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

SH7040 シリーズ　内蔵I/O 編

アプリケーションマニュアル

ルネサスSuperH RISC engine

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　1997.05

はじめに

ＳＨ７０４０、ＳＨ７０４１、ＳＨ７０４２、ＳＨ７０４３は、内部３２ビット構成のＲＩＳＣ（Reduced Instruction Set Computer）タイプの命令セットを持ったＳＨ－２ＣＰＵを核に、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、ＲＯＭ、１６ビットマルチファンクションタイマパルスユニット（MTU）、シリアルコミュニケーションインタフェース（ＳＣＩ）、ポートアウトプットイネーブル（ＰＯＥ）、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、小規模システムから大規模システムまで幅広い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、ＳＨ７０４０シリーズの内蔵周辺機能を使用したタスク例について述べており、ユーザにてソフトウェア設計の際、ご参考として役立てていただけるようにまとめたものです。

なお、本アプリケーションノートに掲載されている各タスクのプログラム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には、改めて動作確認の上ご使用くださいますようにお願い致します。

目　次

1. ＳＨ７０４０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、ＳＨ７０４０シリーズの各周辺機能の動作例を簡単なタスク例を基に説明したものです。

１．１　内蔵Ｉ／Ｏ編構成

内蔵Ｉ／Ｏ編は図１．１に示す構成で周辺機能の使用方法について説明しています。

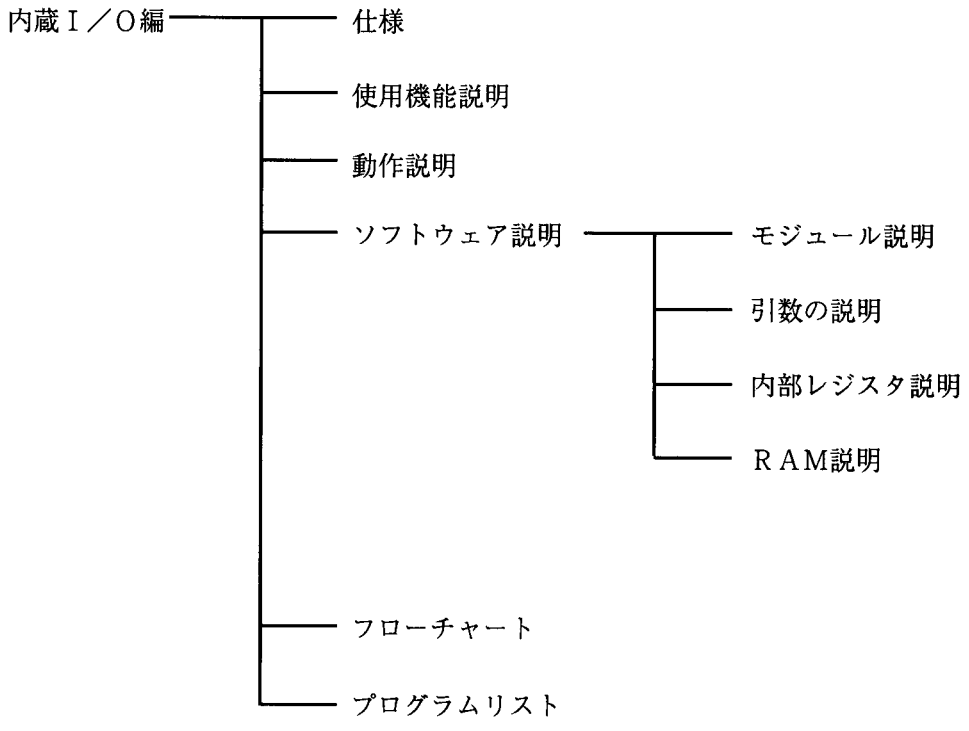


図１．１　内蔵Ｉ／Ｏ編構成

（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ます。

（5）フローチャート
タスク例を実行するソフトウェアについてゼネラルフローチャートを使用し説明します。

（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ます。

2. SH7040内蔵I／O編

目　次

2.1　パルスのHighおよびLow幅測定	MCU	SH7040	使用機能	MTU（インプットキャプチャ）

仕様

（1）図2.1.1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に格納します。
（2）28.7MHz動作時、パルスのHigh幅およびLow幅は34.8nsから2.28msまで34.8ｎｓ単位で測定可能です。

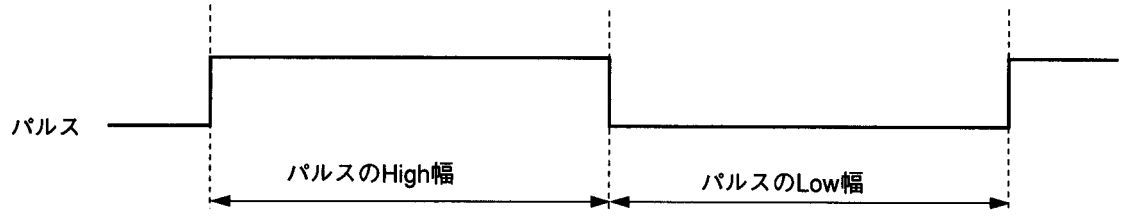


図2.1.1　パルス幅測定タイミング

使用機能説明

（1）本タスク例では、ch0を使用してパルスのHigh幅およびLow幅を測定します。
（a）図2.1.2にch0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エッジ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。（カウンタクリア）
・パルスの立ち上がりエッジ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。

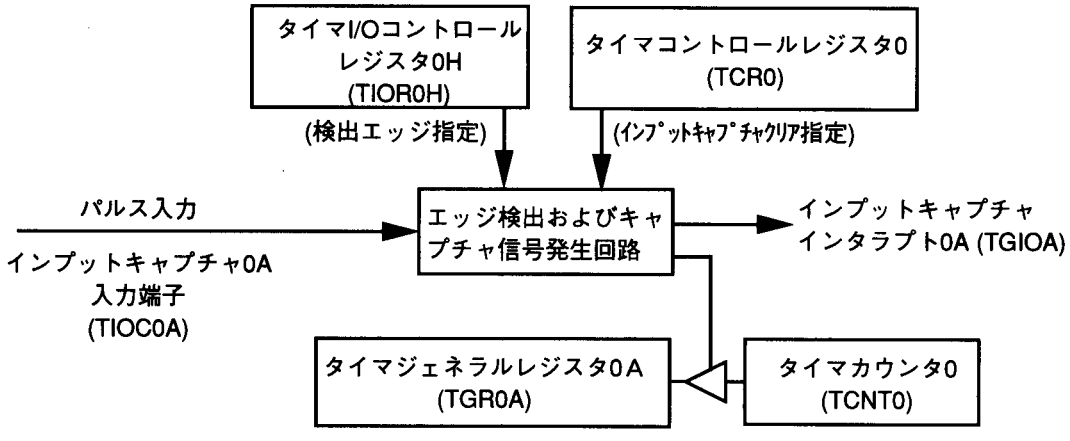


図2.1.2　MTU/ch0　ブロック図

パルスのHighおよびLow幅測定	MCU	SH7040	使用機能	MTU（インプットキャプチャ）

使用機能説明

（2）表2.1.1に本タスク例の機能割り付けを示します。表に示すようにMTUの機能を割り付け、パルスのHighおよびLow幅を測定しています。

表2.1.1　機能割り付け

端子、レジスタ名	機能割り付け
TCR0	カウンタクリア要因の選択
TIOR0H	インプットキャプチャ信号の入力エッジを選択
TIOC0A	測定するパルスを入力
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
TGI0A	パルスの立ち上がりおよび立ち下がりで、パルスのHighおよびLow幅測定を起動

動作説明

図2.1.3に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理によりパルスのHighおよびLow幅を測定します。

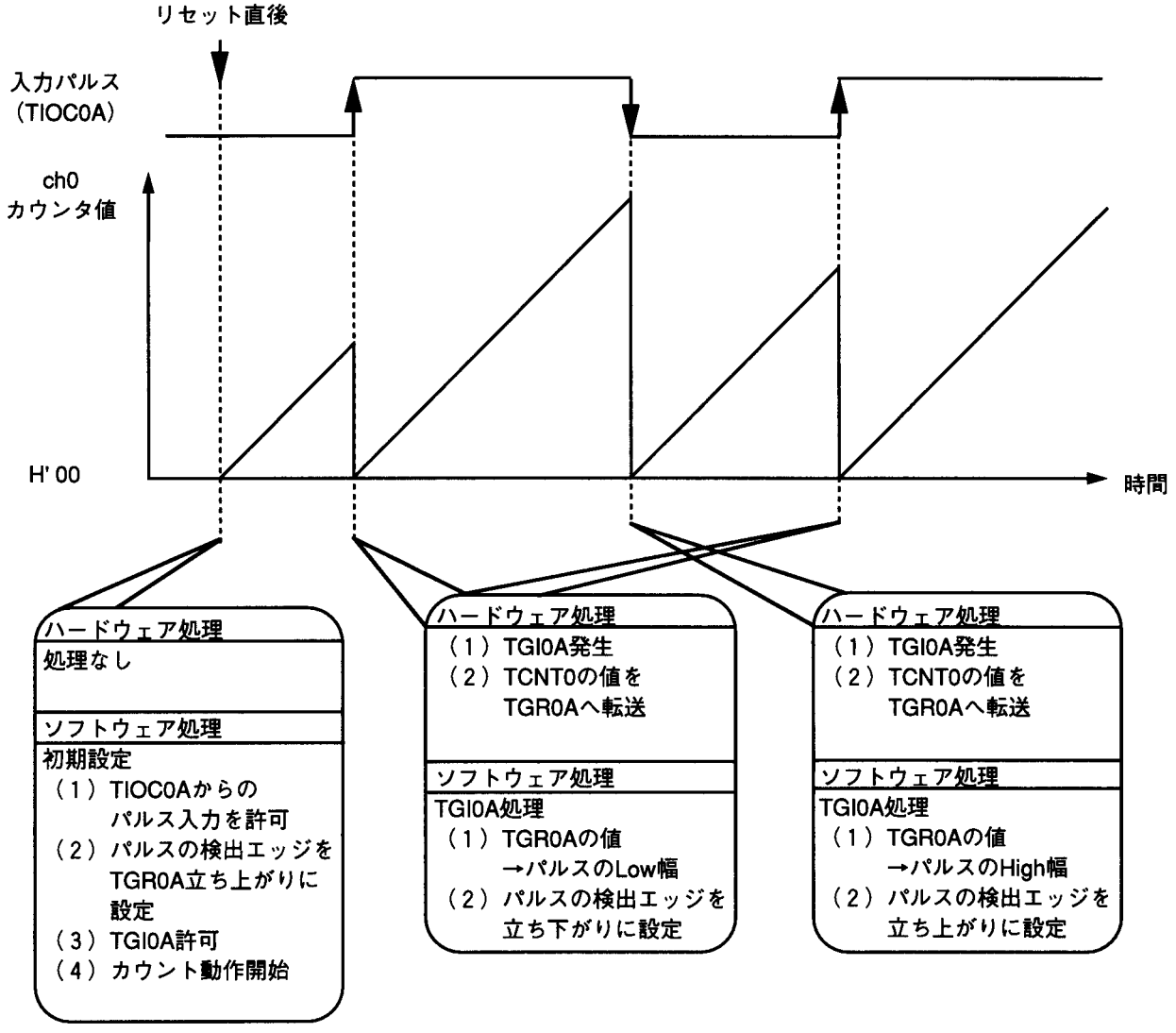


図2.1.3　パルス幅測定動作原理

パルスのHighおよびLow幅測定	MCU	SH7040	使用機能	MTU（インプットキャプチャ）

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	機能割り付け
メインルーチン	pwhlmn	MTUの初期設定
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

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	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 パルスのHigh幅は以下の式にて算出 ﾊﾟﾙｽのHigh幅(ns)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8ns)	1ワード	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 パルスのLow幅は以下の式にて算出 ﾊﾟﾙｽのLow幅(ns)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8ns)	1ワード		

（3）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
PFCE.PECR2	TIOC0Aからのパルス入力の許可	ﾒｲﾝﾙｰﾁﾝ
T0.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	ﾒｲﾝﾙｰﾁﾝ、ﾊﾟﾙｽのHighおよびLow幅測定
T0.TIOR0H	パルスの立ち上がり検出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	ﾒｲﾝﾙｰﾁﾝ
T0.TIER0	TGI0Aによる割り込みの許可	
T0.TGRA0	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、この値からパルスの周期を算出	ﾊﾟﾙｽのHighおよびLow幅測定
T0.TSR0	インプットキャプチャAの発生状況	
INTC.IPRD	TGI0Aの割り込み優先レベルを15に設定	ﾒｲﾝﾙｰﾁﾝ
TMRSH.TSTR	タイマカウンタの動作／禁止を設定	

（4）使用RAM説明

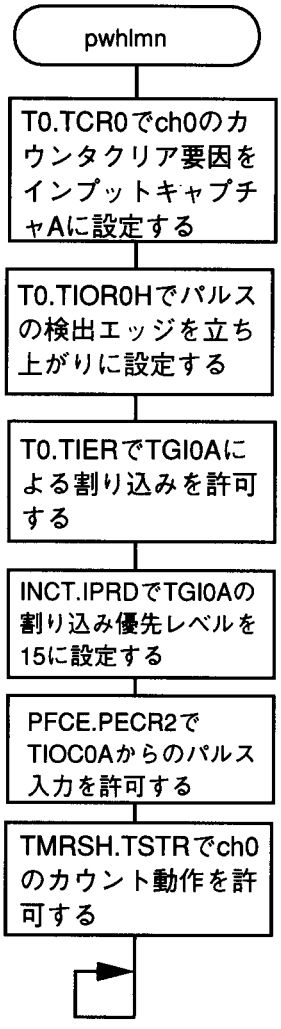
本タスク例では引数以外のRAMは使用していません。

注）レジスタのラベル名はSH7040ヘッダファイルの名前を使用しています。

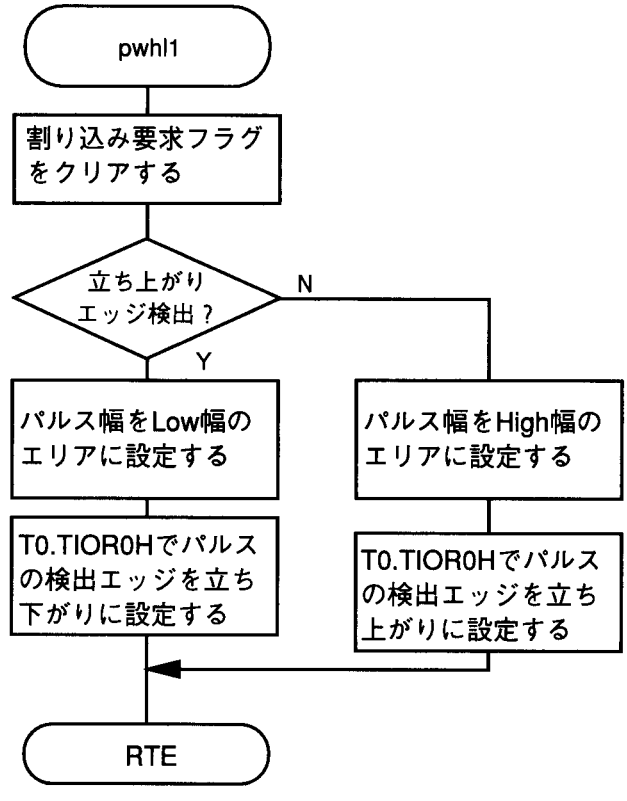
パルスのHighおよびLow幅測定	MCU	SH7040	使用機能	MTU（インプットキャプチャ）

フローチャート

（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	SH7040	使用機能	MTU（インプットキャプチャ）

プログラムリスト

```
/*------------------------------------------------------------------------------*/
/*                                INCLUDE FILE                                  */
/*------------------------------------------------------------------------------*/
#include <machine.h>
#include "SH7040.H"
/*------------------------------------------------------------------------------*/
/*                                PROTOTYPE                                     */
/*------------------------------------------------------------------------------*/
void pwhlmn(void);
#pragma interrupt(pwhl1)
/*------------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                                  */
/*------------------------------------------------------------------------------*/
#define pwh_hdata       (*(unsigned short *)0xfffff000)
#define pwh_ldata       (*(unsigned short *)0xfffff002)
/*------------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                                    */
/*------------------------------------------------------------------------------ -*/
void pwhlmn(void)
{
    T0.TCR0 = 0x20;                 /* timer clear input caputure TGRA0 */
    T0.TIOR0H = 0x08;               /* input caputure rising edge TIOC0A */
    T0.TIER0 = 0x01;                /* initialize TGI0A0*/
    INTC.IPRD = 0xf000;             /* set initialize level=15 */
    set_imask(0x0);                 /* set imask level=0 */
    PFCE.PECR2 = 0X0001;            /* TIOCnx sellect */
    TMRSH.TSTR = 0x01;              /* start timer0 */
    while(1);                       /* loop */
}


void pwhl1()
{
    T0.TSR0 &=0xfe;                 /* clear flag */
    if(( T0.TIOR0H & 0x0f ) == 0x08)    /* rising edge ? */
    {
    pwh_hdata = T0.TGR0A;           /* set pwh */
    T0.TIOR0H |= 0x01;              /* input caputure falling edga TIOC0A */
    }
    else
    {
    pwh_ldata = T0.TGR0A;           /* set pwl */
    T0.TIOR0H &=0xfe;               /* input caputure rising edge TIOC0A */
    }
}
```

2.2　パルス出力	MCU	SH7040	使用機能	MTU(アウトプットコンペアマッチ)

仕様

（1）MTUのch0を使用して、図2.2.1に示すように設定された周期のデューティ50%のパルスを出力します。
（2）28.7MHzで動作時、出力するパルスの周期は69.6nSから2.28mSの間で任意に設定できます。

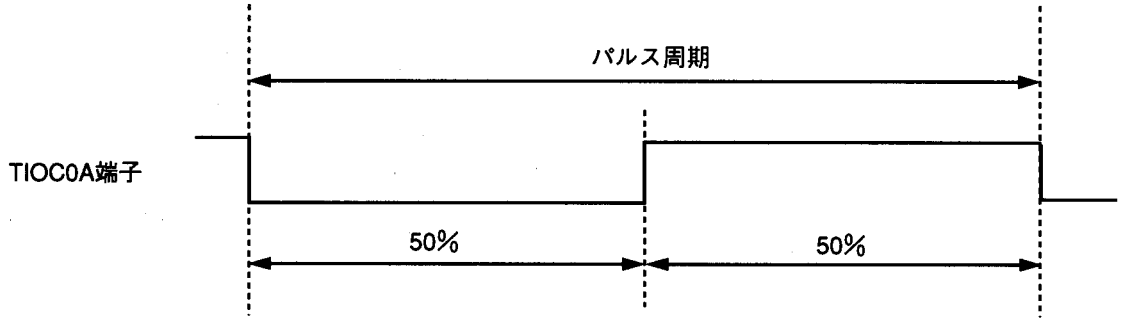


図2.2.1　パルス出力例

使用機能説明

（1）本タスク例ではMTUのch0を使用し、デューティ50%のパルスを出力します。
（a）図2.2.2に本タスク例で使用するMTU/ch0のブロック図を示します。
ch0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）

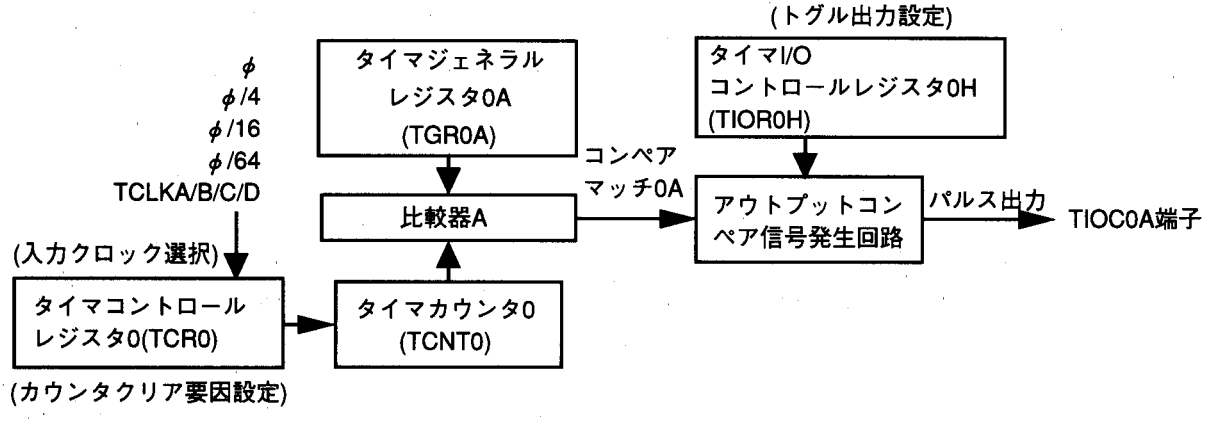


図2.2.2　MTU/ch0　ブロック図

パルス出力	MCU	SH7040	使用機能	MTU(アウトプットコンペアマッチ)

使用機能説明

（2）表2.2.1に本タスク例の機能割り付けを示します。表に示すようにMTUの機能を割り付け、パルスを出力します。

表2.2.1　機能割り付け

端子、レジスタ名	機能割り付け
TIOC0A	パルスの出力端子
TCR0	カウンタクリア要因および入力クロックの選択
TIOR0H	パルスの出力レベルを設定
TGR0A	パルスの1/2周期を設定

動作説明

図2.2.3に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理によりパルスを出力します。

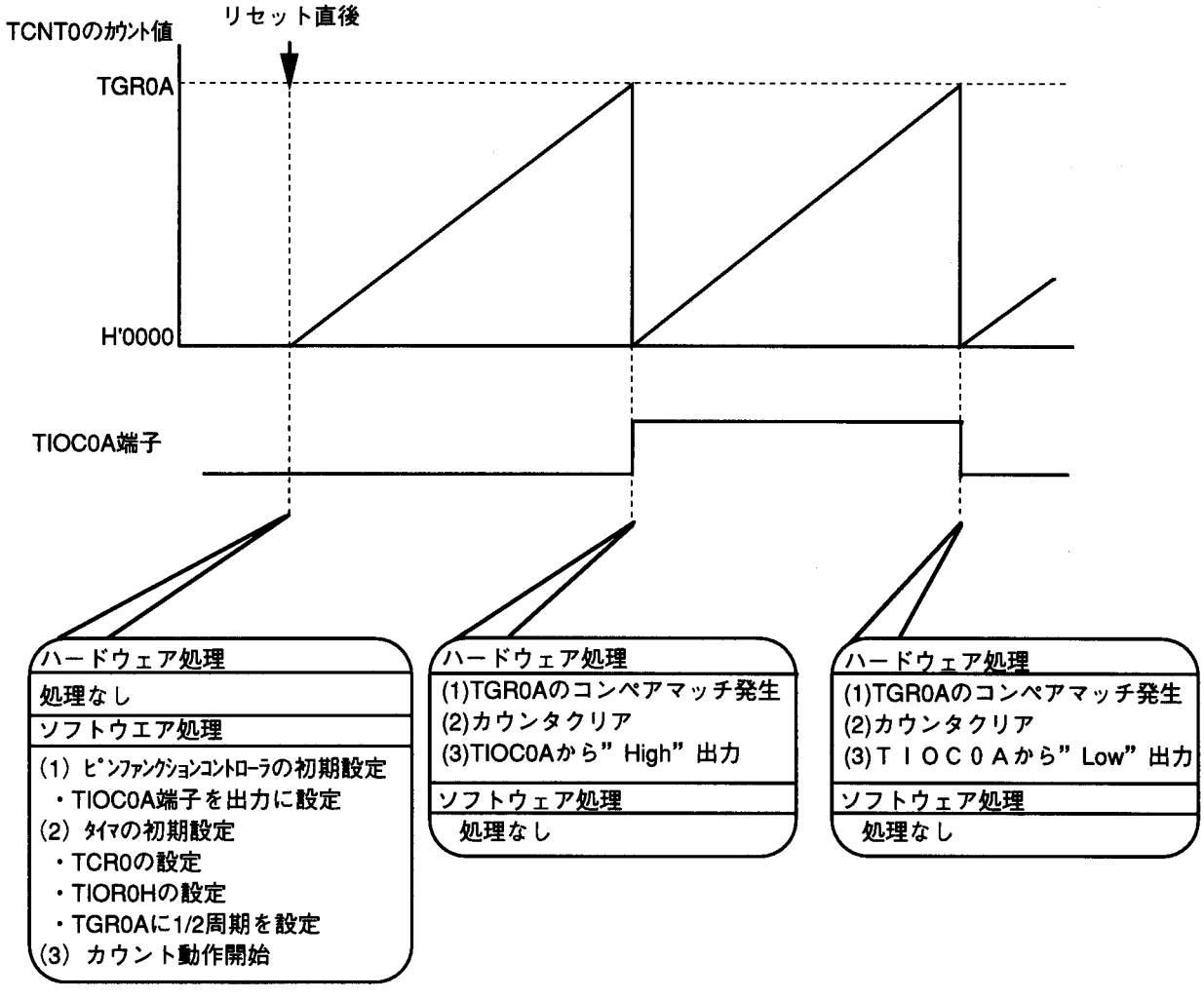


図2.2.3　パルス出力動作原理

パルス出力	MCU	SH7040	使用機能	MTU(アウトプットコンペアマッチ)

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	機能割り付け
メインルーチン	puls_out	PFCおよびパルス出力設定

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
pul_cyc	パルスの1/2周期に相当するタイマ値を設定 パルスの周期は以下の式にて算出 ﾊﾟﾙｽ周期(nS)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8nS)	1ワード	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	機能割り付け	使用モジュール名
PFCE.PECR2	マルチプレクス端子をTIOC0A出力に設定	メインルーチン
TMRSH.TSTR	ch0のタイマカウンタの動作／禁止を設定	
T0.TCR0	カウンタクリア要因をTGR0Aのコンペアマッチに設定し、 入力クロックはφを設定	
T0.TIOR0H	TIOC0Aは初期出力0、アウトプットコンペアでトグル出力	
T0.TGR0A	出力パルスの1/2周期を設定	
T0.TMDR0	ch0は通常モードに設定	

（4）使用ＲＡＭ説明

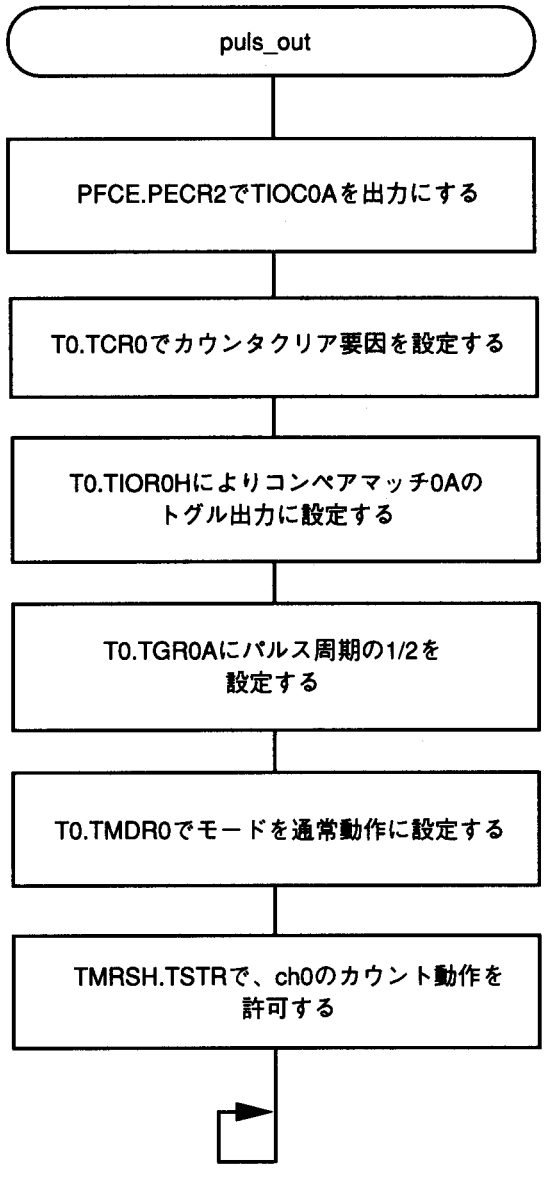
本アプリケーション例では引数以外のRAMは使用していません。

注）レジスタのラベル名は、SH7040ヘッダファイルの名前を使用しています。

パルス出力	MCU	SH7040	使用機能	MTU(アウトプットコンペアマッチ)

フローチャート

（1）メインルーチン



パルス出力	MCU	SH7040	使用機能	MTU(アウトプットコンペアマッチ)

プログラムリスト

```
/*------------------------------------------------------------*/
/*                        INCLUDE FILE                        */
/*------------------------------------------------------------*/
#include<machine.h>
#include"SH7040.H"
/*------------------------------------------------------------*/
/*                        PROTOTYPE                           */
/*------------------------------------------------------------*/
void puls_out(void);
/*------------------------------------------------------------*/
/*                      RAM ALLOCATION                        */
/*------------------------------------------------------------*/
#define pul_cyc          (*(unsigned short *)0xfffff000)
/*------------------------------------------------------------*/
/*                      MAIN PROGRAM                          */
/*------------------------------------------------------------*/
void puls_out(void)
     {
     PFCE.PEIOR = 0x0001;        /* TIOC0A output */
     PFCE.PECR2 = 0x0001;        /* TIOC0A outputcompare/match output */

     TO.TCR0 = 0x20;             /* compare/match clear of TGR0A */
     TO.TIOR0H = 0x03;           /* start'0' compare/match toggle output */
     TO.TGR0A = pul_cyc;         /* 1/2 sycle */
     TO.TCNT0 = 0x0000;          /* timer start value set */
     TO.TMDR0 = 0xc0;            /* mode set */
     TMRSH.TSTR = 0x01;          /* timer count start */
     while(1);
     }
```

2.3 PWM４相出力	MCU	SH７０４０	使用機能	MTU（PWMﾓｰﾄﾞ1）

仕様

（１）MTUのPWMモード1を使用して、設定されたデューティ値及び周期を基に4相のPWM出力をします。

（２）PWMモード1は各チャネルで任意の周期が設定できます。ch0、3、4は各2本づつ、ch1、2は各1本づつの出力が可能です。よってch0、3、4では、同一周期の中でHigh幅の異なる波形の生成が可能です。

（３）デューティは、0%～100%まで1/65535の分解能で、設定できます。

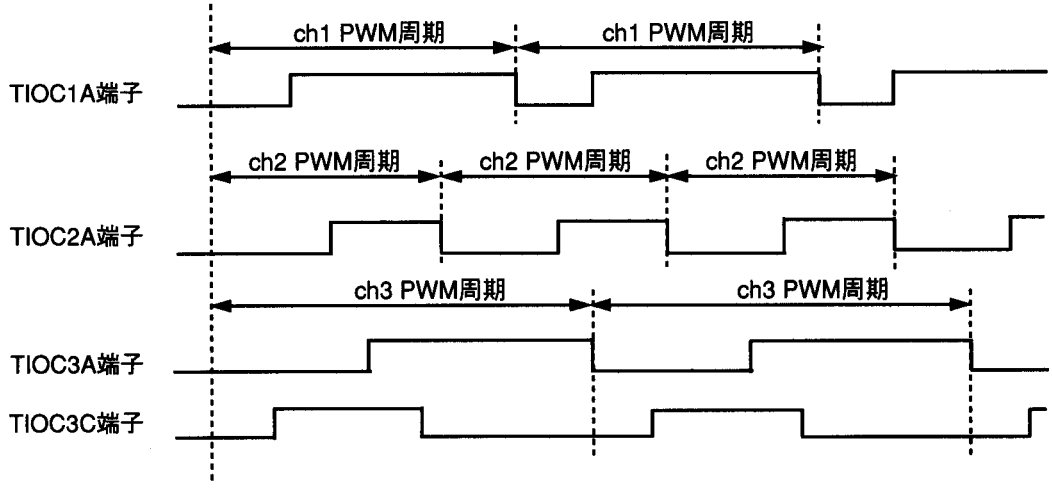


図2.3.1　PWM出力例

使用機能説明

（1）本タスク例ではMTUのch1～3を使用し、4相のPWM出力をします。

・PWMモード1では、TGRAとTGRBレジスタ、TGRCとTGRDレジスタをそれぞれペアで使用して、PWM出力を生成します。ch0～4まで使用することで、最大8相 のPWM出力が可能です。

（a）図2.3.2に本タスク例で使用するMTUのブロック図を示します。

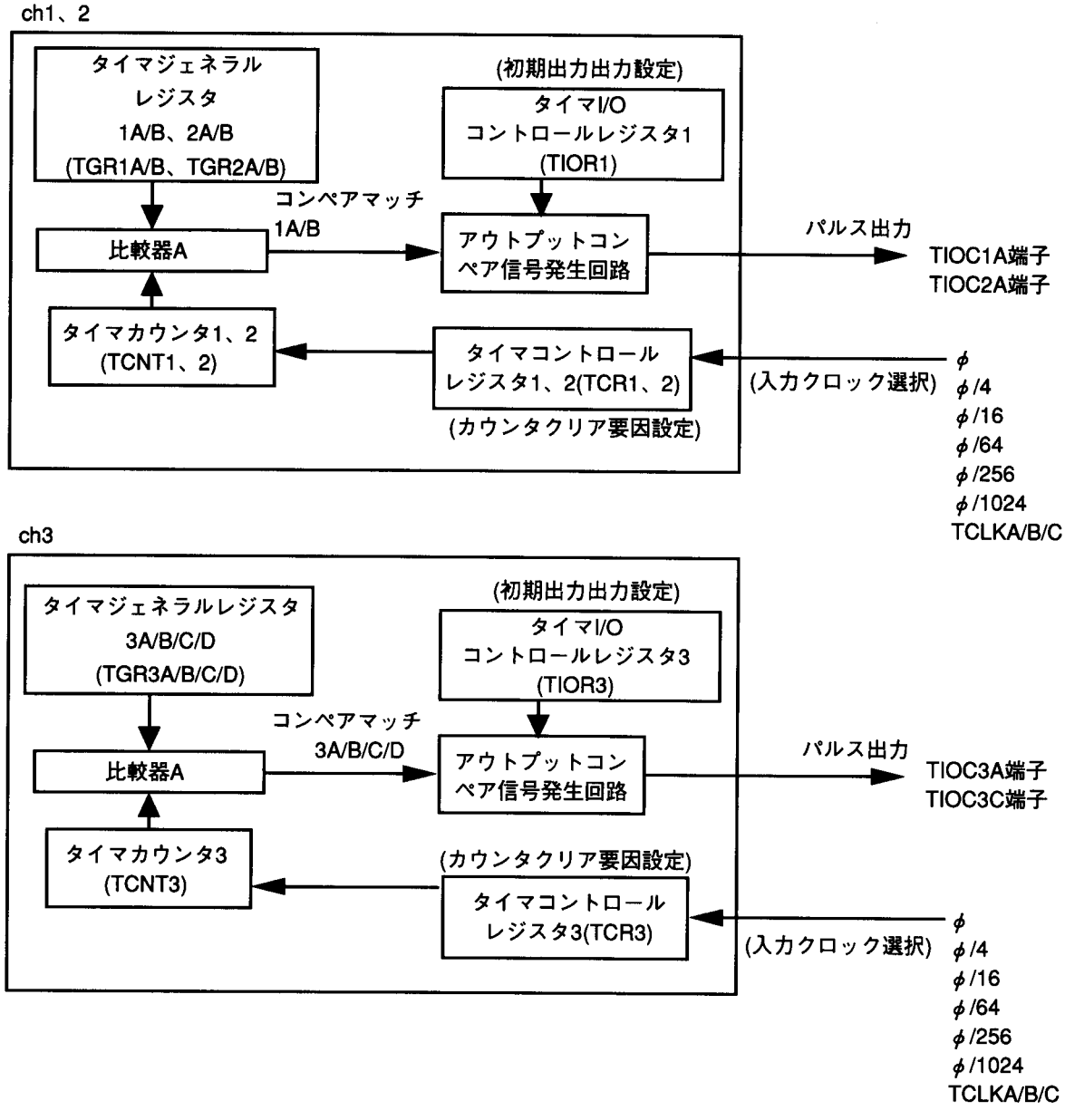


図2.3.2　MTU/ch1、2、3ブロック図

（２）表2.3.1に本タスクの機能割り付けを示します。表に示すようにMTUの機能を割り付け、PWMパルスを出力します。

表2.3.1　機能割り付け

端子、レジスタ名	機能割り付け
TIOC1A TIOC2A TIOC3A TIOC3C	PWMパルス出力端子
TCR1 TCR2 TCR3	ch1～ch3のタイマカウンタのクリア要因と入力クロックを選択
TMDR1 TMDR2 TMDR3	ch1～ch3をPWMﾓｰﾄﾞ1として動作
TGR1A TGR2A TGR3A	PWM周期の設定
TGR1B TGR2B TGR3B TGR3C TGR3D	デューティ値の設定

PWM４相出力	MCU	SH７０４０	使用機能	MTU（PWMﾓｰﾄﾞ1）

動作説明

図2.3.3に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理により、ch1～ch3の各PWM出力端子(TIOC1A、TIOC2A、TIOC3A/C)から4相のPWM出力をします。

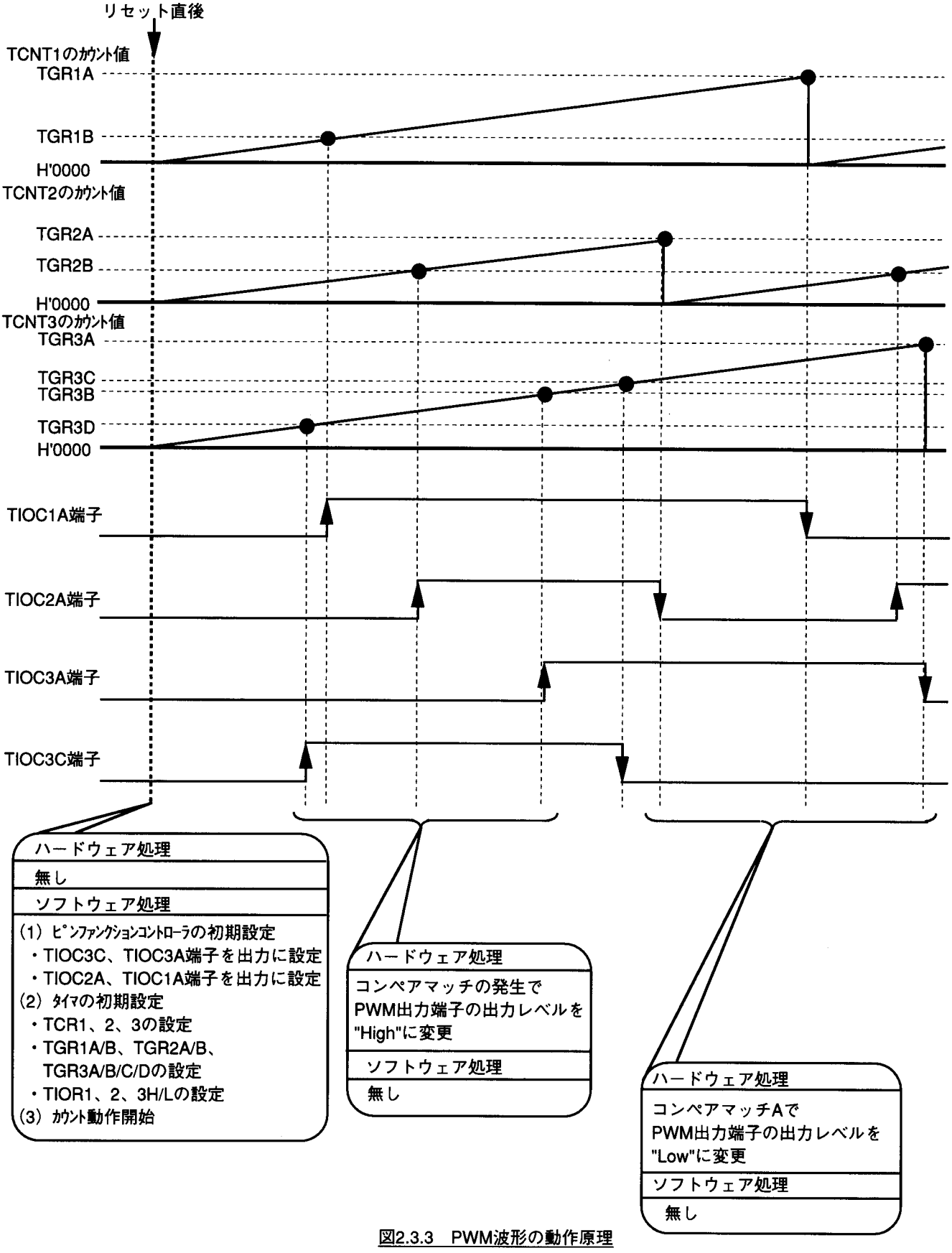


図2.3.3　PWM波形の動作原理

PWM４相出力	MCU	SH７０４０	使用機能	MTU（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	機能割り付け
メインルーチン	pwm_1	PFCおよびPWM出力の設定

（２）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
pul_cyc1 pul_cyc2 pul_cyc3	パルスの周期に相当するタイマ値を設定 パルスの周期は以下の式にて算出 ﾊﾟﾙｽ周期(nS)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8nS)	1ワード	メインルーチン	入力
pul_duty1b pul_duty2b pul_duty3b pul_duty3c pul_duty3d	TIOC端子から出力される波形変化タイミングを設定			

（３）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
TMRSH.TSTR	チャネル1、2、3のタイマカウントのスタートを行う	メインルーチン
T1.TCR1 T2.TCR2 T3.TCR3	タイマカウンタのクリア要因をTGR1A、TGR2A、TGR3Aの コンペアマッチでクリア 入力クロックはφを選択	
T1.TGR1A	チャネル1のPWM周期を設定	
T1.TGR1B	TIOC1Aから"High"出力させるタイマカウンタ値を設定	
T2.TGR2A	チャネル2のPWM周期を設定	
T2.TGR2B	TIOC2Aから"High"出力させるタイマカウンタ値を設定	
T3.TGR3A	チャネル3のPWM周期を設定	
T3.TGR3B	TIOC3Aから"High"出力させるタイマカウンタ値を設定	
T3.TGR3C	TIOC3Cから"Low"出力させるタイマカウンタ値を設定	
T3.TGR3D	TIOC3Cから"High"出力させるタイマカウンタ値を設定	
T1.TIOR1	TGR1Aは初期出力0でアウトプットコンペアで0出力、 TGR1Bは初期出力0でアウトプットコンペアで1出力に設定	
T2.TIOR2	TGR2Aは初期出力0でアウトプットコンペアで0出力、 TGR2Bは初期出力0でアウトプットコンペアで1出力に設定	
T3.TIOR3H	TGR3Aは初期出力0でアウトプットコンペアで0出力、 TGR3Bは初期出力0でアウトプットコンペアで1出力に設定	
T3.TIOR3L	TGR3Cは初期出力0でアウトプットコンペアで0出力、 TGR3Dは初期出力0でアウトプットコンペアで1出力に設定	
T1.TMDR1 T2.TMDR2 T3.TMDR3	動作モードはPWMモード1に設定	

（４）使用ＲＡＭ説明

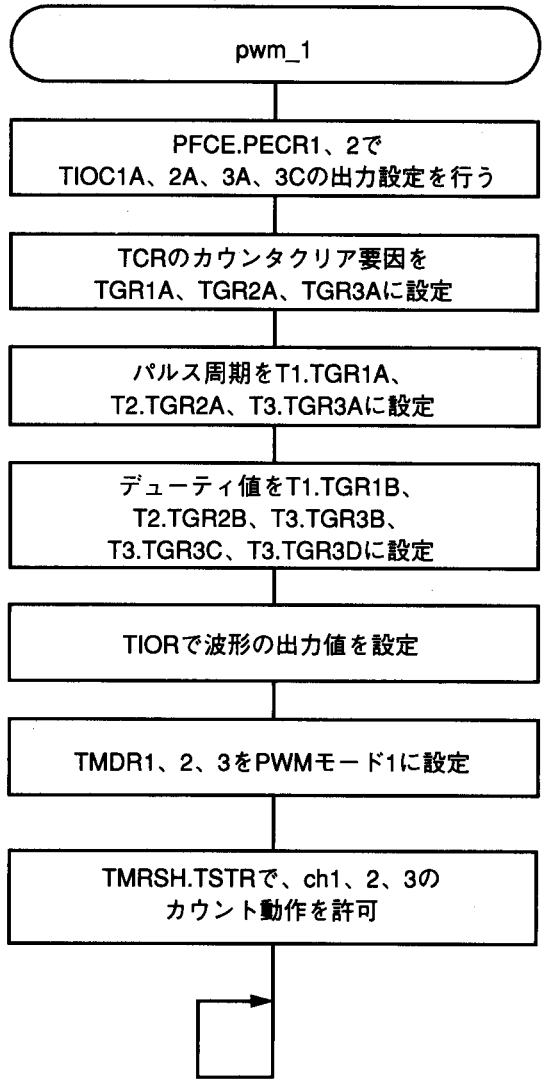
本アプリケーション例では引数以外のRAMは使用していません。

注）レジスタのラベル名は、SH7040ヘッダファイルの名前を使用しています。

PWM４相出力	MCU	SH７０４０	使用機能	MTU（PWMﾓｰﾄﾞ1）

フローチャート

（１）メインルーチン



プログラムリスト

```
/*-------------------------------------------------------------------------*/
/*                              INCLUDE FILE                               */
/*-------------------------------------------------------------------------*/
#include<machine.h>
#include"SH7040.H"
/*-------------------------------------------------------------------------*/
/*                              PROTOTYPE                                  */
/*-------------------------------------------------------------------------*/
void pwm_1(void);
/*-------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                             */
/*-------------------------------------------------------------------------*/
#define pul_cyc1        (*(unsigned short *)0xfffff000)
#define pul_duty1b      (*(unsigned short *)0xfffff002)
#define pul_cyc2        (*(unsigned short *)0xfffff004)
#define pul_duty2b      (*(unsigned short *)0xfffff006)
#define pul_cyc3        (*(unsigned short *)0xfffff008)
#define pul_duty3b      (*(unsigned short *)0xfffff00a)
#define pul_duty3c      (*(unsigned short *)0xfffff00c)
#define pul_duty3d      (*(unsigned short *)0xfffff00e)
/*-------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                               */
/*-------------------------------------------------------------------------*/
void pwm_1(void)
    {
    PFCE.PEIOR = 0x0550;        /* TIOC1A,TIOC2A,TIOC3A/C output */
    PFCE.PECR1 = 0x0011;        /* TIOC3A/C output */
    PFCE.PECR2 = 0x1100;        /* TIOC1A,TIOC2A output */

    T1.TCR1 = 0x20;             /* TGR1A compare/match clear */
    T1.TIOR1 = 0x02;            /* start'0' compare/match '1' output */
    T1.TCNT1 = 0x0000;          /* timer counter'0' set */
    T1.TGR1A = pul_cyc1;        /* timer general register set */
    T1.TGR1B = pul_duty1b;
    T1.TMDR1 = 0xc2;            /* pwm mode 1 */

    T2.TCR2 = 0x20;             /* TGR2A compare/match clear */
    T2.TIOR2 = 0x02;            /* start'0' compare/match '1' output */
    T2.TCNT2 = 0x0000;          /* timer counter'0' set */
    T2.TGR2A = pul_cyc2;        /* timer general register set */
    T2.TGR2B = pul_duty2b;
    T2.TMDR2 = 0xc2;            /* pwm mode 1 */

    T3.TCR3 = 0x20;             /* TGR3A compare/match clear */
    T3.TIOR3H = 0x02;           /* start'0' compare/match '1' output */
    T3.TIOR3L = 0x21;           /* start'0' compare/match '1' output */
    T31.TCNT3 = 0x0000;         /* timer counter '0' set */
    T31.TGR3A = pul_cyc3;       /* timer general register set */
    T31.TGR3B = pul_duty3b;
    T31.TGR3C = pul_duty3c;
    T31.TGR3D = pul_duty3d;
    T3.TMDR3 = 0xc2;            /* pwm mode 1 */

    TMRSH.TSTR = 0x46;          /* timer start */
    while(1);
    }
```

（１）図2.4.1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7相のPWM出力をします。
（２）28.7MHzで動作時、出力するPWM周期は69.6nSから2.28mSの間任意に設定できます。

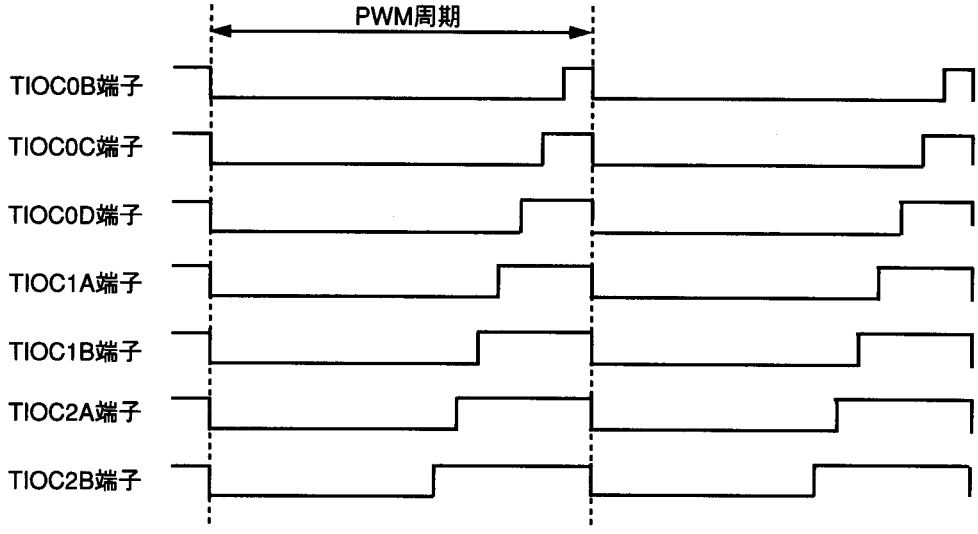


図2.4.1　PWM出力例

（１）本タスク例ではMTUのch0～2まで同期動作させ、7相のPWM出力をします。

（ａ）図2.4.2に本タスク例で使用するMTUのブロック図を示します。

本タスク例では、MTUの以下の機能を使用しています。

・ソフトウエアを介さずハードウエ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マッチ）

・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毎に出力が反転する機能。（トグル出力）

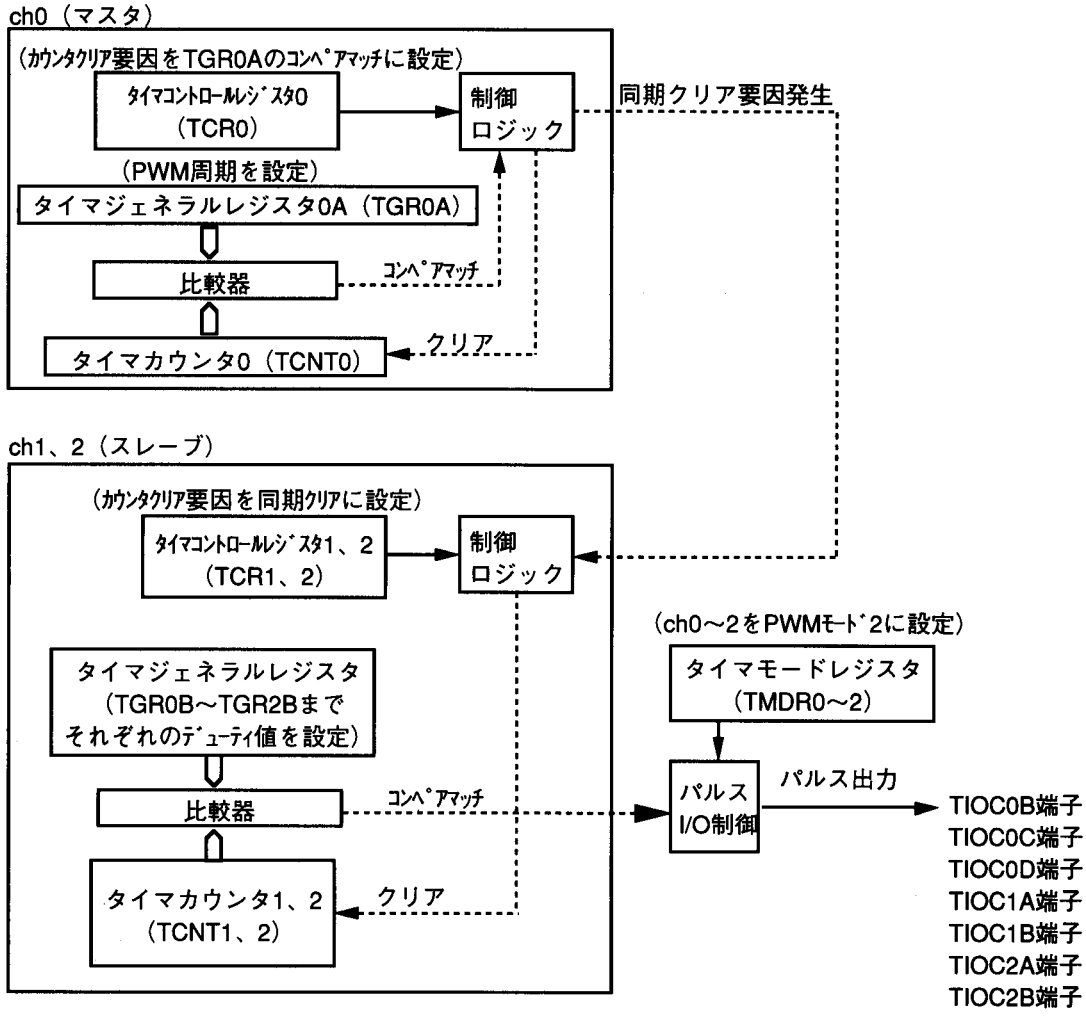


図2.4.2　同期クリアブロック図

（２）表2.4.1に本タスクの機能割り付けを示します。表に示すようにMTUの機能を割り付け、PWMパルスを出力します。

表2.4.1　MTU機能割り付け

端子、レジスタ名	機能割り付け
TIOC0B TIOC0C TIOC0D TIOC1A TIOC1B TIOC2A TIOC2B	PWMパルス出力端子
TSYR	ch0～2は同期動作
TCR0～2	ch0～2のタイマカウンタのクリア要因と入力クロックを選択
TGR0A	PWM周期の設定
TGR0B TGR0C TGR0D TGR1A TGR1B TGR2A TGR2B	デューティ値の設定
TMDR0～2	CH0～2をPWMﾓｰﾄﾞ2として動作

PWM7相出力	MCU	SH7040	使用機能	MTU（PWMﾓｰﾄﾞ2）

動作説明

図2.4.3に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理により、ch0～ch2の各PWM出力端子(TIOC0B/C/D、TIOC1A/B、TIOC2A/B)から7相のPWM出力をします。

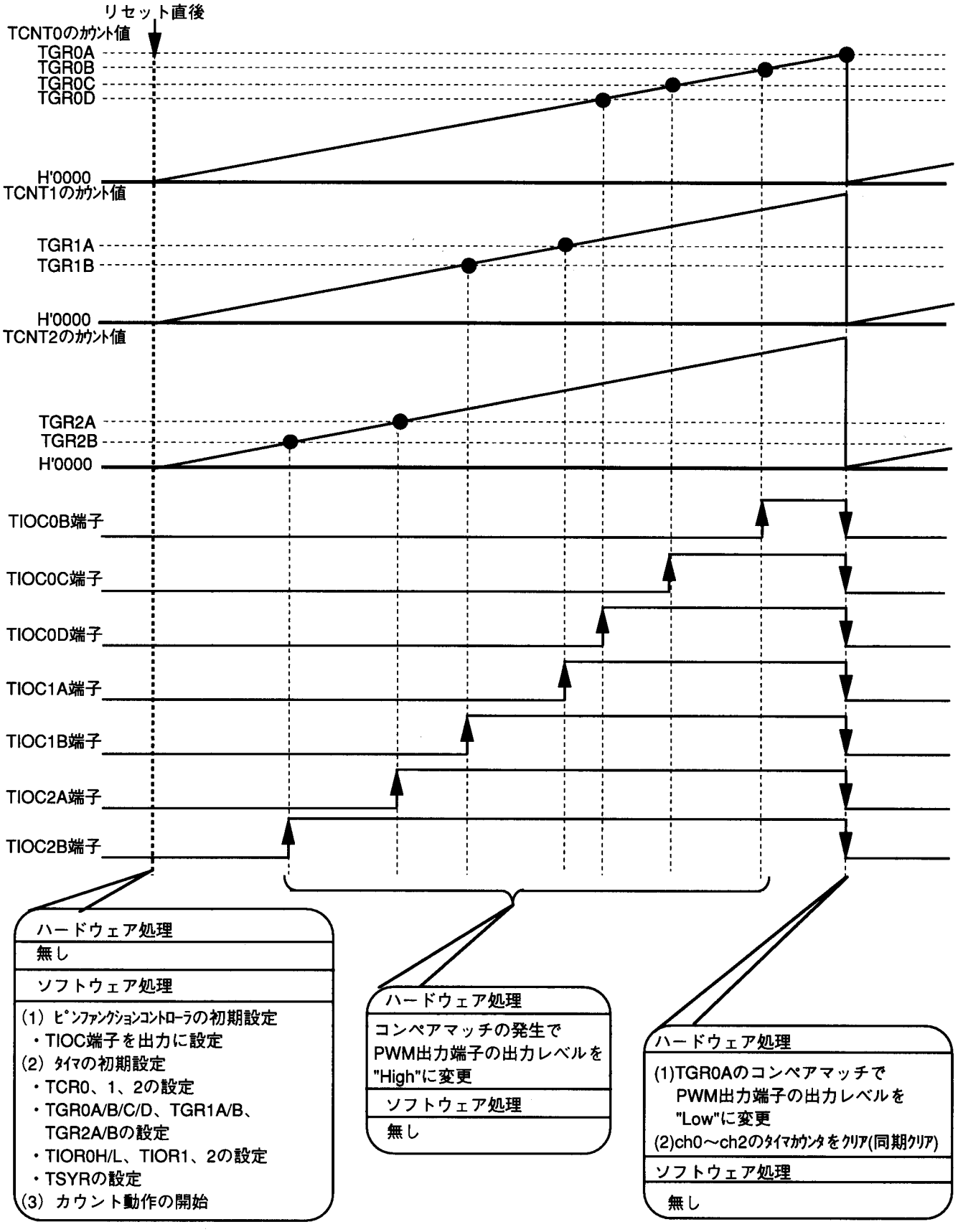


図2.4.3　ノコギリ波形の生成に用いるPWM出力(7相)の動作原理

PWM7相出力	MCU	SH7040	使用機能	MTU（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	機能割り付け
メインルーチン	pwm_2	PFCおよびPWM出力の設定

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
pul_cyc0a	パルスの周期に相当するタイマ値を設定 パルスの周期は以下の式にて算出 ﾊﾟﾙｽ周期(nS)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8nS)	1ワード	メインルーチン	入力
pul_duty0b pul_duty0c pul_duty0d pul_duty1a pul_duty1b pul_duty2a pul_duty2b	TIOC端子から出力される波形変化タイミングを設定			

（3）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
PFCE.PECR2	TIOC0B/C/D、TIOC1A/B、TIOC2A/Bを出力端子に設定	メインルーチン
TMRSH.TSTR	タイマカウントスタートの実行	
TMRSH.TSYR	タイマカウンタ0～1を同期動作に設定	
T0.TCR0 T1.TCR1 T2.TCR2	タイマカウンタのクリア要因をTGR0Aのコンペアマッチでクリア 入力クロックはφを選択	
T0.TGR0A	PWM周期を設定	
T0.TGR0B	TIOC0Bから"High"出力させるタイマカウンタ値を設定	
T0.TGR0C	TIOC0Cから"High"出力させるタイマカウンタ値を設定	
T0.TGR0D	TIOC0Dから"High"出力させるタイマカウンタ値を設定	
T1.TGR1A	TIOC1Aから"High"出力させるタイマカウンタ値を設定	
T1.TGR1B	TIOC1Bから"High"出力させるタイマカウンタ値を設定	
T2.TGR2A	TIOC2Aから"High"出力させるタイマカウンタ値を設定	
T1.TGR2B	TIOC2Bから"High"出力させるタイマカウンタ値を設定	
T0.TIOR0H	TGR0Aは初期出力0でアウトプットコンペアで0出力、 TGR0Bは初期出力0でアウトプットコンペアで1出力に設定	
T0.TIOR0L	TGR0Cは初期出力0でアウトプットコンペアで1出力、 TGR0Dは初期出力0でアウトプットコンペアで1出力に設定	
T1.TIOR1	TGR1Aは初期出力0でアウトプットコンペアで1出力、 TGR1Bは初期出力0でアウトプットコンペアで1出力に設定	
T1.TIOR2	TGR1Aは初期出力0でアウトプットコンペアで1出力、 TGR1Bは初期出力0でアウトプットコンペアで1出力に設定	
T0.TMDR0 T1.TMDR1 T2.TMDR2	各チャネルの動作モードをPWMモード2に設定	

（4）使用ＲＡＭ説明

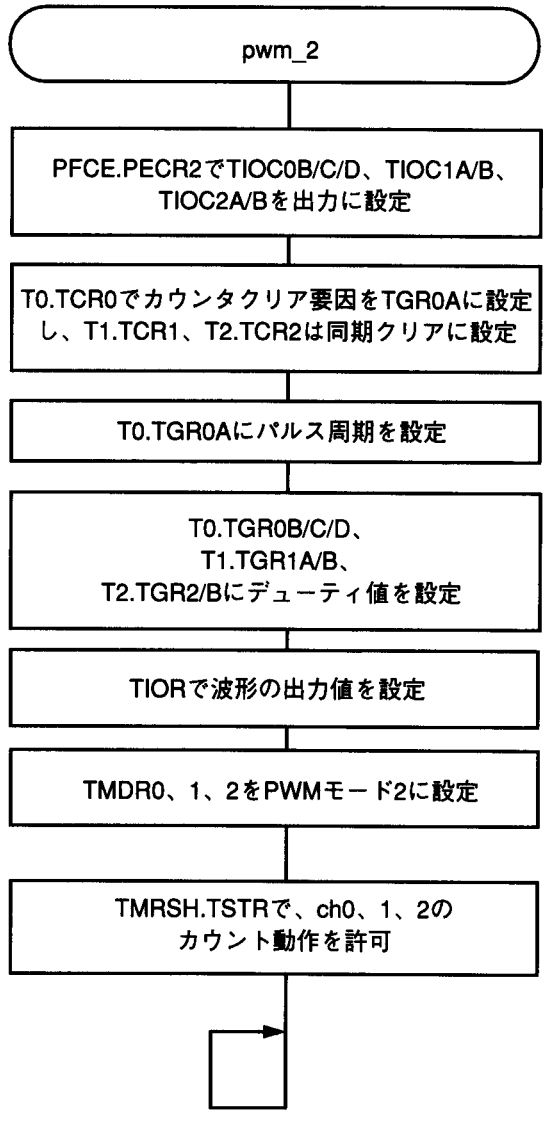
本アプリケーション例では引数以外のRAMは使用していません。

注）レジスタのラベル名は、SH7040ヘッダファイルの名前を使用しています。

PWM7相出力	MCU	SH7040	使用機能	MTU（PWMﾓｰﾄﾞ2）

フローチャート

メインルーチン



プログラムリスト

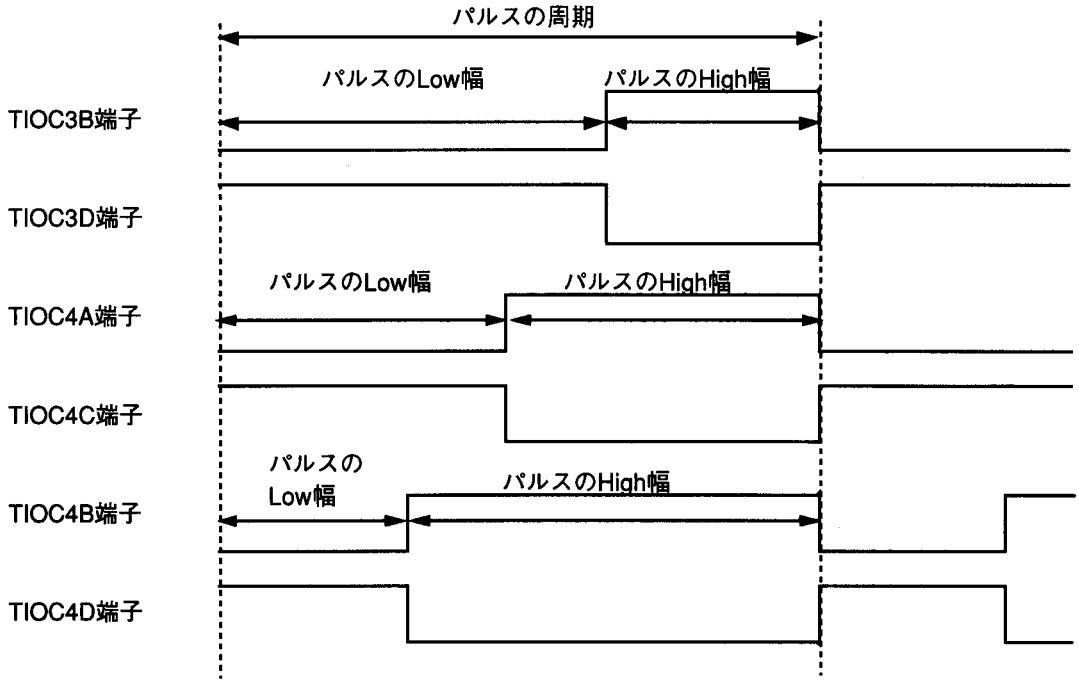
```
/*-------------------------------------------------------------------------*/
/*                              INCLUDE FILE                               */
/*-------------------------------------------------------------------------*/
#include<machine.h>
#include"SH7040.H"
/*-------------------------------------------------------------------------*/
/*                              PROTOTYPE                                  */
/*-------------------------------------------------------------------------*/
void pwm_2(void);
/*-------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                             */
/*-------------------------------------------------------------------------*/
#define pul_cyc0        (*(unsigned short *)0xfffff000)
#define pul_duty0b      (*(unsigned short *)0xfffff002)
#define pul_duty0c      (*(unsigned short *)0xfffff004)
#define pul_duty0d      (*(unsigned short *)0xfffff006)
#define pul_duty1a      (*(unsigned short *)0xfffff008)
#define pul_duty1b      (*(unsigned short *)0xfffff00a)
#define pul_duty2a      (*(unsigned short *)0xfffff00c)
#define pul_duty2b      (*(unsigned short *)0xfffff00e)
/*-------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                               */
/*-------------------------------------------------------------------------*/
void pwm_2(void)
     {
     PFCE.PEIOR = 0x00fe;       /* TIOC0B/C/D,TIOC1A/B,TIOC2A/B output */
     PFCE.PECR2 = 0x5554;       /* TIOC0B/C/D,TIOC1A/B,TIOC2A/B output */

     T0.TCR0 = 0x20;            /* TGR0A compare/match clear */
     T0.TIOR0H = 0x20;          /* start'0' compare/match '1' output */
     T0.TIOR0L = 0x22;          /* start'0' compare/match '1' output */
     T0.TCNT0 = 0x0000;         /* timer counter'0' set */
     T0.TGR0A = pul_cyc0;       /* timer general register set */
     T0.TGR0B = pul_duty0b;
     T0.TGR0C = pul_duty0c;
     T0.TGR0D = pul_duty0d;
     T0.TMDR0 = 0xc3;           /* pwm mode 2 */

     T1.TCR1 = 0x60;            /* TGR0A compare/match clear */
     T1.TIOR1 = 0x22;           /* start'0' compare/match '1' output */
     T1.TCNT1 = 0x0000;         /* timer counter'0' set */
     T1.TGR1A = pul_duty1a;     /* timer general register set */
     T1.TGR1B = pul_duty1b;
     T1.TMDR1 = 0xc3;           /* pwm mode 2 */

     T2.TCR2 = 0x60;            /* TGR0A compare/match clear */
     T2.TIOR2 = 0x22;           /* start'0' compare/match '1' output */
     T2.TCNT2 = 0x0000;         /* timer counter'0' set */
     T2.TGR2A = pul_duty2a;     /* timer general register set */
     T2.TGR2B = pul_duty2b;
     T2.TMDR2 = 0xc3;           /* pwm mode 2 */

     TMRSH.TSYR = 0x07;         /* ch0,1,2 synchronize */
     TMRSH.TSTR = 0x07;         /* timer start */
     while(1);
     }
```

2.5 正相・逆相ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ﾘｾｯﾄ同期PWMﾓｰﾄﾞ）

仕様

（１）図2.5.1に示すようにパルスHigh幅を変化させ、デューティ変化できるパルス（デューティパルス）を正相と逆相で3相出力します。

（２）28.7MHz動作時、出力するパルスの周期は34.8nSから2.28mSの間任意に設定できます。



$$\text{デューティ} = \frac{\text{パルスのHigh幅}}{\text{パルスの周期}} \times 100\ (\%)$$

図2.5.1　正相・逆相PWM 3相出力波形

使用機能説明

（１）本タスク例ではMTUのch3、4を組み合わせて使用し、一方の変化点が共通で、正相、逆相の関係にあるPWM波形を3相出力します。

・リセット同期PWMモードでは、TGRAとTGRCレジスタ、TGRBとTGRDレジスタをそれぞれペアでバッファ動作して、PWM出力を生成します。

（ａ）図２.5.2に本タスク例で使用するMTUのブロック図を示します。

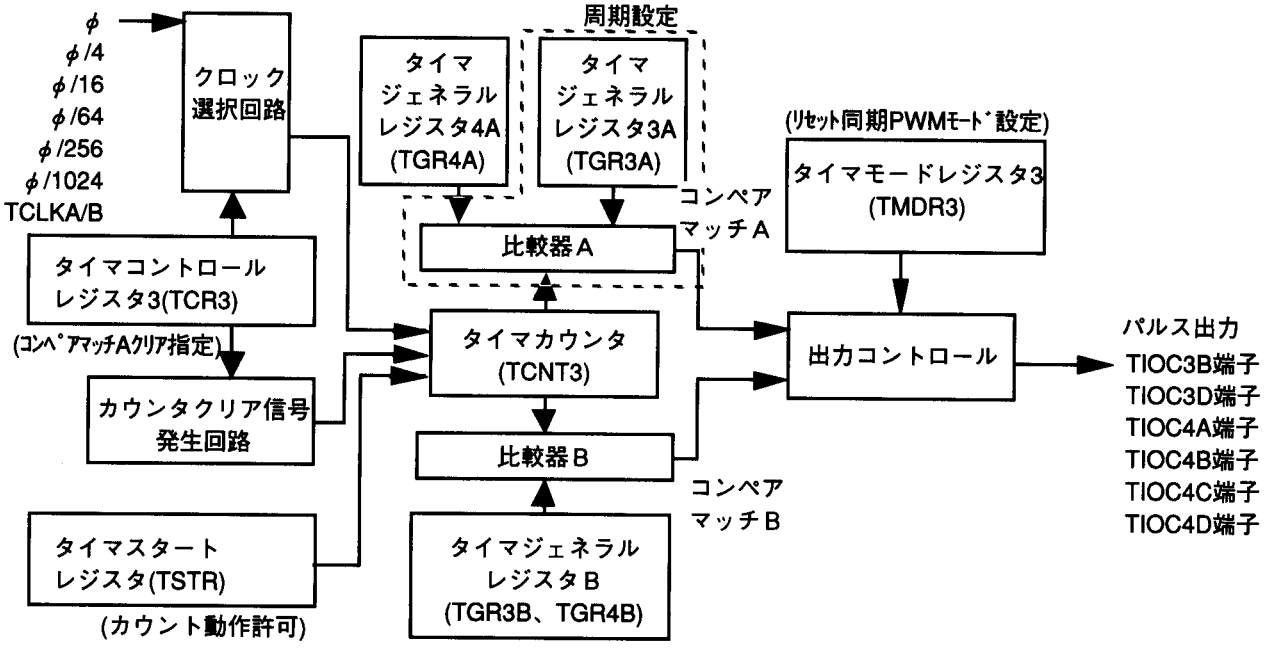


図2.5.2　MTU/ch3、4ブロック図

（２）表2.5.1に本タスクの機能割り付けを示します。表に示すようにMTUの機能を割り付け、PWM出力をします。

表2.5.1　機能割り付け

端子、レジスタ名	機能割り付け
TIOC3B	PWM出力1
TIOC3D	PWM出力1の逆相波形
TIOC4A	PWM出力2
TIOC4B	PWM出力3
TIOC4C	PWM出力2の逆相波形
TIOC4D	PWM出力3の逆相波形
TCR3	ch3のタイマカウンタのクリア要因と入力クロックを選択
TMDR3	ch3をリセット同期PWMモードとして動作
TGR3A	PWM周期を設定
TGR3B TGR4A TGR4B	デューティ値を設定

正相・逆相ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ﾘｾｯﾄ同期PWMﾓｰﾄﾞ）

動作説明

図2.5.3に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理により、各PWM出力端子(TIOC3B/D、TIOC4A/B/C/D)から6相のPWM波形を出力します。

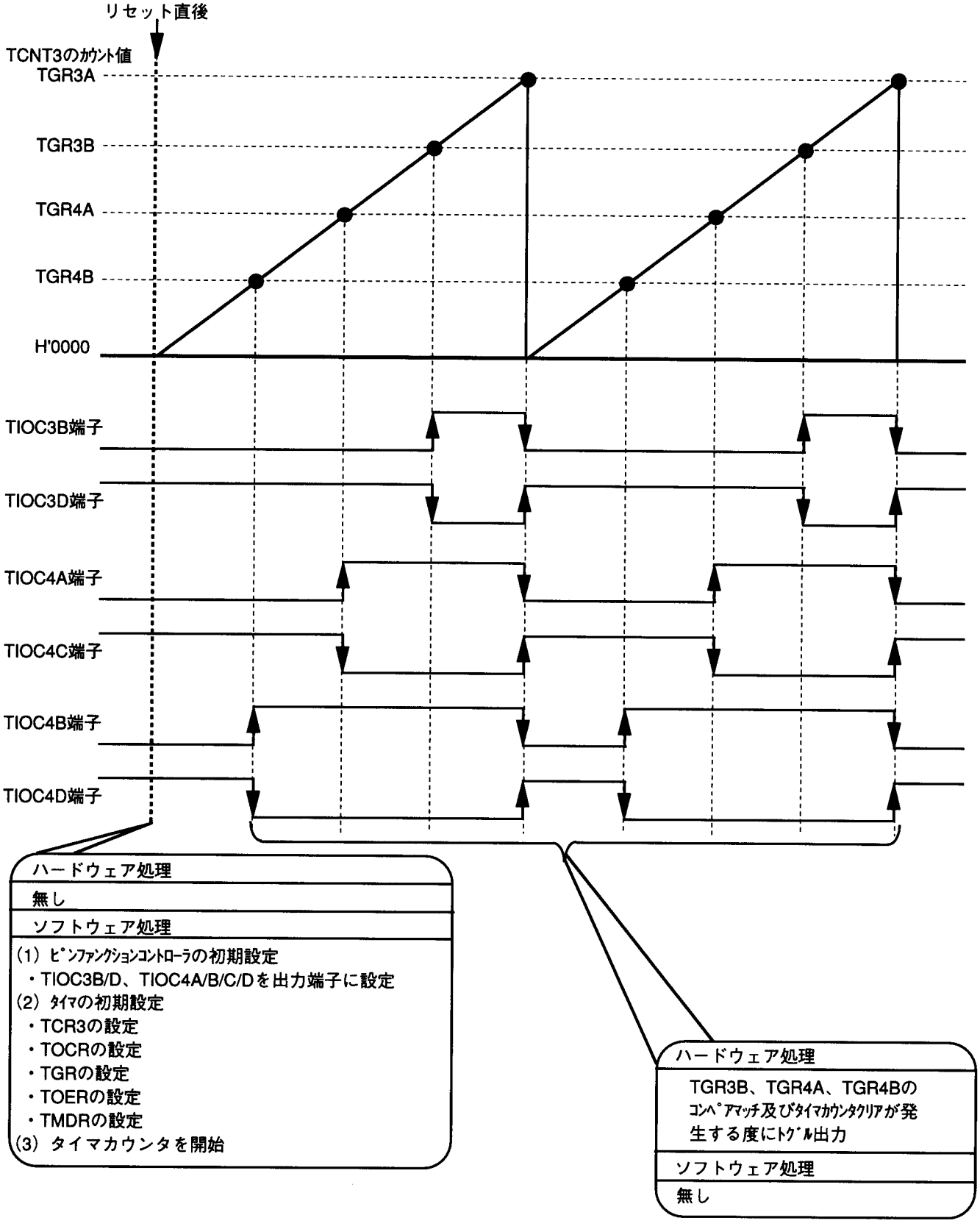


図2.5.3　リセット同期PWM波形の動作原理

正相・逆相ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ﾘｾｯﾄ同期PWMﾓｰﾄﾞ）

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	機能割り付け
メインルーチン	rst_pwm	PFCおよびPWM出力の設定

（２）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
pul_cyc1	パルスの周期に相当するタイマ値を設定 パルスの周期は以下の式にて算出 ﾊﾟﾙｽ周期(nS)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8nS)	1ワード	メインルーチン	入力
pul_duty3b pul_duty4a pul_duty4b	TIOC端子から出力される波形変化タイミングを設定			

（３）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
PFCE.PECR1	TIOC3B/D、TIOC4A/B/C/Dを出力端子に設定	メインルーチン
TMRSH.TSTR	ch3のタイマカウントのスタートを実行	
T3.TCR3	タイマカウンタのクリア要因をTGR3Aのコンペアマッチでクリア 入力クロックはφを選択	
T3.TOCR	PWM周期に同期したトグル出力の許可と、正相、逆相の出力レベルの設定	
T31.TGR3A	PWM周期を設定	
T31.TGR3B	TIOC3B/Dからトグル出力させるタイマカウンタ値を設定	
T31.TGR4A	TIOC4A/Cからトグル出力させるタイマカウンタ値を設定	
T31.TGR4B	TIOC4B/Dからトグル出力させるタイマカウンタ値を設定	
T3.TOER	リセット同期PWMの出力許可設定	
T3.TMDR3	リセット同期PWMモードの設定	

（４）使用ＲＡＭ説明

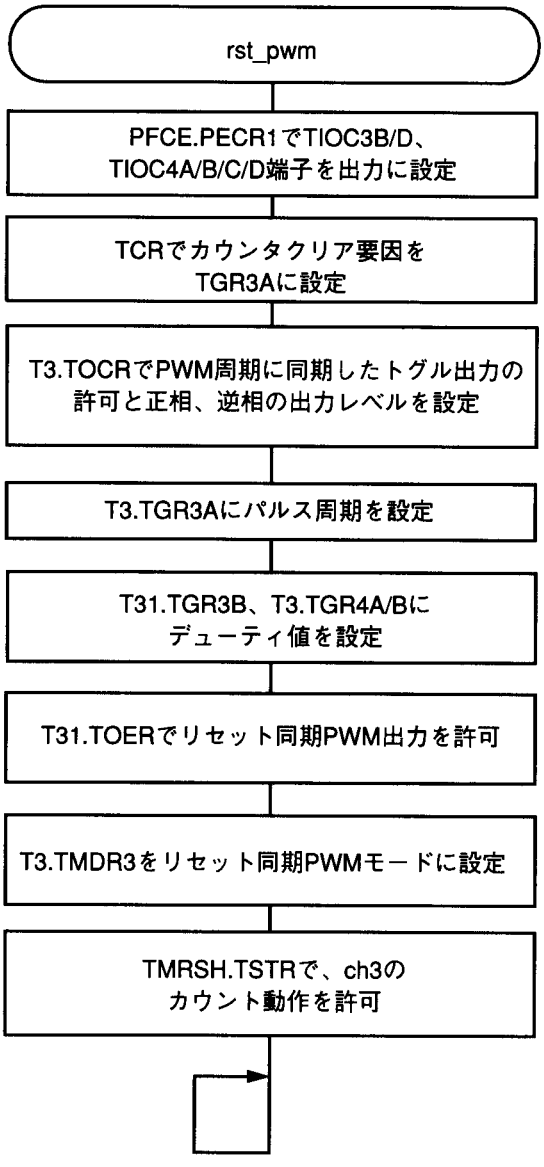
本アプリケーション例では引数以外のRAMは使用していません。

注）レジスタのラベル名は、SH7040ヘッダファイルの名前を使用しています。

正相・逆相ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ﾘｾｯﾄ同期PWMﾓｰﾄﾞ）

フローチャート

（１）メインルーチン



正相・逆相ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ﾘｾｯﾄ同期PWMﾓｰﾄﾞ）

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                              INCLUDE FILE                              */
/*------------------------------------------------------------------------*/
#include<machine.h>
#include"SH7040.H"
/*------------------------------------------------------------------------*/
/*                               PROTOTYPE                                */
/*------------------------------------------------------------------------*/
void rst_pwm(void);
/*------------------------------------------------------------------------*/
/*                             RAM ALLOCATION                             */
/*------------------------------------------------------------------------*/
#define pul_cyc1        (*(unsigned short *)0xfffff000)
#define pul_duty3b      (*(unsigned short *)0xfffff002)
#define pul_duty4a      (*(unsigned short *)0xfffff004)
#define pul_duty4b      (*(unsigned short *)0xfffff006)
/*------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                              */
/*------------------------------------------------------------------------*/
void rst_pwm(void)
    {
    PFCE.PEIOR = 0xfa00;        /* TIOC3B/D,TIOC4A/B/C/D output */
    PFCE.PECR1 = 0x5545;        /* TIOC3B/D,TIOC4A/B/C/D output */

    T3.TCR3 = 0x20;             /* TGR3A compare/match clear */
    T31.TCNT3 = 0x0000;         /* timer counter '0' set */
    T31.TGR3A = pul_cyc1;       /* timer general register set */
    T31.TGR3B = pul_duty3b;

    T31.TCNT4 = 0x0000;         /* timer counter '0' set */
    T31.TGR4A = pul_duty4a;     /* timer general register set */
    T31.TGR4B = pul_duty4b;

    T3.TOER = 0xff;             /* timer output enable register */
    T3.TOCR = 0x43;             /* timer output control register */
    T3.TMDR3 = 0xc8;            /* reset-synchronized pwm mode */
    TMRSH.TSTR = 0x40;          /* timer start */
    while(1);
    }
```

2.6　相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

仕様

（１）図2.6.1に示すように正相・逆相がノンオーバラップの関係にあるPWM波形を3相出力します。

（２）デューティは0%～100%まで任意にRAMに設定することにより変更ができます。

$$\text{デューティ} = \frac{\text{パルスのHigh幅}}{\text{パルスの周期}} \times 100\ (\%)$$

（３）周期に同期したトグル波形出力を行ないます。

（３）28.7MHz動作時、出力するパルスの周期は69.6nSから2.28mSの間任意に設定できます。

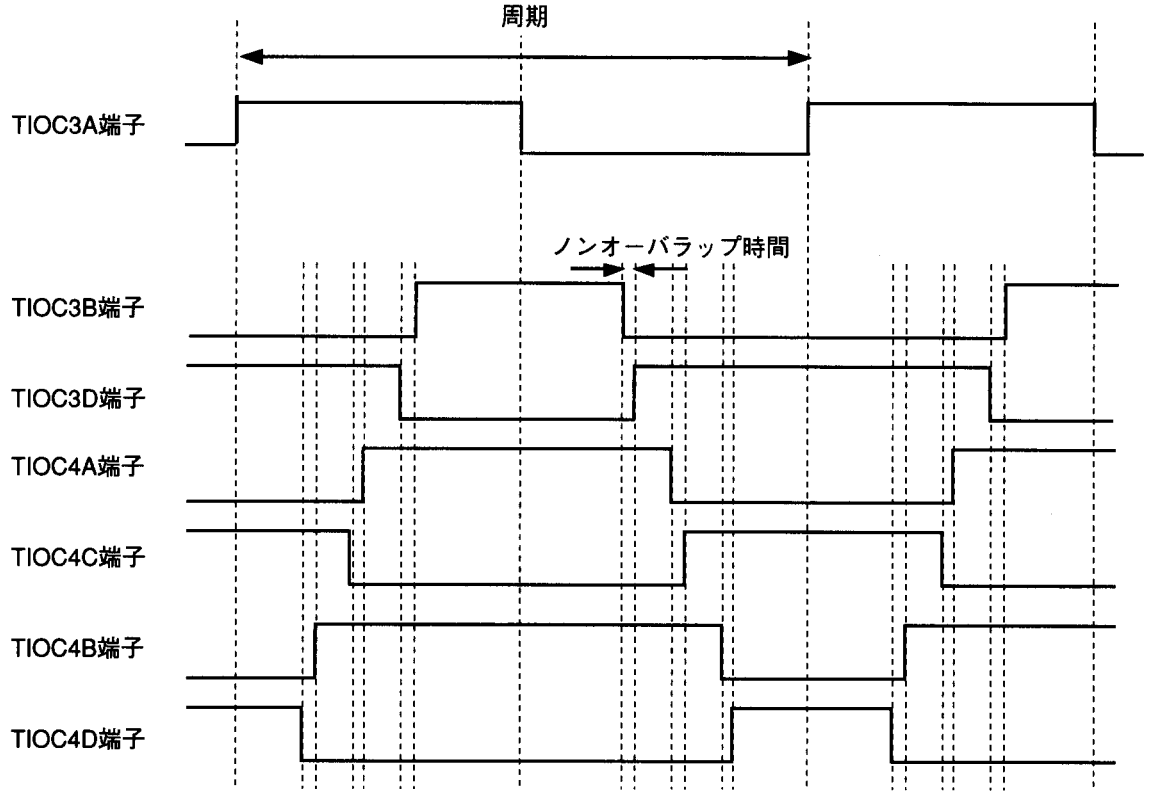


図2.6.1　相補PWM 3相出力波形

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

使用機能説明

（１）本タスク例ではMTUのチャネル3、4を使用して正相と逆相がノンオーバラップの関係にあるPWM波形を3相出力します。また、PWM波形の周期に同期したトグル波形出力をします。

（ａ）図2.6.2に本タスク例で使用するMTU/ch3、4のブロック図を示します。また本タスクでは、以下の機能を使用します。

・正相と逆相がノンオーバラップの関係にあるPWM波形を3相出力する機能。（相補PWMモード）

・コンペアマッチ発生時、バッファレジスタ（TGR3C/D、TGR4C/D）の内容をコンペアレジスタ（TGR3A/B、TGR4A/B）に転送する機能。

・PWM波形の周期に同期したトグル波形を出力する機能。

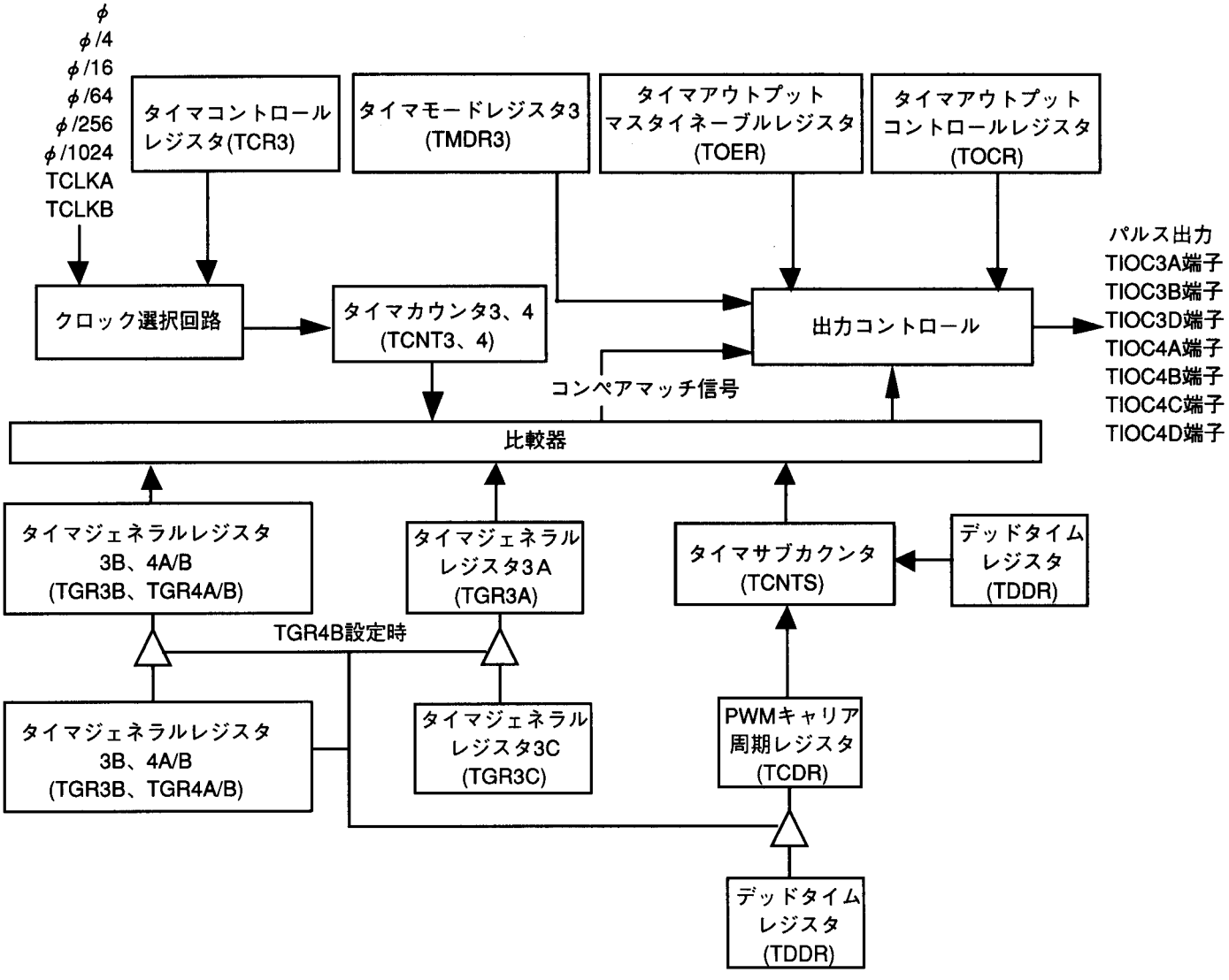


図2.6.2　MTU/ch3、4ブロック図

（２）表2.6.1に本タスクの機能割り付けを示します。表に示すようにMTUの機能を割り付け、PWMパルスを出力します。

表2.6.1　機能割り付け

端子、レジスタ名	機能割り付け
TIOC3A	PWMに同期したトグル出力
TIOC3C	PWM出力1
TIOC3D	PWM出力1とノンオーバラップの関係にある逆相波形
TIOC4A	PWM出力2
TIOC4B	PWM出力3
TIOC4C	PWM出力2とノンオーバラップの関係にある逆相波形
TIOC4D	PWM出力3とノンオーバラップの関係にある逆相波形
TOCR	PWM周期に同期したトグル出力の許可／禁止
TOER	相補PWM出力端子の信号出力の許可／禁止
TCR3	ch3のタイマカウンタのクリア要因と入力クロックを選択
TMDR3	ch3、4を相補PWMﾓｰﾄﾞとして動作
TGR3A	PWM周期の1/2＋デッドタイムの値を設定
TGR3C	TGR3Aのバッファレジスタ
TGR3B	出力パルスの変化点の設定（コンペアレジスタ）
TGR4A	
TGR4B	
TGR4C	TGR4Aのバッファレジスタ
TGR4D	TGR4Bのバッファレジスタ
TDDR	デッドタイムの設定
TCDR	周期の1/2を設定
TCBR	TCDRのバッファレジスタ

図2.6.3に動作原理を示します。SH7040のハードウェア処理およびソフトウェア処理により、相補PWM波形出力をします。

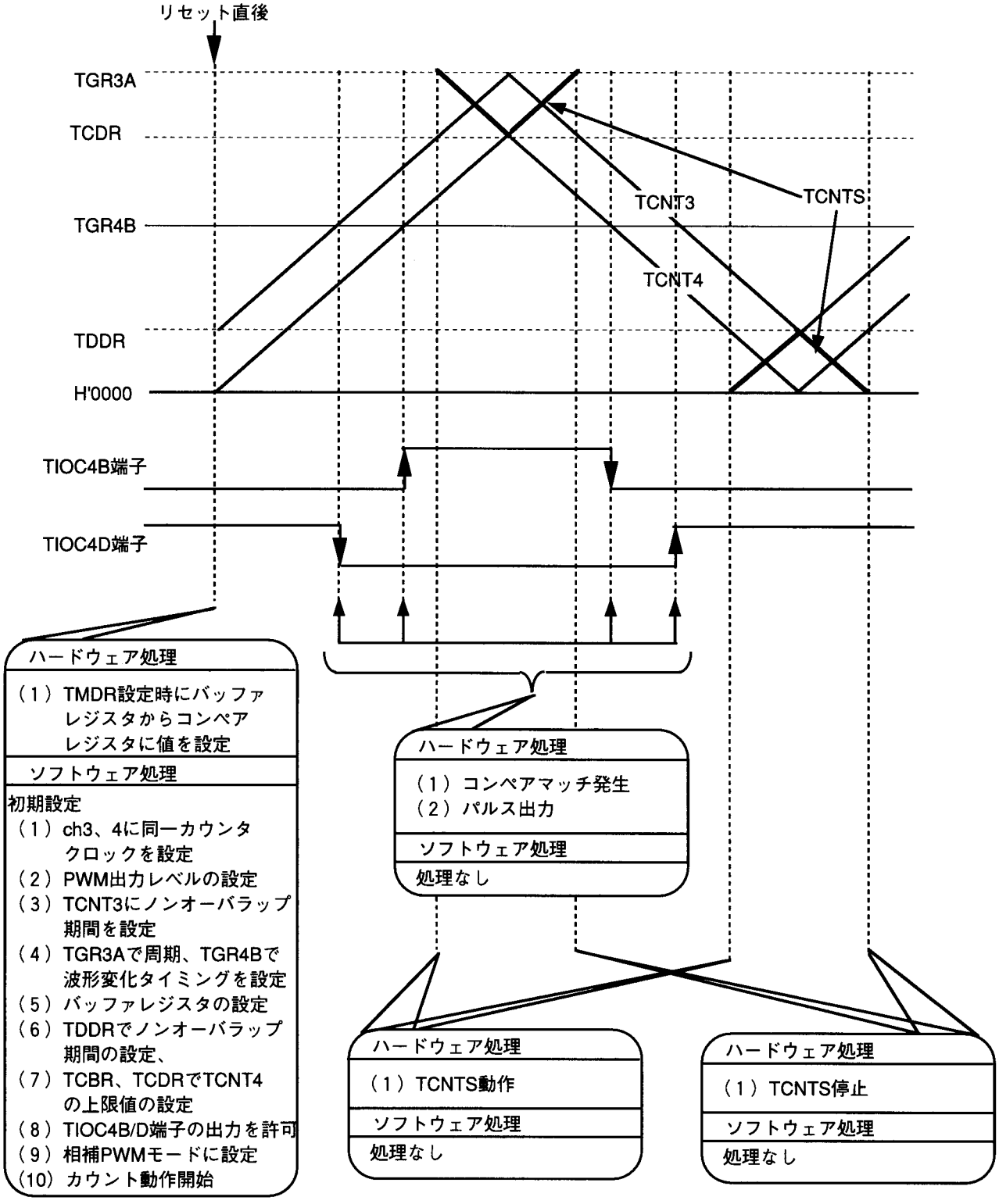


図2.6.3　相補PWM1相波形出力の動作原理

図2.6.4にPWM波形出力方式を示します。相補PWMモード設定時データ転送及びコンペアに関しては以下の法則にしたがっています。

データ転送

・Ta区間では、バッファレジスタにライトされたデータ（TGR4Dにデータを設定した時点）テンポラリレジスタに転送されます。
・Tb1区間では、転送モードが山で転送に設定時、バッファレジスタからテンポラリレジスタにデータは転送されません。Tb2区間ではTa区間と同様の動作をします。
　同様に谷に設定時Tb2区間では転送されません。
・バッファレジスタへのデータ転送は任意に行なうことができます。

コンペア

・Tb区間では、テンポラリレジスタ、コンペアレジスタ2本のレジスタとTCNT3、4及びTCNTSの3本のカウンタが比較されPWM波形を制御します。
・（a）領域では変更前のデータと（3）、（4）のコンペアマッチが優先されます。
・（b）領域では変更後のデータと（1）、（2）のコンペアマッチが優先されます。

但し、出力波形がアクティブレベルとなるコンペアマッチ（（1）、（3）のコンペアマッチ）の発生はそれぞれ出力波形がポジティブレベルとなるコンペア（（4）、（2）のコンペアマッチ）が発生後のみおこります。

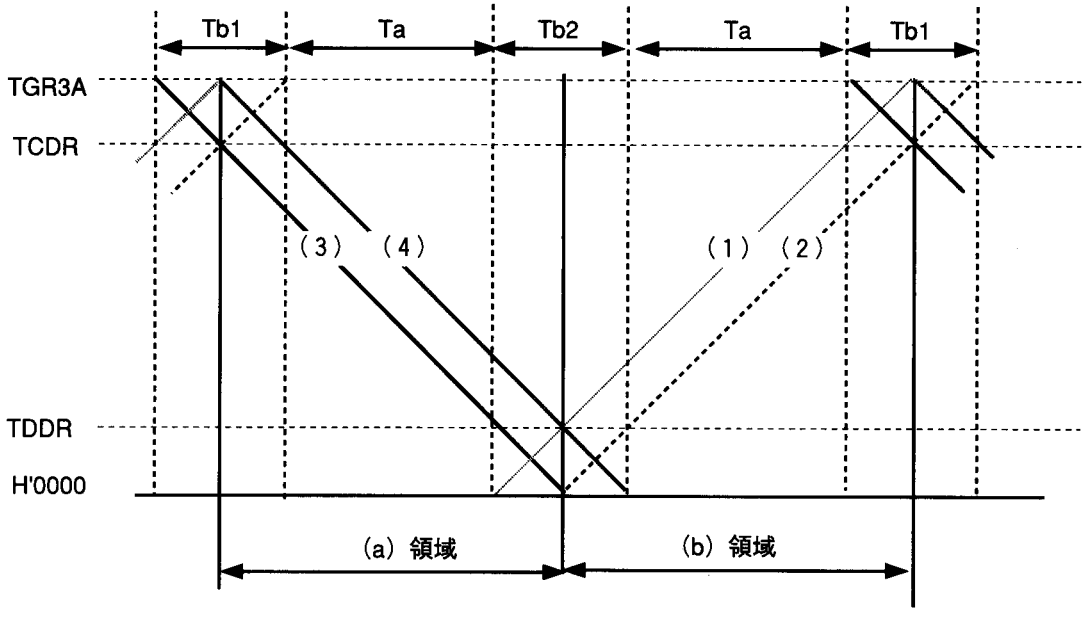


図2.6.4　PWM波形出力方式動作原理

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

図2.6.5に動作原理を示します。SH7040のハードウェア処理およびソフトウェア処理により、相補PWM波形出力をします。但し、転送モードは、谷でデータを変更するモードに選択してあります。

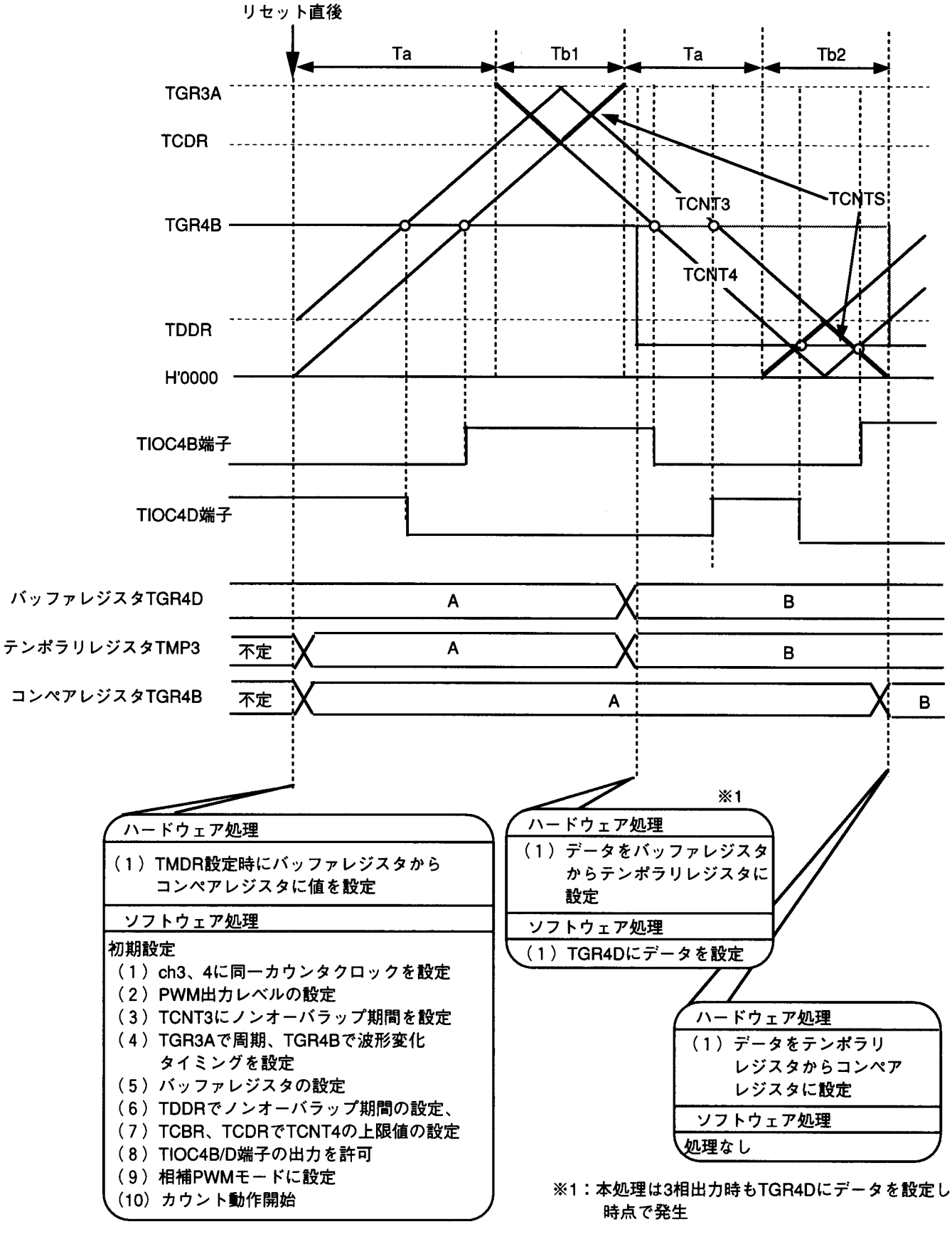
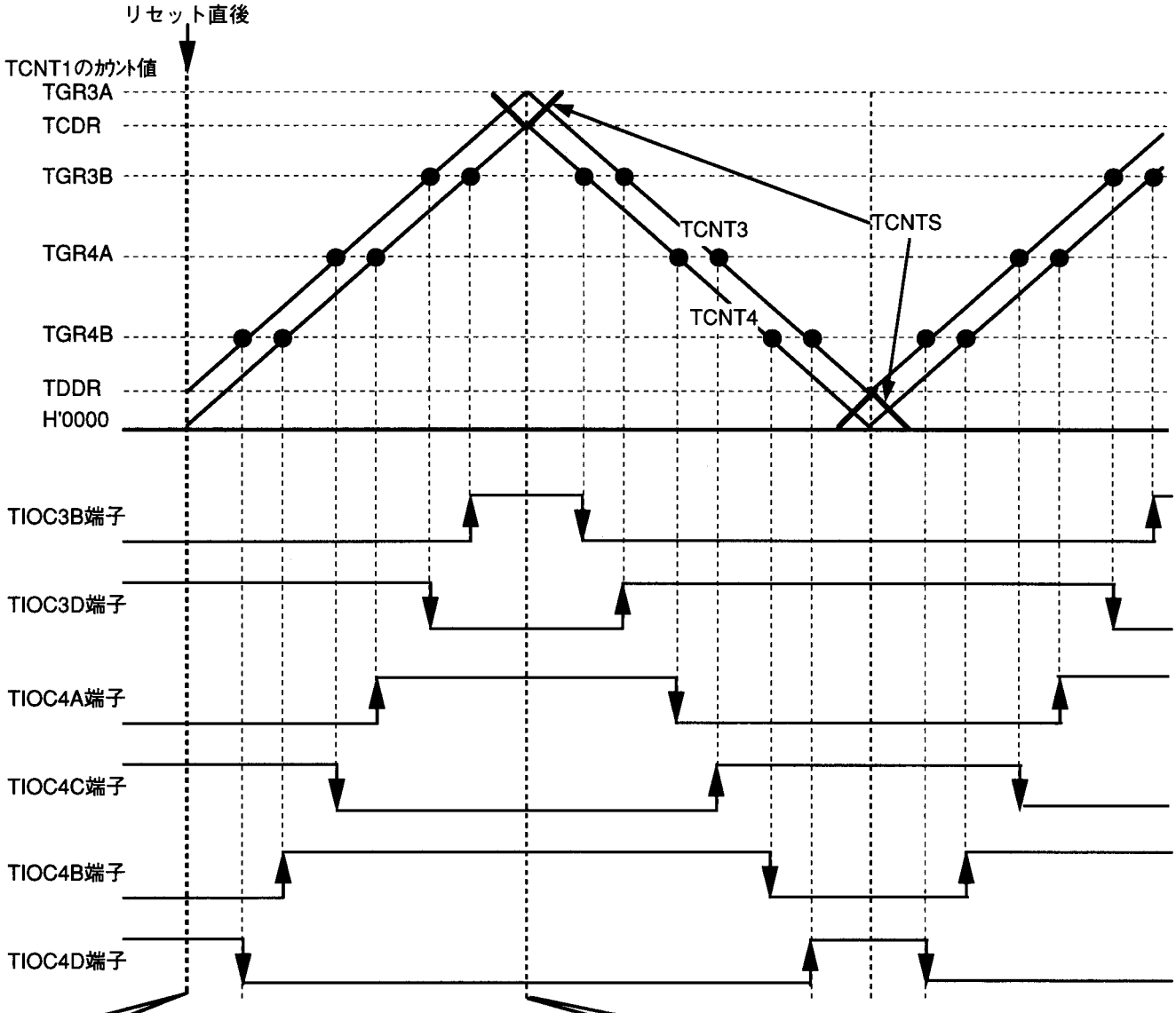


図2.6.5　相補PWM1相波形出力の動作原理

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

動作説明

図2.6.6に動作原理を示します。図に示すようにSH7040のハードウェア処理およびソフトウェア処理により、ch3、ch4の各PWM出力端子(TIOC3B/D、TIOC4A/B/C/D)から6相のPWM出力をします。



ハードウェア処理
無し
ソフトウェア処理
初期設定 （１）ch3、4に同一カウンタクロックを設定 （２）PWM周期に同期した波形の出力に設定 （３）PWM出力レベルの設定 （４）TCNT3にノンオーバラップ期間を設定 （５）TGR3Aで周期、TGR3B、TGR4A/Bで波形変化タイミングを設定 （６）バッファレジスタの設定 （７）TDDRでノンオーバラップ期間の設定、 （８）TCBR、TCDRでTCNT4の上限値の設定 （９）TIOC3A/B/D、TIOC4A/B/C/D端子の出力を許可 （10）相補PWMモードに設定 （11）カウント動作開始

ハードウェア処理
TGR3Aのコンペアマッチ割り込み発生
ソフトウェア処理
（１）RAMの内容をバッファレジスタに設定

図2.6.6　PWM波形の動作原理

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

図2.6.7に動作原理を示します。SH7040のハードウェア処理およびソフトウェア処理により、PWM周期に同期したトグル出力を行ないます。

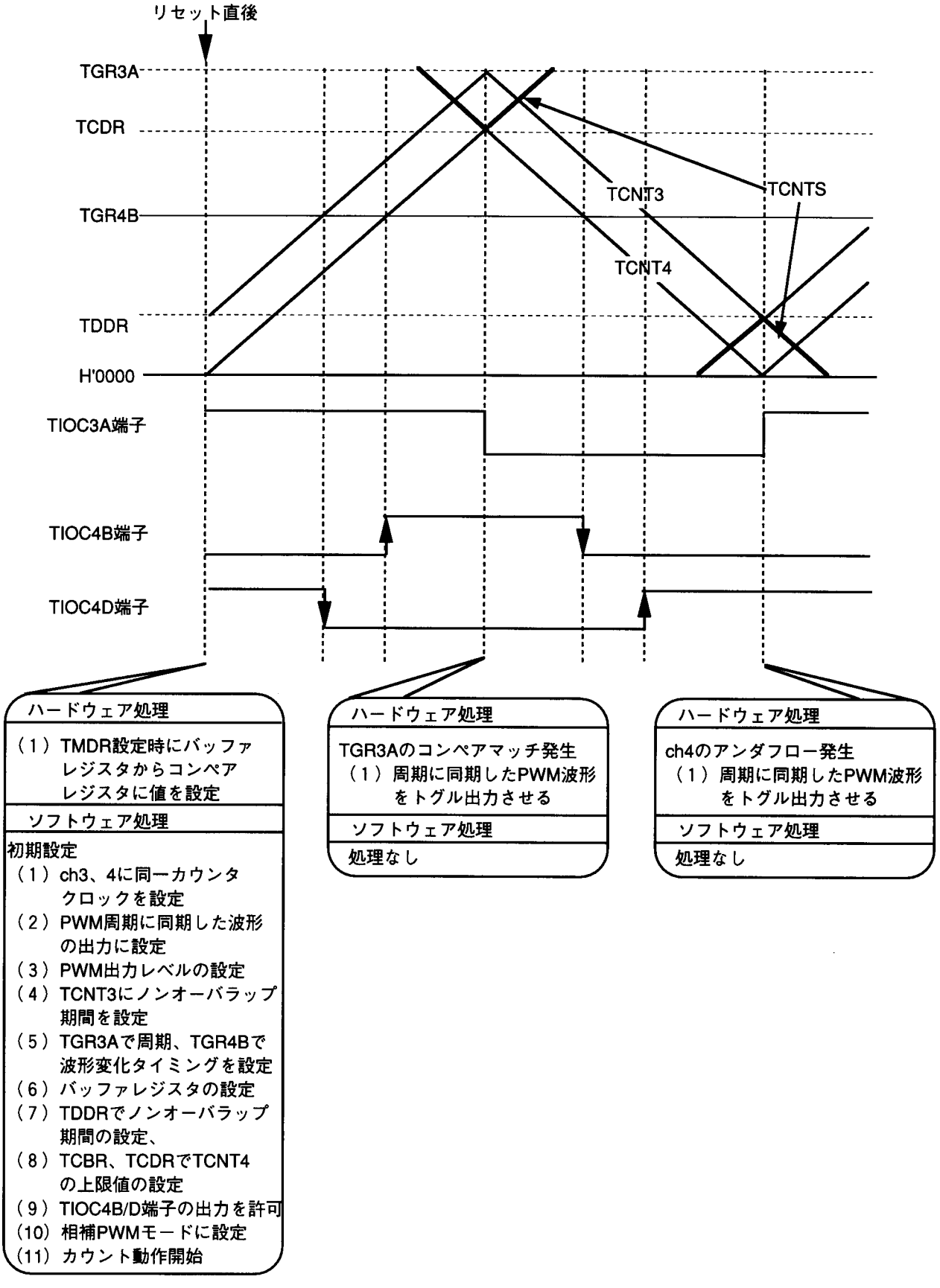


図2.6.7　PWM周期に同期したトグル波形出力の動作原理

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

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	機能割り付け
メインルーチン	comple	相補PWM出力の設定
データ設定	setdata	バッファレジスタに波形変化タイミングを設定

（２）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
pul_cyc1	パルスの1/2周期＋デッドタイム値を設定 パルスの周期は以下の式にて算出 ﾊﾟﾙｽ周期(nS)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8nS)	1ワード	メインルーチン	入力
pul_duty3d pul_duty4c pul_duty4d	TIOC端子から出力される波形変化タイミングを設定			
c_cyc	PWMキャリア周期レジスタの値を設定			
dead_time	ノンオーバラップ期間を設定		メインルーチン データ設定	

（３）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
PFCE.PEIOR	TIOC3B/D、TIOC4A/B/C/D端子を出力に設定	メインルーチン
PFCE.PECR1	TIOC3B/D、TIOC4A/B/C/D端子をMTU出力に設定	
TMRSH.TSTR	タイマカウントスタートを実行	
T3.TCR3	タイマカウンタのクリア要因と入力クロックを選択	
T3.TIER3	TGR3Aの割り込みを許可	
T31.TGR3A	キャリア周期の1/2＋デッドタイムレジスタの値を設定	
T31.TGR3B	TIOC3B、TIOC3Dから出力させるタイマカウンタ値を設定	
T31.TGR3C	T31.TGR3Aのバッファレジスタ	メインルーチン データ設定
T31.TGR3D	T31.TGR3Bのバッファレジスタ	
T31.TCNT3	デッドタイムの値を設定	メインルーチン
T31.TGR4A	TIOC4A、TIOC4Cから出力させるタイマカウンタ値を設定	
T31.TGR4B	TIOC4B、TIOC4Dから出力させるタイマカウンタ値を設定	
T31.TGR4A	T31.TGR4Aのバッファレジスタ	メインルーチン データ設定
T31.TGR4B	T31.TGR4Bのバッファレジスタ	
T31.TDDR	デッドタイムの値を設定	メインルーチン
T31.TCDR	周期の1/2の値を設定	
T31.TCBR	T31.TCDRのバッファレジスタ	メインルーチン データ設定
T3.TOCR	PWM周期に同期したトグル出力の許可と正相、逆相の出力レベルの設定	メインルーチン
T3.TOER	相補PWMの出力許可設定	
T3.TMDR3	相補PWMモードの設定	
INTC.IPRE	TGI0Aコンペアマッチ割り込み優先レベルを15に設定	

（４）使用ＲＡＭ説明

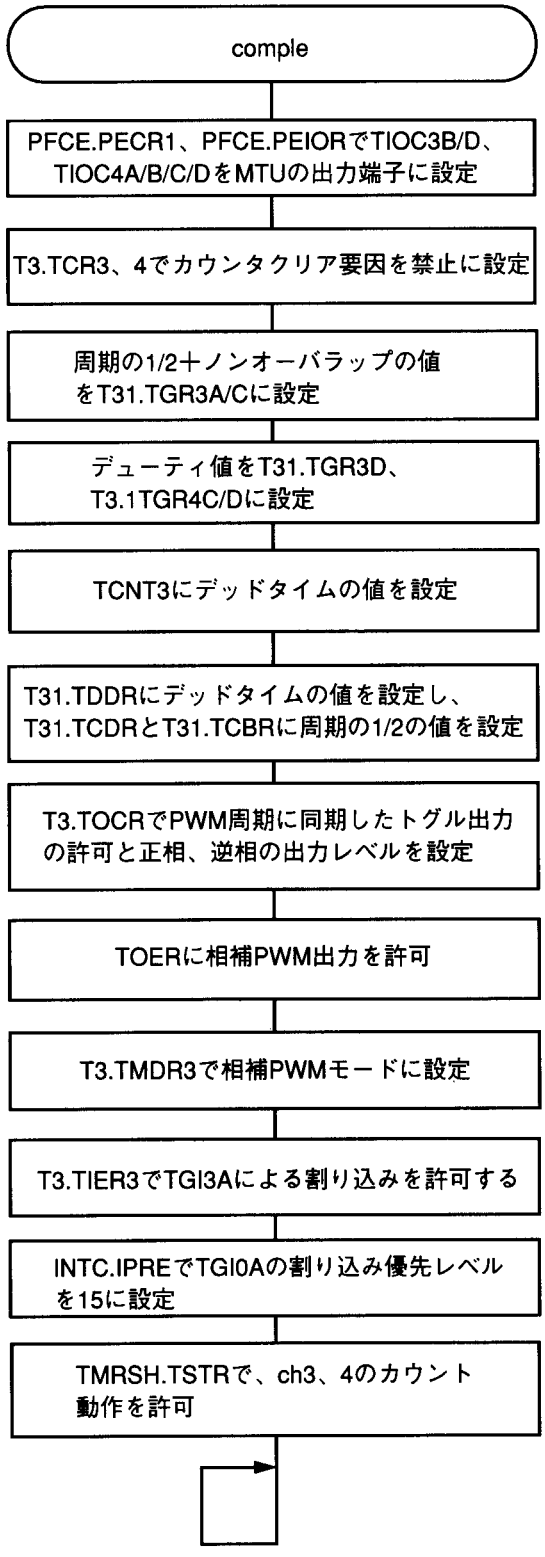
本アプリケーション例では引数以外のRAMは使用していません。

注）レジスタのラベル名は、SH7040ヘッダファイルの名前を使用しています。

相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

フローチャート

（1）メインルーチン



相補ＰＷＭ３相出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（相補PWMﾓｰﾄﾞ）

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                              INCLUDE FILE                              */
/*------------------------------------------------------------------------*/
#include<machine.h>
#include"SH7040.H"
/*------------------------------------------------------------------------*/
/*                              PROTOTYPE                                 */
/*------------------------------------------------------------------------*/
void comple(void);
#pragma interrupt(setdata)
/*------------------------------------------------------------------------*/
/*                            RAM ALLOCATION                              */
/*------------------------------------------------------------------------*/
#define pul_cyc1        (*(unsigned short *)0xfffff000)
#define pul_duty3d      (*(unsigned short *)0xfffff002)
#define pul_duty4c      (*(unsigned short *)0xfffff004)
#define pul_duty4d      (*(unsigned short *)0xfffff006)
#define c_cyc           (*(unsigned short *)0xfffff008)
#define dead_time       (*(unsigned short *)0xfffff00a)
/*------------------------------------------------------------------------*/
/*                            MAIN PROGRAM                                */
/*------------------------------------------------------------------------*/
void comple(void)
     {
     PFCE.PEIOR = 0xfb00;       /* TIOC3B/D,TIOC4A/B/C/D output */
     PFCE.PECR1 = 0x5545;       /* TIOC3B/D,TIOC4A/B/C/D output */

     T3.TCR3 = 0x00;            /* not clear */
     T31.TGR3C = pul_cyc1;      /* TGR3A buffer register */
     T31.TGR3D = pul_duty3d;    /* TGR3D buffer register */
     T31.TCNT3 = dead_time;     /* dead time set */

     T3.TCR4 = 0x00;            /* don't clear */
     T31.TGR4C = pul_duty4c;    /* TGR4A buffer register */
     T31.TGR4D = pul_duty4d;    /* TGR4B buffer register */
     T31.TCNT4 = 0x0000;        /* timer '0' set */
     T31.TDDR = dead_time;      /* dead time register */
     T31.TCBR = c_cyc;          /* TCDR buffer register */

     T3.TOCR = 0x43;            /* timer output control register */
     T3.TOER = 0xff;            /* timer output enable register */
     T3.TMDR3 = 0xff;           /* coplementary-pwm mode */
     T3.TIER3 = 0x01;           /* timer interrupt enable register */
     INTC.IPRE = 0x00f0;        /* set initialize level=15 */
     set_imask(0x0);            /* set imask level=0 */
     TMRSH.TSTR = 0xc0;         /* timer start */
     while(1);
     }

void setdata()
{
    T31.TSR3 &= 0xfe;           /*clear flag */
    T31.TCBR = c_cyc;
    T31.TGR3C = pul_cyc1;
    T31.TGR3D = pul_duty3d;
    T31.TGR4C = pul_duty4c;
    T31.TGR4D = pul_duty4d;
}
```

2.7　2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

仕様

（1）図2.7.1に示すようにch1に2本の外部クロックを入力し、そのパルスの位相差によりカウンタをカウントアップまたはカウントダウンします。またch0に設定した測定時間毎(測定時間1/2)に同期してch1のカウント数を測定し、結果をRAMに設定します。

（2）タイマカウンタの初期値をH'0000とし、ソフトウェアカウンタを用いて-2147483648～2147483647までカウントできます。

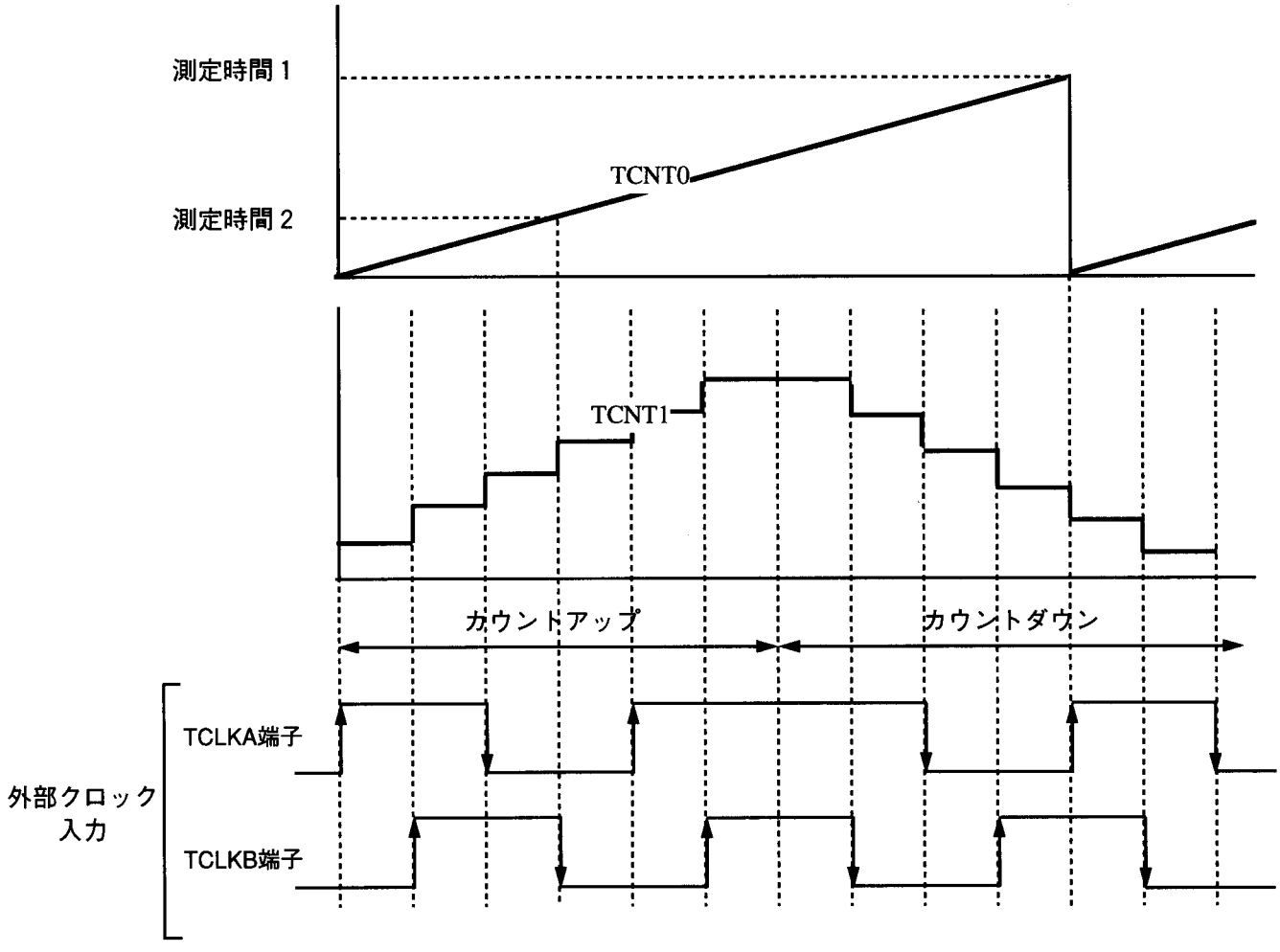


図2.7.1　2相エンコーダカウンタの取り込み

2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

使用機能説明

（1）本タスク例では、MTUのch1をアップ／ダウンカウンタとして使用し、TGR0A/Bに測定時間を設定します。TGR0A/Bのアウトプットコンペアをトリガとしてch1のインプットキャプチャで制御周期時のTCNT1の値を取り込みます。また、ch0のインプットキャプチャを用いてch1のカウンタ入力クロック幅を取り込みます。

（a）図2.7.2にch0のブロック図を示します。ch0では、以下の機能を使用して測定時間毎にch1のインプットキャプチャのトリガを出力します。ch1はインプットキャプチャ信号入力時にTCNT1の値を測定します。

・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）

・パルスの入力エッジの検出を行いタイマ値を内蔵レジスタに取り込む機能。（インプットキャプチャ）

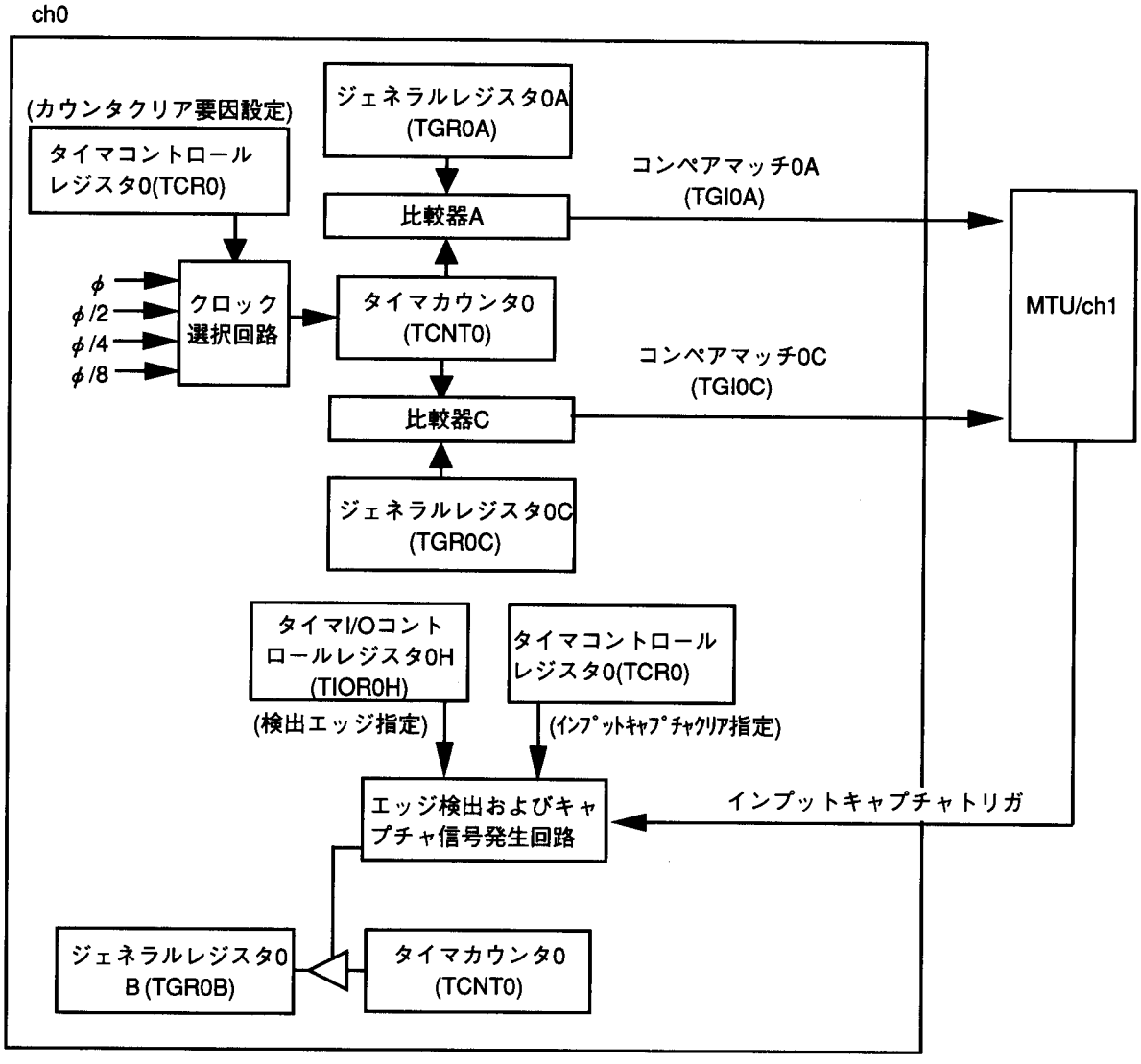


図2.7.2　MTU/ch0ブロック図

（b）図2.7.3にch1のブロック図を示します。ch2では以下の機能を使用してタイマカウンタをアップ／ダウンカウントします。インプットキャプチャ立ち上がりエッジ検出時のカウンタ値を測定結果とします。

・2本の外部クロックの位相差を検出し、タイマカウンタをアップ／ダウンカウントする機能。
（位相計数モード）

・パルスの入力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取り込む機能。
（インプットキャプチャ）

・インプットキャプチャ発生時、割り込み処理を起動する機能。

・パルスの入力エッジ検出時タイマカウンタをクリアする機能。

・タイマカウンタのオーバーフローまたはアンダーフロー検出時、割り込み処理を起動する機能。

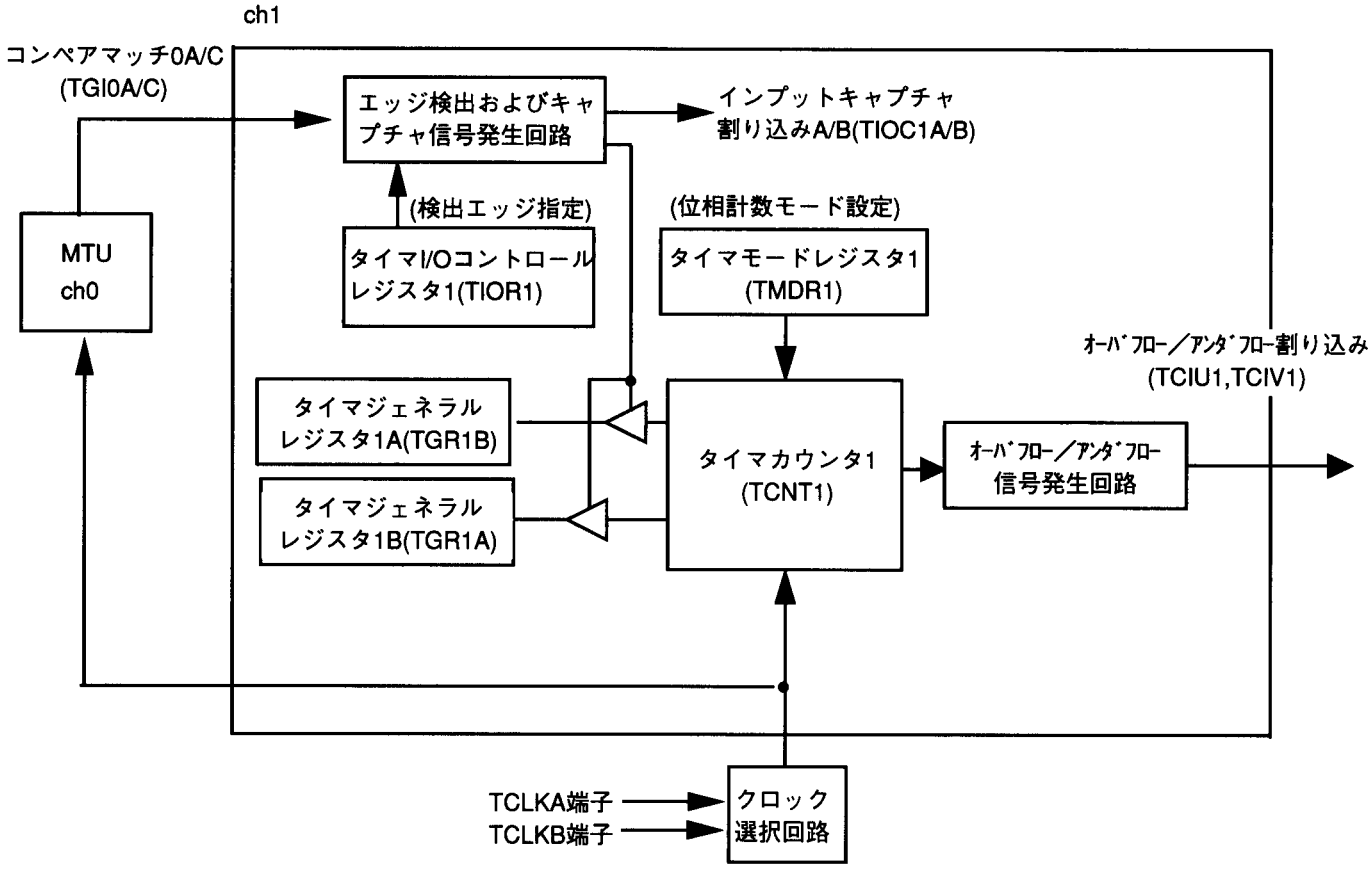


図2.7.3　MTU/ch1ブロック図

（2）表2.7.1に本タスク例の機能割り付けを示します。表に示すようにMTUの機能を割り付け、2相エンコーダパルスの2つの位相差を検出し、カウンタをアップ／ダウンカウントします。

表2.7.1 機能割り付け

端子・レジスタ名	機能割り付け
TCLKA TCLKB	外部クロック入力端子
TSTR	ch0,1のタイマカウンタ動作を許可／禁止
TCR0	カウンタクロック、カウンタクリア要因の選択
TIOR0H	TIOC0Aをアウトプットコンペアに設定。TIOC0Bをch0のアウトプットコンペア発生でインプットキャプチャに設定
TIOR0L	TIOC0Cをアウトプットコンペアに設定
TGR0A	測定時間1の設定
TGR0B	インプットキャプチャBによりカウント結果が格納
TGR0C	測定時間毎2を設定
TMDR1	位相計数モードの設定
TCR1	カウンタクロック、カウンタクリア要因の選択
TIOR0	TIOC0A/Cをch1のアウトプットコンペアの発生でインプットキャプチャに設定
TIER1	TIOC1A/B、TIOU1、TIOV1による割り込みを許可
TGR1A TGR1B	インプットキャプチャAによるカウント結果の格納

動作説明

図2.7.4に動作原理を示します。SH7040のハードウェア処理およびソフトウェア処理によりカウンタをカウントアップまたはカウントダウンします。

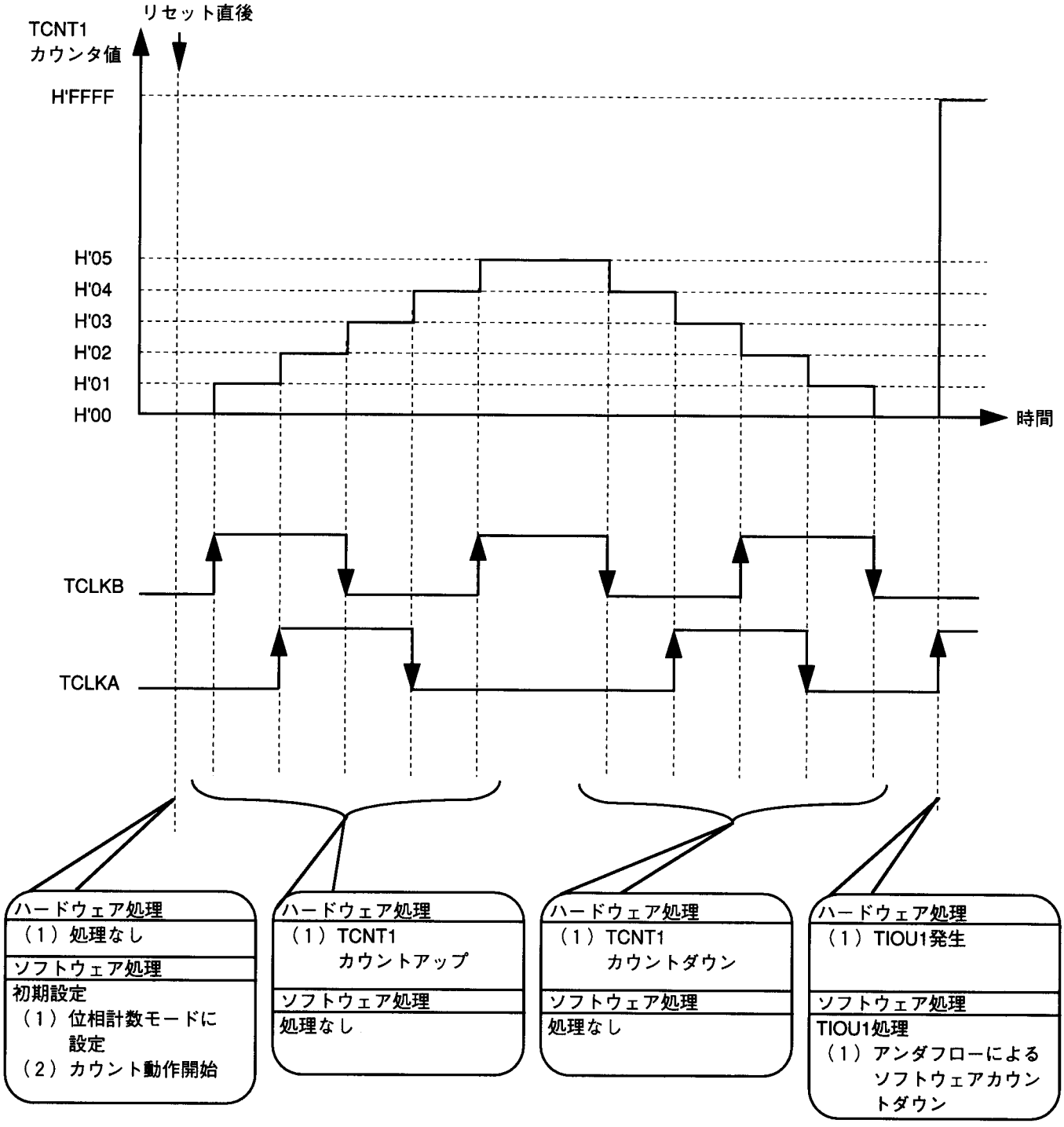
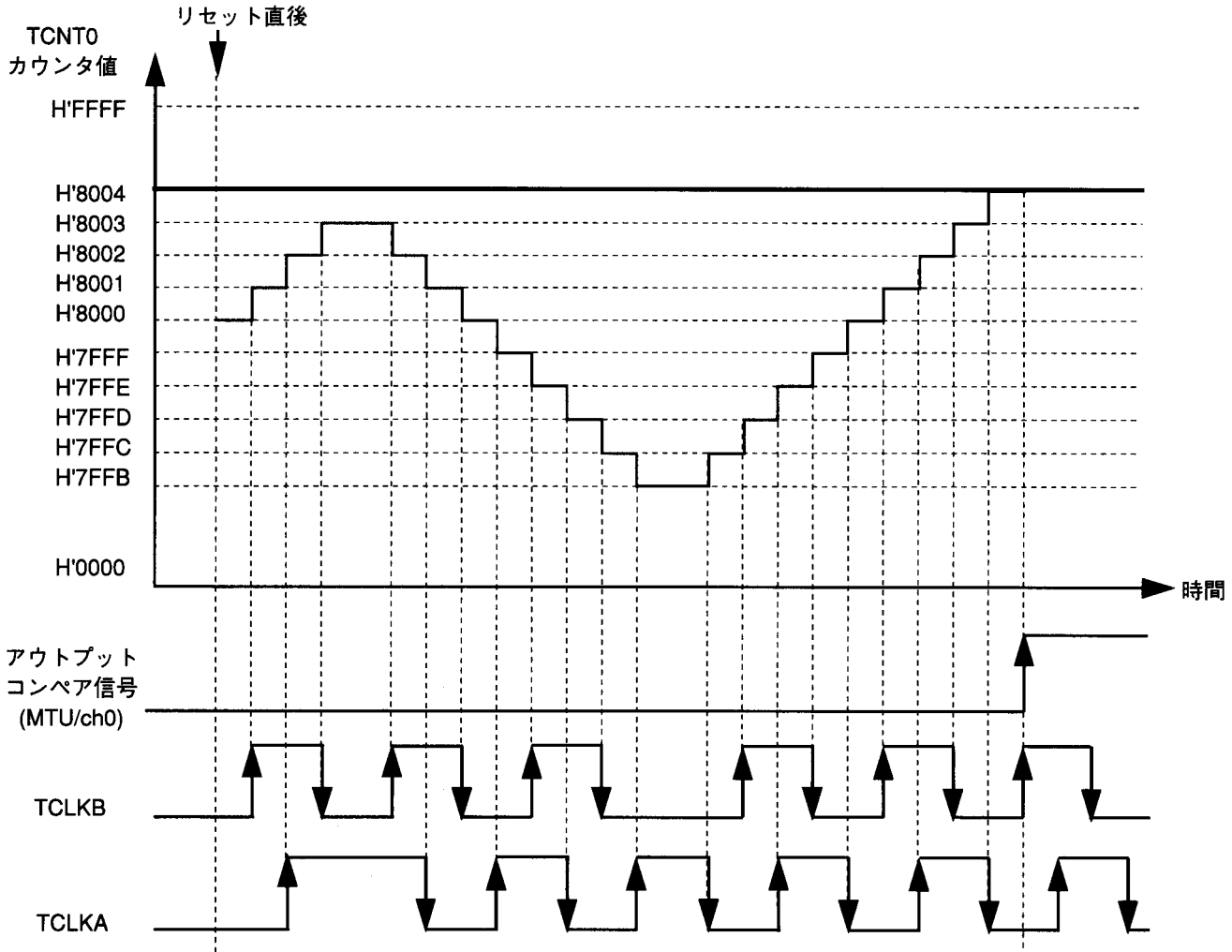


図2.7.4　位相計数モードの動作原理（1）

2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

動作説明

図2.7.5に示すようにSH7040のハードウェア処理およびソフトウェア処理によりカウンタのオーバフロー／アンダフロー時、外部イベント発生時に割り込み処理を実施します。



ハードウェア処理
処理なし
ソフトウェア処理
初期設定 （1）カウンタクロックをφに設定 （2）インプットキャプチャ信号をch0のアウトプットコンペアに設定 （3）測定時間毎をTGR0A/Bに設定 （4）位相計数モードに設定 （5）インプットキャプチャ信号をch1のアウトプットコンペアに設定 （6）インプットキャプチャA、インプットキャプチャB、およびオーバフロー／アンダフロー割り込み許可 （7）カウント動作許可

ハードウェア処理
（1）TGI1A発生 （2）TGI1B発生
ソフトウェア処理
TGI1A/B処理 （1）カウント結果をRAMに設定

図2.7.5位相計数モードの動作原理（2）

2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

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	機能割り付け
メインルーチン	en2	MTU等の初期設定
カウンタ値測定1	phacnt1	TGI1Aにより起動し、TGRAの値からアップ／ダウンカウント結果をRAMに設定 TGRCの値からカウンタ周期結果をRAMに設定
カウンタ値測定2	phacnt2	TGI1Bにより起動し、TGRBの値からアップ／ダウンカウント結果をRAMに設定
オーバフロー	ovf1	TIOV1により起動し、ソフトウェアカウンタのインクリメント
アンダフロー	unf1	TIOU1により起動し、ソフトウェアカウンタのデクリメント

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
msr_tim1 msr_tim2	カウンタ測定時間に相当するタイマ値を設定 測定時間は以下の式にて求める 測定時間(ns)=ﾀｲﾏ値×φ周期(28.7MHz動作時34.8ns)	ワード	メインルーチン	入力
cnt_data1 cnt_data2	アップ／ダウンカウント結果を設定	ロング ワード	カウンタ値測定1 カウンタ値測定2	出力
p_cycle	カウント周期結果を設定	ワード	カウンタ値測定1	

（3）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
TMRSH.TSRT	ch0/1のタイマカウンタ動作の許可／禁止	メインルーチン
T0.TCR0	カウンタクロック、カウンタクリア要因の選択	
T0.TIOR0H	TIOC0Aをアウトプットコンペアに設定する。TIOBC0B をch0のアウトプットコンペアでインプットキャプチャに設定	
T0.TIOR0L	TIOC0Cをアウトプットコンペアに設定	
T0.TGR0A	測定時間1を設定	
T0.TGR0B	インプットキャプチャBによりカウント結果の格納	カウンタ値測定1
T0.TGR0C	測定時間毎2を設定	メインルーチン
T1.TMDR1	位相計数モード設定	
T1.TCR1	カウンタクロック、カウンタクリア要因の選択	
T1.TIOR1	TIOC0A/Cをch1のアウトプットコンペア発生でインプットキャプチャに設定	
T1.TIER1	TGI1A/B、TIOU1、TIOV1による割り込みを許可	
T1.TGR1A	インプットキャプチャAによるカウント結果の格納	カウンタ値測定1
T1.TGR1B	インプットキャプチャBによるカウント結果がの格	カウンタ値測定2
T1.TSR1	TGI1A/B、TIOU1、TIOV1の発生状況	カウンタ値測定1、2 オーバフロー、アンダフロー

(4) 使用RAM説明

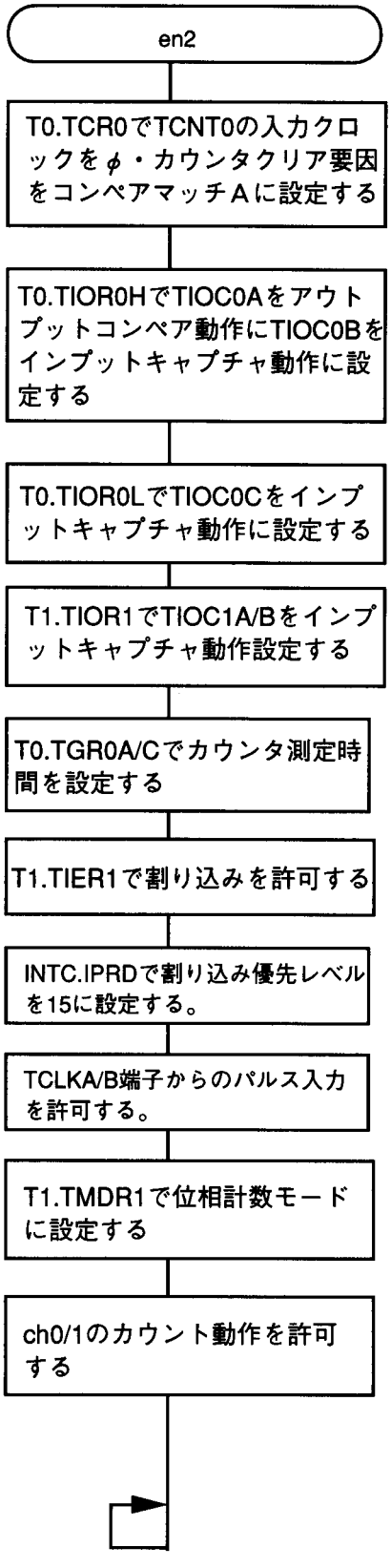
使用モジュール名	ラベル名	機能割り付け
カウンタ値測定1、2	wrk	データ設定時のワークとして使用
全モジュール	cnt	ソフトウェアカウンタ

注）レジスタのラベル名はSH7040ヘッダファイルの名前を使用しています。

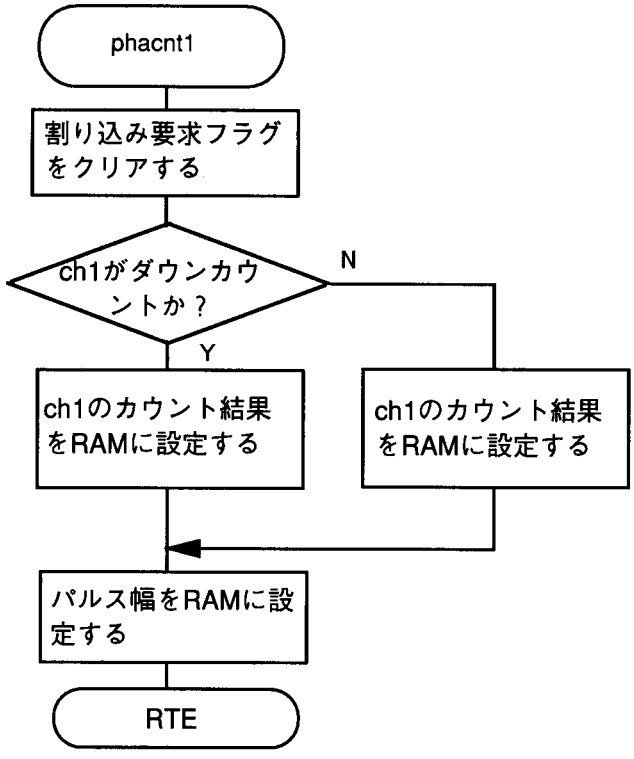
2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

フローチャート

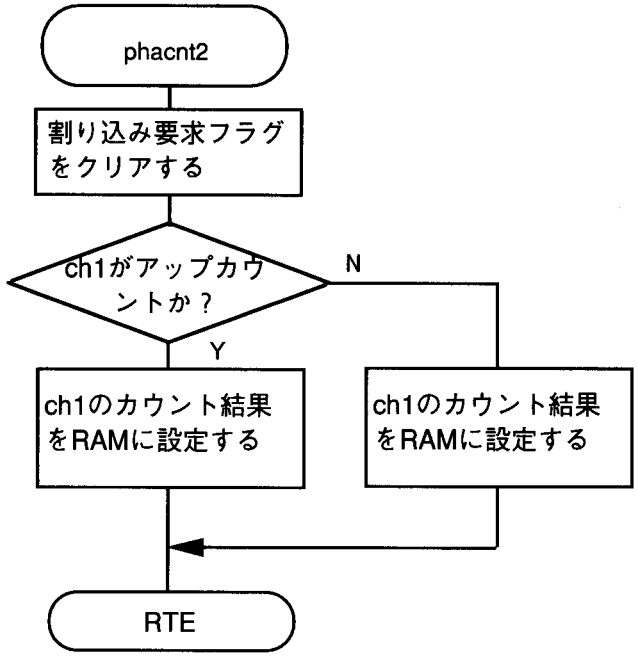
（1）メインルーチン



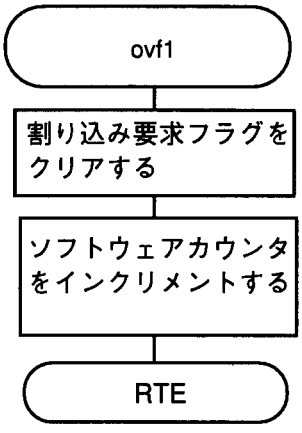
（2）カウンタ値測定1



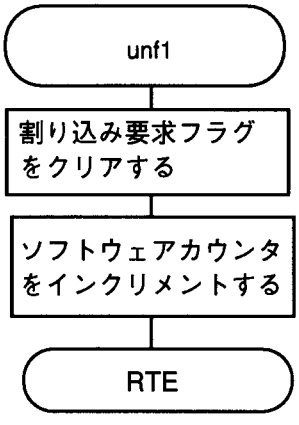
（3）カウンタ値測定2



（4）オーバフロー



（5）アンダフロ



2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

プログラムリスト

```
/*-------------------------------------------------------------------------*/
/*                              INCLUDE FILE                               */
/*-------------------------------------------------------------------------*/
#include <machine.h>
#include "SH7040.H"
/*-------------------------------------------------------------------------*/
/*                              PROTOTYPE                                  */
/*-------------------------------------------------------------------------*/
void en2(void);
#pragma interrupt(phacnt1,phacnt2,ovf1,unf1)
/*-------------------------------------------------------------------------*/
/*                            RAM ALLOCATION                               */
/*-------------------------------------------------------------------------*/
#define msr_tim1 (*(unsigned short *)0xfffff000)
#define msr_tim2 (*(unsigned short *)0xfffff002)
#define cnt_data2  (*(unsigned long *)0xfffff004)
#define cnt_data1  (*(unsigned long *)0xfffff006)
#define p_cycle  (*(unsigned long *)0xfffff008)
#define cnt      (*(unsigned long *)0xfffff00a)
#define wrk      (*(unsigned short *)0xfffff00a)
/*-------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                               */
/*-------------------------------------------------------------------------*/
void en2(void)
{
    T0.TCR0 = 0xa0;                 /* timer clear output compare TGRA0  */
    T0.TIOR0H = 0xf0;               /* output compare TIOC0A */
                                                /* input caputure TIOC0B */
    T0.TIOR0L = 0x00;               /* output compare TIOC0C */
    T1.TIOR1 = 0xff;                /* input caputure TIOC1A,B */
    T1.TIER1 = 0x33;                /* interrupt TIOC1A,TIOC1B,TCIU1,TCIV1 */
    T0.TGR0C = msr_tim2;            /* set posisition cycle */
    T0.TGR0A = msr_tim1;            /* set speed cycle */
    INTC.IPRD = 0x00ff;             /* set interrupt level=15 */
    set_imask(0x0);                 /* set imask level=0 */
    PFCA.PACRL2 = 0X5000;           /* TIOCnx sellect */
    T1.TMDR1 = 0x04;                /* set phase counting mode1*/
    T0.TMDR0 = 0x20;                /* TGRB and TGRD buffer mode */
    TMRSH.TSTR = 0x03;              /* start timer0,1 */
    while(1);                       /* loop */
}
```

2相エンコーダカウント	MCU	SH7040	使用機能	MTU（位相計数モード）

プログラムリスト

```
void ovf1(void)
{
    T1.TSR1 &= 0xef;                                          /* clear flag */
    cnt++;                                                    /* count up */
}


void unf1(void)
{
    T1.TSR1 &= 0xdf;                                          /* clear flag */
    cnt--;                                                    /* count down */
}


void phacnt1(void)
{
    T1.TSR1 &= 0xfe;                                          /* clear flag */
    wrk = T1.TGR1B;
    if(cnt < 0)                                               /* count < 0 */
        cnt_data1 = (unsigned long)wrk-0x010000+cnt*0x010000;  /* set sp */
    else
        cnt_data1 = (unsigned long)wrk+cnt*0x010000;          /* set sp */
        p_cycle = T0.TGR0D;                                   /* set width pulse*/
}


void phacnt2(void)
{
    T1.TSR1 &= 0xfd;                                          /* clear flag */
     wrk = T1.TGR1A;
     if(cnt < 0)
        cnt_data2 = (unsigned long)wrk-0x010000+cnt*0x010000;  /* set po */
     else
        cnt_data2 = (unsigned long)wrk+cnt*0x010000;          /* set po */
}
```

2.8　外部トリガによるタイマ波形の遮断	MCU	SH7040	使用機能	MTU、POE

仕様

（1）図2.8.1に示すように、外部信号の立ち下がりエッジに同期してタイマの出力波形をハイ・インピーダンス状態とすることで波形の遮断を行いますします。

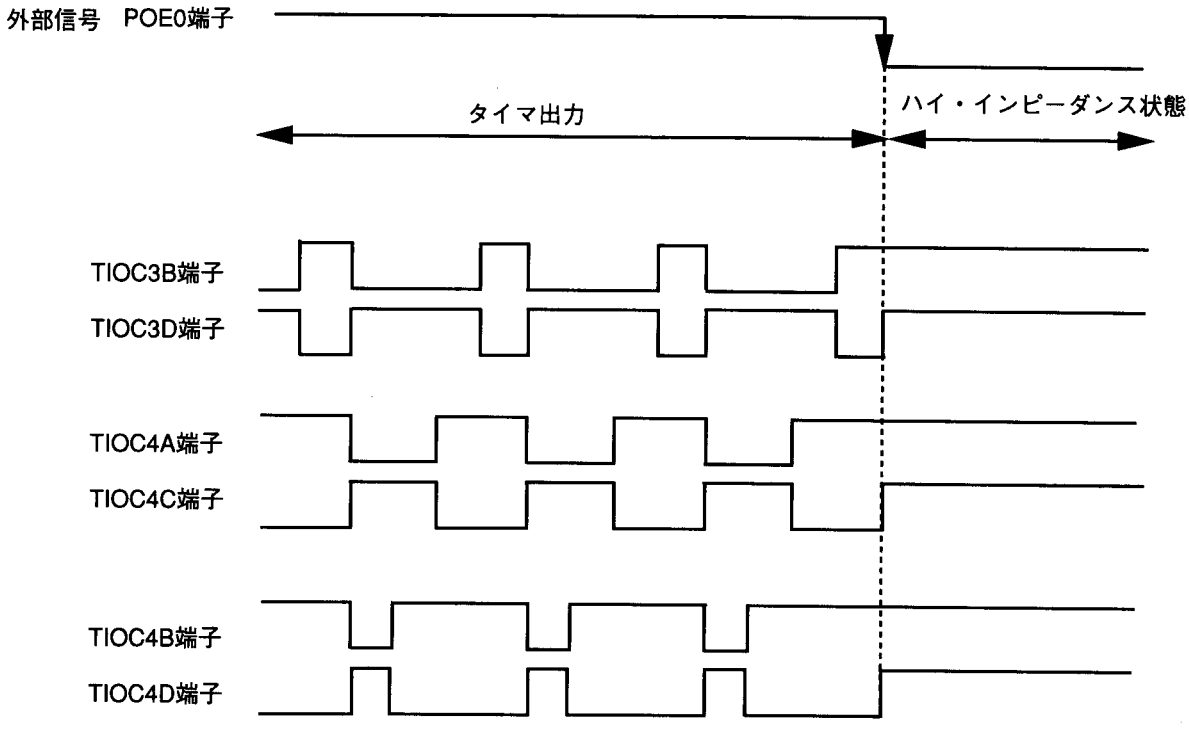


図2.8.1　外部トリガによる波形の遮断例

外部トリガによるタイマ波形の遮断	MCU	SH7040	使用機能	MTU、POE

使用機能説明

（1）本タスク例ではＭＴＵのチャネル3、4（リセット同期PWMモード）で出力している波形を外部信号立ち下がりエッジに同期してハイ・インピーダンス状態にすることで波形出力を遮断します。

（a）図2.8.2にMTU/ch3、4、POEブロック図を示します。

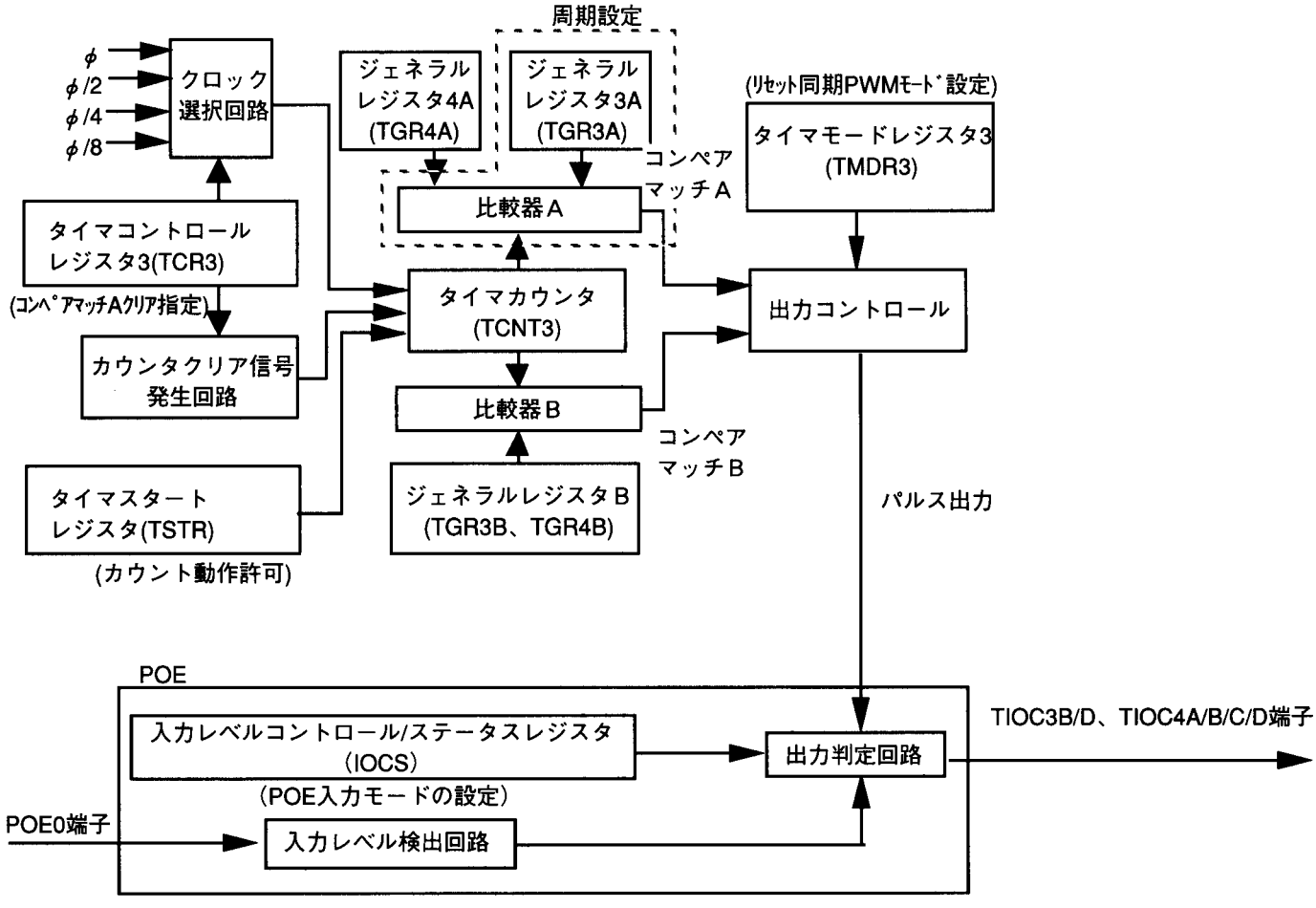


図2.8.2　MTU/ch3、4、POEブロック図

（2）表2.8.1に本タスクの機能割り付けを示します。MTU、POEの機能を割り付け、波形遮断を行います。

表2.8.1　機能割り付け

端子、 レジスタ名	機能割り付け
TIOC3B	パルス出力端子
TIOC3D	
TIOC4A	
TIOC4B	
TIOC4C	
TIOC4D	
POE0	遮断用外部信号入力端子
TSTR3	ch3のタイマカウンタ動作を許可/禁止
TCR3	ch3のタイマカウンタのクリア要因と入力クロックの選択
TMDR3	ch3、4をリセット同期PWMﾓｰﾄﾞに設定
TGR3A	PWM周期の設定
TGR3B	出力させる波形変化タイミングの設定
TGR3C	
TGR3D	
TGR4A	
TGR4B	
TOER	TIOC3B/D、TIOC4A/B/C/D端子のタイマ出力を許可/禁止
ICSR	POE入力モードの選択

図2.8.3に動作原理を示します。波形の遮断をハードウェアで自動的に行います。（リセット同期PWMの動作原理については本アプリケーションノートの2章5項の正相、逆相PWM3相出力を参考にしてください。）

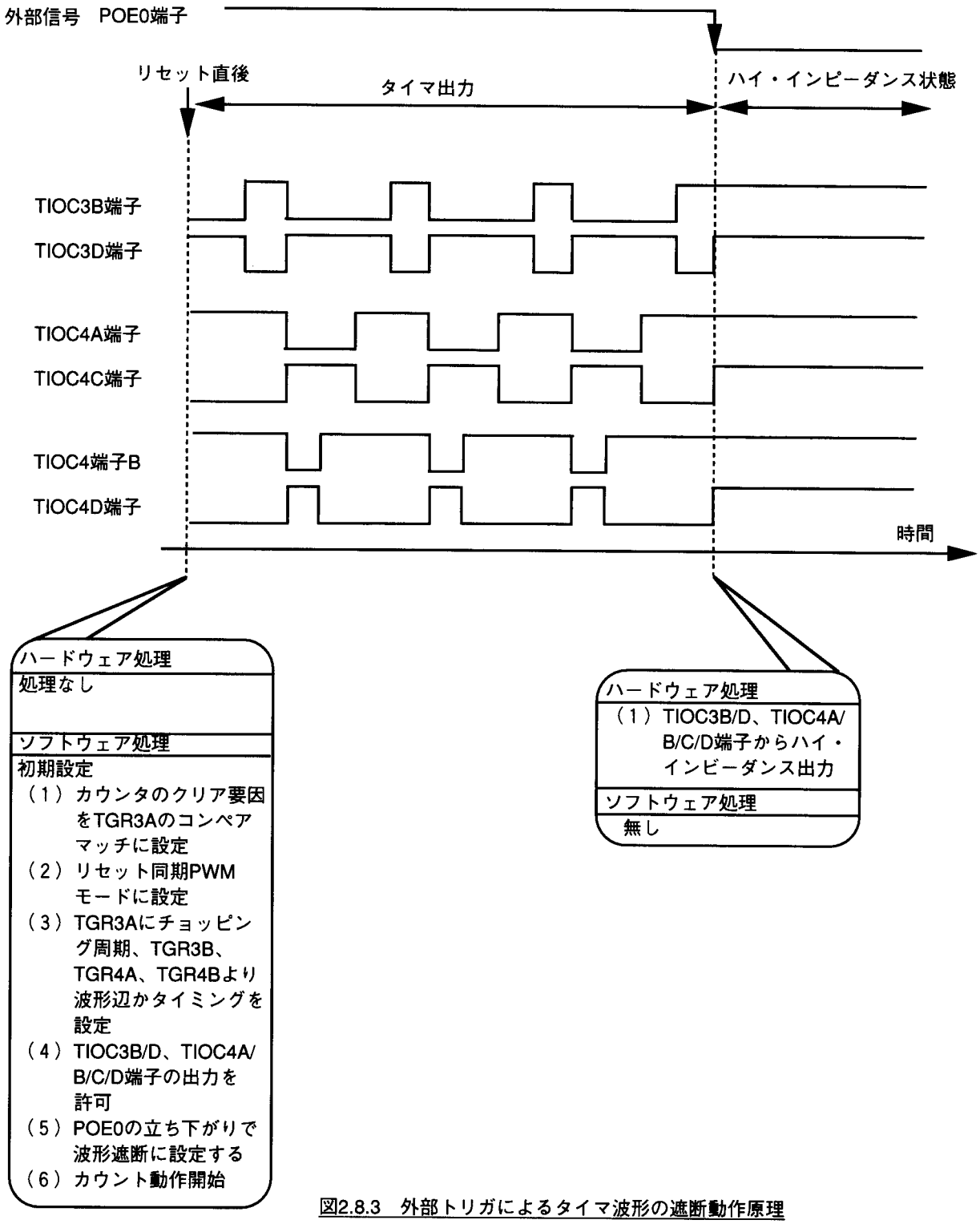


図2.8.3　外部トリガによるタイマ波形の遮断動作原理

外部トリガによるタイマ波形の遮断	MCU	SH7040	使用機能	MTU、POE

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	機能割り付け
メインルーチン	down	DCモータ制御用波形の生成

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
cycle	PWM周期を設定	1ワード	メインルーチン	入力
duk1	TIOC3B/Dから出力させる波形変化タイミングを設定			
duk2	TIOC4A/Cから出力させる波形変化タイミングを設定			
duk3	TIOC4B/Dから出力させる波形変化タイミングを設定			

（3）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
TMRSH.TSTR	タイマカウントの動作	メインルーチン
T3.TCR3	タイマカウンタのクリア要因と入力クロックの選択	
T3.TOCR	PWM周期に同期したトグル出力の許可と正相、逆相の出力レベルの設定	
T31.TGR3A	PWM周期を設定	
T31.TGR3B	TIOC3B、TIOC3Dから出力させる波形変化タイミングを設定	
T31.TGR4A	TIOC4A、TIOC4Cから出力させる波形変化タイミングを設定	
T31.TGR4B	TIOC4B、TIOC4Dから出力させる波形変化タイミングを設定	
T3.TOER	TIOC3B/D、TIOC4A/B/C/D端子をMTU出力に設定	
T3.TMDR3	リセット同期PWMモードの設定	
POE.ICSR	POE0端子入力信号の立ち下がりエッジに同期してハイ・インピーダンス出力に設定	
FPCE.PEIOR	TIOC3B/D、TIOC4A/B/C/D端子を出力に設定	
FPCE.PECR1	TIOC3B/D、TIOC4A/B/C/D端子をMTU出力に設定	

（4）使用ＲＡＭ説明

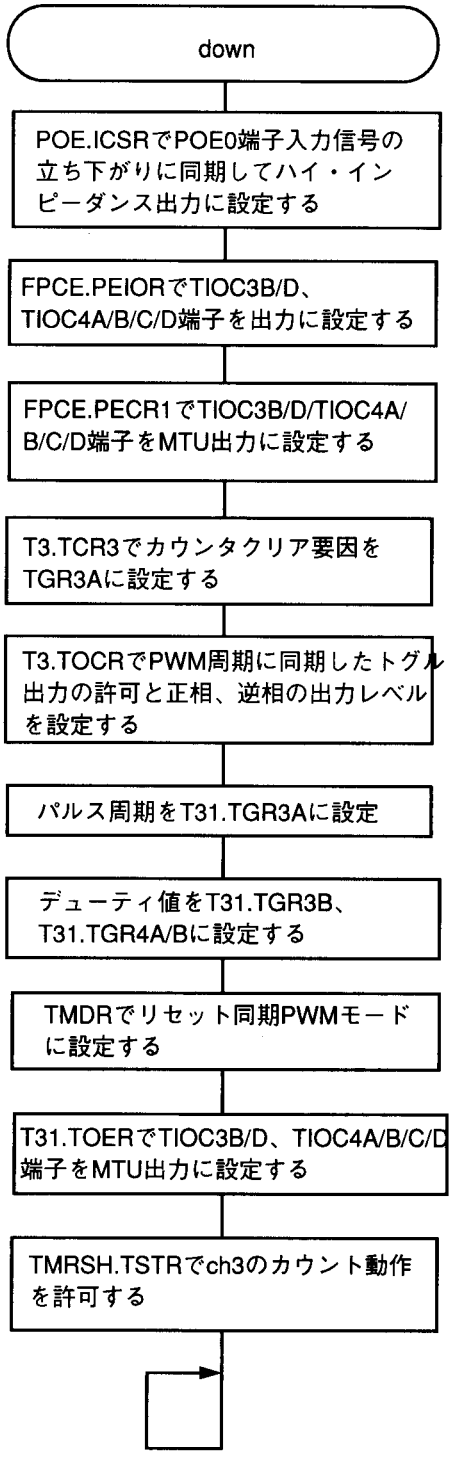
本タスク例では引数以外のＲＡＭは使用していません。

注）レジスタのラベル名はＳＨ７０４０ヘッダファイルの名前を使用しています。

外部トリガによるタイマ波形の遮断	MCU	SH7040	使用機能	MTU、POE

フローチャート

（1）メインルーチン



外部トリガによるタイマ波形の遮断	MCU	SH7040	使用機能	MTU、POE

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                              INCLUDEFILE                               */
/*------------------------------------------------------------------------*/
#include <machine.h>
#include "SH7040.H"
/*------------------------------------------------------------------------*/
/*                              PROTCOL                                   */
/*------------------------------------------------------------------------*/
void down(void);
/*------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                            */
/*------------------------------------------------------------------------*/
#define cycle    (*(unsigned short *)0xfffff000)
#define duk1     (*(unsigned short *)0xfffff002)
#define duk2     (*(unsigned short *)0xfffff004)
#define duk3     (*(unsigned short *)0xfffff006)
/*------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                              */
/*------------------------------------------------------------------------*/
void down(void)
{
    POE.ICSR = 0x00;            /* stop timer POE0 folling eddge */
    POE.OCSR = 0x00;
    PFCE.PEIOR = 0xfa00;        /* set output level */
    PFCE.PECR1 = 0x5544;        /* TIOCnx sellect */
    T0.TCR0 = 0x20;             /* timer clear input caputure TGRA0  */
    T0.TIOR0H = 0x88;           /* input caputure rising edge TIOC0A */
    T0.TIOR0L = 0x08;           /* input caputure rising edge TIOC0A */
    T3.TCR3 = 0x20;             /* timer clear input caputure TGRA3 */
    T3.TOCR = 0x00;             /* set output level */
    T31.TCNT3 = 0x0000;         /* set timer counter 0x0000 */
    T3.TMDR3 = 0xc8;            /* reset-synchoronized pwm mode */
    T31.TGR3A = cycle;          /* cycle set */
    T31.TGR3B = duk1;           /* width set */
    T31.TGR4A = duk2;
    T31.TGR4B = duk3;
    T3.TOER = 0xff;             /* set timer3,4 output */
    TMRSH.TSTR = 0xc0;          /* start timer3,4 */
    while(1);                   /* loop */
}
```

2.9 ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

仕様

（１）図2.9.1に示すように、DCブラシレスモータ制御に必要な波形を出力します。出力波形は、各端子のゲート信号とリセット同期PWM出力をチョッピングして出力します。

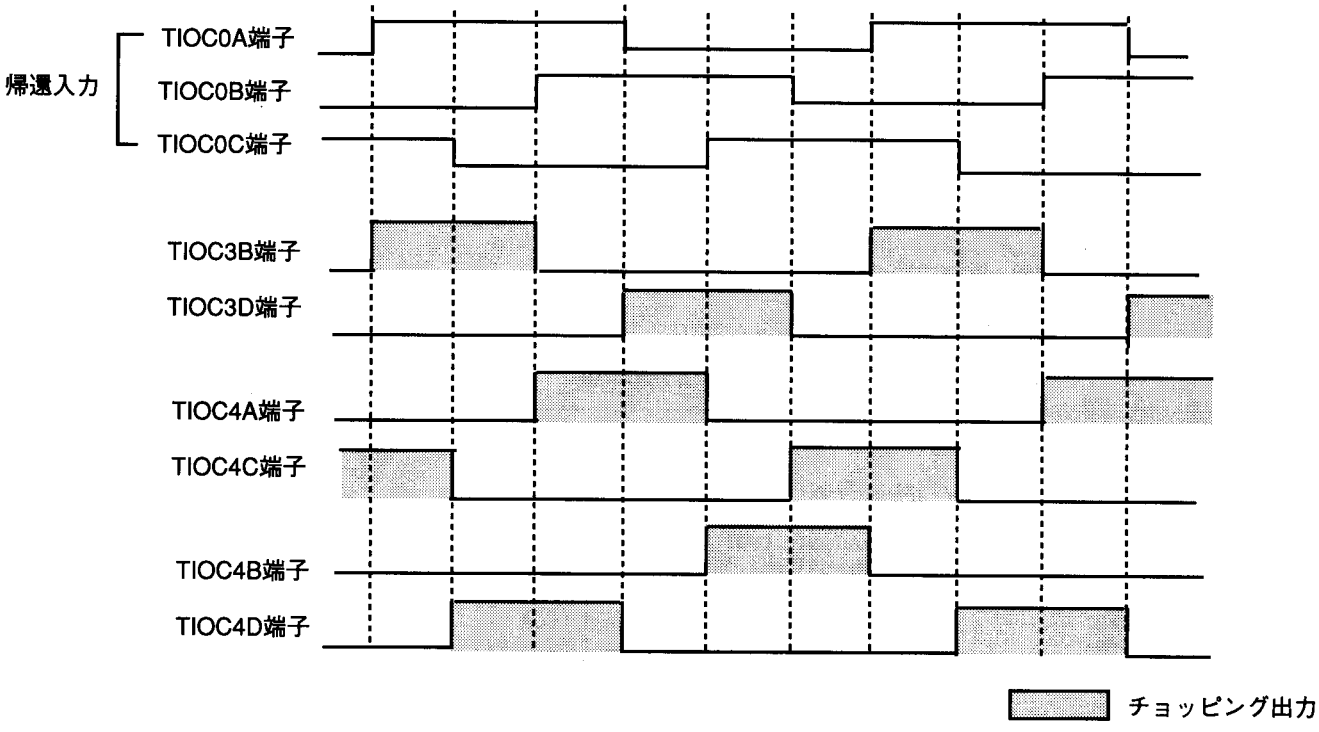


図2.9.1　DCブラシレスモータ制御用信号出力例

ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

使用機能説明

（１）本タスク例ではＭＴＵのチャネル3、4を組み合わせて使用し、一方の変化点が共通で、正相、逆相の関係にあるＰＷＭ波形を3相出力します。作成した波形と帰還入力から生成したゲート信号をチョッピングして出力します。

（ａ）図2.9.2に本タスク例で使用するＭＴＵのブロック図を示します。

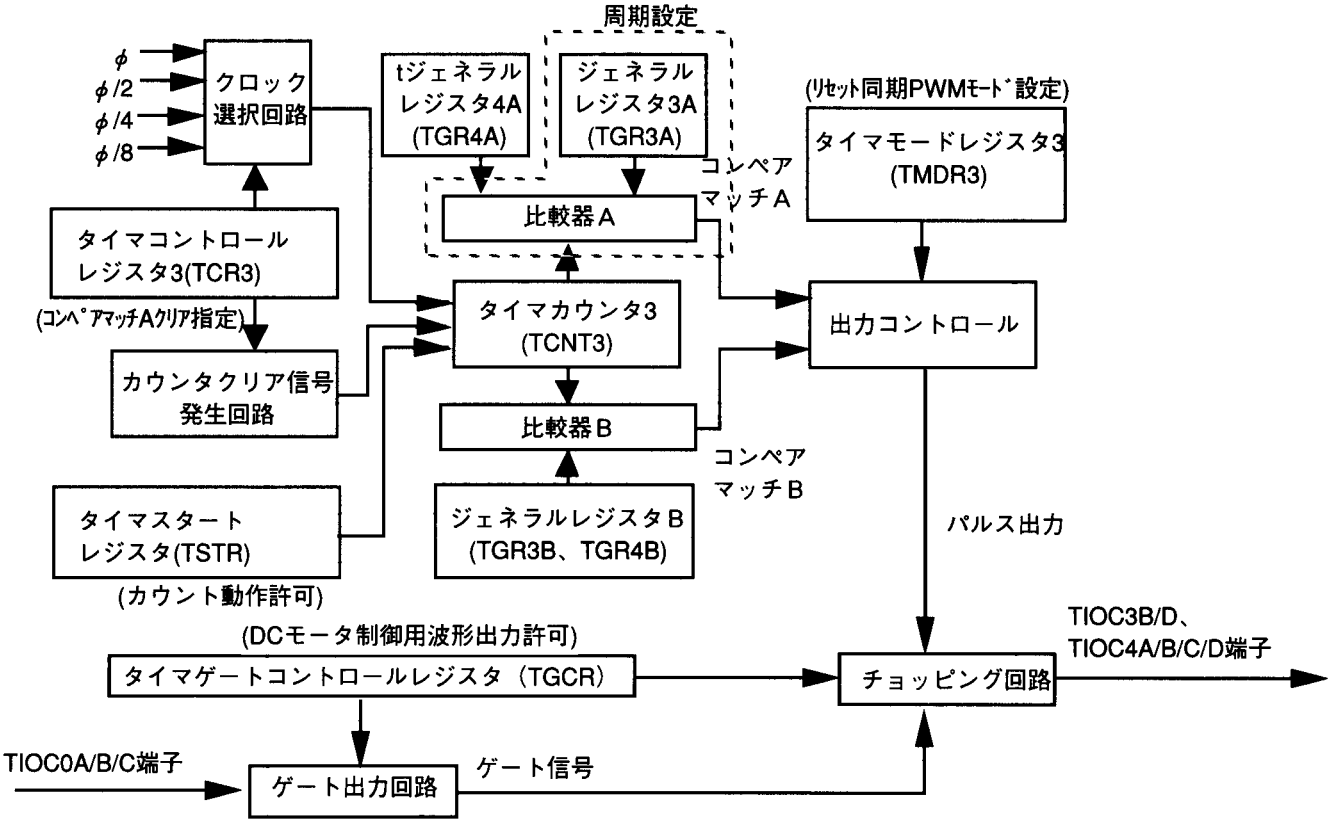


図2.9.2　MTU/ch3、4ブロック図

（２）表2.9.1に本タスクの機能割り付けを示します。ＭＴＵの機能を割り付け、DCモータ制御用波形の出力を行います。

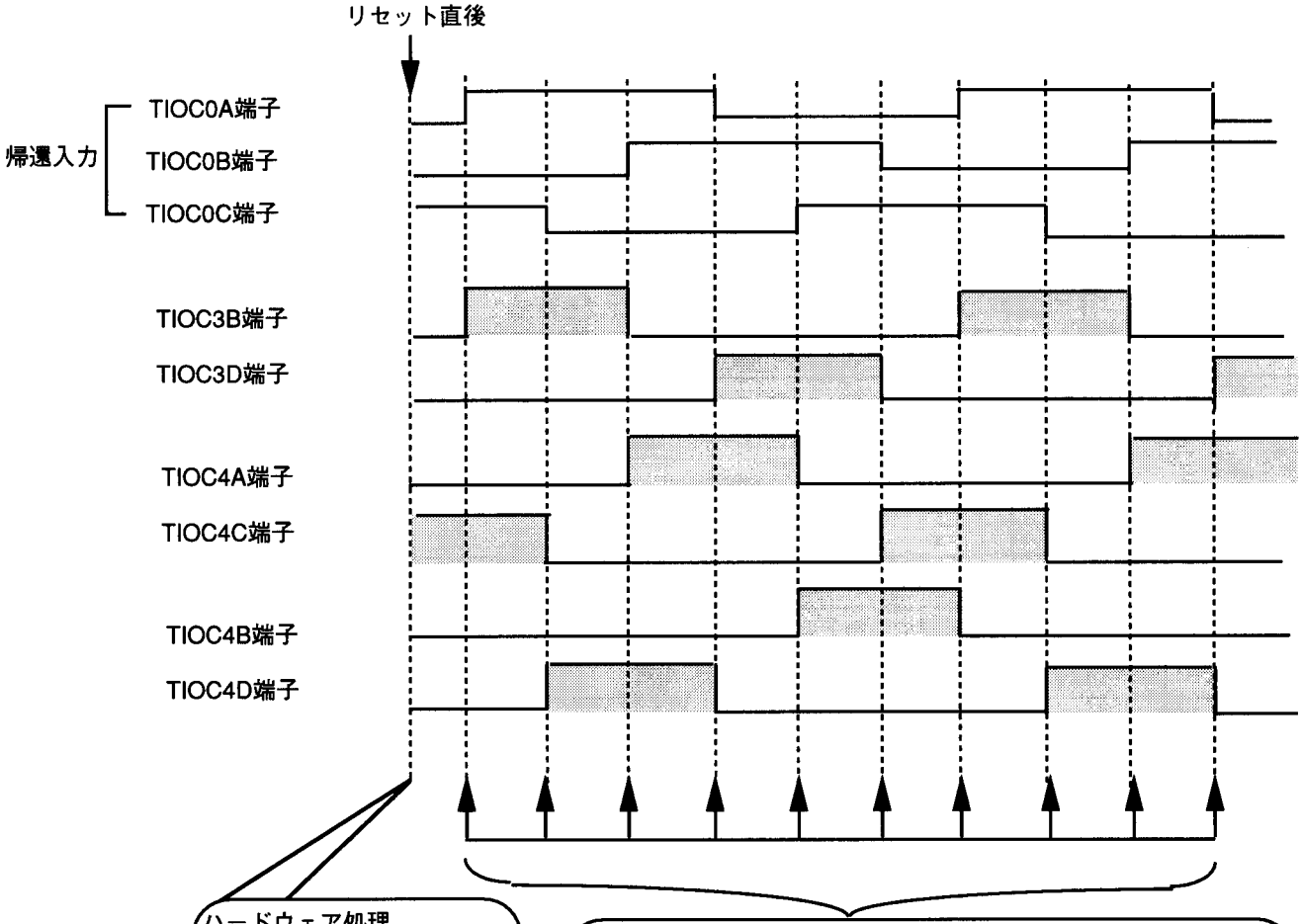
表2.9.1　機能割り付け

端子、 レジスタ名称	機能割り付け
TIOC3B	パルス出力端子
TIOC3D	
TIOC4A	
TIOC4B	
TIOC4C	
TIOC4D	
TIOC0A	帰還信号入力端子
TIOC0B	
TIOC0C	
TSTR	ch3、4のタイマカウンタ動作を許可/禁止
TCR3	ch3のタイマカウンタのクリア要因と入力クロックを選択
TMDR3	ch3、4をリセット同期PWMﾓｰﾄﾞとして動作
TGR3A	ＰWM周期の設定
TGR3B	出力させる波形変化タイミングの設定
TGR3C	
TGR3D	
TGR4A	
TGR4B	
TOER	TIOC3B/D、TIOC4A/B/C/D端子のタイマ出力を許可/禁止
TGCR	DCモータ制御波形の出力を許可/禁止

ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

動作説明

図2.9.3に動作原理を示します。DCモータ制御用波形の出力をハードウェアで自動的に行います。
（リセット同期PWMの動作原理については本アプリケーションノート2章5項の正相、逆相PWM3相出力を参考にしてください。）



ハードウェア処理
処理なし

ソフトウェア処理
初期設定
（１）カウンタのクリア要因をTGR3Aのコンペアマッチに設定
（２）リセット同期PWMモードに設定
（３）TGR3Aにチョッピング周期、TGR3B、TGR4A、TGR4Bより波形辺かタイミングを設定
（４）TIOC3B/D、TIOC4A/B/C/D端子の出力を許可
（５）DCモータ制御用ゲート信号出力の許可
（６）カウント動作開始

ハードウェア処理
（１）TIOC0A/B/Cの立ち上がり/立ち下がりエッジでTIOC3B/D、TIOC4A/B/C/D各出力端子にゲート信号を出力する
（２）出力レベルがHigh時リセット同期PWM出力とゲート出力をチョッピングして出力する

ソフトウェア処理
無し

図2.9.3　ＤＣモータ制御用信号出力1動作原理

ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

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	機能割り付け
メインルーチン	dc_3out	DCモータ制御用波形の生成

（２）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
cycle	PWM周期の設定	１ワード	メインルーチン	入力
duk1	TIOC3B/Dから出力させる波形変化タイミングを設定			
duk2	TIOC4A/Cから出力させる波形変化タイミングを設定			
duk3	TIOC4B/Dから出力させる波形変化タイミングを設定			

（３）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
FPCE.PEIOR	TIOC3B/D、TIOC4A/B/C/D端子を出力に設定	メインルーチン
FPCE.PECR1	TIOC3B/D、TIOC4A/B/C/D端子をMTU出力に設定	
TMRSH.TSTR	タイマカウントの動作/停止の設定	
T3.TCR3	タイマカウンタのクリア要因と入力クロックを選択	
T3.TOCR	PWM周期に同期したトグル出力の許可と正相、逆相の出力レベルの設定	
T31.TGR3A	PWM周期の設定	
T31.TGR3B	TIOC3B、TIOC3Dから出力させる波形変化タイミングの設定	
T31.TGR4A	TIOC4A、TIOC4Cから出力させる波形変化タイミングの設定	
T31.TGR4B	TIOC4B、TIOC4Dから出力させる波形変化タイミングの設定	
T3.TOER	TIOC3B/D、TIOC4A/B/C/D端子をMTU出力に設定	
T3.TMDR3	リセット同期PWMモードの設定	
T3.TGCR	DCモータ制御用波形出力を許可	

（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。

注）レジスタのラベル名はＳＨ７０４０ヘッダファイルの名前を使用しています。

ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

フローチャート

（1）メインルーチン

ＤＣモータ制御用信号出力	ＭＣＵ	ＳＨ７０４０	使用機能	ＭＴＵ（ゲートコントロールレジスタ）

プログラムリスト

```
/*-----------------------------------------------------------------------*/
/*                              INCLUDE FILE                             */
/*-----------------------------------------------------------------------*/
#include <machine.h>
#include "SH7040.H"
/*-----------------------------------------------------------------------*/
/*                                PROTOTYPE                              */
/*-----------------------------------------------------------------------*/
void dc_3(void);
/*-----------------------------------------------------------------------*/
/*                             RAM ALLOCATION                            */
/*-----------------------------------------------------------------------*/
#define cycle   (*(unsigned short *)0xfffff000)
#define duk1    (*(unsigned short *)0xfffff002)
#define duk2    (*(unsigned short *)0xfffff004)
#define duk3    (*(unsigned short *)0xfffff006)
/*-----------------------------------------------------------------------*/
/*                             MAIN PROGRAM                              */
/*-----------------------------------------------------------------------*/
void dc_3(void)
{
    PFCE.PEIOR = 0xfa00;        /* set output level */
    PFCE.PECR1 = 0x5544;        /* TIOCnx sellect */
    PFCE.PECR1 = 0x0015;        /* TIOCnx sellect */

    T0.TCR0 = 0x00;             /* timer clear input caputure TGRA0  */
    T3.TCR3 = 0x20;             /* timer clear input caputure TGRA3 */
    T3.TOCR = 0x00;             /* set output level */
    T31.TCNT3 = 0x0000;         /* set timer counter 0x0000 */
    T3.TMDR3 = 0xc8;            /* reset-synchoronized pwm mode */
    T31.TGR3A = cycle;          /* cycle set */
    T31.TGR3B = duk1;           /* width set */
    T31.TGR4A = duk2;
    T31.TGR4B = duk3;
    T3.TOER = 0xff;             /* set timer3,4 output */
    T3.TGCR = 0x70;             /* set DC motor output */
    TMRSH.TSTR = 0xc1;          /* start timer0,3,4 */
    while(1);                   /* loop */

}
```

2.10 MTUによるA/D変換の起動及び変換結果の格納	MCU	SH7040	使用機能	DTC（ブロック転送）、A/D変換器

仕様

（1）図2.10.1に示すように、8chの電圧を入力し、A/D変換した結果をDTCでRAMに格納します。
（2）A/D変換はグループモード/シングルに設定し2chの同時サンプルを行う。
（3）A/D変換の起動はMTUのTGR0Aのコンペアマッチで行います。
（4）DTCのブロック転送は、A/Dの終了割り込みで行います。

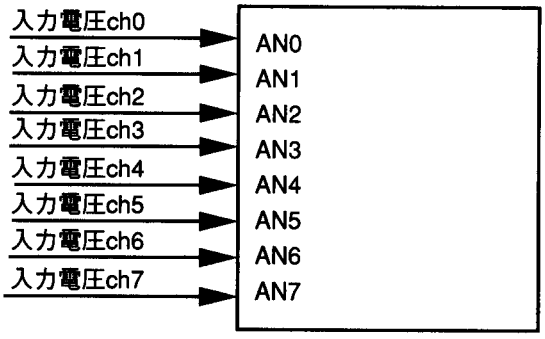


図2.10.1　SH7040による電圧の測定ブロック図

使用機能説明

（1）本タスク例では、MTUのコンペアマッチでA/D変換を起動し、変換結果をDTCでRAMに格納します。

（a）図2.10.2にch0のブロック図を示します。ch0では、以下の機能を使用してA/D変換の起動をします。

・ソフトウェアを介さずMTUのコンペアマッチでA/D変換を起動させる機能。
・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）

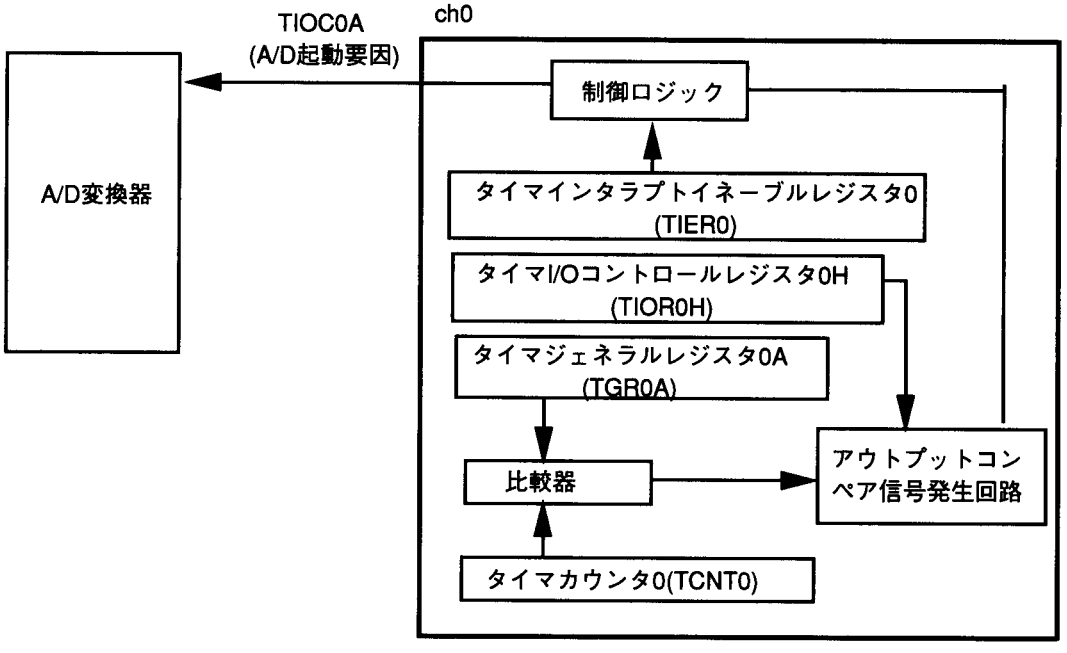
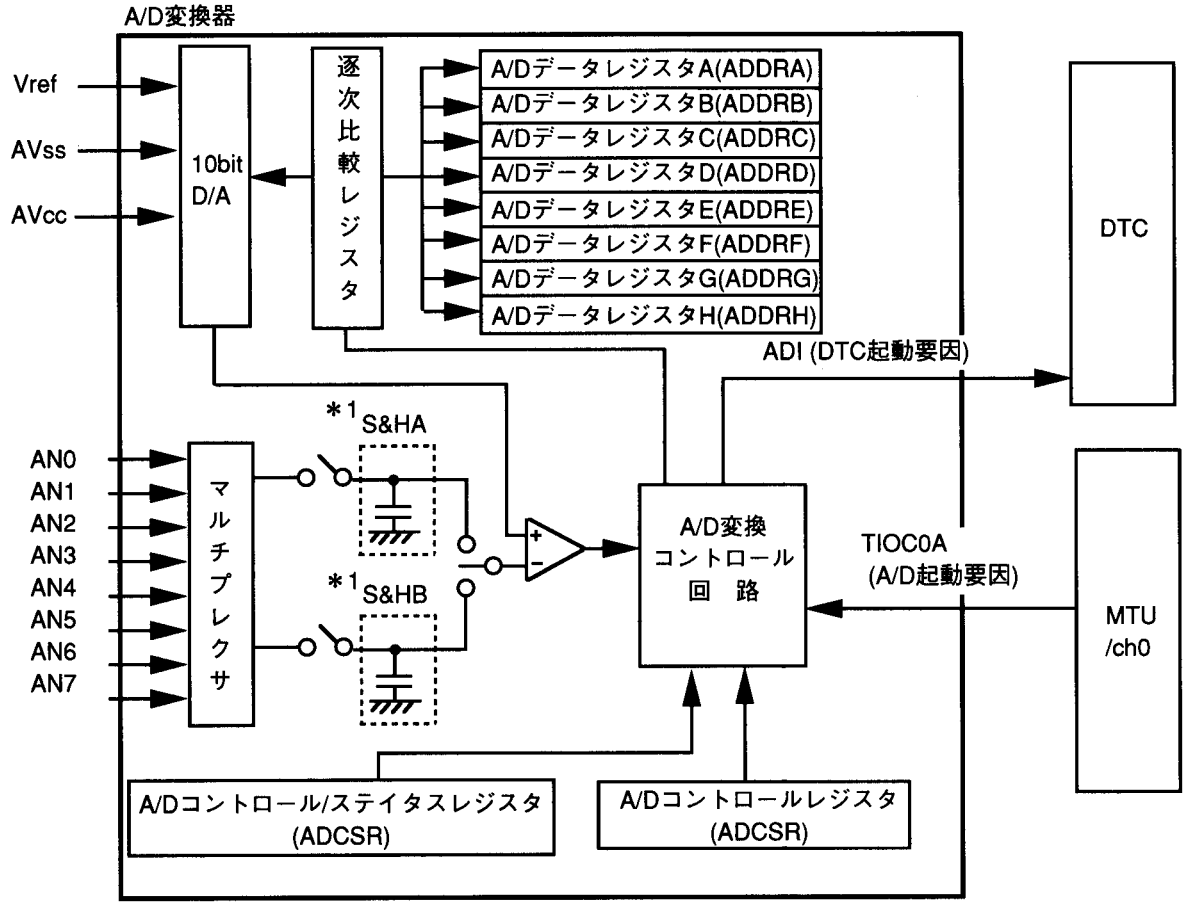


図2.10.2　SH7040によるch0のブロック図

（b）図2.10.3にA/D変換器のブロック図を示します。A/D変換器では、以下の機能を使用してアナログからデジタルへの変換を行っています。また、A/D変換終了時DTCを起動させます。

・複数のチャネル（chA～chH）のA/D変換を1度行う機能。（グループ・シングルモード）
・2チャンネルの入力電圧を同時にサンプルし連続変換する機能。（同時サンプリング）
・A/D変換終了時にＤＴＣを起動させる機能。



＊1　：サンプル＆ホールド回路

図2.10.3　SH7040による電圧の測定ブロック図

（c）図2.10.4にDTCのブロック図を示します。DTCでは、以下の機能を使用してデータの転送を行っています。

・1回の起動要因で1ブロックのデータを転送する機能。（ブロック転送機能）

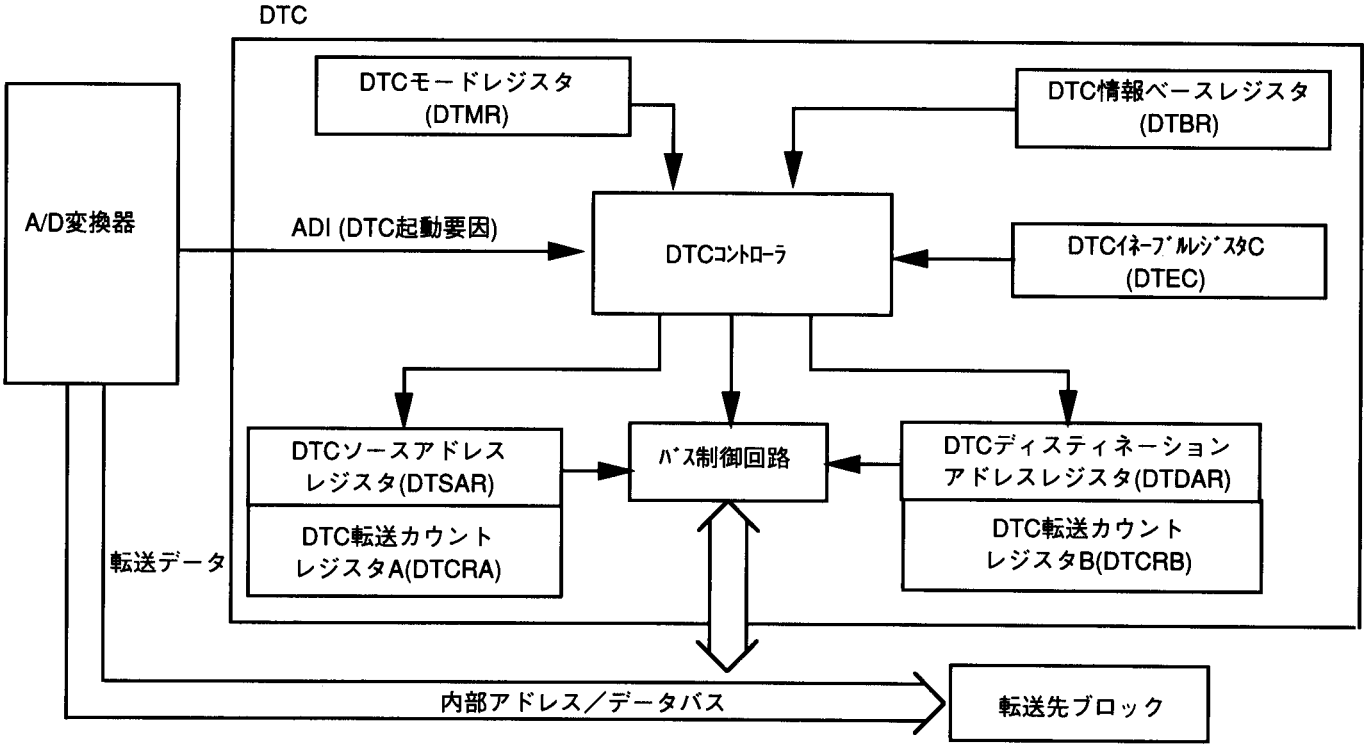


図2.10.4　SH7040によるDTCブロック図

（2）表2.10.1に本タスク例の機能割り付けを示します。

表2.10.1 レジスタ機能割り付け

レジスタ名	機能割り付け
A0～A7	アナログ測定端子
TCR0	カウンタクリア要因の選択
TIOR0H	インプットキャプチャ信号の入力エッジの選択
TIER0	A/D変換開始要求の発生を許可
TGR0A	サンプリング周期の設定
ADCR	A/D変換のモード及び測定端子の設定
ADCSR	変換時間及び起動要因の選択
ADDRA～ ADDRH	A/D変換結果の格納
DTMR	DTCをブロック転送モードに設定
DTCRA	転送回数の指定
DTCRB	ブロック長の指定
DTSAR	転送元アドレスの設定
DTDAR	転送先アドレスの設定
DTBR	DTCベクタ上位16bitの設定
DTEC	A/D変換終了時にDTCの起動を許可

MTUによるA/D変換の起動及び変換結果の格納	MCU	SH7040	使用機能	DTC（ブロック転送）、A/D変換器

動作説明

図2.10.5に動作原理を示します。図に示すように、TGR0Aのコンペアマッチによって A/D変換が起動され、AN0～AN7への入力電圧は2ch同時サンプルを行いAN0～AN7を順次変換します。指定した全てのchが変換終了後DTCが起動され、A/Dの変換結果をRAMに転送します。

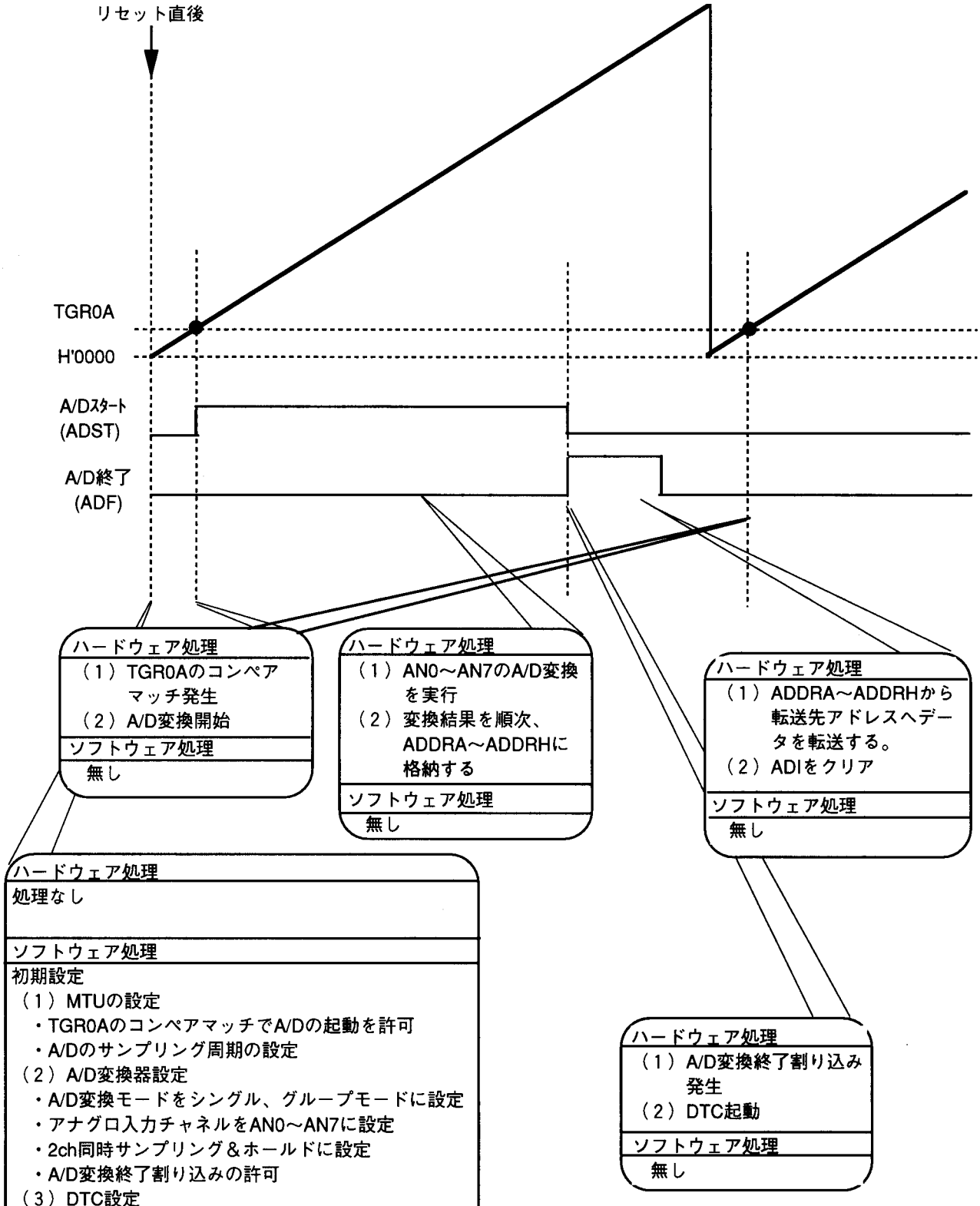


図2.10.5　パルス幅測定動作原理

動作説明

図2.10.6にDTC起動時の動作原理を示します。DTCを実行する場合、起動要因発生前以下の設定を行なってください。
・レジスタ情報の設定。（表2.11.2にブロック転送モード時のレジスタ情報設定例を示します。）
・DTCのベクタテーブルにレジスタ情報を設定した先頭アドレスの下位16bitを設定します。
・DTC情報ベースレジスタにジスタ情報を設定した先頭アドレスの上位16bitを設定します。

図2.11.6に示すようにDTC起動時DTCのベクタテーブルからレジスタ情報の先頭アドレスの下位16bitを読み込みます。読み込んだ値とDTC情報ベースレジスタ（レジスタ情報の先頭アドレスの上位16bit格納）からレジスタ情報先頭アドレスを生成します。レジスタ情報先頭アドレスからレジスタ情報を順次'読み込み転送動作を開始します。

表2.11.2　レジスタ情報一覧

設定アドレス	レジスタ名	データ長
RF	DTCモードレジスタ(DTMR)	ワード
RF+2	DTC転送カウントレジスタA(DTCRA)	ワード
RF+6	DTC転送カウントレジスタB(DTCRB)	ワード
RF+8	DTCソースアドレスレジスタ(DTSAR)	ロングワード
RF+12	DTCディスティネーションアドレスレジスタ(DTDAR)	ロングワード

RF：レジスタ情報の先頭アドレス

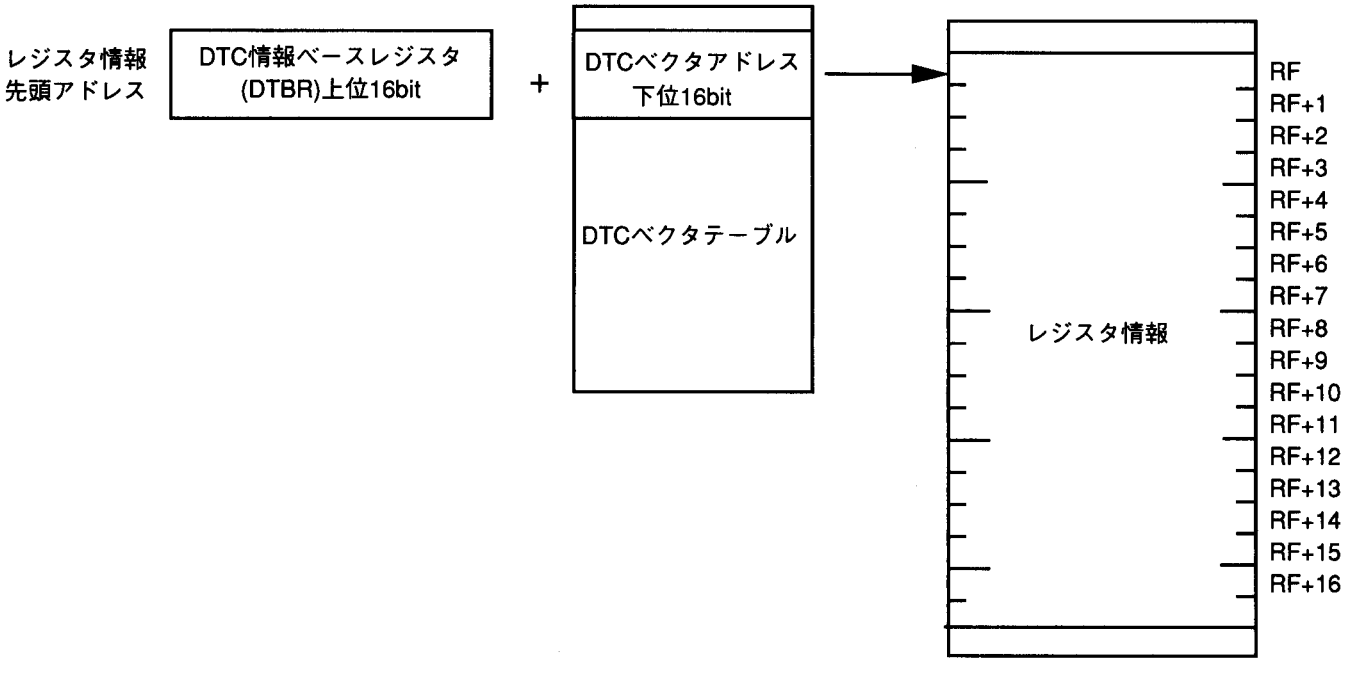


図2.10.6　DTC起動時の動作原理

MTUによるA/D変換の起動及び変換結果の格納	MCU	SH7040	使用機能	DTC（ブロック転送）、A/D変換器

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	機能割り付け
メインルーチン	adscan	MTUによるA/D変換器の起動及びA/D変換終了時のDTCの起動に設定

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
transsadr	DTCの転送先アドレスの設定 8チャネルのA/D変換結果はtranssadrを先頭にデータ長ワード単位で8データ格納 各変換結果は10bitの変換結果は以下のように設定	ロングワード	メインルーチン	入力

上位ﾊﾞｲﾄ							AD9	AD8
下位ﾊﾞｲﾄ	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

（3）使用内部レジスタ説明

レジスタ名	機能割り付け	使用モジュール名
TMRSH0.TSTR	タイマカウンタの動作／禁止の設定	メインルーチン
T0.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アウトプットコンペアAに設定	
T0.TIOR0H	TGR0Aをアウトプットコンペアに設定	
T0.TIER0	A/D変換開始要求の発生を許可	
T0.TGRA0	A/D変換サンプリング周期を設定	
A__D.ADCR	A/D変換モード(シングルモード）、同時サンプリング動作、起動要因をMTUの変換開始トリガに設定	
A__D.ADCSR	A/D変換チャネル(グループモード）、変換チャネル(AN0～AN7)、変換時間の設　　A/D変換終了割り込みを許可	
A__D.ADDRA～A__D.ADDRH	A/D変換結果の格納	—
DTC.DTMR	DTCをブロック転送モードに設定	メインルーチン
DTC.DTCRA	転送回数を1回に設定	
DTC.DTCRB	ブロック長8の指定	
DTC.DTSAR	転送元アドレスにADDRAを設定	
DTC.DTDAR	転送先アドレスにRAMを設定	
DTC.DTBR	DTCベクタ上位16bitを設定	
DTC.DTEC	A/D変換終了時にDTCの起動を許可	
PFCE.PECR2	ＴＩＯＣ０Ａからのパルス入力を許可	
INTC.IPRG	A/D変換終了割り込みレベルを15に設定	

（4）使用ＲＡＭ説明

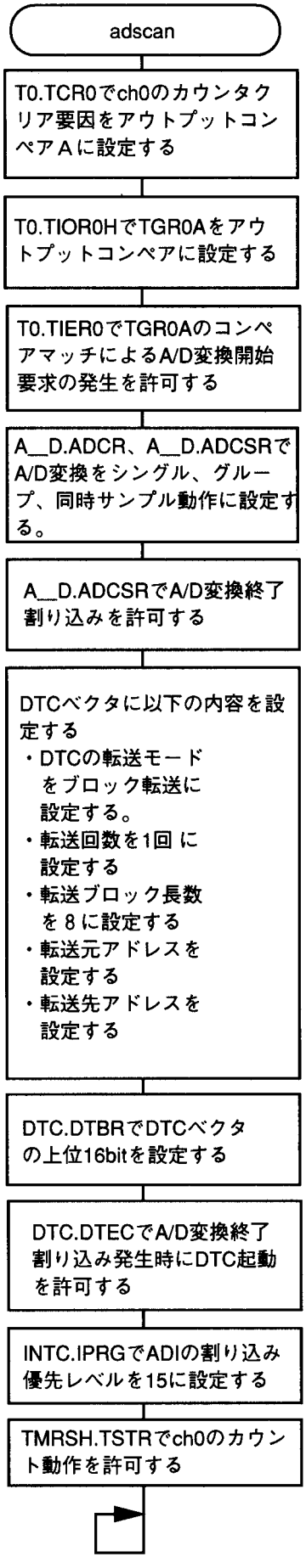
本タスク例では引数以外のＲＡＭは使用していません。

注）レジスタのラベル名はSH7040ヘッダファイルの名前を使用しています。

MTUによるA/D変換の起動及び変換結果の格納	MCU	SH7040	使用機能	DTC（ブロック転送）、A/D変換器

フローチャート

（1）メインルーチン



MTUによるA/D変換の起動及び変換結果の格納	MCU	SH7040	使用機能	DTC（ブロック転送）、A/D変換器

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                              INCLUDEFILE                               */
/*------------------------------------------------------------------------*/
#include <machine.h>
#include "SH7040.H"
/*------------------------------------------------------------------------*/
/*                              PROTOTYPE                                 */
/*------------------------------------------------------------------------*/
void a_d(void);
/*------------------------------------------------------------------------*/
/*                            RAM ALLOCATION                              */
/*------------------------------------------------------------------------*/
#define DTMR    (*(unsigned short *)0xfffff000)
#define DTCRA   (*(unsigned short *)0xfffff002)
#define DTCRB   (*(unsigned short *)0xfffff006)
#define DTSAR   (*(unsigned long *)0xfffff008)
#define DTDAR   (*(unsigned long *)0xfffff00c)
#define dtmr1   (*(unsigned short *)0xfffff010)
#define dtcra1  (*(unsigned short *)0xfffff012)
#define dtcrb1  (*(unsigned short *)0xfffff016)
#define dtsar1  (*(unsigned long *)0xfffff018)
#define dtdar1  (*(unsigned long *)0xfffff01c)
/*------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                              */
/*------------------------------------------------------------------------*/
void a_d(void)
{
    T0.TCR0 = 0x20;                 /* timer clear output compare TGR0A */
    T0.TIOR0H = 0x00;
    T0.TIER0 = 0x80;                /* interrupt TGI00A */
    T0.TGR0A = 0x1000;              /* set get A/D cycle */
    A_D.ADCR = 0x14;                /* set sampling */
    A_D.ADCSR = 0x4f;               /* set mode single/group */
    DTMR = dtmr1;                   /* set transmission mode DTC */
    DTCRA = dtcra1;                 /* set transmission frequency */
    DTCRB = dtcrb1;                 /* set transmission block frequency */
    DTSAR = dtsar1;                 /* set transmission address */
    DTDAR = dtdar1;                 /* set transmission address */
    DTC.DTBR = 0xffff;              /* set DTC data bese register*/
    DTC.DTEC = 0x40;                /* set a/d end start DTC*/
    INTC.IPRG = 0x0f00;             /* set interrupt level=15 */
    set_imask(0x0);                 /* set imask level=0*/
    TMRSH.TSTR = 0x01;              /* TCNT0 start */
    while(1);                       /* loop */
}
```

注）DTCベクタテーブルの0x422番地（DTC起動条件がA/D変換終了時のベクタ）にレジスタ情報先頭アドレスの下位16bitを設定してください。

2.11　DMACを使用したRAMモニタ	MCU	SH7040	使用機能	SCI、DMAC

仕様

（1）図2.11.1に示すように、SH7040のSCIを調歩同期式モードで使用し、コンソールから送信されたRAM参照アドレス（4バイトデータ）を受信し、その内容をRAM上から取りだしSCIでコンソールに送信します。
（2）転送プロトコルは9600bps、8ビットデータ、1ストップビット及びノンパリティとします。
（3）RDRからRAMへのデータ転送は図2.12.2に示すようにDMACの直接アドレスモードを使用し、RDRの受信データをRAM上に格納します。
（4）RAMからTDRへのデータ転送は図2.11.3に示すようにDMACの間接アドレスモードを使用し、以下のように行います。
　（a）RAM上に格納されたデータをDMAC内のﾃﾝﾎﾟﾗﾘﾊﾞｯﾌｧに格納し、それをアドレスとしてRAM上からデータを取り出します。
　（b）取り出したデータをTDRにバイト単位で順次転送します。
（5）DMACの転送条件は表2.11.1、表2.11.2に示す通りです。

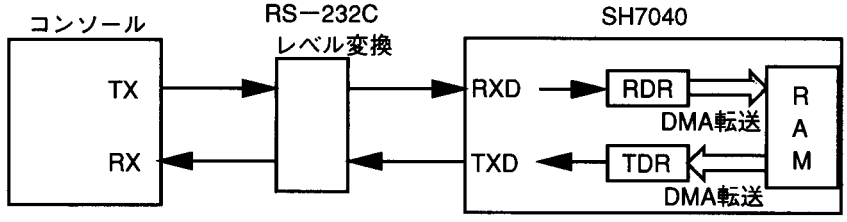


図2.11.1　SH7040によるRAM上データのSCI転送ブロック図

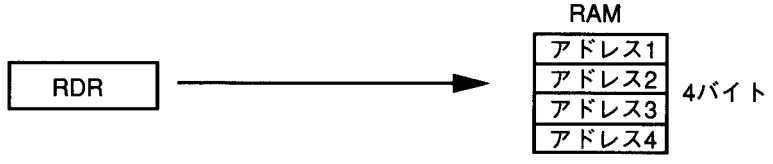


図2.11.2　DMACを用いたデータ転送（転送元直接アドレス）

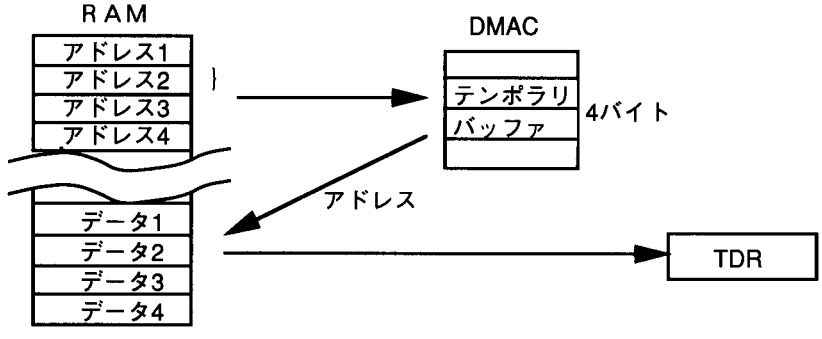


図2.11.3　DMACを用いたデータ転送（転送元間接アドレス）

表2.11.1　SCI受信時のDMAC転送条件（RDR→RAM）

条件項目	内容
DMACﾁｬﾈﾙ	チャネル0
転送元	内蔵SCIﾁｬﾈﾙ0
転送先	内蔵RAM
転送回数	4回
転送元アドレス	固定
転送先アドレス	増加
転送要求元	SCIﾁｬﾈﾙ0
バスモード	サイクルスチール
転送単位	バイト

表2.11.2　SCI送信時のDMAC転送条件（RAM→TDR）

条件項目	内容
DMACﾁｬﾈﾙ	チャネル3
転送元	内蔵RAM
転送先	内蔵SCIﾁｬﾈﾙ0
転送回数	4回
転送元アドレス	増加
転送先アドレス	固定
転送要求元	SCIﾁｬﾈﾙ0
バスモード	サイクルスチール
転送単位	バイト

使用機能説明

（1）本タスク例では、RAMモニタをSCIとDMACを使用して行います。

（a）図2.11.4に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コンソール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）

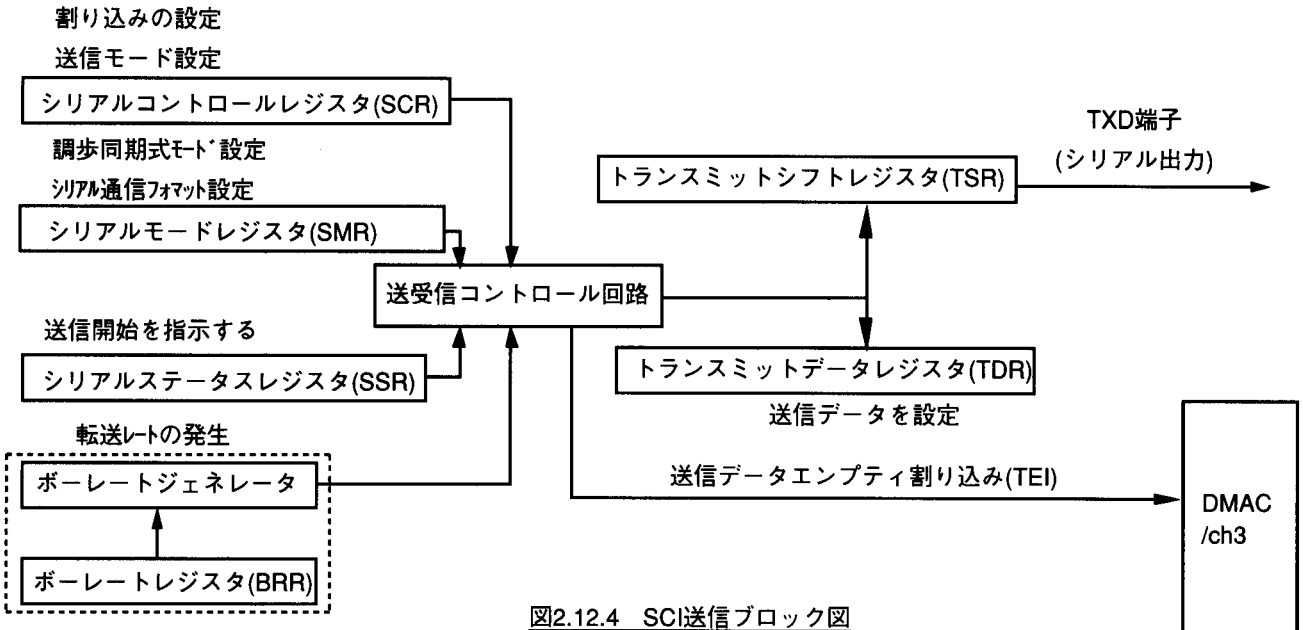


図2.12.4　SCI送信ブロック図

（b）図2.11.5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。本タスクでは図2.11.5に示すSCIの以下の機能を使用してコンソール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）

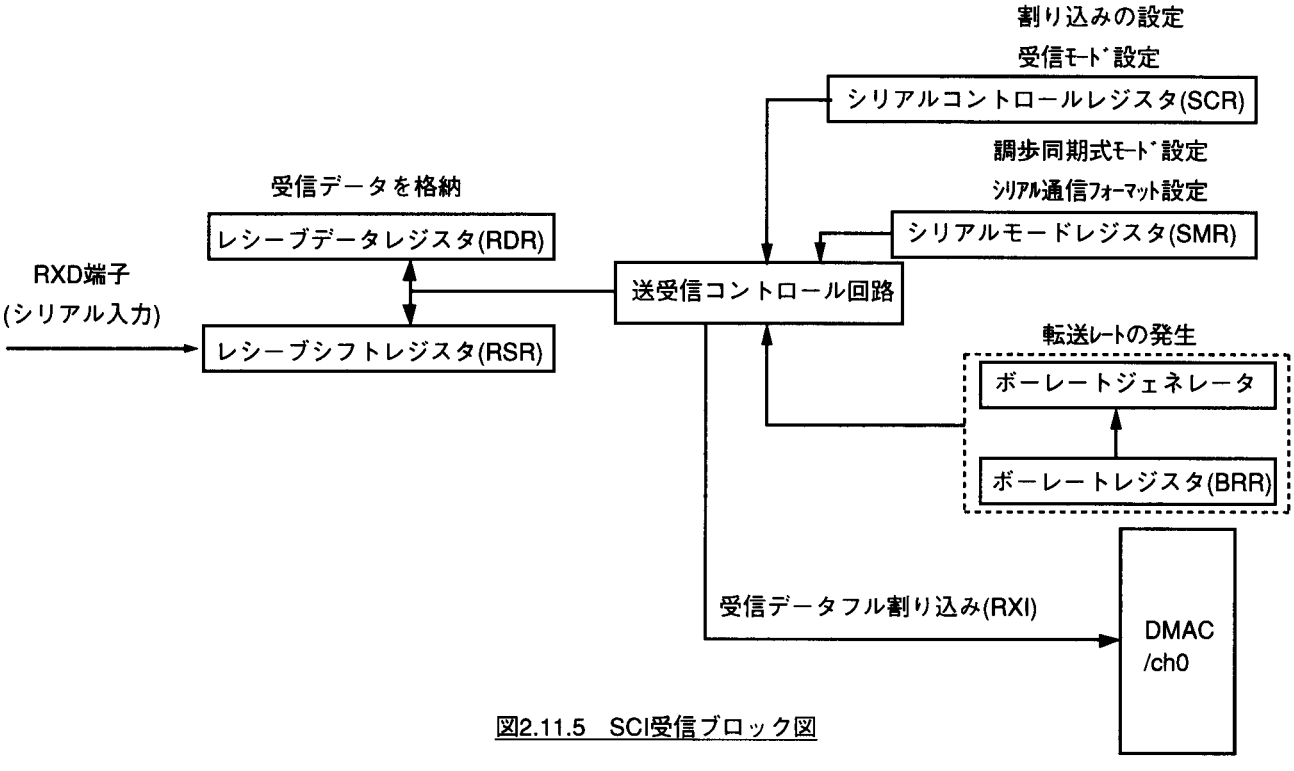


図2.11.5　SCI受信ブロック図

（C）図2.11.6に本タスク例で使用するDMAC/ch0のブロック図を示します。本タスク例はDMAC/ch0の以下の機能を使用してブロック転送を行います。

・DMAC起動時に転送元、転送先ともアドレスのデータを直接転送する機能（直接アドレス転送モード）

・SCIによりDMACを起動する機能

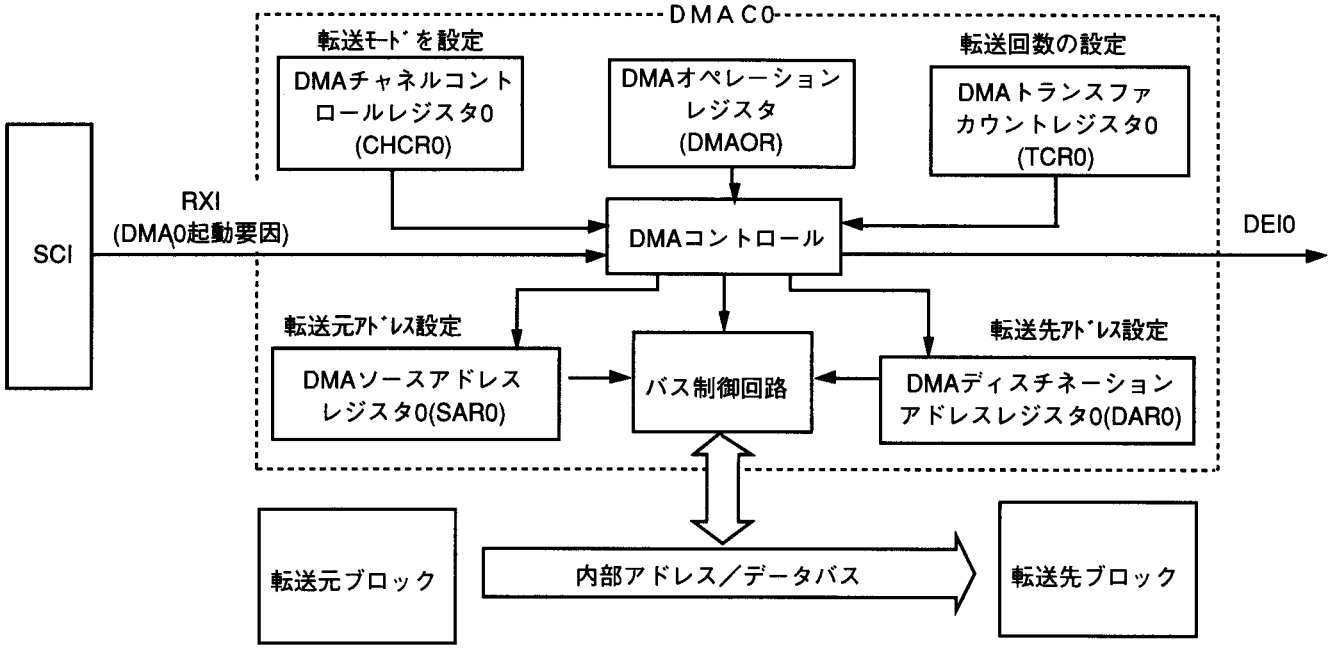


図2.11.6　DMA/ch0ブロック図

（D）図2.11.7に本タスク例で使用するDMAC/ch3のブロック図を示します。本タスク例はDMAC/ch3の以下の機能を使用してブロック転送を行います。

・DMAC起動時に転送元レジスタに格納されているデータを転送する機能。（間接アドレス転送モード）

・SCIによりDMACを起動する機能。

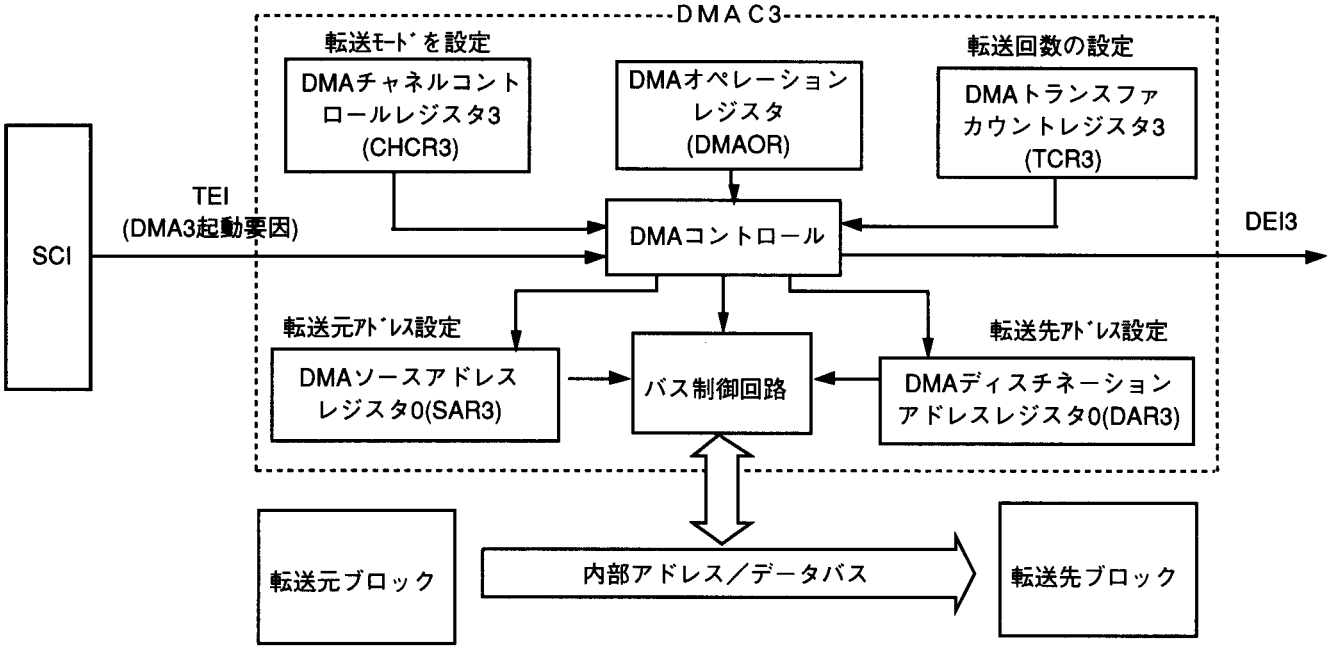


図2.11.7　DMA/ch3ブロック図

使用機能説明

表2.11.1に本タスク例の機能割付を示します。DMAC及びSCIの機能を割付、SCIによるデータ転送の送受信を行います。

表2.11.1　機能割付

端子、レジスタ名	機能割り付け
SAR0 SAR3	転送元アドレスの設定。
DAR0 DAR3	転送先アドレスの設定。
TCR0 TCR3	転送回数の設定。
CHCR0 CHCR3	DMACの動作モード、転送方法等の設定。
DMAOR	DMACの実行するチャネルの優先順位の設定。
RXD	ﾃﾞｰﾀ受信端子。
TXD	ﾃﾞｰﾀ送信端子。
SMR	SCIの送信フォーマットの設定。
SCR	SCIの割り込みの許可／禁止の設定。
SSR	割り込みステータスの設定。
RDR	コンソールから受信ﾃﾞｰﾀを格納。
TDR	コンソールへ送信ﾃﾞｰﾀを格納。
BRR	転送ﾚｰﾄの設定。

図2.11.9に動作原理を示します。図2.12.9に示すように、SH7040のハードウェア処理及びソフトウェア処理により
SCIによるデータ転送の送受信を行います。

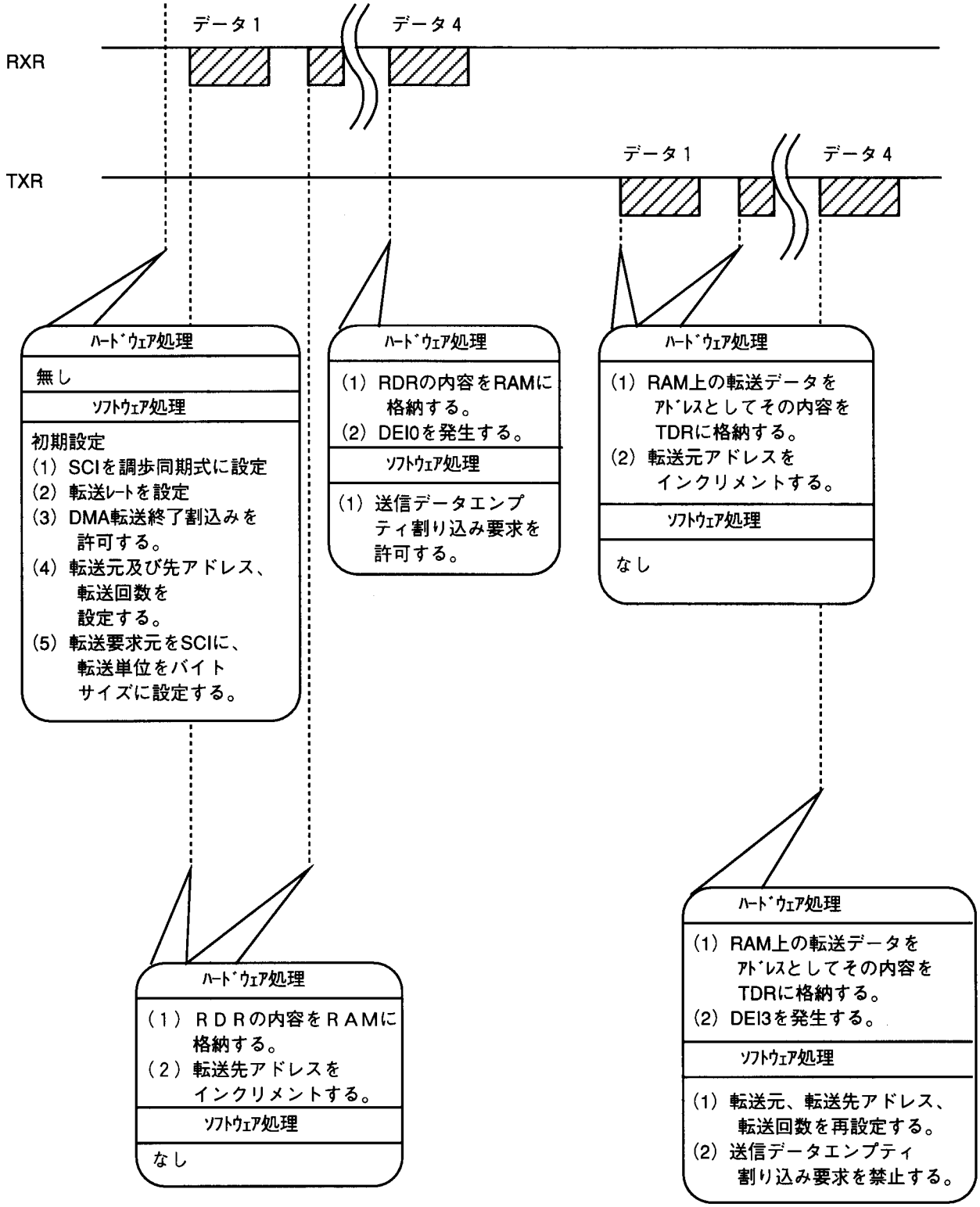


図2.11.8　SCIによるデータ転送動作原理

DMACを使用したRAMモニタ	MCU	SH7040	使用機能	SCI、DMAC

ソフトウェア説明

（1）モジュール説明

ﾓｼﾞｭｰﾙ名	ﾗﾍﾞﾙ名	機能割り付け
メインルーチン	rammon	SCI及びDMACの初期設定を行なう。
受信データ転送	dma_rdr	DEI0で起動し、送信データエンプティ割り込み要求を許可する。
送信データ転送	dma_tdr	DEI3で起動し、転送元、転送先アドレス、転送回数を再設定する。 送信データエンプティ割り込み要求を禁止する。

（2）引数の説明

ラベル名、レジスタ名	機能割り付け	データ長	使用モジュール名	入出力
data0	転送アドレスを格納	ロング ワード	メインルーチン	入力

（3）使用内部レジスタ説明

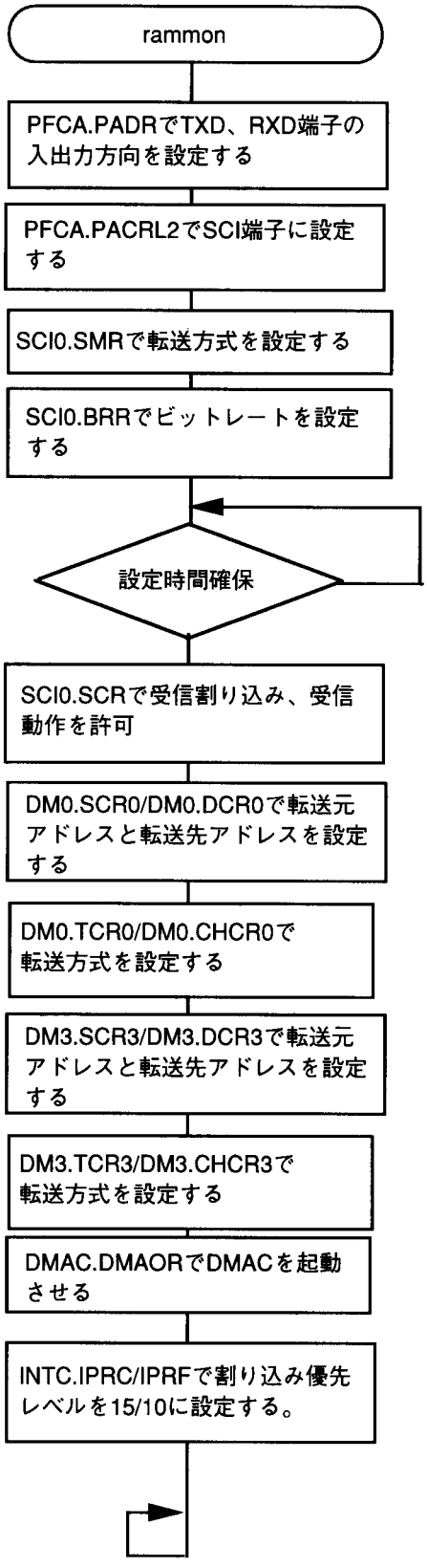
レジスタ名	機能割り付け	使用モジュール名
DM0.SAR0	RDRのアドレスの設定	メインルーチン
DM0.DAR0	転送先ＲＡＭ先頭アドレスの設定	メインルーチン 受信データ転送
DM0.TCR0	H'4（4回転送）を設定	
DM0.CHCR0	DMACの動作モード、転送方法等の設定	メインルーチン
DM3.SAR3	転送元RAM先頭アドレスの設定	
DM3.DAR3	TDRのアドレスの設定	メインルーチン 送信データ転送
DM3.TCR3	H'4（4回転送）を設定	
DM3.CHCR3	DMACの動作モード、転送方法等の設定	メインルーチン
DMAC.DMAOR	DMACの実行するチャネルの優先順位の設定	
PFCA.PADRL	SCIの入出力を設定	
PFCA.PACRL2	端子ﾏﾙﾁﾌﾟﾚｸｽをSCI0の使用に設定	
IINTC.IPRC	DMAC0,3の割り込み優先ﾚﾍﾞﾙを14,15に設定	
SCI0.SMR0	SCIを調歩同期式ﾓｰﾄﾞに設定	
SCI0.SCR0	送信、受信割り込み、送信、受信動作を許可	
SCI0.BRR0	転送ﾚｰﾄを設定	

（4）使用ＲＡＭ

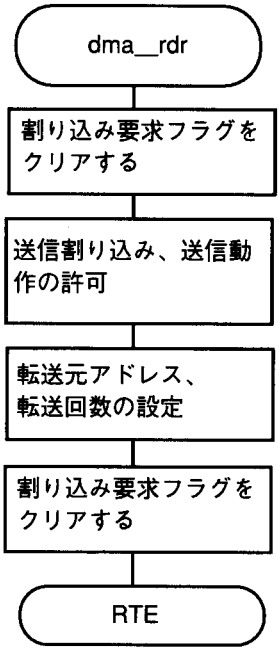
ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
lk_addr	ＲＡＭの参照アドレスを格納する。	unsigned long	ﾒｲﾝﾙｰﾁﾝ

注）レジスタのラベル名はＳＨ７０４０ヘッダファイルの名前を使用してます。

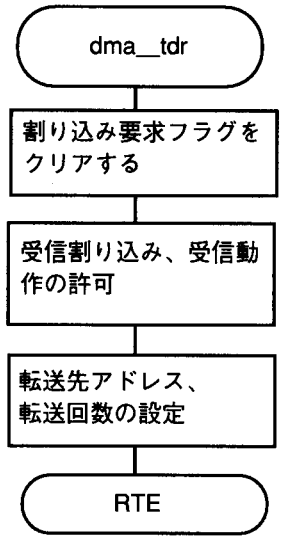
（1）メインルーチン


rammon
PFCA.PADRでTXD、RXD端子の入出力方向を設定する
PFCA.PACRL2でSCI端子に設定する
SCI0.SMRで転送方式を設定する
SCI0.BRRでビットレートを設定する
設定時間確保
SCI0.SCRで受信割り込み、受信動作を許可
DM0.SCR0/DM0.DCR0で転送元アドレスと転送先アドレスを設定する
DM0.TCR0/DM0.CHCR0で転送方式を設定する
DM3.SCR3/DM3.DCR3で転送元アドレスと転送先アドレスを設定する
DM3.TCR3/DM3.CHCR3で転送方式を設定する
DMAC.DMAORでDMACを起動させる
INTC.IPRC/IPRFで割り込み優先レベルを15/10に設定する。

（2）受信データ転送



（3）送信データ転送



DMACを使用したRAMモニタ	MCU	SH7040	使用機能	SCI、DMAC

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                              INCLUDE FILE                              */
/*------------------------------------------------------------------------*/
#include <machine.h>
#include "sh7040.h"
/*------------------------------------------------------------------------*/
/*                              PROTOTYPE                                 */
/*------------------------------------------------------------------------*/
void rammon(void);
#pragma interrupt( dma0 )
#pragma interrupt( dma_sci )
#pragma interrupt( sci_rxi )
/*------------------------------------------------------------------------*/
/*                            RAM ALLOCATION                              */
/*------------------------------------------------------------------------*/
#define data0 (*(volatile  unsigned char *)0xfffff000)
volatile  struct addr
{
    long   addr0;                       /*  transmit address  */
};
#define dat (*(struct addr *)0xfffff010)

/*------------------------------------------------------------------------*/
/*                            MAIN PROGRAM                                */
/*------------------------------------------------------------------------*/
rammon()
{
    signed int lp;

    dat.addr0 = (long)&data0;
    data0 = 'H';
    PFCA.PADRL = 0x0002;                /* out put TXD0,input RXD0 */
    PFCA.PACRL2 = 0x0005;               /* TXD0,RXD0 sellect  */
    SCI0.SCR0 = 0x00;                   /* stop transmit ・receive   */
    SCI0.SMR0 = 0x00;                   /* asynchronous mode、8bit data、not parity */
    SCI0.BRR0 = 0x40;                   /* bit rates 9600bps */
    SCI0.SCR0 = 0x00;                   /* clock sellect  */
    for( lp = 1; lp < 1; lp++ );        /* wait400ns */
    SCI0.SCR0 = 0x50;                   /* interrupt receive sci、receiving enabled */
    DM0.SAR0 = (long)(&SCI0.RDR);       /* source address：RAM */
    DM0.DAR0 = (long)(&dat.addr0);      /* distination address：SCI transmit register */
    DM0.TCR0 = 0x04;                    /* transmit count：4  */
    DM0.CHCR0 = 0x00004c05;             /* indirect,source increment,byte-size transfer */
    DM3.SAR3 = (long)(&dat.addr0);      /* source address：RAM  */
    DM3.DAR3 = (long)(&SCI0.TDR);       /* distination address：SCI transmit register  */
    DM3.CHCR3 = 0x00101b04;             /* indirect,source increment,byte-size transfer */
    DMAC.DMAOR = 0x0001;                /* data transfer enabled */
    INTC.IPRC = 0xe00f;                 /* set interrupt level=15 */
    INTC.IPRF = 0x00a0;                 /* set interrupt level=10 */
    set_imask(0x0);                     /* interrupt requested enabled */
    while(1);                           /* loop */
}
```

プログラムリスト

```
void dma_rdr( void )
{
    DM0.CHCR0 &= 0xfffffffd;                /* clear flag */
    SCI0.SCR0 = 0xa0;                       /* interrupt transmit sci、transmit enabled */
    DM3.SAR3 = (long)(&dat.addr0);          /* source address：RAM */
    DM3.TCR3 = 0x01;                        /* transmit count：4 */
    DM3.CHCR3 |= 0x00000001;                /* clear flag */
}


void dma_tdr( void )
{
    DM3.CHCR3 &= 0xfffffffc;                /* clear flag */
    SCI0.SCR0 = 0x70;                       /* interrupt receive sci、receiving enabled */
    DM0.DAR0 = (long)(&dat.addr0);          /* distination address：SCI transmit register */
    DM0.TCR0 = 0x04;                        /* transmit count：4 */
}
```

3. 付録

目　次

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* Internal I/O register address define                                   */
/*                                                                        */
/*------------------------------------------------------------------------*/

/*------------------------------------------------------------------------*/
/*INTC                                                                    */
/*------------------------------------------------------------------------*/
volatile struct intc
{
        short  IPRA;                    /*interrupt priority register A*/
        short  IPRB;                    /*interrupt priority register B*/
        short  IPRC;                    /*interrupt priority register C*/
        short  IPRD;                    /*interrupt priority register D*/
        short  IPRE;                    /*interrupt priority register E*/
        short  IPRF;                    /*interrupt priority register F*/
        short  IPRG;                    /*interrupt priority register G*/
        short  IPRH;                    /*interrupt priority register H*/
        short  ICR;                     /*interrupt controll register*/
        short  ISR;                     /*IRQ status register*/
};

#define INTC  (*(volatile struct intc  *)0xffff8348)

/*------------------------------------------------------------------------*/
/*UBC                                                                     */
/*------------------------------------------------------------------------*/
volatile struct ubc
{
        short  UBARH;                   /*user break address register H*/
        short  UBARL;                   /*user break address register L*/
        short  UBAMRH;                  /*user break mask register H*/
        short  UBAMRL;                  /*user break mask register L*/
        short  UBBR;                    /*user break bus cycle register */
};

#define UBC  (*(volatile struct  ubc  *)0xffff8600)

/*------------------------------------------------------------------------*/
/*DTC                                                                     */
/*------------------------------------------------------------------------*/
volatile struct dtcsh
{
        char   DTEA;                    /*enable register A*/
        char   DTEB;                    /*enable register B*/
        char   DTEC;                    /*enable register C*/
        char   DTED;                    /*enable register D*/
        char   DTEE;                    /*enable register E*/
        char   res1;
        short  DTCSR;                   /*control/status register*/
        short  DTBR;                    /*information base register*/
};

#define DTC  (*(volatile struct dtcsh  *)0xffff8700)

/*------------------------------------------------------------------------*/
/*CAC                                                                     */
/*------------------------------------------------------------------------*/
volatile struct cac
{
        short   CCR;                    /*cash control register*/
};

#define CAC  (*(volatile struct cac  *)0xffff8740)
```

```
/*------------------------------------------------------------------------*/
/*BSC                                                                     */
/*------------------------------------------------------------------------*/
volatile struct bsc
{
        short  BCR1;                    /*bus control register 1*/
        short  BCR2;                    /*bus control register 2*/
        short  WCR1;                    /*wait control register 1*/
        short  WCR2;                    /*wait control register 2*/
        short  res2;
        short  DCR;                     /*DRAM area control register*/
        short  RTCSR;                   /*refresh timer control register*/
        short  RTCNT;                   /*refresh timer counter*/
        short  RTCOR;                   /*refresh timer constant register*/
};

#define BSC  (*(volatile struct bsc  *)0xffff8620)

/*------------------------------------------------------------------------*/
/*DMAC                                                                    */
/*------------------------------------------------------------------------*/
volatile struct dmac
{
        short   DMAOR;                  /*operation register*/
};

volatile struct dmac0
{
        long    SAR0;                   /*source address register 0*/
        long    DAR0;                   /*destination address register 0*/
        long    TCR0;                   /*transfer count register 0*/
        long    CHCR0;                  /*channel control register 0*/
};

volatile struct dmac1
{
        long    SAR1;                   /*source address register 1*/
        long    DAR1;                   /*destination address register 1*/
        long    TCR1;                   /*transfer count register 1*/
        long    CHCR1;                  /*channel control register 1*/
};

volatile struct dmac2
{
        long    SAR2;                   /*source address register 2*/
        long    DAR2;                   /*destination address register 2*/
        long    TCR2;                   /*transfer count register 2*/
        long    CHCR2;                  /*channel control register 2*/
};

volatile struct dmac3
{
        long    SAR3;                   /*source address register 3*/
        long    DAR3;                   /*destination address register 3*/
        long    TCR3;                   /*transfer count register 3*/
        long    CHCR3;                  /*channel control register 3*/
};

#define DMAC  (*(volatile struct dmacsh  *)0xffff86b0)
#define DM0 (*(volatile struct dmac0 *)0xffff86c0)
#define DM1 (*(volatile struct dmac1 *)0xffff86d0)
#define DM2 (*(volatile struct dmac2 *)0xffff86e0)
#define DM3 (*(volatile struct dmac3 *)0xffff86f0)
```

```
/*-------------------------------------------------------------------*/
/*MTU                                                                */
/*-------------------------------------------------------------------*/
volatile struct tmrsh1
{
        char  TSTR;                     /*timer start register*/
        char  TSYR;                     /*timer synchronous register*/
};

#define TMRSH (*(volatile struct tmrsh1 *)0xffff8240)

/*-------------------------------------------------------------------*/
/*MTU CH0                                                            */
/*-------------------------------------------------------------------*/
volatile struct timer0
{
        char  TCR0;                     /*timer contoll register 0*/
        char  TMDR0;                    /*timer mode register 0*/
        char  TIOR0H;                   /*timer i/o controll register 0H*/
        char  TIOR0L;                   /*timer i/o controll register 0L*/
        char  TIER0;                    /*timer interrupt enable register 0*/
        char  TSR0;                     /*timer stasus register 0*/
        short TCNT0;                    /*timer counter 0*/
        short TGR0A;                    /*timer general register 0A*/
        short TGR0B;                    /*timer general register 0B*/
        short TGR0C;                    /*timer general register 0C*/
        short TGR0D;                    /*timer general register 0D*/
};

#define T0 (*(volatile struct timer0 *)0xffff8260)

/*-------------------------------------------------------------------*/
/*MTU CH1                                                            */
/*-------------------------------------------------------------------*/
volatile struct timer1
{
        char  TCR1;                     /*timer controll register 1*/
        char  TMDR1;                    /*timer mode register 1*/
        char  TIOR1;                    /*timer i/o controll register 1H*/
        char  res3;
        char  TIER1;                    /*timer interrupt enable register 1*/
        char  TSR1;                     /*timer status register 1*/
        short TCNT1;                    /*timer counter 1*/
        short TGR1A;                    /*timer general register 1A*/
        short TGR1B;                    /*timer general register 1B*/
};

#define T1 (*(volatile struct timer1 *)0xffff8280)

/*-------------------------------------------------------------------*/
/*MTU CH2                                                            */
/*-------------------------------------------------------------------*/
volatile struct timer2
{
        char  TCR2;                     /*timer controll register 2*/
        char  TMDR2;                    /*timer mode register 2*/
        char  TIOR2;                    /*timer i/o controll regoster 2H*/
        char  res4;
        char  TIER2;                    /*timer interrupt enable register 2*/
        char  TSR2;                     /*timer status register 2*/
        short TCNT2;                    /*timer counter 2*/
        short TGR2A;                    /*timer general register 2A*/
        short TGR2B;                    /*timer general register 2B*/
};

#define T2 (*(volatile struct timer2 *)0xffff82a0)
```

プログラムリスト

```
/*-------------------------------------------------------------------*/
/*MTU CH3,4                                                          */
/*-------------------------------------------------------------------*/
volatile struct timer3
{
        char  TCR3;                     /* timer contorol register 3 */
        char  TCR4;                     /* timer contorol register 4 */
        char  TMDR3;                    /* timer mode register 3 */
        char  TMDR4;                    /* timer mode register 4 */
        char  TIOR3H;                   /* timer I/O contorol register 3H */
        char  TIOR3L;                   /* timer I/O contorol register 3L */
        char  TIOR4H;                   /* timer I/O contorol register 4H */
        char  TIOR4L;                   /* timer I/O contorol register 4L */
        char  TIER3;                    /* timer interrupt enable register 3 */
        char  TIER4;                    /* timer interrupt enable register 4 */
        char  TOER;                     /* timer output master enable register */
        char  TOCR;                     /* timer output contorol register */
        char  res5;
        char  TGCR;                     /* timer gate contorol register */
};
volatile struct timer31
{
        short TCNT3;                    /* timer counter 3 */
        short TCNT4;                    /* timer counter 4 */
        short TCDR;                     /* timer cycle-data register */
        short TDDR;                     /* timer deadtime-data register */
        short TGR3A;                    /* timer general register 3A */
        short TGR3B;                    /* timer general register 3B */
        short TGR4A;                    /* timer general register 4A */
        short TGR4B;                    /* timer general register 4B */
        short TCNTS;                    /* timer sub-counter */
        short TCBR;                     /* timer cycle buffer register */
        short TGR3C;                    /* timer general register 3C */
        short TGR3D;                    /* timer general register 3D */
        short TGR4C;                    /* timer general register 4C */
        short TGR4D;                    /* timer general register 4D */
        char  TSR3;                     /* timer status register 3 */
        char  TSR4;                     /* timer status register 4 */
};

#define T3 (*(volatile struct timer3 *)0xffff8200)
#define T31 (*(volatile struct timer31 *)0xffff8210)

/*-------------------------------------------------------------------*/
/*WDT                                                                */
/*-------------------------------------------------------------------*/
volatile struct wdt
{
        char   TCSR;                    /* timer contorol status register */
        char   TCNT;                    /* timer counter */
        char   res8;
        char   RSTCSR;                  /* reset contorol status register */
};


#define WDT  (*(volatile struct wdt  *)0xffff8610)

volatile struct wdtw
{
        short  TCSR;                    /* timer control status register */
        short  RSTCSR;                  /* reset control status register */
};

#define WDTW  (*(volatile struct wdtw  *)0xffff8610)
```

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*SCI                                                                     */
/*------------------------------------------------------------------------*/
volatile struct sci0
{
        char    SMR0;                   /* serial mode register 0 */
        char    BRR0;                   /* bit-rate register 0 */
        char    SCR0;                   /* serial contorol register 0 */
        char    TDR0;                   /* transmit data register 0 */
        char    SSR0;                   /* serial status register 0 */
        char    RDR0;                   /* receive data register 0 */
};

volatile struct sci1
{
        char    SMR1;                   /* serial mode register 1 */
        char    BRR1;                   /* bit-rate register 1 */
        char    SCR1;                   /* serial contorol register 1 */
        char    TDR1;                   /* transmit data register 1 */
        char    SSR1;                   /* serial status register 1 */
        char    RDR1;                   /* receive data register 1 */
};

#define SCI0  (*(volatile struct sci0  *)0xffff81a0)
#define SCI1  (*(volatile struct sci1  *)0xffff81b0)

/*------------------------------------------------------------------------*/
/*A/D                                                                     */
/*------------------------------------------------------------------------*/
volatile struct a_d
{
        char    ADCSR;                  /* A/D contorol status register */
        char    ADCR;                   /* A/D contorol register */
};

volatile struct a_d0
{
        short   ADDRA;                  /* A/D data register A */
        short   ADDRB;                  /* A/D data register B */
        short   ADDRC;                  /* A/D data register C */
        short   ADDRD;                  /* A/D data register D */
        short   ADDRE;                  /* A/D data register E */
        short   ADDRF;                  /* A/D data register F */
        short   ADDRG;                  /* A/D data register G */
        short   ADDRH;                  /* A/D data register H */
};

#define A_D   (*(volatile struct a_d   *)0xffff83e0)
#define A_D0  (*(volatile struct a_d0  *)0xffff83f0)

/*------------------------------------------------------------------------*/
/*CMT                                                                     */
/*------------------------------------------------------------------------*/
volatile struct cmt
{
        short   CMSTR;                  /* compare/match timer start register */
        short   CMCSR0;                 /* compare/match timer contorol register 0 */
        short   CMCNT0;                 /* compare/match counter 0 */
        short   CMCOR0;                 /* compare/match constant register 0*/
        short   CMCSR1;                 /* compare/match timer contorol register 1 */
        short   CMCNT1;                 /* compare/match counter 1 */
        short   CMCOR1;                 /* compare/match timer contorol register 1 */
};

#define CMT   (*(volatile struct cmt   *)0xffff83d0)
```

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*PFC                                                                     */
/*------------------------------------------------------------------------*/
volatile struct pfca
{
        short   PADRH;                  /* portA data register H */
        short   PADRL;                  /* portA data register L */
        short   PAIORH;                 /* portA I/O register H */
        short   PAIORL;                 /* portA I/O register L */
        short   PACRH;                  /* portA contorol register H */
        short   res6;
        short   PACRL1;                 /* portA contorol register L1 */
        short   PACRL2;                 /* portA contorol register L2 */
};
volatile struct pfcb
{
        short   PBDR;                   /* portB data register */
        short   PCDR;                   /* portC data register */
        short   PBIOR;                  /* portB I/O register */
        short   PCIOR;                  /* portC I/O register */
        short   PBCR1;                  /* portB contorol register 1 */
        short   PBCR2;                  /* portB contorol register 2 */
        short   PCCR;                   /* portC contorol register */
};
volatile struct pfcd
{
        short   PDDRH;                  /* portD data register H */
        short   PDDRL;                  /* portD data register L */
        short   PDIORH;                 /* portD I/O register H */
        short   PDIORL;                 /* portD I/O register L */
        short   PDCRH1;                 /* portD contorol register H1 */
        short   PDCRH2;                 /* portD contorol register H2 */
        short   PDCRL;                  /* portD contorol register L */
};
volatile struct pfce
{
        short   PEDR;                   /* portE data register H */
        char    res7;
        char    PFDR;                   /* portF data register */
        char    res7;
        short   PEIOR;                  /* portE I/O register */
        short   res8;
        short   PECR1;                  /* portE contorol register 1 */
        short   PECR2;                  /* portE contorol register 2 */
};

volatile struct pfc0
{
        short   IFCR;                   /* IRQOUT contorol register */
};

#define PFCA    (*(volatile struct pfca   *)0xffff8380)
#define PFCB    (*(volatile struct pfcb   *)0xffff8390)
#define PFCD    (*(volatile struct pfcd   *)0xffff83A0)
#define PFCE    (*(volatile struct pfce   *)0xffff83B0)
#define PFC0    (*(volatile struct pfc0  *)0xffff83C8)

/*------------------------------------------------------------------------*/
/*POE                                                                     */
/*------------------------------------------------------------------------*/
volatile struct poe
{
        short   ICSR;                   /* input-level contorol status register */
        short   OCSR;                   /* output-level contorol status register */
};

#define POE     (*(volatile struct poe *)0xffff83c0)
```

ヘッダファイル	MCU	ＳＨ７０４０	使用機能	――――――

プログラムリスト

```
volatile struct sby
{
        short   SBYCR;                  /* standby control register */
};

#define SBY   (*(volatile struct sby *)0xffff8614)
```

SH7040 シリーズ　内蔵 I/O 編
アプリケーションマニュアル

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-502-052A