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「日立製作所」、「日立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日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、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・「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立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。

ルネサステクノロジ ホームページ (http://www.renesas.com)

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

ご注意

安全設計に関するお願い

1. 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

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ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ルネサス テクノロジ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。
2. 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、ルネサス テクノロジは責任を負いません。
3. 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、ルネサス テクノロジは、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。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 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、ルネサス テクノロジはその責任を負いません。
5. 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。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 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、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 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ルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

Application Note

RENESAS

内蔵インタフェース
I²C バスインタフェース編

アプリケーションノート

ルネサスシングルチップマイクロコンピュータ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　2002.01

ご注意

1. 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。
2. 本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。
3. 製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。
4. 弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置， ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。
5. 設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。
   保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。
6. 本製品は耐放射線設計をしておりません。
7. 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。
8. 本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

はじめに

現在、周辺インタフェースは、低コスト化や使い勝手向上のニーズにより、各分野において統一化／標準化が進んでいます。本アプリケーションノートで取り扱う I²C バス*インタフェースも標準インタフェース化されているものの一つで、家電製品の制御用 IC のインタフェース、ノート PC のバッテリパックの制御用インタフェースや PC モニタの制御用インタフェース等に応用されています。

I²C バスはオランダの Philips 社が開発した双方向シリアルバスシステムの規格です。この規格に基づく製品どうしであれば、2 本の配線（クロックライン、データライン）で複数の周辺 IC 間の相互データ通信ができます。

日立オリジナル 8 ビット／16 ビットシングルチップマイクロコンピュータ H8/300 シリーズ、H8/300L シリーズおよび H8S シリーズに内蔵している I²C バスインタフェースは Philips 社の提唱している I²C バスインタフェース方式に準拠しており、サブセット機能を備えています（ただし、使用条件により、一部 I²C バスインタフェースの仕様を満足しない場合がありますのでご注意ください）。

本アプリケーションノートでは、第 1 章と第 2 章で I²C バスの概要と弊社 I²C バスインタフェースモジュールの仕様、機能を概説します。第 3 章では H8/300 シリーズ、H8/300L シリーズによるマルチマスタ構成のシステム例を、第 4 章では H8S シリーズによる I²C バスインタフェースの応用例を紹介しています。

なお、本アプリケーションノートに記載されているハードウェア、ソフトウェア例は動作確認をしておりますが、実際にご使用になる際は、必ず動作確認の上ご使用くださいますようお願い致します。

【注】 I²C **バス：**Inter IC Bus

目次

1. I²C バスの概要

1.1 I²C バスの概要

1.1.1 I²C バスの特長

I²C バスの特徴を以下に示します。

- I²Cバスは、シリアルデータライン（SDA）、シリアルクロックライン（SDA）の2本のバスラインで構成されます。I²Cバス装置の拡張が容易です。
- 装置間にはマスタとスレーブという関係が常に成り立ち、各装置は固有のアドレスを持っています。マスタとなる装置が、最初に通信相手の有する固有アドレスを指定することにより通信のパスが形成され、データ通信が可能となります。
- 任意の装置がマスタになることができます（マルチマスタシステムを構築可能）。そのため、I²Cバスインタフェースは、データ破壊を防ぐためのバス権競合回避のシステムが定義されています。
- データ転送速度は、標準モードで最高100kbps、高速モードで最高400kbpsです（I²Cバス仕様書 Ver2.0では、3.4Mbpsまで定義されています）。
- I²Cバスシステムにおける装置の総数は、システムのバス負荷容量の上限値400pFで決定されます。
- 応用例として SMBus*1、ACCESS. bus*2があります。

【注】 *1 SMBus **は、**Duracell **社と** intel **社が策定したシリアルバスです。**

*2 ACCESS. bus **は、**Digital Equipment **社が策定したシリアルバスです。**

1.1.2 シリアルコミュニケーションインタフェース（SCI）との相違

日立シリアルインタフェースであるシリアルコミュニケーションインタフェース（SCI）との相違点をまとめます。

表 1.1 に示すように、SCI ではデータラインとして、送信用データラインと受信用データラインの 2 本を使用します。データ通信方式は、一般的に 1 対 1 で行われます。

一方、I²C バスはデータライン 1 本で双方向通信します。通信相手先は、マスタとなる装置が通信相手の固有アドレスを指定することで決定されるため、複数の任意の装置とのデータ送受信が可能です。また I²C バスは、バス権の衝突回避メカニズムが定義されているため、任意の装置がマスタとして動作できるマルチマスタシステムに対応します。転送レートは標準モードでは最大 100kbps、高速モードで最大 400kbps となります。

表 1.1 SCI との相違点

	SCI		I²C バス
	クロック同期式	調歩同期式	
使用端子	3 線式	2 線式	2 線式
	送信データ出力	送信データ出力	送受信データ入出力
	受信データ入力	受信データ入力	
	シリアルクロック	シリアルクロック （外部クロック使用時）	シリアルクロック
転送レート	100pbs～4Mbps	110bps～38.4kbps	100kbps（標準モード） 400kbps（高速モード）*
複数の IC との送受信	不可	不可	可能 スレーブをアドレスで制御する

【注】* I²C バス仕様書 Ver.2.0 で定義されている Hs モード（最大転送速度：3.4Mbps）はサポートしていません。

1.1.3 I²C バスインタフェースの接続形式

図 1.1 に I²C バスインタフェースの接続形式を示します。この図のように I²C バスは、クロックライン SCL とデータライン SDA から構成され、それぞれプルアップ抵抗でバス電源 VBB に接続されます。デバイス 1 とデバイス 2 の各 SCL 端子／SDA 端子はそれぞれ SCL ラインと SDA ラインにワイヤード AND 接続されます。

デバイス 1 が SCL ラインを“ Low ”にドライブしているとき、デバイス 2 は SCL ラインの状態をモニタすることにより他のデバイスがバスを使用していることを確認します。またワイヤード AND 接続により、デバイス 1 がバスを使用中で SCL ラインをドライブしていても、デバイス 2 が SCL を“ Low ”にドライブし、デバイス 1 に対して通信動作を“ 待ち ”状態にすることができます（詳細は「**2章　I²C バスインタフェースの機能説明**」を参照ください）。

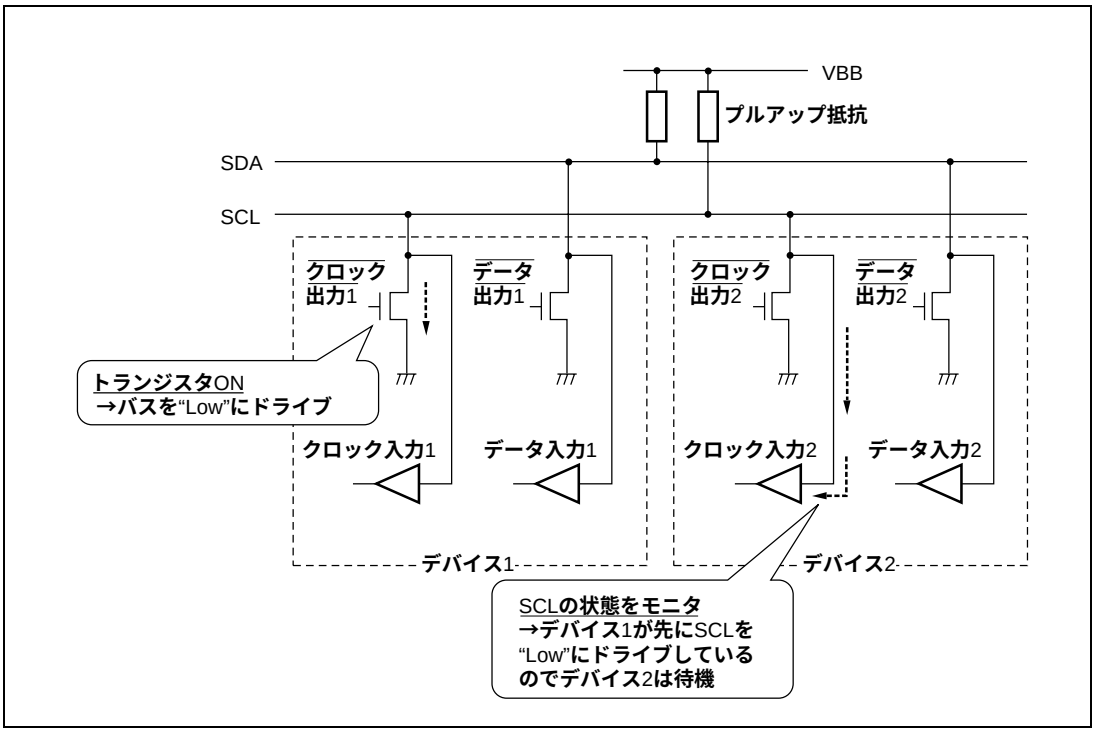


図 1.1　I²C **バスインタフェースの接続形式**

（デバイス 1 **が最初に** SCL **を“** Low **”にドライブした場合）**

1.2 I²C バスを使用したデータ転送方式

1.2.1 I²C バスを使用したデータ転送の基本事項

はじめに I²C バスを使用したデータ転送の基本事項を説明します。

(1) マスタデバイス

マスタデバイスは、データ通信を行うための同期化クロックを生成し、データ通信の開始／停止を示す開始条件／停止条件を発行します。

(2) スレーブデバイス

スレーブデバイスは、マスタデバイス以外の I²C バスデバイスです。

(3) 送信デバイス

送信デバイスとは、データを送信するデバイスです。マスタデバイスとスレーブデバイスの場合があります。

(4) 受信デバイス

受信デバイスとは、データを受信するデバイスです。マスタデバイスとスレーブデバイスの場合があります。

(5) 開始条件

開始条件とは、**図** 1.2 のように SCL ラインが“ High ”のときに、SDA ラインが“ High ”から“ Low ”に変化する動作です。これによりデータ通信動作が開始されます。マスタデバイスが発行します。

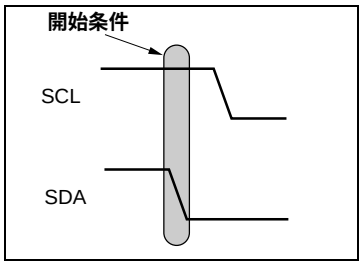


図 1.2 **開始条件**

(6) 停止条件

停止条件とは、**図** 1.3 のように SCL ラインが“ High ”のときに、SDA ラインが“ Low ”から“ High ”に変化する動作です。これによりデータ通信動作が停止されます。マスタデバイスが発行します。

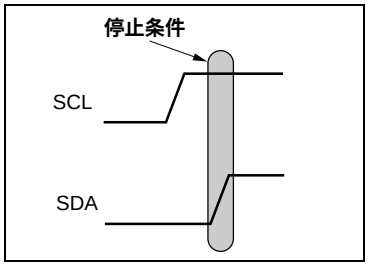


図 1.3 **停止条件**

（7） データの出力タイミング

図 1.4 のようにデータ出力タイミングは、SCL ラインが“ Low ”のとき、SDA ライン上のデータが更新され、SCL ラインが“ High ”のとき SDA ライン上のデータが確定します。SCL ラインが“ High ”のとき SDA ラインが変化するのは前記「開始条件」および「停止条件」のときのみです。

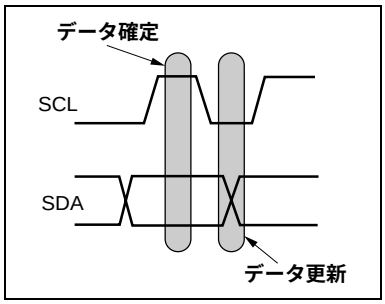


図 1.4　**データ出力タイミング**

（8） マスタ送信動作

マスタ送信動作とは、マスタデバイスが送信デバイスの場合の動作です。開始条件発行後のスレーブアドレスの送信やスレーブデバイスへのコマンドなどの送信をする場合があります。

（9） マスタ受信動作

マスタ受信動作とは、マスタデバイスが受信デバイスの場合の動作です。

（10）スレーブ送信動作

スレーブ送信動作とは、スレーブデバイスが送信デバイスの場合の動作です。

（11）スレーブ受信動作

スレーブ受信動作とは、スレーブデバイスが受信デバイスの場合の動作です。開始条件後のマスタデバイスによるスレーブアドレス送信フレームでは、スレーブデバイスは受信動作となります。

（12）バス解放状態

すべての I²C バスデバイスが通信していない状態です。SCL、SDA ラインとも定常的に“ High ”状態です。

（13）バス占有状態

バス占有状態とは、I²C バスデバイスがデータ通信を行っている状態です。マスタデバイスが停止条件を発行した時点でバス開放状態に戻ります。

（14）データ転送フォーマット

図 1.5 に I²C バスのデータ転送フォーマットを示します。「開始条件」、「停止条件」、SCL クロックはマスタデバイスが生成します。開始条件後の第 1 データはスレーブアドレスとなり、8 ビット目にデータの通信方向を示すビットが付加されます。本ビットは、“ 0 ”のとき、第 2 バイト以降のデータ通信がマスタ送信動作であることを示し、“ 1 ”のときは、第 2 バイト以降のデータ通信はマスタ受信動作をすることを意味します。スレーブアドレスは 7 ビットで定義され[*1]、ユーザが B'0000000～H'1111111 の間で設定しますが、アドレス B'0000000（ゼネラルコールアドレスと呼ぶ[*2]）と一部のアドレスは予約されています。

転送データは 1 バイト（8 ビット）単位になり、9 ビット目に受信デバイスからの確認応答ビット（アクノリッジビット）が付加されます。例えばマスタがスレーブアドレスを送信した場合には、該当するスレーブは 9 クロック目に SDA を“ Low ”にドライブし、マスタにアクノリッジを返します。

1 回の開始条件－停止条件間に転送できるバイトデータ数には制限はありません。データ通信は、「停止条件」をもって終了します。

【注】 *1 I²C **バス仕様書では、**10 **ビットアドレス指定が定義されています。日立の** I²C **バスインタフェースモジュールでは** 10 **ビットアドレス指定をサポートしておりません。**

*2 **ゼネラルコールアドレス** B'0000000 **は、バスに接続されているすべてのスレーブアドレスを指定するときに使用します。**

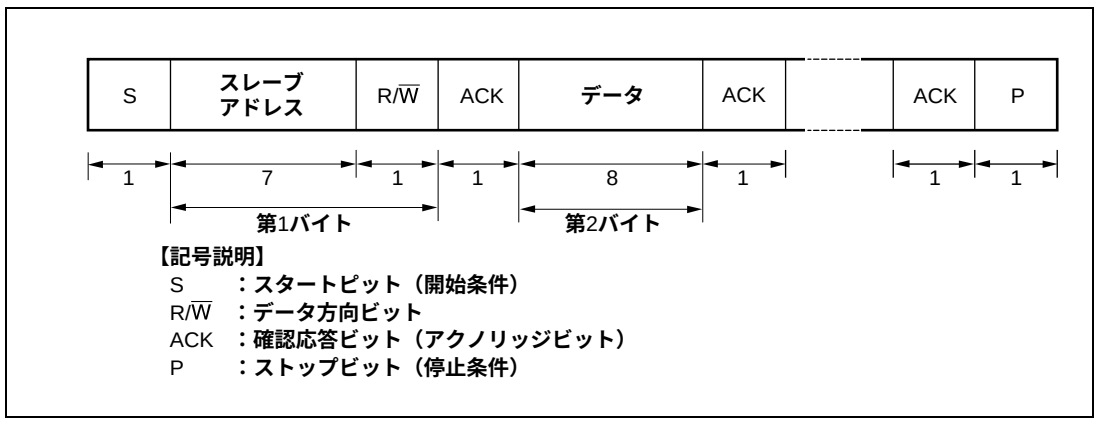


図 1.5 **データ転送フォーマット**

1.2.2 データ転送手順（例：マスタデバイス＝送信デバイス、スレーブデバイス＝受信デバイス）

図 1.6 にマスタデバイスがスレーブデバイスに 1 バイトのデータを送信する場合の例を示します。

まず、マスタデバイスは開始条件を発行し、SCL ラインが“ High ”のときに、SDA ラインを“ High ”から“ Low ”に変化させます。次にマスタは、SCL ライン上にクロックを出力するとともに、SDA ライン上に通信対象となるスレーブのアドレスを出力します。スレーブのアドレスは 7 ビットで定義され、8 ビット目に通信方向を表すビットを付加されます。

マスタデバイスは 9 クロック目に SDA ラインを開放し、スレーブデバイスからのアクノリッジに備えます。スレーブデバイスは、9 クロック目に SDA ラインを“ Low ”にドライブしアクノリッジを返します。

マスタデバイスはスレーブアドレスからのアクノリッジを受信し、次の送信データが準備できるまで、SCL ラインを“ Low ”に保持します。送信データの準備ができたところでマスタデバイスは、SCL ラインにクロックを出力しながら、データを SDA ラインに出力します。前回と同様に 9 クロック目にスレーブデバイスはマスタデバイスにアクノリッジを返し、正常にデータが受信できたことを通知します。マスタデバイスは、スレーブデバイスからのアクノリッジを受け取ると、SCL ラインを“ Low ”に保持します。そして停止条件を発行し、SCL ラインが“ High ”のとき、SDA ラインが“ Low ”から“ High ”に変化させます。

データ通信中、もし、スレーブデバイスが他の処理を行っているため、すぐにデータを受信できない場合は、スレーブデバイス側で SCL ラインを“ Low ”に保持し、マスタデバイスを待ち状態にすることができます。スレーブデバイスが SCL を“ Low ”にドライブできるタイミングは、マスタデバイスが SCL を“ Low ”にドライブしているときです。

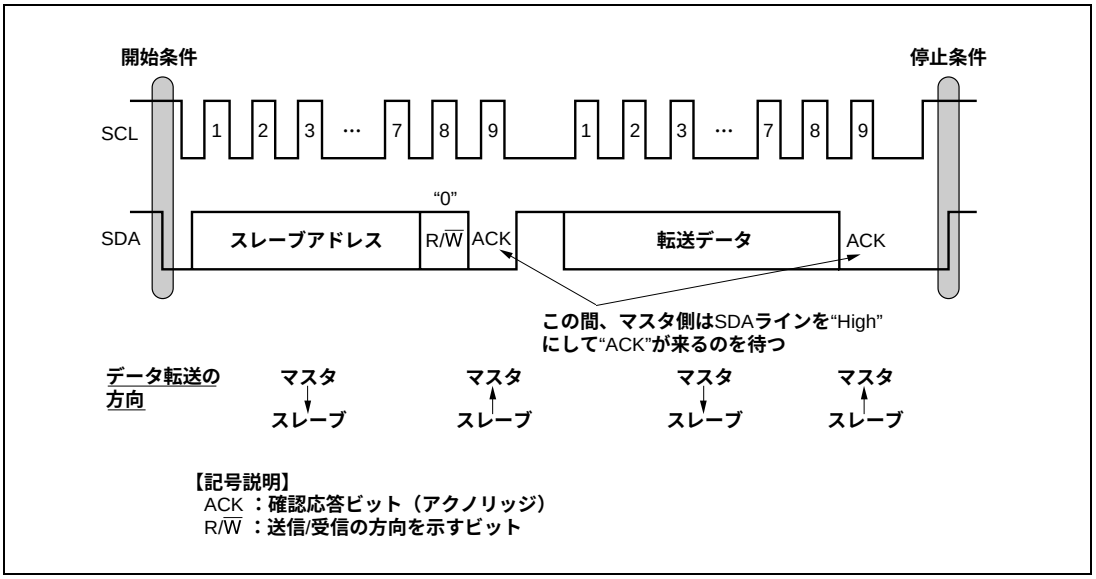


図 1.6　**データ転送フォーマット**

（マスタ＝送信デバイス、スレーブ＝受信デバイスの場合）

1.3 シングルマスタとマルチマスタの構成

1.3.1 シングルマスタ

マスタデバイスは「開始条件」および「停止条件」を発行し、データ通信を管理します。また SCL ライン上にデータを送受信するための同期化クロックやスレーブアドレスを出力します。マスタデバイスが常に固定されている**図** 1.7 のようなシステム構成をシングルマスタ構成といいます。

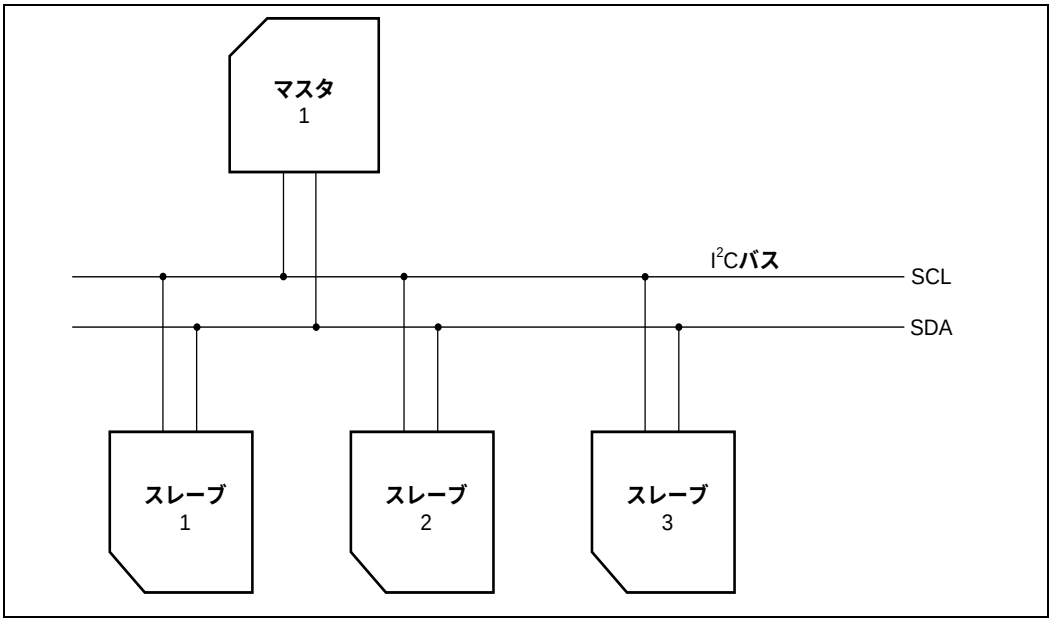


図 1.7　**シングルマスタ構成**

1.3.2 マルチマスタ

図 1.8 のように、1 つのシステム内にマスタと成り得るデバイスが 2 個以上存在する構成をマルチマスタ構成といいます。

マスタデバイスはバスが解放状態のときのみデータ転送を開始することができますがマルチマスタ構成の場合、複数のマスタデバイスが同時にデータ転送を開始しようとする可能性があります。つまり、バス権の衝突が生じます。このため、I²C バスの仕様にはバス権の衝突が生じた場合の通信調整手順が規定されています。詳細は「**1.4 通信調整手順**」を参照してください。

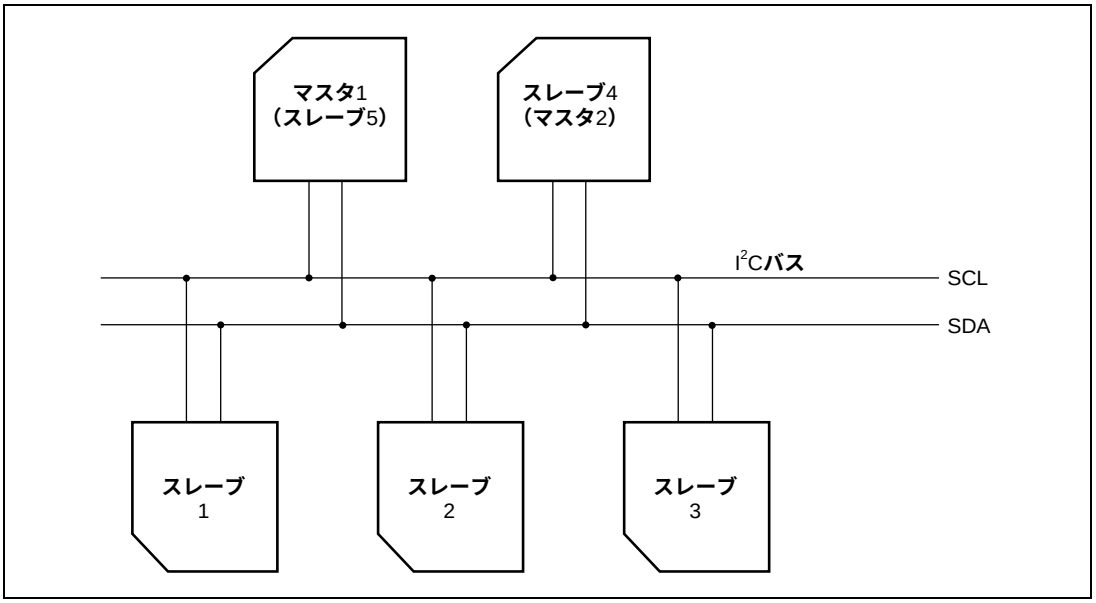


図 1.8　**マルチマスタ構成**

1.4 通信調整手順

I²C バスインタフェースでは、バス権の衝突を回避するための通信調整手順が定義されており、マルチマスタ構成のシステムに対応できます。

マスタデバイスは、バスラインをモニタし、バスが開放されていることを確認し、開始条件を発行します。このとき複数のマスタデバイスによる開始条件発行処理が発生する可能性があります。そこで**図** 1.9 に示すような通信調整手順により、唯一のマスタデバイスが決定されます。

I²C バスでは、SCL ラインが“ High ”期間中に SDA ライン上のデータが確定している必要があります。そこで各デバイスは、開始条件後の SCL ラインの立ち上がりをモニタし、SDA ラインの状態と各デバイス内部のデータ（スレーブアドレス）を比較します。もし、デバイス 1 が SDA を“ High ”に、デバイス 2 が SDA を“ Low ”にしていれば、ワイヤード AND 接続により実際の SDA ラインは“ Low ”になりますので、デバイス 1 は、自分の出そうとしているデータと異なることを確認し、データ出力段をオフにします。本例ではデバイス 2 がマスタデバイスとして動作を継続します（**図** 1.9 参照）。すべてのマスタが同一のスレーブ装置をアドレス指定しようとしている場合にはさらに次の段階に進み、データの比較が行われます。

例えば、**図** 1.10 のように転送データが H'01、H'02 の場合、データ H'01 の方が“ Low ”期間が長いのでデータ H'01 が有効になります。したがって、ゼネラルコールアドレス（H'00）が最優先されます。

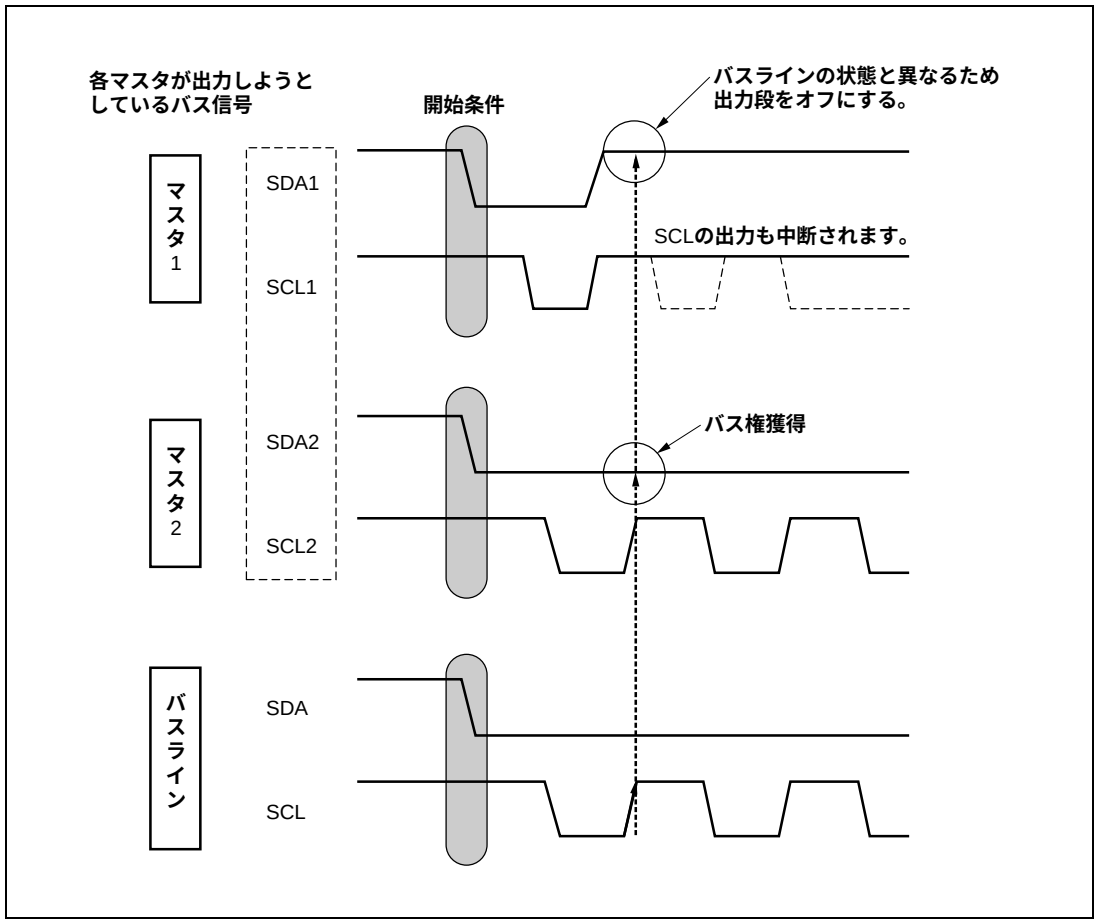


図 1.9　**通信調整手順（バスアービトレーションロストの検出）**

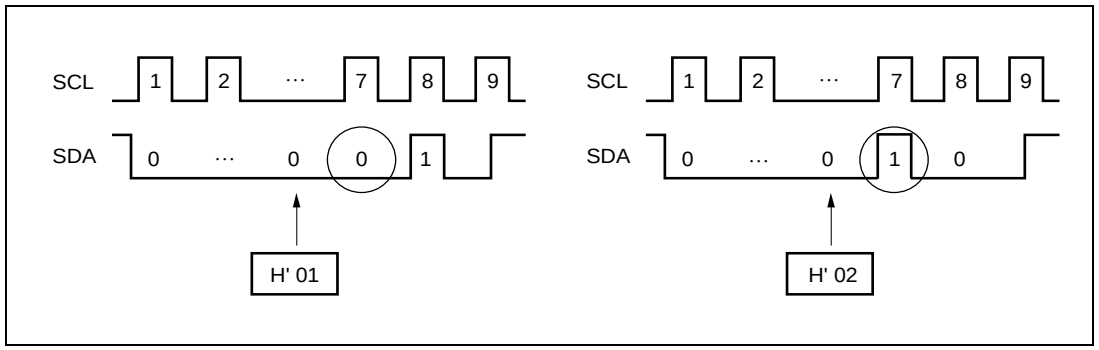


図 1.10　**通信調整の具体例**

2. I²C バスインタフェースの機能説明

2.1 I²C バスインタフェース内蔵製品ラインアップ

日立 I²C バスインタフェースのモジュールは大きく分けて 2 種類あります。

1. H8 系 ：日立が最初に製品化した I²C バスインタフェースモジュール
2. H8S 系：H8 系の機能アップ版

表 2.1 に、日立 I²C バスインタフェース内蔵製品の一覧と I²C バスインタフェースモジュールの対応関係を示します。

表 2.1 I²C **バスインタフェース内蔵製品**

シリーズ名		**品名**	**ピン数**	**チャネル**	MASK*[1]	F-ZTAT™	ZTAT®	I²C **モジュール**
H8/300 **シリーズ**	H8/3217 **シリーズ**	H8/3217	64、80	2ch	○	−	○	H8 **系**
		H8/3216		2ch	○	−	−	
		H8/3214		2ch	○	−	○	
		H8/3212		2ch	○	−	−	
		H8/3202		1ch	○	−	−	
	H8/3337 **シリーズ**	H8/3337Y	80、84	1ch	○	−	○	
		H8/3337YF		1ch	−	○	−	
		H8/3337SF		1ch	−	○	−	
		H8/3336Y		1ch	○	−	−	
		H8/3334Y		1ch	○	−	○	
		H8/3334YF		1ch	−	○	−	
	H8/3437 **シリーズ**	H8/3437	100	1ch	○	−	○	
		H8/3437YF		1ch	−	○	−	
		H8/3437SF		1ch	−	○	−	
		H8/3436		1ch	○	−	−	
		H8/3434		1ch	○	−	○	
		H8/3434F		1ch	−	○	−	

（続く）

【注】 *1 MASK **版はオプションになります。**

*2 **開発中**

表 2.1 I²C バスインタフェース内蔵製品（続き）

シリーズ名		品名	ピン数	チャネル	MASK*1	F-ZTAT™	ZTAT®	I²C モジュール
H8/300 シリーズ	H8/3567 シリーズ	H8/3567	42、44	2ch	○	−	○	H8S 系
		H8/3564		2ch	○	−	−	
		H8/3561		2ch	○	−	−	
		H8/3567U		2ch	○	−	○	
		H8/3564U		2ch	○	−	−	
	H8/3577 シリーズ	H8/3577	64	2ch	○	−	○	
		H8/3574		2ch	○	○	−	
H8/300L シリーズ	H8/3947 シリーズ	H8/3947	100	2ch	○	−	○	H8 系
		H8/3946		2ch	○	−	−	
		H8/3945		2ch	○	−	−	
H8/300H Tiny シリーズ *2	H8/3664 シリーズ	H8/3664	42、64	1ch	○	○	−	H8S 系
H8S シリーズ	H8S/2100 シリーズ	H8S/2127	64、80	2ch	○	−	−	H8S 系
		H8S/2126		2ch	○	−	−	
		H8S/2128F		2ch	−	○	−	
		H8S/2138	80	2ch	○	−	−	
		H8S/2137		2ch	○	−	−	
		H8S/2138F		2ch	−	○	−	
		H8S/2148	100	2ch	○	−	−	
		H8S/2147		2ch	○	−	−	
		H8S/2148F		2ch	−	○	−	
		H8S/2147NF		2ch	−	○	−	
		H8S/2149YVF		2ch	−	○	−	
		H8S/2169YVF	144	2ch	−	○	−	
		H8S/2194	112	1ch	○	−	−	
		H8S/2193		1ch	○	−	−	
		H8S/2192		1ch	○	−	−	
		H8S/2191		1ch	○	−	−	
		H8S/2194F		1ch	−	○	−	
		H8S/2199		2ch	○	−	−	
		H8S/2198		2ch	○	−	−	
		H8S/2197		2ch	○	−	−	

（続く）

【注】 *1 MASK 版はオプションになります。

*2 H8/300H Tiny シリーズ内蔵の I²C バスインタフェースの仕様／使用説明については、別冊でご案内します。

表 2.1 I²C バスインタフェース内蔵製品（続き）

シリーズ名		品名	ピン数	チャネル	MASK*1	F-ZTAT™	ZTAT®	I²C モジュール
H8S シリーズ	H8S/2100 シリーズ	H8S/2196	112	2ch	○	−	−	H8S 系
		H8S/2199F		2ch	−	○	−	
	H8S/2200 シリーズ	H8S/2238	100	2ch	○*2	○*2	−	
		H8S/2236		2ch	○*2	−	−	
		H8S/2258		2ch	○*2	○*2	−	
		H8S/2256		2ch	○*2	−	−	
	H8S/2600 シリーズ	H8S/2633	120、128	2ch	−	○	−	
		H8S/2643*2	144	2ch	−	○	−	

【注】 *1 MASK 版はオプションになります。

*2 開発中

2.2 H8/300、H8/300L シリーズ内蔵 I²C バスインタフェースの仕様［H8 系］

2.2.1 H8/300、H8/300L シリーズ内蔵 I²C バスインタフェースの特長

日立の 8 ビットマイコン H8/300 シリーズ、H8/300L シリーズに内蔵されている、I²C バスインタフェースの主な特長を以下に示します。本モジュールを内蔵する製品群については**表** 2.1 を参照ください。

- 転送データ長：転送ビット数（1～8 ビット）
  ：転送フレーム数（無制限）
- 開始／停止条件の自動生成
- アクノリッジビット自動ロード機能
- ウェイト機能
- 8 種類の内部クロック選択可能
- アクノリッジメントモード／シリアルモード選択可能
- 送信データの出力順序の選択可能（MSB/LSB ファーストの選択）
- オンチップフィルタ（ノイズ除去回路）を持ち、データの信頼性が損なわれません。

2.2.2 H8/300、H8/300L シリーズ内蔵 I²C バスインタフェースの内部ブロック構成

図 2.1 に I²C バスインタフェースの内部ブロック図を示します。プリスケーラ（PS）、クロック制御部、データ制御回路、バス状態判定回路、バスアービトレーション判定回路、アドレス比較回路、割り込み制御部、バス情報およびデータを記憶するレジスタ群から構成されています。

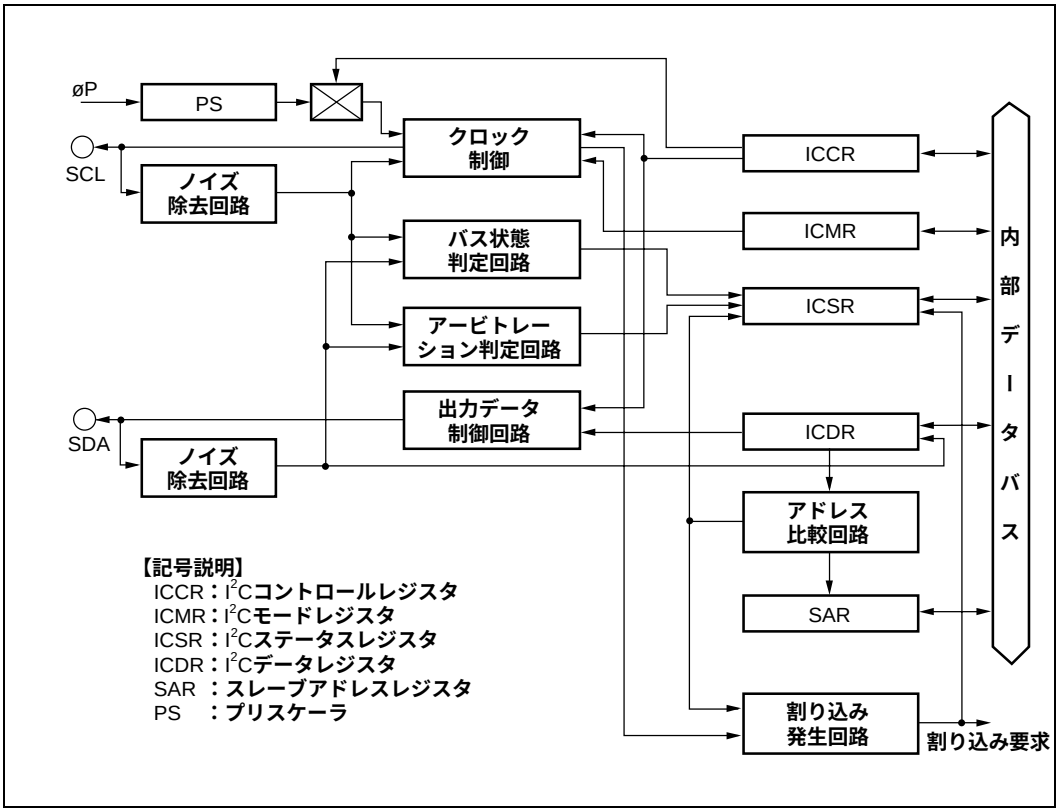


図 2.1 I²C **バスインタフェースブロック図**

表 2.2 にレジスタについて説明します。

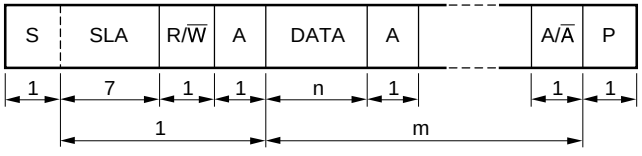
表 2.2 I²C **バスインタフェースの内蔵レジスタ**

名称	略称	機能
I²C バスコントロールレジスタ	ICCR	転送モード設定用レジスタです。
I²C バスステータスレジスタ	ICSR	各状態フラグがセットされます。
I²C バスデータレジスタ	ICDR	送受信データを格納します。
I²C バスモードレジスタ	ICMR	転送フォーマットを設定するレジスタです。
スレーブアドレスレジスタ	SAR	スレーブアドレスを設定するレジスタです。

2.2.3 H8/300、H8/300L シリーズ内蔵 I²C バスインタフェースのデータ転送フォーマット

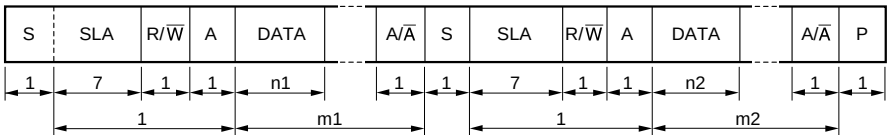
I²C バスインタフェースで転送可能なデータ転送フォーマットには以下の 3 種類があります。転送フレーム数は無制限です。

(1) アドレッシングフォーマット



転送ビット数（n=1〜8）
転送フレーム数（m=1〜）

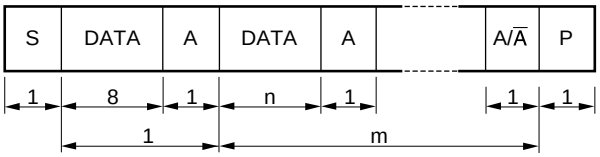
(2) アドレッシングフォーマット（開始条件再送時）



上段：転送ビット数（n1, n2=1〜8）
下段：転送フレーム数（m1, m2=1〜）

開始条件再送時のアドレッシングフォーマットは転送の途中でデータの転送方向を変えたいとき（データ転送の構造化）などに使用します。再送開始条件発行後のスレーブアドレスは開始条件発行時のものと同一にします。

(3) ノンアドレッシングフォーマット



転送ビット数（n=1〜8）
転送フレーム数（m=1〜）

【記号説明】

S	**：開始条件**
SLA	**：スレーブアドレス**
R/W	**：送信／受信の方向を示す**
A	**：アクノリッジ（受信デバイスがSDAを“L”にする）**
DATA	**：送受信データ**
P	**：停止条件**

スレーブアドレス、R/Wビットを認識しないフォーマットです。

2.2.4 H8/300、H8/300L シリーズ内蔵 I²C バスインタフェース内蔵レジスタの機能説明

I²C バスインタフェース用レジスタの各ビットの機能を表 2.3 に示します。

表 2.3 I²C バスインタフェース内蔵レジスタの機能

レジスタ名	ビット名	機能		Master	Slave	設定箇所と属性
WSCR*1	CKDBL	周辺モジュールへの入力クロックは、システムクロックを 2 分周する／しないを設定します。		○		A： 初期設定ルーチンで設定します。以後設定値は保持されます。再設定する場合には、I²C バスインタフェースの処理終了確認後、行ってください。
STCR*1	IICE	I²C バスインタフェースのレジスタをアクセス可能にします。		○	○	
	IICX	ICCR の CKS2～0 と組み合わせて、転送クロック周波数を選択します。		○		
SAR	FS	ICE=0 のとき	スレーブアドレスを認識する／しないを設定します。	○	○	
	SLV6～0		スレーブアドレスを格納します。FS=0 のときのみ有効です。	○	○	
ICMR	MLS	ICE=1 のとき	MSB ファースト／LSB ファーストの選択を行います。	○	○	B： SCL クロックの動作が停止している状態（バス開放時、データ送受信終了時）で設定を行ってください。以後設定値は保持されます。
	WAIT		送信装置がデータとアクノリッジの間にウェイトを挿入するかを設定します。	○		
	BC2～0		転送ビットを指定します。8 ビット以外は転送直前に設定してください。	○	○	
ICCR	ICE*2	SAR 設定後に“1”を設定します。I²C バスインタフェースは転送可能状態になります。		○	○	
	IEIC	割り込みの禁止／許可を設定します。		○	○	
	MST	マスタ／スレーブ、送信／受信動作を設定します。マスタ側で設定された TRS ビットにより、スレーブの通信モード（送信／受信）は自動的に設定されます。		○	○	
	TRS			○	○	
	ACK	8 ビットシリアルデータ送信後に確認応答ビットを挿入する／しないを指定します。		○	○	
	CKS2～0	転送レートを指定します。		○		

（続く）

表 2.3 I²C バスインタフェース内蔵レジスタの機能（続き）

レジスタ名	ビット名	機能	Master	Slave	設定箇所と属性
ICSR	BBSY	BBSY によりバス状態を監視します。 開始／停止条件を発行します。	○	○	C： データ通信の過程で自動的にセットされるフラグです。通信プロトコルにしたがって順次クリアしてください（BBSY と SCP は開始／停止条件の発行にも使用されます）。
	SCP	• BBY=1、SCP=0 の設定により開始条件を発行します。 • BBY=0、SCP=0 の設定により停止条件を発行します。	○		
	IRIC	割り込み要因発生時、“1”にセットされます。	○	○	
	AL	アービトレーションロスト時、“1”にセットされます。	○		
	AAS	マスタが送信したスレーブアドレスが SAR と一致したとき、“1”にセットされます。		○	
	ADZ	ゼネラルコールアドレス（H'00）を認識したとき、“1”にセットされます。		○	
	ACKB	アクノリッジビットの設定／認識を行ないます。	○	○	B に準じます。
ICDR	ICDR7～0	送信用／受信用データレジスタです。	○	○	データ送受信時にアクセスします。

【注】 *1 H8/3337 シリーズ、H8/3437 シリーズ、H8/3217 シリーズのみに適用

*2 ICE ビットは、I²C バスモジュール用 I/O ポートと汎用 I/O ポートの切り替えを制御するビットです。ICE ビットの切り替えを行った場合、汎用 I/O ポートの設定状態によりクロックや開始／停止条件が擬似的に生成される場合があります。結果、他のデバイスに対して不具合発生の原因になる可能性があります。本ビットを操作する場合には、該当するポートを入力状態、または“H”出力になるよう設定しておくことを推奨します。

2.3 H8S シリーズ内蔵 I²C バスインタフェースの仕様［H8S 系］

2.3.1 H8S シリーズ内蔵 I²C バスインタフェースの特長

日立 16 ビットシングルチップマイクロコンピュータ“ H8S/2138 シリーズ ”を例に、H8S シリーズに内蔵されている I²C バスインタフェースの主な特長を以下に示します。

- アドレッシングフォーマット、ノンアドレッシングフォーマットを選択可能
  - I²C バスフォーマット：アドレッシングフォーマットでアクノリッジビットあり、マスタ、スレーブ動作
  - シリアルフォーマット：ノンアドレッシングフォーマットでアクノリッジビットなし、マスタ動作専用
- I²C バスフォーマットは、Philips 社提唱の I²C バスインタフェースに準拠
- I²C バスフォーマットで、スレーブアドレスを 2 通り設定可能
- I²C バスフォーマットで、マスタモード時、開始、停止条件の自動生成
- I²C バスフォーマットで、受信時、アクノリッジの出力レベルを選択可能
- I²C バスフォーマットで、送信時、アクノリッジビットの自動ロード機能
- I²C バスフォーマットで、マスタモード時のウェイト機能
  - アクノリッジを除くデータ転送後、SCL を Low レベルにしてウェイト状態にすることが可能
    ウェイト要求は、次の転送が可能になった時点で解除
- I²C バスフォーマットで、スレーブモード時のウェイト機能
  - アクノリッジを除くデータ転送後、SCL を Low レベルにしてウェイト要求を発生することが可能
  - ウェイト要求は、次の転送が可能になった時点で解除
- 5 種類の割り込み要因
  - 開始条件検出時（マスタモード時）
  - データ転送終了時：SCL の 9 クロック立ち上がり時。I²C バスフォーマットで送信モード遷移時、および、マスタ競合負け後のアドレス受信を含む
  - アドレス一致時：I²C バスフォーマット、スレーブ受信モードで、いずれかのスレーブアドレスが一致したときまたはゼネラルコールアドレスを受信したとき
  - 停止条件検出時（スレーブモード時）
  - 内部フラグ TDRE/RDRF が 1 にセットされたとき（データが ICDRT から ICDRS、または ICDRS から ICDRR に移動したとき）
- マスタモード時、16 種類の内部クロック選択可能
- バスを直接駆動（SCL/SDA 端子）
  - P52/SCL0、P97/SDA0 の 2 端子は、通常時は NMOS プッシュプル出力、バス駆動機能選択時は NMOS オープンドレイン出力
  - P86/SCL1、P42/SDA1 の 2 端子は、通常時は CMOS 端子、バス駆動機能選択時は NMOS のみで出力

- オンチップフィルタ（ノイズ除去回路）を持ち、データの信頼性が損なわれません。
- PC モニタの規格 DD（Display Data Channel）の制御機能をサポート
  - フォーマットレスから I²C バスフォーマットへの自動切り替えが可能（チャネル 0 のみ）
  - スレーブモードのフォーマットレス（開始条件／終了条件なし、ノンアドレッシング）の動作
  - データ端子共通（SDA）、クロック端子独立（VSYNCI、SCL）の端子構成で動作
  - SCL の立ち下がりで、自動的にフォーマットレスから I²C バスフォーマットへ切り替え

2.3.2 H8S シリーズ I²C バスインタフェースの内部ブロック構成

図 2.2 に H8S/2138 シリーズの I²C バスインタフェース内部ブロック図を示します。

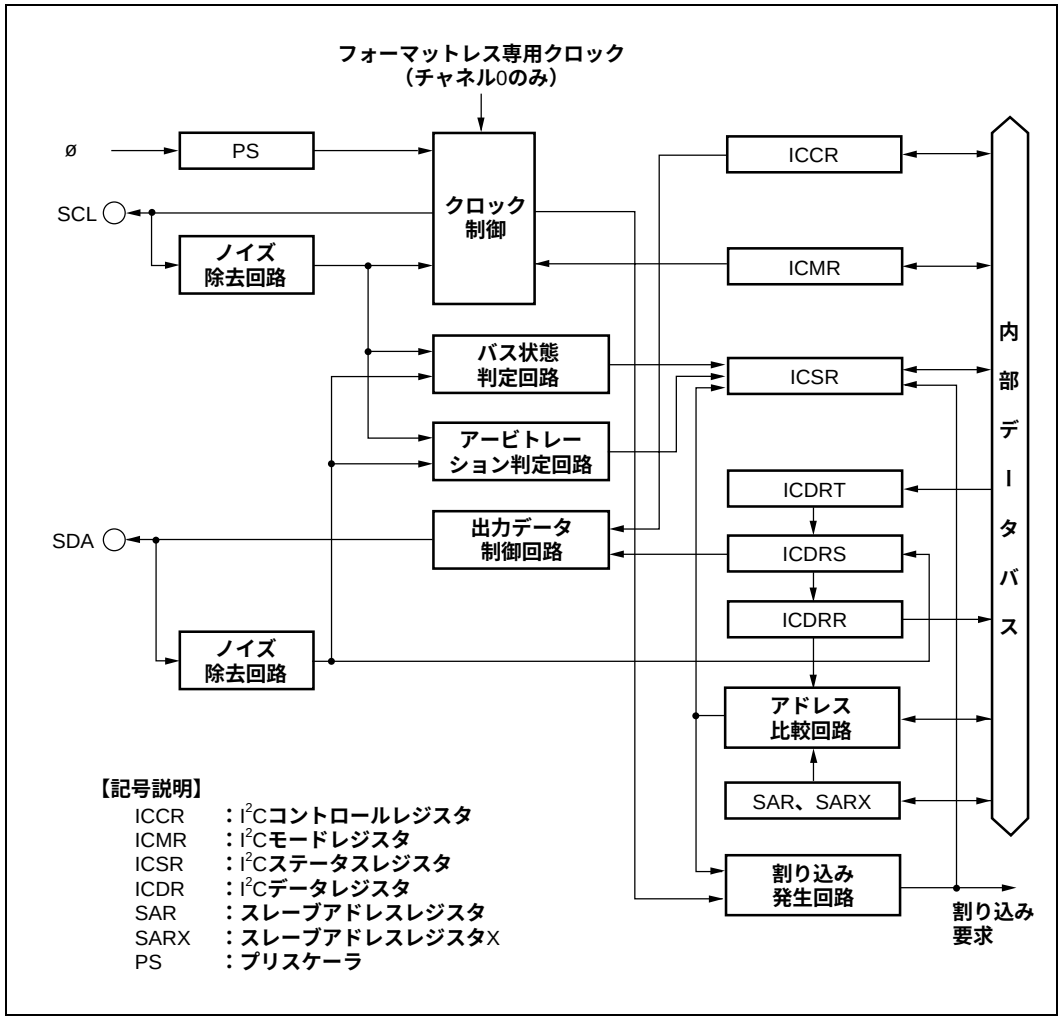


図 2.2 H8S/2138 シリーズ I²C バスインタフェースブロック図

表 2.4 にレジスタについて説明します。

表 2.4　H8S/2138 シリーズ I²C バスインタフェースの内蔵レジスタ

名称	略称	機能
I²C バスコントロールレジスタ	ICCR	転送モード設定用レジスタです。
I²C バスステータスレジスタ	ICSR	各状態フラグがセットされます。
I²C バスデータレジスタ	ICDR	送受信データを格納します。
I²C バスモードレジスタ	ICMR	転送フォーマットを設定するレジスタです。
スレーブアドレスレジスタ	SAR	スレーブアドレスを設定するレジスタです。
スレーブアドレスレジスタ X	SARX	第 2 スレーブアドレスを設定するレジスタです。

2.3.3 H8S シリーズ I²C バスデータフォーマット

図 2.3（a）、（b）に I²C バスフォーマットを示します。I²C バスフォーマットは、開始条件、スレーブを指定するスレーブアドレスフィールド（7 ビットアドレッシング）、通信方向を示す R/$\overline{W}$ ビットフィールド、アクノリッジビットフィールド、データフィールドおよび停止条件から構成されます（記号説明は**表** 2.5 をご参照ください）。

本 I²C バスインタフェースモジュールには、I²C バスフォーマット以外にフォーマットレス（H8S/2138 シリーズでは IIC チャネル 0 で使用可能、他製品については各ハードウエアマニュアルをご確認ください）とシリアルフォーマットが使用できます。これを**図** 2.3（c）、（d）に示します。

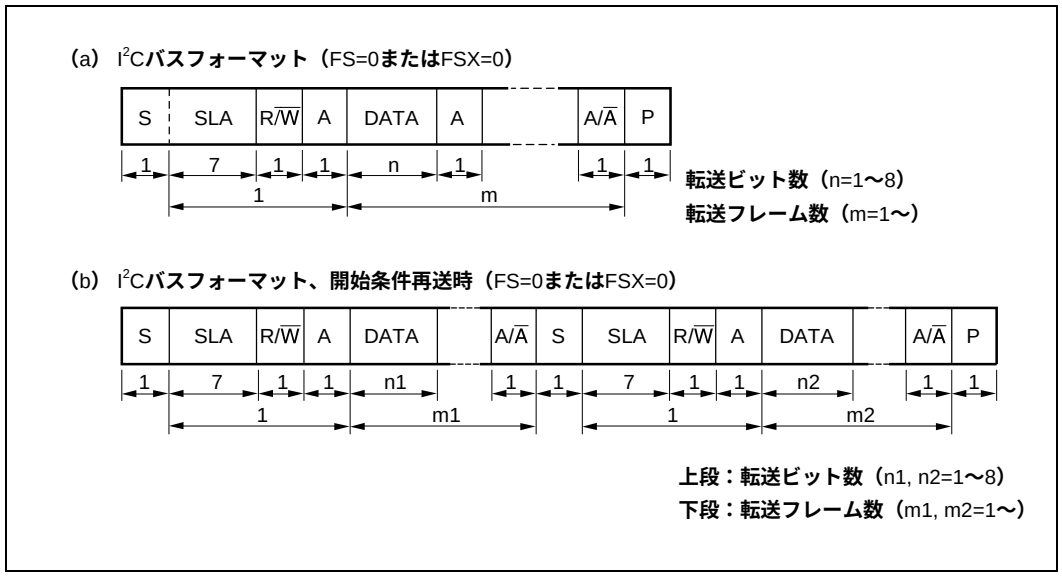
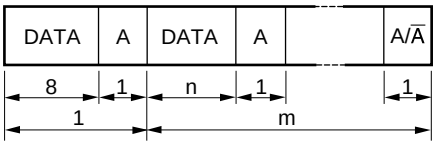


図 2.3 I²C **バスデータフォーマット**

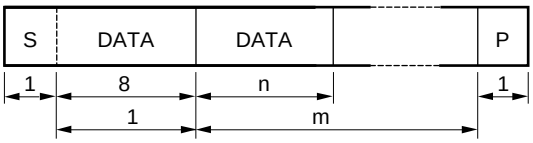
(c) フォーマットレス（IICチャネル0のみ、FS=0またはFSX=0）



転送ビット数（n=1～8）
転送フレーム数（m=1～）

フォーマットレスは、PCモニタシステムの規格DDC（Display Data Channel）に応用されているフォーマットです。

(d) シリアルフォーマット（FS=1かつFSX=1）



転送ビット数（n=1～8）
転送フレーム数（m=1～）

シリアルフォーマットは、スレーブアドレスフィールドとアクノリッジビットフィールドがない、クロック同期式フォーマットです。

図 2.3 その他のフォーマット

表 2.5 に I²C バスデータフォーマットおよび I²C バスタイミングにおける記号説明を示します。

表 2.5 記号説明

記号	機能
S	開始条件を示します。マスタデバイスが SCL=“High”レベルの状態で SDA を“High”レベルから“Low”レベルに変化させます。
SLA	スレーブアドレスを示します。マスタデバイスがスレーブデバイスを選択します。
R/$\overline{W}$	送信／受信の方向を示します。R/$\overline{W}$ ビットが“1”の場合スレーブデバイスからマスタデバイス、R/$\overline{W}$ ビットが“0”の場合マスタデバイスからスレーブデバイスへデータを転送します。
A	アクノリッジを示します。受信デバイスが SDA を“Low”レベルにします（マスタ送信モード時スレーブが、マスタ受信モード時マスタがアクノリッジを返します）。
DATA	送受信データを示します。送受信するデータのビット長は ICMR の BC2～BC0 ビットで設定します。また MSB ファースト／LSB ファーストの切り換えは ICMR の MLS ビットで設定します。
P	停止条件を示します。マスタデバイスが SCL=“High”レベルの状態で SDA を“Low”レベルから“High”レベルに変化させます。

I²C バスのタイミングを**図** 2.4 に示します。

開始条件（S）：SCL が “ High ” の状態で SDA が “ High ” から “ Low ” に変化する動作です。
停止条件（P）：SCL が “ High ” の状態で SDA が “ Low ” から “ High ” に変化する動作です。
データ（SLA/R/$\overline{W}$/DATA）：SCL が “ High ” の期間に SDA 上に確定しています。

AC 特性については、各製品のハードウエアマニュアルをご参照ください。

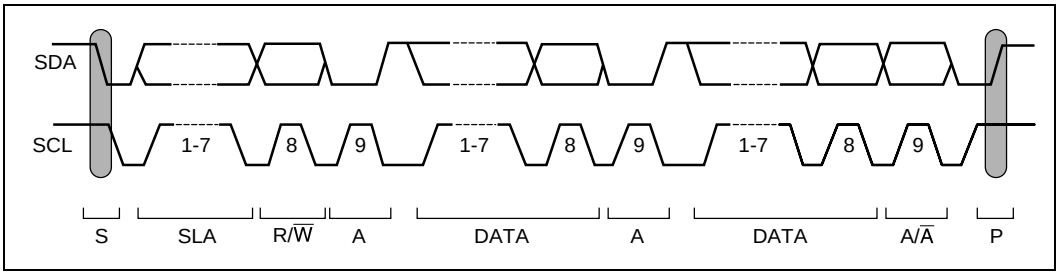


図 2.4 I²C **バスタイミング**

2.3.4 H8S シリーズ I²C バスインタフェース内蔵レジスタの機能説明

H8S シリーズ（H8S/2138 シリーズ）の I²C バスインタフェース内蔵レジスタの機能を**表 2.6** に示します。

表 2.6 内蔵レジスタ機能説明

レジスタ名	ビット名	機能	R/W	初期値
ICDR	ICDR7～0	ICDR は、8 ビットのリード／ライト可能なレジスタで、送信時は送信用データレジスタとして、受信時は受信用データレジスタとして使用します。ICDR は、内部的にシフトレジスタ（ICDRS）、受信バッファ（ICDRR）および送信バッファ（ICDRT）に分かれています。それぞれ CPU からのリード／ライト不可、リード専用およびライト専用となっています。3 本のレジスタ間のデータ転送は、バス状態の変化に関連付けられて自動的に行われ、TDRE や RDRF などの内部フラグの状態に影響を与えます。送信モードで ICDRT の次のデータがある場合（TDRE フラグが 0 の場合）、ICDRS で 1 フレームのデータを送信後、自動的に ICDRT から ICDRS へデータが転送されます。受信モードで ICDRR に以前のデータがない場合（RDRF フラグが 0 の場合）、ICDRS で 1 フレームのデータを正常に受信終了後、自動的に ICDRS から ICDRR にデータが転送されます。ICDR は SARX と同じアドレスに割り付けられており、ICCR の ICE ビットを 1 に設定したときのみ、ICDR のリード／ライトが可能です。	R/W	−
−	TDRE	TDRE は 1 ビットのリード／ライト不可の内部フラグです。 • TDRE=0 送信開始不可、または ICDR（ICDRT）に次のデータが存在することを示します。 【クリア条件】 (1) 送信モード（TRS=1）で ICDR（ICDRT）に送信データをライトしたとき (2) I²C バスフォーマットまたはシリアルフォーマットで停止条件を発行後、バスラインの状態から停止条件成立を検出したとき (3) I²C バスフォーマットで停止条件を検出したとき (4) 受信モード（TRS=0）のとき（転送中の TRS の 0 ライトは、アクノリッジを含めたフレーム受信後に有効） • TDRE=1 ICDR（ICDRT）に次の送信データがライト可能なことを示します。 【セット条件】 (1) 送信モード（TRS=1）のとき、I²C バスフォーマット、シリアルフォーマットのマスタモードで開始条件を発行後、バスラインの状態から開始条件成立を検出したとき (2) フォーマットレスで送信モード（TRS=1）に設定したとき (3) ICDRT から ICDRS にデータが転送されたとき（TRS=1 かつ TDRE=0 で ICDRS が空の場合、ICDRT→ICDRS へデータ転送） (4) 開始条件検出後、受信モード（TRS=0）から送信モード（TRS=1）に切り替えたとき	−	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
−	RDRF	RDRF は 1 ビットのリード／ライト不可の内部フラグです。 • RDRF=0 ICDR（ICDRR）にあるデータが無効なことを示します。 • RDRF=1 ICDR（ICDRR）の受信データがリード可能なことを示します。 【クリア条件】 受信モードで ICDR（ICDRR）の受信データをリードしたとき 【セット条件】 ICDRS から ICDRR にデータが転送されたとき（TRS=0 かつ RDRF=0 で受信正常終了の場合、ICDRS→ICDRR へデータ転送）	−	0
SAR		SAR は、8 ビットのリード／ライト可能なレジスタで、フォーマットの設定およびスレーブアドレスを格納します。アドレッシングモードでスレーブモードの場合、開始条件後に送られてきた第 1 フレームの上位 7 ビットと SAR の上位 7 ビットを比較して一致したとき、マスタデバイスに指定されたスレーブデバイスとして動作します。SAR は ICMR と同じアドレスに割り付けられており、ICCR の ICE ビットを 0 に設定したときのみ、SAR のリード／ライトが可能です。	R/W	H'00
	SVA6〜0	SVA6〜SVA0 ビットには I²C バスにつながっている他のスレーブと異なるユニークなアドレスを設定します。	R/W	0
	FS	SARX の FSX ビット、DDCSWR の SW ビットともに、転送フォーマットを選択します。 • SW=0、FS=0、FSX=0 I²C バスフォーマット（SAR と SARX のスレーブアドレスを認識） • SW=0、FS=0、FSX=1 I²C バスフォーマット（SAR のスレーブアドレスを認識、SARX のスレーブアドレスを無視） • SW=0、FS=1、FSX=0 I²C バスフォーマット（SAR のスレーブアドレスを無視、SARX のスレーブアドレスを認識） • SW=0、FS=1、FSX=1 クロック同期式シリアルフォーマット（SAR と SARX のスレーブアドレスを無視） • SW=1、FS=0、FSX=0 • SW=1、FS=0、FSX=1 • SW=1、FS=1、FSX=0 フォーマットレス（開始条件／停止条件を検出しない、アクノリッジビットあり） • SW=1、FS=1、FSX=1 フォーマットレス*（開始条件／停止条件を検出しない、アクノリッジビットなし）	R/W	0

（続く）

【注】 * DDCSWR の設定により I²C バスフォーマットへの自動切り替えを行う場合は、本モードに設定しないでください。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
SARX		SARX は、8 ビットのリード／ライト可能なレジスタで、フォーマットの設定および第 2 スレーブアドレスを格納します。アドレッシングモードでスレーブモードの場合、開始条件後に送られてきた第 1 フレームの上位 7 ビットと SARX の上位 7 ビットを比較して一致したとき、マスタデバイスに指定されたスレーブデバイスとして動作します。SARX は ICDR と同じアドレスに割り付けられており、ICCR の ICE ビットを 0 に設定したときのみ、SARX のリード／ライトが可能です。	R/W	H'01
	SVAX6～0	SVAX6～SVAX0 ビットには I²C バスにつながっている他のスレーブと異なる固有のアドレスを設定します。	R/W	0
	FSX	スレーブモード時に SARX のスレーブアドレスの認識を行うか否かを選択します。詳細は SAR の FS ビットの項を参照してください。	R/W	1
ICMR		ICMR は、8 ビットのリード／ライト可能なレジスタで、MSB ファースト／LSB ファーストの選択、マスタモードウェイトの制御、マスタモード転送クロック周波数の選択、転送ビット数の選択を行います。ICMR は、SAR と同じアドレスに割り付けられており、ICCR の ICE ビットを 1 に設定したときのみ、ICMR のリード／ライトが可能です。	R/W	H'00
	MLS	MSB ファーストでデータ転送するか、LSB ファーストでデータ転送するかを選択します（1 フレームのアクノリッジを除いたビット数が 8 ビットに満たない場合、送受信データの格納される位置が異なります。送信データは、MLS ビットが 0 のとき MSB 側に、MLS ビットが 1 のとき LSB 側に詰めて書き込んでください。受信データは、MLS ビットが 0 のとき LSB 側から、MLS ビットが 1 のとき MSB 側から読み出したビットを有効にしてください）。なお、I²C バスフォーマットで使用するときは、MLS ビットを 1 にセットしないでください。 • MLS=0 MSB ファースト • MLS=1 LSB ファースト	R/W	0
	WAIT	I²C バスフォーマットでマスタモード時に、アクノリッジを除いたデータ転送後をウェイト状態にするかどうかを設定します。WAIT=1 を設定した場合、データの最終ビットのクロックが立ち下がった後、ICCR の IRIC フラグは 1 にセットされ、ウェイト状態（SCL=“Low”レベル）となります。ICCR の IRIC フラグを 0 にクリアすることで、ウェイト状態を解除しアクノリッジの転送を行います。WAIT=0 を設定した場合、ウェイト状態を挿入せず、データとアクノリッジを連続的に転送します。ICCR の IRIC フラグは、WAIT の設定に関係なく、アクノリッジの転送が完了した時点で 1 にセットされます。 • WAIT=0 データとアクノリッジを連続的に転送 • WAIT=1 データアクノリッジの間にウェイトを挿入	R/W	0

（続く）

表 2.6　内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICMR	CKS2～0	CKS2～CKS0 ビットは、STCR レジスタの IICX1 ビット（チャネル 1）、IICX0（チャネル 0）との組み合わせにより、転送クロックの周波数を選択するビットで、マスタモード時に使用します。必要な転送レートに合わせて設定をしてください。 • IICX=0、CKS2=0、CKS1=0、CKS0=0 転送クロックをφ/28 に設定 • IICX=0、CKS2=0、CKS1=0、CKS0=1 転送クロックをφ/40 に設定 • IICX=0、CKS2=0、CKS1=1、CKS0=0 転送クロックをφ/48 に設定 • IICX=0、CKS2=0、CKS1=1、CKS0=1 転送クロックをφ/64 に設定 • IICX=0、CKS2=1、CKS1=0、CKS0=0 転送クロックをφ/80 に設定 • IICX=0、CKS2=1、CKS1=0、CKS0=1 転送クロックをφ/100 に設定 • IICX=0、CKS2=1、CKS1=1、CKS0=0 転送クロックをφ/112 に設定 • IICX=0、CKS2=1、CKS1=1、CKS0=1 転送クロックをφ/128 に設定 • IICX=1、CKS2=0、CKS1=0、CKS0=0 転送クロックをφ/56 に設定 • IICX=1、CKS2=0、CKS1=0、CKS0=1 転送クロックをφ/80 に設定 • IICX=1、CKS2=0、CKS1=1、CKS0=0 転送クロックをφ/96 に設定 • IICX=1、CKS2=0、CKS1=1、CKS0=1 転送クロックをφ/128 に設定 • IICX=1、CKS2=1、CKS1=0、CKS0=0 転送クロックをφ/160 に設定 • IICX=1、CKS2=1、CKS1=0、CKS0=1 転送クロックをφ/200 に設定 • IICX=1、CKS2=1、CKS1=1、CKS0=0 転送クロックをφ/224 に設定 • IICX=1、CKS2=1、CKS1=1、CKS0=1 転送クロックをφ/256 に設定	R/W	0

（続く）

表 2.6　内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICMR	BC2～0	BC2～BC0 ビットは、次に転送するデータのビット数を指定します。I²C バスフォーマット（SAR の FS ビットまたは SARX の FSX ビット 0 のとき）では、データにアクノリッジ分 1 ビットが加算されて転送されます。BC2～BC0 ビットの転送フレーム間で行ってください。また、BC2～BC0 ビットに 000 以外を設定する場合は、SCL が“Low”状態のときの行ってください。BC2～BC0 ビットは、リセット時および開始条件検出時 000 に初期化されます。また、アクノリッジを含むデータ転送終了後、000 に再び戻ります。 • BC2=0、BC1=0、BC0=0 クロック同期式シリアル = 8 ビット／フレーム I²C バス = 9 ビット／フレーム • BC2=0、BC1=0、BC0=1 クロック同期式シリアル = 1 ビット／フレーム I²C バス = 2 ビット／フレーム • BC2=0、BC1=1、BC0=0 クロック同期式シリアル = 2 ビット／フレーム I²C バス = 3 ビット／フレーム • BC2=0、BC1=1、BC0=1 クロック同期式シリアル = 3 ビット／フレーム I²C バス = 4 ビット／フレーム • BC2=1、BC1=0、BC0=0 クロック同期式シリアル = 4 ビット／フレーム I²C バス = 5 ビット／フレーム • BC2=1、BC1=0、BC0=1 クロック同期式シリアル = 5 ビット／フレーム I²C バス = 6 ビット／フレーム • BC2=1、BC1=1、BC0=0 クロック同期式シリアル = 6 ビット／フレーム I²C バス = 7 ビット／フレーム • BC2=1、BC1=1、BC0=1 クロック同期式シリアル = 7 ビット／フレーム I²C バス = 8 ビット／フレーム	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICSR	AAS	AAS フラグは、I²C バスフォーマットのスレーブ受信モードで、開始条件直後の第 1 フレームが SAR の SVA6～SVA0 と一致した場合、またはゼネラルコールアドレス（H'00）を検出した場合、AAS=1 となります。AAS フラグのクリアは、AAS=1 をリードした後、0 をライトすることで行われます。また ICDR をライト（送信時）またはリード（受信時）すると自動的にリセットされます。 •AAS=0 スレーブアドレスまたはゼネラルコールアドレスを未認識 【クリア条件】 (1) ICDR にデータをライト（送信時）、または ICDR のデータをリード（受信時）したとき (2) AAS=1 の状態をリードした後、0 をライトしたとき (3) マスタモードのとき •AAS=1 スリーブアドレスまたはゼネラルコールアドレスを認識 【クリア条件】 スレーブ受信モードかつ FS=0 でスレーブアドレスまたはゼネラルコールアドレスを検出したとき	R/(W)*	0
	ADZ	ADZ フラグは、I²C バスフォーマットのスレーブ受信モードで、開始条件直後の第 1 フレームがゼネラルコールアドレス（H'00）を検出した場合、ADZ=1 となります。ADZ フラグのクリアは、ADZ=1 をリードした後、0 をライトすることで行われます。また ICDR をライト（送信時）またはリード（受信時）すると自動的にリセットされます。 •ADZ=0 ゼネラルコールアドレスを未認識 【クリア条件】 (1) ICDR にデータをライト（送信時）、または ICDR のデータをリード（受信時）したとき (2) ADZ=1 の状態をリードした後、0 をライトしたとき (3) マスタモードのとき •ADZ=1 ゼネラルコールアドレスを認識 【クリア条件】 スレーブ受信モードかつ（FS=0 または FSX=0）ゼネラルコールアドレスを検出したとき	R/(W)*	0

（続く）

【注】 * フラグをクリアするための 0 ライトのみ可能です。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICSR	ACKB	ACKB ビットは、アクノリッジデータを格納するビットです。送信モードでは、受信デバイスがデータを受信した後、アクノリッジデータを返してくれるので、そのデータを ACKB ビットにロードします。また、受信モードでは送信デバイスに対し、データを受信した後、あらかじめ ACKB ビットに設定されたアクノリッジデータを送出します。ACKB ビットをリードすると、送信時（TRS=1）にはロードした値（受信デバイスから帰ってきた値）が読み出され、受信時（TRS=0）には設定した値が読み出されます。 • ACKB=0 受信時、アクノリッジ出力タイミングで 0 出力 送信時、受信デバイスからアクノリッジがあった（0 だった）ことを示す。 • ACKB=0 受信時、アクノリッジ出力タイミングで 1 出力 送信時、受信デバイスからアクノリッジがあった（1 だった）ことを示す。	R/W	0
ICCR		ICCR は、8 ビットのリード／ライト可能なレジスタで、I²C バスインタフェースの動作／非動作、割り込みの許可／禁止、マスタモード／スレーブモード、送信／受信、アクノリッジの有効／無効の選択、I²C バスインタフェースのバス状態の確認、開始／停止条件の発行、および割り込みフラグの確認を行います。	R/W	H'01
	ICE	ICE ビットは、I²C バスインタフェースを使用する／使用しないを選択します。ICE ビットを 1 にセットすると、I²C バスインタフェースモジュールは転送動作可能状態となり、ポートは SCL、SDA 入出力端子となります。ICE ビットを 0 にクリアすると、I²C バスインタフェースモジュールは機能を停止し、内部状態を初期化します。ICE=0 のとき SAR および SARX が有効になり、ICE=1 のとき、ICMR および ICDR が有効になります。 • ICE=0 I²C バスインタフェースモジュールは非動作状態（SCL/SDA 端子はポート機能） IIC モジュールの内部状態の初期化 SAR、SARX がアクセス可能 • ICE=1 I²C バスインタフェースモジュールは転送動作可能状態（SCL/SDA 端子はバス駆動状態） ICMR、ICDR がアクセス可能	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICCR	IEIC	IEIC ビットは、I²C バスインタフェースから CPU に対する割り込みの許可／禁止を選択します。 •IEIC=0 I²C バスインタフェースの割り込み要求を禁止 •IEIC=1 I²C バスインタフェースの割り込み要求を許可	R/W	0
	MST	MST ビットは、I²C バスインタフェースをマスタモードで使用するか、スレーブモードで使用するかを選択します。 •MST=0 スレーブモード 【クリア条件】 (1) ソフトウェアにより 0 をライトしたとき (2) I²C バスフォーマットのマスタモードで、送信を開始した後バス競合負けしたとき •MST=1 マスタモード 【セット条件】 (1) ソフトウェアにより 1 をライトしたとき（クリア条件(2)以外の場合） (2) MST=0 をリード後、1 をライトしたとき（クリア条件(2)の場合）	R/W	0
	TRS	TRS ビットは、I²C バスインタフェースを受信モードで使用するか、送信モードで使用するかを選択します。 •TRS=0 受信モード 【クリア条件】 (1) ソフトウェアにより 0 をライトしたとき（セット条件(3)以外の場合） (2) TRS=1 をリード後、0 をライトしたとき（セット条件(3)の場合） (3) I²C バスフォーマットのマスタモードで、送信を開始したのちバス競合負けしたとき (4) DDCSWR の SW ビットが 1 から 0 に変化したとき •TRS=1 送信モード 【セット条件】 (1) ソフトウェアにより 1 をライトしたとき（クリア条件(3)、(4)以外の場合） (2) TRS=0 をリード後、1 をライトしたとき（クリア条件(3)、(4)の場合） (3) I²C バスフォーマットのスレーブモードで第 1 フレームの R/$\overline{W}$ ビットとして 1 を受信したとき	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICCR	ACKE	ACKE ビットは、I²C バスフォーマットで受信デバイスから返されるアクノリッジビット内容を無視して連続的に転送を行うか、アクノリッジビットが 1 ならば転送を中断してエラー処理を行うかを選択します。ACKE ビットが 0 の場合には、受信したアクノリッジビットの内容は ACKB ビットに反映されず、ACKB ビットは常時 0 となります。 • ACKE=0 アクノリッジビットの内容を無視して、連続的に転送を行う • ACKE=1 アクノリッジビットが 1 の場合、連続的な転送を中断する	R/W	0
	BBSY	BBSY フラグをリードすることにより、I²C バス（SCL、SDA）が占有されているか解放されているかを確認できます。また、マスタモードでは開始条件、停止条件を発行する際に使用します。BBSY フラグは SCL=“ High ”レベルの状態で SDA が“ High ”レベルから“ Low ”レベルに変化すると開始条件が発行されたと認識し、1 にセットされます。SCL=“ High ”レベルの状態で SDA が“ Low ”レベルから“ High ”レベルに変化すると停止条件が発行されたと認識し、0 にクリアされます。また、開始条件を発行する場合、BBSY=1 かつ SCP=0 をライトします。開始条件の再送信時も同様に行います。停止条件の発行は BBSY=0 かつ SCP=0 をライトすることで行います。開始条件/停止条件の発行は、MOV 命令を用います。スレーブモード時の BBSY フラグのライトは無効です。すなわち、開始条件の発行に先立って、I²C バスインタフェースをマスタ送信モードに設定する必要があります。BBSY=1 かつ SCP=0 をライトする以前に、MST=1、TRS=1 を設定してください。 • BBSY=0 バス解放状態 【クリア条件】 停止条件検出時 • BBSY=1 バス占有状態 【セット状態】 開始条件検出時	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICCR	IRIC	IRIC フラグは、I²C バスインタフェースが CPU に対して割り込み要求を発生させたことを示します。IRIC フラグは、データ転送終了時、スレーブ受信モードでスレーブアドレスまたはゼネラルコールアドレスを検出したとき、マスタ送信モードでバス競合負けをしたとき、または停止条件検出時に 1 にセットされます。SAR の FS ビットと ICMR の WAIT ビットの組み合わせにより IRIC フラグのセットタイミングが異なりますので、注意してください。また、ICCR の ACKE ビットの設定よっても、IRIC フラグがセットされる条件が異なります。IRIC フラグのクリアは、IRIC=1 をリードした後、0 をライトすることで行われます。また、DTC を利用すると IRIC フラグは自動的にクリアされ、CPU を介さない連続的な転送が可能です。 • IRIC=0 転送待ち状態、または転送中 【クリア条件】 (1) IRIC=1 の状態でリードした後、0 をライトしたとき (2) DTC で ICDR をリード／ライトしたとき （TDRE または RDRF フラグが 0 にクリアされたとき） • IRIC=1 割り込みが発生 【セット条件】 1．I²C バスフォーマットでマスタモード (1) 開始条件を発行後、バスラインの状態から開始条件を検出したとき（第 1 フレーム送信のため TDRE フラグが 1 にセットされたとき） (2) WAIT=1 の場合、データとアクノリッジの間にウェイトを挿入したとき (3) データ転送終了時（送受信クロックの 9 クロック目の立ち上がりのとき、およびウェイト挿入時の送受信クロックの 8 クロック目の立ち下がりのとき） (4) バス競合負けの後、スレーブアドレスを受信したとき（AL フラグが 1 にセットされたとき） (5) ACKE ビットが 1 のとき、アクノリッジビットとして 1 を受信したとき（ACKE ビットが 1 にセットされたとき） 2．I²C バスフォーマットでスレーブモード (1) スレーブアドレス（SVA、SVAX）が一致したとき（AAS、AASX フラグが 1 にセットされたとき）、およびその後の再送開始条件または停止条件検出までのデータ転送終了時(TDRE または RDRF フラグが 1 にセットされたとき） (2) ゼネラルコールアドレスを検出したとき（FS=0 かつ ADZ フラグが 1 にセットされたとき）、および、その後の再送開始条件または停止条件検出までのデータ転送終了時（TDRE または RDRF フラグが 1 にセットされたとき）	R/(W)*	0

（続く）

【注】 * フラグをクリアするための 0 ライトのみ可能です。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICCR	IRIC	(3) ACKE ビットが 1 のとき、アクノリッジビットとして 1 を受信したとき（ACKB ビットが 1 にセットされたとき） (4) 停止条件を検出したとき（STOP または ESTP フラグが 1 にセットされたとき） 3．クロック同期式シリアルフォーマット、およびフォーマットレス (1) データ転送終了時（TDRE または RDRF フラグが 1 にセットされたとき） (2) シリアルフォーマットで開始条件を検出したとき (3) DDCSWR の SW ビットを 1 にセットしたとき 上記のほか、TDRE または RDRF が 1 にセットされる条件が発生したとき	R/(W)*	0
	SCP	SCP ビットは、マスタモードでの開始条件／停止条件の発行を制御します。開始条件を発行する場合、BBSY=1 かつ SCP=0 をライトします。開始条件の再送信時も同様に行います。また、停止条件の発行は BBSY=0 かつ SCP=0 をライトすることで行います。SCP ビットは、リードすると常に 1 が読み出されます。また、1 をライトしてもデータは格納されません。 • SCP=0 ライト時、BBSY フラグと組み合わせて開始条件、停止条件を発行 • SCP=1 リード時、常に 1 をリード ライト時、無効	W	1
ICSR		ICSR は、8 ビットのリード／ライト可能なレジスタで、フラグの確認、アクノリッジの確認および制御を行います。	R/W	H'00
	ESTP	ESTP フラグは、I²C バスフォーマットのスレーブモードで、フレームの転送の途中で停止条件を検出したことを示します。 • ESTP=0 エラー停止条件なし 【クリア条件】 (1) ESTP=1 の状態をリードした後、0 をライトしたとき (2) IRIC フラグが 0 にクリアされたとき • ESTP=1 I²C バスフォーマットでスレーブモードのとき、エラー停止条件を検出 【セット条件】 フレームの転送の途中で停止条件を検出したとき • I²C バスフォーマットでスレーブモードのとき、意味なし	R/(W)*	0

（続く）

【注】 * フラグをクリアするための 0 ライトのみ可能です。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICSR	STOP	STOP フラグは、I²C バスフォーマットのスレーブモードで、フレームの転送の完了後に停止条件を検出したことを示します。 •STOP=0 正常停止条件なし 【クリア条件】 (1) STOP=1 の状態をリードした後、0 をライトしたとき (2) IRIC フラグが 0 にクリアされたとき •STOP=1 I²C バスフォーマットでスレーブモードのとき正常停止条件を検出 【セット条件】 フレーム転送の完了後に停止条件を検出したとき • I²C バスフォーマットでスレーブモードのとき以外、意味なし	R/(W)*	0
	STOP	IRTR フラグは、I²C バスインタフェースが CPU に対して割り込み要求を発生させており、その要因が DTC 起動可能な連続送受信動作の 1 フレーム送受信の完了であることを示します。IRTR フラグが 1 にセットされると、同時に IRIC フラグも 1 にセットされます。IRTR フラグのセットは、TDRE または RDRF フラグが 1 にセットされたときに行われます。IRTR フラグのクリアは、IRTR=1 をリードした後、0 をライトすることで行われます。IRIC フラグを 0 にクリアすると IRTR フラグは自動的にクリアされます。	R/(W)*	0
	IRTR	•IRTR=0 転送待ち状態、または転送中 【クリア条件】 (3) IRTR=1 の状態をリードした後、0 をライトしたとき (4) IRIC フラグが 0 にクリアされたとき •IRTR=1 連続転送状態 【セット条件】 (1) I²C バスフォーマットでスレーブモードのとき • AASX=1 の状態で TDRE または RDRF フラグが 1 にセットされたとき (2) I²C バスフォーマットでスレーブモードのとき以外 • TDRE または RDRF フラグが 1 にセットされたとき	R/(W)*	0

（続く）

【注】 * フラグをクリアするための 0 ライトのみ可能です。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
ICSR	AASX	AASX フラグは、I²C バスフォーマットのスレーブ受信モードで、開始条件直後の第 1 フレームが SARX の SVAX6～SVAX0 と一致した場合、AASX=1 となります。AASX フラグのクリアは、AASX=1 をリードした後、0 をライトすることで行われます。また、開始条件を検出すると自動的にクリアされます。 • AASX=0 第 2 スレーブアドレスを未認識 【クリア条件】 (1) AASX=1 の状態をリードした後、0 をライトしたとき (2) 開始条件を検出したとき (3) マスタモードのとき • AASX=1 第 2 スレーブアドレスを認識 【セット条件】 スレーブ受信モードでかつ FSX=0 で第 2 スレーブアドレスを検出したとき	R/(W)*	0
	AL	AL フラグは、マスタモード時にバス競合負けをしたことを示します。複数のマスタがほぼ同時にバスを占有しようとしたときに I²C バスインタフェースは SDA をモニタし、自分が出したデータと異なった場合、AL フラグを 1 にセットしてバスが他のマスタによって占有されたことを示します。AL フラグのクリアは、AL=1 をリードした後、0 をライトすることで行われます。また、ICDR をライト（送信時）またはリード（受信時）すると自動的にリセットされます。 • AL=0 バスを確保 【クリア条件】 (1) ICDR にデータをライト（送信時）、データをリード（受信時）したとき (2) AL=1 の状態をリードした後、0 をライトしたとき • AL=1 バス競合負け（アービトレーションロスト） 【セット条件】 (1) マスタ送信モードで SCL の立ち上がりで内部 SDA と SDA 端子が不一致のとき (2) マスタ受信モードで SCL の立ち下がりで内部 SCL が“ High ”レベルのとき	R/(W)*	0

（続く）

【注】 * フラグをクリアするための 0 ライトのみ可能です。

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
STCR		STCR は 8 ビットのリード／ライト可能なレジスタで、レジスタアクセスの制御、IIC の動作モードの制御（IIC 内蔵オプションの場合）、内蔵フラッシュメモリの制御（F-ZTAT 版の場合）、TCNT の入力クロックの選択を行います。I²C バスインタフェース以外の詳細は省略します。STCR で制御するモジュールを使用しない場合は、当該ビットに 1 をライトしないでください。	R/W	H'00
	IICX1	IICX1 ビットは、ICMR の CKS2～CKS0 と組み合わせて、IIC チャネル 1 のマスタモードでの転送レートを選択します。転送レートの詳細は ICMR の CSK2～CKS0 の項を参照してください。	R/W	0
	IICX0	IICX0 ビットは、ICMR の CKS2～CKS0 と組み合わせて、IIC チャネル 0 のマスタモードでの転送レートを選択します。転送レートの詳細は ICMR の CSK2～CKS0 の項を参照してください。	R/W	0
	IICE	IICE ビットは、I²C バスインタフェースのデータレジスタ、制御レジスタ（ICCR、ICSR、ICDR/SARX、ICMR/SAR）の CPU アクセスを制御します。 • IICE=0 I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを禁止 • IICE=1 I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	R/W	0
DDCSWR		DDCSWR は、8 ビットのリード／ライト可能なレジスタで、IIC チャネル 0 のフォーマット自動切り替え機能の制御および IIC の内部ラッチクリアの制御を行います。	R/W	H'0F
	SWE	SWE ビットは、IIC チャネル 0 でフォーマットレスから I²C バスフォーマットへの自動切り替え機能を選択します。 • SWE=0 IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを禁止 • SWE=1 IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを許可	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
DDCSWR	SW	SW ビットは、IIC チャネル 0 でフォーマットレスと、I²C バスフォーマットを選択します。 • SW=0 IIC チャネル 0 を、I²C バスフォーマットで使用する 【クリア条件】 (1) ソフトウェアにより 0 をライトしたとき (2) SWE=1 で、SCL に立ち下がりエッジを検出したとき • SW=1 IIC チャネル 0 を、フォーマットレスで使用する 【セット条件】 SW=0 の状態をリードした後、1 をライトしたとき	R/W	0
	IE	IE ビットは、IIC チャネル 0 でフォーマットの自動切り替えが実行された場合の CPU への割り込み要求を許可／禁止します。 • IE=0 フォーマット自動切り替え実行時の割り込みを禁止 • IE=1 フォーマット自動切り替え実行時の割り込みを許可	R/W	0
	IF	IF ビットは、IIC チャネル 0 でフォーマット自動切り替えが実行された場合の CPU への割り込み要求フラグです。 • IF=0 フォーマット自動切り替え実行時の割り込み要求なし 【クリア条件】 IF=1 の状態ををリードした後、0 をライトしたとき • IF=1 フォーマット自動切り替え実行時の割り込み要求あり 【セット条件】 SWE=1 で、SCL に立ち下がりエッジを検出したとき	R/W	0

（続く）

表 2.6 内蔵レジスタ機能説明（続き）

レジスタ名	ビット名	機能	R/W	初期値
DDCSWR	CLR3～0	CLR3～CLR0 ビットは IIC0、IIC1 の内部状態の初期化を制御します。CLR3～CLR0 ビットはライト動作のみ可能で、リードすると常に 1 が読み出されます。ライト動作により対応するモジュールの内部ラッチ回路へのクリア信号が発生し、IIC モジュールの内部状態が初期化されます。なお、ライトデータは保持されません。IIC クリアを行う場合は、必ず MOV 命令を使用し、CLR3～CLR0 ビットを同時に書き込んでください。BCLR などのビット操作命令は使用しないでください。再度クリアが必要な場合は、すべてのビットとも設定にしたがい書き込みする必要があります。 •CLR3=0、CLR2=0、CLR1=＊、CLR0=＊ 設定禁止 •CLR3=0、CLR2=1、CLR1=0、CLR0=0 設定禁止 •CLR3=0、CLR2=1、CLR1=0、CLR0=1IIC0 内部ラッチクリア •CLR3=0、CLR2=1、CLR1=1、CLR0=0IIC1 内部ラッチクリア •CLR3=0、CLR2=1、CLR1=1、CLR0=1IIC0、1 内部ラッチクリア •CLR3=1、CLR2=＊、CLR1=＊または CLR0=＊ 設定無効 【注】 ＊： 0 または 1	W*1	1
MSTPCRL	MSTP4	MSTP4 ビットは、IIC チャネル 0 のモジュールを指定します。 •MSTP4=0 IIC チャネル 0 のモジュールストップモード解除 •MSTP4=1 IIC チャネル 0 のモジュールストップモード設定	R/W	1
	MSTP3	MSTP3 ビットは、IIC チャネル 1 のモジュールを指定します。 •MSTP3=0 IIC チャネル 1 のモジュールストップモード解除 •MSTP3=1 IIC チャネル 1 のモジュールストップモード設定	R/W	1

【注】 *1 リードすると常に 1 が読み出されます。

2.3.5 H8S シリーズ（H8S/2138 シリーズ）内蔵 I²C バスインタフェースのフラグと転送状態の関係

I²C バスフォーマットで ICCR の IRIC フラグが 1 にセットされ、割り込みが発生した場合には、IRIC=1 となった要因を調べるために、他のフラグを調べる必要があります。各要因には、それぞれ対応するフラグがありますが、データ転送終了時に関しては注意が必要です。

内部フラグである TDRE または RDRF フラグがセットされたとき、リード可能な IRTR フラグがセットされる場合とされない場合があります。DTC 起動要求フラグである IRTR フラグがデータ転送終了時にセットされないのは、I²C バスフォーマットでスレーブモードの場合に、スレーブアドレス（SVA）またはゼネラルコールアドレスが一致した後の再開始条件または停止条件検出までの期間です。

IRIC フラグ、IRTR フラグがセットされているときでも、内部フラグである TDRE または RDRF フラグがセットされない場合があります。DTC を利用した連続的な転送の場合、設定した回数の転送終了時には、IRIC フラグおよび IRTR フラグはクリアされません。一方、設定した回数の ICDR のリード／ライトは完了しているため TDRE または RDRF フラグはクリアされています。

各フラグと転送状態の関係を**表** 2.7 に示します。

表 2.7 **フラグと転送状態の関係**

MST	TRS	BBSY	ESTP	STOP	IRTR	AASX	AL	AAS	ADZ	ACKB	状態
1/0	1/0	0	0	0	0	0	0	0	0	0	**アイドル状態（フラグクリア要）**
1	1	0	0	0	0	0	0	0	0	0	**開始条件発行**
1	1	1	0	0	1	0	0	0	0	0	**開始条件成立**
1	1/0	1	0	0	0	0	0	0	0	0/1	**マスタモードウェイト**
1	1/0	1	0	0	1	0	0	0	0	0/1	**マスタモード送信／受信終了**
0	0	1	0	0	0	1/0	1	1/0	1/0	0	**アービトレーションロスト**
0	0	1	0	0	0	0	0	1	0	0	**スレーブモード第フレームで SAR に一致**
0	0	1	0	0	0	0	0	1	1	0	**ゼネラルコールアドレスに一致**
0	0	1	0	0	0	1	0	0	0	0	**SARX に一致**
0	1/0	1	0	0	0	0	0	0	0	0/1	**スレーブモード送信／受信終了（SARX 一致後以外）**
0 0	1/0 1	1 1	0 0	0 0	1 0	1 1	0 0	0 0	0 0	0 1	**スレーブモード送信／受信終了（SARX 一致後）**
0	1/0	0	1/0	1/0	0	0	0	0	0	0/1	**停止条件検出**

2.4 I²C バスインタフェースの使用説明

(1) バス状態の確認方法［H8 系、H8S 系］

I²C バスではマスタデバイスはデータ通信を開始する前に、バスが開放状態（SCL/SDA ラインともに定常的に“ High ”状態）にあるかを確認する必要があります。H8 系のモジュールでは、ICSR レジスタの BBSY ビット、H8S 系の場合には ICCR レジスタの BBSY ビットをリードすることによりバス状態を確認することができます。BBSY=“ 0 ”の場合、バス開放状態ですので、マスタデバイスとしてデータ通信を開始することができます。

(2) 開始条件、停止条件の発行方法［H8 系、H8S 系］

開始条件は、SCL が“ High ”の状態で SDA が“ High ”から“ Low ”に変化する動作、停止条件は、SCL が“ High ”の状態で SDA が“ Low ”から“ High ”に変化する動作です。開始条件は BBSY=“ 1 ”、SCP=“ 0 ”を同時にレジスタ（H8 系モジュールでは ICSR レジスタ、H8S 系では ICCR レジスタ）にライトすることにより生成されます。停止条件は BBSY=“ 0 ”、SCP=“ 0 ”を同時にライトします。このため開始条件／停止条件の発行は、MOV 命令を使用してください。

なお、**「2.4 (6)、(7) 命令の連続発行について、(8) 開始条件再発送時の注意」**もご参照ください。

(3) データ送信のしくみ［H8 系、H8S 系］

マスタ動作の場合

ICDR レジスタにデータをライトすることにより、データ送信を開始します。データ送信完了後（または開始条件生成後）、SCL ラインを“ Low ”に保持し、通信待ち状態にします。

スレーブ動作の場合：

ICDR レジスタにデータをライトすることにより、SCL ラインの“ Low ”ドライブを開放し、データ送信の準備をします。マスタデバイスから送信される SCL クロックに同期しデータをマスタデバイスに送信します。データ送信完了後は、SCL ラインを“ Low ”に保持し、マスタデバイスに対して待ち状態を知らせます。最終データの送信終了後、ICDR に H'FF をライトし、SCL ラインを解放してください。これによりマスタデバイスは停止条件を発行することができます。

(4) データ受信のしくみ（H8 系 I²C モジュールの場合）［H8 系］

マスタ動作の場合

ICDR レジスタをリードすることにより、SCL クロックを出力し、データ受信を開始します。最初のデータリードはダミーリードになり、データ受信の完了を確認した後のデータリードが実際のデータ受信になります。データ受信完了後は、次の ICDR リード動作まで SCL ラインを“ Low ”に保持し、通信

待ち状態にします。最終データのリードは、最終データの受信完了を確認後、TRS=“1”にし、送信モードに設定してから行ってください。

スレーブ動作の場合

I²C バスシステムでは、マスタ以外のデバイスはスレーブ受信モードから動作が始まります。最初の 1 バイト目はスレーブアドレス＋R/W ビットになるため、SCL をハイインピーダンス状態にし、スレーブアドレスデータをデータレジスタ（ICDR）にロードします。第 8 ビット目をロードした時点で、スレーブアドレスレジスタ（SAR）と比較を行います。アドレスが一致した場合、第 9 クロック目でアクノリッジをマスタに返し、このとき IRIC フラグがセットされ、I²C バス割り込みが許可されている場合（IEIC=1）には、割り込みが発生します。アドレスが不一致の場合には、IRIC はセットされずスレーブモードとして待機状態になります。

スレーブアドレスフェーズの第 8 ビット目のビットは R/W ビットを意味し、“1”のときはスレーブ側から見て以降の動作が送信モード、“0”のときは受信モードとなります。第 8 ビット目のビットは自動的に TRS ビットに反映されます。

TRS=0 の場合、引き続きスレーブ受信モードになり、CPU が ICDR をリードするまで、SCL を“Low”にドライブし、マスタに待ち状態を知らせます（TRS=1 の場合、スレーブ送信モードになりますので、ICDR にデータを設定するまで、SCL を“Low”にドライブし、マスタに待ち状態を知らせます）。

(5) データ受信のしくみ（H8S 系 I²C モジュールの場合）［H8S 系］

H8S 系 I²C モジュールでは、データ受信バッファは ICDRR（CPU が可能なレジスタ ICDR）と ICDRS（シフトレジスタ）から構成されています。データ受信のトリガ（ICDR レジスタのダミーリード）をかけることにより 2 バイトのデータを受信しますので、複数データを連続してリードするようなアプリケーションでは、CPU への負担を軽減することができます。

マスタ動作の場合

ICDR レジスタをリードすることにより、SCL クロックを出力し、データ受信を開始します。最初のデータリードはダミーリードになり、データ受信の完了を確認した後のデータリードが実際のデータ受信になります。データバッファがダブル構成になっているため、ICDRR（ICDR）レジスタが空または CPU が ICDRR（ICDR）をリードしている場合には、次のデータ受信を行います。ICDRR（ICDR）および ICDRS に受信データが格納されている場合には、次の ICDR リード動作まで SCL ラインを“Low”に保持し、通信待ち状態にします。最終データの受信は、以下のように行ってください。

(a) 複数データ（3 バイト以上）を受信する場合

- 最終データ受信をする前の 2 バイトデータを ICDRR（ICDR）と ICDRS に格納。SCL は“Low”保持状態。
- WAIT ビットを 1 にセット後、上記 2 バイトデータを連続リードし、バッファを空にする。
- 最終データ受信の SCL 8 クロック目の立ち下がりで IRIC フラグ割り込みが発生するので、TRS ビットを 1 にセット（送信モードにセット）する。また ACKB ビットを 1 にセットする。その後、IRIC フラグを

クリアし、9 クロック目を出力させる。

- 最終データ受信の IRIC 割り込みが発生するので、最終データをリードする。
- WAIT ビット、ACKB ビット、IRIC フラグの順で 0 にクリアし、停止条件を発行する。

(b) 2 バイト以下のデータを受信する場合

- データ受信を開始する前に WAIT ビットを 1 にセットする。
- ICDRR（ICDR）をダミーリードし、データ受信を開始する。
- SCL 8 クロック目の立ち下がりで IRIC フラグ割り込みが発生するので、IRIC フラグをクリアし、SCL 9 クロック目を出力させる。
- 9 クロック目の立ち上がりでデータ受信完了。
- ICDRR（ICDR）をリードしデータを取り込む。
- 2 バイト目のデータ受信で、SCL 8 クロック目の立ち下がりで IRIC フラグ割り込みが発生するので、TRS ビットを 1 にセット（送信モードにセット）する。また ACKB ビットを 1 にセットする。その後、IRIC フラグをクリアし、9 クロック目を出力させる。
- 最終データ受信の IRIC 割り込みが発生するので、最終データをリードする。
- WAIT ビット、ACKB ビット、IRIC フラグの順で 0 にクリアし、停止条件を発行する。

マスタ受信動作の具体例は、4 章の応用例を参照ください。

スレーブ動作の場合

本 I²C モジュールはデータレジスタがダブルバッファ構成（ICDRS および ICDRR/ICDR）のため、第 1 データであるスレーブアドレスの受信に引き続き、第 2 データを連続受信することができます。まず、マスタによる開始条件あとのスレーブアドレスをバッファ（ICDRS）に入力し、スレーブアドレスレジスタ（SAR または SARX）値と比較を行います。一致した場合、第 9 クロック目でアクノリッジをマスタに返し、アドレスデータはデータレジスタ（ICDRR/ICDR）にロードされます。このとき IRIC フラグがセットされ、I²C バス割り込みが許可されている場合（IEIC=1）には、割り込みが発生します。アドレスが不一致の場合には、アドレスデータは ICDRR/ICDR レジスタにはロードされずスレーブモードとして待機状態になります。

スレーブアドレスフェーズの第 8 ビット目のビットは R/W ビットを意味し、“1”のときはスレーブ側から見て以降の動作が送信モード、“0”のときは受信モードとなります。第 8 ビット目のビットは自動的に TRS ビットに反映されます。

TRS=0 の場合、引き続きスレーブ受信動作になります。今 ICDRS は空になっているので、マスタの SCL クロックの出力により次のデータを連続受信します。第 9 クロック目でアクノリッジをマスタに返し、CPU がスレーブアドレスデータを ICDR からリードしている場合には、データが ICDRS から ICDRR/ICDR にシフトされます。このとき IRIC フラグが 1 にセットされ、I²C バス割り込みが許可されている場合（IEIC=1）には、割り込みが発生します。ICDRS が再び空になり、次のデータを連続受信します。

上記動作において、他の割り込み処理動作のため I²C バスの割り込み処理が遅延して CPU が以前に受

信したデータを ICDR からリードしていない場合には（RDRF 内部フラグ＝1）、次データは受信完了時に ICDRS に保持されると共に、SCL を“ Low ”にドライブし、マスタに対して通信を待ち状態にします。これにより受信データは保護されます。第 1 データの受信完了割り込みは、第 2 データの受信完了割り込みに丸め込められますが、CPU が ICDRR/ICDR 内の第 1 データをリード後、直ちに ICDRS 内の第 2 データは ICDRR/ICDRS にシフトされ、このとき再度 IRIC がセットされ、I²C バス割り込みが許可されている場合（IEIC=1）には、割り込みが発生します。スレーブ受信時の割り込み手順例を以下に示します。

スレーブ受信時の割り込み手順例［H8S 系］

（a） ステータスレジスタ（ICSR）の内容確認

- スレーブアドレスの一致を確認（AAS または AASX=1）
- 停止条件の検出（STOP=1）
- エラー停止条件の検出（ESTOP=1）
- アービトレーションロストの検出（AL=1）
- ゼネラルコールアドレス（b’0000000）の検出（ADZ=1）

（b） IRIC フラグをクリア

（c） ICDR をリードし、データ取り込み

（d） スレーブアドレス受信後の場合

- ICCR レジスタの TRS ビットを判定し、以降の動作モード（受信／送信モード）を確認（TRS=1 の場合、以降の動作はスレーブ送信モードになり、ICDR にデータを設定するまで、SCL を“ Low ”にドライブし、マスタに待ちを知らせます。）

（6） 命令の連続発行について（H8 系 I²C モジュールの場合）［H8 系］

開始条件発行命令、データ送信／受信開始命令、停止条件発行命令を連続して発行されるようなプログラムは、正常に動作しない場合があります。例えば開始条件命令を発行したとき、その命令タイミングやバスラインの負荷によって実際の開始条件の生成が遅延し、次のデータ送信命令等が内部的に競合し無視される場合があるからです。注意点を以下に示します。

（a） 開始条件発行後のデータ送信命令発行タイミング

- 開始条件発行命令発行後、必要に応じてデータ転送レートの 1 クロック分のウェイト時間を入れ、データ送信命令を実行する。

（b） 開始条件発行後、停止条件を発行する場合

- BBSY=1 であることを確認し、バス権を取得したことを確認する。

（c） 開始条件発行後、通信モードを変更する場合

- BBSY=1 であることを確認し、バス権を取得したことを確認する。

(d) 停止条件発行後、開始条件を発行する場合

- BBSY=0 であることを確認し、バスが開放されていることを確認する。

(e) 停止条件発行後、通信モードを変更する場合

- BBSY=0 であることを確認し、バスが開放されていることを確認する。

(f) データ送受信完了後、次のデータ送受信を開始する場合

- データ送信の場合、データ転送完了（IRIC=“1”）を確認し、IRIC を“0”にクリアした後、次データを ICDR にライトしてください。
- データ受信の場合、データ転送完了（IRIC=“1”）を確認し、ICDR をリードし、IRIC を“0”にクリアしてください。ICDR のリードは、TRS=0 の場合、次のデータ受信のトリガになります。最終データをリードする場合には TRS=1 にしてから、ICDR をリードし、受信データを取り込んでください。

(g) データ送受信完了後、開始条件を再発行する場合（再送開始条件の発行）

- 本動作は、マスタ送信動作からマスタ受信動作に切り替える場合に応用されます。まずデータ送信完了（IRIC=“1”）を確認し、IRIC を“0”にクリアした後、開始条件発行命令を実行してください。

(h) データ送受信完了後、停止条件を発行する場合

- マスタ送信動作の場合、データ送信完了（IRIC=“1”）を確認し、IRIC を“0”にクリアした後、停止条件を発行してください。
- マスタ受信動作の場合、データ受信完了（IRIC=“1”）を確認し、TRS=1 にセット（マスタ送信モード）した後、最終データをリードします。その後 IRIC を“0”にクリアし、停止条件を発行します。

(7) 命令の連続発行について（H8S 系 I²C モジュールの場合）［H8S 系］

(a) 開始条件発行後のデータ送信命令発行タイミング

- 開始条件発行命令実行し、IRIC フラグにて開始条件生成を確認した後、データ送信命令を実行する

(b) 開始条件発行後、停止条件を発行する場合

- 開始条件発行命令実行し、IRIC フラグにて開始条件生成を確認する。BBSY=1 であることを確認した後、停止条件を発行する。

(c) 開始条件発行後、通信モードを変更する場合

- 開始条件発行命令実行し、IRIC フラグにて開始条件生成を確認する。BBSY=1 であることを確認した後、通信モードを変更する。

(d) 停止条件発行後、開始条件を発行する場合

- BBSY=0 であることを確認し、バスが開放されていることを確認する。

(e) 停止条件発行後、通信モードを変更する場合

- BBSY=0 であることを確認し、バスが開放されていることを確認する。

(f) データ送受信完了後、次のデータ送受信を開始する場合

- データ送信の場合、データ転送完了（IRIC=“1”）を確認し、次データを ICDR にライトしてください。次のデータ転送の完了を確認するために、IRIC を“0”にクリアしてください。

- データ受信の場合、データ転送完了（IRIC=“1”）を確認後、ICDR をリードし、IRIC を“0”にクリアしてください。H8S 系 I²C モジュールでは、データ受信バッファが 2 段構成になっているため、ICDRR（ICDR）をリードした後は、2 バイトのデータを連続受信します。データ受信を終了させるためには、最終データ受信中（SCL 第 1 クロック立ち上がりから第 9 クロックの立ち上がりまでの間）に TRS ビットを 1（送信モード）に切り替える必要があります。この設定方法については**「2.4 （13）ウェイト動作［H8 系、H8S 系］」**にて説明します。

（g） データ送受信完了後、開始条件を再発行する場合（再送開始条件の発行）

- 本動作は、マスタ送信動作からマスタ受信動作に切り替える場合に応用されます。まずデータ送信完了（IRIC=“1”）を確認し、IRIC を“0”にクリアした後、開始条件発行命令を実行してください。

（h） データ送受信完了後、停止条件を発行する場合

- マスタ送信動作の場合、データ送信完了（IRIC=“1”）を確認し、IRIC を“0”にクリアした後、停止条件を発行してください。
- マスタ受信動作の場合、データ受信完了（IRIC=“1”）を確認し、TRS=1 にセット（マスタ送信モード）した後、最終データをリードします。その後 IRIC を“0”にクリアし、停止条件を発行します。

（8） 開始条件再送時の注意［H8 系、H8S 系］

再開始条件を発行しデータを転送する場合、次バイト転送命令の実行は、再開始条件発行後、SCL が立ち上がってから（**図** 2.5 中の(A)点）行ってください。

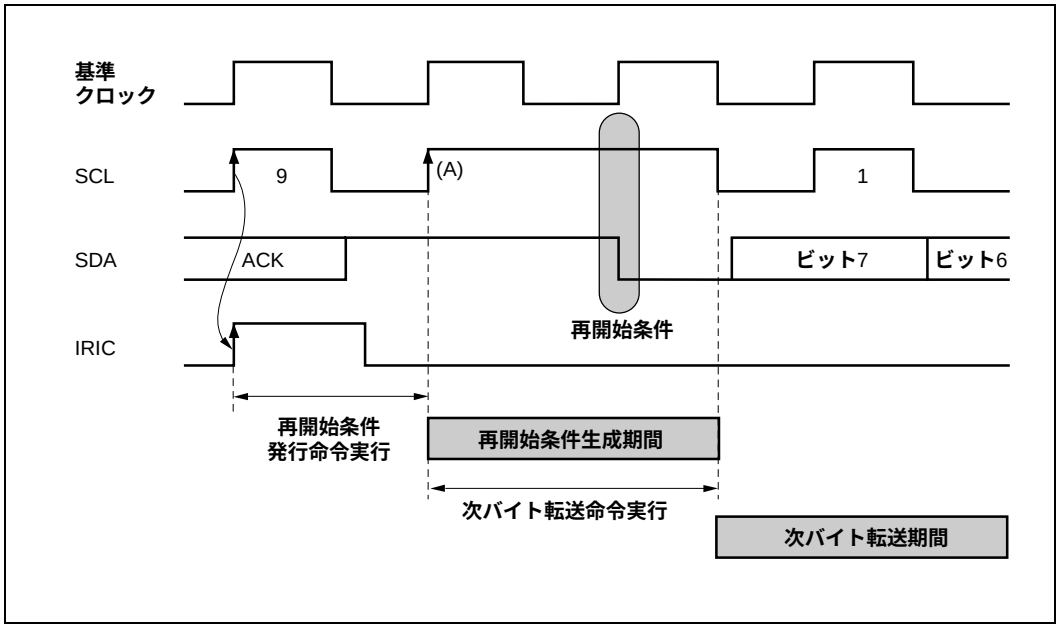


図 2.5 **再開始条件発行時における次バイト転送命令実行タイミング**

具体的には次のように行います。

I²C バスはバス占有時の転送待ち状態が SCL=“ Low ”、SDA=“ High ”状態なので、まず SCL=“ Low ”であることを確認し、再開始条件発行命令を実行します。次に再開始条件生成で SCL が“ Low ”から“ High ”に変化するので、SCL=“ High ”であることを確認し、次バイト転送命令を実行します。H8S 系の場合には、再開始条件の成立で割り込みが発生するので、その後、次バイト転送命令を実行してください。

（9）スレーブアドレス一致の確認［H8 系、H8S 系］

マスタデバイスから送信されたスレーブアドレスは、ビットごとに SAR（H8S 系 I²C モジュールの場合には、2 通りのスレーブアドレス SAR および SARX が選択可能）と比較されます。スレーブアドレスが一致した場合、AAS ビット（H8S 系 I²C モジュールの場合には、AAS または AASX ビット）がセットされ、SCL 第 9 クロックの立ち上がり時の IRIC 割り込み時に自分がマスタデバイスから指定されたスレーブデバイスであることを確認できます。

（10）ゼネラルコールアドレスの認識［H8 系、H8S 系］

ゼネラルコールアドレスは H'00 であり、マスタデバイスがすべての I²C デバイスをスレーブデバイスとして指定するときに使用されます。本 I²C モジュールはこのアドレスを認識すると ADZ フラグを 1 にセットします。SCL 第 9 クロックの立ち上がり時の IRIC 割り込み時に確認します。

（11）アクノリッジビットの認識と設定［H8 系、H8S 系］

データ送信デバイス（TRS=“ 1 ”）は、SCL 第 9 クロック目のタイミングで、データ受信デバイスからアクノリッジビットを受け取ります。この値は ACKB ビットにロードされるので、SCL 第 9 クロックの立ち上がり時の IRIC 割り込み時に確認します。

データ受信デバイス（TRS=“ 0 ”）は、SCL 第 9 クロック目のタイミングで ACKB ビットに設定された値を SDA ラインに出力します。このとき TRS=“ 1 ”が設定されていると、送信モード時の ACKB ビット値が出力されるので注意してください。TRS=“ 1 ”か TRS=“ 0 ”かによって、内部には 2 種類の ACKB ビットがあります。

（12）スレーブ動作時の送受信モードの設定［H8 系、H8S 系］

スレーブ動作時には、マスタデバイスから送出されるスレーブアドレス後の R/$\overline{W}$ ビットの値が自動的に TRS ビットに反映されます。例えば、R/$\overline{W}$ ビットの値が“ 1 ”（マスタデバイスからみてリード動作）であれば、自動的に TRS ビットが“ 1 ”にセットされ、スレーブ送信モードに切り替わります。

（13）ウェイト動作［H8 系、H8S 系］

マスタモード時、WAIT ビットを“1”にセットすることにより、SCL 8 クロックと 9 クロックの間にウェイトを挿入することができます。I²C モジュールは 8 クロックまで出力後、SCL ラインを“Low”に保持します。IRIC フラグを“0”にクリアすることによって、9 クロック目を送出します。

I²C バスや SMBus では、マスタ受信動作における最終データ受信時、スレーブデバイスに対してアクノリッジを返さないというプロトコルがある場合があります。ウェイト動作を応用し、SCL クロックを 8 クロック目で停止させ、ACKB ビットを“0”から“1”に切り替えることによって、アクノリッジビットの制御が容易にできます。

また、H8S 系の I²C モジュールでバイトごとのマスタ受信動作をする場合、本ウェイト動作を応用します。ウェイト動作中に送信モード（TRS =“1”）にすることにより、IRIC フラグクリア後の SCL 9 クロック出力タイミングで、送信モードが有効になり、SCL クロックの出力を停止できます。2.4（5）および（7）の（f）項を併わせてご参照ください。

（14）転送ビット数の確認方法［H8 系、H8S 系］

ICMR レジスタの BC2～0 ビット群は、SCL クロック数を制御するビットカウンタです。1 クロックを出力するごとにダウンカウントされます。このビットを読み出すことにより現在何ビット送出したかを確認できますが、値を書き戻す場合には注意が必要です。例えば I²C モジュールが SCL クロックを出力した直後に、BC2～0 ビットに前と同じ値を書き戻した場合、SCL クロックが 1 つ余分に出力され、スレーブデバイスとのビットずれが発生します。

（15）AL、AAS（AASX）、ADZ ビットのクリア［H8 系、H8S 系］

AL（アービトレーションロストフラグ）、AAS（スレーブアドレス認識フラグ）、ADZ（ゼネラルコールアドレス認識フラグ）のクリアはそれぞれのビットを読み込んだ後、“0”を書き込むことで行います。また AAS と ADZ は ICDR のリード／ライトにより、AASX は開始条件の検出により自動クリアされます。

（16）バスアービトレーション処理［H8 系、H8S 系］

I²C バスはマルチマスタに対応しており、バス権調停のしくみを持っています（詳細は**図** 1.9 をご参照ください）。複数のマスタデバイスが同時に開始条件を発行した場合、各デバイスは SCL ラインのクロックの立ち上がり時に、SDA ライン上のデータと内部の SDA データを比較し、データが異なっている場合、バスのドライブを停止します。つまり SDA ライン上に最後まで“Low”を出力しつづけたデバイスがマスタデバイスとなります。

本 I²C モジュールは、バス権を喪失（バスアービトレーションロスト）した時点で AL フラグを“1”にセットすると共に、バスの出力段を off します。またバス権を取得したマスタデバイスが H8 をスレーブデバイスとして指定している場合があるため、動作モードを自動的にマスタ送信モードからスレーブ受信モードに変更

します。スレーブアドレスが一致している場合（AAS または AASX=1）、SCL の 9 クロック目の立ち上がりで割り込みが入りますので、ここで AL=1 が確認できます。スレーブアドレスが一致しなかった場合、停止条件の検出により割り込みが入りますので、ここで AL=1 を確認します。**図** 2.6 にバスアービトレーション処理のフロー例を示します。

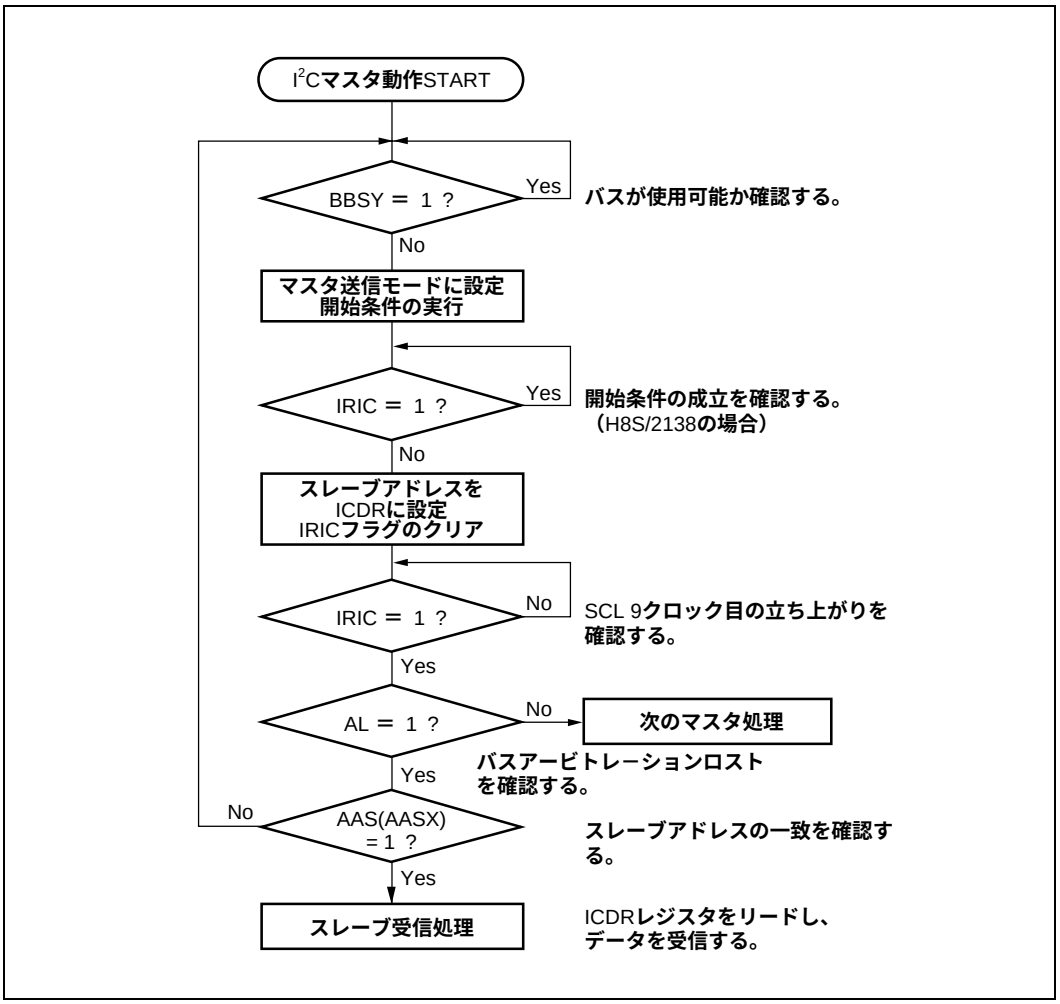


図 2.6　**バスアービトレーション処理**

（17）ICE ビットの制御範囲［H8 系、H8S 系］

ICE ビットは、a）I/O アドレスの割り当て制御（SAR レジスタまたは ICMR レジスタの切り替え）、b）SCL、SDA 端子機能と汎用 I/O ポートへの切り替え（H8/3947 シリーズでは Hi-Z への切り替え）を制御します。

H8S 系 I²C モジュールではさらに ICE ビットをクリアすることにより内部状態を初期化します。例えば通信異常が発生し、マイコンがバスラインを“ Low ”に固定している場合、正常状態への復帰手段として応用できます。

（18）シリアルコミュニケーションインタフェースと I²C バスの併用［H8 系］（H8/3337 シリーズ、H8/3437 シリーズ）

H8/3337 シリーズ、H8/3437 シリーズは、シリアルコミュニケーションインタフェース（SCI）を 2 本（SCI0、SCI1）持っています。そのうち SCI0 は I²C バスインタフェースと一部レジスタのアドレスを共有しています。また、SCI1 の SCK1 端子（クロック端子）と SCL 端子は兼用端子となっています[*1]。SCI を 2 チャネル、I²C バスを 1 チャネルとして使用する場合、以下の点に注意してください。

(a) SCK1 は SCL と端子を共有しているので、SCI1 は調歩同期式モード（UART）で使用してください。

(b) SCI0 と I²C バスを使用する場合には、IICE=0 の状態で SCI0 の設定を行ってください。I²C バスインタフェースレジスタと共有しているレジスタは、SMR と BBR です。これらのレジスタは初期設定用のレジスタなので、一度設定しておけば、通信モードを変更しない限り再設定する必要はありません。この後、IICE=1 に設定し、I²C バスインタフェースレジスタアクセスに切り替え、I²C バスの設定を行ってください。

【注】 *1 H8/3217 シリーズは、SCI を 2 本（H8/3212 は 1 本）、I²C バスインタフェースを 2 本（H8/3202 は 1 本）をそれぞれ独立に持っています。

（19）シリアルコミュニケーションインタフェースと I²C バスの併用［H8S 系］（H8S/2138 シリーズ、H8S/2148 シリーズ）

H8S/2138 シリーズ、H8S/2148 シリーズは、シリアルコミュニケーションインタフェース（SCI）を 3 本（SCI0、SCI1、SCI2）、I²C バスインタフェースを 2 本（IIC0、IIC1）持っています。そのうち SCI0 と SCI1 は I²C バスインタフェースと一部レジスタのアドレスを共有しています。SCI を 3 チャネル、I²C バスを 2 チャネルとして使用する場合、以下の点に注意してください。

(a) SCI の SCK 端子(SCK0、SCK1、SCK2）は I²C バスの端子と共有しているので、SCI は調歩同期式モード（UART)で使用してください。

(b) SCI と I²C バスを使用する場合には、IICE=0 の状態で SCI0、SCI1、SCI2 の設定を行ってください。I²C バスインタフェースレジスタと共有しているレジスタは、SCMR と BRR です。これらのレジスタは、初期設定用のレジスタなので、一度設定しておけば、通信モードを変更しない限り再設定する必要はありません。この後、IICE=1 に設定し、I²C バスインタフェースレジスタアクセスに切り替え、I²C バスの設

定を行ってください。

2.5 I²C バスの通信同期化について

I²C バスの出力ポートの形式はオープンドレインとなります。このため、バスラインの“ Low ”から“ High ”に変化する時間は、バスラインの負荷に依存します。I²C バスの仕様では、SCL ラインの立ち上がり時間が標準モード（最大データ転送レート：100kbps）時は 1000ns、高速モード（最大データ転送レート：100kbps）時は 300ns と定められています。

また I²C バスでは、SCL ライン（クロックライン）が“ High ”期間にデータが確定している必要があります。バスラインの負荷容量やバスラインを電源に接続しているプルアップ抵抗値が不適切な場合、正常にデータ転送が行われるように、実際のデータ転送レートが変化します（通信の同期化）。

図 2.7 に通信の同期化の一例を示します。本 I²C モジュールでは、マスタ動作時、SCL ライン上にあらかじめ設定されたデータ転送レートの内部基準クロックにしたがい、SCL クロックを出力します。SCL ラインを“ Low ”から“ High ”に立ち上げてから、一定のタイミング（**表** 2.7 をご参照ください）で SCL ラインをモニタし、実際に SCL ラインが“ High ”になったかをビットごとに確認します。もし、SCL ラインの立ち上がりが遅れたり、他のデバイスが SCL ラインを“ Low ”にドライブして、電圧レベルが VIH（I/O の“ High ”認識のスレッショルド電圧）に達していない場合、データ通信が正常に行えるように、SCL ラインを“ Low ”にドライブするタイミングを遅らせます。そして SCL ラインが“ High ”になったことを確認できたタイミングで SCL ラインを“ Low ”にドライブします。結果 SCL ラインの“ High ”期間が延ばされ、データ転送レートが下がることになります。

したがって所望のデータ転送レートになるように、プルアップ抵抗やバスラインの負荷容量を調整する必要があります。

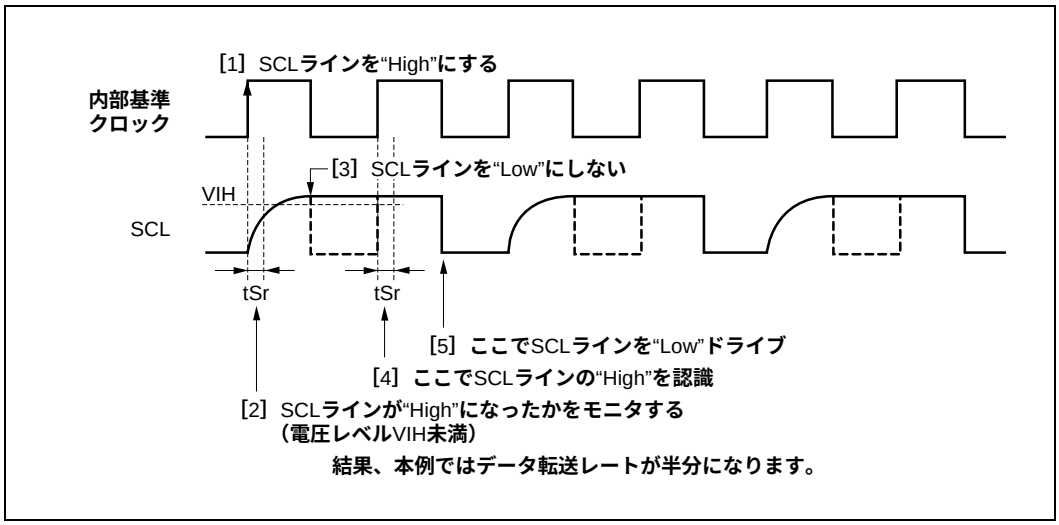


図 2.7 SCL **ラインの立ち上がり時間が遅れた場合**

表 2.7　SCL ラインの立ち上がりモニタタイミング（H8S/2138 シリーズの場合）

IICX ビット	SCL ライン立ち上がりモニタタイミング tSr（tcyc 表示）	モード	tSr 時間表示 [ns]					
			I²C バスの仕様 (max.)	φ=5MHz	8MHz	10MHz	6MHz	20MHz
0	7.5×tcyc*	標準モード	1000	1000	937	750	468	375
		高速モード	300	300	300	300	300	300
1	17.5×tcyc*	標準モード	1000	1000	1000	1000	1000	875
		高速モード	300	300	300	300	300	300

【注】　*　tcyc はマイコンのシステムクロックの周期です。

（ご参考）I²C バス上のプルアップ抵抗の計算例（H8S/2138 シリーズの場合）

I²C バスを電源に接続しているプルアップ抵抗の計算例を示します。

- SCL ラインの負荷容量 CB＝100pF
- SCL ラインの立ち上がり時間 tSr＝300ns
- 電源 Vcc＝5.0V
- I/O の“ High ”判定の電圧レベル VIH＝Vcc×0.7＝3.5V

計算式 Vcc×（1−exp（−t/（CB×R））＝VIH より

∴ R　2.5kΩ

2.6 H8/300、H8/300L シリーズにおけるデータ転送の動作説明 [H8 系]

動作モードは、アドレッシングモード（スレーブアドレスを認識するモード：FS=“0”）を使用し、データの送出は MSB ファースト（MLS=“0”）、ウェイト動作なし（WAIT=“0”）およびアクノリッジメントモード（アクノリッジを認識するモード：ACK=“0”）を使用するものとします。

2.6.1 マスタ送信動作

マスタ送信モードでは、マスタデバイスが送信クロック（SCL ライン）、送信データ（SDA ライン）を出力し、スレーブデバイスがアクノリッジを返します。**図** 2.8 にマスタ送信モードの設定手順と動作について説明します。

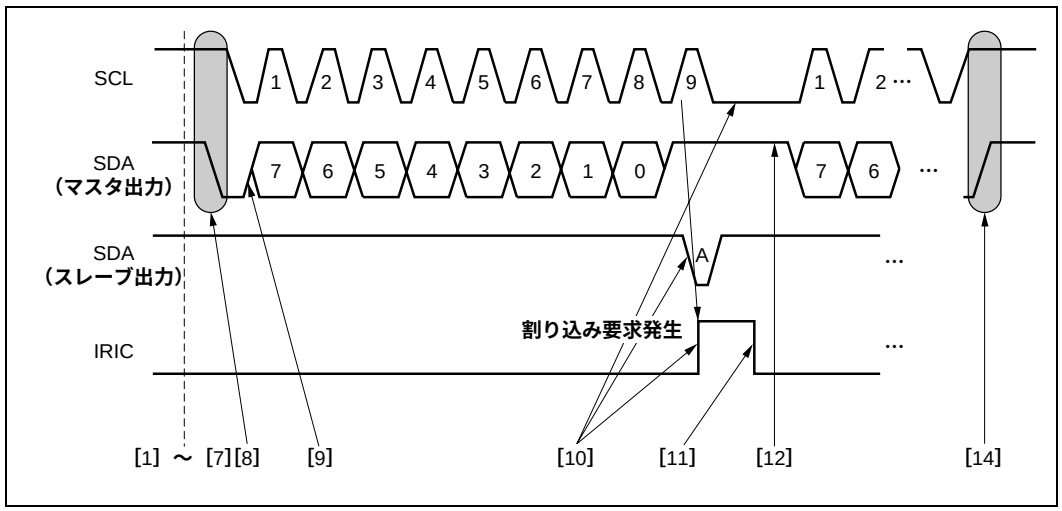


図 2.8 **マスタ送信モード動作タイミング**（MLS=WAIT=ACK=“0”**のとき**）

マスタ送信モード設定手順例

[1] *

ソフトウェア処理　：CKDBL の設定をします。

目的　：周辺用クロックにシステムクロック（φ）を使用するか、2 分周クロック（φ/2）を使用するかを選択します。

[2] *

ソフトウェア処理　：IICE ビットを“1”にセットします。

目的　：I²C バスインタフェースのレジスタアクセスを可能にします。

【注】 * H8/3337 **シリーズ、**H8/3437 **シリーズ、**H8/3217 **シリーズのみの設定です。**

[3]

ソフトウェア処理　：SAR をセットします。SAR は上位 7 ビットをスレーブアドレス、下位 1 ビット（FS ビット）を“ 0 ”とした値です。この設定はシングルマスタ動作の場合も行ってください。

目的　：マスタモード時でもシステムがマルチマスタの場合、スレーブモードになる可能性があるので、SAR を設定します。

[4]

ソフトウェア処理　：ICE ビットを“ 1 ”にセットします。

目的　：SAR と ICMR のアドレスは兼用になっています。これによって SAR アクセスを ICMR アクセスに切り替えます。これにより、データ転送可能状態になります。

[5]

ソフトウェア処理　：ACKB を設定します。

目的　：マスタモードで使用する場合でもバスアービトレーションすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[6]

ソフトウェア処理　：MLS ビット、WAIT ビット、ACK ビットを“ 0 ”に設定します。また、CKS2～0 ビット、IICX ビットおよび IEIC ビットを動作モードに合わせて設定します。

目的　：マスタモードで使用する場合でもバスアービトレーションすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[7]

ソフトウェア処理　：BBSY ビットをリードします。

目的　：バスが解放中か使用中か確認します。解放されていれば、BBSY=“ 0 ”であり、次の設定に進みます。

[8]

ソフトウェア処理　：MSB ビット、TRS ビットをそれぞれ“ 1 ”に設定し、BBSY に“ 1 ”かつ SCP に“ 0 ”をライトします。ただし同時にセットする必要があるので、MOV 命令を使用してください。

目的　：マスタ送信モードにし、開始条件を発行します。

ハードウェア処理　：SCL が“ High ”のときに、SDA が“ High ”から“ Low ”に変化します。

[9]

ソフトウェア処理　：ICDR にデータをライトします。最初のデータはスレーブアドレスと R/$\overline{W}$ ビット（=“ 0 ”）です。

目的　：データ転送を開始します。

ハードウェア処理　：マスタデバイスは**図** 2.8 に示すタイミングで、送信クロックと ICDR にライトされたデータを順次送り出します。

[10]

ソフトウェア処理　：1 バイトのデータ送信が終了し、9 クロック目に IRIC ビットが“ 1 ”にセットされます。また、マスタデバイスはスレーブデバイスからアクノリッジを受信し、ACKB ビットが“ 0 ”にセットされます。このように 1 フレームを転送した後、SCL を内部クロックに同期して“ Low ”レベルに固定します。

目的　：IRIC ビット=“ 1 ”はデータ転送終了、またはバスアービトレーションを知らせます。このとき、IEIC ビットが“ 1 ”にセットされていると、CPU に対し割り込み要求を発生します。スレーブデバイスからのアクノリッジの有無の確認は、ACKB ビットで行います。

[11]

ソフトウェア処理　： IRIC を“ 0 ”にクリアします。

目的　：次のデータ送信のためにクリアします。

[12]

ソフトウェア処理　：ICDR にデータをライトします。

目的　：データ転送を開始します。

[13]

ソフトウェア処理　：[10]～[12]を繰り返し行います。

目的　：データ送信を継続します。

[14]

ソフトウェア処理　：ICSR の BBSY に“ 0 ”、SCP に“ 0 ”をライトします。ただし同時にセットする必要があるので、MOV 命令を使用してください。

目的　：送信を終了するため、停止条件を発行します。

ハードウェア処理　：SCL が“ High ”のときに、SDA が“ Low ”から“ High ”に変化します。

2.6.2 マスタ受信動作

マスタ受信モードでは、マスタデバイスが受信クロック（SCL ライン）を出力し、スレーブデバイスからデータを受信します。また、スレーブデバイスにアクノリッジを返します。

アドレッシングモードでは、最初にマスタ送信モードにてスレーブアドレスを出力します。次に、そのままデータ送信を継続する場合は**「2.6.1 マスタ送信動作」**で示すようになります。データを受信する場合、第 1 フレーム（スレーブアドレスを含むバイトデータ）転送終了後、マスタ受信モードに切り替えます。

図 2.9 にマスタ受信モードの設定手順と動作について説明します。

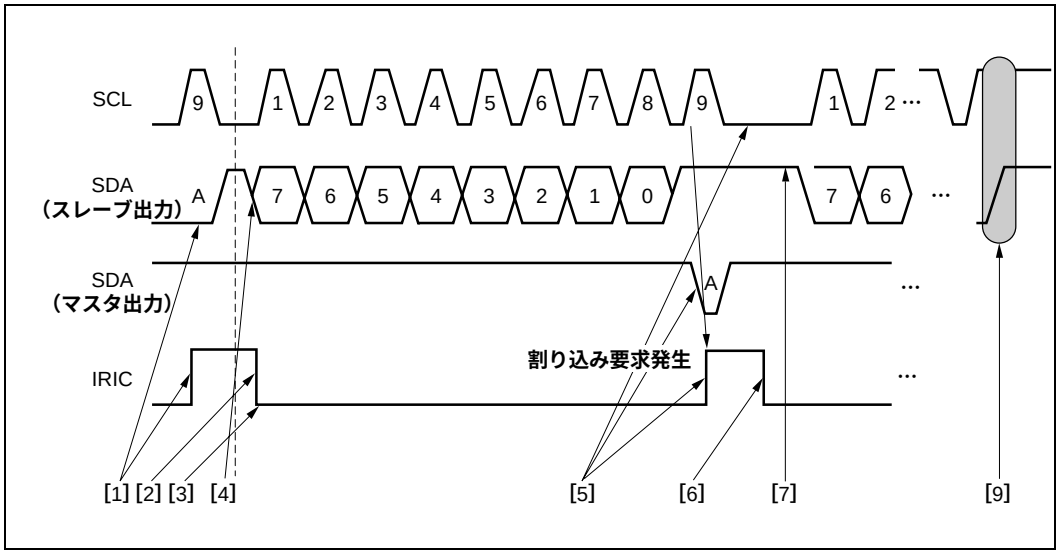


図 2.9 **マスタ受信モード動作タイミング**（MLS=WAIT=ACKB=“ 0 ”**のとき）**

マスタ受信モード設定手順例

[1]

ハードウェア処理 ：マスタデバイスは、マスタ送信モードにて開始条件を発行し、スレーブアドレスを含む第 1 バイトを送り出します。9 クロック目に IRIC ビットが“ 1 ”にセットされます。同時にスレーブデバイスよりアクノリッジを受信し、ACCB ビットが“ 0 ”にセットされます。

目的 ：IRIC=“ 1 ”はスレーブアドレスの一致を知らせます。

[2]

ソフトウェア処理 ：ソフトウェアで IRIC ビットを“ 0 ”にクリアします。

目的 ：次の受信に備えます。

[3]

ソフトウェア処理 ：TRS=“ 0 ”にセットします。

目的 ：マスタ受信モードに切り替えます。

[4]

ソフトウェア処理　：ICDR をリード（ダミーリード）します。

目的　：これにより受信を開始します。

ハードウェア処理　：マスタデバイスは内部クロックに同期して受信クロックを出力し、かつデータを受信します。

[5]

ハードウェア処理　：1 バイトのデータ受信が終了し、9 クロック目に IRIC ビットが“ 1 ”にセットされます。同時に SDA を“ Low ”レベルにし、アクノリッジを返します。また、SCL は 1 フレーム転送終了後、内部クロックに同期して自動的に“ Low ”レベルに固定されます。

目的　：IRIC ビット=“ 1 ”はデータ転送終了を知らせます。このとき、IEIC ビットが“ 1 ”にセットされていると、CPU に対して割り込み要求を発生します。

[6]

ソフトウェア処理　：ソフトウェアで IRIC ビットを“ 0 ”にクリアします。

目的　：次の受信に備えます。

[7]

ソフトウェア処理　：ICDR をリードします。

目的　：内部クロックに同期して次の受信が開始されます。なお、最終バイト受信後にアクノリッジを返さない場合は、受信開始前に ACKB を“ 1 ”にセットしておきます。

[8]

ソフトウェア処理　：［5］～［7］を繰り返し行います。

目的　：データの受信を継続します。

[9]

ソフトウェア処理　：受信を停止する場合は TRS ビットを“ 1 ”にセットし、ICDR をリードした後、BBSY=“ 0 ”かつ SCP=“ 0 ”をライトします。

目的　：データの再受信をしないよう、通信モードを送信モードに切り替えます。次に ICDR のリードで SCL、SDA ラインを解放した後、停止条件を発行します。

ハードウェア処理　：SCL が“ High ”レベルのとき、SDA は“ Low ”レベルから“ High ”レベルに変化します。

2.6.3 スレーブ受信動作

スレーブ受信モードでは、マスタデバイスが送信クロック、送信データを出力し、スレーブデバイスがデータを受信し、アクノリッジを返します。

図 2.10 に、スレーブ受信モードの設定手順と動作を説明します。

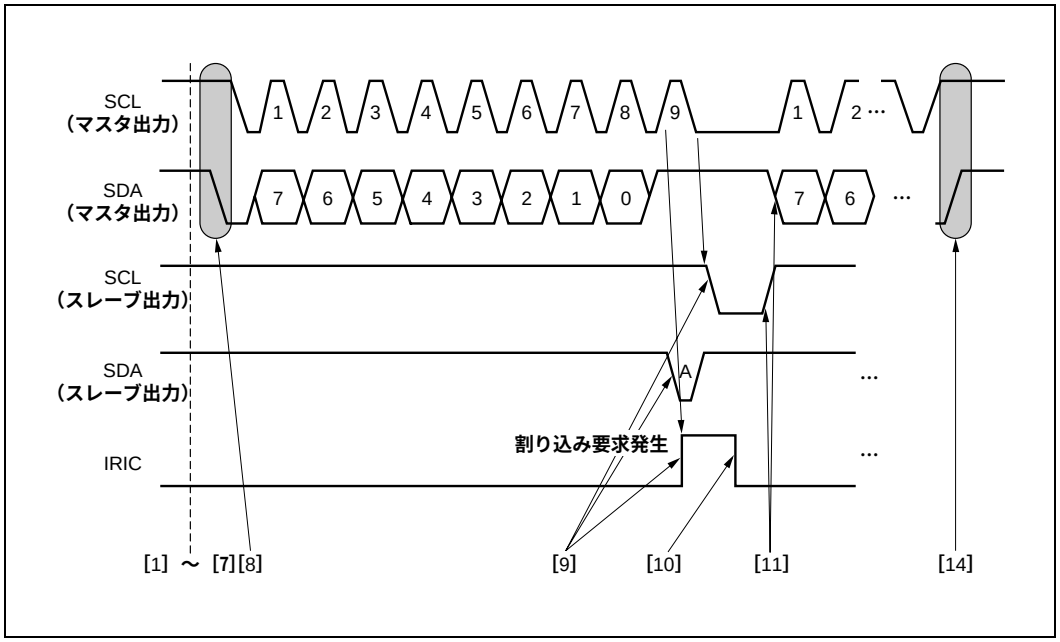


図 2.10 **スレーブ受信モード動作タイミング**（MLS=WAIT=ACKB=“0”**のとき**）

スレーブ受信モード設定手順例

[1]

ソフトウェア処理　：CKDBL の設定をします。

目的　：周辺用クロックにシステムクロック（φ）を使用するか、2 分周クロック（φ/2）を使用するかを選択します。

[2]

ソフトウェア処理　：IICE ビットを“1”にセットします。

目的　：I²C バスインタフェースのレジスタアクセスを可能にします。

[3]

ソフトウェア処理　：SAR を設定します。SAR には上位 7 ビットにスレーブアドレス、下位 1 ビット（FS）に“0”（アドレッシングファーマット時）を書き込みます。

目的　：アドレッシングモードなので、スレーブデバイスにアドレスを割り当てます。

[4]

ソフトウェア処理　：ICE ビットを“1”にセットします。

目的　：SAR と ICMR のアドレスは兼用になっています。これによって SAR アクセスを ICMR アクセスに切り替えます。これにより、データ転送可能状態になります。

[5]

ソフトウェア処理　：MLS=WAIT=ACK=“0”を設定します。また、CKS2～0 ビット、IICX ビットおよび IEIC ビットを設定します。

目的　：MLS ビットでデータの繰出順序を MSB ファーストに、WAIT ビットでウェイト動作なしに、ACK ビットでアクノリッジメントモードに設定します。また、CKS2～0 ビット、IICX ビットの組み合わせにより、転送クロックの周波数を設定します。IEIC ビットは I²C バスインタフェースの割り込み要求の許可／禁止を設定します。

[6]

ソフトウェア処理　：MST=“0”、TRS=“0”に設定します。

目的　：スレーブ受信モードに設定します。

[7]

ソフトウェア処理　：ACKB ビットを“0”にセットします。

目的　：マスタデバイスがデータを受信後、アクノリッジを返すように設定します。

[8]

ハードウェア処理　：マスタデバイスの出力した開始条件を検出すると、BBSY ビットが“1”にセットされます。

目的　：バスが使用中であることを示します（マスタデバイスが第 1 バイトを出力しています）。

[9]

ハードウェア処理　：スレーブデバイスは、開始条件後の第 1 バイトでスレーブアドレスの一致を確認し、9 クロック目に IRIC ビットが“1”にセットされます。同時に SDA を“Low”レベルにしてアクノリッジを返します。またスレーブデバイスは、9 クロック目の受信クロックの立ち下がりから ICDR にデータをリードするまで、SCL を“Low”レベルに固定します。

目的　：IRIC ビット=“1”はスレーブアドレスの一致を知らせます。このとき、IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。

[10]

ソフトウェア処理　：ソフトウェアで IRIC を“0”にクリアします。

目的　：次の受信に備えます。

[11]

ソフトウェア処理　：ICDR をリードします。

目的　：スレーブデバイスは SCL ラインを解放し、次の受信が開始します。

[12]

ソフトウェア処理　：［9］～［11］を繰り返します。

目的　：データの受信を継続します。

[13]

ソフトウェア処理　：マスタデバイスの停止条件の発行により、SCL が“ High ”レベルのとき、SDA は“ Low ”レベルから“ High ”レベルに変化し、停止条件を検出すると、BBSY ビットが“ 0 ”に自動的にクリアされます。

目的　：データの受信を終了します。

2.6.4 スレーブ送信動作

スレーブ送信モードではスレーブデバイスが受信データを出力し、マスタデバイスが受信クロックを出力し、スレーブデバイスにアクノリッジを返します。

アドレッシングモードでは、最初にマスタデバイスがスレーブデバイスにスレーブアドレスを転送してきます。したがってこのとき、スレーブデバイスはスレーブ受信モードになっています。この後、引き続きデータを受信する場合は、**「2.6.3 スレーブ受信動作」**で示すようになります。しかし、スレーブデバイスがマスタデバイスにデータを送信する場合にはスレーブ送信モードに切り替える必要があります。

図 2.11 に、スレーブ送信モードの設定手順と動作を説明します。

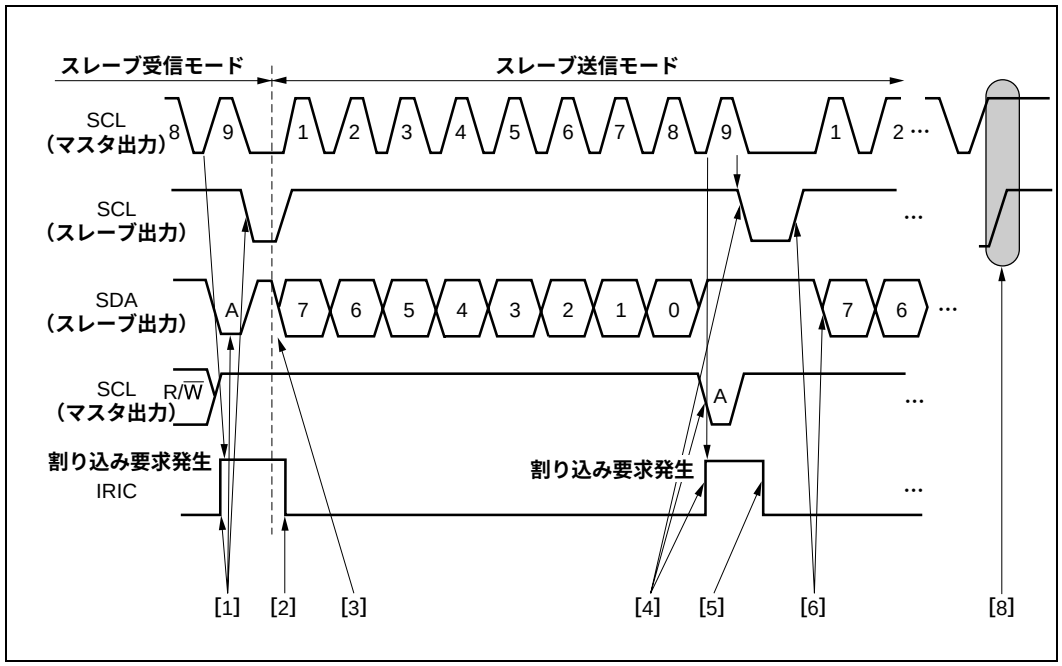


図 2.11 **スレーブ送信モード動作タイミング**（MLS=WAIT=ACK=“0”**のとき）**

スレーブ送信モード設定手順例

[1]

ハードウェア処理　：スレーブデバイスは、開始条件検出後の第 1 バイトでスレーブアドレスの一致を確認し、9 クロック目に IRIC ビットが“1”にセットされます。同時にスレーブデバイスは SDA ラインを“Low”レベルにし、アクノリッジを返します。また、受信データの 8 ビット目の R/$\overline{W}$ ビットが“1”のとき、TRS が“1”にセットされ、自動的にスレーブ送信モードになります。スレーブデバイスは 9 クロック目の送信クロックの立ち下がりから ICDR にデータをライトするまで、SCL ラインを“Low”レベルに固定します。

目的　：IRIC ビット=“1”はスレーブアドレスの一致を知らせます。

[2]

ソフトウェア処理　：IRIC ビットを“ 0 ”にクリアします。

目的　：次のデータ送信の準備です。

[3]

ソフトウェア処理　：ICDR にデータをライトします。

目的　：データ送信を開始させます。

ハードウェア処理　：スレーブデバイスは SCL ラインを“ High ”レベルに解放し、**図** 2.8 に示すタイミングでマスタデバイスが出力するクロックにしたがい、ICDR にライトされたデータを順次送り出します。

[4]

ハードウェア処理　：1 バイトのデータ送信が終了し、9 クロック目の送信クロックの立ち上がりで IRIC ビットが“ 1 ”にセットされます。スレーブデバイスはマスタデバイスからアクノリッジを受信し、ACKB=を“ 0 ”にセットします。また、スレーブデバイスは 9 クロック目の送信クロックの立ち下がりから ICDR にデータをライトするまで、SCL ラインを自動的に“ Low ”レベルに固定します。

目的　：IRIC ビット=“ 1 ”はデータ転送終了を知らせます。このとき、IEIC ビットが“ 1 ”にセットされていると、CPU に対し割り込み要求を発生します。マスタデバイスからのアクノリッジの有無確認は、ACKB ビットで行います。

[5]

ソフトウェア処理　：ソフトウェアで IRIC を“ 0 ”にクリアします。

目的　：次のデータ送信の準備です。

[6]

ソフトウェア処理　：次の送信データを ICDR にライトします。

目的　：スレーブデバイスは SCL ラインを“ High ”に解放し、データ送信を開始します。

[7]

ソフトウェア処理　：［4］から［6］を繰り返し行います。

目的　：データ送信を継続します。

[8]

ソフトウェア処理　：ICDR に H’FF をライトします。

目的　：マスタデバイスが停止条件を発行できるよう、SCL ラインを解放します。

ハードウェア処理　：SCL ラインが“ High ”に解放されます。
マスタデバイスの停止条件の発行により、SCL ラインが“ High ”レベルのとき、SDA ラインが“ Low ”レベルから“ High ”レベルに変化し、停止条件を検出すると、BBSY ビットが“ 0 ”に自動的にクリアされます。

2.7 H8S シリーズ（H8/2138 シリーズ）におけるデータ転送の動作説明［H8S 系］

2.7.1 マスタ送信動作

I²C バスフォーマットによるマスタ送信モードでは、マスタデバイスが送信クロック、送信データを出力し、スレーブデバイスがアクノリッジを返します。**図** 2.12 に、マスタ送信モードにおける設定手順と動作について説明します。

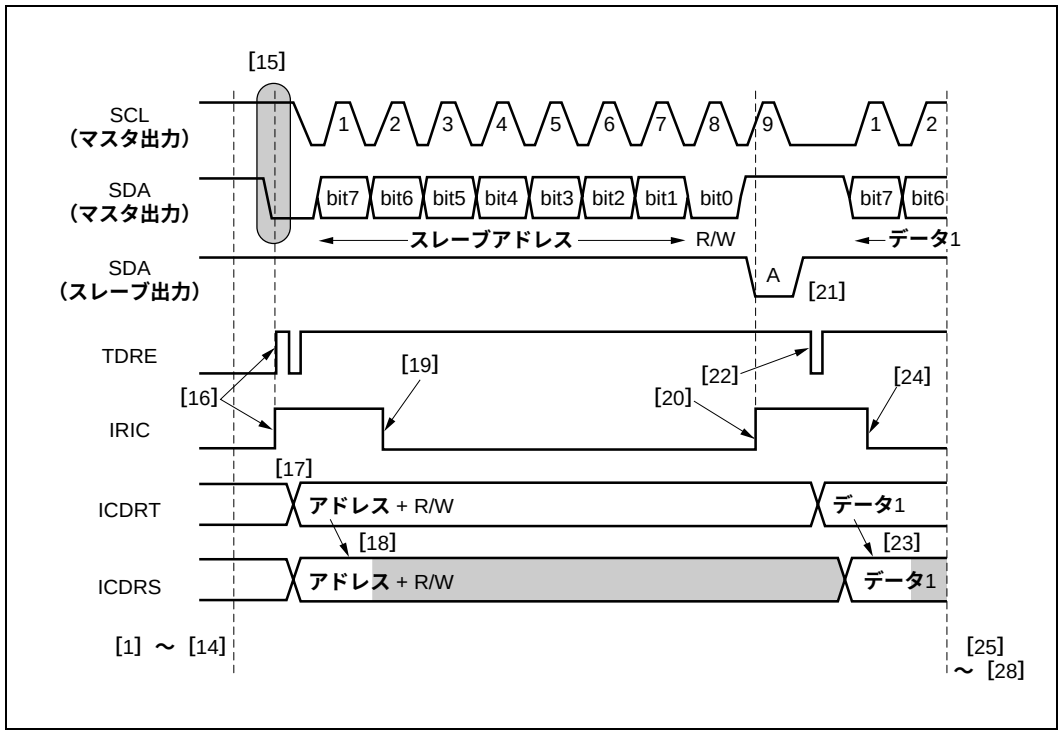


図 2.12 **マスタ送信モード動作タイミング**（MLS=WAIT=“ 0 ”**のとき**）

マスタ送信モード設定手順例

［1］初期設定 1

ソフトウェア処理 ：MSTPCRL の MSTP4 ビットまたは MSTP3 ビットを“ 0 ”にクリアします。

目的 ：IIC チャネル 0 または IIC チャネル 1 のモジュールストップモードを解除します。

［2］初期設定 2

ソフトウェア処理 ：STCR の IICE ビットを“ 1 ”にセットします。

目的 ：I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可します。

[3] 初期設定 3

ソフトウェア処理　：DDCSWR を設定します。

目的　：IIC チャネル 0 のフォーマットレスから I²C バスフォーマットへの自動切り替え機能の許可／禁止の選択、IIC チャネル 0 のフォーマットレス／I²C バスフォーマットの選択、IIC チャネル 0 でフォーマットの自動切り替えが実行された場合の CPU への割り込み要求の許可／禁止の選択を行います。

[4] 初期設定 4

ソフトウェア処理　：ICCR の ICE ビットを“0”にクリアします。

目的　：SAR および SARX へのアクセスを許可します。

[5] 初期設定 5

ソフトウェア処理　：SAR および SARX を設定します。

目的　：DDCSWR の SW ビットとともに転送フォーマットの設定、およびスレーブアドレスの設定を行います。

注意　：マスタモード時でもシステムがマルチマスタの場合、スレーブモードになる可能性があるのでスレーブアドレスを設定してください。

[6] 初期設定 6

ソフトウェア処理　：ICCR の ICE ビットを“1”にセットします。

目的　：ICMR および ICDR へのアクセスを許可し、IIC モジュールを転送可能状態に設定します。

[7] 初期設定 7

ソフトウェア処理　：ICSR の ACKB ビットを設定します。

目的　：受信時に出力するアクノリッジデータを設定します。

注意　：マスタモードで使用する場合でもバスアービトレーションロストすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[8] 初期設定 8

ソフトウェア処理　：STCR の IICX1 ビットまたは IICX0 ビット、および ICMR の CKS2～0 ビットを設定します。

目的　：使用する転送クロックの周波数を選択します。

[9] 初期設定 9

ソフトウェア処理　：ICMR の MLS ビット、WAIT ビットを“0”に設定します。

目的　：データ転送を MSB ファーストに、かつウェイト動作なしに設定します。

[10] 初期設定 10

ソフトウェア処理　：ICCR の ACKE ビットを設定します。

目的　：I²C バスフォーマットで受信デバイスから返されるアクノリッジビットの内容を無視して連続的に転送を行うか、アクノリッジビットが 1 ならば転送を中断してエラー処理等を行うかを選択します。

[11] 初期設定 11

ソフトウェア処理　：IICR の IEIC ビットを設定します。
目的　：I²C バスインタフェースから CPU に対する割り込み要求の許可／禁止を設定します。

[12] バス状態確認

ソフトウェア処理　：BBSY ビットをリードします。
目的　：バスが解放中か使用中かを確認します。
ハードウェア処理　：バスが解放されていれば BBSY=“0”に設定します。

[13] マスタ送信モード設定

ソフトウェア処理　：ICCR の MST ビットを“1”に、TRS ビットを“1”にセットします。
目的　：I²C バスインタフェースの動作モードを“マスタ送信モード”に設定します。

[14] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。
目的　：開始条件検出判定を行います。

[15] 開始条件発行

ソフトウェア処理　：ICCR の BBSY ビットを“1”にセットし、かつ SCP を“0”にクリアします。
目的　：開始条件を発行します。
注意　：BBSY ビットに“1”を、SCP ビットに“0”をライトする場合は、同時にセットする必要があるので、MOV 命令を使用してください。
ハードウェア処理　：SCL が“High”レベルの状態で、SDA は“High”から“Low”レベルに変化します。

[16] 開始条件成立確認

ソフトウェア処理　：IRIC ビットをリードします。
目的　：バスラインの状態から開始条件を検出したかを確認します。
ハードウェア処理　：開始条件が検出されていれば IRIC=“1”、TDRE=“1”に設定します。

[17] スレーブアドレス＋R/W データセット

ソフトウェア処理　：ICDR にスレーブアドレス+R/W データをライトします。
目的　：データ転送を開始します。
ハードウェア処理　：送信モードで ICDR に送信データをライトすると TDRE フラグを“0”にクリアします。

[18] ICDRT→ICDRS データ転送

ハードウェア処理　：TDRE フラグを“0”にクリアします。
目的　：ICDRT から ICDRS に送信データを転送します。

[19] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。
目的　：データ送信終了判定を行います。

[20] 1 バイトデータ送信終了

ハードウェア処理　：送信クロックの 9 クロック目の立ち上がり時に ICCR の IRIC ビットを“1”にセッ

トします。

目的　：IRIC=“1”はデータ送信終了、またはバスアービトレーションロストを示します。このとき、ICCR の IEIC ビットが“1”にセットされていれば、CPU への割り込み要求が発生します。

[21] アクノリッジ確認

ソフトウェア処理　：ICSR の ACKB ビットをリードします。

目的　：スレーブデバイスからのアクノリッジの有無を確認します。

ハードウェア処理　：スレーブデバイスから返されたアクノリッジを ACKB ビットにロードします。

[22] 送信データセット

ソフトウェア処理　：ICDR に送信データをライトします。

目的　：データ送信を開始します。

ハードウェア処理　：送信モードで ICDR に送信データをライトすると TDRE フラグを“0”にクリアします。

[23] ICDRT→ICDRS データ転送

ハードウェア処理　：TDRE フラグを“0”にクリアします。

目的　：ICDRT から ICDRS に送信データを転送します。

[24] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。

目的　：データ送信終了判定を行います。

[25] 1 バイトデータ送信終了

ハードウェア処理　：送信クロックの 9 クロック目の立ち上がり時に ICCR の IRIC ビットを“1”にセットします。

目的　：IRIC=“1”はデータ送信終了、またはバスアービトレーションロストを示します。このとき、ICCR の IEIC ビットが“1”にセットされていれば、CPU への割り込み要求を発生します。

[26] アクノリッジ確認

ソフトウェア処理　：ICSR の ACKB ビットをリードします。

目的　：スレーブデバイスからのアクノリッジの有無を確認します。

ハードウェア処理　：スレーブデバイスから返されたアクノリッジを ACKB ビットにロードします。

[27] データ送信継続

ソフトウェア処理　：［22］～［26］を繰り返します。

目的　：データ送信を継続します。

[28] 停止条件発行

ソフトウェア処理　：ICCR の BBSY ビットを“0”にクリアし、かつ SCP ビットを“0”にクリアします。

目的　：停止条件を発行します。

注意　：BBSY ビットに“0”、SCP ビットに“0”をライトする場合、同時にセットする必

要があるため、MOV 命令を使用してください。

ハードウェア処理　：バスラインの状態から停止条件を検出すると TDRE を“ 0 ”にクリアし、バスが解放されることにより、BBSY ビットを“ 0 ”にクリアします。

2.7.2　マスタ受信動作

I²C バスフォーマットによるマスタ送信モードでは、マスタデバイスが受信クロックを出力し、データを受信し、アクノリッジを返します。スレーブデバイスはデータを送信します。**図** 2.13 に、マスタ受信モードにおける設定手順と動作について説明します。

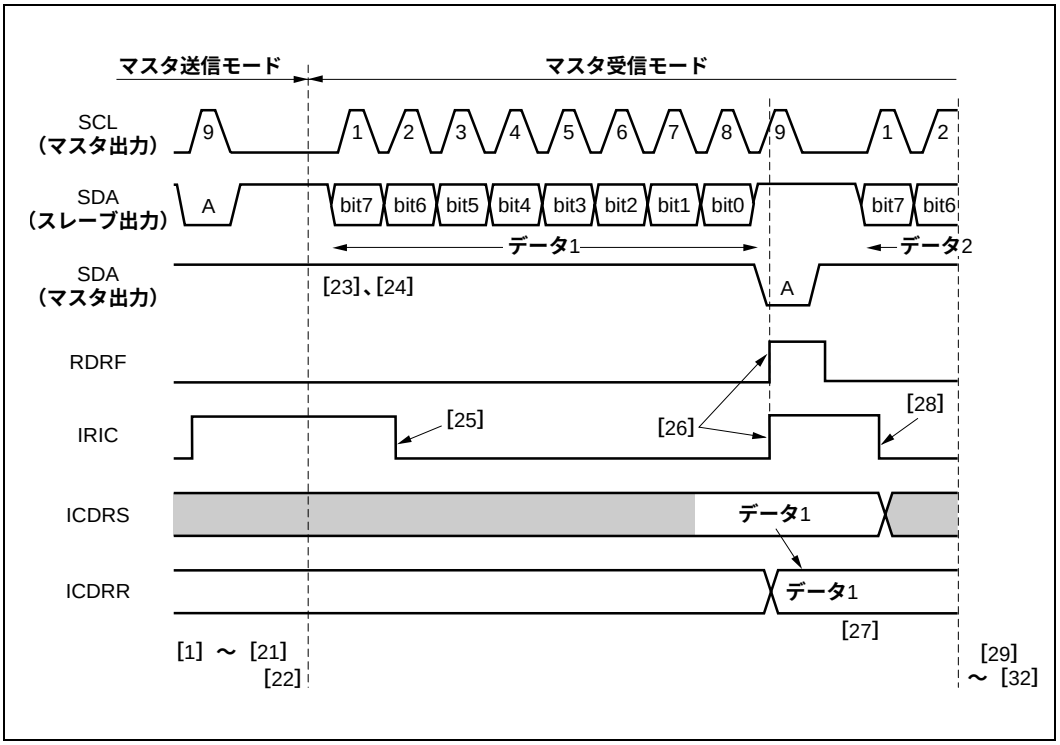


図 2.13　**マスタ受信モード動作タイミング（**MLS=WAIT=ACKB=“ 0 ”**のとき）**

マスタ受信モード設定手順例

[1] 初期設定 1

ソフトウェア処理　：MSTPCRL の MSTP4 ビットまたは MSTP3 ビットを“ 0 ”にクリアします。

目的　：IIC チャネル 0 または IIC チャネル 1 のモジュールストップモードを解除します。

[2] 初期設定 2

ソフトウェア処理　：STCR の IICE ビットを“ 1 ”にセットします。

目的　：I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可します。

[3] 初期設定 3

ソフトウェア処理　：DDCSWR を設定します。

目的　：IIC チャネル 0 のフォーマットレスから I²C バスフォーマットへの自動切り替え機能の許可／禁止の選択、IIC チャネル 0 のフォーマットレス／I²C バスフォーマットの選択、IIC チャネル 0 でフォーマットの自動切り替えが実行された場合の CPU への割り込み要求の許可／禁止の選択を行います。

[4] 初期設定 4

ソフトウェア処理　：ICCR の ICE ビットを“0”にクリアします。

目的　：SAR および SARX へのアクセスを許可します。

[5] 初期設定 5

ソフトウェア処理　：SAR および SARX を設定します。

目的　：DDCSWR の SW ビットとともに転送フォーマットの設定、およびスレーブアドレスの設定を行います。

注意　：マスタモード時でもシステムがマルチマスタの場合、スレーブモードになる可能性があるのでスレーブアドレスを設定してください。

[6] 初期設定 6

ソフトウェア処理　：ICCR の ICE ビットを“1”にセットします。

目的　：ICMR および ICDR へのアクセスを許可、IIC モジュールを転送可能状態に設定します。

[7] 初期設定 7

ソフトウェア処理　：ICSR の ACKB ビットを設定します。

目的　：受信時に出力するアクノリッジデータを設定します。

注意　：マスタモードで使用する場合でもバスアービトレーションロストすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[8] 初期設定 8

ソフトウェア処理　：STCR の IICX1 ビットまたは IICX0 ビット、および ICMR の CKS2～0 ビットを設定します。

目的　：使用する転送クロックの周波数を選択します。

[9] 初期設定 9

ソフトウェア処理　：ICMR の MLS ビット、WAIT ビットを“0”に設定します。

目的　：データ転送を MSB ファーストに、かつウェイト動作なしに設定します。

[10] 初期設定 10

ソフトウェア処理　：ICCR の ACKE ビットを設定します。

目的　：I²C バスフォーマットで受信デバイスから返されるアクノリッジビットの内容を無視して連続的に転送を行うか、アクノリッジビットが 1 ならば転送を中断してエラー処理等を行うかを選択します。

[11] 初期設定 11

ソフトウェア処理　：IICR の IEIC ビットを設定します。
目的　：I²C バスインタフェースから CPU に対する割り込み要求の許可／禁止を設定します。

[12] バス状態確認

ソフトウェア処理　：BBSY ビットをリードします。
目的　：バスが解放中か使用中かを確認します。
ハードウェア処理　：バスが解放されていれば BBSY=“0”に設定します。

[13] マスタ送信モード設定

ソフトウェア処理　：ICCR の MST ビットを“1”に、TRS ビットを“1”にセットします。
目的　：I²C バスインタフェースの動作モードを“マスタ送信モード”に設定します。

[14] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。
目的　：開始条件検出判定を行います。

[15] 開始条件発行

ソフトウェア処理　：ICCR の BBSY ビットを“1”にセットし、かつ SCP を“0”にクリアします。
目的　：開始条件を発行します。
注意　：BBSY ビットに“1”を、SCP ビットに“0”をライトする場合は、同時にセットする必要があるので、MOV 命令を使用してください。
ハードウェア処理　：SCL が“High”レベルの状態で、SDA が“High”から“Low”レベルに変化します。

[16] 開始条件成立確認

ソフトウェア処理　：IRIC ビットをリードします。
目的　：バスラインの状態から開始条件を検出したかを確認します。
ハードウェア処理　：開始条件が検出されていれば IRIC=“1”、TDRE=“1”に設定します。

[17] スレーブアドレス＋R/W データセット

ソフトウェア処理　：ICDR にスレーブアドレス+R/W データをライトします。
目的　：データ転送を開始します。
ハードウェア処理　：送信モードで ICDR に送信データをライトすると TDRE フラグを“0”にクリアします。

[18] ICDRT→ICDRS データ転送

ハードウェア処理　：TDRE フラグを“0”にクリアします。
目的　：ICDRT から ICDRS に送信データを転送します。

[19] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。
目的　：データ送信終了判定を行います。

[20] 1 バイトデータ送信終了

ハードウェア処理　：送信クロックの 9 クロック目の立ち上がり時に ICCR の IRIC ビットを“1”にセット

します。

目的　：IRIC=“1”はデータ送信終了、またはバスアービトレーションロストを示します。このとき、ICCR の IEIC ビットが“1”にセットされていれば、CPU への割り込み要求を発生します。

[21] アクノリッジ確認

ソフトウェア処理　：ICSR の ACKB ビットをリードします。

目的　：スレーブデバイスからのアクノリッジの有無を確認します。

ハードウェア処理　：スレーブデバイスから返されたアクノリッジを ACKB ビットにロードします。

[22] マスタ受信モード設定

ソフトウェア処理　：ICCR の TRS ビットを“0”にクリアします。

目的　：マスタ送信モードからマスタ受信モードに切り替えます。

[23] ACKB=0

ソフトウェア処理　：ICSR の ACKB ビットを“0”にクリアします。

目的　：アクノリッジ出力タイミングで 0 を出力します。

[24] ダミーリード

ソフトウェア処理　：ICDR をリードします。

目的　：データ受信を開始します。

[25] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。

目的　：データ受信終了判定を行います。

[26] 1 バイトデータ受信終了

ハードウェア処理　：受信クロックの 9 クロック目の立ち上がり時に ICCR の IRIC ビットを“1”に、RDRF フラグを“1”にセットします。

目的　：IRIC=“1”はデータ転送終了を知らせます。このとき、IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。また、RDRF 内部フラグが“0”にクリアされていると、RDRF 内部フラグを“1”にセットして引き続き受信動作を行います。

[27] 受信データリード

ソフトウェア処理　：ICDR をリードします。

目的　：データ受信を開始します。

ハードウェア処理　：RDRF フラグを“0”にクリアします。

注意　：最終バイト受信後にアクノリッジを返さない場合は、ICDR をリードする前に ACKB を“1”にセットしてください。

[28] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。

目的　：データ受信終了判定を行います。

[29] データ受信継続

ソフトウェア処理　：［26］～［28］を繰り返します。
目的　：データ受信を継続します。

[30] 受信終了

ソフトウェア処理　：ICCR の TRS ビットを“1”にセットします。
目的　：受信を終了する場合には、次のフレームの受信クロックが立ち上がる前に ICCR の TRS ビットを“1”にセットしておきます。
ハードウェア処理　：TDRE フラグを“1”にセットします。

[31] 最終データリード

ソフトウェア処理　：ICDR をリードします。
目的　：受信データの最終バイトをリードします。

[32] 停止条件発行

ソフトウェア処理　：ICCR の BBSY ビットを“0”にクリアし、かつ SCP ビットを“0”にクリアします。
目的　：停止条件を発行します。
注意　：BBSY ビットに“0”を、SCP ビットに“0”をライトする場合、同時にセットする必要があるため、MOV 命令を使用してください。
ハードウェア処理　：バスラインの状態から停止条件を検出すると TDRE を“0”にクリアし、バスが解放されることにより、BBSY ビットを“0”にクリアします。

2.7.3 スレーブ受信動作

I²C バスフォーマットによるスレーブ受信モードでは、マスタデバイスが送信クロック、送信データを出力し、スレーブデバイスがアクノリッジを返します。**図** 2.14 に、スレーブ受信モードにおける設定手順と動作について説明します。

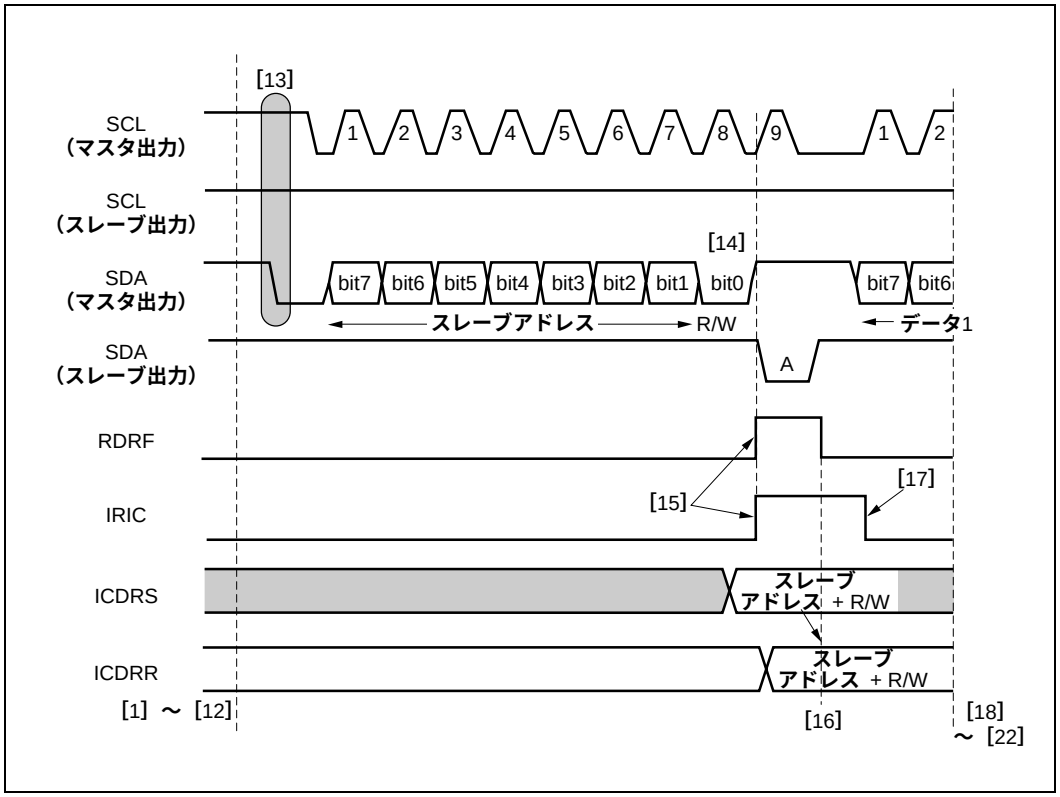


図 2.14 **スレーブ受信モード動作タイミング**（MLS=ACKB=“0”**のとき**）

スレーブ受信モード設定手順例

[1] 初期設定 1

ソフトウェア処理　：MSTPCRL の MSTP4 ビットまたは MSTP3 ビットを“0”にクリアします。

目的　：IIC チャネル 0 または IIC チャネル 1 のモジュールストップモードを解除します。

[2] 初期設定 2

ソフトウェア処理　：STCR の IICE ビットを“1”にセットします。

目的　：I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可します。

[3] 初期設定 3

ソフトウェア処理　：DDCSWR を設定します。

目的　：IIC チャネル 0 のフォーマットレスから I²C バスフォーマットへの自動切り替え機能の許可／禁止の選択、IIC チャネル 0 のフォーマットレス／I²C バスフォーマットの選択、IIC チャネル 0 でフォーマットの自動切り替えが実行された場合の CPU への割り込み要求の許可／禁止の選択を行います。

[4] 初期設定 4

ソフトウェア処理　：ICCR の ICE ビットを“0”にクリアします。

目的　：SAR および SARX へのアクセスを許可します。

[5] 初期設定 5

ソフトウェア処理　：SAR および SARX を設定します。

目的　：DDCSWR の SW ビットとともに転送フォーマットの設定、およびスレーブアドレスの設定を行います。

[6] 初期設定 6

ソフトウェア処理　：ICCR の ICE ビットを“1”にセットします。

目的　：ICMR および ICDR へのアクセスを許可、IIC モジュールを転送可能状態に設定します。

[7] 初期設定 7

ソフトウェア処理　：ICSR の ACKB ビットを設定します。

目的　：受信時に出力するアクノリッジデータを設定します。

注意　：マスタモードで使用する場合でもバスアービトレーションロストすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[8] 初期設定 8

ソフトウェア処理　：STCR の IICX1 ビットまたは IICX0 ビット、および ICMR の CKS2～0 ビットを設定します。

目的　：使用する転送クロックの周波数を選択します。

[9] 初期設定 9

ソフトウェア処理　：ICMR の MLS ビット、WAIT ビットを“0”に設定します。

目的　：データ転送を MSB ファーストに、かつウェイト動作なしに設定します。

[10] 初期設定 10

ソフトウェア処理　：ICCR の ACKE ビットを設定します。

目的　：I²C バスフォーマットで受信デバイスから返されるアクノリッジビットの内容を無視して連続的に転送を行うか、アクノリッジビットが 1 ならば転送を中断してエラー処理等を行うかを選択します。

[11] 初期設定 11

ソフトウェア処理　：IICR の IEIC ビットを設定します。

目的　　：I²C バスインタフェースから CPU に対する割り込み要求の許可／禁止を設定します。

[12] 初期設定 12

ソフトウェア処理　：MST ビットを“0”に、TRS ビットを“0”に設定します。

目的　　：スレーブ受信モードに設定します。

[13] 開始条件検出

ハードウェア処理　：ICCR の BBSY ビットを“1”にセットします。

目的　　：マスタデバイスが出力した開始条件を検出します。

[14] スレーブアドレス受信

ハードウェア処理　：ICCR の TRS ビットを“0”にクリアします。

目的　　：開始条件後の第 1 フレームでスレーブアドレスが一致したとき、マスタデバイスに指定されたスレーブデバイスとして動作、8 ビット目のデータ（R/W）が“0”のとき、ICCR の TRS ビットは“0”のまま変化せず、スレーブ受信動作を行います。

[15] スレーブアドレス一致

ハードウェア処理　：受信フレームの 9 クロック目でスレーブデバイスは SDA を“Low”レベルにし、アクノリッジを返します。同時に ICCR の IRIC ビットを“1”に、RDRF フラグを“1”にセットします。

目的　　：IRIC=“1”はスレーブアドレスの一致を知らせます。このとき、ICCR の IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。

[16] ダミーリード

ソフトウェア処理　：ICDR をリード（ダミーリード）します。

目的　　：受信動作を開始します。

[17] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。

目的　　：受信終了判定を行います。

[18] 受信終了

ハードウェア処理　：受信フレームの 9 クロック目でスレーブデバイスは SDA を“Low”レベルにし、アクノリッジを返します。同時に ICCR の IRIC ビットを“1”に、RDRF フラグを“1”にセットします。

目的　　：IRIC=“1”は転送終了を知らせます。このとき、ICCR の IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。

[19] 受信データリード

ソフトウェア処理　：ICDR をリードします。

目的　　：受信データをリードします。

ハードウェア処理　：ICDR（ICDRR）の受信データをリードすることにより RDRF フラグを“0”にクリアします。

［20］IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“ 0 ”にクリアします。
目的　：受信終了判定を行います。

［21］受信継続

ソフトウェア処理　：［18］～［20］を繰り返します。
目的　：受信動作を継続します。

［22］受信動作終了

ハードウェア処理　：SCL が“ High ”レベルのとき、SDA が“ Low ”レベルから“ High ”レベルに変化し、ICCR の BBSY を“ 0 ”にクリアします。
目的　：マスタデバイスの発行した停止条件を検出します。

2.7.4 スレーブ送信動作

I²C バスフォーマットによるスレーブ送信モードでは、スレーブデバイスが送信データを出力し、マスタデバイスが受信クロックを出力し、アクノリッジを返します。**図** 2.15 に、スレーブ送信モードにおける設定手順と動作について説明します。

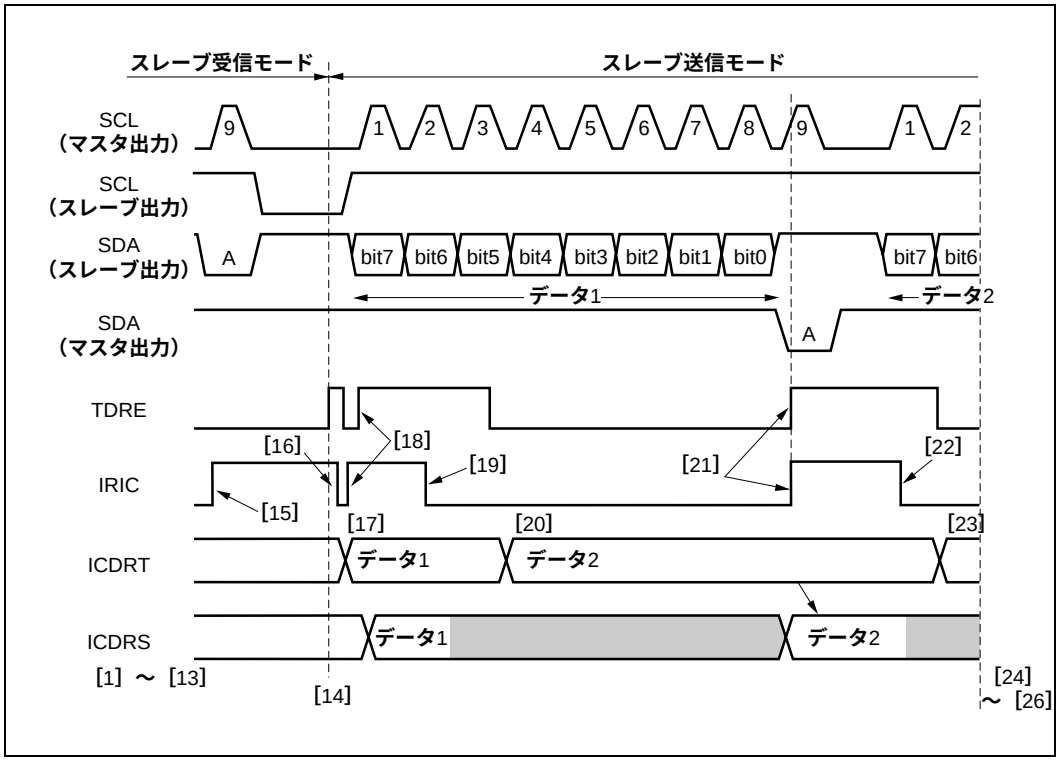


図 2.15 **スレーブ送信モード動作タイミング例（**MLS=“ 0 ”**の場合）**

スレーブ送信モード設定手順例

[1] 初期設定 1

ソフトウェア処理　：MSTPCRL の MSTP4 ビットまたは MSTP3 ビットを“ 0 ”にクリアします。

目的　：IIC チャネル 0 または IIC チャネル 1 のモジュールストップモードを解除します。

[2] 初期設定 2

ソフトウェア処理　：STCR の IICE ビットを“ 1 ”にセットします。

目的　：I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可します。

[3] 初期設定 3

ソフトウェア処理　：DDCSWR を設定します。

目的　：IIC チャネル 0 のフォーマットレスから I²C バスフォーマットへの自動切り替え機能の許可／禁止の選択、IIC チャネル 0 のフォーマットレス／I²C バスフォーマットの選択、IIC チャネル 0 でフォーマットの自動切り替えが実行された場合の CPU への割り込み要求の許可／禁止の選択を行います。

[4] 初期設定 4

ソフトウェア処理　：ICCR の ICE ビットを“0”にクリアします。
目的　：SAR および SARX へのアクセスを許可します。

[5] 初期設定 5

ソフトウェア処理　：SAR および SARX を設定します。
目的　：DDCSWR の SW ビットとともに転送フォーマットの設定、およびスレーブアドレスの設定を行います。

[6] 初期設定 6

ソフトウェア処理　：ICCR の ICE ビットを“1”にセットします。
目的　：ICMR および ICDR へのアクセスを許可し、IIC モジュールを転送可能状態に設定します。

[7] 初期設定 7

ソフトウェア処理　：ICSR の ACKB ビットを設定します。
目的　：受信時に出力するアクノリッジデータを設定します。
注意　：マスタモードで使用する場合でもバスアービトレーションロストすると自動的にスレーブ受信モードに遷移するので、必ず ACKB ビットの設定を行ってください。

[8] 初期設定 8

ソフトウェア処理　：STCR の IICX1 ビットまたは IICX0 ビット、および ICMR の CKS2～0 ビットを設定します。
目的　：使用する転送クロックの周波数を選択します。

[9] 初期設定 9

ソフトウェア処理　：ICMR の MLS ビット、WAIT ビットを“0”に設定します。
目的　：データ転送を MSB ファーストに、かつウェイト動作なしに設定します。

[10] 初期設定 10

ソフトウェア処理　：ICCR の ACKE ビットを設定します。
目的　：I²C バスフォーマットで受信デバイスから返されるアクノリッジビットの内容を無視して連続的に転送を行うか、アクノリッジビットが 1 ならば転送を中断してエラー処理等を行うかを選択します。

[11] 初期設定 11

ソフトウェア処理　：IICR の IEIC ビットを設定します。
目的　：I²C バスインタフェースから CPU に対する割り込み要求の許可／禁止を設定します。

[12] 初期設定 12

ソフトウェア処理　：MST ビットを“0”に、TRS ビットを“0”に設定します。
目的　：スレーブ受信モードに設定します。

[13] 開始条件検出

ハードウェア処理　：ICCR の BBSY ビットを“1”にセットします。
目的　：マスタデバイスが出力した開始条件を検出します。

[14] スレーブアドレス受信

ハードウェア処理　：ICCR の TRS ビットを“0”にクリアし、TDRE フラグを“1”にセットします。
目的　：開始条件後の第 1 フレームでスレーブアドレスが一致したとき、マスタデバイスに指定されたスレーブデバイスとして動作し、8 ビット目のデータ（R/W）が“1”のとき、ICCR の TRS ビットを“1”にセットし、自動的にスレーブ送信モードに変化します。

[15] スレーブアドレス一致

ハードウェア処理　：受信フレームの 9 クロック目でスレーブデバイスは SDA を“Low”レベルにし、アクノリッジを返します。同時に ICCR の IRIC ビットを“1”にセットし、スレーブデバイスは送信クロックの立ち下がりから ICDR にデータをライトするまで SCL を“Low”レベルに固定します。
目的　：IRIC=“1”はスレーブアドレスの一致を知らせします。このとき、ICCR の IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。

[16] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。
目的　：ICDRT→ICDRS へのデータ転送判定します。

[17] 1 バイト目送信データライト

ソフトウェア処理　：ICDR に 1 バイト目の送信データをライトします。
目的　：データ送信を開始します。
ハードウェア処理　：TDRE フラグを“0”にクリアします。

[18] ICDRT→ICDRS データ転送

ハードウェア処理　：TDRE フラグを“1”に、ICCR の IRIC ビットを“1”に、ICSR の IRTR を“1”にセットします。
目的　：ICDRT にライトされたデータを ICDRS に転送します。

[19] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリア
目的　：送信終了を判定します。

[20] 送信データライト

ソフトウェア処理　：ICDR に送信データをライトします。
目的　：データ送信を開始します。

ハードウェア処理　：TDRE フラグを“0”にクリアします。

[21] 送信終了

ハードウェア処理　：1 フレームのデータ送信が終了し、送信クロックの 9 クロック目の立ち上がり時に、ICCR の IRIC ビットを“1”にセットし、スレーブデバイスはマスタデバイスからのアクノリッジを受信し ACKB ビットに格納します。また、スレーブデバイスは送信クロックの 9 クロック目の立ち下がりから ICDR にデータをライトするまで、SCL を自動的に“Low”レベルに固定します。

目的　：IRIC=“1”はデータ転送終了を知らせる。このとき、ICCR の IEIC ビットが“1”にセットされていると、CPU に対し割り込み要求を発生します。マスタデバイスからのアクノリッジの確認は ACKB ビットをリードすることにより行います。

[22] IRIC クリア

ソフトウェア処理　：ICCR の IRIC ビットを“0”にクリアします。

目的　：送信終了を判定します。

[23] 送信データライト

ソフトウェア処理　：ICDR に送信データをライトします。

目的　：スレーブデバイスは SCL を“High”に解放し、データ送信を開始します。

[24] データ送信継続

ソフトウェア処理　：［21］～［23］を繰り返します。

目的　：データ送信動作を継続します。

[25] 送信終了

ソフトウェア処理　：ICCR の TRS ビットを“0”にクリアし、ICDR をリード（ダミーリード）します。

目的　：TRS ビット“0”にクリアすることによりスレーブ受信モードに設定、ICDR をダミーリードすることにより SCL ラインを解放します。

[26] 停止条件検出

ハードウェア処理　：SCL が“High”レベルのときに、SDA が“Low”レベルから“High”レベルに変化し、ICCR の BBSY ビットを“0”にクリアします。

目的　：マスタデバイスが発行した停止条件を検出します。

3. H8/300、H8/300L シリーズ応用例

3.1 システム仕様

システム仕様を以下に示します。また、システムの構成を**図** 3.1 に示します。

- マスタ 2 個、スレーブ 1 個のマルチマスタ構成とします。デバイスは H8/3434F（フラッシュメモリ内蔵）を使用します。
- スイッチ 1（SW1：マスタ 1 側）を押すと「CPU1」、スイッチ 2（SW2：マスタ 2 側）を押すと「CPU2」を 8 セグメント LED に表示します。

1. 各スイッチを押すと、マスタ 1、2 はそれぞれデータ“ H'01 ”、“ H'02 ”をスレーブに送信します。
2. スレーブは受信データを判別し、“ H'01 ”なら「CPU1」、“ H'02 ”なら「CPU2」を 8 セグメント LED に表示します。

- I^2C バス転送レートは 200kbps とします。

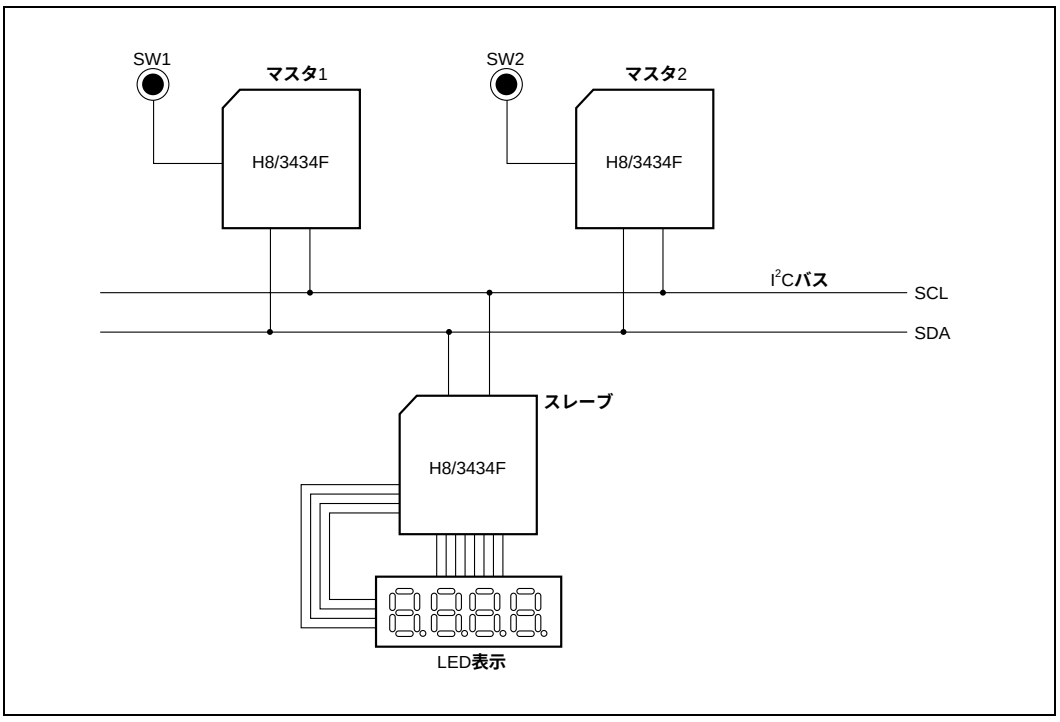


図 3.1 **マルチマスタ評価システム**

- H8/300 シリーズ、H8/300L シリーズ内蔵の I²C バスインタフェースでは、「1.4 **通信調整手順**」で示した通信調整手順のほかに**図** 3.2 に示す調整手順が行われます。各マスタ装置は、SCL の立ち下がりでバスラインをモニタし、自分のレベルと一致しない場合、出力段をオフにします。

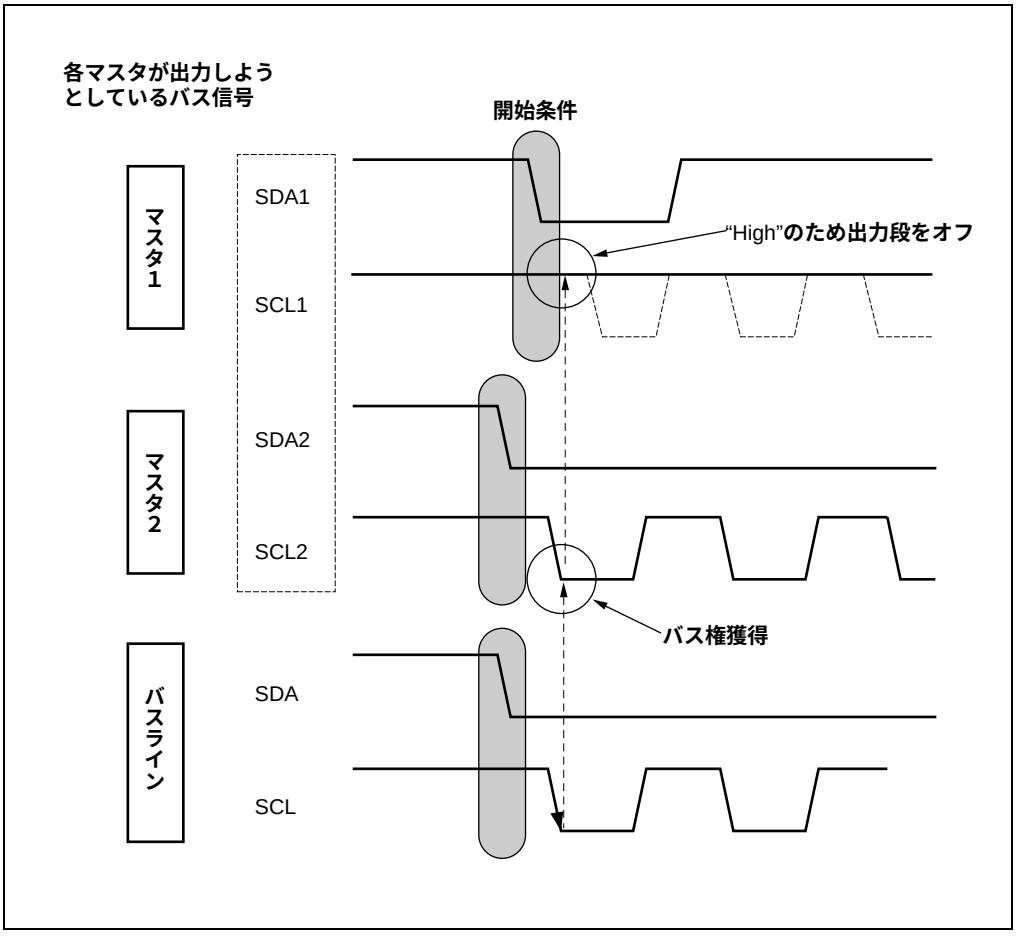


図 3.2 **バスアービトレーションの検出方法**

- マスタ 1、2 が同時にデータ送信を開始した場合（マルチマスタ動作）

1. 衝突検出時、マスタ 1（送信データ：H'01）は、マスタ 2（送信データ：H'02）よりデータライン（SDA）の“ Low ”期間が長いのでマスタ 1 がバス権を獲得します（**図** 3.3 参照）。

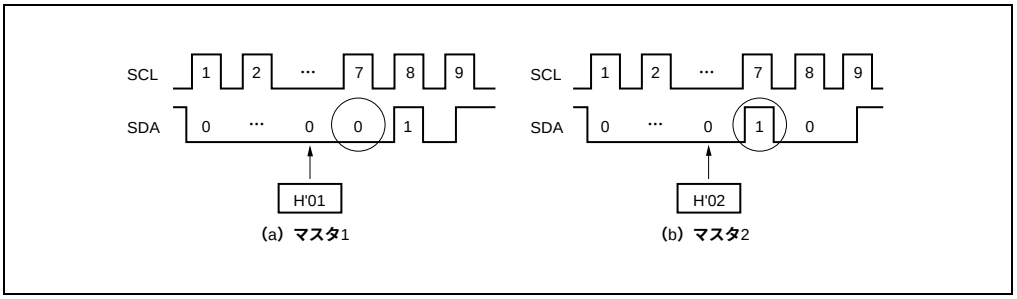


図 3.3 **マスタ** 1 **がバス権を獲得する理由**

2. マスタ 2 はアービトレーションを失い、自動的にスレーブ受信モード遷移します。再びマスタ 2 をマスタ送信モードで使用するためには、再設定が必要です。また、送信できなかったデータは、再度 ICDR に書き込む必要があります。このため、本システムではマスタ 2 のバスアービトレーションロストを確認後、スイッチ入力にかかわらず再度データ送信ルーチンを呼び出すようにしてあります。

3.2 マルチマスタ評価システムの回路

図 3.4 にマルチマスタ構成を持つマルチマスタ評価システムの回路図を示します。

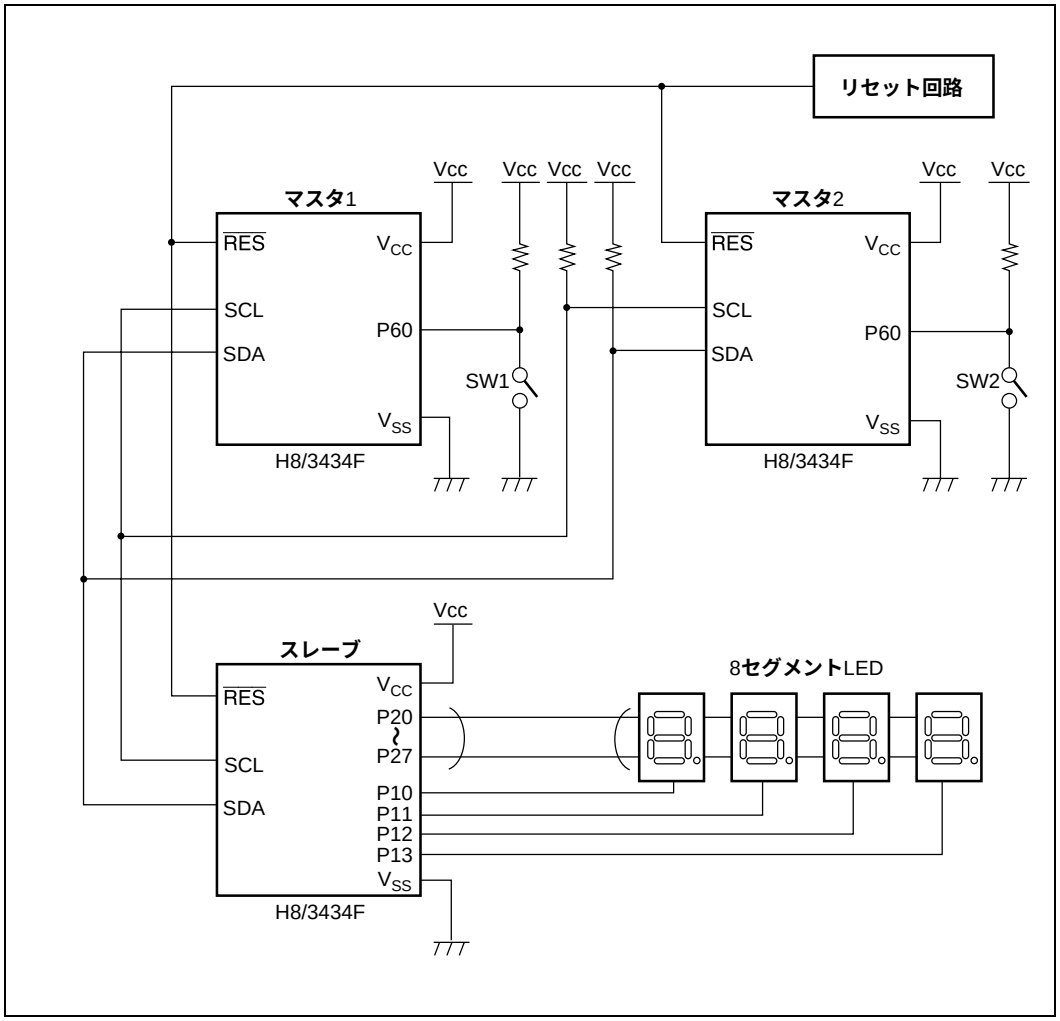


図 3.4 I²C **バス簡易評価システムの回路図**

3.3 ソフトウェアの設計

3.3.1 モジュール説明

マルチマスタ構成を持つシステムのソフトウェア例を示します。以下に分割したプログラムモジュールとその機能を表 3.1 に示します。

表 3.1 モジュール説明

モジュール名	ラベル名	機能
マスタ メインプログラム	Main	(1) 初期設定（スタックポインタ、I²C バスインタフェース、8 ビットタイマ） (2) 割り込みの許可 (3) スイッチの監視とマスタサブルーチンの呼び出し
キースキャンプログラム （割り込みプログラム）	Compare	8 ビットタイマのコンペアマッチ割り込みにより、8ms ごとに I/O ポート 6 のビットを読む
データ送信プログラム	Master	バスの監視とデータの送信
スレーブ メインプログラム	_Main	(1) 初期設定（スタックポインタ、I²C バスインタフェース、8 ビットタイマ） (2) 割り込みの許可 (3) スレーブサブルーチンの呼び出し
8 セグメント LED 点灯プログラム	_Display	8 セグメント LED 表示
データ受信プログラム （割り込みプログラム）	_Receive	(1) データの受信と判定 (2) 8 セグメント LED のデータテーブルの切り替え

3.3.2 マスタ

(1) 使用内部レジスタ説明

表 3.2 マスタ使用内部レジスタ説明

レジスタ	機能	使用モジュール名
STCR	8 ビットタイマの入力クロック選択	データ送信プログラム
ICCR	I^2C バスインタフェースを動作可能状態に設定 割り込みの設定 通信モード選択 アクノリッジメントモード選択 入力クロック周波数の選択	
ICSR	開始／停止条件の発行 アクノリッジの認識と制御	
ICDR	送信／受信データを格納	
ICMR	MSB ファースト／LSB ファーストの選択	
SAR	スレーブアドレスの格納とフォーマットセレクト	
TCR	クロック入力選択 カウンタのクリア条件の選択 コンペアマッチ A 割り込みの許可	メインプログラム
TCSR	コンペアマッチフラグのクリア	
TCORA	コンペアマッチ時間の設定	
P6DR	スイッチ入力ポート	
P6DDR	ポートのモード設定	

(2) 使用汎用レジスタ説明

表 3.3 マスタ使用汎用レジスタ説明

レジスタ	機能	使用モジュール名
R1L,R2L	ワーキングレジスタ	メインプログラム
R3L	送信データー時退避用	データ送信プログラム
R5L	送信データバイト数のカウント用	データ送信プログラム
CCR	割り込みフラグチェック用	メインプログラム

(3) 使用 RAM 説明

表 3.4 マスタ使用 RAM 説明

レジスタ	機能	データ長	使用モジュール名
Switch	チャタリングのカウント用	1 バイト	キースキャンプログラム （割り込みプログラム）

(4) 使用 ROM 説明

表 3.5 マスタ使用 ROM 説明

ラベル名	機能	データ長	使用モジュール名
Table	送信データ格納用	2 バイト	データ送信プログラム

3.3.3 スレーブ

(1) 使用内部レジスタ説明

表 3.6 スレーブ使用内部レジスタ説明

レジスタ	機能	使用モジュール名
STCR	8 ビットタイマの入力クロック選択	メインプログラム
ICCR	I^2C バスインタフェースを動作可能状態に設定 割り込みの設定 通信モード選択 アクノリッジメントモード選択 入力クロック周波数の選択	
ICSR	データの送受信の監視および割り込みの発生チェック	データ受信プログラム （割り込みプログラム）
ICDR	送信／受信データを格納	
ICMR	MSB ファースト／LSB ファーストの選択	
SAR	スレーブアドレスの格納とフォーマットセレクト	
TCR	クロック入力選択 カウンタのクリア条件の選択（コンペアマッチ A でカウンタをクリア）	メインプログラム
TCSR	コンペアマッチフラグの状態チェック	
TCORA	コンペアマッチ時間 A の設定	
TCORB	コンペアマッチ時間 B の設定	
P1DDR	ポート 1 のモード設定	
P1DR	8 セグメント LED ディジットデータ	8 セグメント LED 点灯プログラム
P2DDR	ポート 2 のモード設定	メインプログラム
P2DR	8 セグメント LED セグメントデータ	8 セグメント LED 点灯プログラム

(2) 使用汎用レジスタ説明

表 3.7 スレーブ使用汎用レジスタ説明

レジスタ	機能	使用モジュール名
R1L	ワーキングレジスタ	メインプログラム
R1L	ワーキングレジスタ	8 セグメント LED 点灯プログラム
R6	データテーブル入れ替え用	
R1L	ワーキングレジスタ	データ受信プログラム （割り込みプログラム）
R4	データテーブルのセット用	
CCR	割り込みフラグチェック用	メインプログラム

(3) 使用 RAM 説明

表 3.8 スレーブ使用 RAM 説明

レジスタ	機能	データ長	使用モジュール名
_TABLE	データテーブルの先頭アドレス格納用	1 ワード	データ受信プログラム（割り込みプログラム）
_Count	LED 表示時間管理用	1 バイト	8 セグメント LED 点灯プログラム
_Count2	LED 表示時間管理用	1 バイト	
_D_DATA	ディジットデータの初期値	1 バイト	
_First	1 バイト目受信データ格納用	1 バイト	データ受信プログラム（割り込みプログラム）
_Second	2 バイト目受信データ格納用	1 バイト	

(4) 使用 ROM 説明

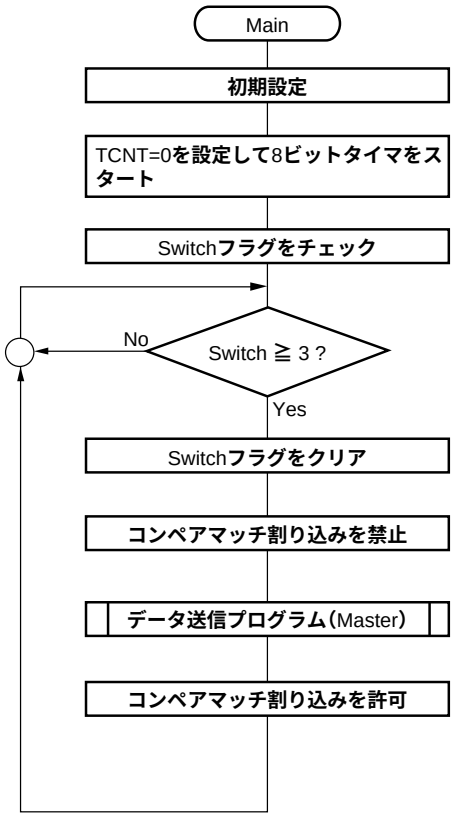
表 3.9 スレーブ使用 ROM 説明

レジスタ	機能	データ長	使用モジュール名
_Table1	8 セグメントデータ格納用	1 バイト	データ受信プログラム（割り込みプログラム）
_Table2	8 セグメントデータ格納用	1 バイト	
_Table3	8 セグメントデータ格納用	1 バイト	
_Table4	8 セグメントデータ格納用	1 バイト	

3.4 フローチャート

3.4.1 マスタプログラム

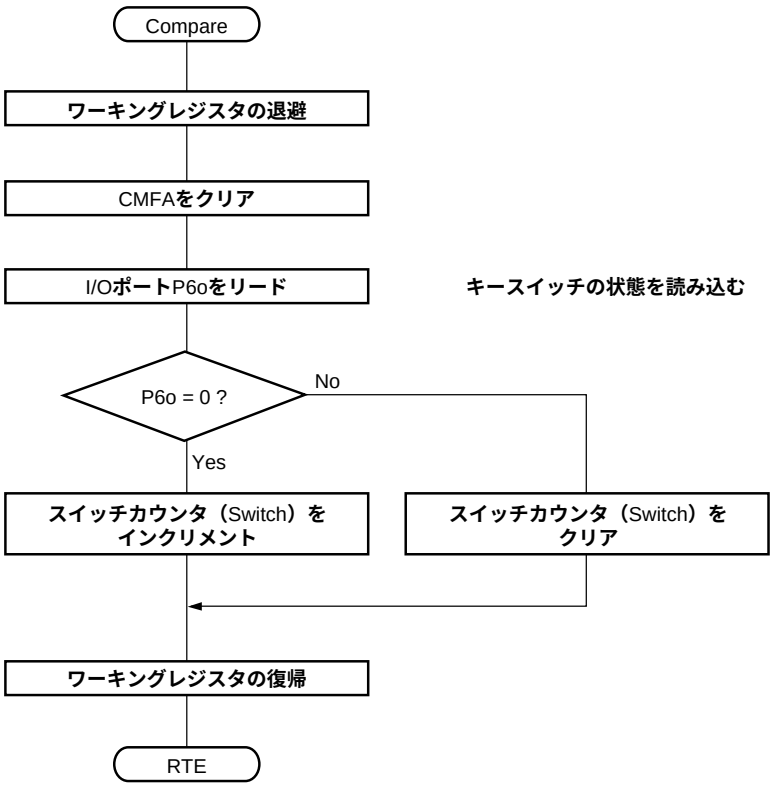
(1) メインプログラム



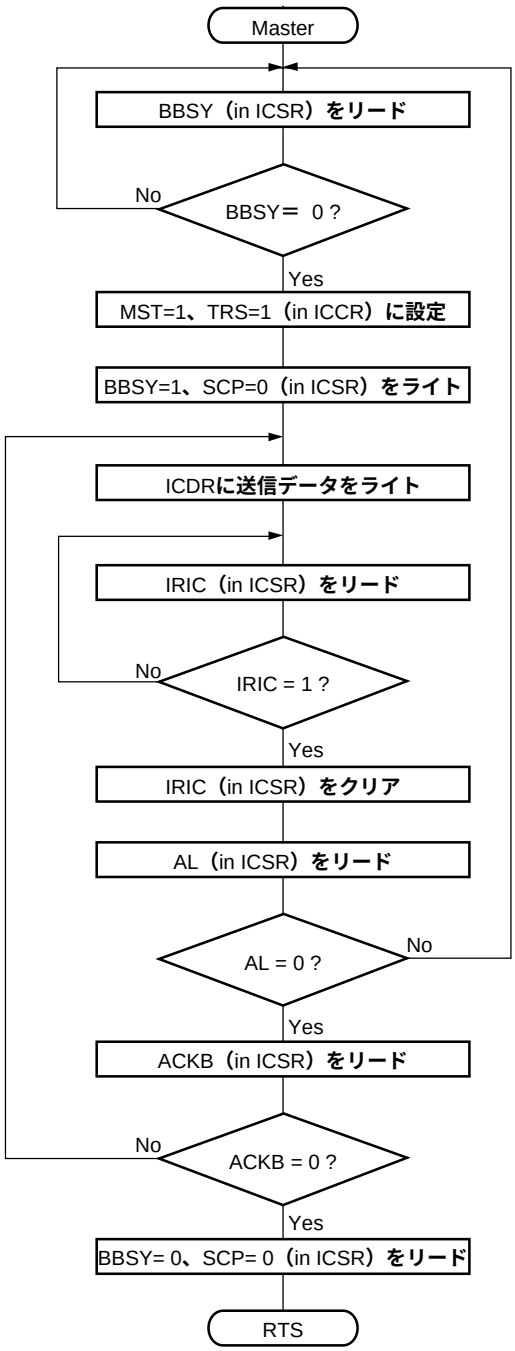
初期設定

- 割り込みベクタの設定
- 変数の領域の確保
- スタックポインタの設定
- I^2C バス I/F の初期設定
- 8 ビットタイマの初期設定
- I/O ポートの初期設定
- 割り込みの許可

(2) キースキャンプログラム（割り込みプログラム）

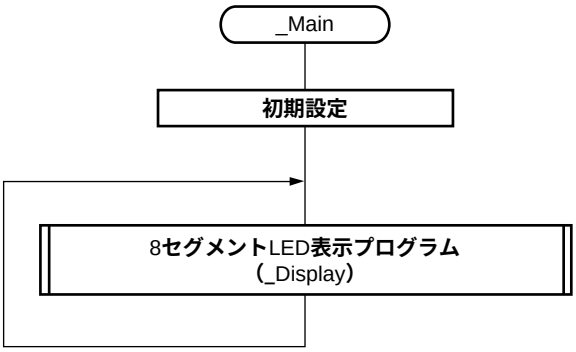


(3) データ送信プログラム



3.4.2 スレーブプログラム

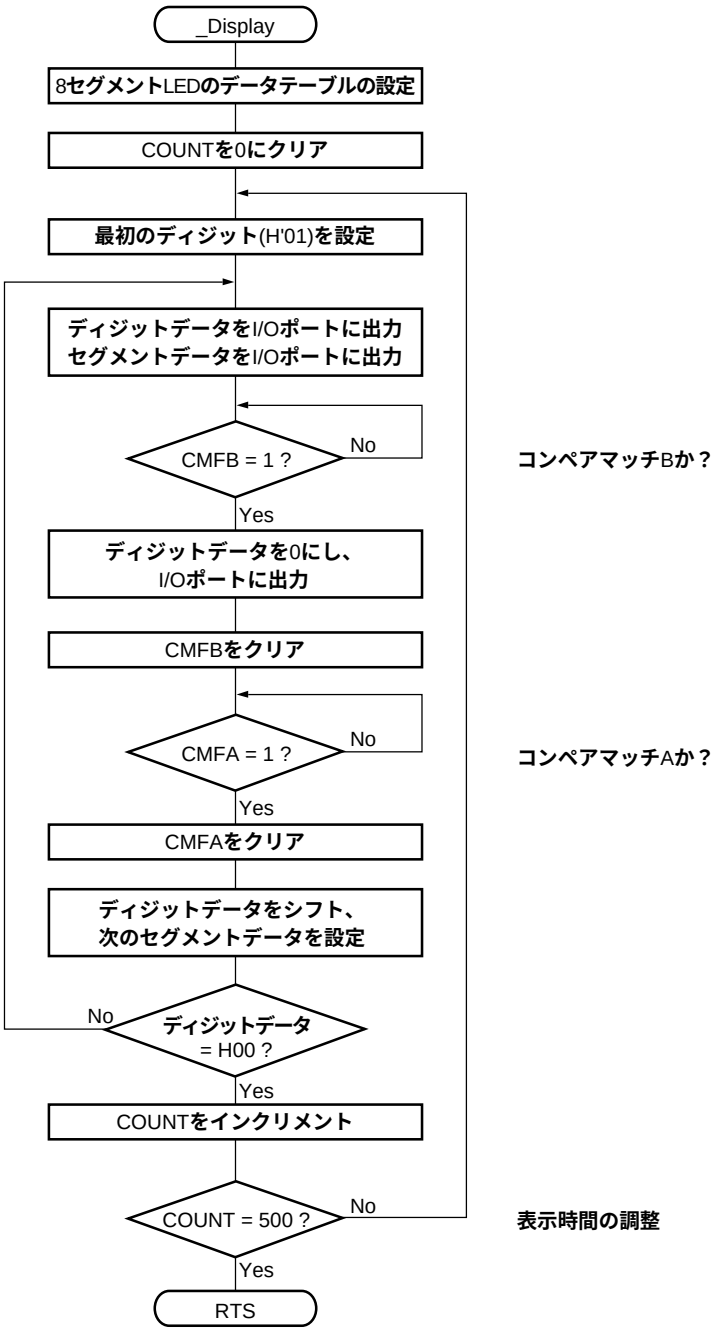
(1) メインプログラム



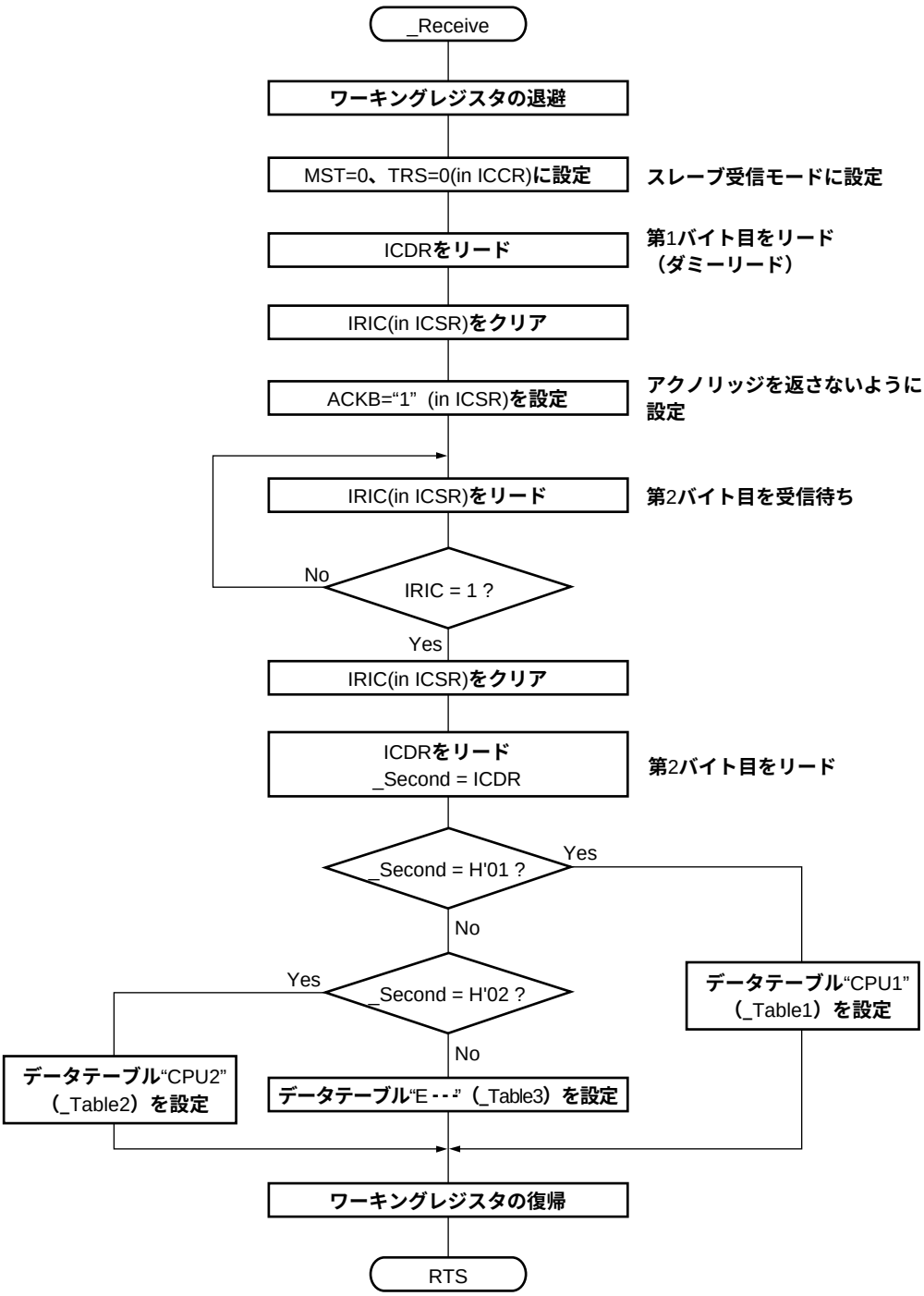
初期設定

- 割り込みベクタの設定
- 変数の領域の確保
- スタックポインタの設定
- I^2C バス I/F の初期設定
- 8 ビットタイマの初期設定
- I/O ポートの初期設定
- 割り込みの許可
- 8 セグメント LED のデータテーブルを “ 000 ” (_Table) に設定

(2) 8 セグメント LED 点灯プログラム



(3) データ受信プログラム



3.5 プログラムリスト

3.5.1 マスタプログラム

```
            .cpu            300
            .output         dbg
;*************************************************************************
; IIC バス評価システム
;           マスタプログラム
;                           (1) キースキャン
;                           (2) データの送信
;*************************************************************************

;*********************************************************
;           ベクタアドレス
;*********************************************************
            .section        VECT,CODE,LOCATE=H'0000

Res         .DATA.W         Main

            .ORG            H'0006

NMI         .DATA.W         Main
IRQ0        .DATA.W         Main
IRQ1        .DATA.W         Main
IRQ2        .DATA.W         Main
IRQ3        .DATA.W         Main
IRQ4        .DATA.W         Main
IRQ5        .DATA.W         Main
IRQ6        .DATA.W         Main
IRQ7        .DATA.W         Main
ICIA        .DATA.W         Main
ICIB        .DATA.W         Main
ICIC        .DATA.W         Main
ICID        .DATA.W         Main
OCIA        .DATA.W         Main
OCIB        .DATA.W         Main
FOVI        .DATA.W         Main
CMI0A       .DATA.W         Compare
```

（続く）

```
CMI0B       .DATA.W         Main
OVI0        .DATA.W         Main
CMI1A       .DATA.W         Main
CMI1B       .DATA.W         Main
OVI1        .DATA.W         Main
MREI        .DATA.W         Main
MWEI        .DATA.W         Main
ERI         .DATA.W         Main
RXI         .DATA.W         Main
TXI         .DATA.W         Main
RDI         .DATA.W         Main
;
;*********************************************************************
;           各種インターフェースの定義
;*********************************************************************
;-------------------------------------------
; IIC バスレジスタの定義
;-------------------------------------------
_STCR       .EQU            H'FFC3          ;シリアルタイマコントロールレジスタ
_ICCR       .EQU            H'FFD8          ;IIC バスコントロールレジスタ
_ICSR       .EQU            H'FFD9          ;IIC バスステータスレジスタ
_ICDR       .EQU            H'FFDE          ;IIC バスデータレジスタ
_ICMR       .EQU            H'FFDF          ;IIC バスモードレジスタ
_SAR        .EQU            H'FFDF          ;スレーブアドレスレジスタ
;-------------------------------------------
; I/O レジスタの定義
;-------------------------------------------
_KMPCR      .EQU            H'FFF2          ;ポート 6 入力プルアップMOS コントロールレジスタ
_P6DDR      .EQU            H'FFB9          ;データディレクションレジスタ
_P6DR       .EQU            H'FFBB          ;データレジスタ（スイッチを接続します。）

;-------------------------------------------
; 8 ビットタイマレジスタの定義
;-------------------------------------------

_TCR        .EQU            H'FFC8
_TCSR       .EQU            H'FFC9
_TCORA      .EQU            H'FFCA
_TCORB      .EQU            H'FFCB          ;未使用
```

（続く）

```
_TCNT       .EQU            H'FFCC

;-----------------------------------------
;           RAM 変数の定義
;-----------------------------------------
            .section        RAM,DATA,LOCATE=H'FB80

_Switch     .RES            1                   ;スイッチの状態を表す変数
;*****************************************************************
;           メインプログラムのスタート
;*****************************************************************
            .section        program,data,locate=H'1000

_Main       MOV.W           #H'FEFE,SP          ;スタックポインタの設定
;-----------------------------------------
; IIC バスレジスタの初期設定
;-----------------------------------------

            MOV.B           #H'10,R1L
            MOV.B           R1L,@_STCR          ;IICE = 1
            MOV.B           #H'B4,R1L
            MOV.B           R1L,@_ICCR          ;ICE = 1,MST = 1,TRS = 1
                                                ;転送クロックを 200bps に設定
;-----------------------------------------
; I/O レジスタの初期設定
;-----------------------------------------
            MOV.B           #H'00,R1L
            MOV.B           R1L,@_P6DDR

            MOV.B           #H'00,R1L
            MOV.B           R1L,@_KMPCR

;-----------------------------------------
; 8 ビットタイマの初期化
;-----------------------------------------

            MOV.B           #H'4B,R1L
            MOV.B           R1L,@_TCR
            MOV.B           #H'7D,R1L
            MOV.B           R1L,@_TCORAe
```

（続く）

```
;-------------------------------------------
; スイッチカウンタの初期化
;-------------------------------------------

            MOV.B           #H'00,R1L
            MOV.B           R1L,@_Switch        ;スイッチカウンタを０に初期化
            MOV.B           #H'00,R1L           ;8 ビット内部カウンタを０にリセット

            MOV.B           R1L,@_TCNT          ;カウント動作スタート

            ANDC            #H'7F,CCR           ;割り込みフラグのクリア

;-------------------------------------------
; スイッチの On/Off の判定
;-------------------------------------------
            MOV.B           #H'03,R2L
SwOn        MOV.B           @_Switch,R1L
            CMP.B           R2L,R1L
            BLT             SwOn
            MOV.B           #H'00,R1L
            MOV.B           R1L,@_Switch        ;スイッチカウンタをクリア
;-------------------------------------------
; データの送信
;-------------------------------------------
            MOV.B           #H'0B,R1L
            MOV.B           R1L,@_TCR           ;CMPA の割り込みを禁止する

            JSR             @Master             ;データ送信プログラムにジャンプ

            MOV.B           #H'4B,R1L
            MOV.B           R1L,@_TCR           ;CMFA の割り込みを許可する

            BRA             SwOn

;-------------------------------------------
; キースキャンルーチン（割り込みルーチン）
;-------------------------------------------
Compare     .EQU            $
            PUSH            R1
            BCLR            #6,@_TCSR           ;CMFA ビットをクリア
```

（続く）

```
            BTST        #0,@_P6DR       ;Switch フラグをチェック
            BNE         Off             ;スイッチカウンタをクリア

            MOV.B       @_Switch,R1L    ;スイッチが Off の場合

            INC         R1L             ;スイッチカウンタをインクリメント
            MOV.B       R1L,@_Switch
            BRA         Clear

Off         MOV.B       #H'00,R1L       ;スイッチが Off の場合
            MOV.B       R1L,@_Switch    ;Switch カウンタをクリア

Clear       POP         R1

            RTE                         ;キースキャンルーチンの終わり

            .include    "master.asm"    ;ファイルの結合

;-----------------------------------------
;           ROM に初期値をセットする
;-----------------------------------------
_Table      .DATA.B     H'EE            ;スレーブアドレス(H'77)と
                                        ;R/W_ビット(=0) -> B'11101110
            .DATA.B     H'01            ;マスタ識別データ（マスタ 2 は H'02）

            .END

;*****************************************************************************
; マスタのデータ送信プログラム
;           第 1 バイト = スレーブアドレス
;           第 2 バイト = マスタの識別データ
;*****************************************************************************
Master      BTST        #7,@_ICSR       ;IIC バスはフリー状態か?
            BNE         Master

            BSET        #5,@_ICCR       ;マスタ送信モードにセット
            BSET        #4,@_ICCR       ;(MST = 0,TRS = 0)

            MOV.B       #H'90,R1L       ;送信開始条件を発行
            MOV.B       R1L,@_ICSR      ;ICSR : 1001 0000
```

（続く）

```
            MOV.B       #H'00,R5L
```

```
Transmit    MOV.B       @(_Table,R5),R3L  ;第１バイト（スレーブアドレス）と
            MOV.B       R3L,@_ICDR        ;第２バイト（マスタ識別データ）の
                                          ;書き込み
            INC         R5L

ChkIRIC1    BTST        #6,@_ICSR
            BEQ         ChkIRIC1          ;IRIC = 1?（送信完了?）
            BCLR        #6,@_ICSR         ;次の送信に控え,IRIC をクリア

            BTST        #3,@_ICSR         ;AL = 0?
            BNE         Master

            BTST        #0,@_ICSR         ;ACKB = 0?
            BEQ         Transmit

            MOV.B       #H'10,R1L         ;送信停止条件の発行
            MOV.B       R1L,@_ICS         ;ICSR : 0001 0000

            RTS                           ;リターンサブルーチン
```

3.5.2 スレーブプログラム

```
            .cpu            300
            .output         dbg
;*************************************************************************
; IIC バス評価システム
;           スレーブプログラム
;                           (1) LED 表示
;                           (2) データの受信
;*************************************************************************

;*********************************************************************
;           オンチップレジスタの定義
;*********************************************************************
_STCR       .EQU            H'FFC3          ;シリアルタイマコントロールレジスタ
_ICCR       .EQU            H'FFD8          ;IIC バスコントロールレジスタ
_ICSR       .EQU            H'FFD9          ;IIC バスステータスレジスタ
_ICDR       .EQU            H'FFDE          ;IIC バスデータレジスタ
_ICMR       .EQU            H'FFDF          ;IIC モードレジスタ
_SAR        .EQU            H'FFDF          ;スレーブアドレスレジスタ
_TCR        .EQU            H'FFC8          ;タイマコントロールレジスタ
_TCSR       .EQU            H'FFC9          ;タイマコントロール/ステータスレジスタ
_TCORA      .EQU            H'FFCA          ;タイムコンスタントレジスタ
_TCORB      .EQU            H'FFCB          ;タイムコンスタントレジスタ
_P1DDR      .EQU            H'FFB0          ;ポート 1 データディレクションレジスタ
_P2DDR      .EQU            H'FFB1          ;ポート 2 データディレクションレジスタ
_P1DR       .EQU            H'FFB2          ;ポート 1 データレジスタ
_P2DR       .EQU            H'FFB3          ;ポート 2 データレジスタ

            .section        VECT,CODE,LOCATE=H'0000
;*********************************************************************
; ベクタアドレス
;*********************************************************************
Res         .DATA.W         _Main
            .ORG            H'0006
NMI         .DATA.W         _Main
IRQ0        .DATA.W         _Main
IRQ1        .DATA.W         _Main
```

（続く）

```
IRQ2        .DATA.W        _Main
IRQ3        .DATA.W        _Main
IRQ4        .DATA.W        _Main
IRQ5        .DATA.W        _Main
IRQ6        .DATA.W        _Main
IRQ7        .DATA.W        _Main
ICIA        .DATA.W        _Main
ICIB        .DATA.W        _Main
ICIC        .DATA.W        _Main
ICID        .DATA.W        _Main
OCIA        .DATA.W        _Main
OCIB        .DATA.W        _Main
FOVI        .DATA.W        _Main
CMI0A       .DATA.W        _Main
CMI0B       .DATA.W        _Main
OVI0        .DATA.W        _Main
CMI1A       .DATA.W        _Main
CMI1B       .DATA.W        _Main
OVI1        .DATA.W        _Main
IBF1        .DATA.W        _Main
IBF2        .DATA.W        _Main
ERI0        .DATA.W        _Main
RXI0        .DATA.W        _Main
TXI0        .DATA.W        _Main
TEI0        .DATA.W        _Main
ERI1        .DATA.W        _Main
RXI1        .DATA.W        _Main
TXI1        .DATA.W        _Main
TEI1        .DATA.W        _Main
ADI         .DATA.W        _Main
WOVF        .DATA.W        _Main
IICI        .DATA.W        _Receive

            .SECTION       RAM,DATA,LOCATE=H'FB80
;--------------------------------------------
; RAM エリアの初期化
;--------------------------------------------
_TABLE      .RES.W         1                ;H'FB80<- 受信データの格納場所
```

（続く）

```
_Count      .RES.B          1               ;H'FB82<- 点灯時間
_Count2     .RES.B          1               ;H'FB83<- 点灯時間
_D_DATA     .RES.B          1               ;H'FB84<- ディジットデータの確保
_First      .RES.B          1               ;H'FB85<- 送信データ 1
_Second     .RES.B          1               ;H'FB86<- 送信データ 2

            .SECTION        PROGRAM,CODE,LOCATE=H'1000
;*****************************************************************
;           メインプログラムのスタート
;*****************************************************************
_Main       MOV.W           #H'FEFE,SP      ;スタックポインタの設定

;-----------------------------------------
; LED 表示用プログラムの設定
;-----------------------------------------
            MOV.B           #H'0A,R1L       ;カウンタのクリア条件を設定
            MOV.B           R1L,@_TCR

            MOV.B           #H'F0,R1L       ;コンペアマッチ B
            MOV.B           R1L,@_TCORB

            MOV.B           #H'FF,R1L       ;コンペアマッチ A
            MOV.B           R1L,@_TCORA

            MOV.B           #H'FF,R1L
            MOV.B           R1L,@_P1DDR     ;すべての端子は出力
            MOV.B           R1L,@_P2DDR     ;すべての端子は出力

;-----------------------------------------
; IIC バスインタフェースレジスタの初期設定
;-----------------------------------------
            MOV.B           #H'11,R1L       ;IICE = 1
            MOV.B           R1L,@_STCR      ;0001 0001

            MOV.B           #H'EE,R1L
            MOV.B           R1L,@_SAR       ;スレーブアドレスの設定
            MOV.B           #H'C4,R1L       ;ICE = 1,IEIC = 1,転送クロック:400MHz
            MOV.B           R1L,@_ICCR      ;B'1100 0100
```

（続く）

```
;-----------------------------------------
; 割り込みマスクの解除
;-----------------------------------------
             ANDC        #H'7F,CCR
;-----------------------------------------
;データテーブルの交換
;-----------------------------------------
            MOV.W          #_Table4,R0
            MOV.W          R0,@_TABLE

LOOP        JSR            @_Display         ;LED 表示ルーチンへジャンプ
            BRA            LOOP

;*****************************************************************
;           LED 表示用サブルーチン
;*****************************************************************
_Display    MOV.W          @_TABLE,R6        ;データテーブルの交換
            MOV.B          R1L,@_Count2      ;Count2 = 0
MORE2       MOV.B          #H'00,R1L
            MOV.B          R1L,@_Count       ;Count = 0

MORE1       MOV.W          @_TABLE,R6        ;データテーブルの先頭アドレスをセット
            MOV.B          #H'08,R1L
            MOV.B          R1L,@_D_DATA      ;ディジットデータ H'01 をセット

NEXT1       MOV.B          @_D_DATA,R1L
            NOT            R1L
            MOV.B          R1L,@_P1DR        ;ディジットデータを出力
            MOV.B          @R6,R1L
            MOV.B          R1L,@_P2DR        ;セグメントデータを出力

CMFB1       BTST           #7,@_TCSR         ;CMFB = 1?
            BEQ            CMFB1
            BCLR           #7,@_TCSR
            MOV.B          #H'FF,R1L
            MOV.B          R1L,@_P1DR        ;ディジットデータ H'FF を出力
CMFA1       BTST           #6,@_TCSR         ;CMFA = 1?
            BEQ            CMFA1
```

（続く）

```
                BCLR            #6,@_TCSR

                MOV.B           @_D_DATA,R1L        ;ディジットデータをシフト
                SHLR            R1L
                MOV.B           R1L,@_D_DATA

                ADDS            #1,R6               ;次の LED データの準備
                CMP.B           #H'00,R1L
                BNE             NEXT1

                MOV.B           @_Count,R1L
                INC             R1L
                MOV.B           R1L,@_Count

                MOV.B           @_Count,R1L
                CMP.B           #H'FF,R1L
                BNE             MORE1

                MOV.B           @_Count2,R1L
                INC             R1L
                MOV.B           R1L,@_Count2
                MOV.B           @_Count2,R1L
                CMP.B           #H'02,R1L
                BNE             MORE2
                RTS

;*************************************************************************
; IIC バスインターフェース割り込みプログラム
;           データの受信と判定,データテーブルの入れ替え
;*************************************************************************
_Receive    PUSH            R1
            PUSH            R4                  ;レジスタの退避

            BCLR            #6,@_ICSR           ;IRIC をクリア

            MOV.B           @_ICDR,R1L          ;データの読み込み（ダミーリード）
            MOV.B           R1L,@_First         ;メモリに格納
            BSET            #0,@_ICSR           ;ACKB = 1
LOOP1       BTST            #6,@_ICSR           ;第二バイト目（マスタ識別データ）
            BEQ             LOOP1               ;の受信終了?
```

（続く）

```
            BCLR        #6,@_ICSR       ;IRIC をクリア

            MOV.B       @_ICDR,R1L
            MOV.B       R1L,@_Second    ;メモリに受信データを格納

            BCLR        #0,@_ICSR       ;ACKB = 0
                                        ;次回以降用の設定

            MOV.B       #H'00,R1L
            MOV.B       R1L,@_ICMR      ;1 フレームあたり 9 ビットに設定

;-------------------------------------------
; 判定
;-------------------------------------------
            MOV.B       @_Second,R1L    ;マスタ識別データをリード

_Judge      CMP.B       #H'01,R1L       ;受信データの判定
            BEQ         EXIT1
            CMP.B       #H'02,R1L
            BEQ         EXIT2
EXIT3       MOV.W       #_Table3,R4
            MOV.W       R4,@_TABLE
            BRA         Clear

EXIT1       MOV.W       #_Table1,R4
            MOV.W       R4,@_TABLE
            BRA         Clear

EXIT2       MOV.W       #_Table2,R4
            MOV.W       R4,@_TABLE
            BRA         Clear

Clear       POP         R1
            POP         R2
            POP         R4              ;レジスタを元に戻す

            RTE
```

（続く）

```
;-----------------------------------------
; ８セグメントＬＥＤのデータテーブル
;-----------------------------------------
_Table1     .DATA.B        H'9C             ;H'004B  LED DATA of "C"
            .DATA.B        H'CE             ;H'004C  LED DATA of "P"
            .DATA.B        H'7C             ;H'004D  LED DATA of "U"
            .DATA.B        H'60             ;H'004E  LED DATA of "1"
_Table2     .DATA.B        H'9C             ;H'004F  LED DATA of "C"
            .DATA.B        H'CE             ;H'0050  LED DATA of "P"
            .DATA.B        H'7C             ;H'0051  LED DATA of "U"
            .DATA.B        H'DA             ;H'0052  LED DATA of "2"
_Table3     .DATA.B        H'9F             ;H'0053  LED DATA of "E"
            .DATA.B        H'02             ;H'0054  LED DATA of "-"
            .DATA.B        H'02             ;H'0055  LED DATA of "-"
            .DATA.B        H'02             ;H'0056  LED DATA of "-"
_Table4     .DATA.B        H'FC             ;H'0053  LED DATA of "0"
            .DATA.B        H'FC             ;H'0054  LED DATA of "0"
            .DATA.B        H'FC             ;H'0055  LED DATA of "0"
            .DATA.B        H'FC             ;H'0056  LED DATA of "0"

            .END
```

4. H8S シリーズ応用例

4.1 H8S シリーズ応用例使用手引き

4.1.1 H8S シリーズ応用例の構成

H8S シリーズ応用例は**図** 4.1 に示す構成で H8S シリーズの I²C バスインタフェースの使用例について説明しています。

デバイスは、H8S/2138 を使用するものとします。

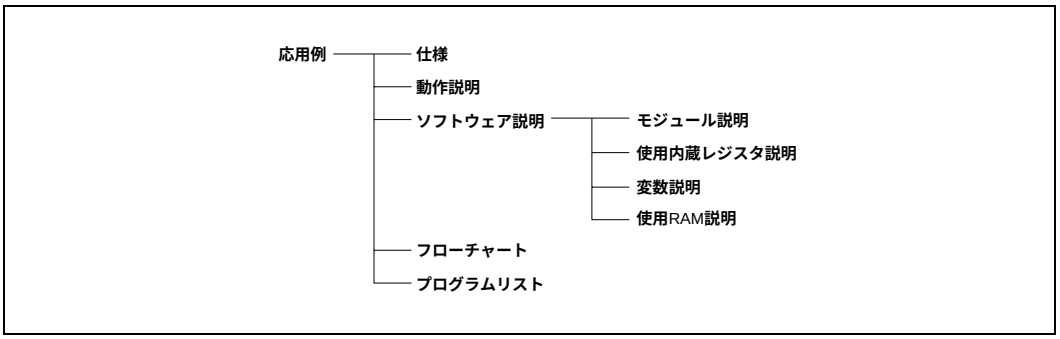


図 4.1 H8S **シリーズ応用例の構成**

(1) 仕様

タスク例のシステム仕様について説明しています。

(2) 動作説明

タスク例の動作をタイミングチャートを使用して説明しています。

(3) ソフトウェア説明

(a) モジュール説明

- タスク例を動作させるソフトウェアのモジュールについて説明しています。

(b) 使用内蔵レジスタ説明

- モジュールで設定する I²C バスインタフェース、およびその他の内蔵レジスタについて説明しています。

(c) 変数説明

- タスク例を動作させるソフトウェアの変数について説明しています。

(d) 使用 RAM 説明

- モジュールで使用する RAM のラベル名および機能について説明しています。

(4) フローチャート

タスク例を実行するソフトウェアについてフローチャートを使用して説明しています。

(5) プログラムリスト

タスク例を実行するソフトウェアのプログラムリストを示しています。

4.1.2 ベクタテーブル定義ファイル説明

以下に C 言語を使用したベクタテーブルの定義ファイルを示します。**図** 4.2 に示すように割り込み処理ルーチンの先頭アドレスを確保したファイルを作成します。割り込み処理を使用する場合には、割り込み処理ルーチンの先頭ラベルを割り込みに対応したベクタ位置に記述します。**図** 4.2 に IIC のチャネル 0 の割り込みを使用する例を示します。先頭アドレス（IIC0INT）を外部参照とします（**図** 4.2-A 参照）。IICI0 の位置のラベル名を IIC0INT にします（**図** 4.2-B 参照）。

```
/******************************************************
*    H8S/2138 Series vector table
*       for mode3(normal,single-chip mode)
******************************************************/

extern void main(void);
extern void IIC0INT(void);
const void (*vect_tbl[])(void) =
{
   main,                                    /* H'0000  Reset          */
   main,                                    /* H'0002  Reserve        */
   main,                                    /* H'0004  Reserve        */
   main,                                    /* H'0006  Reserve        */
   main,                                    /* H'0008  Reserve        */
   main,                                    /* H'000A  Reserve        */
   main,                                    /* H'000C  Direct transfer*/
   main,                                    /* H'000E  NMI            */
   main,                                    /* H'0010  Trap           */
   main,                                    /* H'0012  Trap           */
   main,                                    /* H'0014  Trap           */
   main,                                    /* H'0016  Trap           */
   main,                                    /* H'0018  Reserve        */
   main,                                    /* H'001A  Reserve        */
   main,                                    /* H'001C  Reserve        */
   main,                                    /* H'001E  Reserve        */
   main,                                    /* H'0020  IRQ0           */
   main,                                    /* H'0022  IRQ1           */
   main,                                    /* H'0024  IRQ2           */
   main,                                    /* H'0026  IRQ3           */
   main,                                    /* H'0028  IRQ4           */
```

A

ラベル名“ IIC0INT **”を外部参照します。**

図 4.2 **ベクタ定義ファイル** **（続く）**

```
    main,                                   /* H'002A  IRQ5           */
    main,                                   /* H'002C  IRQ6,KIN7-KIN0 */
    main,                                   /* H'002E  IRQ7           */
    main,                                   /* H'0030  SWDTEND        */
    main,                                   /* H'0032  WOVI0          */
    main,                                   /* H'0034  WOVI1          */
    main,                                   /* H'0036  PC break       */
    main,                                   /* H'0038  ADI            */
    main,                                   /* H'003A  Reserve        */
    main,                                   /* H'003C  Reserve        */
    main,                                   /* H'003E  Reserve        */
    main,                                   /* H'0040  Reserve        */
    main,                                   /* H'0042  Reserve        */
    main,                                   /* H'0044  Reserve        */
    main,                                   /* H'0046  Reserve        */
    main,                                   /* H'0048  Reserve        */
    main,                                   /* H'004A  Reserve        */
    main,                                   /* H'004C  Reserve        */
    main,                                   /* H'004E  Reserve        */
    main,                                   /* H'0050  Reserve        */
    main,                                   /* H'0052  Reserve        */
    main,                                   /* H'0054  Reserve        */
    main,                                   /* H'0056  Reserve        */
    main,                                   /* H'0058  Reserve        */
    main,                                   /* H'005A  Reserve        */
    main,                                   /* H'005C  Reserve        */
    main,                                   /* H'005E  Reserve        */
    main,                                   /* H'0060  ICIA           */
    main,                                   /* H'0062  ICIB           */
    main,                                   /* H'0064  ICIC           */
    main,                                   /* H'0066  ICID           */
    main,                                   /* H'0068  OCIA           */
    main,                                   /* H'006A  OCIB           */
    main,                                   /* H'006C  FOVI           */
    main,                                   /* H'006E  Reserve        */
    main,                                   /* H'0070  Reserve        */
```

図 4.2 **ベクタ定義ファイル（続き）**

```
    main,                                   /* H'0072  Reserve      */
    main,                                   /* H'0074  Reserve      */
    main,                                   /* H'0076  Reserve      */
    main,                                   /* H'0078  Reserve      */
    main,                                   /* H'007A  Reserve      */
    main,                                   /* H'007C  Reserve      */
    main,                                   /* H'007E  Reserve      */
    main,                                   /* H'0080  CMIA0        */
    main,                                   /* H'0082  CMIB0        */
    main,                                   /* H'0084  OVI0         */
    main,                                   /* H'0086  Reserve      */
    main,                                   /* H'0088  CMIA1        */
    main,                                   /* H'008A  CMIB1        */
    main,                                   /* H'008C  OVI1         */
    main,                                   /* H'008E  Reserve      */
    main,                                   /* H'0090  CMIAY        */
    main,                                   /* H'0092  CMIBY        */
    main,                                   /* H'0094  OVIY         */
    main,                                   /* H'0096  ICIX         */
    main,                                   /* H'0098  IBF1         */
    main,                                   /* H'009A  IBF2         */
    main,                                   /* H'009C  Reserve      */
    main,                                   /* H'009E  Reserve      */
    main,                                   /* H'00A0  ERI0         */
    main,                                   /* H'00A2  RXI0         */
    main,                                   /* H'00A4  TXI0         */
    main,                                   /* H'00A6  TEI0         */
    main,                                   /* H'00A8  ERI1         */
    main,                                   /* H'00AA  RXI1         */
    main,                                   /* H'00AC  TXI1         */
    main,                                   /* H'00AE  TEI1         */
    main,                                   /* H'00B0  ERI2         */
    main,                                   /* H'00B2  RXI2         */
    main,                                   /* H'00B4  TXI2         */
IIC0INT,                                    /* H'00B6  TEI2         */
    main,                                   /* H'00B8  IICI0        */
```

B

ラベル名 “ IIC0INT ” を記述します。

図 4.2　**ベクタ定義ファイル（続き）**

```
    main,                                   /* H'00BA  DDCSWI          */
    main,                                   /* H'00BC  IICI1           */
    main,                                   /* H'00BE  Reserve         */
    main,                                   /* H'00C0  Reserve         */
    main,                                   /* H'00C2  Reserve         */
    main,                                   /* H'00C4  Reserve         */
    main,                                   /* H'00C6  Reserve         */
    main,                                   /* H'00C8  Reserve         */
    main,                                   /* H'00CA  Reserve         */
    main,                                   /* H'00CC  Reserve         */
    main,                                   /* H'00CE  Reserve         */
};
```

図 4.2　**ベクタ定義ファイル（続き）**

4.1.3 レジスタ定義ファイル説明

以下に、H8S/2138 シリーズのレジスタ定義ファイルを示します。

H8S/2138 シリーズ　レジスタ定義ファイル　＜2138s.h＞

```
/************************************************************************/
/*     H8S/2138 Series Include File                                     */
/************************************************************************/
union un_kbcomp {                                   /* union KBCOMP */
                unsigned char BYTE;                 /*              */
                struct {                            /*  Bit  Access */
                       unsigned char IrE  :1;       /*   IrE        */
                       unsigned char IrCKS:3;       /*   IrCKS      */
                       unsigned char KBADE:1;       /*   KBADE      */
                       unsigned char KBCH :3;       /*   KBCH       */
                       }      BIT;                  /*              */
};                                                  /*              */
struct st_iic {                                     /* struct IIC   */
              union {                               /* ICCR         */
                    unsigned char BYTE;             /*  Byte Access */
                    struct {                        /*  Bit  Access */
                           unsigned char ICE :1;    /*   ICE        */
                           unsigned char IEIC:1;    /*   IEIC       */
                           unsigned char MST :1;    /*   MST        */
                           unsigned char TRS :1;    /*   TRS        */
                           unsigned char ACKE:1;    /*   ACKE       */
                           unsigned char BBSY:1;    /*   BBSY       */
                           unsigned char IRIC:1;    /*   IRIC       */
                           unsigned char SCP :1;    /*   SCP        */
                           }      BIT;              /*              */
                    }            ICCR;              /*              */
              union {                               /* ICSR         */
                    unsigned char BYTE;             /*  Byte Access */
                    struct {                        /*  Bit  Access */
                           unsigned char ESTP:1;    /*   ESTP       */
                           unsigned char STOP:1;    /*   STOP       */
                           unsigned char IRTR:1;    /*   IRTR       */
```

（続く）

```
                unsigned char AASX :1;        /*   AASX       */
                unsigned char AL   :1;        /*   AL         */
                unsigned char AAS  :1;        /*   AAS        */
                unsigned char ADZ  :1;        /*   ADZ        */
                unsigned char ACKB :1;        /*   ACKB       */
                }      BIT;                   /*              */
        }           ICSR;                     /*              */
char                wk[4];                    /*              */
union {                                       /*              */
        struct {                              /*              */
                union {                       /* SARX         */
                        unsigned char BYTE;   /*  Byte Access */
                        struct {              /*  Bit  Access */
                                unsigned char SVAX:7; /* SVAX */
                                unsigned char FSX :1; /* FSX  */
                                }      BIT;   /*              */
                        } UN_SARX;            /*              */
                union {                       /* SAR          */
                        unsigned char BYTE;   /*  Byte Access */
                        struct {              /*  Bit  Access */
                                unsigned char SVA:7; /* SVA   */
                                unsigned char FS :1; /* FS    */
                                }      BIT;   /*              */
                     } UN_SAR;                /*              */
                } ICE0;                       /*              */
        struct {                              /*              */
                unsigned char UN_ICDR;        /* ICDR         */
                union {                       /* ICMR         */
                        unsigned char BYTE;   /*  Byte Access */
                        struct {              /*  Bit  Access */
                                unsigned char MLS :1; /* MLS  */
                                unsigned char WAIT:1; /* WAIT */
                                unsigned char CKS :3; /* CKS  */
                                unsigned char BC  :3; /* BC   */
                                }      BIT;   /*              */
                        } UN_ICMR;            /*              */
                } ICE1;                       /*              */
```

（続く）

```
                    }           EQU;                    /*             */
};                                                      /*             */
union un_ddcswr {                                       /* union DDCSWR */
                unsigned char BYTE;                     /*             */
                struct {                                /*  Bit  Access */
                       unsigned char SWE:1;             /*   SWE       */
                       unsigned char SW :1;             /*   SW        */
                       unsigned char IE :1;             /*   IE        */
                       unsigned char IF :1;             /*   IF        */
                       }      BIT;                      /*             */
};                                                      /*             */
struct st_intc {                                        /* struct INTC  */
               union {                                  /* ICRA         */
                     unsigned char BYTE;                /*  Byte Access */
                     struct {                           /*  Bit  Access */
                            unsigned char B7:1;         /*   IRQ0      */
                            unsigned char B6:1;         /*   IRQ1      */
                            unsigned char B5:1;         /*   IRQ2,IRQ3 */
                            unsigned char B4:1;         /*   IRQ4,IRQ5 */
                            unsigned char B3:1;         /*   IRQ6,IRQ7 */
                            unsigned char B2:1;         /*   DTC       */
                            unsigned char B1:1;         /*   WDT0      */
                            unsigned char B0:1;         /*   WDT1      */
                            }      BIT;                 /*             */
                     }          ICRA;                   /*             */
               union {                                  /* ICRB         */
                     unsigned char BYTE;                /*  Byte Access */
                     struct {                           /*  Bit  Access */
                            unsigned char B7:1;         /*   A/D       */
                            unsigned char B6:1;         /*   FRT       */
                            unsigned char   :2;         /*             */
                            unsigned char B3:1;         /*   TMR0      */
                            unsigned char B2:1;         /*   TMR1      */
                            unsigned char B1:1;         /*   TMRX,Y    */
                            unsigned char B0:1;         /*   HIF       */
                            }      BIT;                 /*             */
                     }          ICRB;                   /*             */
```

（続く）

```
union {                                 /* ICRC         */
      unsigned char BYTE;               /*  Byte Access */
      struct {                          /*  Bit  Access */
             unsigned char B7:1;        /*   SCI0       */
             unsigned char B6:1;        /*   SCI1       */
             unsigned char B5:1;        /*   SCI2       */
             unsigned char B4:1;        /*   IIC0       */
             unsigned char B3:1;        /*   IIC1       */
             }      BIT;                /*              */
      }           ICRC;                 /*              */
union {                                 /* ISR          */
      unsigned char BYTE;               /*  Byte Access */
      struct {                          /*  Bit  Access */
             unsigned char IRQ7F:1;     /*   IRQ7F      */
             unsigned char IRQ6F:1;     /*   IRQ6F      */
             unsigned char IRQ5F:1;     /*   IRQ5F      */
             unsigned char IRQ4F:1;     /*   IRQ4F      */
             unsigned char IRQ3F:1;     /*   IRQ3F      */
             unsigned char IRQ2F:1;     /*   IRQ2F      */
             unsigned char IRQ1F:1;     /*   IRQ1F      */
             unsigned char IRQ0F:1;     /*   IRQ0F      */
             }      BIT;                /*              */
      }          ISR;                  /*              */
union {                                 /* ISCR         */
      unsigned int WORD;                /*  Word Access */
      struct {                          /*  Byte Access */
             unsigned char H;           /*   ISCRH      */
             unsigned char L;           /*   ISCRL      */
             }      BYTE;               /*              */
      struct {                          /*  Bit  Access */
             unsigned char IRQ7SC:2;    /*   IRQ7SC     */
             unsigned char IRQ6SC:2;    /*   IRQ6SC     */
             unsigned char IRQ5SC:2;    /*   IRQ5SC     */
             unsigned char IRQ4SC:2;    /*   IRQ4SC     */
             unsigned char IRQ3SC:2;    /*   IRQ3SC     */
             unsigned char IRQ2SC:2;    /*   IRQ2SC     */
             unsigned char IRQ1SC:2;    /*   IRQ1SC     */
```

（続く）

```
                unsigned char IRQ0SC:2;     /*   IRQ0SC     */
                }      BIT;                 /*              */
        }           ISR;                    /*              */
char                wk1[6];                 /*              */
union {                                     /* ABRKCR       */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char CMF:1;        /*   CMF        */
                unsigned char    :6;        /*              */
                unsigned char BIE:1;        /*   BIE        */
                }      BIT;                 /*              */
        }           ABRKCR;                 /*              */
unsigned char       BARA;                   /* BARA         */
unsigned char       BARB;                   /* BARB         */
unsigned char       BARC;                   /* BARC         */
char                wk2[202];               /*              */
union {                                     /* IER          */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char IRQ7E:1;      /*   IRQ7E      */
                unsigned char IRQ6E:1;      /*   IRQ6E      */
                unsigned char IRQ5E:1;      /*   IRQ5E      */
                unsigned char IRQ4E:1;      /*   IRQ4E      */
                unsigned char IRQ3E:1;      /*   IRQ3E      */
                unsigned char IRQ2E:1;      /*   IRQ2E      */
                unsigned char IRQ1E:1;      /*   IRQ1E      */
                unsigned char IRQ0E:1;      /*   IRQ0E      */
                }      BIT;                 /*              */
        }           IER;                    /*              */
char                wk3[46];                /*              */
union {                                     /* KMIMR        */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char B7:1;         /*   Bit 7      */
                unsigned char B6:1;         /*   Bit 6      */
                unsigned char B5:1;         /*   Bit 5      */
                unsigned char B4:1;         /*   Bit 4      */
```

（続く）

```
                        unsigned char B3:1;         /*   Bit 3      */
                        unsigned char B2:1;         /*   Bit 2      */
                        unsigned char B1:1;         /*   Bit 1      */
                        unsigned char B0:1;         /*   Bit 0      */
                        }      BIT;                 /*              */
                  }           KMIMR;                /*              */
            char              wk4;                  /*              */
            union {                                 /* KMIMRA       */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char B15:1;        /*   Bit 7      */
                        unsigned char B14:1;        /*   Bit 6      */
                        unsigned char B13:1;        /*   Bit 5      */
                        unsigned char B12:1;        /*   Bit 4      */
                        unsigned char B11:1;        /*   Bit 3      */
                        unsigned char B10:1;        /*   Bit 2      */
                        unsigned char B9 :1;        /*   Bit 1      */
                        unsigned char B8 :1;        /*   Bit 0      */
                        }      BIT;                 /*              */
                  }          KMIMRA;                /*              */
};                                                  /*              */
struct st_dtc {                                     /* struct DTC   */
            union {                                 /* EA           */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char B7:1;         /*   IRQ0       */
                        unsigned char B6:1;         /*   IRQ1       */
                        unsigned char B5:1;         /*   IRQ2       */
                        unsigned char B4:1;         /*   IRQ3       */
                        unsigned char B3:1;         /*   A/D        */
                        unsigned char B2:1;         /*   FRT ICIA   */
                        unsigned char B1:1;         /*   FRT ICIB   */
                        unsigned char B0:1;         /*   FRT OCIA   */
                        }      BIT;                 /*              */
                  }           EA;                   /*              */
            union {                                 /* EB           */
                  unsigned char BYTE;               /*  Byte Access */
```

（続く）

```
        struct {                            /*  Bit  Access  */
                unsigned char B7:1;         /*    FRT OCIB   */
                unsigned char   :4;         /*               */
                unsigned char B2:1;         /*    TMR0 CMIA  */
                unsigned char B1:1;         /*    TMR0 CMIB  */
                unsigned char B0:1;         /*    TMR1 CMIA  */
                }      BIT;                 /*               */
        }            EB;                    /*               */
union {                                     /* EC            */
        unsigned char BYTE;                 /*  Byte Access  */
        struct {                            /*  Bit  Access  */
                unsigned char B7:1;         /*    TMR1 CMIB  */
                unsigned char B6:1;         /*    TMRY CMIA  */
                unsigned char B5:1;         /*    TMRY CMIB  */
                unsigned char B4:1;         /*    HIF1       */
                unsigned char B3:1;         /*    HIF2       */
                unsigned char B2:1;         /*    SCIO RXI   */
                unsigned char B1:1;         /*    SCIO TXI   */
                unsigned char B0:1;         /*    SCI1 RXI   */
                }      BIT;                 /*               */
        }            EC;                    /*               */
union {                                     /* ED            */
        unsigned char BYTE;                 /*  Byte Access  */
        struct {                            /*  Bit  Access  */
                unsigned char B7:1;         /*    SCI1 TXI   */
                unsigned char B6:1;         /*    SCI2 RXI   */
                unsigned char B5:1;         /*    SCI2 TXI   */
                unsigned char B4:1;         /*    IIC0       */
                unsigned char B3:1;         /*    IIC1       */
                }      BIT;                 /*               */
        }            ED;                    /*               */
char                  wk;                   /*               */
union {                                     /* VECR          */
        unsigned char BYTE;                 /*  Byte Access  */
        struct {                            /*  Bit  Access  */
                unsigned char SWDTE:1;      /*    SWDTE      */
                unsigned char DTVEC:7;      /*    DTVEC      */
```

（続く）

```
                        }      BIT;                /*              */
                  }            VECR;               /*              */
};                                                  /*              */
struct st_flash {                                   /* struct FLASH */
            union {                                 /* FLMCR1       */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char FWE:1;        /*   FWE        */
                        unsigned char SWE:1;        /*   SWE        */
                        unsigned char    :2;        /*              */
                        unsigned char EV :1;        /*   EV         */
                        unsigned char PV :1;        /*   PV         */
                        unsigned char E  :1;        /*   E          */
                        unsigned char P  :1;        /*   P          */
                        }      BIT;                 /*              */
                  }          FLMCR1;                /*              */
            union {                                 /* FLMCR2       */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char FLER:1;       /*   FLER       */
                        unsigned char     :5;       /*              */
                        unsigned char ESU :1;       /*   ESU        */
                        unsigned char PSU :1;       /*   PSU        */
                        }      BIT;                 /*              */
                  }          FLMCR2;                /*              */
            union {                                 /* EBR1         */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char wk :6;        /*              */
                        unsigned char EB9:1;        /*   EB9        */
                        unsigned char EB8:1;        /*   EB8        */
                        }      BIT;                 /*              */
                  }          EBR1;                  /*              */
            union {                                 /* EBR2         */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char EB7:1;        /*   EB7        */
```

（続く）

```
                        unsigned char EB6:1;        /*   EB6        */
                        unsigned char EB5:1;        /*   EB5        */
                        unsigned char EB4:1;        /*   EB4        */
                        unsigned char EB3:1;        /*   EB3        */
                        unsigned char EB2:1;        /*   EB2        */
                        unsigned char EB1:1;        /*   EB1        */
                        unsigned char EB0:1;        /*   EB0        */
                        }      BIT;                 /*              */
                  }            EBR2;                /*              */
};                                                  /*              */
struct st_pwm {                                     /* struct PWM   */
              union {                               /* PCSR         */
                    unsigned char BYTE;             /*  Byte Access */
                    struct {                        /*  Bit  Access */
                        unsigned char wk    :5;     /*              */
                        unsigned char PWCKB:1;      /*   PWCKB      */
                        unsigned char PWCKA:1;      /*   PWCKA      */
                        }      BIT;                 /*              */
                    }            PCSR;              /*              */
              char               wk[79];            /*              */
              union {                               /* PWOER        */
                    unsigned int WORD;              /*  Word Access */
                    struct {                        /*  Byte Access */
                        unsigned char B;            /*   PWOERB     */
                        unsigned char A;            /*   PWOERA     */
                        }      BYTE;                /*              */
                    struct {                        /*  Bit  Access */
                        unsigned char OE15:1;       /*   OE15       */
                        unsigned char OE14:1;       /*   OE14       */
                        unsigned char OE13:1;       /*   OE13       */
                        unsigned char OE12:1;       /*   OE12       */
                        unsigned char OE11:1;       /*   OE11       */
                        unsigned char OE10:1;       /*   OE10       */
                        unsigned char OE9 :1;       /*   OE9        */
                        unsigned char OE8 :1;       /*   OE8        */
                        unsigned char OE7 :1;       /*   OE7        */
                        unsigned char OE6 :1;       /*   OE6        */
```

（続く）

```
                unsigned char OE5 :1;       /*   OE5        */
                unsigned char OE4 :1;       /*   OE4        */
                unsigned char OE3 :1;       /*   OE3        */
                unsigned char OE2 :1;       /*   OE2        */
                unsigned char OE1 :1;       /*   OE1        */
                unsigned char OE0 :1;       /*   OE0        */
                }      BIT;                 /*              */
        }            OER;                   /*              */
union {                                     /* PWDPR        */
        unsigned int WORD;                  /*  Word Access */
        struct {                            /*  Byte Access */
                unsigned char B;            /*   PWDPRB     */
                unsigned char A;            /*   PWDPRA     */
                }      BYTE;                /*              */
        struct {                            /*  Bit  Access */
                unsigned char OS15:1;       /*   OS15       */
                unsigned char OS14:1;       /*   OS14       */
                unsigned char OS13:1;       /*   OS13       */
                unsigned char OS12:1;       /*   OS12       */
                unsigned char OS11:1;       /*   OS11       */
                unsigned char OS10:1;       /*   OS10       */
                unsigned char OS9 :1;       /*   OS9        */
                unsigned char OS8 :1;       /*   OS8        */
                unsigned char OS7 :1;       /*   OS7        */
                unsigned char OS6 :1;       /*   OS6        */
                unsigned char OS5 :1;       /*   OS5        */
                unsigned char OS4 :1;       /*   OS4        */
                unsigned char OS3 :1;       /*   OS3        */
                unsigned char OS2 :1;       /*   OS2        */
                unsigned char OS1 :1;       /*   OS1        */
                unsigned char OS0 :1;       /*   OS0        */
                }      BIT;                 /*              */
        }            DPR;                   /*              */
union {                                     /* PWSL         */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char PWCKE:1;      /*   PWCKE      */
```

（続く）

```
                        unsigned char PWCKS:1;      /*   PWCKE      */
                        unsigned char      :2;      /*              */
                        unsigned char RS   :4;      /*   RS         */
                        }      BIT;                 /*              */
                  }            SL;                  /*              */
            unsigned char     DR;                   /*  PWDR0-PWDR15*/
};                                                  /*              */
struct st_hif {                                     /* struct HIF   */
            union {                                 /* SYSCR2       */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char wk    :7;     /*              */
                        unsigned char HI12E:1;      /*   HI12E      */
                        }      BIT;                 /*              */
                  }            SYSCR2;              /*              */
            char               wk[108];             /*              */
            union {                                 /* HICR         */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char wk    :5;     /*              */
                        unsigned char IBFIE2:1;     /*   IBFIE2     */
                        unsigned char IBFIE1:1;     /*   IBFIE1     */
                        unsigned char FGA2OE:1;     /*   FGA2OE     */
                        }      BIT;                 /*              */
                  }            HICR;                /*              */
};                                                  /*              */
struct st_hif1 {                                    /* struct HIF1  */
            unsigned char   IDR;                    /* IDR          */
            unsigned char   ODR;                    /* ODR          */
            union {                                 /* STR          */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char DBU7:1;       /*   DBU        */
                        unsigned char DBU6:1;       /*   DBU        */
                        unsigned char DBU5:1;       /*   DBU        */
                        unsigned char DBU4:1;       /*   DBU        */
                        unsigned char CD   :1;      /*   C/D        */
```

（続く）

```
                        unsigned char DBU2:1;       /*   DBU        */
                        unsigned char IBF :1;       /*   IBF        */
                        unsigned char OBF :1;       /*   OBF        */
                        }      BIT;                 /*              */
                char               wk2[5];          /*              */
};                                                  /*              */
union un_sbycr {                                    /* union SBYCR  */
                unsigned char BYTE;                 /* Byte Access  */
                struct {                            /* Bit  Access  */
                        unsigned char SSBY :1;      /*   SSBY       */
                        unsigned char STS  :3;      /*   STS        */
                        unsigned char      :1;      /*              */
                        unsigned char SCK  :3;      /*   SCK        */
                        }      BIT;                 /*              */
};                                                  /*              */
union un_lpwrcr {                                   /* union LPWRCR */
                 unsigned char BYTE;                /* Byte Access  */
                 struct {                           /* Bit Access   */
                        unsigned char DTON :1;      /*   DTON       */
                        unsigned char LSON :1;      /*   LSON       */
                        unsigned char NESEL:1;      /*   NESEL      */
                        unsigned char EXCLE:1;      /*   EXCLE      */
                        }      BIT;                 /*              */
};                                                  /*              */
union un_mstpcr {                                   /* union MSTPCR */
                 unsigned int WORD;                 /* Word Access  */
                 struct {                           /* Byte Access  */
                        unsigned char H;            /*   MSTPCRH    */
                        unsigned char L;            /*   MSTPCRL    */
                        }      BYTE;                /*              */
                 struct {                           /* Bit Access   */
                        unsigned char wk :1;        /*              */
                        unsigned char B14:1;        /*   DTC        */
                        unsigned char B13:1;        /*   FRT        */
                        unsigned char B12:1;        /*   TMR0,TMR1  */
                        unsigned char B11:1;        /*   PWM,PWMX   */
                        unsigned char B10:1;        /*   D/A        */
```

（続く）

```
                        unsigned char B9 :1;        /*   A/D       */
                        unsigned char B8 :1;        /*   TMRX,TMRY */
                        unsigned char B7 :1;        /*   SCI0      */
                        unsigned char B6 :1;        /*   SCI1      */
                        unsigned char B5 :1;        /*   SCI2      */
                        unsigned char B4 :1;        /*   IIC0      */
                        unsigned char B3 :1;        /*   IIC1      */
                        unsigned char B2 :1;        /*   HIF       */
                        }      BIT;                 /*             */
};                                                  /*             */
union un_stcr {                                     /* union STCR  */
        unsigned char BYTE;                         /* Byte Access */
        struct {                                    /* Bit  Access */
                        unsigned char IICS :1;      /*   IICS      */
                        unsigned char IICX1:1;      /*   IICX1     */
                        unsigned char IICX0:1;      /*   IICX0     */
                        unsigned char IICE :1;      /*   IICE      */
                        unsigned char FLSHE:1;      /*   FLSHE     */
                        unsigned char      :1;      /*             */
                        unsigned char ICKS1:1;      /*   ICKS1     */
                        unsigned char ICKS0:1;      /*   ICKS0     */
                        }      BIT;                 /*             */
};                                                  /*             */
union un_syscr {                                    /* union SYSCR */
        unsigned char BYTE;                         /* Byte Access */
        struct {                                    /* Bit  Access */
                        unsigned char CS2E :1;      /*   CS2E      */
                        unsigned char IOSE :1;      /*   IOSE      */
                        unsigned char INTM :2;      /*   INTM      */
                        unsigned char XRST :1;      /*   XRST      */
                        unsigned char NMIEG:1;      /*   NMIEG     */
                        unsigned char HIE  :1;      /*   HIE       */
                        unsigned char RAME :1;      /*   RAME      */
                        }      BIT;                 /*             */
};                                                  /*             */
union un_mdcr {                                     /* union MDCR  */
        unsigned char BYTE;                         /* Byte Access */
```

（続く）

```
                struct {                                /* Bit  Access  */
                        unsigned char EXPE:1;           /*   EXPE       */
                        unsigned char     :5;           /*              */
                        unsigned char MDS :2;           /*   MDS        */
                        }       BIT;                    /*              */
};                                                      /*              */
union st_sci {                                          /* struct SCI   */
              union {                                   /* SMR          */
                    unsigned char BYTE;                 /* Byte Access  */
                    struct {                            /* Bit  Access  */
                           unsigned char CA   :1;       /*   C/A        */
                           unsigned char CHR  :1;       /*   CHR        */
                           unsigned char PE   :1;       /*   PE         */
                           unsigned char OE   :1;       /*   O/E        */
                           unsigned char STOP :1;       /*   STOP       */
                           unsigned char MP   :1;       /*   MP         */
                           unsigned char CKS  :2;       /*   CKS        */
                           }       BIT;                 /*              */
                    }      SMR;                         /*              */
              unsigned char BRR;                        /* BRR          */
                union {                                 /* SCR          */
                      unsigned char BYTE;               /* Byte Access  */
                      struct {                          /* Bit  Access  */
                             unsigned char TIE  :1;     /*   TIE        */
                             unsigned char RIE  :1;     /*   RIE        */
                             unsigned char TE   :1;     /*   TE         */
                             unsigned char RE   :1;     /*   RE         */
                             unsigned char MPIE :1;     /*   MPIE       */
                             unsigned char TEIE :1;     /*   TEIE       */
                             unsigned char CKE  :2;     /*   CKE        */
                             }       BIT;               /*              */
                      }       SCR;                      /*              */
              unsigned char TDR;                        /* TDR          */
                union {                                 /* SSR          */
                      unsigned char BYTE;               /* Byte Access  */
                      struct {                          /* Bit  Access  */
                             unsigned char TDRE:1;      /*   TDRE       */
```

（続く）

```
                        unsigned char RDRF:1;       /*   RDRF       */
                        unsigned char ORER:1;       /*   ORER       */
                        unsigned char FER :1;       /*   FER        */
                        unsigned char PER :1;       /*   PER        */
                        unsigned char TEND:1;       /*   TEND       */
                        unsigned char MPB :1;       /*   MPB        */
                        unsigned char MPBT:1;       /*   MPBT       */
                       }       BIT;                 /*              */
                 }          SSR;                    /*              */
           unsigned char    RDR;                    /* RDR          */
           union {                                  /* SCMR         */
                 unsigned char BYTE;                /* Byte Access  */
                 struct {                           /* Bit  Access  */
                        unsigned char wk   :4;      /*              */
                        unsigned char SDIR:1;       /*   SDIR       */
                        unsigned char SINV:1;       /*   SINV       */
                        unsigned char     :1;       /*              */
                        unsigned char SMIF:1;       /*   SMIF       */
                       }       BIT;                 /*              */
                 }          SCMR;                   /*              */
};                                                  /*              */
union st_frt {                                      /* struct FRT   */
           union {                                  /* TIER         */
                 unsigned char BYTE;                /* Byte Access  */
                 struct {                           /* Bit  Access  */
                        unsigned char ICIAE:1;      /*   ICIAE      */
                        unsigned char ICIBE:1;      /*   ICIBE      */
                        unsigned char ICICE:1;      /*   ICICE      */
                        unsigned char ICIDE:1       /*   ICIDE      */
                        unsigned char OCIAE:1;      /*   OCIAE      */
                        unsigned char OCIBE:1;      /*   OCIBE      */
                        unsigned char OVIE :1;      /*   OVIE       */
                        }      BIT;                 /*              */
                 }          TIER;                   /*              */
           union {                                  /* TCSR         */
                 unsigned char BYTE;                /* Byte Access  */
                 struct {                           /* Bit  Access  */
```

（続く）

```
                unsigned char ICFA  :1;     /*   ICFA       */
                unsigned char ICFB  :1;     /*   ICFB       */
                unsigned char ICFC  :2;     /*   ICFC       */
                unsigned char ICFD  :1;     /*   ICFD       */
                unsigned char OCFA  :1;     /*   OCFA       */
                unsigned char OCFB  :1;     /*   OCFB       */
                unsigned char OVF   :1;     /*   OVF        */
                unsigned char CCLRA :1;     /*   CCLRA      */
                }       BIT;                /*              */
        }       TCSR;                       /*              */
unsigned int    FRC;                        /* FRC          */
unsigned int    OCRA;                       /* OCRA or OCRB */
union {                                     /* TCR          */
        unsigned char BYTE;                 /* Byte Access  */
        struct {                            /* Bit  Access  */
                unsigned char IEDGA:1;      /*   IEDGA      */
                unsigned char IEDGB:1;      /*   IEDGB      */
                unsigned char IEDGC:1;      /*   IEDGC      */
                unsigned char IEDGD:1;      /*   IEDGD      */
                unsigned char BUFEA:1;      /*   BUFEA      */
                unsigned char BUFEB:1;      /*   BUFEB      */
                unsigned char CKS  :2;      /*   CKS        */
                }       BIT;                /*              */
        }       TCR;                        /*              */
union {                                     /* TCSR         */
        unsigned char BYTE;                 /* Byte Access  */
        struct {                            /* Bit Access   */
                unsigned char ICRDMS:1;     /*   ICRDMS     */
                unsigned char OCRAMS:1;     /*   OCRAMS     */
                unsigned char ICRS  :1;     /*   ICRS       */
                unsigned char OCRS  :1;     /*   OCRS       */
                unsigned char OEA   :1;     /*   OEA        */
                unsigned char OEB   :1;     /*   OEB        */
                unsigned char OLVLA :1;     /*   OLVLA      */
                unsigned char OLVLB :1;     /*   OLVLB      */
                }       BIT;                /*              */
        }       TOCR;                       /*              */
```

（続く）

```
                unsigned int ICRA;                      /* ICRA or OCRAR*/
                unsigned int ICRB;                      /* ICRB or OCRAF*/
                unsigned int ICRC;                      /* ICRC or OCRDM*/
                unsigned int ICRD;                      /* ICRD         */
};                                                      /*              */
union un_pwmx {                                         /* struct PWMX  */
              struct {                                  /*              */
                      union {                           /* DACR         */
                     unsigned char BYTE;                /* Byte Access  */
                     struct {                           /* Bit  Access  */
                             unsigned char TEST    :1;  /*   TEST       */
                             unsigned char PWME    :1;  /*   PWME       */
                             unsigned char ICICE   :1;  /*   ICICE      */
                             unsigned char char    :2;  /*              */
                             unsigned char char OEB:1;  /*   OEB        */
                             unsigned char char OEA:1;  /*   OEA        */
                             unsigned char char OS :1;  /*   OS         */
                             unsigned char char CKS:1;  /*   CKS        */
                             }      BIT;                /*              */
                     }      ST_DACR;                    /*              */
                     char   wk[5];                      /*              */
                     union {                            /* DACNT        */
                            unsigned int WORD;          /* Word Access  */
                            struct {                    /* Bit Access   */
                                   unsigned int wk  :15;/*              */
                                   unsigned int REGS: 1;/*   REGS       */
                                    }      BIT;          /*              */
                           }      ST_DACNT;             /*              */
              } REGS1;                                  /*              */
              struct {                                  /*              */
                     union {                            /* DADRA        */
                            unsigned int WORD;          /* Word Access  */
                            struct {                    /* Bit  Access  */
                                   unsigned int wk :14; /*              */
                                   unsigned int CFS: 1; /*   CFS        */
                            }      BIT;                 /*              */
                     }      ST_DADRA;                  /*              */
```

（続く）

```
                char wk[4];                             /*              */
                union {                                 /* DADRA        */
                      unsigned int WORD;                /* Word Access  */
                      struct {                          /* Bit  Access  */
                             unsigned int wk   :14;     /*              */
                             unsigned int CFS  : 1;     /*   CFS        */
                             unsigned int REGS : 1;     /*   REGS       */
                             }      BIT;                /*              */
                      }      ST_DADRB;                  /*              */
                } REGSO;                                /*              */
};                                                      /*              */
struct st_p1 {                                          /* struct P1    */
             union {                                    /* P1PCR        */
                   unsigned char BYTE;                  /* Byte Access  */
                   struct {                             /* Bit  Access  */
                          unsigned char B7:1;           /*   Bit 7      */
                          unsigned char B6:1;           /*   Bit 6      */
                          unsigned char B5:1;           /*   Bit 5      */
                          unsigned char B4:1;           /*   Bit 4      */
                          unsigned char B3:1;           /*   Bit 3      */
                          unsigned char B2:1;           /*   Bit 2      */
                          unsigned char B1:1;           /*   Bit 1      */
                          unsigned char B0:1;           /*   Bit 0      */
                          }      BIT;                   /*              */
                    }      PCR;                         /*              */
             char wk1[3];                               /*              */
             unsigned char DDR;                         /* P1DDR        */
             char wk2;                                  /*              */
             union {                                    /* P1DR         */
                    unsigned char BYTE;                 /* Byte Access  */
                    struct {                            /* Bit  Access  */
                           unsigned char B7:1;          /*   Bit 7      */
                           unsigned char B6:1;          /*   Bit 6      */
                           unsigned char B5:1;          /*   Bit 5      */
                           unsigned char B4:1;          /*   Bit 4      */
                           unsigned char B3:1;          /*   Bit 3      */
                           unsigned char B2:1;          /*   Bit 2      */
```

（続く）

```
                        unsigned char B1:1;         /*   Bit 1     */
                        unsigned char B0:1;         /*   Bit 0     */
                        }       BIT;                /*             */
                    }           DR;                 /*             */
};                                                  /*             */
struct st_p3 {                                      /* struct P3   */
            union {                                 /* P3PCR       */
                  unsigned char BYTE;               /* Byte Access */
                  struct {                          /* Bit  Access */
                        unsigned char B7:1;         /*   Bit 7     */
                        unsigned char B6:1;         /*   Bit 6     */
                        unsigned char B5:1;         /*   Bit 5     */
                        unsigned char B4:1;         /*   Bit 4     */
                        unsigned char B3:1;         /*   Bit 3     */
                        unsigned char B2:1;         /*   Bit 2     */
                        unsigned char B1:1;         /*   Bit 1     */
                        unsigned char B0:1;         /*   Bit 0     */
                        }       BIT;                /*             */
                  }             PCR;                /*             */
                  char          wk1[5];             /*             */
                  unsigned char DDR;                /* P3DDR       */
                  char          wk2;                /*             */
             union {                                /* P3DR        */
                  unsigned char  BYTE;              /* Byte Access */
                  struct {                          /* Bit  Access */
                         unsigned char B7:1;        /*   Bit 7     */
                         unsigned char B6:1;        /*   Bit 6     */
                         unsigned char B5:1;        /*   Bit 5     */
                         unsigned char B4:1;        /*   Bit 4     */
                         unsigned char B3:1;        /*   Bit 3     */
                         unsigned char B2:1;        /*   Bit 2     */
                         unsigned char B1:1;        /*   Bit 1     */
                         unsigned char B0:1;        /*   Bit 0     */
                        }       BIT;                /*             */
                  }             DR;                 /*             */
};                                                  /*             */
struct st_p4 {                                      /* struct P4   */
```

（続く）

```
                unsigned char DDR;                      /* P4DDR        */
                char          wk;                       /*              */
                union {                                 /* P4DR         */
                      unsigned char BYTE;               /* Byte Access  */
                      struct {                          /* Bit  Access  */
                             unsigned char B7:1;        /*   Bit 7      */
                             unsigned char B6:1;        /*   Bit 6      */
                             unsigned char B5:1;        /*   Bit 5      */
                             unsigned char B4:1;        /*   Bit 4      */
                             unsigned char B3:1;        /*   Bit 3      */
                             unsigned char B2:1;        /*   Bit 2      */
                             unsigned char B1:1;        /*   Bit 1      */
                             unsigned char B0:1;        /*   Bit 0      */
                             }       BIT;               /*              */
                      }              DR;                /*              */
    };                                                  /*              */
    struct st_p5 {                                      /* struct P5    */
                unsigned char DDR;                      /* P5DDR        */
                char          wk;                       /*              */
                union {                                 /* P5DR         */
                      unsigned char BYTE;               /* Byte Access  */
                      struct {                          /* Bit  Access  */
                             unsigned char wk:5;        /*   Bit 7-3    */
                             unsigned char B2:1;        /*   Bit 2      */
                             unsigned char B1:1;        /*   Bit 1      */
                             unsigned char B0:1;        /*   Bit 0      */
                             }       BIT;               /*              */
                      }              DR;                /*              */
    };                                                  /*              */
    struct st_p6 {                                      /* struct P6    */
                unsigned char DDR;                      /* P6DDR        */
                char          wk1;                      /*              */
                union {                                 /* P6DR         */
                      unsigned char BYTE;               /* Byte Access  */
                      struct {                          /* Bit  Access  */
                             unsigned char B7:1;        /*   Bit 7      */
                             unsigned char B6:1;        /*   Bit 6      */
```

（続く）

```
                        unsigned char B5:1;         /*   Bit 5     */
                        unsigned char B4:1;         /*   Bit 4     */
                        unsigned char B3:1;         /*   Bit 3     */
                        unsigned char B2:1;         /*   Bit 2     */
                        unsigned char B1:1;         /*   Bit 1     */
                        unsigned char B0:1;         /*   Bit 0     */
                        }       BIT;                /*             */
                  }             DR;                 /*             */
             char               wk2[54];            /*             */
             union {                                /* P6PCR       */
                   unsigned char BYTE;              /* Byte Access */
                   struct {                         /* Bit  Access */
                          unsigned char B7:1;       /*   Bit 7     */
                          unsigned char B6:1;       /*   Bit 6     */
                          unsigned char B5:1;       /*   Bit 5     */
                          unsigned char B4:1;       /*   Bit 4     */
                          unsigned char B3:1;       /*   Bit 3     */
                          unsigned char B2:1;       /*   Bit 2     */
                          unsigned char B1:1;       /*   Bit 1     */
                          unsigned char B0:1;       /*   Bit 0     */
                          }       BIT;              /*             */
                   }      PCR;                     /*             */
};                                                  /*             */
struct st_p7 {                                      /* struct P7   */
            union {                                 /* P7PIN       */
                  unsigned char BYTE;               /* Byte Access */
                  struct {                          /* Bit  Access */
                         unsigned char B7:1;        /*   Bit 7     */
                         unsigned char B6:1;        /*   Bit 6     */
                         unsigned char B5:1;        /*   Bit 5     */
                         unsigned char B4:1;        /*   Bit 4     */
                         unsigned char B3:1;        /*   Bit 3     */
                         unsigned char B2:1;        /*   Bit 2     */
                         unsigned char B1:1;        /*   Bit 1     */
                         unsigned char B0:1;        /*   Bit 0     */
                         }      BIT;                /*             */
                  }      PIN;                      /*             */
```

（続く）

```
};                                                  /*              */
struct st_p8 {                                      /* struct P8    */
             unsigned char DDR;                     /* P8DDR        */
             char          wk;                      /*              */
             union {                                /* P8DR         */
                   unsigned char BYTE;              /* Byte Access  */
                   struct {                         /* Bit  Access  */
                          unsigned char wk:1;       /*   Bit 7      */
                          unsigned char B6:1;       /*   Bit 6      */
                          unsigned char B5:1;       /*   Bit 5      */
                          unsigned char B4:1;       /*   Bit 4      */
                          unsigned char B3:1;       /*   Bit 3      */
                          unsigned char B2:1;       /*   Bit 2      */
                          unsigned char B1:1;       /*   Bit 1      */
                          unsigned char B0:1;       /*   Bit 0      */
                          }      BIT;               /*              */
                   }             DR;                /*              */
};                                                  /*              */
struct st_p9 {                                      /* struct P9    */
              unsigned char DDR;                    /* P9DDR        */
              union {                               /* P9DR         */
                    unsigned char BYTE;             /* Byte Access  */
                    struct {                        /* Bit  Access  */
                           unsigned char B7:1;      /*   Bit 7      */
                           unsigned char B6:1;      /*   Bit 6      */
                           unsigned char B5:1;      /*   Bit 5      */
                           unsigned char B4:1;      /*   Bit 4      */
                           unsigned char B3:1;      /*   Bit 3      */
                           unsigned char B2:1;      /*   Bit 2      */
                           unsigned char B1:1;      /*   Bit 1      */
                           unsigned char B0:1;      /*   Bit 0      */
                           }      BIT;              /*              */
                    }             DR;               /*              */
};                                                  /*              */
struct st_bsc {                                     /* struct BSC   */
              union {                               /* BCR          */
                    unsigned char BYTE;             /* Byte Access  */
```

（続く）

```
                struct {                          /* Bit  Access  */
                        unsigned char ICIS1 :1; /*   ICIS1      */
                        unsigned char ICIS0 :1; /*   ICIS0      */
                        unsigned char BRSTRM:1; /*   BRSTRM     */
                        unsigned char BRSTS1:1; /*   BRSTS1     */
                        unsigned char BRSTS0:1; /*   BRSTS0     */
                        unsigned char       :1;/*              */
                        unsigned char IOS   :2; /*   IOS        */
                        }     BIT;              /*              */
                }             BCR;              /*              */
          union {                                 /* WSCR         */
                unsigned char BYTE;               /* Byte Access  */
                struct {                          /* Bit  Access  */
                        unsigned char RAMS:1;     /*   RAMS       */
                        unsigned char RAM0:1;     /*   RAM0       */
                        unsigned char ABW :1;     /*   ABW        */
                        unsigned char AST :1;     /*   AST        */
                        unsigned char WMS :2;     /*   WMS        */
                        unsigned char WC  :2;     /*   WC         */
                        }     BIT;                /*              */
                }             WSCR;               /*              */
};                                                /*              */
struct st_tmr {                                   /* struct TMR   */
          union {                                 /* TCR0         */
                unsigned char BYTE;               /* Byte Access  */
                struct {                          /* Bit  Access  */
                        unsigned char CMIEB:1;    /*   CMIEB      */
                        unsigned char CMIEA:1;    /*   CMIEA      */
                        unsigned char OVIE :1;    /*   OVIE       */
                        unsigned char CCLR :2;    /*   CCLR       */
                        unsigned char CKS  :3;    /*   CKS        */
                       }     BIT;                 /*              */
                }             TCR0;               /*              */
          union {                                 /* TCR1         */
                unsigned char BYTE;               /*  Byte Access */
                struct {                          /*  Bit  Access */
                        unsigned char CMIEB:1;    /*   CMIEB      */
```

（続く）

```
                        unsigned char CMIEA:1;      /*   CMIEA      */
                        unsigned char OVIE :1;      /*   OVIE       */
                        unsigned char CCLR :2;      /*   CCLR       */
                        unsigned char CKS  :3;      /*   CKS        */
                        }      BIT;                 /*              */
                  }            TCR1;                /*              */
            union {                                 /* TCSR0        */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMFB:1;       /*   CMFB       */
                        unsigned char CMFA:1;       /*   CMFA       */
                        unsigned char OVF :1;       /*   OVF        */
                        unsigned char ADTE:1;       /*   ADTE       */
                        unsigned char OS  :4;       /*   OS         */
                        }      BIT;                 /*              */
                  }            TCSR0;               /*              */
            union {                                 /* TCSR1        */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMFB:1;       /*   CMFB       */
                        unsigned char CMFA:1;       /*   CMFA       */
                        unsigned char OVF :1;       /*   OVF        */
                        unsigned char     :1;       /*              */
                        unsigned char OS  :4;       /*   OS         */
                        }      BIT;                 /*              */
                  }            TCSR1;               /*              */
            unsigned int       TCORA;               /* TCORA        */
            unsigned int       TCORB;               /* TCORB        */
            unsigned int       TCNT;                /* TCNT         */
};                                                  /*              */
struct st_tmr0 {                                    /* struct TMR0  */
            union {                                 /* TCR          */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMIEB:1;      /*   CMIEB      */
                        unsigned char CMIEA:1;      /*   CMIEA      */
                        unsigned char OVIE :1;      /*   OVIE       */
```

（続く）

```
                        unsigned char CCLR :2;      /*   CCLR      */
                        unsigned char CKS  :3;      /*   CKS       */
                        }      BIT;                 /*             */
                  }             TCR;                /*             */
            char                wk1;                /*             */
            union {                                 /* TCSR        */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMFB:1;       /*   CMFB      */
                        unsigned char CMFA:1;       /*   CMFA      */
                        unsigned char OVF :1;       /*   OVF       */
                        unsigned char ADTE:1;       /*   ADTE      */
                        unsigned char OS  :4;       /*   OS        */
                        }      BIT;                 /*             */
                  }             TCSR;               /*             */
            char                wk2;                /*             */
            unsigned char       TCORA;              /* TCORA       */
            char                wk3;                /*             */
            unsigned char       TCORB;              /* TCORB       */
            char                wk4;                /*             */
            unsigned char       TCNT;               /* TCNT        */
};                                                  /*             */
struct st_tmr1 {                                    /* struct TMR1 */
            union {                                 /* TCR         */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMIEB:1;      /*   CMIEB     */
                        unsigned char CMIEA:1;      /*   CMIEA     */
                        unsigned char OVIE :1;      /*   OVIE      */
                        unsigned char CCLR :2;      /*   CCLR      */
                        unsigned char CKS  :3;      /*   CKS       */
                        }      BIT;                 /*             */
                  }             TCR;                /*             */
            char                wk1;                /*             */
            union {                                 /* TCSR        */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
```

（続く）

```
                        unsigned char CMFB:1;       /*   CMFB       */
                        unsigned char CMFA:1;       /*   CMFA       */
                        unsigned char OVF :1;       /*   OVF        */
                        unsigned char     :1;       /*              */
                        unsigned char OS  :4;       /*   OS         */
                        }      BIT;                 /*              */
                  }            TCSR;                /*              */
            char               wk2;                 /*              */
            unsigned char      TCORA;               /* TCORA        */
            char               wk3;                 /*              */
            unsigned char      TCORB;               /* TCORB        */
            char               wk4;                 /*              */
            unsigned char      TCNT;                /* TCNT         */
};                                                  /*              */
struct st_tmrx {                                    /* struct TMRX  */
            union {                                 /* TCR          */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMIEB:1;      /*   CMIEB      */
                        unsigned char CMIEA:1;      /*   CMIEA      */
                        unsigned char OVIE :1;      /*   OVIE       */
                        unsigned char CCLR :2;      /*   CCLR       */
                        unsigned char CKS  :3;      /*   CKS        */
                        }      BIT;                 /*              */
                  }            TCR;                 /*              */
            union {                                 /* TCSR         */
                  unsigned char BYTE;               /*  Byte Access */
                  struct {                          /*  Bit  Access */
                        unsigned char CMFB:1;       /*   CMFB       */
                        unsigned char CMFA:1;       /*   CMFA       */
                        unsigned char OVF :1;       /*   OVF        */
                        unsigned char ICF :1;       /*   ICF        */
                        unsigned char OS  :4;       /*   OS         */
                        }      BIT;                 /*              */
                  }            TCSR;                /*              */
            unsigned char      TICRR;               /* TICRR        */
            unsigned char      TICRF;               /* TICRF        */
```

（続く）

```
                unsigned char     TCNT;                 /* TCNT         */
                unsigned char     TCORC;                /* TCORC        */
                unsigned char     TCORA;                /* TCORA        */
                unsigned char     TCORB;                /* TCORB        */
};                                                      /*              */
struct st_tmry {                                        /* struct TMRY  */
                union {                                 /* TCR          */
                      unsigned char BYTE;               /*  Byte Access */
                      struct {                          /*  Bit  Access */
                             unsigned char CMIEB:1;     /*   CMIEB      */
                             unsigned char CMIEA:1;     /*   CMIEA      */
                             unsigned char OVIE :1;     /*   OVIE       */
                             unsigned char CCLR :2;     /*   CCLR       */
                             unsigned char CKS  :3;     /*   CKS        */
                             }      BIT;                /*              */
                      }             TCR;                /*              */
                union {                                 /* TCSR         */
                      unsigned char BYTE;               /*  Byte Access */
                      struct {                          /*  Bit  Access */
                             unsigned char CMFB:1;      /*   CMFB       */
                             unsigned char CMFA:1;      /*   CMFA       */
                             unsigned char OVF :1;      /*   OVF        */
                             unsigned char ICIE:1;      /*   ICIE       */
                             unsigned char OS  :4;      /*   OS         */
                             }      BIT;                /*              */
                      }             TCSR;               /*              */
                unsigned char       TCORA;              /* TCORA        */
                unsigned char       TCORB;              /* TCORB        */
                unsigned char       TCNT;               /* TCNT         */
                union {                                 /* TISR         */
                      unsigned char BYTE;               /*  Byte Access */
                      struct {                          /*  Bit  Access */
                             unsigned char wk:7;        /*              */
                             unsigned char IS:1;        /*   IS         */
                             }      BIT;                /*              */
                      }             TISR;               /*              */
};                                                      /*              */
```

（続く）

```
struct st_ad {                                          /* struct A/D   */
             unsigned int      DRA;                     /* ADDRA        */
             unsigned int      DRB;                     /* ADDRB        */
             unsigned int      DRC;                     /* ADDRC        */
             unsigned int      DRD;                     /* ADDRD        */
             union {                                    /* ADCSR        */
                    unsigned char BYTE;                 /*  Byte Access */
                    struct {                            /*  Bit  Access */
                           unsigned char ADF :1;        /*   ADF        */
                           unsigned char ADIE:1;        /*   ADIE       */
                           unsigned char ADST:1;        /*   ADST       */
                           unsigned char SCAN:1;        /*   SCAN       */
                           unsigned char CKS :1;        /*   CKS        */
                           unsigned char CH  :3;        /*   CH         */
                           }      BIT;                  /*              */
                    }             CSR;                  /*              */
             union {                                    /* ADCR         */
                    unsigned char BYTE;                 /*  Byte Access */
                    struct {                            /*  Bit  Access */
                           unsigned char TRGS:2;        /*   TRGS       */
                           }      BIT;                  /*              */
                    }             CR;                   /*              */
};                                                      /*              */
struct st_da {                                          /* struct D/A   */
             unsigned char     DR0;                     /* DADR0        */
             unsigned char     DR1;                     /* DADR1        */
             union {                                    /* DACR         */
                    unsigned char BYTE;                 /*  Byte Access */
                    struct {                            /*  Bit  Access */
                           unsigned char DA0E1:1;       /*   DA0E1      */
                           unsigned char DA0E0:1;       /*   DA0E0      */
                           unsigned char DAE  :1;       /*   DAE        */
                           }      BIT;                  /*              */
                    }             CR;                   /*              */
};                                                      /*              */
struct st_tc {                                          /* struct TC    */
             union {                                    /* TCONRI       */
```

（続く）

```
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char SIMOD:2;      /*    SIMOD     */
                unsigned char SCONE:1;      /*    SCONE     */
                unsigned char ICST :1;      /*    ICST      */
                unsigned char HFINV:1;      /*    HFINV     */
                unsigned char VFINV:1;      /*    VFINV     */
                unsigned char HIINV:1;      /*    HIINV     */
                unsigned char VIINV:1;      /*    VIINV     */
                }       BIT;                /*              */
        }               TCONRI;             /*              */
union {                                     /* TCONR0       */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char HOE   :1;     /*    HOE       */
                unsigned char VOE   :1;     /*    VOE       */
                unsigned char CLOE  :1;     /*    CLOE      */
                unsigned char CBOE  :1;     /*    CBOE      */
                unsigned char HOINV :1;     /*    HOINV     */
                unsigned char VOINV :1;     /*    VOINV     */
                unsigned char CLOINV:1;     /*    CLOINV    */
                unsigned char CBOINV:1;     /*    CBOINV    */
                }       BIT;                /*              */
        }               TCONR0;             /*              */
union {                                     /* TCONRS       */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
                unsigned char TMRXY :1;     /*    TMRXY     */
                unsigned char ISGENE:1;     /*    ISGENE    */
                unsigned char HOMOD :2;     /*    HOMOD     */
                unsigned char VOMOD :2;     /*    VOMOD     */
                unsigned char CLMOD :2;     /*    CLMOD     */
                }       BIT;                /*              */
        }               TCONRS;             /*              */
union {                                     /* SEDGR        */
        unsigned char BYTE;                 /*  Byte Access */
        struct {                            /*  Bit  Access */
```

（続く）

```
                        unsigned char VEDG :1;      /*   VEDG       */
                        unsigned char HEDG  :1;     /*   HEDG       */
                        unsigned char CEDG  :1;     /*   CEDG       */
                        unsigned char HFEDG:1;      /*   HFEDG      */
                        unsigned char VFEDG:1;      /*   VFEDG      */
                        unsigned char PREQF:1;      /*   PREQF      */
                        unsigned char IHI  :1;      /*   IHI        */
                        unsigned char IVI  :1;      /*   IVI        */
                        }      BIT;                 /*              */
                    }          SEDGR;               /*              */
};                                                  /*              */
#define KBCOMP  (*(volatile union  un_kbcomp*)0xFFFEE4) /* KBCOMPAddress */
#define IIC0    (*(volatile struct st_iic0  *)0xFFFFD8) /* IIC0   Address*/
#define IIC1    (*(volatile struct st_iic1  *)0XFFFF88) /* IIC1   Address*/
#define ICDR    EQU.ICE1.UN_ICDR                    /* ICDR   Change */
#define ICMR    EQU.ICE1.UN_ICMR                    /* ICDR   Change */
#define SAR     EQU.ICE0.UN_SAR                     /* SAR    Change */
#define SARX    EQU.ICE0.UN_SARX                    /* SARX   Change */
#define DDCSWR  (*(volatile union  un_ddcswr*)0xFFFEE6) /* DDCSWRAddress */
#define INTC    (*(volatile struct st_intc  *)0xFFFEE8) /* INTC   Address*/
#define DTC     (*(volatile struct st_dtc   *)0xFFFEEE) /* DTC    Address*/
#define FLASH   (*(volatile struct st_flash *)0xFFFF80) /* FLASH Address */
#define PWM     (*(volatile struct st_pwm   *)0xFFFF82) /* PWM    Address*/
#define HIF     (*(volatile struct st_hif   *)0xFFFF83) /* HIF    Address*/
#define HIF1    (*(volatile struct st_hif1  *)0xFFFFF4) /* HIF1   Address*/
#define HIF2    (*(volatile struct st_hif2  *)0xFFFFFC) /* HIF1   Address*/
#define SBYCR   (*(volatile union  un_sbycr *)0xFFFF84) /* SBYCR Address */
#define LPWRCR  (*(volatile union  un_lpwrcr*)0xFFFF85) /* LPWRCRAddress */
#define MSTPCR  (*(volatile union  un_mstpcr*)0xFFFF86) /* MSTPCRAddress */
#define STCR    (*(volatile union  un_stcr  *)0xFFFFC3) /* STCR  Address */
#define SYSCR   (*(volatile union  un_syscr *)0xFFFFC4) /* SYSCR Address */
#define MDCR    (*(volatile union  un_mdcr  *)0xFFFFC5) /* MDCR  Address */
#define SCI0    (*(volatile struct st_sci0  *)0xFFFFD8) /* SCI0  Address */
#define SCI1    (*(volatile struct st_sci1  *)0xFFFF88) /* SCI1  Address */
#define SCI2    (*(volatile struct st_sci2  *)0xFFFFA0) /* SCI2  Address */
#define FRT     (*(volatile struct st_frt   *)0xFFFF90) /* FRT   Address */
#define OCRB    OCRA                                /* OCRB  Change  */
```

（続く）

```
#define OCRAR   ICRA                                    /* OCRAR Change */
#define OCRAF   ICRB                                    /* OCRAF Change */
#define OCRDM   ICRC                                    /* OCRDM Change */
#define PWMX    (*(volatile union  un_pwmx  *)0xFFFFA0)  /* PWMX  Address */
#define DACR    REGS1.ST_DACR                           /* DACR  Change */
#define DACNT   REGS1.ST_DACNT                          /* DACNT Change */
#define DADRA   REGS0.ST_DADRA                          /* DADRA Change */
#define DADRB   REGS0.ST_DADRB                          /* DADRB Change */
#define P1      (*(volatile struct st_p1   *)0xFFFFAC)  /* P1    Address */
#define P2      (*(volatile struct st_p2   *)0xFFFFAD)  /* P2    Address */
#define P3      (*(volatile struct st_p3   *)0xFFFFAE)  /* P3    Address */
#define P4      (*(volatile struct st_p4   *)0xFFFFB5)  /* P4    Address */
#define P5      (*(volatile struct st_p5   *)0xFFFFB8)  /* P5    Address */
#define P6      (*(volatile struct st_p6   *)0xFFFFB9)  /* P6    Address */
#define P7      (*(volatile struct st_p7   *)0xFFFFBE)  /* P7    Address */
#define P8      (*(volatile struct st_p8   *)0xFFFFBD)  /* P8    Address */
#define P9      (*(volatile struct st_p9   *)0xFFFFC0)  /* P9    Address */
#define BSC     (*(volatile struct st_bsc  *)0xFFFFC6)  /* BSC   Address */
#define TMR     (*(volatile struct st_tmr  *)0xFFFFC8)  /* TMR   Address */
#define TMR0    (*(volatile struct st_tmr0 *)0xFFFFC8)  /* TMR0   Address*/
#define TMR1    (*(volatile struct st_tmr1 *)0xFFFFC9)  /* TMR1  Address*/
#define TMRX    (*(volatile struct st_tmrx *)0xFFFFF0)  /* TMRX  Address*/
#define TMRY    (*(volatile struct st_tmry *)0xFFFFF0)  /* TMRY  Address*/
#define AD      (*(volatile struct st_ad   *)0xFFFFE0)  /* A/D   Address */
#define DA      (*(volatile struct st_da   *)0xFFFFF8)  /* D/A   Address */
#define TC      (*(volatile struct st_tc   *)0xFFFFFC)  /* TC    Address */
#define st_hif2 st_hif1                         /* Change Struct HIF2    */
#define st_p2   st_p1                           /* Change Struct P2->P1 */
```

4.1.4 アセンブラ埋め込みファイル説明

本応用例で使用するプログラムリストの中で、スタックの初期化などアセンブリ言語を C 言語の中で使えるようにした、アセンブラ埋め込み機能を使用しています。

C コンパイラ（CH38.EXE）は、アセンブラが埋め込まれると直接オブジェクトファイルを生成することができません。このため、コードオプションで“副ファイル名.src”というアセンブラ展開ファイルを生成し、このファイルをアセンブラ（ASM38.EXE）にてアセンブルしてオブジェクトファイルを生成するようにしてください。

CH38.EXE のアセンブラ展開ファイルを生成するためのコードオプションの指定は、－c＝a としますが、詳細はコンパイラのマニュアルを参照してください。

4.1.5 ファイルのリンク説明

図 4.3 にリンケージ時のサブミットファイルを示します。ベクタ定義ファイル、レジスタ定義ファイルおよび各タスクファイルのリンクは、**図** 4.3 に示すサブミットファイルの情報にしたがいリンケージします。

```
input SMRxd, 2138vec  ····· [1]
lib c : \ch38\lib\c8s26n.lib····· [2]
output SMRxd  ····· [3]
print SMRxd  ····· [4]
start VECT(00000), P(01000), Bramerea(0E100)  ····· [5]
exit
```

[1] ：**ベクタ定義ファイル**（2138vec. obj）、**タスクファイル**（SMRxd. obj）**のオブジェクトファイルをリンク対象とする。**

[2] ：H8S/2600**ノーマルモード用ライブラリ**（c8s26n. lib）**の指定**

[3] ：**オブジェクトファイル名の指定**（SMRxd. abs**で出力される**）

[4] ：**マップファイル名の指定**（SMRxd. map**で出力される**）

[5] ：**開始アドレスの指定（本例ではベクタ**（VECT）**を**H'0000**番地、プログラム**（P）**を**H'1000**番地、未初期化データ領域**（Bramerea）**を**H'E100**番地から配置する）**

図 4.3 **サブミットファイル**

4.2 シングルマスタ送信

4.2.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）に 10 バイトのデータを書き込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地から H'09 番地にデータを書き込みます。
- 書き込むデータは［H'01, H'02, H'03, H'04, H'05, H'06, H'07, H'08, H'09, H'0A］とします。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.4 に H8S/2138 と EEPROM の接続例を示します。

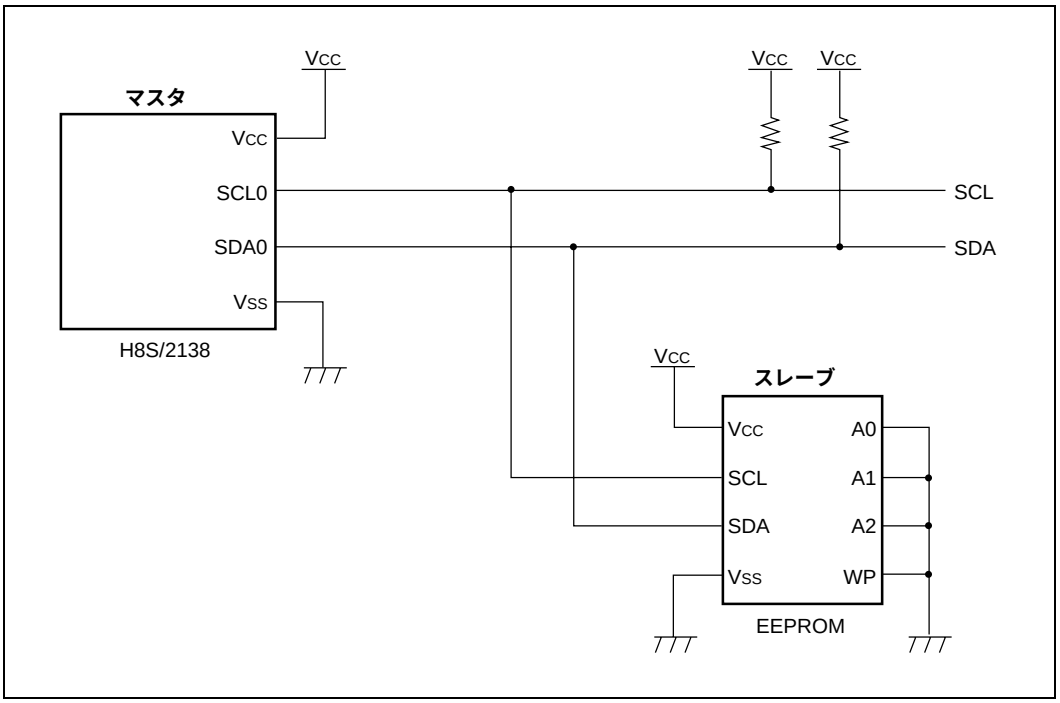


図 4.4 H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.5 に示します。

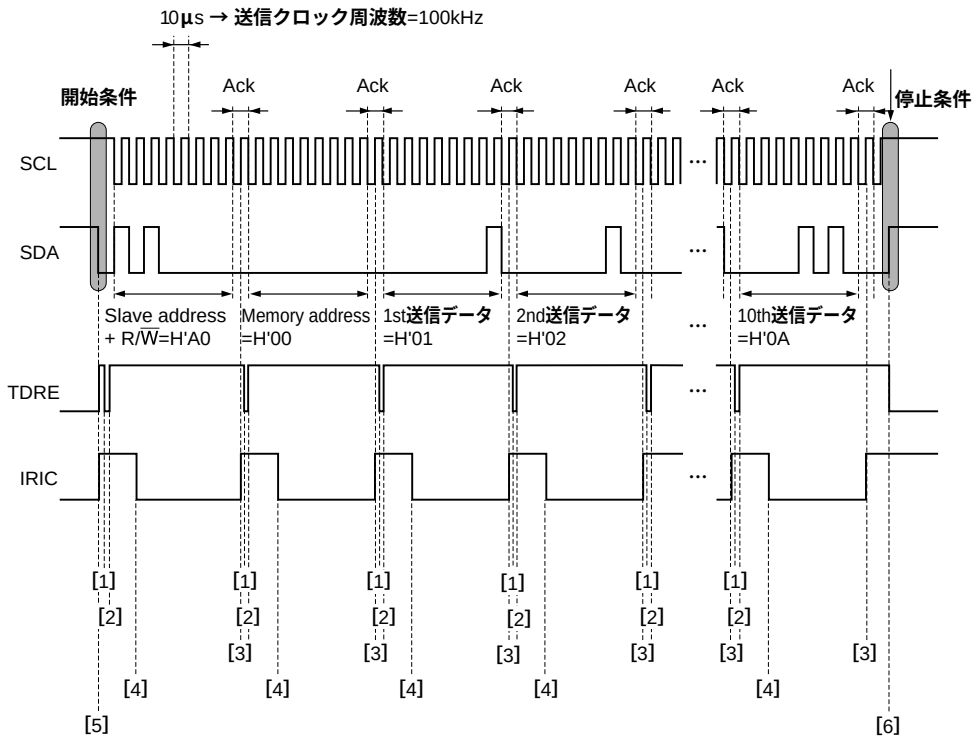
【記号説明】

S	**：開始条件**
SLA	**：**EEPROM**スレーブアドレス**
R/$\overline{W}$	**：送信／受信方向**
A	**：アクノリッジ**
MEA	**：**EEPROM**メモリアドレス**
DATA	**：送信データ**
P	**：停止条件**

図 4.5　**本タスク例で使用する転送フォーマット**

4.2.2 動作説明

図 4.6 に動作原理を示します。



ソフトウェア処理	ハードウェア処理
[1] ICDR0に送信データをライト	TDRE=0（TRS=1の状態でICDRTにデータをライト）
[2] 処理なし	TDRE=1（ICDRTからICDRSにデータを転送）
[3] 処理なし	IRIC=1（データ送信の終了（送信クロックの9クロック目の立ち上がり時））
[4] 送信終了判定のためにIRICを0にクリア	処理なし
[5] 開始条件の発行（BBSY=1、 SCP=0）	IRIC=1、TDRE=1（バスラインの状態から開始条件を検出）
[6] 停止条件の発行（BBSY=0、 SCP=0）	TDRE=0（バスラインの状態から停止条件を検出）

図 4.6 シングルマスタ送信動作原理

4.2.3 ソフトウェア説明

(1) モジュール説明

表 4.1 に本タスク例におけるモジュール説明を示します。

表 4.1 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	Initialize	IIC0 の初期設定
シングルマスタ送信	mst_trs	シングルマスタ送信による EEPROM への 10 バイトのデータの送信
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス＋W データ（H'A0）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信

(2) 使用内蔵レジスタ説明

表 4.2 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.2 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジの連続的な転送を設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMRO、ICDR0／SAR、SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I^2C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1

（続く）

表 4.2 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	TRS	I^2C バスインタフェースを送信モードで使用	H'FFD8 bit4	1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY と組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	EEPROM より送信されたアクノリッジデータを格納	H'FFD9 bit0	−
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、1 ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1

(3) 変数説明

表 4.3 に本タスク例における変数説明を示します。

表 4.3 変数説明

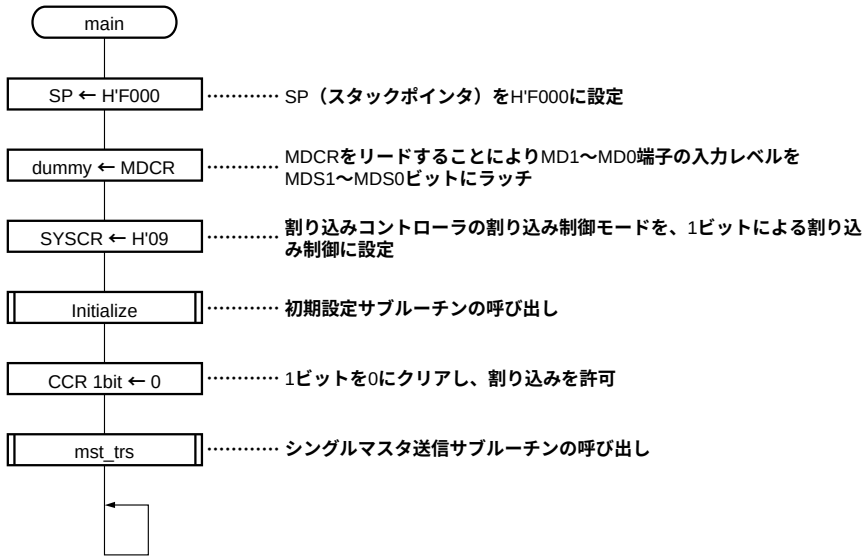
変数	機能	データ長	初期値	使用モジュール名
dt_trs[0]	1 **バイト目送信データ**	1 **バイト**	H'01	mst_trs
dt_trs[1]	2 **バイト目送信データ**	1 **バイト**	H'02	mst_trs
dt_trs[2]	3 **バイト目送信データ**	1 **バイト**	H'03	mst_trs
dt_trs[3]	4 **バイト目送信データ**	1 **バイト**	H'04	mst_trs
dt_trs[4]	5 **バイト目送信データ**	1 **バイト**	H'05	mst_trs
dt_trs[5]	6 **バイト目送信データ**	1 **バイト**	H'06	mst_trs
dt_trs[6]	7 **バイト目送信データ**	1 **バイト**	H'07	mst_trs
dt_trs[7]	8 **バイト目送信データ**	1 **バイト**	H'08	mst_trs
dt_trs[8]	9 **バイト目送信データ**	1 **バイト**	H'09	mst_trs
dt_trs[9]	10 **バイト目送信データ**	1 **バイト**	H'0A	mst_trs
i	**送信データカウンタ**	1 **バイト**	H'00	mst_trs
dummy	MDCR **リード値**	1 **バイト**	−	main

(4) 使用 RAM 説明

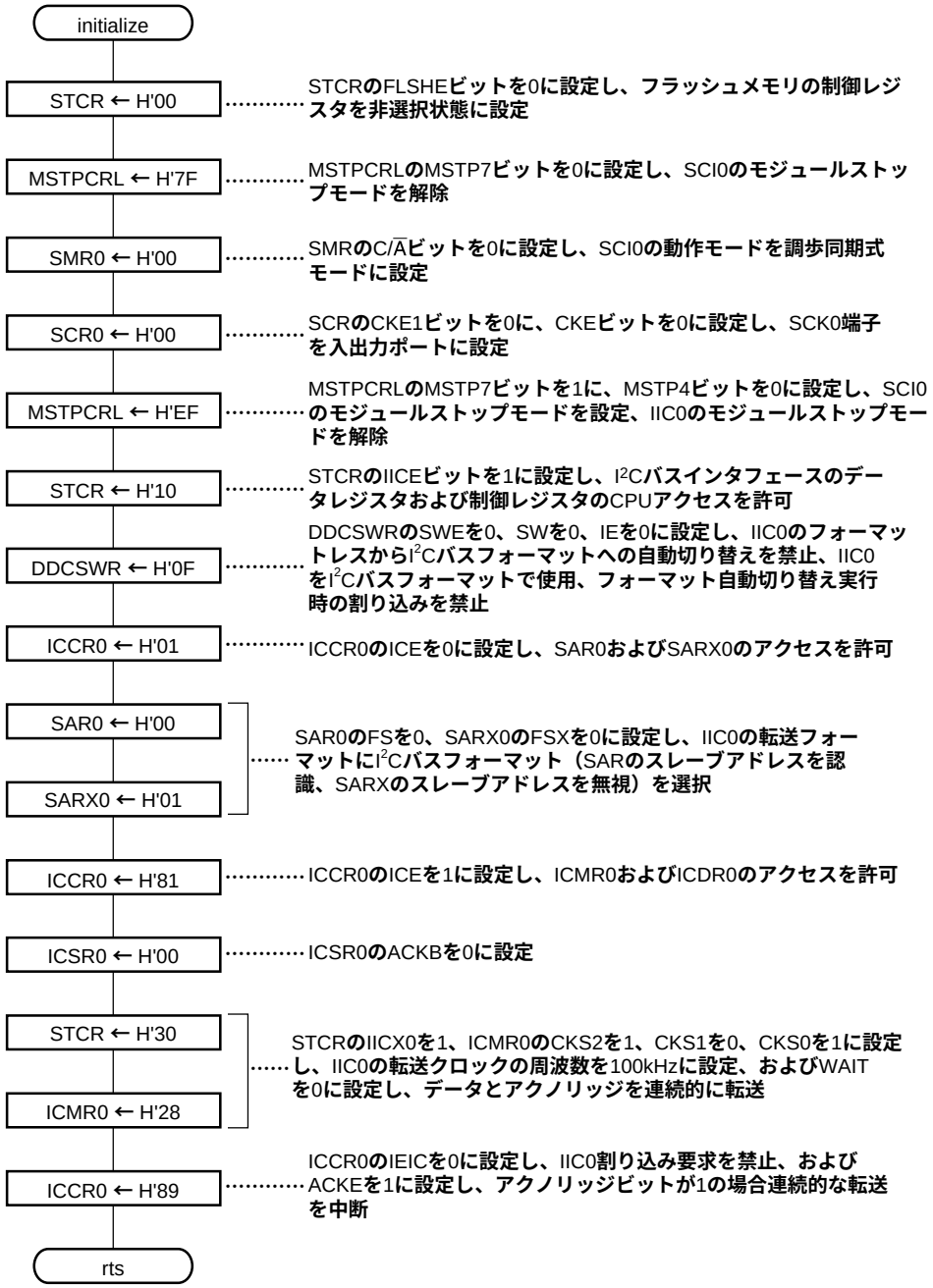
本タスク例では、変数以外の RAM は使用しません。

4.2.4 フローチャート

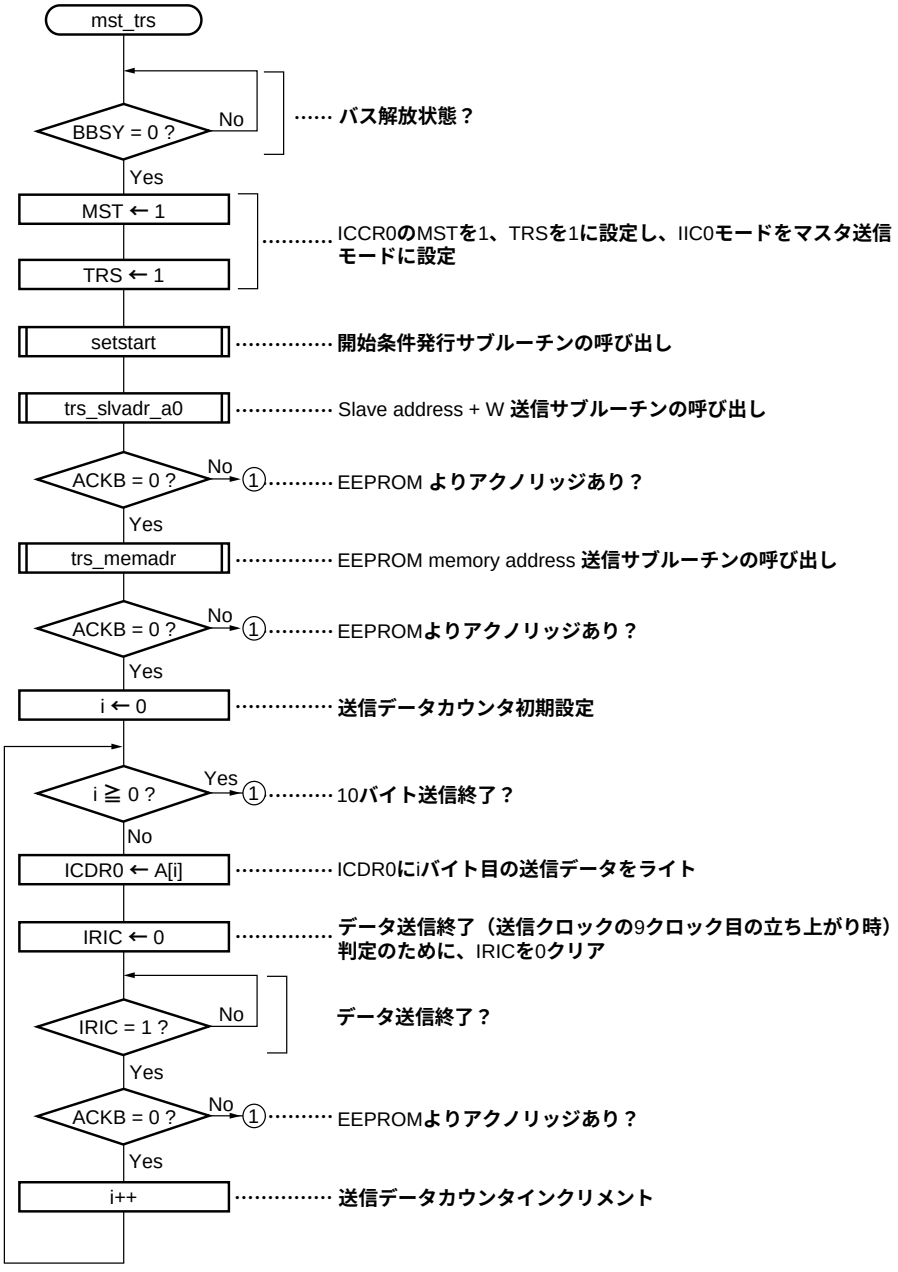
（1） メインルーチン

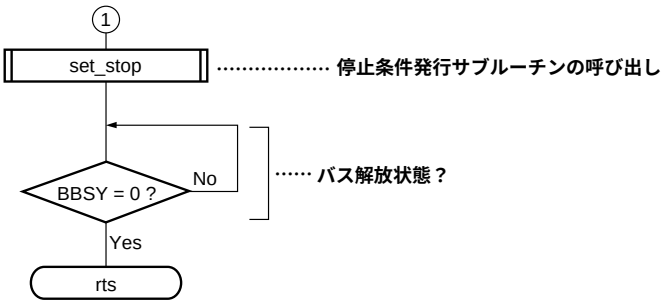


（2） 初期設定サブルーチン

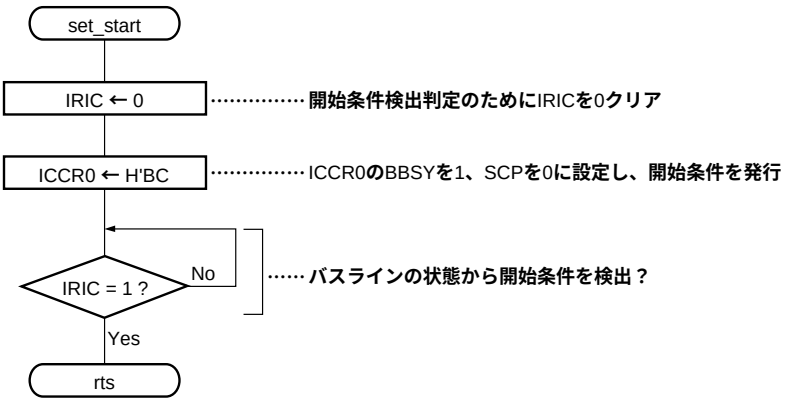


(3) シングルマスタ送信サブルーチン

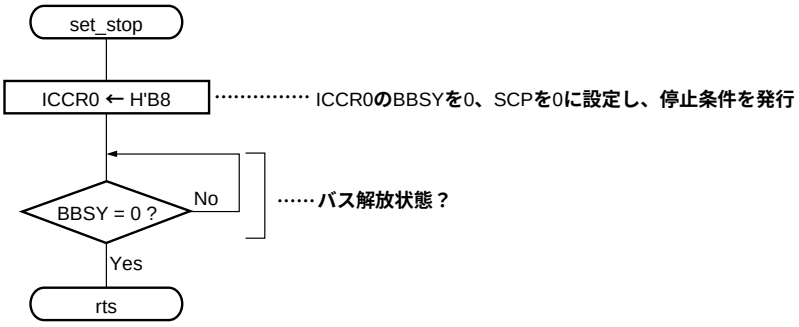




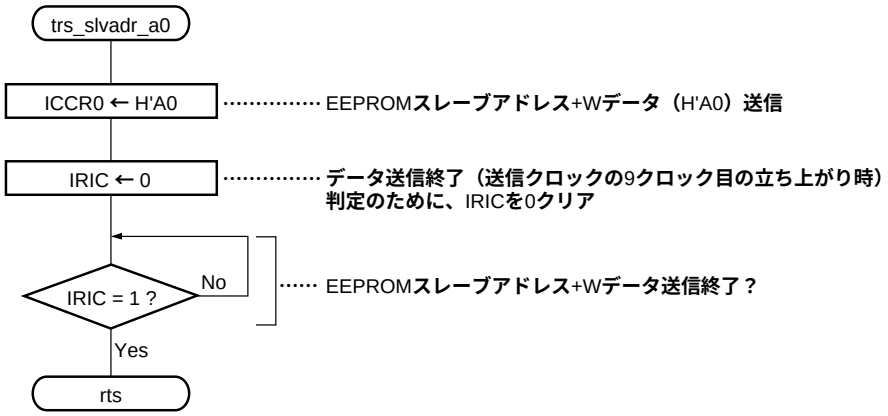
(4) 開始条件発行サブルーチン



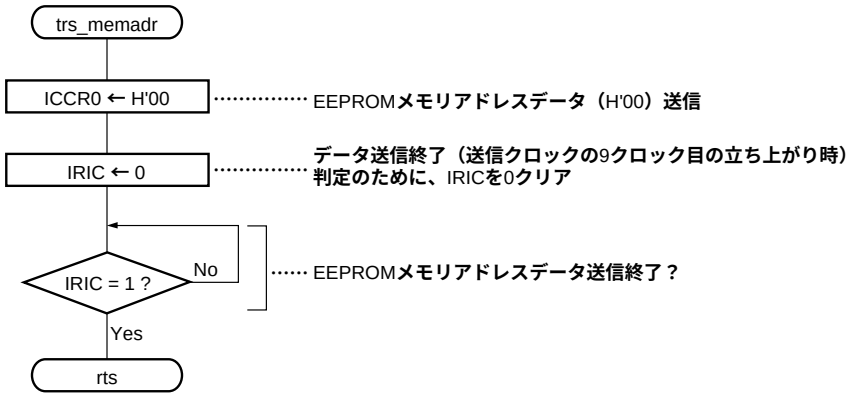
(5) 停止条件発行サブルーチン



（6） Slave address + W サブルーチン



（7） EEPROM memory address 送信サブルーチン



4.2.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*       1.Single master transmit to EEPROM        *
*                          File name   :  SMTxd.c  *
*                          Fai         :  20MHz    *
*                          Mode        :  3        *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                      *
************************************************/
void main(void);                                  /* Main routine */
void initialize(void);                            /* IIC0 initialize */
void mst_trs(void);                               /* Matser transmit to EEPROM */
void set_start(void);                             /* Start condition set */
void set_stop(void);                              /* Stop condition set */
void trs_slvadr_a0(void);                         /* Slave address + W data transmit */
void trs_memadr(void);                            /* EEPROM memory address data transmit */

/************************************************
*  Data table                                     *
************************************************/
const unsigned char dt_trs[10] =                  /* Transmit data (10 byte) */
{
    0x01,                                         /*  1st transmit data */
    0x02,                                         /*  2ns transmit data */
    0x03,                                         /*  3rd tranmist data */
    0x04,                                         /*  4th tranmist data */
    0x05,                                         /*  5th tranmist data */
    0x06,                                         /*  6th tranmist data */
    0x07,                                         /*  7th tranmist data */
    0x08,                                         /*  8th tranmist data */
```

（続く）

```
    0x09,                                   /*  9th tranmist data */
    0x0a                                    /* 10th tranmist data */
};

/************************************************
*  main : Main routine                         *
************************************************/
void main(void)
#pragma asm
         mov.l   #h'f000,sp                 ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                      /* MCU mode set */

    SYSCR.BYTE = 0x09;                      /* Interrupt control mode set */
    initialize();                           /* Initialize */
    set_imask_ccr(0);                       /* Interrupt enable */
    mst_trs();                              /* Master transmit to EPROM */
    while(1);                               /* End */
}

/************************************************
*  initialize : IIC0 Initialize                *
************************************************/
void initialize(void)
{
    STCR.BYTE = 0x00;                       /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                   /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                   /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                   /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                       /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                     /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                  /* ICE = 0 */
    IIC0.SAR.BYTE = 0x00;                   /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                  /* FSX = 1 */
```

（続く）

```
    IIC0.ICCR.BYTE = 0x81;                          /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                          /* ACKB = 0 */
    STCR.BYTE = 0x30;                               /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                          /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                          /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  mst_trs : Master transmit to EEPROM           *
************************************************/
void mst_trs(void)
{
   unsigned char i;                                 /* Tranmit data counter */

   while(IIC0.ICCR.BIT.BBSY == 1);                  /* Bus empty (BBSY=0) ? */

   IIC0.ICCR.BIT.MST = 1;                           /* Master transmit mode set */
   IIC0.ICCR.BIT.TRS = 1;                           /* MST = 1, TRS = 1 */
   set_start();                                     /* Start condition set */
   trs_slvadr_a0();                                 /* Slave address + W data transmit */

   if(IIC0.ICSR.BIT.ACKB == 0)
   {
       trs_memadr();                                /* EEPROM memory address data transmit */

       if(IIC0.ICSR.BIT.ACKB == 0)
       {
           for(i=0; i<10; i++)
           {
               IIC0.ICDR = dt_trs[i];               /* Transmit data write */
               IIC0.ICCR.BIT.IRIC = 0;              /* IRIC = 0 */
               while(IIC0.ICCR.BIT.IRIC == 0);      /* Transmit end (IRIC=1) ? */
               if(IIC0.ICSR.BIT.ACKB == 1)          /* ACKB = 0 ? */
               {
                   break;                           /* ACKB = 1 */
                }
           }
```

（続く）

```
        }
    }
    set_stop();                                 /* Stop condition set */
}

/************************************************
*  set_start : Start condition set              *
************************************************/
void set_start(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                      /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Start condition set (IRIC=1) ? */
}

/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
    IIC0.ICCR.BYTE = 0xb8;                      /* Stop condition set (BBSY=0.SCP=0) */
    while(IIC0.ICCR.BIT.BBSY == 1);             /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
{
    IIC0.ICDR = 0xa0;                           /* Slave address + W data(H'A0) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}
```

（続く）

```
/*************************************************
*  trs_memadr : EEPROM memory address data transmit*
*************************************************/
void trs_memadr(void)
{
    IIC0.ICDR = 0x00;                           /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}
```

4.3 シングルマスタ受信

4.3.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）から 10 バイトのデータを読み込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地から H'09 番地のデータを読み込みます。
- 読み込むデータは RAM の H'E100 番地から H'E1009 番地に格納します。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.7 に H8S/2138 と EEPROM の接続例を示す。

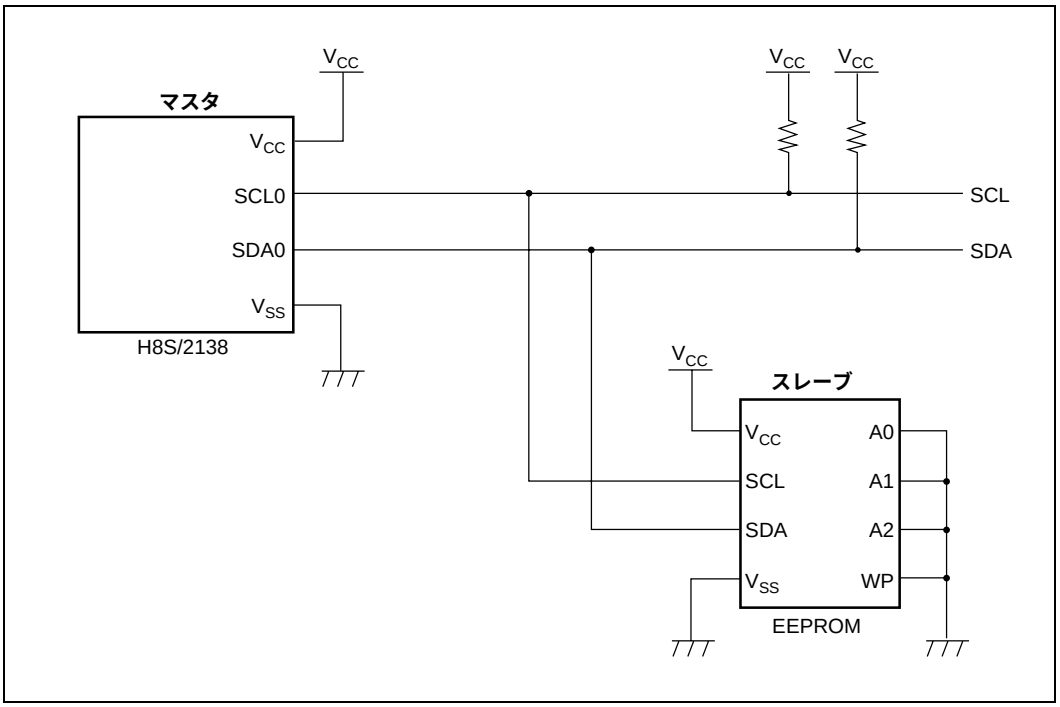


図 4.7 H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.8 に示します。

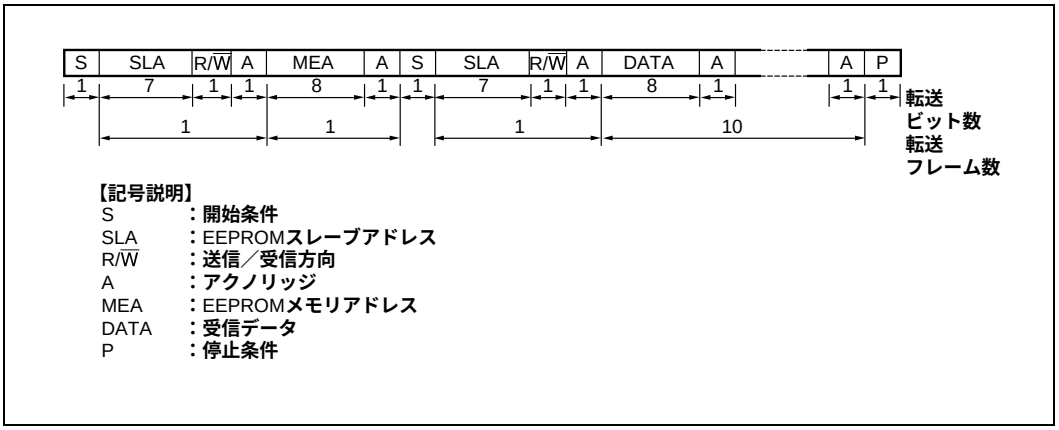
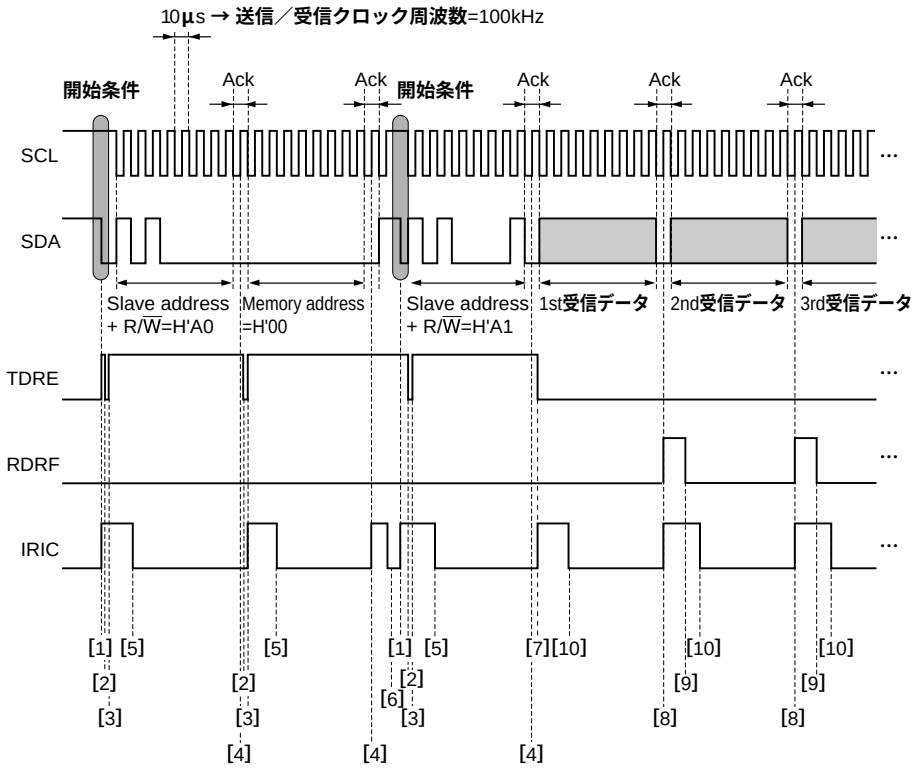


図 4.8 **本タスク例で使用する転送フォーマット**

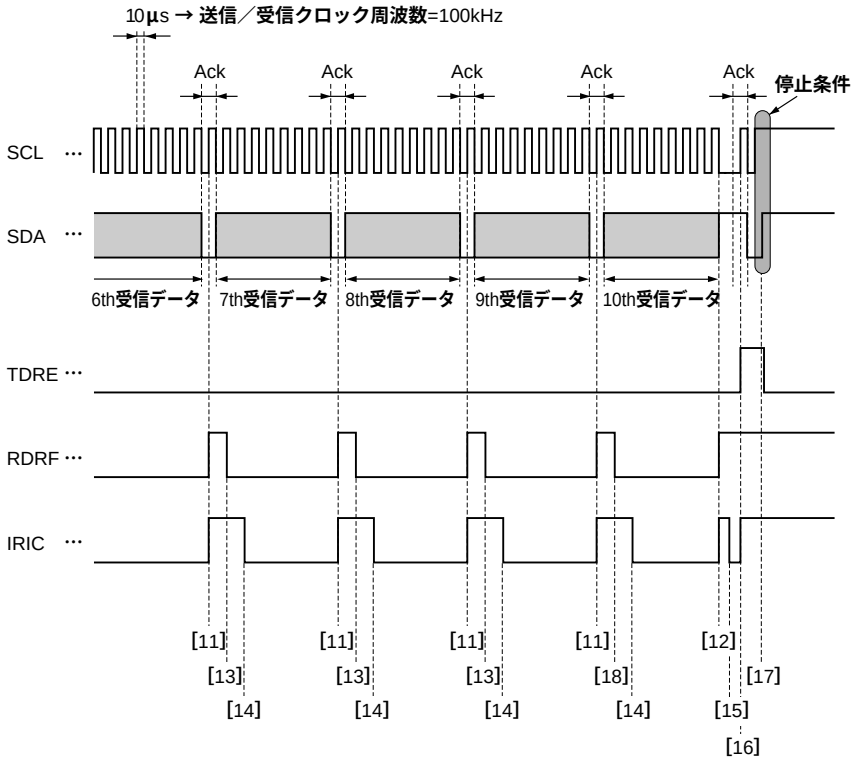
4.3.2 動作説明

図 4.9 および**図** 4.10 に動作原理を示します。



	ソフトウェア処理	ハードウェア処理
[1]	開始条件の発行（BBSY=1、SCP=0）	IRIC=1、TDRE=1（バスラインの状態から開始条件を検出）
[2]	ICDR0に送信データをライト	TDRE=0（TRS=1の状態でICDRTにデータをライト）
[3]	処理なし	TDRE=1（ICDRTからICDRSにデータを転送）
[4]	処理なし	IRIC=1（データ送信の終了（送信クロックの9クロック目の立ち上がり時））
[5]	送信終了判定のためにIRICを0にクリア	処理なし
[6]	開始条件検出判定のためにIRICを0にクリア	処理なし
[7]	マスタ受信モードに設定（MST=1、TRS=0）	TDRE=0（TRS=0のとき）
[8]	処理なし	IRIC=1（データ受信の終了（受信クロックの9クロック目の立ち上がり時））
[9]	ICDR0から受信データをリード	RDRF=0（受信モードでICDRRの受信データをリード）
[10]	受信終了判定のためにIRICを0にクリア	処理なし

図 4.9　シングルマスタ受信動作原理（1）



	ソフトウェア処理	ハードウェア処理
[11]	処理なし	IRIC=1、RDRF=1(WAIT=0)（データ受信の終了（受信クロックの9クロック目の立ち上がり時））
[12]	処理なし	IRIC=1、RDRF=1(WAIT=1)（データ受信の終了（受信クロックの8クロック目の立ち下がり時））
[13]	ICDR0から受信データをリード	RDRF=0（受信モードでICDRRの受信データをリード）
[14]	受信終了判定のためにIRICを0にクリア	処理なし
[15]	受信クロックの9クロック目の出力終了判定のためにIRICを0にクリア	受信クロックの9クロック目の出力開始
[16]	マスタ送信モードに設定（MST=1、TRS=0）	IRIC=1（受信クロックの9クロック目の立ち上がり時） TDRE=1（開始条件検出後、TRS=0からTRS=1に切り替えたとき）
[17]	停止条件の発行（BBSY=0、SCP=0）	TDRE=0（停止条件の発行後、バスラインの状態から停止条件を検出）

図 4.10　シングルマスタ受信動作原理（2）

4.3.3 ソフトウェア説明

(1) モジュール説明

表 4.4 に本タスク例におけるモジュール説明を示します。

表 4.4 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	intialize	使用 RAM 領域、および IIC0 の初期設定
シングルマスタ送信	mst_rec	シングルマスタ送信による EEPROM への 10 バイトのデータの送信
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス+W データ（H'A0）の送信
Slave address + R 送信	trs_slvadr_a1	EEPROM のスレーブアドレス+R データ（H'A1）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信
データ受信	rec_data	10 バイトデータの受信

(2) 使用内蔵レジスタ説明

表 4.5 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.5 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		受信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットとともに、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットとともに、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジの間にウェイトを挿入するか否かを設定	H'FFDF bit6	0/1
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I²C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I²C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1

（続く）

表 4.5 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I^2C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I^2C バスインタフェースの送信／受信モードの設定	H'FFD8 bit4	1/0
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY と組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM から受信したアクノリッジを格納 受信時、EEPROM へ送信するアクノリッジを設定	H'FFD9 bit0	−
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1

(3) 変数説明

表 4.6 に本タスク例における変数説明を示します。

表 4.6 変数説明

変数	機能	データ長	初期値	使用モジュール名
dummy	MDCR リード値	1 バイト	−	Main
i	受信データカウンタ	1 バイト	H'00	initialize rec_data

(4) 使用 RAM 説明

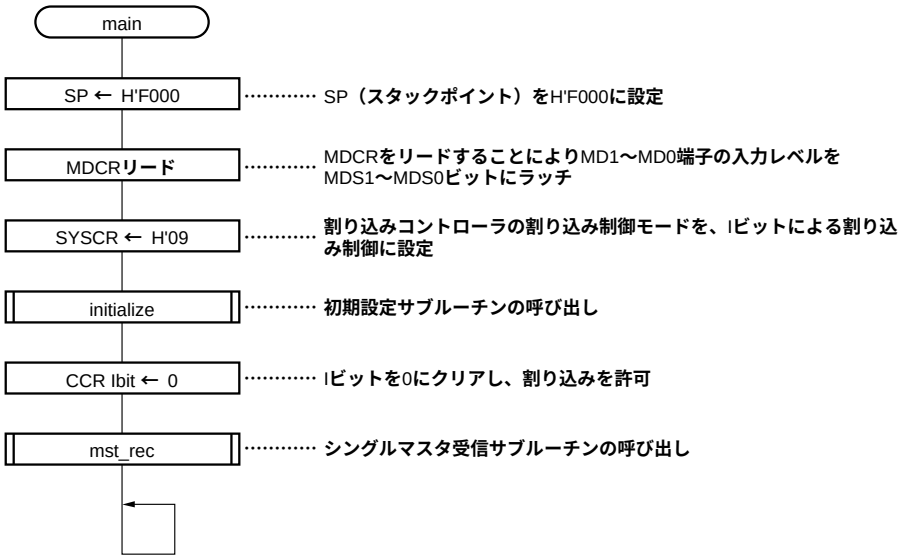
表 4.7 に本タスク例における使用 RAM 説明を示します。

表 4.7 使用 RAM 説明

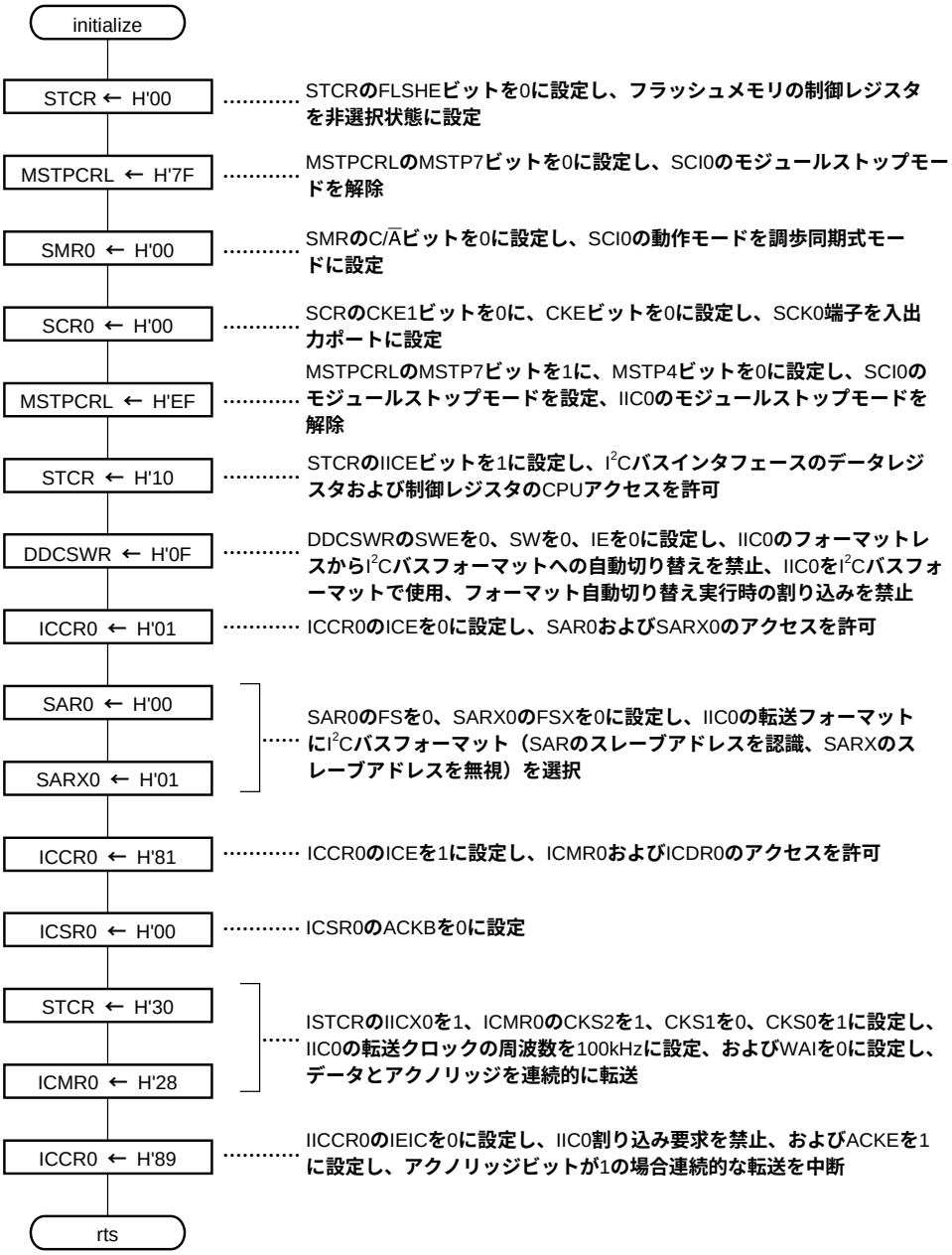
ラベル	機能	データ長	アドレス	使用モジュール名
dt_rec[i]	受信データを格納	10 バイト	H'E100 to H'E109	initialize rec_data

4.3.4 フローチャート

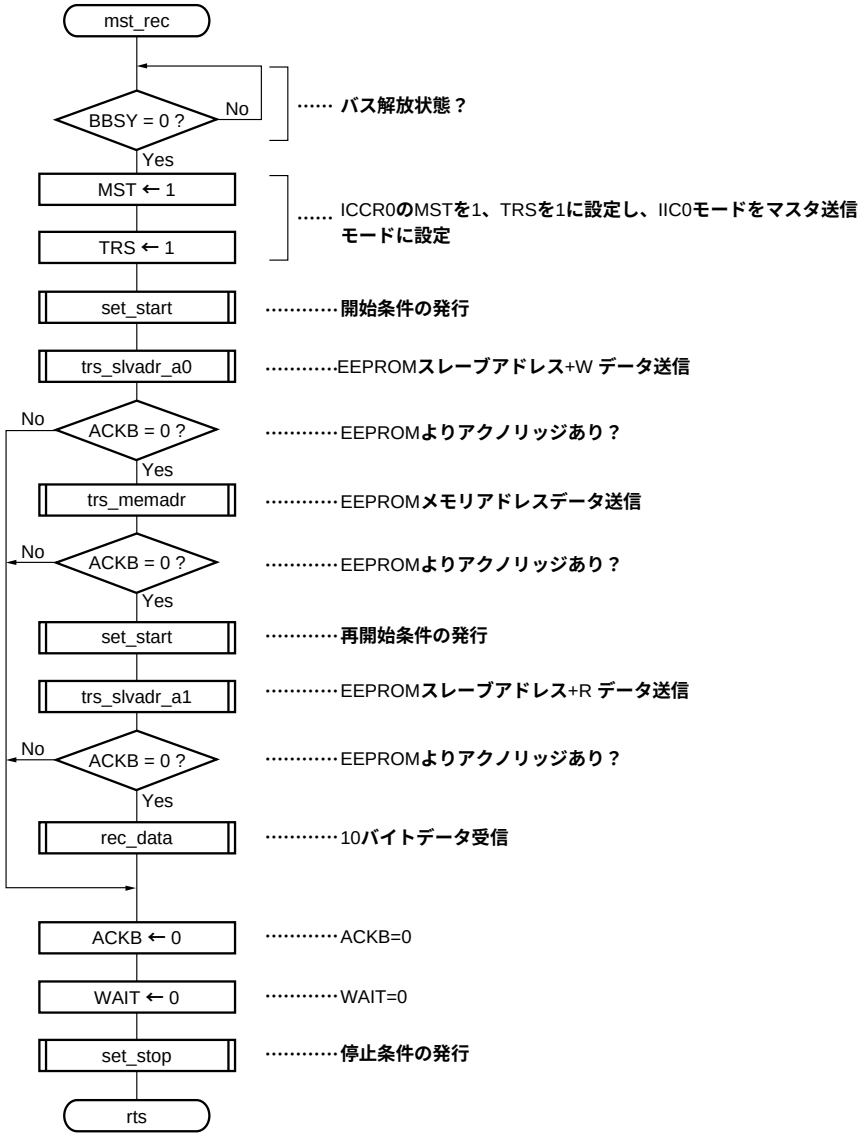
（1） メインルーチン



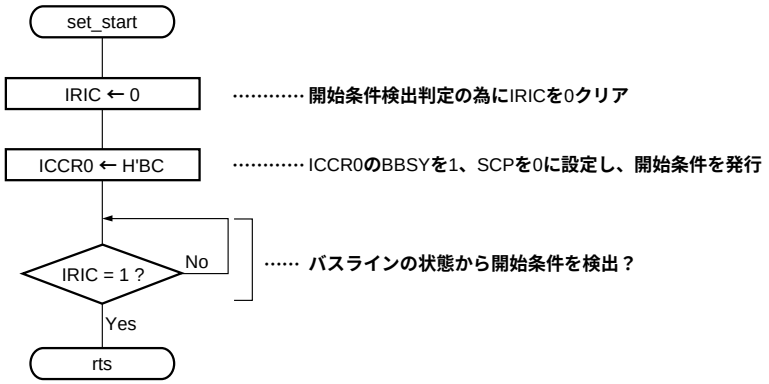
(2) 初期設定サブルーチン



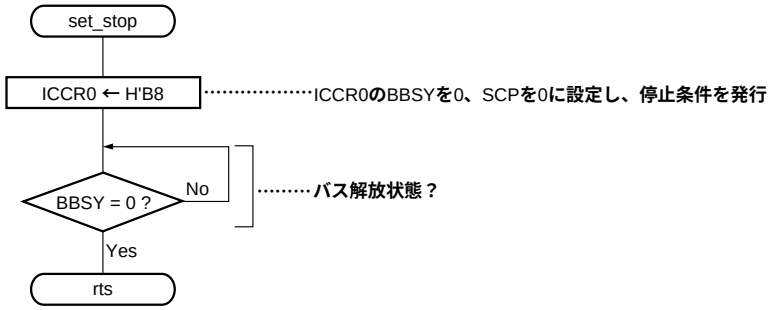
(3) シングルマスタ受信サブルーチン



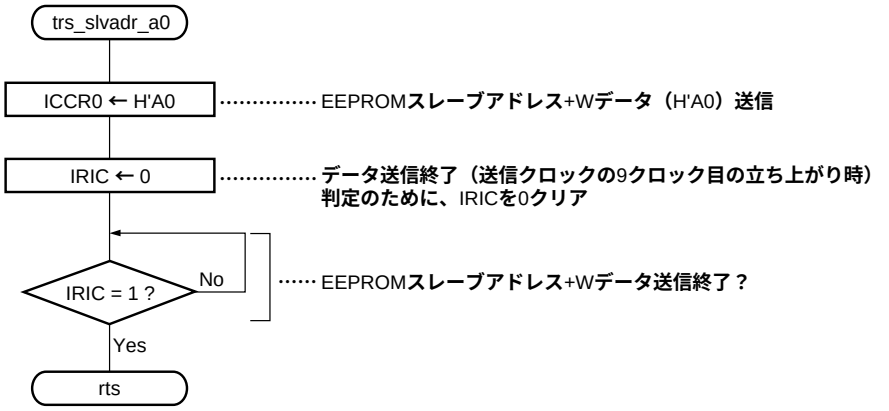
(4) 開始条件発行サブルーチン



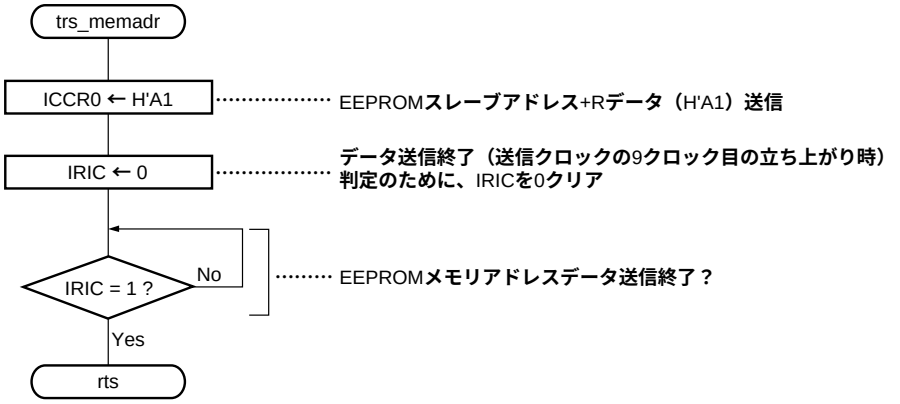
(5) 停止条件発行サブルーチン



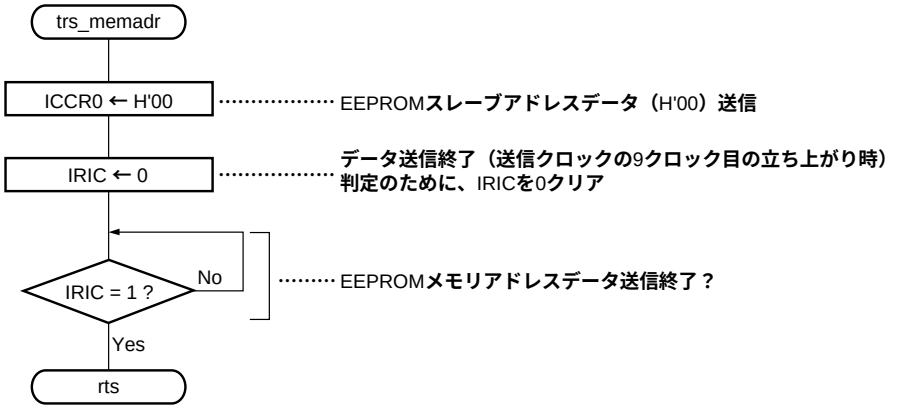
（6） Slave address + W 送信サブルーチン



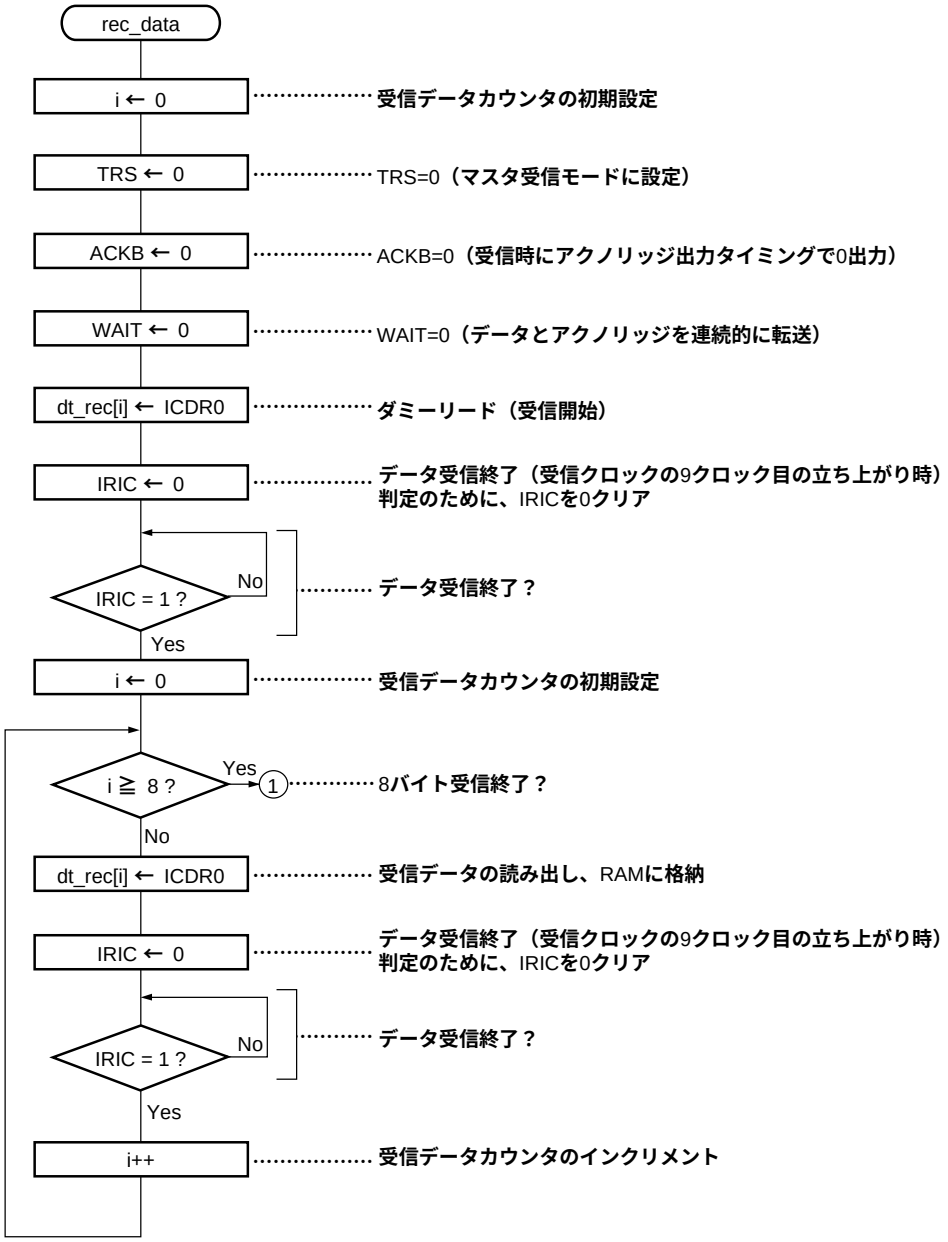
（7） Slave address + R 送信サブルーチン

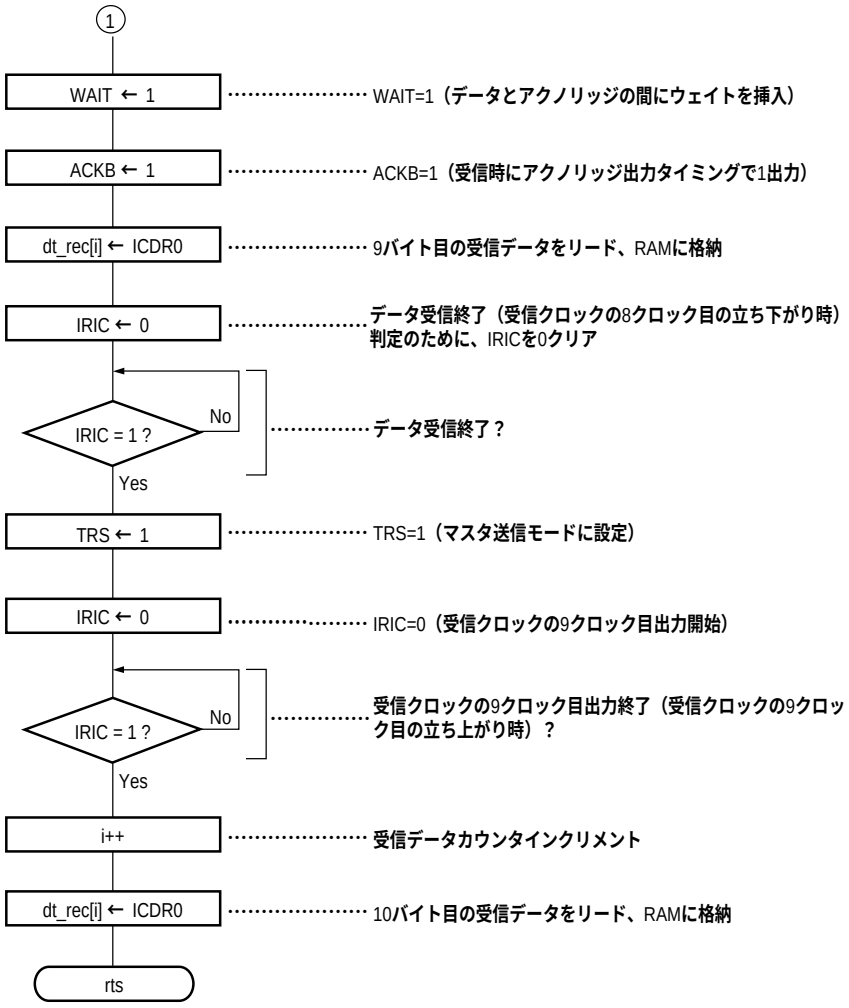


（8） EEPROM memory address 送信サブルーチン



(9) データ受信サブルーチン




1
WAIT ← 1
WAIT=1（データとアクノリッジの間にウェイトを挿入）
ACKB ← 1
ACKB=1（受信時にアクノリッジ出力タイミングで1出力）
dt_rec[i] ← ICDR0
9バイト目の受信データをリード、RAMに格納
IRIC ← 0
データ受信終了（受信クロックの8クロック目の立ち下がり時）
判定のために、IRICを0クリア
IRIC = 1 ?
No
データ受信終了？
Yes
TRS ← 1
TRS=1（マスタ送信モードに設定）
IRIC ← 0
IRIC=0（受信クロックの9クロック目出力開始）
IRIC = 1 ?
No
受信クロックの9クロック目出力終了（受信クロックの9クロック目の立ち上がり時）？
Yes
i++
受信データカウンタインクリメント
dt_rec[i] ← ICDR0
10バイト目の受信データをリード、RAMに格納
rts

4.3.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*       2.Single master receive from EEPROM       *
*                           File name   :  SMRxd.c *
*                           Fai         :  20MHz   *
*                           Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* RAM & IIC0 initialize */
void mst_rec(void);                              /* Master receive from EEPROM */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_slvadr_a1(void);                        /* Slave address + R data transmit */
void trs_memadr(void);                           /* EEPROM memory address data transmit */
void rec_data(void);                             /* 10-byte data receive */

/************************************************
*  RAM allocation                                *
************************************************/
#pragma section ramarea
unsigned char dt_rec[10];                        /* Receive data store area */

/************************************************
*  main : Main routine                           *
************************************************/
#pragma section
void main(void)
```

（続く）

```
#pragma asm
        mov.l   #h'f000,sp                      /*Stack pointer initialize */
#pragma endasm
{
   unsigned char dummy;
   dummy = MDCR.BYTE;                           /* MCU mode set */

   SYSCR.BYTE = 0x09;                           /* Interrupt control mode set */
   initialize();                                /* Initialize */
   set_imask_ccr(0);                            /* Interrupt enable */
   mst_rec();                                   /* Master receive from EPROM */
   while(1);                                    /* End */
}

/************************************************
*  initialize : RAM & IIC0 Initialize          *
************************************************/
void initialize(void)
{
   unsigned char i=0;

   for(i=0; i<10; i++)                          /* Receive data store area initialize */
   {
       dt_rec[i]  = 0x00;
   }
                                                /* IIC0 module initialize */
   STCR.BYTE = 0x00;                            /* FLSHE = 0 */
   MSTPCR.BYTE.L = 0x7f;                        /* SCI0 module stop mode reset */
   SCI0.SMR.BYTE = 0x00;                        /* SCL0 pin function set */
   SCI0.SCR.BYTE = 0x00;
   MSTPCR.BYTE.L = 0xef;                        /* IIC0 module stop mode reset */
   STCR.BYTE = 0x10;                            /* IICE = 1 */
   DDCSWR.BYTE = 0x0f;                          /* IIC bus format initialize */
   IIC0.ICCR.BYTE = 0x01;                       /* ICE = 0 */
   IIC0.SAR.BYTE = 0x00;                        /* FS = 0 */
   IIC0.SARX.BYTE = 0x01;                       /* FSX = 1 */
   IIC0.ICCR.BYTE = 0x81;                       /* ICE = 1 */
```

（続く）

```
    IIC0.ICSR.BYTE = 0x00;                          /* ACKB = 0 */
    STCR.BYTE = 0x30;                               /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                          /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                          /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  mst_rec : Master receive from EEPROM          *
************************************************/
void mst_rec(void)
{
    while(IIC0.ICCR.BIT.BBSY == 1);                 /* Bus empty (BBSY=0) ? */
    IIC0.ICCR.BIT.MST = 1;                          /* Master transmit mode set */
    IIC0.ICCR.BIT.TRS = 1;                          /* MST = 1, TRS = 1 */
    set_start();                                    /* Start condition set */
    trs_slvadr_a0();                                /* EEPROM slave address + W data transmit */

    if(IIC0.ICSR.BIT.ACKB == 0)                     /* ACKB = 0 ? */
    {
        trs_memadr();                               /* EEPROM memory address data transmit */

        if(IIC0.ICSR.BIT.ACKB == 0)                 /* ACKB = 0 ? */
        {
            set_start();                            /* Re-start condition set */
            trs_slvadr_a1();                        /* EEPROM slave address + R data transmit */

            if(IIC0.ICSR.BIT.ACKB == 0)             /* ACKB = 0 ? */
            {
                rec_data();                         /* Data recieve */
            }
        }
    }
    set_stop();
}
/************************************************
*  set_start : Start condition set               *
************************************************/
```

（続く）

```
void set_start(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                          /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Start condition set (IRIC=1) ? */
}

/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
    IIC0.ICCR.BYTE = 0xb8;                          /* Stop condition set (BBSY=0,SCP=0) */
    while(IIC0.ICCR.BIT.BBSY == 1);                 /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
{
    IIC0.ICDR = 0xa0;                               /* Slave address + W data(H'A0) write */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_slvadr_a1 : Slave address + R data transmit*
************************************************/
void trs_slvadr_a1(void)
{
    IIC0.ICDR = 0xa1;                               /* Slave address + R data(H'A1) write */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Transmit end (IRIC=1) ? */
}
```

（続く）

```
/************************************************
*  trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
    IIC0.ICDR = 0x00;                           /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}

/************************************************
*  rec_data : 10-byte data receive             *
************************************************/
void rec_data(void)
{
    unsigned char i=0;                          /* Receive data counter initialize */

    IIC0.ICCR.BIT.TRS = 0;                      /* Master transmit mode set(MST=1,TRS=0) */
    IIC0.ICSR.BIT.ACKB = 0;                     /* ACKB = 0 */
    IIC0.ICMR.BIT.WAIT = 0;                     /* WAIT = 0 */

    dt_rec[i] = IIC0.ICDR;                      /* Dummy read */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* receive end (IRIC=1) ? */

    for(i=0; i<8; i++)                          /* 1st to 8th data receive */
    {
        dt_rec[i] = IIC0.ICDR;                  /* Receive data read */
        IIC0.ICCR.BIT.IRIC = 0;                 /* IRIC = 0 */
        while(IIC0.ICCR.BIT.IRIC == 0);         /* Receive end ? */
    }

    IIC0.ICMR.BIT.WAIT = 1;                     /* WAIT = 1 */
    IIC0.ICSR.BIT.ACKB = 1;                     /* ACKB = 1 */
    dt_rec[i] = IIC0.ICDR;                      /* 9th receive data read */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Receive end (IRIC=1) ? */
```

（続く）

```
    IIC0.ICCR.BIT.TRS = 1;                          /* Master transmit mode set (MST=1,TRS=1) */
    IIC0.ICCR.BIT.IRIC = 0;                         /* 9th clock transmit (IRIC=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* 9th clock transmit end (IRIC=1) ? */

    dt_rec[++i] = IIC0.ICDR;                        /* 10th (last) receive data read */

    IIC0.ICSR.BIT.ACKB = 0;                         /* ACKB = 0 */
    IIC0.ICMR.BIT.WAIT = 0;                         /* WAIT = 0 */
}
```

4.4 シングルマスタ送信による１バイトデータの送信

4.4.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）に１バイトのデータを書き込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地にデータを書き込みます。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.11 に H8S/2138 と EEPROM の接続例を示す。

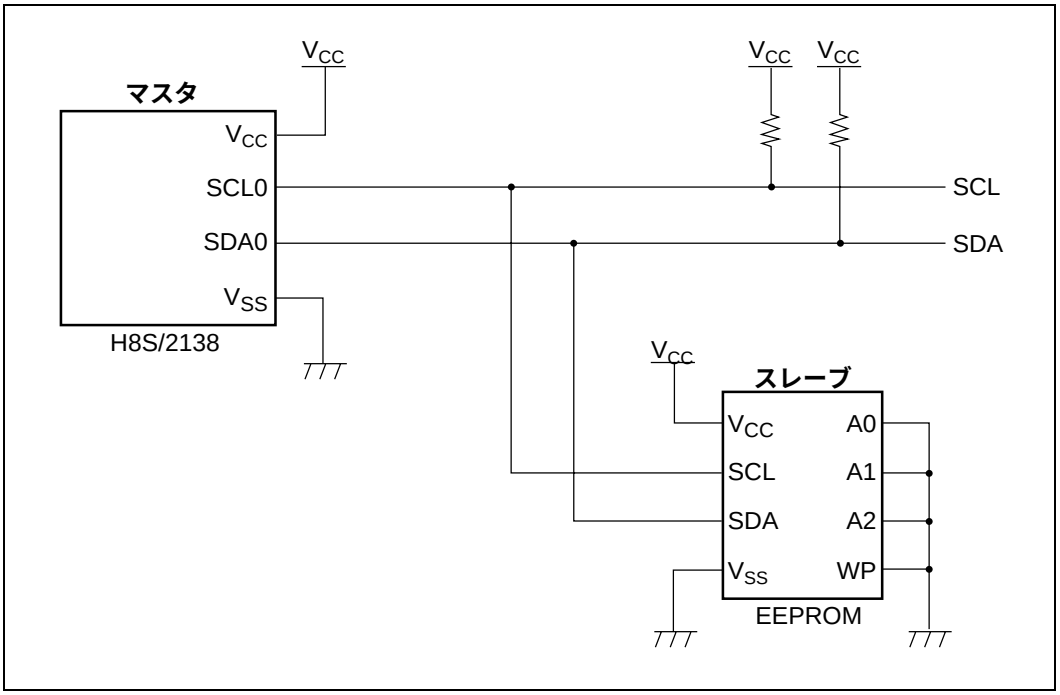


図 4.11 H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.12 に示します。

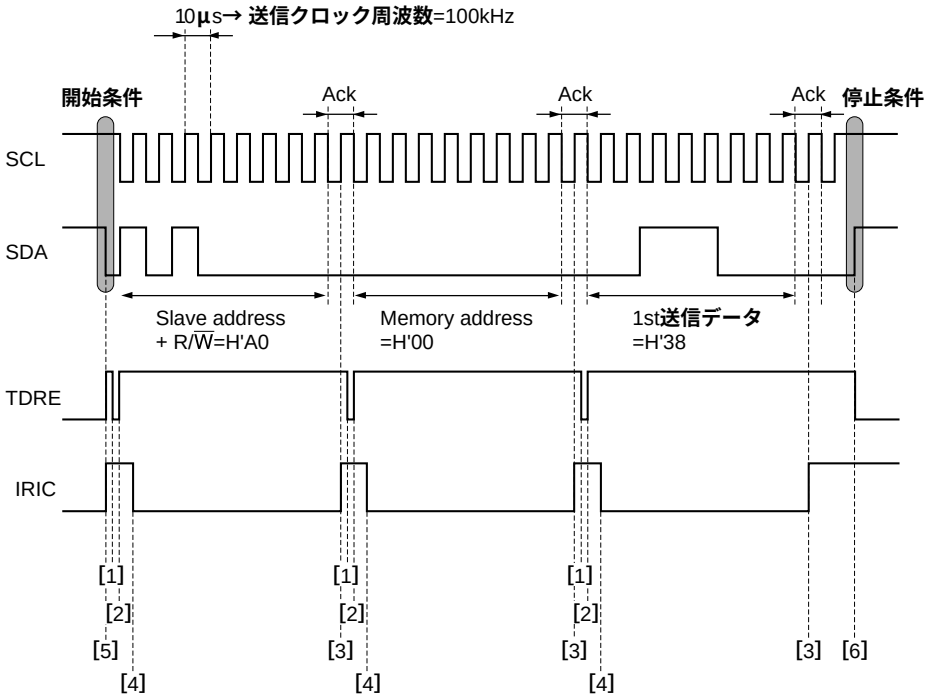
【記号説明】

S	**：開始条件**
SLA	**：**EEPROM**スレーブアドレス**
R/$\overline{W}$	**：送信／受信方向**
A	**：アクノリッジ**
MEA	**：**EEPROM**メモリアドレス**
DATA	**：送信データ**
P	**：停止条件**

図 4.12　**本タスク例で使用する転送フォーマット**

4.4.2 動作説明

図 4.13 に動作原理を示します。



	ソフトウェア処理	ハードウェア処理
[1]	ICDR0に送信データをライト	TDRE=0（TRS=1の状態でICDRTにデータをライト）
[2]	処理なし	TDRE=1（ICDRTからICDRSにデータを転送）
[3]	処理なし	IRIC=1（データ送信の終了（送信クロックの9クロック目の立ち上がり時））
[4]	送信終了判定のためにIRICを0にクリア	処理なし
[5]	開始条件の発行（BBSY=1、SCP=0）	IRIC=1、TDRE=1（バスラインの状態から開始条件を検出）
[6]	停止条件の発行（BBSY=0、SCP=0）	TDRE=0（バスラインの状態から停止条件を検出）

図 4.13　シングルマスタ送信による１バイトデータの送信動作原理

4.4.3 ソフトウェア説明

(1) モジュール説明

表 4.8 に本タスク例におけるモジュール説明を示します。

表 4.8 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	Intialize	IICO の初期設定
シングルマスタ送信	mst_trs	シングルマスタ送信による EEPROM への 1 バイトのデータの送信
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス+W データ（H'A0）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信

(2) 使用内蔵レジスタ説明

表 4.9 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.9 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジの連続的な転送を設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMR0,ICDR0/SAR,SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I^2C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1

（続く）

表 4.9 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	TRS	I²C バスインタフェースを送信モードで使用	H'FFD8 bit4	1/0
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY と組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	EEPROM から送信されたアクノリッジデータを格納	H'FFD9 bit0	−
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I²C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1,0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1

(3) 変数説明

表 4.10 に本タスク例における変数説明を示します。

表 4.10 変数説明

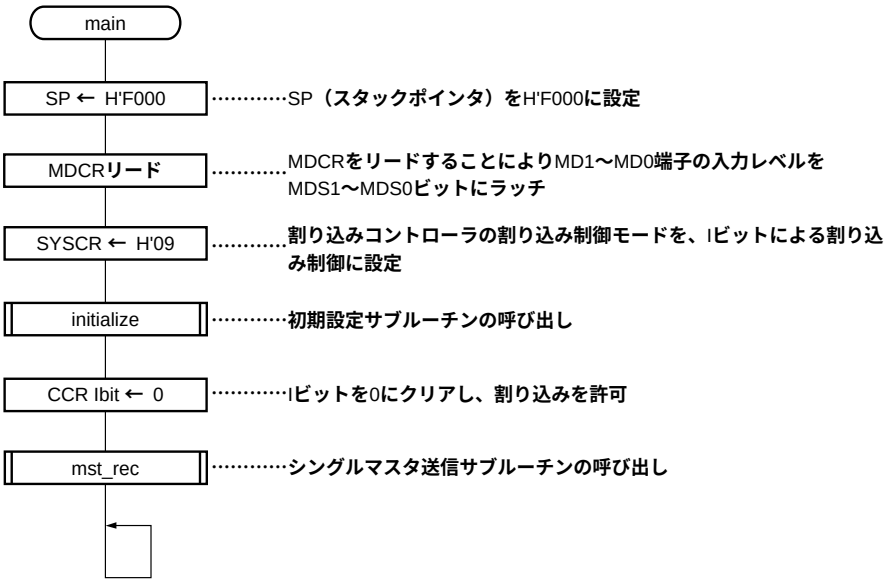
変数	機能	データ長	初期値	使用モジュール名
dt_trs	1バイト送信データ	1 バイト	H'38	mst_trs
dummy	MDCR リード値	1 バイト	−	main

（4） 使用 RAM 説明

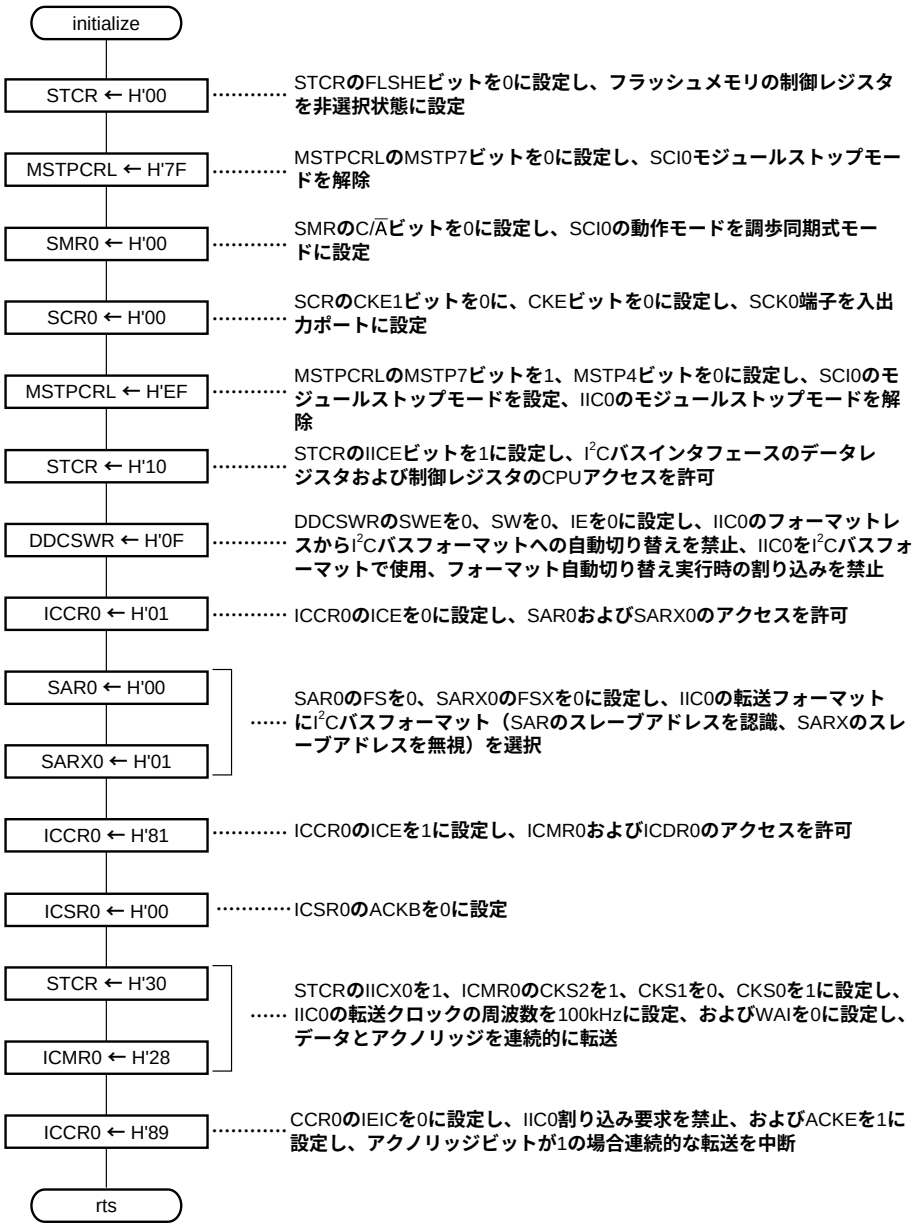
本タスク例では、変数以外の RAM は使用しません。

4.4.4 フローチャート

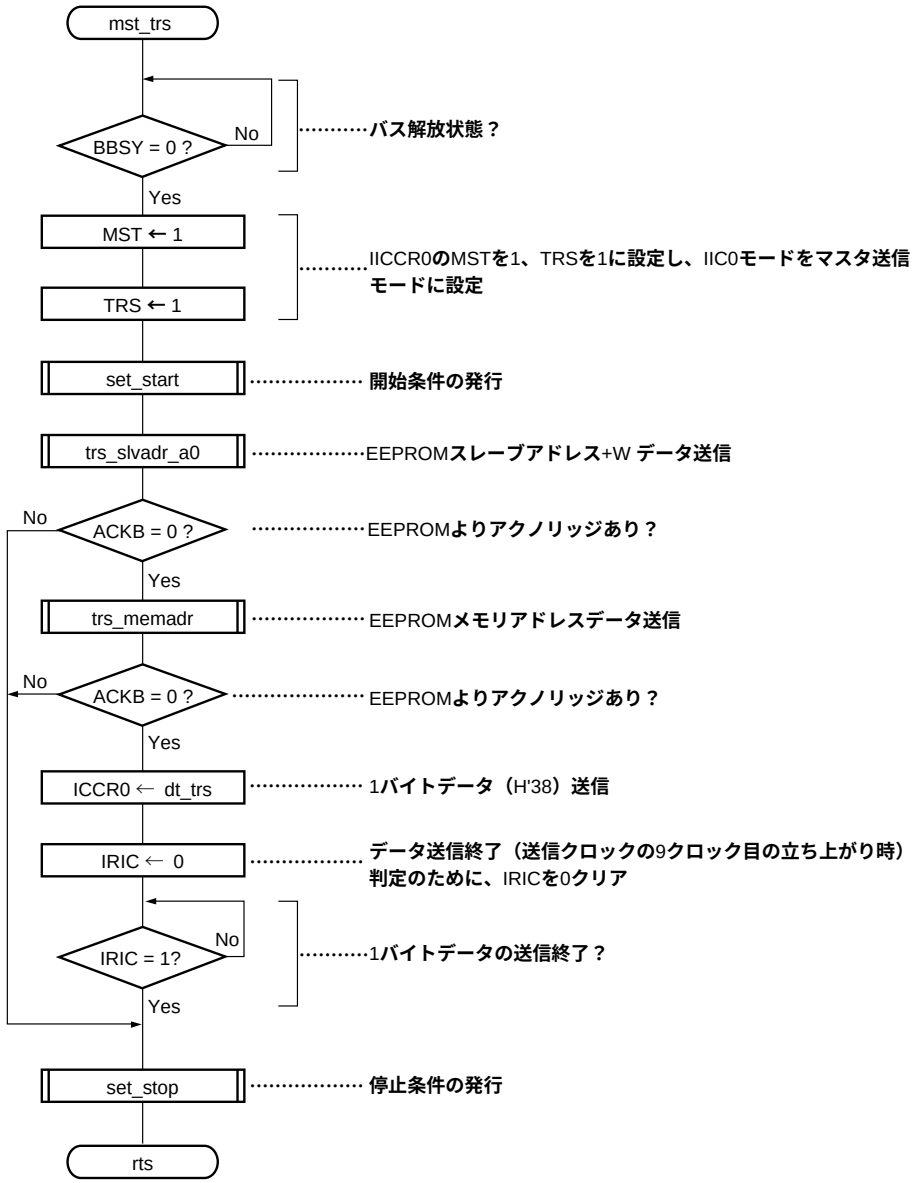
(1) メインルーチン



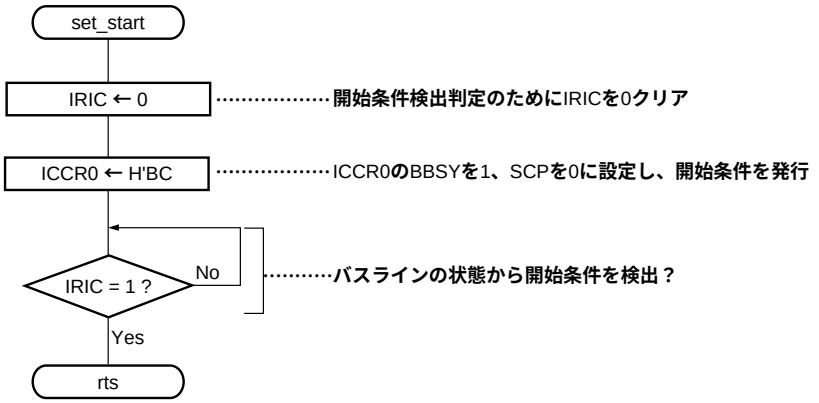
(2) 初期設定サブルーチン



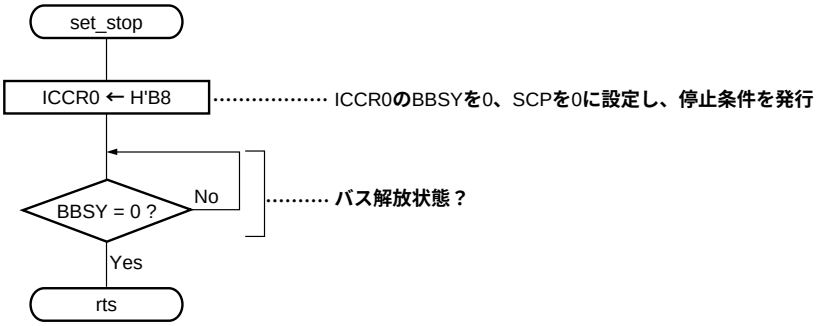
(3) シングルマスタ送信サブルーチン



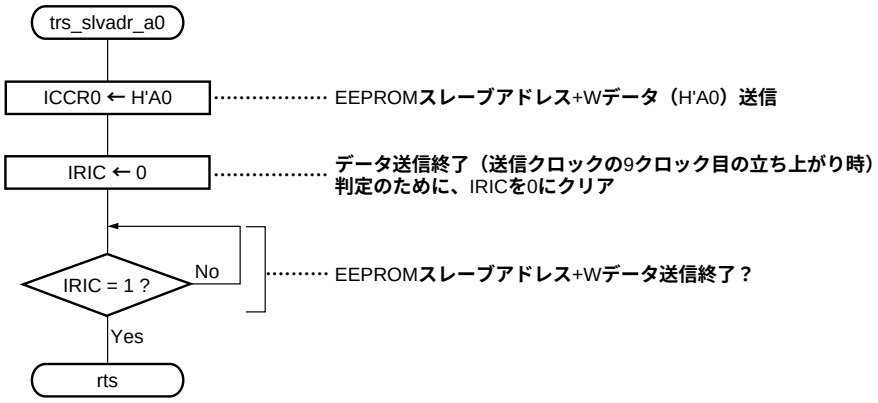
（4） 開始条件発行サブルーチン



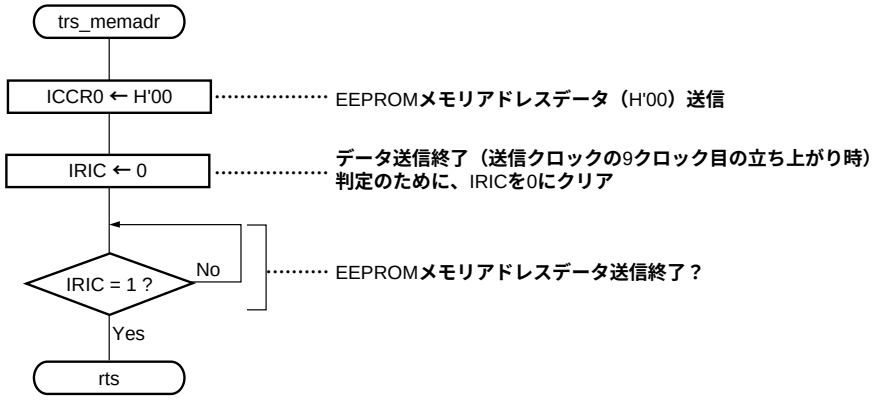
（5） 停止条件発行サブルーチン



（6） Slave address + W 送信サブルーチン



（7） EEPROM memory address 送信サブルーチン



4.4.5 プログラムリスト

```
/*****************************************************
*  H8S/2138 IIC bus application note                 *
*       3.Single master transmit 1byte data to EEPROM*
*                           File name   :  BYTxd.c    *
*                           Fai         :  20MHz      *
*                           Mode        :  3          *
*****************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/***********************************************
*  Prototype                                   *
***********************************************/
void main(void);                                /* Main routine */
void initialize(void);                          /* IIC0 initialize */
void mst_trs(void);                             /* Master transmit to EEPROM */
void set_start(void);                           /* Start condition set */
void set_stop(void);                            /* Stop condition set */
void trs_slvadr_a0(void);                       /* Slave address + W data transmit */
void trs_memadr(void);                          /* EEPROM memory address data transmit */

unsigned char dt_trs = 0x38;                    /* Transmit data (1byte) */

/***********************************************
*  main : Main routine                         *
***********************************************/
void main(void)
#pragma asm
        mov.l   #h'f000,sp                      ;Stack pointer initialize
#pragma endasm
{
   unsigned char dummy;
   dummy = MDCR.BYTE;                           /* MCU mode set */
```

（続く）

```
    SYSCR.BYTE = 0x09;                          /* Interrupt control mode set */
    initialize();                               /* Initialize */
    set_imask_ccr(0);                           /* Interrupt enable */
    mst_trs();                                  /* Master transmit to EPROM */
    while(1);                                   /* End */
}

/************************************************
*  initialize : IIC0 initialize                 *
************************************************/
void initialize(void)
{
    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x00;                       /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  mst_trs : Master transmit to EEPROM          *
************************************************/
void mst_trs(void)
{
    while(IIC0.ICCR.BIT.BBSY == 1);             /* Bus empty (BBSY=0) ? */
    IIC0.ICCR.BIT.MST = 1;                      /* Master transmit mode set */
```

（続く）

```
    IIC0.ICCR.BIT.TRS = 1;                          /* MST = 1, TRS = 1 */
    set_start();                                    /* Start condition set */
    trs_slvadr_a0();                                /* Slave address + W data transmit */

    if (IIC0.ICSR.BIT.ACKB == 0)
    {
        trs_memadr();                               /* EEPROM memory address data transmit */

        if (IIC0.ICSR.BIT.ACKB == 0)
        {
            IIC0.ICDR = dt_trs;                     /* 1 byte data transmit */
            IIC0.ICCR.BIT.IRIC = 0;                 /* IRIC = 0 */
            while(IIC0.ICCR.BIT.IRIC == 0);         /* transmit end (IRIC=1) ? */
        }
    }
    set_stop();                                     /* Stop condition set */
}
/************************************************
*  set_start : Start condition set                *
************************************************/
void set_start(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                          /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Start condition set (IRIC=1) ? */
}

/************************************************
*  set_stop : Stop condition set                  *
************************************************/
void set_stop(void)
{
    IIC0.ICCR.BYTE = 0xb8;                          /* Stop condition set (BBSY=0.SCP=0) */
    while(IIC0.ICCR.BIT.BBSY == 1);                 /* Bus empty (BBSY=0) ? */
}
```

（続く）

```
/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit  *
************************************************/
void trs_slvadr_a0(void)
{
    IIC0.ICDR = 0xa0;                           /* Slave address + W data(H'A0) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
    IIC0.ICDR = 0x00;                           /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}
```

4.5 シングルマスタ受信による 1 バイトデータの受信

4.5.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）から 1 バイトのデータを読み込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地のデータを読み込みます。
- 読み込むデータは RAM の H'E100 番地に格納します。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.14 に H8S/2138 と EEPROM の接続例を示します。

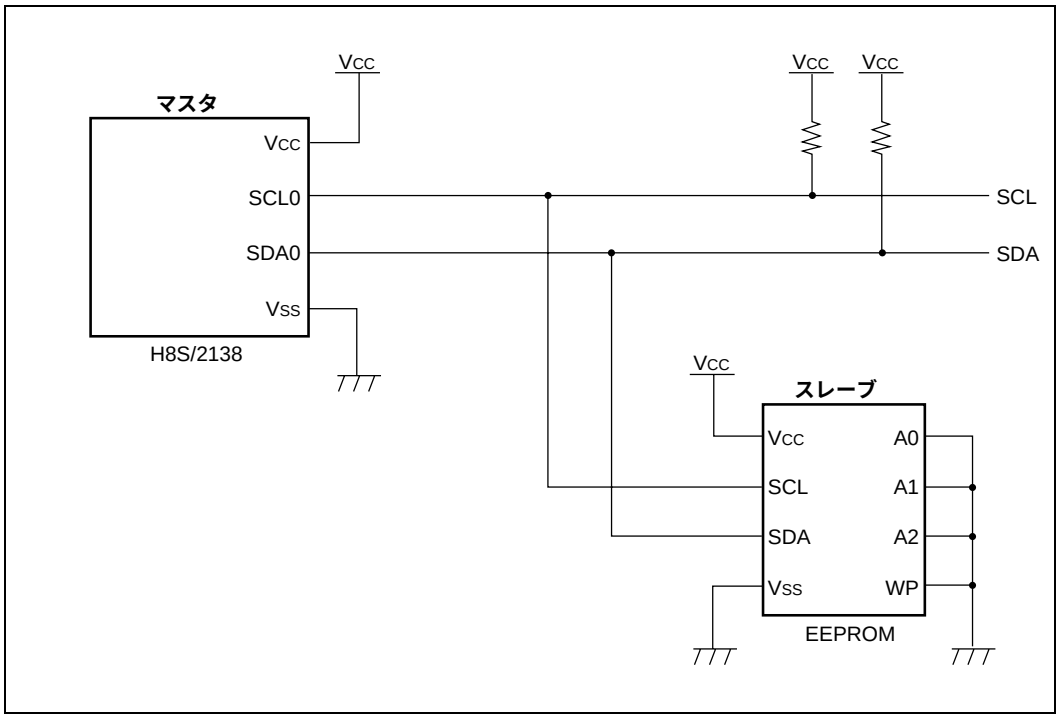
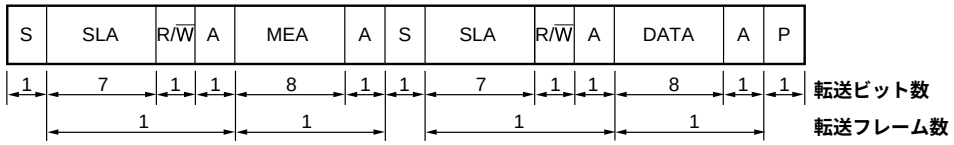


図 4.14　H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.15 に示します。



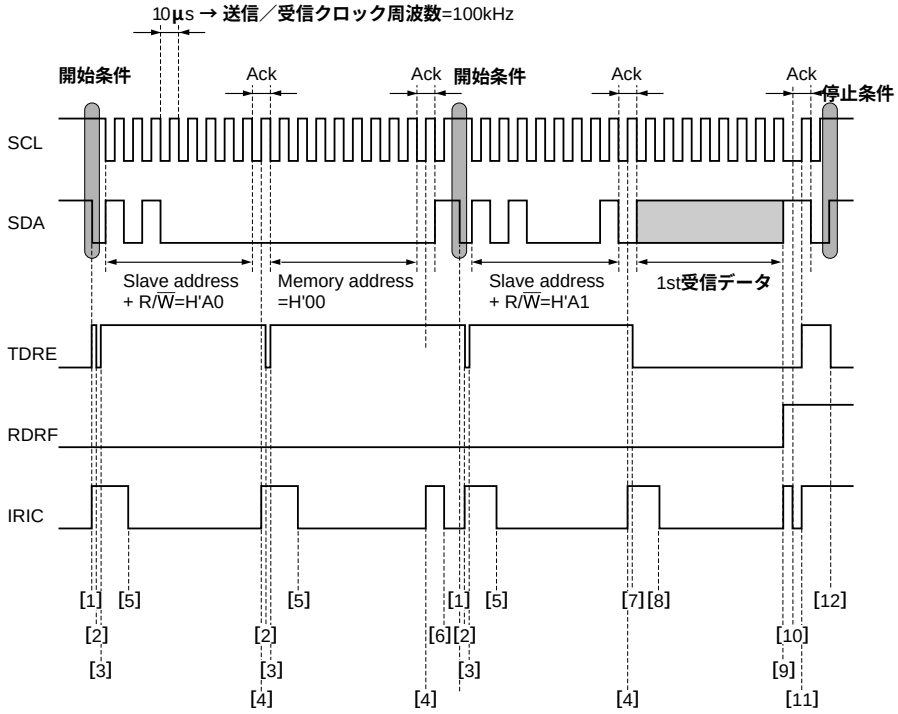
【記号説明】

S	**：開始条件**
SLA	：EEPROM**スレーブアドレス**
R/$\overline{W}$	**：送信／受信方向**
A	**：アクノリッジ**
MEA	：EEPROM**メモリアドレス**
DATA	**：受信データ**
P	**：停止条件**

図 4.15　**本タスク例で使用する転送フォーマット**

4.5.2 動作説明

図 4.16 に動作原理を示します。



	ソフトウェア処理	ハードウェア処理
[1]	開始条件の発行（BBSY=1、SCP=0）	IRIC=1、TDRE=1（バスラインの状態から開始条件を検出）
[2]	ICDR0に送信データをライト	TDRE=0（TRS=1の状態でICDRTにデータをライト）
[3]	処理なし	TDRE=1（ICDRTからICDRSにデータを転送）
[4]	処理なし	IRIC=1（データ送信の終了（送信クロックの9クロック目の立ち上がり時））
[5]	送信終了判定の為にIRICを0にクリア	処理なし
[6]	開始条件検出判定の為にIRICを0にクリア	処理なし
[7]	マスタ受信モードに設定（MST=1、TRS=0）	TDRE=0（TRS=0のとき）
[8]	受信終了判定のためにIRICを0にクリア	処理なし
[9]	処理なし	IRIC=1、RDRF=1(WAIT=1)（データ受信の終了（受信クロックの8クロック目の立ち下がり時））
[10]	受信クロックの9クロック目の出力終了判定のためにIRICを0にクリア	受信クロックの9クロック目の出力開始
[11]	マスタ送信モードに設定（MST=1、TRS=1）	IRIC=1（受信クロックの9クロック目の立ち上がり時） TDRE=1（開始条件検出後、TRS=0からTRS=1に切り替えたとき）
[12]	停止条件の発行（BBSY=0、SCP=0）	TDRE=0（停止条件の発行後、バスラインの状態から停止条件を検出）

図 4.16　**シングルマスタ受信による** 1 **バイトデータの受信動作原理**

4.5.3 ソフトウェア説明

(1) モジュール説明

表 4.11 に本タスク例におけるモジュール説明を示します。

表 4.11 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	使用 RAM 領域、および IIC0 の初期設定
シングルマスタ送信	mst_rec	シングルマスタ受信による EEPROM からの 1 バイトデータの受信
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス＋W データ（H'A0）の送信
Slave address + R 送信	trs_slvadr_a1	EEPROM のスレーブアドレス＋R データ（H'A1）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信
データ受信	rec_data	1 バイトデータの受信

(2) 使用内蔵レジスタ説明

表 4.12 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.12 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信／受信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジ間にウェイトを挿入するか否かを設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF Bit5 to Bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF Bit2 to Bit0	BC2=0 BC1=0 BC0=0

（続く）

表 4.12 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I²C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I²C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I²C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I²C バスインタフェースの送信／受信モードの設定	H'FFD8 bit4	0/1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM から受信したアクノリッジデータを格納 受信時、EEPROM へ送信するアクノリッジを設定	H'FFD9 bit0	-
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I²C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1

(3) 変数説明

表 4.13 に本タスク例における変数説明を示します。

表 4.13 変数説明

変数	機能	データ長	初期値	使用モジュール名
dummy	MDCR リード値	1 バイト	−	main

(4) 使用 RAM 説明

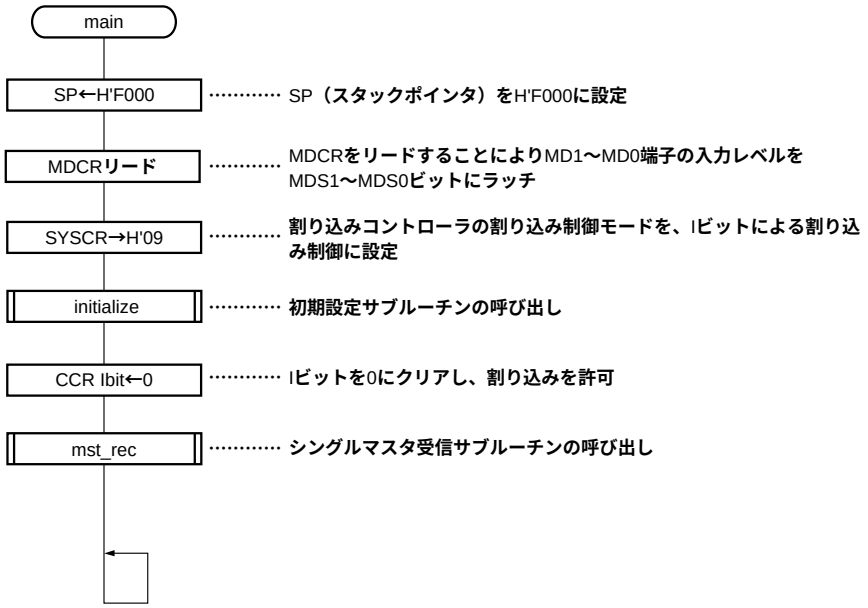
表 4.14 に本タスク例における使用 RAM 説明を示します。

表 4.14 使用 RAM 説明

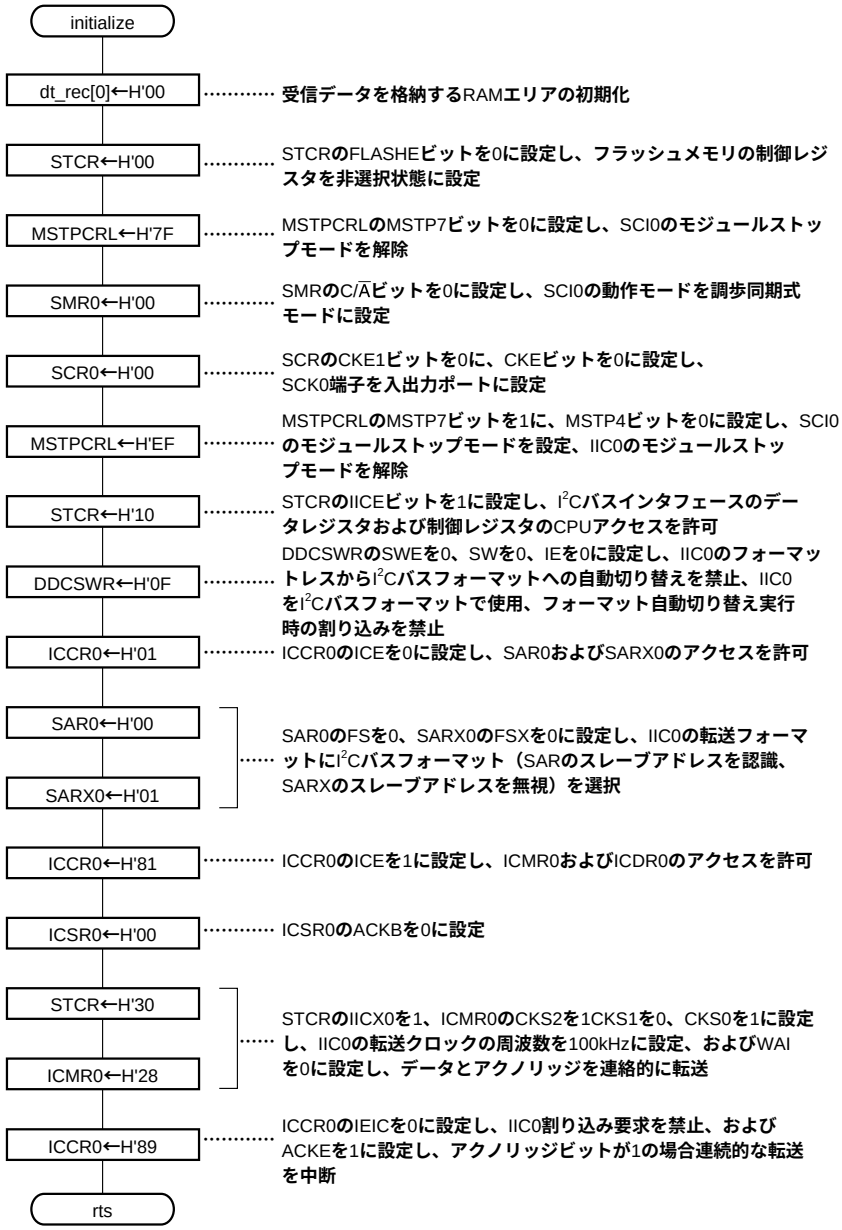
ラベル	機能	データ長	アドレス	使用モジュール名
dt_rec[0]	受信データを格納	1 バイト	H'E100	Initialize rec_data

4.5.4 フローチャート

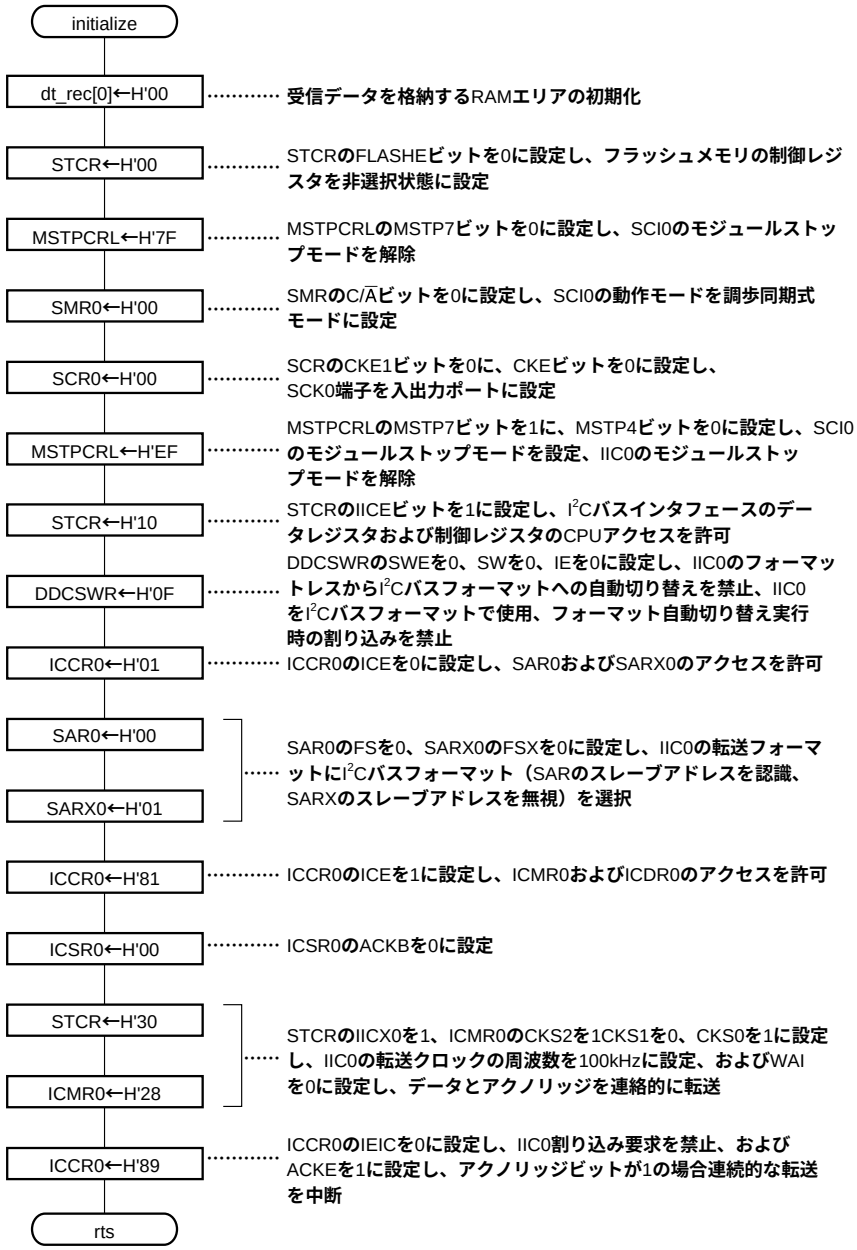
（1） メインルーチン



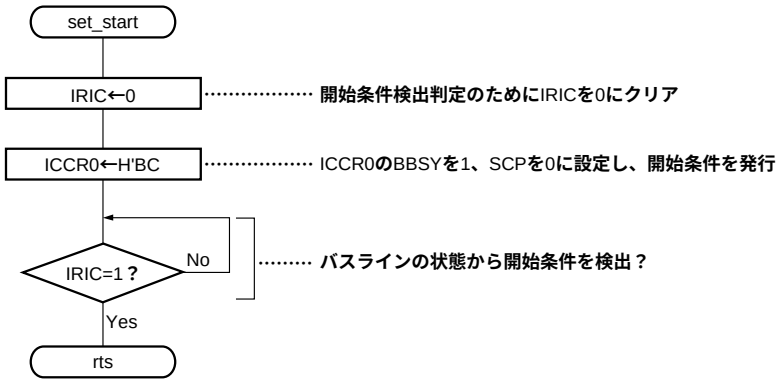
（2） 初期設定サブルーチン



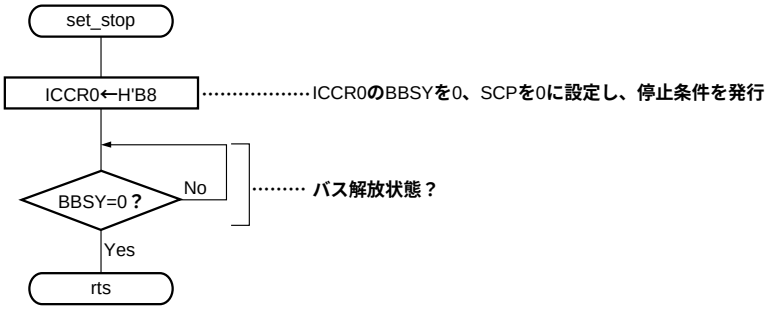
(3) シングルマスタ受信サブルーチン



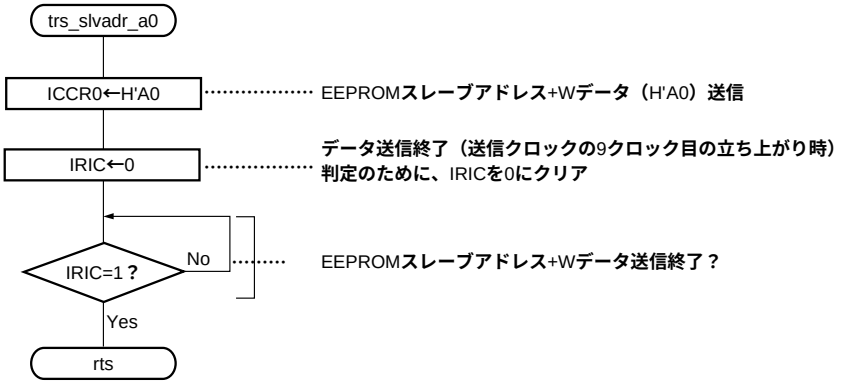
(4) 開始条件発行サブルーチン



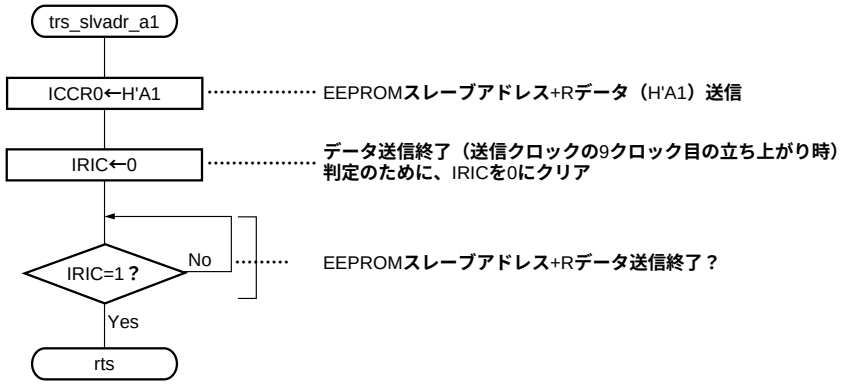
(5) 停止条件発行サブルーチン



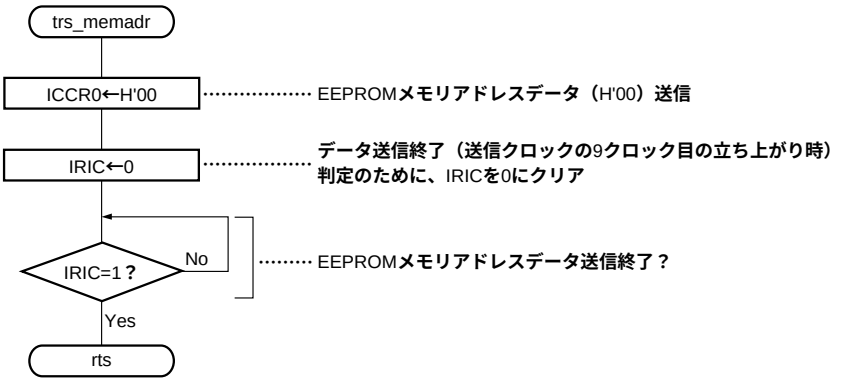
（6） Slave address + W 送信サブルーチン



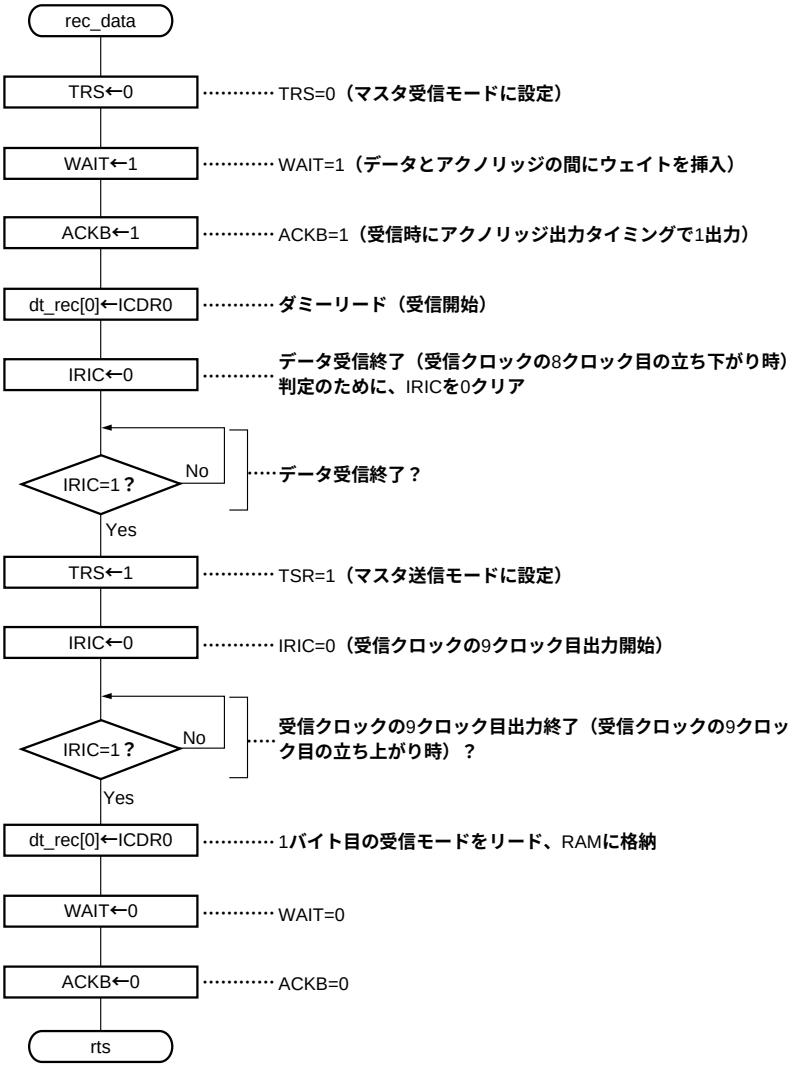
（7） Slave address + R 送信サブルーチン



（8） EEPROM memory address 送信サブルーチン



(9) データ受信サブルーチン



4.5.5 プログラムリスト

```
/*****************************************************
*  H8S/2138 IIC bus application note                 *
*      4.Single master receive 1byte data from EEPROM*
*                          File name    :   BYRxd.c    *
*                          Fai          :   20MHz      *
*                          Mode         :   3          *
*****************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/*************************************************
*  Prototype                                     *
*************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* RAM & IIC0 initialize */
void mst_rec(void);                              /* Master receive from EEPROM */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_slvadr_a1(void);                        /* Slave address + R data transmit */
void trs_memadr(void);                           /* EEPROM memory address data transmit */
void rec_data(void);                             /* 1-byte data receive */

/*************************************************
*  RAM allocation                                *
*************************************************/
#pragma section ramarea
unsigned char dt_rec[1];                         /* Receive data store area */

/*************************************************
*  main : Main routine                           *
*************************************************/
#pragma section
void main(void)
#pragma asm
```

（続く）

```
       mov.l   #h'f000,sp                          ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                             /* MCU mode set */

    SYSCR.BYTE = 0x09;                             /* Interrupt control mode set */
    initialize();                                  /* Initialize */
    set_imask_ccr(0);                              /* Interrupt enable */
    mst_rec();                                     /* Master receive from EPROM */
    while(1);                                      /* End */
}

/************************************************
*  initialize : RAM & IIC0 Initialize             *
************************************************/
void initialize(void)
{
    dt_rec[0] = 0x00;                              /* Receive data store area initialize */

    STCR.BYTE = 0x00;                              /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                          /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                          /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                          /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                              /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                            /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                         /* ICE = 0 */
    IIC0.SAR.BYTE = 0x00;                          /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                         /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                         /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                         /* ACKB = 0 */
    STCR.BYTE = 0x30;                              /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                         /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                         /* IEIC = 0, ACKE = 1 */
}
```

（続く）

```
/************************************************
*  mst_rec : Master receive from EEPROM          *
************************************************/
void mst_rec(void)
{
   while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */

   IIC0.ICCR.BIT.MST = 1;                        /* Master transmit mode set */
   IIC0.ICCR.BIT.TRS = 1;                        /* MST = 1, TRS = 1 */
   set_start();                                  /* Start condition set */
   trs_slvadr_a0();                              /* Slave address + W data transmit */

   if(IIC0.ICSR.BIT.ACKB == 0)                   /* ACKB = 0 ? */
   {
      trs_memadr();                              /* EEPROM memory address data transmit */
      if(IIC0.ICSR.BIT.ACKB == 0)                /* ACKB = 0 ? */
      {
         set_start();                            /* Re-start condition set */
         trs_slvadr_a1();                        /* Slave address + R data transmit */
         if(IIC0.ICSR.BIT.ACKB == 0)             /* ACKB = 0 ? */
         {
            rec_data();                          /* 1-byte data receive */
         }
      }
   }
   set_stop();                                   /* Stop condition set */
}

/************************************************
*  set_start : Start condition set               *
************************************************/
void set_start(void)
{
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   IIC0.ICCR.BYTE = 0xbc;                        /* Start condition set (BBSY=1,SCP=0) */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Start condition set (IRIC=1) ? */
```

（続く）

```
}


/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
   IIC0.ICCR.BYTE = 0xb8;                        /* Stop condition set (BBSY=0, SCP=0) */
   while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
{
   IIC0.ICDR = 0xa0;                             /* Slave address + W data(H'A0) write */
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_slvadr_a1 : Slave addres + R data transmit *
************************************************/
void trs_slvadr_a1(void)
{
   IIC0.ICDR = 0xa1;                             /* Slave address + R data(H'A1) write */
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
```

（続く）

```
    IIC0.ICDR = 0x00;                           /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}

/***********************************************
*  rec_data : 1-byte data receive              *
***********************************************/
void rec_data(void)
{
    IIC0.ICCR.BIT.TRS = 0;                      /* Master receive mode set (MST=1,TRS=0) */
    IIC0.ICMR.BIT.WAIT = 1;                     /* WAIT = 1 */
    IIC0.ICSR.BIT.ACKB = 1;                     /* ACKB = 1 */

    dt_rec[0] = IIC0.ICDR;                      /* Dummy read (Receive start) */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Receive end (IRIC=1) ? */

    IIC0.ICCR.BIT.TRS = 1;                      /* Master transmit mode set (MST=1,TRS=1) */
    IIC0.ICCR.BIT.IRIC = 0;                     /* 9th clock transmit start (IRIC=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* 9th clock transmit end (IRIC=1) ? */

    dt_rec[0] = IIC0.ICDR;                      /* 1-byte receive data read */

    IIC0.ICMR.BIT.WAIT = 0;                     /* WAIT = 0 */
    IIC0.ICSR.BIT.ACKB = 0;                     /* ACKB = 0 */
}
```

4.6 DTC によるシングルマスタ送信

4.6.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）にデータトランスファコントローラ（DTC）を使用して、10 バイトのデータを書き込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地から H'09 番地にデータを書き込みます。
- 書込む 10 バイトのデータは RAM の H'E102 番地から H'E10B 番地に格納します。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.17 に H8S/2138 と EEPROM の接続例を示します。

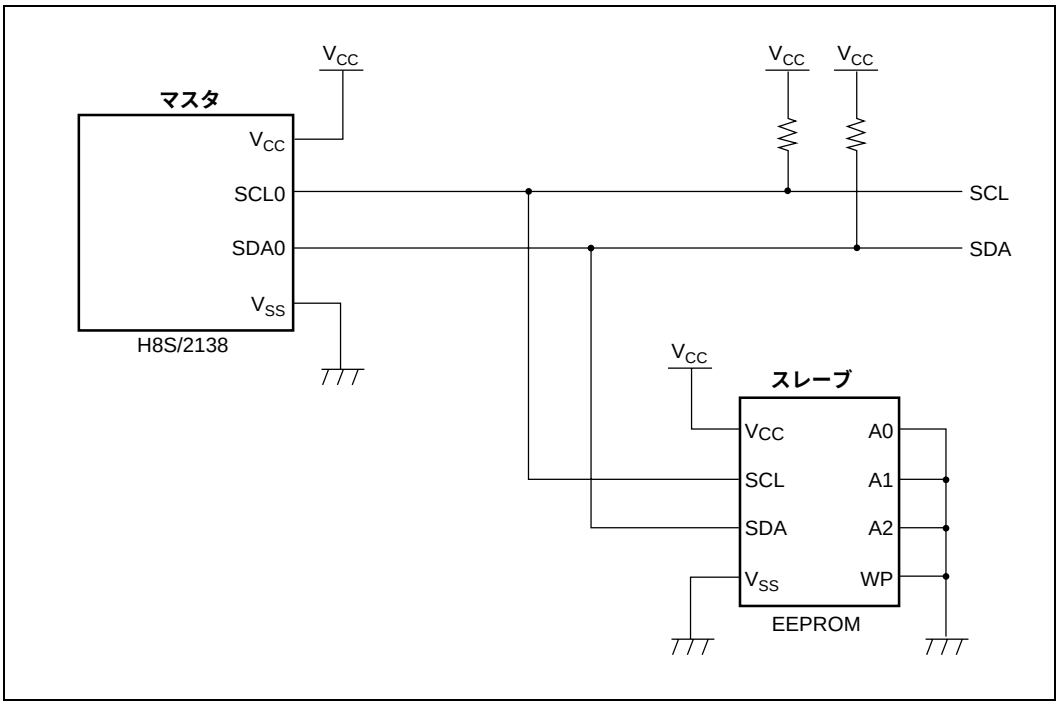


図 4.17 H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.18 に示します。

【記号説明】

S	**：開始条件**
SLA	**：**EEPROM**スレーブアドレス**
R/$\overline{W}$	**：送信／受信方向**
A	**：アクノリッジ**
MEA	**：**EEPROM**メモリアドレス**
DATA	**：送信データ**
P	**：停止条件**

図 4.18　**本タスク例で使用する転送フォーマット**

- 以下に本タスク例で使用する H8S/2138 シリーズのデータトランスファコントローラ（DTC）の使用例について説明します。

1. I²C バスインタフェースのチャネル 0 の割り込み要求（IICI0）により DTC を起動し、送信データの転送を行います。
2. DTC の転送モードはノーマルモードで使用します。
3. **図** 4.19 に本タスク例で使用する DTC のブロック図を示します。

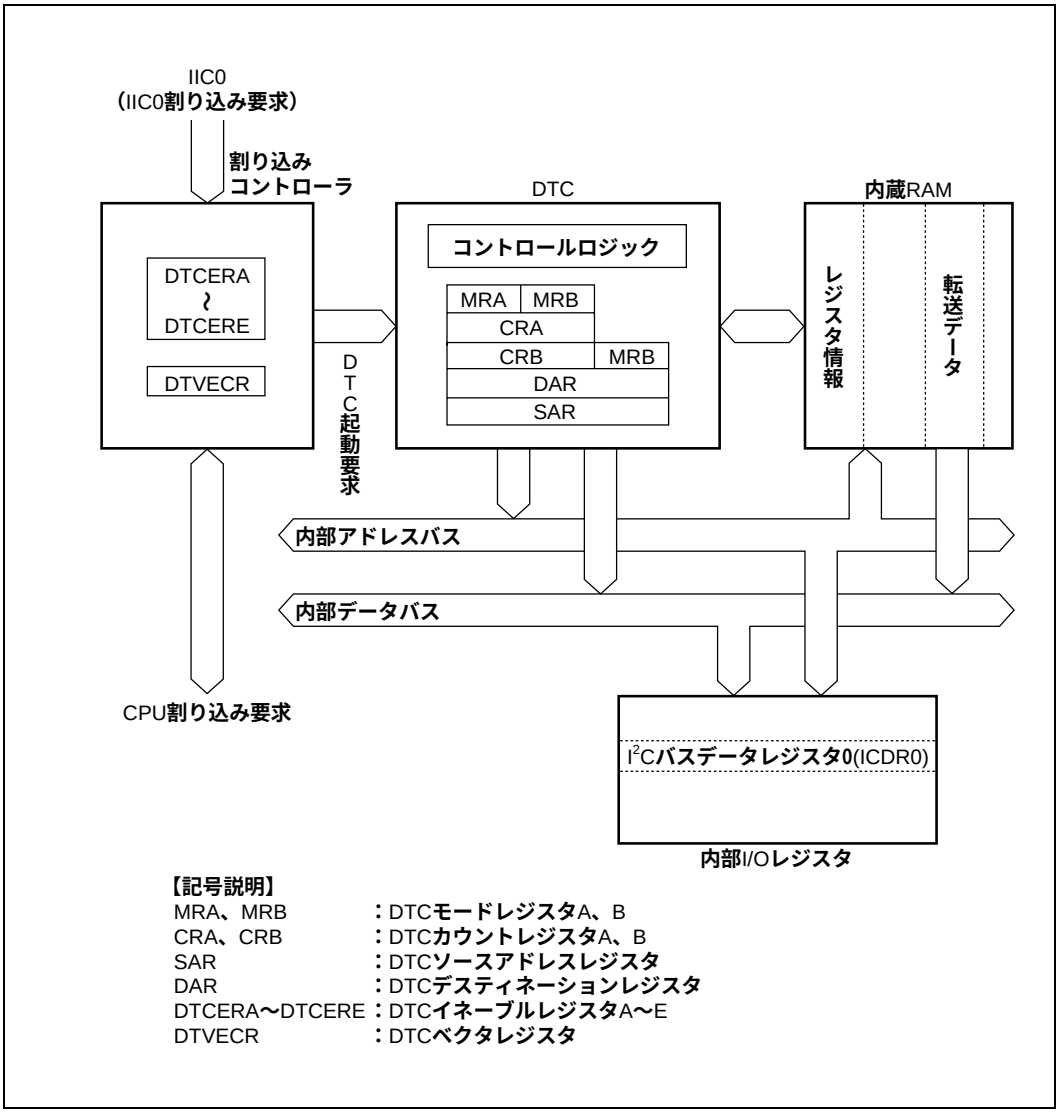


図 4.19 **本タスク例における** DTC **ブロック図**

4. 図 4.20 に内蔵 RAM 上の転送データの配置を示します。

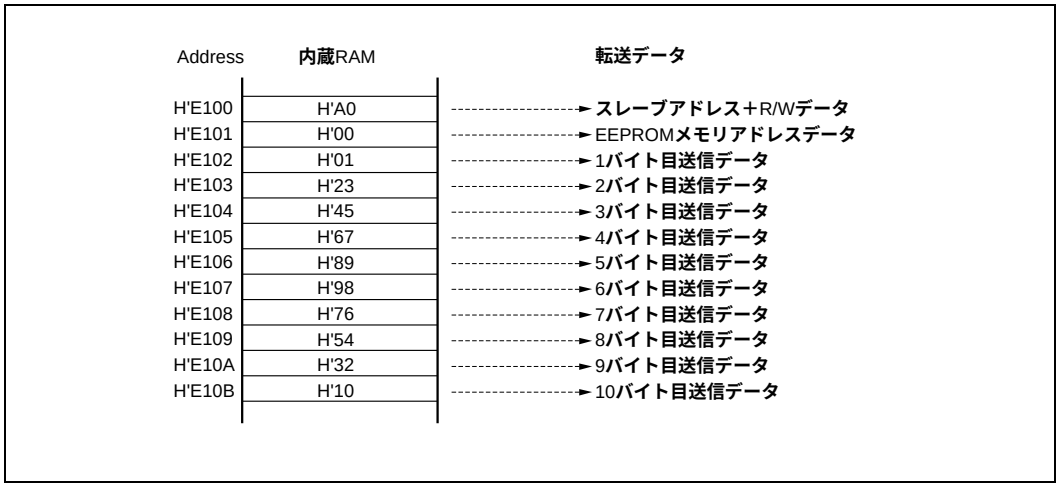


図 4.20　内蔵 RAM 上の転送データの配置

5. 図 4.21 に本タスク例における DTC ベクタテーブルと内蔵 RAM 上のレジスタ情報の配置を示します。H’EC00 番地から MRA、SAR、MRB、DAR、CRA、CRB の順に DTC のレジスタ情報を配置します。

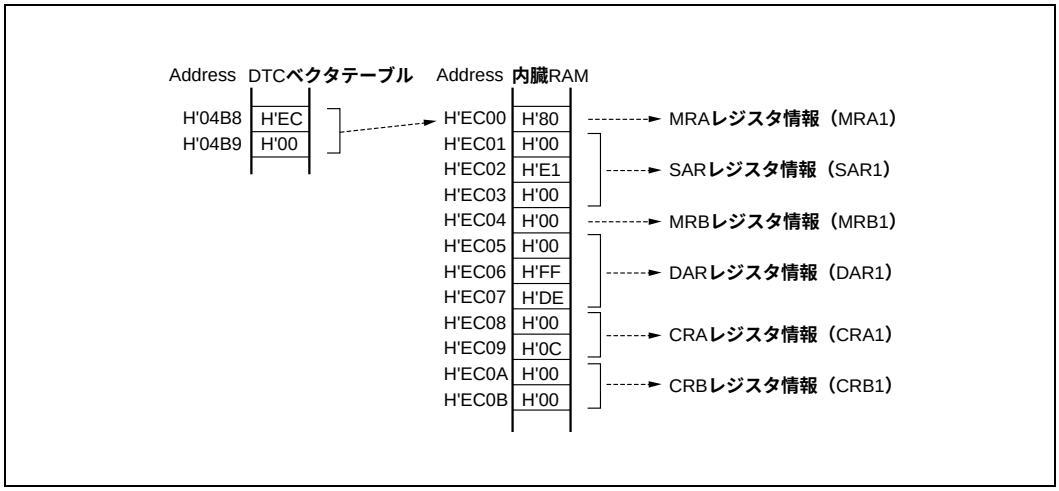


図 4.21　DTC ベクタテーブルと内蔵 RAM 上のレジスタ情報の配置

6. 表 4.15 に本タスク例で使用する DTC のレジスタ説明を示します。

表 4.15 DTC のレジスタ説明

レジスタ		機能
MRA		DTC モードレジスタ A ～DTC の動作モードの制御を行います。
	SM1, 0 (bit7、6)	ソースアドレスモード 1、0 ～データ転送後に、SAR をインクリメントするか、デクリメントするか、または固定とするかを指定します。 • SM1=0、SM0=＊のとき、SAR は固定（＊：“0”または“1”） • SM1=1、SM0=0 のとき、SAR は転送後インクリメント（Sz=0 のとき+1、Sz=1 のとき+2） • SM1=1、SM0=1 のとき、SAR は転送後デクリメント（Sz=0 のとき-1、Sz=1 のとき-2）
	DM1, 0 (bit5、4)	デスティネーションアドレスモード 1、0 ～データ転送後に、DAR をインクリメントするか、デクリメントするか、または固定とするかを指定します。 • DM1=0、DM0=＊のとき、DAR は固定（＊：“0”または“1”） • DM1=1、DM0=0 のとき、DAR は転送後インクリメント（Sz=0 のとき+1、Sz=1 のとき+2） • DM1=1、DM0=1 のとき、DAR は転送後デクリメント（Sz=0 のとき-1、Sz=1 のとき-2）
	MD1, 0 (bit3、2)	DTC モード 1、0 ～DTC の転送モードを指定します。 • MD1=0、MD0=0 のとき、ノーマルモード • MD1=0、MD0=1 のとき、リピートモード • MD1=1、MD0=0 のとき、ブロック転送モード • MD1=1、MD0=1 のとき、設定禁止
	DTS (bit1)	DTC 転送モードセレクト ～リピートモードまたはブロック転送モードのとき、ソース側とデスティネーション側のいずれをリピート領域またはブロック領域とするかを指定します。 • DTS=0 のとき、デスティネーション側がリピート領域またはブロック領域 • DTS=1 のとき、ソース側がリピート領域またはブロック領域
	Sz (bit0)	DTC データトランスファサイズ ～データ転送のデータサイズを指定します。
MRB		DTC モードレジスタ B ～DTC モードの制御を行います。
	CHEN (bit7)	DTC チェイン転送イネーブル ～チェイン転送を指定します。 • CHEN=0 のとき、DTC データ転送終了 • CHEN=1 のとき、DTC チェイン転送

（続く）

表 4.15 DTC のレジスタ説明（続き）

レジスタ		機能
MRB	DISEL (bit6)	DTC インタラプトセレクト 〜1 回のデータ転送後に CPU への割り込み要求の禁止または許可を指定します。 • DISEL=0 のとき、DTC データ転送終了後、転送カウンタが 0 でなければ、CPU への割り込みを禁止 • DISEL=1 のとき、DTC データ転送終了後、CPU への割り込みを許可
SAR		DTC ソースアドレスレジスタ 〜DTC の転送するデータの転送元アドレスを指定します。
DAR		DTC デスティネーションアドレスレジスタ 〜DTC の転送するデータの転送元アドレスを指定します。
CRA		DTC 転送カウントレジスタ A 〜DTC のデータ転送の転送回数を指定します。
CRB		DTC 転送カウントレジスタ B 〜ブロック転送モードのとき、DTC のブロックデータ転送の転送回数を指定します。
DTVECR(H'FEF3)		DTC ベクタレジスタ 〜ソフトウェアによる DTC 起動の許可または禁止の設定、およびソフトウェア起動割り込み用ベクタ番号を設定します。
	SWDTE (bit7)	DTC ソフトウェア起動イネーブル 〜DTC ソフトウェア起動の許可または禁止を設定します。 • SWDTE=0 のとき、DTC ソフトウェア起動を禁止 • SWDTE=1 のとき、DTC ソフトウェア起動を許可
	DTVEC6-0 (bit6-0)	DTC ソフトウェア起動ベクタ 6〜0 〜DTC ソフトウェア起動のベクタ番号を設定します。
DTCERD(H'FEF1)		DTC イネーブルレジスタ 〜各割り込み要因による DTC 起動の許可または禁止を制御します。
	DTCED4 (bit4)	DTC 起動イネーブル D4 • DTCED4=0 のとき、IICI0 割り込みによる DTC 起動を禁止 • DTCED4=2 のとき、IICI0 割り込みによる DTC 起動を許可

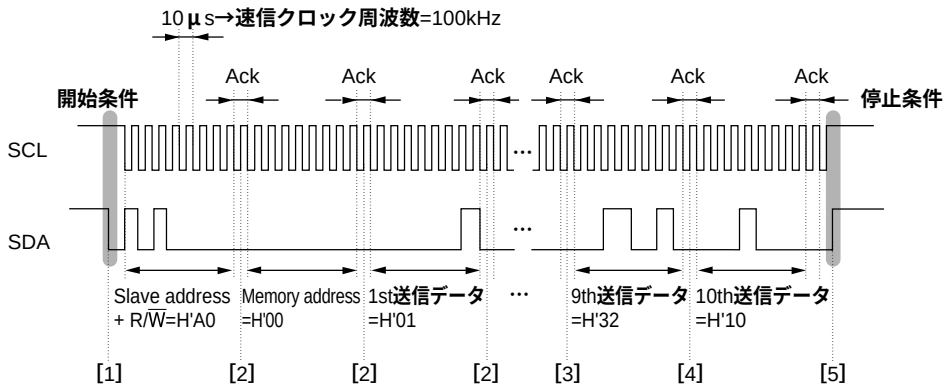
7. I²C バスフォーマットでは、スレーブアドレスと R/$\overline{W}$ ビットによるスレーブデバイスおよび転送方向の選択や、アクノリッジビットによる受信の確認および最終フレームの表示などが行われるため、DTC によるデータの連続転送は、割り込みによる CPU 処理と組み合わせて行う必要があります。**表** 4.16 に本タスク例における DTC を利用したマスタ送信モードにおける処理の例を示します。

表 4.16　DTC **による動作例（マスタ送信モード）**

項目	マスタ送信モード
スレーブアドレス +R/$\overline{W}$ **ビット送信**	DTC **で送信（**ICDR **ライト）**
ダミーデータリード	**－**
本体データ送信	DTC **で送信（**ICDR **ライト）**
ダミーデータ (H'FF)**ライト**	**－**
最終フレーム処理	**不要**
最終フレーム処理後の転送要求処理	1 **回目：**CPU **でクリア** 2 **回目：**CPU **で終了条件発行**
DTC **転送データフレーム数設定**	**送信：実データ数**+1 **（**+1 **はスレーブアドレス**+R/W **ビット分）**

4.6.2 動作説明

図 4.22 に動作原理を示します。



	CPU処理	DTC処理
[1]	開始条件の発行（BBSY=1、SCP=0）	処理なし
[2]	処理なし	送信データをICDRにライト、IRICを“0”にクリア
[3]	IRICを“0”にクリア（1回目IICI0割り込み処理）	送信データをICDRにライト
[4]	IRICを“0”にクリア（2回目IICI0割り込み処理）	処理なし
[5]	停止条件の発行（BBSY=1、SCP=0）	処理なし

図 4.22 DTC **によるシングルマスタの送信動作原理**

4.6.3 ソフトウェア説明

(1) モジュール説明

表 4.17 に本タスク例におけるモジュール説明を示します。

表 4.17 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	使用 RAM 領域、IIC0、DTC の初期設定
送信セットアップ	trs_stup	マスタ送信モードの設定、開始条件の発行
IIC0 割り込み処理	iici0	IRIC のクリア、停止条件の発行

(2) 使用内蔵レジスタ説明

表 4.18 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.18 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信データを格納	H'FFDE	ー
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジ間にウェイトを挿入するか否かを設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて､転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS 2=1 CKS1=0 CKS 0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I^2C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I^2C バスインタフェースを送信モードで使用	H'FFD8 bit4	1
	ACKE	アクノリッジビットが “1” の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1

（続く）

表 4.18 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM から受信したアクノリッジデータを格納 受信時、EEPROM へ送信したアクノリッジを設定	H'FFD9 bit0	-
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1
MRA	SM1、0	データ転送後、SAR をインクリメントに設定	H'EC00 bit7、6	SM1=1 SM0=0
	DM1、0	データ転送後、DAR を固定に設定	H'EC00 bit5、4	DM1=0 DM0=0
	MD1、0	DTC の転送モードをノーマルモードに設定	H'EC00 bit3、2	MD1=0 MD0=0
	DTS	デスティネーション側がリピート領域またはブロック領域に設定	H'EC00 bit1	DTS=0
	Sz	データ転送のデータサイズをバイトサイズに設定	H'EC00 bit0	Sz=0

（続く）

表 4.18 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
MRB	CHNE	DTC チェイン転送をディスエーブルに設定	H'EC04 bit7	CHNE=0
	DISEL	1 回のデータ転送後、転送カウンタが 0 でなければ CPU への割り込みを禁止に設定	H'EC04 bit6	DISSEL=0
SAR		DTC の転送するデータの転送元アドレスを H'E100 に設定	H'EC01	H'00E100
DAR		DTC の転送するデータの転送先アドレスを H'FFDE に設定	H'EC05	H'00FFDE
CRA		DTC のデータ転送の転送回数を 12 回に設定	H'EC08	H'000C
CRB		ブロック転送モード時の DTC のブロックデータ転送の転送回数を 0 回に設定	H'EC0A	H'0000
DTVECR	SWDTE	DTC ソフトウェア起動の禁止を設定	H'FEF3 bit7	0
	DTVEC6 to DTVEC0	DTC ソフトウェア起動のベクタ番号を H'00 に設定	H'FEF3 bit6 to bit0	H'00
DTCERD	DTCED4	IICI0 割り込みによる DTC 起動を許可	H'FEF1 bit4	1
MSTPCRH	MSTP14	DTC のモジュールストップモードの解除	H'FF86 bit6	0

(3) 変数説明

表 4.19 に本タスク例における変数説明を示します。

表 4.19 **変数説明**

変数	機能	データ長	初期値	使用モジュール名
dummy	MDCR **リード値**	1 **バイト**	−	main
i	**送信データカウンタ**	1 **バイト**	H'00	initialize
dt_trs[0]	**スレーブアドレス**+W **データ**	1 **バイト**	H'A0	initialize
dt_trs[1]	EEPROM **メモリアドレスデータ**	1 **バイト**	H'00	initialize
dt_trs[2]	1 **バイト目送信データ**	1 **バイト**	H'01	initialize
dt_trs[3]	2 **バイト目送信データ**	1 **バイト**	H'23	initialize
dt_trs[4]	3 **バイト目送信データ**	1 **バイト**	H'45	initialize
dt_trs[5]	4 **バイト目送信データ**	1 **バイト**	H'67	initialize
dt_trs[6]	5 **バイト目送信データ**	1 **バイト**	H'89	initialize
dt_trs[7]	6 **バイト目送信データ**	1 **バイト**	H'98	initialize
dt_trs[8]	7 **バイト目送信データ**	1 **バイト**	H'76	initialize
dt_trs[9]	8 **バイト目送信データ**	1 **バイト**	H'54	initialize
dt_trs[10]	9 **バイト目送信データ**	1 **バイト**	H'32	initialize
dt_trs[11]	10 **バイト目送信データ**	1 **バイト**	H'10	initialize

(4) 使用 RAM 説明

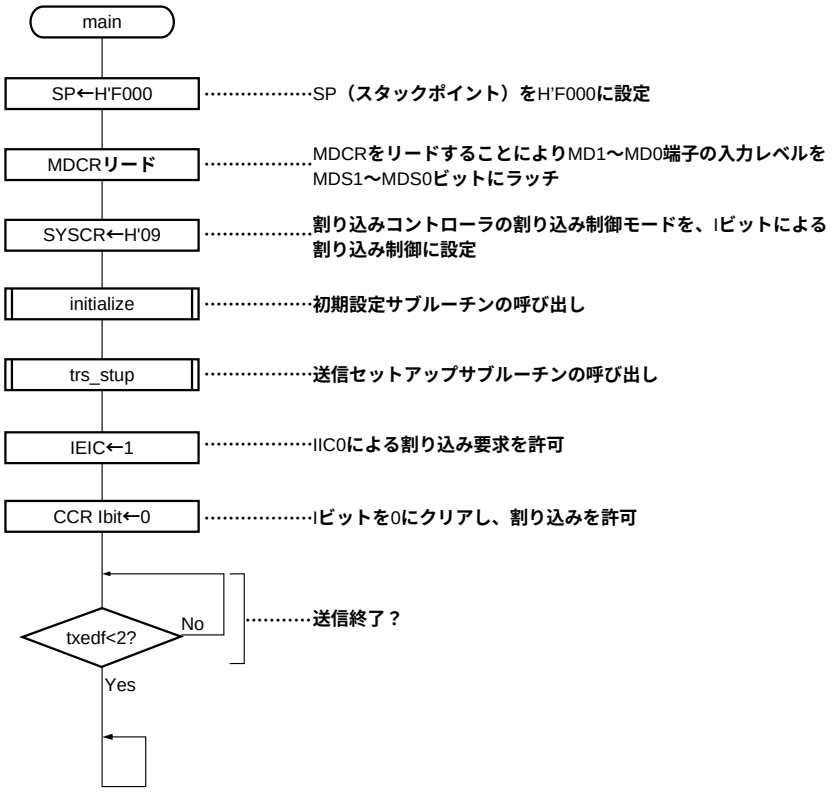
表 4.20 に本タスク例における使用 RAM 説明を示します。

表 4.20 使用 RAM 説明

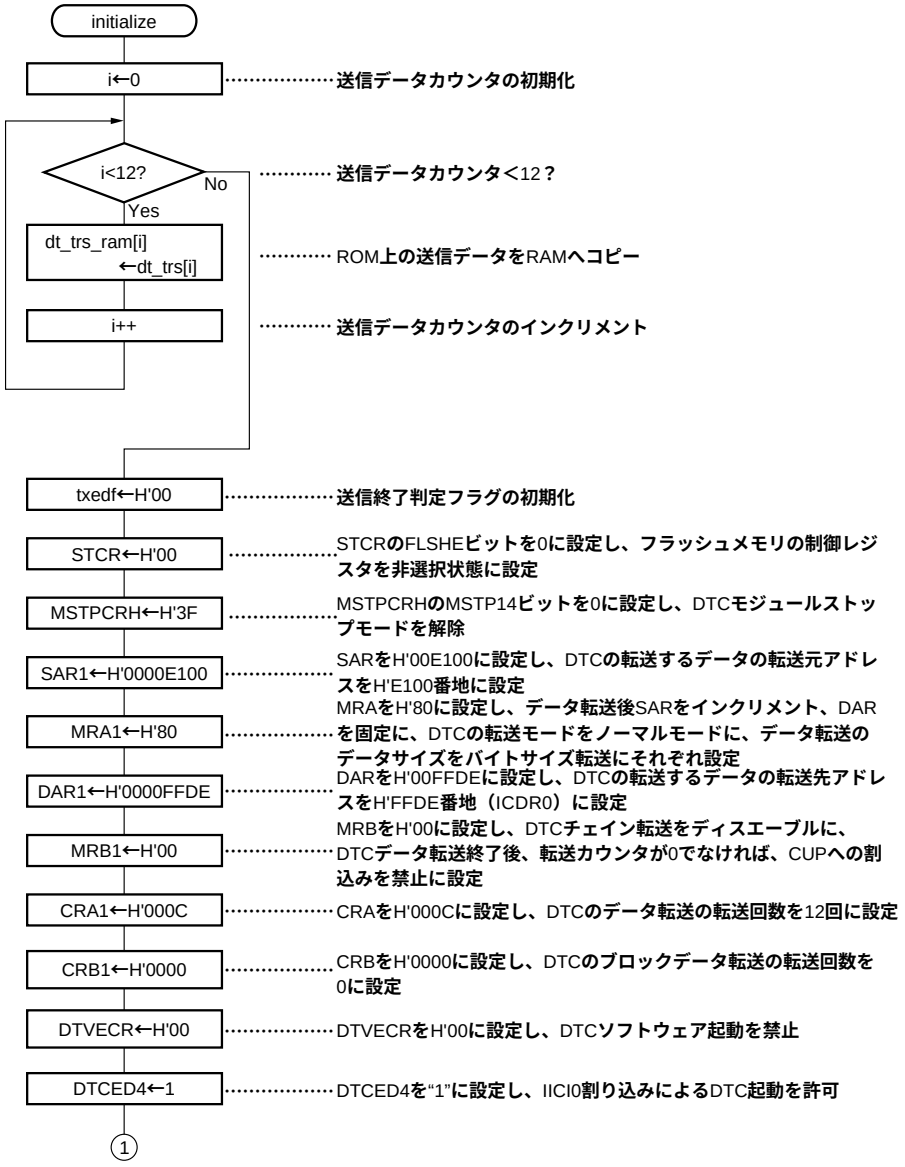
ラベル	機能	データ長	アドレス	使用モジュール名
MRA1	DTC モードレジスタ A（MRA）	1 バイト	H'EC00	initialize
SAR1	DTC ソースアドレスレジスタ（SAR）	4 バイト	H'EC00	initialize
MRB1	DTC モードレジスタ B（MRB）	1 バイト	H'EC04	initialize
DAR1	DTC ディスティネーションアドレスレジスタ（DAR）	4 バイト	H'EC04	initialize
CRA1	DTC 転送カウントレジスタ A（CRA）	2 バイト	H'EC08	initialize
CRB1	DTC 転送カウントレジスタ B（CRB）	2 バイト	H'EC0A	initialize
txedf	送信終了判定フラグ	1 バイト	H'E200	main iici0
dt_trs_ram [0]	Slave address + R/W データを格納	1 バイト	H'E100	initialize
dt_trs_ram [1]	EEPROM メモリアドレスデータを格納	1 バイト	H'E101	initialize
dt_trs_ram [2] to dt_trs_ram [11]	10 バイトの送信データを格納	10 バイト	H'E102 or H'E10B	initialize

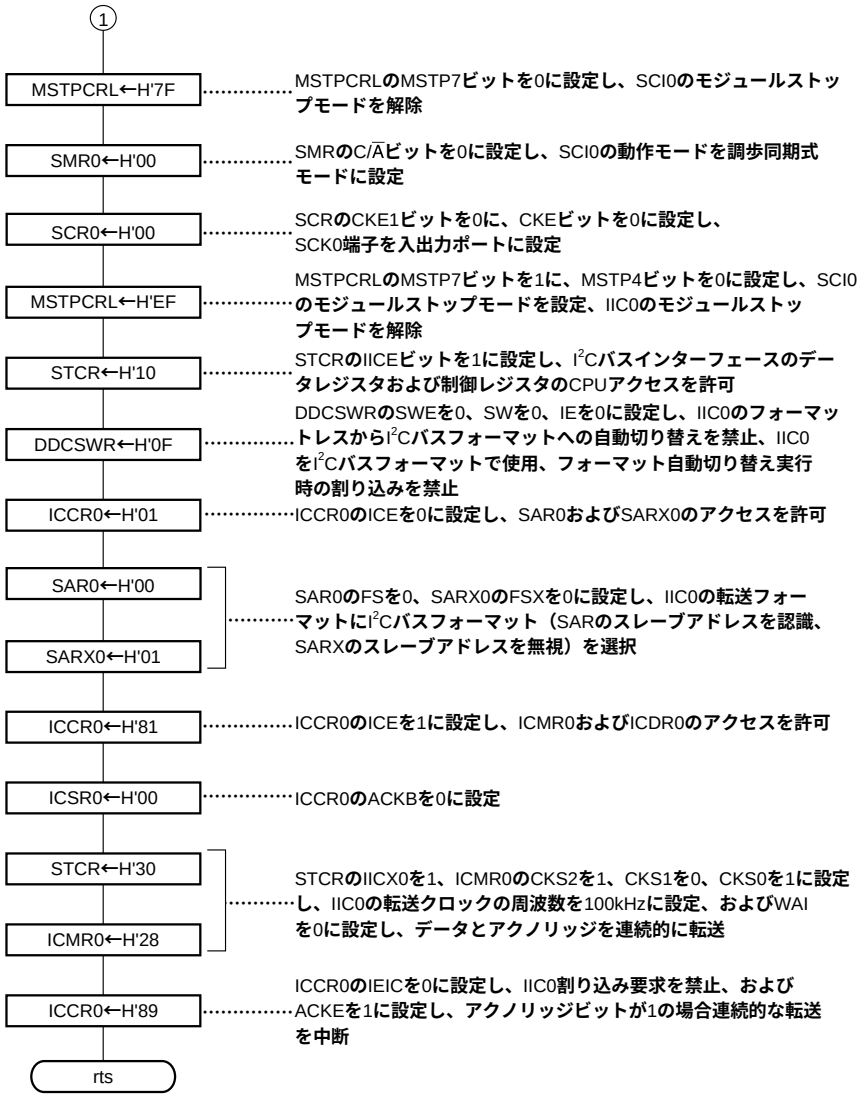
4.6.4 フローチャート

（1） メインルーチン

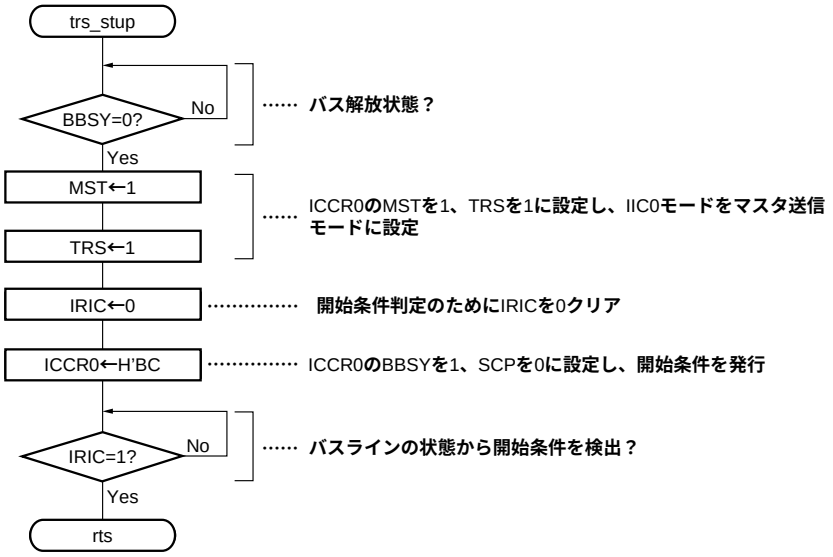


(2) 初期設定サブルーチン

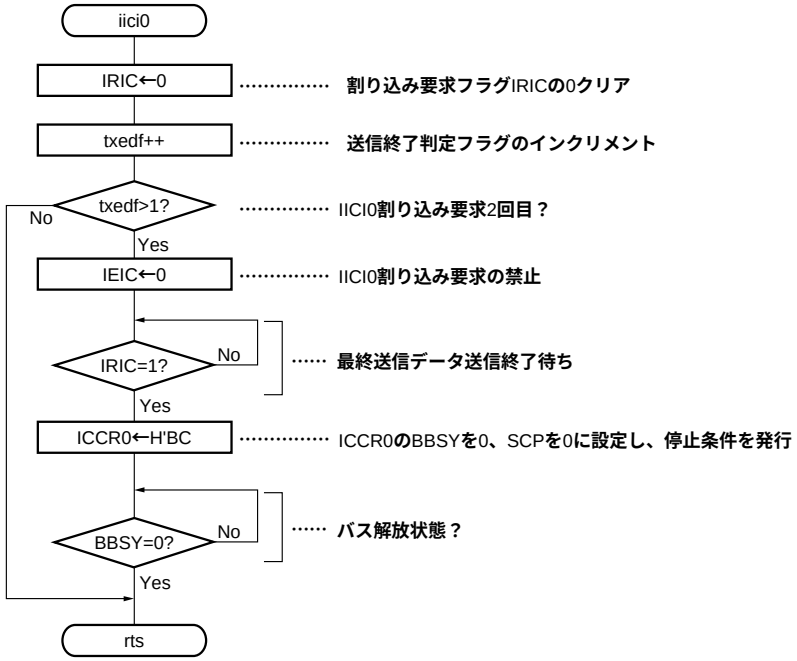



1
MSTPCRL←H'7F
MSTPCRLのMSTP7ビットを0に設定し、SCI0のモジュールストップモードを解除
SMR0←H'00
SMRのC/Āビットを0に設定し、SCI0の動作モードを調歩同期式モードに設定
SCR0←H'00
SCRのCKE1ビットを0に、CKEビットを0に設定し、SCK0端子を入出力ポートに設定
MSTPCRL←H'EF
MSTPCRLのMSTP7ビットを1に、MSTP4ビットを0に設定し、SCI0のモジュールストップモードを設定、IIC0のモジュールストップモードを解除
STCR←H'10
STCRのIICEビットを1に設定し、I2Cバスインターフェースのデータレジスタおよび制御レジスタのCPUアクセスを許可
DDCSWR←H'0F
DDCSWRのSWEを0、SWを0、IEを0に設定し、IIC0のフォーマットレスからI2Cバスフォーマットへの自動切り替えを禁止、IIC0をI2Cバスフォーマットで使用、フォーマット自動切り替え実行時の割り込みを禁止
ICCR0←H'01
ICCR0のICEを0に設定し、SAR0およびSARX0のアクセスを許可
SAR0←H'00
SARX0←H'01
SAR0のFSを0、SARX0のFSXを0に設定し、IIC0の転送フォーマットにI2Cバスフォーマット（SARのスレーブアドレスを認識、SARXのスレーブアドレスを無視）を選択
ICCR0←H'81
ICCR0のICEを1に設定し、ICMR0およびICDR0のアクセスを許可
ICSR0←H'00
ICCR0のACKBを0に設定
STCR←H'30
ICMR0←H'28
STCRのIICX0を1、ICMR0のCKS2を1、CKS1を0、CKS0を1に設定し、IIC0の転送クロックの周波数を100kHzに設定、およびWAIを0に設定し、データとアクノリッジを連続的に転送
ICCR0←H'89
ICCR0のIEICを0に設定し、IIC0割り込み要求を禁止、およびACKEを1に設定し、アクノリッジビットが1の場合連続的な転送を中断
rts

(3) 送信セットアップサブルーチン



（4） IIC0 割り込み処理ルーチン



4.6.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*      5.Single master transmit by DTC            *
*                          File name   :  DTCtx.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                      *
************************************************/
void main(void);                                  /* Main routine */
void initialize(void);                            /* RAM & DTC & IIC0 initialize */
void trs_stup(void);                              /* Master transmit by DTC set up */

/************************************************
*  RAM allocation                                 *
************************************************/
#define MRA1 (*(volatile unsigned char *)0xec00)  /* DTC mode register A */
#define SAR1 (*(volatile unsigned long *)0xec00)  /* DTC source address register */
#define MRB1 (*(volatile unsigned char *)0xec04)  /* DTC mode register B */
#define DAR1 (*(volatile unsigned long *)0xec04)  /* DTC destination address register */
#define CRA1 (*(volatile unsigned short *)0xec08) /* DTC transfer count register A */
#define CRB1 (*(volatile unsigned short *)0xec0a) /* DTC transfer count register B */

#define txedf (*(volatile unsigned char *)0xe200) /* Transmit end flag */

#pragma section ramarea
unsigned char dt_trs_ram[12];                     /* Transmit data store area */
#pragma section
```

（続く）

```
/************************************************
*  Data table                                   *
************************************************/
const unsigned char dt_trs[12] =
{
   0xa0,                                        /* Slave address + W data */
   0x00,                                        /* EEPROM memory address data */
   0x01,                                        /* 1st transmit data */
   0x23,                                        /* 2nd transmit data */
   0x45,                                        /* 3rd transmit data */
   0x67,                                        /* 4th transmit data */
   0x89,                                        /* 5th transmit data */
   0x98,                                        /* 6th transmit data */
   0x76,                                        /* 7th transmit data */
   0x54,                                        /* 8th transmit data */
   0x32,                                        /* 9th transmit data */
   0x10                                         /* 10th transmit data */
};

/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                       ;Stack pointer initialize
#pragma endasm
{
   unsigned char dummy;
   dummy = MDCR.BYTE;                           /* MCU mode set */
   SYSCR.BYTE = 0x09;                           /* Interrupt control mode set */

   initialize();                                /* Initialize */
   trs_stup();                                  /* Master transmit by DTC set up */

   IIC0.ICCR.BIT.IEIC = 1;                      /* IIC0 interrupt enable */
   set_imask_ccr(0);                            /* Interrupt enable */
```

（続く）

```
   while(txedf < 2);                              /* Transmit end ? */
   while(1);                                      /* End */
}

/************************************************
*  initialize : RAM & IIC0 Initialize            *
************************************************/
void initialize(void)
{
   unsigned char i;                               /* Transmit data counter */

   for(i=0; i<12; i++)                            /* Transmit data copy ROM -> RAM */
   {
      dt_trs_ram[i] = dt_trs[i];
   }

   txedf = 0x00;                                  /* Transmit end flag initialize */

   STCR.BYTE = 0x00;                              /* FLSHE = 0 */

   MSTPCR.BYTE.H = 0x3f;                          /* DTC module stop mode reset */
   SAR1 = 0x0000e100;                             /* SAR = H'00E100 */
   MRA1 = 0x80;                                   /* MRA = H'80 */
   DAR1 = 0x0000ffde;                             /* DAR = H'00FFED (ICDR0) */
   MRB1 = 0x00;                                   /* MRB = H'00 */
   CRA1 = 0x000c;                                 /* CRA = H'000C */
   CRB1 = 0x0000;                                 /* CRB = H'0000 */
   DTC.VECR.BYTE = 0x00;                          /* SWDTE = 0, DTVEC = H'00 */
   DTC.ED.BIT.B4 = 1;                             /* DTCED4 = 1 */

   MSTPCR.BYTE.L = 0x7f;                          /* SCI0 module stop mode reset */
   SCI0.SMR.BYTE = 0x00;                          /* SCL0 pin function set */
   SCI0.SCR.BYTE = 0x00;
   MSTPCR.BYTE.L = 0xef;                          /* IIC0 module stop mode reset */
   STCR.BYTE = 0x10;                              /* IICE = 1 */
   DDCSWR.BYTE = 0x0f;                            /* IIC bus format initialize */
   IIC0.ICCR.BYTE = 0x01;                         /* ICE = 0 */
```

（続く）

```
    IIC0.SAR.BYTE = 0x00;                          /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                         /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                         /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                         /* ACKB = 0 */
    STCR.BYTE = 0x30;                              /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                         /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                         /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  trs_stup : Master transmit by DTC set up       *
************************************************/
void trs_stup(void)
{
    while(IIC0.ICCR.BIT.BBSY == 1);                /* Bus empty (BBSY=0) ? */
    IIC0.ICCR.BIT.MST = 1;                         /* Master transmit mode set */
    IIC0.ICCR.BIT.TRS = 1;                         /* MST = 1, TRS = 1 */

    IIC0.ICCR.BIT.IRIC = 0;                        /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                         /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);                /* Start condition set (IRIC=1) ? */
}

/************************************************
*  iici0 : IIC0 interrupt routine                 *
************************************************/
#pragma interrupt(iici0)
void iici0(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                        /* IRIC = 0 */
    txedf++;

    if(txedf > 1)
    {
        IIC0.ICCR.BIT.IEIC = 0;                    /* IIC0 interrupt disable */
        while(IIC0.ICCR.BIT.IRIC == 0);            /* Transmit end (IRIC=0) ? */
```

（続く）

```
        IIC0.ICCR.BYTE = 0xb8;                  /* Stop condition set (BBSY=0,SCP=0) */
        while(IIC0.ICCR.BIT.BBSY == 1);         /* Bus empty (BBSY=0) ? */
    }
}
```

4.7 DTC によるシングルマスタ受信

4.7.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、EEPROM（HN58X2408）からデータトランスファコントローラ（DTC）を使用して、10 バイトのデータを読み込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地から H'09 番地のデータを読み込みます。
- 読み込む 10 バイトのデータは RAM の H'E100 番地から H'E109 番地に格納します。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（EEPROM）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.23 に H8S/2138 と EEPROM の接続例を示します。

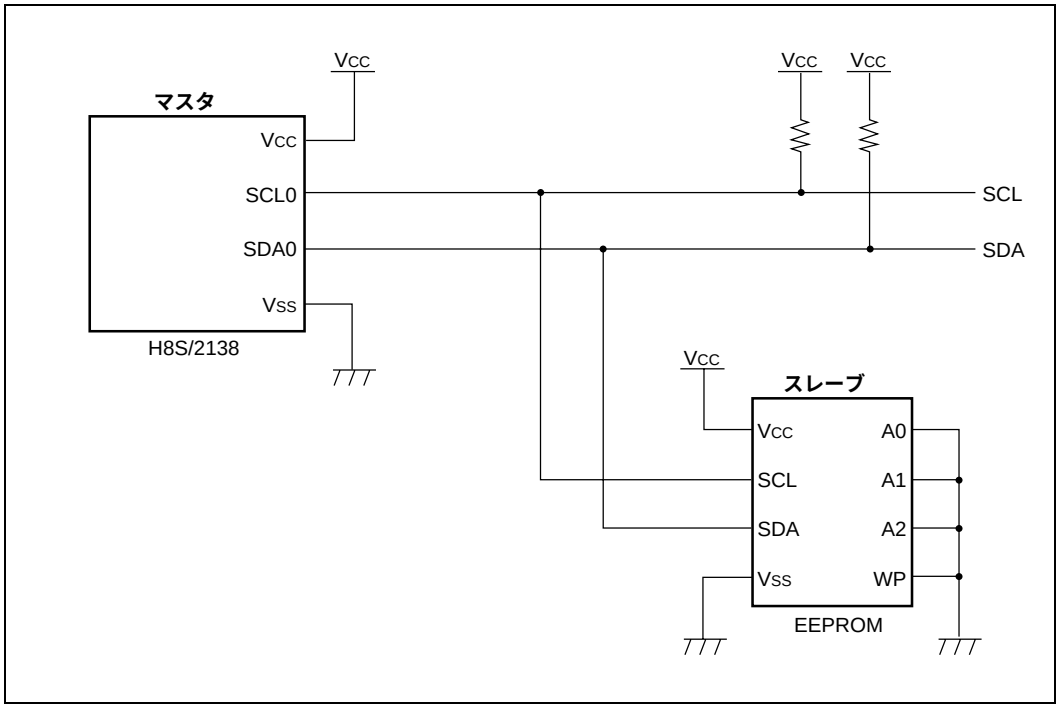
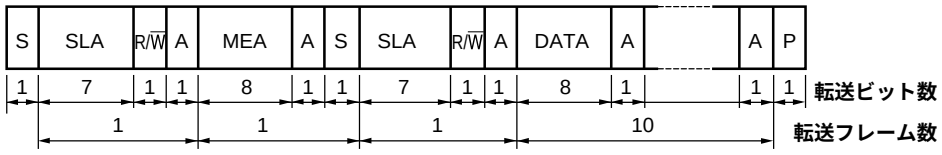


図 4.23　H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.24 に示します。



【記号説明】
S　　**：開始条件**
SLA　　：EEPROM**スレーブアドレス**
R/$\overline{W}$　　**：送信／受信方向**
A　　**：アクノリッジ**
MEA　　：EEPROM**メモリアドレス**
DATA　　**：受信データ**
P　　**：停止条件**

図 4.24　**本タスク例で使用する転送フォーマット**

- 以下に本タスク例で使用する H8S/2138 シリーズのデータトランスファコントローラ（DTC）の使用例について説明します。

1. I²C バスインタフェースのチャネル 0 の割り込み要求（IICI0）により DTC を起動し、送信データの転送を行います。
2. DTC の転送モードはノーマルモードで使用します。
3. **図** 4.25 に本タスク例で使用する DTC のブロック図を示します。

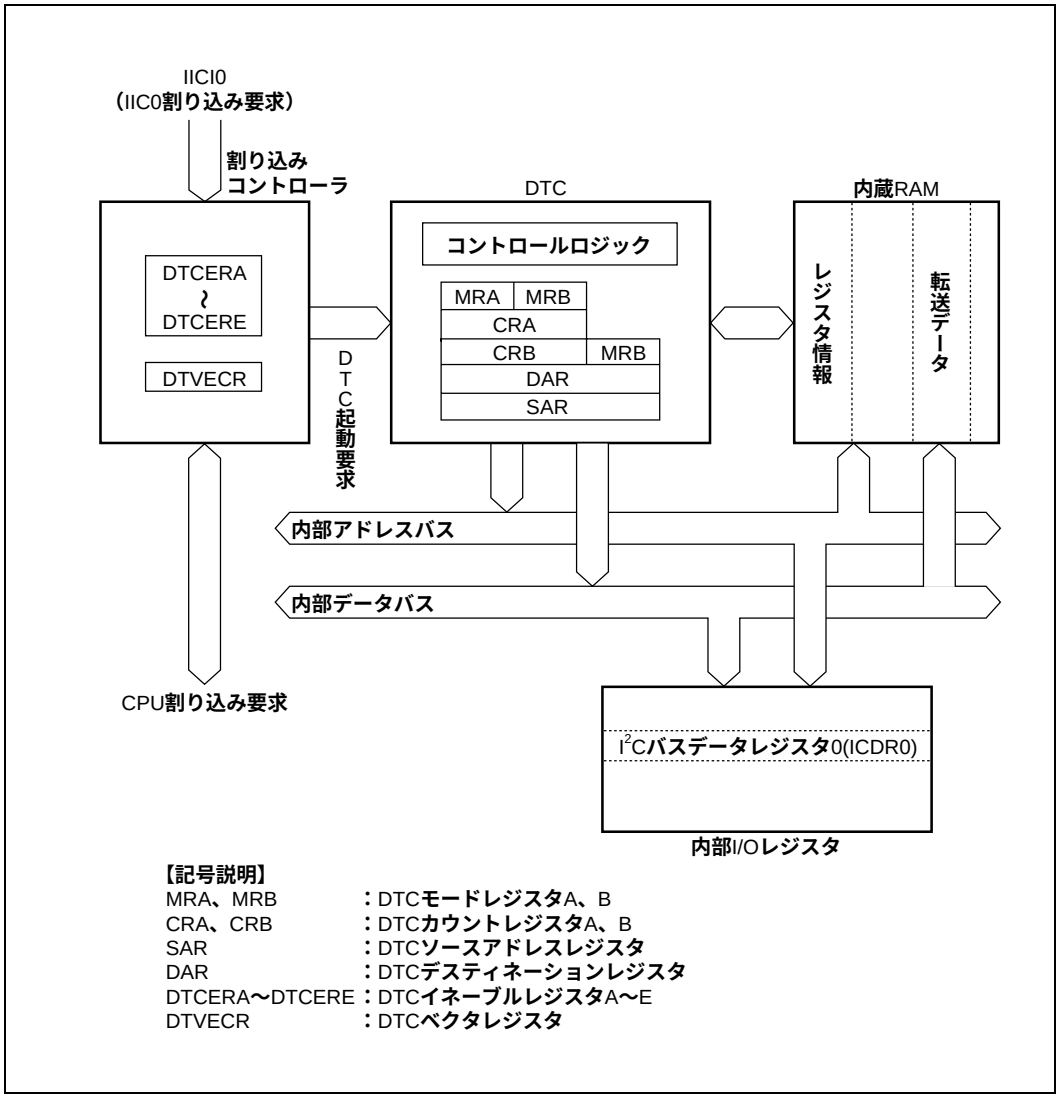


図 4.25 **本タスク例における** DTC **ブロック図**

4. 図 4.26 に内蔵 RAM 上の転送データの配置を示します。

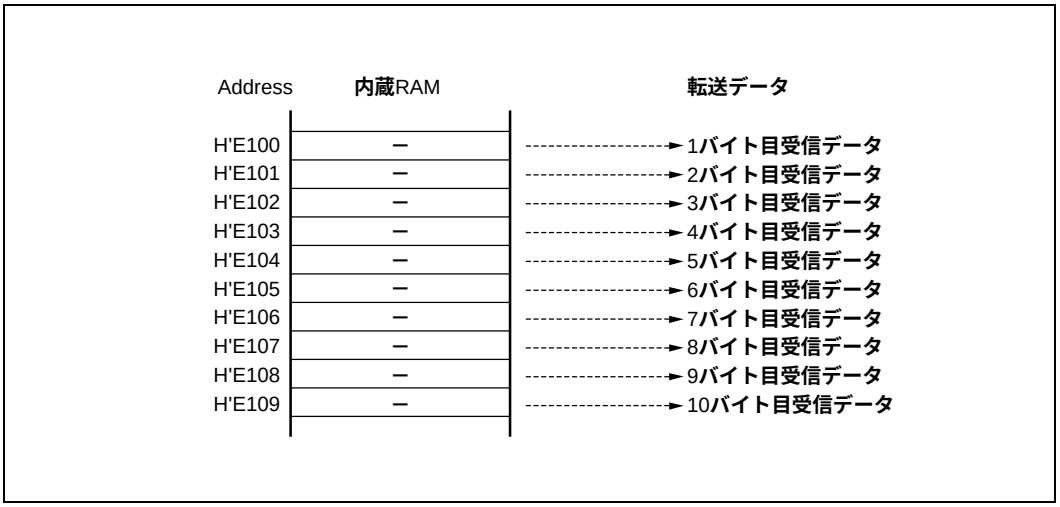


図 4.26 内蔵 RAM 上の転送データの配置

5. 図 4.27 に本タスク例における DTC ベクタテーブルと内蔵 RAM 上のレジスタ情報の配置を示します。
H'EC00 番地から MRA、SAR、MRB、DAR、CRA、CRB の順に DTC のレジスタ情報を配置します。

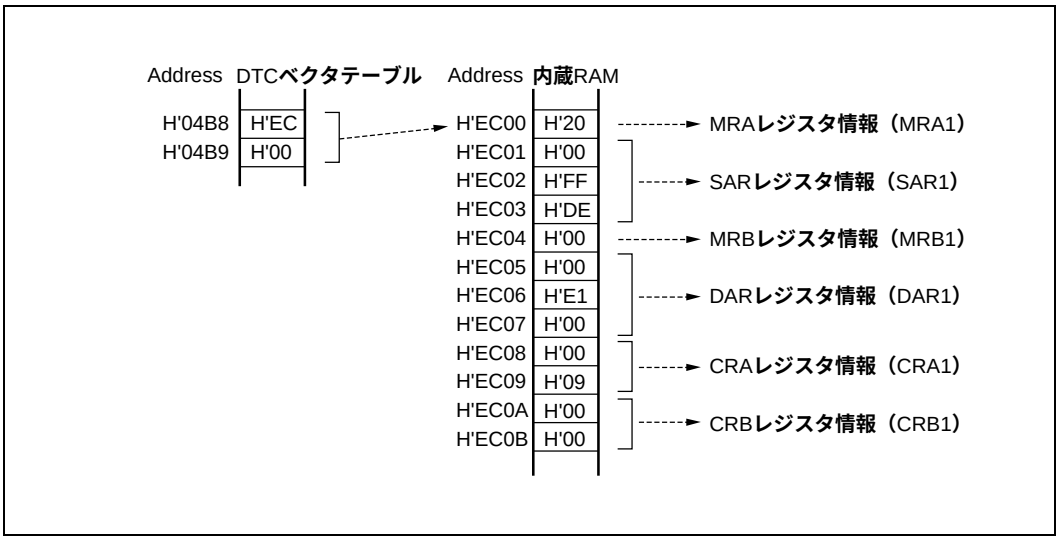


図 4.27 DTC ベクタテーブルと内蔵 RAM 上のレジスタ情報の配置

6. 表 4.21 に本タスク例で使用する DTC のレジスタ説明を示します。

表 4.21 DTC のレジスタ説明

レジスタ		機能
MRA		DTC モードレジスタ A ～DTC の動作モードの制御を行います。
	SM1, 0 （bit7、6）	ソースアドレスモード 1、0 ～データ転送後に、SAR をインクリメントするか、デクリメントするか、または固定とするかを指定します。 • SM1=0、SM0=*のとき、SAR は固定（*：“0”または“1”） • SM1=1、SM0=0 のとき、SAR は転送後インクリメント（Sz=0 のとき+1、Sz=1 のとき+2） • SM1=1、SM0=1 のとき、SAR は転送後デクリメント（Sz=0 のとき-1、Sz=1 のとき-2）
	DM1, 0 （bit5、4）	デスティネーションアドレスモード 1、0 ～データ転送後に、DAR をインクリメントするか、デクリメントするか、または固定とするかを指定します。 • DM1=0、DM0=*のとき、DAR は固定（*：“0”または“1”） • DM1=1、DM0=0 のとき、DAR は転送後インクリメント（Sz=0 のとき+1、Sz=1 のとき+2） • DM1=1、DM0=1 のとき、DAR は転送後デクリメント（Sz=0 のとき-1、Sz=1 のとき-2）
	MD1, 0 （bit3、2）	DTC モード 1、0 ～DTC の転送モードを指定します。 • MD1=0 ひ MD0=0 のとき、ノーマルモード • MD1=0、MD0=1 のとき、リピートモード • MD1=1、MD0=0 のとき、ブロック転送モード • MD1=1、MD0=1 のとき、設定禁止
	DTS （bit1）	DTC 転送モードセレクト ～リピートモードまたはブロック転送モードのとき、ソース側とデスティネーション側のいずれをリピート領域またはブロック領域とするかを指定します。 • DTS=0 のとき、デスティネーション側がリピート領域またはブロック領域 • DTS=1 のとき、ソース側がリピート領域またはブロック領域
	Sz （bit0）	DTC データトランスファサイズ ～データ転送のデータサイズを指定します。
MRB		DTC モードレジスタ B ～DTc モードの制御を行います。
	CHEN （bit7）	DTC チェイン転送イネーブル ～チェイン転送を指定します。 • CHEN=0 のとき、DTC データ転送終了 • CHEN=1 のとき、DTC チェイン転送

表 4.21 DTC のレジスタ説明（続き）

レジスタ		機能
MRB	DISEL （bit6）	DTC インタラプトセレクト 〜1 回のデータ転送後に CPU への割り込み要求の禁止または許可を指定します。 • DISEL=0 のとき、DTC データ転送終了後、転送カウンタが 0 でなければ、CPU への割り込みを禁止 • DISEL=1 のとき、DTC データ転送終了後、CPU への割り込みを許可
SAR		DTC ソースアドレスレジスタ 〜DTC の転送するデータの転送元アドレスを指定します。
DAR		DTC デスティネーションアドレスレジスタ 〜DTC の転送するデータの転送元アドレスを指定します。
CRA		DTC 転送カウントレジスタ A 〜DTC のデータ転送の転送回数を指定します。
CRB		DTC 転送カウントレジスタ B 〜ブロック転送モードのとき、DTC のブロックデータ転送の転送回数を指定します。
DTVECR（H'FEF3）		DTC ベクタレジスタ 〜ソフトウェアによる DTC 起動の許可または禁止の設定、およびソフトウェア起動割り込み用ベクタ番号を設定します。
	SWDTE （bit7）	DTC ソフトウェア起動イネーブル 〜DTC ソフトウェア起動の許可または禁止を設定します。 • SWDTE=0 のとき、DTC ソフトウェア起動を禁止 • SWDTE=1 のとき、DTC ソフトウェア起動を許可
	DTVEC6-0 （bit6-0）	DTC ソフトウェア起動ベクタ 6〜0 〜DTC ソフトウェア起動のベクタ番号を設定します。
DTCERD（H'FEF1）		DTC イネーブルレジスタ 〜各割り込み要因による DTC 起動の許可または禁止を制御します。
	DTCED4 （bit4）	DTC 起動イネーブル D4 • DTCED4=0 のとき、IICI0 割り込みによる DTC 起動を禁止 • DTCED4=2 のとき、IICI0 割り込みによる DTC 起動を許可

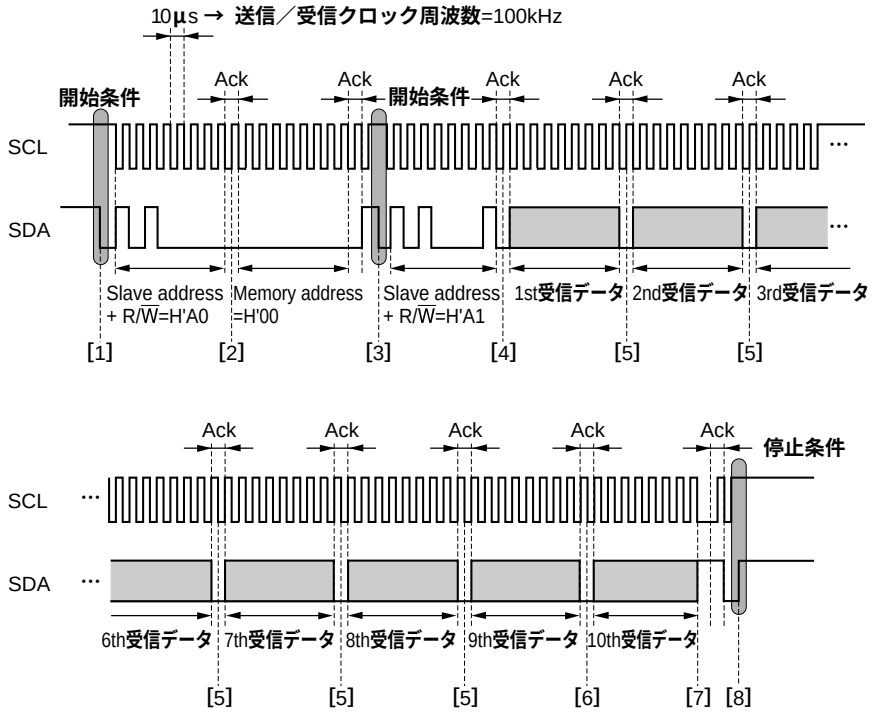
7. I²C バスフォーマットでは、スレーブアドレスと R/$\overline{W}$ ビットによるスレーブデバイスおよび転送方向の選択や、アクノリッジビットによる受信の確認および最終フレームの表示などが行われるため、DTC によるデータの連続転送は、割り込みによる CPU 処理と組み合わせて行う必要があります。**表** 4.22 に本タスク例における DTC を利用したマスタ送信モードにおける処理の例を示します。

表 4.22　DTC **による動作例（マスタ送信モード）**

項目	マスタ送信モード
スレーブアドレス +R/$\overline{W}$ **ビット送信**	CPU **で送信（**ICDR **ライト）**
ダミーデータリード	CPU **で処理（**ICDR **ライト）**
本体データ送信	DTC **で受信（**ICDR **ライト）**
ダミーデータ（H'FF**）ライト**	−
最終フレーム処理	**不要**
最終フレーム処理後の転送要求処理	**不要**
DTC **転送データフレーム数設定**	**受信：実データ数**

4.7.2 動作説明

図 4.28 に動作原理を示します。



	CPU処理	DTC処理
[1]	IRIC=0、開始条件の発行（BBSY=1、SCP=0） スレーブアドレス+Wデータ送信、IRIC=0	処理なし
[2]	EEPROMメモリアドレス送信、IRIC=0	処理なし
[3]	IRIC=0、開始条件の発行（BBSY=1、SCP=0） スレーブアドレス+Rデータ送信、IRIC=0	処理なし
[4]	ダミーリード、IRIC=0、IEIC=1	処理なし
[5]	処理なし	受信データのリード、IRIC=0
[6]	IEIC=0、ACKB=1、WAIT=1 9バイト目受信データのリード、IRIC=0	処理なし
[7]	TRS=1 10バイト目受信データリード、IRIC=0	処理なし
[8]	停止条件の発行（BBSY=0、SCP=0）	処理なし

図 4.28　DTC **によるシングルマスタの受信動作原理**

4.7.3 ソフトウェア説明

(1) モジュール説明

表 4.23 に本タスク例におけるモジュール説明を示します。

表 4.23 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	使用 RAM 領域、IIC0、DTC の初期設定
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス+W データ（H'A0）の送信
Slave address + R 送信	trs_slvadr_a1	EEPROM のスレーブアドレス+R データ（H'A1）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信
IIC0 割り込み処理	iici0	IRIC のクリア、IICI0 割り込み要求の禁止、受信終了判定フラグのセット

(2) 使用内蔵レジスタ説明

表 4.24 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.24 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		受信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジ間にウェイトを挿入するか否かを設定	H'FFDF bit6	0/1
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0/1

表 4.24 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	MST	I^2C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I^2C バスインタフェースを送信／受信モードの設定	H'FFD8 bit4	0/1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ転送の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM から受信したアクノリッジを格納 受信時、EEPROM へ送信するアクノリッジを設定	H'FFD9 bit0	-
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1
MRA	SM1、0	データ転送後、SAR を固定に設定	H'EC00 bit7、6	SM1=0 SM0=0
	DM1、0	データ転送後、DAR をインクリメントに設定	H'EC00 bit5、4	DM1=1 DM0=0
	MD1、0	DTC の転送モードをノーマルモードに設定	H'EC00 bit3、2	MD1=0 MD0=0

表 4.24 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
MRA	DTS	デスティネーション側がリピート領域またはブロック領域に設定	H'EC00 bit1	DTS=0
	Sz	データ転送のデータサイズをバイトサイズに設定	H'EC00 bit0	Sz=0
MRB	CHNE	DTC チェイン転送をディスエーブルに設定	H'EC04 bit7	CHNE=0
	DISEL	1 回のデータ転送後、転送カウンタが 0 でなければ CPU への割り込みを禁止に設定	H'EC04 bit6	DISEL=0
SAR		DTC の転送するデータの転送元アドレスを H'FFDE に設定	H'EC01	H'00FFDE
DAR		DTC の転送するデータの転送先アドレスを H'E100 に設定	H'EC05	H'00E100
CRA		DTC のデータ転送の転送回数を 12 回に設定	H'EC08	H'000C
CRB		ブロック転送モード時の DTC のブロックデータ転送の転送回数を 0 回に設定	H'EC0A	H'0000
DTVECR	SWDTE	DTC ソフトウェア起動の禁止を設定	H'FEF3 bit7	0
	DTVEC6 to DTVEC0	DTC ソフトウェア起動のベクタ番号を H'00 に設定	H'FEF3 bit6 to bit0	H'00
DTCERD	DTCED4	IICI0 割り込みによる DTC 起動を許可	H'FEF1 bit4	1
MSTPCRH	MSTP14	DTC のモジュールストップモードの解除	H'FF86 bit6	0

(3) 変数説明

表 4.25 に本タスク例における変数説明を示します。

表 4.25 変数説明

変数	機能	データ長	初期値	使用モジュール名
dummy	MDCR リード値	1 バイト	−	Main
i	受信データカウンタ	1 バイト	H'00	Initialize

(4) 使用 RAM 説明

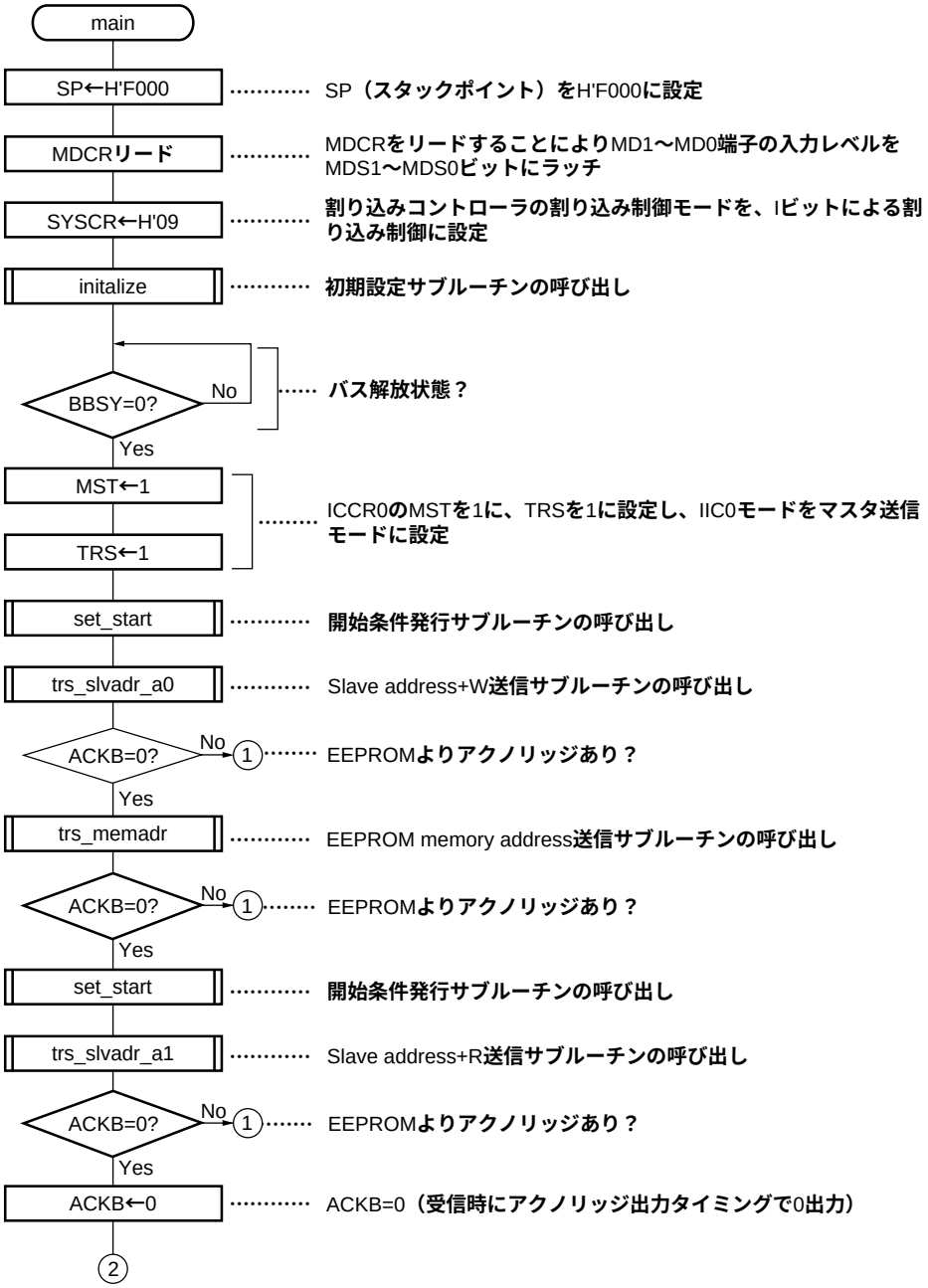
表 4.26 に本タスク例における使用 RAM 説明を示します。

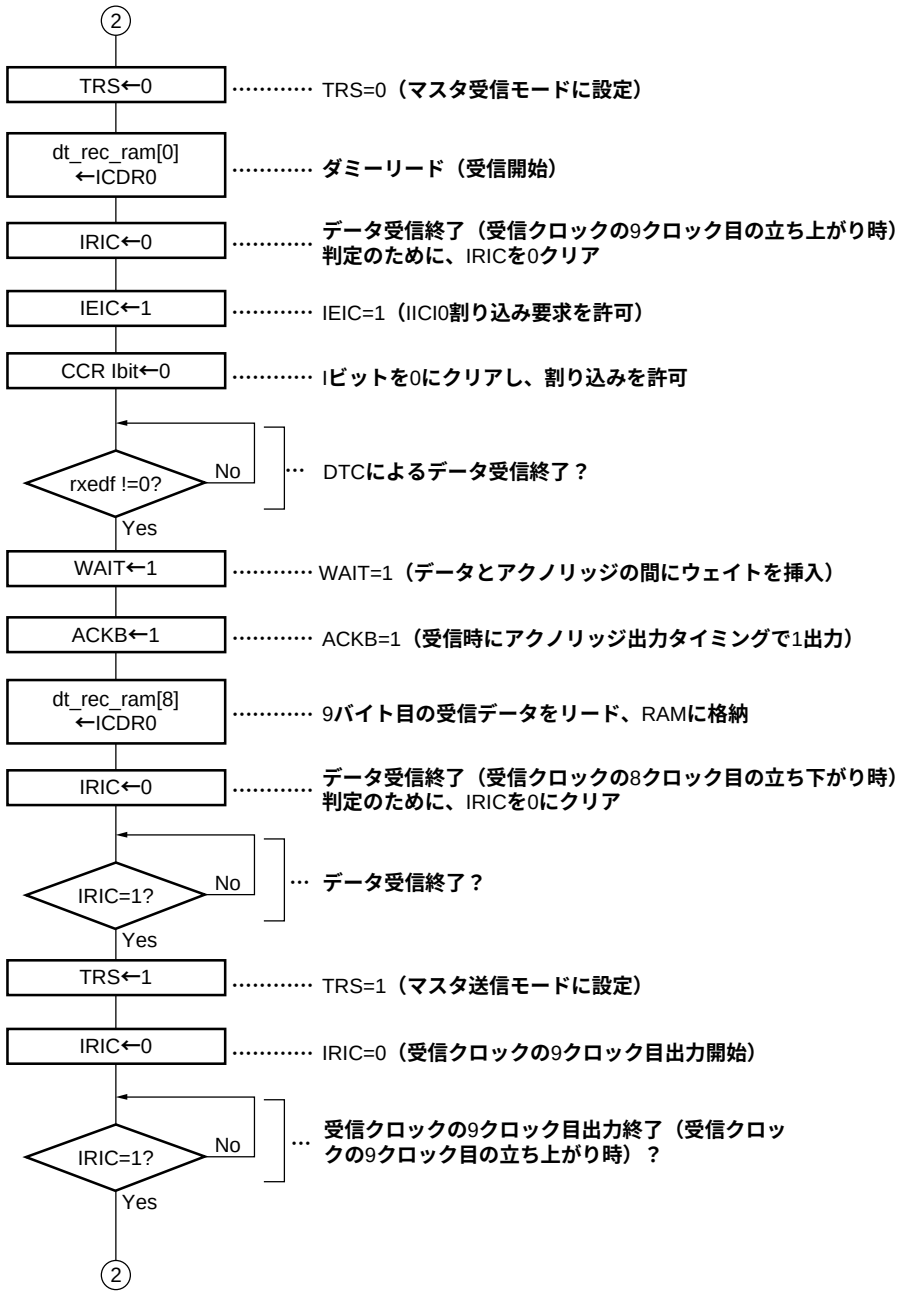
表 4.26 使用 RAM 説明

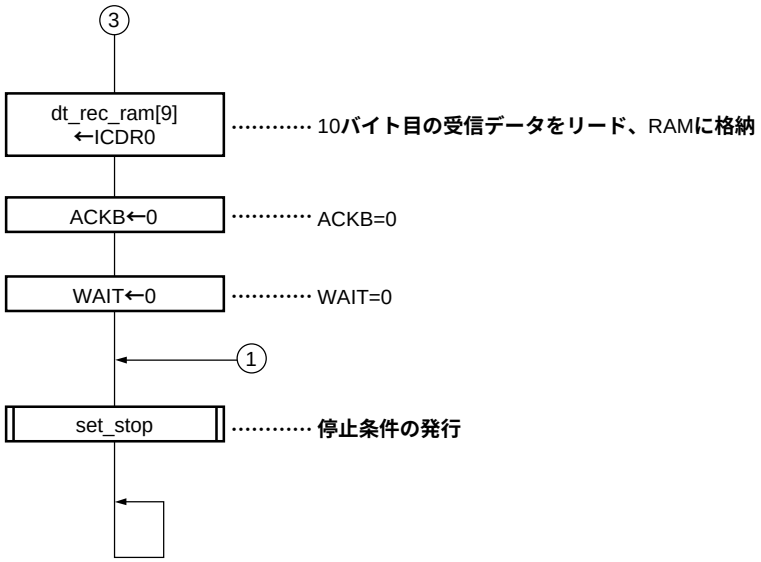
ラベル	機能	データ長	アドレス	使用モジュール名
MRA1	DTC モードレジスタ A（MRA）	1 バイト	H'EC00	initialize
SAR1	DTC ソースアドレスレジスタ（SAR）	4 バイト	H'EC00	initialize
MRB1	DTC モードレジスタ B（MRB）	1 バイト	H'EC04	initialize
DAR1	DTC ディスティネーションアドレスレジスタ（DAR）	4 バイト	H'EC04	initialize
CRA1	DTC 転送カウントレジスタ A（CRA）	2 バイト	H'EC08	initialize
CRB1	DTC 転送カウントレジスタ B（CRB）	2 バイト	H'EC0A	initialize
rxedf	送信終了判定フラグ	1 バイト	H'E200	main iici0
dt_rec_ram [0] to dt_rec_ram [9]	10 バイトの送信データを格納	10 バイト	H'E100 to H'E109	main initialize

4.7.4 フローチャート

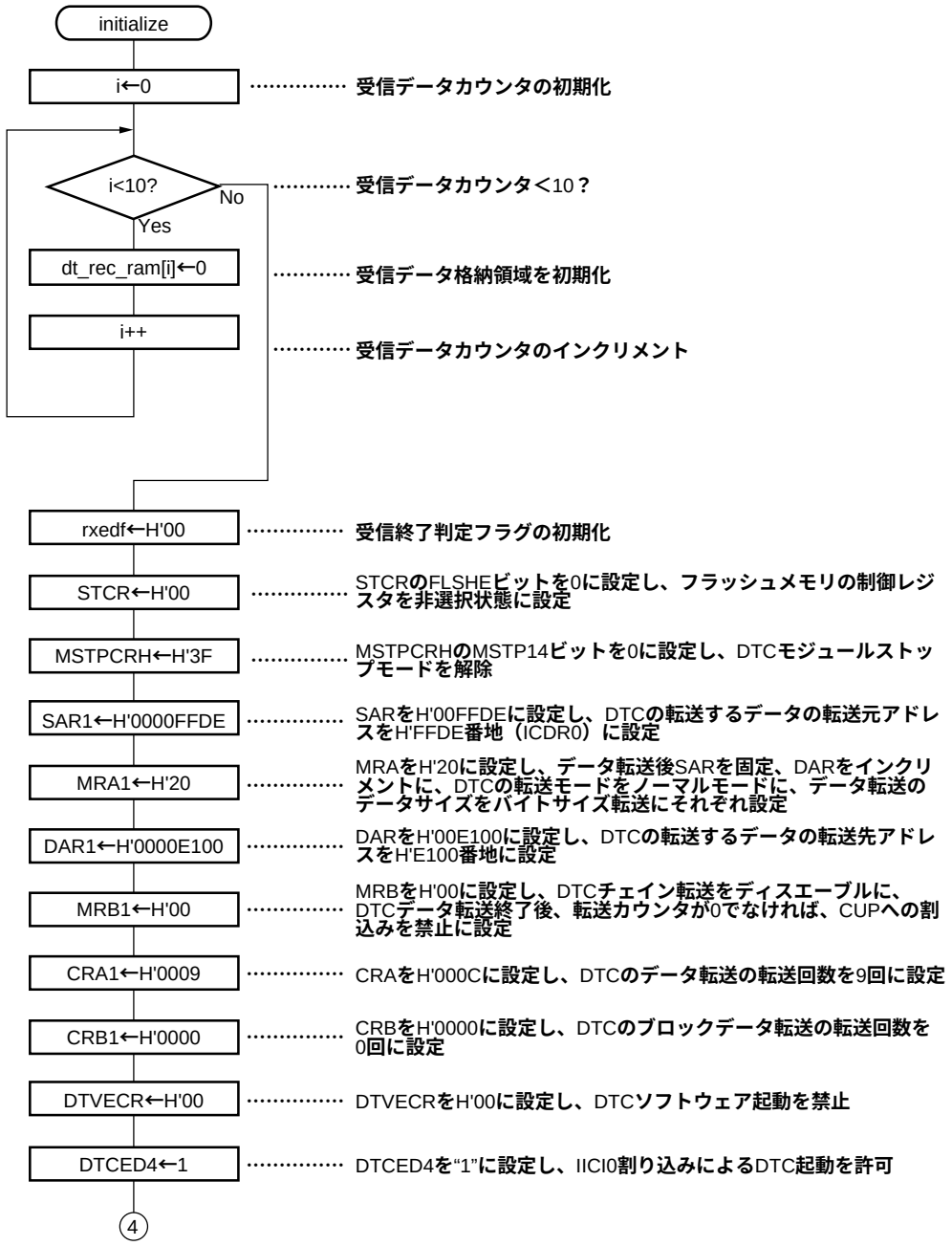
（1） メインルーチン

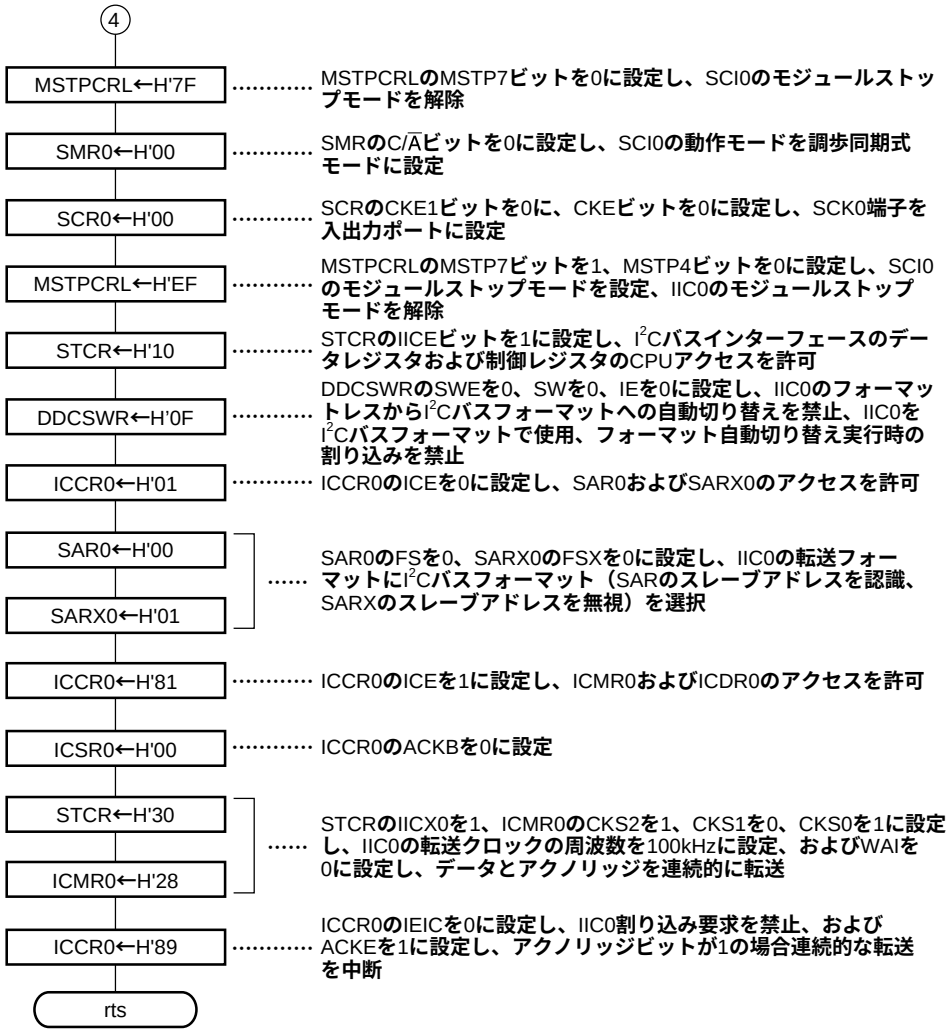



2
TRS←0
TRS=0（マスタ受信モードに設定）
dt_rec_ram[0]
←ICDR0
ダミーリード（受信開始）
IRIC←0
データ受信終了（受信クロックの9クロック目の立ち上がり時）
判定のために、IRICを0クリア
IEIC←1
IEIC=1（IICI0割り込み要求を許可）
CCR Ibit←0
Iビットを0にクリアし、割り込みを許可
rxedf !=0?
No
DTCによるデータ受信終了？
Yes
WAIT←1
WAIT=1（データとアクノリッジの間にウェイトを挿入）
ACKB←1
ACKB=1（受信時にアクノリッジ出力タイミングで1出力）
dt_rec_ram[8]
←ICDR0
9バイト目の受信データをリード、RAMに格納
IRIC←0
データ受信終了（受信クロックの8クロック目の立ち下がり時）
判定のために、IRICを0にクリア
IRIC=1?
No
データ受信終了？
Yes
TRS←1
TRS=1（マスタ送信モードに設定）
IRIC←0
IRIC=0（受信クロックの9クロック目出力開始）
IRIC=1?
No
受信クロックの9クロック目出力終了（受信クロックの9クロック目の立ち上がり時）？
Yes
2

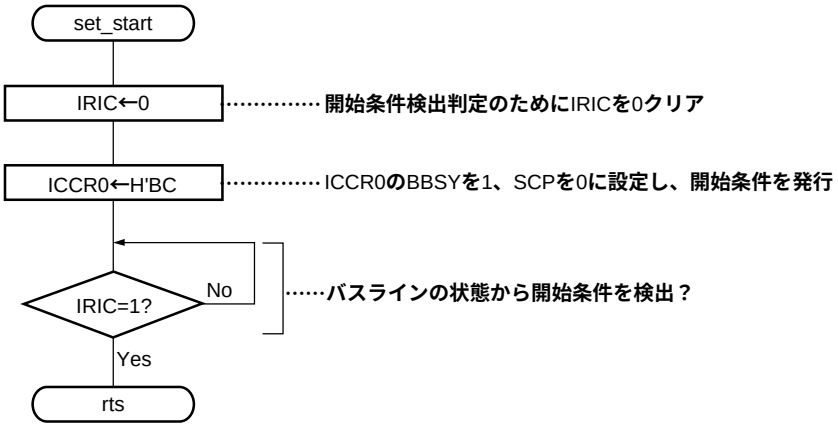

3
dt_rec_ram[9]
←ICDR0
10バイト目の受信データをリード、RAMに格納
ACKB←0
ACKB=0
WAIT←0
WAIT=0
1
set_stop
停止条件の発行

(2) 初期設定サブルーチン

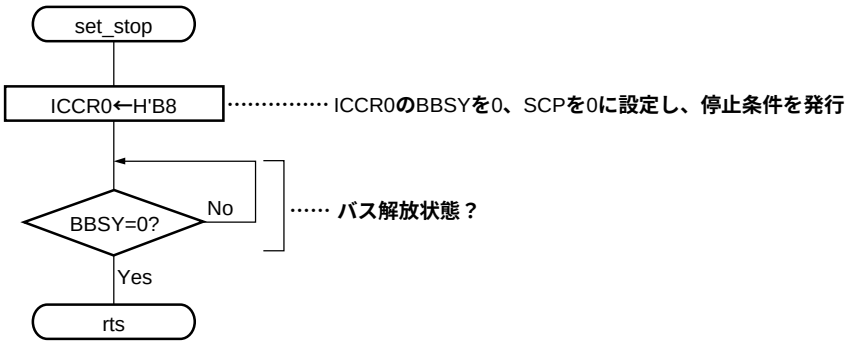



4
MSTPCRL←H'7F
MSTPCRLのMSTP7ビットを0に設定し、SCI0のモジュールストップモードを解除
SMR0←H'00
SMRのC/Aビットを0に設定し、SCI0の動作モードを調歩同期式モードに設定
SCR0←H'00
SCRのCKE1ビットを0に、CKEビットを0に設定し、SCK0端子を入出力ポートに設定
MSTPCRL←H'EF
MSTPCRLのMSTP7ビットを1、MSTP4ビットを0に設定し、SCI0のモジュールストップモードを設定、IIC0のモジュールストップモードを解除
STCR←H'10
STCRのIICEビットを1に設定し、I2Cバスインターフェースのデータレジスタおよび制御レジスタのCPUアクセスを許可
DDCSWR←H'0F
DDCSWRのSWEを0、SWを0、IEを0に設定し、IIC0のフォーマットレスからI2Cバスフォーマットへの自動切り替えを禁止、IIC0をI2Cバスフォーマットで使用、フォーマット自動切り替え実行時の割り込みを禁止
ICCR0←H'01
ICCR0のICEを0に設定し、SAR0およびSARX0のアクセスを許可
SAR0←H'00
SARX0←H'01
SAR0のFSを0、SARX0のFSXを0に設定し、IIC0の転送フォーマットにI2Cバスフォーマット（SARのスレーブアドレスを認識、SARXのスレーブアドレスを無視）を選択
ICCR0←H'81
ICCR0のICEを1に設定し、ICMR0およびICDR0のアクセスを許可
ICSR0←H'00
ICCR0のACKBを0に設定
STCR←H'30
ICMR0←H'28
STCRのIICX0を1、ICMR0のCKS2を1、CKS1を0、CKS0を1に設定し、IIC0の転送クロックの周波数を100kHzに設定、およびWAIを0に設定し、データとアクノリッジを連続的に転送
ICCR0←H'89
ICCR0のIEICを0に設定し、IIC0割り込み要求を禁止、およびACKEを1に設定し、アクノリッジビットが1の場合連続的な転送を中断
rts

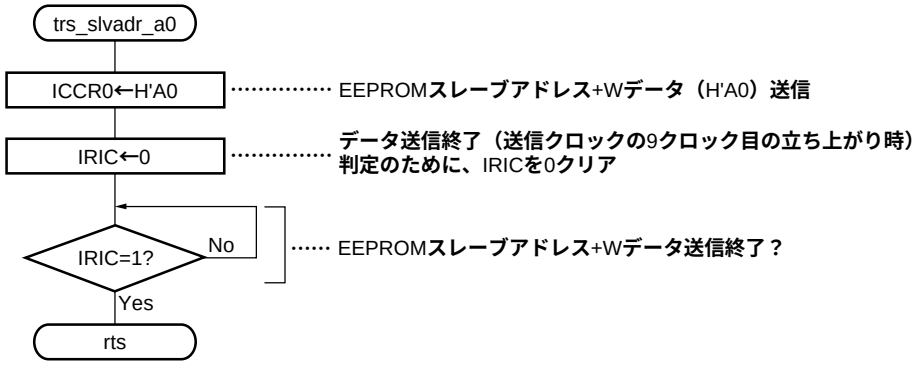
(3) 開始条件発行サブルーチン



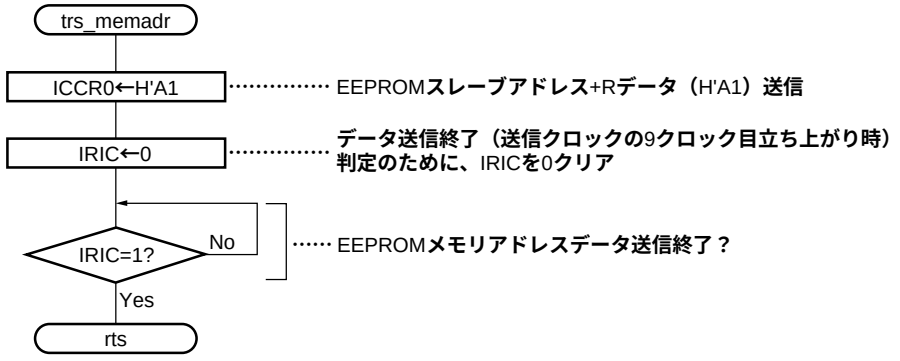
(4) 停止条件発行サブルーチン



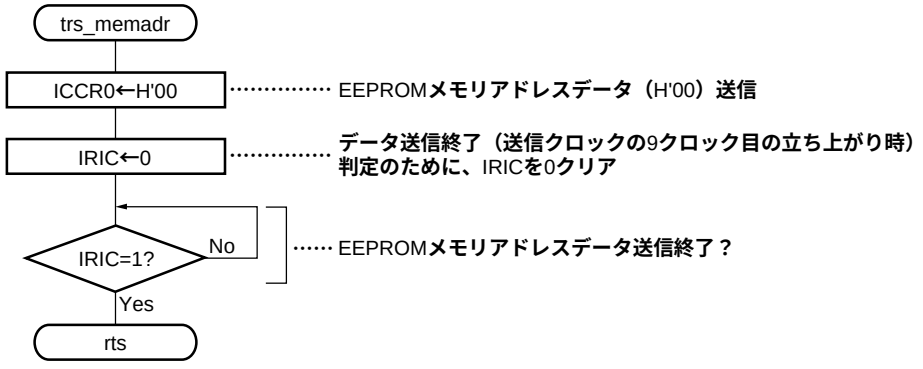
（5） Slave address + W 送信サブルーチン



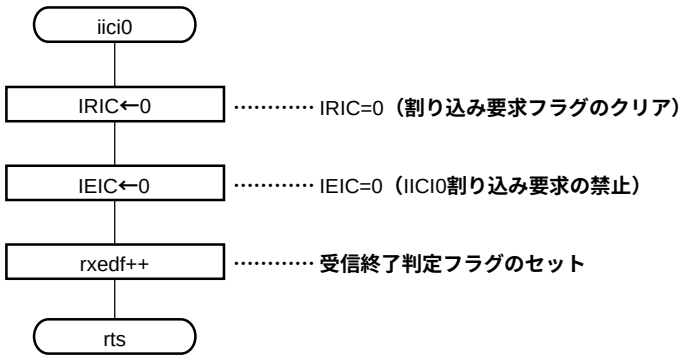
（6） Slave address + R 送信サブルーチン



（7） EEPROM memory address 送信サブルーチン



（8） IIC0 割り込み処理ルーチン



4.7.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note             *
*      6.Single master receive by DTC            *
*                          File name   :  DTCrx.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* RAM & DTC & IIC0 initialize */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_slvadr_a1(void);                        /* Slave address + R data transmit */
void trs_memadr(void);                           /* EEPROM memory address data transmit */

/************************************************
*  RAM allocation                                *
************************************************/
#define MRA1 (*(volatile unsigned char *)0xec00)  /* DTC mode register A */
#define SAR1 (*(volatile unsigned long *)0xec00)  /* DTC source address register */
#define MRB1 (*(volatile unsigned char *)0xec04)  /* DTC mode register B */
#define DAR1 (*(volatile unsigned long *)0xec04)  /* DTC destination address register */
#define CRA1 (*(volatile unsigned short *)0xec08) /* DTC transfer count register A */
#define CRB1 (*(volatile unsigned short *)0xec0a) /* DTC transfer count register B */

#define rxedf (*(volatile unsigned char *)0xe200) /* Receive end flag */

#pragma section ramerea
```

（続く）

```
unsigned char dt_rec_ram[10];                    /* Receive data store area */
#pragma section

/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                        ;Stack pointer initialize
#pragma endasm
{
   unsigned char dummy;
   dummy = MDCR.BYTE;                            /* MCU mode set */
   SYSCR.BYTE = 0x09;                            /* Interrupt control mode set */
   initialize();                                 /* Initialize */

   while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */
   IIC0.ICCR.BIT.MST = 1;                        /* Master transmit mode set */
   IIC0.ICCR.BIT.TRS = 1;                        /* MST=1, TRS=1 */
   set_start();                                  /* Start condition set */
   trs_slvadr_a0();                              /* Slave address + W data transmit */
   if(IIC0.ICSR.BIT.ACKB == 0)                   /* ACKB = 0 ? */
   {
      trs_memadr();                              /* EEPROM memory address data transmit */
      if(IIC0.ICSR.BIT.ACKB == 0)                /* ACKB = 0 ? */
      {
         set_start();                            /* Re-start condition set */
         trs_slvadr_a1();                        /* Slave address + R data transmit */
         if(IIC0.ICSR.BIT.ACKB == 0)             /* ACKB = 0 ? */
         {
            IIC0.ICSR.BIT.ACKB = 0;              /* ACKB = 0 */
            IIC0.ICCR.BIT.TRS = 0;               /* Master receive mode set */

            dt_rec_ram[0] = IIC0.ICDR;           /* Dummy read */

            IIC0.ICCR.BIT.IRIC = 0;              /* IRIC = 0 */
            IIC0.ICCR.BIT.IEIC = 1;              /* IEIC = 1 (IICI0 interrupt enable) */
```

（続く）

```
                set_imask_ccr(0);                   /* Interrupt enable */

                while(rxedf == 0x00);               /* rxedf != 0 ? */

                IIC0.ICMR.BIT.WAIT = 1;             /* WAIT = 1 */
                IIC0.ICSR.BIT.ACKB = 1;             /* ACKB = 1 */

                dt_rec_ram[8] = IIC0.ICDR;          /* 9th receive data read */
                IIC0.ICCR.BIT.IRIC = 0;             /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);     /* Receieve end (IRIC=1) ? */

                IIC0.ICCR.BIT.TRS = 1;              /* Master transmit mode set */

                IIC0.ICCR.BIT.IRIC = 0;             /* 9th clock transmit (IRIC=0) */
                while(IIC0.ICCR.BIT.IRIC == 0);     /* 9th clock transmit end (IRIC=1) ? */

                dt_rec_ram[9] = IIC0.ICDR;          /* 10th (last) receive data read */

                IIC0.ICSR.BIT.ACKB = 0;             /* ACKB = 0 */
                IIC0.ICMR.BIT.WAIT = 0;             /* WAIT = 0 */
            }
        }
    }
    set_stop();                                     /* Stop condition set */

    while(1);                                       /* End */
}

/************************************************
*  initialize : RAM & IIC0 Initialize             *
************************************************/
void initialize(void)
{
    unsigned char i;                                /* Receive data counter */

    for(i=0; i<10; i++)                             /* Receive data store area initialize */
    {
```

（続く）

```
        dt_rec_ram[i] = 0x00;
    }

    rxedf = 0x00;                                   /* Receive end flag initialize */

    STCR.BYTE = 0x00;                               /* FLSHE = 0 */

    MSTPCR.BYTE.H = 0x3f;                           /* DTC module stop mode reset */
    SAR1 = 0x0000ffde;                              /* SAR = H'00FFDE (ICDR0) */
    MRA1 = 0x20;                                    /* MRA = H'20 */
    DAR1 = 0x0000e100;                              /* DAR = H'00E100 */
    MRB1 = 0x00;                                    /* MRB = H'00 */
    CRA1 = 0x0009;                                  /* CRA = H'0009 */
    CRB1 = 0x0000;                                  /* CRB = H'0000 */
    DTC.VECR.BYTE = 0x00;                           /* SWDTE = 0, DTVEC = H'00 */
    DTC.ED.BIT.B4 = 1;                              /* DTCED4 = 1 */

    MSTPCR.BYTE.L = 0x7f;                           /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                           /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                           /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                               /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                             /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                          /* ICE = 0 */
    IIC0.SAR.BYTE = 0x00;                           /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                          /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                          /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                          /* ACKB = 0 */
    STCR.BYTE = 0x30;                               /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                          /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                          /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  set_start : Start condition set              *
************************************************/
void set_start(void)
```

（続く）

```
{
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                          /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Start condition set (IRIC=1) ? */
}

/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
    IIC0.ICCR.BYTE = 0xb8;                          /* Stop condition set (BBSY=0,SCP=0) */
    while(IIC0.ICCR.BIT.BBSY == 1);                 /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave address + W data transmit*
************************************************/
void trs_slvadr_a0(void)
{
    IIC0.ICDR = 0xa0;                               /* Slave address + W data(H'A0) write */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_slvadr_a1 : Slave address + R data transmit*
************************************************/
void trs_slvadr_a1(void)
{
    IIC0.ICDR = 0xa1;                               /* Slave address + R data(H'A1) write */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_memadr : EEPROM memory address data transmit*
************************************************/
```

```
void trs_memadr(void)
{
    IIC0.ICDR = 0x00;                               /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);                 /* Transmit end (IRIC=1) ? */
}

/************************************************
*  iici0 : IIC0 interrupt routine                *
************************************************/
#pragma interrupt(iici0)
void iici0(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
    IIC0.ICCR.BIT.IEIC = 0;                         /* IEIC = 0 (IICI0 interrupt disable) */
    rxedf++;                                        /* rxedf flag set */
}
```

4.8 スレーブ送信

4.8.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、スレーブ送信モードで H8S/2138 に 10 バイトのデータを送信します。
- スレーブ送信側の H8S/2138 のスレーブアドレスは［0011100］とします。
- 送信データは［H'00, H'11, H'22, H'33, H'44, H'55, H'66, H'77, H'88, H'99］とします。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（H8S/2138）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.29 に H8S/2138 の接続例を示します。

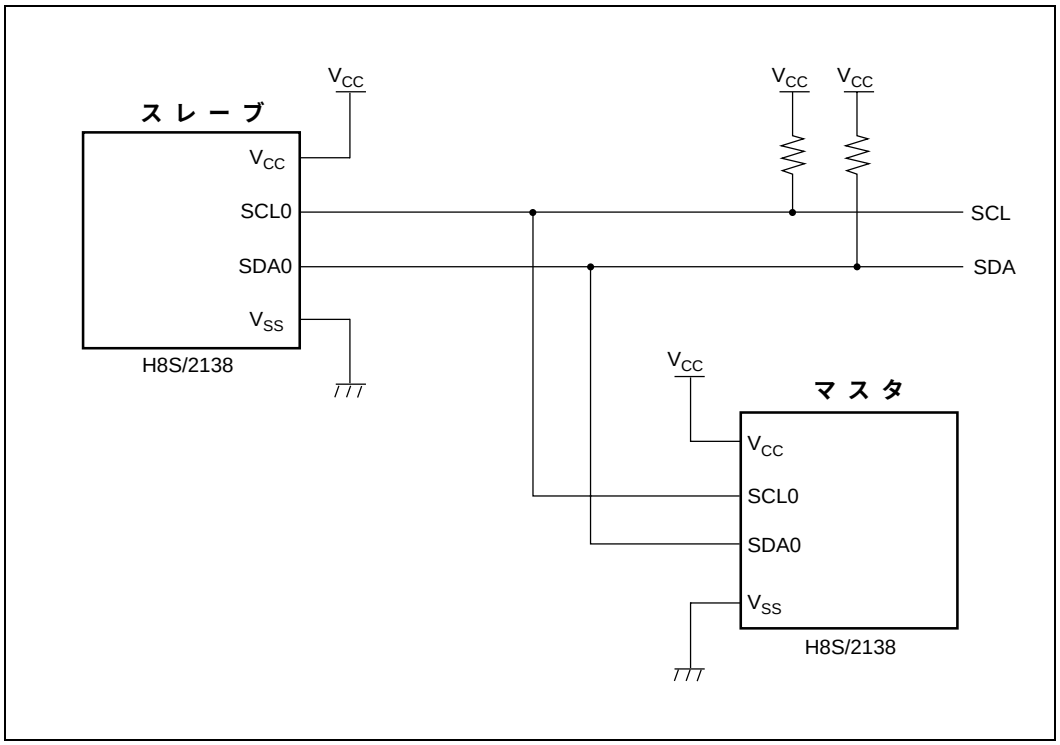


図 4.29　H8S/2138 **の接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.30 に示します。

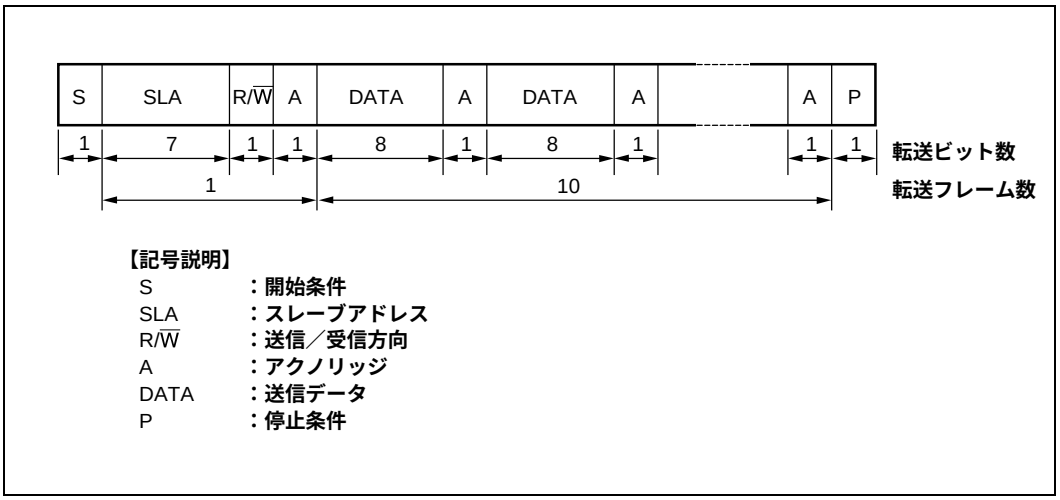
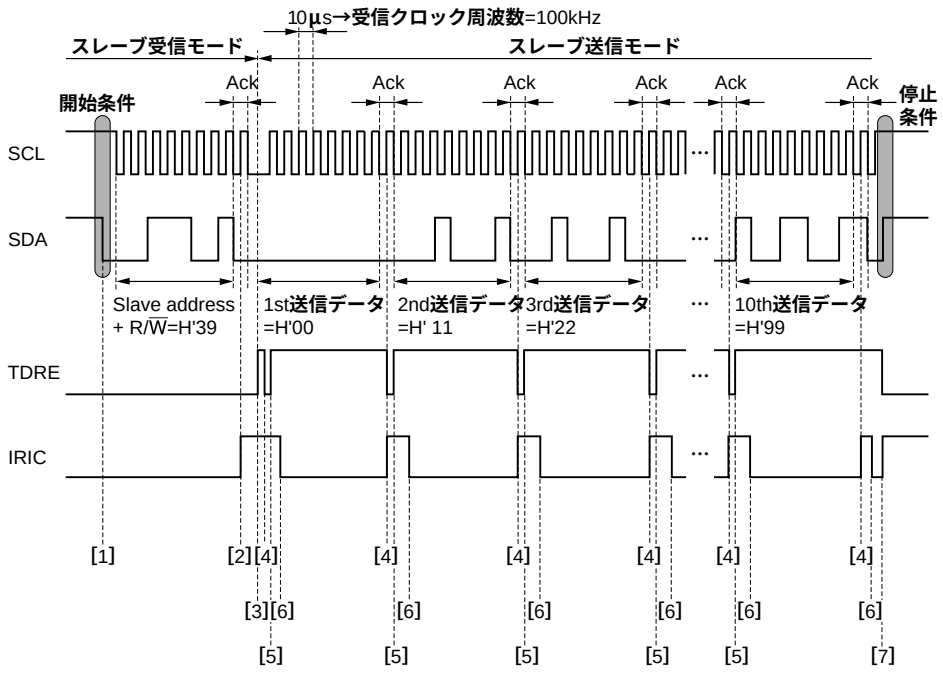


図 4.30 **本タスク例で使用する転送フォーマット**

4.8.2 動作説明

図 4.31 に動作原理を示します。



	ソフトウェア処理	ハードウェア処理
[1]	処理なし	処理なし
[2]	処理なし	IRIC=1(9クロック目の立ち上がり時)
[3]	処理なし	TDRE=1(TRS=0 → TRS=1)
[4]	ICDR0に送信データをライト	TDRE=0(TRS=1の状態でICDRTにデータをライト)
[5]	処理なし	TDRE=1(ICDRTからICDRSにデータを転送)
[6]	IRICを0にクリア	処理なし
[7]	処理なし	IRIC=1、TDRE=0(バスラインの状態から停止条件を検出)

図 4.31 スレーブ送信動作原理

4.8.3 ソフトウェア説明

(1) モジュール説明

表 4.27 に本タスク例におけるモジュール説明を示します。

表 4.27 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	IIC0 の初期設定
スレーブ送信	slv_trs	スレーブ送信による H8S/2138 への 10 バイトデータの送信

(2) 使用内蔵レジスタ説明

表 4.28 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.28 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信データを格納	H'FFDE	ー
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
	SVA6 to SVA0	スレーブアドレスを設定	H'FFDF bit7 to bit1	SVA6=0 SVA5=0 SVA4=1 SVA3=1 SVA2=1 SVA1=0 SVA0=0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジの連続的な転送を設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0

表 4.28 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	MST	I^2C バスインタフェースをスレーブモードで使用	H'FFD8 bit5	1
	TRS	I^2C バスインタフェースを送信モードで使用	H'FFD8 bit4	1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM から受信したアクノリッジを格納 受信時、EEPROM へ送信するアクノリッジを設定	H'FFD9 bit0	-
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1, 0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1, 0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1, 0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5, 4	INTM1=0 INTM0=0
MDCR	MDS1, 0	MD1, 0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1, 0	MDS1=1 MDS0=1

(3) 変数説明

表 4.29 に本タスク例における変数説明を示します。

表 4.29 変数説明

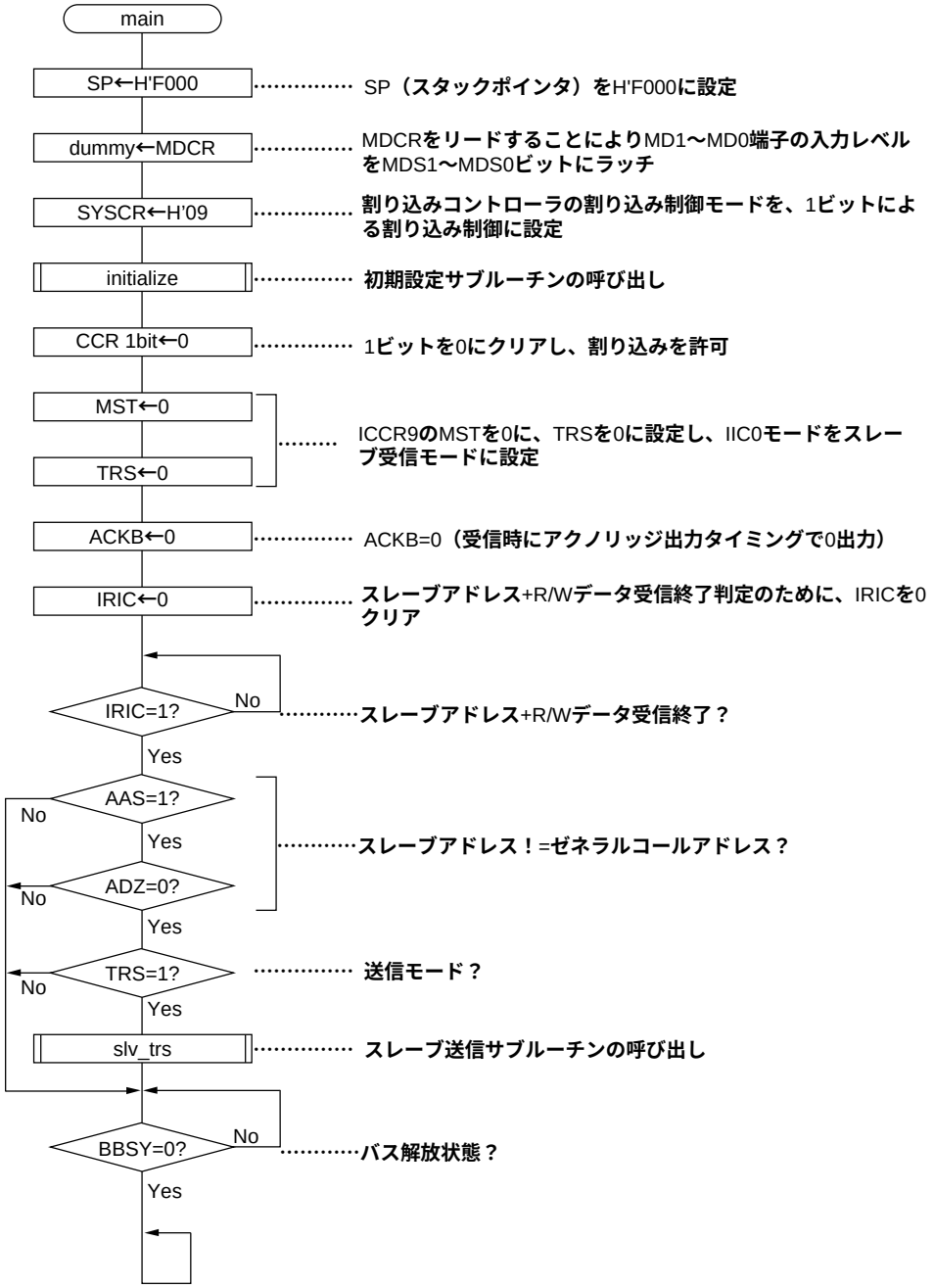
変数	機能	データ長	初期値	使用モジュール名
dt_trs[0]	1 バイト目送信データを格納	1 バイト	H'00	slv_trs
dt_trs[1]	2 バイト目送信データを格納	1 バイト	H'11	slv_trs
dt_trs[2]	3 バイト目送信データを格納	1 バイト	H'22	slv_trs
dt_trs[3]	4 バイト目送信データを格納	1 バイト	H'33	slv_trs
dt_trs[4]	5 バイト目送信データを格納	1 バイト	H'44	slv_trs
dt_trs[5]	6 バイト目送信データを格納	1 バイト	H'55	slv_trs
dt_trs[6]	7 バイト目送信データを格納	1 バイト	H'66	slv_trs
dt_trs[7]	8 バイト目送信データを格納	1 バイト	H'77	slv_trs
dt_trs[8]	9 バイト目送信データを格納	1 バイト	H'88	slv_trs
dt_trs[9]	10 バイト目送信データを格納	1 バイト	H'99	slv_trs
i	送信データカウンタ	1 バイト	H'00	slv_trs
dummy	MDCR リード値	1 バイト	−	main
dmyrd	ダミーリード時の値を格納	1 バイト	−	slv_trs

(4) 使用 RAM 説明

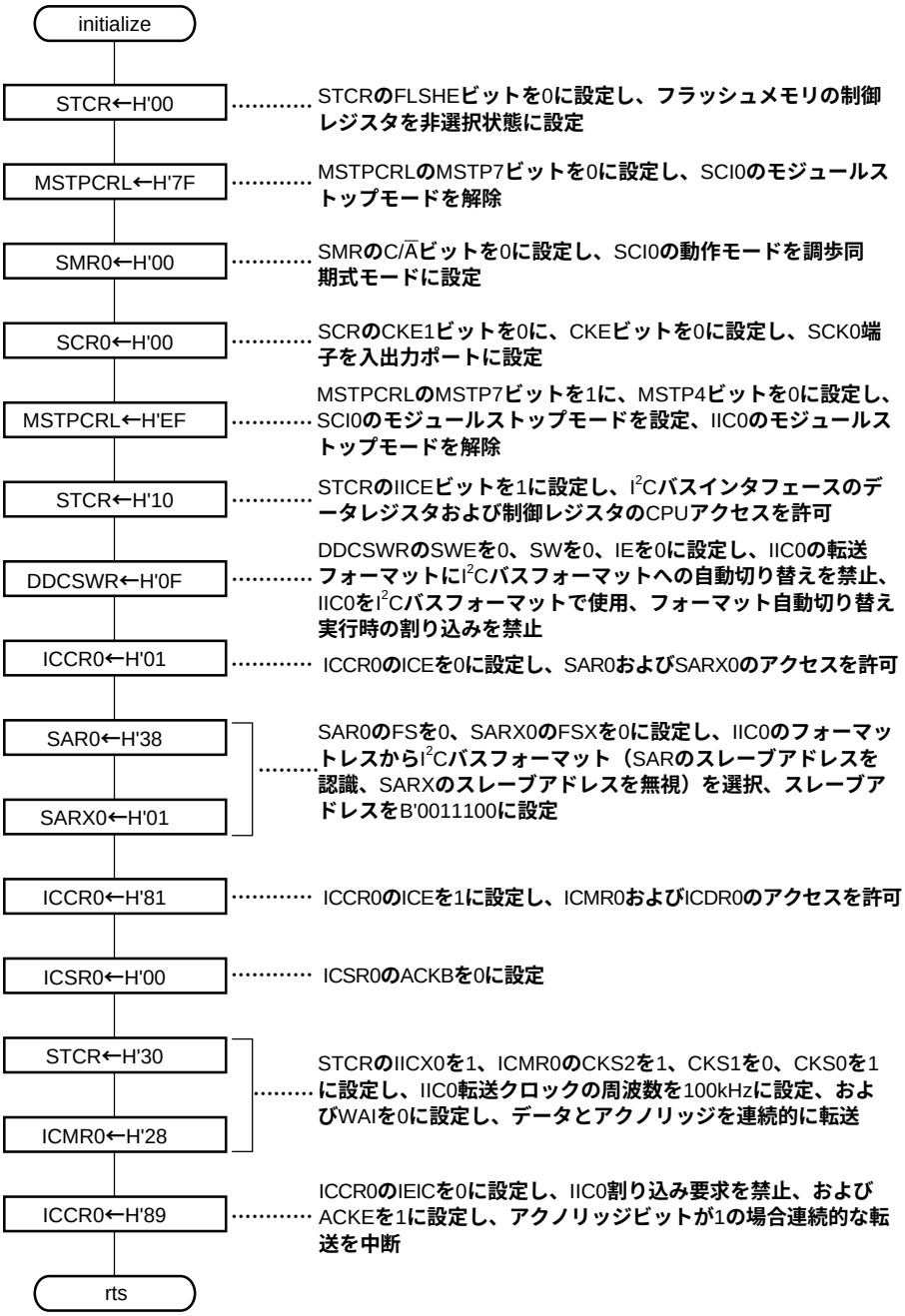
本タスク例では、変数以外の RAM は使用しません。

4.8.4 フローチャート

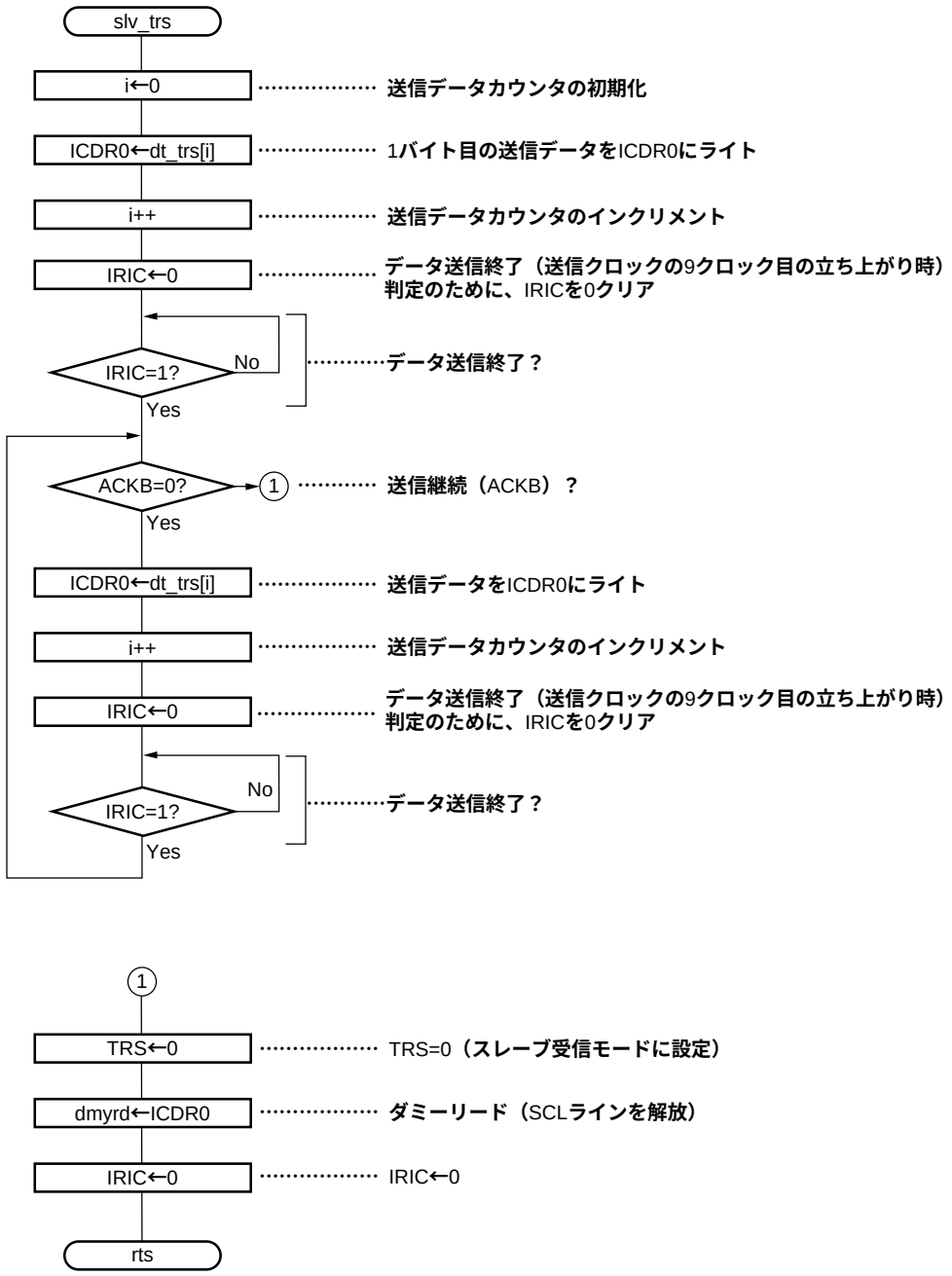
(1) メインルーチン



(2) 初期設定サブルーチン



(3) スレーブ送信サブルーチン



4.8.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*      7.Slave transmit to H8S/2138               *
*                          File name   :  SVTxd.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* IIC0 initialize */
void slv_trs(void);                              /* Slave transmit to H8S/2138 */

/************************************************
*  Data table                                    *
************************************************/
const unsigned char dt_trs[10] =
{
   0x00,                                         /* 1st transmit data */
   0x11,                                         /* 2nd transmit data */
   0x22,                                         /* 3rd transmit data */
   0x33,                                         /* 4th transmit data */
   0x44,                                         /* 5th transmit data */
   0x55,                                         /* 6th transmit data */
   0x66,                                         /* 7th transmit data */
   0x77,                                         /* 8th transmit data */
   0x88,                                         /* 9th transmit data */
   0x99                                          /* 10th transmit data */
};
```

（続く）

```
/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                       ;Stack pointer initialize
#pragma endasm
{
   unsigned char dummy;
   dummy = MDCR.BYTE;                           /* MCU mode set */

   SYSCR.BYTE = 0x09;                           /* Interrupt control mode set */
   initialize();                                /* Initialize */

   set_imask_ccr(0);                            /* Interrupt enable */

   IIC0.ICCR.BIT.MST = 0;                       /* Slave receive mode set */
   IIC0.ICCR.BIT.TRS = 0;                       /* MST = 0, TRS = 0 */
   IIC0.ICSR.BIT.ACKB = 0;                      /* ACKB = 0 */
   IIC0.ICCR.BIT.IRIC = 0;                      /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);              /* Receive end (IRIC=1) ? */

   if(IIC0.ICSR.BIT.AAS == 1)                   /* General call address receive ? */
   {
      if(IIC0.ICSR.BIT.ADZ == 0)
      {
         if(IIC0.ICCR.BIT.TRS == 1)             /* Transmit mode (TRS=1) ? */
         {
            slv_trs();                          /* Slave transmit */
         }
      }
   }
   while(IIC0.ICCR.BIT.BBSY == 1);              /* Bus empty (BBSY=0) ? */
   while(1);                                    /* End */
}

/************************************************
```

（続く）

```
*  initialize : IIC0 Initialize                  *
************************************************/
void initialize(void)
{
    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x38;                       /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  slv_trs : Slave transmit to H8S/2138         *
************************************************/
void slv_trs(void)
{
    unsigned char i = 0;                        /* Transmit data counter initialize */
    unsigned char dmyrd;                        /* Dummy read data store */

    IIC0.ICDR = dt_trs[i++];                    /* 1st transmit data write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */

    while(IIC0.ICSR.BIT.ACKB == 0)              /* Transmit continue (ACKB=0) ? */
    {
        IIC0.ICDR = dt_trs[i++];                /* Transmit data write */
        IIC0.ICCR.BIT.IRIC = 0;                 /* IRIC = 0 */
```

（続く）

```
        while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
    }

    IIC0.ICCR.BIT.TRS = 0;                          /* Slave receive mode set (MST=0,TRS=0) */
    dmyrd = IIC0.ICDR;                              /* Dummy read */
    IIC0.ICCR.BIT.IRIC = 0;                         /* IRIC = 0 */
}
```

4.9 スレーブ受信

4.9.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、スレーブ受信モードで H8S/2138 に 10 バイトのデータを受信します。
- スレーブ受信側の H8S/2138 のスレーブアドレスは［0011100］とします。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（H8S/2138）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- 受信バイト数はスレーブ受信側のアクノリッジ出力により制御します。
- **図** 4.32 に H8S/2138 の接続例を示します。

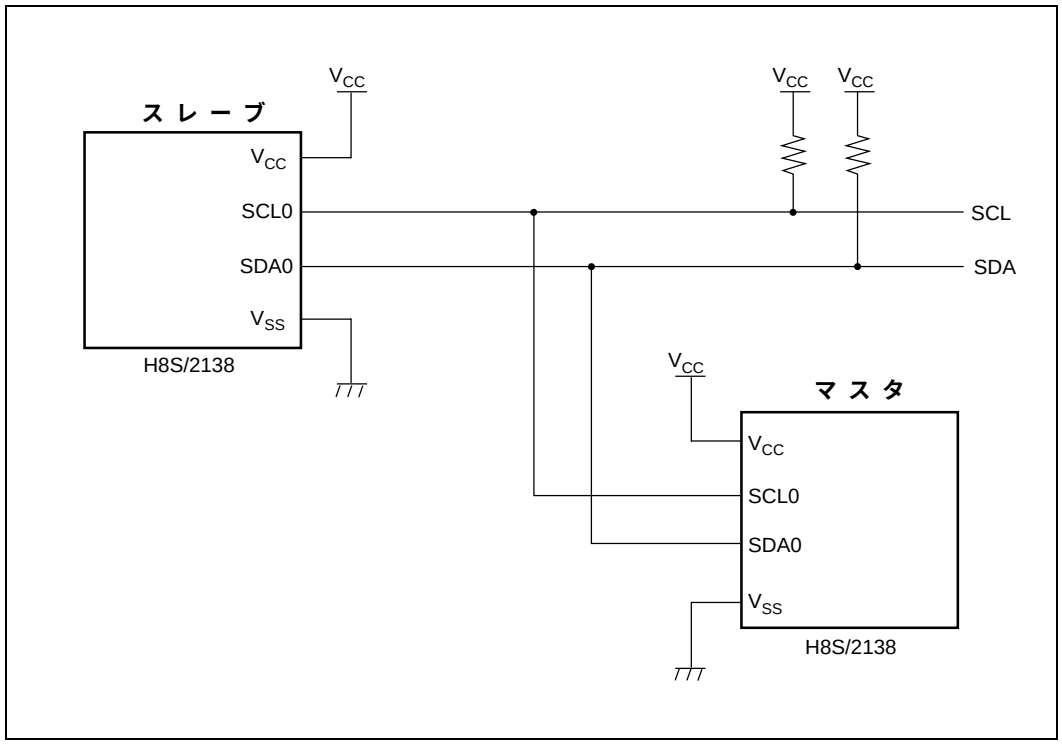


図 4.32　H8S/2138 **の接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.33 に示します。

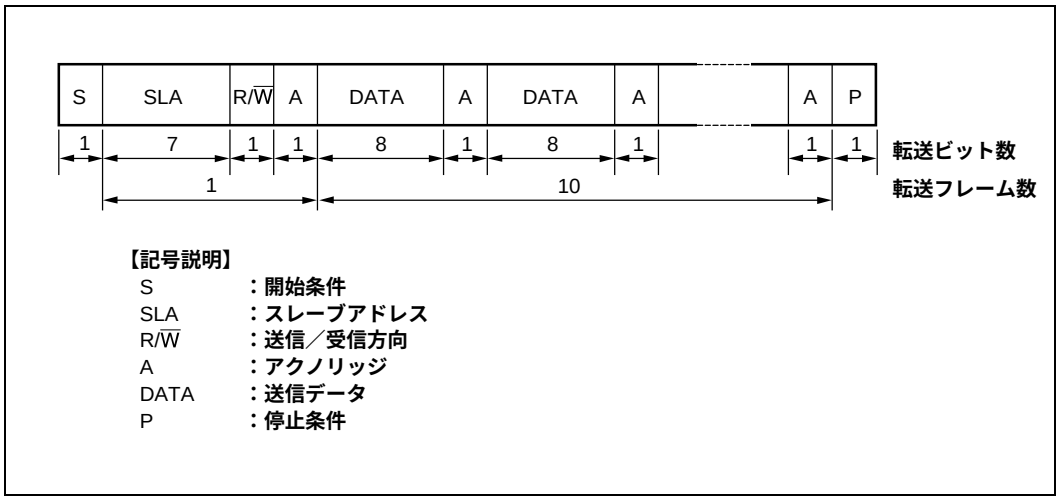
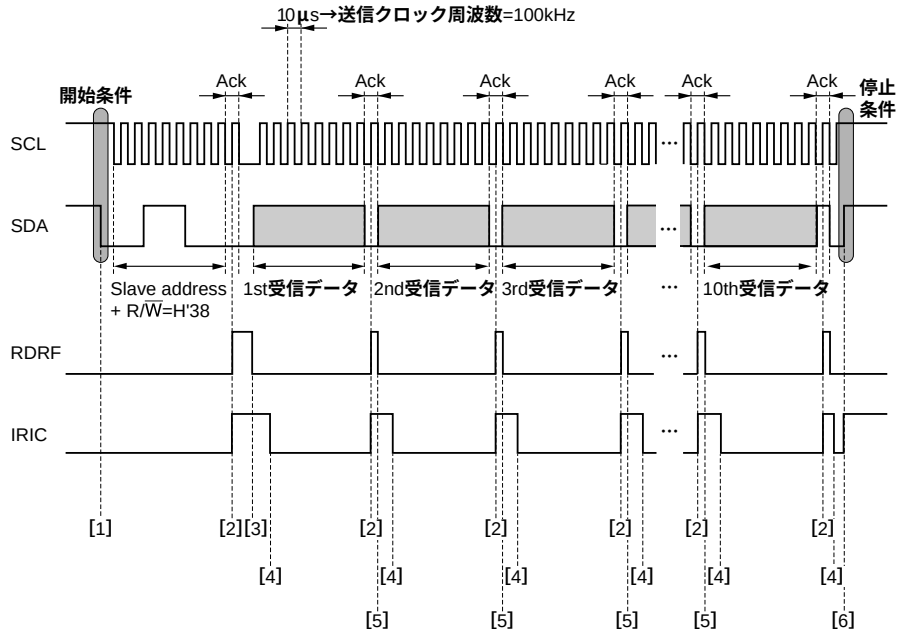


図 4.33 **本タスク例で使用する転送フォーマット**

4.9.2 動作説明

図 4.34 に動作原理を示します。



	ソフトウェア処理	ハードウェア処理
[1]	処理なし	処理なし
[2]	処理なし	RDRF=1、IRIC=1(9クロック目の立ち上がり時)
[3]	ICDR0のダミーリード（受信開始）	RDRF=0
[4]	IRICを0にクリア	処理なし
[5]	ICDR0から受信データのリード	RDRF=0
[6]	処理なし	IRIC=1(バスラインの状態から停止条件を検出)

図 4.34　スレーブ受信動作原理

4.9.3 ソフトウェア説明

(1) モジュール説明

表 4.30 に本タスク例におけるモジュール説明を示します。

表 4.30 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	使用 RAM 領域、IIC0 の初期設定
スレーブ受信	slv_rec	スレーブ受信による H8S/2138 への 10 バイトデータの受信

(2) 使用内蔵レジスタ説明

表 4.31 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.31 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		受信データを格納	H'FFDE	−
SAR0	SVA6 to SVA0	スレーブアドレスを設定	H'FFDF bit7 to bit1	SVA6=0 SVA5=0 SVA4=1 SVA3=1 SVA2=1 SVA1=0 SVA0=0
	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジの連続的な転送を設定	H'FFDF bit6	0
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0
ICCR0	ICE	ICMRO, ICDR0/SAR, SARX レジスタのアクセス制御、I^2C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I^2C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0

表 4.31 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	MST	I^2C バスインタフェースをスレーブモードで使用	H'FFD8 bit5	1
	TRS	I^2C バスインタフェースを受信モードで使用	H'FFD8 bit4	1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I^2C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	アクノリッジ出力タイミングで出力するデータを格納	H'FFD9 bit0	0/1
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I^2C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I^2C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I^2C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1

(3) 変数説明

表 4.32 に本タスク例における変数説明を示します。

表 4.32 変数説明

変数	機能	データ長	初期値	使用モジュール名
i	送信データカウンタ	1 バイト	H'00	initialize
dummy	MDCR リード値	1 バイト	−	main
dmyrd	ICDR0 ダミーリード値	1 バイト	−	slv_trs

(4) 使用 RAM 説明

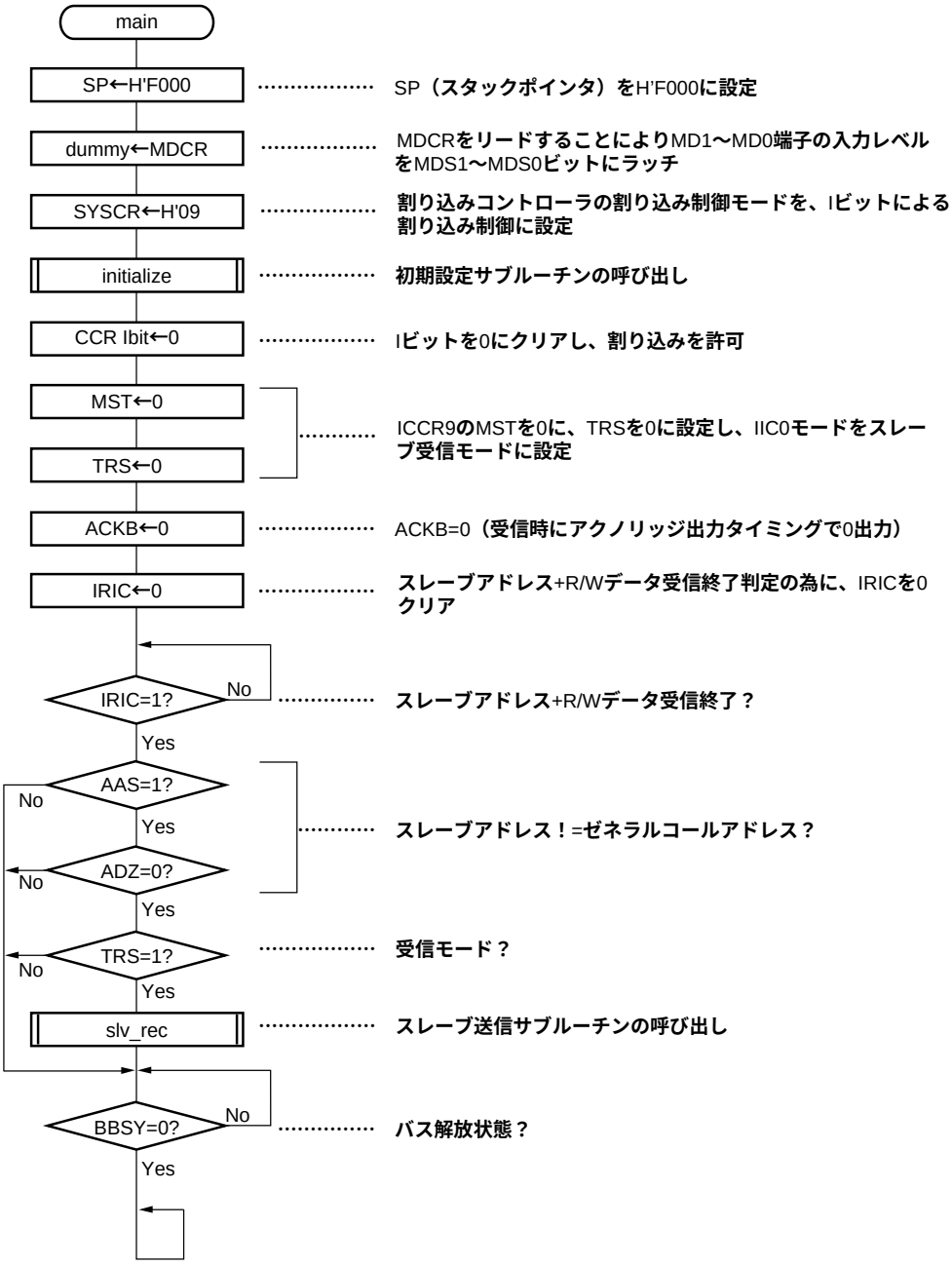
表 4.33 に本タスク例における変数説明を示します。

表 4.33 使用 RAM 説明

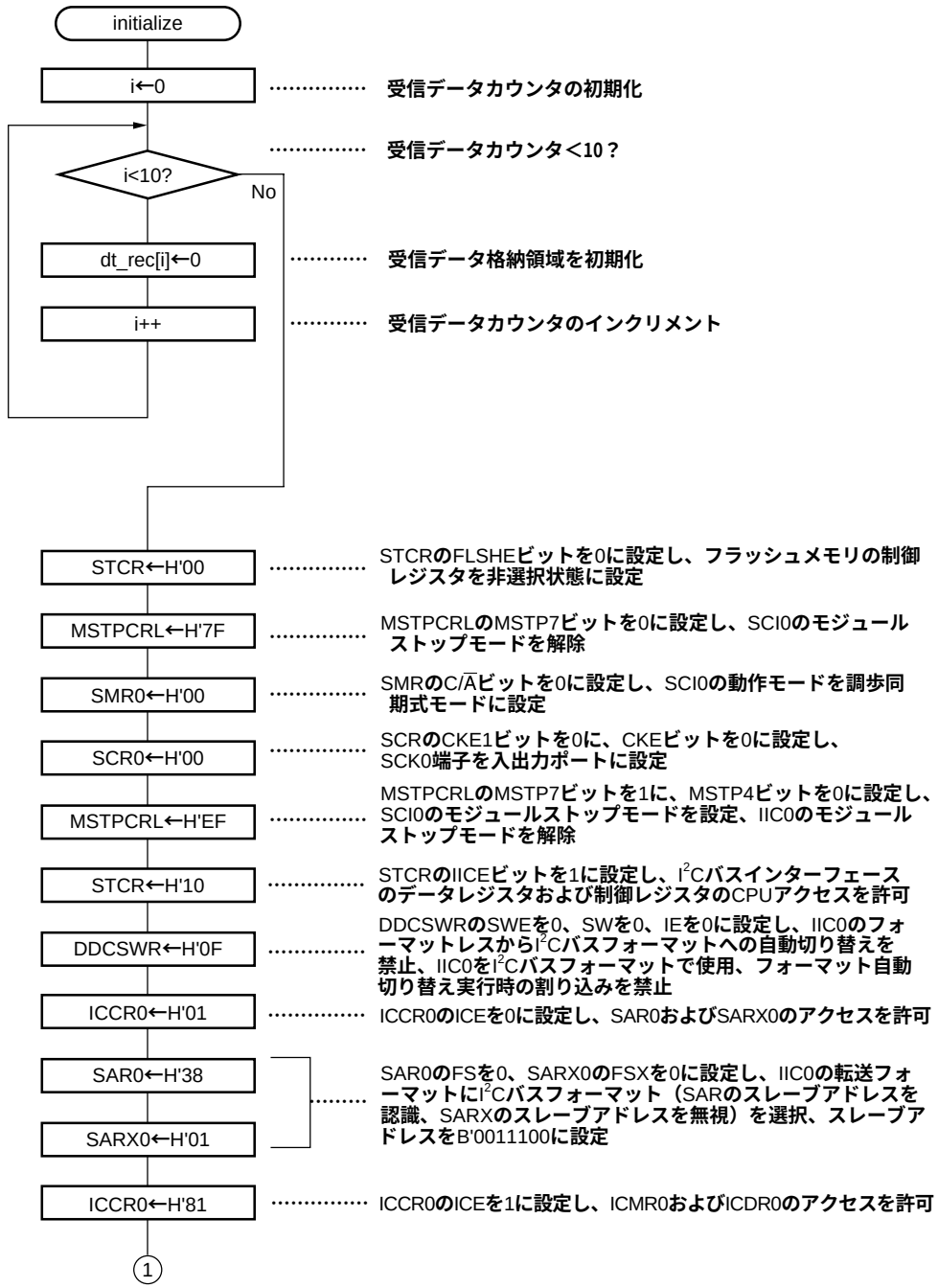
ラベル	機能	データ長	アドレス	使用モジュール名
dt_rec[i]	受信データを格納	10 バイト	H'E100 to H'E109	initialize slv_rec

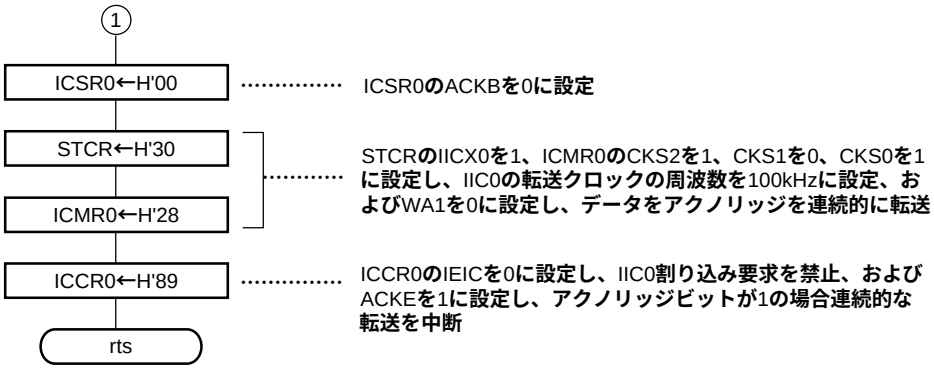
4.9.4 フローチャート

(1) メインルーチン

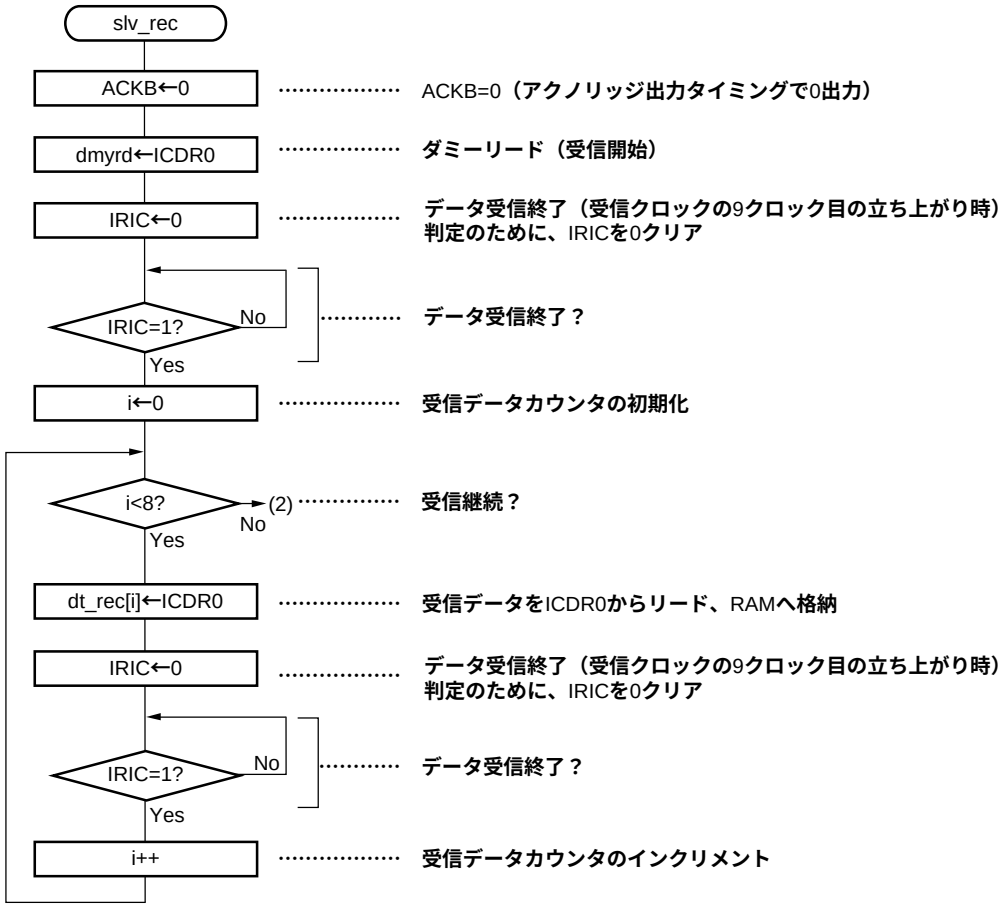


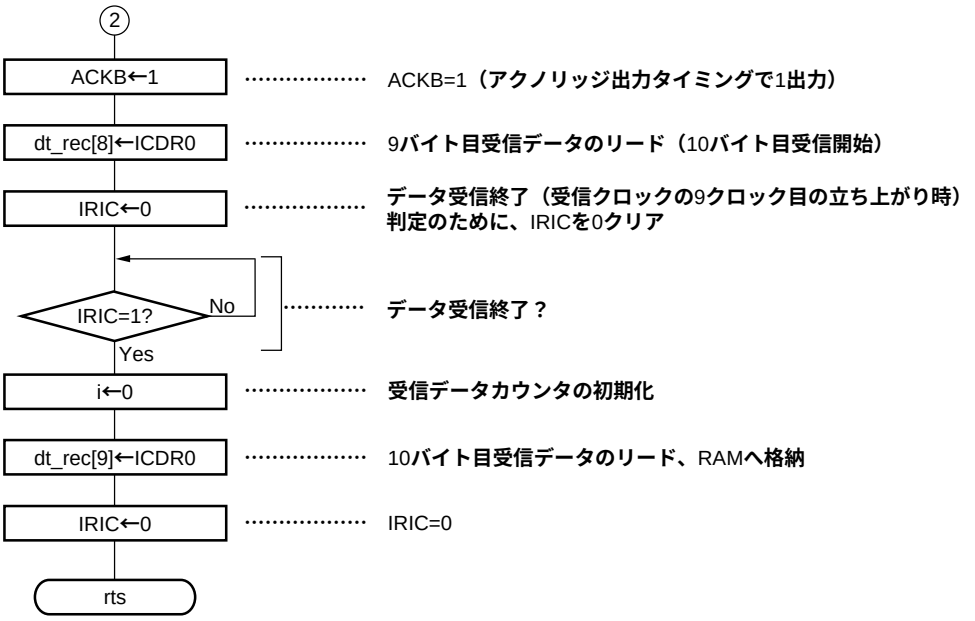
（2） 初期設定サブルーチン




1
ICSR0←H'00
ICSR0のACKBを0に設定
STCR←H'30
ICMR0←H'28
STCRのIICX0を1、ICMR0のCKS2を1、CKS1を0、CKS0を1に設定し、IIC0の転送クロックの周波数を100kHzに設定、およびWA1を0に設定し、データをアクノリッジを連続的に転送
ICCR0←H'89
ICCR0のIEICを0に設定し、IIC0割り込み要求を禁止、およびACKEを1に設定し、アクノリッジビットが1の場合連続的な転送を中断
rts

(3) スレーブ受信サブルーチン




2
ACKB←1
ACKB=1（アクノリッジ出力タイミングで1出力）
dt_rec[8]←ICDR0
9バイト目受信データのリード（10バイト目受信開始）
IRIC←0
データ受信終了（受信クロックの9クロック目の立ち上がり時）
判定のために、IRICを0クリア
IRIC=1?
No
Yes
データ受信終了？
i←0
受信データカウンタの初期化
dt_rec[9]←ICDR0
10バイト目受信データのリード、RAMへ格納
IRIC←0
IRIC=0
rts

4.9.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*      8.Slave receive from H8S/2138              *
*                          File name   :   SVRxd.c *
*                          Fai         :   20MHz   *
*                          Mode        :   3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                      *
************************************************/
void main(void);                                  /* Main routine */
void initialize(void);                            /* RAM & IIC0 initialize */
void slv_rec(void);                               /* Slave transmit to H8S/2138 */

/************************************************
*  RAM allocation                                 *
************************************************/
#pragma section ramerea
unsigned char dt_rec[10];                         /* Receive data store area */
#pragma section

/************************************************
*  main : Main routine                            *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                         ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                            /* MCU mode set */
```

（続く）

```
    SYSCR.BYTE = 0x09;                          /* Interrupt control mode set */
    initialize();                               /* Initialize */

    set_imask_ccr(0);                           /* Interrupt enable */

    IIC0.ICCR.BIT.MST = 0;                      /* Slave receive mode set */
    IIC0.ICCR.BIT.TRS = 0;                      /* MST = 0, TRS = 0 */
    IIC0.ICSR.BIT.ACKB = 0;                     /* ACKB = 0 */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Slave address receive end ? */

    if(IIC0.ICSR.BIT.AAS == 1)                  /* General call address receive ? */
    {
       if(IIC0.ICSR.BIT.ADZ == 0)
       {
          if(IIC0.ICCR.BIT.TRS == 0)            /* Slave receive mode (TRS=0) ? */
          {
             slv_rec();                         /* Slave receive */
          }
       }
    }
    while(IIC0.ICCR.BIT.BBSY == 1);             /* Bus empty (BBSY=0) ? */
    while(1);                                   /* End */
}

/************************************************
*  initialize : RAM & IIC0 Initialize          *
************************************************/
void initialize(void)
{
    unsigned char i=0;                          /* Receive data counter initialize */

    for(i=0; i<10; i++)                         /* Receive data store area initialize */
    {
       dt_rec[i] = 0x00;
    }
```

（続く）

```
    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x38;                       /* FS = 0 , Slave address = b'0011100*/
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  slv_rec : Slave receive from H8S/2138        *
************************************************/
void slv_rec(void)
{
    unsigned char i;                            /* Receive data counter initialize */
    unsigned char dmyrd;                        /* Dummy read data store area */

    IIC0.ICSR.BIT.ACKB = 0;                     /* ACKB = 0 */

    dmyrd = IIC0.ICDR;                          /* Dummy read (Receive start) */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Receive end (IRIC=1) ? */

    for(i=0; i<8; i++)
    {
        dt_rec[i] = IIC0.ICDR;                  /* Receive data read */
        IIC0.ICCR.BIT.IRIC = 0;                 /* IRIC = 0 */
        while(IIC0.ICCR.BIT.IRIC == 0);         /* Receive end (IRIC=1) ? */
    }
```

（続く）

```
    IIC0.ICSR.BIT.ACKB = 1;                     /* ACKB = 1 */
    dt_rec[8] = IIC0.ICDR;                      /* 10th data receive start */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Receive end (IRIC=1) ? */

    dt_rec[9] = IIC0.ICDR;                      /* 10th receive data read */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
}
```

4.10 バス切断時の処理例

4.10.1 仕様

- H8S/2138 の I²C バスインタフェースのチャネル 0 を使用して、マスタ送信モードにより EEPROM（HN58X2408）に 5 バイトのデータを書き込みます。
- 接続する EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00 番地から H'04 番地にデータを書き込みます。
- 書き込みデータは［H'A1、H'B2、H'C3、H'D4、H'E5］とします。
- 転送途中にバスが切断されたことを想定して、3 バイト目の送信クロックを 8 クロックで停止し、ICCR0 の ICE ビットを“0”にクリアして IIC0 モジュールを非動作状態（SCL0/SDA0 端子はポート機能）に設定します。その後、EEPROM のライトサイクル時間経過後、再び最初から EEPROM に 5 バイトのデータ書き込みを行います。
- 本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8S/2138）1 個、スレーブデバイス（H8S/2138）1 個のシングルマスタ構成とします。
- 転送クロックの周波数は 100kHz とします。
- **図** 4.35 に H8S/2138 の接続例を示します。

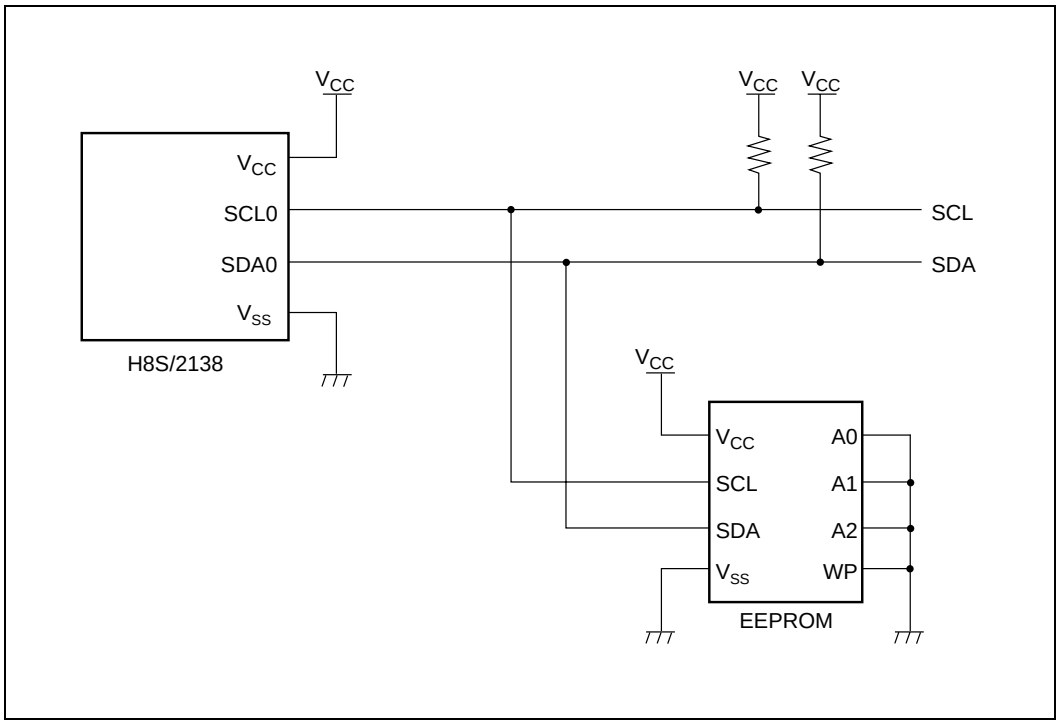
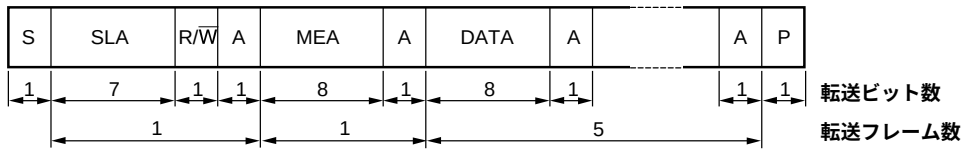


図 4.10.1 H8S/2138 **と** EEPROM **との接続例**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.36 に示します。



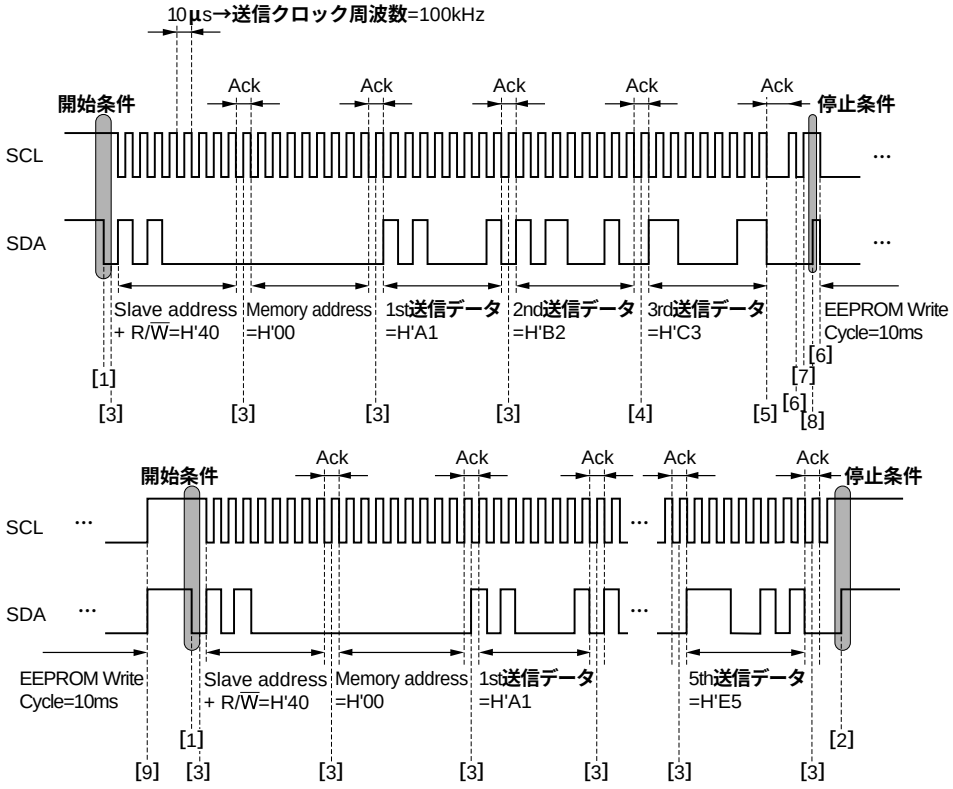
【記号説明】

S	**：開始条件**
SLA	**：**EEPROM**スレーブアドレス**
R/$\overline{W}$	**：送信／受信方向**
A	**：アクノリッジ**
MEA	**：**EEPROM**メモリアドレス**
DATA	**：送信データ**
P	**：停止条件**

図 4.36　**本タスク例で使用する転送フォーマット**

4.10.2 動作説明

図 4.37 に動作原理を示します。



	ソフトウェア処理
[1]	**開始条件の発行**（BBSY=I、SCP=0）
[2]	**停止条件の発行**（BBSY=0、SCP=0）
[3]	ICDR0**に送信データをライト**、IRIC**を**0**にクリア**（9**クロック目立ち上がり時**）
[4]	WAIT=1、ICDR0**に送信データをライト**、IRIC**を**0**にクリア**（9**クロック目立ち上がり時**）
[5]	ICE=0（8**クロック目立ち下がり時**）、P52(SCL0)、P97(SDA0)**端子より**“Low”**出力**
[6]	P52(SCL0)、P97(SDA0)**端子より**“Low”**出力、**9**クロック目生成**
[7]	P52(SCL0)**端子より**“High”**出力、停止条件の生成**
[8]	P97(SDA0)**端子より**“Low”**出力、停止条件の生成**
[9]	IIC0**モジュール初期設定**

図 4.37　**バス切断時の処理動作原理**

4.10.3 ソフトウェア説明

(1) モジュール説明

表 4.33 に本タスク例におけるモジュール説明を示します。

表 4.33 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、割り込みの許可
初期設定	initialize	IIC0 の初期設定
マスタ送信 1	mst_trs_1	マスタ送信による EEPROM への 5 バイトデータの送信（1）
マスタ送信 2	mst_trs_2	マスタ送信による EEPROM への 5 バイトデータの送信（2）
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス+W データ（H'A0）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信
Wait1	wait_1	5μs 待機（20MHz 動作時）
Wait2	wait_2	10ms 待機（20MHz 動作時）

(2) 使用内蔵レジスタ説明

表 4.34 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.34 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FS ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジ間にウェイトを挿入するか否かを設定	H'FFDF bit6	0/1
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I²C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0

表 4.34 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I²C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I²C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I²C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I²C バスインタフェースを送信モードで使用	H'FFD8 bit4	1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	EEPROM より送信されたアクノリッジデータを格納	H'FFD9 bit0	−
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I²C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1
P5DDR	P52DDR	P52 端子を出力端子を設定	H'FFB8 bit2	1
P5DR	P52DR	P52 端子を出力データを設定	H'FFBA bit2	0/1
P9DDR	P97DDR	P97 端子を出力端子に設定	H'FFC0 bit7	1

表 4.34 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
P9DR	P97DR	P97 端子を出力データに設定	H'FFC1 bit7	0/1

（3） 変数説明

表 4.35 に本タスク例における変数説明を示します。

表 4.35 変数説明

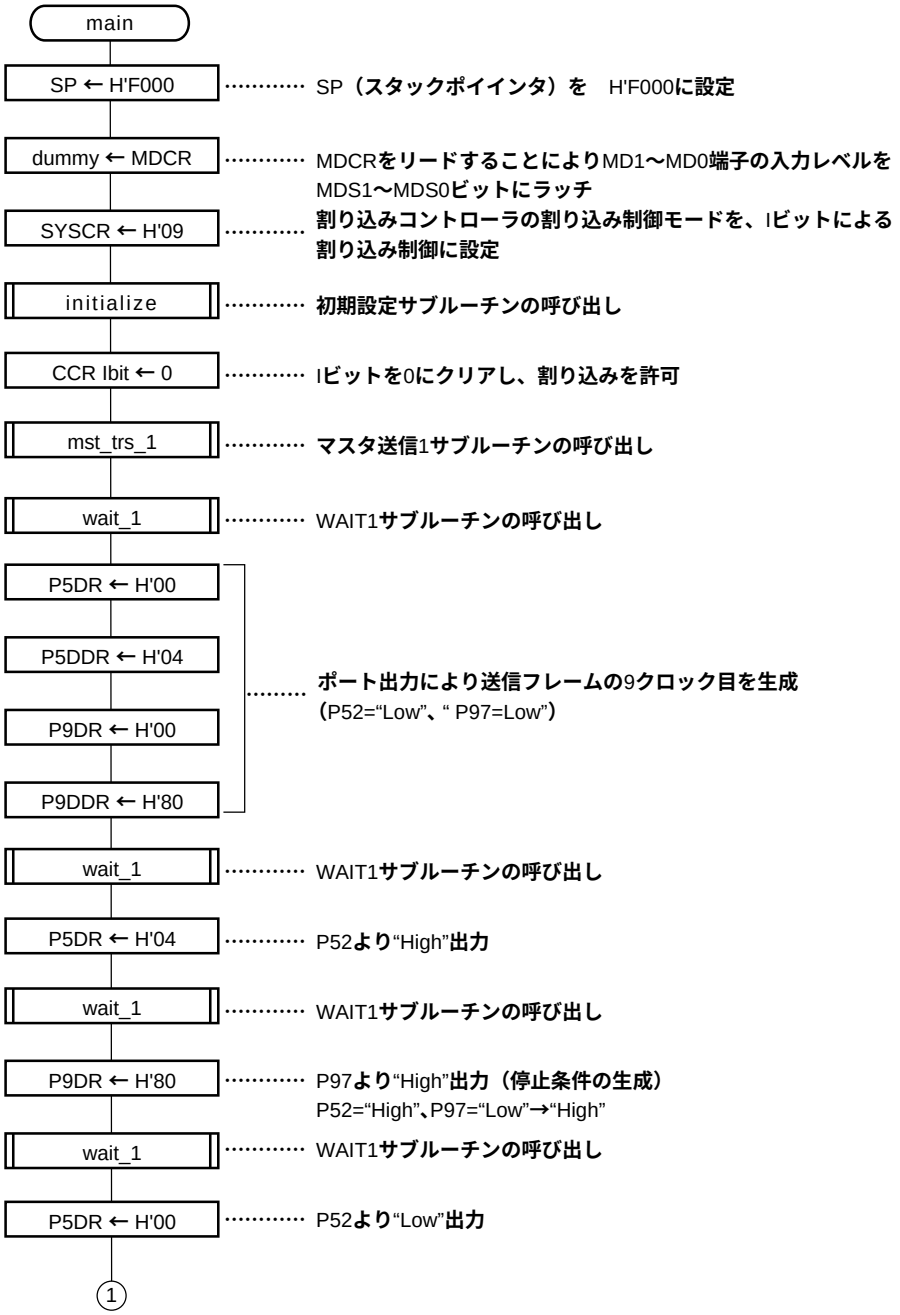
変数	機能	データ長	初期値	使用モジュール名
i	送信データカウンタ	1 バイト	H'00	mst_trs_1 mst_trs_2
dummy	MDCR リード値	1 バイト	−	main
dt_trs[i]	5 バイト送信データ	5 バイト	−	mst_trs_1 mst_trs_2

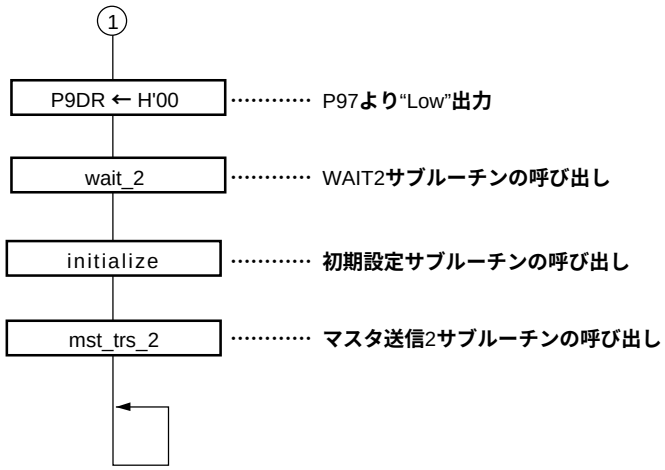
（4） 使用 RAM 説明

本タスク例では、変数以外の RAM は使用しません。

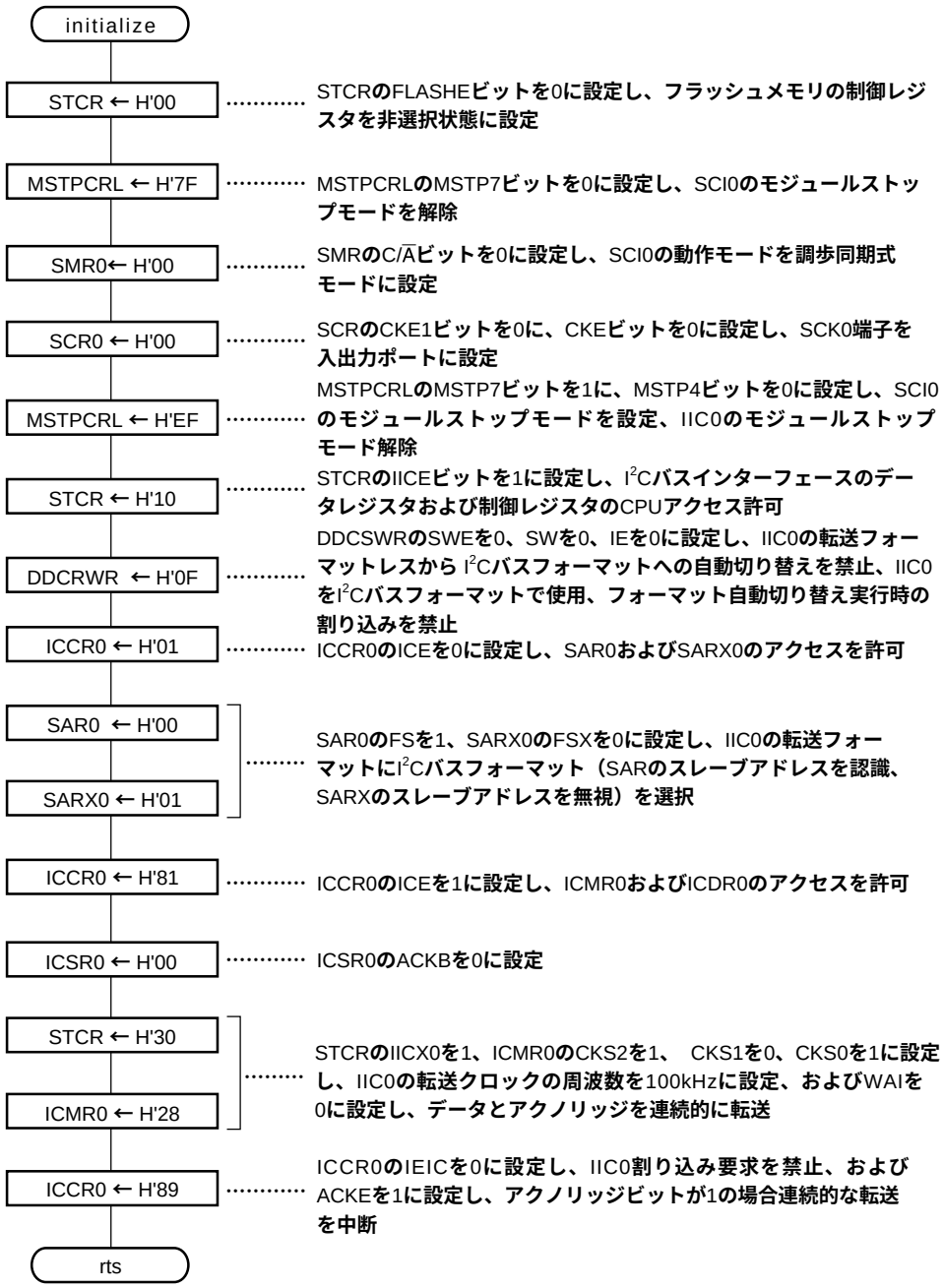
4.10.4 フローチャート

（1） メインルーチン

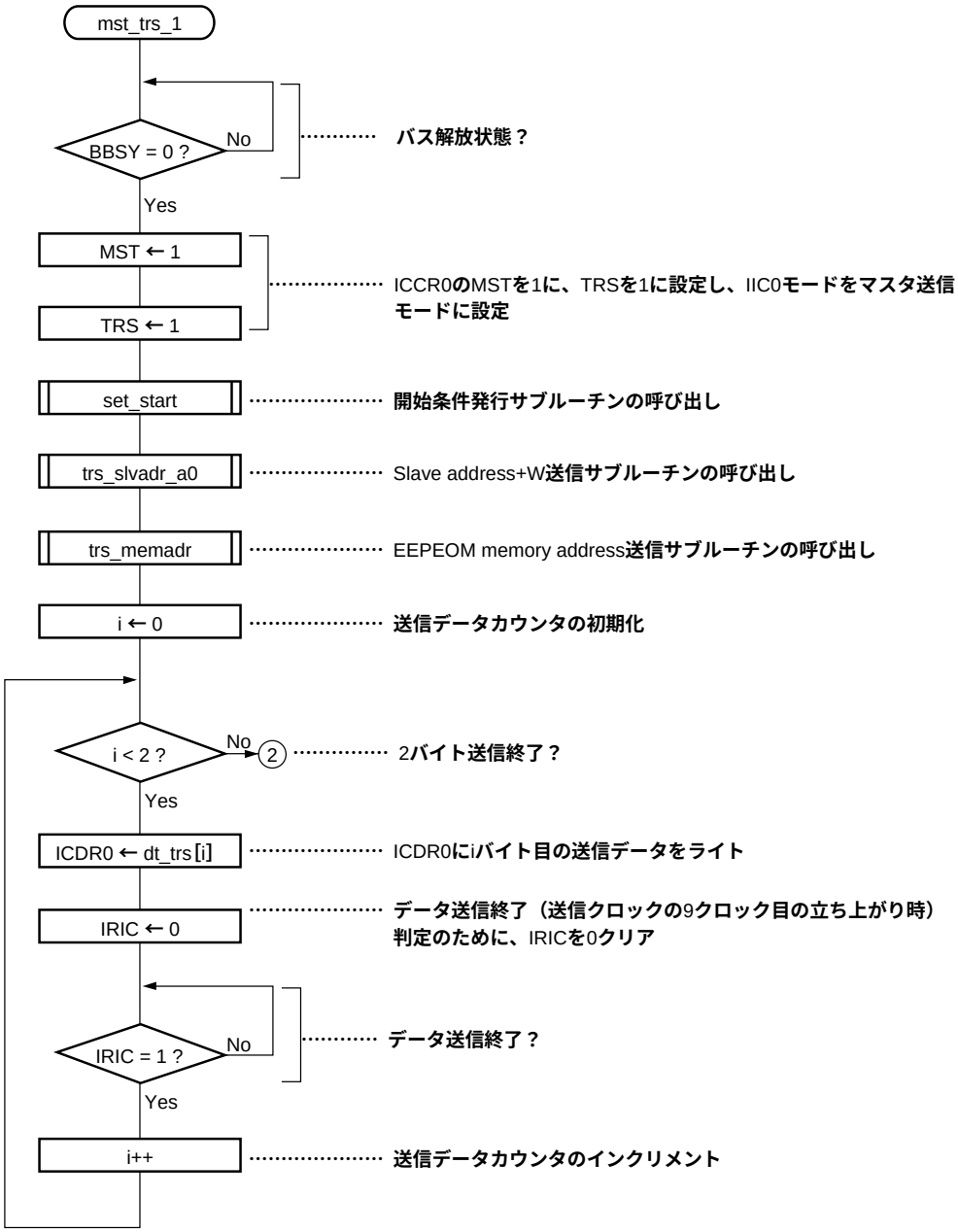


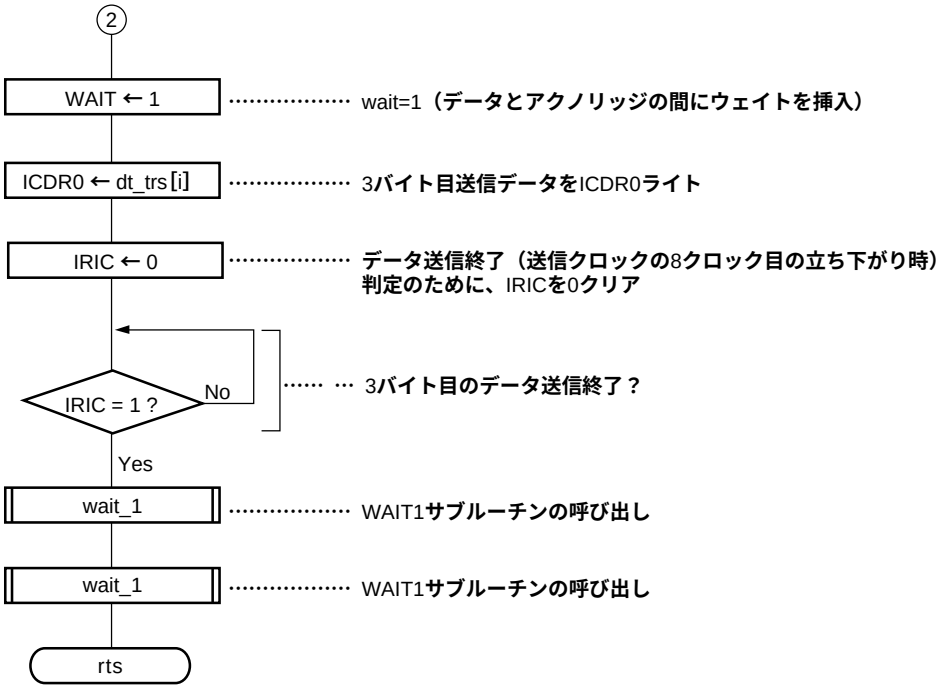

1
P9DR ← H'00
P97より“Low”出力
wait_2
WAIT2サブルーチンの呼び出し
initialize
初期設定サブルーチンの呼び出し
mst_trs_2
マスタ送信2サブルーチンの呼び出し

(2) 初期設定サブルーチン

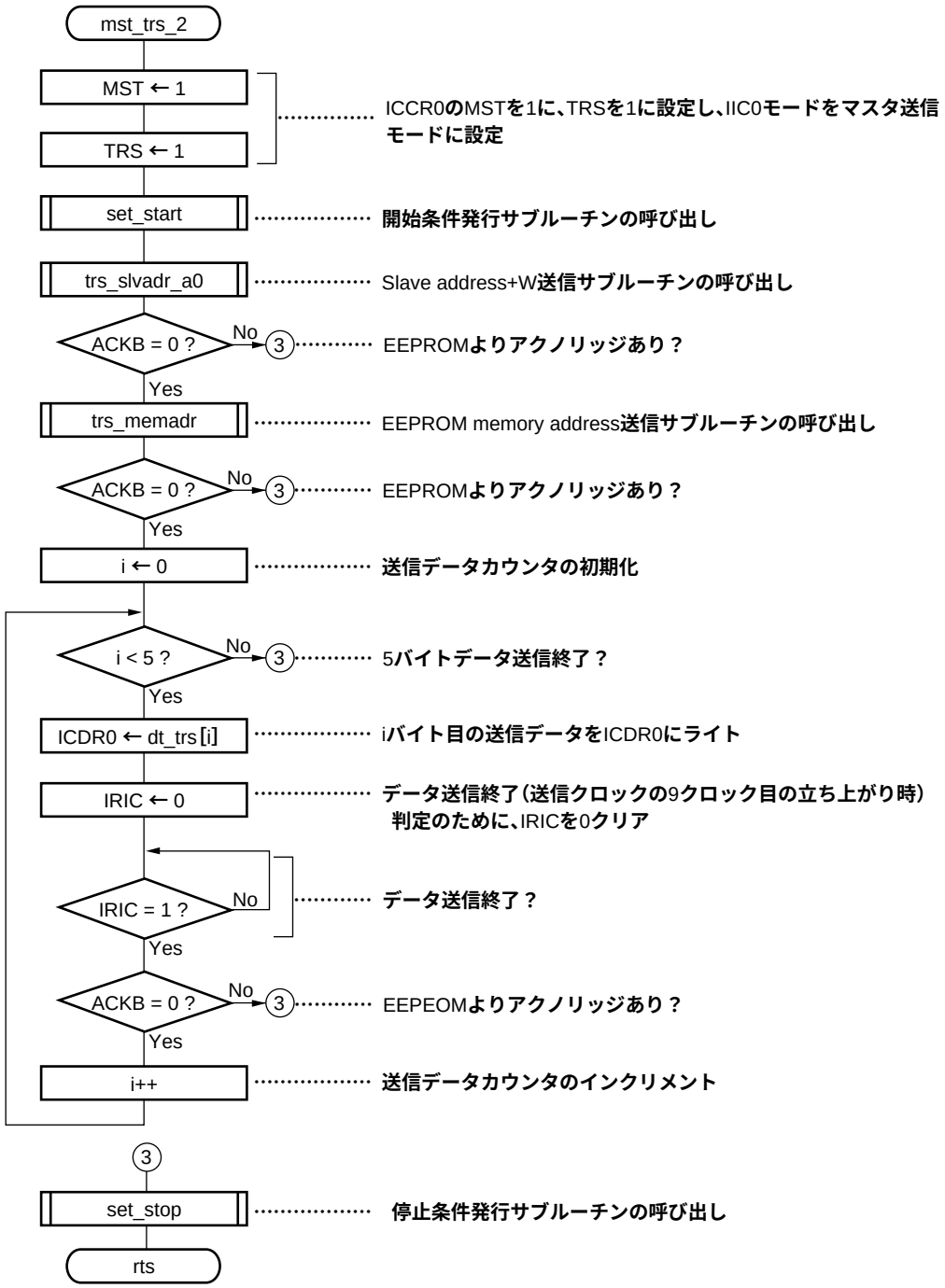


(3) マスタ送信 1 サブルーチン

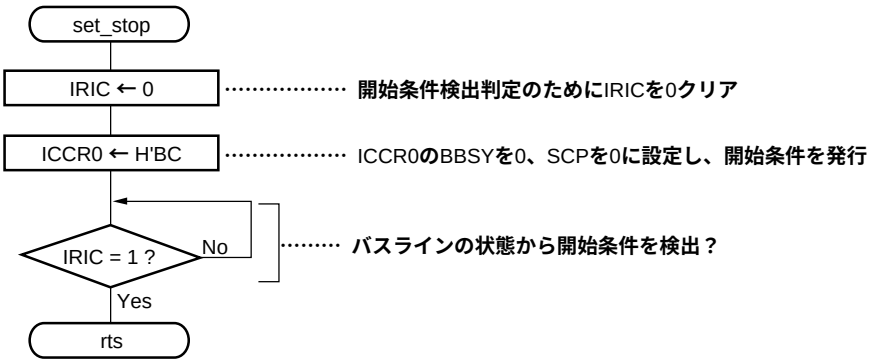



2
WAIT ← 1
wait=1（データとアクノリッジの間にウェイトを挿入）
ICDR0 ← dt_trs[i]
3バイト目送信データをICDR0ライト
IRIC ← 0
データ送信終了（送信クロックの8クロック目の立ち下がり時）
判定のために、IRICを0クリア
IRIC = 1 ?
No
3バイト目のデータ送信終了？
Yes
wait_1
WAIT1サブルーチンの呼び出し
wait_1
WAIT1サブルーチンの呼び出し
rts

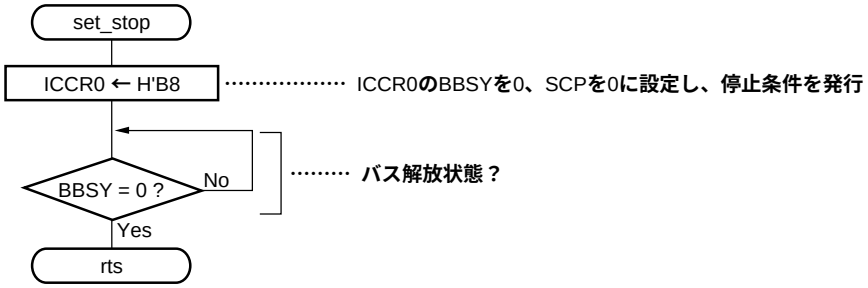
(4) マスタ送信 2 サブルーチン



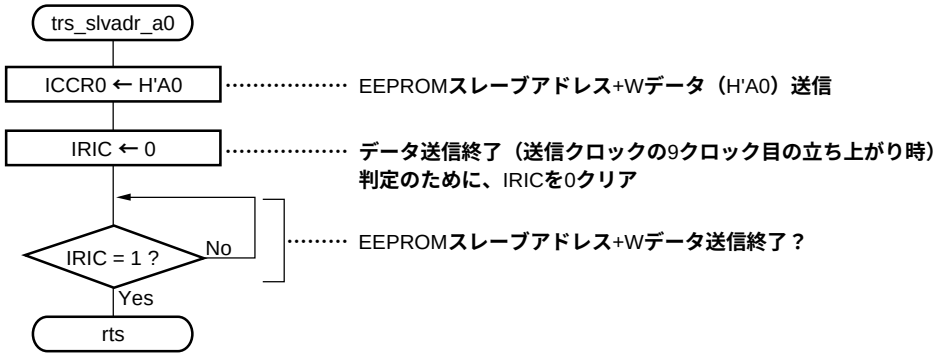
（5） 開始条件発行サブルーチン



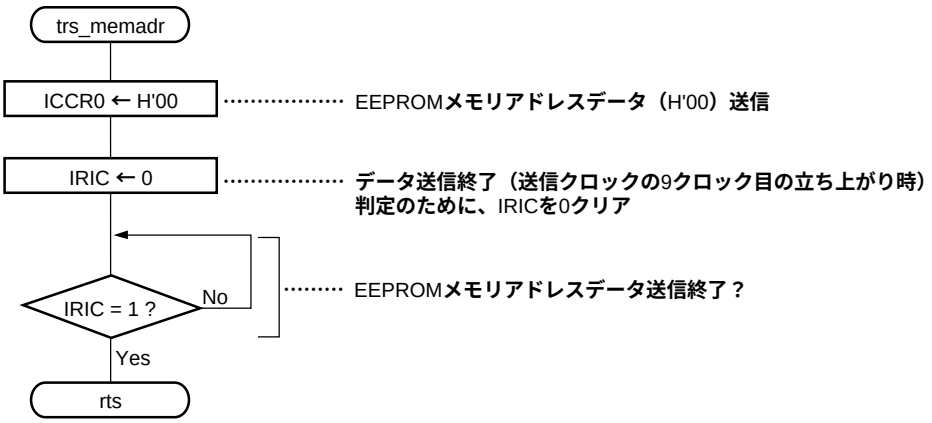
（6） 停止条件発行サブルーチン



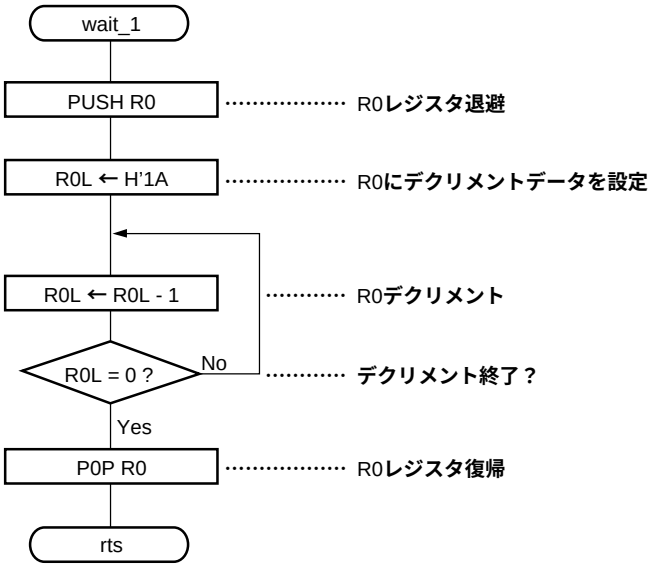
（7） Slave address + W 送信サブルーチン



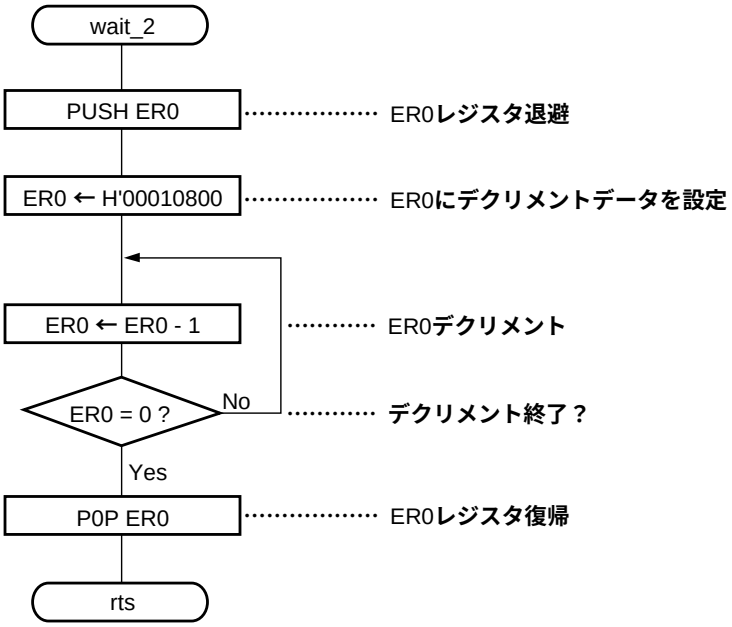
（8） EEPROM memory address 送信サブルーチン



（9） WAIT1 サブルーチン



（10）WAIT2 サブルーチン



4.10.5 プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note             *
*      9.Error process in single master transmit  *
*                          File name   :  Error.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* IIC0 initialize */
void mst_trs_1(void);                            /* Master transmit 1 */
void mst_trs_2(void);                            /* Master transmit 2 */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_memadr(void);                           /* EEPROM memory address data transmit */
void wait_1(void);                               /* Wait 1 (5μs) */
void wait_2(void);                               /* Wait 2 (10ms) */

/************************************************
*  Data table                                    *
************************************************/
const unsigned char dt_trs[5] =                  /* Transmit data (5 byte) */
{
   0xa1,                                         /* 1st transmit data */
   0xb2,                                         /* 2nd transmit data */
   0xc3,                                         /* 3rd transmit data */
   0xd4,                                         /* 4th transmit data */
   0xe5                                          /* 5th transmit data */
```

（続く）

```
};

/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                       ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                          /* MCU mode set */

    SYSCR.BYTE = 0x09;                          /* Interrupt control mode set */
    initialize();                               /* Initialize */

    set_imask_ccr(0);                           /* Interrupt enable */

    mst_trs_1();                                /* Master transmit to EEPROM 1 */

    wait_1();                                   /* 5μs wait */
    P5.DR.BYTE = 0x00;                          /* P52DR = 0 */
    P5.DDR = 0x04;                              /* P52DDR = 1 */
    P9.DR.BYTE = 0x00;                          /* P97DR = 0 */
    P9.DDR = 0x80;                              /* P97DDR = 1 */
    wait_1();                                   /* 5μs wait */
    P5.DR.BYTE = 0x04;                          /* P52DR = 1 */
    wait_1();                                   /* 5μs wait */
    P9.DR.BYTE = 0x80;                          /* P92DR = 1 */
    wait_1();                                   /* 5μs wait */
    P5.DR.BYTE = 0x00;                          /* P52DR = 0 */
    P9.DR.BYTE = 0x00;                          /* P97DR = 0 */
    wait_2();                                   /* 10ms wait (EEPROM write cycle) */

    initialize();                               /* IIC0 initialzie */

    mst_trs_2();                                /* Master transmit to EEPROM 2 */
```

（続く）

```
    while(1);                                   /* End */
}

/************************************************
*  initialize : IIC0 Initialize                 *
************************************************/
void initialize(void)
{
    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    BSC.WSCR.BYTE = 0x33;                       /* SDA0 pin function set */
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x00;                       /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  mst_trs_1 : Master transmit to EEPROM 1      *
************************************************/
void mst_trs_1(void)
{
    unsigned char i;                            /* Transmit data counter */

    while(IIC0.ICCR.BIT.BBSY == 1);             /* Bus empty (BBSY=0) ? */
    IIC0.ICCR.BIT.MST = 1;                      /* Master transmit mode set */
    IIC0.ICCR.BIT.TRS = 1;                      /* MST = 1, TRS = 1 */
    set_start();                                /* Start condition set */
```

（続く）

```
    trs_slvadr_a0();                            /* Slave address + W data transmit */
    trs_memadr();                               /* EEPROM memory address data transmit */
    for(i=0; i<2; i++)
    {
        IIC0.ICDR = dt_trs[i];                  /* Transmit data write */
        IIC0.ICCR.BIT.IRIC = 0;                 /* IRIC = 0 */
        while(IIC0.ICCR.BIT.IRIC == 0);         /* Transmit end (IRIC=1) ? */
    }
    IIC0.ICMR.BIT.WAIT = 1;                     /* WAIT = 1 */
    IIC0.ICDR = dt_trs[i];                      /* 3rd transmit data write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
    wait_1()                                    /* 5µs wait */
    wait_1();                                   /* 5µs wait */
    IIC0.ICCR.BIT.ICE = 0;                      /* ICE = 0 */
}

/***********************************************
*  mst_trs_2 : Master transmit to EEPROM 2     *
***********************************************/
void mst_trs_2(void)
{
    unsigned char i;                            /* Transmit data counter */

    IIC0.ICCR.BIT.MST = 1;                      /* Master transmit mode set */
    IIC0.ICCR.BIT.TRS = 1;                      /* MST = 1, TRS = 1 */
    set_start();                                /* Start condition set */
    trs_slvadr_a0();                            /* Slave address + W data transmit */

    if(IIC0.ICSR.BIT.ACKB == 0)                 /* ACKB = 0 ? */
    {
        trs_memadr();                           /* EEPROM memory address data transmit */

        if(IIC0.ICSR.BIT.ACKB == 0)             /* ACKB = 0 ? */
        {
            for(i=0; i<5; i++)
            {
```

（続く）

```
            IIC0.ICDR = dt_trs[i];              /* Transmit data write */
            IIC0.ICCR.BIT.IRIC = 0;             /* IRIC = 0 */
            while(IIC0.ICCR.BIT.IRIC == 0);     /* Transmit end (IRIC=1) ? */

            if(IIC0.ICSR.BIT.ACKB == 1)         /* ACKB = 0 ? */
            {
               break;
            }
         }
      }
   }
   set_stop();                                  /* Stop condition set */
}

/************************************************
*  set_start : Start condition set              *
************************************************/
void set_start(void)
{
   IIC0.ICCR.BIT.IRIC = 0;                      /* IRIC = 0 */
   IIC0.ICCR.BYTE = 0xbc;                       /* Start condition set (BBSY=1,SCP=0) */
   while(IIC0.ICCR.BIT.IRIC == 0);              /* Start condition set (IRIC=1) ? */
}

/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
   IIC0.ICCR.BYTE = 0xb8;                       /* Stop condition set (BBSY=0, SCP=0) */
   while(IIC0.ICCR.BIT.BBSY == 1);              /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
```

（続く）

```
{
   IIC0.ICDR = 0xa0;                              /* Slave address + W data(H'A0) write */
   IIC0.ICCR.BIT.IRIC = 0;                        /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);                /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
   IIC0.ICDR = 0x00;                              /* EEPROM memory address data(H'00) write */
   IIC0.ICCR.BIT.IRIC = 0;                        /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);                /* Transmit end (IRIC=1) ? */
}

/************************************************
*  wait_1 : xxms wait                            *
************************************************/
void wait_1(void)
{
#pragma asm
       push.w  r0                                 ;Push R0
       mov.b   #h'1a,r0l                          ;Decrement data set
wait1_1:
       dec.b   r0l                                ;Decrement
       bne    wait1_1                             ;Decrement end ?
       pop.w   r0                                 ;Pop R0
#pragma endasm
}

/************************************************
*  wait_2 : 10ms wait                            *
************************************************/
void wait_2(void)
{
#pragma asm
```

（続く）

```
        push.l  er0                             ;Push ER0
        mov.l   #h'00010800,er0                 ;Decrement data set
wait2_1:
        dec.l   #1,er0                          ;Decrement
        bne     wait2_1                         ;Decrement end ?
        pop.l   er0                             ;Pop ER0
#pragma endasm
}
```

4.11 バスの競合

4.11.1 仕様

- システム構成は、マスタデバイス 1（H8S/2138）、マスタデバイス 2（H8S/2138）、スレーブデバイス（EEPROM：HN58X2408）のマルチマスタ構成とします。
- マスタ 1 およびマスタ 2 に接続された IRQ 割り込みスイッチを押すと、マスタ 1 およびマスタ 2 は同時にスレーブデバイスの EEPROM に 2 バイトのデータを書き込みます。
- マスタ 1 の送信データは、7-segment LED に「10」を表示するデータとします。
- マスタ 2 の送信データは、7-segment LED に「20」を表示するデータとします。
- マスタ 1、マスタ 2 が同時にデータ送信を開始するのでバスの競合が起こります。このとき、バス権を獲得したマスタはそのまま EEPROM へのデータ書き込みを継続し、LED を点灯させます。バス競合負けしたマスタは、他のマスタが EEPROM への書き込みを終了した後に、他のマスタが書込んだデータを EEPROM から読み出し、接続してある 7-segment LED に読み出したデータを表示します。
- スレーブデバイスである EEPROM のスレーブアドレスは［1010000］とし、EEPROM メモリアドレスの H'00、H'01 番地を使用してデータの書き込みおよび読み出しを行います。
- 転送クロックの周波数は、送信／受信ともに 100kHz とします。
- **図** 4.38 に本タスク例のシステム構成を示します。

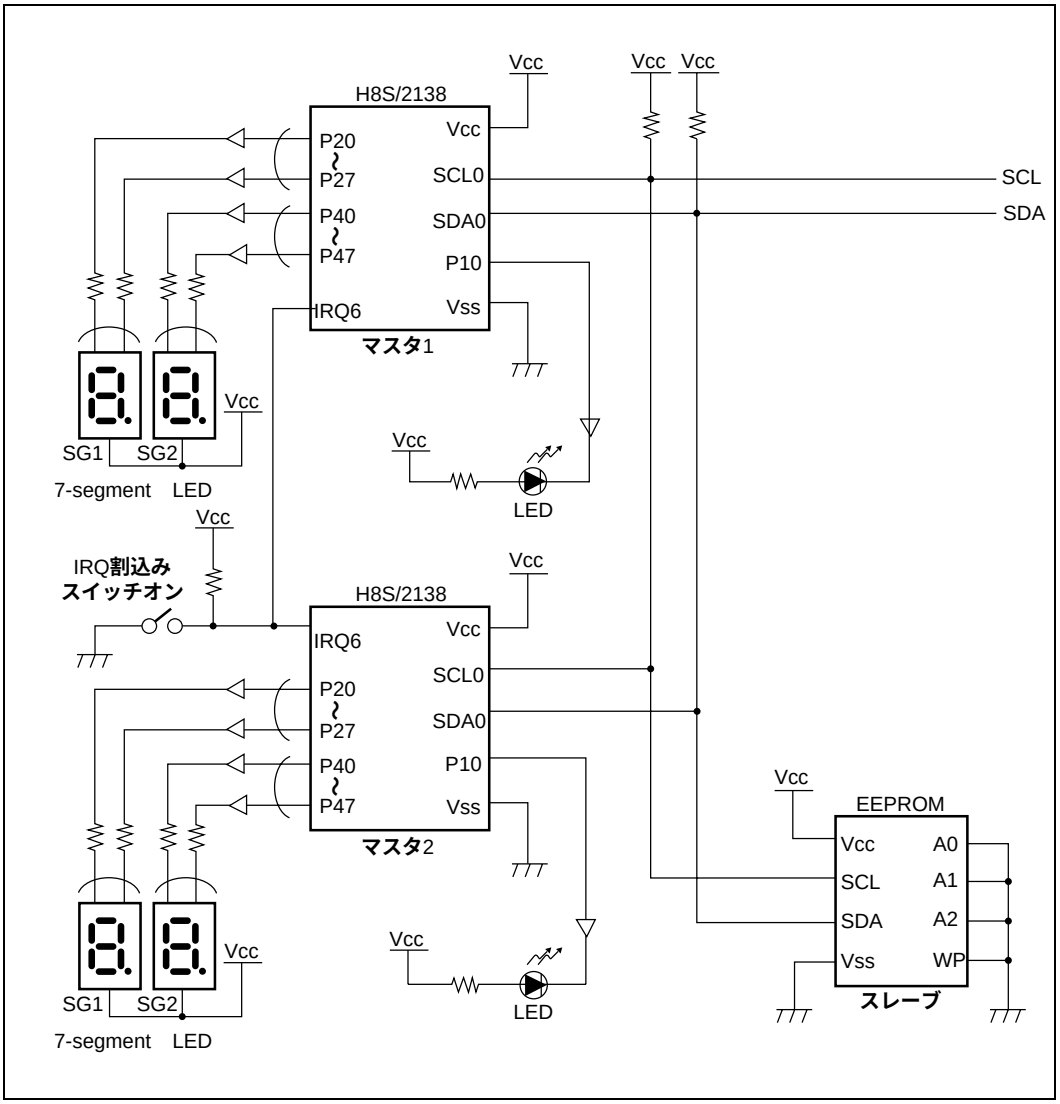

H8S/2138
Vcc
P20
~
P27
P40
~
P47
SCL0
SDA0
P10
IRQ6
Vss
マスタ1
SCL
SDA
SG1
SG2
7-segment
LED
IRQ割込み
スイッチオン
マスタ2
EEPROM
A0
A1
A2
WP
スレーブ

図 4.38 **システム構成**

- 本タスク例で使用する I²C バスフォーマットを**図** 4.39 に示します。

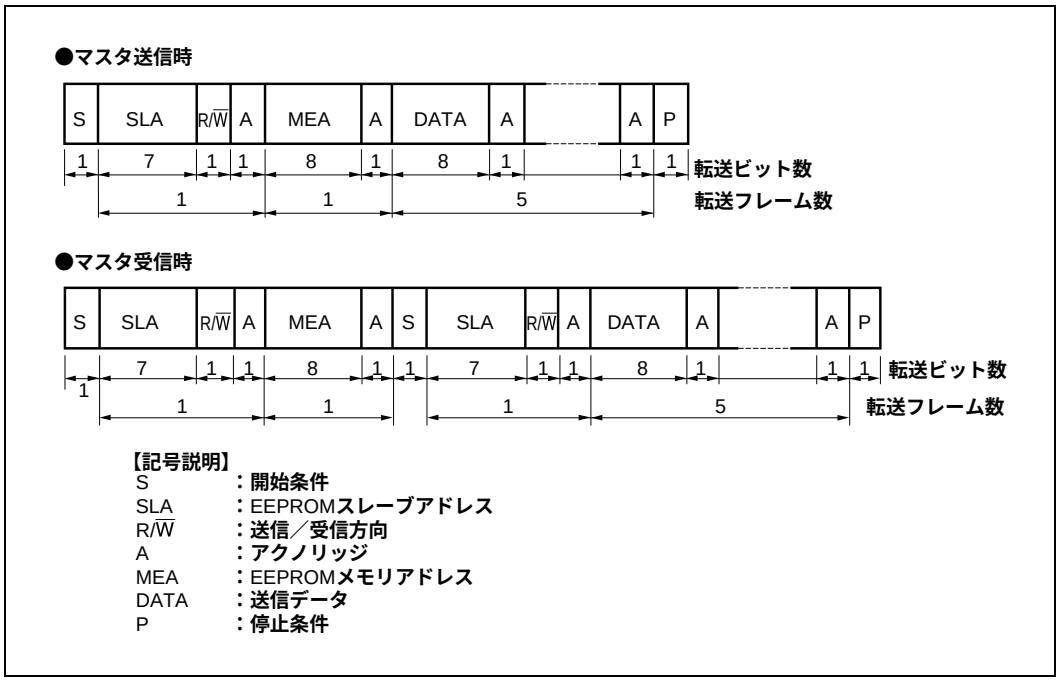


図 4.39 **本タスク例で使用する転送フォーマット**

- H8S シリーズ内蔵の I²C バスインタフェースでは、**「**1.4 **通信調整手順」**で示した通信調整手順のほかに、**図** 4.40 に示す通信調整手順が行われます。各マスタデバイスは SCL の立ち下がりでバスラインをモニタし、自分のレベルと一致しない場合、出力段をオフします。

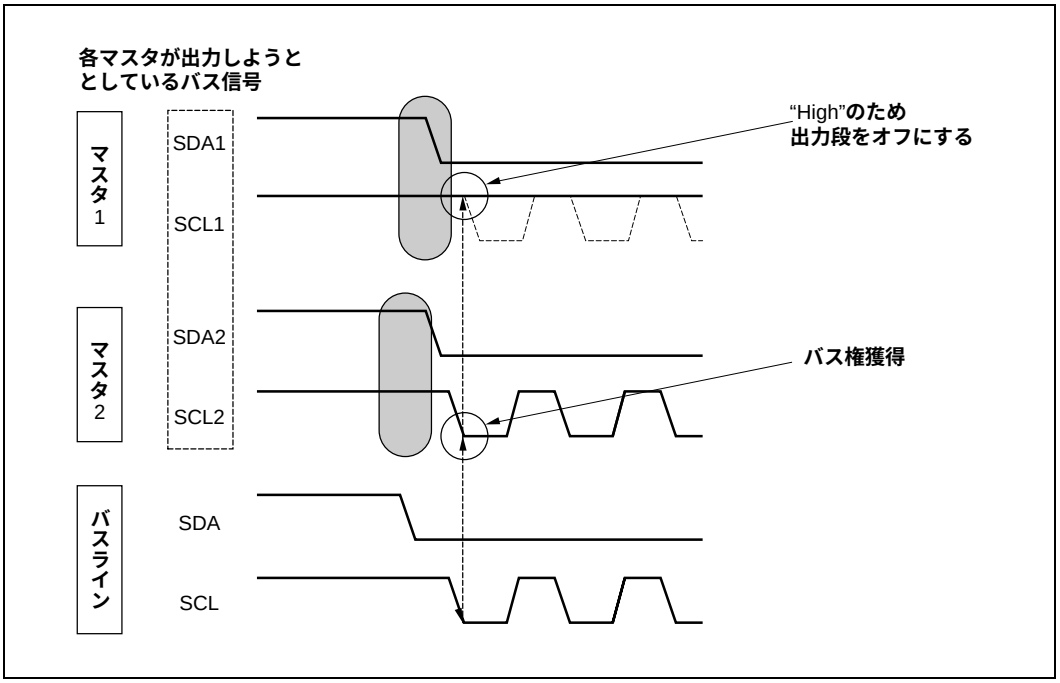


図 4.40　**バスアービトレーション検出方法**

- マスタ 1、およびマスタ 2 が同時に同じスレーブデバイスにデータ送信を開始するので、送信するスレーブアドレス（第 1 フレーム）、および EEPROM メモリアドレス（第 2 フレーム）まで同一のデータを送信するため、通信調整手順は送信データの比較により行われます。マスタ 1 の第 3 フレーム送信データ（1 バイト目送信データ）は H'F9、マスタ 2 のデータは H'A4 であるため、マスタ 2 が先に SDA を“ Low ”にすることによりバス権を獲得します（**図** 4.41 参照）。

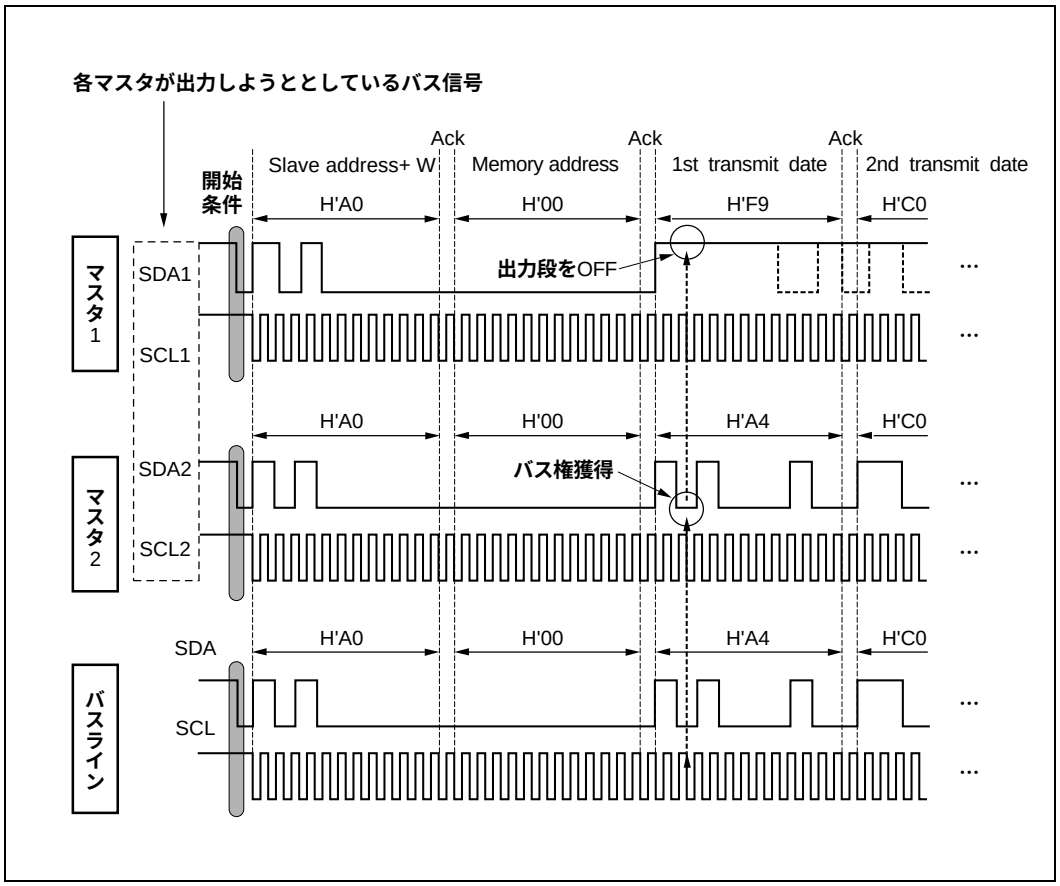
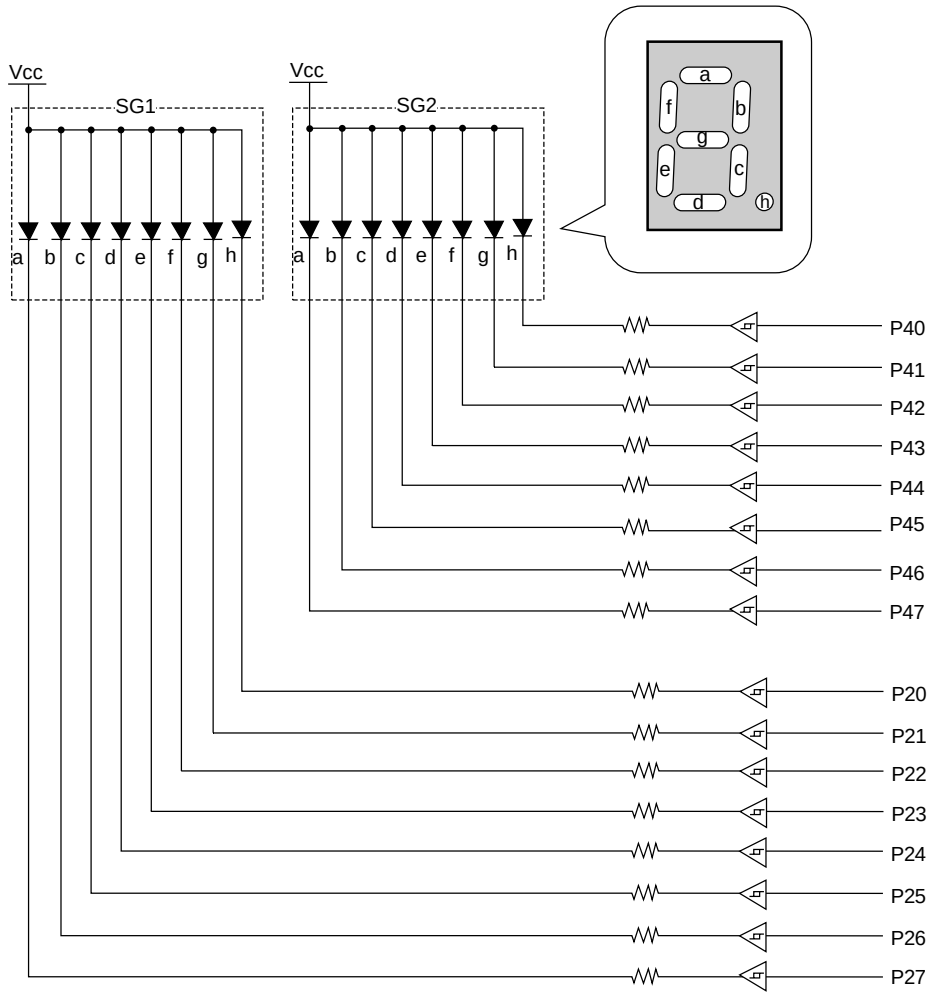


図 4.41　**マスタ** 2 **のバス権獲得理由**

- 本タスク例で使用している 7-segment LED と H8S/2138 との接続図を**図** 4.42 に示します。**図** 4.42 に示すように、ポート 2 およびポート 4 より“ Low ”を出力することにより 7-segment LED を点灯させます。



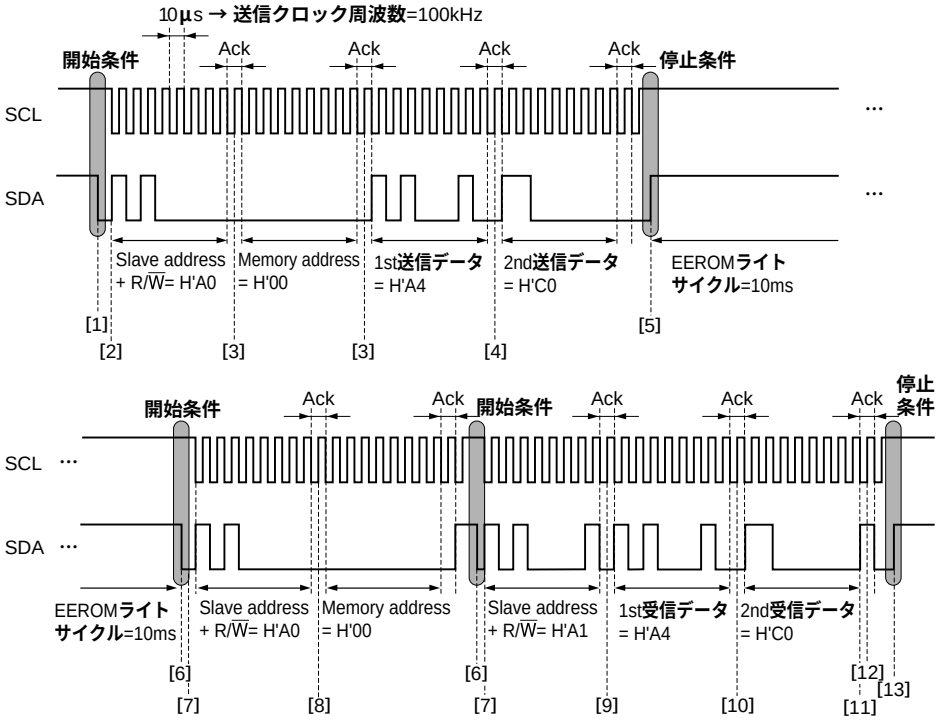
○SG1表示とポート2出力データ

表示	P20	P21	P22	P23	P24	P25	P26	P27	表示	P20	P21	P22	P23	P24	P25	P26	P27
0	1	1	0	0	0	0	0	0	8	1	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	1	9	1	0	0	1	0	0	0	0
2	1	0	1	0	0	1	0	0	A	1	0	0	0	1	0	0	0
3	1	0	1	1	0	0	0	0	B	1	0	0	0	0	0	1	1
4	1	0	0	1	1	0	0	1	C	1	1	0	0	0	1	1	0
5	1	0	0	1	0	0	1	0	D	1	0	1	0	0	0	0	1
6	1	0	0	0	0	0	1	0	E	1	0	0	0	0	1	1	0
7	1	1	0	1	1	0	0	0	F	1	0	0	0	1	1	1	0

図 4.42　7-segment LED **接続図**

4.11.2 動作説明

図 4.43 に動作原理を示します。



	マスタ1処理	マスタ2処理
[1]	開始条件の発行(BBSY=1,SCP=0)	開始条件の発行(BBSY=1、SCP=0)
[2]	開始条件の検出（IRIC=1）、ICDR0に送信データライト、IRIC=1を0にクリア	開始条件の検出(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア
[3]	9クロック目立ち上がり時(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア	9クロック目立ち上がり時(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア
[4]	バス競合負け(AL=1)、スレーブ受信モードに遷移(MST=0、TRS=0)	9クロック目立ち上がり時(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア
[5]	処理なし	停止条件の発行(BBSY=0,SCP=0)
[6]	開始条件の発行(BBSY=0、SCP=1)	処理なし
[7]	開始条件の検出(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア	処理なし
[8]	9クロック目立ち上がり時(IRIC=1)、ICDR0に送信データライト、IRICを0にクリア	処理なし
[9]	開始条件の検出(IRIC=1)、ICDR0をダミーリード、IRICを0にクリア	処理なし
[10]	9クロック目立ち上がり時(IRIC=1)、ACKB=WAIT=1、ICDR0から1バイト目受信データをリード、IRIC=0	処理なし
[11]	8クロック目立ち下がり時(IRIC=1)、マスタ受信モードに設定(TRS=0)、IRICを0にクリア	処理なし
[12]	1クロック目立ち上がり時（IRIC=1)、ICDR0から2バイト目受信データをリード	処理なし
[13]	停止条件の発行(BBSY=0、SCP=0)	処理なし

図 4.43　**バスの競合動作原理**

4.11.3 ソフトウェア説明

(1) モジュール説明

表 4.36 に本タスク例におけるモジュール説明を示します。

表 4.36 モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	スタックポインタの設定、MCU モード設定、IR06 割り込みの設定、割り込みの許可、バス権獲得時はマスタ送信による 2 バイトデータの送信および LED の点灯、バス競合負け時はマスタ受信による 2 バイトデータの受信および 7-segment LED の表示を行う
初期設定	initialize	RAM、ポート、IIC0 の初期設定
開始条件発行	set_start	開始条件の発行
停止条件発行	set_stop	停止条件の発行
Slave address + W 送信	trs_slvadr_a0	EEPROM のスレーブアドレス+W データ（H'A0）の送信
Slave address + R 送信	trs_slvadr_a1	EEPROM のスレーブアドレス+R データ（H'A1）の送信
EEPROM memory address 送信	trs_memadr	EEPROM のメモリアドレスデータ（H'00）の送信
Wait	wait_1	EEPOM ライトサイクル 10ms 待機（20MHz 動作時）

(2) 使用内蔵レジスタ説明

表 4.37 に本タスク例における使用内蔵レジスタ説明を示します。

表 4.37 使用内蔵レジスタ説明

レジスタ		機能	アドレス	設定値
ICDR0		送信／受信データを格納	H'FFDE	−
SAR0	FS	SARX0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDF bit0	0
SARX0	FSX	SAR0 の FSX ビット、DDCSWR の SW ビットと共に、転送フォーマットを設定	H'FFDE bit0	1
ICMR0	MLS	MSB ファーストによるデータ転送の設定	H'FFDF bit7	0
	WAIT	データとアクノリッジ間にウェイトを挿入するか否かを設定	H'FFDF bit6	0/1
	CKS2 to CKS0	STCR の IICX0 ビットと組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFDF bit5 to bit3	CKS2=1 CKS1=0 CKS0=1
	BC2 to BC0	I^2C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	H'FFDF bit2 to bit0	BC2=0 BC1=0 BC0=0

表 4.37 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
ICCR0	ICE	ICMR0, ICDR0/SAR, SARX レジスタのアクセス制御、I²C バスインタフェースの動作（SCL0/SDA0 端子はポート機能）／非動作（SCL/SDA 端子はバス駆動状態）の選択	H'FFD8 bit7	0/1
	IEIC	I²C バスインタフェース割り込み要求を禁止	H'FFD8 bit6	0
	MST	I²C バスインタフェースをマスタモードで使用	H'FFD8 bit5	1
	TRS	I²C バスインタフェースの送信／受信モードの設定	H'FFD8 bit4	0/1
	ACKE	アクノリッジビットが“1”の場合、連続的な転送を中断	H'FFD8 bit3	1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit2	0/1
	IRIC	開始条件の検出、データ送信／受信の終了判定、アクノリッジ=“1”の検出	H'FFD8 bit1	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	H'FFD8 bit0	0
ICSR0	ACKB	送信時、EEPROM からアクノリッジデータを格納、 受信時、EEPROM へ出力するアクノリッジデータを設定	H'FFD9 bit0	−
STCR	IICX0	ICMR0 の CKS2～CKS0 と組み合わせて、転送クロックの周波数を 100kHz に設定	H'FFC3 bit5	1
	IICE	I²C バスインタフェースのデータレジスタおよび制御レジスタの CPU アクセスを許可	H'FFC3 bit4	1
	FLSHE	フラッシュメモリの制御レジスタを非選択状態に設定	H'FFC3 bit3	0
DDCSWR	SWE	IIC チャネル 0 の、フォーマットレスから I²C バスフォーマットへの自動切り替えを禁止	H'FEE6 bit7	0
	SW	IIC チャネル 0 を I²C バスフォーマットで使用	H'FEE6 bit6	0
	IE	フォーマット自動切り替え実行時の割り込みを禁止	H'FEE6 bit5	0
	CLR3 to CLR0	IIC0 の内部状態の初期化を制御	H'FEE6 bit3 to bit0	CLR3=1 CLR2=1 CLR1=1 CLR0=1
MSTPCRL	MSTP7	SCI チャネル 0 のモジュールストップモードの解除	H'FF87 bit7	0
	MSTP4	IIC チャネル 0 のモジュールストップモードの解除	H'FF87 bit4	0
SCR0	CKE1、0	P52/SCK0/SCL0 端子は入出力ポートに設定	H'FFDA bit1、0	CKE1=0 CKE0=0
SMR0	C/$\overline{A}$	SCI0 の動作モードを調歩同期式モードに設定	H'FFD8 bit7	0
SYSCR	INTM1、0	割り込みコントローラの割り込み制御モードを、I ビットによる制御に設定	H'FFC4 bit5、4	INTM1=0 INTM0=0
MDCR	MDS1、0	MD1、0 端子の入力レベルをラッチすることにより MCU 動作モードをモード 3 に設定	H'FFC5 bit1、0	MDS1=1 MDS0=1
P1DDR	P10DDR	P10 端子を出力端子に設定	H'FFB0 bit0	1
P1DR	P10DDR	P10 端子の出力データを設定	H'FFB2 bit0	0/1

表 4.37 使用内蔵レジスタ説明（続き）

レジスタ		機能	アドレス	設定値
P2DDR		ポート 2 を出力端子に設定	H'FFB1	H'FF
P2DR		ポート 2 の出力データを設定	H'FFB3	－
P4DDR		ポート 4 を出力端子に設定	H'FFB5	H'FF
P4DR		ポート 4 の出力データを設定	H'FFB7	－
ISCRH		IRQ6 入力の立ち下がりエッジで割り込み要求を発生	H'FEEC	H'10
ISR	IRQ6F	IRQ6 割り込み要求のステータスを表示	H'FEEB bit6	0/1

（3） 変数説明

表 4.38 に本タスク例における変数説明を示します。

表 4.38 変数説明

変数	機能	データ長	初期値	使用モジュール名
dummy	MDCR リード値	1 バイト	－	main
dt_trs[0]	1 バイト目送信データ	1 バイト	H'F9/A4	main
dt_trs[1]	2 バイト目送信データ	1 バイト	H'C0	main

（4） 使用 RAM 説明

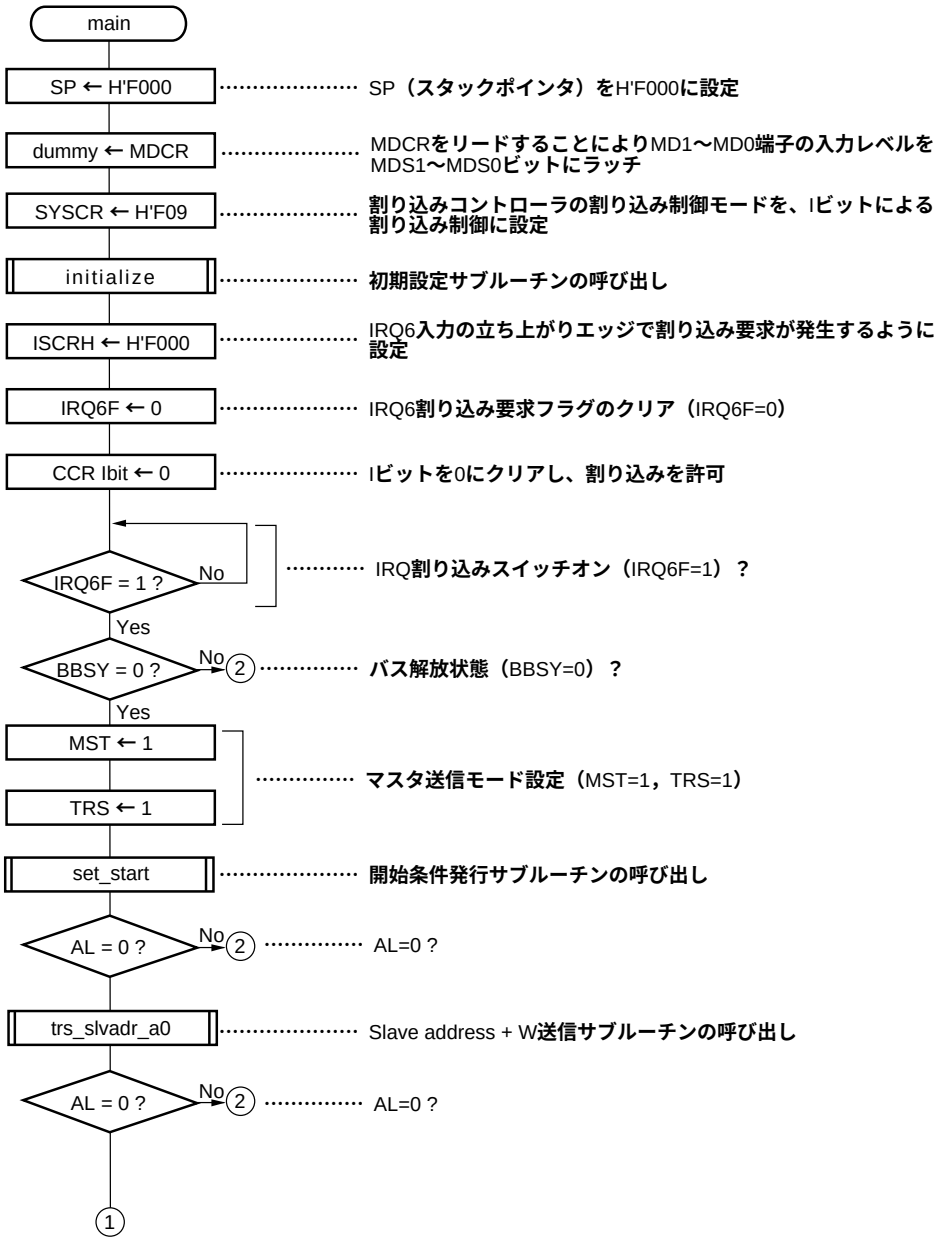
表 4.39 に本タスク例における使用 RAM 説明を示します。

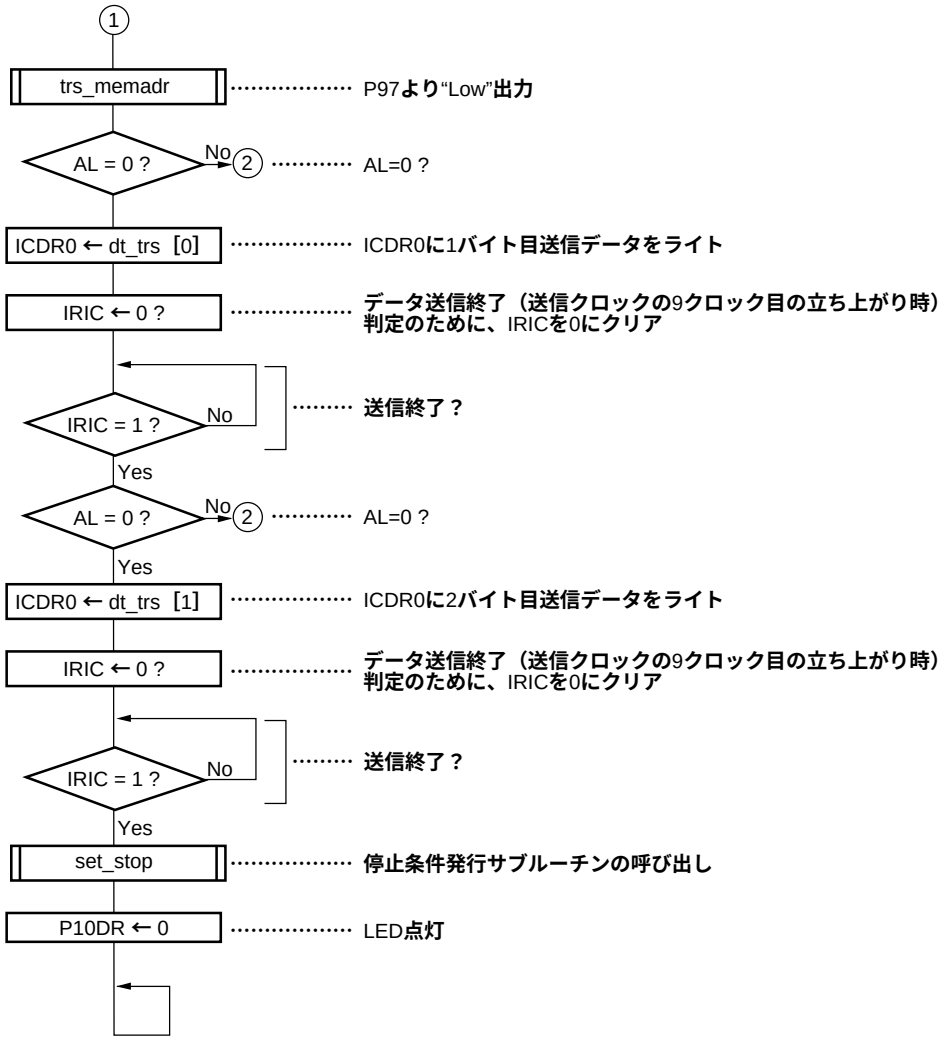
表 4.39 使用 RAM 説明

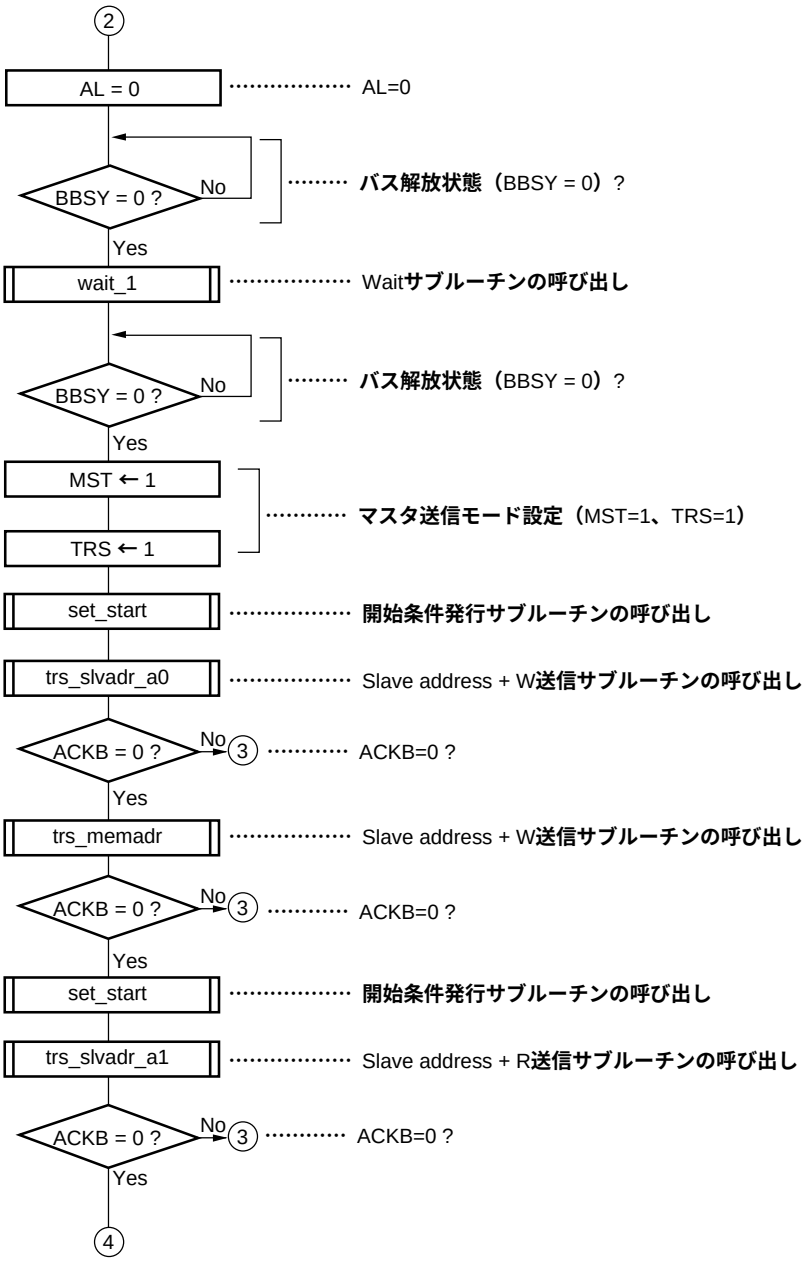
ラベル	機能	データ長	アドレス	使用モジュール名
dt_rec[0]	1 バイト目受信データを格納	1 バイト	H'E100	main、initialize
dt_rec[1]	2 バイト目受信データを格納	1 バイト	H'E101	main、initialize

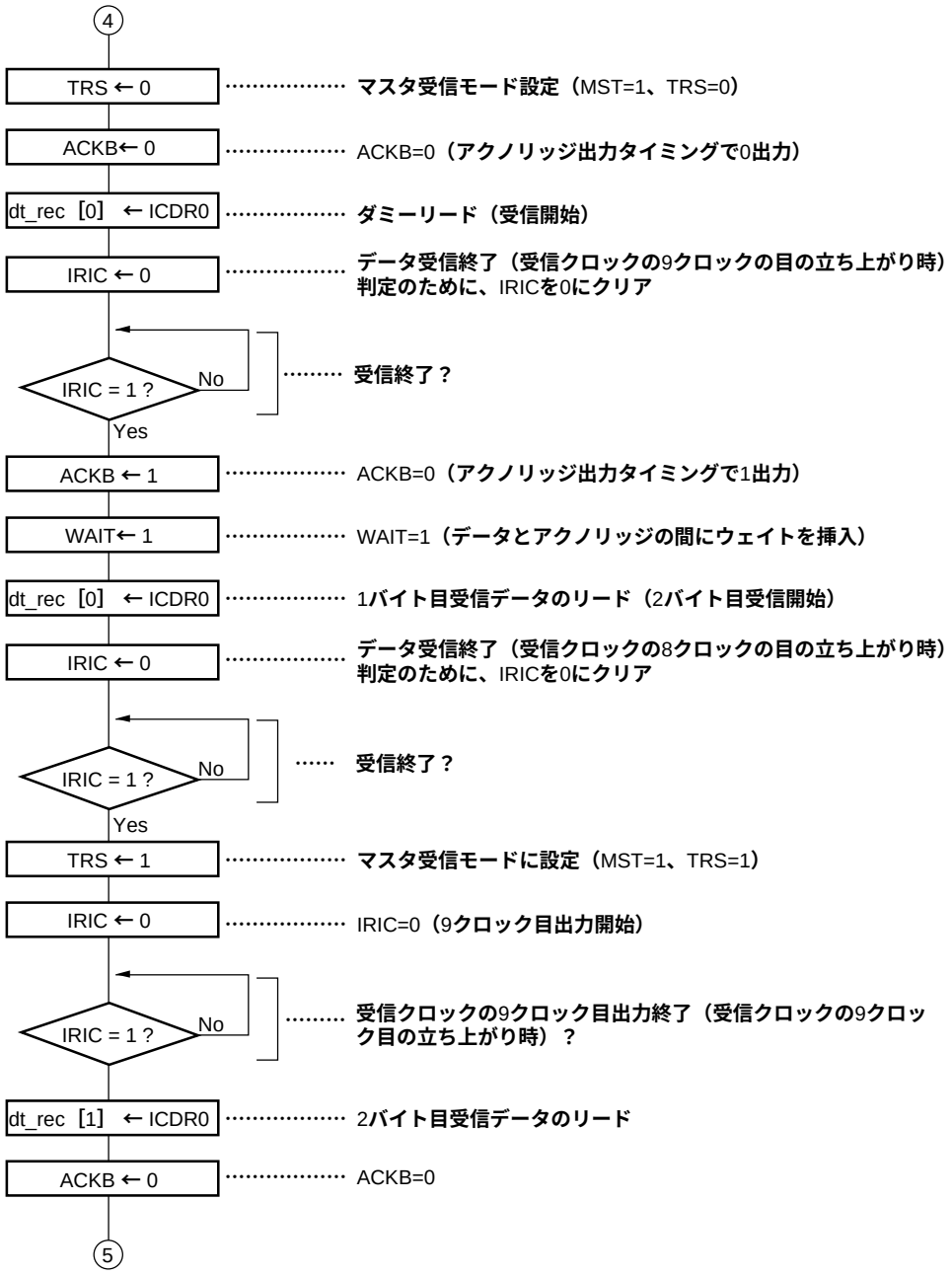
4.11.4 フローチャート

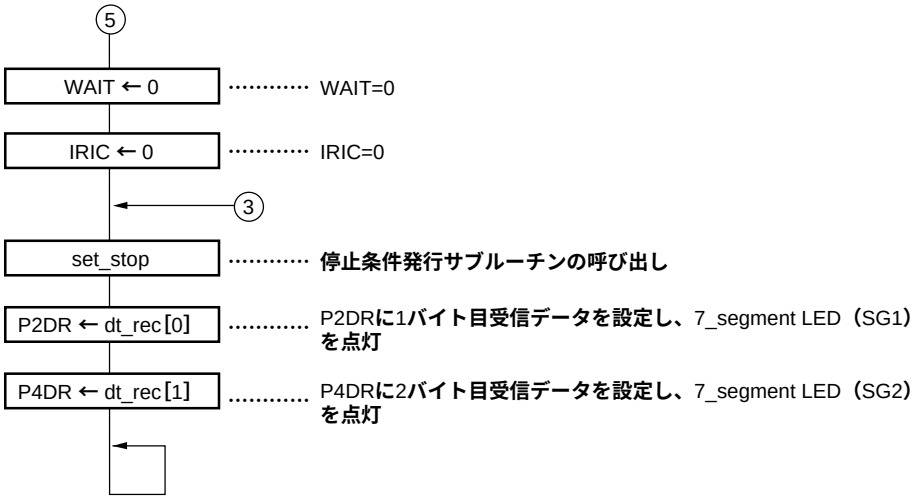
(1) メインルーチン



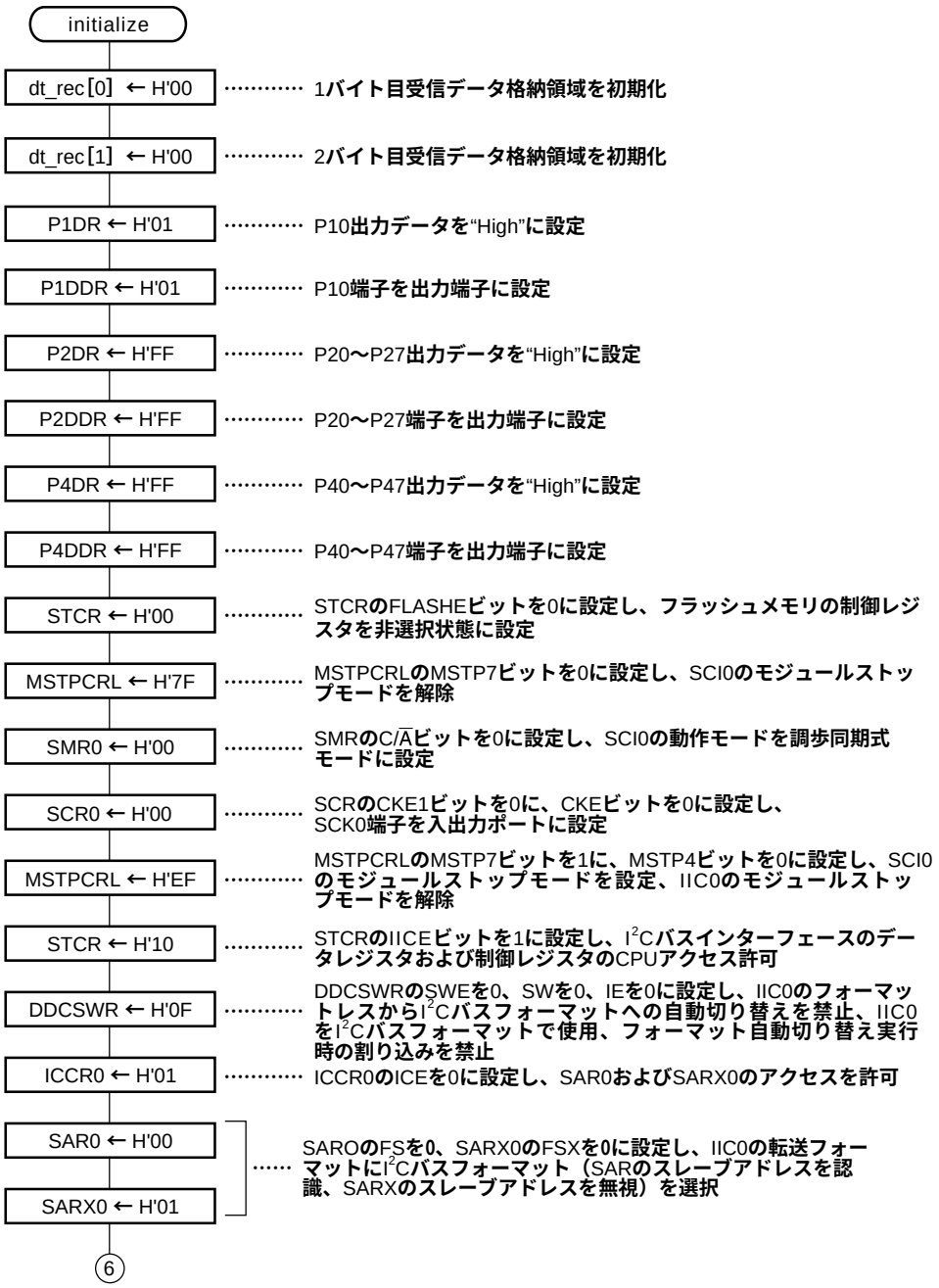

1
trs_memadr
P97より“Low”出力
AL = 0 ?
No
2
AL=0 ?
ICDR0 ← dt_trs [0]
ICDR0に1バイト目送信データをライト
IRIC ← 0 ?
データ送信終了（送信クロックの9クロック目の立ち上がり時）
判定のために、IRICを0にクリア
IRIC = 1 ?
No
送信終了？
Yes
AL = 0 ?
No
2
AL=0 ?
Yes
ICDR0 ← dt_trs [1]
ICDR0に2バイト目送信データをライト
IRIC ← 0 ?
データ送信終了（送信クロックの9クロック目の立ち上がり時）
判定のために、IRICを0にクリア
IRIC = 1 ?
No
送信終了？
Yes
set_stop
停止条件発行サブルーチンの呼び出し
P10DR ← 0
LED点灯

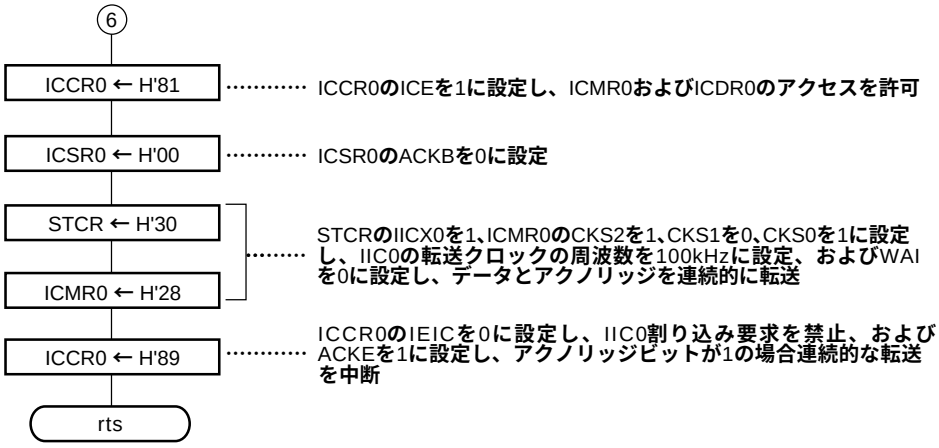

2
AL = 0
AL=0
BBSY = 0 ?
No
バス解放状態（BBSY = 0）?
Yes
wait_1
Waitサブルーチンの呼び出し
BBSY = 0 ?
No
バス解放状態（BBSY = 0）?
Yes
MST ← 1
TRS ← 1
マスタ送信モード設定（MST=1、TRS=1）
set_start
開始条件発行サブルーチンの呼び出し
trs_slvadr_a0
Slave address + W送信サブルーチンの呼び出し
ACKB = 0 ?
No
3
ACKB=0 ?
Yes
trs_memadr
Slave address + W送信サブルーチンの呼び出し
ACKB = 0 ?
No
3
ACKB=0 ?
Yes
set_start
開始条件発行サブルーチンの呼び出し
trs_slvadr_a1
Slave address + R送信サブルーチンの呼び出し
ACKB = 0 ?
No
3
ACKB=0 ?
Yes
4


4
TRS ← 0
マスタ受信モード設定（MST=1、TRS=0）
ACKB← 0
ACKB=0（アクノリッジ出力タイミングで0出力）
dt_rec [0] ← ICDR0
ダミーリード（受信開始）
IRIC ← 0
データ受信終了（受信クロックの9クロックの目の立ち上がり時）
判定のために、IRICを0にクリア
IRIC = 1 ?
No
受信終了？
Yes
ACKB ← 1
ACKB=0（アクノリッジ出力タイミングで1出力）
WAIT← 1
WAIT=1（データとアクノリッジの間にウェイトを挿入）
dt_rec [0] ← ICDR0
1バイト目受信データのリード（2バイト目受信開始）
IRIC ← 0
データ受信終了（受信クロックの8クロックの目の立ち上がり時）
判定のために、IRICを0にクリア
IRIC = 1 ?
No
受信終了？
Yes
TRS ← 1
マスタ受信モードに設定（MST=1、TRS=1）
IRIC ← 0
IRIC=0（9クロック目出力開始）
IRIC = 1 ?
No
受信クロックの9クロック目出力終了（受信クロックの9クロック目の立ち上がり時）？
dt_rec [1] ← ICDR0
2バイト目受信データのリード
ACKB ← 0
ACKB=0
5

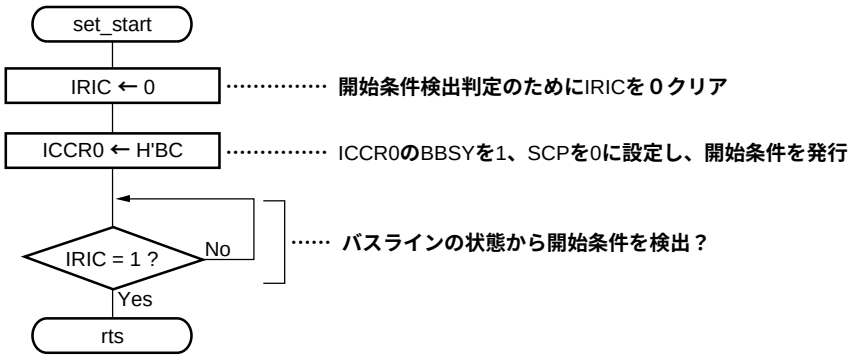

5
WAIT ← 0
WAIT=0
IRIC ← 0
IRIC=0
3
set_stop
停止条件発行サブルーチンの呼び出し
P2DR ← dt_rec[0]
P2DRに1バイト目受信データを設定し、7_segment LED（SG1）を点灯
P4DR ← dt_rec[1]
P4DRに2バイト目受信データを設定し、7_segment LED（SG2）を点灯

（2） 初期設定サブルーチン

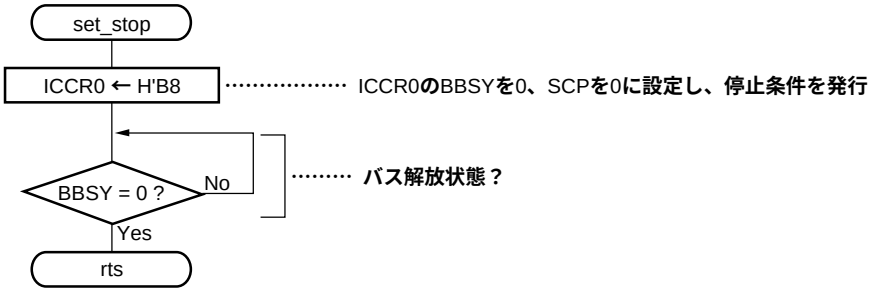



6
ICCR0 ← H'81
ICCR0のICEを1に設定し、ICMR0およびICDR0のアクセスを許可
ICSR0 ← H'00
ICSR0のACKBを0に設定
STCR ← H'30
ICMR0 ← H'28
STCRのIICX0を1、ICMR0のCKS2を1、CKS1を0、CKS0を1に設定し、IIC0の転送クロックの周波数を100kHzに設定、およびWAIを0に設定し、データとアクノリッジを連続的に転送
ICCR0 ← H'89
ICCR0のIEICを0に設定し、IIC0割り込み要求を禁止、およびACKEを1に設定し、アクノリッジビットが1の場合連続的な転送を中断
rts

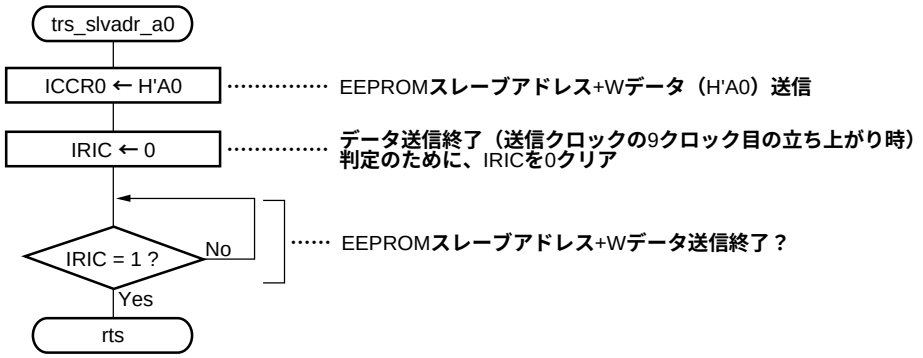
（3） 開始条件発行サブルーチン



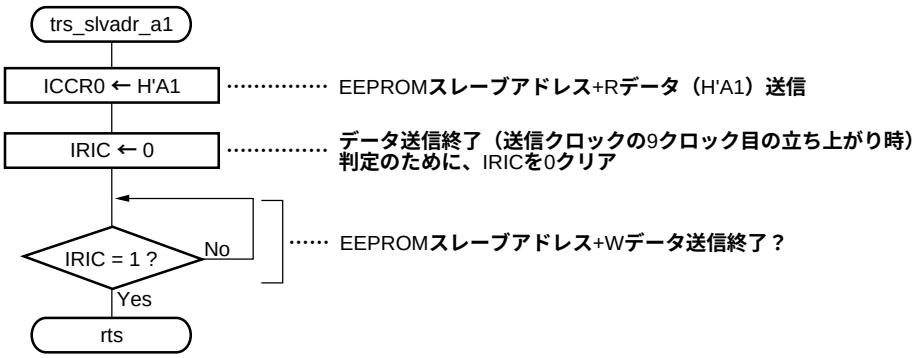
（4） 停止条件発行サブルーチン



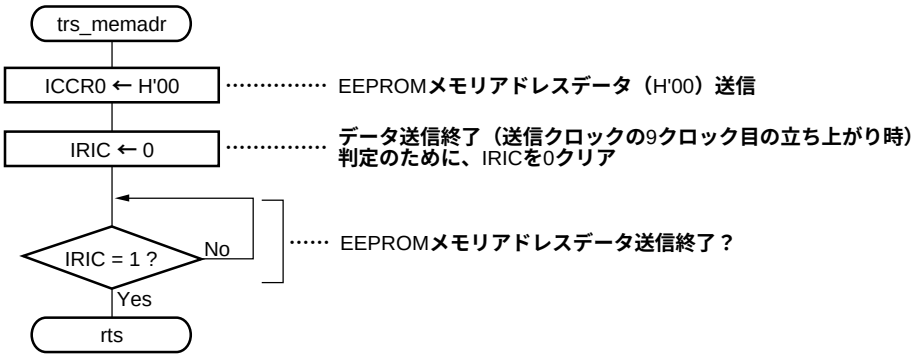
（5） Slave address + W 送信サブルーチン



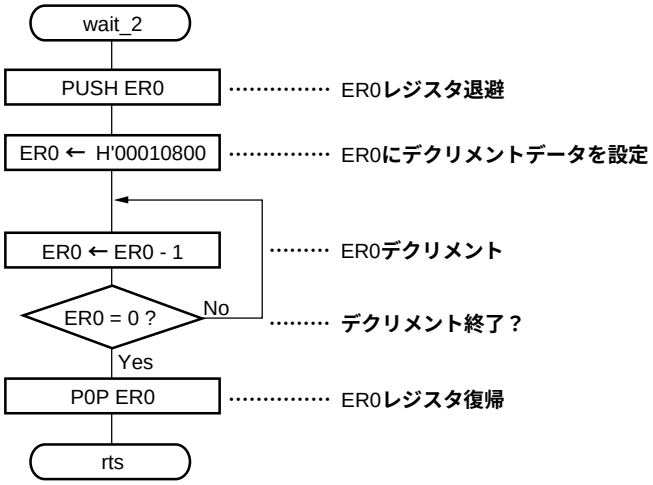
（6） Slave address + R 送信サブルーチン



（7） EEPROM memory address 送信サブルーチン



（8） Wait サブルーチン



4.11.5 【マスタ 1】プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note             *
*      10.Multi master transmit/receive 1        *
*                          File name   :  Mltx1.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* RAM & port & IIC0 initialize */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_slvadr_a1(void);                        /* Slave address + R data transmit */
void trs_memadr(void);                           /* EEPROM memry address data transmit */
void wait_1(void);                               /* EEPROM write cycle(10ms) wait */

/************************************************
*  Data table                                    *
************************************************/
const unsigned char dt_trs[2] =                  /* Transmit data (2 byte) */
{
   0xf9,                                         /* Master 1 1st transmit data */
   0xc0                                          /* Master 1 2nd transmit data */
};

/************************************************
*  RAM allocation                                *
************************************************/
```

（続く）

```
#pragma section ramarea
unsigned char dt_rec[2];                        /* Receive data store area (2byte) */
#pragma section

/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                       ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                          /* MCU mode set */

    SYSCR.BYTE = 0x09;                          /* Interrupt control mode set */
    initialize();                               /* Initialize */

    INTC.ISCR.BYTE.H = 0x10;                    /* IRQ6 edge sense set (faling edge) */
    INTC.ISR.BIT.IRQ6F = 0;                     /* IRQ6 interrupt request flag clear */

    set_imask_ccr(0);                           /* Interrupt enable */

    while(INTC.ISR.BIT.IRQ6F == 0);             /* IRQ interrupt switch on ? */
    INTC.ISR.BIT.IRQ6F = 0;                     /* IRQ6F = 0 */

    if(IIC0.ICCR.BIT.BBSY == 0)                 /* Bus empty (BBSY=0) ? */
    {
        IIC0.ICCR.BIT.MST = 1;                  /* Master transmit mode set */
        IIC0.ICCR.BIT.TRS = 1;                  /* MST = 1, TRS = 1 */
        set_start();                            /* Start condition set */
        if(IIC0.ICSR.BIT.AL == 0)               /* AL = 0 ? */
        {
            trs_slvadr_a0();                    /* Slave address + W data transmit */
            if(IIC0.ICSR.BIT.AL == 0)           /* AL = 0 ? */
            {
                trs_memadr();                   /* EEPROM memory address transmit */
```

（続く）

```
            if(IIC0.ICSR.BIT.AL == 0)         /* AL = 0 ? */
            {
                IIC0.ICDR = dt_trs[0];        /* 1st transmit data write */
                IIC0.ICCR.BIT.IRIC = 0;       /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);/* Transmit end (IRIC=1) ? */
                if(IIC0.ICSR.BIT.AL == 0)     /* AL = 0 ? */
                {
                    IIC0.ICDR = dt_trs[1];    /* 2nd transmit data write */
                    IIC0.ICCR.BIT.IRIC = 0;   /* IRIC = 0 */
                    while(IIC0.ICCR.BIT.IRIC == 0);/* Transmit end (IRIC=1) ? */

                    set_stop();               /* Stop condition set */
                    P1.DR.BIT.B0 = 0;         /* LED on */
                    while(1);                 /* End */
                }
            }
        }
    }
}

IIC0.ICSR.BIT.AL = 0;                         /* AL= 0 */
while(IIC0.ICCR.BIT.BBSY == 1);               /* Transmit end (BBSY=0) ? */
wait_1();                                     /* 10ms wait */

while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */
IIC0.ICCR.BIT.MST = 1;                        /* Master transmit mode set */
IIC0.ICCR.BIT.TRS = 1;                        /* MST = 1, TRS = 1 */

set_start();                                  /* Start condition set */
trs_slvadr_a0();                              /* Slave address + W data transmit */
if(IIC0.ICSR.BIT.ACKB == 0)                   /* ACKB = 0 ? */
{
    trs_memadr();                             /* EEPROM memory address data transmit */
    if(IIC0.ICSR.BIT.ACKB == 0)               /* ACKB = 0 ? */
    {
        set_start();                          /* Re-start condition set */
        trs_slvadr_a1();                      /* Slave address + R data transmit */
```

（続く）

```
            if(IIC0.ICSR.BIT.ACKB == 0)           /* ACKB = 0 ? */
            {
                IIC0.ICCR.BIT.TRS = 0;            /* Master receive mode set (TRS=0) */
                IIC0.ICSR.BIT.ACKB = 0;           /* ACKB = 0 */
                dt_rec[0] = IIC0.ICDR;            /* Dummy read */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* Receive end (IRIC=1) ? */

                IIC0.ICSR.BIT.ACKB = 1;           /* ACKB = 1 */
                IIC0.ICMR.BIT.WAIT = 1;           /* WAIT = 1 */
                dt_rec[0] = IIC0.ICDR;            /* 1st receive data read */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* Receive end (IRIC=1) ? */

                IIC0.ICCR.BIT.TRS = 1;            /* Master transmit mode set (TRS=1) */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* IRIC = 1 ? */

                dt_rec[1] = IIC0.ICDR;            /* 2nd receive data read */
                IIC0.ICSR.BIT.ACKB = 0;           /* ACKB = 0 */
                IIC0.ICMR.BIT.WAIT = 0;           /* WAIT = 0 */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
            }
        }
    }
    set_stop();                                   /* Stop condition set */

    P2.DR.BYTE = dt_rec[0];                       /* SG1 on */
    P4.DR.BYTE = dt_rec[1];                       /* SG2 on */

   while(1);                                      /* End */
}

/************************************************
*  initialize : RAM & Port & IIC0 Initialize      *
************************************************/
void initialize(void)
```

(続く)

```
{
    dt_rec[0] = 0x00;                           /* Receive data store area initialize */
    dt_rec[1] = 0x00;

    P1.DR.BYTE = 0x01;                          /* Port 1 initialize */
    P1.DDR = 0x01;
    P2.DR.BYTE = 0xff;                          /* Port 2 initialize */
    P2.DDR = 0xff;
    P4.DR.BYTE = 0xff;                          /* Port 4 initialize */
    P4.DDR = 0xff;

    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x38;                       /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  set_start : Start condition set              *
************************************************/
void set_start(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                      /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Start condition set (IRIC=1) ? */
}
```

（続く）

```
/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
    IIC0.ICCR.BYTE = 0xb8;                      /* Stop condition set (BBSY=0, SCP=0) */
    while(IIC0.ICCR.BIT.BBSY == 1);             /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
{
    IIC0.ICDR = 0xa0;                           /* Slave address + W data(H'A0) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_slvadr_a1 : Slave addres + R data transmit *
************************************************/
void trs_slvadr_a1(void)
{
    IIC0.ICDR = 0xa1;                           /* Slave address + R data(H'A1) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}
/************************************************
* trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
    IIC0.ICDR = 0x00;                           /* EEPROM memory address data(H'00) write */
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Transmit end (IRIC=1) ? */
}
```

（続く）

```
/************************************************
*   wait_1 : 10ms wait                          *
************************************************/
void wait_1(void)
{
#pragma asm
       push.l  er0                              ;Push ER0
       mov.l   #h'00010800,er0                  ;Decrement data set
wait1_1:
       dec.l   #1,er0                           ;Decrement
       bne     wait1_1                          ;Decrement end ?
       pop.l   er0                              ;Pop ER0
#pragma endasm
}
```

4.11.6 【マスタ 2】プログラムリスト

```
/************************************************
*  H8S/2138 IIC bus application note              *
*      10.Multi master transmit/receive 2         *
*                          File name   :  Mltx2.c *
*                          Fai         :  20MHz   *
*                          Mode        :  3       *
************************************************/
#include <stdio.h>
#include <machine.h>
#include "2138s.h"

/************************************************
*  Prototype                                     *
************************************************/
void main(void);                                 /* Main routine */
void initialize(void);                           /* RAM & port & IIC0 initialize */
void set_start(void);                            /* Start condition set */
void set_stop(void);                             /* Stop condition set */
void trs_slvadr_a0(void);                        /* Slave address + W data transmit */
void trs_slvadr_a1(void);                        /* Slave address + R data transmit */
void trs_memadr(void);                           /* EEPROM memry address data transmit */
void wait_1(void);                               /* EEPROM write cycle(10ms) wait */

/************************************************
*  Data table                                    *
************************************************/
const unsigned char dt_trs[2] =                  /* Transmit data (2 byte) */
{
   0xa4,                                         /* Master 2 1st transmit data */
   0xc0                                          /* Master 2 2nd transmit data */
};

/************************************************
*  RAM allocation                                *
************************************************/
```

（続く）

```
#pragma section ramerea
unsigned char dt_rec[2];                        /* Receive data store area (2byte) */
#pragma section

/************************************************
*  main : Main routine                          *
************************************************/
void main(void)
#pragma asm
       mov.l   #h'f000,sp                       ;Stack pointer initialize
#pragma endasm
{
    unsigned char dummy;
    dummy = MDCR.BYTE;                          /* MCU mode set */

    SYSCR.BYTE = 0x09;                          /* Interrupt control mode set */
    initialize();                               /* Initialize */

    INTC.ISCR.BYTE.H = 0x10;                    /* IRQ6 edge sense set (faling edge) */
    INTC.ISR.BIT.IRQ6F = 0;                     /* IRQ6 interrupt request flag clear */

    set_imask_ccr(0);                           /* Interrupt enable */

    while(INTC.ISR.BIT.IRQ6F == 0);             /* IRQ interrupt switch on ? */
    INTC.ISR.BIT.IRQ6F = 0;                     /* IRQ6F = 0 */

    if(IIC0.ICCR.BIT.BBSY == 0)                 /* Bus empty (BBSY=0) ? */
    {
        IIC0.ICCR.BIT.MST = 1;                  /* Master transmit mode set */
        IIC0.ICCR.BIT.TRS = 1;                  /* MST = 1, TRS = 1 */
        set_start();                            /* Start condition set */
        if(IIC0.ICSR.BIT.AL == 0)               /* AL = 0 ? */
        {
            trs_slvadr_a0();                    /* Slave address + W data transmit */
            if(IIC0.ICSR.BIT.AL == 0)           /* AL = 0 ? */
            {
                trs_memadr();                   /* EEPROM memory address transmit */
```

（続く）

```
            if(IIC0.ICSR.BIT.AL == 0)         /* AL = 0 ? */
            {
                IIC0.ICDR = dt_trs[0];        /* 1st transmit data write */
                IIC0.ICCR.BIT.IRIC = 0;       /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);/* Transmit end (IRIC=1) ? */
                if(IIC0.ICSR.BIT.AL == 0)     /* AL = 0 ? */
                {
                    IIC0.ICDR = dt_trs[1];    /* 2nd transmit data write */
                    IIC0.ICCR.BIT.IRIC = 0;   /* IRIC = 0 */
                    while(IIC0.ICCR.BIT.IRIC == 0);/* Transmit end (IRIC=1) ? */

                    set_stop();               /* Stop condition set */
                    P1.DR.BIT.B0 = 0;         /* LED on */
                    while(1);                 /* End */
                }
            }
        }
    }
}

IIC0.ICSR.BIT.AL = 0;                         /* AL= 0 */
while(IIC0.ICCR.BIT.BBSY == 1);               /* Transmit end (BBSY=0) ? */
wait_1();                                     /* 10ms wait */

while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */
IIC0.ICCR.BIT.MST = 1;                        /* Master transmit mode set */
IIC0.ICCR.BIT.TRS = 1;                        /* MST = 1, TRS = 1 */

set_start();                                  /* Start condition set */
trs_slvadr_a0();                              /* Slave address + W data transmit */
if(IIC0.ICSR.BIT.ACKB == 0)                   /* ACKB = 0 ? */
{
    trs_memadr();                             /* EEPROM memory address data transmit */
    if(IIC0.ICSR.BIT.ACKB == 0)               /* ACKB = 0 ? */
    {
        set_start();                          /* Re-start condition set */
        trs_slvadr_a1();                      /* Slave address + R data transmit */
```

（続く）

```
            if(IIC0.ICSR.BIT.ACKB == 0)           /* ACKB = 0 ? */
            {
                IIC0.ICCR.BIT.TRS = 0;            /* Master receive mode set (TRS=0) */
                IIC0.ICSR.BIT.ACKB = 0;           /* ACKB = 0 */
                dt_rec[0] = IIC0.ICDR;            /* Dummy read */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* Receive end (IRIC=1) ? */

                IIC0.ICSR.BIT.ACKB = 1;           /* ACKB = 1 */
                IIC0.ICMR.BIT.WAIT = 1;           /* WAIT = 1 */
                dt_rec[0] = IIC0.ICDR;            /* 1st receive data read */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* Receive end (IRIC=1) ? */

                IIC0.ICCR.BIT.TRS = 1;            /* Master transmit mode set (TRS=1) */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
                while(IIC0.ICCR.BIT.IRIC == 0);   /* IRIC = 1 ? */

                dt_rec[1] = IIC0.ICDR;            /* 2nd receive data read */
                IIC0.ICSR.BIT.ACKB = 0;           /* ACKB = 0 */
                IIC0.ICMR.BIT.WAIT = 0;           /* WAIT = 0 */
                IIC0.ICCR.BIT.IRIC = 0;           /* IRIC = 0 */
            }
        }
    }
    set_stop();                                   /* Stop condition set */

    P2.DR.BYTE = dt_rec[0];                       /* SG1 on */
    P4.DR.BYTE = dt_rec[1];                       /* SG2 on */

   while(1);                                      /* End */
}

/************************************************
*  initialize : RAM & Port & IIC0 Initialize      *
************************************************/
void initialize(void)
```

（続く）

```
{
    dt_rec[0] = 0x00;                           /* Receive data store area initialize */
    dt_rec[1] = 0x00;

    P1.DR.BYTE = 0x01;                          /* Port 1 initialize */
    P1.DDR = 0x01;
    P2.DR.BYTE = 0xff;                          /* Port 2 initialize */
    P2.DDR = 0xff;
    P4.DR.BYTE = 0xff;                          /* Port 4 initialize */
    P4.DDR = 0xff;

    STCR.BYTE = 0x00;                           /* FLSHE = 0 */
    MSTPCR.BYTE.L = 0x7f;                       /* SCI0 module stop mode reset */
    SCI0.SMR.BYTE = 0x00;                       /* SCL0 pin function set */
    SCI0.SCR.BYTE = 0x00;
    MSTPCR.BYTE.L = 0xef;                       /* IIC0 module stop mode reset */
    STCR.BYTE = 0x10;                           /* IICE = 1 */
    DDCSWR.BYTE = 0x0f;                         /* IIC bus format initialize */
    IIC0.ICCR.BYTE = 0x01;                      /* ICE = 0 */
    IIC0.SAR.BYTE = 0x38;                       /* FS = 0 */
    IIC0.SARX.BYTE = 0x01;                      /* FSX = 1 */
    IIC0.ICCR.BYTE = 0x81;                      /* ICE = 1 */
    IIC0.ICSR.BYTE = 0x00;                      /* ACKB = 0 */
    STCR.BYTE = 0x30;                           /* IICX0 = 1 */
    IIC0.ICMR.BYTE = 0x28;                      /* Transfer rate = 100kHz */
    IIC0.ICCR.BYTE = 0x89;                      /* IEIC = 0, ACKE = 1 */
}

/************************************************
*  set_start : Start condition set             *
************************************************/
void set_start(void)
{
    IIC0.ICCR.BIT.IRIC = 0;                     /* IRIC = 0 */
    IIC0.ICCR.BYTE = 0xbc;                      /* Start condition set (BBSY=1,SCP=0) */
    while(IIC0.ICCR.BIT.IRIC == 0);             /* Start condition set (IRIC=1) ? */
}
```

（続く）

```
/************************************************
*  set_stop : Stop condition set                *
************************************************/
void set_stop(void)
{
   IIC0.ICCR.BYTE = 0xb8;                        /* Stop condition set (BBSY=0,SCP=0) */
   while(IIC0.ICCR.BIT.BBSY == 1);               /* Bus empty (BBSY=0) ? */
}

/************************************************
*  trs_slvadr_a0 : Slave addres + W data transmit *
************************************************/
void trs_slvadr_a0(void)
{
   IIC0.ICDR = 0xa0;                             /* Slave address + W data(H'A0) write */
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Transmit end (IRIC=1) ? */
}

/************************************************
*  trs_slvadr_a1 : Slave addres + R data transmit *
************************************************/
void trs_slvadr_a1(void)
{
   IIC0.ICDR = 0xa1;                             /* Slave address + R data(H'A1) write */
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Transmit end (IRIC=1) ? */
}
/************************************************
* trs_memadr : EEPROM memory address data transmit*
************************************************/
void trs_memadr(void)
{
   IIC0.ICDR = 0x00;                             /* EEPROM memory address data(H'00) write */
   IIC0.ICCR.BIT.IRIC = 0;                       /* IRIC = 0 */
   while(IIC0.ICCR.BIT.IRIC == 0);               /* Transmit end (IRIC=1) ? */
}
```

（続く）

```
/************************************************
*  wait_1 : 10ms wait                           *
************************************************/
void wait_1(void)
{
#pragma asm
       push.l  er0                              ;Push ER0
       mov.l   #h'00010800,er0                  ;Decrement data set
wait1_1:
       dec.l   #1,er0                           ;Decrement
       bne     wait1_1                          ;Decrement end ?
       pop.l   er0                              ;Pop ER0
#pragma endasm
}
```

内蔵インタフェース　I²Cバスインタフェース編
アプリケーションノート

発行年月　1994 年 3 月　第 1 版
　　2002 年 1 月　第 2 版
発　　行　株式会社　日立製作所
　　半導体グループビジネス企画本部
編　　集　株式会社　日立小平セミコン
　　技術ドキュメントグループ

内蔵インタフェース　I^2C バスインタフェース編
アプリケーションノート

RENESAS
ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

ADJ-502-057A