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

H8S/2655シリーズ

アプリケーションノート　ー内蔵I/O編ー

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.1.00　1996.09

はじめに

Ｈ８／２６５５は、内部３２ビット構成のＨ８Ｓ／２６００ＣＰＵを核にして、豊富な各種周辺機能を内蔵した高性能マイクロコンピュータです。

１チップ上にＣＰＵ、ＲＡＭ、１６ビットタイマパルスユニット（ＴＰＵ）、プログラマブルパルスジェネレータ（ＰＰＧ）、シリアルコミュニケーションインタフェース（ＳＣＩ）、データトランスファコントローラ（ＤＴＣ）およびＤＭＡコントローラ（ＤＭＡＣ）等の周辺機能を内蔵しており、高性能かつ小型システムのアプリケーションに適用できます。

本アプリケーションノートは、内蔵周辺機能を単独で使用した場合の動作例を示した基礎編と、内蔵周辺機能を組み合わせて使用した応用編の２部構成となっています。

なお、本アプリケーションノートに掲載されているプログラム、回路等の動作は確認しておりますが、実際にご使用になる場合は、改めて動作確認の上ご使用くださいますようお願い致します。

目　次

1. H8S／2655シリーズ アプリケーションノート使用手引

目　次

本アプリケーションノートは図１．１に示すように２部構成になっています。

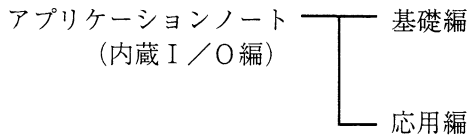


図１．１　アプリケーションノート構成

（１）基礎編
単体タスク例をもとにＨ８Ｓ／２６５５の各周辺機能の動作例を説明したものです。

（２）応用編
組み合わせタスク例をもとにＨ８Ｓ／２６５５の各周辺機能を組み合わせたときの動作例を説明したものです。

１．１　基礎編構成

基礎編は図１．２に示す構成で周辺機能を単体で使用する方法について説明しています。

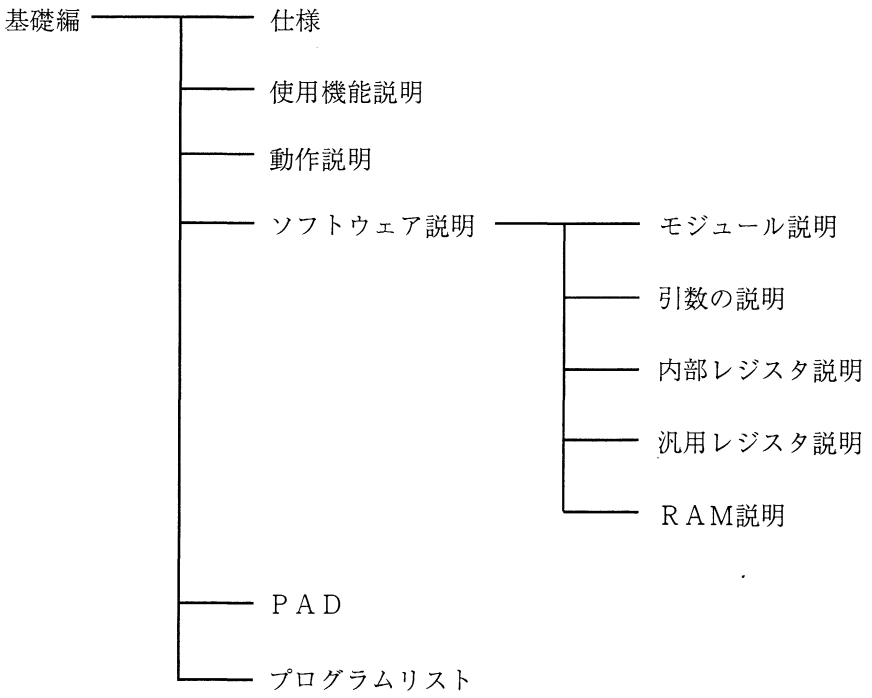


図１．２　基礎編の構成

（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）使用機能説明
タスク例で使用する周辺機能の特長および周辺機能の割り付けについて説明しています。

（3）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説明しています。

（4）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）引数の説明
モジュールを実行する際に必要な入力引数と、実行後の出力引数について説明しています。

（c）内部レジスタ説明
モジュールで設定する周辺機能の内部レジスタ（タイマコントロールレジスタ、シリアルモードレジスタ等）について説明しています。

（d）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。

（5）ＰＡＤ
タスク例を実行するソフトウェアについてＰＡＤを使用し説明しています。

（6）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。

１．２　応用編構成

応用編は図１．３に示す構成で周辺機能を組み合わせて使用する方法について説明しています。

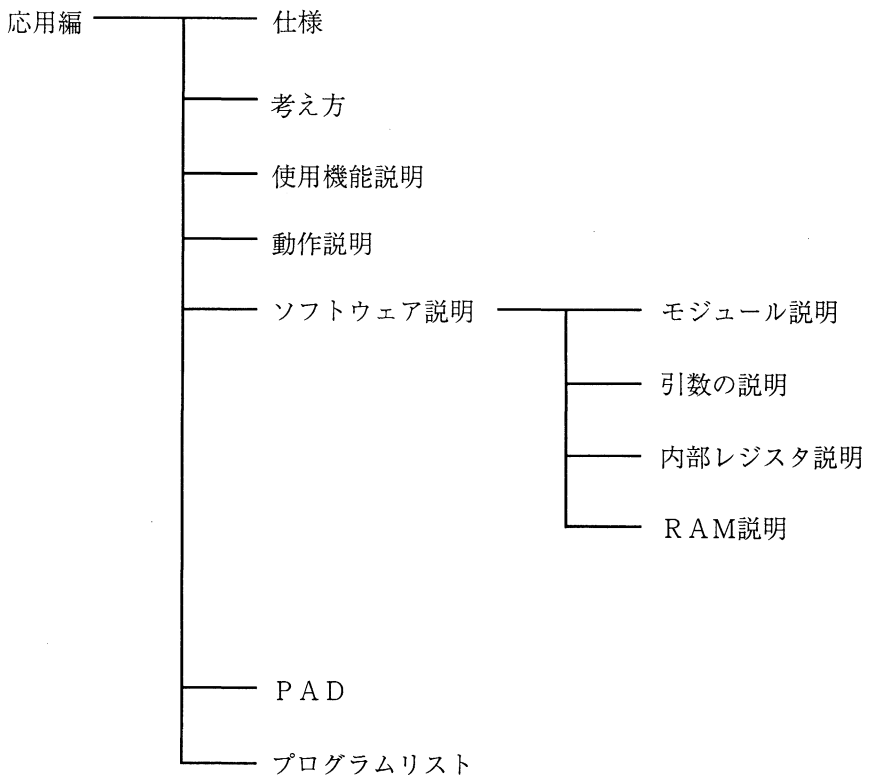


図１．３　応用編の構成

（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）考え方
タスク例のシステムを実現するための方法を説明しています。

（3）使用機能説明
タスク例で使用する周辺機能の特長および周辺機能の割り付けについて説明しています。

（4）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説明しています。

（5）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）引数の説明
モジュールを実行する際に必要な入力引数と、実行後の出力引数について説明しています。

（c）内部レジスタ説明
モジュールで設定する周辺機能の内部レジスタ（タイマコントロールレジスタ、シリアルモードレジスタ等）について説明しています。

（d）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。

（6）ＰＡＤ
タスク例を実行するソフトウェアについてＰＡＤを使用し説明しています。

（7）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。

2．各タスクで使用する共通ファイルの説明

目　次

２．１　　ベクタテーブル定義ファイル

図２．１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２．１に示す様に割り込み処理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２．１にＴＰＵのチャネル０のコンペアマッチＡ割り込みを使用する例を示します。先頭アドレス（ＰＷＨＬ１）を外部参照とします（Ａ参照）。ＴＧＩ０Ａの位置のラベル名をＰＷＨＬ１にします（Ｂ参照）。

```
/*****************************************/
/*                                       */
/*      H8S/2600 VECTOR TABLE            */
/*                                       */
/*****************************************/
#pragma section VECT

extern void INIT( void );
extern void PWHL1  (void);
const void (*const vect_tbl[])(void) =
{
    INIT,                           /* H'000000    POWER-ON Reset          */
    INIT,                           /* H'000004    Manual Reset            */
    INIT,                           /* H'000008    (System Reserve)        */
    INIT,                           /* H'00000C    (System Reserve)        */
    INIT,                           /* H'000010    (System Reserve)        */
    INIT,                           /* H'000014    Trace                   */
    INIT,                           /* H'000018    (System Reserve)        */
    INIT,                           /* H'00001C    NMI                     */
    INIT,                           /* H'000020    TRAPA1                  */
    INIT,                           /* H'000024    TRAPA2                  */
    INIT,                           /* H'000028    TRAPA3                  */
    INIT,                           /* H'00002C    TRAPA4                  */
    INIT,                           /* H'000030    (System Reserve)        */
    INIT,                           /* H'000034    (System Reserve)        */
    INIT,                           /* H'000038    (System Reserve)        */
    INIT,                           /* H'00003C    (System Reserve)        */
    INIT,                           /* H'000040    IRQ0                    */
    INIT,                           /* H'000044    IRQ1                    */
    INIT,                           /* H'000048    IRQ2                    */
    INIT,                           /* H'00004C    IRQ3                    */
    INIT,                           /* H'000050    IRQ4                    */
    INIT,                           /* H'000054    IRQ5                    */
    INIT,                           /* H'000058    IRQ6                    */
    INIT,                           /* H'00005C    IRQ7                    */
    INIT,                           /* H'000060    SWDTEND                 */
    INIT,                           /* H'000064    WOVI                    */
    INIT,                           /* H'000068    CMI                     */
    INIT,                           /* H'00006C    (System Reserve)        */
    INIT,                           /* H'000070    A/D                     */
    INIT,                           /* H'000074    (System Reserve)        */
    INIT,                           /* H'000078    (System Reserve)        */
    INIT,                           /* H'00007C    (System Reserve)        */
    PWHL1,                          /* H'000080    TGI0A                   */
    INIT,                           /* H'000084    TGI0B                   */
    INIT,                           /* H'000088    TGI0C                   */
    INIT,                           /* H'00008C    TGI0D                   */
    INIT,                           /* H'000090    TCI0V                   */
    INIT,                           /* H'000094    (System Reserve)        */
    INIT,                           /* H'000098    (System Reserve)        */
    INIT,                           /* H'00009C    (System Reserve)        */
```

A　ラベル名”PWHL1”を外部参照します。

B　ラベル名”PWHL1”を記述します

図２．１　ベクタテーブル定義ファイル（１）

```
    INIT,                   /* H'0000A0    TGI1A                */
    INIT,                   /* H'0000A4    TGI1B                */
    INIT,                   /* H'0000A8    TCI1V                */
    INIT,                   /* H'0000AC    TCI1U                */
    INIT,                   /* H'0000B0    TGI2A                */
    INIT,                   /* H'0000B4    TGI2B                */
    INIT,                   /* H'0000B8    TCI2V                */
    INIT,                   /* H'0000BC    TCI2U                */
    INIT,                   /* H'0000C0    TGI3A                */
    INIT,                   /* H'0000C4    TGI3B                */
    INIT,                   /* H'0000C8    TGI3C                */
    INIT,                   /* H'0000CC    TGI3D                */
    INIT,                   /* H'0000D0    TCI3V                */
    INIT,                   /* H'0000D4    (System Reserve)     */
    INIT,                   /* H'0000D8    (System Reserve)     */
    INIT,                   /* H'0000DC    (System Reserve)     */
    INIT,                   /* H'0000E0    TGI4A                */
    INIT,                   /* H'0000E4    TGI4B                */
    INIT,                   /* H'0000E8    TCI4V                */
    INIT,                   /* H'0000EC    TCI4U                */
    INIT,                   /* H'0000F0    TGI5A                */
    INIT,                   /* H'0000F4    TGI5B                */
    INIT,                   /* H'0000F8    TCI5V                */
    INIT,                   /* H'0000FC    TCI5U                */
    INIT,                   /* H'000100    CMIA0                */
    INIT,                   /* H'000104    CMIB0                */
    INIT,                   /* H'000108    OVI0                 */
    INIT,                   /* H'00010C    (System Reserve)     */
    INIT,                   /* H'000110    CMIA1                */
    INIT,                   /* H'000114    CMIB1                */
    INIT,                   /* H'000118    OVI1                 */
    INIT,                   /* H'00011C    (System Reserve)     */
    INIT,                   /* H'000120    DEND0A               */
    INIT,                   /* H'000124    DEND0B               */
    INIT,                   /* H'000128    DEND1A               */
    INIT,                   /* H'00012C    DEND1B               */
    INIT,                   /* H'000130    (System Reserve)     */
    INIT,                   /* H'000134    (System Reserve)     */
    INIT,                   /* H'000138    (System Reserve)     */
    INIT,                   /* H'00013C    (System Reserve)     */
    INIT,                   /* H'000140    ERI0                 */
    INIT,                   /* H'000144    RXI0                 */
    INIT,                   /* H'000148    TXI0                 */
    INIT,                   /* H'00014C    TEI0                 */
    INIT,                   /* H'000150    ERI1                 */
    INIT,                   /* H'000154    RXI1                 */
    INIT,                   /* H'000158    TXI1                 */
    INIT,                   /* H'00015C    TEI1                 */
    INIT,                   /* H'000160    ERI2                 */
    INIT,                   /* H'000164    RXI2                 */
    INIT,                   /* H'000168    TXI2                 */
    INIT,                   /* H'00016C    TEI2                 */
};
#pragma section
```

図２．１　ベクタテーブル定義ファイル（２）

2．2　レジスタ定義ファイル

付録にレジスタ定義ファイルを示します。付録に示すように各レジスタのビットは、ビットフィールドを宣言しているためビットアクセスが可能です。

2．3　スタック初期化ファイル

図2．2にスタック初期化ファイルを示します。スタックの初期化はC言語で記述できません。このためアセンブリ言語をC言語に埋め込み、スタックの初期化後、各タスクを呼び出すようにしてあります。

Cｺﾝﾊﾟｲﾗ(CH38.EXE)は、ｱｾﾝﾌﾞﾗが埋め込まれると直接ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙを生成することができません。このため、ｺｰﾄﾞｵﾌﾟｼｮﾝで、副ﾌｧｲﾙ名．SRCというｱｾﾝﾌﾞﾗ展開ﾌｧｲﾙを生成し、このﾌｧｲﾙをｱｾﾝﾌﾞﾗ(ASM38.EXE)にてｱｾﾝﾌﾞﾙしてｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙを生成するようにしてください。

CH38.EXEのｱｾﾝﾌﾞﾗ展開ﾌｧｲﾙを生成するためのｺｰﾄﾞｵﾌﾟｼｮﾝの指定は、-c=a としますが、詳細はｺﾝﾊﾟｲﾗのﾏﾆｭｱﾙを参照してください。

```
#include <machine.h>
#pragma noregsave(INIT)

void PWHLMN(void);
void _INITSCT(void);

void INIT(void)
#pragma asm
        mov.l   #h'fffc00,er7
#pragma endasm
{
        _INITSCT();
        PWHLMN();
        sleep();
}
```

図2．2　スタック初期化ファイル

2．4　ファイルのリンク

図2．3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、スタック初期化ファイル及び各タスクのリンクは、図2．3に示すサブミットファイルの情報に従いリンケージします。

```
input   vec3,init3,ap3 ←―― ﾍﾞｸﾀ定義ﾌｧｲﾙ(vec3.obj)、ｽﾀｯｸ初期化ﾌｧｲﾙ(init3.obj)、ﾀｽｸﾌｧｲﾙ(ap3.obj)のｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙをﾘﾝｸ対象とする。
library c:¥ch38¥lib¥c8s26a ←― H8S/2600ｱﾄﾞﾊﾞｽﾄﾓｰﾄﾞ用ﾗｲﾌﾞﾗﾘ(c8s26a.lib)の指定
output  ap3 ←―――――――― ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名の指定(ap3.absで出力される)
print   ap3 ←―――――――― ﾏｯﾌﾟﾌｧｲﾙ名の指定(ap3.mapで出力される)
DEBUG ←―――――――――― ﾃﾞﾊﾞｯｸﾞｵﾌﾟｼｮﾝの指定
start   VECT(0),P,C,D(2000) ← 開始ｱﾄﾞﾚｽの指定(本例ではﾍﾞｸﾀをH'0000。ﾌﾟﾛｸﾞﾗﾑ(P)、定数(C)、ﾃﾞｰﾀ(D)の順でH'2000番地から配置する。)
rom     (D,R)
form    a
exit
```

図2．3　サブミットファイル

3. 基礎編

目　次

３．１　　パルス出力

パルス出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	16ビットタイマパルスユニット（ＴＰＵ）

仕様

（１）図１に示すようにＲＡＭに設定された周期データをもとにデューティ５０％のパルスを出力します。
（２）２０ＭＨｚで動作時、出力するパルスの周期は１００ｎｓから３．２７ｍｓの間で任意に設定できます。

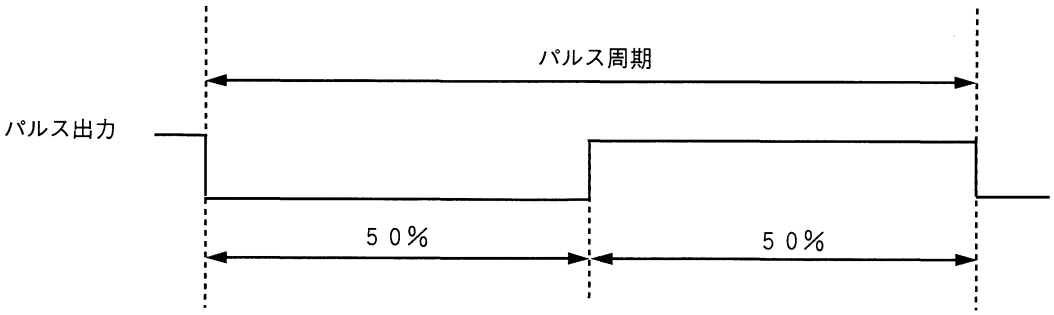


図１　パルス出力例

使用機能説明

（１）本タスク例ではＴＰＵ０を使用してデューティ５０％のパルスを出力します。
（ａ）図２に本タスク例で使用するＴＰＵ０のブロック図を示します。
ＴＰＵ０では以下の機能を使用します。
・ソフトウェアを介さずハードウェアで自動的にパルスを出力する機能。（アウトプットコンペア）
・コンペアマッチ時、カウンタをクリアする機能。（カウンタクリア）
・コンペアマッチが起きるごとに出力が反転する機能。（トグル出力）

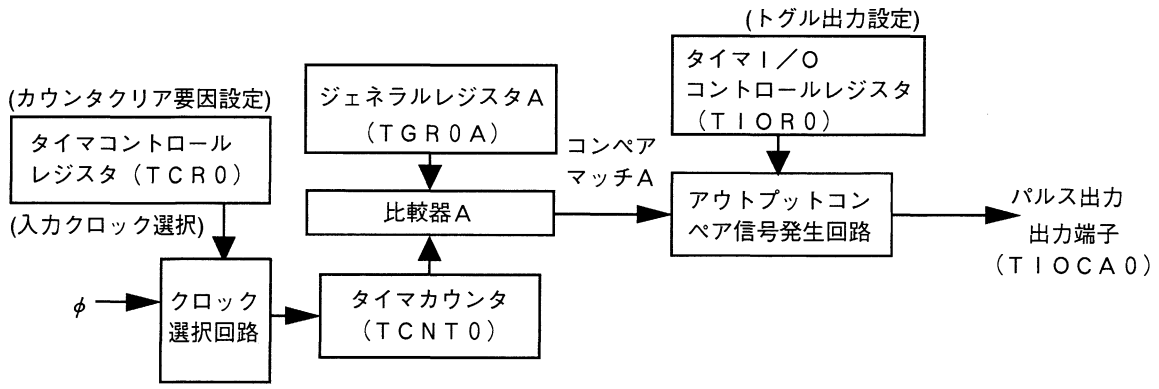


図２　パルス出力ブロック図

パルス出力	MCU	H8S／2655	使用機能	16ビットタイマパルスユニット（TPU）

使用機能説明

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、パルスを出力します。

表1　H8S／2655機能割り付け

H8S/2655機能	機能
TCR0	TCNT0に入力するクロック、およびカウンタクリア要因を選択する
TIOCA0	パルスを出力する
TIOR0	パルスの出力レベルを設定する
TGR0A	パルスの1/2周期を設定する

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理によりパルスを出力します。

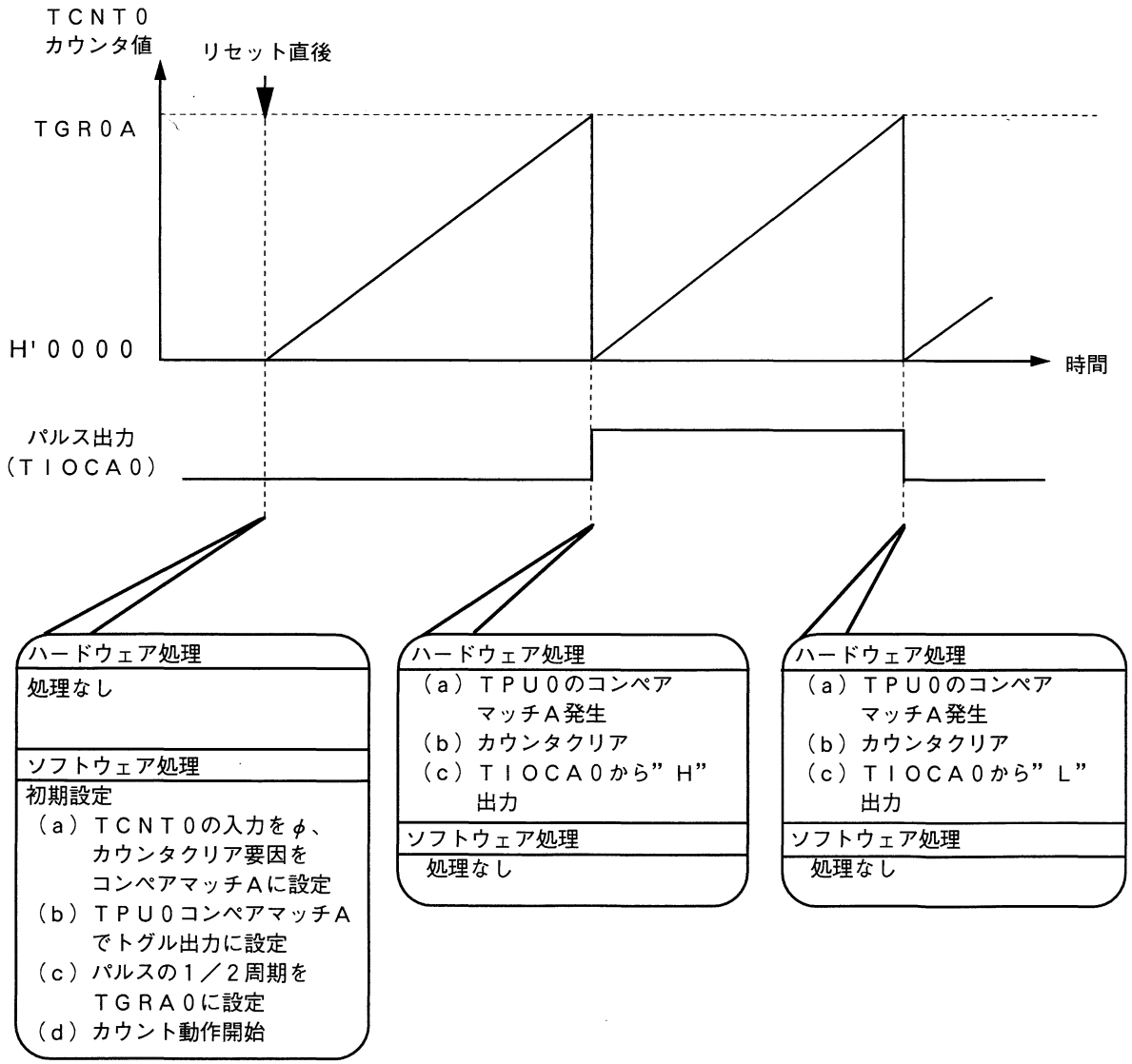


図3　パルス出力動作原理

パルス出力	MCU	H8S／2655	使用機能	16ビットタイマパルスユニット（TPU）

ソフトウェア説明

（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	poutmn	TPU、RAMの初期設定、およびパルス出力を行う

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
pul_cyc	パルスの周期に相当するタイマ値を設定する パルスの周期は以下の式にて求める ﾊﾟﾙｽ周期(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short	ﾒｲﾝﾙｰﾁﾝ	入力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
TSTR	タイマカウンタの動作／禁止を設定する	ﾒｲﾝﾙｰﾁﾝ
TCR0	TCNTに入力するクロックおよびカウンタクリア要因を設定する	ﾒｲﾝﾙｰﾁﾝ
TIOR0	コンペアマッチA発生時に、出力するパルスのレベルを設定する	ﾒｲﾝﾙｰﾁﾝ
TGR0A	出力パルスの1/2周期を設定する	ﾒｲﾝﾙｰﾁﾝ
MSTPCR	TPUのモジュールストップモードを解除する	ﾒｲﾝﾙｰﾁﾝ

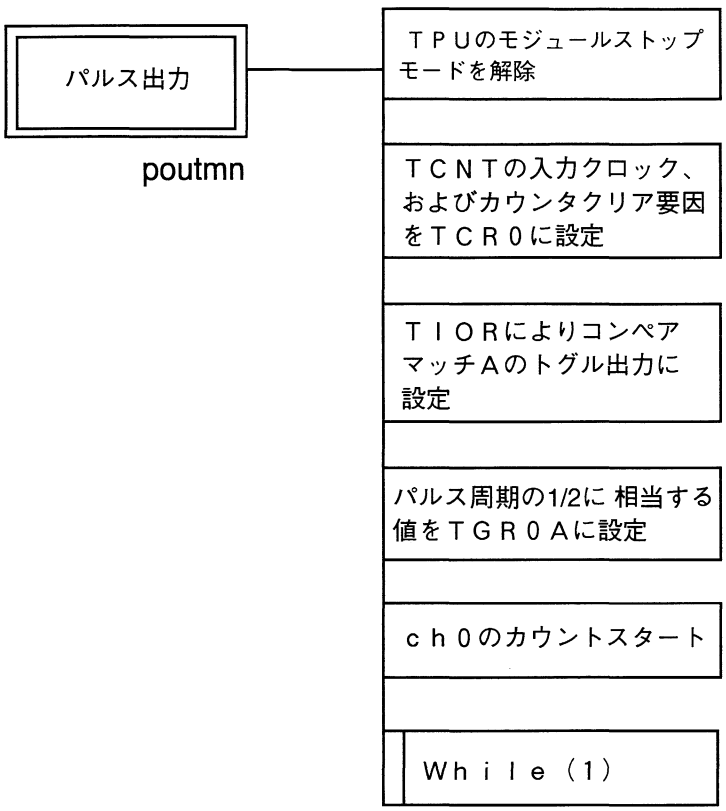
（4）使用RAM説明

本アプリケーション例では引数以外のRAMは使用していません。

パルス出力	MCU	H8S／2655	使用機能	16ビットタイマパルスユニット（TPU）

PAD

（1）メインルーチン



パルス出力	MCU	H8S／2655	使用機能	16ビットタイマパルスユニット（TPU）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/****************************************/
/*      PROTCOL                         */
/****************************************/
void poutmn(void);

/****************************************/
/*      RAM ALLOCATION                  */
/****************************************/
#define pul_cyc (*(unsigned short *)0xffec00)

/****************************************/
/*     MAIN PROGRAM : poutmn            */
/****************************************/
void poutmn(void)
{
        MSTPCR = 0x1fff;
        TPU_TCR0 = 0x20;          /* initialize TCR0 */
        TIOR0H = 0x03;            /* initialize TIOR0 */
        TGR0A = pul_cyc/2;        /* set data to TGR0A */
        TSTR = 0x01;              /* TCNT0 start */
        while(1);                 /* loop */
}
```

3．2　　2相エンコーダカウント

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

仕様

図1に示すように、外部クロック端子(TCLKA、TCLKB)に入力される2相エンコーダパルスの位相差を検出し、測定時間内にカウントアップ／カウントダウンされたカウント数をRAMに設定します。

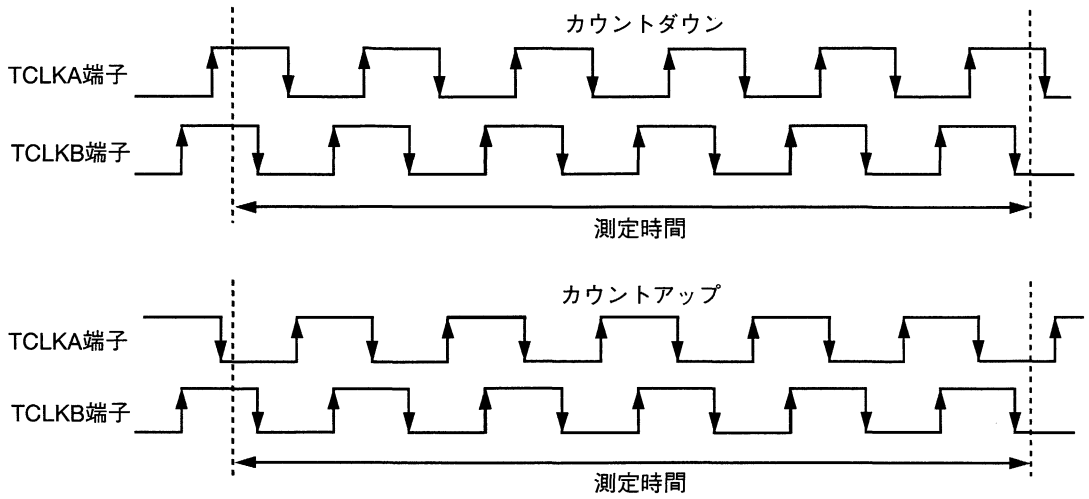


図1　2相エンコーダカウント

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

使用機能説明

（1）本タスク例ではＴＰＵ１を使用して２相エンコーダカウントを行います。図２に２相エンコーダカウントのブロック図を示します。

（a）外部クロック端子TCLKA, TCLKBに入力される２相エンコーダパルスの位相差を検出し、ＴＰＵ１のカウンタをカウントアップ／カウントダウンする機能（位相計数モード）。

（b）外部クロックで動作しているカウンタの値を他のチャネルのコンペアマッチでジェネラルレジスタに転送する機能。

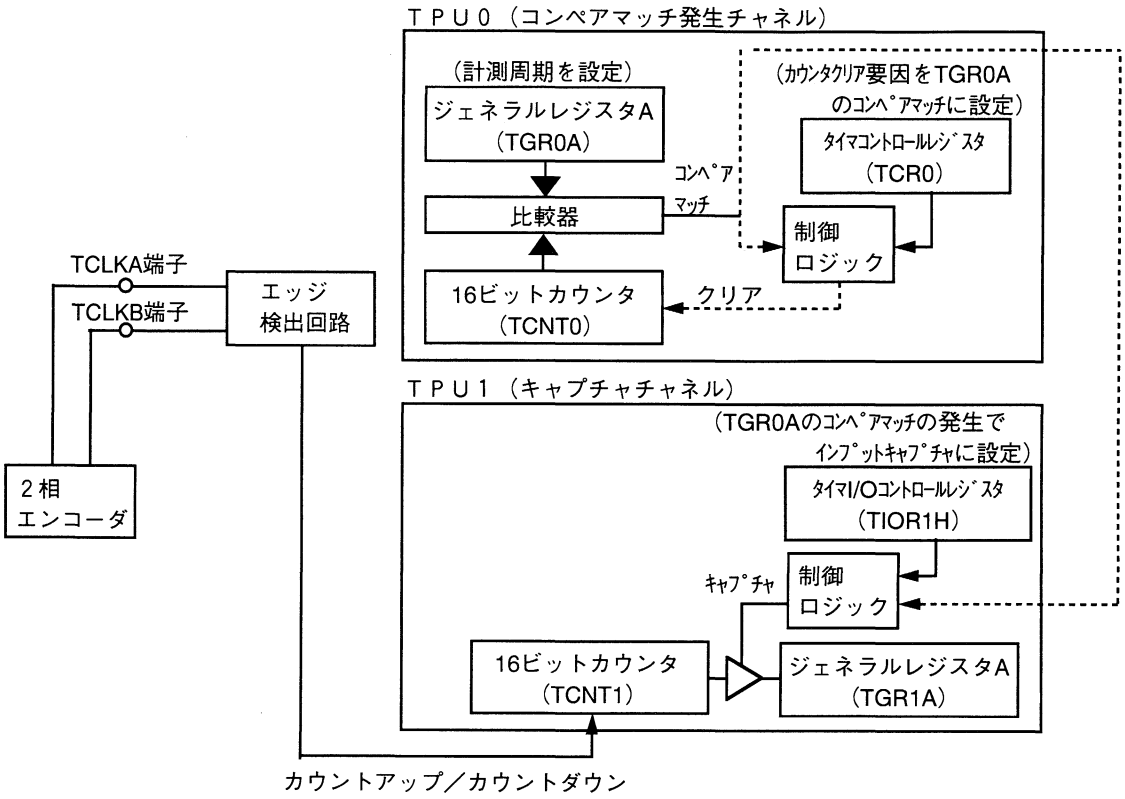


図２　２相エンコーダカウントブロック図

（2）表１に本タスク例の機能割り付けを示します。表１に示すようにＨ８Ｓ／２６５５の機能を割り付け、２相エンコーダカウントを行います。

表１　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能	機能
TCLKA、B	2相エンコーダパルスの入力端子
TCR0	カウンタクリア要因をTGR0Aのコンペアマッチに設定
TGR0A	測定時間を設定
TIOR1	TGR0Aのコンペアマッチの発生でインプットキャプチャに設定
TGR1A	インプットキャプチャＡによりカウント結果を設定

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェア処理により2相エンコーダカウント（カウントアップ）を行います。

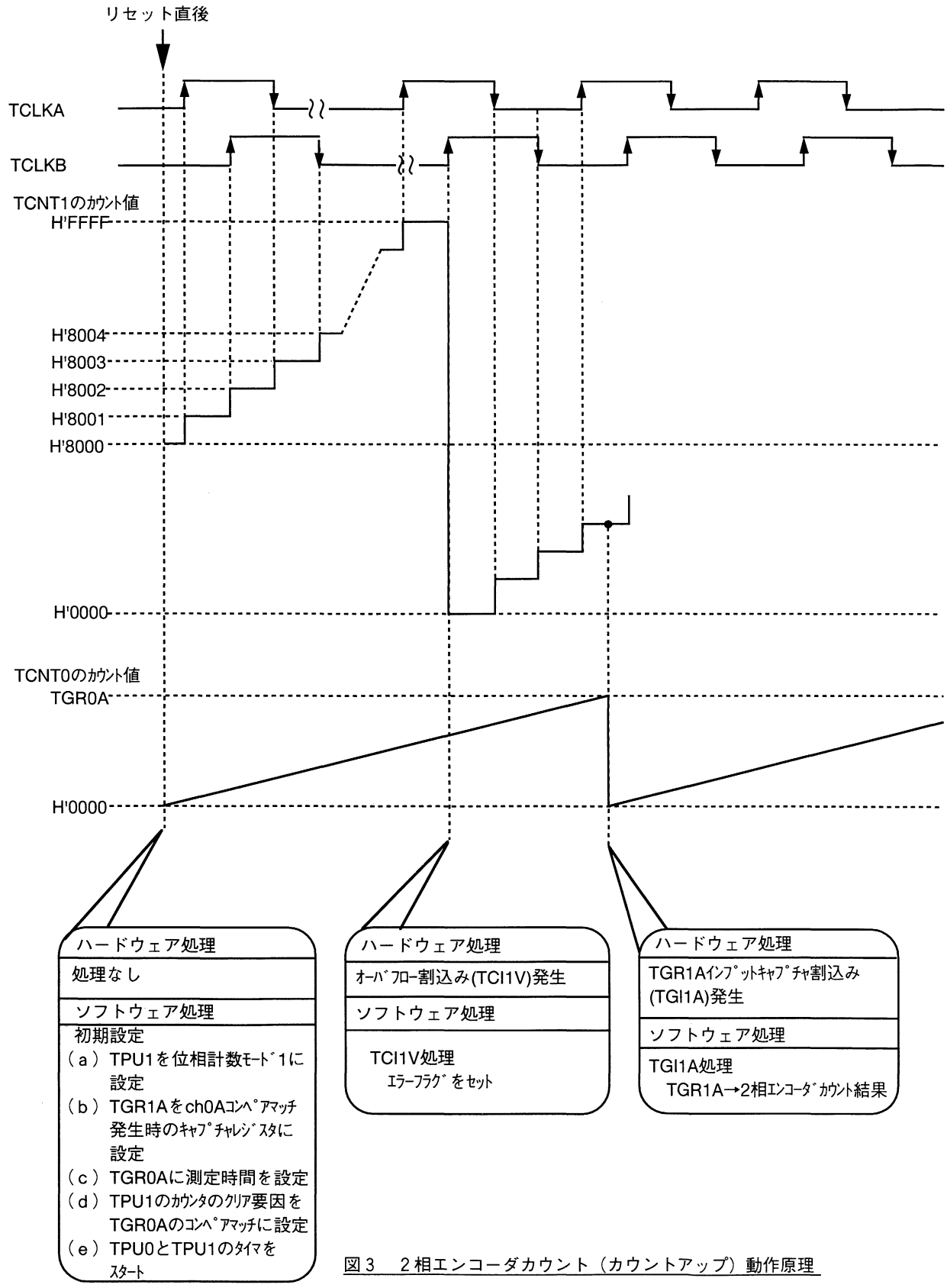


図3　2相エンコーダカウント（カウントアップ）動作原理

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

動作説明

図4に動作原理を示します。図4に示すようにH8S／2655のハードウェア処理およびソフトウェア処理により2相エンコーダカウント（カウントダウン）を行います。

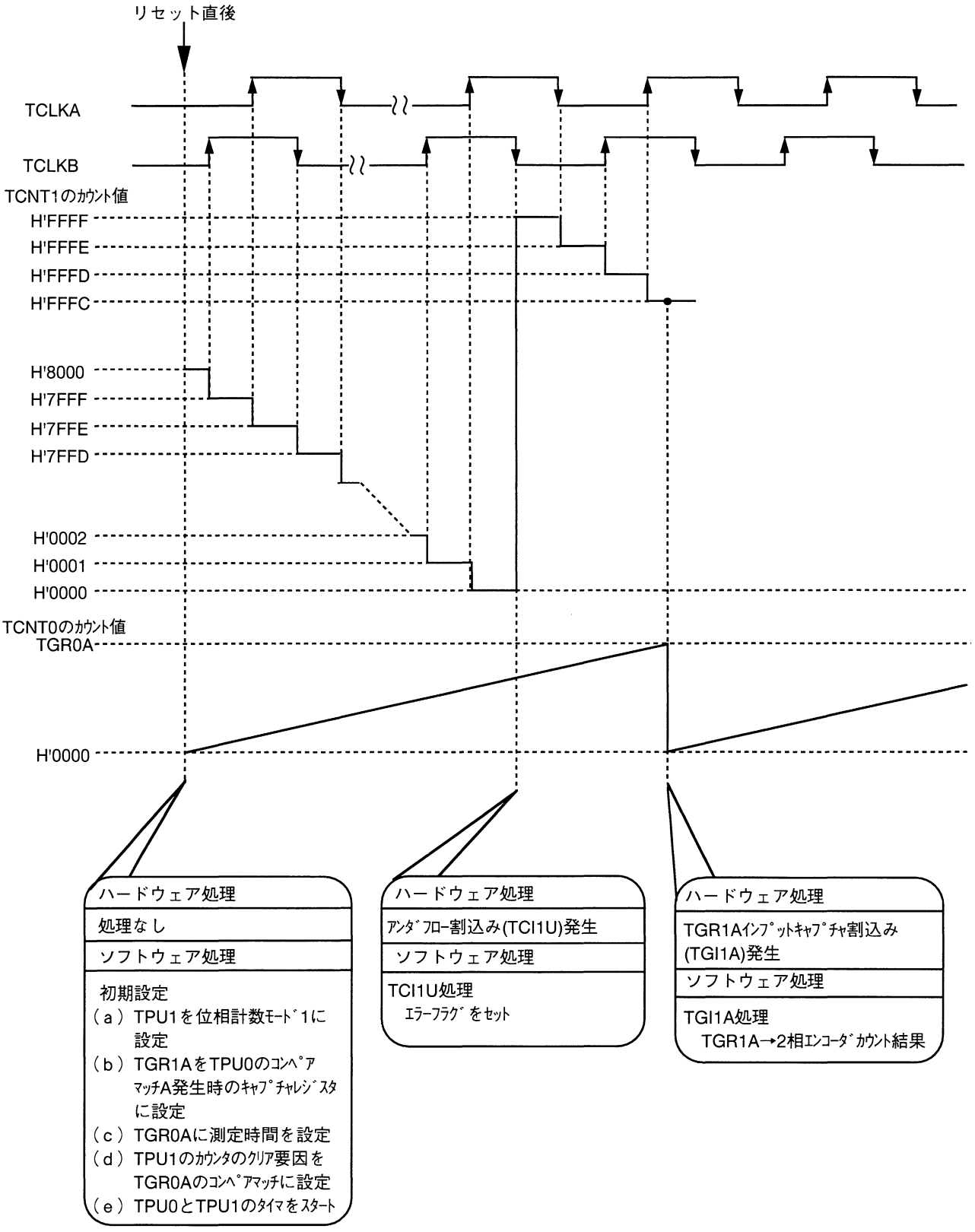


図4　2相エンコーダカウント（カウントダウン）動作原理

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

ソフトウェア説明

（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	cntmn	2相エンコーダカウントの初期設定を行う
キャプチャ割込み	ramset	カウント結果をRAMに格納する
オーバフロー検出	error1	オバーフロー発生フラグをセットする
アンダーフロー検出	error2	アンダーフロー発生フラグをセットする

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
count	測定時間内のカウント結果を設定する	unsigned short	キャプチャ割込み	出力
err_over	オーバフローの発生の有無を示す 1：オーバフロー有り　0：オーバフロー無し	unsigned char	ｵｰﾊﾞﾌﾛｰ検出	出力
err_under	アンダーフローの発生の有無を示す 1：アンダーフロー有り　　0：アンダーフロー無し	unsigned char	ｱﾝﾀﾞｰﾌﾛｰ検出	出力
cnttim	測定時間を設定する	unsigned short	メインルーチン	入力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
TSTR	TPU0、TPU1のタイマカウンタのカウントのスタート/ストップを制御する	ﾒｲﾝﾙｰﾁﾝ
TCR0	カウンタクリア要因をTGR0Aのコンペアマッチに設定する	ﾒｲﾝﾙｰﾁﾝ
TIOR0	TGR0Aをアウトプットコンペアレジスタに設定する	ﾒｲﾝﾙｰﾁﾝ
TMDR1	TPU1を位相計数モード１に設定する	ﾒｲﾝﾙｰﾁﾝ
TCR1	カウンタクリア要因をTGR0Aのコンペアマッチに設定する	ﾒｲﾝﾙｰﾁﾝ
TIOR1	TGR1AをTGR0Aのコンペアマッチ発生によるキャプチャレジスタに設定する	ﾒｲﾝﾙｰﾁﾝ
TCNT1	初期値H'8000に設定する	ﾒｲﾝﾙｰﾁﾝ
TIER1	TGFA、TCFU、TCFVビットによる割込み要求を許可する	ﾒｲﾝﾙｰﾁﾝ
TSR1	インプットキャプチャ、およびオーバフロー／アンダフロー割込みを許可する	ﾒｲﾝﾙｰﾁﾝ ｷｬﾌﾟﾁｬ割込み
MSTPCR	TPUのモジュールストップモードを解除する	ﾒｲﾝﾙｰﾁﾝ

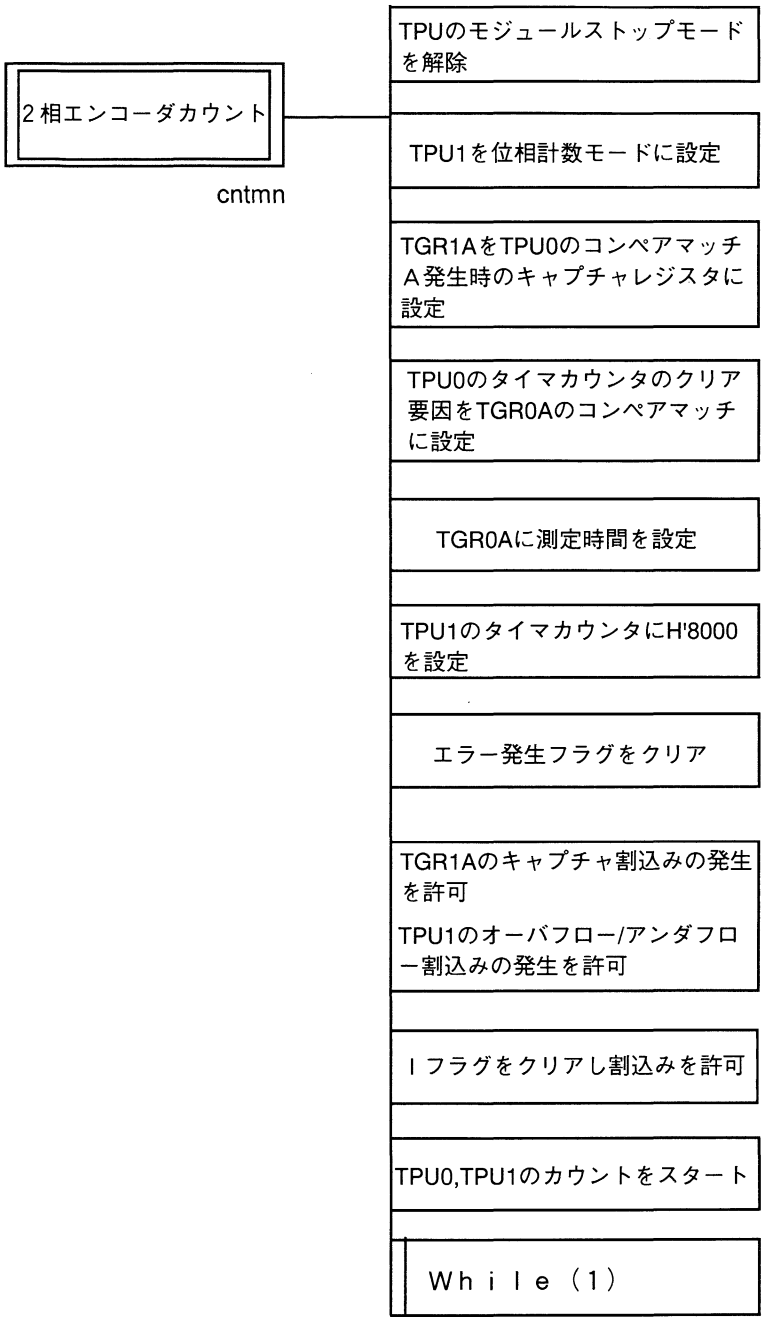
（4）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

PAD

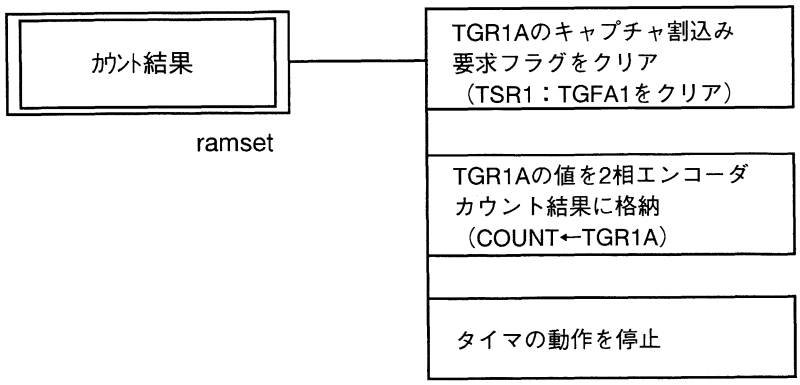
（1）メインルーチン



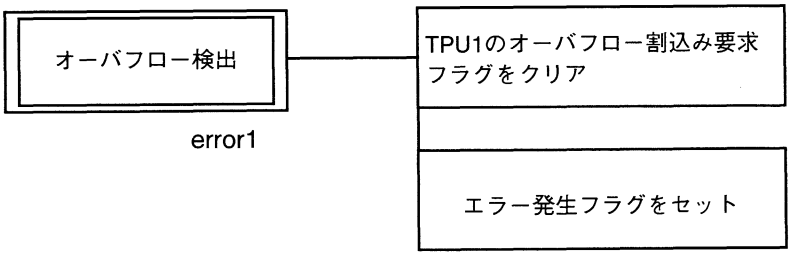
2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

PAD

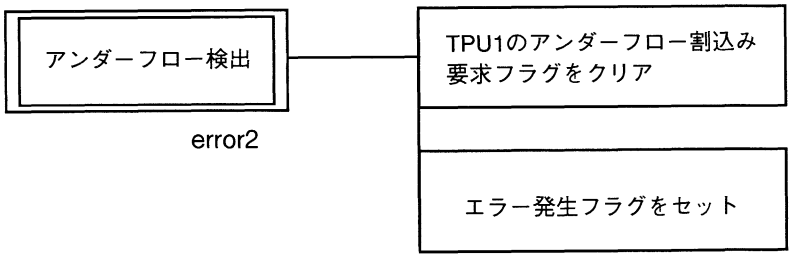
（2）キャプチャ割込み



（3）オーバフロー検出



（4）アンダーフロー検出



2相エンコーダカウント	MCU	H8S／2655	使用機能	TPU

プログラムリスト

```
#include <machine.h>
#include "..¥h8sapn¥h8s.h"
/*****************************************/
/*            PROTCOL                    */
/*****************************************/
void cntmn(void);
/*****************************************/
/*         RAM ALLOCATION                */
/*****************************************/
#define count (*(unsigned short *)0xffec00)
#define cnttime (*(unsigned short *)0xffec02)
volatile struct ERROR{
        char    over;
        char    under;
};
#define err (*(struct ERROR *)0xffec04)
/*****************************************/
/*         MAIN PROGRAM : cntmnd         */
/*****************************************/
void cntmn(void)
{
        MSTPCR = 0x1fff;        /* Disable module(TPU) stop mode*/
        TMDR1 = 0xc4;           /* Initialize TMDR1 ch1:phase counting model */
        TIOR0H = 0;             /* Initialize TIOR0H */
        TIOR1H = 0x0c;          /* Initialize TIOR1H *
        TPU_TCR0 = 0x20;        /* Initialize TCR0 */
        TPU_TCR1 = 0x20;        /* Initialize TCR1 */
        TGR0A = cnttime;        /* Set counting time */
        TPU_TCNT1 = 0x8000;     /* Counter start H'8000 */
        err.over = 0x00;        /* Over flow flag clear */
        err.under = 0x00;       /* Under flow flag clear */
        TIER1 = 0x71;           /* Enable timer interrupt */
        set_imask_ccr(0);       /* CCR Ibit clear */
        TSTR = 0x03;            /* Start TCNT0,1 */
        while(1);
}
/*****************************************/
/*           NAME : ramset               */
/*****************************************/
#pragma interrupt (ramset)
void ramset(void)
{
        TSR1_BP.TGFA1 = 0;      /* Clear TGFA1 request */
        count = TGR1A;          /* Store count data */
        TSTR = 0x00;            /* Stop counter */
}
/*****************************************/
/*           NAME : error1               */
/*****************************************/
#pragma interrupt (error1)
void error1(void)
{
        TSR1_BP.TCFV1 = 0;      /* Clear TCFV1 request */
        err.over = 0x01;
}
/*****************************************/
/*           NAME : error2               */
/*****************************************/
#pragma interrupt (error2)
void error2(void)
 {
        TSR1_BP.TCFU1 = 0;      /* Clear TCFU1 request */
        err.under = 0x01;
}
```

3．3　パルスのＨｉｇｈおよびＬｏｗ幅測定

パルスのＨｉｇｈおよびＬｏｗ幅測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ

仕様

（1）図1に示すように、パルスのＨｉｇｈ幅およびＬｏｗ幅の時間を測定し、結果をＲＡＭに設定します。
（2）２０ＭＨｚで動作時、パルスのＨｉｇｈ幅およびＬｏｗ幅は、１．９μｓから３．２７ｍｓまで５０ｎｓ単位で測定可能です。

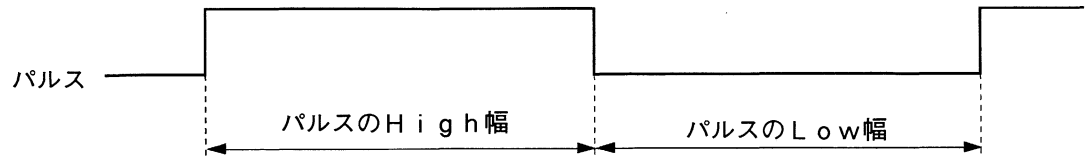


図1　パルス幅測定タイミング

使用機能説明

（1）本タスク例では、ＴＰＵ０を使用してパルスのＨｉｇｈおよびＬｏｗ幅を測定します。
（a）図2にＴＰＵ０のブロック図を示します。本タスクでは、以下の機能を使用します。
・パルスの立ち上がりおよび立ち下がりエッジの検出を行い、そのときのタイマ値を内部レジスタに設定する機能。（インプットキャプチャ)
・インプットキャプチャ発生時タイマカウンタをクリアする機能。
・パルスの立ち上がりおよび立ち下がりエッジ検出時、割り込み処理を起動する機能。

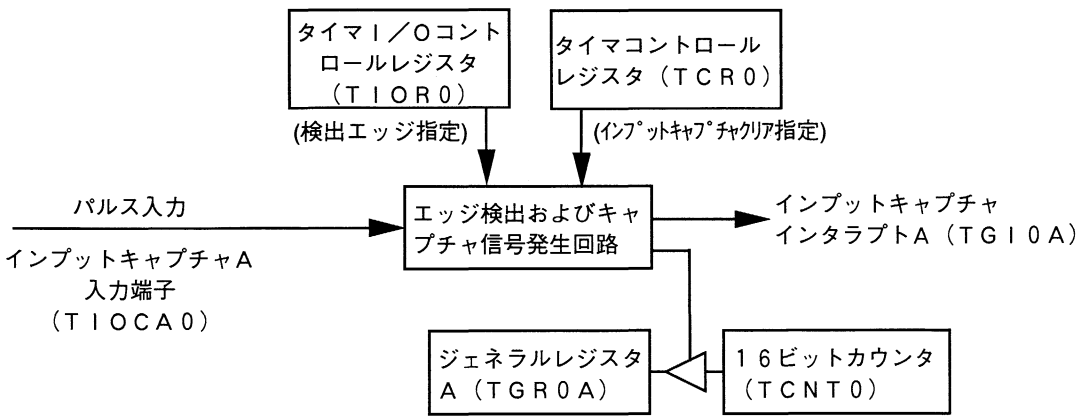


図2　ﾊﾟﾙｽのHighおよびLow幅測定ブロック図

パルスのＨｉｇｈおよびＬｏｗ幅測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ

使用機能説明

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにＨ８Ｓ／２６５５の機能を割り付け、パルスのＨｉｇｈおよびＬｏｗ幅を測定します。

表1　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能	機能
ＴＣＲ０	カウンタクリア要因を選択する
ＴＩＯＲ０	インプットキャプチャ信号の入力エッジを選択する
ＴＩＯＣＡ０	測定するパルスを入力する
ＴＧＲ０Ａ	パルスの立ち上がりおよび立ち下がり時のカウンタ値を検出する
ＴＧＩ０Ａ	パルスの立ち上がりおよび立ち下がりにより、パルスのＨｉｇｈおよびＬｏｗ幅測定を起動する

動作説明

図3に動作原理を示します。図3に示すようにＨ８Ｓ／２６５５のハードウェア処理およびソフトウェアの処理によりパルスのＨｉｇｈおよびＬｏｗ幅を測定します。

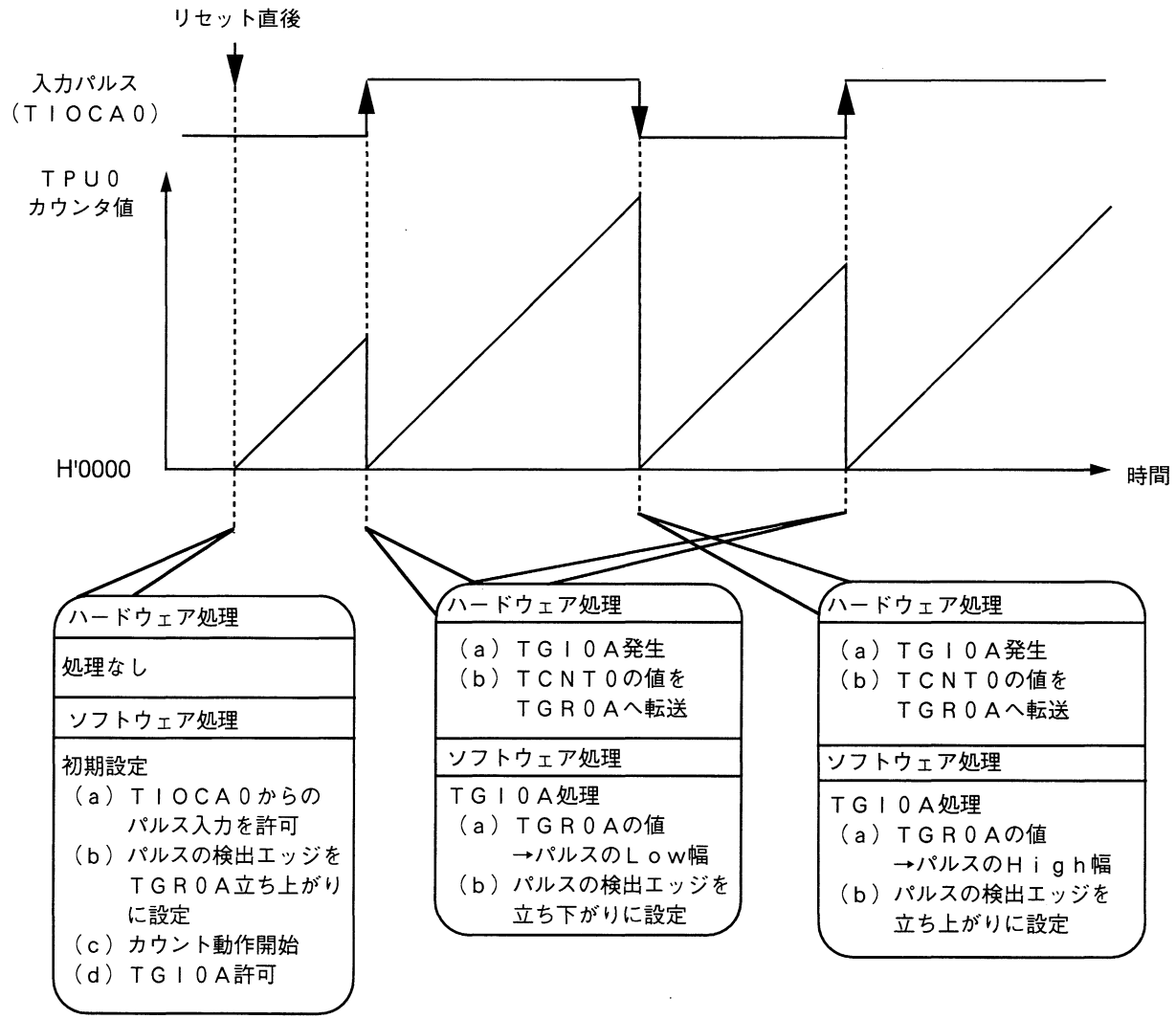


図3　パルス幅測定動作原理

パルスのＨｉｇｈおよびＬｏｗ幅測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ

ソフトウェア説明

（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	PWHLMN	ＴＰＵおよびＲＡＭの初期設定を行う
パルスのＨｉｇｈおよびＬｏｗ幅測定	PWHL1	ＴＧＩ０Ａにより起動し、ＴＧＲ０Ａの値からパルスのＨｉｇｈ幅およびＬｏｗ幅を測定し、ＲＡＭに設定する

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
ｐｗｈ_ｈｄａｔａ	パルスのＨｉｇｈ幅に相当するタイマ値を設定する パルスのＨｉｇｈ幅は以下の式にて求まる ﾊﾟﾙｽのHigh幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short	ﾊﾟﾙｽのHighおよびLow幅測定	出力
ｐｗｈ_ｌｄａｔａ	パルスのＬｏｗ幅に相当するタイマ値を設定する パルスのＬｏｗ幅は以下の式にて求まる ﾊﾟﾙｽのLow幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short		

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
ＴＳＴＲ	タイマカウンタの動作／禁止を設定する	ﾒｲﾝﾙｰﾁﾝ
ＴＣＲ０	ＴＣＮＴのカウンタクロックの選択、およびカウンタクリア要因をインプットキャプチャＡに設定するをする	ﾒｲﾝﾙｰﾁﾝ、ﾊﾟﾙｽのHighおよびLow幅測定
ＴＩＯＲ０	パルスの立ち上がりおよび立ち下がり検出によりＴＣＮＴからＴＧＲ０Ａへの転送を行うように設定する	ﾒｲﾝﾙｰﾁﾝ
ＴＩＥＲ０	ＴＧＩ０Ａによる割り込みを許可する	ﾒｲﾝﾙｰﾁﾝ
ＴＧＲ０Ａ	パルスの立ち上がりおよび立ち下がり時のＴＣＮＴの値が設定され、この値によりパルスの周期を求める	ﾊﾟﾙｽのHighおよびLow幅測定
ＴＳＲ０	インプットキャプチャＡの発生を示す	ﾊﾟﾙｽのHighおよびLow幅測定
ＭＳＴＰＣＲ	ＴＰＵのモジュールストップモードを解除する	ﾒｲﾝﾙｰﾁﾝ

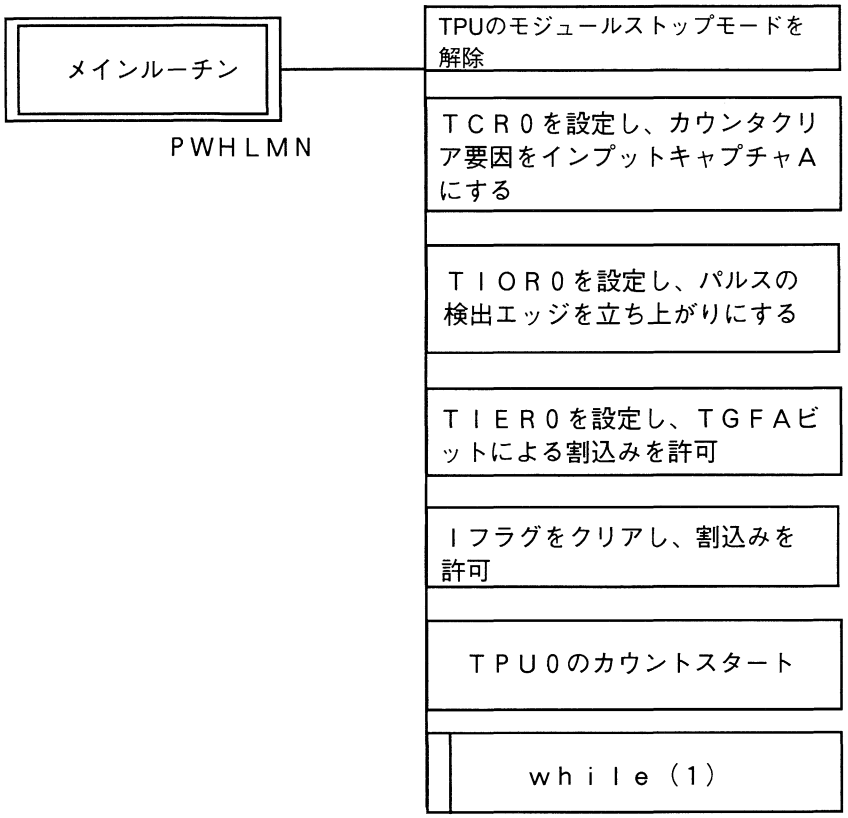
（4）使用ＲＡＭ説明

本タスク例では引数以外のＲＡＭは使用していません。

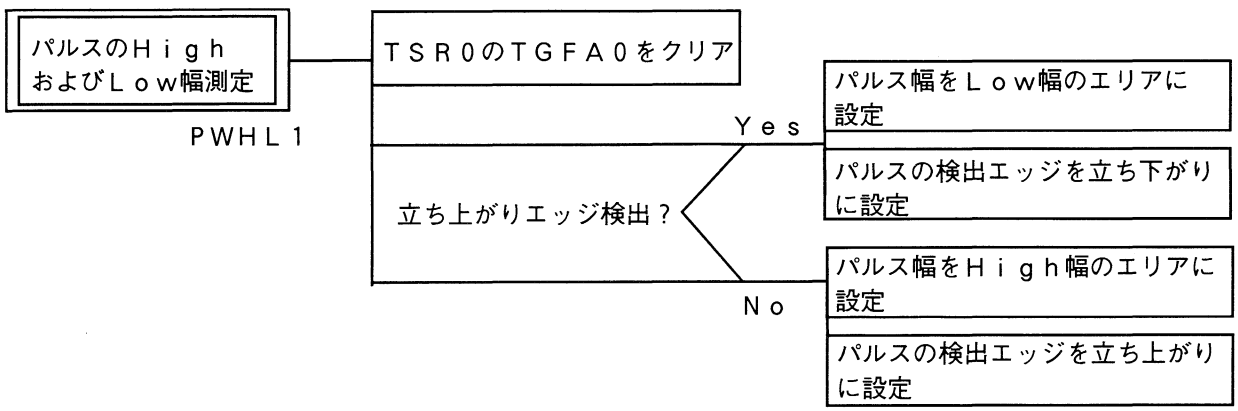
パルスのＨｉｇｈおよびＬｏｗ幅測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ

ＰＡＤ

（１）メインルーチン



（２）パルスのＨｉｇｈおよびＬｏｗ幅測定



パルスのＨｉｇｈおよびＬｏｗ幅測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ

プログラムリスト

```
#include <machine.h>
#include <H8S.H>
/**********************************/
/*           PROTOCOL             */
/**********************************/
void PWHLMN(void);
#pragma interrupt (PWHL1)
/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/

# define pwh_ldata    (*(unsigned short * )0xffec00)      /* Pulse high width time */
# define pwh_hdata    (*(unsigned short * )0xffec02)      /* Pulse low width time */

/**********************************/
/*     MAIN PROGRAM: PWHLMN       */
/**********************************/
void PWHLMN(void)
{
        MSTPCR = 0xdfff;                /* Disable module(TPU) stop mode*/
        TPU_TCR0 = 0x20;                /* Initialize TCR0 */
        TIOR0H = 0x08;                  /* Initialize TIOR0H */
        TIER0 = 0x41;                   /* Initialize TIER0 */
        set_imask_ccr(0);               /* Enable interrupt */
        TSTR  = 0x01;                   /* Start TCNT0 */
        while(1);                       /* Loop */
}

/**********************************/
/*   INTERRUPT PROGRAM: PWHL1     */
/**********************************/
void PWHL1(void)
{
        TSR0_BP.TGFA0 = 0;              /* Clear TGFA0 request */
        if(TIOR0H == 0x08)              /* Edge is "high"? */
                {                       /* Yes */
                pwh_ldata = TGR0A;
                TIOR0H = 0x09;          /* Set TGR0A captures falling edge of input */
                }
        else{                           /* No */
                pwh_hdata = TGR0A;
                TIOR0H = 0x08;          /* Set TGR0A captures rising edge of input */
                }
}
```

3．4　　長周期パルス出力

長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

仕様

（1）図1に示すように32ビットカウンタ動作を行い、パルスのＨｉｇｈ幅を変化させデューティ変化できる長周期パルスを出力します。
（2）デューティは0％～100％まで、1／65535の分解能で設定できます。
（3）20MHz動作時、パルスの周期は6．55ｍｓから214．7ｓまで3．27ｍｓ単位で設定可能です。

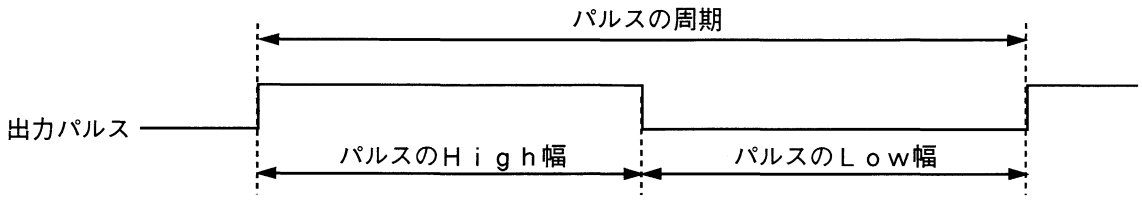


図1　長周期パルス出力例

長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

使用機能説明

（1）本タスク例ではＴＰＵ１とＴＰＵ２の１６ビットカウンタを接続して３２ビットカウンタ動作を行い、ＴＰＵ１から長周期パルスを出力します。図２に本タスク例で使用するＴＰＵのブロック図を示します。本タスク例では以下の機能を使用します。

・２チャネルの16ビットカウンタを接続し32ビットカウンタとして動作させる機能（ｶｽｹｰﾄﾞ接続動作）
・ソフトウェアを介さずハードウェアで自動的にパルスを出力する機能。（ｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱ）
・ＴＧＲ１ＡとＴＧＲ１Ｂをペアで使用しＰＷＭ出力を生成。（PWMﾓｰﾄﾞ1）

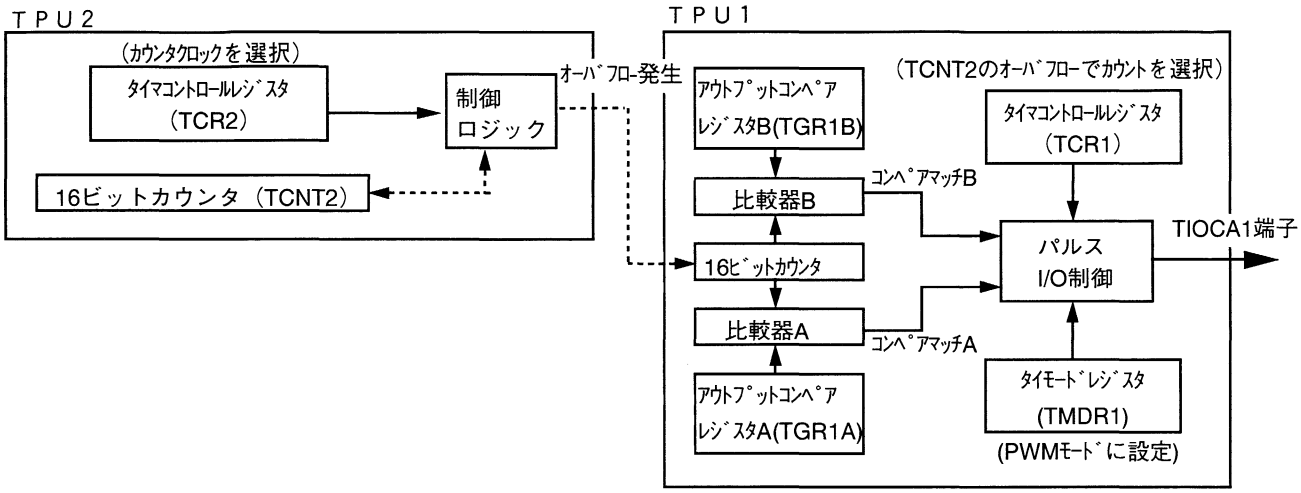


図２　長周期パルス出力ブロック図

（2）表１に本タスク例の機能割り付けを示します。表１に示すようにＨ８Ｓ／２６５５の機能を割り付け、３２ビット動作をするタイマカウンタを生成します。

表１　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能		機能
ＴＰＵ１	ＴＭＤＲ１	ＰＷＭモード１を選択する
	ＴＣＲ１	ＴＣＮＴ１に入力するクロック、およびカウンタクリア要因を設定する
	ＴＣＮＴ１	３２ビットカウンタの上位１６ビット
	ＴＧＲ１Ａ	パルスのLow幅設定
	ＴＧＲ１Ｂ	パルスのHigh幅設定
	ＴＩＯＣＡ１	パルスを出力する
ＴＰＵ２	ＴＣＲ２	ＴＣＮＴ２に入力するクロックを選択する
	ＴＣＮＴ２	３２ビットカウンタの下位１６ビット

長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

動作説明

（1）図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理により長周期パルスを出力します。

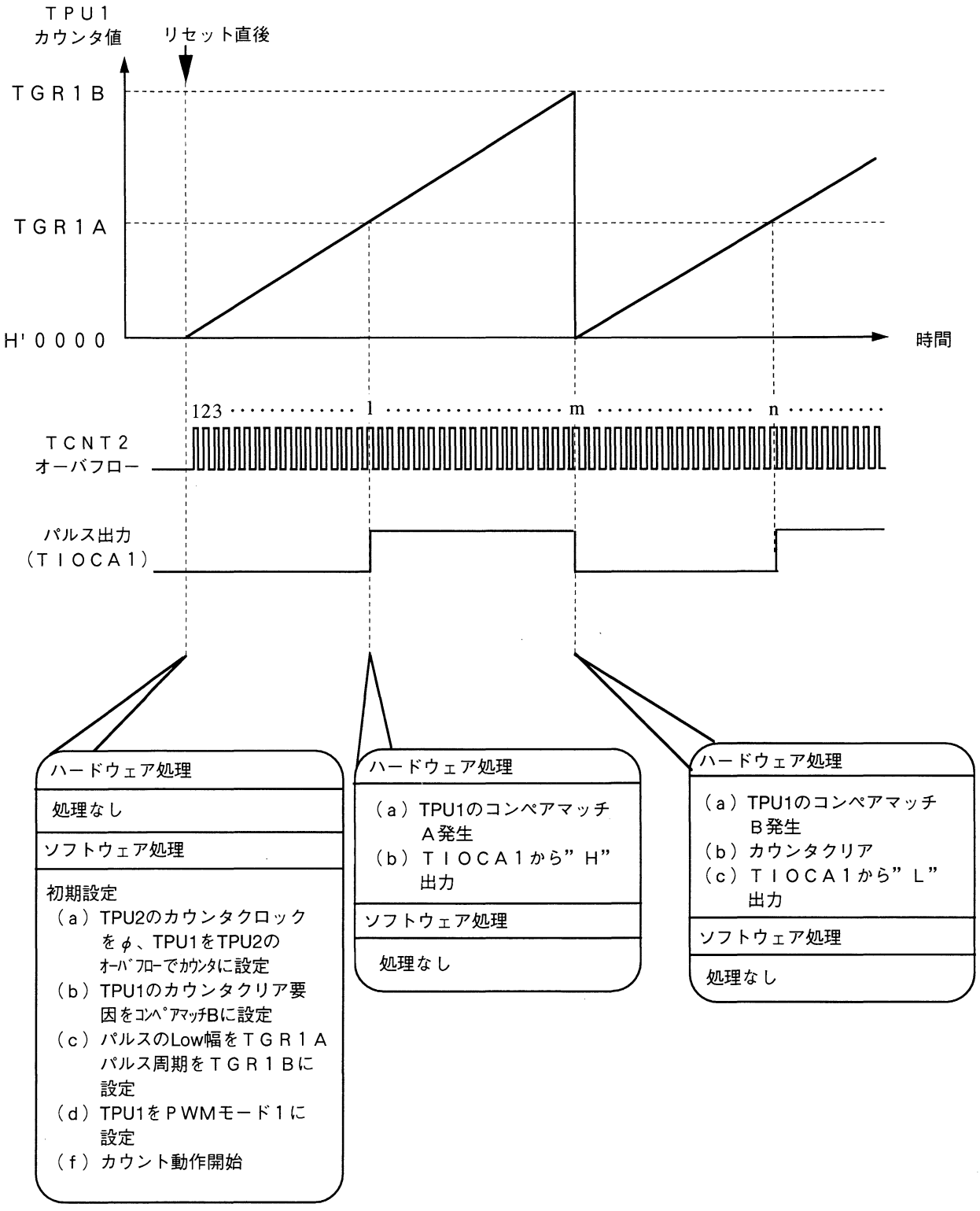


図3　長周期パルス出力動作原理

長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

ソフトウェア説明

（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	LPULMN	TPU1、TPU2のカウンタを使用して32ビットカウンタ動作を行い、長周期パルスを出力します

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
lpul_wid	出力パルスのＬｏｗ幅に相当するタイマ値を設定する パルスのＬｏｗ幅は以下の式にて求まる Ｌｏｗ幅(ms)=ﾀｲﾏ値×外部ｸﾛｯｸ(20MHz動作時3.27ms)	unsigned short	メインルーチン	入力
lpul_cyc	出力パルスの周期に相当するタイマ値を設定する 周期は以下の式にて求まる 周期(ms)=ﾀｲﾏ値×*外部ｸﾛｯｸ(20MHz動作時3.27ms)	unsigned short	メインルーチン	入力

*外部ｸﾛｯｸ：TCNT2のオーバフロー出力

（3）使用内部レジスタ説明

レジスタ名		機能	使用モジュール名
TPU1	TSTR	タイマカウンタの動作／禁止を設定する	ﾒｲﾝﾙｰﾁﾝ
	TMDR1	PWMモード1を選択する	ﾒｲﾝﾙｰﾁﾝ
	TCR1	TCNT1に入力するクロック、およびカウンタクリア要因を設定する	ﾒｲﾝﾙｰﾁﾝ
	TCNT1	TCNT2のｵｰﾊﾞﾌﾛｰでカウントし、32ビットカウンタ動作する	ﾒｲﾝﾙｰﾁﾝ
	TGR1A	パルスのLow幅設定	ﾒｲﾝﾙｰﾁﾝ
	TGR1B	パルスのHigh幅設定	ﾒｲﾝﾙｰﾁﾝ
	TIOCA1	パルスを出力する	ﾒｲﾝﾙｰﾁﾝ
TPU2	TCR2	TCNT2に入力するクロックを選択する	ﾒｲﾝﾙｰﾁﾝ
	TCNT2	16ビットフリーランカウンタ	ﾒｲﾝﾙｰﾁﾝ

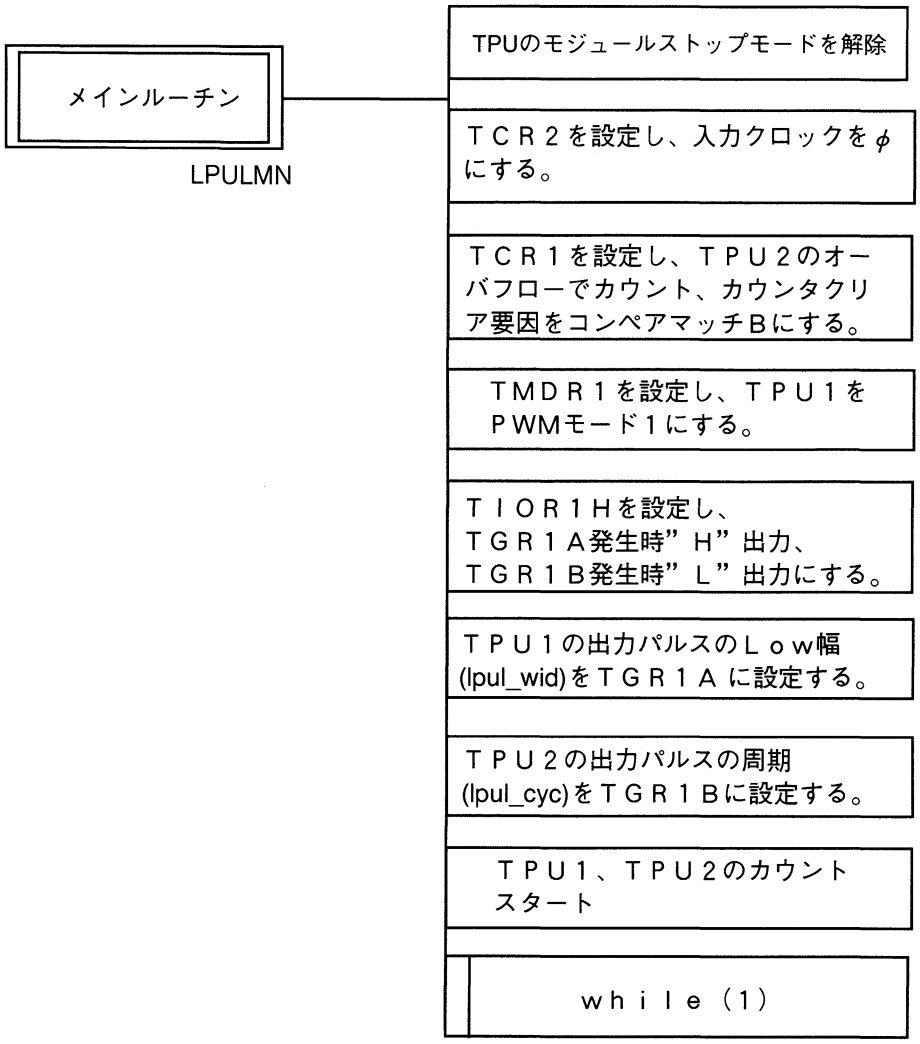
（4）使用ＲＡＭ説明

本タスク例では引数以外のＲＡＭは使用していません。

長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

PAD

（1）メインルーチン



長周期パルス出力	MCU	H8S／2655	使用機能	TPU（ｶｽｹｰﾄﾞ接続）

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/**********************************/
/*           PROTOCOL             */
/**********************************/
void LPULMN(void);

/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/

# define lpul_wid  (*(unsigned short * )0xffec00) /* Pulse width */
# define lpul_cyc  (*(unsigned short * )0xffec02) /* Pulse cycle */

/**********************************/
/*     MAIN PROGRAM: LPULMN       */
/**********************************/
void LPULMN(void)
{
        MSTPCR = 0xdfff;        /* Disable module(TPU) stop mode*/
        TCR2 = 0;               /* Initialize TCR2 */
        TPU_TCR1 = 0x47;        /* Initialize TCR1 */
        TMDR1 = 0xC2;           /* Initialize TMDR1 ch1:PWM mode */
        TIOR1H = 0x52;          /* Initialize TIOR1H */
        TGR1A = lpul_wid;       /* Set pulse low period time */
        TGR1B = lpul_cyc;       /* Set pulse cycle time */
        TSTR = 0x06;            /* Start TCNT1,2 */
        while(1);               /* Loop */
}
```

3．5　　ＰＷＭ15相出力

ＰＷＭ１５相出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

仕様

（１）図１に示すようにパルスのＨｉｇｈ幅を変化させ、デューティ変化できるＰＷＭ波形を１５相出力します。
（２）２０ＭＨｚで動作時、出力するＰＷＭ周期は１００ｎｓから３．２７ｍｓの間任意に設定できます。

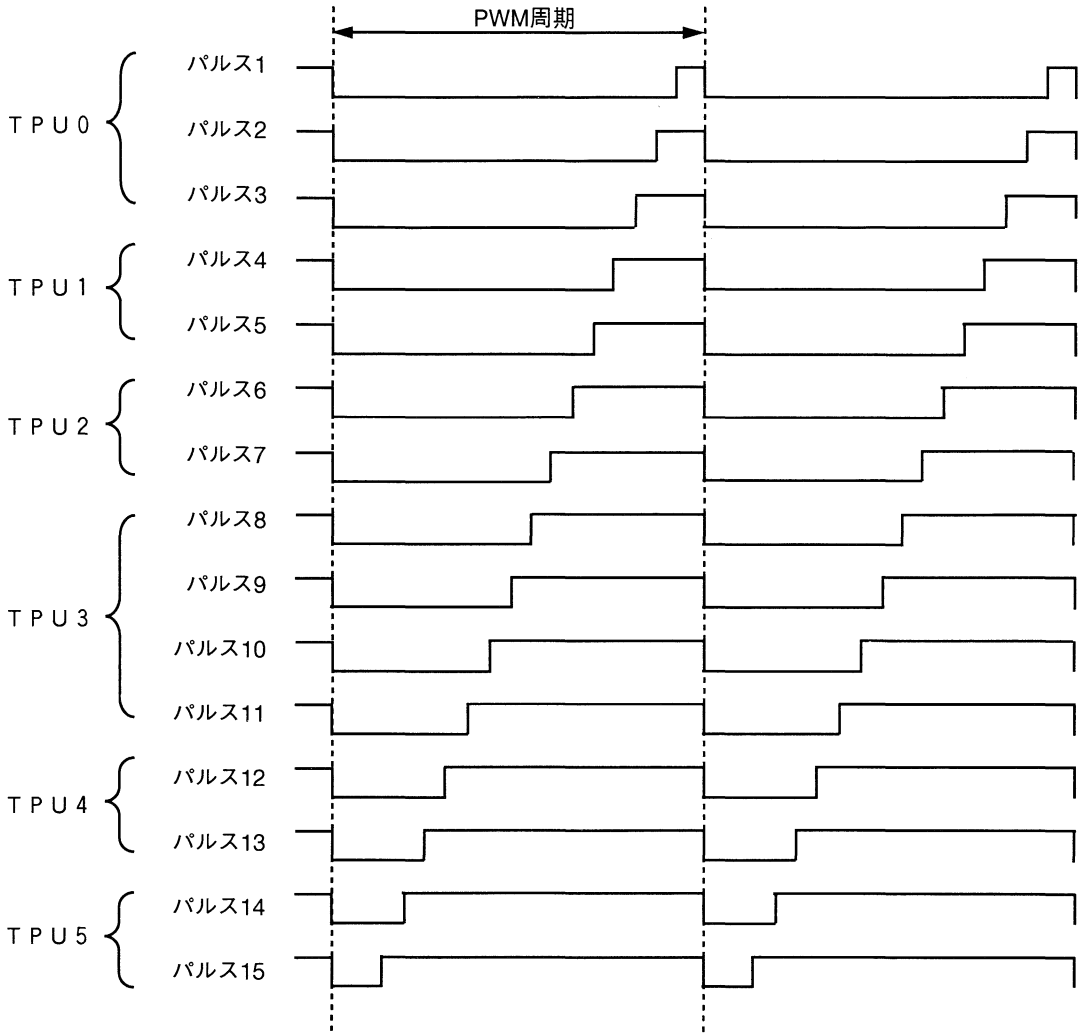


図１　PWM波形出力例

（１）本タスク例ではＴＰＵ０～５まで同期動作させ、１５相のＰＷＭ波形を出力します。

（ａ）図２に本タスク例で使用するＴＰＵのブロック図を示します。

本タスク例では、ＴＰＵの以下の機能を使用しています。

・同期動作と組み合わせることにより、最大15相のＰＷＭ出力が可能（PWMモード2）

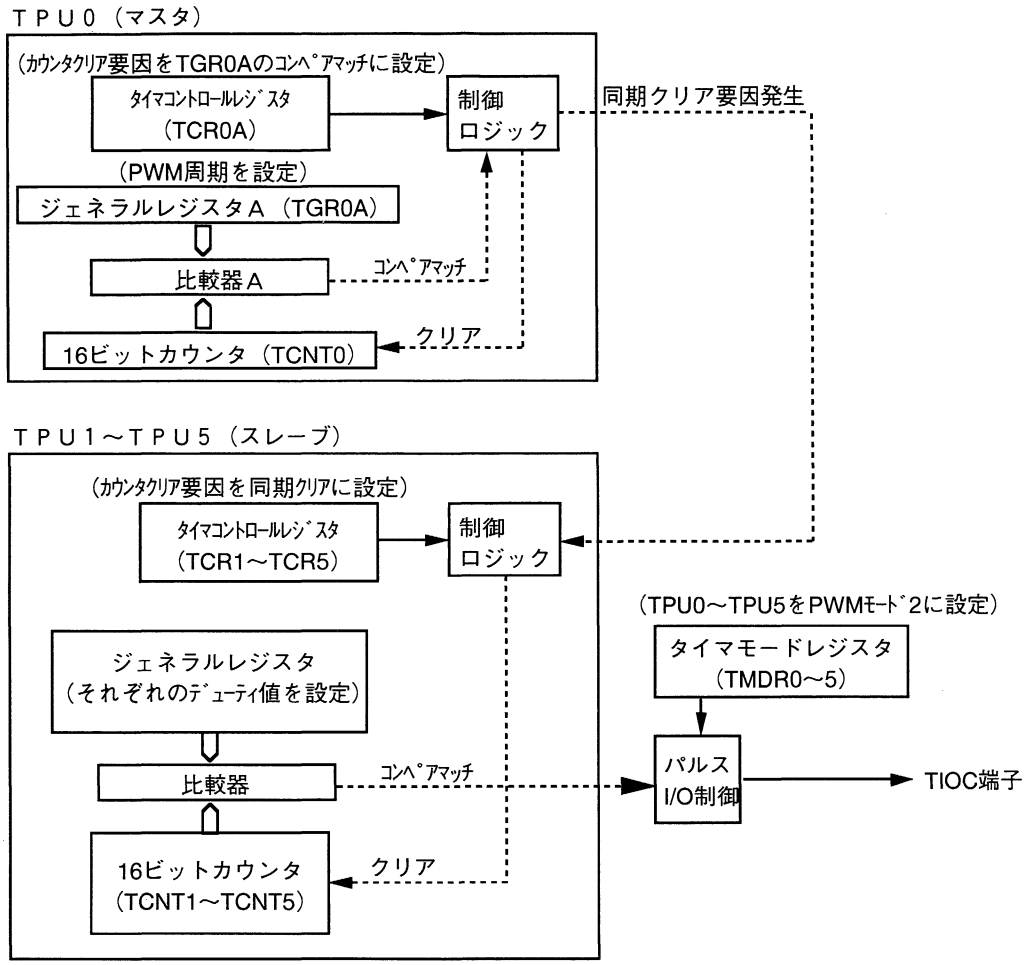


図２　PWM15相出力ブロック図

ＰＷＭ１５相出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

使用機能説明

（2）表1に本タスクの機能割り付けを示します。表1に示すようにＨ８Ｓ／２６５５の機能を割り付け、PWMパルスを出力します。

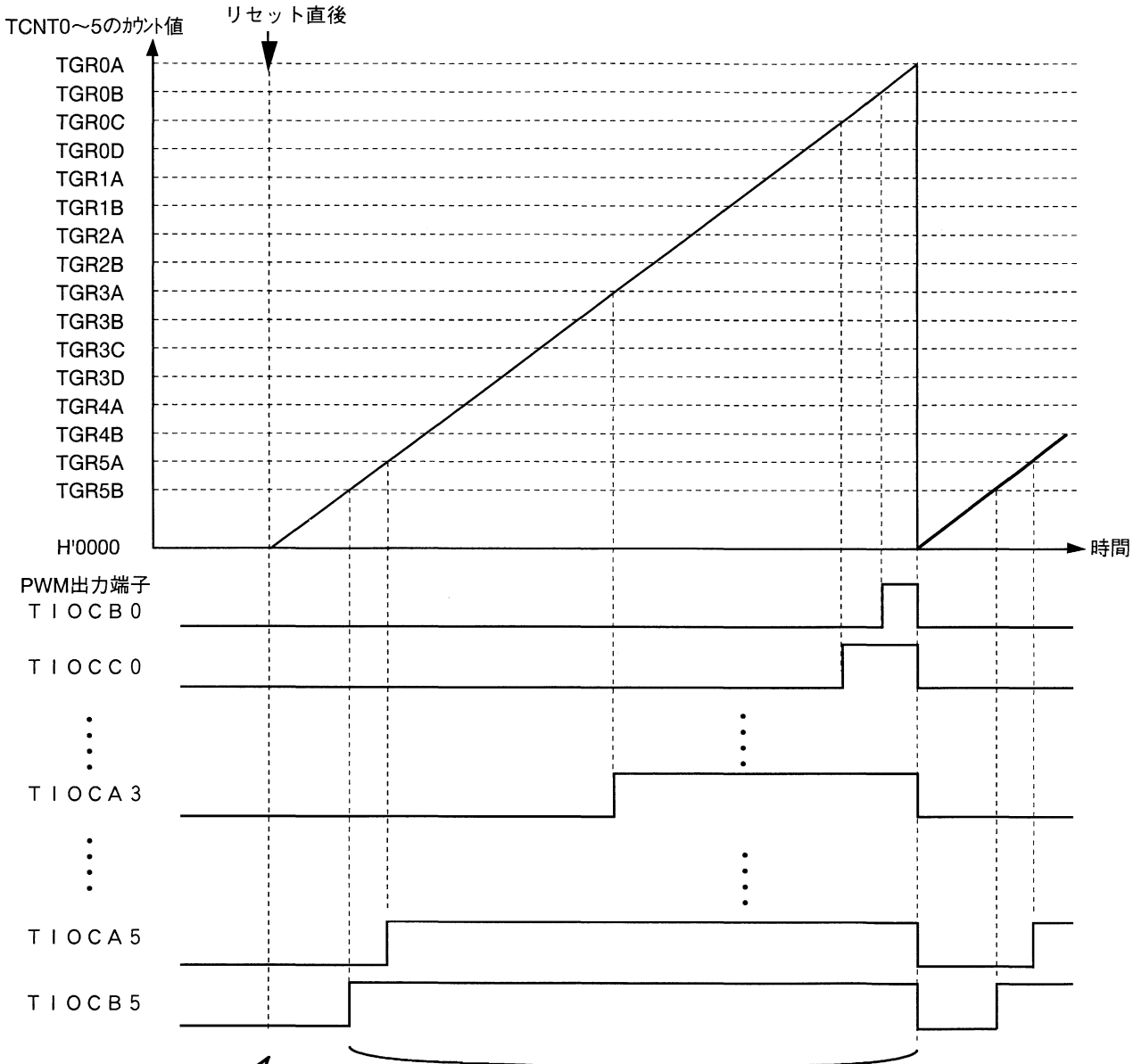
表1　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能	機能
TIOCA1～5 TIOCB0～5、 TIOCC0,3、 TIOCD1,3	PWMパルス出力端子
TCR0 ～TCR5	TPU0～TPU5のタイマカウンタのクリア要因を選択する
TMDR0 ～TMDR5	TPU0～TPU5をPWMﾓｰﾄﾞ2として動作させる。
TGR0A	ＰＷＭ周期を設定する。
TGR0B ～TGR5B	デューティ値を設定する。

PWM１５相出力	MCU	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

動作説明

図３にPWM15相パルス出力の動作原理を示します。図３に示すようにＨ８Ｓ／２６５５のハードウェア処理およびソフトウェアの処理によりTPU0～TPU5のPWM出力端子からパルスを出力します。



ハードウェア処理
処理なし
ソフトウェア処理
初期設定 （ａ）カウンタクロックをφに設定 （ｂ）ＴＣＮＴのクリア要因とするＴＧＲを選択 （ｃ）ＴＧＲをアウトプットコンペアレジスタに設定し、初期値と出力値を選択 （ｄ）ＰＷＭモード２を選択 （ｅ）パルスの周期を周期レジスタ(ＴＧＲ０Ａ)に、デューティをその他のＴＧＲに設定 （ｆ）カウント動作開始

ハードウェア処理
（ａ）ＴＧＲのコンペアマッチで”Ｈ”出力
ソフトウェア処理
処理なし

ハードウェア処理
（ａ）ＴＧＲ０Ａのコンペアマッチでカウンタクリア （ｂ）パルス”Ｌ”出力
ソフトウェア処理
処理なし

図３　ＰＷＭ１５相出力動作原理

PWM１５相出力	MCU	H８S／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

ソフトウェア説明

（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	pwm15mn	ch0～ch5の同期クリアおよびPWM出力の設定を行う

（２）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
pwm[0] ～pwm[14]	パルスのHigh幅に相当するタイマカウンタ値を設定する パルスのHigh幅は以下の式にて求まる ﾊﾟﾙｽのHigh幅(ns)=ﾀｲﾏｶｳﾝﾀ値×φ周期(20MHz動作時50ns) ×各CHの入力ｸﾛｯｸの分周比	unsigned short	ﾒｲﾝﾙｰﾁﾝ	入力
pwm_cyc	PWM周期に相当するタイマカウンタ値を設定する PWM周期は以下の式にて求まる PWM周期(ns)=ﾀｲﾏｶｳﾝﾀ値×φ周期(20MHz動作時50ns) ×各CHの入力ｸﾛｯｸの分周比	unsigned short	ﾒｲﾝﾙｰﾁﾝ	入力

（３）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
TSTR	TPU0～TPU5のタイマカウンタのカウントスタートおよびカウントストップを行う	メインルーチン
TSYR	TPU0～TPU5のTCNTの同期動作を選択する	
TCR0	タイマカウンタのクリア要因をTGR0Aのコンペアマッチに設定する	
TCR1 ～TCR5	タイマカウンタのクリア要因を同期クリアに設定する	
TIOR0 ～TIOR5	各PWM出力端子の出力レベルを設定する	
TMDR0 ～TMDR5	PWMモード２を選択する	
TGR0A	PWM周期を設定する	
TGR0B ～TGR5B	PWM出力端子から"H"出力させるタイマカウンタ値を設定する	
MSTPCR	TPUのモジュールストップモードを解除する	

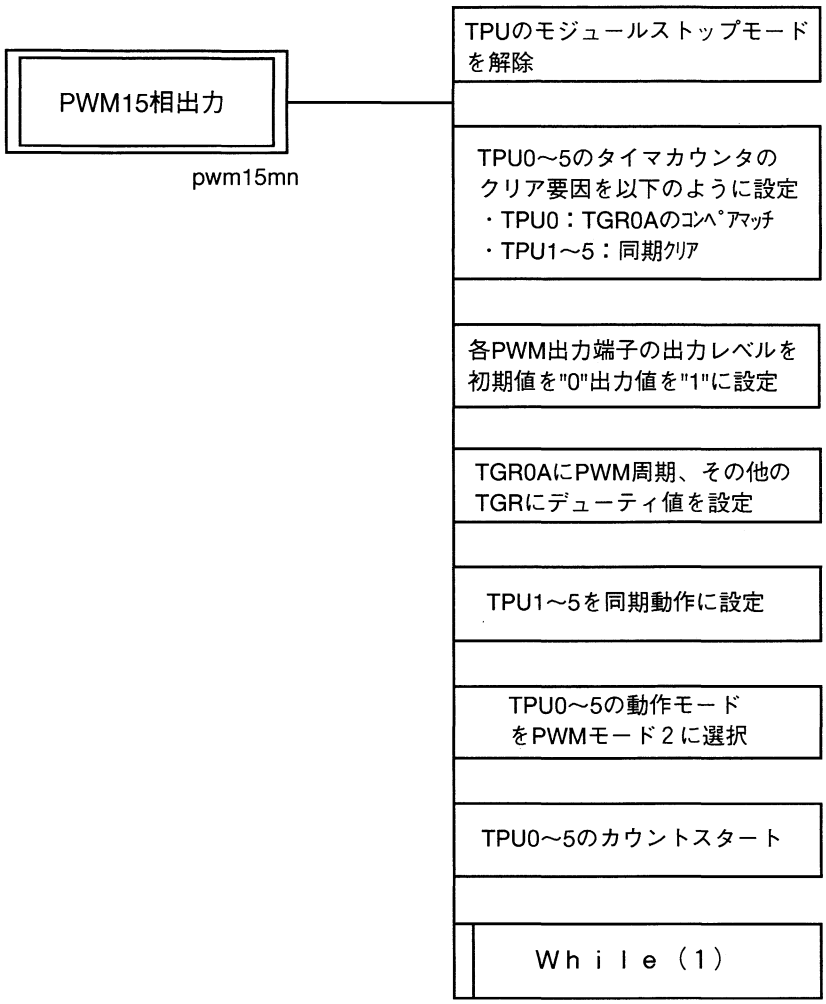
（４）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

PWM１５相出力	MCU	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

ＰＡＤ

メインルーチン



PWM１５相出力	MCU	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（PWMﾓｰﾄﾞ2）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTOCOL                     */
/*****************************************/
void pwm15mn(void);

/*****************************************/
/*        RAM ALLOCATION                 */
/*****************************************/
# define pwm      ((unsigned short * ) 0xffec00)        /* Pulse high width */
# define pwm_cyc  (*(unsigned short * )0xffecle)       /* Pulse cycle */

/*****************************************/
/*     MAIN PROGRAM : pwm15mn            */
/*****************************************/
void pwm15mn(void)
{
        MSTPCR = 0x1fff;        /* Disable module stop mode*/

        TPU_TCR0 = 0x20;        /* Initialize TCR0 */
        TPU_TCR1 = 0x60;        /* Initialize TCR1 */
        TCR2 = 0x60;            /* Initialize TCR2 */
        TCR3 = 0x60;            /* Initialize TCR3 */
        TCR4 = 0x60;            /* Initialize TCR4 */
        TCR5 = 0x60;            /* Initialize TCR5 */

        TIOR0H = 0x20;          /* Initialize TIOR0H */
        TIOR0L = 0x22;          /* Initialize TIOR0L */
        TIOR1H = 0x22;          /* Initialize TIOR1H */
        TIOR2H = 0x22;          /* Initialize TIOR2H */
        TIOR3H = 0x22;          /* Initialize TIOR3H */
        TIOR3L = 0x22;          /* Initialize TIOR3L */
        TIOR4H = 0x22;          /* Initialize TIOR4H */
        TIOR5H = 0x22;          /* Initialize TIOR5H */

        TGR0A = pwm_cyc;        /* Set PWM cycle */
        TGR0B = pwm[0];         /* Set duty */
        TGR0C = pwm[1];
        TGR0D = pwm[2];
        TGR1A = pwm[3];
        TGR1B = pwm[4];
        TGR2A = pwm[5];
        TGR2B = pwm[6];
        TGR3A = pwm[7];
        TGR3B = pwm[8];
        TGR3C = pwm[9];
        TGR3D = pwm[10];
        TGR4A = pwm[11];
        TGR4B = pwm[12];
        TGR5A = pwm[13];
        TGR5B = pwm[14];
    TSYR = 0x3f;            /* Set synchronization mode ch0-5 */

    TMDR0 = 0xc3;           /* Set PWM mode2 */
    TMDR1 = 0xc3;           /* Set PWM mode2 */
    TMDR2 = 0xc3;           /* Set PWM mode2 */
    TMDR3 = 0xc3;           /* Set PWM mode2 */
    TMDR4 = 0xc3;           /* Set PWM mode2 */
    TMDR5 = 0xc3;           /* Set PWM mode2 */

    TSTR = 0x3f;            /* Start TCNT0-5 */
    while(1);               /* Loop */
}
```

3．6　外部トリガによる７相パルス出力

外部トリガによる７相パルス出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（同期動作／PWMﾓｰﾄﾞ1）

仕様

（１）図１に示すように外部信号の立ち下がりに同期して７相のパルスを出力します。
（２）外部信号の立ち下がりからの遅延時間およびパルス幅は以下に示す範囲で可変できます。

２００ｎｓ≦遅延時間＜外部信号の周期－パルス幅

５０ｎｓ≦パルス幅＜外部信号の周期－遅延時間

（３）２０ＭＨｚ動作時、外部信号のパルス周期は２５０ｎｓ～３．２７ｍｓの間任意に設定できます。

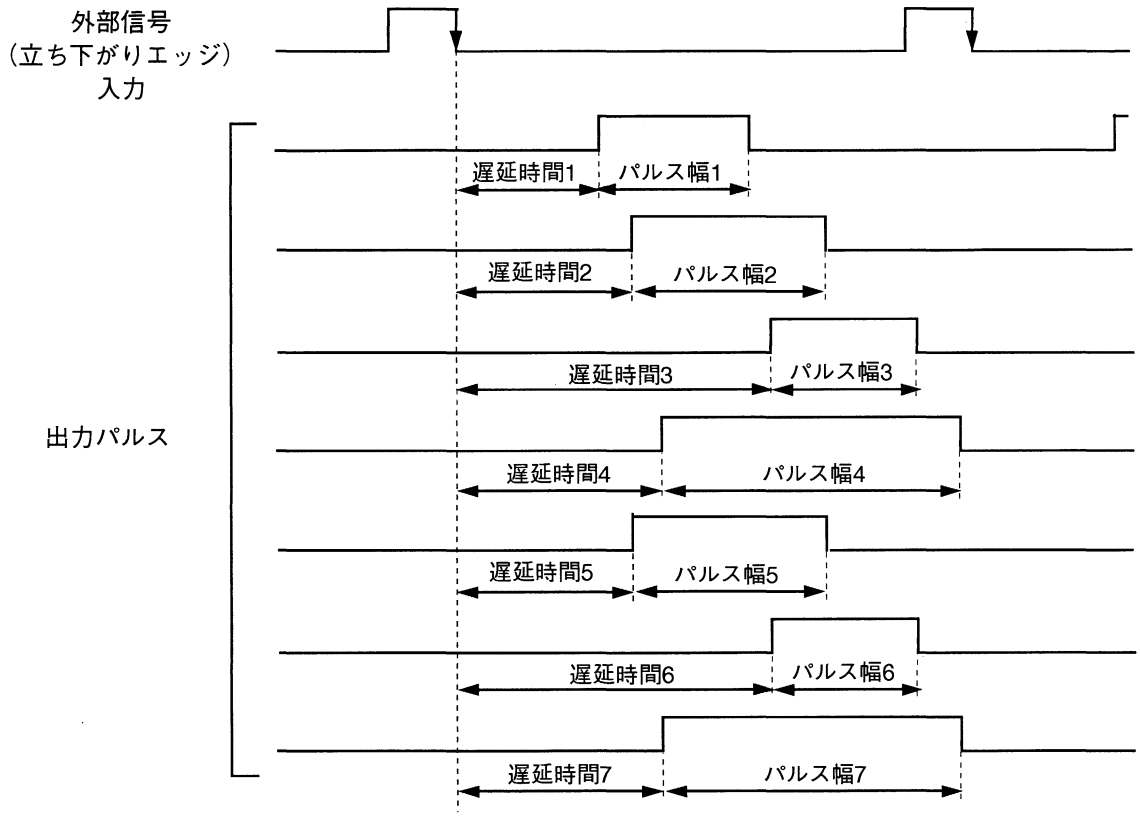


図１　同期パルス出力例

外部トリガによる７相パルス出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（同期動作／PWMﾓｰﾄﾞ1）

使用機能説明

（１）本タスク例では、外部信号を使用した複数のタイマカウンタ同時リセットを行い、７相のパルスを出力します。

（ａ）図２に本タスク例で使用するＴＰＵのブロック図を示します。ＴＰＵの以下の機能を使用し外部信号に同期した７相のパルスを出力します。

・パルスの立ち下がりエッジ検出時タイマカウンタをクリアする機能。

・複数のタイマカウンタを同時クリアする機能。（同期動作）

・ＴＧＲＡとＴＧＲＢ、ＴＧＲＣとＴＧＲＤをペアで使用してＰＷＭ出力を生成。（ＰＷＭモード１）

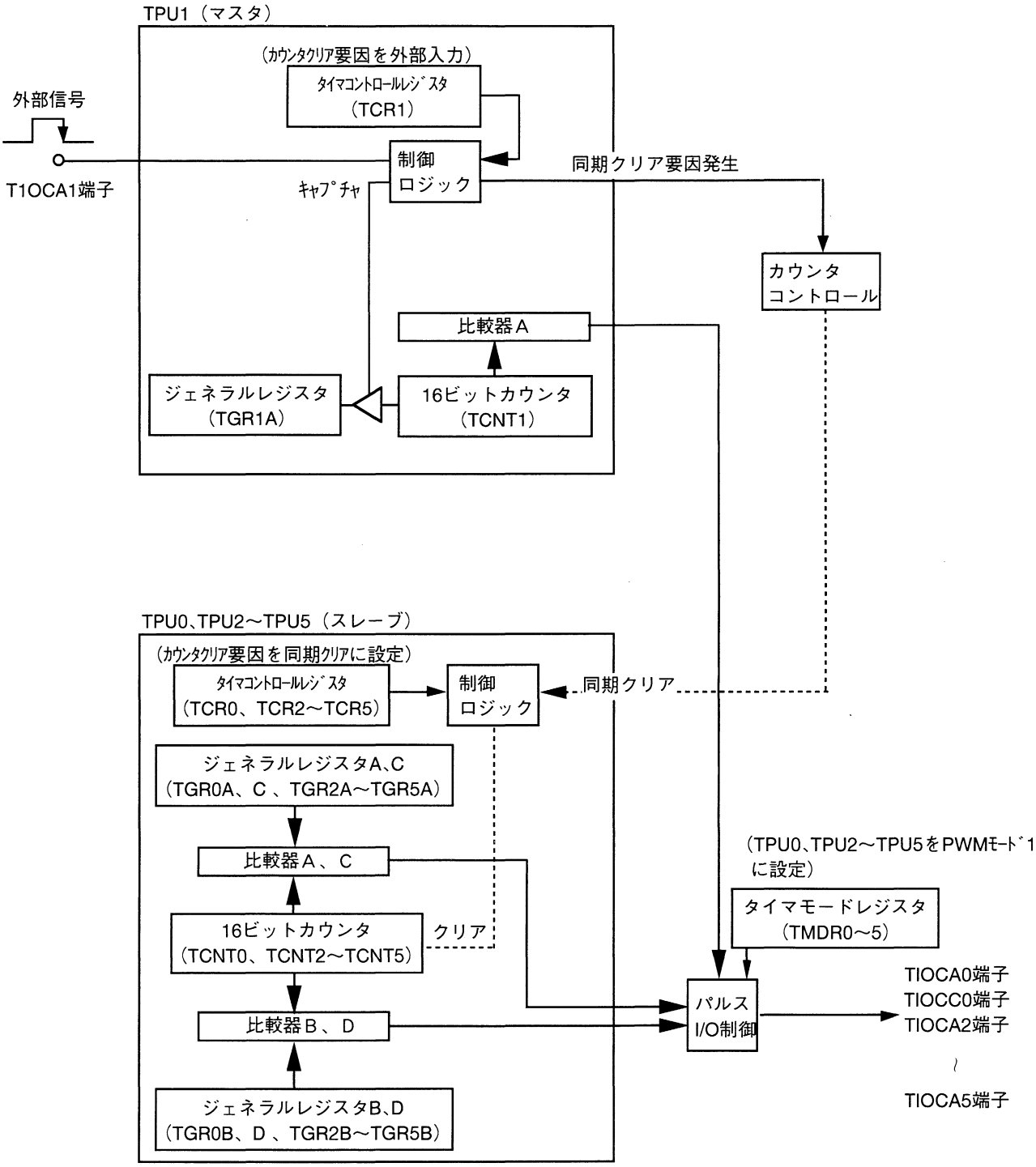


図２　外部ﾄﾘｶﾞによる7相ﾊﾟﾙｽ出力ブロック図

使用機能説明

（２）表１に本タスク例の機能割り付けを示します。表１に示すようにＨ８Ｓ／２６５５の機能を割り付け、外部トリガによる７相パルスを出力します。

表１　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能	機能
TMDR0～5	TPU0、TPU2～5の動作モードをＰＷＭモード１に選択する
TCR0～TCR5	タイマカウンタのクリア要因の選択を行う
TIOCA1	トリガ信号を入力する
TIOCA0端子、TIOCC0端子 TIOCA2端子～TIOCA5端子	ＰＷＭパルスを出力する
TGR0A、C、TGR2A～ TGR5A	パルスの出力レベルを"H"にする（遅延時間）
TGR0B、D、TGR2B～ TGR5B	パルスの出力レベルを"L"にする（パルス幅）

外部トリガによる7相パルス出力	MCU	H8S／2655	使用機能	TPU（同期動作／PWMﾓｰﾄﾞ1）

動作説明

図3に外部信号に同期した7相パルス出力の動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理によりPWMパルスを出力します。

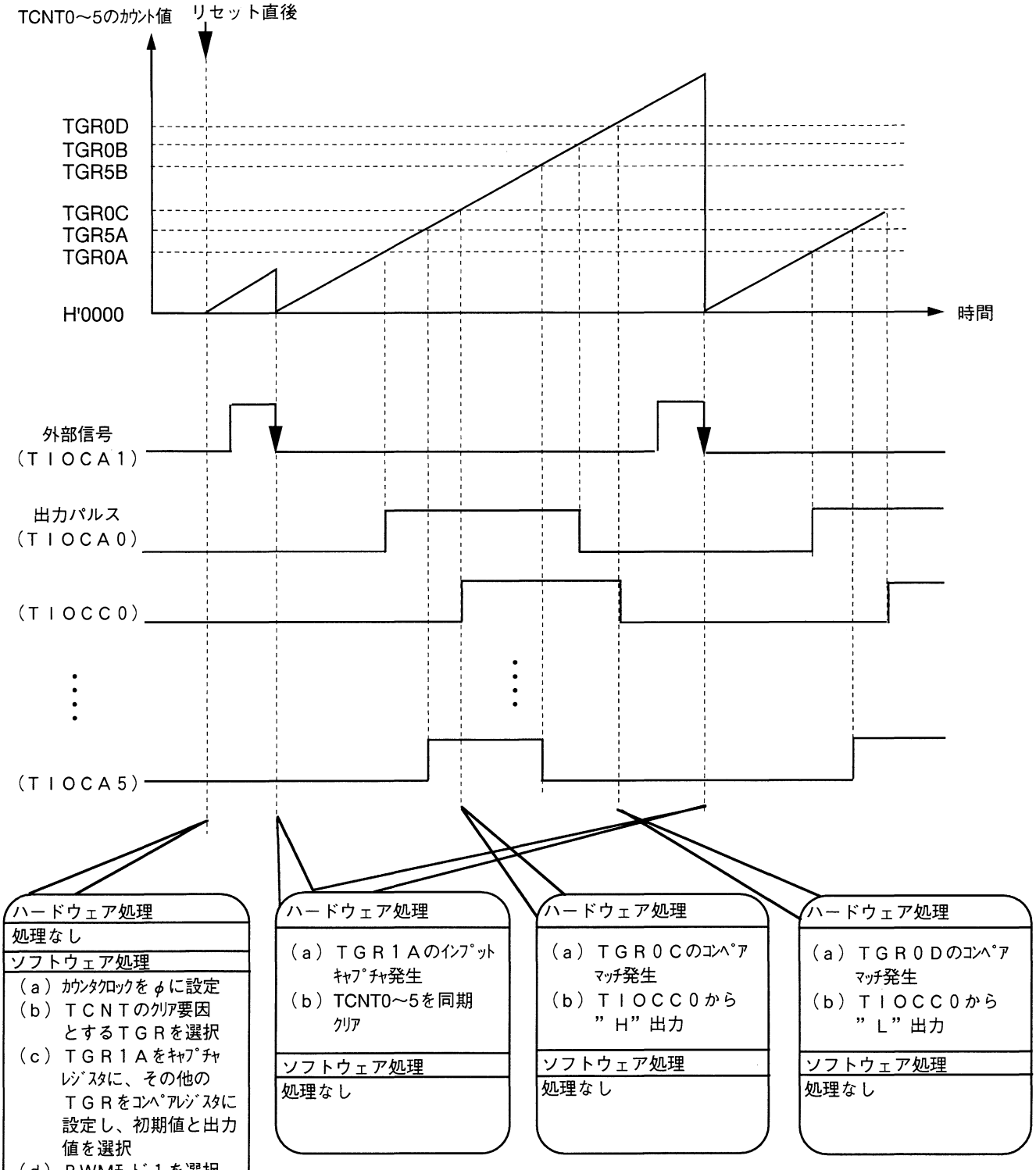


図3　パルス出力動作原理

外部トリガによる７相パルス出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（同期動作／PWMﾓｰﾄﾞ1）

ソフトウェア説明

（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	cntrsmn	TPU0～TPU5の同期クリアおよびPWM出力の設定を行う

（２）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
set_wid[0] ～set_wid[6]	パルスのHigh幅に相当するタイマカウンタ値を設定する パルスのHigh幅は以下の式にて求まる ﾊﾟﾙｽのHigh幅(ns)=ﾀｲﾏｶｳﾝﾀ値×φ周期(20MHz動作時50ns) ×各chの入力ｸﾛｯｸの分周比	unsigned short	ﾒｲﾝﾙｰﾁﾝ	入力
set_dly[0] ～set_dly[6]	外部入力パルスの立ち下がりからの遅延時間に相当する タイマカウンタ値を設定する 遅延時間は以下の式にて求まる 遅延時間(ns)=ﾀｲﾏｶｳﾝﾀ値×φ周期(20MHz動作時50ns) ×各chの入力ｸﾛｯｸの分周比	unsigned short	ﾒｲﾝﾙｰﾁﾝ	入力

（３）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
TSTR	TPU0～TPU5のタイマカウンタのカウントスタートおよびカウントストップを行う	メインルーチン
TSYR	TPU0～TPU5のTCNTの同期動作を選択する	
TCR0	タイマカウンタのクリア要因をTGR0Aのインプットキャプチャに設定する	
TCR1 ～TCR5	タイマカウンタのクリア要因を同期クリアに設定する	
TIOR0 ～TIOR5	各PWM出力端子の出力レベルを設定する TGRA、C：初期出力値"0"　TGRB、D：初期出力値"1"	
TMDR0 ～TMDR5	PWMモード１を選択する	
TGR0A、0C TGR2A～5A	外部入力パルスの立ち下がりからの遅延時間に相当するタイマカウンタ値を設定する	
TGR0A、0C TGR2A～5A	PWM出力端子から"L"出力させるタイマカウンタ値を設定する	
MSTPCR	TPUのモジュールストップモードを解除する	

（４）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

外部トリガによる７相パルス出力	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＴＰＵ（同期動作／PWMﾓｰﾄﾞ1）

ＰＡＤ

（１）メインルーチン

外部ﾄﾘｶﾞによる
７相ﾊﾟﾙｽ出力

cntrsmn

TPUのモジュールストップモード
を解除

TPU0～TPU5のタイマカウンタの
クリア要因を以下のように設定
・TPU1：TGR1Aのｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ
・TPU0、TPU2～5：同期ｸﾘｱ

各PWM出力端子の出力レベルを
TGRA、Cの初期出力値を"0"、
TGRB、Dの初期出力値を"1"に設定

TGR0A、C、TGR2～5Aに遅延時間
TGR0B、D、TGR2～5Dにパルスの
リセット値を設定

TPU0～TPU5のカウンタを
同期動作に設定

TPU0～TPU5の動作モード
をPWMモード１に選択

TPU0～TPU5のカウントスタート

Ｗｈｉｌｅ（１）

外部トリガによる7相パルス出力	MCU	H8S／2655	使用機能	TPU（同期動作／PWMﾓｰﾄﾞ1）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTOCOL                     */
/*****************************************/
void cntrsmn(void);

/*****************************************/
/*      RAM ALLOCATION                   */
/*****************************************/
# define set_wid  ((unsigned short * ) 0xffec00)        /* Pulse width time*/
# define set_dly  ((unsigned short * ) 0xffec0e)        /* Delay time */
# define point    ((unsigned short * ) 0xffeclc)        /* Work */

/*****************************************/
/*     MAIN PROGRAM : cntrsmn            */
/*****************************************/
void cntrsmn(void)
{
        MSTPCR = 0x1fff;        /* Disable module(TPU) stop mode*/
        TPU_TCR0 = 0x60;        /* Initialize TCR0 */
        TPU_TCR1 = 0x20;        /* Initialize TCR1 */
        TCR2 = 0x60;            /* Initialize TCR2 */
        TCR3 = 0x60;            /* Initialize TCR3 */
        TCR4 = 0x60;            /* Initialize TCR4 */
        TCR5 = 0x60;            /* Initialize TCR5 */
        TIOR0H = 0x52;          /* Set TGR0A capture falling edge of input */
        TIOR0L = 0x52;          /* Initialize TIOR0L */
        TIOR1H = 0x09;          /* Initialize TIOR1H */
        TIOR2H = 0x52;          /* Initialize TIOR2H */
        TIOR3H = 0x52;          /* Initialize TIOR3H */
        TIOR3L = 0x52;          /* Initialize TIOR3L */
        TIOR4H = 0x52;          /* Initialize TIOR4H */
        TIOR5H = 0x52;          /* Initialize TIOR5H */

        TGR0A = set_dly[0];     /* Set delay time */
        TGR0C = set_dly[1];
        TGR2A = set_dly[2];
        TGR3A = set_dly[3];
        TGR3C = set_dly[4];
        TGR4A = set_dly[5];
        TGR5A = set_dly[6];

        TGR0B = set_wid[0];     /* Set reset timing */
        TGR0D = set_wid[1];
        TGR2B = set_wid[2];
        TGR3B = set_wid[3];
        TGR3D = set_wid[4];
        TGR4B = set_wid[5];
        TGR5B = set_wid[6];

        TSYR = 0x3f;            /* Set synchronization mode ch0-5 */

        TMDR0 = 0xc2;           /* Set PWM model */
        TMDR1 = 0xc0;           /* Set usual mode */
        TMDR2 = 0xc2;           /* Set PWM model  */
        TMDR3 = 0xc2;           /* Set PWM model */
        TMDR4 = 0xc2;           /* Set PWM model */
        TMDR5 = 0xc2;           /* Set PWM model */

        TSTR = 0x3f;            /* Start TCNT0-5 */
        while(1);               /* Loop */
}
```

3．7　　ワンショットパルス出力

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

仕様

（1）図1に示すように外部信号の立ち下がりに同期してワンショットパルスを出力します。
（2）外部信号の立ち下がりからの遅延時間およびパルス幅は以下に示す範囲で可変できます。
　　1μs≦遅延時間＜基準パルスの周期－パルス幅
　　50ns≦パルス幅＜基準パルスの周期－遅延時間
（3）基準パルスの周波数は、　Hzから入力可能です。

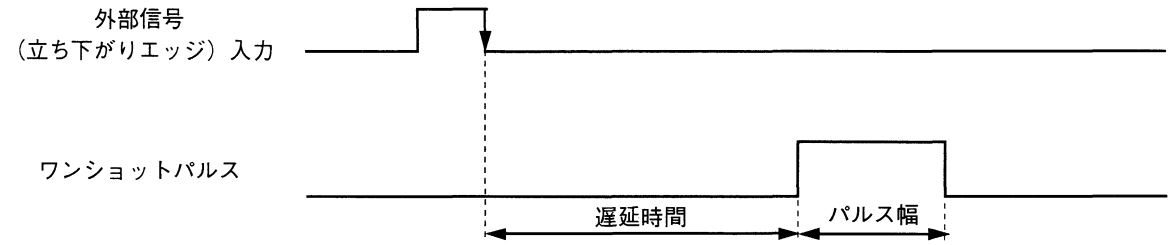


図1　ワンショットパルス出力

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

使用機能説明

（1）本タスク例では、DMAC0A、DMAC0BおよびTPU0を使用してワンショットパルスを出力します。

（a）図2に本タスク例で使用する内蔵機能のブロック図を示します。

本タスク例はTPUとDMACの以下の機能を使用し、ワンショットパルスを出力します。

【TPU】

・コンペアマッチ発生時、バッファレジスタの内容をジェネラルレジスタに転送する機能。（ﾊﾞｯﾌｧ動作）

・各レジスタごとにアウトプット／インプットキャプチャレジスタの設定が可能。

・インプットキャプチャによるカウンタクリアが可能。

【DMAC】

・TPUのインプットキャプチャ発生時にDMACを起動する機能。

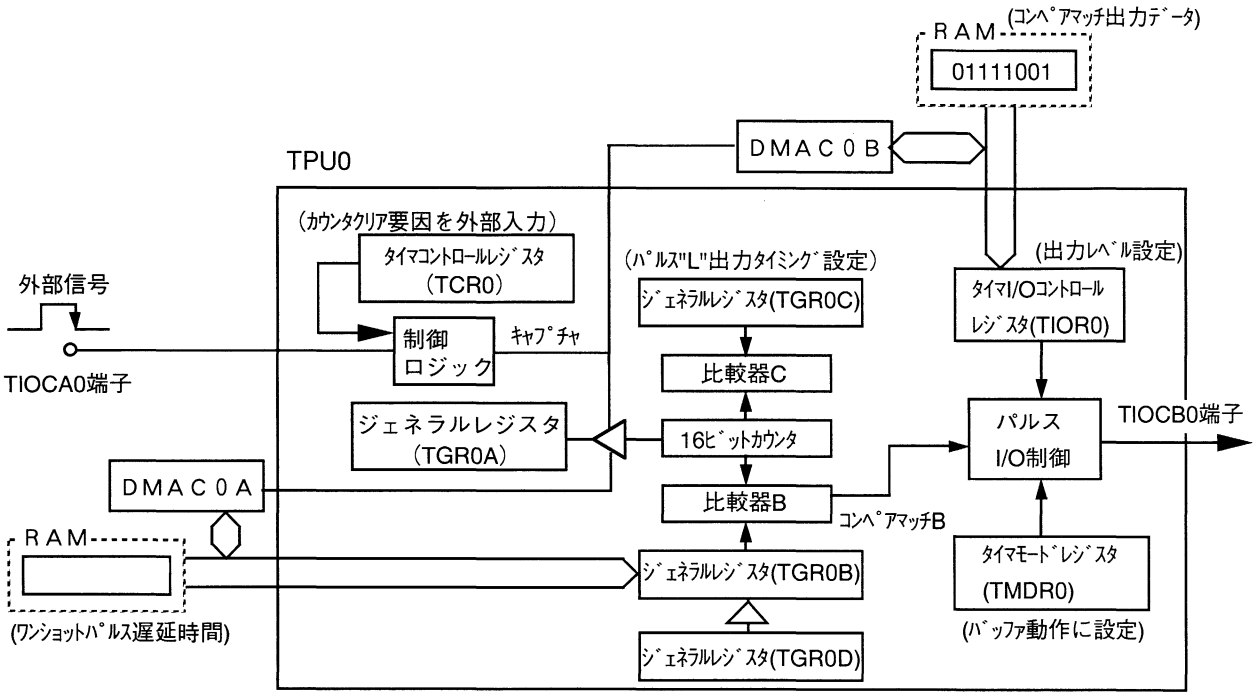


図2　ワンショットパルス出力ブロック図

（2）表1に本タスク例の機能割り付けを示します。本タスク例は表1に示すようにH8S／2655の機能を割り付け、ワンショットパルスを出力します。

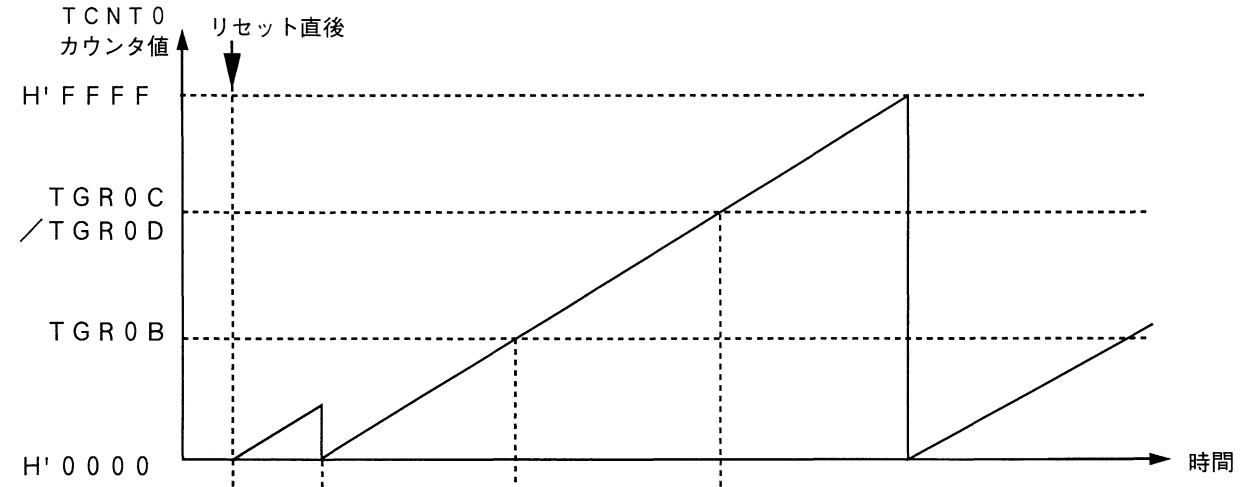
表1　H8S／2655機能割り付け

H8S／2655機能		機　能
TPU0	TCR0	カウンタクリア要因を設定する
	TIER0	TGI0Cによる割り込みを許可する
	TIOR0	TGR0Aをｷｬﾌﾟﾁｬﾚｼﾞｽﾀ、TGR0BとTGR0Cをｺﾝﾍﾟｱﾏｯﾁﾚｼﾞｽﾀに設定する
	TMDR0	バッファ動作の設定をする
	TGR0B	ワンショットパルスの遅延時間の設定をする
	TGR0C	ワンショットパルスの出力禁止タイミング値を設定する
	TGR0D	ワンショットパルスのリセットタイミング値を設定する
	TIOCA0	外部信号を入力する
	TIOCB0	ワンショットパルスを出力する
DMAC	DMABCRH,L	DMACの各チャネルの動作を制御する
	DMACR0A,B	各ﾁｬﾈﾙの転送サイズ、モード、起動要因を設定する
	MAR0A,B	転送元アドレスを設定する
	IORA0A,B	転送先アドレスを設定する
	ETCR0A,B	転送回数を設定する

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理により、ワンショットパルスを出力します。



ハードウェア処理

処理なし

ソフトウェア処理

初期設定
DMAC0A／B
（a）転送フォーマットを設定
（b）データ転送起動要因をch0のインプットキャプチャAに選択
（c）DMAC0A／Bを起動
TPU0
（a）カウント動作許可
（b）インプットキャプチャA検出エッジを立ち下がりに設定
（c）カウンタクリア要因をインプットキャプチャAクリアに設定
（d）バッファモードに設定

ハードウェア処理

TPU0
インプットキャプチャA発生

DMAC0A
ワンショットパルス出力許可データをTIOR0に転送

DMAC0B
ワンショットパルスの遅延時間をTGR0Bに転送

ソフトウェア処理

処理なし

ハードウェア処理

TPU0
（a）コンペアマッチB発生
（b）TGR0D→TGR0Bデータ転送
（c）TIOCB0のコンペアマッチ出力

ソフトウェア処理

処理なし

ハードウェア処理

TGI0C発生

ソフトウェア処理

TGI0C処理
ワンショットパルス出力禁止

図3　ワンショットパルス出力動作原理

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

ソフトウェア説明

（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	ONEMN	遅延時間およびパルス幅をTGR0B、TGR0Dに、ワンショットパルスリセット値をTGR0C設定し、ワンショットパルスを出力する
ﾊﾟﾙｽ出力禁止	POUTDLE	パルス出力を禁止する

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
set_dly	ワンショットパルスの遅延時間に相当するタイマ値を設定する。遅延時間は以下の式にて求まる 遅延時間(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short	メインルーチン	入力
one_rst	ワンショットパルスのリセットタイミングに相当するタイマ値を設定する。リセットタイミングは以下の式にて求まる ﾊﾟﾙｽﾘｾｯﾄﾀｲﾐﾝｸﾞ(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short	メインルーチン	入力
io_cntr	ワンショットパルス出力許可データを設定する （インプットキャプチャAを立ち下がり、コンペアマッチBをトグル出力）	unsigned char	メインルーチン	出力

（3）使用内部レジスタ説明

レジスタ名		機能	使用ﾓｼﾞｭｰﾙ名
TPU0	TSTR	タイマカウンタの動作／停止を選択する	ﾒｲﾝﾙｰﾁﾝ
	TMDR	TGR0B、TGR0Dをバッファ動作に設定する	
	TCR0	TCNTに入力するクロック、およびカウンタクリア要因を設定する	
	TIOR0	入力パルスの立ち下がりエッジを検出する。	
		コンペアマッチB発生時に、TIOCB0から出力するレベルを設定する	ﾊﾟﾙｽ出力禁止
	TIER0	TGI0Cによる割り込みを許可する	ﾒｲﾝﾙｰﾁﾝ／ﾊﾟﾙｽ出力禁止
	TSR0	TGR0Bによるコンペアマッチの発生を示す	ﾒｲﾝﾙｰﾁﾝ
	TGR0B	ワンショットパルスの遅延時間を設定する	
	TGR0C	ワンショットパルスのパルス出力禁止タイミング値を設定する	
	TGR0D	ワンショットパルスのリセットタイミング値を設定する	
DMAC	DMABCR0 DMACR0 A／B	DMACの各チャネルの動作を設定する	
	MAR0 A／B	各レジスタに転送するデータのアドレスを設定する	
	IOAR0 A／B	各chの転送先レジスタのアドレスを設定する	
	ETCR0 A／B	各chの転送回数を設定する	
MSTPCR		TPU、DMACのモジュールストップモードを解除する	

（4）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

PAD

（1）メインルーチン

メインルーチン

ONEMN

TPU、DMACのモジュールストップモードを解除

TCR0を設定し、カウンタクリア要因をインプットキャプチャA

TMDR0を設定し、TGR0B、TGR0Dをバッファ動作に設定

ワンショットパルスのﾘｾｯﾄﾀｲﾐﾝｸﾞ（one_rst）をTGR0Dに設定

ワンショットパルスのコンペマッチ出力禁止タイミングをTGR0Cに設定

TIOCB0出力許可データを（io_cntr）に設定

ワンショットパルス遅延時間の設定アドレス（set_dly）を転送元アドレスとしてMAR0Aに設定

TPU0のコンペアマッチ出力設定アドレス（io_cntr）を転送元アドレスとしてMAR0Bに設定

1

1

TGR0Bのアドレスを転送先アドレスとして、IOAR0Aに設定

TIOR0のアドレスを転送先アドレスとして、IOAR0Bに設定

転送回数をETCR0A／Bにそれぞれ設定

DMACR0AによりDMAC0Aの動作設定
・起動要因をITU0のｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA
・ﾘﾋﾟｰﾄﾓｰﾄﾞ
・1ﾊﾞｲﾄﾃﾞｰﾀ転送

DMACR0BによりDMAC0Bの動作設定
・起動要因をITU0のｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA
・ﾘﾋﾟｰﾄﾓｰﾄﾞ
・1ﾜｰﾄﾞﾃﾞｰﾀ転送

DMACR0Lをリード

DMACR0H／LによりDMAC0A、Bを転送許可に設定

DMAC起動要因となる割り込み許可をTIER0に設定

TIER0を設定し、TGFCによる割り込みを許可

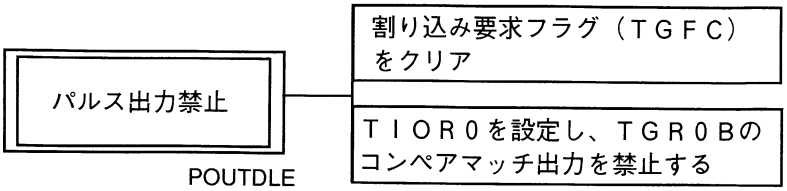
Iフラグをクリアし割り込みを許可

TPU0のカウントスタート

while（1）

ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)
PAD				

（2）パルス出力禁止



ワンショットパルス出力	MCU	H8S／2655	使用機能	TPU(バッファ動作)

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/**********************************/
/*          PROTOCOL              */
/**********************************/
void ONEMN(void);
#pragma interrupt (POUTDLE)
/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/
# define set_dly  ((unsigned short * )0xffec00)  /* Pulse delay time */
# define one_rst  (*(unsigned short * )0xffec02) /* Pulse reset timing */
# define io_cntr  ((unsigned char * )0xffec04)   /* I/O control */

/**********************************/
/*     MAIN PROGRAM: ONEMN        */
/**********************************/
void ONEMN(void)
{
        MSTPCR = 0x5fff;        /* Disable module(DMA,TPU) stop mode*/
        TPU_TCR0 = 0x20;        /* Initialize TCR0 */
        TMDR0 = 0xE0;           /* TGR0B,D: buffer register */
        TGR0D = one_rst;
        TGR0C = one_rst;
        io_cntr[0] = 0x39;      /* Set output toggles at GRB compare match */
                                /* Set TGRA0 captures falling edge of input */

        MAR0A_W = set_dly;      /* Set base address */
        MAR0B_B = io_cntr;      /* Set base address */
        IOAR0A = 0xffda;        /* Set excute (TGR0B) address */
        IOAR0B = 0xffd2;        /* Set excute (TIOR0H) address */
        ETCR0A = 0x0101;        /* Set excute count */
        ETCR0B = 0x0101;        /* Set excute count */
        DMACR0A = 0xA8;         /* Initialize DMACR0 */
        DMACR0B = 0x28;

        DMABCRH = 0x03;         /* Initialize DMABCR0 */
        DMABCRL |= 0x30;


        TIER0_BP.TGIEA0 = 1;    /* Enable DMAC */
        TIER0_BP.TGIEC0 = 1;    /* Enable TGI0C */
        set_imask_ccr(0);       /* Enable interrupt */
        TSTR = 0x01;            /* Start TCNT0 */
        while(1);               /* Loop */
}

/**********************************/
/*   INTERRUPT PROGRAM: POUTDLE   */
/**********************************/
void POUTDLE(void)
{
        TSR0_BP.TGFC0 = 0;      /* Clear TGFC0 request */
        TIOR0H = 0x09;          /* Set disenable output data */
}
```

3．8　　4ビット×4系統出力

4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

仕様

（1）図1に示すように、ＰＰＧ出力を使用して4ビット×4系統の非同期パルスを出力します。
（2）ＰＰＧの起動要因には、ＴＰＵのコンペアマッチを使用します。

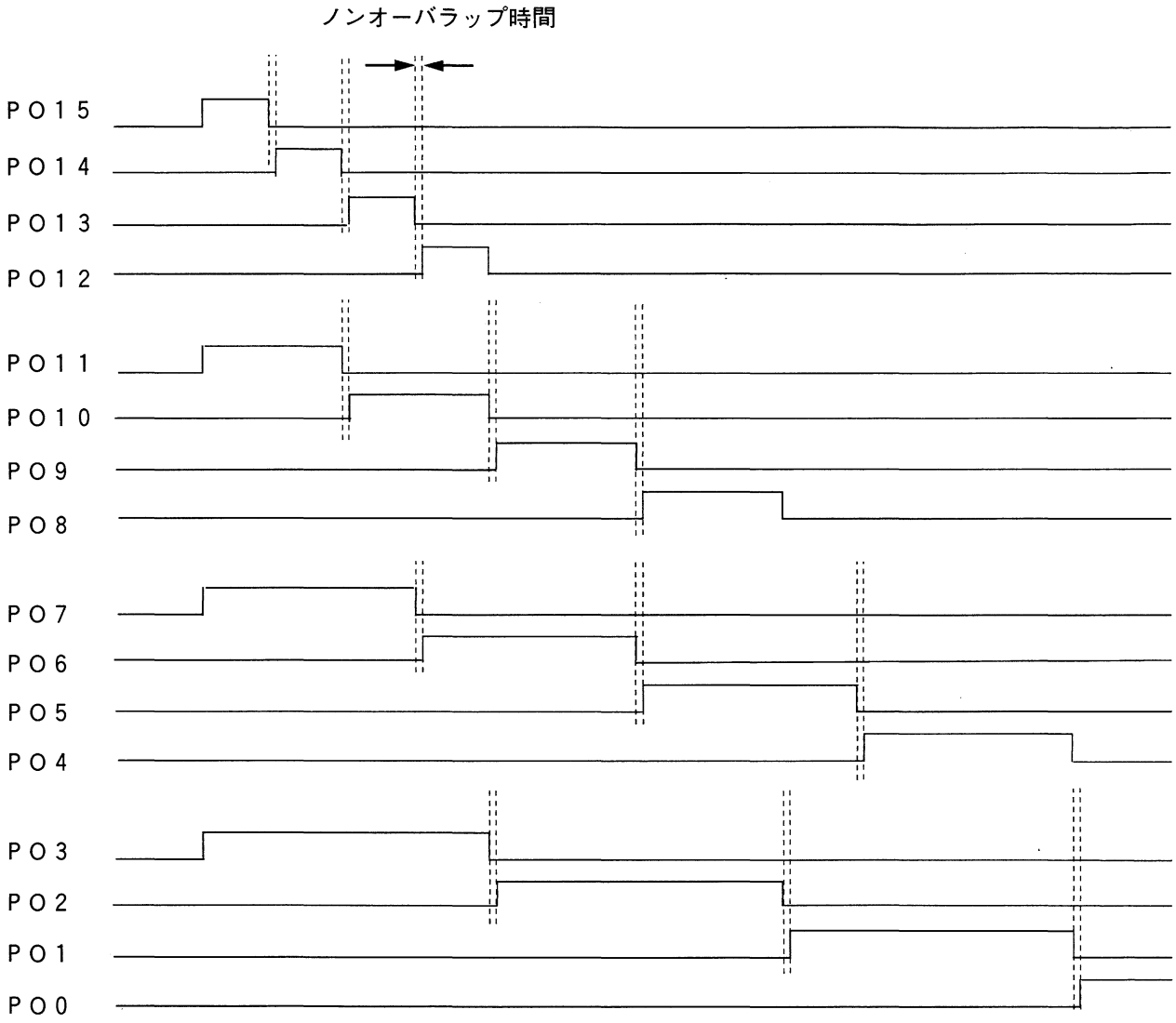


図1　4ビット×4系統出力例

4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

使用機能説明

（1）本タスク例では、ＴＰＵ0～ＴＰＵ3およびＰＰＧ出力グループ3～0を使用し、4ビット×4系統の非同期パルスを出力します。

（a）図2に本タスク例で使用するＴＰＵ3、ＰＰＧ／グループ3を例にパルス出力のブロック図を示します。
本タスク例では、以下の機能を使用しています。

・4ビット単位のグループで出力トリガ信号が選択可能で最大4ビット×4系統出力が可能。

・ＴＰＵの4チャネルのコンペアマッチ信号の中からグループごとに出力トリガ信号の選択が可能。

・複数のパルス出力の間のノンオーバラップ期間を設定することが可能。

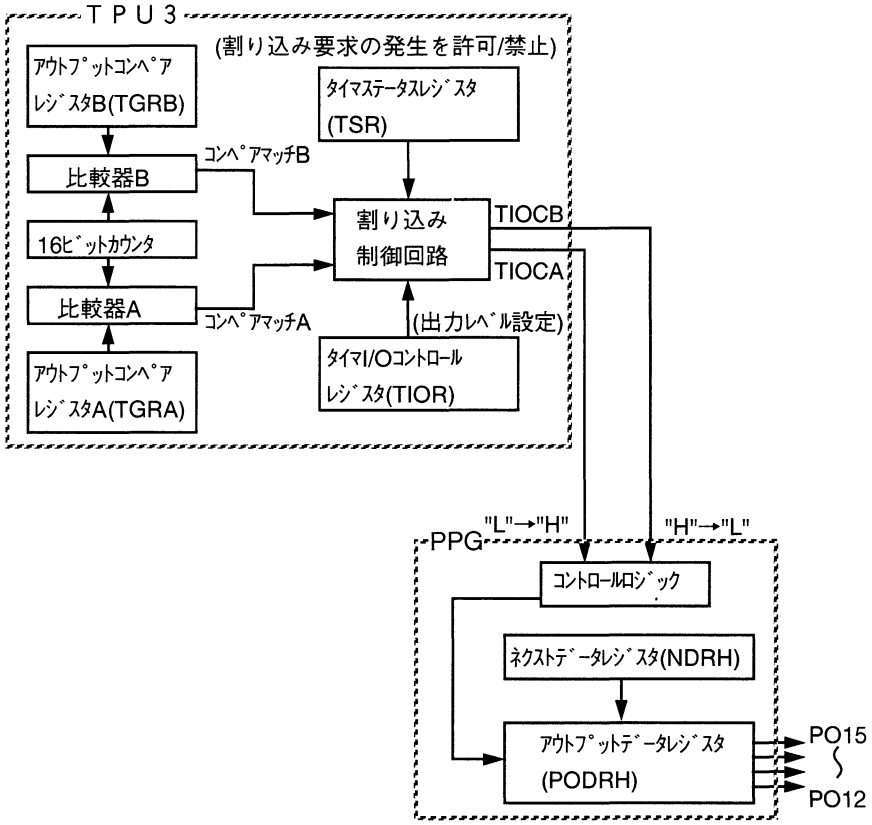


図2　4ﾋﾞｯﾄ×4系統出力(ｸﾞﾙｰﾌﾟ3)ブロック図

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、4ビット×4系統出力を行います。

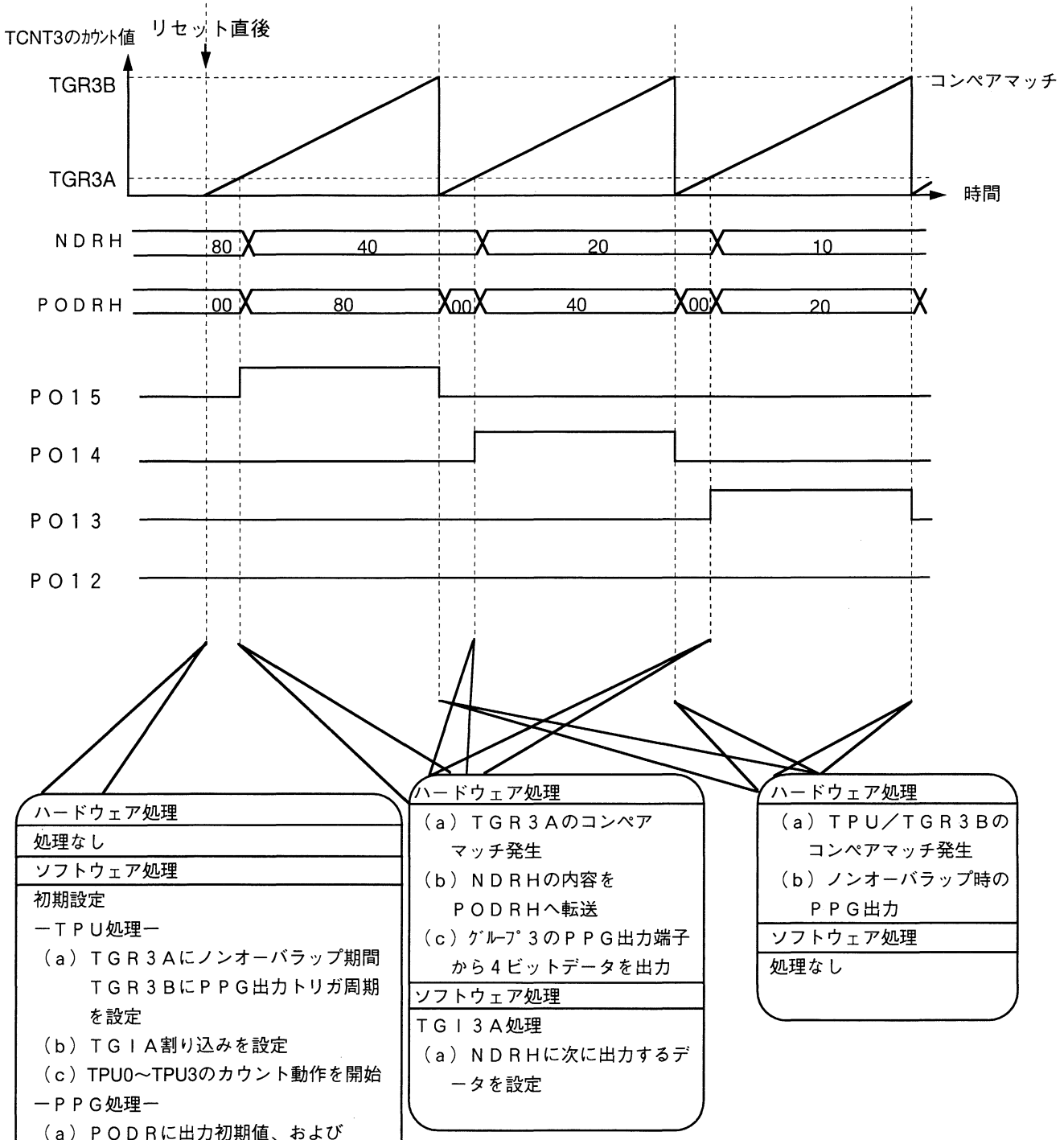
表1　H8S／2655機能割り付け

H8S/2655機能		機能
PPG	PO15～PO0	4ビット×4系統の出力を行う
	PMR	ノンオーバラップモードを設定する
	PCR	PPG出力の出力トリガ信号を設定する
	NDERH	PO15～PO8のPPG出力を許可する
	NDERL	PO7～PO0のPPG出力を許可する
	NDRH	次に出力するPPG出力データを格納する
	NDRL	次に出力するPPG出力データを格納する
	P1DDR	PPG出力端子の設定をする
	P2DDR	PPG出力端子の設定をする
	PODRH	PO15～PO8の出力データを格納する
	PODRL	PO7～PO0の出力データを格納する
TPU	TGR0A～3A	ノンオーバラップ時間を設定する
	TGR0B～3B	PPG出力トリガの周期を設定する
	TCR0～3	カウンタクロックおよびカウンタクリア要因を設定する
	TSR0～3	コンペアマッチの発生を示す
	TIOR0～3	TGRを制御する
	TSTR	タイマカウンタの動作／停止を選択する
MSTPCR		TPU、DMACのモジュールストップモードを解除する

4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

動作説明

図3にＰＰＧ出力グループ３を使用したデータ出力動作原理を示します。Ｈ８Ｓ／２６５５のハードウェア処理およびソフトウェアの処理により４相のノンオーバラップ出力を行います。



ハードウェア処理
処理なし
ソフトウェア処理
初期設定
－ＴＰＵ処理－
（ａ）ＴＧＲ３Ａにノンオーバラップ期間ＴＧＲ３ＢにＰＰＧ出力トリガ周期を設定
（ｂ）ＴＧＩＡ割り込みを設定
（ｃ）TPU0～TPU3のカウント動作を開始
－ＰＰＧ処理－
（ａ）ＰＯＤＲに出力初期値、およびＰＰＧ出力の設定
（ｂ）グループ３のＰＰＧ出力許可、および転送トリガを選択
（ｃ）ノンオーバラップ動作ｸﾞﾙｰﾌﾟ３を選択
（ｄ）ＮＤＲにパルス出力の次の出力値を設定

ハードウェア処理
（ａ）ＴＧＲ３Ａのコンペアマッチ発生
（ｂ）ＮＤＲＨの内容をＰＯＤＲＨへ転送
（ｃ）ｸﾞﾙｰﾌﾟ３のＰＰＧ出力端子から４ビットデータを出力
ソフトウェア処理
ＴＧＩ３Ａ処理
（ａ）ＮＤＲＨに次に出力するデータを設定

ハードウェア処理
（ａ）ＴＰＵ／ＴＧＲ３Ｂのコンペアマッチ発生
（ｂ）ノンオーバラップ時のＰＰＧ出力
ソフトウェア処理
処理なし

図３　4ﾋﾞｯﾄ×4系統出力(ｸﾞﾙｰﾌﾟ3)動作原理

4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

ソフトウェア説明

（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	ppg16mn	PPG、TPUの初期設定を行う。
データ設定0	setdat0	次に出力するデータをNDR0（ｸﾞﾙｰﾌﾟ0）に設定する。
データ設定1	setdat1	次に出力するデータをNDR1（ｸﾞﾙｰﾌﾟ1）に設定する。
データ設定2	setdat2	次に出力するデータをNDR2（ｸﾞﾙｰﾌﾟ2）に設定する。
データ設定3	setdat3	次に出力するデータをNDR3（ｸﾞﾙｰﾌﾟ3）に設定する。

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
addcnt0	PPGグループ0出力の転送カウンタ	unsigned char	データ設定0	出力
addcnt1	PPGグループ1出力の転送カウンタ	unsigned char	データ設定1	出力
addcnt2	PPGグループ2出力の転送カウンタ	unsigned char	データ設定2	出力
addcnt3	PPGグループ3出力の転送カウンタ	unsigned char	データ設定3	出力

（3）使用内部レジスタ説明

レジスタ名		機能	使用モジュール
PPG	P1DDR	PO15～PO8までのPPG出力を許可する	メインルーチン
	P2DDR	PO7～PO0までのPPG出力を許可する	メインルーチン
	P1DR	PO15～PO8出力パターンデータを格納する	メインルーチン
	P2DR	PO7～PO0出力パターンデータを格納する	メインルーチン
	PMR	PO15～PO0をノンオーバラップ出力に設定する	メインルーチン
	PCR	ﾊﾟﾙｽ出力の出力ﾄﾘｶﾞ信号をｸﾞﾙｰﾌﾟ単位で選択する ｸﾞﾙｰﾌﾟ3：TPU3のコンペアマッチ ｸﾞﾙｰﾌﾟ2：TPU2のコンペアマッチ ｸﾞﾙｰﾌﾟ1：TPU1のコンペアマッチ ｸﾞﾙｰﾌﾟ0：TPU0のコンペアマッチ	メインルーチン
	NDERL	PPG出力PO7～PO0を許可する	メインルーチン
	NDERH	PPG出力PO15～PO8を許可する	メインルーチン
	NDRL	PO7～PO0の次に出力するパターンを設定する	メインルーチン データ設定0，1
	NDRH	PO15～PO8の次に出力するパターンを設定する	メインルーチン データ設定2，3
TPU	TGR0A～3A	ノンオーバラップ時間を設定する	メインルーチン
	TGR0B～3B	PPG出力トリガの周期を設定する	メインルーチン
	TCR0～3	TCRを以下のように設定する ・TGRBのコンペアマッチでカウンタクリア ・内部クロックφでカウント	メインルーチン
	TSR0～3	コンペアマッチの発生を示す	メインルーチン
	TIER0～3	TGIA割り込みを許可する	メインルーチン
	TSTR	TCNTのカウント動作を許可する	メインルーチン
MSTPCR		TPU、PPGのモジュールストップモードを解除する	メインルーチン

（4）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

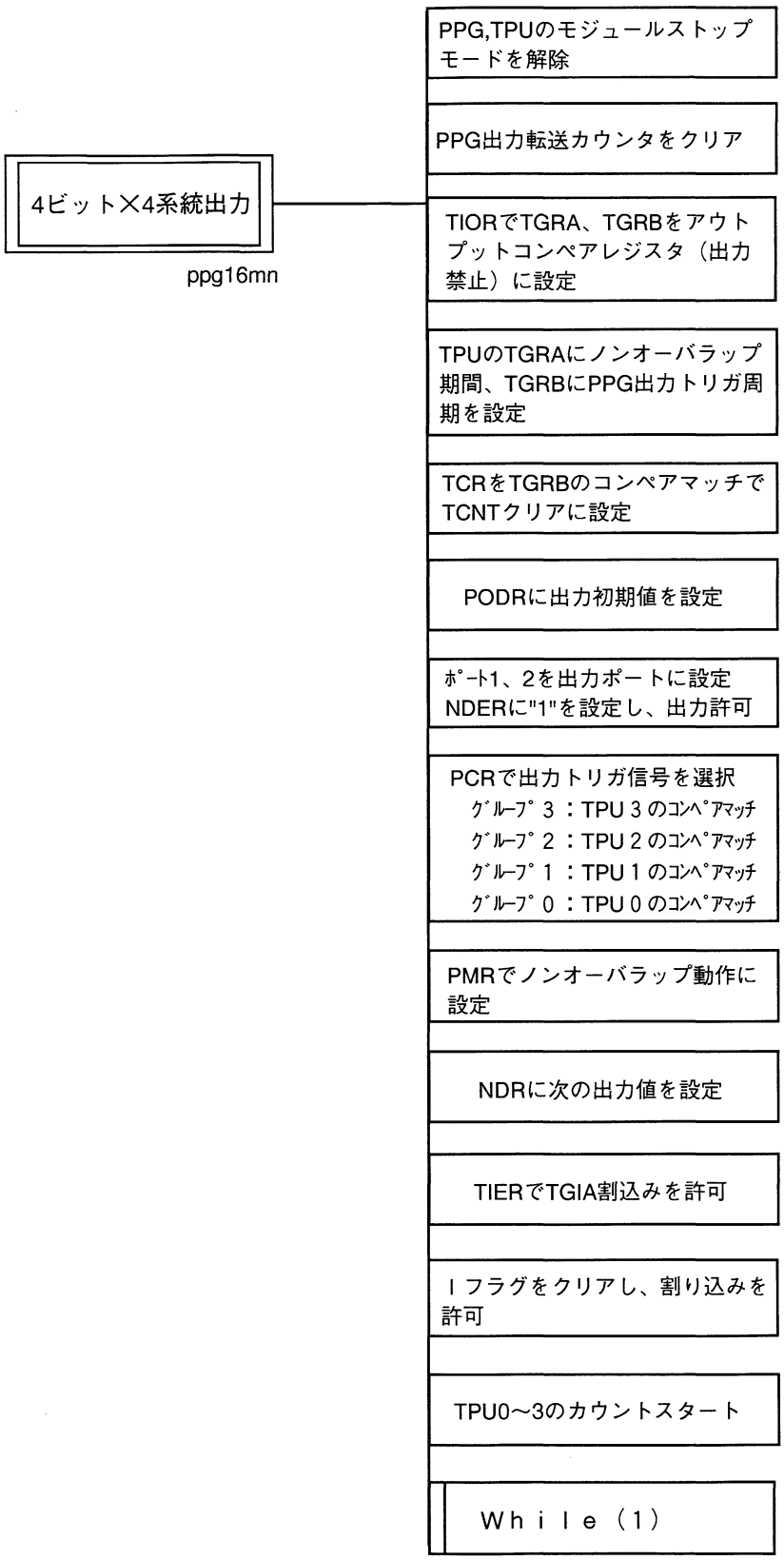
（5）データテーブル説明

テーブル名	機能	データ長	データ容量
ndat_tab0	PPGグループ0から出力するデータを格納	unsigned char	4byte
ndat_tab1	PPGグループ1から出力するデータを格納	unsigned char	4byte
ndat_tab2	PPGグループ2から出力するデータを格納	unsigned char	4byte
ndat_tab3	PPGグループ3から出力するデータを格納	unsigned char	4byte

4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

PAD

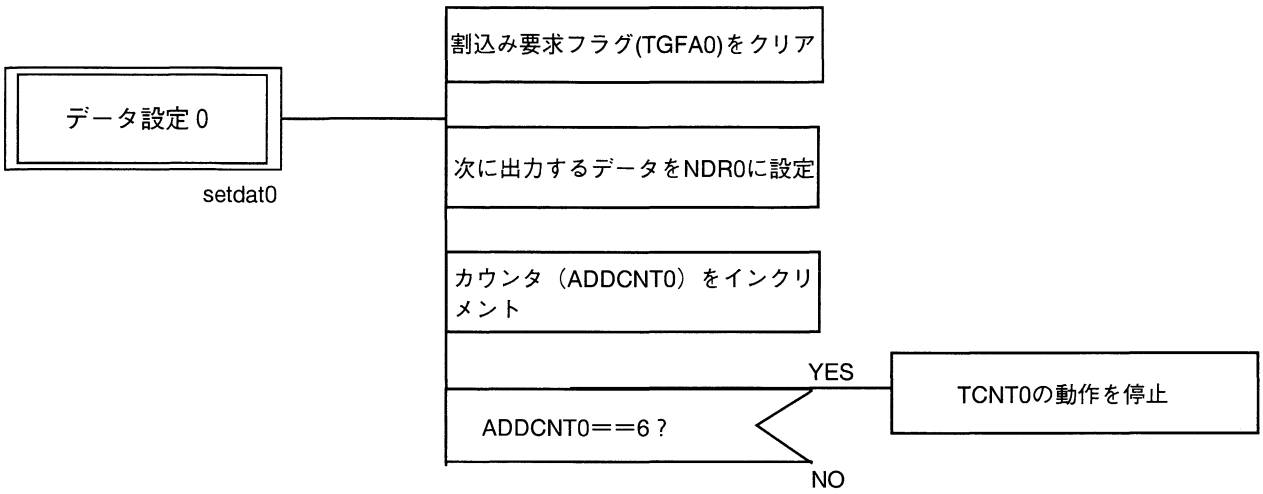
（1）メインルーチン



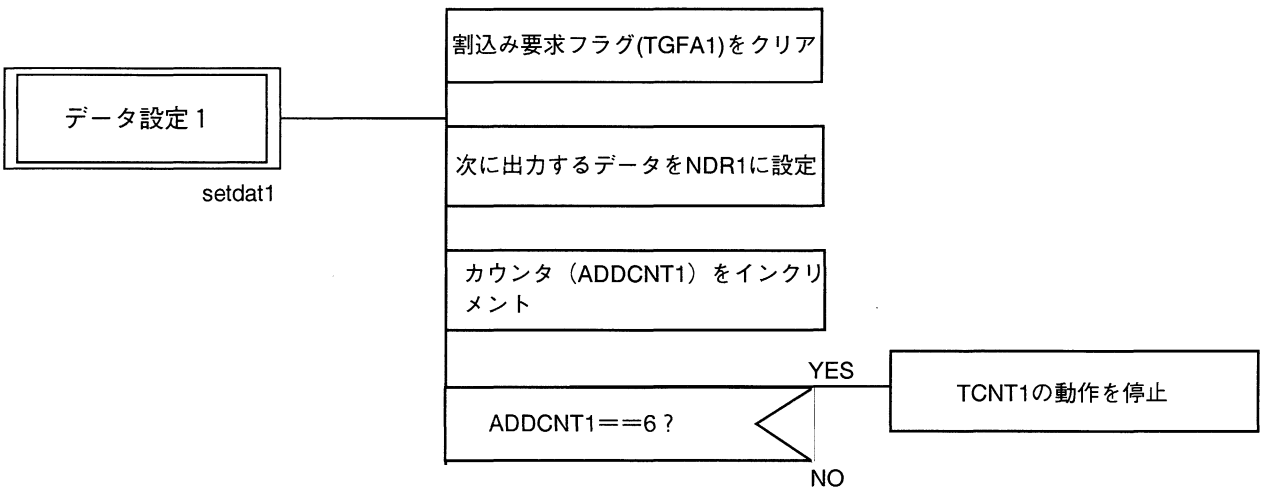
4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

PAD

（2）データ設定0



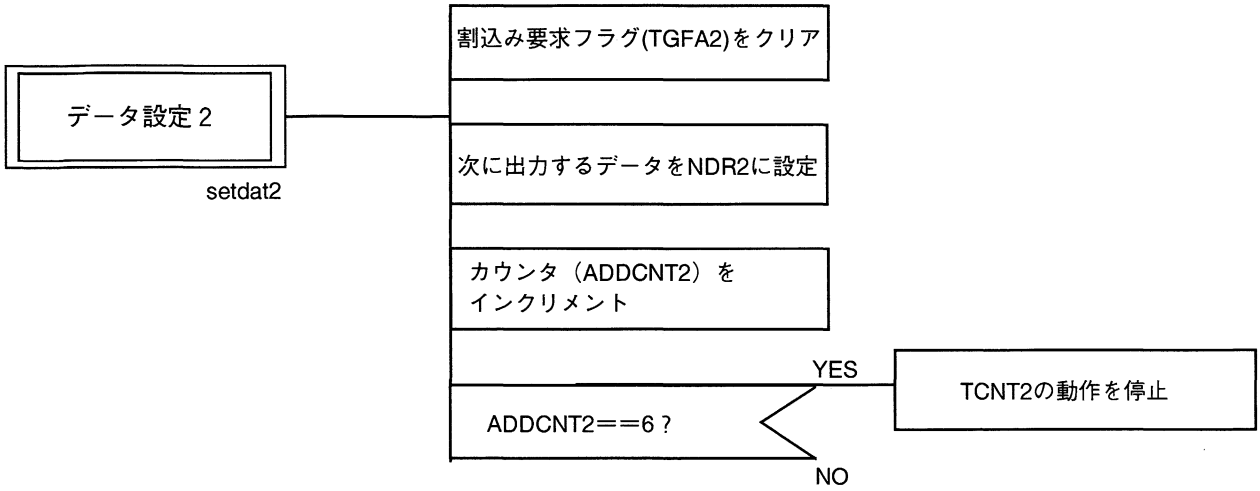
（3）データ設定1



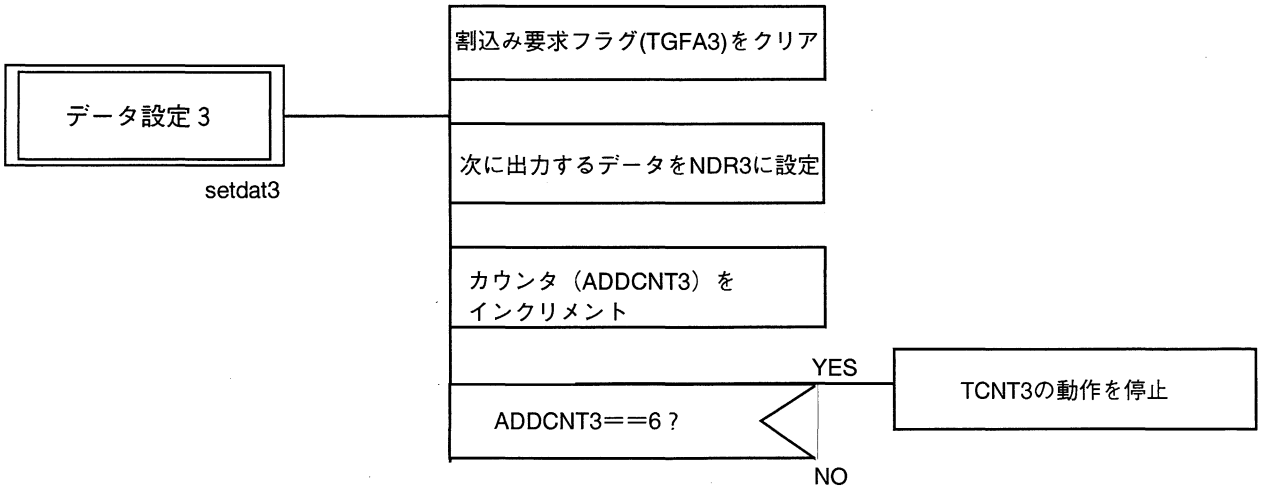
4ビット×4系統出力	MCU	H8S／2655	使用機能	PPG

PAD

（5）データ設定2



（6）データ設定3



プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTOCOL                     */
/*****************************************/
void ppgl6mn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
# define ndat_tab0 ((unsigned char * ) 0xffec00)        /* Outout data table */
# define addcnt0  (*(unsigned char * ) 0xffec05)        /* Group0 transmit counter */
# define ndat_tabl ((unsigned char * ) 0xffec06)        /* Outout data table */
# define addcntl  (*(unsigned char * ) 0xffec0b)        /* Groupl transmit counter */
# define ndat_tab2 ((unsigned char * ) 0xffec0c)        /* Outout data table */
# define addcnt2  (*(unsigned char * ) 0xffecll)        /* Group2 transmit counter */
# define ndat_tab3 ((unsigned char * ) 0xffecl2)        /* Outout data table */
# define addcnt3  (*(unsigned char * ) 0xffecl7)        /* Group3 transmit counter */

/*****************************************/
/*     MAIN PROGRAM : ppgl6mn            */
/*****************************************/
void ppgl6mn(void)
{
        MSTPCR = 0x17ff;        /* Disable module(PPG,TPU) stop mode */

        addcnt0 = 0;            /* Transmit counter clear */
        addcnt1 = 0;
        addcnt2 = 0;
        addcnt3 = 0;

        TIOR0H = 0x00;          /* Initialize TIOR0H */
        TIOR1H = 0x00;          /* Initialize TIOR1H */
        TIOR2H = 0x00;          /* Initialize TIOR2H */
        TIOR3H = 0x00;          /* Initialize TIOR3H */

        TGR0A = 0x00c8;         /* Set non overlap time */
        TGR0B = 0x1388;         /* Set PPG output cycle */
        TGR1A = 0x00c8;         /* Set non overlap time */
        TGR1B = 0x2710;         /* Set PPG output cycle */
        TGR2A = 0x00c8;         /* Set non overlap time */
        TGR2B = 0x3a98;         /* Set PPG output cycle */
        TGR3A = 0x00c8;         /* Set non overlap time */
        TGR3B = 0x4e20;         /* Set PPG output cycle */

        TPU_TCR0 = 0x40;        /* Initialize TCR0 */
        TPU_TCR1 = 0x40;        /* Initialize TCR1 */
        TCR2 = 0x40;            /* Initialize TCR2 */
        TCR3 = 0x40;            /* Initialize TCR3 */

        PODRH = 0;              /* Set first output data */
        PODRL = 0;              /* Set first output data */

        P1DDR = 0xff;           /* Port1: output */
        P2DDR = 0xff;           /* Port2: output */
        NDERH = 0xff;           /* Enable transmit data */
        NDERL = 0xff;           /* Enable transmit data */

        PCR = 0xe4;             /* Set output toriga */
        PMR = 0xff;             /* Set non overlap mode */
```

プログラムリスト

```
        NDR0 = ndat_tab0[addcnt0++];    /* Set second output data */
        NDR1 = ndat_tab1[addcnt1++];    /* Set second output data */
        NDR2 = ndat_tab2[addcnt2++];    /* Set second output data */
        NDR3 = ndat_tab3[addcnt3++];    /* Set second output data */

        TIER0 = 0x41;                   /* Initialize TIER0 */
        TIER1 = 0x41;                   /* Initialize TIER1 */
        TIER2 = 0x41;                   /* Initialize TIER2 */
        TIER3 = 0x41;                   /* Initialize TIER3 */
        set_imask_ccr(0);               /* Enable interrupt */
        TSTR = 0x0f;                    /* Start TCNT0-3 */
        while(1);                       /* Loop */
}
/*********************************/
/*   INTERRUPT PROGRAM: setdat0   */
/*********************************/
#pragma interrupt (setdat0)
void setdat0(void)
{
        TSR0_BP.TGFA0 = 0;              /* Clear TGFA0 request*/
        NDR0 = ndat_tab0[addcnt0++];    /* Set next output data */
          if(addcnt0 == 6)              /* All output end ? */
          TSTR_BP.CST0 = 0;             /* Stop TCNT0 */
}

/*********************************/
/*   INTERRUPT PROGRAM: setdat1   */
/*********************************/
#pragma interrupt (setdat1)
void setdat1(void)
{
        TSR1_BP.TGFA1 = 0;              /* Clear TGFA1 request*/
        NDR1 = ndat_tab1[addcnt1++];    /* Set next output data */
          if(addcnt1 == 6)              /* All output end ? */
          TSTR_BP.CST1 = 0;             /* Stop TCNT1 */
}

/*********************************/
/*   INTERRUPT PROGRAM: setdat2   */
/*********************************/
#pragma interrupt (setdat2)
void setdat2(void)
{
        TSR2_BP.TGFA2 = 0;              /* Clear TGFA2 request */
        NDR2 = ndat_tab2[addcnt2++];    /* Next output data */
          if(addcnt2 == 6)              /* All output end ? */
          TSTR_BP.CST2 = 0;             /* Stop TCNT2 */
}

/*********************************/
/*   INTERRUPT PROGRAM: setdat3   */
/*********************************/
#pragma interrupt (setdat3)
void setdat3(void)
{
        TSR3_BP.TGFA3 = 0;              /* Clear TGFA3 request */
        NDR3 = ndat_tab3[addcnt3++];    /* Set next output data */
          if(addcnt3 == 6)              /* All output end ? */
          TSTR_BP.CST3 = 0;             /* Stop TCNT3 */
}
```

３．９　調歩同期式ＳＣＩ

調歩同期式ＳＣＩ	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

仕様

（１）図１に示すようにH8S/2655とH8/3314間で、１バイトのデータを調歩同期式で送受信します。
（２）転送フォーマットは９６００ＢＰＳ、８ビットデータ、１ストップビットおよびノンパリティーです。
（３）ＲＴＳ、ＣＴＳによる通信制御を行います。

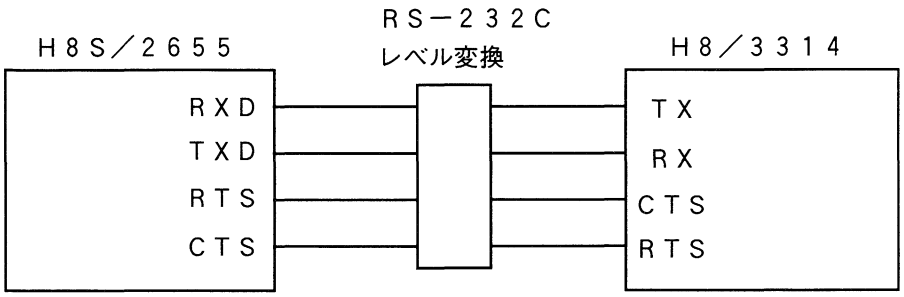


図１　Ｈ８Ｓ／２６５５による調歩同期式ＳＣＩブロック図

使用機能説明

（１）本タスク例ではデータの送受信をＳＣＩ０を使用して行います。また、通信制御端子（ＲＴＳ、ＣＴＳ）はポート３を使用します。
（ａ）図２に本タスク例で使用するＳＣＩの送信ブロック図を示します。
本タスクではＳＣＩの以下の機能を使用してＨ８／３３１４へデータの送信を行います。
・キャラクタ単位で同期をとる調歩同期方式でデータの通信を行う機能。（調歩同期式モード）
・送信完了時に割り込みを発生する機能（ＴＥＩ割り込み）

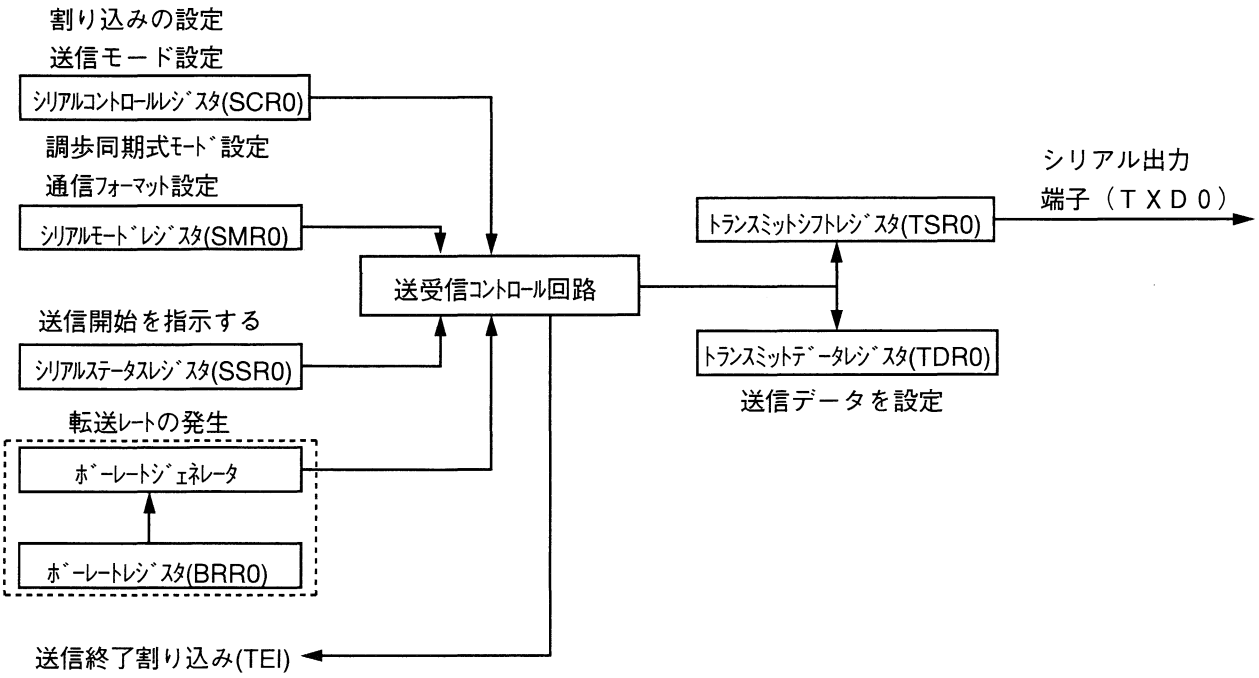


図２　ＳＣＩ送信ブロック図

（ｂ）図３に本タスク例で使用するＳＣＩの受信ブロック図を示します。
本タスクではＳＣＩの以下の機能を使用してＨ８／３３１４からのデータを受信します。

・キャラクタ単位で同期をとる調歩同期方式でデータの通信を行う機能。（調歩同期式モード）
・受信完了時に割り込みを発生する機能（ＲＸＩ割り込み）

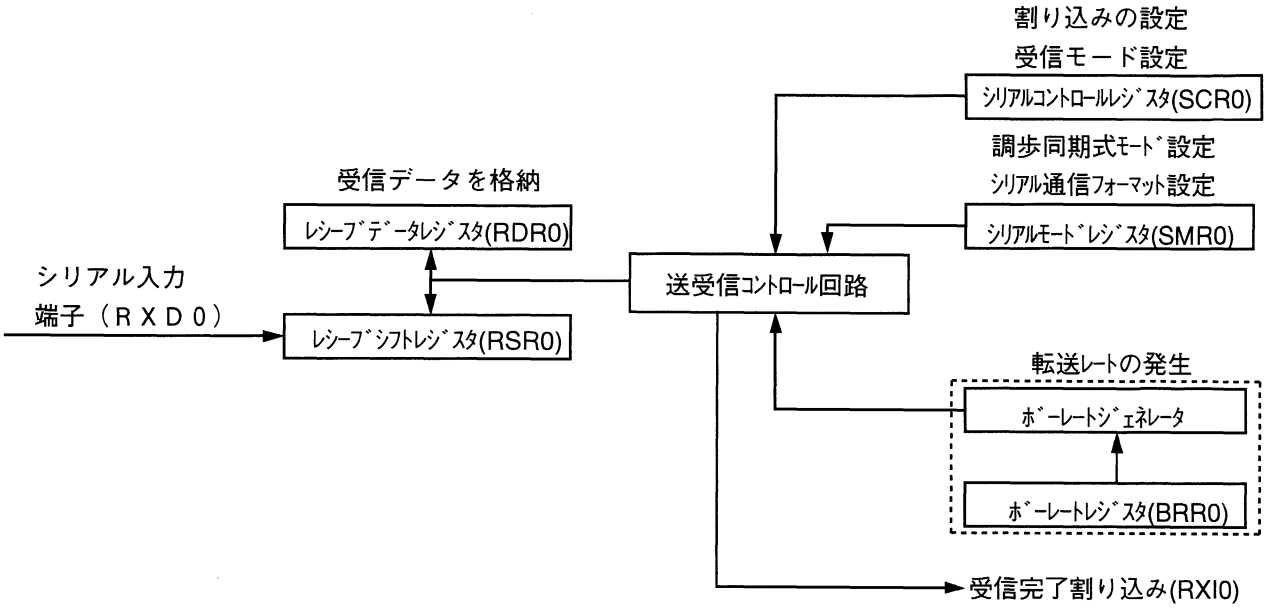


図３　ＳＣＩ受信ブロック図

（２）表１に本タスク例の機能り割り付けを示します。表１に示すようにＨ８Ｓ／２６５５の機能を割り付け、H8/3314とインタフェースを行います。

表１　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能	機能
ＲＸＤ０	コンソールからデータを受信する
ＴＸＤ０	コンソールへデータを送信する
ＳＭＲ０	ＳＣＩを調歩同期式モードに設定し、転送フォーマットを設定する
ＳＣＲ０	送受信割込みを許可し、ＳＣＩを送受信モードに設定する
ＳＳＲ０	ＴＤＲＥにて送信開始を指示する
ＲＤＲ０	コンソールから受信したデータを格納する
ＴＤＲ０	コンソールへ送信するデータを設定する
ＢＲＲ０	転送レートを設定する

調歩同期式ＳＣＩ	MCU	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

動作説明

図４に本タスクの動作原理を示します。図４に示すタイミングでハードウェア処理およびソフトウェア処理を行い、H8/3314とのインタフェースを行います。

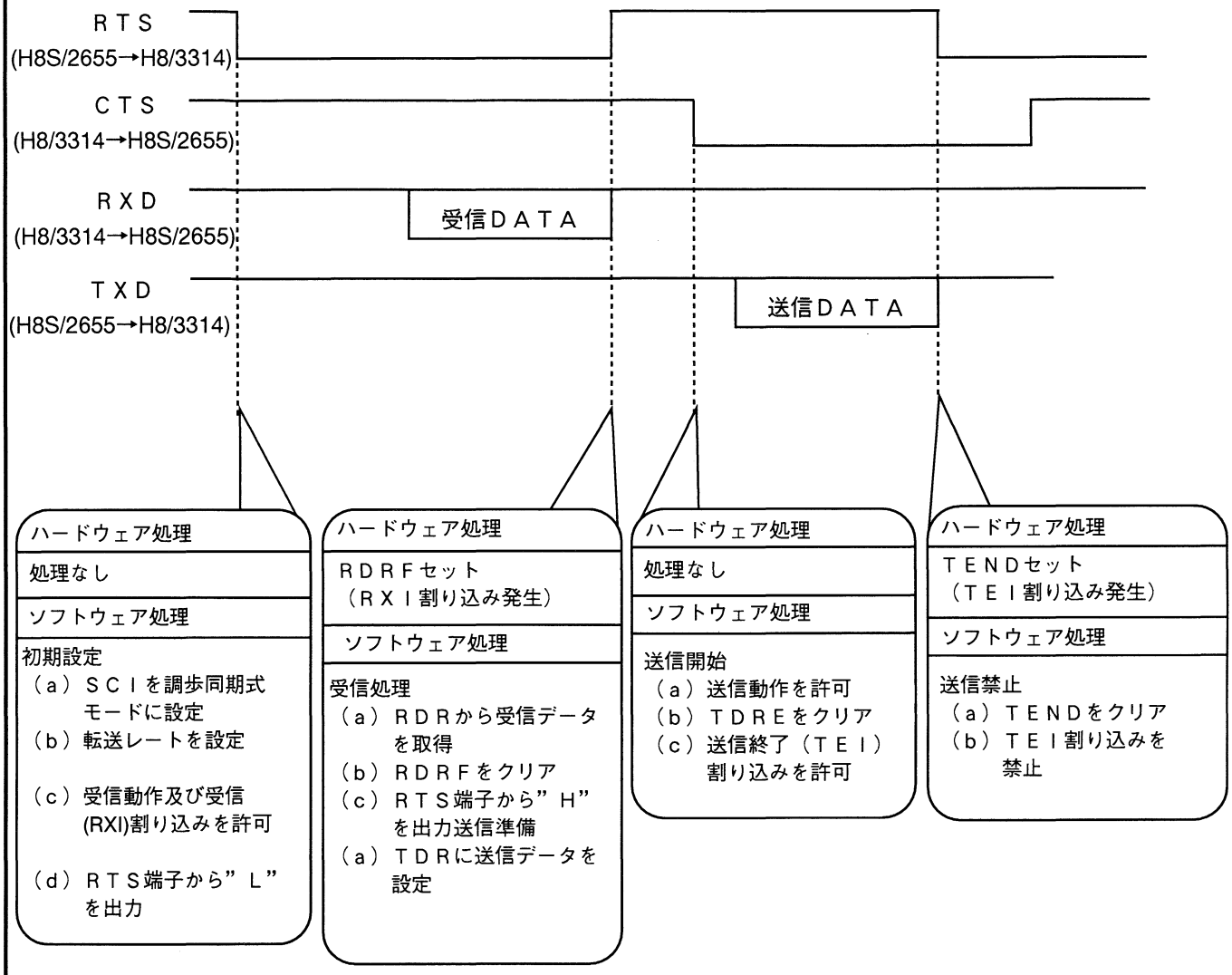


図４　調歩同期式ＳＣＩ動作原理

調歩同期式ＳＣＩ	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

ソフトウェア説明

（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	ASCMN	ＳＣＩの初期設定、および送受信の管理を行う
データ受信完了	ASCRX	ＲＸＩ割り込みで起動し、データの受信を行う
データ送信完了	ASCTE	ＴＥＩ割り込みで起動し、送信が完了したことを通知する

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
ｒｃｖ_ｄａｔａ	コンソールから受信したデータを設定する	unsigned char	データ受信完了	出力
			メインルーチン	入力
ｒｘｅｎｄｆ	受信完了を示すフラグ １：受信完了　　０：受信動作中	unsigned char	データ受信完了	出力
			メインルーチン	入力
ｔｘｅｎｄｆ	送信完了を示すフラグ １：送信完了　　０：送信動作中	unsigned char	データ送信完了	出力
			メインルーチン	入力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
ＳＭＲ０	ＳＣＩのモード（調歩同期式）、転送フォーマットおよびボーレートジェネレータへのクロック選択（φクロック入力）を設定する	メインルーチン
ＳＣＲ０	割り込み（ＲＸＩ、ＴＥＩ）を許可し、ＳＣＩの送受信の許可を設定する	メインルーチン
ＳＳＲ０	ＴＤＲＥ（ｂ７）をクリアすることにより、送信開始を指示する	メインルーチン
ＲＤＲ０	コンソールから受信したデータを設定する	データ受信完了
ＴＤＲ０	コンソールへ送信するデータを設定する	メインルーチン
ＢＲＲ０	転送レートを設定する	メインルーチン
Ｐ３ＤＤＲ	ポート３の入出力を設定する	メインルーチン
Ｐ３ＤＲ	ＲＴＳ端子およびＣＴＳ端子の操作を行う	メインルーチン
ＭＳＴＰＣＲ	ＳＣＩのモジュールストップモードを解除する	メインルーチン

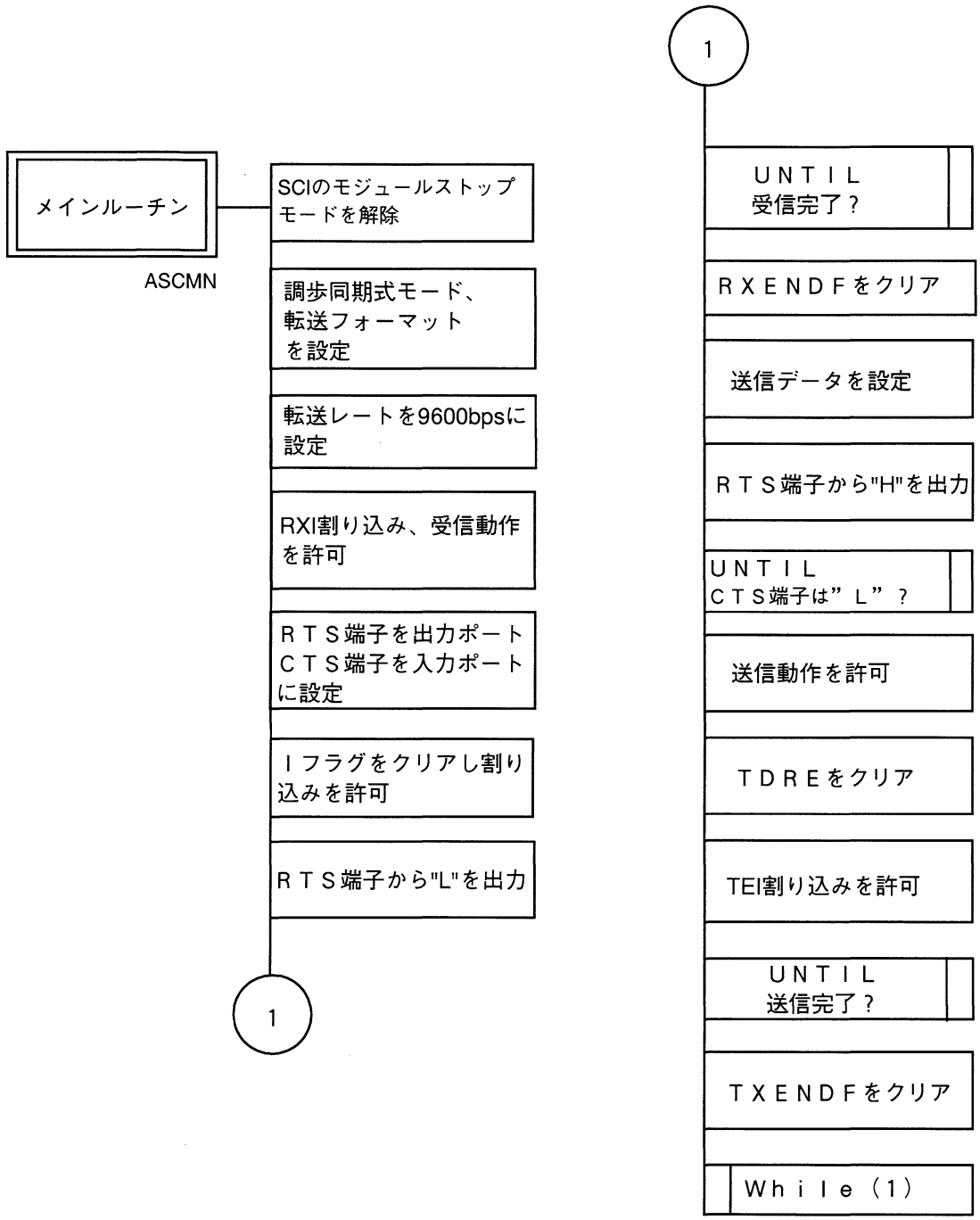
（4）使用ＲＡＭ説明

本タスク例では引数以外のＲＡＭは使用しません。

調歩同期式ＳＣＩ	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

ＰＡＤ

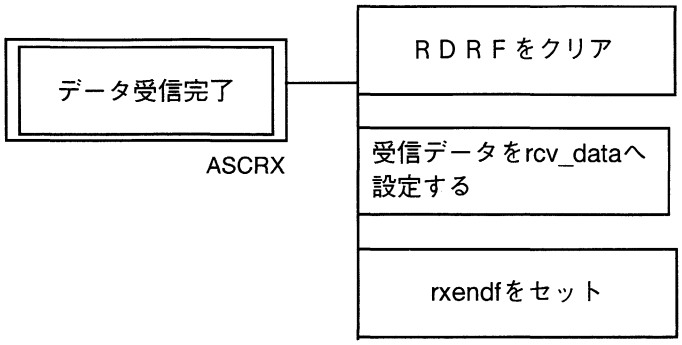
（１）メインルーチン



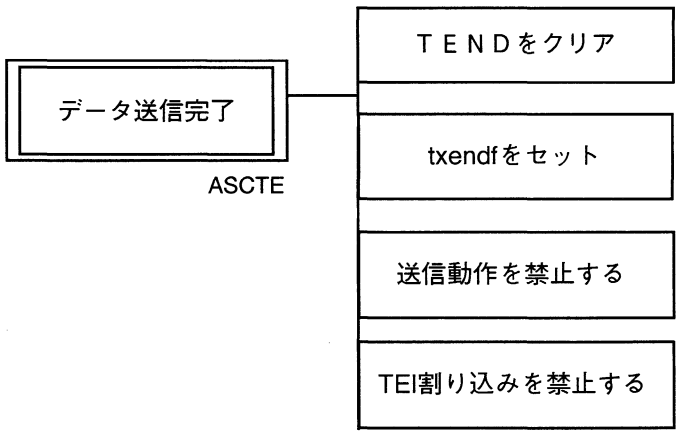
調歩同期式ＳＣＩ	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

ＰＡＤ

（２）データ受信完了



（３）データ送信完了



調歩同期式ＳＣＩ	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/**********************************/
/*          PROTOCOL              */
/**********************************/
void ASCMN(void);
#pragma interrupt (ASCRX)
#pragma interrupt (ASCTE)

/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/

# define rcv_data  (*(unsigned char * )0xffec00)  /* Receive data from console */
# define rxendf    (*(unsigned char * )0xffec01)  /* Receive end flag */
# define txendf    (*(unsigned char * )0xffec02)  /* Transmit end flag */

/**********************************/
/*     MAIN PROGRAM: ASCMN        */
/**********************************/
void ASCMN(void)
        {
        MSTPCR = 0xffdf;                /* Disable module(SCI0) stop mode */
        SMR0 = 0;                       /* Initialize SMR0 */
        BRR0 = 64;                      /* Set 9600bps */
        SCR0 = 0x50;                    /* Enable RXI interrupt */
        P3DDR = 0xC9;                   /* RTS(P33):output port CTS(P31):input port */
        P3DR = 0xFF;

        set_imask_ccr(0);               /* Enable interrupt */
        P3DR_BP.P33DR=0;                /* Output RTS low */

        while (rxendf == 0);            /* Receive complete ? */
        rxendf = 0;
        TDR0 = rcv_data;                /* Set transmit data (receive data) */
        P3DR_BP. P33DR=1;               /* Output RTS high */

        while (PORT3_BP. P31== 1);

        SCR0_BP. TE0 = 1;               /* Enable transmit */
        SSR0_BP. TDRE0 = 0;             /* Start transmit */
        SCR0_BP. TEIE0 = 1;             /* Enable TEI interrupt */

        while (txendf==0);              /* Transmit complete? */
        txendf = 0;

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

調歩同期式ＳＣＩ	MCU	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ (調歩同期式モード)

プログラムリスト

```
/**********************************/
/*   INTERRUPT PROGRAM: ASCRX     */
/**********************************/
void ASCRX(void)
        {
        SSR0_BP. RDRF0 = 0;             /* Clear RDRF */
        rcv_data = RDR0;                /* Get receive data from RDR0 */
        rxendf = 1;                     /* Set rxendf */
        }

/**********************************/
/*   INTERRUPT PROGRAM: ASCTE     */
/**********************************/
void ASCTE(void)
        {
        SSR0_BP. TDRE0 = 0;             /* TEND clear */
        txendf = 1;                     /* Set txendf */
        SCR0_BP. TE0 = 0;               /* Disable transmit */
        SCR0_BP. TEIE0 = 0;             /* Disable TEI interrupt */
        }
```

３．１０　同時送受信動作

同時送受信動作	MCU	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ（同時送受信）

仕様

（１）図１に示すように、Ｈ８Ｓ／２６５５とＨ８／３３１４との間で１バイトのデータの送受信を行います。
（２）データの送受信はクロック同期式フォーマットで行います。クロックはマスタ側のＨ８Ｓ／２６５５が、スレーブ側のＨ８／３３１４に供給します。
（３）Ｈ８Ｓ／２６５５はデータの送信と同時にＨ８／３３１４からのデータを受信します。

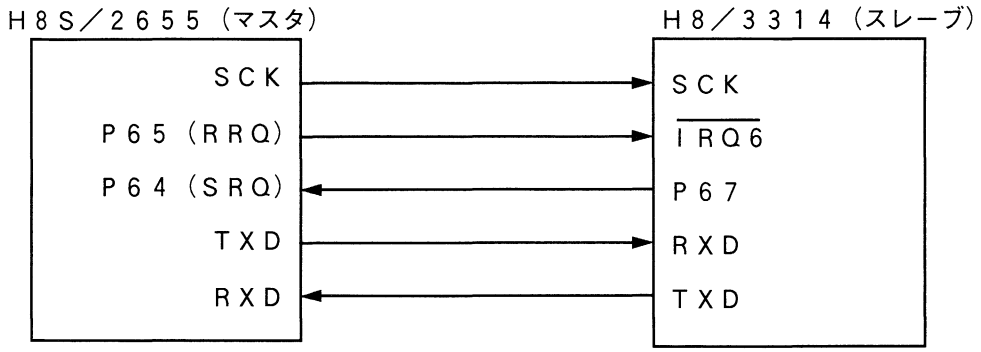


図１　Ｈ８Ｓ／２６５５によるクロック同期式ＳＣＩブロック図

使用機能説明

（１）本タスク例ではデータの送受信をＳＣＩ１を使用して行います。また、通信制御端子（ＲＲＱ、ＳＲＱ）はポート６を使用します。
（ａ）図２に本タスク例で使用するＳＣＩのブロック図を示します。本タスク例では、ＳＣＩの以下の機能を使用して送受信動作を同時に行います。
・クロックに同期してシリアルデータの通信を行う機能。（クロック同期式モード）
・送信と受信を同時に行う機能。（同時送受信動作）
・受信完了時に割り込みを発生する機能（ＲＸＩ割り込み）

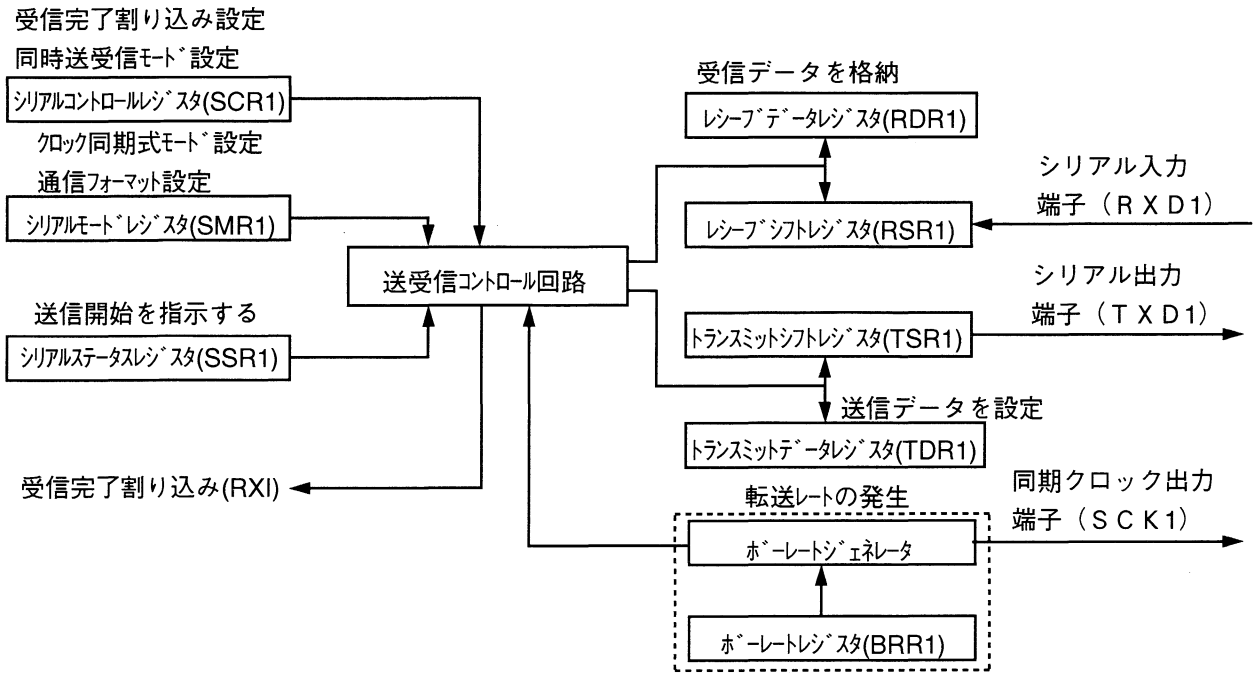


図２　ＳＣＩブロック図

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、H8／3314とデータの同時送受信を行います。

表1　H8S／2655機能割り付け

H8S/2655機能	機能
SCK1	同期クロックを送信する
RXD1	H8／3314からデータを受信する
TXD1	H8／3314へデータを送信する
SMR1	SCIをクロック同期式モードに設定する
SCR1	受信割込みを許可し、SCIを送受信モードに設定する
SSR1	TDREビットにより送信開始を指示する
RDR1	H8／3314から受信したデータを設定する
TDR1	H8／3314へ送信するデータを設定する
BRR1	転送レートを設定する
P6DDR	ポート6の入出力を設定する
P6DR	RRQの送信およびSRQの受信を行う

同時送受信動作	MCU	H8S／2655	使用機能	SCI（同時送受信）

動作説明

図3に本タスクの動作原理を示します。図3に示すタイミングでハードウェア処理およびソフトウェア処理を行い、H8／3314とインタフェースを行います。

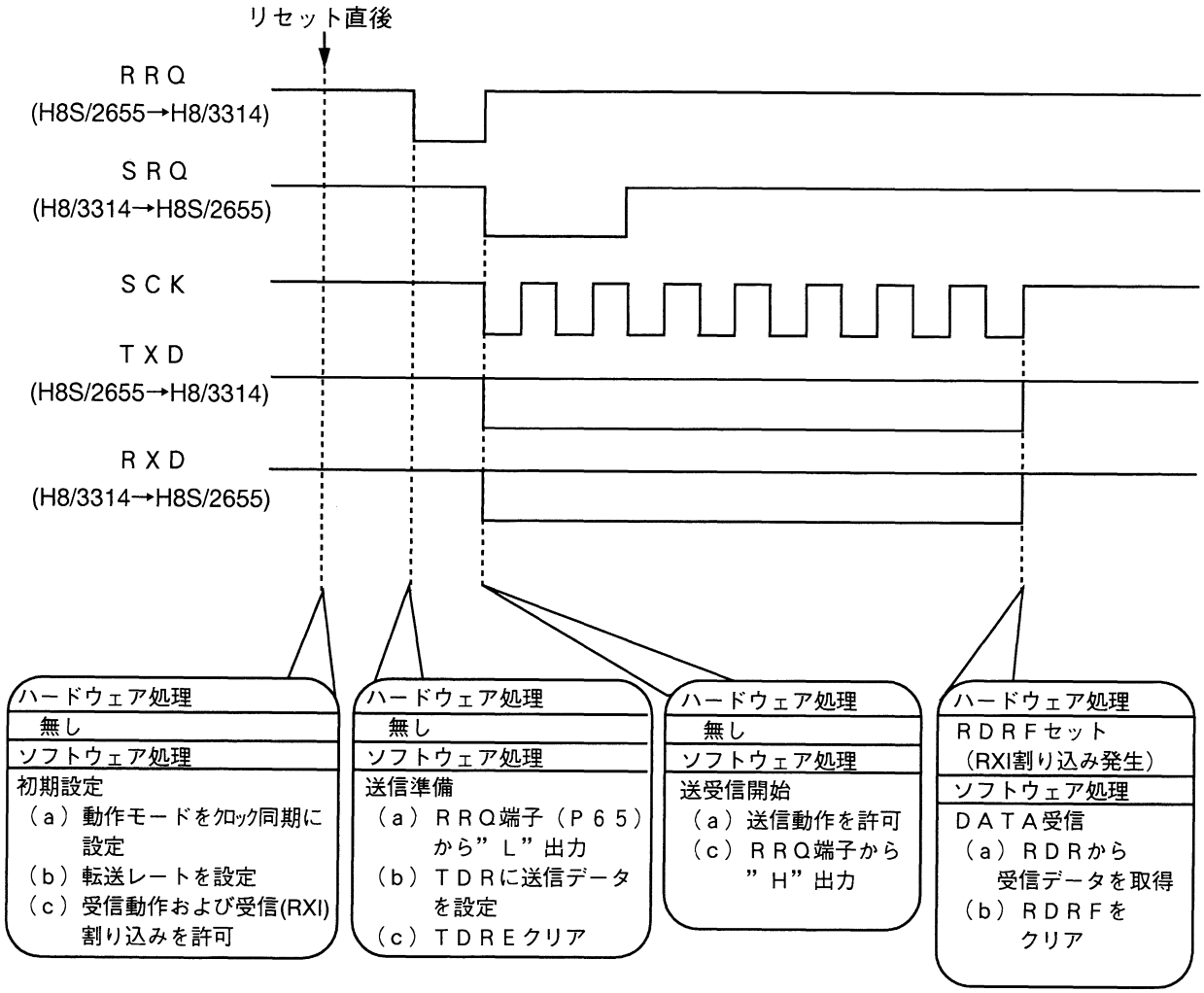


図3　同時送受信動作原理

同時送受信動作	MCU	H8S／2655	使用機能	SCI（同時送受信）

ソフトウェア説明

（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	simtrmn	SCIの初期設定、および送受信の管理を行う
データ受信完了	rxend	RXI割り込みで起動し、データの受信を行う

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
revend	受信完了を示すフラグ 1：受信完了　　0：受信動作中	unsigned char	データ受信完了	出力
			メインルーチン	入力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
SMR1	SCIのモード（ｸﾛｯｸ同期式）、転送フォーマットおよびボーレートジェネレータへのクロック選択（φクロック入力）を設定する	メインルーチン
SCR1	割り込み（RXI）を許可し、SCIの送受信の許可を設定する	メインルーチン
SSR1	TDREをクリアすることにより、送信開始を指示する	メインルーチン
RDR1	H8/3314から受信したデータを設定する	データ受信完了
TDR1	H8/3314へ送信するデータを設定する	メインルーチン
BRR1	転送レートを設定する	メインルーチン
P6DDR	ポート6の入出力を設定する	メインルーチン
P6DR	RRQ端子およびSRQ端子の操作を行う	メインルーチン
MSTPCR	SCIのモジュールストップモードを解除する	メインルーチン

（4）使用ＲＡＭ説明

ラベル名	機能	データ長	使用モジュール名
rvdata	受信したデータを設定する	unsigned char	メインルーチン
trdata	送信するデータを設定する	unsigned char	メインルーチン

同時送受信動作	MCU	H8S／2655	使用機能	SCI（同時送受信）

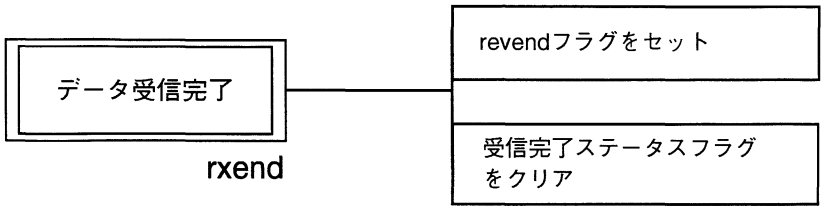
PAD

（1）メインルーチン

同時送受信動作	MCU	H8S／2655	使用機能	SCI（同時送受信）

PAD

（2）データ受信完了



同時送受信動作	MCU	H8S／2655	使用機能	SCI（同時送受信）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void simtrmn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define trdata (*(unsigned char *)0xffec00)
#define rvdata (*(unsigned char *)0xffec01)
#define revend (*(unsigned char *)0xffec02)

/*****************************************/
/*     MAIN PROGRAM : simtrmn            */
/*****************************************/
void simtrmn(void)
{
        MSTPCR = 0xffbf;
        SCR1 = 0x00;                    /* select clock mode */
        SMR1 = 0x80;                    /* init SCI1 */
        BRR1 = 0x04;                    /* set 1MBPS */
        P6DDR = 0x20;                   /* init port */
        P6DR = 0x20;
        SCR1 = 0x70;                    /* enable RX,TX,RIE */
        set_imask_ccr(0);

        P6DR_BP.RRQ = 0;                /* "Low"-> RRQ */

        while(PORT6_BP.SRQ == 1);

        P6DR_BP.RRQ = 1;                /* "High"-> RRQ */
        while(SSR1_BP.TDRE1 == 0);
        TDR1 = trdata;                  /* set transmit data to TDR */
        SSR1 &= 0x7f;                   /* start transmit */

        while(revend == 0);             /* receive complete? */

        rvdata = RDR1;                  /* set receive data */
        SSR1 &= 0xbf;

        while(1);
}

/*****************************************/
/*    NAME : rxend(data receive end)     */
/*****************************************/
#pragma interrupt(rxend)
void rxend(void)
{
        revend = 0x01;
        SSR1 &= 0xbf;
}
```

３．１１　マルチプロセッサ通信

マルチプロセッサ通信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ（マルチプロセッサ通信）

仕様

（１）図１に示すようにＨ８Ｓ／２６５５と２つのＨ８／３３１４間でシリアル通信回線を共有したデータの送受信を行います。

（２）Ｈ８Ｓ／２６５５が送信局の場合、Ｈ８／３３１４へデータを送信し、Ｈ８／３３１４は自局へのデータのみ受信します。受信局の場合、自局のＩＤと比較し一致した場合データを受信します。

（３）データは９６００ＢＰＳ、８ビットデータ、１ストップビットおよびノンパリティーで送受信します。

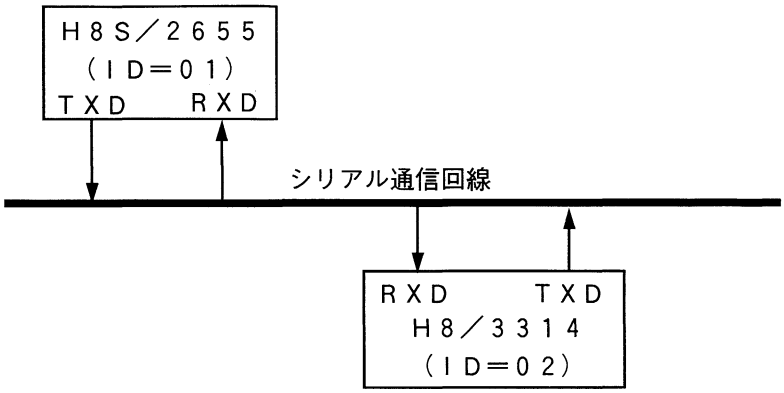


図１　マルチプロセッサ機能を使用した調歩同期式ＳＣＩブロック図

使用機能説明

（１）本タスク例では、ＳＣＩのマルチプロセッサ通信機能を使用して、マルチプロセッサ通信を行います。

（ａ）図２に本タスク例で使用する、送信局ＳＣＩのブロック図を示します。
本タスク例では、ＳＣＩの以下の機能を使用して送信を行います。
・キャラクタ単位で同期をとる調歩同期方式でデータの通信を行う機能。（調歩同期式モード）
・マルチプロセッサビットを付加したフォーマットで通信する機能。（マルチプロセッサ通信機能）
・送信開始時に割り込みを発生する機能。（ＴＸＩ割り込み）

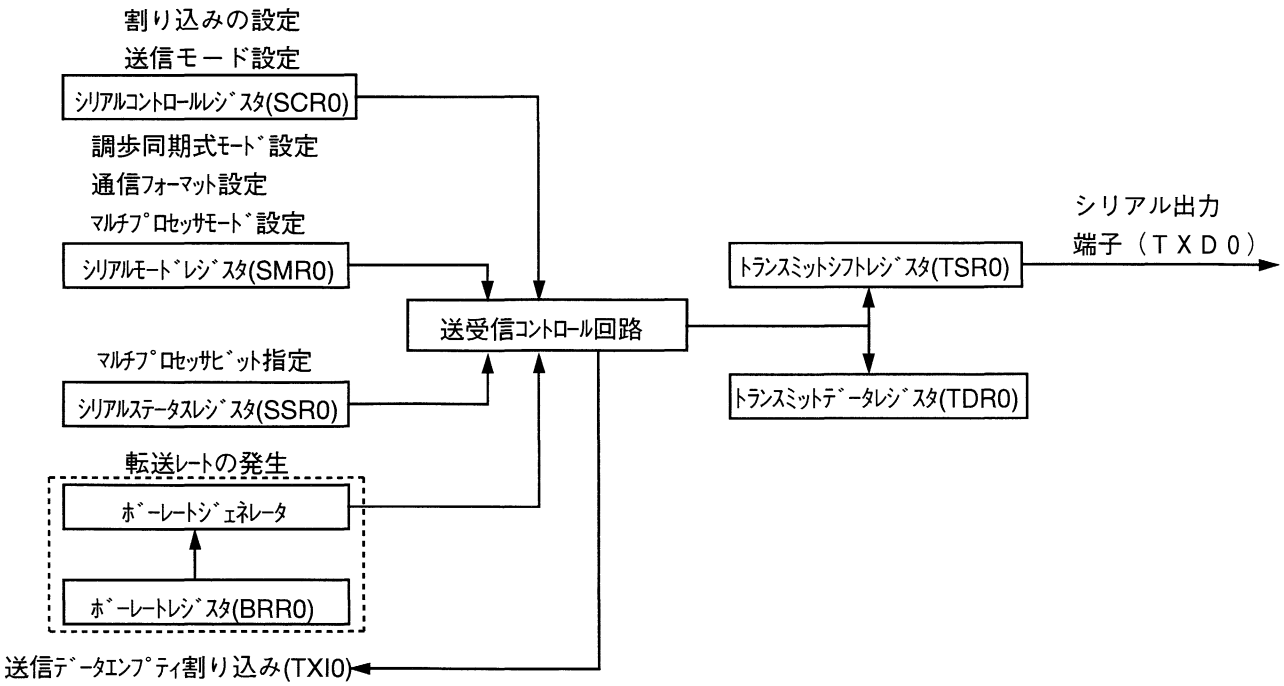


図２　送信局ＳＣＩブロック図

（b）図3に本タスク例で使用する、受信局ＳＣＩのブロック図を示します。本タスク例では、ＳＣＩの以下の機能を使用して受信を行います。

・キャラクタ単位で同期をとる調歩同期方式でデータの通信を行う機能。（調歩同期式モード）
・マルチプロセッサビットを付加したフォーマットで通信する機能。（マルチプロセッサ通信機能）
・マルチプロッセサビット受信時に割り込みを発生する機能。（マルチプロセッサ割り込み）
・受信完了時に割り込みを発生する機能。（ＲＸＩ割り込み）

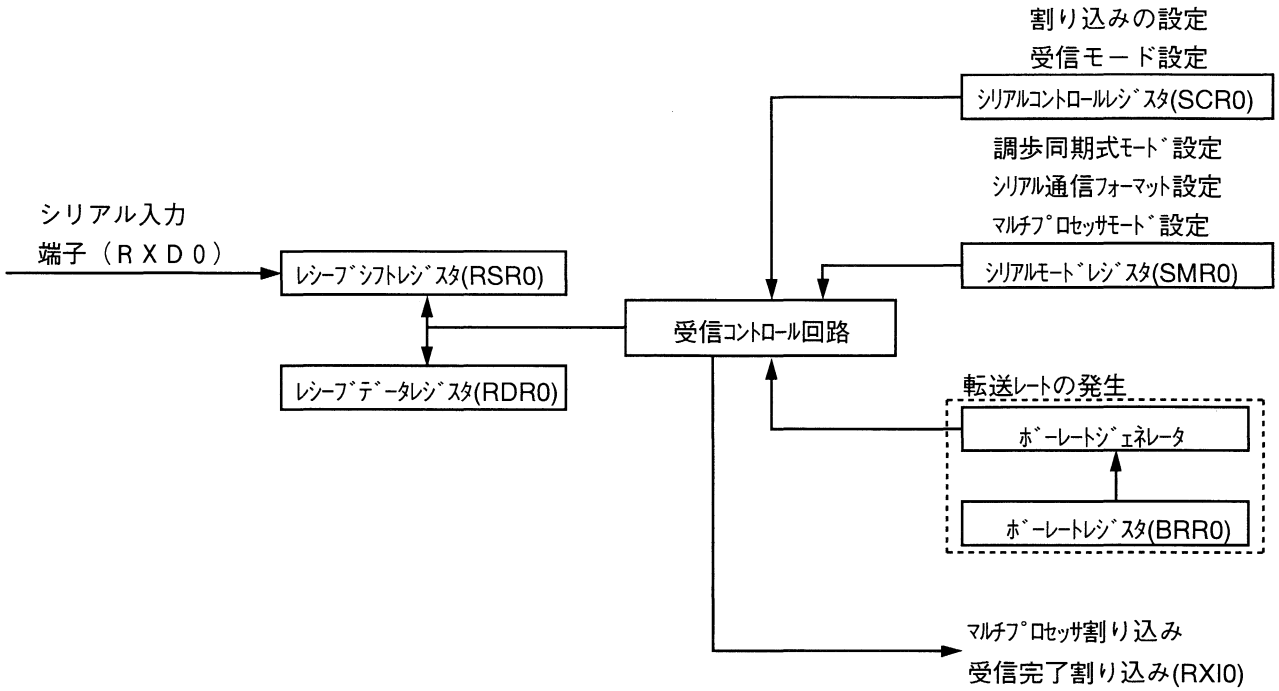


図３　受信局ＳＣＩブロック図

（２）表１に本タスク例の機能割り付けを示します。表１に示すようにＳＣＩの機能を割り付け、マルチプロセッサ通信を行います。

表１　ＳＣＩ機能割り付け

ＳＣＩ機能	機能
ＲＸＤ０	Ｈ８／３３１４からデータを受信する
ＴＸＤ０	Ｈ８／３３１４へデータを送信する
ＳＭＲ０	ＳＣＩを調歩同期式ﾓｰﾄﾞ、ﾏﾙﾁﾌﾟﾛｾｯｻﾓｰﾄﾞに設定する
ＳＣＲ０	送受信割込みを許可し、ＳＣＩを送／受信モードに設定する
ＳＳＲ０	送信を開始する／ﾏﾙﾁﾌﾟﾛｾｯｻﾋﾞｯﾄを設定する
ＲＤＲ０	Ｈ８／３３１４から受信したデータを設定する
ＴＤＲ０	Ｈ８／３３１４へ送信するデータを設定する
ＢＲＲ０	転送レートを設定する

（1）送信動作

図4に本タスクの送信動作原理を示します。図4に示すタイミングでハードウェア処理およびソフトウェア処理を行い、受信局のH8／3314へデータを送信します。

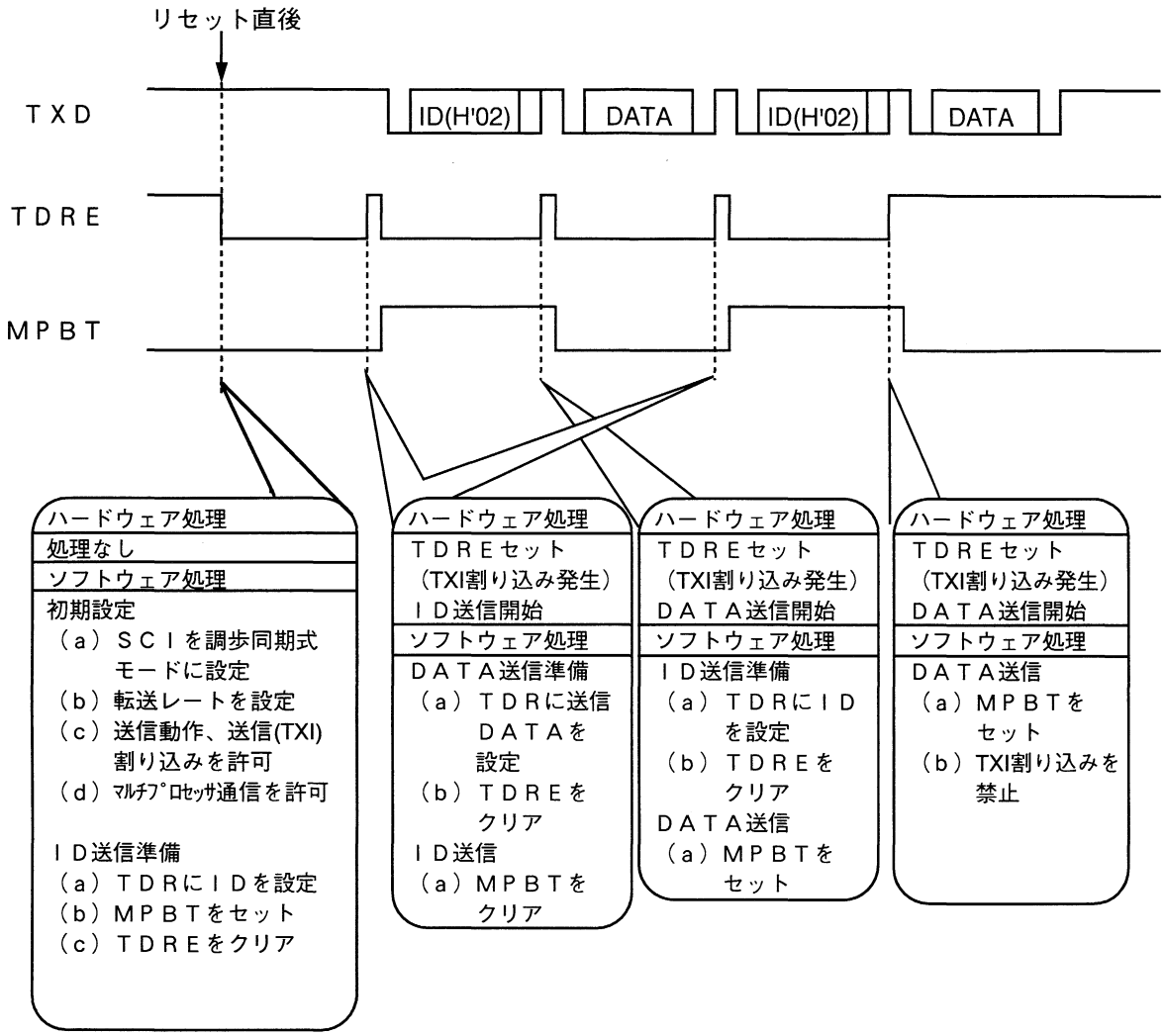
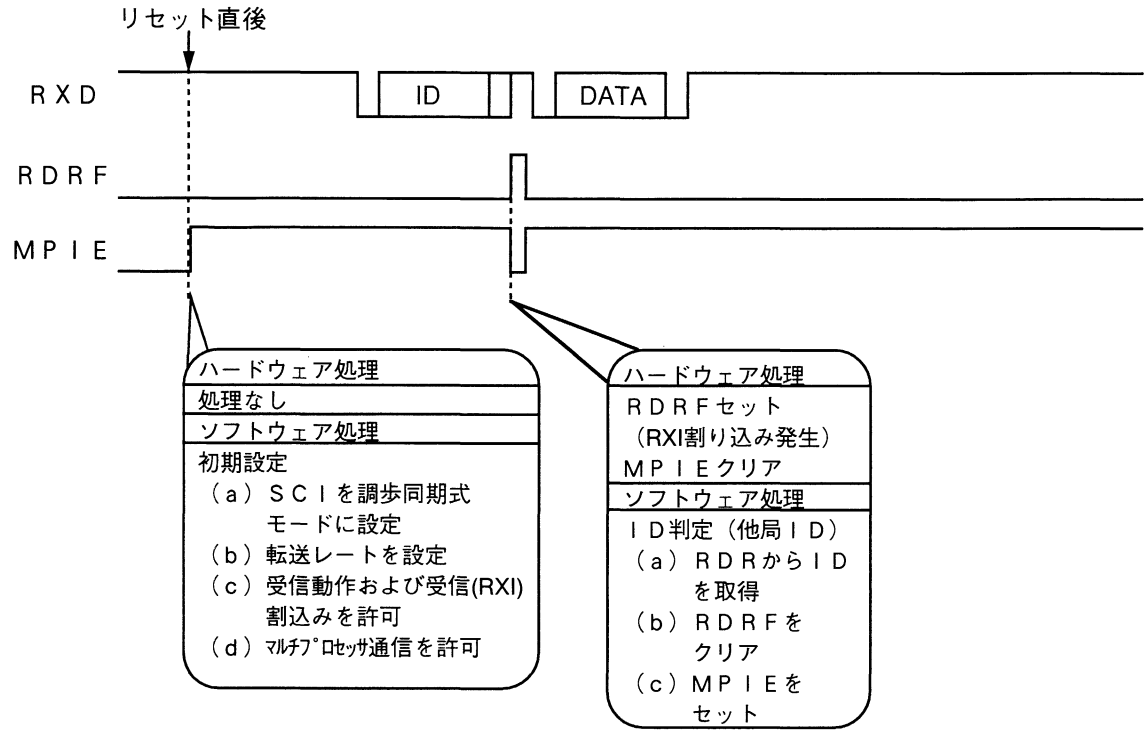


図4　マルチプロセッサ通信（送信局）動作原理

（2）受信動作

図5に本タスクの受信動作原理を示します。図5に示すタイミングでハードウェア処理およびソフトウェア処理を行い、送信局からのデータを受信します。

（a）他局ID受信時動作



（b）自局ID受信時動作

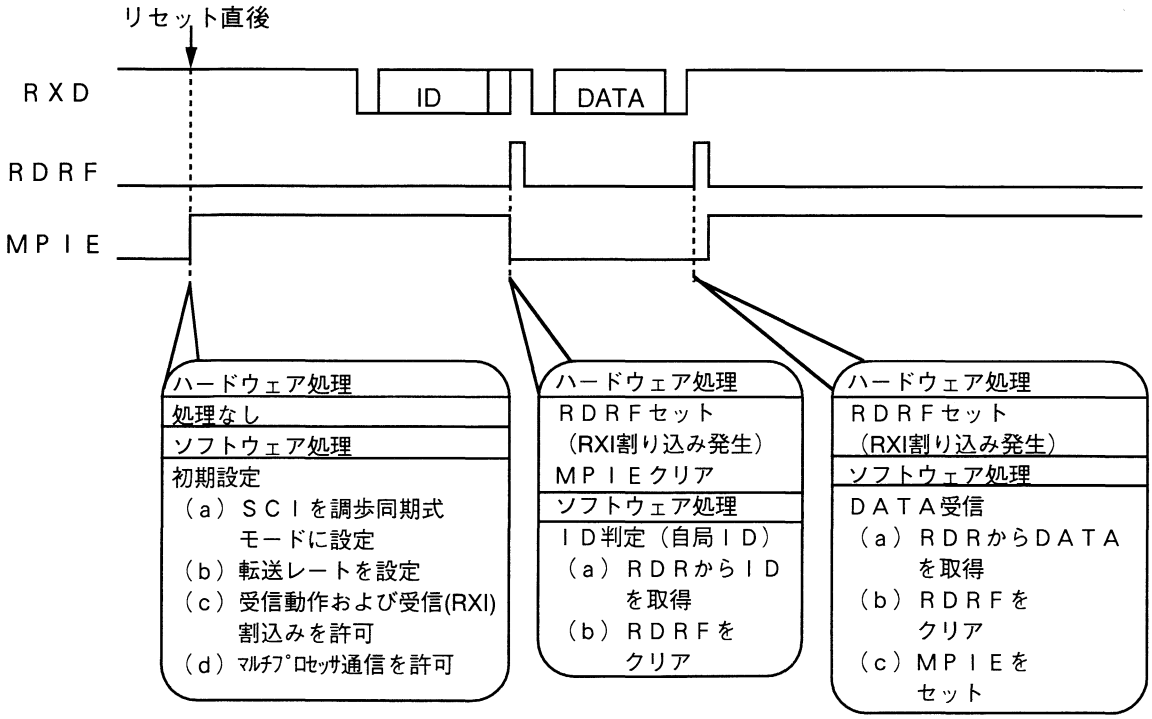


図5　マルチプロセッサ通信（受信局）動作原理

（1）送信局ソフトウェア説明

（a）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	MPMASMN	SCIの初期設定を行う
データ送信	MPSCITX	TXI割り込みで起動し、IDおよびデータの送信を行う

（b）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
txdata	受信局H8／3314へ送信するID、データを格納するバッファ	unsigned char	メインルーチン	出力
			データ送信	入力
txendf	送信終了を示す 1：送信終了　　0：送信中	unsigned char	メインルーチン	入力
			データ送信	出力

（c）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
SMR0	SCIのモード（調歩同期式）、転送フォーマットおよびボーレートジェネレータへのクロック選択（φクロック入力）を設定する	メインルーチン
SCR0	割り込み（TXI）を許可し、SCIの送信許可／禁止を設定する	メインルーチン
		送信終了
SSR0	TDRE（b7）をクリアすることにより、送信開始を指示する	メインルーチン
		データ送信
TDR0	受信局H8／3314へ送信するID、データを設定する	メインルーチン
		データ送信
BRR0	転送レートを設定する	メインルーチン
MSTPCR	SCIのモジュールストップモードを解除する	メインルーチン

（d）使用RAM説明

ラベル名	使用モジュール名	データ長	機能
txcnt	データ送信	unsigned char	送信したデータ数をカウントする

（2）受信局ソフトウェア説明

（a）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	MPSRVMN	SCIの初期設定を行う
データ受信	MPSCIRX	RXI割り込みで起動し、IDおよびデータの受信を行う

（b）引き数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
rcv_data	受信したID、データを設定する	unsigned char	データ受信	出力
			メインルーチン	入力
idrcvf	自局ID受信を示すフラグ 1：ID受信あり　0：ID受信なし	unsigned char	データ受信	出力
			メインルーチン	入力
dtrcvf	データ受信を示すフラグ 1：ﾃﾞｰﾀ受信あり　0：ﾃﾞｰﾀ受信なし	unsigned char	データ受信	出力
			メインルーチン	入力

（c）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
SMR0	SCIのモード（調歩同期式）、転送フォーマットおよびボーレートジェネレータへのクロック選択（φクロック入力）を設定する	メインルーチン
SCR0	割り込み（RXI）を許可し、SCIの受信許可を設定する	メインルーチン
RDR0	送信局H8／3314から受信したID、データを設定する	データ受信
BRR0	転送レートを設定する	メインルーチン
MSTPCR	SCIのモジュールストップモードを解除する	メインルーチン

（d）使用RAM説明

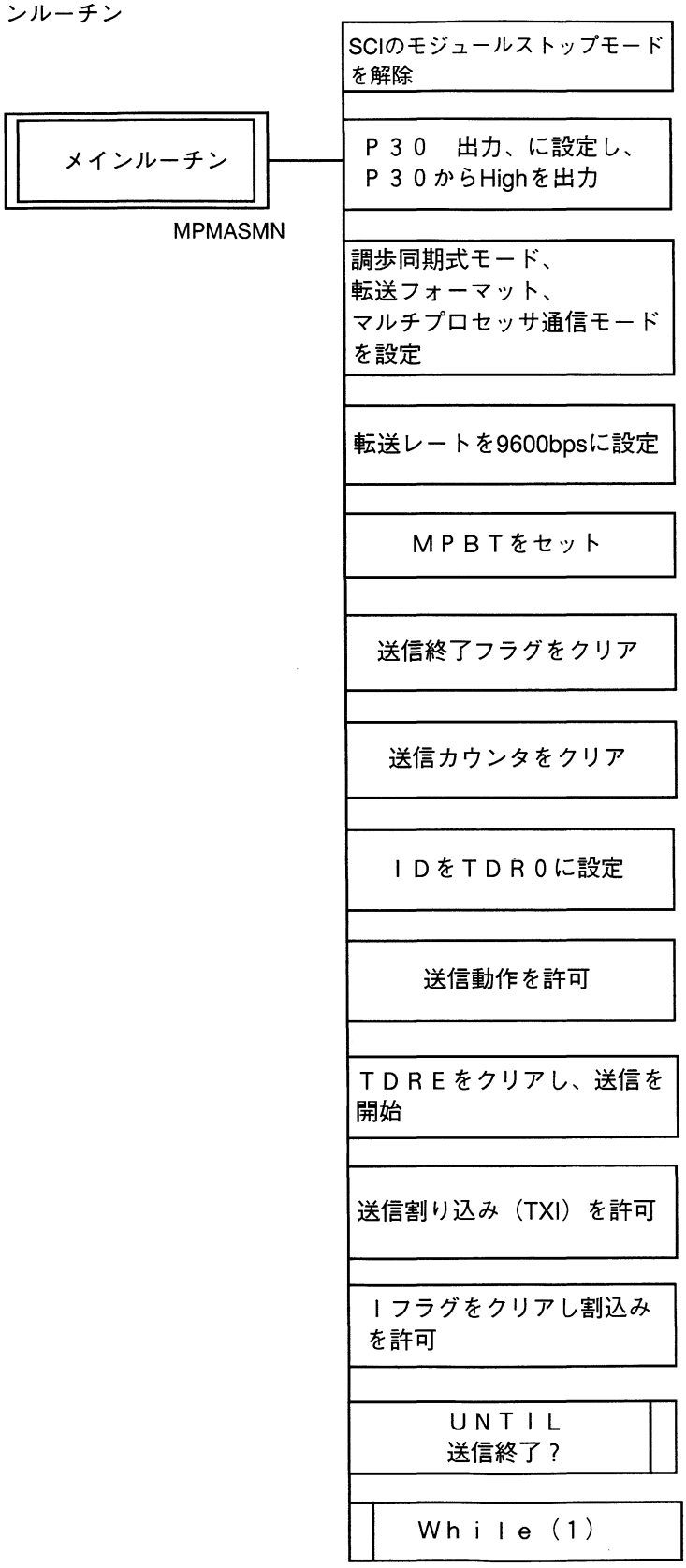
ラベル名	使用モジュール名	データ長	機能
rxid	メインルーチン	unsigned char	受信したIDを設定する
rxdata	メインルーチン	unsigned char	受信したデータを設定する
myid	データ受信	unsigned char	自局IDを設定する

マルチプロセッサ通信	MCU	H8S／2655	使用機能	SCI（マルチプロセッサ通信）

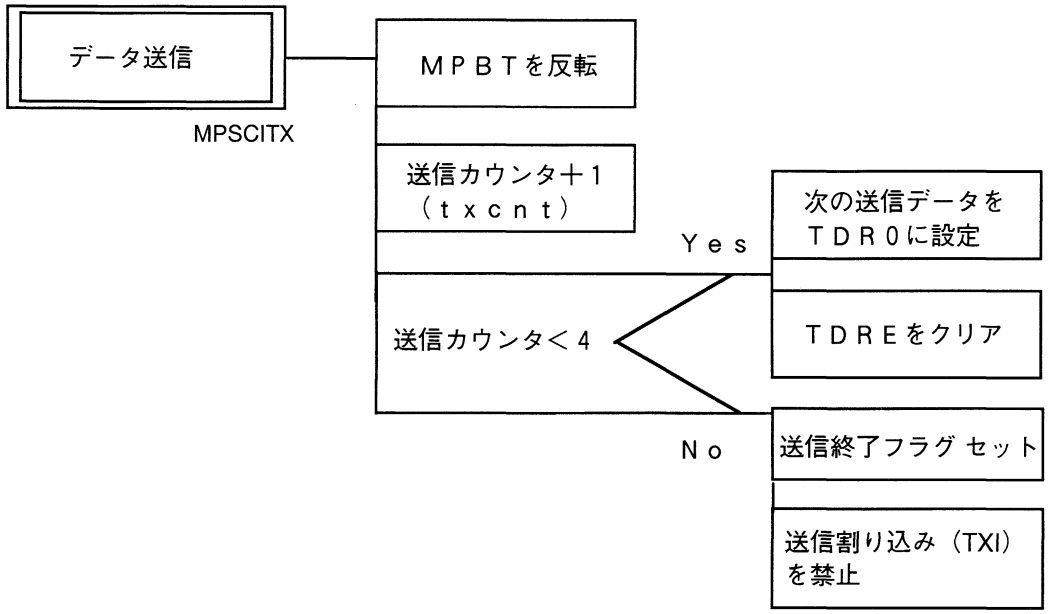
PAD

（1）送信局

（a）メインルーチン



（b）データ送信

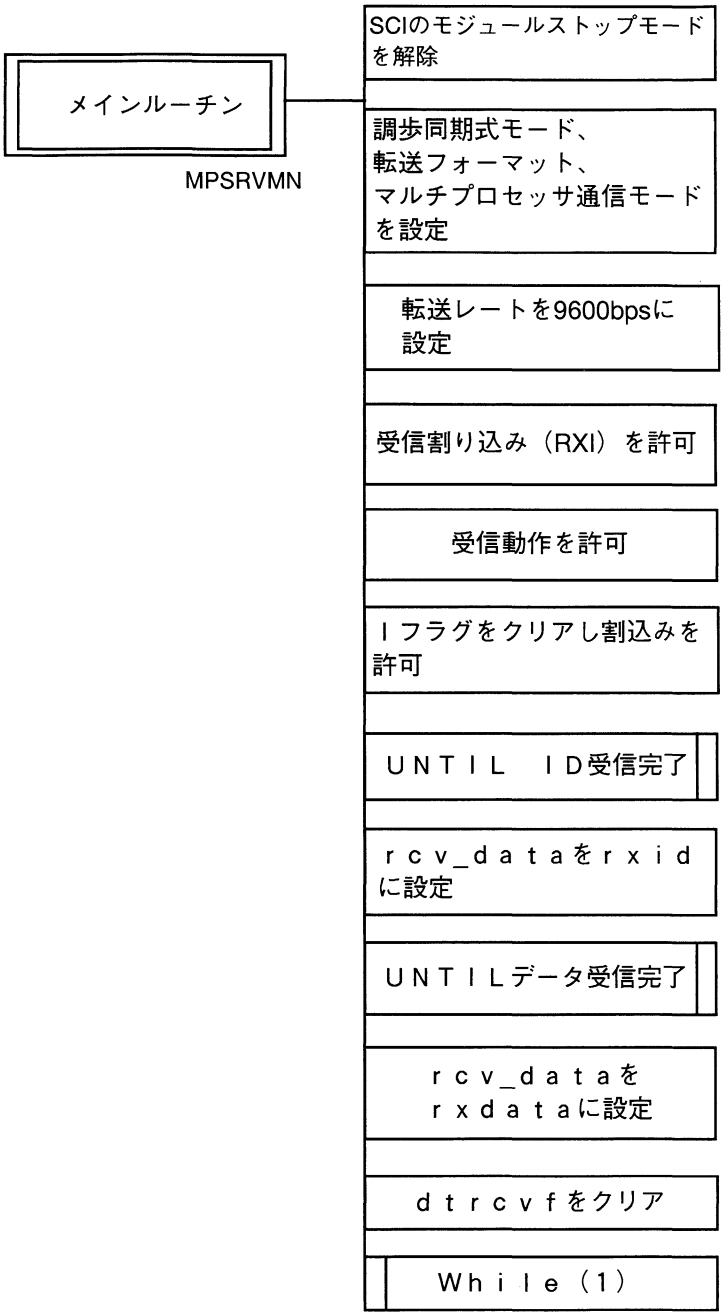

データ送信
MPSCITX
MPBTを反転
送信カウンタ＋1
（txcnt）
Yes
次の送信データを
TDR0に設定
送信カウンタ＜4
TDREをクリア
No
送信終了フラグ セット
送信割り込み（TXI）
を禁止

マルチプロセッサ通信	MCU	H8S／2655	使用機能	SCI（マルチプロセッサ通信）

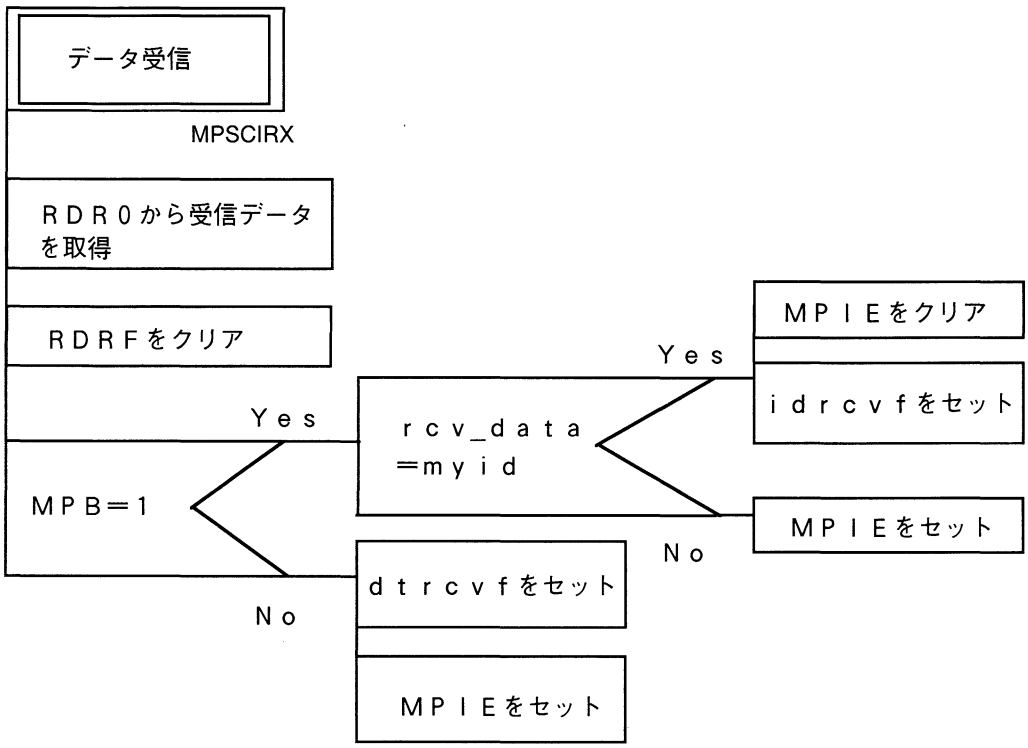
PAD

（2）受信局

（a）メインルーチン



（b）データ受信


データ受信
MPSCIRX
RDR0から受信データを取得
RDRFをクリア
Yes
MPB＝1
No
rcv_data＝myid
Yes
MPIEをクリア
idrcvfをセット
No
MPIEをセット
dtrcvfをセット
MPIEをセット

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/**********************************/
/*          PROTOCOL              */
/**********************************/
void MPMASMN(void);
#pragma interrupt (MPSCITX)

/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/
# define txdata  ((unsigned char * )0xffec00)  /* Transmit data */
# define txcnt   (*(unsigned char * )0xffec04) /* Transmit counter */
# define txendf  (*(unsigned char * )0xffec05) /* Transmit end flag */


/**********************************/
/*     MAIN PROGRAM: MPAMASMN     */
/**********************************/
void MPMASMN(void)
{

        MSTPCR = 0xffdf;        /* Disable module(SCI0) stop mode*/
        P3DDR_BP.P30DDR = 1;    /* P30 high output */
        P3DR_BP.P30DR = 1;
        SMR0 = 0x04;            /* Set muliti processer mode */
        BRR0 = 64;              /* Set 9600bps */
        SSR0_BP.MPBT0=1;        /* Set MPBT0 */
        txendf = 0;             /* Set txendf=0 */
        txcnt = 0;              /* Set txcnt=0 */
        TDR0 = txdata[txcnt];   /* Set ID(H'01) to TDR0 */
        SCR0_BP. TE0 = 1;       /* Enable transmit */
        SSR0_BP.TDRE0=0;        /* Start transmit */
        SCR0_BP.TIE0=1;         /* Enable TXI */
        set_imask_ccr(0);       /* Enable interrupt */

        while (txendf == 0);

        while (1);
}


/**********************************/
/*   INTERRUPT PROGRAM: MPSCITX   */
/**********************************/
void MPSCITX(void)
{
        SSR0_BP. MPBT0 = ~SSR0_BP. MPBT0;/* Invert MPBT0 */
        txcnt = txcnt + 1;

        if (txcnt<4)
                {
                TDR0 = txdata[txcnt];   /* Load next tans data */
                SSR0_BP.TDRE0=0;        /* Start transmit */
                }
                else
                {
                txendf = 1;
                SCR0_BP. TIE0=0;        /* Disable TXI */
                }
}
```

マルチプロセッサ通信	MCU	H8S／2655	使用機能	SCI（マルチプロセッサ通信）

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/*********************************/
/*          PROTOCOL             */
/*********************************/
void MPSRVMN(void);
#pragma interrupt (MPSCIRX)
/*********************************/
/*     SYMBOL DEFINITIONS        */
/*********************************/
# define rcv_data   (*(unsigned char * )0xffec00)  /* Receive ID,data */
# define idrcvf     (*(unsigned char * )0xffec01)  /* ID code compare flag */
# define dtrcvf     (*(unsigned char * )0xffec02)  /* Data receive flag */
# define rxid       (*(unsigned char * )0xffec03)  /* Receive ID code */
# define rxdata     (*(unsigned char * )0xffec04)  /* Receive data */
# define myid       (*(unsigned char * )0xffec05)  /* My ID code */
/*********************************/
/*     MAIN PROGRAM: MPSRVMN     */
/*********************************/
void MPSRVMN(void)
{
        MSTPCR = 0xffdf;        /* Disable module(SCI0) stop mode*/
        SMR0 = 0x04;            /* Set muliti processer mode */
        BRR0 = 64;              /* Set 9600bps */
        SCR0_BP.MPIE0=1;        /* Enable multiproccesor com. */
        SCR0_BP.RIE0=1;         /* Enable RXI */
        SCR0_BP.RE0=1;          /* Enable receive */
        set_imask_ccr(0);       /* Enable interrupt */

        while (1)
        {
        while (idrcvf == 0);    /* Receive ID ? */
        rxid = rcv_data;        /* Store ID code */
        idrcvf = 0;             /* Clear idrcvf */

        while (dtrcvf == 0);    /* Data receive? */
        rxdata = rcv_data;      /* Store data */
        dtrcvf = 0;             /* Clear dtrcvf */
        }
}
/*********************************/
/*   INTERRUPT PROGRAM: MPSCIRX  */
/*********************************/
void MPSCIRX(void)
{
        rcv_data =RDR0;                 /* Store receive data */
        SSR0_BP. RDRF0 = 0;             /* Clear RDRF */

        if (SSR0_BP.MPB0 == 1)          /* MPB0 = 1 */
        {
            if (rcv_data == myid )      /* Receive ID = my ID */
            {
            SCR0_BP.MPIE0 =0;           /* Clear MPIE0 */
            idrcvf = 1;                 /* Set idrcvf*/
            }
            else
            SCR0_BP.MPIE0=1;            /* Set MPIE0 */

        }
        else
        {
        dtrcvf = 1;                     /* Set dtrcvf */
        SCR0_BP.MPIE0=1;                /* Set MPEIE0 */
        }
}
```

3．12　スキャンモードによるA/D変換

スキャンモードによるA/D変換	MCU	H8S／2655	使用機能	A／D

仕　様

（1）図1に示すように、4チャネルの電圧をH8S／2655に入力し、A／D変換した結果をRAMに格納します。
（2）A／D変換器の起動は外部トリガにより行います。

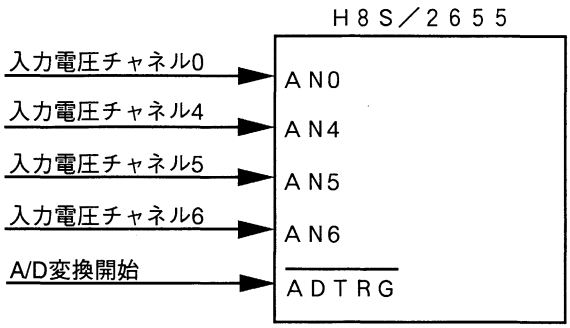


図1　H8S／2655による電圧の測定ブロック図

使用機能説明

（1）図2に4チャネルA／D変換のブロック図を示します。本タスク例ではA／D変換器の以下に示す機能を使用します。
（a）4チャネル（AN0、AN4～6）のA／D変換をソフトウェアを介さずに自動的に行う機能（ｸﾞﾙｰﾌﾟｽｷｬﾝﾓｰﾄﾞ）。
（b）当該チャネルの変換が終了すると、変換結果を別のADDRに転送する機能（バッファ動作）。
（c）外部トリガ端子（ADTRG）によるA／D変換の開始。
（d）A／D変換終了時に割込みを発生させる機能。

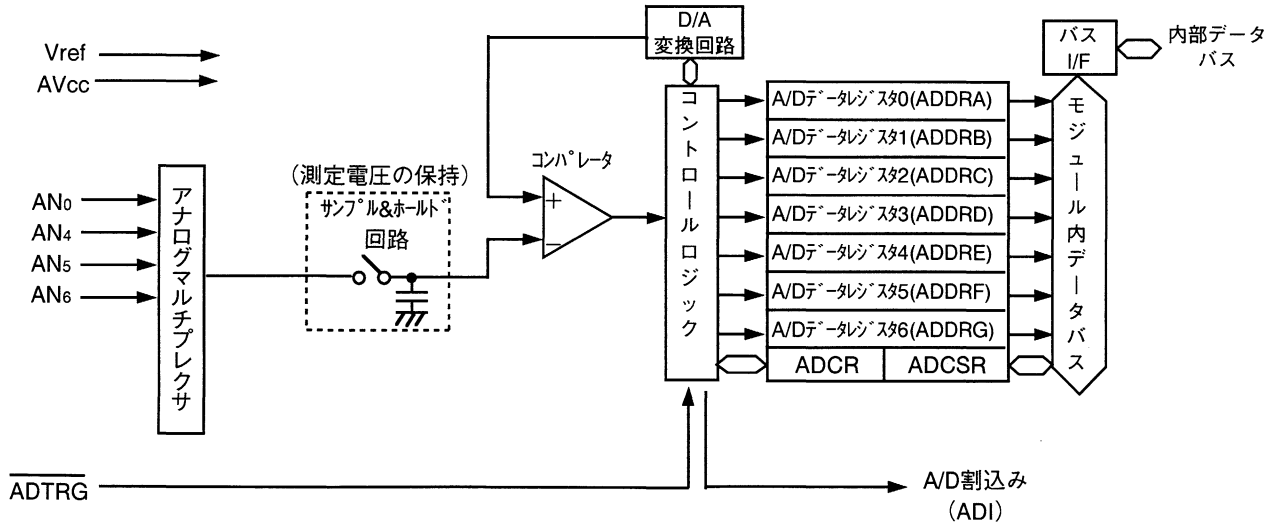


図2　A/D変換器ブロック図

使用機能説明

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、A／D変換を行います。

表1　H8S／2655機能割り付け

H8S/2655機能	機能
ADCSR	A/D変換のﾁｬﾈﾙの選択(ｸﾞﾙｰﾌﾟ)、およびステータスの表示を行う
ADCR	開始トリガ信号の選択および動作モード(ｽｷｬﾝ)を設定する
ADDRA～ ADDR G	A/D変換の結果を格納する
$\overline{\text{ADTRG}}$	A/D外部ﾄﾘｶﾞ入力端子

スキャンモードによるA/D変換	MCU	H8S／2655	使用機能	A／D

動作説明

図3に動作原理を示します。図3に示すように、外部トリガ $\overline{\text{ADTRG}}$ によってA/D変換器が起動され、AN0、AN4～6の4チャネルのA/D変換を繰り返し行います。ADSTビットは、ソフトウェアで0にクリアするまで1を保持し、この期間、選択された入力チャンネルのA/D変換を繰り返します。ここでは、バッファ動作（4段1組）を使用します。また、ADDRA～ADDRGに格納されたA/D変換結果をSCN0～6の140バイトのRAMに格納します。

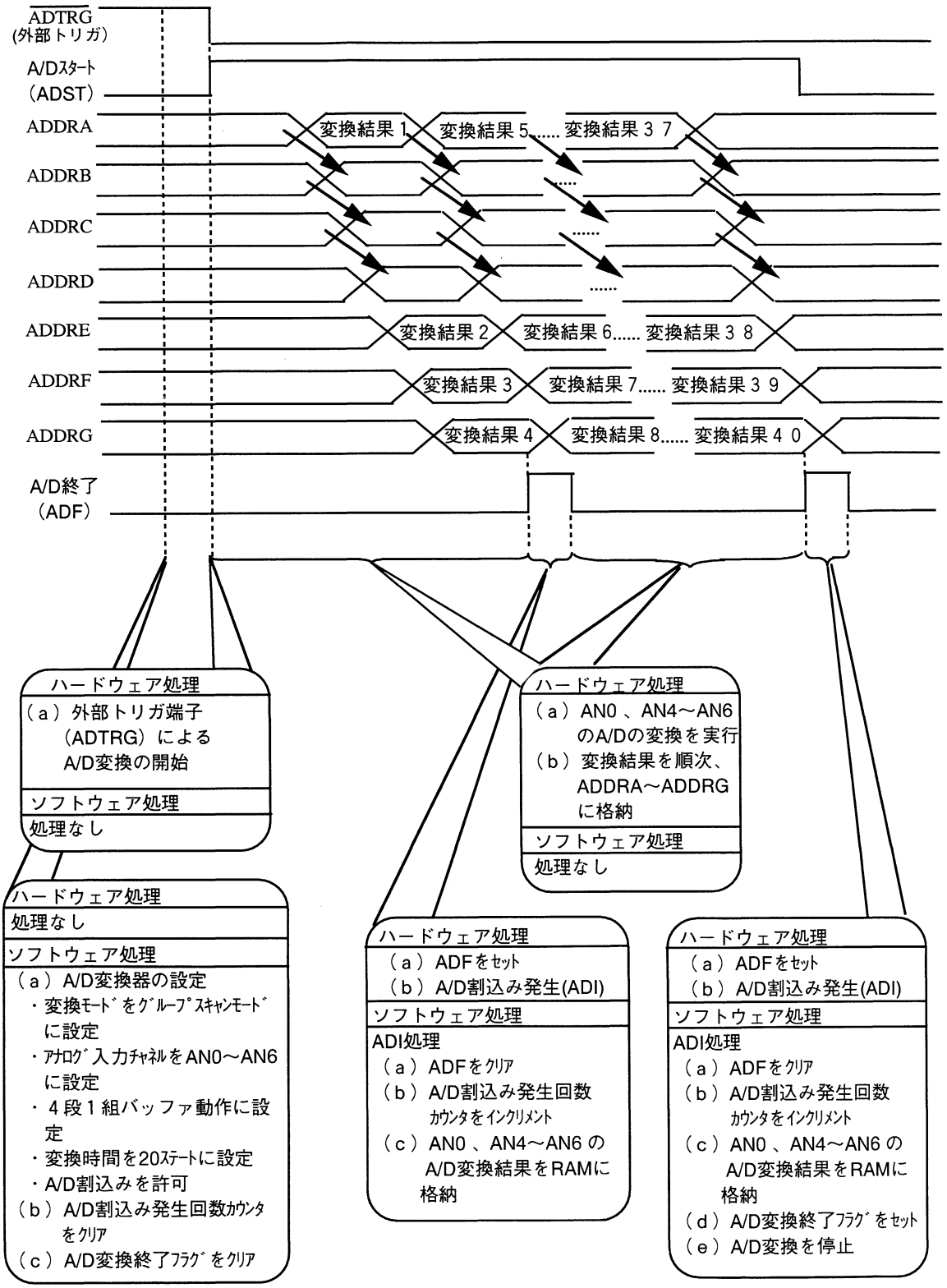


図3　スキャンモードによるA/D変換動作原理

スキャンモードによるA/D変換	MCU	H8S／2655	使用機能	A／D

ソフトウェア説明

（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	ADSCNMN	A/D変換器および外部トリガによるA/D変換器の起動の設定を行う
A/D割込み	SCNEND	ADIにより起動し、A/D変換結果のRAMへの格納およびA/D変換の停止を行う

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
scn	4チャネルのA/D変換結果を設定する 10ビットの変換結果は以下のように設定されている bit7 … bit0 SCN_RE0～6 上位バイト: AD9 AD8 AD7 AD6 AD5 AD4 AD3 AD2 SCN_RE0～6 下位バイト: AD1 AD0 AD0～9はA/D変換の結果のビットNｏを示す	unsigned short	A/D割込み	出力
scn_endf	4チャンネルのA/D変換が全て終了したことを示すフラグ 1：A/D変換終了　　0：A/D変換中	unsigned char	A/D割込み	出力
			メインルーチン	入力

（3）使用内部レジスタ説明

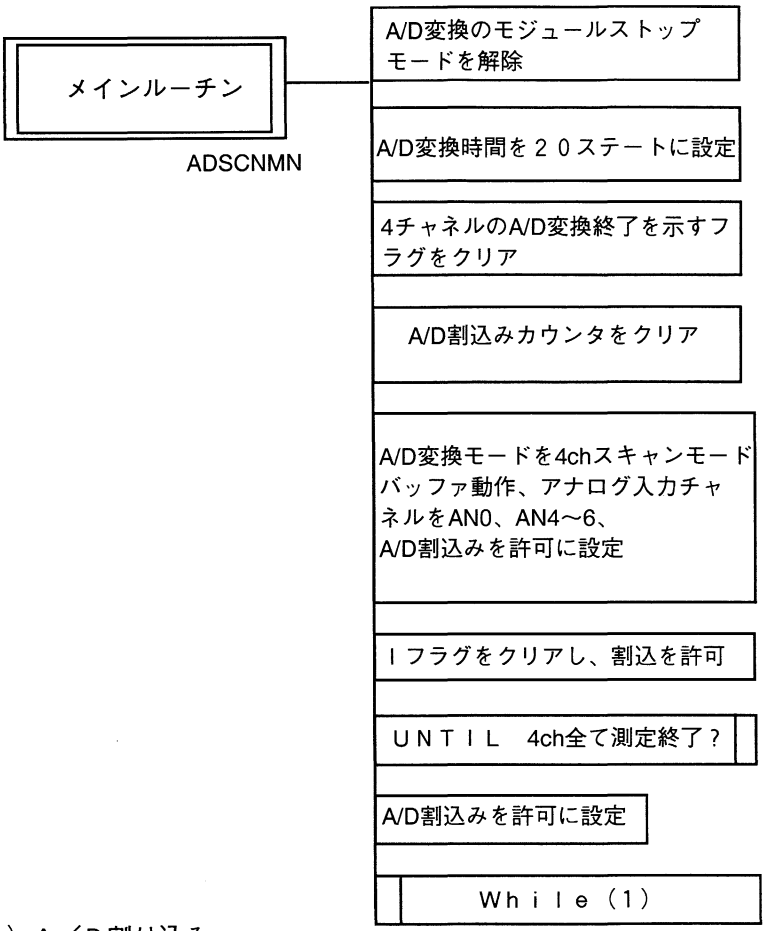
レジスタ名	機能	使用モジュール名
ADCSR	A/D変換時間、アナログ入力チャネル、A/D変換終了時のA/D割込み許可／禁止の選択を行う	メインルーチン A/D割込み
ADCR	A/D変換モード（スキャンモード）、バッファ動作の選択を行う	メインルーチン
ADDR0～ ADDRG	A/D変換の結果が格納される	A/D割込み
MSTPCR	A/D変換のモジュールストップモードを解除する	

（4）使用RAM説明

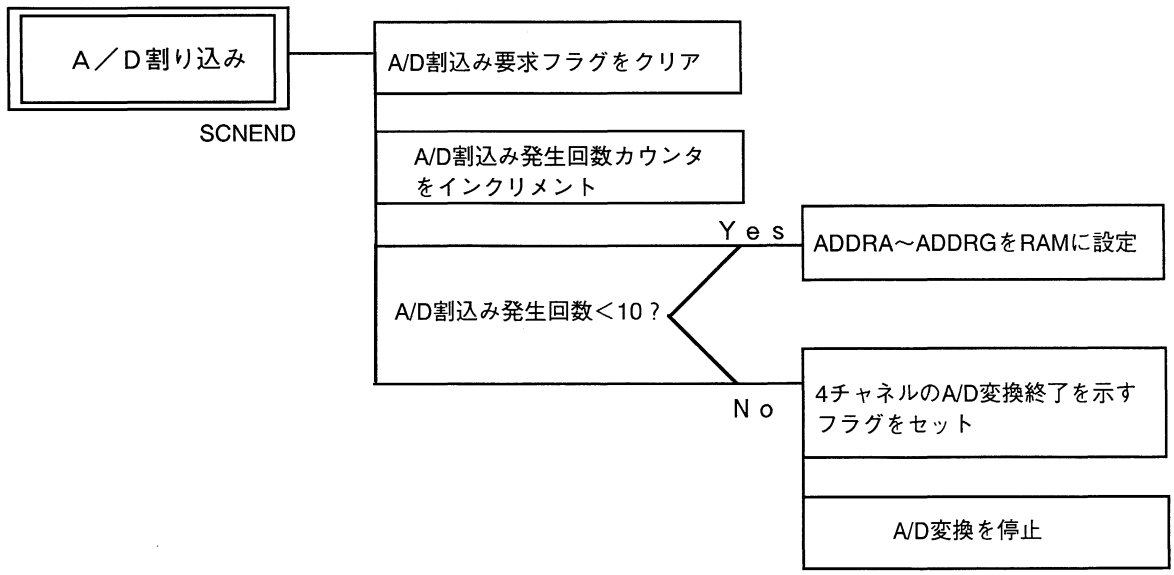
使用モジュール名	ラベル名	機能
A/D割込み	adicnt	A/D割込みの発生回数をカウントする
A/D割込み	scn_cnt	データをRAMに先頭番地から格納するためのカウンタ

スキャンモードによるA/D変換	MCU	H8S／2655	使用機能	A／D
PAD				

（1）メインルーチン



（2）A／D割り込み



スキャンモードによるA/D変換	MCU	H8S／2655	使用機能	A／D

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/***********************************/
/*          PROTOCOL               */
/***********************************/
void ADSCNMN(void);
#pragma interrupt (SCNEND)

/***********************************/
/*     SYMBOL DEFINITIONS          */
/***********************************/

# define scn       ((unsigned short * )0xffec00) /* Result of A/D conversion */
# define scn_cnt   (*(unsigned char * )0xffec8c) /* Work */
# define scn_endf  (*(unsigned char * )0xffec8d) /* A/D conversion end flag */
# define adicnt    (*(unsigned char * )0xffec8e) /* A/D conversion counter */

/***********************************/
/*     MAIN PROGRAM: ADSCNMN       */
/***********************************/
void ADSCNMN(void)
{
        MSTPCR = 0x3dff;        /* Disable module(A/D) stop mode */
        scn_endf = 0;
        scn_cnt = 0;
        adicnt = 0;
        ADCR = 0x3b;            /* Initialize ADCR */
        ADCSR = 0x4E;           /* Initialize ADCSR */
        set_imask_ccr(0);       /* Enable interrupt */

        while (scn_endf == 0)   /* A/D conversion finish ? */

        ADCSR_BP.ADIE=1;        /* Enable A/D interrupt */

        while (1);              /* Loop */
}

/***********************************/
/*   INTERRUPT PROGRAM: SCNEND     */
/***********************************/
void SCNEND(void)
{
        ADCSR_BP.ADF =0;        /* Clear ADF */

        if (adicnt<10)
                {
                scn[scn_cnt++] = ADDRA; /* Set RAM address to store the data */
                scn[scn_cnt++] = ADDRB;
                scn[scn_cnt++] = ADDRC;
                scn[scn_cnt++] = ADDRD;
                scn[scn_cnt++] = ADDRE;
                scn[scn_cnt++] = ADDRF;
                scn[scn_cnt++] = ADDRG;

                adicnt = adicnt+1;

                }
                else
                {
                scn_endf = 1;   /* Set scn_endf */
                ADCSR_BP.ADST=0;/* Stop A/D conversion */
                }
}
```

３．１３　ブロック転送

ブロック転送	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	データトランスファコントローラ（ＤＴＣ）（ブロック転送）

仕様

図１に示すように、外部信号の立ち下がりエッジ検出ごとにＲＯＭに設定した３０バイト（６バイト×５ブロック）のデータをＩ／Ｏポートに転送しパルスを出力します。

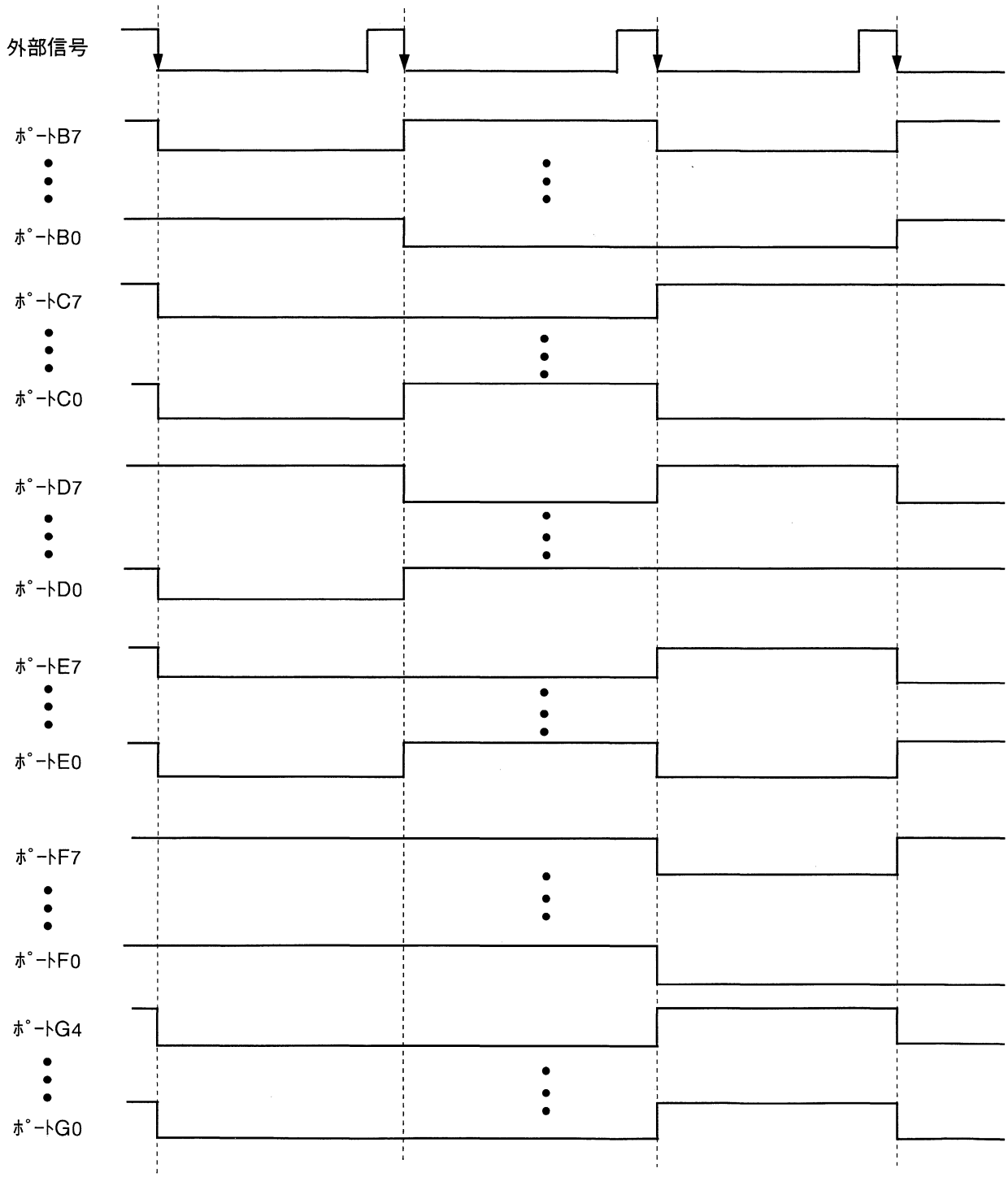


図１　波形出力例

ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

使用機能

（1）本タスク例では、IRQ0の立ち下がりエッジごとにDTCを起動し、6バイトのデータをポートBからポートGへ出力します。

（a）図2に本タスク例で使用するDTCのブロック図を示します。本タスク例では以下の機能を使用して、ブロック転送を行います。

・外部リクエストによりDTCを起動する機能（IRQによるDTC起動）

・DTC起動時にブロック単位でデータを転送する機能（ブロック転送モード）

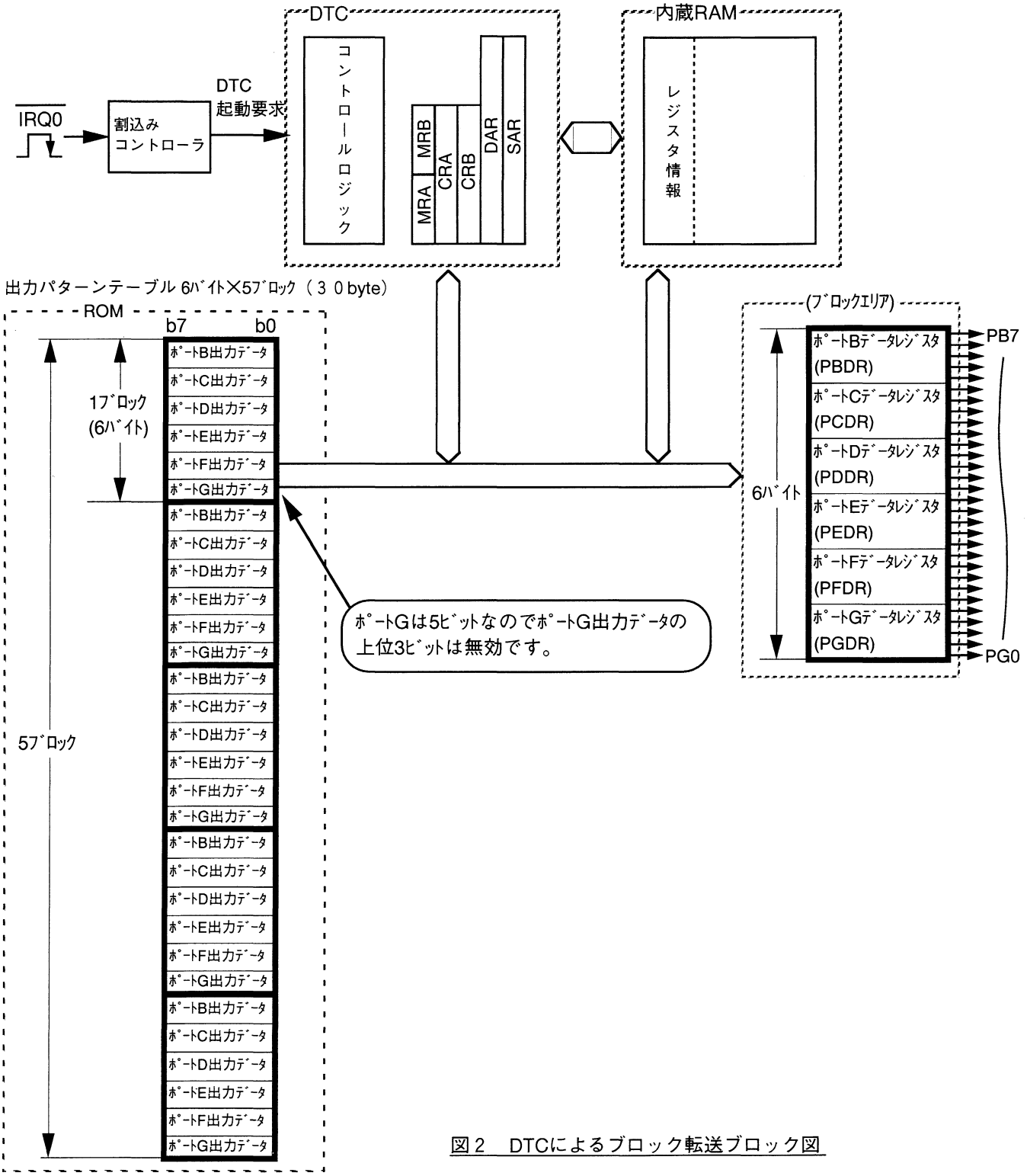


図2　DTCによるブロック転送ブロック図

使用機能

（b）図3にDTCのベクタテーブルとメモリ上の配置を示します。H'FFF800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。

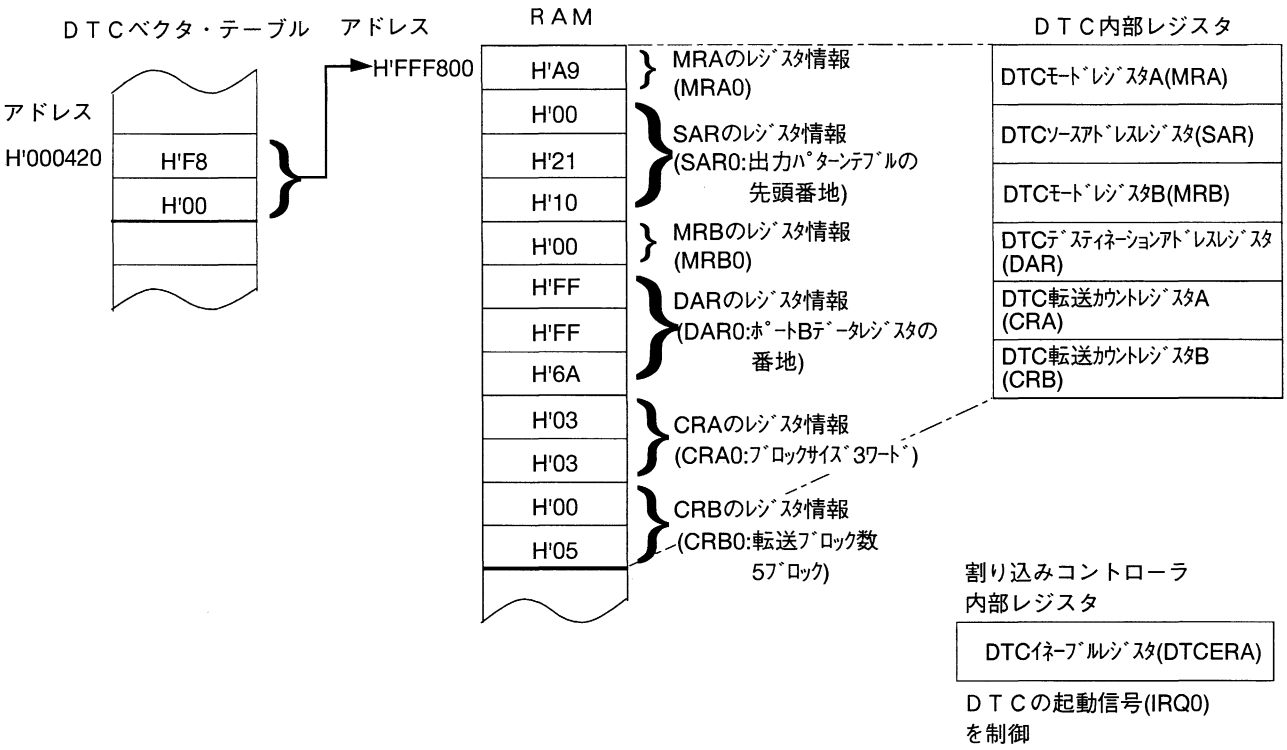


図3　DTCベクタテーブルとメモリ上の配置例

（2）表1に本タスク例の機能割り付けを示します。表1に示すように機能を割り付け、ブロック転送を行います。

表1　H8S／2600機能割り付け

DTC機能	機能
MRA,B	DTCモードを制御する
SAR	転送元アドレスを指定する
DAR	転送先アドレスを指定する
CRA	データ転送の転送回数を指定する
CRB	ブロック転送モード時の転送回数を指定する
DTCER	各割込み要因によるDTC起動の許可／禁止を制御する

ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

動作説明

（1）図4にDTCを使用した場合の動作原理を示します。図4に示すタイミングでハードウェア処理およびソフトウェア処理を行い、ブロック転送を実行します。

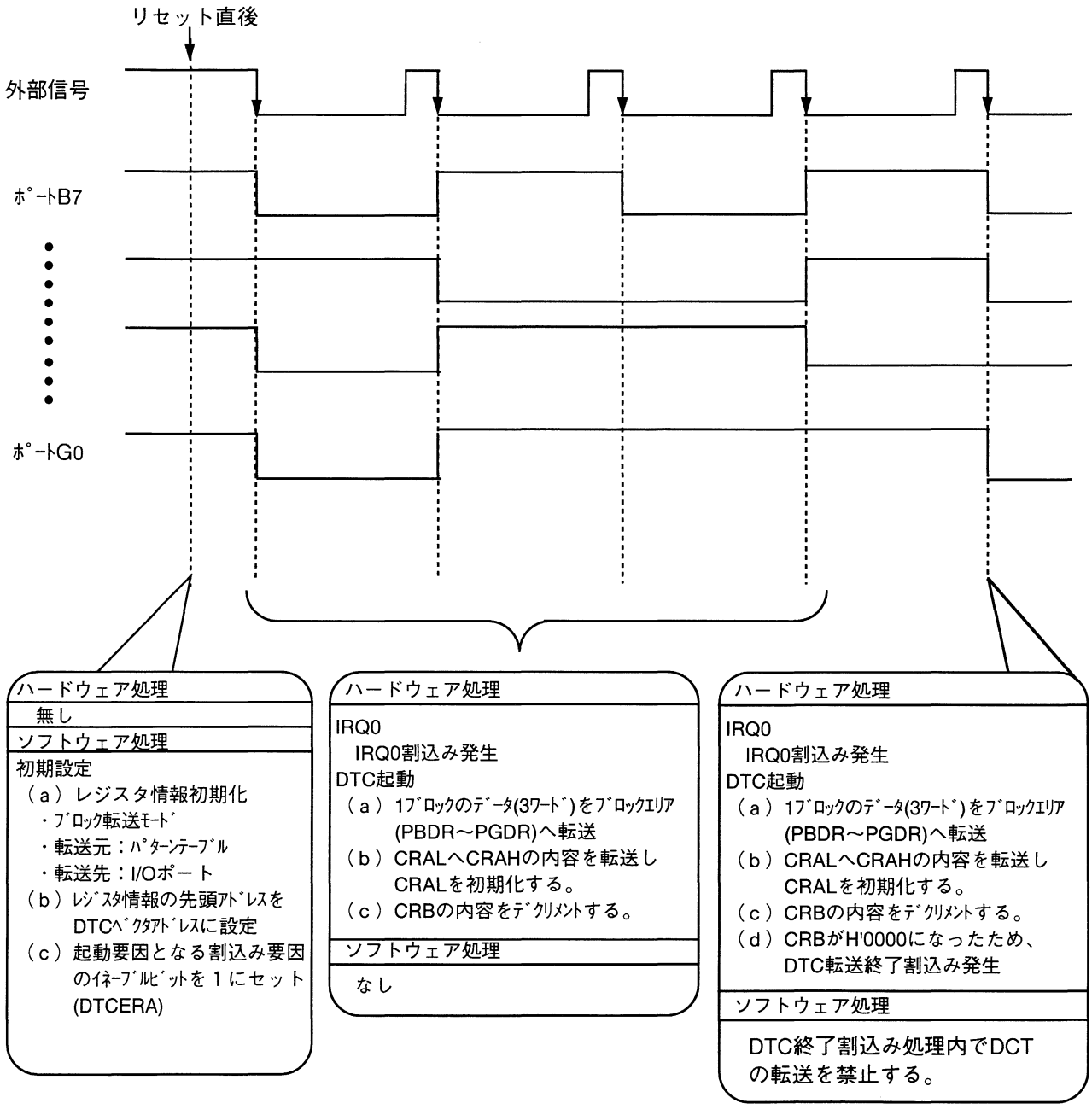


図4　ブロック転送動作原理

ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

ソフトウェア説明

（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	blkmn	DTCの初期設定を行う
転送完了	txend	DTC転送終了割込みで起動し、DTCによる転送禁止を行う

（2）引数の説明

本タスク例ではモジュール間の引数はありません。

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
DTCER	IRQ0割込みによるDTC起動の許可する	メインルーチン
MSTPCR	DTCのモジュールストップモードを制御する	
ISCRL	IRQ0の立ち下がりエッジで割込み要求に設定する	
IER	IRQ0の割込みを許可する	
ISR	IRQ0入力のステータスを示す	

（4）使用ＲＡＭ説明

ラベル名	機能	データ長	使用モジュール名
MRA0	DTC0 をブロック転送モードに設定する	unsigned char	メインルーチン
MRB0	CPUへの割込み禁止する	unsigned char	
SAR0	転送元アドレス（PATTBL1）を設定する	unsigned long	
DAR0	転送先アドレス（PBDR）を設定する	unsigned long	
CRA0	ブロックサイズを設定する	unsigned short	
CRB0	転送ブロック数を設定する	unsigned short	
txendf	転送終了フラグ	unsigned char	転送終了

（5）データテーブル説明

テーブル名	機能	データ長	データ容量
PATTBL1	出力パターンを設定する	unsigned short	15WORD

ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

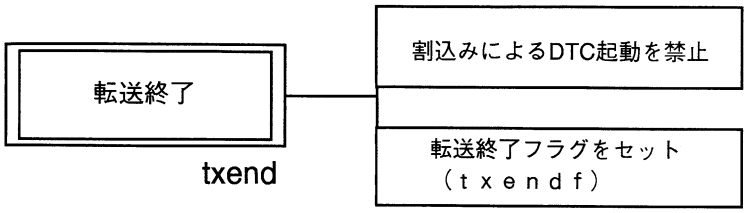
PAD

（1）メインルーチン

ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

PAD

（2）転送終了



ブロック転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

プログラムリスト

```
#include <machine.h>
#include "..¥h8sapn¥h8s.h"
/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void blkmn(void);
#pragma interrupt (txend)

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define txendf  (*(volatile unsigned char *)0xffec00)
#define SAR0    (*(volatile unsigned long *)0xfff800)
#define MRA0    (*(volatile unsigned char *)0xfff800)
#define DAR0    (*(volatile unsigned long *)0xfff804)
#define MRB0    (*(volatile unsigned char *)0xfff804)
#define CRA0    (*(volatile unsigned short *)0xfff808)
#define CRB0    (*(volatile unsigned short *)0xfff80a)

/*****************************************/
/*         DATA TABLE                    */
/*****************************************/
const unsigned short PATTBL1[5][3] = {{0x1111,0x2222,0x3333,}, {0x4444,0x5555,0x6666},
                                      {0x7777,0x8888,0x9999}, {0x1010,0x2020,0x3030},
                                       {0x4040,0x5050,0x6060}};
                                      /* Output data table */

/*****************************************/
/*        MAIN PROGRAM : blkmn           */
/*****************************************/
void blkmn(void)
{
        MSTPCR = 0x3fff;          /* Disable module(DTC) stop mode*/
        PBDDR = 0xff;             /* PB-PG : output */
        PCDDR = 0xff;
        PDDDR = 0xff;
        PEDDR = 0xff;
        PFDDR = 0xff;
        PGDDR = 0xff;
        SAR0 = (long)(PATTBL1); /* Set base address */
        DAR0 = (long)(&PBDR);   /* Set excute address */
        MRA0 = 0xa9;            /* Block translation mode */
        MRB0 = 0x00;            /* Initialize MRB0 */
        CRA0 = 0x0303;          /* Set excute count */
        CRB0 = 0x0005;          /* Set block excute count */
        DTCERA_BP.IRQ0 = 1;     /* Enable DTC */

        txendf = 0;             /* Clear DTC end flag */

        ISCRL = 0x01;           /* Initialize ISCRL */
        ISR_BP.IRQ0F = 0;       /* Clear IRQ flag */
        IER = 0x01;             /* Enable IRQ0 interrupt */
        set_imask_ccr(0);       /* Enable interrupt */
        while(txendf == 0);
        while(1);
}

/*****************************************/
/*        NAME : txend                   */
/*****************************************/
void txend(void)
{
        DTCERA_BP.IRQ0 = 0;     /* Disable DTC */
        txendf =1;              /* Set DTC end flag */
}
```

３．１４　　ソフトウェア起動によるデータ転送

ソフトウェア起動によるデータ転送	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	データトランスファコントローラ（ＤＴＣ）（ブロック転送）

仕様

（１）図１に示すように、ポートの立ち下がりエッジ検出時にＤＴＣを起動させ、１２８バイト１ブロックの転送を行います。
（２）転送エリアはH’Ａ００００００～H’Ａ０００ＦＦです。
（３）Ｈ８Ｓ／２６５５の内部動作周波数は２０ＭＨｚで使用します。

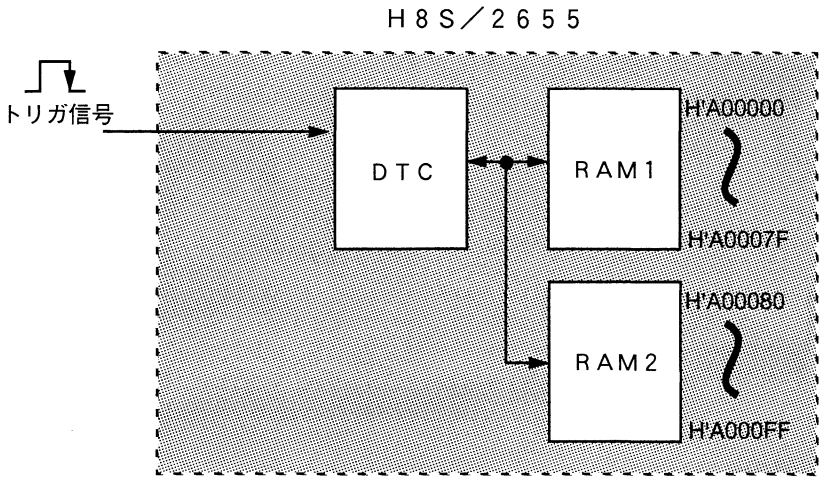


図１　ソフトウェア起動によるデータ転送ブロック図

ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

使用機能

（1）本タスク例では、ソフトウェアによりDTCを起動させ、１２８バイトのデータをRAMに転送します。

（a）図2に本タスク例で使用するDTCのブロック図を示します。

本タスク例では以下の機能を使用して、データ転送を行います。

・ソフトウェアによりDTCを起動する機能（ソフトウェアによるDTC起動）

・データ転送の終了後に、CPUに対する割込み要求を発生可能

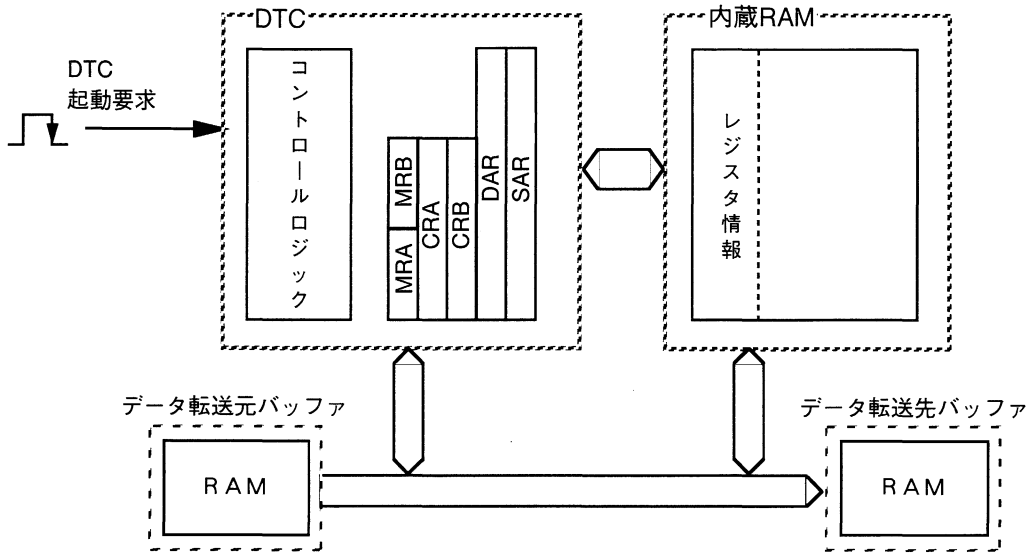


図2　ソフトウェア起動によるデータ転送ブロック図

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH８S／２６５５の機能を割り付け、ブロック転送を行います。

表1　H８S／２６５５の機能割り付け

H8S/2655機能	機能
MRA,B	ＤＴＣモードを制御する
ＳＡＲ	転送元アドレスを指定する
ＤＡＲ	転送先アドレスを指定する
ＣＲＡ	データ転送の転送回数を指定する
ＤＴＣＥＲ	各割込み要因によるＤＴＣ起動の許可／禁止を制御する
Ｐ３ＤＲ	トリガ信号を入力する

ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

動作説明

図3にDTCを使用した場合の動作原理を示します。図3に示すタイミングでハードウェア処理およびソフトウェア処理を行い、ブロック転送を実行します。

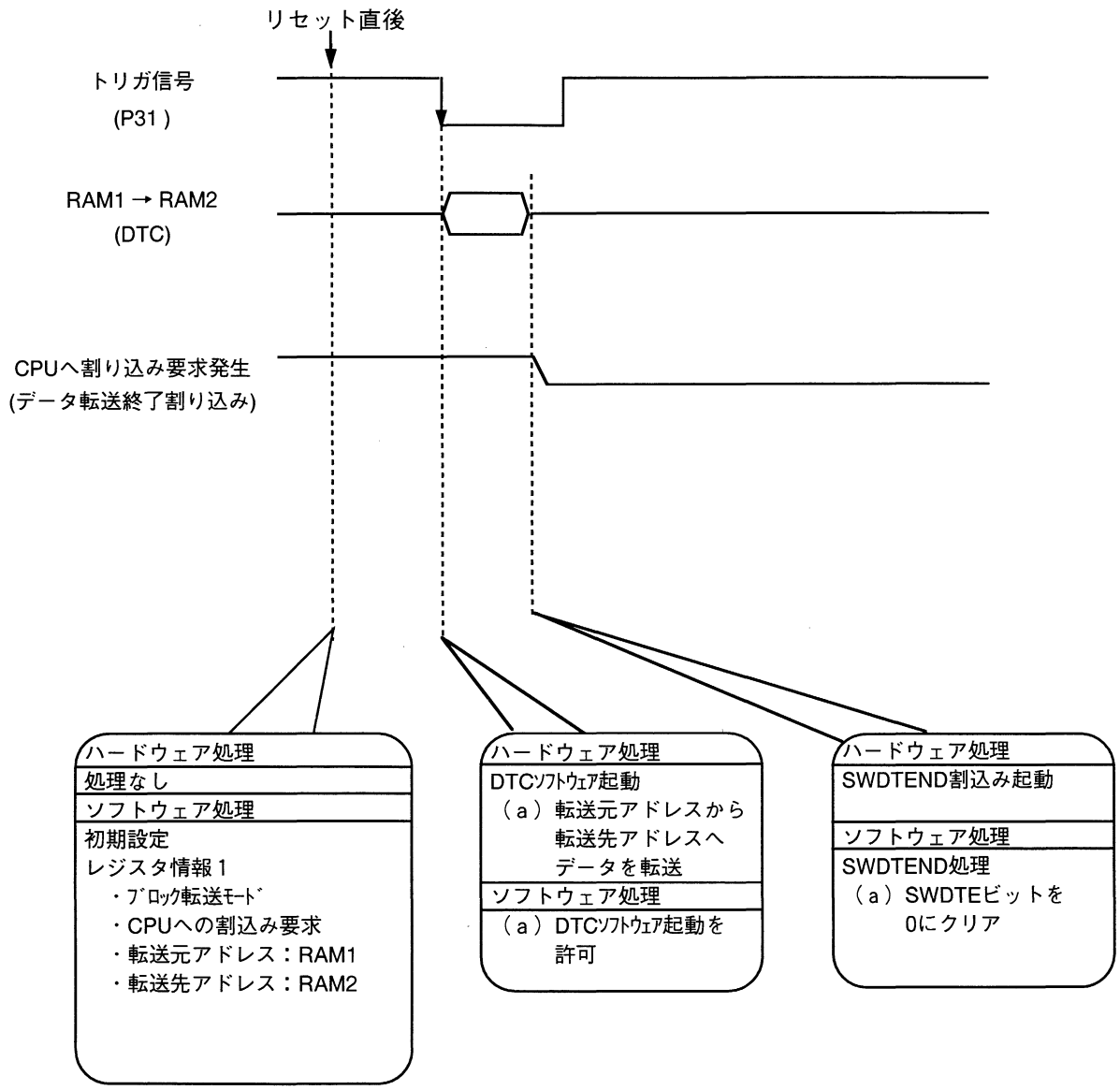


図3　ソフトウェア起動によるデータ転送動作原理

ソフトウェア起動によるデータ転送	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	データトランスファコントローラ（ＤＴＣ）（ブロック転送）

ソフトウェア説明

（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	dtcsftmn	DTCの初期設定を行う
転送完了	trsend	DTC転送終了割込みで起動し、転送終了フラグのセットを行う

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
trs_end	転送終了を示すフラグ 1：転送終了　0：転送中	unsigned char	データ転送終了	出力
			メインルーチン	入力
err	DTC起動エラーを示すフラグ 1：起動失敗　0：起動	unsigned char	メインルーチン	出力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
DTVECR	ソフトウェアによるDTC起動の許可する	メインルーチン
MSTPCR	DTCのモジュールストップモードを制御する	メインルーチン

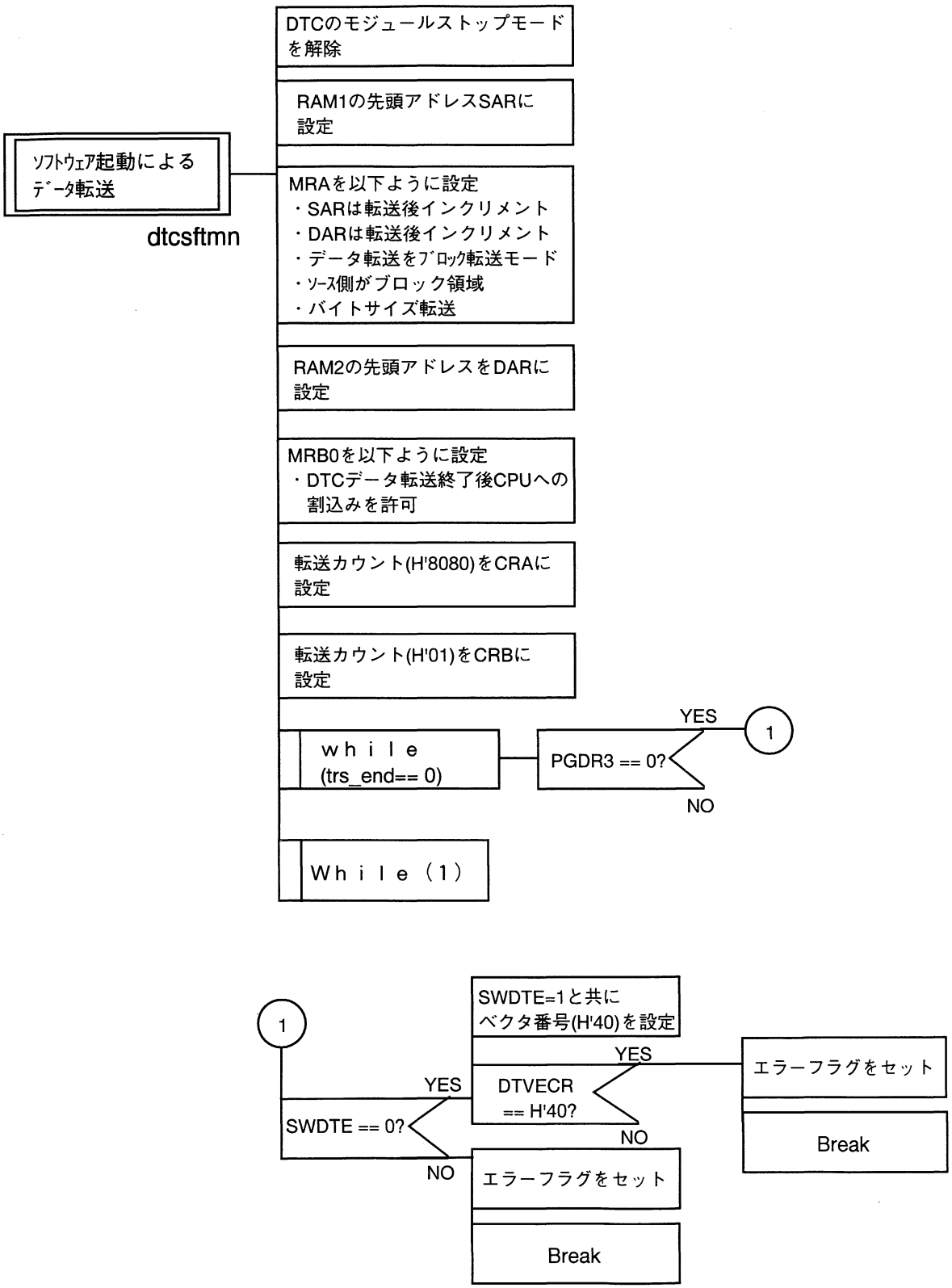
（4）使用ＲＡＭ説明

ラベル名	機能	データ長	使用モジュール名
MRA	DTC をブロック転送モードに設定する	unsigned char	メインルーチン
MRB	データ転送後、CPUへの割込みを許可する	unsigned char	
SAR	転送元アドレス（RAM1）を設定する	unsigned long	
DAR	転送先アドレス（RAM2）を設定する	unsigned long	
CRA	ブロックサイズ(H'8080)を設定する	unsigned short	
CRB	転送回数(H'0001)を設定する	unsigned short	

ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

PAD

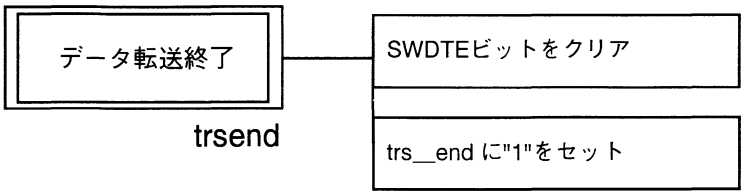
（1）メインルーチン



ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

PAD

（2）データ転送終了



ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void dtcsftmn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define trs_end (*(volatile unsigned char *)0xffec00)
#define err (*(volatile unsigned char *)0xffec01)

volatile struct databuf
{
              unsigned char  ram1[128];
              unsigned char  ram2[128];
};
#define dat (*(struct databuf *)0xA00000)

#define MRA (*(volatile unsigned char *)0xfff800)
#define SAR (*(volatile unsigned long *)0xfff800)
#define MRB (*(volatile unsigned char *)0xfff804)
#define DAR (*(volatile unsigned long *)0xfff804)
#define CRA (*(volatile unsigned short *)0xfff808)
#define CRB (*(volatile unsigned short *)0xfff80a)
```

ソフトウェア起動によるデータ転送	MCU	H8S／2655	使用機能	データトランスファコントローラ（DTC）（ブロック転送）

プログラムリスト

```
/*****************************************/
/*        MAIN PROGRAM : dtcsftmn        */
/*****************************************/
void dtcsftmn(void)
{
        MSTPCR = 0x3fff;
        SAR = (long)(&dat.ram1);
        MRA = 0xa8;
        DAR = (long)(&dat.ram2);
        MRB = 0x40;
        CRA = 0x8080;
        CRB = 0x0001;

        while (trs_end == 0)
        {
                if (P3DR_BP.P31DR == 0)
                {
                        if (DTVECR_BP.SWDTE == 0){
                                DTVECR = 0xc0;

                                if (DTVECR_BP.VECR != 0x40)
                                        err = 1;
                                        break;
                        }
                        else{
                                err = 1;
                                break;
                        }
                }
        }

        while(1);
}

/*****************************************/
/*        NAME : trsend                  */
/*****************************************/
#pragma interrupt(trsend)
void trsend(void)
{
        DTVECR_BP.SWDTE = 0;
                trs_end = 1;
}
```

3．15　シングルアドレスモードによるデータ転送

シングルアドレスモードによるデータ転送	MCU	H8S／2655	使用機能	DMAコントローラ（シングルアドレスモード）

仕様

（1）図1に示すように、DMACのシングルアドレスモードを使用し、転送元または転送先のいずれか一方がアドレスによって指定される外部空間と、アドレスにかかわらず、DACKストローブにより選択動作する外部デバイスとの転送を行います。

（2）DMACの起動は、外部信号の立ち下がりエッジ検出により起動します。

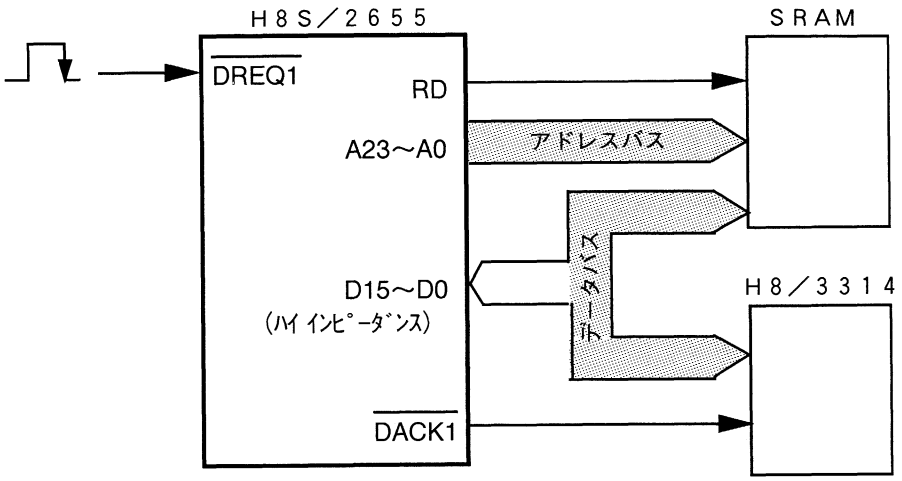


図1　シングルアドレスモードのデータバス

使用機能説明

（1）本タスク例ではDMACのシングルアドレスモード（アイドルモード指定）を使用し、外部メモリ（SRAM）から外部デバイス（H8／3314）へデータを転送します。

（a）図2に本タスク例で使用するDMACブロック図を示します。
本タスク例はDMACの以下の機能を使用してブロック転送を行います。
・外部リクエストによりDMACを起動する機能（DREQによるDMAC起動）
・一回の転送要求に対して、1バイトまたは1ワードずつ指定された回数だけ外部メモリと外部デバイス間を転送（シングルアドレスモード）

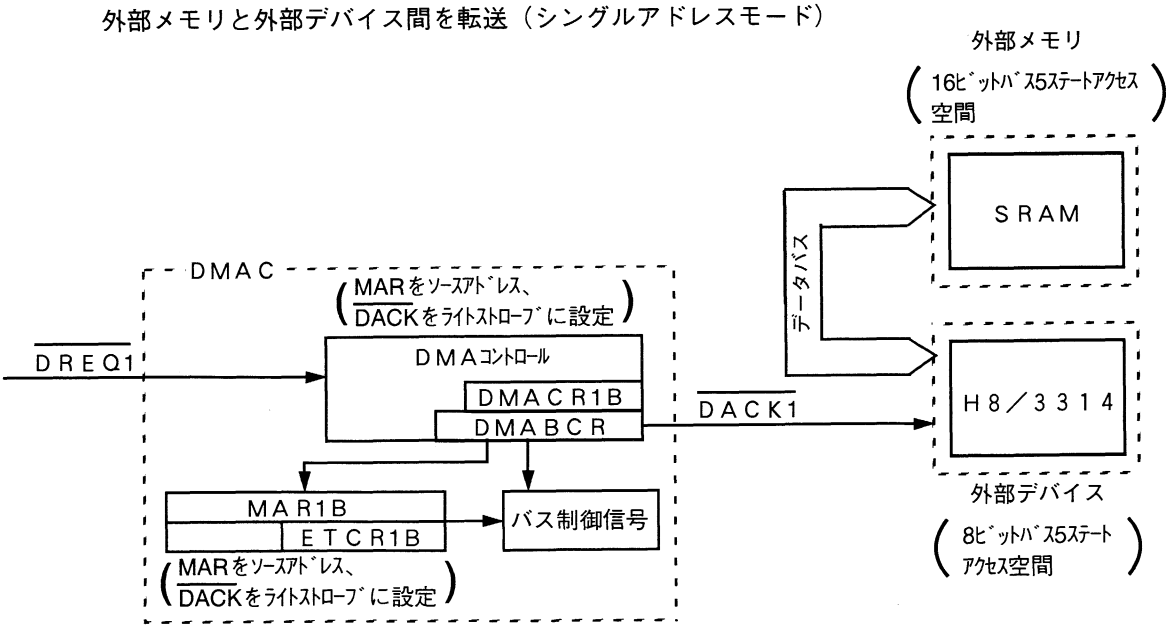


図2　DMAコントローラブロック図

ｼﾝｸﾞﾙｱﾄﾞﾚｽﾓｰﾄﾞによるﾃﾞｰﾀ転送	MCU	H8S／2655	使用機能	DMAコントローラ （ｼﾝｸﾞﾙｱﾄﾞﾚｽﾓｰﾄﾞ）

使用機能

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、ブロック転送を行います。

表1　H8S／2655機能割り付け

H8S/2655機能	機能
$\overline{\text{DREQ1}}$	DMAC起動トリガとなる外部パルスを入力する
DACK1	データ転送アクノレッジ
DMABCR	各チャネルの動作を制御する
DMACR1B	DMACをアイドルモードに設定する
MAR1B	転送元アドレスを設定する
ETCR1B	転送回数を設定する

シングルアドレスモードによるデータ転送	MCU	H8S／2655	使用機能	DMAコントローラ（シングルアドレスモード）

動作説明

図3に動作原理を示します。図3に示すようにH8S/2655のハードウェア処理およびソフトウェア処理により外部16ビット5ステートアクセス空間から外部デバイス8ビット5ステートアクセス空間へ1バイト転送します。

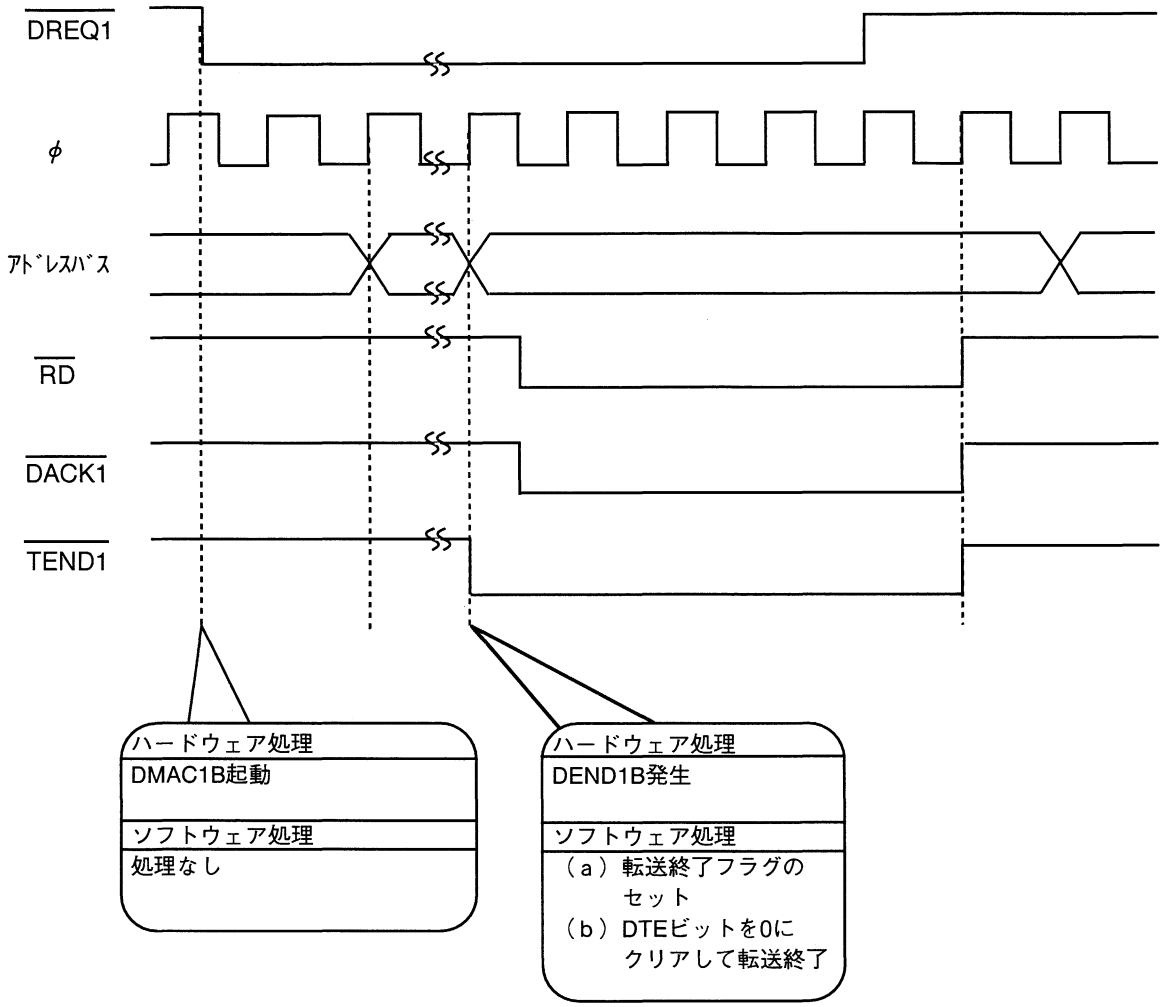


図3　シングルアドレスモード（バイトリード）転送動作原理

シングルアドレスモードによるデータ転送	MCU	H8S／2655	使用機能	DMAコントローラ（シングルアドレスモード）

ソフトウェア説明

（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	singlemn	DMACの初期設定を行う
データ転送終了	transend	転送終了フラグのセットを行う

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
status	データ転送終了を示すフラグ 1：転送終了　　0：動作中	unsigned char	メインルーチン	入力
			転送終了	出力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
DMABCRH	DMAC1Bをショートアドレスモードのシングルアドレスモードに設定する	メインルーチン
DMABCRL	データ転送を許可する	メインルーチン
DMACR1B	DMACRを以下のように設定する ・バイトサイズ転送 ・アイドルモードに設定 ・データ転送後MARをインクリメント ・データ転送方向（MARをソースアドレス、DACK端子をライトストローブ） ・起動要因をDREQ端子の立ち下がりエッジ入力に設定する	メインルーチン
MAR1B	転送元アドレスを設定する	メインルーチン
ETCR	転送回数を設定する	メインルーチン
MSTPCR	DMACのモジュールストップモードを解除する	メインルーチン

（4）使用RAM説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名
transdata	送信するデータを設定する	unsigned char	メインルーチン

シングルアドレスモードによるデータ転送	MCU	H8S／2655	使用機能	DMAコントローラ（シングルアドレスモード）

PAD

（1）メインルーチン

シングルアドレスモードによるデータ転送

singlemn

DMACのモジュールストップモードを解除

P6DRを#H'0Fに設定
P6DDRを#H'0Bに設定

ｓｔａｔｕｓをクリア

TEND1端子出力を許可

転送用データをtransdataに設定

DMABCRHを以下ように設定
・シングルアドレスモード
・内部割込みクリア禁止

転送元アドレス（transdata）をMAR1Bに設定

ETCRに転送回数（1回）を設定

DMACR1Bを以下ように設定
・バイトサイズ転送
・アイドルモードに設定
・データ転送方向
（MARをソースアドレス、
DACK端子をライトストローブ）
・起動要因をDREQ端子の
立ち下がりエッジ入力に設定

DMABCRのリード

DMABCRを以下ように設定
・チャネル0Bの転送許可
・チャネル0Bの転送終了割込み
を許可

Iフラグをクリアし、割込みを許可

ｗｈｉｌｅ（1）

（2）データ転送終了

データ転送終了

transend

データ転送終了フラグのセット
（ｓｔａｔｕｓ）

転送終了割込み禁止

シングルアドレスモードによるデータ転送	MCU	H8S／2655	使用機能	DMAコントローラ （シングルアドレスモード）

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void singlemn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define status (*(volatile unsigned char *)0xffec00)
#define transdata (*(volatile unsigned char *)0x800000)

/*****************************************/
/*      MAIN PROGRAM : singlemn          */
/*****************************************/
void singlemn(void)
{
        MSTPCR = 0x7fff;

        status = 0x00;                  /*ユーザーフラグクリア*/

        P6DR = 0x0F;
        P6DDR = 0x0B;                   /*対応するポートを出力にする*/

        DMATCR =  0x20;                 /*転送終了後TEND1からLOW出力*/
        transdata = 0xaa;               /*転送用データセット*/

        DMABCRH = 0x20;                 /*ch1Bをショートアドレスモードにする
                                          ch1Bをシングルアドレスモードにする
                                          ch1Bを内部割込みクリア禁止にする*/
        MAR1B = (long)(&transdata);     /*転送元アドレスをいれる*/
        ETCR1B = 0x0001;                /*転送回数をいれる*/
        DMACR1B = 0x22;                 /*バイトサイズ転送&アイドルモード&
                                        MARをインクリメント&DREQ1立ち下がりエッジ*/
        DMABCRL |= 0x88;                /*ch1Bデータ転送許可&
                                                ch1B転送終了割込み許可*/
        set_imask_ccr(0);

        while(1);
}


/*****************************************/
/*    NAME : transend(set end flag)      */
/*****************************************/
#pragma interrupt(transend)
void transend(void)
{
        status = 0x01;                  /*ユーザーフラグクリア*/
        DMABCRL &= 0x77;                /*フラグクリア*/

}
```

３．１６　パルス数の測定

パルス数の測定	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	８ビットタイマ

仕様

（１）図１に示すように、デューティ５０％のパルスを任意の数出力します。

（２）２０ＭＨｚ動作時、パルスの周期は１．２μｓから１０２μｓまで０．４μｓ毎に設定が可能、
また出力するパルスの数は、１～２５６の間、任意に設定できます。

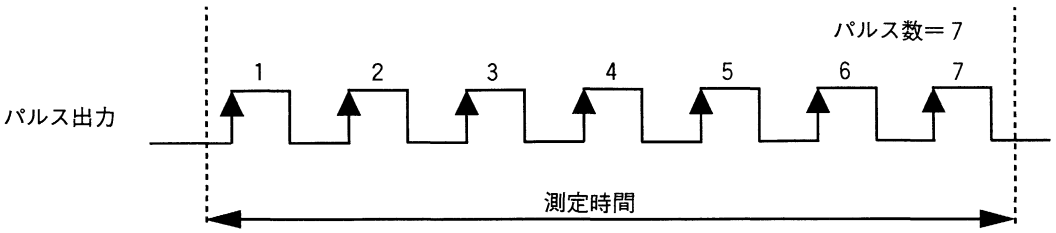


図１　パルス出力タイミング

使用機能説明

（１）図２に本タスク例で使用する８ビットタイマのブロック図を示します。以下に本タスクに使用する機能を示す。
（ａ）２チャネルの８ビットタイマをカスケード接続し、チャネル０のコンペアマッチをチャネル１のタイマでカウントする機能。（コンペアマッチカウントモード）
（ｂ）設定されたカウント回数に割り込みを発生させる機能。
本タスク例ではこの機能を図２に示すように使用し、パルスの立ち上がりをカウントします。

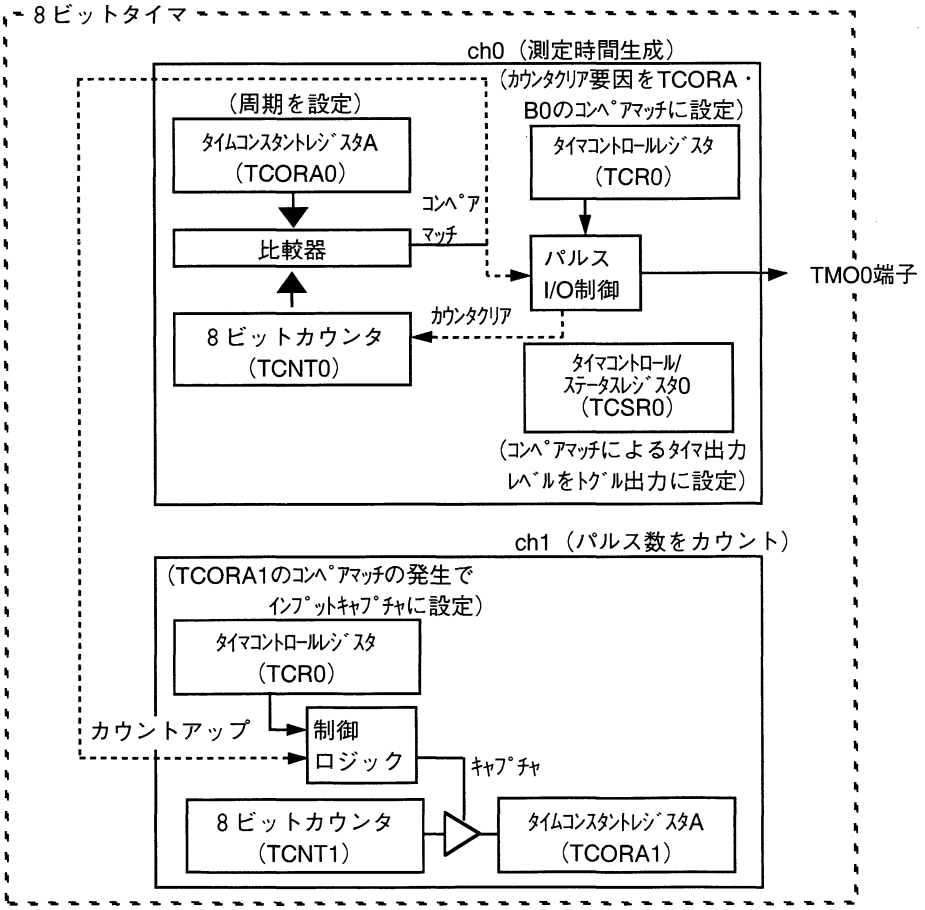


図２　出力パルスカウントブロック図

パルス数の測定	MCU	H8S／2655	使用機能	8ビットタイマ

使用機能説明

（2）表1に本タスク例の機能割付を示します。表1に示すようにH8S／2655の機能を割付け、パルス数の測定を行ないます。

表1　H8S／2655機能割付

H8S/2655機能	機能
TCNT0	コンペアマッチA・B発生用
TCORA0	コンペアマッチA発生用
TCORB0	コンペアマッチB発生用
TCSR0	コンペアマッチA毎に1出力：コンペアマッチB毎に0出力
TMO0	タイマ出力端子（コンペアマッチ出力）
TCR0	コンペアマッチAによりカウンタクリア：入力クロックの選択（φ/8）
TCNT1	チャネル0のコンペアマッチA発生回数をカウントする
TCORA1	コンペアマッチA発生用
TCR1	コンペアマッチAによりカウンタクリア：コンペアマッチ（A）割り込み許可に設定する

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理及びソフトウェアの処理によりパルス数を測定します。

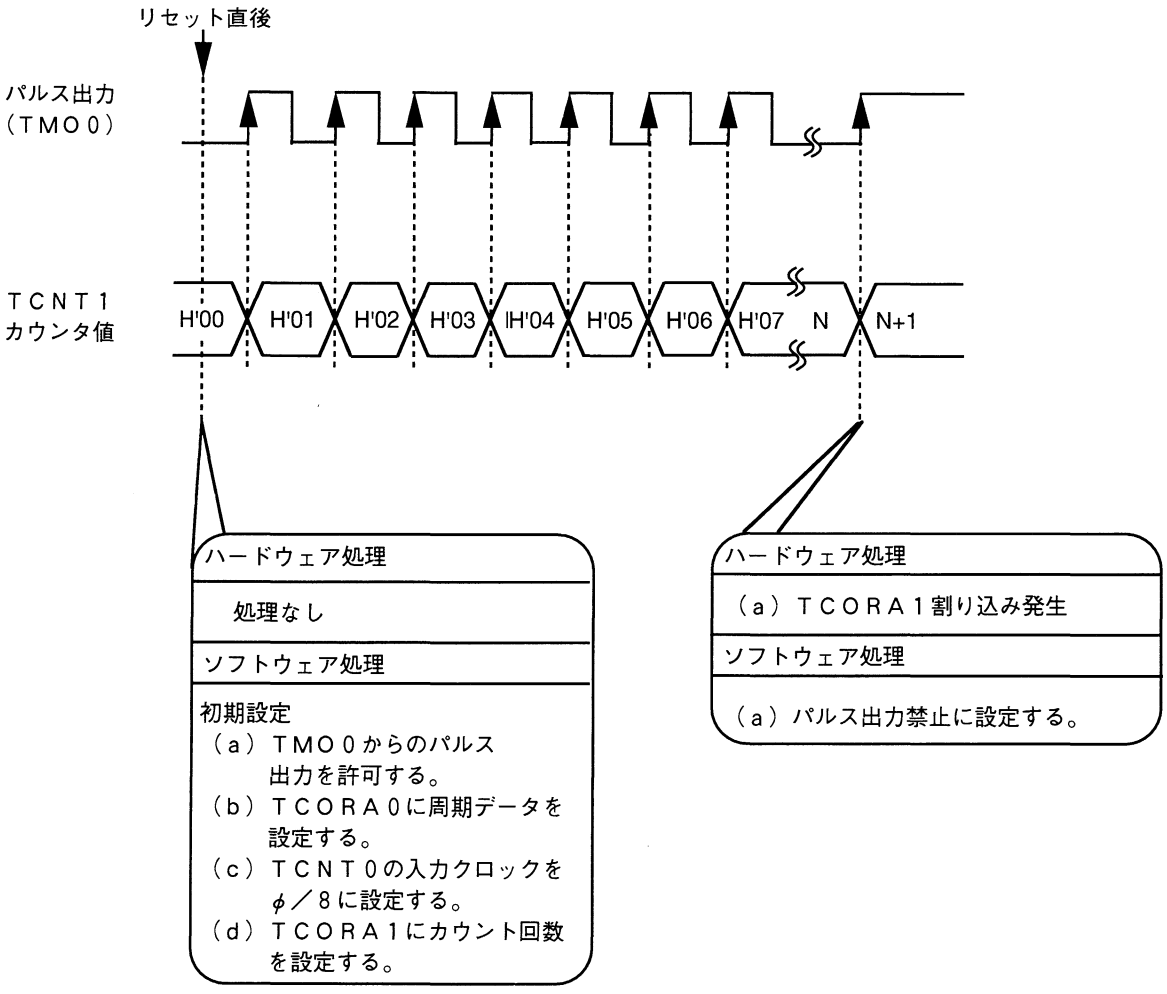


図3　パルス数測定動作原理

パルス数の測定	MCU	H8S／2655	使用機能	8ビットタイマ

ソフトウェア説明

（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	pulsemn	8ビットタイマを初期設定する
パルス出力停止	pulend	TCORA1割り込みで起動し、TCNT1に設定されたパルス数を出力引数に設定する

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
pulse_cycle	パルスの周期を設定する	1バイト	メインルーチン	入力
pulse_count	パルスのカウント数を設定する	1バイト	メインルーチン	入力

（3）使用内部レジスタ説明

レジスタ名	機能	使用モジュール名
TCORA0	コンペアマッチA発生用	メインルーチン
TCORB0	コンペアマッチB発生用	メインルーチン
TCSR0	コンペアマッチA毎に1出力：コンペアマッチB毎に0出力	メインルーチン
TCR0	コンペアマッチAによりカウンタクリア	メインルーチン
	入力クロックの選択（ϕ/8）	
TCR1	チャネル0のコンペアマッチA発生回数をカウントする	メインルーチン
	コンペアマッチAによりカウンタクリア	
	コンペアマッチ（A）割込みを許可に設定する	
TCORA1	コンペアマッチA発生用	ﾒｲﾝﾙｰﾁﾝ、測定終了
MSTPCR	8ビットタイマのモジュールストップモードを解除する	メインルーチン

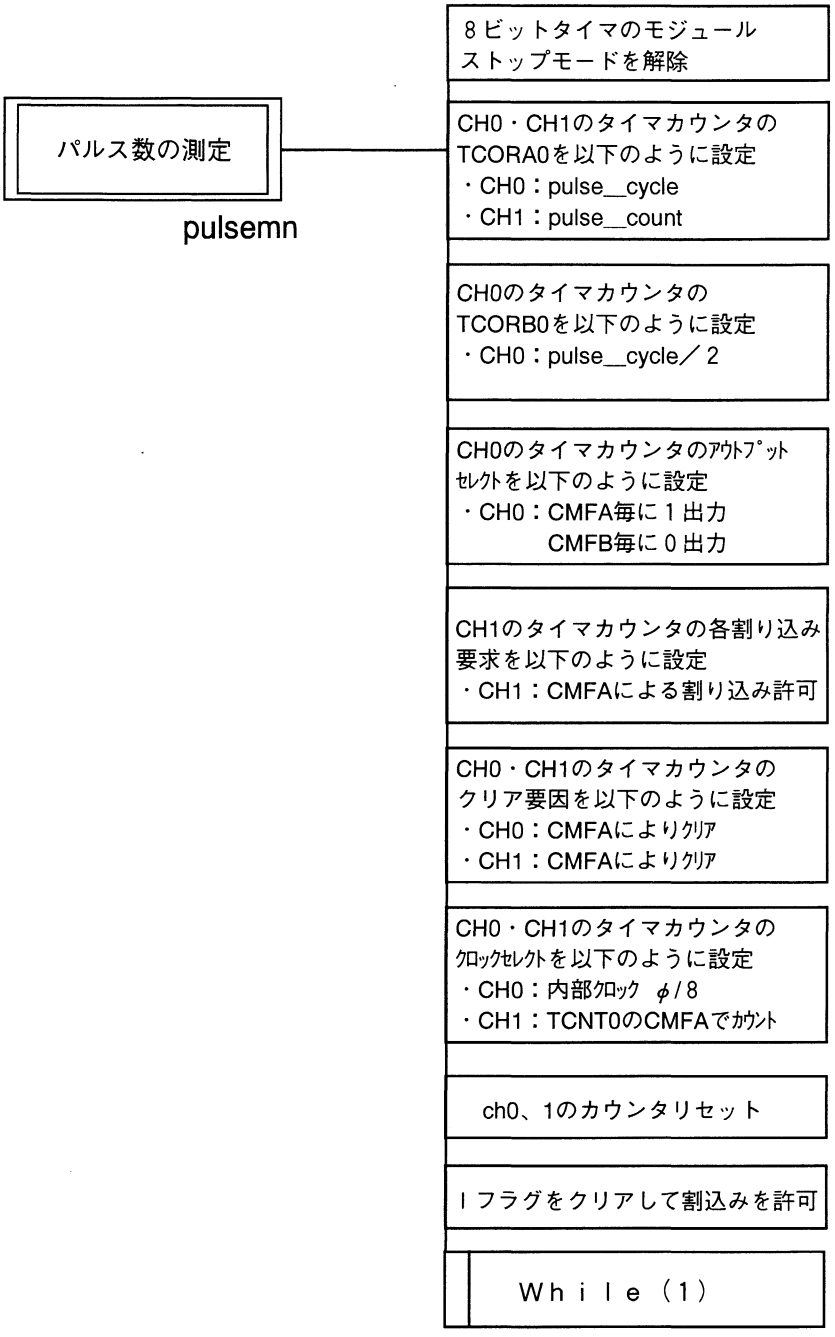
（4）使用RAM説明

本タスク例では引数以外のRAMは使用していません。

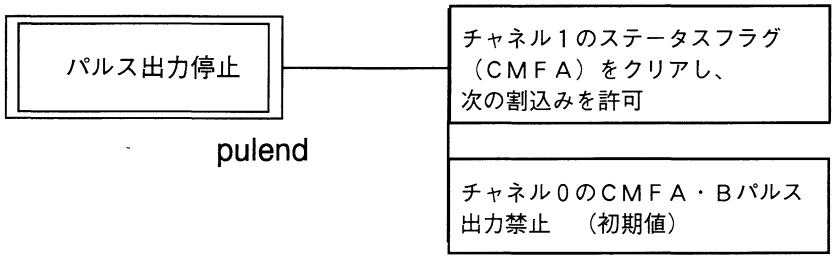
パルス数の測定	MCU	H8S／2655	使用機能	8ビットタイマ

PAD

（1）メインルーチン



（2）パルス出力停止



パルス数の測定	MCU	H8S／2655	使用機能	8ビットタイマ

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTOCOL                     */
/*****************************************/
void pulsemn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
# define pulse_cycle (*(unsigned char * ) 0xffec00)
# define pulse_count (*(unsigned char * ) 0xffec01)

/*****************************************/
/*      MAIN PROGRAM : pulsemn           */
/*****************************************/
void pulsemn(void)
{
        MSTPCR = 0xefff;                /* disable module stop mode*/

        TCORA0 = pulse_cycle;           /* set pulse cycle time */
        TCORB0 = pulse_cycle/2;         /* set "low"pulse time */
        TCORA1 = pulse_count;           /* set pulse counter */

        TCSR0 = 0x06;                   /* initialize TCSR0 */
        TCSR1 = 0x10;                   /* initialize TCSR1 */

        TCR0 = 0x09;                    /* Initialize TCR0 */
        TCR1 = 0x4c;                    /* initialize TCR1 */

        TCNT0 = 0;                      /* reset counter */
        TCNT1 = 0;

        set_imask_ccr(0);

        while(1);                       /* loop */
}

/*****************************************/
/*      NAME : pulend(output disble)     */
/*****************************************/
#pragma interrupt (pulend)
void pulend(void)
{
          TCSR0 = 0;            /* output disable */
}
```

4．応用編

目　次

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

仕様

図1に示すように外部信号の立ち上がりエッジを検出するごとに、12ビットのデータを出力します。

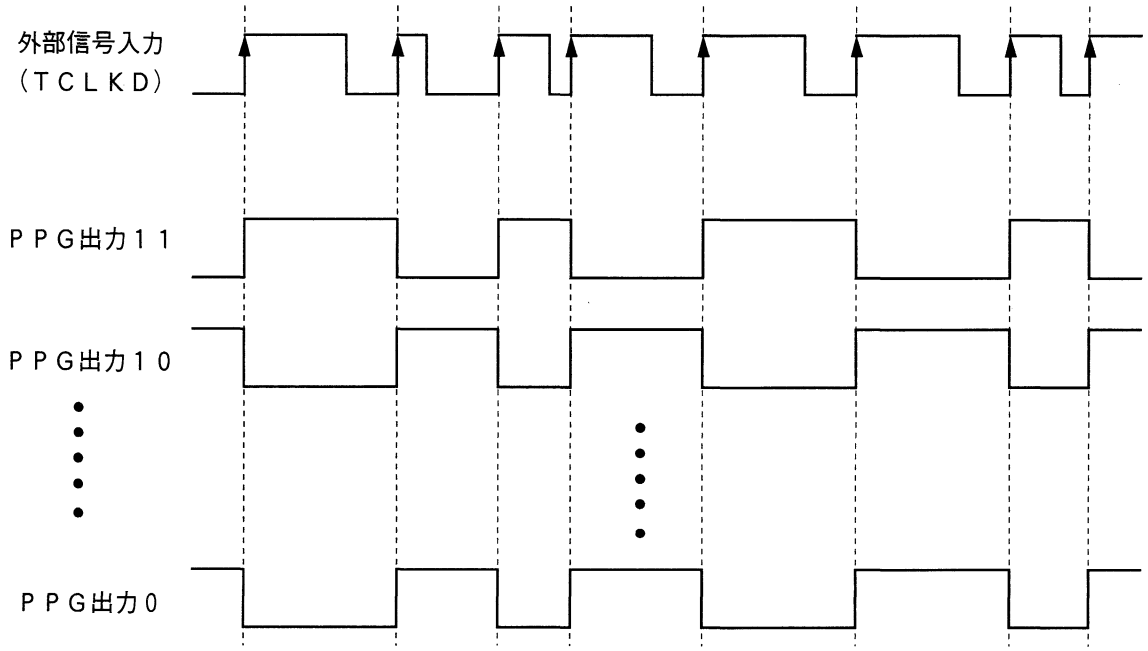


図1　12ビットデータ出力例

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

使用機能説明

（1）図2に本タスクで使用する内蔵機能のブロック図を示します。
本タスク例は、H8S／2655の機能を以下に示すように使用して高速にデータを出力します。

【出力パターンデータテーブル】
PPGから出力するデータのパターンをRAM上に設定します。
【TPU】
コンペアマッチAが発生するごとにDMAC0AおよびPPGを起動します。
（TGRAにはH’0000、外部クロックの立ち上がりエッジでカウントアップに設定）
【DMAC0A】
TPUのコンペアマッチAで起動し、出力パターンデータテーブルから出力データを
NDRに転送します。
【PPG】
TPUのコンペアマッチAで起動し、12ビットのデータを出力します。

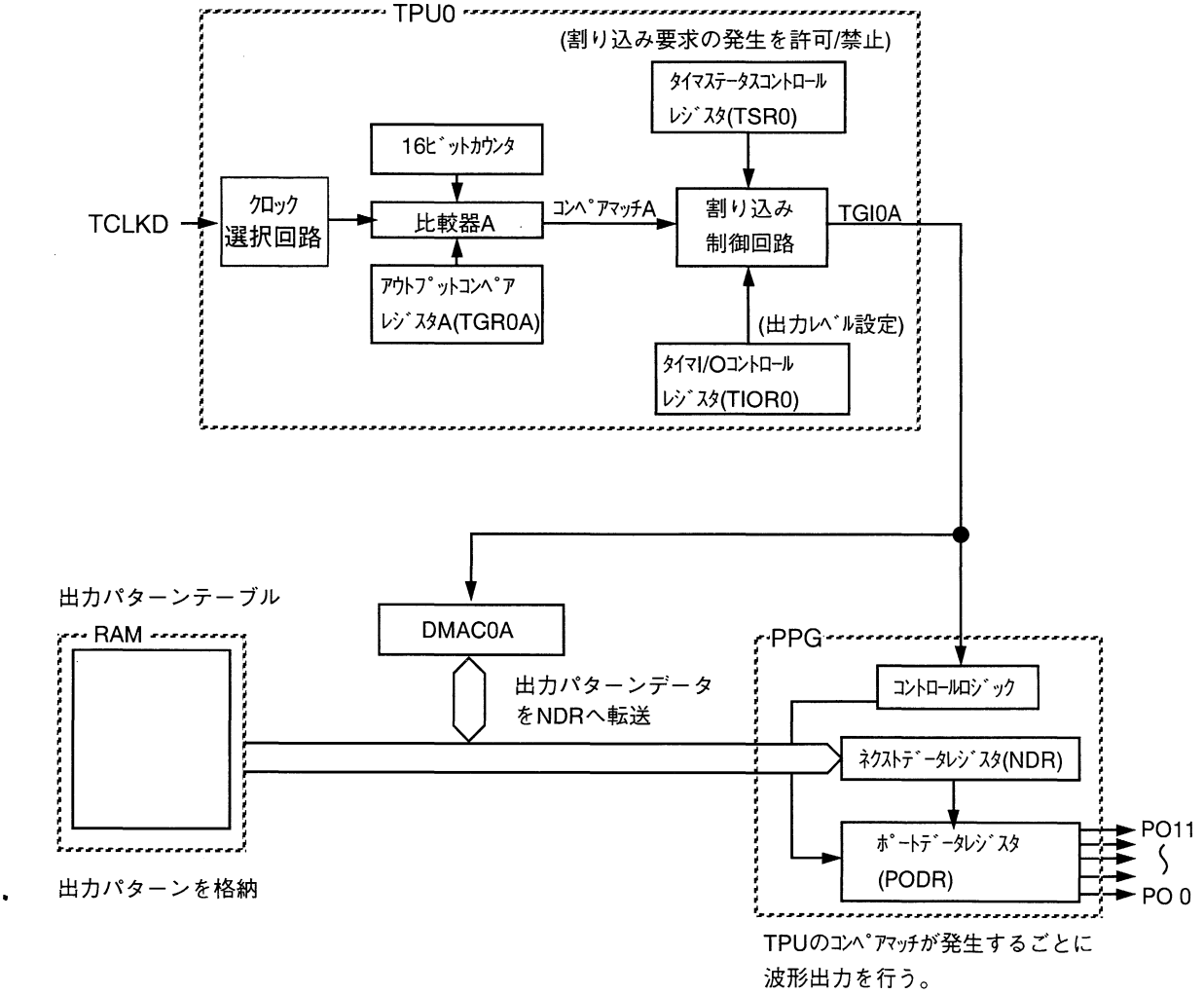


図2　高速データ出力ブロック図

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、高速データ出力を行います。

表1　H8S／2655機能割り付け

H8S／2655機能		機能
TPU0	TCLKD	外部信号入力端子
	TCNT0	16ビットカウンタ
	TGR0A	アウトプットコンペアレジスタ
	TCR0	カウントクロックの選択、カウンタクリア要因の選択
	TIOR0	TGR0Aをアウトプットコンペアレジスタに設定
	TSR0	コンペアマッチの発生やオーバフローの発生を示す
DMAC0A	DMABCR H／L	各チャネルの動作を制御
	DMACR0A	DMAC0Aの動作を制御
	MAR0A	出力パターンデータテーブルの先頭アドレスを設定
	IORA0A	NDRのアドレスを設定
	ETCR0A	転送回数を設定
PPG	PODRH	PPG出力グループ2、3の出力データを格納
	PODRL	PPG出力グループ0、1の出力データを格納
	PCR	PPG出力の出力トリガ信号を選定
	NDERH	PPG出力PO15～PO8を許可
	NDERL	PPG出力PO7～PO0を許可
	NDRH	PPG出力の次に出力するデータを格納
	NDRL	PPG出力の次に出力するデータを格納
	PO11～PO8	グループ2のパルス出力端子
	PO7～PO4	グループ1のパルス出力端子
	PO3～PO0	グループ0のパルス出力端子

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理により高速にデータの出力を行います。

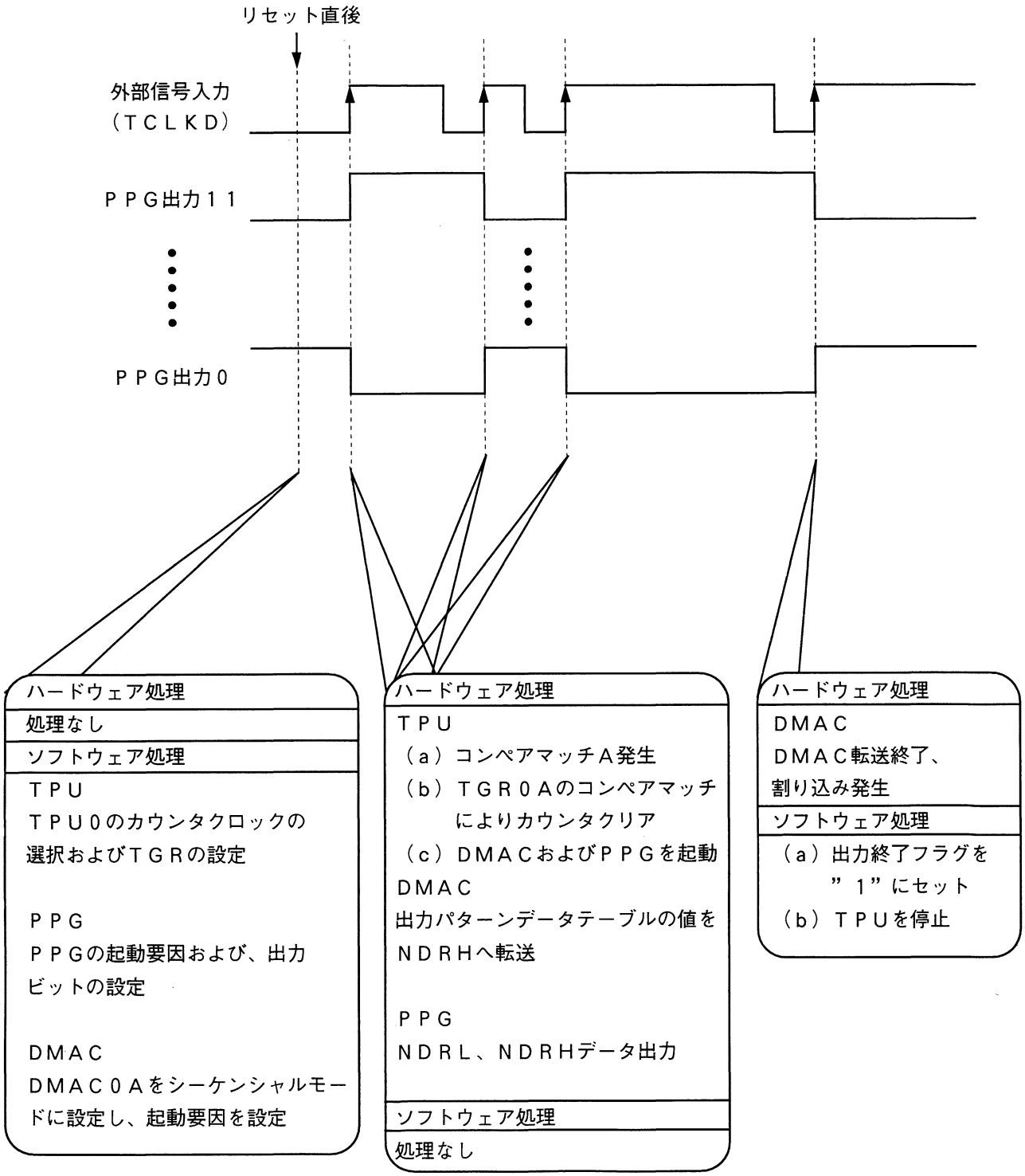


図3　高速データ出力動作原理

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

ソフトウェア説明

（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	STPCMN	TPU、PPGおよびDMAC0Aの初期設定を行う。
DMA割り込み	DMAEND	出力終了フラグをセットする。

（2）引数の説明

ラベル名、レジスタ名	機能	データ長	使用モジュール名	入出力
dma_fla	出力が全て終了したことを示すフラグ 0：全ビット出力中 1：全ビット出力終了	unsigned char	メインルーチン	入力
			DMA割り込み	出力

（3）使用内部レジスタ

内蔵機能	レジスタ名	機能
TPU0	TGR0A	アウトプットコンペアの値（H’ 0000）を設定
	TCR0	TPUを以下のように設定する ・TGR0Aのコンペアマッチでカウンタクリア ・外部信号の立ち上がりエッジでカウント ・TCLKD端子でカウント
	TIOR0	TGR0Aをアウトプットコンペアレジスタに設定し、端子出力を禁止
	TIER0	TGI0A割り込みを許可
	TSTR	TCNT0のカウント動作を開始
PPG	PODRH	PO11～PO8までの出力データを格納
	PODRL	PO7～PO0までの出力データを格納
	TPMR	PO11～PO0までを通常動作に設定
	TPCR	PO11～PO0の出力トリガをTPU0のコンペアマッチに設定
	NDERH	PPG出力PO11～PO8を許可
	NDERL	PPG出力PO7～PO0を許可
	NDRL	次の出力パターンデータを格納
	NDRH	次の出力パターンデータを格納
DMAC0A	DMACR0A	データサイズをワードサイズに設定 MARをインクリメントに設定 データ転送をシーケンシャルモードに設定 起動要因をTPU0のコンペアマッチAに設定
	DMABCR H／L	データ転送、転送終了割り込み　許可／禁止の設定
	MAR0A	出力パターンデータテーブルの転送元アドレスを設定
	IOAR0A	NDRHのアドレス（転送先）を設定
	ETCR0A	転送回数を設定
MSTPCR		DMAC、TPUおよびPPGのモジュールストップモードを解除する

（4）使用RAM説明

本タスクではRAMを使用しません。

（5）データテーブルの説明

テーブル名	機能	データ長	データ容量
opat_tab	PPGより出力するデータを格納	unsigned short	30byte

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

PAD

（1）メインルーチン

メインルーチン

STPCMN

DMAC、TPU、PPGのモジュールストップモードを解除

P20～P27およびP10～P13を出力端子に設定

TGR0Aにアウトプットコンペアの値H’0000を設定

TGR0Aのコンペアマッチでカウンタクリア、外部信号の立ち上がりエッジでカウントアップ

初期出力データをPODRに設定

NDERHおよびNDERLの出力を行うビットを”1”に設定

PPGの出力トリガをTPU0に設定

次の出力データをNDRに設定

DMAC0Aの転送元アドレスを出力パターンデータテーブルの先頭アドレスに設定

DMAC0Aの転送先アドレスをNDRHのアドレスに設定

転送回数を15回に設定

①

①

データサイズをワードサイズ、MARをインクリメント、データ転送をシーケンシャルモード、起動要因をTPU0のコンペアマッチA

DMABCRLをリード

転送終了割り込みを許可、DMAC0Aを起動

dma_flaをクリア

TIER0でTGI0A割り込みを許可

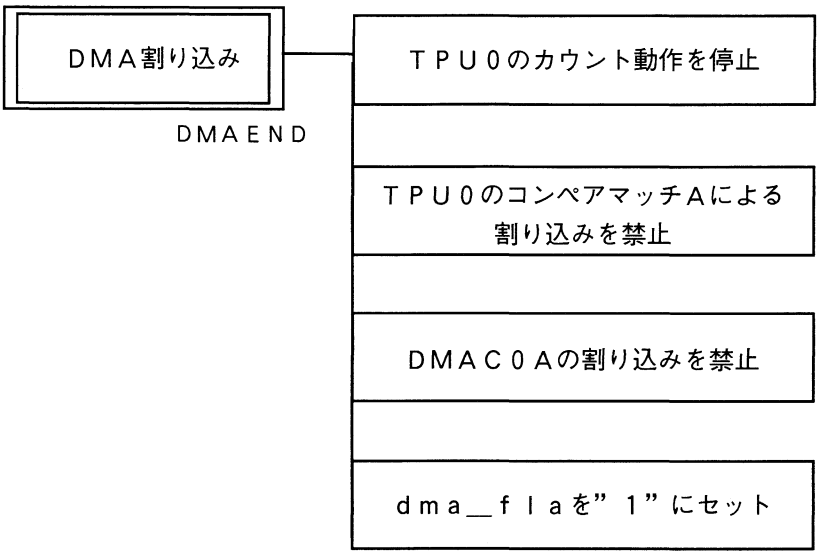
Iフラグをクリアして割り込みを許可

ch0のカウンタをスタート

UNTIL dma_flaは”1”か？

While（1）

（2）DMA割り込み


DMA割り込み
DMAEND
TPU0のカウント動作を停止
TPU0のコンペアマッチAによる
割り込みを禁止
DMAC0Aの割り込みを禁止
dma__fl aを”1”にセット

高速データ出力	MCU	H8S／2655	使用機能	TPU、PPG、DMAC

プログラムリスト

```
#include <machine.h>
#include "H8S.H"
/**********************************/
/*           PROTOCOL             */
/**********************************/
void STPCMN(void);
#pragma interrupt (DMAEND)
/**********************************/
/*     SYMBOL DEFINITIONS         */
/**********************************/

# define opt_tab   ((unsigned short * )0xffec00) /* Output data table */
# define dma_fla   (*(unsigned char * )0xffecle) /* DMAC end flag */

/**********************************/
/*     MAIN PROGRAM: STPCM        */
/**********************************/
void STPCMN(void)
{
        MSTPCR = 0x17ff;        /* Disable module(DMA,TPU,PPG) stop mode*/

        P1DDR = 0x4f;           /* P1:output port */
        P2DDR = 0xff;           /* P2:output port */

        TIOR0H = 0x00;          /* Initialize TIOR0H */
        TGR0A = 0x0000;         /* Set non overlap time */
        TPU_TCR0 = 0x27;        /* Initialize TCR0 */

        PODRH = 0x00;           /* Output first data */
        PODRL = 0x00;           /* Output first data */
        NDERH = 0x0f;           /* Enable next data output */
        NDERL = 0xff;           /* Enable next data output */
        PCR = 0x00;             /* Output toriga TPU0'S compare match */
        NDRH = 0xff;            /* Set second output data */
        NDRL = 0xff;            /* Set second output data */

        MAR0A_W = opt_tab;      /* Set base address */
        IOAR0A = 0xff4c;        /* Set excute address */
        ETCR0A = 0x000f;        /* Set excute count */
        DMACR0A = 0x88;         /* Initialize DMACR0A */
        DMABCRH = 0x01;         /* Initialize DMABCRH */
        DMABCRL |= 0x11;        /* Initialize DMABCRL */

        dma_fla = 0;            /* Clear dma_fla */
        TIER0_BP.TGIEA0=1;      /* Enable TGI0EA interrupt */
        set_imask_ccr(0);       /* Enable interrupt */
        TSTR = 0x01;            /* Start TCNT0 */
        while(dma_fla==0);      /* DMAC end? */
        while(1);               /* Loop */
}

/**********************************/
/*   INTERRUPT PROGRAM: DMAEND    */
/**********************************/
void DMAEND(void)
{
        TSTR_BP.CST0 = 0;       /* Stop TCNT0 */
        TIER0_BP.TGIEA0=0;      /* Disable timer compare match interrupt */
        DMABCR_BP.DTIE0A=0;     /* Disable DMAC end interrupt */
        DMABCR_BP.DTE0A=0;      /* Disable data translation */
        dma_fla =1;             /* Set dma_fla */
}
```

4．2　ＳＣＩ連続送受信

ＳＣＩ連続送受信	MCU	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、DMAC

仕様

（1）Ｈ８Ｓ／２６５５のＳＣＩをクロック同期式モードに設定し、Ｈ８／３３１４と４８バイトのデータを連続送信および連続受信します。

（2）ＤＭＡＣを使用し、ＣＰＵを介さずにデータをメモリ→ＴＤＲ、ＲＤＲ→メモリへ転送します。

（3）送信側がクロックマスタになります。

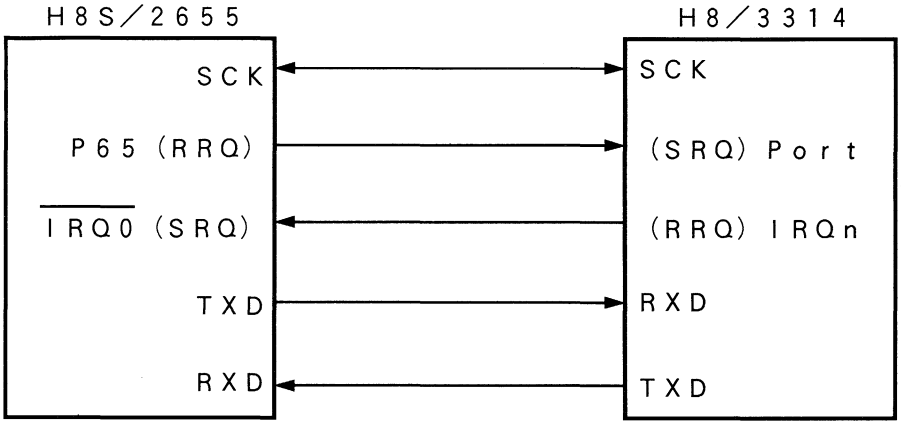


図１　Ｈ８Ｓ／２６５５によるクロック同期式ＳＣＩブロック図

ＳＣＩ連続送受信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、ＤＭＡＣ

使用機能説明

（１）図２に本タスクで使用するＨ８Ｓ／２６５５の内蔵機能を示します。図２に示すようにＤＭＡＣ０Ａ、ＤＭＡＣ０ＢおよびＳＣＩ１を使用して高速にシリアル通信を行います。

【データバッファ】
送受信データを格納するバッファＲＡＭです。

【ＤＭＡＣ０Ａ】
シーケンシャルモードで動作します。ＳＣＩ送信完了割り込みで起動し、送信用データバッファの内容をＳＣＩに転送します。

【ＤＭＡＣ０Ｂ】
シーケンシャルモードで動作します。ＳＣＩ受信完了割り込みで起動し、受信データを受信用データバッファへに転送します。

【ＳＣＩ１】
シリアルデータの送信および受信を行います。

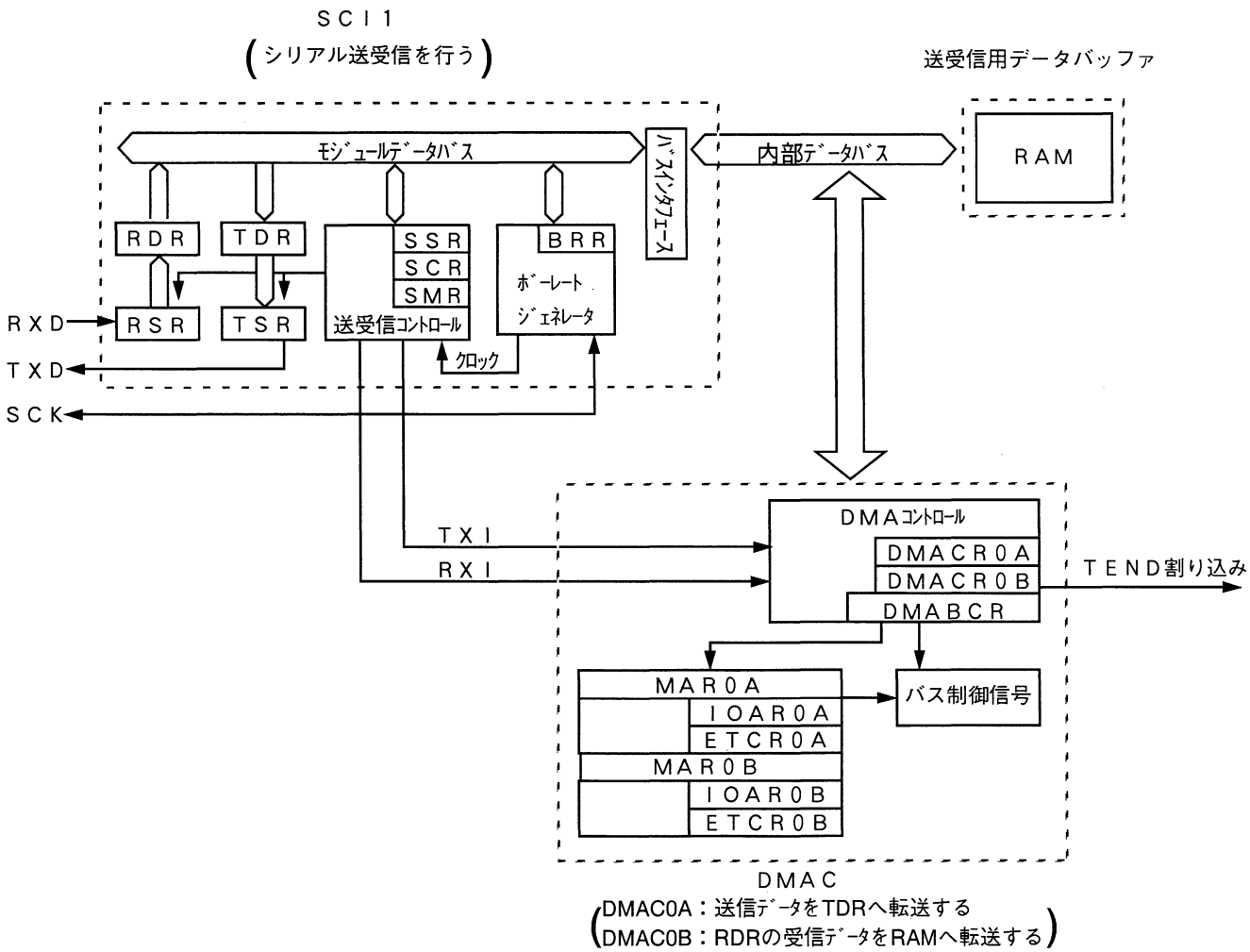


図２　ＳＣＩ連続送受信ブロック図

ＳＣＩ連続送受信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、ＤＭＡＣ

使用機能説明

（２）表１に本タスク例の機能割り付けを示します。表１に示すようにＨ８Ｓ／２６５５の機能を割り付け、送受信ﾃﾞｰﾀをＣＰＵを介さずに転送します。

表１　Ｈ８Ｓ／２６５５機能割り付け

H8S/2655機能		機能
割り込みコントローラ	ＩＳＣＲＬ	ＩＲＱ０入力の立ち下がりエッジで割り込み発生を選択
	ＩＥＲ	ＩＲＱ０の割り込みを許可
	ＩＳＲ	ＩＲＱ０割り込み要求のステータスを示す。
ＳＣＩ１	ＳＣＫ１	転送クロックを送信する。また、受信時には転送クロックを受信
	ＲＸＤ１	受信データ入力端子
	ＴＸＤ１	送信データ出力端子
	ＳＭＲ１	ＳＣＩをクロック同期式モードに設定
	ＳＣＲ１	送信および受信の設定
	ＳＳＲ１	受信、送信のステータスを示す
	ＲＤＲ１	受信したデータが格納
	ＴＤＲ１	送信するデータを設定
	ＢＲＲ１	転送レートを設定
ポート６	Ｐ６ＤＤＲ	ポート６の入出力を設定
	Ｐ６ＤＲ	ＲＲＱの送信を行う
ＤＭＡＣ	ＤＭＡＢＣＲ	各ﾁﾁャネルの動作を制御
	ＤＭＡＣＲ０Ａ	ＤＭＡＣ０Ａの動作を制御
	ＭＡＲ０Ａ	転送元アドレス（データバッファ）を設定
	ＩＯＡＲ０Ａ	転送先アドレス（ＴＤＲ）を設定
	ＥＴＣＲ０Ａ	転送回数を設定
	ＤＭＡＣＲ０Ｂ	ＤＭＡＣ０Ｂの動作を制御
	ＭＡＲ０Ｂ	転送先アドレス（データバッファ）を設定
	ＩＯＡＲ０Ｂ	転送元アドレス（ＲＤＲ）を設定
	ＥＴＣＲ０Ｂ	転送回数を設定

（１）データ送信

図３にデータ送信時の動作原理を示します。図３に示すタイミングでＩ／Ｏポート及びクロック同期式ＳＣＩを制御し、インタフェースを行います。

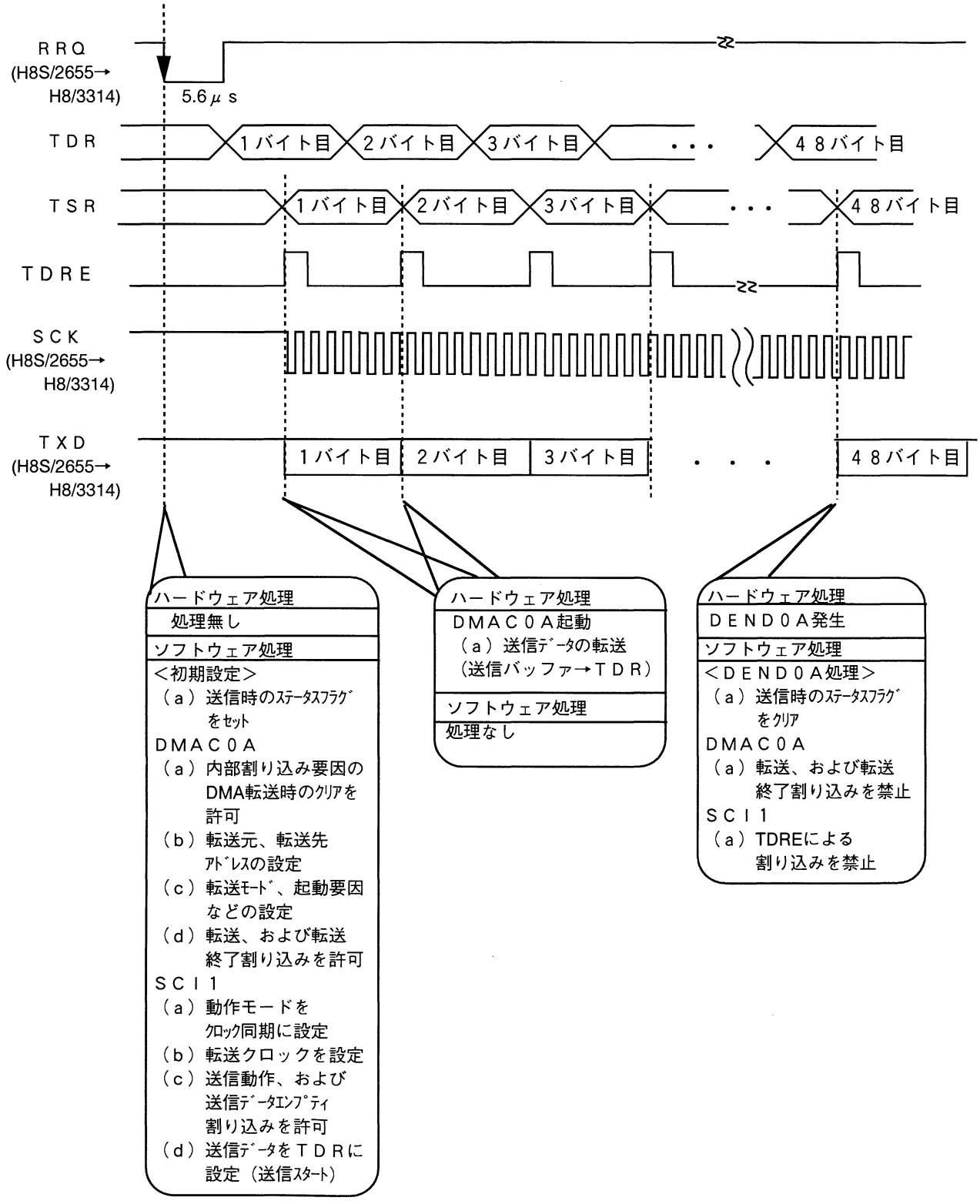


図３　データ送信動作原理

動作説明

（２）データ受信

図４にデータ受信時の動作原理を示します。図４に示すタイミングでＩ／Ｏポート及びクロック同期式ＳＣＩを制御し、インタフェースを行います。

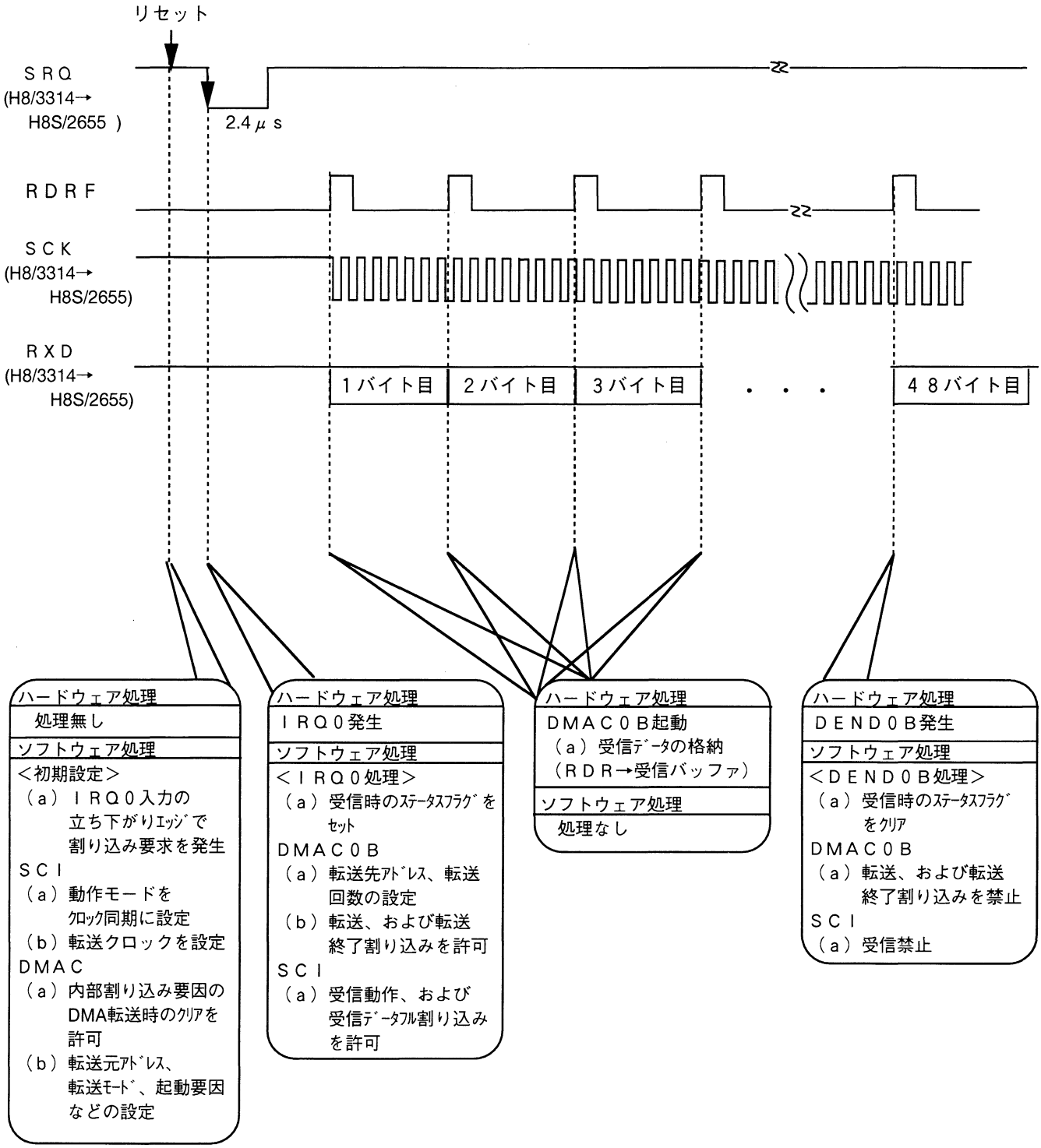


図４　データ受信動作原理

ＳＣＩ連続送受信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、ＤＭＡＣ

ソフトウェア説明

（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	hiscimn	Ｉ／Ｏポート、ＳＣＩおよびＤＭＡＣの初期設定を行う
データ送信	txstart	ＤＭＡＣを転送許可して、ＳＣＩの送信動作を開始する
データ受信	rxstart	ＩＲＱ０割り込みで起動し、ＤＭＡＣ転送を許可しＳＣＩの受信動作を開始する
送信完了	txend	ＤＭＡＣ０Ａ転送終了割り込みで起動し、ｓｔａｔ＿ｔｘのｸﾘｱおよび送信処理を禁止する
受信完了	rxend	ＤＭＡＣ０Ｂ転送終了割り込みで起動し、ｓｔａｔ＿ｒｘのｸﾘｱおよび受信処理を禁止する

（２）引数の説明

レジスタ名	機能	データ長	使用モジュール名	入出力
stat_tx	送信中であることを示すフラグ	unsigned char	データ送信 データ受信	出力 入力
stat_rx	受信中であることを示すフラグ	unsigned char	データ送信 データ受信	入力 出力

（3）使用内部レジスタ

内蔵機能	レジスタ名	機能
ＳＣＩ１	ＳＭＲ１	ＳＣＩを以下のように設定 ・ＳＣＩの動作モードをクロック同期式モードに設定 ・ボーレートジェネレータのクロックソースをφに設定
	ＳＣＲ１	送信時と受信時にＳＣＩを以下のように設定 送信動作：送信データエンプティ割り込みを許可 送信動作を許可 ＳＣＫ端子を同期クロック出力 受信動作：受信データフル割り込みを許可 受信動作を許可 ＳＣＫ端子を同期クロック入力
	ＳＳＲ１	送信動作：ＴＤＲＥをクリアして送信動作を開始 受信動作：ＲＤＲＦをクリアして受信動作を許可
	ＲＤＲ１	受信したデータを格納
	ＴＤＲ１	送信するデータを設定
	ＢＲＲ１	転送レートを設定
ＤＭＡＣ	ＤＭＡＢＣＲ	ＤＭＡＣ０Ａ、０Ｂを以下のように設定 ・転送モードをショートアドレスモードに設定 ・内部割込み要因のＤＭＡ転送時のクリアを許可 ・データ転送、転送終了割込みを許可
	ＤＭＡＣＲ０Ａ	ＤＭＡＣ０Ａを以下のように設定 ・データサイズをバイトサイズに設定 ・ＭＡＲをインクリメントに設定 ・データ転送をシーケンシャルモードに設定 ・データ転送の方向を設定（ch0A：MAR→IOAR） ・起動要因をＳＣＩの送信完了割り込みに設定
	ＭＡＲ０Ａ	送信バッファのアドレスを設定
	ＩＯＡＲ０Ａ	ＴＤＲのアドレスを設定
	ＥＴＣＲ０Ａ	転送回数を設定
	ＤＭＡＣＲ０Ｂ	ＤＭＡＣ０Ｂを以下のように設定 ・データサイズをバイトサイズに設定 ・ＭＡＲをインクリメントに設定 ・データ転送をシーケンシャルモードに設定 ・データ転送の方向を設定（ch0B：IOAR→MAR） ・起動要因をＳＣＩの受信完了割り込みに設定
	ＭＡＲ０Ｂ	受信バッファのアドレスを設定
	ＩＯＡＲ０Ｂ	ＲＤＲのアドレスを設定
	ＥＴＣＲ０Ｂ	転送回数を設定
Ｉ／Ｏ	Ｐ６ＤＤＲ	ポート６の入出力を設定
	Ｐ６ＤＲ	ＲＲＱの送信を行う
割り込みコントローラ	ＩＥＲ	ＩＲＱ０割り込みを許可
	ＩＳＣＲ	ＩＲＱ０入力の立ち下がりエッジで割込み要求発生に設定
	ＩＳＲ	ＩＲＱ０入力のステータスを示す
ＭＳＴＰＣＲ		ＳＣＩ、ＤＭＡＣのモジュールストップモードを解除

（4）使用ＲＡＭ説明

レジスタ名	機能	データ長	使用モジュール名
buffer	送受信データを格納	４８バイト	データ送信

ＳＣＩ連続送受信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、ＤＭＡＣ

ＰＡＤ

（１）メインルーチン

SCI連続送受信

hiscimn

SCI、DMACのモジュールストップモードを解除

SMRを以下ように設定
・クロック同期式モード
・クロックソースをϕ

P6DDR・P6DRにH'20を設定し、RRQを出力に設定

DMABCRHを以下ように設定
・ショートアドレスモード
・内部割込みクリアの許可

転送先アドレス（IOAR0A）をTDR(H'ff83)に設定

転送元アドレス（MAR0A）をH'ffec01に設定

転送回数（ETCR0A）をH'2fに設定

DMACR0Aを以下のように設定
・データサイズをバイトサイズ
・MARをインクリメント
・データ転送をｼｰｹﾝｼｬﾙﾓｰﾄﾞ
・転送方向の設定
・起動要因をSCI送信完了割込み

転送元アドレス（IOAR0B）をRDR(H'ff85)に設定

転送元アドレス（MAR0B）をH'ffec00 (buffer)に設定

転送回数（ETCR0B）をH'30に設定

DMACR0Bを以下ように設定
・データサイズをバイトサイズ
・MARをインクリメント
・データ転送をｼｰｹﾝｼｬﾙﾓｰﾄﾞ
・転送方向の設定
・起動要因をSCI受信完了割込み

ライト/リード動作中であることを示すフラグクリア

1

1

立ち下がりエッジで割込み要求

IRQ0ステータスフラグクリア

IRQ0割込みを許可

Iフラグをクリアし割込みを許可

While(stat_rx) != 0
受信終了か？

データ送信

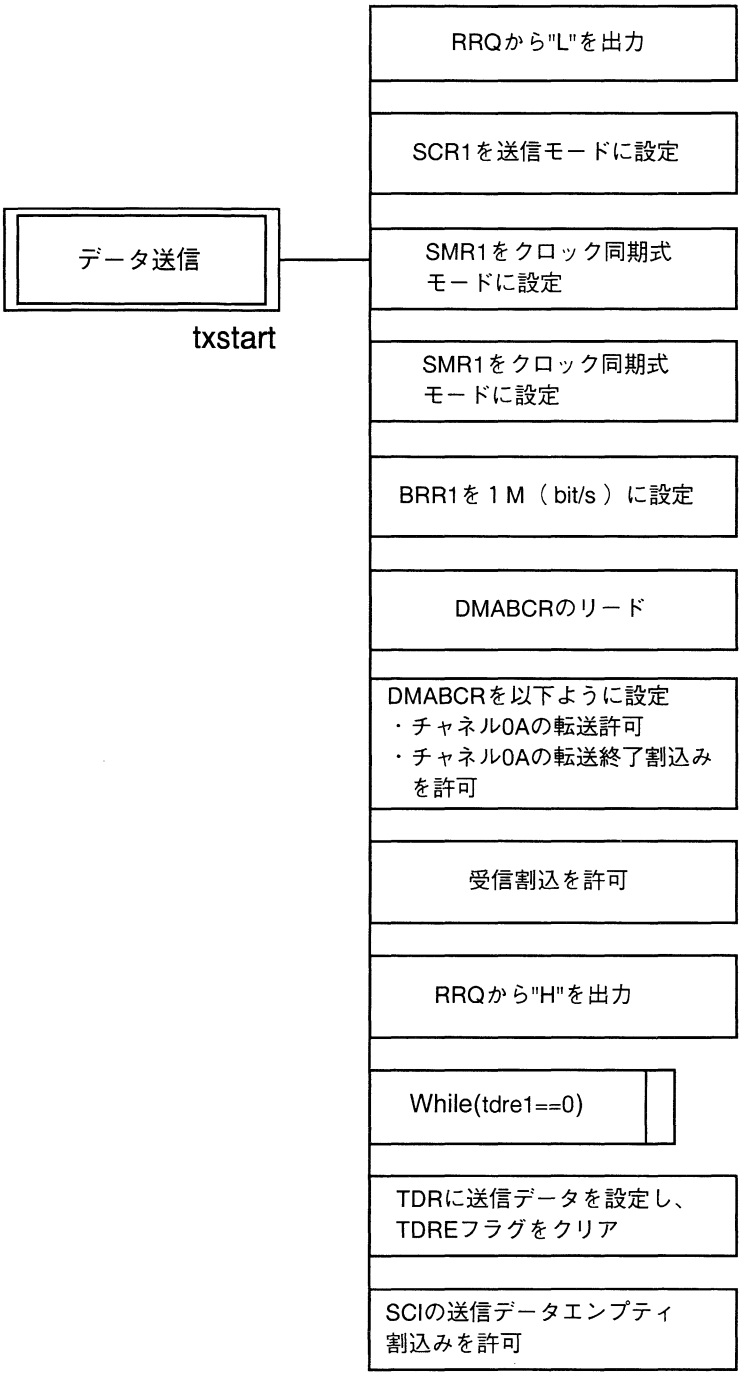
txstart

Ｗｈｉｌｅ（１）

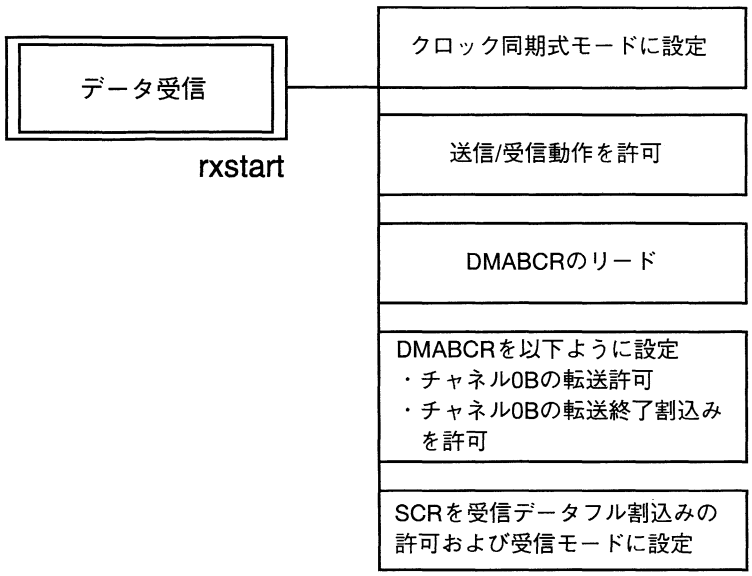
ＳＣＩ連続送受信	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＳＣＩ、ＤＭＡＣ

ＰＡＤ

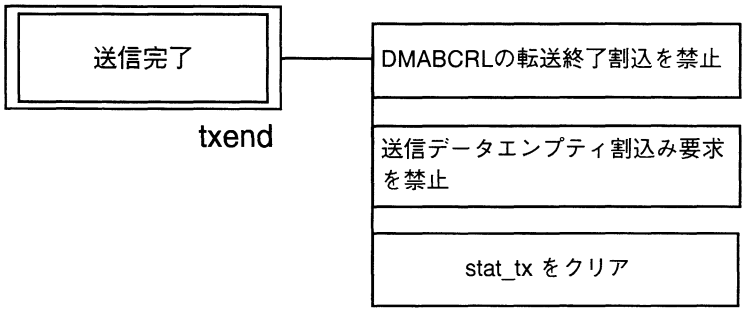
（２）データ送信



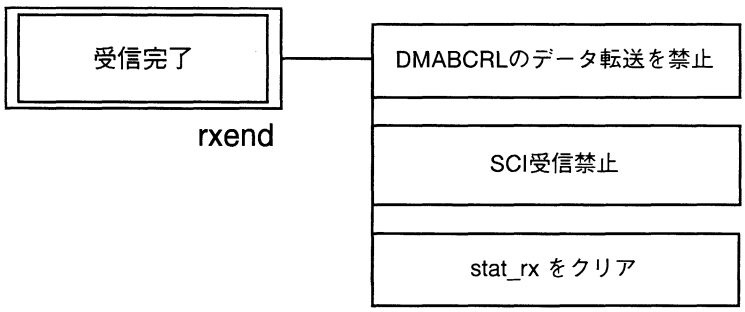
（３）データ受信


データ受信
rxstart
クロック同期式モードに設定
送信/受信動作を許可
DMABCRのリード
DMABCRを以下ように設定
・チャネル0Bの転送許可
・チャネル0Bの転送終了割込み
を許可
SCRを受信データフル割込みの
許可および受信モードに設定

（４）送信完了


送信完了
txend
DMABCRLの転送終了割込を禁止
送信データエンプティ割込み要求を禁止
stat_tx をクリア

（５）受信完了


受信完了
rxend
DMABCRLのデータ転送を禁止
SCI受信禁止
stat_rx をクリア

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void hiscimn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define buffer (*(volatile unsigned char *)0xffec00)
#define stat_rx (*(volatile unsigned char *)0xffec30)
#define stat_tx (*(volatile unsigned char *)0xffec31)

/*****************************************/
/*     MAIN PROGRAM : hiscimn            */
/*****************************************/
void hiscimn(void)
{
        MSTPCR = 0x7fbf;
        SCR1 = 0x00;
        P6DDR = 0x20;                   /* init port */
        P6DR = 0x20;

        DMABCRH = 0x03;
        IOAR0A = 0xff83;
        MAR0A = 0xffec01;
        ETCR0A = 0x2f;
        DMACR0A = 0x06;

        IOAR0B = 0xff85;
        MAR0B = (long)(&buffer);
        ETCR0B = 0x30;
        DMACR0B = 0x17;

        stat_rx = 0xff;
        stat_tx = 0xff;

        ISCRL = 0x01;
        ISR_BP.IRQ0F = 0;
        IER = 0x01;

        set_imask_ccr(0);

        while(stat_rx != 0);            /* receive complete? */
        txstart();

        while(1);
}
```

プログラムリスト

```
/*****************************************/
/* NAME : txstart(set transmit data )    */
/*****************************************/
txstart()
{
        P6DR_BP.RRQ = 0;
        SCR1 = 0x00;                    /* select clock mode */
        SMR1 = 0x80;                    /* init SCI1 */
        BRR1 = 0x04;                    /* set 1MBPS */
        DMABCRL |= 0x11;
        SCR1 = 0x20;
        P6DR_BP.RRQ = 1;

        while(SSR1_BP.TDRE1 == 0);
        TDR1 = buffer;                  /* set transmit data to TDR */
        SSR1_BP.TDRE1 = 0;              /* start transmit */
        SCR1_BP.TIE1 = 1;
}

/*****************************************/
/* NAME : rxstart(set SCI to receive operation) */
/*****************************************/
#pragma interrupt(rxstart)
void rxstart(void)
{
        SCR1 &= 0xcf;
        SMR1 = 0x80;                    /* init SCI1 */
        DMABCRL |= 0x22;
        SCR1 = 0x52;
}

/*****************************************/
/* NAME : txend(set transmit end flag)   */
/*****************************************/
#pragma interrupt(txend)
void txend(void)
{
        DMABCRL &= 0xee;
        SCR1_BP.TIE1 = 0;
        stat_tx = 0;
}

/*****************************************/
/* NAME : rxend(set receive end flag)    */
/*****************************************/
#pragma interrupt(rxend)
void rxend(void)
{
        SCR1 = 0x00;
        DMABCRL &= 0xdd;
        stat_rx = 0;
}
```

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

仕様

（1）図1に示すようにH8S／2655の内蔵機能のうち、TPU、PPG、DTCを使用し、4個の4相ステッピングモータを制御します。

（2）ステッピングモータは2相励磁方式で制御します。

（3）本タスクでは、ステッピングモータを停止→正転→停止→逆転→停止の動作を繰り返します。

（4）本タスクではソフトウェアの介在なしに、スルーアップ及びスルーダウンの処理を行います。

（5）ドライバの保護のため貫通電流防止期間を設けます。

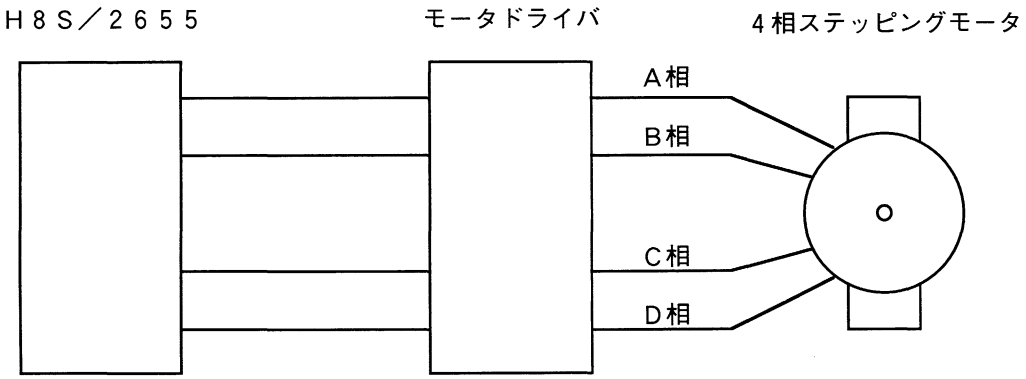


図1　4相ステッピングモータ制御回路図

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

考え方

（1）ステッピングモータ動作例

図2に2相励磁で4相ステッピングモータを動作させる例を示します。動作概要は以下の通りです。

（a）図2に示すようにパルスが”H”になっているときに、対応する相を励磁します。

（b）図2①では、D相とA相を同時に励磁します。このときロータは、D相とA相の中間に位置します。

（c）図2②では、A相とB相を同時に励磁します。このときロータは、A相とB相の中間に位置します。以下、2相励磁方式は、隣り合う2つの相（A、D相、A、B相、B、C相、C、D相）を励磁し、ロータを回転させます。

（d）逆転動作の場合は、D、C相→C、B相→B、A相→A、D相を順番に励磁することでステッピングモータを逆転させます。

（e）停止動作は、正転動作の最後の相または、逆転動作の最後の相を一定時間励磁し続けることでステッピングモータを停止させます。

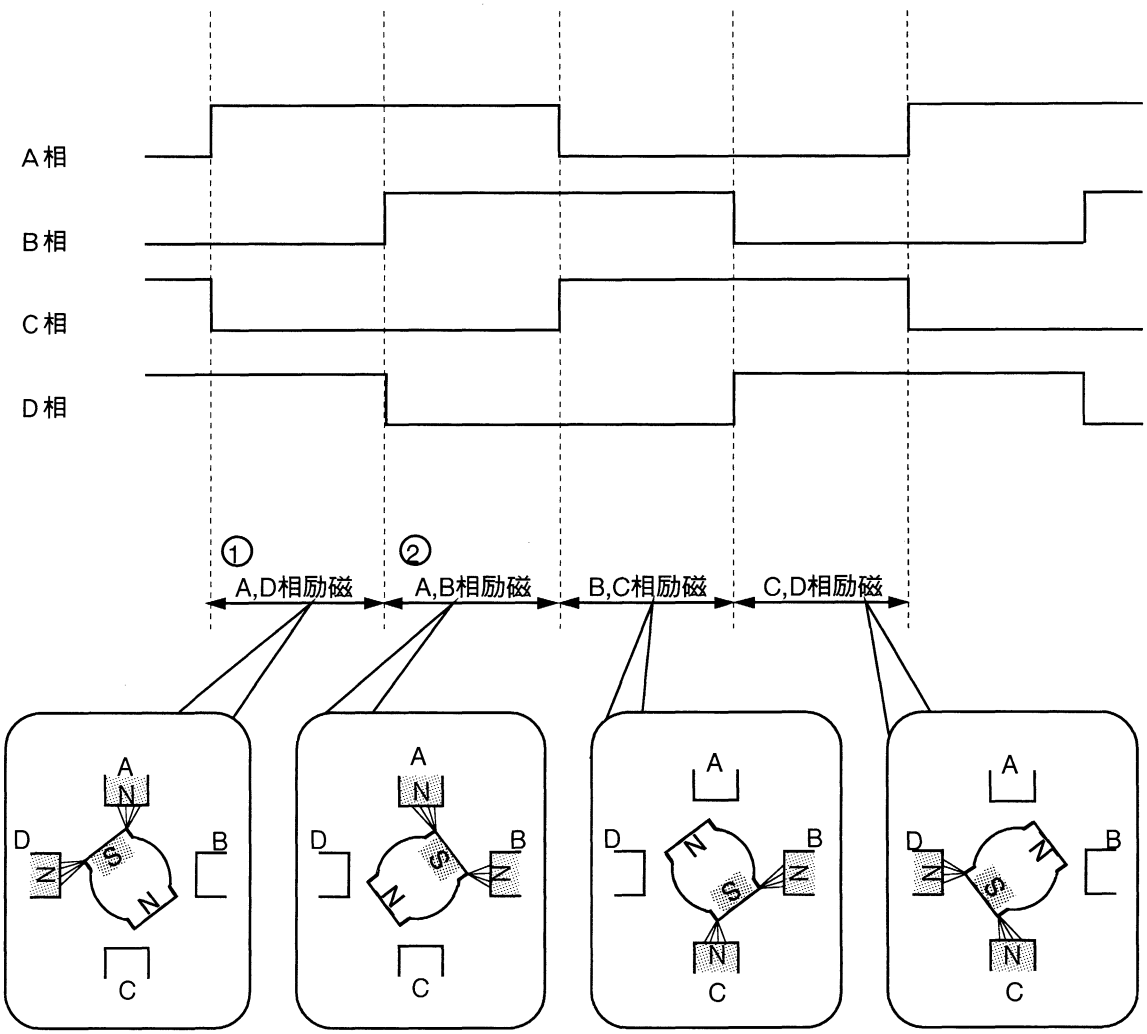


図2　ステッピングモータ動作例

（2）ノンオーバラップ時間

図3に示すように出力パターン切り換え時に貫通電流防止期間n（ノンオーバラップ時間）を挿入します。励磁相の切り換え時に生じるターンオフの遅れにより、ドライバが破壊する危険性があり、これを防ぐためにノンオーバラップ時間を挿入し、時間遅れをもたせて対策します。

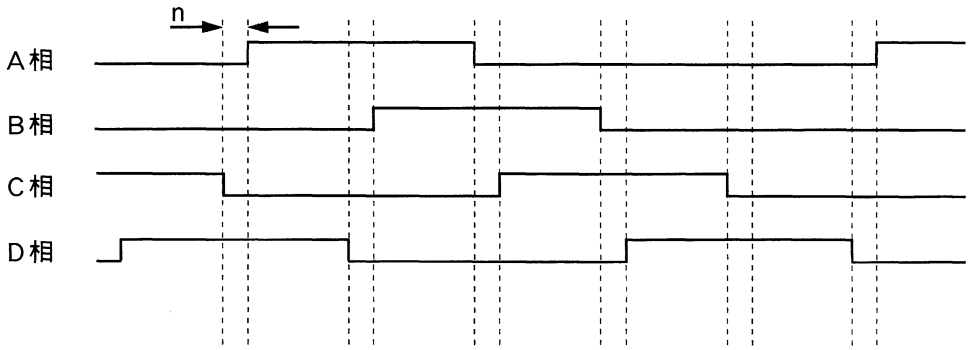


図3　ノンオーバラップ時間出力例

（3）スルーアップ、スルーダウン動作

図4に示すような加減速制御されたパルスを出力します。スルーアップ、スルーダウン動作を行うことにより、モータの脱調を防止します。

モータを動作させる際に、急に周期の短いパルスを出力すると、モータは負荷に追い付けず、回転しないことがあります。これを防止する対策としてスルーアップおよびスルーダウンを行います。

動作原理を以下に示します。

（a）徐々にパルス周期を短くし、設定した量のパルスを出力する。（スルーアップ）
（b）一定のパルス周期で設定した量のパルスを出力する。（定速）
（c）徐々にパルス周期を長くし、設定した量のパルスを出力する。（スルーダウン）

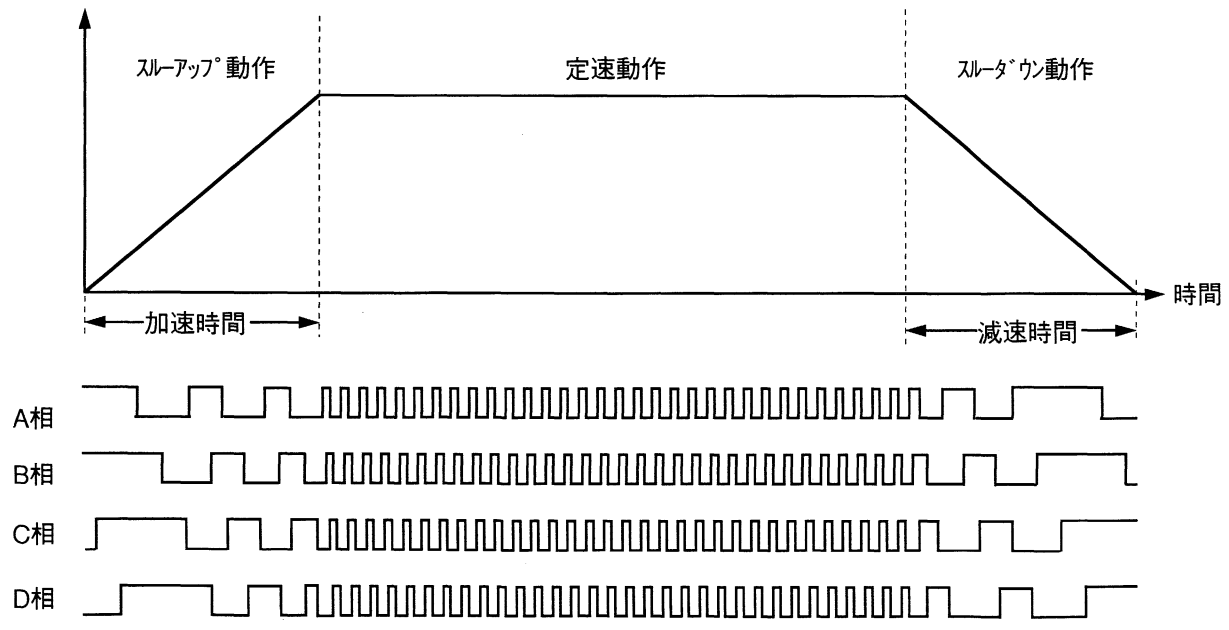


図4　スルーアップ、スルーダウン動作

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

使用機能説明

（1）本タスク例ではDTCの以下の機能を使用して、TPUのコンペアマッチAにより、4相パルス出力パターンをPPGに、パルス周期をTPUのTGRBに転送し、4相のパルスを出力します。

（a）図5に本タスク例で使用するH8S/2655内蔵機能のブロック図を示します。
本タスク例は以下の機能を使用し、モータ出力波形を生成します。

【DTC】
TPUのコンペアマッチAで起動します。
出力パターンデータテーブルから出力パターンをPPGのNDRに転送します。転送後、
チェイン転送で周期データテーブルからパルスの周期データをTPUのTGRBに転送します。

【TPU】
コンペアマッチA：DTCおよびPPGを起動します。
コンペアマッチB：タイマカウンタのクリアを行います。また、PPGを起動します。

【PPG】
本タスク例ではノンオーバラップ期間をもった16ビットパルスを出力します。
コンペアマッチB：波形が”H”から”L”に変化するパルスの出力を行います。
”L”から”H”に変化する出力は保留します。
コンペアマッチA：コンペアマッチBで保留された”L”から”H”の出力を行います。
（TGRAの設定値分遅延します）

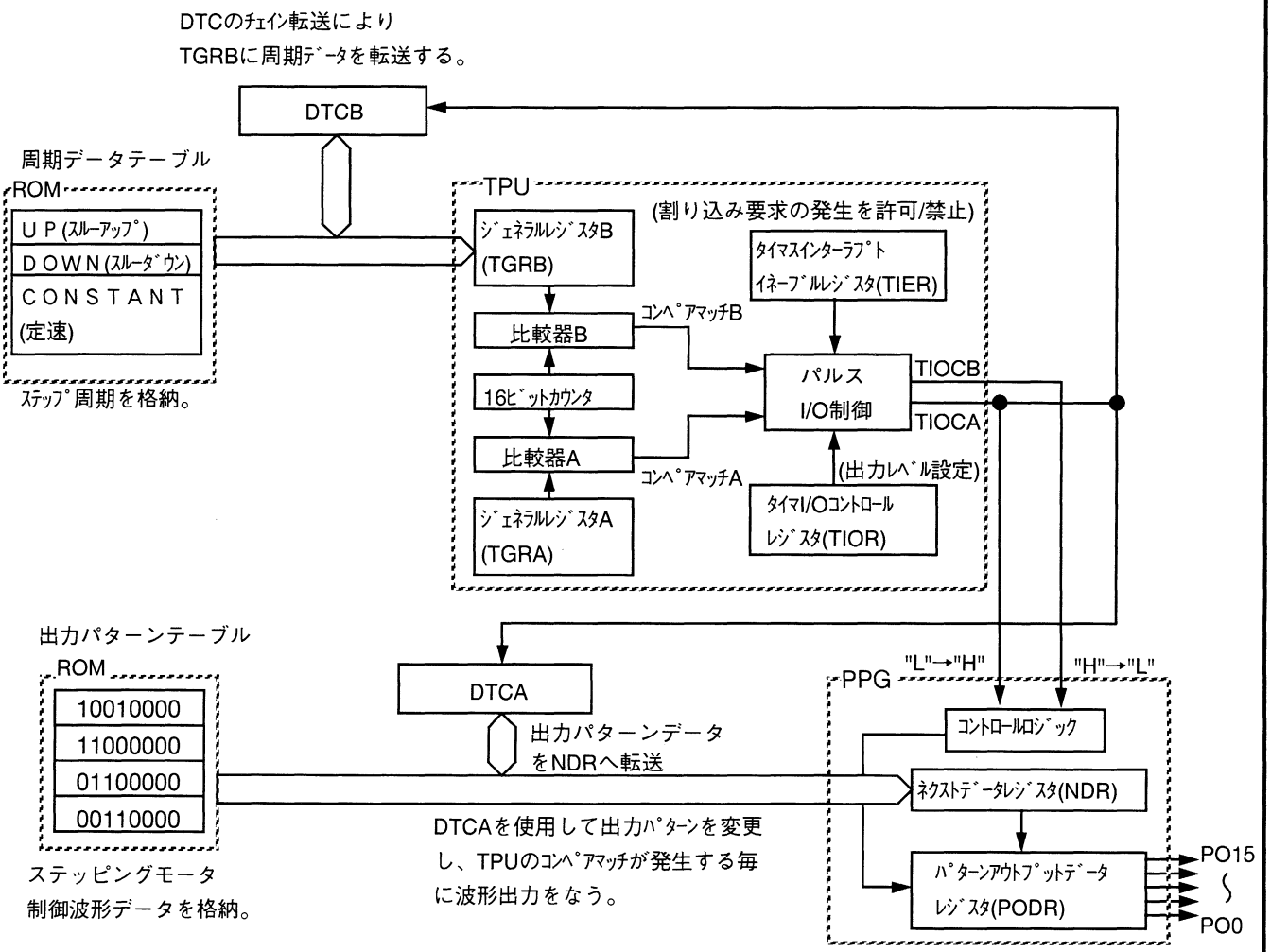


図5　4相ステッピングモータ制御ブロック図

（b）図6にDTCのベクタテーブルとメモリ上の配置例を示します。H'FFF800番地から、MRA、SAR、MRB、DAR、CRA、CRBの順にDTCのレジスタ情報を設定しておきます。

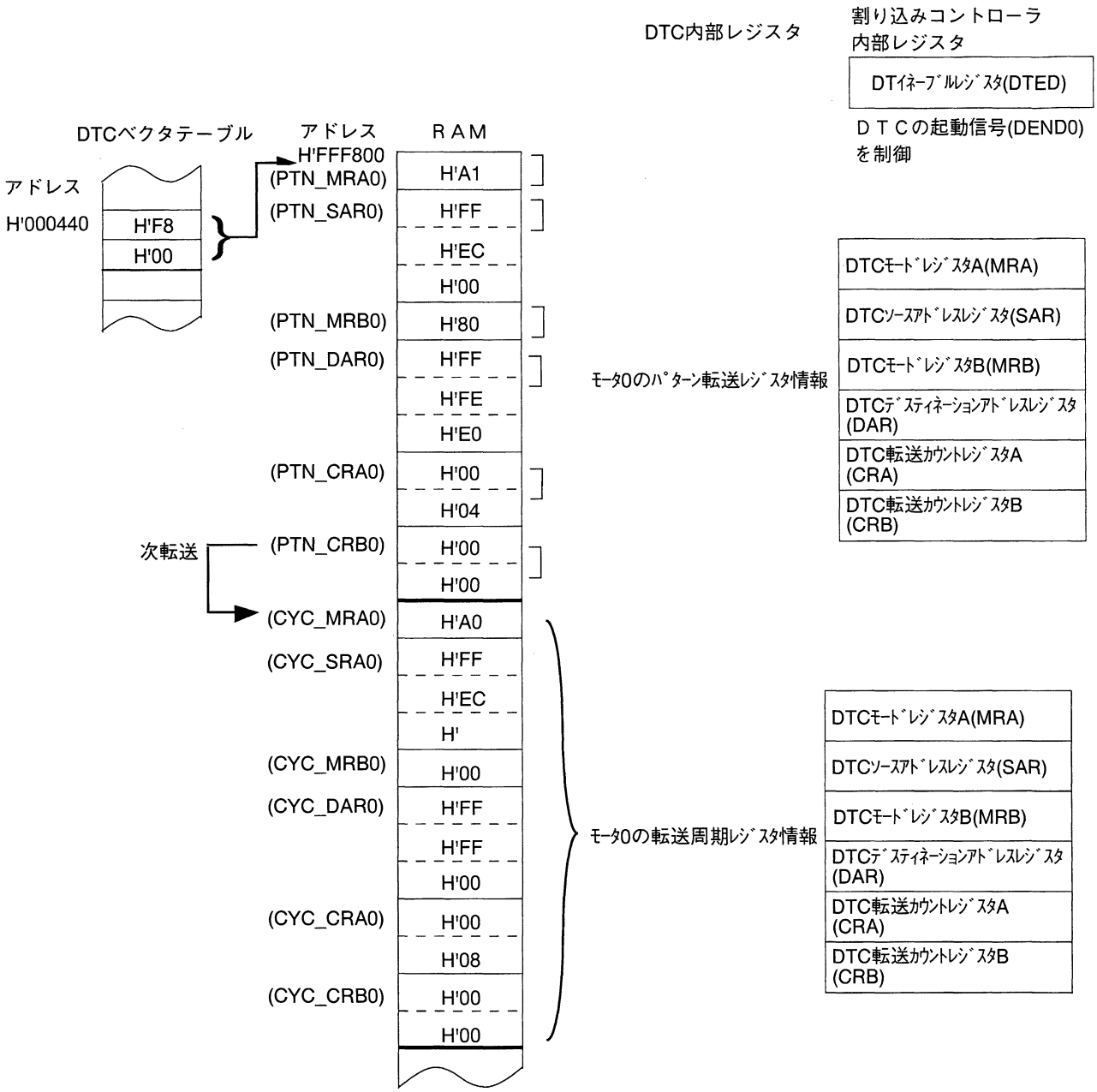


図6　DTCベクタテーブルとメモリ上の配置例

（2）表1に本タスク例の機能割付を示します。表1に示すようにH8S／2655の機能を割付、2相励磁のステッピングモータ出力波形を生成します。

表1　H8S／2655機能割り付け

H8S/2655機能		機能
TPU	TCNT	16ビットカウンタ
	TGRA	アウトプットコンペアレジスタA
	TGRB	アウトプットコンペアレジスタB
	TCR	カウントクロックの選択及びカウンタクリア要因の選定
	TIOR	TGRA及びTGRBをアウトプットコンペアレジスタに設定
	TSR	コンペアマッチやオーバフローのステータスレジスタ
DTC	MRA	DTCの動作モードを制御
	MRB	DTCチェイン転送を指定
	DTCER	各割り込み要因によるDTC起動の許可／禁止
	DTVECR	ソフトウェア起動割り込み用ベクタ番号を設定
	SAR	転送元アドレスの設定
	DAR	転送先アドレスの設定
	CRA	転送回数を設定
PPG	PODR	PPG出力の出力データを格納
	NDR	PPG出力の次に出力するデータを格納
	PO15～PO0	4相のステッピングモータ出力波形を発生

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

動作説明

（1） 4相パルス出力例

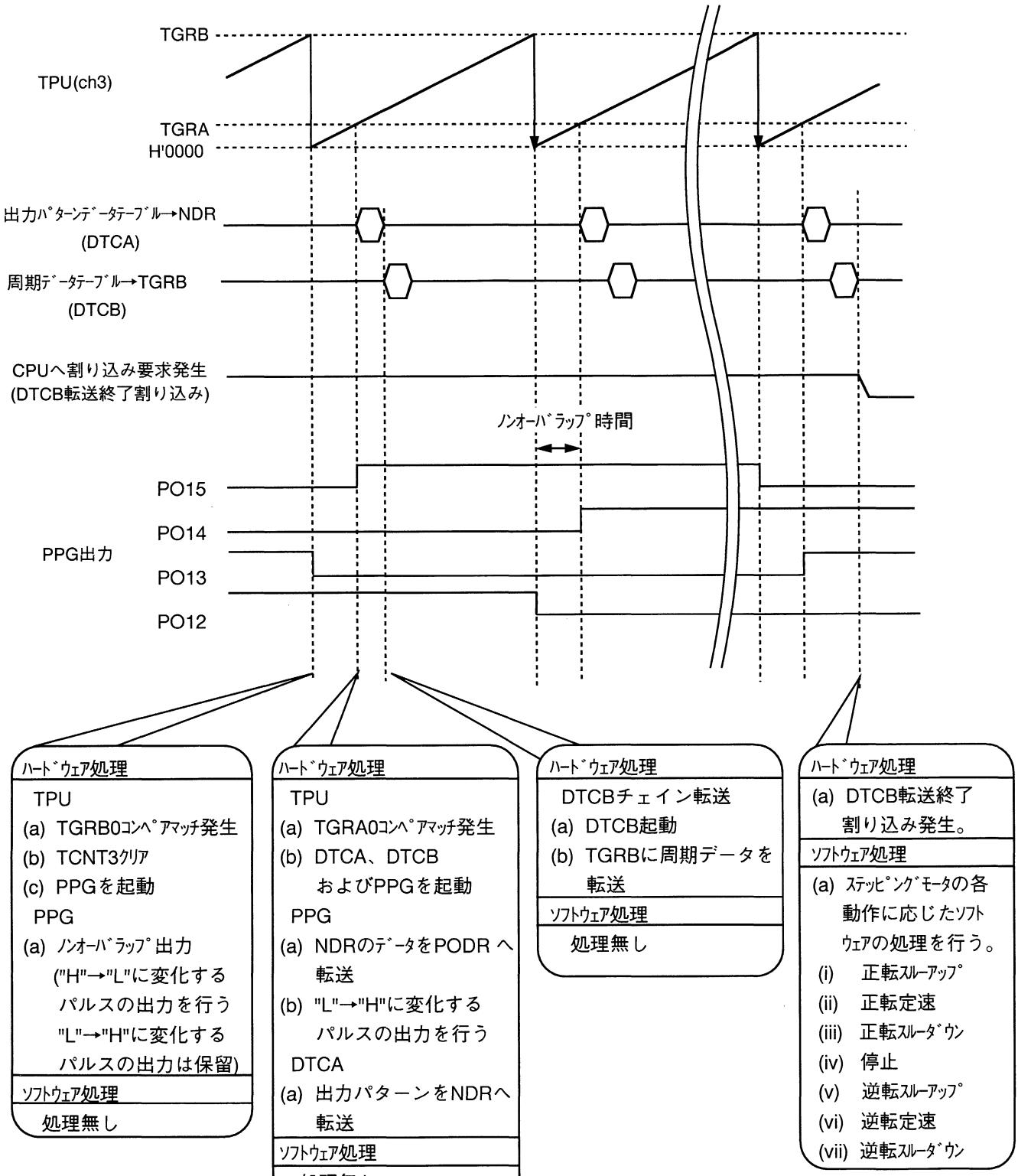


図7　ステッピングモータタイミング説明

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

ソフトウェア説明

（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	stp4mn	TPU、PPGおよびDTCの初期設定を行い、正転のスルーアップ動作の設定を行う。
モータ制御	mtrctl0 mtrctl1 mtrctl2 mtrctl3	モータの各動作の制御を行う。
正転ｽﾙｰｱｯﾌﾟ	fslueup0~3	逆転ｽﾄｯﾌﾟ動作終了後起動し、正転ｽﾙｰｱｯﾌﾟ動作のDTC転送ﾓｰﾄﾞ切替えを行う
正転定速	fconst0~3	正転ｽﾙｰｱｯﾌﾟ動作終了後起動し、正転定速動作のDTC転送ﾓｰﾄﾞ切替えを行う
正転ｽﾙｰﾀﾞｳﾝ	fsldwn0~3	正転定速動作終了後起動し、正転ｽﾙｰﾀﾞｳﾝ動作のDTC転送ﾓｰﾄﾞ切替えを行う
正転停止	fstop0~3	正転ｽﾙｰﾀﾞｳﾝ動作終了後起動し、正転停止動作のDTC転送ﾓｰﾄﾞ切替えを行う
逆転ｽﾙｰｱｯﾌﾟ	rslueup0~3	正転停止動作終了後起動し、逆転ｽﾙｰｱｯﾌﾟ動作のDTC転送ﾓｰﾄﾞ切替えを行う
逆転定速	rconst0~3	逆転ｽﾙｰｱｯﾌﾟ動作終了後起動し、逆転定速動作のDTC転送ﾓｰﾄﾞ切替えを行う
逆転ｽﾙｰﾀﾞｳﾝ	rsludwn0~3	逆転定速動作終了後起動し、逆転ｽﾙｰﾀﾞｳﾝ動作のDTC転送ﾓｰﾄﾞ切替えを行う
逆転停止	rstop0~3	逆転ｽﾙｰﾀﾞｳﾝ動作終了後起動し、逆転停止動作のDTC転送ﾓｰﾄﾞ切替えを行う

（2）引数の説明

本タスクでは引数を使用していません。

（3）使用内部レジスタの説明

内蔵機能	レジスタ名	機能
TPU	TGRA	ノンオーバラップ時間を設定
	TGRB	タイマ周期を設定
	TIER	TGFA割り込みを許可
	TCR	TPUを以下のように設定 ・TGRBのコンペアマッチでカウンタクリア ・内部クロックφでカウント
	TIOR	TGRAおよびTGRBをアウトプットコンペアレジスタに設定し、端子出力を禁止
	TSTR	TCNTのカウント動作を許可
DTC	DTCER	TGIA割込みによるDTC起動の許可
PPG	PODR	出力パターンデータを格納
	PMR	PO15～PO0をノンオーバラップ出力に設定
	PCR	パルス出力の出力トリガ信号をグループ単位で選択 グループ3：TPU3のコンペアマッチ グループ2：TPU2のコンペアマッチ グループ1：TPU1のコンペアマッチ グループ0：TPU0のコンペアマッチ
	NDERL	PPG出力PO7～PO0を許可
	NDERH	PPG出力PO15～PO8を許可
	NDRL	PO7～PO0の次に出力するパターンデータを格納
	NDRH	PO15～PO8の次に出力するパターンデータを格納
MSTPCR		TPU、DTC、PPGのモジュールストップモードを解除

（4）使用RAM説明

ラベル名	機能	データ長	使用ﾓｼﾞｭｰﾙ名
nextmode0 nextmode1 nextmode2 nextmode3	ステッピングモータの動作状態を示す H'01：正転ｽﾙｰｱｯﾌﾟ制御　H'05：逆転ｽﾙｰｱｯﾌﾟ制御 H'02：正転定速制御　H'06：逆転定速制御 H'03：正転ｽﾙｰﾀﾞｳﾝ制御　H'07：逆転ｽﾙｰﾀﾞｳﾝ制御 H'04：正転停止制御　H'08：逆転停止制御	unsigned char	ﾓｰﾀ制御
PTN_MRA0	転送モードをリピートモードに設定	unsigned char	メインルーチン
PTN_MRB0	チェイン転送を許可	unsigned char	
PTN_SAR0	転送元アドレス（PATTBL）を設定	unsigned long	
PTN_DAR0	転送先アドレス（NDR0）を設定	unsigned long	
PTN_CRA0	ブロックサイズを設定	unsigned short	
CYC_MRA0	転送モードをノーマルモードに設定	unsigned char	
CYC_MRB0	転送終了後CPUへの割込み要求を許可	unsigned char	
CYC_SAR0	転送元アドレス（UPTBL）を設定	unsigned long	
CYC_DAR0	転送先アドレス（TGRB0）を設定	unsigned long	
CYC_CRA0	転送回数を設定	unsigned short	
PTN_MRA1	転送モードをリピートモードに設定	unsigned char	
PTN_MRB1	チェイン転送を許可	unsigned char	
PTN_SAR1	転送元アドレス（PATTBL）を設定	unsigned long	
PTN_DAR1	転送先アドレス（NDR1）を設定	unsigned long	
PTN_CRA1	ブロックサイズを設定	unsigned short	
CYC_MRA1	転送モードをノーマルモードに設定	unsigned char	
CYC_MRB1	転送終了後CPUへの割込み要求を許可	unsigned char	
CYC_SAR1	転送元アドレス（UPTBL）を設定	unsigned long	
CYC_DAR1	転送先アドレス（TGRB1）を設定	unsigned long	
CYC_CRA1	転送回数を設定	unsigned short	
PTN_MRA2	転送モードをリピートモードに設定	unsigned char	
PTN_MRB2	チェイン転送を許可	unsigned char	
PTN_SAR2	転送元アドレス（PATTBL）を設定	unsigned long	
PTN_DAR2	転送先アドレス（NDR2）を設定	unsigned long	
PTN_CRA2	ブロックサイズを設定	unsigned short	
CYC_MRA2	転送モードをノーマルモードに設定	unsigned char	
CYC_MRB2	転送終了後CPUへの割込み要求を許可	unsigned char	
CYC_SAR2	転送元アドレス（UPTBL）を設定	unsigned long	
CYC_DAR2	転送先アドレス（TGRB2）を設定	unsigned long	
CYC_CRA2	転送回数を設定	unsigned short	
PTN_MRA3	転送モードをリピートモードに設定	unsigned char	
PTN_MRB3	チェイン転送を許可	unsigned char	
PTN_SAR3	転送元アドレス（PATTBL）を設定	unsigned long	
PTN_DAR3	転送先アドレス（NDR3）を設定	unsigned long	
PTN_CRA3	ブロックサイズを設定	unsigned short	
CYC_MRA3	転送モードをノーマルモードに設定	unsigned char	
CYC_MRB3	転送終了後CPUへの割込み要求を許可	unsigned char	
CYC_SAR3	転送元アドレス（UPTBL）を設定	unsigned long	
CYC_DAR3	転送先アドレス（TGRB3）を設定	unsigned long	
CYC_CRA3	転送回数を設定	unsigned short	

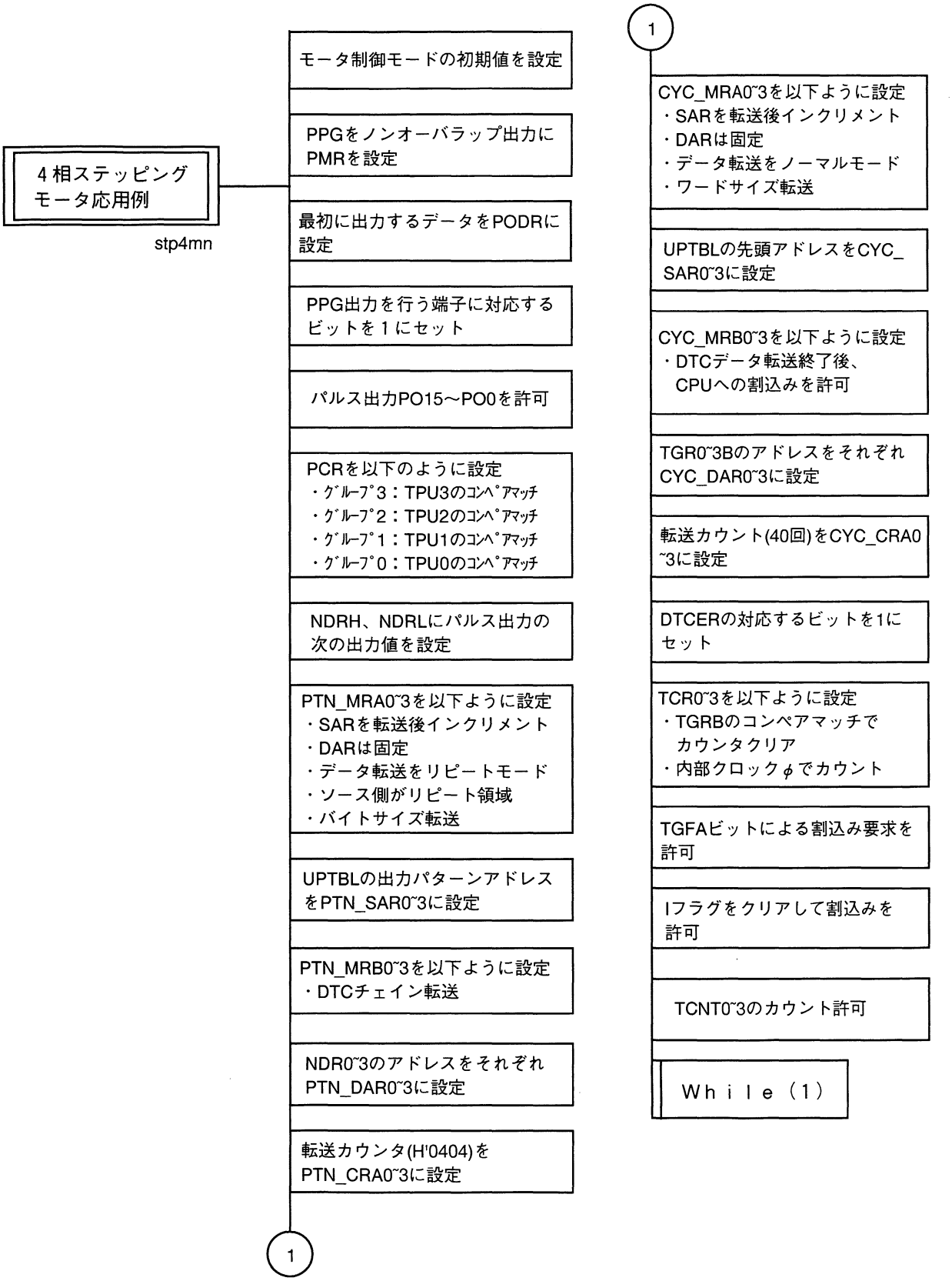
（5）データテーブル説明

テーブル名	機能	データ長	データ容量
PATTBL0, PATTBL1	4相パルスを出力するためのパターンを設定します。	unsigned char	4バイト
UPTBL DOWNTBL CNSTBL	ステップ周期を変更するためのデータを設定します。	unsigned short	121ワード

4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

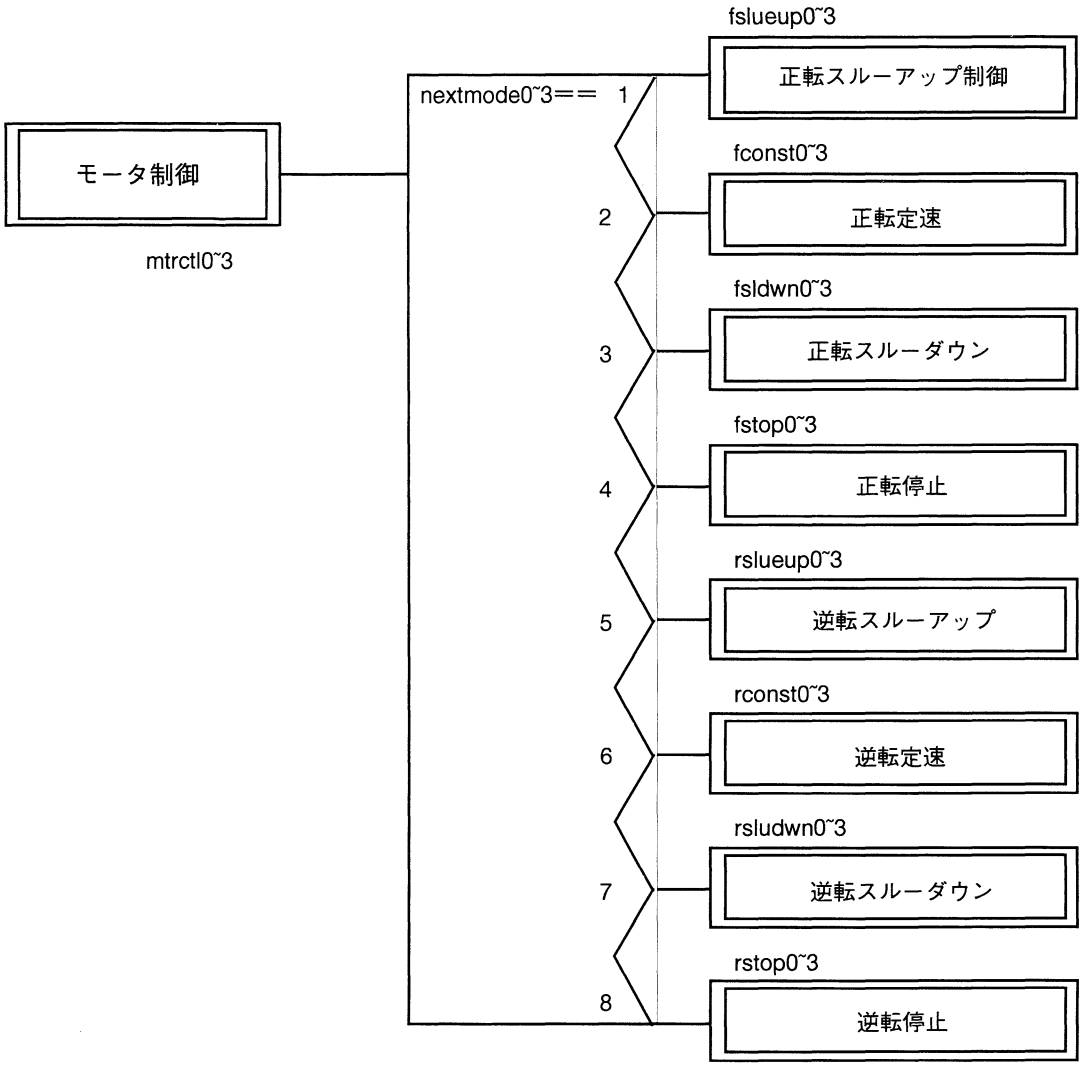
（1）メインルーチン



4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

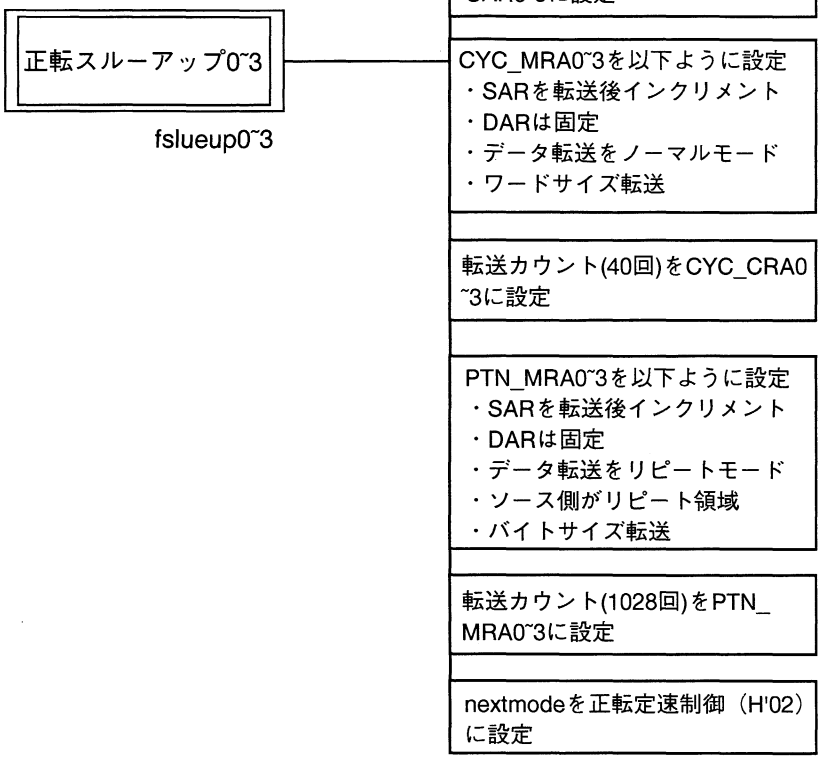
（2）モータ制御



4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

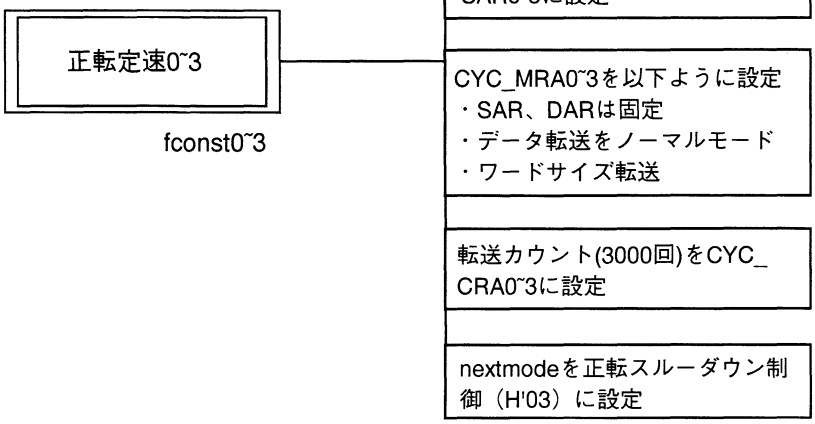
（3）正転スルーアップ0～3



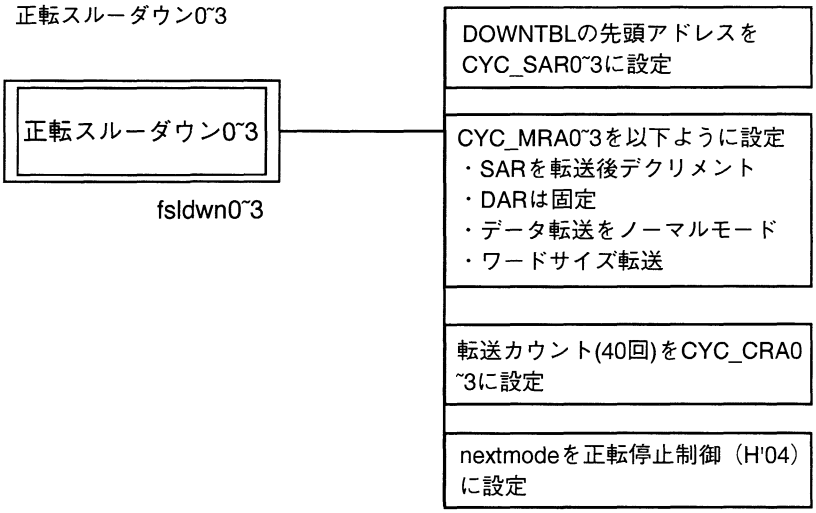
4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

（4）正転定速0~3



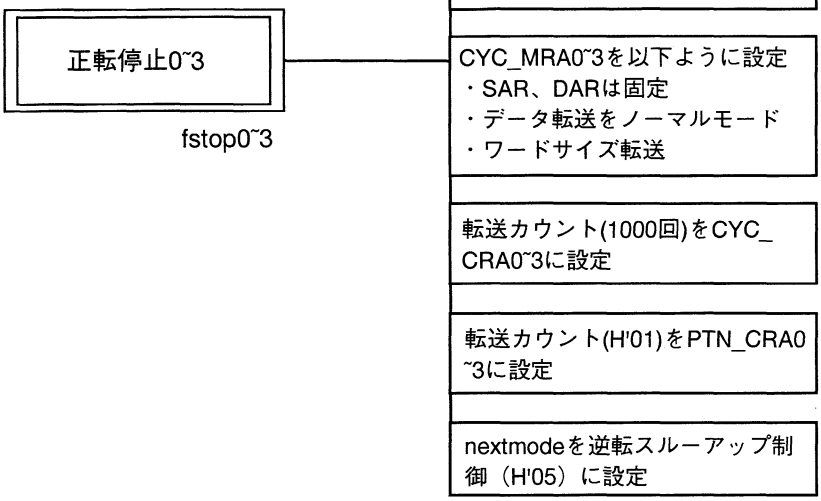
（5）正転スルーダウン0~3



4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

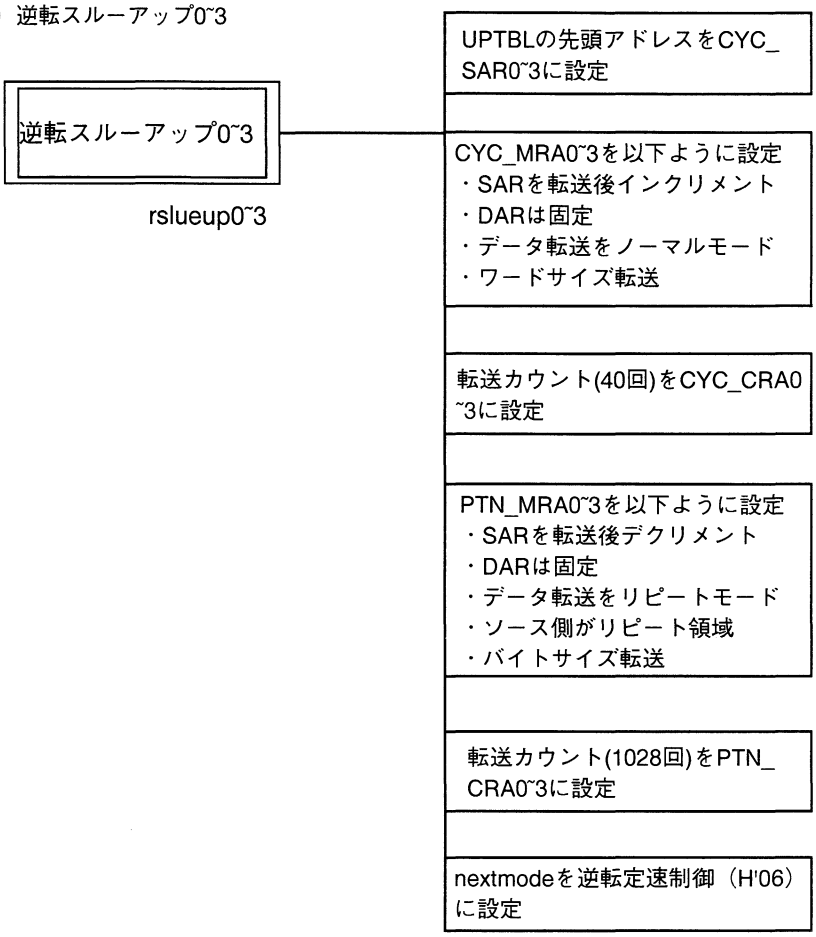
（6）正転停止0~3



4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

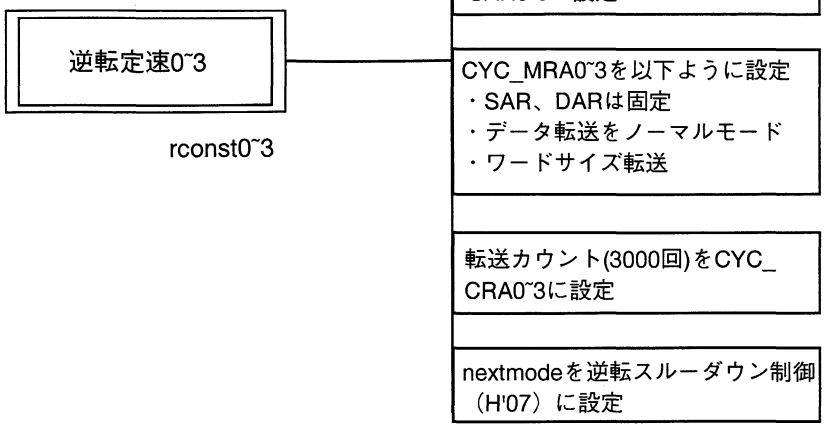
（7）逆転スルーアップ0~3



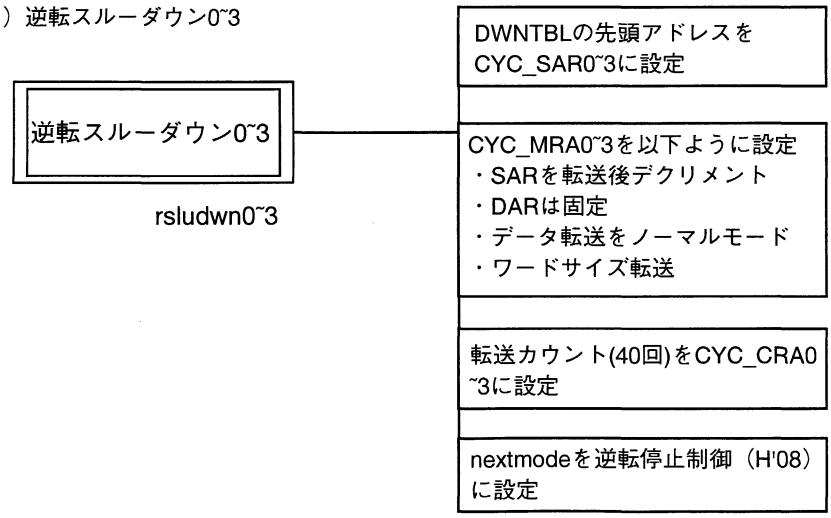
4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

（8）逆転定速0~3



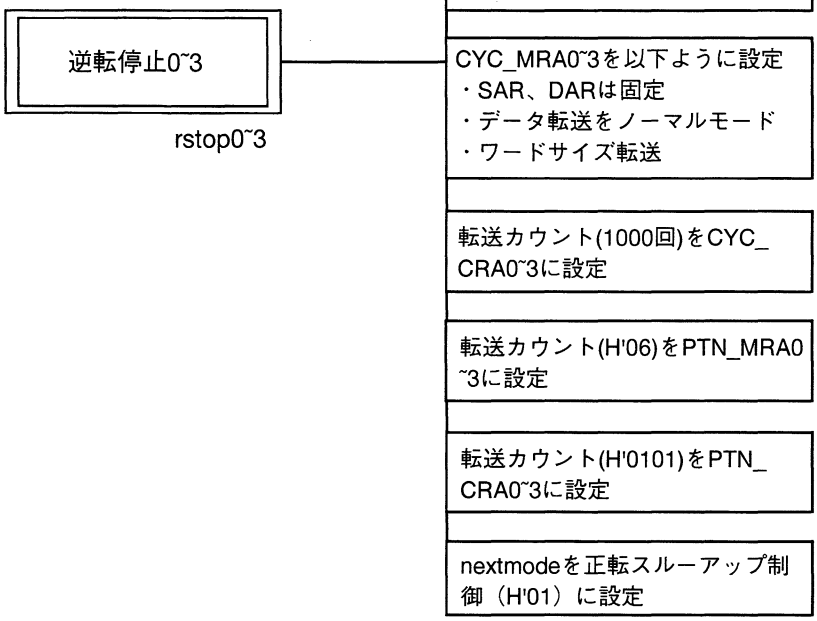
（9）逆転スルーダウン0~3



4相ステッピングモータ応用例	MCU	H8S／2655	使用機能	TPU、PPG、DTC

PAD

（10）逆転停止0~3



```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void stp4mn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/

#define PTN0_MRA (*(volatile unsigned char *)0xfff800)
#define PTN0_SAR (*(volatile unsigned long *)0xfff800)
#define PTN0_MRB (*(volatile unsigned char *)0xfff804)
#define PTN0_DAR (*(volatile unsigned long *)0xfff804)
#define PTN0_CRA (*(volatile unsigned short *)0xfff808)
#define PTN0_CRB (*(volatile unsigned short *)0xfff80a)

#define CYC0_MRA (*(volatile unsigned char *)0xfff80c)
#define CYC0_SAR (*(volatile unsigned long *)0xfff80c)
#define CYC0_MRB (*(volatile unsigned char *)0xfff810)
#define CYC0_DAR (*(volatile unsigned long *)0xfff810)
#define CYC0_CRA (*(volatile unsigned short *)0xfff814)
#define CYC0_CRB (*(volatile unsigned short *)0xfff816)

#define PTN1_MRA (*(volatile unsigned char *)0xfff818)
#define PTN1_SAR (*(volatile unsigned long *)0xfff818)
#define PTN1_MRB (*(volatile unsigned char *)0xfff81c)
#define PTN1_DAR (*(volatile unsigned long *)0xfff81c)
#define PTN1_CRA (*(volatile unsigned short *)0xfff820)
#define PTN1_CRB (*(volatile unsigned short *)0xfff822)

#define CYC1_MRA (*(volatile unsigned char *)0xfff824)
#define CYC1_SAR (*(volatile unsigned long *)0xfff824)
#define CYC1_MRB (*(volatile unsigned char *)0xfff828)
#define CYC1_DAR (*(volatile unsigned long *)0xfff828)
```

```
#define CYC1_CRA (*(volatile unsigned short *)0xfff82c)
#define CYC1_CRB (*(volatile unsigned short *)0xfff82e)

#define PTN2_MRA (*(volatile unsigned char *)0xfff830)
#define PTN2_SAR (*(volatile unsigned long *)0xfff830)
#define PTN2_MRB (*(volatile unsigned char *)0xfff834)
#define PTN2_DAR (*(volatile unsigned long *)0xfff834)
#define PTN2_CRA (*(volatile unsigned short *)0xfff838)
#define PTN2_CRB (*(volatile unsigned short *)0xfff83a)

#define CYC2_MRA (*(volatile unsigned char *)0xfff83c)
#define CYC2_SAR (*(volatile unsigned long *)0xfff83c)
#define CYC2_MRB (*(volatile unsigned char *)0xfff840)
#define CYC2_DAR (*(volatile unsigned long *)0xfff840)
#define CYC2_CRA (*(volatile unsigned short *)0xfff844)
#define CYC2_CRB (*(volatile unsigned short *)0xfff846)

#define PTN3_MRA (*(volatile unsigned char *)0xfff848)
#define PTN3_SAR (*(volatile unsigned long *)0xfff848)
#define PTN3_MRB (*(volatile unsigned char *)0xfff84c)
#define PTN3_DAR (*(volatile unsigned long *)0xfff84c)
#define PTN3_CRA (*(volatile unsigned short *)0xfff850)
#define PTN3_CRB (*(volatile unsigned short *)0xfff852)

#define CYC3_MRA (*(volatile unsigned char *)0xfff854)
#define CYC3_SAR (*(volatile unsigned long *)0xfff854)
#define CYC3_MRB (*(volatile unsigned char *)0xfff858)
#define CYC3_DAR (*(volatile unsigned long *)0xfff858)
#define CYC3_CRA (*(volatile unsigned short *)0xfff85c)
#define CYC3_CRB (*(volatile unsigned short *)0xfff85e)

#define nextmode0 (*(volatile unsigned char *)0xfff860)
#define nextmode1 (*(volatile unsigned char *)0xfff861)
#define nextmode2 (*(volatile unsigned char *)0xfff862)
#define nextmode3 (*(volatile unsigned char *)0xfff863)
```

プログラムリスト

```
/*****************************************/
/*         DATA TABLE                    */
/*****************************************/
const unsigned char PATTBL0[4] = {0x6f,0x3f,0x9f,0xcf};
const unsigned char PATTBL1[4] = {0xf6,0xf3,0xf9,0xfc};

/*
const unsigned short UPTBL[40] = {  47723, 43885, 40848, 38365, 36286,
                                    34513, 32977, 31629, 30434, 29365,
                                    28402, 27527, 26729, 25996, 25321,
                                    24695, 24114, 23572, 23065, 22589,
                                    22141, 21720, 21321, 20944, 20585,
                                    20245, 19921, 19612, 19317, 19035,
                                    18765, 18506, 18258, 18019, 17790,
                                    17569, 17356, 17150, 16952, 16760     };
const unsigned short DOWNTBL[] = { 16760 };
const unsigned short CNSTBL[] = { 16760 };

*/

const unsigned short UPTBL[12] = {  35350, 14637, 11225, 9462,
                                    8337, 7537, 6937, 6450, 6062,
                                    5725, 5450, 5212     };

const unsigned short DOWNTBL[] = { 5000 };
const unsigned short CNSTBL[] = { 5000 };


/*****************************************/
/*        MAIN PROGRAM : stp4mn          */
/*****************************************/
void stp4mn(void)
{
      MSTPCR = 0x97ff;
      nextmode0 = 0x02;
      nextmode1 = 0x02;
      nextmode2 = 0x02;
      nextmode3 = 0x02;
```

プログラムリスト

```
/* PPG INITIALIZE */
      PMR = 0xff;
      PODRH = 0xcc;
      PODRL = 0xcc;
      P1DDR = 0xff;
      P2DDR = 0xff;
      NDERH = 0xff;
      NDERL = 0xff;
      PCR = 0xe4;
      NDR3 = 0xcf;
      NDR2 = 0xfc;
      NDR1 = 0xcf;
      NDR0 = 0xfc;

/* DTC INITIALIZE */
      PTN0_SAR = (long)PATTBL1;
      PTN1_SAR = (long)PATTBL0;
      PTN2_SAR = (long)PATTBL1;
      PTN3_SAR = (long)PATTBL0;
      PTN0_MRA = 0x86;
      PTN1_MRA = 0x86;
      PTN2_MRA = 0x86;
      PTN3_MRA = 0x86;
      PTN0_DAR = (long)(&NDR0);
      PTN1_DAR = (long)(&NDR1);
      PTN2_DAR = (long)(&NDR2);
      PTN3_DAR = (long)(&NDR3);
      PTN0_MRB = 0x80;
      PTN1_MRB = 0x80;
      PTN2_MRB = 0x80;
      PTN3_MRB = 0x80;
      PTN0_CRA = 0x0404;
      PTN1_CRA = 0x0404;
      PTN2_CRA = 0x0404;
      PTN3_CRA = 0x0404;
```

プログラムリスト

```
        CYC0_SAR = (long)UPTBL;
        CYC1_SAR = (long)UPTBL;
        CYC2_SAR = (long)UPTBL;
        CYC3_SAR = (long)UPTBL;
        CYC0_MRA = 0x81;
        CYC1_MRA = 0x81;
        CYC2_MRA = 0x81;
        CYC3_MRA = 0x81;
        CYC0_DAR = (long)(&TGR0B);
        CYC1_DAR = (long)(&TGR1B);
        CYC2_DAR = (long)(&TGR2B);
        CYC3_DAR = (long)(&TGR3B);
        CYC0_MRB = 0x00;
        CYC1_MRB = 0x00;
        CYC2_MRB = 0x00;
        CYC3_MRB = 0x00;
        CYC0_CRA = 13;
        CYC1_CRA = 13;
        CYC2_CRA = 13;
        CYC3_CRA = 13;
        DTCERB_BP.TGI0A = 1;
        DTCERB_BP.TGI1A = 1;
        DTCERC_BP.TGI2A = 1;
        DTCERC_BP.TGI3A = 1;

    /* TPU INITIALIZE */
        TPU_TCR0 = 0x42;
        TPU_TCR1 = 0x42;
        TCR2 = 0x42;
        TCR3 = 0x42;
        TIER0 = 0x41;
        TIER1 = 0x41;
        TIER2 = 0x41;
        TIER3 = 0x41;
        TGR0A = 100;
        TGR1A = 100;
        TGR2A = 100;
        TGR3A = 100;
        TGR0B = 62500;
        TGR1B = 62500;
        TGR2B = 62500;
        TGR3B = 62500;

        set_imask_ccr(0);
        TSTR = 0x0f;
        while(1);
    }
```

プログラムリスト

```
/*****************************************/
/*        NAME : mtrcnt10                */
/*****************************************/
#pragma interrupt(mtrcnt10)
void mtrcnt10(void)
{
        switch(nextmode0){
            case 1:
                  fslueup0();
                  break;
            case 2:
                  fconst0();
                  break;
            case 3:
                  fsldwn0();
                  break;
            case 4:
                  fstop0();
                  break;
            case 5:
                  rslueup0();
                  break;
            case 6:
                  rconst0();
                  break;
            case 7:
                  rsldwn0();
                  break;
            case 8:
                  rstop0();
                  break;
            default:
                  break;
            }

      DTCERB_BP.TGI0A = 1;
      TSR0 &= 0xfe;
}
```

プログラムリスト

```
/*********** forward slue-up0 *************/
fslueup0()
{
      CYC0_SAR = (long)UPTBL;
      CYC0_MRA = 0x81;
      CYC0_CRA = 0x000d;
      PTN0_SAR = (long)PATTBL1;
      PTN0_MRA = 0x86;
      PTN0_CRA = 0x0404;
      nextmode0++;
}

/******** forward constant speed0 *********/
fconst0()
{
      CYC0_SAR = (long)CNSTBL;
      CYC0_MRA = 0x01;
      CYC0_CRA = 0x0bb9;
      PTN0_SAR = (long)PATTBL1;
      PTN0_MRA = 0x86;
      PTN0_CRA = 0x0404;
      nextmode0++;
}

/********** forward slue-down0 ************/
fsldwn0()
{
      CYC0_SAR = (long)DOWNTBL;
      CYC0_MRA = 0xc1;
      CYC0_CRA = 0x000d;
      PTN0_SAR = (long)PATTBL1;
      PTN0_MRA = 0x86;
      PTN0_CRA = 0x0404;
      nextmode0++;
}
/************* forward stop0 **************/
fstop0()
{
      CYC0_SAR = (long)UPTBL;
      CYC0_MRA = 0x01;
      CYC0_CRA = 0x03e9;
      PTN0_SAR = (long)PATTBL1;
      PTN0_MRA = 0x06;
      PTN0_CRA = 0x0404;
      nextmode0++;
}
```

プログラムリスト

```
/*********** reverse slue-up0 *************/
rslueup0()
{
        CYC0_SAR = (long)UPTBL;
        CYC0_MRA = 0x81;
        CYC0_CRA = 0x000d;
        PTN0_SAR = (long)(PATTBL1+3);
        PTN0_MRA = 0xc6;
        PTN0_CRA = 0x0404;
        nextmode0++;
}

/******** reverse constant speed0 *********/
rconst0()
{
        CYC0_SAR = (long)CNSTBL;
        CYC0_MRA = 0x01;
        CYC0_CRA = 0x0bb9;
        PTN0_SAR = (long)(PATTBL1+3);
        PTN0_MRA = 0xc6;
        PTN0_CRA = 0x0404;
        nextmode0++;
}

/********** reverse slue-down0 ************/
rsldwn0()
{
        CYC0_SAR = (long)DOWNTBL;
        CYC0_MRA = 0xc1;
        CYC0_CRA = 0x000d;
        PTN0_SAR = (long)(PATTBL1+3);
        PTN0_MRA = 0xc6;
        PTN0_CRA = 0x0404;
        nextmode0++;
}

/************* reverse stop0 **************/
rstop0()
{
        CYC0_SAR = (long)UPTBL;
        CYC0_MRA = 0x01;
        CYC0_CRA = 0x03e9;
        PTN0_SAR = (long)(PATTBL1+3);
        PTN0_MRA = 0x06;
        PTN0_CRA = 0x0404;
        nextmode0 = 0x01;
}
```

プログラムリスト

```
/*****************************************/
/*        NAME : mtrcntl1                */
/*****************************************/
#pragma interrupt(mtrcntl1)
void mtrcntl1(void)
{
        switch(nextmode1){
            case 1:
                  fslueup1();
                  break;
            case 2:
                  fconst1();
                  break;
            case 3:
                  fsldwn1();
                  break;
            case 4:
                  fstop1();
                  break;
            case 5:
                  rslueup1();
                  break;
            case 6:
                  rconst1();
                  break;
            case 7:
                  rsldwn1();
                  break;
            case 8:
                  rstop1();
                  break;
            default:
                  break;
            }

      DTCERB_BP.TGI1A = 1;
      TSR1 &= 0xfe;
}
```

プログラムリスト

```
/********** forward slue-up1 ************/
fslueup1()
{
      CYC1_SAR = (long)UPTBL;
      CYC1_MRA = 0x81;
      CYC1_CRA = 0x000d;
      PTN1_SAR = (long)PATTBL0;
      PTN1_MRA = 0x86;
      PTN1_CRA = 0x0404;
      nextmode1++;
}

/******** forward constant speed1 *********/
fconst1()
{
      CYC1_SAR = (long)CNSTBL;
      CYC1_MRA = 0x01;
      CYC1_CRA = 0x0bb9;
      PTN1_SAR = (long)PATTBL0;
      PTN1_MRA = 0x86;
      PTN1_CRA = 0x0404;
      nextmode1++;
}

/********** forward slue-down1 ************/
fsldwn1()
{
      CYC1_SAR = (long)DOWNTBL;
      CYC1_MRA = 0xc1;
      CYC1_CRA = 0x000d;
      PTN1_SAR = (long)PATTBL0;
      PTN1_MRA = 0x86;
      PTN1_CRA = 0x0404;
      nextmode1++;
}

/************* forward stop1 **************/
fstop1()
{
      CYC1_SAR = (long)UPTBL;
      CYC1_MRA = 0x01;
      CYC1_CRA = 0x03e9;
      PTN1_SAR = (long)PATTBL0;
      PTN1_MRA = 0x06;
      PTN1_CRA = 0x0404;
      nextmode1++;
}
```

プログラムリスト

```
/********** reverse slue-up1 ************/
rslueup1()
{
        CYC1_SAR = (long)UPTBL;
        CYC1_MRA = 0x81;
        CYC1_CRA = 0x000d;
        PTN1_SAR = (long)(PATTBL0+3);
        PTN1_MRA = 0xc6;
        PTN1_CRA = 0x0404;
        nextmode1++;
}

/******** reverse constant speed1 *********/
rconst1()
{
        CYC1_SAR = (long)CNSTBL;
        CYC1_MRA = 0x01;
        CYC1_CRA = 0x0bb9;
        PTN1_SAR = (long)(PATTBL0+3);
        PTN1_MRA = 0xc6;
        PTN1_CRA = 0x0404;
        nextmode1++;
}
/********** reverse slue-down1 ************/
rsldwn1()
{
        CYC1_SAR = (long)DOWNTBL;
        CYC1_MRA = 0xc1;
        CYC1_CRA = 0x000d;
        PTN1_SAR = (long)(PATTBL0+3);
        PTN1_MRA = 0xc6;
        PTN1_CRA = 0x0404;
        nextmode1++;
}

/************* reverse stop1 **************/
rstop1()
{
        CYC1_SAR = (long)UPTBL;
        CYC1_MRA = 0x01;
        CYC1_CRA = 0x03e9;
        PTN1_SAR = (long)(PATTBL0+3);
        PTN1_MRA = 0x06;
        PTN1_CRA = 0x0404;
        nextmode1 = 0x01;
}
```

プログラムリスト

```
/*****************************************/
/*        NAME : mtrcntl2                */
/*****************************************/
#pragma interrupt(mtrcntl2)
void mtrcntl2(void)
{
        switch(nextmode2){
            case 1:
                  fslueup2();
                  break;
            case 2:
                  fconst2();
                  break;
            case 3:
                  fsldwn2();
                  break;
            case 4:
                  fstop2();
                  break;
            case 5:
                  rslueup2();
                  break;
            case 6:
                  rconst2();
                  break;
            case 7:
                  rsldwn2();
                  break;
            case 8:
                  rstop2();
                  break;
            default:
                  break;
            }

      DTCERC_BP.TGI2A = 1;
      TSR2 &= 0xfe;
}
```

プログラムリスト

```
/*********** forward slue-up2 *************/
fslueup2()
{
            CYC2_SAR = (long)UPTBL;
            CYC2_MRA = 0x81;
            CYC2_CRA = 0x000d;
            PTN2_SAR = (long)PATTBL1;
            PTN2_MRA = 0x86;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/******** forward constant speed2 *********/
fconst2()
{
            CYC2_SAR = (long)CNSTBL;
            CYC2_MRA = 0x01;
            CYC2_CRA = 0x0bb9;
            PTN2_SAR = (long)PATTBL1;
            PTN2_MRA = 0x86;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/********** forward slue-down2 ************/
fsldwn2()
{
            CYC2_SAR = (long)DOWNTBL;
            CYC2_MRA = 0xc1;
            CYC2_CRA = 0x000d;
            PTN2_SAR = (long)PATTBL1;
            PTN2_MRA = 0x86;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/************* forward stop2 **************/
fstop2()
{
            CYC2_SAR = (long)UPTBL;
            CYC2_MRA = 0x01;
            CYC2_CRA = 0x03e9;
            PTN2_SAR = (long)PATTBL1;
            PTN2_MRA = 0x06;
            PTN2_CRA = 0x0404;
            nextmode2++;
}
```

プログラムリスト

```
/*********** reverse slue-up2 *************/
rslueup2()
{
            CYC2_SAR = (long)UPTBL;
            CYC2_MRA = 0x81;
            CYC2_CRA = 0x000d;
            PTN2_SAR = (long)(PATTBL1+3);
            PTN2_MRA = 0xc6;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/******** reverse constant speed2 *********/
rconst2()
{
            CYC2_SAR = (long)CNSTBL;
            CYC2_MRA = 0x01;
            CYC2_CRA = 0x0bb9;
            PTN2_SAR = (long)(PATTBL1+3);
            PTN2_MRA = 0xc6;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/********** reverse slue-down2 ************/
rsldwn2()
{
            CYC2_SAR = (long)DOWNTBL;
            CYC2_MRA = 0xc1;
            CYC2_CRA = 0x000d;
            PTN2_SAR = (long)(PATTBL1+3);
            PTN2_MRA = 0xc6;
            PTN2_CRA = 0x0404;
            nextmode2++;
}

/************* reverse stop2 **************/
rstop2()
{
            CYC2_SAR = (long)UPTBL;
            CYC2_MRA = 0x01;
            CYC2_CRA = 0x03e9;
            PTN2_SAR = (long)(PATTBL1+3);
            PTN2_MRA = 0x06;
            PTN2_CRA = 0x0404;
            nextmode2 = 0x01;
}
```

プログラムリスト

```
/*****************************************/
/*        NAME : mtrcntl3                */
/*****************************************/
#pragma interrupt(mtrcntl3)
void mtrcntl3(void)
{
        switch(nextmode3){
                        case 1:
                              fslueup3();
                              break;
                        case 2:
                              fconst3();
                              break;
                        case 3:
                              fsldwn3();
                              break;
                        case 4:
                              fstop3();
                              break;
                        case 5:
                              rslueup3();
                              break;
                        case 6:
                              rconst3();
                              break;
                        case 7:
                              rsldwn3();
                              break;
                        case 8:
                              rstop3();
                              break;
                        default:
                              break;
                }

                DTCERC_BP.TGI3A = 1;
                TSR3 &= 0xfe;
}
```

プログラムリスト

```
/*********** forward slue-up3 *************/
fslueup3()
{
            CYC3_SAR = (long)UPTBL;
            CYC3_MRA = 0x81;
            CYC3_CRA = 0x000d;
            PTN3_SAR = (long)PATTBL0;
            PTN3_MRA = 0x86;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/******** forward constant speed3 *********/
fconst3()
{
            CYC3_SAR = (long)CNSTBL;
            CYC3_MRA = 0x01;
            CYC3_CRA = 0x0bb9;
            PTN3_SAR = (long)PATTBL0;
            PTN3_MRA = 0x86;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/********** forward slue-down3 ************/
fsldwn3()
{
            CYC3_SAR = (long)DOWNTBL;
            CYC3_MRA = 0xc1;
            CYC3_CRA = 0x000d;
            PTN3_SAR = (long)PATTBL0;
            PTN3_MRA = 0x86;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/************* forward stop3 **************/
fstop3()
{
            CYC3_SAR = (long)UPTBL;
            CYC3_MRA = 0x01;
            CYC3_CRA = 0x03e9;
            PTN3_SAR = (long)PATTBL0;
            PTN3_MRA = 0x06;
            PTN3_CRA = 0x0404;
            nextmode3++;
}
```

プログラムリスト

```
/*********** reverse slue-up3 *************/
rslueup3()
{
            CYC3_SAR = (long)UPTBL;
            CYC3_MRA = 0x81;
            CYC3_CRA = 0x000d;
            PTN3_SAR = (long)(PATTBL0+3);
            PTN3_MRA = 0xc6;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/******** reverse constant speed3 *********/
rconst3()
{
            CYC3_SAR = (long)CNSTBL;
            CYC3_MRA = 0x01;
            CYC3_CRA = 0x0bb9;
            PTN3_SAR = (long)(PATTBL0+3);
            PTN3_MRA = 0xc6;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/********** reverse slue-down3 ************/
rsldwn3()
{
            CYC3_SAR = (long)DOWNTBL;
            CYC3_MRA = 0xc1;
            CYC3_CRA = 0x000d;
            PTN3_SAR = (long)(PATTBL0+3);
            PTN3_MRA = 0xc6;
            PTN3_CRA = 0x0404;
            nextmode3++;
}

/************* reverse stop3 **************/
rstop3()
{
            CYC3_SAR = (long)UPTBL;
            CYC3_MRA = 0x01;
            CYC3_CRA = 0x03e9;
            PTN3_SAR = (long)(PATTBL0+3);
            PTN3_MRA = 0x06;
            PTN3_CRA = 0x0404;
            nextmode3 = 0x01;
}
```

4．4　タイマのトリガによるＡ／Ｄ変換

ﾀｲﾏのﾄﾘｶﾞによるＡ／Ｄ変換	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	Ａ／Ｄ、ＤＭＡＣ、ＴＰＵ

仕様

（１）図１に示すようにＴＰＵの変換開始トリガによりＡ／ＤおよびＤＭＡＣを起動させ、音声信号をＡ／Ｄ変換し、ＤＭＡＣでＲＡＭに転送します。
（２）転送ＲＡＭエリアは、H'A00000～H'A0FFFFおよびH'A10000～H'A1FFFFです。
（３）Ａ／Ｄ変換器はＴＰＵのＴＧＲＡコンペアマッチにより起動します。
（４）Ｈ８Ｓ／２６５５の内部動作周波数は２０ＭＨｚで使用します。

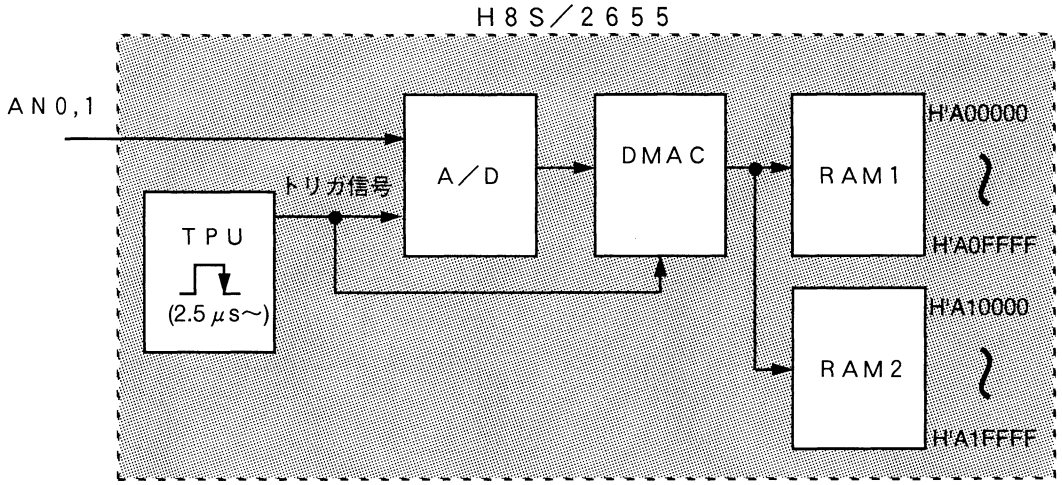


図１　タイマトリガによるＡ／Ｄ変換ブロック図

タイマのトリガによるA／D変換	MCU	H8S／2655	使用機能	A／D、DMAC、TPU

使用機能説明

（1）図2に本タスク例で使用するDMAC、A/D、TPUのブロック図を示します。

DMACの以下の機能を使用してA/D変換結果をRAMに転送します。

（a）TPUのコンペアマッチA割込みにより、DMACの動作を開始

A/Dの以下の機能を使用してサンプリングを行います。

（a）TPUによる変換開始トリガが可能

（b）2チャネルの入力電圧を同時にサンプリング（同時サンプリング動作）

TPUの以下の機能を使用してサンプリングをします。

（a）A/D変換器の変換スタートトリガを生成可能

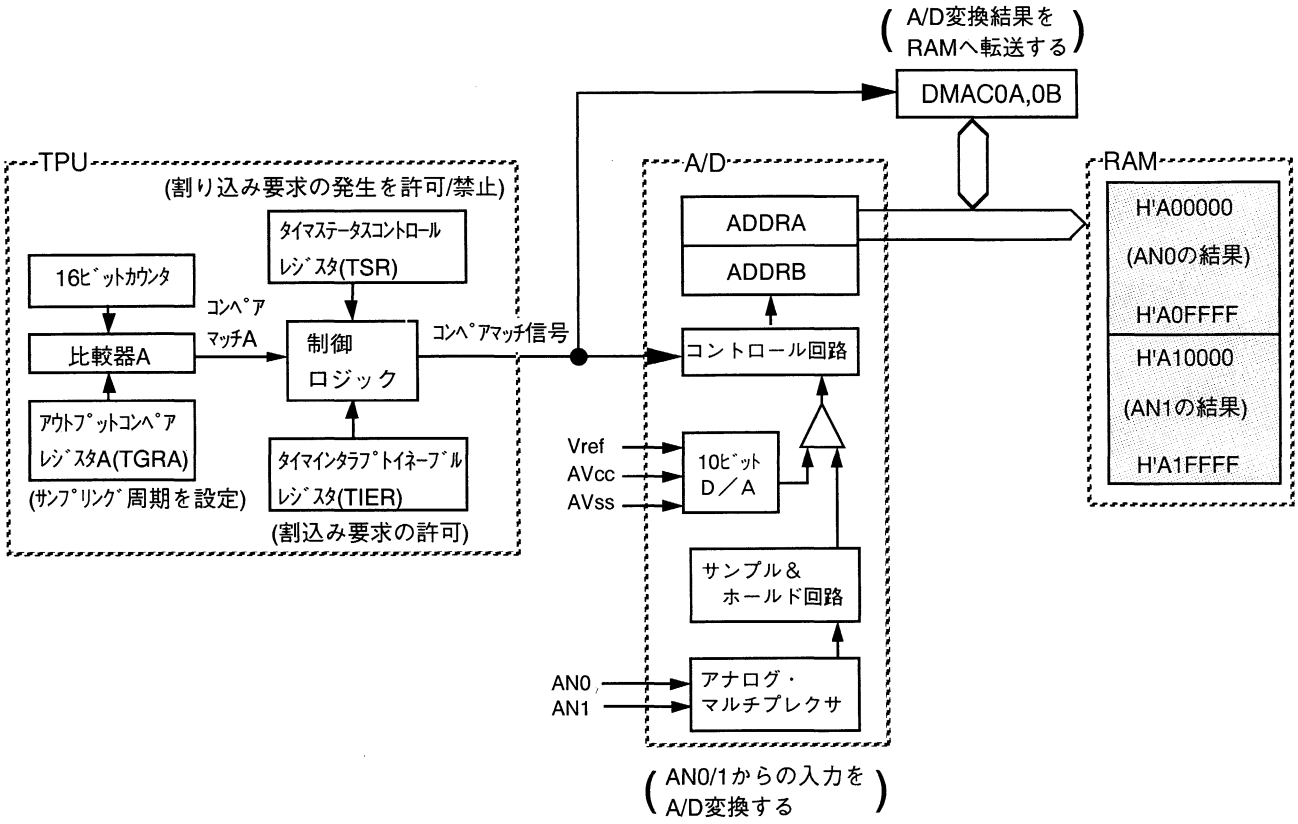


図2　タイマトリガによるA／D変換ブロック図

使用機能説明

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、A/D変換結果をRAMへ転送します。

表1　H8S／2655機能割り付け

H8S／2655の機能		機能
A／D	AN0、1	アナログ信号入力端子
	ADDRA、B	A／D変換結果を格納
DMAC	DMABCR	各チャネルの動作を制御
	DMACR	転送モードをシーケンシャルモードに設定
	MAR	転送先アドレスを設定
	IOAR	転送元アドレスを設定
	ETCR	転送回数を設定
TPU	TGR	周期を設定
	TCR	クロック、カウンタクリア要因などを選択
	TIOR	TGRをアウトプットコンペアレジスタに設定

タイマのトリガによるA／D変換	MCU	H8S／2655	使用機能	A／D、DMAC、TPU

動作説明

図3に動作原理を示します。図3に示すようにH8S/2655のハードウェア処理によりA/D変換値をRAMに格納します。

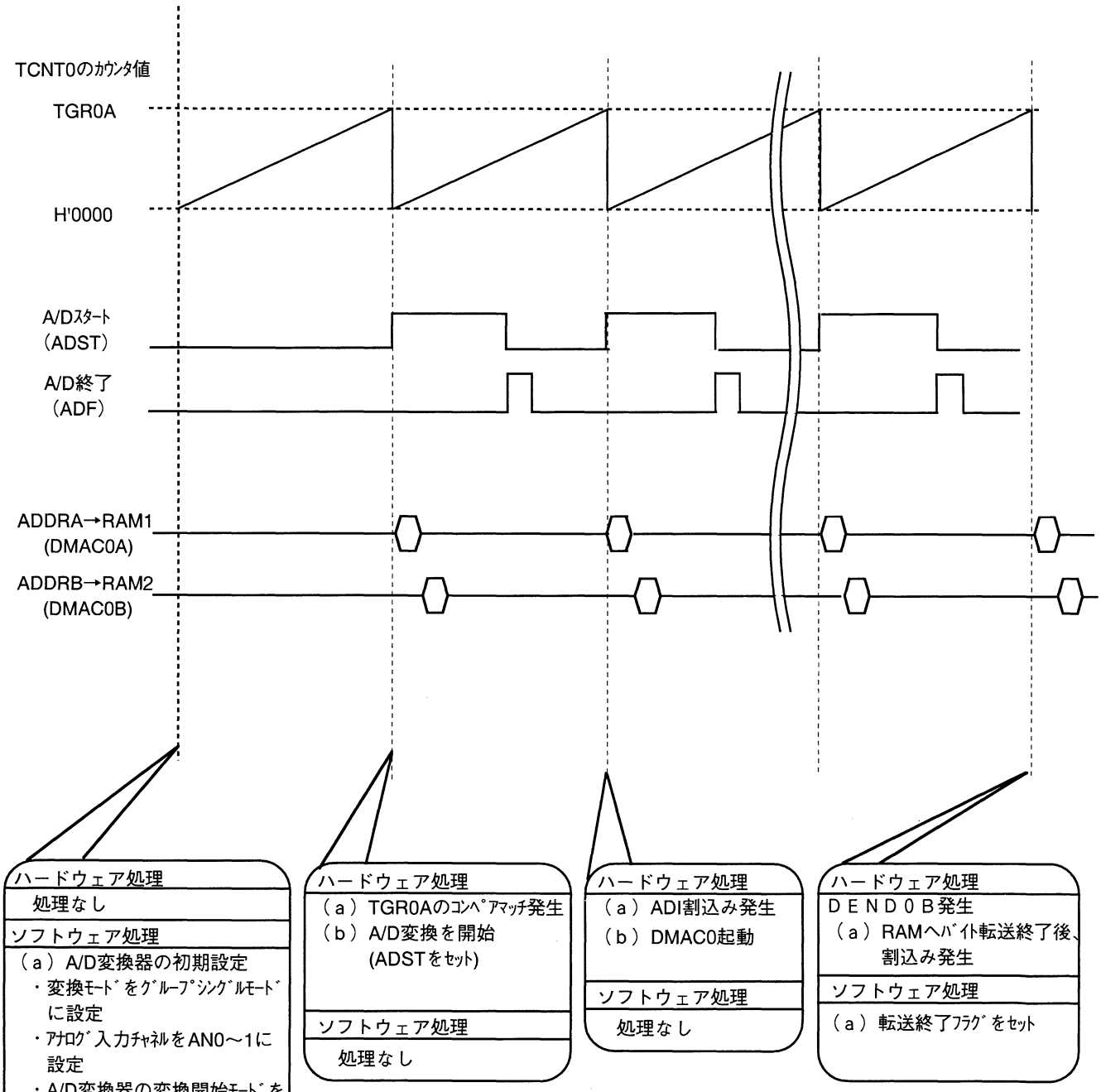


図3　タイマトリガによるA／D変換動作原理

ﾀｲﾏのﾄﾘｶﾞによるA／D変換	MCU	H8S／2655	使用機能	A／D、DMAC、TPU

ソフトウェア説明

（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	tpuadmn	TPU、DMACおよびA/Dの初期設定、使用RAMの設定を行う。
A/D変換終了	adend	A/D変換終了フラグの設定を行う。

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
ad_end	H'A00000～H'A1FFFFまでのデータ転送の終了を示す 1：データ転送終了 0：データ転送中	unsigned char	メインルーチン A/D変換終了	入力 出力
ad_data	AN0,1のA/D変換結果は、DMA転送によりそれぞれaddata0,1を先頭にバイト単位で格納 変換結果は以下のようにRAMに転送 上位ﾋﾞｯﾄ AD9 AD8 AD7 AD6 AD5 AD4 AD3 AD2	unsigned char	メインルーチン	入力
sum_cyc	A/D変換のサンプリング周期に相当するタイマ値を設定 周期(ns)=ﾀｲﾏｶｳﾝﾀ値×φ周期(20MHz動作時50ns)	unsigned short	メインルーチン	入力

（3）使用内部レジスタの説明

内蔵機能	レジスタ名	機能
TPU	TGRA	A/D変換のサンプリング周期を設定
	TIER	TGIEA割り込みを許可
	TCR	TPU0を以下のように設定 ・TGRAのコンペアマッチでカウンタクリア ・内部クロックφでカウント
	TIOR	TGRAをアウトプットコンペアレジスタに設定し、端子出力を禁止
	TSTR	TCNT0のカウント動作を許可
DMAC	DMABCR	各チャネルの動作を制御
	DMACR0A	DMAC0Aを以下のように設定 ・バイトサイズ転送 ・シーケンシャルモード ・内部割込み要因のDMA転送時クリアの許可 ・データ転送の許可
	DMACR0B	DMAC0Bを以下のように設定 ・バイトサイズ転送 ・シーケンシャルモード ・内部割込み要因のDMA転送時クリアの許可 ・データ転送、および転送終了割込みの許可
	IOAR0	転送元アドレスを設定
	MAR0	転送先アドレスを設定
	ETCR0	転送回数（H'0000）を設定
A/D	ADCR	ADCRを以下のように設定 ・高速スタートモード ・TPU0によるA/D変換の開始 ・シングルモード ・同時サンプリング動作
	ADCSR	ADCSRを以下のように設定 ・A/D変換終了割込みを許可 ・グループモード ・入力チャネルをAN0～AN1
MSTPCR		モジュールストップモードを解除

（4）使用ＲＡＭ説明

本タスク例では引数以外のRAMは使用していません。

ﾀｲﾏのﾄﾘｶﾞによるA／D変換	MCU	H8S／2655	使用機能	A／D、DMAC、TPU

PAD

（1）メインルーチン

ﾀｲﾏのﾄﾘｶﾞによる
A／D変換

tpuadmn

A/D、TPU、DMACのモジュール
ストップモードを解除

ADCRを以下のように設定
・高速スタートモード
・TPUによるA/D変換の開始
・シングルモード
・同時サンプリング動作

ADCSRを以下のように設定
・A/D変換終了割込みを許可
・グループモード

DMABCRHを以下のように設定
・ﾁｬﾈﾙ0をショートアドレスモード
・内部割込み要因のDMA転送時
クリア許可

転送元アドレス(ADDRA)を
IOAR0Aに転送先アドレス
(H'A00000)をMAR0Aに設定

転送元アドレス(ADDRB)を
IOAR0Bに転送先アドレス
(H'A10000)をMAR0Bに設定

転送回数（65536回）を
ETCRにそれぞれ設定

DMACR0Aを以下のように設定
・転送データサイズをバイト
・TPUのｺﾝﾍﾟｱﾏｯﾁAで起動
・シーケンシャルモードで転送

DMACR0Bを以下のように設定
・転送データサイズをバイト
・TPUのｺﾝﾍﾟｱﾏｯﾁAで起動
・シーケンシャルモードで転送

DMABCRLのリード

DMABCRLでチャネル0A,0Bの
データ転送、チャネル0Bの転送
終了割込みを許可に設定

1

1

TCRをTGRAのコンペアマッチで
TCNTクリアに設定

TIORでTGRAをアウトプット
コンペアレジスタに設定

TPUのTGRAにサンプリング周期
を設定

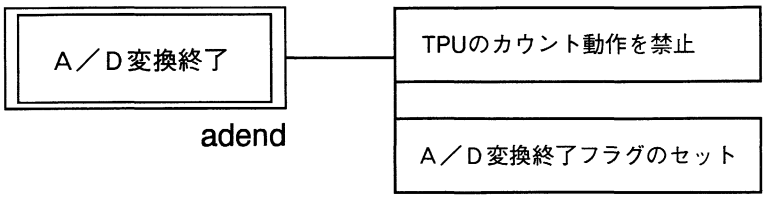
TIERでTGIA割込みを許可

Iフラグをクリアし割込みを許可

TPU0のカウント動作を許可

While（1）

（２）Ａ／Ｄ変換終了



プログラムリスト

```
/****************************************/
/*          FILE NAME : ap21.c          */
/****************************************/
#include <machine.h>
#include <h8s.h>

/****************************************/
/*          PROTCOL                     */
/****************************************/
void tpuadmn(void);

/****************************************/
/*          RAM ALLOCATION              */
/****************************************/
#define ad_end (*(volatile unsigned char *)0xffec00)
#define sum_cyc (*(volatile unsigned short *)0xffec01)
volatile struct ad_data
{
                unsigned char   data1[65536];
                unsigned char   data2[65536];
};
#define ad (*(struct ad_data *)0xA00000)

/****************************************/
/*       MAIN PROGRAM : tpuadmn         */
/****************************************/
void tpuadmn(void)
{
        MSTPCR = 0x5dff;
                ADCR = 0x54;
                ADCSR = 0x49;

                DMABCRH = 0x03;
                IOAR0A = (long)(&ADDRA);
                IOAR0B = (long)(&ADDRB);
                MAR0A = (long)(&ad.data1);
                MAR0B = (long)(&ad.data2);
                ETCR0A = 0x0000;
                ETCR0B = 0x0000;
                DMACR0A = 0x11;
                DMACR0B = 0x11;
                DMABCRL |= 0x32;

                TPU_TCR0 = 0x20;
                TIOR0H = 0x00;
                TGR0A = sum_cyc;
                TIER0 = 0xc0;

                set_imask_ccr(0);

                TSTR = 0x01;
                while(1);
}
```

プログラムリスト

```
/****************************************
/
/*    NAME : adend(set end flag)          *
/
/****************************************
/
#pragma interrupt(adend)
void adend(void)
{
                TSTR = 0x00;
                ad_end = 0x01;
                DMABCRL &= 0xcd;
                ADCSR_BP.ADIE = 0;
}
```

4．5　　D／A変換

D／A変換	MCU	H8S／2655	使用機器	TPU、DMAC、D／A

仕様

（1）図1に示すようにＴＰＵのｃｈ０，１でＤＭＡＣを起動し、ＲＡＭに格納したデータをＤ／Ａ変換します。
（2）ＲＡＭエリアは、H’Ａ００００００～H’Ａ１ＦＦＦＦです。
（3）Ｈ８Ｓ／２６５５の内部動作周波数は２０ＭＨｚで使用します。

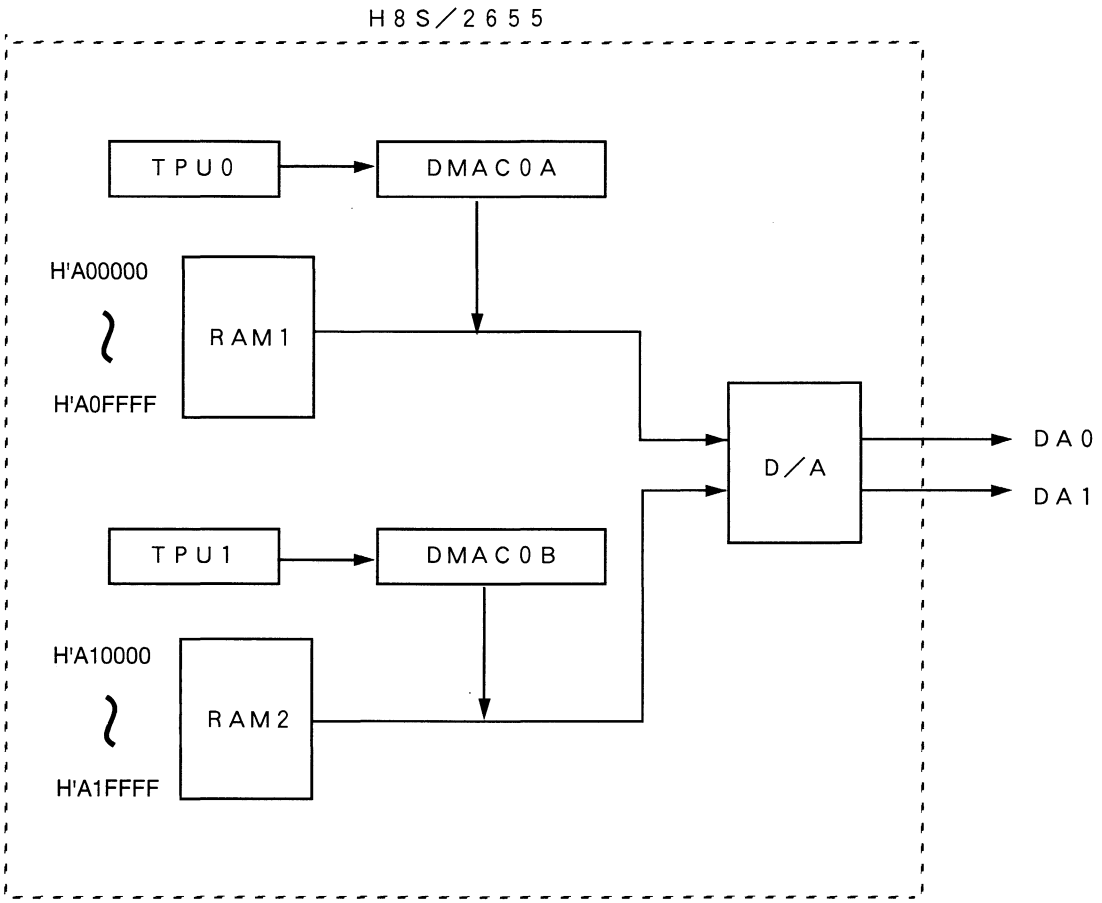


図1　D／A変換ブロック図

D／A変換	MCU	H8S／2655	使用機能	TPU、DMAC、D／A

使用機能説明

（1）図2に本タスクで使用するDMAC、D／A、TPUのブロック図を示します。本タスク例では、H8S／2655の機能を以下のように使用し、D／A変換を行います。

【DMAC】
TPUのコンペアマッチAで起動し、データバッファからD/AのDADRに転送します。

【TPU】
ch0、ch1を同期動作させ、DMACを起動します。
ch1のコンペアマッチAが起きるごとにタイマカウンタをクリアします。

【D／A】
DADRに変換データがライトされるとただちに、D/A変換が開始され変換時間経過後に変換結果が出力されます。また、AVccを基準電圧として、アナログ変換電圧範囲を設定することができます。

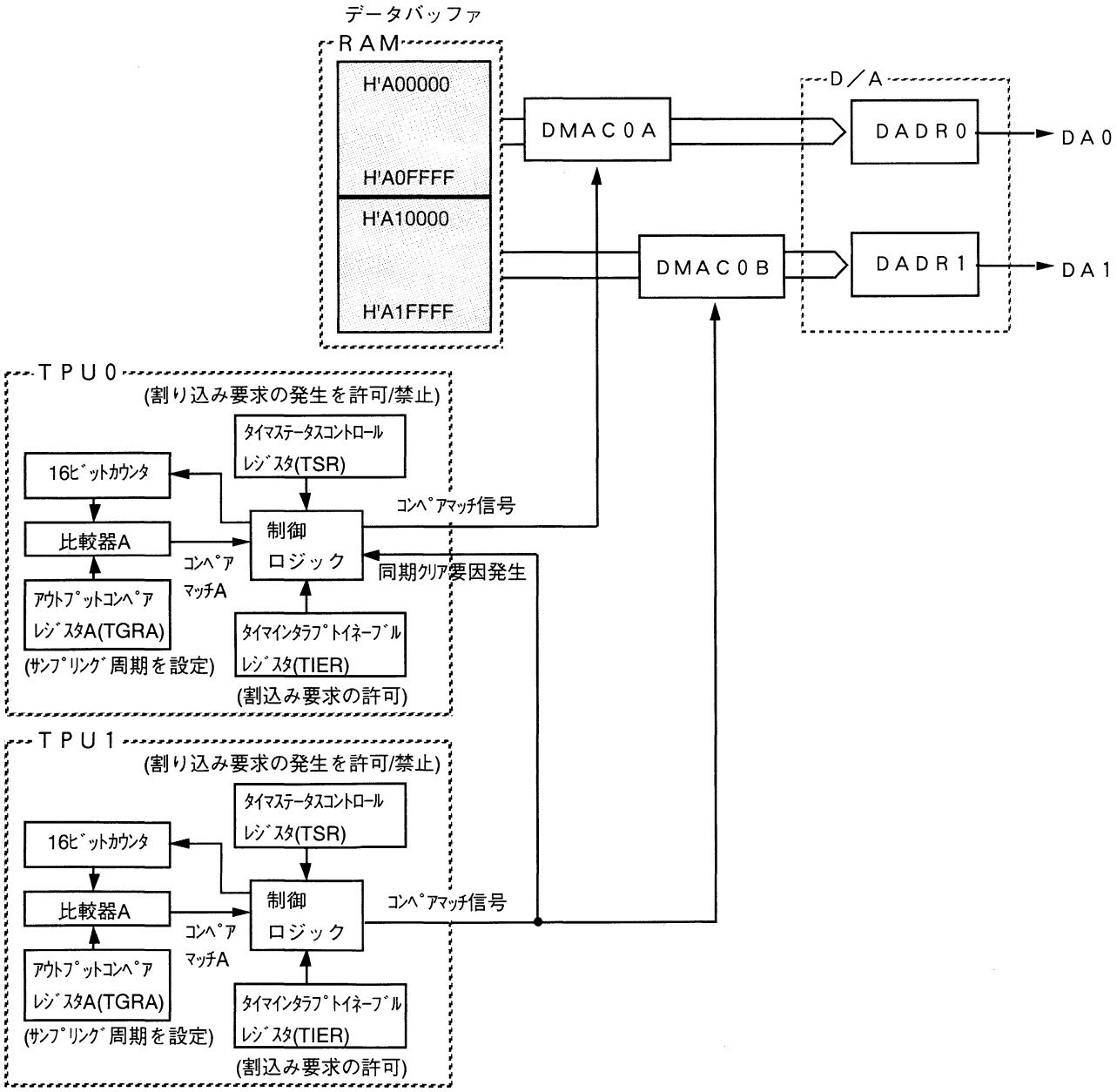


図2　アナログ出力回路ブロック図

（2）表1に本タスク例の機能割り付けを示します。表1に示すようにH8S／2655の機能を割り付け、D/A変換を行います。

表1　H8S／2655の機能割り付け

H8S／2655の機能		機能
TPU	TCNT0	16ビットカウンタ
	TGR0A	アウトプットコンペアレジスタ
	TCR0	カウンタクロックの選択およびカウンタクリア要因を選択
	TSR0	コンペアマッチやオーバフローのステータスを示す
	TIER0	割り込みの許可／禁止を選択
	TCNT1	16ビットカウンタ
	TGR1A	アウトプットコンペアレジスタ
	TCR1	カウンタクロックの選択およびカウンタクリア要因を選択
	TSR1	コンペアマッチやオーバフローのステータスを示す
	TIER1	割り込みの許可／禁止を選択
	TSYR	ch0、ch1を同期動作に設定
DMAC	DMABCR	各チャネルの動作を制御
	DMACR0	転送モードをシーケンシャルモードに設定
	MAR0A	データの先頭アドレスを設定
	MAR0B	データの先頭アドレスを設定
	IOAR0A	DADR0のアドレスを設定
	IOAR0B	DADR1のアドレスを設定
	ETCR0A	転送回数を設定
	ETCR0B	転送回数を設定
D/A	DADR0	変換を行うデータを格納（AN0側）
	DADR1	変換を行うデータを格納（AN1側）
	DACR	D/A変換器の動作を制御
	AVcc	アナログ部の電源および基準電圧
	AVss	アナログ部のグランドおよび基準電圧
	DA0	アナログ出力
	DA1	アナログ出力

D／A変換	MCU	H8S／2655	使用機能	TPU、DMAC、D／A

動作説明

図3に動作原理を示します。図3に示すようにH8S／2655のハードウェア処理およびソフトウェアの処理によりD/A変換を行います。

（1）アナログ出力

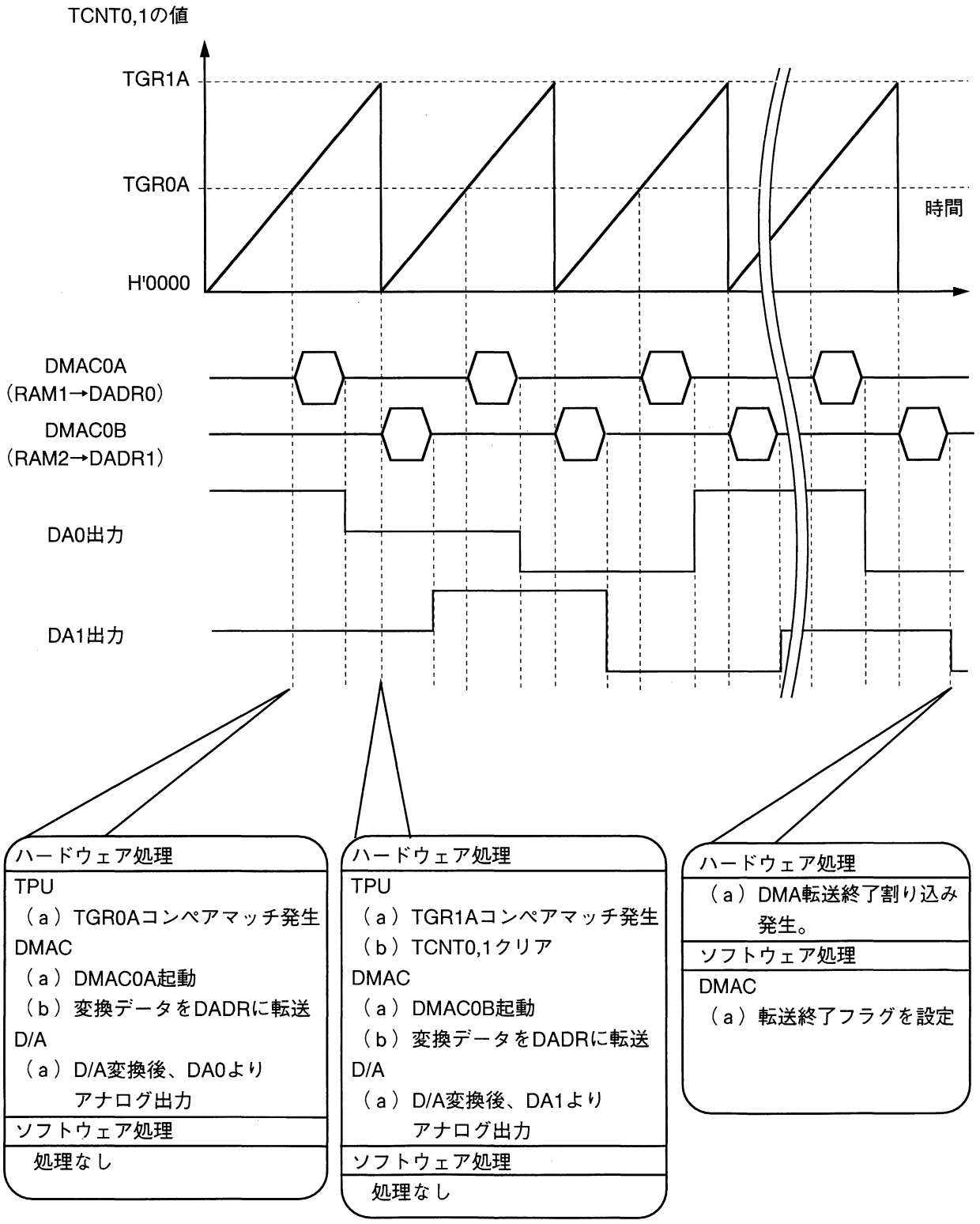


図3　アナログ出力動作原理

D／A変換	MCU	H8S／2655	使用機能	TPU、DMAC、D／A

ソフトウェア説明

（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	dacvtmn	TPU、DMACおよびD/Aの初期設定、使用RAMの設定を行う。
D/A変換終了	datrend	D/A変換終了フラグの設定を行う。

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名	入出力
da_end	H'A00000～H'A1FFFFまでのデータ転送の終了を示す。 1：データ転送終了 0：データ転送中	unsigned char	メインルーチン D/A変換終了	入力 出力

（3）使用内部レジスタの説明

内蔵機能	レジスタ名	機能
TPU	TGR0A	A/D変換のサンプリング周期を設定
	TIER0	TGIA割り込みを許可
	TCR0	TPU0を以下のように設定 ・同期クリアに設定 ・内部クロックφでカウント
	TIOR0	TGR0Aをアウトプットコンペアレジスタに設定し、端子出力を禁止
	TGR1A	A/D変換のサンプリング周期を設定
	TIER1	TGIA割り込みを許可
	TCR1	TPU0を以下のように設定 ・TGR1Aのコンペアマッチでカウンタクリア ・内部クロックφでカウント
	TIOR1	TGR1Aをアウトプットコンペアレジスタに設定し、端子出力を禁止
	TSTR	TCNT0、1のカウント動作を許可
	TSYR	ﾁｬﾈﾙ0、1を同期動作に設定
DMAC	DMABCR	各チャネルの動作を制御
	DMACR0A	DMAC0Aを以下のように設定 ・バイトサイズ転送 ・シーケンシャルモード ・内部割込み要因のDMA転送時クリアの許可 ・データ転送の許可
	DMACR0B	DMAC0Bを以下のように設定 ・バイトサイズ転送 ・シーケンシャルモード ・内部割込み要因のDMA転送時クリアの許可 ・データ転送、および転送終了割込みの許可
	MAR0A	転送元アドレス（RAM1先頭番地）を設定
	MAR0B	転送元アドレス（RAM2先頭番地）を設定
	IOAR0A	転送先アドレス（DADR0）を設定
	IOAR0B	転送先アドレス（DADR1）を設定
	ETCR0A	転送回数（H'0000）を設定
	ETCR0B	転送回数（H'0000）を設定
D/A	DACR0	DACRを以下のように設定 ・ﾁｬﾈﾙ0のD/A変換、およびアナログ出力DA0を許可
	DADR0	変換を行うデータを格納
	DADR1	変換を行うデータを格納
MSTPCR		モジュールストップモードを解除

（4）使用ＲＡＭ説明

ラベル名	機能	データ長	データ容量
da_data1,2	D/A変換データを格納する	unsigned char	128Kバイト

D／A変換	MCU	H8S／2655	使用機能	TPU、DMAC、D／A

PAD

（1）メインルーチン

D／A変換

dacvtmn

D/A、TPU、DMACのモジュールストップモードを解除

DACRを以下のように設定
・ﾁｬﾈﾙ0のD/A変換、およびアナログ出力DA0を許可

DMABCRHを以下のように設定
・ﾁｬﾈﾙ0をショートアドレスモード
・内部割込み要因のDMA転送時クリア許可

転送先アドレス(ADDRA)をIOAR0Aに転送元アドレス(H'A00000)をMAR0Aに設定

転送先アドレス(ADDRB)をIOAR0Bに転送元アドレス(H'A10000)をMAR0Bに設定

転送回数（65536回）をETCRにそれぞれ設定

DMACR0Aを以下のように設定
・転送データサイズをバイト
・TPU0のｺﾝﾍﾟｱﾏｯﾁAで起動
・シーケンシャルモードで転送

DMACR0Bを以下のように設定
・転送データサイズをバイト
・TPU1のｺﾝﾍﾟｱﾏｯﾁAで起動
・シーケンシャルモードで転送

DMABCRLのリード

DMABCRLでチャネル0A,0Bのデータ転送、チャネル0Bの転送終了割込みを許可に設定

1

1

ﾁｬﾈﾙ0、1を同期動作に設定

TCR0のカウンタクリア要因を同期クリアに設定

TCR1のカウンタクリア要因をコンペアマッチAに設定

TPU0、1のTGRAにサンプリング周期を設定

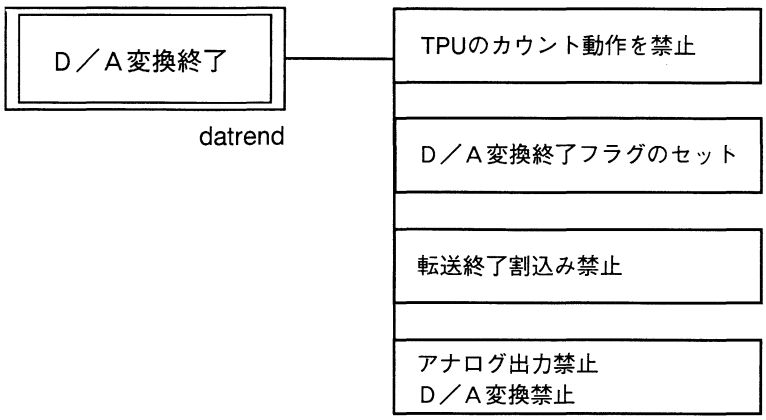
TIER0、1でTGIA割込みを許可

Iフラグをクリアし、割込みを許可

TPU0、1のカウント動作を許可

While（1）

（2）D／A変換終了


D／A変換終了
datrend
TPUのカウント動作を禁止
D／A変換終了フラグのセット
転送終了割込み禁止
アナログ出力禁止
D／A変換禁止

プログラムリスト

```
#include <machine.h>
#include <h8s.h>

/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void dacvtmn(void);

/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define trs_end (*(volatile unsigned char *)0xffec00)
#define da (*(struct da_data *)0xA00000)
volatile struct da_data
{
                unsigned char   data1[65536];
                unsigned char   data2[65536];
};

/*****************************************/
/*       MAIN PROGRAM : dacvtmn          */
/*****************************************/
void dacvtmn(void)
{
        MSTPCR = 0x5bff;
        DACR = 0x5f;

        DMABCRH = 0x03;
        IOAR0A = (long)(&DADR0);
        IOAR0B = (long)(&DADR1);
        MAR0A = (long)(&da.data1);
        MAR0B = (long)(&da.data2);
        ETCR0A = 0x0000;
        ETCR0B = 0x0000;
        DMACR0A = 0x08;
        DMACR0B = 0x09;
        DMABCRL |= 0x32;

        TSYR = 0x03;
        TPU_TCR0 = 0xe0;
        TPU_TCR1 = 0x20;
        TGR0A = 0x00c8;
        TGR1A = 0x0190;
        TIER0 = 0x41;
        TIER1 = 0x41;

        set_imask_ccr(0);

        TSTR = 0x03;
        while(1);
}
```

プログラムリスト

```
/*****************************************/
/*    NAME : datrend(set end flag)       */
/*****************************************/
#pragma interrupt(datrend)
void datrend(void)
{
        TSTR = 0x00;
        trs_end = 0x01;
        DMABCRL &= 0xcd;
        DACR = 0x1f;

}
```

４．６　　ＤＴＣ、ＤＭＡＣ、ＣＰＵ同時起動

DTC、DMAC、CPU同時起動	ＭＣＵ	Ｈ８Ｓ／２６５５	使用機能	ＤＴＣ, ＤＭＡＣ, ＴＰＵ

仕様

（１）図１に示すように、タイマコンペアマッチ発生ごとにＤＴＣ、ＤＭＡＣおよびＣＰＵ起動します。
ＤＴＣは、データテーブル（ＲＯＭ）からＰＰＧのＮＤＲへデータを転送してパルスを出力します。
ＤＭＡＣは、ＲＡＭ１に格納されている５１２バイトのデータをＲＡＭ２へ転送します。
ＣＰＵでは、ポートの状態を監視しポートがＬｏｗになったとき、ＤＴＣ、ＤＭＡＣの転送を停止します。但し、ＣＰＵへの割込みは継続します。

（２）ＤＭＡＣで転送するデータの格納番地はＨ’Ａ００００００～Ｈ’Ａ００２ＦＦです。

（３）Ｈ８Ｓ／２６５５の内部動作周波数は２０ＭＨｚで使用します。

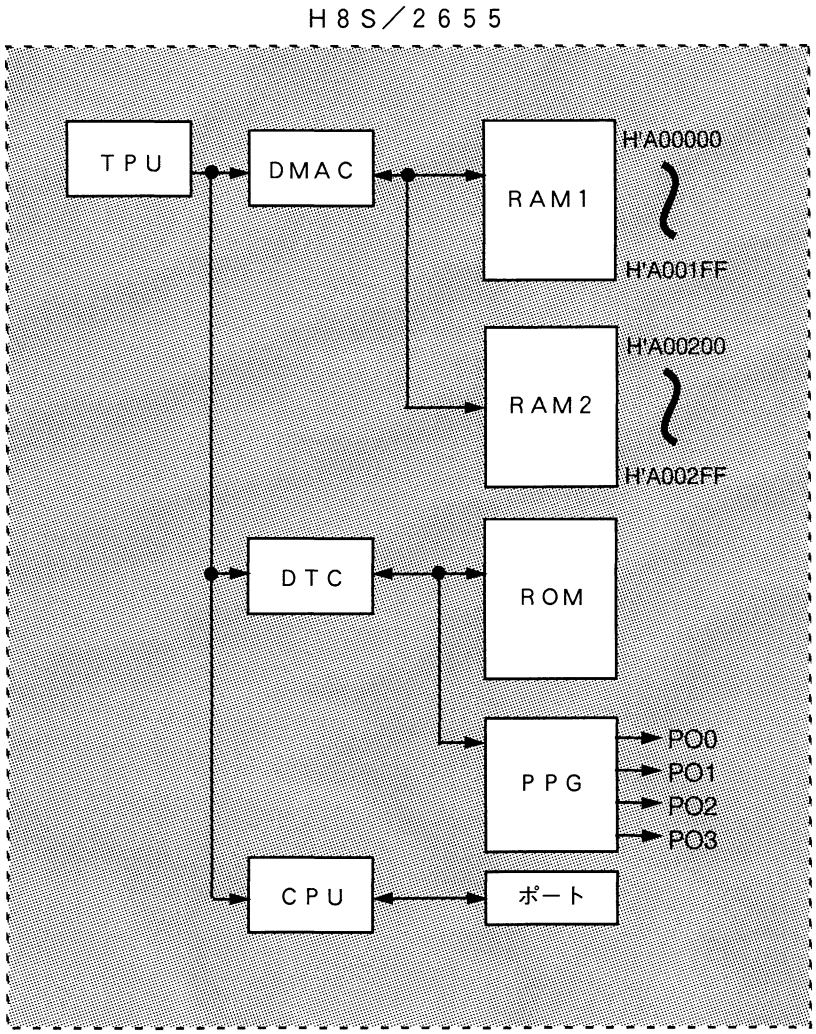


図１　ＤＴＣ、ＤＭＡＣ、ＣＰＵ同時起動 ブロック図

DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC, DMAC,TPU

使用機能

（1）本タスク例では、ＴＰＵのコンペアマッチ発生ごとにDMAC、DTC、CPUを起動させます。

（a）図2に本タスク例で使用するH8S/2655内蔵機能のブロック図を示します。

本タスク例は以下の機能を使用し、DTC、DMAC、CPUを同時に起動させデータ転送および、ポートの状態を監視します。

【ＴＰＵ】

コンペアマッチを発生させ、DTC、DMACへの転送要求およびCPU割り込み要求を発生させます。

【ＤＭＡＣ】

TPUのコンペアマッチで起動し、512バイトデータをRAM1からRAM2へ転送します。

【ＤＴＣ】、【ＰＰＧ】

TPUのコンペアマッチで起動し、データテーブルから4バイトのデータをPPGのNDRへ転送します。

【ＣＰＵ】

TPUのコンペアマッチで割込み処理を実行します。割込み処理の中で、ポートの状態を監視しDMAC、DTCの転送を制御します。

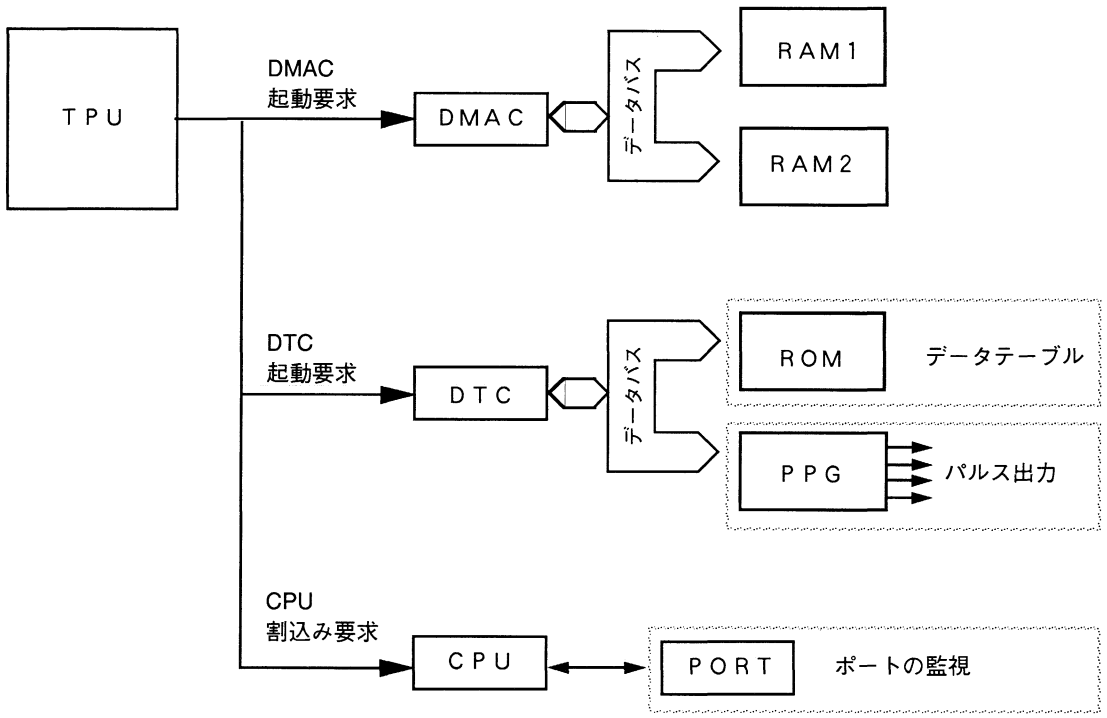


図2　ＤＴＣ、ＤＭＡＣ、ＣＰＵ同時起動ブロック

（2）表1に本タスク例の機能割り付けを示します。表1に示すように H8S／2655の機能を割り付け、データ転送を行います。

表1　H8S／2655機能割り付け

H8S/2655の機能		機能
DMAC	DMABCR	DMAC0Aを以下のように設定する ・転送モードをフルアドレスモードに設定 ・内部割込み要因のDMA転送時のクリアを禁止 ・データ転送、転送終了割込みを許可
	DMACR0A	DMAC0Aを以下のように設定する ・データサイズをバイトサイズに設定 ・MARをインクリメントに設定 ・データ転送をブロック転送モードに設定 ・データ転送の方向を設定（DMAC0A：MAR→IOAR） ・起動要因をTPU0Aに設定
	MAR0A	RAM1の先頭アドレス（転送元）を設定
	MAR0B	RAM2の先頭アドレス（転送先）を設定
	ETCR0A	転送回数を設定
TPU	TCR0	コンペアマッチでTCNTクリア
	TIOR0	コンペアマッチ出力禁止に設定
	TIER0	コンペアマッチ割り込みを許可
	TSR0	TGRAのコンペアマッチ割り込み要求フラグの設定
DTC	DTCER	TGIA割り込みによるDTC起動の許可
PPG	NDER	パルス出力PO0～PO15を許可
	NDR	パルス出力の次のデータを格納
	PCR	PPGの出力要求をTPUチャネル0のコンペアマッチに設定
	PMR	PPGの出力を直接出力に設定

DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC, DMAC, TPU

動作説明

図3に動作原理を示します。図3に示すタイミングでハードウェア処理およびソフトウェア処理を行い、DTC、DMAC、CPU割り込みの同時起動を要求します。

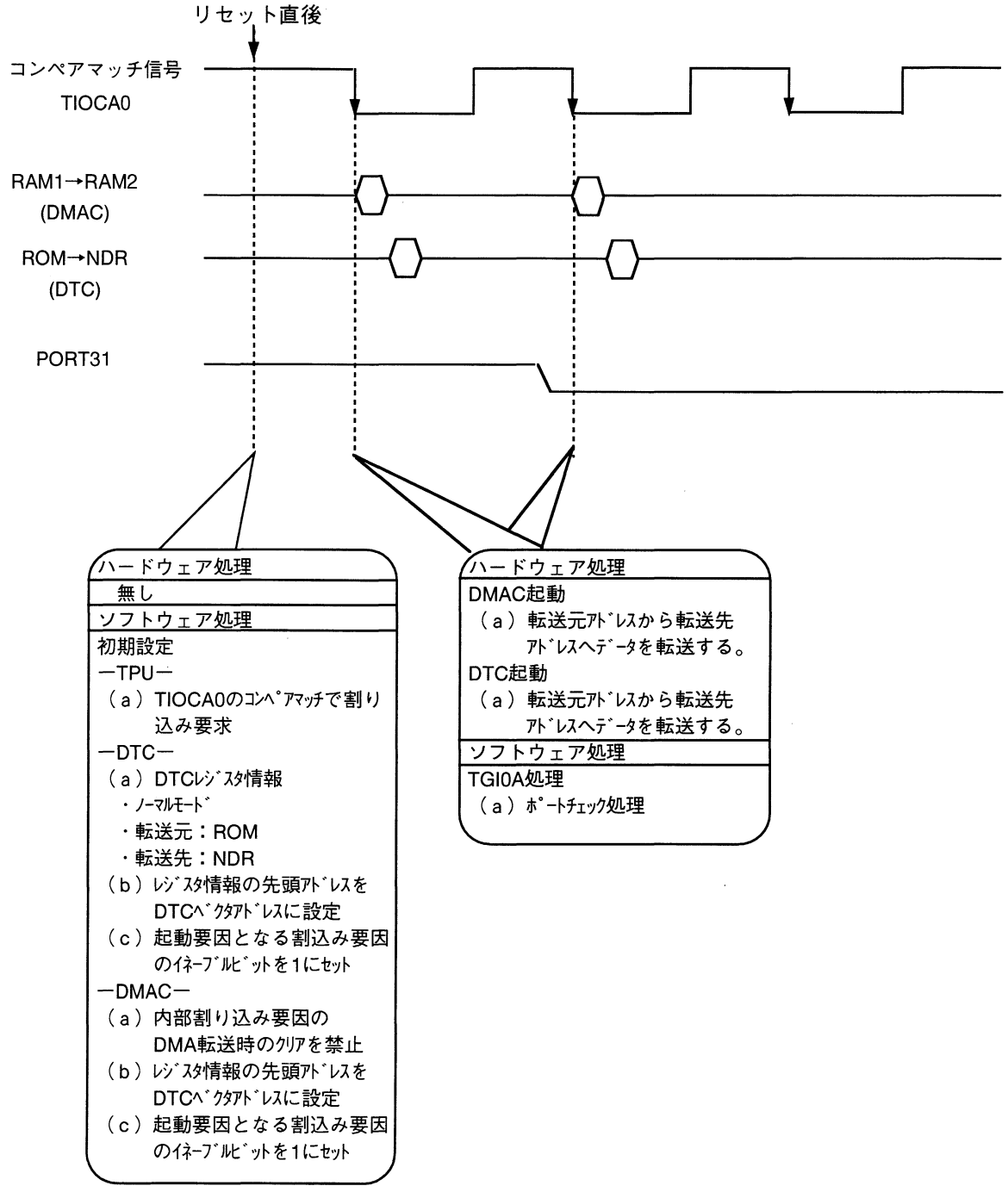


図3　DTC、DMAC、CPU同時起動動作原理

DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC, DMAC,TPU

ソフトウェア説明

（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	simbsrmn	TPU 、DTC、PPG、DMAC、割り込み処理の初期設定を行う。
ポートチェック	portchk	ポートのチェック、および転送禁止処理を行う。
データ転送終了	trsend	データ転送終了フラグの設定を行う。

（2）引数の説明

レジスタ名	機能	データ長	使用モジュール名	入出力
status	ポート31の状態を示す 0：転送許可　　1：データ転送禁止	unsigned char	ポートチェック	出力
trs_end	512バイト転送終了を示すフラグ 0：転送許可　　1：データ転送禁止	unsigned char	データ転送終了	出力

（3）使用内部レジスタ説明

内蔵機能	レジスタ名	機能
DMAC	DMABCR	DMAC0Aを以下のように設定する ・転送モードをフルアドレスモードに設定 ・内部割込み要因のDMA転送時のクリアを禁止 ・データ転送、転送終了割込みを許可
	DMACR0A	DMAC0Aを以下のように設定する ・データサイズをバイトサイズに設定 ・MARをインクリメントに設定 ・データ転送をブロック転送モードに設定 ・データ転送の方向を設定（ch0A：MAR→IOAR） ・起動要因をTPU0Aに設定
	MAR0A	RAM1の先頭アドレス（trs）を設定
	MAR0B	RAM2の先頭アドレス（rev）を設定
	ETCR0A	転送回数を設定
TPU	TCR0	コンペアマッチでTCNTクリア
	TIOR0	コンペアマッチ出力禁止に設定
	TIER0	コンペアマッチ割り込みを許可
	TSR0	TGRAのキャプチャ割り込み要求フラグの設定
DTC	DTCER	TGIA割り込みによるDTC起動の許可
PPG	NDER	パルス出力PO0～PO15を許可
	NDR	パルス出力の次のデータを格納
	PCR	全てのパルス出力グループをTPUチャネル0のコンペアマッチに設定
	PMR	全てのパルス出力グループを直接出力に設定
MSTPCR		DTC,TPU,DMAC,PPGのモジュールストップモードを制御

（4）使用RAM説明

ラベル名	機能	データ長	使用モジュール名
MRA0	DTC0 をノーマルモードに設定	unsigned char	メインルーチン
MRB0	CPUへの割込み許可	unsigned char	
SAR0	転送元アドレス（PATTBL1）を設定	unsigned long	
DAR0	転送先アドレス（P1DR）を設定	unsigned long	
CRA0	転送回数を設定	unsigned short	
trs	送信データを格納	512バイト	
rev	受信データを格納	512バイト	
PATTBL1	PPG出力データを格納	4バイト	

DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC, DMAC,TPU

PAD

（1）メインルーチン

同時起動

simbsrmn

TPU、DMA、DTCのモジュールストップモードを解除

DMABCRHを以下ように設定
・フルアドレスモード
・内部割込みクリアの禁止

転送元アドレス（MAR0A）をＲＡＭ１の先頭アドレスに設定

転送先アドレス（MAR0B）をＲＡＭ２の先頭アドレスに設定

DMACRAを以下のように設定
・データサイズをバイトサイズ
・MARAをインクリメント
・ﾌﾞﾛｯｸ転送ﾓｰﾄﾞに設定
・TPU ch0のｺﾝﾍﾟｱﾏｯﾁで起動

ETCRに転送回数（128回）を設定

DMABCRLのリード

DMABCRLを以下ように設定
・チャネル0の転送許可
・チャネル0の転送終了割込みを許可

送信ﾃﾞｰﾀﾊﾞｯﾌｧの先頭アドレスをSAR0に設定

MRA0を以下ように設定
・SARは転送後インクリメント
・DARは転送後インクリメント
・データ転送をノーマルモード
・バイトサイズ転送

NDRのアドレスをDAR0に設定

CPUへの割り込みを許可

転送カウント(H'04)をCRA0に設定

TIORをｺﾝﾍﾟｱﾏｯﾁ許可に設定

TIERでTGIA割込みを許可

Iフラグをクリアし割込みを許可

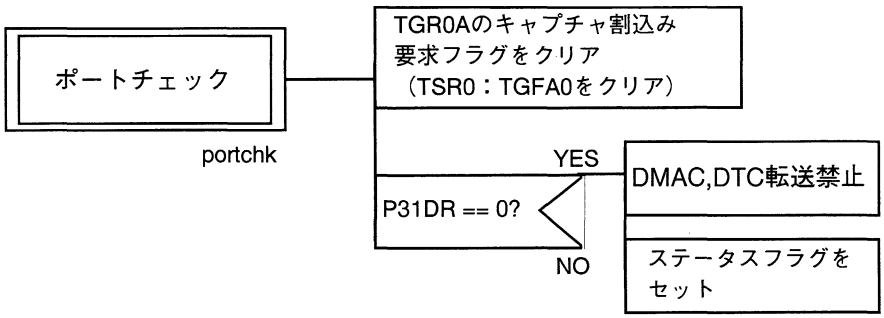
TPU0のカウント動作を許可

Ｗｈｉｌｅ（1）

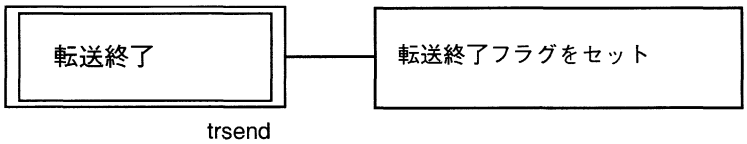
DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC, DMAC,TPU

PAD

（2）ポートチェック



（2）データ転送終了



DTC、DMAC、CPU同時起動	MCU	H8S／2655	使用機能	DTC,DMAC,TPU

プログラムリスト

```
/*****************************************/
/*        FILE NAME : ap17.c             */
/*****************************************/
#include <machine.h>
#include "..¥h8sapn¥h8s.h"
/*****************************************/
/*          PROTCOL                      */
/*****************************************/
void simbsrmn (void);
#pragma interrupt(trsend)
#pragma interrupt(portchk)
/*****************************************/
/*          RAM ALLOCATION               */
/*****************************************/
#define status (*(volatile unsigned char *)0xffec00)
#define trs_end (*(volatile unsigned char *)0xffec01)
volatile struct databuf
{
                unsigned char   trs[512];
                unsigned char   rev[512];
};
#define dat (*(struct databuf *)0xa00000)
#define SAR0 (*(volatile unsigned long *)0xfff800)
#define MRA0 (*(volatile unsigned char *)0xfff800)
#define DAR0 (*(volatile unsigned long *)0xfff804)
#define MRB0 (*(volatile unsigned char *)0xfff804)
#define CRA0 (*(volatile unsigned char *)0xfff808)
#define CRB0 (*(volatile unsigned char *)0xfff80a)
const unsigned char PATTBL1[4] = {0xf6,0xf3,0xf9,0xfc};
/*****************************************/
/*        MAIN PROGRAM : simbsrmn        */
/*****************************************/
void simbsrmn(void)
{
        status = 0;                     /* flag clr */
        trs_end = 0;                    /* flag clr */
        MSTPCR = 0x17ff;                /* Disable module(PPG,DMAC,DTC) stop mode*/
        P1DDR = 0xff;                   /* P1,2 : output */
        P2DDR = 0xff;
        NDERH = 0xff;                   /* Set next data enable */
        NDERL = 0xff;
        PCR = 0x00;                     /* Set trigger TPU0 compare match */
        PMR = 0xf0;                     /* Set normal mode */
        DMABCRH = 0x40;                 /* Initialize DMABCRH */
        MAR0A = (long)(&dat.trs);       /* Set base address */
        MAR0B = (long)(&dat.rev);       /* Set excute address */
        ETCR0A = 0x01ff;                /* Set excute count */
        DMACR0A = 0x20;                 /* Initialize DMACR0 */
        DMACR0B = 0x27;
        DMABCRL |= 0x30;                /* Initialize DMABCRL */
        SAR0 = (long)(PATTBL1);         /* Set base address */
        MRA0 = 0x86;                    /* Set repeat mode */
        DAR0 = (long)(&NDRH);           /* Set excute address */
        MRB0 = 0x40;                    /* Initialize MRB */
        CRA0 = 0x0404;                  /* Set excute count */
        DTCERB_BP.TGI0A = 1;            /* Enable DTC */
        TCR0 = 0x20;
        TGR0A = 0x1000;                 /* Initialize TGR0A */
        TIOR0H = 0x00;                  /* Initialize TIOR0H */
        TIER0 = 0x01;                   /* Enable TGI0A interrupt */
        set_imask_ccr(0);               /* Enable interrupt */
        TSTR = 0x01;
        while(1);
}
```

プログラムリスト

```
/*****************************************/
/*        NAME : portchk                 */
/*****************************************/
void portchk(void)
{
        TSR0_BP.TGFA0 = 0;                          /* Clear TGFA0 flag */

                if (P3DR_BP.P31DR == 1)             /* Check port31 */
                        {
                        status = 1;
                        DMABCRL |= 0x30;            /* Initialize DMABCRL */
                        DTCERB_BP.TGI0A = 1;        /* Disble DTC */
                        }
                else
                        {
                        DMABCRL |= 0x00;            /* Disable DMAC */
                        DTCERB_BP.TGI0A = 0;        /* Disble DTC */
                        }
}
/*****************************************/
/*        NAME : trsend                  */
/*****************************************/
void trsend(void)
{
                ETCR0A = 0x01ff;                    /* Set excute count */
                trs_end = 1;                        /* Set DMAC end flag */
}
```

5．付録

目　次

H８S／２６５５　ヘッダーファイル（１）　<H８S．H>

```
/***************************************/
/*         SYMBOL DEFNITIONS           */
/***************************************/
struct ISCR_S{                          /* IRQ sense control register */
        unsigned char   IRQ7SCB:1;
        unsigned char   IRQ7SCA:1;
        unsigned char   IRQ6SCB:1;
        unsigned char   IRQ6SCA:1;
        unsigned char   IRQ5SCB:1;
        unsigned char   IRQ5SCA:1;
        unsigned char   IRQ4SCB:1;
        unsigned char   IRQ4SCA:1;
        unsigned char   IRQ3SCB:1;
        unsigned char   IRQ3SCA:1;
        unsigned char   IRQ2SCB:1;
        unsigned char   IRQ2SCA:1;
        unsigned char   IRQ1SCB:1;
        unsigned char   IRQ1SCA:1;
        unsigned char   IRQ0SCB:1;
        unsigned char   IRQ0SCA:1;
};
#define ISCR_BP (*(struct ISCR_S *)0xffff2c)

struct ISR_S{                           /* IRQ0 status register */
        unsigned char   IRQ7F:1;
        unsigned char   IRQ6F:1;
        unsigned char   IRQ5F:1;
        unsigned char   IRQ4F:1;
        unsigned char   IRQ3F:1;
        unsigned char   IRQ2F:1;
        unsigned char   IRQ1F:1;
        unsigned char   IRQ0F:1;
};
#define ISR_BP (*(struct ISR_S *)0xffff2f)

struct IPRA_S{                          /* interrupt priority registerA *
/
        unsigned char   dummy1:1;
        unsigned char   IPRA6:1;
        unsigned char   IPRA5:1;
        unsigned char   IPRA4:1;
        unsigned char   dummy2:1;
        unsigned char   IPRA2:1;
        unsigned char   IPRA1:1;
        unsigned char   IPRA0:1;
};
#define IPRA_BP (*(struct IPRA_S *)0xfffec4)

struct DMABCR_S{                        /* DMA band control register */
        unsigned char   FAE1:1;
        unsigned char   FAE0:1;
        unsigned char   SAE1:1;
        unsigned char   SAE0:1;
        unsigned char   DTA1B:1;
        unsigned char   DTA1A:1;
        unsigned char   DTA0B:1;
        unsigned char   DTA0A:1;
        unsigned char   DTE1B:1;
        unsigned char   DTE1A:1;
        unsigned char   DTE0B:1;
        unsigned char   DTE0A:1;
        unsigned char   DTIE1B:1;
        unsigned char   DTIE1A:1;
        unsigned char   DTIE0B:1;
        unsigned char   DTIE0A:1;
};
#define DMABCR_BP (*(struct DMABCR_S *)0xffff06)
```

```
struct DTCERA_S{                                    /* DTC vector register */
        unsigned char   IRQ0:1;
        unsigned char   IRQ1:1;
        unsigned char   IRQ2:1;
        unsigned char   IRQ3:1;
        unsigned char   IRQ4:1;
        unsigned char   IRQ5:1;
        unsigned char   IRQ6:1;
        unsigned char   IRQ7:1;
};
#define DTCERA_BP (*(struct DTCERA_S *)0xffff30)

struct DTCERB_S{                                    /* DTC vector register */
        unsigned char   dummy3:1;
        unsigned char   ADI:1;
        unsigned char   TGI0A:1;
        unsigned char   TGI0B:1;
        unsigned char   TGI0C:1;
        unsigned char   TGI0D:1;
        unsigned char   TGI1A:1;
        unsigned char   TGI1B:1;
};
#define DTCERB_BP (*(struct DTCERB_S *)0xffff31)

struct DTCERC_S{                                    /* DTC vector register */
        unsigned char   TGI2A:1;
        unsigned char   TGI2B:1;
        unsigned char   TGI3A:1;
        unsigned char   TGI3B:1;
        unsigned char   TGI3C:1;
        unsigned char   TGI3D:1;
        unsigned char   TGI4A:1;
        unsigned char   TGI4B:1;
};
#define DTCERC_BP (*(struct DTCERC_S *)0xffff32)

struct DTVECR_S{                                    /* DTC vector register */
        unsigned char   SWDTE:1;
        unsigned char   VECR:7;
};
#define DTVECR_BP (*(struct DTVECR_S *)0xffff37)

struct P2DR_S{                              /* port2 data register */
        unsigned char   P27DR:1;
        unsigned char   P26DR:1;
        unsigned char   P25DR:1;
        unsigned char   P24DR:1;
        unsigned char   P23DR:1;
        unsigned char   P22DR:1;
        unsigned char   P21DR:1;
        unsigned char   P20DR:1;
};
#define P2DR_BP (*(struct P2DR_S *)0xffff61)

struct P3DDR_S{
        unsigned char P37:1;
        unsigned char P36:1;
        unsigned char P35DDR:1;
        unsigned char P34DDR:1;
        unsigned char P33DDR:1;
        unsigned char P32DDR:1;
        unsigned char P31DDR:1;
        unsigned char P30DDR:1;
};
#define P3DDR_BP (*(struct P3DDR_S *)0xfffeb2 )
```

```
struct P3DR_S{                          /* port3 data register */
        unsigned char   dummy60:1;
        unsigned char   dummy61:1;
        unsigned char   P35DR:1;
        unsigned char   P34DR:1;
        unsigned char   P33DR:1;
        unsigned char   P32DR:1;
        unsigned char   P31DR:1;
        unsigned char   TRG:1;
};
#define P3DR_BP (*(struct P3DR_S *)0xffff62)

struct PORT3_S{
unsigned char dummy43:1;
unsigned char dummy44:1;
unsigned char P35:1;
unsigned char P34:1;
unsigned char P33:1;            /*RTS*/
unsigned char P32:1;
unsigned char P31:1;            /*CTS*/
unsigned char P30:1;
};
#define PORT3_BP (*(struct PORT3_S *)0xffff52 )

struct P6DR_S{                          /* port6 data register */
        unsigned char   IRQ3:1;
        unsigned char   IRQ2:1;
        unsigned char   RRQ:1;
        unsigned char   IRQ0:1;
        unsigned char   dummy4:1;
        unsigned char   dummy5:1;
        unsigned char   dummy6:1;
        unsigned char   dummy7:1;
};
#define P6DR_BP (*(struct P6DR_S *)0xffff65)

struct PORT6_S{                         /* port6 data register */
        unsigned char   IRQ3:1;
        unsigned char   IRQ2:1;
        unsigned char   RRQ:1;
        unsigned char   SRQ:1;
        unsigned char   dummy4:1;
        unsigned char   dummy5:1;
        unsigned char   dummy6:1;
        unsigned char   dummy7:1;
};
#define PORT6_BP (*(struct PORT6_S *)0xffff55)

struct TIER0_S{                         /* timer interrupt enable
register0 */
        unsigned char   TTGE0:1;
        unsigned char   dummy8:1;
        unsigned char   dummy9:1;
        unsigned char   TCIEV0:1;
        unsigned char   TGIED0:1;
        unsigned char   TGIEC0:1;
        unsigned char   TGIEB0:1;
        unsigned char   TGIEA0:1;
};
#define TIER0_BP (*(struct TIER0_S *)0xffffd4)
```

```
struct TSR0_S{                          /* timer status register0 */
        unsigned char   dummy10:1;
        unsigned char   dummy11:1;
        unsigned char   dummy12:1;
        unsigned char   TCFV0:1;
        unsigned char   TGFD0:1;
        unsigned char   TGFC0:1;
        unsigned char   TGFB0:1;
        unsigned char   TGFA0:1;
};
#define TSR0_BP (*(struct TSR0_S *)0xffffd5)

struct TIER1_S{                         /* timer interrupt enable
register1 */
        unsigned char   TTGE1:1;
        unsigned char   dummy13:1;
        unsigned char   TCIEU1:1;
        unsigned char   TCIEV1:1;
        unsigned char   dummy14:1;
        unsigned char   dummy15:1;
        unsigned char   TGIEB1:1;
        unsigned char   TGIEA1:1;
};
#define TIER1_BP (*(struct TIER1_S *)0xffffe4)

struct TSR1_S{                          /* timer status register1 */
        unsigned char   TCFD1:1;
        unsigned char   dummy16:1;
        unsigned char   TCFU1:1;
        unsigned char   TCFV1:1;
        unsigned char   dummy17:1;
        unsigned char   dummy18:1;
        unsigned char   TGFB1:1;
        unsigned char   TGFA1:1;
};
#define TSR1_BP (*(struct TSR1_S *)0xffffe5)

struct TIER2_S{                         /* timer interrupt enable
register2 */
        unsigned char   TTGE2:1;
        unsigned char   dummy19:1;
        unsigned char   TCIEU2:1;
        unsigned char   TCIEV2:1;
        unsigned char   dummy20:1;
        unsigned char   dummy21:1;
        unsigned char   TGIEB2:1;
        unsigned char   TGIEA2:1;
};
#define TIER2_BP (*(struct TIER2_S *)0xfffff4)

struct TSR2_S{                          /* timer status register2 */
        unsigned char   TCFD2:1;
        unsigned char   dummy22:1;
        unsigned char   TCFU2:1;
        unsigned char   TCFV2:1;
        unsigned char   dummy23:1;
        unsigned char   dummy24:1;
        unsigned char   TGFB2:1;
        unsigned char   TGFA2:1;
};
#define TSR2_BP (*(struct TSR2_S *)0xfffff5)
```

```
struct TIER3_S{                         /* timer interrupt enable
register3 */
        unsigned char   TTGE3:1;
        unsigned char   dummy25:1;
        unsigned char   dummy26:1;
        unsigned char   TCIEV3:1;
        unsigned char   TGIED3:1;
        unsigned char   TGIEC3:1;
        unsigned char   TGIEB3:1;
        unsigned char   TGIEA3:1;
};
#define TIER3_BP (*(struct TIER3_S *)0xfffe84)

struct TSR3_S{                          /* timer status register3 */
        unsigned char   dummy27:1;
        unsigned char   dummy28:1;
        unsigned char   dummy29:1;
        unsigned char   TCFV3:1;
        unsigned char   TGFD3:1;
        unsigned char   TGFC3:1;
        unsigned char   TGFB3:1;
        unsigned char   TGFA3:1;
};
#define TSR3_BP (*(struct TSR3_S *)0xfffe85)

struct TIER4_S{                         /* timer interrupt enable
register4 */
        unsigned char   TTGE4:1;
        unsigned char   dummy30:1;
        unsigned char   TCIEU4:1;
        unsigned char   TCIEV4:1;
        unsigned char   dummy31:1;
        unsigned char   dummy32:1;
        unsigned char   TGIEB4:1;
        unsigned char   TGIEA4:1;
};
#define TIER4_BP (*(struct TIER4_S *)0xfffe94)

struct TSR4_S{                          /* timer status register4 */
        unsigned char   TCFD4:1;
        unsigned char   dummy33:1;
        unsigned char   TCFU4:1;
        unsigned char   TCFV4:1;
        unsigned char   dummy34:1;
        unsigned char   dummy35:1;
        unsigned char   TGFB4:1;
        unsigned char   TGFA4:1;
};
#define TSR4_BP (*(struct TSR4_S *)0xfffe95)

struct TIER5_S{                         /* timer interrupt enable
register5 */
        unsigned char   TTGE5:1;
        unsigned char   dummy36:1;
        unsigned char   TCIEU5:1;
        unsigned char   TCIEV5:1;
        unsigned char   dummy37:1;
        unsigned char   dummy38:1;
        unsigned char   TGIEB5:1;
        unsigned char   TGIEA5:1;
};
#define TIER5_BP (*(struct TIER5_S *)0xfffea4)
```

```
struct TSR5_S{                          /* timer status register5 */
        unsigned char   TCFD5:1;
        unsigned char   dummy39:1;
        unsigned char   TCFU5:1;
        unsigned char   TCFV5:1;
        unsigned char   dummy40:1;
        unsigned char   dummy41:1;
        unsigned char   TGFB5:1;
        unsigned char   TGFA5:1;
};
#define TSR5_BP (*(struct TSR5_S *)0xfffea5)

struct TSTR_S{                          /* timer start register */
        unsigned char   dummy42:1;
        unsigned char   dummy43:1;
        unsigned char   CST5:1;
        unsigned char   CST4:1;
        unsigned char   CST3:1;
        unsigned char   CST2:1;
        unsigned char   CST1:1;
        unsigned char   CST0:1;
};
#define TSTR_BP (*(struct TSTR_S *)0xffffc0)

struct SCR0_S{                          /* serial control register0 */
        unsigned char   TIE0:1;
        unsigned char   RIE0:1;
        unsigned char   TE0:1;
        unsigned char   RE0:1;
        unsigned char   MPIE0:1;
        unsigned char   TEIE0:1;
        unsigned char   CKE10:1;
        unsigned char   CKE00:1;
};
#define SCR0_BP (*(struct SCR0_S *)0xffff7a)

struct SSR0_S{                          /* serial status register0 */
        unsigned char   TDRE0:1;
        unsigned char   RDRF0:1;
        unsigned char   OPER0:1;
        unsigned char   FER0:1;
        unsigned char   PER0:1;
        unsigned char   TEND0:1;
        unsigned char   MPB0:1;
        unsigned char   MPBT0:1;
};
#define SSR0_BP (*(struct SSR0_S *)0xffff7c)

struct SCR1_S{                          /* serial control register1 */
        unsigned char   TIE1:1;
        unsigned char   RIE1:1;
        unsigned char   TE1:1;
        unsigned char   RE1:1;
        unsigned char   MPIE1:1;
        unsigned char   TEIE1:1;
        unsigned char   CKE11:1;
        unsigned char   CKE01:1;
};
#define SCR1_BP (*(struct SCR1_S *)0xffff82)
```

```
struct SSR1_S{                          /* serial status register1 */
        unsigned char   TDRE1:1;
        unsigned char   RDRF1:1;
        unsigned char   OPER1:1;
        unsigned char   FER1:1;
        unsigned char   PER1:1;
        unsigned char   TEND1:1;
        unsigned char   MPB1:1;
        unsigned char   MPBT1:1;
};
#define SSR1_BP (*(struct SSR1_S *)0xffff84)

struct SCR2_S{                          /* serial control register2 */
        unsigned char   TIE2:1;
        unsigned char   RIE2:1;
        unsigned char   TE2:1;
        unsigned char   RE2:1;
        unsigned char   MPIE2:1;
        unsigned char   TEIE2:1;
        unsigned char   CKE12:1;
        unsigned char   CKE02:1;
};
#define SCR2_BP (*(struct SCR2_S *)0xffff8a)

struct SSR2_S{                          /* serial status register2 */
        unsigned char   TDRE2:1;
        unsigned char   RDRF2:1;
        unsigned char   OPER2:1;
        unsigned char   FER2:1;
        unsigned char   PER2:1;
        unsigned char   TEND2:1;
        unsigned char   MPB2:1;
        unsigned char   MPBT2:1;
};
#define SSR2_BP (*(struct SSR2_S *)0xffff8c)

struct ADCSR_S{                         /* serial status register2 */
        unsigned char   ADF:1;
        unsigned char   ADIE:1;
        unsigned char   ADST:1;
        unsigned char   CKS:1;
        unsigned char   GRP:1;
        unsigned char   CH2:1;
        unsigned char   CH1:1;
        unsigned char   CH0:1;
};
#define ADCSR_BP (*(struct ADCSR_S *)0xffffa0)

struct MSTPCR_S{                        /* module stop mode */
        unsigned char   MSTP15:1;       /* DMA controller */
        unsigned char   MSTP14:1;       /* DTC */
        unsigned char   MSTP13:1;       /* TPU */
        unsigned char   MSTP12:1;       /* 8bit timer */
        unsigned char   MSTP11:1;       /* PPG */
        unsigned char   MSTP10:1;       /* D/A */
        unsigned char   MSTP9:1;        /* A/D */
        unsigned char   MSTP8:1;
        unsigned char   MSTP7:1;        /* SCI2 */
        unsigned char   MSTP6:1;        /* SCI1 */
        unsigned char   MSTP5:1;        /* SCI0 */
        unsigned char   MSTP4:1;
        unsigned char   MSTP3:1;
        unsigned char   MSTP2:1;
        unsigned char   MSTP1:1;
        unsigned char   MSTP0:1;
};
#define MSTPCR_BP (*(struct MSTPCR_S *)0xffff3c)
```

```
#define ISCRH   (*(volatile unsigned char *)0xffff2c)
#define ISCRL   (*(volatile unsigned char *)0xffff2d)
#define IER     (*(volatile unsigned char *)0xffff2e)
#define ISR     (*(volatile unsigned char *)0xffff2f)
#define IPRA    (*(volatile unsigned char *)0xfffec4)

#define ABWCR   (*(volatile unsigned char *)0xfffed0)
#define ASTCR   (*(volatile unsigned char *)0xfffed1)
#define WCRH    (*(volatile unsigned char *)0xfffed2)
#define WCRL    (*(volatile unsigned char *)0xfffed3)
#define BCRH    (*(volatile unsigned char *)0xfffed4)
#define BCRL    (*(volatile unsigned char *)0xfffed5)
#define MCR     (*(volatile unsigned char *)0xfffed6)
#define DRAMCR  (*(volatile unsigned char *)0xfffed7)

#define MAR0A_B (*(volatile unsigned char **)0xfffee0)
#define MAR0A_W (*(volatile unsigned short **)0xfffee0)
#define MAR0A_L (*(volatile unsigned long **)0xfffee0)
#define MAR0B_B (*(volatile unsigned char **)0xfffee8)
#define MAR0B_W (*(volatile unsigned short **)0xfffee8)
#define MAR0B_L (*(volatile unsigned long **)0xfffee8)

#define MAR0A   (*(volatile unsigned long *)0xfffee0)
#define IOAR0A  (*(volatile unsigned short *)0xfffee4)
#define ETCR0A  (*(volatile unsigned short *)0xfffee6)
#define MAR0B   (*(volatile unsigned long *)0xfffee8)
#define IOAR0B  (*(volatile unsigned short *)0xfffeec)
#define ETCR0B  (*(volatile unsigned short *)0xfffeee)

#define MAR1A   (*(volatile unsigned long *)0xfffef0)
#define IOAR1A  (*(volatile unsigned short *)0xfffef4)
#define ETCR1A  (*(volatile unsigned short *)0xfffef6)
#define MAR1B   (*(volatile unsigned long *)0xfffef8)
#define IOAR1B  (*(volatile unsigned short *)0xfffefc)
#define ETCR1B  (*(volatile unsigned short *)0xfffefe)

#define DMAWER  (*(volatile unsigned char *)0xffff00)
#define DMATCR  (*(volatile unsigned char *)0xffff01)
#define DMACR0A (*(volatile unsigned char *)0xffff02)
#define DMACR0B (*(volatile unsigned char *)0xffff03)
#define DMACR1A (*(volatile unsigned char *)0xffff04)
#define DMACR1B (*(volatile unsigned char *)0xffff05)

#define DMABCRH (*(volatile unsigned char *)0xffff06)
#define DMABCRL (*(volatile unsigned char *)0xffff07)

#define DTCERA  (*(volatile unsigned char *)0xffff30)
#define DTCERB  (*(volatile unsigned char *)0xffff31)
#define DTCERC  (*(volatile unsigned char *)0xffff32)
#define DTCERD  (*(volatile unsigned char *)0xffff33)
#define DTCERE  (*(volatile unsigned char *)0xffff34)
#define DTCERF  (*(volatile unsigned char *)0xffff35)
#define DTVECR  (*(volatile unsigned char *)0xffff37)
```

```
#define P1DDR   (*(volatile unsigned char *)0xfffeb0)
#define P1DR    (*(volatile unsigned char *)0xffff60)
#define P2DDR   (*(volatile unsigned char *)0xfffeb1)
#define P2DR    (*(volatile unsigned char *)0xffff61)
#define P3DDR   (*(volatile unsigned char *)0xfffeb2)
#define P3DR    (*(volatile unsigned char *)0xffff62)
#define P5DDR   (*(volatile unsigned char *)0xfffeb4)
#define P5DR    (*(volatile unsigned char *)0xffff64)
#define P6DDR   (*(volatile unsigned char *)0xfffeb5)
#define P6DR    (*(volatile unsigned char *)0xffff65)
#define PADDR   (*(volatile unsigned char *)0xfffeb9)
#define PADR    (*(volatile unsigned char *)0xffff69)
#define PBDDR   (*(volatile unsigned char *)0xfffeba)
#define PBDR    (*(volatile unsigned char *)0xffff6a)
#define PCDDR   (*(volatile unsigned char *)0xfffebb)
#define PCDR    (*(volatile unsigned char *)0xffff6b)
#define PDDDR   (*(volatile unsigned char *)0xfffebc)
#define PDDR    (*(volatile unsigned char *)0xffff6c)
#define PEDDR   (*(volatile unsigned char *)0xfffebd)
#define PEDR    (*(volatile unsigned char *)0xffff6d)
#define PFDDR   (*(volatile unsigned char *)0xfffebe)
#define PFDR    (*(volatile unsigned char *)0xffff6e)
#define PGDDR   (*(volatile unsigned char *)0xfffebf)
#define PGDR    (*(volatile unsigned char *)0xffff6f)

#define TPU_TCR0 (*(volatile unsigned char *)0xffffd0)
#define TMDR0   (*(volatile unsigned char *)0xffffd1)
#define TIOR0H  (*(volatile unsigned char *)0xffffd2)
#define TIOR0L  (*(volatile unsigned char *)0xffffd3)
#define TIER0   (*(volatile unsigned char *)0xffffd4)
#define TSR0    (*(volatile unsigned char *)0xffffd5)
#define TPU_TCNT0 (*(volatile unsigned short *)0xffffd6)
#define TGR0A   (*(volatile unsigned short *)0xffffd8)
#define TGR0B   (*(volatile unsigned short *)0xffffda)
#define TGR0C   (*(volatile unsigned short *)0xffffdc)
#define TGR0D   (*(volatile unsigned short *)0xffffde)

#define TPU_TCR1 (*(volatile unsigned char *)0xffffe0)
#define TMDR1   (*(volatile unsigned char *)0xffffe1)
#define TIOR1H  (*(volatile unsigned char *)0xffffe2)
#define TIER1   (*(volatile unsigned char *)0xffffe4)
#define TSR1    (*(volatile unsigned char *)0xffffe5)
#define TPU_TCNT1 (*(volatile unsigned short *)0xffffe6)
#define TGR1A   (*(volatile unsigned short *)0xffffe8)
#define TGR1B   (*(volatile unsigned short *)0xffffea)

#define TCR2    (*(volatile unsigned char *)0xfffff0)
#define TMDR2   (*(volatile unsigned char *)0xfffff1)
#define TIOR2H  (*(volatile unsigned char *)0xfffff2)
#define TIOR2L  (*(volatile unsigned char *)0xfffff3)
#define TIER2   (*(volatile unsigned char *)0xfffff4)
#define TSR2    (*(volatile unsigned char *)0xfffff5)
#define TCNT2   (*(volatile unsigned short *)0xfffff6)
#define TGR2A   (*(volatile unsigned short *)0xfffff8)
#define TGR2B   (*(volatile unsigned short *)0xfffffa)
```

```
#define TCR3    (*(volatile unsigned char *)0xfffe80)
#define TMDR3   (*(volatile unsigned char *)0xfffe81)
#define TIOR3H  (*(volatile unsigned char *)0xfffe82)
#define TIOR3L  (*(volatile unsigned char *)0xfffe83)
#define TIER3   (*(volatile unsigned char *)0xfffe84)
#define TSR3    (*(volatile unsigned char *)0xfffe85)
#define TCNT3   (*(volatile unsigned short *)0xfffe86)
#define TGR3A   (*(volatile unsigned short *)0xfffe88)
#define TGR3B   (*(volatile unsigned short *)0xfffe8a)
#define TGR3C   (*(volatile unsigned short *)0xfffe8c)
#define TGR3D   (*(volatile unsigned short *)0xfffe8e)

#define TCR4    (*(volatile unsigned char *)0xfffe90)
#define TMDR4   (*(volatile unsigned char *)0xfffe91)
#define TIOR4H  (*(volatile unsigned char *)0xfffe92)
#define TIOR4L  (*(volatile unsigned char *)0xfffe93)
#define TIER4   (*(volatile unsigned char *)0xfffe94)
#define TSR4    (*(volatile unsigned char *)0xfffe95)
#define TCNT4   (*(volatile unsigned short *)0xfffe96)
#define TGR4A   (*(volatile unsigned short *)0xfffe98)
#define TGR4B   (*(volatile unsigned short *)0xfffe9a)

#define TCR5    (*(volatile unsigned char *)0xfffea0)
#define TMDR5   (*(volatile unsigned char *)0xfffea1)
#define TIOR5H  (*(volatile unsigned char *)0xfffea2)
#define TIOR5L  (*(volatile unsigned char *)0xfffea3)
#define TIER5   (*(volatile unsigned char *)0xfffea4)
#define TSR5    (*(volatile unsigned char *)0xfffea5)
#define TCNT5   (*(volatile unsigned short *)0xfffea6)
#define TGR5A   (*(volatile unsigned short *)0xfffea8)
#define TGR5B   (*(volatile unsigned short *)0xfffeaa)

#define TSTR    (*(volatile unsigned char *)0xffffc0)
#define TSYR    (*(volatile unsigned char *)0xffffc1)


#define PCR     (*(volatile unsigned char *)0xffff46)
#define PMR     (*(volatile unsigned char *)0xffff47)
#define NDERH   (*(volatile unsigned char *)0xffff48)
#define NDERL   (*(volatile unsigned char *)0xffff49)
#define PODRH   (*(volatile unsigned char *)0xffff4A)
#define PODRL   (*(volatile unsigned char *)0xffff4B)
#define NDRH    (*(volatile unsigned char *)0xffff4C)
#define NDRL    (*(volatile unsigned char *)0xffff4D)
#define NDR3    (*(volatile unsigned char *)0xffff4C)
#define NDR2    (*(volatile unsigned char *)0xffff4E)
#define NDR1    (*(volatile unsigned char *)0xffff4D)
#define NDR0    (*(volatile unsigned char *)0xffff4F)


#define TCR0    (*(volatile unsigned char *)0xffffb0)
#define TCSR0   (*(volatile unsigned char *)0xffffb2)
#define TCORA0  (*(volatile unsigned char *)0xffffb4)
#define TCORB0  (*(volatile unsigned char *)0xffffb6)
#define TCNT0   (*(volatile unsigned char *)0xffffb8)

#define TCR1    (*(volatile unsigned char *)0xffffb1)
#define TCSR1   (*(volatile unsigned char *)0xffffb3)
#define TCORA1  (*(volatile unsigned char *)0xffffb5)
#define TCORB1  (*(volatile unsigned char *)0xffffb7)
#define TCNT1   (*(volatile unsigned char *)0xffffb9)
```

```
#define SMR0    (*(volatile unsigned char *)0xffff78)
#define BRR0    (*(volatile unsigned char *)0xffff79)
#define SCR0    (*(volatile unsigned char *)0xffff7a)
#define TDR0    (*(volatile unsigned char *)0xffff7b)
#define SSR0    (*(volatile unsigned char *)0xffff7c)
#define RDR0    (*(volatile unsigned char *)0xffff7d)
#define SCMR0   (*(volatile unsigned char *)0xffff7e)

#define SMR1    (*(volatile unsigned char *)0xffff80)
#define BRR1    (*(volatile unsigned char *)0xffff81)
#define SCR1    (*(volatile unsigned char *)0xffff82)
#define TDR1    (*(volatile unsigned char *)0xffff83)
#define SSR1    (*(volatile unsigned char *)0xffff84)
#define RDR1    (*(volatile unsigned char *)0xffff85)
#define SCMR1   (*(volatile unsigned char *)0xffff86)

#define SMR2    (*(volatile unsigned char *)0xffff88)
#define BRR2    (*(volatile unsigned char *)0xffff89)
#define SCR2    (*(volatile unsigned char *)0xffff8a)
#define TDR2    (*(volatile unsigned char *)0xffff8b)
#define SSR2    (*(volatile unsigned char *)0xffff8c)
#define RDR2    (*(volatile unsigned char *)0xffff8d)
#define SCMR2   (*(volatile unsigned char *)0xffff8e)


#define ADDRA   (*(volatile unsigned short *)0xffff90)
#define ADDRB   (*(volatile unsigned short *)0xffff92)
#define ADDRC   (*(volatile unsigned short *)0xffff94)
#define ADDRD   (*(volatile unsigned short *)0xffff96)
#define ADDRE   (*(volatile unsigned short *)0xffff98)
#define ADDRF   (*(volatile unsigned short *)0xffff9a)
#define ADDRG   (*(volatile unsigned short *)0xffff9c)
#define ADDRH   (*(volatile unsigned short *)0xffff9e)
#define ADCSR   (*(volatile unsigned char *)0xffffa0)
#define ADCR    (*(volatile unsigned char *)0xffffa1)


#define DADR0   (*(volatile unsigned char *)0xffffa4)
#define DADR1   (*(volatile unsigned char *)0xffffa5)
#define DACR    (*(volatile unsigned char *)0xffffa6)


#define MSTPCR  (*(volatile unsigned short *)0xffff3c)
```

H8S/2655シリーズ 内蔵I/O編 アプリケーションノート

発行年月　平成8年9月　第1版
発　　行　株式会社 日立製作所
　　半導体グループ電子統括営業本部
編　　集　株式会社 超Lメディア
　　技術ドキュメントグループ

株式会社 日立製作所

半導体グループ	〒100-0004	東京都千代田区大手町二丁目6番2号 (日本ビル)	(03) 3270-2111 (大代)
北海道支社	〒060-0002	札幌市中央区北二条西四丁目1番地 (札幌三井ビル)	(011) 261-3131 (大代)
北見営業所	〒090-0833	北見市とん田東町603番地10	(0157) 22-7121
道北営業所	〒070-0035	旭川市5条通九丁目左1号 (安田生命旭川ビル)	(0166) 24-3567
道東営業所	〒085-0015	釧路市北大通十丁目1番地 (北銀住生ビル)	(0154) 23-2551
帯広営業所	〒080-0016	帯広市西6条南六丁目3番地 (ソネビル)	(0155) 24-0818
室蘭営業所	〒050-0074	室蘭市中島町四丁目9番6号 (日協産業ビル)	(0143) 44-3327
函館営業所	〒040-0001	函館市五稜郭町35番1号 (日産火災函館ビル)	(0138) 52-6072
東北支社	〒980-8531	仙台市青葉区一番町二丁目4番1号 (興和ビル)	(022) 223-0121 (大代)
青森営業所	〒030-0801	青森市新町二丁目2番4号 (青森新町第一生命ビル)	(0177) 75-1371
盛岡営業所	〒020-0021	盛岡市中央通二丁目1番21号 (安田生命盛岡ビル)	(019) 624-0056
秋田営業所	〒010-0975	秋田市八橋字戌川原64番地2 (秋田県JAビル)	(018) 864-2234
山形営業所	〒990-0039	山形市香澄町二丁目2番36号 (山形センタービル)	(023) 623-5333
福島営業所	〒960-8041	福島市大町5番6号 (日生福島ビル)	(024) 523-0241～3
郡山営業所	〒963-8005	郡山市清水台二丁目13-23 (郡山第一ビル)	(024) 923-3944
電機システム統括営業本部	〒101-8010	東京都千代田区神田駿河台四丁目6番地 (日立本社ビル)	(03) 3258-1111 (大代)
新潟支店	〒950-0087	新潟市東大通一丁目4番1号 (マルタケビル9階)	(025) 241-8161 (代)
半導体グループ電子統括営業本部	〒100-0004	東京都千代田区大手町二丁目6番2号 (日本ビル)	(03) 3270-2111 (大代)
茨城営業所	〒312-0034	ひたちなか市堀口832番地2 (日立システムプラザ勝田1階)	(029) 271-9411(代)
松本営業所	〒390-0815	松本市深志一丁目2番11号 (昭和ビル7階)	(0263) 36-6632
高崎営業所	〒370-0841	高崎市栄町16番11号 (高崎イーストタワービル11階)	(027) 325-2161
横浜支社	〒220-0011	横浜市西区高島二丁目6番32号 (日産横浜ビル)	(045) 451-5000 (代)
県央支店	〒243-0018	厚木市中町三丁目16番1号 (TYG第11中町ビル)	(0462) 96-6800 (代)
川崎営業所	〒210-0006	川崎市川崎区砂子二丁目4番地10号 (富士川崎ビル6階)	(044) 246-1501 (代)
沼津営業所	〒410-0801	沼津市大手町五丁目6番7号 (ヌマヅスルガビル4階)	(0559) 51-3530 (代)
北陸支社	〒930-0004	富山市桜橋通り5番13号 (富山興銀ビル)	(0764) 33-8511 (大代)
金沢支店	〒920-0853	金沢市本町二丁目15番1号 (ポルテ金沢)	(076) 263-2351 (ダイヤル)
福井営業所	〒910-0006	福井市中央一丁目3番12号 (ユアーズ大手ビル8階)	(0776) 23-8378 (代)
中部支社	〒460-8435	名古屋市中区栄三丁目17番12号 (大津通電気ビル)	(052) 243-3111 (代)
静岡支店	〒420-0859	静岡市栄町3番地の9 (朝日生命静岡ビル)	(054) 254-7341 (代)
豊田支店	〒471-0842	豊田市土橋町四丁目67番地の2 (豊田日立ビル)	(0565) 29-1031 (代)
岐阜営業所	〒500-8844	岐阜市吉野町六丁目16番地17号 (大同生命ビル)	(0582) 63-0834
三重営業所	〒510-0067	四日市市浜田町5番27号 (第三加藤ビル)	(0593) 52-7111 (代)
関西支社	〒559-8515	大阪市住之江区南港東八丁目3番45号 (日立関西ビル)	(06) 6616-1111 (大代)
京都支店	〒600-8008	京都市下京区四条通烏丸東入ル長刀鉾町20番地(四条烏丸FTスクエア4階)	(075) 223-5611 (代)
奈良営業所	〒630-8115	奈良市大宮町五丁目3番14号 (不動ビル)	(0742) 36-2321 (代)
和歌山営業所	〒640-8106	和歌山市三木町中ノ丁15 (和歌山フコク生命ビル)	(0734) 33-1258 (代)
神戸支店	〒651-0096	神戸市中央区雲井通四丁目2番2号 (神戸いすゞリクルートビル)	(078) 261-9677 (代)
中国支社	〒730-0011	広島市中区基町11番10号 (千代田生命ビル)	(082) 223-4111 (代)
鳥取営業所	〒680-0822	鳥取市今町二丁目251番地 (日本生命鳥取駅前ビル)	(0857) 22-4270 (代)
山陰営業所	〒690-0003	松江市朝日町498番地6 (松江駅前第一生命ビル)	(0852) 26-7366 (代)
岡山支店	〒700-0907	岡山市下石井一丁目1番3号 (日本生命岡山第二ビル)	(086) 224-5271 (代)
福山営業所	〒720-0043	福山市船町7番23号 (安田生命福山ビル)	(0849) 24-6738 (代)
山口支店	〒754-0014	山口県吉敷郡小郡町高砂町1番8号 (安田生命小郡ビル)	(0839) 72-3039 (代)
徳山営業所	〒745-0073	徳山市代々木通一丁目4番1号 (三井生命徳山ビル)	(0834) 31-1515 (代)
宇部営業所	〒755-0043	宇部市相生町8番1号 (宇部興産ビル)	(0836) 31-3610 (代)
四国支社	〒760-0007	高松市中央町5番31号 (中央町ビル)	(087) 831-2111 (代)
愛媛支店	〒790-0003	松山市三番町四丁目4番6号 (松山第二東邦生命ビル)	(089) 943-1333 (代)
徳島営業所	〒770-0841	徳島市八百屋町三丁目15番地 (徳島あおば生命ビル6階)	(088) 654-5535 (代)
高知営業所	〒780-0870	高知市本町二丁目1番10号 (安田生命高知ビル)	(0888) 24-0511 (代)
九州支社	〒814-8577	福岡市早良区百道浜二丁目1番1号 (日立九州ビル)	(092) 852-1111 (代)
北九州支店	〒802-0081	北九州市小倉北区紺屋町12番23号 (小倉あおば生命ビル6階)	(093) 533-5500 (代)
佐賀営業所	〒840-0801	佐賀市駅前中央一丁目9番45号 (三井生命佐賀駅前ビル3階)	(0952) 29-7981 (代)
長崎営業所	〒850-0033	長崎市万才町6番34号 (日産・時事長崎ビル3階)	(095) 821-6313 (代)
熊本支店	〒860-0802	熊本市中央街2番11号 (熊本サンニッセイビル2階)	(096) 359-7070 (代)
大分営業所	〒870-0044	大分市舞鶴町一丁目4番35 (大分三井ビル2階)	(097) 534-0860 (代)
宮崎営業所	〒880-0805	宮崎市橘通東四丁目7番28号 (宮崎第一生命ビル3階)	(0985) 29-1721 (代)
鹿児島営業所	〒890-0053	鹿児島市中央町12番2号 (明治生命西鹿児島ビル2階)	(099)256-9021 (代)
沖縄支店	〒900-0032	那覇市松山一丁目1番14号 (千代田生命那覇共同ビル6階)	(098) 868-8176 (代)

■ 資料のご請求は、上記の担当営業または下記へどうぞ。
株式会社 日立製作所　半導体グループ　電子統括営業本部　半導体ドキュメント管理室
〒100-0004　東京都千代田区大手町二丁目6番2号 (日本ビル)　電話 (03) 5201-5189 (直)　FAX (03) 3270-3277

● 製品仕様は、改良のため変更することがあります。
(株)日立製作所　半導体グループのWWWにおいて、製品情報がより豊富にお届けできるようになりました。ぜひご覧ください。
http://www.hitachi.co.jp/Sicd/

H8S/2655 シリーズ
アプリケーションノート　ー内蔵 I/O 編ー

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-502-060