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。

アプリケーション・ノート

RENESAS

78K/Ⅱシリーズ

8ビット・シングルチップ・マイクロコンピュータ

基礎編

μPD78214シリーズ
μPD78218Aシリーズ
μPD78224シリーズ
μPD78234シリーズ
μPD78244シリーズ

資料番号　IEA−607E（第6版）
発行年月　April 1997　N

アプリケーション・ノート

NEC

78K/Ⅱシリーズ

8ビット・シングルチップ・マイクロコンピュータ

基礎編

μPD78214シリーズ
μPD78218Aシリーズ
μPD78224シリーズ
μPD78234シリーズ
μPD78244シリーズ

本製品のうち，外国為替および外国貿易管理法の規定により戦略物資等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

本資料に掲載の応用回路および回路定数は，例示的に示したものであり，量産設計を対象とするものではありません。

○本資料の内容は，後日変更する場合があります。
○文書による当社の承諾なしに本資料の転載複製を禁じます。
○本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的所有権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。
○当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。
○当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「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の上ご使用願います。

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

特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

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，生命維持のための装置またはシステム等

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。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

○この製品は耐放射線設計をしておりません。

本版で改訂された主な箇所

箇　　所	内　　容
全　般	μPD78P244 に関する記述削除

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このアプリケーション・ノートは，78K/IIシリーズ製品の機能を理解し，78K/IIシリーズ製品を用いたアプリケーション・プログラムを設計するユーザのエンジニアを対象とします。

78K/IIシリーズ製品

- μPD78214シリーズ　：μPD78212，78213，78214，78P214，78212(A)，78213(A)，78214(A)，78P214(A)
- μPD78218Aシリーズ：μPD78217A，78218A，78P218A，78218A(A)
- μPD78224シリーズ　：μPD78220，78224，78P224
- μPD78234シリーズ　：μPD78233，78234，78237，78238，78P238 μPD78234(A)，78238(A)
- μPD78244シリーズ　：μPD78243，78244

目　的　このアプリケーション・ノートは，78K/IIシリーズ製品の基礎的な機能について，応用プログラム例を用いてユーザに理解していただくことを目的とします。
なお，掲載のプログラムおよびハードウエア構成は例示的に示したものであり，量産設計を対象とするものではありません。

構　成　このアプリケーション・ノートでは，次の内容について説明しています。

- ●概　説
- ●ソフトウエア編
- ●ハードウエアのプログラム例

なお，次の２つのアプリケーション・ノートも別に用意しています。

- ●応用編（IEA-700）
- ●浮動小数点演算プログラム編（IEA-686）

読 み 方　このアプリケーション・ノートでは，特に断りがない場合は，78K/IIシリーズ全製品を対象としています。

違いがある場合は，インデクスで区別ができるようになっています。

○インデクスの見方

 ：そのまま使用できます。

 ：使用できる部分とできない部分があります。

な　　し：このままでは使用できません。

なお，詳細は本文を参照してください。

●一通り使い方の例を見たい場合

→目次に従って読んでください。

●命令の使い方の例を見たい場合

→目次に従って読んでください。

●内蔵ハードウエアの使い方の例を見たい場合

→一通り見たい場合は順番に，特定のものは目次または付録Bを利用してください。

●特定の製品の応用例を見たい場合

→付録Bを利用してください。

●応用方法が分かっていて例を見たい場合

→目次と付録Bを併用してください。

品質水準　μPD78212(A)，78213(A)，78214(A)，78P214(A)はμPD78212，78213，78214，78P214の，μPD78218A(A)はμPD78218Aの『特別』品質水準品です。またμPD78234(A)，78238(A)はμPD78234，78238の『特別』品質水準品です。

このアプリケーション・ノート中の使用例は，一般電子機器用の『標準』品質水準品用に作成してあります。『特別』品質水準品を要求する用途にこのアプリケーション・ノート中の使用例を使用する場合は，実際に使用する各部品および回路について，その品質水準についてご検討のうえご使用ください。

品質水準とその応用分野の詳細については当社発行の資料「NEC 半導体デバイスの品質水準」(IEI-620)をご覧ください。

応用分野　○標準品

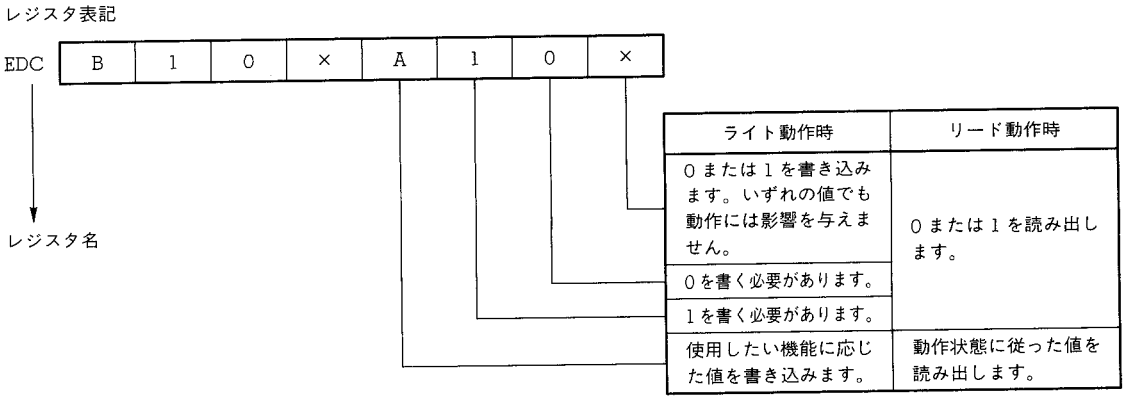
- プリンタ
- カメラ
- タイプライタ
- PPC
- 電子楽器
- エアコンなど

○特別品

- 自動車電装
- 燃料制御など

凡　例

データ表記の重み	：左側が上位桁，右側が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称の上に線）
注	：本文中に付けた注の説明
注意	：特に気をつけていただきたい内容
備考	：本文の補足説明
数の表記	：2進数…××××Bまたは××××
	：10進数…××××
	：16進数…××××H



本文中のレジスタ表記に『設定禁止』と書いてあるコードの組み合わせは，絶対に書き込まないでください。

まぎらわしい文字　：0（ゼロ），O（オー）

：1（イチ），l（エル），I（アイ）

関連資料

○共通資料一覧

資料名		資料番号
ユーザーズ・マニュアル　命令編		IEU-754
SBI　ユーザーズ・マニュアル		IEM-5040
アプリケーション・ノート	基礎編	このアプリケーション・ノート
	応用編	IEA-700
	浮動小数点演算プログラム編	IEA-686
セレクション・ガイド		IF-304
開発ツール　セレクション・ガイド		EF-231
インストラクション活用表		IEM-5101
インストラクション・セット		IEM-5102

○個別資料

μPD78214 シリーズ

品名 / 資料名	μPD78212	μPD78213	μPD78214	μPD78P214
パンフレット	IB-5036			
データ・シート	IC-8149	IC-7649		IC-7732
ユーザーズ・マニュアル　ハードウエア編	IEM-5119			
モード・レジスタ活用表	IEM-5100			

品名 / 資料名	μPD78212(A)	μPD78213(A)	μPD78214(A)	μPD78P214(A)
データ・シート	IC-8147	IC-8234		IC-8589
ユーザーズ・マニュアル　ハードウエア編	IEM-5119			
モード・レジスタ活用表	IEM-5100			

μPD78218A シリーズ

品名 / 資料名	μPD78217A	μPD78218A	μPD78P218A	μPD78218A(A)
パンフレット	IF-288			
データ・シート	IC-8132	IC-8131	IC-8133	IC-8685
ユーザーズ・マニュアル　ハードウエア編	IEU-755			
特殊機能レジスタ活用表	IEM-5532			

μPD78224 シリーズ

品名 / 資料名	μPD78220	μPD78224	μPD78P224
パンフレット	IB-5011		
データ・シート	IC-5457		IC-7757
ユーザーズ・マニュアル　ハードウエア編	IEM-5019		
特殊機能レジスタ活用表	IEM-999		

μPD78234 シリーズ

資料名＼品名	μPD78233	μPD78234	μPD78237	μPD78238	μPD78P238
パンフレット	IF-207				
データ・シート	IC-7902		IC-8348	IC-8028	IC-8030
ユーザーズ・マニュアル　ハードウエア編	IEU-718				
特殊機能レジスタ活用表	IEM-5515				

資料名＼品名	μPD78234(A)	μPD78238(A)
パンフレット	−	
データ・シート	IC-8146	IC-8727
ユーザーズ・マニュアル　ハードウエア編	IEU-718	
特殊機能レジスタ活用表	−	

μPD78244 シリーズ

資料名＼品名	μPD78243	μPD78244
パンフレット	IF-214	
データ・シート	IC-8355	IC-8070
ユーザーズ・マニュアル　ハードウエア編	IEU-747	
特殊機能レジスタ活用表	IEM-5528	

目　　次

図の目次（1/2）

図の目次（2/2）

表の目次（1/1）

第1章　概　　説

1.1　本書の見方

それぞれのプログラム例は次の項目について説明しています。

(1) ソフトウエア編

使用レジスタ，使用メモリ，入力条件，出力条件，処理手順，ステップ数，フロー・チャート，プログラム・リスト

(2) ハードウエアのプログラム例

動作概要，プログラム説明，モード・レジスタ設定例，入出力パラメータ，使用レジスタ，プログラム使用例，フロー・チャート，プログラム・リスト

1.1.1　ソフトウエア編の見方

(1) 使用レジスタ

プログラム中で使用しているレジスタです。すでに使用しているレジスタの内容を破壊したくない場合は，このプログラムを呼び出す前にPUSH命令などであらかじめ退避する必要があります。

(2) 使用メモリ

プログラム中で使用しているメモリ，あるいは入力条件となるメモリです。ただしワーク・エリアの場合は，このプログラムを使用した場合，内容が不定となります。

(3) 入力条件

プログラムを使用する場合に必要となる入力条件です。

(4) 出力条件

プログラムを実行した結果，出力されるものです。

(5) 処理手順

プログラムの流れを示しています。あわせて，フロー・チャートおよびプログラム・リストを参照してください。

(6) ステップ数

このアプリケーション・プログラムは，サブルーチン形式となっており，リロケータブル・アセンブラ　RA78K/2のリンカ　LK78K2によって，リンク（結合）して使用します。したがって，プログラムの終端はすべてRET命令となっています。ここで使用するステップ数とは，このRET命令を含んだものです。

(7) プログラム・リスト

プログラム・リストはすべてソース・リストで記載してあります。実際に配置されるアドレスはリンク条件により異なります。

1.1.2　ハードウエアのプログラム例の見方

(1) 動作概要

ハードウエアの動作概要を示しています。ブロック図などが記載されています。

(2) プログラム説明

プログラムの処理手順，モード・レジスタおよびメモリなどの設定例を示しています。

(3) モード・レジスタ設定例

モード・レジスタの設定例を示しています。

(4) 入出力パラメータ

ハードウエアを動作させるにあたり，与えなければならないパラメータおよび出力されるパラメータです。

多くのプログラムはリロケータブル・プログラムで掲載しています。したがって，例として，ユーザ・プログラムにおいてEQU疑似命令で定義したのち，パブリック宣言を行ってください。

(5) 使用レジスタ

プログラム中で使用しているレジスタです。すでに使用しているレジスタの内容を破壊したくない場合は，このプログラムを呼び出す前にPUSH命令などであらかじめ退避する必要があります。

(6) プログラム使用例

プログラムの使用例を示しています。

(7) プログラム・リスト

プログラム・リストは，すべてソース・リストで記載してあります。実際に配置されるアドレスはリンク条件により異なります。

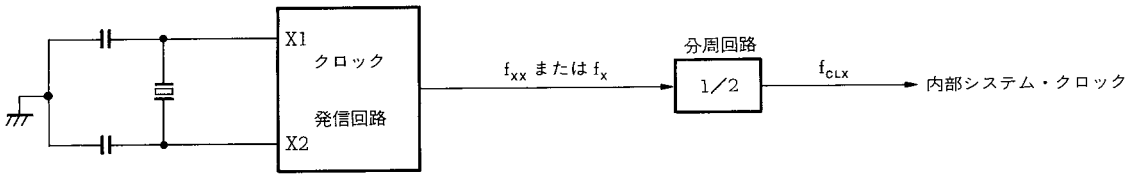
1.2 アプリケーション・プログラムの利用方法

基本的な利用方法は，各アプリケーション・プログラムの解説にしたがってください。

アプリケーション・プログラムによっては，他のアプリケーション・プログラムを呼び出す構成となっているものがありますので，その場合は解説にしたがって，リンカによりリンクしてください。また，ワーク・エリアを必要とする場合がありますので，解説にしたがって領域の確保を行い，パブリック宣言を行ってください。

ハードウエア編において，内蔵ハードウエアは，内部システム・クロックに動作が依存します。78K/IIシリーズ製品では，外部から供給されるクロック（f_{XX}，f_X）を2分周して，内部システム・クロック（f_{CLK}）としています。

図1-1　クロック発生回路のブロック図



備考　f_{XX}　：クリスタル/セラミック発振周波数

f_X　：外部クロック周波数

f_{CLK}　：内部システム・クロック周波数（$=\frac{1}{2}f_{XX}$ または $\frac{1}{2}f_X$）

以後，このアプリケーション・ノートでは内部システム・クロックを f_{CLK} と表現します。

1.3 78K/IIシリーズ製品の特徴

78K/IIシリーズは，78Kシリーズに属する8ビット・シングルチップ・マイクロコンピュータです。高性能8ビットCPU，ROM，RAMを搭載し，さらに各種の周辺ハードウエアをワンチップに集積しています。

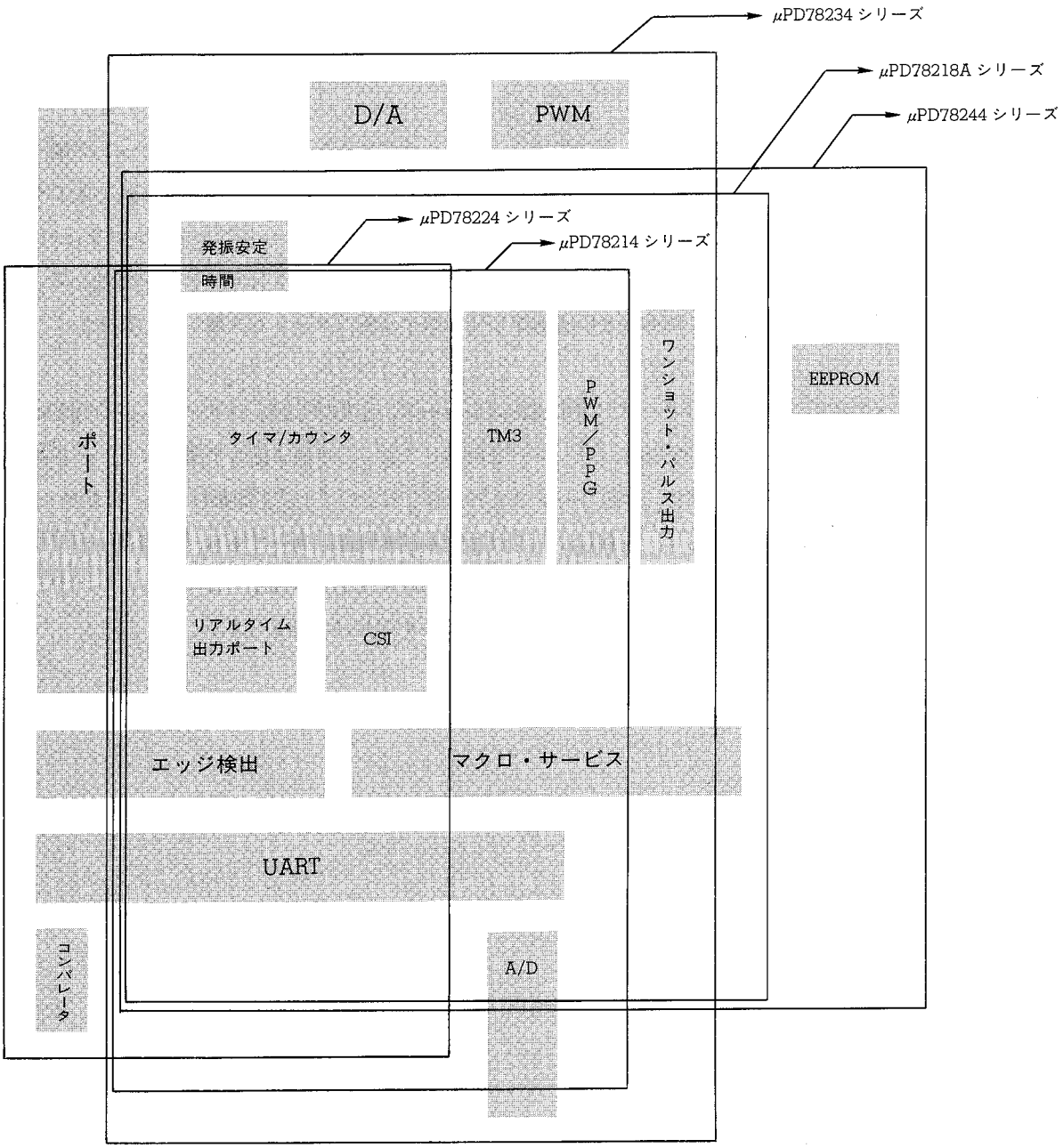
78K/IIシリーズは，16ビット演算命令，乗除算命令，ビット操作命令などの豊富な命令セットを備えています。また，大容量データに対応するために，データ用のアドレス空間を1Mバイトに拡張できます。

OA機器などへの応用に適した周辺ハードウエアとして，タイマ/カウンタ，シリアル・インタフェース，アナログ入力およびハードウエア・データ転送処理を備えた高機能割り込みコントローラを内蔵しています。また，ステッピング・モータの制御などに有効なリアルタイム出力ポートも内蔵しています。

78K/IIシリーズとして，5つのシリーズ（μPD78214シリーズ，μPD78218Aシリーズ，μPD78224シリーズ，μPD78234シリーズ，μPD78244シリーズ）を用意しており，用途に応じて最適なシリーズを選択することができます。各シリーズは，周辺ハードウエアが異なるだけでCPUは同じです。したがって，命令セットはすべて共通です。さらに，各周辺ハードウエアも基本機能は同一であり，各周辺ハードウエアを制御するプログラムもそのほとんどを共通に使用することができます（**図1-2**参照）。

また，シリーズ内の各製品については，内蔵するメモリのサイズだけが異なります。

図1-2　周辺機能の関連図



第2章　ソフトウエア編

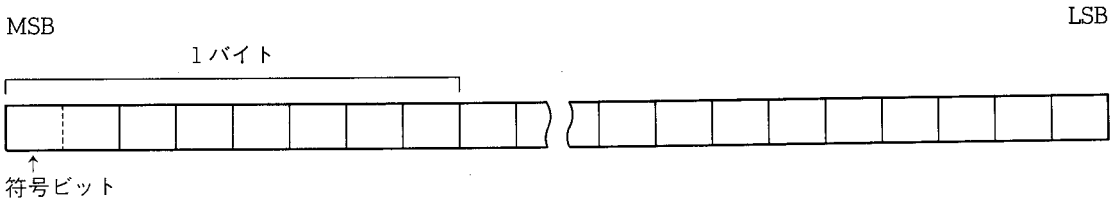
ソフトウエア編では，プログラムによってRAM上に，演算結果や数値データを配置します。したがって，次のプログラムを参考に，領域の確保を行ってください。

```
        PUBLIC  BMLCND,BMLIER,BRSLT     ; 2進乗算
        PUBLIC  DVISOR,DEND,DRMND       ; 2進除算
        PUBLIC  DMLCND,DMLIER,DRSLT     ; 10進乗算
        PUBLIC  DIVSOR,DIVIND,RMIND     ; 10進除算
        PUBLIC  ERROR                   ; エラー処理

        PUBLIC  SFLAG                   ; サイン・フラグ
        PUBLIC  CARRY                   ; キャリー・データ

        BSEG
SFLAG   DBIT                    ; サイン・フラグ

SOFT_D1 DSEG    SADDR
CARRY:  DS      1               ; キャリー・データ
                .
                .
;************************************************
;       データ領域
;************************************************
SOFT_D2 DSEG
BMLCND: DS      4               ; 2進乗算用領域
BMLIER: DS      4
BRSLT:  DS      8

DVISOR: DS      4               ; 2進除算用領域
DEND:   DS      4
DRMND:  DS      4

DMLCND: DS      4               ; 10進乗算用領域
DMLIER: DS      4
DRSLT:  DS      8

DIVSOR: DS      4               ; 10進除算用領域
DIVIND: DS      4
RMIND:  DS      4

        CSEG

ERROR:                          ; エラー処理を記述してください
```

2.1　2進演算

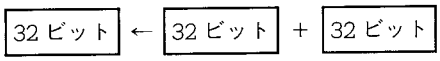
最上位ビットを符号ビットとし，残りのビットで数値を表現します。負の数の表現は2の補数表現を用います。

図2-1　2進数の表現

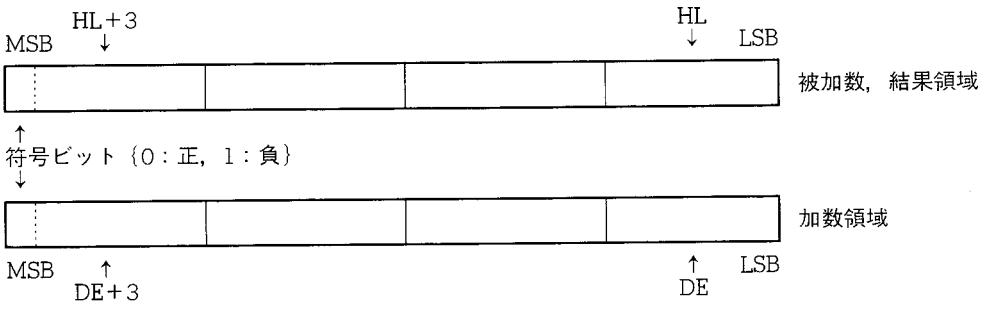


2進演算では，演算前も，演算後も，それぞれの数値はメモリ上に配置します。

2.1.1　2進加算



(1) 使用メモリ



(2) 使用レジスタ

A，C，DE，HL

(3) 入力条件

(1) で示すように，HL，DE レジスタの内容を次のように設定します。

- HL ← 被加数 32 ビットが格納してある領域の最下位アドレス。
- DE ← 加数 32 ビットが格納してある領域の最下位アドレス。

備考　プログラム中の BYTNUM の値をかえることにより，4 バイト（32 ビット）以外の加算が可能です。

(4) 出力条件

演算結果が（1）で示す領域（HL，HL＋1，HL＋2，HL＋3）に格納されます。

(5) 処理手順

(a)　バイト・カウンタ（C レジスタ）に 4 を設定する。

(b)　キャリー・フラグをあらかじめクリア（0）する。

(c)　加数アドレス(DE レジスタ)で示される 1 バイトを A レジスタに読み込み，加数アドレス(DE レジスタ）をインクリメントする。

(d)　被加数アドレス(HL レジスタ)で示される 1 バイトをキャリー・フラグとともに A レジスタに加える。

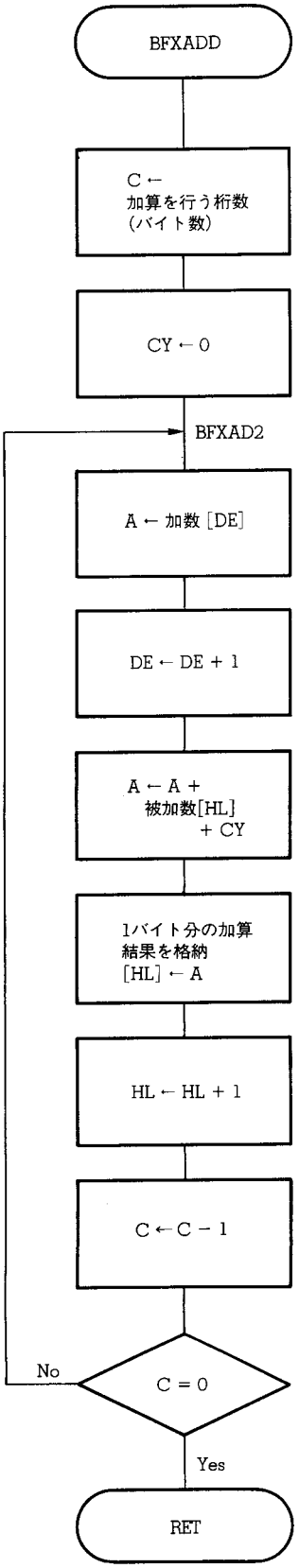
(e)　A レジスタの値を被加数アドレス（HL レジスタ）で示されるメモリに格納し，被加数アドレス（HL レジスタ）をインクリメントする。

(f)　バイト・カウンタ（C レジスタ）をデクリメントし，バイト・カウンタ（C レジスタ）が 0 になるまで (a) から (e) を繰り返す。

(6) ステップ数

7 ステップ

(7) フロー・チャート


BFXADD
C ←
加算を行う桁数
(バイト数)
CY ← 0
BFXAD2
A ← 加数 [DE]
DE ← DE + 1
A ← A +
被加数[HL]
+ CY
1バイト分の加算
結果を格納
[HL] ← A
HL ← HL + 1
C ← C − 1
No
C = 0
Yes
RET

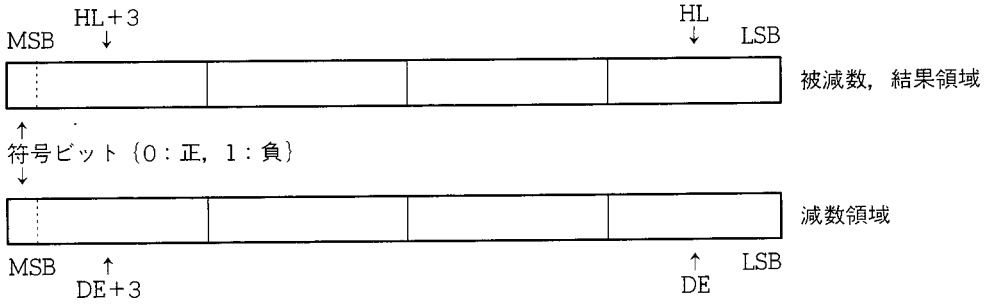
(8) プログラム・リスト

```
        NAME    BFXADR

;*****************************************************
;*      binary addition                              *
;*           32 bit <- 32 bit + 32 bit               *
;*       input condition                              *
;*            HL-register <- augnend top.address     *
;*            DE-register <- addend  top.address     *
;*       output condition                             *
;*            result <- (HL,HL+1,HL+2,HL+3)          *
;*****************************************************

        PUBLIC  BFXADD
;
BYTNUM  EQU     4
;
        CSEG
BFXADD:
        MOV     C,#BYTNUM
BFXAD1:
        CLR1    CY
BFXAD2:
        MOV     A,[DE+]
        ADDC    A,[HL]
        MOV     [HL+],A
        DBNZ    C,$BFXAD2

        RET
;
        END
```

2.1.2　2進減算

32ビット ← 32ビット − 32ビット

(1) 使用メモリ



(2) 使用レジスタ

A，C，DE，HL

(3) 入力条件

(1) で示すように，HL，DEレジスタの内容を次のように設定します。

- HL ← 被減数32ビットが格納してある領域の最下位アドレス。
- DE ← 減数32ビットが格納してある領域の最下位アドレス。

備考　プログラム中のBYTNUMの値をかえることにより，4バイト（32ビット）以外の減算が可能です。

(4) 出力条件

演算結果が（1）で示す領域（HL，HL+1，HL+2，HL+3）に格納されます。

(5) 処理手順

(a)　バイト・カウンタ（Cレジスタ）に4を設定する。

(b)　キャリー・フラグをあらかじめクリア（0）する。

(c)　被減数アドレス（HLレジスタ）で示される1バイトをAレジスタに読み込む。

(d)　減数アドレス(DEレジスタ)で示される1バイトをAレジスタからキャリー・フラグとともに引く。その後，減数アドレス（DEレジスタ）をインクリメントする。

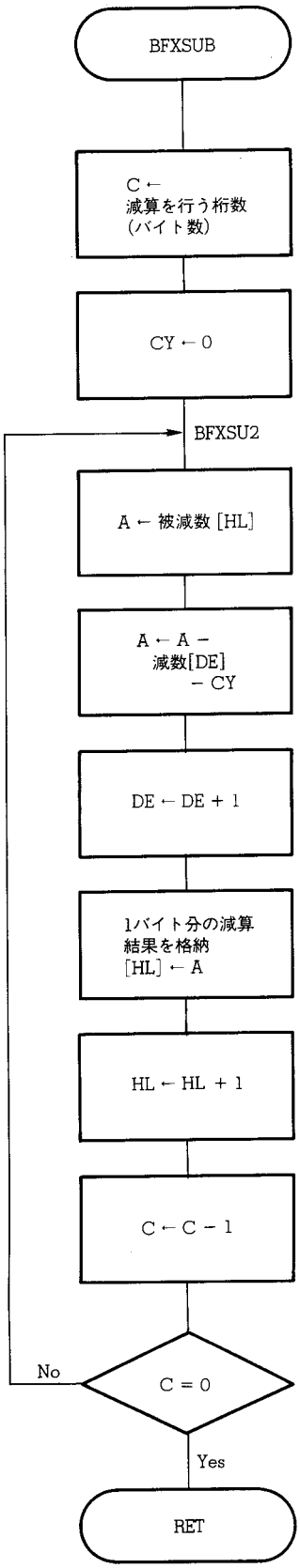
(e)　Aレジスタの値を被減数アドレス（HLレジスタ）で示されるメモリに格納し，被減数アドレス（HLレジスタ）をインクリメントする。

(f)　バイト・カウンタ（Cレジスタ）をデクリメントし，バイト・カウンタ（Cレジスタ）が0になるまで(a)から(e)を繰り返す。

(6) ステップ数

7ステップ

(7) フロー・チャート


BFXSUB
C ←
減算を行う桁数
(バイト数)
CY ← 0
BFXSU2
A ← 被減数 [HL]
A ← A −
減数[DE]
− CY
DE ← DE + 1
1バイト分の減算
結果を格納
[HL] ← A
HL ← HL + 1
C ← C − 1
No
C = 0
Yes
RET

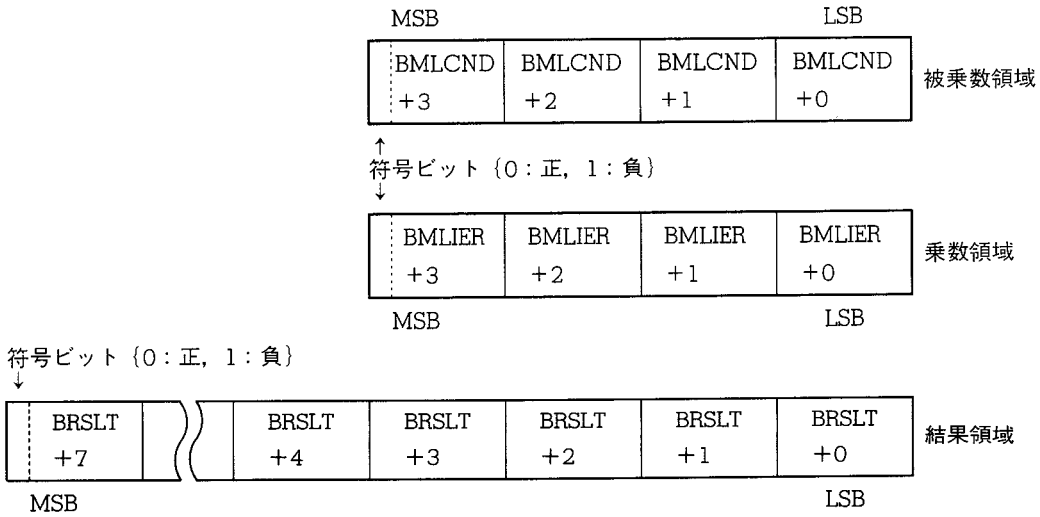
(8) プログラム・リスト

```
        NAME    BFXSBR

;*******************************************************
;*      binary subtraction                             *
;*           32 bit <- 32 bit - 32 bit                 *
;*       input condition                               *
;*            HL-register <- minus value top.address   *
;*            DE-register <- subtrahend top.address    *
;*       output condition                              *
;*            result <- (HL,HL+1,HL+2,HL+3)            *
;*******************************************************

        PUBLIC  BFXSUB
;
BYTNUM  EQU     4
;
        CSEG

BFXSUB:
        MOV     C,#BYTNUM
BFXSU1:
        CLR1    CY
BFXSU2:
        MOV     A,[HL]
        SUBC    A,[DE+]
        MOV     [HL+],A
        DBNZ    C,$BFXSU2

        RET
;
        END
```

2.1.3　2進乗算

64ビット ← 32ビット × 32ビット

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように被乗数32ビット，乗数32ビットをそれぞれ，次の領域に格納します。

- 被乗数：BMLCND，BMLCND＋1，BMLCND＋2，BMLCND＋3
- 乗数　：BMLIER，BMLIER＋1，BMLIER＋2，BMLIER＋3

(4) 出力条件

演算結果が（1）で示す結果領域（BRSLT，BRSLT＋1，…，BRSLT＋7）に格納されます。

(5) 処理手順

78K/IIの乗算命令を使用して処理を行うため，手計算における筆算に似た処理を行っています。次に処理手順を示します。

(a) 結果領域をクリア（0）する。

(b) 乗数，被乗数の絶対値をとる。乗数，被乗数が異符号なら符号フラグ（ユーザ・フラグ）をセット（1）し，同符号ならクリア（0）する。

(c) HLレジスタに結果領域の先頭アドレスを設定する。

(d) 乗数の桁ポインタ（DEレジスタ），ループ・カウンタ（B，Cレジスタ）にそれぞれ初期値として0を設定する。

(e) Xレジスタに（BMLCND＋Bレジスタ）で示される乗数1バイトを読み込む。

(f) Aレジスタに（BMLIER＋DEレジスタ）で示される被乗数1バイトを読み込み，Aレジスタ とXレジスタの乗算を行い，結果領域（HLレジスタ）に加算する。

(g) (f)の結果，桁あふれが生じた場合は桁上げ処理を行う。ただし，BRSLT＋7を越える領域には桁上げを行わない。

(h) ループ・カウンタ（Bレジスタ）をインクリメントする。

(i) 乗算結果のポインタ（HLレジスタ）の値をBRSLT＋B＋Cとする。

(j) 乗数4桁分の乗算が終了したか，Bレジスタの値をテストし，4でなければ(e)から(i)を繰り返す。

(k) ループ・カウンタ（Bレジスタ）を0に再設定し，被乗数ポインタ（DEレジスタ），ループ・カウンタ（Cレジスタ）をインクリメントする。

(l) 乗算結果ポインタ（HLレジスタ）の値をBRSLT＋Cとする。

(m) 被乗数4桁分の乗算が終了したか，Cレジスタの値をテストし，4でなければ(e)から(m)を繰り返す。

(n) 符号フラグが“1”ならば，乗算結果の2の補数を乗算結果とする。

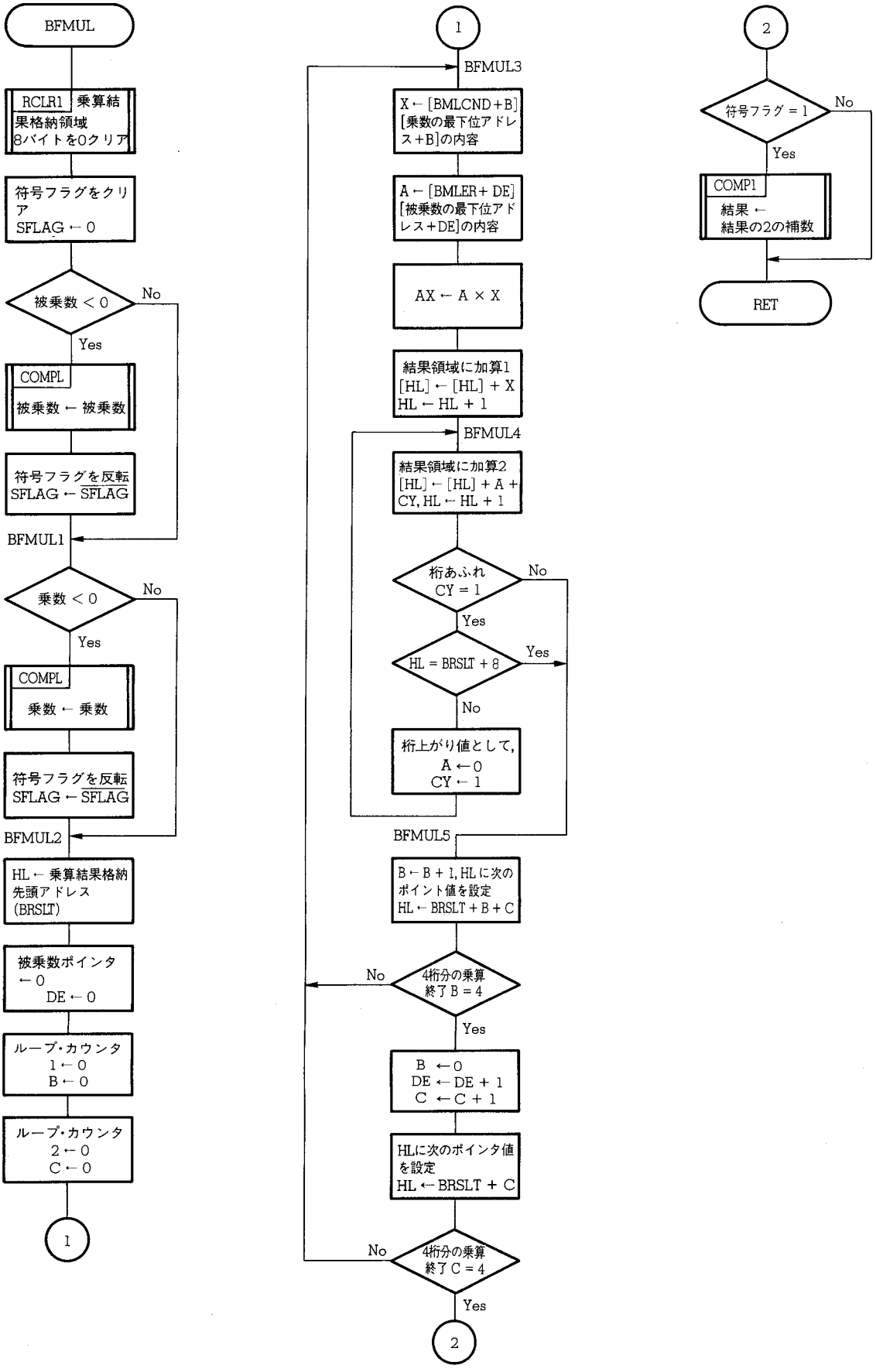
(6) ステップ数

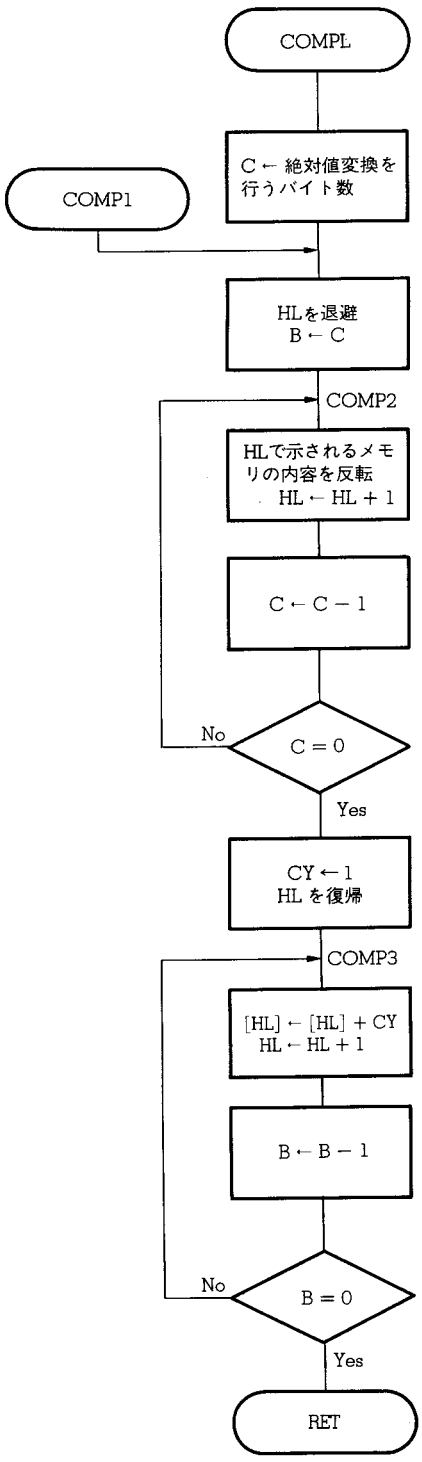
BFMUL　：59ステップ

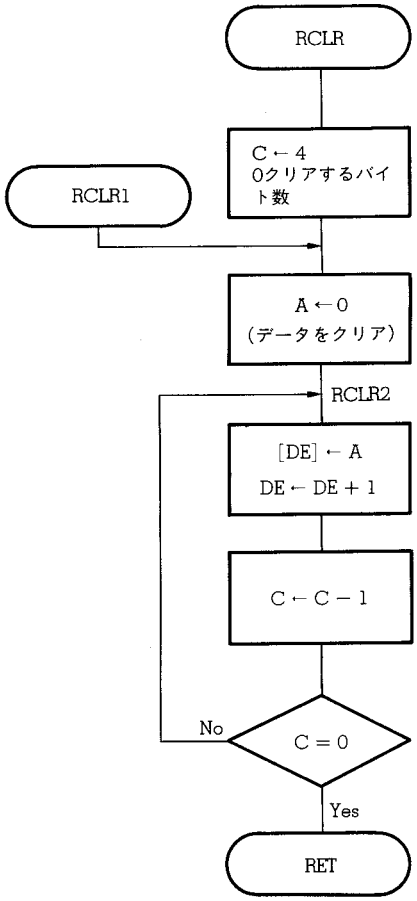
RCLR　：5ステップ

COMPL：14ステップ

(7) フロー・チャート


BFMUL
RCLR1 乗算結果格納領域 8バイトを0クリア
符号フラグをクリア SFLAG ← 0
被乗数 < 0
No
Yes
COMPL 被乗数 ← 被乗数
符号フラグを反転 SFLAG ← SFLAG
BFMUL1
乗数 < 0
No
Yes
COMPL 乗数 ← 乗数
符号フラグを反転 SFLAG ← SFLAG
BFMUL2
HL ← 乗算結果格納先頭アドレス (BRSLT)
被乗数ポインタ ← 0 DE ← 0
ループ・カウンタ 1 ← 0 B ← 0
ループ・カウンタ 2 ← 0 C ← 0
1
1
BFMUL3
X ← [BMLCND+B] [乗数の最下位アドレス+B]の内容
A ← [BMLER+ DE] [被乗数の最下位アドレス+DE]の内容
AX ← A × X
結果領域に加算1 [HL] ← [HL] + X HL ← HL + 1
BFMUL4
結果領域に加算2 [HL] ← [HL] + A + CY, HL ← HL + 1
桁あふれ CY = 1
No
Yes
HL = BRSLT + 8
Yes
No
桁上がり値として, A ← 0 CY ← 1
BFMUL5
B ← B + 1, HL に次のポイント値を設定 HL ← BRSLT + B + C
4桁分の乗算終了 B = 4
No
Yes
B ← 0 DE ← DE + 1 C ← C + 1
HLに次のポインタ値を設定 HL ← BRSLT + C
4桁分の乗算終了 C = 4
No
Yes
2
2
符号フラグ = 1
No
Yes
COMP1 結果 ← 結果の2の補数
RET


COMPL
COMP1
C ← 絶対値変換を行うバイト数
HLを退避
B ← C
COMP2
HLで示されるメモリの内容を反転
HL ← HL + 1
C ← C − 1
No
C = 0
Yes
CY ← 1
HL を復帰
COMP3
[HL] ← [HL] + CY
HL ← HL + 1
B ← B − 1
No
B = 0
Yes
RET


RCLR
RCLR1
C ← 4
0クリアするバイト数
A ← 0
(データをクリア)
RCLR2
[DE] ← A
DE ← DE + 1
C ← C − 1
No
C = 0
Yes
RET

(8) プログラム・リスト

```
        NAME    BFMULR

;*********************************************************
;*      binary multiplication                            *
;*         input condition                               *
;*              multiplicand <- (BMLCND+3,...,BMLCND)    *
;*              multiplier   <- (BMLIER+3,...,BMLIER)    *
;*         output condition                              *
;*              result <- (BRSLT+7,BRSLT+6...,BRSLT)     *
;*********************************************************

        PUBLIC  BFMUL
        EXTRN   BMLCND,BMLIER,BRSLT
        EXTRN   RCLR,RCLR1,COMPL,COMP1
        EXTBIT  SFLAG           ; sign-flag
;
BYTNUM  EQU     4               ; value length
;
        CSEG
BFMUL:
;
;       *** result area 0-clear ***
;
        MOV     C,#8            ; set area length
        MOVW    DE,#BRSLT       ; HL-reg. <- BRSLT
        CALL    !RCLR1          ; clear subroutine
;
;       *** compliment convert ***
;
        CLR1    SFLAG           ; sign-flag <- 0
        MOVW    HL,#BMLCND      ; HL-reg. <- BMLCND
        MOV     A,[HL+3]        ; check sign
        BF      A.7,$BFMUL1     ; if data<0 goto BFMUL1
        CALL    !COMPL          ; complement subroutine
        NOT1    SFLAG           ; not sign-flag
BFMUL1:
        MOVW    HL,#BMLIER      ; HL-reg. <- BMLIER
        MOV     A,[HL+3]
        BF      A.7,$BFMUL2
        CALL    !COMPL          ; complement subroutine
        NOT1    SFLAG           ; not sign-flag
;
;       *** set multiplication counter ***
;
BFMUL2:
        MOVW    HL,#BRSLT       ; result top address
        MOVW    DE,#0           ; sub pointer_1
        MOVW    BC,#0           ; loop counter_1,2
;
;       *** binary multiplication process ***
;
BFMUL3:
        MOV     A,BMLCND[B]     ; read base value
        MOV     X,A
        MOV     A,BMLIER[DE]    ; read multiplier
        MULU    X
        XCH     A,X
        ADD     A,[HL]          ; write result
        MOV     [HL+],A
        MOV     A,X
```

```
BFMUL4:
        ADDC    A,[HL]
        MOV     [HL+],A
        BNC     $BFMUL5
        MOVW    AX,HL           ; check end of value
        CMPW    AX,#BRSLT+8
        BZ      $BFMUL5
        MOV     A,#0            ; multiplication carry
        SET1    CY
        BR      BFMUL4
BFMUL5:
        INC     B               ; increment loop counter_1
        MOV     X,B             ; next result pointer
        ADD     X,C
        MOV     A,#0
        ADDW    AX,#BRSLT
        MOVW    HL,AX
        MOV     A,#BYTNUM       ; check loop counter 1
        CMP     B,A
        BNZ     $BFMUL3

        MOV     B,#0
        INCW    DE              ; increment sub pointer
        INC     C               ; increment loop counter_2
        MOV     X,C             ; next result pointer
        MOV     A,#0
        ADDW    AX,#BRSLT
        MOVW    HL,AX
        MOV     A,#BYTNUM       ; check loop counter_2
        CMP     C,A
        BNZ     $BFMUL3

        BT      SFLAG,$BFMUL6   ; if sflag=1 complement convert
        RET
BFMUL6:
        MOV     C,#8
        MOVW    HL,#BRSLT
        CALL    !COMP1
        RET

        END
```

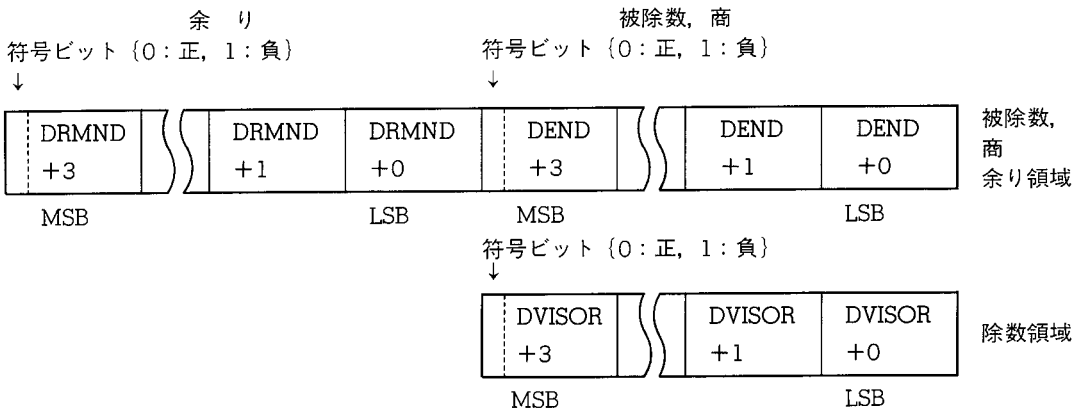
```
        NAME    CLR

;*******************************************************
;*      0-clear process                                *
;*          input condition                            *
;*              DE-register <- 0-clear start address *
;*                                                     *
;*******************************************************

        PUBLIC  RCLR,RCLR1,RCLR2
        PUBLIC  COMPL,COMP1
;
BYTNUM  EQU     4
;
        CSEG
RCLR:
        MOV     C,#BYTNUM       ; C-register <- 4
RCLR1:
        MOV     A,#0            ; Acc <- 0
RCLR2:
        MOV     [DE+],A
        DBNZ    C,$RCLR2
        RET
;
;*********************************************************
;*      complement convert subroutine                    *
;*              input  condition                         *
;*                HL-register <- complement top.address *
;*              output condition                         *
;*                (HL+3,HL+2,...,HL) <- convert data     *
;*                                                       *
;*********************************************************

        CSEG
COMPL:
        MOV     C,#BYTNUM
COMP1:
        PUSH    HL              ; save HL-register
        MOV     B,C
COMP2:
        MOV     A,#0FFH
        XOR     A,[HL]
        MOV     [HL+],A
        DBNZ    C,$COMP2

        SET1    CY
        POP     HL
COMP3:
        MOV     A,#0
        ADDC    A,[HL]
        MOV     [HL+],A
        DBNZ    B,$COMP3

        RET
;
        END
```

2.1.4　2進除算

商　[32ビット] ← [32ビット] ÷ [32ビット]

余り　[32ビット]

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように被除数32ビット，除数32ビットをそれぞれ，次の領域に格納します。

- 被除数：DEND，DEND＋1，DEND＋2，DEND＋3
- 除数　：DVISOR，DVISOR＋1，DVISOR＋2，DVISOR＋3

(4) 出力条件

商と余りが次の領域に格納されます。

- 商が（1）で示す商領域　　：DEND，DEND＋1，DEND＋2，DEND＋3
- 余りが（1）で示す余り領域：DRMND，DRMND＋1，DRMND＋2，DRMND＋3

(5) 処理手順

この除算プログラムでは，被除数（DEND，…，DEND＋3），余り（DRMND，…，DRMND＋3）を８バイトの連続した領域とします。被除数，余りの１桁（4ビット）を左シフトすることにより被除数の最上位桁が余りの最下位領域に転送され，商が最上位から１桁ずつ被除数の最下位領域に入ります。

また，商の各桁は，“余り－除数”の結果が負になるまで繰り返したときの回数とします。

次に処理手順を示します。

(a)　除数の０チェックを行う。０ならばエラー処理へ分岐する。

(b)　余り領域をクリア（0）する。

(c)　被除数，除数の絶対値をとる。商符号フラグをＸレジスタの第０ビット，余り符号フラグをＸレジスタの第１ビットとする。被除数，除数のどちらか一方だけが負ならば，商符号フラグをセット（1）する。

被除数が負ならば，余り符号フラグをセット（1）する。

(d)　被除数のバイト数として，Ｂレジスタに８を設定する。

(e)　８バイト続きの商，余り領域を４ビット左にシフトする。

(f)　“余り←余り－除数”を行う。

負ならば(h)へジャンプする。

(g)　商（DEND）をインクリメントする。(e)へジャンプする。

(h)　引きすぎたので復元するために，“余り←余り＋除数”を行う。

(i)　Ｂレジスタをデクリメントし，０になるまで(e)から(h)を繰り返す。

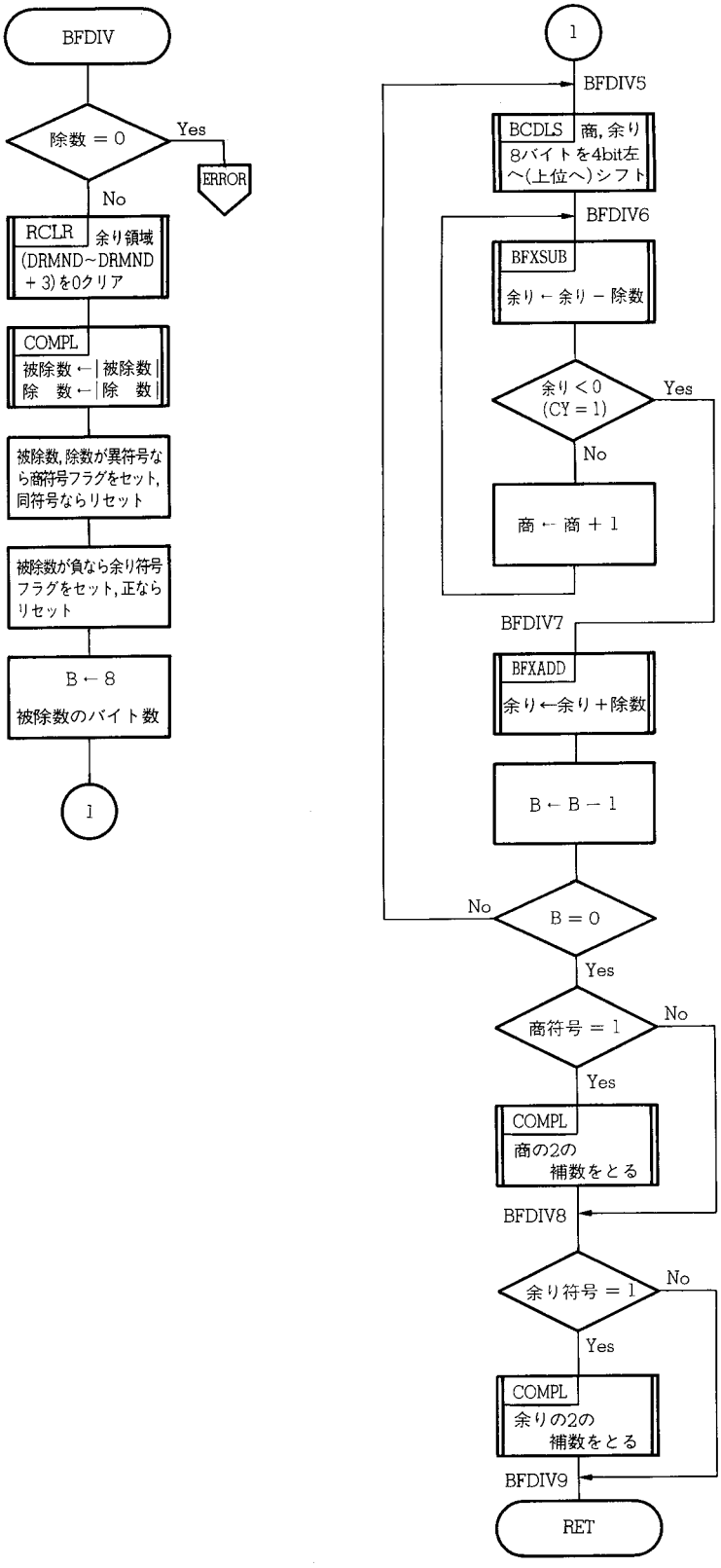
(j)　商符号フラグを調べ，セット（1）されていれば商の２の補数をとる。

余り符号フラグを調べ，セット（1）されていれば余りの２の補数をとる。

(6) ステップ数

48ステップ

(7) フロー・チャート


BFDIV
除数 = 0
Yes
ERROR
No
RCLR
余り領域
(DRMND~DRMND
+3)を0クリア
COMPL
被除数←|被除数|
除　数←|除　数|
被除数,除数が異符号なら商符号フラグをセット,同符号ならリセット
被除数が負なら余り符号フラグをセット,正ならリセット
B←8
被除数のバイト数
1
1
BFDIV5
BCDLS
商,余り
8バイトを4bit左へ(上位へ)シフト
BFDIV6
BFXSUB
余り←余り－除数
余り<0
(CY=1)
Yes
No
商←商＋1
BFDIV7
BFXADD
余り←余り＋除数
B←B－1
No
B=0
Yes
商符号 = 1
No
Yes
COMPL
商の2の
補数をとる
BFDIV8
余り符号 = 1
No
Yes
COMPL
余りの2の
補数をとる
BFDIV9
RET

(8) プログラム・リスト

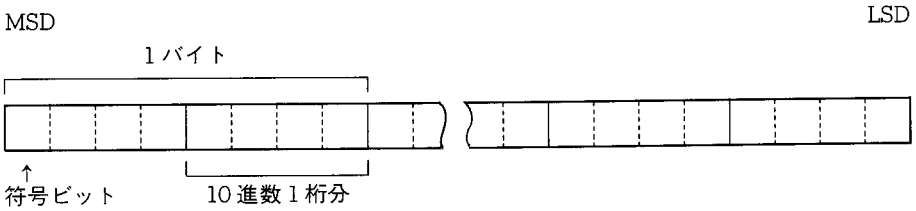
```
        NAME    BFDIVR

;*************************************************
;*      binary division                          *
;*           32 bit <- 32 bit / 32 bit           *
;*       input condition                         *
;*            dividend <- (DEND+3,...,DEND)      *
;*            divisor  <- (DVISOR+3,...,DVISOR)  *
;*       output condition                        *
;*            quotient  <- (DEND+3,...,DEND)     *
;*            remainder <- (DRMND+3,...,DRMND)   *
;*************************************************

        PUBLIC  BFDIV
        EXTRN   BFXADD,BFXSUB
        EXTRN   RCLR,COMPL,BCDLS,ERROR
        EXTRN   DEND,DVISOR,DRMND
;
SF_REM  EQU     X.0
SF_QUO  EQU     X.1
BYTNUM  EQU     4
;
        CSEG
BFDIV:
;
;       ****  check / divisor = 0 ?  ****
;
        MOV     C,#BYTNUM       ; C-register <- 4
        MOV     A,#0            ; Acc <- 0
        MOVW    HL,#DVISOR      ; HL <- DVISOR
BFDIV1:
        CMP     A,[HL+]
        BNZ     $BFDIV2         ; [HL] = 0 ?
        DBNZ    C,$BFDIV1
;
;       ****  divisor = 0  ****
;
        BR      ERROR           ; OVER FLOW
;
;       ****  quotient 0-clear  ****
;
BFDIV2:
        MOVW    DE,#DRMND       ; DE-register <- DRMND
        CALL    !RCLR
;
;       ****  complement convert  ****
;
        CLR1    SF_REM          ; clear remainder sign-flag
        CLR1    SF_QUO          ; clear quotient sign-flag
        MOVW    HL,#DEND        ; HL-register <- DEND
        MOV     A,[HL+3]
        BF      A.7,$BFDIV3
        CALL    !COMPL          ; complement subroutine
        SET1    SF_REM          ; set remainder sign-flag
        NOT1    SF_QUO          ; not quotient sign-flag
BFDIV3:
        MOVW    HL,#DVISOR      ; HL-register <- DVISOR
        MOV     A,[HL+3]
        BF      A.7,$BFDIV4
        CALL    !COMPL          ; complement subroutine
        NOT1    SF_QUO          ; not quotient sign-flag
```

```
;
;       ****  byte counter set  ****
;
BFDIV4:
        MOV     B,#8            ; B-register <- 8
;
;       ****  dividend,remainder 1-byte left shift  ****
;
BFDIV5:
        MOVW    HL,#DEND        ; HL <- DEND
        MOV     C,#8            ; C-register <- 8
        CALL    !BCDLS
;
;       ****  subtract divisor from dividend  ****
;
BFDIV6:
        MOVW    DE,#DVISOR      ; DE <- DVISOR
        MOVW    HL,#DRMND       ; HL <- DRMND
        CALL    !BFXSUB

        DECW    HL              ; decrement HL
        MOV     A,[HL]
        BT      A.7,$BFDIV7     ; if borrow

        MOV     A,#1            ; Acc <- 1
        MOVW    HL,#DEND
        ADD     A,[HL]          ; increment DEND
        MOV     [HL],A

        BR      BFDIV6
;
;       ****  if borrow divisor + dividend  ****
;
BFDIV7:
        MOVW    DE,#DVISOR      ; DE <- DVISOR
        MOVW    HL,#DRMND       ; HL <- DRMND
        CALL    !BFXADD
;
;       ****  check / division end ?  ****
;
        DBNZ    B,$BFDIV5

        BF      SF_REM,$BFDIV8
        MOVW    HL,#DRMND
        CALL    !COMPL
BFDIV8:
        BF      SF_QUO,$BFDIV9
        MOVW    HL,#DEND
        CALL    !COMPL
BFDIV9:
        RET
;
        END
```

2.2　10進演算

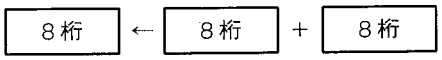
最上位ビットを符号ビットとし，残りのビットで数値を表現します。10進数はBCDコードで表現されます。

図2-2　10進数の表現

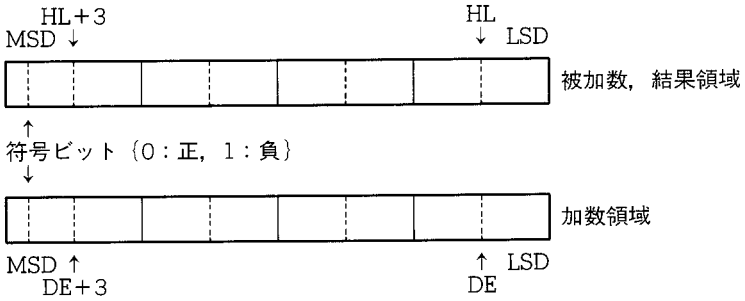


10進演算では，演算前も，演算後も，それぞれの数値はメモリ上に配置するものとします。

2.2.1　10進加算



(1) 使用メモリ



(2) 使用レジスタ

A，C，B，DE，HL

(3) 入力条件

(1) で示すように，HL，DEレジスタの内容を次のように設定します。

- HL ← 被加数8桁（4バイト）が格納してある領域の最下位アドレス。
- DE ← 加数8桁（4バイト）が格納してある領域の最下位アドレス。

(4) 出力条件

演算結果が（1）で示す結果領域に格納されます。ただし，HLレジスタの内容は破壊されます。また，オーバフローまたはアンダフローが発生した場合は，エラー処理へ分岐します。

注意　演算範囲は −79999999 から 79999999 です。

(5) 処理手順

この加算プログラムでは，加数，被加数が同符号ならば加算を行い，異符号ならば減算を行います。

次に処理手順を示します。

(a) Cレジスタに10進加算を行うバイト数を設定する。
　Bレジスタに符号を含まない10進加算を行うバイト数として，“C−1”を設定する。

(b) 加数，被加数が異符号ならば（o）の処理へ。

(c) キャリー・フラグ，符号フラグをクリア（0）する。

(d) 被加数アドレスで示す加数1バイトをAレジスタに読み込む。

(e) 加数アドレスで示す加数1バイトをキャリー・フラグとともにAレジスタに加算し，加数アドレスをインクリメントする。演算結果を10進補正し，被加数アドレスで示されるメモリに格納する。その後，被加数アドレスをインクリメントする。

(f) Bレジスタをデクリメントし，0になるまで(d)から(e)の処理を繰り返す。

(g) 被加数アドレスで示される被加数1バイトをAレジスタに読み込む。

(h) 加数アドレスで示される加数1バイトをキャリー・フラグとともにAレジスタに加算する。

(i) キャリー・フラグが“0”ならば(k)の処理へ。

(j) 符号フラグをセット（1）し，キャリー・フラグをクリア（0）する。

(k) Aレジスタを10進補正する。

(l) キャリー・フラグが“1”またはAレジスタの第7ビットが“1”ならばオーバフローなので，エラー処理へ。

(m) 符号フラグが“1”ならばAレジスタの第7ビットをセット（1）する。

(n) Aレジスタの内容を被加数アドレスで示されるメモリに格納し，演算を終了する。

(o) 減数を正にし，符号フラグをクリア（0）する。

(p) 被減数が負ならば被減数を正にし，符号フラグをセット（1）する。

(q) キャリー・フラグをクリア（0）する。

(r) 被減数アドレスで示される被減数1バイトをAレジスタに読み込む。

(s) Aレジスタから被減数アドレスで示される減数1バイトをキャリー・フラグとともに引き，減数アドレスをインクリメントする。演算結果を10進補正し，被減数アドレスで示されるメモリに格納する。その後，被減数アドレスをインクリメントする。

(t) Cレジスタをデクリメントし，0になるまで(r)から(s)の処理を繰り返す。

(u) キャリー・フラグが“0”ならば(w)の処理へ。

(v)　結果の 10 の補数をとり，符号フラグを反転する。

(w)　結果が“0”ならば演算終了とする。

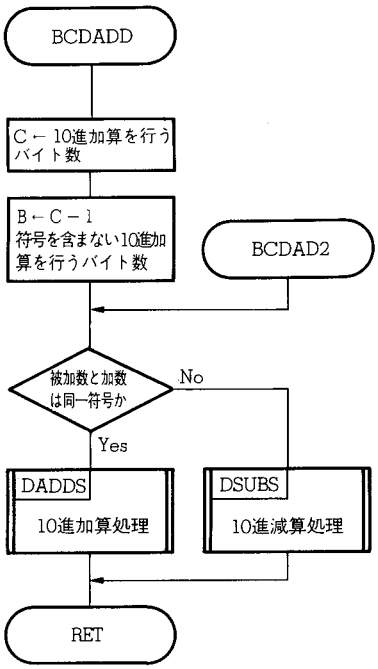
(x)　符号フラグが“1”ならば(y)の処理へ，“0”ならば演算終了とする。

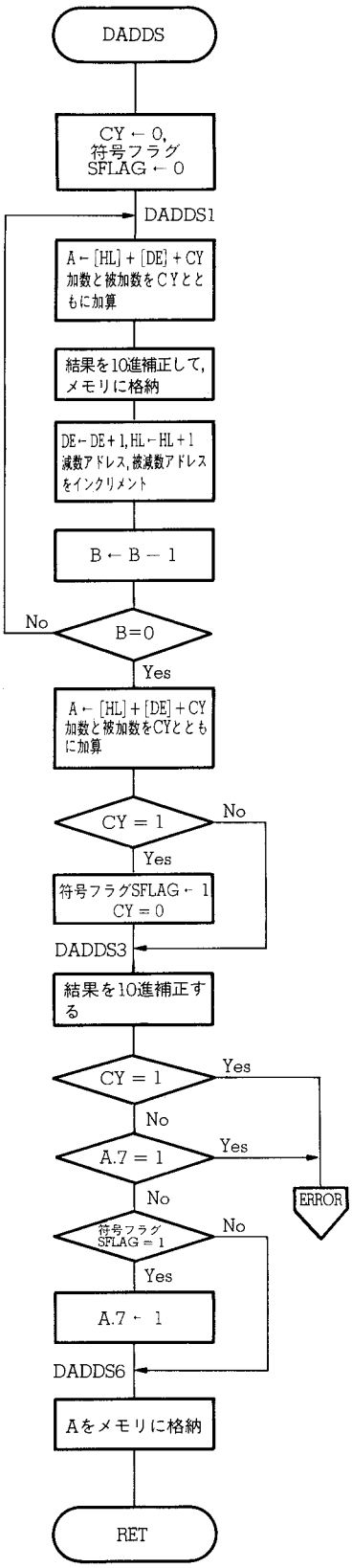
(y)　結果の符号ビットをセット（1）し，演算終了とする。

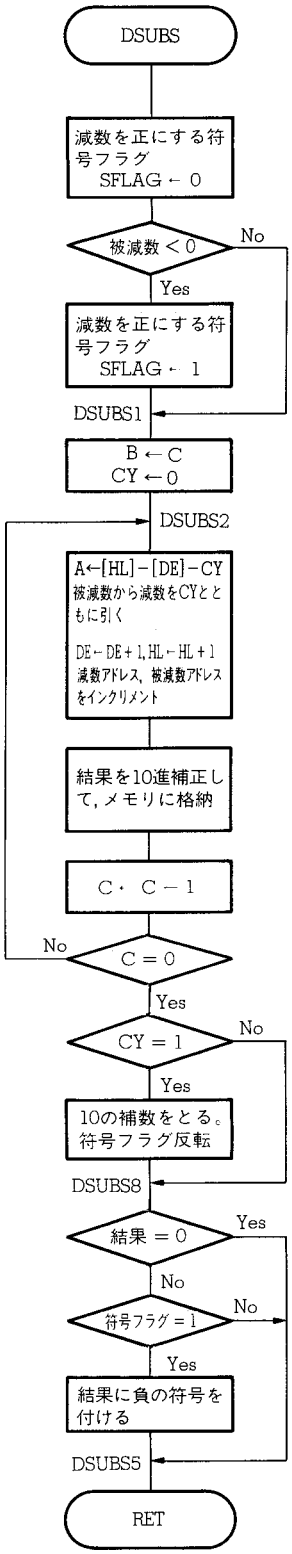
(6) ステップ数

83 ステップ

(7) フロー・チャート


BCDADD
C ← 10進加算を行うバイト数
B ← C − 1
符号を含まない10進加算を行うバイト数
BCDAD2
被加数と加数は同一符号か
No
Yes
DADDS
10進加算処理
DSUBS
10進減算処理
RET


DADDS
CY ← 0,
符号フラグ
SFLAG ← 0
DADDS1
A ← [HL] + [DE] + CY
加数と被加数をCYとともに加算
結果を10進補正して,
メモリに格納
DE ← DE + 1, HL ← HL + 1
減数アドレス, 被減数アドレス
をインクリメント
B ← B − 1
No
B=0
Yes
A ← [HL] + [DE] + CY
加数と被加数をCYとともに加算
CY = 1
No
Yes
符号フラグSFLAG ← 1
CY = 0
DADDS3
結果を10進補正する
CY = 1
Yes
No
A.7 = 1
Yes
No
ERROR
符号フラグ
SFLAG = 1
No
Yes
A.7 ← 1
DADDS6
Aをメモリに格納
RET


DSUBS
減数を正にする符号フラグ
SFLAG ← 0
被減数 < 0
No
Yes
減数を正にする符号フラグ
SFLAG ← 1
DSUBS1
B ← C
CY ← 0
DSUBS2
A ← [HL] − [DE] − CY
被減数から減数をCYとともに引く
DE ← DE + 1, HL ← HL + 1
減数アドレス, 被減数アドレス
をインクリメント
結果を10進補正して, メモリに格納
C ← C − 1
No
C = 0
Yes
CY = 1
No
Yes
10の補数をとる。
符号フラグ反転
DSUBS8
結果 = 0
Yes
No
符号フラグ = 1
No
Yes
結果に負の符号を付ける
DSUBS5
RET

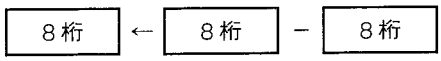
(8) プログラム・リスト

```
        NAME    BCDADR

;*********************************************************
;*      decimal addition                                 *
;*           8 digit <- 8 digit + 8 digit                *
;*       input condition                                 *
;*            HL-register <- augend area top.address     *
;*            DE-register <- addend  area top.address    *
;*       output condition                                *
;*            result <- (HL,HL+1,HL+2,HL+3)              *
;*********************************************************

        PUBLIC  BCDADD,BCDAD1,BCDAD2
        PUBLIC  DADDS
        PUBLIC  DSUBS
        EXTRN   ERROR
        EXTBIT  SFLAG           ; work flag for sign flag
;
BYTNUM  EQU     4
;
        CSEG
BCDADD:
        MOV     C,#BYTNUM       ; C-register <- 4
BCDAD1:
        MOV     B,C             ; B-register <- C-register - 1
        DEC     B
BCDAD2:
        MOV     A,[HL+BYTNUM-1]
        XOR     A,[DE+BYTNUM-1]

        BT      A.7,$BCDAD3
        CALL    !DADDS
        RET
BCDAD3:
        CALL    !DSUBS
        RET

;=====================================================
;       ***** decimal addition subroutine *****
;=====================================================

DADDS:
        CLR1    CY
        CLR1    SFLAG           ; clear sign-flag
DADDS1:
        MOV     A,[HL]
        ADDC    A,[DE+]
        ADJBA                   ; decimal adjust
        MOV     [HL+],A
        DBNZ    B,$DADDS1

        MOV     A,[HL]
        ADDC    A,[DE]
DADDS2:
        BNC     $DADDS3
        SET1    SFLAG           ; set sign-flag
        CLR1    CY
DADDS3:
        ADJBA                   ; decimal adjust
        BNC     $DADDS4
        BR      ERROR
```

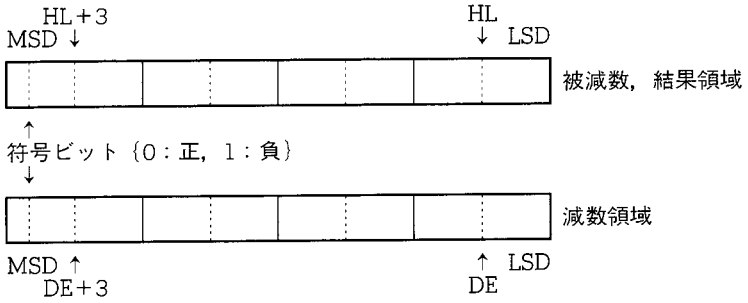
```
DADDS4:
        BF      A.7,$DADDS5
        BR      ERROR
DADDS5:
        BF      SFLAG,$DADDS6
        SET1    A.7
DADDS6:
        MOV     [HL],A
        RET

;=====================================================
;       ***** decimal subtraction subroutine *****
;=====================================================

DSUBS:
        PUSH    HL              ; save HL-register
        CLR1    SFLAG           ; clear sign-flag
        MOV     A,[DE+BYTNUM-1]
        CLR1    A.7
        MOV     [DE+BYTNUM-1],A
        MOV     A,[HL+BYTNUM-1]
        BF      A.7,$DSUBS1

        CLR1    A.7
        MOV     [HL+BYTNUM-1],A
        SET1    SFLAG           ; set sign-flag
DSUBS1:
        MOV     B,C             ; save C-register
        CLR1    CY
DSUBS2:
        MOV     A,[HL]
        SUBC    A,[DE+]
        ADJBS                   ; decimal adjust
        MOV     [HL+],A
        DBNZ    C,$DSUBS2

        BNC     $DSUBS5
        POP     HL              ; load HL-register
        PUSH    HL              ; save HL-register
        MOV     C,B             ; load C-register
DSUBS3:
        MOV     A,#99H          ; (HL) <- 9 - (HL)
        SUB     A,[HL]          ;     increment HL-register
        ADJBS                   ; decimal adjust
        MOV     [HL+],A
        DBNZ    C,$DSUBS3

        POP     HL              ; load HL-register
        PUSH    HL              ; save HL-register
        SET1    CY

        MOV     C,B             ; load C-register
DSUBS4:
        MOV     A,#0            ; Acc <- 0
        ADDC    A,[HL]
        ADJBA                   ; decimal adjust
        MOV     [HL+],A
        DBNZ    C,$DSUBS4
        NOT1    SFLAG
```

```
;
;       ****  check / result = 0  ****
;
DSUBS5:
        MOV     C,B             ; load C-register
        POP     HL              ; load HL-register
        PUSH    HL              ; save HL-register
        MOV     A,#0
DSUBS6:
        CMP     A,[HL+]
        BNZ     $DSUBS7
        DBNZ    C,$DSUBS6
        POP     HL              ; load HL-register
        RET
DSUBS7:
        POP     HL              ; load HL-register
        BF      SFLAG,$DSUBS8
        MOV     A,[HL+BYTNUM-1]
        SET1    A.7             ; sign set
        MOV     [HL+BYTNUM-1],A
DSUBS8:
        RET
;
        END
```

2.2.2　10進減算



(1) 使用メモリ



(2) 使用レジスタ

A，C，B，DE，HL

(3) 入力条件

(1) で示すように，HL，DEレジスタの内容を次のように設定します。

- HL ← 被減数8桁（4バイト）が格納してある領域の最下位アドレス。
- DE ← 減数8桁（4バイト）が格納してある領域の最下位アドレス。

(4) 出力条件

演算結果が（1）で示す結果領域に格納されます。

ただし，オーバフローまたはアンダフローが発生した場合は，エラー処理へ分岐します。

注意　演算範囲は −79999999 から 79999999 です。

(5) 処理手順

この減算プログラムでは，“被減数－減数”を“被減数＋（－減数)”という加算処理に変換して行っています。

次に処理手順を示します。

(a)　Cレジスタに10進減算を行うバイト数を設定する。

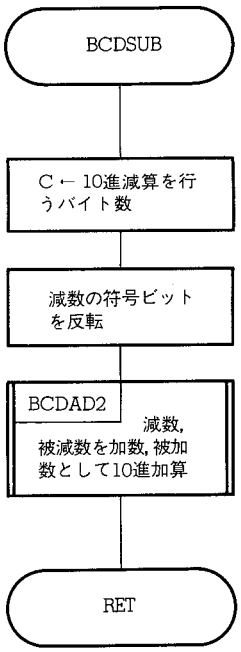
(b)　減数の符号ビットを反転する。

(c)　被減数，減数をそれぞれ被加数，加数として10進加算を行う。

(6) ステップ数

8ステップ

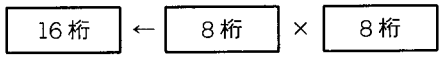
(7) フロー・チャート



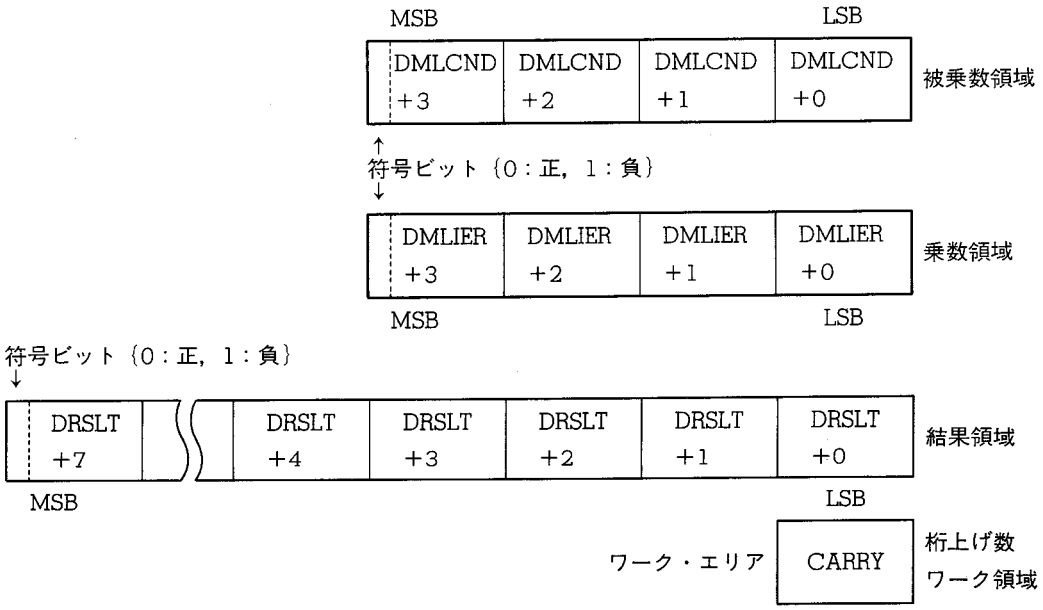
(8) プログラム・リスト

```
        NAME    BCDSUR

;*************************************************************
;*      decimal subtraction                                  *
;*           8 digit <- 8 digit - 8 digit                    *
;*       input condition                                     *
;*            HL-register <- minus value area top.address    *
;*            DE-register <- subtrahend area top.address     *
;*       output condition                                    *
;*            result <- (HL,HL+1,HL+2,HL+3)                  *
;*************************************************************

        PUBLIC  BYTNUM
        PUBLIC  BCDSUB
        EXTRN   BCDADD,BCDAD2
;
BYTNUM  EQU     4
;
        CSEG
BCDSUB:
        MOV     C,#BYTNUM       ; C-register <- 4
BCDSU1:
        MOV     B,C             ; B-register <- C-register - 1
        DEC     B

        MOV     A,[DE+BYTNUM-1]
        NOT1    A.7
        MOV     [DE+BYTNUM-1],A
        CALL    !BCDAD2
        RET
;
        END
```

2.2.3　10進乗算



(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように被乗数8桁，乗数8桁をそれぞれ，次の領域に格納します。

- 被乗数：DMLCND，DMLCND＋1，DMLCND＋2，DMLCND＋3
- 乗数　：DMLIER，DMLIER＋1，DMLIER＋2，DMLIER＋3

(4) 出力条件

演算結果が (1) で示す結果領域（DRSLT，DRSLT＋1，…，DRSLT＋7）に格納されます。

注意1．乗数と被乗数の有効範囲は −79999999 から 79999999 です。

2．演算範囲は −6399999840000001 から 6399999840000001 です。

(5) 処理手順

この乗算プログラムでは，乗数を1桁（4ビット）右シフトすることにより，最下位桁から1桁ずつ加数カウンタにロードします。これをカウンタとして，“結果←結果＋被乗数”を繰り返しています。

また，加算カウンタによる加算が終了後，1桁上位の乗数によって加算を行うため，結果領域を1桁（4ビット）右シフトしておきます。

次に処理手順を示します。

(a) 結果領域をクリア（0）する。

(b) 乗数，被乗数の絶対値をとる。乗数，被乗数が異符号ならば符号フラグをセット（1）し，同符号ならば符号フラグをクリア（0）する。

(c) 桁カウンタ（Bレジスタ）を8に設定する。

(d) 乗数を1桁右シフトすることにより，最下位桁を加数カウンタ（Cレジスタ）にロードする。

(e) 桁上げ数（CARRY）をクリアする。

(f) 加数カウンタが0なら(i)の処理へ。

(g) 10進加算“結果（上位8桁）←結果（上位8桁）＋被乗数”を行い，オーバフローが生じたら桁上げ数（CARRY）をインクリメントする。

(h) 加数カウンタをデクリメントし，0になるまで(g)の処理を繰り返す。

(i) 桁上げ数（CARRY）を含めて，結果領域を1桁（4ビット）右シフトする。
桁上げ数（CARRY）が結果領域の最上位桁に格納される。

(j) 桁カウンタをデクリメントし，0になるまで(d)から(i)の処理を繰り返す。

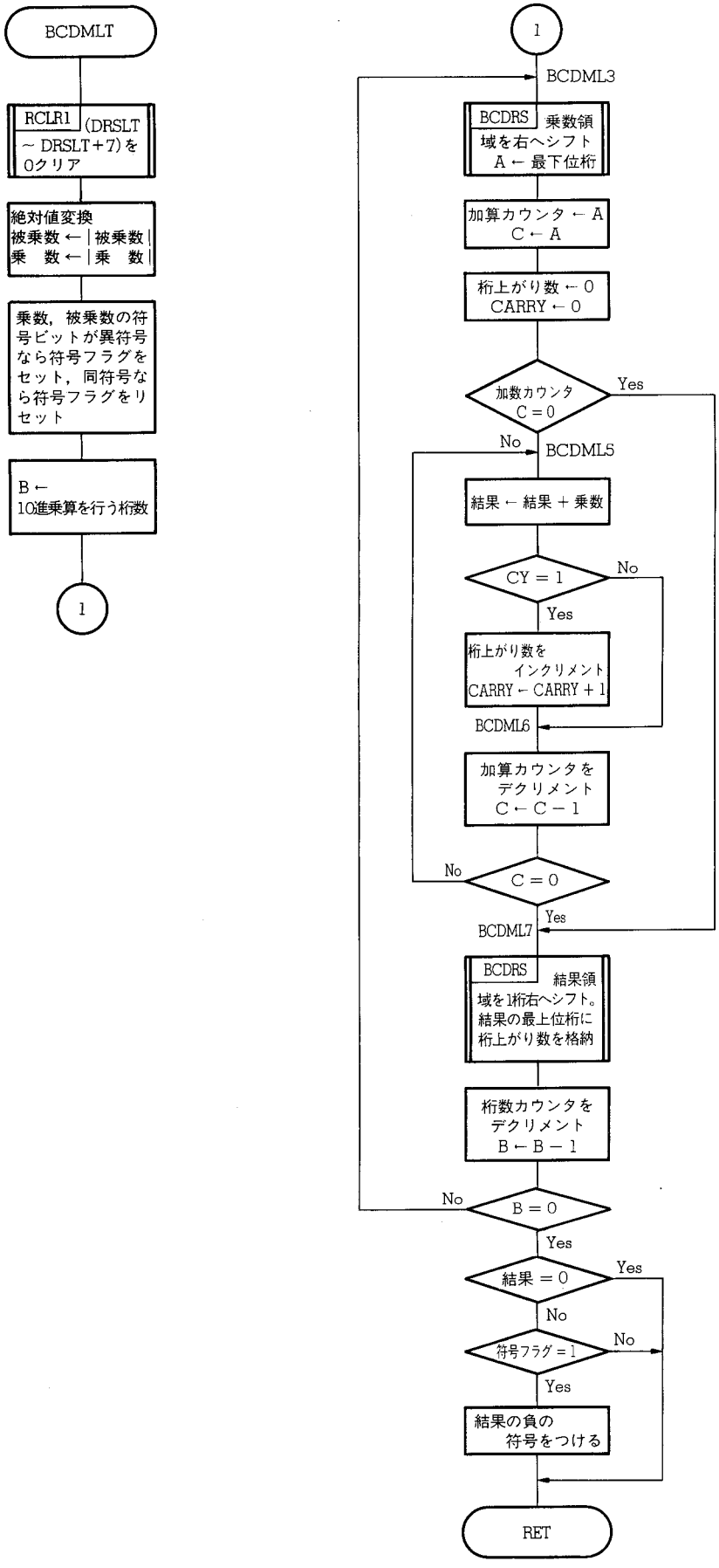
(k) 結果が0なら演算終了とする。

(l) 符号フラグが“1”ならば結果領域の符号ビットをセット（1）する。

(6) ステップ数

58ステップ

(7) フロー・チャート

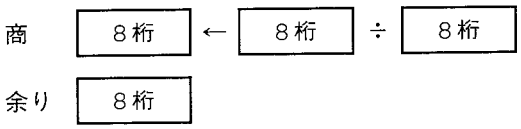

BCDMLT
RCLR1 (DRSLT ～ DRSLT+7)を0クリア
絶対値変換
被乗数 ← |被乗数|
乗　数 ← |乗　数|
乗数，被乗数の符号ビットが異符号なら符号フラグをセット，同符号なら符号フラグをリセット
B ←
10進乗算を行う桁数
1
1
BCDML3
BCDRS 乗数領域を右へシフト
A ← 最下位桁
加算カウンタ ← A
C ← A
桁上がり数 ← 0
CARRY ← 0
加数カウンタ C = 0
Yes
No
BCDML5
結果 ← 結果 + 乗数
CY = 1
No
Yes
桁上がり数を インクリメント
CARRY ← CARRY + 1
BCDML6
加算カウンタを デクリメント
C ← C − 1
No
C = 0
Yes
BCDML7
BCDRS 結果領域を1桁右へシフト。結果の最上位桁に桁上がり数を格納
桁数カウンタを デクリメント
B ← B − 1
No
B = 0
Yes
結果 = 0
Yes
No
符号フラグ = 1
No
Yes
結果の負の 符号をつける
RET

(8) プログラム・リスト

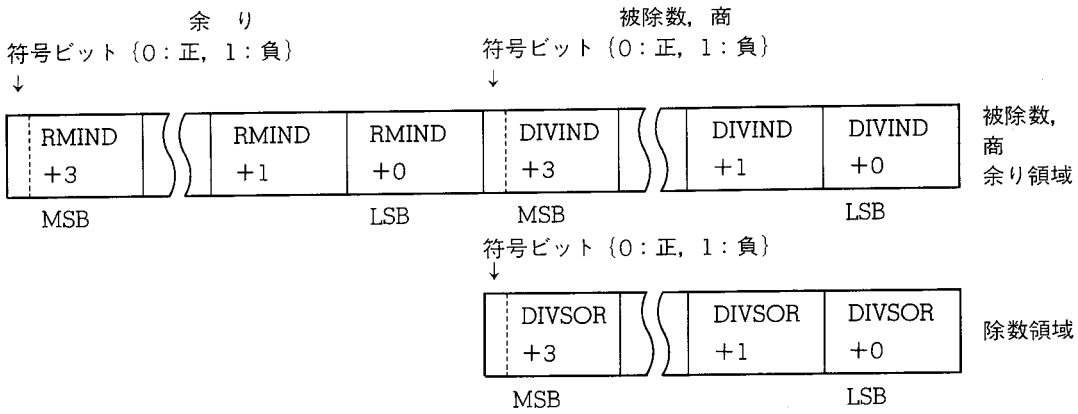
```
        NAME    BCDMLR

;*********************************************************
;*      decimal multiplication                           *
;*           16 digit <- 8 digit * 8 digit               *
;*       input condition                                 *
;*            multiplicand <- (DMLCND+3,...,DMLCND)      *
;*            multiplier   <- (DMLIER+3,...,DMLIER)      *
;*       output condition                                *
;*            result <- (DRSLT+7,...,DRSLT)              *
;*********************************************************

        PUBLIC  BCDMLT
        EXTRN   RCLR1,BCDRS,BCDRS1
        EXTRN   DMLCND,DMLIER,DRSLT
        EXTRN   CARRY           ; 1-byte carry area
        EXTBIT  SFLAG           ; work flag for sign flag
;
        CSEG
BCDMLT:
;
;       ****  result area 0-clear  ****
;
        MOV     C,#8            ; C-register <- 8
        MOVW    DE,#DRSLT       ; DE <- DRSLT
        CALL    !RCLR1
;
;       ****  check / sign  ****
;
        MOVW    DE,#DMLCND+3
        MOVW    HL,#DMLIER+3
        CLR1    SFLAG           ; clear sign-flag
        MOV     A,[DE]
        BF      A.7,$BCDML1
        CLR1    A.7
        MOV     [DE],A
        NOT1    SFLAG           ; not sign-flag
BCDML1:
        MOV     A,[HL]
        BF      A.7,$BCDML2
        CLR1    A.7
        MOV     [HL],A
        NOT1    SFLAG           ; not sign-flag
;
;       ****  digit counter set  ****
;
BCDML2:
        MOV     B,#8            ; B-register <- 8
;
;       ****  multiplier right shift  ****
;
BCDML3:
        MOVW    HL,#DMLIER+3
        MOV     C,#4            ; C-register <- 4
        CALL    !BCDRS
        MOV     C,A             ; C-register <- Acc
        MOV     CARRY,#0        ; carry <- 0
;
;       ****  check / multiplier = 0 ?  ****
;
        ADD     A,#0
```

```
        BZ      $BCDML7         ; if Acc = 0 then goto BCDML6
;
;       ****  result <- DMLCND + result  ****
;
BCDML4:
        MOVW    DE,#DMLCND      ; DE <- DMLCND
        MOVW    HL,#DRSLT+4     ; HL <- DRSLT+4
        CLR1    CY              ; clear carry
        PUSH    AX              ; save AX-register
        PUSH    BC              ; save BC-register
        MOV     C,#4            ; C-register <- 4
BCDML5:
        MOV     A,[HL]
        ADDC    A,[DE+]
        ADJBA                   ; decimal adjust
        MOV     [HL+],A
        DBNZ    C,$BCDML5
        POP     BC              ; load BC-register
        POP     AX              ; load AX-register
        BNC     $BCDML6
        INC     CARRY
BCDML6:
        DBNZ    C,$BCDML4
;
;       ****  result right shift with carry  ****
;
BCDML7:
        MOV     A,CARRY
        MOVW    HL,#DRSLT+7     ; HL <- DRSLT+7
        MOV     C,#8
        CALL    !BCDRS1
;
;       ****  check / multiply end ?  ****
;
        DBNZ    B,$BCDML3

;
;       ****  check / multiply = 0  ****
;
        MOVW    HL,#DRSLT
        MOV     C,#8
        MOV     A,#0
BCDML8:
        CMP     A,[HL+]
        BNZ     $BCDML9
        DBNZ    C,$BCDML8
        RET
;
;       ****  check / sign-flag  ****
;
BCDML9:
        BF      SFLAG,$BCDM10
        MOVW    HL,#DRSLT+7
        MOV     A,[HL]
        SET1    A.7
        MOV     [HL],A
BCDM10:
        RET
;
        END
```

2.2.4　10進除算



(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように被除数8桁，除数8桁をそれぞれ，次の領域に格納します。

- 被除数：DIVIND，DIVIND＋1，DIVIND＋2，DIVIND＋3
- 除数　：DIVSOR，DIVSOR＋1，DIVSOR＋2，DIVSOR＋3

(4) 出力条件

商と余りが次の領域に格納されます。

- 商が (1) で示す商領域　　：DIVIND，DIVIND＋1，DIVIND＋2，DIVIND＋3
- 余りが (1) で示す余り領域：RMIND，RMIND＋1，RMIND＋2，RMIND＋3

(5) 処理手順

この除算プログラムでは，被除数（DIVIND，…，DIVIND＋3），余り（RMIND，…，RMIND＋3）を８バイトの連続した領域とします。被除数，余りの１桁左シフトを行うことにより被除数の最上位桁が余りの最下位領域に転送され，商が最上位から１桁ずつ被除数の最下位領域に入ります。

また，商の各桁は，“余り－除数”の結果が負になるまで繰り返したときの回数とします。

次に処理手順を示します。

(a) 除数の０チェックを行う。０ならばエラー処理へ分岐する。

(b) 余り領域をクリア（0）する。

(c) 被除数，除数の絶対値をとる。商符号フラグをＸレジスタの第０ビット，余り符号フラグをＸレジスタの第１ビットとする。被除数，除数のどちらか一方だけが負ならば，商符号フラグをセット（1）する。

被除数が負ならば，余り符号フラグをセット（1）する。

(d) 被除数のバイト数として，Ｃレジスタに８を設定する。

(e) ８バイト続きの商，余り領域を４ビット左にシフトする。

(f) 10進減算“余り←余り－除数”を行う。

負ならば（h）へジャンプする。

(g) 商（DIVIND）をインクリメントする。(f)へジャンプする。

(h) 引きすぎたので復元するために，10進加算“余り←余り＋除数”を行う。

(i) Ｃレジスタをデクリメントし，０になるまで(e)から(h)を繰り返す。

(j) 商が０ならば（1）の処理へ。

(k) 商符号フラグが“1”ならば商領域の符号ビットをセット（1）する。

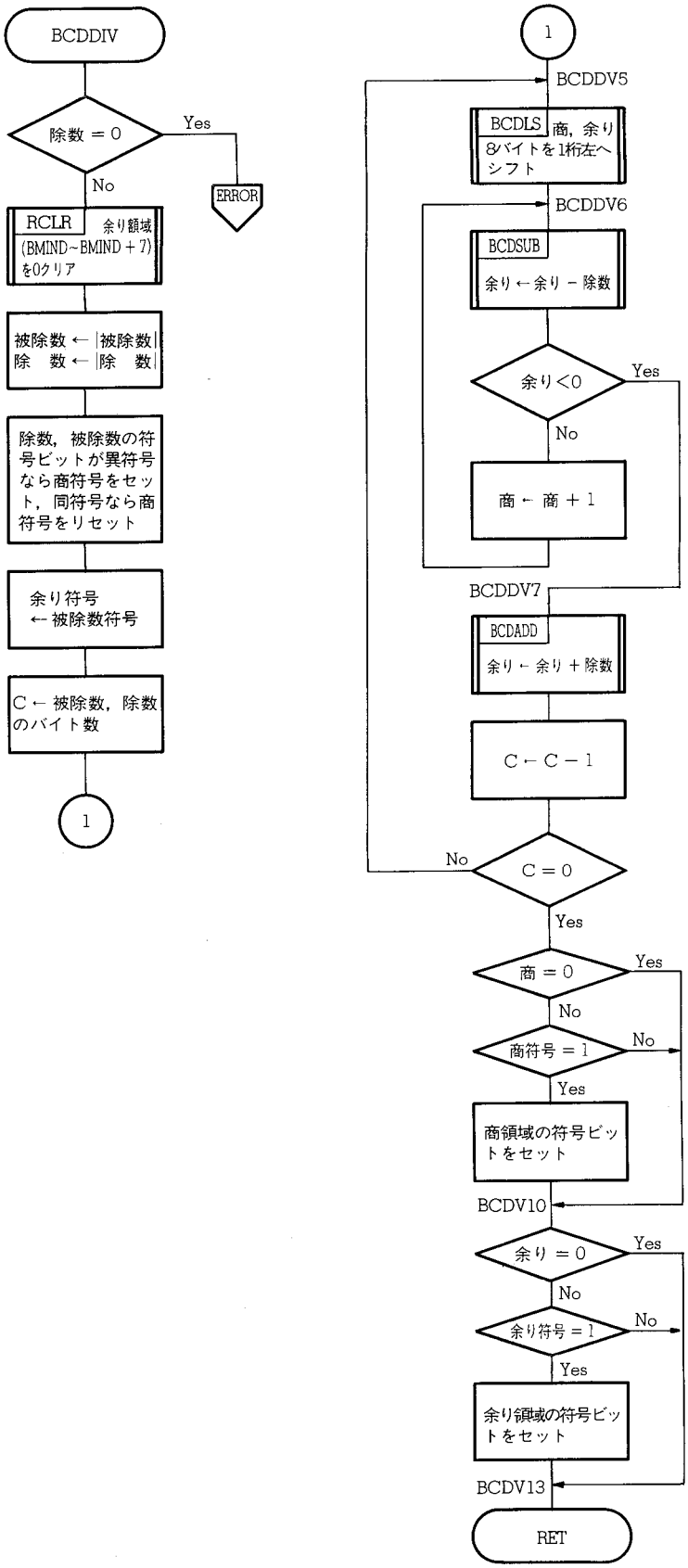
(l) 余りが０ならば演算を終了する。

(m) 余り符号フラグが“1”ならば余り領域の符号ビットをセット（1）する。

(6) ステップ数

70ステップ

(7) フロー・チャート


BCDDIV
除数 = 0
Yes
No
ERROR
RCLR 余り領域(BMIND~BMIND+7)を0クリア
被除数 ← |被除数|
除　数 ← |除　数|
除数，被除数の符号ビットが異符号なら商符号をセット，同符号なら商符号をリセット
余り符号 ← 被除数符号
C ← 被除数，除数のバイト数
1
1
BCDDV5
BCDLS 商，余り8バイトを1桁左へシフト
BCDDV6
BCDSUB 余り ← 余り − 除数
余り<0
Yes
No
商 ← 商 + 1
BCDDV7
BCDADD 余り ← 余り + 除数
C ← C − 1
No
C = 0
Yes
商 = 0
Yes
No
商符号 = 1
No
Yes
商領域の符号ビットをセット
BCDV10
余り = 0
Yes
No
余り符号 = 1
No
Yes
余り領域の符号ビットをセット
BCDV13
RET

(8) プログラム・リスト

```
        NAME    BCDIVR

;*************************************************
;*      decimal division                         *
;*           8 digit <- 8 digit / 8 digit        *
;*       input condition                          *
;*            dividend <- (DIVIND+3,...,DIVIND)   *
;*            divisor  <- (DIVSOR+3,...,DIVSOR)   *
;*       output condition                         *
;*            quotient  <- (DIVIND+3,...,DIVIND)  *
;*            remainder <- (RMIND+3,...,RMIND)    *
;*************************************************

        PUBLIC  BCDDIV
        EXTRN   ERROR,RCLR,BCDLS,BCDSUB,BCDADD
        EXTRN   DIVIND,DIVSOR,RMIND
;
SF_QUO  EQU     X.0
SF_REM  EQU     X.1
;
        CSEG
BCDDIV:
;
;       ****  check / divisor = 0 ?  ****
;
        MOV     C,#4            ; C-register <- 4
        MOV     A,#0            ; Acc <- 0
        MOVW    HL,#DIVSOR      ; HL <- DIVSOR
BCDDV1:
        CMP     A,[HL+]
        BNZ     $BCDDV2         ; (HL) = 0 ?
        DBNZ    C,$BCDDV1

        BR      ERROR           ; OVER FLOW
;
;       ****  result,remind 0-clear  ****
;
BCDDV2:
        MOVW    DE,#RMIND       ; DE <- RMIND
        CALL    !RCLR
;
;       ****  check / sign  ****
;
        CLR1    SF_QUO          ; clear quotient sign-flag
        CLR1    SF_REM          ; clear remainder sign-flag
        MOVW    HL,#DIVIND+3
        MOV     A,[HL]
        BF      A.7,$BCDDV3
        CLR1    A.7
        MOV     [HL],A
        SET1    SF_REM          ; set remainder sign-flag
        NOT1    SF_QUO          ; not quotient sign-flag
BCDDV3:
        MOVW    HL,#DIVSOR+3
        MOV     A,[HL]
        BF      A.7,$BCDDV4
        CLR1    A.7
        MOV     [HL],A
        NOT1    SF_QUO
```

```
;
;       ****  digit counter set  ****
;
BCDDV4:
        MOV     C,#8
;
;       ****  quotient,remind left shift  ****
;
BCDDV5:
        PUSH    BC
        MOVW    HL,#DIVIND      ; HL <- DIVIND
        MOV     C,#16/2         ; C-register <- 8
        CALL    !BCDLS          ; N-digit data left shift
;
;       ****  subtract divisor from dividend  ****
;
BCDDV6:
        MOVW    DE,#DIVSOR      ; DE <- DIVSOR
        MOVW    HL,#RMIND       ; HL <- RMIND
        CALL    !BCDSUB         ; decimal subtraction

        MOVW    HL,#RMIND+3
        MOV     A,[HL]
        BT      A.7,$BCDDV7     ; if borrow then goto BCDDV7

        MOV     A,#1
        MOVW    HL,#DIVIND
        ADD     A,[HL]          ; increment (DIVIND)
        MOV     [HL],A

        BR      BCDDV6
;
;       ****  if borrow then divisor + dividend  ****
;
BCDDV7:
        MOVW    DE,#DIVSOR      ; DE <- DIVSOR
        MOVW    HL,#RMIND       ; HL <- RMIND
        CALL    !BCDADD         ; decimal addition
;
;       ****  check / division end ?  ****
;
        POP     BC
        DBNZ    C,$BCDDV5
;
;       ****  check / quotient = 0  ****
;
        MOVW    HL,#DIVIND
        MOV     A,#0
        MOV     C,#4
BCDDV8:
        CMP     A,[HL+]
        BNZ     $BCDDV9
        DBNZ    C,$BCDDV8
        BR      BCDV10
;
;       ****  check / quotient sign-flag  ****
;
BCDDV9:
        BF      SF_QUO,$BCDV10
        MOVW    HL,#DIVIND+3
        MOV     A,[HL]
        SET1    A.7
        MOV     [HL],A
```

```
;
;       ****  check / remainder = 0  ****
;
BCDV10:
        MOVW    HL,#RMIND
        MOV     A,#0
        MOV     C,#4
BCDV11:
        CMP     A,[HL+]
        BNZ     $BCDV12
        DBNZ    C,$BCDV11
        RET
;
;       ****  check / remainder sign-flag  ****
;
BCDV12:
        BF      SF_REM,$BCDV13
        MOVW    HL,#RMIND+3
        MOV     A,[HL]
        SET1    A.7
        MOV     [HL],A
BCDV13:
        RET
;
        END
```

2.3　シフト処理

78K/II には汎用レジスタ（X, A, C, B, E, D, L, H）および汎用レシスタ・ペア（AX, BC, DE, HL）の1ビット単位シフト命令ならびに，[ROR4, ROL4] の4ビット単位シフト命令が用意されています。

このプログラム例では，次の2種類のシフト例を示します。

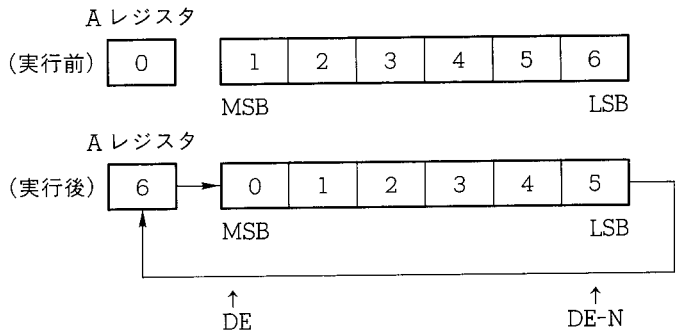
(a) バイト単位のシフト

(b) 4ビット単位のシフト

2.3.1　Nバイト・データの右シフト

メモリ内にあるNバイト・データの右シフトを行う例を示します。

このプログラムの実行後，次の図のようにAレジスタにあらかじめ設定しておいた値が最上位の1バイトに格納され，最下位の1バイトの内容がAレジスタに出力されます。



(1) 使用レジスタ

A, C, DE

(2) 入力条件

- A　←最上位メモリに転送する値
- C　←シフトの対象となるバイト数（N）
- DE ←Nバイト・データの最上位アドレス

(3) 出力条件

1バイト右シフトした結果が（1）で示す結果領域に格納されています。

最上位メモリにはAレジスタの内容が転送されます。

Aレジスタには［DE-N］の内容が転送されます。

(4) ステップ数

4ステップ

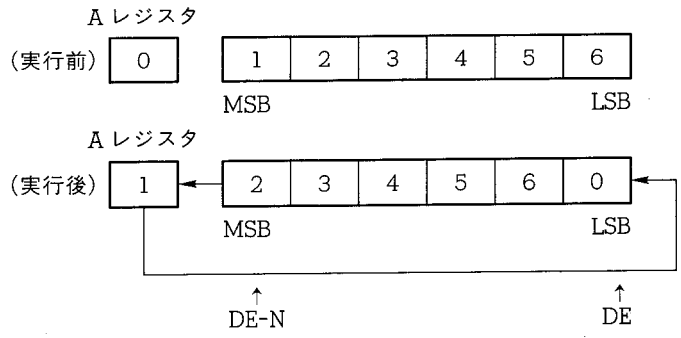
(5) プログラム・リスト

```
        NAME    BYTRSR

;*****************************************************
;*      N-byte data right shift                      *
;*          input condition                          *
;*              DE-register <- MSB of N-byte data    *
;*              C -register <- byte counter          *
;*          output condition                         *
;*              Acc <- LSB of N-byte data            *
;*****************************************************

        PUBLIC  BYTRST,BYTRS1
        PUBLIC  BYTLST,BYTLS1
;
        CSEG
BYTRST:
        MOV     A,#0            ; Acc <- 0
BYTRS1:
        XCH     A,[DE-]
        DBNZ    C,$BYTRS1
        RET
```

2.3.2　Nバイト・データの左シフト

メモリ内にあるNバイト・データの左シフトを行う例を示します。

このプログラムの実行後，次の図のようにAレジスタにあらかじめ設定しておいた値が最下位の1バイトに格納され，最上位の1バイトの内容がAレジスタに出力されます。



(1) 使用レジスタ

A，C，DE

(2) 入力条件

- A　←最下位メモリに転送する値
- C　←シフトの対象となるバイト数（N）
- DE　←Nバイト・データの最下位アドレス

(3) 出力条件

1バイト左シフトした結果が，(1) で示す結果領域に格納されています。

最下位メモリにはAレジスタの内容が転送されます。

Aレジスタには［DE+N］の内容が転送されます。

(4) ステップ数

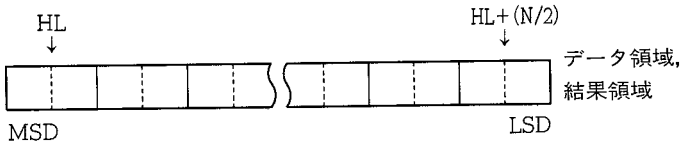
4ステップ

(5) プログラム・リスト

```
;*****************************************************
;*      N-byte data left shift                       *
;*          input condition                          *
;*              DE-register <- LSB of N-byte data    *
;*              C -register <- byte counter          *
;*          output condition                         *
;*              Acc <- MSB of N-byte data            *
;*****************************************************

BYTLST:
        MOV     A,#0            ; Acc <- 0
BYTLS1:
        XCH     A,[DE+]
        DBNZ    C,$BYTLS1
        RET

        END
```

2.3.3　N桁データの1桁右シフト（10進 1/10 処理）

(1) 使用メモリ



(2) 使用レジスタ

A, C, HL

(3) 入力条件

(1) で示すように，HL，Cレジスタの内容を次のように設定します。

- HL ← N桁データの最上位アドレス。
- C ← バイト数 (N/2)

(4) 出力条件

1桁右シフトした結果が (1) で示す結果領域に格納されています。

最上位桁にはAレジスタの内容が転送されます。

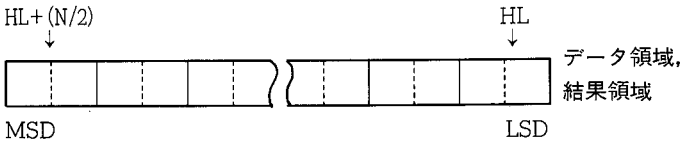
Aレジスタには最下位桁の内容が転送されます。

(5) プログラム・リスト

```
        NAME    BCDRSR

;*********************************************************
;*      N-digit data right shift                         *
;*          input condition                              *
;*              HL-register <- MSD of N-digit data       *
;*              C -register <- digit counter             *
;*          output condition                             *
;*              Acc <- LSD of N-digit data               *
;*********************************************************

        PUBLIC  BCDRS,BCDRS1
        PUBLIC  BCDLS,BCDLS1
;
        CSEG
BCDRS:
        MOV     A,#0            ; Acc <- 0
BCDRS1:
        ROR4    [HL]
        DECW    HL              ; decrement (HL)
        DBNZ    C,$BCDRS1
        RET
```

2.3.4　N桁データの1桁左シフト（10進 10倍 処理）

(1) 使用メモリ



(2) 使用レジスタ

A, C, HL

(3) 入力条件

(1) で示すように，HL，Cレジスタの内容を次のように設定します。

- HL ← N桁データの最下位アドレス。
- C ← バイト数 (N/2)。

(4) 出力条件

1桁左シフトした結果が (1) で示す結果領域に格納されています。

最下位桁にはAレジスタの内容が転送されます。

Aレジスタには最下位桁の内容が転送されます。

(5) プログラム・リスト

```
;*****************************************************
;*      N-digit data left shift                      *
;*          input condition                          *
;*              HL-register <- LSD of N-digit data   *
;*              C -register <- digit counter         *
;*          output condition                         *
;*              Acc <- MSD of N-digit data           *
;*****************************************************

BCDLS:
        MOV     A,#0
BCDLS1:
        ROL4    [HL]
        INCW    HL              ; increment (HL)
        DBNZ    C,$BCDLS1
        RET
;
        END
```

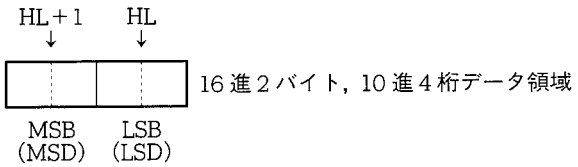
2.4　データ変換処理

ここでは，数値データの表現形式を16進と10進，16進とASCIIの各形式間で変換するプログラムについて説明します。

2.4.1　16進（HEX）を10進（BCD）に変換

16進2バイトのデータを10進4桁へ変換します。

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように，HLレジスタの内容を次のように設定します。

• HL ← 入力データ16進2バイトが格納してある領域の最下位アドレス。

(4) 出力条件

CY＝1：16進データが270FH（＝9999）より大きいため，変換できません。

CY＝0：変換された10進4桁が（HL，HL+1）に格納されます。

(5) 処理手順

この変換プログラムでは，10進4桁（2バイト）の変換値を最下位桁から1桁ずつ求めています。除数を10，被除数を16進データとし，除算結果の余りをBCDコードとして生成しています。次に処理手順を示します。

(a) 16進入力データと10000を比較する。

(b) 16進入力データが10000より大きい場合は，変換不能のためキャリー・フラグ＝1として変換を終了する。

(c) 変換桁数4をBレジスタに設定。

(d) 除数を10に設定。

(e) “16進入力データ/除数”を行う。

(f) 余りをAレジスタに設定し，結果領域を1桁右シフト。

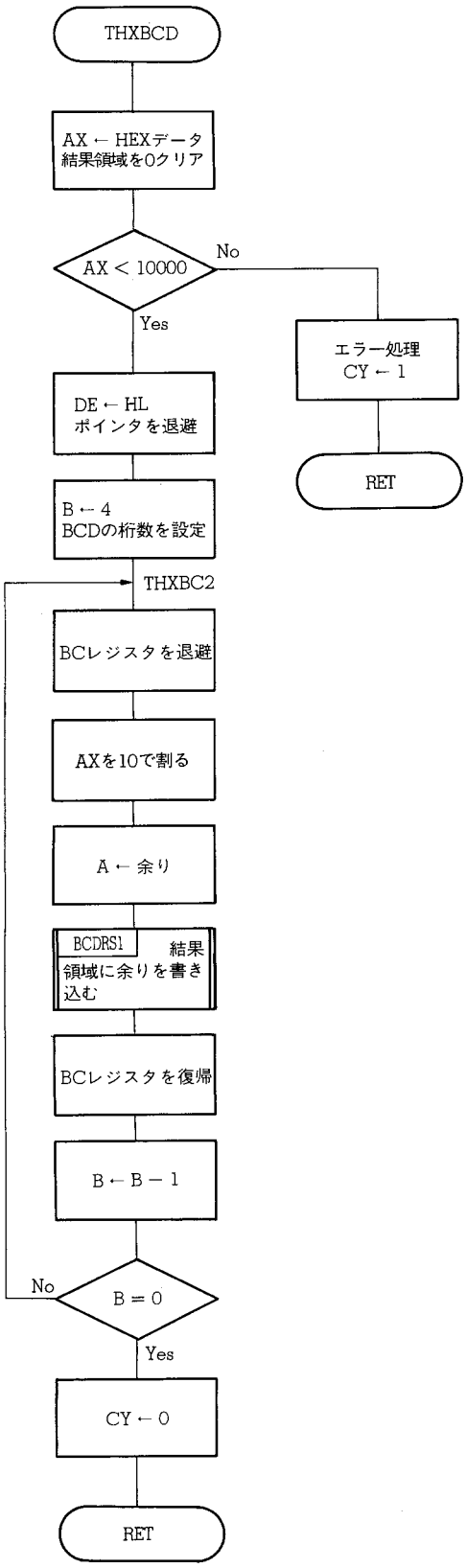
(g) Bレジスタをデクリメントし，0でなければ(e)から(f)を繰り返す。

(h) キャリー・フラグ＝0として変換終了。

(6) ステップ数

24ステップ

(7) フロー・チャート


THXBCD
AX ← HEXデータ
結果領域を0クリア
AX < 10000
No
Yes
エラー処理
CY ← 1
RET
DE ← HL
ポインタを退避
B ← 4
BCDの桁数を設定
THXBC2
BCレジスタを退避
AXを10で割る
A ← 余り
BCDRS1
結果領域に余りを書き込む
BCレジスタを復帰
B ← B − 1
No
B = 0
Yes
CY ← 0
RET

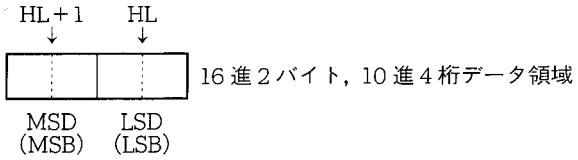
(8) プログラム・リスト

```
        NAME    TRBCDR

;*****************************************************
;*      transform BCD <- HEX                         *
;*         input condition                           *
;*              HL-register <- HEX-2byte data        *
;*                                   LSB address     *
;*         output condition                          *
;*              normal ... cy = 0                    *
;*                  decimal 4-digit -> (HL,HL+1)     *
;*              overflow ... cy = 1                  *
;*                  HEX data > 9999                  *
;*****************************************************
;
        PUBLIC  THXBCD
        EXTRN   BCDRS1
;
        CSEG
THXBCD:
        MOVW    AX,#0           ; AX <-> [HL](hex data)
        XCH     A,[HL+]
        XCH     A,X
        XCH     A,[HL-]

        CMPW    AX,#10000       ; hex data >= 10000 then ret.
        BC      $THXBC1
        SET1    CY              ; 'CY' <- 1
        RET
;
THXBC1:
        MOVW    DE,HL           ; save HL-register
        MOV     B,#4            ; loop counter
THXBC2:
        PUSH    BC              ; save loop counter
        MOV     B,#10           ; set divisor
        DIVUW   B               ; AX / C
        PUSH    AX              ; save AX-register
        MOV     A,B
        MOVW    HL,DE           ; load HL-register

        MOV     C,#4            ; set length
        INCW    HL              ; set pointer
        CALL    !BCDRS1         ; 1-digit left shift
        POP     AX              ; load AX-register
        POP     BC              ; restore loop counter
        DBNZ    B,$THXBC2

        CLR1    CY              ; 'CY' <- 0
        RET
;
        END
```

2.4.2　10進（BCD）を16進（HEX）に変換

10進4桁のデータを16進2バイトに変換します。

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL

(3) 入力条件

(1) で示すように，HLレジスタの内容を次のように設定します。

• HL ← 入力データ10進4桁が格納してある領域の最下位桁アドレス。

(4) 出力条件

CY = 1：入力データが10進でないため，変換できません。

CY = 0：変換された16進2バイトが（HL，HL+1）に格納されます。

(5) 処理手順

この変換プログラムでは，10進入力データを最上位桁から1桁左シフトにより，1桁ずつAレジスタに転送し，次の処理を4回繰り返して変換します。

（格納エリア）←（格納エリア）×10+Aレジスタ

次に処理手順を示します。

(a) BCDコードの桁数4をBレジスタに設定。

(b) 変換結果格納レジスタ（DE）をクリア（0）する。

(c) 10進入力データを1桁左シフトし，最上位桁をAレジスタに読み込む。

(d) 最上位桁が10進データ(0-9)かをチェックする。10進でなければ，変換不能としてキャリー・フラグをセット（1）する。

(e) “変換結果格納レジスタ ← 変換値格納レジスタ×10+Aレジスタ”を行う。

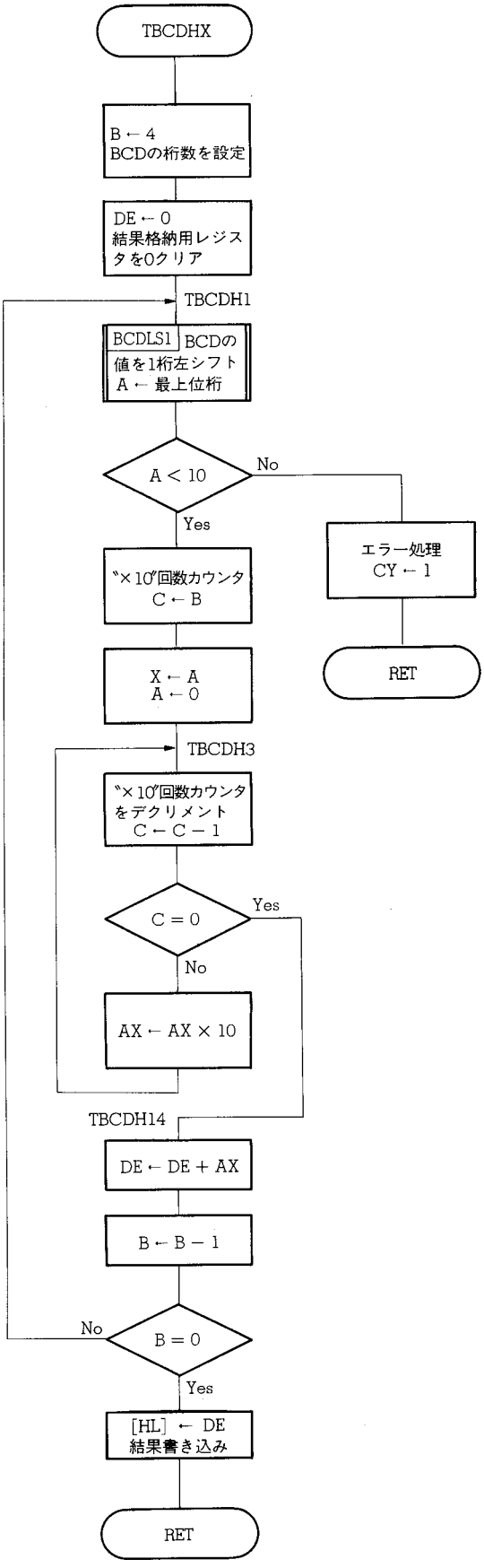
(f) Bレジスタをデクリメントし，90でなければ(c)から(e)を繰り返す。

(g) 変換結果格納レジスタ（DE）を格納領域に格納する。

(6) ステップ数

32 ステップ

(7) フロー・チャート


TBCDHX
B ← 4
BCDの桁数を設定
DE ← 0
結果格納用レジスタを0クリア
TBCDH1
BCDLS1 BCDの値を1桁左シフト
A ← 最上位桁
A < 10
No
Yes
エラー処理
CY ← 1
RET
"×10"回数カウンタ
C ← B
X ← A
A ← 0
TBCDH3
"×10"回数カウンタをデクリメント
C ← C − 1
C = 0
Yes
No
AX ← AX × 10
TBCDH14
DE ← DE + AX
B ← B − 1
B = 0
No
Yes
[HL] ← DE
結果書き込み
RET

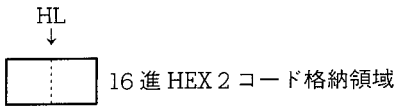
(8) プログラム・リスト

```
        NAME    TRHEXR

;*************************************************************
;*      transform HEX <- BCD                                 *
;*          input condition                                  *
;*              HL-register <- decimal 4 digit data          *
;*                                      LSD address          *
;*          output condition                                 *
;*              normal ... cy = 0                            *
;*                    HEX 2 byte -> (HL,HL+1)                *
;*              error  ... cy = 1                            *
;*************************************************************

        PUBLIC  TBCDHX
        EXTRN   BCDLS1
;
        CSEG
TBCDHX:
        MOV     B,#4            ; BCD length
        MOVW    DE,#0           ; result work
TBCDH1:
        PUSH    HL              ; save pointer
        MOV     C,#2            ; shift counter
        MOV     A,#0
        CALL    !BCDLS1         ; BCD left shift
        POP     HL              ; restore pointer
        CMP     A,#10           ; error check
        NOT1    CY
        BNC     $TBCDH2
        RET                     ; error return
TBCDH2:
        MOV     C,B
        MOV     X,#0
        XCH     A,X
TBCDH3:
        DEC     C
        BZ      $TBCDH4

        PUSH    BC              ; AX <- AX * 10
        MOVW    BC,AX
        SHLW    AX,2
        ADDW    AX,BC
        SHLW    AX,1
        POP     BC
        BR      TBCDH3
TBCDH4:
        ADDW    AX,DE           ; result addition
        MOVW    DE,AX
        DBNZ    B,$TBCDH1       ; check length

        MOVW    AX,DE           ; write result to memory
        XCH     A,X
        MOV     [HL+],A
        MOV     A,X
        MOV     [HL],A
        RET
;
        END
```

2.4.3　ASCIIを16進（HEX）に変換

ASCII2コード（30H-39H，41H-46H）を16進HEX2コード（0-FFH）に変換します。

(1) 使用メモリ



(2) 使用レジスタ

A，C，B，HL

(3) 入力条件

(1) で示すように，BC，HLレジスタの内容を次のように設定します。

- BC ← ASCII2コード
- HL ←16進2コード格納領域のアドレス

(4) 出力条件

CY = 1：入力データがASCIIコードでないため，変換できません。

CY = 0：変換された16進2コードが（1）で示す領域に格納されます。

(5) 処理手順

(a) ASCII上位1コード（Bレジスタ）をAレジスタに読み込む。

(b) Aレジスタの内容が30H-39H，41H-46Hの範囲にあるかチェックする。範囲になければ，変換不能としてキャリー・フラグをセット（1）する。

(c) Aレジスタの内容が30H-39Hならば30Hを引く。
Aレジスタの内容が41H-46Hならば37Hを引く。

(d) HLレジスタの示すアドレスの内容を4ビット・シフトし，下位4ビットにAレジスタの内容を格納する。

(e) ASCII下位4ビット1コード（Cレジスタ）をAレジスタに読み込み，(b) から (d) の処理を行う。

(6) ステップ数

19ステップ

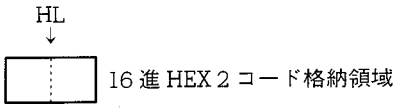
(7) プログラム・リスト

```
        NAME    GHEXR

;*************************************************
;*      transform   HEX <-  ASCII                *
;*                  (2code)  (2code)             *
;*                                               *
;*              input  condition                 *
;*                BC-register <- ASCII           *
;*                                               *
;*              output condition                 *
;*                (HL)  <-  hex                  *
;*************************************************

        PUBLIC  GETHEX
        PUBLIC  SHEX
;
        CSEG
GETHEX:
        MOV     A,B             ; ASCII upper-code load
        CALL    !SHEX           ; get  hex 1th code
        BC      $GTHEX1

        ROL4    [HL]
        MOV     A,C             ; ASCII lower-code load
        CALL    !SHEX           ; get  hex 2th code
        BC      $GTHEX1
        ROL4    [HL]
GTHEX1:
        RET

;*********************************************
;*      subroutine  /  get hex 1-code(Acc)   *
;*********************************************

SHEX:
        CMP     A,#'9'          ; check / ASCII > 39H
        BNC     $SHEX1
        SUB     A,#30H
        RET
SHEX1:
        CMP     A,#'F'          ; check / ASCII < 46H
        BNC     $SHEX2
        SUB     A,#37H
        RET
SHEX2:
        SET1    CY              ; error
        RET
;
        END
```

2.4.4　16進（HEX）をASCIIに変換

16進HEX2コード（0-FFH）をASCII2コード（30H-39H，41H-46H）に変換します。

(1) 使用メモリ



(2) 使用レジスタ

A，C，B，HL

(3) 入力条件

(1) で示すように，HLレジスタの内容を次のように設定します。

• HL ← 16進2コード格納領域のアドレス

(4) 出力条件

BC ← 変換されたASCII2コード

(5) 処理手順

(a) HLレジスタの示すアドレスの上位4ビットをAレジスタに転送。

(b) Aレジスタが10以上かチェックする。10未満ならば(d)の処理へ。

(c) Aレジスタに7を加える。

(d) Aレジスタに30Hを加える。

(e) BレジスタにAレジスタを転送する。

(f) HLレジスタの下位4ビットをAレジスタに転送する。

(g) (b)から(c)の処理を行い，CレジスタにAレジスタの内容を転送する。

(6) ステップ数

14ステップ

(7) プログラム・リスト

```
        NAME    ASCII

;*************************************************
;*      transform    ASCII  <-  HEX              *
;*                  (2code)    (2code)          *
;*                                              *
;*              input  condition                *
;*                 (HL) <- hex 2-code           *
;*              output condition                *
;*                 BC-register <- ASCII 2-code  *
;*************************************************

        PUBLIC  GETASC
        PUBLIC  SASC
;
        CSEG
GETASC:
        MOV     A,#0
        ROL4    [HL]            ; hex upper code load
        CALL    !SASC
        MOV     B,A             ; store result

        MOV     A,#0
        ROL4    [HL]            ; hex lower code load
        CALL    !SASC
        MOV     C,A             ; store result
        RET

;*******************************************************
;*      subroutine  /  get ASCII 1-code(BC-register) *
;*******************************************************

SASC:
        CMP     A,#0AH          ; check / hex > 9
        BC      $SASC1
        ADD     A,#07H          ; bias (+7)
SASC1:
        ADD     A,#30H          ; bias (+30H)
        RET
;
        END
```

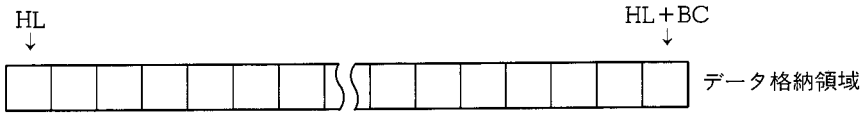
2.5　データ処理

ここでは，データ処理の例としてデータの整列ならびに検索のプログラム例を示します。

2.5.1　データの整列

1つのデータ長が8ビットであるデータ・ファイルの内容を昇順に整列します。整列の方法としてはバブル・ソートを用いています。

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，HL（ただし，HLレジスタは保持されます）。

(3) 入力条件

(1) で示すように，HL，BCレジスタの内容を次のように設定します。

- HL ← 整列を行うデータ列の先頭アドレス
- BC ← データの個数（バイト数）

(4) 出力条件

(1) で示された領域のデータが昇順に整列されます。

データ列の先頭アドレスを示すHLレジスタの値は保持されます。

(5) 処理手順

この整列プログラムでは，整列の方法としてバブル・ソートを用いています。次に処理手順を示します。

(a) 交換が行われたことを示すフラグ CHNGFL フラグをクリア（0）する。

(b) データのバイト数を示す BC レジスタの値をデクリメントする。0 ならば整列処理終了。

(c) HL レジスタ，BC レジスタを退避する。

(d) （HL）の値と次のアドレス（HL＋1）の値を比較する。（HL）の値と等しい場合または（HL）の値が大きい場合は(f)の処理へ。

(e) HL レジスタで示されるメモリの内容と，HL レジスタ＋1 で示されるメモリの内容を交換し，CHNGFL フラグをセット（1）する。

(f) HL レジスタをインクリメント，BC レジスタをデクリメントする。

(g) BC レジスタの値が 0 でなければ，(d)から(f)の処理を繰り返す。

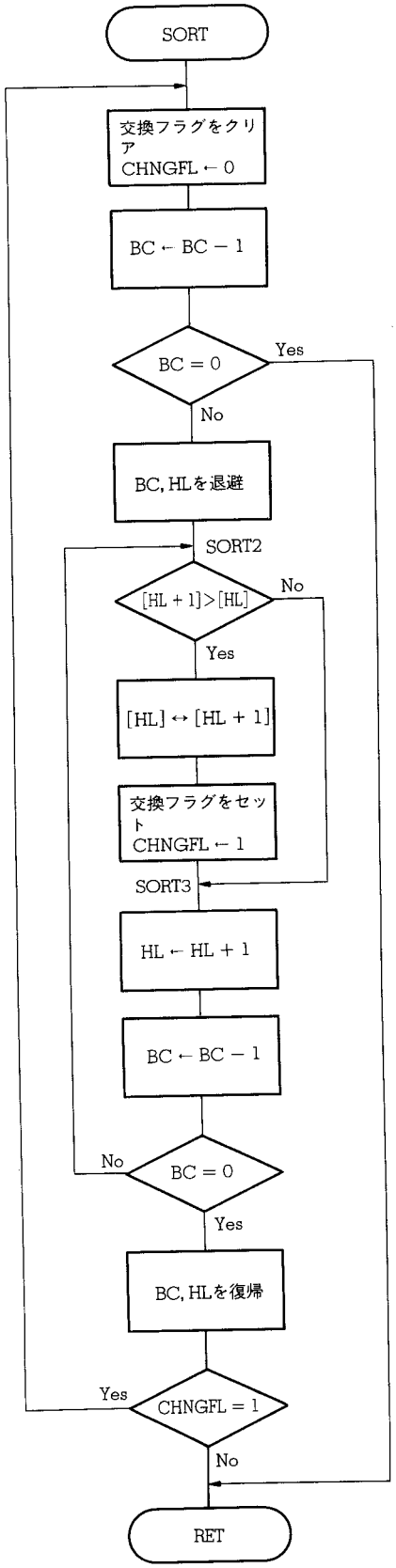
(h) HL レジスタ，BC レジスタを復帰する。

(i) 交換フラグ CHNGFL がセット（1）されていれば，(a)から(h)の処理を繰り返す。セット（1）されていなければ，整列処理を終了する。

(6) ステップ数

24 ステップ

(7) フロー・チャート


SORT
交換フラグをクリア
CHNGFL ← 0
BC ← BC − 1
BC = 0
Yes
No
BC, HLを退避
SORT2
[HL + 1]>[HL]
No
Yes
[HL] ↔ [HL + 1]
交換フラグをセット
CHNGFL ← 1
SORT3
HL ← HL + 1
BC ← BC − 1
No
BC = 0
Yes
BC, HLを復帰
Yes
CHNGFL = 1
No
RET

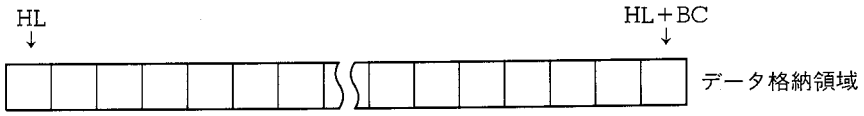
(8) プログラム・リスト

```
        NAME    SORTR

;*************************************************
;*      bubble sort                              *
;*       input condition                         *
;*            BC-register <- number of data      *
;*            HL-register <- data top.address    *
;*        output condition                        *
;*            HL-register <- data top.address    *
;*************************************************

        PUBLIC  SORT

        BSEG
CHNGFL  DBIT                    ; change-flag

SORT_D  DSEG    SADDR
CNTSTK: DS      1               ; counter save area (saddr area)
;
        CSEG
SORT:
        CLR1    CHNGFL          ; change-flag <- 0
        DECW    BC
        MOV     A,B
        OR      A,C
        BNZ     $SORT1
        RET
SORT1:
        PUSH    BC              ; save pointer/counter
        PUSH    HL
SORT2:
        MOV     A,[HL]          ; change process
        CMP     A,[HL+1]
        BC      $SORT3          ; A <= [HL+1] goto SORT3
        BZ      $SORT3
        XCH     A,[HL+1]
        MOV     [HL],A
        SET1    CHNGFL          ; change-flag <- 1
SORT3:
        INCW    HL              ; increment pointer
        DECW    BC
        MOV     A,B
        OR      A,C
        BNZ     $SORT2
        POP     HL              ; restore pointer/counter
        POP     BC
        BT      CHNGFL,$SORT
        RET
;
        END
```

2.5.2　データの検索

特定のデータを検索し，発見したらそのデータの格納アドレスを返します。検索の方法として２分探索を用いています。

(1) 使用メモリ



(2) 使用レジスタ

X，A，C，B，DE，HL（ただし，AXレジスタは保持されます）。

(3) 入力条件

(1) で示すように，A，HL，BCレジスタの内容を次のように設定します。

- A　← 検索データ
- HL ← 検索を行うデータ列の先頭アドレス
- BC ← データの個数（バイト数）

(4) 出力条件

- CY ＝ 0：HLレジスタに検索データが置かれているアドレスが出力されます。
- CY ＝ 1：検索データが発見されなかった場合です。HLレジスタの値は不定となります。

(5) 処理手順

この検索プログラムでは，検索の方法として２分探索（バイナリ・サーチ）を用いています。次に処理手順を示します。

(a)　検索データを退避する。

(b)　DEレジスタにデータの先頭アドレス，BCレジスタにデータの最終アドレスを設定する。

(c)　検索データを復帰する。

(d)　HLレジスタに検索範囲の中間アドレスを設定する。

(e)　DEレジスタとBCレジスタの値を比較して，DE ＜ BCならば検索データを復帰する。その後，キャリー・フラグをセット（1）して処理を終了する。

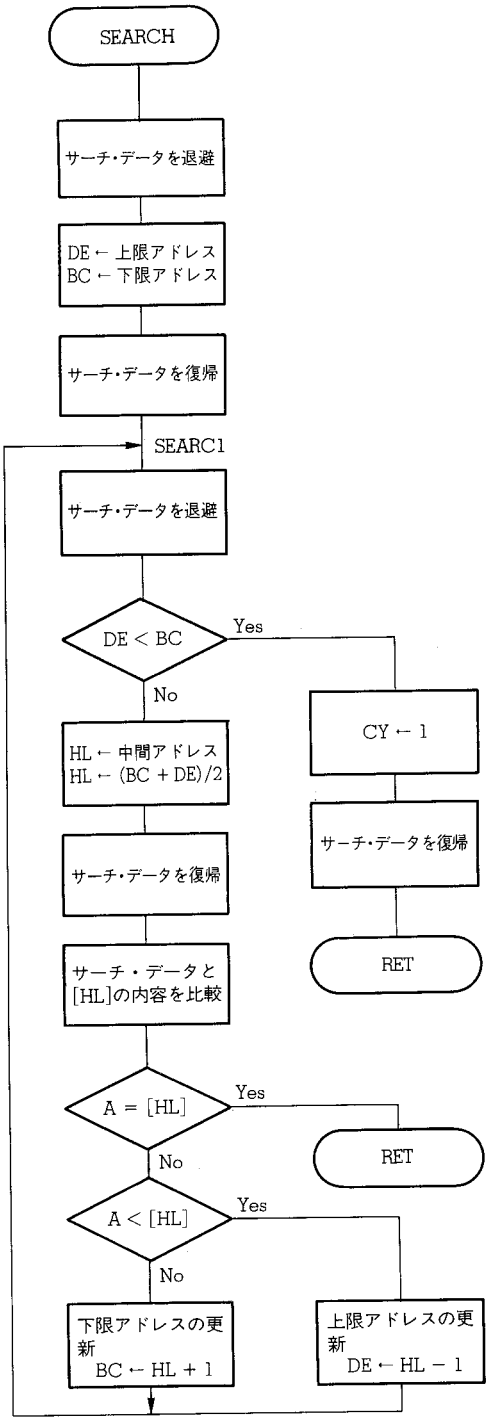
(f)　検索データとHLレジスタで示される内容を比較する。一致（発見）した場合は検索処理を終了する。

(g)　キャリー・フラグ＝１ならば，“BC ← HL－1”を行い，最終アドレスをあらたに設定する。CYフラグ＝０ならば，“DE ← HL＋1”を行い先頭アドレスをあらたに設定する。(d) へジャンプ。

(6) ステップ数

27 ステップ

(7) フロー・チャート


SEARCH
サーチ・データを退避
DE ← 上限アドレス
BC ← 下限アドレス
サーチ・データを復帰
SEARC1
サーチ・データを退避
DE < BC
Yes
No
CY ← 1
HL ← 中間アドレス
HL ← (BC + DE)/2
サーチ・データを復帰
サーチ・データを復帰
RET
サーチ・データと
[HL]の内容を比較
A = [HL]
Yes
RET
No
A < [HL]
Yes
No
下限アドレスの更新
BC ← HL + 1
上限アドレスの更新
DE ← HL − 1

(8) プログラム・リスト

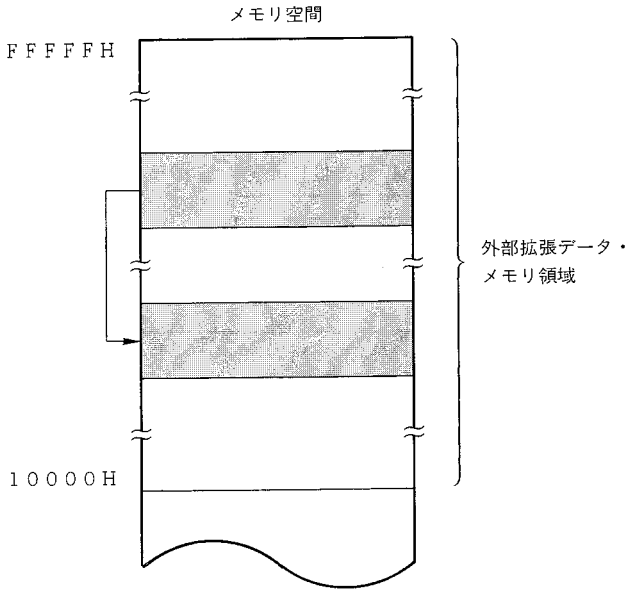
```
        NAME    SEARCR

;*****************************************************
;*      bubble sort                                  *
;*       input condition                             *
;*            A-register  <- search data             *
;*            BC-register <- number of data          *
;*            HL-register <- data top.address        *
;*        output condition                           *
;*            HL-register <- found data address      *
;*****************************************************

        PUBLIC  SEARCH
;
        CSEG
SEARCH:
        PUSH    AX              ; save search data
        MOVW    AX,HL
        ADDW    AX,BC
        MOVW    DE,AX           ; DE-register <- upper.address
        MOVW    BC,HL           ; BC-register <- lower.address
        POP     AX              ; restore search data
SEARC1:
        PUSH    AX              ; save search data
        MOVW    AX,DE           ; HL-register <- center.address
        SUBW    AX,BC
        BC      $SEARC4         ; search end check
        SHRW    AX,1
        ADDW    AX,BC
        MOVW    HL,AX

        POP     AX
        CMP     A,[HL]
        BNZ     $SEARC2
        RET                     ; found data
SEARC2:
        BC      $SEARC3
        INCW    HL              ; 'CY' = 0
        MOVW    BC,HL
        BR      SEARC1
SEARC3:
        DECW    HL              ; 'CY' = 1
        MOVW    DE,HL
        BR      SEARC1
SEARC4:
        POP     AX
        SET1    CY
        RET
;
        END
```

2.6　外部拡張データ・メモリ空間におけるデータ転送

外部拡張データ・メモリ（10000H-FFFFFH）間で，1バイト・データの連続転送を行います。

(1) 使用メモリ

図2-3　外部拡張データ・メモリ空間におけるデータ転送のメモリ・ブロック図



(2) 使用レジスタ

A，B，DE，HL（レジスタ・バンク0）

(3) 入力条件

表2-1のパラメータを定義します。

表2-1　外部拡張データ・メモリ空間におけるデータ転送の入力パラメータ

パラメータ	固定/可変	内　　容
TRSADR	固定値	転送元アドレスの下位16ビット
TRDADR	固定値	転送先アドレスの下位16ビット
MBANKS	固定値	転送元アドレスの上位4ビット（メモリ・バンク）
MBANKD注	固定値	転送先アドレスの上位4ビット（メモリ・バンク）
TRCOUNT	固定値	転送データ数

注　μPD78224シリーズでは，MBANKDは0にして使用してください。

(4) 出力条件

なし

(5) 処理手順

転送元メモリ・バンク（0H-FH）をP6レジスタに，転送先メモリ・バンク（0H-FH）をPM6レジスタに設定します。

μPD78224シリーズでは，PM6の下位4ビットは0固定です。

(a)　1Mバイト拡張モードに設定します（下図の**モード・レジスタ設定例**参照）。

メモリ拡張モード・レジスタ

	7	6	5	4	3	2	1	0	
MM	IFCH	MM6	PW21	PW20	0	MM2	MM1	MM0	（リセットで20H）
設定例	×	1	×	×	0	×	×	×	×：操作しません

MM2	MM1	MM0	モード	P50-P57	P40-P47	P65	P64
0	0	0	シングル・チップ・モード	ポート・モード	入力ポート	ポート・モード	
0	0	1			出力ポート		
1	1	1	外部メモリ拡張モード	A8-A15	AD0-AD7	$\overline{WR}$	$\overline{RD}$

PW21	PW20	ウエイト数（指定範囲 00000H-0FFFFH）
0	0	0ウエイト
0	1	1ウエイト
1	0	2ウエイト
1	1	$\overline{WAIT}$ 端子入力のロウ・レベル期間に相当するウエイト数

MM6	1Mバイト拡張モード指定
0	P60-P63は汎用出力ポート
1	P60-P63の出力ラッチ（P60-P63）は外部メモリ拡張モード時の上位アドレス（A16-A19）を記憶し，P60-P63はA16-A19の出力端子として機能する

IFCH	内部フェッチ・サイクル制御
0	外部ROMフェッチ・サイクルと同様な命令実行サイクルとなる
1	高速に内部ROMフェッチ動作を行う （実行サイクルは外部ROMフェッチのときより速くなる）

(b)　転送データ数，転送元，転送先アドレスを設定します。

(c)　データ数のカウンタ（Bレジスタ）が，0になるまでポインタを進めながら，書き込みを行います。

(6) ステップ数

10ステップ

(7) 使用スタック

2バイト：1レベル

(8) プログラム使用例

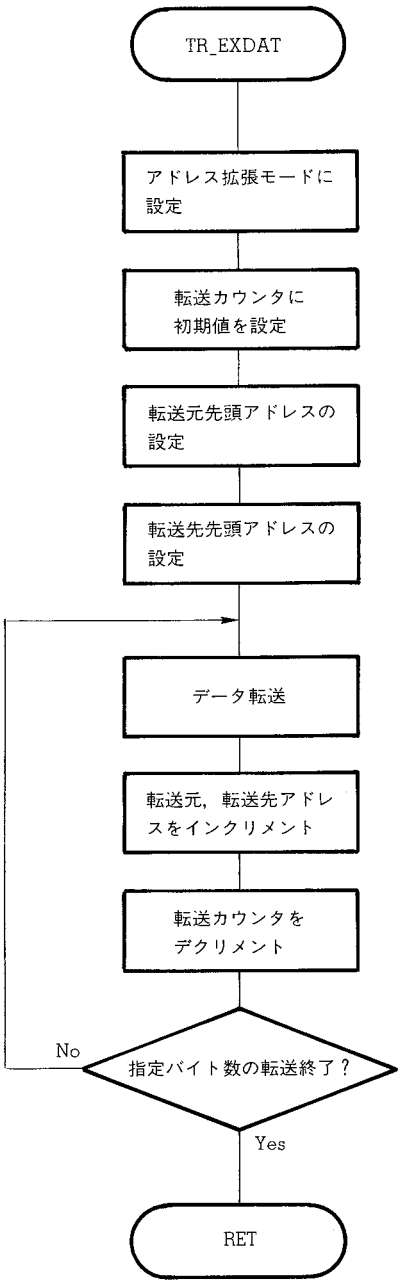
60000H番地から32バイトの連続データを，40000H番地以降に転送する場合の使用例を示します。

ユーザ・プログラムで次のようにパラメータを定義し，TR_EXDATサブルーチンを呼び出してください。

```
        PUBLIC  TRSADR,TRDADR,TRCOUNT   ; data
        PUBLIC  MBANKS,MBANKD           ; data
        EXTRN   TR_EXDAT                ; package

;       --- define data ---

TRSADR  EQU     0000H                   ; source address of transmission
TRDADR  EQU     0000H                   ; destination address of transmission
TRCOUNT EQU     32                      ; data count of transmission
MBANKS  EQU     6                       ; memory bank NO.  (source)
MBANKD  EQU     4                       ;                  (destination)

        CSEG
          :
          :
        CALL    !TR_EXDAT               ; data transmission package

          :
          :
```

備考　外部拡張データ・メモリ空間において，リロケータブルにシンボル定義を行う場合は，リンク時にディレクティブ・ファイルを使用します。詳細は，**付録　B.3　1Mバイト拡張データ・メモリ空間の使用方法**を参照してください。

(9) フロー・チャート


TR_EXDAT
アドレス拡張モードに設定
転送カウンタに初期値を設定
転送元先頭アドレスの設定
転送先先頭アドレスの設定
データ転送
転送元，転送先アドレスをインクリメント
転送カウンタをデクリメント
No
指定バイト数の転送終了？
Yes
RET

(10) プログラム・リスト

```
        NAME    TREXDAT
;*************************************************************
;*      DATA TRANSMISSION IN                                 *
;*              EXTERNALLY EXTENDED MEMORY AREA              *
;*************************************************************

        PUBLIC  TR_EXDAT                ; package
        EXTRN   TRSADR,TRDADR,TRCOUNT   ; data
        EXTRN   MBANKS,MBANKD           ; data

EXDATCS CSEG
TR_EXDAT:
        OR      MM,#01000000B           ; extended address mode
        MOV     B,#TRCOUNT              ; count of transmission
        MOV     P6,#MBANKS              ; memory bank NO. (source)
        MOV     PM6,#MBANKD             ;                 (destination)
        MOVW    HL,#TRSADR              ; source address of transmission
        MOVW    DE,#TRDADR              ; destination address of transmission

TR_LOOP:
        MOV     A,&[HL+]                ; transmit data
        MOV     [DE+],A
        DBNZ    B,$TR_LOOP              ; transmit until counter is 0

        RET
        END
```

第3章　タイマ/カウンタのプログラム例

78K/II シリーズのタイマ/カウンタの構成および機能は，各製品で異なっています。
それぞれの構成および機能を表3-1，表3-2，表3-3に示します。

表3-1　μPD78214 シリーズのタイマ/カウンタの機能と種類

種類と機能＼ユニット		16ビット・タイマ/カウンタ	8ビット・タイマ/カウンタ1	8ビット・タイマ/カウンタ2	8ビット・タイマ/カウンタ3
種類	インターバル・タイマ	2ch	2ch	2ch	1ch
	外部イベント・カウンタ	−	−	○	−
	ワンショット・タイマ	−	−	○	−
機能	タイマ出力	2ch	−	2ch	−
	トグル出力	○	−	○	−
	PWM/PPG 出力	○	−	○	−
	ワンショット・パルス出力	−	−	−	−
	リアルタイム出力	−	○	−	−
	パルス幅測定	○	○	○	−
	割り込み要求数	2	2	2	1
	シリアル・インタフェースのクロック・ソース	−	−	−	○

表3-2　μPD78218Aシリーズ，78234シリーズ，78244シリーズのタイマ/カウンタの機能と種類

種類と機能＼ユニット		16ビット・タイマ/カウンタ	8ビット・タイマ/カウンタ1	8ビット・タイマ/カウンタ2	8ビット・タイマ/カウンタ3
種類	インターバル・タイマ	2ch	2ch	2ch	1ch
	外部イベント・カウンタ	−	−	○	−
	ワンショット・タイマ	−	−	○	−
機能	タイマ出力	2ch	−	2ch	−
	トグル出力	○	−	○	−
	PWM/PPG出力	○	−	○	−
	ワンショット・パルス出力	○	−	−	−
	リアルタイム出力	−	○	−	−
	パルス幅測定	○	○	○	−
	割り込み要求数	2	2	2	1
	シリアル・インタフェースのクロック・ソース	−	−	−	○

表3-3　μPD78224シリーズのタイマ/カウンタの機能と種類

種類と機能＼ユニット		16ビット・タイマ/カウンタ	8ビット・タイマ/カウンタ1	8ビット・タイマ/カウンタ2
種類	インターバル・タイマ	2ch	2ch	1ch
	外部イベント・カウンタ	−	−	○
	ワンショット・タイマ	−	−	○
機能	タイマ出力	2ch	−	2ch
	トグル出力	○	−	○
	PWM/PPG出力	−	−	−
	ワンショット・パルス出力	−	−	−
	リアルタイム出力	−	○	−
	割り込み要求数	2	2	1
	パルス幅測定	○	○	○
	シリアル・インタフェースのクロック・ソース	−	−	−

ここでは，これらのタイマ/カウンタの機能のうち次のプログラム例を紹介します。

(i) 内部インターバル・タイマ（**3.1**）

(ii) トグル出力（**3.2**）

(iii) フリー・ランニング・インターバル・タイマ（**3.3**）

(iv) PWM/PPG出力（**3.4**）

(v) パルス周期測定（**3.5**）

3.1　内部インターバル・タイマ

インターバル・タイマはCPUに対して一定周期で割り込み要求を発生させます。

(1) 16ビット・タイマ/カウンタを用いたプログラム例

16ビット・タイマ/カウンタによるインターバル・タイマは，1.3 μsの分解能で1.3 μsから87.4 ms（f_{CLK}＝6 MHz）までのインターバル時間を設定できます。ここでは，INTC01割り込み要求によるインターバル・タイマを生成するプログラム例を示します。

(a) 動作概要

INTC01インターバル・タイマの機能を図3-1ブロック図に示します。

図3-1　INTC01割り込み要求を発生させるインターバル・タイマ

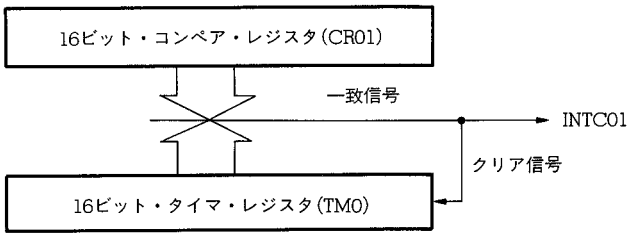
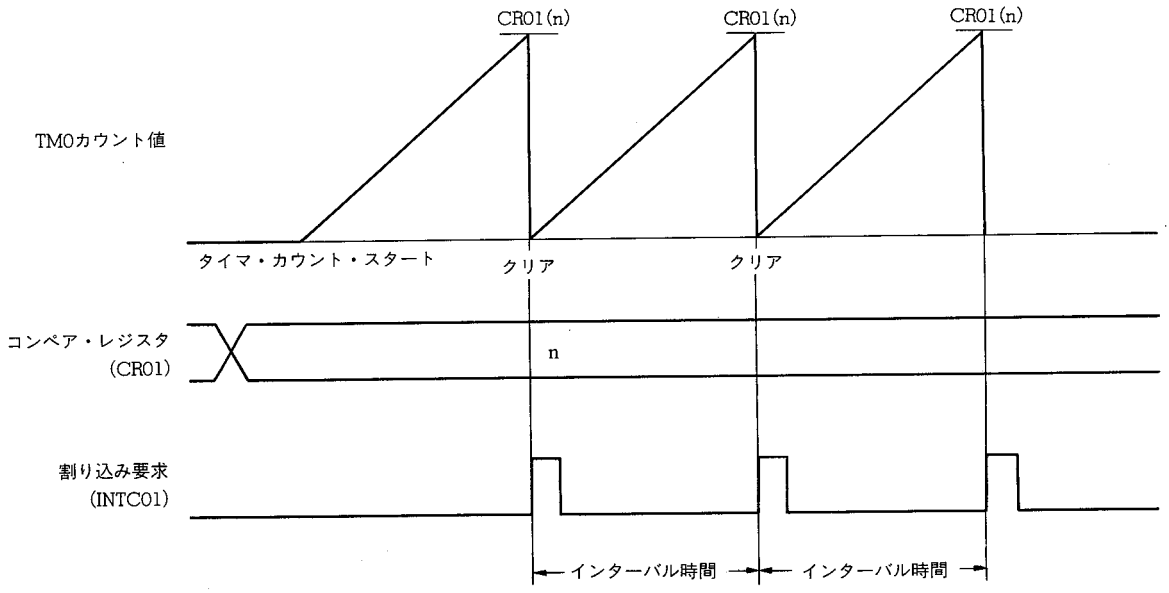


図3-2　INTC01割り込み要求を発生させるインターバル・タイマのタイミング・チャート



インターバル時間＝（n＋1）×8/f_{CLK} {n：0≦n≦FFFFH}

INTC01 割り込み要求は 16 ビット・タイマ・レジスタ（TM0）と 16 ビット・コンペア・レジスタ（CR01）の一致によって発生します。インターバル・タイマとして，INTC01 割り込み要求を発生させるためには，一致時に 16 ビット・タイマ・レジスタ（TM0）をクリアしてください。

また，16 ビット・タイマ/カウンタには 2 つの 16 ビット・コンペア・レジスタ（CR00, CR01）が存在しますが，タイマのクリア機能は CR01 のみにしかありません。したがって，任意のインターバル時間を設定するためには通常 CR01 にインターバル時間を設定します。

(b)　プログラム説明〔レーベル名称：IITVL1〕……**(h)　プログラム・リスト**参照

(i) 16 ビット・タイマ・レジスタ（TM0）と 16 ビット・コンペア・レジスタ（CR01）の一致による，16ビット・タイマ・レジスタ（TM0）のクリアを許可します。

(ii) 16 ビット・コンペア・レジスタ（CR01）にインターバル時間を設定します。

(iii) INTC01 割り込み要求のマスクを解除します。

(iv) 16 ビット・タイマ・レジスタ（TM0）のカウント動作を許可します。

(c)　モード・レジスタ設定例

タイマ・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
TMC0	CE	0	0	0	CE0	OVF0	0	0	（リセットで 00H）
設定例	0	0	0	0	1	0	0	0	

16 ビット・タイマ/カウンタ

OVF0	TM0 のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFFFH から 0000H へのカウント・アップ)

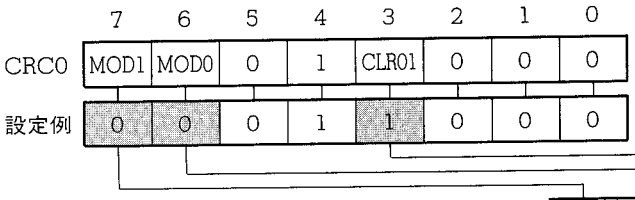
備考　このビットはソフトウエアでのみリセットされます。

CE0	TM0 のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

8 ビット・タイマ/カウンタ・ユニット 3

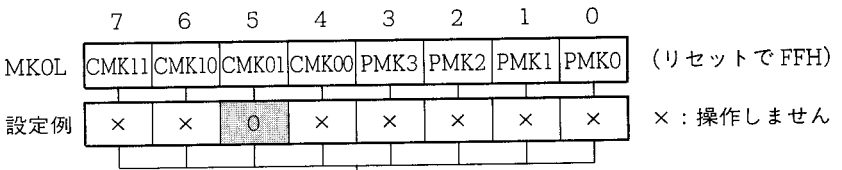
CE	TM3 のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

キャプチャ/コンペア・コントロール・レジスタ0



MOD1	MOD0	CLR01	タイマ出力モード指定 TO0	タイマ出力モード指定 TO1	TM0＝CR01時のTM0のクリア動作
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

割り込みマスク・レジスタL



MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理禁止

(d)　入出力パラメータ

INTVL1：インターバル時間を決定するパラメータをセットします。

16ビット・タイマ/カウンタのカウント・クロックは$f_{CLK}/8$固定ですので，INTVL1に設定された値に対するインターバル時間は次式で求められます。

インターバル時間＝INTVL1×8/f_{CLK}　{INTVL1：0≦INTVL1≦FFFFH}

(e)　使用レジスタ

なし

(f)　プログラム使用例

10 msごとにINTC01割り込み要求を発生させる場合の例を示します。ただし，f_{CLK}＝6 MHzとします。

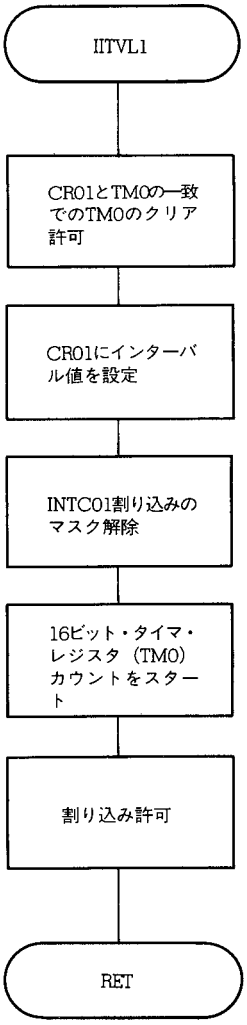
10 msごとにINTC01割り込み要求を発生させる場合，入力パラメータINTVL1の値は，次のようになります。

$$INTVL1 = \frac{10\times10^{-3}}{8/(6\times10^{6})} = 7500$$

したがって，外部定義，外部参照疑似命令を次のように定義してインターバル・タイマを作ります。

```
        PUBLIC  INTVL1          ; PARAMETER
        EXTRN   !ITVL1          ; PACKAGE
                .
                .
INTVL1  EQU     7500            ; PARAMETER FOR INTERVAL
                .
                .
        CALL    !!ITVL1
                .
                .
```

(g)　フロー・チャート


IITVL1
CR01とTM0の一致でのTM0のクリア許可
CR01にインターバル値を設定
INTC01割り込みのマスク解除
16ビット・タイマ・レジスタ（TM0）カウントをスタート
割り込み許可
RET

(h)　プログラム・リスト

```
        NAME    IITV1M
;
;*************************************************************
;* 16bit-Timer / Counter Unit                                *
;*      internal interval timer                              *
;*************************************************************
;
        PUBLIC  IITVL1
        EXTRN   INTVL1          ; Compare data for interval timer

CMK01   EQU     MK0L.5          ; INTC01 mask flag
;
        CSEG
IITVL1:
        MOV     CRC0,#00011000B ; clear enable TM0 by CR01
        MOVW    CR01,#INTVL1-1  ; set interval time
        CLR1    CMK01           ; open INTC01 mask
        MOV     TMC0,#00001000B ; timer start
        EI                      ; interrupt enable

        RET
;
        END
```

(2) 8ビット・タイマ/カウンタ2を用いたプログラム例

INTC21割り込み要求によるインターバル・タイマを生成するプログラム例です。

(a) 動作概要

INTC21インターバル・タイマの機能を図3-3のブロック図に示します。

図3-3　INTC21割り込み要求を発生させるインターバル・タイマ

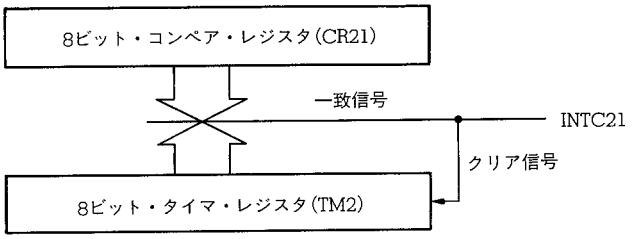
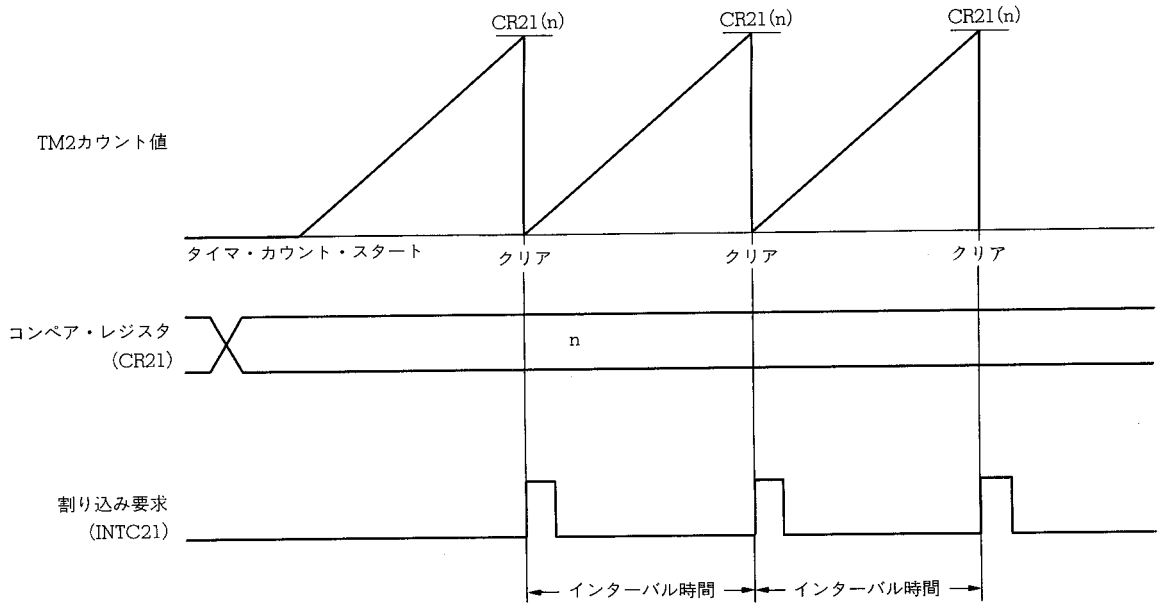


図3-4　INTC21割り込み要求を発生させるインターバル・タイマのタイミング・チャート



インターバル時間＝ (n + 1) ×X/f_{CLK} {n : 0 ≦ n ≦ FFH}

X = 16, 32, 64, 128, 256, 512

INTC21 割り込み要求は 8 ビット・タイマ・レジスタ(TM2)と 8 ビット・コンペア・レジスタ（CR21）の一致によって発生します。インターバル・タイマとして，INTC21 割り込み要求を発生させるためには，一致時に 8 ビット・タイマ・レジスタ（TM2）をクリアしてください。

(b)　プログラム説明〔レーベル名称：IITVL2〕……(h)　**プログラム・リスト**参照

(i) 8 ビット・タイマ/カウンタ 2 のカウント・クロックを $f_{CLK}/128$ に指定します。

(ii) 8 ビット・タイマ・レジスタ（TM2）と 8 ビット・コンペア・レジスタ（CR21）の一致による，8 ビット・タイマ・レジスタ（TM2）のクリアを許可します。

(iii) 8 ビット・コンペア・レジスタ（CR21）にインターバル時間を設定します。

(iv) INTC21 割り込み要求のマスクを解除します。

(v) 8 ビット・タイマ・レジスタ 2（TM2）のカウント動作を許可します。

(c)　モード・レジスタ設定例

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	(リセットで00H)
設定例	0	1	0	1	0	0	0	0	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタ1のカウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$注
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

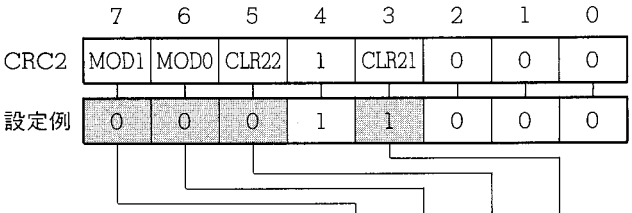
8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタ3のカウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

注　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

3
214
218A
224
234
244

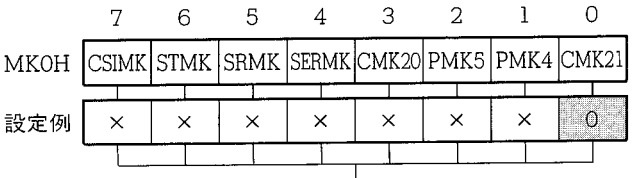
キャプチャ/コンペア・コントロール・レジスタ2



MOD1	MOD0	CLR22	CLR21	タイマ出力モード		TM2のクリア動作
				TO2	TO3	
0	0	0	0	トグル出力	トグル出力	クリアしない
0	0	0	1	トグル出力	トグル出力	TM2とCR21レジスタの一致でクリア
0	0	1	0	トグル出力	トグル出力	TM2の内容をCR22レジスタへキャプチャ後にクリア
0	0	1	1	トグル出力	トグル出力	TM2とCR21レジスタの一致またはTM2の内容をCR22レジスタへキャプチャ後にクリア
0	1	0	0	PWM出力	トグル出力	クリアしない
1	0	0	0	PWM出力	PWM出力	クリアしない
1	1	0	1	PPG出力	トグル出力	TM2とCR21レジスタの一致でクリア

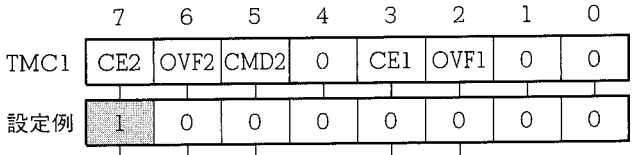
注意　上記以外の組み合わせは禁止

割り込みマスク・レジスタ H



MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理禁止

タイマ・コントロール・レジスタ 1



8ビット・タイマ/カウンタ 1

OVF1	TM1 のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFH から 00H へのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	TM1 のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ 2

CMD2	TM2 の動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	TM2 のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFH から 00H へのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	TM2 のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

3
214
218A
224
234
244

(d) 入出力パラメータ

INTVL2：インターバル時間を決定するパラメータをセットします。

8ビット・タイマ/カウンタのカウント・クロックは，$f_{CLK}/16$，$f_{CLK}/32$，$f_{CLK}/64$，$f_{CLK}/128$，$f_{CLK}/256$，$f_{CLK}/512$ および外部クロックが選択できます。このプログラムでは $f_{CLK}/128$ に固定しています。

INTVL2 に設定された値に対するインターバル時間は次式で求められます。

インターバル時間＝ INTVL2×X/f_{CLK} {INTVL2：0 ≦ INTVL2 ≦ FFH}

X = 16, 32, 64, 128, 256, 512

(e) 使用レジスタ

なし

(f) プログラム使用例

3.2 ms ごとに INTC21 割り込み要求を発生させる場合の例を示します。ただし，f_{CLK}＝ 6 MHz とします。

また，8ビット・タイマ/カウンタ2のカウント・クロックは$f_{CLK}/128$ とします。

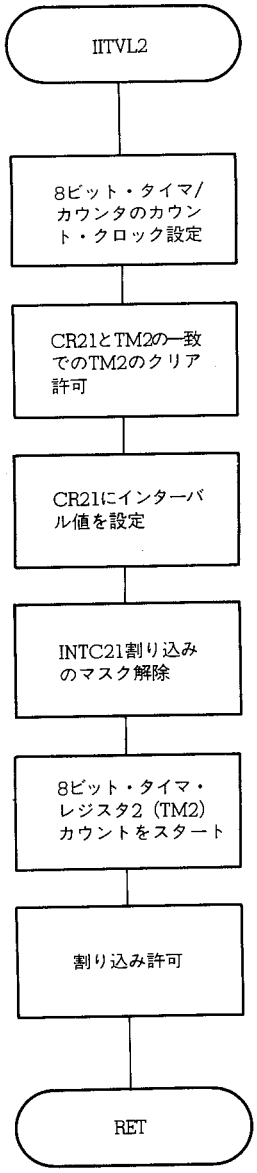
3.2 ms ごとに INTC21 割り込み要求を発生させる場合，入力パラメータ INTVL2 の値は，次のようになります。

$$INTVL2 = \frac{3.2\times10^{-3}}{128/(6\times10^{6})} = 150$$

したがって，外部定義，外部参照疑似命令を次のように定義してインターバル・タイマを作ります。

```
        PUBLIC  INTVL2          ; PARAMETER
        EXTRN   !!TVL2          ; PACKAGE
                .
                .
INTVL2  EQU     150             ; PARAMETER FOR INTERVAL
                .
                .
        CALL    !!!TVL2
                .
                .
```

(g)　フロー・チャート


IITVL2
8ビット・タイマ/カウンタのカウント・クロック設定
CR21とTM2の一致でのTM2のクリア許可
CR21にインターバル値を設定
INTC21割り込みのマスク解除
8ビット・タイマ・レジスタ2（TM2）カウントをスタート
割り込み許可
RET

(h)　プログラム・リスト

```
        NAME    IITV2M
;
;*************************************************************
;* 8bit- Timer / Counter Unit-2                              *
;*      internal interval timer                              *
;*************************************************************
;
        PUBLIC  IITVL2
        EXTRN   INTVL2          ; Compare data for interval timer

CMK21   EQU     MK0H.0          ; INTC21 mask flag
;
        CSEG
IITVL2:
        MOV     PRM1,#01010000B ; select fclk/128 (TM2)
        MOV     CRC2,#00011000B ; clear enable TM2 by CR21
        MOV     CR21,#LOW(INTVL2-1)     ; set interval time
        CLR1    CMK21           ; open INTC21 mask
        MOV     TMC1,#10000000B ; timer start
        EI                      ; interrupt enable

        RET
;
        END
```

3.2　プログラマブル矩形波出力

プログラマブル矩形波出力は，基本的にはインターバル・タイマの機能を用い，割り込み要求信号によって外部への出力信号を反転させます。したがって，デューティ比は50％です。

ここでは，16ビット・タイマ/カウンタを用いTO1端子から矩形波を出力する例を示します。

16ビット・タイマ/カウンタ・ユニットによる矩形波の出力は，2.6 μsの分解能で2.6 μsから174.8 ms（f_{CLK}＝6 MHz）までの矩形波の周期を設定できます。

(1) 動作概要

TO1端子からの矩形波出力の機能を図3-5のブロック図に示します。

図3-5　TO1端子からの矩形波出力

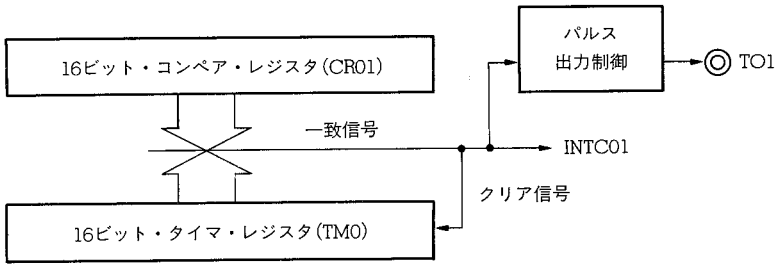
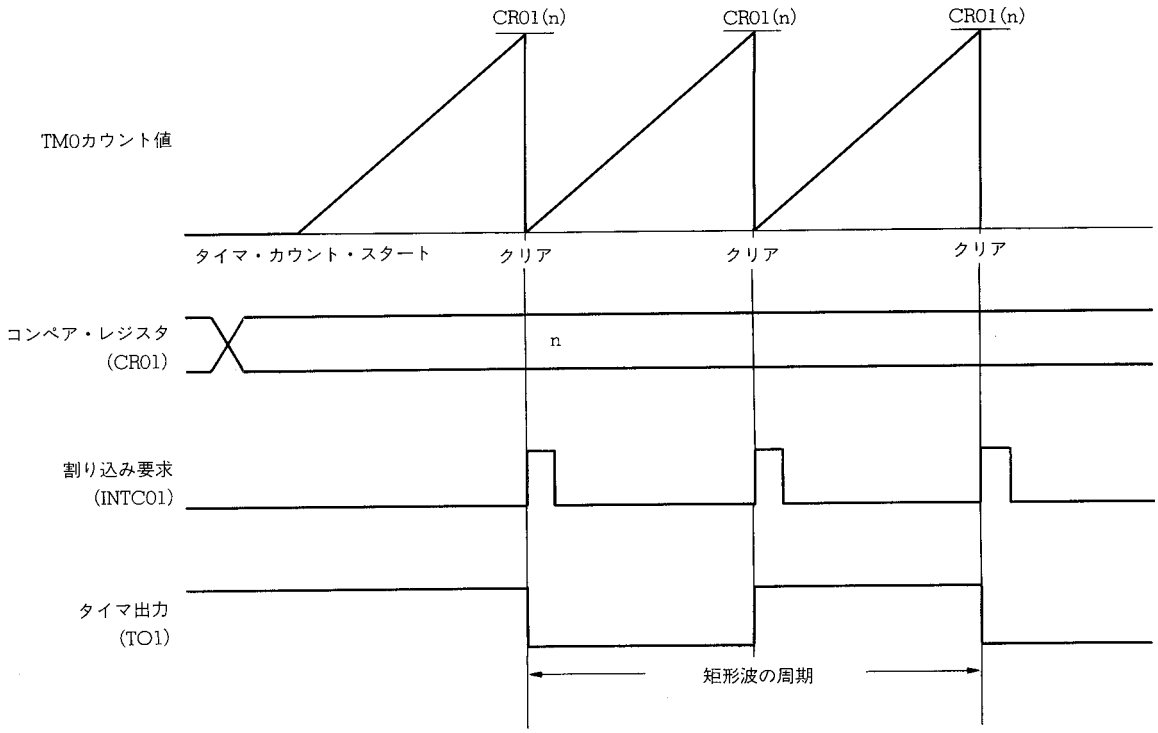


図3-6　TO1端子からの矩形波出力のタイミング・チャート



矩形波の周期＝（n＋1）×2×8/f_{CLK}　{n：0≦n≦FFFFH}

INTC01割り込み要求は16ビット・タイマ・レジスタ（TM0）と16ビット・コンペア・レジスタ（CR01）の一致によって発生します。インターバル・タイマとして，INTC01割り込み要求を発生させるためには，一致時に16ビット・タイマ・レジスタ（TM0）をクリアしてください。

タイマ出力はINTC01割り込み要求によって反転されるため，周期は**3.1.1 (1)**でとりあげたインターバル・タイマによる周期の2倍になります。

(2) プログラム説明［レーベル名称：TOUT］……(8) プログラム・リスト参照

(a) TO1タイマ出力のアクティブ・レベルをロウ・レベルに設定し，タイマ出力を許可します。

(b) P35をTO1出力端子として用いるためにコントロール・ポートに指定します。

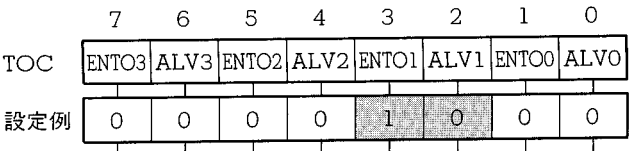
(c) 16ビット・タイマ・レジスタ（TM0）と16ビット・コンペア・レジスタ（CR01）の一致による，16ビット・タイマ・レジスタ（TM0）のクリアを許可します。

(d) 16ビット・コンペア・レジスタ（CR01）にインターバル周期を設定します。

(e) 16ビット・タイマ・レジスタ（TM0）のカウント動作を許可します。

(3) モード・レジスタ設定例

タイマ出力コントロール・レジスタ



	7	6	5	4	3	2	1	0	
TOC	ENTO3	ALV3	ENTO2	ALV2	ENTO1	ALV1	ENTO0	ALV0	(リセットで00H)
設定例	0	0	0	0	1	0	0	0	

ALV0	TO0端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENTO0	TO0端子の動作指定
0	$\overline{\text{ALV0}}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENTO1	TO1端子の動作指定
0	$\overline{\text{ALV1}}$を出力
1	パルス出力許可

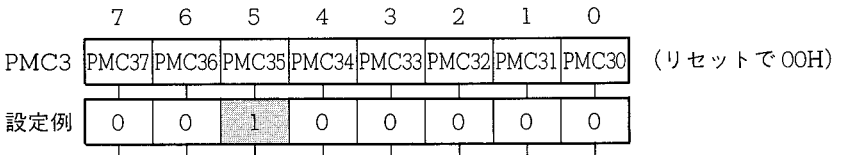
ALV2	TO2端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENTO2	TO2端子の動作指定
0	$\overline{\text{ALV2}}$を出力
1	パルス出力許可

ALV3	TO3端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENTO3	TO3端子の動作指定
0	$\overline{\text{ALV3}}$を出力
1	パルス出力許可

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SB0出力モード/SO出力モード

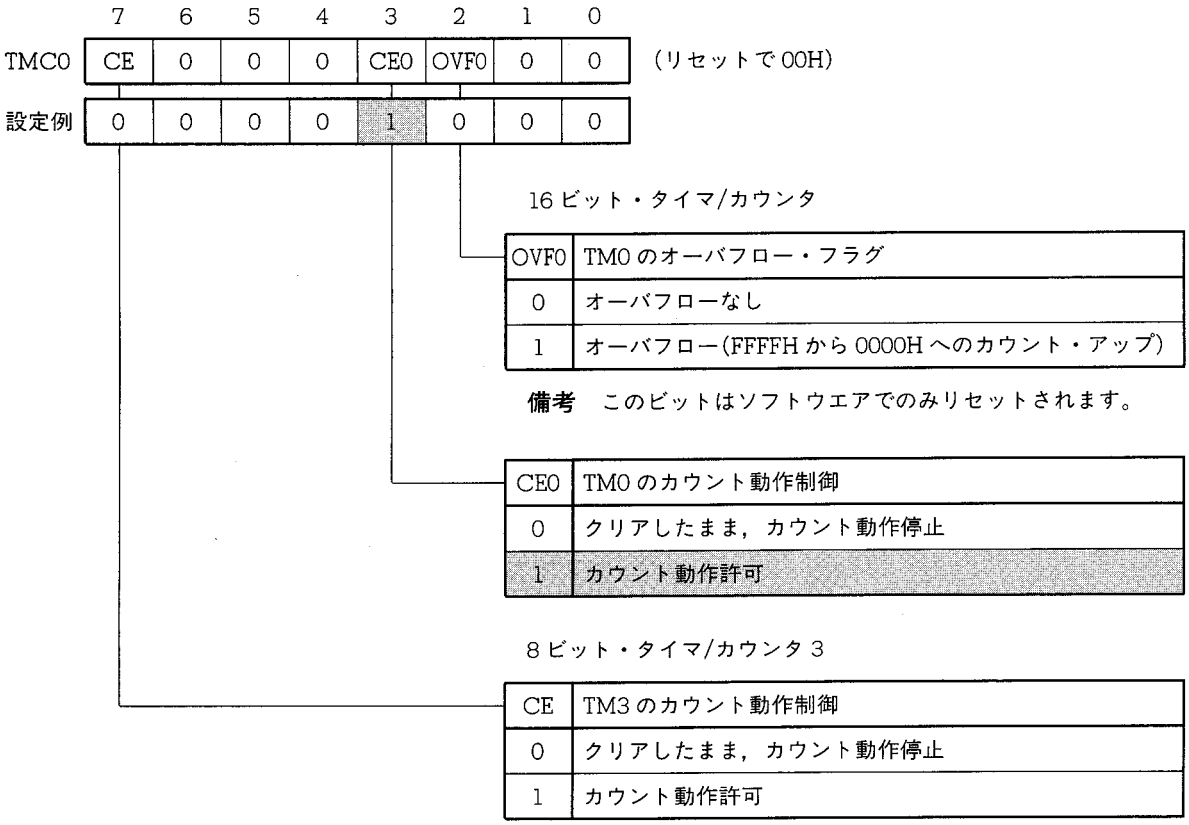
PMC3n	P3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

キャプチャ/コンペア・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
CRC0	MOD1	MOD0	0	1	CLR01	0	0	0	（リセットで10H）
設定例	0	0	0	1	1	0	0	0	

MOD1	MOD0	CLR01	タイマ出力モード指定		TM0＝CR01時のTM0のクリア動作
			TO0	TO1	
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

タイマ・コントロール・レジスタ0



備考　このビットはソフトウエアでのみリセットされます。

(4) 入出力パラメータ

INTVL3：矩形波の周期を決定するパラメータをセットします。

16ビット・タイマ/カウンタのカウント・クロックは$f_{CLK}/8$固定ですので，INTVL3に設定された値に対する矩形波の周期は次式で求められます。

矩形波の周期＝ INTVL3×2×8/f_{CLK}　{INTVL3：0 ≦ INTVL3 ≦ FFFFH}

(5) 使用レジスタ

なし

(6) プログラム使用例

TO1端子から周波数1 kHzの矩形波を出力する例を示します。

この場合，INTC01の発生タイミングは500 μs（2 kHz）です。f_{CLK}＝6 MHzとすると，入力パラメータINTVL3の値は，次のようになります。

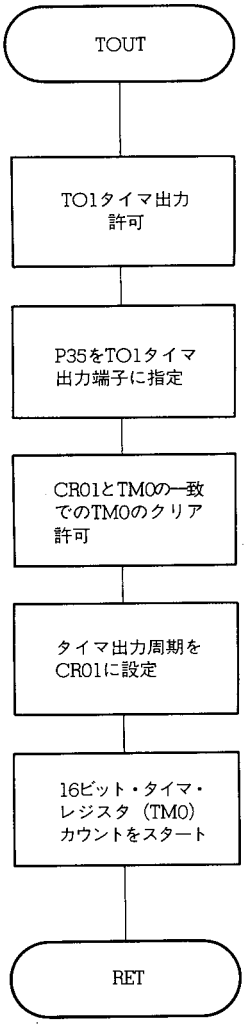
$$INTVL3 = \frac{500\times10^{-6}}{8/(6\times10^{6})} = 375$$

プログラム例として次のように用います。

```
        PUBLIC  INTVL3          ; PARAMETER
        EXTRN   TOUT            ; PACKAGE
                .
                .
INTVL3  EQU     375             ; PARAMETER FOR INTERVAL
                .
                .
        CALL    !TOUT
```

3
214
218A
224
234
244

(7) フロー・チャート


TOUT
TO1タイマ出力
許可
P35をTO1タイマ
出力端子に指定
CR01とTM0の一致
でのTM0のクリア
許可
タイマ出力周期を
CR01に設定
16ビット・タイマ・
レジスタ（TM0）
カウントをスタート
RET

(8) プログラム・リスト

```
        NAME    TOUTM
;
;*************************************************************
;* programable pulse output                                  *
;*************************************************************
;
        PUBLIC  TOUT
        EXTRN   INTVL3          ; timer output frequency
;
        CSEG
TOUT:
        MOV     TOC,#00001000B  ; enable TO1 timer output
        MOV     PMC3,#00100000B ; P35=control port
        MOV     CRC0,#00011000B ; clear enable TM0 by CR01
        MOVW    CR01,#INTVL3-1  ; set interval time
        MOV     TMC0,#00001000B ; timer start

        RET
;
        END
```

3.3　フリー・ランニング・インターバル・タイマ

フリー・ランニング・インターバル・タイマはタイマをフリー・ランニングさせ，割り込み要求処理中にコンペア・レジスタに一定の値を加算することにより，インターバルを作るものです。

ここでは，1つのタイマに対し，2本のコンペア・レジスタを使用して，2種類の周期のインターバルを作り，さらに，そのインターバルの周期によって2種類のタイマ出力を行う例を示します（タイマ出力のデューティは50％です）。

(1) 16ビット・タイマ/カウンタを用いたプログラム例

INTC00, INTC01割り込み要求を使用して，TO0, TO1端子から2種類のタイマ出力を行うプログラム例です。

(a) 動作概要

TO0, TO1端子からのタイマ出力の機能を図3-7のブロック図に示します。

図3-7　TO0, TO1端子からのタイマ出力

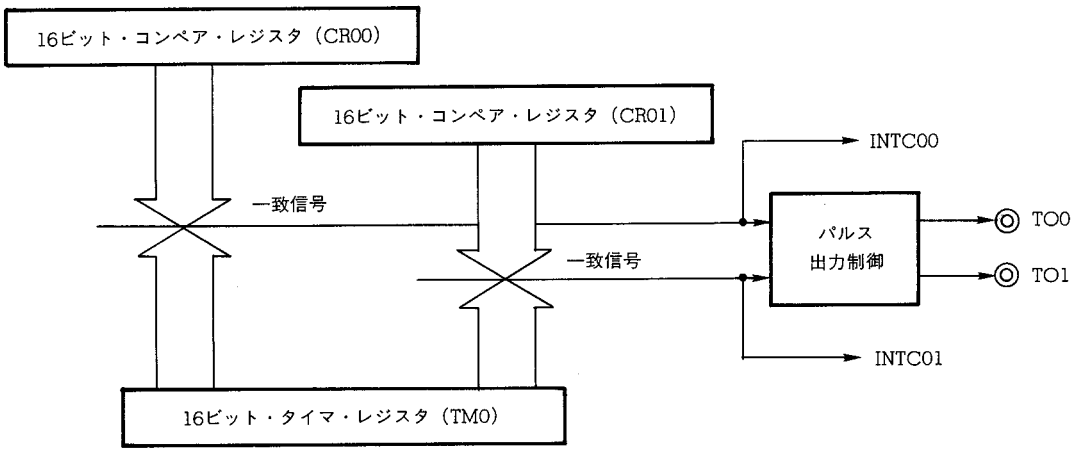
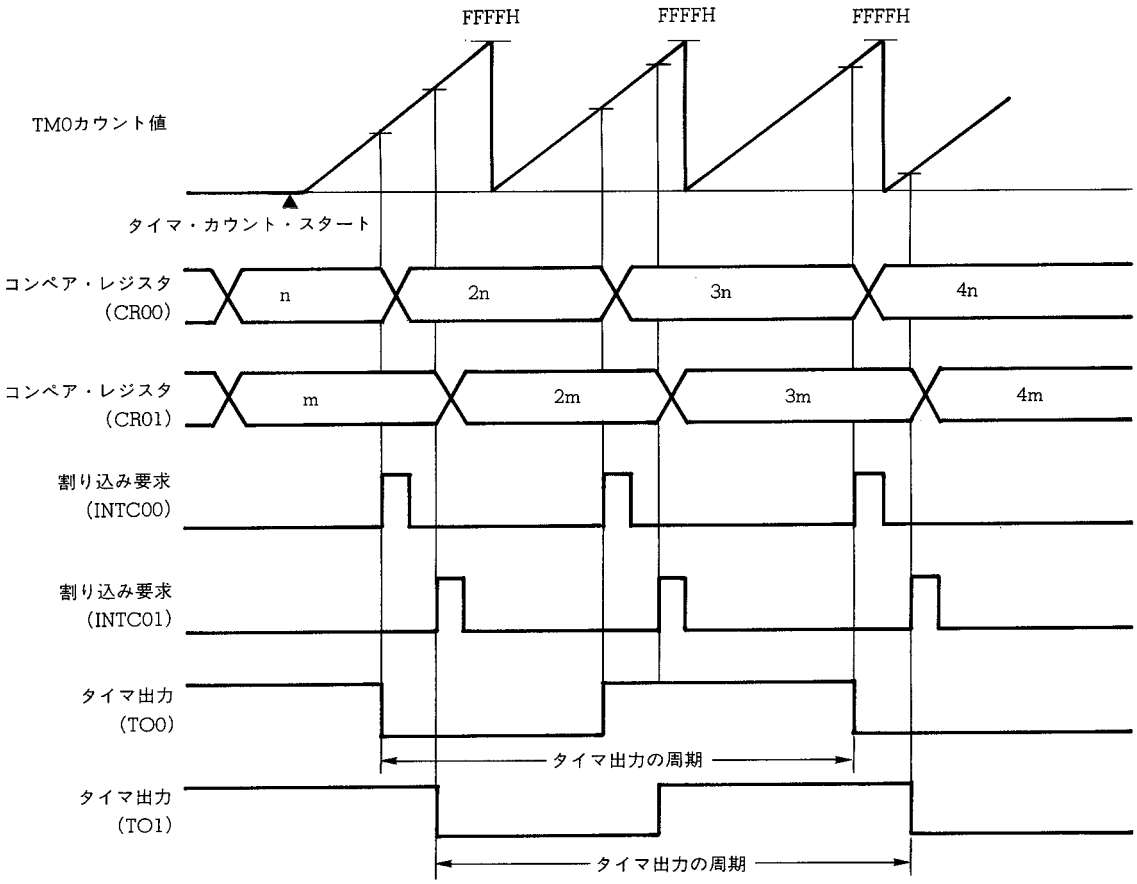


図3-8　TO0，TO1端子からのタイマ出力のタイミング・チャート



タイマ出力の周期＝ $n(m) \times 2 \times 8/f_{CLK}$ $\{n(m) : 1 \leqq n(m) \leqq FFFFH\}$

2種類の周期のインターバルを作るには，まずタイマをフリー・ランニングさせてください。その後，INTC00，INTC01割り込み要求発生時に，現在のコンペア・レジスタ（CR00，CR01）の値にそれぞれのインターバル周期分の値を加算することによって，一定周期の2つのタイマ出力を行うことができます。

(b)　プログラム説明……(h)　プログラム・リスト参照

(i) イニシャライズ処理［レーベル名称：FRUNO］

① TO0, TO1 タイマ出力のアクティブ・レベルをロウ・レベルに設定し，タイマ出力を許可します。

② P34, P35 を TO0, TO1 出力端子として用いるためにコントロール・ポートに指定します。

③ 16 ビット・タイマ・レジスタ（TM0）と 16 ビット・コンペア・レジスタ（CR00, CR01）の一致による，16 ビット・タイマ・レジスタ（TM0）のクリアを禁止します（フリー・ランニング・モード）。

④ 16 ビット・コンペア・レジスタ（CR00, CR01）にそれぞれのインターバル周期を設定します。

⑤ INTC00, INTC01 割り込み要求のマスクを解除します。

⑥ 16 ビット・タイマ/カウンタのカウント動作を許可します。

(ii) INTC00 割り込み処理［レーベル名称：INTC00］

① レジスタ・バンク 1 に切り替えます。

② 現在のコンペア・レジスタ（CR00）値にインターバル周期分の値を加算します。加算値のオーバフローは無視します。

③ 加算値をコンペア・レジスタ（CR00）に設定します。

(iii) INTC01 割り込み処理［レーベル名称：INTC01］

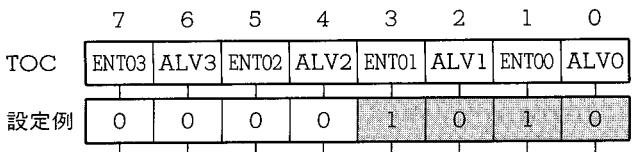
① レジスタ・バンク 1 に切り替えます。

② 現在のコンペア・レジスタ（CR01）値にインターバル周期分の値を加算します。加算値のオーバフローは無視します。

③ 加算値をコンペア・レジスタ（CR01）に設定します。

(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



ALV0	TO0端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT00	TO0端子の動作指定
0	$\overline{ALV0}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT01	TO1端子の動作指定
0	$\overline{ALV1}$を出力
1	パルス出力許可

ALV2	TO2端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT02	TO2端子の動作指定
0	$\overline{ALV2}$を出力
1	パルス出力許可

ALV3	TO3端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT03	TO3端子の動作指定
0	$\overline{ALV3}$を出力
1	パルス出力許可

ポート3モード・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	（リセットで00H）
設定例	0	0	1	1	0	0	0	0	

PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

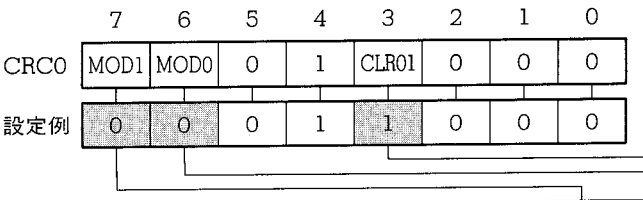
PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{\text{SCK}}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

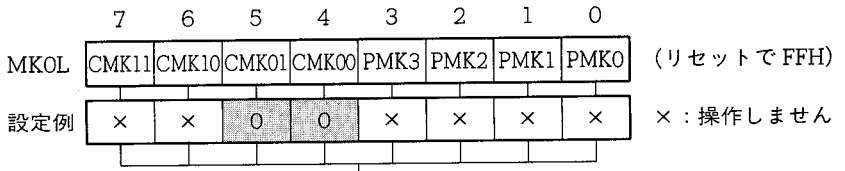
PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

キャプチャ/コンペア・コントロール・レジスタ0



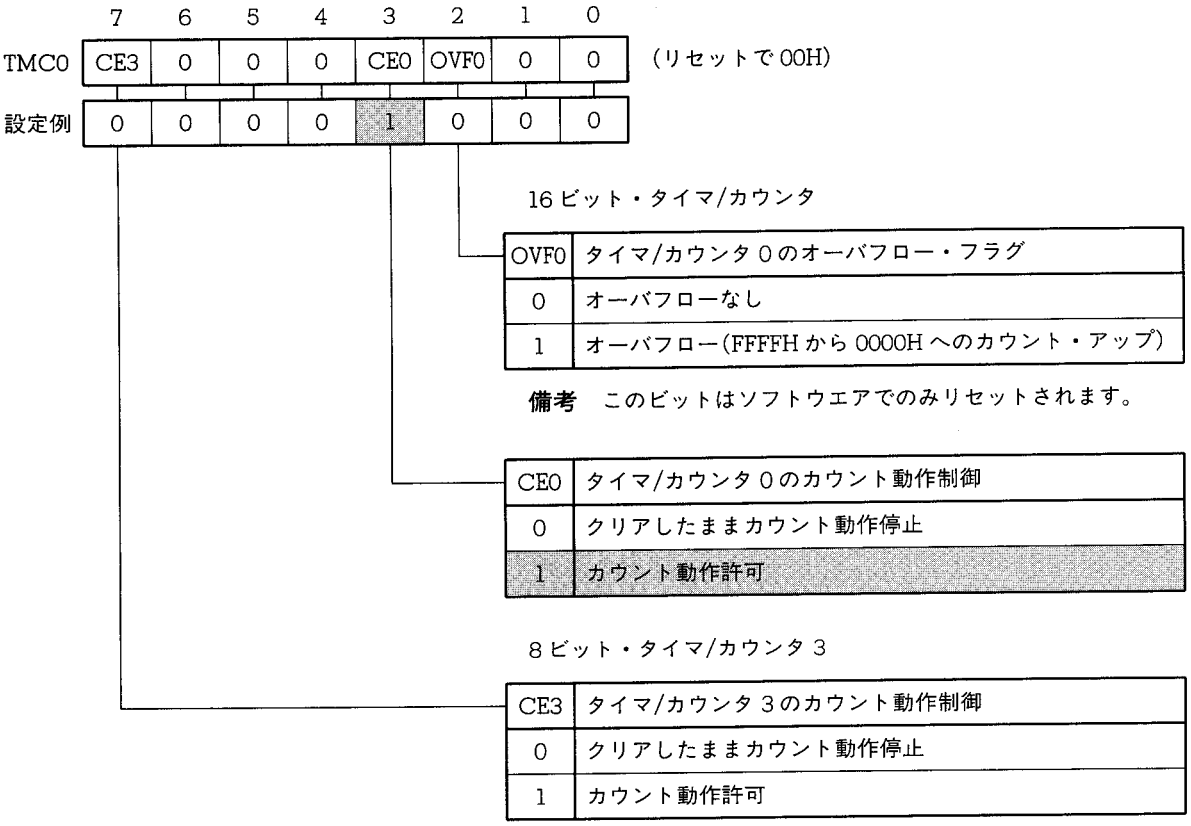
MOD1	MOD0	CLR01	タイマ出力モード指定		TM0＝CR01時のTM0のクリア動作
			TO0	TO1	
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

割り込みマスク・レジスタL



MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

タイマ・コントロール・レジスタ0



備考　このビットはソフトウエアでのみリセットされます。

(d)　入力パラメータ

INTVL4：　TO0端子からのタイマ出力の周期を決定する値です。1回割り込み要求が発生するごとに，割り込み処理で現在のコンペア・レジスタ（CR00）の値にこの値を加算します。

INTVL5：　TO1端子からのタイマ出力の周期を決定する値です。1回割り込み要求が発生するごとに，割り込み処理で現在のコンペア・レジスタ（CR01）の値にこの値を加算します。

16ビット・タイマ/カウンタのカウント・クロックは$f_{CLK}/8$固定ですので，INTVL4，INTVL5に設定された値に対するタイマ出力の周期は次式で求められます。

タイマ出力の周期＝INTVL4（INTVL5）×2×8/f_{CLK}
{INTVL4（INTVL5）：0≦INTVL4（INTVL5）≦FFFFH}

(e)　使用レジスタ

AX（レジスタ・バンク1）

(f)　プログラム使用例

TO0端子から周波数500 Hz，TO1端子から周波数1 kHzのタイマ出力を行う例を示します。この場合，INTC00を1 msごとに，INTC01を500 μsごとにそれぞれ発生させます。

f_{CLK}＝6 MHzとすると，入力パラメータINTVL4，INTVL5の値は，次のようになります。

$$INTVL4 = \frac{1\times10^{-3}}{8/(6\times10^{6})} = 750$$

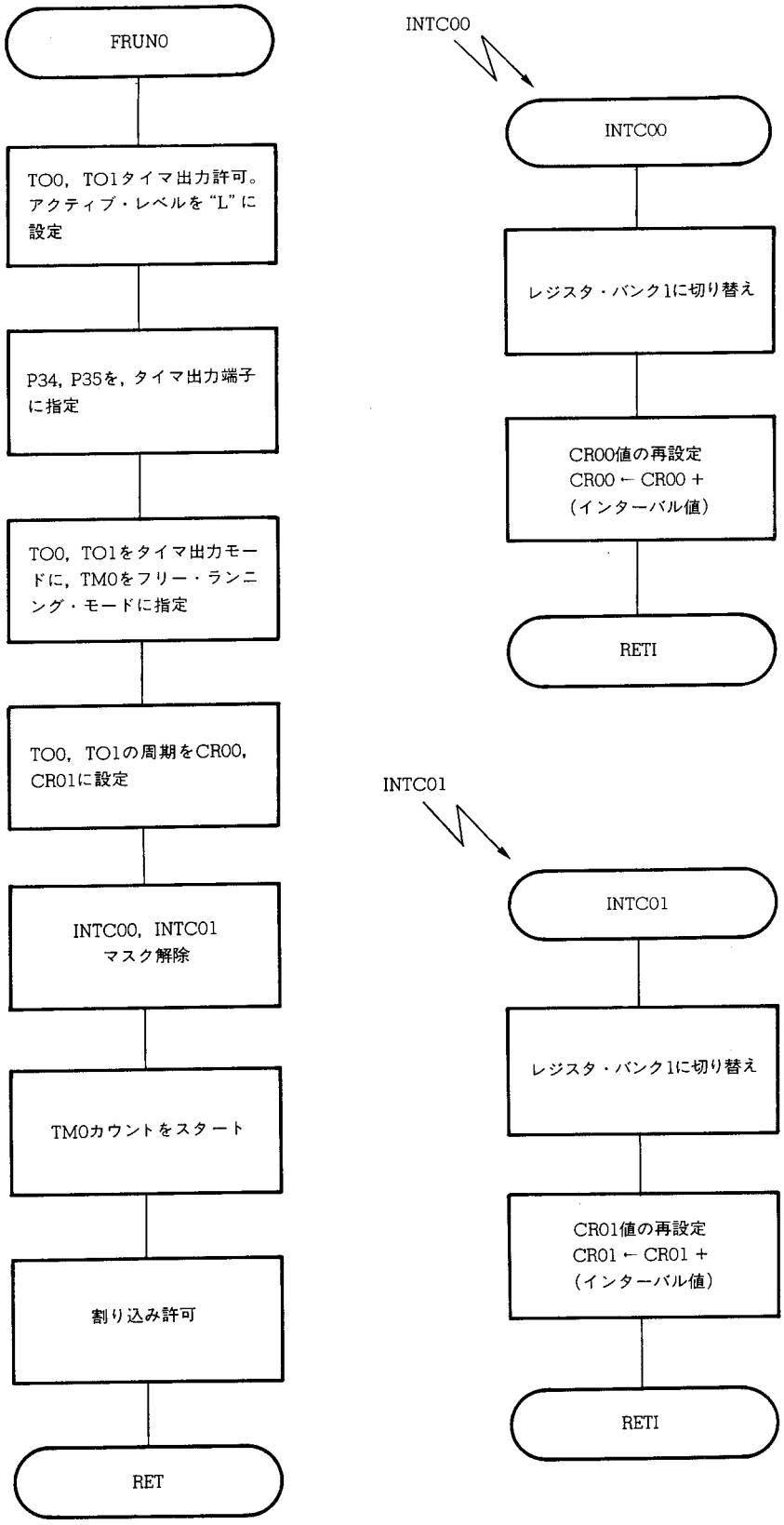
$$INTVL5 = \frac{500\times10^{-6}}{8/(6\times10^{6})} = 375$$

3
214
218A
224
234
244

プログラム例として次のように用います。

```
        PUBLIC  INTVL4,INTVL5   ; PARAMETER
        EXTRN   FRUNO           ; PACKAGE
                .
                .
INTVL4  EQU     750             ; free running timer INTC00 compare data
INTVL5  EQU     375             ; free running timer INTC01 compare data
                .
                .
        CALL    !FRUNO
                .
                .
```

(g)　フロー・チャート


FRUN0
TO0，TO1タイマ出力許可。アクティブ・レベルを“L”に設定
P34，P35を，タイマ出力端子に指定
TO0，TO1をタイマ出力モードに，TM0をフリー・ランニング・モードに指定
TO0，TO1の周期をCR00，CR01に設定
INTC00，INTC01マスク解除
TM0カウントをスタート
割り込み許可
RET
INTC00
INTC00
レジスタ・バンク1に切り替え
CR00値の再設定
CR00 ← CR00 +
（インターバル値）
RETI
INTC01
INTC01
レジスタ・バンク1に切り替え
CR01値の再設定
CR01 ← CR01 +
（インターバル値）
RETI

(h)　プログラム・リスト

```
        NAME    F_RUNO
;
;*************************************************************
;* 16bit-Timer / Counter                                     *
;*      free running interval timer                          *
;*************************************************************
;
        PUBLIC  FRUNO
        EXTRN   INTVL4,INTVL5

CMK00   EQU     MK0L.4          ; INTC00 mask flag
CMK01   EQU     MK0L.5          ; INTC01 mask flag
;
INTC00VT CSEG   AT 00014H
        DW      INTC00          ; INTC00
INTC01VT CSEG   AT 00016H
        DW      INTC01          ; INTC01
;
        CSEG
FRUNO:
        MOV     TOC,#00001010B  ; timer output,active level low
        MOV     PMC3,#00110000B ; P3 control port
        MOV     CRC0,#00010000B ; set timer free running mode
        MOVW    CR00,#INTVL4    ; set interval time
        MOVW    CR01,#INTVL5    ; set interval time
        CLR1    CMK00           ; open INTC00 mask
        CLR1    CMK01           ; open INTC01 mask
        MOV     TMC0,#00001000B ; start timer
        EI                      ; interrupt enable

        RET
;
;*************************************************************
;*              INTC00 interrupt routine                     *
;*************************************************************
INTC00:
        SEL     RB1             ; register bank change
        MOVW    AX,CR00
        ADDW    AX,#INTVL4      ; count INTC00 compare data
        MOVW    CR00,AX
        RETI
;
;*************************************************************
;*              INTC01 interrupt routine                     *
;*************************************************************
INTC01:
        SEL     RB1             ; register bank change
        MOVW    AX,CR01
        ADDW    AX,#INTVL5      ; count INTC01 compare data
        MOVW    CR01,AX
        RETI
;
        END
```

(2) 8ビット・タイマ/カウンタ2を用いたプログラム例

INTC20, INTC21割り込みソースを使用して，TO2, TO3端子から2種類のタイマ出力を行うプログラム例です。

(a) 動作概要

TO2, TO3端子からのタイマ出力の機能を図3-9のブロック図に示します。

図3-9　TO2, TO3端子からのタイマ出力

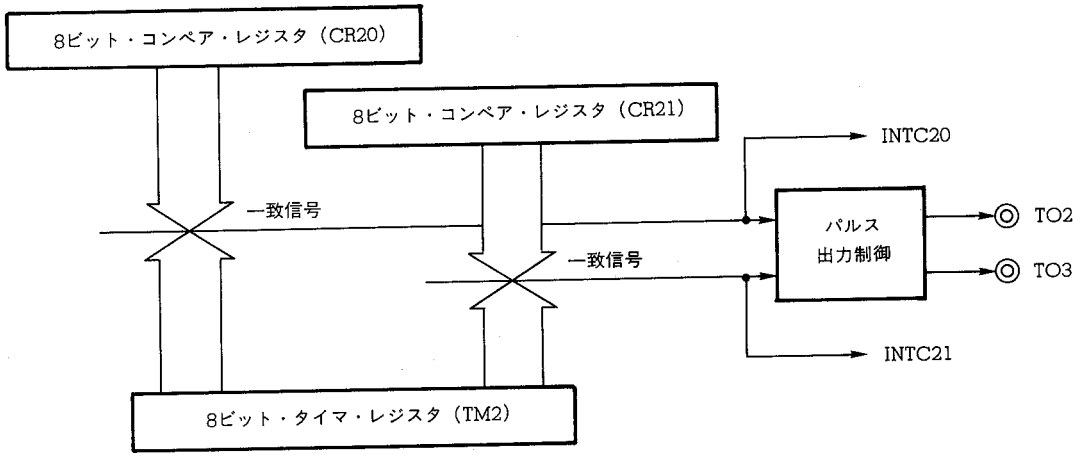
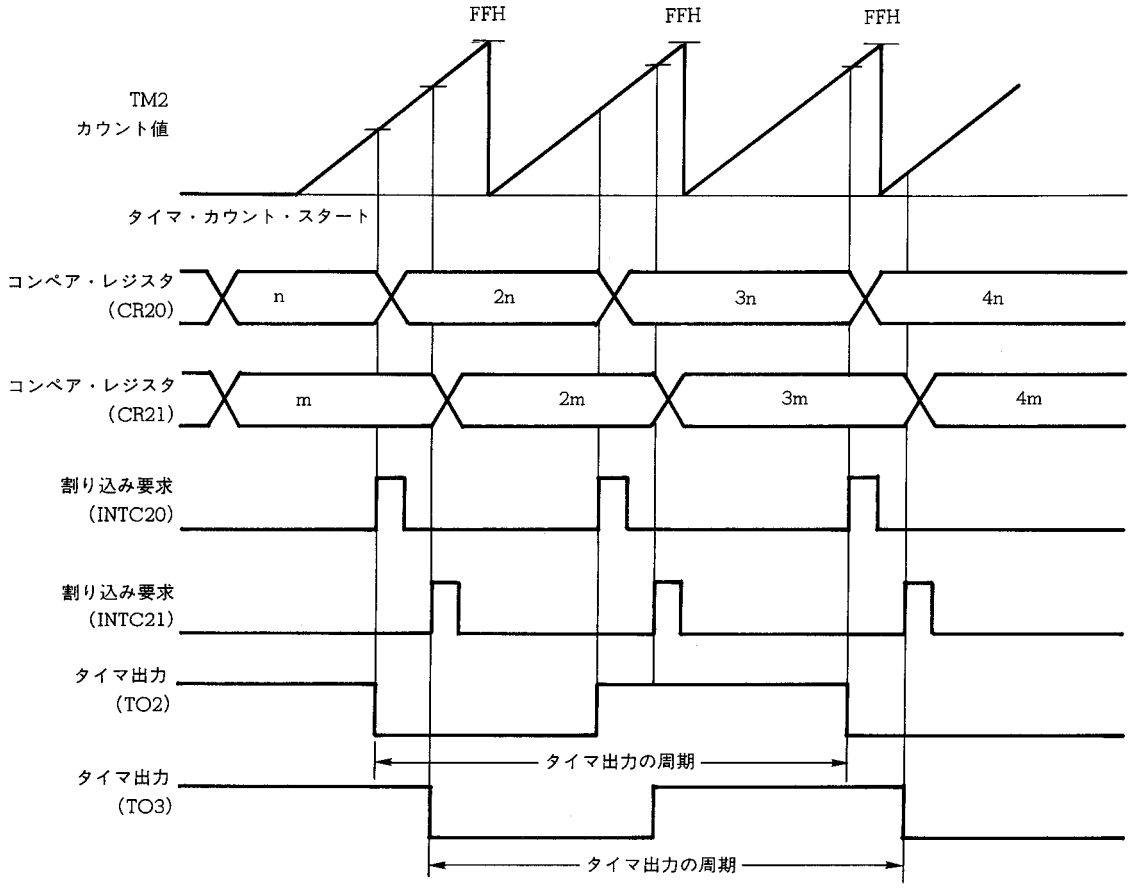


図3-10　TO2，TO3端子からのタイマ出力のタイミング・チャート



タイマ出力の周期＝ n (m) × 2 ×X/f_{CLK} {n (m) : 1 ≦ n (m) ≦ FFH}

X＝16, 32, 64, 128, 256, 512

2種類の周期のインターバルを作るには，まずタイマをフリー・ランニングさせてください。

INTC20, INTC21割り込み要求発生時に，現在のコンペア・レジスタ（CR20, CR21）の値にそれぞれのインターバル周期分の値を加算することによって，一定周期の2つのタイマ出力を行うことができます。

(b)　プログラム説明……(h)　プログラム・リスト参照

(i) イニシャライズ処理［レーベル名称：FRUN2］

① TO2, TO3タイマ出力のアクティブ・レベルをロウ・レベルに設定し，タイマ出力を許可します。

② P36, P37をTO2, TO3出力端子として用いるためにコントロール・ポートに指定します。

③ 8ビット・タイマ・レジスタ(TM2)と8ビット・コンペア・レジスタ(CR20, CR21)の一致による，8ビット・タイマ・レジスタ（TM2）のクリアを禁止します（フリー・ランニング・モード)。

④ 8ビット・タイマ/カウンタ2のカウント・クロックを$f_{CLK}/16$に設定します。

⑤ 8ビット・コンペア・レジスタ（CR20, CR21）にそれぞれのインターバル周期を設定します。

⑥ INTC20, INTC21割り込み要求のマスクを解除します。

⑦ 8ビット・タイマ/カウンタ2のカウント動作を許可します。

(ii) INTC20割り込み処理［レーベル名称：INTC20］

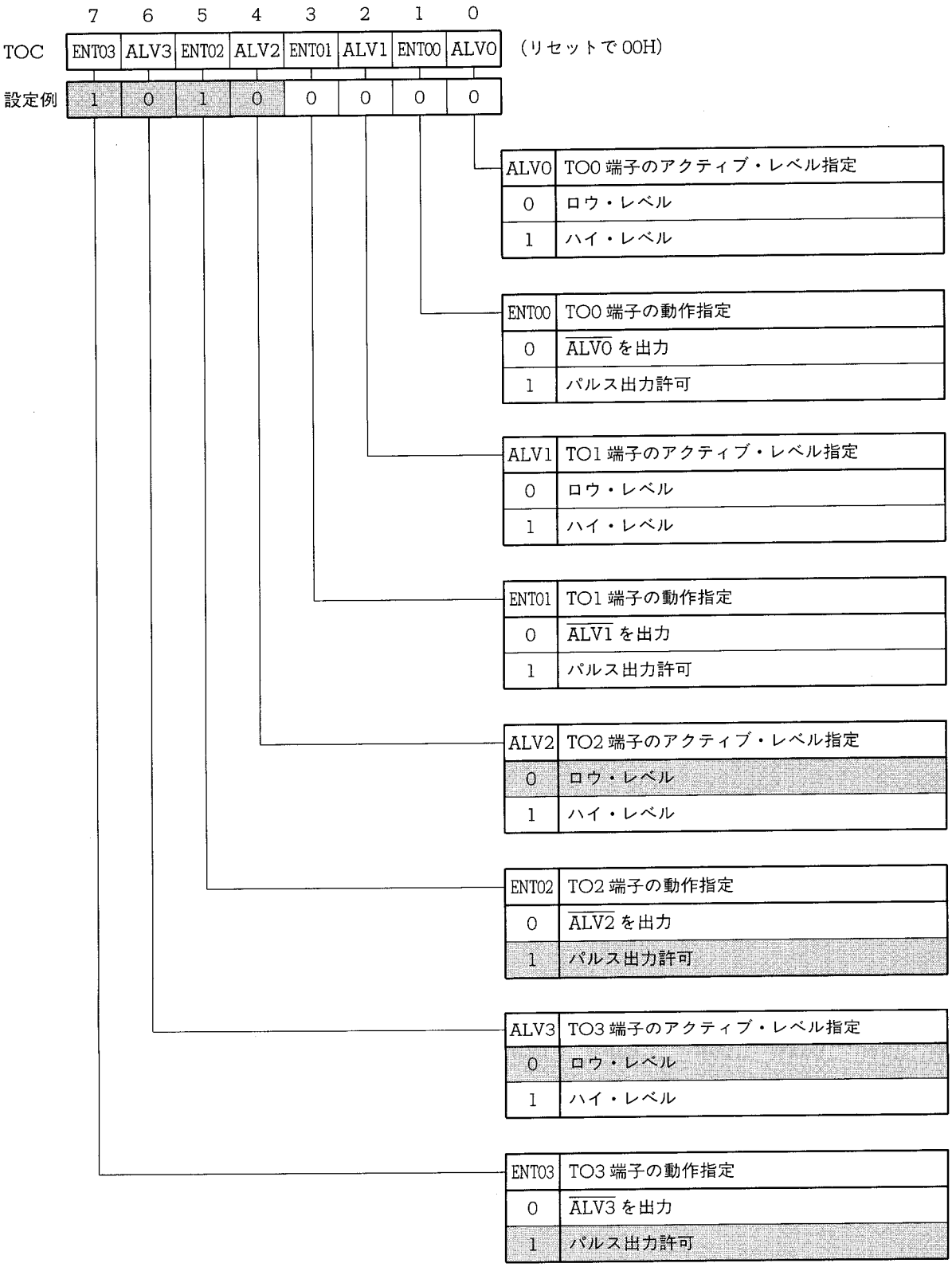
① コンペア・レジスタ（CR20）にインターバル周期分の値を加算します。
加算値のオーバフローは無視します。

(iii) INTC21割り込み処理［レーベル名称：INTC21］

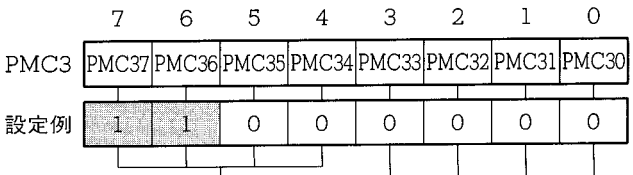
① コンペア・レジスタ（CR21）にインターバル周期分の値を加算します。
加算値のオーバフローは無視します。

(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

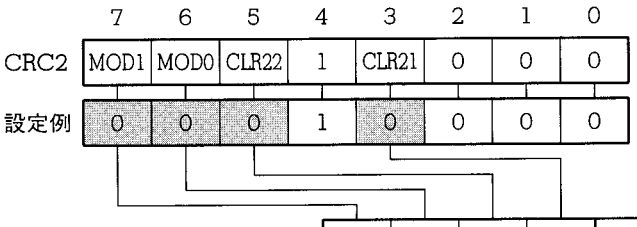
PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	PM3nの端子コントロール・モード指定(n＝4−7)
0	入出力ポート・モード
1	TOm出力モード(m＝0−3)

キャプチャ/コンペア・コントロール・レジスタ 2



MOD1	MOD0	CLR22	CLR21	タイマ出力モード		TM2 のクリア動作
				TO2	TO3	
0	0	0	0	トグル出力	トグル出力	クリアしない
0	0	0	1	トグル出力	トグル出力	TM2 と CR21 レジスタの一致でクリア
0	0	1	0	トグル出力	トグル出力	TM2 の内容を CR22 レジスタへキャプチャ後にクリア
0	0	1	1	トグル出力	トグル出力	TM2 と CR21 レジスタの一致または TM2 の内容を CR22 レジスタへキャプチャ後にクリア
0	1	0	0	PWM 出力	トグル出力	クリアしない
1	0	0	0	PWM 出力	PWM 出力	クリアしない
1	1	0	1	PPG 出力	トグル出力	TM2 と CR21 レジスタの一致でクリア

注意　上記以外の組み合わせは禁止

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	（リセットで00H）
設定例	0	0	0	0	0	0	0	0	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタ1のカウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$注
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタ3のカウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

注　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

割り込みマスク・レジスタ

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	（リセットでFFH）
設定例	×	×	×	×	0	×	×	0	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

タイマ・コントロール・レジスタ1

	7	6	5	4	3	2	1	0	
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0	(リセットで00H)
設定例	1	0	0	0	0	0	0	0	

8ビット・タイマ/カウンタ1

OVF1	タイマ/カウンタ1のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	タイマ/カウンタ1のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ2

CMD2	タイマ/カウンタ2の動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	タイマ/カウンタ2のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	タイマ/カウンタ2のカウント動作制御
0	クリアしたままカウンタ動作停止
1	カウント動作許可

(d)　入力パラメータ

(i) INTVL6：　TO2端子からのタイマ出力の周期を決定する値です。1回割り込み要求が発生するごとに，割り込み処理で現在のコンペア・レジスタ（CR20）の値にこの値を加算します。

(ii) INTVL7：　TO3端子からのタイマ出力の周期を決定する値です。1回割り込み要求が発生するごとに，割り込み処理で現在のコンペア・レジスタ（CR21）の値にこの値を加算します。

8ビット・タイマ/カウンタ2のカウント・クロックは$f_{CLK}/16$，$f_{CLK}/32$，$f_{CLK}/64$，$f_{CLK}/128$，$f_{CLK}/256$，$f_{CLK}/512$および外部クロックが選択できます。

このプログラムでは，$f_{CLK}/16$を選択しています。INTVL6，INTVL7に設定された値に対するタイマ出力の周期は次式で求められます。

タイマ出力の周期＝ INTVL6（INTVL7）×2×16/f_{CLK}

{INTVL6（INTVL7）：0 ≦ INTVL6（INTVL7）≦ FFH}

(e)　使用レジスタ

なし

(f)　プログラム使用例

TO2端子から周波数1.5 kHz，TO3端子から周波数2.5 kHzのタイマ出力を行う例を示します。この場合，INTC20を333 μsごとに，INTC21を200 μsごとにそれぞれ発生させます。

f_{CLK}＝6 MHzとし，カウント・クロックを$f_{CLK}/16$に設定すると，入力パラメータINTVL6，INTVL7の値は，次のようになります。

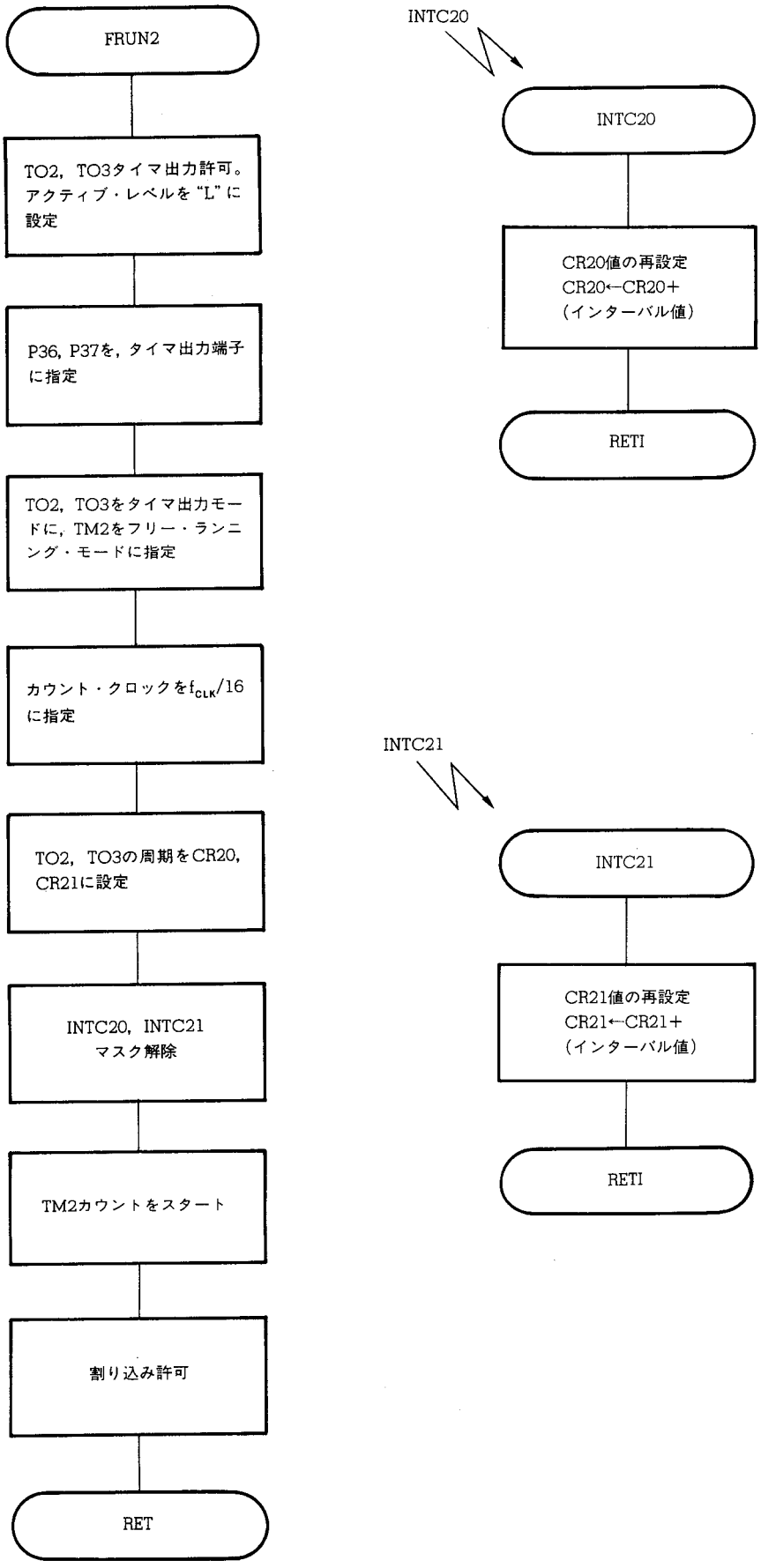
$$INTVL6 = \frac{333\times10^{-6}}{16/(6\times10^{6})} = 125$$

$$INTVL7 = \frac{200\times10^{-6}}{16/(6\times10^{6})} = 75$$

プログラム例として次のように用います。

```
        PUBLIC  INTVL6,INTVL7   ; PARAMETER
        EXTRN   FRUN2           ; PACKAGE
                .
                .
INTVL6  EQU     125             ; free running timer INTC20 compare data
INTVL7  EQU     75              ; free running timer INTC21 compare data
                .
                .
        CALL    !FRUN2
                .
                .
```

(g)　フロー・チャート


FRUN2
TO2，TO3タイマ出力許可。アクティブ・レベルを“L”に設定
P36，P37を，タイマ出力端子に指定
TO2，TO3をタイマ出力モードに，TM2をフリー・ランニング・モードに指定
カウント・クロックをf_{CLK}/16に指定
TO2，TO3の周期をCR20，CR21に設定
INTC20，INTC21マスク解除
TM2カウントをスタート
割り込み許可
RET
INTC20
INTC20
CR20値の再設定
CR20←CR20＋（インターバル値）
RETI
INTC21
INTC21
CR21値の再設定
CR21←CR21＋（インターバル値）
RETI

(h)　プログラム・リスト

```
        NAME    F_RUN2
;
;*************************************************************
;* 8bit-Timer / Counter_2                                    *
;*      free running interval timer                          *
;*************************************************************
;
        PUBLIC  FRUN2
        EXTRN   INTVL6,INTVL7

CMK20   EQU     MK0H.3          ; INTC20 mask flag
CMK21   EQU     MK0H.0          ; INTC21 mask flag
;
INTC20VT CSEG   AT 00012H
        DW      INTC20          ; INTC20
INTC21VT CSEG   AT 0001CH
        DW      INTC21          ; INTC21
;
        CSEG
FRUN2:
        MOV     TOC,#10100000B  ; timer output,active level low
        MOV     PMC3,#11000000B ; P3 control port
        MOV     CRC2,#00010000B ; timer free running mode
        MOV     PRM1,#00000000B ; set prescaler fclk/16
        MOV     CR20,#LOW(INTVL6)       ; set interval time
        MOV     CR21,#LOW(INTVL7)       ; set interval time
        CLR1    CMK20           ; open INTC20 mask
        CLR1    CMK21           ; open INTC21 mask
        MOV     TMC1,#10000000B ; start timer
        EI                      ; interrupt enable
        RET
;
;*************************************************************
;*              INTC20 interrupt routine                     *
;*************************************************************
INTC20:
        ADD     CR20,#LOW(INTVL6)  ; count INTC20 compare data
        RETI
;
;*************************************************************
;*              INTC21 interrupt routine                     *
;*************************************************************
INTC21:
        ADD     CR21,#LOW(INTVL7)  ; count INTC21 compare data
        RETI
;
        END
```

3.4　PWM/PPG 出力

PWM/PPG 出力は，μPD78214 シリーズ，78218A シリーズ，78234 シリーズ，78244 シリーズが備えている機能で，タイマ割り込み要求を用いた可変デューティの矩形波出力です。

PWM 出力ではタイマがフルカウントする期間が 1 周期となり，PPG 出力では，1 つのタイマから発生する 2 系統の一致信号のうちの一方によって 1 周期の幅が決定されます。

(1) 16 ビット・タイマ/カウンタを用いた PWM 出力プログラム例

TO0 端子から，INTC00 割り込み要求によってパルス幅を決定する PWM 波形を出力するプログラム例を示します。

デューティを変更する場合は，RAM 中のワーク・エリアにデューティを決定する値をセットして，デューティ変更のサブルーチンを呼び出します。

(a)　動作概要

TO0 端子からの PWM 出力の機能を図 3 - 11 のブロック図に示します。

図 3 - 11　TO0 端子からの PWM 出力

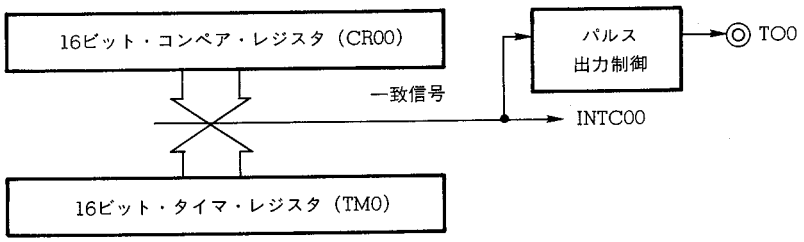
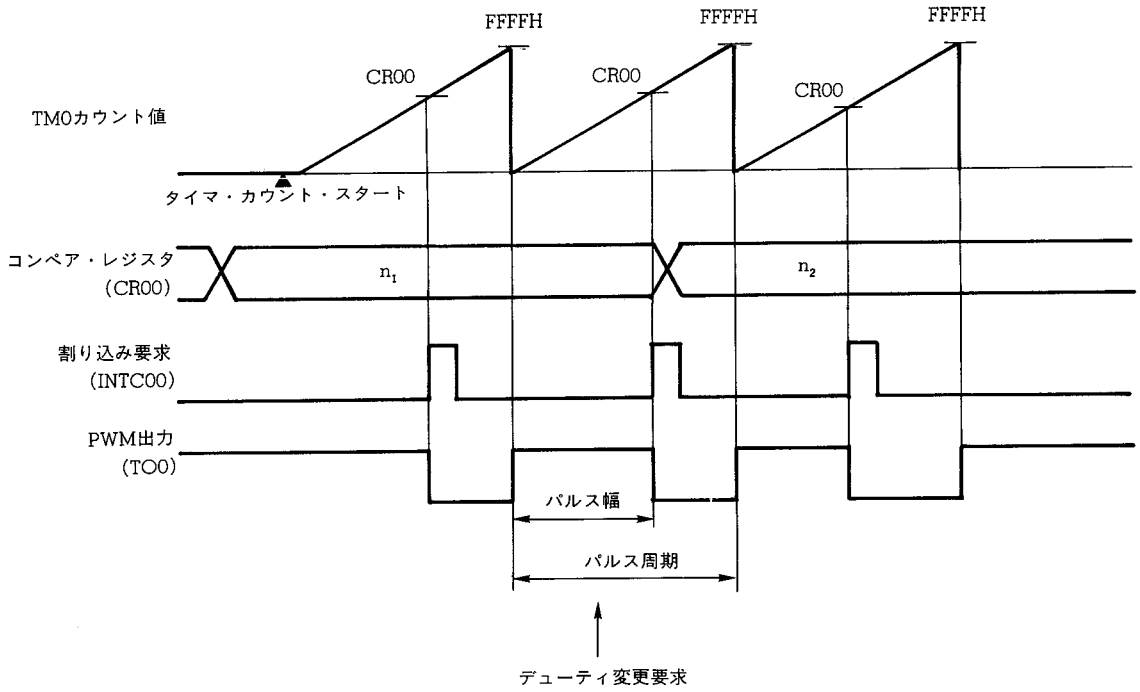


図3-12　TO0端子からのPWM出力のタイミング・チャート



備考　ALV0 = 0

パルス周期 = 65536 × 8/f_{CLK}

パルス幅　= n × 8/f_{CLK} {n : 1 ≦ n ≦ FFFFH}

出力パルスの1周期は，16ビット・タイマ・レジスタ（TM0）がフルカウント（FFFFH）する期間になりますので，タイマはフリー・ランニングさせてください。なお1周期は，f_{CLK} = 6 MHzの場合約87.4 msとなります。

またデューティ変更要求があった場合は，直後のタイマ・レジスタ（TM0）とコンペア・レジスタ（CR00）の一致によるINTC00の割り込み要求発生時に，その割り込み処理内で，パルス幅を決定するコンペア・レジスタ（CR00）の値を書き換えます。

備考　TO0, TO1端子より同時に2種類のPWM出力を行う場合は，16ビット・コンペア・レジスタ（CR00, CR01）にそれぞれのパルス幅を決定する値を設定しておきます。16ビット・コンペア・レジスタ(CR00, CR01)と16ビット・タイマ・レジスタ(TM0)の一致によって発生する2本の割り込み要求（INTC00, INTC01）によって，同時に2種類のPWM出力を行うことができます。

3
214
218A
234
244

このプログラムでは，デューティを決定する値を格納するエリアとして，2バイトのワーク・エリアを使用します。デューティ変更要求があった場合は，このエリアに次のデューティを決定する値を格納します。ワーク・エリアは，ショート・ダイレクト・アドレシングの適用範囲に配置してください。

表3-4　TM0によるPWM出力のプログラムで用いるワーク・エリア

ワーク・エリア名称	用　　途
DUTY1	デューティを決定する値を格納する

(b)　プログラム説明

(i) イニシャライズ処理［レーベル名称：PWM00］

① TO0タイマ出力のアクティブ・レベルをハイ・レベルに設定し，タイマ出力を許可します。

注意　PWM/PPG出力では，タイマ・コントロール・レジスタ(TOC)のALV0ビット＝0に設定した場合，アクティブ・レベルはハイ・レベルとなります。

② P34をTO0出力端子として用いるためにコントロール・ポートに指定します。

③ 16ビット・タイマ・レジスタ（TM0）と16ビット・コンペア・レジスタ（CR00）の一致による，16ビット・タイマ・レジスタ（TM0）のクリアを禁止し（フリー・ランニング・モード），TO0端子をPWM出力モードに設定します。

④ 16ビット・コンペア・レジスタ（CR00）にTO0端子からのPWM出力のパルス幅を決定する値を設定します。

⑤ 16ビット・タイマ/カウンタのカウント動作を許可します。

(ii) デューティ変更要求処理［レーベル名称：C_DTY0］

① INTC00割り込み要求フラグをクリアします。

② INTC00割り込み要求のマスクを解除します。

(iii) INTC00割り込み処理［レーベル名称：INTC00］

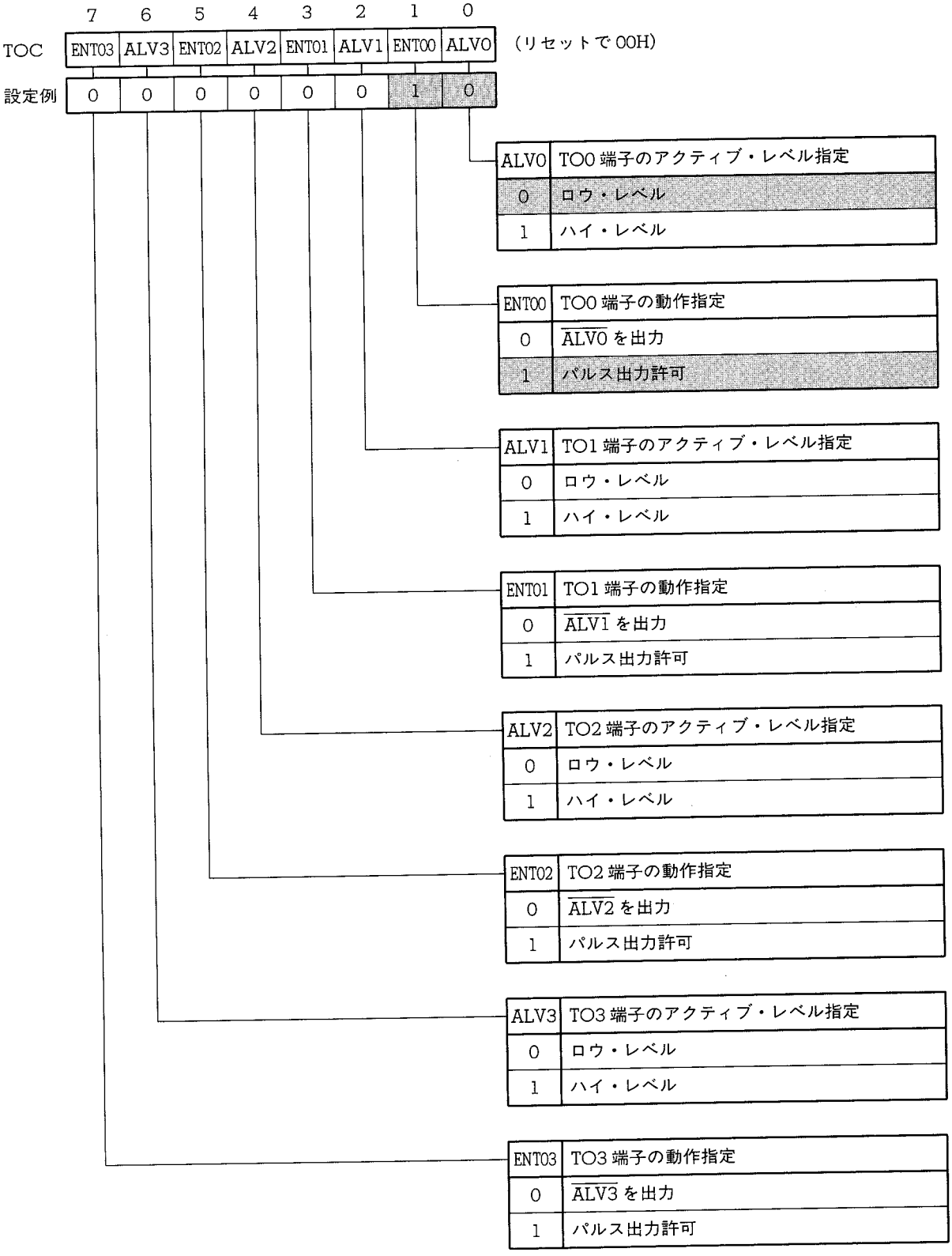
① レジスタ・バンク1に切り替えます。

② 16ビット・コンペア・レジスタ（CR00）の値を書き換えます。

③ INTC00割り込み要求をマスクします。

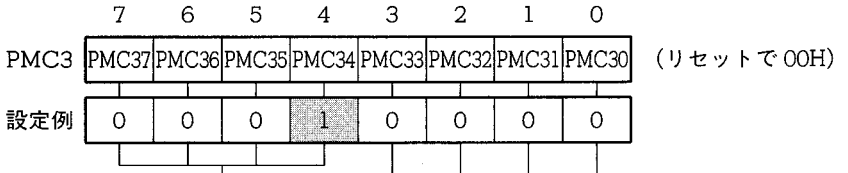
(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



3
214
218A
234
244

ポート3モード・コントロール・レジスタ



	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	（リセットで00H）
設定例	0	0	0	1	0	0	0	0	

PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

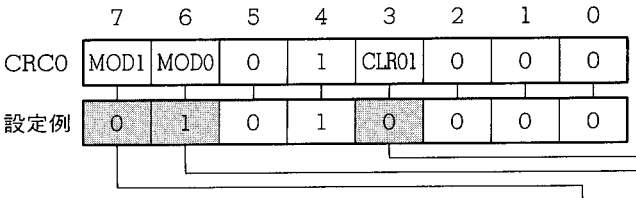
PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{\text{SCK}}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

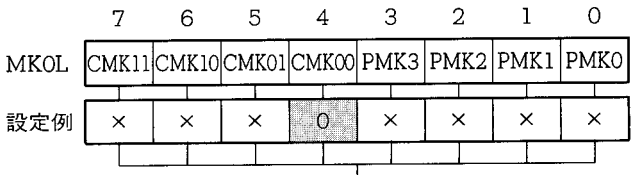
PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

キャプチャ/コンペア・コントロール・レジスタ0



MOD1	MOD0	CLR01	タイマ出力モード指定 TO0	タイマ出力モード指定 TO1	TM0＝CR01時のTM0のクリア動作
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

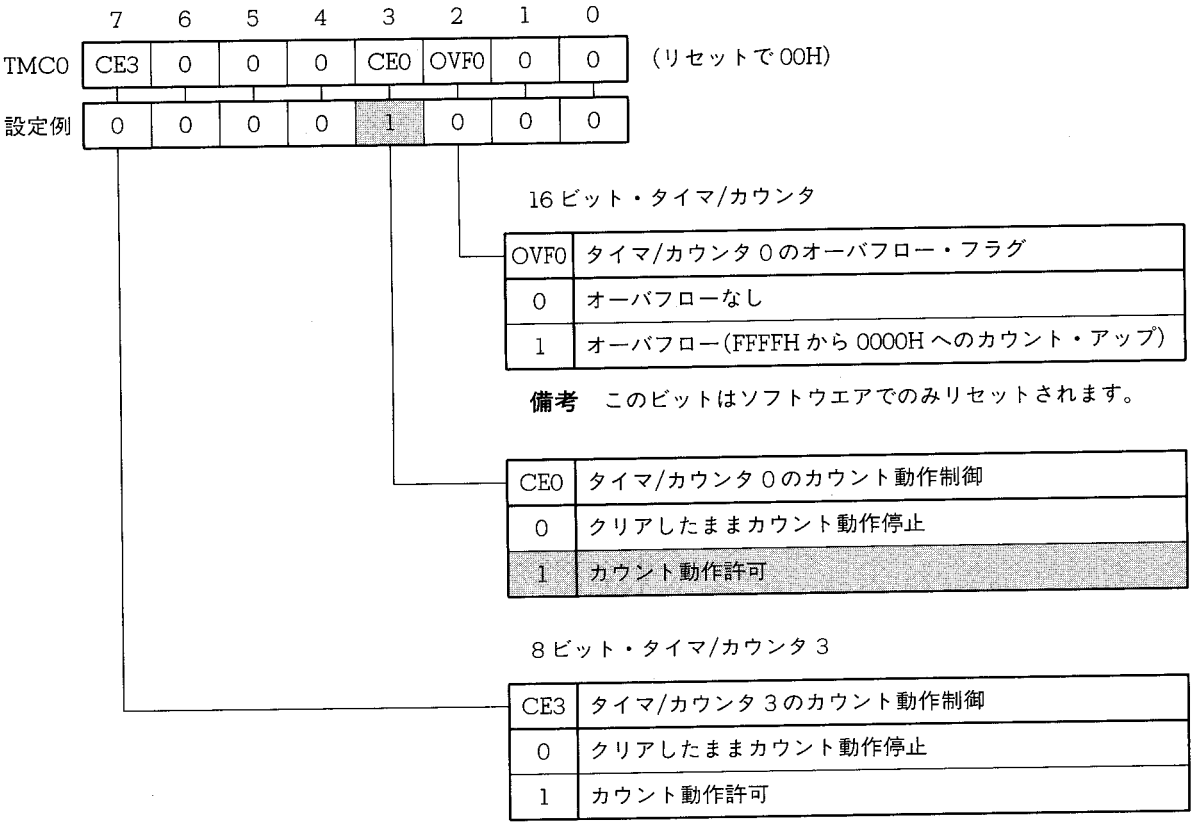
割り込みマスク・レジスタL



MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

3
214
218A
234
244

タイマ・コントロール・レジスタ0



備考　このビットはソフトウエアでのみリセットされます。

(d)　入力パラメータ

DUTY1：デューティを決定する値を設定します。

(e)　使用レジスタ

PWM00処理　：AX

C_DTY0処理　：なし

割り込み処理　：AX（レジスタ・バンク1）

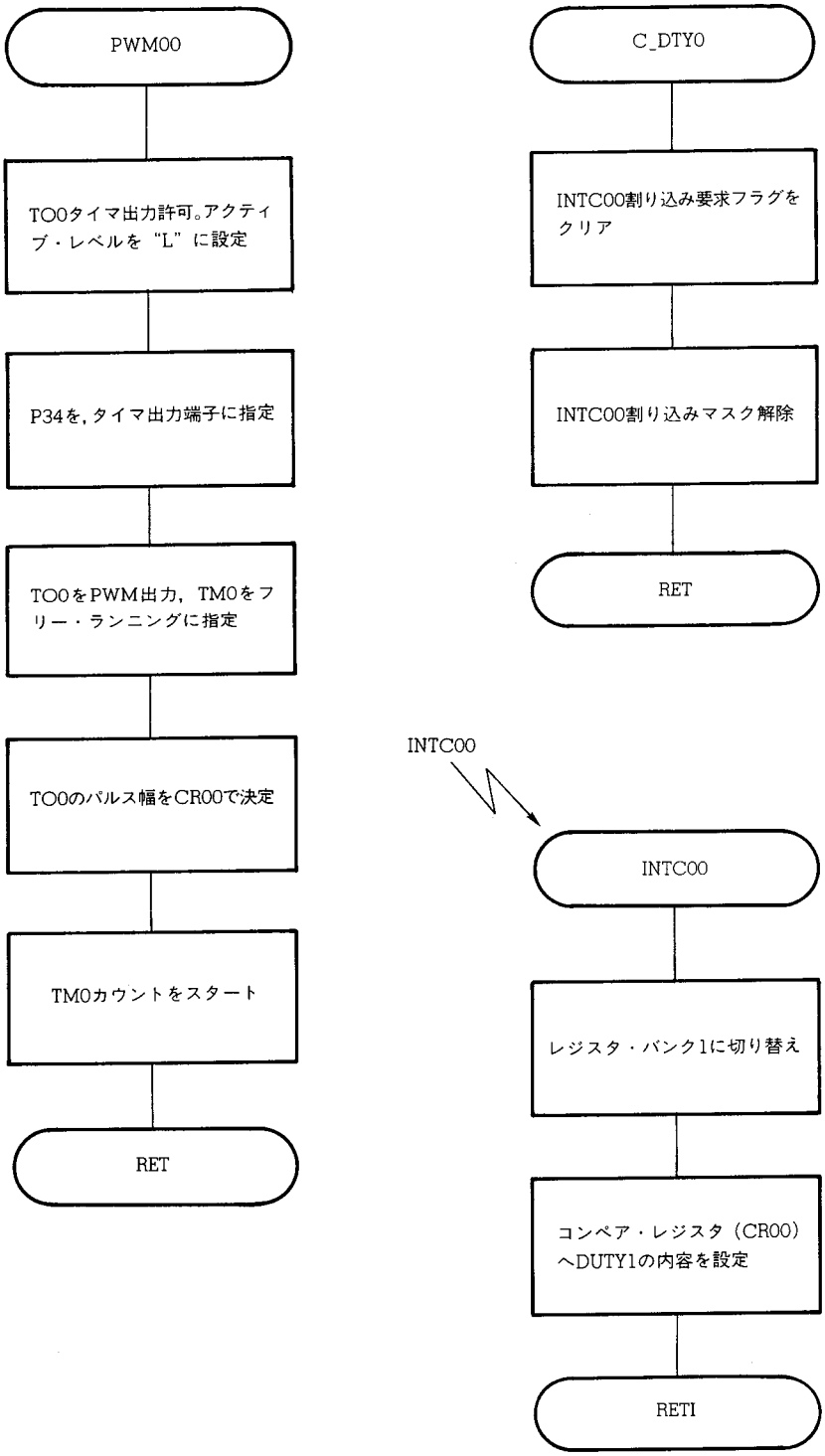
(f)　プログラム使用例

TO0端子からのPWM出力と，デューティ変更要求処理の使用例を示します。

なおデューティ変更要求処理の使用例において，次のPWM出力のデューティを決定する値は，何らかの方法でAXレジスタにセットされているものとしています。

```
        PUBLIC  DUTY1
        EXTRN   C_DTY0,PWM00
                .
                .
DUTY1_D DSEG    SADDR
DUTY1:  DS      2               ; work area for duty
                .
                .
        CSEG
OUT00:
                .
                .
        MOVW    DUTY1,#7FFFH    ; set first duty
        CALL    !PWM00          ; PWM initialize routine
        EI
                .
                .
;       << デューティ変更要求 >>

C_DTY:
                .
                .
        MOVW    DUTY1,AX        ; set next duty
        CALL    !C_DTY0         ; change duty routine
                .
                .
```

(g)　フロー・チャート


PWM00
TO0タイマ出力許可。アクティブ・レベルを“L”に設定
P34を，タイマ出力端子に指定
TO0をPWM出力，TM0をフリー・ランニングに指定
TO0のパルス幅をCR00で決定
TM0カウントをスタート
RET
C_DTY0
INTC00割り込み要求フラグをクリア
INTC00割り込みマスク解除
RET
INTC00
INTC00
レジスタ・バンク1に切り替え
コンペア・レジスタ（CR00）へDUTY1の内容を設定
RETI

(h)　プログラム・リスト

```
        NAME    PWM_0
;
;*************************************************************
;* 16bit-Timer / Counter                                     *
;*      PWM output                                           *
;*************************************************************
;
        PUBLIC  PWM00,C_DTY0
        EXTRN   DUTY1
;
CMK00   EQU     MK0L.4          ; INTC00 mask flag
CIF00   EQU     IF0L.4          ; INTC00 request flag
;
INTC00VT CSEG   AT 00014H
        DW      INTC00          ; INTC00 vector
;
        CSEG
PWM00:
        MOV     TOC,#00000010B  ; timer output,active level high
        MOV     PMC3,#00010000B ; P3 control port
        MOV     CRC0,#01010000B ; TM0 free funning mode
        MOVW    AX,DUTY1        ; set first data of duty
        MOVW    CR00,AX
        MOV     TMC0,#00001000B ; start timer
        RET
;
;       ***** request of change duty *****

C_DTY0:
        CLR1    CIF00           ; clear request flag of INTC00
        CLR1    CMK00           ; open mask of INTC00
        RET
;
;*************************************************************
;*              INTC00 interrupt routine                     *
;*************************************************************
INTC00:
        SEL     RB1
        MOVW    AX,DUTY1        ; set next data of duty
        MOVW    CR00,AX
        SET1    CMK00           ; mask INTC00
        RETI
;
        END
```

(2) 8ビット・タイマ/カウンタ2を用いたPWM出力プログラム例

TO2端子から，INTC20割り込み要求によってパルス幅を決定するPWM波形を出力するプログラム例を示します。

デューティを変更する場合は，RAM中のワーク・エリアにデューティを決定する値をセットして，デューティ変更のサブルーチンを呼び出します。

(a) 動作概要

TO2端子からのPWM出力の機能を図3-13のブロック図に示します。

図3-13　TO2端子からのPWM出力

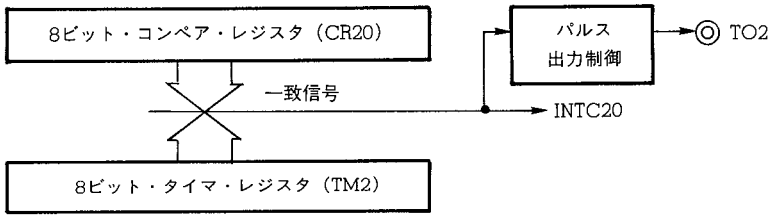
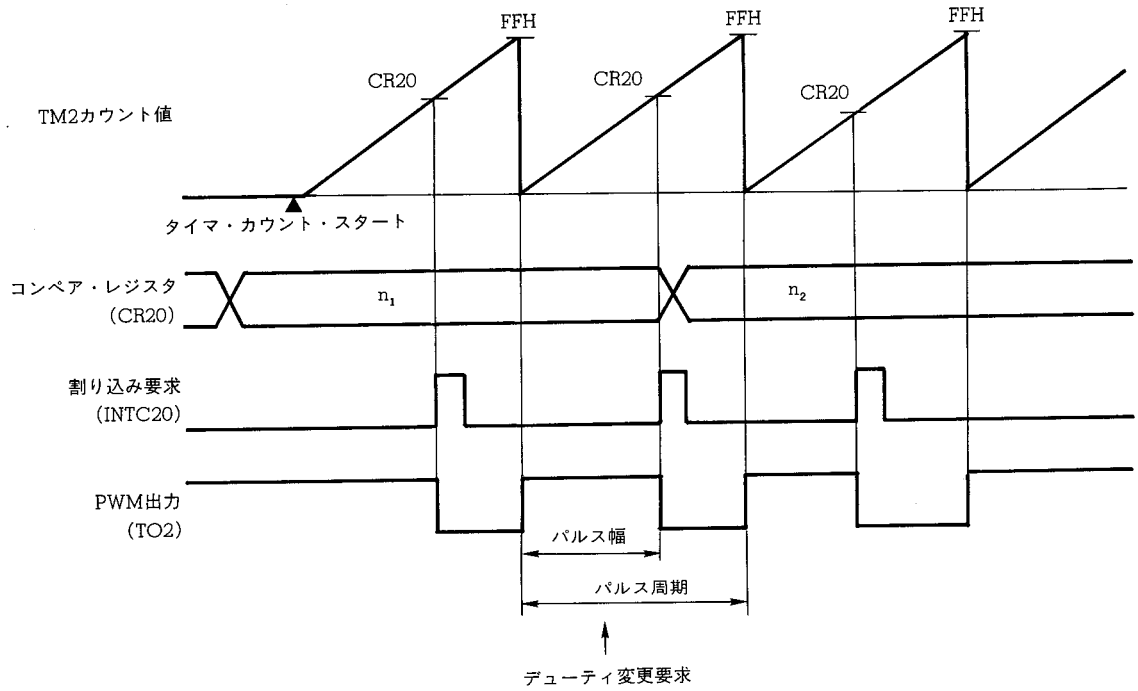


図3-14　TO2端子からのPWM出力のタイミング・チャート



備考　ALV2 = 0

パルス周期 = 256 × X/f_{CLK}

パルス幅　= n × X/f_{CLK} {n : 1 ≦ n ≦ FFH}

X = 16, 32, 64, 128, 256, 512

出力パルスの1周期は，8ビット・タイマ・レジスタ（TM2）がフル・カウント（FFH）する期間になりますので，タイマはフリー・ランニングさせてください。なお1周期の期間は，タイマのカウント・クロックによって決定します。

デューティ変更要求があった場合は，直後のタイマ・レジスタ（TM2）とコンペア・レジスタ（CR20）の一致によるINTC20割り込み要求発生時に，その割り込み処理内で，パルス幅を決定するコンペア・レジスタ（CR20）の値を書き換えます。

備考　TO2, TO3端子より同時に2種類のPWM出力を行う場合は，8ビット・コンペア・レジスタ（CR20, CR21）にそれぞれのパルス幅を決定する値を設定しておきます。8ビット・コンペア・レジスタ(CR20, CR21)と8ビット・タイマ・レジスタ(TM2)の一致によって発生する2本の割り込み要求（INTC20, INTC21）によって，同時に2種類のPWM出力を行うことができます。

このプログラムでは，デューティを決定する値を格納するエリアとして，1バイトのワーク・エリアを使用します。デューティ変更要求があった場合は，このエリアに次のデューティを決定する値を格納します。ワーク・エリアは，ショート・ダイレクト・アドレシングの適用範囲に配置してください。

表3-5　TM2によるPWM出力のプログラムで用いるワーク・エリア

ワーク・エリア名称	用　　途
DUTY3	デューティを決定する値を格納する

3
214
218A
234
244

(b) プログラム説明

(i) イニシャライズ処理［レーベル名称：PWM20］

① TO2タイマ出力のアクティブ・レベルをハイ・レベルに設定し，タイマ出力を許可します。

注意　PWM/PPG出力では，タイマ・コントロール（TOC）のALV2ビット＝0に設定した場合，アクティブ・レベルはハイ・レベルとなります。

② P36をTO2出力端子として用いるためにコントロール・ポートに指定します。

③ 8ビット・タイマ・レジスタ（TM2）と8ビット・コンペア・レジスタ（CR20）の一致による，8ビット・タイマ・レジスタ（TM2）のクリアを禁止し（フリー・ランニング・モード），TO2端子をPWM出力モードに設定します。

④ 8ビット・タイマ/カウンタ2のカウント・クロックをf_{CLK}/16に設定します。

⑤ 8ビット・コンペア・レジスタ(CR20)にTO2端子からのPWM出力のパルス幅を決定する値を設定します。

⑥ 8ビット・タイマ/カウンタ2のカウンタ動作を許可します。

(ii) デューティ変更要求処理［レーベル名称：C_DTY2］

① INTC20割り込み要求フラグをクリアします。

② INTC20割り込み要求のマスクを解除します。

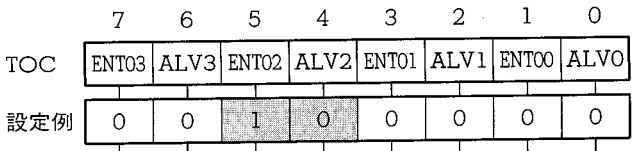
(iii) INTC20割り込み処理［レーベル名称：INTC20］

① 8ビット・コンペア・レジスタ（CR20）の値を書き換えます。

② INTC20割り込み要求をマスクします。

(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



	7	6	5	4	3	2	1	0	
TOC	ENT03	ALV3	ENT02	ALV2	ENT01	ALV1	ENT00	ALV0	(リセットで00H)
設定例	0	0	1	0	0	0	0	0	

ALV0	TO0端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT00	TO0端子の動作指定
0	$\overline{\text{ALV0}}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT01	TO1端子の動作指定
0	$\overline{\text{ALV1}}$を出力
1	パルス出力許可

ALV2	TO2端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

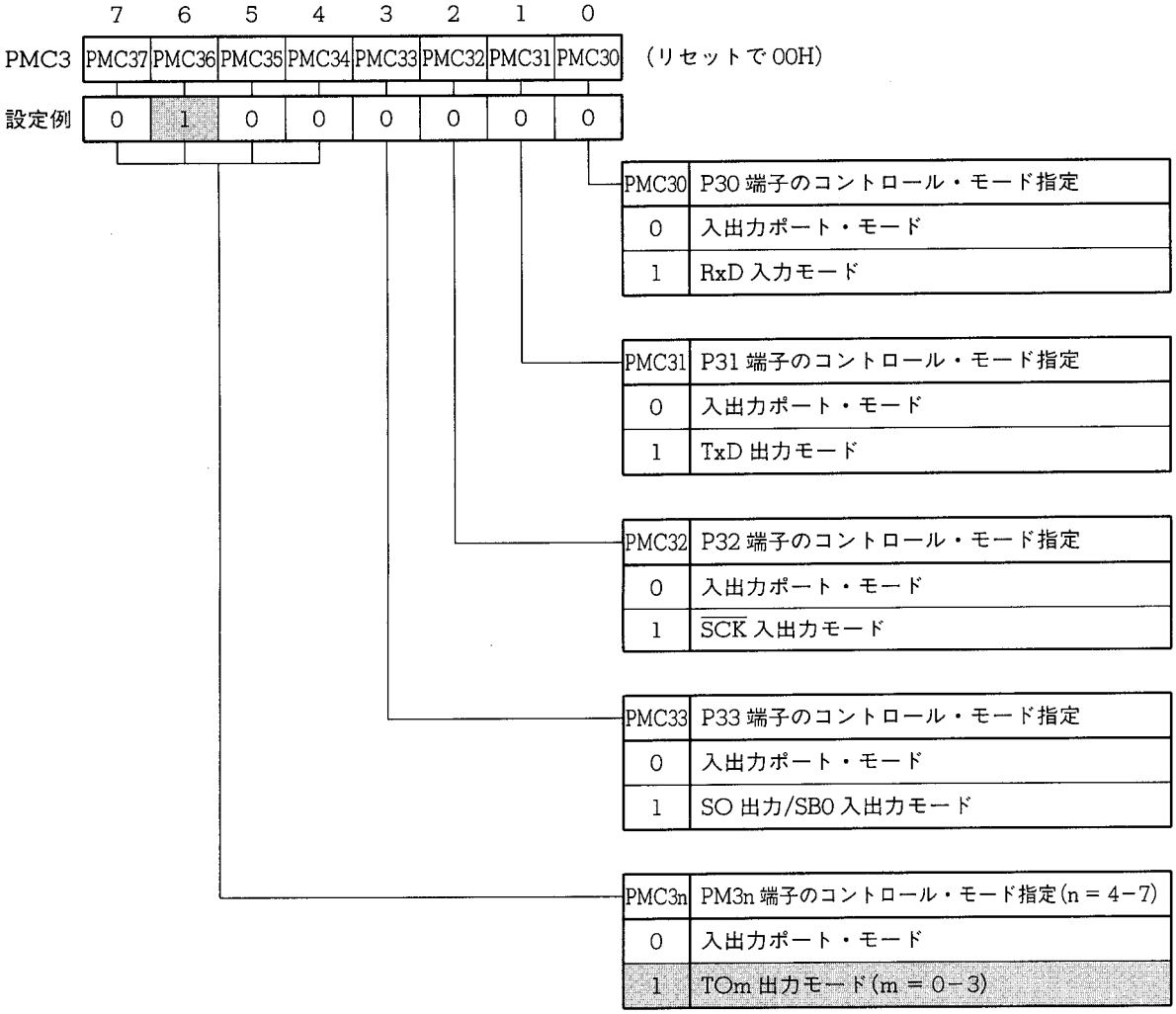
ENT02	TO2端子の動作指定
0	$\overline{\text{ALV2}}$を出力
1	パルス出力許可

ALV3	TO3端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

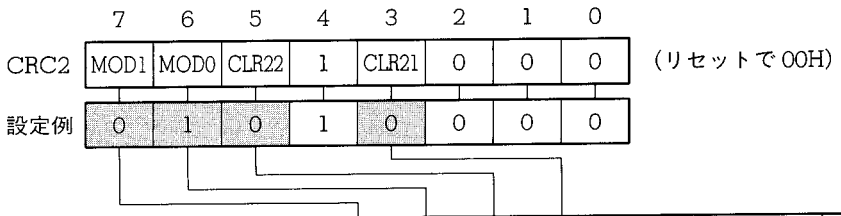
ENT03	TO3端子の動作指定
0	$\overline{\text{ALV3}}$を出力
1	パルス出力許可

3
214
218A
234
244

ポート3モード・コントロール・レジスタ



キャプチャ/コンペア・コントロール・レジスタ2



MOD1	MOD0	CLR22	CLR21	タイマ出力モード		TM2のクリア動作
				TO2	TO3	
0	0	0	0	トグル出力	トグル出力	クリアしない
0	0	0	1	トグル出力	トグル出力	TM2とCR21レジスタの一致でクリア
0	0	1	0	トグル出力	トグル出力	TM2の内容をCR22レジスタへキャプチャ後にクリア
0	0	1	1	トグル出力	トグル出力	TM2とCR21レジスタの一致またはTM2の内容をCR22レジスタへキャプチャ後にクリア
0	1	0	0	PWM出力	トグル出力	クリアしない
1	0	0	0	PWM出力	PWM出力	クリアしない
1	1	0	1	PPG出力	トグル出力	TM2とCR21レジスタの一致でクリア

注意　上記以外の組み合わせは禁止

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	（リセットで00H）
設定例	0	0	1	1	0	0	0	0	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタ1のカウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$[注]
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタ3のカウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

注　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

割り込みマスク・レジスタH

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	(リセットでFFH)
設定例	×	×	×	×	0	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

タイマ・コントロール・レジスタ1

	7	6	5	4	3	2	1	0	
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0	(リセットで00H)
設定例	1	0	0	0	0	0	0	0	

8ビット・タイマ/カウンタ1

OVF1	タイマ/カウンタ1のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	タイマ/カウンタ1のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ2

CMD2	タイマ/カウンタ2の動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	タイマ/カウンタ2のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	タイマ/カウンタ2のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

3
214
218A
234
244

(d)　入力パラメータ

DUTY3：デューティを決定する値を設定します。

(e)　使用レジスタ

なし

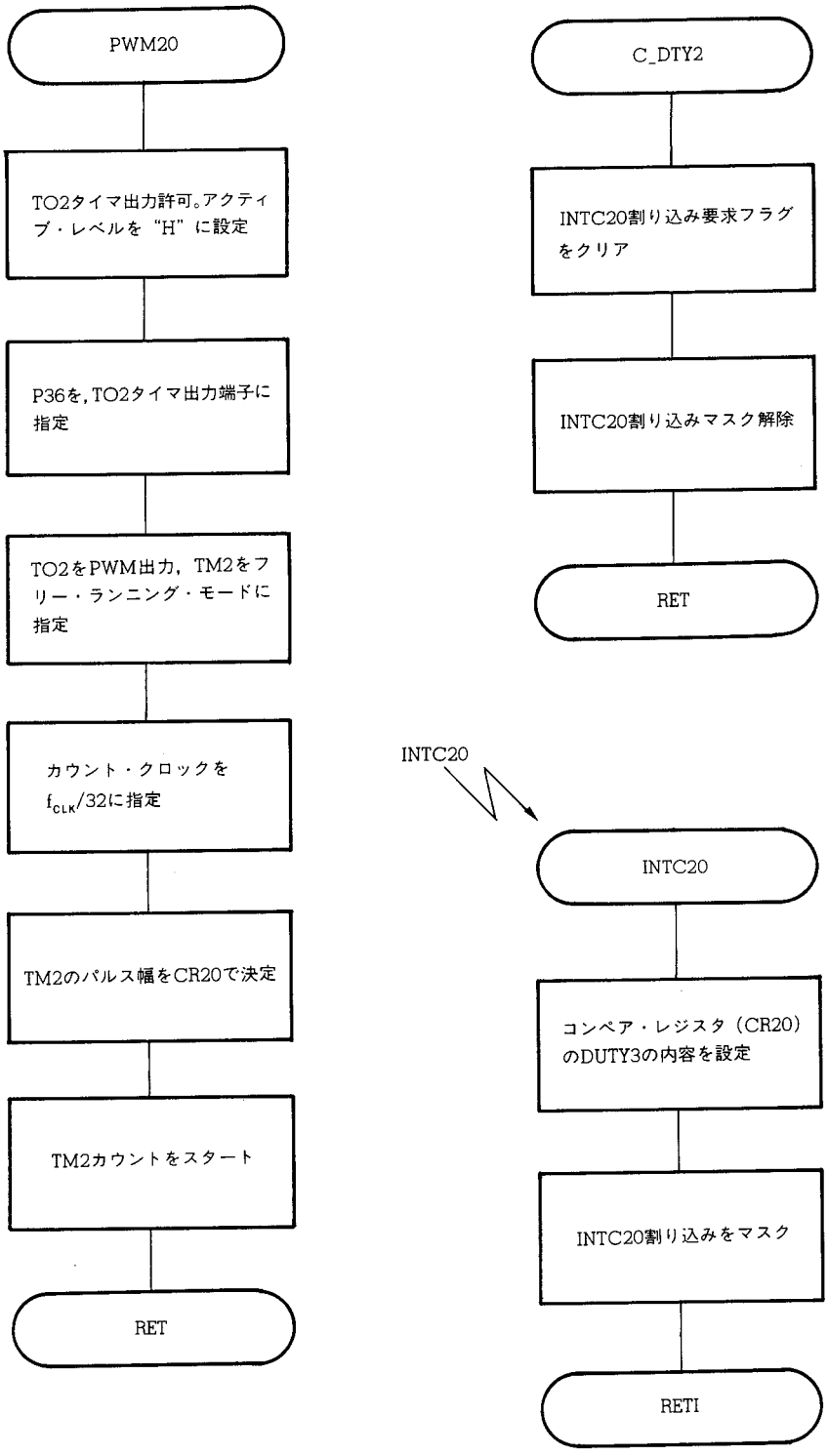
(f)　プログラム使用例

TO2端子からのPWM出力と，デューティ変更要求処理の使用例を示します。

なおデューティ変更要求処理の使用例において，次のPWM出力のデューティを決定する値は，何らかの方法でAレジスタにセットされているものとしています。

```
        PUBLIC  DUTY3
        EXTRN   C_DTY2,PWM20
                .
                .
DUTY3_D DSEG    SADDR
DUTY3:  DS      1               ; work area for duty
                .
                .
        CSEG
OUT20:
                .
                .
        MOV     DUTY3,#7FH      ; set first duty
        CALL    !PWM20          ; PWM initialize routine
        EI
                .
                .
;       << デューティ変更要求 >>

C_DTY:
                .
                .
        MOV     DUTY3,A         ; set next duty
        CALL    !C_DTY2
                .
                .
```

(g)　フロー・チャート



3
214
218A
234
244

(h)　プログラム・リスト

```
        NAME    PWM_2
;
;*************************************************************
;* 8bit-Timer / Counter-2                                    *
;*      PWM output                                           *
;*************************************************************
;
        PUBLIC  PWM20,C_DTY2
        EXTRN   CYCL2,DUTY3
;
CMK20   EQU     MK0H.3          ; INTC20 mask flag
CIF20   EQU     IF0H.3          ; INTC20 request flag
;
INTC20VT CSEG   AT 00012H
        DW      INTC20          ; INTC20 vector
;
        CSEG
PWM20:
        MOV     TOC,#00100000B  ; timer output,active level high
        MOV     PMC3,#01000000B ; P3 control port
        MOV     CRC2,#01010000B ; TM2 free funning mode
        MOV     PRM1,#00110000B ; TM2 prescaler fclk/32
        MOV     CR20,DUTY3      ; set first duty
        MOV     TMC1,#10000000B ; start timer
        RET
;
;       ***** request of change duty *****

C_DTY2:
        CLR1    CIF20           ; clear request flag of INTC20
        CLR1    CMK20           ; open mask of INTC20
        RET
;
;*************************************************************
;*              INTC20 interrupt routine                     *
;*************************************************************

INTC20:
        MOV     CR20,DUTY3      ; set next duty
        SET1    CMK20           ; mask INTC20
        RETI
;
        END
```

(3) 16ビット・タイマ/カウンタを用いたPPG出力プログラム例

TO0端子から，INTC01割り込み要求によって周期を，INTC00割り込み要求によってパルス幅を決定するPPG波形を出力するプログラム例を示します。

デューティを変更する場合は，RAM中のワーク・エリアにデューティを決定する値をセットして，デューティ変更のサブルーチンを呼び出します。

(a) 動作概要

TO0端子からのPPG出力の機能を図3-15のブロック図に示します。

図3-15　TO0端子からのPPG出力

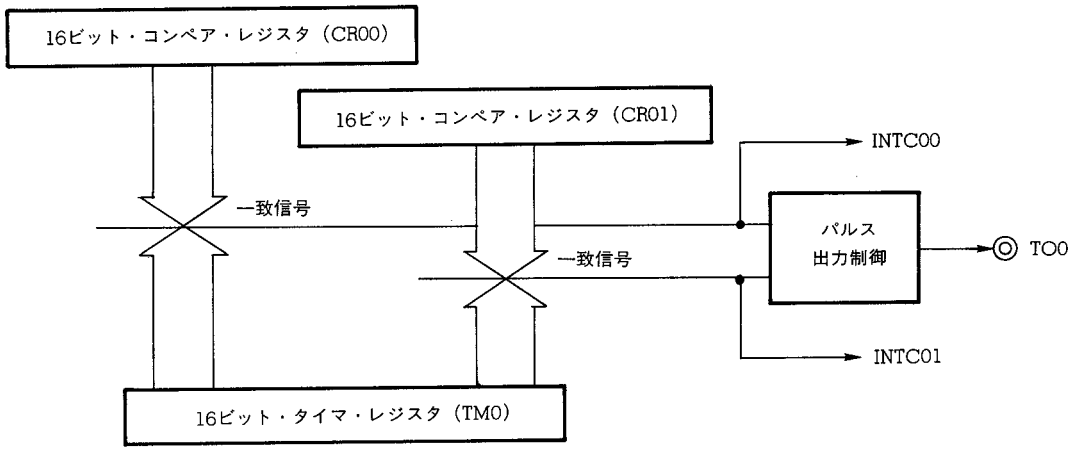
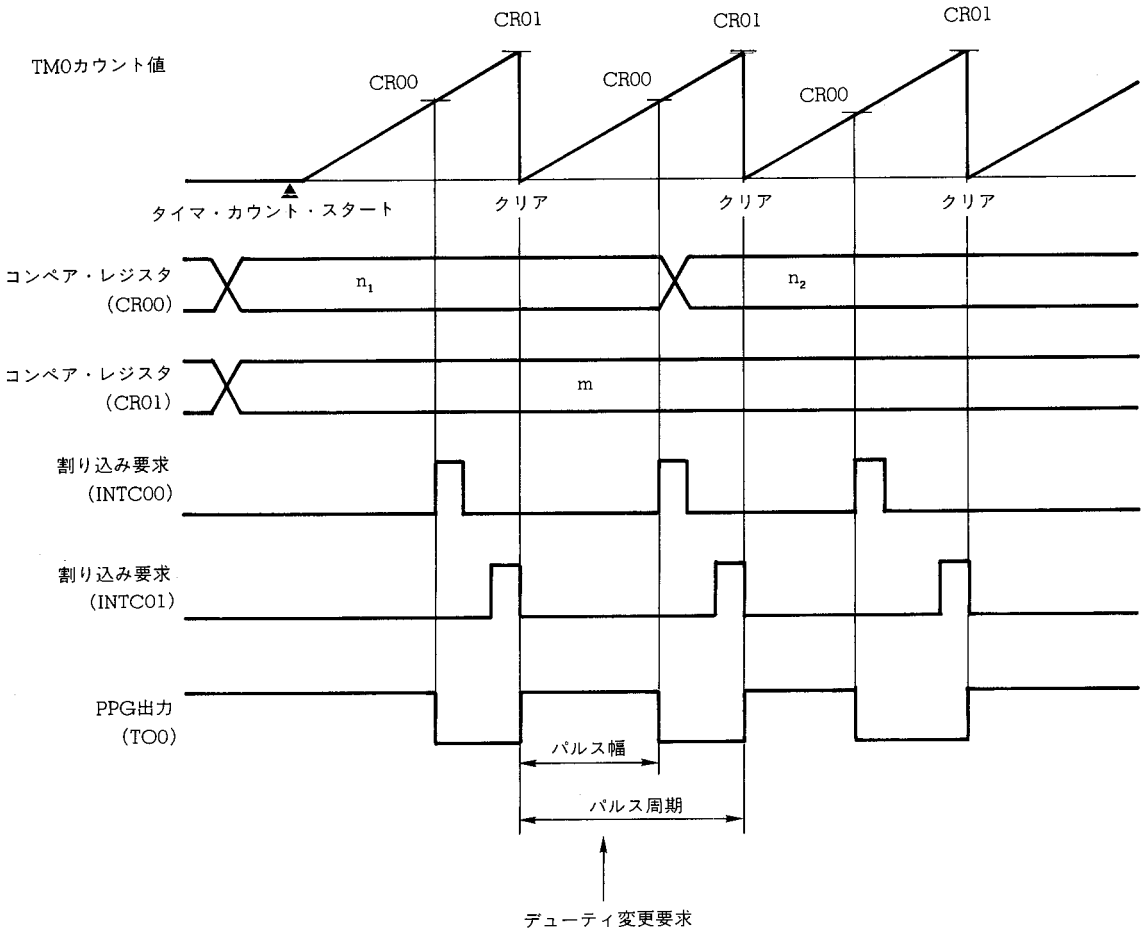


図3-16　TO0端子からのPPG出力のタイミング・チャート



備考　ALV0 = 0

パルス周期＝ (m+1) × 8/f_{CLK} {m : 2 ≦ m ≦ FFFFH}

パルス幅　＝ n × 8/f_{CLK} {n : 1 ≦ n ≦ FFFFH}

ただし，パルス幅（CR00）≦パルス周期（CR01）

出力パルスの1周期は，16ビット・コンペア・レジスタ(CR01)の値と16ビット・タイマ・レジスタ（TM0）の値が一致するまでの期間になりますので，タイマは，TM0とCR01の一致でのクリアを許可にします。パルス幅は，16ビット・コンペア・レジスタ（CR00）の値と16ビット・タイマ・レジスタ（TM0）の値が一致するまでの期間になります。

またデューティ変更要求があった場合は，直後のタイマ・レジスタ（TM0）とコンペア・レジスタ（CR00）の一致によるINTC00割り込み要求発生時に，その割り込み処理内で，パルス幅を決定するコンペア・レジスタ（CR00）の値を書き換えます。

このプログラムでは，デューティを決定する値を格納するエリアとして，2バイトのワーク・エリアを使用します。デューティ変更要求があった場合は，このエリアに次のデューティを決定する値を格納します。ワーク・エリアは，ショート・ダイレクト・アドレシングの適用範囲に配置してください。

表3-6　TM0によるPPG出力のプログラムで用いるワーク・エリア

ワーク・エリア名称	用　　途
DUTY2	デューティを決定する値を格納する

(b)　プログラム説明……(h)　プログラム・リスト参照

(i) イニシャライズ処理［レーベル名称：PPG00］

① TO0タイマ出力のアクティブ・レベルをハイ・レベルに設定し，タイマ出力を許可します。

注意　PWM/PPG出力では，タイマ・コントロール・レジスタ(TOC)のALV0ビット＝0に設定した場合，アクティブ・レベルはハイ・レベルとなります。

② P34をTO0出力端子として用いるためにコントロール・ポートに指定します。

③ 16ビット・タイマ・レジスタ（TM0）と16ビット・コンペア・レジスタ（CR01）の一致による，16ビット・タイマ・レジスタ（TM0）のクリアを許可し，TO0端子をPPG出力モードに設定します。

④ 16ビット・コンペア・レジスタ（CR01）にTO0端子からのPPG出力の周期を決定する値を設定します。また16ビット・コンペア・レジスタ（CR00）にTO0端子からのPPG出力パルス幅を決定する値を設定します。

⑤ 16ビット・タイマ/カウンタのカウント動作を許可します。

(ii) デューティ変更要求処理［レーベル名称：C_DTY0］

① INTC00割り込み要求フラグをクリアします。

② INTC00割り込み要求のマスクを解除します。

(iii) INTC00割り込み処理［レーベル名称：INTC00］

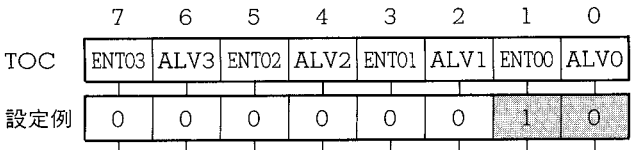
① レジスタ・バンク1に切り替えます。

② 16ビット・コンペア・レジスタ（CR00）の値を書き換えます。

③ INTC00割り込み要求をマスクします。

(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



ALV0	TO0端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT00	TO0端子の動作指定
0	$\overline{\text{ALV0}}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT01	TO1端子の動作指定
0	$\overline{\text{ALV1}}$を出力
1	パルス出力許可

ALV2	TO2端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

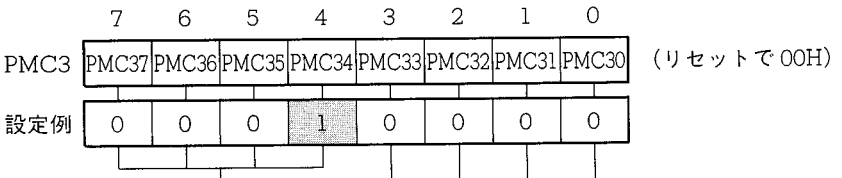
ENT02	TO2端子の動作指定
0	$\overline{\text{ALV2}}$を出力
1	パルス出力許可

ALV3	TO3端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT03	TO3端子の動作指定
0	$\overline{\text{ALV3}}$を出力
1	パルス出力許可

3
214
218A
234
244

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

キャプチャ/コンペア・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
CRC0	MOD1	MOD0	0	1	CLR01	0	0	0	（リセットで10H）
設定例	1	1	0	1	1	0	0	0	

MOD1	MOD0	CLR01	タイマ出力モード指定		TM0＝CR01時のTM0のクリア動作
			TO0	TO1	
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

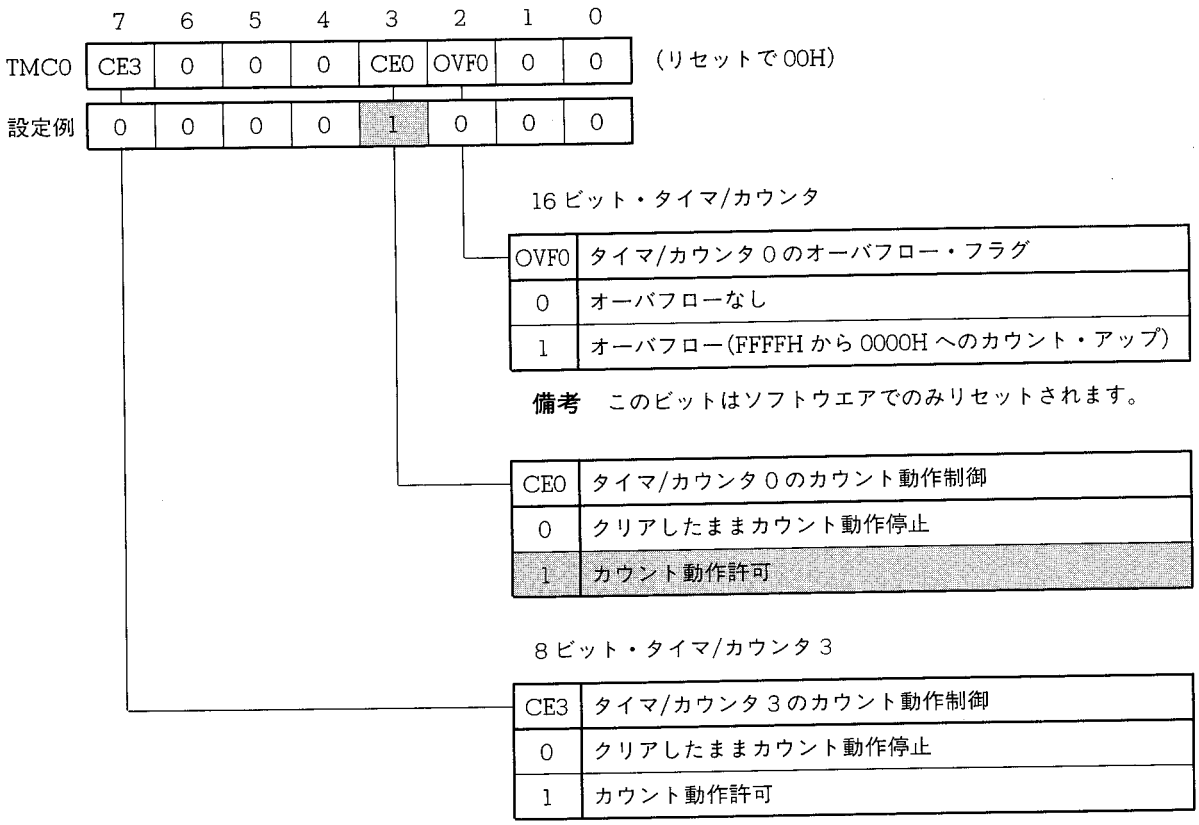
割り込みマスク・レジスタL

	7	6	5	4	3	2	1	0	
MK0L	CMK11	CMK10	CMK01	CMK00	PMK3	PMK2	PMK1	PMK0	（リセットでFFH）
設定例	×	×	×	0	×	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

3
214
218A
234
244

タイマ・コントロール・レジスタ0



備考　このビットはソフトウエアでのみリセットされます。

(d)　入力パラメータ

CYCL0　：TO0 端子からの PPG 出力の周期を決定する値です。

DUTY2　：デューティを決定する値を設定します。

(e)　使用レジスタ

PPG00 処理　：AX

C_DTY0 処理　：なし

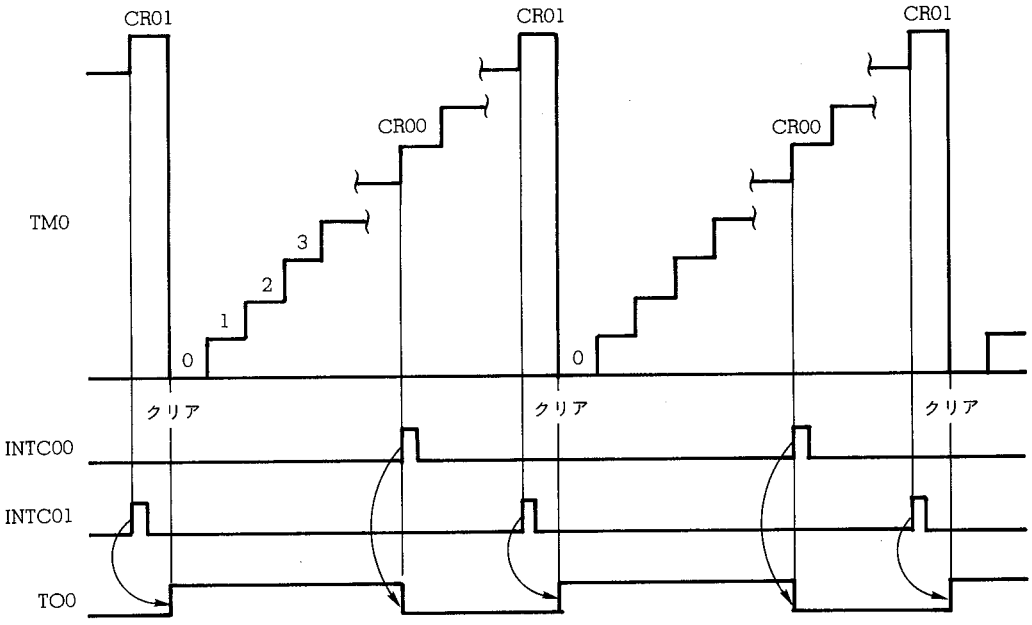
割り込み処理　：AX （レジスタ・バンク 1）

(f)　プログラム使用例

TO0 端子からの PPG 出力と，デューティ変更要求処理の使用例を示します。周期は 12.5 Hz 固定とします。

備考　PPG 出力の場合，INTC00, INTC01 の発生タイミングと PPG 出力のタイミングは，図 3 - 17 のようになります。

図 3 - 17　PPG出力タイミング（TM0）

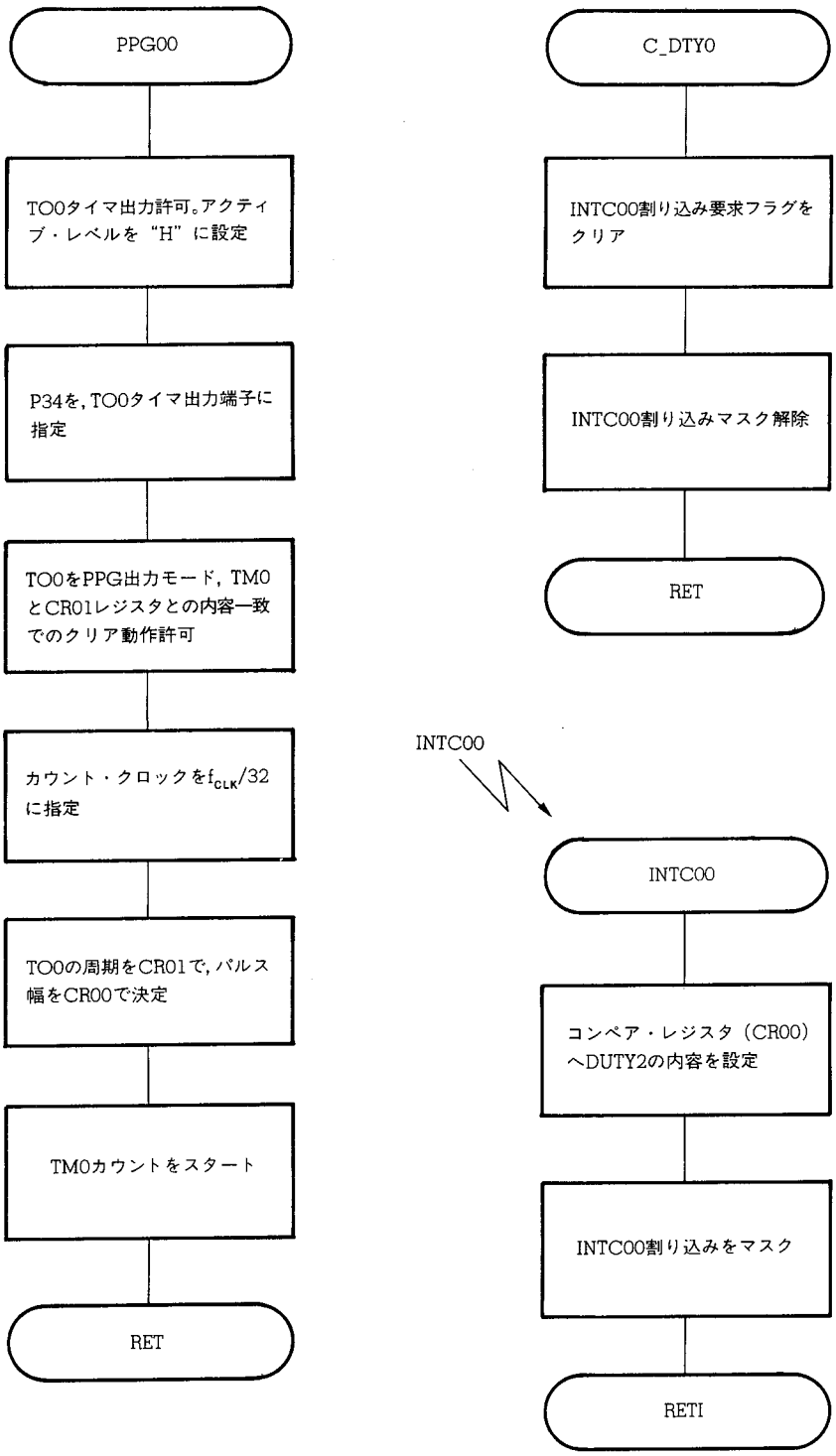


周期を決定する INTC01 発生後，タイマの 1 カウント分あとにタイマ出力が反転します。したがってコンペア・レジスタ（CR01）には，設定したい 1 周期の時間から計算される値－1 を設定します。

プログラム例として次のように用います。なおデューティ変更要求処理の使用例において，次のデューティを決定する値は，何らかの方法でAXレジスタにセットされているものとしています。

```
        PUBLIC  DUTY2,CYCL0
        EXTRN   C_DTY0,PPG00
                .
                .
CYCL0   EQU     59999           ; PPG output cycle

DUTY2_D DSEG    SADDR
DUTY2:  DS      2               ; work area for duty
                .
                .
        CSEG
OUT00:
                .
                .
        MOVW    DUTY2,#7FFFH    ; set first duty
        CALL    !PPG00          ; PPG initialize routine
        EI
                .
                .
;       << デューティ変更要求 >>

C_DTY:
                .
                .
        MOVW    DUTY2,AX        ; set next duty
        CALL    !C_DTY0
                .
                .
```

(g)　フロー・チャート


PPG00
TO0タイマ出力許可。アクティブ・レベルを“H”に設定
P34を，TO0タイマ出力端子に指定
TO0をPPG出力モード，TM0とCR01レジスタとの内容一致でのクリア動作許可
カウント・クロックをf_{CLK}/32に指定
TO0の周期をCR01で，パルス幅をCR00で決定
TM0カウントをスタート
RET
C_DTY0
INTC00割り込み要求フラグをクリア
INTC00割り込みマスク解除
RET
INTC00
INTC00
コンペア・レジスタ（CR00）へDUTY2の内容を設定
INTC00割り込みをマスク
RETI

(h)　プログラム・リスト

```
        NAME    PPG_0
;
;*************************************************************
;* 16bit-Timer / Counter                                     *
;*      PPG output                                           *
;*************************************************************
;
        PUBLIC  PPG00,C_DTY0
        EXTRN   CYCL0,DUTY2
;
CMK00   EQU     MK0L.4          ; INTC00 mask flag
CIF00   EQU     IF0L.4          ; INTC00 request flag
;
INTC00VT CSEG   AT 00014H
        DW      INTC00          ; INTC00 vector
;
        CSEG
PPG00:
        MOV     TOC,#00000010B  ; timer output,active level high
        MOV     PMC3,#00010000B ; P3 control port
        MOV     CRC0,#11011000B ; TM0 PPG output mode
        MOVW    CR01,#CYCL0     ; set output cycle
        MOVW    AX,DUTY2        ; set first output duty
        MOVW    CR00,AX
        MOV     TMC0,#00001000B ; start timer

        RET
;
;       ***** request of change duty *****

C_DTY0:
        CLR1    CIF00           ; clear request flag of INTC00
        CLR1    CMK00           ; open mask of INTC00
        RET
;
;*************************************************************
;*               INTC00 interrupt routine                    *
;*************************************************************

INTC00:
        SEL     RB1
        MOVW    AX,DUTY2        ; set data of duty
        MOVW    CR00,AX
        SET1    CMK00           ; mask INTC00
        RETI
;
        END
```

(4) 8ビット・タイマ/カウンタ2を用いたPPG出力プログラム例

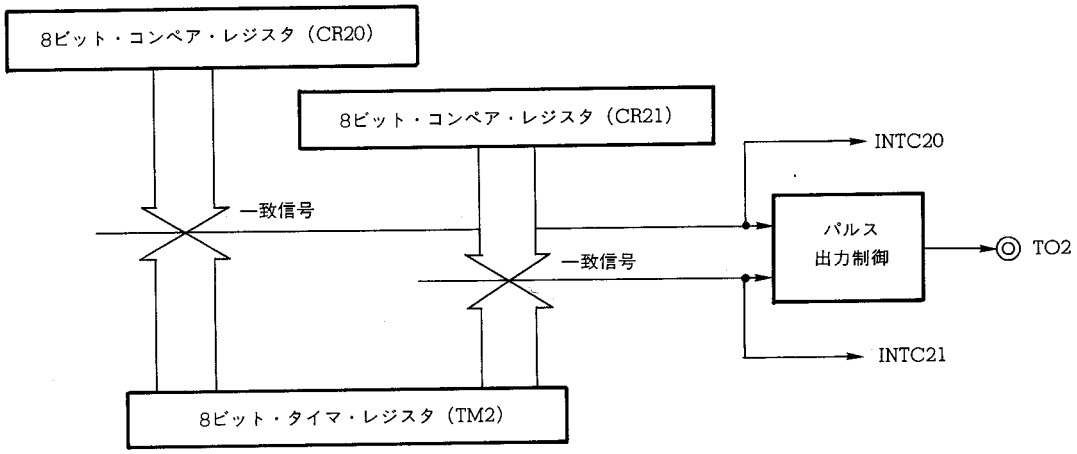
TO2端子から，INTC21割り込み要求によって周期を，INTC20割り込み要求によってパルス幅を決定するPPG波形を出力するプログラム例を示します。

デューティを変更する場合は，RAM中のワーク・エリアにデューティを決定する値をセットして，デューティ変更のサブルーチンを呼び出します。

(a) 動作概要

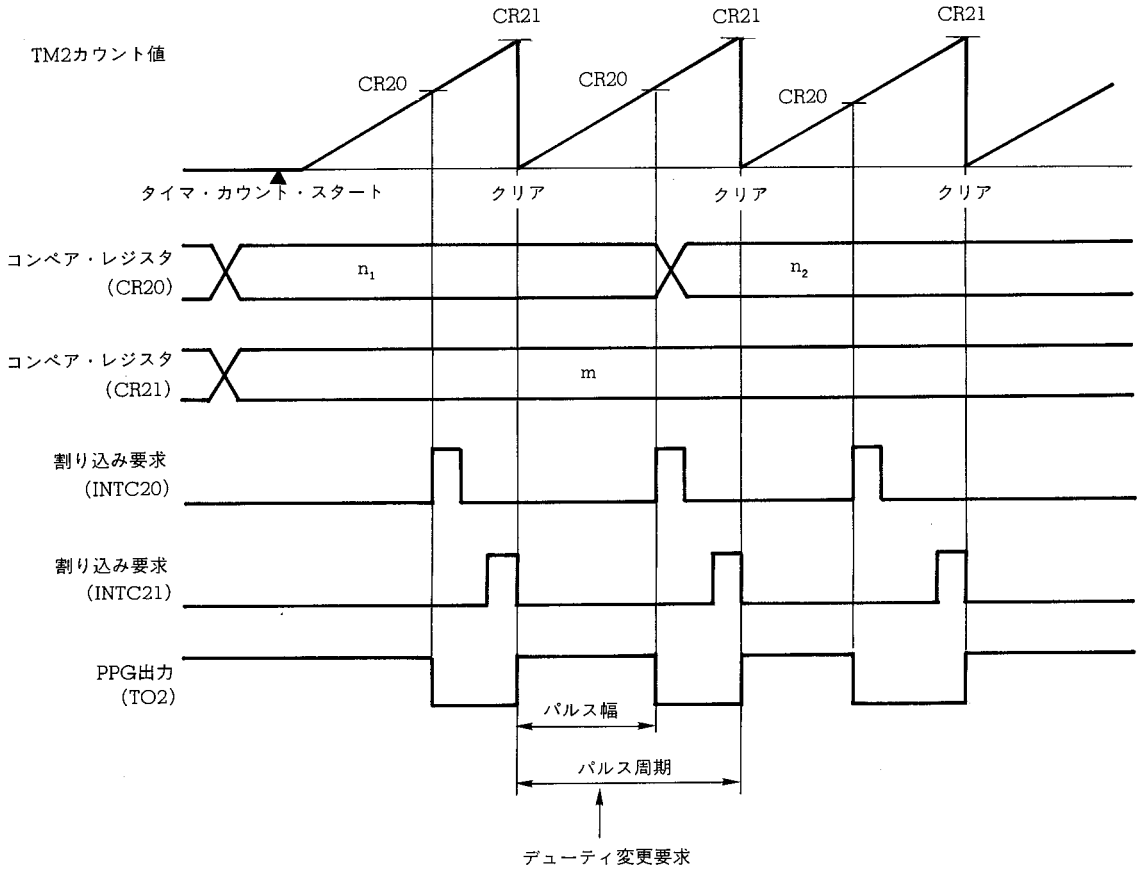
TO2端子からのPPG出力の機能を図3-18のブロック図に示します。

図3-18　TO2端子からのPPG出力



214
218A
234
244

図3-19　TO2端子からのPPG出力のタイミング・チャート



備考　ALV2 = 0

パルス周期＝ (m+1) × X/f_{CLK} {m : $2 \leqq m \leqq FFH$}

X = 16, 32, 64, 128, 256, 512

パルス幅　＝ n × X/f_{CLK} {n : $1 \leqq n \leqq FFH$}

ただし，パルス幅（CR20）≦パルス周期（CR21）

出力パルスの1周期は，8ビット・コンペア・レジスタ（CR21）の値と8ビット・タイマ・レジスタ（TM2）の値が一致するまでの期間になりますので，タイマは，TM2とCR21の一致でのクリアを許可にします。パルス幅は，8ビット・コンペア・レジスタ（CR20）の値と8ビット・タイマ・レジスタ2（TM2）の値が一致するまでの期間になります。

またデューティ変更要求があった場合は，直後のタイマ・レジスタ（TM2）とコンペア・レジスタ（CR20）の一致によるINTC20割り込み要求発生時に，その割り込み処理内で，パルス幅を決定するコンペア・レジスタ（CR20）の値を書き換えます。

このプログラムでは，デューティを決定する値を格納するエリアとして，1バイトのワーク・エリアを使用します。デューティ変更要求があった場合は，このエリアに次のデューティを決定する値を格納します。ワーク・エリアは，ショート・ダイレクト・アドレシングの適用範囲に配置してください。

表3-7　TM2によるPPG出力のプログラムで用いるワーク・エリア

ワーク・エリア名称	用　　途
DUTY4	デューティを決定する値を格納する

214
218A
234
244

(b)　プログラム説明……(h)　プログラム・リスト参照

(i) イニシャライズ処理［レーベル名称：PPG20］

① TO2タイマ出力のアクティブ・レベルをハイ・レベルに設定し，タイマ出力を許可します。

注意　PWM/PPG出力では，タイマ・コントロール・レジスタ(TOC)のALV2ビット＝0に設定した場合，アクティブ・レベルはハイ・レベルとなります。

② P36をTO2出力端子として用いるためにコントロール・ポートに指定します。

③ 8ビット・タイマ・レジスタ（TM2）と8ビット・コンペア・レジスタ（CR21）の一致による8ビット・タイマ・レジスタ（TM2）のクリアを許可し，TO2端子をPPG出力モードに設定します。

④ 8ビット・タイマ/カウンタ2のカウント・クロックをf_{CLK}/64に指定します。

⑤ 8ビット・コンペア・レジスタ（CR21）にTO2端子からのPPG出力の周期を決定する値を，8ビット・コンペア・レジスタ（CR20）にTO2端子からのPPG出力のパルス幅を決定する値を設定します。

⑥ 8ビット・タイマ/カウンタ2のカウント動作を許可します。

(ii) デューティ変更要求処理［レーベル名称：C_DTY2］

① INTC20割り込み要求フラグをクリアします。

② INTC20割り込み要求のマスクを解除します。

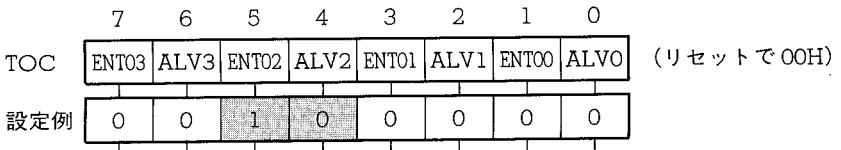
(iii) INTC20割り込み処理［レーベル名称：INTC20］

① 8ビット・コンペア・レジスタ（CR20）の値を書き換えます。

② INTC20割り込み要求をマスクします。

(c)　モード・レジスタ設定例

タイマ出力コントロール・レジスタ



ALV0	TO0端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT00	TO0端子の動作指定
0	$\overline{\text{ALV0}}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT01	TO1端子の動作指定
0	$\overline{\text{ALV1}}$を出力
1	パルス出力許可

ALV2	TO2端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

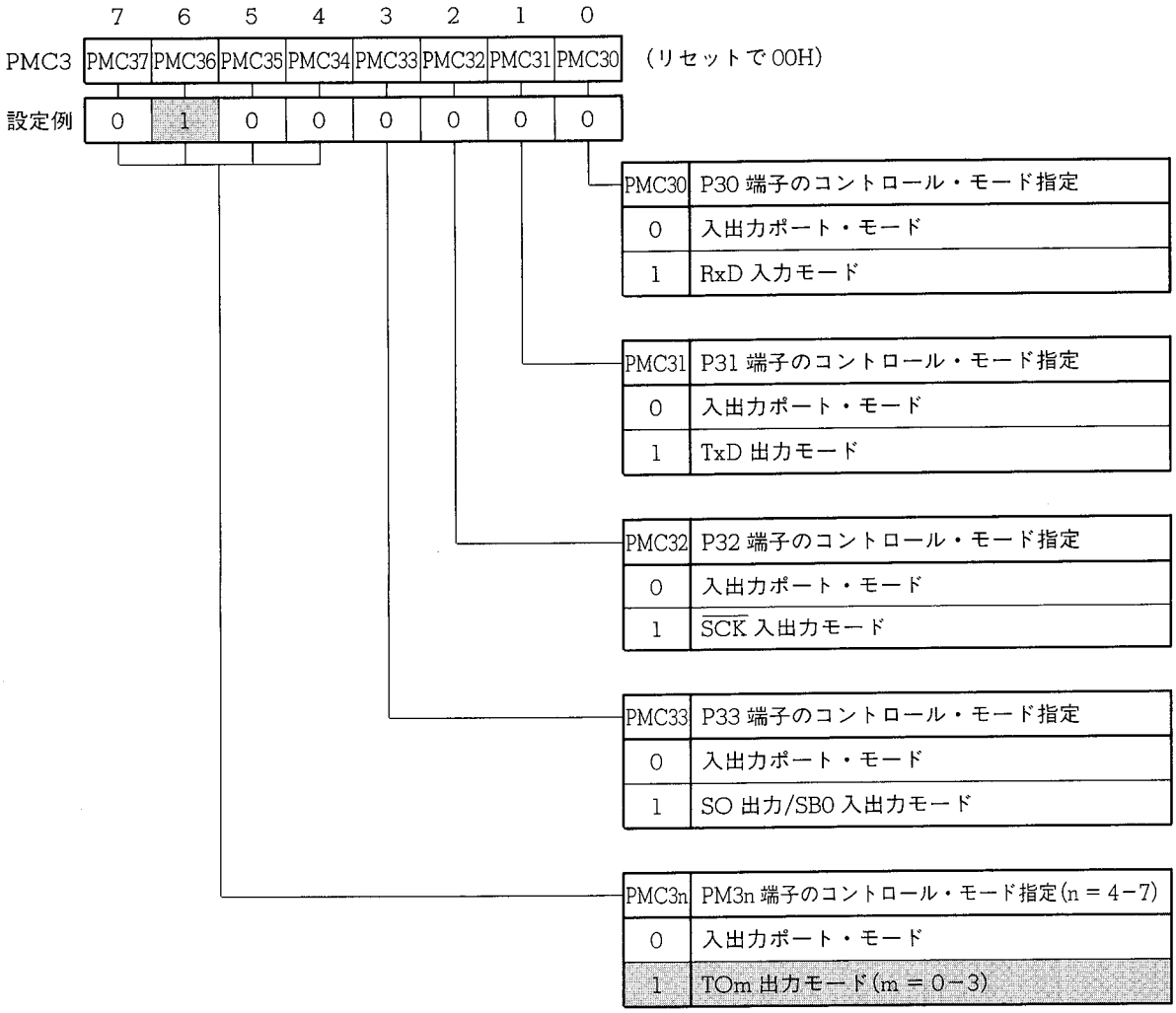
ENT02	TO2端子の動作指定
0	$\overline{\text{ALV2}}$を出力
1	パルス出力許可

ALV3	TO3端子のアクティブ・レベル指定
0	ロウ・レベル
1	ハイ・レベル

ENT03	TO3端子の動作指定
0	$\overline{\text{ALV3}}$を出力
1	パルス出力許可

3
214
218A
234
244

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

キャプチャ/コンペア・コントロール・レジスタ2

	7	6	5	4	3	2	1	0	
CRC2	MOD1	MOD0	CLR22	1	CLR21	0	0	0	（リセットで00H）
設定例	1	1	0	1	1	0	0	0	

MOD1	MOD0	CLR22	CLR21	タイマ出力モード		TM2のクリア動作
				TO2	TO3	
0	0	0	0	トグル出力	トグル出力	クリアしない
0	0	0	1	トグル出力	トグル出力	TM2とCR21レジスタの一致でクリア
0	0	1	0	トグル出力	トグル出力	TM2の内容をCR22レジスタへキャプチャ後にクリア
0	0	1	1	トグル出力	トグル出力	TM2とCR21レジスタの一致またはTM2の内容をCR22レジスタへキャプチャ後にクリア
0	1	0	0	PWM出力	トグル出力	クリアしない
1	0	0	0	PWM出力	PWM出力	クリアしない
1	1	0	1	PPG出力	トグル出力	TM2とCR21レジスタの一致でクリア

注意　上記以外の組み合わせは禁止

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	（リセットで00H）
設定例	0	1	0	0	0	0	0	0	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタの1カウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$注
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタの3カウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

注　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

割り込みマスク・レジスタH

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK1	（リセットでFFH）
設定例	×	×	×	×	0	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

タイマ・コントロール・レジスタ1

	7	6	5	4	3	2	1	0	
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0	（リセットで00H）
設定例	1	0	0	0	0	0	0	0	

8ビット・タイマ/カウンタ1

OVF1	タイマ/カウンタ1のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	タイマ/カウンタ1のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ2

CMD2	タイマ/カウンタ2の動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	タイマ/カウンタ2のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	タイマ/カウンタ2のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

3
214
218A
234
244

(d)　入力パラメータ

CYCL2　：TO2 端子からの PPG 出力の周期を決定する値です。

DUTY4　：デューティを決定する値を設定します。

(e)　使用レジスタ

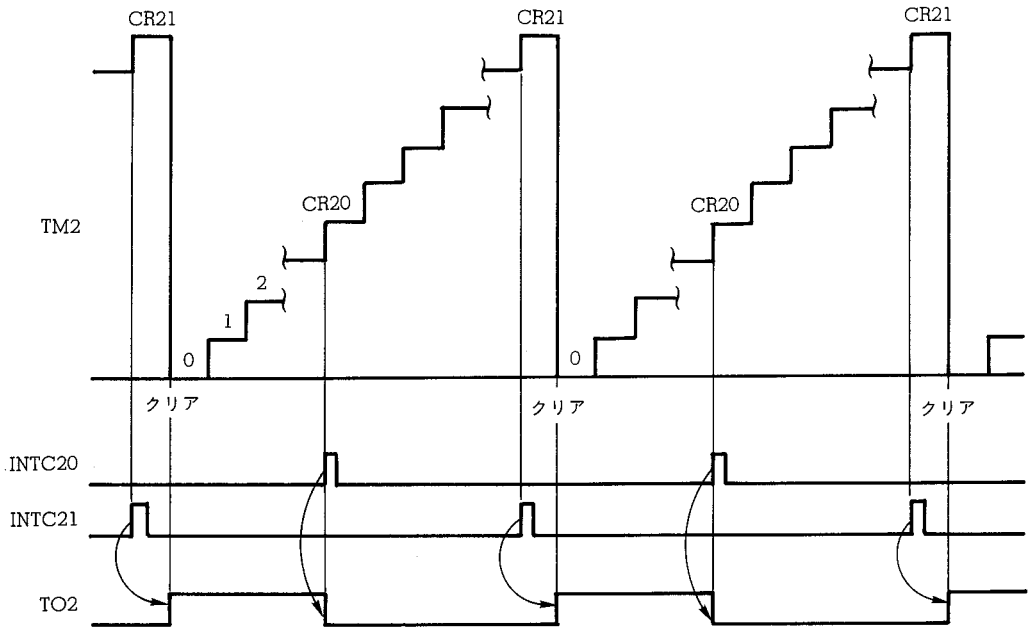
なし

(f)　プログラム使用例

TO2 端子からの PPG 出力と，デューティ変更要求処理の使用例を示します。周期は 750 Hz 固定とします。

備考　PPG 出力の場合，INTC20, INTC21 の発生タイミングと PPG 出力のタイミングは，図 3 - 20 のようになります。

図 3 - 20　PPG 出力タイミング（TM2）

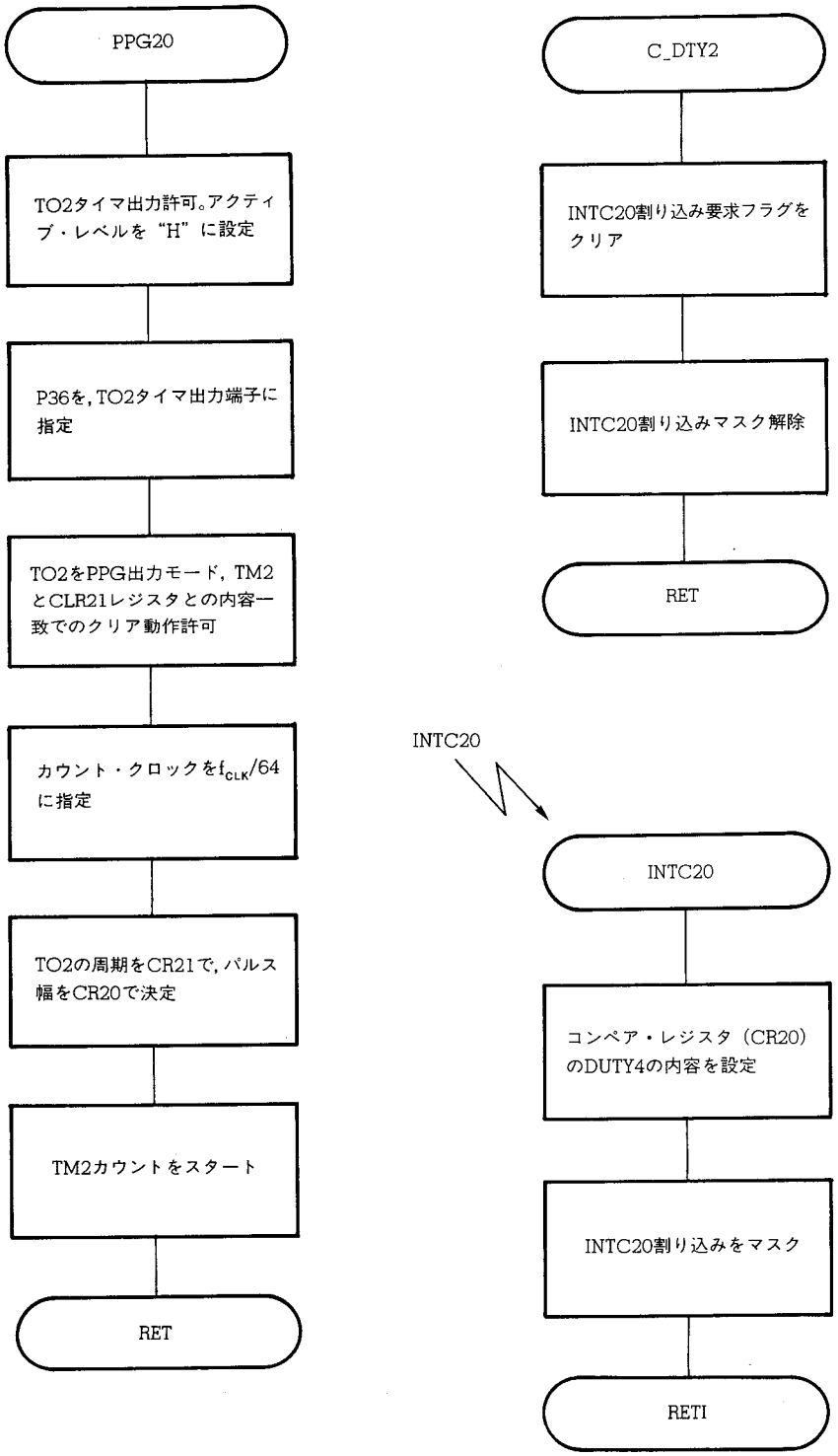


周期を決定する INTC21 発生後，タイマの 1 カウント分あとにタイマ出力が反転します。したがってコンペア・レジスタ（CR21）には，設定したい 1 周期の時間から計算される値－1 を設定します。

プログラム例として次のように用います。なおデューティ変更要求処理の使用例において，次のデューティを決定する値は，何らかの方法でＡレジスタにセットされているものとしています。

```
        PUBLIC  DUTY4,CYCL2
        EXTRN   C_DTY2,PPG20
                .
                .
CYCL2   EQU     124             ; PPG output cycle
                .
                .
DUTY4_D DSEG    SADDR
DUTY4:  DS      1               ; work area for duty
                .
                .
        CSEG
OUT20:
                .
                .
        MOV     DUTY4,#40H      ; set first duty
        CALL    !PPG20          ; PPG initialize routine
        EI
                .
                .
;       << デューティ変更要求 >>

C_DTY:
                .
                .
        MOV     DUTY4,A         ; set next duty
        CALL    !C_DTY2         ; change duty routine
                .
                .
```

(g)　フロー・チャート


PPG20
TO2タイマ出力許可。アクティブ・レベルを“H”に設定
P36を，TO2タイマ出力端子に指定
TO2をPPG出力モード，TM2とCLR21レジスタとの内容一致でのクリア動作許可
カウント・クロックをf_{CLK}/64に指定
TO2の周期をCR21で，パルス幅をCR20で決定
TM2カウントをスタート
RET
C_DTY2
INTC20割り込み要求フラグをクリア
INTC20割り込みマスク解除
RET
INTC20
INTC20
コンペア・レジスタ（CR20）のDUTY4の内容を設定
INTC20割り込みをマスク
RETI

(h)　プログラム・リスト

```
        NAME    PPG_2
;
;*************************************************************
;* 8bit-Timer / Counter-2                                    *
;*      PPG output                                           *
;*************************************************************
;
        PUBLIC  PPG20,C_DTY2
        EXTRN   CYCL2,DUTY4
;
CMK20   EQU     MK0H.3          ; INTC20 mask flag
CIF20   EQU     IF0H.3          ; INTC20 request flag
;
INTC20VT CSEG   AT 00012H
        DW      INTC20          ; INTC20 vector
;
        CSEG
PPG20:
        MOV     TOC,#00100000B  ; timer output,active level high
        MOV     PMC3,#01000000B ; P3 control port
        MOV     CRC2,#11011000B ; TM2 PPG mode
        MOV     PRM1,#01000000B ; TM2 prescaler fclk/64
        MOV     CR21,#LOW(CYCL2)  ; set cycle
        MOV     CR20,DUTY4      ; set first duty
        MOV     TMC1,#10000000B ; start timer
        RET
;
;       ***** request of change duty *****

C_DTY2:
        CLR1    CIF20           ; clear request flag of INTC20
        CLR1    CMK20           ; open mask of INTC20
        RET
;
;*************************************************************
;*              INTC20 interrupt routine                     *
;*************************************************************

INTC20:
        MOV     CR20,DUTY4      ; set next duty
        SET1    CMK20           ; mask INTC20
        RETI
;
        END
```


214
218A
234
244

3.5　ソフト・トリガド・ワンショット・パルス出力

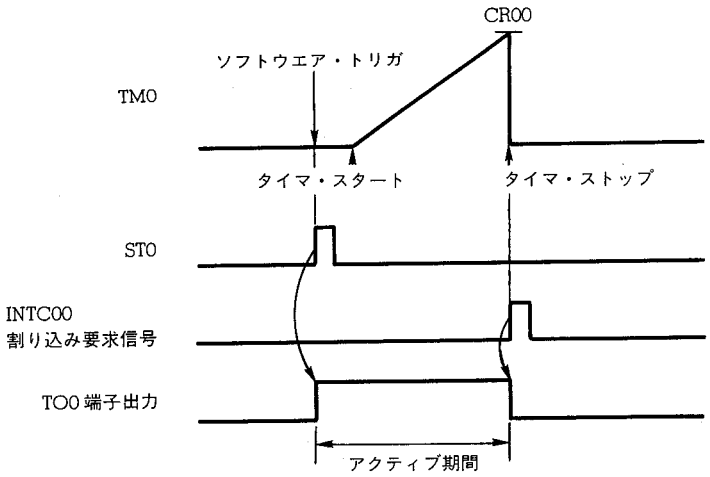
ソフト・トリガド・ワンショット・パルス出力機能は，16ビット・タイマ/カウンタ（TM0）が備えている機能で，ソフトウエアでトリガを設定し，ワンショット・パルスをTOn（n=0，1）端子から出力するモードです。ソフト・トリガド・ワンショット・パルス出力機能を備えている製品は次のとおりです。

- μPD78218Aシリーズ，μPD78234シリーズ，μPD78244シリーズ

(1) 動作概要

TO0端子からワンショット・パルスを出力する例を，図3-21のタイミング・チャートに示します。

図3-21　TO0端子からのワンショット・パルス出力例



備考　ALV0=1（アクティブ・ハイ）の場合

ソフトウエア・トリガ（OSPCレジスタのST0ビットをセットすること）によって，TO0端子からアクティブ・レベルが出力されます。その後，タイマ（TM0）のカウントを開始すると，TM0のカウント値とCR00の設定値が一致するまで，TO0はアクティブ・レベルを保持します。

TM0とCR00が一致すると，TO0は反転します（インアクティブ・レベル出力）。INTC00割り込み要求で，タイマ（TM0）のカウントを停止させ，その後，再びソフトウエア・トリガを設定するまで，TO0はインアクティブ・レベルを出力します。

(2) プログラム説明

(a) 処理概要（「(7) プログラム使用例」参照）

(i) 初期設定処理

ワンショット・パルス出力の初期設定，割り込みの許可を行います。

(ii) ソフトウエア・トリガ設定処理

ワンショット・パルス出力を許可し，タイマ（TM0）のカウントを開始します。

(iii) INTCOO 割り込み処理

タイマ（TM0）のカウントを停止します。

(b) 使用する RAM

なし

(3) 入出力パラメータ

なし

(4) 使用レジスタ

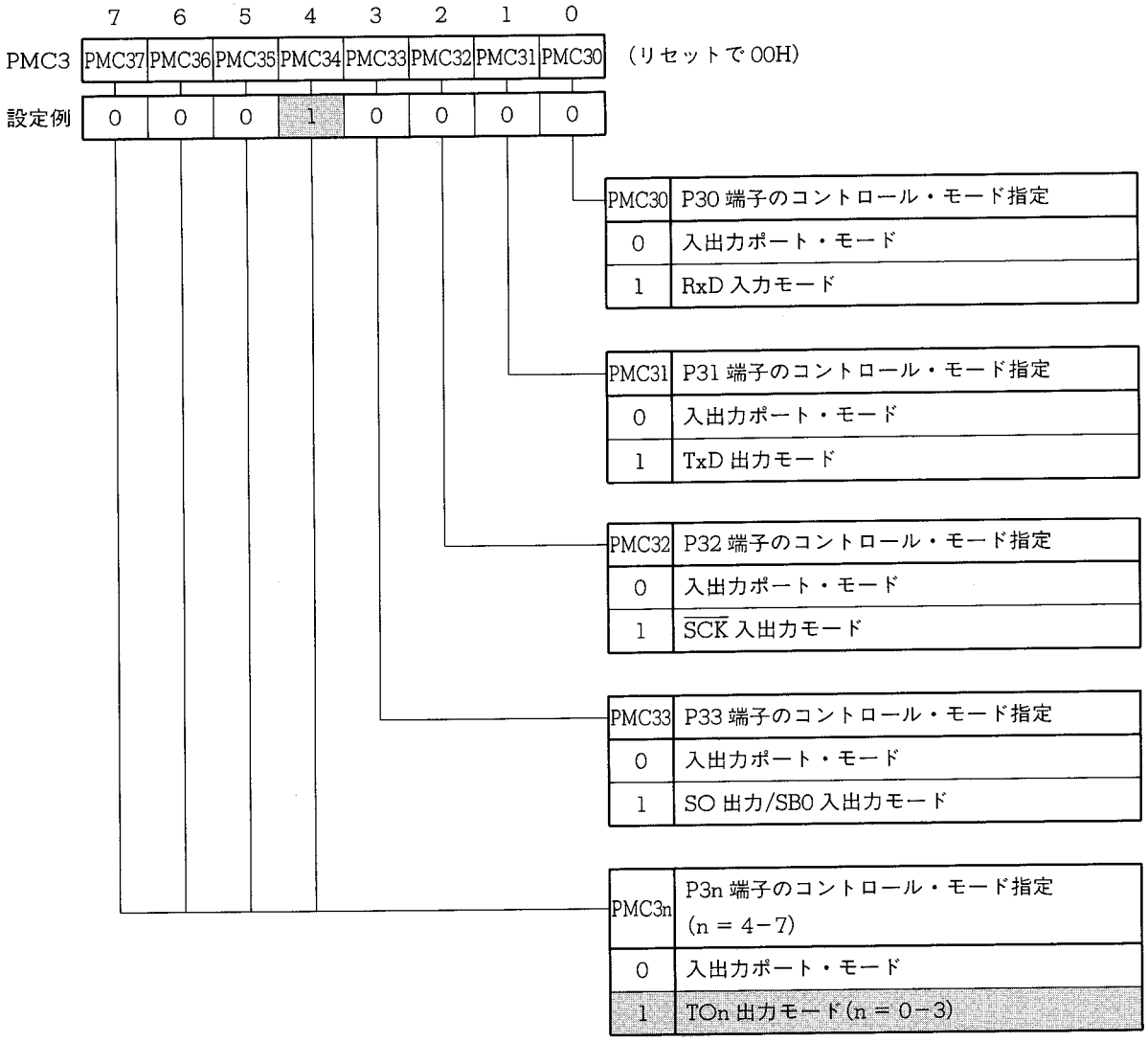
なし

(5) 使用スタック

なし

(6) モード・レジスタ設定例

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	P3n端子のコントロール・モード指定 (n = 4−7)
0	入出力ポート・モード
1	TOn出力モード(n = 0−3)

キャプチャ/コンペア・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
CRC0	MOD1	MOD0	0	1	CLR01	0	0	0	（リセットで10H）
設定例	0	0	0	1	0	0	0	0	

MOD1	MOD0	CLR01	タイマ出力モード指定		TM0＝CR01時のTM0のクリア動作
			TO0	TO1	
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

3
218A
234
244

ワンショット・パルス出力制御レジスタ

	7	6	5	4	3	2	1	0	
OSPC	ST1	RT1	0	OS1	ST0	RT0	0	OS0	（リセットで00H）
設定例	0	0	0	0	0	0	0	1	

OS0	パルス出力の種類選択
0	トグル出力，PWM出力，PPG出力を選択
1	ソフト・トリガド・ワンショット・パルス選択

ST0	RT0	TO0出力の制御
0	0	出力を変化させない
0	1	TO0にインアクティブ・レベルを出力
1	0	TO0にアクティブ・レベルを出力
1	1	設定禁止

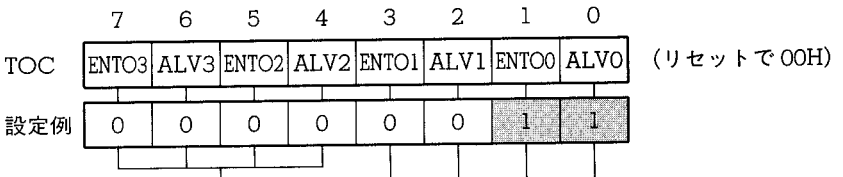
OS1	パルス出力の種類選択
0	トグル出力，PWM出力，PPG出力を選択
1	ソフト・トリガド・ワンショット・パルス出力

ST1	RT1	TO1出力の制御
0	0	出力を変化させない
0	1	TO1にインアクティブ・レベルを出力
1	0	TO1にアクティブ・レベルを出力
1	1	設定禁止

備考 1. RT0，ST0，RT1，ST1の各ビットは書き込みのみが可能で，読み出し時は“0”になります。

2. 端子からのパルス出力の禁止／許可およびアクティブ・レベルの指定は，タイマ出力制御レジスタ（TOC）で行います。

タイマ出力コントロール・レジスタ



	7	6	5	4	3	2	1	0	
TOC	ENTO3	ALV3	ENTO2	ALV2	ENTO1	ALV1	ENTO0	ALV0	（リセットで00H）
設定例	0	0	0	0	0	0	1	1	

ALV0	TO0端子のアクティブ・レベル	
	トグル出力指定時およびワンショット・パルス出力指定時	PWM/PPG出力指定時
0	ロウ・レベル	ハイ・レベル
1	ハイ・レベル	ロウ・レベル

ENTO0	TO0端子の動作指定
0	$\overline{ALV0}$を出力
1	パルス出力許可

ALV1	TO1端子のアクティブ・レベル	
	トグル出力指定時およびワンショット・パルス出力指定時	PWM/PPG出力指定時
0	ロウ・レベル	ハイ・レベル
1	ハイ・レベル	ロウ・レベル

ENTO1	TO1端子の動作指定
0	$\overline{ALV1}$を出力
1	パルス出力許可

8ビット・タイマ／カウンタ2によるタイマ出力(TO2, TO3端子出力）を制御します。

割り込みマスク・レジスタL

	7	6	5	4	3	2	1	0	
MK0L	CMK11	CMK10	CMK01	CMK00	PMK3	PMK2	PMK1	PMK0	（リセットでFFH）
設定例	×	×	×	0	×	×	×	×	

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

タイマ・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
TMC0	CE	0	0	0	CE0	OVF0	0	0	（リセットで00H）
設定例	0	0	0	0	注	0	0	0	

注　処理によって，0または1に設定

16ビット・タイマ/カウンタ

OVF0	TM0のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFFFHから0000Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE0	TM0のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ・ユニット3

CE	TM3のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

(7) プログラム使用例

TO0 端子から 500 μs のワンショット・パルスを出力する例を示します。

この場合，コンペア・レジスタ（CR00）への設定値は，次のようになります。

$$\frac{500\times10^{-6}}{8/(6\times10^{6})}=375$$

割り込み処理で，ソフトウエア・トリガを設定する場合，次のようなプログラムでワンショット・パルス出力を行うことができます。ただし，割り込みの初期設定（マスク解除など）は含んでいませんので，処理を追加する必要があります。

```
                :
                :
OSPWID  EQU     375                     ; one-shot pulse width

;       *** 初期設定 ***
                :
                :
        MOV     PMC3,#00010000B         ; set TO0 output mode
        MOV     CRC0,#00010000B         ; set TO0 timer out, disable clear TM0
        MOV     OSPC,#00000001B         ; set TO0 one-shot pulse output mode
        MOVW    CR00,#OSPWID            ; set one-shot pulse width
        MOV     TOC,#00000011B          ; set TO0 high active, enable output
        CLR1    CMK00                   ; open INTC00 mask
        EI                              ; enable interrupt
                :
                :

;       *** ソフトウエア・トリガ設定 ***
INT:
        SET1    ST0                     ; output active level from TO0
        MOV     TMC0,#00001000B         ; start TM0
                :
                :
        RETI
```

```
;       *** INTC00割り込み処理 ***
INTC00:
        MOV     TMC0,#00000000B         ; stop TM0
                :
                :
        RETI
                :
                :
```

3.6　パルス周期測定

ここでは，16ビット・タイマ/カウンタを用いたパルス周期測定の例を示します。

パルス周期測定は，外部割り込み要求（INTP3）入力端子に入力される外部パルスの周期を検出します。

INTP3端子に入力するパルス幅はハイ・レベル，ロウ・レベルとも12システム・クロック（2 μs：f_{CLK}＝6 MHz）以上必要で，これ以下の場合には有効エッジが検出されずキャプチャ動作を行いません。

このパルス周期測定では，分解能1.3 μsで2 μsから87.4 ms（f_{CLK}＝6 MHz）のパルス周期を測定できます。

(1) 動作概要

図3-22に示すようにINTP3端子入力の有効エッジ（ここでは立ち上がりエッジ）に同期して，カウント中の16ビット・タイマ・レジスタ（TM0）の値をキャプチャ・レジスタ（CR02）に取り込み保持します。パルス周期は，n回目の有効エッジ検出によりキャプチャ・レジスタ（CR02）に取り込み保持されたTM0カウント値(Dn)と，n－1回目の有効エッジ検出によるカウント値(D_{n-1})との差の値と，カウント・クロック（$8/f_{CLK}$）との積から求められます。

図3-22　パルス周期測定のブロック図

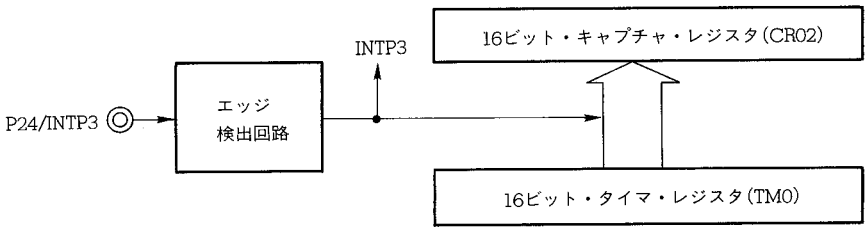
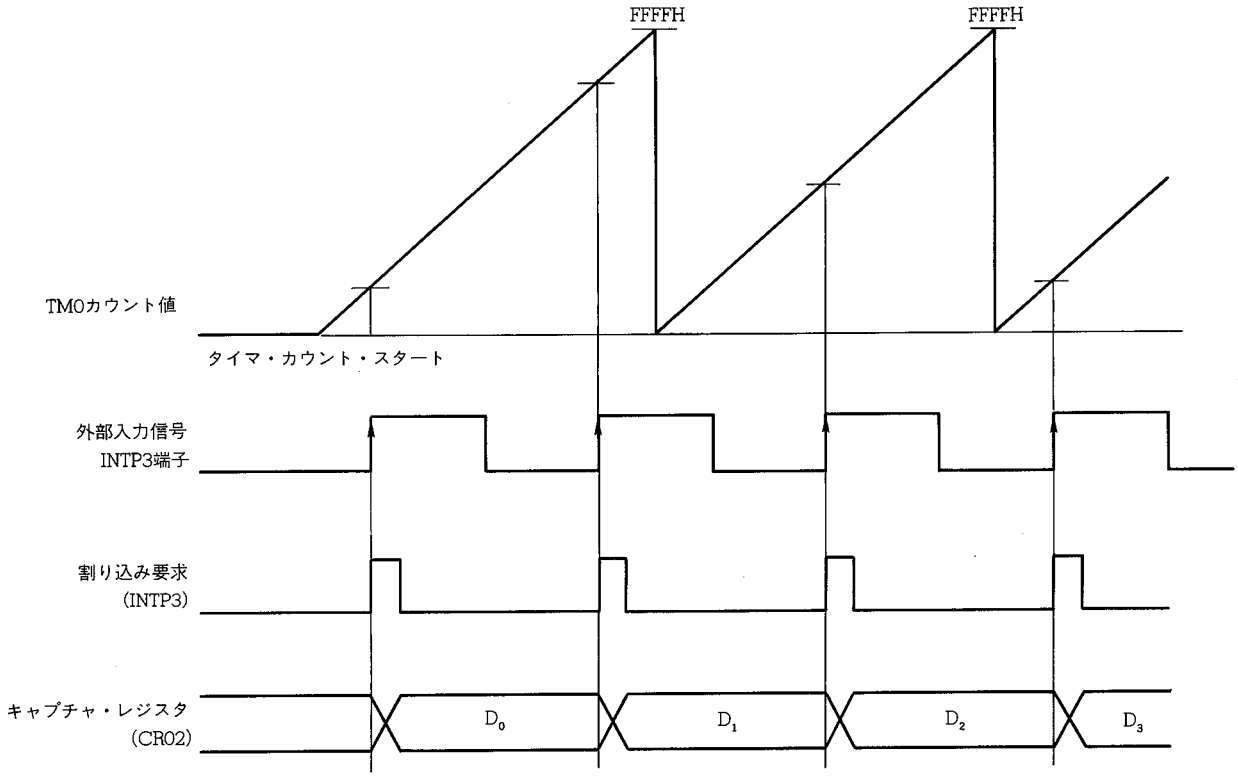


図3-23　INTP3入力のパルス周期測定のタイミング・チャート



パルス周期＝（$D_{n+1}-D_n$）×8/f_{CLK}

ここでは，INTP3端子への立ち上がりエッジを検出して，INTP3入力のパルス周期を測定します。ただし，オーバフローを考慮していませんので，測定範囲は2 μsから87.4 msです。

(2) プログラム説明……(7) プログラム・リスト参照

(a) フォアグランド処理［レーベル名称：**PULSE**］

(i) INTP3 入力の有効エッジを立ち上がりエッジとします。

(ii) 16 ビット・タイマ・レジスタ（TM0）と16 ビット・コンペア・レジスタ（CR01）の一致による，16 ビット・タイマ・レジスタ（TM0）のクリアを禁止します。

(iii) 前回のキャプチャ値保存用のワーク・エリア CAPWK および，パルス周期の算出結果のワーク・エリア WIDTH をクリアします。

(iv) 16 ビット・タイマ・レジスタ（TM0）のカウンタ動作を許可します。

(v) INTP3 割り込み要求のマスクを解除します。

(b) バックグランド処理［レーベル名称：**PLSANA**］

INTP3 割り込み要求によるベクタ割り込み処理です。

(i) BC レジスタに前回キャプチャした値を読み込みます。

(ii) AX レジスタにキャプチャ値を 16 ビット・キャプチャ・レジスタ(CR02)から読み込み，AX レジスタの値をキャプチャ値保存用の CAPWK に格納します。

(iii) AX レジスタの値と BC レジスタの値との差を取り，パルス周期の算出結果として WIDTH に格納します。

(3) モード・レジスタ設定例

外部割り込みモード・レジスタ 1

	7	6	5	4	3	2	1	0	
INTM1	0	0	ES51	ES50	ES41	ES40	ES31	ES30	(リセットで 00H)
設定例	0	0	0	0	0	0	0	1	

ES31	ES30	INTP3 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES41	ES40	INTP4 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	INTC30 を選択
1	1	設定禁止

ES51	ES50	INTP5 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	

キャプチャ/コンペア・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
CRC0	MOD1	MOD0	0	1	CLR01	0	0	0	（リセットで10H）
設定例	0	0	0	1	0	0	0	0	

MOD1	MOD0	CLR01	タイマ出力モード指定		TM0＝CR01時のTM0のクリア動作
			TO0	TO1	
0	0	0	トグル出力	トグル出力	禁　止
0	0	1	トグル出力	トグル出力	許　可
0	1	0	PWM出力	トグル出力	禁　止
0	1	1	設定禁止		
1	0	0	PWM出力	PWM出力	禁　止
1	0	1	設定禁止		
1	1	0	設定禁止		
1	1	1	PPG出力	トグル出力	許　可

タイマ・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
TMC0	CE	0	0	0	CE0	OVF0	0	0	(リセットで00H)
設定例	0	0	0	0	1	0	0	0	

16ビット・タイマ/カウンタ

OVF0	TM0のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFFFHから0000Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE0	TM0のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ・ユニット3

CE	TM3のカウント動作制御
0	クリアしたまま，カウンタ動作停止
1	カウント動作許可

割り込みマスク・レジスタL

	7	6	5	4	3	2	1	0	
MK0L	CMK11	CMK10	CMK01	CMK00	PMK3	PMK2	PMK1	PMK0	(リセットでFFH)
設定例	×	×	×	×	0	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

(4) 入出力パラメータ

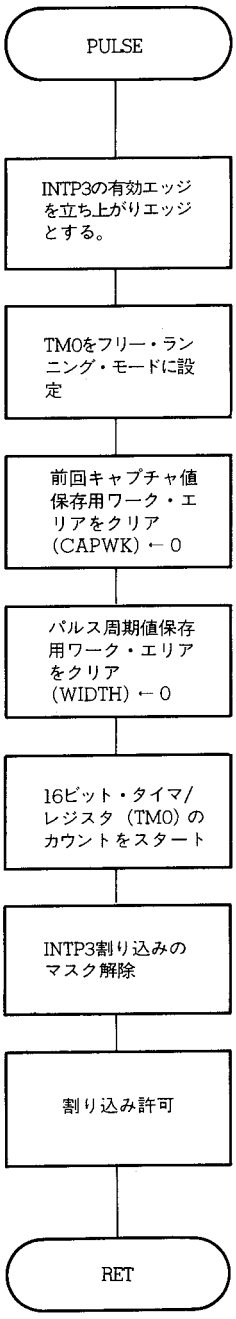
WIDTH：パルス周期の16進値が生成されます。

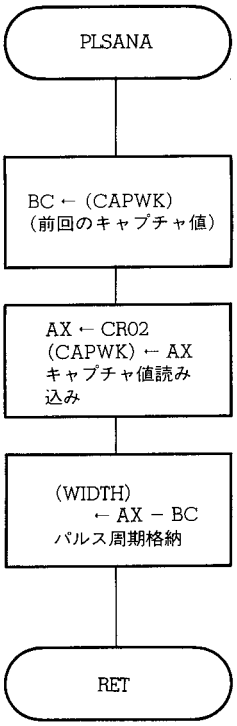
ワーク・エリアとして用いるCAPWKおよびWIDTHはショート・ダイレクト・アドレシング特有の命令でアクセスしているため，ショート・ダイレクト・アドレシングの適用範囲(FE20H-FEF7H)に配置してください。

(5) 使用レジスタ

なし

(6) フロー・チャート


PULSE
INTP3の有効エッジを立ち上がりエッジとする。
TM0をフリー・ランニング・モードに設定
前回キャプチャ値保存用ワーク・エリアをクリア
(CAPWK) ← 0
パルス周期値保存用ワーク・エリアをクリア
(WIDTH) ← 0
16ビット・タイマ/レジスタ (TM0) のカウントをスタート
INTP3割り込みのマスク解除
割り込み許可
RET


PLSANA
BC ← (CAPWK)
(前回のキャプチャ値)
AX ← CR02
(CAPWK) ← AX
キャプチャ値読み込み
(WIDTH)
← AX − BC
パルス周期格納
RET

(7) プログラム・リスト

```
        NAME    PULSEM
;
;*************************************************************
;* 16bit-Timer / Counter Unit                                *
;*      measure pulse cycle                                  *
;*************************************************************
;
        PUBLIC  PULSE,PLSANA
        EXTRN   CAPWK,WIDTH     ; work area

PMK3    EQU     MKOL.3          ; INTP3 mask flag
;
        CSEG
PULSE:
        MOV     INTM1,#00000001B
                                ; INTP3's enable edge is rise
        MOV     CRC0,#00010000B ; clear disable TM0 by CR01
        MOVW    CAPWK,#0        ; clear result
        MOVW    WIDTH,#0


        MOV     TMC0,#00001000B ; timer start
        CLR1    PMK3            ; open INTP3 mask
        EI                      ; interrupt enable

        RET
;
PLSANA:
        PUSH    BC              ; save register
        PUSH    AX
        MOVW    AX,CAPWK        ; read last caputure data
        MOVW    BC,AX
        MOVW    AX,CR02         ; caputure read
        MOVW    CAPWK,AX        ; save caputure data
        SUBW    AX,BC
        MOVW    WIDTH,AX
        POP     AX              ; restore data
        POP     BC

        RET
;
        END
```

第4章　PWM出力ユニットのプログラム例（μPD78234）

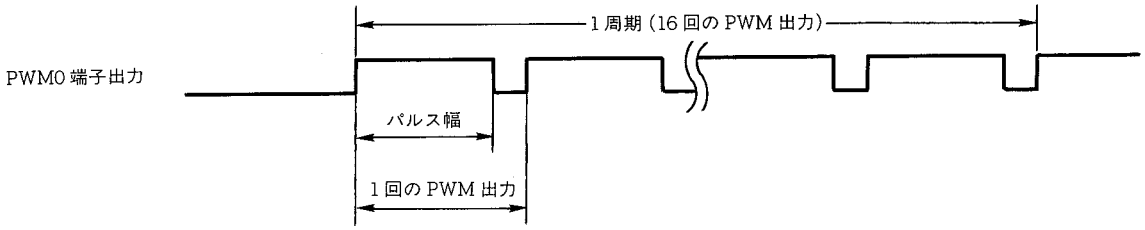
PWM出力ユニットは，μPD78234シリーズが備えているハードウエアです。PWM0，PWM1の2チャネルから12ビット分解能のPWM出力を行うことができます。

一般的にPWM出力信号は，ロウ・パス・フィルタを通して波形整形し，電圧制御に使用します。

(1) 動作概要

PWM0端子からPWM出力を行う例を，図4-1のタイミング・チャートに示します。

図4-1　PWM0端子からのPWM出力のタイミング・チャート

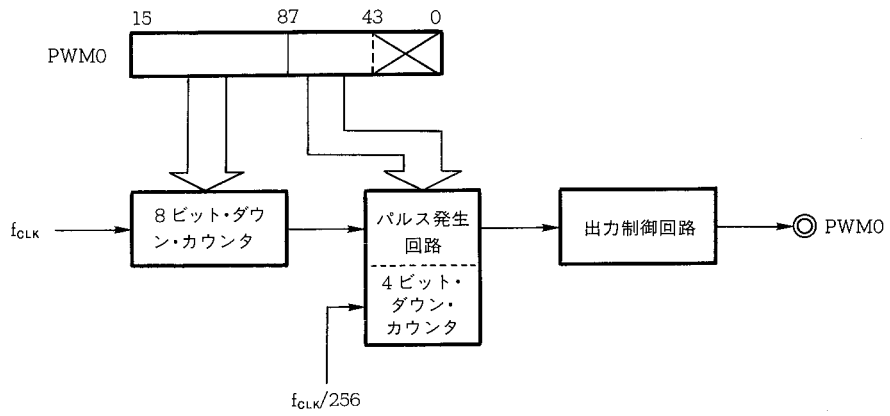


備考　ALV0＝1（アクティブ・ハイ）の場合

PWM出力のパルス幅は，PWMモジュロ・レジスタ（PWM0）に値（16ビット）を設定することによって決定します。

また，図4-2に示すように，8ビットのPWM信号（8ビット・ダウン・カウンタでカウント）を16回（4ビット・ダウン・カウンタでカウント）出力することで，12ビットの分解能を実現していますので，PWM出力の1周期は，16回のPWM出力の軸となります。

図4-2　PWM出力機能のブロック図



PWM出力の1周期の幅は次のようになりますので，この周期でPWM出力のパルス幅が切り替わることになります。

$1/f_{CLK} \times 2^{12}$ (4096) ≒ 682.7 μs (f_{CLK} = 6 MHz)

(2) プログラム説明

(a) 処理概要（「(7) プログラム使用例」参照）

(i) PWM出力の初期パルス幅を設定し，PWM出力を許可します。

(ii) PWMパルス幅を変更します。次にモジュロ・レジスタ(PWM0)の設定値を書き換えるまで，このパルスを出力し続けます。

(b) 使用するRAM

なし

(3) 入出力パラメータ

なし

(4) 使用レジスタ

なし

(5) 使用スタック

なし

(6) モード・レジスタ設定例

PWMコントロール・レジスタ

	7	6	5	4	3	2	1	0	
PWMC	SYN1	0	SYN0	0	EN1	ALV1	EN0	ALV0	（リセットで05H）
設定例	0	0	0	0	0	1	1	1	

(n=0, 1)

ALVn	PWMn端子のPWMアクティブ・レベルの設定
0	ロウ・アクティブ
1	ハイ・アクティブ

ENn	PWMn端子のPWM出力制御
0	出力禁止。端子レベルはPM1, P1, PUOレジスタの内容で決定（ポート・モード）
1	PWM出力許可

SYNn	PWMパルス幅書き換え周期の指定
0	PWM16サイクル（$2^{12}/f_{CLK}$）ごとに書き換える
1	PWM1サイクル（$2^{8}/f_{CLK}$）ごとに書き換える

(7) プログラム使用例

PWM0端子から，デューティ80％のPWM信号を出力する例を示します。

この場合，PWMモジュロ・レジスタ（PWM0）への設定値は，次のようになります。

$$(4096\times0.8)-1=3275.8$$
$$\fallingdotseq\underline{3276}(CCCH)$$

次のようなプログラムで，デューティ80％のPWM出力を行うことができます。

```
            :
            :
MOVW    PWM0,#0000H             ; set PWM0 duty 0%
MOV     PWMC,#00000111B         ; high active, 16 cycle, enable output
MOVW    PWM0,#0CCC0H            ; set PWM0 duty 80%
            :
            :
```

注意　モジュロ・レジスタ（PWM0, PWM1）への設定値は，16ビット・データ（下位4ビットは無視されます）としてください。

第５章　アシンクロナス・シリアル・インタフェースのプログラム例

78K/II シリーズのアシンクロナス・シリアル・インタフェース（以後 UART）には，ボー・レートの設定方法が３種類あり，デバイス品種によって設定方法が異なります。表５－１に品種ごとのボー・レートの設定方法を示します。

表５－１　78K/II の UART のボー・レート設定方法

設定方法		デバイス品種	
		μPD78214 シリーズ μPD78218A シリーズ μPD78234 シリーズ μPD78244 シリーズ	μPD78224 シリーズ
８ビット・タイマ/カウンタ３（内部ボー・レート・ジェネレータ注）		○	○
ボー・レート・ジェネレータ	ボー・レート・ジェネレータの入力クロック	○	×
	ボー・レート・ジェネレータの入力クロック＋分周カウンタ	○	×

注　μPD78224 の場合

以上の３種類のボー・レート設定方法を用いた UART のプログラム例を示します。

本文中の説明では，(a)　μPD78214 シリーズ，78218A シリーズ，78234 シリーズ，78244 シリーズ，(b)　μPD78224 シリーズのように，項目ごとに特有の説明になっています。なお，μPD78214 シリーズ，78218A シリーズ，78234 シリーズ，78244 シリーズでは，(a)，(b) 両方のプログラムが使用できます。

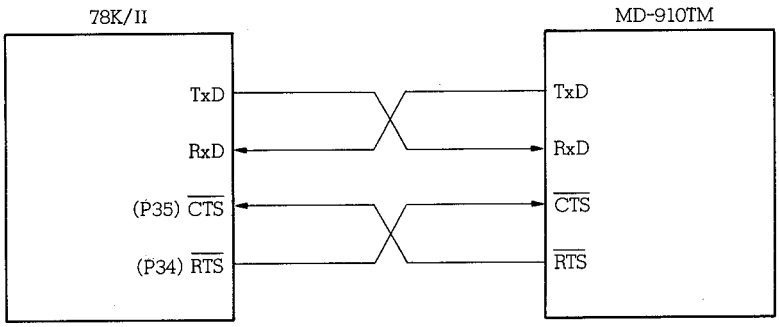
プログラム例として，キャラクタ・ディスプレイ・ターミナル（MD-910TM）との接続例を示します。

MD-910TM のキー・ボードより入力したキャラクタを受信し，このデータを MD-910TM に送信することにより，MD-910TM のディスプレイに表示させます。

5.1　動作概要

図5-1に78K/IIシリーズとMD-910TMとの接続を示します。ソフトウエアの動作としては，ループ・バック動作となります。データ・ラインとハンドシェーク・ラインはそれぞれTxD, RxD, $\overline{CTS}$, $\overline{RTS}$とし，互いにクロス接続とします。

図5-1　78K/IIシリーズとMD-910TMとの接続



ボー・レートを除く仕様を表5-2に示します。

表5-2　UARTのプログラム例の仕様

項　目	仕　様
ストップ・ビット	2ビット
キャラクタ長	8ビット
パリティ・ビット	なし
$\overline{CTS}$ 端子	P35
$\overline{RTS}$ 端子	P34

このプログラムでは次の3つのワーク・エリアを使用します。ワーク・エリアは，ショート・ダイレクト・アドレシングの適用範囲に配置してください。

表5-3　UARTのプログラム例で用いるワーク・エリア

ワーク・エリア名称	用　　途
RCV_DT	受信データ用ワーク・エリア
TRN_DT	送信データ用ワーク・エリア

表5-4　UARTのプログラム例で用いるフラグ

フラグ名称	用　　途
RCVFLG	受信完了フラグ

5.2　プログラム説明

(1) イニシャライズ処理［レーベル名称：ASY214，ASY220］

(i) P30，P31をコントロール・ポート（RxD，TxD）として使用します。

(ii) $\overline{\text{RTS}}$ =1として，送信要求を解除します。この場合MD-910TMからの送信は行われません。

(iii) $\overline{\text{CTS}}$端子として使用するP35を入力ポート，$\overline{\text{RTS}}$ 端子として使用するP34を出力ポートに設定します。

(iv) アシンクロナス・シリアル・インタフェース・モード・レジスタをイニシャライズします。

(v) シリアル・クロック（ボー・レート）の設定を行います。このプログラム例では，デバイス品種（μPD78214，μPD78220）ごとに異なった方法で行っています。

(vi) 送信処理において前回の送信が終了したことをSTIF（UART送信終了）割り込み要求フラグでチェックしているため，あらかじめSTIF割り込み要求フラグをセット（1）しておきます。

78K/IIのUARTではシフト・レジスタ（TXS）が空という要因で，STIFフラグがセット（1）されることはありません。常にシフト・レジスタ（TXS）に書き込んだデータの送信が終了することによってのみSTIFフラグがセット（1）されます。

(vii) INTSR（UART受信終了）割り込み要求のマスクを解除します。

(viii) 割り込みを許可します。

(ix) $\overline{\text{RTS}}$ = 0として送信要求を行います。

(2) 受信処理［レーベル名称：RECIV］

(i) $\overline{\text{RTS}}$ = 1として，送信要求を解除します。

(ii) Aレジスタに受信データを受信バッファ（RXB）から読み込みます。

(iii) メモリに受信データを保存します。

(iv) 受信完了フラグ（RCVFLG）をセット（1）します。

備考　78K/IIのUARTにおいて，受信時に受信エラーが発生した場合には，INTSR（UART受信終了）フラグとINTSER（UART受信エラー）フラグが同時にセット（1）されます。このとき，双方ともベクタ割り込みで処理をする場合には，INTSERの方がディフォールト・プライオリティが高いために，先に受け付けられます。

(3) 送信要求処理［レーベル名称：CLRRTS］

(i) $\overline{\text{RTS}}$ = 0としてMD-910TMに対して送信要求を行います。

(4) 送信処理［レーベル名称：TRANS］

(i) 前回の送信が終了したことをSTIFフラグでチェックしています。STIF＝1となるまでウエイトします。STIF＝1の場合，クリアして，次の$\overline{CTS}$のチェックを行います。

(ii) MD-910TMが受信可能かをチェックします。$\overline{CTS}$＝0になるまでウエイトします。

(iii) シフト・レジスタ（TXS）に送信データを書き込みます。

5.3　モード・レジスタ設定例

(a)　μPD78214のプログラム

アシンクロナス・シリアル・インタフェース・モード・レジスタ

	7	6	5	4	3	2	1	0	
ASIM	1	RXE	PS1	PS0	CL	SL	0	SCK	(リセットで80H)
設定例	1	1	0	0	1	1	0	1	

SCK	シリアル・クロックの指定
0	タイマ/カウンタ3出力
1	ボー・レート・ジェネレータ出力

SL	ストップ・ビット長
0	1ビット
1	2ビット

CL	データのキャラクタ長
0	7ビット
1	8ビット

PS0	PS1	パリティ・ビットの指定
0	0	パリティなし
0	1	送信＝0パリティ付加 受信＝パリティ・エラーを発生しない
1	0	奇数パリティ
1	1	偶数パリティ

RXE	受信許可制御
0	受信禁止状態
1	受信許可状態

ボー・レート・ジェネレータ・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
BRGC	CE	TPS2	TPS1	TPS0	MDL3	MDL2	MDL1	MDL0	（リセットで00H）
設定例	1	0	1	0	0	1	0	0	

4ビット・モジュロ・レジスタへの設定値m（1−15）

TPS2	TPS1	TPS0	分周回路のタップn
0	0	0	1/2
0	0	1	1/4
0	1	0	1/8
0	1	1	1/16
1	0	0	1/32
1	0	1	1/64
1	1	0	1/128
1	1	1	1/256

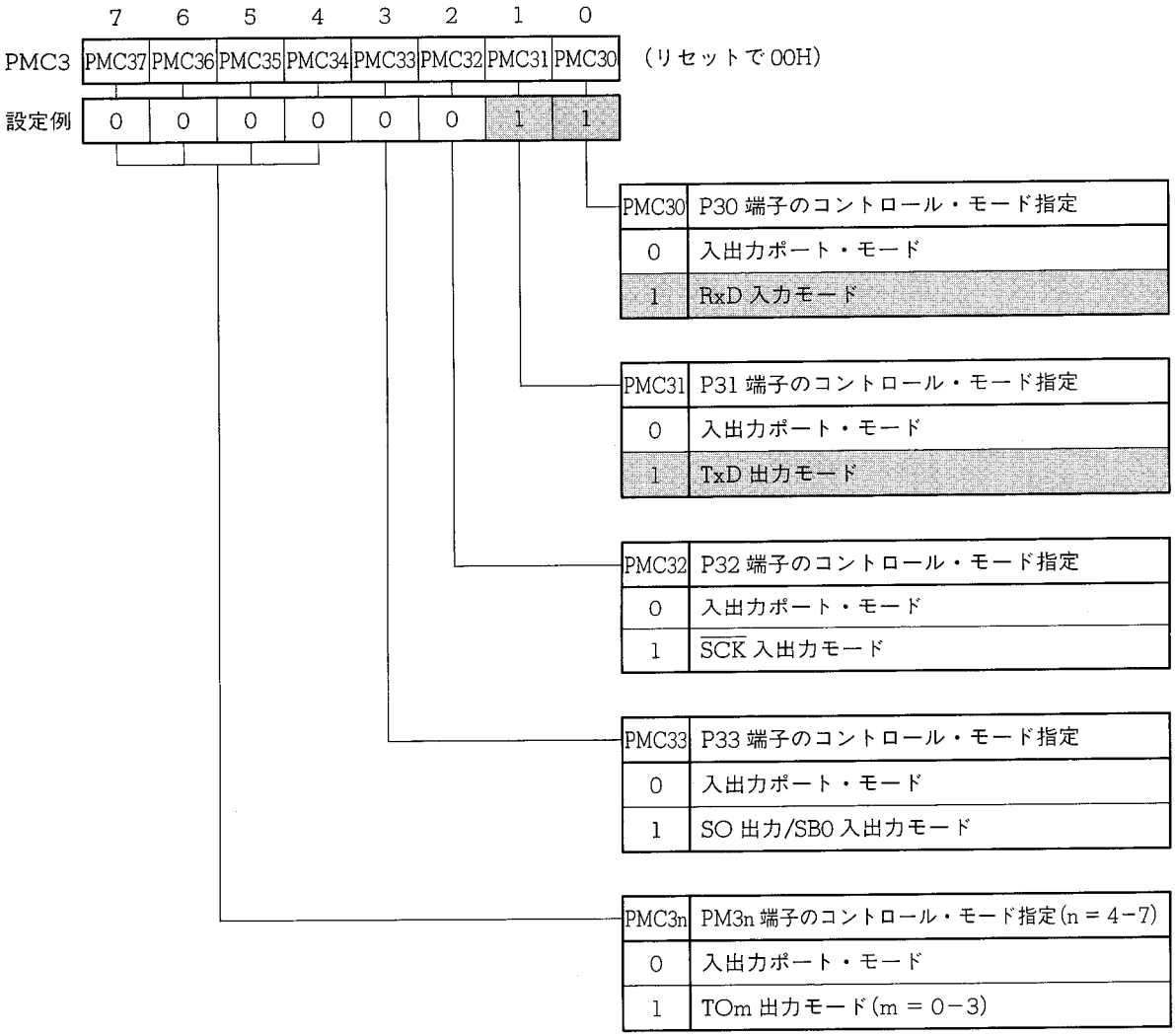
TPS2	4ビット・カウンタおよび分周回路の動作
1	停止
0	カウント動作

$$ボー・レート = \frac{f_{xx}}{2} \times \frac{1}{m+1} \times \frac{1}{n} \times \frac{1}{16}$$

f_{xx}：システム・クロック周波数

5
214
218A
234
244

ポート3モード・コントロール・レジスタ



	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	（リセットで00H）
設定例	0	0	0	0	0	0	1	1	

PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

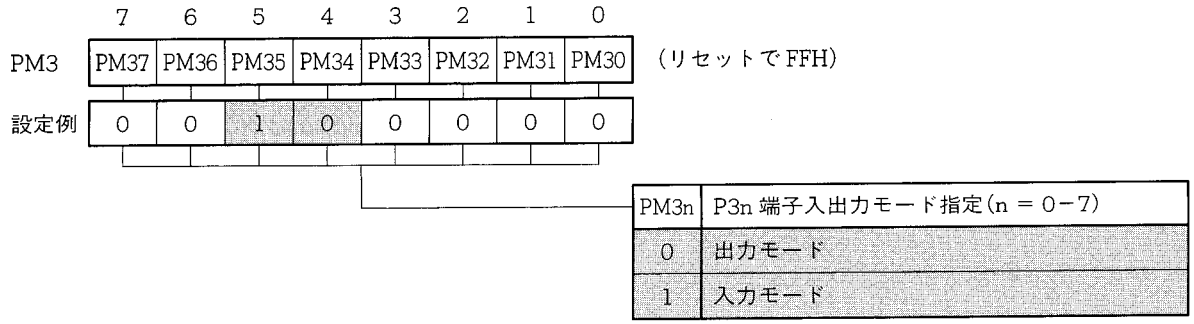
PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

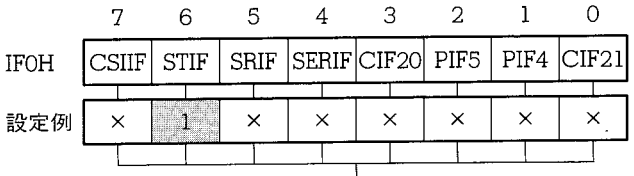
ポート3モード・レジスタ



	7	6	5	4	3	2	1	0	
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	（リセットでFFH）
設定例	0	0	1	0	0	0	0	0	

PM3n	P3n端子入出力モード指定(n = 0−7)
0	出力モード
1	入力モード

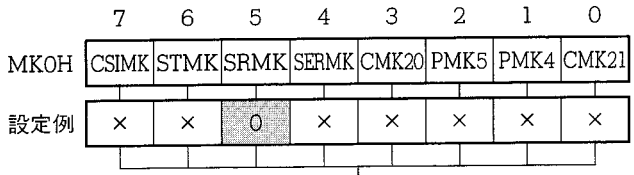
割り込み要求フラグ・レジスタH



	7	6	5	4	3	2	1	0	
IF0H	CSIIF	STIF	SRIF	SERIF	CIF20	PIF5	PIF4	CIF21	（リセットで00H）
設定例	×	1	×	×	×	×	×	×	×：操作しません

IF	割り込み要求フラグ
0	割り込み要求が発生していない
1	割り込み要求発生

割り込みマスク・レジスタH



	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	（リセットでFFH）
設定例	×	×	0	×	×	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

5
214
218A
234
244

(b)　μPD78224のプログラム

アシンクロナス・シリアル・インタフェース・モード・レジスタ

	7	6	5	4	3	2	1	0	
ASIM	1	RXE	PS1	PS0	CL	SL	0	SCK	（リセットで80H）
設定例	1	1	0	0	1	1	0	0	

SCK	シリアル・クロックの指定
0	内部ボー・レート・ジェネレータ出力
1	内部クロック　$f_{CLK}/8$注

SL	ストップ・ビット長
0	1ビット
1	2ビット

CL	データのキャラクタ長
0	7ビット
1	8ビット

PS0	PS1	パリティ・ビットの指定
0	0	パリティなし
0	1	送信＝0パリティ付加 受信＝パリティ・エラーを発生しない
1	0	奇数パリティ
1	1	偶数パリティ

RXE	受信許可制御
0	受信禁止状態
1	受信許可状態

注　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

プリスケーラ・モード・レジスタ0

	7	6	5	4	3	2	1	0
PRM0	PRS3	PRS2	PRS1	PRS0	0	0	0	0
設定例	0	0	1	0	0	0	0	0

8ビット・タイマ/カウンタ3[注]

PRS3	−	PRS1	PRS0	−	μPD78224
PRS33	PRS32	PRS31	PRS30	その他	−
0	0	0	0	$f_{CLK}/8$	$f_{CLK}/8$
0	0	0	1		
0	0	1	0	$f_{CLK}/16$	$f_{CLK}/16$
0	0	1	1	$f_{CLK}/32$	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$	設定禁止
0	1	0	1	$f_{CLK}/128$	
0	1	1	0	$f_{CLK}/256$	
0	1	1	1	$f_{CLK}/512$	
1	0	0	0	設定禁止	高速転送モード
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	×	×	設定禁止	

注　μPD78224では内部ボー・レート・ジェネレータ

備考　f_{CLK}：内部システム・クロック

タイマ・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
TMC0	CE3[注1]	0	0	0	CE0	OVF0	0	0	（リセットで00H）
設定例	1	0	0	0	0	0	0	0	

16ビット・タイマ/カウンタ

OVF0	タイマ/カウンタ0のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFFFHから0000Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE0	タイマ/カウンタ0のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

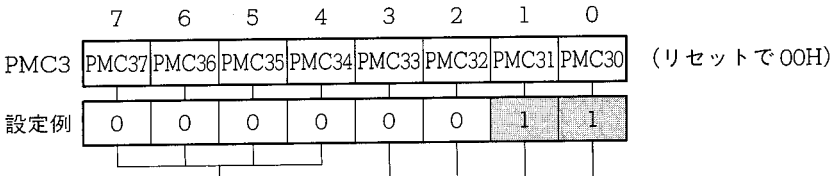
8ビット・タイマ/カウンタ3[注2]

CE3	タイマ/カウンタ3のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

注1. μPD78224ではCE

2. μPD78224では内部ボー・レート・ジェネレータ

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SO出力/SB0入出力モード

PMC3n	PM3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

ポート3モード・レジスタ

	7	6	5	4	3	2	1	0	
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	（リセットでFFH）
設定例	0	0	1	0	0	0	0	0	

PM3n	P3n端子入出力モード指定(n = 0−7)
0	出力モード
1	入力モード

割り込み要求フラグ・レジスタH

	7	6	5	4	3	2	1	0	
IF0H	CSIIF	STIF	SRIF	SERIF	CIF20	PIF5	PIF4	CIF21	（リセットで00H）
設定例	×	1	×	×	×	×	×	×	×：操作しません

IF	割り込み要求フラグ
0	割り込み要求が発生していない
1	割り込み要求発生

割り込みマスク・レジスタH

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	（リセットでFFH）
設定例	×	×	0	×	×	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

(1) 両者共通パラメータ

入力

TRN_DT　：送信データです。送信データをこのメモリに設定して［TRANS］を呼び出してください。

出力

RCV_DT　：受信データはこのメモリに格納されます。

RCVFLG　：受信完了フラグです。INTSR 割り込み処理により 1 バイト受信するごとにセット（1）されるフラグです。

(2) デバイスごとに異なる入力パラメータ

(a) μPD78214 のプログラム

BRGCD　：ボー・レート・ジェネレータ・コントロール・レジスタ（BRGCD）に設定する値です。ボー・レート・ジェネレータの入力クロックとその分周比を設定します。

ボー・レートの計算方法は，次のとおりです。

$$\text{ボー・レート} = \frac{\text{ボー・レート・ジェネレータのカウント・クロック}}{k+1} \times \frac{1}{n} \times \frac{1}{16}$$

k　：BRGC レジスタの MDL3-MDL0 ビットの設定値

1/n　：分周回路のタップ

BRGCD の値は，k，n の値によって決定します。

5
214
218A
224
234
244

(b) μPD78224 のプログラム

PRM0D　：プリスケーラ・モード・レジスタ（PRM0）に設定する値です。内部ボー・レート・ジェネレータのカウント・クロックを設定します。

CR30D　：8ビット・コンペア・レジスタ（CR30）に設定する値です。

ボー・レートの計算方法は，次のとおりです。

$$\text{ボー・レート} = \frac{\text{内部ボー・レート・ジェネレータのカウント・クロック}}{\text{CR30} + 1} \times \frac{1}{2} \times \frac{1}{16}$$

CR30の値は次式で求められます。

$$\text{CR30} = \frac{\text{カウント・クロック}}{\text{ボー・レート}} \times \frac{1}{2} \times \frac{1}{16}$$

(3) 使用レジスタ

(i) ASY214, ASY210,　：なし

(ii) RECIV　：Aレジスタ

(iii) CLRRTS　：なし

(iv) TRANS　：Aレジスタ

(4) プログラム使用例

前述のプログラムではUARTのイニシャライズ処理，送受信処理についてふれています。ここで，ボー・レートを実際に設定した場合のプログラム例とループ・バック処理の呼び出しを行ったプログラム例について示します。

(a) μPD78214 のプログラム

MD-910TMとの通信を9600 bps（f_{CLK}=6 MHz）で行います。

$$9600 = \frac{6 \times 10^6}{k + 1} \times \frac{1}{n} \times \frac{1}{16}$$

上記の計算式よりk = 9, n = 4となり，ボー・レート・ジェネレータのカウント・クロックはf_{CLK}/10，分周回路のタップは1/4となります。したがって，入力パラメータBRGCDへの設定値は'10100100B'となります。

この場合に作成されるボー・レートは，実際には，次のようになりますので，9600 bpsに対して −2.3％の誤差が生じることになります。

$$\text{ボー・レート} = \frac{6 \times 10^6}{9 + 1} \times \frac{1}{4} \times \frac{1}{16} = 9375 \text{ (bps)}$$

プログラム例として次のように用います。

```
        PUBLIC  TRN_DT,RCV_DT                           ; DATA WORK
        PUBLIC  RCVFLG                                  ; FLAG
        PUBLIC  BRGCD

        EXTRN   ASY214                                  ; PACKAGE
        EXTRN   TRANS,RECIV,CLRRTS                      ; PACKAGE
                .
                .
;       BAUD RATE :             9600bps
;       STOP BIT :              2bit
;       CHARACTER LENGTH :      8bit
;       PARITY :                NO PARITY

BRGCD   EQU     10100100B       ; BRGC DATA (9600bps,fclk/5)

        BSEG
RCVFLG  DBIT                    ; RECEIVE FLAG

ASY14_D DSEG    SADDR
TRN_DT: DS      1               ; TRANSMIT DATA
RCV_DT: DS      1               ; RECEIVE DATA

INTSRVT CSEG    AT 00022H
        DW      INTSR           ; INTSR
                .
                .
        CALL    !ASY214         ; <<< ASY214 >>>
SER_L1:
        BTCLR   RCVFLG,$SER_J1  ; CHECK RECEIVE FLAG
        BR      $SER_L1
SER_J1:
        MOV     TRN_DT,RCV_DT   ; TRANS DATA
        CALL    !TRANS          ; <<< TRANS >>>
        CALL    !CLRRTS         ; <<< CLRRTS >>>
        BR      SER_L1

INTSR:
        CALL    !RECIV          ; <<< RECIV >>>
        RETI
```

(b)　μPD78224 のプログラム

MD-910TM との通信を 4800 bps(f_{CLK}= 5 MHz)で行います。内部ボー・レート・ジェネレータのカウント・クロックを $f_{CLK}/16$ に指定した場合，入力パラメータ CR30D に設定する値は次式のようになります。なおこの方法は，μPD78214 においても可能です。

$$CR30D = \frac{5\times10^6/16}{4800} \times \frac{1}{16} \times \frac{1}{2} - 1 \fallingdotseq 1$$

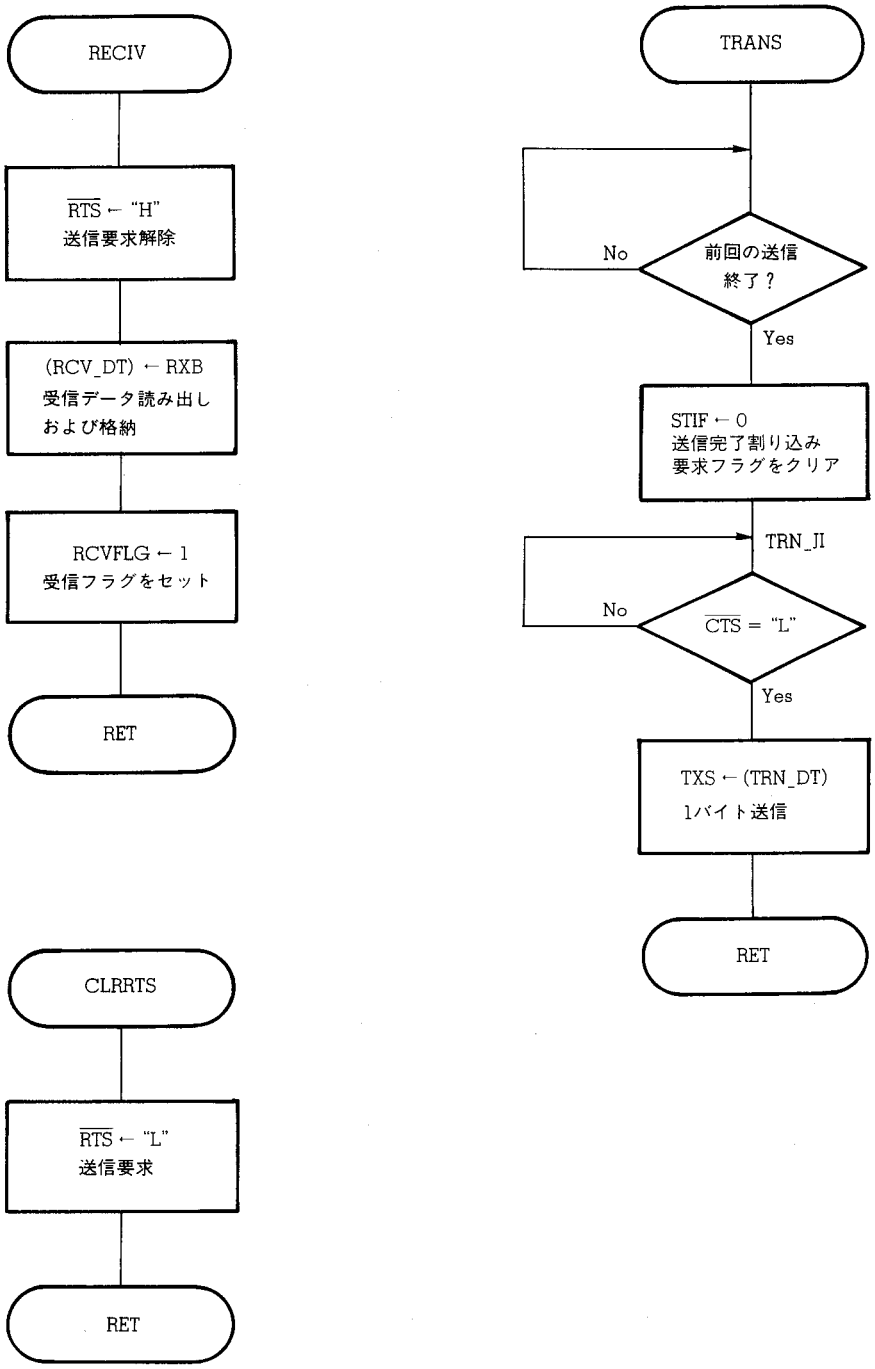
CR30D に 1 を設定した場合に作成されるボー・レートは，実際には次のようになりますので，4800 bps に対して，＋1.7％の誤差が生じることになります。

$$\text{ボー・レート} = \frac{5\times10^6/16}{1+1} \times \frac{1}{16} \times \frac{1}{2} \fallingdotseq 4883\ (\text{bps})$$

プログラム例として次のように用います。

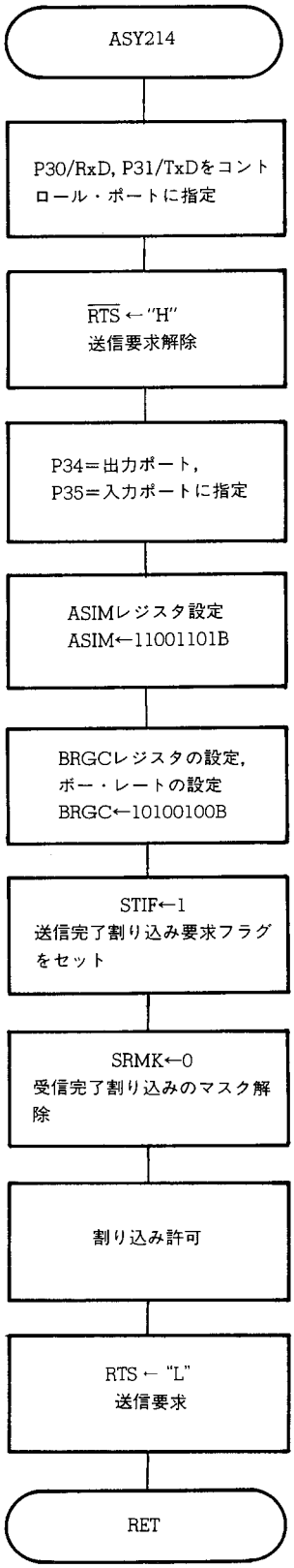
```
        PUBLIC  TRN_DT,RCV_DT                           ; DATA WORK
        PUBLIC  RCVFLG                                  ; FLAG
        PUBLIC  PRMOD,CR30D

        EXTRN   ASY220                                  ; PACKAGE
        EXTRN   TRANS,RECIV,CLRRTS                      ; PACKAGE
                .
                .
;       BAUD RATE :             4800bps
;       STOP BIT :              2bit
;       CHARACTER LENGTH :      8bit*
;       PARITY :                NO PARITY

PRMOD   EQU     00100000B       ; PRM0 DATA (fclk/16)
CR30D   EQU     1               ; CR30 DATA

        BSEG
RCVFLG  DBIT                    ; RECEIVE FLAG

ASY20_D DSEG    SADDR
TRN_DT: DS      1               ; TRANSMIT DATA
RCV_DT: DS      1               ; RECEIVE DATA
                .
                .
INTSRVT CSEG    AT 00022H
        DW      INTSR           ; INTSR

        CSEG
                .
                .
        CALL    !ASY220         ; <<< ASY220 >>>
SER_L1:
        BTCLR   RCVFLG,$SER_J1  ; CHECK RECEIVE FLAG
        BR      SER_L1
SER_J1:
        MOV     TRN_DT,RCV_DT   ; TRANS DATA
        CALL    !TRANS          ; <<< TRANS >>>
        CALL    !CLRRTS         ; <<< CLRRTS >>>
        BR      SER_L1

INTSR:
        CALL    !RECIV          ; <<< RECIV >>>
        RETI
```

(5) 両者共通フロー・チャート

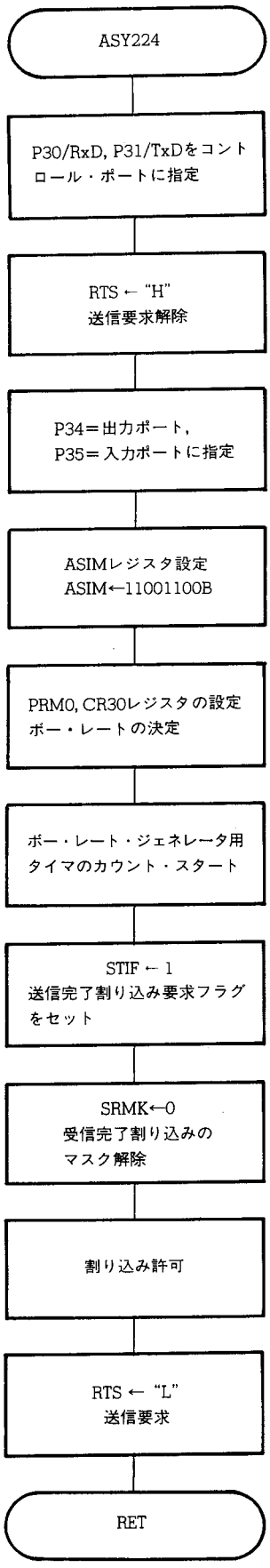


(6) デバイスごとに異なるフロー・チャート

(a) μPD78214のプログラム



(b)　μPD78224 のプログラム



(7) プログラム・リスト

(a) μPD78214のプログラム

```
        NAME    AS214

;*************************************************************
;       asynchronous serial interface for 78214
;                       at 12MHz
;*************************************************************

        PUBLIC  ASY214,RECIV,TRANS,CLRRTS
        EXTRN   TRN_DT,RCV_DT
        EXTRN   BRGCD
        EXTBIT  RCVFLG          ; receive flag
STIF    EQU     IFOH.6          ; INTST flag
SRMK    EQU     MKOH.5          ; mask of INTSR
CTS     EQU     P3.5            ; CTS for RS-232-C
RTS     EQU     P3.4            ; RTS for RS-232-C

        CSEG
ASY214:
        OR      PMC3,#00000011B ; P3.0,P3.1 = Control port
        SET1    RTS             ; RTS <- 1
        MOV     PM3,#00100000B  ; P3.5=IN , P3.4=OUT
        MOV     ASIM,#11001101B ; ASIM initialize
        MOV     BRGC,#LOW(BRGCD)  ; Baud rate = 9600bps
        SET1    STIF            ; dummy set INTST
        CLR1    SRMK            ; open mask of INTSR
        EI                      ; interrupt enable
        CLR1    RTS             ; receive enable
        RET

;*************************************************************
;       receive process
;*************************************************************

RECIV:
        SET1    RTS             ; RTS <- 1
        MOV     A,RXB           ; read receive data
        MOV     RCV_DT,A
        SET1    RCVFLG          ; set receive flag
        RET
CLRRTS:                         ; clear RTS
        CLR1    RTS
        RET

;*************************************************************
;       transmit process
;*************************************************************

TRANS:
        BTCLR   STIF,$TRN_J1    ; check complete of transmit
        BR      TRANS
TRN_J1:
        BT      CTS,$TRN_J1     ; check CTS
        MOV     A,TRN_DT
        MOV     TXS,A           ; transmit
        RET

        END
```

(b) μPD78224のプログラム

```
        NAME    AS220

;*************************************************************
;       asynchronous serial interface for 78220
;                       at 10MHz
;*************************************************************

        PUBLIC  ASY220,RECIV,TRANS,CLRRTS
        EXTRN   TRN_DT,RCV_DT
        EXTRN   PRMOD,CR30D
        EXTBIT  RCVFLG          ; receive flag
;
STIF    EQU     IF0H.6          ; INTST flag
SRMK    EQU     MK0H.5          ; mask of INTSR
CTS     EQU     P3.5            ; CTS for RS-232-C
RTS     EQU     P3.4            ; RTS for RS-232-C
;
        CSEG
ASY220:
        OR      PMC3,#00000011B ; P3.0,P3.1 = Control port
        SET1    RTS             ; RTS <- 1
        MOV     PM3,#00100000B  ; P3.5=IN , P3.4=OUT
        MOV     ASIM,#11001100B ; ASIM initialize
        MOV     PRM0,#LOW(PRMOD)  ; Baud rate = 4800bps
        MOV     CR30,#LOW(CR30D)
        MOV     TMC0,#10000000B
        SET1    STIF            ; dummy set INTST
        CLR1    SRMK            ; open mask of INTSR
        EI                      ; interrupt enable
        CLR1    RTS             ; receive enable
        RET

;*************************************************************
;       receive process
;*************************************************************

RECIV:
        SET1    RTS             ; RTS <- 1
        MOV     A,RXB           ; read receive data
        MOV     RCV_DT,A
        SET1    RCVFLG          ; set receive flag
        RET
CLRRTS:                         ; clear RTS
        CLR1    RTS
        RET

;*************************************************************
;       transmit process
;*************************************************************

TRANS:
        BTCLR   STIF,$TRN_J1    ; check complete of transmit
        BR      TRANS
TRN_J1:
        BT      CTS,$TRN_J1     ; check CTS
        MOV     A,TRN_DT
        MOV     TXS,A           ; transmit
        RET

        END
```

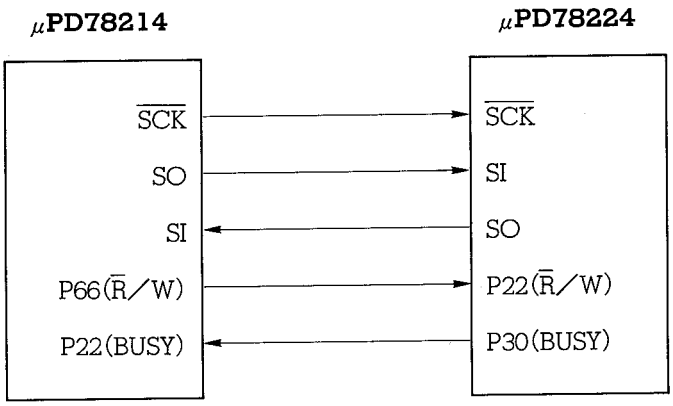
第6章　3線式シリアル・インタフェースのプログラム例

3線式シリアル・インタフェースのプログラム例として，78K/II シリーズのデバイス間での通信例を示します。

(1) 動作概要

デバイスとしてマスタ・デバイス，スレーブ・デバイスをそれぞれ μPD78214，μPD78224 とします。図6-1に接続図を示します。

図6-1　μPD78214 と μPD78224 との接続

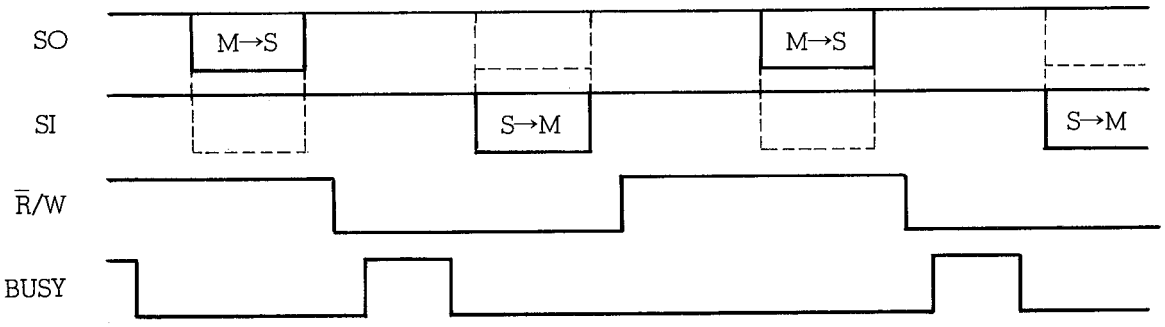


通信プロトコルの仕様を次に示します。

ボー・レートは 312.5 kbps とします。ただし，f_{CLK} = 5 MHz とします。ハンドシェーク・ラインとして，R/W と BUSY を設けます。マスタ（μPD78214）とスレーブ（μPD78224）で交互に送信を行います。

マスタ側からみたタイミング・チャートを図6-2に示します。

図6-2　マスタ側からみたタイミング・チャート



備考　M→S：マスタからスレーブへの送信
　　S→M：スレーブからマスタへの送信

このプログラムではマスタ，スレーブそれぞれ次の3つのワーク・エリアを使用します。ただし，ショート・ダイレクト・アドレシングの適用範囲（FE20H-FEF7H）に配置してください。

表6-1　3線式シリアル・インタフェースのプログラム例で用いるワーク・エリア（マスタ側）

ワーク・エリア名称	用　　途
RCV_DT	受信データ用ワーク・エリア
TRN_DT	送信データ用ワーク・エリア

表6-2　3線式シリアル・インタフェースのプログラム例で用いるフラグ（マスタ側）

フラグ名称	用　　途
TRNEND	送信完了フラグ
RCVEND	受信完了フラグ

表6-3　3線式シリアル・インタフェースのプログラム例で用いるワーク・エリア（スレーブ側）

ワーク・エリア名称	用　　途
RCV_DT	受信データ用ワーク・エリア
TRN_DT	送信データ用ワーク・エリア

表6-4　3線式シリアル・インタフェースのプログラム例で用いるフラグ（スレーブ側）

フラグ名称	用　　途
RCVEND	受信完了フラグ

(2) プログラム説明 …… (7) プログラム・リスト参照

(a) マスタ側イニシャライズ処理［レーベル名称：CLKMIN］

(i) P32，P33 コントロール・ポート（$\overline{SCK}$，SO）として使用します。

(ii) RWFLAG = 1 として，通信方向を“マスタ → スレーブ”とします。

(iii) P66 を出力ポートに設定します。

(iv) 3線式シリアル・インタフェースのイニシャライズを行います。送信許可。

(v) シリアル・クロック（ボー・レート）の設定を行います。

(vi) BUSY 端子（INTP1）の有効エッジを立ち下がりエッジに設定します。

(vii) INTP1 割り込み要求フラグをクリアします。

(viii) INTCSI（クロック同期式シリアル・インタフェース転送終了）割り込みのマスクを解除します。

(b) マスタ側送信処理［レーベル名称：TRANS］

(i) スレーブの BUSY が解除されるまでウエイトします。

(ii) 送信許可とします。

(iii) シフト・レジスタ（SIO）に送信データを書き込みます。

(c) マスタ側受信処理［レーベル名称：RECIVE］

受信許可とします。78K/II のクロック同期式シリアル・インタフェースでは，マスタ側（シリアル・クロックを供給する側）の場合，受信禁止の状態から受信許可とした場合，およびシフト・レジスタ（SIO）に対するリード動作によってシリアル・クロックが出力されます。

(d) マスタ側転送終了割り込み処理［レーベル名称：ENDTR］

(i) RWFLAG フラグを反転します。

(ii) RWFLAG = 1（前回の転送は“スレーブ → マスタ”）の場合は（v）へ。

(iii) スレーブが受信したことは PIF1 フラグをポーリングして調べます。

(iv) 送信禁止とし，送信完了フラグ TRNEND をセット（1）して戻ります。

(v) 受信禁止として，受信データをメモリに書き込みます。

(vi) 受信完了フラグ RCVEND をセット（1）して戻ります。

6
214
218A
224
234
244

(e)　**スレーブ側イニシャライズ処理［レーベル名称：CLKSIN］**

(i) P32，P33 コントロール・ポート（$\overline{SCK}$，SO）として使用します。

(ii) P30 を BUSY 端子として用いるために，BUSY 状態，出力ポートとして設定します。BUSY の解除は，他のイニシャライズが終了してから行ってください。

(iii) 3 線式シリアル・インタフェースのイニシャライズを行います。送受信許可。

(iv) $\overline{R}$/W 端子（INTP1）の有効エッジを両エッジに設定します。

(v) INTP1 割り込み要求フラグをクリアします。

(vi) INTCSI（クロック同期式シリアル・インタフェース転送終了）割り込みのマスクを解除します。

(f)　**スレーブ側送信処理［レーベル名称：TRANS］**

(i) 送信データをシフト・レジスタ（SIO）に書き込みます。

(ii) BUSY を解除します。

(g)　**スレーブ側転送終了割り込み処理［レーベル名称：ENDTR］**

(i) $\overline{R}$/W が変化するまでウエイトします。

(ii) $\overline{R}$/W が“0”ならば（iv）へ。

(iii) BUSY を解除して戻ります。

(iv) BUSY 状態に設定します。

(v) 受信データをメモリに書き込みます。

(vi) 受信完了フラグ RCVEND をセット（1）して戻ります。

(3) モード・レジスタ設定例

(a) マスタ側のモード・レジスタ設定例

ポート3モード・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	(リセットで00H)
設定例	0	0	0	0	1	1	0	0	

PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

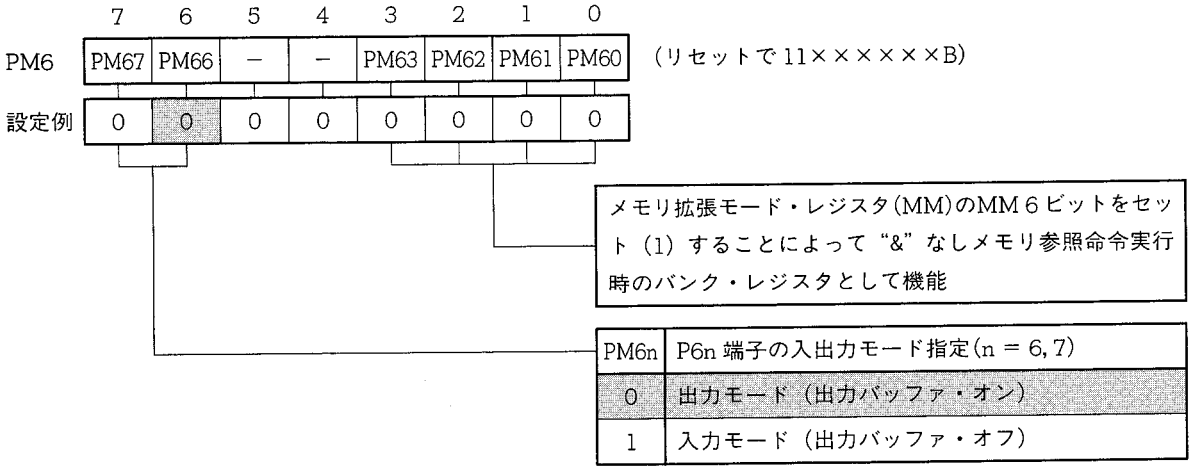
PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SB0入出力モード/SO出力モード

PMC3n	P3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

6
214
218A
224
234
244

ポート６モード・レジスタ



注意　“－”は，１でも０でもかまいません。

クロック同期式シリアル・インタフェース・モード・レジスタ

	7	6	5	4	3	2	1	0	
CSIM	CTXE	CRXE	WUP	0	M0D1	0	CLS1	CLS0	（リセットで00H）
設定例	1	0	0	0	0	0	0	1	

CLS1	CLS0	シリアル・クロックの選択		$\overline{SCK}$ 端子
0	0	外部クロック		入力
0	1	内部クロック	8ビット・タイマ/カウンタ3	出力
1	0		$f_{CLK}/32$[注]	
1	1		$f_{CLK}/8$[注]	

注　f_{CLK}：内部システム・クロック

M0D1	クロック同期式シリアル・インタフェース動作モードの選択
0	3線式シリアルI/Oモード
0	SBIモード

WUP	ウエイク・アップ機能の指定
0	すべてのモードでシリアルを転送後ごとに，割り込み要求信号を発生
1	SBIモードのみアドレスを受信した場合のみ割り込み要求信号を発生

CRXE	受信動作
0	禁止
1	許可

CTXE	送信動作
0	禁止
1	許可

備考　f_{CLK}：内部システム・クロック

プリスケーラ・モード・レジスタ0

	7	6	5	4	3	2	1	0
PRM0	PRS3	PRS2	PRS1	PRS0	0	0	0	0
設定例	0	0	0	0	0	0	0	0

8ビット・タイマ/カウンタ3[注]

PRS3	−	PRS1	PRS0	−	μPD78224
PRS33	PRS32	PRS31	PRS30	その他	−
0	0	0	0	$f_{CLK}/8$	$f_{CLK}/8$
0	0	0	1		
0	0	1	0	$f_{CLK}/16$	$f_{CLK}/16$
0	0	1	1	$f_{CLK}/32$	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$	設定禁止
0	1	0	1	$f_{CLK}/128$	
0	1	1	0	$f_{CLK}/256$	
0	1	1	1	$f_{CLK}/512$	
1	0	0	0	設定禁止	高速転送モード
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	×	×	設定禁止	

注　μPD78224では内部ボー・レート・ジェネレータ

備考　f_{CLK}：内部システム・クロック

タイマ・コントロール・レジスタ0

	7	6	5	4	3	2	1	0	
TMC0	CE	0	0	0	CE0	CVF0	0	0	（リセットで00H）
設定例	1	0	0	0	0	0	0	0	

16ビット・タイマ/カウンタ

OVF0	TM0のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFFFHから0000Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE0	TM0のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ・ユニット3

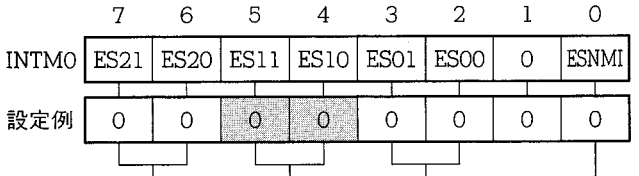
CE	TM3のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

割り込みマスク・レジスタH

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	（リセットでFFH）
設定例	0	×	×	×	×	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

外部割り込みモード・レジスタ0



ESNMI	NMI端子入力，検出エッジ指定
0	立ち下がりエッジ
1	立ち上がりエッジ

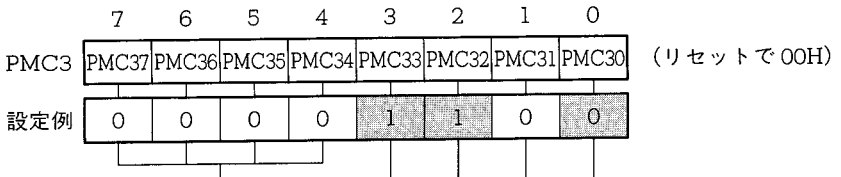
ES01	ES00	INTP0端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES11	ES10	INTP1端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES21	ES20	INTP2端子入力，検出エッジ
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

(b)　スレーブ側のモード・レジスタ設定例

ポート3モード・コントロール・レジスタ



PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

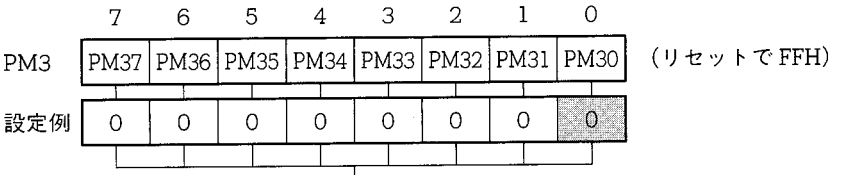
PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	SCK入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SB0入出力モード/SO出力モード

PMC3n	P3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

ポート3モード・レジスタ



PM3n	P3n端子入出力モード指定(n＝0−7)
0	出力モード（出力バッファ・オン）
1	入力モード（出力バッファ・オフ）

クロック同期式シリアル・インタフェース・モード・レジスタ

	7	6	5	4	3	2	1	0	
CSIM	CTXE	CRXE	WUP	0	MOD1	0	CLS1	CLS0	（リセットで00H）
設定例	1	1	0	0	0	0	0	0	

CLS1	CLS0	シリアル・クロックの選択		SCK 端子
0	0	外部クロック		入力
0	1	内部クロック	8ビット・タイマ/カウンタ3	出力
1	0		$f_{CLK}/32$注	
1	1		$f_{CLK}/8$注	

注　f_{CLK}：内部システム・クロック

MOD1	クロック同期式シリアル・インタフェース動作モードの選択
0	3線式シリアルI/Oモード
0	SBIモード

WUP	ウエイク・アップ機能の指定
0	すべてのモードでシリアルを転送後ごとに，割り込み要求信号を発生
1	SBIモードのみアドレスを受信した場合のみ割り込み要求信号を発生

CRXE	受信動作
0	禁止
1	許可

CTXE	送信動作
0	禁止
1	許可

備考　f_{CLK}：内部システム・クロック

6
214
218A
224
234
244

割り込みマスク・レジスタH

	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	PMK21	（リセットでFFH）
設定例	0	×	×	×	×	×	×	×	×：操作しません

MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

外部割り込みモード・レジスタ0

	7	6	5	4	3	2	1	0	
INTM0	ES21	ES20	ES11	ES10	ES01	ES00	0	ESNMI	（リセットで00H）
設定例	0	0	1	1	0	0	0	0	

ESNMI	NMI端子入力，検出エッジ指定
0	立ち下がりエッジ
1	立ち上がりエッジ

ES01	ES00	INTP0端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES11	ES10	INTP1端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES21	ES20	INTP2端子入力，検出エッジ
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

(4) 入出力パラメータ

(a) PRM0D：プリスケーラ・モード・レジスタ（PRM0）に設定する値です。

CR30D：8ビット・コンペア・レジスタ（CR30）に設定する値です。

PRM0D, CR30Dはシリアル・クロック（ボー・レート）の設定パラメータです。ボー・レートの計算方法は，次のとおりです。

$$\text{ボー・レート} = \frac{\text{TM3のカウント・クロック}}{\text{CR30} + 1} \times \frac{1}{2}$$

$$= \frac{5 \times 10^6/8}{\text{CR30} + 1} \times \frac{1}{2}$$

したがって，312.5 kbpsの場合のPRM0 = 00000000B，CR30 = 0です。

(b) TRN_DT　：送信データです。送信データをこのメモリに設定して［TRANS］を呼び出してください。

(c) RCV_DT　：受信データはこのメモリに格納されます。

(d) TRNEND　：送信完了フラグです。1バイト送信するごとにセット（1）されるフラグです。

(e) RCVEND　：受信完了フラグです。INTCSI割り込み処理により1バイト受信するごとにセット（1）されるフラグです。

(5) 使用レジスタ

マスタ側

(a) CLKMIN　：なし

(b) TRANS　：Aレジスタ

(c) RECIVE　：なし

(d) ENDTR　：Aレジスタ

スレーブ側

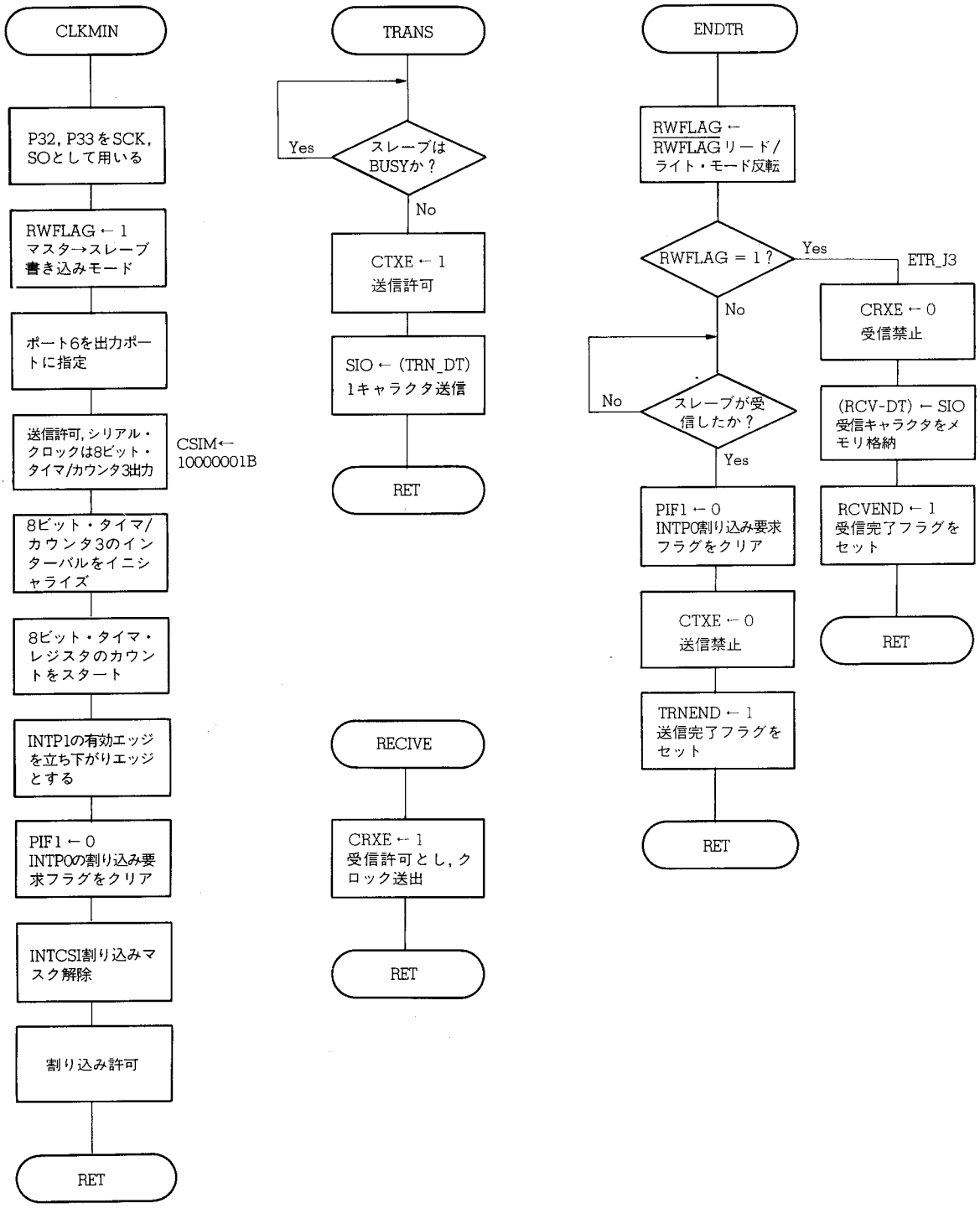
(a) CLKSIN　：Aレジスタ

(b) TRANS　：Aレジスタ

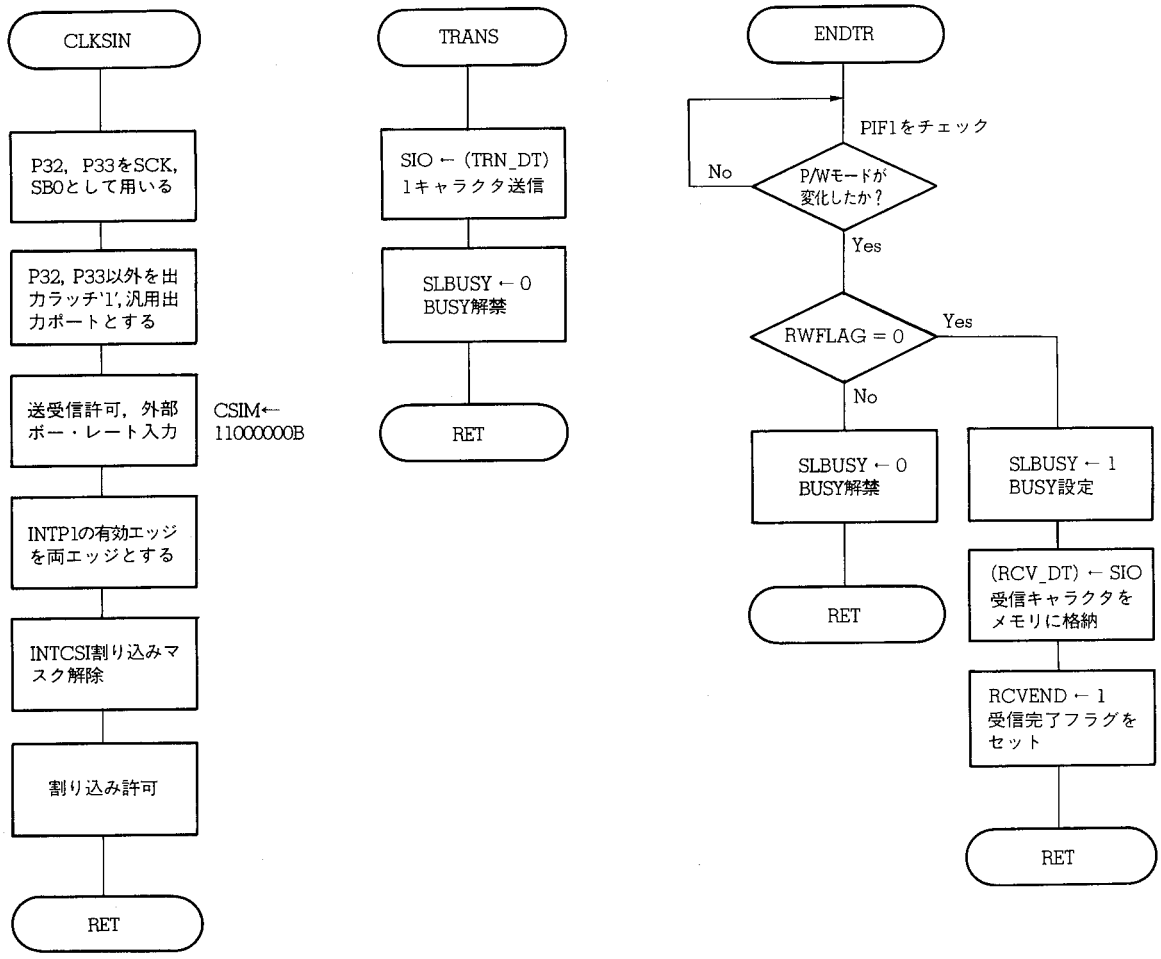
(c) ENDTR　：Aレジスタ

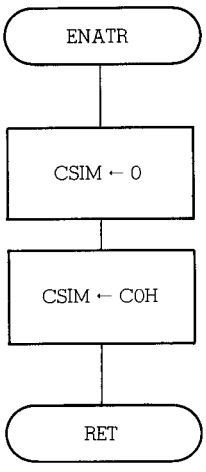
(6) フロー・チャート

(a) マスタ側



(b)　スレーブ側





6
214
218A
224
234
244

(7) プログラム・リスト

(a) マスタ側

```
        NAME    CLKDMR
;
;*************************************************************
;       clocked serial I/O interface
;*************************************************************
;
        PUBLIC  CLKMIN,TRANS,RECIVE,ENDTR
        EXTRN   TRN_DT,RCV_DT
        EXTRN   PRMOD,CR30D
        EXTBIT  TRNEND,RCVEND

CTXE    EQU     CSIM.7          ; transmit enable
CRXE    EQU     CSIM.6          ; receive enable
CSIMK   EQU     MK0H.7          ; mask of INTCSI
PIF1    EQU     IF0L.1          ; slave busy flag
SLBUSY  EQU     P2.2            ; SLAVE BUSY FLAG
RWFLAG  EQU     P6.6            ; R/W FLAG 0:READ 1:WRITE
;
;       *** initialize of clocked serial I/O ***
;
        CSEG
CLKMIN:
        MOV     PMC3,#00001100B ; P32 -> SCK  ,  P33 -> SO
        SET1    RWFLAG          ; default=write mode
        MOV     PM6,#00H        ; P66 -> R/W
        MOV     CSIM,#10000001B ; CSIM initialize ( RCV:bad , TRN:ok )
        MOV     PRM0,#LOW(PRMOD)  ; initialize PRM0
        MOV     CR30,#LOW(CR30D)  ; initialize CR30
        MOV     TMC0,#10000000B ; internal serial clock start
        MOV     INTM0,#00000000B
                                ; INTP0 rise edge enable
        CLR1    PIF1            ; clear INTP1 flag
        CLR1    CSIMK           ; INTCSI enable
        EI                      ; interrupt enable

        RET
;
;       *** trans process ***
;
TRANS:
        BT      SLBUSY,$TRANS   ; check busy
        SET1    CTXE            ; transmit enable
        MOV     A,TRN_DT
        MOV     SIO,A           ; transmit data
        RET
;
;       *** receive data ***
;
RECIVE:
        SET1    CRXE            ; receive enable
        RET
```

```
;
;       *** transmit/receive complete ***
;
ENDTR:
        NOT1    RWFLAG          ; change read/write mode
        BT      RWFLAG,$ETR_J3  ; check R/W mode
ETR_J1:
        BTCLR   PIF1,$ETR_J2    ; slave receive check
        BR      ETR_J1
ETR_J2:
        CLR1    CTXE            ; transmit disable
        SET1    TRNEND          ; set transmit complete flag
        RET
ETR_J3:
        CLR1    CRXE            ; receive disable
        MOV     A,SIO           ; read data
        MOV     RCV_DT,A
        SET1    RCVEND          ; set receive end flag
        RET
;
        END
```

(b)　スレーブ側

```
        NAME    CLKDSR
;
;************************************************************
;       clocked serial I/O interface
;************************************************************
;
        PUBLIC  CLKSIN,TRANS,ENDTR
        EXTRN   TRN_DT,RCV_DT
        EXTBIT  RCVEND

PIF1    EQU     IFOL.1          ; INTP0 flag
CSIMK   EQU     MKOH.7          ; mask of INTCSI
SLBUSY  EQU     P3.0            ; SLAVE BUSY FLAG
RWFLAG  EQU     P2.2            ; R/W FLAG 0:READ 1:WRITE
;
;       *** initialize of clocked serial I/O ***
;
        CSEG
CLKSIN:
        MOV     PMC3,#00001100B ; P32 -> SCK  ,  P33 -> SO
        MOV     P3,#0FFH        ; initialize P3
        MOV     PM3,#0
        MOV     CSIM,#11000000B ; CSIM initialize ( RCV:ok , TRN:ok )
        MOV     INTM0,#00110000B
                                ; INTP0 all edge enable
        CLR1    PIF1            ; clear INTP1 flag
        CLR1    CSIMK           ; INTCSI enable
        EI                      ; interrupt enable

        RET
;
;       *** transmit ***
;
TRANS:
        MOV     A,TRN_DT
        MOV     SIO,A           ; transmit data
        CLR1    SLBUSY          ; busy clear
        RET
;
;       *** transmit/receive complete ***
;
ENDTR:
        BTCLR   PIF1,$EDT_J1    ; check R/W edge
        BR      ENDTR
EDT_J1:
        BF      RWFLAG,$EDT_J2  ; check R/W mode
        CLR1    SLBUSY          ; clear busy
        RET
EDT_J2:
        SET1    SLBUSY          ; SET BUSY
        MOV     A,SIO           ; receive data
        MOV     RCV_DT,A
        SET1    RCVEND
        RET
;
        END
```

第7章　割り込み処理のプログラム例

78K/IIシリーズは割り込み要求に対する処理として，表7-1のような2つの処理をプログラムで選択できます。

表7-1　78K/IIシリーズの割り込み要求の処理

処理モード	処理の主体	処　　理	PC，PSWの内容
ベクタ割り込み	ソフトウエア	処理ルーチンへ分岐して実行	退避，復帰を行う
マクロ・サービス	ファームウエア	メモリーI/O間のデータ転送処理などを実行	保持

両者の比較として，ここでは次の項目についてプログラム例を示します。

(i) アシンクロナス・シリアル・インタフェース（UART）受信処理

(ii) 外部割り込み要求に同期したパラレル・データ入力

(iii) ステッピング・モータの開ループ制御（マクロ・サービスのみ）

マクロ・サービスの場合(i)はタイプA，(ii)はタイプB，(iii)はタイプCでそれぞれ処理を行います。

7.1　UART受信処理

内蔵のUARTを用いて，データを連続入力する例について示します。ただし，f_{CLK}＝6MHzとします。

(1) 動作概要

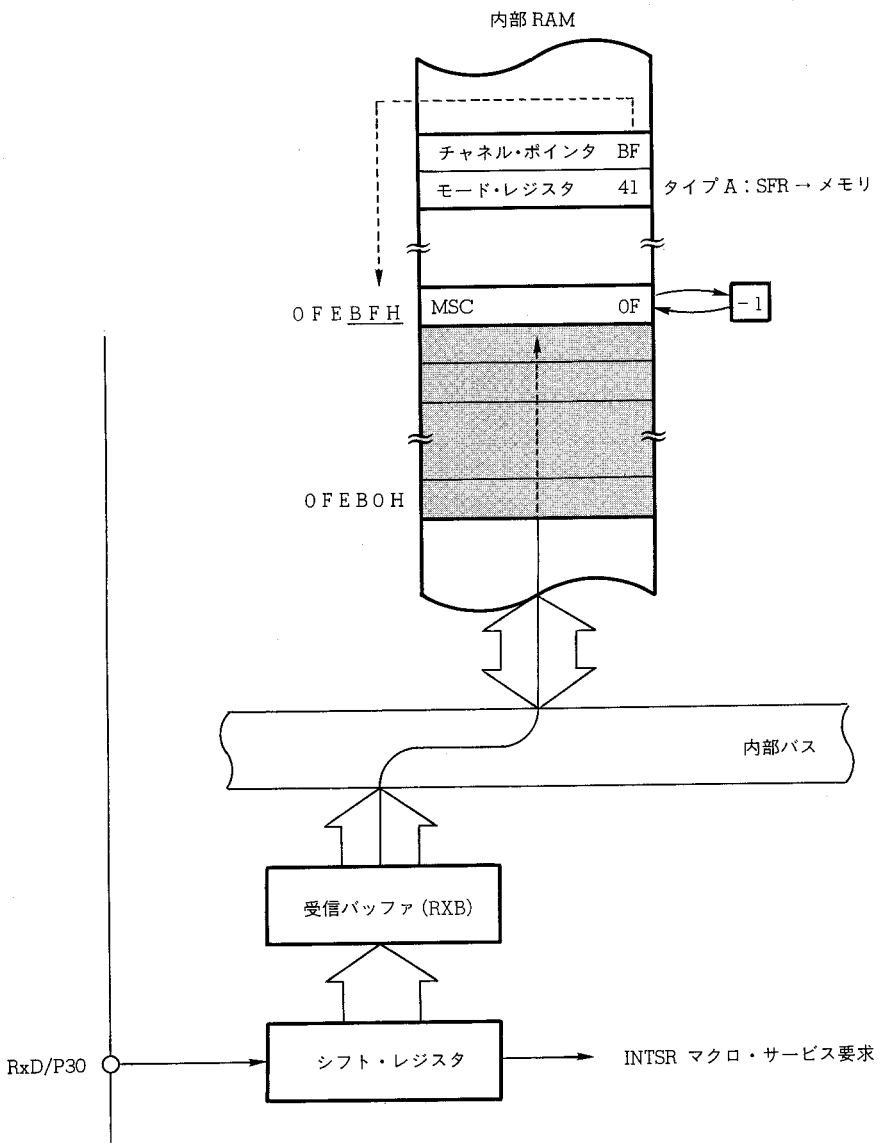
(a) マクロ・サービス

UARTにより，受信した15バイトのデータをマクロ・サービス・タイプAを用いて，内部RAMのバッファ領域内に転送します。マクロ・サービス・カウンタ(MSC)の格納アドレスは，FEBFHとし，初期値を15(FH)とします。マクロ・サービス完了割り込み処理は，マクロ・サービス・カウンタ(MSC)および，チャネル・ポインタ(CHP)に初期設定値を再設定します。

(b) ベクタ割り込み

(a) マクロ・サービス処理と同じワーク・エリアを用いてベクタ割り込み処理によって同一の動作をシミュレートします。

図7-1　マクロ・サービスを用いた場合のメモリ・マップ（UART受信処理）



(2) プログラム説明 … (8) プログラム・リスト参照

マクロ・サービス

(a) マクロ・サービス・イニシャライズ処理〔レーベル名称：TY_AMI〕

(i) マクロ・サービス・モード・レジスタの設定を行います。
　タイプA，SFR→メモリに設定します。

(ii) チャネル・ポインタの設定を行います。

(iii) マクロ・サービス・カウンタに転送回数を設定します。

(iv) INTSR割り込み要求に対する処理をマクロ・サービスとします。

(v) 割り込みを許可します。

(b) マクロ・サービス完了割り込み処理〔レーベル名称：TY_AMR〕

再起動する処理を行っています。

(i) マクロ・サービス・カウンタの値を再設定します。

(ii) マクロ・サービス完了割り込み時はINTSRはベクタ割り込みモードになっていますので，再び処理モードをマクロ・サービスとします。

(c) ベクタ割り込み：UART受信処理〔レーベル名称：TY_AIR〕

この処理がマクロ・サービスの転送処理に相当します。

(i) 受信データを（STRDTA＋(SPINTA)）に格納します。

(ii) (SPINTA) をインクリメントします。

(iii) (SPINTA) ＝ CNTNMAならば（SPINTA）←0とします。

(iv) 受信完了フラグENDRCVをセット(1) します。

(3) モード・レジスタ設定例

マクロ・サービス・モード・レジスタ

7	6	5	4	3	2	1	0	
CHT2	CHT1	CHT0	0	MOD3	MOD2	MOD1	MOD0	(リセットで00H)
設定例 0	1	0	0	0	0	0	1	

CHT0	0	1	0	1
CHT1	1	0	1	
CHT2	0	1	1	

MOD3	MOD2	MOD1	MOD0	タイプA	タイプB	タイプC	
						MPTL保持	MPTLインクリメント
0	0	0	0	メモリからSFRへデータ転送			
0	0	0	1	SFRからメモリへデータ転送			
1	0	0	0			リング制御なし / データ転送のみ	POL
1	0	0	1				POH
1	0	1	0			リング制御なし / 自動加算つき	POL
1	0	1	1				POH
1	1	0	0			リング制御つき / データ転送のみ	POL
1	1	0	1				POH
1	1	1	0			リング制御つき / 自動加算つき	POL
1	1	1	1				POH

割り込みサービス・モード・レジスタH

7	6	5	4	3	2	1	0	
ISM0H CSIISM	STISM	SRISM	0	0	PISM5	PISM4	0	(リセットで00H)
設定例 ×	×	1	×	×	×	×	×	×：操作しません

ISM	割り込みサービス・モード・フラグ
0	ベクタ割り込みで処理
1	マクロ・サービスで処理

(4) 入出力パラメータ

●マクロ・サービス

(a) CNTNMA：マクロ・サービス・カウンタへの設定値です。この設定回数分の転送が行われたのちにマクロ・サービス完了割り込みが発生します。この割り込みはINTSRによって起動されますので，ベクタ・テーブルの参照はINTSRのベクタ・アドレス(0022H)となります。

(b) CHASR　：チャネル・ポインタへ設定するアドレスです。タイプAのマクロ・サービスですから，後述のマクロ・サービス・カウンタの配置アドレスはCHASRとなります。

(c) MSCSR　：マクロ・サービス・カウンタの配置アドレスです。

●ベクタ割り込み

(a) SPINTA　：受信データの書き込み先のポインタの格納されているアドレスです。書き込み先のアドレスは，STRDTA+（SPINTA）で表されます。

(b) STRDTA　：受信データの転送先のベース・アドレスです。

(c) CNTNMA：ひとまとまりの受信回数です。この設定値の回数の受信が完了すると，受信完了フラグENDRCVがセット(1)されます。

(d) ENDRCV　：ひとまとまりの受信が完了するとセット(1)されるフラグです。

●UARTイニシャライズ

(a) BRGC　：ボー・レート・ジェネレータ・コントロール・レジスタ（BRGC）に設定する値です。

ボー・レートの計算方法は次のとおりです。

$$\text{ボー・レート} = \frac{\text{ボー・レート・ジェネレータのカウント・クロック}}{k+1} \times \frac{1}{n} \times \frac{1}{16}$$

k：BRGCレジスタのMDL3-MDL0ビットの設定値

1/n：分周回路のタップ

(5) 使用レジスタ

●マクロ・サービス

なし

●ベクタ割り込み

A，B

●UARTイニシャライズ

なし

(6) プログラム使用例

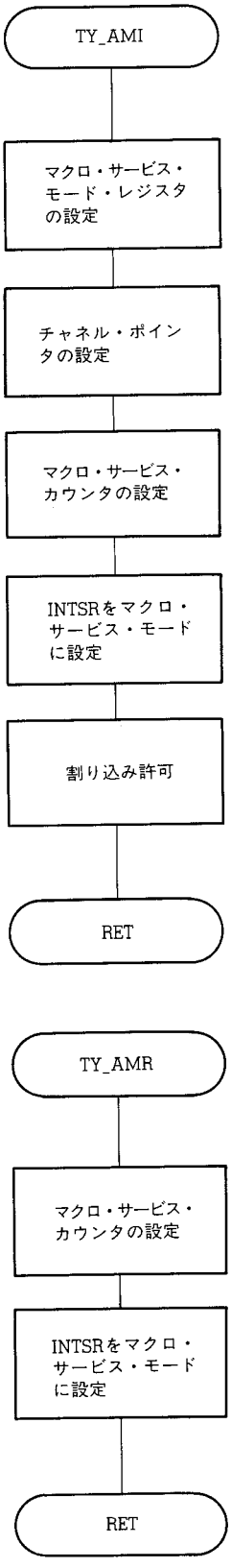
ベクタ割り込みを用いた場合，マクロ・サービスを用いた場合のどちらも，UARTのイニシャライズ処理「ASY214」を呼び出していますので，モジュール名「AS214」をリンクしてください。

マクロ・サービスの場合のプログラム例を示します。

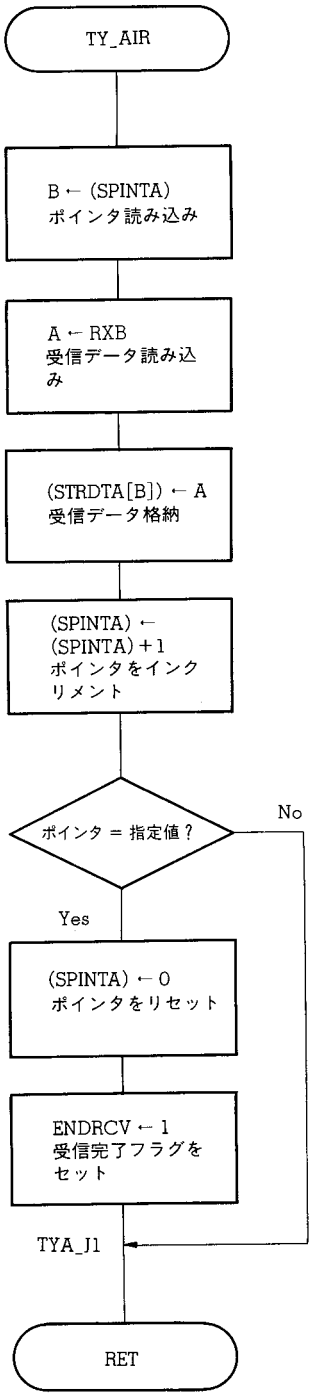
```
;       --- FOR TYPE A ---
;
        PUBLIC  BRGCD                           ; PARAMETER
        PUBLIC  CNTNMA,MSCSR,CHASR              ; PARAMETER
        PUBLIC  TRN_DT,RCV_DT,RCVFLG            ; DUMMY PUBLIC FOR /ASY214/

        EXTRN   ASY214,TY_AMI,TY_AMR            ; PACKAGE
                .
                .
;       BAUD RATE :             9600bps
;       STOP BIT :              2bit
;       CHARACTER LENGTH :      8bit*
;       PARITY :                NO PARITY
;
BRGCD   EQU     10100100B       ; BRGC DATA (9600bps,fclk/5)

SRMK    EQU     MKOH.5          ; MASK OF INTSR
CTS     EQU     P3.5            ; CTS FOR RS-232-C
RTS     EQU     P3.4            ; RTS FOR RS-232-C

CNTNMA  EQU     OFH             ; STORE POINTER DATA
MCHSR   DSEG    AT OFEBOH
BASR:   DS      CNTNMA
MSCSR:  DS      1               ; MACRO SERVICE COUNTER
CHASR   EQU     $-1

TRN_DT  EQU     OFEFFH          ; DUMMY FOR ASYNCHRONOUS MODULE EXTRN
RCV_DT  EQU     TRN_DT          ; DUMMY FOR ASYNCHRONOUS MODULE EXTRN
RCVFLG  EQU     TRN_DT.0        ; DUMMY FOR ASYNCHRONOUS MODULE EXTRN

INTSRVT CSEG    AT 00022H
        DW      INTSR           ; INTSR  vector
                .
                .
TY_A_M:
                .
                .
        CALL    !TY_AMI         ; <<< TY_AMI >>>
        CALL    !ASY214         ; SERIAL INTERFACE INITIALIZE
TYA_L1:                         ; DUMMY LOOP
        BR      TYA_L1
;
INTSR:                          ; FOR END OF RECEIVE
        SEL     RB1             ; CHANGE REG BANK
        SET1    RTS             ; RTS -> 1
                .
                .
        CALL    !TY_AMR         ; <<< TY_AMR >>>
        CLR1    RTS             ; RTS <- 0
        RETI
```

(7) フロー・チャート

(a) マクロ・サービス



(b) ベクタ割り込み



(8) プログラム・リスト

(a) マクロ・サービス

```
        NAME    TYA_MR

;*************************************************************
;* macro service type-A application at macro service      *
;*      asynchronous serial receive                       *
;*************************************************************

        PUBLIC  TY_AMI,TY_AMR

        EXTRN   CNTNMA,MSCSR,CHASR

MCWSR   DSEG    AT OFEDEH
MSMSR:  DS      1               ; INTSR macro service mode register
CHPSR:  DS      1               ; INTSR macro service channel pointer

MKSR    EQU     MKOH.5          ; INTSR mask
SRISM   EQU     ISMOH.5         ; INTSR macro service mode
;
;       *** initialize ***
;
        CSEG
TY_AMI:
        MOV     MSMSR,#01000001B
                                ; set macro service mode register
        MOV     CHPSR,#LOW(CHASR)
                                ; set macro service channel pointer
        MOV     MSCSR,#LOW(CNTNMA)
                                ; set macro service counter
        SET1    SRISM           ; initialize interrupt
        EI
        RET
;
;       *** interrupt of complete macro-service ***
;
TY_AMR:
        MOV     MSCSR,#LOW(CNTNMA)
                                ; restore macro service counter
        SET1    SRISM           ; set interrupt service mode
        RET
;
        END
```

(b)　ベクタ割り込み

```
        NAME    TYA_IR
;
;*************************************************************
;* macro service type-A application at interrupt             *
;*      asynchronous serial receive                          *
;*************************************************************
;
        PUBLIC  TY_AIR
        EXTRN   STRDTA,SPINTA,CNTNMA
        EXTBIT  ENDRCV
;
;       *** receive data ***
;
        CSEG
TY_AIR:
        MOV     A,SPINTA        ; read store pointer
        MOV     B,A             ; save A-register
        MOV     A,RXB           ; read receive data
        MOV     STRDTA[B],A     ; store receive data
        INC     SPINTA          ; increment store pointer
        CMP     SPINTA,#LOW(CNTNMA)
        BNZ     $TYA_J1
        MOV     SPINTA,#0
        SET1    ENDRCV          ; set end flag
TYA_J1:
        RET
;
        END
```

7
214
218A
224
234
244

7.2　外部割り込み要求に同期したパラレル・データ入力

外部割り込み要求に同期して，ポートに入力されたデータを連続的にメモリに転送します。

(1) 動作概要

(a) マクロ・サービス

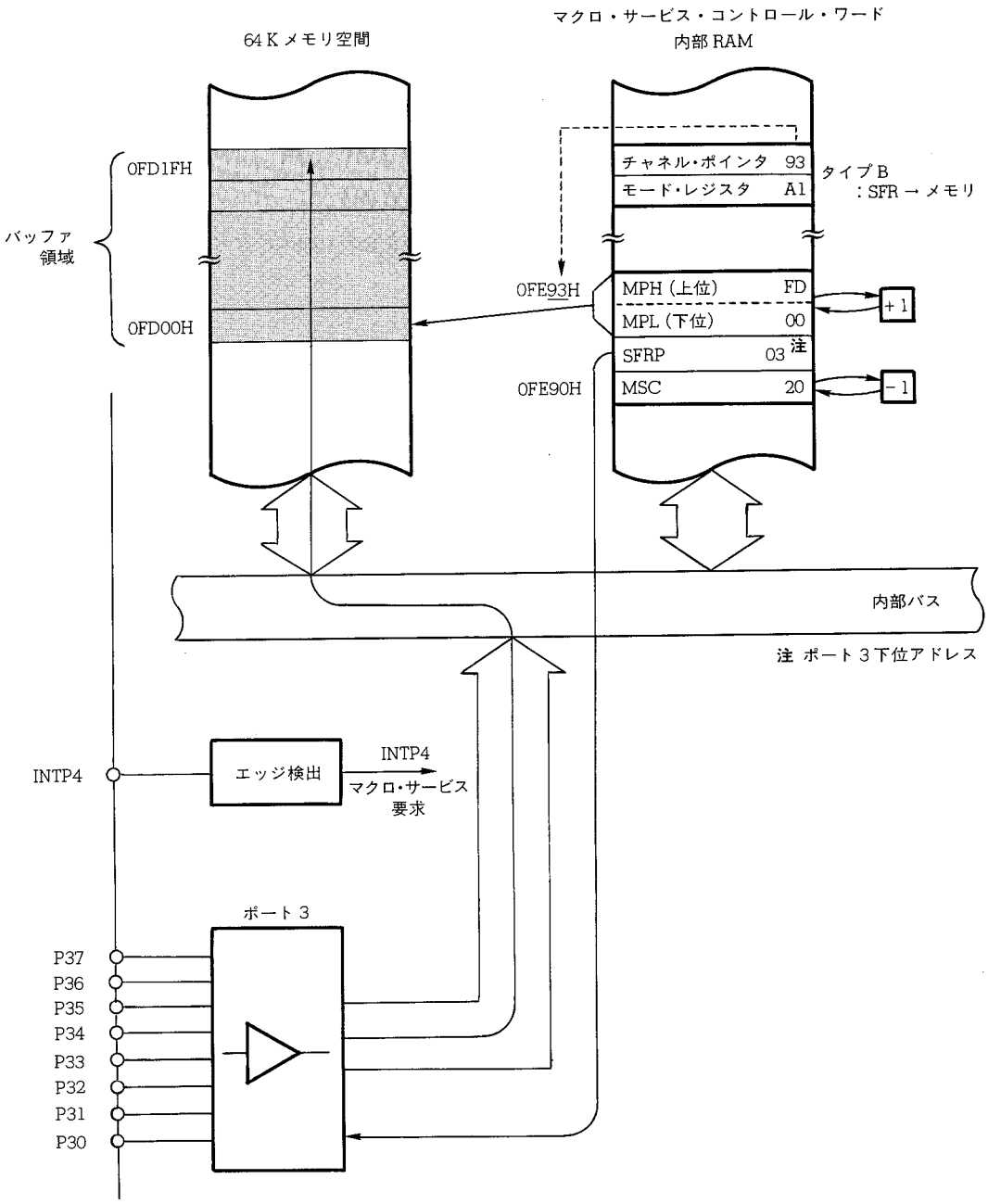
マクロ・サービス・タイプBを用いて外部割り込み要求INTP4に同期して，ポート3からパラレル・データを入力します。チャネル・ポインタ（CHP）の値は93Hとし，マクロ・サービス・カウンタ（MSC）の値は32（20H）とします。また，バッファ領域の先頭アドレスをFD00Hとします。

マクロ・サービス完了割り込み処理は，マクロ・サービス・カウンタ（MSC），チャネル・ポインタ（CHP）およびマクロ・サービス・ポインタ（MPH，MPL）に初期設定値を再設定します。

(b) ベクタ割り込み

(a) **マクロ・サービス処理**と同じワーク・エリアを用いてベクタ割り込み処理によって同一の動作をシミュレートします。

図7-2　マクロ・サービスを用いた場合のメモリ・マップ（パラレル・データ入力）



(2) プログラム説明…(8) プログラム・リスト参照

●マクロ・サービス

(a) マクロ・サービス・イニシャライズ処理〔レーベル名称：TY_BMI〕

(i) ポート3をポート・モード，入力ポートに設定します。

(ii) INTP4の有効エッジを立ち上がりエッジとします。

(iii) マクロ・サービス・モード・レジスタの設定を行います。タイプB, SFR → メモリに設定します。

(iv) チャネル・ポインタの設定を行います。

(v) マクロ・サービス・カウンタに転送回数を設定します。

(vi) SFRポインタにポート3の配置アドレスの下位8ビットを設定します。

(vii) マクロ・サービス・ポインタに転送先の先頭アドレスを設定します。

(viii) INTP4割り込み要求に対する処理をマクロ・サービスとします。

(ix) INTP4割り込みのマスクを解除します。

(x) 割り込みを許可します。

(b) マクロ・サービス完了割り込み処理〔レーベル名称：TY_BMR〕

再起動する処理を行っています。

(i) マクロ・サービス・カウンタの値を再設定します。

(ii) マクロ・サービス・ポインタの値を元に戻します。

(iii) マクロ・サービス完了割り込み時はINTP4はベクタ割り込みモードになっていますので，再び処理モードをマクロ・サービスとします。

●ベクタ割り込み

(a)　割り込み，ワーク・エリアのイニシャライズ処理〔レーベル名称：TY_BII〕

(i) ポート３をポート・モード，入力ポートに設定します。

(ii) INTP4 の有効エッジを立ち上がりエッジとします。

(iii) INTP4 割り込みのマスクを解除します。

(iv) INTP4 割り込み要求に対する処理をベクタ割り込みとします。

(v) (SPINTAB) に転送先の先頭アドレスを設定します。

(vi) (SCUNTB) に転送回数を設定します。

(vii) 割り込みを許可します。

(b)　パラレル・データ入力処理〔レーベル名称：TY_BIR〕

この処理がマクロ・サービスの転送処理に相当します。

(i) ポート３の入力データを (SPINTB) に格納します。

(ii) (SPINTB) をインクリメントします。

(iii) (SCUNTB) をデクリメントします。SCUNTB＝０ならば処理を終了します。

(iv) サンプリング完了フラグ ENDSMP をセット(1) します。

(3) モード・レジスタ設定例

ポート3モード・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	(リセットで00H)
設定例	0	0	0	0	0	0	0	0	

PMC30	P30端子のコントロール・モード指定
0	入出力ポート・モード
1	RxD入力モード

PMC31	P31端子のコントロール・モード指定
0	入出力ポート・モード
1	TxD出力モード

PMC32	P32端子のコントロール・モード指定
0	入出力ポート・モード
1	$\overline{SCK}$入出力モード

PMC33	P33端子のコントロール・モード指定
0	入出力ポート・モード
1	SB0入出力モード/SO出力モード

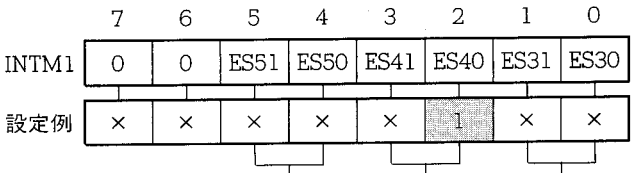
PMC3n	P3n端子のコントロール・モード指定(n = 4−7)
0	入出力ポート・モード
1	TOm出力モード(m = 0−3)

ポート3モード・レジスタ

	7	6	5	4	3	2	1	0	
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	(リセットでFFH)
設定例	1	1	1	1	1	1	1	1	

PM3n	P3n端子入出力モード指定(n = 0−7)
0	出力モード（出力バッファ・オン）
1	入力モード（出力バッファ・オフ）

外部割り込みモード・レジスタ 1



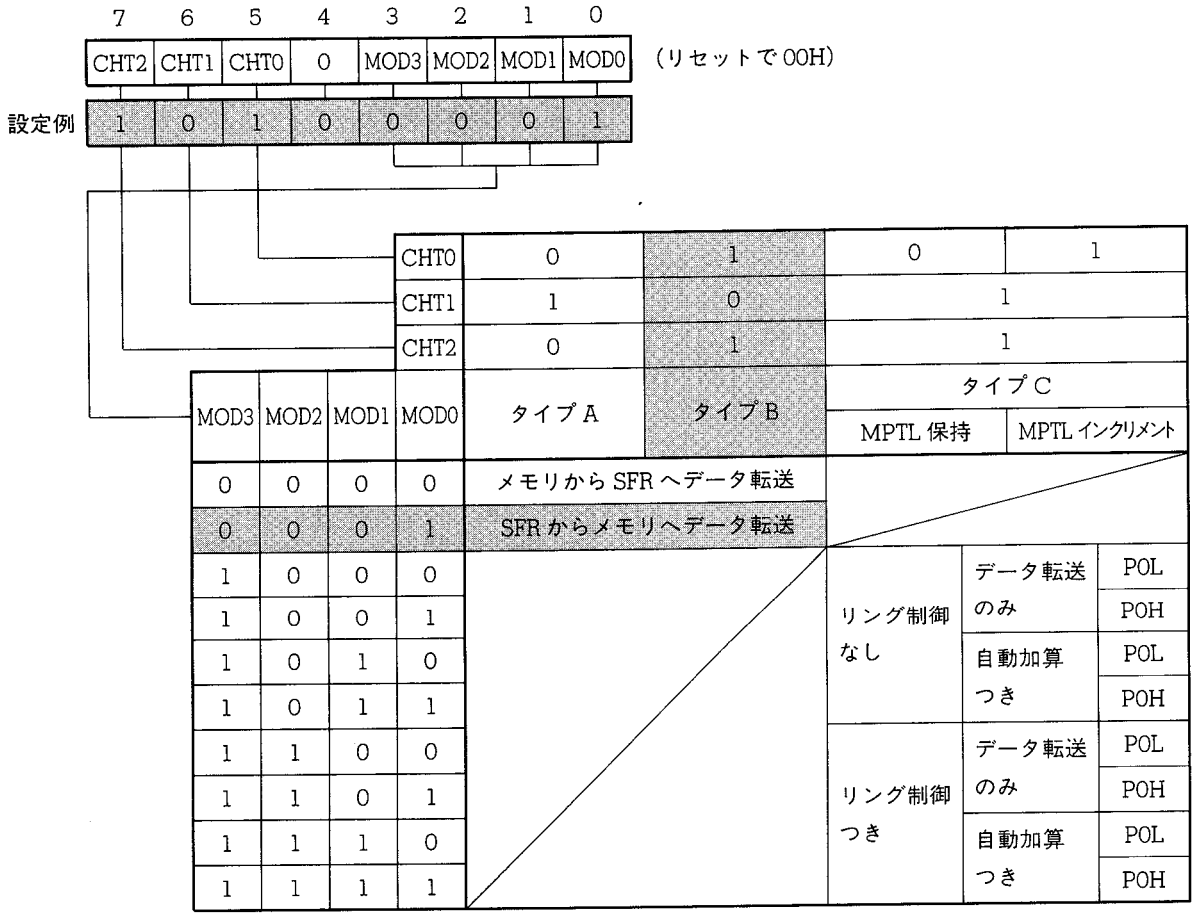
	7	6	5	4	3	2	1	0	
INTM1	0	0	ES51	ES50	ES41	ES40	ES31	ES30	（リセットで00H）
設定例	×	×	×	×	×	1	×	×	×：操作しません。

ES31	ES30	INTP3 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	立ち下がり，立ち上がり両エッジ

ES41	ES40	INTP4 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	INTC30 を選択
1	1	設定禁止

ES51	ES50	INTP5 端子入力，検出エッジ指定
0	0	立ち下がりエッジ
0	1	立ち上がりエッジ
1	0	設定禁止
1	1	

マクロ・サービス・モード・レジスタ

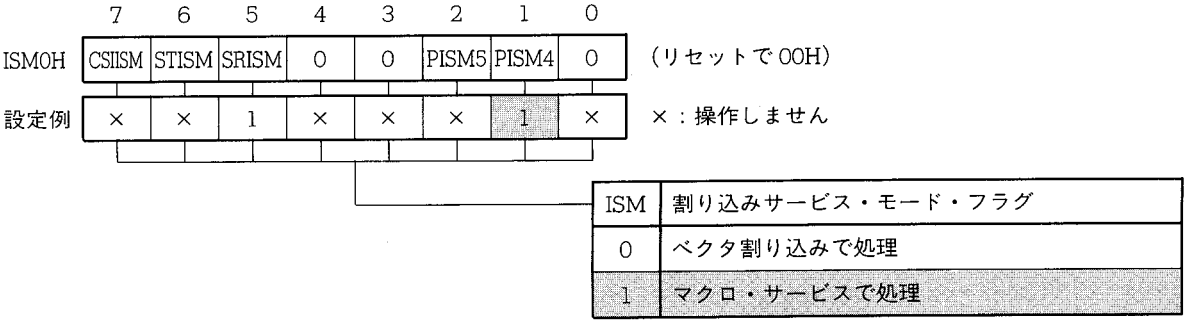


7	6	5	4	3	2	1	0	
CHT2	CHT1	CHT0	0	MOD3	MOD2	MOD1	MOD0	（リセットで00H）
設定例 1	0	1	0	0	0	0	1	

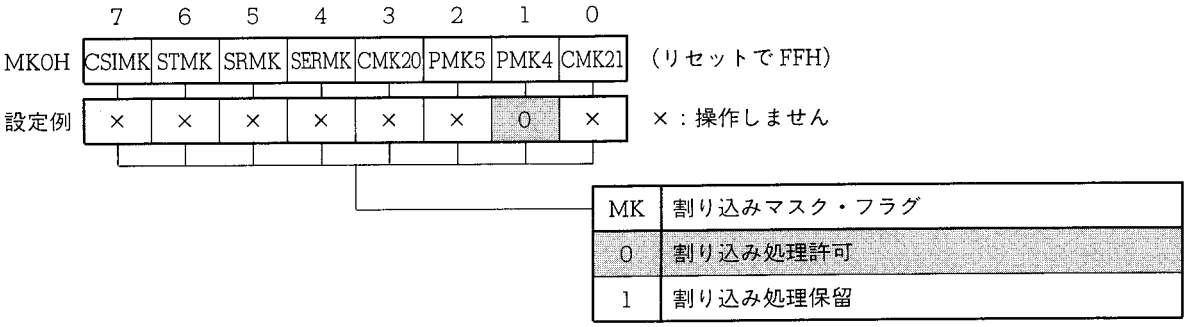
CHT0	0	1	0	1
CHT1	1	0	1	
CHT2	0	1	1	

MOD3	MOD2	MOD1	MOD0	タイプA	タイプB	タイプC	
						MPTL保持	MPTLインクリメント
0	0	0	0	メモリからSFRへデータ転送			
0	0	0	1	SFRからメモリへデータ転送			
1	0	0	0		リング制御なし	データ転送のみ	POL
1	0	0	1				POH
1	0	1	0			自動加算つき	POL
1	0	1	1				POH
1	1	0	0		リング制御つき	データ転送のみ	POL
1	1	0	1				POH
1	1	1	0			自動加算つき	POL
1	1	1	1				POH

割り込みサービス・モード・レジスタH



割り込みマスク・レジスタH



(4) 入出力パラメータ

●マクロ・サービス

(a) STRDTB　：マクロ・サービス・ポインタへの設定値です。転送先の先頭アドレスを設定します。

(b) CNTNMB：マクロ・サービス・カウンタへの設定値です。この設定回数分の転送が行われたのちにマクロ・サービス完了割り込みが発生します。この割り込みはINTP4によって起動されますので，ベクタ・テーブルの参照はINTP4のベクタ・アドレス(000EH)となります。

●ベクタ割り込み

(a) STRDTB　：パラレル入力データの書き込み先の先頭アドレスです。

(b) CNTNMB：ひとまとまりのパラレル・データ入力回数です。この設定値の回数の入力が完了すると，サンプリング完了フラグENDSMPがセット(1)されます。

(c) SPINTB　：転送先のアドレス［STRDTB］を格納するメモリのアドレスです。

(d) SCUNTB　：パラレル・データ入力回数［CNTNMB］を格納するメモリのアドレスです。

(e) ENDSMP　：ひとまとまりの入力が完了するとセット(1)されるフラグです。

(5) 使用レジスタ

●マクロ・サービス

なし

●ベクタ割り込み

X，A，HL

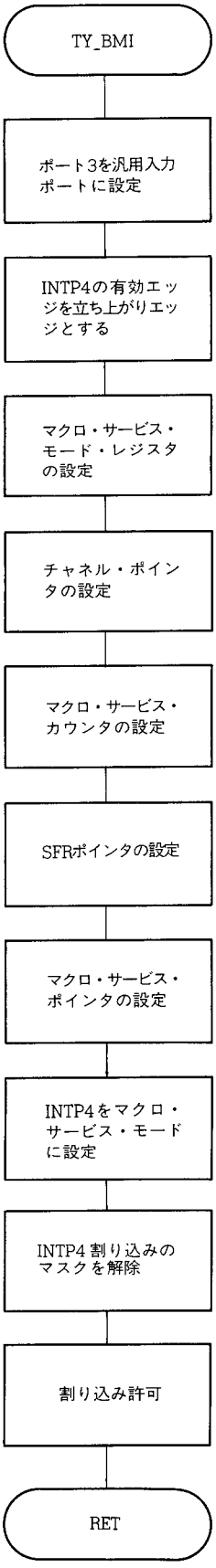
(6) プログラム使用例

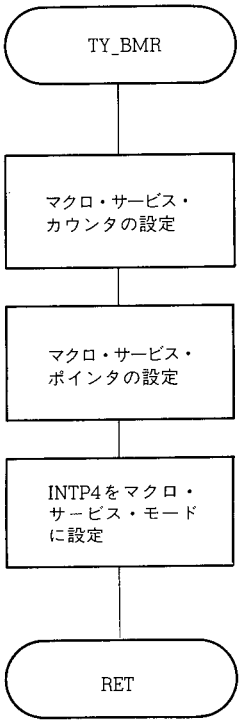
マクロ・サービスの場合についてプログラム例を示します。

```
;       --- FOR TYPE B ---
        PUBLIC  STRDTB,CNTNMB                   ; PARAMETER
        EXTRN   TY_BMI,TY_BMR                   ; PACKAGE
                .
                .
STRDTB  EQU     0A000H          ; sampling data store area
CNTNMB  EQU     20H             ; number of data
STRAREA DSEG    AT STRDTB
        DS      CNTNMB

INTP4VT CSEG    AT 0000EH
        DW      INTP4           ; INTP4 vector
                .
                .
TY_B_M:
                .
                .
        CALL    !TY_BMI         ; <<< TY_B_II >>>
TYB_L1:                         ; DUMMY LOOP
        BR      TYB_L1
;
INTP4:                          ; FOR END OF SAMPLING
        SEL     RB2             ; CHANGE REG BANK
                .
                .
        CALL    !TY_BMR         ; <<< TY_BMR >>>
        RETI
```

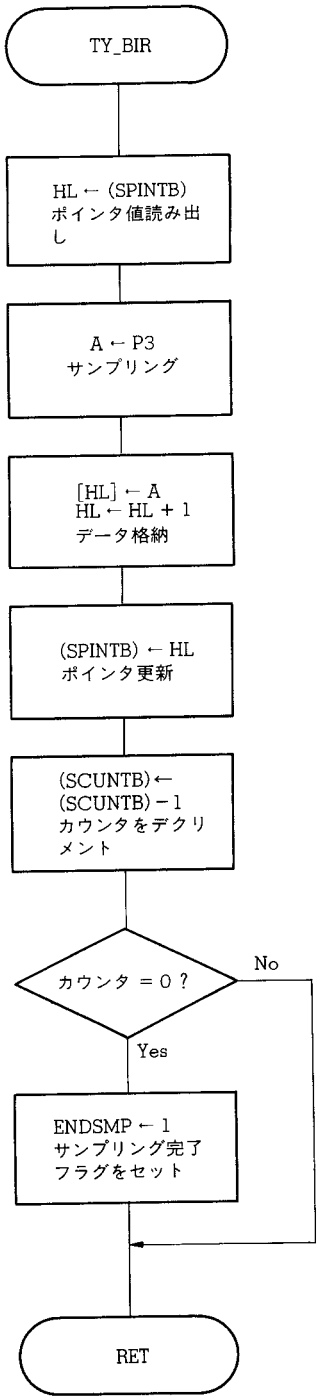
(7) フロー・チャート

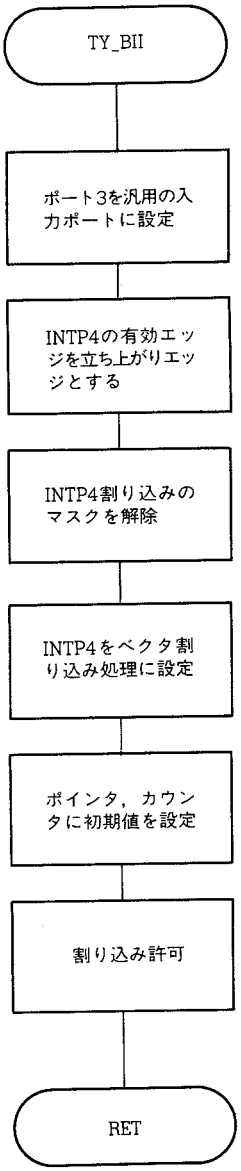
(a) マクロ・サービス





(b)　ベクタ割り込み


TY_BIR
HL ← (SPINTB)
ポインタ値読み出し
A ← P3
サンプリング
[HL] ← A
HL ← HL + 1
データ格納
(SPINTB) ← HL
ポインタ更新
(SCUNTB) ←
(SCUNTB) − 1
カウンタをデクリメント
カウンタ = 0 ?
No
Yes
ENDSMP ← 1
サンプリング完了フラグをセット
RET


TY_BII
ポート3を汎用の入力ポートに設定
INTP4の有効エッジを立ち上がりエッジとする
INTP4割り込みのマスクを解除
INTP4をベクタ割り込み処理に設定
ポインタ，カウンタに初期値を設定
割り込み許可
RET

(8) プログラム・リスト

(a) マクロ・サービス

```
        NAME    TYB_MR
;
;*************************************************************
;* macro service type-B application at macro service        *
;*      synchronous parallel data input                      *
;*************************************************************
;
        PUBLIC  TY_BMI,TY_BMR
        EXTRN   STRDTB,CNTNMB
;
MCHP4   DSEG    AT OFE90H
MSCP4:  DS      1               ; macro service counter
SFRP4:  DS      1               ; SFR pointer
MPP4P:  DS      2               ; macro service pointer
CHAP4   EQU     $-1

MCWP4   DSEG    AT OFED4H
MSMP4:  DS      1               ; INTP4 macro service mode register
CHPP4:  DS      1               ; INTP4 macro service channel pointer
;
PISM4   EQU     ISM0H.1         ; INTP4 interrupt service mode
PMK4    EQU     MK0H.1          ; INTP4 mask
;
;       *** initialize ***
;
        CSEG
TY_BMI:
        MOV     PMC3,#0         ; initialize P3
        MOV     PM3,#0FFH
        MOV     INTM1,#00000100B  ; INTP4 active edge is rise
        MOV     MSMP4,#10100001B
                                ; set macro service mode register
        MOV     CHPP4,#LOW(CHAP4)
                                ; set macro service channel pointer
        MOV     MSCP4,#LOW(CNTNMB)
                                ; set macro service counter
        MOV     SFRP4,#LOW(P3)  ; set SFR pointer
        MOVW    MPP4P,#STRDTB   ; set macro service memory pointer

        SET1    PISM4           ; initialize interrupt
        CLR1    PMK4
        EI
        RET
;
;       *** interrupt of complete macro-service ***
;
TY_BMR:
        MOV     MSCP4,#LOW(CNTNMB)
                                ; restore macro service counter
        MOVW    MPP4P,#STRDTB   ; restore macro service memory pointer
        SET1    PISM4           ; set interrupt service mode
        RET
;
        END
```

(b)　ベクタ割り込み

```
        NAME    TYB_IR
;
;*************************************************************
;* macro service type-B application at interrupt             *
;*      synchronous parallel data input                      *
;*************************************************************
;
        PUBLIC  TY_BII,TY_BIR
        EXTRN   STRDTB,CNTNMB,SPINTB,SCUNTB
        EXTBIT  ENDSMP

ES40    EQU     INTM1.2         ; INTP4 active edge
;
        CSEG
TY_BII:
        MOV     PMC3,#0         ; initialize port-3
        MOV     PM3,#0FFH

        SET1    ES40            ; INTP4 active edge is rise
        MOV     MK0H,#11111101B ; open INTP4 mask
        MOV     ISM0H,#00

        MOVW    SPINTB,#STRDTB  ; set store pointer
        MOV     SCUNTB,#LOW(CNTNMB)
                                ; set store counter
        EI
        RET
;
;       *** sampling data ***
;
TY_BIR:
        MOVW    AX,SPINTB       ; read store pointer
        MOVW    HL,AX
        MOV     A,P3            ; sampling
        MOV     [HL+],A         ; store data
        MOVW    AX,HL
        MOVW    SPINTB,AX       ; write new store pointer

        DEC     SCUNTB          ; increment store counter
        BNZ     $TYB_J1
        SET1    ENDSMP          ; set end of sampling
TYB_J1:
        RET
;
        END
```

7
214
218A
224
234
244

7.3　ステッピング・モータの開ループ制御　その1

マクロ・サービス・タイプCを用いて，リアルタイム出力ポートPOLに接続されている，4相ステッピング・モータの開ループ制御を行う例を示します。

この例ではf_{CLK}＝6 MHzとします。

(1) 動作概要

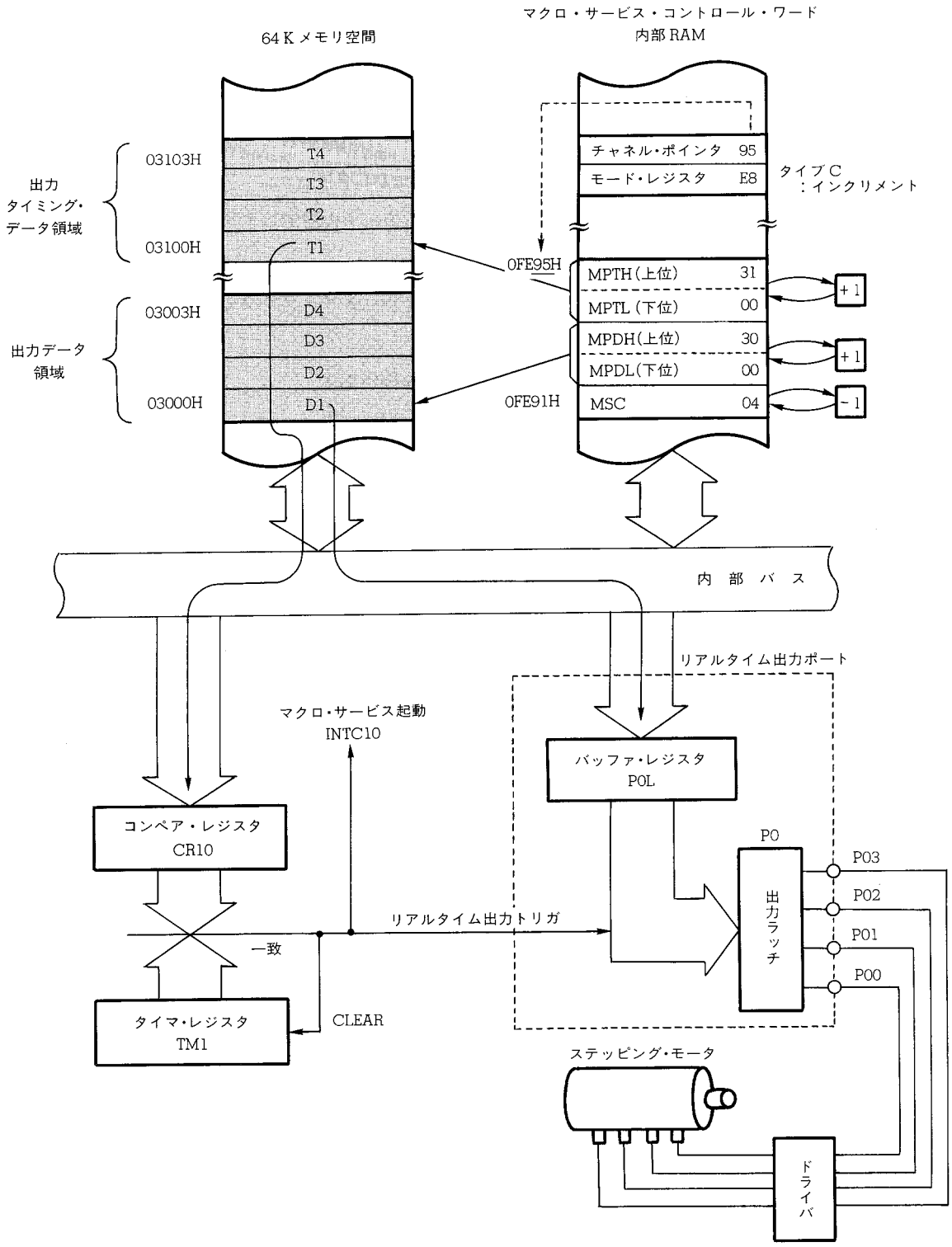
リアルタイム出力ポートを用いて，ステッピング・モータを回転させます。2系統のリアルタイム出力ポートの下位4ビット（POL）を用い，200 ppsで定速度回転させます。励磁方式は1相または2相励磁とします。したがって，パターン数は4です。

各ポインタの値として，チャネル・ポインタ（CHP）の値は95Hとし，マクロ・サービス・カウンタ（MSC）の値は4とします。また，マクロ・サービス・ポインタの値は，タイマ用，データ用それぞれMPT＝3100H，MPD＝3000Hとします。

マクロ・サービス完了割り込み処理は，それぞれの初期設定値を再設定して，連続回転を行います。

ここで用いるステッピング・モータの最小ステップ角は1.8度です。

図7-3　ステッピング・モータの開ループ制御のメモリ・マップ



(2) プログラム説明…(8) プログラム・リスト参照

●マクロ・サービス・イニシャライズ処理〔レーベル名称：TY_CMI〕

(a) ポート0に初期励磁データをセットします。

(b) ポート0を出力ポートに設定します。

(c) ポート0の下位4ビットをリアルタイム出力ポートに設定します。

(d) 8ビット・タイマ/カウンタ1（TM1）をリセットします。

(e) TM1とCR10の一致によるTM1をクリアするモードに設定します。

(f) TM1のカウント・クロックを設定します。

(g) マクロ・サービス・モード・レジスタの設定を行います。タイプCに設定します。

(h) チャネル・ポインタの設定を行います。

(i) データ用マクロ・サービス・ポインタの設定を行います。

(j) タイマ用マクロ・サービス・ポインタの設定を行います。

(k) マクロ・サービス・カウンタに励磁パターンの数を設定をします。

(l) 8ビット・タイマ/カウンタ1のカウントをスタートさせます。

(m) INTC10をマクロ・サービス・モードに設定します。

(n) INTC10のマスクを解除します。

(o) 割り込みを許可します。

●マクロ・サービス完了割り込み処理〔レーベル名称：INTC10〕

(a) データ用のマクロ・サービス・ポインタの再設定を行います。

(b) タイマ用のマクロ・サービス・ポインタの再設定を行います。

(c) マクロ・サービス・カウンタに励磁パターン数を再設定します。

(d) INTC10をマクロ・サービス・モードに再設定します。

(3) モード・レジスタ設定例

リアルタイム出力ポート・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
RTPC	BYTE	0注	0注	P0MH	EXTR	0注	0注	P0ML	（リセットで00H）
設定例	0	0	0	0	0	0	0	1	

P0ML	P0Lの機能指定
0	ポート・モード
1	リアルタイム出力ポート・モード

EXTR	INTP0によるバッファ・レジスタから出力ラッチへのデータ転送の許可
0	許可しない
1	許可する　BYTE = 0：P0Lのみ転送される BYTE = 1：P0L/Hが転送される

P0MH	P0Hの機能指定
0	ポート・モード
1	リアルタイム出力ポート・モード

BYTE	リアルタイム出力ポート・モードの動作モード
0	4ビット・セパレート・リアルタイム出力ポート
1	8ビット・リアルタイム出力ポート

注　ビット1，2，5，6には必ず“0”を書き込んでください。

ポート0モード・レジスタ

	7	6	5	4	3	2	1	0	
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	（リセットで00H）
設定例	0	0	0	0	0	0	0	0	

PM0n	P0n端子入出力モード指定（n = 0−7）
00H	出力モード（出力バッファ・オン）
FFH	ハイ・インピーダンス状態（出力バッファ・オフ）
上記以外	設定禁止

7
214
218A
224
234
244

タイマ・コントロール・レジスタ1

	7	6	5	4	3	2	1	0	
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0	(リセットで00H)
設定例	×	×	×	0	1	0	0	0	×：操作しません

8ビット・タイマ/カウンタ1

OVF1	TM1のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	TM1のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ2

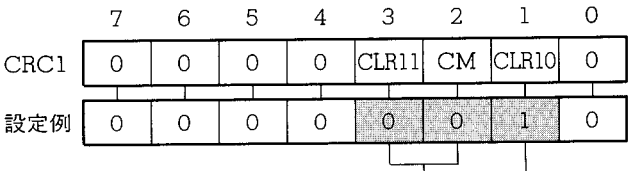
CMD2	TM2のカウント動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	TM2のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	TM2のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

キャプチャ/コンペア・コントロール・レジスタ1



CLR10	TM1とCR10レジスタの内容一致でのクリア動作
0	禁止
1	許可

CLR11	CM	CR11レジスタの動作指定	TM1のクリア動作
0	0	コンペア・レジスタ	禁止
1	0		許可(TM1とCR11レジスタの内容が一致したとき)
0	1	キャプチャ・レジスタ	禁止
1	1		許可(TM1の内容をCR11レジスタにキャプチャしたとき)

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	（リセットで00H）
設定例	0	0	0	0	0	1	1	1	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタ1のカウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$[注]
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタ3のカウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

備考　f_{CLK}：内部システム・クロック周波数（$f_{XX}/2$）

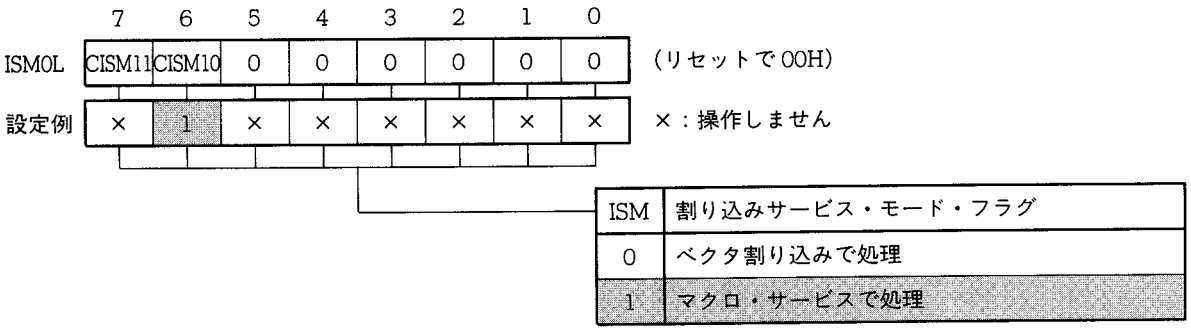
マクロ・サービス・モード・レジスタ

7	6	5	4	3	2	1	0	
CHT2	CHT1	CHT0	0	MOD3	MOD2	MOD1	MOD0	（リセットで00H）
設定例 1	1	1	0	1	0	0	0	

CHT0	0	1	0	1
CHT1	1	0	1	
CHT2	0	1	1	

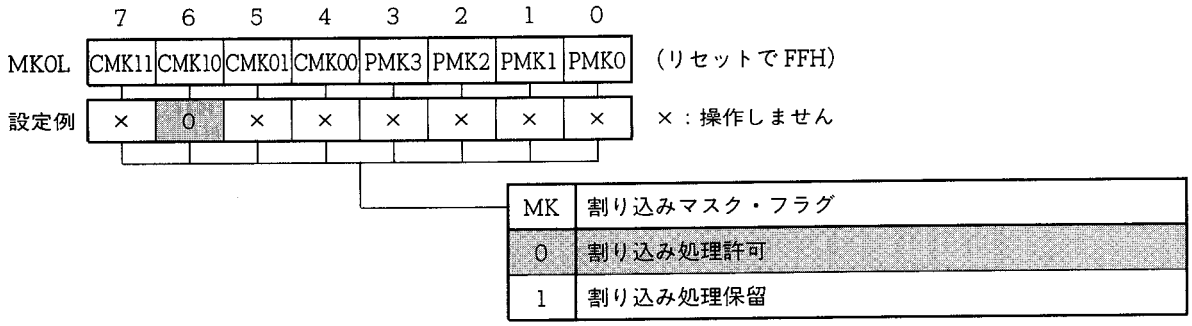
MOD3	MOD2	MOD1	MOD0	タイプA	タイプB	タイプC	
						MPTL保持	MPTLインクリメント
0	0	0	0	メモリからSFRへデータ転送			
0	0	0	1	SFRからメモリへデータ転送			
1	0	0	0			リング制御なし / データ転送のみ	POL
1	0	0	1				POH
1	0	1	0			リング制御なし / 自動加算つき	POL
1	0	1	1				POH
1	1	0	0			リング制御つき / データ転送のみ	POL
1	1	0	1				POH
1	1	1	0			リング制御つき / 自動加算つき	POL
1	1	1	1				POH

割り込みサービス・モード・レジスタL



ISM	割り込みサービス・モード・フラグ
0	ベクタ割り込みで処理
1	マクロ・サービスで処理

割り込みマスク・レジスタL



MK	割り込みマスク・フラグ
0	割り込み処理許可
1	割り込み処理保留

(4) 入出力パラメータ

この応用例では，マクロ・サービスに必要なパラメータを**(8) プログラム・リスト**に記述しているため，入出力パラメータはありません。なお，ステップ・レートの計算方法を次に示します。

ステップ・レートとは単位時間にステッピング・モータに与えられるパルス数です。すなわち，200 pps の場合には，1 秒間にステッピング・モータは 200 ステップ回転します。ここで，使用するステッピング・モータの最小ステップ角は 1.8 度ですから，200 pps は回転数に直すと 1 rps (60 rpm) となります。

したがって，1 つのパルスをステッピング・モータに与えてから次のパルスを与えるまでの間隔は 5 ms となりますから，INTC10 割り込み要求が 5 ms ごとに発生するように 8 ビット・タイマ/カウンタ 1 のコンペア・レジスタ CR10 に値を設定します。

CR10＝(5(ms)×TM1 のカウント・クロック)－1
　　＝(5(ms)×6(MHz)/512)－1
　　＝$(5\times10^{-3}\times6\times10^{6}/512)-1$
　　≒58.6－1
　　＝57.6 → 58

(5) 使用レジスタ

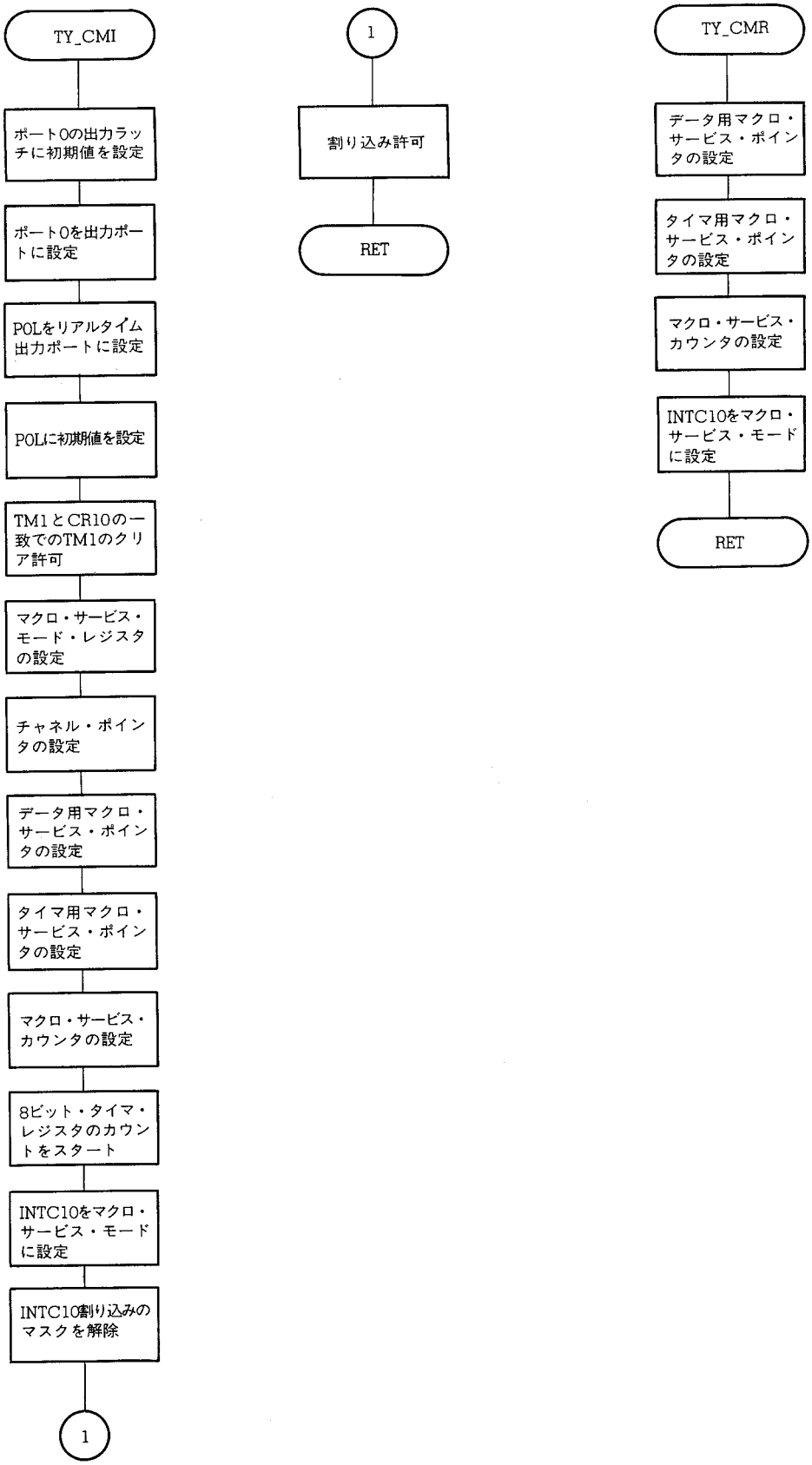
なし

(6) プログラム使用例

次にプログラム使用例を示します。

```
;       --- FOR TYPE C ---
;
        EXTRN   TY_CMI          ; PACKAGE

TY_C_M:
        CALL    !TY_CMI         ; <<< TY_CMI >>>
TYC_L1:
        BR      TYC_L1          ; DUMMY LOOP
```

(7) フロー・チャート


TY_CMI
ポート0の出力ラッチに初期値を設定
ポート0を出力ポートに設定
POLをリアルタイム出力ポートに設定
POLに初期値を設定
TM1とCR10の一致でのTM1のクリア許可
マクロ・サービス・モード・レジスタの設定
チャネル・ポインタの設定
データ用マクロ・サービス・ポインタの設定
タイマ用マクロ・サービス・ポインタの設定
マクロ・サービス・カウンタの設定
8ビット・タイマ・レジスタのカウントをスタート
INTC10をマクロ・サービス・モードに設定
INTC10割り込みのマスクを解除
1
1
割り込み許可
RET
TY_CMR
データ用マクロ・サービス・ポインタの設定
タイマ用マクロ・サービス・ポインタの設定
マクロ・サービス・カウンタの設定
INTC10をマクロ・サービス・モードに設定
RET

(8) プログラム・リスト

```
        NAME    TYC_MR
;
;*************************************************************
;* macro service type-C application at macro service       *
;*      for stepping motor control                         *
;*************************************************************
;
        PUBLIC  TY_CMI
;
MCHC10  DSEG    AT OFE91H
MSCC10: DS      1               ; macro service counter
MPDC10P:DS      2               ; macro service pointer for data
MPTC10P:DS      2               ; macro service pointer for timing
CHAC10  EQU     $-1             ; macro service channel address

MCWC10  DSEG    AT OFED8H
MSMC10: DS      1               ; INTC10 macro service mode register
CHPC10: DS      1               ; INTC10 channel pointer

CISM10  EQU     ISMOL.6
CMK10   EQU     MKOL.6
;
INTC10VT CSEG   AT 00018H
        DW      INTC10          ; INTC10 vector
;
;       *** initialize ***
;
        CSEG
TY_CMI:
        MOV     PO,#LOW(D4)     ; PO output latch <- 1st data
        MOV     PMO,#0          ; PO <- output port
        MOV     RTPC,#00000001B ; initialize real time output port
        MOV     POL,#LOW(D4)    ; 1st data

        MOV     CRC1,#00000010B ; enable clear TM1
        MOV     PRM1,#00000111B ; set TM1 prescaler fclk/512

        MOV     MSMC10,#11101000B
                                ; set macro service mode register
        MOV     CHPC10,#LOW(CHAC10)
                                ; set macro service channel pointer
        MOVW    MPDC10P,#PFDT   ; set memory pointer for data
        MOVW    MPTC10P,#PFTM   ; set memory pointer for timing
        MOV     MSCC10,#LOW(CYCNUM)
                                ; set macro service counter
        OR      TMC1,#00001000B ; timer start
        SET1    CISM10          ; initialize interrupt
        CLR1    CMK10
        EI
        RET
;
;       *** complete of macro service ***
;
INTC10:
        MOVW    MPDC10P,#PFDT   ; restore memory pointer for data
        MOVW    MPTC10P,#PFTM   ; restore memory pointer for timing
        MOV     MSCC10,#LOW(CYCNUM)
                                ; restore macro service counter
        SET1    CISM10          ; set interrupt service mode
        RETI
```

```
;
;       *** data profile ***
;
D1      EQU     00000011B       ; define output data
D2      EQU     00000110B
D3      EQU     00001100B
D4      EQU     00001001B
TN      EQU     58              ; define timing data
CYCNUM  EQU     4               ; number of step cycle

DTFLIE  CSEG    AT 0300OH
PFDT:   DB      D1,D2,D3,D4     ; output data
TMFILE  CSEG    AT 0310OH
PFTM:   DB      TN,TN,TN,TN     ; timing data
;
        END
```

7.4　ステッピング・モータの開ループ制御　その2

μPD78218Aシリーズ，μPD78234シリーズ，μPD78244シリーズは，μPD78214シリーズ，μPD78224シリーズに対してマクロ・サービス機能をアップしているため，マクロ・サービス・カウンタに16ビット・データを設定することができます。

この場合のマクロ・サービス・タイプCのプログラム例を示します。

(1) 動作概要

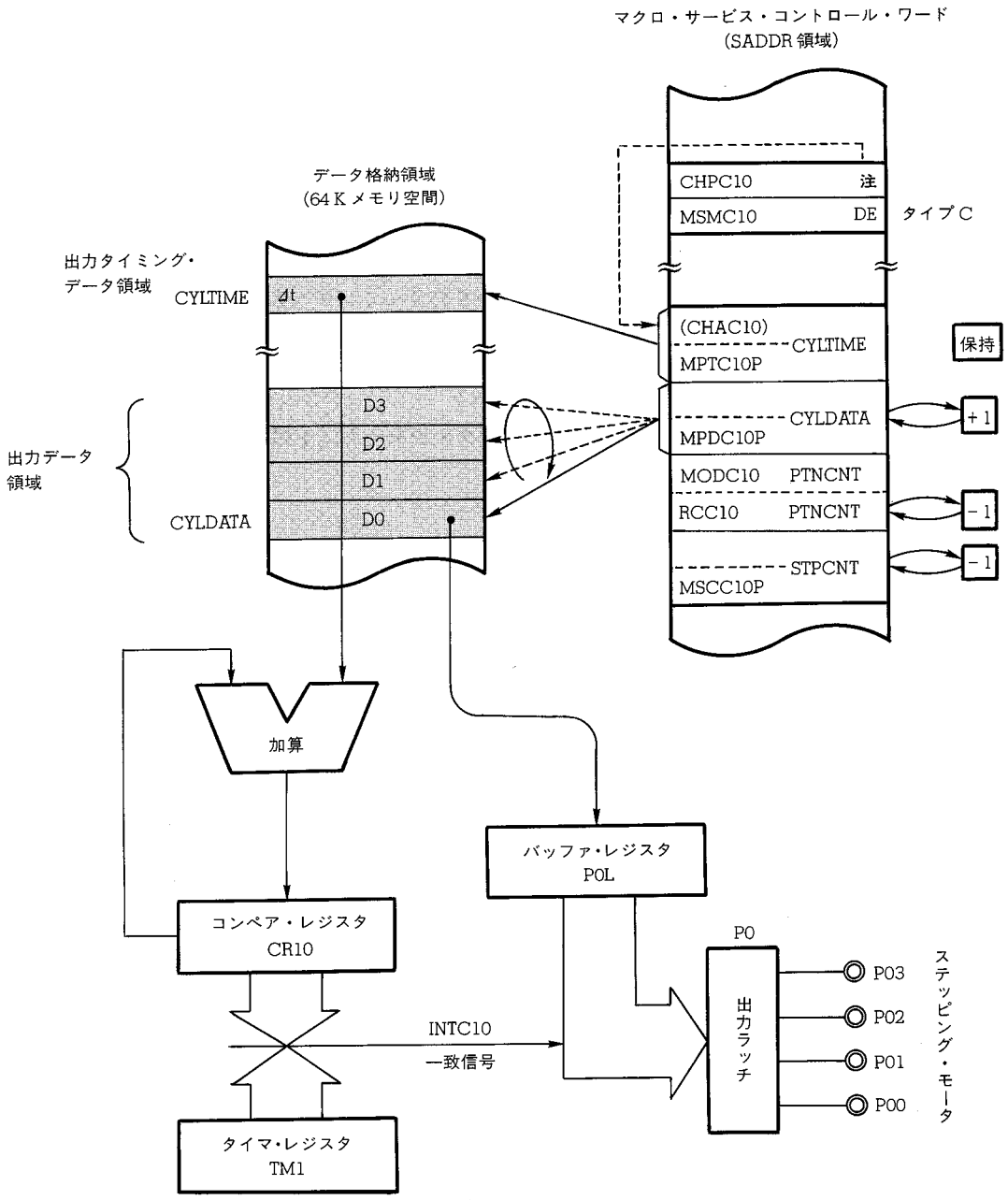
7.3　ステッピング・モータの開ループ制御　その1と同様の動作を，次のモードを使用して実現します。

- ●マクロ・サービス・カウンタ（MSC）＝16ビット
- ●タイミング・データ用ポインタ保持
- ●リング制御
- ●自動加算

なお，この応用例では，f_{CLK} ＝6MHzとします。また，8ビット・タイマ/カウンタ1のカウント・クロックは，$f_{CLK}/512$を選択します。

図7-4マクロ・サービスのメモリ・マップを示します。

図7-4　マクロ・サービス（MSC＝16ビットの場合）のメモリ・マップ



注 CHAC10（=MPTC10P＋1）の下位バイト

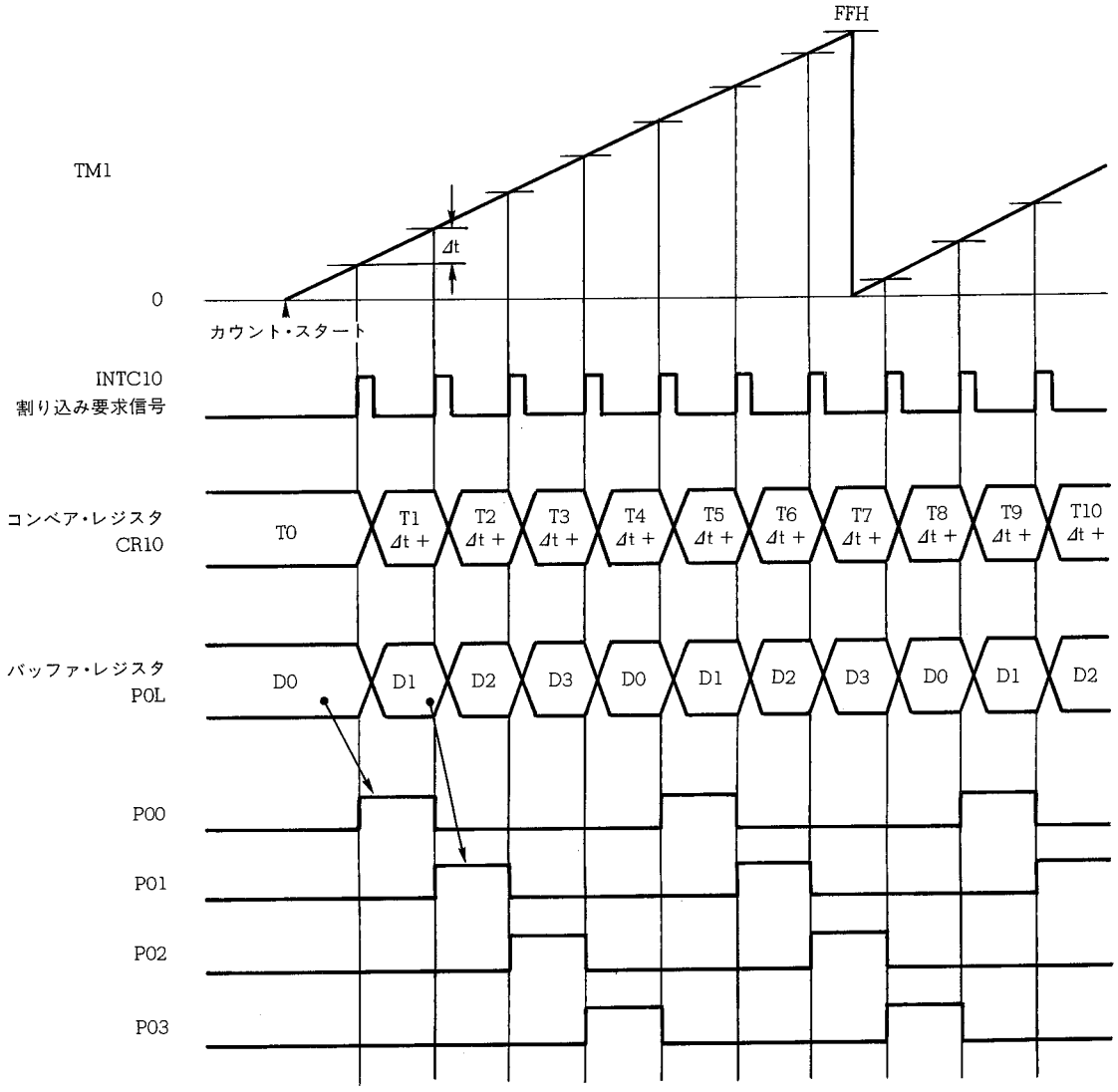
7
218A
234
244

ステッピング・モータを一定速度で回転させる場合には，マクロ・サービス完了割り込みで行う処理が，マクロ・サービスの再設定しかありませんので，1回のマクロ・サービス起動（CISM10←1）で，より多くのステップ数を動かした方が，完了割り込みの発生頻度が少ない分，CPUのサービス時間が向上します。

マクロ・サービス・カウンタ（MSC）の設定値としては，0が最大となり，65536回の転送動作が行われます。この場合，リング制御を使用し，リング・カウンタ(RC)，モジュロ・レジスタ(MOD)に励磁パターン数（1相，2相励磁の場合：4，1−2の相励磁の場合：8）を設定しておけば，励磁データのポインタが最終を越えても，自動的に元の位置に戻されます。

さらに，自動加算モードを使用すると，図7-5のタイミング・チャートのような動作が実現できます。

図7-5　自動加算モードのタイミング・チャート（1相励磁の場合）



タイミング・データ用ポインタを保持に設定しておくと，タイミング・データは，Δtの値が，前のコンペア・レジスタ（CR10）の値に加算されて出力されます。この場合タイマ（TM1）は，フリー・ランニング・モードで使用しますので，もう1系統のステッピング・モータを動作させる場合でも，INTC11マクロ・サービスを使用して，容易に行うことができます。

(2) プログラム説明

(a) 処理概要（「(8) フロー・チャート」参照）

(i) マクロ・サービスのイニシャライズ処理〔レーベル名称：TYC16_M〕

ポート0，リアルタイム出力ポート，8ビット・タイマ/カウンタ1，INTC10マクロ・サービスのイニシャライズを行います。

(ii) INTC10マクロ・サービス完了割り込み処理〔レーベル名称：INTC10〕

INTC10のサービス・モードをマクロ・サービスに再設定します。

この応用例では，マクロ・サービス・カウンタに0を初期設定しているため，完了割り込み発生時点では(MSC＝0なので)，マクロ・サービス・カウンタの再設定の必要がありません。

また，リング制御を使用しているので，マクロ・サービス・ポインタは，自動的に設定されるため，これも再設定の必要がありません。

(b) 使用するRAM

この応用例では，マクロ・サービス・チャネルとして，8バイトのRAMを使用します。表7-2に使用RAM一覧を示します。

表7-2　マクロ・サービス（MSC＝16ビット）の使用RAM一覧

RAM名称	配置領域	用　途	バイト数	初期値
MPTC10P	SADDR注	INTC10タイミング用 マクロ・サービス・ポインタ	2	CYLTIME
MPDC10P		INTC10データ用 マクロ・サービス・ポインタ	2	CYLDATA
MODC10		INTC10マクロ・サービス・モジュロ・レジスタ	1	PTNCNT
RCC10		INTC10マクロ・サービス・リング・カウンタ	1	PTNCNT
MSCC10P		INTC10マクロ・サービス・カウンタ	2	STPCNT

注　SADDR：ショート・ダイレクト・アドレシングの適用範囲（0FE20H-0FEFFH）

(3) 入出力パラメータ

なし

(4) 使用レジスタ

なし

(5) 使用スタック

この応用例のサブルーチン，および割り込み処理で使用するスタック・サイズは，3バイトです（ネスティング：1レベル）。

(6) モード・レジスタ設定例

ポート0モード・レジスタ

	7	6	5	4	3	2	1	0	
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	(リセットで00H)
設定例	0	0	0	0	0	0	0	0	

PM0n	P0n端子入出力モード指定 (n = 0-7)
00H	出力モード(出力バッファ・オン)
FFH	ハイ・インピーダンス状態(出力バッファ・オフ)
上記以外	設定禁止

リアルタイム出力ポート・コントロール・レジスタ

	7	6	5	4	3	2	1	0	
RTPC	BYTE	0注	0注	P0MH	EXTR	0注	0注	P0ML	(リセットで00H)
設定例	0	0	0	0	0	0	0	1	

P0ML	P0Lの機能指定
0	ポート・モード
1	リアルタイム出力ポート・モード

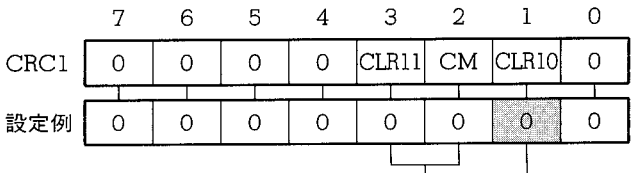
EXTR	INTP0によるバッファ・レジスタから出力ラッチへのデータ転送の許可
0	許可しない
1	許可する　BYTE = 0 : P0Lのみ転送される BYTE = 1 : P0L/Hが転送される

P0MH	P0Hの機能指定
0	ポート・モード
1	リアルタイム出力ポート・モード

BYTE	リアルタイム出力ポート・モードの動作モード
0	4ビット・セパレート・リアルタイム出力ポート
1	8ビット・リアルタイム出力ポート

注　ビット1, 2, 5, 6には必ず“0”を書き込んでください。

キャプチャ/コンペア・コントロール・レジスタ1



CLR10	TM1とCR10レジスタの内容一致でのクリア動作
0	禁止
1	許可

CLR11	CM	CR11レジスタの動作指定	TM1のクリア動作
0	0	コンペア・レジスタ	禁止
1	0		許可(TM1とCR11レジスタの内容が一致したとき)
0	1	キャプチャ・レジスタ	禁止
1	1		許可(TM1の内容をCR11レジスタにキャプチャしたとき)

7
218A
234
244

プリスケーラ・モード・レジスタ1

	7	6	5	4	3	2	1	0	
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10	（リセットで00H）
設定例	0	0	0	0	0	1	1	1	

8ビット・タイマ/カウンタ1

PRS12	PRS11	PRS10	タイマ/カウンタ1のカウント・クロック周波数の指定
0	0	0	$f_{CLK}/16$[注]
0	0	1	
0	1	0	
0	1	1	$f_{CLK}/32$
1	0	0	$f_{CLK}/64$
1	0	1	$f_{CLK}/128$
1	1	0	$f_{CLK}/256$
1	1	1	$f_{CLK}/512$

8ビット・タイマ/カウンタ2

PRS23	PRS22	PRS21	PRS20	タイマ/カウンタ3のカウント・クロック周波数の指定
0	0	0	0	$f_{CLK}/16$
0	0	0	1	
0	0	1	0	
0	0	1	1	$f_{CLK}/32$
0	1	0	0	$f_{CLK}/64$
0	1	0	1	$f_{CLK}/128$
0	1	1	0	$f_{CLK}/256$
0	1	1	1	$f_{CLK}/512$
1	1	1	1	外部クロック(CI)

備考　f_{CLK}：内部システム・クロック周波数（$f_{CLK}/2$）

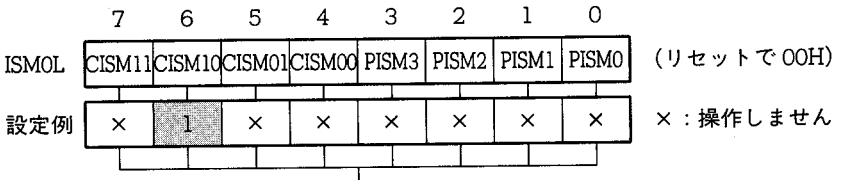
マクロ・サービス・モード・レジスタ

7	6	5	4	3	2	1	0	
CHT2	CHT1	CHT0	0	MOD3	MOD2	MOD1	MOD0	（リセットで00H）
設定例 1	1	0	1	1	1	1	0	

CHT0	0	1	0	1			
CHT1	1	0	1				
CHT2	0	1	1				
MOD3	MOD2	MOD1	MOD0	タイプA	タイプB	タイプC	
						MPTL保持	MPTLインクリメント
0	0	0	0	メモリからSFRへデータ転送			
0	0	0	1	SFRからメモリへデータ転送			
1	0	0	0			リング制御なし	データ転送のみ	POL
1	0	0	1					POH
1	0	1	0				自動加算つき	POL
1	0	1	1					POH
1	1	0	0			リング制御つき	データ転送のみ	POL
1	1	0	1					POH
1	1	1	0				自動加算つき	POL
1	1	1	1					POH

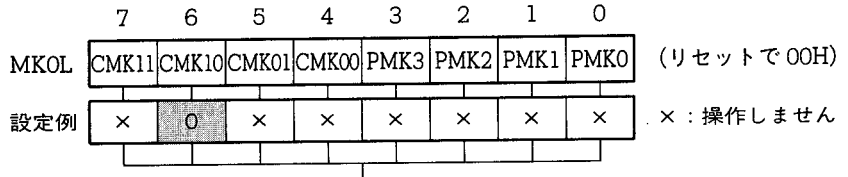
7
218A
234
244

割り込みサービス・モード・レジスタL



ISM	割り込みサービス・フラグ
0	ベクタ割り込みで処理
1	マクロ・サービスで処理

割り込みマスク・レジスタL



MK	割り込みマスク・フラグ
0	割り込み要求が発生していない。
1	割り込み要求発生。

タイマ・コントロール・レジスタ1

	7	6	5	4	3	2	1	0	
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0	(リセットで00H)
設定例	0	0	0	0	1	0	0	0	

8ビット・タイマ/カウンタ1

OVF1	TM1のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE1	TM1のカウント動作制御
0	クリアしたままカウント動作停止
1	カウント動作許可

8ビット・タイマ/カウンタ2

CMD2	TM2のカウント動作モード指定
0	通常モード
1	ワンショット・モード

OVF2	TM2のオーバフロー・フラグ
0	オーバフローなし
1	オーバフロー(FFHから00Hへのカウント・アップ)

備考　このビットはソフトウエアでのみリセットされます。

CE2	TM2のカウント動作制御
0	クリアしたまま，カウント動作停止
1	カウント動作許可

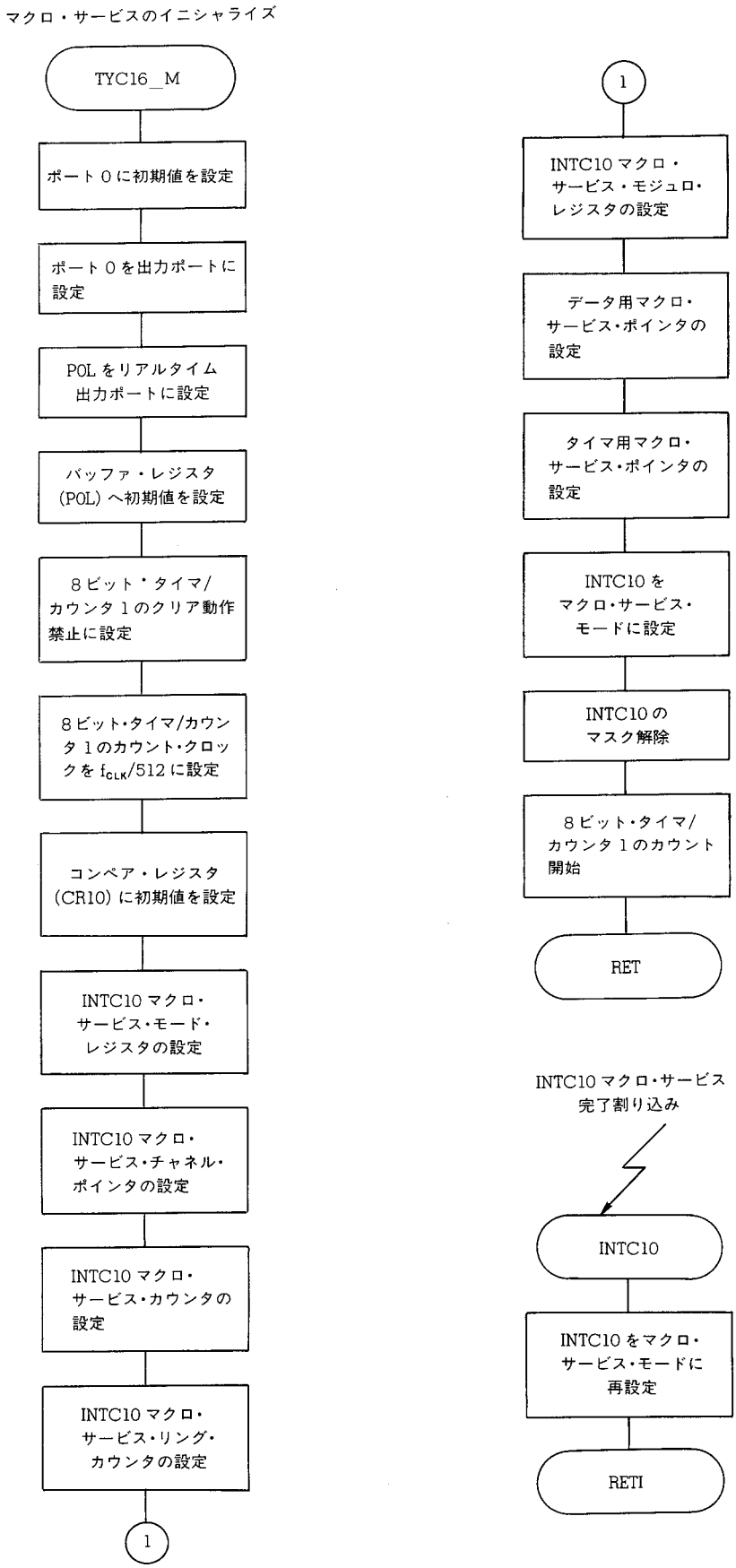
7
218A
234
244

(7) プログラム使用例

このプログラム例の使用例を示します。

```
        EXTRN   TYC16_M                 ; package
                :
                :
        CSEG
MAIN:
                :
                :
        CALL    !TYC16_M                ; call TYC16_M routine
        EI                              ; enable interrupt
                :
                :
```

(8) フロー・チャート



7
218A
234
244

(9) プログラム・リスト

このプログラム例では，ソース・プログラムの先頭で，ベクタ・テーブル・アドレスとマクロ・サービスの領域の定義用ファイル(INTMS.DEF)，およびSFRビット名定義ファイル(SFRBIT.DEF)をインクルードしています。

```
$       IC(INTMS.DEF)
        NAME    TYC16M
;*************************************************************
;*      MACRO SERVICE TYPE C MSC=16bit                       *
;*                              uPD78234,uPD78244            *
;*              for stepping motor control                   *
;*************************************************************

$       IC(SFRBIT.DEF)
        PUBLIC  TYC16_M                 ; package

STPCNT  EQU     0                       ; step count (65536)
PTNCNT  EQU     4                       ; pattern count

;       *** define INTC10 macro service area ***

        DSEGCR16 C10,'SADDR'            ; macro service channel(MSC=16bit)
        CMCW10                          ; macro service control word

;       *** define vector table ***

        CVENT10                         ; INTC10 vector table

TYC16MS CSEG
TYC16_M:
        MOV     P0,#00001000B           ; P0 output latch <- first data
        MOV     PM0,#00000000B          ; set P0 output port
        MOV     RTPC,#00000001B         ; initialize realtime output port
        MOV     P0L,#00001000B          ; set P0 buffer register first data

        MOV     CRC1,#00000000B         ; disable clear TM1
        MOV     PRM1,#00000111B         ; set TM1 prescaler fCLK/512
        MOV     CR10,#0                 ; set first timing data

        MOV     MSMC10,#11011110B       ; set mode register (TYPE C)
        MOV     CHPC10,#LOW CHAC10      ; set channel pointer
        MOVW    MSCC10P,#STPCNT         ; set macro service counter
        MOV     RCC10,#PTNCNT           ; set ring counter
        MOV     MODC10,#PTNCNT          ; set modulo register
        MOVW    MPDC10P,#CYLDATA        ; set pointer for data
        MOVW    MPTC10P,#CYLTIME        ; set pointer for timing
```

```
        SET1    CISM10                  ; set INTC10 macro service mode
        CLR1    CMK10                   ; open INTC10 mask

        MOV     TMC1,#00001000B         ; start TM1
        RET

;*****************************************
;*      end of INTC10 macro service      *
;*****************************************

INTC10:
        SET1    CISM10                  ; set INTC10 macro service mode again
        RETI

;*****************************************
;*      define data                      *
;*****************************************

;       *** for data ***

CYLDATA:
        DB      00000001B
        DB      00000010B
        DB      00000100B
        DB      00001000B

;       *** for timing ***

CYLTIME:
        DB      58

        END
```

7
218A
234
244

第8章　A/Dコンバータのプログラム例（μPD78214）

μPD78214は，6マルチプレクスト・アナログ入力（AN0-AN7）をもつアナログ/ディジタル（A/D）コンバータを内蔵しています。

変換方式は逐次比較で，変換結果を8ビットのA/D変換結果レジスタ（ADCR）に保持します。A/D変換のモードには次の2つのモードがあります。

○スキャン・モード：複数のアナログ入力を順次選択し，各端子からの変換データを得ます。
○セレクト・モード：アナログ入力を1端子に固定し，連続的な変換値を得ます。

ここでは，AN0-AN3入力をスキャンするモードとし，各チャネルとも，16回サンプリングするごとに平均値を求めるプログラム例を紹介します。

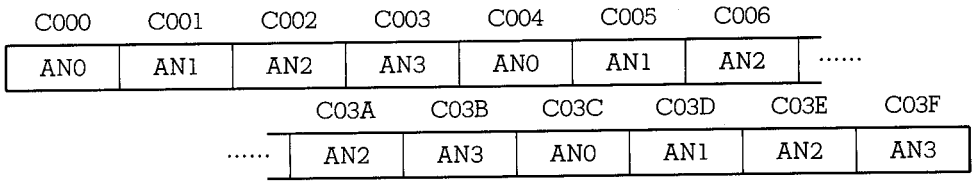
(1) 動作概要

使用メモリは，C000H-C03FHをサンプリング・データ格納領域とし，C040H-C043Hを平均値の格納領域とします。

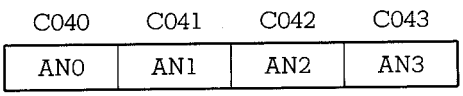
A/Dコンバータの変換時間は30 μs（f_{CLK}=6 MHz）を使用しています。したがって，CPUのオーバ・ヘッドを抑えるためにサンプリングは，マクロ・サービス機能のタイプBを用いて行い，平均値の算出は，マクロ・サービス完了割り込み処理内で行います。マクロ・サービス完了割り込み処理では，変換動作は停止します。

図8-1　A/Dコンバータのプログラム例の使用メモリ

(a)　変換結果格納領域



(b)　平均値格納領域



(2) プログラム説明…(6) プログラム・リスト参照

●システム・イニシャライズ処理〔レーベル名称：ADMAIN〕

(a) メモリ関連の各モード・レジスタの設定を行います。

(b) ポート6モード・レジスタに0を設定して0000H-FFFFHのバンクを選択します。

(c) スタック・ポインタをFC00Hに設定します。

(d) INTADマクロ・サービスのイニシャライズを行います。

●INTADマクロ・サービスのイニシャライズ処理〔レーベル名称：MS_AD〕

(a) マクロ・サービス・モード・レジスタの設定を行います。

タイプB，SFR→メモリに設定します。

(b) チャネル・ポインタの設定を行います。

(c) マクロ・サービス・カウンタに40Hを設定します。

(d) SFRポインタにADCRのアドレスの下位8ビットとして，6AHを設定します。

(e) マクロ・サービス・ポインタに変換結果領域の先頭アドレスC000Hを設定します。

(f) INTADをマクロ・サービス・モードに設定します。

(g) INTAD割り込みのマスクを解除します。

(h) 割り込みを許可します。

(i) AN0-AN3のスキャン・モードとして，A/Dコンバータの変換動作をスタートします。

●INTADマクロ・サービス完了割り込み処理〔レーベル名称：END_AD〕

(a) A/Dコンバータの変換動作を停止します。

(b) A/D変換終了割り込み要求フラグ（ADIF）をクリア（0）します。

(c) レジスタを退避します。

(d) それぞれのチャネルの変換結果の合計を求めます。

(e) 合計値を右に4回シフトすることにより，平均値を算出します。

(f) 平均値をC040H-C043Hに格納します。

(g) マクロ・サービス・カウンタとマクロ・サービス・ポインタを再設定します。

(h) レジスタを復帰します。

(i) INTADをマクロ・サービス・モードに設定します。

(j) A/Dコンバータの変換動作を再スタートします。

(3) モード・レジスタ設定例

メモリ拡張モード・レジスタ

	7	6	5	4	3	2	1	0	
MM	0	MM6	PW21	PW20	0	0	0	0	（リセットで20H）
設定例	0	1	1	0	0	0	0	0	

PW21	PW20	ウエイト数（指定範囲00000H-0FE7FH）
0	0	0ウエイト
0	1	1ウエイト
1	0	2ウエイト
1	1	外部ウエイト端子によるウエイト数

MM6	1Mバイト拡張制御
0	P60-P63は汎用出力ポート
1	P60-P63は外部メモリ拡張モード時の上位アドレス（A16-A19）の出力端子として機能する。

プログラマブル・ウエイト制御レジスタ

	7	6	5	4	3	2	1	0	
PM	PW31	PW30	0	0	0	0	0	0	（リセットで80H）
設定例	1	0	0	0	0	0	0	0	

PW31	PW30	ウエイト数（指定範囲10000H-FFFFFH）
0	0	0ウエイト
0	1	1ウエイト
1	0	2ウエイト
1	1	外部ウエイト端子によるウエイト数

8
214
218A
234
244

リフレッシュ・モード・レジスタ

	7	6	5	4	3	2	1	0	
RFM	RFLV	0	0	RFEN	0	0	RFT1	RFT0	（リセットで00H）
設定例	0	0	0	0	0	0	0	0	

f_{CLK} = 6 MHz

PFT1	PFT0	リフレッシュ・パルス出力周期指定
0	0	$16/f_{CLK}$ (2.6 μs)
0	1	$32/f_{CLK}$ (5.3 μs)
1	0	$64/f_{CLK}$ (10.7 μs)
1	1	$128/f_{CLK}$ (21.3 μs)

備考　f_{CLK}：内部システム・クロック

RFLV	RFEN	$\overline{\text{RFREQ}}$ 端子出力制御
×	0	ポート・モード
0	1	セルフ・リフレッシュ動作（$\overline{\text{REFRQ}}$ ロウ・レベル出力）
1		リフレッシュ・パルス出力許可

備考　×は，1または0

ポート6モード・レジスタ

	7	6	5	4	3	2	1	0	
PM6	PM67	PM66	−	−	PM63	PM62	PM61	PM60	（リセットでFFH）
設定例	0	0	0	0	0	0	0	0	

メモリ拡張モード・レジスタ(MM)のMM6ビットをセット（1）することによって“&”なしメモリ参照命令実行時のバンク・レジスタとして機能

PM6n	P6n端子入出力モード指定(n = 6, 7)
0	出力モード（出力バッファ・オン）
1	入力モード（出力バッファ・オフ）

注意　“−”は，1でも0でもかまいません。

マクロ・サービス・モード・レジスタ

7	6	5	4	3	2	1	0	
CHT2	CHT1	CHT0	0	MOD3	MOD2	MOD1	MOD0	（リセットで00H）
設定例 1	0	1	0	0	0	0	1	

CHT0	0	1	0	1		
CHT1	1	0	1			
CHT2	0	1	1			
MOD3	MOD2	MOD1	MOD0	タイプA	タイプB	タイプC	
						MPTL保持	MPTLインクリメント
0	0	0	0	メモリからSFRへデータ転送			
0	0	0	1	SFRからメモリへデータ転送			
1	0	0	0			リング制御なし	データ転送のみ　POL
1	0	0	1				POH
1	0	1	0				自動加算つき　POL
1	0	1	1				POH
1	1	0	0			リング制御つき	データ転送のみ　POL
1	1	0	1				POH
1	1	1	0				自動加算つき　POL
1	1	1	1				POH

A/D コンバータ・モード・レジスタ

	7	6	5	4	3	2	1	0	
ADM	CS	TRG	0	FR	ANI2	ANI1	ANI0	MS	（リセットで 20H）
設定例	1	0	0	0	0	1	1	0	

ANI2	ANI1	ANI0	MS	A/D 変換動作モードの指定	
0	0	0	0	スキャン・モード	AN0 入力をスキャン
0	0	1	0		AN0, AN1 入力をスキャン
0	1	0	0		AN0-AN2 入力をスキャン
0	1	1	0		AN0-AN3 入力をスキャン
1	0	0	0		AN0-AN4 入力をスキャン
1	0	1	0		AN0-AN5 入力をスキャン
1	1	0	0		AN0-AN6 入力をスキャン
1	1	1	0		AN0-AN7 入力をスキャン
0	0	0	1	セレクト・モード	AN0 入力をセレクト
0	0	1	1		AN1 入力をセレクト
0	1	0	1		AN2 入力をセレクト
0	1	1	1		AN3 入力をセレクト
1	0	0	1		AN4 入力をセレクト
1	0	1	1		AN5 入力をセレクト
1	1	0	1		AN6 入力をセレクト
1	1	1	1		AN7 入力をセレクト

FR	変換時間制御	
0	$180/f_{CLK}$ 注	$f_{CLK} > 4$ MHz
1	$120/f_{CLK}$ 注	$f_{CLK} \leqq 4$ MHz

TRG	外部端子トリガの制御
0	外部トリガ禁止
1	外部トリガ許可

CS	A/D 変換動作制御
0	A/D 変換動作停止
1	A/D 変換動作開始

注　f_{CLK}：システム・クロック周波数

割り込みサービス・モード・レジスタ H

	7	6	5	4	3	2	1	0	
ISM0H	CSIISM	STISM	SRISM	0	0	PISM5	PISM4	0	（リセットで 00H）
設定例	×	×	×	×	×	1	×	×	×：操作しません

ISM	割り込みサービス・モード・フラグ
0	ベクタ割り込みで処理
1	マクロ・サービスで処理

割り込みマスク・レジスタ H

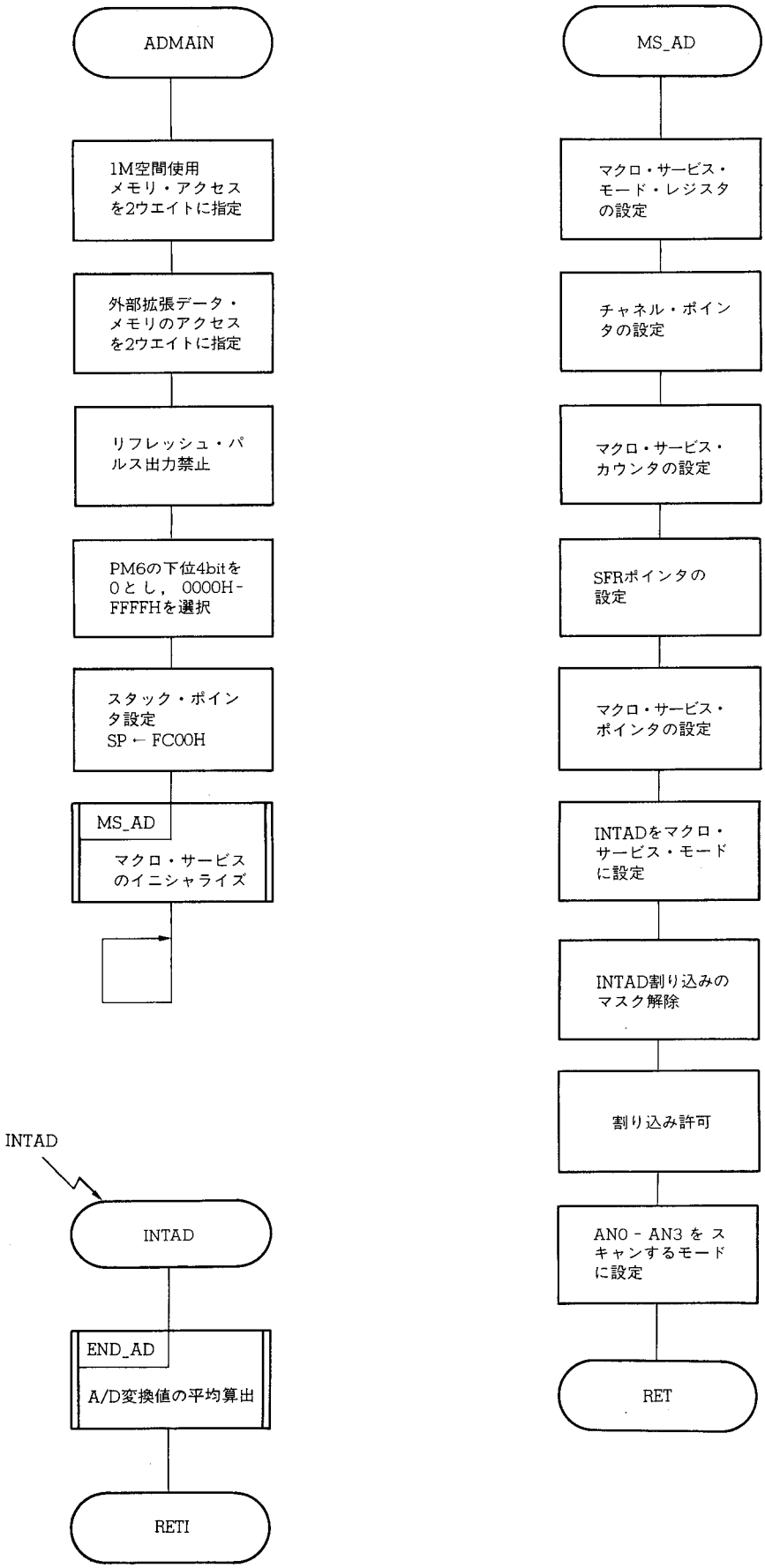
	7	6	5	4	3	2	1	0	
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21	（リセットで FFH）
設定例	×	×	×	×	×	0	×	×	×：操作しません

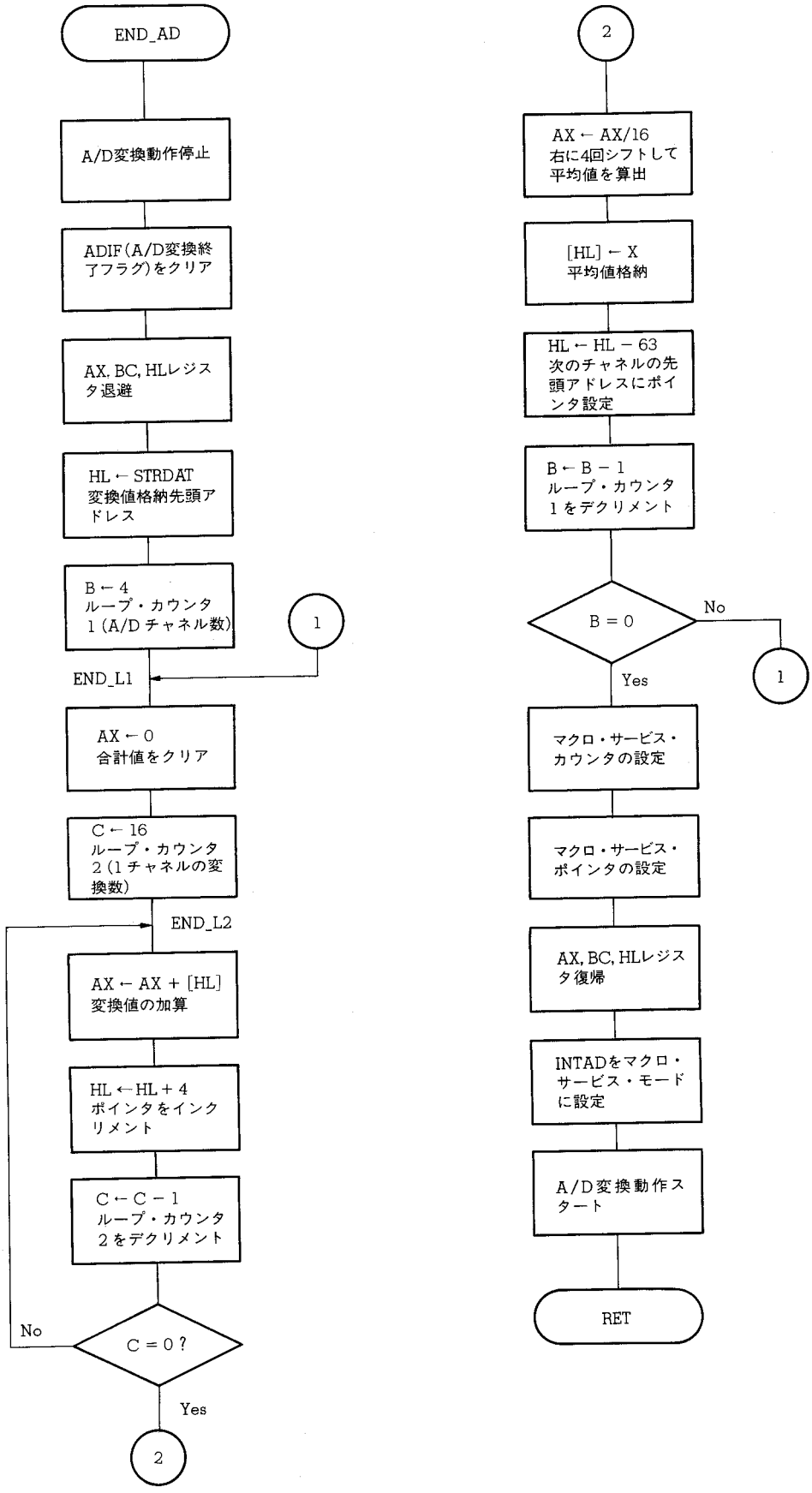
MK	割り込みマスク・フラグ
0	割り込み処理可能
1	割り込み処理保留

(4) 入出力パラメータ，使用レジスタ

アブソリュート・プログラムですから，省略します。

(5) フロー・チャート


ADMAIN
1M空間使用
メモリ・アクセス
を2ウエイトに指定
外部拡張データ・
メモリのアクセス
を2ウエイトに指定
リフレッシュ・パ
ルス出力禁止
PM6の下位4bitを
0とし，0000H-
FFFFHを選択
スタック・ポイン
タ設定
SP ← FC00H
MS_AD
マクロ・サービス
のイニシャライズ
MS_AD
マクロ・サービス・
モード・レジスタ
の設定
チャネル・ポイン
タの設定
マクロ・サービス・
カウンタの設定
SFRポインタの
設定
マクロ・サービス・
ポインタの設定
INTADをマクロ・
サービス・モード
に設定
INTAD割り込みの
マスク解除
割り込み許可
AN0 - AN3 を ス
キャンするモード
に設定
RET
INTAD
INTAD
END_AD
A/D変換値の平均算出
RETI



8
214
218A
234
244

(6) プログラム・リスト

```
        NAME    MS_ADR

;*************************************************************
;       A/D Converter application by macro service
;*************************************************************

STACK   EQU     0FC00H
M214    EQU     01000111B       ; uPD78214 MM data

PWDAT   EQU     10000000B       ; programable wait
RFDAT   EQU     10010000B       ; refresh mode
PM6DAT  EQU     0F0H            ; PM6 data

MSMAD   EQU     0FED2H          ; INTAD macro service mode register
CHPAD   EQU     0FED3H          ; INTAD macro service channel pointer
MSCAD   EQU     0FECEH          ; INTAD macro service counter
SFRAD   EQU     0FECFH          ; INTAD SFR pointer
MPADP   EQU     0FED0H          ; INTAD memory pointer

ADISM   EQU     ISM0H.2         ; ISMAD flag
CSAD    EQU     ADM.7           ; A/D convert start bit
ADMK    EQU     MK0H.2          ; INTAD mask flag
ADIF    EQU     IF0H.2          ; INTAD flag
;
;       --- for data area ---
;
CNTNUM  EQU     40H
LP2     EQU     16
LP1     EQU     CNTNUM/LP2
AD_D    DSEG    AT      0C000H  ; result store area
STRDAT: DS      CNTNUM
;
;       --- vector table ---
;
RSTVT   CSEG    AT 00000H
        DW      RST             ; reset
INTADVT CSEG    AT 00010H
        DW      INTAD           ; INTAD
;
;       --- main ---
;
AD_C    CSEG
        ORG     0080H
RST:
        MOV     MM,#M214        ; set memory mapping register
        MOV     PW,#PWDAT       ; set programable wait register
        MOV     RFM,#RFDAT      ; set refresh mode register
        MOV     PM6,#PM6DAT
        MOVW    SP,#STACK       ; set stack pointer

        CALL    !MS_AD          ; macro service initialize for INTAD
DMLP:                           ; dummy loop
        BR      DMLP
;
INTAD:                          ; complete of INTAD macro service
        CALL    !END_AD
        RETI
```

```
;
;       *** initialize ***
;
MS_AD:
        MOV     MSMAD,#10100001B
                                ; set macro service mode register
        MOV     CHPAD,#0D1H     ; set macro service channel pointer
        MOV     MSCAD,#CNTNUM   ; set macro service counter
        MOV     SFRAD,#LOW(ADCR)  ; set SFR pointer
        MOVW    MPADP,#STRDAT   ; set macro service memory pointer
;
        SET1    ADISM           ; set interrupt mode
        CLR1    ADMK            ; open mask of INTAD
        EI                      ; interrupt enable
        MOV     ADM,#10000110B  ; initialize A/D converter
        RET
;
;       *** interrupt of complete macro-service ***
;
END_AD:
        CLR1    CSAD            ; A/D converter stop
        CLR1    ADIF            ; INTAD flag clear
        PUSH    AX              ; save register
        PUSH    BC
        PUSH    HL

        MOVW    HL,#STRDAT      ; pointer set
        MOV     B,#LP1          ; loop counter_1 set (4)
EAD_L1:
        MOVW    AX,#0           ; sum clear
        MOV     C,#LP2          ; loop counter_2 set (16)
EAD_L2:
        XCH     A,X             ; addition result
        ADD     A,[HL]
        XCH     A,X
        ADDC    A,#0            ; carry addition

        INCW    HL              ; pointer <- pointer + 4
        INCW    HL
        INCW    HL
        INCW    HL
        DBNZ    C,$EAD_L2       ; check loop counter-2

        SHRW    AX,4            ; average of result
        XCH     A,X
        MOV     [HL],A
        MOVW    AX,HL           ; next pointer set
        SUBW    AX,#63          ; 16*4-1
        MOVW    HL,AX
        DBNZ    B,$EAD_L1       ; check loop counter-1

        MOV     MSCAD,#CNTNUM   ; set macro service counter
        MOVW    MPADP,#STRDAT   ; set macro service memory pointer

        POP     HL              ; restore register
        POP     BC
        POP     AX
        SET1    ADISM
        SET1    CSAD            ; A/D convert start

        RET

        END
```

8
214
218A
234
244

第９章　コンパレータのプログラム例（μPD78224）

μPD78224，78220はスレッシュ・ホールド電圧可変機能付きの８ビット入力端子（ポートT）を内蔵しています。

ここでは，コンパレータを使用して，電圧を測定するプログラムを紹介します。

(1) 動作概要

PT4端子への入力電圧をバイナリ・サーチによって４ビット分解能で逐次変換します。

結果をアキュームレータに格納します。逐次変換方式を図９-１に示します。PT4端子の印加電圧を V_{PT4}，比較基準電圧を V_{DD} とします。

(i) 比較電圧 $V_{DA} = V_{DD}/2$ とします。

$V_{DA} > V_{PT4}$ なので，MSB = 0 とします（$V_{DD}/2$ビット = 0）。

(ii) $V_{DA} = V_{DD}/4$ とします。

$V_{DA} < V_{PT4}$ なので $V_{DD}/4$ ビットを 1 とします。

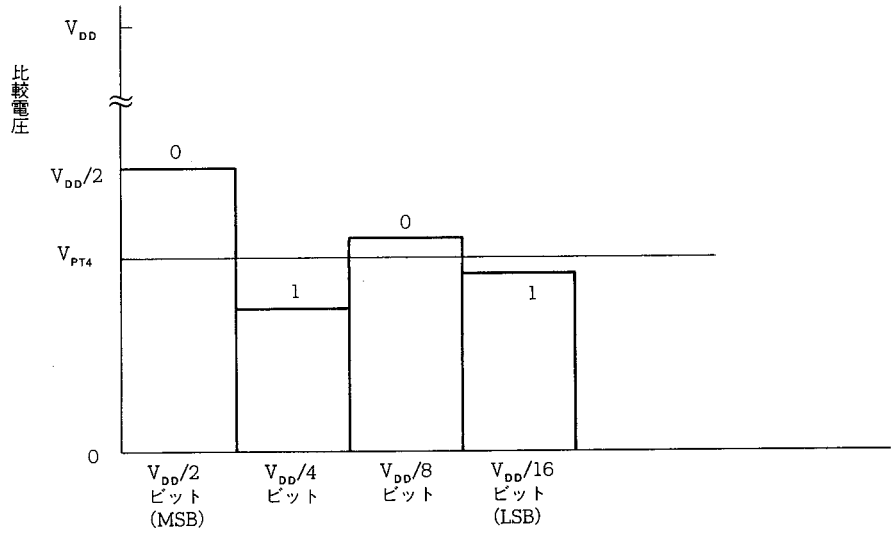
(iii) $V_{DA} = V_{DD} \times 3/8$ $(V_{DD}/4 + V_{DD}/8)$ とします。

$V_{DA} > V_{PT4}$ なので $V_{DD}/8$ ビットを 0 とします。

(iv) $V_{DA} = V_{DD} \times 5/16$ $(V_{DD}/4 + V_{DD}/16)$ とします。

$V_{DA} < V_{PT4}$ なので，LSB = 1 とします（$V_{DD}/16$ビット = 1）。

図９-１　ポートTによるA/D変換動作



(2) プログラム説明…(7) プログラム・リスト参照

(a)　初期比較電圧を $V_{DD}/2$ とします。

(b)　重み付けとなる値としてXレジスタに08Hを設定します。

(c)　変換待ちをします。変換時間は15.8 μs（f_{CLK} = 5 MHz）です。

(d)　Aレジスタに比較電圧を読み込みます。

(e)　重み付けのXレジスタを1/2します。

(f)　キャリーが発生したら（k）へ。

(g)　PT4端子の比較結果をチェックします。$V_{PT4} < V_{REF}$ならば（i）へ。

(h)　前回の比較電圧（Aレジスタ）にXレジスタを加算します。(j) へ。

(i)　前回の比較電圧（Aレジスタ）からXレジスタを減算します。

(j)　PMTレジスタ比較電圧としてAレジスタを設定します。(c) へ。

(k)　最下位ビットの比較結果を読み込み，これをAレジスタの最下位ビットとします。

(l)　Aレジスタの上位ビットを0とします。

(3) モード・レジスタ設定例

ポートTモード・レジスタ

	7	6	5	4	3	2	1	0	
PMT	PUPH	PUPL	0	0	MT3	MT2	MT1	MT0	（リセットで00H）
設定例	0	0	0	0	−	−	−	−	−：場合によって設定値が変化します。

MT3	MT2	MT1	MT0	基準電圧(V_{REF})の指定
0	0	0	0	V_{REF}の供給停止[注]
0	0	0	1	$1/16\times V_{DD}$
0	0	1	0	$2/16\times V_{DD}$
0	0	1	1	$3/16\times V_{DD}$
0	1	0	0	$4/16\times V_{DD}$
0	1	0	1	$5/16\times V_{DD}$
0	1	1	0	$6/16\times V_{DD}$
0	1	1	1	$7/16\times V_{DD}$
1	0	0	0	$8/16\times V_{DD}$
1	0	0	1	$9/16\times V_{DD}$
1	0	1	0	$10/16\times V_{DD}$
1	0	1	1	$11/16\times V_{DD}$
1	1	0	0	$12/16\times V_{DD}$
1	1	0	1	$13/16\times V_{DD}$
1	1	1	0	$14/16\times V_{DD}$
1	1	1	1	$15/16\times V_{DD}$

PUPL/PUPH	プルアップ抵抗の指定
0	プルアップ抵抗をOFFする
1	プルアップ抵抗をONする

注　スタンバイ・モードに設定する際，このモードに設定してください。

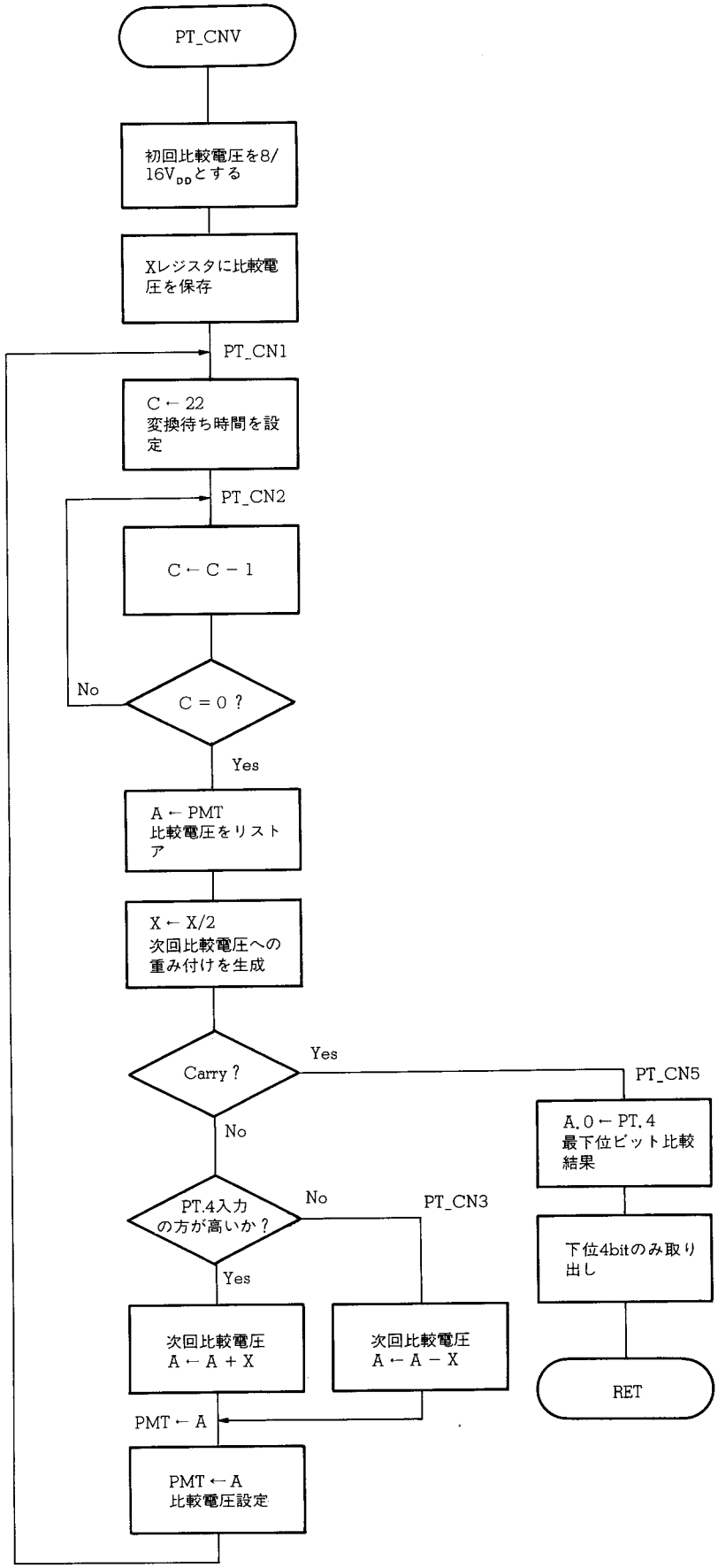
(4) 入出力パラメータ

Aレジスタに PT4 端子への入力電圧が 0-15 の値として生成されます。

(5) 使用レジスタ

X, A, C

(6) フロー・チャート


PT_CNV
初回比較電圧を8/16VDDとする
Xレジスタに比較電圧を保存
PT_CN1
C ← 22
変換待ち時間を設定
PT_CN2
C ← C − 1
No
C = 0 ?
Yes
A ← PMT
比較電圧をリストア
X ← X/2
次回比較電圧への重み付けを生成
Carry ?
Yes
PT_CN5
No
A.0 ← PT.4
最下位ビット比較結果
PT.4入力の方が高いか ?
No
PT_CN3
Yes
下位4bitのみ取り出し
次回比較電圧 A ← A + X
次回比較電圧 A ← A − X
RET
PMT ← A
PMT ← A
比較電圧設定

(7) プログラム・リスト

```
        NAME    PTCNVR
;
;*************************************************************
;* analogue-digital convert on                               *
;*              programable threshold port                   *
;*                      for only uPD78224,78220              *
;*      sampling port                                        *
;*              PT4                                          *
;*      result                                               *
;*              A                                            *
;*************************************************************
;
        PUBLIC  PT_CNV

PTWAIT  EQU     22              ; read result wait time
SCANPT  EQU     PT.4            ; scan port
;
        CSEG
PT_CNV:
        MOV     PMT,#08H        ; select 1/2 VDD
        MOV     X,#08H          ; compare data
PT_CN1:
        MOV     C,#PTWAIT       ; time delay
PT_CN2:
        DBNZ    C,$PT_CN2
        MOV     A,PMT
        SHR     X,1             ; add/sub parameter -> 1/2
        BC      $PT_CN5
        BF      SCANPT,$PT_CN3  ; result check

        ADD     A,X
        BR      PT_CN4
PT_CN3:
        SUB     A,X
PT_CN4:
        MOV     PMT,A           ; next compare
        BR      PT_CN1
PT_CN5:
        MOV1    CY,SCANPT
        MOV1    A.0,CY
        AND     A,#0FH
        RET

        END
```

第10章　EEPROMのプログラム例（μPD78244）

EEPROMは，μPD78244シリーズが備えているハードウエアで，データ・メモリ空間内の0FB00H-0FCFFHの512バイトの領域にマッピングされています。

EEPROMの用途の1つに，製品（カメラなど）の検査時に，1回だけその製品特有のパラメータ（同種の製品でも特性にばらつきが生じる）を書き込んでおくというものがあります。このパラメータは，工場出荷後には変更しない値となります。

もう1つの用途として，ユーザが製品を使用中に，随時変化し，電源切断後も保存しておきたいパラメータを書き込むというものがあります。

ここでは，この両者の場合のプログラム例を示します。

なお，この応用例に掲載しているプログラム例は，78K/IIシリーズ　構造化アセンブラ（ST78K2）で，ソース・プログラムが記述されています。

10.1　動作概要

(1) 概　　要

EEPROMの連続した領域に，内部RAMデータを連続的に書き込み，その領域にライト・プロテクトを設定します（以降，連続書き込みモードと表現します）。

その後，プログラムがリセットされると，書き込み要求があった場合に，別の1バイトの領域にイミーディエト・データを1回書き込みます（以降，1バイト書き込みモードと表現します）。

また，書き込みエラーが発生した場合は，エラー処理を行います。

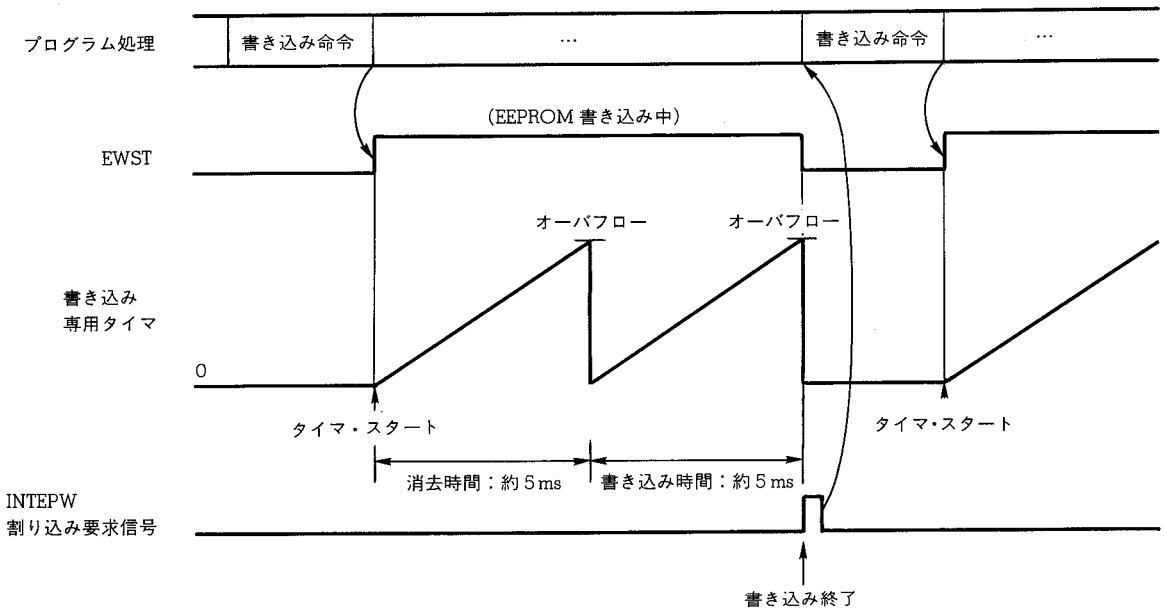
(2) プログラムでの書き込み方法

EEPROMへの書き込みには，約10 msの時間（書き込み処理時間）が必要です。この時間は，内蔵の書き込み専用タイマによって，ハードウエア的に確保されています。

EEPROMへ連続的に書き込みを行う場合は，1回書き込むごとに，書き込み処理時間が経過する（前回の書き込みが終了する）のを待つ必要があります。これを待たずに書き込もうとすると，オーバライト・エラーが発生します。

図10-1に，EEPROMへの書き込みタイミング・チャートを示します。

図 10 - 1　EEPROM への書き込みタイミング・チャート



書き込み終了待ち時間中は，他のプログラム処理が可能ですので，割り込み処理や，メインでのフラグ・チェックなどで書き込み処理を行うと，処理効率のよいプログラムとなります。

次に示すように，EEPROM 専用の機能を用いると便利です。

① 書き込み終了割り込み（INTEPW）処理内で行う。

② 書き込みステータス・フラグ（EWST = EWC.7）をチェックして，EWST = 0（書き込み中でない）の場合に行う。

このプログラム例では，連続書き込みモード時は①の方法で，1バイト書き込みモード時は②の方法で書き込みを行います。

10.2　プログラム説明

(1) 処理概要（「(8) SPDチャート」参照）

(a)　連続書き込みモード〔モジュール名称：WRC〕

(i) 連続書き込みモードのイニシャライズ処理〔レーベル名称：WRC_INI〕

① 書き込み制御レジスタ（EWC）のイニシャライズを行います。

動作周波数（f_{XX}）＝12 MHzとしますので，書き込み専用タイマのクロックは，65536/f_{XX}を選択します。

② 書き込み処理に必要なワーク・エリア（WRPNTSP，WRPNTDP，CNTAREA）のイニシャライズを行います。

③ 最初の1バイトを書き込むために，ダミーでINTEPW割り込み要求フラグ(EPWIF)をセット（1）し，マスクを解除して，1回目の割り込みが発生するようにします。

(ii) INTEPW割り込み処理〔レーベル名称：INTEPW〕

指定バイト数の書き込みが終了していない場合は，1バイト書き込んで，書き込みポインタを進めます。

(b)　1バイト書き込みモード〔モジュール名称：WR1〕

(i) 1バイト書き込みモードのイニシャライズ処理〔レーベル名称：WR1_INI〕

連続書き込み終了後，プログラムがリセットされると，この処理を行います。

連続書き込みモードで書き込んだ領域に対して，ライト・プロテクトを設定すると同時に，1バイト書き込みを許可します。

このプログラム例では，連続書き込み処理を行っていない場合でも，特定領域にライト・プロテクトが設定されます。

(ii) 1バイト書き込み処理〔レーベル名称：WR_1BYTE〕

書き込みステータス・フラグ(EWST)をチェックし，前回の書き込みが終了する(EWST＝0になる）まで待ちます。その後，1バイト書き込みを行います。

(c)　書き込みエラー処理〔モジュール名称：WRERR〕

(i) INTEER割り込み処理〔レーベル名称：INTEER〕

このプログラム例では，エラー処理に入ると，単に永久ループしていますが，実際に使用する場合は，ターゲット・システムに合わせ，適切な処理を行ってください。

(2) 使用する RAM

このプログラム例では，連続書き込みモード（モジュール名称：WRC）で，5バイトの RAM をワーク・エリアとして使用します。

また，この他に書き込みデータを格納するエリアが，書き込みバイト数分必要です。

表 10 - 1 に使用 RAM 一覧を示します。

表 10 - 1　EEPROM のプログラム例の使用 RAM 一覧

モジュール	RAM 名称	配置領域	用　途	バイト数	初期値
WRC	WRPNTSP	SADDR[注1]	書き込み元ポインタを格納	2	WRCDATA
	WRPNTDP		書き込み先ポインタを格納	2	WRCADRS
	CNTAREA		書き込みバイト数をカウント	1	WRCNT+1
	任意[注3]	RAM[注2]	書き込み元データを格納	任意	任意

注 1. SADDR：ショート・ダイレクト・アドレシングの適用範囲（0FE20H-0FEFFH）

2. RAM　：メモリ・バンク 0 内の任意 RAM の領域。

3. 任意とは，ユーザ側で設定することを意味します。

(3) 入出力パラメータ

(a) 入力パラメータ

表 10 - 2　EEPROM のプログラム例の入力パラメータ

モジュール	パラメータ	固定/可変	内　容
WRC	WRCDATA	固定値	書き込み元アドレス（内部 RAM）
	WRCADRS	固定値	書き込み先アドレス（EEPROM）
	WRCNT	固定値	書き込みバイト数
WR1	A レジスタ	可変値	書き込みデータを設定
	HL レジスタ	可変値	書き込みアドレスを設定
WRERR	なし	−	−

(b) 出力パラメータ

なし

(4) 使用レジスタ

表10-3　EEPROMのプログラム例の使用レジスタ

モジュール	ルーチン	レーベル	レジスタ	バンク
WRC	連続書き込みモードのイニシャライズ	WRC_INI	なし	−
	INTEPW割り込み処理	INTEPW	AX, DE, HL	1
WR1	1バイト書き込みモードのイニシャライズ	WR1_INI	なし	−
	1バイト書き込み処理	WR_1BYTE	A, HL	0
WRERR	書き込みエラー処理	INTEER	なし	−

(5) 使用スタック

このプログラム例のサブルーチン，および割り込み処理で使用するスタック・サイズは，3バイトです（ネスティング・レベル：1レベル）。

(6) モード・レジスタ設定例

(a) 連続書き込みモード

EEPROM 書き込み制御レジスタ

	7	6	5	4	3	2	1	0	
EWC	EWST	EOVW	WPA1	WPA0	EWE	EWP	EWTC1	EWTC0	（リセットで 34H）
設定例	0	0	1	1	1	0	0	0	

EEPROM 書き込み専用タイマ・クロック選択ビット

EWTC1	EWTC0	カウント・クロックの選択
0	0	65536/f_{xx}（f_{xx} = 12 MHz）
0	1	49152/f_{xx}（8 MHz≦f_{xx}<12 MHz）
1	0	32768/f_{xx}（6 MHz≦f_{xx}<8 MHz）
1	1	24576/f_{xx}（4 MHz≦f_{xx}<6 MHz）

ライト・プロテクト指定許可/禁止ビット

EWP	ライト・プロテクト（書き込み禁止）の指定
0	禁止(EEPROM 全領域のライト・プロテクトを解除)
1	許可(ライト・プロテクト領域に指定した部分をプロテクト)

EEPROM 書き込み許可/禁止制御ビット

EWE	EEPROM への書き込み動作
0	禁止する
1	許可する

EEPROM ライト・プロテクト領域指定ビット

WPA1	WPA0	ライト・プロテクト領域指定
0	0	エリア 3（FC80H-FCFFH）
0	1	エリア 3-2（FC00H-FCFFH）
1	0	エリア 3-1（FB80H-FCFFH）
1	1	エリア 3-0（FB00H-FCFFH）

EEPROM オーバライト・エラー検出フラグ

EOVW	オーバライト・エラー発生の有無
0	オーバライト・エラーなし
1	オーバライト・エラー発生

EEPROM 書き込みステータス・フラグ

EWST	書き込みステータス
0	EEPROM 書き込み中ではない
1	EEPROM 書き込み中

備考　f_{xx} はクリスタル/セラミック発振周波数を示します。外部クロック周波数を使用するときには，f_{xx} を f_x と読み替えてお使いください。

(b)　1バイト書き込みモード

EEPROM 書き込み制御レジスタ

	7	6	5	4	3	2	1	0	
EWC	EWST	EOVW	WPA1	WPA0	EWE	EWP	EWTC1	EWTC0	（リセットで 34H）
設定例	0	0	0	0	1	1	0	0	

EEPROM 書き込み専用タイマ・クロック選択ビット

EWTC1	EWTC0	カウント・クロックの選択
0	0	$65536/f_{xx}$（f_{xx}＝12 MHz）
0	1	$49152/f_{xx}$（8 MHz≦f_{xx}＜12 MHz）
1	0	$32768/f_{xx}$（6 MHz≦f_{xx}＜8 MHz）
1	1	$24576/f_{xx}$（4 MHz≦f_{xx}＜6 MHz）

ライト・プロテクト指定許可/禁止ビット

EWP	ライト・プロテクト（書き込み禁止）の指定
0	禁止（EEPROM 全領域のライト・プロテクトを解除）
1	許可（ライト・プロテクト領域に指定した部分をプロテクト）

EEPROM 書き込み許可/禁止制御ビット

EWE	EEPROM への書き込み動作
0	禁止する
1	許可する

EEPROM ライト・プロテクト領域指定ビット

WPA1	WPA0	ライト・プロテクト領域指定
0	0	エリア 3（FC80H-FCFFH）
0	1	エリア 3-2（FC00H-FCFFH）
1	0	エリア 3-1（FB80H-FCFFH）
1	1	エリア 3-0（FB00H-FCFFH）

EEPROM オーバライト・エラー検出フラグ

EOVW	オーバライト・エラー発生の有無
0	オーバライト・エラーなし
1	オーバライト・エラー発生

EEPROM 書き込みステータス・フラグ

EWST	書き込みステータス
0	EEPROM 書き込み中ではない
1	EEPROM 書き込み中

備考　f_{xx} はクリスタル/セラミック発振周波数を示します。外部クロック周波数を使用するときには，f_{xx} を f_x と読み替えてお使いください。

(7) プログラム使用例

連続書き込み，1バイト書き込みを，同一プログラムで行う例を示します。リセット直後のポートのレベルで，連続書き込みモード，1バイト書き込みモードを判断します。この使用例ではP26を使用しており，ポート・レベルがロウ（WRMODE = 0）の場合に連続書き込みモード，ポート・レベルがハイ（WRMODE = 1）の場合に1バイト書き込みモードとなります。

```
        PUBLIC  WRCDATA,WRCADRS,WRCNT   ; data
        EXTRN   WRC_INI,WR1_INI,WR_1BYTE; package

EERMK   EQU     MK1L.0                  ; INTEER mask flag
WRMODE  EQU     P2.6                    ; judge writing mode port

;       --- define area for writing 1byte ---

WR1DATA EQU     OFFH                    ; write data

EPRMS1  DSEG
WR1ADRS:DS      1                       ; write area (EEPROM)

;       --- define area for writing continuously ---

WRCNT   EQU     32                      ; write count

WRDS    DSEG
WRCDATA:DS      WRCNT                   ; source area of writing

EPRMS2  DSEG
WRCADRS:DS      WRCNT                   ; destination area of writing(EEPROM)
             :
             :
        CSEG
RST:
             :
             :
        CLR1    EERMK                   ; open INTEER mask
        if_bit(!WRMODE)                 ; IF writing continuously mode
                CALL    !WRC_INI        ; initialize of writing continuously
                EI                      ; enable interrupt
                while(forever)          ; LOOP forever
                endw
        endif                           ; IF writing 1 byte mode
        CALL    !WR1_INI                ; initialize of writing 1 byte
        EI                              ; enable interrupt
              :
              :
        HL=#WR1ADRS                     ; set write address
        A=#WR1DATA                      ; set write data
        CALL    !WR_1BYTE               ; writing 1 byte
              :
              :
```

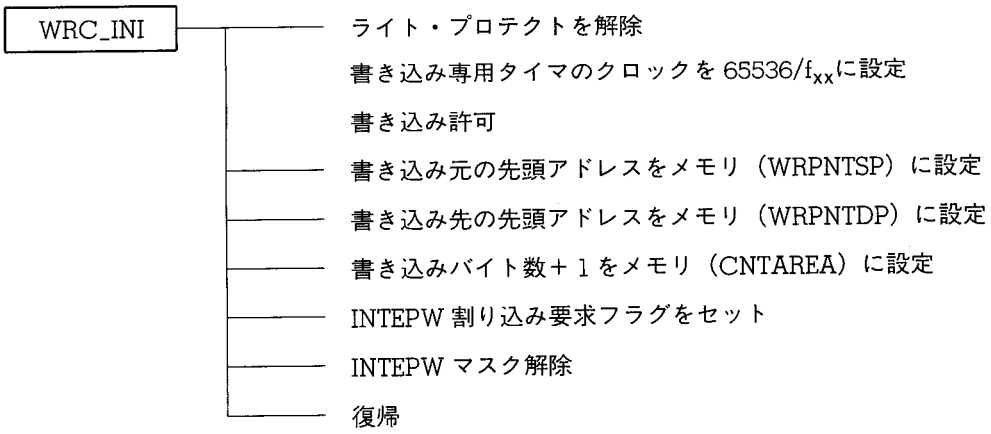
注意　EEPROM領域（この使用例では，データ・セグメントEPRMS1，EPRMS2）は，あらかじめ，リンク・ディレクティブ・ファイル内でアドレスを定義しておき，プログラムのリンク時に，このディレクティブ・ファイルをリンクするようにしてください。

次に，ディレクティブ・ファイル内容の例を示します。

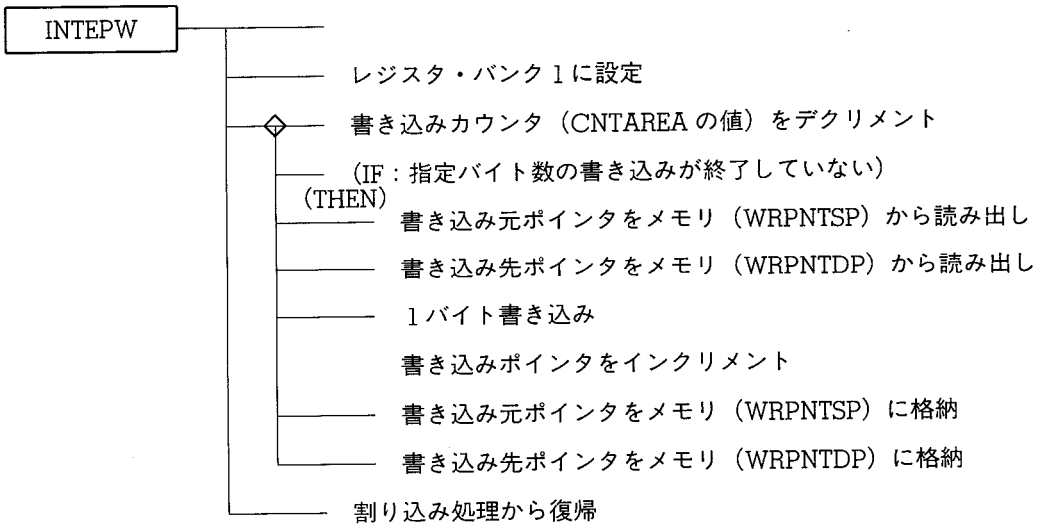
```
MEMORY  EEPROM0 : (0FB00H,128) / REGULAR
MEMORY  EEPROM1 : (0FB80H,128) / REGULAR
MEMORY  EEPROM2 : (0FC00H,128) / REGULAR
MEMORY  EEPROM3 : (0FC80H,128) / REGULAR
;
MERGE   EPRMS1  : = EEPROM0
MERGE   EPRMS2  : = EEPROM3
```

(8) SPD チャート

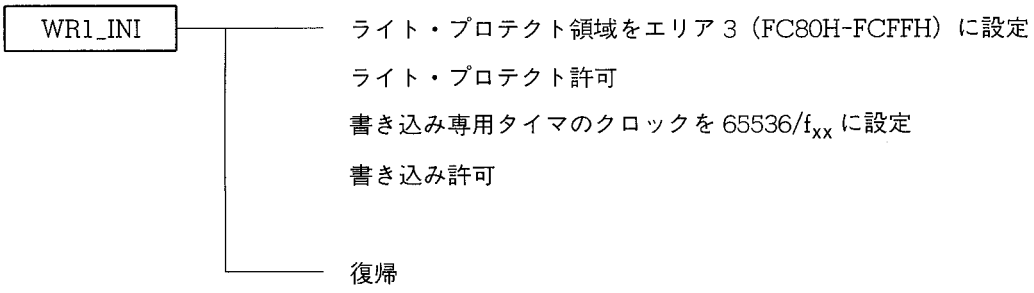
●連続書き込みイニシャライズ処理



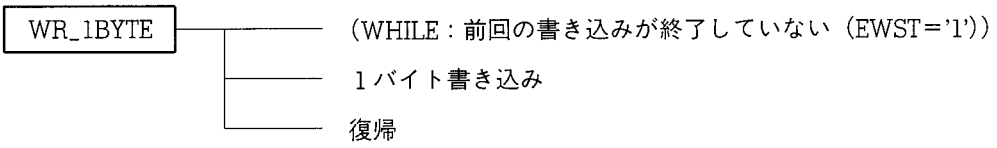
●連続書き込み処理（割り込み）



●1バイト書き込みイニシャライズ処理



●1バイト書き込み処理



●書き込みエラー処理

(9) プログラム・リスト

(a) 連続書き込みモード

このプログラム例では，ソース・プログラムの先頭で，ベクタ・テーブル・アドレスとマクロ・サービスの領域の定義用ファイル（INTMS.DEF）をインクルードしています。

```
$       IC(INTMS.DEF)
        NAME    WRC
;*************************************************************
;*      WRITING TO EEPROM                                    *
;*                      uPD78244                             *
;*              writing continuously mode                    *
;*************************************************************

        PUBLIC  WRC_INI         ; package
        EXTRN   WRCDATA,WRCADRS ; data
        EXTRN   WRCNT           ; data

EPWIF   EQU     IF1L.1          ; INTEPW request flag
EPWMK   EQU     MK1L.1          ; INTEPW mask flag

;       *** work area for writing to EEPROM ***

WRWOKD  DSEG    SADDR
WRPNTSP:DS      2               ; sorce pointer store area
WRPNTDP:DS      2               ; destination pointer store area
CNTAREA:DS      1               ; writing count store area

;       *** vector table ***

        EPWVENT                 ; INTEPW vector table

;*****************************************
;*      initialize of                    *
;*              writing continuously     *
;*****************************************

WRCS    CSEG
WRC_INI:
        EWC=#00111000B          ; open write protect,
                                ; timer clock=65536/fxx, enable writing
```

```
        WRPNTSP=#WRCDATA        ; set sorce address of first writing
        WRPNTDP=#WRCADRS        ; set destination address of first writing
        CNTAREA=#WRCNT+1        ; set writing count
        SET1    EPWIF           ; set INTEPW request flag
        CLR1    EPWMK           ; open INTEPW mask
        RET

;****************************************
;*      writing continuously routine    *
;*              by INTEPW               *
;****************************************

INTEPW:
        SEL     RB1             ; set register bank 1
        CNTAREA--               ;       decrement writing counter
        if(CNTAREA>#0)          ; not end writing of specified byte ?
                                ; THEN
                HL=WRPNTSP (AX) ;       read write pointer
                DE=WRPNTDP (AX)
                [DE+]=[HL+] (A) ;       write 1 byte & increment write pointer
                WRPNTSP=HL (AX) ;       store sorce pointer
                WRPNTDP=DE (AX) ;       store destination pointer
        endif
        RETI

        END
```

(b)　1バイト書き込みモード

このプログラム例では，SFRビット名定義ファイル（SFRBIT.DEF）をインクルードしています。

```
        NAME    WR1
;*************************************************************
;*      WRITING TO EEPROM                                    *
;*                      uPD78244                             *
;*              writing 1byte mode                           *
;*                                                           *
;*      input condition                                      *
;*              A  register <-- write data                   *
;*              HL register <-- write address                *
;*************************************************************

$       IC(SFRBIT.DEF)
        PUBLIC  WR1_INI,WR_1BYTE        ; package

;*****************************************
;*      initialize of                    *
;*              writing 1 byte           *
;*****************************************

WR1S    CSEG
WR1_INI:
        EWC=#00001100B          ; enable write protect about AREA 3
                                ; timer clock=65536/fxx
                                ; enable writing, protect area=FC80H-FCFFH
        RET

;*****************************************
;*      writing 1 byte routine           *
;*****************************************

WR_1BYTE:
        while_bit(EWST)         ; wait writing data
        endw
        [HL]=A                  ; write 1 byte
        RET

        END
```

(c)　書き込みエラー処理

このプログラム例では，ソース・プログラムの先頭で，ベクタ・テーブル・アドレスとマクロ・サービスの領域の定義用ファイル（INTMS.DEF）をインクルードしています。

```
$       IC(INTMS.DEF)
        NAME    WRERR
;*************************************************************
;*      WRITING TO EEPROM                                    *
;*                      uPD78244                             *
;*              ERROR of writing routine                     *
;*************************************************************

;       *** define vector table ***

        EERVENT                 ; INTEER vector table

WRERRS  CSEG
INTEER:
        while(forever)          ; LOOP forever
        endw

        END
```

付録A　78K/IIシリーズのプログラムの留意事項

A.1　ベクタ割り込み処理の留意事項

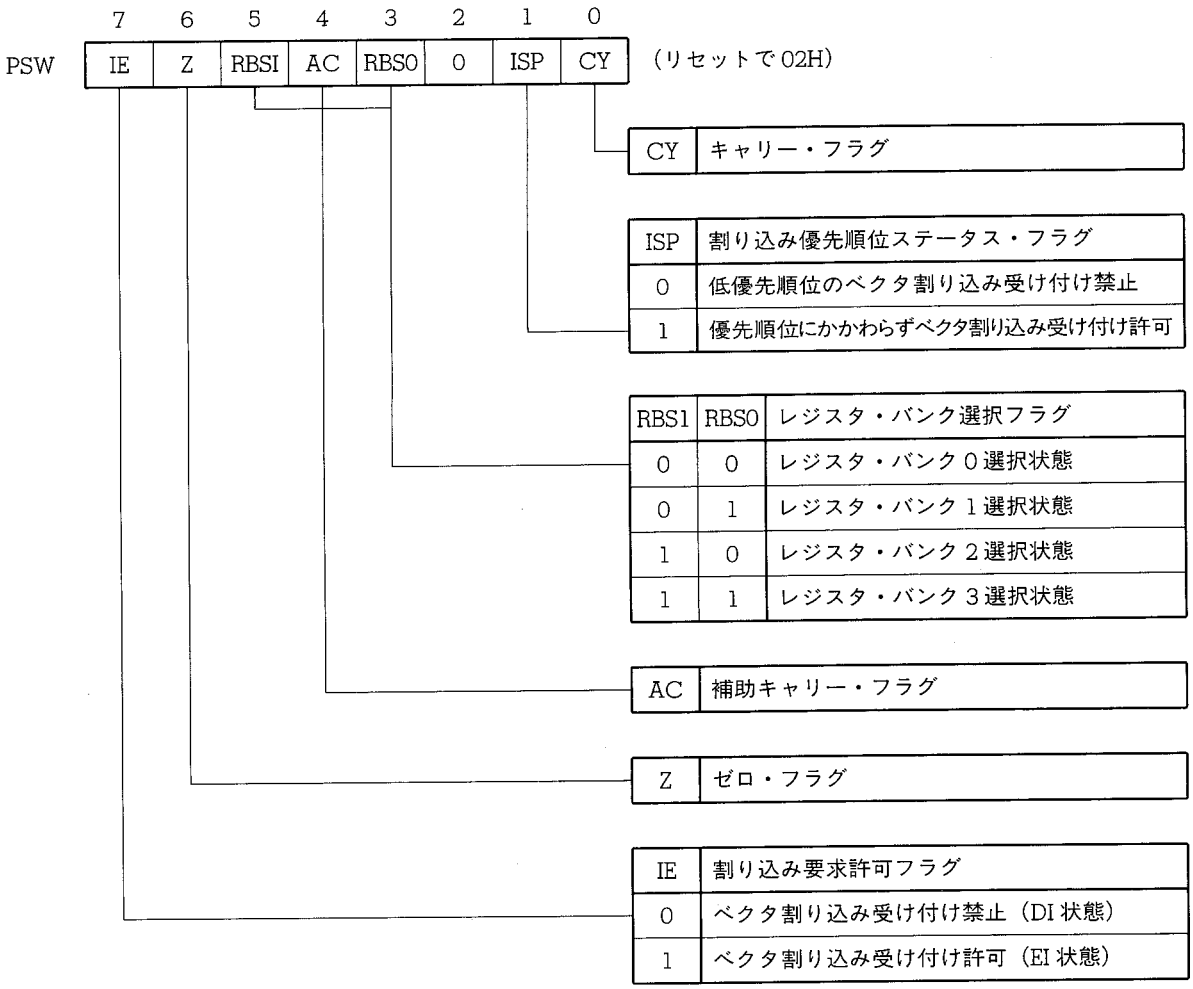
78K/IIシリーズでは図A-1に示したPSW（プログラム・ステータス・ワード）内に次の情報を含んでいます。

(a)　割り込み要求許可フラグ（IE）

(b)　割り込み優先順位ステータス・フラグ（ISP）

(c)　レジスタ・バンク選択フラグ（RBS0，RBS1）

図A-1　プログラム・ステータス・ワード



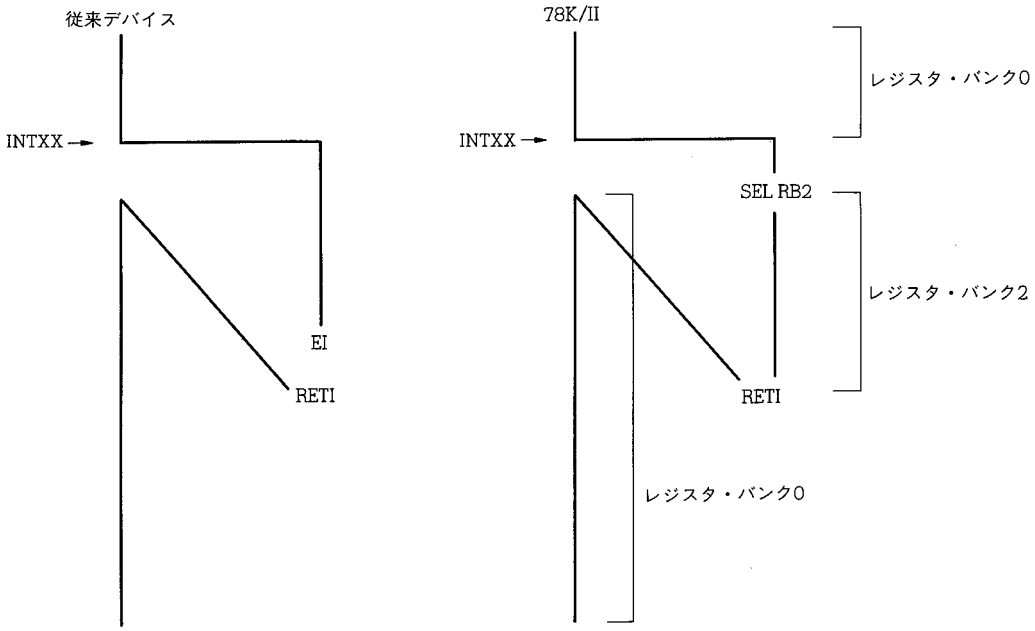
ベクタ割り込みの受け付けによりプログラム・カウンタ（PC），PSWがスタックに退避されます。このとき，割り込み許可フラグ（IE）は0となりベクタ割り込みの受け付けが禁止の状態となります。ベクタ割り込みサービス・プログラムが終了し，元のプログラムに復帰するためには“RETI”命令を実行します。

従来のデバイスの場合，次からのベクタ割り込みの受け付けを可能状態とするために，通常“EI”命令を“RETI”命令の前に実行します。

78K/IIシリーズの場合には“RETI”命令によってPC，PSWがスタックから復帰されますので，自動的に割り込みが許可状態となります。

また，ベクタ割り込み受け付け時にレジスタ・バンクを切り替えて対応した場合であっても，“RETI”命令によって元のレジスタ・バンクに戻ります。

図A-2　ベクタ割り込みとPSWの関係



次に割り込み優先順位ステータス・フラグ（ISP）は，高優先順位のベクタ割り込み（PROL/Hで指定したもの）が発生すると，PSWが退避されたのち，ISP←0となり，低優先順位のベクタ割り込みが受け付け禁止の状態となります。“RETI”命令で元のプログラムに復帰したときに優先順位にかかわらず，ベクタ割り込みの受け付けが可能となりますが，ISP←1とすることにより高優先順位のベクタ割り込み処理内でも優先順位にかかわらず（ただし“EI”命令を実行した場合：IE＝1）ベクタ割り込みの受け付けが可能です。

A.2　78K/IIシリーズの外部拡張データ・メモリのアクセス方法の留意事項

μPD7822× では，アクセスするメモリに相当するオペランドに“&”を付加することにより，あらかじめP6（ポート6）の出力ラッチに設定しておいた値がアドレス・バスの上位4ビットに出力されます。

μPD7821× では，上記の方法に加えて，すべてのメモリ空間に共通のアドレシング（ダイレクト・アドレシング，レジスタ・インダイレクト・アドレシング，インデクスト・アドレシング，ベース・アドレシング）によって，PM6（ポート6モード・レジスタ）の下位4ビットに設定された値が，常にアドレス・バスの上位4ビットとして出力されます。

注意　$\overline{\text{RESET}}$ 解除後，PM6 は OF × H に初期化され，下位4ビットが不定となります。したがって，MM レジスタのビット6（MM6）をセット（1）して，外部拡張データ・メモリのアクセスを許可する前に，PM6 レジスタの下位4ビットに“0”を設定するなどの初期化が必要です。

A.3　ポート0とリアルタイム出力ポートのアクセスの注意点

ポート0をリアルタイム出力ポートに設定した場合，ポート0への直接アクセスによって出力ラッチの内容を書き換えることができません。このときの出力ラッチの書き換えは，バッファ・レジスタからのハードウエアによる転送によってのみ行われます。

付録 B　RA78K/II

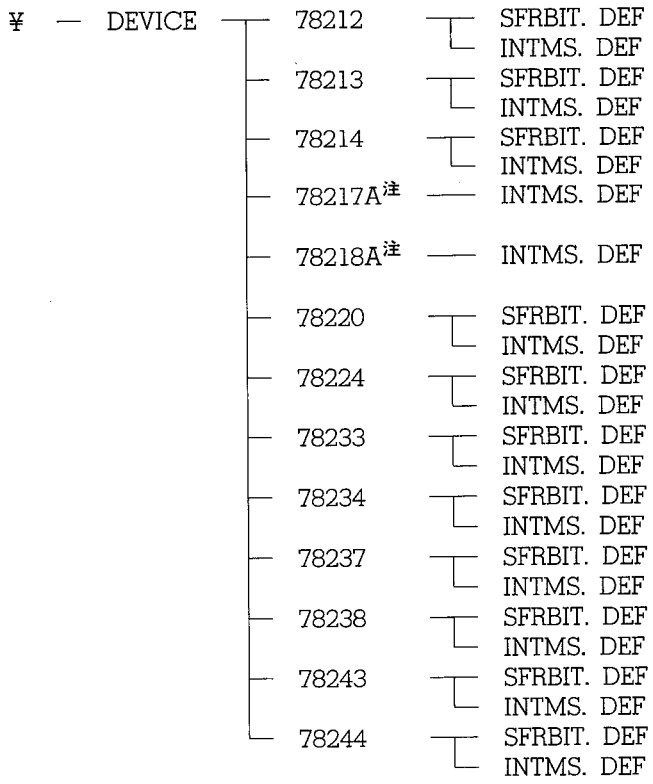
B.1　RA78K/II添付ファイルについて

このアプリケーション・ノートでも使用していますが，バージョン 3.00 以上のアセンブラには，アセンブラ・パッケージのプログラムのほかに，アセンブル時にインクルードして使用することのできるテキスト・ファイルが添付されています。

添付ファイルは，RA78K/IIアセンブラ・パッケージを使用する場合によく使用すると考えられる特殊機能レジスタ（SFR）のビット名を定義したファイル（SFRBIT. DEF）注と，割り込みのベクタ・テーブルやマクロ・サービスで使用する領域を確保するマクロを定義したファイル（INTMS. DEF）です。

これらのファイルは，¥DEVICE というディレクトリ中にある各デバイス用のディレクトリ内に用意してあります。

注　バージョン 4.00 以降では，アセンブラ本体に組み込まれており，インクルードする必要はありません。したがって，SFRBIT. DEF のファイルもありません。



注　バージョン 4.00 以降のみ

各ファイルの詳細な使用方法は，アセンブラ・パッケージに添付されている資料 **RA78K/IIアセンブラ・パッケージ添付ファイルの使用方法について（SUD-M-0319）** を参照してください。

付

B.1.1　SFRBIT.DEF

SFRBIT. DEF は，特殊機能レジスタのビットのうち，ビット単位での操作が主となると考えられるビットについて，そのビット名を定義したファイルです。ビット名は各デバイスのユーザーズ・マニュアルに記載してある名前と同一です。

なお，このファイルはバージョン 3.00 にのみ存在し，バージョン 4.00 からはアセンブラ本体の予約語として扱われます。バージョン 4.00 以上をご使用の場合は，SFRBIT. DEF のインクルードを行う必要はありません。また，バージョン 3.00 用に作成したソース・ファイルを使用する場合は，SFRBIT. DEF をインクルードしている制御命令を削除してください。

B.1.2　INTMS. DEF

INTMS. DEF は，割り込みおよびマクロ・サービスで使用するメモリ領域の確保などを行うマクロを定義したファイルです。マクロを参照することで領域の確保などを行います。これらを使用することにより，絶対アドレスを気にすることなしに割り込みやマクロ・サービスが使用できます。

マクロは，定義する内容により３種類に分類されます。

(1) ベクタ・テーブル定義用マクロ

ベクタ・テーブルの定義を行います。各割り込みごとにマクロが定義されています。

(2) マクロ・サービス・コントロール・ワード用領域確保マクロ

マクロ・サービス・コントロール・ワード用の領域確保とレーベルの定義を行います。各割り込み（マクロ・サービスが使用できるものに限る）ごとにマクロが定義されています。

(3) マクロ・サービス・チャネル用領域確保マクロ

マクロ・サービス・チャネル用の領域確保とレーベルの定義を行います。各マクロ・サービスのタイプごとにマクロが定義されています。

B.2　レジスタ・バンクの領域確保の方法

78K/IIシリーズでは汎用レジスタを内蔵RAM上にマッピングしています。アセンブリ言語では一般的にデータ・メモリの領域を確保するのにDS疑似命令を使用しますが，RA78K/IIでは，このDS疑似命令による領域確保時にレジスタ・バンクと共通になっている領域もデータ・メモリの領域として確保してしまう場合があります。

ここでは，使用しているレジスタ・バンクの領域だけをアセンブラ・パッケージの機能を使用してデータ・メモリ領域と重ならないようにする方法を示します。

B.2.1 準　備

リスト1-5のファイルを作成します。このうち，SELRB. DEFは，ソース・ファイルでインクルードして使用します。また，SELRB0. ASM，SELRB1. ASM，SELRB2. ASM，SELRB3. ASMは，アセンブルを行い，ライブラリアンにより1つのライブラリ・ファイルとしてリンク時に使用します。

●リスト1　SELRB. DEF

```
$       NOLIST
        EXTRN   DSRB0
SELDRB1 SET     0FFFFH
SELDRB2 SET     0FFFFH
SELDRB3 SET     0FFFFH
;
SELRB0  MACRO
$       NOGEN
        SEL     RB0
$       GEN
        ENDM
;
SELRB1  MACRO
$       NOGEN
$       _IF     SELDRB1 EQ 0FFFFH
        EXTRN   DSRB1
SELDRB1 SET     0
$       ENDIF
        SEL     RB1
$       GEN
        ENDM
;
SELRB2  MACRO
$       NOGEN
$       _IF     SELDRB2
        EXTRN   DSRB2
SELDRB2 SET     0
$       ENDIF
        SEL     RB2
$       GEN
        ENDM
;
SELRB3  MACRO
$       NOGEN
$       _IF     SELDRB3
        EXTRN   DSRB3
SELDRB3 SET     0
$       ENDIF
        SEL     RB3
$       GEN
        ENDM
$       LIST
```

●リスト2 SELRB0.ASM

```
        NAME    DSRB
        PUBLIC  DSRB0
;
SEGRB0  DSEG    AT 0FEF8H
DSRB0:  DS      8
;
        END
```

●リスト3 SELRB1.ASM

```
        NAME    DSRB
        PUBLIC  DSRB1
;
SEGRB1  DSEG    AT 0FEF0H
DSRB1:  DS      8
;
        END
```

●リスト4 SELRB2.ASM

```
        NAME    DSRB
        PUBLIC  DSRB2
;
SEGRB2  DSEG    AT 0FEE8H
DSRB2:  DS      8
;
        END
```

●リスト5 SELRB3.ASM

```
        NAME    DSRB
        PUBLIC  DSRB3
;
SEGRB3  DSEG    AT 0FEF0H
DSRB3:  DS      8
;
        END
```

MS-DOS上で，B.1に紹介したINTMS. DEFと同じディレクトリ内にSELRB. DEFをコピーし，ライブラリ・ファイル（SELRB.LIB）を作成するバッチ・ファイルなどを紹介します。

●バッチ・ファイル本体　MKLIBALL. BAT

```
FOR %%D IN(212 213 214 217A 218A 220 224 233 234 237 238 243 244)DO COMMAND/C MKLIB %%D %1
```

●バッチ・ファイルで呼び出すバッチ・ファイル　MKLIB. BAT

```
FOR %%F IN (0 1 2 3) DO RA78K2 DSRB%%F.ASM -C%1 -NP
LB78K2 < MKLIB.CMD
COPY DSRB.LIB %2¥DEVICE¥78%1
COPY DSRB.DEF %2¥DEVICE¥78%1
DEL DSRB?.REL
DEL DSRB.LIB
```

●ライブラリアンに入力するコマンド・ファイル　MKLIB.CMD

```
C DSRB.LIB
A DSRB.LIB DSRB0.REL
A DSRB.LIB DSRB1.REL
A DSRB.LIB DSRB2.REL
A DSRB.LIB DSRB3.REL
E
```

実際の作業はコマンド・ライン上で，次のように入力することで開始します。

MKLIBALL　アセンブラ・パッケージのあるディレクトリ

B.2.2　使用方法

(1) ソース・ファイルでの記述方法

ソース・プログラムのモジュール・ボディの先頭で SELRB. DEF のインクルードを指定します。

また, レジスタ・バンクを選択する命令である, SEL RBn 命令を記述する代わりにマクロ名 SELRBn を記述します。

例

```
$       TITLE ('TEST')
$       IC (INTMS. DEF)
$       IC (SELRB. DEF)
;
        NAME TEST
;
$       IC (SFRBIT. DEF)
              ⋮
SELRB2    ; レジスタ・バンク 2 を選択する
```

上記の例の場合, アセンブル時にコマンド・ラインでインクルード・ファイル読み込みパス指定をするか, MS-DOS の環境変数でインクルード・ファイル読み込みパスの指定が必要です。

付

例 1. コマンド・ライン上での指定

```
RA78K2 TEST  -IA : ¥DEVICE¥78214 -C214
```

2. 環境変数の設定方法

```
SET INC78K2 = A : ¥DEVICE¥78214
```

(2) リンク時の指定

リンク時に, ライブラリ・ファイル (SELRB. LIB) を使用するように指定します。ライブラリ・ファイルの指定は, コマンド・ラインまたはパラメータ・ファイルで指定します。

例　LK78K2 TEST -BA : ¥DEVICE¥78214¥SELRB. LIB

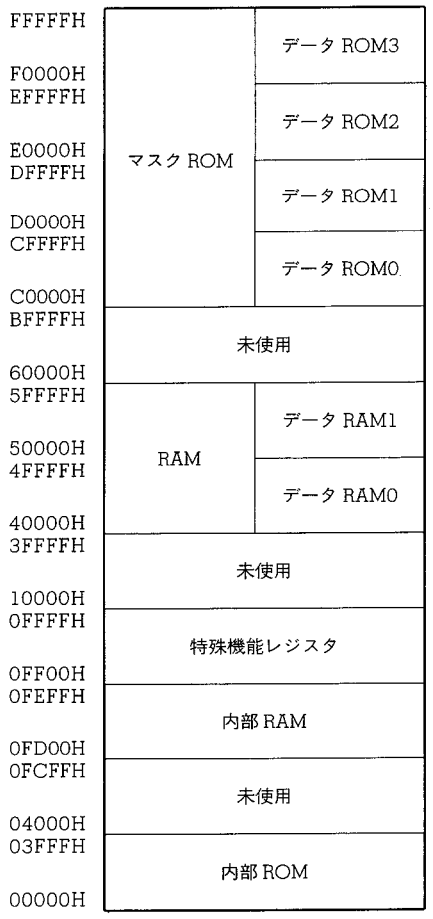
B.3　1Mバイト拡張データ・メモリ空間の使用方法

78K/IIシリーズでは，データ・メモリ空間を1Mバイトに拡張することができます。この拡張したデータ・メモリ空間を使用する場合のRA78K/II上での記述方法などの例を示します。

(1) メモリ空間の使用例

図B-1に示すようにメモリ空間を使用します。78K/IIシリーズではメモリ空間は64Kバイトを1バンクとして使用するため，大容量のメモリは64Kバイト単位の複数のバンクとして使用します。

図B-1　メモリ空間の使用例



(2) ソース・プログラムでの記述方法

ソース・プログラム上では，拡張データ・メモリ空間に配置するデータの領域確保などは，必ず名前付きのセグメント内で行います。

ここでは，表B-1のようにセグメント名を決めています。

表B-1 セグメント名と配置場所

セグメント名	配置場所
LINEBUFF	データ RAM0
SUBBUFF	
MAINBUFF	データ RAM1
SYSPARA	データ ROM0
KDATA	データ ROM1
RDATA	
FONT1	データ ROM2
FONT2	データ ROM3

具体的な記述例は，次のようになります。

```
LINEBUFF    DSEG

LL0 :       DS   1024
LM0 :       DS   1024
            ⋮

SUBBUF      DSEG
SB :        DS   512
            ⋮

SYSPARA     CSEG
PARA1 :     DB   50H, 43H, 7FH
            ⋮
```

付

参照する場合には，あらかじめPM6またはP6レジスタでメモリ・バンクを指定して行います。

```
例  MOVW    HL, #LM0
    MOV     P6, #4
    MOV     &[HL], A
            ⋮

    MOV     P6, #12     ; 0CH
    MOV     A, &!PARA1
```

(3) リンク時の指定方法

リンカは，ディレクティブ・ファイルによって配置アドレスを決定します。

ディレクティブ・ファイルでは，メモリ領域の定義とセグメントのメモリ領域への配置を指定します。

メモリ領域の定義は，各メモリ・バンク内[注]の一部または全部に名前を付けます。1つのメモリ・バンク内には，いくつの領域があってもかまいません。ここで定義されたメモリ領域にセグメントを配置することになります。

注　RA78K シリーズのマニュアルでは，他の機能との関連で「メモリ空間」と表現しています。

ここでは表B-2のようにメモリ領域を定義します。

表B-2　メモリ領域の定義

メモリ領域名	配置場所
ERAM0	データ RAM0
ERAM1	データ RAM1
EROM0	データ ROM0
EROM1_0	データ ROM1 の 0000H-7FFFH[注]
EROM1_1	データ ROM1 の 8000H-FFFFH[注]
EROM2	データ ROM2
EROM3	データ ROM3

注　バンク内のアドレス，絶対アドレスでは，それぞれ D0000H-D7FFFH，D8000H-DFFFFH

セグメントの配置は，表B-3のように行うこととします。

表B-3　セグメントの配置

セグメント名	配置するメモリ領域
LINEBUFF	ERAM0
SUBBUFF	
MAINBUFF	ERAM1
SYSPARA	EROM0
KDATA	EROM1_0
RDATA	EROM1_1
FONT1	EROM2
FONT2	EROM3

この場合のディレクティブ・ファイルは次のようになります。

```
MEMORY    ERAM0       : (0, 0FFFFH)/EX4
MEMORY    ERAM1       : (0, 0FFFFH)/EX5
MEMORY    EROM0       : (0, 0FFFFH)/EX12
MEMORY    EROM1_0     : (0, 80000H)/EX13
MEMORY    EROM1_1     : (8000H, 8000H)/EX13
MEMORY    EROM2       : (0, 0FFFFH)/EX14
MEMORY    EROM3       : (0, 0FFFFH)/EX15
;
MERGE     LINEBUFF    : =ERAM0/EX4
MERGE     SUBBUFF     : =ERAM0/EX4
MERGE     MAINBUFF    : =ERAM1/EX5
MERGE     SYSPARA     : =EROM0/EX12
MERGE     KDATA       : =EROM1_0/EX13
MERGE     PDATA       : =EROM1_1/EX13
MERGE     FONT1       : =EROM2/EX14
MERGE     FONT2       : =EROM3/EX14
```

リンク実行時にコマンド・ラインまたはパラメータ・ファイルで －D オプションを使用して，ディレクトリ・ファイルを指定してください。

付録C　78K/IIシリーズ製品一覧

次ページ以降に，78K/IIシリーズの製品を一覧表にして表します。

表C-1　78K/IIシリーズ製品一覧

シリーズ名		μPD78214シリーズ			μPD78218Aシリーズ		μPD78224シリーズ	
項目＼品名		μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
基本命令数		65（78K/IIシリーズ共通）						
最小命令実行時間		333 ns	500 ns	333 ns	500 ns	333 ns	500 ns	333 ns
PUSH PSW命令の実行時間（クロック数）		スタック・エリアが内部デュアル・ポートRAMの場合：5または7 上記以外：7または9			スタック・エリアが内部デュアル・ポートRAMの場合：6 上記以外：8		スタック・エリアが内部デュアル・ポートRAMの場合：5または7 上記以外：7または9	
動作温度範囲と電源電圧範囲		μPD78P218A以外：−40～+85 °C，V_{DD} = +5 V±10 % μPD78P218A　：−40～+85 °C，V_{DD} = +5 V±0.3 V					−40～+85 °C，V_{DD} = +5 V±5 % −10～+70 °C，V_{DD} = +5 V±10 %	
汎用レジスタ		8ビット×8×4バンク						
バンク・レジスタ		P6とPM6					P6のみ	
内蔵メモリ	ROM	8Kバイト	なし	16Kバイト	なし	32Kバイト	なし	16Kバイト
	EEPROM	なし						
	RAM	384バイト	512バイト		1024バイト		640バイト	
メモリ空間		プログラム・メモリ空間：64Kバイト，データ・メモリ空間：1Mバイト						
I/O端子	入　力	14					8	
	出　力	12					12	20
	入出力	28	10	28	10	28	25	35
	合　計	54	36	54	36	54	45	63
注付加機能付き端子	プルアップ抵抗付き端子	34	16	34	16	34	なし	
	LEDダイレクト・ドライブ出力	16	0	16	0	16	8	
	トランジスタ・ダイレクト・ドライブ出力	8					なし	
	P0	8ビット出力ポート						
	P1	−					8ビット入出力ポート	
	P2	8ビット入力ポート						
	P3	8ビット入出力ポート						
	P4	8ビット入出力ポート	−	8ビット入出力ポート	−	8ビット入出力ポート	−	8ビット入出力ポート
	P5	8ビット入出力ポート	−	8ビット入出力ポート	−	8ビット入出力ポート	−	8ビット出力ポート
	P6	4ビット出力ポート＋4ビット入出力ポート	4ビット出力ポート＋2ビット入出力ポート	4ビット出力ポート＋4ビット入出力ポート	4ビット出力ポート＋2ビット入出力ポート	4ビット出力ポート＋4ビット入出力ポート	4ビット出力ポート＋2ビット入出力ポート	4ビット出力ポート＋4ビット入出力ポート
	P7	6ビット入力ポート					7ビット入出力ポート	

注　付加機能付き端子は，I/O端子の中に含まれています。

(1/3)

μPD78234シリーズ				μPD78244シリーズ	
μPD78233	μPD78234	μPD78237	μPD78238 (μPD78P238)	μPD78243	μPD78244
65（78K/IIシリーズ共通）					
500 ns	333 ns	500 ns	333 ns	500 ns	333 ns
スタック・エリアが内部デュアル・ポートRAMの場合：6 上記以外：8					
−40～+85 °C, V_{DD} = +5 V±10 %				−10～+70 °C, V_{DD} = +5 V±10 %	
8ビット×8×4バンク					
P6とPM6					
なし	16Kバイト	なし	32Kバイト (32K/16Kバイト注)	なし	16Kバイト
なし				512バイト	
640バイト		1024バイト	1024バイト (1024/640バイト注)	512バイト	
プログラム・メモリ空間：64Kバイト，データ・メモリ空間：1Mバイト					
16				14	
12					
18	36	18	36	10	28
46	64	46	64	36	54
24	42	24	42	16	34
8	24	8	24	0	16
8					
8ビット出力ポート					
8ビット入出力ポート				−	
8ビット入力ポート					
8ビット入出力ポート					
−	8ビット 入出力ポート	−	8ビット 入出力ポート	−	8ビット 入出力ポート
−	8ビット 入出力ポート	−	8ビット 入出力ポート	−	8ビット 入出力ポート
4ビット 出力ポート + 2ビット 入出力ポート	4ビット 出力ポート + 4ビット 入出力ポート	4ビット 出力ポート + 2ビット 入出力ポート	4ビット 出力ポート + 4ビット 入出力ポート	4ビット 出力ポート + 2ビット 入出力ポート	4ビット 出力ポート + 4ビット 入出力ポート
8ビット入力ポート				6ビット入力ポート	

注　ソフトウエアにより設定

表C-1　78K/IIシリーズ製品一覧

シリーズ名		μPD78214シリーズ			μPD78218Aシリーズ		μPD78224シリーズ	
項目＼品名		μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
PWM出力		なし						
コンパレータ		なし					4ビット×8	
A/Dコンバータ		8ビット×8					なし	
	変換時間の選択	動作周波数に応じて選択					−	
	AV_{REF}入力電圧範囲	3.4 V～V_{DD}			3.6 V～V_{DD}		−	
	入力電圧に関する制限	ADMレジスタのANI0-ANI2ビットで選択されている端子のみ，常時0VからAV_{REF}端子電圧まで			A/D変換の対象となっている端子について，A/D変換中のみ0VからAV_{REF}端子電圧まで		−	
D/Aコンバータ		なし						
タイマ／カウンタ	16ビット・タイマ/カウンタ	1						
	8ビット・タイマ/カウンタ	3					2	
	タイマ出力	4						
	PWM/PPG出力	あり					なし	
	ワンショット・パルス	なし			あり		なし	
	割り込みソース	7					5	
外部SFR領域		0FFD0H-0FFDFHの16バイト					なし	
シリアル・インタフェース	UART	1チャネル						
	CSI	1チャネル（SBI対応）						
	BRG用タイマ	あり（タイマ/カウンタ3と兼用）					あり	
	専用ボー・レート・ジェネレータ	あり					なし	
	外部ボー・レート・クロック入力	あり					なし	
リアルタイム出力ポート		4ビット×2または8ビット×1						

(2/3)

μPD78234シリーズ				μPD78244シリーズ	
μPD78233	μPD78234	μPD78237	μPD78238 (μPD78P238)	μPD78243	μPD78244
12ビット×2				なし	
なし					
8ビット×8					
任意に選択可能				動作周波数に応じて選択	
3.4 V～V_{DD}				3.6 V～V_{DD}	
A/D変換の対象となっている端子について，A/D変換中のみ， 0 VからAV$_{REF}$端子電圧まで					
8ビット×2				なし	
1					
3					
4					
あり					
あり					
7					
0FFD0H-0FFDFHの16バイト					
1チャネル					
1チャネル（SBI対応）					
あり（タイマ/カウンタ3と兼用）					
あり					
あり					
4ビット×2または8ビット×1					

表C-1　78K/IIシリーズ製品一覧

シリーズ名	μPD78214シリーズ			μPD78218Aシリーズ		μPD78224シリーズ	
項目＼品名	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
割り込み	2レベル（プログラマブル），ベクタ/マクロ・サービス						
外　部	7					8	
内　部	12					9	
マクロ・サービス使用可能割り込み数	15					6	
マクロ・サービス・カウンタのビット数	8ビットのみ			8ビット/16ビットの選択が可能 (ただし，タイプAを除く)		8ビットのみ	
マクロ・サービス・タイプCのMPD, MPTのインクリメント	下位8ビットのみインクリメント (上位は変化せず)			16ビットでインクリメント		下位8ビットのみインクリメント（上位は変化せず）	
マクロ・サービスの実行時間	μPD78214シリーズとμPD78224シリーズは同一です。 μPD78218Aシリーズ，μPD78234シリーズおよびμPD78244シリーズは同一です。 実行時間はモードによって異なります。各製品のユーザーズ・マニュアルで比較してください。						
マクロ・サービス・タイプAのメモリからSFRへのデータ転送時の制限事項	転送データがD0H-DFHのときに発生			転送バッファ（メモリ）のアドレスがOFED0H-OFEDFHのときに発生		転送データがD0H-DFHのときに発生	
スタンバイ機能	HALT/STOPモード						
STOPモード解除時の発振安定時間	固　定			2種類の時間から選択が可能		固　定	
疑似SRAMリフレッシュ機能	あり（リフレッシュ・パルス幅は，$1/f_{CLK}$）					あり（リフレッシュ・パルス幅は，$1.5/f_{CLK}$）	
メモリ・アクセスに関する制限	なし					リフレッシュ機能使用時は，FC80H-FDFFHのアクセス不可	
ROMレス・モードの設定	$\overline{EA}$端子＝ロウ・レベル	−	$\overline{EA}$端子＝ロウ・レベル	−	$\overline{EA}$端子＝ロウ・レベル	−	$\overline{EA}$端子＝ロウ・レベル
パッケージ	●64ピン・プラスチック・シュリンクDIP（750 mil） ●68ピン・プラスチックQFJ：μPD78212を除く ●64ピン・プラスチックQFP（本体14×14 mm） ●74ピン・プラスチックQFP（本体20×20 mm） ●64ピン・プラスチックQUIP：μPD78212を除く ●64ピン・セラミック窓付きシュリンクDIP：μPD78P214のみ			●64ピン・プラスチック・シュリンクDIP（750 mil） ●64ピン・プラスチックQFP：（本体14×14 mm） ●64ピン・セラミック窓付きシュリンクDIP：μPD78P218Aのみ		●84ピン・プラスチックQFJ ●94ピン・プラスチックQFP（本体20×20 mm）	

(3/3)

μPD78234シリーズ				μPD78244シリーズ	
μPD78233	μPD78234	μPD78237	μPD78238 (μPD78P238)	μPD78243	μPD78244
2レベル（プログラマブル），ベクタ/マクロ・サービス					
7					
12				14	
15					
8ビット/16ビットの選択が可能（ただし，タイプAを除く）					
16ビットでインクリメント					
μPD78214シリーズとμPD78224シリーズは同一です。 μPD78218Aシリーズ，μPD78234シリーズおよびμPD78244シリーズは同一です。 実行時間はモードによって異なります。各製品のユーザーズ・マニュアルで比較してください。					
転送データがD0H-DFHのときに発生				転送元バッファ（メモリ）のアドレスが0FED0H-0FEDFHのときに発生	
HALT/STOPモード					
2種類の時間から選択が可能					
あり（リフレッシュ・パルス幅は，$1/f_{CLK}$）					
なし					
−	MODE端子= ハイ・レベル	−	MODE端子= ハイ・レベル （設定不可）	−	$\overline{EA}$端子= ロウ・レベル
●84ピン・プラスチックQFJ ●80ピン・プラスチックQFP（本体14×14 mm） ●94ピン・プラスチックQFP（本体20×20 mm） ●94ピン・セラミックWQFN：μPD78P238のみ				●64ピン・プラスチック・シュリンクDIP（750 mil） ●64ピン・プラスチックQFP（本体14×14 mm）	

付録D　プログラム，デバイス，内蔵周辺ハードウエアの対応表

項　　目	対象デバイス					内蔵周辺ハードウエア														
						タイマ/カウンタ				アシンクロナス・シリアル・インタフェース	3線式シリアル・インタフェース	SBI	リアルタイム出力ポート	A/D コンバータ	ポートT	PWM	EEPROM	割り込み			
																		マクロ・サービス			ベクタ
	μPD78214 シリーズ	μPD78218A シリーズ	μPD78224 シリーズ	μPD78234 シリーズ	μPD78244 シリーズ	16ビット・タイマ/カウンタ	8ビット・タイマ/カウンタ1	8ビット・タイマ/カウンタ2	8ビット・タイマ/カウンタ3									タイプA	タイプB	タイプC	割り込み
第3章　タイマ/カウンタのプログラム例 3.1　内部インターバル・タイマ (1)　16ビット・タイマ/カウンタを用いたプログラム例	○	○	○	○	○	○															○
(2)　8ビット・タイマ/カウンタ2を用いたプログラム例	○	○	○	○	○		△	○	△												○
3.2　プログラマブル矩形波出力	○	○	○	○	○	○															○
3.3　フリー・ランニング・インターバル・タイマ (1)　16ビット・タイマ/カウンタを用いたプログラム例	○	○	○	○	○	○															○
(2)　8ビット・タイマ/カウンタ2を用いたプログラム例	○	○	○	○	○		△	○													○
3.4　PWM/PPG 出力(μPD78214) (1)　16ビット・タイマ/カウンタを用いたPWM出力プログラム例	○	○		○	○	○															○
(2)　8ビット・タイマ/カウンタ2を用いたPWM出力プログラム例	○	○		○	○																○
(3)　16ビット・タイマ/カウンタを用いたPPG出力プログラム例	○	○		○	○																○
(4)　8ビット・タイマ/カウンタ2を用いたPPG出力プログラム例	○	○		○	○			○													○
3.5　ソフト・トリガド・ワンショット・パルス出力		○		○	○																
3.6　パルス周期測定	○	○	○	○	○	○	△	△													○
第4章　PWM 出力ユニットのプログラム例(μPD78234)				○												○					
第5章　アシンクロナス・シリアル・インタフェースのプログラム例 (a)　μPD78214 のプログラム	○	○		○	○					○											
(b)　μPD78224 のプログラム	○	○	○	○	○					○											
第6章　3線式シリアル・インタフェースのプログラム例	○	○	○	○	○						○										
第7章　割り込み処理のプログラム例 7.1　UART 受信処理	○	○	○	○	○					○								○	△		○
7.2　外部割り込み要求に同期したパラレル・データ入力	○	○	○	○	○														○		○
7.3　ステッピング・モータの開ループ制御　その1	○	○	○	○	○		○													○	
7.4　ステッピング・モータの開ループ制御　その2		○		○	○		○						○					△	○	○	
第8章　A/D コンバータのプログラム例	○	○		○	○								○	○							
第9章　コンパレータのプログラム例			○												○						
第10章　EEPROM のプログラム例(μPD78244)					○												○				

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］ **78K/IIシリーズ　アプリケーション・ノート　基礎編**

（ IEA-607E（第６版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
そ　の　他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは

NEC 販売員，特約店販売員，NEC 半応技本部員，その他（　　）

ご協力ありがとうございました。

下記あてに FAX で送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC 半導体応用技術本部インフォメーションセンター
FAX：(044)548-7900

キリトリ

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01　東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3454-1111　(大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460　名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540　大阪市中央区城見一丁目4番24号（NEC関西ビル）	大　阪 大　阪 大　阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社	札　幌 (011)231-0161	太田支店	太　田 (0276)46-4011	富山支店	富　山 (0764)31-8461
東北支社	仙　台 (022)267-8740	宇都宮支店	宇都宮 (028)621-2281	三重支店	津　 (0592)25-7341
岩手支店	盛　岡 (019)651-4344	小山支店	小　山 (0285)24-5011	京都支社	京　都 (075)344-7824
山形支店	山　形 (0236)23-5511	長野支社	松　本 (0263)35-1662	神戸支社	神　戸 (078)333-3854
郡山支店	郡　山 (0249)23-5511	甲府支店	甲　府 (0552)24-4141	中国支社	広　島 (082)242-5504
いわき支店	いわき (0246)21-5511	埼玉支社	大　宮 (048)641-1411	鳥取支店	鳥　取 (0857)27-5311
長岡支店	長　岡 (0258)36-2155	立川支社	立　川 (0425)26-5981	岡山支店	岡　山 (086)225-4455
土浦支店	土　浦 (0298)23-6161	千葉支社	千　葉 (043)238-8116	四国支社	高　松 (0878)36-1200
水戸支店	水　戸 (029)226-1717	静岡支社	静　岡 (054)255-2211	新居浜支店	新居浜 (0897)32-5001
神奈川支社	横　浜 (045)324-5524	北陸支社	金　沢 (0762)23-1621	松山支店	松　山 (089)945-4149
群馬支店	高　崎 (0273)26-1255	福井支店	福　井 (0776)22-1866	九州支社	福　岡 (092)271-7700

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 マイクロコンピュータ技術部	〒210　川崎市幸区塚越三丁目484番地	川　崎	(044)548-7924
半導体販売技術本部 東日本販売技術部	〒108-01　東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3798-9619
半導体販売技術本部 中部販売技術部	〒460　名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋	(052)222-2125
半導体販売技術本部 西日本販売技術部	〒540　大阪市中央区城見一丁目4番24号（NEC関西ビル）	大　阪	(06) 945-3383

半導体
インフォメーションセンター
FAX(044)548-7900
(FAXにてお願い致します)