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

SH7050 シリーズ　内蔵I/O 編

アプリケーションノート

ルネサスSuperH RISC engine

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　1996.03

はじめに

ＳＨ７０５０シリーズはＲＩＳＣ方式のＣＰＵをコアにして、システム構成に必要な周辺機能を集積したシングルチップＲＩＳＣマイクロコンピュータです。

１チップ上にＣＰＵ、ＲＯＭ、ＲＡＭ、ＤＭＡＣ、タイマ、ＳＣＩ、Ａ／Ｄ変換器、割り込みコントローラ、Ｉ／Ｏポート等を内蔵しており、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。

ＳＨ７０５０アプリケーションノート（内蔵Ｉ／Ｏ編）は、ＳＨ７０５０の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及びハードウェア設計の際、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タスク例は動作確認しておりますが、実際にご使用になる場合には、必ず動作確認の上ご使用くださいますようにお願いします。

目次

1. SH7050シリーズ アプリケーションノート使用手引

目　次

１．１　内蔵Ｉ／Ｏ編構成

内蔵Ｉ／Ｏ編は図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。レジスタのラベル名は各タスク共通のヘッダファイルの名前を使用しています。

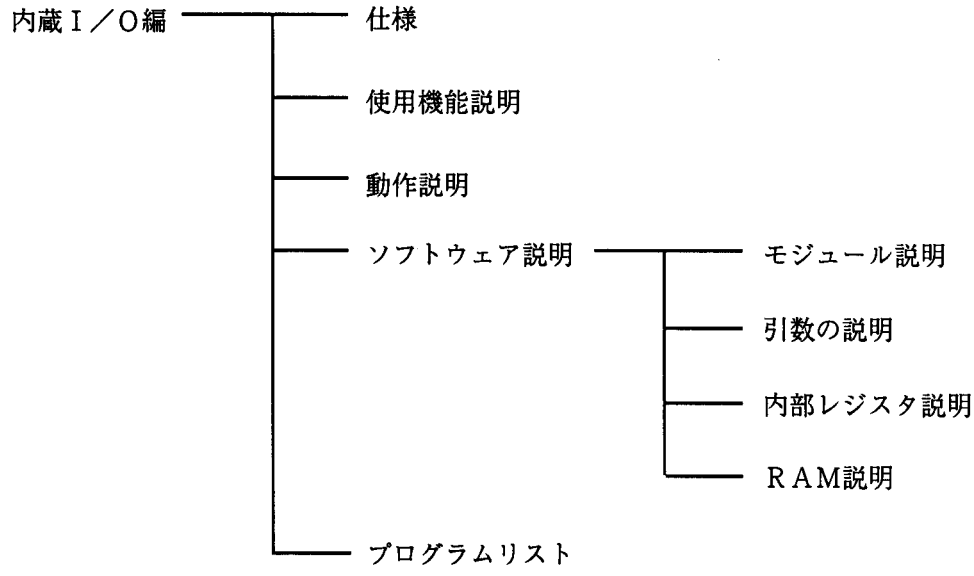


図１　内蔵Ｉ／Ｏ編構成

（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）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。尚、各プログラムで使用するヘッダファイルにはライブラリ関数用ヘッダファイル、組み込み関数用ヘッダファイル、ＳＨ７０５０内蔵Ｉ／Ｏレジスタ用ヘッダファイルがあります。ライブラリ関数用ヘッダファイルと組み込み関数用ヘッダファイルの仕様はＳＨシリーズＣコンパイラを参照してください。ＳＨ７０５０内蔵Ｉ／Ｏレジスタ用ヘッダファイルの内容は２．２４に掲載しています。

１．２　付録

付録には周辺ＬＳＩ（ＲＯＭ、ＲＡＭ等）とのインターフェース例を添付しています。図２に示す構成でインターフェース例について説明しています。

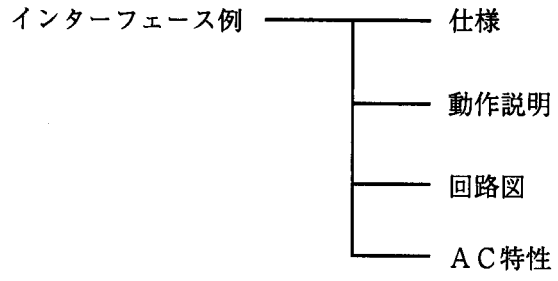


図２　インターフェース例構成

（１）仕様
　接続する周辺ＬＳＩ名及びメモリマップ等の回路仕様について説明しています。

（２）動作説明
　回路の動作をタイミングチャートを使用し説明しています。

（３）回路図
　周辺ＬＳＩとインターフェースする回路図を示します。

（４）ＡＣ特性
　ＳＨ７０５０シリーズマイクロコンピュータ及び周辺ＬＳＩのＡＣ特性を示します。

2. SH7050内蔵I／O編

目　次

仕様

（1）図1に示すようにユーザブレークレジスタに設定したブレークポイント（H' 00001012番地）にある命令の手前でユーザブレーク割り込みを発生します。

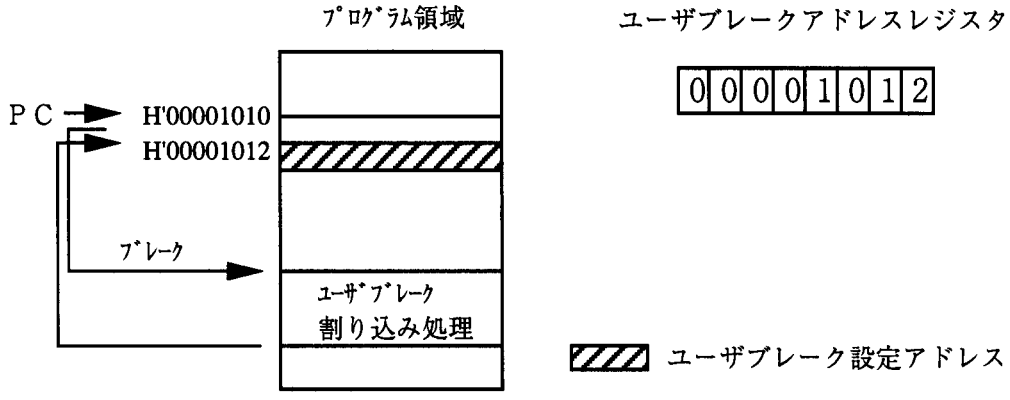


図1 ユーザーブレークコントローラの動作ブロック図

使用機能説明

表1に本タスク例の機能割付を示します。表1に示すようにSH7050に内蔵しているUBCの機能を割付、ユーザブレークを行ないます。

表1 UBC機能割付

UBCレジスタ	機能
UBAR	ユーザブレークアドレスを設定する。
UBAMR	ユーザブレークマスクアドレスを設定する。
UBBR	ユーザブレーク条件を設定する。

動作説明

図2にﾕｰｻﾞﾌﾞﾚｰｸｺﾝﾄﾛｰﾗを使用したソフトウェア例を示します。あらかじめ初期設定でプログラムブレークを設定し、ブレークポイントの手前の命令でブレーク割り込みを発生します。ブレーク処理ではブレークの有無を示すフラグをセットします。

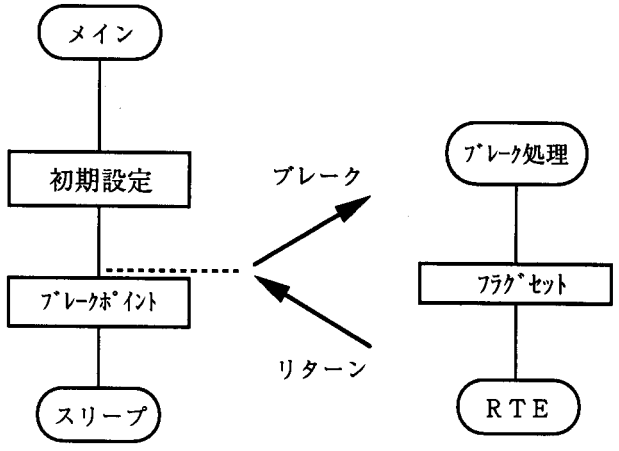


図2　ユーザブレークコントローラ機能を使用したソフトウェア例

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
ﾒｲﾝﾙｰﾁﾝ	ubcmn	ＵＢＣの初期設定を行なう。
ﾌﾞﾚｰｸ処理	ubcbk	ﾕｰｻﾞﾌﾞﾚｰｸで起動し、ブレークの有無を示すフラグをセットする。

（2）引数の説明

本タスクでは引き数は使用していません。

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　　能	設定値	使用ﾓｼﾞｭｰﾙ名
UBC.UBAR	命令フェッチアドレスを設定する。	0x00001012 *	ﾒｲﾝﾙｰﾁﾝ
UBC.UBBR	ﾌﾞﾚｰｸ条件をＣＰＵｻｲｸﾙ、命令ﾌｪｯﾁｻｲｸﾙに設定する。	0x0054	ﾒｲﾝﾙｰﾁﾝ

＊ｺﾝﾊﾟｲﾙ後ﾘｽﾄﾌｧｲﾙより参照

（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
ｐｃｆ	命令フェッチに使用する変数。	unsined char	ﾒｲﾝﾙｰﾁﾝ
ｂｒｋ	ブレークの有無を示すフラグ		

```
#include <machine.h>                    /* ライブラリ関数用ヘッダファイル                */
#include "sh7050.h"                     /* 周辺レジスタ定義ヘッダファイル                */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void ubcmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define pcf       (*(unsigned char  *)0xFFFFE800) /* 命令フェッチに使用する変数  */
#define brk       (*(unsigned char  *)0xFFFFE801) /* ブレーク判定フラグ          */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  メインルーチン                                                         */
/*                                                                         */
/*-------------------------------------------------------------------------*/
void ubcmn( void )
{
    *(long *)(&UBC.UBARH) = 0x00001012;/* ブレークアドレスの設定                   */
    UBC.UBBR = 0x0054;                 /* バスサイクル:CPU、命令フェッチ、リード   */
    set_imask(0x0);                    /* 割り込み許可                             */
    pcf = 1;                           /* 命令フェッチブレーク設定行               */
    sleep();
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  ブレーク処理                                                           */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( ubcbk )
void ubcbk( void )
{
    brk = 1;                           /* ブレーク判定フラグセット                 */
}
```

仕様

（1）図１に示すようにユーザブレーク アドレスレジスタに設定したブレークポイント（ＦＦＦＦＥ８００番地）にあるデータをアクセスし、ユーザブレーク割り込みを発生します。
（2）ブレーク条件はＣＰＵによる１バイトデータのライトです。

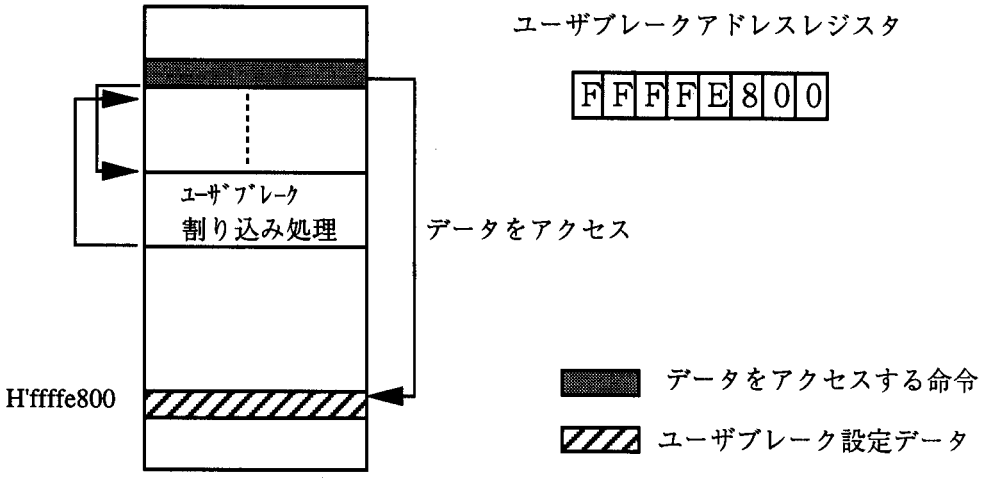


図１　ﾕｰｻﾞｰﾌﾞﾚｰｸｺﾝﾄﾛｰﾗの動作ブロック図

使用機能説明

表１に本タスク例の機能割付を示します。表１に示すようにＵＢＣの機能を割付、ユーザブレークを行ないます。

表１　ＵＢＣ機能割付

ＵＢＣレジスタ	機　　能
UBAR	ユーザブレークアドレスを設定する。
UBAMR	ユーザブレークマスクアドレスを設定する。
UBBR	ユーザブレーク条件を設定する。

動作説明

図２にﾕｰｻﾞﾌﾞﾚｰｸｺﾝﾄﾛｰﾗを使用したソフトウェア例を示します。あらかじめ初期設定でデータアクセスブレークを変数に設定し、ブレーク設定変数アクセス時にブレーク割り込みを発生します。ブレーク処理ではブレークの有無を示すフラグをセットします。

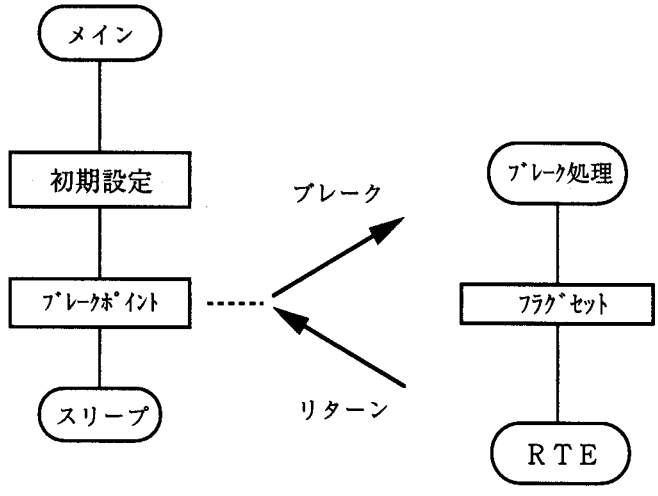


図２　ユーザブレークコントローラ機能を使用したソフトウェア例

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
ﾒｲﾝﾙｰﾁﾝ	ubcmn	ＵＢＣの初期設定を行なう。
ﾌﾞﾚｰｸ処理	ubcbk	ﾕｰｻﾞﾌﾞﾚｰｸで起動し、ブレークの有無を示すフラグをセットする。

（2）引数の説明

本タスクでは引数は使用してません。

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
UBC.UBAR	データアクセスアドレスを設定する。	&dat_ac	ﾒｲﾝﾙｰﾁﾝ
UBC.UBBR	ユーザブレーク条件を設定する。	0x006a	

（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
ｄａｔ＿ａｃ	アクセスブレークを設定する変数。	unsined char	ﾒｲﾝﾙｰﾁﾝ
ｂｒｋ	ブレークの有無を示すフラグ		

プログラムリスト

```
#include <machine.h>                    /* ライブラリ関数用ヘッダファイル          */
#include "sh7050.h"                     /* 周辺レジスタ定義ヘッダファイル          */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void ubcmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define dat_ac    (*(unsigned char  *)0xFFFFE800) /* ブレーク設定変数       */
#define brk       (*(unsigned char  *)0xFFFFE801) /* ブレーク判定フラグ     */
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  メインルーチン                                                         */
/*                                                                         */
/*-------------------------------------------------------------------------*/
void ubcmn( void )
{
    UBCL_UBAR = (long)&dat_ac;          /* ブレークアドレスの設定          */
    UBC.UBBR = 0x006d;                  /* バスサイクル:CPU、命令フェッチ、リード */
    set_imask(0x0);                     /* 割り込み許可                    */
    dat_ac = 1;                         /* ブレーク実行フラグセット        */
    sleep();
}

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  ブレーク処理                                                           */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( ubcbk )
void ubcbk( void )
{
    brk = 1;                            /* ブレーク判定フラグのセット      */
}
```

2.3　DMACを使用したRAMモニタ	MCU	SH7050	使用機能	SCI、DMAC

仕様

（1）図1に示すように、SH7050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にバイト単位で転送します。
（5）DMACの転送条件は表1、表2に示す通りです。

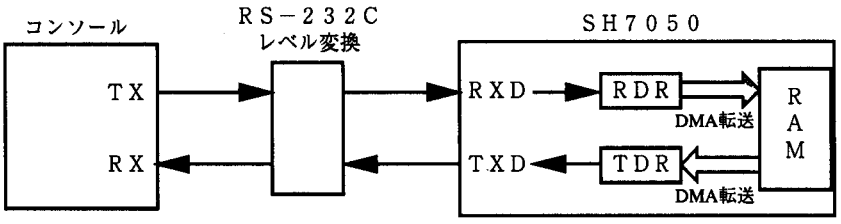


図1　SH7050によるRAM上データのSCI転送ブロック図

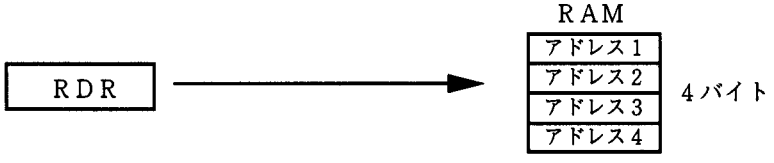


図2　DMACを用いたデータ転送（転送元直接アドレス）

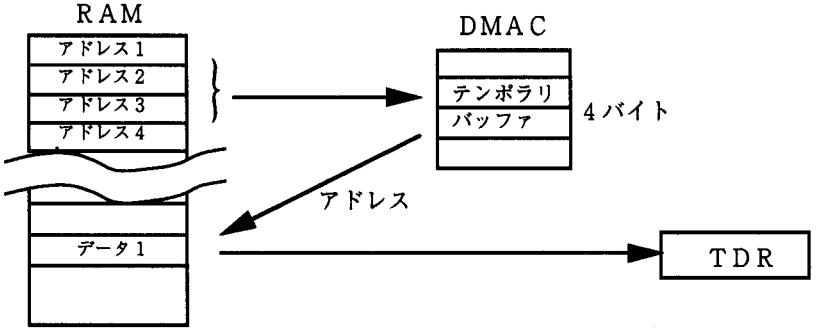


図3　DMACを用いたデータ転送（転送元間接アドレス）

表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	内蔵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　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	1回
転送元アドレス	増加
転送先アドレス	固定
転送要求元	内蔵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

表3、表4、表5に本タスク例の機能割付を示します。表3、表4、表5に示すようにSH7050に内蔵しているDMAC、SCI及びPFC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3　DMAC機能割付

DMACレジスタ	機　能
SAR0	転送元アドレスを設定する。
SAR3	
DAR0	転送先アドレスを設定する。
DAR3	
TCR0	転送回数を設定する。
TCR3	
CHCR0	DMACの動作モード、転送方法等を設定する。
CHCR3	
DMAOR	DMACの実行するチャネルの優先順位を設定する。

表4　SCI機能割付

SCI機能		機　能
端子	RXD	コンソールからﾃﾞｰﾀを受信する。
	TXD	コンソールへﾃﾞｰﾀを送信する。
レジスタ	SMR	SCIの送信フォーマットを設定する。
	SCR	SCIの割り込みの許可／禁止を設定する。
	SSR	割り込みステータスを設定する。
	RDR	コンソールから受信したﾃﾞｰﾀを設定する。
	TDR	コンソールへ送信するﾃﾞｰﾀを設定する。
	BRR	転送ﾚｰﾄを設定する。

表5　PFC機能割付

PFCレジスタ	機　能
PGIOR	端子の入出力方向を設定する。
PGCR2	端子の機能を選択する。

動作説明

図4に動作原理を示します。図4に示すように、SH7050のハードウェア処理及びソフトウェア処理によりシリアルによるデータの送受信を行います。

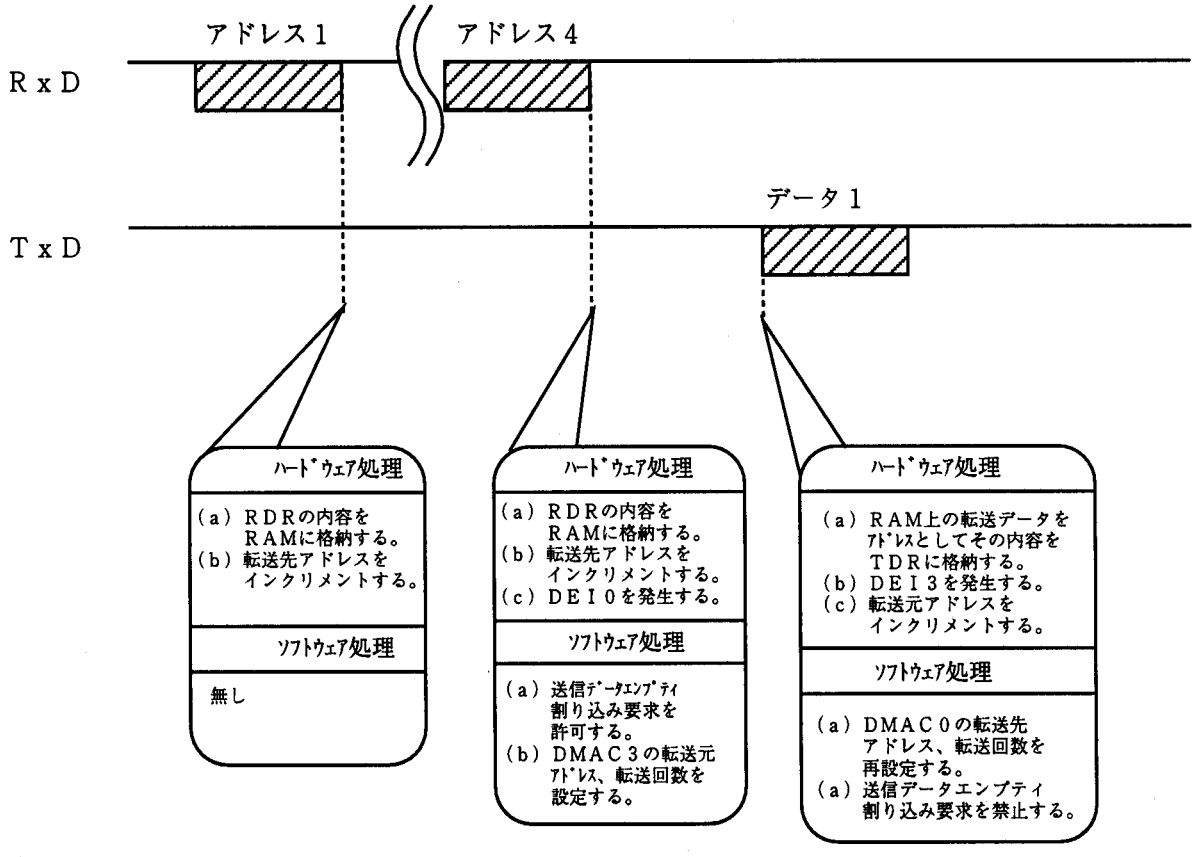


図4　SCIによるデータ転送動作原理

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
ﾒｲﾝﾙｰﾁﾝ	dma_scimn	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

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
dat.addr0	ＲＡＭの参照アドレスを格納する。	unsigned long	ﾒｲﾝﾙｰﾁﾝ
data0	参照データを格納する。	unsigned char	ﾒｲﾝﾙｰﾁﾝ

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
DMA.SAR0	ＲＤＲのアドレスを設定する。	&SCI0.RDR	ﾒｲﾝﾙｰﾁﾝ
DMA.DAR0	転送先ＲＡＭ先頭アドレスを設定する。	&dat.addr0	ﾒｲﾝﾙｰﾁﾝ 受信データ転送
DMA.TCR0	転送回数（４回）を設定する。	0x04	ﾒｲﾝﾙｰﾁﾝ 受信データ転送
DMA.CHCR0	ＤＭＡＣの動作モード、転送方法等を設定する。	0x00004905	ﾒｲﾝﾙｰﾁﾝ
DMA.SAR3	転送元ＲＡＭ先頭アドレスを設定する。	&dat.addr0	ﾒｲﾝﾙｰﾁﾝ
DMA.DAR3	ＴＤＲのアドレスを設定する。	&SCI0.TDR	ﾒｲﾝﾙｰﾁﾝ 送信データ転送
DMA.TCR3	転送回数（１回）を設定する。	0x01	ﾒｲﾝﾙｰﾁﾝ 送信データ転送
DMA.CHCR3	ＤＭＡＣの動作モード、転送方法等を設定する。	0x00101805	ﾒｲﾝﾙｰﾁﾝ
DMA.OR	ＤＭＡＣの実行するチャネルの優先順位を設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
PFC.PGIOR	ＳＣＩの入出力を設定する。	0x0004	ﾒｲﾝﾙｰﾁﾝ
PFC.PGCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをＳＣＩ０の使用にする。	0x0060	ﾒｲﾝﾙｰﾁﾝ
INT.IPRC	DMAC0,3の割り込み優先ﾚﾍﾞﾙを14,15に設定する。	0xef00	ﾒｲﾝﾙｰﾁﾝ
INT.IPRH	SCI0の割り込み優先ﾚﾍﾞﾙを10に設定する。	0xa000	ﾒｲﾝﾙｰﾁﾝ
SCI0.SMR	SCIを調歩同期式ﾓｰﾄﾞに設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
SCI0.SCR	送信、受信割り込み、送信、受信動作を許可する。	0xf0	ﾒｲﾝﾙｰﾁﾝ
SCI0.BRR	転送ﾚｰﾄを設定する。	0x40	ﾒｲﾝﾙｰﾁﾝ

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                    */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ                */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void dma_scimn(void);
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define data0 (*(volatile  unsigned char *)0xffffe800)
volatile  struct addr
{
    long   addr0;                       /* 転送アドレス０                  */
};
#define dat (*(struct addr *)0xFFFFE810)

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*                    メインルーチン                                       */
/*                                                                         */
/*-------------------------------------------------------------------------*/
void dma_scimn(void)
{
    signed int lp;
    dat.addr0 = (long)&data0;
    data0 = 'H';
    PFC.PGIOR = 0x0004;                 /* TXD0出力,RXD0入力               */
    PFC.PGCR2 = 0x0060;                 /* TXD0,RXD0使用                   */
    SCI0.SCR = 0x00;                    /* 送信・受信動作を禁止            */
    SCI0.SMR = 0x00;                    /* 調歩同期式､8ﾋﾞｯﾄﾃﾞｰﾀ、ﾊﾟﾘﾃｨなし  */
    SCI0.BRR = 0x40;                    /* ビットレート９６００ｂｐｓ      */
    SCI0.SCR = 0x00;                    /* 内部クロック                    */
    for( lp = 1; lp < 1; lp++ );        /* ４００ｎｓウエィト              */
    SCI0.SCR = 0x50;                    /* ＳＣＩ受信割り込み、受信動作を許可 */
    DMA.SAR0 = (long)(&SCI0.RDR);       /* 転送元アドレス：SCIﾚｼｰﾌﾞﾃﾞｰﾀﾚｼﾞｽﾀ */
    DMA.DAR0 = (long)(&dat.addr0);      /* 転送先アドレス：ＲＡＭ          */
    DMA.TCR0 = 0x04;                    /* 転送回数：４回                  */
    DMA.CHCR0 = 0x00004905;             /* ｿｰｽ固定、ﾃﾞｨｽﾃｨﾈｰｼｮﾝ増加,ﾊﾞｲﾄ転送 */
    DMA.SAR3 = (long)(&dat.addr0);      /* 転送元アドレス：ＲＡＭ          */
    DMA.DAR3 = (long)(&SCI0.TDR);       /* 転送先アドレス：SCIﾄﾗﾝｽﾐｯﾄﾚｼﾞｽﾀ */
    DMA.CHCR3 = 0x00181804;             /* ｲﾝﾀﾞｲﾚｸﾄ,ｿｰｽ増加,ﾊﾞｲﾄ転送     */
    DMAOR = 0x0001;                     /* ＤＭＡＣ起動許可                */
    INT.IPRC = 0xef00;                  /* DMA0,3割り込み優先ﾚﾍﾞﾙを14,15に設定*/
    INT.IPRH = 0xa000;                  /* SCI0割り込み優先ﾚﾍﾞﾙを１０に設定 */
    set_imask(0x0);                     /* 割り込み許可                    */
    while(1);                           /* 無限ループ（割り込み待ち）      */
}
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ＤＭＡ０転送終了割り込みルーチン                                       */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( dma0 )
void dma0( void )
{
    DMA.CHCR0 &= 0xfffffffd;            /* TEフラグクリア                 */
    SCI0.SCR = 0xa0;                    /* ＳＣＩ送信割り込み、送信動作を許可 */
    DMA.SAR3 = (long)(&dat.addr0);      /* 転送元アドレス：ＲＡＭ         */
    DMA.TCR3 = 0x01;                    /* 転送回数：１回                 */
    DMA.CHCR3 |= 0x00000001;            /* TEフラグクリア                 */
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ＤＭＡ３転送終了割り込みルーチン                                       */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( dma_sci )
void dma_sci( void )
{
    DMA.CHCR3 &= 0xfffffffc;            /* TEﾌﾗｸﾞｸﾘｱ, 転送終了割り込み禁止 */
    SCI0.SCR = 0x70;                    /* ＳＣＩ受信割り込み、受信動作を許可 */
    DMA.DAR0 = (long)(&dat.addr0);      /* 転送先アドレス：ＲＡＭ         */
    DMA.TCR0 = 0x04;                    /* 転送回数：４回                 */
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* データレシーブ割り込みルーチン                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( sci_rxi )
void sci_rxi( void )
{
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* データエンプティ割り込みルーチン                                       */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( sci_txi )
void sci_txi( void )
{
}
```

仕様

（1）図1に示すように、SH7050のSCIを調歩同期式モードで使用し、コンソールに32バイトのデータを送信します。
（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）RAMからTDRへのデータ転送は図2に示すようにDMACの間接アドレス転送モードを使用し、CPUによるデータのライト時にユーザブレーク割り込みでDMACを起動し、以下に示すようなデータ転送を行います。
（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。
（4）DMACの転送条件は表1に示す通りです。

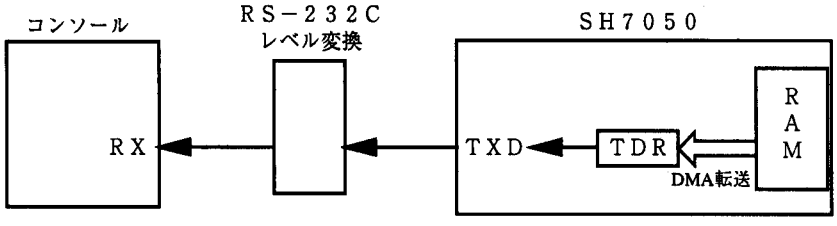


図1　SH7050によるRAM上データのSCI転送ブロック図

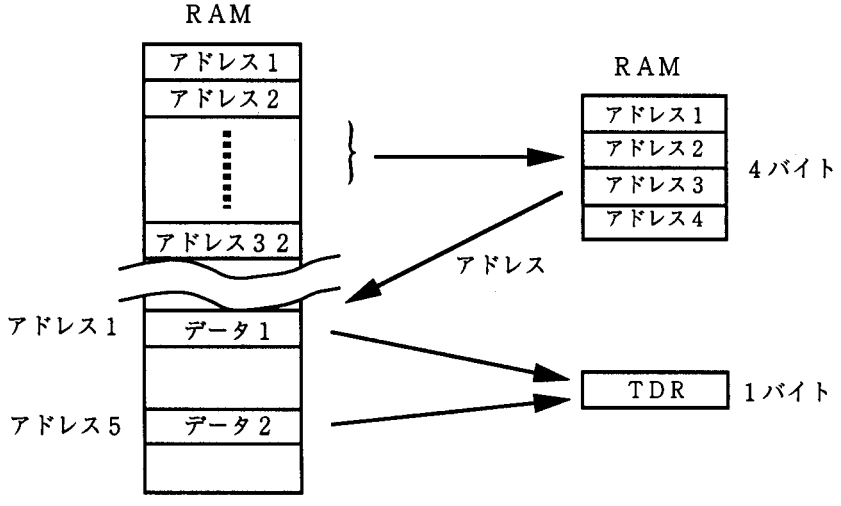


図2　DMACを用いたデータ転送（転送元間接アドレス）

表1　DMA転送条件

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	16回
転送元アドレス	増加
転送先アドレス	固定
転送要求元	内蔵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

表2、表3、表4、表5に本タスク例の機能割付を示します。表2、表3、表4、表5に示すようにSH7050に内蔵しているDMAC、SCI、UBC及びPFCの機能を割付、RAM上のデータをSCIによって転送します。

表2　DMAC機能割付

DMACレジスタ	機　能
SAR3	転送元アドレスを設定する。
DAR3	転送先アドレスを設定する。
TCR3	転送回数を設定する。
CHCR3	DMACの動作モード、転送方法等を設定する。
DMAOR	DMACの実行するチャネルの優先順位を設定する。

表3　SCI機能割付

SCI機能		機　能
端子	RXD	コンソールからﾃﾞｰﾀを受信する。
	TXD	コンソールへﾃﾞｰﾀを送信する。
レジスタ	SMR	SCIの送信フォーマットを設定する。
	SCR	SCIの割り込みの許可／禁止を設定する。
	SSR	割り込みステータスを設定する。
	RDR	コンソールから受信したﾃﾞｰﾀを設定する。
	TDR	コンソールへ送信するﾃﾞｰﾀを設定する。
	BRR	転送ﾚｰﾄを設定する。

表4　UBC機能割付

SCIレジスタ	機　能
UBAR	ユーザブレークアドレスを設定する。
UBBR	ユーザブレーク条件を設定する。

表5　PFC機能割付

PFCレジスタ	機　能
PGIOR	端子の入出力方向を設定する。
PGCR2	端子の機能を選択する。

動作説明

図３に動作原理を示します。図３に示すように、ＳＨ７０５０のハードウェア処理及びソフトウェア処理によりシリアルによるデータの転送を行います。

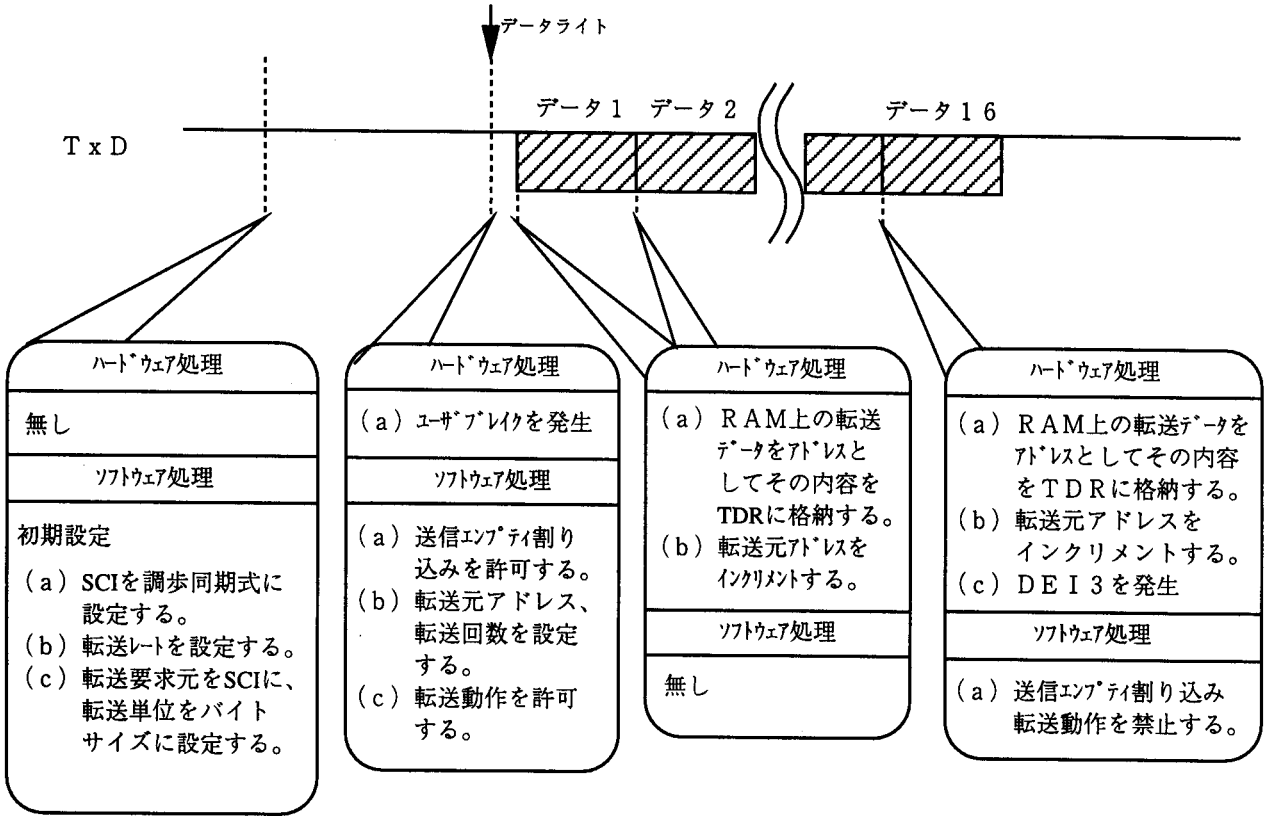


図３　ＳＣＩによるデータ転送動作原理

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
ﾒｲﾝﾙｰﾁﾝ	dma_scimn	SCI及びDMACの初期設定を行なう。
転送終了	dma3	DEI3で起動し、SCI送信割り込み、DMACの転送動作を禁止する。
ﾕｰｻﾞｰﾌﾞﾚｲｸ	ubcbk	SCI送信割り込み、DMACの転送動作を許可する。
SCI送信終了	sci_tr	DMACによって送信終了フラグをクリアする。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
dat.addr0 〜 dataddr15	参照アドレスを格納する。	unsigned long	ﾒｲﾝﾙｰﾁﾝ
data0 〜 data15	参照データ	unsigned char	ﾒｲﾝﾙｰﾁﾝ

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
DMA.SAR3	転送元RAM先頭アドレスを設定する。	&dat.addr0	ﾒｲﾝﾙｰﾁﾝ
DMA.DAR3	TDRのアドレスを設定する。	&SCI0.TDR	ﾒｲﾝﾙｰﾁﾝ 転送終了
DMA.TCR3	転送回数（16回）を設定する。	0x10	ﾒｲﾝﾙｰﾁﾝ 転送終了
DMA.CHCR3	DMACの動作ﾓｰﾄﾞ、転送方法割り込みの有無を設定する。	0x00101805	ﾒｲﾝﾙｰﾁﾝ
DMAOR	DMACの実行するチャネルの優先順位を設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
PFC.PGIOR	SCIの入出力を設定する。	0x0004	ﾒｲﾝﾙｰﾁﾝ
PFC.PGCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをSCI0の使用にする。	0x0060	ﾒｲﾝﾙｰﾁﾝ
INT.IPRC	DMAC3の割り込み優先ﾚﾍﾞﾙを15に設定する。	0x0f00	ﾒｲﾝﾙｰﾁﾝ
INT.IPRH	SCI0の割り込み優先ﾚﾍﾞﾙを14に設定する。	0xe000	ﾒｲﾝﾙｰﾁﾝ
SCI0.SMR	SCIを調歩同期式ﾓｰﾄﾞに設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
SCI0.SCR	送信割り込み、送信動作を許可する。	0xa0	ﾒｲﾝﾙｰﾁﾝ
SCI0.BRR	転送ﾚｰﾄを設定する。	0x40	ﾒｲﾝﾙｰﾁﾝ
UBC.UBAR	転送元RAM先頭アドレスを設定する。	&data0	ﾒｲﾝﾙｰﾁﾝ
UBC.UBBR	ブレーク条件をデータアクセス、ライト時、バイトサイズに設定する。	0x0069	ﾒｲﾝﾙｰﾁﾝ

（4）使用RAM

本タスクでは引数以外のRAMは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void dat_set( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define data0 (*(volatile  unsigned char *)0xffffe800)
#define data1 (*(volatile  unsigned char *)0xffffe801)
#define data2 (*(volatile  unsigned char *)0xffffe804)
#define data3 (*(volatile  unsigned char *)0xffffe805)
#define data4 (*(volatile  unsigned char *)0xffffe806)
#define data5 (*(volatile  unsigned char *)0xffffe807)
#define data6 (*(volatile  unsigned char *)0xffffe80a)
#define data7 (*(volatile  unsigned char *)0xffffe80f)
#define data8 (*(volatile  unsigned char *)0xffffe818)
#define data9 (*(volatile  unsigned char *)0xffffe819)
#define data10 (*(volatile  unsigned char *)0xffffe822)
#define data11 (*(volatile  unsigned char *)0xffffe823)
#define data12 (*(volatile  unsigned char *)0xffffe828)
#define data13 (*(volatile  unsigned char *)0xffffe82a)
#define data14 (*(volatile  unsigned char *)0xffffe830)
#define data15 (*(volatile  unsigned char *)0xffffe833)

volatile  struct addr
{
    long   addr0;                       /* 転送アドレス０                  */
    long   addr1;                       /* 転送アドレス１                  */
    long   addr2;                       /* 転送アドレス２                  */
    long   addr3;                       /* 転送アドレス３                  */
    long   addr4;                       /* 転送アドレス４                  */
    long   addr5;                       /* 転送アドレス５                  */
    long   addr6;                       /* 転送アドレス６                  */
    long   addr7;                       /* 転送アドレス７                  */
    long   addr8;                       /* 転送アドレス８                  */
    long   addr9;                       /* 転送アドレス９                  */
    long   addr10;                      /* 転送アドレス１０                */
    long   addr11;                      /* 転送アドレス１１                */
    long   addr12;                      /* 転送アドレス１２                */
    long   addr13;                      /* 転送アドレス１３                */
    long   addr14;                      /* 転送アドレス１４                */
    long   addr15;                      /* 転送アドレス１５                */
};
#define dat (*(struct addr *)0xFFFFE880)
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* メインルーチン                                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
void dat_set( void )
{
    signed int lp;
    dat_set();                          /* データ設定                     */
    PFC.PGIOR = 0x0104;                 /* TXD0出力,RXD0入力              */
    PFC.PGCR2 = 0x0060;                 /* TXD0,RXD0使用                  */
    SCI0.SCR = 0x00;                    /* 送信・受信動作を禁止           */
    SCI0.SMR = 0x00;                    /* 調歩同期式､8ﾋﾞｯﾄﾃﾞｰﾀ、ﾊﾟﾘﾃｨなし  */
    SCI0.BRR = 0x40;                    /* ビットレート９６００ｂｐｓ     */
    SCI0.SCR = 0x00;                    /* 内部クロック                   */
    for( lp = 1; lp < 1; lp++ );        /* ４００ｎｓウエィト             */
    SCI0.SCR = 0x20;                    /* 送信動作を許可する。           */
    UBCL_UBAR = (long)&data0;           /* ﾕｰｻﾞﾌﾞﾚｲｸｱﾄﾞﾚｽをＲＡＭに設定する */
    UBC.UBBR = 0x0069;                  /* データのライトサイクルでブレーク */
    DMA.SAR3 = (long)(&dat.addr0);      /* 転送元アドレス：ＲＡＭ         */
    DMA.DAR3 = (long)(&SCI0.TDR);       /* 転送先アドレス：SCIﾄﾗﾝｽﾐｯﾄﾚｼﾞｽﾀ */
    DMA.CHCR3 = 0x00181804;             /* ｲﾝﾀﾞｲﾚｸﾄ,ﾘﾛｰﾄﾞ,ｿｰｽ増加,ﾊﾞｲﾄ転送 */
    DMAOR = 0x0001;                     /* ＤＭＡＣ起動許可               */
    INT.IPRC = 0x0d00;                  /* DMA3割り込み優先ﾚﾍﾞﾙを１３に設定 */
    INT.IPRH = 0xc000;                  /* SCI0割り込み優先ﾚﾍﾞﾙを１２に設定 */
    set_imask(0x0);                     /* 割り込み許可                   */
    while(1);
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ユーザブレイク割り込みルーチン                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( ubcbk )
void ubcbk( void )
{
    SCI0.SCR |= 0x80;                   /* ＳＣＩ受信割り込みを許可する   */
    DMA.SAR3 = (long)(&dat.addr0);      /* 転送元アドレス：ＲＡＭ         */
    DMA.TCR3 = 0x10;                    /* 転送回数：３２回               */
    DMA.CHCR3 |= 0x00000001;            /* TEフラグクリア                 */
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ＤＭＡ転送終了割り込みルーチン                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( dma_sci )
void dma_sci( void )
{
    DMA.CHCR3 &= 0xfffffffc;            /* TEフラグクリア                 */
    SCI0.SCR &= 0x7f;                   /* ＳＣＩ受信割り込みを禁止する   */
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* データエンプティ割り込みルーチン                                       */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( sci_txi )
void sci_txi( void )
{
}
```

プログラムリスト

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* データ設定ルーチン                                                     */
/*                                                                        */
/*------------------------------------------------------------------------*/
void dat_set( void )
{
    dat.addr0 = (long)&data0;
    dat.addr1 = (long)&data1;
    dat.addr2 = (long)&data2;
    dat.addr3 = (long)&data3;
    dat.addr4 = (long)&data4;
    dat.addr5 = (long)&data5;
    dat.addr6 = (long)&data6;
    dat.addr7 = (long)&data7;
    dat.addr8 = (long)&data8;
    dat.addr9 = (long)&data9;
    dat.addr10 = (long)&data10;
    dat.addr11 = (long)&data11;
    dat.addr12 = (long)&data12;
    dat.addr13 = (long)&data13;
    dat.addr14 = (long)&data14;
    dat.addr15 = (long)&data15;

    data0 =   'S';
    data1 =   'U';
    data2 =   'P';
    data3 =   'E';
    data4 =   'R';
    data5 =   ' ';
    data6 =   'H';
    data7 =   ' ';
    data8 =   '7';
    data9 =   '0';
    data10 =  '5';
    data11 =  '0';
    data12 =  ' ';
    data13 =  ' ';
    data14 =  ' ';
    data15 =  ' ';
}
```

仕様

（1）図1に示すように、1グループ（AN12～15の4チャネル）に入力される電圧をSH7050のA／D変換器を使用し、測定します。
（2）図2に示すようにDMACを用いてA／Dデータレジスタ内の測定結果を、ADDR12～15の8バイト毎に4回（計32バイト）RAMに格納します。
4回転送後は転送元（アドレスリロード機能を使用）、転送先を最初のアドレスに戻し、再び転送を繰り返します。転送条件を表1に示します。
（3）A／D変換器の起動はATUによるタイマ割り込みを使用し、起動周期は5msです。
（4）入力する電圧は0～5Vの範囲です。

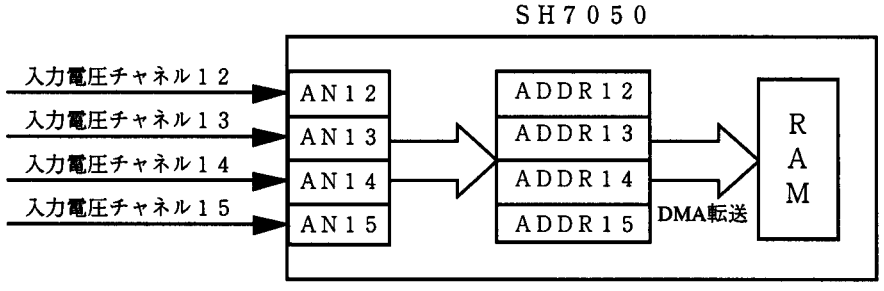


図1　SH7050による電圧の測定ブロック図

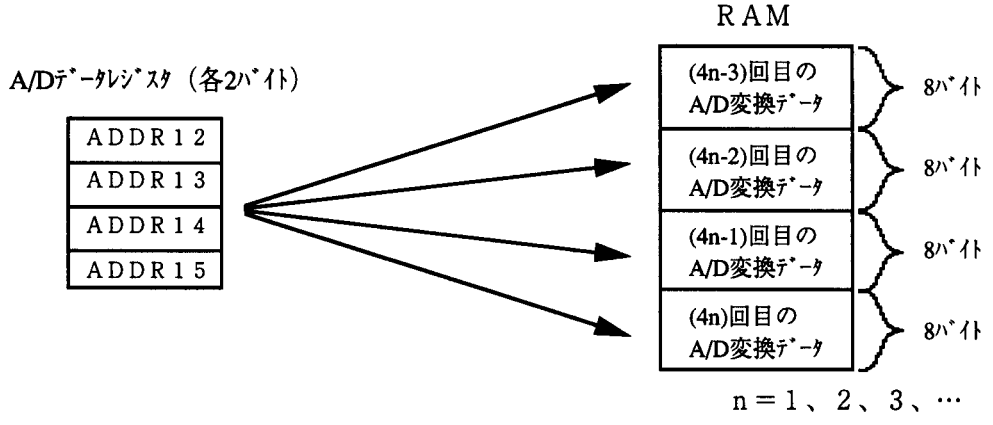


図2　DMACを用いたデータ転送

表1　DMA転送条件

条件項目	内容
DMACﾁｬﾈﾙ	チャネル2
転送元	内蔵A／D変換器
転送先	内蔵RAM
転送回数	16回（リロード回数4回）
転送元アドレス	増加
転送先アドレス	増加
転送要求元	内蔵A／D変換器
バスモード	バースト
転送単位	ワード

使用機能説明

表2、表3に本タスク例の機能割付を示します。表2、表3に示すようにSH7050に内蔵しているDMAC、A／D変換の機能を割付、A／D変換及び変換データのRAMへの転送を行ないます。

表2　DMAC機能割付

DMACレジスタ	機　能
SAR 2	転送元アドレスを設定する。
DAR 2	転送先アドレスを設定する。
TCR 2	転送回数を設定する。
CHCR 2	DMACの動作モード、転送方法等を設定する。
DMAOR	DMACの実行するチャネルの優先順位を設定する。

表3　A／D変換機能割付

A／D変換器レジスタ	機　能
ADCSR1	A／D変換のモード（単一モード／スキャンモード）、 測定端子の選択を設定する。
ADCR1	A／D変換器のクロックの選択、測定の開始及び終了を設定する。
ADDR12～15	A／D変換の結果を設定する。

動作説明

図3に動作原理を示します。図3に示すように、SH7050のハードウェア処理及びソフトウェア処理により4チャネルのA／D変換及び変換データのRAMへの転送を行います。

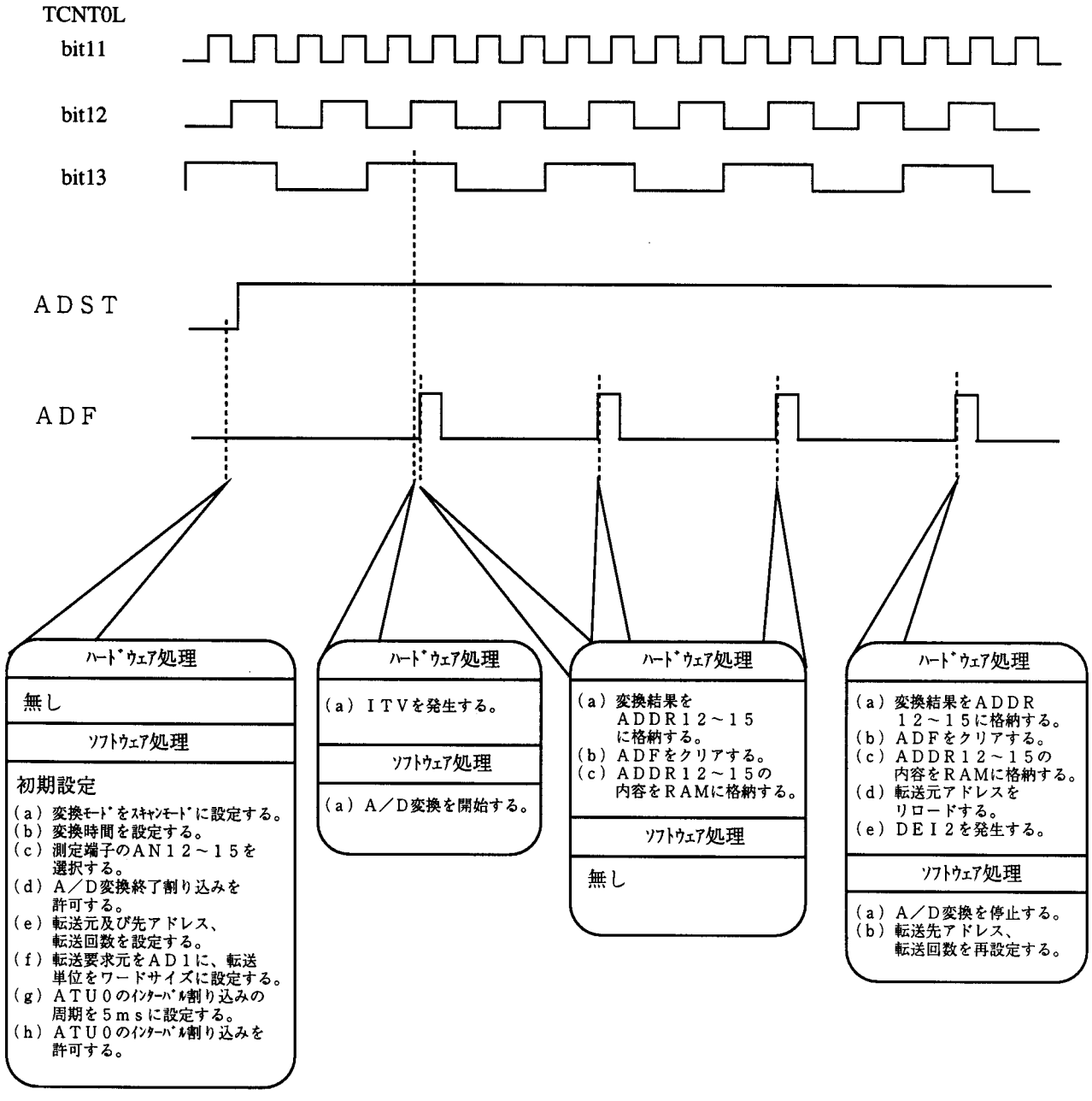


図3　A／D変換動作原理

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
ﾒｲﾝﾙｰﾁﾝ	dma_admn	A／D変換器及びDMACの初期設定を行なう。
転送終了	dma_ad	DEI2で起動し、転送先アドレス、転送回数を再設定する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ
ad__dat0～3	AN12～15に入力した電圧のA／D変換（ｽｷｬﾝﾓｰﾄﾞ（4n-3）回目）結果を格納する。	unsigned short	無し
ad__dat4～7	AN12～15に入力した電圧のA／D変換（ｽｷｬﾝﾓｰﾄﾞ（4n-2）回目）結果を格納する。		
ad__dat8～11	AN12～15に入力した電圧のA／D変換（ｽｷｬﾝﾓｰﾄﾞ（4n-1）回目）結果を格納する。		
ad__dat12～16	AN12～15に入力した電圧のA／D変換（ｽｷｬﾝﾓｰﾄﾞ（4n）回目）結果を格納する。		

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
DMA.SAR2	ADDR12のアドレスを設定する。	&AD.ADDR12	ﾒｲﾝﾙｰﾁﾝ
DMA.DAR2	転送先RAM先頭アドレスを設定する。	&ad_dat0	ﾒｲﾝﾙｰﾁﾝ 転送終了
DMA.TCR2	転送回数（16回）を設定する。	0x10	ﾒｲﾝﾙｰﾁﾝ 転送終了
DMA.CHCR2	DMACの動作モード、転送方法等を設定する。	0x00085f2d	ﾒｲﾝﾙｰﾁﾝ
DMAOR	DMACの起動を設定する。	0x0001	
INT.IPRC	DMA2,ATU01の割り込み優先ﾚﾍﾞﾙを13、15に設定する。	0x0df0	
INT.IPRG	A／D1の割り込み優先ﾚﾍﾞﾙを14に設定する。	0x000e	
C1.TSTR	TCNT0のカウントを開始する。	0x0001	
C1.PSCR1	TCNT0クロックをφ／6に設定する	0x06	
C0.ITVRR	TCNT0の13ﾋﾞｯﾄ目でｲﾝﾀｰﾊﾞﾙ割り込みを発生する。	0x08	
AD.ADCSR1	A／D変換器の変換ﾓｰﾄﾞをｽｷｬﾝﾓｰﾄﾞ、ｸﾛｯｸを134ｽﾃｰﾄ、A／D変換終了割り込み許可、測定端子AN12～15に設定する。	0x5b	

（4）使用RAM

本タスクでは引数以外のRAMは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void dma_admn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
volatile  struct add
{
    short  dat0;                        /* A／D変換データ０                */
    short  dat1;                        /* A／D変換データ１                */
    short  dat2;                        /* A／D変換データ２                */
    short  dat3;                        /* A／D変換データ３                */
    short  dat4;                        /* A／D変換データ４                */
    short  dat5;                        /* A／D変換データ５                */
    short  dat6;                        /* A／D変換データ６                */
    short  dat7;                        /* A／D変換データ７                */
    short  dat8;                        /* A／D変換データ８                */
    short  dat9;                        /* A／D変換データ９                */
    short  dat10;                       /* A／D変換データ１０              */
    short  dat11;                       /* A／D変換データ１１              */
    short  dat12;                       /* A／D変換データ１２              */
    short  dat13;                       /* A／D変換データ１３              */
    short  dat14;                       /* A／D変換データ１４              */
    short  dat15;                       /* A／D変換データ１５              */
};
#define ad (*(struct add *)0xFFFFE800)
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void dma_admn( void )
{
    CO.ITVRR = 0x08;                    /* TCNT0 13ﾋﾞｯﾄ目で割り込み発生(820us)*/
    C1.PSCR1 = 0x06;                    /* ﾌﾟﾘｽｹｰﾗ１段目  φ/6              */
    C1.TSTR = 0x0001;                   /* ﾁｬﾈﾙ0  ｶｳﾝﾄｽﾀｰﾄ                  */
    AD.ADCSR1 = 0x5b;                   /* ｽｷｬﾝﾓｰﾄﾞ,測定端子AN12～15,134ｽﾃｰﾄ */
    AD.ADCR1 = 0x00;                    /* 外部トリガによるA／D変換禁止    */
    DMA.SAR2 = (long)(&AD.ADDR12);      /* 転送元アドレス：A/Dﾃﾞｰﾀﾚｼﾞｽﾀ      */
    DMA.DAR2 = (long)(&ad.dat0);        /* 転送先アドレス：ＲＡＭ          */
    DMA.TCR2 = 0x10;                    /* 転送回数：１６回                */
    DMA.CHCR2 = 0x00085f2d;             /* ﾘｿｰｽｱﾄﾞﾚｽﾘﾛｰﾄﾞ,ｿｰｽ/ﾃﾞｨｽﾄﾈｰｼｮﾝ増加 */
    DMAOR = 0x0001;                     /* ＤＭＡＣ起動許可                */
    INT.IPRC = 0x0df0;                  /* DMA2,ATU01割込優先ﾚﾍﾞﾙを13,15に設定*/
    INT.IPRG = 0x000e;                  /* A/D1割り込み優先ﾚﾍﾞﾙを１４に設定 */
    set_imask(0x0);                     /* 割り込み許可                    */
    while(1);                           /* 無限ループ（割り込み待ち）      */
}
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ＤＭＡ転送終了割り込みルーチン                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( dma_ad )
void  dma_ad( void )
{
    AD.ADCSR1 &= 0xdf;              /* Ａ／Ｄ変換停止                     */
    DMA.CHCR2 &= 0xfffffffc;        /* TEフラグクリア                     */
    DMA.DAR2 = (long)(&ad.dat0);    /* 転送先アドレス：ＲＡＭ             */
    DMA.TCR2 = 0x10;                /* 転送回数：１６回                   */
    DMA.CHCR2 |= 0x00000001;        /* TEフラグクリア                     */
}
/*------------------------------------------------------------------------*/
/*                                                                        */
/* インターバル割り込みルーチン                                           */
/*                                                                        */
/*------------------------------------------------------------------------*/
#pragma interrupt( int5ms )
void int5ms( void )
{
    CO.TSRAH &= 0xf7;
    AD.ADCSR1 |= 0x20;              /* Ａ／Ｄ変換開始                     */
}
#pragma interrupt( ad1 )
void  ad1( void )
{
}
```

2.6　パルスの周期測定（３２ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

仕様

（1）図１に示すように、パルスの周期を測定し、結果をＲＡＭに設定します。

（2）パルスの周期は５μｓから２１４ｓまで５０ｎｓ単位で測定可能です。

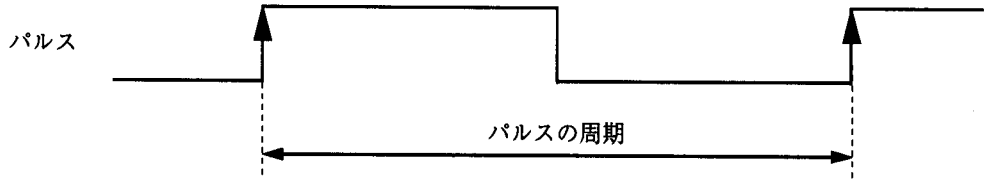


図１　パルス周期測定タイミング

使用機能説明

表１、表２に本タスク例の機能割り付けを示します。本タスク例は表１、表２に示すようにＳＨ７０５０に内蔵されているＡＴＵ、ＰＦＣの機能を割り付け、パルスの周期を測定します。

表１　　ＡＴＵ機能割り付け

ＡＴＵの内蔵機能		機能
端子	TIA0	測定するパルスを入力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TIOR0A	ATUﾁｬﾈﾙ０のｴｯｼﾞ検出を選択する。
	TIERA	ATUﾁｬﾈﾙ０の割り込み要求を設定する
	TSTR	ATUﾁｬﾈﾙ０のカウント開始を設定する。
	ICR0A	入力パルスの立ち上がり時のカウンタ値を検出する。

表２　　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
PEIOR	端子の入出力方向を設定する。
PECR1	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスの周期測定を行います。

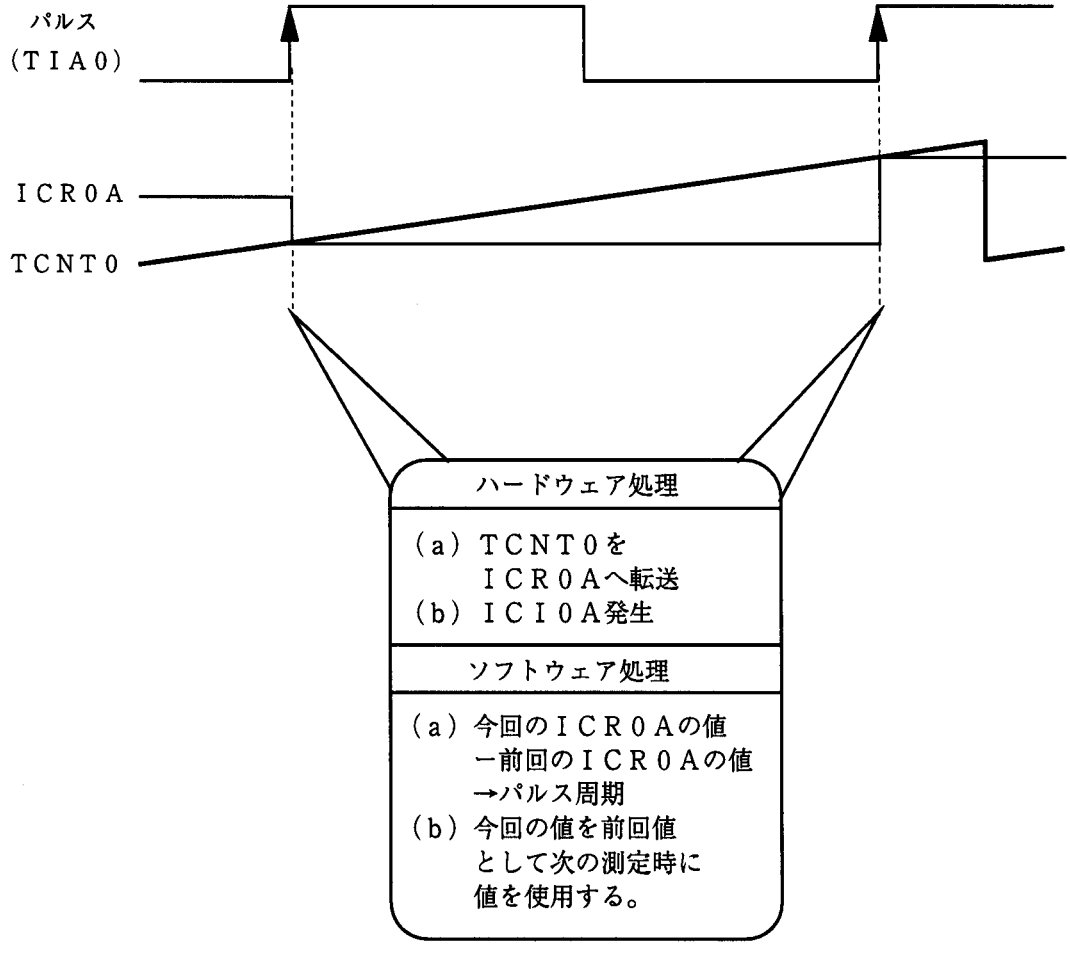


図２　パルス周期計測動作原理

パルスの周期測定（３２ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

ソフトウェア説明

（１）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	ｐｗｍｎ３２	ＡＴＵの初期設定を行なう。
パルスの周期測定	ｐｗｉｎｔ３２	ＩＣＩ０Ａにより起動し、ＩＣＲ０Ａの値からパルスの周期を測定する。

（２）引数の説明

ラベル名	機　能	データ長	使用モジュール名
ref_32	パルスの周期に相当するタイマ値を設定する。 パルスの周期は以下の式にて求まる。 ﾊﾟﾙｽの周期(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns) ×１（ﾌﾟﾘｽｹｰﾗの分周比）	unsigned long	パルスの 周期測定
ref_cntl	入力ﾊﾟﾙｽの立ち上がり時のｶｳﾝﾀ値を格納する。		

（３）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PEIOR	ＰＥ８を入力端子に設定する。	0x0000	ﾒｲﾝﾙｰﾁﾝ
PFC.PECR	端子ﾏﾙﾁﾌﾟﾚｸｽをＴＩＡ０端子の使用に設定する。	0x0010	
C1.PSCR1	ATUﾁｬﾈﾙ０のﾌﾟﾘｽｹｰﾗをφ／１に設定をする。	0x00	
C0.TIOR0A	ATUﾁｬﾈﾙ0のICR0Aを立ち上がりｴｯｼﾞでｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬするようにに設定する。	0x01	
C0.TIERA	ATUﾁｬﾈﾙ０のICF0Aによる割り込み要求を許可する	0x01	
C1.TSTR	ATUﾁｬﾈﾙ０のカウント開始を設定する。	0x0001	
INT.IPRC	ATU01の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	
C0.ICR0AH	入力ﾊﾟﾙｽの立ち上がり時のｶｳﾝﾀ値を設定する。		パルスの周期測定

（４）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

パルスの周期測定（３２ビット）	MCU	SH7050	使用機能	ATU

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwmn32( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define ref_t32   (*(unsigned long  *)0xFFFFE800) /*  パルス幅              */
#define ref_cntl  (*(unsigned long  *)0xFFFFE804) /*  パルス幅ワーク用      */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwmn32( void )
{
    PFC.PEIOR  = 0x0000;            /* PE8を入力端子に設定                   */
    PFC.PECR   = 0x0100;            /*  端子ﾏﾙﾁﾌﾟﾚｸｽをTIA0端子に設定          */
    C1.PSCR1 = 0x00;                /* ﾌﾟﾘｽｹｰﾗ1段目  φ/1                     */
    C0.TIOR0A = 0x01;               /* 立ち上がりｴｯｼﾞでICR0Dへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ   */
    C0.TIERA = 0x01;                /* ICF0Aによる割り込み要求を許可         */
    C1.TSTR = 0x0001;               /* ﾁｬﾈﾙ0  ｶｳﾝﾄｽﾀｰﾄ                       */
    INT.IPRC = 0x000f;              /* ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定   */
    set_imask(0x0);                 /* 割り込み許可                          */
    while(1);                       /* 無限ループ（割り込み待ち）            */
}
/*-------------------------------------------------------------------------*/
/* パルスの周期計測ルーチン                                                */
/*-------------------------------------------------------------------------*/
#pragma interrupt( pwint32 )
void pwint32( void )
{
    C0.TSRAL  &= 0xfe;                          /* ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ   */
    ref_t32 = *(unsigned long *)(&C0.ICR0AH) -
        ref_cntl;                               /* パルスの周期算出          */
    ref_cntl =
            *(unsigned long *)(&C0.ICR0AH);     /* ｷｬﾌﾟﾁｬ値保存              */
}
```

2.7　パルスの周期測定（１６ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

仕様

（1）図１に示すように、パルスの周期を測定し、結果をＲＡＭに設定します。

（2）パルスの周期は５μｓから１３ｍｓまで２００ｎｓ単位で測定可能です。

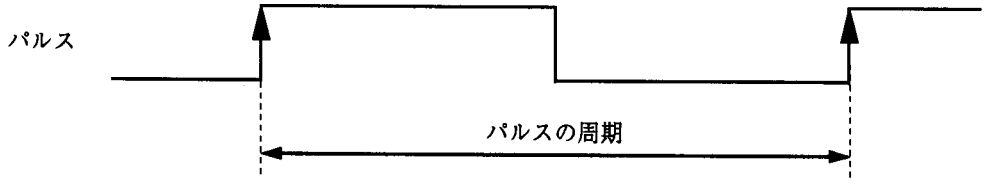


図１　パルス周期測定タイミング

使用機能説明

表１、表２に本タスク例の機能割り付けを示します。本タスク例は表１、表２に示すようにＳＨ７０５０に内蔵されているＡＴＵ及びＰＦＣの機能を割り付け、ＰＷＭを出力します。

表１　　ＡＴＵ機能割り付け

ＡＴＵの内蔵機能		機能
端子	TIOA4	ＰＷＭを出力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TIOR4A	ATUﾁｬﾈﾙ４のレジスタの機能を選択する。
	TIERDL	ATUﾁｬﾈﾙ４の割り込み要求を設定する
	TSTR	ATUﾁｬﾈﾙ４のカウント開始を設定する。
	GR4A	パルスのエッジ検出時のTCNT4を設定する。
	TCR4	ATUﾁｬﾈﾙ４のカウンタのソースクロックを設定する。

表２　　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスの周期測定を行います。

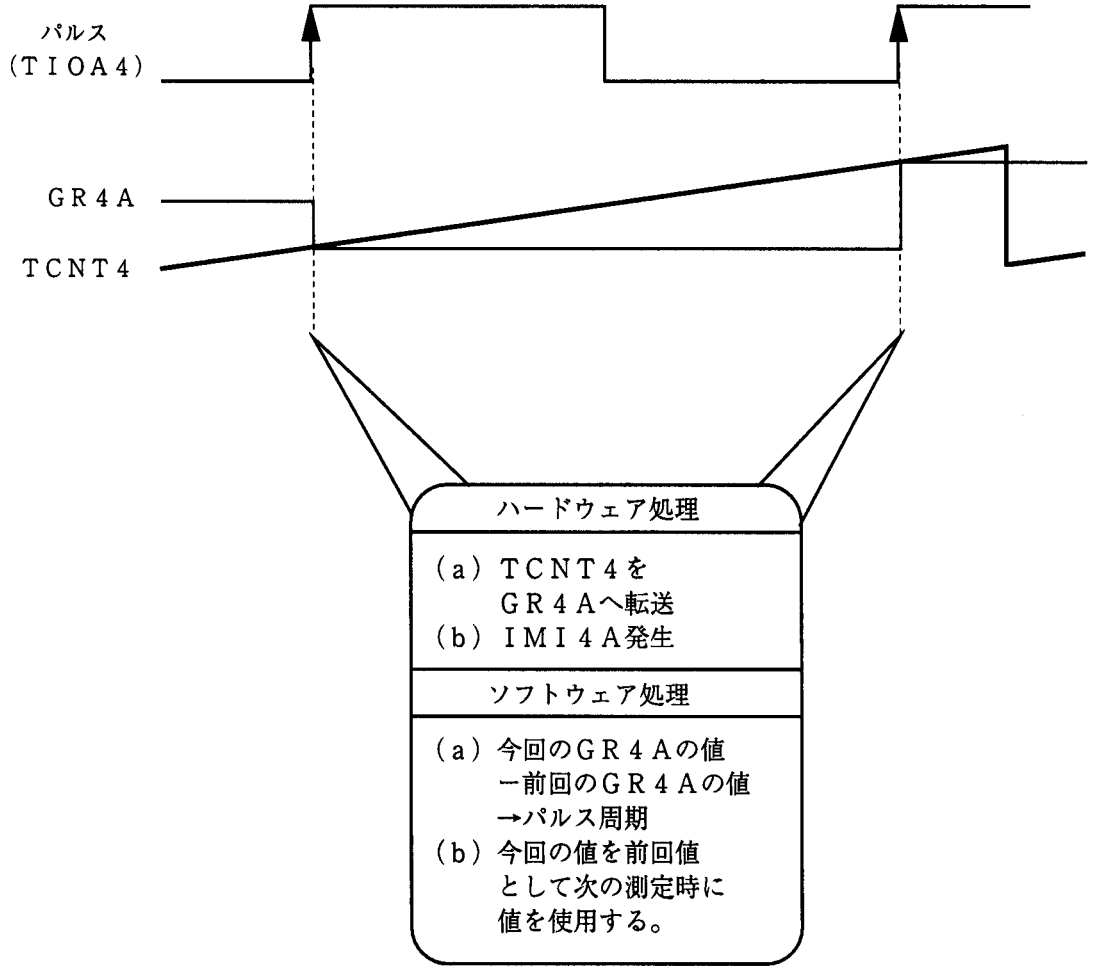


図２　パルス周期計測動作原理

パルスの周期測定（１６ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

ソフトウェア説明

（１）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	ｐｗｍｎ１６	ＡＴＵの初期設定を行なう。
パルスの周期測定	ｐｗｉｎｔ１６	ＩＭＩ４Ａにより起動し、ＧＲ４Ａの値からパルスの周期を測定する。

（２）引数の説明

ラベル名	機能	データ長	使用モジュール名
shu_16	パルスの周期に相当するタイマ値を設定する。 パルスの周期は以下の式にて求める。 ﾊﾟﾙｽの周期(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns) ×４（ﾌﾟﾘｽｹｰﾗの分周比）	unsigned short	パルスの 周期測定
ref_cntw	測定ﾊﾟﾙｽの立ち上がりｴｯｼﾞ検出時のTCNT4を設定する。		

（３）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PGIOR	TG10を入力端子に設定する。	0x0000	ﾒｲﾝﾙｰﾁﾝ
PFC.PGCR1	端子ﾏﾙﾁﾌﾟﾚｸｽをTIOA4端子に設定する。	0x0010	
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／１に設定をする。	0x00	
C35.TCR4	ATUﾁｬﾈﾙ4のﾌﾟﾘｽｹｰﾗをφ／４に設定をする。	0x02	
C35.TIOR4A	ATUﾁｬﾈﾙ4を立ち上がりｴｯｼﾞでICR0Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬするように設定する。	0x05	
C35.TIERDL	ATUﾁｬﾈﾙ4のIME4Aによる割り込み要求を許可する	0x08	
C1.TSTR	ATUﾁｬﾈﾙ4のカウント開始を設定する。	0x0010	
INT.IPRE	ATU41の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	
C35.GR4A	測定パルスの立ち上がりｴｯｼﾞ検出時のTCNT4を設定する。		パルスの周期測定

（４）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwmn16( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define shu_16    (*(unsigned short *)0xFFFFE800) /*  パルス幅             */
#define ref_cntw  (*(unsigned short *)0xFFFFE802) /*  パルス幅ワーク用     */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwmn16( void )
{
    PFC.PGIOR = 0x0000;     /* PG10を入力端子に設定                        */
    PFC.PGCR1 = 0x0010;     /*  端子ﾏﾙﾁﾌﾟﾚｸｽをTIOA4端子に設定              */
    C1.PSCR1 = 0x00;        /* ﾌﾟﾘｽｹｰﾗ１段目  φ/1                         */
    C35.TCR4 = 0x02;        /* ﾁｬﾈﾙ4  内部ｸﾛｯｸ φ'/4 でｶｳﾝﾄ                */
    C35.TIOR4A = 0x05;      /* 立ち上がりｴｯｼﾞでICR4Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ        */
    C35.TIERDL = 0x08;      /* IME4Aによる割り込み要求を許可               */
    C1.TSTR = 0x0010;       /* ﾁｬﾈﾙ4  ｶｳﾝﾄｽﾀｰﾄ                             */
    INT.IPRE = 0x000f;      /* ATU41の割り込み優先ﾚﾍﾞﾙを１５に設定         */
    set_imask(0x0);         /* 割り込み許可                                */
    while(1);               /* 無限ループ（割り込み待ち）                  */
}
/*-------------------------------------------------------------------------*/
/* パルスの周期測定ルーチン                                                */
/*-------------------------------------------------------------------------*/
#pragma interrupt( pwint16 )
void pwint16( void )
{
    C35.TSRDL &= 0xf7;                      /* T4ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ    */
    shu_16  = C35.GR4A - ref_cntw;          /* パルスの周期算出            */
    ref_cntw = C35.GR4A;                    /* ｷｬﾌﾟﾁｬ値保存                */
}
```

2.8　パルスのＨｉｇｈ幅測定（３２ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

仕様

（１）図１に示すようにパルスのＨｉｇｈ幅を測定し、結果をＲＡＭに設定します。

（２）パルスのＨｉｇｈ幅は５μｓから２１４ｓまで５０ｎｓ単位で測定可能です。

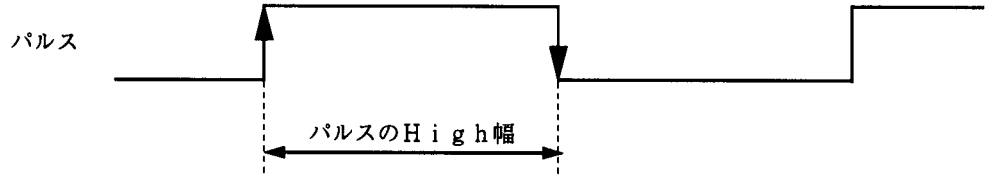


図１　パルスのＨｉｇｈ幅測定タイミング

使用機能説明

表１、表2に本タスク例の機能割り付けを示します。本タスク例は表１、表2に示すようにＳＨ７０５０に内蔵されているＡＴＵ、ＰＦＣの機能を割り付け、パルスのＨｉｇｈ幅を測定します。

表１　　ＡＴＵ機能割り付け

ＡＴＵの内蔵機能		機能
端子	TIA0	測定するパルスを入力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TIOR0A	ATUﾁｬﾈﾙ０のｴｯｼﾞ検出を選択する。
	TIERA	ATUﾁｬﾈﾙ０の割り込み要求を設定する
	TSTR	ATUﾁｬﾈﾙ０のカウント開始を設定する。
	ICR0A	入力パルスの立ち上がり時のカウンタ値を検出する。

表２　　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
PEIOR	端子の入出力方向を設定する。
PECR1	端子の機能を選択する。

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスのＨｉｇｈ幅測定を行います。

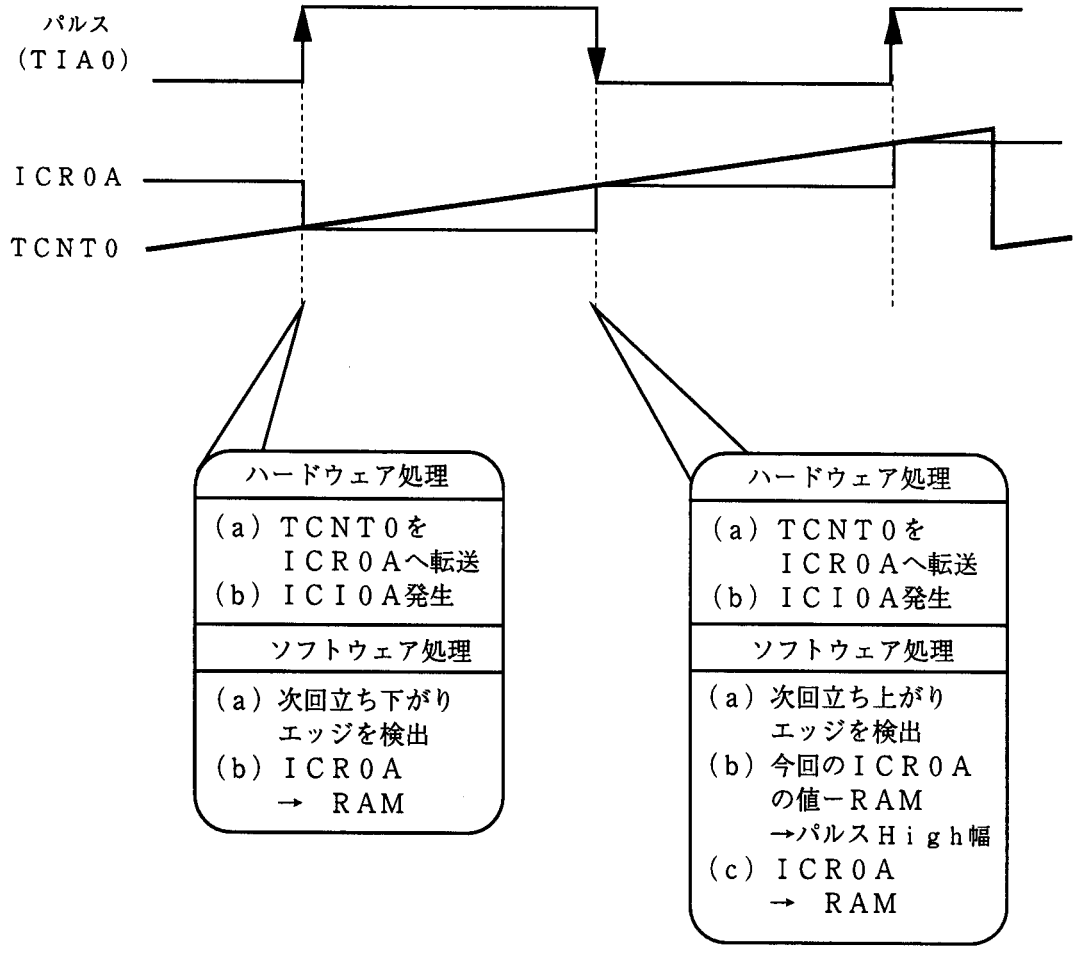


図２　パルスのＨｉｇｈ幅計測動作原理

パルスのＨｉｇｈ幅測定（３２ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

ソフトウェア説明

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	ｐｗｈｍｎ３２	ＡＴＵ及びＲＡＭの初期設定を行なう。
パルスの Ｈｉｇｈ幅測定	ｐｗｈｉｎｔ３２	ＩＣＩ０Ａにより起動し、ＩＣＲ０Ａの値からパルスのＨｉｇｈ幅を測定する。

（2）引数の説明

ラベル名	機能	データ長	使用モジュール名
ｒｅｆ＿ｔ３２	パルスのＨｉｇｈ幅に相当するタイマ値を設定する。 パルスのＨｉｇｈ幅は以下の式にて求める。 ﾊﾟﾙｽのＨｉｇｈ幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns) ×１（ﾌﾟﾘｽｹｰﾗの分周比）	ｌｏｎｇ	パルスの Ｈｉｇｈ幅測定

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PEIOR	ＰＥ８を入力端子に設定する。	0x0000	ﾒｲﾝﾙｰﾁﾝ
PFC.PECR	端子ﾏﾙﾁﾌﾟﾚｸｽをＴＩＡ０端子に設定する。	0x0010	
C1.PSCR1	ATUﾁｬﾈﾙ０のﾌﾟﾘｽｹｰﾗをφ／１に設定をする。	0x00	
C0.TIOR0A	ATUﾁｬﾈﾙ0のICR0Aを立ち上がりｴｯｼﾞでｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬするようにに設定する。	0x01	
C0.TIERA	ATUﾁｬﾈﾙ０のICF0Aによる割り込み要求を許可する	0x01	
C1.TSTR	ATUﾁｬﾈﾙ０のカウント開始を設定する。	0x0001	
INT.IPRC	ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	
C0.ICR0AH	入力ﾊﾟﾙｽの立ち上がり時のｶｳﾝﾀ値を格納する。		パルスの Ｈｉｇｈ幅測定

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                    */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ                */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwhmn32( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define ref_t32   (*(unsigned long  *)0xFFFFE800) /*  ﾊﾟﾙｽHigh幅            */
#define ref_cntl  (*(unsigned long  *)0xFFFFE804) /*  ﾊﾟﾙｽHigh幅ﾜｰｸ用       */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwhmn32( void )
{
    PFC.PEIOR  = 0x0000;            /* PE8を入力端子に設定                   */
    PFC.PECR   = 0x0100;            /*  端子ﾏﾙﾁﾌﾟﾚｸｽをTIO0端子に設定          */
    C1.PSCR1 = 0x00;                /* ﾌﾟﾘｽｹｰﾗ1段目  φ/1                     */
    C0.TIOR0A = 0x01;               /* 立ち上がりｴｯｼﾞでICR0Dへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ    */
    C0.TIERA = 0x01;                /* ICF0Aによる割り込み要求を許可         */
    C1.TSTR = 0x0001;               /* ﾁｬﾈﾙ0  ｶｳﾝﾄｽﾀｰﾄ                       */
    INT.IPRC = 0x000f;              /* ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定   */
    set_imask(0x0);                 /* 割り込み許可                          */
    while(1);                       /* 無限ループ（割り込み待ち）            */
}
/*-------------------------------------------------------------------------*/
/* パルスのＨｉｇｈ幅測定ルーチン                                          */
/*-------------------------------------------------------------------------*/
#pragma interrupt( pwhint32 )
void pwhint32( void )
{
    C0.TSRAL  &= 0xfe;                          /* ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ      */
    if(( C0.TIOR0A & 0x01 ) == 1 )              /* 立ち上がりエッジか？        */
    {
        C0.TIOR0A   = 0x02;                     /* 次回立ち下がりｴｯｼﾞでﾁｬﾌﾟﾁｬ */
    }
    else
    {
        C0.TIOR0A   = 0x01;                     /* 次回立ち上がりｴｯｼﾞでﾁｬﾌﾟﾁｬ */
        ref_t32 = *(unsigned long *)(&C0.ICR0AH) -
            ref_cntl;                           /* パルスのＨｉｇｈ幅算出      */
    }
    ref_cntl =
        *(unsigned long *)(&C0.ICR0AH) ;        /* ｷｬﾌﾟﾁｬ値保存               */
}
```

2.9 パルスのＨｉｇｈ幅測定（１６ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

仕様

（1）図１に示すように、パルスのＨｉｇｈ幅を測定し、結果をＲＡＭに設定します。

（2）パルスのＨｉｇｈ幅は５μｓから１３ｍｓまで２００ｎｓ単位で測定可能です。

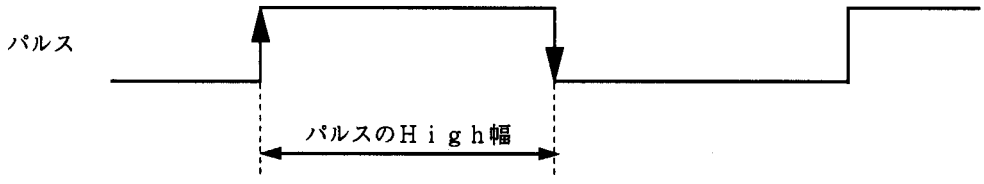


図１　パルスのＨｉｇｈ幅測定タイミング

使用機能説明

表１、表２に本タスク例の機能割り付けを示します。本タスク例は表１、表２に示すようにＳＨ７０５０に内蔵されているＡＴＵ及びＰＦＣの機能を割り付け、パルスのＨｉｇｈ幅を測定します。

表１　　ＡＴＵ機能割り付け

ＡＴＵの内蔵機能		機能
端子	TIOA4	ＰＷＭを出力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TIOR4A	ATUﾁｬﾈﾙ４のレジスタの機能を選択する。
	TIERDL	ATUﾁｬﾈﾙ４の割り込み要求を設定する
	TSTR	ATUﾁｬﾈﾙ４のカウント開始を設定する。
	GR4A	パルスのエッジ検出時のTCNT4を設定する。
	TCR4	ATUﾁｬﾈﾙ４のカウンタのソースクロックを設定する。

表２　　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスのＨｉｇｈ幅測定を行います。

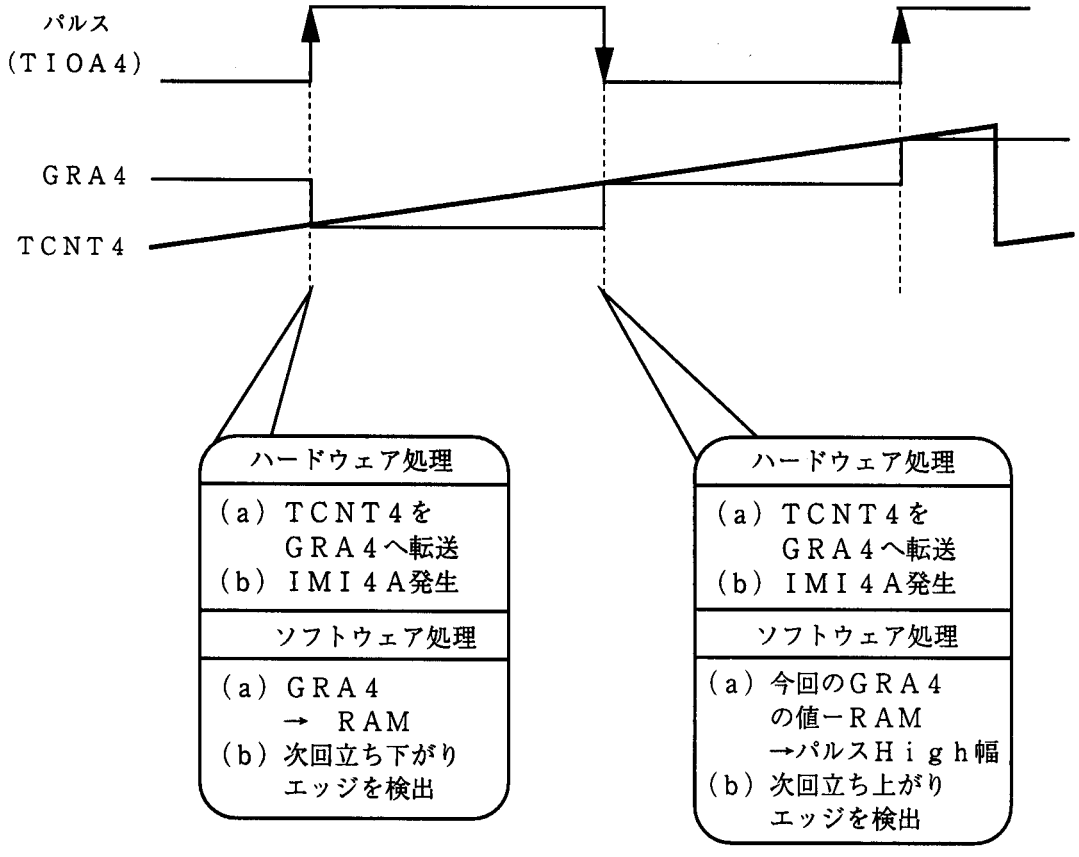


図２　パルスのＨｉｇｈ幅計測動作原理

パルスのＨｉｇｈ幅測定（１６ビット）	MCU	ＳＨ７０５０	使用機能	ＡＴＵ

ソフトウェア説明

（１）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	ｐｗｈｍｎ１６	ＡＴＵ及びＲＡＭの初期設定を行なう。
パルスの Ｈｉｇｈ幅測定	ｐｗｈｉｎｔ１６	ＩＭＩ４Ａにより起動し、ＧＲＡ４の値からパルスのＨｉｇｈ幅を測定する。

（２）引数の説明

ラベル名	機　能	データ長	使用モジュール名
shu_16	パルスのＨｉｇｈ幅に相当するタイマ値を設定する。 パルスのＨｉｇｈ幅は以下の式にて求める。 ﾊﾟﾙｽのＨｉｇｈ幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns) ×４（ﾌﾟﾘｽｹｰﾗの分周比）	unsigned short	パルスの Ｈｉｇｈ幅測定
ref_cntw	測定パルスの立ち上がりｴｯｼﾞ検出時のTCNT4を格納する。		

（３）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PGIOR	TG10を入力端子に設定する。	0x0000	ﾒｲﾝﾙｰﾁﾝ
PFC.PGCR1	端子ﾏﾙﾁﾌﾟﾚｸｽをTIOA4端子の使用に設定する。	0x0010	
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／１に設定をする。	0x00	
C35.TCR4	ATUﾁｬﾈﾙ4のﾌﾟﾘｽｹｰﾗをφ／４に設定をする。	0x02	
C35.TIOR4A	ATUﾁｬﾈﾙ4を立ち上がりｴｯｼﾞでICR0Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬするように設定する。	0x05	
C35.TIERDL	ATUﾁｬﾈﾙ4のIME4Aによる割り込み要求を許可する	0x08	
C1.TSTR	ATUﾁｬﾈﾙ4のカウント開始を設定する。	0x0010	
INT.IPRE	ATU41の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	
C35.GR4A	測定パルスの立ち上がりｴｯｼﾞ検出時のTCNT4を設定する。		パルスの Ｈｉｇｈ幅測定

（４）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwhmn16( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define shu_16    (*(unsigned short *)0xFFFFE800) /*  パルスＨｉｇｈ幅      */
#define ref_cntw  (*(unsigned short *)0xFFFFE802) /*  パルス幅ワーク用      */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwhmn16( void )
{
    PFC.PGCR1 = 0x0010;     /* ＴＩＯＡ４端子を使用                        */
    C1.PSCR1 = 0x00;        /* ﾌﾟﾘｽｹｰﾗ１段目　φ/1                         */
    C35.TCR4 = 0x02;        /* ﾁｬﾈﾙ4　内部ｸﾛｯｸ φ' /4 でｶｳﾝﾄ               */
    C35.TIOR4A = 0x05;      /* 立ち上がりｴｯｼﾞでICR4Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ          */
    C35.TIERDL = 0x08;      /* IME4Aによる割り込み要求を許可                */
    C1.TSTR = 0x0010;       /* ﾁｬﾈﾙ4　ｶｳﾝﾄｽﾀｰﾄ                             */
    INT.IPRE = 0x000f;      /* ATU41の割り込み優先ﾚﾍﾞﾙを１５に設定          */
    set_imask(0x0);         /* 割り込み許可                                 */
    while(1);               /* 無限ループ（割り込み待ち）                   */
}
/*-------------------------------------------------------------------------*/
/* パルスのＨｉｇｈ幅測定ルーチン                                          */
/*-------------------------------------------------------------------------*/
#pragma interrupt( pwhint16 )
void pwhint16( void )
{
    C35.TSRDL &= 0xf7;                          /* T4ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ    */
    if(( C35.TIOR4A & 0x05 ) == 0x05 )          /* 立ち上がりエッジか？        */
    {
        C35.TIOR4A = 0x06;                      /* 次回立ち下がりｴｯｼﾞでﾁｬﾌﾟﾁｬ */
    }
    else
    {
        C35.TIOR4A = 0x05;          /* 立ち上がりｴｯｼﾞでICR4Aへｲﾝﾌﾟｯﾄｷｬﾌﾟｷｬ    */
        shu_16  = C35.GR4A - ref_cntw;          /* パルスのＨｉｇｈ幅算出      */
    }
    ref_cntw = C35.GR4A;                        /* ｷｬﾌﾟﾁｬ値保存                */
}
```

2.10　ワンショットパルス出力（ﾁｬﾈﾙ1、10連動）	MCU	SH7050	使用機能	ATU

仕様

（1）図1に示すように外部信号の立ち上がりに同期してワンショットパルスを出力します。オフセットは外部クロックカウント値、パルス幅は内部クロックカウント値を設定します。外部信号の立ち上がり時にオフセット及びパルス幅を設定し、割り込み処理不要でハードウェアのみでパルスを出力します。

（2）外部信号の立ち上がりからのオフセットおよびパルス幅は以下に示す範囲で可変できます。

外部クロック×1＜オフセット＜外部クロック×65536 ＊注

50ns≦パルス幅＜3．28ms

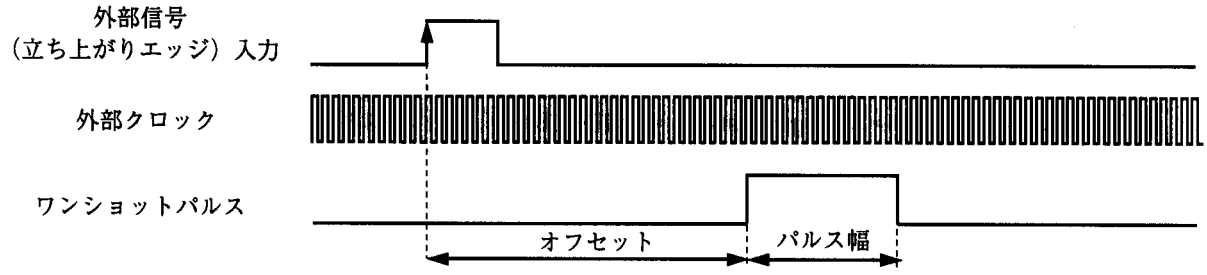


図1　ワンショットパルス出力

＊注：ｵﾌｾｯﾄは外部信号の周期より小さいこと

使用機能説明

表1、表2に本タスク例の機能割り付けを示します。本タスク例は表1、表2に示すようにSH7050に内蔵しているATU、PFCの機能を割り付け、ワンショットパルスを出力します。

表1　ATU機能割り付け

ATUの内蔵機能		機能
端子	TIA0	外部信号を入力する。
	TOA10	ワンショットパルスを出力する。
	TCLKA	外部クロックを入力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TCR1	TCNT1のクロックソースを設定する。
	TCNR	DEC10とOFF1の接続有無を設定する。
	OSBR	外部信号の立ち上がり時のカウンタ値を検出する。
	GR1A	ワンショットパルスのオフセットの設定をする。
	DCNT10A	ワンショットパルスのパルス幅を設定する。

表2　PFC機能割り付け

PFCレジスタ	機能
PBIOR	端子の入出力方向を設定する。
PBCR	端子の機能を選択する。
PCIOR	端子の入出力方向を設定する。
PCCR2	端子の機能を選択する。
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理により
ワンショットパルスを出力します。

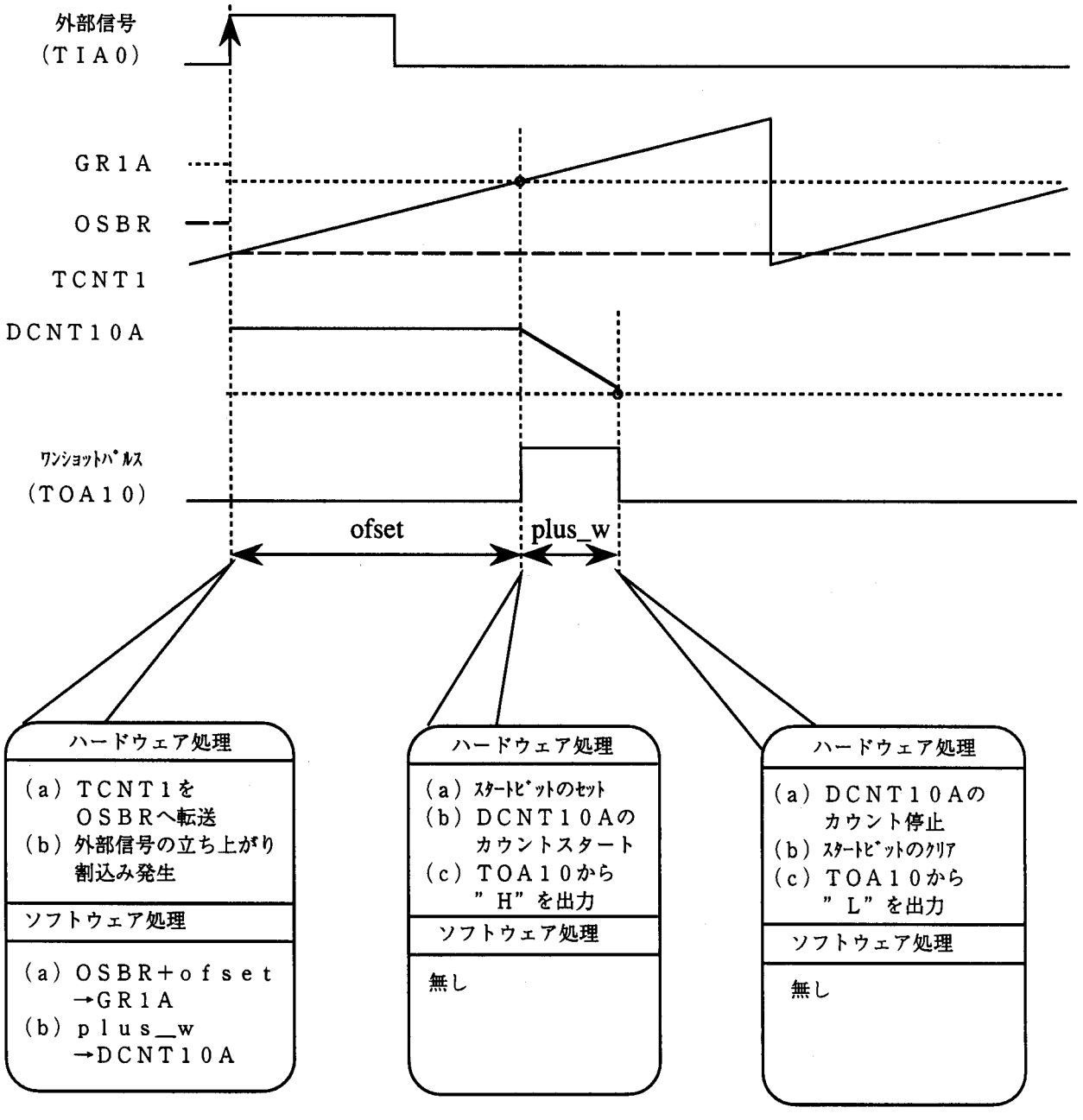


図２　ワンショットパルス出力動作原理

ワンショットパルス出力（ﾁｬﾈﾙ1、10連動）	MCU	SH7050	使用機能	ATU

ソフトウェア説明

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	oneofmn	ATUの初期設定を行なう。
ワンショット パルス出力	oneofpout	オフセットおよびパルス幅をGR1A、DCNT10Aに設定し、 ワンショットパルスを出力する。

（2）引数の説明

ラベル名	機能	データ長	使用ﾓｼﾞｭｰﾙ名
ofset	ワンショットパルスのオフセットに相当するタイマ値を設定する。 オフセットは以下の式にて求める。 オフセット=ﾀｲﾏ値×外部クロック	unsigned short	ﾒｲﾝﾙｰﾁﾝ
plus_w	ワンショットパルスのパルス幅に相当するタイマ値を設定する。 パルス幅は以下の式にて求める。 パルス幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)		ﾜﾝｼｮｯﾄ ﾊﾟﾙｽ出力

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／1に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TCR1	ATUﾁｬﾈﾙ1のﾌﾟﾘｽｹｰﾗを外部ｸﾛｯｸに設定をする。	0x26	
C1.TCR10	ATUﾁｬﾈﾙ10のﾌﾟﾘｽｹｰﾗをφ／16に設定をする。	0x40	
C0.TIOR0A	ATUﾁｬﾈﾙ0を立ち上がりｴｯｼﾞでICR0Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ するように設定する。	0x01	
C1.TIOR1A	ATUﾁｬﾈﾙ4のGR1Aをｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀに設定する。	0x00	
C0.TIERA	ATUﾁｬﾈﾙ0のICF0Aによる割り込み要求を許可する	0x01	
C1.TCNR	DEC10AとOFF1Aの接続を許可する	0x01	
C1.TSTR	ATUﾁｬﾈﾙ0、ﾁｬﾈﾙ1のカウント開始を設定する。	0x0003	
PFC.PBCR	端子ﾏﾙﾁﾌﾟﾚｸｽをＴＣＬＫの使用にする。	0x0010	
PFC.PCIOR	PC7を出力に設定する。	0x0080	
PFC.PCCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをTOA10の使用にする。	0x4000	
PFC.PECR	端子ﾏﾙﾁﾌﾟﾚｸｽをTIA0の使用にする。	0x0100	
INT.IPRC	ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	
C1.GR1A	ﾜﾝｼｮｯﾄﾊﾟﾙｽのｵﾌｾｯﾄを設定する。		ワンショット パルス出力
C1.OSBR	ATUﾁｬﾈﾙ0のｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬでTCNT1を設定する。		
C10.DCN10A	ﾜﾝｼｮｯﾄﾊﾟﾙｽのﾊﾟﾙｽ幅を設定する。		

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void oneofmn(void);
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define ofset     (*(unsigned short *)0xFFFFE800)  /*  オフセット           */
#define plus_w    (*(unsigned short *)0xFFFFE802)  /*  パルス幅             */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void oneofmn( void )
{
    PFC.PBCR =  0x0010;     /* ＡＴＵｸﾛｯｸ入力端子を使用する                */
    PFC.PCCR2 =  0x4000;    /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＯＡ１０の使用にする。       */
    PFC.PCIOR = 0x0080;     /* ＰＣ７を出力に設定する。                     */
    PFC.PECR  = 0x0100;     /* 端子ﾏﾙﾁﾌﾟﾚｸｽをTIA0端子の使用にする。         */
    C1.PSCR1 = 0x00;        /* ﾌﾟﾘｽｹｰﾗ１段目  φ/1                          */
    C1.TCR1 = 0x26;         /* 外部ｸﾛｯｸ A端子入力  両ｴｯｼﾞでｶｳﾝﾄ            */
    C1.TCR10 = 0x40;        /* ﾁｬﾈﾙ10  内部ｸﾛｯｸ φ'/１６でｶｳﾝﾄ             */
    C0.TIOR0A = 0x01;       /* 立ち上がりｴｯｼﾞでICR0Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ          */
    C1.TIOR1A = 0x00;       /* ＧＲ１Ａはｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽ                 */
    C0.TIERA = 0x01;        /* ICF0Aによる割り込み要求を許可する。          */
    C1.TCNR = 0x01;         /* DEC10AとOFF1Aの接続を許可する。              */
    C1.TSTR = 0x0003;       /* ﾁｬﾈﾙ0,ﾁｬﾈﾙ1  ｶｳﾝﾄｽﾀｰﾄ                      */
    INT.IPRC = 0x000f;      /* ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定する。     */
    ofset  = 0x0020;        /* オフセット ＝  H' ００２０                   */
    plus_w = 0x0400;        /* パルス幅   ＝  H' ０４００                   */
    set_imask(0x0);         /* 割り込み許可                                 */
    while(1);               /* 無限ループ（割り込み待ち）                   */
}
/*-------------------------------------------------------------------------*/
/* ワンショットパルス出力ルーチン                                          */
/*-------------------------------------------------------------------------*/
#pragma interrupt( oneofpout )
void oneofpout( void )
{
    C0.TSRAL  &= 0xfe;                  /* ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ           */
    C1.GR1A = C1.OSBR + ofset;          /* オフセット設定                       */
    C10.DCNT10A = plus_w;               /* パルス幅設定                         */
}
```

2.11　6端子同時ﾜﾝｼｮｯﾄ出力（ﾁｬﾈﾙ1、10非連動）	MCU	SH7050	使用機能	ATU

仕様

（1）図1に示すように外部信号の立ち上がりに同期してﾜﾝｼｮｯﾄﾊﾟﾙｽを6端子から同時に出力します。
　　オフセットは外部クロックカウント値、パルス幅は内部クロックカウント値を設定します。
（2）外部信号の立ち上がりからのオフセットおよびパルス幅は以下に示す範囲で可変できます。
　　外部クロック×1＜オフセット＜外部クロック×65536　＊注
　　50ns≦パルス幅＜3．28ms

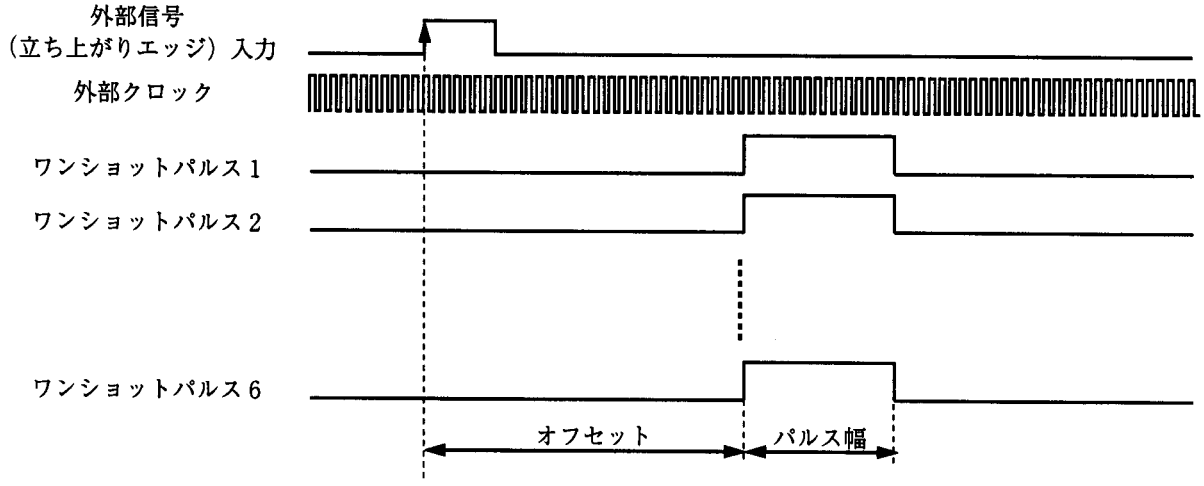


図1　ワンショットパルス出力

＊注：ｵﾌｾｯﾄは外部信号の周期より小さいこと

使用機能説明

表1、表2に本タスク例の機能割り付けを示します。本タスク例は表1、表2に示すようにSH7050に内蔵しているATU、PFCの機能を割り付け、ワンショットパルスを出力します。

表1　ATU機能割り付け

ATUの内蔵機能		機能
端子	TIA0	外部信号を入力する。
	TOA10～F10	ワンショットパルスを出力する。
	TCLKA	外部クロックを入力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TCR1	TCNT1のクロックソースを設定する。
	TCNR	DEC10とOFF1の接続有無を設定する。
	OSBR	外部信号の立ち上がり時のカウンタ値を検出する。
	GR1A	ワンショットパルスのオフセットの設定をする。
	DCNT10A～10F	ワンショットパルスのパルス幅を設定する。

表2　PFC機能割り付け

PFCレジスタ	機能
PBIOR	端子の入出力方向を設定する。
PBCR	端子の機能を選択する。
PCIOR	端子の入出力方向を設定する。
PCCR1,2	端子の機能を選択する。
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

図2に動作原理を示します。図2に示すようにSH7050のハードウェア処理及びソフトウェア処理により
ワンショットパルスを出力します。

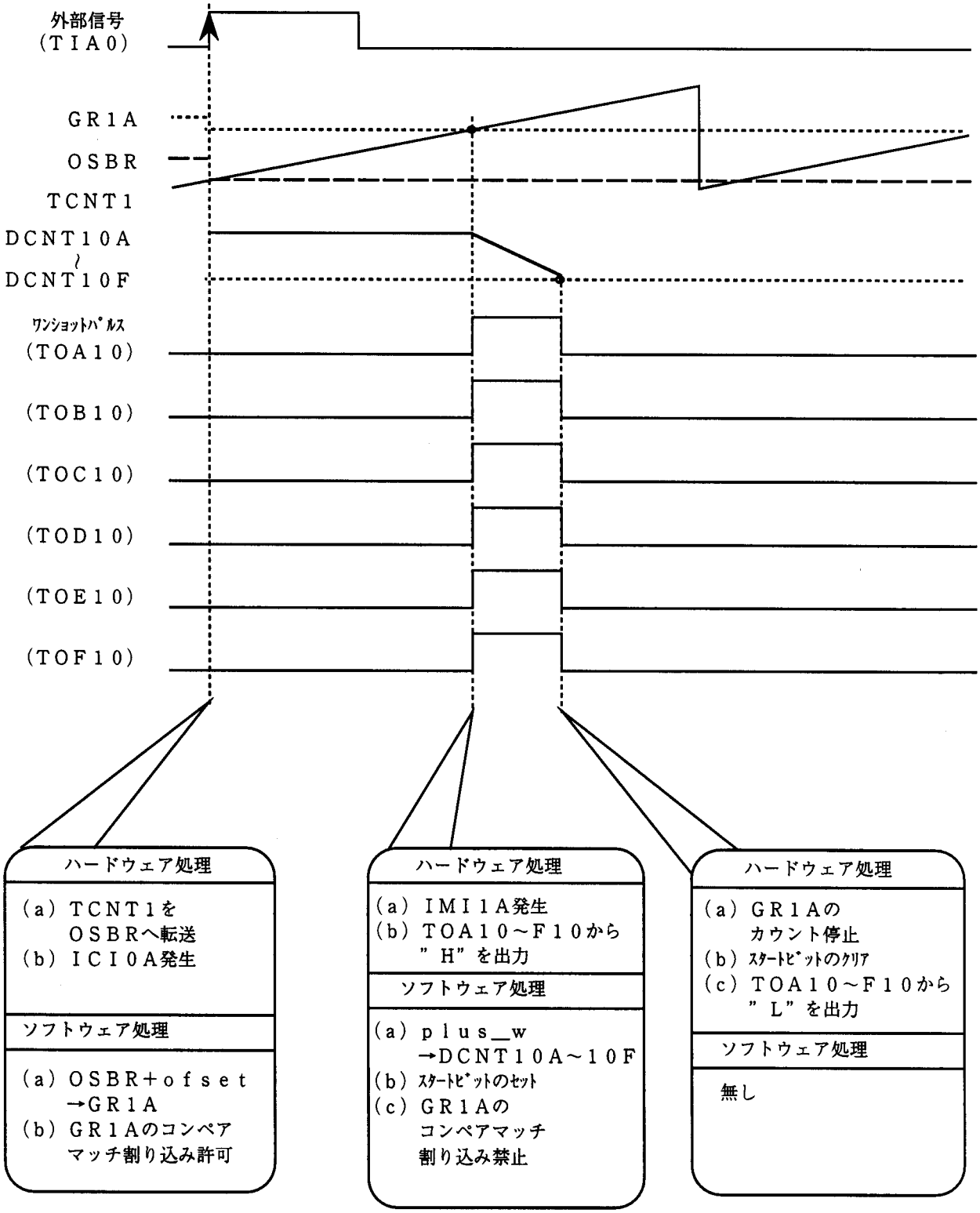


図2　6端子同時ワンショットパルス出力動作原理

6端子同時ﾜﾝｼｮｯﾄ出力（ﾁｬﾈﾙ1、10非連動）	MCU	SH7050	使用機能	ATU

ソフトウェア説明

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	oneofmn	ＡＴＵの初期設定を行なう。
ﾜﾝｼｮｯﾄｵﾌｾｯﾄ設定	setoneof	オフセットをＧＲ１Ａに設定し、コンペアマッチ割り込みを発生させる。
ﾜｼｮｯﾄﾊﾟﾙｽ出力	onepout	パルス幅をＤＣＮＴ１０Ａに設定し、ワンショットパルスを出力する。

（2）引数の説明

ラベル名	機能	データ長	使用ﾓｼﾞｭｰﾙ名
ｏｆｓｅｔ	ワンショットパルスのオフセットに相当するタイマ値を設定する。 オフセットは以下の式にて求める。 オフセット=ﾀｲﾏ値×外部クロック	unsigned short	ﾒｲﾝﾙｰﾁﾝ ﾜﾝｼｮｯﾄ ｵﾌｾｯﾄ設定
ｐｌｕｓ＿ｗ	ワンショットパルスのパルス幅に相当するタイマ値を設定する。 パルス幅は以下の式にて求める。 パルス幅(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns)	unsigned short	ﾒｲﾝﾙｰﾁﾝ ﾜﾝｼｮｯﾄ ﾊﾟﾙｽ出力

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／１に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TCR1	ATUﾁｬﾈﾙ1のﾌﾟﾘｽｹｰﾗを外部ｸﾛｯｸに設定をする。	0x26	ﾒｲﾝﾙｰﾁﾝ
C1.TCR10	ATUﾁｬﾈﾙ10のﾌﾟﾘｽｹｰﾗをφ／１６に設定をする。	0x40	ﾒｲﾝﾙｰﾁﾝ
C0.TIOR0A	ATUﾁｬﾈﾙ0を立ち上がりｴｯｼﾞでICR0Aへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬするように設定する。	0x01	ﾒｲﾝﾙｰﾁﾝ
C1.TIOR1A	ATUﾁｬﾈﾙ4のGR1Aをｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀに設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
C0.TIERA	ATUﾁｬﾈﾙ0のICF0Aによる割り込み要求を許可する	0x01	ﾒｲﾝﾙｰﾁﾝ
C0.TIERB	ATUﾁｬﾈﾙ1のIMF1Aによる割り込み要求を許可する	0x01	ワンショット オフセット設定
C1.TCNR	DEC10AとOFF1Aの接続を禁止する	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TSTR	ATUﾁｬﾈﾙ0、ﾁｬﾈﾙ1のカウント開始を設定する。	0x0003	ﾒｲﾝﾙｰﾁﾝ
PFC.PBCR	端子ﾏﾙﾁﾌﾟﾚｸｽをＴＣＬＫの使用にする。	0x0010	ﾒｲﾝﾙｰﾁﾝ
PFC.PCIOR	PC7を出力に設定する。	0x9f80	ﾒｲﾝﾙｰﾁﾝ
PFC.PCCR1	端子ﾏﾙﾁﾌﾟﾚｸｽをTOB10-F10の使用にする。	0xc155	ﾒｲﾝﾙｰﾁﾝ
PFC.PCCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをTOA10の使用にする。	0x4000	ﾒｲﾝﾙｰﾁﾝ
PFC.PECR	端子ﾏﾙﾁﾌﾟﾚｸｽをTIA0の使用にする。	0x0100	ﾒｲﾝﾙｰﾁﾝ
INT.IPRC	ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	ﾒｲﾝﾙｰﾁﾝ
INT.IPRD	ATU11の割り込み優先ﾚﾍﾞﾙを１４に設定する。	0x0e00	ﾒｲﾝﾙｰﾁﾝ
C1.GR1A	ﾜﾝｼｮｯﾄﾊﾟﾙｽのｵﾌｾｯﾄを設定する。		ワンショット オフセット設定
C1.OSBR	ATUﾁｬﾈﾙ0のｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬでTCNT1を設定する。		ワンショット オフセット設定
C10.DCN10A～10F	ﾜﾝｼｮｯﾄﾊﾟﾙｽのﾊﾟﾙｽ幅を設定する。		ワンショット パルス出力

（4）使用ＲＡＭ

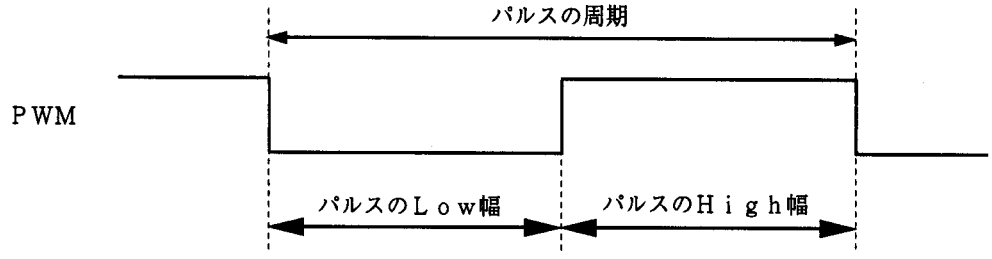
本タスクでは引数以外のＲＡＭは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                    */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ                */
/*-------------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                          */
/*-------------------------------------------------------------------------------*/
void oneofmn( void );
/*-------------------------------------------------------------------------------*/
/* 変数定義                                                                      */
/*-------------------------------------------------------------------------------*/
#define ofset     (*(unsigned short *)0xFFFFE800) /*  オフセット                 */
#define plus_w    (*(unsigned short *)0xFFFFE802) /*  パルス幅                   */
/*-------------------------------------------------------------------------------*/
/* メインルーチン                                                                */
/*-------------------------------------------------------------------------------*/
void oneofmn( void )
{
    PFC.PBCR =  0x0010;     /* ＡＴＵｸﾛｯｸ入力端子を使用する                     */
    PFC.PCCR1 |=  0xc155;   /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＯＢ１０～Ｆ１０の使用にする。*/
    PFC.PCCR2 |=  0x4000;   /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＯＡ１０の使用にする。           */
    PFC.PCIOR = 0x9f80;     /* ＰＣ７～１２を出力に設定する。                    */
    PFC.PECR  = 0x0100;     /* 端子ﾏﾙﾁﾌﾟﾚｸｽをTIA0端子の使用にする。             */
    C1.PSCR1 = 0x00;        /* ﾌﾟﾘｽｹｰﾗ１段目  φ/1                               */
    C1.TCR1 = 0x26;         /* 外部ｸﾛｯｸ A端子入力  両ｴｯｼﾞでｶｳﾝﾄ                 */
    C1.TCR10 = 0x40;        /* ﾁｬﾈﾙ10  内部ｸﾛｯｸ φ'/１６でｶｳﾝﾄ                  */
    C0.TIOR0A = 0x01;       /* 立ち上がりｴｯｼﾞでICR0Dへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ             */
    C1.TIOR1A = 0x00;       /* ＧＲ１Ａはｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽ                      */
    C0.TIERA = 0x01;        /* ICF0Aによる割り込み要求を許可                     */
    C1.TCNR = 0x00;         /* DEC10AとOFF1Aの接続を禁止                         */
    C1.TSTR = 0x0003;       /* ﾁｬﾈﾙ0,ﾁｬﾈﾙ1  ｶｳﾝﾄｽﾀｰﾄ                           */
    INT.IPRC = 0x000f;      /* ATU01の割り込み優先ﾚﾍﾞﾙを１５に設定               */
    INT.IPRD = 0x0e00;      /* ATU11の割り込み優先ﾚﾍﾞﾙを１４に設定               */
    ofset  = 0x0020;        /* オフセット ＝  Ｈ'０２００                        */
    plus_w = 0x0400;        /* パルス幅   ＝  Ｈ'０８００                        */
    set_imask(0x0);         /* 割り込み許可                                      */
    while(1);               /* 無限ループ（割り込み待ち）                        */
}
/*-------------------------------------------------------------------------------*/
/* ワンショットオフセット設定ルーチン                                            */
/*-------------------------------------------------------------------------------*/
#pragma interrupt( oneofpout )
void oneofpout( void )
{
    C0.TSRAL  &= 0xfe;                  /* ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ              */
    C1.GR1A = C1.OSBR + ofset;          /* オフセット設定                        */
    C1.TIERB = 0x01;                    /* ICF0Bによる割り込み要求を許可         */
}
/*-------------------------------------------------------------------------------*/
/* ワンショットパルス出力ルーチン                                                */
/*-------------------------------------------------------------------------------*/
#pragma interrupt( onepout )
void onepout( void )
{
    C1.TSRB  &= 0xfe;                   /* IMF1Aのｸﾘｱ                           */
    C10.DCNT10A = plus_w;               /* パルス幅設定                          */
    C10.DCNT10B = plus_w;               /* パルス幅設定                          */
    C10.DCNT10C = plus_w;               /* パルス幅設定                          */
    C10.DCNT10D = plus_w;               /* パルス幅設定                          */
    C10.DCNT10E = plus_w;               /* パルス幅設定                          */
    C10.DCNT10F = plus_w;               /* パルス幅設定                          */
    C1.DCER = 0x3f;                     /* DCNT10Aのダウンカウントを許可         */
    C1.TIERB = 0x00;                    /* ICF0Bによる割り込み要求を禁止         */
}
```

2.12　PWM出力（汎用）	MCU	SH7050	使用機能	ATU

仕様

（1）図1に示すように、デューティ及び周期を変化できるパルスを出力します。
（2）20MHzで動作時、出力するパルスの周期は5μsから6．5msの間で任意に設定出来ます。



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

図1　PWM出力タイミング

使用機能説明

表1、表2に本タスク例の機能割り付けを示します。本タスク例は表1、表2に示すようにSH7050に内蔵されているATU及びPFCの機能を割り付け、PWMを出力します。

表1　ATU機能割り付け

ATUの内蔵機能		機能
端子	TIOA4	PWMを出力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TIOR4A	ATUﾁｬﾈﾙ4のレジスタの機能を選択する。
	TIERDL	ATUﾁｬﾈﾙ4の割り込み要求を設定する
	TSTR	ATUﾁｬﾈﾙ4のカウント開始を設定する。
	GR4A	PWMの周期及びデューティを設定する。
	TCR4	ATUﾁｬﾈﾙ4のカウンタのソースクロックを設定する。

表2　PFC機能割り付け

PFCレジスタ	機能
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

図2に動作原理を示します。図2に示すようにSH7050のハードウェア処理及びソフトウェア処理によりパルスを出力します。

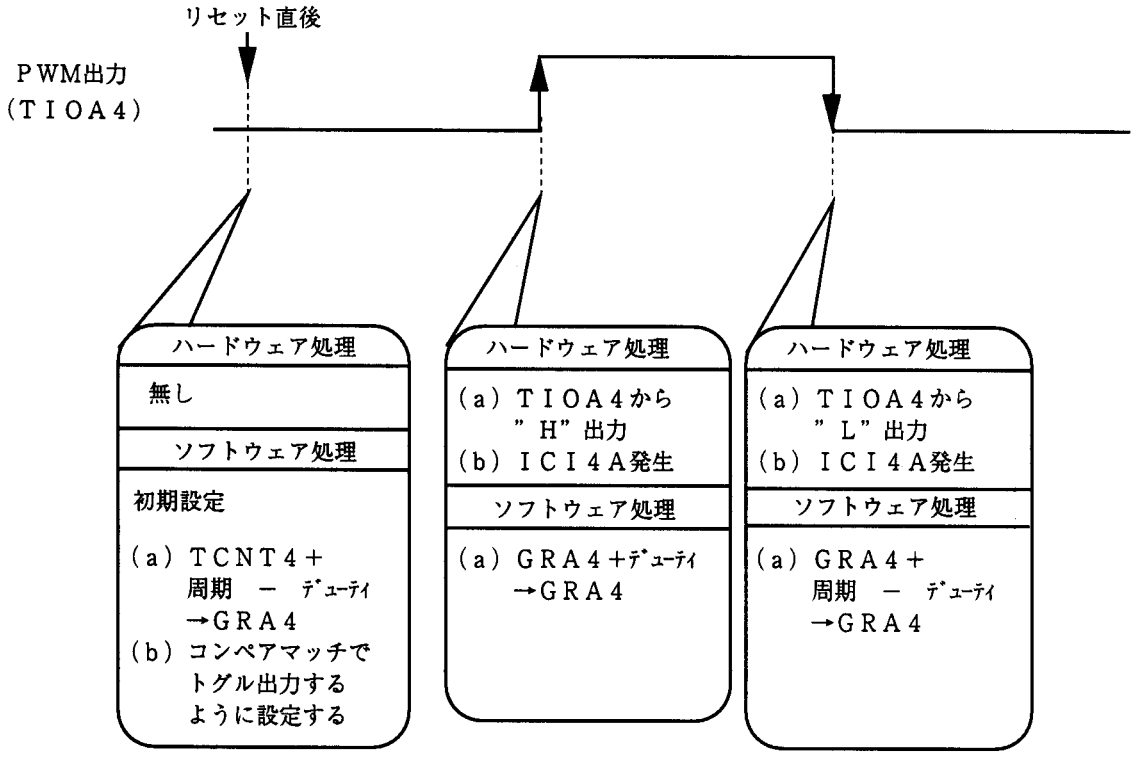


図2　PWM出力動作原理

PWM出力（汎用）	MCU	SH7050	使用機能	ATU

ソフトウェア説明

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	pwmmn	ATUの初期設定を行う。
PWM出力	pwmout	デューティ及び周期をGRA4に設定し、PWMを出力する。

（2）引数の説明

ラベル名	機能	データ長	使用ﾓｼﾞｭｰﾙ名
pwm_shu	パルスの周期に相当するタイマ値を設定する。 パルスの周期は以下の式にて求める。 ﾊﾟﾙｽの周期(ns)=ﾀｲﾏ値×φ周期(20MHz動作時50ns) ×8（ﾌﾟﾘｽｹｰﾗの分周比）	unsigned short	ﾒｲﾝﾙｰﾁﾝ PWM出力
pwm_dut	デューティに相当するタイマ値を以下の式にて求め 16進数に変換した値を設定する。 $\text{タイマ値} = \frac{\text{希望するデューティ（\%）}}{100} \times \text{パルスの周期}$		
pwm_hi	ON／OFFデューティの判定用フラグ	unsigned char	

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PGIOR	TIOA4を出力端子に設定する。	0x0400	ﾒｲﾝﾙｰﾁﾝ
PFC.PGCR1	端子ﾏﾙﾁﾌﾟﾚｸｽをTIOA4端子に設定する。	0x0010	
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／1に設定をする。	0x00	
C35.TCR4	ATUﾁｬﾈﾙ4のﾌﾟﾘｽｹｰﾗをφ／8に設定をする。	0x03	
C35.TIOR4A	ATUﾁｬﾈﾙ4のICR0Aをｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀとして使用し トグル出力するように設定する。	0x03	
C35.TIERDL	ATUﾁｬﾈﾙ4のIME4Aによる割り込み要求を許可する	0x08	
C1.TSTR	ATUﾁｬﾈﾙ4のカウント開始を設定する。	0x0010	
INT.IPRE	ATU41の割り込み優先ﾚﾍﾞﾙを15に設定する。	0x000f	
C35.GR4A	PWMの周期及びデューティを設定する。		PWM出力

（4）使用RAM

本タスクでは引数以外のRAMは使用していません。

PWM出力（汎用）	MCU	SH7050	使用機能	ATU

プログラムリスト

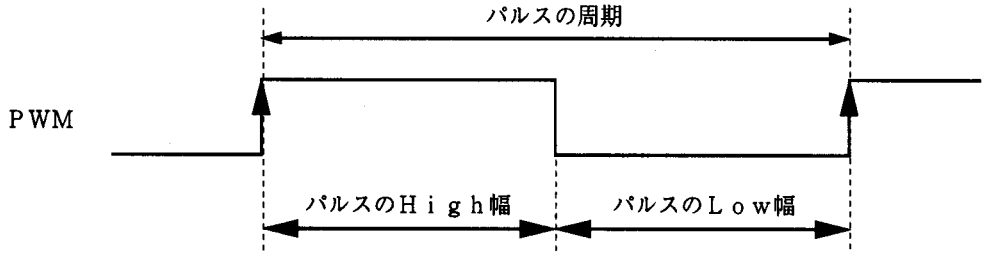
```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwmmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define pwm_shu   (*(unsigned short *)0xFFFFE800)  /* ＰＷＭ周期           */
#define pwm_dut   (*(unsigned short *)0xFFFFE802)  /* ＰＷＭデューティ     */
#define pwm_hi    (*(unsigned char  *)0xFFFFE884)  /* ＰＷＭ設定判定フラグ */

/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwmmn( void )
{
    PFC.PGIOR = 0x0400;      /* ＰＧ１０を出力端子に設定する               */
    PFC.PGCR1 = 0x0010;      /* 端子ﾏﾙﾁﾌﾟﾚｸｽをTIOA4の使用に設定する        */
    C1.PSCR1 = 0x00;         /* ﾌﾟﾘｽｹｰﾗ１段目  φ/1                         */
    C35.TCR4 = 0x03;         /* ﾁｬﾈﾙ4  内部ｸﾛｯｸ φ'/8 でｶｳﾝﾄ               */
    C35.TIOR4A = 0x03;       /* GR4Aはｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀ,ﾄｸﾞﾙ出力          */
    C35.TIERDL = 0x08;       /* IME4Aによる割り込み要求を許可              */
    C1.TSTR = 0x0010;        /* ﾁｬﾈﾙ4  ｶｳﾝﾄｽﾀｰﾄ                            */
    INT.IPRE = 0x000f;       /* ATU41の割り込み優先ﾚﾍﾞﾙを１５に設定        */
    pwm_shu = 0x6400;        /* 周期  ＝  Ｈ'６４００                      */
    pwm_dut = 0x3200;        /* デューティ  ＝  Ｈ'３２００                */
    pwm_hi = 0x00;           /* デューティ判定フラグクリア                 */
    C35.GR4A  = C35.TCNT4 + S
        pwm_shu - pwm_dut;   /* ＯＦＦデューティ設定                       */
    set_imask(0x0);          /* 割り込み許可                               */
    while(1);                /* 無限ループ（割り込み待ち）                 */
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* ＰＷＭ出力ルーチン                                                      */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( pwmout )
void pwmout( void )
{
    C35.TSRDL &= 0xf7;                          /* IMF4Aのｸﾘｱ              */
    if( pwm_hi == 0 )                           /* ＯＮデューティ設定か？  */
    {
        C35.GR4A  = C35.GR4A + pwm_dut;         /* ＯＮデューティ設定      */
    }
    else
    {
        C35.GR4A  = C35.GR4A + pwm_shu-pwm_dut; /* ＯＦＦデューティ設定    */
    }
    pwm_hi = ~pwm_hi;
}
```

2.13　PWM出力（専用）	MCU	SH7050	使用機能	ATU

仕様

（1）図1に示すように、デューティ及び周期を変化できるパルスを出力します。
（2）20MHzで動作時、出力するパルスの周期は0．1μsから3．2msの間で任意に設定できます。



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

図1　PWM出力タイミング

使用機能説明

表1、表2に本タスク例の機能割り付けを示します。本タスク例は表1、表2に示すようにSH7050に内蔵されているATU、PFCの機能を割り付け、PWMを出力します。

表1　ATU機能割り付け

ATUの内蔵機能		機能
端子	TO6	PWMを出力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TSTR	ATUﾁｬﾈﾙ6のカウント開始を設定する。
	CYLR6	PWMの周期を設定する。
	BFR6	PWMのデューティを設定する。
	TCR6	ATUﾁｬﾈﾙ6のカウンタのソースクロックを設定する。

表2　PFC機能割り付け

PFCレジスタ	機能
PBIOR	端子の入出力方向を設定する。
PBCR1	端子の機能を選択する。

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスを出力します。

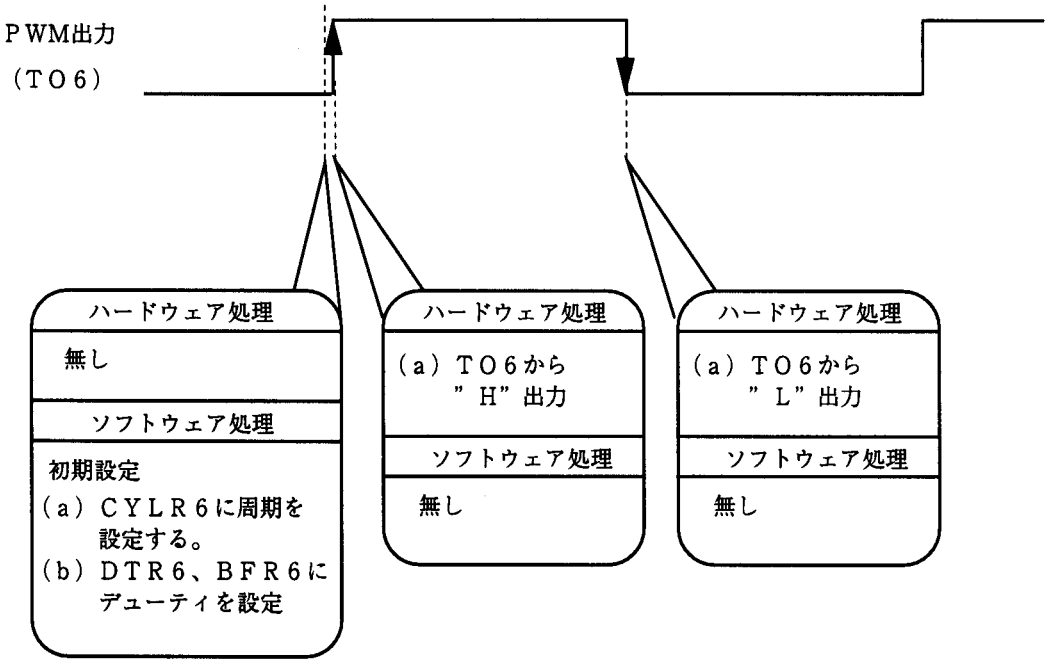


図２　PWM出力動作原理

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	pwmbfmn	デューティ及び周期をBFR6及びCYLR6に設定し、PWMを出力する。

（2）引数の説明

本タスク例では引数は使用していません。

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PBIOR	PB0を出力端子に設定する。	0x0001	ﾒｲﾝﾙｰﾁﾝ
PFC.PBCR	端子ﾏﾙﾁﾌﾟﾚｸｽをTO6端子に設定する。	0x0001	
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／1に設定をする。	0x00	
C69.TCR6	ATUﾁｬﾈﾙ6のﾌﾟﾘｽｹｰﾗをφ／1に設定をする。	0x00	
C1.TSTR	ATUﾁｬﾈﾙ6のカウント開始を設定する。	0x0040	
C69.CYLR6	PWMの周期を設定する。	0x03e8	
C69.DTR6	PWMのデューティを設定する。	0x01f4	
C69.BFR6	PWMのデューティを設定する。	0x01f4	

（4）使用RAM

本タスクでは引数以外のRAMは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwmbfmn( void );
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwmbfmn( void )
{
    PFC.PBIOR = 0x0001;     /* ＰＢ０を出力端子に設定する                  */
    PFC.PBCR = 0x0001;      /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＯ６端子の使用にする        */
    C1.PSCR1 = 0x00;        /* ﾌﾟﾘｽｹｰﾗ１段目　φ/1                         */
    C69.TCR6 = 0x00;        /* ﾁｬﾈﾙ6　内部ｸﾛｯｸ φ' /1 でｶｳﾝﾄ               */
    C1.TSTR = 0x0040;       /* ﾁｬﾈﾙ6　ｶｳﾝﾄｽﾀｰﾄ                             */
    C69.CYLR6 = 0x03E8;     /* ＰＷＭ周期　＝　Ｈ’　０３Ｅ８              */
    C69.DTR6  = 0x01F4;     /* ＰＷＭデューティ　＝　Ｈ’　０１Ｆ４        */
    C69.BFR6  = 0x01F4;     /* ＰＷＭデューティ　＝　Ｈ’　０１Ｆ４        */
    set_imask(0x0);         /* 割り込み許可                                */
    while(1);               /* 無限ループ（割り込み待ち）                  */
}
```

2.14　PWMによる正弦波生成	MCU	SH7050	使用機能	ATU

仕様

（1）図1に示すようにPWMを出力し、ローパスフィルタで正弦波を生成します。
（2）PWMの周期は50μsに固定、デューティは周期割り込み毎にﾃﾞｰﾀﾃｰﾌﾞﾙ（ﾃﾞｰﾀ数360個）のﾃﾞｰﾀに従い変更します。生成する正弦波の周期は18msです。

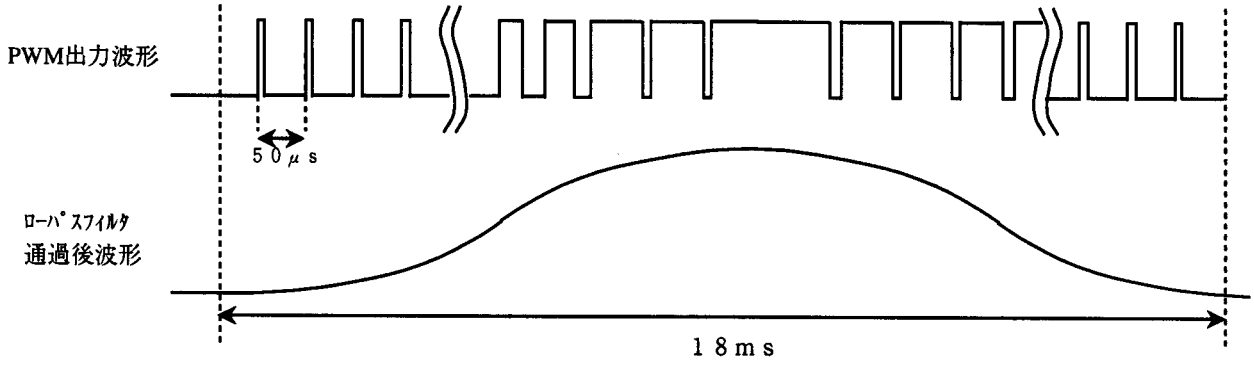


図1　PWMによる正弦波生成

使用機能説明

表1、表2に本タスク例の機能割り付けを示します。本タスク例は表1、表2に示すようにSH7050に内蔵しているATU、PFCに機能を割り付け、PWMを出力します。

表1　ATU機能割り付け

ATUの内蔵機能		機能
端子	TO6	PWMを出力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TSTR	ATUﾁｬﾈﾙ6のカウント開始を設定する。
	CYLR6	PWMの周期を設定する。
	BFR6	PWMのデューティを設定する。

表2　PFC機能割り付け

PFCレジスタ	機能
PBIOR	端子の入出力方向を設定する。
PBCR	端子の機能を選択する。

図2に動作原理を示します。図2に示すようにSH7050のハードウェア処理及びソフトウェア処理によりパルスを出力します。

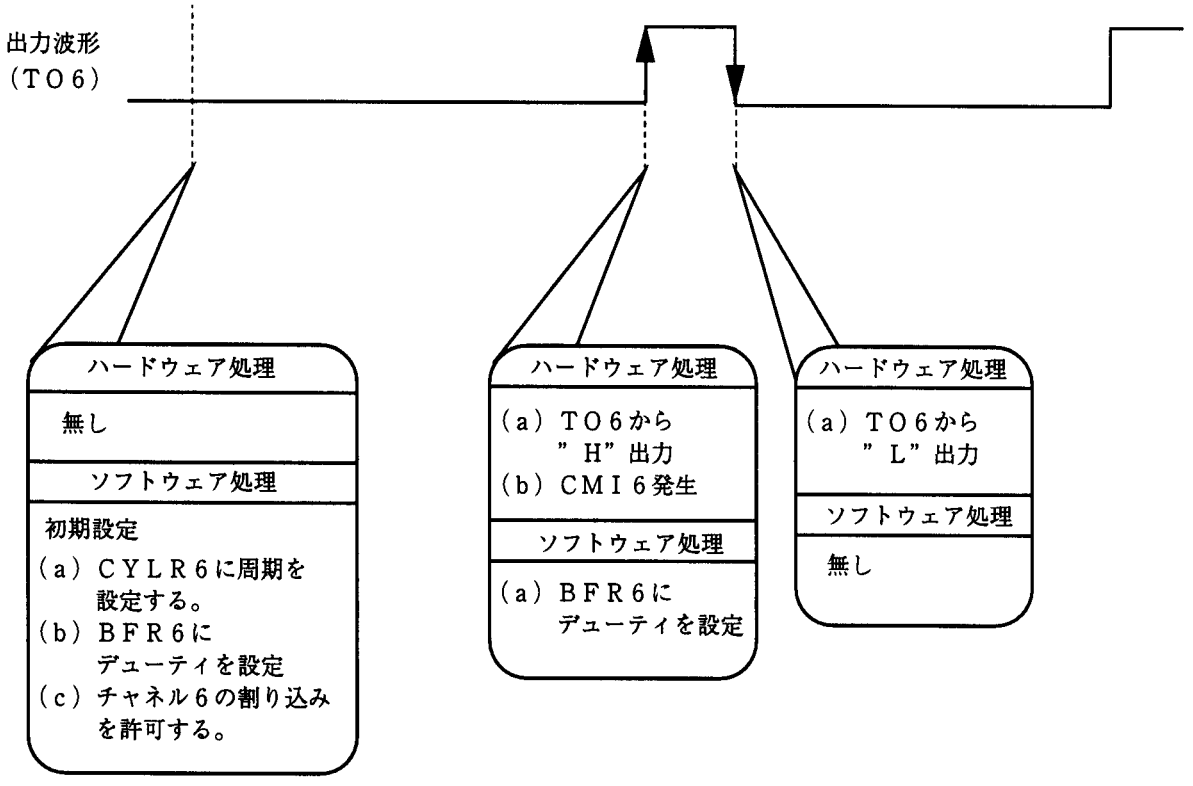


図2　PWM出力動作原理

（1）モジュール説明

モジュール名	ラベル名	機能
メインルーチン	pwmbfmn	ATU6の初期設定及び、PWMの周期をCYLR6に設定する。
ﾃﾞｭｰﾃｨ更新	chg_dut	PWMのデューティを更新する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
sin_dat[360]	正弦波データを格納する。	unsigned short	ﾒｲﾝﾙｰﾁﾝ

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／1に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TSTR	ATUﾁｬﾈﾙ6のカウント開始を設定する。	0x0040	
C69.TCR6	ATUﾁｬﾈﾙ6のﾌﾟﾘｽｹｰﾗをφ／1に設定する。	0x00	
C69.TIERE	ATUﾁｬﾈﾙ6のｺﾝﾍﾟｱﾏｯﾁ割り込みを許可する。	0x40	
C69.CYLR6	PWMの周期を設定する。	0x03e8	
INT.IPRF	ATU6の割り込みレベルを14に設定する。	0x00f0	
PFC.PBIOR	PB0を出力に設定する。	0x0001	
PFC.PBCR	端子ﾏﾙﾁﾌﾟﾚｸｽをATUのPWM出力にする。	0x0001	
C69.BFR6	PWMのデューティを設定する。	sin_dat	ﾃﾞｭｰﾃｨ更新

（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
lp	正弦波の角度を格納する。	unsigned short	ﾒｲﾝﾙｰﾁﾝ

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include <math.h>                               /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ          */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void pwmbfmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
unsigned short sin_dat[720];
unsigned short lp;
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void pwmbfmn( void )
{
    for( lp = 0; lp < 360; lp++ )       /* 正弦波データの算出                  */
    {
        sin_dat[ lp ] =
            (unsigned short)( 500*( 1 + sin((float)(lp-90)/57.29577951)));
    }
    PFC.PBIOR = 0x0001;                 /* ＰＢ０を出力端子に設定する          */
    PFC.PBCR = 0x0001;                  /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＯ６の使用にする    */
    C1.PSCR1 = 0x00;                    /* ﾌﾟﾘｽｹｰﾗ１段目　φ/1                 */
    C69.TIERE = 0x40;                   /* ＣＭＦ６の割り込みを許可する        */
    C69.TCR6 = 0x00;                    /* ﾁｬﾈﾙ6　内部ｸﾛｯｸ φ' /1 でｶｳﾝﾄ       */
    C69.CYLR6 = 0x03e8;                 /* ＰＷＭ周期　→　ＣＹＬＲ６          */
    C69.BFR6  = 0x0000;                 /* ＰＷＭデューティ　→　ＢＦＲ６      */
    C1.TSTR = 0x0040;                   /* ﾁｬﾈﾙ6　ｶｳﾝﾄｽﾀｰﾄ                     */
    INT.IPRF = 0x00f0;                  /* ATU6割り込み優先ﾚﾍﾞﾙを１５に設定    */
    set_imask(0x0);                     /* 割り込み許可                        */
    while(1);                           /* 無限ループ（割り込み待ち）          */
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* デューティの更新                                                        */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( chg_dut )
void chg_dut( void )
{
    C69.TSRE &= 0xbf;                   /* ﾁｬﾈﾙ6　フラグクリア                  */
    C69.BFR6 = sin_dat[ lp ];           /* デューティの更新                    */
    if( lp >= 360 )
    {
       lp=0;
    }
    else
    {
       lp++;
    }
}
```

2.15　複数パルスの出力	MCU	SH7050	使用機能	APC、ATU

仕様

（1）図1に示すように外部信号の立ち上がりに同期して主パルスを出力します。このパルスをＡＰＣで制御し、複数の端子からパルスを出力します。パルスのＡ期間及びＢ期間は内部クロックカウント値を設定します。

（2）Ａ期間及びＢ期間は以下に示す範囲で可変できます。

５０ｎｓ＜Ａ期間及びＢ期間＜３．３ｍｓ

（3）パルスの出力パターンは１端子のみの単独出力、シーケンシャル出力や図２、図３に示すような同時出力、外部信号とオーバラップした出力ができます。

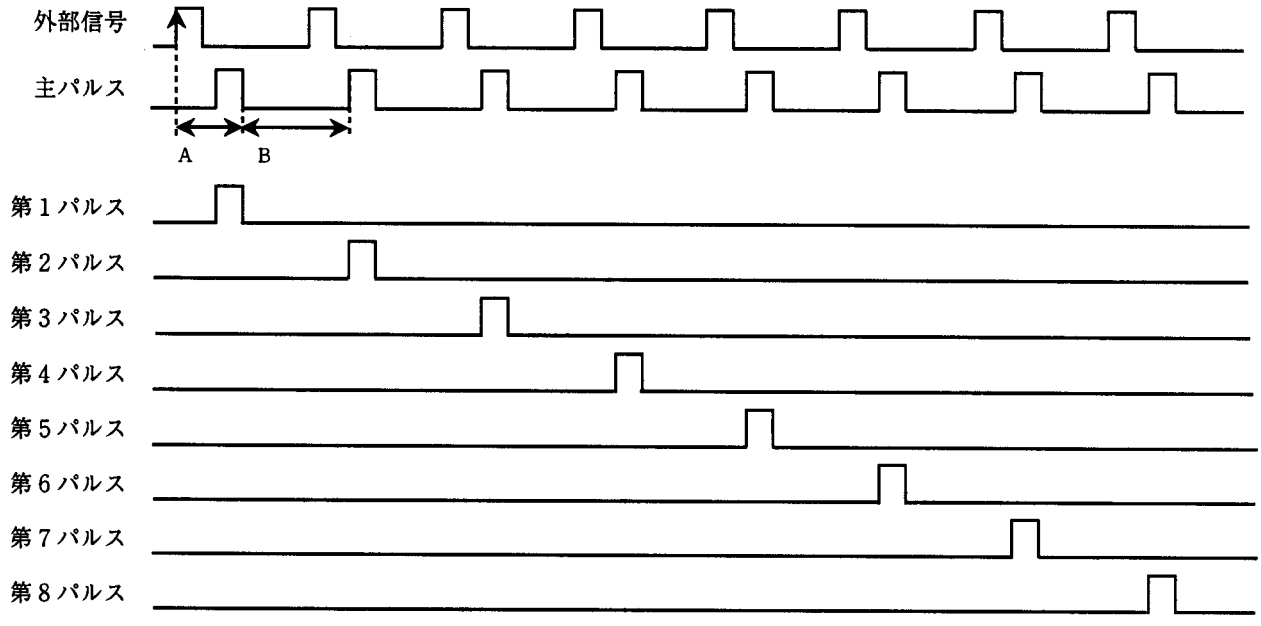


図1　点火信号出力

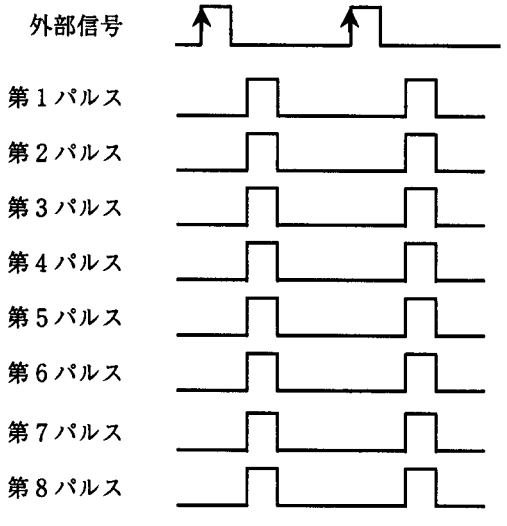


図2　同時出力

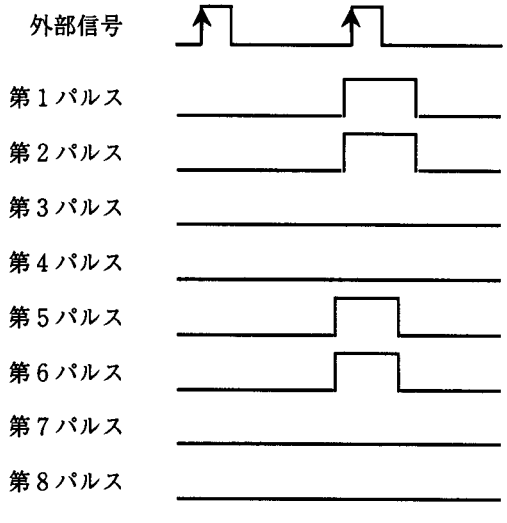


図3　ｵｰﾊﾞｰﾗｯﾌﾟ出力

使用機能説明

表1、表2、表3に本タスク例の機能割り付けを示します。本タスク例は表1、表2、表3に示すようにSH7050に内蔵しているAPC、ATU及びPFCの機能を割り付け、点火信号を出力します。

表1　APC機能割り付け

APC機能		機能
端子	PULS0～7	パルスを出力する。
レジスタ	POPCR	パルス出力端子（PULS0～7）の選択をする。

表2　ATU機能割り付け

ATUの内蔵機能		機能
端子	TIA0	外部信号を入力する。
	TCLKA	外部クロックを入力する。
レジスタ	PSCR1	ATUのﾌﾟﾘｽｹｰﾗの設定をする。
	TCR1	TCNT1のクロックソースを設定する。
	TCR2	TCNT2のクロックソースを設定する。
	OSBR	外部信号の立ち上がり時のカウンタ値を検出する。
	GR2A	B期間を設定する
	GR2B	A期間を設定する。

表3　PFC機能割り付け

PFCレジスタ	機能
PGIOR	端子の入出力方向を設定する。
PGCR1	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりパルスを出力します。

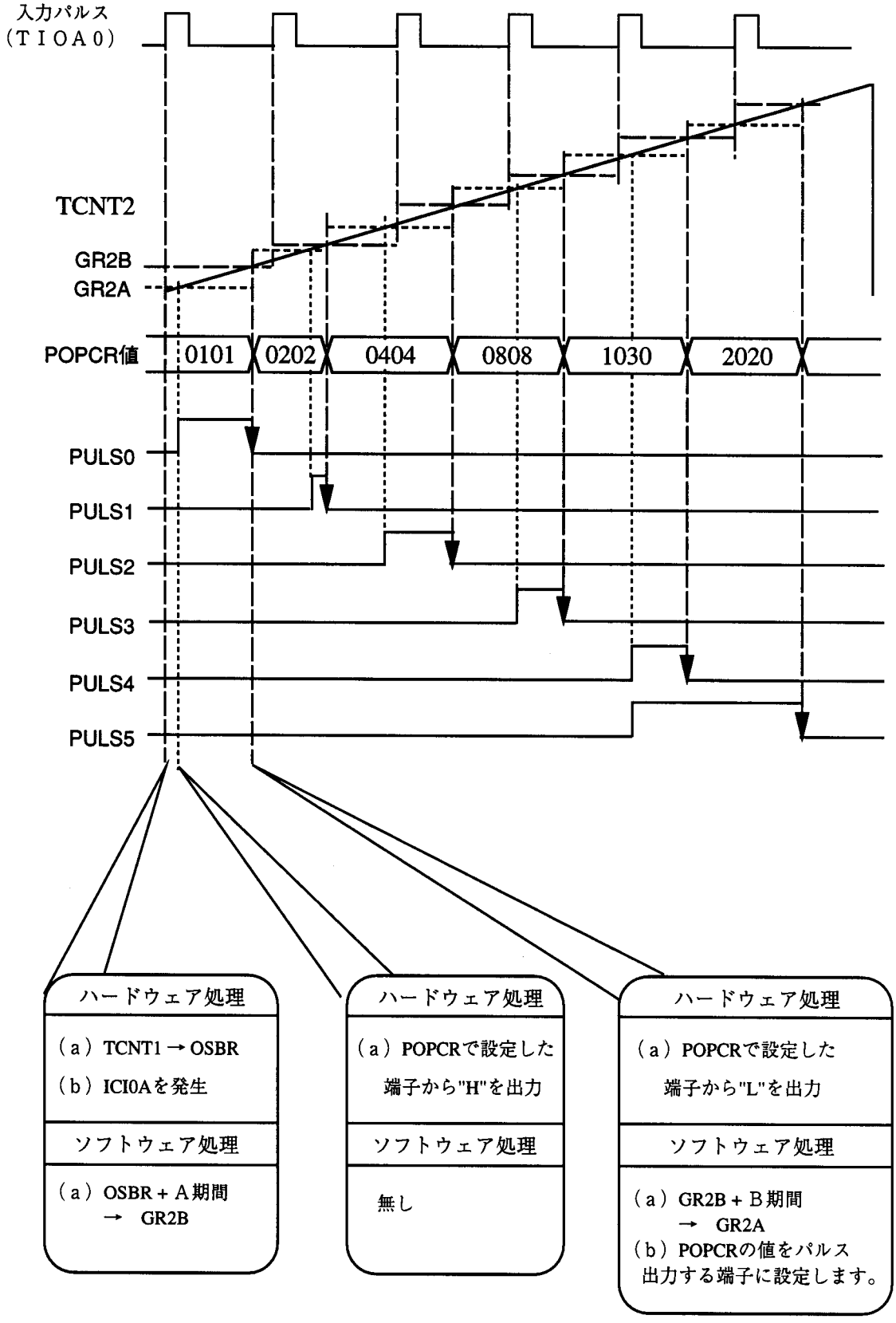


図２　ＡＰＣを使用したパルス出力原理

複数パルスの出力	MCU	SH7050	使用機能	APC、ATU

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
ﾒｲﾝﾙｰﾁﾝ	ignmn	ＡＰＣ、ＡＴＵの初期設定を行なう。
エッジ検出	get_egze	ＩＣＩ０Ａで起動し、Ａ期間をＧＲ２Ｂに設定する。
Ｂ期間設定	ign	ＩＭＩ２Ａで起動し、Ｂ期間をＧＲ２Ａに設定する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
sig.up1～８	Ａ期間を格納する。	unsined short	ﾒｲﾝﾙｰﾁﾝ エッジ検出
out.pls1～８	パルスを出力する端子を示すデータを格納する。	unsined short	ﾒｲﾝﾙｰﾁﾝ Ｂ期間設定
pdwn	Ｂ期間を格納する		
adv_cnt	パルスを出力する端子番号を設定する。		

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
C0.POPCR	パルス出力端子（PULS0～7）の選択を行います	out.pls1～８	ﾒｲﾝﾙｰﾁﾝ Ｂ期間設定
C0.TIERA	ATUﾁｬﾈﾙ0のICF0Aによる割り込み要求を許可する	0x01	ﾒｲﾝﾙｰﾁﾝ
C0.TIOR0A	入力ﾊﾟﾙｽの立ち上がりｴｯｼﾞでTCNT0をICR0Aへｷｬﾌﾟﾁｬするように設定する。	0x01	ﾒｲﾝﾙｰﾁﾝ
C1.TIERC	ATUﾁｬﾈﾙ2のIME2Bによる割り込み要求を許可する	0x01	ﾒｲﾝﾙｰﾁﾝ
C1.TIOR2A	ATUﾁｬﾈﾙ4のGR1Aをｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀに設定する。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.PSCR1	ATUのﾌﾟﾘｽｹｰﾗ1段目をφ／１に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TCR1	ATUﾁｬﾈﾙ1のﾌﾟﾘｽｹｰﾗをφ／１に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TCR2	ATUﾁｬﾈﾙ2のﾌﾟﾘｽｹｰﾗをφ／１に設定をする。	0x00	ﾒｲﾝﾙｰﾁﾝ
C1.TSTR	ATUﾁｬﾈﾙ0、ﾁｬﾈﾙ1、ﾁｬﾈﾙ2のカウント開始を設定する。	0x0003	ﾒｲﾝﾙｰﾁﾝ
C1.OSBR	ATUﾁｬﾈﾙ0のｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬでTCNT1を設定する。	TCNT１	エッジ検出
C1.GR2A	Ｂ期間を設定する。	sig.up1～８	Ｂ期間設定
C1.GR2B	Ａ期間を設定する。	pdwn	エッジ検出 Ｂ期間設定
PFC.PECR	端子ﾏﾙﾁﾌﾟﾚｸｽをＴＩＡ０の使用にする。	0xfff0	ﾒｲﾝﾙｰﾁﾝ
PFC.PFIOR	ＰＦ４～１１を出力に設定する。	0x0ff0	ﾒｲﾝﾙｰﾁﾝ
PFC.PFCR1	端子ﾏﾙﾁﾌﾟﾚｸｽをＡＰＣパルス出力端子の使用にする。	0xffae	ﾒｲﾝﾙｰﾁﾝ
PFC.PFCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをＡＰＣパルス出力端子の使用にする。	0xaa00	ﾒｲﾝﾙｰﾁﾝ
INT.IPRC	ＡＴＵ０２の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x0f00	ﾒｲﾝﾙｰﾁﾝ
INT.IPRE	ＡＴＵ２の割り込み優先ﾚﾍﾞﾙを１４に設定する。	0xe000	ﾒｲﾝﾙｰﾁﾝ

（4）使用ＲＡＭ

本タスクでは引き数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-----------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                  */
/*-----------------------------------------------------------------------*/
void ignmn( void );
/*-----------------------------------------------------------------------*/
/* 変数定義                                                              */
/*-----------------------------------------------------------------------*/
#define pdwn      (*(unsigned short *)0xFFFFE800)  /*  B期間データ格納      */
#define adv_cnt   (*(unsigned short *)0xFFFFE802)  /*  出力端子ｶｳﾝﾀ         */

volatile  struct pl
{
    unsigned short  pls1;               /* 出力端子データ1                   */
    unsigned short  pls2;               /* 出力端子データ2                   */
    unsigned short  pls3;               /* 出力端子データ3                   */
    unsigned short  pls4;               /* 出力端子データ4                   */
    unsigned short  pls5;               /* 出力端子データ5                   */
    unsigned short  pls6;               /* 出力端子データ6                   */
    unsigned short  pls7;               /* 出力端子データ7                   */
    unsigned short  pls8;               /* 出力端子データ8                   */
};
#define out (*(struct pl *)0xFFFFE804)
volatile  struct add
{
    unsigned short  up1;                /* A期間データ1                      */
    unsigned short  up2;                /* A期間データ2                      */
    unsigned short  up3;                /* A期間データ3                      */
    unsigned short  up4;                /* A期間データ4                      */
    unsigned short  up5;                /* A期間データ5                      */
    unsigned short  up6;                /* A期間データ6                      */
    unsigned short  up7;                /* A期間データ7                      */
    unsigned short  up8;                /* A期間データ8                      */
};
#define sig (*(struct add *)0xFFFFE814)
```

```
/*------------------------------------------------------------------------*/
/* メインルーチン                                                         */
/*------------------------------------------------------------------------*/
void ignmn( void )
{
    PFC.PECR  = 0x0100;          /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＴＩＡ０の使用にする      */
    PFC.PFIOR = 0xfff0;          /* ＰＦ４～１１を出力端子に設定する        */
    PFC.PFCR1 = 0xffae;          /* 端子ﾏﾙﾁﾌﾟﾚｸｽをAPCﾊﾟﾙｽ出力の使用にする  */
    PFC.PFCR2 = 0xaa00;          /* 端子ﾏﾙﾁﾌﾟﾚｸｽをAPCﾊﾟﾙｽ出力の使用にする  */
    C1.PSCR1 = 0x00;             /* ﾌﾟﾘｽｹｰﾗ１段目  φ/1                     */
    C1.TCR2 = 0x26;              /* 外部ｸﾛｯｸ A端子入力  両ｴｯｼﾞでｶｳﾝﾄ       */
    C1.TCR1 = 0x26;              /* 外部ｸﾛｯｸ A端子入力  両ｴｯｼﾞでｶｳﾝﾄ       */
    C0.TIOR0A = 0x01;            /* 立ち上がりｴｯｼﾞでICR0Dへｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ   */
    C1.TIOR2A = 0x12;            /* GR2A,GR2Bをｱｳﾄﾌﾟｯﾄｺﾝﾍﾟｱﾚｼﾞｽﾀとして使用 */
    C0.TIERA = 0x01;             /* IC10Aによる割り込み要求を許可           */
    C1.TSTR = 0x0007;            /* ﾁｬﾈﾙ0,ﾁｬﾈﾙ1,ﾁｬﾈﾙ2 ｶｳﾝﾄｽﾀｰﾄ            */
    INT.IPRC = 0x000f;           /* ATU02の割り込み優先ﾚﾍﾞﾙを１５に設定     */
    INT.IPRE = 0xe000;           /* ATU2 の割り込み優先ﾚﾍﾞﾙを１４に設定     */
    C1.GR2A = C1.TCNT2 + 4;      /* Ｈｉｇｈ出力開始                        */
    adv_cnt = 0;
    out.pls1 = 0x0101;           /* 出力端子データ１にデータを設定          */
    out.pls2 = 0x0202;           /* 出力端子データ２にデータを設定          */
    out.pls3 = 0x0404;           /* 出力端子データ３にデータを設定          */
    out.pls4 = 0x0808;           /* 出力端子データ４にデータを設定          */
    out.pls5 = 0x1030;           /* 出力端子データ５にデータを設定          */
    out.pls6 = 0x2020;           /* 出力端子データ６にデータを設定          */
    out.pls7 = 0x1030;           /* 出力端子データ７にデータを設定          */
    out.pls8 = 0x2020;           /* 出力端子データ８にデータを設定          */
    POPCR = out.pls1;            /* 出力端子設定                            */
    sig.up1 = 0x0040;            /* Ａ時期データ１にデータを設定            */
    sig.up2 = 0x0010;            /* Ａ時期データ２にデータを設定            */
    sig.up3 = 0x0020;            /* Ａ時期データ３にデータを設定            */
    sig.up4 = 0x0020;            /* Ａ時期データ４にデータを設定            */
    sig.up5 = 0x0030;            /* Ａ時期データ５にデータを設定            */
    sig.up6 = 0x0030;            /* Ａ時期データ６にデータを設定            */
    sig.up7 = 0x0030;            /* Ａ時期データ７にデータを設定            */
    sig.up8 = 0x0030;            /* Ａ時期データ８にデータを設定            */
    pdwn  = 0x00c0;              /* Ｂ期間データを設定する                  */
    set_imask(0x0);              /* 割り込み許可                            */
    while(1);                    /* 無限ループ（割り込み待ち）              */
}
/*------------------------------------------------------------------------*/
/* Ａ期間設定ルーチン                                                     */
/*------------------------------------------------------------------------*/
#pragma interrupt( get_egze )
void get_egze( void )
{
    C0.TSRAL  &= 0xfe;                  /* ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬA ﾌﾗｸﾞｸﾘｱ          */
    C1.GR2B = C1.OSBR +
        *(unsigned short *)(&sig.up1 + adv_cnt); /* Ａ期間設定             */
    C1.TIERC = 0x02;                    /* ICF0Cによる割り込み要求を許可する */
}
/*------------------------------------------------------------------------*/
/* Ｂ期間設定ルーチン                                                     */
/*------------------------------------------------------------------------*/
#pragma interrupt( ign )
void ign( void )
{
    C1.TSRC  &= 0xfd;                   /* IMF2Bのｸﾘｱ                     */
    C1.GR2A = C1.GR2B +  pdwn;          /* Ｂ期間設定                      */
    adv_cnt++;                          /* 出力端子カウンタのインクリメント */
    if( adv_cnt >= 8 )
    {
        adv_cnt = 0;                    /* 出力端子カウンタのクリア        */
    }
    POPCR = *(unsigned short *)(&out.pls1 + adv_cnt);  /* 出力端子設定     */
    C1.TIERC = 0x00;                    /* ICF0Cによる割り込み要求を禁止   */
}
```

仕様

（1）ウォッチドッグタイマを用いて図１のようなパルスを出力します。
（2）２０ＭＨｚ動作時、パルスの周期は２０．４８ｍｓ、デューティは５０％です。

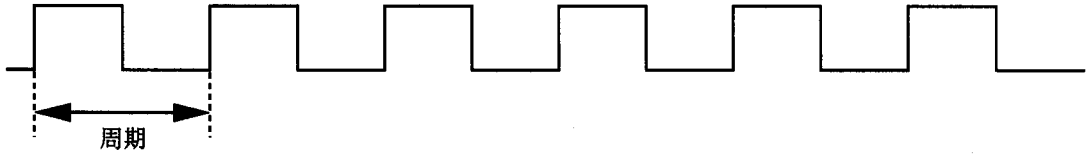


図１　ウォッチドッグタイマによるパルス出力

使用機能説明

表１、表２に本タスク例の機能割付を示します。表１、表２に示すようにＳＨ７０５０に内蔵されているＷＤＴ、ＰＦＣの機能を割り付け、パルスの出力を行ないます。

表１　ＷＤＴ機能割付

WDTレジスタ	機能
ＴＣＮＴ	オーバフロー周期を設定します。
ＴＣＳＲ	ｲﾝﾀｰﾊﾞﾙﾀｲﾏﾓｰﾄﾞに設定します。

表２　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
ＰＡＩＯＲ	端子の入出力方向を設定する。
ＰＡＣＲ	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェアの処理によりパルスを出力します。

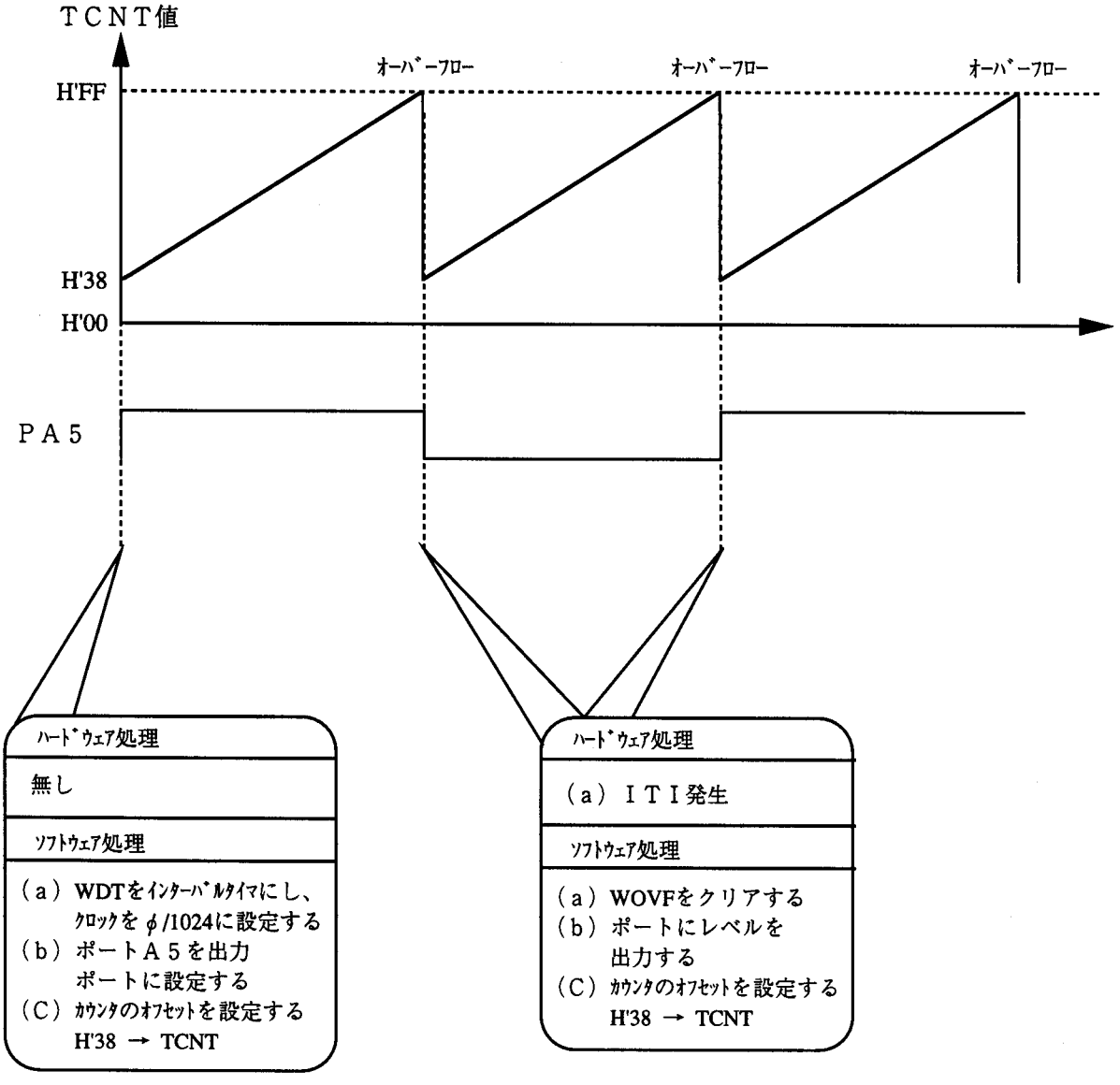


図２　パルス出力原理

内蔵ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏによるパルス出力	MCU	ＳＨ７０５０	使用機能	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏ

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
ﾒｲﾝﾙｰﾁﾝ	ｗｄｔｍｎ	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏの初期設定を行なう。
パルス出力	ｉｎｔｗｄｔ	ＩＴＩで起動し１０ｍｓ周期でポートにＨＩとＬＯＷを交互に出力する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
ｏｕｔ＿ｄａｔ	ポート出力データ	unsigned short	パルス出力
ｄｍｙ＿ｒｄ	ＴＣＳＲのダミーリード用	unsigned char	

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
WDT_TCNT	ｵｰﾊﾞｰﾌﾛｰの周期を１０．２４ｍｓに設定する。	0x5a38	ﾒｲﾝﾙｰﾁﾝ
WDT_TCSR	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏをｲﾝﾀｰﾊﾞﾙﾀｲﾏﾓｰﾄﾞに設定する。	0xa53d	ﾒｲﾝﾙｰﾁﾝ
PFC.PAIOR	ＰＡ５を出力に設定する。	0x0040	ﾒｲﾝﾙｰﾁﾝ
PFC.PACR	端子ﾏﾙﾁﾌﾟﾚｸｽをﾎﾟｰﾄの使用にする。	0x0000	ﾒｲﾝﾙｰﾁﾝ
INT.IPRH	ＩＴＩの割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x000f	ﾒｲﾝﾙｰﾁﾝ
PFC.PADR	レベル出力を行なう。		パルス出力

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void wdtmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define out_dat (*(unsigned short *)0xFFFFE800) /* ポート出力データ格納      */
#define dmy_rd  (*(unsigned char  *)0xFFFFE802) /* TCSRのダミーリード用      */
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*                    メインルーチン                                       */
/*                                                                         */
/*-------------------------------------------------------------------------*/
void wdtmn( void )
{
    WDT_TCSRW = 0xa53d;                 /* ｲﾝﾀｰﾊﾞﾙ割り込み発生，φ／４０９６  */
    WDT_TCNTW = 0x5a38;                 /* ｵﾌｾｯﾄの設定（周期２０．４８ｍｓ）  */
    PFC.PAIOR = 0x0040;                 /* ＰＡ５を出力端子に設定する。        */
    PFC.PACR = 0x0000;                  /* 端子ﾏﾙﾁﾌﾟﾚｸｽをポート使用にする。    */
    INT.IPRH = 0x000f;                  /* WDT割り込み優先ﾚﾍﾞﾙを15に設定する。*/
    out_dat = 0x0001;                   /* ポート出力データの設定する。        */
    set_imask(0x0);                     /* 割り込み許可                        */
    while(1);                           /* 無限ループ（割り込み待ち）          */
}

/*-------------------------------------------------------------------------*/
/*                                                                         */
/* インターバル割り込みルーチン                                            */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( intwdt )
void intwdt( void )
{
    dmy_rd = WDT_TCSRR;                 /* ＴＣＳＲのダミーリード              */
    WDT_TCSRW = 0xa53d;                 /* オーバーフローフラグクリア          */
    PFC.PADR = out_dat;                 /* データ出力                          */
    out_dat = ~out_dat;                 /* 出力データの反転                    */
    WDT_TCNTW = 0x5a38;                 /* ｵﾌｾｯﾄの設定（周期２０．４８ｍｓ）  */
}
```

仕様

（１）図１に示すようにウォッチドッグタイマがオーバーフロした時に内部リセット信号を発生し、
　　ＳＨ７０５０をリセットします。
（２）オーバフロ周期は１３ｍｓです。

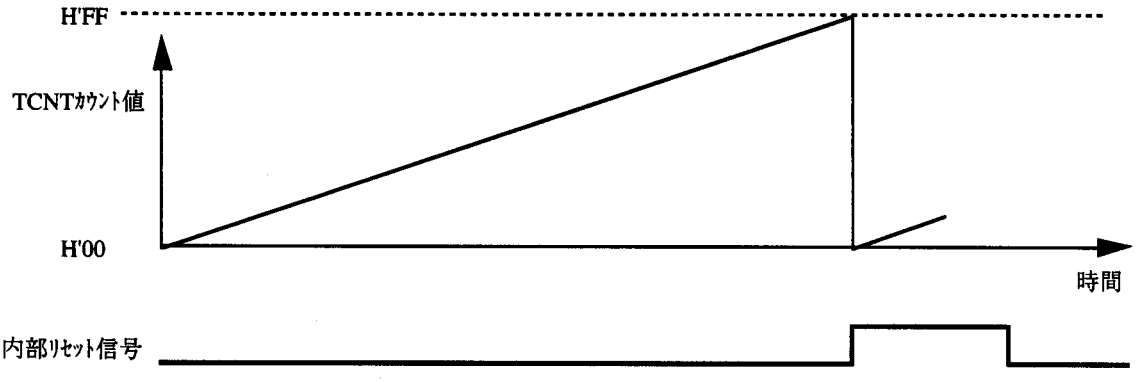


図１　ウォッチドッグタイマによる内部リセット

使用機能説明

表１、表２に本タスク例の機能割付を示します。表１、表２に示すようにＳＨ７０５０に内蔵している
ＷＤＴ、ＡＴＵの機能を割付、パルスの出力を行ないます。

表１　ＷＤＴ機能割付

ﾚｼﾞｽﾀ名	機　能
ＴＣＮＴ	ｵｰﾊﾞｰﾌﾛｰにより、ＩＴＩが発生する。
ＴＣＳＲ	ｲﾝﾀｰﾊﾞﾙﾀｲﾏﾓｰﾄﾞを設定する。

表２　ＡＴＵ機能割付

ﾚｼﾞｽﾀ名	機　能
ＩＴＶＲＲ	インターバルタイマの周期を設定する。

動作説明

図２に動作原理を示します。図２に示すようにＳＨ７０５０のハードウェア処理及びソフトウェア処理によりシステムの監視を行います。

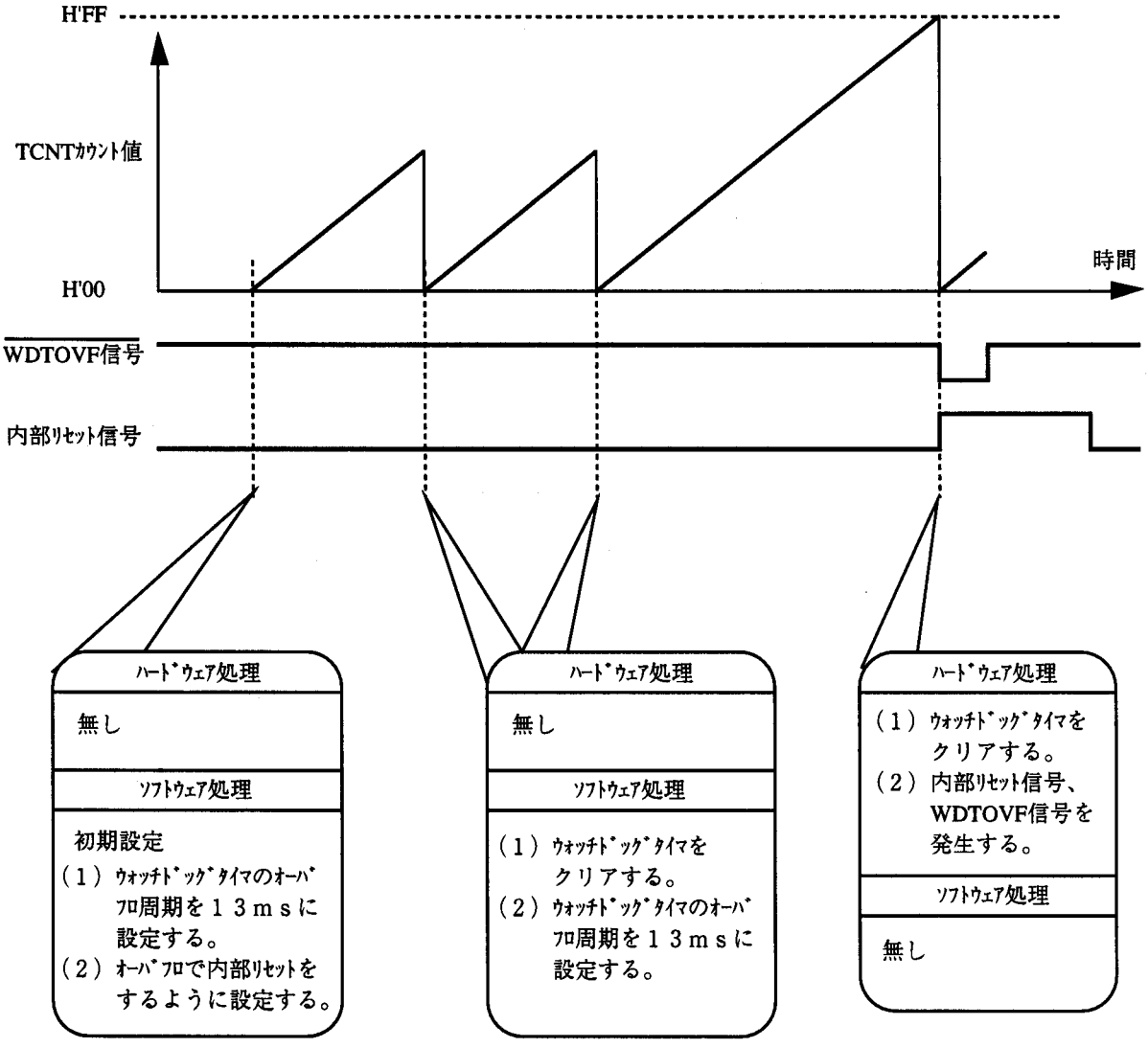


図２　ウォッチドッグタイマ動作原理

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
ﾒｲﾝﾙｰﾁﾝ	ｗｄｍｎ	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏの初期設定を行なう。
スイッチ検出	ｉｎｔ５ｍｓ	５ｍｓ毎にＳＷのＯＮ－ＯＦＦ検出及び、ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏのｸﾘｱを行なう。

（2）引数の説明

本タスクでは引数は使用していません。

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
WDT_TCSR	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏﾓｰﾄﾞを設定します。	0xa55d	ﾒｲﾝﾙｰﾁﾝ
WDT_RSTCSRW	ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏのｵｰﾊﾞｰﾌﾛで内部リセットするように設定する。	0x5a5f	
C1.PSCR1	ﾁｬﾈﾙ０のプリスケーラをφ／６にする。	0x06	
C0.ITVRR	TCNT0 13ﾋﾞｯﾄ目で割り込み発生(820us)するように設定する。	0x08	
C1.TSTR	TCNT0のｶｳﾝﾄ開始を設定する。	0x0001	
INT.IPRC	ATU01の割り込み優先ﾚﾍﾞﾙを１５に設定する。	0x00f0	

（4）使用ＲＡＭ

本タスクではＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                              */
/*-------------------------------------------------------------------*/
void wdmn( void );
/*-------------------------------------------------------------------*/
/* 変数定義                                                          */
/*-------------------------------------------------------------------*/
#define cnt     (*(unsigned char  *)0xFFFFE800)  /* WDTクリア回数格納        */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*                    メインルーチン                                 */
/*                                                                   */
/*-------------------------------------------------------------------*/
void wdmn( void )
{
    CO.ITVRR = 0x08;                    /* TCNT0 13ﾋﾞｯﾄ目で割り込み発生(820us)*/
    C1.PSCR1 = 0x06;                    /* ﾌﾟﾘｽｹｰﾗ１段目   φ/6               */
    C1.TSTR = 0x0001;                   /* ﾁｬﾈﾙ0  ｶｳﾝﾄｽﾀｰﾄ                    */
    WDT_TCSRW = 0xa57d;                 /* WDTOVF信号出力,ｵｰﾊﾞｰﾌﾛ周期13.1ms  */
    WDT_RSTCSRW = 0x5a5f;               /* 内部リセット                      */
    INT.IPRC = 0x00f0;                  /* ATU01割り込み優先ﾚﾍﾞﾙを１５に設定 */
    set_imask(0x0);                     /* 割り込み許可                      */
    while(1);                           /* 無限ループ（割り込み待ち）        */
}

/*-------------------------------------------------------------------*/
/*                                                                   */
/* スイッチ検出ルーチン                                              */
/*                                                                   */
/*-------------------------------------------------------------------*/
#pragma interrupt( int5ms )
void int5ms( void )
{
    CO.TSRAH &= 0xf7;
    WDT_RSTCSRW = 0xa500;               /* オーバーフローフラグクリア        */
    WDT_TCNTW = 0x5a00;                 /* ウォッチドッグタイマクリア        */
    cnt++;                              /* WDTクリア回数更新                 */
    if( cnt > 1 )
    {
        while(1);
    }
}
```

2.18　クロック同期式シリアル送受信	MCU	ＳＨ７０５０	使用機能	ＳＣＩ

仕様

（1）図１に示すように、ＳＨ７０５０のＳＣＩをクロック同期式モードで使用し、ＳＨ１と４バイトのデータを同時に送受信します。
（2）転送プロトコルは転送レート１Ｍｂｐｓ、データ長８ビット、パリティ無しです。

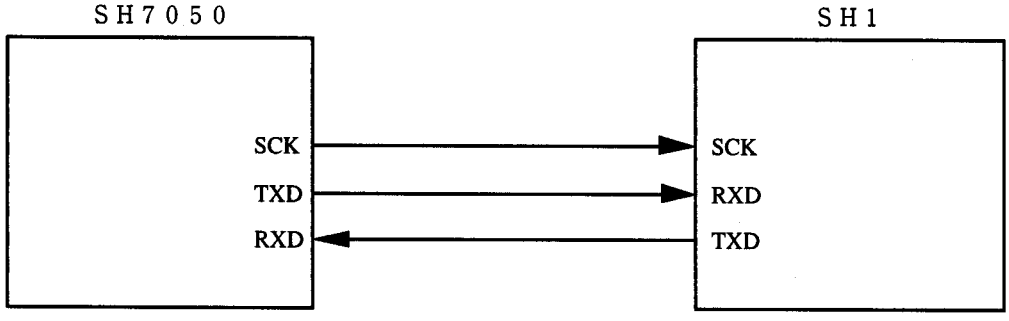


図１　ＳＨ７０５０によるクロック同期式シリアルインタフェースブロック図

使用機能説明

表１、表２に本タスク例の機能割付を示します。表１、表２に示すようにＳＨ７０５０に内蔵しているＳＣＩ、ＰＦＣの機能を割付、ＳＨ１とインタフェースを行ないます。

表１　ＳＣＩ機能割付

ＳＣＩ機能		機　能
端子	RXD	コンソールからﾃﾞｰﾀを受信する。
	TXD	コンソールへﾃﾞｰﾀを送信する。
	SCK	シリアルクロックを入出力する。
レジスタ	SMR	ＳＣＩの送信フォーマットを設定する。
	SCR	ＳＣＩの割り込みの許可／禁止を設定する。
	SSR	割り込みステータスを設定する。
	RDR	コンソールから受信したﾃﾞｰﾀを設定する。
	TDR	コンソールへ送信するﾃﾞｰﾀを設定する。
	BRR	転送ﾚｰﾄを設定する。

表２　ＰＦＣ機能割付

ＰＦＣレジスタ	機　能
ＰＧＩＯＲ	端子の入出力方向を設定する。
ＰＧＣＲ２	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すタイミングでクロック同期式ＳＣＩを制御し、ＳＨ１とインタフェースを行なっています。

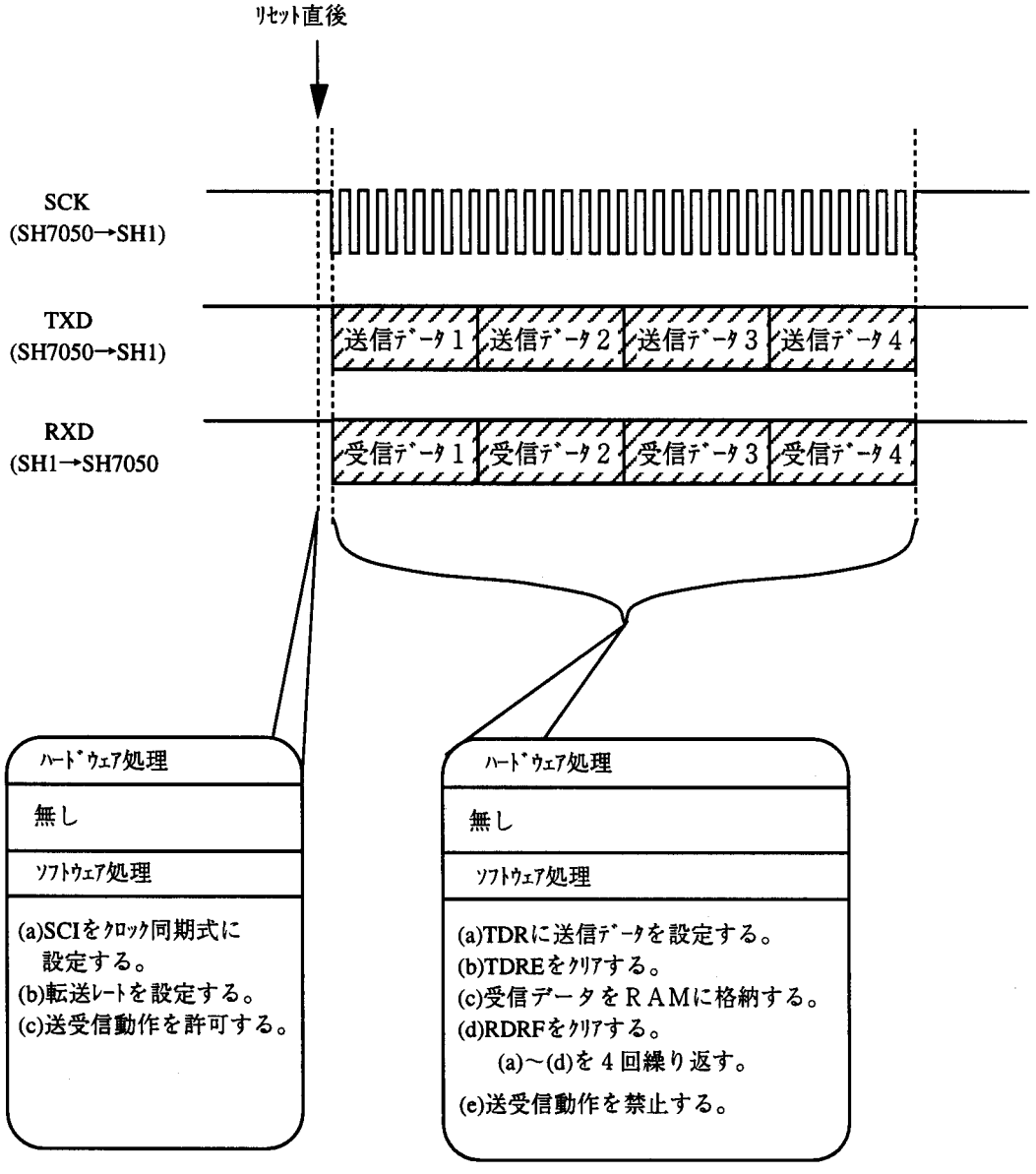


図２　ＳＨ１とのインタフェース動作原理

（1）モジュール説明

ﾓｼﾞｭｰﾙ名	ﾗﾍﾞﾙ名	機　能
ﾒｲﾝﾙｰﾁﾝ	scimn	ＳＣＩの初期設定及びデータの送受信を行なう。

（2）引数の説明

本タスクでは引数は使用していません。

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
SCI.SMR	SCIのﾓｰﾄﾞ(ｸﾛｯｸ同期式)、転送ﾌｫｰﾏｯﾄ及び ﾎﾞｰﾚｲﾄｼﾞｪﾈﾚｰﾀへのｸﾛｯｸ選択(φｸﾛｯｸ入力)を設定する。	0x80	ﾒｲﾝﾙｰﾁﾝ
SCI.SCR	送信・受信動作を許可する。	0x30	
SCI.RDR	SH1から受信したﾃﾞｰﾀを設定する。	ｔｄａｔ	
SCI.TDR	SH1へ送信するﾃﾞｰﾀを設定する。	ｒｄａｔ	
SCI.BRR	転送ﾚｰﾄを１Ｍｂｐｓに設定する。	0x04	
PFC.PGIOR	RXD0を入力、SCK0、TXDを出力に設定する。	0x0006	
PFC.PGCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをRXD、SCK0、TXDの使用にする。	0x0070	

（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
ｌｐ	ＳＣＩ送受信の回数を格納する。	unsined char	ﾒｲﾝﾙｰﾁﾝ
ｔｄａｔ	送信データを格納する。	unsined short	
ｒｄａｔ	受信データを格納する。	unsined short	

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void scimn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define tdat    (*(unsigned long  *)0xffffe800)  /* 送信データ             */
#define rdat    (*(unsigned long  *)0xffffe804)  /* 受信データ             */
/*-------------------------------------------------------------------------*/
/* SCI送受信                                                               */
/*-------------------------------------------------------------------------*/
void scimn( void )
{
    signed int lp;
    PFC.PGIOR = 0x0006;                 /* TXD0出力,RXD0入力                */
    PFC.PGCR2 = 0x0070;                 /* TXD0,RXD0使用                    */
    SCI0.SCR = 0x00;                    /* 送信・受信動作を禁止する         */
    SCI0.SMR = 0x80;                    /* ｸﾛｯｸ同期式､8ﾋﾞｯﾄﾃﾞｰﾀ、ﾊﾟﾘﾃｨなし  */
    SCI0.BRR = 0x04;                    /* ビットレート１Ｍｂｐｓ           */
    SCI0.SCR = 0x00;                    /* 内部クロック                     */
    for( lp = 1; lp < 1; lp++ );        /* ４００ｎｓウエィト               */
    SCI0.SCR = 0x30;                    /* 送信・受信動作を許可する         */
    tdat = 0x5511ffaa;
    for( lp = 0; lp < 4; lp++ )
    {
        while(( SCI0.SSR & 0x80 ) != 0x80 );
        SCI0.TDR = *(unsigned char *)( (long)&tdat + lp ); /* ﾃﾞｰﾀの送信    */
        SCI0.SSR &= 0x7f;               /* 送信フラグをクリアする           */
        while(( SCI0.SSR & 0x40 ) != 0x40 );
        *(unsigned char *)( (long)&rdat + lp ) = SCI0.RDR; /* 受信ﾃﾞｰﾀの格納 */
        SCI0.SSR &= 0xbf;               /* 受信フラグをクリアする           */
    }
    while(( SCI0.SSR & 0x04 ) != 0x04 );
    SCI0.SCR = 0x00;                    /* 送信・受信動作を禁止する         */
}
```

仕様

（1）図1に示すように、1チャネルに入力される電圧をＳＨ７０５０のA／D変換器を使用し、測定します。
（2）測定結果は2バイトのＲＡＭに格納します。
（3）入力する電圧は0～5Ｖの範囲です。

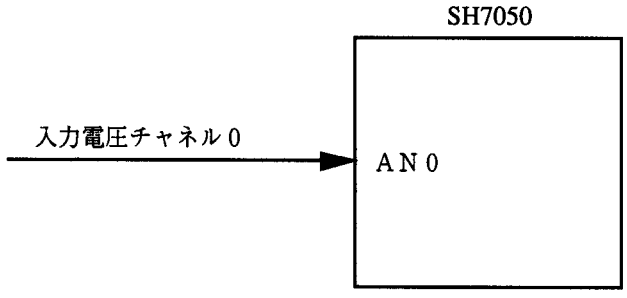


図1　ＳＨ７０５０による電圧の測定ブロック図

使用機能説明

表1に本タスク例の機能割付を示します。表1に示すようにＳＨ７０５０に内蔵されているA／D変換器の機能を割付、A／D変換を行ないます。

表1　A／D変換機能割付

A／D変換機能		機　能
端子	AN0	アナログ電圧を入力する。
レジスタ	ADCSR0	A／D変換のモード（単一モード／スキャンモード）、測定端子の選択を設定する。
	ADCR0	A／D変換器のクロックの選択、測定の開始及び終了を設定する。
	ADDR0	A／D変換の結果を設定する。

動作説明

図2に動作原理を示します。図2に示すように、SH7050のハードウェア処理及びソフトウェア処理により1チャネルのA／D変換を行います。

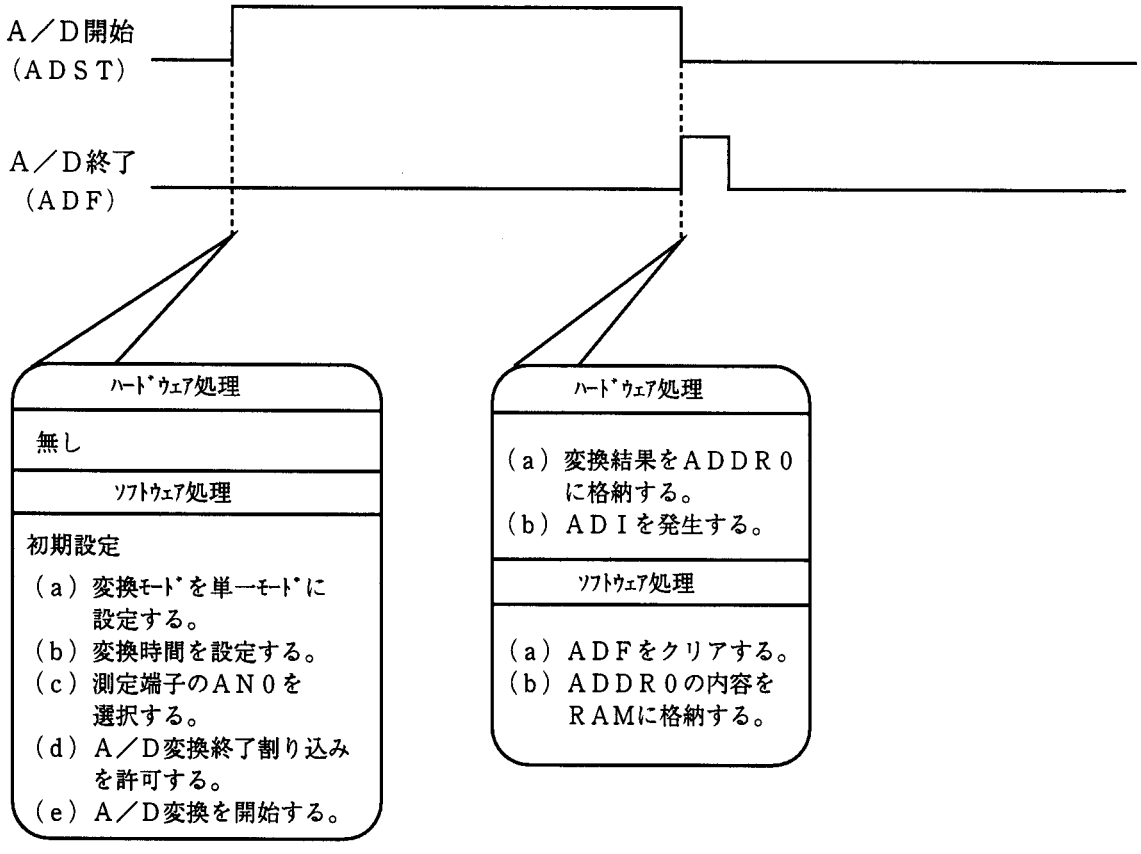


図2　A／D変換動作原理

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
ﾒｲﾝﾙｰﾁﾝ	adonemn	A／D変換器の初期設定を行なう。
A／D変換終了	adend	ＡＤＩにより起動し、結果をＲＡＭに格納する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
ａｄ＿ｄａｔ	A／D変換データを格納する。	unsigned short	A／D変換終了

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
AD.ADCSR0	A／D変換モードを単一モードに設定、 測定端子をＡＮ０に設定する。	0x40	ﾒｲﾝﾙｰﾁﾝ
AD.ADCR0	A／D変換器のクロックを１３４ステートに設定する。	0x7f	
INT.IPRG	A／Ｄ０の割り込み優先ﾚﾍﾞﾙを設定する。	0x00f0	
AD.ADDR0	A／D変換の結果を設定する。		A／D変換終了

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void adonemn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define ad_dat    (*(unsigned short *)0xFFFFE800) /* A/D変換ﾃﾞｰﾀ格納          */

/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void adonemn( void )
{
    AD.ADCSR0 = 0x40;                   /* A/D割込み許可,単一ﾓｰﾄﾞ,測定端子:AN0*/
    AD.ADCR0 = 0x7f;                    /* 測定時間134ｽﾃｰﾄ,A/D変換スタート     */
    INT.IPRG = 0x00f0;                  /* A/D0割り込み優先ﾚﾍﾞﾙを１５に設定    */
    set_imask(0x0);                     /* 割り込み許可                       */
    while(1);                           /* 無限ループ（割り込み待ち）         */
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* Ａ／Ｄ変換終了割り込みルーチン                                          */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( adend)
void  adend( void )
{
    AD.ADCSR0 &= 0x7f;                  /* Ａ／Ｄエンドフラグクリア           */
    ad_dat = AD.ADDR0;                  /* Ａ／Ｄデータの格納                 */
}
```

仕様

（1）図1に示すように、1グループ（AN0～11の12チャネル）に入力される電圧をSH7050のA／D変換器を使用し、測定します。
（2）測定結果は24バイトのRAMに格納します。
（3）入力する電圧は0～5Vの範囲です。

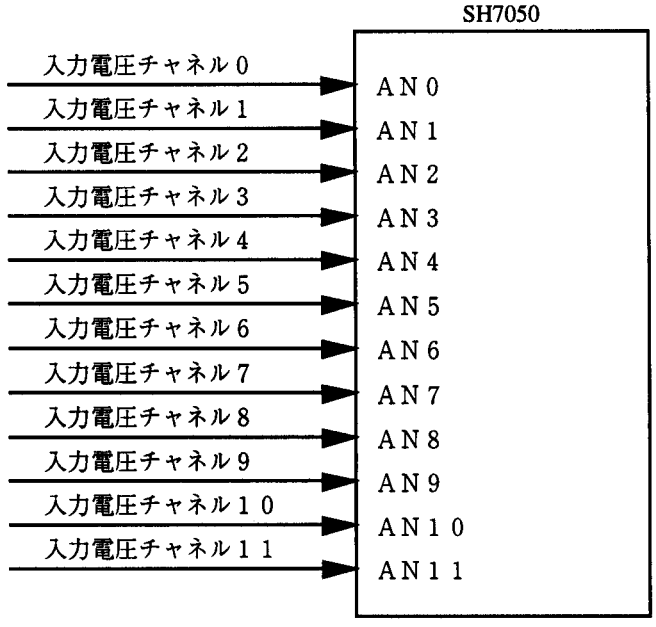


図1　SH7050による電圧の測定ブロック図

使用機能説明

表1に本タスク例の機能割付を示します。表1に示すようにSH7050に内蔵されているA／D変換器の機能を割付、A／D変換を行ないます。

表1　A／D変換機能割付

A／D変換機能		機　能
端子	AN0～11	アナログ電圧を入力する。
レジスタ	ADCSR0	A／D変換のモード（単一モード／スキャンモード）、測定端子の選択を設定する。
	ADCR0	A／D変換器のクロックの選択、測定の開始及び終了を設定する。
	ADDR0～11	A／D変換の結果を設定する。

動作説明

図２に動作原理を示します。図２に示すように、ＳＨ７０５０のハードウェア処理及びソフトウェア処理により１２チャネルのＡ／Ｄ変換を行います。

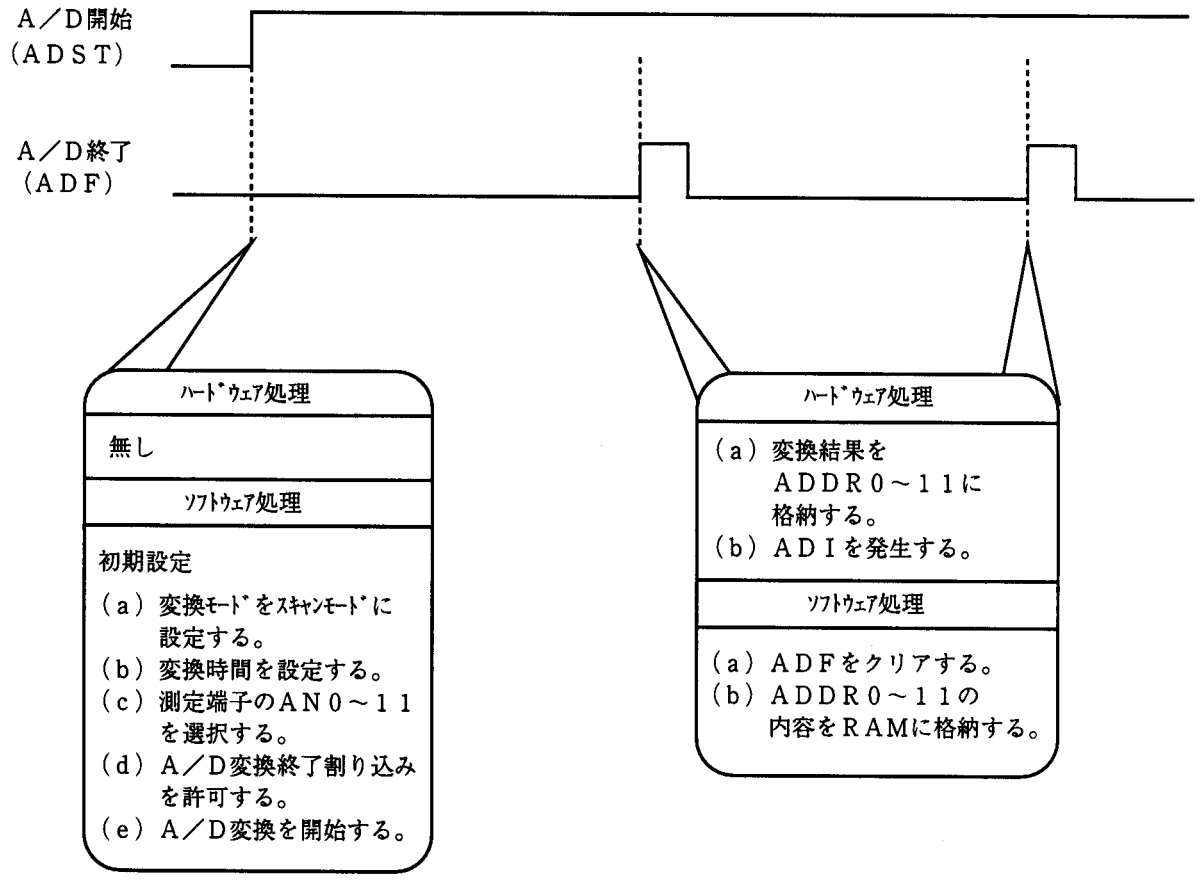


図２　Ａ／Ｄ変換動作原理

スキャンモードによるA／D変換	MCU	SH7050	使用機能	A／D

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
ﾒｲﾝﾙｰﾁﾝ	adscanmn	A／D変換器の初期設定を行なう。
A／D変換終了	adend	ＡＤＩにより起動し、結果をＲＡＭに格納する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
ａｄ．ｄａｔ０	ＡＮ０に入力した電圧のA／D変換データを格納する。	unsigned short	A／D変換終了
ａｄ．ｄａｔ１	ＡＮ１に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ２	ＡＮ２に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ３	ＡＮ３に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ４	ＡＮ４に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ５	ＡＮ５に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ６	ＡＮ６に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ７	ＡＮ７に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ８	ＡＮ８に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ９	ＡＮ９に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ１０	ＡＮ１０に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ１１	ＡＮ１１に入力した電圧のA／D変換データを格納する。		

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
AD.ADCSR0	A／D変換モードを１２キャネルスキャンモードに設定、測定端子をＡＮ０～１１に設定する。	0x73	ﾒｲﾝﾙｰﾁﾝ A／D変換終了
AD.ADCR0	A／D変換器のクロックを１３４ステートに設定する。	0x7f	ﾒｲﾝﾙｰﾁﾝ
INT.IPRG	A／D０の割り込み優先ﾚﾍﾞﾙを設定する。	0x00f0	ﾒｲﾝﾙｰﾁﾝ
AD.ADDR0	A／D変換の結果を設定する。		A／D変換終了

（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
ｏｆｓｅｔ	A／D変換ﾃﾞｰﾀの格納先頭ｱﾄﾞﾚｽからのｵﾌｾｯﾄを格納する。	unsigned short	A／D変換終了

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void adscanmn(void);
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
volatile  struct add
{
    short  dat0;                        /* A／D変換データ0                 */
    short  dat1;                        /* A／D変換データ1                 */
    short  dat2;                        /* A／D変換データ2                 */
    short  dat3;                        /* A／D変換データ3                 */
    short  dat4;                        /* A／D変換データ4                 */
    short  dat5;                        /* A／D変換データ5                 */
    short  dat6;                        /* A／D変換データ6                 */
    short  dat7;                        /* A／D変換データ7                 */
    short  dat8;                        /* A／D変換データ8                 */
    short  dat9;                        /* A／D変換データ9                 */
    short  dat10;                       /* A／D変換データ10                */
    short  dat11;                       /* A／D変換データ11                */
};
#define ad (*(struct add *)0xFFFFE800)
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void adscanmn( void )
{
    AD.ADCSR0 = 0x73;           /*A/D割り込み許可,ｽｷｬﾝﾓｰﾄﾞ,測定端子:AN0～11  */
    AD.ADCR0 = 0x7f;                    /* 測定時間134ｽﾃｰﾄ,A/D変換スタート    */
    INT.IPRG = 0x00f0;                  /* A/D0割り込み優先ﾚﾍﾞﾙを15に設定     */
    set_imask(0x0);                     /* 割り込み許可                       */
    while(1);                           /* 無限ループ（割り込み待ち）         */
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* A／D変換終了割り込みルーチン                                            */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( adend)
void  adend( void )
{
    char ofset;

    AD.ADCSR0 &= 0x7f;                  /* A／Dエンドフラグクリア          */
    for( ofset = 0; ofset < 12; ofset++ )
    {
        *(short *)(&ad.dat0 + ofset) =
            *(short *)(&AD.ADDR0+ofset); /* A／Dデータの格納(ADDR0～11)     */
    }
}
```

2.21　外部トリガによるA／D変換	MCU	ＳＨ７０５０	使用機能	A／D

仕様

（1）図1に示すように、1チャネルに入力される電圧をＳＨ７０５０のA／D変換器を使用し、測定します。
（2）測定結果は2バイトのＲＡＭに格納します。
（3）入力する電圧は0～5Vの範囲です。

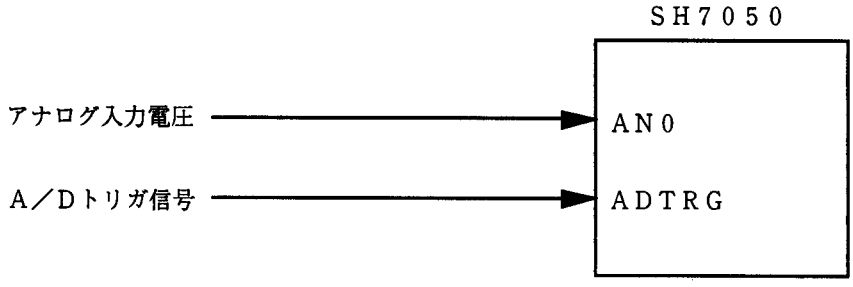


図1　ＳＨ７０５０による電圧の測定ブロック図

使用機能説明

表1、表2に本タスク例の機能割付を示します。表1、表2に示すようにＳＨ７０５０に内蔵されている
A／D変換器、ＰＦＣの機能を割り付け、A／D変換を行ないます。

表1　A／D変換機能割付

A／D変換機能		機能
端子	ＡＮ０	アナログ電圧を入力する。
	ＡＤＴＲＧ	A／Dトリガ信号を入力する。
レジスタ	ＡＤＣＳＲ０	A／D変換のモード（単一モード／スキャンモード）の設定、測定端子、クロックの選択をする。
	ＡＤＣＲ０	A／D変換器の測定の開始及び終了を設定する。
	ＡＤＤＲ０	A／D変換の結果を設定する。

表2　　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
ＰＧＩＯＲ	端子の入出力方向を設定する。
ＰＧＣＲ２	端子の機能を選択する。

動作説明

図2に動作原理を示します。図2に示すように、SH7050のハードウェア処理及びソフトウェア処理により1チャネルのA／D変換を行います。

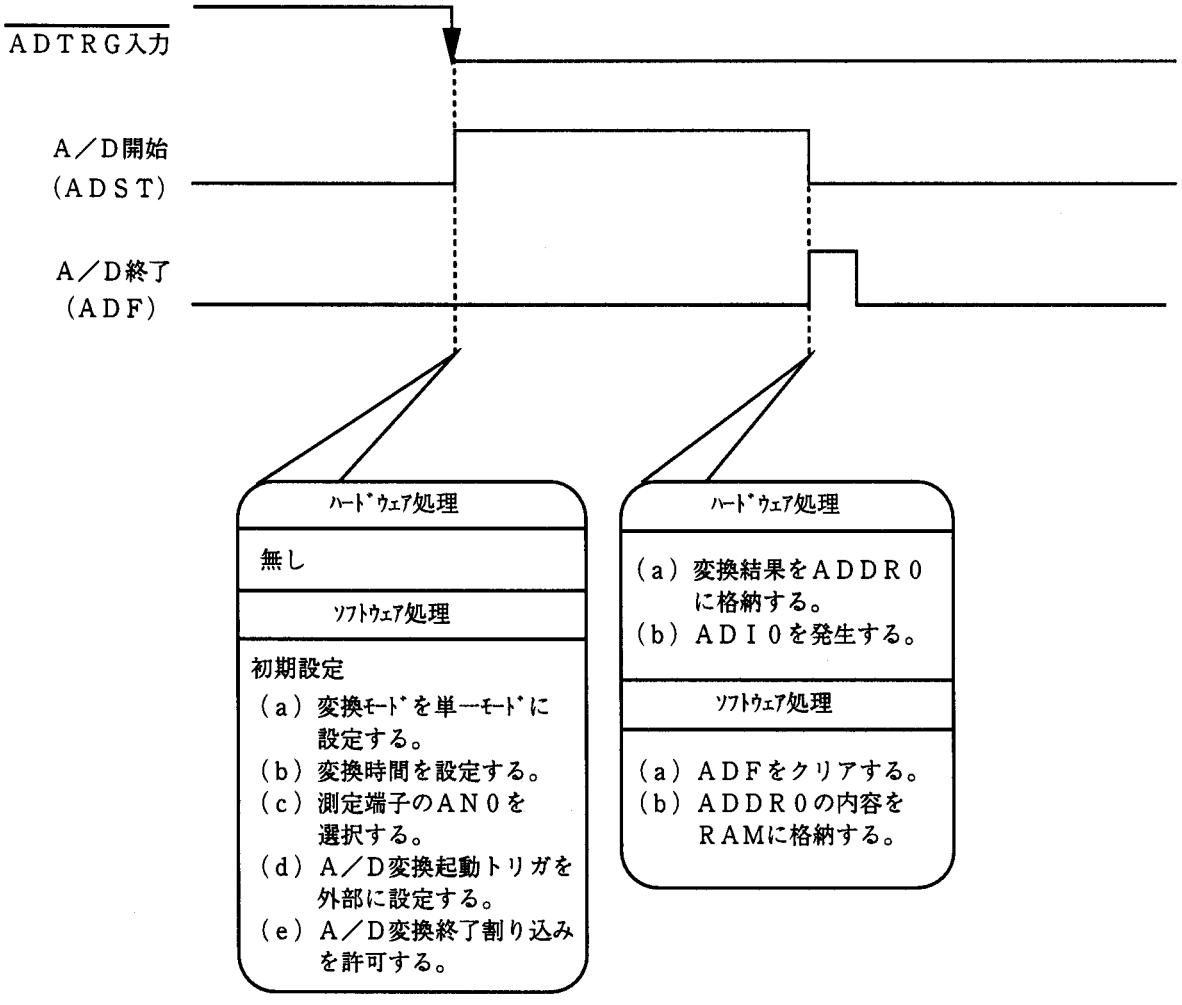


図2　A／D変換動作原理

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
ﾒｲﾝﾙｰﾁﾝ	adtrgmn	A／D変換器の初期設定を行なう。
A／D変換終了	adend	ＡＤＩ０により起動し、結果をＲＡＭに格納する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
ａｄ＿ｄａｔ	A／D変換データを格納する。	unsigned short	A／D変換終了

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
AD.ADCSR0	A／D変換モードを単一モードに設定、 測定端子をＡＮ０に設定する。	0x40	ﾒｲﾝﾙｰﾁﾝ
AD.ADCR0	A／D変換器のクロックを１３４ステートに設定する。	0xdf	
INT.IPRG	A／Ｄ０の割り込み優先ﾚﾍﾞﾙを設定する。	0x00f0	
PFC.PGIOR	ＰＧ０を入力に設定する。	0x0000	
PFC.PGCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをA／D変換トリガ入力に設定する。	0x0004	
AD.ADDR0	A／D変換の結果を設定する。		A／D変換終了

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ             */
/*-------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                    */
/*-------------------------------------------------------------------------*/
void adtrgmn( void );
/*-------------------------------------------------------------------------*/
/* 変数定義                                                                */
/*-------------------------------------------------------------------------*/
#define  ad_dat  (*(unsigned short *)0xFFFFE800) /* A/D変換ﾃﾞｰﾀ格納          */
/*-------------------------------------------------------------------------*/
/* メインルーチン                                                          */
/*-------------------------------------------------------------------------*/
void adtrgmn( void )
{
    PFC.PGCR2 = 0x0004;                 /* Ａ／Ｄトリガ入力端子に設定          */
    AD.ADCSR0 = 0x40;                   /* A/D割込許可,単一ﾓｰﾄﾞ,測定端子:AN0   */
    AD.ADCR0 = 0xdf;                    /* 外部ﾄﾘｶﾞ,測定時間134ｽﾃｰﾄ           */
    INT.IPRG = 0x00f0;                  /* A/D0割り込み優先ﾚﾍﾞﾙを１５に設定    */
    set_imask(0x0);                     /* 割り込み許可                        */
    while(1);                           /* 無限ループ（割り込み待ち）          */
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* Ａ／Ｄ変換終了割り込みルーチン                                          */
/*                                                                         */
/*-------------------------------------------------------------------------*/
#pragma interrupt( adend)
void  adend( void )
{
    AD.ADCSR0 &= 0x7f;                  /* Ａ／Ｄエンドフラグクリア            */
    ad_dat = AD.ADDR0;                  /* Ａ／Ｄデータの格納                  */
}
```

2.22　A／D変換ｴﾝﾄﾞ信号によるA／D端子切り替え	MCU	ＳＨ７０５０	使用機能	A／D

仕様

（1）図1に示すように、8チャネルに入力される電圧をＳＨ７０５０のA／D変換器を使用し、測定します。
　　チャネルの切り換えはA／D変換終了信号をカウンタでカウントし、マルチプレクサによって切り換えます。
（2）測定結果は１６バイトのＲＡＭに格納します。
（3）入力する電圧は０～５Ｖの範囲です。

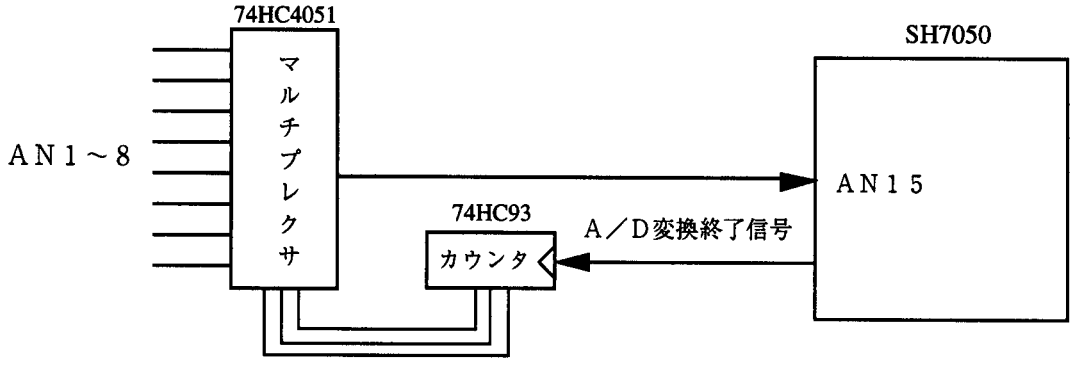


図1　ＳＨ７０５０による電圧の測定ブロック図

使用機能説明

表1、表2に本タスク例の機能割付を示します。表1、表2に示すようにＳＨ７０５０に内蔵されているA／D変換器、ＰＦＣの機能を割付、A／D変換を行ないます。

表1　A／D変換機能割付

A／D変換機能		機　能
端子	ＡＮ１５	アナログ電圧を入力する。
	ＡＤＥＮＤ	A／D変換終了信号を出力する。
レジスタ	ＡＤＣＳＲ１	A／D変換のモード（単一モード／スキャンモード）の設定、測定端子、クロックの選択をする。
	ＡＤＣＲ１	A／D変換器の測定の開始及び終了を設定する。
	ＡＤＤＲ１５	A／D変換の結果を設定する。

表2　ＰＦＣ機能割り付け

ＰＦＣレジスタ	機能
ＰＣＩＯＲ	端子の入出力方向を設定する。
ＰＣＣＲ２	端子の機能を選択する。

動作説明

図２に動作原理を示します。図２に示すように、ＳＨ７０５０のハードウェア処理及びソフトウェア処理により８チャネルのＡ／Ｄ変換を行います。

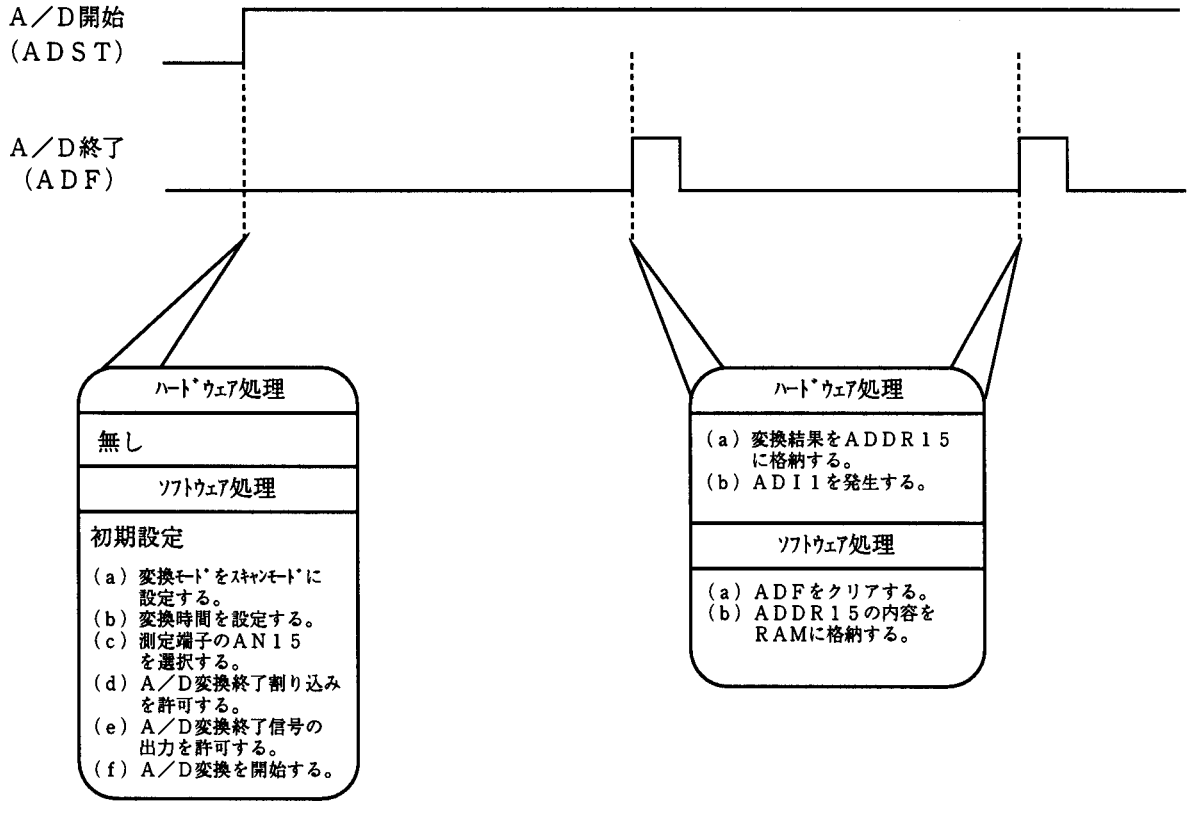


図２　Ａ／Ｄ変換動作原理

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
ﾒｲﾝﾙｰﾁﾝ	adendmn	A／D変換器の初期設定を行なう。
A／D変換終了	adend	ＡＤＩ１により起動し、結果をＲＡＭに格納する。

（2）引数の説明

ﾗﾍﾞﾙ名	機　能	ﾃﾞｰﾀ長	使用ﾓｼﾞｭｰﾙ名
ａｄ．ｄａｔ０	ＡＮ０に入力した電圧のA／D変換データを格納する。	unsigned short	A／D変換終了
ａｄ．ｄａｔ１	ＡＮ１に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ２	ＡＮ２に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ３	ＡＮ３に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ４	ＡＮ４に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ５	ＡＮ５に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ６	ＡＮ６に入力した電圧のA／D変換データを格納する。		
ａｄ．ｄａｔ７	ＡＮ７に入力した電圧のA／D変換データを格納する。		
ｏｆｓｅｔ	A／D変換データの格納場所を設定する。		

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
AD.ADCSR1	A／D変換モードをｽｷｬﾝモード、変換器のｸﾛｯｸを134ｽﾃｰﾄ、測定端子をＡＮ１５に設定する。	0x7b	ﾒｲﾝﾙｰﾁﾝ
AD.ADCR1	外部トリガによるA／D変換禁止に設定する。	0x00	
INT.IPRE	A／D１の割り込み優先ﾚﾍﾞﾙを設定する。	0x000f	
PFC.PCIOR	ＰＣ６を出力に設定する。	0x0040	
PFC.PCCR2	端子ﾏﾙﾁﾌﾟﾚｸｽをA／D変換終了信号出力に設定する。	0x3000	
AD.ADDR15	A／D変換の結果を設定する。		A／D変換終了

（4）使用ＲＡＭ

本タスクでは引数以外のＲＡＭは使用していません。

プログラムリスト

```
#include <machine.h>                    /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"                     /* 周辺レジスタ定義ﾍｯﾀﾞﾌｧｲﾙ            */
/*--------------------------------------------------------------------------*/
/* 関数プロトタイプ宣言                                                     */
/*--------------------------------------------------------------------------*/
void adendmn( void );
/*--------------------------------------------------------------------------*/
/* 変数定義                                                                 */
/*--------------------------------------------------------------------------*/
volatile  struct add
{
    short  dat0;                        /* A／D変換データ0                  */
    short  dat1;                        /* A／D変換データ1                  */
    short  dat2;                        /* A／D変換データ2                  */
    short  dat3;                        /* A／D変換データ3                  */
    short  dat4;                        /* A／D変換データ4                  */
    short  dat5;                        /* A／D変換データ5                  */
    short  dat6;                        /* A／D変換データ6                  */
    short  dat7;                        /* A／D変換データ7                  */
};
#define ad     (*(struct add *)0xFFFFE800)
#define ofset  (*(unsigned char  *)0xFFFFE810)    /* 変換データ格納ｵﾌｾｯﾄ     */

/*--------------------------------------------------------------------------*/
/* メインルーチン                                                           */
/*--------------------------------------------------------------------------*/
void adendmn( void )
{
    PFC.PCIOR = 0x0040;                 /* ＰＣ６を出力端子に設定する       */
    PFC.PCCR2 = 0x3000;                 /* 端子ﾏﾙﾁﾌﾟﾚｸｽをADENDの使用にする  */
    AD.ADCSR1 = 0x7b;                   /* ｽｷｬﾝﾓｰﾄﾞ,測定端子AN15,134ｽﾃｰﾄ   */
    AD.ADCR1 = 0x00;                    /* 外部トリガによるＡ／Ｄ変換禁止   */
    ofset = 0;                          /* 変換データ格納オフセット初期化   */
    INT.IPRG = 0x000f;                  /* A/D1割り込み優先ﾚﾍﾞﾙを１５に設定 */
    set_imask(0x0);                     /* 割り込み許可                     */
    while(1);                           /* 無限ループ（割り込み待ち）       */
}
/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Ａ／Ｄ変換終了割り込みルーチン                                           */
/*                                                                          */
/*--------------------------------------------------------------------------*/
#pragma interrupt( adend)
void  adend( void )
{
    if( ofset >= 8 )
    {
        ofset = 0;                      /* 変換データ格納オフセット初期化   */
    }
    *( &ad.dat0 + ofset ) = AD.ADDR15; /* Ａ／Ｄデータの格納                */
    ofset++;
    AD.ADCSR1 &= 0x7f;                  /* Ａ／Ｄエンドフラグクリア          */
}
```

仕様

（1）図１に示すようにＰＯＥ端子にＬＯＷレベルを入力した時、ＰＡ０端子の出力を遮断します。

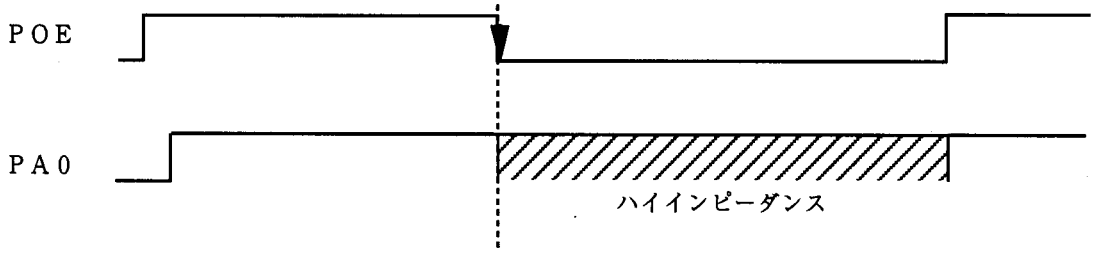


図１　端子の出力遮断動作ブロック図

機能説明

表１に本タスク例の機能割付を示します。表１に示すようにＳＨ７０５０に内蔵しているＩ／Ｏポートの機能を割付、端子の出力遮断を行ないます。

表１　Ｉ／Ｏポート機能割付

Ｉ／Ｏポート機能	機　能
PFC.PAIOR	端子の入出力を設定する。
PFC.PBIOR	
PFC.PACR	端子の機能を選択する。
PFC.PBCR	
PFC.PADR	ポートから入出力をする。
PFC.PBDR	

動作説明

図２に動作原理を示します。図２に示すように、ＳＨ７０５０のハードウェア処理及びソフトウェア処理により端子の出力遮断を行います。

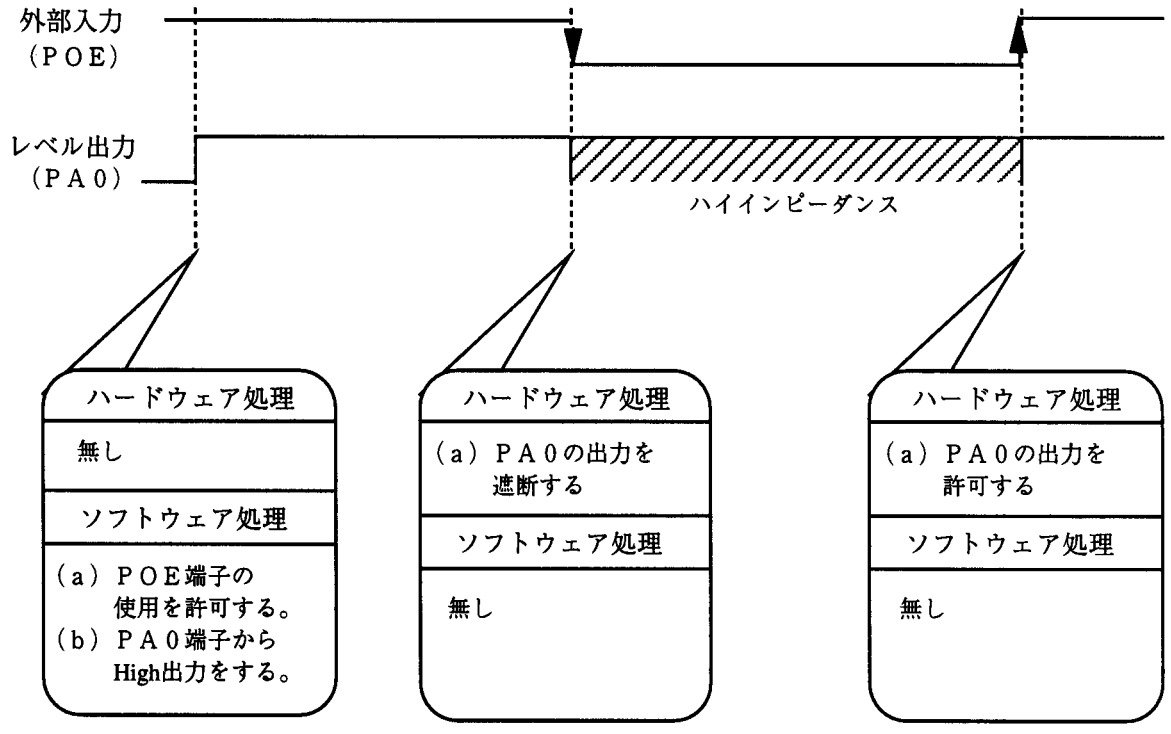


図２　端子の出力遮断動作原理

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
ﾒｲﾝﾙｰﾁﾝ	portmn	I／Oポートの初期設定を行なう。

（2）引数の説明

本タスクでは引き数は使用してません

（3）使用内部レジスタ説明

ﾚｼﾞｽﾀ名	機　能	設定値	使用ﾓｼﾞｭｰﾙ名
PFC.PAIOR	ＰＡ６を出力端子に設定する。	0x0001	ﾒｲﾝﾙｰﾁﾝ
PFC.PACR	端子ﾏﾙﾁﾌﾟﾚｸｽをﾎﾟｰﾄの使用にする。	0x0000	
PFC.PBIOR	ＰＢ１１を入力端子に設定する。	0x0000	
PFC.PBCR	端子ﾏﾙﾁﾌﾟﾚｸｽをＰＯＥの使用にする。	0xc0c0	
PFC.PADR	Ｈｉｇｈレベルを出力する。	0x0001	

（4）使用ＲＡＭ

本タスクでは引き数以外のＲＡＭは使用してません

プログラムリスト

```
#include <machine.h>                /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ                */
#include "sh7050.h"
/*-----------------------------------------------------------------------------*/
/*              関数プロトタイプ宣言                                           */
/*-----------------------------------------------------------------------------*/
void portmn( void );
/*-----------------------------------------------------------------------------*/
/*                                                                             */
/*              メインルーチン                                                 */
/*                                                                             */
/*-----------------------------------------------------------------------------*/
void portmn( void )
{
  PFC.PAIOR = 0x0001;           /* ＰＡ０を出力端子に設定する          */
  PFC.PACR  = 0x0000;           /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＰＡ０の使用にする */
  PFC.PBIOR = 0x0000;           /* ＰＯＥを入力端子に設定する          */
  PFC.PBCR  = 0xc0c0;           /* 端子ﾏﾙﾁﾌﾟﾚｸｽをＰＯＥの使用にする  */
  PFC.PADR = 0x0001;            /* Ｈｉｇｈレベル出力                  */
  sleep();
}
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* インターラプトプライオリティ                                           */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct intr
{
    short  IPRA;          /* 割り込み優先レベル設定 レジスタＡ            */
    short  IPRB;          /* 割り込み優先レベル設定 レジスタＢ            */
    short  IPRC;          /* 割り込み優先レベル設定 レジスタＣ            */
    short  IPRD;          /* 割り込み優先レベル設定 レジスタＤ            */
    short  IPRE;          /* 割り込み優先レベル設定 レジスタＥ            */
    short  IPRF;          /* 割り込み優先レベル設定 レジスタＦ            */
    short  IPRG;          /* 割り込み優先レベル設定 レジスタＧ            */
    short  IPRH;          /* 割り込み優先レベル設定 レジスタＨ            */
    short  ICR;           /* 割り込みコントロールレジスタ                 */
    short  ISR;           /* ＩＲＱステータスレジスタ                     */
};
#define INT (*(volatile  struct intr *)0xffff8348)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* ユーザブレイクコントローラ                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct ubc
{
    short  UBARH ;        /* ユーザブレークアドレスレジスタＨ             */
    short  UBARL ;        /* ユーザブレークアドレスレジスタＬ             */
    short  UBAMRH;        /* ユーザブレークアドレスマスクレジスタＨ       */
    short  UBAMRL;        /* ユーザブレークアドレスマスクレジスタＬ       */
    short  UBBR  ;        /* ユーザブレークバスサイクルレジスタ           */
};
#define UBC (*(volatile  struct ubc *)0xffff8600)
/*------------------------------------------------------------------------*/
/* ユーザブレークアドレスレジスタ（３２ビットアクセス用）                 */
/*------------------------------------------------------------------------*/
#define UBCL_UBAR (*(long *)0xffff8600)
/*------------------------------------------------------------------------*/
/* ユーザブレークアドレスマスクレジスタ（３２ビットアクセス用）           */
/*------------------------------------------------------------------------*/
#define UBCL_UBAMR (*(long *)0xffff8604)
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ウエイトステートコントロール                                           */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct wt
{
    short  BCR1;          /* バスコントロール レジスタ１                  */
    short  BCR2;          /* バスコントロール レジスタ２                  */
    short  WCR1;          /* ウエイトコントロール レジスタ１              */
    short  WCR2;          /* ウエイトコントロール レジスタ２              */
    short  RAMER;         /* ＲＡＭエミュレーションレジスタ               */
};
#define WT (*(volatile  struct wt *)0xffff8620)
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ダイレクトメモリアクセスコントロール                                   */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct dma
{
    long  SAR0;           /* DMAソースアドレスレジスタ０                  */
    long  DAR0;           /* DMAディスチネーションアドレスレジスタ０      */
    long  TCR0;           /* DMAトランスファカウントレジスタ０            */
    long  CHCR0;          /* DMAチャネルコントロールレジスタ０            */
    long  SAR1;           /* DMAソースアドレスレジスタ１                  */
    long  DAR1;           /* DMAディスチネーションアドレスレジスタ１      */
    long  TCR1;           /* DMAトランスファカウントレジスタ１            */
    long  CHCR1;          /* DMAチャネルコントロールレジスタ１            */
    long  SAR2;           /* DMAソースアドレスレジスタ２                  */
    long  DAR2;           /* DMAディスチネーションアドレスレジスタ２      */
    long  TCR2;           /* DMAトランスファカウントレジスタ２            */
    long  CHCR2;          /* DMAチャネルコントロールレジスタ２            */
    long  SAR3;           /* DMAソースアドレスレジスタ３                  */
    long  DAR3;           /* DMAディスチネーションアドレスレジスタ３      */
    long  TCR3;           /* DMAトランスファカウントレジスタ３            */
    long  CHCR3;          /* DMAチャネルコントロールレジスタ３            */
} ;
#define DMA (*(volatile  struct dma *)0xffff86c0)
/*------------------------------------------------------------------------*/
/* DMAオペレーションレジスタ                                              */
/*------------------------------------------------------------------------*/
#define DMAOR (*(volatile  unsigned short *)0xffff86b0)
/*------------------------------------------------------------------------*/
/*                                                                        */
/* アドバンストタイマユニット                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/

/*------------------------------------------------------------------------*/
/* チャネル０                                                             */
/*------------------------------------------------------------------------*/
volatile  struct ch0
{
    char   TGSR;          /* トリガセクションレジスタ                     */
    char   TIOR0A;        /* タイマI/Oコントロールレジスタ０Ａ            */
    char   ITVRR;         /* インターバルインターラプトリクエストレジスタ */
    char   TSRAH;         /* タイマステータスレジスタＡＨ                 */
    char   TIERA;         /* タイマインターラプトトイネーブルレジスタＡ    */
    char   TSRAL;         /* タイマステータスレジスタＡＬ                 */
    short  dummy1;        /* 不使用                                       */
    short  TCNT0H;        /* タイマカウンタレジスタ０Ｈ                   */
    short  TCNT0L;        /* タイマカウンタレジスタ０Ｌ                   */
    short  ICR0AH;        /* インプットキャプチャレジスタ０ＡＨ           */
    short  ICR0AL;        /* インプットキャプチャレジスタ０ＡＬ           */
    short  ICR0BH;        /* インプットキャプチャレジスタ０ＢＨ           */
    short  ICR0BL;        /* インプットキャプチャレジスタ０ＢＬ           */
    short  ICR0CH;        /* インプットキャプチャレジスタ０ＣＨ           */
    short  ICR0CL;        /* インプットキャプチャレジスタ０ＣＬ           */
    short  ICR0DH;        /* インプットキャプチャレジスタ０ＤＨ           */
    short  ICR0DL;        /* インプットキャプチャレジスタ０ＤＬ           */
} ;
#define C0 (*(volatile  struct ch0 *)0xffff8280)
```

```
/*-------------------------------------------------------------------------*/
/* チャネル０（３２ビットアクセス用）                                      */
/*-------------------------------------------------------------------------*/
volatile  struct ch0l
{
    long  TCNT0;          /* タイマカウンタレジスタ                        */
    long  ICR0A;          /* インプットキャプチャレジスタ０Ａ              */
    long  ICR0B;          /* インプットキャプチャレジスタ０Ｂ              */
    long  ICR0C;          /* インプットキャプチャレジスタ０Ｃ              */
    long  ICR0D;          /* インプットキャプチャレジスタ０Ｄ              */
} ;
#define C0L (*(volatile  struct ch0l *)0xffff8288)

/*-------------------------------------------------------------------------*/
/* チャネル１，２                                                          */
/*-------------------------------------------------------------------------*/
volatile  struct ch1
{
    char   TCR1;          /* タイマコントロールレジスタ１                  */
    char   TIOR1A;        /* タイマI/Oコントロールレジスタ１Ａ             */
    char   TIOR1B;        /* タイマI/Oコントロールレジスタ１Ｂ             */
    char   TIOR1C;        /* タイマI/Oコントロールレジスタ１Ｃ             */
    char   TIERB;         /* タイマインターラプトトイネーブルレジスタＢ    */
    char   TSRB;          /* タイマステータスレジスタＢ                    */
    char   TCR2;          /* タイマコントロールレジスタ２                  */
    char   TIOR2A;        /* タイマI/Oコントロールレジスタ２Ａ             */
    char   TIERC;         /* タイマインターラプトトイネーブルレジスタＣ    */
    char   TSRC;          /* タイマステータスレジスタＣ                    */
    short  TCNT2;         /* タイマカウンタレジスタ２                      */
    short  GR2A;          /* ジェネラルレジスタ２Ａ                        */
    short  GR2B;          /* ジェネラルレジスタ２Ｂ                        */
    short  TCNT1;         /* タイマカウンタレジスタ１                      */
    short  GR1A;          /* ジェネラルレジスタ１Ａ                        */
    short  GR1B;          /* ジェネラルレジスタ１Ｂ                        */
    short  GR1C;          /* ジェネラルレジスタ１Ｃ                        */
    short  GR1D;          /* ジェネラルレジスタ１Ｄ                        */
    short  GR1E;          /* ジェネラルレジスタ１Ｅ                        */
    short  GR1F;          /* ジェネラルレジスタ１Ｆ                        */
    short  OSBR;          /* ジェネラルレジスタ１Ｆ                        */
    char   TCR10;         /* タイマコントロールレジスタ１０                */
    char   TCNR;          /* タイマコネクションレジスタ                    */
    char   TIERF;         /* タイマインタラプトイネーブルレジスタＦ        */
    char   TSRF;          /* タイマステータスレジスタＦ                    */
    char   dummy2;        /* 不使用                                        */
    char   DCER;          /* ダウンカウントイネーブレジスタ                */
    short  dummy3;        /* 不使用                                        */
    char   dummy4;        /* 不使用                                        */
    char   PSCR1;         /* プリスケーラ２                                */
    short  TSTR;          /* タイマスタートレジスタ                        */
} ;
#define C1 (*(volatile  struct ch1 *)0xffff82c0)
```

```
/*-----------------------------------------------------------------------*/
/* チャネル３～５                                                        */
/*-----------------------------------------------------------------------*/
volatile  struct ch35
{
    short  TMDR;          /* タイマモードレジスタ                        */
    char   TIERDH;        /* タイマインタラプトイネーブルレジスタＤＨ    */
    char   TSRDH;         /* タイマステータスレジスタＤＨ                */
    char   TIERDL;        /* タイマインタラプトイネーブルレジスタＤＬ    */
    char   TSRDL;         /* タイマステータスレジスタＤＬ                */
    char   TCR3;          /* タイマコントロールレジスタ３                */
    char   TCR4;          /* タイマコントロールレジスタ４                */
    char   TIOR3A;        /* タイマＩ／Ｏコントロールレジスタ３Ａ        */
    char   TIOR3B;        /* タイマＩ／Ｏコントロールレジスタ３Ｂ        */
    char   TIOR4A;        /* タイマＩ／Ｏコントロールレジスタ４Ａ        */
    char   TIOR4B;        /* タイマＩ／Ｏコントロールレジスタ４Ｂ        */
    char   TCR5;          /* タイマコントロールレジスタ５                */
    char   TIOR5A;        /* タイマＩ／Ｏコントロールレジスタ５Ａ        */
    short  TCNT3;         /* タイマカウンタ３                            */
    short  GR3A;          /* ゼネラルレジスタ３Ａ                        */
    short  GR3B;          /* ゼネラルレジスタ３Ｂ                        */
    short  GR3C;          /* ゼネラルレジスタ３Ｃ                        */
    short  GR3D;          /* ゼネラルレジスタ３Ｄ                        */
    short  TCNT4;         /* タイマカウンタ４                            */
    short  GR4A;          /* ゼネラルレジスタ４Ａ                        */
    short  GR4B;          /* ゼネラルレジスタ４Ｂ                        */
    short  GR4C;          /* ゼネラルレジスタ４Ｃ                        */
    short  GR4D;          /* ゼネラルレジスタ４Ｄ                        */
    short  TCNT5;         /* タイマカウンタ５                            */
    short  GR5A;          /* ゼネラルレジスタ５Ａ                        */
    short  GR5B;          /* ゼネラルレジスタ５Ｂ                        */
} ;
#define C35 (*(volatile  struct ch35 *)0xffff8200)
/*-----------------------------------------------------------------------*/
/* チャネル６～９                                                        */
/*-----------------------------------------------------------------------*/
volatile  struct ch69
{
    char   TIERE;         /* タイマインタラプトイネーブルレジスタＥ      */
    char   TSRE;          /* タイマステータスレジスタＥ                  */
    char   TCR7;          /* タイマコントロールレジスタ７                */
    char   TCR6;          /* タイマコントロールレジスタ６                */
    char   TCR9;          /* タイマコントロールレジスタ９                */
    char   TCR8;          /* タイマコントロールレジスタ８                */
    short  TCNT6;         /* タイマカウンタレジスタ６                    */
    short  CYLR6;         /* サイクルレジスタ６                          */
    short  BFR6;          /* バッファーレジスタ６                        */
    short  DTR6;          /* デューティレジスタ６                        */
    short  TCNT7;         /* タイマカウンタレジスタ７                    */
    short  CYLR7;         /* サイクルレジスタ７                          */
    short  BFR7;          /* バッファーレジスタ７                        */
    short  DTR7;          /* デューティレジスタ７                        */
    short  TCNT8;         /* タイマカウンタレジスタ８                    */
    short  CYLR8;         /* サイクルレジスタ８                          */
    short  BFR8;          /* バッファーレジスタ８                        */
    short  DTR8;          /* デューティレジスタ８                        */
    short  TCNT9;         /* タイマカウンタレジスタ９                    */
    short  CYLR9;         /* サイクルレジスタ９                          */
    short  BFR9;          /* バッファーレジスタ９                        */
    short  DTR9;          /* デューティレジスタ９                        */
} ;
#define C69 (*(volatile  struct ch69 *)0xffff8240)
```

```
/*------------------------------------------------------------------------*/
/* チャネル１０                                                           */
/*------------------------------------------------------------------------*/
volatile  struct ch10
{
    short  DCNT10A;         /* ダウンカウントレジスタ１０Ａ               */
    short  DCNT10B;         /* ダウンカウントレジスタ１０Ｂ               */
    short  DCNT10C;         /* ダウンカウントレジスタ１０Ｃ               */
    short  DCNT10D;         /* ダウンカウントレジスタ１０Ｄ               */
    short  DCNT10E;         /* ダウンカウントレジスタ１０Ｅ               */
    short  DCNT10F;         /* ダウンカウントレジスタ１０Ｆ               */
    short  DCNT10G;         /* ダウンカウントレジスタ１０Ｇ               */
    short  DCNT10H;         /* ダウンカウントレジスタ１０Ｈ               */
} ;
#define C10 (*(volatile  struct ch10 *)0xffff82f0)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* アドバンストパルスコントローラ                                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define POPCR (*(volatile  short *)0xffff83c0)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* ウォッチドッグタイマ                                                   */
/*                                                                        */
/*------------------------------------------------------------------------*/
/* タイマコントロールレジスタ                                             */
#define WDT_TCSRR     (*(volatile  char  *)0xffff8610)
/* タイマコントロールレジスタ                                             */
#define WDT_TCSRW     (*(volatile  short *)0xffff8610)
/* タイマカウンタ (ﾗｲﾄ)                                                   */
#define WDT_TCNTW     (*(volatile  short *)0xffff8610)
/* タイマカウンタ (ﾘｰﾄﾞ)                                                  */
#define WDT_TCNTR     (*(volatile  char  *)0xffff8611)
/* リセットコントロール／ステータスﾚｼﾞｽﾀ (ﾗｲﾄ)                           */
#define WDT_RSTCSRW   (*(volatile  short *)0xffff8612)
/* リセットコントロール／ステータスﾚｼﾞｽﾀ (ﾘｰﾄﾞ)                          */
#define WDT_RSTCSRR   (*(volatile  char  *)0xffff8613)
/* スタンバイコントロールレジスタ                                         */
#define WDT_SBYCR     (*(volatile  short *)0xffff8614)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* コンペアマッチタイマ                                                   */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct cm
{
    short  CMSTR;           /* コンペアマッチタイマスタートレジスタ       */
    short  CMCSR0;          /* コンペアマッチタイﾏｺﾝﾄﾛｰﾙ／ｽﾃｰﾀｽレジスタ０ */
    short  CMCNT0;          /* コンペアマッチカウンタ０                   */
    short  CMCOR0;          /* コンペアマッチコンスタントレジスタ０       */
    short  CMCSR1;          /* コンペアマッチタイﾏｺﾝﾄﾛｰﾙ／ｽﾃｰﾀｽレジスタ１ */
    short  CMCNT1;          /* コンペアマッチカウンタ１                   */
    short  CMCOR1;          /* コンペアマッチコンスタントレジスタ１       */
};
#define CM (volatile  struct cm *)0xffff83d0)
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/* ＳＣＩ                                                                 */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct sci
{
    char   SMR;             /* シリアルモードレジスタ                     */
    char   BRR;             /* ビットレートレジスタ                       */
    char   SCR;             /* シリアルコントロールレジスタ               */
    char   TDR;             /* トランスミットデータレジスタ               */
    char   SSR;             /* シリアルステータスレジスタ                 */
    char   RDR;             /* レシーブデータレジスタ                     */
};
/*------------------------------------------------------------------------*/
/* チャネル０                                                             */
/*------------------------------------------------------------------------*/
#define SCI0 (*(volatile  struct sci *)0xffff81a0)
/*------------------------------------------------------------------------*/
/* チャネル１                                                             */
/*------------------------------------------------------------------------*/
#define SCI1 (*(volatile  struct sci *)0xffff81b0)
/*------------------------------------------------------------------------*/
/* チャネル２                                                             */
/*------------------------------------------------------------------------*/
#define SCI2 (*(volatile  struct sci *)0xffff81c0)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* Ａ／Ｄ変換                                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/
volatile  struct adc
{
    short  ADDR0;           /* Ａ／Ｄデータレジスタ０                     */
    short  ADDR1;           /* Ａ／Ｄデータレジスタ１                     */
    short  ADDR2;           /* Ａ／Ｄデータレジスタ２                     */
    short  ADDR3;           /* Ａ／Ｄデータレジスタ３                     */
    short  ADDR4;           /* Ａ／Ｄデータレジスタ４                     */
    short  ADDR5;           /* Ａ／Ｄデータレジスタ５                     */
    short  ADDR6;           /* Ａ／Ｄデータレジスタ６                     */
    short  ADDR7;           /* Ａ／Ｄデータレジスタ７                     */
    short  ADDR8;           /* Ａ／Ｄデータレジスタ８                     */
    short  ADDR9;           /* Ａ／Ｄデータレジスタ９                     */
    short  ADDR10;          /* Ａ／Ｄデータレジスタ１０                   */
    short  ADDR11;          /* Ａ／Ｄデータレジスタ１１                   */
    char   ADCSR0;          /* Ａ／Ｄコントロール／ステータスレジスタ０   */
    char   ADCR0;           /* Ａ／Ｄコントロールレジスタ０               */
    short  dummy5;          /* 不使用                                     */
    short  dummy6;          /* 不使用                                     */
    short  dummy7;          /* 不使用                                     */
    short  ADDR12;          /* Ａ／Ｄデータレジスタ１２                   */
    short  ADDR13;          /* Ａ／Ｄデータレジスタ１３                   */
    short  ADDR14;          /* Ａ／Ｄデータレジスタ１４                   */
    short  ADDR15;          /* Ａ／Ｄデータレジスタ１５                   */
    char   ADCSR1;          /* Ａ／Ｄコントロール／ステータスレジスタ１   */
    char   ADCR1;           /* Ａ／Ｄコントロールレジスタ１               */
};
#define AD (*(volatile  struct adc *)0xffff85d0)
```

```
/*-------------------------------------------------------------------------*/
/* A／Dトリガレジスタ                                                      */
/*-------------------------------------------------------------------------*/
#define ADTRGR  (*(volatile  char *)0xffff83b8)

/*-------------------------------------------------------------------------*/
/*                                                                         */
/* ピンファンクションコントローラ・I／Oポート                             */
/*                                                                         */
/*-------------------------------------------------------------------------*/
volatile  struct pt
{
    short  PADR;          /* ポートAデータ レジスタ1                       */
    short  PAIOR;         /* ポートA  I／Oレジスタ                         */
    short  PACR;          /* ポートAコントロール レジスタ1                 */
    short  PBDR;          /* ポートBデータ レジスタ1                       */
    short  PBIOR;         /* ポートB  I／Oレジスタ                         */
    short  PBCR;          /* ポートBコントロール レジスタ1                 */
    short  dummy8;        /* 不使用                                        */
    short  dummy9;        /* 不使用                                        */
    short  PCDR;          /* ポートCデータ レジスタ1                       */
    short  PCIOR;         /* ポートC  I／Oレジスタ                         */
    short  PCCR1;         /* ポートCコントロール レジスタ1                 */
    short  PCCR2;         /* ポートCコントロール レジスタ2                 */
    short  PDDR;          /* ポートDデータ レジスタ1                       */
    short  PDIOR;         /* ポートD  I／Oレジスタ                         */
    short  PDCR;          /* ポートDコントロール レジスタ1                 */
    short  dummy10;       /* 不使用                                        */
    short  PEDR;          /* ポートEデータ レジスタ1                       */
    short  PEIOR;         /* ポートE  I／Oレジスタ                         */
    short  PECR;          /* ポートEコントロール レジスタ1                 */
    short  PFDR;          /* ポートFデータ レジスタ1                       */
    short  PFIOR;         /* ポートF  I／Oレジスタ                         */
    short  PFCR1;         /* ポートFコントロール レジスタ1                 */
    short  PFCR2;         /* ポートFコントロール レジスタ2                 */
    short  PGDR;          /* ポートGデータ レジスタ1                       */
    short  PGIOR;         /* ポートG  I／Oレジスタ                         */
    short  PGCR1;         /* ポートGコントロール レジスタ1                 */
    short  PGCR2;         /* ポートGコントロール レジスタ2                 */
    short  PHDR;          /* ポートHデータ レジスタ1                       */
};
#define PFC (*(volatile  struct pt *)0xffff8380)

/*-------------------------------------------------------------------------*/
/*                                                                         */
/* フラッシュメモリ                                                        */
/*                                                                         */
/*-------------------------------------------------------------------------*/
volatile  struct fm
{
    char   FLMCR;         /* フラッシュメモリコントロールレジスタ          */
    char   dummy11;       /* 不使用                                        */
    char   EBR1;          /* 書込み／消去ブロック指定レジスタ1             */
    char   EBR2;          /* 書込み／消去ブロック指定レジスタ2             */
};
#define FM (*(volatile  struct fm *)0xffff8580)
```

```
/*-----------------------------------------------------------------------*/
/* ＲＡＭエミュレーションレジスタ                                        */
/*-----------------------------------------------------------------------*/
#define RAMER (*(volatile  char *)0xffff8628)

/*-----------------------------------------------------------------------*/
/*                                                                       */
/* システムコントロールレジスタ                                          */
/*                                                                       */
/*-----------------------------------------------------------------------*/
#define SYSCR (*(volatile  char *)0xffff83c8)
```

A. 付録

目　次

A.1 SRAMインタフェース例	MCU	SH7050(20MHz)	使用機能	モード2(内蔵ROM有効拡張モード)

仕様

（1）図1に示すように、SH7050のモード2(内蔵ROM有効拡張モード)により、256kビット(32k×8ビット) SRAM(HM62832H-35) 2個とのインタフェースを行います。

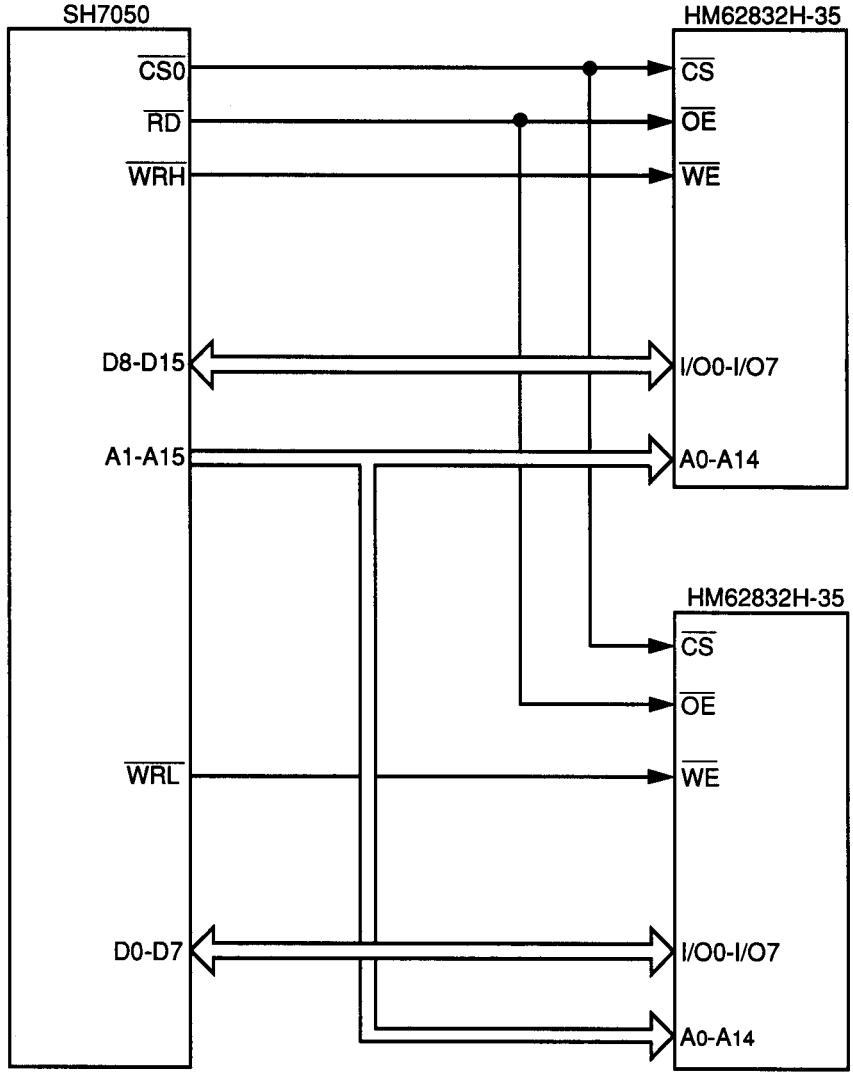


図1　SH7050およびHM62832H-35 接続ブロック図

（2）HM62832H-35は図2に示すように割り付けます。SH7050のバスステートコントローラを設定し、CS0空間を2ステート、ワードアクセス空間にします。

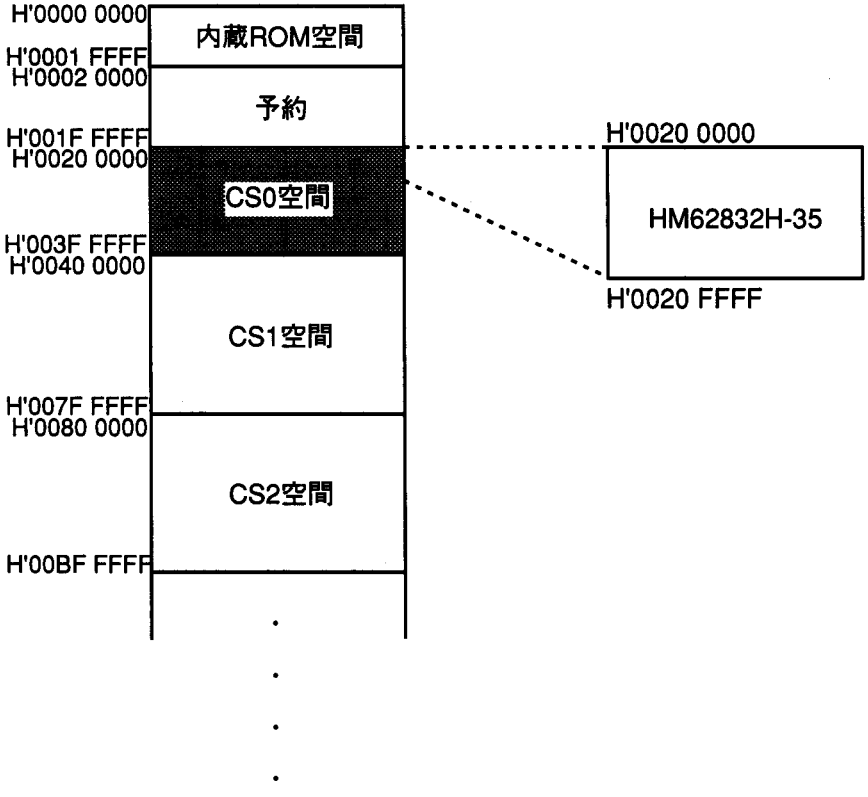


図2　メモリマップ

(3)　バスステートコントローラの各レジスタを表1のように設定します。

CS0空間の条件		レジスタ名	設定値
バスサイズ	ワードサイズ	BCR1	H'000F
アイドルサイクル	無し	BCR2	H'0000
ウェイト	無し	WCR1	H'FFF0

表1　各レジスタの設定値

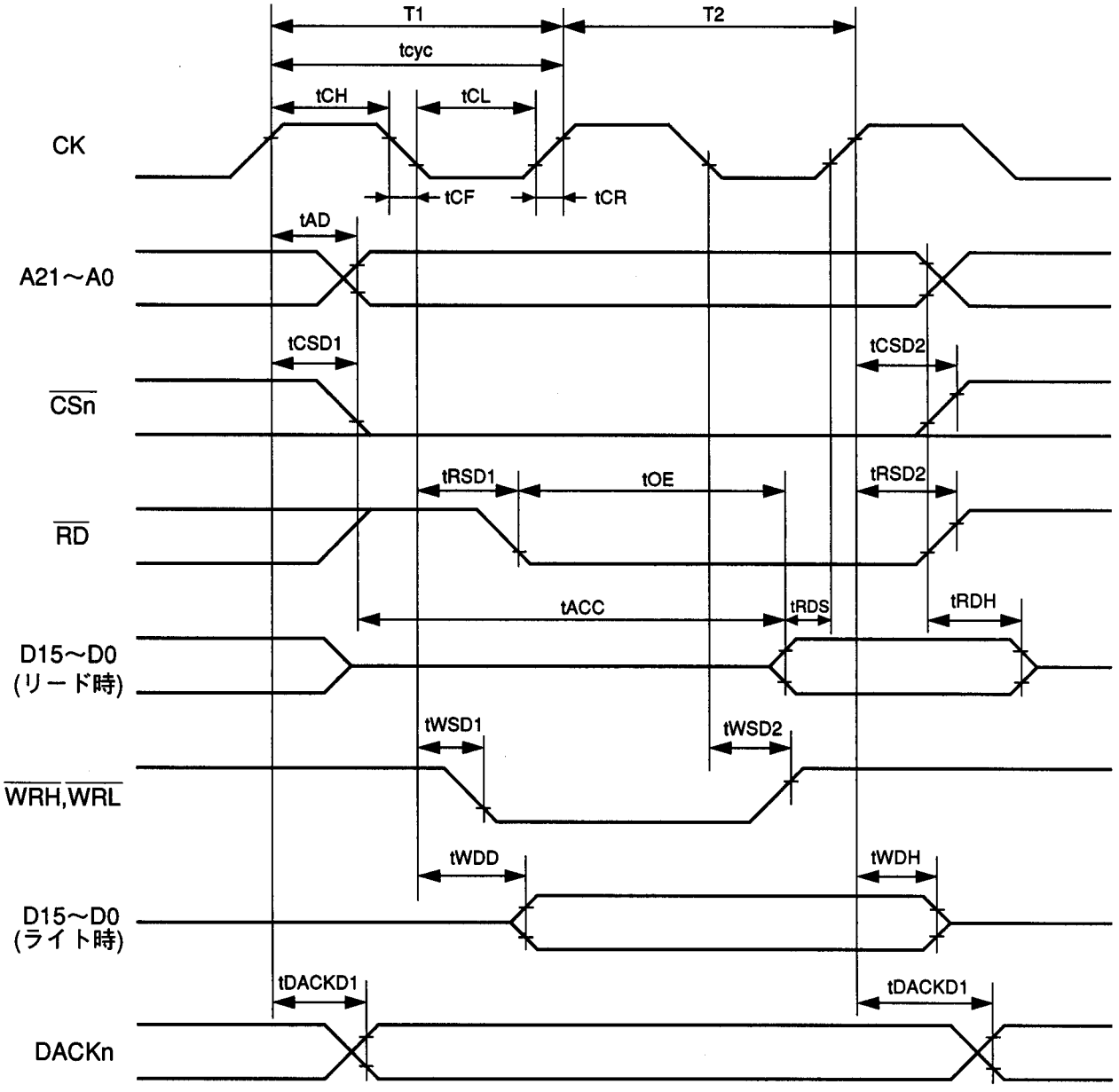
動作説明

（1）SH7050基本バスサイクル

図3にSH7050の基本バスサイクルを示します。図3に示すように2ステートで外部デバイス(HM62832H-35)とインタフェースを行います。

SH7050は、データリード時、T2の立ち上がりでD15～D0のデータをサンプリングします。データライト時は、T1の立下がりからtWDD(ライトデータ遅延時間)分遅れてからデータを出力します。

タイミングチャートの各数値については、AC特性の表バスタイミングを参照してください。



[注] tRDH：A21～A0，CSn，RDの最も早いネゲートタイミングから規定

図3 SH7050基本バスサイクル (2ステート)

動作説明

（2）データのリード／ライト

図4にデータのリード／ライトタイミングチャートを示します。SH7050とHM62832H-35を直接接続する場合、SH7050のtACC(リードデータアクセス時間)、tOE(リードストローブアクセス時間)、tRDH(リードデータホールド時間)、およびHM62832H-35のtDW(入力データセット時間)、tDH(入力データホールド時間)、tWP(ライトパルス幅)が満足されているかを確認します。
図4から各タイミングは以下のようになります。

（a）SH7050のtACCおよびtOE

tACC = tACS(max)
= 35ns ≦ 65ns (SH7050　tACC)
tOE = tOE(max)
= 15ns ≦ 40ns (SH7050　tOE)

（b）SH7050のtRDH

tRDH = tOH(max)
= 5ns ≧ 0ns (SH7050　tRDH)

（c）HM62832H-35のtDWおよびtDH

tDW = tCL(min) + tCR(max) + tCH(min) + tCF(max) + tWSD2(max) - tWDD(max)
= 20ns + 5ns + 20ns + 5ns + 25ns - 40ns
= 35ns ≧ 15ns (HM62832H-35　tDW)
tDH = tCL(min) + tCR(max) + tWDH(min) -tWSD2(max)
= 20ns + 5ns +0ns - 25ns
= 0ns ≧ 0ns (HM62832H-35　tDH)

（d）HM62832H-35のtWP

tWP = tCL(min) + tCR(max) + tCH(min) + tCF(max) + tWSD2(max) - tWSD1(max)
= 20ns + 5ns + 20ns + 5ns + 25ns - 25ns
= 50ns ≧ 20ns (HM62832H-35　tWP)

SRAMインタフェース例	MCU	SH7050(20MHz)	使用機能	モード2(内蔵ROM有効拡張モード)

動作説明

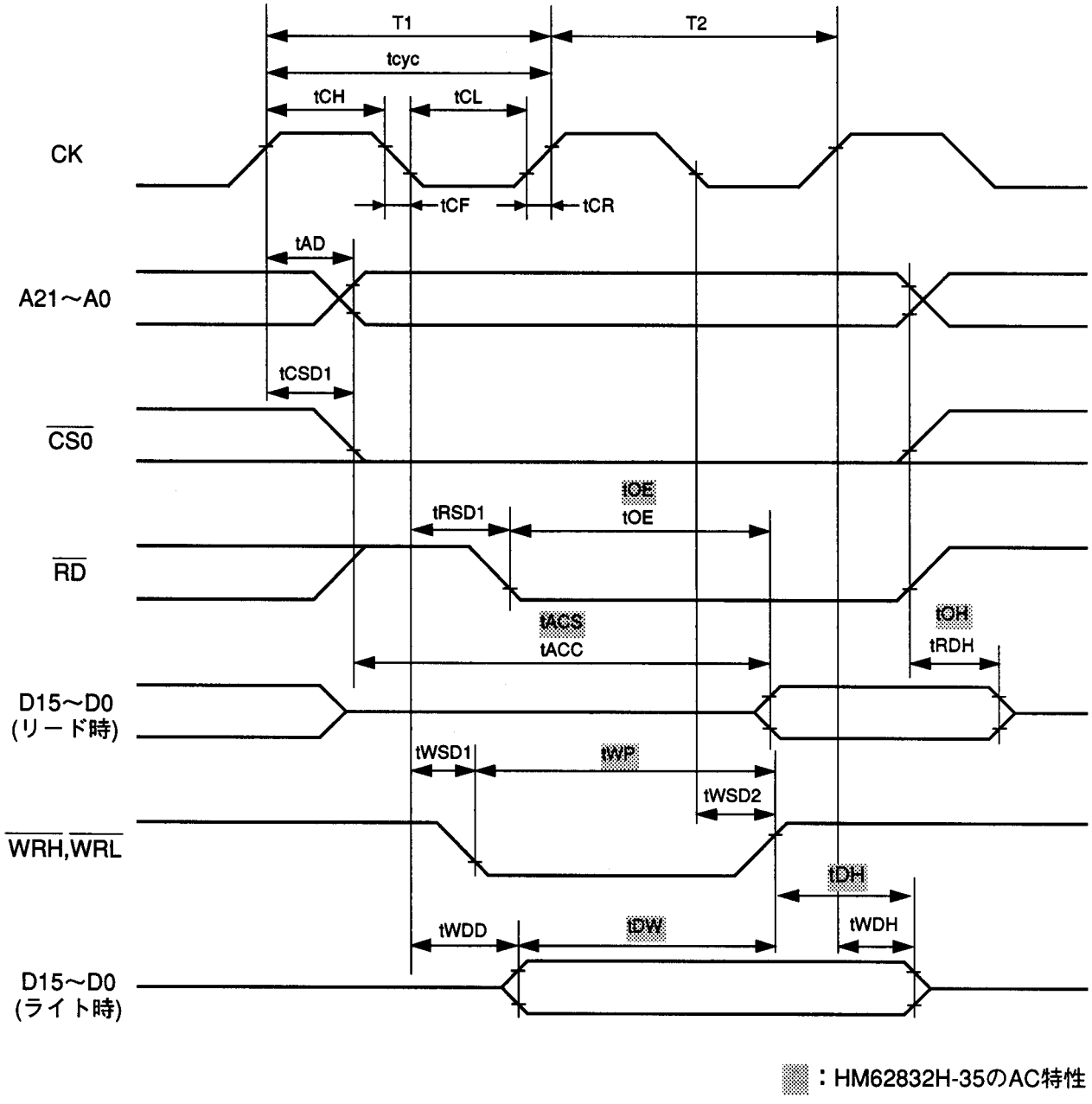


図4　リード／ライトタイミングチャート

AC特性

（1）SH7050　AC特性

(条件：Vcc=5.0V±10%、AVcc=5.0V±10%、AVcc=Vcc±10%、AVref=4.5V～AVcc、Vss=AVss=0V、Ta=-40～+85℃)

項　目	記号	min	max	単位
アドレス遅延時間	tAD	T.B.D	25	ns
$\overline{CS}$遅延時間1	tCSD1	—	30	ns
$\overline{CS}$遅延時間2	tCSD2	—	30	ns
リードストローブ遅延時間1	tRSD1	—	25	ns
リードストローブ遅延時間2	tRSD2	—	25	ns
リードデータセットアップ時間	tRDS	15	—	ns
リードデータホールド時間	tRDH	0	—	ns
ライトストローブ遅延時間1	tWSD1	T.B.D	25	ns
ライトストローブ遅延時間2	tWSD2	T.B.D	25	ns
ライトデータ遅延時間	tWDD	—	40	ns
ライトデータホールド時間	tWDH	0	—	ns
$\overline{WAIT}$セットアップ時間	tWTS	15	—	ns
$\overline{WAIT}$ホールド時間	tWTH	10	—	ns
リードデータアクセス時間	tACC	tcyc×(n+2) -35	—	ns
リードストローブからの アクセス時間	tOE	tcyc×(n+1.5) -35	—	ns
DACK遅延時間1	tDACKD1	—	30	ns

表2　バスタイミング

項　目	記号	min	max	単位
動作周波数	fOP	T.B.D	20	MHz
クロックサイクル時間	tcyc	50	T.B.D	ns
クロックローレベルパルス幅	tCL	20	—	ns
クロックハイレベルパルス幅	tCH	20	—	ns
クロック立ち上がり時間	tCR	—	5	ns
クロック立ち下がり時間	tCF	—	5	ns

表3　クロックタイミング

（2）HM62832H-35　AC特性

（a）リードサイクル

項　目	記　号	HM62832H-35		単位
		min	max	
リードサイクル時間	t_{RC}	35	—	ns
アドレスアクセス時間	t_A	—	35	ns
チップセレクトアクセス時間	t_{ACS}	—	35	ns
$\overline{CS}$出力セット時間	t_{CLZ}	5	—	ns
出力イネーブルアクセス時間	t_{OE}	—	15	ns
出力イネーブル・出力セット時間	t_{OLZ}	0	—	ns
チップディセレクト・出力フローティング時間	t_{CHZ}	0	15	ns
出力ディスエイブル・出力フローティング時間	t_{OHZ}	0	15	ns
出力保持時間	t_{OH}	5	—	ns

表3　リードサイクル AC特性

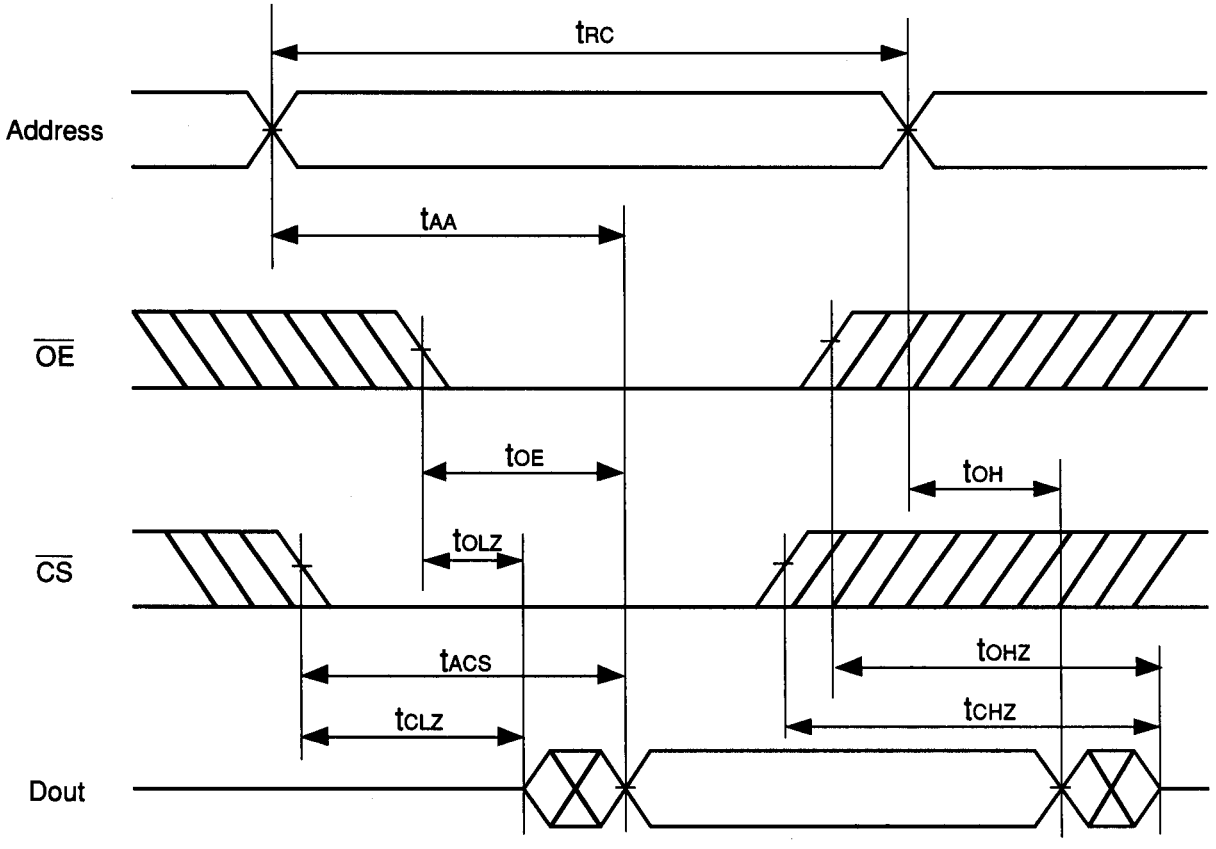


図5　リードサイクル波形

（b）ライトサイクル

項　目	記　号	HM62832H-35		単位
		min	max	
ライトサイクル時間	t_{WC}	35	—	ns
チップセレクト時間	t_{CW}	20	—	ns
アドレス有効時間	t_{AW}	30	—	ns
アドレスセットアップ時間	t_{AS}	0	—	ns
ライトパルス幅	t_{WP}	20	—	ns
アドレス保持時間	t_{WR}	0	—	ns
出力ディスエイブル・出力フローティング時間	t_{OHZ}	0	15	ns
$\overline{WE}$・出力フローティング時間	t_{WHZ}	0	15	ns
入力データセット時間	t_{DW}	15	—	ns
入力データ保持時間	t_{DH}	0	—	ns
$\overline{WE}$出力カセット時間	t_{OW}	5	—	ns

表4　ライトサイクル　AC特性

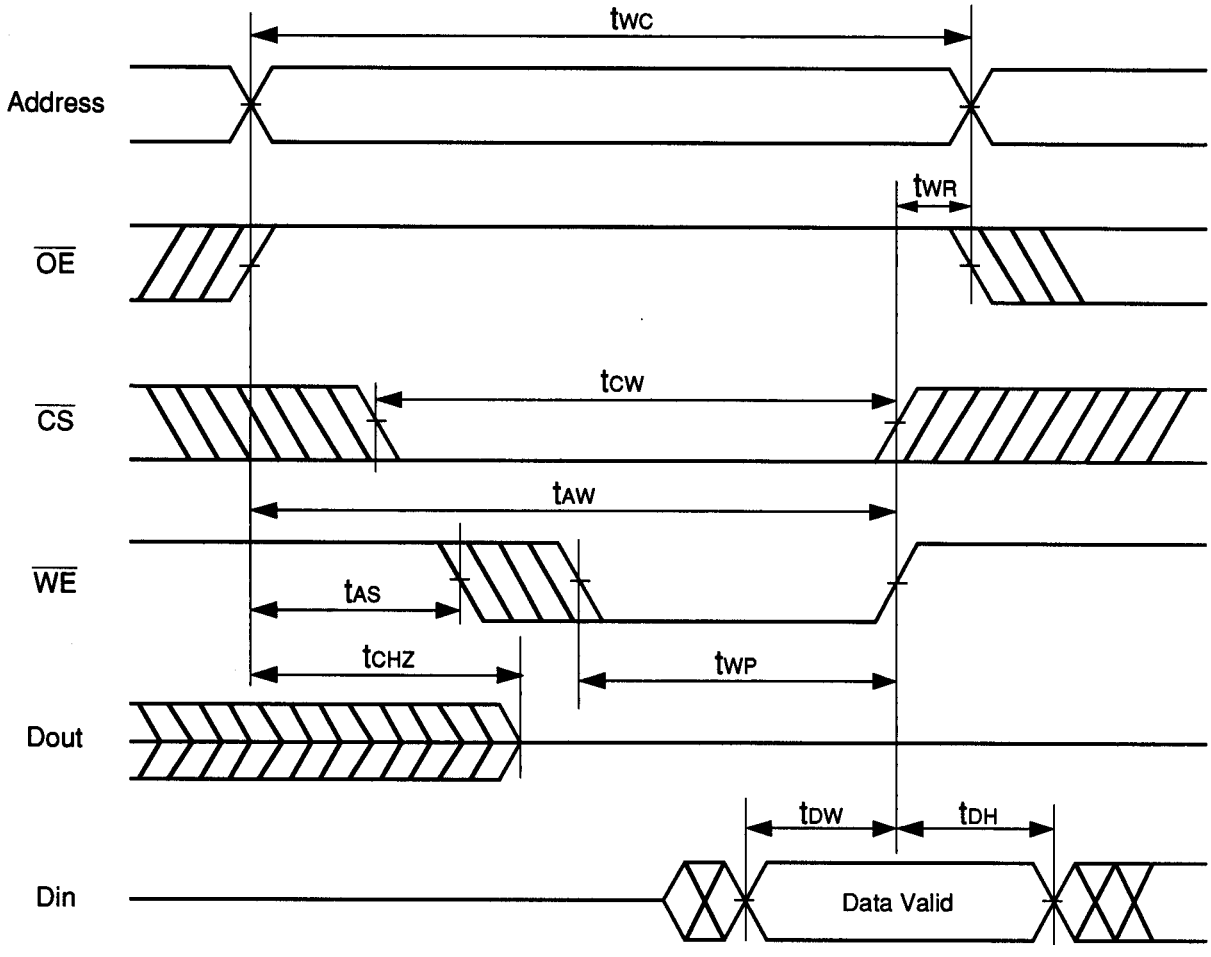
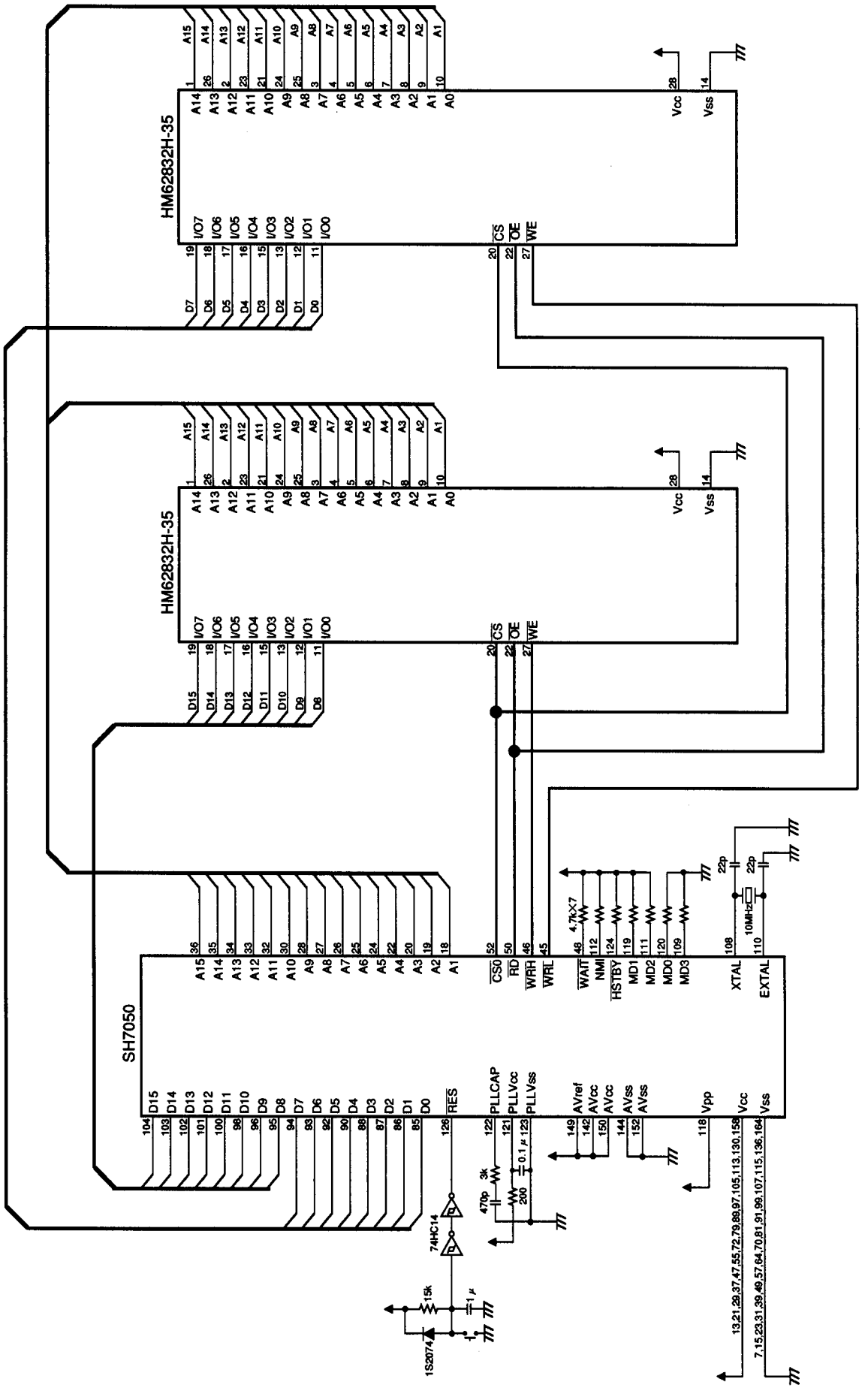


図6　ライトサイクル波形

SRAMインタフェース例	MCU	SH7050(20MHz)	使用機能	モード2(内蔵ROM有効拡張モード)

回路図



A.2 EPROMインタフェース例	MCU	SH7050(20MHz)	使用機能	モード1(内蔵ROM無効拡張モード)

仕様

（1）図1に示すように、SH7050のモード1(内蔵ROM無効拡張モード)により、1Mビット(64k×16ビット) EPROM(HN27C1024HG-85) とのインタフェースを行います。

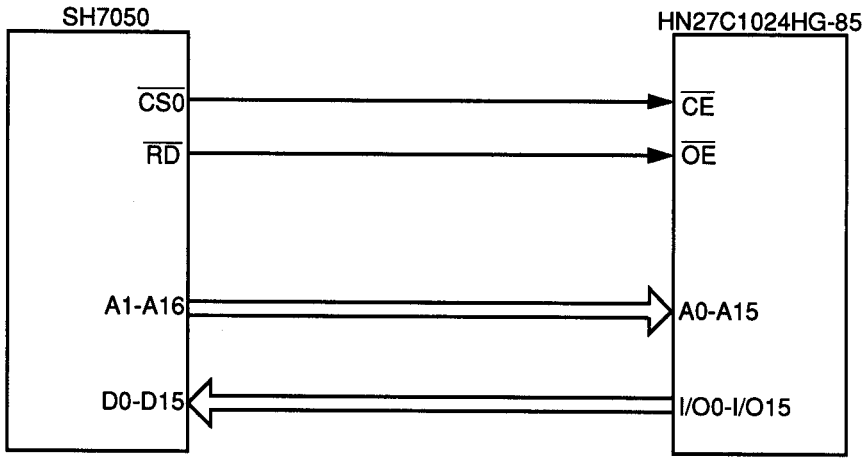


図1　SH7050およびHN27C1024HG-85 接続ブロック図

（2）HN27C1024HG-85は図2に示すように割り付けます。SH7050のバスステートコントローラを設定し、CS0空間を3ステート、ワードアクセス空間にします。

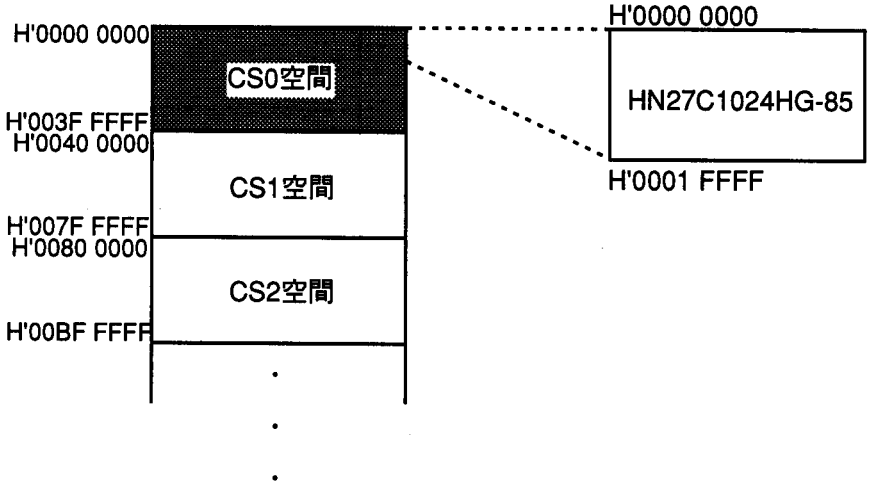


図2　メモリマップ

(3)　バスステートコントローラの各レジスタを表1のように設定します。

CS0空間の条件		レジスタ名	設定値
バスサイズ	ワードサイズ	BCR1	H'000F
アイドルサイクル	無し	BCR2	H'0000
ウェイト	1ウェイト	WCR1	H'FFF1

表1　各レジスタの設定値

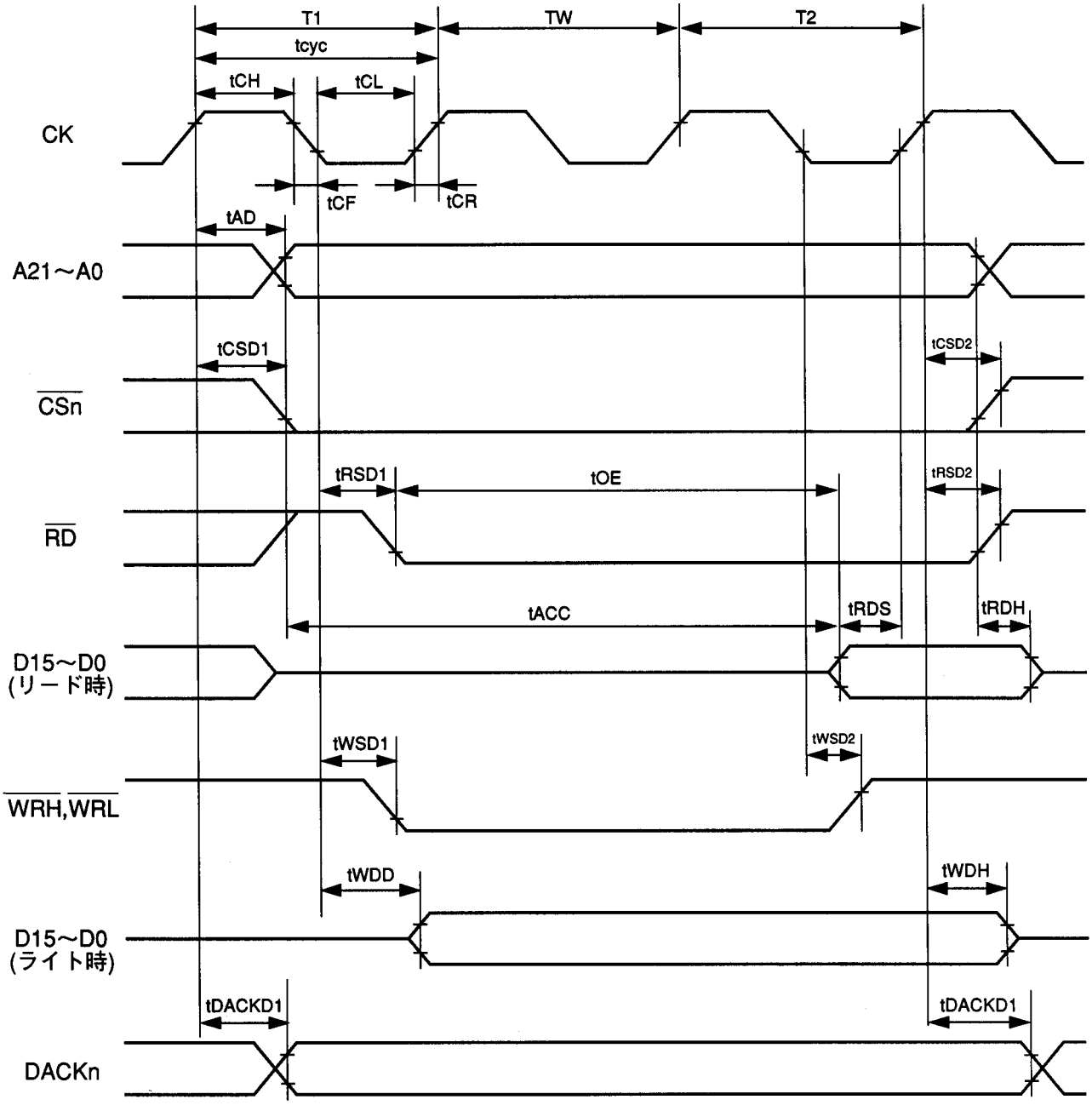
動作説明

（1）SH7050基本バスサイクル(1ソフトウェアウェイト)

図3にSH7050の1ソフトウェアウェイトの基本バスサイクルを示します。図3に示すように3ステートで外部デバイス(HN27C1024HG-85)とインタフェースを行います。

SH7050は、データリード時、T2の立ち上がりでD15～D0のデータをサンプリングします。

タイミングチャートの各数値については、AC特性の表バスタイミングを参照してください。



[注] tRDH：A21～A0，CSn，RDの最も早いネゲートタイミングから規定

図3 SH7050基本バスサイクル (1ソフトウェアウェイト)

（2）データのリード

図4にデータのリードタイミングチャートを示します。SH7050とHN27C1024HG-85を直接接続する場合、SH7050のtACC(リードデータアクセス時間)、tOE(リードストローブアクセス時間)、およびtRDH(リードデータホールド時間)が満足されているかを確認します。
図4から各タイミングは以下のようになります。

（a）SH7050のtACCおよびtOE

tACC = tCE (max)
= 85ns ≦ 115ns (SH7050　tACC)
tOE = tOE (max)
= 45ns ≦ 90ns (SH7050　tOE)

（b）SH7050のtRDH

tRDH = tOH (max)
= 0ns ≧ 0ns (SH7050　tRDH)

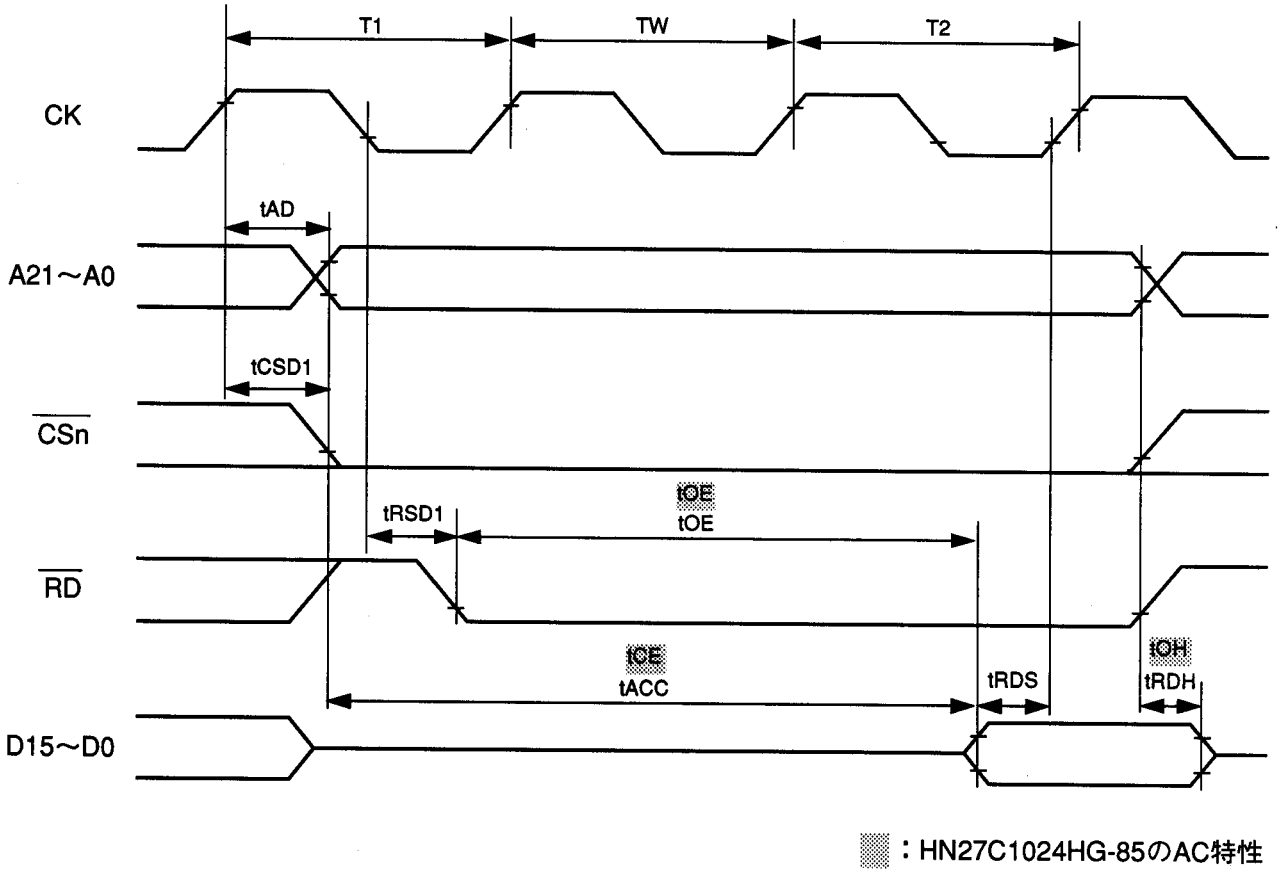


図4　リードタイミングチャート

AC特性

（1）SH7050　AC特性

(条件：Vcc=5.0V±10%、AVcc=5.0V±10%、AVcc=Vcc±10%、AVref=4.5V～AVcc、Vss=AVss=0V、Ta=-40～+85℃)

項　目	記号	min	max	単位
アドレス遅延時間	tAD	T.B.D	25	ns
$\overline{CS}$遅延時間1	tCSD1	—	30	ns
$\overline{CS}$遅延時間2	tCSD2	—	30	ns
リードストローブ遅延時間1	tRSD1	—	25	ns
リードストローブ遅延時間2	tRSD2	—	25	ns
リードデータセットアップ時間	tRDS	15	—	ns
リードデータホールド時間	tRDH	0	—	ns
ライトストローブ遅延時間1	tWSD1	T.B.D	25	ns
ライトストローブ遅延時間2	tWSD2	T.B.D	25	ns
ライトデータ遅延時間	tWDD	—	40	ns
ライトデータホールド時間	tWDH	0	—	ns
$\overline{WAIT}$セットアップ時間	tWTS	15	—	ns
$\overline{WAIT}$ホールド時間	tWTH	10	—	ns
リードデータアクセス時間	tACC	tcyc×(n+2) -35	—	ns
リードストローブからの アクセス時間	tOE	tcyc×(n+1.5) -35	—	ns
DACK遅延時間1	tDACKD1	—	30	ns

表2　バスタイミング

項　目	記号	min	max	単位
動作周波数	fOP	T.B.D	20	MHz
クロックサイクル時間	tcyc	50	T.B.D	ns
クロックローレベルパルス幅	tCL	20	—	ns
クロックハイレベルパルス幅	tCH	20	—	ns
クロック立ち上がり時間	tCR	—	5	ns
クロック立ち下がり時間	tCF	—	5	ns

表3　クロックタイミング

（2）HN27C1024HG-85　AC特性

（a）リードサイクル

項　目	記号	HN27C1024HG-85		単位
		min	max	
アクセス時間	tACC	—	85	ns
$\overline{CE}$・出力遅延時間	tCE	—	85	ns
$\overline{OE}$・出力遅延時間	tOE	—	45	ns
出力ディスエイブル遅延時間	tDF	0	30	ns
データ出力ホールド時間	tOH	0	—	ns

表3　リードサイクル AC特性

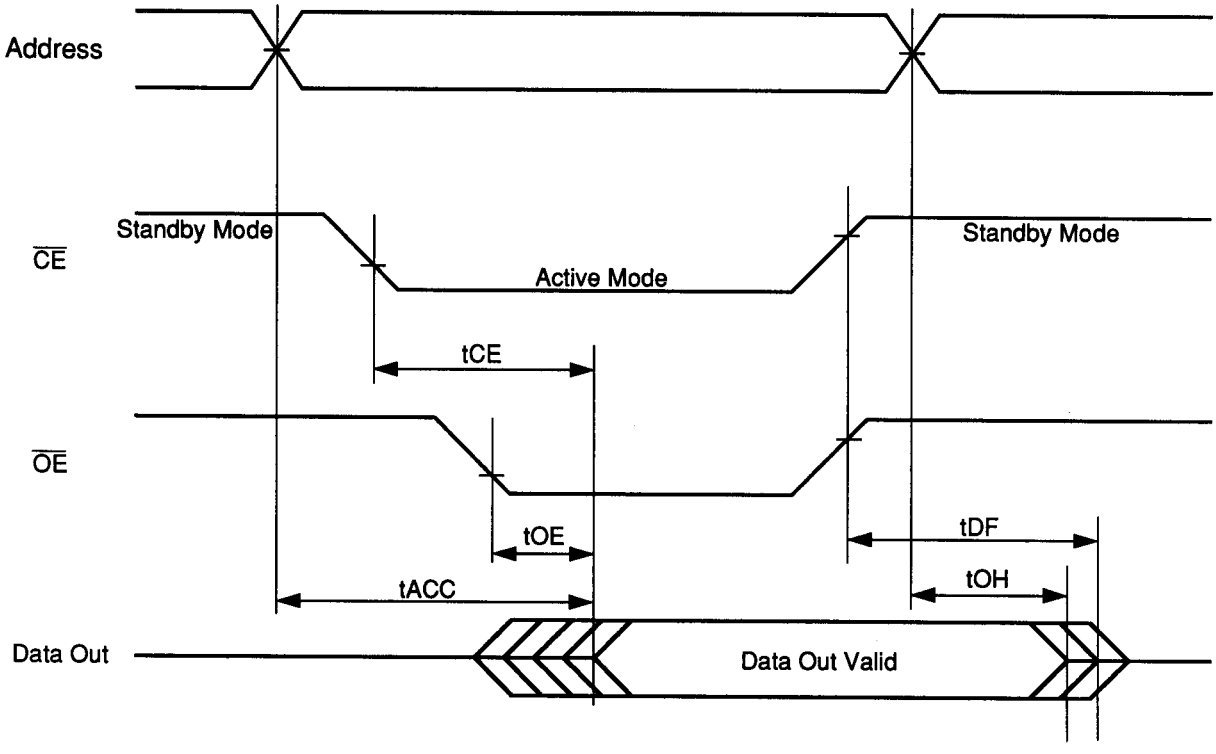
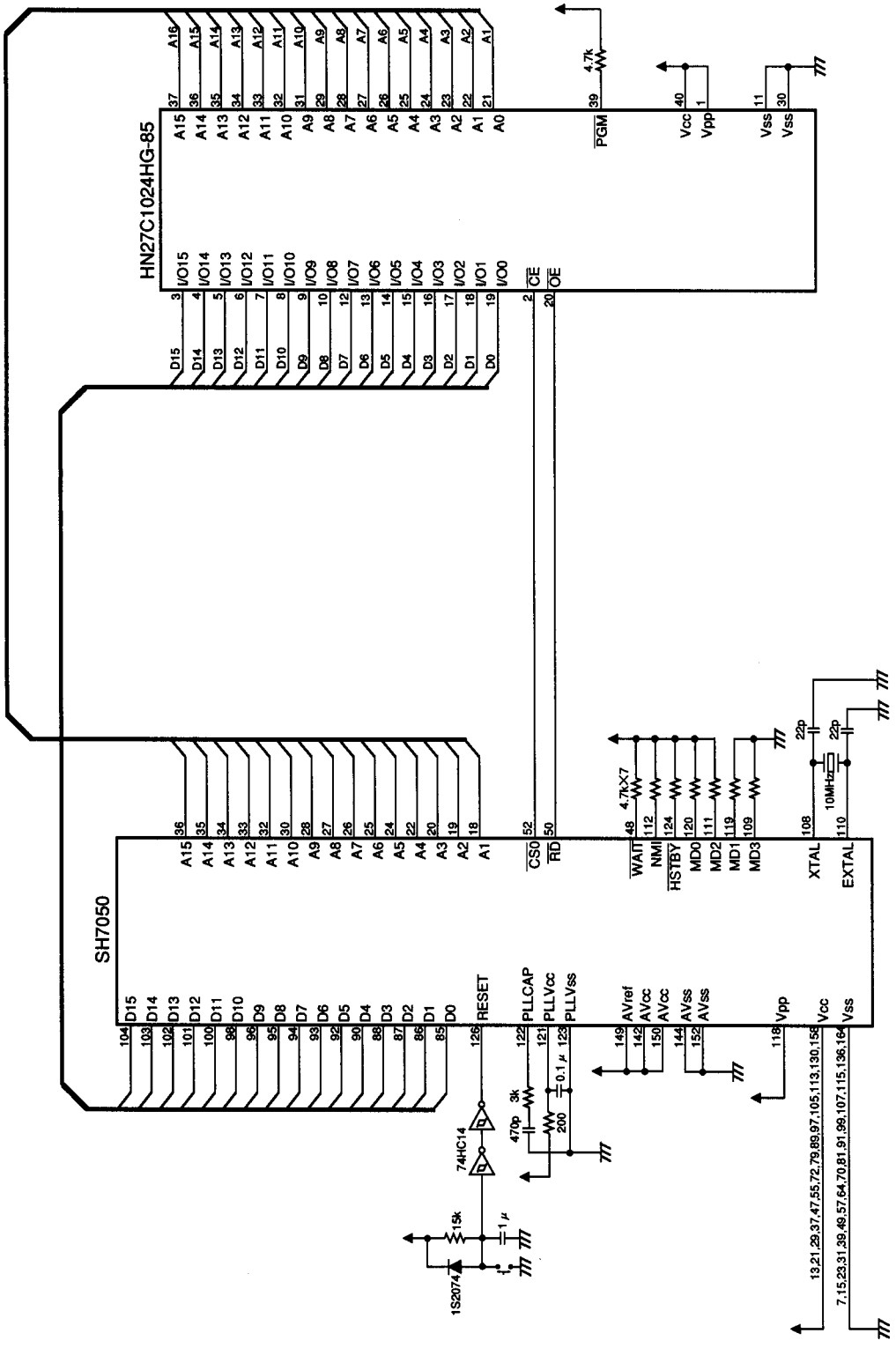


図5　リードサイクル波形

EPROMインタフェース例	MCU	SH7050(20MHz)	使用機能	モード1(内蔵ROM無効拡張モード)

回路図



HN27C1024HG-85　インタフェース回路図

SH7050 シリーズ　内蔵 I/O 編
アプリケーションノート

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-502-053