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。

Application Note

32

# SH7046シリーズ

## アプリケーションノート　内蔵周辺機能　DTC編

ルネサス32ビットRISCマイクロコンピュータ
SuperH™ RISC engineファミリ／ SH7046シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.11

# 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。

# はじめに

SH7046F、SH7148、SH7047F、SH7049 は、内部 32 ビット構成の RISC（Reduced Instruction Set Computer）タイプの命令セットを持った SH-2 CPU を核に、豊富な各種周辺機能を内蔵した高性能マイクロコンピュータです。

1 チップ上に CPU、RAM、ROM、16 ビットマルチファンクションタイマパルスユニット（MTU）、シリアルコミュニケーションインタフェース（SCI）、ポートアウトプットイネーブル（POE）、データトランスファコントローラ（DTC）、およびモータマネジメントタイマ（MMT）などの周辺機能を内蔵しており、小規模システムから大規模システムまで幅広いアプリケーションに適用できます。

本アプリケーションノートは、SH7046 シリーズの内蔵周辺機能を使用したタスク例について述べており、ユーザにてソフトウェア設計の際、ご参考として役立てていただけるようにまとめたものです。

なお、本アプリケーションノートに掲載されている各タスクのプログラムなどの動作は確認しておりますが、実際にご使用になる場合には改めて動作確認の上ご使用下さいますようお願いいたします。

# 目次

# 1. SH7046 シリーズ アプリケーションノート使用の手引き

## 1.1 本アプリケーションノートの構成

本アプリケーションノートは、図 1.1 に示すように 2 部構成になります。


図 1.1　アプリケーションノートの構成

(1)　SH7046 シリーズ　アプリケーションノート使用の手引き

SH7046 シリーズ　アプリケーションノートの使用法について説明します。

(2)　内蔵周辺機能　DTC 編

SH7046 シリーズの内蔵周辺機能のうち、主に DTC の使用法をタスク例をもとに説明します。

## 1.2 構成

図 1.2 に示す構成で内蔵周辺機能の使用法について説明します。

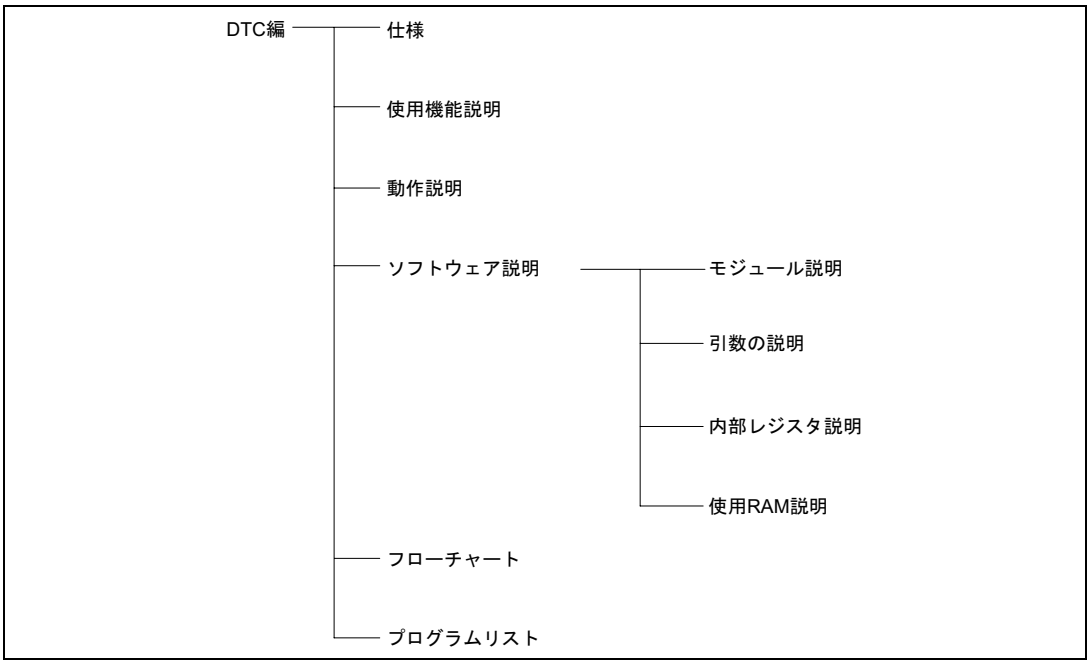



図 1.2 構成

(1) 仕様
タスク例のシステム仕様について説明します。

(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て説明します。

(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など）について説明します。

(d) 使用RAM説明
モジュールで使用するRAMのラベル名および機能について説明します。

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

(6) プログラムリスト
タスク例を実行するソフトウェアのプログラムリストを示します。

# 2.　内蔵周辺機能　DTC 編

## 2.1　DTC ノーマルモードによるデータ転送（CMT、DTC）

| DTC ノーマルモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

（1）　図2.1に示すように、CMT（Compare Match Timer）のコンペアマッチ割り込みでDTC（Data Transfer Controller）を起動させ、内蔵RAMから内蔵RAMへのデータ転送を行います。

（2）　DTCデータ転送では、ノーマルモード使用し、図2.2に示すように、3バイトの転送を行います。

（3）　DTCの転送条件を表2.1に示します。

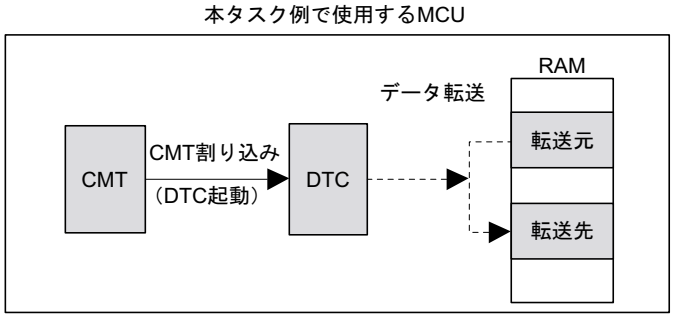



図 2.1　DTC を用いたデータ転送

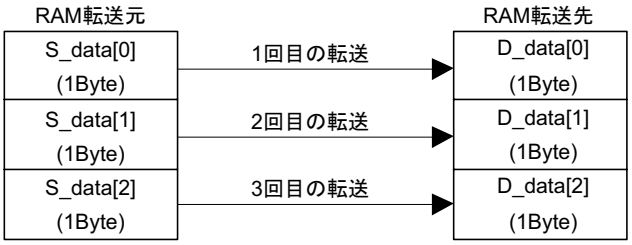



図 2.2　DTC ノーマルモードでのデータ転送

表 2.1　DTC 転送条件

| 条件 | 内容 |
|---|---|
| 転送モード | ノーマルモード |
| 転送回数 | 3 回 |
| 転送データサイズ | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM |
| 転送先 | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント |
| 転送先アドレス | 転送後に転送先アドレスをインクリメント |
| 起動要因 | CMT ch0 のコンペアマッチ割り込み（CMI0）で起動 |
| 割り込み処理 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

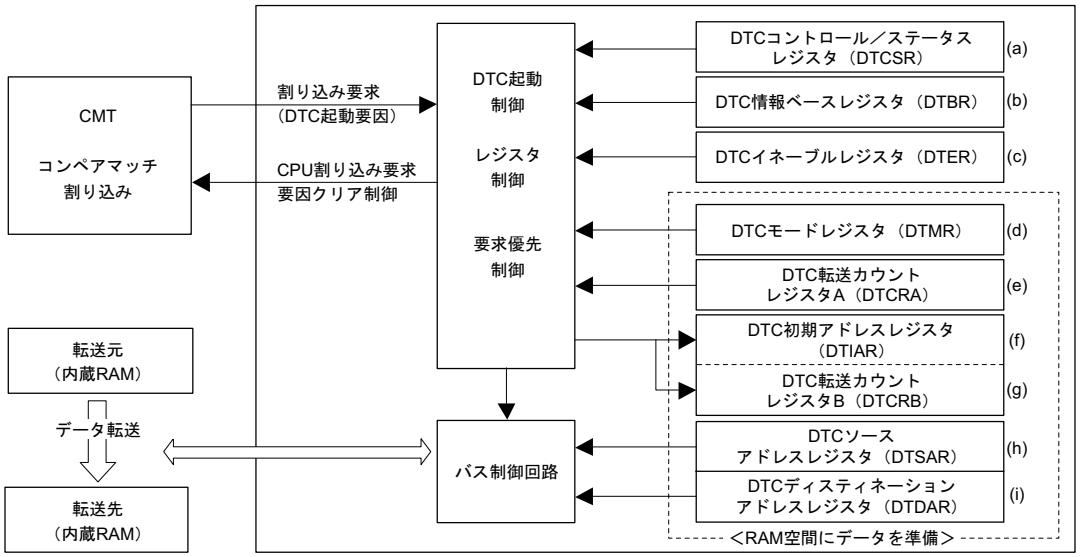
| DTC ノーマルモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

**使用機能説明**

(a) **図2.3**にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ノーマルモードを使用してデータ転送を行います。DTC起動要因をCMTのコンペアマッチ割り込みとして、内蔵RAMから内蔵RAMへのデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.3　DTC ブロック図

使用機能説明

(b)　図2.4にCMT ch0のブロック図を示します。本タスクでは、CMT ch0のコンペアマッチ割り込みを起動要因としてDTCデータ転送を行います。以下にブロック図について説明します。

- コンペアマッチタイマスタートレジスタ（CMSTR）は、チャネル 0、1 のカウンタ（CMCNT）を動作させるか、停止させるかの設定を行う、16 ビットのレジスタです。
- 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（CMCSR_0）は、コンペアマッチ発生の表示、割り込みの許可／禁止の設定、カウントアップに用いられるクロックの選択をする 16 ビットのレジスタです。
- コンペアマッチタイマカウンタ_0（CMCNT_0）は、割り込み要求を発生させるためのアップカウンタとして使用する 16 ビットのレジスタです。
- コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）は、CMCNT とのコンペアマッチ周期を設定する 16 ビットのレジスタです。

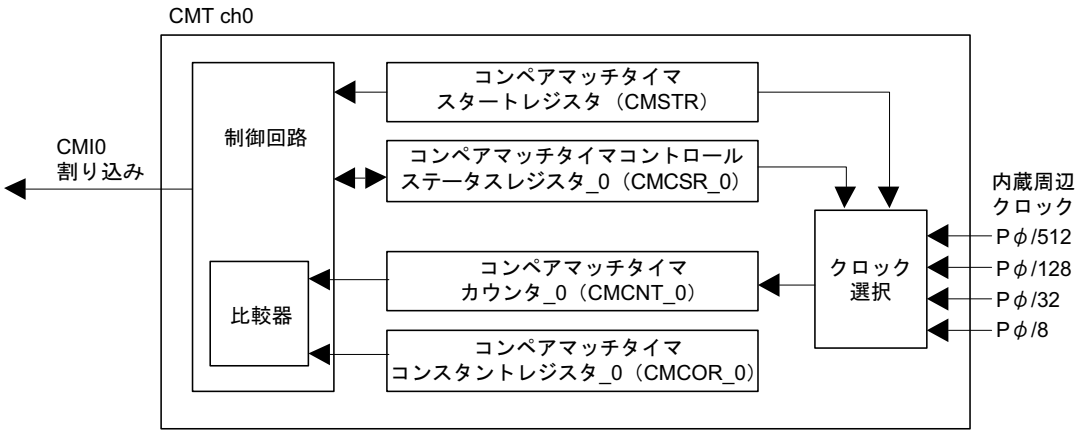



図 2.4　CMT ブロック図

(4)　表2.2に本タスク例の機能割り付けを示します。

表 2.2　機能割り付け

| 機能 | 分類 | 機能割り付け |
| --- | --- | --- |
| DTMR | DTC | DTC をノーマルモードに設定 |
| DTCRA | DTC | 転送回数の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | CMT ch0 の CMI 割り込みで DTC の起動を許可 |
| CMSTR | CMT | CMT カウントのスタート |
| CMCSR_0 | CMT ch0 | カウントクロックの選択、割り込みの制御 |
| CMCNT_0 | CMT ch0 | カウンタ |
| CMCOR_0 | CMT ch0 | 周期の設定 |

動作説明

(1) 図2.5に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理によりDTCによる内蔵RAMから内蔵RAMへのデータ転送を行います。

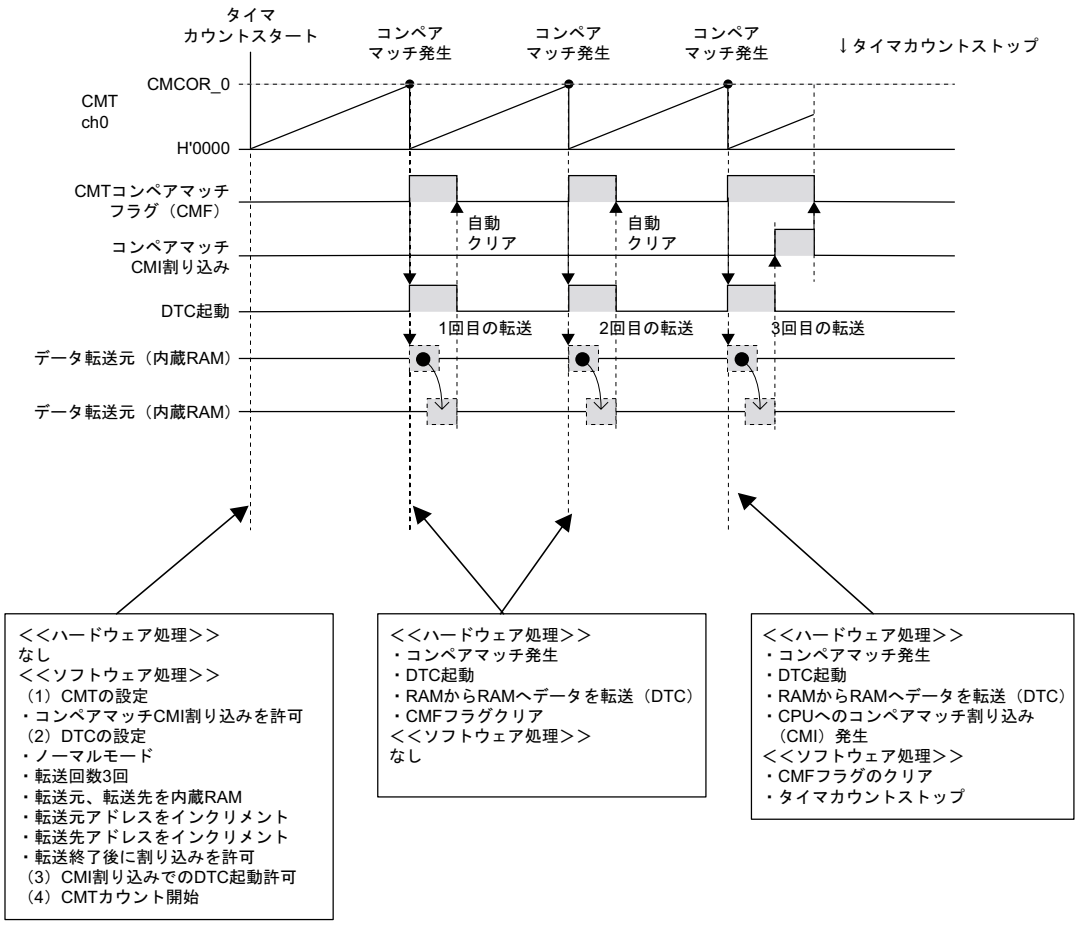



図 2.5　動作原理

動作説明

（2） 図2.6にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込みデータ転送を行います。

本タスクでは、CMTのコンペアマッチ割り込みがDTCが起動要因となります。
表2.7にノーマル転送モード時のレジスタ情報の構成を示します。

表 2.3　DTC レジスタ情報一覧（ノーマルモード）

| 設定アドレス | レジスタ名 | データ長 |
| --- | --- | --- |
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

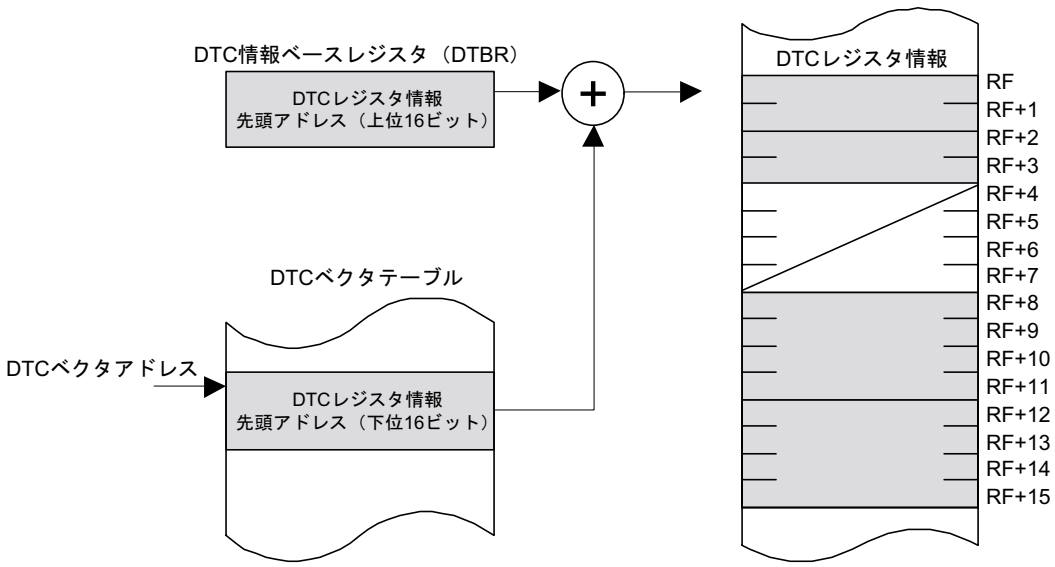



図 2.6　DTC ベクタアドレスと転送情報との対応

ソフトウェア説明

(1) モジュール説明

**表**2.4に、本タスク例のモジュールを示します。

表 2.4 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | CMT タイマの設定、DTC の初期設定、タイマスタート |
| CMI0 割り込み | cmt0_cmi0_dtc | CMT ch0 コンペアマッチ割り込み（CMI0）。DTC 指定回数転送終了時に割り込み発生 |

(2) 引数の説明

**表**2.5に、本タスク例で使用する引数を示します。

表 2.5 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| S_data [0]～[2] | DTC 転送元の転送データ格納 | メインルーチン | 1 バイト | 出力 |
| D_data [0]～[2] | DTC 転送先の転送データ格納 | メインルーチン | 1 バイト | 入力 |

(3) 使用内部レジスタ説明

**表**2.6、**表**2.7に、本タスク例の使用する内部レジスタを示します。

表 2.6 使用内蔵レジスタ説明（1）

| レジスタ名 | | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | ビット | | ビット | |
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1、<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ値を設定 | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| P_STBY.MSTCR2 | MSTP12 | モジュールスタンバイコントロールレジスタ 2<br>CMT モジュールスタンバイ制御ビット<br>:MSTP12=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 12 | 0 |
| P_INTC.IPRG | CMT0 | インタラプトプライオリティレジスタ G（IPRG）<br>CMT0 の CMI0 割り込み優先レベルの設定<br>:CMT0=b'1010(10)のとき、CMI0 割り込みは、優先レベル 10 に設定 | H'FFFF8354<br>ビット<br>7～4 | 10 |
| P_CMT.CMSTR | | コンペアマッチタイマスタートレジスタ（CMTSR）<br>CMCNT の動作／停止を選択する 16 ビットレジスタ | H'FFFF83D0 | H'0001 |
| | STR1 | カウンタスタート 1<br>：STR1=b'0 のとき、TCNT_1 のカウント動作は停止 | ビット 1 | |
| | STR0 | カウンタスタート 0<br>：STR0=b'1 のとき、TCNT_0 のカウント動作 | ビット 0 | |

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_CMT.CMCSR_0 | | 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（CMCSR_0）<br>コンペアマッチ発生の表示、割り込み設定、タイマのクロック設定 | H'FFFF83D2 | H'0043 |
| | CMF | コンペアマッチフラグ<br>：CMCNT と CMCOR の値が一致したとき CMF=1 となる | ビット 7 | |
| | CMIE | コンペアマッチ割り込みイネーブル<br>：CMIE=1 のとき、コンペアマッチ割り込み（CMI）を許可 | ビット 6 | |
| | CKS1<br>CKS0 | CMCNT のカウンタクロックの選択<br>：CKS[1:0]=b'11 とき、内部クロック：Pφ／512 でカウント | ビット 1<br>ビット 0 | |
| P_CMT.CMCNT_0 | | コンペアマッチタイマカウンタ_0（CMCNT_0）<br>割り込み要求を発生させるためのアップカウンタ、16 ビットレジスタ | H'FFFF83D4 | H'0000 |
| P_CMT.CMCOR_0 | | コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）<br>CMCNT とのコンペアマッチ周期を設定、16 ビットレジスタ<br>CMCOR_0=H'1e84 のとき、100ms 周期でコンペアマッチ<br>（Pφ/512 カウント、Pφ=40MHz） | H'FFFF83D6 | H'1e84 |

表 2.7　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| DTC_N.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。 | 内蔵 RAM に配置 | H'a000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマルモード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、デスティネーション側がブロック領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
| --- | --- | --- | --- | --- |
| DTC_N.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_N.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S_data[0]; |
| DTC_N.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &D_data[0]; |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTED | CMI0 | DTC イネーブルレジスタ D（DTED）<br>1 をセットすると対応する割り込み要因が DTC 起動要因として選択<br>：CMI0(DTED5)=b'1 のとき、CMT0 の CMI0 割り込みが起動要因 | H'FFFF8703<br>ビット 5 | 1 |

（4） 使用RAM説明

表2.8に、本タスクで使用するRAMの説明を示します。

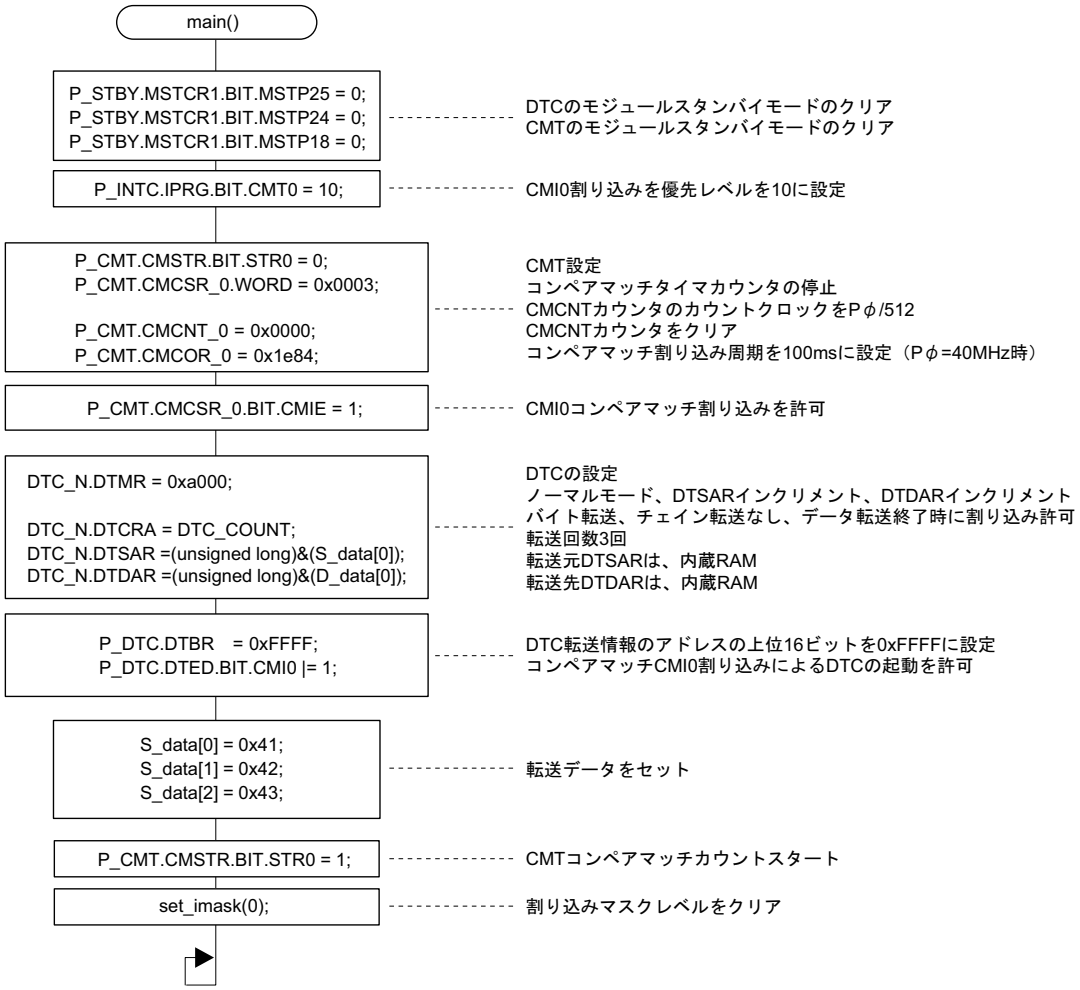
表 2.8　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
| --- | --- | --- | --- |
| S_data | DTC 転送データの格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |
| D_data | DTC データ転送後のデータ格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |

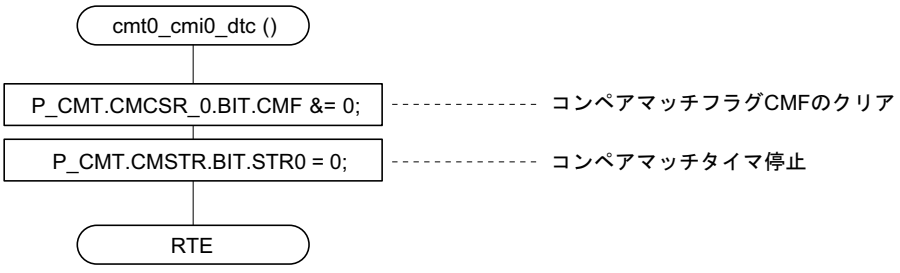
| DTC ノーマルモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

フローチャート

(a)　メイン処理




(b)　コンペアマッチ割り込み処理



プログラムリスト

```
/******************************************************/
/* SH7046F Series -SH7047- Application Note           */
/* Data transfer Controller(DTC)                      */
/*     Normal  mode                                   */
/* Function                                           */
/*  :Data transfer Controller(DTC)                    */
/*  :Compare Match Timer(CMT ch0)                     */
/*                                                    */
/* External input clock    :10MHz                     */
/* Internal CPU clock     :40MHz                      */
/* Internal peripheral clock :40MHz                   */
/*                                                    */
/* Written  :  2002/3/1  Rev.1.0                      */
/******************************************************/


#include "iodefine_7047v13.1.h"
#include <machine.h>



/*------------ Symbol Defnition ----------------------------------------------*/
struct st_dtc_normal{        /* DTC Normal  Mode information */
    unsigned short DTMR;     /* DTC Mode Register            */
    unsigned short DTCRA;    /* Transfer counter             */
    unsigned short dummy1;   /* Reserved                     */
    unsigned short dummy2;   /* Reserved                     */
    unsigned long DTSAR;     /* source address register      */
    unsigned long DTDAR;     /* destination address register */
};

#define DTC_COUNT 3                /* DTC Transmit count    */
#define DTC_N (*(volatile struct st_dtc_normal*)0xFFFFE000)  /* DTC information
address */



/*------------ Function Definition -------------------------------------------*/
```

プログラムリスト

```
void main(void);
void cmt0_cmi0_dtc(void);


/*------------ RAM allocation Definition ------------------------------------*/
unsigned char S_data[DTC_COUNT];      /* source buffer memory           */
unsigned char D_data[DTC_COUNT];      /* destination buffer memory      */



/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
    /* Set standby mode   */
    P_STBY.MSTCR1.BIT.MSTP25 = 0;       /* Disable DTC standby mode       */
    P_STBY.MSTCR1.BIT.MSTP24 = 0;       /* Disable DTC standby mode       */
    P_STBY.MSTCR2.BIT.MSTP12 = 0;       /* Disable CMT standby mode       */

    /* Set interrupt priority level (0 to 15)  */
    P_INTC.IPRG.BIT.CMT0 = 10;        /* CMT0 CMI0 interrupt level 10     */
    /* Initialize CMT0 for Interval timer                */
    P_CMT.CMSTR.BIT.STR0 = 0;    /* timer count stop          */
    P_CMT.CMCSR_0.WORD = 0x0003;
        /* CMF=0;         clear compare match flag            */
        /* CMIE=0;        compare match interrupt disable        */
        /* CKS[1:0]=b'11; clock = peripheral clock(Pφ)/512      */
    P_CMT.CMCNT_0 = 0x0000;      /* timer counter clear             */
    P_CMT.CMCOR_0 = 0x1e84;      /* 100ms@Pφ=40MHz              */
    P_CMT.CMCSR_0.BIT.CMIE = 1;  /* compare match interrupt enable      */
```

プログラムリスト

```
    /* DTC information     */
    DTC_N.DTMR = 0xa000;         /* */
         /* SM[1:0]=b'10;    DTSAR is incremented            */
         /* DM[1:0]=b'10;    DTDAR is incremented            */
         /* MD[1:0]=b'00;    Normal transfer mode            */
         /* SZ[1:0]=b'00;    byte-size transfer              */
         /* DTS=0;           destination is block area (not used)     */
         /* CHNE=0;          Chain transfer is disable          */
         /* DISEL=0;         Interrupt->transfer ends           */
         /* NMIM=0;          NMI->Terminate DTC transfer        */
    DTC_N.DTCRA = DTC_COUNT;        /* DTC transfer Count         */
    DTC_N.DTSAR =(unsigned long)&(S_data[0]);  /* set source address         */
    DTC_N.DTDAR =(unsigned long)&(D_data[0]);  /* set destination address    */

    P_DTC.DTBR = 0xFFFF;           /* DTC information base register     */
    /* DTC transmit enable      */
    P_DTC.DTED.BIT.CMI0 |= 1;      /* interrupt sources CMT ch0(CMI0)     */



    /*  set transmit data  */
    S_data[0] = 0x41;
    S_data[1] = 0x42;
    S_data[2] = 0x43;

    P_CMT.CMSTR.BIT.STR0 = 1;    /* CMT0 timer count start            */

    set_imask(0);                /* clear interrupt mask level         */
    while(1);
}
```

プログラムリスト

```
/************************************************************/
/* CMT0 Interrupt                                           */
/* Interval interrupt                                       */
/************************************************************/
#pragma interrupt(cmt0_cmi0_dtc)
void cmt0_cmi0_dtc(void)
{

    P_CMT.CMCSR_0.BIT.CMF &= 0;     /* Clear CMT0 compare match flag         */

    P_CMT.CMSTR.BIT.STR0 = 0;    /* CMT0 timer count stop           */

}
```

## 2.2 DTC リピートモードによるデータ転送（CMT、DTC）

| DTC リピートモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

（1） 図2.7に示すように、CMT（Compare Match Timer）のコンペアマッチ割り込みでDTC（Data Transfer Controller）を起動させ、内蔵RAMから内蔵RAMへのデータ転送を行います。
（2） DTCデータ転送では、リピートモード使用し、図2.8に示すように、転送元の3バイトデータを内蔵RAMの固定エリアに繰り返し転送を行います。
（3） DTCの転送条件を表2.9に示します。

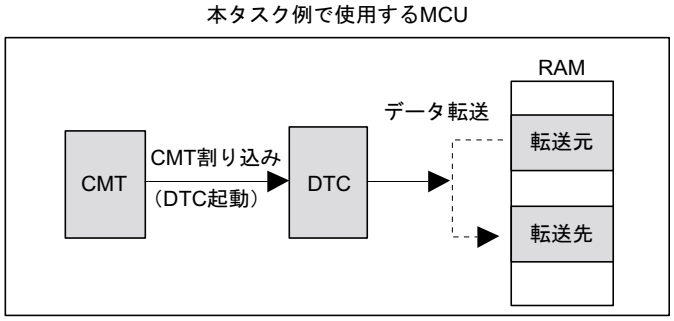
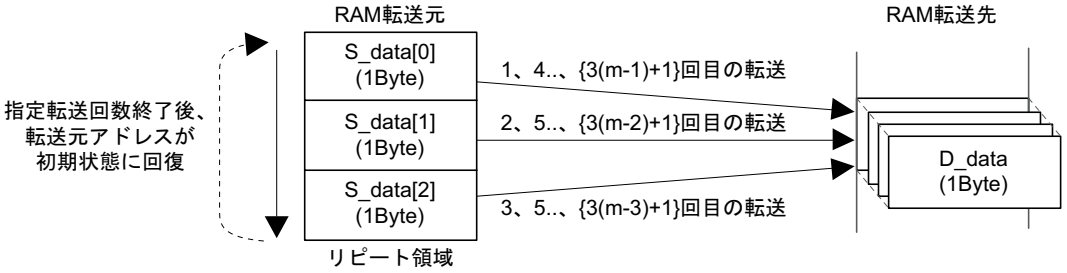



図 2.7　DTC を用いたデータ転送




※：mはリピート回数（m=1,2...）

図 2.8　DTC リピートモードでのデータ転送

表 2.9　DTC 転送条件

| 条件 | 内容 |
|---|---|
| 転送モード | リピートモード、ソース側（転送元）がリピート領域 |
| 転送回数 | 3 回 |
| 転送データサイズ | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM（リピート領域） |
| 転送先 | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント |
| 転送先アドレス | 転送先アドレスを固定 |
| 起動要因 | CMT ch0 のコンペアマッチ割り込み（CMI0）で起動 |
| 割り込み処理 | DTC 転送のたびに CPU に対して割り込みを許可 |

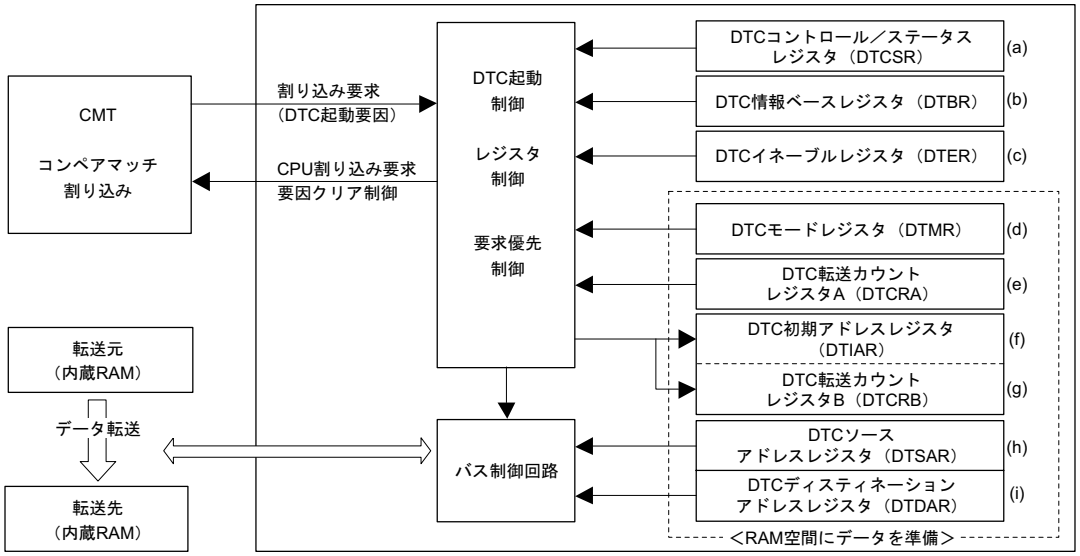
| DTC リピートモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

**使用機能説明**

(a) 図2.9にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ノーマルモードを使用してデータ転送を行います。DTC起動要因をCMTのコンペアマッチ割り込みとして、内蔵RAMから内蔵RAMへのデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.9　DTC ブロック図

使用機能説明

(b)　図2.10にCMT ch0のブロック図を示します。本タスクでは、CMT ch0のコンペアマッチ割り込みを起動要因としてDTCデータ転送を行います。以下にブロック図について説明します。

- コンペアマッチタイマスタートレジスタ（CMSTR）は、チャネル 0、1 のカウンタ（CMCNT）を動作させるか、停止させるかの設定を行います。16 ビットのレジスタです。
- 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（CMCSR_0）は、コンペアマッチ発生の表示、割り込みの許可／禁止の設定、カウントアップに用いられるクロックの選択をする 16 ビットのレジスタです。
- コンペアマッチタイマカウンタ_0（CMCNT_0）は、割り込み要求を発生させるためのアップカウンタとして使用する 16 ビットのレジスタです。
- コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）は、CMCNT とのコンペアマッチ周期を設定する 16 ビットのレジスタです。

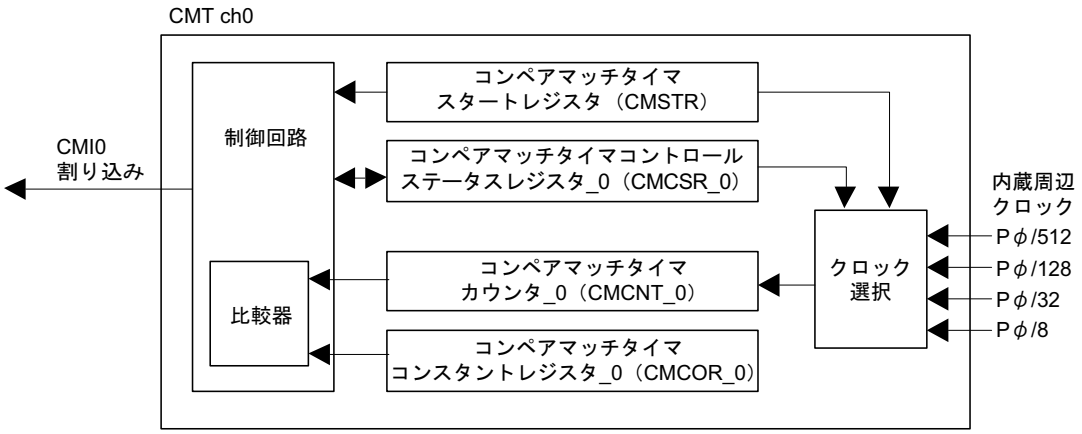



図 2.10　CMT ブロック図

(4)　表2.10に本タスク例の機能割り付けを示します。

表 2.10　機能割り付け

| 機能 | 分類 | 機能割り付け |
|---|---|---|
| DTMR | DTC | DTC をリピートモードに設定、ソース側（転送元）をリピート領域 |
| DTCRA | DTC | 転送回数の設定 |
| DTIAR | DTC | 転送元（リピート領域）の初期アドレスを設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | CMT ch0 の CMI 割り込みで DTC の起動を許可 |
| CMSTR | CMT | CMT カウントのスタート |
| CMCSR_0 | CMT ch0 | カウントクロックの選択、割り込みの制御 |
| CMCNT_0 | CMT ch0 | カウンタ |
| CMCOR_0 | CMT ch0 | 周期の設定 |

## 動作説明

(1)　図2.11に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理によりDTCによる内蔵RAMから内蔵RAMへのデータ転送を行います。

本タスクでは、DTCリピートモードを使います。

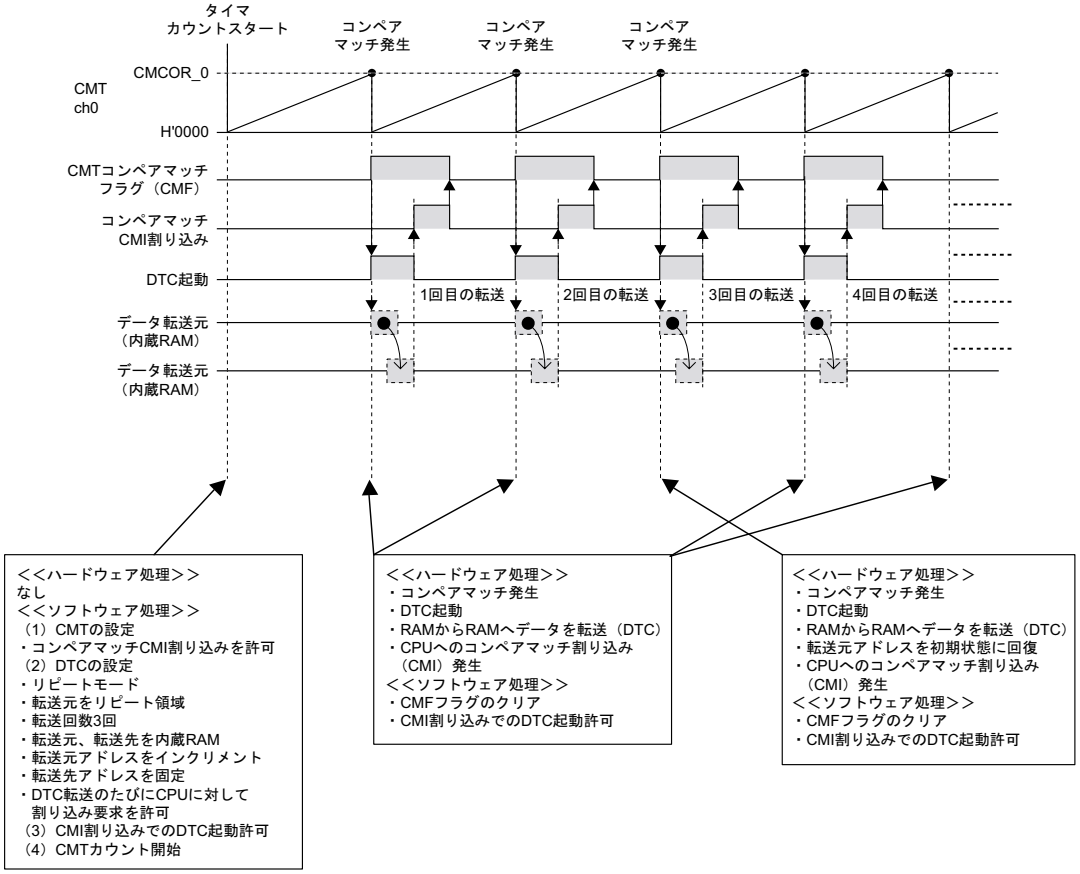



図 2.11　動作原理

動作説明

（2） 図2.12にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込みデータ転送を行います。

本タスクでは、CMTのコンペアマッチ割り込みがDTCが起動要因となります。
表2.11にリピートモード時のレジスタ情報の構成を示します。

表 2.11　DTC レジスタ情報一覧（リピートモード）

| 設定アドレス | レジスタ名 | データ長 |
| --- | --- | --- |
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ AH（DTCRAH） | バイト（1Byte） |
| RF+3 | DTC 転送カウントレジスタ AL（DTCRAL） | バイト（1Byte） |
| RF+4 | DTC 初期アドレスレジスタ（DTIAR） | ロングワード（4Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

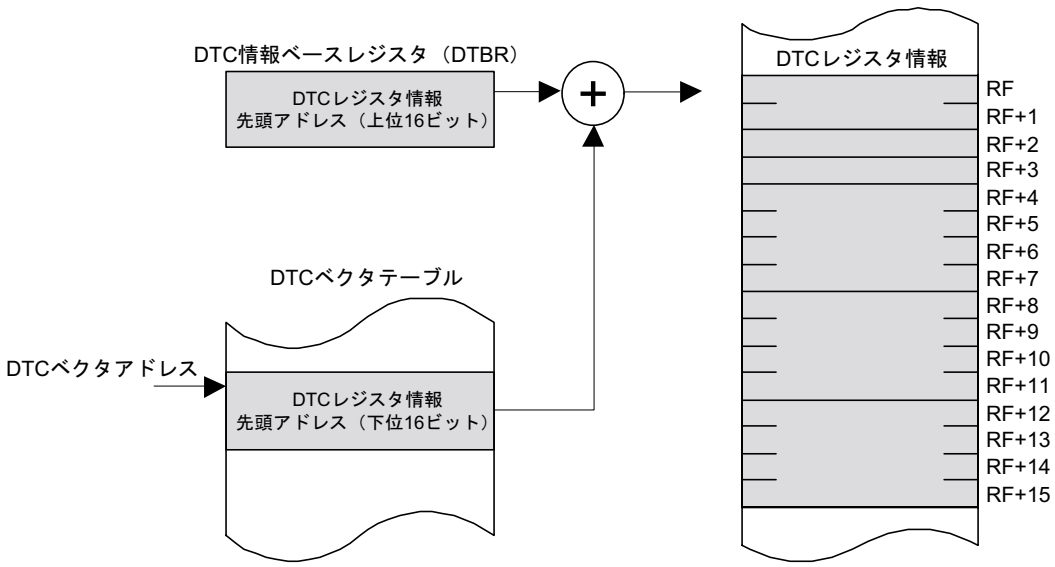



図 2.12　DTC ベクタアドレスと転送情報との対応

| DTC リピートモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

**表2.12**に、本タスク例のモジュールを示します。

表 2.12 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | CMT タイマの設定、DTC の初期設定、タイマスタート |
| CMI0 割り込み | cmt0_cmi0_dtc | CMT ch0 コンペアマッチ割り込み（CMI0）。DTC 転送のたびに割り込み発生 |

（2） 引数の説明

**表2.13**に、本タスク例で使用する引数を示します。

表 2.13 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| S_data[0]～[2] | DTC 転送元の転送データ格納 | メインルーチン | 1 バイト | 出力 |
| D_data | DTC 転送先の転送データ格納 | メインルーチン | 1 バイト | 入力 |

（3） 使用内部レジスタ説明

**表2.14**、**表2.15**に、本タスク例の使用する内部レジスタを示します。

表 2.14 使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ値を設定 | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| P_STBY.MSTCR2 | MSTP12 | モジュールスタンバイコントロールレジスタ 2<br>CMT モジュールスタンバイ制御ビット<br>:MSTP12=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 12 | 0 |
| P_INTC.IPRG | CMT0 | インタラプトプライオリティレジスタ G（IPRG）<br>CMT0 の CMI0 割り込み優先レベルの設定<br>:CMT0=b'1010(10)のとき、CMI0 割り込みは、優先レベル 10 に設定 | H'FFFF8354<br>ビット<br>7～4 | 10 |
| P_CMT.CMSTR | | コンペアマッチタイマスタートレジスタ（CMTSR）<br>CMCNT の動作／停止を選択する 16 ビットレジスタ | H'FFFF83D0 | H'0001 |
| | STR1 | カウンタスタート 1<br>：STR1=b'0 のとき、TCNT_1 のカウント動作は停止 | ビット 1 | |
| | STR0 | カウンタスタート 0<br>：STR0=b'1 のとき、TCNT_0 のカウント動作 | ビット 0 | |

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | | | ビット | |
| P_CMT.CMCSR_0 | | 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（CMCSR_0）<br>コンペアマッチ発生の表示、割り込み設定、タイマのクロック設定 | H'FFFF83D2 | H'0043 |
| | CMF | コンペアマッチフラグ<br>：CMCNT と CMCOR の値が一致したとき CMF=1 となる | ビット 7 | |
| | CMIE | コンペアマッチ割り込みイネーブル<br>：CMIE=1 のとき、コンペアマッチ割り込み（CMI）を許可 | ビット 6 | |
| | CKS1<br>CKS0 | CMCNT のカウンタクロックの選択<br>：CKS[1:0]=b'11 とき、内部クロック：Pφ／512 でカウント | ビット 1<br>ビット 0 | |
| P_CMT.CMCNT_0 | | コンペアマッチタイマカウンタ_0（CMCNT_0）<br>割り込み要求を発生させるためのアップカウンタ、16 ビットレジスタ | H'FFFF83D4 | H'0000 |
| P_CMT.CMCOR_0 | | コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）<br>CMCNT とのコンペアマッチ周期を設定、16 ビットレジスタ<br>CMCOR_0=H'1e84 のとき、100ms 周期でコンペアマッチ（Pφ/512 カウント、Pφ=40MHz） | H'FFFF83D6 | H'1e84 |

表 2.15　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | | | ビット | |
| DTC_R.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。 | 内蔵 RAM に配置 | H'84a0 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'00 のとき、DTDAR は固定 | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'01 のとき、リピートモード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'1 のとき、ソース側（DTSAR）がリピート領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'1 のとき、DTC 転送のたびに CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTCNMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| DTC_R.DTCRAH | | DTC 転送カウントレジスタ AH（DTCRAH）<br>DTC データ転送の転送回数の保持値を指定<br>3 回に設定 | 内蔵 RAM に配置 | H'03 |
| DTC_R.DTCRAL | | DTC 転送カウントレジスタ AH（DTCRAH）<br>DTC データ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'03 |
| DTC_R.DTIAR | | DTC 初期アドレスレジスタ（DTIAR）<br>DTC 転送データの転送元の初期アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S_data[0] |
| DTC_R.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC 転送データの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S_data[0] |
| DTC_R.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC 転送データの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &D_data |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTED | CMI0 | DTC イネーブルレジスタ D（DTED）<br>1 をセットすると対応する割り込み要因が DTC 起動要因として選択<br>：CMI0(DTED5)=b'1 のとき、CMT0 の CMI0 割り込みが起動要因 | H'FFFF8703<br>ビット 5 | 1 |

（4） 使用RAM説明

表2.16に、本タスクで使用するRAMの説明を示します。

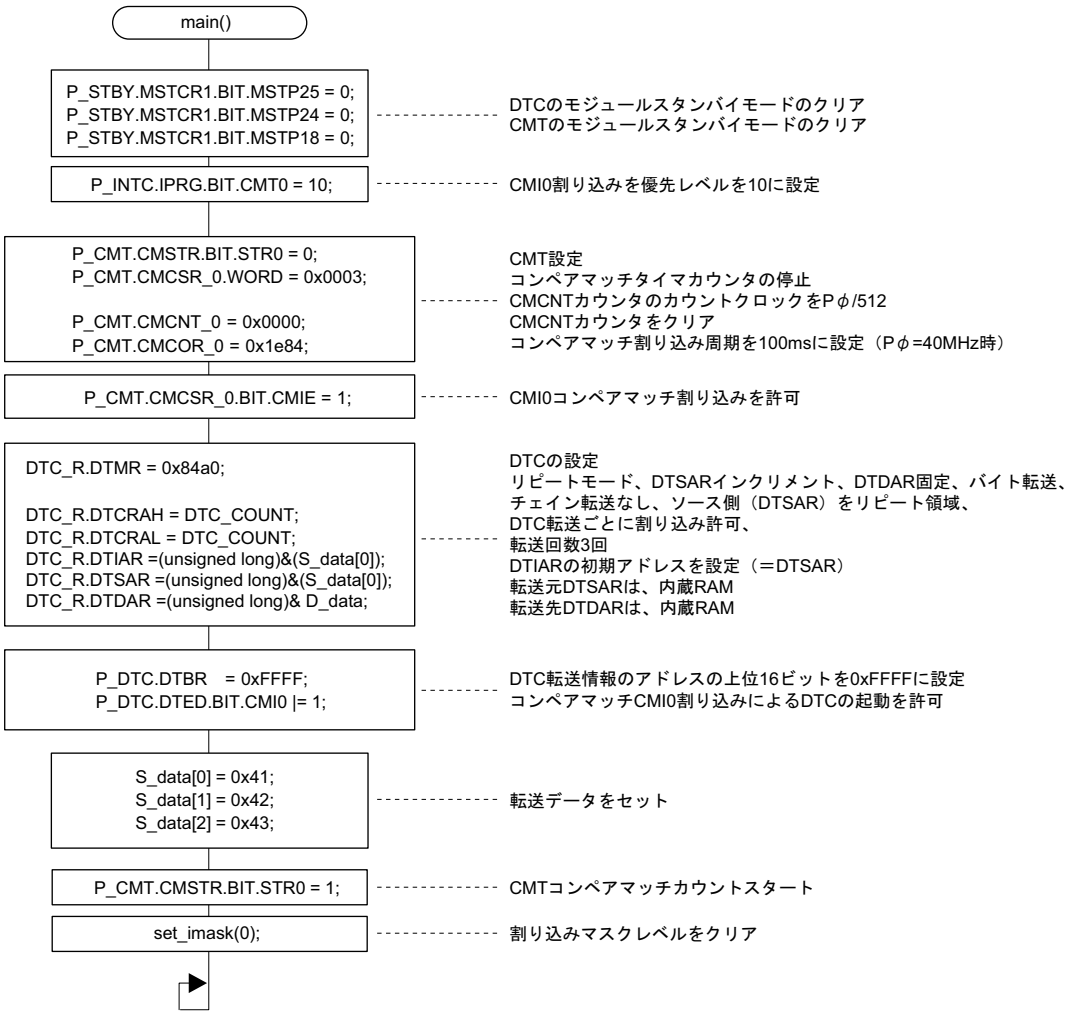
表 2.16　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
|---|---|---|---|
| S_data | DTC 転送データの格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |
| D_data | DTC データ転送後のデータ格納<br>1Byte データを格納 | 内蔵 RAM | メインルーチン |

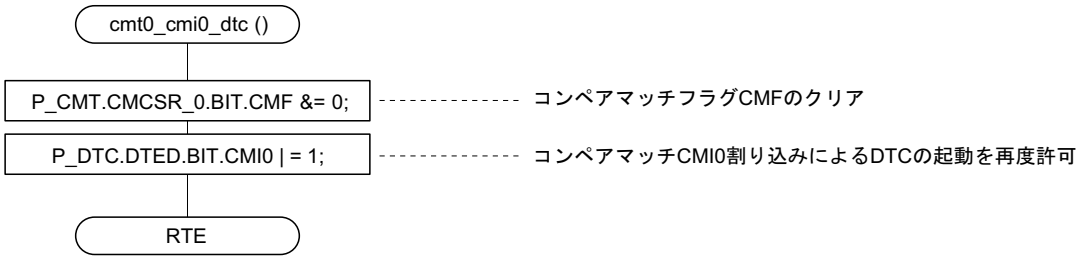
| DTC リピートモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

フローチャート

(a)　メイン処理




(b)　コンペアマッチ割り込み処理



プログラムリスト

```
/************************************************************/
/* SH7046F Series -SH7047- Application Note                 */
/* Data transfer Controller(DTC)                            */
/*    Repeat mode                                           */
/* Function                                                 */
/*:Data transfer Controller(DTC)                            */
/*:Compare Match Timer(CMT ch0)                             */
/*                                                          */
/* External input clock    :10MHz                           */
/* Internal CPU clock    :40MHz                             */
/* Internal peripheral clock :40MHz                         */
/*                                                          */
/* Written   :   2002/3/1  Rev.1.0                          */
/************************************************************/

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

/*------------ Symbol Definition ---------------------------------------------*/
struct st_dtc_repeat{          /* DTC  Repeat Mode information */
    unsigned short DTMR;       /* DTC Mode Register            */
    unsigned char DTCRAH;      /* maintains the Transfer count */
    unsigned char DTCRAL;      /* Transfer counter             */
    unsigned long DTIAR;       /* Initial Address Register     */
    unsigned long DTSAR;       /* source address register      */
    unsigned long DTDAR;       /* destination address register */
};

#define DTC_COUNT 3                  /* DTC Transmit count    */
#define DTC_R (*(volatile struct st_dtc_repeat*)0xFFFFE000)  /* DTC information
address */



/*------------ Function Definition -------------------------------------------*/
void main(void);
void cmt0_cmi0_dtc(void);
```

プログラムリスト

```
/*------------ RAM allocation Definition ------------------------------------*/
unsigned char S_data[DTC_COUNT];      /* source buffer memory           */
unsigned char D_data;                 /* destination buffer memory      */




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

    /* Set standby mode   */
    P_STBY.MSTCR1.BIT.MSTP25 = 0;       /* Disable DTC standby mode       */
    P_STBY.MSTCR1.BIT.MSTP24 = 0;       /* Disable DTC standby mode       */
    P_STBY.MSTCR2.BIT.MSTP12 = 0;       /* Disable CMT standby mode       */

    /* Set interrupt priority level (0 to 15)  */
    P_INTC.IPRG.BIT.CMT0 = 10;        /* CMT0 CMI0 interrupt level 10     */
    /* Initialize CMT0 for Interval timer                */
    P_CMT.CMSTR.BIT.STR0 = 0;    /* timer count stop             */
    P_CMT.CMCSR_0.WORD = 0x0003;
          /* CMF=0;     clear compare match flag                   */
          /* CMIE=0;    compare match interrupt disable            */
          /* CKS[1:0]=b'11;  clock = peripheral clock(Pφ)/512      */
    P_CMT.CMCNT_0 = 0x0000;      /* timer counter clear             */
    P_CMT.CMCOR_0 = 0x1e84;      /* 100ms@Pφ=40MHz                  */

    P_CMT.CMCSR_0.BIT.CMIE = 1;  /* compare match interrupt enable      */
```

プログラムリスト

```
    /* DTC information     */
    DTC_R.DTMR = 0x84a0;
         /* SM[1:0]=b'10;    DTSAR is incremented          */
         /* DM[1:0]=b'00;    DTDAR is fixed                */
         /* MD[1:0]=b'01;    Repeat mode                   */
         /* SZ[1:0]=b'00;    byte-size transfer            */
         /* DTS=1;       Source is Repeat area             */
         /* CHNE=0;      Chain transfer is disable         */
         /* DISEL=1;       Interrupt- every time           */
         /* NMIM=0;      NMI->Terminate DTC transfer       */

    DTC_R.DTCRAH = DTC_COUNT;       /* maintains the Transfer count          */
    DTC_R.DTCRAL = DTC_COUNT;       /* DTC transfer Count                    */
    DTC_R.DTIAR = (unsigned long)&(S_data[0]); /* set Initial address        */
    DTC_R.DTSAR = (unsigned long)&(S_data[0]); /* set source address         */
    DTC_R.DTDAR = (unsigned long)&D_data;  /* set destination address       */
    P_DTC.DTBR = 0xFFFF;          /* DTC information base register           */

    /* DTC transmit enable      */
    P_DTC.DTED.BIT.CMI0 |= 1;    /* interrupt sources CMT ch0(CMI0)          */

    /*  set transmit data  */
    S_data[0] = 0x41;
    S_data[1] = 0x42;
    S_data[2] = 0x43;


    P_CMT.CMSTR.BIT.STR0 = 1;  /* CMT0 timer count start              */

    set_imask(0);              /* clear interrupt mask level          */

    while(1);
}
```

| DTC リピーモードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

プログラムリスト

```
/************************************************************/
/* CMT0 Interrupt                                           */
/* Interval interrupt                                       */
/************************************************************/
#pragma interrupt(cmt0_cmi0_dtc)
void cmt0_cmi0_dtc(void)
{
    P_CMT.CMCSR_0.BIT.CMF &= 0;    /* Clear CMT0 compare match flag       */

    P_DTC.DTED.BIT.CMI0 |= 1;  /* set interrupt sources CMT ch0(CMI0)     */

}
```

## 2.3　DTC ブロック転送モードによるデータ転送（CMT、DTC）

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

（1）　図2.13に示すように、CMT（Compare Match Timer）のコンペアマッチ割り込みでDTC（Data Transfer Controller）を起動させ、内蔵RAMから内蔵RAMへのデータ転送を行います。

（2）　DTCデータ転送では、ブロック転送モードを使用し、図2.14に示すように、1ブロック4バイトのデータを3回転送します。

（3）　DTCの転送条件を表2.17に示します。

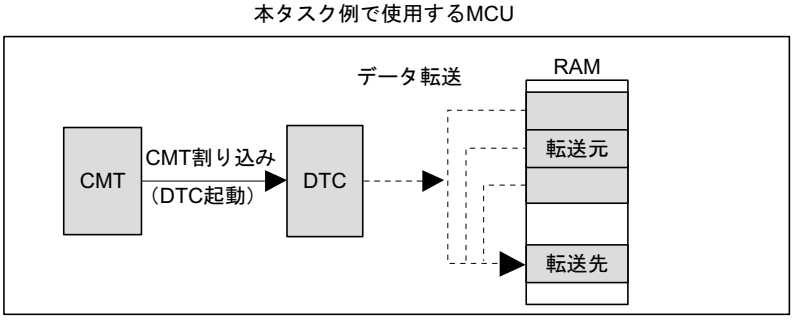



図 2.13　DTC を用いたデータ転送

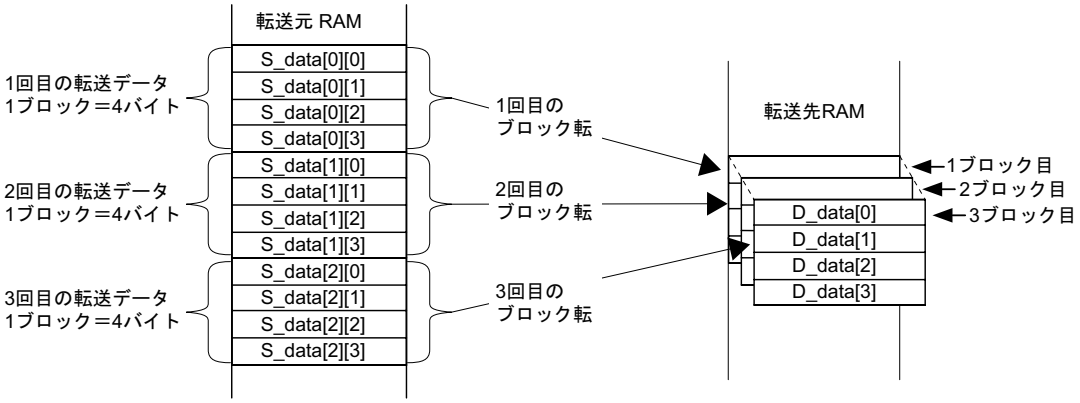



図 2.14　DTC ノーマルモードでのデータ転送

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

表 2.17　DTC 転送条件

| 条件 | 内容 |
|---|---|
| 転送モード | ブロック転送モード、ディスティネーション側（転送先）がブロック領域 |
| 転送回数 | 3 回 |
| ブロック長 | 4 |
| 転送データサイズ | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM |
| 転送先 | 内蔵 RAM（ブロック領域） |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント |
| 転送先アドレス | 転送後に転送先アドレスをインクリメント |
| 起動要因 | CMT ch0 のコンペアマッチ割り込み（CMI0）で起動 |
| 割り込み処理 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

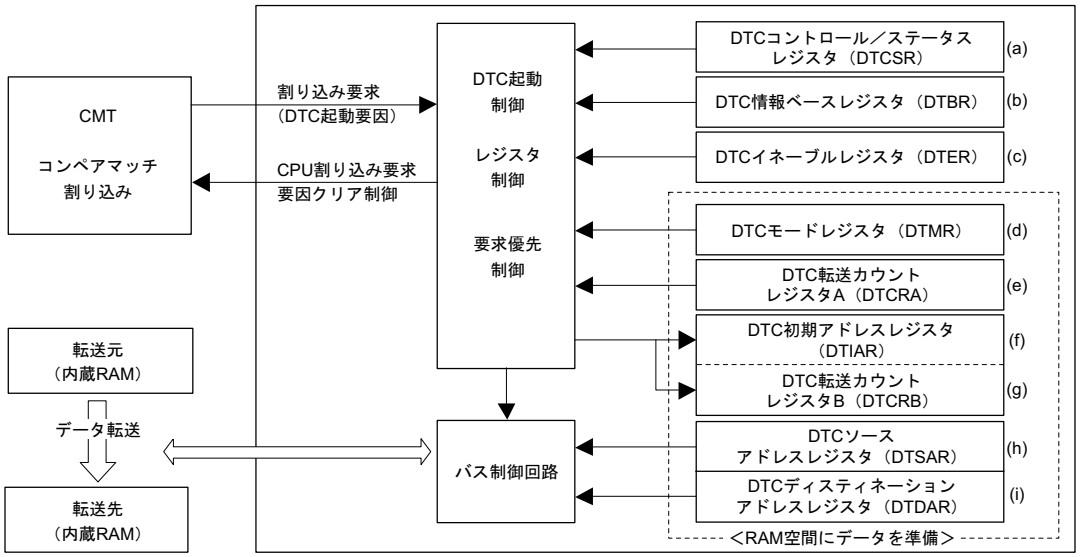
**使用機能説明**

(a) 図2.15にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ブロック転送モードを使用してデータ転送を行います。DTC起動要因をCMTのコンペアマッチ割り込みとして、内蔵RAMから内蔵RAMへのデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.15　DTC ブロック図

(b)　図2.16にCMT ch0のブロック図を示します。本タスクでは、CMT ch0のコンペアマッチ割り込みを起動要因としてDTCデータ転送を行います。以下にブロック図について説明します。

- コンペアマッチタイマスタートレジスタ（CMSTR）は、チャネル 0、1 のカウンタ（CMCNT）を動作させるか、停止させるかの設定を行う、16 ビットのレジスタです。
- 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（CMCSR_0）は、コンペアマッチ発生の表示、割り込みの許可／禁止の設定、カウントアップに用いられるクロックの選択をする 16 ビットのレジスタです。
- コンペアマッチタイマカウンタ_0（CMCNT_0）は、割り込み要求を発生させるためのアップカウンタとして使用する 16 ビットのレジスタです。
- コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）は、CMCNT とのコンペアマッチ周期を設定する 16 ビットのレジスタです。

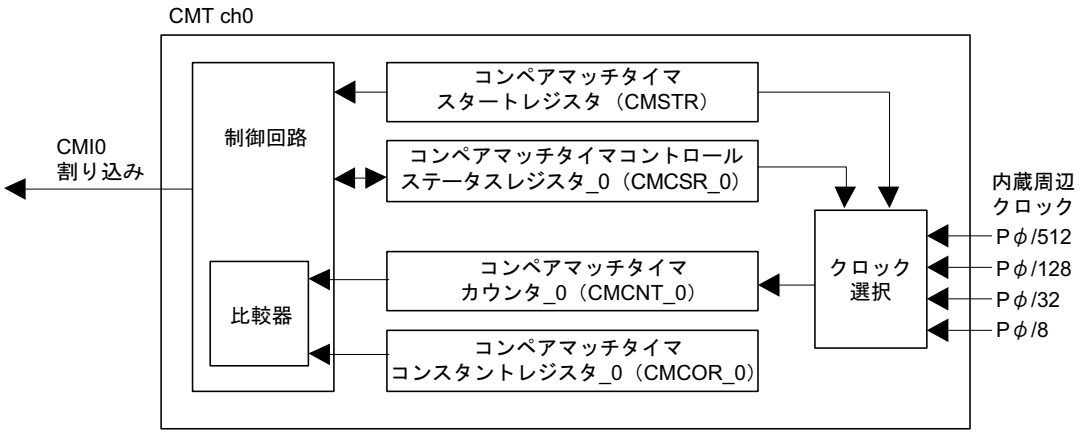



図 2.16　CMT ブロック図

(4)　表2.18に本タスク例の機能割り付けを示します。

表 2.18　機能割り付け

| 機能 | 分類 | 機能割り付け |
| --- | --- | --- |
| DTMR | DTC | ブロック転送モード、ディスティネーション側（転送先）がブロック領域 |
| DTCRA | DTC | 転送回数の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | CMT ch0 の CMI 割り込みで DTC の起動を許可 |
| CMSTR | CMT | CMT カウントのスタート |
| CMCSR_0 | CMT ch0 | カウントクロックの選択、割り込みの制御 |
| CMCNT_0 | CMT ch0 | カウンタ |
| CMCOR_0 | CMT ch0 | 周期の設定 |

動作説明

（1） 図2.17に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理によりDTCによる内蔵RAMから内蔵RAMへのデータ転送を行います。

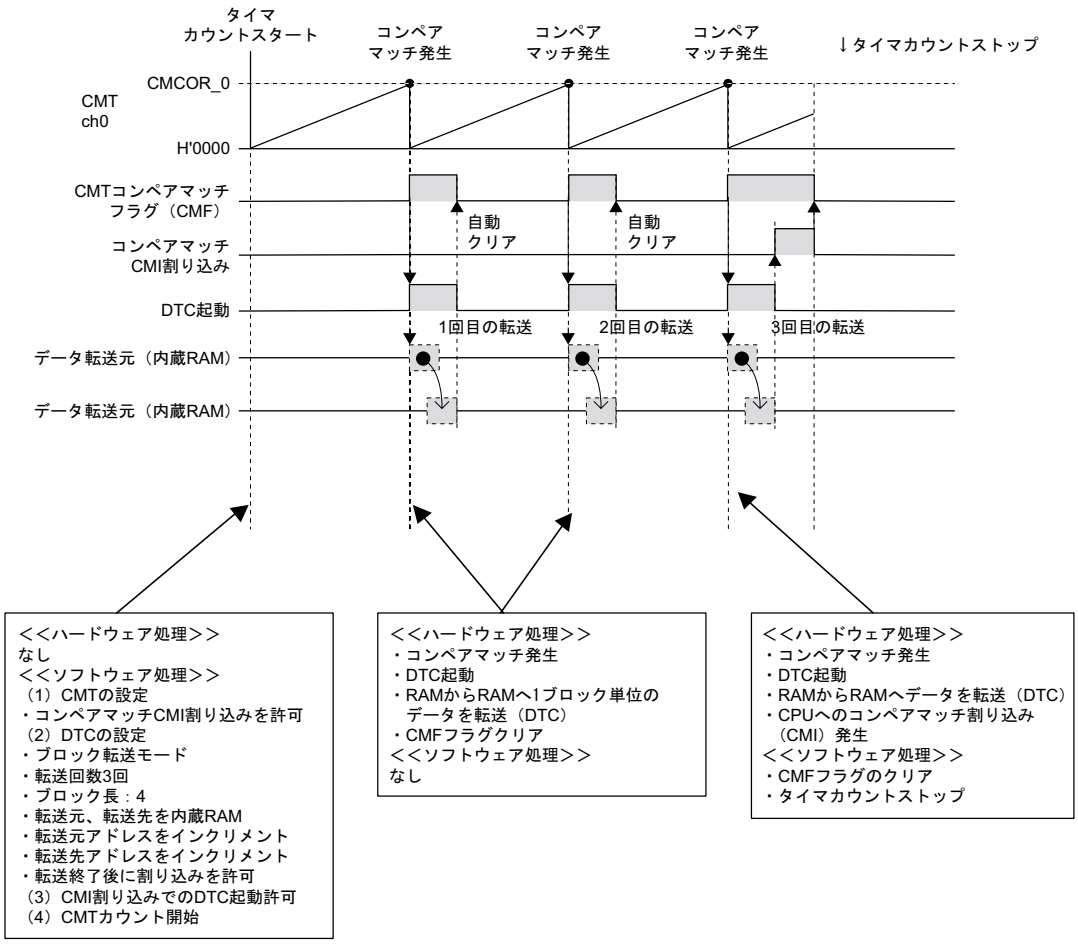



図 2.17　動作原理

**動作説明**

（2） 図2.18にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込みデータ転送を行います。

本タスクでは、シリアル送信のデータ転送の場合、TXI_2割り込みが起動要因となり、シリアル受信のデータ転送の場合、RXI_2割り込みが起動要因となります。
表2.19にブロック転送モード時のレジスタ情報の構成を示します。

表 2.19　DTC レジスタ情報一覧（ブロック転送モード）

| 設定アドレス | レジスタ名 | データ長 |
| --- | --- | --- |
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+6 | DTC 転送カウントレジスタ B（DTCRB） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

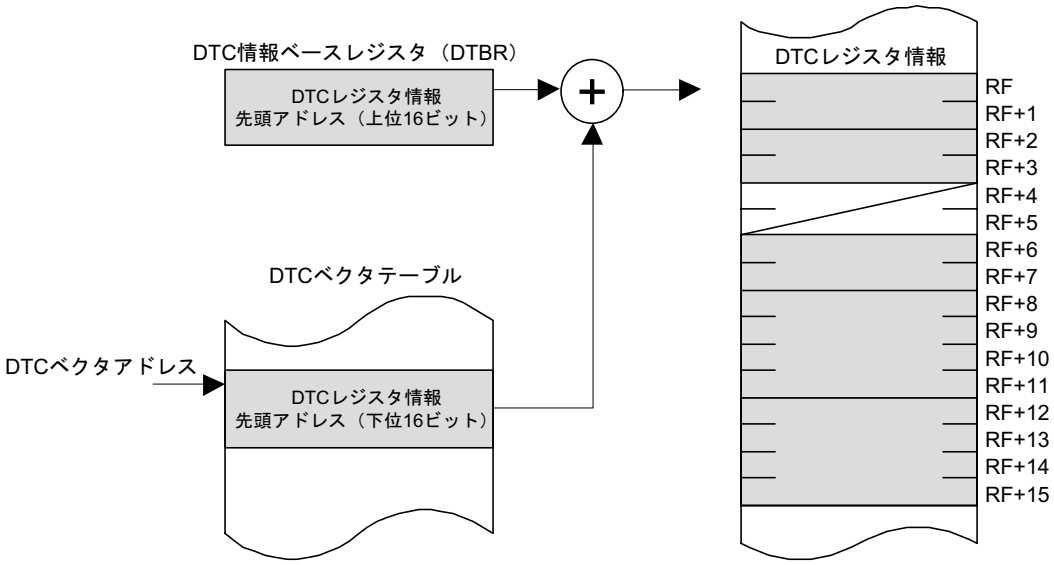



図 2.18　DTC ベクタアドレスと転送情報との対応

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

**表**2.20に、本タスク例のモジュールを示します。

表 2.20 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | CMT タイマの設定、DTC の初期設定、タイマスタート |
| CMI0 割り込み | cmt0_cmi0_dtc | CMT ch0 コンペアマッチ割り込み（CMI0）。DTC 指定回数転送終了時に割り込み発生 |

（2） 引数の説明

**表**2.21に、本タスク例で使用する引数を示します。

表 2.21 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| S_data[0][0]～[2][3] | DTC 転送元の転送データ格納 | メインルーチン | 1 バイト | 出力 |
| D_data[0]～[3] | DTC 転送先の転送データ格納 | メインルーチン | 1 バイト | 入力 |

（3） 使用内部レジスタ説明

**表**2.22、**表**2.23に、本タスク例の使用する内部レジスタを示します。

表 2.22 使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ値を設定 | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| P_STBY.MSTCR2 | MSTP12 | モジュールスタンバイコントロールレジスタ 2<br>CMT モジュールスタンバイ制御ビット<br>:MSTP12=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 12 | 0 |
| P_INTC.IPRG | CMT0 | インタラプトプライオリティレジスタ G（IPRG）<br>CMT0 の CMI0 割り込み優先レベルの設定<br>:CMT0=b'1010(10)のとき、CMI0 割り込みは、優先レベル 10 に設定 | H'FFFF8354<br>ビット<br>7～4 | 10 |
| P_CMT.CMSTR | | コンペアマッチタイマスタートレジスタ（CMTSR）<br>CMCNT の動作／停止を選択する 16 ビットレジスタ | H'FFFF83D0 | H'0001 |
| | STR1 | カウンタスタート 1<br>：STR1=b'0 のとき、TCNT_1 のカウント動作は停止 | ビット 1 | |
| | STR0 | カウンタスタート 0<br>：STR0=b'1 のとき、TCNT_0 のカウント動作 | ビット 0 | |

| レジスタ名 | ビット | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | | | ビット | |
| P_CMT.CMCSR_0 | | 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（CMCSR_0）<br>コンペアマッチ発生の表示、割り込み設定、タイマのクロック設定 | H'FFFF83D2 | H'0043 |
| | CMF | コンペアマッチフラグ<br>：CMCNT と CMCOR の値が一致したとき CMF=1 となる | ビット 7 | |
| | CMIE | コンペアマッチ割り込みイネーブル<br>：CMIE=1 のとき、コンペアマッチ割り込み（CMI）を許可 | ビット 6 | |
| | CKS1<br>CKS0 | CMCNT のカウンタクロックの選択<br>：CKS[1:0]=b'11 とき、内部クロック：Pφ／512 でカウント | ビット 1<br>ビット 0 | |
| P_CMT.CMCNT_0 | | コンペアマッチタイマカウンタ_0（CMCNT_0）<br>割り込み要求を発生させるためのアップカウンタ、16 ビットレジスタ | H'FFFF83D4 | H'0000 |
| P_CMT.CMCOR_0 | | コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）<br>CMCNT とのコンペアマッチ周期を設定、16 ビットレジスタ<br>CMCOR_0=H'1e84 のとき、100ms 周期でコンペアマッチ（Pφ/512 カウント、Pφ=40MHz） | H'FFFF83D6 | H'1e84 |

表 2.23　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | | | ビット | |
| DTC_B.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定 | 内蔵 RAM に配置 | H'a800 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'10 のとき、ブロック転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、デスティネーション側がブロック領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_B.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_B.DTCRB | | DTC 転送カウントレジスタ B（DTCRB）<br>DTC のブロック長を指定<br>ブロック長 4 に設定 | 内蔵 RAM に配置 | H'0004 |
| DTC_B.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S_data[0][0];<br> |
| DTC_B.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &D_data[0]; |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTED | CMI0 | DTC イネーブルレジスタ D（DTED）<br>1 をセットすると対応する割り込み要因が DTC 起動要因として選択<br>：CMI0(DTED5)=b'1 のとき、CMT0 の CMI0 割り込みが起動要因 | H'FFFF8703<br>ビット 5 | 1 |

（4） 使用RAM説明

表2.24に、本タスクで使用するRAMの説明を示します。

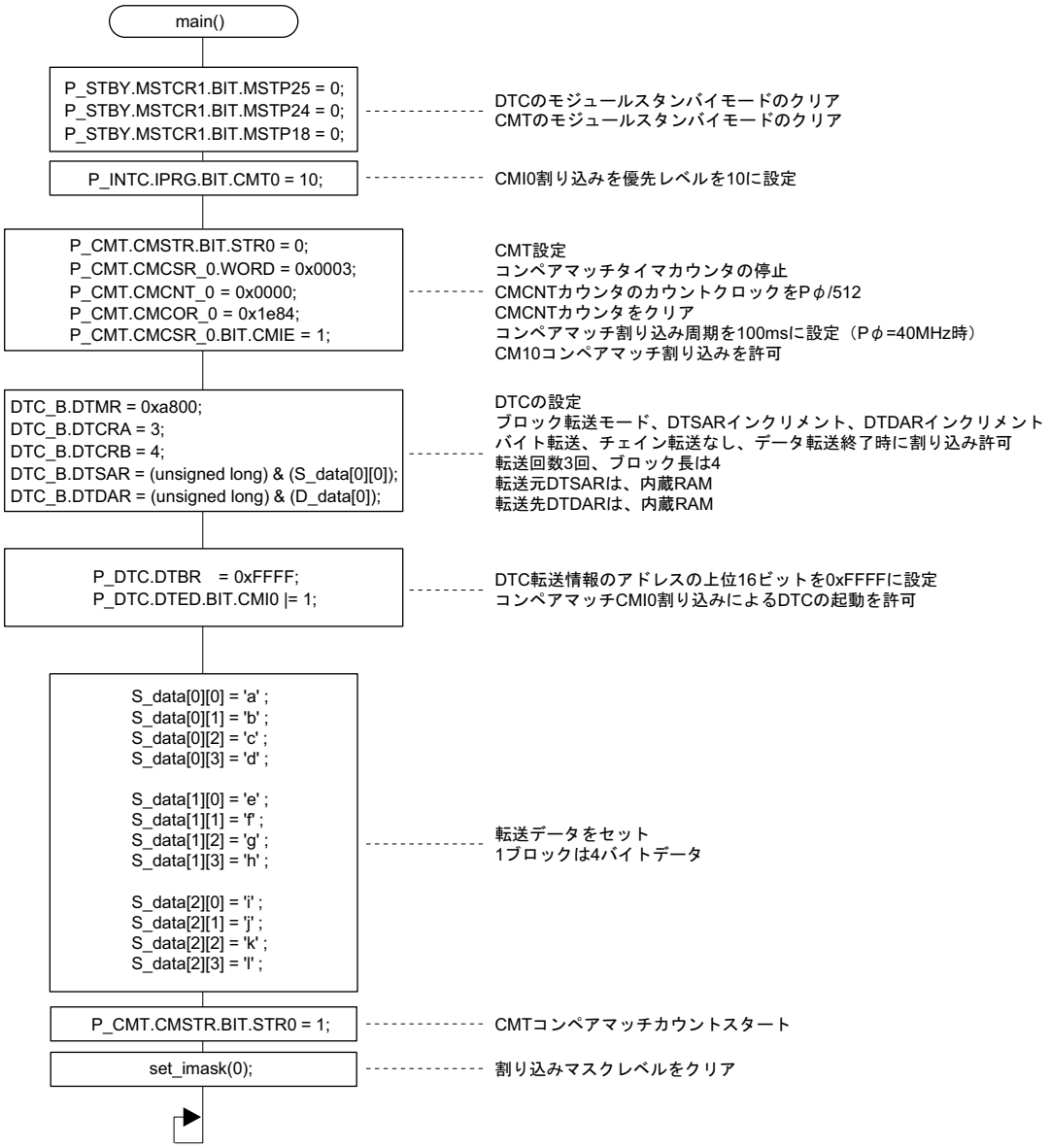
表 2.24 使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
|---|---|---|---|
| S_data | DTC 転送データの格納<br>1 ブロック（4Byte データ）を 3 ブロック格納する配列 | 内蔵 RAM | メインルーチン |
| D_data | DTC データ転送後のデータ格納<br>4Byte データを格納する配列 | 内蔵 RAM | メインルーチン |

| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

フローチャート

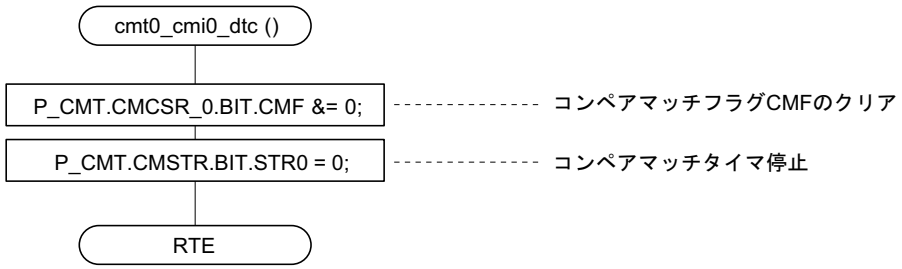
(a)　メイン処理



| DTC ブロック転送モードによるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

フローチャート

（b）　コンペアマッチ割り込み処理



プログラムリスト

```
/******************************************************/
/* SH7046F Series -SH7047- Application Note            */
/* Data transfer Controller(DTC)                      */
/*    Block Transfer mode                             */
/* Function                                           */
/*:Data transfer Controller(DTC)                      */
/*:Compare Match Timer(CMT ch0)                       */
/*                                                    */
/* External input clock    :10MHz                     */
/* Internal CPU clock    :40MHz                       */
/* Internal peripheral clock :40MHz                   */
/*                                                    */
/* Written   :   2002/3/1  Rev.1.0                    */
/******************************************************/


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


/*------------ Symbol Definition -----------------------------------------*/
struct st_dtc_block{       /* DTC Block Transfer Mode information */
    unsigned short DTMR;       /* DTC Mode Register                   */
    unsigned short DTCRA;      /* Transfer counter                    */
    unsigned short dummy;      /* Reserved                            */
    unsigned short DTCRB;      /* Block length                        */
    unsigned long DTSAR;       /* source address register             */
    unsigned long DTDAR;       /* destination address register        */
};

#define DTC_COUNT 3                /* DTC Transmit count    */
#define DTC_BLOCK_LENG  4          /* DTC block length      */
#define DTC_B (*(volatile struct st_dtc_block*)0xFFFFE000)    /* DTC information
address */


/*------------ Function Definition ---------------------------------------*/
```

プログラムリスト

```
void main(void);
void cmt0_cmi0_dtc(void);


/*------------ RAM allocation Definition ------------------------------------*/
unsigned char S_data[DTC_COUNT][DTC_BLOCK_LENG];  /* source buffer memory     */
unsigned char D_data[DTC_BLOCK_LENG];             /* destination buffer memory*/



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

    /* Set standby mode  */
    P_STBY.MSTCR1.BIT.MSTP25 = 0;      /* Disable DTC standby mode      */
    P_STBY.MSTCR1.BIT.MSTP24 = 0;      /* Disable DTC standby mode      */
    P_STBY.MSTCR2.BIT.MSTP12 = 0;      /* Disable CMT standby mode      */


    /* Set interrupt priority level (0 to 15) */
    P_INTC.IPRG.BIT.CMT0 = 10;       /* CMT0 CMI0 interrupt level 5  */
    /* Initialize CMT0 for Interval timer                 */
    P_CMT.CMSTR.BIT.STR0 = 0;     /* timer count stop         */
    P_CMT.CMCSR_0.WORD = 0x0003;
          /* CMF=0;     clear compare match flag           */
          /* CMIE=0;    compare match interrupt disable        */
          /* CKS[1:0]=b'11;   clock = peripheral clock(Pφ)/512      */
    P_CMT.CMCNT_0 = 0x0000;       /* timer counter clear            */
    P_CMT.CMCOR_0 = 0x1e84;       /* 100ms@Pφ=40MHz             */
    P_CMT.CMCSR_0.BIT.CMIE = 1;     /* compare match interrupt enable

    /* DTC information     */
    DTC_B.DTMR = 0xa800;        /* */
```

プログラムリスト

```
            /* SM[1:0]=b'10;    DTSAR is incremented            */
            /* DM[1:0]=b'00;    DTDAR is incremented          */
            /* MD[1:0]=b'10;    Block transfer mode             */
            /* SZ[1:0]=b'00;    byte-size transfer              */
            /* DTS=0;     destination is Block area             */
            /* CHNE=0;    Chain transfer is canceled          */
            /* DISEL=0;   Interrupt->transfer ends            */
            /* NMIM=0;    NMI->Terminate DTC transfer         */
        DTC_B.DTCRA = DTC_COUNT;        /* DTC transfer Count        */
        DTC_B.DTCRB = DTC_BLOCK_LENG;      /* DTC transfer Block length    */
        DTC_B.DTSAR =(unsigned long)&(S_data[0][0]);  /* set source address      */
        DTC_B.DTDAR =(unsigned long)&(D_data[0]); /* set destination address      */
        P_DTC.DTBR= 0xFFFF;           /* DTC information base register    */

        /* DTC transmit enable       */
        P_DTC.DTED.BIT.CMI0 |= 1;     /* interrupt sources CMT ch0(CMI0)       */

        /* set transmit data block 1*/
        S_data[0][0] = 'a';
        S_data[0][1] = 'b';
        S_data[0][2] = 'c';
        S_data[0][3] = 'd';
        /* set transmit data block 2*/
        S_data[1][0] = 'e';
        S_data[1][1] = 'f';
        S_data[1][2] = 'g';
        S_data[1][3] = 'h';
        /* set transmit data block 3*/
        S_data[2][0] = 'i';
        S_data[2][1] = 'j';
        S_data[2][2] = 'k';
        S_data[2][3] = 'l';

        P_CMT.CMSTR.BIT.STR0 = 1;   /* CMT0 timer count start          */
        set_imask(0);               /* clear interrupt mask level      */
        while(1);
    }
```

プログラムリスト

```
/************************************************************/
/* CMT0 Interrupt                                           */
/* Interval interrupt                                       */
/************************************************************/
#pragma interrupt(cmt0_cmi0_dtc)
void cmt0_cmi0_dtc(void)
{
     P_CMT.CMCSR_0.BIT.CMF &= 0;     /* Claer CMT0 compare match flag    */
     P_CMT.CMSTR.BIT.STR0 = 0;       /* CMT0 timer count stop      */

}
```

## 2.4 DTC チェイン転送によるデータ転送（CMT、DTC）

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

（1） 図2.19に示すように、CMT（Compare Match Timer）のコンペアマッチ割り込みでDTC（Data Transfer Controller）を起動させ、内蔵RAMから内蔵RAMへのデータ転送を行います。

（2） DTCデータ転送は、チェイン転送で行います。図2.20に示すように、3バイトの転送を連続して行います。

（3） DTCの転送条件を表2.25に示します。

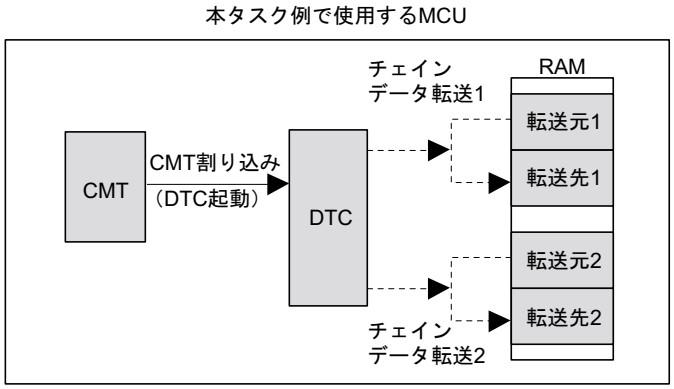



図 2.19　DTC を用いたデータ転送

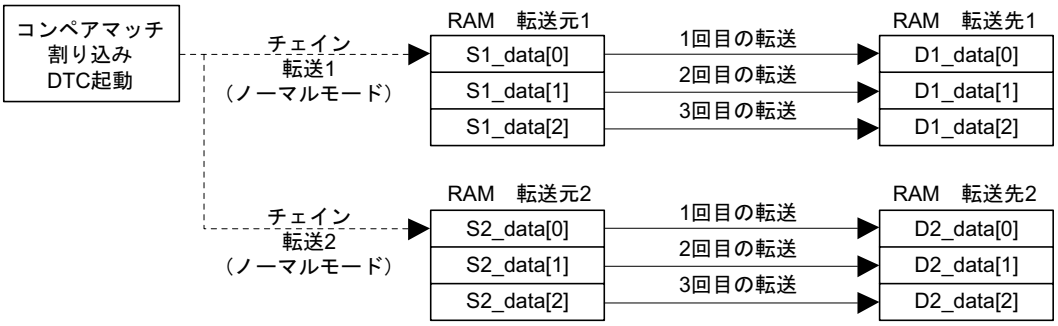



図 2.20　DTC ノーマルモードでのデータ転送

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

仕様

表 2.25　DTC 転送条件

| 条件 | チェイン転送 1 の内容 | チェイン転送 2 の内容 |
|---|---|---|
| 転送モード | ノーマルモード、チェイン転送 | ノーマルモード |
| 転送回数 | 3 回 | 3 回 |
| 転送データサイズ | バイト（Byte）転送 | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM | 内蔵 RAM |
| 転送先 | 内蔵 RAM | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスを<br>インクリメント | 転送後に転送元アドレスを<br>インクリメント |
| 転送先アドレス | 転送後に転送先アドレスを<br>インクリメント | 転送後に転送先アドレスを<br>インクリメント |
| 起動要因 | CMT ch0 のコンペアマッチ割り込み（CMI0）で起動 | チェイン転送 1 が終了に実行 |
| 割り込み処理 | なし（チェイン転送のため） | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

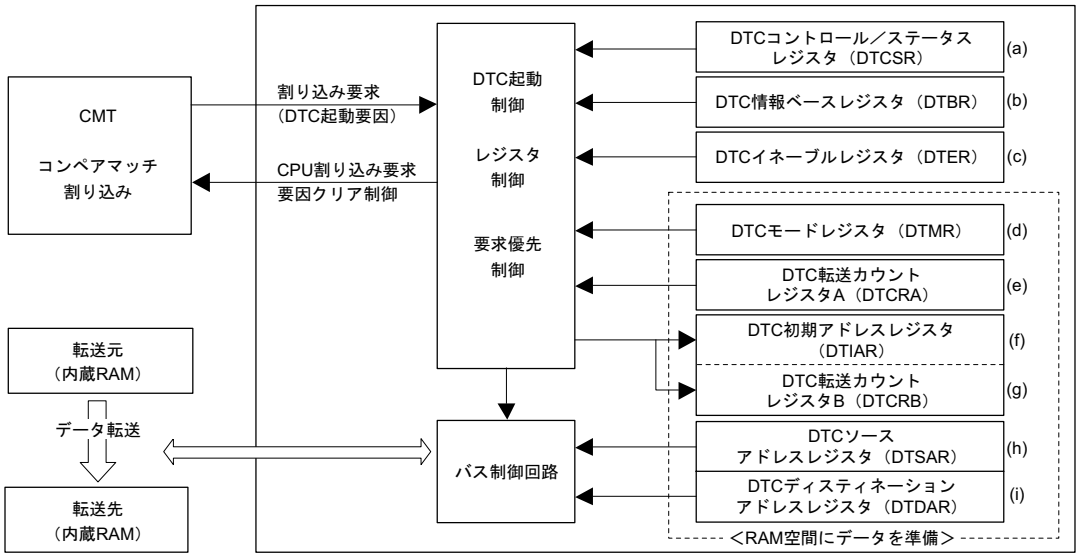
**使用機能説明**

(a) **図2.21**にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ノーマルモードを使用してデータ転送を行います。DTC起動要因をCMTのコンペアマッチ割り込みとして、内蔵RAMから内蔵RAMへのデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.21　DTC ブロック図

使用機能説明

(b)　図2.22にCMT ch0のブロック図を示します。本タスクでは、CMT ch0のコンペアマッチ割り込みを起動要因としてDTCデータ転送を行います。以下にブロック図について説明します。

- コンペアマッチタイマスタートレジスタ（CMSTR）は、チャネル 0、1 のカウンタ（CMCNT）を動作させるか、停止させるかの設定を行います。16 ビットのレジスタです。
- 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（CMCSR_0）は、コンペアマッチ発生の表示、割り込みの許可／禁止の設定、カウントアップに用いられるクロックの選択をする 16 ビットのレジスタです。
- コンペアマッチタイマカウンタ_0（CMCNT_0）は、割り込み要求を発生させるためのアップカウンタとして使用する 16 ビットのレジスタです。
- コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）は、CMCNT とのコンペアマッチ周期を設定する 16 ビットのレジスタです。

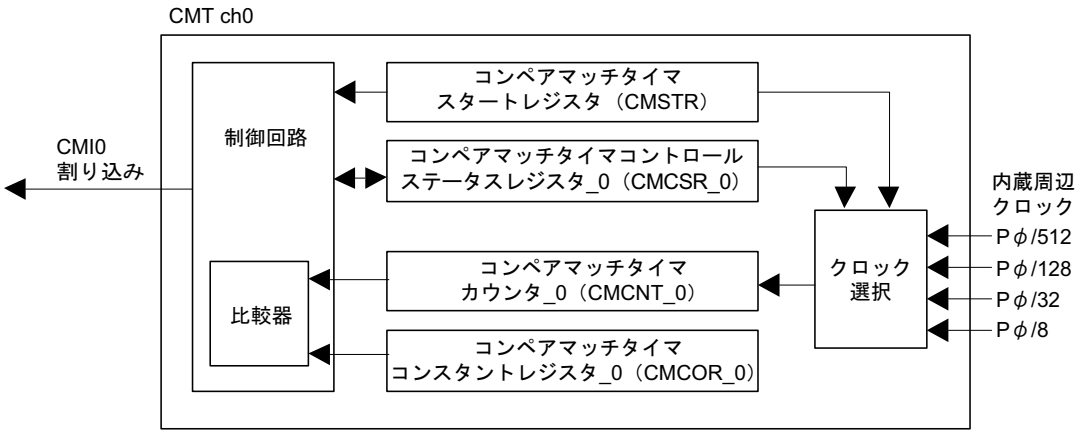



図 2.22　CMT ブロック図

(4)　表2.26に本タスク例の機能割り付けを示します。

表 2.26　機能割り付け

| 機能 | 分類 | 機能割り付け |
| --- | --- | --- |
| DTMR | DTC | DTC をノーマルモード、チェン転送に設定 |
| DTCRA | DTC | 転送回数の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | CMT ch0 の CMI 割り込みで DTC の起動を許可 |
| CMSTR | CMT | CMT カウントのスタート |
| CMCSR_0 | CMT ch0 | カウントクロックの選択、割り込みの制御 |
| CMCNT_0 | CMT ch0 | カウンタ |
| CMCOR_0 | CMT ch0 | 周期の設定 |

動作説明

(1) 図2.23に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理によりDTCによる内蔵RAMから内蔵RAMへのデータ転送を行います。

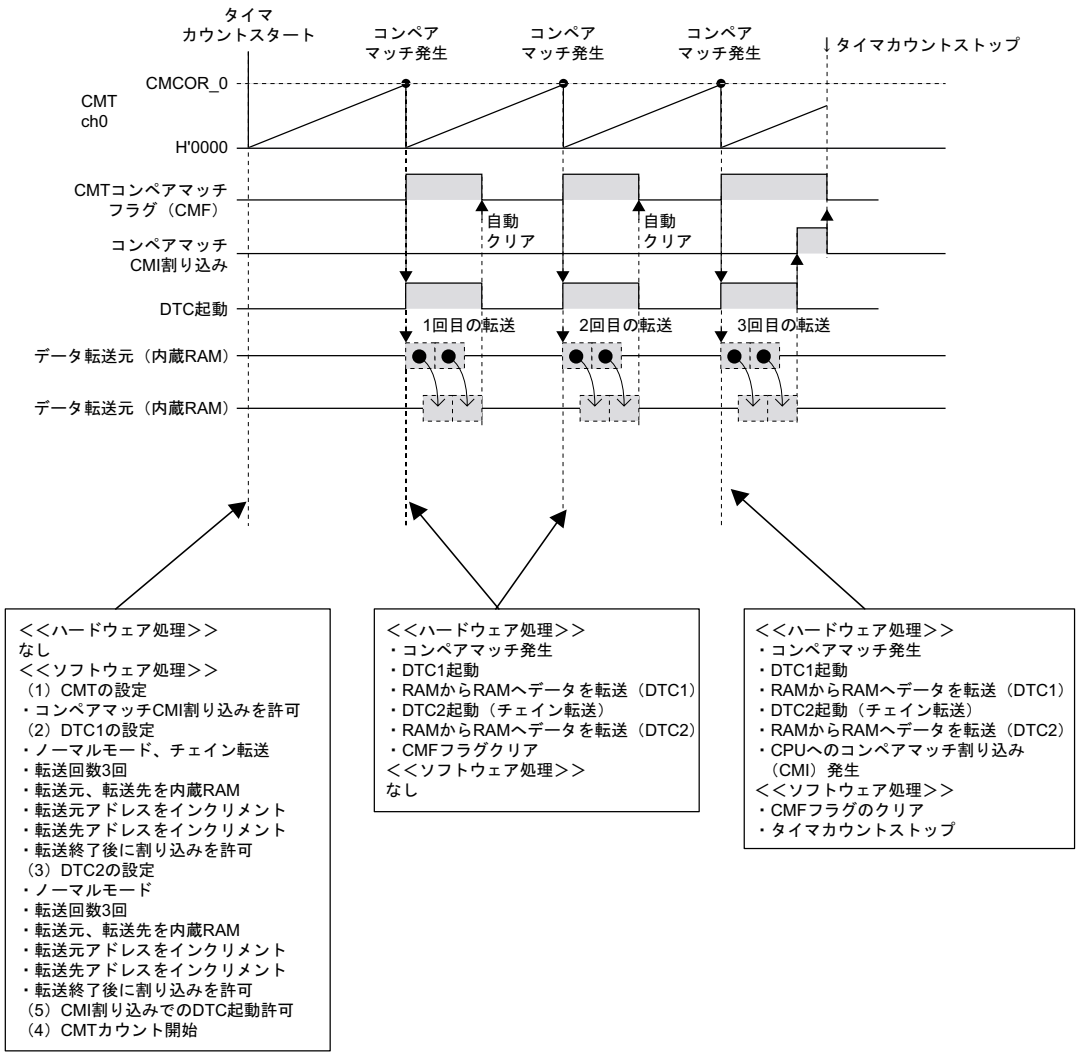



図 2.23　動作原理

動作説明

（2） 図2.24にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込み、データ転送を行います。

本タスクでは、CMTのコンペアマッチ割り込みがDTCが起動要因となります。
表2.27にノーマルモードでのチェイン転送時のレジスタ情報の構成を示します。

表 2.27　DTC レジスタ情報一覧（ノーマルモード、チェイン転送）

| 設定アドレス | レジスタ名 | データ長 |
| --- | --- | --- |
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

動作説明

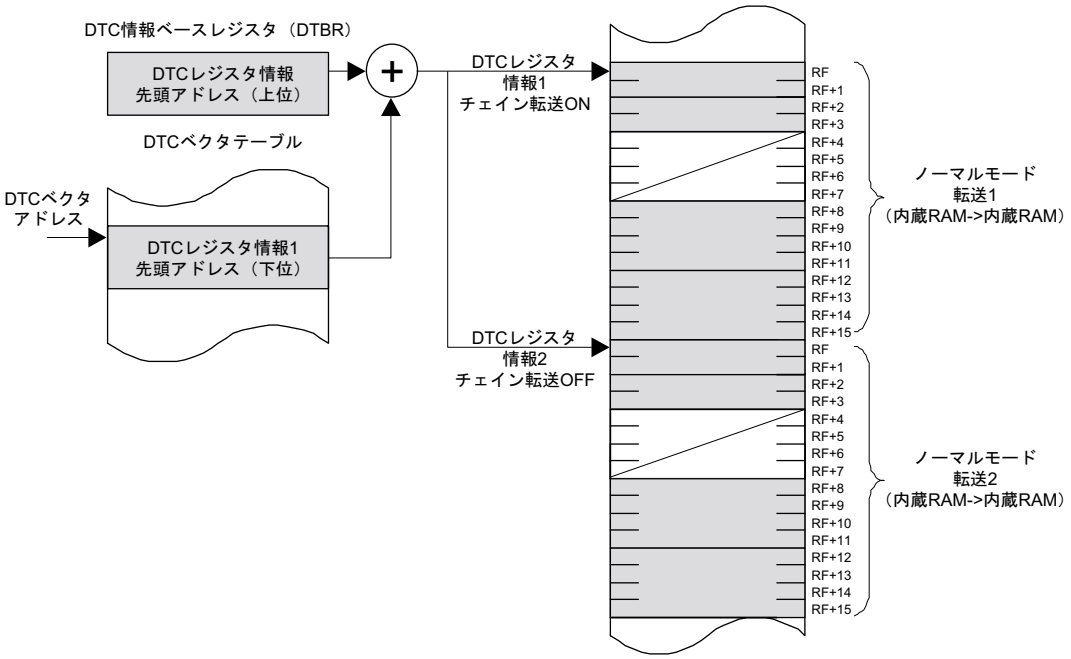



図 2.24　DTC ベクタアドレスと転送情報との対応（ノーマルモードによるチェイン転送）

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

**表2.28**に、本タスク例のモジュールを示します。

表 2.28 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | CMT タイマの設定、DTC の初期設定、タイマスタート |
| CMI0 割り込み | cmt0_cmi0_dtc | CMT ch0 コンペアマッチ割り込み（CMI0）。DTC 指定回数転送終了時に割り込み発生 |

（2） 引数の説明

**表2.29**に、本タスク例で使用する引数を示します。

表 2.29 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| S1_data[0]～[2] | DTC1 転送元の転送データ格納 | メインルーチン | 1 バイト | 出力 |
| D1_data[0]～[2] | DTC1 転送先の転送データ格納 | メインルーチン | 1 バイト | 入力 |
| S2_data[0]～[2] | DTC2 転送元の転送データ格納 | メインルーチン | 1 バイト | 出力 |
| D2_data[0]～[2] | DTC2 転送先の転送データ格納 | メインルーチン | 1 バイト | 入力 |

（3） 使用内部レジスタ説明

**表2.30**、**表2.31**に、本タスク例の使用する内部レジスタを示します。

表 2.30 使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ値を設定 | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| P_STBY.MSTCR2 | MSTP12 | モジュールスタンバイコントロールレジスタ 2<br>CMT モジュールスタンバイ制御ビット<br>:MSTP12=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 12 | 0 |
| P_INTC.IPRG | CMT0 | インタラプトプライオリティレジスタ G（IPRG）<br>CMT0 の CMI0 割り込み優先レベルの設定<br>:CMT0=b'1010(10)のとき、CMI0 割り込みは優先レベル 10 に設定 | H'FFFF8354<br>ビット<br>7～4 | 10 |
| P_CMT.CMSTR | | コンペアマッチタイマスタートレジスタ（CMTSR）<br>CMCNT の動作／停止を選択する 16 ビットレジスタ | H'FFFF83D0 | H'0001 |
| | STR1 | カウンタスタート 1<br>：STR1=b'0 のとき、TCNT_1 のカウント動作は停止 | ビット 1 | |
| | STR0 | カウンタスタート 0<br>：STR0=b'1 のとき、TCNT_0 のカウント動作 | ビット 0 | |

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_CMT.CMCSR_0 | | 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（CMCSR_0）<br>コンペアマッチ発生の表示、割り込み設定、タイマのクロック設定 | H'FFFF83D2 | H'0043 |
| | CMF | コンペアマッチフラグ<br>：CMCNT と CMCOR の値が一致したとき CMF=1 となる | ビット 7 | |
| | CMIE | コンペアマッチ割り込みイネーブル<br>：CMIE=1 のとき、コンペアマッチ割り込み（CMI）を許可 | ビット 6 | |
| | CKS1<br>CKS0 | CMCNT のカウンタクロックの選択<br>：CKS[1:0]=b'11 とき、内部クロック：Pφ／512 でカウント | ビット 1<br>ビット 0 | |
| P_CMT.CMCNT_0 | | コンペアマッチタイマカウンタ_0（CMCNT_0）<br>割り込み要求を発生させるためのアップカウンタ、16 ビットレジスタ | H'FFFF83D4 | H'0000 |
| P_CMT.CMCOR_0 | | コンペアマッチタイマコンスタントレジスタ_0（CMCOR_0）<br>CMCNT とのコンペアマッチ周期を設定、16 ビットレジスタ<br>CMCOR_0=H'1e84 のとき、100ms 周期でコンペアマッチ（Pφ/512 カウント、Pφ=40MHz） | H'FFFF83D6 | H'1e84 |

表 2.31　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| DTC_N1.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定 | 内蔵 RAM に配置 | H'a040 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマルモード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、デスティネーション側がブロック領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'1 のとき、チェイン転送をあり | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_N1.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定、3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_N1.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S1_data[0] |
| DTC_N1.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &D1_data[0] |
| DTC_N2.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定 | 内蔵 RAM に配置 | H'a000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマルモード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、デスティネーション側がブロック領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき､指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTCNMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |
| DTC_N2.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定、3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_N2.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &S2_data[0] |
| DTC_N2.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | &D2_data[0] |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTED | CMI0 | DTC イネーブルレジスタ D（DTED）<br>1 をセットすると対応する割り込み要因が DTC 起動要因として選択<br>：CMI0(DTED5)=b'1 のとき、CMT0 の CMI0 割り込みが起動要因 | H'FFFF8703<br>ビット 5 | 1 |

ソフトウェア説明

（4） 使用RAM説明

表2.32に、本タスクで使用するRAMの説明を示します。

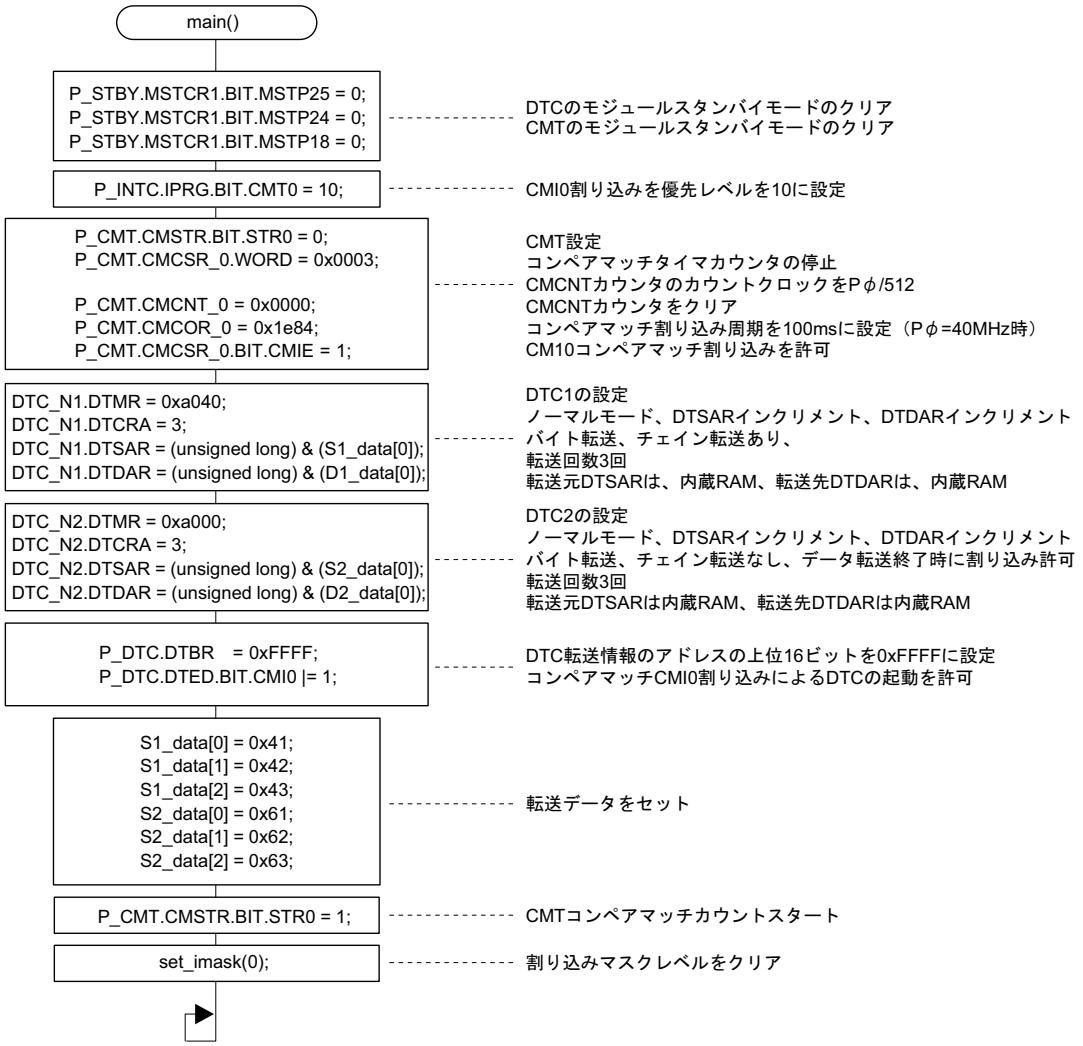
表 2.32　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
|---|---|---|---|
| S1_data | DTC1 転送データの格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |
| D1_data | DTC1 データ転送後のデータ格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |
| S2_data | DTC2 転送データの格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |
| D2_data | DTC2 データ転送後のデータ格納<br>3Byte データを格納する配列 | 内蔵 RAM | メインルーチン |

| DTC チェイン転送によるデータ転送（CMT、DTC） | 使用機能 | CMT、DTC |
|---|---|---|

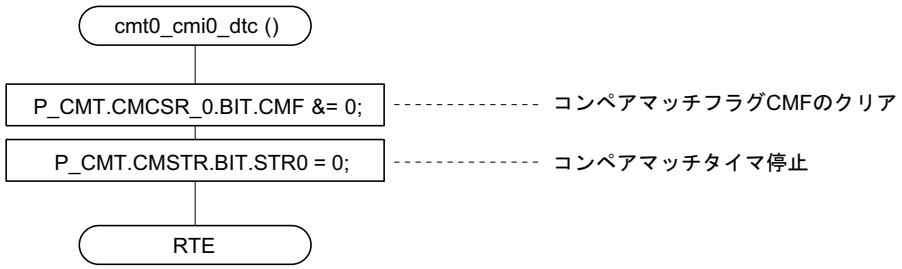
フローチャート

(a)　メイン処理



フローチャート

(b)　コンペアマッチ割り込み処理



プログラムリスト

```
/******************************************************/
/* SH7046F Series -SH7047- Application Note            */
/* Data transfer Controller(DTC)                      */
/*    Normal mode with Chain Transfer                 */
/* Function                                           */
/*:Data transfer Controller(DTC)                      */
/*:Compare Match Timer(CMT ch0)                       */
/*                                                    */
/* External input clock    :10MHz                     */
/* Internal CPU clock    :40MHz                       */
/* Internal peripheral clock :40MHz                   */
/*                                                    */
/* Written   :   2002/3/1  Rev.1.0                    */
/******************************************************/

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


/*------------ Symbol Definition -----------------------------------------*/
struct st_dtc_n{        /* DTC Normal Transfer Mode information  */
    unsigned short DTMR;       /* DTC Mode Register            */
    unsigned short DTCRA;      /* Transfer counter             */
    unsigned short dummy1;     /* Reserved                     */
    unsigned short dummy2;     /* Reserved                     */
    unsigned long DTSAR;       /* source address register      */
    unsigned long DTDAR;       /* destination address register */
};

#define DTC_COUNT13                /* DTC Transmit count    */
#define DTC_COUNT23                /* DTC Transmit count    */
#define DTC_N1 (*(volatile struct st_dtc_n*)0xFFFFE000)  /* DTC information
address */
#define DTC_N2 (*(volatile struct st_dtc_n*)0xFFFFE010)  /* DTC information
address */
```

プログラムリスト

```
/*------------ Function Definition -------------------------------------------*/
void main(void);
void cmt0_cmi0_dtc(void);


/*------------ RAM allocation Definition -------------------------------------*/
unsigned char S1_data[DTC_COUNT1];           /* buffer memory       */
unsigned char D1_data[DTC_COUNT1];           /* buffer memory       */
unsigned char S2_data[DTC_COUNT2];           /* buffer memory       */
unsigned char D2_data[DTC_COUNT2];           /* buffer memory       */


/*************************************************************/
/* main Program                                              */
/*************************************************************/
void main( void )
{
    /* Set standby mode  */
    P_STBY.MSTCR1.BIT.MSTP25 = 0;      /* Disable DTC standby mode      */
    P_STBY.MSTCR1.BIT.MSTP24 = 0;      /* Disable DTC standby mode      */
    P_STBY.MSTCR2.BIT.MSTP12 = 0;      /* Disable CMT standby mode   */

    /* Set interrupt priority level (0 to 15) */
    P_INTC.IPRG.BIT.CMT0 = 10;       /* CMT0 CMI0 interrupt level 10   */
    /* Initialize CMT0 for Interval timer                 */
    P_CMT.CMSTR.BIT.STR0 = 0;     /* timer count stop         */
    P_CMT.CMCSR_0.WORD = 0x0003;
          /* CMF=0;     clear compare match flag           */
          /* CMIE=0;    compare match interrupt disable        */
          /* CKS[1:0]=b'11;   clock = peripheral clock(Pφ)/512      */
    P_CMT.CMCNT_0 = 0x0000;       /* timer counter clear           */
    P_CMT.CMCOR_0 = 0x1e84;       /* 100ms clock=Pφ/512 Pφ=40MHz      */
    P_CMT.CMCSR_0.BIT.CMIE = 1;   /* compare match interrupt enable      */

    /* DTC1 information    */
    DTC_N1.DTMR = 0xa040;         /* */
```

プログラムリスト

```
        /* SM[1:0]=b'10;    DTSAR is incremented            */
        /* DM[1:0]=b'10;    DTDAR is incremented            */
        /* MD[1:0]=b'00;    Normal transfer mode            */
        /* SZ[1:0]=b'00;    byte-size transfer              */
        /* DTS=0;           Source is block area            */
        /* CHNE=1;          Chain transfer is enable          */
        /* DISEL=0;         Interrupt->transfer ends            */
        /* NMIM=0;          NMI->Terminate DTC transfer         */
    DTC_N1.DTCRA = DTC_COUNT1;        /* DTC transfer Count       */
    DTC_N1.DTSAR =(unsigned long)&(S1_data[0]); /* set source address      */
    DTC_N1.DTDAR =(unsigned long)&(D1_data[0]); /* set destination address    */
    /* DTC2 information    */
    DTC_N2.DTMR = 0xa000;         /* */
        /* SM[1:0]=b'10;    DTSAR is incremented            */
        /* DM[1:0]=b'10;    DTDAR is incremented            */
        /* MD[1:0]=b'00;    Normal transfer mode            */
        /* SZ[1:0]=b'00;    byte-size transfer              */
        /* DTS=0;           Source is block area            */
        /* CHNE=0;          Chain transfer is canceled          */
        /* DISEL=0;         Interrupt->transfer ends            */
        /* NMIM=0;      NMI->Terminate DTC transfer         */
    DTC_N2.DTCRA = DTC_COUNT2;        /* DTC transfer Count       */
    DTC_N2.DTSAR =(unsigned long)&(S2_data[0]); /* set source address      */
    DTC_N2.DTDAR =(unsigned long)&(D2_data[0]); /* set destination address    */

    P_DTC.DTBR= 0xFFFF;           /* DTC information base register    */
    P_DTC.DTED.BIT.CMI0 |= 1;     /* interrupt sources CMT ch0(CMI0)       */

    /* set transmit data  */
    S1_data[0] = 0x41;
    S1_data[1] = 0x42;
    S1_data[2] = 0x43;
    S2_data[0] = 0x61;
    S2_data[1] = 0x62;
    S2_data[2] = 0x63;
```

プログラムリスト

```
    set_imask(0);        /* clear interrupt mask level          */
    P_CMT.CMSTR.BIT.STR0 = 1;   /* CMT0 timer count start            */
    while(1);
}
/************************************************************/
/* CMT0 Interrupt                                           */
/* Interval interrupt                                       */
/************************************************************/
#pragma interrupt(cmt0_cmi0_dtc)
void cmt0_cmi0_dtc(void)
{
 P_CMT.CMCSR_0.BIT.CMF &= 0;   /* Clear CMT0 compare match flag   */
   P_CMT.CMSTR.BIT.STR0 = 0;   /* CMT0 timer count stop      */
}
```

## 2.5　調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC）

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

仕様

（1）　図2.25に示すように、調歩同期式シリアル転送機能と、DTCによるデータ転送機能を使用して3バイトデータの同時送受信動作を行います。

（2）　図2.26に示すようにDTC（Data Transfer Controller）の転送機能を使用して、シリアル送信データの転送、シリアル受信データの内蔵RAMへの格納を行います。

（3）　送受信するデータのフォーマットは、データ長：8ビット、偶数パリティ、ストップビット長：1ビット、データの最下位ビットから送受信するLSBファースト方式により、ビットレート1Mbpsで通信を行います。

（4）　DTCの転送条件を表2.33に示します。

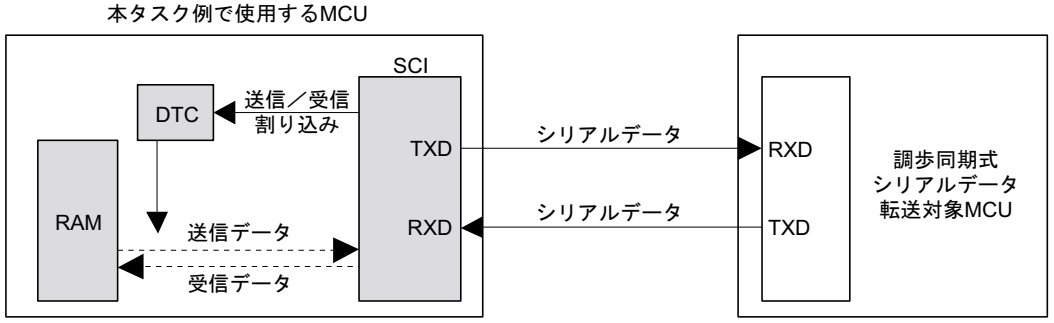



図 2.25　調歩同期式シリアルデータ同時送受信

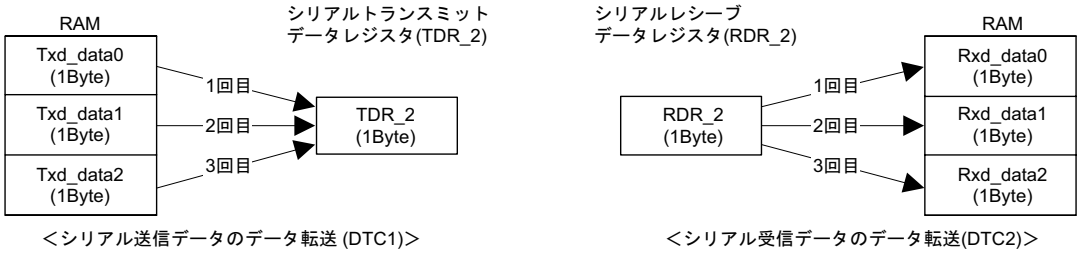



図 2.26　DTC を用いたデータ転送

表 2.33　DTC 転送条件

| 条件 | シリアル送信側 DTC の転送条件（DTC1） | シリアル受信側 DTC の転送条件（DTC2） |
|---|---|---|
| 転送モード | ノーマルモード | ノーマルモード |
| 転送回数 | 3 回 | 3 回 |
| 転送サイズ | バイト（Byte）転送 | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM | シリアルレシーブデータレジスタ（RDR_2） |
| 転送先 | シリアルトランスミットデータレジスタ（TDR_2） | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント | 転送元アドレスは固定 |
| 転送先アドレス | 転送先アドレスは固定 | 転送後に転送先アドレスをインクリメント |
| 起動要因 | SCI ch2 の送信割り込み（TXI_2）で起動 | SCI ch2 の受信割り込み（RXI_2）で起動 |
| 割り込み処理 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

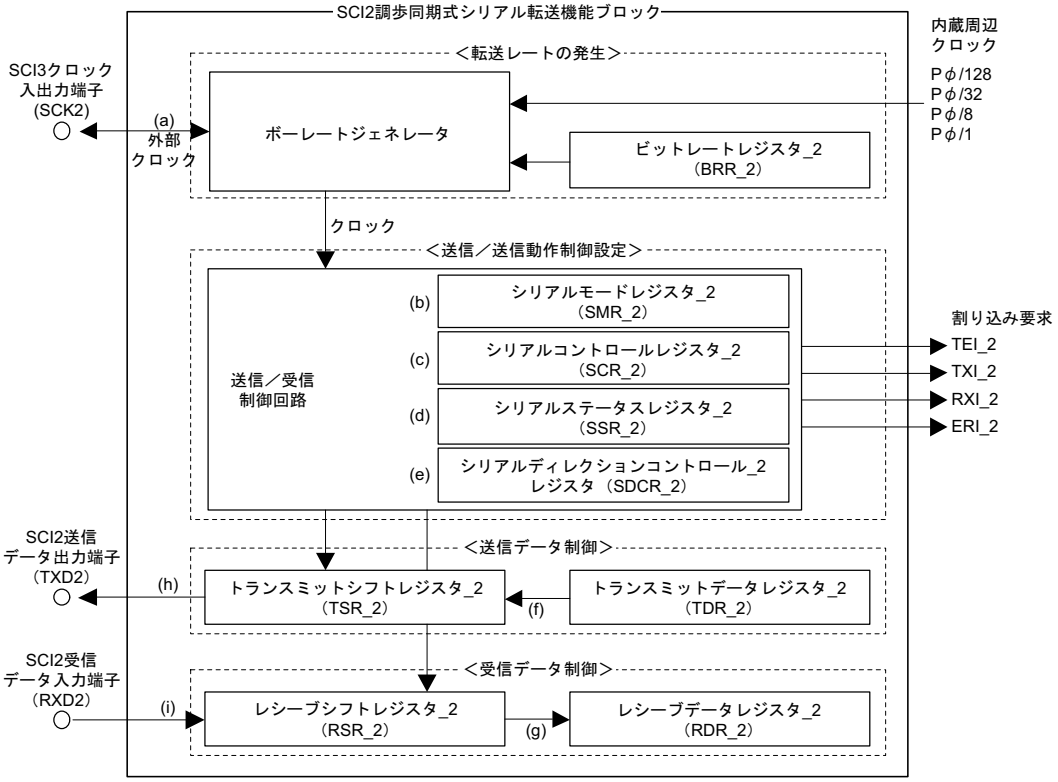
**使用機能説明**

（1） 本タスク例では、シリアルコミュニケーションインタフェース（SCI：Serial Communication Interface）とデータトランスファコントローラ（DTC:Data Transfer Controller）を使用して、調歩同期式のシリアルデータの同時送受信動作を行います。

（a） 図2.27に調歩同期式シリアルデータ送受信同時動作のブロック図を示します。以下に調歩同期式シリアルデータ同時送受信のブロック図について説明します。

- 調歩同期式モードは、キャラクタ単位で同期をとる調歩同期方式でシリアルデータ通信を行います。調歩同期方式では Universal Asynchronous Receiver/Transmitter（UART）や、Asynchronous Communication Interface Adapter（ACIA）などの標準の調歩同期式通信用 LSI とのシリアル通信ができます。また、調歩同期式モードでは複数のプロセッサ間のシリアル通信機能（マルチプロセッサ通信機能）を備えています。
- 内蔵周辺クロック Pφは、内蔵周辺機能を動作させるための基準クロックです。
- レシーブシフトレジスタ（RSR_2）は、シリアルデータを受信するためのレジスタです。RSR_2 に RXD2 端子から入力されたシリアルデータを、LSB（ビット 0）から受信した順にセットしパラレルデータに変換します。1 バイトのデータを受信すると、データは自動的に RDR_2 へ転送されます。CPU から RSR_2 を直接リード/ライトすることはできません。
- レシーブデータレジスタ_2（RDR_2）は、受信したシリアルデータを格納する 8 ビットのレジスタです。1 バイトのデータの受信が終了すると、受信したデータを RSR_2 から RDR_2 へ転送し、受信動作を完了します。その後 RSR_2 は受信可能となります。RSR_2 と RDR_2 はダブルバッファになっているため連続した受信動作が可能です。RDR_2 は受信専用レジスタなので CPU からライトできません。
- トランスミットシフトレジスタ_2（TSR_2）は、シリアルデータを送信するためのレジスタです。TDR_2 から送信データをいったん TSR_2 に転送し、LSB（ビット 0）から順に TXD 端子に送出することでシリアルデータ送信を行います。1 バイトのデータを送信すると、自動的に TDR_2 から TSR_2 へ次の送信データを転送し、送信を開始します。ただし、TDR_2 にデータが書き込まれていない（TDRE に"1"がセットされている）場合には TDR_2 から TSR_2 へのデータ転送は行いません。CPU から TSR_2 を直接リード／ライトすることはできません。
- トランスミットデータレジスタ_2（TDR_2）は、送信データを格納する 8 ビットのレジスタです。TSR_2 の"空"を検出すると、TDR_2 に書き込まれた送信データを TSR_2 に転送し、シリアルデータ送信を開始します。TSR_2 のシリアルデータ送信中に、TDR_2 に次の送信データをライトしておくと、連続送信が可能です。TDR_2 は、常に CPU によるリード／ライトが可能です。
- シリアルモードレジスタ_2（SMR_2）は、シリアルデータ通信フォーマットの設定と、内蔵ボーレートジェネレータのクロックソースを選択するための 8 ビットのレジスタです。
- シリアルコントロールレジスタ_2（SCR_2）は、送信／受信動作および送信／受信クロックソースの選択を行う 8 ビットのレジスタです。
- シリアルステータスレジスタ_2（SSR_2）は、SCI2 のステータスフラグと送受信マルチプロセッサビットで構成されています。TDRE、RDRF、OER、PER、FER はクリアのみ可能です。
- シリアルディレクションコントロール_2 レジスタ（SDCR_2）は、DIR ビットにより、LSB ファースト／MSB ファーストの選択を行います。シリアル通信モードによらず、8 ビット長の場合 LSB ファースト／MSB ファーストの選択が可能です。7 ビット長の場合 LSB ファーストを選択し、MSB ファーストの選択は行わないでください。
- ビットレートレジスタ_2（BRR_2）は、ビットレートを調整するための 8 ビットのレジスタです。

使用機能説明




【注】
(a) クロックを出力する。
(b) シリアルデータ通信フォーマットの設定と、ボーレートジェネレータのクロックソースを選択を行う。
(c) 送信／受信動作、クロックソース、SCK端子の機能選択を行う。
(d) ステータスフラグ（トランスミットデータレジスタエンプティ、レシーブデータレジスタフル、オーバランエラー、フレーミングエラー、パリティエラー）によりSCI2の動作状態を示す。
(e) LSBファースト／MSBファーストの選択を行う。
(f) TSR_2の"空"を検出することにより、TDR_2に書き込まれた送信データをTSR_2に転送。
(g) 1バイトのデータの受信が終了すると、受信したデータをRSR_2からRDR_2へ転送。

図 2.27　調歩同期式シリアルデータの送受信ブロック図

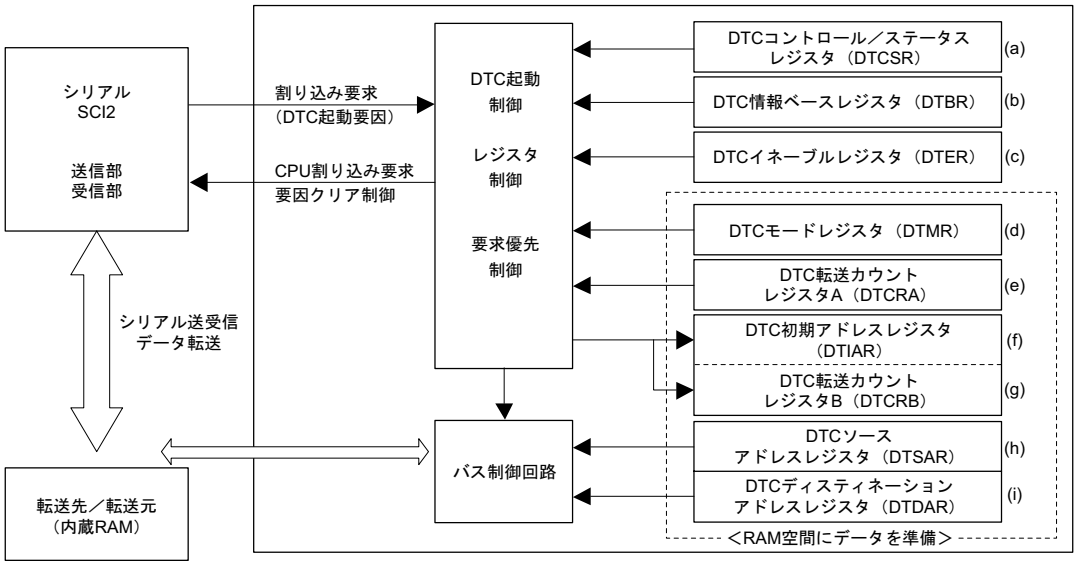
| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

**使用機能説明**

(b)　**図2.28**にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ノーマル転送モードを使用して、シリアル送受信データの転送を行います。DTC起動要因を、シリアルの送信のTXI割り込み、シリアル受信のRXI割り込みとして、DTCデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。
- 本タスクでは、シリアル送信データの転送とシリアル受信データの転送に、それぞれ DTC ノーマルモードでのデータ転送を行います。ノーマルモードのレジスタ情報（DTMR、DTSAR、DTDAR、DTCRA、DTCRB）をシリアル送信用、シリアル受信用と、2 つ準備してください。

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.28　DTC ブロック図

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

使用機能説明

(2)　表2.34に本タスク例の機能割り付けを示します。

表 2.34　機能割り付け

| 機能 | 分類 | 機能割り付け |
|---|---|---|
| TXD2 | 端子 | チャネル 2 の送信データ出力端子 |
| RXD2 | 端子 | チャネル 2 の受信データ入力端子 |
| SMR_2 | SCI2 | 通信フォーマットの設定、調歩同期式モードに設定 |
| SCR_2 | SCI2 | 送受信動作、割り込みを許可 |
| SSR_2 | SCI2 | SCI2 の動作状態を示すステータスフラグ |
| SDCR_2 | SCI2 | LSB ファーストに設定 |
| BBR_2 | SCI2 | 送受信のビットレートを設定 |
| TSR_2 | SCI2 | シリアルデータを送信するためのレジスタ |
| TDR_2 | SCI2 | 送信データを格納するレジスタ |
| RSR_2 | SCI2 | シリアルデータを受信するためのレジスタ |
| RDR_2 | SCI2 | 受信データを格納するレジスタ |
| DTMR | DTC | DTC をノーマル転送モードに設定 |
| DTCRA | DTC | 転送回数の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | シリアル受信、シリアル送信時に DTC の起動を許可 |

動作説明

（1） 図2.29に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理により調歩同期式シリアルデータの同時送受信を行います。

（a） 送信処理
- 調歩同期式シリアルで 3 バイトのデータを送信。
- 3 バイトの送信データは DTC を用いて、内蔵 RAM から SIC に転送。
- DTC の起動には、シリアルの TXI_2 割り込みを使用。

（b） 受信処理
- 調歩同期式シリアルで 3 バイトのデータを受信。
- 3 バイトの受信データは DTC を用いて、SCI から内蔵 RAM に転送。
- DTC の起動には、シリアルの RXI_2 割り込みを使用。

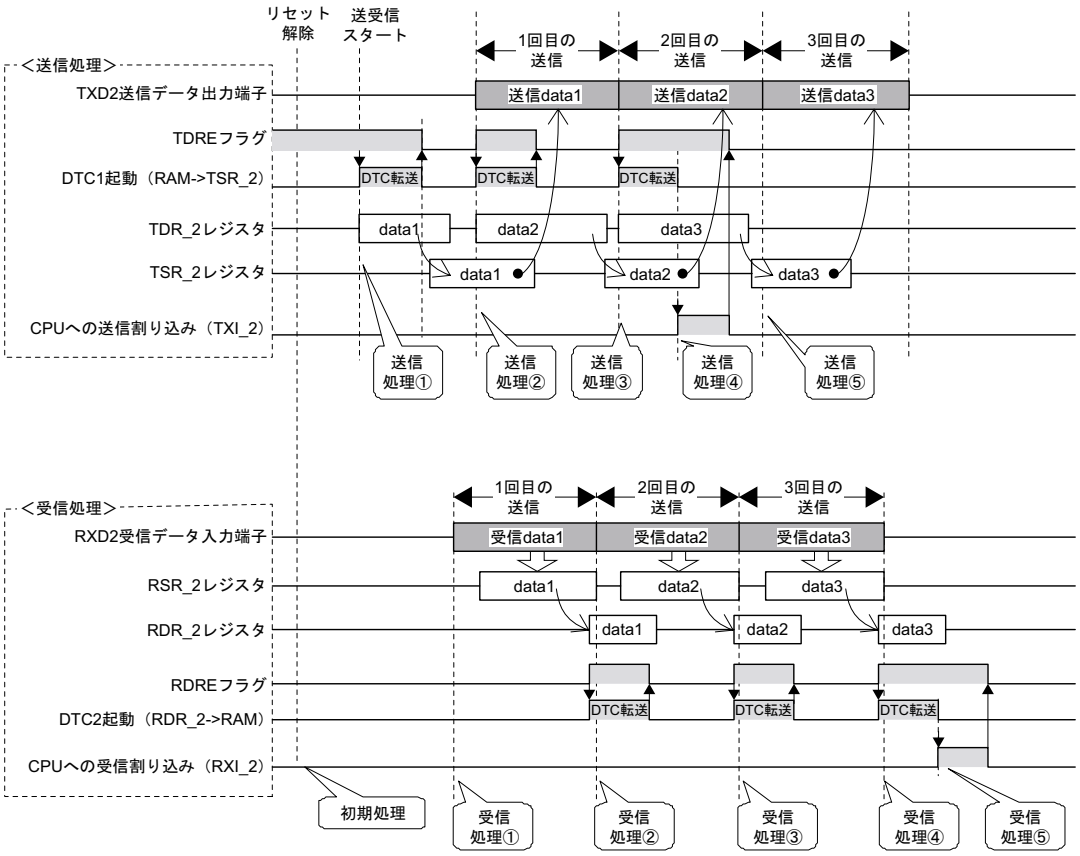



図 2.29 動作原理

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

動作説明

以下に図 2.29 の説明を示します。

<table>
<tr><td>初期処理</td><td colspan="3"><<ソフトウェア処理>><br>（1）SCI2 の設定<br>・調歩同期式モードに設定、LSB ファースト転送<br>・送信割り込み、受信割り込み、送信動作、受信動作を許可<br>（2）DTC 設定<br>・ノーマル転送モードに設定<br>・シリアル送信割り込み（TXI_2）での DTC1 起動を許可、シリアル受信割り込み（RXI_2）での DTC2 起動を許可<br>・初期処理</td></tr>
<tr><td rowspan="2">送信処理①</td><td><<ハードウェア処理>><br>・TXI_2 割り込みで DTC1 起動（1 回目）<br>・送信データ data1 を内蔵 RAM から TDR_2 レジスタに転送（DTC1）<br>・TDRE をクリア（DTC1）</td><td rowspan="2">受信処理①</td><td><<ハードウェア処理>><br>・受信スタート<br>・受信データ data1 を RSR レジスタに取り込む</td></tr>
<tr><td><<ソフトウェア処理>><br>なし</td><td><<ソフトウェア処理>><br>なし</td></tr>
<tr><td rowspan="2">送信処理②</td><td><<ハードウェア処理>><br>・TDRE フラグが 0 のとき、TDR_2 から TSR_2 レジスタに送信データ data1 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・送信を開始（SCI2）<br>・TXI_2 割り込みで DTC1 起動（2 回目）<br>・送信データ data2 を内蔵 RAM から TDR_2 に転送（DTC1）<br>・TDRE フラグをクリア（DTC1）</td><td rowspan="2">受信処理②</td><td><<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data1 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・次のフレーム受信を開始（SCI2）<br>・受信データ data2 を RSR レジスタに取り込む（SCI2）<br>・RXI_2 割り込みで DTC2 起動（1 回目）<br>・受信データ data1 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2）</td></tr>
<tr><td><<ソフトウェア処理>><br>なし</td><td><<ソフトウェア処理>><br>なし</td></tr>
<tr><td rowspan="2">送信処理③</td><td><<ハードウェア処理>><br>・最終ビットを送り出すとき、TDRE フラグをチェック（SCI2）<br>・TDRE が 0 のとき、TDR_2 から TSR_2 レジスタに送信データ data2 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・次のフレーム送信を開始（SCI2）<br>・TXI_2 割り込みで DTC1 起動（3 回目）<br>・送信データ data3 を内蔵 RAM から TDR_2 に転送し終了（DTC1）、TDRE フラグはクリアされない</td><td rowspan="2">受信処理③</td><td><<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data2 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・次のフレーム受信を開始<br>・受信データ data3 を RSR レジスタに取り込む<br>・RXI_2 割り込みで DTC2 起動（2 回目）<br>・受信データ data2 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2）</td></tr>
<tr><td><<ソフトウェア処理>><br>なし</td><td><<ソフトウェア処理>><br>なし</td></tr>
</table>

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

動作説明

| | | | |
|---|---|---|---|
| 送信処理④ | <<ハードウェア処理>><br>・CPU に対して TXI_2 割り込みを発生 | 受信処理④ | <<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data3 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・RXI_2 割り込みで DTC2 起動（2 回目）<br>・受信データ data2 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2） |
| | <<ソフトウェア処理>><br>・TDRE フラグをクリア<br>・TXI_2 割り込みを禁止 | | <<ソフトウェア処理>><br>なし |
| 送信処理⑤ | <<ハードウェア処理>><br>・最終ビットを送り出すとき、TDRE フラグをチェック（SCI2）<br>・TDRE が 0 のとき、TDR_2 から TSR_2 レジスタに送信データ data3 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・最後のフレーム送信を開始（SCI2） | 受信処理⑤ | <<ハードウェア処理>><br>CPU に対して TXI_2 割り込みを発生 |
| | <<ソフトウェア処理>><br>なし | | <<ソフトウェア処理>><br>・TDRE フラグをクリア<br>・TXI_2 割り込みを禁止 |

（2） 図2.30にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込みデータ転送を行います。

動作説明

本タスクでは、シリアル送信のデータ転送の場合、TXI_2割り込みが起動要因となり、シリアル受信のデータ転送の場合、RXI_2割り込みが起動要因となります。
表2.35にノーマルモード時のレジスタ情報の構成を示します。

表 2.35　DTC レジスタ情報一覧（ノーマルモード）

| 設定アドレス | レジスタ名 | データ長 |
|---|---|---|
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

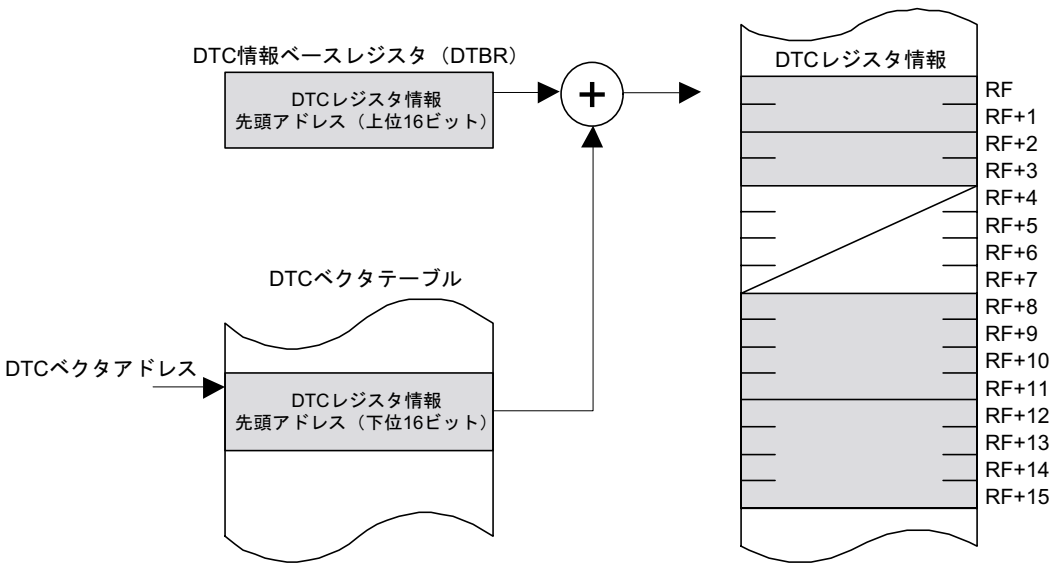



図 2.30　DTC ベクタアドレスと転送情報との対応

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

**表2.36**に、本タスク例のモジュールを示します。

表 2.36 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | SCI ch2 クロック同期式シリアル通信、DTC の初期設定、シリアル通信をスタート |
| SCI 送信完了割り込み | txi2_end | SCI ch2 の送信完了割り込み。DTC 指定回数転送終了時に割り込み発生 |
| SCI 受信完了割り込み | rxi2_end | SCI ch2 の受信完了割り込み。DTC 指定回数転送終了時に割り込み発生 |

（2） 引数の説明

**表2.37**に、本タスク例で使用する引数を示します。

表 2.37 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| Txd_data[0]～[2] | クロック同期式シリアル送信データの格納 | メインルーチン | 1 バイト | 出力 |
| Rxd_data[0]～[2] | クロック同期式シリアル受信データの格納 | メインルーチン | 1 バイト | 入力 |

（3） 使用内部レジスタ説明

**表2.38**～**表2.41**に、本タスク例の使用する内部レジスタを示します。

表 2.38 使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ設定する | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| | MSTP18 | モジュールスタンバイコントロールレジスタ 2<br>シリアルコミュニケーションインタフェース 2 スタンバイ制御ビット<br>:MSTP18=0 のとき、モジュールスタンバイ解除 | H'FFFF861C<br>ビット 2 | 0 |
| P_INTC.IPRI | SCI2 | インタラプトプライオリティレジスタ I（IPRI）<br>SCI2 の各割り込み（ERI、RXI、TXI、TEI）の割り込み優先レベルの設定<br>:SCI2=b'1010(10)のとき、割り込みは、割り込み優先レベル 10 に設定 | H'FFFF835C<br>ビット<br>12～15 | 10 |

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_1.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。シリアル送信用 | 内蔵 RAM に配置 | H'8000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'00 のとき、DTDAR は固定 | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマル転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、ディストネーション側がブロック領域<br>ノーマルモードでは未使用 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |
| DTC_1.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_1.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ<br>送信データ格納領域の先頭アドレスに設定 | 内蔵 RAM に配置 | Txd_data |
| DTC_1.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ<br>シリアルの送信データレジスタ（TDR_2）に設定 | 内蔵 RAM に配置 | &P_SCI2.TDR |

表 2.39　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_2.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。シリアル受信用 | 内蔵 RAM に配置 | H'2000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'00 のとき、DTSAR は固定 | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマル転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、ディストネーション側がブロック領域<br>ノーマルモードでは未使用 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |
| DTC_2.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'03 |
| DTC_2.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ<br>シリアルの受信データレジスタ（RDR_2）に設定 | 内蔵 RAM に配置 | &P_SCI2.RDR |
| DTC_2.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ<br>受信データ格納領域の先頭アドレスに設定 | 内蔵 RAM に配置 | Rxd_data |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTEE | TXI_2 | DTC イネーブルレジスタ E（DTEE）<br>：TXI_2(DTEE2)=b'1 のとき、SCI2 送信終了割り込み（TXI_2）が起動要因 | H'FFFF8710<br>ビット 2 | 1 |
| | RXI_2 | DTC イネーブルレジスタ E（DTEE）<br>：RXI_2(DTEE3)=b'1 のとき、SCI2 受信終了割り込み（RXI_2）が起動要因 | H'FFFF8710<br>ビット 3 | 1 |

表 2.40　使用内蔵レジスタ説明（3）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_SCI2.SCR.BYTE | | シリアルコントロールレジスタ_2（SCR_2）<br>送受信制御と割り込み制御、送受信クロックソースの選択 | H'FFFF81C2 | H'f0 |
| | TIE | トランスミットインタラプトイネーブル<br>：TIE=1 のとき、TXI 割り込み要求がイネーブル | ビット 7 | |
| | RIE | レシーブインタラプトイネーブル<br>：RIE=1 のとき、RXI および ERI 割り込み要求がイネーブル | ビット 6 | |
| | TE | トランスミットイネーブル<br>：TE=1 のとき、送信動作が可能 | ビット 5 | |
| | RE | レシーブイネーブル<br>：RE=1 のとき、受信動作が可能 | ビット 4 | |
| | MPIE | マルチプロセッサインタラプトイネーブル<br>（調歩同期式モードで SMR の MP＝1 のとき有効）<br>本タスクでは MP=0 のため設定は無効 | ビット 3 | |
| | TEIE | トランスミットエンドインタラプトイネーブル<br>：TEIE=0 のとき、TEI 割り込み要求は禁止 | ビット 2 | |
| | CKE1<br>CKE0 | クロックイネーブル 1～0<br>クロックソースおよび SCK 端子の機能を選択<br>：CKE1[1:0]=b'00 のとき、クロックソースは内部クロック、SCK 端子は入力端子（入力端子は無視） | ビット 1<br>ビット 0 | |
| P_SCI2.SMR.BYTE | | シリアルモードレジスタ_2（SMR_2）<br>通信フォーマットと内蔵ボーレートジェネレータのクロックを選択 | H'FFFF81C0 | H'20 |
| | C/A | コミュニケーションモード<br>：C/A=0 のとき、調歩同期式モードで動作 | ビット 7 | |
| | CHR | キャラクタレングス（調歩同期式モードのみ有効）<br>：CHR=0 のとき、データ長 8 ビットで送受信 | ビット 6 | |
| | PE | パリティイネーブル（調歩同期式モードのみ有効）<br>：PE=1 のとき、送信時はパリティビットを付加し、受信時はパリティチェックを行う | ビット 5 | |
| | O/E | パリティモード（調歩同期式モードで PE＝1 のときのみ有効）<br>：O/E=0 のとき、偶数パリティで送受信 | ビット 4 | |
| | STOP | ストップビットレングス（調歩同期式モードのみ有効）<br>送信時のストップビットの長さを選択<br>：STOP=0 のとき、1 ストップビット | ビット 3 | |
| | MP | マルチプロセッサモード（調歩同期式モードのみ有効）<br>MP=0 のとき、マルチプロセッサ通信機能は無効 | ビット 2 | |
| | CKS1<br>CKS0 | クロックセレクト 1～0<br>内蔵ボーレートジェネレータのクロックソースを選択<br>：CKS[1:0]=b'00 のとき、Pφ/1 クロック（n=0）に設定 | ビット 1<br>ビット 0 | |

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値<br>ビット |
|---|---|---|---|---|
| P_SCI2.SDCR | DIR | シリアルディレクションコントロールレジスタ_2（SDCR_2）<br>データトランスファディレクション<br>：DIR=0 のとき、TDR の内容を LSB ファーストで送信受信データを LSB ファーストとして RDR に格納 | H'FFFF81C6<br>ビット 3 | 1 |
| P_SCI2.BRR | | ビットレートレジスタ_2（BRR_2）<br>：BRR_2=21 のとき、ビットレートは約 57600bps（クロックソース Pφ/1、Pφ=40MHz の場合） | H'FFFF81C1 | 9 |
| P_SCI2.TDR | | トランスミットデータレジスタ_2（TDR_2）<br>TDR は送信データを格納するための 8 ビットのレジスタ | H'FFFF81C3 | |
| P_SCI2.RDR | | レシーブデータレジスタ_2（RDR_2）<br>TDR は送信データを格納するための 8 ビットのレジスタ | H'FFFF81C5 | |
| P_SCI2.SSR | TDRE | シリアルステータスレジスタ_2（SSR_2）<br>トランスミットデータレジスタエンプティフラグ | H'FFFF81C4<br>ビット 7 | 1 |
| | RDRF | シリアルステータスレジスタ_2（SSR_2）<br>レシーブデータレジスタフルフラグ | H'FFFF81C4<br>ビット 6 | 0 |
| | ORER | シリアルステータスレジスタ_2（SSR_2）<br>オーバランエラーフラグ | H'FFFF81C4<br>ビット 5 | 0 |
| | FER | シリアルステータスレジスタ_2（SSR_2）<br>フレミングエラーフラグ | H'FFFF81C4<br>ビット 4 | 0 |
| | PER | シリアルステータスレジスタ_2（SSR_2）<br>パリティエラーフラグ | H'FFFF81C4<br>ビット 3 | 0 |

表 2.41　使用内蔵レジスタ説明（4）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値<br>ビット |
| --- | --- | --- | --- | --- |
| P_PORTA.PACRL3 | PA0MD2 | ポート A コントロールレジスタ L3<br>ポート A コントロールレジスタ L2 | H'FFFF838A<br>ビット 0 | b'1 |
| P_PORTA.PACRL2 | PA0MD[1]<br>PA0MD[0] | PA0 モードビット、PA0/A0/POE0/RXD2 端子の機能を選択<br>：(PA0MD2,PA0MD[1],PA0MD[0])=b'110 のとき、端子機能は RXD2 入力（SCI）に設定 | H'FFFF838E<br>ビット 1<br>ビット 0 | b'10 |
| P_PORTA.PACRL3 | PA1MD2 | ポート A コントロールレジスタ L3<br>ポート A コントロールレジスタ L2 | H'FFFF838A<br>ビット 1 | b'1 |
| P_PORTA.PACRL2 | PA1MD[1]<br>PA1MD[0] | PA1 モードビット、PA1/A1/POE1/TXD2 端子の機能を選択<br>：(PA1MD2,PA1MD[1],PA1MD[0])=b'110 のとき、端子機能は TXD2 出力（SCI）に設定 | H'FFFF838E<br>ビット 3<br>ビット 2 | b'10 |

（4）　使用RAM説明

表2.42に、本タスクで使用するRAMの説明を示します。

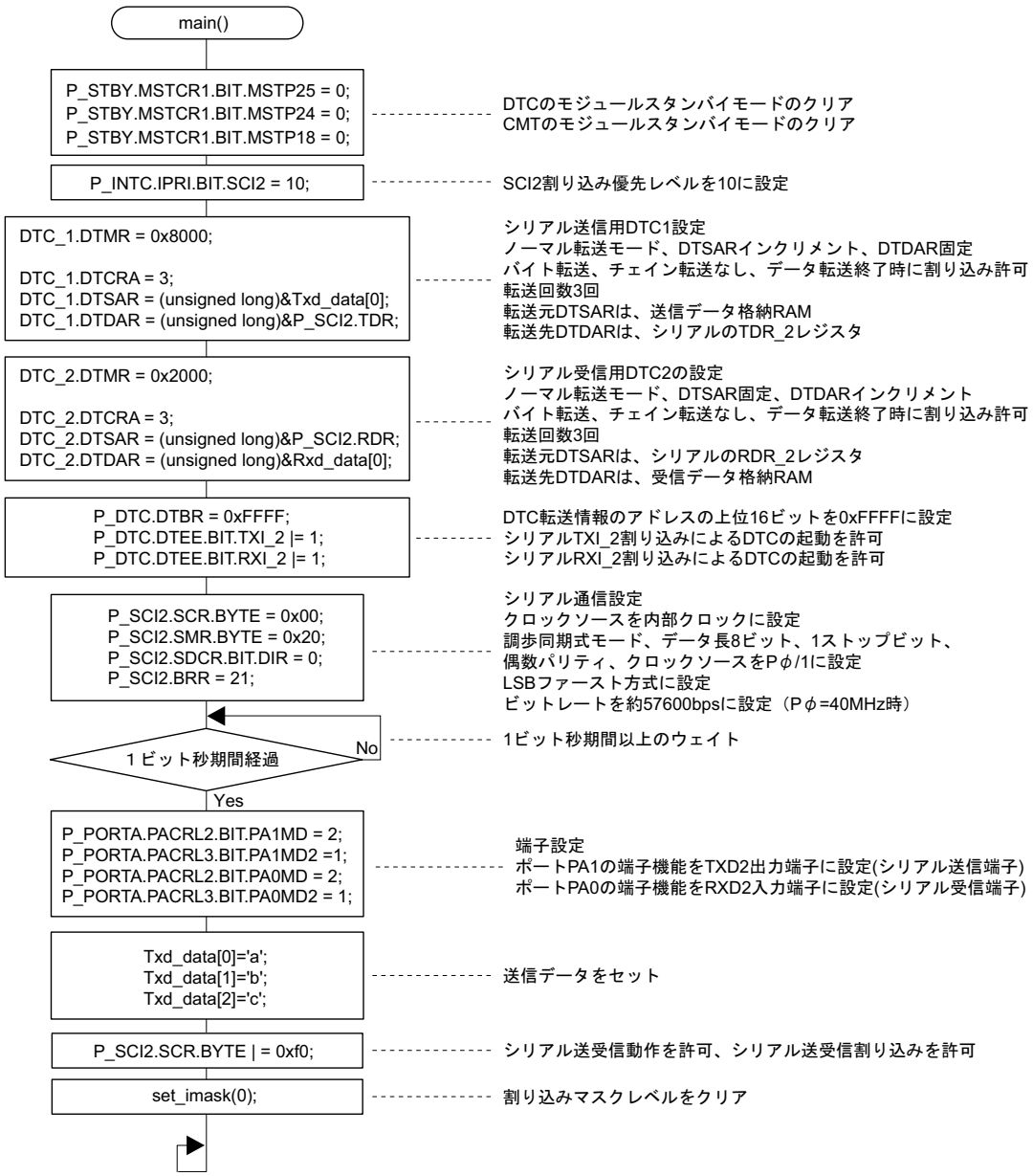
表 2.42　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
| --- | --- | --- | --- |
| Txd_data[0] | クロック同期式シリアル送信データの 1 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Txd_data[1] | クロック同期式シリアル送信データの 2 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Txd_data[2] | クロック同期式シリアル送信データの 3 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[0] | クロック同期式シリアル受信データの 1 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[1] | クロック同期式シリアル受信データの 2 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[2] | クロック同期式シリアル受信データの 3 バイト目を格納 | 内蔵 RAM | メインルーチン |

| 調歩同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

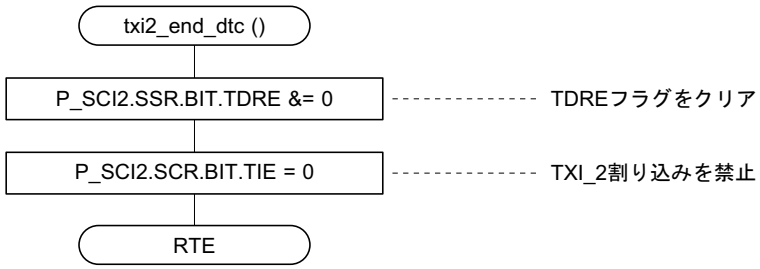
フローチャート

(a)　メイン処理

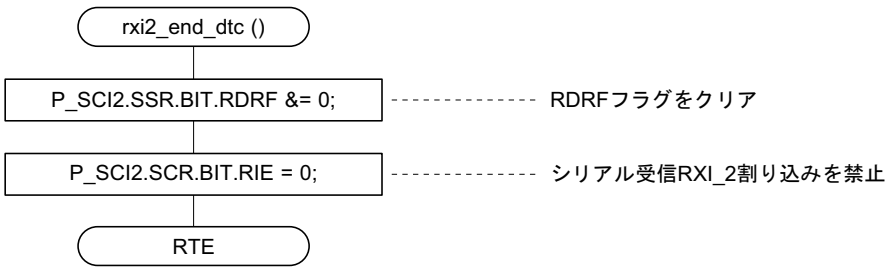


フローチャート

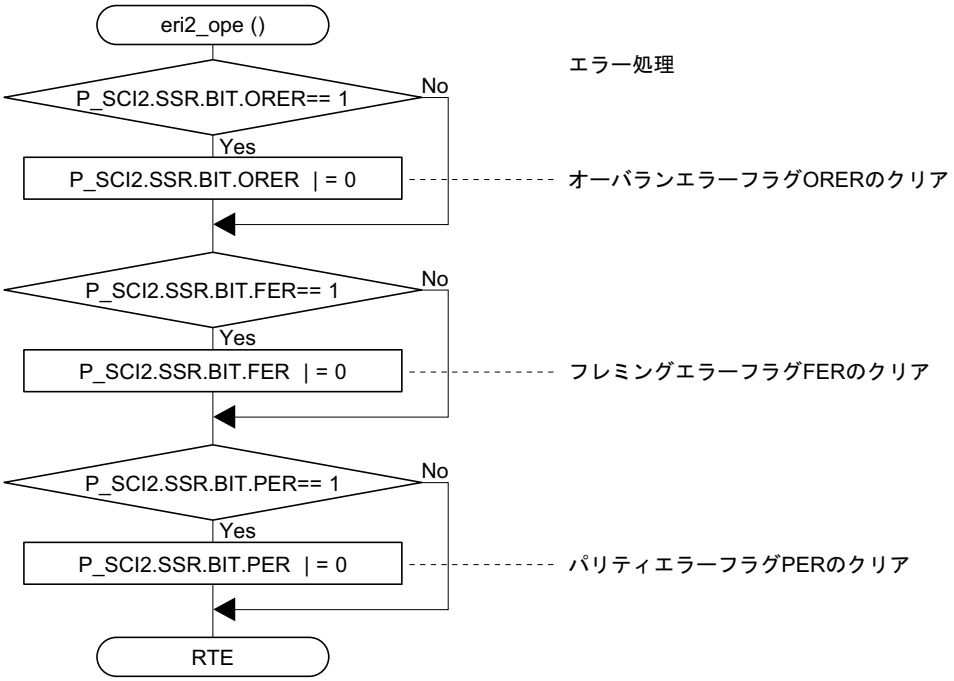
(b)　シリアル送信TXI_2割り込み処理




(c)　シリアル受信RXI_2割り込み処理




(d)　シリアルエラー割り込み処理



プログラムリスト

```
/******************************************************/
/* SH7046F Series -SH7047- Application Note           */
/* Synchronous Serial Data Transmission with DTC      */
/*                                                    */
/* Function                                           */
/* :Serial Communication Interface(SCI)               */
/*  Asynchronous Serial Mode                          */
/*  -Transmitting/Receiving-                          */
/* :Data Transfer Controller(DTC)                     */
/*                                                    */
/* External input clock    :10MHz                     */
/* Internal CPU clock    :40MHz                       */
/* Internal peripheral clock :40MHz                   */
/*                                                    */
/* Written   :  2002/3/01  Rev.1.0                    */
/******************************************************/

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

/*------------ Symbol Definition ------------------------*/
struct st_dtc_tn {            /* DTC Normal Transfer information  */
    unsigned short DTMR;      /* DTC Mode Register            */
    unsigned short DTCRA;     /* transfer counter             */
    unsigned short dummy1;    /* Reserved                     */
    unsigned short dummy2;    /* Reserved                     */
    unsigned long DTSAR;      /* source address register      */
    unsigned long DTDAR;      /* destination address register */
};

#define DTC_COUNT 3               /* DTC Transmit count  */
#define DTC_1 (*(volatile struct st_dtc_tn *)0xFFFFE000) /* Transmit DTC     */
#define DTC_2 (*(volatile struct st_dtc_tn *)0xFFFFE010) /* Receive DTC      */
```

プログラムリスト

```
/*------------ RAM allocation Definition  --------------------*/
volatile unsigned char Txd_data[DTC_COUNT];  /* Transmit data  */
volatile unsigned char Rxd_data[DTC_COUNT];/* Receive data   */


/*------------ Function Definition -------------------------*/
void main(void);
void txi2_end(void);
void rxi2_end(void);


/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
     unsigned long i;

     /* Set standby mode  */
     P_STBY.MSTCR1.BIT.MSTP25 = 0;      /* Disable DTC standby mode   */
     P_STBY.MSTCR1.BIT.MSTP24 = 0;
     P_STBY.MSTCR1.BIT.MSTP18 = 0;      /* Disable SCI2 standby mode*/

     /* Set interrupt priority level (0 to 15) */
     P_INTC.IPRI.BIT.SCI2 = 10;       /* SCI2 interrupt level 10   */
     /* SIC2 Transmit DTC information     */
     DTC_1.DTMR = 0x8000;
            /* SM[1:0]=b'10;  DTSAR is incremented               */
            /* DM[1:0]=0;   DTDAR is  fixed                      */
            /* MD[1:0]=0;   Transfer mode :Normal mode           */
            /* SZ[1:0]=0;   Byte-size transfer                   */
            /* DTS=0;     destination is block area(no use)      */
            /* CHNE=0;    Chain transfer is canceled             */
            /* DISEL=0;     Interrupt->transfer ends             */
            /* NMIM=0;    NMI->Terminate DTC transfer            */
     DTC_1.DTCRA = DTC_COUNT;      /* Transfer Count             */
```

プログラムリスト

```
DTC_1.DTSAR = (unsigned long)&Txd_data[0];/* set SCI2 Transmit data    */
DTC_1.DTDAR = (unsigned long)&P_SCI2.TDR; /* set SCI2 TDR register   */

/* SIC2 Receive DTC information      */
DTC_2.DTMR = 0x2000;
       /* SM[1:0]=0;   DTSAR is fixed                        */
       /* DM[1:0]=b'10;  DTDAR is incremented                */
       /* MD[1:0]=0;   Transfer mode :Normal mode            */
       /* SZ[1:0]=0;   Byte-size transfer                    */
       /* DTS=0;     destination is block area(no use)       */
       /* CHNE=0;    Chain transfer is canceled              */
       /* DISEL=0;     Interrupt->transfer ends              */
       /* NMIM=0;    NMI->Terminate DTC transfer             */
DTC_2.DTCRA = DTC_COUNT;      /* Transfer Count              */
DTC_2.DTSAR = (unsigned long)&P_SCI2.RDR; /* set SCI2 RDR register   */
DTC_2.DTDAR = (unsigned long)&Rxd_data[0];/* set SCI2 Receive Buffer  */

P_DTC.DTBR= 0xFFFF;           /* information base register */
/* DTC Transmit enable       */
P_DTC.DTEE.BIT.TXI_2 |= 1;      /* interrupt sources: TXI_2(SCI2) */
P_DTC.DTEE.BIT.RXI_2 |= 1;      /* interrupt sources: RXI_2(SCI2) */

/* Initialize SCI2 clocked synchronous mode */
P_SCI2.SCR.BYTE = 0x00;
       /* TIE=0;     clear TIE                */
       /* RIE=0;     clear RIE                */
       /* TE=0;      clear TE                 */
       /* RE=0;      clear RE                 */
       /* MPIE=0;      clear MPIE,TEIE        */
       /* TEIE=0;      clear TEIE             */
       /* CKE[1:0]=b'00;    clock source: internal ,SCK :input  */
P_SCI2.SMR.BYTE = 0x20;
       /* CA=0;      Asynchronous mode        */
       /* CHR=0;       data length 8bits      */
       /* PE=1;      parity enable            */
       /* OE=0;      even parity              */
```

プログラムリスト

```
            /* STOP=0;      1 stop bit             */
            /* MP=0;      disable Multiprocessor Mode          */
            /* CKS[1:0]=b'00;   clock source =Pφ/1             */
        P_SCI2.SDCR.BIT.DIR = 0;      /* LSB first send          */
        P_SCI2.BRR = 21;          /* 57600bps@ Pφ=40MHz          */
        for( i=0; i < 0x500 ; i++);      /* Wait 1bit over       */

        /* Initialize SCI2 port        */
        P_PORTA.PACRL2.BIT.PA1MD = 2;    /* set TXD2(PA1:73pin@SH7047)   */
        P_PORTA.PACRL3.BIT.PA1MD2 =1;
        P_PORTA.PACRL2.BIT.PA0MD = 2;    /* set RXD2(PA0:75pin@SH7047)   */
        P_PORTA.PACRL3.BIT.PA0MD2 = 1;
        /* set transmit data */
        Txd_data[0] = 'a';
        Txd_data[1] = 'b';
        Txd_data[2] = 'c';

        P_SCI2.SCR.BYTE |= 0xf0;       /* Transmit/Receive Enable      */
                /* TIE=1;     TXI_2 interrupt Enable        */
                /* RIE=1;     RXI_2,ERI_2 interrupt Enable  */
                /* TE=1;      Transmit Enable               */
                /* RE=1;      Receive Enable                */

        set_imask(0);           /* clear interrupt mask level   */

        while(1);

    }

    /*********************************************************/
    /* SIC2:TXI_2 Interrupt                                  */
    /* Transmission of DTC data transfer termination         */
    /*********************************************************/
    #pragma interrupt(txi2_end)
    void txi2_end(void)
    {
```

プログラムリスト

```
    P_SCI2.SSR.BIT.TDRE &= 0;     /* TDRE=0 flag clear          */
    P_SCI2.SCR.BIT.TIE = 0;       /* TXI_2 interrupt disable    */
}

/************************************************************/
/* SIC2 RXI_2 Interrupt                                     */
/* Reception of DTC data transfer termination               */
/************************************************************/
#pragma interrupt(rxi2_end)
void rxi2_end(void)
{
    P_SCI2.SSR.BIT.RDRF &= 0;     /* RDRF=0 flag clear              */
    P_SCI2.SCR.BIT.RIE = 0;       /* RXI_2,ERI_2 interrupt disable  */
}

/************************************************************/
/* SIC2:ERI_2 Interrupt                                     */
/* SCI Reception Error                                      */
/************************************************************/
#pragma interrupt(eri2_ope)
void eri2_ope(void)
{
    if(P_SCI2.SSR.BIT.ORER==1){   /*  Overrun Error     */
      P_SCI2.SSR.BIT.ORER |= 0;   /*  ORER=0 flag clear */
    }
    if(P_SCI2.SSR.BIT.FER==1){    /*  Framing Error     */
      P_SCI2.SSR.BIT.FER |= 0;    /*  FER=0 flag clear    */
    }
    if(P_SCI2.SSR.BIT.PER==1){    /*  Parity Error      */
      P_SCI2.SSR.BIT.PER |= 0;    /*  PER=0 flag clear    */
    }

}
```

## 2.6　クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC）

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

仕様

（1）　図2.31に示すように、クロック同期式シリアル転送機能と、DTCによるデータ転送機能を使用してシリアルデータの同時送受信動作を行います。

（2）　図2.32に示すようにDTC（Data Transfer Controller）の転送機能を使用して、シリアル送信データの転送、シリアル受信データの内蔵RAMへの格納を行います。

（3）　送受信するデータのデータ長は8ビット固定、データの最下位ビットから送受信するLSBファースト方式により、ビットレート1Mbpsで通信を行います。

（4）　DTCの転送条件を表2.43に示します。

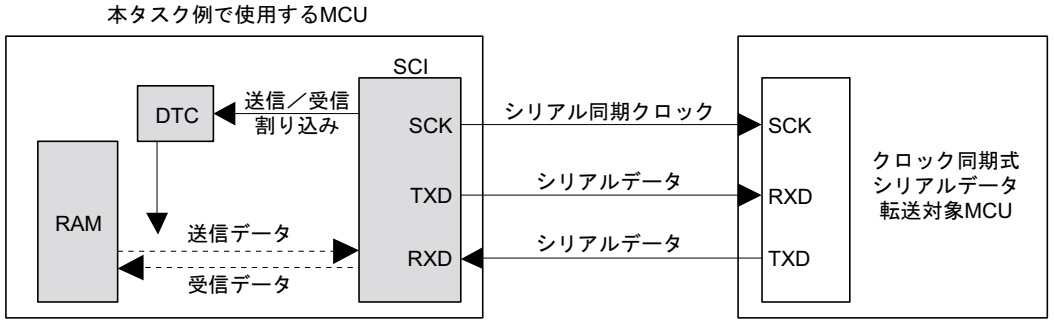



図 2.31　クロック同期式シリアルデータ同時送受信

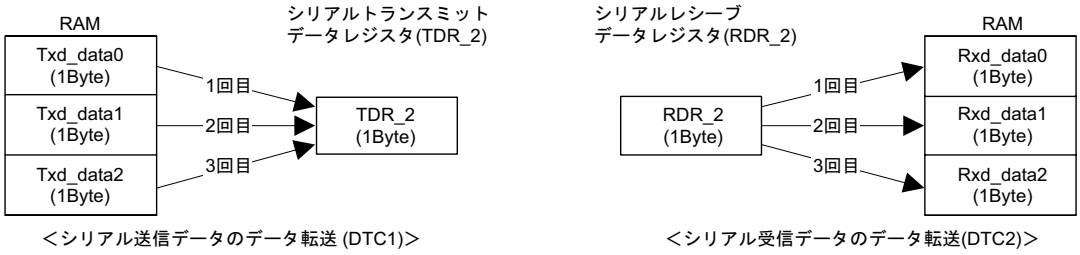



図 2.32　DTC を用いたデータ転送

表 2.43　DTC 転送条件

| 条件 | シリアル送信側 DTC の転送条件（DTC1） | シリアル受信側 DTC の転送条件（DTC2） |
|---|---|---|
| 転送モード | ノーマルモード | ノーマルモード |
| 転送回数 | 3 回 | 3 回 |
| 転送サイズ | バイト（Byte）転送 | バイト（Byte）転送 |
| 転送元 | 内蔵 RAM | シリアルレシーブデータレジスタ（RDR_2） |
| 転送先 | シリアルトランスミットデータレジスタ（TDR_2） | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント | 転送元アドレスは固定 |
| 転送先アドレス | 転送先アドレスは固定 | 転送後に転送先アドレスをインクリメント |
| 起動要因 | SCI ch2 の送信割り込み（TXI_2）で起動 | SCI ch2 の受信割り込み（RXI_2）で起動 |
| 割り込み処理 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

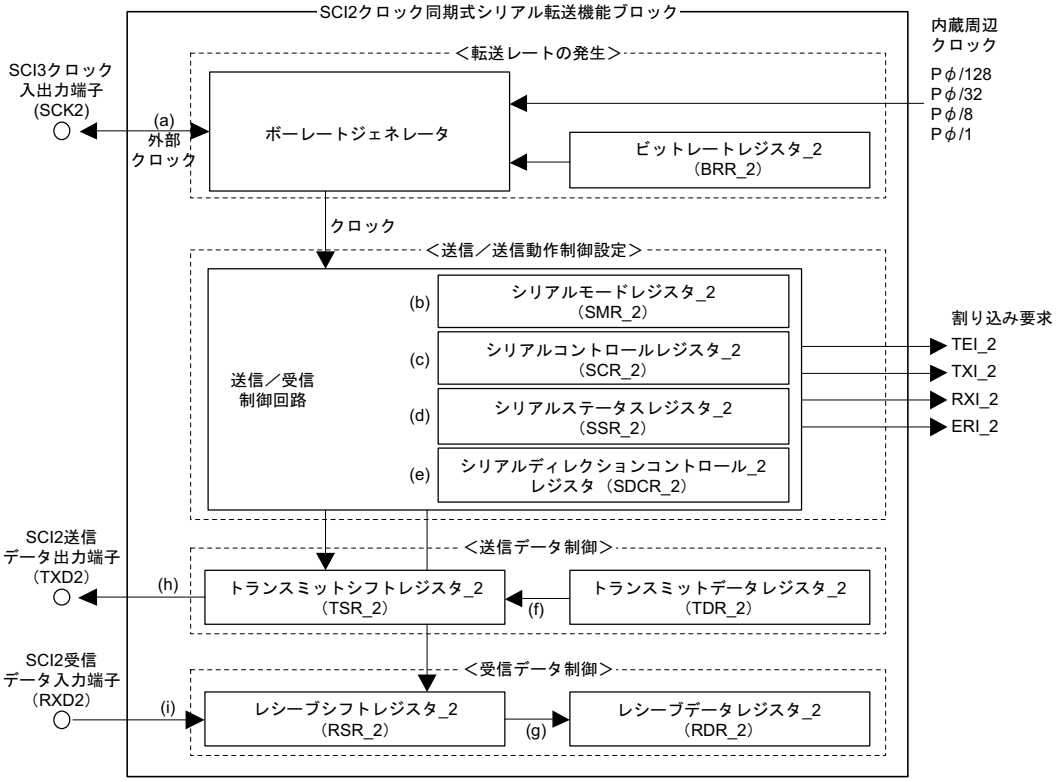
| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

**使用機能説明**

（a） 本タスク例では、シリアルコミュニケーションインタフェース（SCI：Serial Communication Interface）とデータトランスファコントローラ（DTC:Data Transfer Controller）を使用して、クロック同期式のシリアルデータの同時送受信動作を行います。

（b） 図2.33にクロック同期式シリアルデータ送受信同時動作のブロック図を示します。以下にクロック同期式シリアルデータ同時送受信のブロック図について説明します。

- 内蔵周辺クロック Pφは、内蔵周辺機能を動作させるための基準クロックです。
- クロック同期式モードではデータ長は 8 ビット固定になります。
- レシーブシフトレジスタ（RSR_2）は、シリアルデータを受信するためのレジスタです。RSR_2 に RXD2 端子から入力されたシリアルデータを LSB（ビット 0）から受信した順にセットし、パラレルデータに変換します。1 バイトのデータを受信すると、データは自動的に RDR_2 へ転送されます。CPU から RSR_2 を直接リード／ライトすることはできません。
- レシーブデータレジスタ_2（RDR_2）は、受信したシリアルデータを格納する 8 ビットのレジスタです。1 バイトのデータの受信が終了すると、受信したデータを RSR_2 から RDR_2 へ転送し、受信動作を完了します。その後 RSR_2 は受信可能となります。RSR_2 と RDR_2 はダブルバッファになっているため連続した受信動作が可能です。RDR_2 は受信専用レジスタのため CPU からライトできません。
- トランスミットシフトレジスタ_2（TSR_2）は、シリアルデータを送信するためのレジスタです。TDR_2 から送信データをいったん TSR_2 に転送し、LSB（ビット 0）から順に TXD 端子に送出することでシリアルデータ送信を行います。1 バイトのデータを送信すると、自動的に TDR_2 から TSR_2 へ次の送信データを転送し、送信を開始します。ただし、TDR_2 にデータが書き込まれていない（TDRE に"1"がセットされている）場合には TDR_2 から TSR_2 へのデータ転送は行いません。CPU から TSR_2 を直接リード／ライトすることはできません。
- トランスミットデータレジスタ_2（TDR_2）は、送信データを格納する 8 ビットのレジスタです。TSR_2 の"空"を検出すると、TDR_2 に書き込まれた送信データを TSR_2 に転送し、シリアルデータ送信を開始します。TSR_2 のシリアルデータ送信中に、TDR_2 に次の送信データをライトしておくと、連続送信が可能です。TDR_2 は、常に CPU によるリード／ライトが可能です。
- シリアルモードレジスタ_2（SMR_2）は、シリアルデータ通信フォーマットの設定と、内蔵ボーレートジェネレータのクロックソースを選択するための 8 ビットのレジスタです。
- シリアルコントロールレジスタ_2（SCR_2）は、送信／受信動作および送信／受信クロックソースの選択を行う 8 ビットのレジスタです。
- シリアルステータスレジスタ_2（SSR_2）は、SCI2 のステータスフラグと送受信マルチプロセッサビットで構成されています。TDRE、RDRF、OER、PER、FER はクリアのみ可能です。
- シリアルディレクションコントロール_2 レジスタ（SDCR_2）は、DIR ビットにより、LSB ファースト／MSB ファーストの選択を行います。シリアル通信モードによらず、8 ビット長の場合 LSB ファースト／MSB ファーストの選択が可能です。7 ビット長の場合 LSB ファーストを選択し、MSB ファーストの選択は行わないでください。
- ビットレートレジスタ_2（BRR_2）はビットレートを調整するための 8 ビットのレジスタです。
- 送信データは、転送クロックの立ち下がりから次の立ち下がりまで出力されます。また、受信データは転送クロックの立ち上がりで取り込まれます。




【注】
(a) クロックを出力する。
(b) シリアルデータ通信フォーマットの設定と、ボーレートジェネレータのクロックソースの選択を行う。
(c) 送信／受信動作、クロック同期式モードでのクロック出力端子の選択を行う。
(d) ステータスフラグ（トランスミットデータレジスタエンプティ、レシーブデータレジスタフル、オーバランエラー）によりSCI2の動作状態を示す。
(e) LSBファースト／MSBファーストの選択を行う。
(f) TSR_2の"空"を検出することにより、TDR_2に書き込まれた送信データをTSR_2に転送。
(g) 1バイトのデータの受信が終了すると、受信したデータをRSR_2からRDR_2へ転送。

図 2.33　クロック同期式シリアルデータの送受信ブロック図

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

**使用機能説明**

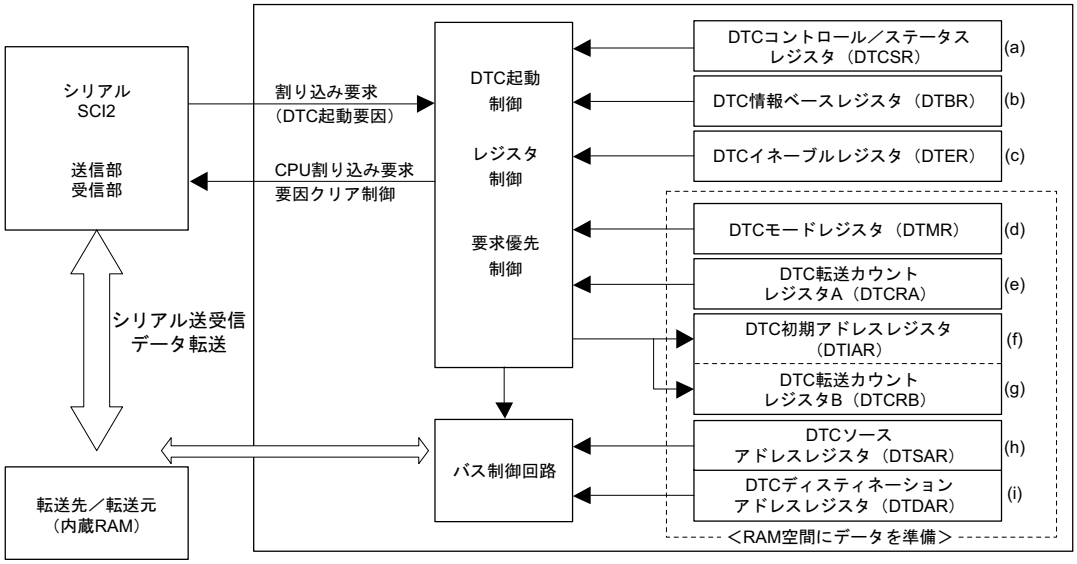
（c） 図2.34にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ノーマル転送モードを使用して、シリアル送受信データの転送を行います。DTC起動要因を、シリアルの送信のTXI割り込み、シリアル受信のRXI割り込みとして、DTCデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

使用機能説明

本タスクでは、シリアル送信データの転送とシリアル受信データの転送に、それぞれDTCノーマルモードでのデータ転送を行います。ノーマルモードのレジスタ情報（DTMR、DTSAR、DTDAR、DTCRA、DTCRB）をシリアル送信用、シリアル受信用と、2つ準備してください。




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.34　DTC ブロック図

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

使用機能説明

(5)　表2.44に本タスク例の機能割り付けを示します。

表 2.44　機能割り付け

| 機能 | 分類 | 機能割り付け |
|---|---|---|
| SCK2 | 端子 | チャネル 2 のクロック出力端子 |
| TXD2 | 端子 | チャネル 2 の送信データ出力端子 |
| RXD2 | 端子 | チャネル 2 の受信データ入力端子 |
| SMR_2 | SCI2 | 通信フォーマットの設定、クロック同期式モードに設定 |
| SCR_2 | SCI2 | 送受信動作、割り込みを許可、SCK2 をクロック出力端子に設定 |
| SSR_2 | SCI2 | SCI2 の動作状態を示すステータスフラグ |
| SDCR_2 | SCI2 | LSB ファーストに設定 |
| BBR_2 | SCI2 | 送受信のビットレートを設定 |
| TSR_2 | SCI2 | シリアルデータを送信するためのレジスタ |
| TDR_2 | SCI2 | 送信データを格納するレジスタ |
| RSR_2 | SCI2 | シリアルデータを受信するためのレジスタ |
| RDR_2 | SCI2 | 受信データを格納するレジスタ |
| DTMR | DTC | DTC をノーマル転送モードに設定 |
| DTCRA | DTC | 転送回数の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTER | DTC | シリアル受信、シリアル送信時に DTC の起動を許可 |

動作説明

（1） 図2.35に動作原理を示します。
図に示すように、ハードウェア処理およびソフトウェア処理によりクロック同期式シリアルデータの同時送受信を行います。

（a） 送信処理
- クロック同期式シリアルで 3 バイトのデータを送信。
- 3 バイトの送信データは DTC を用いて、内蔵 RAM から SIC に転送。
- DTC の起動には、シリアルの TXI_2 割り込みを使用。

（b） 受信処理
- クロック同期式シリアルで 3 バイトのデータを受信。
- 3 バイトの受信データは DTC を用いて、SCI から内蔵 RAM に転送。
- DTC の起動には、シリアルの RXI_2 割り込みを使用。

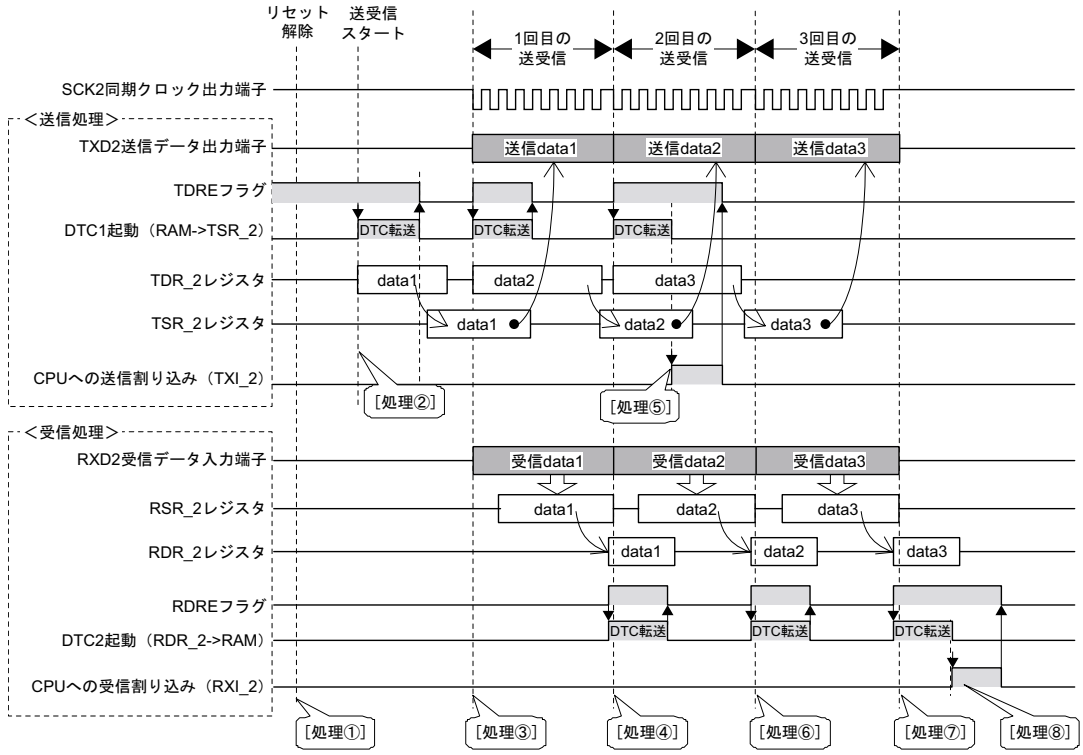



図 2.35　動作原理

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

動作説明

以下に図2.35の説明を示します。

| | シリアル送信側の処理 | シリアル受信側の処理 |
|---|---|---|
| 処理① | 初期設定<br>（1）SCI2 の設定<br>・クロック同期式モードに設定、SCK2 同期クロック出力。LSB ファースト転送<br>・送信割り込み、受信割り込み、送信動作、受信動作を許可<br>（2）DTC 設定<br>・ノーマル転送モードに設定<br>・シリアル送信割り込み（TXI_2）での DTC1 起動を許可、シリアル受信割り込み（RXI_2）での DTC2 起動を許可 | |
| 処理② | <<ハードウェア処理>><br>・TXI_2 割り込みで DTC1 起動（1 回目）<br>・送信データ data1 を内蔵 RAM から TDR_2 レジスタに転送（DTC1）<br>・TDRE をクリア（DTC1） | なし |
| | <<ソフトウェア処理>>なし | |
| 処理③ | <<ハードウェア処理>><br>・TDRE フラグが 0 のとき、TDR_2 から TSR_2 レジスタに送信データ data1 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・送信を開始（SCI2）<br>・TXI_2 割り込みで DTC1 起動（2 回目）<br>・送信データ data2 を内蔵 RAM から TDR_2 に転送（DTC1）<br>・TDRE フラグをクリア（DTC1） | <<ハードウェア処理>><br>・受信スタート<br>・受信データ data1 を RSR レジスタに取り込む |
| | <<ソフトウェア処理>>なし | <<ソフトウェア処理>>なし |
| 処理④ | <<ハードウェア処理>><br>・最終ビットを送り出すとき、TDRE フラグをチェック（SCI2）<br>・TDRE が 0 のとき、TDR_2 から、TSR_2 レジスタに送信データ data2 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・次のフレーム送信を開始（SCI2）<br>・TXI_2 割り込みで DTC1 起動（3 回目）<br>・送信データ data3 を内蔵 RAM から TDR_2 に転送し終了（DTC1）、TDRE フラグはクリアされない | <<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data1 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・次のフレーム受信を開始（SCI2）<br>・受信データ data2 を RSR レジスタに取り込む（SCI2）<br>・RXI_2 割り込みで DTC2 起動（1 回目）<br>・受信データ data1 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2） |
| | <<ソフトウェア処理>>なし | <<ソフトウェア処理>>なし |
| 処理⑤ | <<ハードウェア処理>><br>・CPU に対して TXI_2 割り込みが発生 | なし |
| | <<ソフトウェア処理>><br>・TDRE フラグをクリア<br>・TXI_2 割り込みを禁止 | |

動作説明

| | シリアル送信側の処理 | シリアル受信側の処理 |
|---|---|---|
| 処理⑥ | <<ハードウェア処理>><br>・最終ビットを送り出すとき、TDRE フラグをチェック（SCI2）<br>・TDRE が 0 のとき、TDR_2 から TSR_2 レジスタに送信データ data3 を転送（SCI2）<br>・TDRE フラグを 1 にセット（SCI2）<br>・次のフレーム送信を開始（SCI2） | <<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data2 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・次のフレーム受信を開始<br>・受信データ data3 を RSR レジスタに取り込む<br>・RXI_2 割り込みで DTC2 起動（2 回目）<br>・受信データ data2 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2） |
| | <<ソフトウェア処理>>なし | <<ソフトウェア処理>>なし |
| 処理⑦ | なし | <<ハードウェア処理>><br>・RSR から RDR レジスタに受信データ data3 を転送（SCI2）<br>・RDRE フラグを 1 にセット（SCI2）<br>・RXI_2 割り込みで DTC2 起動（2 回目）<br>・受信データ data2 を RDR_2 から内蔵 RAM に転送（DTC2）<br>・RDRE フラグをクリア（DTC2） |
| | | <<ソフトウェア処理>>なし |
| 処理⑧ | なし | <<ハードウェア処理>><br>・RDRE フラグが 1 にセットされた状態のため、CPU に対して TXI_2 割り込みが発生 |
| | | <<ソフトウェア処理>><br>・TDRE フラグをクリア<br>・TXI_2 割り込みを禁止 |

（2） 図2.36にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込みデータ転送を行います。

動作説明

本タスクでは、シリアル送信のデータ転送の場合、TXI_2割り込みが起動要因となり、シリアル受信のデータ転送の場合、RXI_2割り込みが起動要因となります。
表2.45にノーマルモード時のレジスタ情報の構成を示します。

表 2.45　DTC レジスタ情報一覧（ノーマルモード）

| 設定アドレス | レジスタ名 | データ長 |
|---|---|---|
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

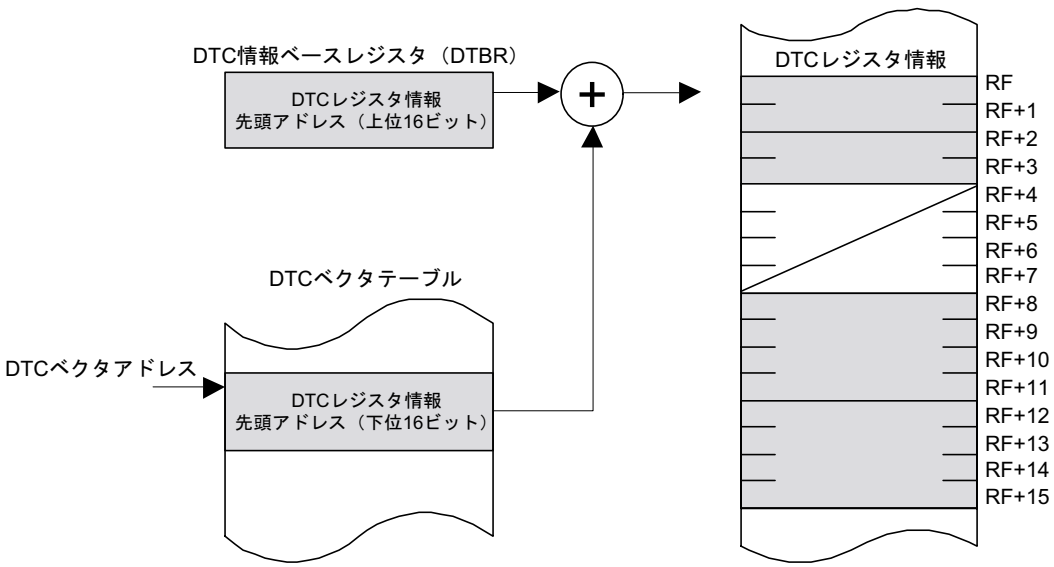



図 2.36　DTC ベクタアドレスと転送情報との対応

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

表2.46に、本タスク例のモジュールを示します。

表 2.46 モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | SCI ch2 クロック同期式シリアル通信、DTC の初期設定、シリアル通信をスタート |
| SCI 送信完了割り込み | txi2_end | SCI ch2 の送信完了割り込み。DTC 指定回数転送終了時に割り込み発生 |
| SCI 受信完了割り込み | rxi2_end | SCI ch2 の受信完了割り込み。DTC 指定回数転送終了時に割り込み発生 |

（2） 引数の説明

表2.47に、本タスク例で使用する引数を示します。

表 2.47 引数の説明

| 引数名 | 機能 | モジュール名 | データ長 | 入出力 |
|---|---|---|---|---|
| Txd_data[0]～[2] | クロック同期式シリアル送信データの格納 | メインルーチン | 1 バイト | 出力 |
| Rxd_data[0]～[2] | クロック同期式シリアル受信データの格納 | メインルーチン | 1 バイト | 入力 |

（3） 使用内部レジスタ説明

表2.48～表2.51に、本タスク例の使用する内部レジスタを示します。

表 2.48 使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値<br>ビット |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ設定する | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| | MSTP18 | モジュールスタンバイコントロールレジスタ 2<br>シリアルコミュニケーションインタフェース 2 スタンバイ制御ビット<br>:MSTP18=0 のとき、モジュールスタンバイ解除 | H'FFFF861C<br>ビット 2 | 0 |
| P_INTC.IPRI | SCI2 | インタラプトプライオリティレジスタ I（IPRI）<br>SCI2 の各割り込み（ERI、RXI、TXI、TEI）の割り込み優先レベルの設定<br>:SCI2=b'1010(10)のとき、割り込みは、割り込み優先レベル 10 に設定 | H'FFFF835C<br>ビット<br>12～15 | 10 |

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_1.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。シリアル送信用 | 内蔵 RAM に配置 | H'8000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'00 のとき、DTDAR は固定 | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマル転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、ディスティネーション側がブロック領域<br>ノーマルモードでは未使用 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |
| DTC_1.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_1.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ<br>送信データ格納領域の先頭アドレスに設定 | 内蔵 RAM に配置 | Txd_data |
| DTC_1.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ<br>シリアルの送信データレジスタ（TDR_2）に設定 | 内蔵 RAM に配置 | &P_SCI2.TDR |

表 2.49　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_2.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。シリアル受信用 | 内蔵 RAM に配置 | H'2000 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'00 のとき、DTSAR は固定 | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'00 のとき、ノーマル転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'00 のとき、バイト（1byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'0 のとき、ディスティネーション側がブロック領域<br>ノーマルモードでは未使用 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |
| DTC_2.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'0003 |
| DTC_2.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ<br>シリアルの受信データレジスタ（RDR_2）に設定 | 内蔵 RAM に配置 | &P_SCI2.RDR |
| DTC_2.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ<br>受信データ格納領域の先頭アドレスに設定 | 内蔵 RAM に配置 | Rxd_data |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTEE | TXI_2 | DTC イネーブルレジスタ E（DTEE）<br>：TXI_2(DTEE2)=b'1 のとき、SCI2 送信終了割り込み（TXI_2）が起動要因 | H'FFFF8710<br>ビット 2 | 1 |
| | RXI_2 | DTC イネーブルレジスタ E（DTEE）<br>：RXI_2(DTEE3)=b'1 のとき、SCI2 受信終了割り込み（RXI_2）が起動要因 | H'FFFF8710<br>ビット 3 | 1 |

表 2.50　使用内蔵レジスタ説明（3）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_SCI2.SCR.BYTE | | シリアルコントロールレジスタ_2（SCR_2）<br>送受信制御と割り込み制御、送受信クロックソースの選択 | H'FFFF81C2 | H'f1 |
| | TIE | トランスミットインタラプトイネーブル<br>：TIE=1 のとき、TXI 割り込み要求がイネーブル | ビット 7 | |
| | RIE | レシーブインタラプトイネーブル<br>：RIE=1 のとき、RXI および ERI 割り込み要求がイネーブル | ビット 6 | |
| | TE | トランスミットイネーブル<br>：TE=1 のとき、送信動作が可能 | ビット 5 | |
| | RE | レシーブイネーブル<br>：RE=1 のとき、受信動作が可能 | ビット 4 | |
| | MPIE | マルチプロセッサインタラプトイネーブル<br>（調歩同期式モードで SMR の MP＝1 のとき有効）<br>本タスクでは設定は無効 | ビット 3 | |
| | TEIE | トランスミットエンドインタラプトイネーブル<br>：TEIE=0 のとき、TEI 割り込み要求は禁止 | ビット 2 | |
| | CKE1<br>CKE0 | クロックイネーブル 1〜0<br>クロックソースおよび SCK 端子の機能を選択<br>：CKE1[1:0]=b'01 のとき、クロックソースは内部クロック、SCK 端子は同期クロック出力 | ビット 1<br>ビット 0 | |
| P_SCI2.SMR.BYTE | | シリアルモードレジスタ_2（SMR_2）<br>通信フォーマットと内蔵ボーレートジェネレータのクロックを選択 | H'FFFF81C0 | H'80 |
| | C/A | コミュニケーションモード<br>：C/A=1 のとき、クロック同期式モードで動作 | ビット 7 | |
| | CHR | キャラクタレングス（調歩同期式モードのみ有効）<br>：CHR=0 のとき、データ長 8 ビットで送受信。<br>クロック同期式モードではデータ長は 8 ビット固定<br>本タスクでは設定は無効 | ビット 6 | |
| | PE | パリティイネーブル（調歩同期式モードのみ有効）<br>：PE=1 のとき、送信時はパリティビットを付加し、受信時はパリティチェックを行う。本タスクでは設定は無効 | ビット 5 | |
| | O/E | パリティモード（調歩同期式モードで PE＝1 のときのみ有効）<br>：O/E=0 のとき、偶数パリティで送受信。本タスクでは設定は無効 | ビット 4 | |
| | STOP | ストップビットレングス（調歩同期式モードのみ有効）<br>送信時のストップビットの長さを選択<br>：STOP=0 のとき、1 ストップビット。本タスクでは設定は無効 | ビット 3 | |
| | MP | マルチプロセッサモード（調歩同期式モードのみ有効）<br>MP=1 のとき、マルチプロセッサ通信機能がイネーブル。<br>本タスクでは設定は無効 | ビット 2 | |
| | CKS1<br>CKS0 | クロックセレクト 1〜0<br>内蔵ボーレートジェネレータのクロックソースを選択<br>：CKS[1:0]=b'00 のとき、Pφ/1 クロック（n=0）に設定 | ビット 1<br>ビット 0 | |

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_SCI2.SDCR | DIR | シリアルディレクションコントロールレジスタ_2（SDCR_2）<br>データトランスファディレクション<br>：DIR=0 のとき、TDR の内容を LSB ファーストで送信受信データを LSB ファーストとして RDR に格納 | H'FFFF81C6<br>ビット 3 | 1 |
| P_SCI2.BRR | | ビットレートレジスタ_2（BRR_2）<br>ビットレートを調整するための 8 ビットのレジスタ<br>：BRR_2=9 のとき、ビットレートは 1Mbps（クロックソース Pφ/1、Pφ=40MHz の場合） | H'FFFF81C1 | 9 |
| P_SCI2.TDR | | トランスミットデータレジスタ_2（TDR_2）<br>TDR は送信データを格納するための 8 ビットのレジスタ。 | H'FFFF81C3 | |
| P_SCI2.RDR | | レシーブデータレジスタ_2（RDR_2）<br>TDR は送信データを格納するための 8 ビットのレジスタ。 | H'FFFF81C5 | |
| P_SCI2.SSR | TDRE | シリアルステータスレジスタ_2（SSR_2）<br>トランスミットデータレジスタエンプティ | H'FFFF81C4<br>ビット 7 | 1 |
| | RDRF | シリアルステータスレジスタ_2（SSR_2）<br>レシーブデータレジスタフル | H'FFFF81C4<br>ビット 6 | 0 |
| | ORER | シリアルステータスレジスタ_2（SSR_2）<br>オーバランエラー | H'FFFF81C4<br>ビット 5 | 0 |

表 2.51　使用内蔵レジスタ説明（3）

| レジスタ名 | | 機能 | アドレス | 設定値 |
| --- | --- | --- | --- | --- |
| | ビット | | ビット | |
| P_PORTA.PACRL3 | PA0MD2 | ポート A コントロールレジスタ L3<br>ポート A コントロールレジスタ L2 | H'FFFF838A<br>ビット 0 | b'1 |
| P_PORTA.PACRL2 | PA0MD[1]<br>PA0MD[0] | PA0 モードビット、PA0/A0/POE0/RXD2 端子の機能を選択<br>：（PA0MD2,PA0MD[1],PA0MD[0])=b'110 のとき、端子機能は RXD2 入力（SCI）に設定 | H'FFFF838E<br>ビット 1<br>ビット 0 | b'10 |
| P_PORTA.PACRL3 | PA1MD2 | ポート A コントロールレジスタ L3<br>ポート A コントロールレジスタ L2 | H'FFFF838A<br>ビット 1 | b'1 |
| P_PORTA.PACRL2 | PA1MD[1]<br>PA1MD[0] | PA1 モードビット、PA1/A1/POE1/TXD2 端子の機能を選択<br>：（PA1MD2,PA1MD[1],PA1MD[0])=b'110 のとき、端子機能は TXD2 出力（SCI）に設定 | H'FFFF838E<br>ビット 3<br>ビット 2 | b'10 |
| P_PORTA.PACRL3 | PA2MD2 | ポート A コントロールレジスタ L3<br>ポート A コントロールレジスタ L2 | H'FFFF838A<br>ビット 2 | b'1 |
| P_PORTA.PACRL2 | PA2MD[1]<br>PA2MD[0] | PA2 モードビット、PA2/IRQ0/A2/PCIO/SCK2 端子の機能を選択<br>：（PA2MD2,PA2MD[1],PA2MD[0])=b'110 のとき、端子機能は SCK2 入出力（SCI）に設定 | H'FFFF838E<br>ビット 5<br>ビット 4 | b'10 |
| P_PORTA.PAIORL | PA2IOR | ポート A・IO レジスタ L<br>ポート A の端子の入出力方向を設定<br>PA2IOR=1 のとき、SCK2（PA2）端子は出力設定 | H'FFFF8386<br>ビット 2 | b'1 |

（6）　使用RAM説明

表2.52に、本タスクで使用するRAMの説明を示します。

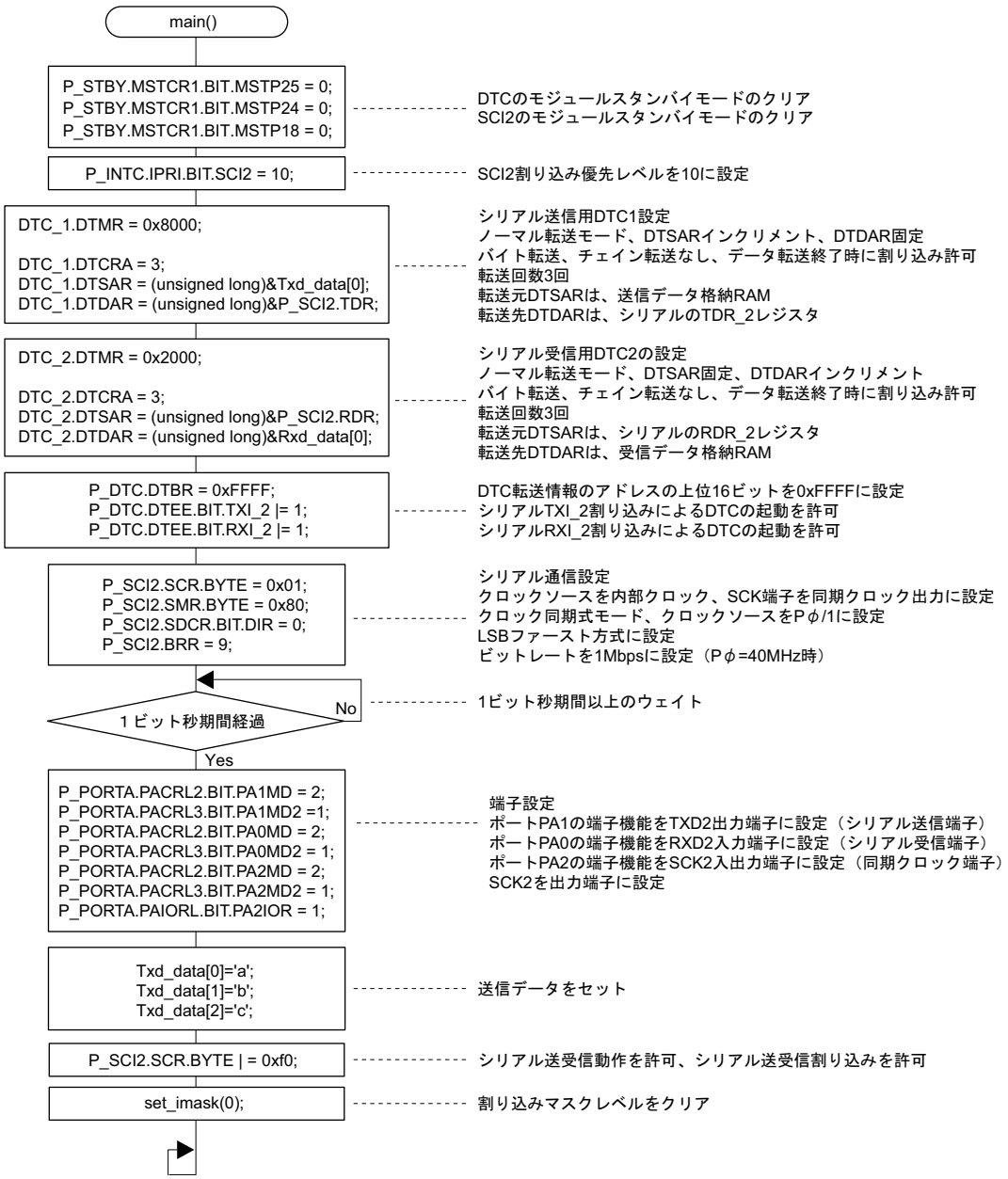
表 2.52　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
| --- | --- | --- | --- |
| Txd_data[0] | クロック同期式シリアル送信データの 1 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Txd_data[1] | クロック同期式シリアル送信データの 2 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Txd_data[2] | クロック同期式シリアル送信データの 3 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[0] | クロック同期式シリアル受信データの 1 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[1] | クロック同期式シリアル受信データの 2 バイト目を格納 | 内蔵 RAM | メインルーチン |
| Rxd_data[2] | クロック同期式シリアル受信データの 3 バイト目を格納 | 内蔵 RAM | メインルーチン |

| クロック同期式シリアルデータ同時送受信と DTC データ転送（SCI、DTC） | 使用機能 | SCI、DTC |
|---|---|---|

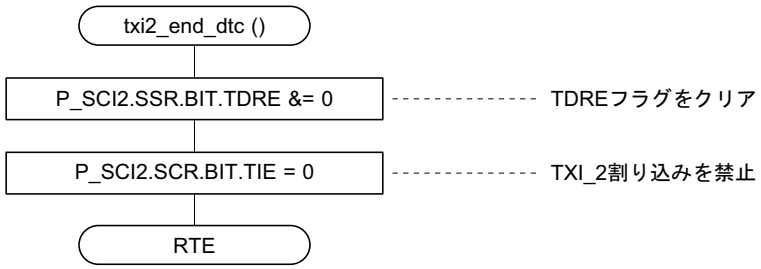
フローチャート

(a)　メイン処理

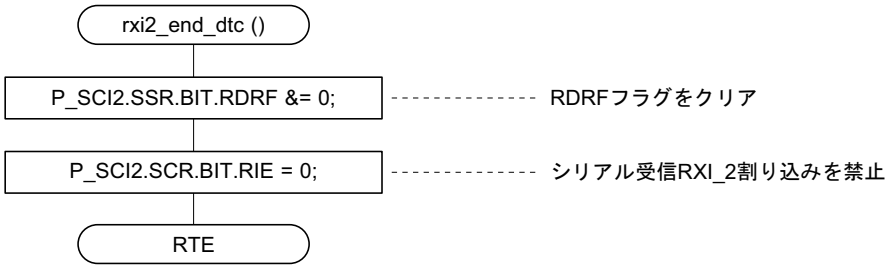


フローチャート

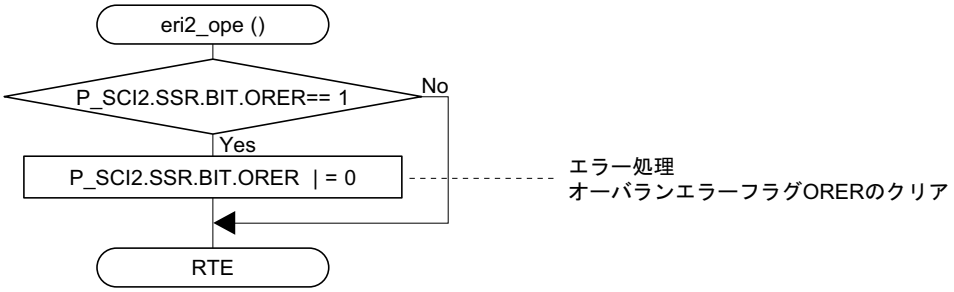
(b)　シリアル送信TXI_2割り込み処理




(c)　シリアル受信RXI_2割り込み処理




(d)　シリアルエラー割り込み処理



プログラムリスト

```
/******************************************************/
/* SH7046F Series -SH7047- Application Note           */
/* Synchronous Serial Data Transmission with DTC      */
/*                                                    */
/* Function                                           */
/* :Serial Communication Interface(SCI)               */
/*  Synchronous Serial Mode                           */
/*  -Transmitting/Receiving-                          */
/* :Data Transfer Controller(DTC)                     */
/*                                                    */
/* External input clock    :10MHz                     */
/* Internal CPU clock    :40MHz                       */
/* Internal peripheral clock :40MHz                   */
/*                                                    */
/* Written   :  2002/3/01  Rev.1.0                    */
/******************************************************/

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

/*------------ Symbol Definition ------------------------*/
struct st_dtc_tn {            /* DTC Normal Transfer information  */
    unsigned short DTMR;      /* DTC Mode Register            */
    unsigned short DTCRA;     /* transfer counter             */
    unsigned short dummy1;    /* Reserved                     */
    unsigned short dummy2;    /* Reserved                     */
    unsigned long DTSAR;      /* source address register      */
    unsigned long DTDAR;      /* destination address register */
};

#define DTC_COUNT 3               /* DTC Transmit count  */
#define DTC_1 (*(volatile struct st_dtc_tn *)0xFFFFE000) /* Transmit DTC     */
#define DTC_2 (*(volatile struct st_dtc_tn *)0xFFFFE010) /* Receive DTC      */
```

プログラムリスト

```
/*------------ RAM allocation Definition  --------------------*/
volatile unsigned char Txd_data[DTC_COUNT];  /* Transmit data  */
volatile unsigned char Rxd_data[DTC_COUNT];/* Receive data   */


/*------------ Function Definition -------------------------*/
void main(void);
void txi2_end(void);
void rxi2_end(void);


/************************************************************/
/* main Program                        */
/************************************************************/
void main( void )
{
    unsigned long i;

    /* Set standby mode  */
    P_STBY.MSTCR1.BIT.MSTP25 = 0;       /* Disable DTC standby mode   */
    P_STBY.MSTCR1.BIT.MSTP24 = 0;
    P_STBY.MSTCR1.BIT.MSTP18 = 0;       /* Disable SCI2 standby mode*/

    /* Set interrupt priority level (0 to 15)  */
    P_INTC.IPRI.BIT.SCI2 = 10;        /* SCI2 interrupt level 10  */
    /* SIC2 Transmit DTC information     */
    DTC_1.DTMR = 0x8000;
         /* SM[1:0]=b'10;  DTSAR is incremented                  */
         /* DM[1:0]=0;     DTDAR is  fixed                       */
         /* MD[1:0]=0;     Transfer mode :Normal mode            */
         /* SZ[1:0]=0;     Byte-size transfer                    */
         /* DTS=0;         destination is block area(no use)     */
         /* CHNE=0;        Chain transfer is canceled            */
         /* DISEL=0;       Interrupt->transfer ends              */
         /* NMIM=0;        NMI->Terminate DTC transfer           */
```

プログラムリスト

```
DTC_1.DTCRA = DTC_COUNT;      /* Transfer Count      */
DTC_1.DTSAR = (unsigned long)&Txd_data[0]; /* set SCI2 Transmit data    */
DTC_1.DTDAR = (unsigned long)&P_SCI2.TDR;  /* set SCI2 TDR register   */



/* SIC2 Receive DTC information      */
DTC_2.DTMR = 0x2000;
     /* SM[1:0]=0;   DTSAR is fixed                        */
     /* DM[1:0]=b'10;  DTDAR is incremented                */
     /* MD[1:0]=0;   Transfer mode :Normal mode            */
     /* SZ[1:0]=0;   Byte-size transfer                    */
     /* DTS=0;     destination is block area(no use)       */
     /* CHNE=0;    Chain transfer is canceled              */
     /* DISEL=0;     Interrupt->transfer ends              */
     /* NMIM=0;    NMI->Terminate DTC transfer             */
DTC_2.DTCRA = DTC_COUNT;      /* Transfer Count            */
DTC_2.DTSAR = (unsigned long)&P_SCI2.RDR;  /* set SCI2 RDR register     */
DTC_2.DTDAR = (unsigned long)&Rxd_data[0]; /* set SCI2 Receive Buffer   */

P_DTC.DTBR = 0xFFFF;           /* information base register */
/* DTC Transmit enable      */
P_DTC.DTEE.BIT.TXI_2 |= 1;      /* interrupt sources: TXI_2(SCI2) */
P_DTC.DTEE.BIT.RXI_2 |= 1;      /* interrupt sources: RXI_2(SCI2) */



/* Initialize SCI2 clocked synchronous mode  */
P_SCI2.SCR.BYTE = 0x01;
     /* TIE=0;     clear TIE          */
     /* RIE=0;     clear RIE          */
     /* TE=0;      clear TE           */
     /* RE=0;      clear RE           */
     /* MPIE=0;      clear MPIE,TEIE                      */
     /* TEIE=0;      clear TEIE                           */
     /* CKE[1:0]=b'01;    clock: external ,SCK:output     */
P_SCI2.SMR.BYTE = 0x80;
```

プログラムリスト

```
        /* CA=1;      clocked synchronous mode        */
        /* CKS[1:0]=b'00;   clock source =Pφ/1        */

    P_SCI2.SDCR.BIT.DIR = 0;      /* LSB first send        */
    P_SCI2.BRR = 9;               /* 1Mbps@ Pφ=40MHz       */
    for( i=0; i < 0x100 ; i++);   /* Wait 1bit over        */


    /* Initialize SCI2 port         */
    P_PORTA.PACRL2.BIT.PA1MD = 2;     /* set TXD2(PA1:73pin@SH7047)   */
    P_PORTA.PACRL3.BIT.PA1MD2 =1;
    P_PORTA.PACRL2.BIT.PA0MD = 2;     /* set RXD2(PA0:75pin@SH7047)   */
    P_PORTA.PACRL3.BIT.PA0MD2 = 1;
    P_PORTA.PACRL2.BIT.PA2MD = 2;     /* set SCK2(PA2:71pin@SH7047)   */
    P_PORTA.PACRL3.BIT.PA2MD2 = 1;
    P_PORTA.PAIORL.BIT.PA2IOR = 1;    /* set SCK2 output         */
    /* set transmit data */
    Txd_data[0] = 'a';
    Txd_data[1] = 'b';
    Txd_data[2] = 'c';

    P_SCI2.SCR.BYTE |= 0xf0;        /* Transmit/Receive Enable      */
           /* TIE=1;     TXI_2 interrupt Enable      */
           /* RIE=1;     RXI_2,ERI_2 interrupt Enable  */
           /* TE=1;      Transmit Enable          */
           /* RE=1;      Receive Enable         */

    set_imask(0);            /* clear interrupt mask level   */

    while(1);

}
```

プログラムリスト

```
/************************************************************/
/* SIC2:TXI_2 Interrupt                                     */
/* Transmission of DTC data transfer termination            */
/************************************************************/
#pragma interrupt(txi2_end)
void txi2_end(void)
{
    P_SCI2.SSR.BIT.TDRE &= 0;     /* TDRE=0 flag clear          */

    P_SCI2.SCR.BIT.TIE = 0;       /* TXI_2 interrupt disable    */
}



/************************************************************/
/* SIC2 RXI_2 Interrupt                                     */
/* Reception of DTC data transfer termination               */
/************************************************************/
#pragma interrupt(rxi2_end)
void rxi2_end(void)
{
    P_SCI2.SSR.BIT.RDRF &= 0;    /* RDRF=0 flag clear             */

    P_SCI2.SCR.BIT.RIE = 0;      /* RXI_2,ERI_2 interrupt disable  */
}
```

プログラムリスト

```
/************************************************************/
/* SIC2:ERI_2 Interrupt                                     */
/* SCI Reception Error                                      */
/************************************************************/
#pragma interrupt(eri2_ope)
void eri2_ope(void)
{
    if( P_SCI2.SSR.BIT.ORER==1){   /*  Overrun Error       */
    P_SCI2.SSR.BIT.ORER |= 0;      /*  ORER=0 flag clear   */
 }
}
```

## 2.7　MTU による A/D 変換の起動および変換結果の格納（A/D、DTC）

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

**仕様**

（1）　図2.37に示すように、AD入力端子8チャネルに電圧を印加し、A/D変換した結果をデータトランスファコントローラ（DTC:Data Transfer Controller）で内蔵RAMに格納します。

（2）　A/D変換器は、マルチファンクションタイマパルスユニット（MTU:Multi-Function Timer Pulse Unit）タイマch0のTGRA_0コンペアマッチで、2モジュール（AD0、AD1）を同時に起動します。

（3）　2モジュール（AD0、AD1）のA/D変換は、それぞれ4チャネルスキャンモード、1サイクルスキャンに設定し、2モジュール同時にサンプリングを行います。1回のMTUのコンペアマッチで、8チャネル（AN8からAN15）分のAD変換を行います。

（4）　DTCは、ブロック転送モードを使用します。図2.38に示すように、A/Dモジュール0のAD変換終了割り込みでDTCを起動し、A/Dモジュール0およびA/Dモジュール1の8チャネル分の変換結果を、1回のブロック転送で内蔵RAMに転送します。

（5）　本タスクでは、3回のブロック転送を行い、内蔵RAMに8チャネル×3回で24チャネル分（48バイト、1チャネルは1バイト）の変換結果を格納します。

（6）　DTCの転送条件を表2.53に示します。

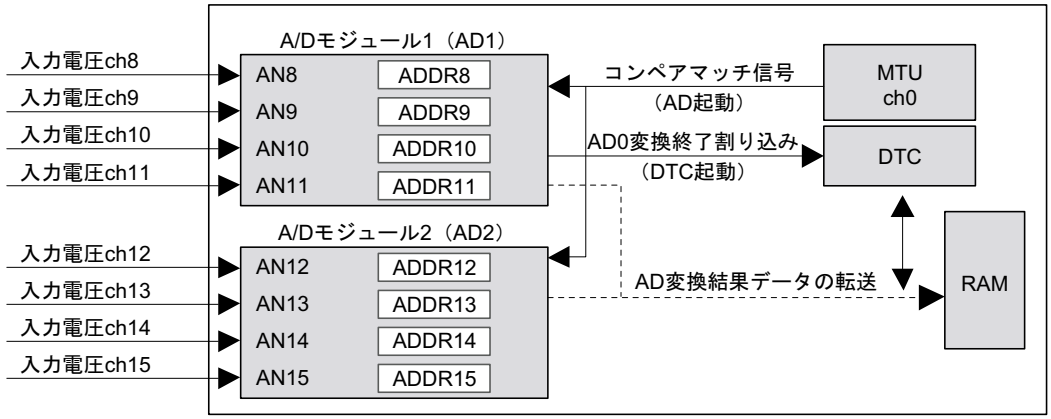



図 2.37　AD 入力電圧測定ブロック図

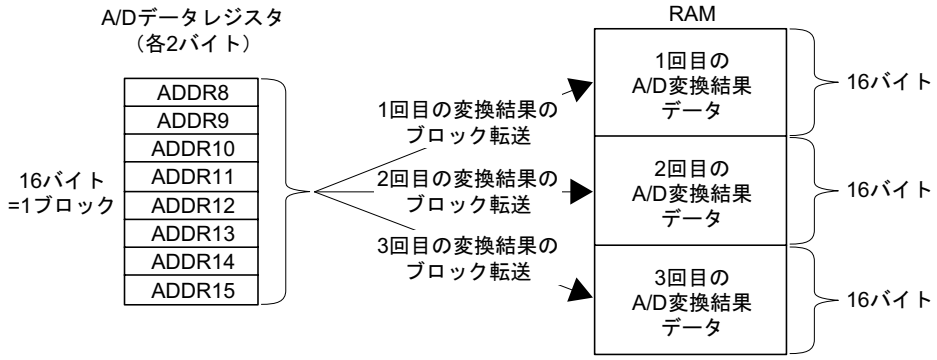



図 2.38　DTC を用いたデータ転送

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

仕様

表 2.53　DTC 転送条件

| 条件 | 内容 |
|---|---|
| 転送モード | ブロック転送モード、ソース側（転送元）がブロック領域 |
| 転送回数 | 3 回 |
| ブロック長 | 8 |
| 転送データサイズ | バイト（Byte）転送 |
| 転送元 | A/D 変換器の AD データレジスタ |
| 転送先 | 内蔵 RAM |
| 転送元アドレス | 転送後に転送元アドレスをインクリメント |
| 転送先アドレス | 転送後に転送先アドレスをインクリメント |
| 起動要因 | MTU ch0 のコンペアマッチで起動 |
| 割り込み処理 | 指定されたデータ転送を終了したときだけ CPU に対して割り込みを許可 |

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

**使用機能説明**

（1） 本タスク例では、MTUのタイマch0のコンペアマッチでA/D変換を起動し、変換結果をDTCで内蔵RAMに格納します。

（a） 図2.39にMTUタイマch0のブロック図を示します。本タスクでは、128ms周期で、TGRAのコンペアマッチを発生させ、ソフトウェアを介さずコンペアマッチ信号で、A/D変換を自動的に開始させる機能を使用します。以下にブロック図について説明します。

- タイマコントロールレジスタ_0（TCR_0）は、TCNT を制御する 8 ビットのレジスタです。TCNT のカウンタクリア要因の選択、入力クロックのエッジの選択、TCNT のカウンタクロックを選択します。
- タイマモードレジスタ_0（TMDR_0）は、8 ビットのレジスタで、動作モードの設定、バッファ動作の設定を行います。
- タイマ I/O コントロールレジスタ H_0（TIORH_0）は、タイマジェネラルレジスタ B_0（TGRB_0）とタイマジェネラルレジスタ A_0（TGRA_0）を制御する 8 ビットレジスタです。
- タイマ I/O コントロールレジスタ L_0（TIORL_0）は、タイマジェネラルレジスタ C_0（TGRC_0）とタイマジェネラルレジスタ D_0（TGRD_0）を制御する 8 ビットレジスタです。
- タイマインタラプトイネーブルレジスタ_0（TIER_0）は、8 ビットのレジスタで、割り込み要求の許可、禁止を制御します。
- タイマステータスレジスタ_0（TSR_0）は、8 ビットのレジスタで、ステータスの表示を行います
- タイマカウンタ_0（TCNT_0）は、TCNT は 16 ビットのカウンタです。8 ビット単位でのアクセスは禁止です。常に 16 ビット単位でアクセスしてください。
- タイマジェネラルレジスタ A_0（TGRA_0）は、16 ビットのアウトプットコンペア／インプットキャプチャ兼用のレジスタです。
- タイマジェネラルレジスタ B_0（TGRB_0）は、16 ビットのアウトプットコンペア／インプットキャプチャ兼用のレジスタです。
- タイマジェネラルレジスタ C_0（TGRC_0）は、16 ビットのアウトプットコンペア／インプットキャプチャ兼用のレジスタです。TGRC_0 は、TGRA_0 と組み合わせてバッファレジスタとして動作を設定することができます。
- タイマジェネラルレジスタ D_0（TGRD_0）は、16 ビットのアウトプットコンペア／インプットキャプチャ兼用のレジスタです。TGRD_0 は、TGRB_0 と組み合わせてバッファレジスタとして動作を設定することができます。
- タイマスタートレジスタ（TSTR）は、8 ビットのレジスタで、TCNT の動作／停止を選択します

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

使用機能説明

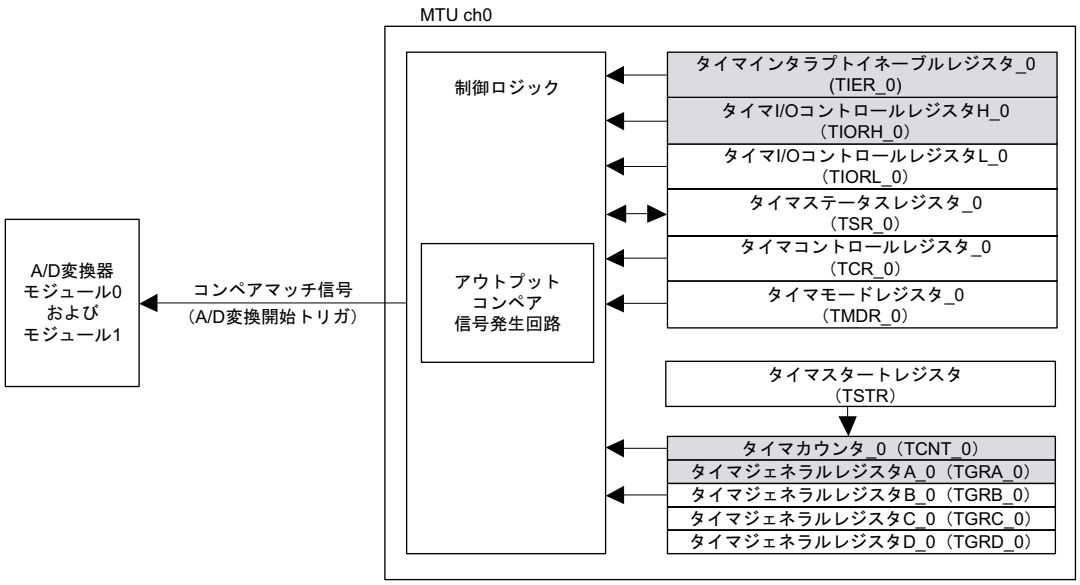



図 2.39　MTU タイマ ch0 機能ブロック図

(b)　図2.40にA/D変換器のブロック図を示します。A/D変換器では、以下の機能を使用してアナログからデジタルへの変換を行います。A/D変換終了割り込みでDTCを起動し変換結果を転送します。

- 複数のチャネル（ch8～ch11、ch12～ch15）の A/D 変換を 1 回行う。
  （4チャネル、1サイクルスキャンモード）。
- A/D モジュール 0（ch8～ch11）および A/D モジュール 1（ch12～ch15）の入力電圧を同時にサンプリングし変換する機能。（同時サンプリング）
- MTU のコンペアマッチを変換開始トリガとして、A/D 変換を開始。
- A/D 変換終了時に DTC を起動させる機能。

以下にブロック図について説明します。

- A/D データレジスタ 8～11（ADDR8～ADDR11）は、A/D 変換された結果を格納するための 16 ビットのリード専用レジスタ、10 ビットの変換データは ADDR のビット 15 からビット 6 に格納。
- A/D コントロール／ステータスレジスタ_0,1（ADCSR_0,1）は、A/D 変換動作を制御。
- A/D コントロールレジスタ_0,1（ADCR_0,1）は、外部トリガによる A/D 変換開始制御および動作クロックの選択を行う。
- A/D トリガセレクトレジスタ（ADTSR）は、外部トリガによる A/D 変換開始をイネーブル。

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

使用機能説明

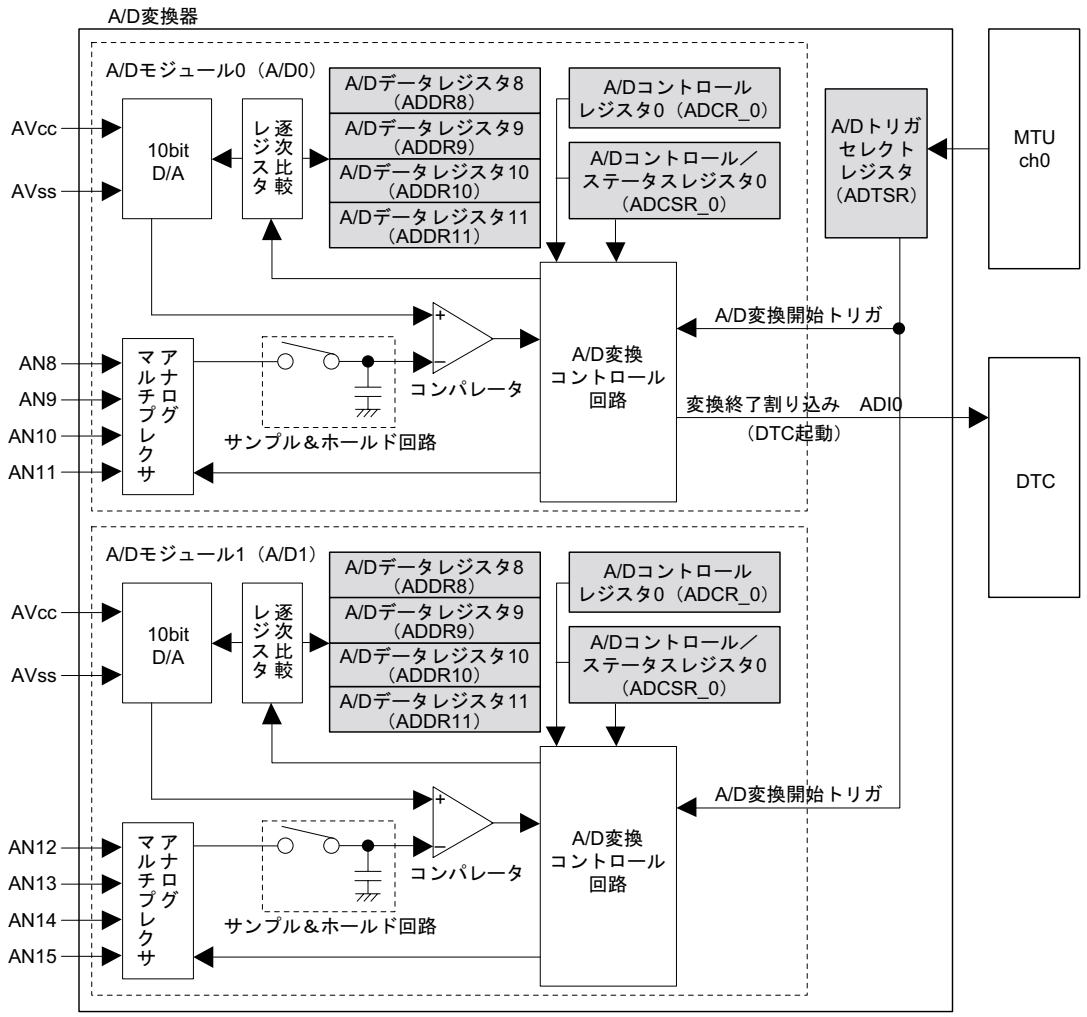



図 2.40　電圧測定、A/D 変換ブロック図

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

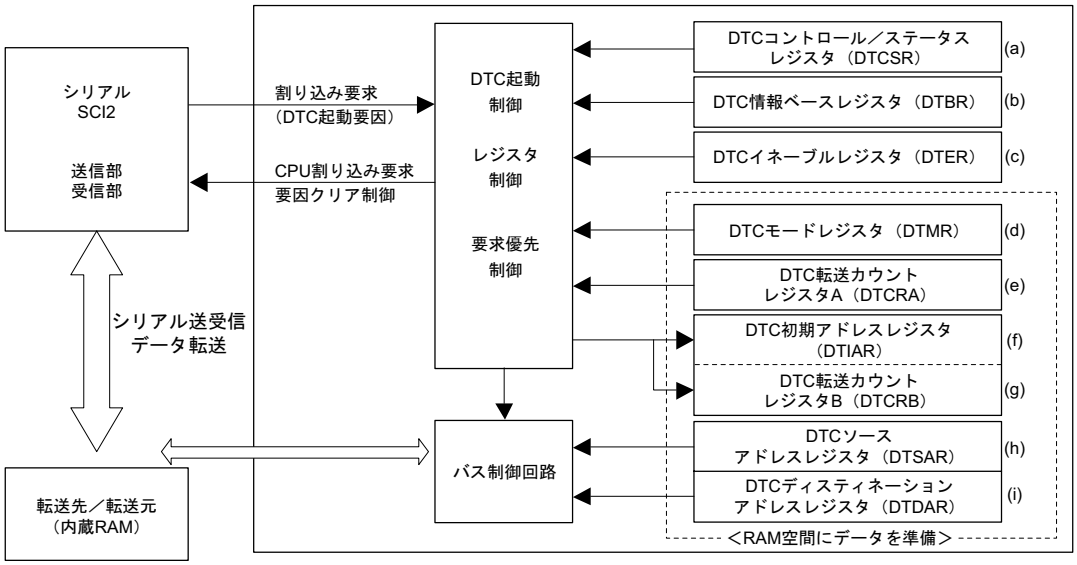
使用機能説明

(c) 図2.41にDTCのブロック図を示します。本タスクでは、DTCの3種類（ノーマルモード、リピートモード、ブロック転送モード）の転送モードのうち、ブロック転送モードを使用して、AD変換結果のデータ転送を行います。DTC起動要因を、AD変換終了割り込みADI0割り込みとして、DTCデータ転送を行います。以下にブロック図について説明します。

- DTC モードレジスタ（DTMR）は、16 ビットのレジスタで、DTC の動作モードの制御を行います。
- DTC ソースアドレスレジスタ（DTSAR）は、32 ビットのレジスタで、DTC の転送するデータの転送元アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC デスティネーションアドレスレジスタ（DTDAR）は、32 ビットのレジスタで、DTC の転送するデータの転送先アドレスを指定します。ワードサイズの場合は偶数アドレス、ロングワードの場合は 4 の倍数アドレスを指定してください。
- DTC 初期アドレスレジスタ（DTIAR）は、32 ビットのレジスタで、リピートモードのときに転送元／転送先の初期アドレスを指定します。リピートモードにおいて、DTS ビットが 1 のとき、リピートエリアにおける転送元アドレスの初期アドレスを指定してください。DTS ビットが 0 のとき、リピートエリアにおける転送先アドレスの初期アドレスを指定してください。
- DTC 転送カウントレジスタ A（DTCRA）は、16 ビットのレジスタで、DTC のデータ転送の転送回数を指定します。ノーマルモードでは、16 ビットの転送カウンタ（1～65,536）として機能します。リピートモードでは、上位 8 ビットの DTCRAH は転送回数を保持し、下位 8 ビットの DTCRAL は 8 ビット転送カウンタとして機能します。ブロック転送モードでは、16 ビットの転送カウンタとして機能します。
- DTC 転送カウントレジスタ B（DTCRB）は、16 ビットのレジスタで、ブロック転送モードのとき、ブロック長を指定します。
- DTC イネーブルレジスタ（DTER）は、DTC を起動する割り込み要因を選択するためのレジスタで、DTEA～DTEF があります。
- DTC コントロール／ステータスレジスタ（DTCSR）は、16 ビットのレジスタで、ソフトウェアによる DTC 起動の許可／禁止の設定、およびソフトウェア起動による DTC ベクタアドレスを設定します。また、DTC 転送の状態も示します。
- DTC 情報ベースレジスタ（DTBR）は、読み出し／書き込み可能な 16 ビットのレジスタで、DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定します。DTBR のアクセスは、必ずワードまたはロングワード単位で行ってください。バイト単位でアクセスすると、書き込み時はレジスタの内容が不定になり、また読み出し時は不定値が読み出されます。
- DTC モードレジスタ（DTMR）、DTC ソースアドレスレジスタ（DTSAR）、DTC デスティネーションアドレスレジスタ（DTDAR）、DTC 初期アドレスレジスタ（DTIAR）、DTC 転送カウントレジスタ A（DTCRA）、DTC 転送カウントレジスタ B（DTCRB）の 6 本のレジスタ情報は、CPU から直接アクセスすることはできません。DTC 起動要因が発生すると内蔵 RAM 上に配置された任意の組のレジスタ情報から該当するレジスタ情報をこれらのレジスタに転送して DTC 転送を行い、転送が終了するとこれらのレジスタの内容が RAM に戻されます。したがって、レジスタ情報は、ユーザプログラム上で任意の内蔵 RAM 上に準備してください。

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

使用機能説明




【注】
(a) ソフトウェアによるDTC起動の許可／禁止、ソフトウェア起動によるDTCベクタアドレスの設定を行う。
(b) DTC転送情報を格納するメモリアドレスの上位16ビットの指定を行う。
(c) DTCを起動する割り込み要因を選択、レジスタはDTEAからDTEFの6個あります。
(d) DTC動作モードの設定を行う。
(e) DTCのデータ転送の転送回数を指定。
(f) リピートモードのときに、転送元／転送先の初期アドレスを指定。ノーマルモードでは未使用。ブロック転送モードではDTCRBレジスタとして機能します。
(g) ブロック転送モードのときに、ブロック長を指定。ノーマルモードでは未使用。リピートモードではDTIARレジスタとして機能します。
(h) DTCの転送するデータの転送元アドレスを指定。
(i) DTCの転送するデータの転送先アドレスを指定。

図 2.41　DTC ブロック図

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

使用機能説明

（2） 表2.54に本タスク例の機能割り付けを示します。

表 2.54　機能割付け

| 機能 | 分類 | 機能割り付け |
|---|---|---|
| AN8～AN11 | 端子 | アナログ測定端子 |
| TCR_0 | MTU ch0 | カウンタクリア要因の選択 |
| TIER_0 | MTU ch0 | A/D 変換開始要求の発生を許可 |
| TGRA_0 | MTU ch0 | サンプリング周期の設定 |
| ADCR_0 | A/D0 | AD 変換モード、測定端子の設定 |
| ADCSR_0 | A/D0 | 変換時間、起動要因の設定 |
| ADCR_1 | A/D1 | AD 変換モード、測定端子の設定 |
| ADCSR_1 | A/D1 | 変換時間、起動要因の設定 |
| ADDR8～ADDR11 | A/D0 | AD モジュール 0 変換結果の格納レジスタ |
| ADDR12～ADDR15 | A/D1 | AD モジュール 1 変換結果の格納レジスタ |
| ADTSR | AD | AD 変換開始を MTU トリガに設定 |
| DTMR | DTC | DTC をブロック転送モードに設定 |
| DTCRA | DTC | 転送回数の設定 |
| DTCRB | DTC | ブロック長の設定 |
| DTSAR | DTC | 転送元アドレスの設定 |
| DTDAR | DTC | 転送先アドレスの設定 |
| DTBR | DTC | DTC ベクタ上位 16 ビットの設定 |
| DTEC | DTC | AD 変換終了時に DTC の起動を許可 |

（1） 図2.42に動作原理を示します。
MTU ch0のTGRA_0のコンペアマッチによってA/D変換が起動し、AN8～AN11、AN12～AN15のAD入力端子電圧を順次変換します。指定したすべてのチャネルの変換終了後、DTCが起動されAD変換結果をRAMに転送します。

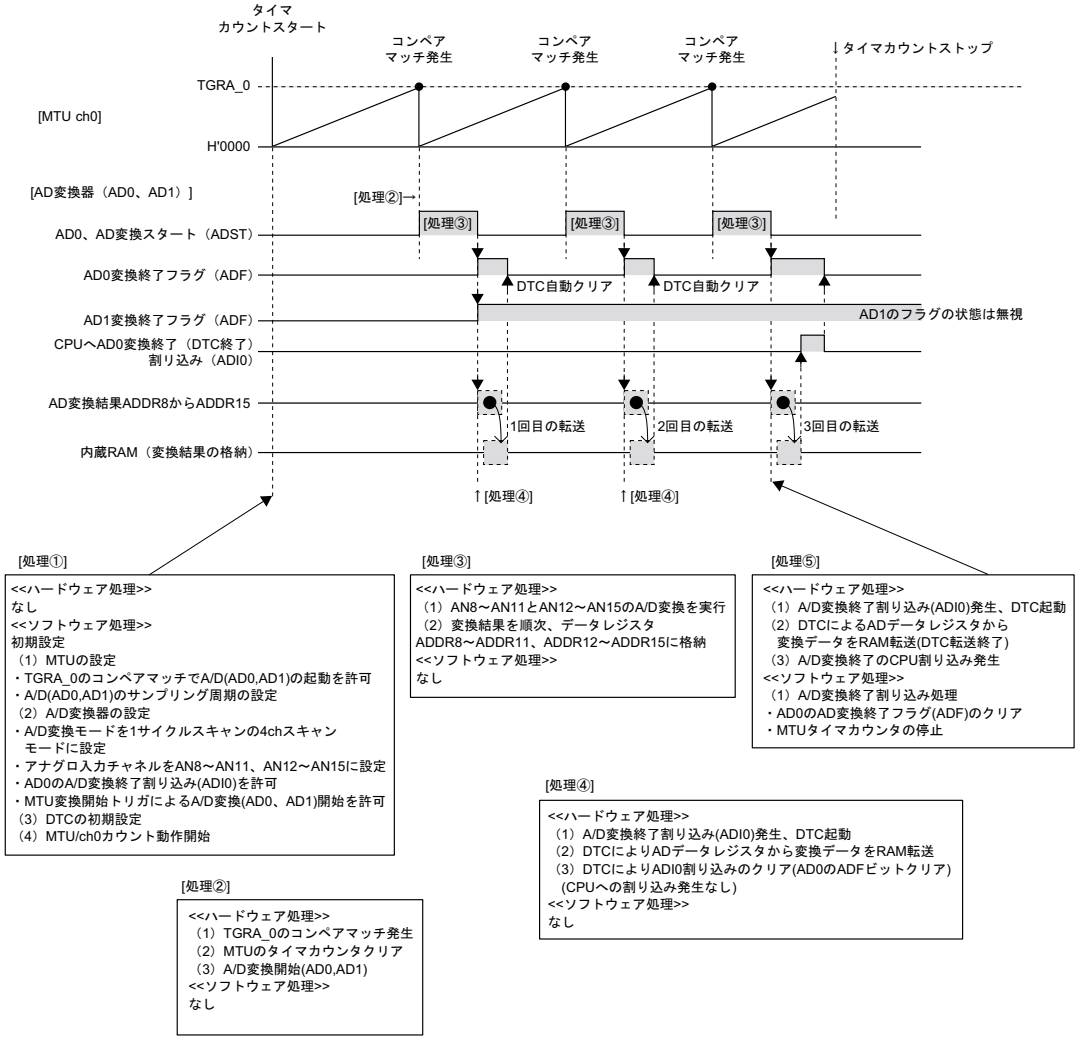



図 2.42　動作原理

動作説明

（2） 図2.43にDTC起動の動作原理を示します。DTCを実行する場合、起動要因が発生する前に以下の設定を行ってください。

- DTC レジスタ情報の設定、DTC レジスタ情報は RAM 上に配置してください。
- DTC ベクタテーブルに DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。
- DTC 情報ベースレジスタ（DTMR）に DTC レジスタ情報の先頭アドレス（32 ビット）の下位 16 ビットを設定します。

以下の処理で、DTCが起動します。

- DTC 起動要因の割り込みが発生。
- DTC のベクタテーブルの起動要因に該当するアドレスから、DTC レジスタ情報の先頭アドレスの下位 16 ビットを読み込みます。
- DTC 情報ベースレジスタ（DTMR）から、DTC レジスタ情報の先頭アドレスの上位 16 ビットを読み込みます。
- 読み込んだ先頭アドレス下位 16 ビットと上位 16 ビットから、DTC レジスタ情報の 32 ビットの先頭アドレスを生成。
- DTC レジスタ情報先頭アドレスから、DTC レジスタ情報先頭を順次読み込み、データ転送を行います。

本タスクでは、AD0のA/D変換終了割り込みがDTC起動要因となります。
表2.55にブロック転送モード時のレジスタ情報の構成を示します。

表 2.55　DTC レジスタ情報一覧（ブロック転送モード）

| 設定アドレス | レジスタ名 | データ長 |
|---|---|---|
| RF | DTC モードレジスタ（DTMR） | ワード（2Byte） |
| RF+2 | DTC 転送カウントレジスタ A（DTCRA） | ワード（2Byte） |
| RF+6 | DTC 転送カウントレジスタ B（DTCRB） | ワード（2Byte） |
| RF+8 | DTC ソースアドレスレジスタ（DTSAR） | ロングワード（4Byte） |
| RF+12 | DTC ディスティネーションアドレスレジスタ（DTDAR） | ロングワード（4Byte） |

【注】 RF:DTC レジスタ情報の先頭アドレス（内蔵 RAM 上）

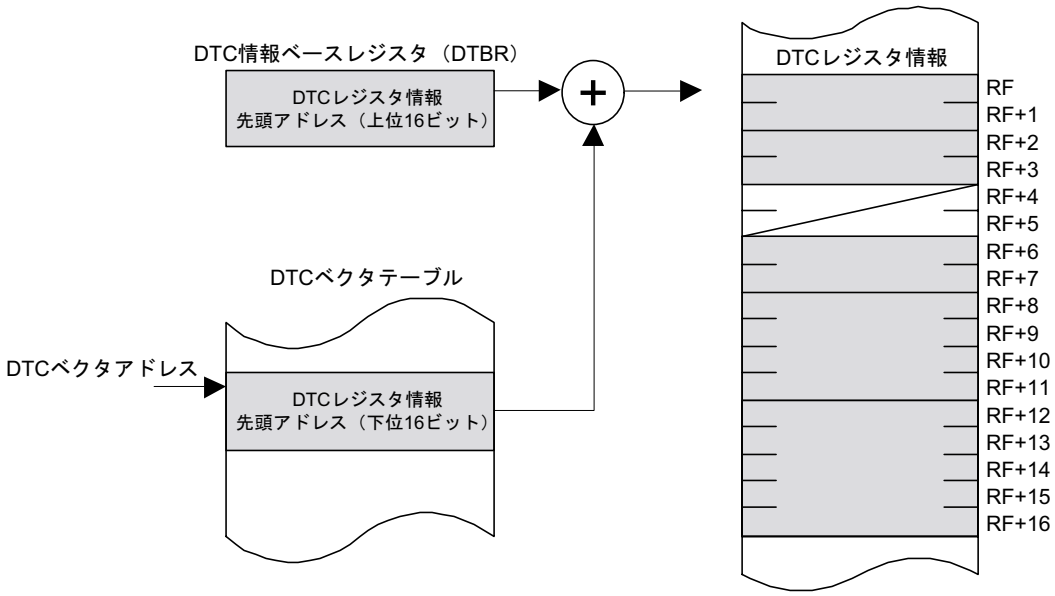



図 2.43　DTC ベクタアドレスと転送情報との対応

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

ソフトウェア説明

（1） モジュール説明

**表2.56**に、本タスク例のモジュールを示します。

表 2.56　モジュール説明

| モジュール名 | ラベル名 | 機能 |
|---|---|---|
| メインルーチン | main | MTU チャネル 0、AD 変換器（AD0,AD1）、DTC の初期設定を行う |
| A/D 変換終了（AD0）割り込み | ad_adi0_dtc | AD0 モジュールの AD 変換終了割り込み。DTC 指定回数転送終了時に割り込み発生 |

（2） 引数の説明

本タスク例では、引数を使用しません。

（3） 使用内部レジスタ説明

**表2.57**～**表2.60**に、本タスク例の使用する内部レジスタを示します。

表 2.57　使用内蔵レジスタ説明（1）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_STBY.MSTCR1 | MSTP25<br>MSTP24 | モジュールスタンバイコントロールレジスタ 1<br>DTC モジュールスタンバイ制御ビット<br>:MSTP25=MSTP24=1 のとき、モジュールスタンバイ状態<br>:MSTP25=MSTP24=0 のとき、モジュールスタンバイ解除<br>MSTP25,24 には同じ設定する | H'FFFF861C<br>ビット 9<br>ビット 8 | b'00 |
| P_STBY.MSTCR2 | MSTP13 | モジュールスタンバイコントロールレジスタ 2<br>MTU モジュールスタンバイ制御ビット<br>:MSTP13=1 のとき、モジュールスタンバイ状態<br>:MSTP13=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 13 | 0 |
| | MSTP5 | モジュールスタンバイコントロールレジスタ 2<br>A/D 変換器（AD1）モジュールスタンバイ制御ビット<br>:MSTP5=1 のとき、モジュールスタンバイ状態<br>:MSTP5=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 5 | 0 |
| | MSTP4 | モジュールスタンバイコントロールレジスタ 2<br>A/D 変換器（AD0）モジュールスタンバイ制御ビット<br>:MSTP4=1 のとき、モジュールスタンバイ状態<br>:MSTP4=0 のとき、モジュールスタンバイ解除 | H'FFFF861E<br>ビット 4 | 0 |
| P_INTC.IPRG | AD01 | インタラプトプライオリティレジスタ G（IPRG）<br>AD 変換器（AD0 および AD1）の AD 変換終了割り込み（ADI0 および ADI1）の割り込み優先レベルの設定<br>:AD01=b'1000(8）のとき、ADI0 および ADI1 割り込みは、割り込み優先レベル 8 に設定 | H'FFFF8354<br>ビット<br>12～15 | 8 |

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_B.DTMR | | DTC モードレジスタ（DTMR）<br>DTC の動作モードの制御設定。 | 内蔵 RAM に配置 | H'a980 |
| | SM1<br>SM0 | ソースアドレスモード<br>：SM[1:0]=b'10 のとき、転送後 DTSAR をインクリメント | ビット 15<br>ビット 14 | |
| | DM1<br>DM0 | デスティネーションアドレスモード<br>：DM[1:0]=b'10 のとき、転送後 DTDAR をインクリメント | ビット 13<br>ビット 12 | |
| | MD1<br>MD0 | DTC の転送モード<br>：MD[1:0]=b'10 のとき、ブロック転送モード | ビット 11<br>ビット 10 | |
| | SZ1<br>SZ0 | DTC データトランスファサイズ<br>：SZ[1:0]=b'01 のとき、ワード（2Byte）転送 | ビット 9<br>ビット 8 | |
| | DTS | DTC 転送モードセレクト<br>：DTS=b'1 のとき、ソース側がブロック領域 | ビット 7 | |
| | CHNE | DTC チェイン転送イネーブル<br>：CHNE=b'0 のとき、チェイン転送を解除 | ビット 6 | |
| | DISEL | DTC インタラプトセレクト<br>：DISEL=b'0 のとき、指定されたデータ転送を終了したときだけ CPU に対して割り込み要求を発生 | ビット 5 | |
| | NMIM | DTC NMI モード<br>：NMIM=b'0 のとき、NMI により DTC 転送を中断 | ビット 4 | |

表 2.58　使用内蔵レジスタ説明（2）

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| DTC_B.DTCRA | | DTC 転送カウントレジスタ A（DTCRA）<br>DTC のデータ転送の転送回数を指定<br>3 回転送に設定 | 内蔵 RAM に配置 | H'03 |
| DTC_B.DTCRB | | DTC 転送カウントレジスタ B（DTCRB）<br>ブロック転送モードのとき、ブロック長を指定<br>AD データレジスタ数と同じ 8 ブロックに設定 | 内蔵 RAM に配置 | H'08 |
| DTC_B.DTSAR | | DTC ソースアドレスレジスタ（DTSAR）<br>DTC の転送するデータの転送元アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | P_AD.ADDR8.WORD |
| DTC_B.DTDAR | | DTC デスティネーションアドレスレジスタ（DTDAR）<br>DTC の転送するデータの転送先アドレスを指定、32 ビットレジスタ | 内蔵 RAM に配置 | Ad_data; |
| P_DTC.DTBR | | DTC 情報ベースレジスタ（DTBR）<br>DTC 転送情報を格納するメモリアドレスの上位 16 ビットを指定 | H'FFFF8708 | 0xFFFF |
| P_DTC.DTEC | ADI0 | DTC イネーブルレジスタ E（DTEC）<br>1 をセットすると対応する割り込み要因が DTC 起動要因として選択<br>：ADI0(DTEC6)=b'1 のとき、A/D 変換器（AD0）の AD0 変換終了割り込み（ADI0）が起動要因 | H'FFFF8702<br>ビット 6 | 1 |

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス | 設定値 |
|---|---|---|---|---|
| | | | ビット | |
| P_MTU34.TSTR | | MTU タイマスタートレジスタ（TSTR）<br>TCNT の動作／停止を選択 | H'FFFF8240 | H'01 |
| | CST4 | カウンタスタート 4<br>：CST4=b'0 のとき、TCNT_4 のカウント動作は停止 | ビット 7 | |
| | CST3 | カウンタスタート 3<br>：CST3=b'0 のとき、TCNT_3 のカウント動作は停止 | ビット 6 | |
| | CST2 | カウンタスタート 2<br>：CST2=b'0 のとき、TCNT_2 のカウント動作は停止 | ビット 2 | |
| | CST1 | カウンタスタート 1<br>：CST1=b'0 のとき、TCNT_1 のカウント動作は停止 | ビット 1 | |
| | CST0 | カウンタスタート 0<br>：CST0=b'1 のとき、TCNT_0 のカウント動作 | ビット 0 | |
| P_MTU0.TCR_0 | | MTU タイマコントロールレジスタ_0（TCR_0）<br>TCNT を制御レジスタ | H'FFFF8260 | H'23 |
| | CCLR2<br>CCLR1<br>CCLR0 | TCNT_0 のカウンタクリア要因の選択<br>：CCLR[2:0]=b'001 のとき、TGRA のコンペアマッチ/インプットキャプチャで TCNT クリア | ビット 7<br>ビット 6<br>ビット 5 | |
| | CKEG1<br>CKEG0 | 入力クロックのエッジの選択<br>CKEG[1:0]=b'00 とき、立ち上がりエッジでカウント | ビット 4<br>ビット 3 | |
| | TPSC2<br>TPSC1<br>TPSC0 | TCNT のカウンタクロックの選択<br>：TPSC[2:0]=b'011 とき、内部クロック：Pφ／64 でカウント | ビット 2<br>ビット 1<br>ビット 0 | |
| P_MTU0.TMDR_0 | | MTU タイマモードレジスタ_0（TMDR_0）<br>各チャネルの動作モードの設定を行う | H'FFFF8261 | H'00 |
| | BFB | バッファ動作 B<br>：BFA=b'0 のとき、TGRB と TGRD は通常動作 | ビット 5 | |
| | BFA | バッファ動作 A<br>：BFB=b'0 のとき、TGRA と TGRC は通常動作 | ビット 4 | |
| | MD3<br>MD2<br>MD1<br>MD0 | タイマ動作モードの設定<br>：MD[3:0]=b'0000 のとき、タイマは通常動作モード | ビット 3<br>ビット 2<br>ビット 1<br>ビット 0 | |
| P_MTU0.TIORH_0 | | MTU タイマ I/O コントロールレジスタ H_0（TIORH_0）<br>TGR を制御 | H'FFFF8262 | H'00 |
| | IOB3<br>IOB2<br>IOB1<br>IOB0 | I/O コントロール B3～0<br>：IOB[3:0]=b'0000 のとき、TGRB_0 はアウトプットコンペアマッチレジスタ、TIOC0B 端子は出力禁止 | ビット 7<br>ビット 6<br>ビット 5<br>ビット 4 | |
| | IOA3<br>IOA2<br>IOA1<br>IOA0 | I/O コントロール A3～0<br>：IOA[3:0]=b'0000 のとき、TGRA_0 はアウトプットコンペアマッチレジスタ、TIOC0A 端子は出力禁止 | ビット 3<br>ビット 2<br>ビット 1<br>ビット 0 | |

表 2.59　使用内蔵レジスタ説明（3）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_MTU0.TIORL_0 | | MTU タイマ I/O コントロールレジスタ L_0（TIORL_0）<br>TGR を制御 | H'FFFF8263 | H'00 |
| | IOD3<br>IOD2<br>IOD1<br>IOD0 | I/O コントロール D3～0<br>：IOD[3:0]=b'0000 のとき、TGRD_0 はアウトプットコンペアマッチレジスタ、TIOC0D 端子は出力禁止 | ビット 7<br>ビット 6<br>ビット 5<br>ビット 4 | |
| | IOC3<br>IOC2<br>IOC1<br>IOC0 | I/O コントロール C3～0<br>：IOC[3:0]=b'0000 のとき、TGRC_0 はアウトプットコンペアマッチレジスタ、TIOC0D 端子は出力禁止 | ビット 3<br>ビット 2<br>ビット 1<br>ビット 0 | |
| P_MTU0.TIER_0 | | MTU タイマインタラプトイネーブルレジスタ_0（TIER_0）<br>各チャネルの割り込み要求の許可、禁止を制御 | H'FFFF8264 | H'c0 |
| | TTGE | A/D 変換開始要求イネーブル<br>TGRA のインプットキャプチャ/コンペアマッチによる A/D 変換器開始要求の発生を許可または禁止<br>TTGE=b'1：A/D 変換開始要求の発生を許可 | ビット 7 | |
| | TGIEU | アンダフローインタラプトイネーブル<br>TSR の TCFU フラグによる割り込み要求（TCIU）を許可または禁止<br>：TGIEU=b'0 のとき TCFU による割り込み要求（TCIU）を禁止 | ビット 5 | |
| | TGIEV | オーバフローインタラプトイネーブル<br>TSR の TCFV フラグによる割り込み要求（TCIV）を許可または禁止<br>：TGIEV=b'0 のとき、TCFV による割り込み要求（TCIV）を禁止 | ビット 4 | |
| | TGIED | TGR インタラプトイネーブル D<br>TSR の TGFD ビットによる割り込み要求（TGID）を許可または禁止<br>：TGIED=b'0 のとき、TGFD ビットによる割り込み要求（TGID）を禁止 | ビット 3 | |
| | TGIEC | TGR インタラプトイネーブル C<br>TSR の TGFC ビットによる割り込み要求（TGIC）を許可または禁止<br>：TGIEC=b'0 のとき、TGFC ビットによる割り込み要求（TGIC）を禁止 | ビット 2 | |
| | TGIEB | TGR インタラプトイネーブル B<br>TSR の TGFB ビットによる割り込み要求（TGIB）を許可または禁止<br>：TGIEB=b'0 のとき、TGFB ビットによる割り込み要求（TGIB）を禁止 | ビット 1 | |
| | TGIEA | TGR インタラプトイネーブル A<br>TSR の TGFA ビットによる割り込み要求（TGIA）を許可または禁止<br>：TGIEA=b'1 のとき、TGFA ビットによる割り込み要求（TGIA）を許可 | ビット 0 | |
| P_MTU0.TCNT_0 | | MTU タイマカウンタ 0（TCNT_0） | H'FFFF8266 | H'0000 |
| P_MTU0.TGRA_0 | | MTU ジェネラルレジスタ A_0（TGRA_0） | H'FFFF8268 | H'9c40 |

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

ソフトウェア説明

| レジスタ名 | ビット | 機能 | アドレス<br>ビット | 設定値 |
|---|---|---|---|---|
| P_AD.ADCSR_0 | | A/D コントロール／ステータスレジスタ_0（ADCSR_0）<br>A/D 変換動作を制御 | H'FFFF8480 | H'5f |
| | ADIE | A/D インタラプト（ADI）イネーブル<br>：ADIE=b'1 のとき、ADI0 割り込みを許可 | ビット 6 | |
| | ADM1<br>ADM0 | A/D 変換の動作モードの選択<br>：ADM[1:0]=b'01 のとき、4 チャネルスキャンモード | ビット 5<br>ビット 4 | |
| | CH2<br>CH1<br>CH0 | チャネルセレクト 2～0<br>A/D 変換するアナログ入力チャネルを選択<br>：CH[2:0]=b'111 のとき、AD 入力端子 AN12～AN15 を選択 | ビット 2<br>ビット 1<br>ビット 0 | |
| P_AD.ADCR_0 | | A/D コントロールレジスタ_0（ADCR_0）<br>外部トリガによる A/D 変換開始制御および動作クロックの選択 | H'FFFF8489 | H'e7 |
| | TRGE | トリガイネーブル<br>:TRGE=b'1 のとき、トリガによる AD 変換開始は有効 | ビット 7 | |
| | CKS1<br>CKS0 | クロックセレクト 1～0<br>CKS[1:0]=b11'のとき、クロック Pφ/4 で変換 | ビット 6<br>ビット 5 | |
| | ADST | A/D スタート<br>ADST=b'0 のとき、A/D 変換は待機状態（本タスクでは MTU タイマトリガで変換開始） | ビット 4 | |
| | ADCS | A/D 連続スキャン<br>：ADCS=b'0 のとき、1 サイクルスキャン | ビット 3 | |

表 2.60　使用内蔵レジスタ説明（4）

| レジスタ名 | ビット | 機能 | アドレス / ビット | 設定値 |
|---|---|---|---|---|
| P_AD.ADCSR_1. | | A/D コントロール／ステータスレジスタ_1（ADCSR_1）<br>A/D 変換動作を制御 | H'FFFF8481 | H'1f |
| | ADIE | A/D インタラプト（ADI）イネーブル<br>：ADIE=b'0 のとき、ADI1 割り込みを禁止 | ビット 6 | |
| | ADM1<br>ADM0 | A/D 変換の動作モードの選択<br>：ADM[1:0]=b'01 のとき、4 チャネルスキャンモード | ビット 5<br>ビット 4 | |
| | CH2<br>CH1<br>CH0 | チャネルセレクト 2～0<br>A/D 変換するアナログ入力チャネルを選択<br>：CH[2:0]=b'111 のとき、AD 入力端子 AN8～AN11 を選択 | ビット 2<br>ビット 1<br>ビット 0 | |
| P_AD.ADCR_1 | | A/D コントロールレジスタ_1（ADCR_1）<br>外部トリガによる A/D 変換開始制御および動作クロックの選択 | H'FFFF8489 | H'e7 |
| | TRGE | トリガイネーブル<br>:TRGE=b'1 のとき、トリガによる AD 変換開始は有効 | ビット 7 | |
| | CKS1<br>CKS0 | クロックセレクト 1～0<br>CKS[1:0]=b11'のとき、クロック Pφ/4 で変換 | ビット 6<br>ビット 5 | |
| | ADST | A/D スタート<br>ADST=b'0 のとき、A/D 変換は待機状態（本タスクでは MTU タイマトリガで変換開始） | ビット 4 | |
| | ADCS | A/D 連続スキャン<br>：ADCS=b'0 のとき、1 サイクルスキャン | ビット 3 | |
| P_AD.ADTSR | | A/D トリガセレクトレジスタ（ADTSR）<br>トリガ信号による A/D モジュールの変換開始をイネーブルにする | H'FFFF87F4 | H'0a |
| | TRG2S1<br>TRG2S0 | AD トリガ 2 セレクト 1～0<br>：TRG2S[1:0]=b'00 のとき、外部トリガ端子（ADTRG）または MTU のトリガを選択（未使用） | ビット 5<br>ビット 4 | |
| | TRG1S1<br>TRG1S0 | AD トリガ 1 セレクト 1～0<br>：TRG1S[1:0]=b'10 のとき、MTU の変換開始トリガを選択 | ビット 3<br>ビット 2 | |
| | TRG0S1<br>TRG0S0 | AD トリガ 0 セレクト 1～0<br>：TRG0S[1:0]=b'10 のとき、MTU の変換開始トリガを選択 | ビット 1<br>ビット 0 | |

（4）　使用RAM説明

表2.61に、本タスクで使用するRAMの説明を示します。

表 2.61　使用 RAM 説明

| ラベル名 | 機能 | アドレス | 使用モジュール名 |
|---|---|---|---|
| Ad_data | AD 変換データ（2byte）の格納<br>8×3 の unsigned short 型の 2 次元配列<br>AD 変換結果データ、8ch×3 セットを格納 | 内蔵 RAM | メインルーチン |

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

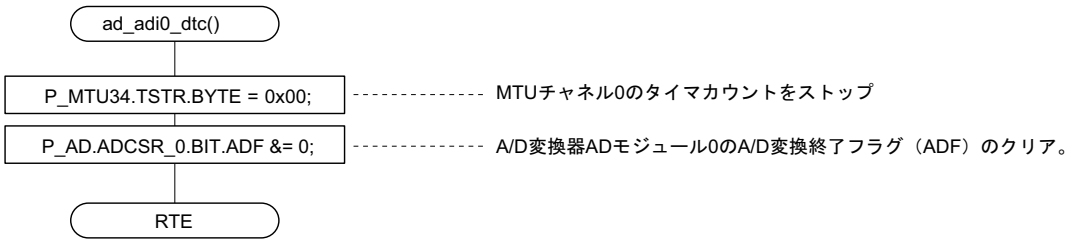
フローチャート

(a)　メイン処理

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

フローチャート

（b） A/D変換終了（ADモジュール0）割り込み処理



| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

プログラムリスト

```
/*******************************************************/
/* SH7046F Series -SH7047- Application Note            */
/*    A/D Conversion with DTC Transmission             */
/* Function                                            */
/* :Data transfer Controller(DTC)                      */
/* :Multi-Function Timer Pulse Unit(MTU ch0)           */
/* :A/D Converter(A/D ch0 ch1)                         */
/*                                                     */
/* External Input clock    :10MHz                      */
/* Internal CPU clock    :40MHz                        */
/* Internal Peripheral clock :40MHz                    */
/*                                                     */
/* Written   :   2001/12/01  Rev.1.0                   */
/*                                                     */
/*******************************************************/

#include "iodefine_7047v13.1.h"
#include <machine.h>

/*------------ Symbol Definition ---------------------------------------------*/
struct st_dtc_b{        /* DTC Block Transfer Mode information */
    unsigned short DTMR;       /* DTC Mode Register            */
    unsigned short DTCRA;      /* Transfer counter             */
    unsigned short dummy;      /* Reserved                     */
    unsigned short DTCRB;      /* Block length                 */
    unsigned long DTSAR;       /* source address register      */
    unsigned long DTDAR;       /* destination address register */
};

#define DTC_COUNT 3                    /* DTC Transmit count    */
#define DTC_BLOCK_LENG  8              /* DTC Block length      */


/*------------ Function Definition -------------------------------------------*/
void main(void);
void ad_adi0_dtc(void);
```

プログラムリスト

```
/*------------ RAM allocation Definition ------------------------------------*/
unsigned short Ad_data[DTC_COUNT][DTC_BLOCK_LENG];    /* buffer memory     */
#define DTC_B (*(volatile struct st_dtc_b*)0xFFFFE000)   /* DTC information
address */



/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
     /* Set standby mode  */
     P_STBY.MSTCR1.BIT.MSTP25 = 0;      /* Disable DTC standby mode       */
     P_STBY.MSTCR1.BIT.MSTP24 = 0;      /* Disable DTC standby mode       */
     P_STBY.MSTCR2.BIT.MSTP13 = 0;      /* Disable MTU standby mode       */
     P_STBY.MSTCR2.BIT.MSTP5 = 0;       /* Disable AD1 standby mode       */
     P_STBY.MSTCR2.BIT.MSTP4 = 0;       /* Disable AD0 standby mode       */
     /* Set interrupt priority level (0 to 15) */
     P_INTC.IPRG.BIT.AD01 = 8;        /* A/D ADI0,1 interrupt level8    */



     /* DTC information     */
     DTC_B.DTMR = 0xa980;        /* */
            /* SM[1:0]=b'10;    DTSAR is incremented            */
            /* DM[1:0]=b'10;    DTDAR is incremented            */
            /* MD[1:0]=b'10;    Block transfer mode             */
            /* SZ[1:0]=b'01;    word-size transfer              */
            /* DTS=b'1;     Source is block area                */
            /* CHNE=b'0;    Chain transfer is canceled          */
            /* DISEL=b'0;     Interrupt->transfer ends          */
            /* NMIM=b'0;    NMI->Terminate DTC transfer         */
```

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

プログラムリスト

```
    DTC_B.DTCRA = DTC_COUNT;           /* DTC transfer Count                  */
    DTC_B.DTCRB = DTC_BLOCK_LENG;      /* DTC transfer Block length           */
    DTC_B.DTSAR = (unsigned long)&P_AD.ADDR8.WORD; /* set source address      */
    DTC_B.DTDAR = (unsigned long)Ad_data;   /* set destination address        */

    P_DTC.DTBR= 0xFFFF;           /* DTC information base register    */
    /* DTC transmit enable      */
    P_DTC.DTEC.BIT.ADI0 |= 1;     /* interrupt sources AD ch0(ADI0)      */




    /* Initialize MTU channel 0   */
    P_MTU0.TCR_0.BYTE = 0x23;     /* */
          /* CCLR[2:0]=b'001;  TCNT cleared by TGRA compare match       */
          /* CKEG[1:0]=b'00;   Count at rising edge                     */
          /* TPSC[2:0]=b'011;  TCNT use Internal clock Pφ/64            */

    P_MTU0.TMDR_0.BYTE = 0x00;    /* TGRB,TGRD,TGRA,TGRD operate normally
    */
          /* MD[3:0]=b'0000;   Normal timer operation mode          */

    P_MTU0.TIORH_0.BYTE = 0x00;   /* TGRA_0:Output compare register, Output
disabled  */
          /* IOB[3:0]=b'0000;  TGRB_0:Output compare register, Output disabled
    */
          /* IOA[3:0]=b'0000;  TGRA_0Output compare register, Output disabled*/

    P_MTU0.TIORL_0.BYTE = 0x00;   /* */
          /* IOD[3:0]=b'0000;  TGRD_0:Output compare register, Output disabled
    */
          /* IOC[3:0]=b'0000;  TGRC_0Output compare register, Output disabled*/

    P_MTU0.TIER_0.BYTE = 0xc0;  /* */
          /* TTGE=1;  Enable, A/D conversion start by TGRA compare  */
          /* TGIEA=0;   all Interrupt requests disabled          */

    P_MTU0.TCNT_0 = 0x0000;     /* clear TCNT counter             */
    P_MTU0.TGRA_0 = 0x9c40;     /* compare match=64ms @Pφ/64  Pφ=40MHz     */
```

プログラムリスト

```
    /* Initialize A/D chanel0,chanel1  */
    P_AD.ADCSR_0.BYTE = 0x5f;   /* */
          /* ADIE=b'1;      ch0 A/D ADI0 Interrupt Enable                    */
          /* ADM[1:0]=b'01;    ch0 conversion mode -> 4channel scan mode    */
          /* CH[2:0]=b111';    Analog Input Channels = AN8 to AN11          */

    P_AD.ADCR_0.BYTE = 0xe7;  /* */
           /* TRGE=b'1;      ch0 conversion triggering is enabled     */
           /* CKS[1:0]=b'11;    ch0 Clock Select  Pφ/4                */
           /* ADST=b'0;      ch0 stops A/D conversion                 */
           /* ADCS=b'0;      ch0 Single-cycle scan                    */
    P_AD.ADCSR_1.BYTE = 0x1f;   /* */
           /* ADIE=b'0;      ch1 A/D ADI1 Interrupt disable                  */
           /* ADM[1:0]=b'01;    ch1 conversion mode -> 4channel scan mode    */
           /* CH[2:0]=b'111;    ch1 Analog Input Channels = AN12 to AN15     */
    P_AD.ADCR_1.BYTE = 0xe7;  /* */
           /* TRGE=b'1;      ch1 conversion triggering is enabled     */
           /* CKS[1:0]=b'11;    ch1 Clock Select  Pφ/4                */
           /* ADST=b'0;      ch1 stops A/D conversion                 */
           /* ADCS=b'0;      ch1 Single-cycle scan                    */


    P_AD.ADTSR.BYTE = 0x0a;   /* A/D ch0,ch1 conversion start by MTU trigger  */
           /* TRG1S[1:0]=b'10;  ch1 conversion start by MTU trigger      */
           /* TRG0S[1:0]=b'10;  ch0 conversion start by MTU trigger      */


    set_imask(0x00);        /* clear interrupt mask level        */


    P_MTU34.TSTR.BYTE = 0x01;     /* MTU ch0 timer count start       */


    while(1);

}
```

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
|---|---|---|

プログラムリスト

```
/************************************************************/
/*  ADI interrupt                                           */
/************************************************************/
#pragma interrupt(ad_adi0_dtc)
void ad_adi0_dtc(void)
{

    P_MTU34.TSTR.BYTE = 0x00;     /* MTU ch0 timer count stop          */

    P_AD.ADCSR_0.BIT.ADF &=0;     /* ch0 ADF  flag clear               */

}
```

| MTU による A/D 変換の起動および変換結果の格納（A/D、DTC） | 使用機能 | MTU、DTC、A/D |
| --- | --- | --- |
| プログラムリスト | | |

# 3. 付録

## 3.1 ヘッダファイル

```
/************************************************************************/
/*                                                                      */
/*  FILE       :iodefine.h                                              */
/*  DATE       :Fri, Nov 07, 2003                                        */
/*  DESCRIPTION :Definition of I/O Register                              */
/*  CPU TYPE   :SH7046                                                   */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.3.1).      */
/*                                                                      */
/************************************************************************/



/************************************************************************/
/*     7047 Include File                            Ver.HEW2.0_2001.12 */
/************************************************************************/
struct st_sci {                                      /* struct SCI   */
      union {                                       /* SMR          */
           unsigned char BYTE;                      /*  Byte Access */
           struct {                                 /*  Bit Access  */
                unsigned char CA:1;                 /*   C/A        */
                unsigned char CHR:1;                /*   CHR        */
                unsigned char PE:1;                 /*   PE         */
                unsigned char OE:1;                 /*   O/E        */
                unsigned char STOP:1;               /*   STOP       */
                unsigned char MP:1;                 /*   MP         */
                unsigned char CKS:2;                /*   CKS        */
                } BIT;                              /*              */
           } SMR;                                   /*              */
      unsigned char BRR;                            /* BRR          */
      union {                                       /* SCR          */
```

```
            unsigned char BYTE;                    /*  Byte Access */
            struct {                               /*  Bit Access  */
                  unsigned char TIE:1;             /*    TIE       */
                  unsigned char RIE:1;             /*    RIE       */
                  unsigned char TE:1;              /*    TE        */
                  unsigned char RE:1;              /*    RE        */
                  unsigned char MPIE:1;            /*    MPIE      */
                  unsigned char TEIE:1;            /*    TEIE      */
                  unsigned char CKE:2;             /*    CKE       */
                  } BIT;                           /*              */
            } SCR;                                 /*              */
      unsigned char TDR;                           /* TDR          */
      union {                                      /* SSR          */
            unsigned char BYTE;                    /*  Byte Access */
            struct {                               /*  Bit Access  */
                  unsigned char TDRE:1;            /*    TDRE      */
                  unsigned char RDRF:1;            /*    RDRF      */
                  unsigned char ORER:1;            /*    ORER      */
                  unsigned char FER:1;             /*    FER       */
                  unsigned char PER:1;             /*    PER       */
                  unsigned char TEND:1;            /*    TEND      */
                  unsigned char MPB:1;             /*    MPB       */
                  unsigned char MPBT:1;            /*    MPBT      */
                  } BIT;                           /*              */
            } SSR;                                 /*              */
      unsigned char RDR;                           /* RDR          */
      union {                                      /* SDCR         */
            unsigned char BYTE;                    /*  Byte Access */
            struct {                               /*  Bit Access  */
                  unsigned char :4;                /*              */
                  unsigned char DIR:1;             /*    DIR       */
                  unsigned char :3;                /*              */
                  } BIT;                           /*              */
            } SDCR;                                /*              */
};                                                 /*              */
struct st_mtu34 {                                  /* struct MTU34 */
      union {                                      /* TCR_3        */
            unsigned char BYTE;                    /*  Byte Access */
```

```
     struct {                              /*  Bit Access  */
          unsigned char CCLR:3;            /*    CCLR      */
          unsigned char CKEG:2;            /*    CKEG      */
          unsigned char TPSC:3;            /*    TPSC      */
          } BIT;                           /*              */
     } TCR_3;                              /*              */
union {                                    /* TCR_4        */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char CCLR:3;            /*    CCLR      */
          unsigned char CKEG:2;            /*    CKEG      */
          unsigned char TPSC:3;            /*    TPSC      */
          } BIT;                           /*              */
     } TCR_4;                              /*              */
union {                                    /* TMDR_3       */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char :2;                /*              */
          unsigned char BFB:1;             /*    BFB       */
          unsigned char BFA:1;             /*    BFA       */
          unsigned char MD:4;              /*    MD        */
          } BIT;                           /*              */
     } TMDR_3;                             /*              */
union {                                    /* TMDR_4       */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char :2;                /*              */
          unsigned char BFB:1;             /*    BFB       */
          unsigned char BFA:1;             /*    BFA       */
          unsigned char MD:4;              /*    MD        */
          } BIT;                           /*              */
     } TMDR_4;                             /*              */
union {                                    /* TIORH_3      */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char IOB:4;             /*    IOB       */
          unsigned char IOA:4;             /*    IOA       */
          } BIT;                           /*              */
```

```
     } TIORH_3;                          /*             */
union {                                  /* TIORL_3     */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char IOD:4;           /*   IOD        */
          unsigned char IOC:4;           /*   IOC        */
          } BIT;                         /*              */
     } TIORL_3;                          /*              */
union {                                  /* TIORH_4      */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char IOB:4;           /*   IOB        */
          unsigned char IOA:4;           /*   IOA        */
          } BIT;                         /*              */
     } TIORH_4;                          /*              */
union {                                  /* TIORL_4      */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char IOD:4;           /*   IOD        */
          unsigned char IOC:4;           /*   IOC        */
          } BIT;                         /*              */
     } TIORL_4;                          /*              */
union {                                  /* TIER_3       */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char TTGE:1;          /*   TTGE       */
          unsigned char :2;              /*              */
          unsigned char TCIEV:1;         /*   TCIEV      */
          unsigned char TGIED:1;         /*   TGIED      */
          unsigned char TGIEC:1;         /*   TGIEC      */
          unsigned char TGIEB:1;         /*   TGIEB      */
          unsigned char TGIEA:1;         /*   TGIEA      */
          } BIT;                         /*              */
     } TIER_3;                           /*              */
union {                                  /* TIER_4       */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char TTGE:1;          /*   TTGE       */
```

```
            unsigned char :2;               /*            */
            unsigned char TCIEV:1;          /*   TCIEV    */
            unsigned char TGIED:1;          /*   TGIED    */
            unsigned char TGIEC:1;          /*   TGIEC    */
            unsigned char TGIEB:1;          /*   TGIEB    */
            unsigned char TGIEA:1;          /*   TGIEA    */
            } BIT;                          /*            */
      } TIER_4;                             /*            */
union {                                     /* TOER       */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char :2;               /*              */
            unsigned char OE4D:1;           /*   OE4D       */
            unsigned char OE4C:1;           /*   OE4C       */
            unsigned char OE3D:1;           /*   OE3D       */
            unsigned char OE4B:1;           /*   OE4B       */
            unsigned char OE4A:1;           /*   OE4A       */
            unsigned char OE3B:1;           /*   OE3B       */
            } BIT;                          /*              */
      } TOER;                               /*              */
union {                                     /* TOCR         */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char :1;               /*              */
            unsigned char PSYE:1;           /*   PSYE       */
            unsigned char :4;               /*              */
            unsigned char OLSN:1;           /*   OLSN       */
            unsigned char OLSP:1;           /*   OLSP       */
            } BIT;                          /*              */
      } TOCR;                               /*             */
unsigned char wk0[1];                       /*             */
union {                                     /* TGCR        */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char :1;               /*              */
            unsigned char BDC:1;            /*   BDC        */
            unsigned char N:1;              /*   N          */
            unsigned char P:1;              /*   P          */
```

```
            unsigned char FB:1;             /*   FB        */
            unsigned char WF:1;             /*   WF        */
            unsigned char VF:1;             /*   VF        */
            unsigned char UF:1;             /*   UF        */
            } BIT;                          /*             */
      } TGCR;                               /*             */
unsigned char wk1[2];                       /*             */
unsigned short TCNT_3;                      /* TCNT_3      */
unsigned short TCNT_4;                      /* TCNT_4      */
unsigned short TCDR;                        /* TCDR        */
unsigned short TDDR;                        /* TDDR        */
unsigned short TGRA_3;                      /* TGRA_3      */
unsigned short TGRB_3;                      /* TGRB_3      */
unsigned short TGRA_4;                      /* TGRA_4      */
unsigned short TGRB_4;                      /* TGRB_4      */
unsigned short TCNTS;                       /* TCNTS       */
unsigned short TCBR;                        /* TCBR        */
unsigned short TGRC_3;                      /* TGRC_3      */
unsigned short TGRD_3;                      /* TGRD_3      */
unsigned short TGRC_4;                      /* TGRC_4      */
unsigned short TGRD_4;                      /* TGRD_4      */
union {                                     /* TSR_3       */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char TDFD:1;           /*   TDFD       */
            unsigned char :2;               /*              */
            unsigned char TCFV:1;           /*   TCFV       */
            unsigned char TGFD:1;           /*   TGFD       */
            unsigned char TGFC:1;           /*   TGFC       */
            unsigned char TGFB:1;           /*   TGFB       */
            unsigned char TGFA:1;           /*   TGFA       */
            } BIT;                          /*              */
      } TSR_3;                              /*              */
union {                                     /* TSR_4        */
      unsigned char BYTE;                   /*  Byte Access  */
      struct {                              /*  Bit Access   */
            unsigned char TDFD:1;           /*   TDFD       */
            unsigned char :2;               /*              */
```

```
                unsigned char TCFV:1;           /*   TCFV       */
                unsigned char TGFD:1;           /*   TGFD       */
                unsigned char TGFC:1;           /*   TGFC       */
                unsigned char TGFB:1;           /*   TGFB       */
                unsigned char TGFA:1;           /*   TGFA       */
                } BIT;                          /*              */
          } TSR_4;                              /*              */
     unsigned char wk2[18];                     /*              */
     union {                                    /* TSTR         */
          unsigned char BYTE;                   /*  Byte Access */
          struct {                              /*  Bit Access  */
                unsigned char CST4:1;           /*   CST4      */
                unsigned char CST3:1;           /*   CST3       */
                unsigned char :3;               /*              */
                unsigned char CST:3;            /*   CST        */
                } BIT;                          /*              */
          } TSTR;                               /*              */
     union {                                    /* TSYR         */
          unsigned char BYTE;                   /*  Byte Access */
          struct {                              /*  Bit Access  */
                unsigned char SYNC4:1;          /*   SYNC4      */
                unsigned char SYNC3:1;          /*   SYNC3      */
                unsigned char :3;               /*              */
                unsigned char SYNC2:1;          /*   SYNC2      */
                unsigned char SYNC1:1;          /*   SYNC1      */
                unsigned char SYNC0:1;          /*   SYNC0      */
                } BIT;                          /*             */
          } TSYR;                               /*              */
};                                              /*              */
struct st_mtu0 {                                /* struct MTU0  */
     union {                                    /* TCR_0        */
          unsigned char BYTE;                   /*  Byte Access */
          struct {                              /*  Bit Access  */
                unsigned char CCLR:3;           /*   CCLR       */
                unsigned char CKEG:2;           /*   CKEG       */
                unsigned char TPSC:3;           /*   TPSC       */
                } BIT;                          /*              */
          } TCR_0;                              /*              */
```

```
union {                                 /* TMDR_0       */
     unsigned char BYTE;                 /*  Byte Access */
     struct {                            /*  Bit Access  */
          unsigned char :2;              /*              */
          unsigned char BFB:1;           /*   BFB        */
          unsigned char BFA:1;           /*   BFA        */
          unsigned char MD:4;            /*   MD         */
          } BIT;                        /*              */
     } TMDR_0;                          /*              */
union {                                 /* TIORH_0      */
     unsigned char BYTE;                /*  Byte Access */
     struct {                           /*  Bit Access  */
          unsigned char IOB:4;          /*   IOB        */
          unsigned char IOA:4;          /*   IOA        */
          } BIT;                        /*              */
     } TIORH_0;                         /*              */
union {                                 /* TIORL_0      */
     unsigned char BYTE;                /*  Byte Access */
     struct {                           /*  Bit Access  */
          unsigned char IOD:4;          /*   IOD        */
          unsigned char IOC:4;          /*   IOC        */
          } BIT;                        /*              */
     } TIORL_0;                         /*              */
union {                                 /* TIER_0       */
     unsigned char BYTE;                /*  Byte Access */
     struct {                           /*  Bit Access  */
          unsigned char TTGE:1;         /*   TTGE       */
          unsigned char :2;             /*              */
          unsigned char TCIEV:1;        /*   TCIEV      */
          unsigned char TGIED:1;        /*   TGIED      */
          unsigned char TGIEC:1;        /*   TGIEC      */
          unsigned char TGIEB:1;        /*   TGIEB      */
          unsigned char TGIEA:1;        /*   TGIEA      */
          } BIT;                        /*              */
     } TIER_0;                          /*              */
union {                                 /* TSR_0        */
     unsigned char BYTE;                /*  Byte Access */
     struct {                           /*  Bit Access  */
```

```
                unsigned char :3;               /*             */
                unsigned char TCFV:1;           /*   TCFV      */
                unsigned char TGFD:1;           /*   TGFD      */
                unsigned char TGFC:1;           /*   TGFC      */
                unsigned char TGFB:1;           /*   TGFB      */
                unsigned char TGFA:1;           /*   TGFA      */
                } BIT;                          /*            */
          } TSR_0;                              /*            */
     unsigned short TCNT_0;                     /* TCNT_0     */
     unsigned short TGRA_0;                     /* TGRA_0     */
     unsigned short TGRB_0;                     /* TGRB_0     */
     unsigned short TGRC_0;                     /* TGRC_0     */
     unsigned short TGRD_0;                     /* TGRD_0     */
};                                              /*            */
struct st_mtu1 {                               /* struct MTU1  */
     union {                                   /* TCR_1        */
          unsigned char BYTE;                  /*  Byte Access */
          struct {                             /*  Bit Access  */
                unsigned char :1;              /*              */
                unsigned char CCLR:2;          /*   CCLR      */
                unsigned char CKEG:2;          /*   CKEG      */
                unsigned char TPSC:3;          /*   TPSC      */
                } BIT;                         /*             */
          } TCR_1;                             /*             */
     union {                                   /* TMDR_1       */
          unsigned char BYTE;                  /*  Byte Access */
          struct {                             /*  Bit Access   */
                unsigned char :4;              /*              */
                unsigned char MD:4;            /*   MD         */
                } BIT;                         /*              */
          } TMDR_1;                            /*              */
     union {                                   /* TIOR_1       */
          unsigned char BYTE;                  /*  Byte Access */
          struct {                             /*  Bit Access   */
                unsigned char IOB:4;           /*   IOB        */
                unsigned char IOA:4;           /*   IOA        */
                } BIT;                         /*              */
          } TIOR_1;                            /*              */
```

```
        unsigned char wk0[1];                    /*             */
        union {                                  /* TIER_1      */
             unsigned char BYTE;                 /*  Byte Access */
             struct {                            /*  Bit Access  */
                  unsigned char TTGE:1;          /*   TTGE       */
                  unsigned char :1;              /*              */
                  unsigned char TCIEU:1;         /*   TCIEU      */
                  unsigned char TCIEV:1;         /*   TCIEV      */
                  unsigned char :2;              /*              */
                  unsigned char TGIEB:1;         /*   TGIEB      */
                  unsigned char TGIEA:1;         /*   TGIEA      */
                  } BIT;                         /*              */
             } TIER_1;                           /*             */
        union {                                  /* TSR_1       */
             unsigned char BYTE;                 /*  Byte Access */
             struct {                            /*  Bit Access  */
                  unsigned char TCFD:1;          /*   TCFD       */
                  unsigned char :1;              /*              */
                  unsigned char TCFU:1;          /*   TCFU       */
                  unsigned char TCFV:1;          /*   TCFV       */
                  unsigned char :2;              /*              */
                  unsigned char TGFB:1;          /*   TGFB       */
                  unsigned char TGFA:1;          /*   TGFA       */
                  } BIT;                         /*              */
             } TSR_1;                            /*              */
        unsigned short TCNT_1;                   /* TCNT_1       */
        unsigned short TGRA_1;                   /* TGRA_1       */
        unsigned short TGRB_1;                   /* TGRB_1       */
   };                                            /*              */
   struct st_mtu2 {                              /* struct MTU2  */
        union {                                  /* TCR_2        */
             unsigned char BYTE;                 /*  Byte Access */
             struct {                            /*  Bit Access  */
                  unsigned char :1;              /*             */
                  unsigned char CCLR:2;          /*   CCLR      */
                  unsigned char CKEG:2;          /*   CKEG      */
                  unsigned char TPSC:3;          /*   TPSC      */
                  } BIT;                         /*             */
```

```
     } TCR_2;                              /*            */
union {                                    /* TMDR_2     */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char :4;                /*              */
          unsigned char MD:4;              /*   MD         */
          } BIT;                           /*              */
     } TMDR_2;                             /*              */
union {                                    /* TIOR_2       */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char IOB:4;             /*   IOB        */
          unsigned char IOA:4;             /*   IOA        */
          } BIT;                           /*              */
     } TIOR_2;                             /*              */
unsigned char wk0[1];                      /*              */
union {                                    /* TIER_2       */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char TTGE:1;            /*   TTGE       */
          unsigned char :1;                /*              */
          unsigned char TCIEU:1;           /*   TCIEU      */
          unsigned char TCIEV:1;           /*   TCIEV      */
          unsigned char :2;                /*              */
          unsigned char TGIEB:1;           /*   TGIEB      */
          unsigned char TGIEA:1;           /*   TGIEA      */
          } BIT;                           /*              */
     } TIER_2;                             /*              */
union {                                    /* TSR_2        */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char TCFD:1;            /*   TCFD       */
          unsigned char :1;                /*              */
          unsigned char TCFU:1;            /*   TCFU       */
          unsigned char TCFV:1;            /*   TCFV       */
          unsigned char :2;                /*              */
          unsigned char TGFB:1;            /*   TGFB       */
          unsigned char TGFA:1;            /*   TGFA       */
```

```
                } BIT;                          /*            */
            } TSR_2;                            /*            */
        unsigned short TCNT_2;                  /* TCNT_2     */
        unsigned short TGRA_2;                  /* TGRA_2     */
        unsigned short TGRB_2;                  /* TGRB_2     */
};                                              /*            */
struct st_intc {                                /* struct INTC  */
        union {                                 /* IPRA         */
            unsigned short WORD;                /*  Word Access */
            struct {                            /*  Bit Access  */
                unsigned short IRQ0:4;          /*    IRQ0      */
                unsigned short IRQ1:4;          /*    IRQ1      */
                unsigned short IRQ2:4;          /*    IRQ2      */
                unsigned short IRQ3:4;          /*    IRQ3      */
                } BIT;                          /*              */
            } IPRA;                             /*              */
        unsigned char wk0[4];                   /*              */
        union {                                 /* IPRD         */
            unsigned short WORD;                /*  Word Access */
            struct {                            /*  Bit Access  */
                unsigned short TGI_0:4;         /*    TGI_0     */
                unsigned short TCI_0:4;         /*    TCI_0     */
                unsigned short TGI_1:4;         /*    TGI_1     */
                unsigned short TCI_1:4;         /*    TCI_1     */
                } BIT;                          /*              */
            } IPRD;                             /*              */
        union {                                 /* IPRE         */
            unsigned short WORD;                /*  Word Access */
            struct {                            /*  Bit Access  */
                unsigned short TGI_2:4;         /*    TGI_2     */
                unsigned short TCI_2:4;         /*    TCI_2     */
                unsigned short TGI_3:4;         /*    TGI_3     */
                unsigned short TCI_3:4;         /*    TCI_3     */
                } BIT;                          /*              */
            } IPRE;                             /*              */
        union {                                 /* IPRF         */
            unsigned short WORD;                /*  Word Access */
            struct {                            /*  Bit Access  */
```

```
            unsigned short TGI_4:4;          /*   TGI_4     */
            unsigned short TCI_4:4;          /*   TCI_4     */
            unsigned short :8;               /*             */
            } BIT;                           /*             */
      } IPRF;                                /*             */
union {                                      /* IPRG        */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short AD01:4;           /*   A/D0,1     */
            unsigned short DTC:4;            /*   DTC        */
            unsigned short CMT0:4;           /*   CMT0       */
            unsigned short CMT1:4;           /*   CMT1       */
            } BIT;                           /*              */
      } IPRG;                                /*              */
union {                                      /* IPRH         */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short WDT:4;            /*   WDT        */
            unsigned short IOMTU:4;          /*   I/O(MTU)   */
            unsigned short :8;               /*              */
            } BIT;                           /*             */
      } IPRH;                                /*             */
union {                                      /* ICR1        */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short NMIL:1;           /*   NMIL       */
            unsigned short :6;               /*             */
            unsigned short NMIE:1;           /*   NMIE       */
            unsigned short IRQ0S:1;          /*   IRQ0S      */
            unsigned short IRQ1S:1;          /*   IRQ1S      */
            unsigned short IRQ2S:1;          /*   IRQ2S      */
            unsigned short IRQ3S:1;          /*   IRQ3S      */
            unsigned short :4;               /*              */
            } BIT;                           /*             */
      } ICR1;                                /*             */
union {                                      /* ISR         */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
```

```
            unsigned short :8;              /*            */
            unsigned short IRQ0F:1;         /*   IRQ0F    */
            unsigned short IRQ1F:1;         /*   IRQ1F    */
            unsigned short IRQ2F:1;         /*   IRQ2F    */
            unsigned short IRQ3F:1;         /*   IRQ3F    */
            unsigned short :4;              /*            */
            } BIT;                          /*            */
      } ISR;                                /*            */
union {                                     /* IPRI       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short SCI2:4;          /*   SCI2      */
            unsigned short SCI3:4;          /*   SCI3      */
            unsigned short SCI4:4;          /*   SCI4      */
            unsigned short MMT:4;           /*   MMT       */
            } BIT;                          /*            */
      } IPRI;                               /*            */
union {                                     /* IPRJ       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short AD2:4;           /*   A/D2      */
            unsigned short :12;             /*            */
            } BIT;                          /*            */
      } IPRJ;                               /*            */
union {                                     /* IPRK       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short IOMMT:4;         /*   I/O(MMT)  */
            unsigned short :4;              /*            */
            unsigned short HCAN2:4;         /*   HCAN1     */
            unsigned short :4;              /*            */
            } BIT;                          /*            */
      } IPRK;                               /*            */
unsigned char wk1[4];                       /*            */
union {                                     /* ICR2       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short IRQ0ES:2;        /*   IRQ0ES    */
```

```
                unsigned short IRQ1ES:2;        /*   IRQ1ES    */
                unsigned short IRQ2ES:2;        /*   IRQ2ES    */
                unsigned short IRQ3ES:2;        /*   IRQ3ES    */
                unsigned short :8;              /*             */
                } BIT;                          /*             */
          } ICR2;                               /*             */
};                                              /*             */
struct st_porta {                               /* struct PORTA */
     union {                                    /* PADRL       */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short PA15DR:1;        /*   PA15DR     */
                unsigned short PA14DR:1;        /*   PA14DR     */
                unsigned short PA13DR:1;        /*   PA13DR     */
                unsigned short PA12DR:1;        /*   PA12DR     */
                unsigned short PA11DR:1;        /*   PA11DR     */
                unsigned short PA10DR:1;        /*   PA10DR     */
                unsigned short PA9DR:1;         /*   PA9DR      */
                unsigned short PA8DR:1;         /*   PA8DR      */
                unsigned short PA7DR:1;         /*   PA7DR      */
                unsigned short PA6DR:1;         /*   PA6DR      */
                unsigned short PA5DR:1;         /*   PA5DR      */
                unsigned short PA4DR:1;         /*   PA4DR      */
                unsigned short PA3DR:1;         /*   PA3DR      */
                unsigned short PA2DR:1;         /*   PA2DR      */
                unsigned short PA1DR:1;         /*   PA1DR      */
                unsigned short PA0DR:1;         /*   PA0DR      */
                } BIT;                          /*             */
          } PADRL;                              /*             */
     unsigned char wk0[2];                      /*             */
     union {                                   /* PAIORL       */
          unsigned short WORD;                 /*  Word Access */
          struct {                             /*  Bit Access  */
                unsigned short PA15IOR:1;      /*   PA15IOR    */
                unsigned short PA14IOR:1;      /*   PA14IOR    */
                unsigned short PA13IOR:1;      /*   PA13IOR    */
                unsigned short PA12IOR:1;      /*   PA12IOR    */
                unsigned short PA11IOR:1;      /*   PA11IOR    */
```

```
            unsigned short PA10IOR:1;        /*   PA10IOR   */
            unsigned short PA9IOR:1;         /*   PA9IOR    */
            unsigned short PA8IOR:1;         /*   PA8IOR    */
            unsigned short PA7IOR:1;         /*   PA7IOR    */
            unsigned short PA6IOR:1;         /*   PA6IOR    */
            unsigned short PA5IOR:1;         /*   PA5IOR    */
            unsigned short PA4IOR:1;         /*   PA4IOR    */
            unsigned short PA3IOR:1;         /*   PA3IOR    */
            unsigned short PA2IOR:1;         /*   PA2IOR    */
            unsigned short PA1IOR:1;         /*   PA1IOR    */
            unsigned short PA0IOR:1;         /*   PA0IOR    */
            } BIT;                           /*             */
      } PAIORL;                              /*             */
unsigned char wk1[2];                        /*             */
union {                                      /* PACRL3      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short PA15MD2:1;        /*   PA15MD2   */
            unsigned short PA14MD2:1;        /*   PA14MD2   */
            unsigned short PA13MD2:1;        /*   PA13MD2   */
            unsigned short PA12MD2:1;        /*   PA12MD2   */
            unsigned short PA11MD2:1;        /*   PA11MD2   */
            unsigned short PA10MD2:1;        /*   PA10MD2   */
            unsigned short PA9MD2:1;         /*   PA9MD2    */
            unsigned short PA8MD2:1;         /*   PA8MD2    */
            unsigned short PA7MD2:1;         /*   PA7MD2    */
            unsigned short PA6MD2:1;         /*   PA6MD2    */
            unsigned short PA5MD2:1;         /*   PA5MD2    */
            unsigned short PA4MD2:1;         /*   PA4MD2    */
            unsigned short PA3MD2:1;         /*   PA3MD2    */
            unsigned short PA2MD2:1;         /*   PA2MD2    */
            unsigned short PA1MD2:1;         /*   PA1MD2    */
            unsigned short PA0MD2:1;         /*   PA0MD2    */
            } BIT;                           /*              */
      } PACRL3;                              /*             */
union {                                      /* PACRL1       */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
```

```
                unsigned short PA15MD:2;          /*   PA15MD    */
                unsigned short PA14MD:2;          /*   PA14MD    */
                unsigned short PA13MD:2;          /*   PA13MD    */
                unsigned short PA12MD:2;          /*   PA12MD    */
                unsigned short PA11MD:2;          /*   PA11MD    */
                unsigned short PA10MD:2;          /*   PA10MD    */
                unsigned short PA9MD:2;           /*   PA9MD     */
                unsigned short PA8MD:2;           /*   PA8MD     */
                } BIT;                            /*             */
          } PACRL1;                               /*             */
     union {                                     /* PACRL2       */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short PA7MD:2;           /*   PA7MD     */
                unsigned short PA6MD:2;           /*   PA6MD     */
                unsigned short PA5MD:2;           /*   PA5MD     */
                unsigned short PA4MD:2;           /*   PA4MD     */
                unsigned short PA3MD:2;           /*   PA3MD     */
                unsigned short PA2MD:2;           /*   PA2MD     */
                unsigned short PA1MD:2;           /*   PA1MD     */
                unsigned short PA0MD:2;           /*   PA0MD     */
                } BIT;                            /*            */
          } PACRL2;                               /*             */
};                                                /*             */
struct st_portb {                                 /* struct PORTB */
     union {                                     /* PBDR         */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short :10;               /*             */
                unsigned short PB5DR:1;           /*   PB5DR     */
                unsigned short PB4DR:1;           /*   PB4DR     */
                unsigned short PB3DR:1;           /*   PB3DR     */
                unsigned short PB2DR:1;           /*   PB2DR     */
                unsigned short PB1DR:1;           /*   PB1DR     */
                unsigned short PB0DR:1;           /*   PB0DR     */
                } BIT;                            /*            */
          } PBDR;                                 /*            */
     unsigned char wk0[2];                        /*            */
```

```
        union {                                  /* PBIOR       */
             unsigned short WORD;                /*  Word Access */
             struct {                            /*  Bit Access  */
                  unsigned short :10;            /*              */
                  unsigned short PB5IOR:1;       /*   PB5IOR     */
                  unsigned short PB4IOR:1;       /*   PB4IOR     */
                  unsigned short PB3IOR:1;       /*   PB3IOR     */
                  unsigned short PB2IOR:1;       /*   PB2IOR     */
                  unsigned short PB1IOR:1;       /*   PB1IOR     */
                  unsigned short PB0IOR:1;       /*   PB0IOR     */
                  } BIT;                         /*              */
             } PBIOR;                            /*              */
        unsigned char wk1[2];                    /*              */
        union {                                  /* PBCR1        */
             unsigned short WORD;                /*  Word Access */
             struct {                            /*  Bit Access  */
                  unsigned short :2;             /*              */
                  unsigned short PB5MD2:1;       /*   PB5MD2     */
                  unsigned short PB4MD2:1;       /*   PB4MD2     */
                  unsigned short PB3MD2:1;       /*   PB3MD2     */
                  unsigned short PB2MD2:1;       /*   PB2MD2     */
                  unsigned short PB1MD2:1;       /*   PB1MD2     */
                  unsigned short :9;             /*              */
                  } BIT;                         /*              */
             } PBCR1;                            /*              */
        union {                                  /* PBCR2       */
             unsigned short WORD;                /*  Word Access */
             struct {                            /*  Bit Access  */
                  unsigned short :4;             /*              */
                  unsigned short PB5MD:2;        /*   PB5MD      */
                  unsigned short PB4MD:2;        /*   PB4MD      */
                  unsigned short PB3MD:2;        /*   PB3MD      */
                  unsigned short PB2MD:2;        /*   PB2MD      */
                  unsigned short PB1MD:2;        /*   PB1MD      */
                  unsigned short PB0MD:2;        /*   PB0MD      */
                  } BIT;                         /*              */
             } PBCR2;                            /*              */
   };                                            /*              */
```

```
struct st_portd {                                /* struct PORTD */
      union {                                    /* PDDRL       */
           unsigned short WORD;                  /*  Word Access */
           struct {                              /*  Bit Access  */
                unsigned short :7;               /*              */
                unsigned short PD8DR:1;          /*   PD8DR      */
                unsigned short PD7DR:1;          /*   PD7DR      */
                unsigned short PD6DR:1;          /*   PD6DR      */
                unsigned short PD5DR:1;          /*   PD5DR      */
                unsigned short PD4DR:1;          /*   PD4DR      */
                unsigned short PD3DR:1;          /*   PD3DR      */
                unsigned short PD2DR:1;          /*   PD2DR      */
                unsigned short PD1DR:1;          /*   PD1DR      */
                unsigned short PD0DR:1;          /*   PD0DR      */
                } BIT;                           /*              */
           } PDDRL;                              /*              */
      unsigned char wk0[2];                      /*              */
      union {                                    /* PDIORL       */
           unsigned short WORD;                  /*  Word Access */
           struct {                              /*  Bit Access  */
                unsigned short :7;               /*              */
                unsigned short PD8IOR:1;         /*   PD8IOR     */
                unsigned short PD7IOR:1;         /*   PD7IOR     */
                unsigned short PD6IOR:1;         /*   PD6IOR     */
                unsigned short PD5IOR:1;         /*   PD5IOR     */
                unsigned short PD4IOR:1;         /*   PD4IOR     */
                unsigned short PD3IOR:1;         /*   PD3IOR     */
                unsigned short PD2IOR:1;         /*   PD2IOR     */
                unsigned short PD1IOR:1;         /*   PD1IOR     */
                unsigned short PD0IOR:1;         /*   PD0IOR     */
                } BIT;                           /*              */
           } PDIORL;                             /*              */
      unsigned char wk1[4];                      /*              */
      union {                                    /* PDCRL1       */
           unsigned short WORD;                  /*  Word Access */
           struct {                              /*  Bit Access  */
                unsigned short :7;               /*              */
                unsigned short PD8MD0:1;         /*   PD8MD0     */
```

```
                unsigned short PD7MD0:1;          /*   PD7MD0   */
                unsigned short PD6MD0:1;          /*   PD6MD0   */
                unsigned short PD5MD0:1;          /*   PD5MD0   */
                unsigned short PD4MD0:1;          /*   PD4MD0   */
                unsigned short PD3MD0:1;          /*   PD3MD0   */
                unsigned short PD2MD0:1;          /*   PD2MD0   */
                unsigned short PD1MD0:1;          /*   PD1MD0   */
                unsigned short PD0MD0:1;          /*   PD0MD0   */
                } BIT;                            /*            */
          } PDCRL1;                               /*            */
     union {                                       /* PDCRL2     */
          unsigned short WORD;                     /*  Word Access */
          struct {                                 /*  Bit Access  */
                unsigned short :7;                 /*             */
                unsigned short PD8MD1:1;           /*   PD8MD1    */
                unsigned short PD7MD1:1;           /*   PD7MD1    */
                unsigned short PD6MD1:1;           /*   PD6MD1    */
                unsigned short PD5MD1:1;           /*   PD5MD1    */
                unsigned short PD4MD1:1;           /*   PD4MD1    */
                unsigned short PD3MD1:1;           /*   PD3MD1    */
                unsigned short PD2MD1:1;           /*   PD2MD1    */
                unsigned short PD1MD1:1;           /*   PD1MD1    */
                unsigned short PD0MD1:1;           /*   PD0MD1    */
                } BIT;                             /*             */
          } PDCRL2;                                /*             */
};                                                 /*             */
struct st_porte {                                  /* struct PORTE */
     union {                                       /* PEDRL        */
          unsigned short WORD;                     /*  Word Access */
          struct {                                 /*  Bit Access  */
                unsigned short PE15DR:1;           /*   PE15DR    */
                unsigned short PE14DR:1;           /*   PE14DR    */
                unsigned short PE13DR:1;           /*   PE13DR    */
                unsigned short PE12DR:1;           /*   PE12DR    */
                unsigned short PE11DR:1;           /*   PE11DR    */
                unsigned short PE10DR:1;           /*   PE10DR    */
                unsigned short PE9DR:1;            /*   PE9DR     */
                unsigned short PE8DR:1;            /*   PE8DR     */
```

```
            unsigned short PE7DR:1;          /*   PE7DR    */
            unsigned short PE6DR:1;          /*   PE6DR    */
            unsigned short PE5DR:1;          /*   PE5DR    */
            unsigned short PE4DR:1;          /*   PE4DR    */
            unsigned short PE3DR:1;          /*   PE3DR    */
            unsigned short PE2DR:1;          /*   PE2DR    */
            unsigned short PE1DR:1;          /*   PE1DR    */
            unsigned short PE0DR:1;          /*   PE0DR    */
            } BIT;                           /*            */
      } PEDRL;                               /*            */
unsigned char wk0[2];                        /*            */
union {                                      /* PEIORL     */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short PE15IOR:1;        /*   PE15IOR   */
            unsigned short PE14IOR:1;        /*   PE14IOR   */
            unsigned short PE13IOR:1;        /*   PE13IOR   */
            unsigned short PE12IOR:1;        /*   PE12IOR   */
            unsigned short PE11IOR:1;        /*   PE11IOR   */
            unsigned short PE10IOR:1;        /*   PE10IOR   */
            unsigned short PE9IOR:1;         /*   PE9IOR    */
            unsigned short PE8IOR:1;         /*   PE8IOR    */
            unsigned short PE7IOR:1;         /*   PE7IOR    */
            unsigned short PE6IOR:1;         /*   PE6IOR    */
            unsigned short PE5IOR:1;         /*   PE5IOR    */
            unsigned short PE4IOR:1;         /*   PE4IOR    */
            unsigned short PE3IOR:1;         /*   PE3IOR    */
            unsigned short PE2IOR:1;         /*   PE2IOR    */
            unsigned short PE1IOR:1;         /*   PE1IOR    */
            unsigned short PE0IOR:1;         /*   PE0IOR    */
            } BIT;                           /*             */
      } PEIORL;                              /*             */
union {                                      /* PEIORH      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access   */
            unsigned short :10;              /*              */
            unsigned short PE21IOR:1;        /*   PE21IOR    */
            unsigned short PE20IOR:1;        /*   PE20IOR    */
```

```
            unsigned short PE19IOR:1;        /*   PE19IOR   */
            unsigned short PE18IOR:1;        /*   PE18IOR   */
            unsigned short PE17IOR:1;        /*   PE17IOR   */
            unsigned short PE16IOR:1;        /*   PE16IOR   */
            } BIT;                           /*             */
      } PEIORH;                              /*             */
union {                                      /* PECRL1      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short PE15MD:2;         /*   PE15MD     */
            unsigned short PE14MD:2;         /*   PE14MD     */
            unsigned short PE13MD:2;         /*   PE13MD     */
            unsigned short PE12MD:2;         /*   PE12MD    */
            unsigned short PE11MD:2;         /*   PE11MD     */
            unsigned short PE10MD:2;         /*   PE10MD     */
            unsigned short PE9MD:2;          /*   PE9MD      */
            unsigned short PE8MD:2;          /*   PE8MD      */
            } BIT;                           /*             */
      } PECRL1;                              /*             */
union {                                      /* PECRL2      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short PE7MD:2;          /*   PE7MD      */
            unsigned short PE6MD:2;          /*   PE6MD      */
            unsigned short PE5MD:2;          /*   PE5MD      */
            unsigned short PE4MD:2;          /*   PE4MD      */
            unsigned short PE3MD:2;          /*   PE3MD      */
            unsigned short PE2MD:2;          /*   PE2MD      */
            unsigned short PE1MD:2;          /*   PE1MD      */
            unsigned short PE0MD:2;          /*   PE0MD      */
            } BIT;                           /*             */
      } PECRL2;                              /*             */
union {                                      /* PECRH       */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Acces s  */
            unsigned short :4;               /*             */
            unsigned short PE21MD:2;         /*   PE21MD     */
            unsigned short PE20MD:2;         /*   PE20MD     */
```

```
                unsigned short PE19MD:2;          /*   PE19MD    */
                unsigned short PE18MD:2;          /*   PE18MD    */
                unsigned short PE17MD:2;          /*   PE17MD    */
                unsigned short PE16MD:2;          /*   PE16MD    */
                } BIT;                            /*             */
          } PECRH;                                /*             */
     union {                                      /* PEDRH       */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short :10;               /*              */
                unsigned short PE21DR:1;          /*   PE21DR     */
                unsigned short PE20DR:1;          /*   PE20DR     */
                unsigned short PE19DR:1;          /*   PE19DR     */
                unsigned short PE18DR:1;          /*   PE18DR     */
                unsigned short PE17DR:1;          /*   PE17DR     */
                unsigned short PE16DR:1;          /*   PE16DR     */
                } BIT;                            /*             */
          } PEDRH;                                /*             */
};                                                /*             */
struct st_portf {                                 /* struct PORTF */
     union {                                      /* PFDR         */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short PF15DR:1;          /*   PF15DR    */
                unsigned short PF14DR:1;          /*   PF14DR    */
                unsigned short PF13DR:1;          /*   PF13DR    */
                unsigned short PF12DR:1;          /*   PF12DR    */
                unsigned short PF11DR:1;          /*   PF11DR    */
                unsigned short PF10DR:1;          /*   PF10DR    */
                unsigned short PF9DR:1;           /*   PF9DR     */
                unsigned short PF8DR:1;           /*   PF8DR     */
                unsigned short PF7DR:1;           /*   PF7DR     */
                unsigned short PF6DR:1;           /*   PF6DR     */
                unsigned short PF5DR:1;           /*   PF5DR     */
                unsigned short PF4DR:1;           /*   PF4DR     */
                unsigned short PF3DR:1;           /*   PF3DR     */
                unsigned short PF2DR:1;           /*   PF2DR     */
                unsigned short PF1DR:1;           /*   PF1DR     */
```

```
                unsigned short PF0DR:1;          /*   PF0DR     */
                } BIT;                           /*             */
          } PFDR;                                /*             */
};                                               /*             */
struct st_mtu {                                  /* struct MTU  */
     union {                                     /* ICSR1       */
          unsigned short WORD;                   /*  Word Access */
          struct {                               /*  Bit Access  */
                unsigned short POE3F:1;          /*   POE3F     */
                unsigned short POE2F:1;          /*   POE2F     */
                unsigned short POE1F:1;          /*   POE1F     */
                unsigned short POE0F:1;          /*   POE0F     */
                unsigned short :3;               /*             */
                unsigned short PIE:1;            /*   PIE       */
                unsigned short POE3M:2;          /*   POE3M     */
                unsigned short POE2M:2;          /*   POE2M     */
                unsigned short POE1M:2;          /*   POE1M     */
                unsigned short POE0M:2;          /*   POE0M     */
                } BIT;                           /*             */
          } ICSR1;                               /*             */
     union {                                     /* OCSR        */
          unsigned short WORD;                   /*  Word Access */
          struct {                               /*  Bit Access  */
                unsigned short OSF:1;            /*   OSF       */
                unsigned short :5;               /*             */
                unsigned short OCE:1;            /*   OCE       */
                unsigned short OIE:1;            /*   OIE       */
                unsigned short :8;               /*             */
                } BIT;                           /*             */
          } OCSR;                                /*             */
};                                               /*             */
struct st_mmt {                                  /* struct MMT  */
     union {                                     /* ICSR2       */
          unsigned short WORD;                   /*  Word Access */
          struct {                               /*  Bit Access  */
                unsigned short :1;               /*             */
                unsigned short POE6F:1;          /*   POE6F     */
                unsigned short POE5F:1;          /*   POE5F     */
```

```
            unsigned short POE4F:1;          /*   POE4F     */
            unsigned short :3;               /*             */
            unsigned short PIE:1;            /*   PIE       */
            unsigned short :2;               /*             */
            unsigned short POE6M:2;          /*   POE6M     */
            unsigned short POE5M:2;          /*   POE5M     */
            unsigned short POE4M:2;          /*   POE4M      */
            } BIT;                           /*             */
      } ICSR2;                               /*             */
unsigned char wk0[1594];                     /*             */
union {                                      /* MMT_TMDR    */
      unsigned char BYTE;                    /*  Byte Access */
      struct {                               /*  Bit Access  */
            unsigned char CKS:4;             /*   CKS        */
            unsigned char OLSN:1;            /*   OLSN       */
            unsigned char OLSP:1;            /*   OLSP       */
            unsigned char MD:2;              /*   MD         */
            } BIT;                           /*              */
      } MMT_TMDR;                            /*              */
unsigned char wk1[1];                        /*              */
union {                                      /* TCNR         */
      unsigned char BYTE;                    /*  Byte Access */
      struct {                               /*  Bit Access  */
            unsigned char TTGE:1;            /*   TTGE       */
            unsigned char CST:1;             /*   CST        */
            unsigned char RPRO:1;            /*   RPRO       */
            unsigned char :3;                /*              */
            unsigned char TGIEN:1;           /*   TGIEN      */
            unsigned char TGIEM:1;           /*   TGIEM      */
            } BIT;                           /*              */
      } TCNR;                                /*              */
unsigned char wk2[1];                        /*              */
union {                                      /* MMT_TSR     */
      unsigned char BYTE;                    /*  Byte Access */
      struct {                               /*  Bit Access  */
            unsigned char TCFD:1;            /*   TCFD       */
            unsigned char :5;                /*              */
            unsigned char TGFN:1;            /*   TGFN       */
```

```
                unsigned char TGFM:1;          /*   TGFM      */
                } BIT;                        /*              */
           } MMT_TSR;                         /*              */
      unsigned char wk3[1];                   /*              */
      unsigned short MMT_TCNT;                /* MMT_TCNT     */
      unsigned short TPDR;                    /* TPDR         */
      unsigned short TPBR;                    /* TPBR         */
      unsigned short MMT_TDDR;                /* MMT_TDDR     */
      unsigned char wk4[2];                   /*              */
      unsigned short TBRU_B;                  /* TBRU_B       */
      unsigned short TGRUU;                   /* TGRUU        */
      unsigned short TGRU;                    /* TGRU         */
      unsigned short TGRUD;                   /* TGRUD        */
      unsigned short TDCNT0;                  /* TDCNT0       */
      unsigned short TDCNT1;                  /* TDCNT1       */
      unsigned short TBRU_F;                  /* TBRU_F       */
      unsigned char wk5[2];                   /*              */
      unsigned short TBRV_B;                  /* TBRV_B       */
      unsigned short TGRVU;                   /* TGRVU        */
      unsigned short TGRV;                    /* TGRV         */
      unsigned short TGRVD;                   /* TGRVD        */
      unsigned short TDCNT2;                  /* TDCNT2       */
      unsigned short TDCNT3;                  /* TDCNT3       */
      unsigned short TBRV_F;                  /* TBRV_F       */
      unsigned char wk6[2];                   /*              */
      unsigned short TBRW_B;                  /* TBRW_B       */
      unsigned short TGRWU;                   /* TGRWU        */
      unsigned short TGRW;                    /* TGRW         */
      unsigned short TGRWD;                   /* TGRWD        */
      unsigned short TDCNT4;                  /* TDCNT4       */
      unsigned short TDCNT5;                  /* TDCNT5       */
      unsigned short TBRW_F;                  /* TBRW_F       */
};                                            /*              */
struct st_portg {                             /* struct PORTG */
      union {                                 /* PGDR         */
           unsigned char BYTE;                /*  Byte Access */
           struct {                           /*  Bit Access  */
                unsigned char :4;             /*              */
```

```
                unsigned char PG3DR:1;          /*   PG3DR    */
                unsigned char PG2DR:1;          /*   PG2DR    */
                unsigned char PG1DR:1;          /*   PG1DR    */
                unsigned char PG0DR:1;          /*   PG0DR    */
                } BIT;                         /*             */
          } PGDR;                              /*             */
};                                             /*             */
struct st_cmt {                                /* struct CMT   */
      union {                                  /* CMSTR        */
           unsigned short WORD;                /*  Word Access */
           struct {                            /*  Bit Access  */
                unsigned short :14;            /*              */
                unsigned short STR:2;          /*   STR        */
                } BIT;                         /*              */
           } CMSTR;                            /*              */
      union {                                  /* CMCSR_0      */
           unsigned short WORD;                /*  Word Access */
           struct {                            /*  Bit Access  */
                unsigned short :8;             /*              */
                unsigned short CMF:1;          /*   CMF        */
                unsigned short CMIE:1;         /*   CMIE       */
                unsigned short :4;             /*              */
                unsigned short CKS:2;          /*   CKS        */
                } BIT;                         /*             */
           } CMCSR_0;                          /*             */
      unsigned short CMCNT_0;                  /* CMCNT_0      */
      unsigned short CMCOR_0;                  /* CMCOR_0      */
      union {                                  /* CMCSR_1      */
           unsigned short WORD;                /*  Word Access */
           struct {                            /*  Bit Access  */
                unsigned short :8;             /*              */
                unsigned short CMF:1;          /*   CMF        */
                unsigned short CMIE:1;         /*   CMIE       */
                unsigned short :4;             /*              */
                unsigned short CKS:2;          /*   CKS        */
                } BIT;                         /*              */
           } CMCSR_1;                          /*              */
      unsigned short CMCNT_1;                  /* CMCNT_1      */
```

```
      unsigned short CMCOR_1;                  /* CMCOR_1     */
};                                             /*             */
struct st_ad {                                 /* struct AD   */
      union {                                   /* ADDR0       */
           unsigned short WORD;                 /*  Word Access */
           struct {                             /*  Byte Access */
                unsigned char ADH;              /*   AD H       */
                unsigned char wk;               /*              */
                } BYTE;                         /*              */
           struct {                             /*  Bit Access  */
                unsigned short AD:10;           /*   AD         */
                unsigned short :6;              /*              */
                } BIT;                          /*              */
           } ADDR0;                             /*              */
      union {                                   /* ADDR1        */
           unsigned short WORD;                 /*  Word Access */
           struct {                             /*  Byte Access */
                unsigned char ADH;              /*   AD H       */
                unsigned char wk;               /*              */
                } BYTE;                         /*              */
           struct {                             /*  Bit Access  */
                unsigned short AD:10;           /*   AD         */
                unsigned short :6;              /*              */
                } BIT;                          /*              */
           } ADDR1;                             /*              */
      union {                                   /* ADDR2        */
           unsigned short WORD;                 /*  Word Access */
           struct {                             /*  Byte Access */
                unsigned char ADH;              /*   AD H       */
                unsigned char wk;               /*              */
                } BYTE;                         /*              */
           struct {                             /*  Bit Access  */
                unsigned short AD:10;           /*   AD         */
                unsigned short :6;              /*              */
                } BIT;                          /*              */
           } ADDR2;                             /*              */
      union {                                   /* ADDR3        */
           unsigned short WORD;                 /*  Word Access */
```

```
      struct {                          /*  Byte Access   */
            unsigned char ADH;          /*    AD H        */
            unsigned char wk;           /*                */
            } BYTE;                     /*                */
      struct {                          /*  Bit Access    */
            unsigned short AD:10;       /*    AD          */
            unsigned short :6;          /*                */
            } BIT;                      /*                */
      } ADDR3;                          /*                */
union {                                 /* ADDR4          */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H        */
            unsigned char wk;           /*                */
            } BYTE;                     /*                */
      struct {                          /*  Bit Access  */
            unsigned short AD:10;       /*    AD          */
            unsigned short :6;          /*                */
            } BIT;                      /*                */
      } ADDR4;                          /*                */
union {                                 /* ADDR5          */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H        */
            unsigned char wk;           /*                */
            } BYTE;                     /*                */
      struct {                          /*  Bit Access  */
            unsigned short AD:10;       /*    AD          */
            unsigned short :6;          /*                */
            } BIT;                      /*                */
      } ADDR5;                          /*                */
union {                                 /* ADDR6          */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H        */
            unsigned char wk;           /*                */
            } BYTE;                     /*                */
      struct {                          /*  Bit Access  */
```

```
            unsigned short AD:10;           /*   AD          */
            unsigned short :6;              /*               */
            } BIT;                          /*               */
      } ADDR6;                              /*               */
union {                                     /* ADDR7         */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Byte Access */
            unsigned char ADH;              /*   AD H       */
            unsigned char wk;               /*               */
            } BYTE;                         /*               */
      struct {                              /*  Bit Access  */
            unsigned short AD:10;           /*   AD         */
            unsigned short :6;              /*               */
            } BIT;                          /*              */
      } ADDR7;                              /*               */
union {                                     /* ADDR8       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Byte Access */
            unsigned char ADH;              /*   AD H       */
            unsigned char wk;               /*              */
            } BYTE;                         /*              */
      struct {                              /*  Bit Access  */
            unsigned short AD:10;           /*   AD         */
            unsigned short :6;              /*              */
            } BIT;                          /*               */
      } ADDR8;                              /*               */
union {                                     /* ADDR9         */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Byte Access */
            unsigned char ADH;              /*   AD H       */
            unsigned char wk;               /*               */
            } BYTE;                         /*               */
      struct {                              /*  Bit Access  */
            unsigned short AD:10;           /*   AD         */
            unsigned short :6;              /*              */
            } BIT;                          /*              */
      } ADDR9;                              /*             */
union {                                     /* ADDR10      */
```

```
      unsigned short WORD;                /*  Word Access */
      struct {                            /*  Byte Access */
            unsigned char ADH;            /*    AD H       */
            unsigned char wk;             /*               */
            } BYTE;                       /*               */
      struct {                            /*  Bit Access  */
            unsigned short AD:10;         /*    AD         */
            unsigned short :6;            /*                */
            } BIT;                        /*                */
      } ADDR10;                           /*                */
union {                                   /* ADDR11        */
      unsigned short WORD;                /*  Word Access */
      struct {                            /*  Byte Access */
            unsigned char ADH;            /*    AD H       */
            unsigned char wk;             /*               */
            } BYTE;                       /*               */
      struct {                            /*  Bit Access  */
            unsigned short AD:10;         /*    AD         */
            unsigned short :6;            /*               */
            } BIT;                        /*               */
      } ADDR11;                           /*               */
union {                                   /* ADDR12        */
      unsigned short WORD;                /*  Word Access */
      struct {                            /*  Byte Access */
            unsigned char ADH;            /*    AD H      */
            unsigned char wk;             /*              */
            } BYTE;                       /*              */
      struct {                            /*  Bit Access  */
            unsigned short AD:10;         /*    AD         */
            unsigned short :6;            /*              */
            } BIT;                        /*             */
      } ADDR12;                           /*              */
union {                                   /* ADDR13       */
      unsigned short WORD;                /*  Word Access */
      struct {                            /*  Byte Access */
            unsigned char ADH;            /*    AD H      */
            unsigned char wk;             /*              */
            } BYTE;                       /*              */
```

```
      struct {                          /*  Bit Access  */
            unsigned short AD:10;       /*    AD        */
            unsigned short :6;          /*              */
            } BIT;                      /*              */
      } ADDR13;                         /*              */
union {                                 /* ADDR14       */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H      */
            unsigned char wk;           /*              */
            } BYTE;                     /*              */
      struct {                          /*  Bit Access  */
            unsigned short AD:10;       /*    AD        */
            unsigned short :6;          /*              */
            } BIT;                      /*              */
      } ADDR14;                         /*              */
union {                                 /* ADDR15       */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H      */
            unsigned char wk;           /*              */
            } BYTE;                     /*              */
      struct {                          /*  Bit Access   */
            unsigned short AD:10;       /*    AD         */
            unsigned short :6;          /*               */
            } BIT;                      /*               */
      } ADDR15;                         /*              */
union {                                 /* ADDR16       */
      unsigned short WORD;              /*  Word Access */
      struct {                          /*  Byte Access */
            unsigned char ADH;          /*    AD H      */
            unsigned char wk;           /*              */
            } BYTE;                     /*              */
      struct {                          /*  Bit Access  */
            unsigned short AD:10;       /*    AD        */
            unsigned short :6;          /*              */
            } BIT;                      /*              */
      } ADDR16;                         /*              */
```

```
union {                                 /* ADDR17        */
     unsigned short WORD;               /*  Word Access */
     struct {                           /*  Byte Access */
          unsigned char ADH;            /*    AD H       */
          unsigned char wk;             /*               */
          } BYTE;                       /*               */
     struct {                           /*  Bit Access  */
          unsigned short AD:10;         /*    AD         */
          unsigned short :6;            /*               */
          } BIT;                        /*               */
     } ADDR17;                          /*               */
union {                                 /* ADDR18       */
     unsigned short WORD;               /*  Word Access */
     struct {                           /*  Byte Access */
          unsigned char ADH;            /*    AD H       */
          unsigned char wk;             /*               */
          } BYTE;                       /*               */
     struct {                           /*  Bit Access  */
          unsigned short AD:10;         /*    AD         */
          unsigned short :6;            /*               */
          } BIT;                        /*              */
     } ADDR18;                          /*               */
union {                                 /* ADDR19       */
     unsigned short WORD;               /*  Word Access */
     struct {                           /*  Byte Access */
          unsigned char ADH;            /*    AD H      */
          unsigned char wk;             /*              */
          } BYTE;                       /*              */
     struct {                           /*  Bit Access  */
          unsigned short AD:10;         /*    AD         */
          unsigned short :6;            /*               */
          } BIT;                        /*               */
     } ADDR19;                          /*               */
unsigned char wk0[56];                  /*               */
union {                                 /* ADCSR_0       */
     unsigned char BYTE;                /*  Byte Access */
     struct {                           /*  Bit Access   */
          unsigned char ADF:1;          /*    ADF        */
```

```
            unsigned char ADIE:1;           /*   ADIE     */
            unsigned char ADM:2;            /*   ADM      */
            unsigned char :1;               /*            */
            unsigned char CH:3;             /*   CH       */
            } BIT;                          /*            */
      } ADCSR_0;                            /*            */
union {                                     /* ADCSR_1     */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char ADF:1;            /*   ADF       */
            unsigned char ADIE:1;           /*   ADIE      */
            unsigned char ADM:2;            /*   ADM       */
            unsigned char :1;               /*             */
            unsigned char CH:3;             /*   CH        */
            } BIT;                          /*             */
      } ADCSR_1;                            /*             */
union {                                     /* ADCSR_2     */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char ADF:1;            /*   ADF       */
            unsigned char ADIE:1;           /*   ADIE      */
            unsigned char ADM:2;            /*   ADM       */
            unsigned char :1;               /*             */
            unsigned char CH:3;             /*   CH        */
            } BIT;                          /*             */
      } ADCSR_2;                            /*             */
unsigned char wk1[5];                       /*             */
union {                                     /* ADCR_0      */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char TRGE:1;           /*   TRGE       */
            unsigned char CKS:2;            /*   CKS        */
            unsigned char ADST:1;           /*   ADST       */
            unsigned char ADCS:1;           /*   ADCS       */
            unsigned char :3;               /*              */
            } BIT;                          /*              */
      } ADCR_0;                             /*              */
union {                                     /* ADCR_1      */
```

```
            unsigned char BYTE;                  /*  Byte Access */
            struct {                             /*  Bit Access  */
                  unsigned char TRGE:1;          /*    TRGE      */
                  unsigned char CKS:2;           /*    CKS       */
                  unsigned char ADST:1;          /*    ADST      */
                  unsigned char ADCS:1;          /*    ADCS      */
                  unsigned char :3;              /*              */
                  } BIT;                         /*              */
            } ADCR_1;                            /*              */
      union {                                    /* ADCR_2       */
            unsigned char BYTE;                  /*  Byte Access */
            struct {                             /*  Bit Access  */
                  unsigned char TRGE:1;          /*    TRGE      */
                  unsigned char CKS:2;           /*    CKS       */
                  unsigned char ADST:1;          /*    ADST      */
                  unsigned char ADCS:1;          /*    ADCS      */
                  unsigned char :3;              /*              */
                  } BIT;                         /*              */
            } ADCR_2;                            /*              */
      unsigned char wk2[873];                    /*              */
      union {                                    /* ADTSR        */
            unsigned char BYTE;                  /*  Byte Access */
            struct {                             /*  Bit Access  */
                  unsigned char :2;              /*              */
                  unsigned char TRG2S:2;         /*    TRG2S     */
                  unsigned char TRG1S:2;         /*    TRG1S     */
                  unsigned char TRG0S:2;         /*    TRG0S     */
                  } BIT;                         /*              */
            } ADTSR;                             /*              */
};                                               /*              */
struct st_flash {                                /* struct FLASH */
      union {                                    /* FLMCR1       */
            unsigned char BYTE;                  /*  Byte Access */
            struct {                             /*  Bit Access  */
                  unsigned char FWE:1;           /*    FWE       */
                  unsigned char SWE:1;           /*    SWE       */
                  unsigned char ESU:1;           /*    ESU       */
                  unsigned char PSU:1;           /*    PSU       */
```

```
            unsigned char EV:1;             /*   EV       */
            unsigned char PV:1;             /*   PV       */
            unsigned char E:1;              /*   E        */
            unsigned char P:1;              /*   P        */
            } BIT;                          /*            */
      } FLMCR1;                             /*            */
union {                                     /* FLMCR2     */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char FLER:1;           /*   FLER      */
            unsigned char :7;               /*             */
            } BIT;                          /*             */
      } FLMCR2;                             /*             */
union {                                     /* EBR1        */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char EB:8;             /*   EB        */
            } BIT;                          /*             */
      } EBR1;                               /*             */
union {                                     /* EBR2        */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char :4;               /*             */
            unsigned char EB11:1;           /*   EB11       */
            unsigned char EB10:1;           /*   EB10       */
            unsigned char EB9:1;            /*   EB9        */
            unsigned char EB8:1;            /*   EB8        */
            } BIT;                          /*             */
      } EBR2;                               /*             */
unsigned char wk0[164];                     /*             */
union {                                     /* RAMER       */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short :12;             /*             */
            unsigned short RAMS:1;          /*   RAMS      */
            unsigned short RAM:3;           /*   RAM       */
            } BIT;                          /*             */
      } RAMER;                              /*             */
```

```
};                                                  /*             */
struct st_ubc {                                     /* struct UBC  */
      unsigned short UBARH;                          /* UBARH       */
      unsigned short UBARL;                          /* UBARL       */
      union {                                       /* UBAMRH      */
           unsigned short WORD;                      /*  Word Access */
           struct {                                 /*  Bit Access  */
                unsigned short UBM31:1;              /*   UBM31     */
                unsigned short UBM30:1;              /*   UBM30     */
                unsigned short UBM29:1;              /*   UBM29     */
                unsigned short UBM28:1;              /*   UBM28     */
                unsigned short UBM27:1;              /*   UBM27     */
                unsigned short UBM26:1;              /*   UBM26     */
                unsigned short UBM25:1;              /*   UBM25     */
                unsigned short UBM24:1;              /*   UBM24     */
                unsigned short UBM23:1;              /*   UBM23     */
                unsigned short UBM22:1;              /*   UBM22     */
                unsigned short UBM21:1;              /*   UBM21     */
                unsigned short UBM20:1;              /*   UBM20     */
                unsigned short UBM19:1;              /*   UBM19     */
                unsigned short UBM18:1;              /*   UBM18     */
                unsigned short UBM17:1;              /*   UBM17     */
                unsigned short UBM16:1;              /*   UBM16     */
                } BIT;                               /*             */
           } UBAMRH;                                 /*             */
      union {                                       /* UBAMRL      */
           unsigned short WORD;                      /*  Word Access */
           struct {                                 /*  Bit Access  */
                unsigned short UBM15:1;              /*   UBM15     */
                unsigned short UBM14:1;              /*   UBM14     */
                unsigned short UBM13:1;              /*   UBM13     */
                unsigned short UBM12:1;              /*   UBM12     */
                unsigned short UBM11:1;              /*   UBM11     */
                unsigned short UBM10:1;              /*   UBM10     */
                unsigned short UBM9:1;               /*   UBM9      */
                unsigned short UBM8:1;               /*   UBM8      */
                unsigned short UBM7:1;               /*   UBM7      */
                unsigned short UBM6:1;               /*   UBM6      */
```

```
                unsigned short UBM5:1;          /*   UBM5      */
                unsigned short UBM4:1;          /*   UBM4      */
                unsigned short UBM3:1;          /*   UBM3      */
                unsigned short UBM2:1;          /*   UBM2      */
                unsigned short UBM1:1;          /*   UBM1      */
                unsigned short UBM0:1;          /*   UBM0      */
                } BIT;                          /*             */
          } UBAMRL;                             /*             */
     union {                                   /* UBBR         */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short :8;              /*              */
                unsigned short CP:2;            /*   CP         */
                unsigned short ID:2;            /*   ID         */
                unsigned short RW:2;            /*   RW         */
                unsigned short SZ:2;            /*   SZ         */
                } BIT;                          /*              */
          } UBBR;                               /*              */
     union {                                   /* UBCR         */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short :13;             /*              */
                unsigned short CKS:2;           /*   CKS        */
                unsigned short UBID:1;          /*   UBID       */
                } BIT;                          /*               */
          } UBCR;                               /*              */
};                                              /*              */
struct st_wdt {                                 /* struct WDT   */
     union {                                    /* TCSR         */
          unsigned char BYTE;                   /*  Byte Access */
          struct {                              /*  Bit Access  */
                unsigned char OVF:1;            /*   OVF       */
                unsigned char WTIT:1;           /*   WT/IT     */
                unsigned char TME:1;            /*   TME       */
                unsigned char :2;               /*             */
                unsigned char CKS:3;            /*   CKS       */
                } BIT;                          /*             */
          } TCSR;                               /*             */
```

```
      unsigned char TCNT;                    /* TCNT          */
      union {                                /* RSTCSR        */
           unsigned char BYTE;               /*  Byte Access */
           struct {                          /*  Bit Access  */
                unsigned char WOVF:1;        /*    WOVF       */
                unsigned char RSTE:1;        /*    RSTE       */
                unsigned char RSTS:1;        /*    RSTS       */
                unsigned char :5;            /*               */
                } BIT;                       /*               */
           } RSTCSR;                         /*               */
};                                           /*               */
struct st_stby {                             /* struct STBY  */
      union {                                /* SBYCR        */
           unsigned char BYTE;               /*  Byte Access */
           struct {                          /*  Bit Access  */
                unsigned char SSBY:1;        /*    SSBY       */
                unsigned char HIZ:1;         /*    HIZ        */
                unsigned char :5;            /*               */
                unsigned char IRQEL:1;       /*    IRQEL      */
                } BIT;                       /*               */
           } SBYCR;                          /*               */
      unsigned char wk0[3];                  /*               */
      union {                                /* SYSCR        */
           unsigned char BYTE;               /*  Byte Access */
           struct {                          /*  Bit Access  */
                unsigned char :6;            /*               */
                unsigned char AUDSRST:1;     /*    AUDSRST    */
                unsigned char RAME:1;        /*    RAME       */
                } BIT;                       /*               */
           } SYSCR;                          /*               */
      unsigned char wk1[3];                  /*               */
      union {                                /* MSTCR1       */
           unsigned short WORD;              /*  Word Access */
           struct {                          /*  Bit Access  */
                unsigned short :4;           /*               */
                unsigned short MSTP27:1;     /*    MSTP27     */
                unsigned short MSTP26:1;     /*    MSTP26     */
                unsigned short MSTP25:1;     /*    MSTP25     */
```

```
                unsigned short MSTP24:1;          /*   MSTP24   */
                unsigned short :3;                /*            */
                unsigned short MSTP20:1;          /*   MSTP20   */
                unsigned short MSTP19:1;          /*   MSTP19   */
                unsigned short MSTP18:1;          /*   MSTP18   */
                unsigned short :2;                /*            */
                } BIT;                            /*            */
          } MSTCR1;                               /*            */
     union {                                      /* MSTCR2     */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short :1;                /*             */
                unsigned short MSTP14:1;          /*   MSTP14    */
                unsigned short MSTP13:1;          /*   MSTP13    */
                unsigned short MSTP12:1;          /*   MSTP12    */
                unsigned short :2;                /*             */
                unsigned short MSTP9:1;           /*   MSTP9     */
                unsigned short :2;                /*             */
                unsigned short MSTP6:1;           /*   MSTP6     */
                unsigned short MSTP5:1;           /*   MSTP5     */
                unsigned short MSTP4:1;           /*   MSTP4     */
                unsigned short MSTP3:1;           /*   MSTP3     */
                unsigned short MSTP2:1;           /*   MSTP2     */
                unsigned short :1;                /*             */
                unsigned short MSTP0:1;           /*   MSTP0     */
                } BIT;                            /*             */
          } MSTCR2;                               /*             */
};                                                /*             */
struct st_bsc {                                   /* struct BSC  */
     union {                                      /* BCR1        */
          unsigned short WORD;                    /*  Word Access */
          struct {                                /*  Bit Access  */
                unsigned short :1;                /*             */
                unsigned short MMTRWE:1;          /*   MMTRWE    */
                unsigned short MTURWE:1;          /*   MTURWE    */
                unsigned short :12;               /*             */
                unsigned short A0SZ:1;            /*   A0SZ      */
                } BIT;                            /*             */
```

```
            } BCR1;                                 /*              */
      union {                                       /* BCR2         */
            unsigned short WORD;                     /*  Word Access */
            struct {                                 /*  Bit Access  */
                  unsigned short :6;                 /*              */
                  unsigned short IW:2;               /*    IW        */
                  unsigned short :3;                 /*              */
                  unsigned short CW0:1;              /*    CW0       */
                  unsigned short :3;                  /*             */
                  unsigned short SW0:1;              /*    SW0       */
                  } BIT;                            /*              */
            } BCR2;                                 /*              */
      union {                                       /* WCR1         */
            unsigned short WORD;                     /*  Word Access */
            struct {                                 /*  Bit Access  */
                  unsigned short :12;                /*              */
                  unsigned short W:4;                /*    W         */
                  } BIT;                            /*              */
            } WCR1;                                 /*              */
};                                                  /*              */
struct st_dtc {                                     /* struct DTC   */
      union {                                       /* DTEA         */
            unsigned char BYTE;                      /*  Byte Access */
            struct {                                 /*  Bit Access  */
                  unsigned char TGI4A:1;             /*              */
                  unsigned char TGI4B:1;             /*              */
                  unsigned char TGI4C:1;             /*              */
                  unsigned char TGI4D:1;             /*              */
                  unsigned char TGI4V:1;             /*              */
                  unsigned char TGI3A:1;             /*              */
                  unsigned char TGI3B:1;             /*              */
                  unsigned char TGI3C:1;             /*              */
                  } BIT;                            /*              */
            } DTEA;                                 /*              */
      union {                                       /* DTEB         */
            unsigned char BYTE;                      /*  Byte Access */
            struct {                                 /*  Bit Access  */
                  unsigned char TGI3D:1;             /*              */
```

```
            unsigned char TGI2A:1;          /*            */
            unsigned char TGI2B:1;          /*            */
            unsigned char TGI1A:1;          /*            */
            unsigned char TGI1B:1;          /*            */
            unsigned char TGI0A:1;          /*            */
            unsigned char TGI0B:1;          /*            */
            unsigned char TGI0C:1;          /*            */
            } BIT;                          /*            */
      } DTEB;                               /*            */
union {                                     /* DTEC       */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char TGI0D:1;          /*             */
            unsigned char ADI0:1;           /*             */
            unsigned char IRQ0:1;           /*             */
            unsigned char IRQ1:1;           /*             */
            unsigned char IRQ2:1;           /*             */
            unsigned char IRQ3:1;           /*             */
            unsigned char b1:1;             /*             */
            unsigned char b0:1;             /*             */
            } BIT;                          /*             */
      } DTEC;                               /*             */
union {                                     /* DTED        */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char b7:1;             /*             */
            unsigned char b6:1;             /*             */
            unsigned char CMI0:1;           /*             */
            unsigned char CMI1:1;           /*             */
            unsigned char b3:1;             /*             */
            unsigned char b2:1;             /*             */
            unsigned char b1:1;             /*             */
            unsigned char b0:1;             /*             */
            } BIT;                          /*             */
      } DTED;                               /*             */
unsigned char wk0[2];                       /*             */
union {                                     /* DTCSR       */
      unsigned short WORD;                  /*  Word Access */
```

```
     struct {                              /*  Bit Access  */
          unsigned short :5;               /*              */
          unsigned short NMIF:1;           /*    NMIF      */
          unsigned short AE:1;             /*    AE        */
          unsigned short SWDTE:1;          /*    SWDTE     */
          unsigned short DTVEC7:1;         /*    DTVEC7    */
          unsigned short DTVEC6:1;         /*    DTVEC6    */
          unsigned short DTVEC5:1;         /*    DTVEC5    */
          unsigned short DTVEC4:1;         /*    DTVEC4    */
          unsigned short DTVEC3:1;         /*    DTVEC3    */
          unsigned short DTVEC2:1;         /*    DTVEC2    */
          unsigned short DTVEC1:1;         /*    DTVEC1    */
          unsigned short DTVEC0:1;         /*    DTVEC0    */
          } BIT;                           /*              */
     } DTCSR;                              /*              */
unsigned short DTBR;                       /* DTBR         */
unsigned char wk1[6];                      /*              */
union {                                    /* DTEE         */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char b7:1;              /*              */
          unsigned char b6:1;              /*              */
          unsigned char ADI1:1;            /*              */
          unsigned char ADI2:1;            /*              */
          unsigned char RXI_2:1;           /*              */
          unsigned char TXI_2:1;           /*              */
          unsigned char RXI_3:1;           /*              */
          unsigned char TXI_3:1;           /*              */
          } BIT;                           /*              */
     } DTEE;                               /*              */
union {                                    /* DTEF         */
     unsigned char BYTE;                   /*  Byte Access */
     struct {                              /*  Bit Access  */
          unsigned char RXI_4:1;           /*              */
          unsigned char TXI_4:1;           /*              */
          unsigned char TGN:1;             /*              */
          unsigned char TGM:1;             /*              */
          unsigned char b3:1;              /*              */
```

```
                unsigned char RM1:1;            /*             */
                unsigned char b1:1;             /*             */
                unsigned char b0:1;             /*             */
                } BIT;                          /*             */
          } DTEF;                               /*             */
};                                              /*             */
struct st_hudi {                                /* struct HUDI  */
     union {                                    /* SDIR         */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short TS:4;            /*    TS        */
                unsigned short :12;             /*              */
                } BIT;                          /*              */
          } SDIR;                               /*              */
     union {                                    /* SDSR         */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short :15;             /*              */
                unsigned short SDTRF:1;         /*   SDTRF      */
                } BIT;                          /*              */
          } SDSR;                               /*              */
     unsigned short SDDRH;                      /* SDDRH        */
     unsigned short SDDRL;                      /* SDDRL        */
};                                              /*              */
struct st_hcan2 {                               /* struct HCAN2 */
     union {                                    /* MCR          */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short :8;              /*              */
                unsigned short MCR7:1;          /*   MCR7       */
                unsigned short :1;              /*              */
                unsigned short MCR5:1;          /*   MCR5       */
                unsigned short :2;              /*              */
                unsigned short MCR2:1;          /*   MCR2       */
                unsigned short MCR1:1;          /*   MCR1       */
                unsigned short MCR0:1;          /*   MCR0       */
                } BIT;                          /*              */
          } MCR;                                /*              */
```

```
union {                                  /* GSR           */
     unsigned short WORD;                 /*  Word Access */
     struct {                             /*  Bit Access  */
          unsigned short :10;             /*              */
          unsigned short GSR5:1;          /*    GSR5      */
          unsigned short GSR4:1;          /*    GSR4      */
          unsigned short GSR3:1;          /*    GSR3      */
          unsigned short GSR2:1;          /*    GSR2      */
          unsigned short GSR1:1;          /*    GSR1      */
          unsigned short GSR0:1;          /*    GSR0      */
          } BIT;                         /*              */
     } GSR;                              /*              */
union {                                  /* HCAN2_BCR1   */
     unsigned short WORD;                 /*  Word Access */
     struct {                             /*  Bit Access  */
          unsigned short TSG1:4;          /*    TSG1      */
          unsigned short :1;              /*              */
          unsigned short TSG2:3;          /*    TSG2     */
          unsigned short :2;              /*              */
          unsigned short SJW:2;           /*    SJW       */
          unsigned short :3;              /*              */
          unsigned short BSP:1;           /*    BSP       */
          } BIT;                         /*              */
     } HCAN2_BCR1;                       /*              */
union {                                  /* HCAN2_BCR0   */
     unsigned short WORD;                /*  Word Access */
     struct {                            /*  Bit Access   */
          unsigned short :8;             /*              */
          unsigned short BRP:8;          /*    BRP        */
          } BIT;                         /*              */
     } HCAN2_BCR0;                       /*              */
union {                                  /* IRR          */
     unsigned short WORD;                /*  Word Access */
     struct {                            /*  Bit Access   */
          unsigned short IRR15:1;        /*    IRR15      */
          unsigned short IRR14:1;        /*    IRR14      */
          unsigned short IRR13:1;        /*    IRR13      */
          unsigned short IRR12:1;        /*    IRR12      */
```

```
            unsigned short :2;              /*            */
            unsigned short IRR9:1;          /*   IRR9     */
            unsigned short IRR8:1;          /*   IRR8     */
            unsigned short IRR7:1;          /*   IRR7     */
            unsigned short IRR6:1;          /*   IRR6     */
            unsigned short IRR5:1;          /*   IRR5     */
            unsigned short IRR4:1;          /*   IRR4     */
            unsigned short IRR3:1;          /*   IRR3     */
            unsigned short IRR2:1;          /*   IRR2     */
            unsigned short IRR1:1;          /*   IRR1     */
            unsigned short IRR0:1;          /*   IRR0     */
            } BIT;                          /*            */
      } IRR;                                /*            */
union {                                     /* IMR        */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short IMR15:1;         /*   IMR15     */
            unsigned short IMR14:1;         /*   IMR14     */
            unsigned short IMR13:1;         /*   IMR13     */
            unsigned short IMR12:1;         /*   IMR12     */
            unsigned short :2;              /*             */
            unsigned short IMR9:1;          /*   IMR9      */
            unsigned short IMR8:1;          /*   IMR8      */
            unsigned short IMR7:1;          /*   IMR7      */
            unsigned short IMR6:1;          /*   IMR6      */
            unsigned short IMR5:1;          /*   IMR5      */
            unsigned short IMR4:1;          /*   IMR4      */
            unsigned short IMR3:1;          /*   IMR3      */
            unsigned short IMR2:1;          /*   IMR2      */
            unsigned short IMR1:1;          /*   IMR1      */
            unsigned short :1;             /*             */
            } BIT;                         /*            */
      } IMR;                               /*            */
unsigned char TEC;                         /* TEC        */
unsigned char REC;                         /* REC        */
unsigned char wk0[18];                     /*            */
union {                                    /* TXPR1      */
      unsigned short WORD;                /*  Word Access  */
```

```
      struct {                              /*  Bit Access  */
            unsigned short TXPR31:1;        /*    TXPR31    */
            unsigned short TXPR30:1;        /*    TXPR30    */
            unsigned short TXPR29:1;        /*    TXPR29    */
            unsigned short TXPR28:1;        /*    TXPR28    */
            unsigned short TXPR27:1;        /*    TXPR27    */
            unsigned short TXPR26:1;        /*    TXPR26    */
            unsigned short TXPR25:1;        /*    TXPR25    */
            unsigned short TXPR24:1;        /*    TXPR24    */
            unsigned short TXPR23:1;        /*    TXPR23    */
            unsigned short TXPR22:1;        /*    TXPR22    */
            unsigned short TXPR21:1;        /*    TXPR21    */
            unsigned short TXPR20:1;        /*    TXPR20    */
            unsigned short TXPR19:1;        /*    TXPR19    */
            unsigned short TXPR18:1;        /*    TXPR18    */
            unsigned short TXPR17:1;        /*    TXPR17    */
            unsigned short TXPR16:1;        /*    TXPR16    */
            } BIT;                          /*              */
      } TXPR1;                              /*              */
union {                                     /* TXPR0        */
      unsigned short WORD;                  /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short TXPR15:1;        /*    TXPR15    */
            unsigned short TXPR14:1;        /*    TXPR14    */
            unsigned short TXPR13:1;        /*    TXPR13    */
            unsigned short TXPR12:1;        /*    TXPR12    */
            unsigned short TXPR11:1;        /*    TXPR11    */
            unsigned short TXPR10:1;        /*    TXPR10    */
            unsigned short TXPR9:1;         /*    TXPR9     */
            unsigned short TXPR8:1;         /*    TXPR8     */
            unsigned short TXPR7:1;         /*    TXPR7     */
            unsigned short TXPR6:1;         /*    TXPR6     */
            unsigned short TXPR5:1;         /*    TXPR5     */
            unsigned short TXPR4:1;         /*    TXPR4     */
            unsigned short TXPR3:1;         /*    TXPR3     */
            unsigned short TXPR2:1;         /*    TXPR2     */
            unsigned short TXPR1:1;         /*    TXPR1     */
            unsigned short :1;              /*              */
```

```
            } BIT;                              /*              */
      } TXPR0;                                  /*              */
unsigned char wk1[4];                           /*              */
union {                                         /* TXCR1        */
      unsigned short WORD;                      /*  Word Access */
      struct {                                  /*  Bit Access  */
            unsigned short TXCR31:1;            /*    TXCR31    */
            unsigned short TXCR30:1;            /*    TXCR30    */
            unsigned short TCR29:1;             /*    TCR29     */
            unsigned short TXCR28:1;            /*    TXCR28    */
            unsigned short TXCR27:1;            /*    TXCR27    */
            unsigned short TSCR26:1;            /*    TSCR26    */
            unsigned short TXCR25:1;            /*    TXCR25    */
            unsigned short TXCR24:1;            /*    TXCR24    */
            unsigned short TXCR23:1;            /*    TXCR23    */
            unsigned short TXCR22:1;            /*    TXCR22    */
            unsigned short TXCR21:1;            /*    TXCR21    */
            unsigned short TXCR20:1;            /*    TXCR20    */
            unsigned short TXCR19:1;            /*    TXCR19    */
            unsigned short TXCR18:1;            /*    TXCR18    */
            unsigned short TXCR17:1;            /*    TXCR17    */
            unsigned short TXCR16:1;            /*    TXCR16    */
            } BIT;                              /*              */
      } TXCR1;                                  /*              */
union {                                         /* TXCR0        */
      unsigned short WORD;                      /*  Word Access */
      struct {                                  /*  Bit Access  */
            unsigned short TXCR15:1;            /*    TXCR15    */
            unsigned short TXCR14:1;            /*    TXCR14    */
            unsigned short TCR13:1;             /*    TCR13     */
            unsigned short TXCR12:1;            /*    TXCR12    */
            unsigned short TXCR11:1;            /*    TXCR11    */
            unsigned short TSCR10:1;            /*    TSCR10    */
            unsigned short TXCR9:1;             /*    TXCR9     */
            unsigned short TXCR8:1;             /*    TXCR8     */
            unsigned short TXCR7:1;             /*    TXCR7     */
            unsigned short TXCR6:1;             /*    TXCR6     */
            unsigned short TXCR5:1;             *    TXCR5     */
```

```
            unsigned short TXCR4:1;          /*   TXCR4    */
            unsigned short TXCR3:1;          /*   TXCR3    */
            unsigned short TXCR2:1;          /*   TXCR2    */
            unsigned short TXCR1:1;          /*   TXCR1    */
            unsigned short :1;               /*            */
            } BIT;                           /*            */
      } TXCR0;                              /*            */
unsigned char wk2[4];                        /*            */
union {                                     /* TXACK1     */
      unsigned short WORD;                   /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short TXACK31:1;        /*   TXACK31   */
            unsigned short TXACK30:1;        /*   TXACK30   */
            unsigned short TXACK29:1;        /*   TXACK29   */
            unsigned short TXACK28:1;        /*   TXACK28   */
            unsigned short TXACK27:1;        /*   TXACK27   */
            unsigned short TXACK26:1;        /*   TXACK26   */
            unsigned short TXACK25:1;        /*   TXACK25   */
            unsigned short TXACK24:1;        /*   TXACK24   */
            unsigned short TXACK23:1;        /*   TXACK23   */
            unsigned short TXACK22:1;        /*   TXACK22   */
            unsigned short TXACK21:1;        /*   TXACK21   */
            unsigned short TXACK20:1;        /*   TXACK20   */
            unsigned short TXACK19:1;        /*   TXACK19   */
            unsigned short TXACK18:1;        /*   TXACK18   */
            unsigned short TXACK17:1;        /*   TXACK17   */
            unsigned short TXACK16:1;        /*   TXACK16   */
            } BIT;                           /*            */
      } TXACK1;                             /*            */
union {                                     /* TXACK0     */
      unsigned short WORD;                   /*  Word Access */
      struct {                              /*  Bit Access  */
            unsigned short TXACK15:1;        /*   TXACK15   */
            unsigned short TXACK14:1;        /*   TXACK14   */
            unsigned short TXACK13:1;        /*   TXACK13   */
            unsigned short TXACK12:1;        /*   TXACK12   */
            unsigned short TXACK11:1;        /*   TXACK11   */
            unsigned short TXACK10:1;        /*   TXACK10   */
```

```
            unsigned short TXACK9:1;          /*   TXACK9   */
            unsigned short TXACK8:1;          /*   TXACK8   */
            unsigned short TXACK7:1;          /*   TXACK7   */
            unsigned short TXACK6:1;          /*   TXACK6   */
            unsigned short TXACK5:1;          /*   TXACK5   */
            unsigned short TXACK4:1;          /*   TXACK4   */
            unsigned short TXACK3:1;          /*   TXACK3   */
            unsigned short TXACK2:1;          /*   TXACK2   */
            unsigned short TXACK1:1;          /*   TXACK1   */
            unsigned short :1;                /*            */
            } BIT;                            /*            */
      } TXACK0;                               /*            */
unsigned char wk3[4];                         /*            */
union {                                       /* ABACK1     */
      unsigned short WORD;                    /*  Word Access */
      struct {                                /*  Bit Access  */
            unsigned short ABACK31:1;         /*   ABACK31   */
            unsigned short ABACK30:1;         /*   ABACK30   */
            unsigned short ABACK29:1;         /*   ABACK29   */
            unsigned short ABACK28:1;         /*   ABACK28   */
            unsigned short ABACK27:1;         /*   ABACK27   */
            unsigned short ABACK26:1;         /*   ABACK26   */
            unsigned short ABACK25:1;         /*   ABACK25   */
            unsigned short ABACK24:1;         /*   ABACK24   */
            unsigned short ABACK23:1;         /*   ABACK23   */
            unsigned short ABACK22:1;         /*   ABACK22   */
            unsigned short ABACK21:1;         /*   ABACK21   */
            unsigned short ABACK20:1;         /*   ABACK20   */
            unsigned short ABACK19:1;         /*   ABACK19   */
            unsigned short ABACK18:1;         /*   ABACK18   */
            unsigned short ABACK17:1;         /*   ABACK17   */
            unsigned short ABACK16:1;         /*   ABACK16   */
            } BIT;                            /*             */
      } ABACK1;                               /*             */
union {                                       /* ABACK0      */
      unsigned short WORD;                    /*  Word Access */
      struct {                                /*  Bit Access  */
            unsigned short ABACK15:1;         /*   ABACK15   */
```

```
            unsigned short ABACK14:1;         /*   ABACK14   */
            unsigned short ABACK13:1;         /*   ABACK13   */
            unsigned short ABACK12:1;         /*   ABACK12   */
            unsigned short ABACK11:1;         /*   ABACK11   */
            unsigned short ABACK10:1;         /*   ABACK10   */
            unsigned short ABACK9:1;          /*   ABACK9    */
            unsigned short ABACK8:1;          /*   ABACK8    */
            unsigned short ABACK7:1;          /*   ABACK7    */
            unsigned short ABACK6:1;          /*   ABACK6    */
            unsigned short ABACK5:1;          /*   ABACK5    */
            unsigned short ABACK4:1;          /*   ABACK4    */
            unsigned short ABACK3:1;          /*   ABACK3    */
            unsigned short ABACK2:1;          /*   ABACK2    */
            unsigned short ABACK1:1;          /*   ABACK1    */
            unsigned short :1;                /*             */
            } BIT;                            /*             */
      } ABACK0;                               /*             */
unsigned char wk4[4];                         /*             */
union {                                       /* RXPR1       */
      unsigned short WORD;                    /*  Word Access */
      struct {                                /*  Bit Access  */
            unsigned short RXPR31:1;          /*   RXPR31     */
            unsigned short RXPR30:1;          /*   RXPR30     */
            unsigned short RXPR29:1;          /*   RXPR29     */
            unsigned short RXPR28:1;          /*   RXPR28     */
            unsigned short RXPR27:1;          /*   RXPR27     */
            unsigned short RXPR26:1;          /*   RXPR26     */
            unsigned short RXPR25:1;          /*   RXPR25     */
            unsigned short RXPR24:1;          /*   RXPR24     */
            unsigned short RXPR23:1;          /*   RXPR23     */
            unsigned short RXPR22:1;          /*   RXPR22     */
            unsigned short RXPR21:1;          /*   RXPR21     */
            unsigned short RXPR20:1;          /*   RXPR20     */
            unsigned short RXPR19:1;          /*   RXPR19     */
            unsigned short RXPR18:1;          /*   RXPR18     */
            unsigned short RXPR17:1;          /*   RXPR17     */
            unsigned short RXPR16:1;          /*   RXPR16     */
            } BIT;                            /*             */
```

```
     } RXPR1;                                /*             */
union {                                      /* RXPR0       */
     unsigned short WORD;                    /*  Word Access */
     struct {                                /*  Bit Access  */
          unsigned short RXPR15:1;           /*   RXPR15    */
          unsigned short RXPR14:1;           /*   RXPR14    */
          unsigned short RXPR13:1;           /*   RXPR13    */
          unsigned short RXPR12:1;           /*   RXPR12    */
          unsigned short RXPR11:1;           /*   RXPR11    */
          unsigned short RXPR10:1;           /*   RXPR10    */
          unsigned short RXPR9:1;            /*   RXPR9     */
          unsigned short RXPR8:1;            /*   RXPR8     */
          unsigned short RXPR7:1;            /*   RXPR7     */
          unsigned short RXPR6:1;            /*   RXPR6     */
          unsigned short RXPR5:1;            /*   RXPR5     */
          unsigned short RXPR4:1;            /*   RXPR4     */
          unsigned short RXPR3:1;            /*   RXPR3     */
          unsigned short RXPR2:1;            /*   RXPR2     */
          unsigned short RXPR1:1;            /*   RXPR1     */
          unsigned short RXPR0:1;            /*   RXPR0     */
          } BIT;                             /*             */
     } RXPR0;                                /*             */
unsigned char wk5[4];                        /*             */
union {                                      /* RFPR1       */
     unsigned short WORD;                    /*  Word Access */
     struct {                                /*  Bit Access  */
          unsigned short RFPR31:1;           /*   RFPR31     */
          unsigned short RFPR30:1;           /*   RFPR30     */
          unsigned short RFPR29:1;           /*   RFPR29     */
          unsigned short RFPR28:1;           /*   RFPR28     */
          unsigned short RFPR27:1;           /*   RFPR27     */
          unsigned short RFPR26:1;           /*   RFPR26     */
          unsigned short RFPR25:1;           /*   RFPR25     */
          unsigned short RFPR24:1;           /*   RFPR24     */
          unsigned short RFPR23:1;           /*   RFPR23     */
          unsigned short RFPR22:1;           /*   RFPR22     */
          unsigned short RFPR21:1;           /*   RFPR21     */
          unsigned short RFPR20:1;           /*   RFPR20     */
```

```
            unsigned short RFPR19:1;          /*   RFPR19   */
            unsigned short RFPR18:1;          /*   RFPR18   */
            unsigned short RFPR17:1;          /*   RFPR17   */
            unsigned short RFPR16:1;          /*   RFPR16   */
            } BIT;                            /*            */
      } RFPR1;                                /*            */
union {                                       /* RFPR0      */
      unsigned short WORD;                    /*  Word Access */
      struct {                                /*  Bit Access  */
            unsigned short RFPR15:1;          /*   RFPR15    */
            unsigned short RFPR14:1;          /*   RFPR14    */
            unsigned short RFPR13:1;          /*   RFPR13    */
            unsigned short RFPR12:1;          /*   RFPR12    */
            unsigned short RFPR11:1;          /*   RFPR11    */
            unsigned short RFPR10:1;          /*   RFPR10    */
            unsigned short RFPR9:1;           /*   RFPR9     */
            unsigned short RFPR8:1;           /*   RFPR8     */
            unsigned short RFPR7:1;           /*   RFPR7     */
            unsigned short RFPR6:1;           /*   RFPR6     */
            unsigned short RFPR5:1;           /*   RFPR5     */
            unsigned short RFPR4:1;           /*   RFPR4     */
            unsigned short RFPR3:1;           /*   RFPR3     */
            unsigned short RFPR2:1;           /*   RFPR2     */
            unsigned short RFPR1:1;           /*   RFPR1     */
            unsigned short RFPR0:1;           /*   RFPR0     */
            } BIT;                            /*            */
      } RFPR0;                                /*            */
unsigned char wk6[4];                         /*            */
union {                                       /* MBIMR1     */
      unsigned short WORD;                    /*  Word Access */
      struct {                                /*  Bit Access  */
            unsigned short MBIMR31:1;         /*   MBIMR31   */
            unsigned short MBIMR30:1;         /*   MBIMR30   */
            unsigned short MBIMR29:1;         /*   MBIMR29   */
            unsigned short MBIMR28:1;         /*   MBIMR28   */
            unsigned short MBIMR27:1;         /*   MBIMR27   */
            unsigned short MBIMR26:1;         /*   MBIMR26   */
            unsigned short MBIMR25:1;         /*   MBIMR25   */
```

```
            unsigned short MBIMR24:1;        /*   MBIMR24   */
            unsigned short MBIMR23:1;        /*   MBIMR23   */
            unsigned short MBIMR22:1;        /*   MBIMR22   */
            unsigned short MBIMR21:1;        /*   MBIMR21   */
            unsigned short MBIMR20:1;        /*   MBIMR20   */
            unsigned short MBIMR19:1;        /*   MBIMR19   */
            unsigned short MBIMR18:1;        /*   MBIMR18   */
            unsigned short MBIMR17:1;        /*   MBIMR17   */
            unsigned short MBIMR16:1;        /*   MBIMR16   */
            } BIT;                           /*             */
      } MBIMR1;                              /*             */
union {                                      /* MBIMR0      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short MBIMR15:1;        /*   MBIMR15   */
            unsigned short MBIMR14:1;        /*   MBIMR14   */
            unsigned short MBIMR13:1;        /*   MBIMR13   */
            unsigned short MBIMR12:1;        /*   MBIMR12   */
            unsigned short MBIMR11:1;        /*   MBIMR11   */
            unsigned short MBIMR10:1;        /*   MBIMR10   */
            unsigned short MBIMR9:1;         /*   MBIMR9    */
            unsigned short MBIMR8:1;         /*   MBIMR8    */
            unsigned short MBIMR7:1;         /*   MBIMR7    */
            unsigned short MBIMR6:1;         /*   MBIMR6    */
            unsigned short MBIMR5:1;         /*   MBIMR5    */
            unsigned short MBIMR4:1;         /*   MBIMR4    */
            unsigned short MBIMR3:1;         /*   MBIMR3    */
            unsigned short MBIMR2:1;         /*   MBIMR2    */
            unsigned short MBIMR1:1;         /*   MBIMR1    */
            unsigned short MBIMR0:1;         /*   MBIMR0    */
            } BIT;                           /*             */
      } MBIMR0;                              /*             */
unsigned char wk7[4];                        /*             */
union {                                      /* UMSR1       */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short UMSR31:1;         /*   UMSR31     */
            unsigned short UMSR30:1;         /*   UMSR30     */
```

```
            unsigned short UMSR29:1;          /*   UMSR29   */
            unsigned short UMSR28:1;          /*   UMSR28   */
            unsigned short UMSR27:1;          /*   UMSR27   */
            unsigned short UMSR26:1;          /*   UMSR26   */
            unsigned short UMSR25:1;          /*   UMSR25   */
            unsigned short UMSR24:1;          /*   UMSR24   */
            unsigned short UMSR23:1;          /*   UMSR23   */
            unsigned short UMSR22:1;          /*   UMSR22   */
            unsigned short UMSR21:1;          /*   UMSR21   */
            unsigned short UMSR20:1;          /*   UMSR20   */
            unsigned short UMSR19:1;          /*   UMSR19   */
            unsigned short UMSR18:1;          /*   UMSR18   */
            unsigned short UMSR17:1;          /*   UMSR17   */
            unsigned short UMSR16:1;          /*   UMSR16   */
            } BIT;                           /*            */
      } UMSR1;                               /*            */
union {                                      /* UMSR0      */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short UMSR15:1;          /*   UMSR15   */
            unsigned short UMSR14:1;          /*   UMSR14   */
            unsigned short UMSR13:1;          /*   UMSR13   */
            unsigned short UMSR12:1;          /*   UMSR12   */
            unsigned short UMSR11:1;          /*   UMSR11   */
            unsigned short UMSR10:1;          /*   UMSR10   */
            unsigned short UMSR9:1;           /*   UMSR9    */
            } BIT;                           /*            */
      } UMSR0;                               /*            */
unsigned char wk8[36];                       /*            */
unsigned short TCNTR;                        /* TCNTR      */
union {                                      /* TCR        */
      unsigned short WORD;                   /*  Word Access */
      struct {                               /*  Bit Access  */
            unsigned short TCR15:1;           /*   TCR15    */
            unsigned short TCR14:1;           /*   TCR14    */
            unsigned short TCR13:1;           /*   TCR13    */
            unsigned short TCR12:1;           /*   TCR12    */
            unsigned short TCR11:1;           /*   TCR11    */
```

```
                unsigned short TCR10:1;          /*   TCR10     */
                unsigned short TCR9:1;           /*   TCR9      */
                unsigned short TCR8:1;           /*   TCR8      */
                unsigned short TCR7:1;           /*   TCR7      */
                unsigned short :1;              /*             */
                unsigned short TPSC:6;           /*   TPSC     */
                } BIT;                          /*             */
          } TCR;                                /*             */
     union {                                    /* TSR         */
          unsigned short WORD;                  /*  Word Access */
          struct {                              /*  Bit Access  */
                unsigned short :13;             /*             */
                unsigned short TSR2:1;           /*   TSR2      */
                unsigned short TSR1:1;           /*   TSR1      */
                unsigned short TSR0:1;           /*   TSR0      */
                } BIT;                          /*             */
          } TSR;                                /*             */
     unsigned short TDCR;                       /* TDCR        */
     unsigned short LOSR;                       /* LOSR        */
     unsigned char wk9[2];                      /*             */
     unsigned short HCAN2_ICR0;                 /* HCAN2_ICR0  */
     unsigned short HCAN2_ICR1;                 /* HCAN2_ICR1  */
     unsigned short TCMR0;                      /* TCMR0       */
     unsigned short TCMR1;                      /* TCMR1       */
     unsigned char wk10[108];                   /*             */

struct st_mb {
          union {                               /* MB0         */
          unsigned char BYTE;                   /*  Byte Access */
          struct {                              /*  Bit Access  */
                unsigned char :1;               /*             */
                unsigned char STDID10:1;         /*   STDID10   */
                unsigned char STDID9:1;          /*   STDID9    */
                unsigned char STDID8:1;          /*   STDID8    */
                unsigned char STDID7:1;          /*   STDID7    */
                unsigned char STDID6:1;          /*   STDID6    */
                unsigned char STDID5:1;          /*   STDID5    */
                unsigned char STDID4:1;          /*   STDID4    */
```

```
            } BIT;                          /*             */
      } MB0;                                /*             */
union {                                     /* MB1         */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char STDID:4;          /*   STDID      */
            unsigned char RTR:1;            /*   RTR        */
            unsigned char IDE:1;            /*   IDE        */
            unsigned char EXTID17:1;        /*   EXTID17    */
            unsigned char EXTID16:1;        /*   EXTID16    */
            } BIT;                          /*              */
      } MB1;                                /*              */
union {                                     /* MB2         */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char EXTID15:1;        /*   EXTID15    */
            unsigned char EXTID14:1;        /*   EXTID14    */
            unsigned char EXTID13:1;        /*   EXTID13    */
            unsigned char EXTID12:1;        /*   EXTID12    */
            unsigned char EXTID11:1;        /*   EXTID11    */
            unsigned char EXTID10:1;        /*   EXTID10    */
            unsigned char EXTID9:1;         /*   EXTID9     */
            unsigned char EXTID8:1;         /*   EXTID8     */
            } BIT;                          /*              */
      } MB2;                                /*              */
union {                                     /* MB3         */
      unsigned char BYTE;                   /*  Byte Access */
      struct {                              /*  Bit Access  */
            unsigned char EXTID7:1;         /*   EXTID7     */
            unsigned char EXTID6:1;         /*   EXTID6     */
            unsigned char EXTID5:1;         /*   EXTID5     */
            unsigned char EXTID4:1;         /*   EXTID4     */
            unsigned char EXTID3:1;         /*   EXTID3     */
            unsigned char EXTID2:1;         /*   EXTID2     */
            unsigned char EXTID1:1;         /*   EXTID1     */
            unsigned char EXTID0:1;         /*   EXTID0     */
            } BIT;                          /*              */
      } MB3;                                /*              */
```

```
        union {                                 /* MB4          */
             unsigned char BYTE;                /*  Byte Access */
             struct {                           /*  Bit Access  */
                  unsigned char CCM:1;          /*    CCM       */
                  unsigned char TTE:1;          /*    TTE       */
                  unsigned char NMC:1;          /*    NMC       */
                  unsigned char ATX:1;          /*    ATX       */
                  unsigned char DART:1;         /*    DART      */
                  unsigned char MBC:3;          /*    MBC       */
                  } BIT;                        /*              */
             } MB4;                             /*              */
        union {                                 /* MB5          */
             unsigned char BYTE;                /*  Byte Access */
             struct {                           /*  Bit Access  */
                  unsigned char PTE:1;          /*    PTE       */
                  unsigned char TCT:1;          /*    TCT       */
                  unsigned char CBE:1;          /*    CBE       */
                  unsigned char :1;             /*              */
                  unsigned char DLC:4;          /*    DLC       */
                  } BIT;                        /*              */
             } MB5;                             /*              */
        unsigned char TIME_STAMP;               /* TIME_STAMP   */
        unsigned char wk11[1];                  /*              */
        unsigned char MSG_DATA[8];              /* MSG_DATA     */
        unsigned char LAFM0[2];                 /* LAFM0        */
        unsigned char LAFM1[2];                 /* LAFM1        */
        unsigned char wk12[12];                 /*              */
              }mb[32];
};                                              /*              */

#define P_SCI2 (*(volatile struct st_sci *)0xFFFF81C0)  /* SCI2 Address   */
#define P_SCI3 (*(volatile struct st_sci *)0xFFFF81D0)  /* SCI3 Address   */
#define P_SCI4 (*(volatile struct st_sci *)0xFFFF81E0)  /* SCI4 Address   */
#define P_MTU34 (*(volatile struct st_mtu34 *)0xFFFF8200)/* MTU34 Address */
#define P_MTU0 (*(volatile struct st_mtu0 *)0xFFFF8260)  /* MTU0 Address   */
#define P_MTU1 (*(volatile struct st_mtu1 *)0xFFFF8280)  /* MTU1 Address   */
#define P_MTU2 (*(volatile struct st_mtu2 *)0xFFFF82A0)  /* MTU2 Address   */
#define P_INTC (*(volatile struct st_intc *)0xFFFF8348)  /* INTC Address   */
```

```
#define P_PORTA (*(volatile struct st_porta *)0xFFFF8382)/* PORTA Address   */
#define P_PORTB (*(volatile struct st_portb *)0xFFFF8390)/* PORTB Address   */
#define P_PORTD (*(volatile struct st_portd *)0xFFFF83A2)/* PORTD Address   */
#define P_PORTE (*(volatile struct st_porte *)0xFFFF83B0)/* PORTE Address   */
#define P_PORTF (*(volatile struct st_portf *)0xFFFF83B2)/* PORTF Address   */
#define P_MTU (*(volatile struct st_mtu *)0xFFFF83C0)    /* MTU Address    */
#define P_MMT (*(volatile struct st_mmt *)0xFFFF83C4)    /* MMT Address    */
#define P_PORTG (*(volatile struct st_portg *)0xFFFF83CD)/* PORTG Address   */
#define P_CMT (*(volatile struct st_cmt *)0xFFFF83D0)    /* CMT Address     */
#define P_AD (*(volatile struct st_ad *)0xFFFF8420)      /* AD Address      */
#define P_FLASH (*(volatile struct st_flash *)0xFFFF8580)/* FLASH Address   */
#define P_UBC (*(volatile struct st_ubc *)0xFFFF8600)    /* UBC Address    */
#define P_WDT (*(volatile struct st_wdt *)0xFFFF8610)    /* WDT Address    */
#define P_STBY (*(volatile struct st_stby *)0xFFFF8614)  /* STBY Address   */
#define P_BSC (*(volatile struct st_bsc *)0xFFFF8620)    /* BSC Address    */
#define P_DTC (*(volatile struct st_dtc *)0xFFFF8700)    /* DTC Address    */
#define P_HUDI (*(volatile struct st_hudi *)0xFFFF8A50)  /* HUDI Address   */
#define P_HCAN2 (*(volatile struct st_hcan2 *)0xFFFFB000)/* HCAN2 Address */
```

SH7046シリーズ
アプリケーションノート　内蔵周辺機能　DTC編

発行年月　2003 年 11 月 12 日　Rev.1.00
発　　行　株式会社ルネサス テクノロジ 営業企画統括部
〒100-0004 東京都千代田区大手町 2-6-2
編　　集　株式会社ルネサス小平セミコン 技術ドキュメント部

株式会社ルネサス テクノロジ 営業企画統括部 〒100-0004 東京都千代田区大手町2-6-2 日本ビル

営業お問合せ窓口
株式会社ルネサス販売

http://www.renesas.com

| | | | |
|---|---|---|---|
| 本社 | 〒100-0004 | 千代田区大手町2-6-2 (日本ビル) | (03) 5201-5350 |
| 京浜支社 | 〒212-0058 | 川崎市幸区鹿島田890-12 (新川崎三井ビル) | (044) 549-1662 |
| 西東京支社 | 〒190-0023 | 立川市柴崎町2-2-23 (第二高島ビル2F) | (042) 524-8701 |
| 札幌支店 | 〒060-0002 | 札幌市中央区北二条西4-1 (札幌三井ビル5F) | (011) 210-8717 |
| 東北支社 | 〒980-0013 | 仙台市青葉区花京院1-1-20 (花京院スクエア13F) | (022) 221-1351 |
| いわき支店 | 〒970-8026 | いわき市平小太郎町4-9 (損保ジャパンいわき第二ビル3F) | (0246) 22-3222 |
| 茨城支社 | 〒312-0034 | ひたちなか市堀口832-2 (日立システムプラザ勝田1F) | (029) 271-9411 |
| 新潟支店 | 〒950-0087 | 新潟市東大通1-4-2 (新潟三井物産ビル3F) | (025) 241-4361 |
| 松本支社 | 〒390-0815 | 松本市深志1-2-11 (昭和ビル7F) | (0263) 33-6622 |
| 中部営業本部 | 〒460-0008 | 名古屋市中区栄3-13-20 (栄センタービル4F) | (052) 261-3000 |
| 浜松支店 | 〒430-7710 | 浜松市板屋町111-2（浜松アクトタワー10F） | (053) 451-2131 |
| 西部営業本部 | 〒541-0044 | 大阪市中央区伏見町4-1-1 (大阪明治生命館ランドアクシスタワー10F) | (06) 6233-9500 |
| 北陸支社 | 〒920-0031 | 金沢市広岡3-1-1 (金沢パークビル8F) | (076) 233-5980 |
| 中国支社 | 〒730-0036 | 広島市中区袋町5-25 (広島袋町ビルディング8F) | (082) 244-2570 |
| 松山支店 | 〒790-0003 | 松山市三番町4-4-6 (GEエジソンビル松山2号館3F) | (089) 933-9595 |
| 鳥取支店 | 〒680-0822 | 鳥取市今町2-251 (日本生命鳥取駅前ビル) | (0857) 21-1915 |
| 九州支社 | 〒812-0011 | 福岡市博多区博多駅前2-17-1 (ヒロカネビル本館5F) | (092) 481-7695 |
| 鹿児島支店 | 〒890-0053 | 鹿児島市中央町12-2 (明治生命西鹿児島ビル2F) | (099) 284-1748 |

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：カスタマサポートセンタ E-Mail: csc@renesas.com

Colophon 1.0

# SH7046 シリーズ
# アプリケーションノート　内蔵周辺機能　DTC 編

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

RJJ05B0307-0100H