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。

H8/300L SLP シリーズ

IIC バス EEPROM との接続

要旨

SCL（Serial Clock）と SDA（Serial Data）の 2 線により接続された IIC バス方式シリアル EEPROM と接続します。マイコン R0M 上のデータを EEPOROM に Write し，EEPROM に Write されたデータを再びマイコン RAM に Read します。LSB ファースト方式による転送を行います。

動作確認デバイス

H8/38024

目次

1. 仕様

(1) 図 1 に示すように，SCL（Serial Clock）と SDA（Serial Data）の 2 線により接続された IIC バス方式シリアル EEPROM と接続します。
(2) マイコン R0M 上のデータを EEPOROM に Write し, EEPROM に Write されたデータを再びマイコン RAM に Read します。
(3) LSB ファースト方式による転送を行います。

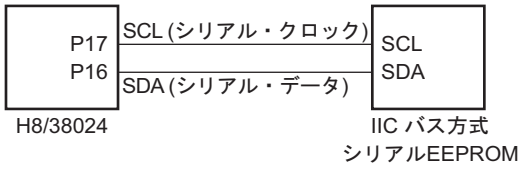


図 1　EEPROM との接続例

2. 考え方

(1) 本タスク例で使用する IIC バス EEPROM のバス・タイミングを図に示します。

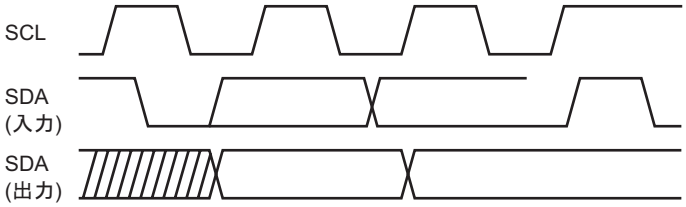


図 2　IIC バス EEPROM のバス・タイミング

(2) 使用する IIC バス EEPROM は，図 2 に示すようなバス・タイミングです。本タスク例では，ソフトウェアにより P17 端子レベルを"High", "Low"に設定し, SCL に出力するシリアルクロックを生成しています。ソフトウェアにより生成したシリアルクロックに同期して，P16 端子からシリアルデータを出力/入力し，IIC バス EEPROM へのアクセスを実現しています。図 3 に P17 端子，および P16 端子レベルのタイミング波形を示します。

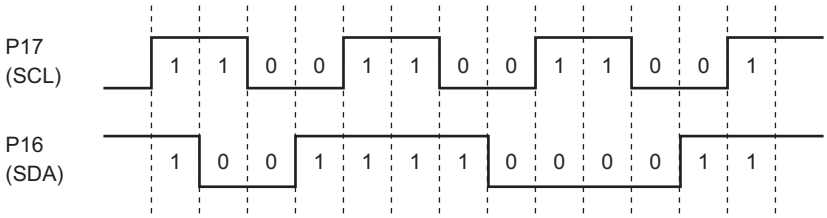


図 3　P17 端子，および P16 端子レベルのタイミング波形

3. 使用機能説明

(1) 本タスク例では, 図 4 に示すように H8/38024 と IIC バス EEPROM を接続します。また, IIC バス EEPROM のピン説明を表 1 に示します

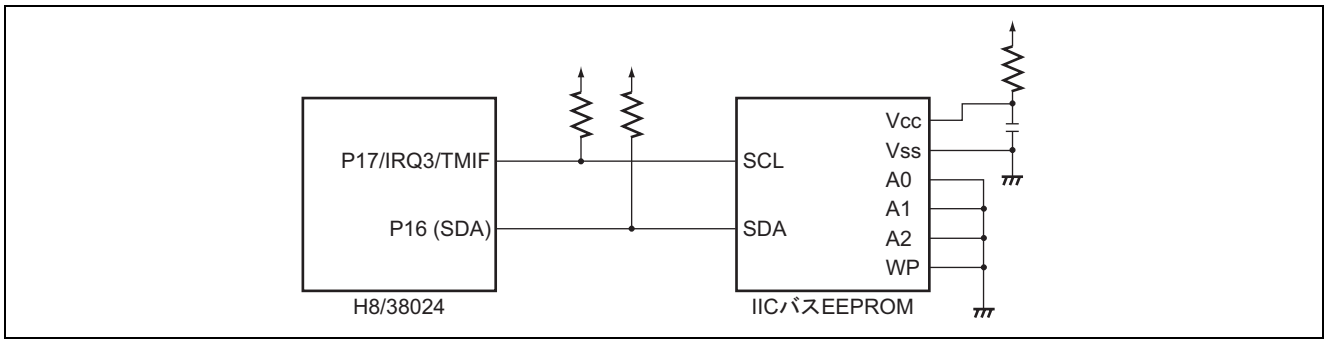


図 4　H8/38024 と IIC バス EEPROM の接続図

表 1　IIC バス EEPROM のピン説明

Pin 名称	機能
A0～A2	**デバイス・アドレス**
SCL	**シリアル・クロック入力**
SDA	**シリアル・データ入力/出力**
WP	**ライト・プロテクト**
Vcc	Vcc
Vss	GND

(2) 図 5 に示すようなブロック図により，H8/38024 と IIC バス EEPROM を接続します。以下に H8/38024 の機能について説明します。

(a) ポート 1 は，4 ビットの入出力ポートで，P17/~IRQ3/TMIF，P16，P14/~IRQ4/~ADTRG，P13/TMIG の 4 つの端子により構成されています。

(b) P17/~IRQ3/TMIF 端子を IIC バス EEPROM の SCL 端子に接続し，シリアル・クロックの出力端子として使用します。

(c) P16 端子を IIC バス EEPROM の SDA 端子に接続し，シリアル・データの入出力端子として使用します。

(d) P17/~IRQ3/TMIF 端子は，P17 入力端子に設定します。

(e) P16 端子は，PCR1 のポートコントロールレジスタ 16（PCR16）を"0"に設定すると P16 入力端子として，PCR16 を"1"に設定すると P16 出力端子として機能します。

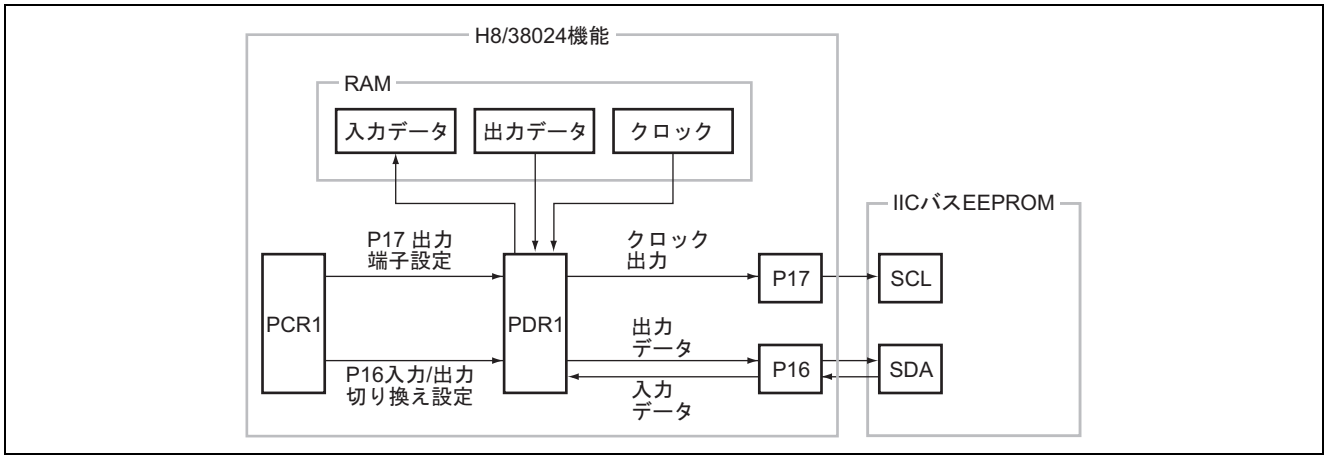


図 5　IIC バス EEPROM との接続のブロック図

(3) 本タスク例の機能割り付けを表 2 に示します。表 2 に示すように機能を割り付け，IIC バス EEPROM と接続します。

表 2　機能割り付け

機能	機能割り付け
P17 端子	シリアル・クロック出力
P17	P17 端子のデータを格納
PCR17	P17 入出力端子機能の設定
P16 端子	シリアル・データ入出力
P16	P16 端子のデータを格納
PCR16	P16 入出力端子機能の設定

(4) 以下に本タスク例で使用している IIC バス EEPROM の仕様について説明します。

(a) IIC バス EEPROM は，2 線式シリアルインタフェースの EEPROM（電気的に書き換え可能な ROM）です。本タスク例では，Renesas 社製 64k ビット EEPROM（HN58X2464FPI），または FAIRCHILD 社製 2k ビット EEPROM（FM24C03UFLM8）を使用しています。

(b) 以下に本タスク例で使用している EEPROM の特徴を示します。

Renesas 社製 64k ビット EEPROM（HN58X2464FPI）

単一電源		1.8V～5.5V
2 線式シリアルインタフェース		IIC バスインターフェース
動作周波数		400kHz
消費電流	スタンバイ時	3μA（max）
	読み出し時	1mA（max）
	書き換え時	3mA（max）
ページ書き換え		ページサイズ 32 バイト
書き換え時間		10ms（2.7～5.5V 以上）/ 15ms（1.8V～2.7V）
書き換え回数		10^5 回（ページ書き換え時）

FAIRCHILD 社製 2k ビット EEPROM（FM24C03UFLM8）

単一電源		2.7V～5.5V
2 線式シリアルインタフェース		IIC バスインターフェース
動作周波数		400kHz
消費電流	スタンバイ時	10μA（max）
	読み出し時	1mA（max）
	書き換え時	1mA（max）
ページ書き換え		ページサイズ 16 バイト
書き換え時間		10ms（4.5～5.5V 以上）/ 15ms（2.7V～4.5V）
書き換え回数		10^6 回（ページ書き換え時）

(c) Read，Write の動作を開始するには，SCL 入力が High の期間に，SDA 入力を High から Low にするスタート・コンディションにする必要があります。
また，SCL 入力が High の期間に，SDA 入力を Low から High にすることで，ストップ・コンディションになります。Read の場合，ストップ・コンディションを入力すると Read が終了し，スタンバイ状態になります。Write の場合は，ストップ・コンディション入力で書き換えデータの入力終了となり，メモリへの書き込みを書き換え時間（twc）の期間実施した後，スタンバイモードになります。スタート・コンディション，ストップ・コンディションのタイミング波形を図 6 に示します。

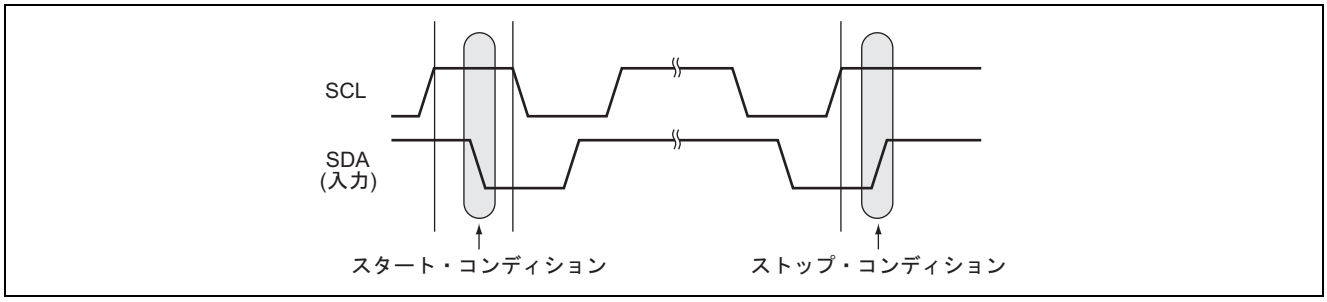


図 6　スタート・コンディション，およびストップ・コンディション信号出力タイミング波形

(d) アドレス情報，Read 情報等のシリアルデータは，8 ビット単位で送受信が行われます。Acknowledge 信号は，この 8 ビットのデータが正常に送信または受信されたことを示す信号で，SCL の 9 クロック目に受信側が"0"を出力します。送信側は，この 9 クロック目で Acknowledge 信号を受信するために，バスを開放します。
EEPROM から見ると，Write の場合は全て受信となるため，8 ビットの受信が完了したら，9 クロック目に EEPROM から Acknowledge 信号の"0"を出力します。Read の場合は，スタート・コンディションの後の 8 ビット受信後に Acknowledge 信号の"0"を出力します。これに続いて，EEPROM は Read データを 8 ビット単位で出力しますが，出力後はバスを開放し，マスタ側から Acknowledge 信号の"0"が送られるのを待ちます。Acknowledge 信号の"0"を検出すると，EEPROM は次のアドレスの Read データを出力します。Acknowledge 信号の"0"が検出されずにストップ・コンディションを受信すると，Read 動作を終了し，スタンバイ状態になります。なお，Acknowledge 信号の"0"が検出されず，かつストップ・コンディションも送られてこない場合は，データを出力せずにバス解放状態を持続します。
Acknowledge 信号のタイミング波形を図 7 に示します。

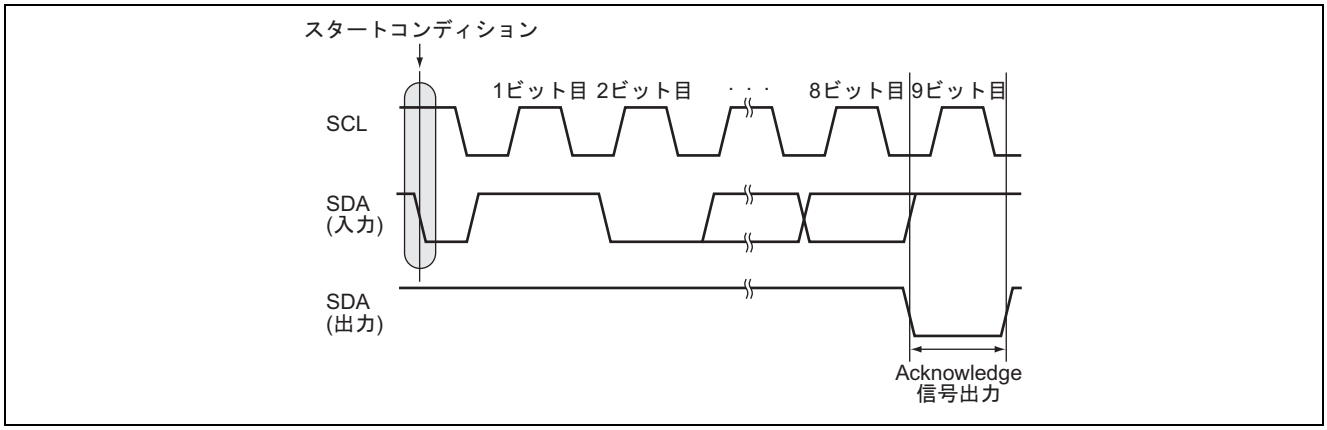


図 7　Acknowledge 信号出力タイミング波形

(e) スタート・コンディションに続いて 8 ビットのデバイス・アドレス・ワードを入力します。この入力でデバイスは Read，Write の動作を開始します。デバイス・アドレス・ワードはデバイス・コード 4 ビット，デバイス・アドレス・コード 3 ビット，Read/Write コード 1 ビットの 3 つのコードで構成されています。

デバイス・アドレス・ワードの上位 4 ビットは，デバイス・タイプを識別するデバイス・コードで，本タスク例で使用している EEPROM では，"1010"の固定コードになります。

デバイス・コードに続けてデバイス・アドレス・コード 3 ビットを A2，A1，A0 の順に入力します。デバイス・アドレス・コードはバスに最大 8 ケ接続されたデバイスのうち，どれを選択するかを決定します。本タスク例で使用する EEPROM のデバイス・アドレス・コードは"000"に設定しています。

デバイス・アドレス・ワードの 8 ビット目は R/W コードです。"0"入力の場合は Write 動作，"1"入力の場合は Read 動作になります。なお，デバイス・コードが"1010"でない場合，もしくはデバイス・アドレス・コードが一致しない場合は，Read/Write 動作に入らず，スタンバイモードになります。図 8 にデバイス・アドレス・ワードについて示します

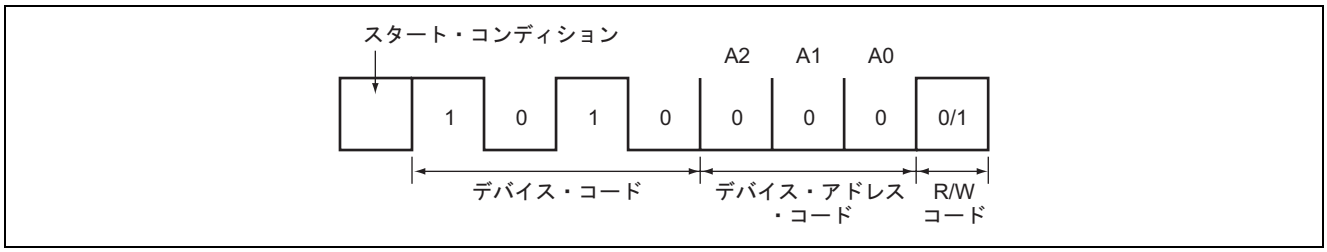


図 8　デバイス・アドレス・ワード

(f) 本タスク例では，32 バイト（または 16 バイト）までの任意のバイト数を一度に書き換える Page Write 機能を使用して Write 動作を行います。スタート・コンディション→デバイス・アドレス・ワード→メモリ・アドレス（n）→Write データ（Dn）の順に，9 ビットごとの Acknowledge 信号の"0"出力を確認しながら入力します。Write データ（Dn+1）を入力すると，Page Write モードに入ります。Write データ（Dn+1）を入力した時点で，ページ内アドレス（a0～a4）は自動的にインクリメントされ（n+1）番地になります。このように，Write データを次々と入力することができ，Write データ入力ごとにページ内アドレスがインクリメントされ，最大 32 バイト（または 16 バイト）の Write データを入力できます。ページ内アドレス（a0～a4）がページの最終番地に達した場合は，アドレスは"Roll Over"して，ページの先頭アドレスに戻ります。"Roll Over"した場合は，同一アドレスに Write データが 2 度（以上）入力されることになりますが，最後に入力した Write データが有効になります。ストップ・コンディションを入力すると，Write データの入力を終了し，書き換え動作に入ります。図 7 に Page Write 動作について示します。

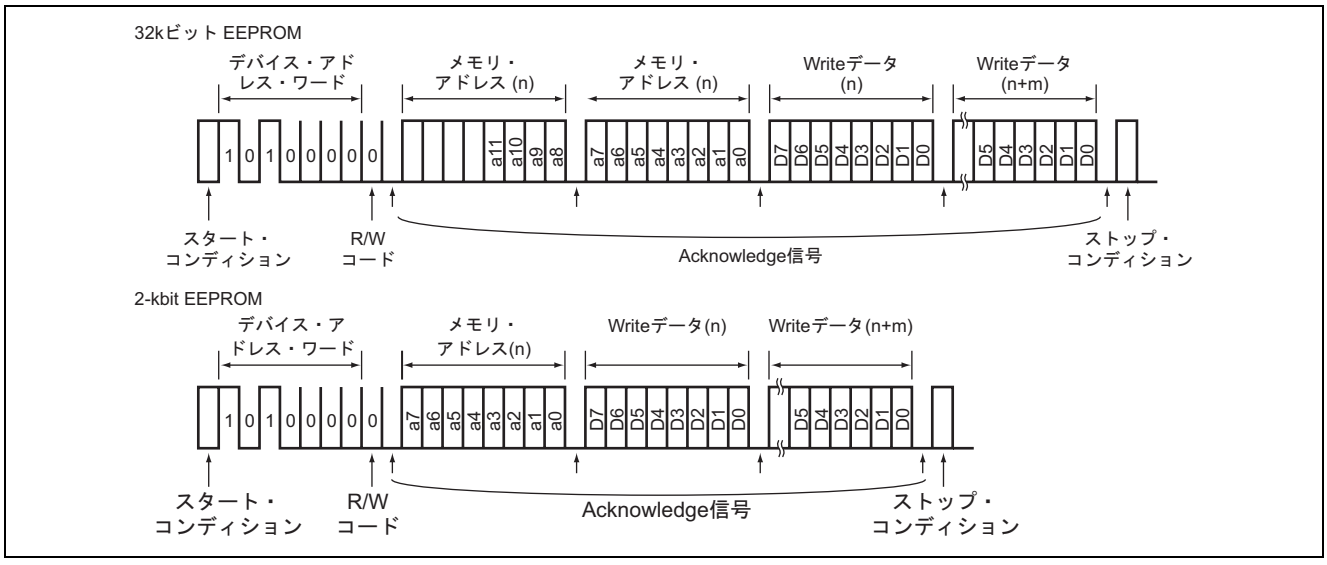


図 9　Page Write 動作

(g) EEPROM が書き換え中か否かを判定する機能として，Acknowledge Polling があります。書き換え期間中にスタート・コンディションに続いてデバイス・アドレス・ワード 8 ビットを入力します。Acknowledge Polling の場合，Read/Write コードは"1"/"0"のどちらでもかまいません。9 ビット目の Acknowledge 信号で書き換え中か否かを判定します。Acknowledge 信号が"1"のときは書き換え中，Acknowledge 信号が"0"のときは書き換え終了を示します。Acknowledge Polling は，Write データ入力後，ストップ・コンディションが入力された時点から機能します。

(h) 本タスク例では，データを連続して Read する Sequential Read モードを使用して Read 動作を行います。ダミーの Write モードで Read するデータの先頭アドレスを入力します。8 ビットのデータを出力した後，Acknowledge 信号の"0"を入力すると，アドレスがインクリメントされ，次の 8 ビットのデータが出力されます。データ出力後に Acknowledge 信号の"0"の入力を続けるとアドレスをインクリメントしながら次々とデータを出力します。アドレスが最終アドレスになった場合は，0 番地に"Roll Over"します。"Roll Over"後も Sequential Read が可能です。動作を終了するには，Acknowledge 信号の"1"（Acknowledge 信号の入力をせずに，バスを解放しても可）→ストップ・コンディションの順で入力します。図 10 に Sequential Read 動作について示します。

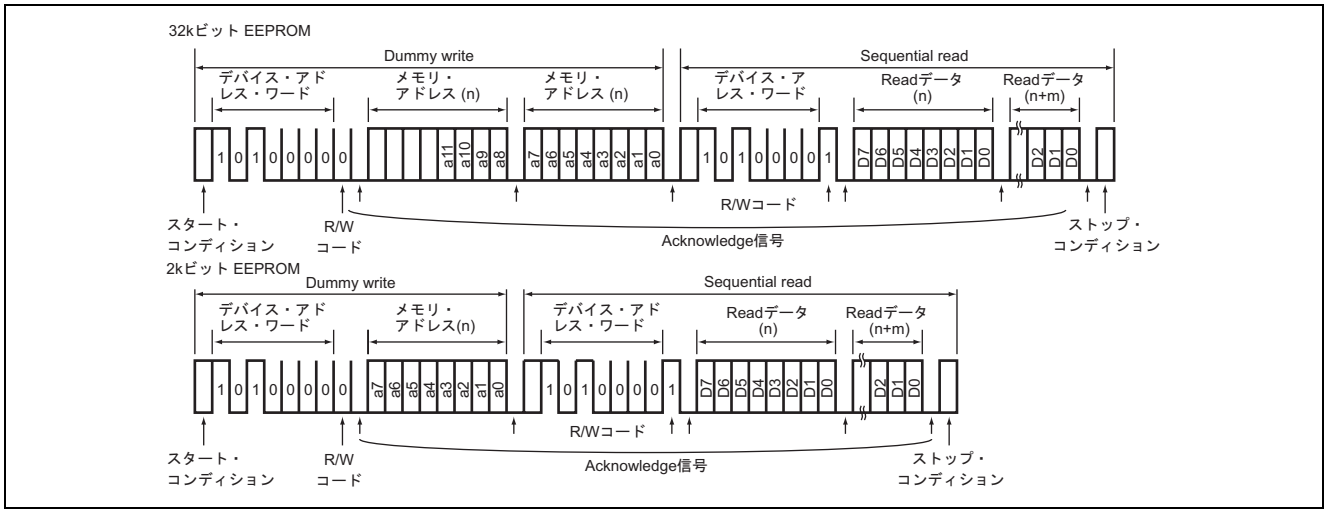


図 10　Sequential Read 動作

(i) IIC バス EEPROM は，Write 動作が終了し，Read 動作に入るときには，書き換え時間（twc）の期間，待機しなければなりません。書き換え時間（twc）は，5V 動作時，10ms（max）です。書き換え時間のタイミング波形を図 11 に示します。

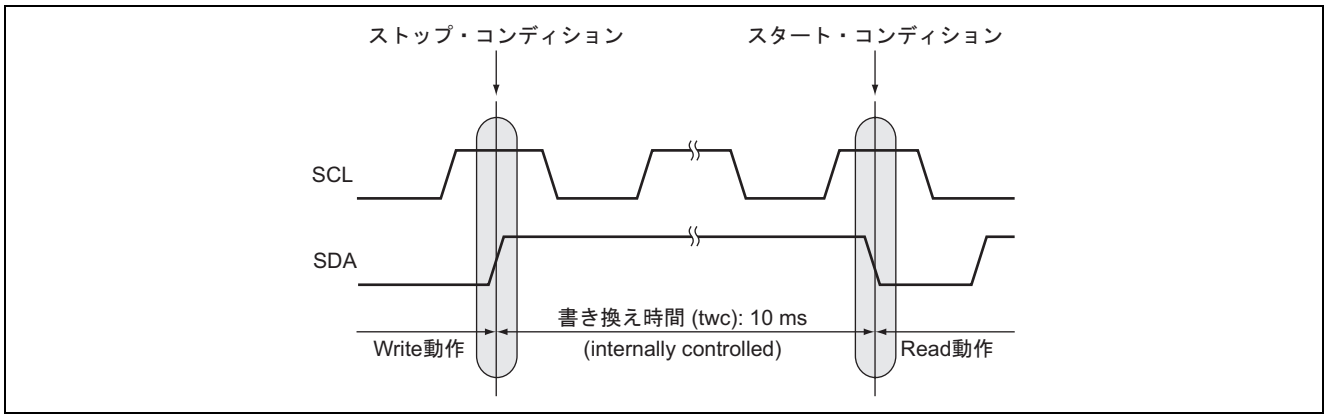


図 11　書き換え時間のタイミング波形

4. 動作説明

(1) Write（送信）時の動作原理を図 12 に示します。図 12 に示すように H8/38024 のハードウェア処理，およびソフトウェア処理により EEPROM への Write（送信）を行います。

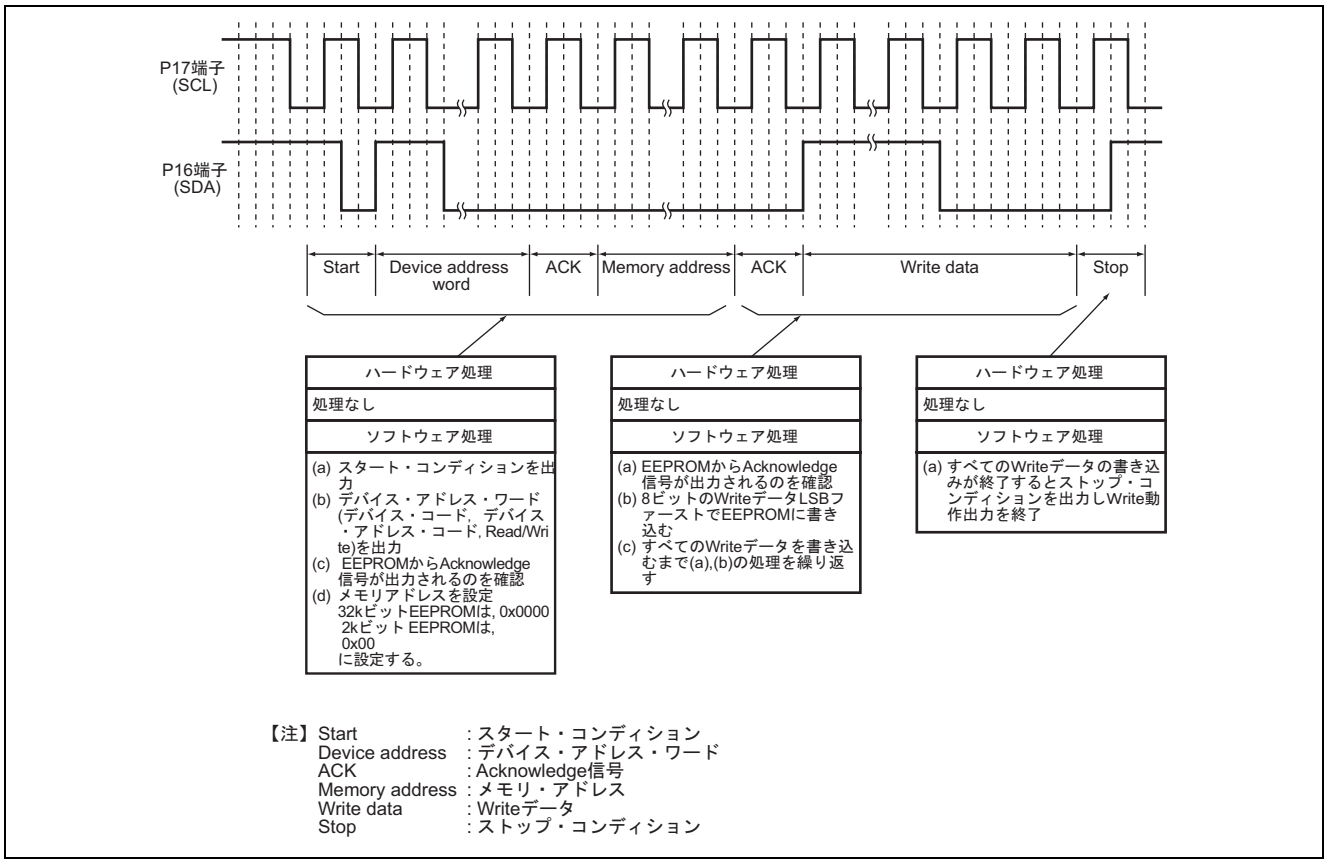


図 12　EEPROM への Write 時の動作原理

(2) Read（受信）時の動作原理を図 13 に示します。図 13 に示すように H8/38024 のハードウェア処理，およびソフトウェア処理により EEPROM からの Read（受信）を行います。

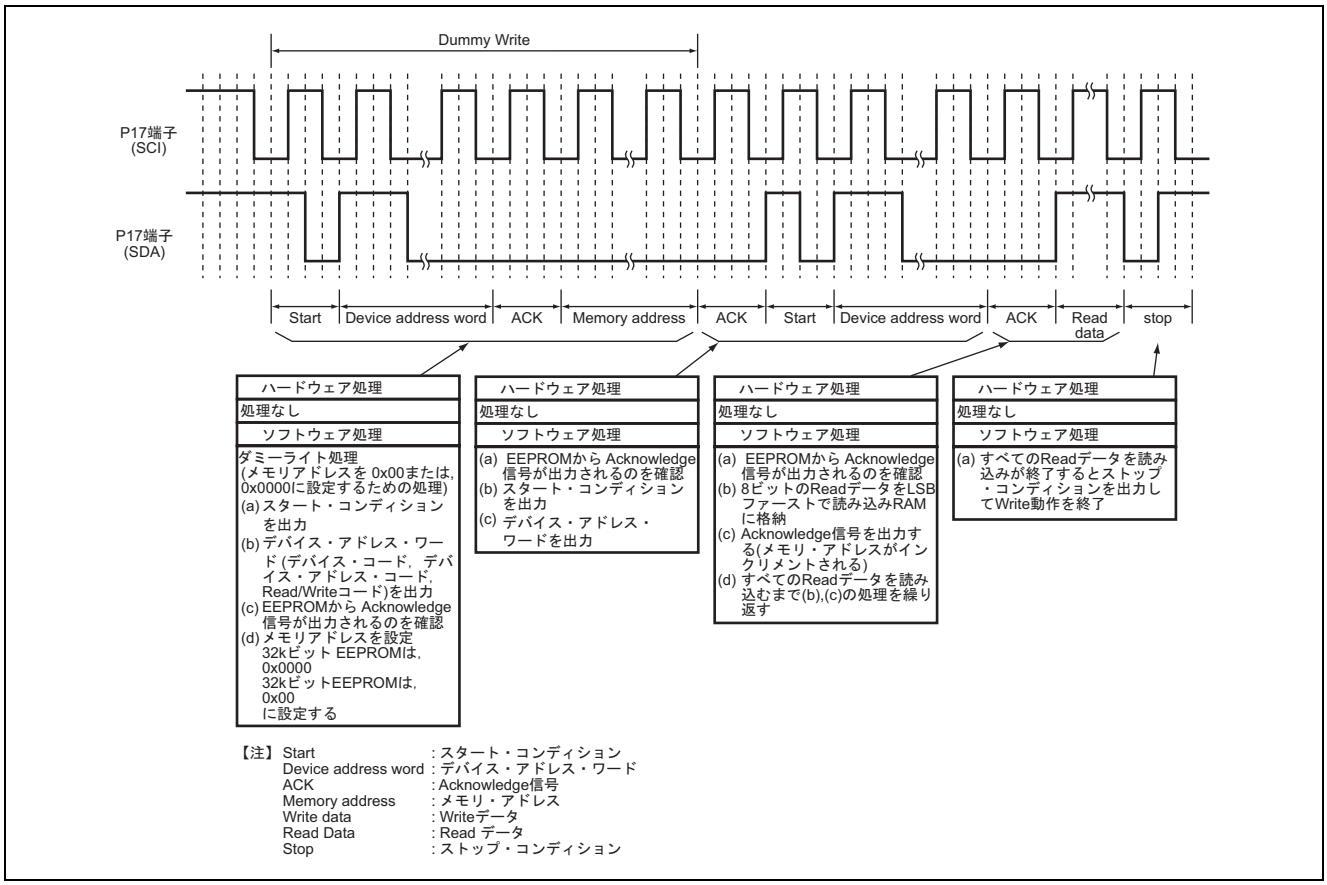


図 13　EEPROM からの Read 時の動作原理

(3) 本タスク例で使用しているポート 1 への入出力動作について表に示します。表に示すような設定により，シリアル・クロックの出力，およびシリアル・データの入出力を行います。

表 3　入出力動作

		IIC バス EEPROM に送信	IIC バス EEPROM から受信
PCR1	PCR17	1	1
	PCR16	1	0
PDR1	P17	**クロック送信**	**クロック送信**
	P16	**送信データ**	**受信データ**

(4) 本タスク例で使用している H8/38024 のメモリマップを図14 に示します。

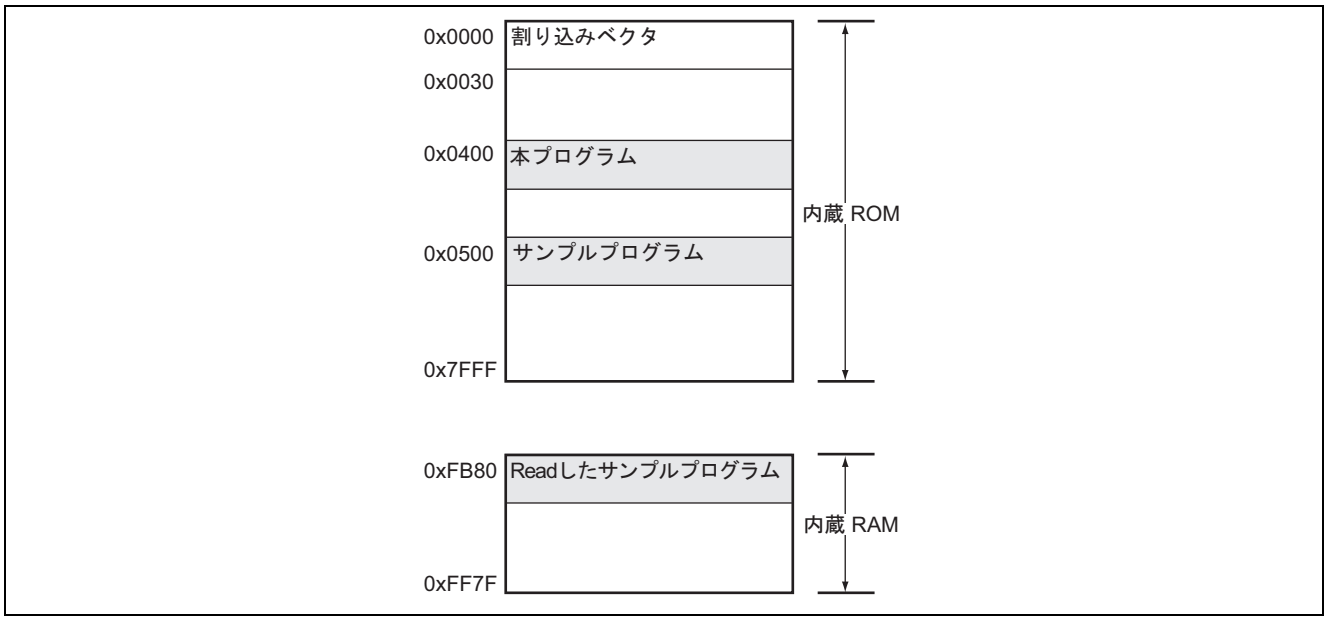


図 14　本タスク例で使用する H8/38024 のメモリマップ

5. ソフトウェア説明

(1) モジュール説明

本タスク例のモジュールを表 4 に示します。

表 4　モジュール説明

モジュール名	関数名	機能
メインルーチン	main	割り込みの禁止，書き換え時間の待機，EEPROM の書き込み，読み出しの制御を行う。
IIC バスデータ送信	send_code	指定されたデータを IIC バス方式の信号に変換し，EEPROM に送信する。
SCL,SDA データセット	send_p1a	EEPROM の SCL,SDA と接続している P17,P16 に，データをセットする。
スタートコンディションコード出力	send_start	スタートコンディションコードを出力する。
ストップコンディションコード出力	send_stop	ストップコンディションコードを出力する。
ビット"1"出力	send_bit1	IIC バス方式のビット"1"を出力する。
ビット"0"出力	send_bit0	IIC バス方式のビット"0"を出力する。
IIC バスデータ受信	rcv_code	EEPROM からデータを読み出し，RAM に格納する。
Acknowledge 信号受信	rcv_ack	Acknowledge 信号の受信。
受信サンプルプログラム	splpgm	EEPROM から受信したサンプルプログラム。

(2) 引数の説明

表 5　引数の説明

引数名	機能	関数名	データ長	入出力
*inpdt	送信するデータの先頭アドレス	send_code	1 バイト	入力
SIZE	送信するデータ数	send_code	1 バイト	入力
*outpdt	受信したデータを格納する領域の先頭アドレス	rcv_code	1 バイト	出力
SIZE	受信するデータ数	rcv_code	1 バイト	入力
dt	P17,P16 の設定値	send_p1a	1 バイト	入力

(3) 使用内部レジスタ説明

本タスク例の使用内部レジスタを表 6 に示します。

表 6　使用内部レジスタ説明

レジスタ名		機能	アドレス	設定値
PDR1	P17	ポートデータレジスタ 1（ポートデータレジスタ 17） ：P17=0 のとき，P17（SCL）端子の出力レベルは"Low" ：P17=1 のとき，P17（SCL）端子の出力レベルは"High"	0xFFD4 ビット 7	—
	P16	ポートデータレジスタ 1（ポートデータレジスタ 16） ：P16=0 のとき，P16（SDA）端子の出力レベルは"Low" ：P16=1 のとき，P16（SDA）端子の出力レベルは"High"	0xFFD4 ビット 6	—
PCR1	PCR17	ポートコントロールレジスタ 1（ポートコントロールレジスタ 17） ：PCR17=0 のとき，P17（SCL）を出力端子に設定 ：PCR17=1 のとき，P17（SCL）を入力端子に設定	0xFFE4 ビット 7	1
	PCR16	ポートコントロールレジスタ 1（ポートコントロールレジスタ 16） ：PCR16=0 のとき，P16（SDA）を出力端子に設定 ：PCR16=1 のとき，P16（SDA）を入力端子に設定	0xFFE4 ビット 6	—

(4) 使用 RAM 説明

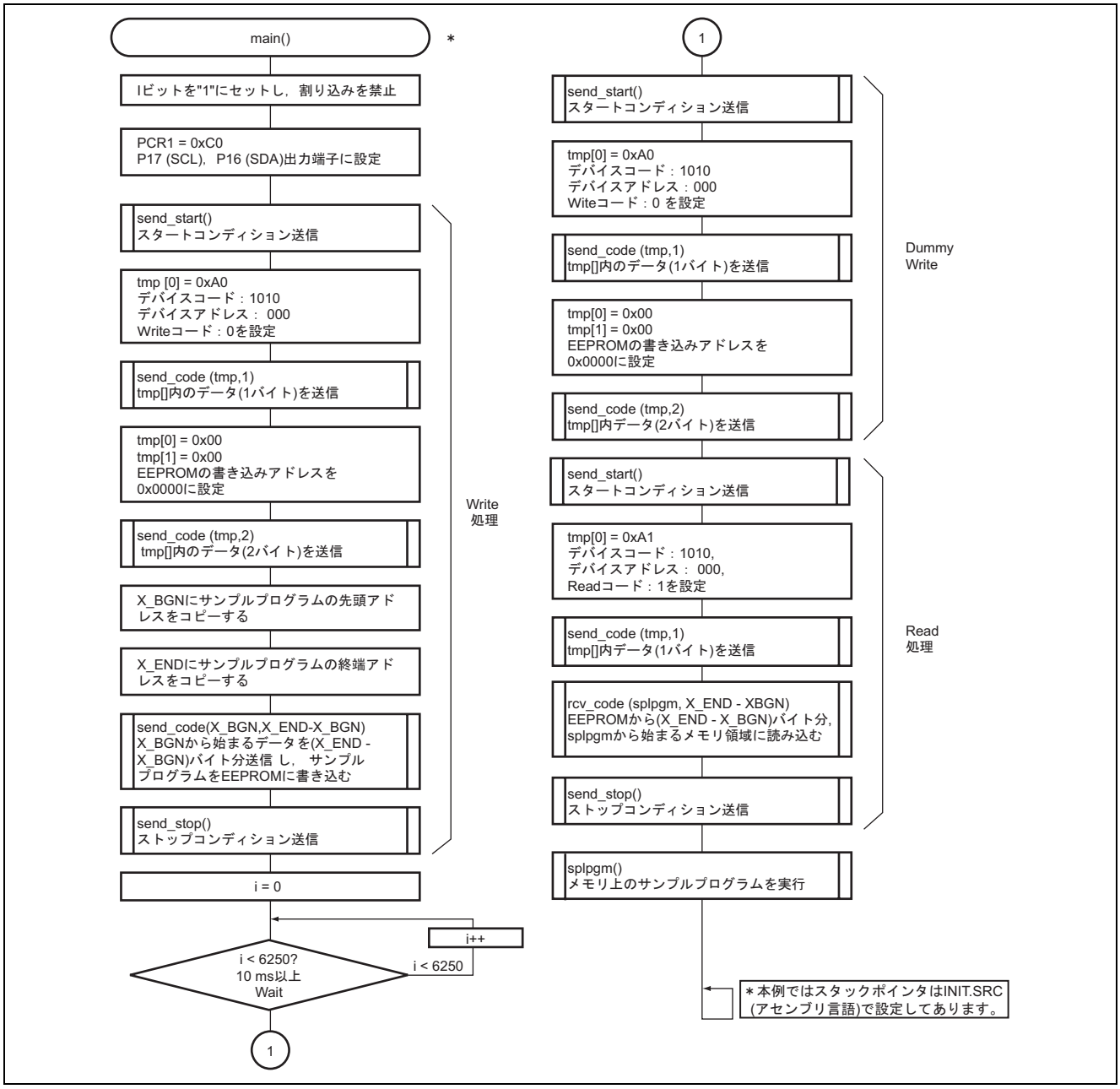
本タスク例の使用 RAM を表 7 に示します。

表 7　使用 RAM 説明

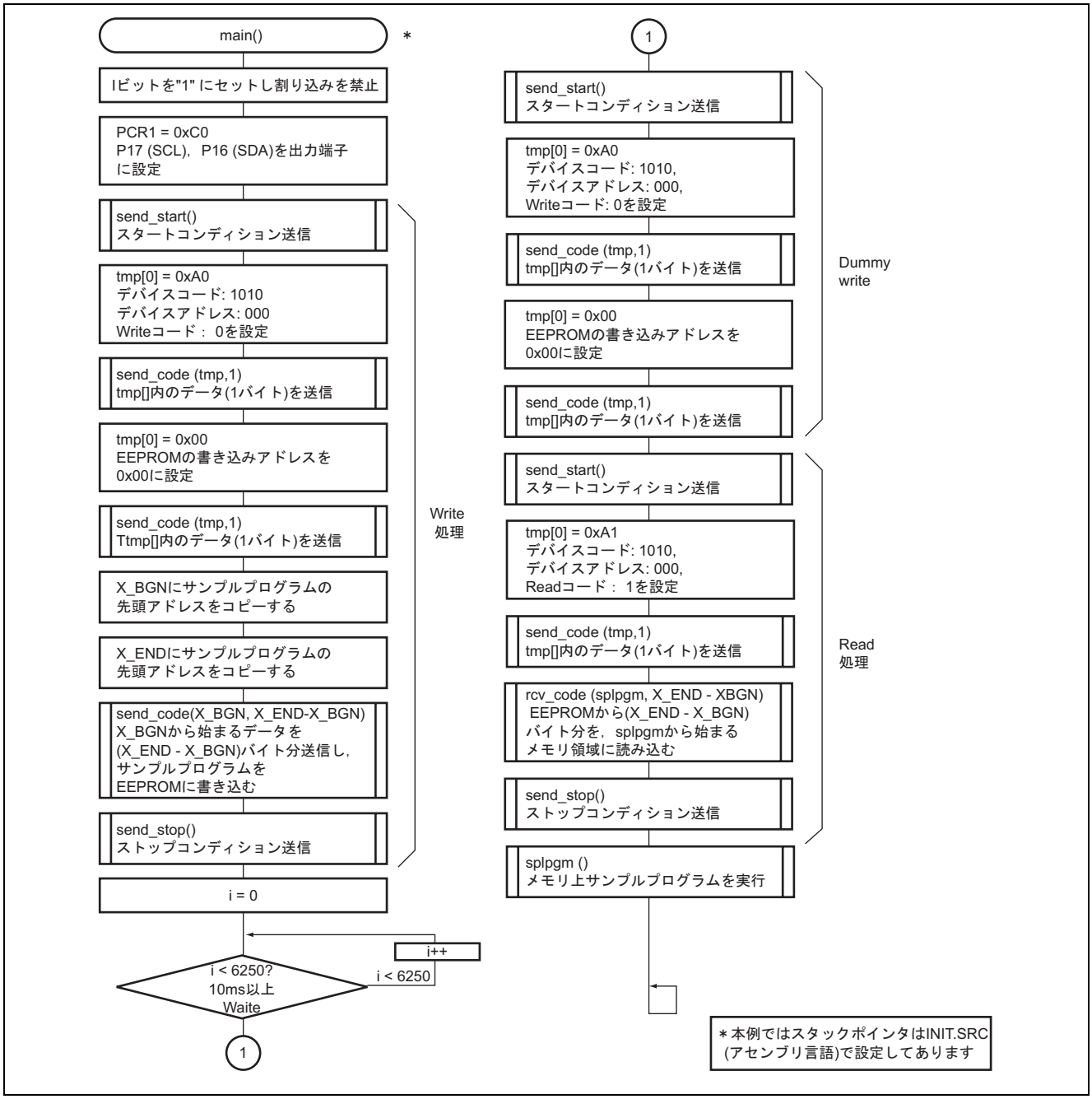
ラベル名	機能	メモリ消費量	使用モジュール名
splpgm	EEPROM から読み出したサンプルプログラムを格納する	6byte	main

6. フローチャート

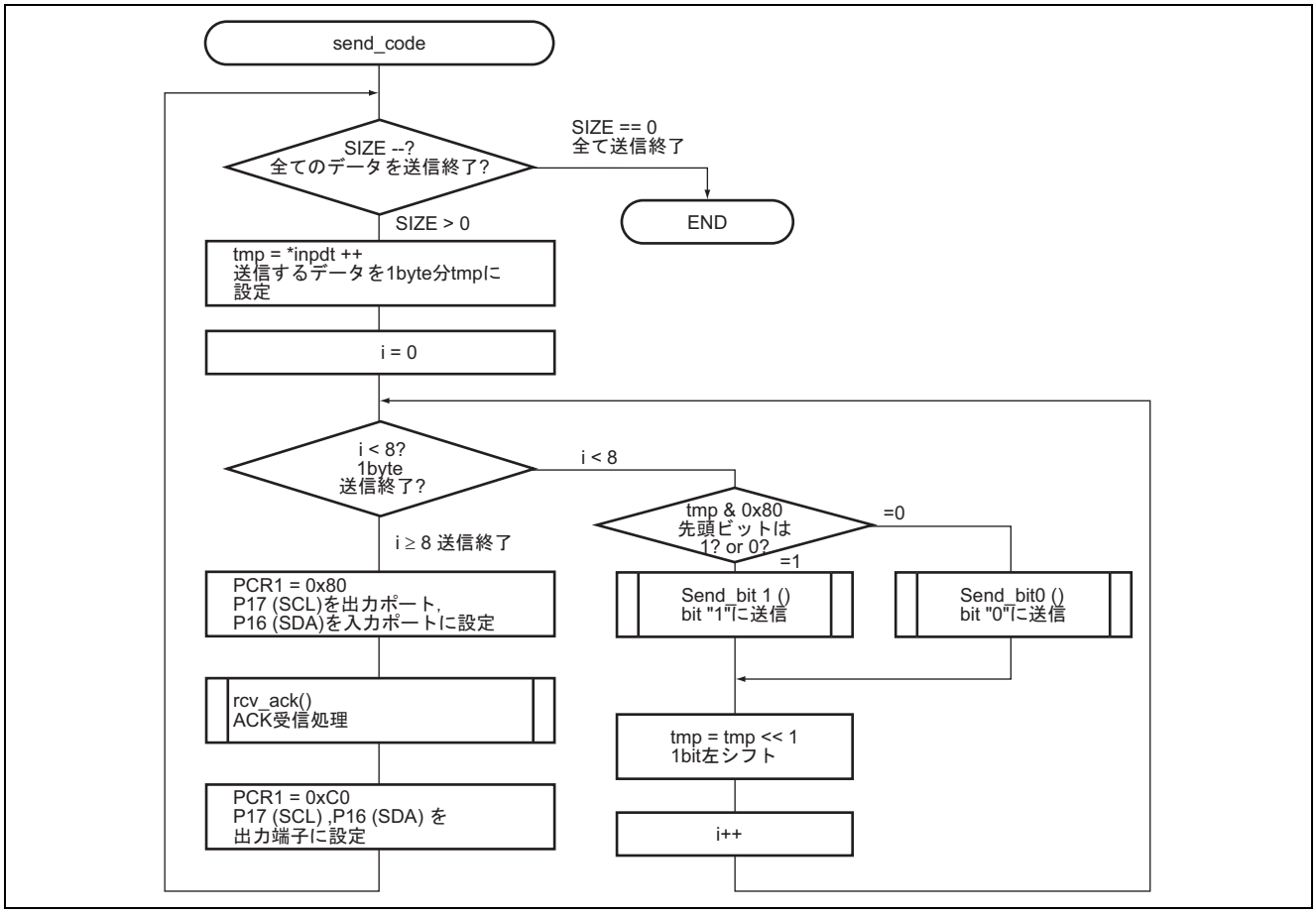
(1) 64k ビット EEPROM のメインルーチン



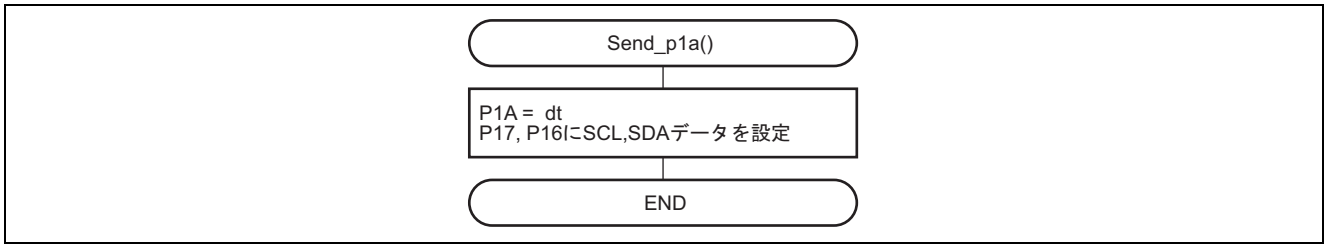
(2) 2k ビット EEPROM のメインルーチン



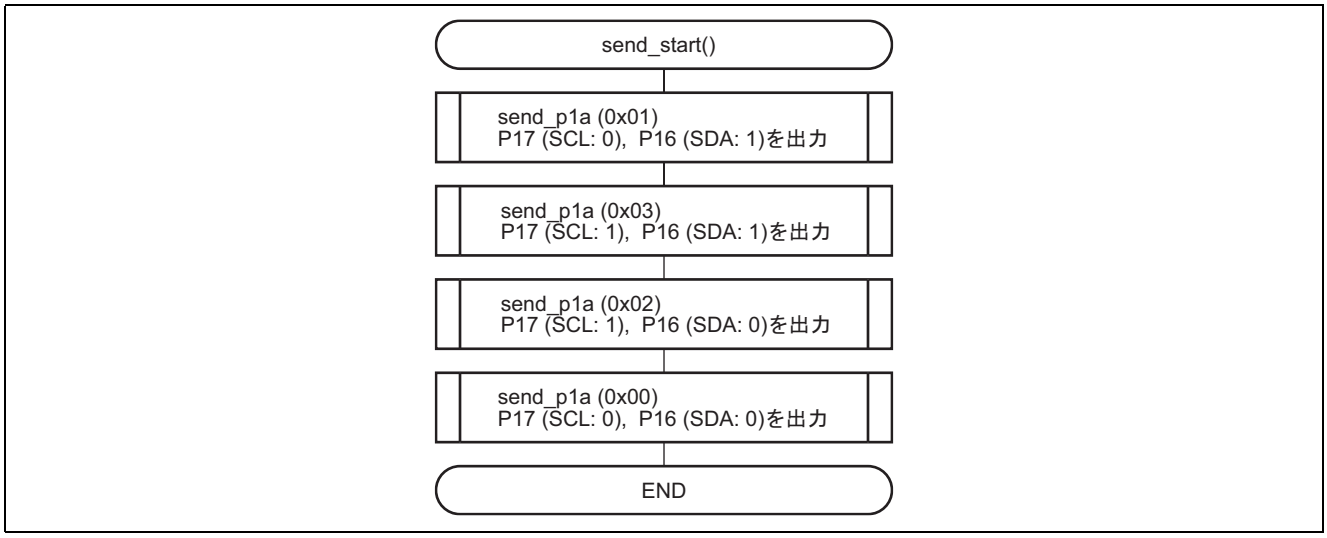
(3) IIC バスデータ送信



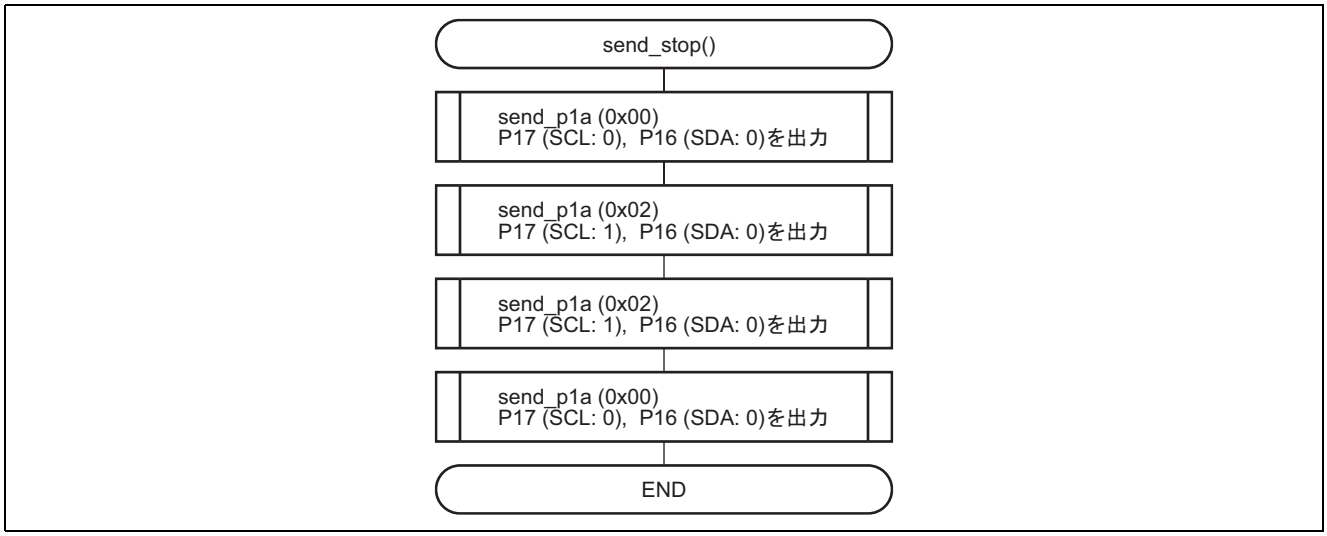
(4) SCL,SDA データセット



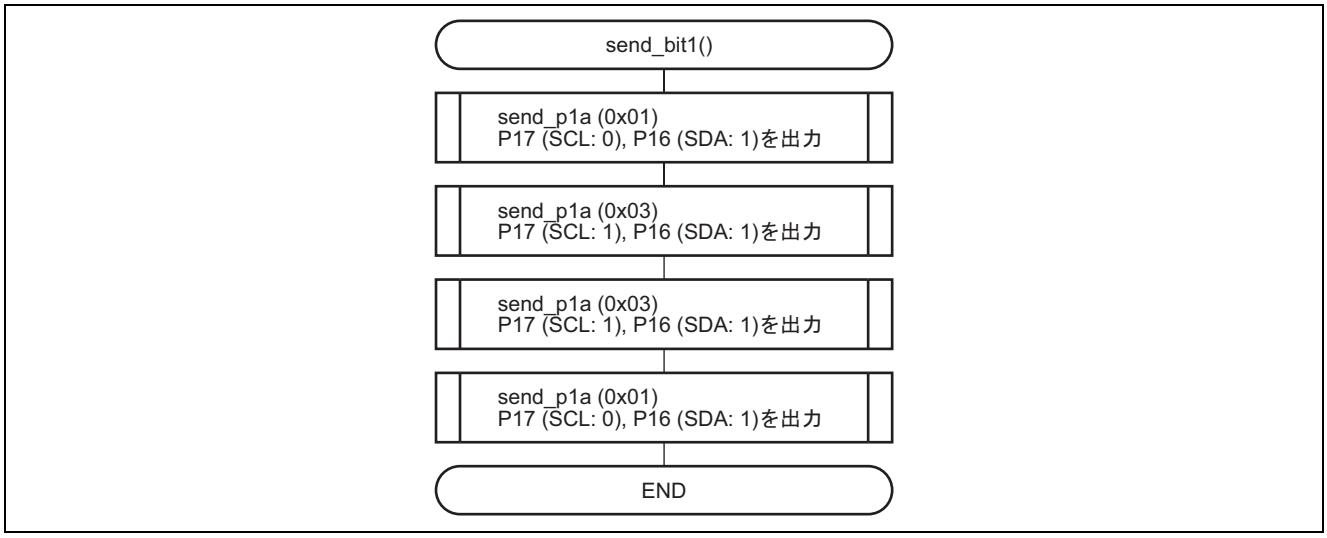
(5) スタートコンディションコード出力



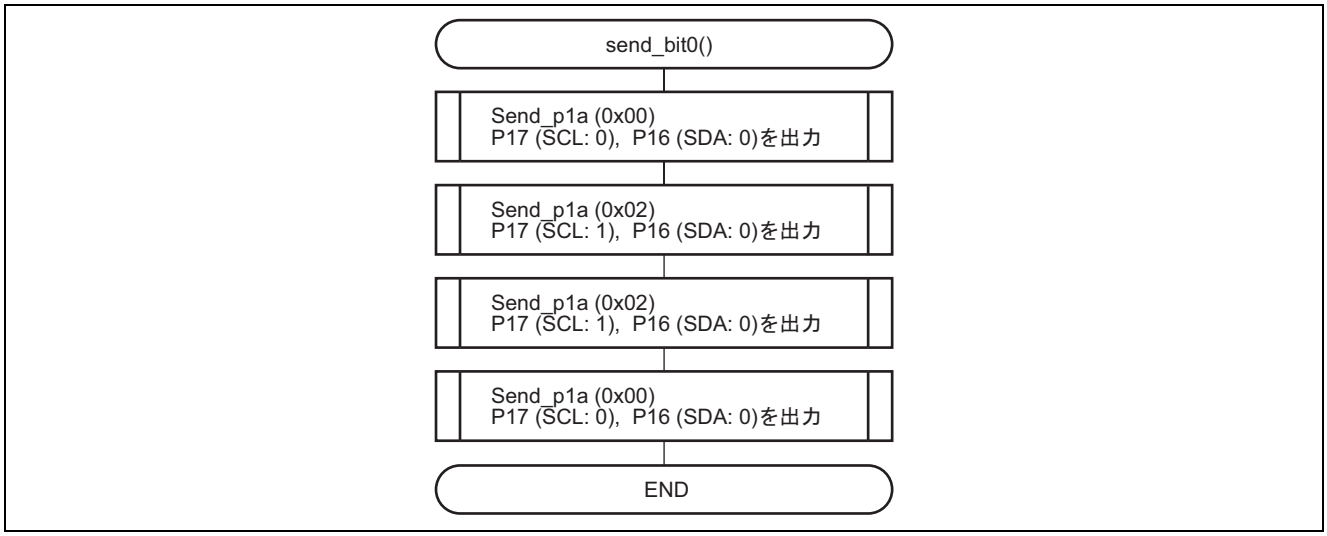
(6) ストップコンディションコード出力



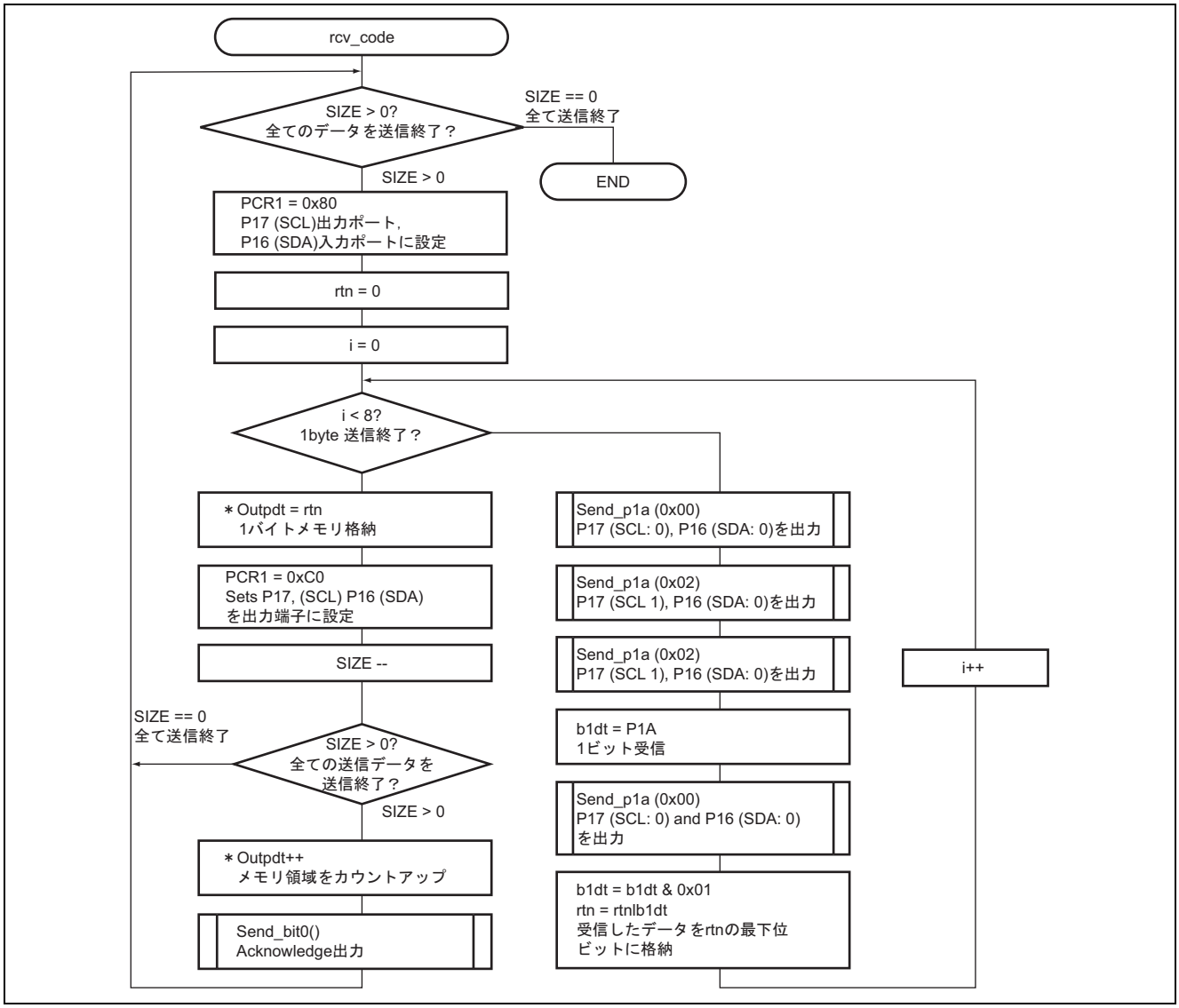
(7) ビット"1"出力



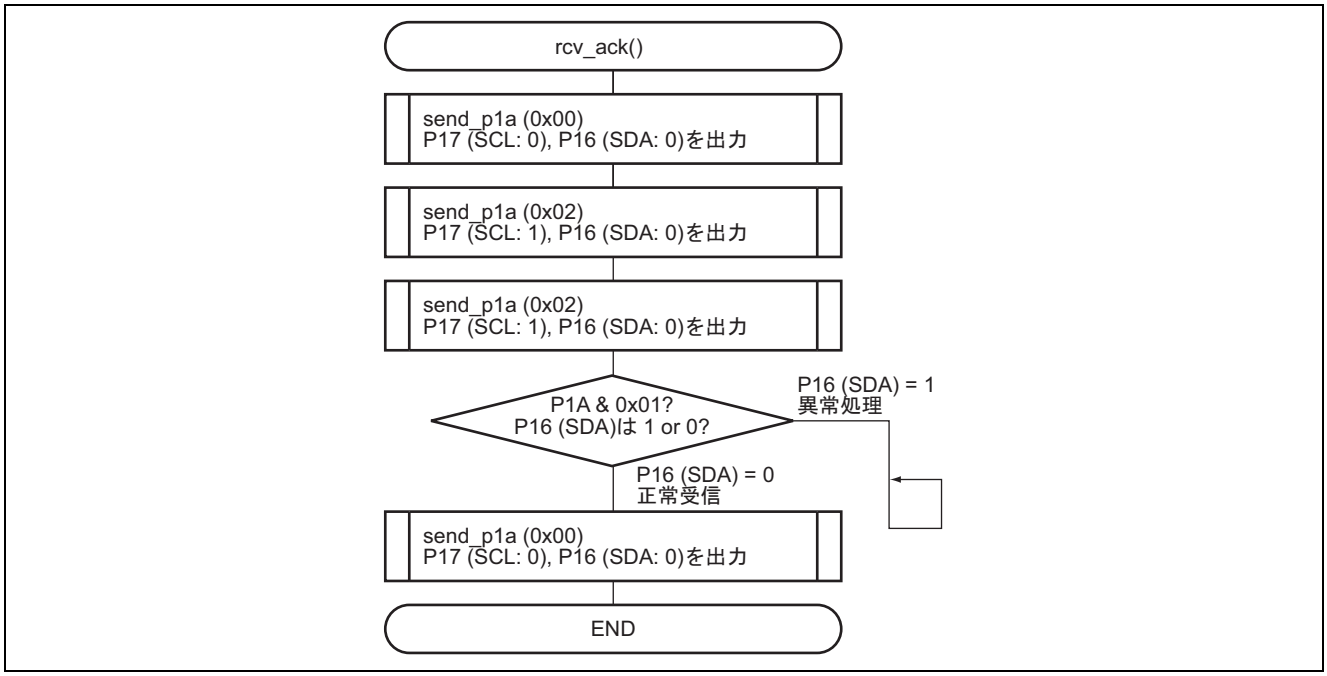
(8) ビット"0"出力



(9) IC バスデータ受信


rcv_code
SIZE > 0?
全てのデータを送信終了？
SIZE == 0
全て送信終了
END
SIZE > 0
PCR1 = 0x80
P17 (SCL)出力ポート,
P16 (SDA)入力ポートに設定
rtn = 0
i = 0
i < 8?
1byte 送信終了？
＊Outpdt = rtn
1バイトメモリ格納
PCR1 = 0xC0
Sets P17, (SCL) P16 (SDA)
を出力端子に設定
SIZE --
SIZE == 0
全て送信終了
SIZE > 0?
全ての送信データを
送信終了？
SIZE > 0
＊Outpdt++
メモリ領域をカウントアップ
Send_bit0()
Acknowledge出力
Send_p1a (0x00)
P17 (SCL: 0), P16 (SDA: 0)を出力
Send_p1a (0x02)
P17 (SCL 1), P16 (SDA: 0)を出力
Send_p1a (0x02)
P17 (SCL 1), P16 (SDA: 0)を出力
b1dt = P1A
1ビット受信
Send_p1a (0x00)
P17 (SCL: 0) and P16 (SDA: 0)
を出力
b1dt = b1dt & 0x01
rtn = rtnlb1dt
受信したデータをrtnの最下位
ビットに格納
i++

(10) Acknowledge 信号受信



7. プログラムリスト

7.1 INIT.SRC（プログラムリスト）

```
      .export _INIT
      .import _main
  ;
      .section P,CODE
  _INIT:
      mov.w   #h'ff80,r7
      ldc.b   #b'10000000,ccr
      jmp @_main
  ;
      .end
```

(1) 64k ビット EEPROM のプログラムリスト

```
  /************************************************************/
  /*                                                          */
  /*  H8/300L Super Low Power Series                            */
  /*     -H8/38024 Series-                                    */
  /*  Application Note                                        */
  /*                                                          */
  /*  '64kbit EEPROM Write & Read Control'                      */
  /*                                                          */
  /*  Function                                                */
  /*  : I/O Port Base                                         */
  /*                                                          */
  /*  External Clock : 10MHz                                  */
  /*  Internal Clock : 5MHz                                   */
  /*  Sub Clock      : 32.768kHz                              */
  /*                                                          */
  /************************************************************/

  #include    <machine.h>

  /************************************************************/
  /*  Symbol Definition                                       */
  /************************************************************/
  struct BIT {
     unsigned char   b7:1;      /* bit7  */
     unsigned char   b6:1;      /* bit6  */
     unsigned char   b5:1;      /* bit5  */
     unsigned char   b4:1;      /* bit4  */
     unsigned char   b3:1;      /* bit3  */
     unsigned char   b2:1;      /* bit2  */
     unsigned char   b1:1;      /* bit1  */
     unsigned char   b0:1;      /* bit0  */
  };

  struct BIT2 {
     unsigned char   ba:2;      /* bit7,6 */
     unsigned char    :5;      /* other  */
  };

  #define    PDR1       *(volatile unsigned char *)0xFFD4   /* Port Data Register 1          */
  #define    PDR1_BIT2   (*(struct BIT2 *)0xFFD4)          /* Port Data Register 1          */
  #define    P1A        PDR1_BIT2.ba                    /* P17,P16                      */
```

```
#define    PCR1      *(volatile unsigned char *)0xFFE4   /* Port Control Register 1        */

/************************************************************/
/*  Function define                                         */
/************************************************************/
extern void INIT( void );                                  /* SP Set                       */
extern void splpgm( void );

void main( void );
void send_code( unsigned char *inpdt, unsigned char SIZE );
void send_p1a( unsigned char dt );
void send_start( void );
void send_stop( void );
void send_bit1( void );
void send_bit0( void );
void rcv_code( unsigned char *outpdt, unsigned char SIZE );
void rcv_ack( void );

/************************************************************/
/*  Vector Address                                          */
/************************************************************/
#pragma section    V1                                      /* VECTOR SECTOIN SET            */
void (*const VEC_TBL1[])(void) = {                           /* 0x00 - 0x0f */
   INIT                                                   /* 00 Reset                     */
};

#pragma section                                            /* P                             */
/************************************************************/
/*  Main Program                                            */
/************************************************************/
void main( void )
{
   unsigned char   tmp[2];
   unsigned char   *X_BGN;
   unsigned char   *X_END;
   unsigned short  i;

   set_imask_ccr(1);                                       /* Interrupt Disable             */

   PCR1 = 0xC0;                                           /* P17,P16 Set Output Port       */
   send_start();                                          /* Send Start Condition          */
   tmp[0] = 0xA0;                                         /* Set Write Code                */
   send_code(tmp,1);                                       /* Send 1byte                    */
   tmp[0] = 0x00;                                         /* Set EEPROM Write Address       */
   tmp[1] = 0x00;
   send_code(tmp,2);                                       /* Send 2byte                    */

   X_BGN = __sectop("SPLPG");                               /* Sample Program Top address     */
   X_END = __secend("SPLEND");                              /* Sample Program End address     */
   send_code(X_BGN,X_END-X_BGN);                            /* Send and Write Sample Program   */
   send_stop();                                           /* Send Stop Condition           */

   for(i = 0; i < 6250; i++);                               /* Need to wait 10 msec          */

   send_start();                                          /* Dummy Write                   */
   tmp[0] = 0xA0;
   send_code(tmp,1);
```

```
    tmp[0] = 0x00;
    tmp[1] = 0x00;
    send_code(tmp,2);

    send_start();                                   /* Send Start Condition          */
    tmp[0] = 0xA1;                                  /* Set Read Code                 */
    send_code(tmp,1);                               /* Send 1byte                    */
    rcv_code((unsigned char*)splpgm,X_END-X_BGN);       /* Read EEPROM --> RAM(splpgm)     */
    send_stop();                                    /* Send Stop Condition           */

    splpgm();                                       /* Go to Sample Program          */

    while(1);
}

/*********************************************************/
/*  Send IIC Communication Data                          */
/*********************************************************/
void send_code( unsigned char *inpdt, unsigned char SIZE )
{
    unsigned char i,tmp;

    while(SIZE--){
        tmp = *inpdt++;                             /* Set 1 byte                    */
        for(i = 0; i < 8; i++){                       /* 1 byte Send Finish?           */
            if(tmp & 0x80)
                send_bit1();                        /* Send bit"1"                   */
            else
                send_bit0();                        /* Send bit"0"                   */

            tmp = tmp<<1;                           /* Next bit                      */
        }
        PCR1 = 0x80;                                /* P17:Output Port   P16:Input Port */
        rcv_ack();                                  /* Receive Acknowledge           */
        PCR1 = 0xC0;                                /* P17,P16 Set Output Port       */
    }
}

/*********************************************************/
/*  Set SCL,SDA                                          */
/*********************************************************/
void send_p1a( unsigned char dt )
{
    P1A = dt;
}

/*********************************************************/
/*  Start Condition                                      */
/*********************************************************/
void send_start( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x02);
    send_p1a(0x00);
}
```

```
/*************************************************************/
/*  Stop Condition                                           */
/*************************************************************/
void send_stop( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x03);
    send_p1a(0x01);
}

/*************************************************************/
/*  Send bit "1"                                             */
/*************************************************************/
void send_bit1( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x03);
    send_p1a(0x01);
}

/*************************************************************/
/*  Send bit "0"                                             */
/*************************************************************/
void send_bit0( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    send_p1a(0x00);
}

/*************************************************************/
/*  Receive IIC Communication Data                           */
/*************************************************************/
void rcv_code( unsigned char *outpdt, unsigned char SIZE )
{
    unsigned char i,b1dt,rtn;

    while(SIZE > 0){
        PCR1 = 0x80;                                /* P17:Output Port   P16:Input Port */
        rtn = 0;
        for(i = 0; i < 8; i++){                       /* 1 byte Receive Finish?         */
            rtn = rtn<<1;

            send_p1a(0x00);
            send_p1a(0x02);
            send_p1a(0x02);
            b1dt = P1A;                             /* Receive 1 bit                */
            send_p1a(0x00);
            b1dt = b1dt & 0x01;

            rtn = rtn|b1dt;
        }
        *outpdt = rtn;                              /* Set Receive Data             */
```

```
        PCR1 = 0xC0;                                    /* P17,P16 Set Output Port        */
        SIZE--;
        if(SIZE > 0){
            *outpdt++;                                  /* Ram Address Countup            */
            send_bit0();                                /* Output Acknowledge             */
        }
    }
}

/*********************************************************/
/*  Receive Acknowledge                                  */
/*********************************************************/
void rcv_ack( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    if(P1A & 0x01)                                      /* Receive Cheak Acknowledge      */
        while(1);
    send_p1a(0x00);
}
```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
P	0x0100
SPLPG,PSPLPG,SPLEND	0x0500
PRAM	0xFB80

(2) 2k ビット EEPROM のプログラムリスト

```
/************************************************************/
/*                                                          */
/*  H8/300L Super Low Power Series                          */
/*     -H8/38024 Series-                                    */
/*  Application Note                                        */
/*                                                          */
/*  '2kbit EEPROM Write & Read Control'                     */
/*                                                          */
/*  Function                                                */
/*  : I/O Port Base                                         */
/*                                                          */
/*  External Clock : 10MHz                                  */
/*  Internal Clock : 5MHz                                   */
/*  Sub Clock      : 32.768kHz                              */
/*                                                          */
/************************************************************/

#include   <machine.h>

/************************************************************/
/*  Symbol Definition                                       */
/************************************************************/
struct BIT {
   unsigned char   b7:1;      /* bit7  */
   unsigned char   b6:1;      /* bit6  */
   unsigned char   b5:1;      /* bit5  */
   unsigned char   b4:1;      /* bit4  */
   unsigned char   b3:1;      /* bit3  */
   unsigned char   b2:1;      /* bit2  */
   unsigned char   b1:1;      /* bit1  */
   unsigned char   b0:1;      /* bit0  */
};

struct BIT2 {
   unsigned char   ba:2;      /* bit7,6 */
   unsigned char    :5;      /* other  */
};

#define    PDR1       *(volatile unsigned char *)0xFFD4   /* Port Data Register 1           */
#define    PDR1_BIT2   (*(struct BIT2 *)0xFFD4)          /* Port Data Register 1           */
#define    P1A        PDR1_BIT2.ba                    /* P17,P16                      */
#define    PCR1       *(volatile unsigned char *)0xFFE4   /* Port Control Register 1        */

/************************************************************/
/*  Function define                                         */
/************************************************************/
extern void INIT( void );                                  /* SP Set                       */
extern void splpgm( void );

void main( void );
void send_code( unsigned char *inpdt, unsigned char SIZE );
void send_p1a( unsigned char dt );
void send_start( void );
void send_stop( void );
void send_bit1( void );
void send_bit0( void );
```

```
void rcv_code( unsigned char *outpdt, unsigned char SIZE );
void rcv_ack( void );

/*********************************************************/
/*  Vector Address                                       */
/*********************************************************/
#pragma section    V1                                     /* VECTOR SECTOIN SET            */
void (*const VEC_TBL1[])(void) = {                          /* 0x00 - 0x0f */
   INIT                                                 /* 00 Reset                      */
};

#pragma section                                           /* P                             */
/*********************************************************/
/*  Main Program                                         */
/*********************************************************/
void main( void )
{
   unsigned char   tmp[2];
   unsigned char   *X_BGN;
   unsigned char   *X_END;
   unsigned short  i;

   set_imask_ccr(1);                                      /* Interrupt Disable             */

   PCR1 = 0xC0;                                         /* P17,P16 Set Output Port        */
   send_start();                                        /* Send Start Condition           */
   tmp[0] = 0xA0;                                       /* Set Write Code                 */
   send_code(tmp,1);                                     /* Send 1byte                    */
   tmp[0] = 0x00;                                       /* Set EEPROM Write Address        */
   send_code(tmp,1);                                     /* Send 1byte                    */

   X_BGN = __sectop("SPLPG");                             /* Sample Program Top address      */
   X_END = __secend("SPLEND");                            /* Sample Program End address      */
   send_code(X_BGN,X_END-X_BGN);                          /* Send and Write Sample Program    */
   send_stop();                                         /* Send Stop Condition            */

   for(i = 0; i < 6250; i++);                             /* Need to wait 10 msec           */

   send_start();                                        /* Dummy Write                   */
   tmp[0] = 0xA0;
   send_code(tmp,1);
   tmp[0] = 0x00;
   send_code(tmp,1);

   send_start();                                        /* Send Start Condition           */
   tmp[0] = 0xA1;                                       /* Set Read Code                 */
   send_code(tmp,1);                                     /* Send 1byte                    */
   rcv_code((unsigned char*)splpgm,X_END-X_BGN);          /* Read EEPROM --> RAM(splpgm)     */
   send_stop();                                         /* Send Stop Condition            */

   splpgm();                                            /* Go to Sample Program           */

   while(1);
}

/*********************************************************/
/*  Send IIC Communication Data                            */
```

```
/*********************************************************/
void send_code( unsigned char *inpdt, unsigned char SIZE )
{
    unsigned char i,tmp;

    while(SIZE--){
        tmp = *inpdt++;                                 /* Set 1 byte                    */
        for(i = 0; i < 8; i++){                          /* 1 byte Send Finish?             */
            if(tmp & 0x80)
                send_bit1();                            /* Send bit"1"                   */
            else
                send_bit0();                            /* Send bit"0"                   */

            tmp = tmp<<1;                               /* Next bit                      */
        }
        PCR1 = 0x80;                                    /* P17:Output Port   P16:Input Port */
        rcv_ack();                                      /* Receive Acknowledge            */
        PCR1 = 0xC0;                                    /* P17,P16 Set Output Port         */
    }
}

/*********************************************************/
/*  Set SCL,SDA                                          */
/*********************************************************/
void send_p1a( unsigned char dt )
{
    P1A = dt;
}

/*********************************************************/
/*  Start Condition                                      */
/*********************************************************/
void send_start( void )
{
    send_p1a(0x01);
    send_p1a(0x03);
    send_p1a(0x02);
    send_p1a(0x00);
}

/*********************************************************/
/*  Stop Condition                                       */
/*********************************************************/
void send_stop( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x03);
    send_p1a(0x01);
}

/*********************************************************/
/*  Send bit "1"                                         */
/*********************************************************/
void send_bit1( void )
{
    send_p1a(0x01);
```

```
    send_p1a(0x03);
    send_p1a(0x03);
    send_p1a(0x01);
}

/*********************************************************/
/*  Send bit "0"                                         */
/*********************************************************/
void send_bit0( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
    send_p1a(0x00);
}

/*********************************************************/
/*  Receive IIC Communication Data                       */
/*********************************************************/
void rcv_code( unsigned char *outpdt, unsigned char SIZE )
{
    unsigned char i,b1dt,rtn;

    while(SIZE > 0){
        PCR1 = 0x80;                                /* P17:Output Port   P16:Input Port */
        rtn = 0;
        for(i = 0; i < 8; i++){                       /* 1 byte Receive Finish?         */
            rtn = rtn<<1;

            send_p1a(0x00);
            send_p1a(0x02);
            send_p1a(0x02);
            b1dt = P1A;                             /* Receive 1 bit                */
            send_p1a(0x00);
            b1dt = b1dt & 0x01;

            rtn = rtn|b1dt;
        }
        *outpdt = rtn;                              /* Set Receive Data              */

        PCR1 = 0xC0;                                /* P17,P16 Set Output Port        */
        SIZE--;
        if(SIZE > 0){
            *outpdt++;                              /* Ram Address Countup           */
            send_bit0();                            /* Output Acknowledge            */
        }
    }
}

/*********************************************************/
/*  Receive Acknowledge                                  */
/*********************************************************/
void rcv_ack( void )
{
    send_p1a(0x00);
    send_p1a(0x02);
    send_p1a(0x02);
```

```
    if(P1A & 0x01)                          /* Receive Cheak Acknowledge       */
       while(1);
    send_p1a(0x00);
}
```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
P	0x0100
SPLPG,PSPLPG,SPLEND	0x0500
PRAM	0xFB80

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.12.19	—	初版発行

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。