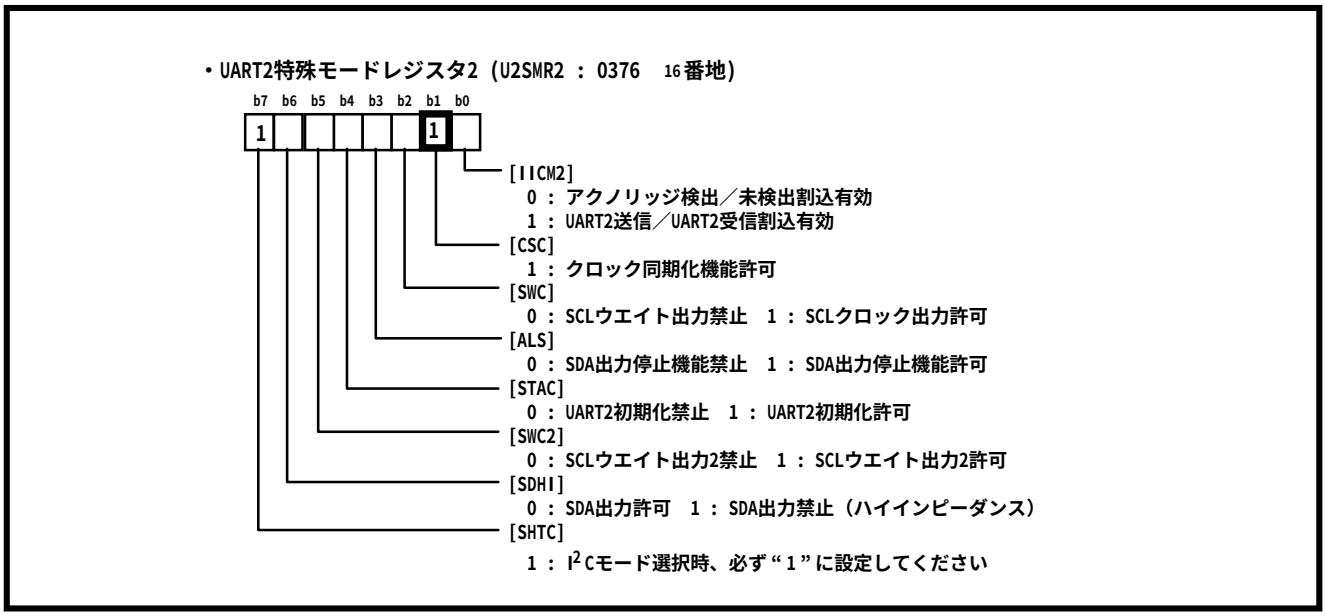


図 3.3.26 **関連レジスタ**

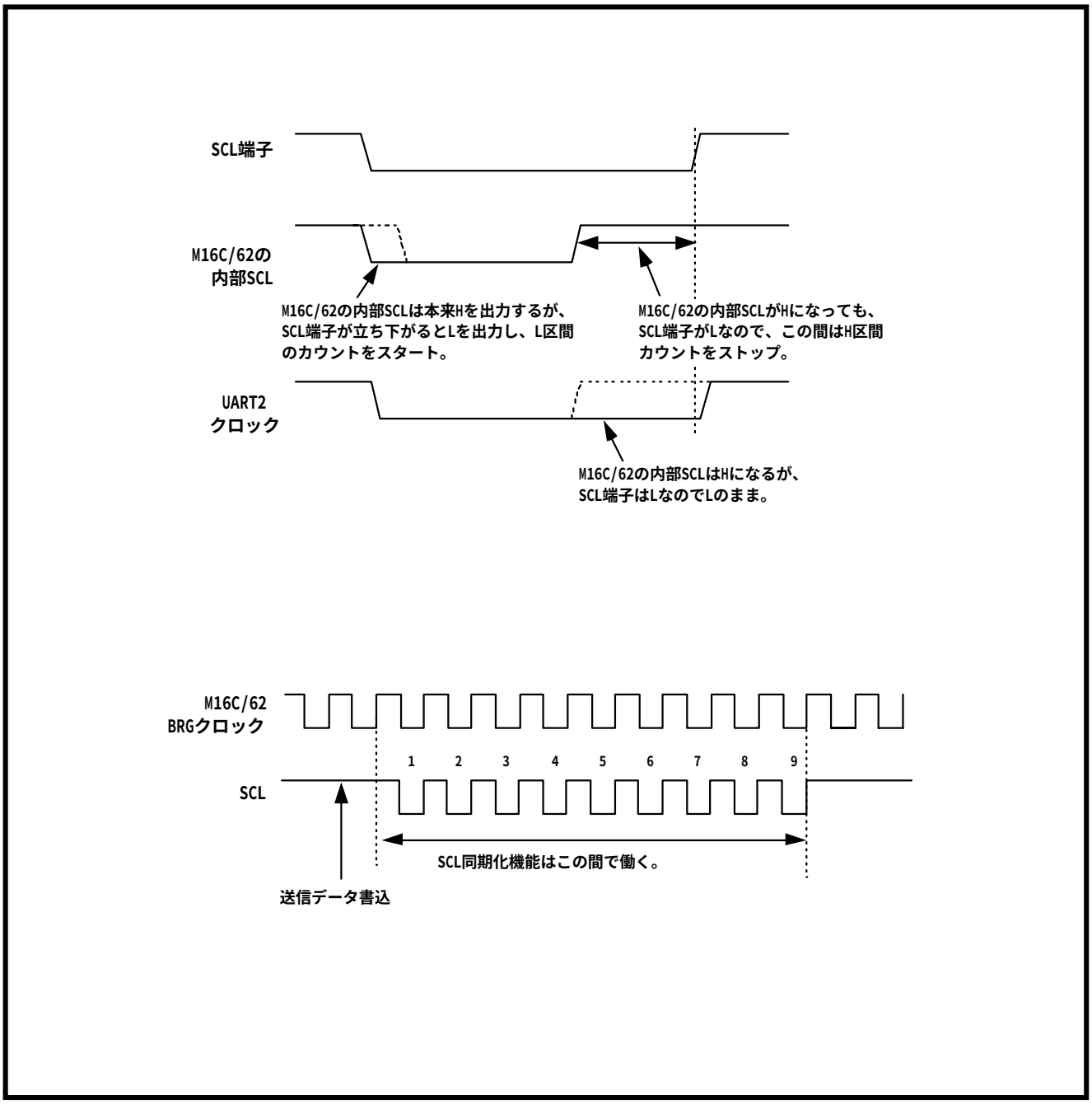


図 3.3.27　**タイミング図**

3.3.6 その他の機能

M16C/62 の簡易 I²C バスモードは、上記に説明した他に、I²C バス制御を容易にする機能として以下の機能もハードウエアで備えています。

SCLL 同期出力機能

M16C/62 の簡易 I²C バスモードは、SCLL 同期出力機能を持ちます。本機能は、SCLL 同期出力許可ビット[LSYN]を"1"にすることで許可され、"0"にすることで禁止されます。SCLL 同期出力許可ビット[LSYN]を"1"にセットすることで、SCL 端子のレベルが"L"になることに同期して p7_1（SCL 端子が配置されているポート）のデータレジスタを"0"にリセットします。本機能は SCL/SDA をポートとして使用する場合（シリアル I/O モード選択ビット[SMD0,SMD1,SMD2]="000"時）のみ有効で、シリアル I/O モードでは無効となります。

M16C/62 でスレーブ受信を行う場合、本機能を使用することにより第一バイトのアクノリッジの送出も可能ですが、通常は SCL 端子 L 出力機能を使用してください。

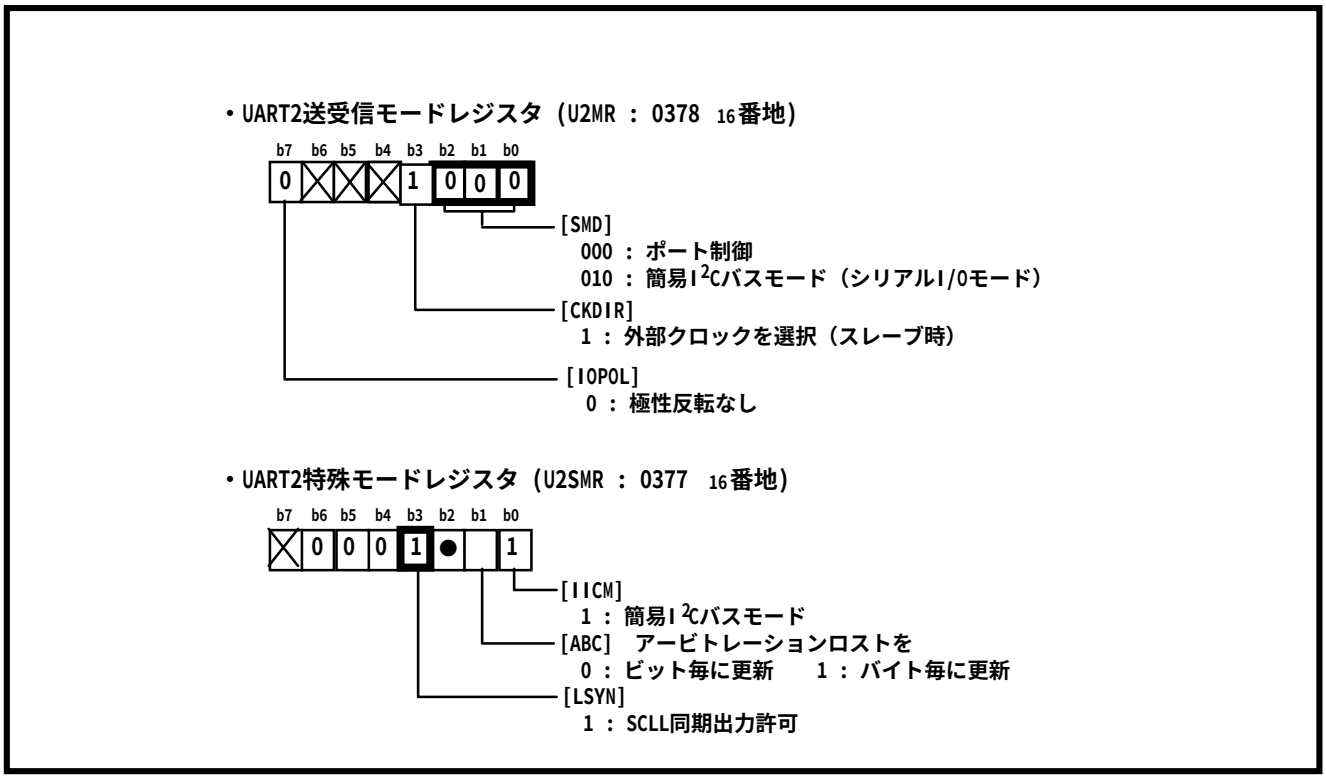


図 3.3.28 **関連レジスタ**

SDA 出力禁止機能

M16C/62 をスレーブとして使用し、スタートコンディション受信後の 1 バイト目でアドレス判定を行う際、マスタが指定したアドレスが自アドレスと異なれば M16C/62 は SDA の出力をオフ（ハイインピーダンス）にしなければなりません。その場合、SCL の 9 クロック毎（受信割り込み要求が発生する毎）に M16C/62 の送信バッファレジスタに "1FFh"を設定することにより SDA 出力をオフにします。また、M16C/62 が備えている SDA 出力禁止機能を使用することでも SDA 出力をオフすることができます。本機能は、SDA 出力禁止ビット [SDHI]を"1"にすることで有効となり、送信バッファレジスタに"1FFh"を設定しなくても M16C/62 の SDA 出力をハイインピーダンスにできます。[SDHI]を"0"にすることで本機能は解除され、次の SCL の入力に同期して送信バッファに設定した値が出力されます。

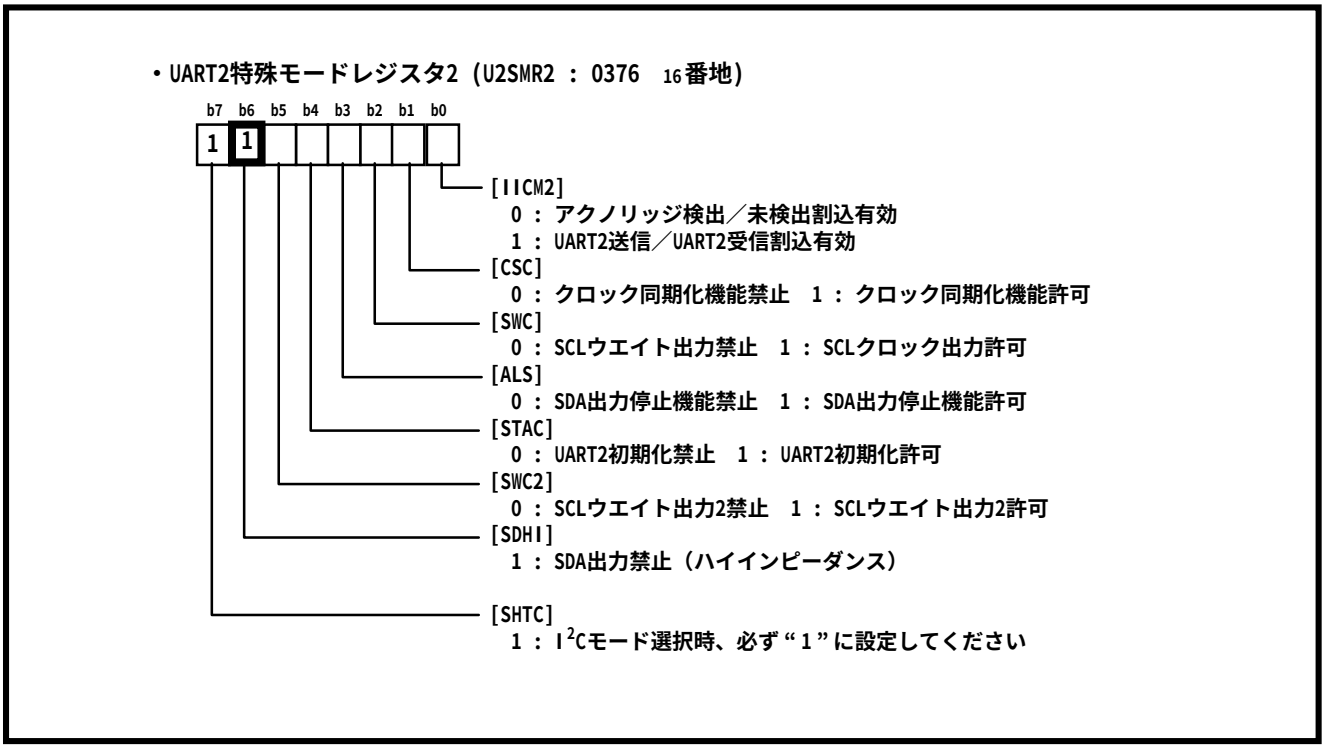


図 3.3.29 **関連レジスタ**

UART2 初期化機能

M16C/62 の簡易 I²C バスモードは、スタートコンディション検出のタイミングで自動的に UART2 を初期化する機能を備えています。本機能は M16C/62 をスレーブとする際に使用します。スタートコンディション初期化ビット[STAC]を"1"にすることにより機能が許可され、"0"にすることにより禁止されます。スタートコンディションを検出すると、以下の初期化が行われます。

① 送信レジスタは初期化され、送信バッファレジスタの内容が送信レジスタに転送されます。これにより、データ受信時に送信バッファレジスタにデータを再設定する必要はなく、次に入力されたクロックを 1 ビット目として送信が開始されます。ただし、その時の送信データは最後に送信していたデータと同じものとなりますので、SDA 出力禁止ビット[SDHI]を"1"として送信データの出力を禁止してください。

② 受信レジスタは初期化され、次に入力されたクロックを 1 ビット目として受信が開始されます。受信バッファレジスタを読み出す前のタイミングで初期化され受信が開始しても、オーバランエラーは発生しません。

③ ウエイト出力ビット[SWC]が"1"にセットされます。これにより SCL 端子 L 出力機能が有効となり、転送クロックの 9 ビット目のたち下がりで SCL 端子からＬが出力されます。

本機能は外部クロック選択でのみ使用してください。また本機能を使用しUART2の送受信を開始した場合、送信バッファ空フラグの値は変化しませんのでご注意ください。

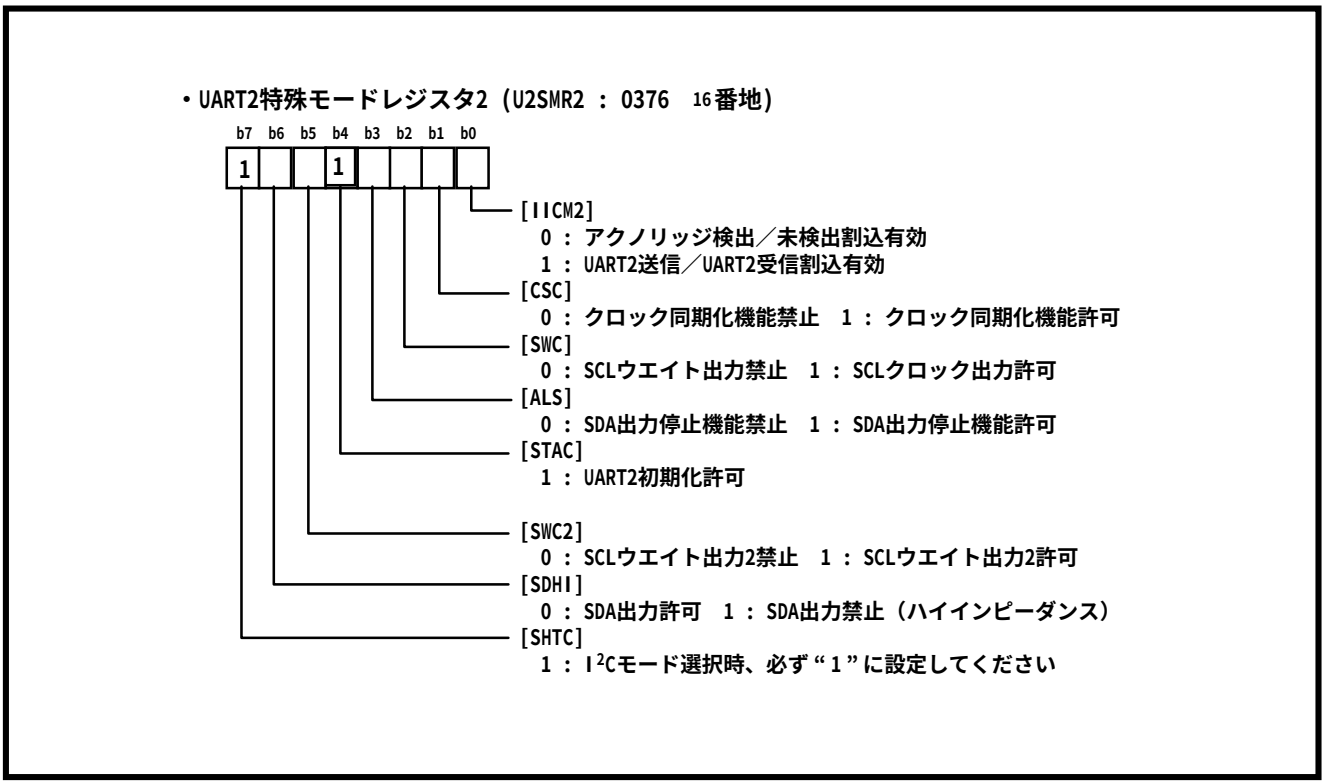


図 3.3.30 **関連レジスタ**

SDA ディジタル遅延機能

I²C バス規格では、SCL 立ち下がり幅の未定義領域を埋めるために、SDA 信号用に最低 300ns のホールド時間を内部的に提供することを要求しています。そのため、M16C/62 の SDA 出力には、ディレイ回路が付加され、SCL が十分"L"になった後、SDA 出力が変化します。ディレイ回路は、SDA ディジタル遅延選択ビット[SDDS]によって、アナログ遅延とディジタル遅延を選択することができます。本ビットでディジタル遅延を選択した場合、遅延値を SDA ディジタル遅延値設定ビット[DL0～DL2]によって、f1 の 2 サイクルから 8 サイクルまで選択することができます。その場合、アナログ遅延は付加されず、ディジタル遅延のみとなります。

通常は、ディジタル遅延を選択し、遅延値を I²C バス規格にあわせて設定してください。

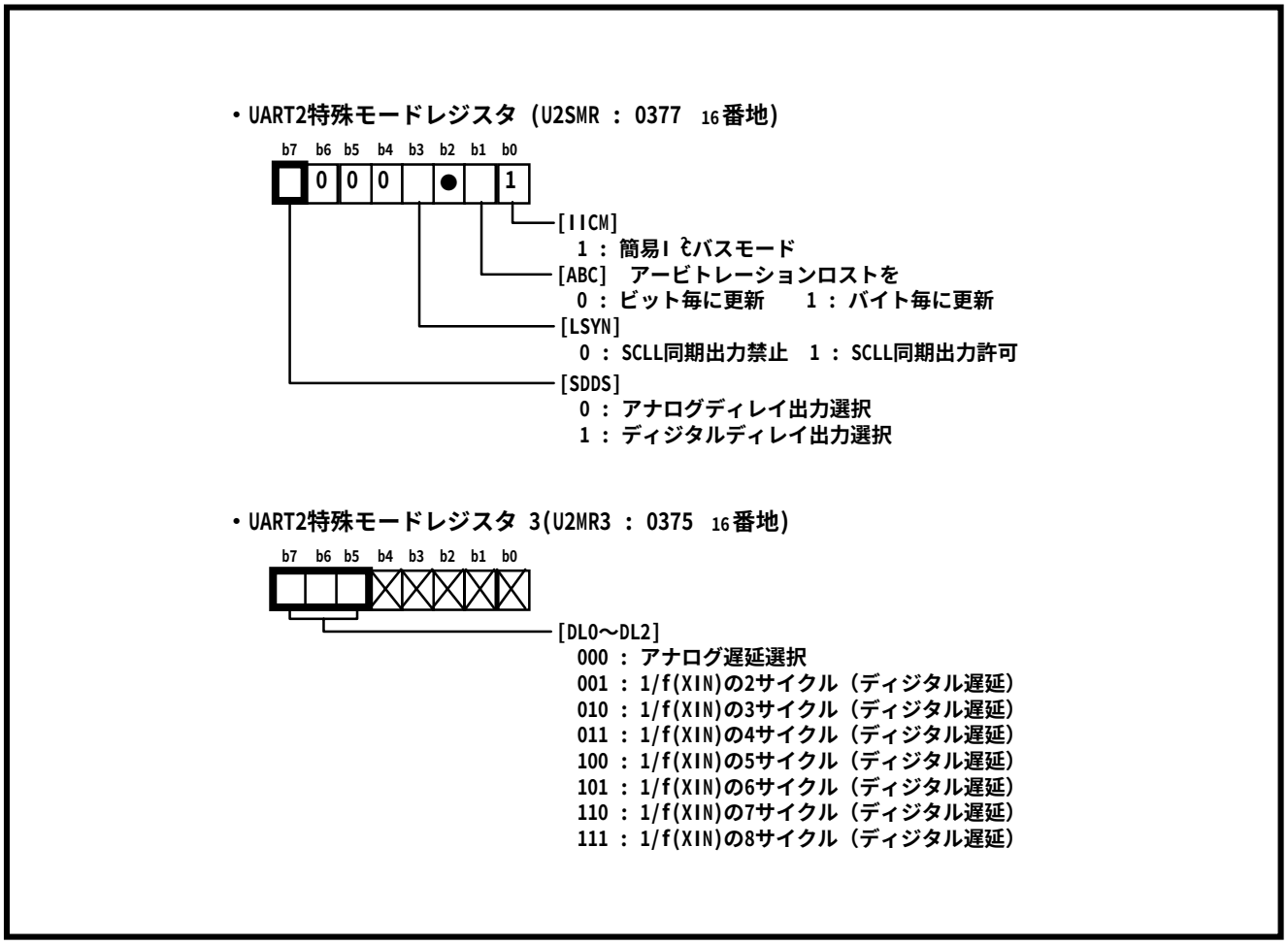


図 3.3.31 **関連レジスタ**

3.4 簡易 I²C バスモード使用時の注意事項

M16C/62 の簡易 I²C バスモードを使用し、I²C バスのプロトコル制御を行う場合、以下の注意事項および制限事項があります。

3.4.1 電気的特性

M16C/62 の電気的特性は、I²C バス規格と一部異なる点があります。（I²C バス規格は 3.1.7「I²C バスの SDA および SCL バス・ラインの特性」参照）

LOW レベル／HIGH レベル入力電圧

M16C/62 の電気的特性は、2.7V ～5.5V 動作時において

"H"入力電圧(V_{IH})=min.0.8Vcc （保証値）
"L"入力電圧(V_{IL})=max.0.2Vcc （保証値）

となります。従って、I²C バスの規格値

5V 動作時：V_{IH} =3V 、V_{IL} =1.5V
5V 以外 ：V_{IH} =0.7Vcc 、V_{IL} =0.3Vcc

とは異なります。

また M16C/62 の"L"出力電圧は、V_{cc} =5V 、 I_{OL} =5mA 時に

"L"出力電圧(V_{OL})=max.2.0V （保証値）

となります。

I²C バスの規格値

"L"出力電圧(V_{OL})=max.0.6V （I_{OL} =6mA 時）

とは異なります。

ただし M16C/62 の標準特性においては、V_{cc} =5V 、 I_{OL} =5mA 時に

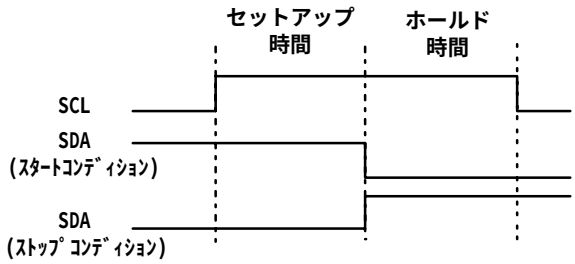
"L"出力電圧(V_{OL})=0.6V

程度となります

スタート／ストップコンディションのセットアップタイム･ホールドタイム

M16C/62 のスタートコンディション／ストップコンディション検出時のセットアップタイムおよびホールドタイムは、I²C バスの規格値と異なる場合があります。（高速モード時）

M16C/62 のスタートコンディション／ストップコンディション検出時のセットアップタイムおよびホールドタイムは、以下の値となります。（注 1）



セットアップ時間 ＞ 3 ～6 サイクル（注 2 ）
ホールド時間 ＞ 3 ～6 サイクル（注 2 ）

（注 1）スタート／ストップコンディション SHTC ビットは、必ず “ 1 ” に設定してく ださい。
（注 2）サイクル数はメインクロック入力発振周波数 f((XIN)のサイクル数を示します。

高速モード I²C バスの規格において、スタートコンディションおよびストップコンディションのセットアップ／ホールドタイムはどちらも Min.600ns です。それに対し、M16C/62 のセットアップ／ホールドタイムは Min.6 サイクル(f(XIN)のサイクル数)となります。従って、メインクロック（f(XIN)）を 10MHz で使用した場合は、M16C/62 の簡易 I²C バスのセットアップ／ホールドタイムは Min.600ns となり高速モード I²C バス規格にも対応できますが、メインクロックを 10MHz 未満で使用する場合は、セットアップ／ホールドタイムは高速モード I²C バス規格を満たすことができなくなります。

3.4.2 BRG **カウントソースによる最大転送速度の制限**

M16C/62 が SCL のレベルを認識するまでの時間はサンプリング周期に依存し、最大 BRG カウントソースの 3 クロック分を要します。従って、動作周波数および BRG カウントソース設定ビットで設定した BRG カウントソースの速度によって、M16C/62 の簡易 I^2C バスに接続可能な I^2C バスの最大転送速度が制限されます。以下の条件を満たす転送速度で使用しない場合、ビットずれを起こす可能性があります。

I^2C バスの最大転送速度 (Hz)＜ BRG カウントソース (Hz)/ 3

（例）源発振 10MHz 、BRG カウントソースに fc32 を選択した場合

$$I^2C \text{ バスの最大転送速度(Hz)} \quad < \quad \frac{10MHz/32}{\text{(BRG カウントソース)}} \quad / 3 \quad = \quad 104Kbps$$

この場合の I^2C バスの最大転送速度は、104Kbps になります。

3.4.3 **マルチマスタで使用する場合の制限**

同一スレーブに対し、データ長の異なるマスタ送受信を行うマルチマスタの環境では、
M16C/62 の I^2C バスモードは使用できません。

4.0 参考プログラム

最後に、M16C/62 の簡易 I²C バスモードを使用してマルチ・マスタで送受信を行う場合の参考プログラムをご紹介します。本プログラムは通信動作を保証するものではありませんので、ご使用の場合は十分な評価を行ってください。

ソフト I²C バスコントローラ仕様書

[目次]

4.1 概要

本ソフトウェアは M16C/62 グループに内蔵された簡易 I²C バスモードの H/W を制御し、I²C バスの通信プロトコルを実現するものです。

下記使用条件により I²C バスの通信プロトコルに準拠します。

＜動作条件＞
発信周波数：10MHz（ノンウェイト、分周なし）

＜仕様上の制限事項＞
- I²C バス上でバスラインがロックしない事が前提です。もしバスがロックした場合は、本ソフトウェア内で処理がロックしユーザプログラムへの復帰が出来ません。 上記対策は上位アプリケーションによるバスラインの監視及びロック状 態からの復帰処理（ウォッチドックタイマ等）での対応をご推奨申し上げます。

- 10 ビットアドレスを持つスレーブユニットとの通信は対応していません。

- C- BUS、M3L- BUS 等の I²C バス互換プロトコルとの混在は対応していません。

- リスタート条件を用いて、スレーブの切り替えを行う通信フォーマットをバス上に流さないで下さい。(通信に異常をきたします。)

4.2 機能説明

4.2.1 アドレス

<マスタ装置>
７ビットアドレスを持つスレーブ装置との送受信

<スレーブ装置>
７ ビットアドレスを持つ
注：特別なアドレス（ジェネラルコール等のアドレス）に対する送受信は対応していません。

4.2.2 転送速度

転送速度は、0～100Kbps です。よって、高速モードのマスタとの通信はできません。なお本プログラムでは、スタートコンディションの送出時に SCL 同期化機能が有効になるまで、15us のソフトウエアウエイトを挿入しています。これは 100Kbps での通信を想定した値であり、100Kbps 以外で使用する場合は、挿入するソフトウエアウエイトの時間を変更してください。

4.2.3 転送データ長

<マスタ装置>
1 ～256 バイトのデータ送受信が可能です。

<スレーブ装置>
24LC01(128 バイトの I²C バス対応 E₂PROM)と同様の動作をします。ただし、ソフト I²C バスは自装置のアドレスユーザのプログラムで任意に設定できます。この機能は、主にバスに接続されている他のマスタの為にある機能であるため、ステータスは設けていません。
また、自装置からこの領域に書き込み、あるいは読み込みを行う場合にバスを介さずに行うことができるというメリットもあります。

4.2.4 マルチマスタ

I²C バスに接続している、複数の装置が他の装置へのデータ伝送を行うことができます。ただし、マスタとして複合フォーマットの使用はできません。(リスタート条件を発生できないため)このため、同一のソフト I²C バスでバス制御する M16C/62 同士の通信で、E₂PROM ランダムアクセスリードが行えません。

4.3 ハードウェア説明

I²C バスプロトコルを実現するために、M16C/62 の UART2+内蔵簡易 I²C バス H/W のみを使用します。プルアップ抵抗は、システムに合わせて適切に選定してください。

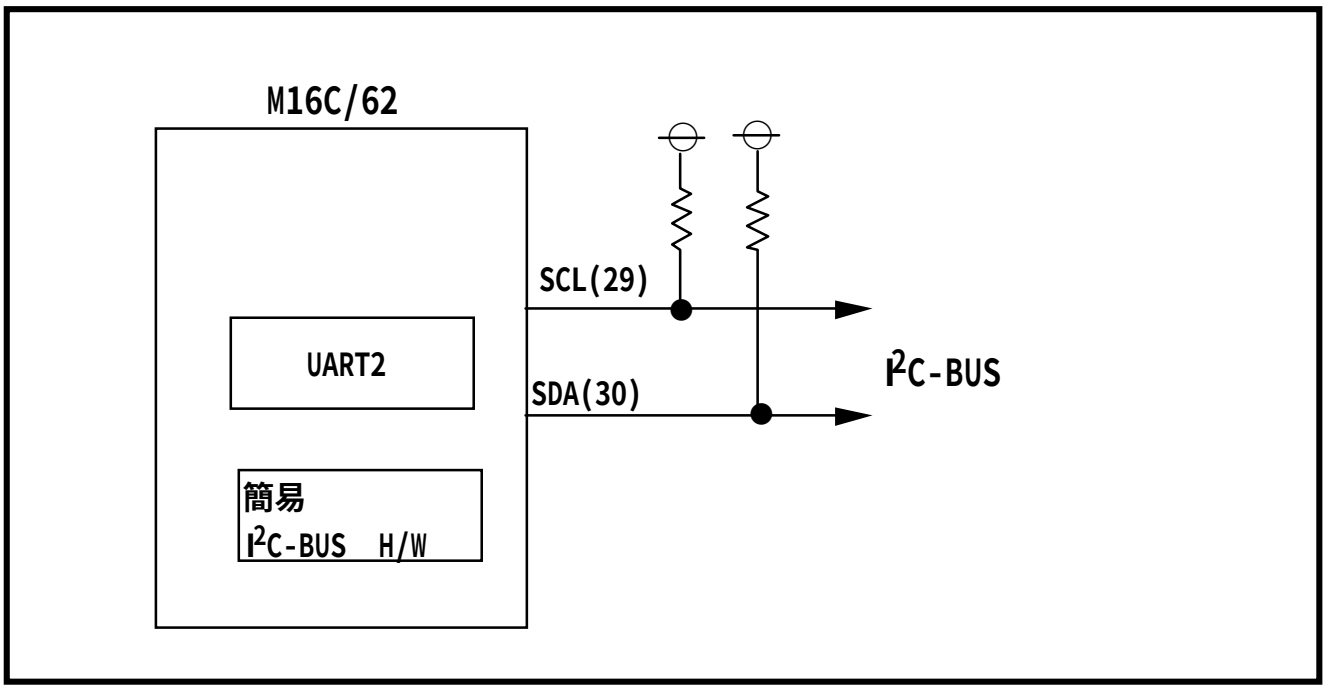


図 4.1 システム構成

4.4 使用方法

4.4.1 組み込み方法

［ncrta30］

① near 領域の最後のセクションエントリとして以下の記述を加えてください。これにより、ソフト I²C バスで使用する RAM 領域 14 バイトを確保する場所を確定します 。

```
.                    section    iicbus,data,align
```

② 以下の割込ベクタ追加してください。

```
                ソフトウェア割込番号 10（バス衝突検出割込）
.glb                 stsp_int
.lword               stsp_int

                ソ フトウェア割込番号 16(UART2  受信割込)
.glb                 u2rcv_int
.lword               u2rcv_int
```

［i2cbus.a30］

ソフト I²C バスで使用する最大の割込レベルを設定します。
i2cbus.a30 の 2 行目の

```
IICIPL               .equ       X
```

の X の部分に、2 ～7 の値を記述してください。ソフト I²C バスでは、X-1～X までの IPL を使用します。X の値は通信速度と実行処理時間に大きく関わってきますので、適切な値を設定してください。通信速度と割込禁止時間はトレードオフの関係にあります。
すなわち、X の値が大きいほど通信速度が早くなる可能性がありますが、割込禁止時間は長くなる可能性があります。

本プログラムでは、UART2 受信割り込みとバス衝突検出割り込みを使用しており、二つの割り込みレベルの関係は、

UART2 受信割り込みレベル ＜ バス衝突検出割り込みレベル

となっています。両割り込み優先レベルが同一レベル、かつソフトウエアで割り込みを禁止している状態で、ストップコンディションを検出し、次のフレームの第一バイトを受信することによりバス衝突検出割り込みと UART2 受信割り込みの二つの割り込み要求が発生した場合は、以下の動作を行います。

上記状態から割り込みを許可状態にした場合、ハードウエアの優先順位から UART2 受信割り込みが優先的に実行され、その後バス衝突検出割り込み処理が実行されます。この場合、前回受信していた（バス衝突検出で終了した）フレームが正常に受信されたか否か判断が難しくなります。

以上の理由から、本プログラムでは UART2 受信割り込みレベルをバス衝突割り込みレベルより 1 レベル低く設定し、バス衝突検出割り込み処理を優先的に実行しています。

[アクセス禁止レジスタ]
以下に示すレジスタを変更しないで下さい。

表 4.1　**アクセス禁止レジスタ**

アドレス	レジスタ名	7	6	5	4	3	2	1	0
04A 16	BUS衝突検出割込制御レジスタ	×	×	×	×	×	×	×	×
04F 16	UART2送信割込制御レジスタ	×	×	×	×	×	×	×	×
050 16	UART2受信割込制御レジスタ	×	×	×	×	×	×	×	×
376 16	UART2特殊モードレジスタ2	×	×	×	×	×	×	×	×
377 16	UART2特殊モードレジスタ	×	×	×	×	×	×	×	×
378 16	UART2送受信モードレジスタ	×	×	×	×	×	×	×	×
379 16	UART2転送速度レジスタ	×	×	×	×	×	×	×	×
37A 16	UART2送信バッファレジスタ	×	×	×	×	×	×	×	×
37B 16		×	×	×	×	×	×	×	×
37C 16	UART2送受信制御レジスタ0	×	×	×	×	×	×	×	×
37D 16	UART2送受信制御レジスタ1	×	×	×	×	×	×	×	×
37E 16	UART2受信バッファ	×	×	×	×	×	×	×	×
37F 16		×	×	×	×	×	×	×	×
3ED 16	ポート7	○	○	○	○	○	○	×	×
3EF 16	ポート7方向レジスタ	○	○	○	○	○	○	×	×

注)アクセス禁止ビットと許可ビットが混在するレジスタにデータ書き込みを行う場合、bset、bclr 命令等を使用し 1 命令で書込み動作が完了する方法で行ってください。

4.4.2　使用メモリ

RAM サイズ	データ	14 バイト
	スタック	9 バイト
	割込スタック	16 バイト
ROM サイズ		1,239 バイト

4.4.2 関数

○ void iic_ini(unsigned char ID,unsigned char *E2PROM)

機能：I²C バスで送受信するための初期化処理を行います。この処理が終了し、割込許可の状態であればスレーブ装置として動作します。また、以下に述べるマスタ送受信を開始する関数をコールすることで、マスタ装置として動作するようになります。

スタック使用量：５バイト

引数：ID … 自己アドレスを設定してください。
自己アドレスは、特殊アドレス以外のアドレスを下位 7 ビットに設定してください。

*E2PROM … E₂PROM のように使用する RAM の先頭番地を設定します。ユーザプログラムで、128Byte の大きさのグローバル変数として、unsigned char 型の配列を用意し、その先頭番地を設定してください。

戻り値：なし

その他：実行している間割込禁止状態となります。

○ unsigned char iic__stop(void)

機能：I²C バスの送受信機能を停止します。スレーブ送受信中はこの関数により、I²C バスの送受信機能を停止することができるが、マスタ送受信中は、送受信機能を停止することはできません。この関数で送受信機能を停止したのち、I²C バス通信動作を行いませんので、マスタ送受信を開始する関数をコールしても送信動作は開始されません。再び送受信機能を生かす場合には、iic_ini をもう一度コールする必要があります。

スタック使用量：５バイト

引数：なし

戻り値：0 …I²C バスの送受信機能を停止できました。
1 …自装置がマスタであるため I²C バスの送受信機能を停止できません。

その他：実行している間割込禁止状態になります。

○ unsigned char iic__mr_start(unsigned char MR_LNG,
unsigned char *MR_DATA,
unsigned char MR_SLAVE)
機能：マスタ受信を開始します。
この関数を使用するには、iic_ini により I²C バスを使用できる状態にしておく必要があります。
スタック使用量：9 バイト
引数：MR_LNG … マスタ受信するデータ長を指定します。
(ただし、0 は 256 バイトを意味します。)
*MR_DATA … マスタ受信したデータを格納する先頭アドレスを指定します。
MR_SLAVE … データを受信したい装置を指定します。
下位 7 ビットにスレーブ装置のアドレスを指定してください。
上位 1 ビットは反映されません。
戻り値： 0 …マスタ受信が開始されました。
1 …バスビジーのためマスタ受信は開始されませんでした。
(リトライは行いません。)
その他：実行している期間割込禁止状態となります。

○ unsigned char iic__mw_start(unsigned char MW_LNG,
unsigned char *MW_DATA,
unsigned char MW_SLAVE)
機能：マスタ送信を開始します。
この関数を使用するには、iic_ini により I²C バスを使用できる状態にしておく必要があります。
スタック使用量：9 バイト
引数：MW_LNG … マスタ送信するデータ数を指定します。
(ただし、0 は 256Byte を意味します。)
*MW_DATA … マスタ送信するデータの先頭番地を指定します。
MW_SLAVE … データを送信したい装置を指定します。
下位 7 ビットにスレーブ装置のアドレスを指定してください。
上位 1 ビットは反映されません。
戻り値：0 … マスタ送信が開始されました。(リトライは行いません。)
その他：実行している期間割込禁止状態となります。

[ユーザが作成するソフト I²C バス関数]
ソフト I²C バスでは、送受信のステータスと、そのデータ数を引数とする以下の関数を呼び出します。
この関数は、ソフト I²C バスに対してユーザが提供しなければなりません。

○ void iic__mw_end(unsigned char MW_STATUS,unsigned char MW_LNG)
機能：マスタ送信の終了状態を以下に示す２つの引数によりユーザに知らせます。
スタック使用量：16 バイト（割込スタック）+関数内 auto 変数
引数：MW_STATUS … マスタ送信の終了状態を示します。
０ ＝ 正常にマスタ送信を終了しました。
１ ＝ 第一バイトでスレーブが NACK を返しました。
２ ＝ データ領域でスレーブが NACK を返しました。
３ ＝ ＢＵＳ 競合検出しマスタ動作を終了しました。
４ ＝ 不正なスタートコンディションを検出しました。
５ ＝ 不正なストップコンディションを検出しました。
MW_LNG … マスタ受信データ数を示します。
戻り値：なし
その他：ソフト I²C バスの割込処理の中からコールしています。

○ void iic__mr_end(unsigned char MR_STATUS,unsigned char MR_LNG)
機能： マスタ送信の終了状態を以下に示す２つの引数によりユーザに知らせます
スタック使用量：16 バイト（割込スタック）+関数内 auto 変数
引数：MR_STATUS …マスタ受信の終了状態を示します。
０ ＝ 正常にマスタ受信を終了しました。
１ ＝ 第一バイトでスレーブが NACK を返しました。
３ ＝ ＢＵＳ 競合検出しマスタ動作を終了しました。
４ ＝ 不正なスタートコンディションを検出しました。
５ ＝ 不正なストップコンディションを検出しました。
MW_LNG … マスタ受信データ数を示します。
戻り値：なし
その他：ソフト I²C バスの割込処理の中からコールしています。

4.4.4 通信方法

4.4.4.1 準備

I²C バスの通信を行うためには、以下の例のようにイニシャルルーチンなどのプログラム動作初期段階で iic_ini をコールする必要があります。このとき、Ｉフラグは 0 でも 1 でもかまいませんが、iic_ini を実行している最中は割込禁止状態になります。iic_ini をコールするときには、2 つの引数を渡します。第 1 引数は、自己アドレスを示し、以降このアドレスで他のマスタからスレーブ指定を受けます。自己アドレスには、機能アドレスを指定しないでください。第 2 引数は、I²C バスのスレーブ送受信に用いる RAM 領域の先頭アドレスを指定します。よって、iic_ini をコールする前にこの領域を確保する必要があります。この領域は、near 属性の静的変数領域に 128 バイト確保してください。

例)

```
// グローバル変数宣言
:
unsigned char iic_ram[ 128] ;          // I²C のスレーブ送受信に用いる RAM 領域
                                       // 静的変数になるようにグローバル変数
:
(システム初期化処理)
iic_ini(0x54,iic_ram);                 // I²C-BUS 初期化
                                       // 自己アドレス 1010100b
                                       // iic_ram の先頭アドレス
asm("fset I");                         // Ｉ フラグセット
```

4.4.4.2 マスタ通信

(1)マスタ送信

マスタ送信を開始するには、iic_mw_start をコールします。iic_mw_start をコールするときには、3 つの引数を渡します。第１引数は、送信データ長を指定します。０を指定すると最大送信データ長である、256 バイトを送信します。

第２引数は、送信データ格納先先頭アドレスを指定します。この、送信データ格納先は、マスタ送信が終了するまでに解放されなければ near 属性の RAM 領域のどこにでも配置できます。

第３引数は、送信相手のアドレスを指定します。送信相手のアドレスには、機能アドレスを指定しないでください。また、iic_mw_start は戻り値をもっています。マスタ送信が開始された場合 0、開始されなかった場合 1 を返します。以下の例では、55_{16} というアドレスを持つスレーブ装置に iic_ram から５バイトのデータを送信します。

例)

```
if (iic_mw_start(0x05,iic_ram,0x55)!=0 ){
                                                    // マスタ送信失敗確認処理
}
else{
                                                    // マスタ送信開始確認処理
}
```

マスタ送信が終了すると、ソフト I²C バスは iic_mw_end をコールします。この関数は、ユーザが作成する必要があります。ソフト I²C バスが iic_mw_end をコールするとき、２つの引数渡します。第１引数は、マスタ送信終了ステータスを示しています。ステータスの内容は 4.4.3 項に示してあります

第２引数は、実際に送信したデータ数を示します。

次に iic_mw_end の例を示します。

```
// プロトタイプ
void iic_mw_end(unsigned char,unsigned char);

// マスタ送信終了関数
void iic_mw_end(unsigned char status,unsigned char length){
        switch (status){
                case 0:
                        // 正常終了
                        break;
                case 1:
                        // 第１バイトに NACK
                        break;
                case 2:
                        // 第２バイト以降に NACK
                        break;
                case 3:
                        // ＢＵＳ競合負け
                        break;
                case 4:
                        // 不正スタート条件
                        break;
                case 5:
                        // 不正ストップ条件
                        default:
                        break;
                }
}
```

(2)マスタ受信
　マスタ送信を開始するには、iic_mr_start をコールします。iic_mr_start をコールするときには、3 つの引数を渡します。第 1 引数は、受信データ長を指定します。0 を指定すると最大受信データ長である、256 バイトを受信します。
　第 2 引数は、受信データ格納先先頭アドレスを指定します。この、送信データ格納先は、マスタ受信が終了するまでに解放されなければ near 属性の RAM 領域のどこにでも配置できます。
　第 3 引数は、受信相手のアドレスを指定します。受信相手のアドレスには、機能アドレスを指定しないでください。また、iic_mr_start は戻り値をもっています。マスタ受信が開始された場合 0、開始されなかった場合 1 を返します。

　以下の例では、55_{16} というアドレスを持つスレーブ装置から iic_ram へ 5 バイトのデータを受信します。

例)

```
if (iic_mr_start(0x05,iic_ram,0x55)!=0 ){
        // マスタ受信失敗確認処理
}
else{
        // マスタ受信開始確認処理
}
```

マスタ受信が終了すると、ソフト I²C バスは iic_mr_end をコールします。この関数は、ユーザが作成する必要があります。ソフト I²C バスが iic_mr_end をコールするとき、２つの引数を渡します。

第１引数は、マスタ受信終了ステータスを示しています。ステータスの内容は 4.4.3 項に示してあります

第２引数は、実際に受信したデータ数を示します。

以下に iic_mr_end の例を示します。

```
// プロトタイプ
void iic_mr_end(unsigned char,unsigned char);

// マスタ送信終了関数
void iic_mr_end(unsigned char status,unsigned char length){
  switch (status){
        case 0:
              // 正常終了
              break;
        case 1:
              // 第１バイトに NACK
              break;
        case 3:
              // バス競合負け
              break;
        case 4:
              // 不正スタート条件
              break;
        case 5:
              // 不正ストップ条件
              default:
              break;
        }
  }
```

4.4.4.3 スレーブ通信

ソフト I²C バスは、他のマスタからは 24LC01(E2PROM)と同様に制御することができます。ただし、装置アドレスを(機能アドレスを除き)自由に選べる点と、アドレスインクリメントが 128 バイトリニアになっている点で異なります。

4.5 評価

評価の際は、i2cbus.a30 ファイルの３行目以下を以下のようにし、ロジアナを使用して各モジュールの動作時間を調べることを推奨します。５ ,６行目に未使用ポートを設定し、ユーザプログラムの初期化処理でここに記述したポートを出力に設定してください。(本例では、ポート９)本ソフトウェアの実行時間は、組み込まれるシステムにより変化するので、このような方法で実行時間(割込禁止時間)を調べるようにしてください。

```
DEBUG           .equ       1
.if             DEBUG ==1
dp      .equ    3f1h
ddp     .equ    3f3h
.endif
```

4.6 プログラムリスト

```
;;; ./i2cbus.a30 start
IICIPL .equ   2             ; IPL
DEBUG  .equ   1             ; 各モジュールの実行時間を知りたい場合 1
.if    DEBUG == 1           ;
dp     .equ   3f1h          ; 実行時間計測に使用するポートを設定
ddp    .equ   3f3h          ; デフォルト P9
.endif                      ; 出力設定は main 処理で行う
;;; ***************************************************************************
;;; システム名    :    IIC-BUS F/W   Ver0.03(参考プログラム)
;;; 概要説明      :    M16C/62 改訂 IIC-BUSF/W(クロック 10MHz ノーウェイト)
;;;               :     マルチマスタ
;;;               :     通信速度 100Kbps
;;;               :     機能アドレス及び 10Bit アドレス禁止
;;;               :     インタフェースはＣ言語のみサポート
;;;               :     複合フォーマットはスレーブ時の送受のみサポート
;;;               :     C バス,M3Low 等は接続できない
;;;               :     Sr を発生できない
;;;               :     一切のフェイルセーフなし
;;; 日付          :    １９９９年1月８日(金)
;;; ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名:    i2cbus.a30
;;; ﾏｲｺﾝ型名      :    M3062xMx
;;; 著作権者      :    三菱電機セミコンダクシステム(株)
;;;               :    Copyright 1997 MITSUBISHI ELECTRIC CORPORATION
;;;               :           AND MITSUBISHI ELECTRIC
;;;               :           SEMICONDUCTOR SYSTEMS CORPORATION
;;; ---------------------------------------------------------------------------
;;; IPL SET
STSPIPL       .equ   IICIPL        ; STSP の IPL
S2RIPL .equ   IICIPL-1      ; UART2 受信 IPL
;;; グローバル宣言
       .glb   _iic_ini      ; F/W 初期化関数
       .glb   _iic_stop     ; IIC バス使用不能
       .glb   _iic_mr_start ; マスタ受信開始関数
       .glb   _iic_mw_start ; マスタ送信開始関数
       .glb   $iic_mw_end   ; マスタ送信終了関数(USER が作成する)
       .glb   $iic_mr_end   ; マスタ受信終了関数(USER が作成する)
       .glb   stsp_int      ; スタートストップコンディション検出割り込み
       .glb   u2rcv_int     ; Uart2 受信割込
;;; ---------------------------------------------------------------------------
;;; SFR シンボル
;;; IIC モード
U2SMR2 .equ   376h          ; UART2 特殊モードレジスタ
IICM2  .btequ 0,U2SMR2      ; IIC モード選択ビット 2
CSC    .btequ 1,U2SMR2      ; クロック同期化ビット
SWC    .btequ 2,U2SMR2      ; SCL ウェイト出力ビット
ALS    .btequ 3,U2SMR2      ; SDA 出力停止ビット
STCntl .btequ 4,U2SMR2      ; UART2 初期化ビット
```

```
SWC2    .btequ 5,U2SMR2       ; SCL ウェイト出力ビット 2
SDHI    .btequ 6,U2SMR2       ; SDA 出力禁止ビット
SHTC    .btequ 7,U2SMR2       ; スタート/ストップコンディション条件制御ビット
U2SMR   .equ   377h           ; UART2 特殊モードレジスタ
IICM    .btequ 0,U2SMR              ; IIC モード選択ビット
ABC     .btequ 1,U2SMR              ; アービトレーションロスト検出フラグ制御
BBS     .btequ 2,U2SMR              ; バスビジーフラグ
LSYN    .btequ 3,U2SMR              ; SCL L 同期出力許可ビット
;;; UART2
U2MR    .equ   378h           ; UART2 モードレジスタ
U2BRG   .equ   379h           ; UART2 速度レジスタ
U2TB    .equ   37ah           ; UART2 送信バッファレジスタ
U2C0    .equ   37ch           ; UART2 送受信制御レジスタ 0
U2C1    .equ   37dh           ; UART2 送受信制御レジスタ 1
U2RB    .equ   37eh           ; UART2 受信バッファレジスタ
ABT     .btequ 3,U2RB+1       ; アービトレーションロスト検出フラグ
;;; 割込制御レジスタ
STSPIC  .equ   004ah          ; スタート／ストップコンディション検出割込制御レジスタ
S2RIC   .equ   0050h          ; UART2(8.5)受信割込制御レジスタ
;;; ポート
P7      .equ   3edh           ; ポート 7
SDA     .btequ 0,P7           ; SDA 端子
SCL     .btequ 1,P7           ; SCL 端子
PD7     .equ   3efh           ; ポート 7 方向レジスタ
SDAD    .btequ 0,PD7          ; SDA 方向
SCLD    .btequ 1,PD7          ; SCL 方向
;;; ------------------------------------------------------------------------------
        .section iicbus,data,align
        .align
mw_data:        .blkw  1      ; マスタ送信データ格納アドレス先頭番地
mr_data:        .blkw  1      ; マスタ受信データ格納アドレス先頭番地
slave_ram:      .blkw  1      ; スレーブ送受信データ格納アドレス先頭番地
mw_lng:                .blkb  1      ; マスタ送信データ長
mr_lng:                .blkb  1      ; マスタ受信データ長
md_cnt:                .blkb  1      ; マスタ送受信データカウンタ
sd_p:           .blkb  1      ; スレーブ送受信データカウンタ
mw_target:      .blkb  1      ; マスタ送信スレーブアドレス
mr_target:      .blkb  1      ; マスタ受信スレーブアドレス
id_adr:                .blkb  1      ; 自己アドレス
;;; ------------------------------------------------------------------------------
m_iic:          .blkb  1      ; モード管理
                              ;  00h : スレーブ待機モード
                              ;  02h : マスタ送信モード
                              ;  03h : マスタ受信モード
                              ;  04h : スレーブ送信モード
                              ;  05h : スレーブ受信モード
                              ;  bit0 0:Wite,1:Read
                              ;  bit1 0:NM 1:Master
                              ;  bit2 0:NM 1:Slave
;;; ------------------------------------------------------------------------------
```

```
        .section      program,code,align
;;; ********************************************************************************
;;; 関数機能名   :      I²C ＢＵＳ初期設定
;;; ********************************************************************************
;;; void iic_ini(R0L,A0)
;;;     引数１：自己アドレス
;;;     引数２；スレーブ送受信データ先頭番地
;;;     スタック：５バイト
;;; --------------------------------------------------------------------------------
        .align                  ;
_iic_ini:                       ; IIC バス F/W 初期化開始
.if     DEBUG == 1              ; ---DEBUG---
        bset    0,dp            ; ---DEBUG---
.endif                          ; ---DEBUG---
        pushc   FLG             ; FLG の退避
        fclr    I               ; 割込禁止
;;; for C
        and.b   #7fh,R0H        ; 自己アドレスの設定 --- arg1 ---
        mov.b   R0H,id_adr      ; 自己アドレスの格納
        mov.w   A0,slave_ram    ; スレーブ送受信データ先頭番地 --- arg2 ---
;;; ******** PORT 関連 ********
        and.b   #0fch,PD7       ; SCL,SDA 解放
;;; ******** I2C 関連 ********
        mov.b   #01h,U2SMR      ; L 同期禁止,ABT 検出ビット毎,IIC モード
        mov.b   #0d1h,U2SMR2    ; 8.5 ビット受信割込選択,8.5L ホールド許可,UART2 初期化許可
                                ; クロック同期化禁止,SDA 出力停止禁止,SCL"L"出力禁止
;;; ******** UART 関連 ********
        bset    SDA             ; SDA=H(SI/O 初期値)
        mov.b   #0ah,U2MR       ; 9 ビット SI/O(外部クロック)
        mov.b   #49,U2BRG       ; 100kbps に設定
        mov.b   #90h,U2C0       ; MSB,CTS/RTS 禁止,f1
        mov.b   #05h,U2C1       ; 送信許可,受信許可
;;; ******** 割込関連 ********
        mov.b   #00h,STSPIC     ; ST/SP 検出割込禁止(要求クリア)
        mov.b   #S2RIPL,S2RIC   ; 8.5 ビット受信割込許可
;;; ******** RAM 初期値 ********
        mov.b   #00h,m_iic      ; スレーブ待機モード
        mov.b   #00h,md_cnt     ; マスタ送受信データ数カウンタ初期化
        mov.b   #00h,sd_p       ; スレーブ送受信データカウンタ初期化
        popc    FLG             ; 割込許可
.if     DEBUG == 1              ; ---DEBUG---
        bclr    0,dp            ; ---DEBUG---
.endif                          ; ---DEBUG---
        rts                     ; IIC バス F/W 初期化修了
```

```
;;; ***************************************************************************
;;; 関数機能名　：　　ＩＩＣバスストップ
;;; ***************************************************************************
;;; unsigned char iic_kill(void)
;;;     引数：なし
;;;     戻り値=0：IIC 機能を停止した
;;;     戻り値=1；マスタ送信中のため IIC 機能を停止出来ない
;;;     スタック：8byte
;;; ---------------------------------------------------------------------------
       .align                ;
_iic_stop:                   ; IIC バスストップ開始
.if    DEBUG == 1            ; ---DEBUG---
       bset   0,dp           ; ---DEBUG---
.endif                       ; ---DEBUG---
       pushc  FLG            ; FLG 退避
       fclr   I              ; 割込禁止
       btst   1,m_iic              ; マスタか？
       jz     iic_stop_slave       ;
iic_stop_master:             ;
       mov.b  #01h,R0L       ; ストップ中止 --- return(R0L) ---
       jmp    iic_stop_common       ;
iic_stop_slave:                     ;
       or.b   #03h,P7               ; SDA/SCL HiZ
       and.b  #0fch,PD7      ; SDA/SCL 端子入力
       mov.b  #00h,U2C1      ; UART2 送受信禁止
       mov.b  #00h,U2MR      ; UART2 ポートモード
       mov.b  #00h,U2SMR     ; IIC モード off
       mov.b  #80h,U2SMR2    ; IIC モード off
       mov.b  #00h,STSPIC    ; スタート/ストップ割込禁止
       mov.b  #00h,S2RIC     ; UART2 受信割込禁止
       mov.b  #00h,R0L       ; ストップ完了 --- return(R0L) ---
iic_stop_common:             ; 共通処理
       popc   FLG            ; FLG 復旧
.if    DEBUG == 1            ; ---DEBUG---
       bclr   0,dp           ; ---DEBUG---
.endif                       ; ---DEBUG---
       rts                   ; IIC バスストップ終了
```

```
;;; ********************************************************************************
;;; 関数機能名　:　　マスタ送信開始
;;; ********************************************************************************
;;; unsigned char iic_mw_start(R0H,A0,R0L)
;;;    引数１ ： 送信データ長
;;;    引数２ ： 送信データ格納先先頭アドレス
;;;    引数３ ： ターゲットアドレス
;;;    戻り値=0：マスタ送信が開始された
;;;    戻り値=1：バスビジーの為マスタ送信が開始されなかった
;;;    スタック：8byte
;;; --------------------------------------------------------------------------------
       .align               ;
_iic_mw_start:                      ; マスタ送信開始処理開始
.if    DEBUG == 1           ; --- DEBUG ---
       bset   1,dp          ; --- DEBUG ---
.endif                      ; --- DEBUG ---
       pushc  FLG           ; FLG 退避
       fclr   I             ; 割込禁止
;;; ---C 言語インタフェース---
       mov.b  R0H,mw_lng    ; マスタ送信データ長のセット --- arg1 ---
       mov.w  A0,mw_data    ; マスタ送信データ先頭アドレス --- arg2 ---
       fclr   C             ; C フラグクリア(W 準備) --- arg3 ---
       rolc.b R0L           ; W フラグ書き込み
       mov.b  R0L,mw_target ; マスタ送信ターゲットアドレス
;;; バスが空いているか確認
       btst   BBS           ; バスビジーフラグ確認
       jc     m_Ereturn     ; エラーリターン処理
       mov.b  #02h,m_iic    ; マスタ送信モード
       jmp    m_start              ; マスタ開始処理
;;; ********************************************************************************
;;; 関数機能名　:　　マスタ受信開始
;;; ********************************************************************************
;;; unsigned char iic_mr_start(R0H,R2,R0L)
;;;    引数１ ： 受信データ長
;;;    引数２ ： 受信データ格納先先頭アドレス
;;;    引数３ ： ターゲットアドレス
;;;    戻り値=0：マスタ受信が開始された
;;;    戻り値=1：バスビジーの為マスタ受信が開始されなかった
;;;    スタック：8byte
;;; --------------------------------------------------------------------------------
       .align               ;
_iic_mr_start:                      ; マスタ受信開始
.if    DEBUG == 1           ; --- DEBUG ---
       bset   2,dp          ; --- DEBUG ---
.endif                      ; --- DEBUG ---
       pushc  FLG           ; FLG 退避
       fclr   I             ; 割込禁止
;;; ---C 言語インタフェース---
       dec.b  R0H           ; NACK を返すデータ長 --- arg1 ---
       mov.b  R0H,mr_lng    ; 比較を簡易にするため-1
```

```
       mov.w  A0,mr_data    ; マスタ受信データ格納先先頭アドレス --- arg2 ---
       fset   C             ; C フラグセット(R 準備) --- arg3 ---
       rolc.b R0L           ; R フラグ書き込み
       mov.b  R0L,mr_target ; マスタ受信ターゲットアドレス
;;; バスが空いているか確認
       btst   BBS           ; バスビジーフラグチェック
       jc     m_Ereturn     ; エラーリターン処理
       mov.b  #03h,m_iic    ; マスタ受信モード
;;; ------------------------------------------------------------------------
;;;  マスタ送受信関数共通処理
;;; ------------------------------------------------------------------------
       .align               ;
m_start:                    ; マスタ開始
;;; バスアイドル送信処理続行
       bclr   ABT           ; ABT フラグクリア
       mov.b  #01h,R0H      ; nack 出力データ
       bclr   4,U2SMR             ;
       mov.b  #8dh,U2SMR2   ; UART2 初期化禁止,クロック同期化許可,SDA 出力停止許可
       mov.b  #00h,U2MR     ; SCL,SDA ポート制御
       bclr   SDA           ; SDA ラッチ L(入力なので出力されない)
       mov.b  #02h,U2MR     ; SI/O 制御開始
                            ; (ラッチ L なのでスタートコンディション発生)
       mov.w  R0,U2TB             ; 第 1 バイト送信
       bset   CSC           ; クロック同期化開始
       jsr    wait_4us      ; ソフトウェイト(4μs)
       bset   SWC2          ; 強制 L 出力(他のユニットの SCL 立ち上がり回避)
       jsr    wait_5us      ; ソフトウェイト(15μs)
       jsr    wait_5us      ; ソフトウェイト(15μs)
       jsr    wait_5us      ; ソフトウェイト(15μs)
       bclr   SWC2          ; SCL 解放
;;; for C
       mov.b  #00h,R0L      ; マスタ送信正常開始 --- return(R0L) ---
       jmp    m_start_end   ;
m_Ereturn:                  ; バスビジー時
       mov.b  #01h,R0L      ; マスタバスビジー中止 --- return(R0L) ---
m_start_end:                ; 終了処理
       popc   FLG           ; フラグの復旧
.if    DEBUG == 1           ; --- DEBUG ---
       mov.b  #00h,dp             ; --- DEBUG ---
.endif                      ; --- DEBUG ---
       rts                  ; マスタ終了
;;; ************************************************************************
;;; 関数機能名  :       ｽﾀｰﾄ/ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ検出割込処理
;;; ************************************************************************
;;; スタック：バイト
;;; ------------------------------------------------------------------------
       .align               ;
stsp_int:                   ; ST/SP 検出処理開始
.if    DEBUG == 1           ; --- DEBUG ---
       bset   5,dp          ; --- DEBUG ---
.endif                      ; --- DEBUG ---
```

```
       push.b R1L            ; レジスタ退避
       or.b   #S2RIPL,S2RIC  ; UART2 受信割込許可
       btst   BBS            ; バスビジーフラグの確認
       jz     stop_int       ; バスアイドルへ
;;; ------------------------------------------------------------------------
;;;    不正スタートコンディション or リスタートコンディション処理
;;; ------------------------------------------------------------------------
start_int:                   ; バスビジー
       mov.b  #08h,STSPIC    ; スタート割込要求(第１バイト)フラグセット
       btst   1,m_iic              ; マスタ？
       jz     stsp_int_end   ;
start_int10:                 ;
;;; ------ 不正なスタートコンディション ------
;;; 強制的にストップコンディションを発生させてマスタを終了する
       jsr    make_stop      ; ストップコンディション発生
       mov.b  #04h,R1L       ; 不正なスタートコンディション --- arg1 ---
       jmp    stsp_common_master
;;; ------------------------------------------------------------------------
;;; サブルーチン機能名：ストップコンディション処理
;;; ------------------------------------------------------------------------
       .align                ;
stop_int:                    ; ストップコンディション
       mov.b  #00h,STSPIC    ; STSP 割込禁止
       mov.b  #0d1h,U2SMR2   ;
       cmp.b  #00h,m_iic     ; スレーブ待機モード？
       jeq    stsp_int_end   ;
stop_int10:                  ;
;;; ---------------------------
       btst   1,m_iic              ; マスタ？
       jz     stsp_int_end   ;
;;; ------ 不正なストップコンディション ------
;;; 強制的なストップコンディション検出によるマスタ終了
       mov.b  #05h,R1L       ; 不正なストップコンディション検出で終了 --- arg1 ---
;;; ------------------------------------------------------------------------
;;;    マスタ共通処理
;;; ------------------------------------------------------------------------
stsp_common_master:          ;
       push.b md_cnt         ; マスタ送受信データ数 --- arg2 ---
       ;; 飛先関数決定
       btst   0,m_iic              ; R or W
       jc     stsp_common_master10
       ;; W
       jsr    $iic_mw_end    ; マスタ送信終了関数呼び出し
       jmp    stsp_common_master_end
       ;; R
       .align                ;
stsp_common_master10:               ;
       jsr    $iic_mr_end    ; マスタ受信終了関数呼び出し
stsp_common_master_end:             ;
       pop.b  md_cnt         ; 引数の解放
       mov.b  #00h,md_cnt    ; マスタ送受信カウンタ初期化
```

```
;;; ------------------------------------------------------------------------
;;;     スレーブ共通処理
;;; ------------------------------------------------------------------------
stsp_int_end:                ;
;       bclr   ABT           ; 競合検出フラグクリア
        mov.b  #00h,m_iic    ; スレーブ待機モード
        pop.b  R1L           ; レジスタ復旧
.if     DEBUG == 1           ; ---DEBUG---
        bclr   5,dp          ; ---DEBUG---
.endif                       ; ---DEBUG---
        reit                 ; ST/SP 検出処理終了
;;; ************************************************************************
;;; 関数機能名  :      UART2(8.5 ｸﾛｯｸ)受信割込処理
;;; ************************************************************************
        .align               ;
u2rcv_int:                   ; UART2 受信割込処理開始
.if     DEBUG == 1           ; --- DEBUG ---
        bset   6,dp          ; --- DEBUG ---
.endif                       ; --- DEBUG ---
        pushm  A0,R1         ; レジスタ退避
        bclr   6,U2SMR2      ;
        btst   3,STSPIC      ; 第１バイト
        jz     byte_n        ; 第二バイト以降の処理へ
;;; 第一バイト処理分岐選択
        bclr   3,STSPIC      ; ST/SP 割込要求ビットクリア
        or.b   #STSPIPL,STSPIC       ; STSP 割込許可
byte_1:                              ; 第一バイト処理開始
        btst   ABT           ; 競合検出確認
        jc     slave_1               ; スレーブ処理へ移行
        ;; 競合無し
master_1:                    ; 第一バイトマスタ処理開始
        cmp.b  #02h,m_iic    ; マスタ送信?
        jz     ms_snd_1      ; マスタ送信処理へ
        cmp.b  #03h,m_iic    ; マスタ受信？
        jz     ms_rcv_1      ; マスタ受信処理へ
slave_1:                     ;
        jsr    wait_5us      ;
        jsr    wait_5us      ;
        jsr    wait_5us      ;
        jsr    wait_5us      ;
        mov.w  U2RB,R1               ; 第１バイト受信
        and.b  #7fh,R1L      ; MSB は不定値の為ﾏｽｸ
        cmp.b  id_adr,R1L    ; アドレス比較
        jeq    sl_next               ;
        ;; マスタアービトレーションロストチェック
        btst   1,m_iic               ;
        jz     slave_taiki   ; 継続
        jsr    abt_lost      ;
        jmp    slave_taiki   ;
        .align               ;
        ;; スレーブ開始処理
```

```
sl_next:                        ; スレーブ開始処理開始
       bclr   ALS               ; アービトレーション時 SDA 強制 HiZ 禁止
       and.b  #0fch,P7          ; SCL=L,SDA=L(ACK)出力ﾃﾞｰﾀ設定
       or.b   #03h,PD7          ; SDA,SCL 出力設定
       mov.b  #00h,U2MR         ; ﾎﾟｰﾄ制御に切替(ACK 送出)
       btst   8,R1              ; r/w 判定
       jc     sl_snd_1          ; 1:read(ｽﾚｰﾌﾞ送信)
       jmp    sl_rcv_1          ; 0:write(ｽﾚｰﾌﾞ受信)
;;; 第二バイト以降処理分岐選択
byte_n:                                 ; 第二バイト以降処理開始
       cmp.b  #00h,m_iic        ; スレーブ待機モード?
       jeq    u2rcv_int_end     ;
       btst   1,m_iic                   ; マスタ?
       jz     slave_n                   ; スレーブ第二バイト以降処理へ
       ;; マスタ第二バイト以降の処理
master_n:                       ; マスタ第二バイト以降処理開始
       btst   0,m_iic                   ; 送信 or 受信 ?
       jc     ms_rcv_n          ; 0:マスタ受信第二バイト以降処理へ
       jmp    ms_snd_n          ; 1:マスタ送信第二バイト以降処理へ
       ;; スレーブ第二バイト以降の処理
slave_n:                        ; スレーブ第二バイト以降処理開始
       btst   0,m_iic                   ; 送信 or 受信 ?
       jc     sl_rcv_n          ; スレーブ受信第二バイト以降処理へ
       jmp    sl_snd_n          ; スレーブ送信第二バイト以降処理へ
;;; ============================================================================
;;;    マスタ送信（第１バイト目 ACK 判定 & 第２バイト目送信）
;;; ============================================================================
       .align                   ;
ms_snd_1:                       ; マスタ送信第一バイト処理開始
.if    DEBUG == 1               ; --- DEBUG ---
       bset   1,dp              ; --- DEBUG ---
.endif                          ; --- DEBUG ---
       mov.w  mw_data,A0        ; 送信データ先頭番地の取り出し
       mov.b  [A0],R1L          ; 送信データの取り出し
       mov.b  #01h,R1H          ; nack
       mov.b  #83h,U2SMR2       ; L ホールド解除,ALS 禁止
ms_s_l1:                        ; ---SCL 解放確認 wait---
       btst   SCL               ; ---SCL 解放確認 wait---
       jz     ms_s_l1                   ; ---SCL 解放確認 wait---
       btst   SDA               ; ack 判定
       jnz    snd_nack_err_1            ; nack 時ストップコンディション送出
       mov.w  R1,U2TB                   ; 送信データセット
       mov.b  #8fh,U2SMR2       ; 8.5 ｸﾛｯｸ L ﾎｰﾙﾄﾞ許可,ALS 許可
       jsr    wait_2us          ; SCL"H"期間(5μs)
       bset   SWC2              ; 強制"L"出力(他ユニットの SCL 立ち上がり回避)
       bclr   ABT               ; 競合検出フラグクリア(ACK の場合セットされる為必須)
       jsr    wait_5us          ; SI/O 出力開始待ち時間(15μs)
       jsr    wait_5us          ;           〃
       jsr    wait_5us          ;           〃
       bclr   SWC2              ; UART2 SCL 有効(L ﾎｰﾙﾄﾞ解除)
       jmp    u2rcv_int_end     ;
```

```
;;; ===========================================================================
;;;     ﾏｽﾀ送信（第ｎﾊﾞｲﾄ目 ACK 判定 & 第ｎ＋１ﾊﾞｲﾄ目送信）
;;; ===========================================================================
        .align                  ;
ms_snd_n:                       ; マスタ送信第二バイト以降処理開始
.if     DEBUG == 1              ; --- DEBUG ---
        bset    1,dp            ; --- DEBUG ---
.endif                          ; --- DEBUG ---
        btst    ABT             ; 競合負け?
        jz      ms_snd_n10      ;
        jsr     abt_lost        ; 競合負け処理へ
        jmp     slave_taiki     ;
        .align                  ;
ms_snd_n10:                     ;
        ;; マスタ継続
        inc.b   md_cnt          ; カウンタを 1 進める
        mov.b   md_cnt,R1L      ; マスタ送信データカウンタの読み出し
        mov.b   #00h,R1H        ; R1 初期化
        mov.w   mw_data,A0      ; マスタ送信データ先頭番地の取り出し
        add.w   R1,A0           ; マスタ送信データ取り出しアドレス設定
        mov.b   [A0],R1L        ; 送信データ取り出し
        mov.b   #01h,R1H        ; nack セット
        mov.b   #83h,U2SMR2     ; IIC モード 2,クロック同期化
ms_s_ln:                        ; ---SCL 解放確認 wait---
        btst    SCL             ; ---SCL 解放確認 wait---
        jz      ms_s_ln                 ; ---SCL 解放確認 wait---
        btst    SDA             ; ack 判定
        jnz     snd_nack_err_n          ; nack 時ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ発生
        cmp.b   md_cnt,mw_lng   ; ﾃﾞｰﾀ数判定
        jeq     snd_end                 ; 正常終了時ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ発生
        ;; ACK で処理継続
        mov.w   R1,U2TB                 ; 送信ﾃﾞｰﾀ設定
        mov.b   #8fh,U2SMR2     ; 8.5ﾋﾞｯﾄ Lﾎｰﾙﾄﾞ許可,競合時 SDA-Hiz
        jsr     wait_2us        ; SCL"H"期間(5μs)
        bset    SWC2            ; 強制"L"出力(他ﾕﾆｯﾄの SCL 立ち上がり回避)
        bclr    ABT             ; ｱｰﾋﾞﾄﾛｽﾄﾌﾗｸﾞｸﾘｱ(ACK の場合ｾｯﾄされる為必須)
        jsr     wait_5us        ; SI/O 出力開始待ち時間(15μs)
        jsr     wait_5us        ;           〃
        jsr     wait_5us        ;           〃
        bclr    SWC2            ; UART2 SCL 有効(Lﾎｰﾙﾄﾞ解除)
        jmp     u2rcv_int_end   ;
;;; ----------------------------
;;; マスタ送信終了
;;; ----------------------------
        .align                  ;
snd_nack_err_1:                         ; スレーブを指定出きなかった
        mov.b   #01h,R1L        ; マスタ失敗 --- arg 1 ---
        jmp     ms_s_end        ;
;;; ----------------------------
        .align                  ;
snd_nack_err_n:                         ; マスタ送信第二バイト以降処理 NACK 終了
```

```
        mov.b   #02h,R1L        ; マスタ送信失敗 --- arg1 ---
        jmp     ms_s_end        ;
;;; ---------------------------
        .align                  ;
snd_end:                        ;
        mov.b   #00h,R1L        ; マスタ正常終了 --- arg1 ---
ms_s_end:                       ; マスタ送信第二バイト以降処理正常終了
        jsr     make_stop       ; ストップコンディション発生
        mov.b   #00h,m_iic      ; ｽﾚｰﾌﾞ待機ﾓｰﾄﾞ
;;; for C
        push.b  md_cnt          ; 送信データ数 --- arg2 ---
        jsr     $iic_mw_end     ; マスタ終了処理
        pop.b   md_cnt          ; スタック解放
        mov.b   #00h,md_cnt     ; マスタデータカウンタのクリア
        jmp     u2rcv_int_end   ;
;;; ============================================================================
;;;     ﾏｽﾀ受信（第１ﾊﾞｲﾄ目 ACK 判定 & 第２ﾊﾞｲﾄ目受信準備）
;;; ============================================================================
        .align                  ;
ms_rcv_1:                       ; マスタ受信第一バイト処理開始
.if     DEBUG == 1              ; ---DEBUG---
        bset    2,dp            ; ---DEBUG---
.endif                          ; ---DEBUG---
        mov.b   #83h,U2SMR2     ; L ﾎｰﾙﾄﾞ解除,ALS 許可
ms_r_l1:                        ; ---SCL 解放確認 wait---
        btst    SCL             ; ---SCL 解放確認 wait---
        jz      ms_r_l1                 ; ---SCL 解放確認 wait---
        btst    SDA             ; ack 判定
        jnz     rcv_nack_err    ; nack 時ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ送出
        ;; ACK 受信
        mov.w   #0ffh,U2TB      ; 受信用ダミーデータのセット
        bset    SWC             ; 8.5 ｸﾛｯｸ L ﾎｰﾙﾄﾞ許可
        jsr     wait_2us        ; SCL"H"期間(5μs)
        bset    SWC2            ; 強制"L"出力(他ﾕﾆｯﾄの SCL 立ち上がり回避)
        jsr     wait_5us        ; SI/O 出力開始待ち時間(15μs)
        jsr     wait_5us        ;          〃
        jsr     wait_5us        ;          〃
        jsr     wait_5us        ;          〃
        bclr    SWC2            ; UART2 SCL 有効(L ﾎｰﾙﾄﾞ解除)
        jmp     u2rcv_int_end   ;
;;; ============================================================================
;;;     ﾏｽﾀ受信（第ｎﾊﾞｲﾄ目受信 & 第ｎ＋１ﾊﾞｲﾄ目受信準備）
;;; ============================================================================
        .align                  ;
ms_rcv_n:                       ; マスタ受信 n バイト処理開始
.if     DEBUG == 1              ; ---DEBUG---
        bset    2,dp            ; ---DEBUG---
.endif                          ; ---DEBUG---
        ;; 格納先アドレス準備
        mov.b   #00h,R1H        ; R1 初期化
        mov.b   md_cnt,R1L      ; マスタ受信データカウントの読み出し
```

```
        mov.w   mr_data,A0      ; マスタ受信データ格納先先頭アドレス
        add.w   R1,A0           ; マスタ受信データ格納先アドレス
        ;; 受信データ加工
        mov.w   U2RB,R1                 ; UART2 受信バッファ読み取り
        btst    8,R1            ; LSB 判定
        rolc.b  R1L             ; LSB ｼﾌﾄ挿入
        mov.b   R1L,[A0]        ; 受信データ格納
        inc.b   md_cnt          ; 受信データカウントアップ
        cmp.b   mr_lng,md_cnt   ; ﾃﾞｰﾀ数判定
        rolc.b  R1H             ; ack/nack
        mov.b   #0ffh,R1L       ; 次受信用ダミーデータ完成
        bclr    SWC             ; SCL 解放(SI/O)
ms_r_ln:                        ; ---SCL 解放確認 wait---
        btst    SCL             ; ---SCL 解放確認 wait---
        jz      ms_r_ln                 ; ---SCL 解放確認 wait---
        btst    SDA             ; ack 判定(ｱｰﾋﾞﾄﾚｰｼｮﾝ無しの前提)
        jnz     rcv_end                 ; nack 時ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ送出
        mov.w   R1,U2TB                 ; 受信用ダミーデータセット
        bset    SWC             ; 8.5 ｸﾛｯｸ L ﾎｰﾙﾄﾞ許可
        jsr     wait_2us        ; SCL"H"期間(5μs)
        bset    SWC2            ; 強制"L"出力(他ﾕﾆｯﾄの SCL 立ち上がり回避)
        jsr     wait_5us        ; SI/O 出力開始待ち時間(15μs)
        jsr     wait_5us        ;         〃
        jsr     wait_5us        ;         〃
        bclr    SWC2            ; UART2 SCL 有効(L ﾎｰﾙﾄﾞ解除)
        jmp     u2rcv_int_end   ;
;;; ---------------------------
        .align                  ;
;;; マスタ受信終了処理
rcv_nack_err:                   ; スレーブ指定出来なかった
        mov.b   #01h,R1L        ; マスタ受信 NACK 終了 --- arg1 ---
        jmp     ms_r_end        ;
;;; ---------------------------
        .align                  ;
rcv_end:                        ;
        mov.b   #00h,R1L        ; マスタ受信正常終了 --- arg1 ---
ms_r_end:                       ; マスタ受信正常終了処理開始
        jsr     make_stop       ; ストップコンディション発生
        mov.b   #00h,m_iic      ; スレーブ待機モード
;;; for C
        push.b  md_cnt          ; マスタ受信データ数 --- arg2 ---
        mov.b   #00h,R1L        ; マスタ受信正常終了 --- arg1 ---
        jsr     $iic_mr_end     ; マスタ受信終了関数呼び出し
        pop.b   md_cnt          ; スタック解放
        mov.b   #00h,md_cnt     ; マスタデータカウンタクリア
        jmp     u2rcv_int_end   ;
;;; ============================================================================
;;;     ｽﾚｰﾌﾞ受信 (第１ﾊﾞｲﾄ目 ack 返信 & 第２ﾊﾞｲﾄ目受信準備)
;;; ============================================================================
        .align                  ;
sl_rcv_1:                       ; スレーブ受信第一バイト処理開始
```

```
.if    DEBUG == 1           ; ---DEBUG---
       bset   4,dp          ; ---DEBUG---
.endif                      ; ---DEBUG---
       bclr   SWC           ; L ﾎｰﾙﾄﾞ解除(SI/O)
       bclr   SCLD          ; L ﾎｰﾙﾄﾞ解除(ﾎﾟｰﾄ)
sl_r_l1:                    ; ---SCL 解放確認処理---
       btst   SCL           ; ---SCL 解放確認処理---
       jz     sl_r_l1              ; ---SCL 解放確認処理---
       mov.b  #0ah,U2MR     ; 第２ﾊﾞｲﾄ受信待機
       mov.w  #0ffh,U2TB    ; ACK 出力ﾃﾞｰﾀ設定
       bset   SWC           ; 8.5ﾋﾞｯﾄ L ﾎｰﾙﾄﾞ許可
       bclr   SDAD          ; SDA 端子入力設定
       ;; アービトレーションロストチェック
       btst   1,m_iic               ; アービトレーションロスト?
       jz     sl_rcv10      ;
       jsr    abt_lost      ;
sl_rcv10:                   ;
       bset   7,sd_p        ;
       mov.b  #05h,m_iic    ; ｽﾚｰﾌﾞ受信ﾓｰﾄﾞ
       jmp    u2rcv_int_end ;
;;; ============================================================================
;;;    ｽﾚｰﾌﾞ送信（第１ﾊﾞｲﾄ目 ack 返信 & 第２ﾊﾞｲﾄ目送信）
;;; ============================================================================
       .align               ;
sl_snd_1:                   ; スレーブ送信第一バイト処理開始
.if    DEBUG == 1           ; ---DEBUG---
       bset   3,dp          ; ---DEBUG---
.endif                      ; ---DEBUG---
       mov.b  sd_p,R1L      ; スレーブ送信データカウンタの読み出し
       mov.b  #00h,R1H      ; RO 初期化
       mov.w  slave_ram,A0  ; スレーブ送信データデータ先頭番地の取り出し
       add.w  R1,A0         ; スレーブ送信データ取り出しアドレス設定
       mov.b  [A0],R1L      ; 送信データの取り出し
       mov.b  #01h,R1H      ; nack
       inc.b  sd_p          ;
       and.b  #07fh,sd_p    ;
       bclr   SWC           ; L ﾎｰﾙﾄﾞ解除(SI/O)
       bclr   SCLD          ; L ﾎｰﾙﾄﾞ解除(ﾎﾟｰﾄ)
sl_s_l1:                    ; ---SCL 解放確認 wait---
       btst   SCL           ; ---SCL 解放確認 wait---
       jz     sl_s_l1              ; ---SCL 解放確認 wait---
       mov.b  #0ah,U2MR     ; 第２ﾊﾞｲﾄ送信待機
       mov.w  R1,U2TB               ; 送信ﾃﾞｰﾀ設定
       bset   SWC           ; 8.5ﾋﾞｯﾄ L ﾎｰﾙﾄﾞ許可
       bclr   SDAD          ;
       ;; アービトレーションロストチェック
       btst   1,m_iic               ; アービトレーションロスト?
       jz     sl_snd10      ;
       jsr    abt_lost      ;
sl_snd10:                   ;
       mov.b  #04h,m_iic    ; ｽﾚｰﾌﾞ送信ﾓｰﾄﾞ
```

```
        jmp     u2rcv_int_end ;
;;; ================================================================================
;;;     ｽﾚｰﾌﾞ受信（第ｎﾊﾞｲﾄ目受信 & 第ｎ＋１ﾊﾞｲﾄ目受信準備）
;;; ================================================================================
        .align                ;
sl_rcv_n:                     ; スレーブ受信処理開始
.if     DEBUG == 1            ; --- DEBUG ---
        bset    4,dp          ; --- DEBUG ---
.endif                        ; --- DEBUG ---
        btst    SWC           ; 8.5 ビットローホールド機能 ON?
        jz      u2rcv_int_end ;
        ;; 格納先アドレス準備
        mov.b   sd_p,R1L      ; スレーブ受信データカウントの読み出し
        mov.b   #00h,R1H      ; R1 初期化
        mov.w   slave_ram,A0  ; スレーブ受信データ格納先アドレス
        add.w   R1,A0         ; スレーブ受信データ格納先アドレス
        ;; 8.5 ビットローホールド機能 ON
        mov.w   U2RB,R1               ; UART2 受信バッファから受信データの取り出し
        btst    8,R1          ; LSB 判定
        rolc.b  R1L           ; LSB ｼﾌﾄ挿入
;;;
;;;
;;;
        btst    7,sd_p        ;
        jz      sl_rcv_n03    ;
        mov.b   R1L,sd_p      ;
        jmp     sl_rcv_n05    ; データカウンタセット
sl_rcv_n03:                   ;
        mov.b   R1L,[A0]      ; 受信データ取り出し
        inc.b   sd_p          ; 受信データカウントアップ
sl_rcv_n05:                   ;
        and.b   #07fh,sd_p
;;;
;;;
;;;
        bclr    SWC           ; SCL 解放(SI/O)
sl_r_ln:                      ; --SCL 解放確認 wait---
        btst    SCL           ; --SCL 解放確認 wait---
        jz      sl_r_ln             ; --SCL 解放確認 wait---
        mov.b   #0ah,U2MR     ; 第ｎﾊﾞｲﾄ受信待機
        mov.w   #00ffh,U2TB   ; ダミーデータセット(ack)
        bset    SWC           ; 8.5ﾋﾞｯﾄ Lﾎｰﾙﾄﾞ許可
        jmp     u2rcv_int_end ;
;;; ================================================================================
;;;     ｽﾚｰﾌﾞ送信（第ｎﾊﾞｲﾄ目 ACK 判定 & 第ｎ＋１ﾊﾞｲﾄ目送信）
;;; ================================================================================
        .align                ;
sl_snd_n:                     ; スレーブ送信第二バイト以降処理開始
.if     DEBUG == 1            ; --- DEBUG ---
        bset    3,dp          ; --- DEBUG ---
.endif                        ; --- DEBUG ---
```

```
       mov.b   sd_p,R1L       ; スレーブ送信データカウンタの読み出し
       mov.b   #00h,R1H       ; R1 初期化
       mov.w   slave_ram,A0   ; スレーブ送信データデータ先頭番地の取り出し
       add.w   R1,A0          ; スレーブ送信データ取り出しアドレス設定
       mov.b   [A0],R1L       ; 送信データの取り出し
       mov.b   #01h,R1H       ; nack
       bclr    SWC            ; SCL 解放(SI/O)
sl_s_ln:                      ; --SCL 解放確認 wait---
       btst    SCL            ; --SCL 解放確認 wait---
       jz      sl_s_ln                ; --SCL 解放確認 wait---
       btst    SDA            ; ack 判定
       jnz     sl_s_end       ; スレーブ送信終了処理へ
       ;; ACK
       mov.b   #0ah,U2MR      ; 第 n ﾊﾞｲﾄ受信待機
       mov.w   R1,U2TB                ; 送信ﾃﾞｰﾀ設定
       bset    SWC            ; 8.5 ﾋﾞｯﾄ Lﾎｰﾙﾄﾞ許可
       inc.b   sd_p           ; スレーブ送信カウンタを進める
       and.b   #07fh,sd_p     ;
       jmp     u2rcv_int_end  ;
;;; ----------------------------
       .align                 ;
sl_s_end:                     ; スレーブ送信処理終了処理開始
       mov.b   #0ah,U2MR      ; 第 n ﾊﾞｲﾄ受信待機
       mov.b   #0d1h,U2SMR2   ; U2SMR2 初期化
       mov.b   #00h,m_iic     ; スレーブ待機モード
       jmp     u2rcv_int_end  ;
;;; ----------------------------
       .align                 ;
slave_taiki:                  ; スレーブ待機処理開始(スレーブ指定されなかった)
       and.b   #08h,S2RIC     ; 受信割込禁止
       bset    SDHI           ; SDA 強制 HiZ(Nack 出力)
       bclr    SWC            ; Lﾎｰﾙﾄﾞ解除(SI/O)
sl_t_l1:                      ; --SCL 解放確認 wait---
       btst    SCL            ; --SCL 解放確認 wait---
       jz      sl_t_l1                ; --SCL 解放確認 wait---
       bset    SDA            ; SDA'H'ラッチ
       mov.b   #0d1h,U2SMR2   ; U2SMR2 初期化
       mov.b   #0ah,U2MR      ; 9 ﾋﾞｯﾄ SI/O(外部ｸﾛｯｸ)
       mov.b   #00h,m_iic     ; スレーブ待機モード
u2rcv_int_end:                       ;
       popm    A0,R1          ; レジスタ復旧
.if    DEBUG == 1             ; --- DEBUG ---
       mov.b   #00h,dp                ; --- DEBUG ---
.endif                        ; --- DEBUG ---
       reit                   ; スレーブ待機処理終了
;;; ------------------------------------------------------------------------------
;;; abt_lost サブルーチン
;;; ------------------------------------------------------------------------------
;;; アービトレーションロストでマスタ終了
abt_lost:                     ; 競合負け処理開始
.if    DEBUG == 1             ; --- DEBUG ---
```

```
       bset   7,dp           ; --- DEBUG ---
.endif                       ; --- DEBUG ---
       mov.b  #03h,R1L       ; アービトレーションロスト --- arg1 ---
       push.b md_cnt         ; 送受信カウンタ --- arg2 ---
       btst   0,m_iic              ; R or W
       jc     abt_lost10     ;
       ;; W
       jsr    $iic_mw_end    ; マスタ送信終了関数呼び出し
       jmp    abt_lost_end   ;
       .align                ;
abt_lost10:                  ;
       jsr    $iic_mr_end    ; マスタ終了処理
abt_lost_end:                ;
       pop.b  md_cnt         ; スタック解放
       mov.b  #00h,md_cnt    ; マスタデータカウンタのクリア
       rts                   ;
;;; ------------------------------------------------------------------------
;;; make_stop サブルーチン
;;; ------------------------------------------------------------------------
;;; ストップコンディションを発生させる
make_stop:                   ;
       and.b  #0fch,P7       ; SCL,SDA 端子 L ラッチ
       or.b   #03h,PD7       ; SCL,SDA 端子出力開始
       jsr    wait_0us       ; SCL"H"期間(5μs)
       mov.b  #00h,U2MR      ; SCL,SDA ポートモード
       bclr   SCL            ; SCL = L
       bclr   SDA            ; SDA = L
       jsr    wait_5us       ; SCL"L"期間(5μs)
       bclr   SCLD           ; SCL 解放
make_stop_wait:                    ; --- スレーブの LowHold 解除待ち ---
       btst   SCL            ; --- スレーブの LowHold 解除待ち ---
       jz     make_stop_wait       ; --- スレーブの LowHold 解除待ち ---
       jsr    wait_2us       ; ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝｾｯﾄｱｯﾌﾟﾀｲﾑ(4μs)
       bclr   SDAD           ; SDA 解放
       bset   SDA            ; SDA 端子 H ラッチ
       mov.b  #0ah,U2MR      ; 9ﾋﾞｯﾄ SI/O(外部ｸﾛｯｸ)
       mov.b  #0d1h,U2SMR2   ; U2SMR2 初期化
       bclr   ABT            ; アービトレーションロストフラグクリア
       rts                   ;
;;; ------------------------------------------------------------------------
;;; time_wait サブルーチン
;;; ------------------------------------------------------------------------
       .align                ;
wait_5us:                    ; コール 8 サイクル
       .mrepeat 10           ;
       nop                   ; 10 サイクル
       .endr                 ;
wait_4us:                    ; コール 8 サイクル
       .mrepeat 20           ;
       nop                   ; 20 サイクル
       .endr                 ;
```

```
wait_2us:                       ; コール 8 サイクル
       .mrepeat 6               ;
       nop                      ; 6 サイクル
       .endr                    ;
wait_0us:                       ;
       rts                      ; リターン 6 サイクル
;;; ------------------------------------------------------------------------
       .end                     ;
;;;
;;; ./i2cbus.a30 end
;;;
```

安全設計に関するお願い

- 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切な三菱半導体製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について三菱電機が所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、三菱電機は責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、三菱電機は、予告なしに、本資料に記載した製品または仕様を変更することがあります。三菱半導体製品のご購入に当たりましては、事前に三菱電機または特約店へ最新の情報をご確認頂きますとともに、三菱電機半導体情報ホームページ（www.MitsubishiElectric.co.jp/semiconductors）などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、三菱電機はその責任を負いません。
- 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。三菱電機は、適用可否に対する責任は負いません。
- 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、三菱電機または特約店へご照会ください。
- 本資料の転載、複製については、文書による三菱電機の事前の承諾が必要です。
- 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたら三菱電機または特約店までご照会ください。