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。

SH7145 グループ

DTC を併用した I²C バスインタフェース

要旨

I²C バス(Inter IC Bus)インタフェースによるデータの送受信を DTC(Data Transfer Controller)モジュールにより自動的に行う動作について述べます。マスタデバイスを SH7145F とし、スレーブデバイスに EEPROM を接続します。

動作確認デバイス

SH7145F

目次

1. DTC を併用した I²C バスシングルマスタ送信

1.1 仕様

SH7145F の DTC(Data Transfer Controller)を併用して I²C バス(Inter IC Bus)によるデータの送信を行います。

図 1 に示すように、マスタデバイスを SH7145F とし、スレーブデバイスに EEPROM(HN58X2464、64k bit ,8word×8bit)を接続しています。SH7145F の I²C バスインタフェースを用いて、内蔵 RAM に格納している 10 バイトのデータを EEPROM へ書き込みます。DTC を用いて内蔵 RAM から IIC モジュールのデータレジスタへのライトデータを転送します。

図 2 に DTC によるデータ転送を示します。また、表 1 に DTC の、表 2 に I²C バスインタフェースの設定を示します。

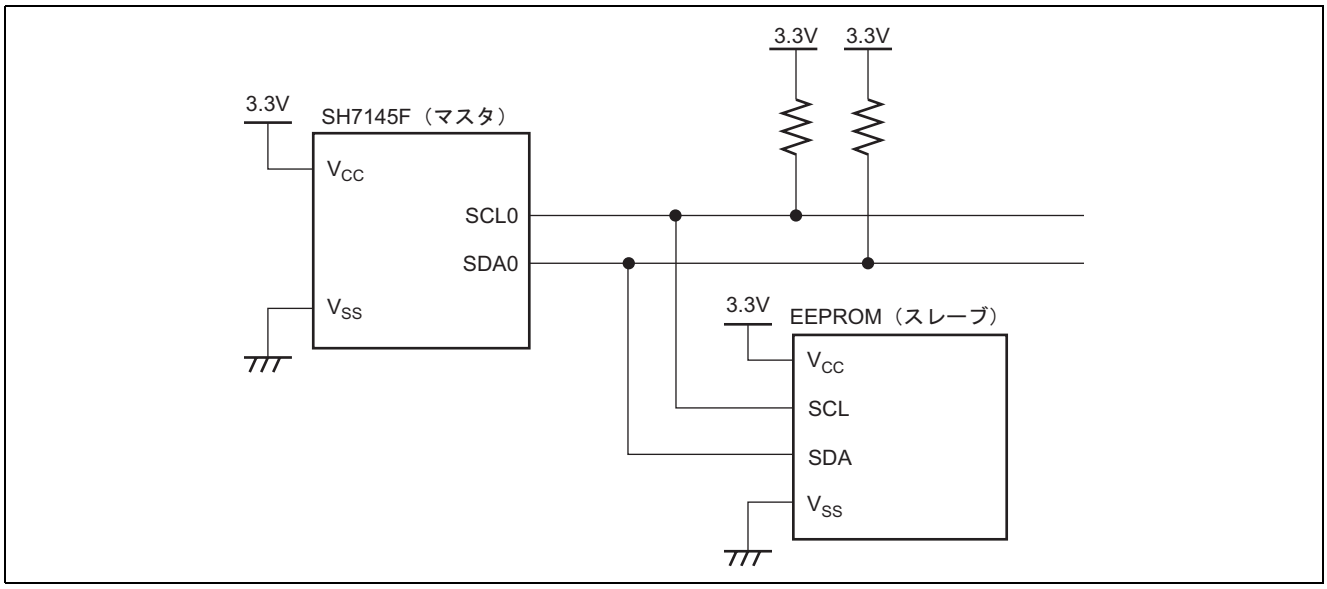


図 1　SH7145F と EEPROM の接続図

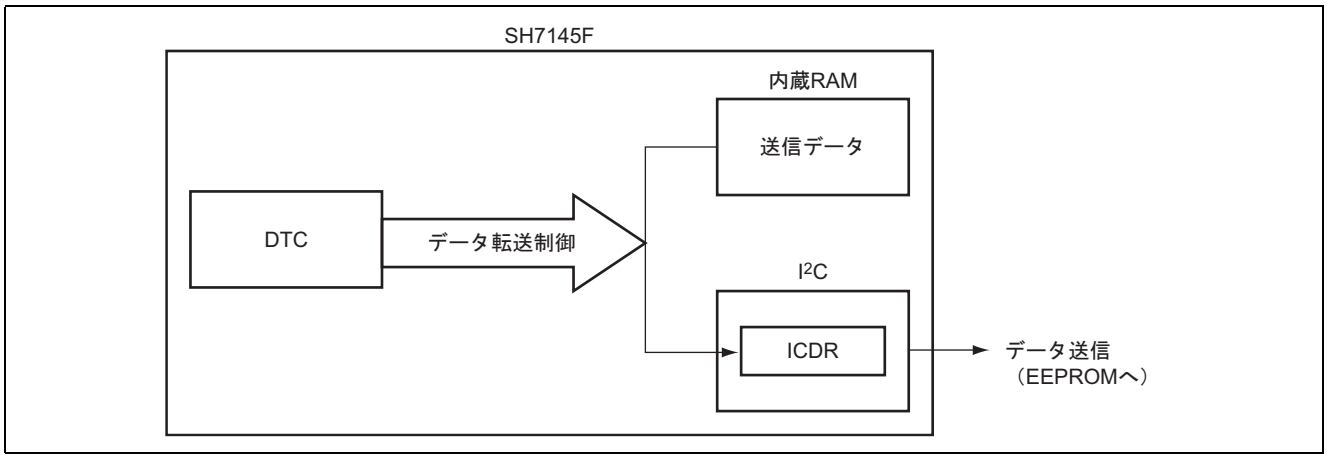


図 2　DTC によるデータ転送

表 1　DTC の設定

条件	内容
転送モード	ノーマルモード
転送データサイズ	1 バイト
転送回数（DTCRA）	13
転送元	内蔵 RAM
転送先	ICDR（I²C バスデータレジスタ；H'FFFF880E）
転送元アドレス	転送後にアドレスをインクリメント
転送先アドレス	アドレス固定
転送情報格納アドレス	H'FFFFF000
起動要因	I²C による割り込み（ICI）で起動
割り込み	指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可

表 2　I²C の設定

フォーマット内容	設定
動作	マスタ送信
転送クロック	156kHz（Pφ=40MHz）
データのビット数	9 ビット（ACK 含む）
データと ACK 間のウェイト	無し
割り込み	許可
ACK の判定	ACK=1 を受信したとき転送を中断する

1.2 使用機能説明

本タスク例では、I²C バス（Inter IC Bus）を用いて EEPROM へのライトを行います。ライトデータは、DTC（Data Transfer Controller）を用いて内蔵 RAM から I²C バスインタフェースのデータレジスタへと転送します。

1.2.1 I²C バス（Inter IC Bus）インタフェース

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。図 3 に I²C モジュールのブロック図を示し、以下に説明します。

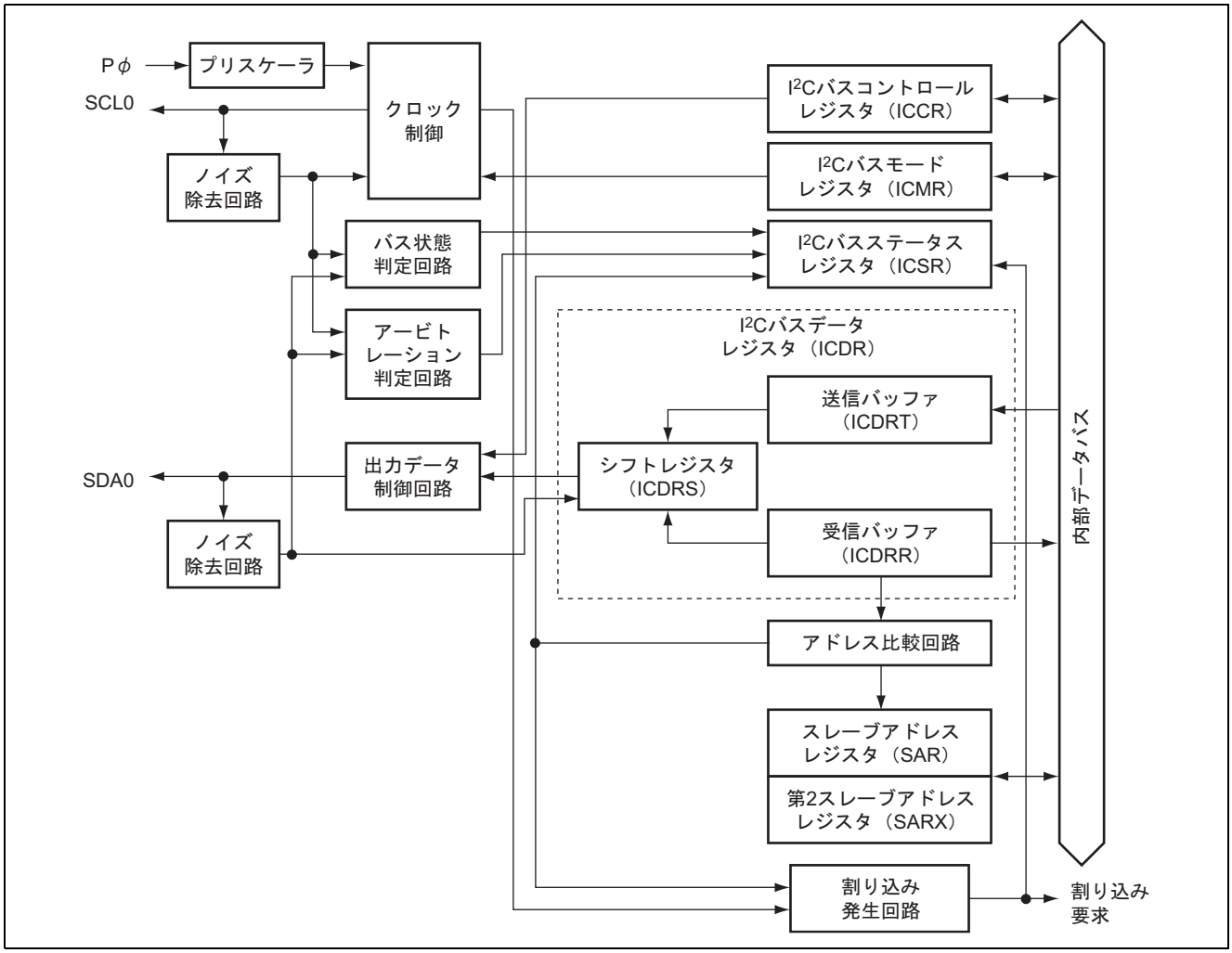


図 3　I²C モジュールのブロック図

- I²C バスコントロールレジスタ（ICCR）は、I²C バスインタフェースの動作設定を行います。
- I²C バスモードレジスタ（ICMR）は、MSB/LSB ファーストの選択および、ウェイト、転送クロック、転送ビットの選択を行います。ICMR は、ICCR の ICE ビットを 1 にセットしているときのみアクセス可能です。
- I²C バスステータスレジスタ（ICSR）は、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。
- I²C バスデータレジスタ（ICDR）は、送信および受信を兼ねたデータレジスタです。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。送信時に ICDR へ送信データをライトすると、ICDRT へ格納され、前のデータの送信後、自動的に ICDRS へ転送されます。ICDRS がデータを受信すると、自動的に ICDRR へと転送され、ICDR をリードすることで受信データを読み取ることができます。ICDR は、ICCR の ICE ビットを 1 にセットしているときのみアクセス可能です。
- スレーブアドレスレジスタ（SAR）は、SARX の FSX ビットと組み合わせてフォーマットの設定を行います。また、スレーブ動作時のスレーブアドレスを格納します。SAR は、ICCR の ICE ビットを 0 にクリアしているときのみアクセス可能です。
- 第 2 スレーブアドレスレジスタ（SARX）は、SAR の FS ビットと組み合わせてフォーマットの設定を行います。また、スレーブ動作時の第 2 スレーブアドレスを格納します。SARX は、ICCR の ICE ビットを 0 にクリアしているときのみアクセス可能です。

1.2.2 DTC（Data Transfer Controller）

本タスク例では、DTC を用いて送信データの内蔵 RAM から I²C モジュールのデータレジスタへの転送を行います。図 4 に DTC モジュールのブロック図を示し、以下に説明します。

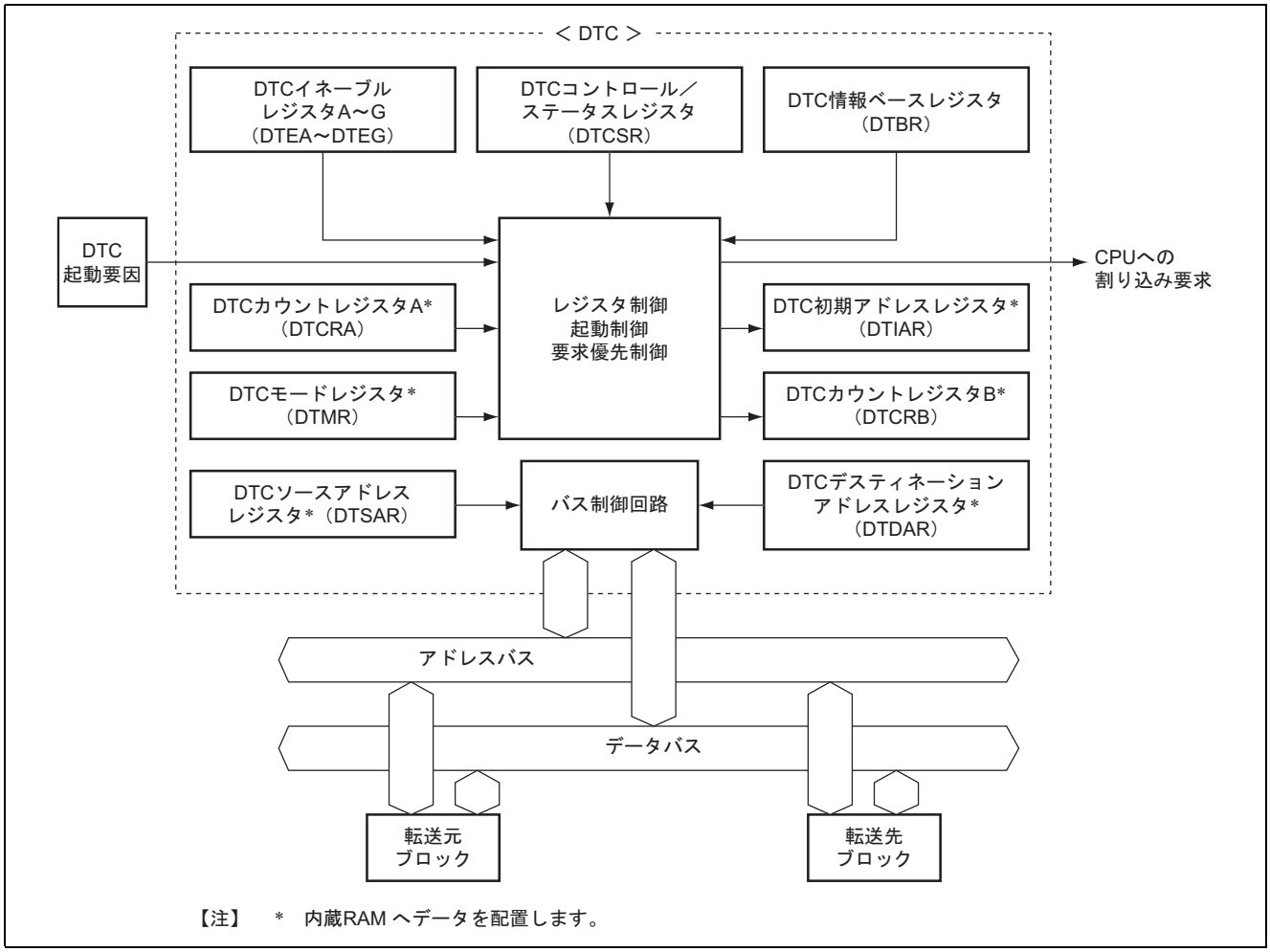


図 4 DTC モジュールのブロック図

- DTC を使用するには、DTC ベクタテーブルが必要です。ベクタアドレス等は、ハードウェアマニュアルを参照してください。
- 図 4 に示してある※印のレジスタは、CPU から直接アクセスすることはできません。これらのレジスタ情報は、内蔵 RAM に配置し、その先頭アドレスの上位 16 ビットを DTBR に、下位 16 ビットを DTC ベクタテーブルにセットします。DTC 動作時には、自動的に内蔵 RAM よりレジスタ情報をリードして転送を行います。
- DTC モードレジスタ（DTMR）は、転送データサイズなど、DTC の動作モードの設定を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、DTC 転送を行う転送元アドレスを指定します。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、DTC 転送を行う転送先アドレスを指定します。
- DTC 初期アドレスレジスタ（DTIAR）は、リピートモードの時に転送元／転送先の初期アドレスを指定します。
- DTC 転送カウントレジスタ A（DTCRA）は、DTC の転送回数を指定します。転送回数は、設定値が H’0000 のとき 65536 回となります。
- DTC 転送カウントレジスタ B（DTCRB）は、ブロック転送モードのとき、ブロック長を指定します。ブロック長は、設定値が H’0000 のとき 65536 となります。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタです。詳細はハードウェアマニュアルを参照してください。
- DTC コントロール/ステータスレジスタ（DTCSR）は、ソフトウェアによる DTC 起動の許可/禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。
- DTC 情報ベースレジスタ（DTBR）は、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR と DTC ベクタテーブルにより、DTC 転送情報の格納アドレスを指定します。DTBR へは、必ずワードまたはロングワード単位でアクセスを行ってください。

1.3 動作説明

本タスク例の EEPROM へのライト時のデータ内容を図 5 に示します。

開始条件発行後、スレーブアドレスと R/W ビットを 0（ライト指定）にして送信します。次に、データをライトする EEPROM の先頭アドレスの上位バイト・下位バイトを送信します。それから、実際にライトするデータを順次送信します。EEPROM からの ACK が 0 のとき、次のデータを送信します。最後に停止条件を発行します。

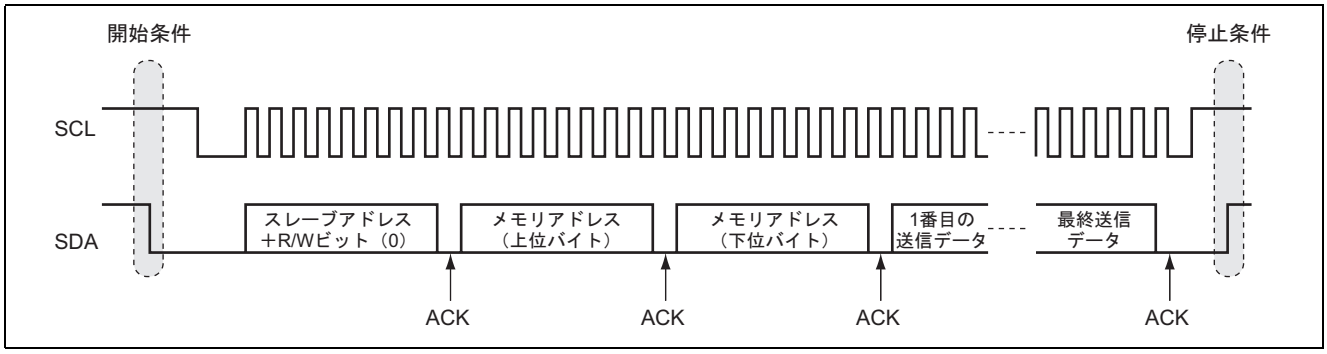


図 5　EEPROM へのライト時のデータ内容

EEPROM へのデータライト開始時の動作内容を図 6 に、終了時の動作内容を図 7 に示します。また、図 6 と図 7 の説明として、ソフトウェアおよびハードウェア処理の内容をそれぞれ表 3、表 4 に示します。

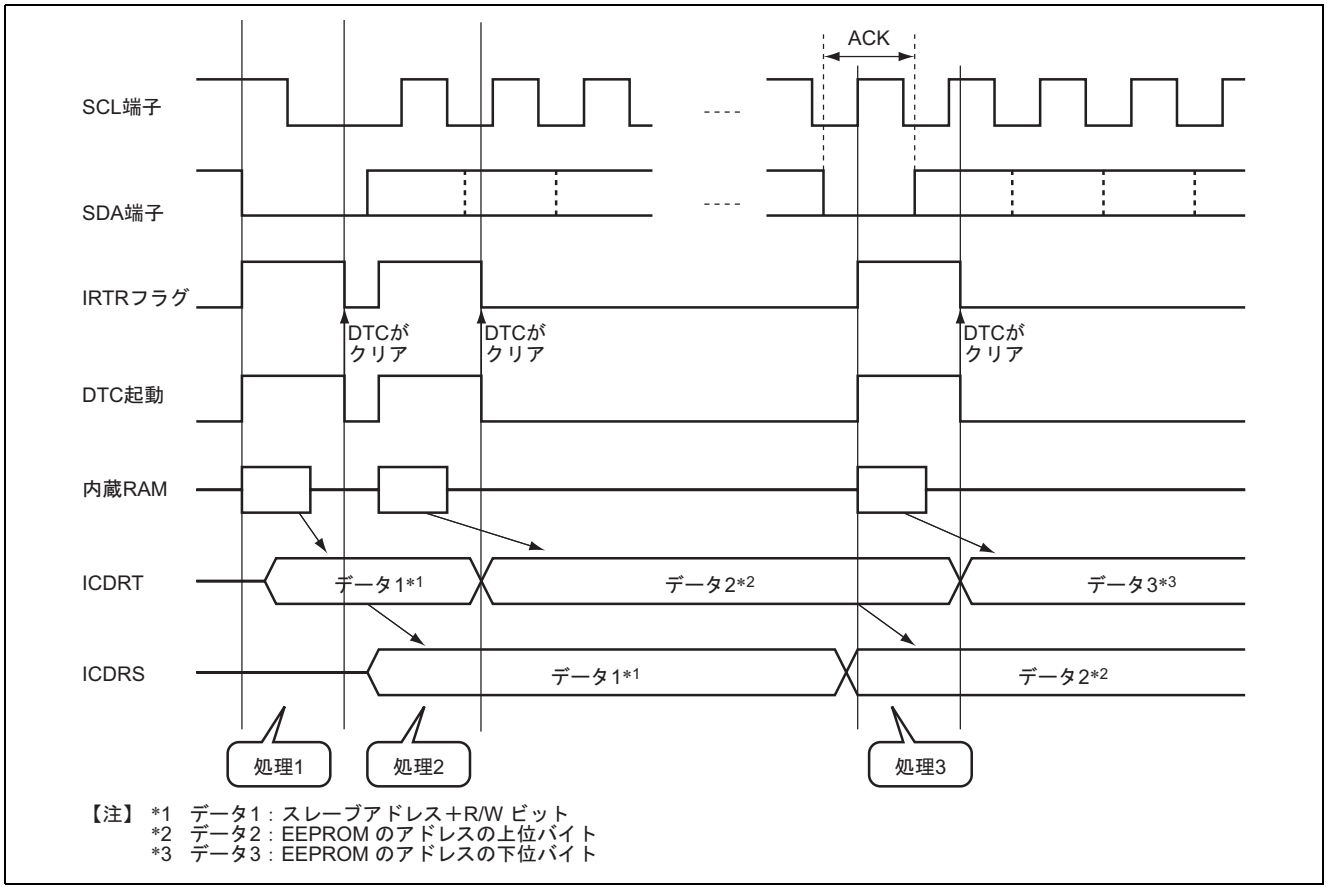


図 6　EEPROM へのデータライト開始時の動作

表 3　データライト開始時のソフトウェアおよびハードウェア処理内容

	ソフトウェア処理	ハードウェア処理
処理①	・ BBSY=1、SCP=0 をセットして開始条件を発行	・ 開始条件を検出して IRTR フラグを 1 にセット ・ IRTR フラグにより DTC 起動 ・ DTC が内蔵 RAM 上の送信データ（DATA1）を ICDRT（ICDR 内の送信バッファ）に転送 ・ DTC が IRTR フラグをクリア
処理②	—	・ ICDRT 上のデータ（DATA1）を ICDRS（ICDR 内のシフトレジスタ）に転送 ・ ICDRS 上のデータ（DATA1）を送信し、IRTR フラグを 1 にセット ・ IRTR フラグにより DTC 起動 ・ DTC が内蔵 RAM 上の送信データ（DATA2）を ICDRT に転送 ・ DTC が IRTR フラグをクリア
処理③	—	・ DATA1 の送信終了後、ICDRT のデータ（DATA2）を ICDRS に転送 ・ ICDRS 上のデータ（DATA2）を送信し、IRTR フラグを 1 にセット ・ IRTR フラグにより DTC 起動 ・ DTC が内蔵 RAM 上の送信データ（DATA3）を ICDRT に転送 ・ DTC が IRTR フラグをクリア

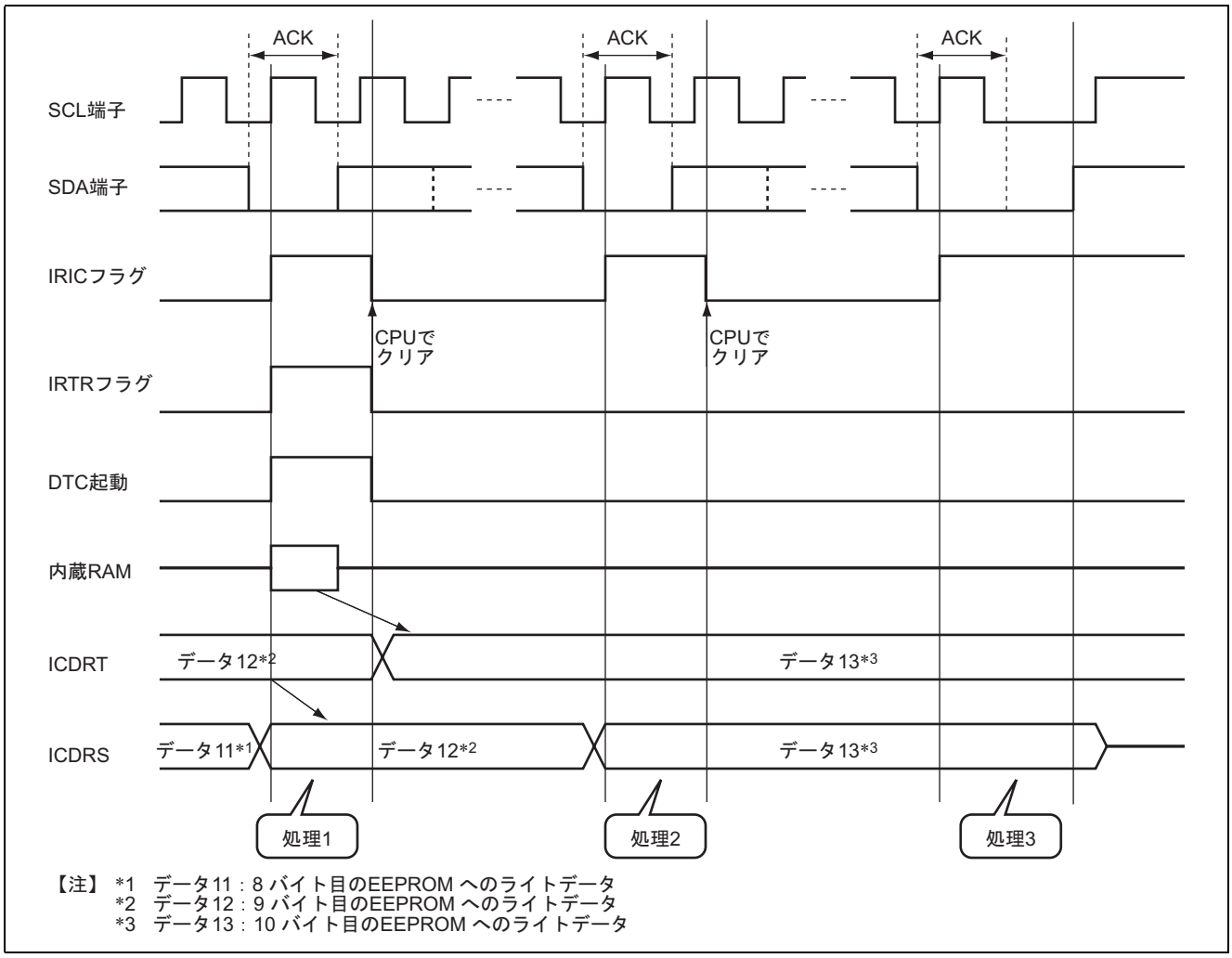


図 7　EEPROM へのデータライト終了時の動作

表 4　データライト終了時のソフトウェアおよびハードウェア処理内容

	ソフトウェア処理	ハードウェア処理
処理①	・ICI 割り込みを禁止 ・IRIC フラグをクリア ・IRIC フラグが 1 にセットされるまでウェイト	・DATA11 の送信終了後、IRTR および IRIC フラグを 1 にセット ・IRTR フラグにより DTC が起動 ・DTC が内蔵 RAM 上の送信データ（DATA13）を ICDRT に転送 ・DTC による転送が終了し、CPU へ割り込みを要求
処理②	・IRIC フラグをクリア ・IRIC フラグが 1 にセットされるまでウェイト	・DATA12 送信終了後、IRIC フラグを 1 にセット ・ICDRT 上のデータ（DATA13）を ICDRS に転送し、送信
処理③	・ICCR レジスタの ACKE ビットを 0 クリア ・BBSY=SCP=0 をセットして停止条件を発行	・DATA13 送信終了後、IRIC フラグを 1 にセット

1.4 ソフトウェア説明

1.4.1 モジュール説明

表 5 に本タスク例のモジュールを示します。

表 5　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	I²C モジュールのスタンバイ解除、端子機能およびライトデータの設定
I²C マスタ送信ルーチン	iic_mast_trans	EEPROM へのデータライト
I²C 割り込みルーチン	int_iic	最終フレーム処理と停止条件の発行

1.4.2 内部レジスタ説明

表 6 に本タスク例で使用する内部レジスタを示します。なお、設定値は本タスク例において使用している値であり、初期値とは異なります。

表 6　使用内部レジスタ説明（1）

レジスタ名	ビット	ビット名	設定値	機能
MSTCR1				モジュールスタンバイコントロールレジスタ 1
	9 8	MSTP25 MSTP24	0 0	DTC のスタンバイ制御ビット MSTP25=MSTP24=0 のとき DTC のスタンバイ解除
	5	MSTP21	0	I²C のスタンバイ制御ビット MSTP21=0 のとき I²C のスタンバイ解除
SCRX				シリアルコントロールレジスタ X
	7-6	—	0	リザーブビット
	5	IICX	1	I²C トランスファレートセレクト ICMR の CKS2～0 と組み合わせて、転送レートを選択
	4	IICE	1	I²C マスタイネーブル IICE=1 のとき ICDR および ICMR への CPU からのアクセスを許可
	3	HNDS	1	ハンドシェイク受信ビット HNDS=1 のとき連続受信動作を禁止
	2	—	0	リザーブビット
	1	ICDRF	0	ICDRF=0 のとき ICDR に有効な受信データがない
	0	STOPIM	1	停止条件検出割り込みマスク STOPIM=1 のときスレーブモード時に停止条件を検出した場合でも IRIC 割り込みの発生を禁止
ICMR				I²C バスモードレジスタ
	7	MLS	0	MSB ファースト/LSB ファースト選択 MLS=0 のとき MSB ファースト
	6	WAIT	0	ウェイト挿入ビット WAIT=0 のときデータと ACK を連続的に転送
	5 4 3	CKS2 CKS1 CKS0	1 1 1	転送クロック選択 2～0 SCRX の IICX0 と組み合わせて転送クロックの周波数を選択 （本タスク例では周波数=156kHz）
	2 1 0	BC2 BC1 BC0	0 0 0	ビットカウンタ 次に転送するデータのビット数を指定 （本タスク例ではビット/フレーム=9）

表 6　使用内部レジスタ説明（2）

レジスタ名	ビット	ビット名	設定値	機能
ICCR			—	I²C バスコントロールレジスタ
	7	ICE	1	I²C バスインタフェースイネーブル ICE=1 のとき I²C モジュールは転送可能状態
	6	IEIC	1	I²C バスインタフェース割り込みイネーブル IEIC=1 のとき CPU への割り込み許可
	5	MST	1	マスタ/スレーブ選択 MST=1 のとき I²C バスをマスタモードで使用
	4	TRS	1	送信/受信選択 TRS=1 のとき送信モード
	3	ACKE	1	アクノリッジビット判定選択 ACKE=1 のとき ACK=1 を検出した場合連続的な転送を中断
	2	BBSY	※1	バスビジーフラグ BBSY=0 のときバス開放状態 BBSY=1 のときバス占有状態
	1	IRIC	※2	I²C バスインタフェース割り込み要求フラグ IRIC=0 のとき転送待ち状態または転送中 IRIC=1 のとき割り込みが発生
	0	SCP	※1	開始条件/停止条件発行禁止ビット SCP=0 のとき BBSY と組み合わせて開始条件および停止条件を発行 SCP=1 のとき無効
PBCR1			H’0C00	ポート B コントロールレジスタ PBCR2 と組み合わせてポート B の端子機能の設定 本タスク例では SCL0(クロック入出力),SDA0(データ入出力)端子に設定
PBCR2			H’0000	ポート B コントロールレジスタ PBCR1 と組み合わせてポート B の端子機能の設定 本タスク例では SCL0(クロック入出力),SDA0(データ入出力)端子に設定
DTCRA			13	DTC 転送カウントレジスタ A DTC のデータ転送の回数を指定
DTSAR			& WR_DATA[0]	DTC ソースアドレスレジスタ DTC による転送の転送元アドレスを指定 I²C による送信データの格納アドレスを設定（内蔵 RAM）
DTDAR			H’FFFF880E	DTC デスティネーションアドレスレジスタ DTC による転送の転送先アドレスを指定 I²C モジュールの ICDR レジスタに設定
DTBR			H’FFFF	DTC 情報ベースレジスタ DTC 転送情報を格納するアドレスの上位 16 ビットを指定
DTEG			H’80	DTC イネーブルレジスタ G DTC を起動する割り込み要因を ICI 割り込みに設定

【注】※1　BBSY=1 かつ SCP=0 で開始条件を、BBSY=0 かつ SCP=0 で停止条件を発行します。
※ 2　ハードウェアにより 1 にセットされます。

表 6　使用内部レジスタ説明（3）

レジスタ名			設定値	機能
	ビット	ビット名		
DTMR			—	DTC モードレジスタ
	15 14	SM1 SM0	1 0	ソースアドレスモード SM[1,0]=B’10 のとき DTSAR はインクリメント
	13 12	DM1 DM0	0 0	デスティネーションアドレスモード DM[1,0]=B’0X のとき DTDAR は固定
	11 10	MD1 MD0	0 0	DTC モード MD[1,0]=B’00 のときノーマルモード転送
	9 8	Sz1 Sz0	0 0	DTC データトランスファサイズ Sz[1,0]=B’00 のときバイト転送
	7	DTS	0	DTC 転送モードセレクト 本タスク例ではノーマルモード転送のため影響無し
	6	CHNE	0	DTC チェイン転送イネーブル CHNE=0 のときチェイン転送を解除
	5	DISEL	0	DTC インタラプトセレクト DISEL=0 のとき指定した全データの転送終了後 CPU へ割り込み要求を発生
	4	NMIM	0	DTC NMI モード NMIM=0 のとき NMI により DTC 転送を中断
	3-0	—	0	リザーブビット

1.4.3　使用 RAM 説明

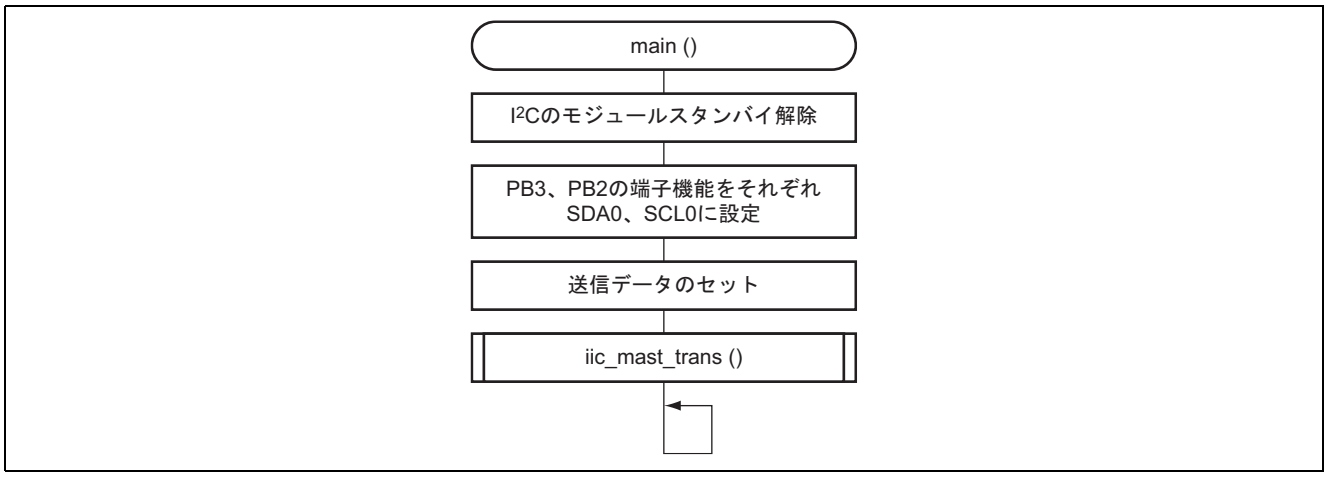
本タスク例での RAM の使用状況を表 7 に示します。

表 7　使用 RAM 説明

ラベル名	内容	アドレス	使用モージュール
WR_DATA[0-12]	EEPROM へのライトデータの格納 （送信時の DTC 転送元アドレス）	内蔵 RAM	メインルーチン I²C マスタ送信ルーチン

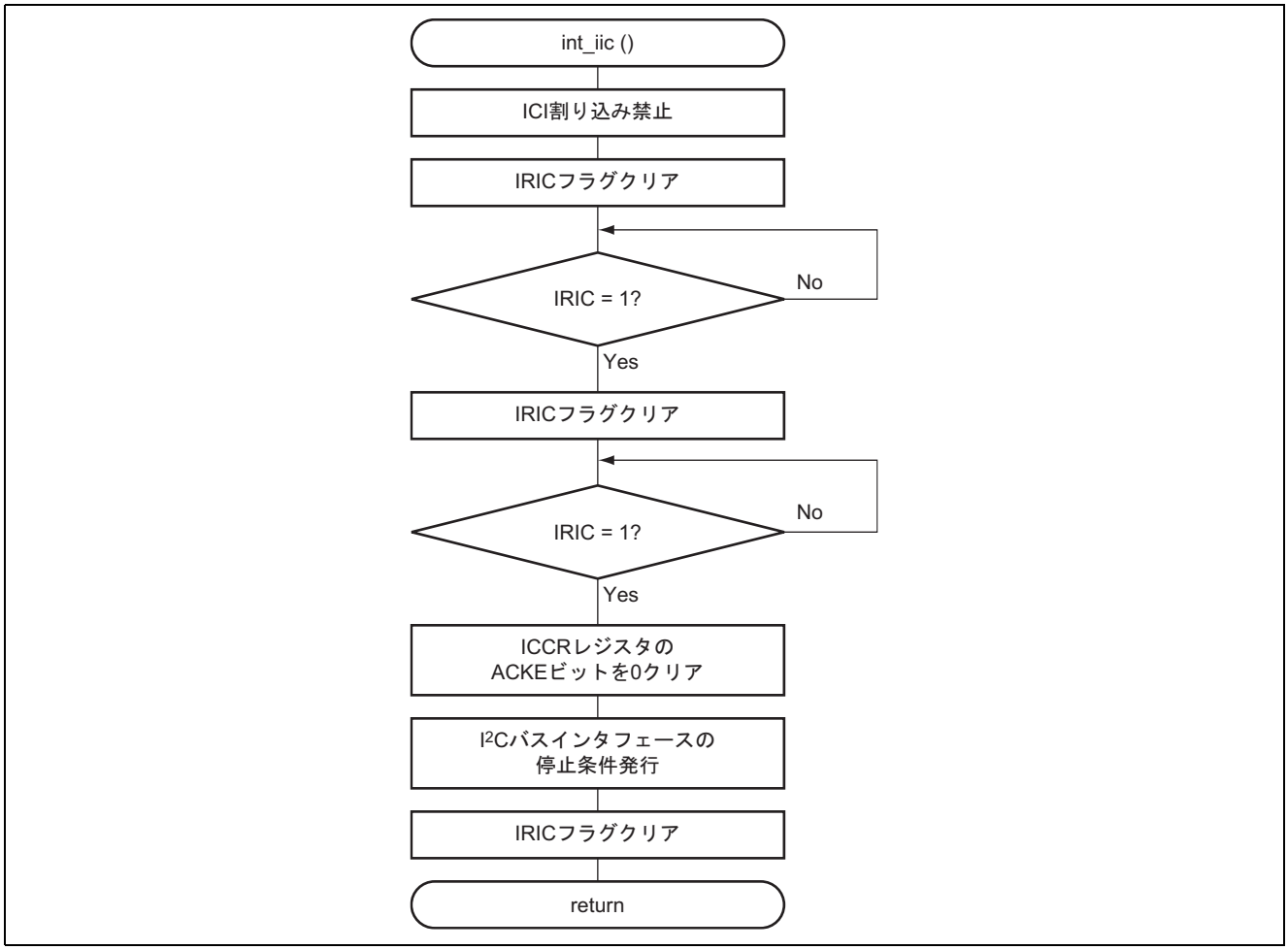
1.5 フローチャート

1.5.1 メインルーチン



1.5.2 I²C マスタ送信ルーチン

1.5.3 I²C 割り込みルーチン



1.6 プログラムリスト

```
/************************************************************************/
/* SH7145F  Application Note                                            */
/*                                                                      */
/* Function                                                             */
/* :I2C,DTC(EEPROM;HN58X2464,64k bit,8word×8bit)                        */
/* :Single Master Transmit                                              */
/*                                                                      */
/* External input clock         :10MHz                                  */
/* Internal CPU clock           :40MHz                                  */
/* Internal peripheral clock    :40MHz                                  */
/*                                                                      */
/* Written     :2004/1     Rev.1.0                                      */
/************************************************************************/

#include "iodefine.h"
#include <machine.h>

/*------------ Symbol Definition -------------------------------------------*/
struct st_dtc_iic {
   unsigned short DTMR;
   unsigned short DTCRA;
   unsigned short dummy1;
   unsigned short dummy2;
   unsigned long DTSAR;
   unsigned long DTDAR;
};

#define D_CODE 0xA0          /* device code of EEPROM                       */
#define A_CODE 0x00          /* device address code of EEPROM               */
#define WR     0x00    /* write data B'0                                */

#define UP_ADDR   0x00       /* upper address of EEPROM                     */
#define LO_ADDR   0x00       /* lower address of EEPROM                     */

#define WR_NUM 13    /* the number of write data to EEPROM              */

/*------------ Function Definition -----------------------------------------*/
void main(void);
void iic_mast_trans(void);

void int_iic(void);
void dummy_f(void);

/*------------ RAM allocation Definition -----------------------------------*/

unsigned char WR_DATA[WR_NUM];     /* write data                            */

#define DTC_ICI (*(volatile struct st_dtc_iic*)0xFFFFF000)
               /* DTC information address                        */

/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
```

```
    P_STBY.MSTCR1.BIT.MSTP21 = 0;   /* disable IIC standby mode                */

    /* set of I2C pin function           */
    P_PORTB.PBCR1.WORD = 0x0C00;
    P_PORTB.PBCR2.WORD = 0x0000;    /* SDA0(PB3-32pin@SH7145F),SCL0(PB2-
31pin@SH7145F) */

    /* set of transmitting data          */
    WR_DATA[0] = (D_CODE|A_CODE|WR);     /* device code + R/W bit(0;Write)   */
    WR_DATA[1] = UP_ADDR;           /* set of upper address to EEPROM         */
    WR_DATA[2] = LO_ADDR;           /* set of lower address to EEPROM         */
    WR_DATA[3] = 0x11;              /* write data(H'0000:EEPROM address)      */
    WR_DATA[4] = 0x22;              /* write data(H'0001:EEPROM address)      */
    WR_DATA[5] = 0x33;              /* write data(H'0002:EEPROM address)      */
    WR_DATA[6] = 0x44;              /* write data(H'0003:EEPROM address)      */
    WR_DATA[7] = 0x55;              /* write data(H'0004:EEPROM address)      */
    WR_DATA[8] = 0x66;              /* write data(H'0005:EEPROM address)      */
    WR_DATA[9] = 0x77;              /* write data(H'0006:EEPROM address)      */
    WR_DATA[10] = 0x88;             /* write data(H'0007:EEPROM address)      */
    WR_DATA[11] = 0x99;             /* write data(H'0008:EEPROM address)      */
    WR_DATA[12] = 0xAA;             /* write data(H'0009:EEPROM address)      */

    iic_mast_trans();               /* write routine                          */

    while(1);                 /* LOOP                                  */
}

/************************************************************************/
/* function : iic_mast_trans                                            */
/* operation : data write to EEPROM by the I2C interface                */
/* argument : non-argument                                              */
/* return : non-return                                                  */
/************************************************************************/
void iic_mast_trans(void)
{
    P_IIC.SCRX.BIT.IICE = 1;        /* register access enable from CPU        */

    P_IIC.ICCR0.BYTE = 0xF9;
          // ICE   [7]=B'1 : I2C interface enable
          // IECE  [6]=B'1 : I2C interrupt enable
          // MST   [5]=B'1 : master mode
          // TRS   [4]=B'1 : transmit mode
          // ACKE  [3]=B'1 : continuous data transfer is halted
          // BBSY  [2]=B'0
          // IRIC  [1]=B'0
          // SCP   [0]=B'1
    P_IIC.ICMR0.BYTE = 0x38;
          // MLS   [7]=B'0 : MSB first
          // WAIT  [6]=B'0    data and an acknowledge are transmitted continuously
          // CKS   [5-3]=B'111 : transfer clock(with IICX0) = 156kHz(Pφ=40MHz)
          // BC [2-0]=B'000
    P_IIC.SCRX.BYTE = 0x39;
          // Reserve[7-6]=B'00
          // IICX0    [5]=B'1 : transfer rate select, reference CKS bit
          // IICE     [4]=B'1 : register access enable from CPU
          // HNDS     [3]=B'1
          // Reserve  [2]=B'0
```

```
        // ICDRF [1]=B'0
        // STOPIM   [0]=B'1 : disables interrupt requests

    /* DTC initialize   */
    DTC_ICI.DTMR = 0x8000;              /* set DTC mode                      */
        // SM [15-14]=B'10 : source address increment
        // DM [13-12]=B'00 : destination address unchanging
        // MD [11-10]=B'00 : normal mode
        // Sz [9-8]=B'00 : transfer data size = byte
        // DTS   [7]=B'0 : not used
        // CHNE  [6]=B'0 : not used
        // DISEL [5]=B'0 : DTC interrupt disable
        // NMIM  [4]=B'0 : NMI->Terminate DTC transfer
        // Reserve[3-0]=B'0000
    DTC_ICI.DTCRA = WR_NUM;         /* transfer count                        */
    DTC_ICI.DTSAR = (unsigned long)&(WR_DATA[0]);
                  /* set source address                    */
    DTC_ICI.DTDAR = (unsigned long)&(P_IIC.ICDR0.BYTE);
                  /* set destination address                    */

    P_DTC.DTBR = 0xFFFF;                /* DTC information base register     */
    P_DTC.DTEG.BYTE |= 0x80;        /* interrupt sources IIC                 */

    P_INTC.IPRJ.BIT.IIC = 10;           /* set of IIC interrupt level        */
    set_imask(0);             /* clear interrupt mask level             */

    P_IIC.ICCR0.BYTE = ((P_IIC.ICCR0.BYTE & 0xFE) | 0x04);
                  /* generate the start condition               */
}

/****************************************/
/* Interruption Program                 */
/****************************************/
/*************************************************************************/
/* function : int_iic                                                    */
/* operation : transmission end interruption                             */
/* argument : non-argument                                               */
/* return : non-return                                                   */
/*************************************************************************/
#pragma interrupt(int_iic)
void int_iic(void)
{
    P_IIC.ICCR0.BIT.IEIC = 0;           /* IEIC interrupt disable            */

    P_IIC.ICCR0.BIT.IRIC = 0;           /* clear IRIC flag                   */
    while(P_IIC.ICCR0.BIT.IRIC != 1);         /* wait 1 byte transmitted     */

    P_IIC.ICCR0.BIT.IRIC = 0;           /* clear IRIC flag                   */
    while(P_IIC.ICCR0.BIT.IRIC != 1);         /* wait 1 byte transmitted     */
    P_IIC.ICCR0.BIT.ACKE = 0;       /* clear ACKE bit                        */
    P_IIC.ICCR0.BYTE = P_IIC.ICCR0.BYTE & 0xFA; /* generate stop condition    */
}

/************************************************/
/* other Interruption Program                   */
/************************************************/
#pragma interrupt(dummy_f)
```

```
void dummy_f(void)
{
   /* Other Interrupt  */
}
```

2. DTC を併用した I²C バスシングルマスタ受信

2.1 仕様

SH7145F の DTC（Data Transfer Controller）を併用した I²C バス（Inter IC Bus）によるデータの受信を行います。

図 8 に示すように、マスタデバイスを SH7145F とし、スレーブデバイスに EEPROM(HN58X2464、64k bit ,8word×8bit)を接続しています。SH7145F の I²C バスインタフェースを用いて、EEPROM から 10 バイトのデータを読み出し、内蔵 RAM に格納します。読み出したデータは、DTC を用いて内蔵 RAM へ転送します。

図 9 に DTC によるデータ転送を示します。また、表 8 に DTC の、表 9 に I²C バスインタフェースの設定を示します。

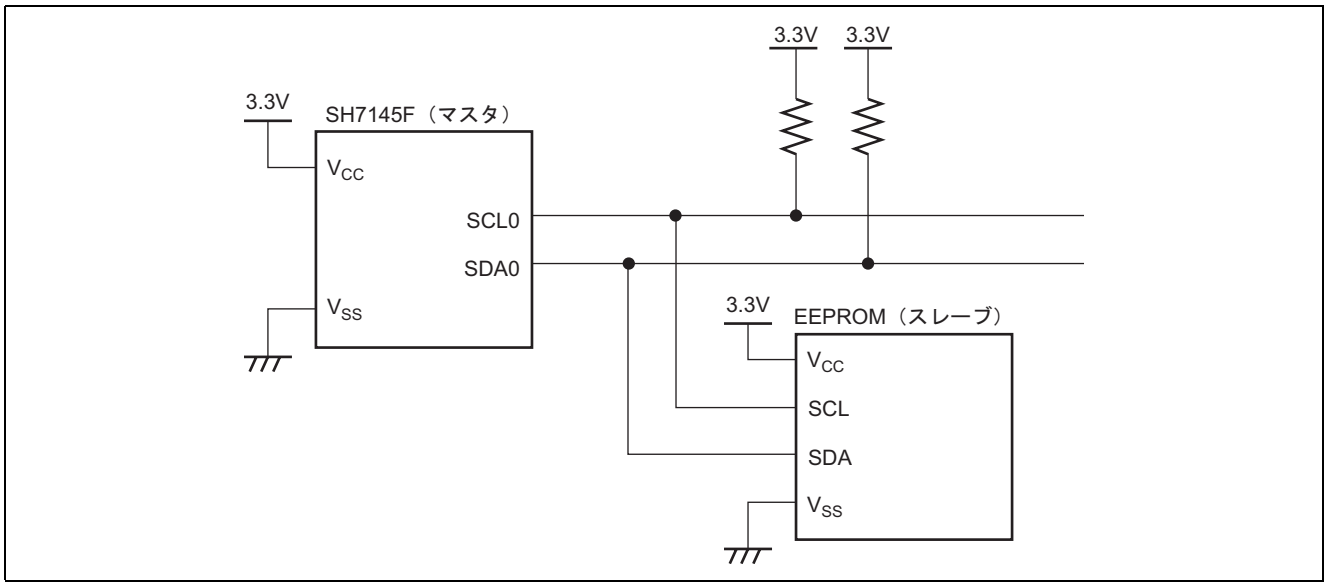


図 8　SH7145F と EEPROM の接続図

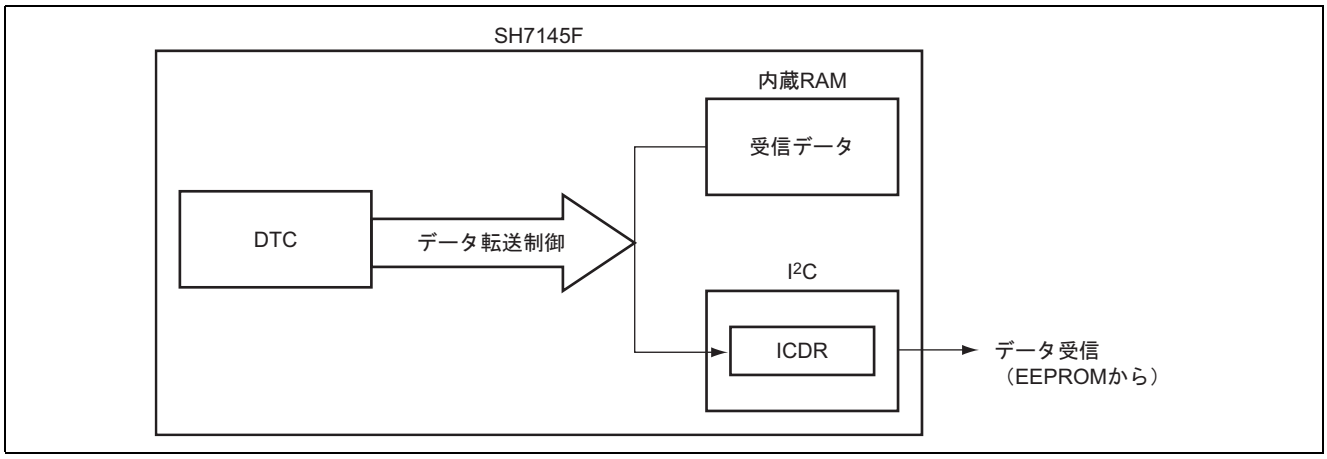


図 9　DTC によるデータ転送

表 8　DTC の設定

条件	内容
転送モード	ノーマルモード
転送データサイズ	1 バイト
転送回数（DTCRA）	13
転送元	ICDR（I²C バスデータレジスタ；H'FFFF880E）
転送先	内蔵 RAM
転送元アドレス	アドレス固定
転送先アドレス	転送後にアドレスをインクリメント
転送情報格納アドレス	H'FFFFF000
起動要因	I²C による割り込み（ICI）で起動
割り込み	禁止

表 9　I²C の設定

フォーマット内容	設定
動作	マスタ受信
転送クロック	156kHz（Pφ=40MHz）
データのビット数	9 ビット（ACK 含む）
データと ACK 間のウェイト	無し
割り込み	許可
ACK の送信	最終データ受信時のみ 1 を出力 連続受信時は 0 を出力

2.2 使用機能説明

本タスク例では、I²C バス（Inter IC Bus）を用いて EEPROM からデータのリードを行います。リードデータは、DTC（Data Transfer Controller）を用いて I²C バスインタフェースのデータレジスタから内蔵 RAM へと転送します。

2.2.1 I²C バス（Inter IC Bus）インタフェース

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。図 10 に I²C モジュールのブロック図を示し、以下に説明します。

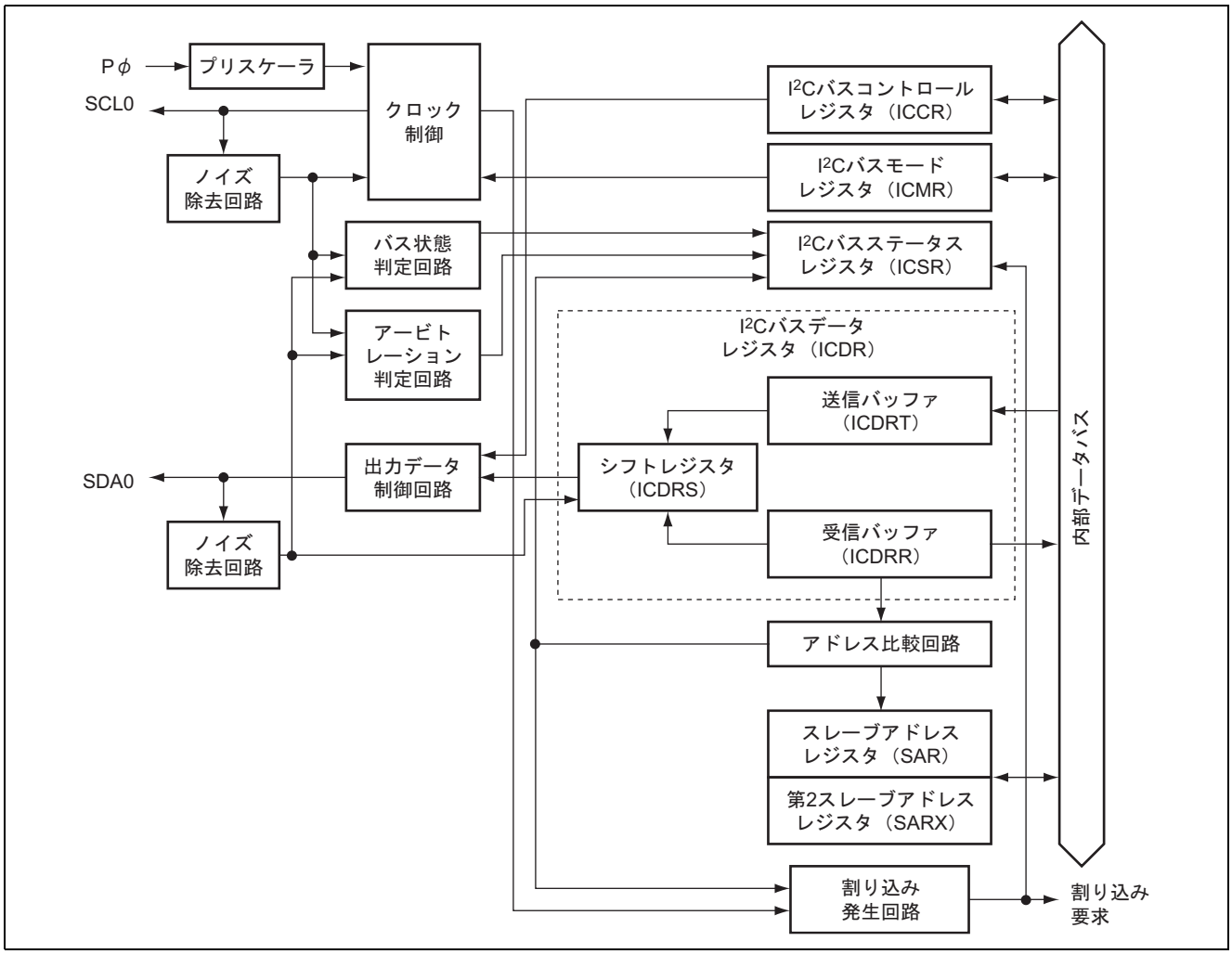


図 10　I²C モジュールのブロック図

- I²C バスコントロールレジスタ（ICCR）は、I²C バスインタフェースの動作設定を行います。
- I²C バスモードレジスタ（ICMR）は、MSB/LSB ファーストの選択および、ウェイト、転送クロック、転送ビットの選択を行います。ICMR は、ICCR の ICE ビットを 1 にセットしているときのみアクセス可能です。
- I²C バスステータスレジスタ（ICSR）は、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。
- I²C バスデータレジスタ（ICDR）は、送信および受信を兼ねたデータレジスタです。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。送信時に ICDR へ送信データをライトすると、ICDRT へ格納され、前のデータの送信後、自動的に ICDRS へ転送されます。ICDRS がデータを受信すると、自動的に ICDRR へと転送され、ICDR をリードすることで受信データを読み取ることができます。ICDR は、ICCR の ICE ビットを 1 にセットしているときのみアクセス可能です。
- スレーブアドレスレジスタ（SAR）は、SARX の FSX ビットと組み合わせてフォーマットの設定を行います。また、スレーブ動作時のスレーブアドレスを格納します。SAR は、ICCR の ICE ビットを 0 にクリアしているときのみアクセス可能です。
- 第 2 スレーブアドレスレジスタ（SARX）は、SAR の FS ビットと組み合わせてフォーマットの設定を行います。また、スレーブ動作時の第 2 スレーブアドレスを格納します。SARX は、ICCR の ICE ビットを 0 にクリアしているときのみアクセス可能です。

2.2.2 DTC（Data Transfer Controller）

本タスク例では、DTC を用いて受信データを I²C モジュールのデータレジスタから内蔵 RAM へ転送します。図 11 に DTC モジュールのブロック図を示し、以下に説明します。

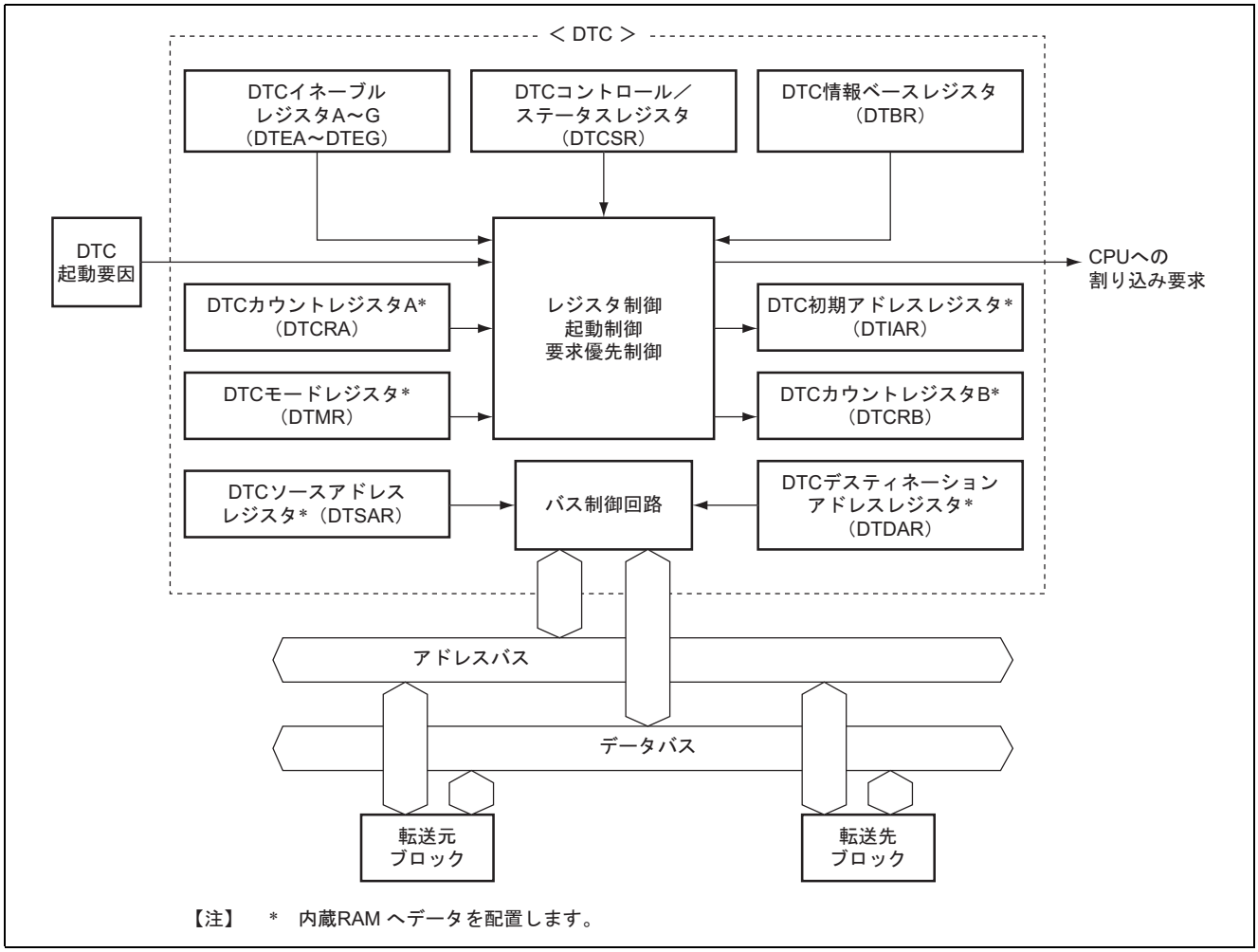


図 11　DTC モジュールのブロック図

- DTC を使用するには、DTC ベクタテーブルが必要です。ベクタアドレス等は、ハードウェアマニュアルを参照してください。
- 図 11 に示してある※印のレジスタは、CPU から直接アクセスすることはできません。これらのレジスタ情報は、内蔵 RAM に配置し、その先頭アドレスの上位 16 ビットを DTBR に、下位 16 ビットを DTC ベクタテーブルにセットします。DTC 動作時には、自動的に内蔵 RAM よりレジスタ情報をリードして転送を行います。
- DTC モードレジスタ（DTMR）は、転送データサイズなど、DTC の動作モードの設定を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、DTC 転送を行う転送元アドレスを指定します。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、DTC 転送を行う転送先アドレスを指定します。
- DTC 初期アドレスレジスタ（DTIAR）は、リピートモードの時に転送元／転送先の初期アドレスを指定します。
- DTC 転送カウントレジスタ A（DTCRA）は、DTC の転送回数を指定します。転送回数は、設定値が H’0000 のとき 65536 回となります。
- DTC 転送カウントレジスタ B（DTCRB）は、ブロック転送モードのとき、ブロック長を指定します。ブロック長は、設定値が H’0000 のとき 65536 となります。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタです。詳細はハードウェアマニュアルを参照してください。
- DTC コントロール/ステータスレジスタ（DTCSR）は、ソフトウェアによる DTC 起動の許可/禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。
- DTC 情報ベースレジスタ（DTBR）は、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR と DTC ベクタテーブルにより、DTC 転送情報の格納アドレスを指定します。DTBR へは、必ずワードまたはロングワード単位でアクセスを行ってください。

2.3 動作説明

本タスク例の EEPROM からのリード時のデータ内容を図 12 に示します。

開始条件発行後、スレーブアドレスと R/W ビットを 0（ライト指定）にして送信します。次に、リードする EEPROM の先頭アドレスの上位バイト・下位バイトを送信し、停止条件を発行します。

2 回目の開始条件発行後、スレーブアドレスと R/W ビットを 1（リード指定）にして送信します。それから、EEPROM から I²C フォーマットにのっとってデータが順次出力されます。マスタは、データを受信するとアクノリッジビット（ACK）を出力します。EEPROM は、ACK=0 を受信すると次のデータを出力します。マスタは、最後のデータを受信するときに ACK=1 を出力し、停止条件を発行します。

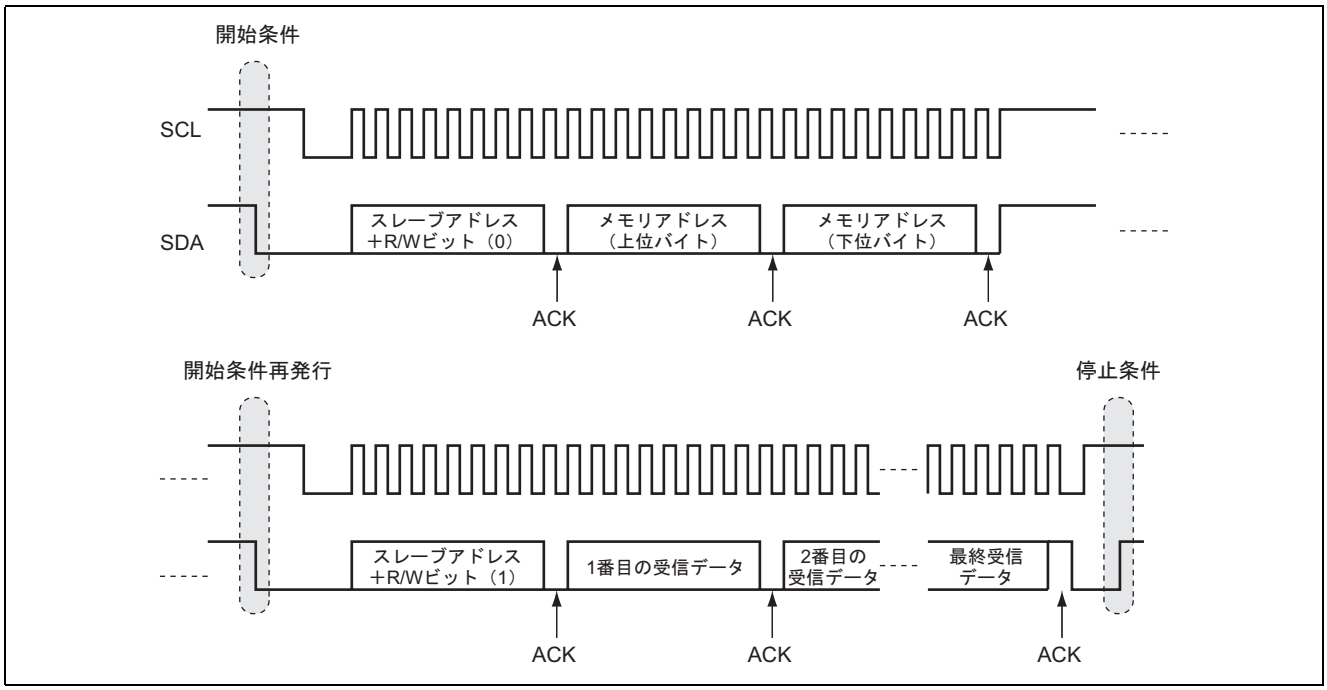


図 12　EEPROM からのリード時のデータ内容

EEPROM からのデータリード開始時の動作内容を図 13 に、終了時の動作内容を図 14 に示します。また、図 13 と図 14 の説明として、ソフトウェアおよびハードウェア処理の内容をそれぞれ表 10、表 11 に示します。

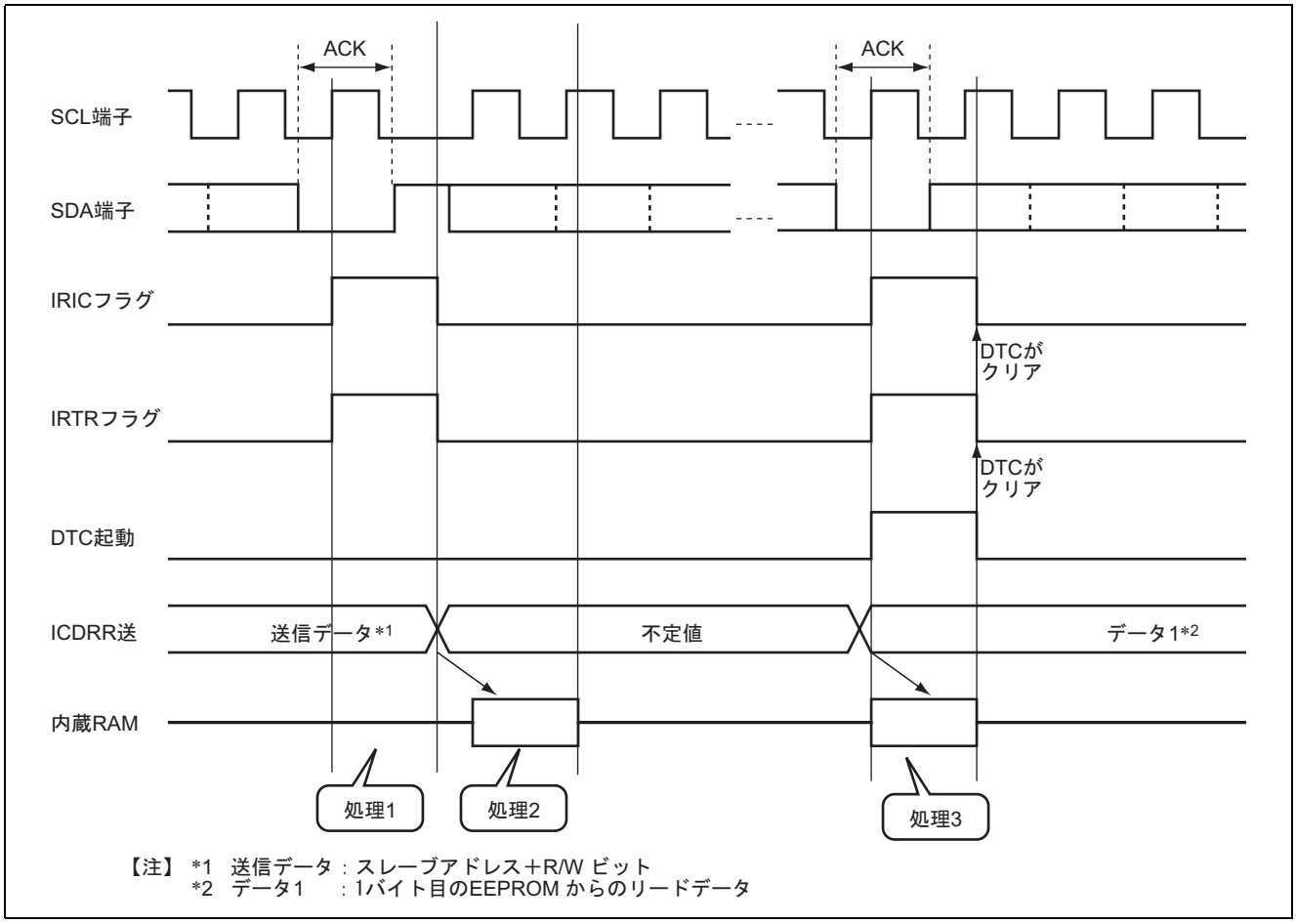


図 13　EEPROM からのデータリード開始時の動作

表 10　データリード開始時のソフトウェアおよびハードウェア処理内容

	ソフトウェア処理	ハードウェア処理
処理①	・IRIC フラグをクリア	・スレーブアドレス+R/W ビット送信後 IRIC および IRTR フラグを 1 にセット
処理②	・マスタ受信モード、ACK=1 に設定 ・ICI 割り込みを許可 ・DTC の転送条件を設定 ・ICDRR（ICDR 内の受信バッファ）上の不定値をダミーリード	・EEPROM からのリードデータ受信開始
処理③	—	・EEPROM からのリードデータ受信後、IRIC および IRTR フラグを 1 にセット ・IRTR フラグにより DTC が起動 ・受信データを ICDRR から内蔵 RAM に転送 ・IRIC および IRTR フラグをクリア

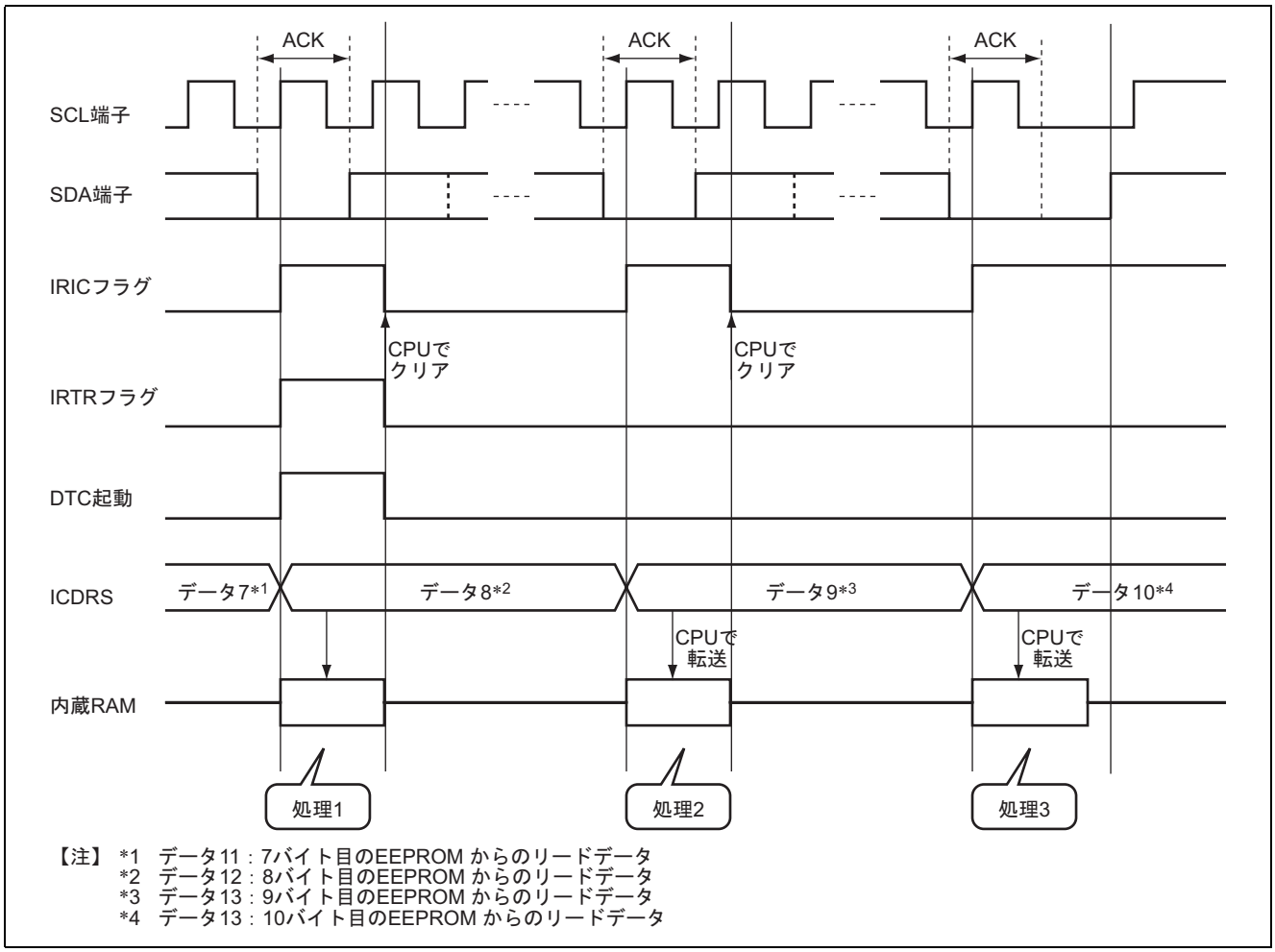


図 14　EEPROM からのデータリード終了時の動作

表 11　データリード終了時のソフトウェアおよびハードウェア処理内容

	ソフトウェア処理	ハードウェア処理
処理①	—	・EEPROM からのリードデータ受信後、IRIC および IRTR フラグを 1 にセット ・IRTR フラグにより DTC が起動 ・受信データを ICDRR から内蔵 RAM に転送 ・IRIC および IRTR フラグをクリア
処理②	・IRIC フラグをクリア ・ICI 割り込みを禁止 ・ACK=1 に設定 ・9 バイト目の受信データをリード	・EEPROM からのリードデータ受信後、IRIC および IRTR フラグを 1 にセット
処理③	・マスタ送信モードに設定 ・IRIC フラグをクリア ・10 バイト目の受信データをリード ・BBSY=SCP=0 をセットし、停止条件を発行	・EEPROM からのリードデータ受信後、IRIC および IRTR フラグを 1 にセット

2.4 ソフトウェア説明

2.4.1 モジュール説明

表 12 に本タスク例のモジュールを示します。

表 12 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	I²C モジュールのスタンバイ解除および端子機能設定
I²C マスタ受信ルーチン	iic_mast_rcv	EEPROM からのデータリード
I²C 割り込みルーチン	int_iic	最終フレーム処理と停止条件の発行
アドレス送信ルーチン	addr_EEPROM	EEPROM のリード開始アドレスの設定

2.4.2 内部レジスタ説明

表 13 に本タスク例で使用する内部レジスタを示します。なお、設定値は本タスク例において使用している値であり、初期値とは異なります。

表 13　使用内部レジスタ説明（1）

レジスタ名	ビット	ビット名	設定値	機能
MSTCR1				モジュールスタンバイコントロールレジスタ 1
	9	MSTP25	0	DTC のスタンバイ制御ビット
	8	MSTP24	0	MSTP25=MSTP24=0 のとき DTC のスタンバイ解除
	5	MSTP21	0	I²C のスタンバイ制御ビット MSTP21=0 のとき I²C のスタンバイ解除
SCRX				シリアルコントロールレジスタ X
	7-6	—	0	リザーブビット
	5	IICX	1	I²C トランスファレートセレクト ICMR の CKS2～0 と組み合わせて、転送レートを選択
	4	IICE	1	I²C マスタイネーブル IICE=1 のとき ICDR および ICMR への CPU からのアクセスを許可
	3	HNDS	1	ハンドシェイク受信ビット HNDS=1 のとき連続受信動作禁止
	2	—	0	リザーブビット
	1	ICDRF	0	ICDRF=0 のとき ICDR に有効な受信データがない
	0	STOPIM	1	停止条件検出割り込みマスク STOPIM=1 のときスレーブモード時に停止条件を検出した場合でも IRIC 割り込みの発生を禁止
ICMR				I²C バスモードレジスタ
	7	MLS	0	MSB ファースト/LSB ファースト選択 MLS=0 のとき MSB ファースト
	6	WAIT	0	ウェイト挿入ビット WAIT=0 のときデータと ACK を連続的に転送
	5	CKS2	1	転送クロック選択 2～0
	4	CKS1	1	SCRX の IICX と組み合わせて転送クロックの周波数を選択
	3	CKS0	1	（本タスク例では周波数=156kHz）
	2	BC2	0	ビットカウンタ
	1	BC1	0	次に転送するデータのビット数を指定
	0	BC0	0	（本タスク例ではビット/フレーム=9）

表 13　使用内部レジスタ説明（2）

レジスタ名	ビット	ビット名	設定値	機能
ICCR			—	I²C バスコントロールレジスタ
	7	ICE	1	I²C バスインタフェースイネーブル ICE=1 のとき I²C モジュールは転送可能状態
	6	IEIC	1	I²C バスインタフェース割り込みイネーブル IEIC=1 のとき CPU への割り込み許可
	5	MST	1	マスタ/スレーブ選択 MST=1 のとき I²C バスをマスタモードで使用
	4	TRS	0	送信/受信選択 TRS=0 のとき受信モード
	3	ACKE	1	アクノリッジビット判定選択 ACKE=1 のとき ACK=1 を検出した場合連続的な転送を中断
	2	BBSY	※1	バスビジーフラグ BBSY=0 のときバス開放状態 BBSY=1 のときバス占有状態
	1	IRIC	※2	I²C バスインタフェース割り込み要求フラグ IRIC=0 のとき転送待ち状態または転送中 IRIC=1 のとき割り込みが発生
	0	SCP	※1	開始条件/停止条件発行禁止ビット SCP=0 のとき BBSY と組み合わせて開始条件および停止条件を発行 SCP=1 のとき無効
PBCR1			H’0C00	ポート B コントロールレジスタ PBCR2 と組み合わせてポート B の端子機能の設定 本タスク例では SCL0(クロック入出力),SDA0(データ入出力)端子に設定
PBCR2			H’0000	ポート B コントロールレジスタ PBCR1 と組み合わせてポート B の端子機能の設定 本タスク例では SCL0(クロック入出力),SDA0(データ入出力)端子に設定
DTCRA			9	DTC 転送カウントレジスタ A DTC のデータ転送の回数を指定
DTSAR			H’FFFF880E	DTC ソースアドレスレジスタ DTC による転送の転送元アドレスを指定 I²C モジュールの ICDR レジスタに設定
DTDAR			&RD_DATA[0]	DTC デスティネーションアドレスレジスタ DTC による転送の転送先アドレスを指定 EEPROM からのリードデータの格納先を設定（内蔵 RAM）
DTBR			H’FFFF	DTC 情報ベースレジスタ DTC 転送情報を格納するアドレスの上位 16 ビットを指定
DTEG			H’80	DTC イネーブルレジスタ G DTC を起動する割り込み要因を I²C に設定

【注】※1　BBSY=1 かつ SCP=0 で開始条件を、BBSY=0 かつ SCP=0 で停止条件を発行します。
※ 2　ハードウェアにより 1 にセットされます。

表 13　使用内部レジスタ説明（2）

レジスタ名			設定値	機能
	ビット	ビット名		
DTMR			—	DTC モードレジスタ
	15 14	SM1 SM0	0 0	ソースアドレスモード SM[1,0]=B'0X のとき DTSAR は固定
	13 12	DM1 DM0	1 0	デスティネーションアドレスモード DM[1,0]=B'10 のとき DTDAR はインクリメント
	11 10	MD1 MD0	0 0	DTC モード MD[1,0]=B'00 のときノーマルモード転送
	9 8	Sz1 Sz0	0 0	DTC データトランスファサイズ Sz[1,0]=B'00 のときバイト転送
	7	DTS	0	DTC 転送モードセレクト 本タスク例ではノーマルモード転送のため影響無し
	6	CHNE	0	DTC チェイン転送イネーブル CHNE=0 のときチェイン転送を解除
	5	DISEL	0	DTC インタラプトセレクト DISEL=0 のとき指定した全データの転送終了後 CPU へ割り込み要求を発生
	4	NMIM	0	DTC NMI モード NMIM=0 のとき NMI により DTC 転送を中断
	3-0	—	0	リザーブビット

2.4.3　使用 RAM 説明

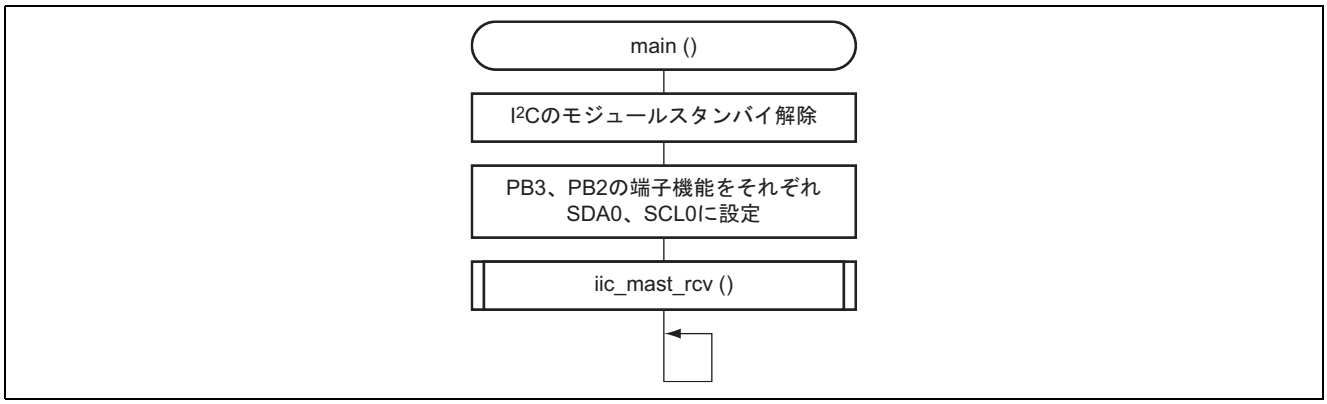
本タスク例での RAM の使用状況を表 14 に示します。

表 14　使用 RAM 説明

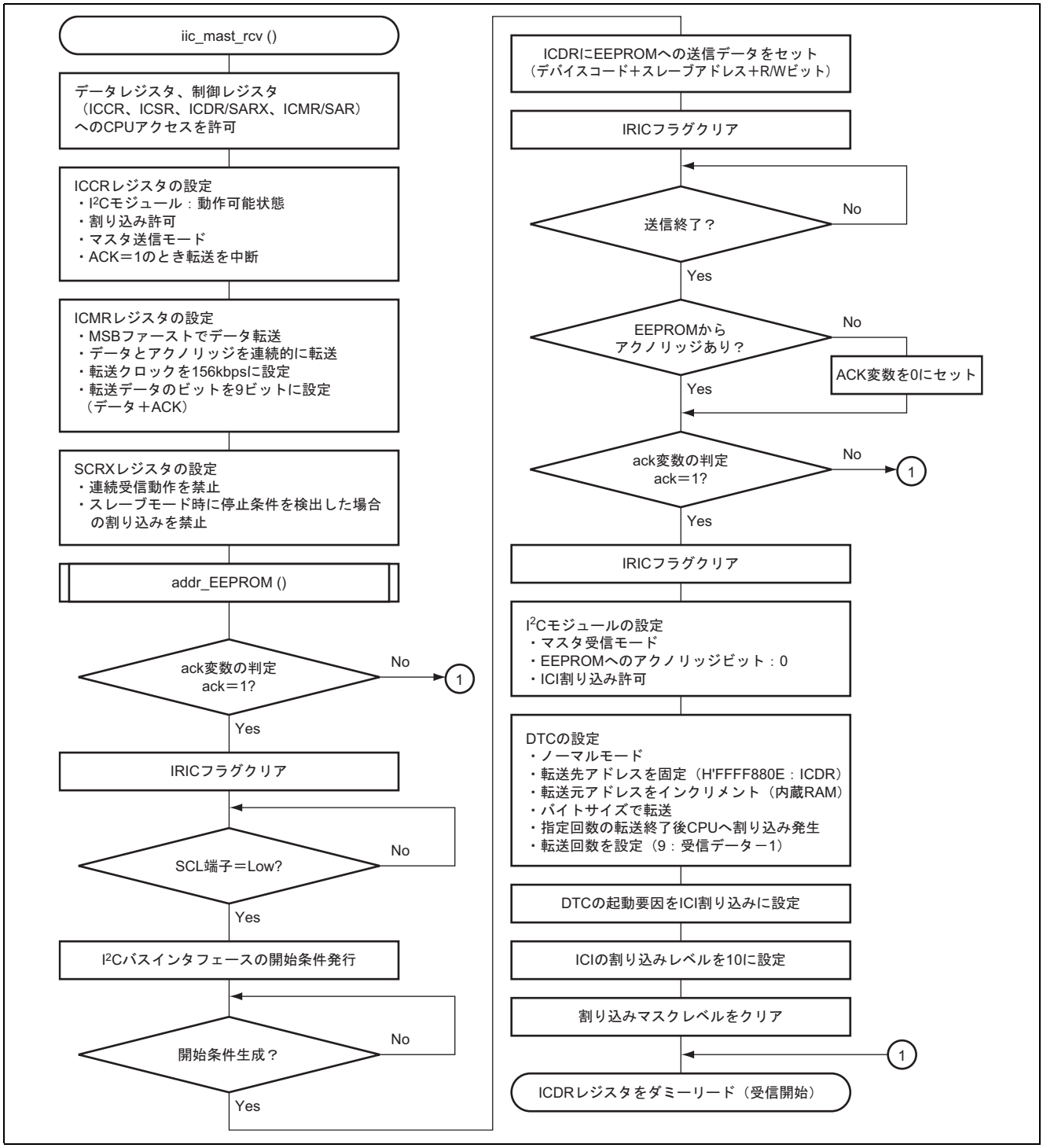
ラベル名	内容	アドレス	使用モージュール
RD_DATA[0-9]	EEPROM からのリードデータの格納 （受信時の DTC 転送先アドレス）	内蔵 RAM	I²C マスタ受信ルーチン I²C 割り込みルーチン
Dummy	ダミーリードデータの格納	内蔵 RAM	I²C マスタ受信ルーチン

2.5 フローチャート

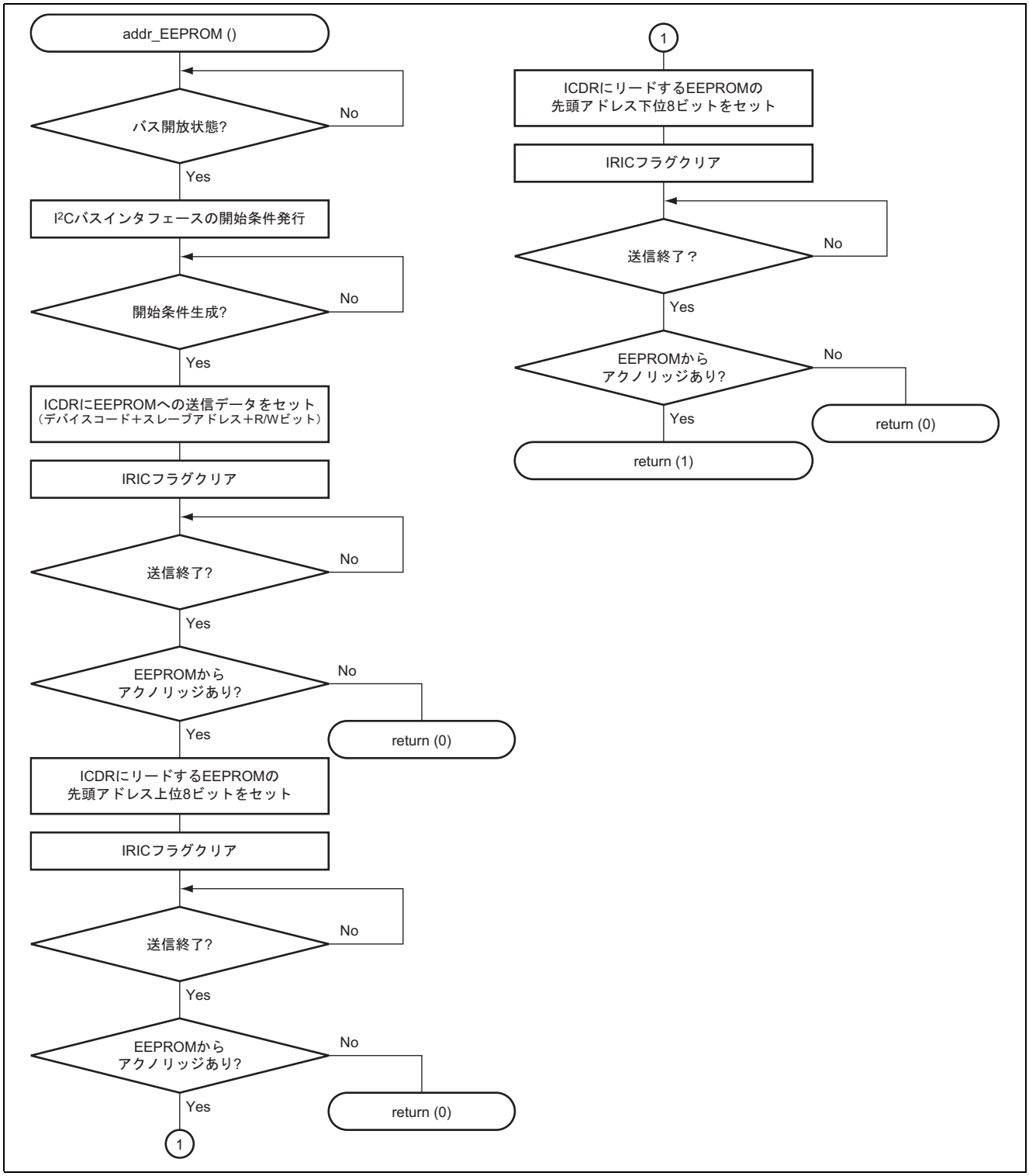
2.5.1 メインルーチン



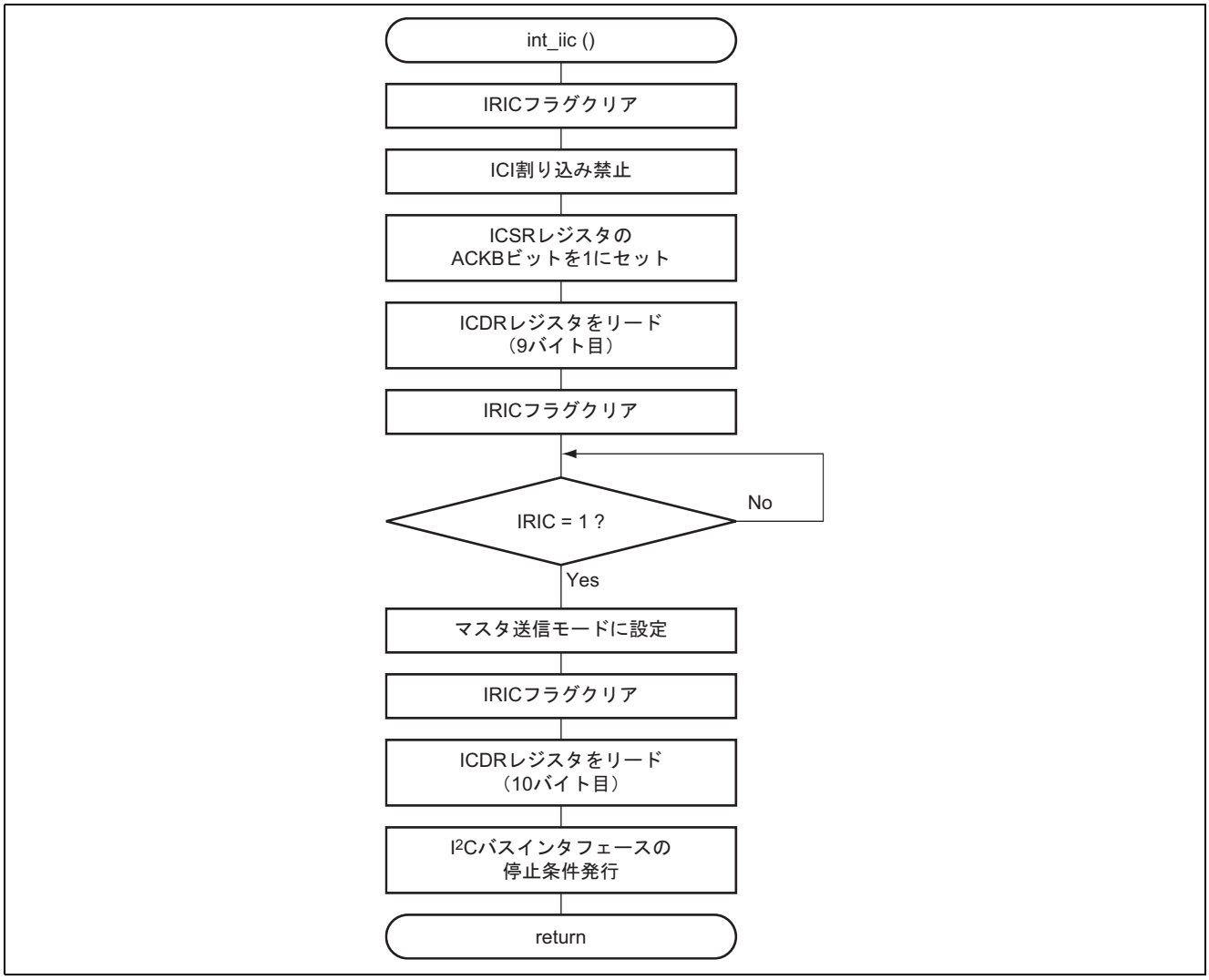
2.5.2 I²C マスタ受信ルーチン



2.5.3 アドレス送信ルーチン



2.5.4 I²C 割り込みルーチン



2.6 プログラムリスト

```
/*****************************************************************/
/* SH7145F  Application Note                                     */
/*                                                               */
/* Function                                                      */
/* :I2C,DTC(EEPROM;HN58X2464,64k bit,8word×8bit)                 */
/* :Single Master Receive                                        */
/*                                                               */
/* External input clock         :10MHz                           */
/* Internal CPU clock           :40MHz                           */
/* Internal peripheral clock    :40MHz                           */
/*                                                               */
/* Written  :2004/1  Rev.1.0                                     */
/*****************************************************************/

#include "iodefine.h"
#include <machine.h>

/*------------ Symbol Definition ------------------------------------------*/
struct st_dtc_iic {
   unsigned short DTMR;
   unsigned short DTCRA;
   unsigned short dummy1;
   unsigned short dummy2;
   unsigned long DTSAR;
   unsigned long DTDAR;
};

#define D_CODE 0xA0          /* device code of EEPROM                    */
#define A_CODE 0x00          /* device address code of EEPROM            */
#define RD     0x01    /* read data B'1                              */
#define WR     0x00    /* write data B'0                             */

#define UP_ADDR   0x00       /* upper address of EEPROM                  */
#define LO_ADDR   0x00       /* lower address of EEPROM                  */

#define RD_NUM 10      /* the number of read data from EEPROM          */

/*------------ Function Definition ----------------------------------------*/
void main(void);
void iic_mast_rcv(void);
unsigned char addr_EEPROM(void);

void int_iic(void);
void dummy_f(void);

/*------------ RAM allocation Definition ----------------------------------*/

unsigned char RD_DATA[RD_NUM];     /* read data                          */
unsigned char Dummy;         /* dummy read data                          */

#define DTC_ICI (*(volatile struct st_dtc_iic*)0xFFFFF000)
               /* DTC information address                          */
/*****************************************************************/
/* main Program                                                  */
/*****************************************************************/
```

```
void main( void )
{
   P_STBY.MSTCR1.BIT.MSTP21 = 0;   /* disable IIC standby mode               */

   /* set of I2C pin function           */
   P_PORTB.PBCR1.WORD = 0x0C00;
   P_PORTB.PBCR2.WORD = 0x0000;    /* SDA0(PB3-32pin@SH7145F),SCL0(PB2-
31pin@SH7145F) */

   iic_mast_rcv();                 /* read routine                           */

   while(1);                 /* LOOP                              */
}
/*************************************************************************/
/* function : iic_mast_rcv                                               */
/* operation : data read from EEPROM by the I2C interface                */
/* argument : non-argument                                               */
/* return : non-return                                                   */
/*************************************************************************/
void iic_mast_rcv(void)
{
   unsigned short d_count,data;
   unsigned char ack;

   P_IIC.SCRX.BIT.IICE = 1;        /* register access enable from CPU       */

   P_IIC.ICCR0.BYTE = 0xB9;
         // ICE   [7]=B'1 : I2C interface enable
         // IECE  [6]=B'0 : I2C interrupt enable
         // MST   [5]=B'1 : master mode
         // TRS   [4]=B'1 : transmit mode
         // ACKE  [3]=B'1 : continuous data transfer is halted
         // BBSY  [2]=B'0
         // IRIC  [1]=B'0
         // SCP   [0]=B'1
   P_IIC.ICMR0.BYTE = 0x38;
         // MLS   [7]=B'0 : MSB first
         // WAIT  [6]=B'0 : a wait state is inserted between DATA and ACK
         // CKS   [5-3]=B'111 : transfer clock(with IICX0) = 156kHz(Pφ=40MHz)
         // BC [2-0]=B'000
   P_IIC.SCRX.BYTE = 0x39;
         // Reserve  [7-6]=B'00
         // IICX0    [5]=B'1 : transfer rate select, reference CKS bit
         // IICE     [4]=B'1 : register access enable from CPU
         // HNDS     [3]=B'1
         // Reserve  [2]=B'0
         // ICDRF [1]=B'0
         // STOPIM   [0]=B'1 : disables interrupt requests

   ack = addr_EEPROM();

   if(ack == 1){             /* ACK OK                                */
      P_IIC.ICCR0.BIT.IRIC = 0;          /* clear IRIC flag                */
      while(P_PORTB.PBDR.BIT.PB2DR != 0);      /* check SCL0 pin            */

      P_IIC.ICCR0.BYTE = ((P_IIC.ICCR0.BYTE & 0xFE) | 0x04);
                  /* generate start condition                   */
```

```
        while(P_IIC.ICCR0.BIT.IRIC != 1);  /* wait 1byte transmitted           */

        P_IIC.ICDR0.BYTE = (D_CODE|A_CODE|RD);
                    /* device code + R/W bit(1;Read)              */
        P_IIC.ICCR0.BIT.IRIC = 0;          /* clear IRIC flag                  */
        while(P_IIC.ICCR0.BIT.IRIC != 1);  /* wait 1byte transmitted           */
        if(P_IIC.ICSR0.BIT.ACKB != 0){     /* ACK = 1                          */
           ack = 0;      /* EEPROM write error                          */
        }
    }

    if(ack == 1){
        P_IIC.ICCR0.BIT.IRIC = 0;          /* clear IRIC flag                  */

        P_IIC.ICCR0.BIT.TRS = 0;           /* receive mode                     */
        P_IIC.ICSR0.BIT.ACKB = 0;    /* set ACK=0                              */
        P_IIC.ICCR0.BIT.IEIC = 1;          /* IIC interrupt enable             */

    /* DTC initialize   */
        DTC_ICI.DTMR = 0x2000;       /* set DTC mode                    */
              // SM    [15-14]=B'00 : source address unchanging
              // DM    [13-12]=B'10 : destination address increment
              // MD    [11-10]=B'00 : normal mode
              // Sz    [9-8]=B'00 : transfer data size = byte
              // DTS     [7]=B'0 : not used
              // CHNE    [6]=B'0 : not used
              // DISEL   [5]=B'0 : DTC interrupt disable
              // NMIM    [4]=B'0 : NMI->Terminate DTC transfer
              // Reserve [3-0]=B'0000
        DTC_ICI.DTCRA = (RD_NUM - 1);      /* transfer count                   */
        DTC_ICI.DTSAR = (unsigned long)&(P_IIC.ICDR0.BYTE);
                    /* set source address                         */
        DTC_ICI.DTDAR = (unsigned long)&(RD_DATA[0]);
                    /* set destination address                    */

        P_DTC.DTBR = 0xFFFF;         /* DTC information base register          */
        P_DTC.DTEG.BYTE |= 0x80;     /* interrupt sources IIC                  */

        P_INTC.IPRJ.BIT.IIC = 10;          /* set IIC interrupt level          */
        set_imask(0);          /* clear interrupt mask level                   */

        Dummy = P_IIC.ICDR0.BYTE;    /* dummy read                             */
    }
}
/*************************************************************************/
/* function : addr_EEPROM                                                */
/* operation : transmission of a slave address and a memory-address      */
/* argument : non-argument                                               */
/* return : ack                                                          */
/*************************************************************************/
unsigned char addr_EEPROM(void)
{
    while(P_IIC.ICCR0.BIT.BBSY != 0);          /* judging bus condition     */

    P_IIC.ICCR0.BYTE = ((P_IIC.ICCR0.BYTE & 0xFE) | 0x04);
                    /* generate start condition                   */
    while(P_IIC.ICCR0.BIT.IRIC != 1);          /* wait 1 byte transmitted   */
```

```
    // transmission of a slave address
    P_IIC.ICDR0.BYTE = (D_CODE|A_CODE|WR);     /* set of transmission data   */
    P_IIC.ICCR0.BIT.IRIC = 0;             /* clear IRIC flag                 */
    while(P_IIC.ICCR0.BIT.IRIC != 1);     /* wait 1 byte transmitted         */
    if(P_IIC.ICSR0.BIT.ACKB != 0){        /* judging ACK bit                 */
       return(0);       /* no ACK bit                                 */
    }

    // transmission of a upper-byte memory-address
    P_IIC.ICDR0.BYTE = UP_ADDR;           /* set of transmission data        */
    P_IIC.ICCR0.BIT.IRIC = 0;             /* clear IRIC flag                 */
    while(P_IIC.ICCR0.BIT.IRIC != 1);     /* wait 1 byte transmitted         */
    if(P_IIC.ICSR0.BIT.ACKB != 0){        /* judging ACK bit                 */
       return(0);       /* no ACK bit                                 */
    }

    // transmission of a lower-byte memory-address
    P_IIC.ICDR0.BYTE = LO_ADDR;           /* set of transmission data        */
    P_IIC.ICCR0.BIT.IRIC = 0;             /* clear IRIC flag                 */
    while(P_IIC.ICCR0.BIT.IRIC != 1);     /* wait 1byte transmitted          */
    if(P_IIC.ICSR0.BIT.ACKB != 0){        /* judging ACK bit                 */
       return(0);       /* no ACK bit                                 */
    }
    return(1);          /* ACK OK                           */
}
/*****************************************/
/* Interruption Program                  */
/*****************************************/
/************************************************************************/
/* function : int_iic                                                   */
/* operation : reception end interruption                               */
/* argument : non-argument                                              */
/* return : non-return                                                  */
/************************************************************************/
#pragma interrupt(int_iic)
void int_iic(void)
{
   P_IIC.ICCR0.BIT.IRIC = 0;              /* clear IRIC flag                 */
   P_IIC.ICCR0.BIT.IEIC = 0;              /* IEIC interrupt disable          */

   P_IIC.ICSR0.BIT.ACKB = 1;       /* set of ACK(=1)                         */

   RD_DATA[8] = P_IIC.ICDR0.BYTE;         /* read ICDR0 register             */
   P_IIC.ICCR0.BIT.IRIC = 0;              /* clear IRIC flag                 */
   while(P_IIC.ICCR0.BIT.IRIC != 1);      /* wait 1 byte to be received      */

   P_IIC.ICCR0.BIT.TRS = 1;               /* IIC transmit mode               */

   P_IIC.ICCR0.BIT.IRIC = 0;              /* clear IRIC flag                 */

   RD_DATA[9] = P_IIC.ICDR0.BYTE;         /* read ICDR0 register             */

   P_IIC.ICSR0.BIT.ACKB = 0;       /* set of ACK(=0)                         */

   P_IIC.ICCR0.BYTE = P_IIC.ICCR0.BYTE & 0xFA; /*generate stop condition     */
}
```

```
/************************************************/
/* other Interruption Program                   */
/************************************************/
#pragma interrupt(dummy_f)
void dummy_f(void)
{
   /* Other Interrupt  */
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.09.15	—	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。