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。

RENESAS

ユーザーズ・マニュアル

μPD17145サブシリーズ

4ビット・シングルチップ・マイクロコントローラ

μPD17145
μPD17147
μPD17149
μPD17P149

資料番号　U10261JJ2V0UM00（第2版）
（旧資料番号　IEU-867）
発行年月　July 1995 P

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

本製品が外国為替および外国貿易管理法の規定による戦略物資等（または役務）に該当するか否かは，ユーザ（仕様を決定した者）が判定してください。

○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	μPD17P149　開発中→開発済み
p. 114	図13－1　8ビット・タイマ・カウンタの構成を修正
p. 120	13.1.6　インターバル時間の設定を追加
p. 121	13.1.7　インターバル時間の誤差を追加
p. 143	図13－21　単発モード（コンペア動作）のタイミングを変更
p. 170	表15－1　スタンバイ・モード中の状態の注意2の文を修正
p. 177	15.3.3　STOPの設定条件（1）$\overline{\text{RLS}}$入力による解除の場合を変更
p. 183	17.3　POC回路使用時の注意事項を追加
p. 199	20.3　命令セット一覧に注1を追加
p. 277	付録B　μPD17145サブシリーズとμPD17135A，17137Aの機能比較を修正
p. 287	付録F　改版履歴を追加

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

ご利用対象者　このマニュアルは，μPD17145サブシリーズの機能を理解し，それを用いたアプリケーション・システムを設計するユーザを対象としています。

目　的　このマニュアルはμPD17145サブシリーズの機能を記述したもので，プログラムを作る際の参考資料としていただくことを目的としています。

読み方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータの一般知識を必要とします。

●一通りμPD17145サブシリーズの機能を理解しようとするとき
→目次に従って読んでください。

●ニモニックが分かっているときの命令機能を調べるとき
→付録E　命令索引をご利用ください。

●ニモニックは知らないが大体の機能が分かっている命令を調べたいとき
→20.3　命令セット一覧でその命令のニモニックを調べ，20.5　命令の個別説明でその機能を調べてください。

●μPD17145サブシリーズの電気的特性を知りたいとき
→別冊のデータ・シートを参照してください。

凡　例

データの表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称に上線）
メモリ・マップのアドレス	：上部ー下位，下部ー上位
注	：本文中につけた注の説明
注　意	：気をつけて読んでいただきたい内容
備　考	：本文の補足説明
数の表記	：2進数…××××または××××B
	10進数…××××または××××D
	16進数…××××H

この資料では，特に断りがないかぎりμPD17149を代表品種として説明しています。

関連資料　次の資料とあわせてご利用ください。

表の中の番号は資料番号です。

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

資料＼製品	μPD17145	μPD17147	μPD17149	μPD17P149
データ・シート	IC - 8793〔IC-3283〕			IC-8978〔IC-3505〕
ユーザーズ・マニュアル	このマニュアル			
IE - 17K（Ver.1.6） ユーザーズ・マニュアル	EEU - 929〔EEU-1467〕			
IE - 17K - ET（Ver.1.6） ユーザーズ・マニュアル	EEU - 931〔EEU-1466〕			
SIMPLEHOST™ ユーザーズ・マニュアル	EEU - 723〔EEU-1336〕（入門編） EEU - 724〔EEU-1337〕（レファレンス編）			
AS17K（Ver1.11） ユーザーズ・マニュアル	EEU - 603〔EEU-1287〕			
デバイス・ファイル ユーザーズ・マニュアル	EEU - 949〔EEU-1486〕			
SEボード ユーザーズ・マニュアル（暫）	EEU - 945〔EEU-1475〕			

備考　〔　〕内は英文の資料番号です。

目　　次

図の目次（1/3）

図の目次（2/3）

図の目次（3/3）

表の目次（1/2）

表の目次（2/2）

〔メ　モ〕

第1章　概　　説

μPD17149は，8ビットA/Dコンバータ（4チャネル），タイマ機能（3チャネル），シリアル・インタフェースを内蔵した4ビット・シングルチップ・マイクロコントローラです。マスク・オプションにより，POC回路を内蔵することができます。

μPD17P149は，ワン・タイムPROM製品であり，システム開発時のプログラム評価や少量生産に適しています。

次に特徴を示します。

●17Kアーキテクチャ　汎用レジスタ方式，命令長16ビット固定

●命令実行時間　2μs（fx＝8MHz：セラミック発振）

●プログラム・メモリ（ROM）　μPD17145　：2Kバイト（1024×16ビット）
　　μPD17147　：4Kバイト（2048×16ビット）
　　μPD17149　：8Kバイト（4096×16ビット）
　　μPD17P149：8Kバイト（4096×16ビット，ワン・タイムPROM）

●データ・メモリ（RAM）　110×4ビット

●A/Dコンバータ　4チャネル（8ビット分解能，逐次比較型）

●タイマ　3チャネル（8ビット・タイマ・カウンタ×2チャネル，ベーシック・インターバル・タイマ注）

●シリアル・インタフェース　1チャネル（クロック同期3線式）

●POC回路（マスク・オプション）

●電源電圧　V_{DD} = 4.5 ～ 5.5 V（fx = 400 kHz ～ 8 MHz動作時）
　　V_{DD} = 2.7 ～ 5.5 V（fx = 400 kHz ～ 2 MHz動作時）

注　ベーシック・インターバル・タイマを利用して，内部リセット信号を発生可能です（ウォッチドッグ・タイマ機能）。

μPD17149は，アナログ電圧測定を伴う制御用，サブマイコン用に適した特徴を持っており，次のような応用分野に適用できます。

●家庭電化製品

●バッテリ・チャージャ

●カメラ

●電子計測機器

1.1 機能一覧

項目＼品名	μPD17145	μPD17147	μPD17149	μPD17P149
ROM容量	マスクROM			ワン・タイムPROM
	2Kバイト（1024X16ビット）	4Kバイト（2048X16ビット）	8Kバイト（4096X16ビット）	
RAM容量	110×4ビット			
スタック	アドレス・スタック×5，割り込みスタック×3			
入出力ポート数	23本 { ・入出力 ：20本 ・入力専用 ：2本 ・センス入力（INT端子[注]） ：1本			
A/Dコンバータ入力	4チャネル（ポート端子兼用），絶対確度 ±1.5LSB以下			
タイマ	3チャネル { ・8ビット・タイマ・カウンタ：2チャネル(16ビット・タイマ1チャネル応用可) ・7ビット・ベーシック・インターバル・タイマ：1チャネル(ウォッチドッグ・タイマ応用可)			
シリアル・インタフェース	1チャネル（3線式）			
割り込み	・ハードウエアによる多重割り込み可（最大3レベル） ・外部割り込み：1本（INT） { 立ち上がり検出 立ち下がり検出 立ち上がりと立ち下がりの両エッジ検出 } 選択可 ・内部割り込み：4本 { ・タイマ0（TM0） ・タイマ1（TM1） ・ベーシック・インターバル・タイマ（BTM） ・シリアル・インタフェース（SIO）			
命令実行時間	2μs（fx＝8MHz動作時：セラミック発振）			
スタンバイ機能	HALT，STOP			
★ POC回路	マスク・オプション （V_{DD}＝5±10％，fx＝400 kHz～4 MHzの応用回路で使用できます。）			なし
電源電圧	V_{DD}＝2.7～5.5 V（fx＝400 kHz～2 MHz動作時） V_{DD}＝4.5～5.5 V（fx＝400 kHz～8 MHz動作時）			
パッケージ	28ピン・プラスチック・シュリンクDIP（400 mil） 28ピン・プラスチックSOP（375 mil）			

★ 注　INT端子は外部割り込み機能を使用しない場合に，入力専用端子（センス入力）として使用できます。センス入力では端子の状態をポート・レジスタではなく，コントロール・レジスタのINTフラグで読みます。

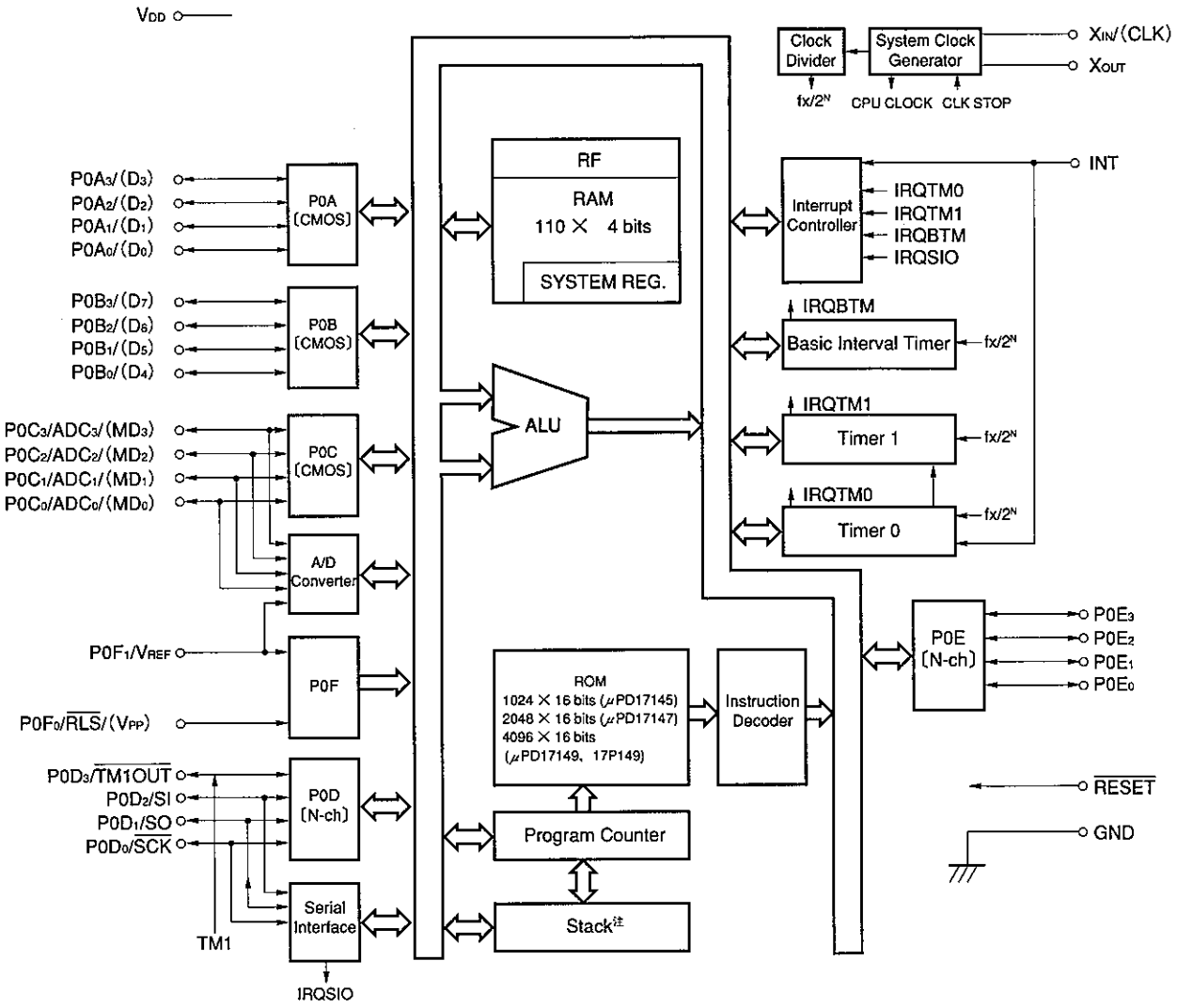
★ **注意　PROM製品は，マスクROM製品と機能的には高い互換性がありますが，内部ROM回路や電気的特性の一部などに違いがあります。PROM製品からマスクROM製品に切り替える際には，マスクROM製品のサンプルによる応用評価を十分に行ってください。**

1.2　オーダ情報

オーダ名称	パッケージ	内蔵ROM
μPD17145CT-×××	28ピン・プラスチック・シュリンクDIP（400 mil）	マスクROM
μPD17145GT-×××	28ピン・プラスチックSOP（375 mil）	〃
μPD17147CT-×××	28ピン・プラスチック・シュリンクDIP（400 mil）	〃
μPD17147GT-×××	28ピン・プラスチックSOP（375 mil）	〃
μPD17149CT-×××	28ピン・プラスチック・シュリンクDIP（400 mil）	〃
μPD17149GT-×××	28ピン・プラスチックSOP（375 mil）	〃
μPD17P149CT	28ピン・プラスチック・シュリンクDIP（400 mil）	ワン・タイムPROM
μPD17P149GT	28ピン・プラスチックSOP（375 mil）	〃

備考　×××はROMコード番号です。

1.3　ブロック図



注　スタックの容量は製品によって異なります。

備考1．（ ）内はμPD17P149のプログラム・メモリ書き込み／ベリファイ・モード時のみ有効です。

2．〔 〕内のCMOS，N-chは，ポートの出力形式を表します。

CMOS：CMOSプッシュプル出力

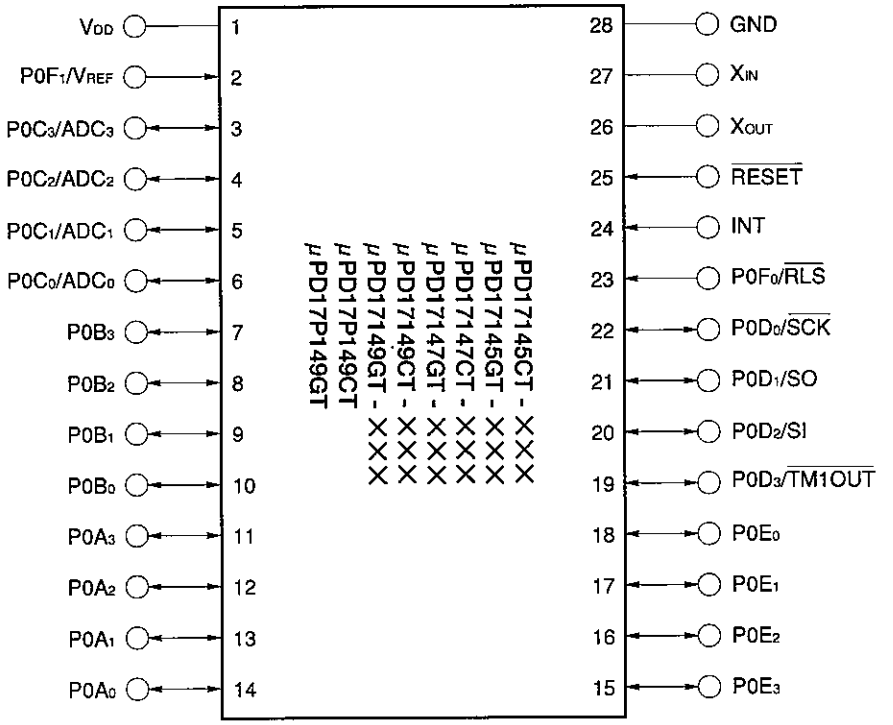
N-ch：N-chオープン・ドレーン出力

1.4　端子接続図（Top View）

（1）通常動作モード

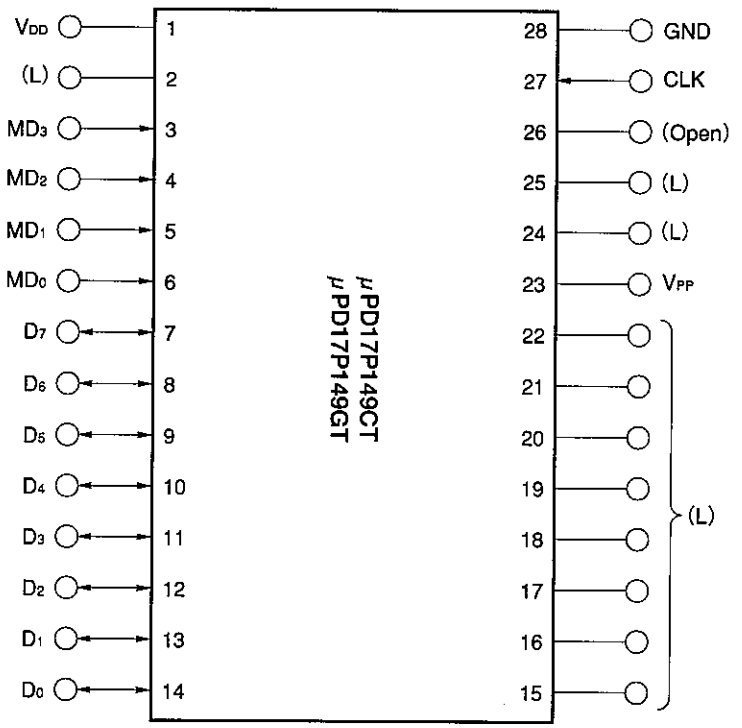
28ピン・プラスチック・シュリンクDIP（400 mil）

28ピン・プラスチックSOP（375 mil）



ADC_0 - ADC_3	：アナログ入力	$\overline{RESET}$	：リセット入力
GND	：グランド	$\overline{RLS}$	：スタンバイ解除信号入力
INT	：外部割り込み入力	$\overline{SCK}$	：シリアル・クロック入出力
$P0A_0$ - $P0A_3$	：ポート0A	SI	：シリアル・データ入力
$P0B_0$ - $P0B_3$	：ポート0B	SO	：シリアル・データ出力
$P0C_0$ - $P0C_3$	：ポート0C	$\overline{TM1OUT}$	：タイマ1出力
$P0D_0$ - $P0D_3$	：ポート0D	V_{DD}	：電源
$P0E_0$ - $P0E_3$	：ポート0E	V_{REF}	：A/Dコンバータ基準電圧
$P0F_0$, $P0F_1$	：ポート0F	X_{IN}, X_{OUT}	：システム・クロック

（2）プログラム・メモリ書き込み／ベリファイ・モード



CLK　：アドレス更新用クロック入力　　MD0 - MD3：動作モード選択入力

D0 - D7：データ入出力　　VDD　　：電源

GND　：グランド　　VPP　　：プログラム電圧印加

注意　（　）内はプログラム・メモリ書き込み／ベリファイ・モードでは使用しない端子の処理です。

L　　：個別にプルダウン抵抗を介してGND に接続してください。

Open ：何も接続しないでください。

第2章　端子機能

2.1　端子機能説明

2.1.1　通常動作モード時の端子

端子番号	記　号	機　能	出力形式	リセット時
1	V_{DD}	電源です。	—	—
2	$P0F_1/V_{REF}$	ポート0FおよびA/Dコンバータの基準電圧入力です。 ・マスク・オプションによるプルアップ抵抗を内蔵可能[注] ・$P0F_1$ ・2ビット入力ポート(P0F)のビット1 ・V_{REF} ・A/Dコンバータの基準電圧入力端子	入力	入力 ($P0F_1$)
3 | 6	$P0C_3/ADC_3$ | $P0C_0/ADC_0$	ポート0CおよびA/Dコンバータのアナログ入力です。 ・$P0C_3$ - $P0C_0$ ・4ビット入出力ポート ・1ビット単位で入力／出力設定可能 ・ADC_3 - ADC_0 ・A/Dコンバータのアナログ入力	CMOSプッシュプル	入力 (P0C)
7 8 9 10	$P0B_3$ $P0B_2$ $P0B_1$ $P0B_0$	ポート0Bです。 ・4ビット入出力ポート ・4ビット単位で入力／出力設定可能 ・4ビット単位でソフトウエアによるプルアップ抵抗を内蔵可能	CMOSプッシュプル	入力
11 12 13 14	$P0A_3$ $P0A_2$ $P0A_1$ $P0A_0$	ポート0Aです。 ・4ビット入出力ポート ・4ビット単位で入力／出力設定可能 ・4ビット単位でソフトウエアによるプルアップ抵抗を内蔵可能	CMOSプッシュプル	入力
15 16 17 18	$P0E_3$ $P0E_2$ $P0E_1$ $P0E_0$	ポート0Eです。 ・4ビット入出力ポート ・4ビット単位で入力／出力設定可能 ・4ビット単位でソフトウエアによるプルアップ抵抗を内蔵可能	N-ch オープン・ドレーン	入力

注　μPD17P149には，マスク・オプションによるプルアップ抵抗は内蔵されていません。 ★

端子番号	記　号	機　能	出力形式	リセット時
19	$P0D_3/\overline{TM1OUT}$	ポート0D，タイマ1出力，シリアル・データ入力，シリアル・データ出力およびシリアル・クロック入出力です。 ・1ビット単位でソフトウエアによるプルアップ抵抗を内蔵可能 ・$P0D_3$-$P0D_0$ ・4ビット入出力ポート ・1ビット単位で入力／出力設定可能 ・$\overline{TM1OUT}$ ・タイマ1出力	N-ch オープン・ドレーン	入力
20	$P0D_2$/SI	・SI ・シリアル・データ入力		
21	$P0D_1$/SO	・SO ・シリアル・データ出力		
22	$P0D_0/\overline{SCK}$	・$\overline{SCK}$ ・シリアル・クロック入出力		
23	$P0F_0/\overline{RLS}$	ポート0Fおよびスタンバイ解除信号の入力です。 ・マスク・オプションによるプルアップ抵抗を内蔵可能[注] ・$P0F_0$ ・2ビット入力ポート（P0F）のビット0 ・$\overline{RLS}$ ・スタンバイ解除信号の入力	入力	入力
24	INT	外部割り込み要求信号の入力です。スタンバイ解除信号の入力にも使用できます。 ・マスク・オプションによるプルアップ抵抗を内蔵可能[注]	入力	入力
25	$\overline{RESET}$	システム・リセット入力です。 ・マスク・オプションによるプルアップ抵抗を内蔵可能[注]	入力	入力
26 27	X_{OUT} X_{IN}	システム・クロック発振用です。 X_{IN}, X_{OUT}間にセラミック発振子を接続します。	—	—
28	GND	GNDです。	—	—

★ 注　μPD17P149には，マスク・オプションによるプルアップ抵抗は内蔵されていません。

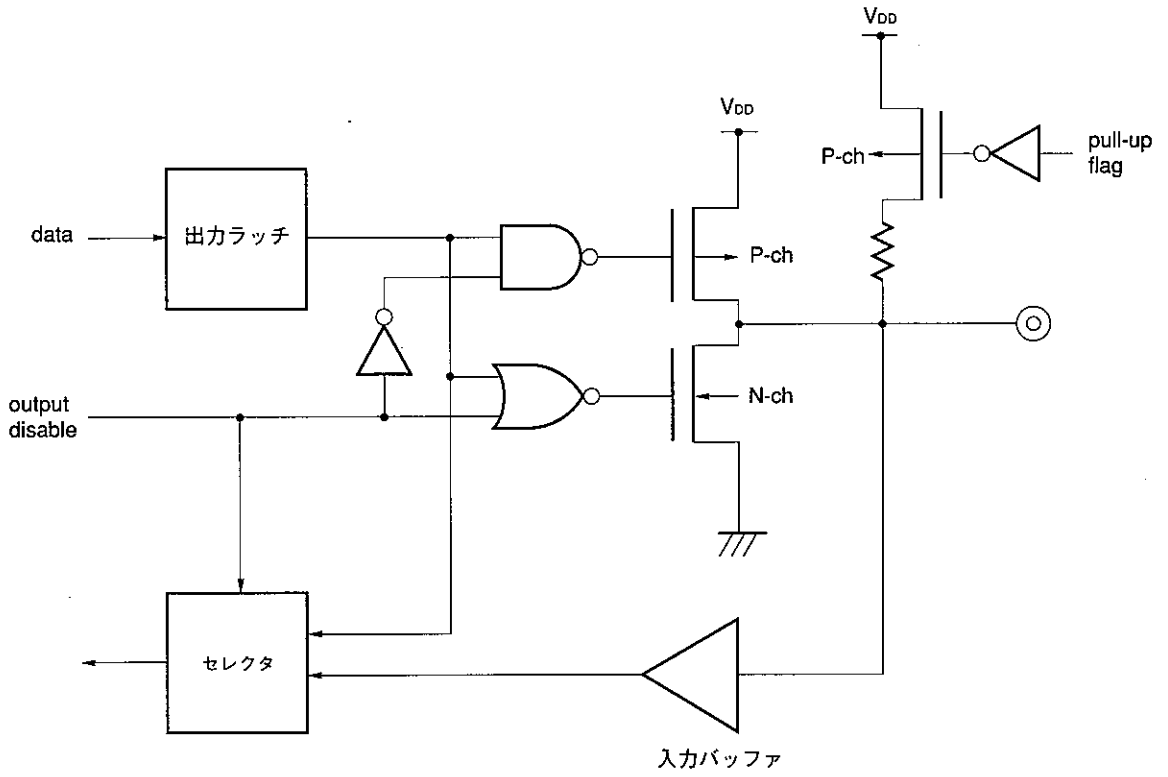
2.1.2　プログラム・メモリ書き込み／ベリファイ・モード時の端子 …μPD17P149のみ

端子番号	記　号	機　能	入出力
1	V_{DD}	正電源です。 プログラム・メモリ書き込み／ベリファイ時には＋6Vを印加します。	－
3 ｜ 6	MD_3 ｜ MD_0	プログラム・メモリ書き込み／ベリファイ時に動作モードを選択するための入力です。	入　力
7 ｜ 14	D_7 ｜ D_0	プログラム・メモリ書き込み／ベリファイ時の8ビット・データ入出力です。	入出力
23	V_{PP}	プログラム・メモリ書き込み／ベリファイ時のプログラム電圧印加端子です。 ＋12.5Vを印加します。	－
27	CLK	プログラム・メモリ書き込み／ベリファイ時のアドレス更新用クロック入力です。	入　力
28	GND	GNDです。	－

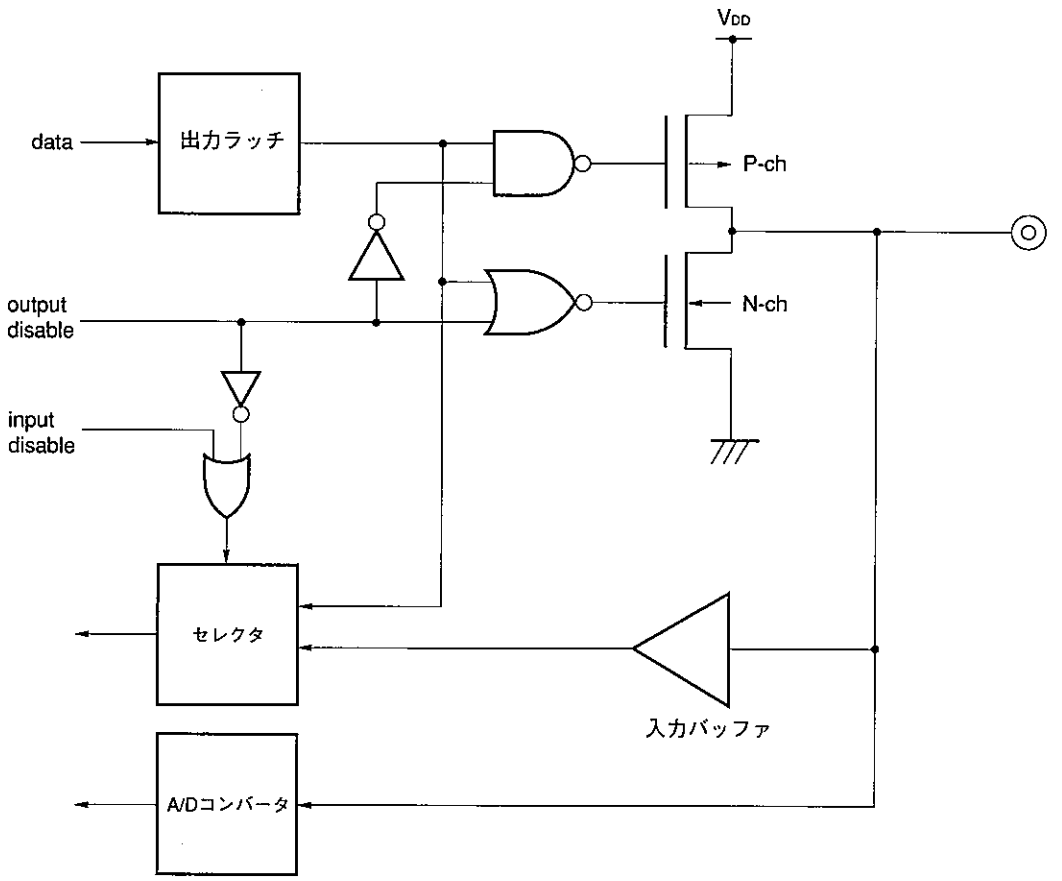
２．２　端子の等価回路

各端子の入出力回路を一部簡略化した形式を用いて示します。

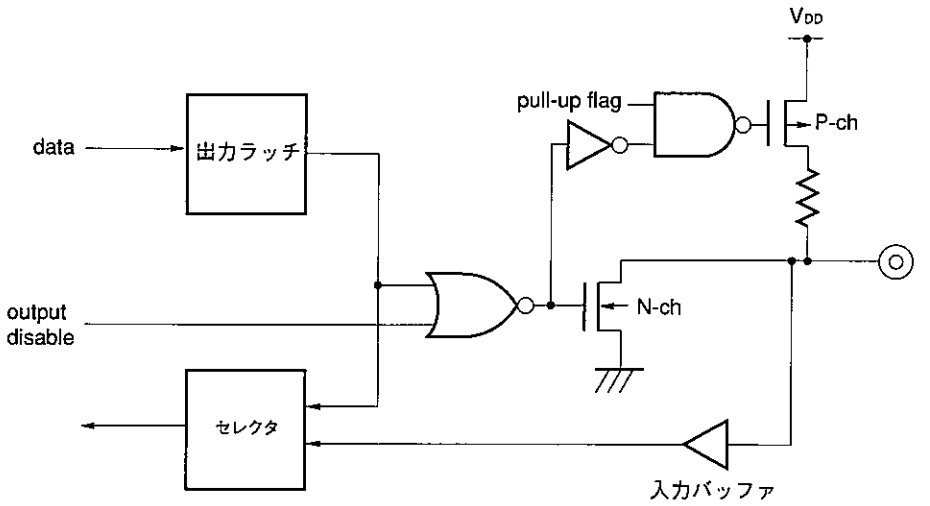
（１）$P0A_0$ - $P0A_3$, $P0B_0$ - $P0B_3$



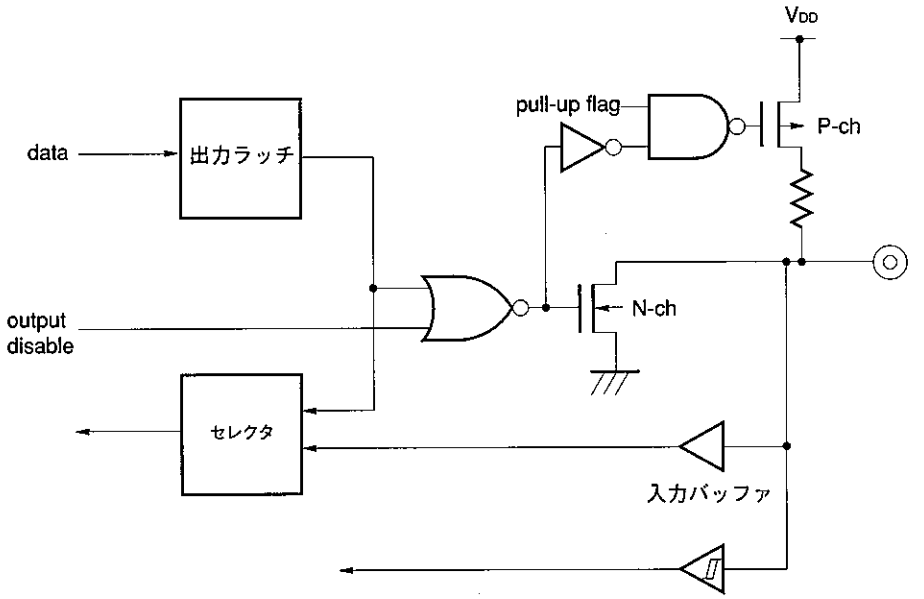
（2）P0C0/ADC0 - P0C3/ADC3



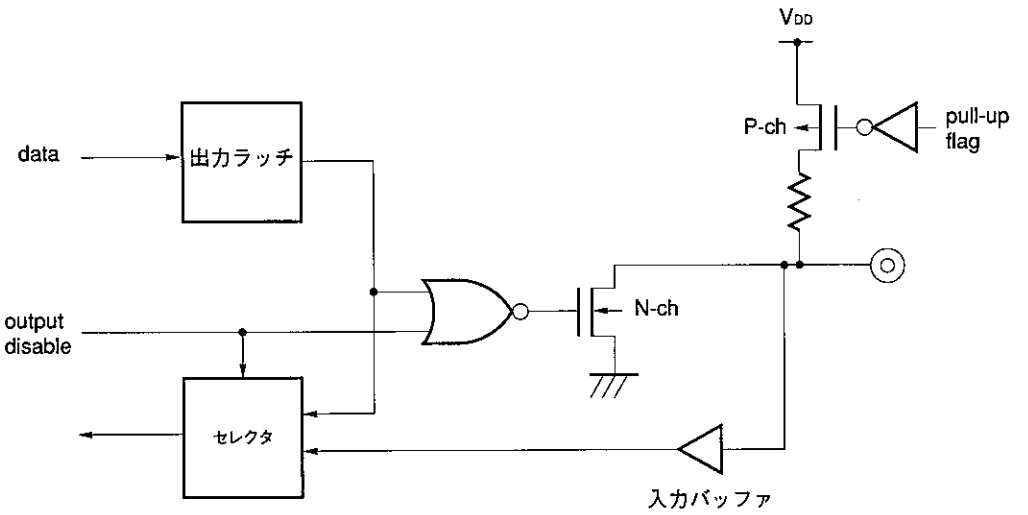
（3）P0D3/$\overline{\text{TM1OUT}}$, P0D1/SO



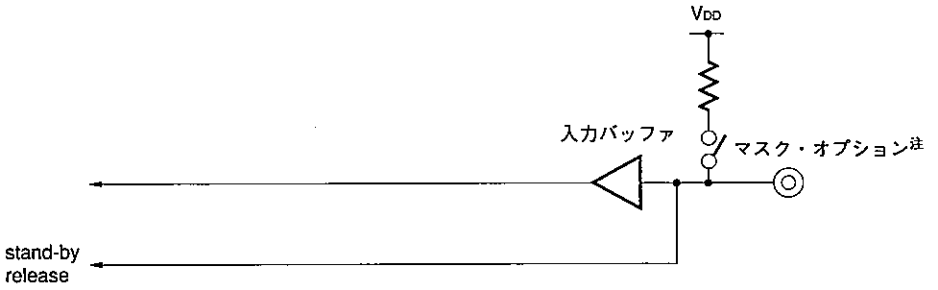
（４）$P0D_2/SI$, $P0D_0/\overline{SCK}$



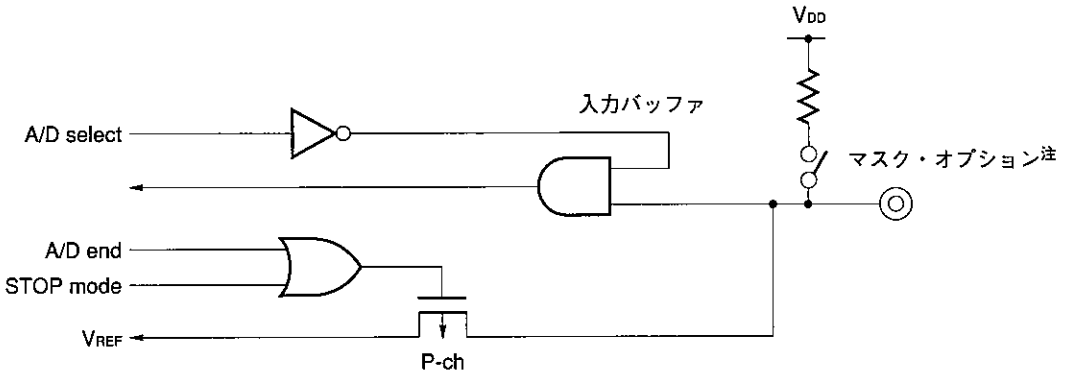
（５）$P0E_0$ - $P0E_3$



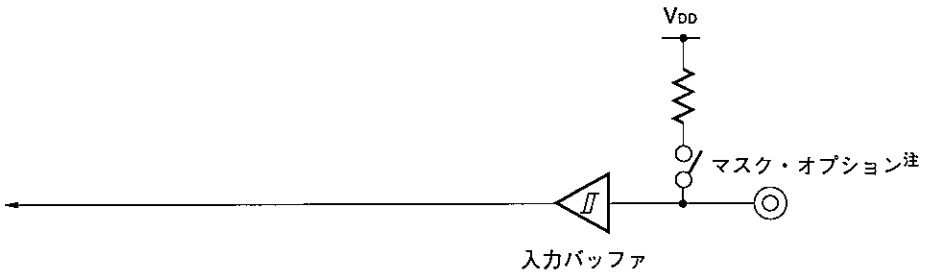
（6）P0F0/$\overline{RLS}$



（7）P0F1/VREF ★



（8）$\overline{RESET}$, INT



注　μPD17P149には，マスク・オプションによるプルアップ抵抗は内蔵されていません。 ★

2.3　未使用端子の処理

通常動作モード時，未使用端子は，次に示すような処置をしてください。

★ 表2－1　未使用端子の処理

端子名			処理方法	
			マイコン内部	マイコン外部
ポート	入力モード	P0A，P0B，P0D，P0E	ソフトウエアによるプルアップ抵抗を内蔵する	オープン
		P0C	—	プルアップ抵抗を介してV_{DD}に接続または，プルダウン抵抗を介してGNDに接続[注1]
		$P0F_1$	マスク・オプションによるプルアップ抵抗を内蔵しない	V_{DD}またはGNDに直接接続
			マスク・オプションによるプルアップ抵抗を内蔵する	オープン
		$P0F_0$[注2]	マスク・オプションによるプルアップ抵抗を内蔵しない	GNDに直接接続
	出力モード	P0A,P0B,P0C（CMOSポート）	—	オープン
		P0D(N-chオープン・ドレーン・ポート)	ロウ・レベルを出力する	
		P0E(N-chオープン・ドレーン・ポート)	ソフトウエアによるプルアップ抵抗を内蔵しないで，ロウ・レベルを出力する	
			ソフトウエアによるプルアップ抵抗を内蔵して，ハイ・レベルを出力する	
外部割り込み(INT)			マスク・オプションによるプルアップ抵抗を内蔵しない	V_{DD}またはGNDに直接接続
			マスク・オプションによるプルアップ抵抗を内蔵する	オープン
$\overline{RESET}$[注3]（内蔵のPOC回路だけを使用する場合）			マスク・オプションによるプルアップ抵抗を内蔵しない	V_{DD}に直接接続
			マスク・オプションによるプルアップ抵抗を内蔵する	

注1．外部でプルアップまたはプルダウンする場合には，ポートのドライブ能力や消費電流に注意してください。また，高い抵抗値でプルアップまたはプルダウンする場合には，その端子にノイズが乗らないように注意してください。応用回路にもよりますが，プルアップまたはプルダウンの抵抗値は，数十kΩ程度が一般的です。

2．$P0F_0/\overline{RLS}$端子はテスト・モードの設定機能を兼用しているので，未使用の場合はマスク・オプションによるプルアップ抵抗を内蔵しないで，直線GNDに接続してください。

注3．高い信頼性を必要とする応用回路では，必ず外部から$\overline{\text{RESET}}$信号を入力するように設計してください。また，$\overline{\text{RESET}}$端子はテスト・モードの設定機能を兼用しているので，未使用の場合は直接V_{DD}に接続してください。

注意　入出力モード，ソフトウエアによるプルアップ抵抗，端子の出力レベルは，プログラムの各ループ内で繰り返し設定することによって固定することを推奨します。

備考　μPD17P149には，マスク・オプションによるプルアップ抵抗とPOC回路は内蔵されていません。

2.4　$\overline{\text{RESET}}$端子とP0F0/$\overline{\text{RLS}}$端子の使用上の注意（通常動作モード時のみ）

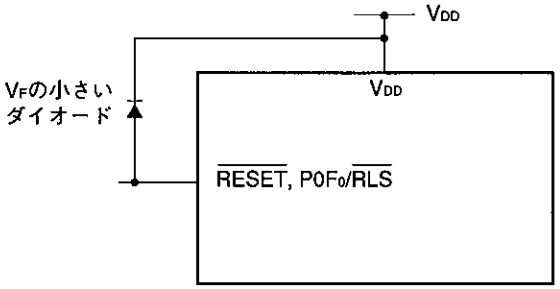
$\overline{\text{RESET}}$端子とP0F0/$\overline{\text{RLS}}$端子は，2.1　端子機能説明に示した機能のほかに，μPD17149の内部動作をテストするテスト・モードを設定する機能（ICテスト専用）を持っています。

これらの端子のいずれかにV_{DD}を越える電圧を印加すると，テスト・モードに設定されます。このため，通常動作時であってもV_{DD}を越えるようなノイズが加わった場合にはテスト・モードに入ってしまい，通常動作に支障をきたすことがあります。

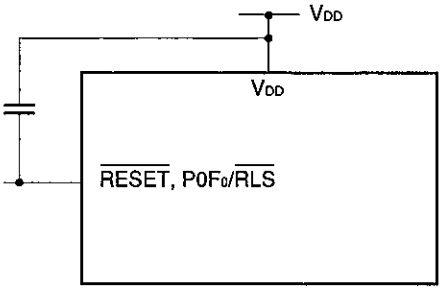
たとえば，$\overline{\text{RESET}}$端子またはP0F0/$\overline{\text{RLS}}$端子の配線の引き回しが長い場合などでは，これらの端子に布線間ノイズが加わって上記の問題を起こしてしまうことがあります。

したがって，できるだけ布線間ノイズを抑えるような配線を行ってください。どうしてもノイズが抑えられない場合は，下図のような外付け部品によるノイズ対策を実施してください。

○V_{DD}との間にV_Fの小さいダイオードを接続



○V_{DD}との間にコンデンサを接続



第3章　プログラム・メモリ（ROM）

表3－1にμPD17145, 17147, 17149, 17P149のプログラム・メモリ構成を示します。

表3－1　プログラム・メモリ構成

品　名	プログラム・メモリ容量	プログラム・メモリ番地
μPD17145	2Kバイト (1024×16ビット)	0000H - 03FFH
μPD17147	4Kバイト (2048×16ビット)	0000H - 07FFH
μPD17149	8Kバイト (4096×16ビット)	0000H - 0FFFH
μPD17P149		

プログラム・メモリは，プログラムおよび定数データ・テーブルなどを格納します。プログラム・メモリの先頭部分は，リセット・スタート・アドレスと割り込みベクタ・アドレスに割り当てられています。 ★

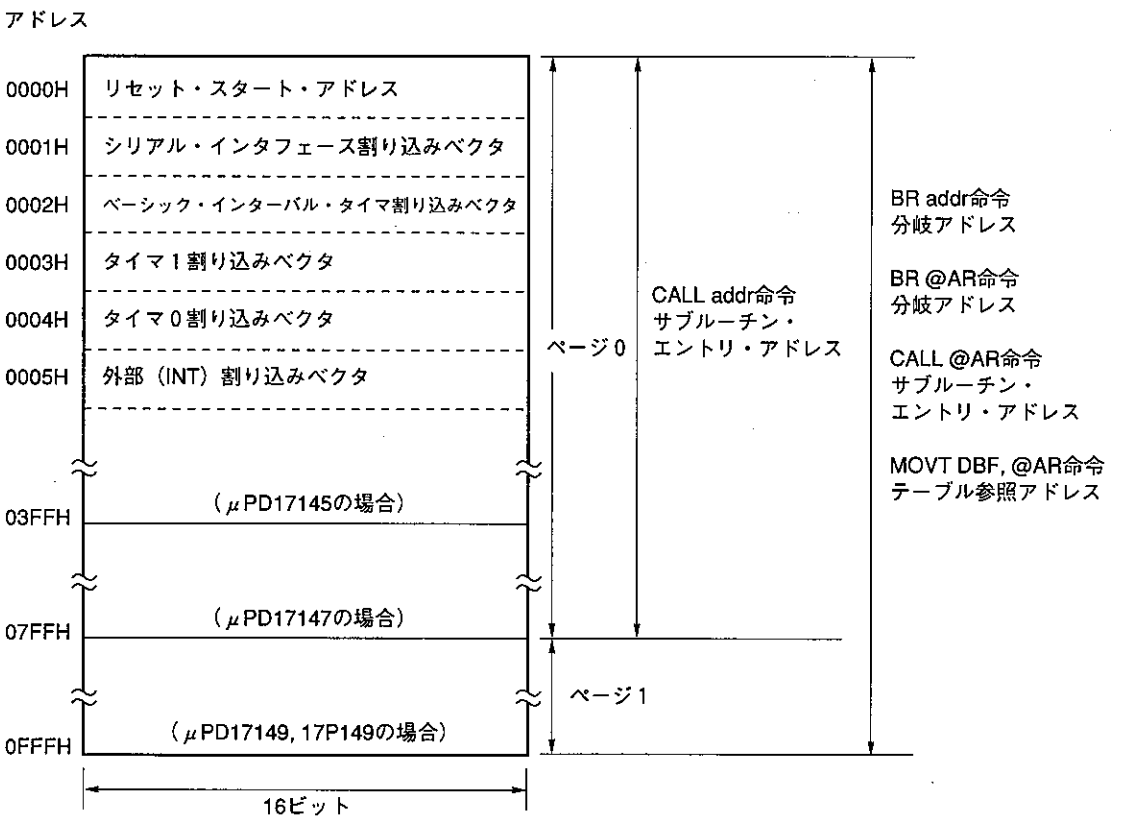
プログラム・メモリは，プログラム・カウンタによってそのアドレスを指定されます。

3.1　プログラム・メモリの構成

図3－1にプログラム・メモリ・マップを示します。プログラム・メモリは16ビットを1ステップとしており，2Kステップごとに“ページ”と呼ぶ単位に分けられています。

直接サブルーチン・コール命令によるアドレス指定可能な範囲は，プログラム・メモリの0000H - 07FFH番地（ページ0）です。分岐命令，間接サブルーチン・コール命令，テーブル参照命令によるアドレス指定可能な範囲は，それぞれのプログラム・メモリの全範囲です。

図3－1　プログラム・メモリ・マップ



3.2　プログラム・メモリの使い方

プログラム・メモリは，大別して以下の2つの機能があります。

（1）プログラムを格納しておく
（2）定数データを格納しておく

プログラムとはCPU（Central Processing Unit）を動作させる“命令”の集まりです。CPUはプログラムに書かれた命令に従い，順次処理を実行していきます。すなわち，プログラム・メモリに格納されているプログラムから順次，命令を読み出していき，各命令に従って処理を実行します。

命令はすべて16ビット長の一語命令であるため，プログラム・メモリの1つの番地に1つの命令を格納することができます。

定数データとは，たとえば表示用出力パターンのように，あらかじめ決まっているようなデータです。MOVT命令を使用することによりプログラム・メモリ上の定数データをデータ・メモリ上のデータ・バッファ（DBF）に読み出すことができます。このようにプログラム・メモリ上の定数データを読み出すことを“テーブル参照”と呼びます。

プログラム・メモリは読み出し専用メモリ（ROM：Read Only Memory）であるため，命令により書き換えることはできません。

3.2.1　プログラムの流れ

プログラム・メモリに格納されているプログラムは，通常0000H番地から，1番地ごとに順次実行されます。しかし，たとえばある条件により異なるプログラムを実行させるような場合は，プログラムの流れを変える必要があります。このような場合には，分岐命令（BR命令）を使用します。

また，同一のプログラムが何箇所にも現れる場合は，その度に同一プログラムを用いると，プログラム・メモリの利用効率が低下します。このような場合は，一箇所にプログラムをまとめておき，CALL命令で，同一のプログラムを呼び出すようにします。このプログラムを“サブルーチン”と呼びます。サブルーチンに対して通常実行しているプログラムを“メイン・ルーチン”と呼びます。

また，プログラムの流れとは関係なく，ある条件が成立したときに実行させたいプログラムがあるときは，割り込み機能を使用します。割り込み機能を使用すると，ある条件が成立したとき現在のプログラムの流れとは関係なく，あらかじめ決められた番地（ベクタ・アドレスと呼ぶ）へ分岐することができます。

（1）－（4）に，割り込みおよび命令により，プログラムが分岐する場合について説明します。

（1）ベクタ・アドレス

リセットあるいは割り込み発生時に分岐するアドレス（ベクタ・アドレス）を，表3－2に示します。

表3－2　ベクタ・アドレス

ベクタ・アドレス	割り込み要因
0000H	リセット
0001H	シリアル・インタフェース割り込み
0002H	ベーシック・インターバル・タイマ割り込み
0003H	タイマ1割り込み
0004H	タイマ0割り込み
0005H	外部割り込み（INT端子）

（2）直接分岐

直接分岐命令（BR addr）では，オペランド（addr）の値をアドレスとして分岐します。

μPD17145の場合，オペランドの下位10ビットで分岐先のプログラム・メモリ・アドレスを指定します（ただし03FFH以上を指定することはできません。もしこの範囲を越えるアドレスを指定した場合，アセンブラでエラーを発生します）。

μPD17147の場合，オペランドの全ビット（11ビット）で分岐先のアドレスを指定します（ただし07FFH以上を指定することはできません。もしこの範囲を越えるアドレスを指定した場合，アセンブラでエラーを発生します）。

μPD17149の場合，オペレーション・コード（以下オペ・コード）の最下位1ビットとオペランドの全ビット（11ビット）の計12ビットで分岐先のプログラム・メモリ・アドレスを指定します（ただし，0FFFH以上を指定することはできません。もしこの範囲を越えるアドレスを指定した場合，アセンブラでエラーを発生します）。

したがって，BR addr命令により，すべてのプログラム・メモリ・アドレスに分岐することができます。

● μPD17149のディバグ時の注意

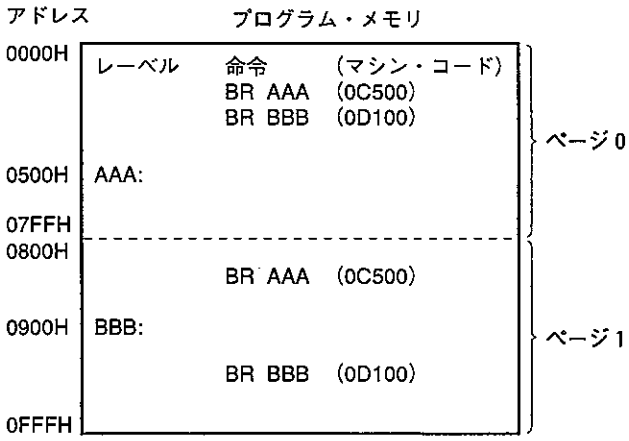
BR addr命令のマシン・コードにおいて，プログラム・メモリの番地を指すビットは，11ビット分しかありません。このため，分岐先の番地がどのページにあるかによって，BR addrのオペ・コードを変えています。したがって，BR addr命令の分岐先がページ1にある場合，マシン・コードと実際のプログラム上のアドレスとが異なることになります。

★　オペ・コードの使い分けは，通常アセンブラが自動的に行うため気にする必要はありません。ただしアセンブラを使わずに直接機械語でパッチを当てる場合などには注意が必要です。

表3－3　BR addr命令の分岐先とマシン・コードの対応

分岐先の番地	BR addrのマシン・コード
0000H - 07FFH（ページ0）	0C000H - 0C7FFH
0800H - 0FFFH（ページ1）	0D000H - 0D7FFH

図3－2　BR命令マシン・コード例



（3）間接分岐

間接分岐命令（BR @AR）ではアドレス・レジスタ（AR）の内容をアドレスとして分岐します。したがって，BR @AR命令によりすべてのプログラム・メモリ・アドレスに分岐することができます。

8.2　アドレス・レジスタ（AR）も参照してください。

（4）サブルーチン

サブルーチンへの分岐にはサブルーチン・コール命令（CALL）を使います。

CALL命令には，オペランド（addr）の値をアドレスとして分岐する直接サブルーチン・コール命令（CALL addr）とアドレス・レジスタの内容をアドレスとして分岐する間接サブルーチン・コール命令（CALL @AR）とがあります。

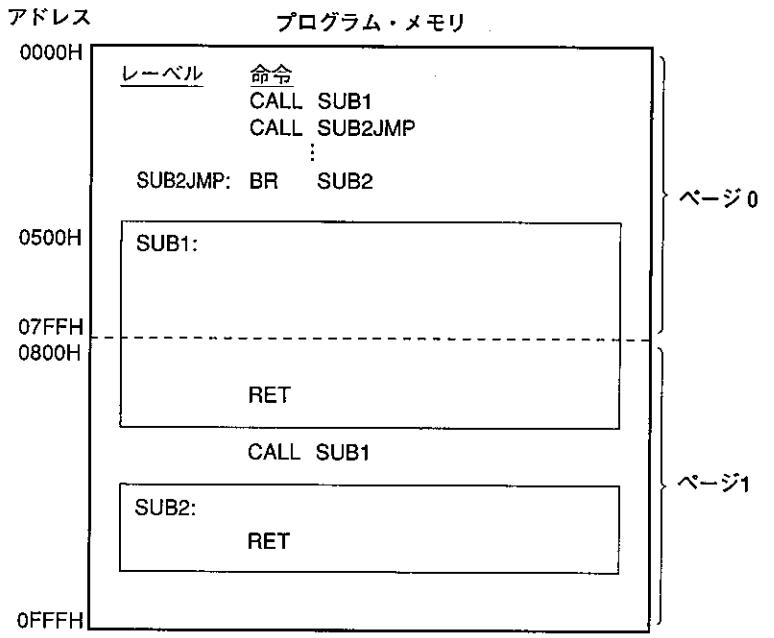
サブルーチンからの復帰命令にはRET命令またはRETSK命令を使用します。RETまたはRETSK命令を実行することによりCALL命令を実行した次のプログラム・メモリ・アドレスへ復帰します。

またRETSK命令の場合には復帰した最初の命令をNOP命令として実行します。

① 直接サブルーチン・コール

直接サブルーチン・コール命令（CALL addr）は，オペランドの全ビット（11ビット）で分岐先のプログラム・メモリ・アドレスを指定します。したがって，CALL addrを使用する場合は，サブルーチンの先頭番地をページ0内（0000H - 07FFH）に置く必要があります。そうでないサブルーチンには直接分岐することができません。このようなサブルーチンに分岐するためには，図3－3に示すように，ページ0内に分岐命令（BR）を設け，このBR命令を介して実際のサブルーチン（SUB2）を呼び出します。

図3－3　CALL addr命令



② 間接サブルーチン・コール

間接サブルーチン・コール命令（CALL @AR）は，アドレス・レジスタ（AR）の値をサブルーチン・コール先のアドレスとします。したがって，すべてのプログラム・メモリ・アドレスに分岐することができます。8.2　アドレス・レジスタ（AR）を参照してください。

3.3　テーブル参照

テーブル参照は，プログラム・メモリ内の定数データを参照するときに使用します。

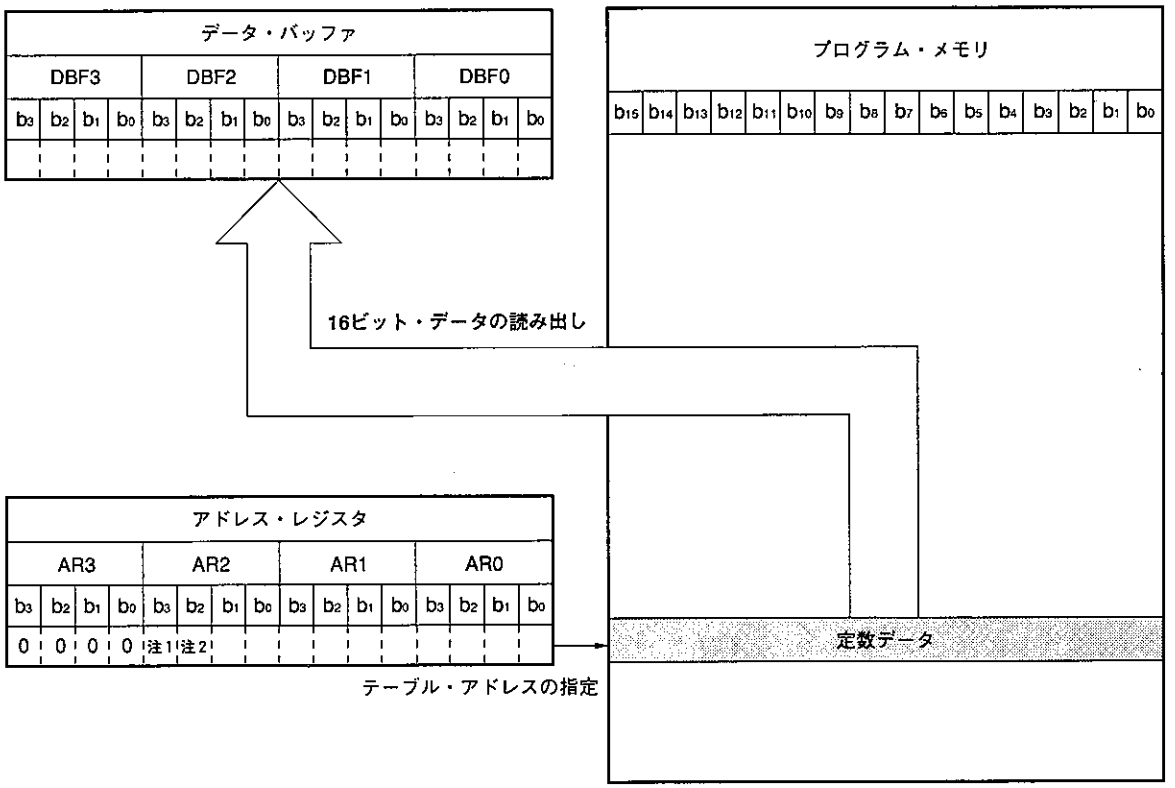
テーブル参照命令（MOVT DBF, @AR）を実行すると，アドレス・レジスタで指定されるプログラム・メモリ・アドレスの内容が，データ・バッファに格納されます。

プログラム・メモリの内容は，16ビットで構成されているので，MOVT命令でデータ・バッファに格納される定数データは16ビットになります。アドレス・レジスタを使用することにより，すべてのプログラム・メモリをテーブル参照することができます。

注意　テーブル参照を行うときは，一時的にアドレス・スタックが1レベル使用されますので，使用可能なスタック・レベルを越えないように注意してください。8.2　アドレス・レジスタ（AR）および第10章　データ・バッファ（DBF）も参照してください。

備考　テーブル参照命令の実行には，例外的に2命令サイクルを必要とします。 ★

図3－4　MOVT DBF, @AR命令



注1．μPD17145, 17147では0に固定されています。

2．μPD17145では0に固定されています。

（1）定数データ・テーブル

例1に定数データ・テーブルのテーブル参照プログラム例を示します。

例1．定数データ・テーブルのデータを読み出すプログラム

```
OFFSET      MEM     0.00H           ;オフセット・アドレスの格納エリア

ROMREF:
            ;BANK0
                                    ;定数データ・テーブルが入っている
                                    ;先頭番地をARレジスタに格納します。
            MOV     AR3, #.DL.TABLE SHR 12 AND 0FH
            MOV     AR2, #.DL.TABLE SHR 8 AND 0FH
            MOV     AR1, #.DL.TABLE SHR 4 AND 0FH
            MOV     AR0, #.DL.TABLE AND 0FH

            ;MOV     RPH, #0         ;レジスタ・ポインタをロウ・アドレ
            MOV     RPL, #7 SHL 1   ;ス7に設定します。

            ADD     AR0, OFFSET     ;オフセット・アドレスを加算します。
            ADDC    AR1, #0
            ADDC    AR2, #0
            ADDC    AR3, #0
            MOVT    DBF, @AR        ;定数データの読み出し

TABLE:
            DW      0001H           ;OFFSET = 0Hのとき
            DW      0002H
            DW      0004H
            DW      0008H
            DW      0010H
            DW      0020H
            DW      0040H
            DW      0080H
            DW      0100H
            DW      0200H
            DW      0400H
            DW      0800H
            DW      1000H
            DW      2000H
            DW      4000H
            DW      8000H           ;OFFSET = 0FHのとき

            END
```

（2）分岐先アドレス・テーブル

例2に分岐先アドレス・テーブルのテーブル参照プログラム例を示します。

例2．分岐先アドレス・テーブルのアドレスに分岐するプログラム

```
OFFSET      MEM     0.00H           ；オフセット・アドレスの格納エリア

ROMREF：
            ；BANK0                 ；定数データ・テーブル先頭番地をAR
                                    ；レジスタに格納します。
            MOV     AR3, #.DL.TABLE SHR 12 AND 0FH
            MOV     AR2, #.DL.TABLE SHR 8 AND 0FH
            MOV     AR1, #.DL.TABLE SHR 4 AND 0FH
            MOV     AR0, #.DL.TABLE AND 0FH

            ；MOV   RPH, #0         ；レジスタ・ポインタをロウ・アドレ
            MOV     RPL, #7 SHL 1   ；ス7に設定します。

            ADD     AR0, OFFSET     ；オフセット・アドレスを加算します。
            ADDC    AR1, #0
            MOVT    DBF,@AR         ；分岐先アドレスの読み出し
            PUT     AR, DBF         ；AR←分岐先アドレス
            BR      @AR

TABLE：
            DW      0001H           ；OFFSET＝0Hのとき
            DW      0002H
            DW      0004H
            DW      0008H
            DW      0010H
            DW      0020H
            DW      0040H
            DW      0080H
            DW      0100H
            DW      0200H           ；OFFSET＝9Hのとき

            END
```

〔メ　モ〕

第4章　プログラム・カウンタ（PC）

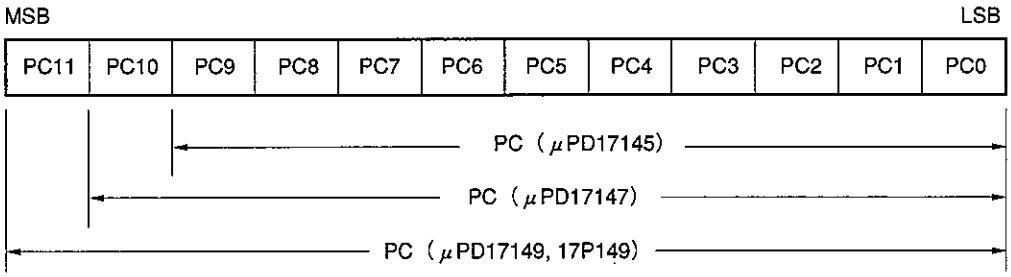
プログラム・カウンタは，プログラム・メモリのアドレスを指定するために使用します。

4.1　プログラム・カウンタの構成

プログラム・カウンタは，図4－1に示すように12ビットのバイナリ・カウンタで構成されています。

備考　プログラム・カウンタのサイズは製品ごとに異なります。μPD17145は10ビット，μPD17147は11ビットです。

図4－1　プログラム・カウンタ



4.2　プログラム・カウンタの動作

プログラム・カウンタは，通常，命令を1つ実行するたびに自動的にインクリメリントされます。また，リセット時，分岐命令，サブルーチン・コール命令，リターン命令，テーブル参照命令が実行されたときおよび割り込みが受け付けられたときには，次に実行すべきプログラム・メモリのアドレスがプログラム・カウンタに設定されます。

4.2.1 - 4.2.7に各命令実行時のプログラム・カウンタの動作を示します。

図4－2　命令実行後のプログラム・カウンタの値

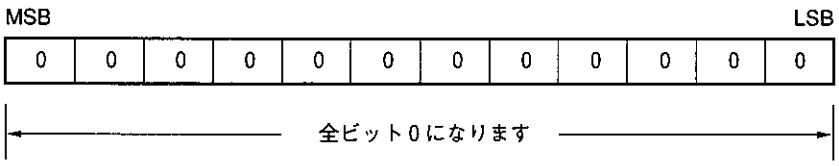
プログラム・カウンタのビット / 命令	プログラム・カウンタの値											
	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
リセット時	0	0	0	0	0	0	0	0	0	0	0	0
BR addr	0	addrで指定した値										
	1											
CALL addr	0											
BR @AR CALL @AR （MOVT DBF, @AR）	アドレス・レジスタ（AR）の内容											
RET RETSK RETI	スタック・ポインタで示されるアドレス・スタックの内容（戻り番地）											
割り込み受け付け時	各割り込みのベクタ・アドレス											

備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

4.2.1　リセット時

$\overline{\text{RESET}}$端子をロウ・レベルにすることにより，プログラム・カウンタが000Hになります。

図4－3　リセット時のプログラム・カウンタの値

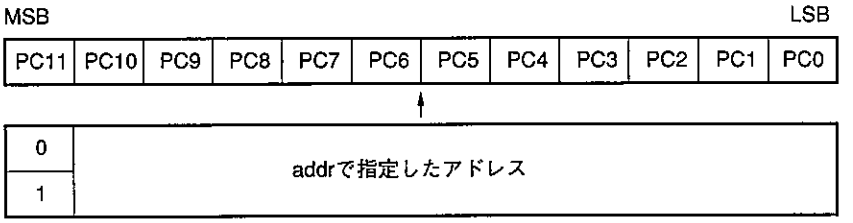


4.2.2　分岐命令（BR）実行時

分岐命令には，オペランドに指定したアドレスに分岐する直接分岐命令（BR addr）とアドレス・レジスタが示すアドレスに分岐する間接分岐命令（BR @AR）があります。

BR addr命令により指定したアドレスがプログラム・カウンタに設定されます。

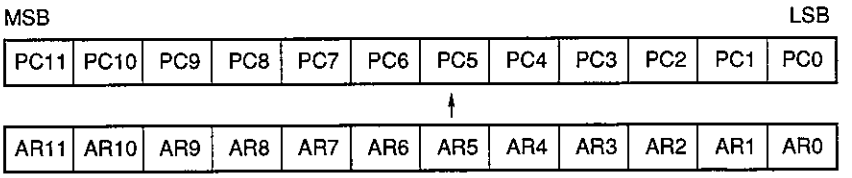
図4－4　BR addr命令実行時のプログラム・カウンタの値



備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

BR @AR命令によりアドレス・レジスタの値がプログラム・カウンタに設定されます。

図4－5　BR @AR命令実行時のプログラム・カウンタの値



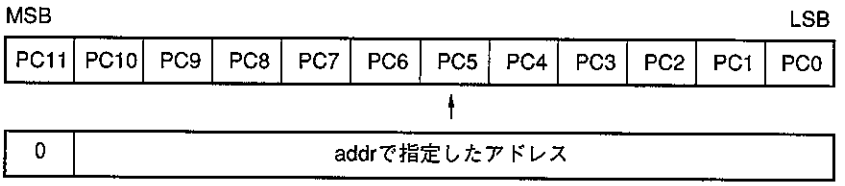
備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

4.2.3　サブルーチン・コール命令（CALL）実行時

サブルーチン・コール命令には，オペランドに指定したアドレスに分岐する直接サブルーチン・コール命令（CALL addr）とアドレス・レジスタが示すアドレスに分岐する間接サブルーチン・コール命令（CALL @AR）があります。

CALL addr命令により，プログラム・カウンタの値がアドレス・スタック・レジスタに退避されたあと，オペランドに記述したアドレスがプログラム・カウンタに設定されます。CALL addr命令で指定できるアドレスの範囲は0000H - 07FFHです。

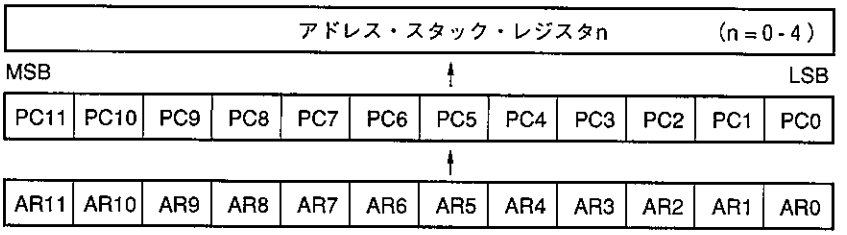
図4－6　CALL addr命令実行時のプログラム・カウンタの値



備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

CALL @AR命令により，プログラム・カウンタの値がアドレス・スタック・レジスタに退避されたあと，アドレス・レジスタの値がプログラム・カウンタに設定されます。

図4－7　CALL @AR命令実行時のプログラム・カウンタの値

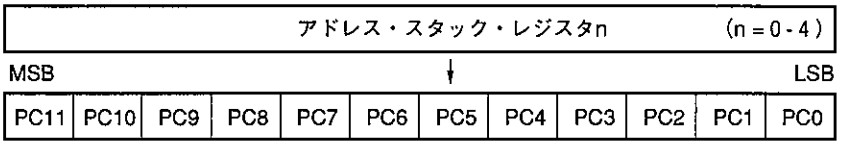


備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

4.2.4　リターン命令（RET, RETSK, RETI）実行時

RET命令，RETSK命令，RETI命令により，アドレス・スタック・レジスタに退避された値が，プログラム・カウンタに復帰されます。

図4－8　RET命令，RETSK命令，RETI命令実行時のプログラム・カウンタの値



備考　μPD17145にはPC11, PC10がありません。μPD17147にはPC11がありません。

4.2.5　テーブル参照命令（MOVT）実行時

“MOVT DBF, @AR” 命令により，プログラム・カウンタの値がアドレス・スタック・レジスタに退避されたあと，アドレス・レジスタの値がプログラム・カウンタに設定され，そのプログラム・メモリの内容がデータ・バッファ（DBF）に読み出されます。また，プログラム・メモリの内容を読み出したあと，アドレス・スタック・レジスタに退避された値がプログラム・カウンタに復帰します。

注意　テーブル参照を行うときは，一時的にスタックが1レベル使用されますのでスタック・レベルにご注意ください。

4.2.6　スキップ命令（SKE, SKGE, SKLT, SKNE, SKT, SKF）実行時

スキップ命令のスキップ条件が成立したとき，スキップ命令の直後の命令はNOP命令として実行されます。したがって，スキップ条件の成否にかかわらず，実行する命令数および命令実行時間は変わりません。

4.2.7　割り込み受け付け時

割り込みが受け付けられると，プログラム・カウンタの値がアドレス・スタックに退避されたあと，受け付けられた割り込みに対するベクタ・アドレスがプログラム・カウンタに設定されます。

〔メ　モ〕

第5章　スタック

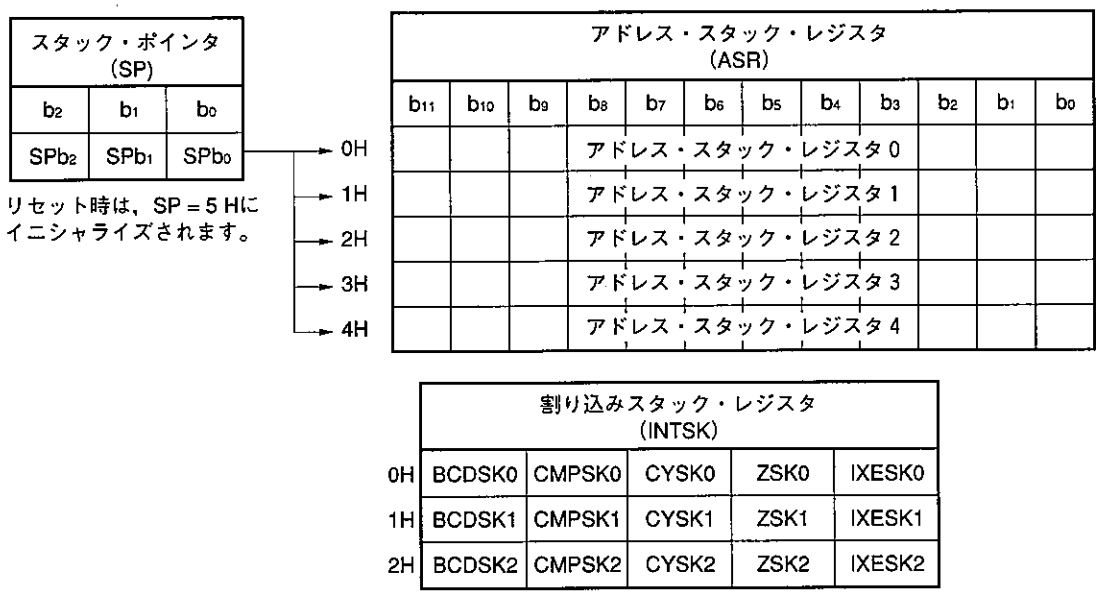
スタックとはサブルーチン・コール時や割り込み受け付け時にプログラムの戻り番地や後述するシステム・レジスタの内容を退避するためのレジスタです。

5.1　スタックの構成

スタックの構成を図5－1に示します。

スタックは，3ビットのバイナリ・カウンタであるスタック・ポインタ（SP）1個，10ビット（μPD17145）/11ビット（μPD17147）/12ビット（μPD17149）のアドレス・スタック・レジスタ（ASR）5個および5ビットの割り込みスタック・レジスタ（INTSK）3個より構成されています。

図5－1　スタックの構成



5.2　スタックの機能

スタックは，サブルーチン・コール命令実行時やテーブル参照命令実行時に戻り番地を退避するために使用します。また，割り込み受け付け時には，プログラムの戻り番地およびプログラム・ステータス・ワード（PSWORD）が自動的に退避されます。なお，退避後PSWORDは全ビットが0にクリアされます。

5.3　アドレス・スタック・レジスタ（ASR）

アドレス・スタック・レジスタ（ASR）は，図5－1に示すように5×12ビットで構成されるレジスタです。

●CALL addr 命令，CALL @AR命令の実行時，“MOVT DBF，@AR” 命令の第1命令サイクル実行時，および割り込み受け付け時に戻り番地が格納されます。
●PUSH AR 命令実行時は，アドレス・レジスタ（AR）の内容が格納されます。データが格納されるASRは，命令実行時のスタック・ポインタ（SP）の値に－1した値で指定されます。
●RET 命令，RETSK 命令の実行時，“MOVT DBF，@AR” 命令の第2命令サイクル実行時，RETI命令実行時はスタック・ポインタで指定されたASR の内容（戻り番地）をプログラム・カウンタに復帰して，スタック・ポインタの値を＋1します。
●POP AR命令実行時はスタック・ポインタで指定されたASR の値をアドレス・レジスタに転送して，スタック・ポインタの値を＋1します。

注意　CALL addr 命令，CALL @AR命令や割り込みを実行した結果，スタック・ポインタがアンダフローした場合には暴走したものと見なし，内部リセット信号を発生し，ハードウエアを初期状態にイニシャライズして，0000H 番地スタートします。

備考　ASRのサイズは製品ごとに異なります。μPD17145は5×10ビット，μPD17147は5×11ビットです。

5.4　割り込みスタック・レジスタ（INTSK）

割り込みスタック・レジスタ（INTSK）は，図5－1に示すように3×5ビットで構成されるレジスタです。

●割り込みが受け付けられると，後述するシステム・レジスタ（SYSREG）の中のプログラム・ステータス・ワード（PSWORD）の各フラグ（BCD,CMP,CY,Z,IXE），計5ビットをINTSKに退避します。なお，退避後PSWORDは全ビットが0にクリアされます。
●RETI命令が実行されると，INTSK の内容をPSWORDに復帰します。
●INTSK は，割り込みが受け付けられるごとにデータを退避していきます。

注意　3レベルを越えて割り込みが受け付けられると，最初のデータは失われてしまいます。

5.5　スタック・ポインタ（SP）と割り込みスタック・レジスタ

スタック・ポインタ（SP）は図5－1に示したように5個のアドレス・スタック・レジスタのアドレスを指定する3ビットのバイナリ・カウンタで，レジスタ・ファイルの01H番地に割り当てられており，リセット時には，5Hにイニシャライズされます。

- ●SPは，表5－1に示すようにCALL addr命令，CALL @AR命令の実行時，“MOVT DBF,@AR”命令の第1命令サイクルの実行時，PUSH AR命令実行時および割り込み受け付け時に－1されます。
- ●RET命令，RETSK命令の実行時，“MOVT DBF,@AR”命令の第2命令サイクル，POP AR命令の実行時およびRETI命令の実行時に＋1されます。

なお，割り込みが受け付けられるとSPのほかに割り込みスタック・レジスタのカウンタも－1されます。割り込みスタック・レジスタのカウンタが＋1されるのはRETI命令の実行時だけです。

表5－1　スタック・ポインタの動作

命　令	スタック・ポインタ(SP)の値	割り込みスタック・レジスタのカウンタ
CALL addr CALL @AR MOVT DBF, @AR(第1命令サイクル) PUSH AR	－1	変化しない
RET RETSK MOVT DBF, @AR(第2命令サイクル) POP AR	＋1	
割り込み受け付け	－1	－1
RETI	＋1	＋1

備考　“MOVT DBF,@AR”命令の実行には，例外的に2命令サイクルを必要とします。 ★

スタック・ポインタ（SP）は前述したように3ビットのバイナリ・カウンタであるため，とり得る値は0H-7Hまでの8通りになりますが，アドレス・スタック・レジスタは5個しかないためスタック・ポインタ（SP）の値が6以上になると内部リセット信号が発生します（暴走対策）。

また，スタック・ポインタ（SP）はレジスタ・ファイル上に配置されているため，POKE命令を用いてレジスタ・ファイルを操作することにより，値を直接書き込むことが可能です。このときはスタック・ポインタ（SP）の値は変わりますが，アドレス・スタック・レジスタの値には何の影響も与えません。もちろん，PEEK命令を用いてスタック・ポインタ（SP）を読み出すことも可能です。

リセット時にはスタック・ポインタ（SP）は5Hになります。

5.6　スタックの動作

5.6.1-5.6.3に命令ごとのスタック動作を示します。

5.6.1　CALL命令，RET 命令，RETSK 命令実行時

表5－2にCALL命令，RET 命令，RETSK 命令実行時のスタック・ポインタ（SP），アドレス・スタック・レジスタおよびプログラム・カウンタ（PC）の動作を示します。

表5－2　CALL命令，RET命令，RETSK命令の動作

命　令	動　作
CALL addr CALL @AR	①スタック・ポインタ(SP)の値を－1 ②スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタにプログラム・カウンタ(PC)の値を退避 ③命令のオペランド(addrまたは@AR)で指定される値をプログラム・カウンタに転送
RET RETSK	①スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタの値をプログラム・カウンタ(PC)に復帰 ②スタック・ポインタ(SP)の値を＋1

RETSK命令実行時は復帰後の最初の命令はNOP命令になります。

5.6.2　テーブル参照命令（MOVT DBF, @AR命令）

表5－3にテーブル参照命令実行時の動作を示します。

表5－3　“MOVT DBF, @AR”命令の動作

命　令	命令サイクル	動　作
MOVT DBF,@AR	第一	①スタック・ポインタ(SP)の値を－1 ②スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタにプログラム・カウンタ(PC)の値を退避 ③アドレス・レジスタ(AR)の値をプログラム・カウンタ(PC)へ転送
	第二	④プログラム・カウンタ(PC)で指定されるプログラム・メモリ(ROM)の内容をデータ・バッファ(DBF)へ転送 ⑤スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタの値をプログラム・カウンタ(PC)へ復帰 ⑥スタック・ポインタ(SP)の値を＋1

★　注意　“MOVT DBF,@AR”命令を実行すると，一時的にアドレス・スタックが1レベル使用されますので，使用可能なスタック・レベルを越えないように注意してください。

★　備考　“MOVT DBF,@AR”命令の実行には，例外的に2命令サイクルを必要とします。

5.6.3　割り込み受け付け時とRETI命令実行時

表5－4に割り込み受け付け時とRETI命令実行時のスタック動作を示します。

表5－4　割り込み受け付け時とRETI命令の動作

命　令	動　作
割り込み受け付け時	①スタック・ポインタ(SP)の値を－1 ②スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタにプログラム・カウンタ(PC)の値を退避 ③割り込みスタック・レジスタへPSWORDの各フラグ(BCD, CMP, CY, Z, IXE)の値を退避 ④ベクタ・アドレスをプログラム・カウンタ(PC)へ転送
RETI	①PSWORDの各フラグ(BCD, CMP, CY, Z, IXE)へ割り込みスタック・レジスタの値を復帰 ②スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタの値をプログラム・カウンタ(PC)へ復帰 ③スタック・ポインタ(SP)を＋1

5.7　スタックのネスティング・レベルとPUSH命令およびPOP命令

スタック・ポインタ（SP）はサブルーチン・コール命令やリターン命令などで単純に＋1，－1される3ビットのカウンタとして動作しています。したがってスタック・ポインタ（SP）の値が0Hのときに CALL命令，MOVT命令の実行および割り込み受け付けが行われ，スタック・ポインタ（SP）の値は－1されて7Hになると，μPD17149は，プログラムが正常に動作していないと判断し，内部リセット信号を発生します。

アドレス・スタック・レジスタを多く使用する場合には，このような事態を避けるため，必要に応じてPUSH命令およびPOP命令を用いてアドレス・スタック・レジスタの内容を退避／復帰します。

表5－5にPUSH命令およびPOP命令の動作を示します。

表5－5　PUSH命令およびPOP命令の動作

命　令	動　作
PUSH	①スタック・ポインタ(SP)の値を－1 ②スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタにアドレス・レジスタ(AR)の値を転送
POP	①スタック・ポインタ(SP)で指定されるアドレス・スタック・レジスタの値をアドレス・レジスタ(AR)に転送 ②スタック・ポインタ(SP)の値を＋1

〔メ　モ〕

第6章　データ・メモリ（RAM）

データ・メモリ（RAM）とは，演算，制御等のデータを記憶するメモリです。命令により常時データの書き込みや読み出しが行えます。

6．1　データ・メモリの構成

データ・メモリは，7ビットからなる番地（アドレス）が付けられており，上位3ビットを“ロウ・アドレス”下位4ビットを“カラム・アドレス”と呼びます。

たとえば，1AHというアドレスを考えれば，ロウ・アドレスが1Hで，カラム・アドレスが0AHということになります。

1つのアドレスは4ビット（＝1ニブル）のメモリで構成されています。

備考　8ビットをひとまとめにした“バイト（Byte）”という単位があるように，4ビットをひとまとめにした“ニブル（Nibble）”という単位があります。

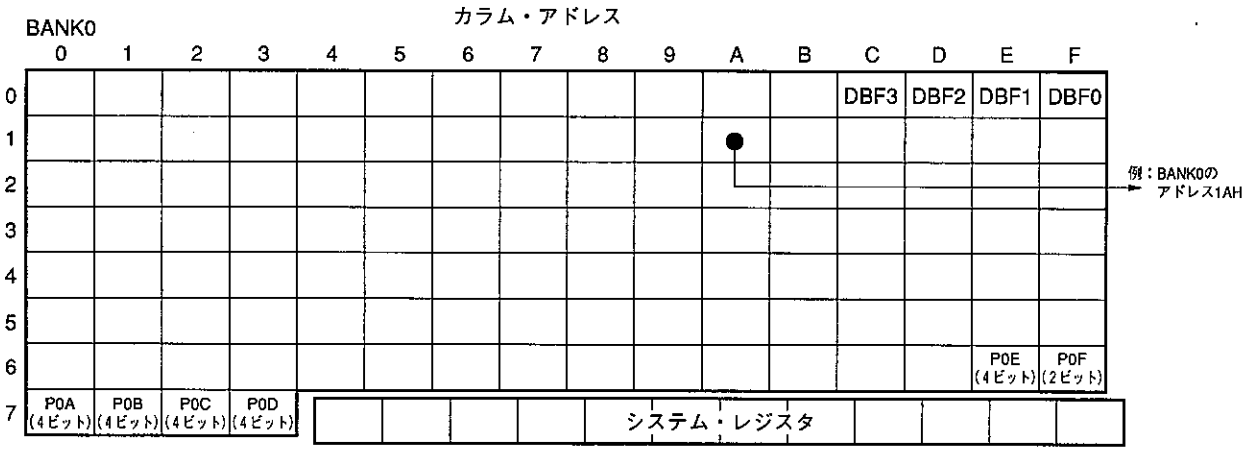
4ビット＝1ニブル
8ビット＝2ニブル＝1バイト
16ビット＝4ニブル＝2バイト

データ・メモリには，ユーザが自由にデータなどを格納しておける領域以外に，あらかじめ特別な機能が割り当てられている領域があります。

特別な機能を持つ領域は次のとおりです。

- ●システム・レジスタ（SYSREG）　（第8章　システム・レジスタ（SYSREG）参照）
- ●データ・バッファ（DBF）　（第10章　データ・バッファ（DBF）参照）
- ●ポート・レジスタ　（第12章　ポ ー ト参照）

図6－1　データ・メモリの構成



第7章　ジェネラル・レジスタ（GR）

ジェネラル・レジスタは，その名が示すように汎用のレジスタで，データ転送，演算などに使用します。

17Kシリーズでは，ジェネラル・レジスタは固定された領域ではなく，ジェネラル・レジスタ・ポインタ（RP）により，データ・メモリ上に指定される領域です。したがって，データ・メモリ領域の一部を必要に応じて，汎用レジスタとして指定できますので，データ・メモリ間のデータ転送やデータ・メモリに対する演算などを1命令で実現できます。

7.1　ジェネラル・レジスタ・ポインタ（RP）

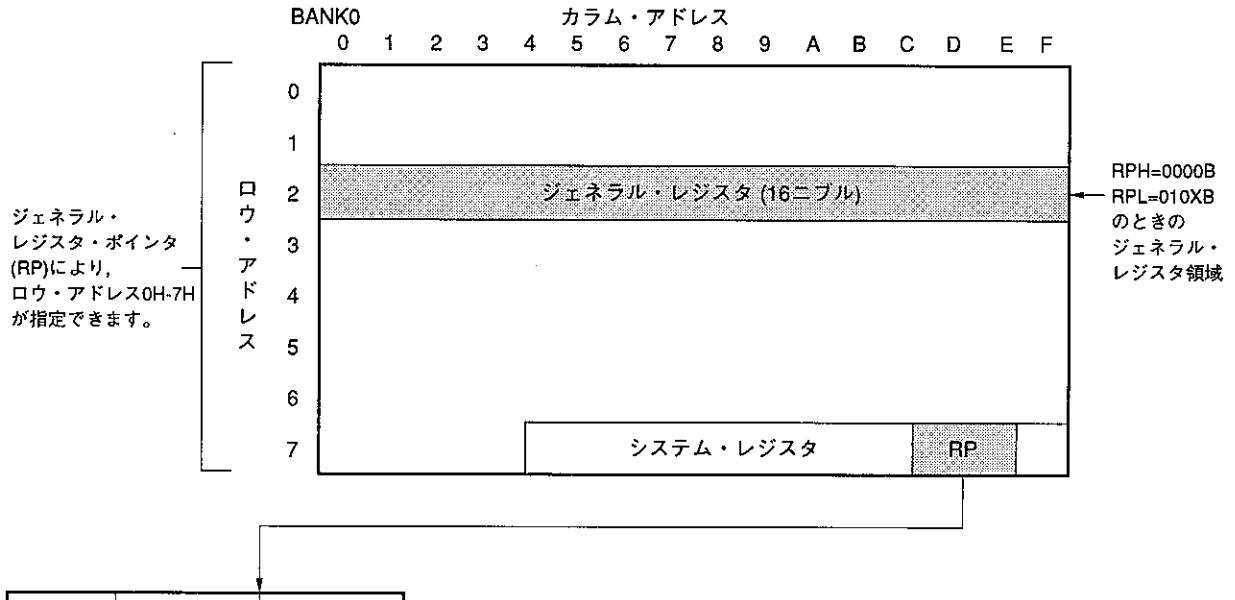
RPは，データ・メモリの一部を汎用レジスタに指定するポインタです。RPには，汎用レジスタに指定したいデータ・メモリのバンクとロウ・アドレスを設定します。RPはシステム・レジスタ（第8章　システム・レジスタ（SYSREG）参照）の7DH（RPH）と7EH（RPL）の上位3ビットの計7ビットに割り付けられています。

RPHにはバンクを，RPLにはデータ・メモリ・ロウ・アドレスを設定します。

注意　RPLの最下位ビットには，BCDフラグ（8.7　プログラム・ステータス・ワード（PSWORD）参照）が割り付けられています。

備考　μPD17149のRPHは，“0固定”になっており，必ずバンク0になります（バンク0以外になることを防止しています）。

図7－1　ジェネラル・レジスタ・ポインタの構成



アドレス	7DH				7EH			
名　称	ジェネラル・レジスタ・ポインタ (RP)							
記　号	RPH				RPL			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	0	0	0	0				BCD
リセット	0	0	0	0	0	0	0	

第8章　システム・レジスタ（SYSREG）

システム・レジスタ（SYSREG）は，直接CPUの制御を行うためのレジスタでデータ・メモリ上に配置されています。

8.1　システム・レジスタの構成

図8－1にシステム・レジスタのデータ・メモリ上の配置を示します。図8－1に示すようにシステム・レジスタは，データ・メモリの74H - 7FH番地に配置されています。

また，システム・レジスタはデータ・メモリ上に配置されているので，すべてのデータ・メモリ操作命令で操作することができます。したがって，システム・レジスタをジェネラル・レジスタに指定することも可能です。

図8－1　システム・レジスタのデータ・メモリ上の配置

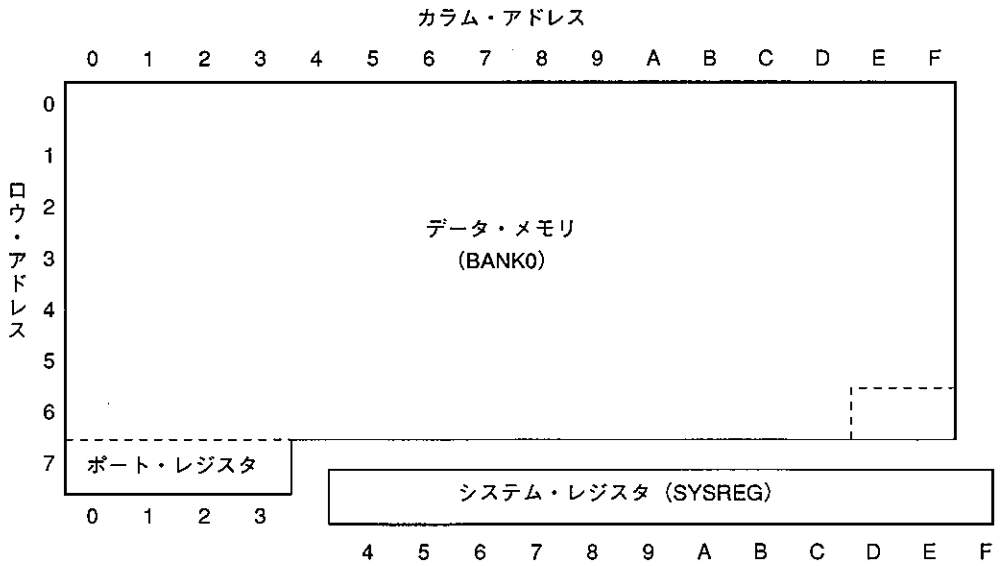


図8－2にシステム・レジスタの構成を示します。図8－2に示すようにシステム・レジスタは，次の7個のレジスタで構成されています。

- アドレス・レジスタ　(AR)
- ウインドウ・レジスタ　(WR)
- バンク・レジスタ　(BANK)
- インデクス・レジスタ　(IX)
- データ・メモリ・ロウ・アドレス・ポインタ　(MP)
- ジェネラル・レジスタ・ポインタ　(RP)
- プログラム・ステータス・ワード　(PSWORD)

図8－2　システム・レジスタの構成

アドレス	74H				75H				76H				77H				78H				79H				7AH				7BH				7CH				7DH				7EH				7FH			
名　称	アドレス・レジスタ（AR）																ウインドウ・レジスタ（WR）				バンク・レジスタ（BANK）				インデクス・レジスタ（IX） データ・メモリ・ロウ・アドレス・ポインタ（MP）												ジェネラル・レジスタ・ポインタ（RP）								プログラム・ステータス・ワード（PSWORD）			
記　号	AR3				AR2				AR1				AR0				WR				BANK				IXH MPH				IXM MPL				IXL				RPH				RPL				PSW			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ注1	0	0	0	0	注2				(AR)												0	0	0	0	MPE	0	0	0	(IX) (MP)								0	0	0	0 (RP)				BCD	CMP	CY	Z	IXE
リセット時の初期値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	不定				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

注1．この欄の0が書かれている部分は“0固定”を表します。

2．μPD17145の場合，AR2のb_3，b_2は0に固定されています。μPD17147の場合，AR2のb_3は0に固定されています。

8.2　アドレス・レジスタ（AR）

8.2.1　アドレス・レジスタの構成

図8－3にアドレス・レジスタの構成を示します。

図8－3に示すようにアドレス・レジスタはシステム・レジスタの74H - 77H番地の16ビットで構成されています。ただし，上位4/5/6ビットは常に0に固定されているために実際は12/11/10ビットで構成されています。リセット時は，16ビットすべてが0にリセットされます。

図8－3　アドレス・レジスタの構成

アドレス	74H				75H				76H				77H			
名　称	アドレス・レジスタ（AR）															
記　号	AR3				AR2				AR1				AR0			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	←							(AR)								→
	0	0	0	0	注	注										
リセット	0				0				0				0			

注　μPD17145の場合，AR2のb_3，b_2は0に固定されています。

　　μPD17147の場合，AR2のb_3は0に固定されています。

8.2.2　アドレス・レジスタの機能

アドレス・レジスタ（AR）は，BR @AR命令，CALL @AR命令，“MOVT DBF, @AR”命令実行時にプログラム・メモリ・アドレスを指定します。また，スタック操作命令（PUSH AR, POP AR）によりARの内容をアドレス・スタック・レジスタ（ASR）に書き込んだり，ASRの内容をARに読み出すことができます。

アドレス・レジスタは専用のインクリメント命令（INC AR）を使用することにより，インクリメントすることができます。

（1）間接分岐命令（BR @AR）

BR @AR命令を実行することにより，アドレス・レジスタの内容をプログラム・メモリ・アドレスとして分岐することができます。

（2）間接サブルーチン・コール命令（CALL @AR）

CALL @AR命令を実行することにより，アドレス・レジスタの内容をプログラム・メモリ・アドレスとしてサブルーチンを呼び出すことができます。

（3）テーブル参照命令（MOVT DBF, @AR）

“MOVT DBF, @AR”命令を実行することにより，アドレス・レジスタの内容をプログラム・メモリ・アドレスとするプログラム・メモリの内容（16ビットのデータ）をデータ・バッファ（データ・メモリのBANK0の0CH - 0FH）に読み出すことができます。

（4）スタック操作命令（PUSH AR, POP AR）

PUSH AR命令はスタック・ポインタ（SP）をデクリメントしたあと，スタック・ポインタで指定されるアドレス・スタック・レジスタにアドレス・レジスタの内容を格納します。

POP AR命令はスタック・ポインタで指定されるアドレス・スタック・レジスタの内容をアドレス・レジスタに転送したあと，スタック・ポインタをインクリメントします。

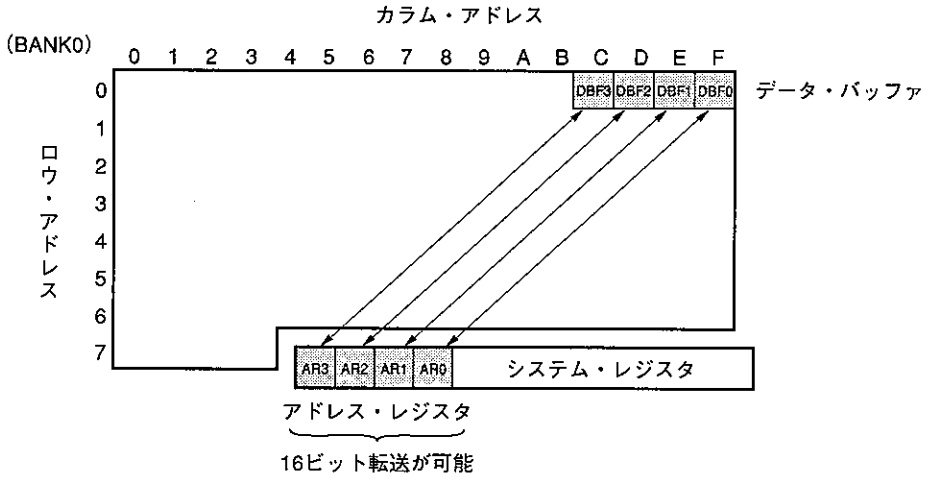
第5章　スタックも参照してください。

（5）周辺ハードウエア・レジスタとしてのアドレス・レジスタ

アドレス・レジスタは，4ビットごとに操作することはもちろん，アドレス・レジスタを周辺ハードウエア・レジスタとして扱うことによって，データ・バッファと16ビット・データ転送をすることもできます。すなわち，“PUT AR, DBF”および“GET DBF, AR”命令を用いることにより，データ・バッファと16ビット・データ転送ができます。

なお，データ・バッファは，データ・メモリのBANK0の0CH - 0FHに割り当てられています。

図8－4　周辺ハードウエア・レジスタとしてのアドレス・レジスタ



8.3　ウインドウ・レジスタ（WR）

8.3.1　ウインドウ・レジスタの構成

図8－5にウインドウ・レジスタの構成を示します。

図8－5に示すようにウインドウ・レジスタはシステム・レジスタの78H番地の4ビットで構成されています。リセット時は不定になります。なお，HALTモード，STOPモードからの解除に$\overline{\text{RESET}}$入力を用いたときは以前の状態を保持しています。

図8－5　ウインドウ・レジスタの構成

アドレス	78H			
名　称	ウインドウ・レジスタ			
記　号	WR			
ビット	b_3	b_2	b_1	b_0
データ	←			→
リセット	不定			

8.3.2　ウインドウ・レジスタの機能

ウインドウ・レジスタはレジスタ・ファイル（RF）とのデータ転送に使用します。

レジスタ・ファイルとのデータ転送は“PEEK WR, rf”命令および“POKE rf, WR”命令で行います。

詳細は9.2.3　レジスタ・ファイルの操作命令を参照してください。

8.4　バンク・レジスタ（BANK）

図8－6にバンク・レジスタの構成を示します。

バンク・レジスタはシステム・レジスタの79H番地の4ビットで構成されています。ただし4ビットは0に固定されています。

図8－6　バンク・レジスタの構成

アドレス	79H			
名　称	バンク・レジスタ			
記　号	BANK			
ビット	b_3	b_2	b_1	b_0
データ	0	0	0	0
	(BANK)			
	←			→
リセット	0			

8.5　インデクス・レジスタ（IX）とデータ・メモリ・ロウ・アドレス・ポインタ（メモリ・ポインタ：MP）

8.5.1　インデクス・レジスタ（IX）

IXは，データ・メモリのアドレス修飾に使います。MPとの違いはその修飾対象がバンクおよびオペランドmで指定されるアドレスであるという点です。

図8－7に示すように，IXはシステム・レジスタの7AH（IXH），7BH（IXM），7CH（IXL）の計12ビットに割り付けられています。また，IXによるアドレス修飾を許可するインデクス・レジスタ・イネーブル・フラグ（IXE）がPSWの最下位ビットに割り付けられています。

IXE＝1のとき，オペランドmで指定されたデータ・メモリのアドレスはmではなく，mとIXM-IXLとの論理和で表されるアドレスになります。このとき指定バンクもBANKとIXHとの論理和で表されるバンクになります。

備考　μPD17149のIXHは“0固定”になっており，IXE＝1であってもバンクが修飾されることはありません（バンク0以外になることを防止しています）。

8.5.2　データ・メモリ・ロウ・アドレス・ポインタ（メモリ・ポインタ：MP）

MPは，データ・メモリのアドレス修飾に使用します。IXとの違いは，その修飾対象がバンクおよびオペランド@rで間接指定されたアドレスのロウ・アドレスであるという点です。

図8－7に示すように，MPHとIXH，MPLとIXMは，同じアドレス（システム・レジスタの7AH，7BH）に割り付けられています。実際にMPとして機能するのはMPHの下位3ビットとMPLの計7ビットであり，MPHの最上位ビットには，MPによるアドレス修飾を許可するメモリ・ポインタ・イネーブル・フラグ（MPE）が割り付けられています。

MPE＝1のとき，オペランド@rで間接指定されたデータ・メモリのバンクとロウ・アドレスは，BANKとm_Rではなく，MPで指定されたアドレスになります（カラム・アドレスは，MPEとは無関係に，rの内容で指定されます）。このとき，MPHの下位3ビットとMPLの最上位ビットの計4ビットがBANKを，MPLの下位3ビットがロウ・アドレスを指します。

備考　μPD17149のMPHの下位3ビットとMPLの最上位ビットは，“0固定”になっており，MPE＝1であっても，必ずバンク0になります（バンク0以外になることを防止しています）。

図8－7　インデクス・レジスタとメモリ・ポインタの構成

アドレス	7AH				7BH				7CH				7FH			
名　称	インデクス・レジスタ（IX） メモリ・ポインタ（MP）												プログラム・ステータス・ワード（PSWORD）の下位4ビット			
シンボル名	IXH MPH				IXM MPL				IXL				PSW			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
フラグ名	MPE															IXE
データ		←(IX)→														
		←(MP)→														
		0	0	0	0											
リセット時の値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図8－8　インデクス・レジスタとメモリ・ポインタによるデータ・メモリ・アドレスの修飾

IXE	MPE	mで指定されるデータ・メモリ・アドレス											@rで指定される間接転送アドレス									
		バンク				ロウ・アドレス			カラム・アドレス				バンク				ロウ・アドレス			カラム・アドレス		
		b_3	b_2	b_1	b_0	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_2	b_1	b_0	b_3	b_2	b_1	b_0
0	0	BANK				m							BANK				m_R			(r)			
0	1	同上											MPH				MPL			(r)			
1	0	BANK				Logical OR			m				BANK				Logical OR			m_R			
		IXH				IXM			IXL				IXH				IXM			(r)			
1	1	設定禁止																					

BANK　：バンク・レジスタ
IX　　：インデクス・レジスタ
　IXE　：インデクス・イネーブル・フラグ
　IXH　：インデクス・レジスタのビット10 - 8
　IXM　：インデクス・レジスタのビット7 - 4
　IXL　：インデクス・レジスタのビット3 - 0
m　　：m_R, m_Cで示されるデータ・メモリ・アドレス
　m_R　：データ・メモリ・ロウ・アドレス
　m_C　：データ・メモリ・カラム・アドレス
MP　　：メモリ・ポインタ
　MPE　：メモリ・ポインタ・イネーブル・フラグ
　MPH　：メモリ・ポインタの上位3ビット
　MPL　：メモリ・ポインタの下位4ビット
r　　：ジェネラル・レジスタ・カラム・アドレス
　RP　：ジェネラル・レジスタ・ポインタ
（×）：×でアドレスされる内容
　　×：rなどのダイレクト・アドレス

表8－1　アドレス修飾される命令群

算術演算	ADD ADDC SUB SUBC	r , m m, #n4
論理演算	AND OR XOR	r , m m, #n4
判断	SKT SKF	m, #n
比較	SKE SKGE SKLT SKNE	m, #n4
転送	LD	r , m
	ST	m , r
	MOV	m, #n4 @r, m m, @r

8.5.3　IXE＝0, MPE＝0のとき（データ・メモリ修飾なし）

図8－8に示したようにデータ・メモリ・アドレスはインデクス・レジスタと，データ・メモリ・ロウ・アドレス・ポインタの影響を受けません。

（1）データ・メモリ操作命令

例1．ジェネラル・レジスタがロウ・アドレス0にあるときの“ADD r, m”の実行

```
R003    MEM     0.03H
M061    MEM     0.61H
        ADD     R003, M061      ；メモリ間加算　(0.03H) ← (0.03H) ＋ (0.61H)
```

この例を実行すると，図8－9に示すようにジェネラル・レジスタR003とデータ・メモリM061の内容を加算し，結果をジェネラル・レジスタR003に格納します。

（2）ジェネラル・レジスタ間接転送（水平間接転送）

例2．ジェネラル・レジスタがロウ・アドレス0にあるときの“MOV @r, m”の実行

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     R005, #8        ；R005 ← 8（@rのカラム・アドレス設定）
        MOV     @R005, M034     ；レジスタ間接転送　(0.38H) ← (0.34H)
```

この例を実行すると図8－9に示すようにデータ・メモリM034の内容がデータ・メモリの38H番地へ転送されます。

“MOV @r, m”命令は，mで指定されるデータ・メモリの内容をmと同じロウ・アドレス上にあり，カラム・アドレスが@rで指定されるアドレスのデータ・メモリへ転送します。

したがって，上記の例では，M034のデータをM034と同じロウ・アドレス（＝3）でカラム・アドレスがR005の内容（＝8）で指定される38Hへ，転送します。

例3．ジェネラル・レジスタがロウ・アドレス0にあるときの“MOV m, @r”の実行

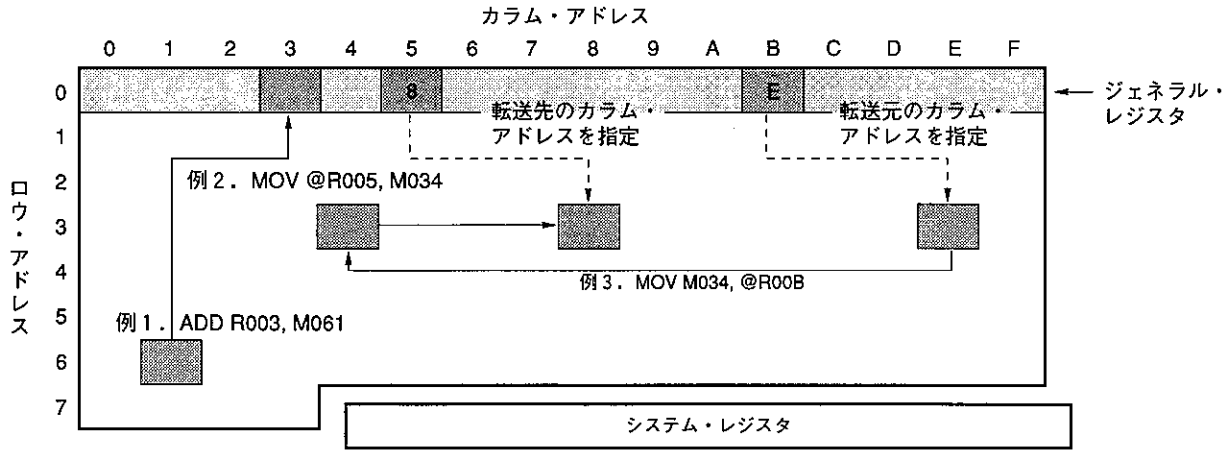
```
R00B    MEM     0.0BH
M034    MEM     0.34H
        MOV     R00B, #0EH      ; R00B ← 0EH（@rのカラム・アドレス設定）
        MOV     M034, @R00B     ; レジスタ間接転送　(0.34H) ← (0.3EH)
```

この例を実行すると，図8－9に示すようにデータ・メモリM034へ，アドレス3EH番地のデータ・メモリの内容を転送します。

“MOV m, @r”命令は，mと同じロウ・アドレス上にあり，カラム・アドレスが@rで指定されるアドレスのデータ・メモリの内容をmで指定されるデータ・メモリへ転送します。

したがって，上記の例では，M034と同じロウ・アドレス（＝3）で，カラム・アドレスがR00Bの内容（＝0EH）で指定される3EHのデータをM034へ転送します。

図8－9　IXE＝0，MPE＝0のときの動作例



例1のアドレス生成

ADD R003, M061

	バンク	ロウ・アドレス	カラム・アドレス
データ・メモリ・アドレス　M	0000	110	0001
ジェネラル・レジスタ・アドレス　R	0000	000	0011

例2のアドレス生成

MOV @R005, M034

	バンク	ロウ・アドレス	カラム・アドレス
データ・メモリ・アドレス　M	0000	011	0100
ジェネラル・レジスタ・アドレス　R	0000	000	0101
間接転送アドレス　@R	0000	011	1000
	Mと同一		Rの内容

8.5.4　IXE = 0, MPE = 1のとき（ななめ間接転送）

図8－8に示したようにジェネラル・レジスタ間接転送命令（MOV @r, mおよびMOV m, @r）を行ったときのみ，@rで指定される間接転送アドレスのバンクとロウ・アドレスがデータ・メモリ・ロウ・アドレス・ポインタの値になります。

例1．ジェネラル・レジスタがロウ・アドレス0のときの“MOV @r, m”の実行

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     MPL, #0110B     ; MP ← 6（@rのロウ・アドレス設定）
        MOV     MPH, #1000B     ; MPE ← 1, バンク ← 0
        MOV     R005, #8        ; R005 ← 8（@rのカラム・アドレス設定）
        MOV     @R005, M034     ; レジスタ間接転送　（0.68H）← （0.34H）
```

この例を実行すると図8－10に示すように，データ・メモリM034の内容が，データ・メモリの68H番地に転送されます。

すなわちMPE = 1のときに“MOV @r, m”命令を実行すると，mで指定されるデータ・メモリの内容をメモリ・ポインタで指定したロウ・アドレスの@rで指定されるカラム・アドレスへ転送します。

このとき@rで指定される間接アドレスは，バンクとロウ・アドレスがデータ・メモリ・ロウ・アドレス・ポインタの値（上記例ではロウ・アドレス6）で，カラム・アドレスがrで指定されるジェネラル・レジスタの内容（上記例では8）になります。

すなわち上記では68Hとなります。

これは8.5.3例2のMPE = 0のときと比べると，@rで指定される間接アドレスのバンクとロウ・アドレスをデータ・メモリ・ロウ・アドレス・ポインタで指定する点が異なります（8.5.3例2では間接アドレスのバンクとロウ・アドレスはmと同じになる）。

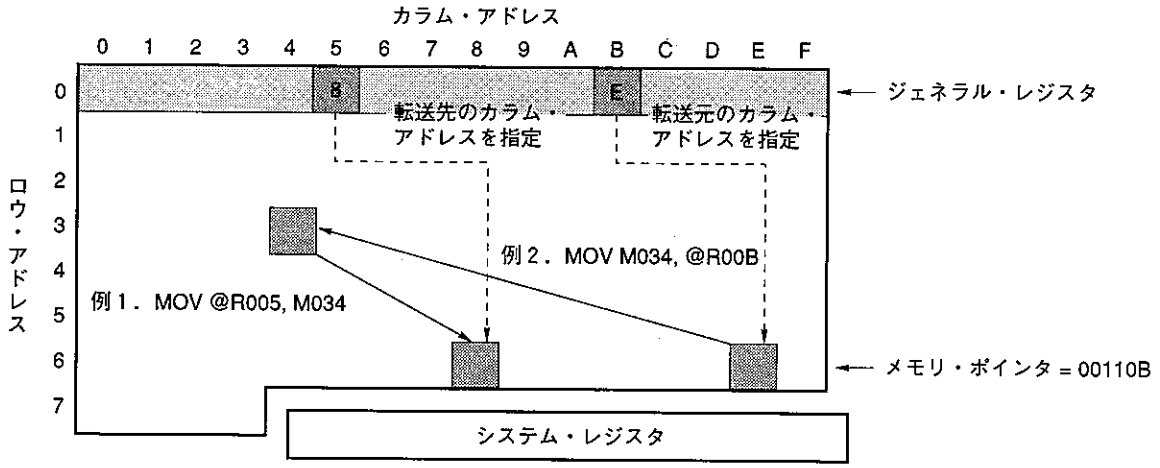
したがって，MPE = 1とすることによりジェネラル・レジスタ間接転送をななめ方向に行うことが可能になります。

例2．ジェネラル・レジスタがロウ・アドレス0のときの“MOV m, @r”の実行

```
R00B    MEM     0.0BH
M034    MEM     0.34H
        MOV     MPL, #0110B     ; MP ← 6（@rのロウ・アドレス設定）
        MOV     MPH, #1000B     ; MPE ← 1, バンク ← 0
        MOV     R00B, #0EH      ; R00B ← 0EH（@rのカラム・アドレス設定）
        MOV     M034, @R00B     ; レジスタ間接転送　(0.34H) ← (0.6EH)
```

この例を実行すると，図8－10に示すようにデータ・メモリM034へ，アドレス6EH番地のデータ・メモリの内容を転送します。

図8－10　IXE＝0, MPE＝1のときの動作例



例1のアドレス生成

MOV @R005, M034

	バンク	ロウ・アドレス	カラム・アドレス
データ・メモリ・アドレス　M	0000	011	0100
ジェネラル・レジスタ・アドレス　R	0000	000	0101
間接転送アドレス　@R	0000	110	1000
	MPの内容		Rの内容

例2のアドレス生成

MOV M034, @R00B

	バンク	ロウ・アドレス	カラム・アドレス
データ・メモリ・アドレス　M	0000	011	0100
ジェネラル・レジスタ・アドレス　R	0000	000	1011
間接転送アドレス　@R	0000	110	1110
	MPの内容		Rの内容

8.5.5　IXE = 1, MPE = 0のとき（インデクス修飾）

図8－8に示したようにデータ・メモリ操作命令を行うと，命令のオペランド“m”で直接指定されたすべてのデータ・メモリのバンクとアドレスが，インデクス・レジスタによりアドレス修飾されます。

また，ジェネラル・レジスタ間接転送命令（MOV @r, mおよびMOV m, @r）を行ったときは，@rで指定される間接転送アドレスのバンクとロウ・アドレスもインデクス・レジスタによりアドレス修飾されます。

アドレス修飾の方法は，データ・メモリ・アドレスとインデクス・レジスタの内容がOR演算され，その演算結果で指定されるデータ・メモリ・アドレス（実アドレスと呼ぶ）に対して命令が実行されます。

以下に例を示します。

例1．ジェネラル・レジスタがロウ・アドレス0のときの“ADD r, m”の実行

```
R003    MEM     0.03H
M061    MEM     0.61H
        MOV     IXL, #0010B             ; IX ← 00000010010B
        MOV     IXM, #0001B             ;
        MOV     IXH, #0000B             ; MPE ← 0
        OR      PSW, #.DF.IXE AND 0FH   ; IXE ← 1
        ADD     R003, M061              ; (0.03H) ← (0.03H) ＋ (0.73H)
```

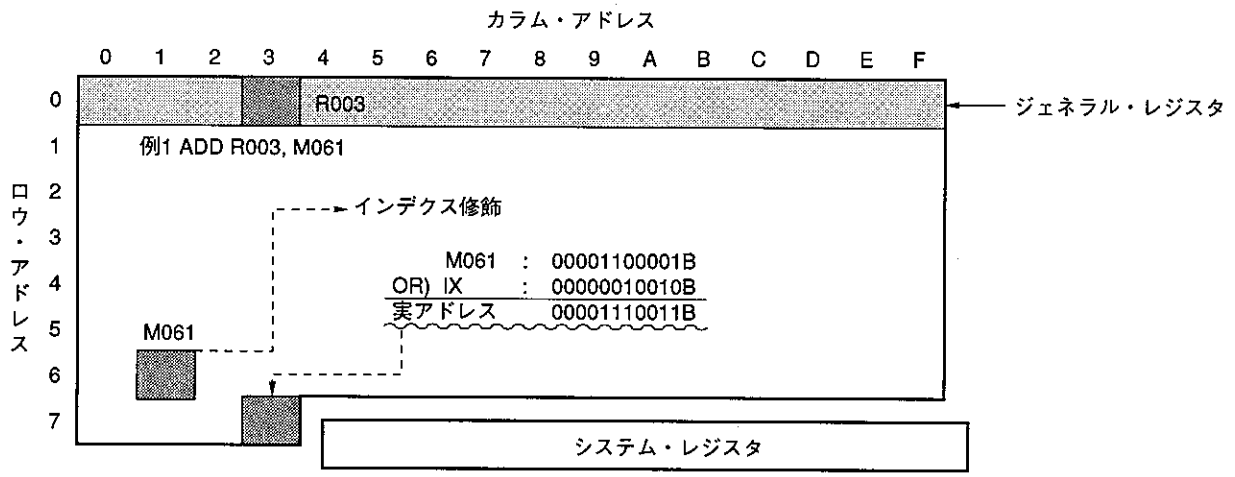
この例を実行すると，図8－11に示すようにアドレスが73H番地のデータ・メモリ（実アドレス）の内容とジェネラル・レジスタR003（アドレスの03H番地）の内容を加算し，結果をジェネラル・レジスタR003へ格納します。

すなわち，“ADD r, m”命令を実行すると“m”で指定されたデータ・メモリ・アドレス（上記では61H番地）がインデクス修飾されます。

修飾の方法はデータ・メモリM061のアドレスである61H番地（2進で00001100001B）と，インデクス・レジスタの値（上記例では00000010010B）をOR演算し，その結果の00001110011Bを実アドレス（73H番地）として，実アドレスに対して命令を実行します。

これはIXE = 0のとき（8.5.3の例）と比べると命令のオペランド“m”で直接指定されるデータ・メモリのアドレスがインデクス・レジスタにより修飾（OR演算）されたことになります。

図8－11　IXE＝1, MPE＝0のときの動作例



例1のアドレス生成

ADD R003, M061

		バンク	ロウ・アドレス	カラム・アドレス
データ・メモリ・アドレス　M		0000	110	0001
ジェネラル・レジスタ・アドレス　R		0000	000	0011
インデクス修飾	M061	0000	110	0001
		BANK	m	
	IX	0000	001	0010
		IXH	IXM	IXL
	実アドレス（OR演算）	0000	111	0011

このアドレスに対して命令が実行される

例2．ジェネラル・レジスタ間接転送（“MOV @r, m”の実行）

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     IXL, #0001B             ;カラム・アドレス←5（4と1のOR）
        MOV     IXM, #0000B             ;ロウ・アドレス←3（3と0のOR）
        MOV     IXH, #0000B             ;MPE ← 0，バンク ←0（0と0のOR）
        OR      PSW, #.DF.IXE AND 0FH   ;IXE ← 1
        MOV     R005, #8                ;R005 ← 8（@rのカラム・アドレス設定）
        MOV     @R005, M034             ;レジスタ間接転送　(0.38H) ← (0.35H)
```

この例を実行すると図8－12に示すように，データ・メモリのアドレス35H番地の内容が，データ・メモリの38H番地に転送されます。

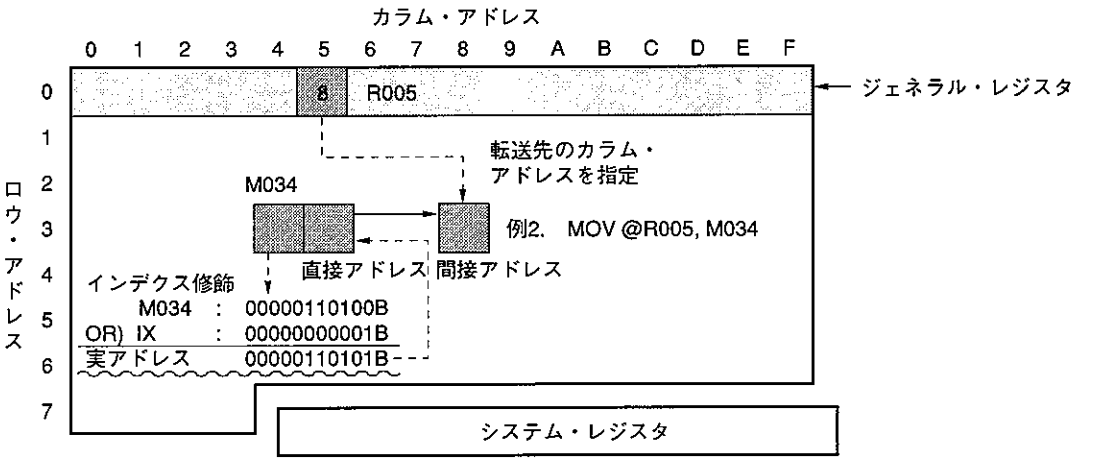
すなわち，IXE＝1のときに“MOV @r, m”命令を実行すると，“m”で指定されるデータ・メモリ・アドレス（直接アドレス）をインデクス・レジスタの内容でアドレス修飾し，“@r”で指定される間接アドレスのバンクとロウ・アドレスもインデクス・レジスタで修飾されます。

“m”で指定される直接アドレスはバンク，ロウ・アドレスおよびカラム・アドレスのすべてが修飾され，@rで指定される間接アドレスは，バンクとロウ・アドレスが修飾されます。

すなわち上記では直接アドレス“m”が35Hとなり，間接アドレス“@r”が38H番地になります。

これは8.5.3例3のIXE＝0のときと比べると，“m”で指定される直接アドレスのバンク，ロウ・アドレスおよびカラム・アドレスがインデクス・レジスタでアドレス修飾され，このアドレス修飾されたデータ・メモリ・アドレスと同一ロウ・アドレスにジェネラル・レジスタ間接転送することになります（8.5.3例3では直接アドレスは修飾されない）。

図8－12　IXE＝1, MPE＝0のときのジェネラル・レジスタ間接転送動作例



例3．00H - 6FHのデータ・メモリの内容を0にクリアする

```
M000    MEM     0.00H
        MOV     IXL, #0                 ; IX ← 0
        MOV     IXM, #0                 ;
        MOV     IXH, #0                 ; MPE ← 0
LOOP：
        OR      PSW, #.DF.IXE AND 0FH   ; IXE ← 1
        MOV     M000, #0                ; IXで指定されたデータ・メモリを0にする。
        INC     IX                      ; IX ← IX＋1
        AND     PSW, #1110B             ; IXE ← 0，IXEは7FH番地のためIXでアド
                                        ; レス修飾しても7FH番地のまま。
        SKE     IXM, #7                 ; ロウ・アドレス7になったか。
        BR      LOOP                    ; 7でなければLOOP（ロウ・アドレスはクリ
                                        ; アしない）
```

例4．配列の処理

図8－13に示すように，1つの要素が8ビットの1次元配列の要素A（N）に

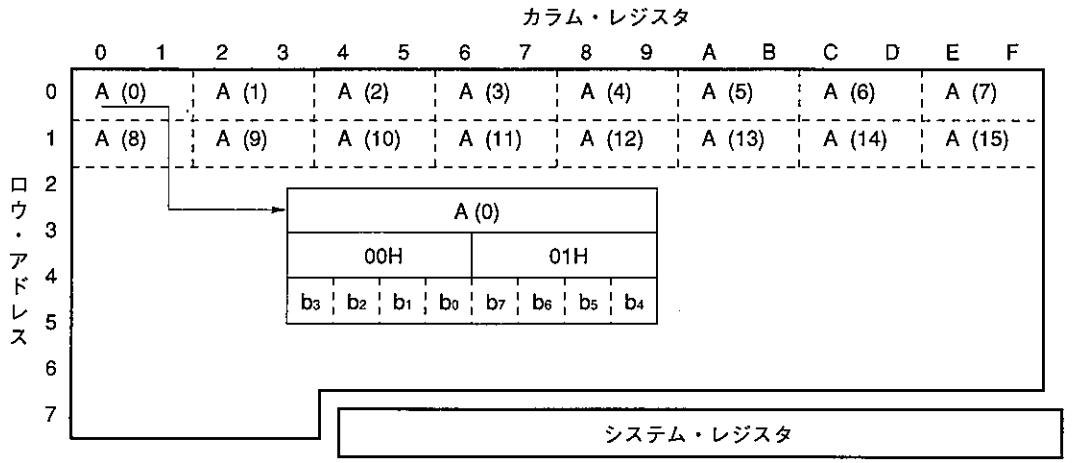
A（N）＝A（N）＋4　（0 ≦ N ≦ 15）

という演算を行うためには以下の命令を実行します。

```
M000    MEM     0.00H
M001    MEM     0.01H
        MOV     IXH, #0
        MOV     IXM, #N SHR 3           ；ロウ・アドレスのオフセットを設定
        MOV     IXL, #N SHL 1 AND 0FH   ；カラム・アドレスのオフセットを設定
        OR      PSW, #.DF.IXE AND 0FH   ；IXE ← 1
        ADD     M000, #4
        ADDC    M001, #0                ；A（N）←A（N）＋4
```

上記の例では，1つの要素が8ビットであるため，インデクス・レジスタにNの値を1ビット左シフトした値（Nを2倍した値）を設定しています。

図8－13　IXE＝1，MPE＝0のときの動作例（配列の処理）



8.6　ジェネラル・レジスタ・ポインタ（RP）

ジェネラル・レジスタのバンクとロウ・アドレスを指定するレジスタです（第7章　ジェネラル・レジスタ（GR）参照）。

8.7　プログラム・ステータス・ワード（PSWORD）

8.7.1　プログラム・ステータス・ワードの構成

図8－14にプログラム・ステータス・ワードの構成を示します。

図8－14　プログラム・ステータス・ワードの構成

アドレス	7EH				7FH			
名　称	(RP)				プログラム・ステータス・ワード (PSWORD)			
記　号	RPL				PSW			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ				BCD	CMP	CY	Z	IXE
リセット時	0				0			

図8－14に示すようにプログラム・ステータス・ワードはシステム・レジスタの7EH番地（RPL）の最下位ビットと7FH番地（PSW）の4ビットの計5ビットで構成されています。

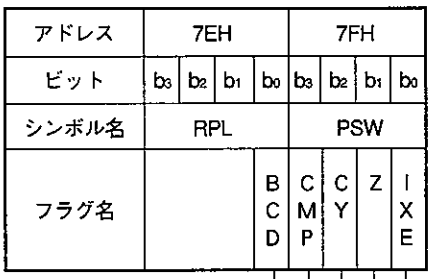
プログラム・ステータス・ワードはさらに1ビットずつ機能が分かれており，それぞれバイナリ・コーデッド・デシマル・フラグ（BCD），コンペア・フラグ（CMP），キャリー・フラグ（CY），ゼロ・フラグ（Z）およびインデクス・イネーブル・フラグ（IXE）から構成されています。

リセット時および割り込みスタック・レジスタへの退避時はすべて0にクリアされます。

8.7.2　プログラム・ステータス・ワードの機能

プログラム・ステータス・ワードの各フラグは，演算および転送命令の条件を設定したり，演算結果の状態を示すために使用します。図8－15にプログラム・ステータス・ワードの機能概要を示します。

図8－15　プログラム・ステータス・ワードの機能概要



アドレス	7EH				7FH			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
シンボル名	RPL				PSW			
フラグ名				BCD	CMP	CY	Z	IXE

フラグ名	PSWORDの機能
IXE	データ・メモリ操作命令実行時のデータ・メモリ・アドレスをインデクス修飾指定するときに使用します。 0：インデクス修飾しません。 1：インデクス修飾します。
Z	算術演算結果が0になったときセットされます。 0：算術演算結果が0以外になったことを示します。 1：算術演算結果が0になったことを示します。
CY	加算命令実行時キャリーが発生したとき，または減算命令実行時ボローが発生したときセットされます。 0：キャリーあるいはボローが発生しなかったことを示します。 1：キャリーあるいはボローが発生したことを示します。
CMP	算術演算の結果をデータ・メモリあるいは汎用レジスタに格納せず，CY，Zフラグのみ変化させたいときに使用します。 0：算術演算の結果を格納します。 1：算術演算の結果を格納しません。
BCD	算術演算の方法を設定します。 0：算術演算を2進4ビット演算で行います。 1：算術演算をBCD演算で行います。

8.7.3　インデクス・イネーブル・フラグ（IXE）

IXEフラグは，データ・メモリのアドレスのインデクス修飾を許可するために使用します。

詳しくは8.5.1　インデクス・レジスタ（IX）を参照してください。

8.7.4　ゼロ・フラグ（Z）とコンペア・フラグ（CMP）

Zフラグは算術演算の結果が0であることを示すフラグであり，CMPフラグは算術演算の結果をデータ・メモリやジェネラル・レジスタへ格納しないように設定するためのフラグです。

Zフラグのセットおよびリセット条件はCMPフラグの状態により表8－2のようになります。

表8－2　ゼロ・フラグ（Z）とコンペア・フラグ（CMP）

条　　件	CMP＝0のとき	CMP＝1のとき
算術演算の結果が“0”になったとき	Z←1	Zは変化しない
算術演算の結果が“0”以外になったとき	Z←0	Z←0

ZフラグおよびCMPフラグはジェネラル・レジスタの内容とデータ・メモリの内容の比較を行うときなどに使用します。Zフラグは算術演算以外，CMPフラグはビット判断以外では変化しません。

12ビット・データの比較の例

```
；M001，M002，M003に格納された12ビットのデータが，456Hに等しいか？
CMP456：
        SET2    CMP, Z
        SUB     M001, #4 ；M001，M002，M003に格納されたデータは壊れない。
        SUB     M002, #5
        SUB     M003, #6
        ；CLR1  CMP
        SKT1    Z        ；CMPはビット判断命令で自動的にクリア
        BR      DIFFER   ；≠456H
        BR      AGREE    ；＝456H
```

8.7.5　キャリー・フラグ（CY）

CYフラグは加算命令および減算命令実行後のキャリーまたはボローの発生を示すフラグです。

CYフラグは，算術演算の結果，キャリーまたはボローが発生するとセット（1）され，発生しなければリセット（0）されます。

また，RORC r命令（rでアドレス指定されるジェネラル・レジスタの内容を右に1ビット分シフトする）を実行したとき，命令を実行する直前のCYフラグの値をジェネラル・レジスタの最上位ビットにシフトし，最下位ビットをCYフラグにシフトします。

CYフラグはキャリーやボローが発生した場合に次の命令をスキップしたいときなどに使用すると便利です。

算術演算および回転処理以外では変化しません。また，CMPフラグの影響を受けません。

8.7.6　バイナリ・コーデッド・デシマル・フラグ（BCD）

BCDフラグはBCD演算を行うためのフラグです。

BCDフラグをセット（1）することにより，すべての算術演算がBCD演算で行われます。リセット（0）すると，2進4ビットで行われます。

論理演算，ビット判断，比較判断，回転処理には影響を与えません。

8.7.7　算術演算時の注意

プログラム・ステータス・ワード（PSWORD）に対して算術演算（加算および減算）を行うときは以下の点に注意が必要です。

すなわち，プログラム・ステータス・ワードに対して算術演算を行うと，算術演算の“結果”が格納されるという点です。

以下に例を示します。

例

```
MOV   PSW,   #0001B
ADD   PSW,   #1111B
```

上記の命令を実行するとキャリーが発生するためPSWのビット2であるCYフラグがセット（1）されるように見えますが，算術演算の結果は0000Bであるため PSWには0000Bが格納されます。

8.8　システム・レジスタ使用時の注意

8.8.1　システム・レジスタの予約語

システム・レジスタはデータ・メモリ上に配置されているため，すべてのデータ・メモリ操作命令を使用できます。このとき，17Kシリーズのアセンブラを使用するうえでは例1に示すように，命令のオペランドには直接データ・メモリ・アドレスを記述できないため，あらかじめデータ・メモリ・アドレスをシンボル定義しておく必要があります。

ここで，システム・レジスタはデータ・メモリでもありますが，通常の汎用データ・メモリと異なり専用の機能を持っているため，アセンブラ上であらかじめ“予約語”としてシンボル定義されています。

システム・レジスタの予約語はアドレス74H - 7FH番地に割り当てられており，図8－2　システム・レジスタの構成に示した記号（AR3, AR2……PSW）で定義されています。

この予約語を使用すれば，例2に示すようにシンボル定義する必要はありません。

予約語については第21章　アセンブラ予約語も参照してください。

```
例1.            MOV     34H, #0101B  ；オペランドにデータ・メモリ・アドレスを34Hや76H
                MOV     76H, #1010B  ；と記述すると，アセンブラでエラーとなる。
        M037    MEM     0.37H        ；汎用データ・メモリは，MEM疑似命令でデータ・メ
                MOV     M037, #0101B ；モリ・アドレスをシンボル定義する必要がある。

    2.          MOV     AR1, #1010B  ；予約語であるAR1（アドレス76H）を使用すれば
                                       ；シンボル定義の必要はない。（予約語AR1は“AR1
                                       ；MEM 0.76H”としてデバイス定義されている。）
```

また，アセンブラ（AS17K）にはフラグ型シンボル操作命令として以下のマクロ命令が組み込まれています。

SETn	：フラグに“1”をセット
CLRn	：フラグを“0”にリセット
SKTn	：フラグがすべて“1”であればスキップ
SKFn	：フラグがすべて“0”であればスキップ
NOTn	：フラグを反転
INITFLG	：フラグをイニシャライズ

したがって，これらの組み込みマクロ命令を使用することにより，以下の例3に示すようにデータ・メモリをフラグとして操作することができます。

プログラム・ステータス・ワードおよびメモリ・ポインタ・イネーブル・フラグはビット単位（フラグ単位）で機能が分けられているため，それぞれのビットに予約語BCD, CMP, CY, Z, IXE, MPEが定義されています。

したがって，このフラグ型予約語を使用すれば例4に示すようにそのまま組み込みマクロ命令を使用できます。

例3．
```
F0003   FLG     0.00.3    ；フラグ型シンボル定義
        SET1    F0003     ；組み込みマクロ
```

マクロ展開
```
OR      .MF.F0003 SHR 4, #.DF.F0003 AND 0FH
                  ；BANK0のアドレス00Hのビット3をセットする。
```

4．
```
        SET1 BCD          ；組み込みマクロ
```

マクロ展開
```
OR      .MF.BCD SHR 4, #.DF.BCD AND 0FH
                  ；BCDフラグをセット
                  ；BCDは“BCD FLG 0.7EH.0”で定義されている。
```

```
        CLR2 Z, CY        ；同一アドレスのフラグ
```

マクロ展開
```
AND     .MF.Z SHR 4, #.DF.（NOT（Z OR CY）AND 0FH）
```

```
        CLR2 Z, BCD       ；異なるアドレスのフラグ
```

マクロ展開
```
AND     .MF.Z SHR 4, #.DF.（NOT Z AND 0FH）
AND     .MF.BCD SHR 4, #.DF.（NOT BCD AND 0FH）
```

8.8.2　“0”に固定されているシステム・レジスタの注意点

システム・レジスタの中であらかじめ“0”に固定されている領域（図8－2　システム・レジスタの構成参照）については，デバイス動作，エミュレータ動作およびアセンブラ動作に注意が必要です。

これを，（1），（2）および（3）に示します。

（1）デバイス動作

“0”に固定されているデータに書き込み命令を行っても，何ら影響を受けません。また読み出し命令を行ったときは常に“0”が読み出されます。

（2）17 Kシリーズのインサーキット・エミュレータ（IE-17 K, IE-17 K-ET）を使用するとき

“0”に固定されているデータに“1”を書き込む命令が実行されると，エラーが発生します。

すなわち以下に示すような命令が実行されると，インサーキット・エミュレータはエラーを発生します。

```
例1．MOV    BANK, #0100B ；0に固定されているビット2に1を書き込む。

  2．MOV    IXL, #1111B    ;
     MOV    IXM, #0111B    ;
     ADD    IXL, #1        ;
     ADDC   IXM, #0        ;
```

ただし，INC ARおよびINC IX命令については，例2のように有効ビットがすべて“1”のときに実行されても，インサーキット・エミュレータはエラーを発生しません。これは，アドレス・レジスタとインデクス・レジスタの有効ビットがすべて“1”のときにINC命令が実行されると有効ビットがすべて“0”になるためです。

また，アドレス・レジスタに限り，上記の例1，例2のように“0”で固定されているビットに“1”を書き込んでも，インサーキット・エミュレータはエラーを発生しません。

（3）17 Kシリーズのアセンブラを使用するとき

“0”に固定されているビットに“1”を書き込む命令があってもエラーは出力されません。すなわち，例1で示した，

```
MOV     BANK, #0100B
```

命令を用いると，アセンブラはエラーを発生せず，インサーキット・エミュレータが命令を実行したときに初めてエラーを発生します。

アセンブラがエラーを発生しない理由は，シンボル（予約語を含む）とデータ・メモリ操作命令の対象となるデータ・メモリ・アドレスとの対応を判断していないためです。

ただし，次の場合にはアセンブラがエラーを発生します。

“BANKn”組み込みマクロ命令で“n”に1以上の値を用いたとき

これは，アセンブラがμPD17145サブシリーズではBANK 0以外の組み込みマクロ命令を使えないと判断しているためです。

〔メ　モ〕

第９章　レジスタ・ファイル（RF）

レジスタ・ファイルは主として周辺ハードウエアの条件設定等を行うためのレジスタです。

９．１　レジスタ・ファイルの構成

９．１．１　レジスタ・ファイルの構成

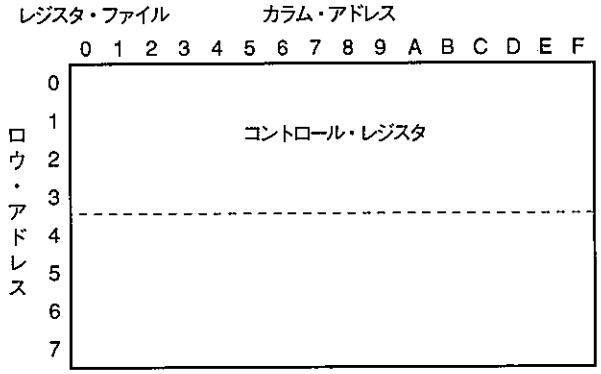
図９－１にレジスタ・ファイルの構成を示します。

図９－１に示すようにレジスタ・ファイルは128ニブル（128×４ビット）で構成されるレジスタです。

レジスタ・ファイルはデータ・メモリと同様に４ビット単位でアドレス（番地）が割り当てられており，ロウ・アドレスが0H - 7Hでカラム・アドレスが0H - 0FHの計128ニブルになります。

また，アドレス00Hから3FH番地まではコントロール・レジスタと呼びます。

図９－１　レジスタ・ファイルの構成



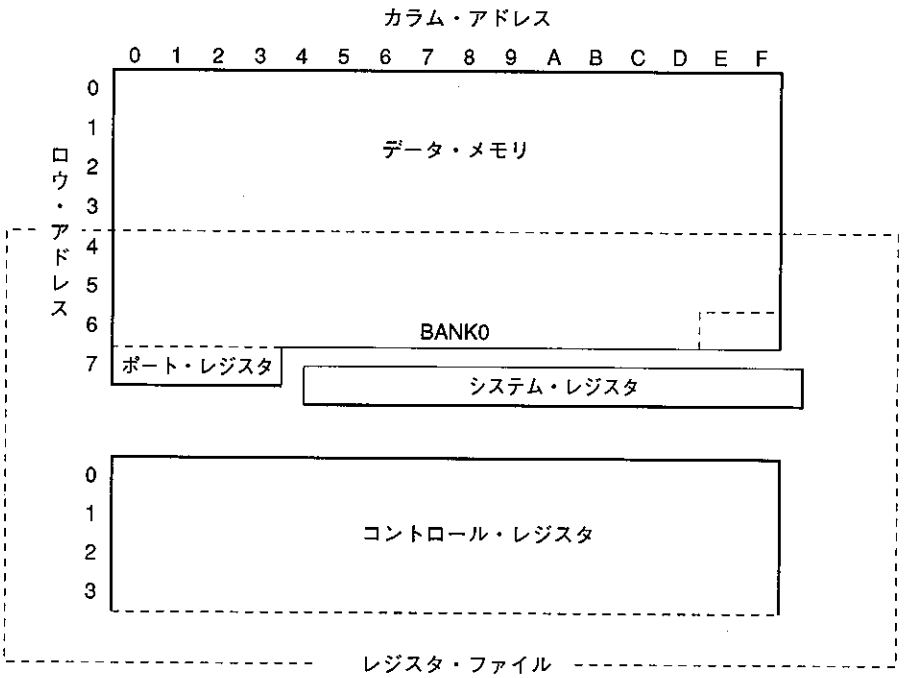
9.1.2　レジスタ・ファイルとデータ・メモリ

図9－2にレジスタ・ファイルとデータ・メモリの関係を示します。

図9－2に示すようにレジスタ・ファイルのアドレス40Hから7FH番地までは，データ・メモリと重なっています。

すなわちプログラム上は，レジスタ・ファイルの40H - 7FH番地に，データ・メモリのアドレス40Hから7FH番地と同じメモリがあるように見えます。

図9－2　レジスタ・ファイルとデータ・メモリの関係



9.2　レジスタ・ファイルの機能

9.2.1　レジスタ・ファイルの機能

レジスタ・ファイルは，PEEK命令またはPOKE命令により，周辺ハードウエアの条件設定をするレジスタ群です。

周辺ハードウエアを制御するレジスタは，00H - 3FH番地に割り付けられており，この部分をコントロール・レジスタと呼びます。

レジスタ・ファイルの40H - 7FH番地には，通常のデータ・メモリが見えています。したがって，この部分はMOV命令だけでなく，PEEK命令，POKE命令による読み書きが可能です。

9.2.2　コントロール・レジスタの機能

コントロール・レジスタにより条件設定を行う周辺ハードウエアを以下に示します。

周辺ハードウエアとコントロール・レジスタの詳細については各周辺ハードウエアの項を参照してください。

・ポート
・8ビット・タイマ・カウンタ（TM0, TM1）
・ベーシック・インターバル・タイマ（BTM）
・A/Dコンバータ
・シリアル・インタフェース（SIO）
・割り込み機能
・スタック・ポインタ（SP）

9.2.3　レジスタ・ファイルの操作命令

レジスタ・ファイルへのデータ書き込みおよびレジスタ・ファイルからのデータの読み込みは，システム・レジスタの中のウインドウ・レジスタ（WR：アドレス78H）を介して行います。

データの書き込みおよびデータ読み出しは以下の専用命令を使用します。

PEEK　WR, rf：WRにrfでアドレス指定されるレジスタ・ファイルのデータを読み出す。
POKE　rf, WR：rfでアドレス指定されるレジスタ・ファイルにWRのデータを書き込む。

以下にレジスタ・ファイルの操作例を示します。

```
例    RF02     MEM  0.82H     ；シンボル定義
      RF1F     MEM  0.9FH     ；レジスタ・ファイルの00H - 3FH番地のシンボル定義は
      RF53     MEM  0.53H     ；BANK0の80H - BFHとして定義する必要があります。
      RF6D     MEM  0.6DH     ；詳しくは9.4　レジスタ・ファイル使用時の注意を参照して
                              ；ください。
      RF70     MEM  1.70H     ；
      RF71     MEM  1.71H     ；
               ；BANK0
  ①           PEEK  WR, RF02 ；
  ②           POKE  RF1F, WR ；
  ③           PEEK  WR, RF53 ；
  ④           POKE  RF6D, WR ；
```

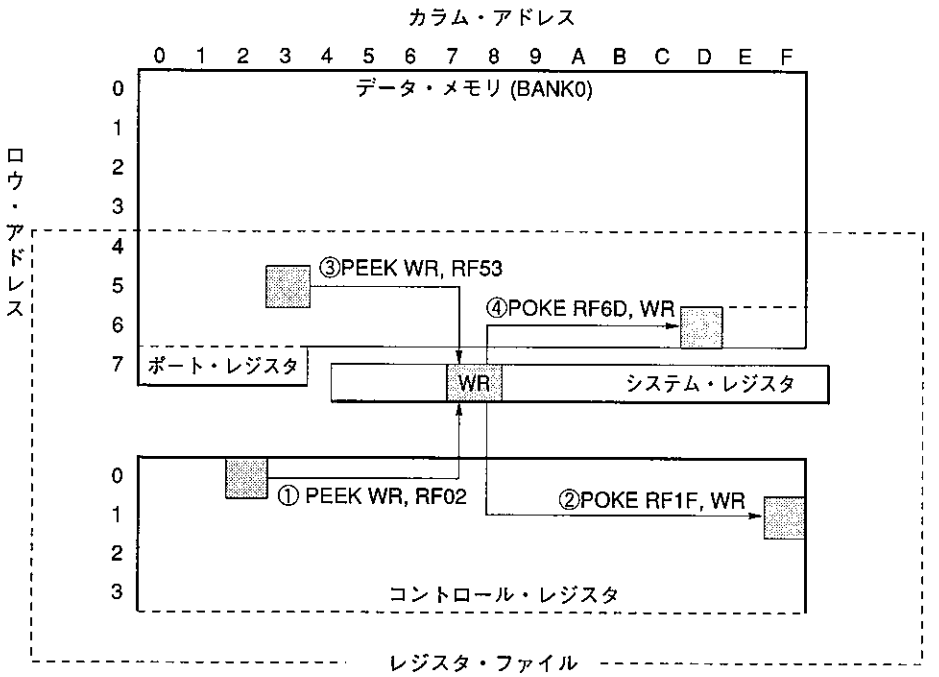
図9－3に動作例を示します。

図9－3に示すように，コントロール・レジスタ（00H - 3FH番地）は“PEEK WR, rf”および“POKE rf, WR”命令が実行されると，“rf”でアドレス指定されるレジスタ・ファイルの内容をウインドウ・レジスタに対してのみ読み出しまたは書き込みを行います。

レジスタ・ファイルの40H - 7FH番地は，データ・メモリと重なっているため，“PEEK WR, rf”および“POKE rf, WR”命令を実行すると，データ・メモリ・アドレス“rf”に対して命令を実行します。

また，レジスタ・ファイルの40H - 7FH番地は通常のメモリ操作命令でも操作できます。

図9－3　PEEK, POKE命令によるレジスタ・ファイルのアクセス



9.3　コントロール・レジスタ

9.3.1　コントロール・レジスタの構成

コントロール・レジスタは，レジスタ・ファイルのアドレス00H - 3FH番地の計64ニブル（64×4ビット）から構成されています。

ただし，そのうち実際に使用しているのは25ニブルです。残りの39ニブルは未使用レジスタで読み出しおよび書き込みは禁止されています。

各コントロール・レジスタは1ニブルずつ属性を持っており，それぞれ読み出し書き込み可能（R/W），読み出し専用（R）の2種類があります。

ただし，読み出し書き込み可能（R/W）なフラグの中には，読み出し時，必ず“0”が読み出されるフラグがありますので，注意してください。

読み出し時，必ず“0”が読み出される読み出し書き込み可能（R/W）なフラグは，次のとおりです。

- WDTRES　（RF：03H, ビット3）
- WDTEN　（RF：03H, ビット0）
- TM0RES　（RF：11H, ビット2）
- TM1RES　（RF：12H, ビット2）
- BTMRES　（RF：13H, ビット2）
- ADCSTRT　（RF：20H, ビット0）

また，1ニブルの中の4ビット・データのうち，“0”に固定されているビットは，読み出したときは常に“0”となり，書き込みを行っても“0”を保持します。

未使用レジスタの39ニブルは，内容を読み出すと不定の値が読み出され，書き込みを行っても何も変化しません。

コントロール・レジスタの構成については図21－2　コントロール・レジスタの構成を参照してください。

9.4　レジスタ・ファイル使用時の注意

9.4.1　コントロール・レジスタ（読み出し専用および未使用レジスタ）操作時の注意

コントロール・レジスタ（レジスタ・ファイルのアドレス00H - 3FH番地）の読み出し専用レジスタ（R）および未使用レジスタを操作するときは以下に示すようにデバイス動作と，17 Kシリーズのアセンブラおよびインサーキット・エミュレータ（IE - 17 K, IE - 17 K - ET）使用時に注意が必要です。

（1）デバイス動作

読み出し専用レジスタに書き込みを行っても何も変化しません。

未使用部分を読み出すと不定な値が読み出され，書き込みを行っても何も変化しません。

（2）アセンブラ使用時

読み出し専用レジスタに書き込みを行う命令にエラーが発生します。

未使用部分を読み出したり，書き込みを行う命令にエラーが発生します。

（3）インサーキット・エミュレータ（IE - 17 K, IE - 17 K - ET）使用時（バッチ処理等で操作したとき）

読み出し専用レジスタに書き込みを行っても何も変化せず，エラーも発生しません。

未使用部分を読み出すと不定な値が読み出され，書き込みを行っても何も変化せず，エラーも発生しません。

9.4.2　レジスタ・ファイルのシンボル定義と予約語

17 Kシリーズのアセンブラを使用する際に，“PEEK WR, rf”命令および“POKE rf, WR”命令のオペランド“rf”に直接数値でレジスタ・ファイル・アドレスを記述するとエラーが発生します。

したがって，例1に示すようにレジスタ・ファイルのアドレスをあらかじめシンボルとして定義しておく必要があります。

例1．エラーになる場合

```
PEEK    WR, 02H          ;
POKE    21H, WR          ;
```

エラーにならない場合

```
RF71    MEM     0.71H    ; シンボル定義
        PEEK    WR, RF71 ;
```

このとき次の点に注意してください。

・コントロール・レジスタを，データ・メモリ・アドレス型としてシンボル定義する場合，BANK0のアドレス80H-BFHとして定義する必要があります。

これは，コントロール・レジスタがウインドウ・レジスタを介して操作する仕組みになっているため，PEEKおよびPOKE以外の命令で操作した場合に，アセンブラでエラーを発生させる必要があるからです。

ただし，データ・メモリと重なっているレジスタ・ファイル（アドレス40H-7FH番地）は，そのままのアドレスでシンボル定義できます。

次にその例を示します。

```
例2．RF71    MEM     0.71H     ；データ・メモリと重なっているレジスタ・ファイル
     RF02    MEM     0.82H     ；コントロール・レジスタ

             PEEK    WR, RF71  ；RF71はデータ・メモリの71H番地になる。
             PEEK    WR, RF02  ；RF02はコントロール・レジスタの02H番地になる。
```

★

また，アセンブラ（AS17K）には，フラグ型シンボル操作命令として以下のマクロ命令があらかじめ組み込まれています。

SETn	：フラグに“1”をセット
CLRn	：フラグを“0”にリセット
SKTn	：フラグがすべて“1”であればスキップ
SKFn	：フラグがすべて“0”であればスキップ
NOTn	：フラグを反転
INITFLG	：フラグをイニシャライズ（4ビット単位のデータ設定）

したがって，これらの組み込みマクロ命令を使用することによりレジスタ・ファイルの内容を1ビット単位で操作することができます。

ここで，コントロール・レジスタは1ビット単位で操作するフラグが多いため，アセンブラであらかじめフラグ型シンボルとして“予約語”が定義されています。

ただし，コントロール・レジスタの中でもスタック・ポインタにはフラグ型の予約語はありません。スタック・ポインタの予約語はデータ・メモリ型として“SP”で定義されています。このため，スタック・ポインタに対して予約語を使ったフラグの操作命令は，使用できません。

〔メ　モ〕

第10章　データ・バッファ（DBF）

データ・バッファは，データ・メモリのBANK0のアドレス0CH - 0FHに割り当てられた４ニブルで構成されています。

この領域はGET, PUT命令によってCPUの周辺ハードウエア（アドレス・レジスタ，シリアル・インタフェース，タイマ０，タイマ１，A/Dコンバータ）とデータの受け渡しを行うデータ格納領域です。また，“MOVT DBF, @AR”命令によりプログラム・メモリ上の定数データをデータ・バッファ上に読み込むことができます。

10.1　データ・バッファの構成

図10－１にデータ・バッファのデータ・メモリ上の配置を示します。

図10－１に示すように，データ・バッファは，データ・メモリのアドレス0CH - 0FHが割り当てられており，４ニブル（４×４ビット）の計16ビットから構成されています。

図10－１　データ・バッファの配置

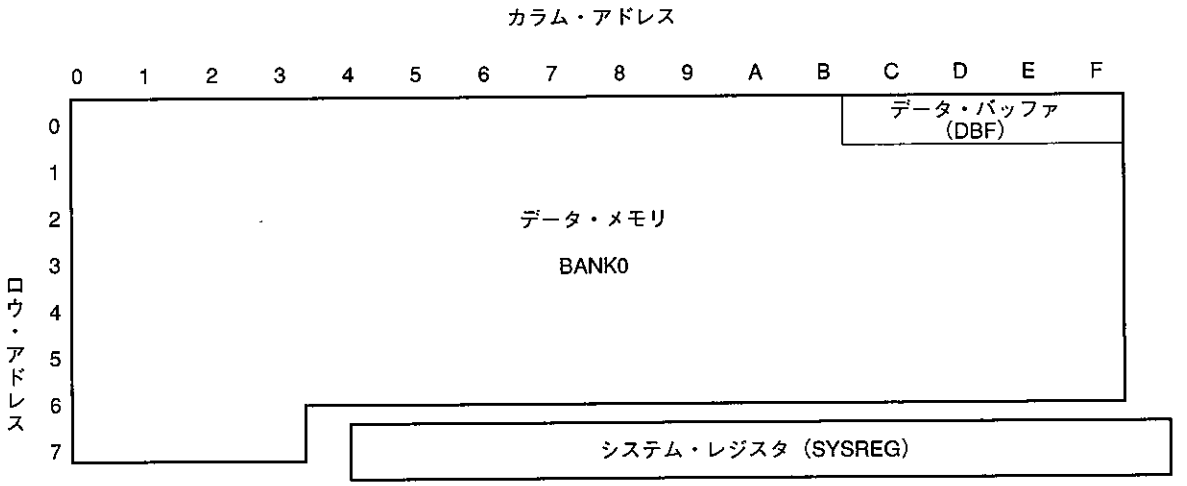


図10－２にデータ・バッファの構成を示します。図10－２に示すようにデータ・バッファはデータ・メモリの0FH番地のビット０をLSBとし，0CH番地のビット３をMSBとする16ビットで構成されています。

図10－2　データ・バッファの構成

データ・メモリ BANK0	アドレス	0CH				0DH				0EH				0FH			
	ビット	b3	b2	b1	b0	b3	b2	b1	b0	b3	b2	b1	b0	b3	b2	b1	b0
データ・バッファ	ビット	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	記　号	DBF3				DBF2				DBF1				DBF0			
	データ	^MSBv				データ											^LSBv

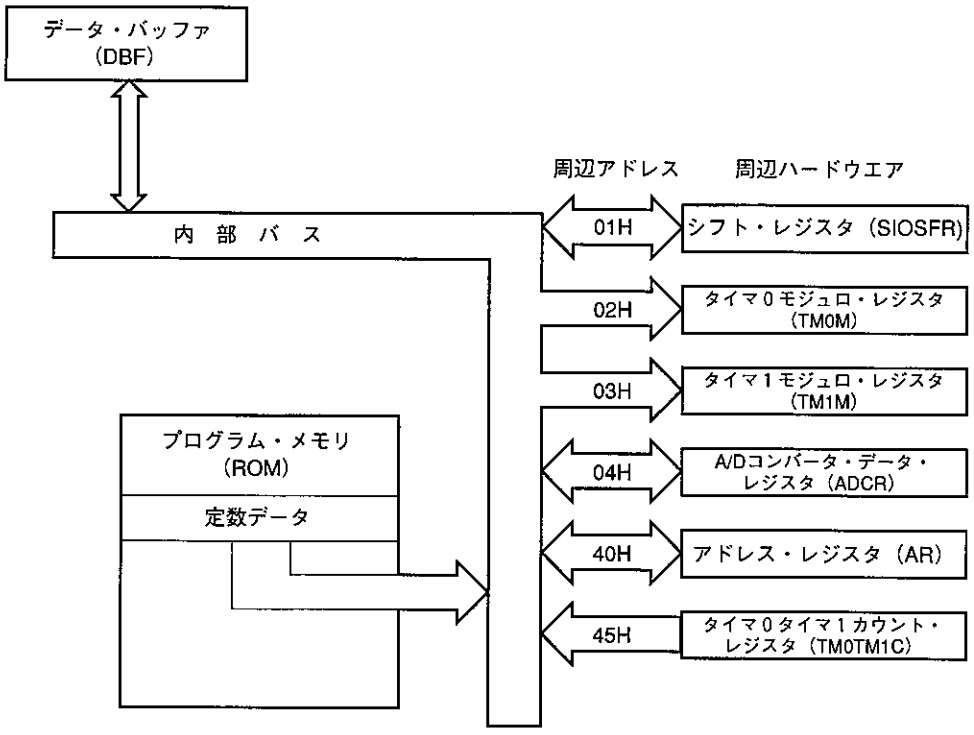
データ・バッファは，データ・メモリ上に配置されているため，すべてのデータ・メモリ操作命令で操作できます。リセット時は，16ビットすべてが不定になります。

10.2　データ・バッファの機能

データ・バッファは，大別して２つの機能があります。

１つは周辺ハードウエアとのデータ転送機能で，もう１つはプログラム・メモリ上の定数データの読み込み（テーブル参照）機能です。図10－３にデータ・バッファと周辺ハードウエアの関係を示します。

図10－3　データ・バッファと周辺ハードウエア



10.2.1　データ・バッファと周辺ハードウエア

表10－1に，データ・バッファを介してデータ転送を行う周辺ハードウエアを示します。

これらの周辺ハードウエアには，それぞれアドレス（周辺アドレスと呼ぶ）が割り付けられています。この周辺アドレスに対して，GETおよびPUT命令を行うことにより，データ・バッファと各周辺ハードウエアとのデータ転送を行うことができます。

命　令	動　　作
GET DBF, p	データ・バッファにpで指定された周辺ハードウエアのデータを読み出す。
PUT p, DBF	pで指定された周辺ハードウエアにデータ・バッファのデータを書き込む。

周辺ハードウエアには書き込み読み出し可能，書き込み専用および読み出し専用ハードウエアがあります。

このとき，書き込み専用や，読み出し専用の周辺ハードウエアに対してそれぞれGET, PUT命令を行うと，以下のようになります。

・書き込み専用周辺ハードウエアに対して読み出し命令（GET）実行時は，不定の値が読み出されます。

・読み出し専用周辺ハードウエアに対して書き込み命令（PUT）実行時は，何ら影響を与えません（NOP命令扱い）。

表10－1　周辺ハードウエア

（1）8ビット単位で入出力を行う周辺ハードウエア

周辺アドレス	名　称	周辺ハードウエア	データ方向		実効ビット長
			PUT	GET	
01H	SIOSFR	SIOシフト・レジスタ	○	○	8ビット
02H	TM0M	タイマ0モジュロ・レジスタ	○	×	8ビット
03H	TM1M	タイマ1モジュロ・レジスタ	○	×	8ビット
04H	ADCR	A/Dコンバータ・データ・レジスタ	○	○	8ビット

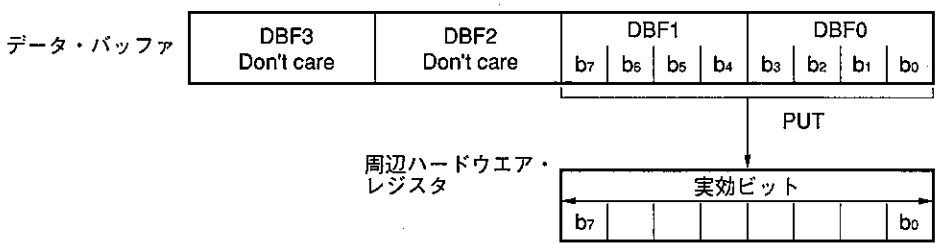
（2）16ビット単位で入出力を行う周辺ハードウエア

周辺アドレス	名　称	周辺ハードウエア	データ方向		実効ビット長
			PUT	GET	
40H	AR	アドレス・レジスタ	○	○	10ビット（μPD17145） 11ビット（μPD17147） 12ビット（μPD17149, μPD17P149）
45H	TM0TM1C	タイマ0タイマ1カウント・レジスタ	×	○	16ビット

10.2.2　周辺ハードウエアとのデータ転送

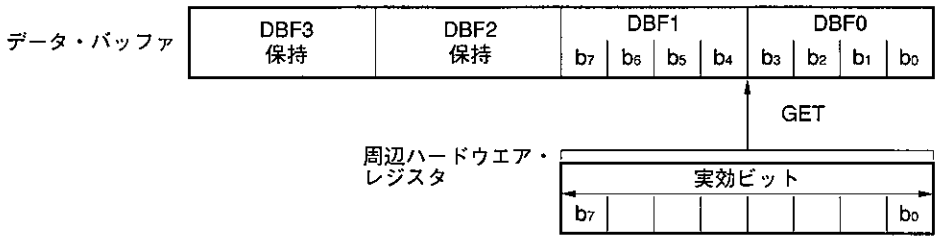
データ・バッファと各周辺ハードウエアとのデータ転送時は8ビットまたは16ビット単位で行います。このとき，PUTおよびGET命令は，データ・ビットが16ビットであっても1命令サイクルで実行できます。

例1．PUT命令時（周辺ハードウエアの実効ビットがビット7-ビット0の8ビットであるとき）



8ビット・データを書き込むときはデータ・バッファの上位8ビット（DBF3, DBF2）はDon't care（どんな値であってもよい）となります。

例2．GET命令時（周辺ハードウエアの実効ビットがビット7-ビット0の8ビットであるとき）



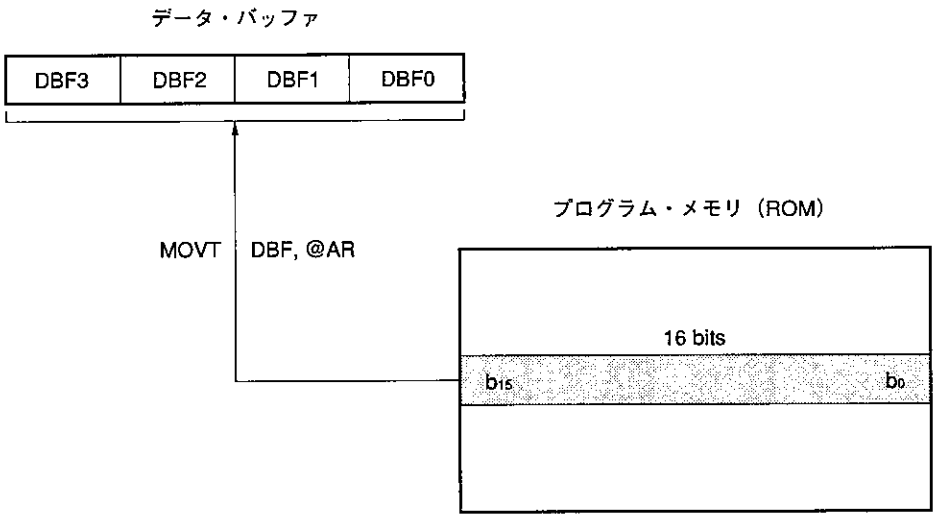
8ビット・データ読み込み時にはデータ・バッファの上位8ビット（DBF3, DBF2）の値は変化しません。

10.2.3　テーブル参照

MOVT命令を用いることにより，プログラム・メモリ（ROM）上の定数データを，データ・バッファ上に読み込むことができます。

以下にMOVT命令について説明します。

MOVT DBF, @AR　　；アドレス・レジスタ（AR）の内容によって指定されるプログラム・メモリの内容を，データ・バッファ（DBF）に読み出します。



〔メ　モ〕

第11章　ALUブロック

ALUは4ビット・データの算術演算，論理演算，ビット判断および回転処理を行います。

11.1　ALUブロックの構成

図11－1にALUブロックの構成を示します。

図11－1に示すようにALUブロックは4ビットのデータ処理を行うALU本体と，ALUの周辺回路である一時記憶用レジスタA，Bと，ALUの状態を制御するステータス・フリップフロップと，BCD演算使用時の10進補正回路から構成されています。

ステータス・フリップフロップは図11－1に示すように，ゼロ・フラグ用FF，キャリー・フラグ用FF，コンペア・フラグ用FFおよびBCDフラグ用FFから構成されています。

ステータス・フリップフロップはシステム・レジスタの中のプログラム・ステータス・ワード（PSWORD：アドレス7EH，7FH）の各フラグであるゼロ・フラグ（Z），キャリー・フラグ（CY），コンペア・フラグ（CMP）およびBCDフラグ（BCD）と1対1に対応しています。

11.2　ALUブロックの機能

ALUはプログラムに書かれた命令により，算術演算，論理演算，ビット判断，比較判断および回転処理を行います。表11－1に各演算，判断，および回転処理命令の一覧を示します。

表11－1に示した各命令を実行することにより4ビット単位の演算，判断および回転処理または1桁の10進算術演算が1命令で実行できます。

11.2.1　ALUの機能

算術演算には加算と減算があります。算術演算はジェネラル・レジスタの内容とデータ・メモリの内容との演算，またはデータ・メモリの内容とイミーディエト・データとの演算が行えます。また，算術演算は2進数による4ビットの演算と10進数による1桁の演算（BCD演算）が可能です。

論理演算には論理積（AND），論理和（OR）および排他的論理和（XOR）があります。論理演算は，ジェネラル・レジスタの内容とデータ・メモリの内容との演算，またはデータ・メモリの内容とイミーディエト・データとの演算が行えます。

ビット判断は，データ・メモリの4ビット・データのうち“0”であるビットまたは“1”であるビットの判断を行います。

比較判断はデータ・メモリの内容とイミーディエト・データとの比較を行い，“等しい”，“等しくない”，“以上”および“未満”の判断を行います。

回転処理はジェネラル・レジスタの4ビット・データを下位ビットの方向へ1ビット，シフトします（右へ回転する）。

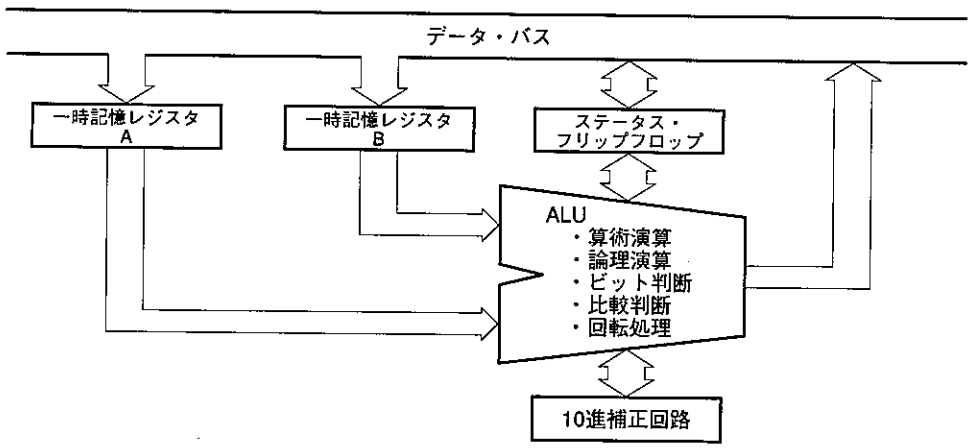
表11－1　ALU処理命令一覧（1/2）

ALU機能		命　令	動　　作	説　　明
算術演算	加算	ADD r, m	(r) ← (r)＋(m)	ジェネラル・レジスタとデータ・メモリの内容を加算。 結果をジェネラル・レジスタへ格納。
		ADD m, #n4	(m) ← (m)＋n4	データ・メモリとイミーディエト・データの内容を加算。結果をデータ・メモリへ格納。
		ADDC r, m	(r) ← (r)＋(m)＋CY	ジェネラル・レジスタとデータ・メモリの内容をCYフラグとともに加算。 結果をジェネラル・レジスタへ格納。
		ADDC m, #n4	(m) ← (m)＋n4＋CY	データ・メモリとイミーディエト・データの内容をCYフラグとともに加算。 結果をデータ・メモリへ格納。
	減算	SUB r, m	(r) ← (r) － (m)	ジェネラル・レジスタの内容からデータ・メモリの内容を減算。 結果をジェネラル・レジスタへ格納。
		SUB m, #n4	(m) ← (m) － n4	データ・メモリの内容からイミーディエト・データを減算。 結果をデータ・メモリへ格納。
		SUBC r, m	(r) ← (r) － (m) － CY	ジェネラル・レジスタの内容からデータ・メモリの内容とCYフラグを減算。 結果をジェネラル・レジスタへ格納。
		SUBC m, #n4	(m) ← (m) － n4 － CY	データ・メモリの内容からイミーディエト・データとCYフラグを減算。 結果をデータ・メモリへ格納。
論理演算	論理和	OR r, m	(r) ← (r) ∨ (m)	ジェネラル・レジスタとデータ・メモリの内容をOR。 結果をジェネラル・レジスタへ格納。
		OR m, #n4	(m) ← (m) ∨ n4	データ・メモリとイミーディエト・データの内容をOR。結果をデータ・メモリへ格納。
	論理積	AND r, m	(r) ← (r) ∧ (m)	ジェネラル・レジスタとデータ・メモリの内容をAND。 結果をジェネラル・レジスタへ格納。
		AND m, #n4	(m) ← (m) ∧ n4	データ・メモリとイミーディエト・データの内容をAND。 結果をデータ・メモリへ格納。
	排他的論理和	XOR r, m	(r) ← (r) ∀ (m)	ジェネラル・レジスタとデータ・メモリの内容をXOR。 結果をジェネラル・レジスタへ格納。
		XOR m, #n4	(m) ← (m) ∀ n4	データ・メモリとイミーディエト・データの内容をXOR。 結果をデータ・メモリへ格納。
ビット判断	True	SKT m, #n	CMP ← 0, if(m) ∧n = n, then skip	データ・メモリの内容のうち、nで指定されたビットがすべてTrue（1）ならスキップ。結果は格納されない。
	False	SKF m, #n	CMP ← 0, if(m) ∧n = 0, then skip	データ・メモリの内容のうち、nで指定されたビットがすべてFalse（0）ならスキップ。結果は格納されない。
比較判断	等しい	SKE m, #n4	(m) － n4, skip if zero	データ・メモリの内容がイミーディエト・データと等しいときスキップ。 結果は格納されない。
	等しくない	SKNE m, #n4	(m) － n4, skip if not zero	データ・メモリの内容がイミーディエト・データと等しくないときスキップ。 結果は格納されない。
	以上	SKGE m, #n4	(m) － n4, skip if not borrow	データ・メモリの内容がイミーディエト・データより以上のときスキップ。 結果は格納されない。
	未満	SKLT m, #n4	(m) － n4, skip if borrow	データ・メモリの内容がイミーディエト・データより未満のときスキップ。 結果は格納されない。
回転	右回転	RORC r	┌→ CY → $(r)_{b3}$ → $(r)_{b2}$ → $(r)_{b1}$ → $(r)_{b0}$ ─┐	ジェネラル・レジスタの内容をCYフラグとともに右へ回転。 結果をジェネラル・レジスタへ格納。

表11－1　ALU処理命令一覧（2/2）

ALU機能	プログラム・ステータス・ワード（PSWORD）による動作のちがい					
算術演算	BCDフラグの値	CMPフラグの値	演算動作	CYフラグ	Zフラグ	IXE＝1による修飾
	0	0	2進演算 結果を格納する	キャリー ボロー 発生で セット， 発生しなければ リセット	演算結果0000Bでセット 0000B以外はリセット	あり
	0	1	2進演算 結果を格納しない		演算結果0000Bで状態保持 0000B以外はリセット	
	1	0	BCD演算 結果を格納する		演算結果0000Bでセット 0000B以外はリセット	
	1	1	BCD演算 結果を格納しない		演算結果0000Bで状態保持 0000B以外はリセット	
論理演算	Don't care （保持）	Don't care （保持）	変わらない	Don't care （保持）	Don't care （保持）	あり
ビット判断	Don't care （保持）	リセット される	変わらない	Don't care （保持）	Don't care （保持）	あり
比較判断	Don't care （保持）	Don't care （保持）	変わらない	Don't care （保持）	Don't care （保持）	あり
回転	Don't care （保持）	Don't care （保持）	変わらない	ジェネラル・ レジスタの b_0の値	Don't care （保持）	あり

図11－1　ALUブロックの構成



アドレス	7EH	7FH			
名　称	プログラム・ステータス・ワード（PSWORD）				
ビット	b_0	b_3	b_2	b_1	b_0
フラグ	BCD	CMP	CY	Z	IXE

ステータス・フリップフロップ

BCDフラグ用FF	CMPフラグ用FF	CYフラグ用FF	Zフラグ用FF

機　能　の　概　要
算術演算結果が0であることを示す
算術演算時のキャリーまたはボローを格納
算術演算結果を格納するかを指定
算術演算時に10進補正を行うかを指定

11.2.2　一時記憶レジスタAおよびBの機能

一時記憶レジスタAおよびBは4ビット・データを一度に処理するために必要なレジスタであり，処理されるデータと，処理するデータを一時的に蓄えておくレジスタです。

11.2.3　ステータス・フリップフロップの機能

ステータス・フリップフロップはALUの動作制御および，処理されたデータの状態を格納するフリップフロップです。ステータス・フリップフロップはシステム・レジスタのプログラム・ステータス・ワード（PSWORD）の各フラグと1対1に対応しているため，システム・レジスタを操作すれば，ステータス・フリップフロップも同時に操作されます。次にプログラム・ステータス・ワードの各フラグについて説明します。

（1）Zフラグ

算術演算の結果が0000Bになるとセット（1）され，0000B以外になるとリセット（0）されます。

ただし，CMPフラグの状態により次のようにセット（1）される条件が異なります。

（i）CMPフラグ＝0のとき

演算結果が0000Bであればセット（1）され0000B以外であればリセット（0）されます。

（ii）CMPフラグ＝1のとき

演算結果が0000Bであれば以前の状態を保持し，0000B以外であればリセット（0）されます。

算術演算以外では変化しません。

（2）CYフラグ

算術演算の結果，キャリーまたはボローが発生するとセット（1）され，発生しなければリセット（0）されます。

算術演算がキャリーまたはボローとともに演算を行う場合はCYフラグの内容を最下位ビットに演算します。

回転処理（RORC命令）を行うときは，そのときのCYフラグの内容をジェネラル・レジスタの最上位ビット（b_3）とし，ジェネラル・レジスタの最下位ビットの内容がCYフラグの内容になります。

算術演算および回転処理以外では変化しません。

（3）CMPフラグ

CMPフラグがセット（1）されているときに実行された算術演算は，結果がジェネラル・レジスタおよびデータ・メモリに格納されません。

ビット判断命令を実行するとCMPフラグはリセット（0）されます。

比較判断，論理演算，回転処理には影響を与えません。

（4）BCDフラグ

BCDフラグがセット（1）されているときは，すべての算術演算が10進補正されます。

リセット（0）されているときは2進4ビットで行われます。

論理演算，ビット判断，比較判断，回転処理には影響を与えません。

これらのフラグは，プログラム・ステータス・ワード（PSWORD）を直接操作することにより値を変化させることも可能です。このとき，変化したフラグに対応するステータス・フリップフロップも同様に変化します。

11.2.4　2進4ビット演算

BCDフラグが0のとき，算術演算は，2進数による4ビット単位の演算を行います。

11.2.5　BCD演算

BCDフラグが1のとき算術演算は，2進4ビットで行った演算に対して10進補正を行います。2進4ビット演算結果とBCD演算結果の違いを表11－2に示します。10進補正を行った場合に加算結果が20以上になったとき，あるいは10進補正を行った場合に減算結果が－10～＋9以外になったときにはデータ・メモリに1010B（0AH）以上のデータが格納されます（表11－2の網掛け部分）。

表11－2　2進4ビット演算結果とBCD演算結果

演算結果	2進4ビット加算		BCD加算	
	CY	演算結果	CY	演算結果
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	0000
11	0	1011	1	0001
12	0	1100	1	0010
13	0	1101	1	0011
14	0	1110	1	0100
15	0	1111	1	0101
16	1	0000	1	0110
17	1	0001	1	0111
18	1	0010	1	1000
19	1	0011	1	1001
20	1	0100	1	1110
21	1	0101	1	1111
22	1	0110	1	1100
23	1	0111	1	1101
24	1	1000	1	1110
25	1	1001	1	1111
26	1	1010	1	1100
27	1	1011	1	1101
28	1	1100	1	1010
29	1	1101	1	1011
30	1	1110	1	1100
31	1	1111	1	1101

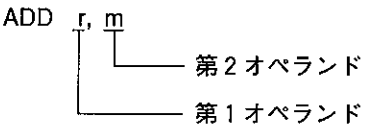
演算結果	2進4ビット減算		BCD減算	
	CY	演算結果	CY	演算結果
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	1100
11	0	1011	1	1101
12	0	1100	1	1110
13	0	1101	1	1111
14	0	1110	1	1100
15	0	1111	1	1101
−16	1	0000	1	1110
−15	1	0001	1	1111
−14	1	0010	1	1100
−13	1	0011	1	1101
−12	1	0100	1	1110
−11	1	0101	1	1111
−10	1	0110	1	0000
− 9	1	0111	1	0001
− 8	1	1000	1	0010
− 7	1	1001	1	0011
− 6	1	1010	1	0100
− 5	1	1011	1	0101
− 4	1	1100	1	0110
− 3	1	1101	1	0111
− 2	1	1110	1	1000
− 1	1	1111	1	1001

11.2.6　ALUブロック処理手順

プログラム上で算術演算命令，論理演算命令，ビット判断命令，比較判断命令および回路処理命令が実行されると，演算，判断または処理されるデータおよび処理するデータがそれぞれ一時記憶レジスタAおよびBに格納されます。

処理されるデータとは，各命令の第１オペランドでアドレス指定されるジェネラル・レジスタの内容またはデータ・メモリの内容であり，４ビットのデータです。処理するデータとは各命令の第２オペランドでアドレス指定されるデータ・メモリの内容または第２オペランドで直接指定されるイミーディエト・データで４ビットのデータです。

たとえば



命令において処理されるデータはrでアドレス指定されるジェネラル・レジスタの内容であり，処理するデータはmでアドレス指定されるデータ・メモリの内容となります。また

```
ADD m, #n4
```

命令は，処理されるデータはmでアドレス指定されるデータ・メモリの内容であり，処理するデータは#n4で指定されるイミーディエト・データになります。また回転処理命令である

```
RORC r
```

命令は，処理する方法が決まっているため処理されるデータのみ必要となり，rでアドレス指定されるジェネラル・レジスタの内容になります。

次に，一時記憶レジスタAおよびBに格納されたデータは，各命令に従いALUで算術演算，論理演算，ビット判断，比較判断および回転処理を実行します。実行された命令が算術演算，論理演算および回転処理のときは，ALUで処理されたデータを，命令の第１オペランドで指定されるジェネラル・レジスタまたはデータ・メモリに格納して動作を終了します。また，実行された命令がビット判断および比較判断であるときは，ALUで処理された結果によりプログラム上の次の命令をスキップ（次の命令をNOP命令として実行）して動作を終了します。

ALUブロック動作については次の点に注意が必要です。

（1）算術演算は，プログラム・ステータス・ワードのCMPフラグおよびBCDフラグの影響を受ける。

（2）論理演算は，プログラム・ステータス・ワードのCMPフラグおよびBCDフラグの影響は受けない。またZフラグ，CYフラグには影響を与えない。

（3）ビット判断はプログラム・ステータス・ワードのCMPフラグをリセットする。

（4）算術演算，論理演算，ビット判断，比較判断および回転処理は，プログラム・ステータス・ワードのIXEフラグがセット（1）されていると，インデクス・レジスタによる修飾を受ける。

11.3　算術演算（2進4ビット加減算およびBCD加減算）

表11－3に示すように，算術演算は，加算と減算に大別され，さらにキャリーとともに加算およびボローとともに減算とに分けられます。算術演算命令はこの4種類に分けられ，それぞれ，ADD, ADDC, SUB, SUBC命令を使用します。

ADD, ADDC, SUB, SUBC命令は，さらにジェネラル・レジスタとデータ・メモリの加減算およびデータ・メモリとイミーディエト・データの加減算に分けられます。これは各命令のオペランドに記述する値により決定されます。すなわちオペランドが“r, m”であればジェネラル・レジスタとデータ・メモリの加減算になり“m, #n4”であればデータ・メモリとイミーディエト・データの加減算になります。

算術演算命令はステータス・フリップフロップすなわち，システム・レジスタのプログラム・ステータス・ワード（PSWORD）の影響を受けます。プログラム・ステータス・ワード（PSWORD）のBCDフラグにより2進4ビット演算およびBCD演算を行い，CMPフラグにより，演算結果をどこにも格納しないことができます。

11.3.1 - 11.3.4に各算術演算命令とプログラム・ステータス・ワード（PSWORD）について説明します。

表11－3　算術演算の種類

算術演算	加算	キャリーは無視 ADD	ジェネラル・レジスタとデータ・メモリ	ADD r, m
			データ・メモリとイミーディエト・データ	ADD m, #n4
		キャリーとともに加算 ADDC	ジェネラル・レジスタとデータ・メモリ	ADDC r, m
			データ・メモリとイミーディエト・データ	ADDC m, #n4
	減算	ボローは無視 SUB	ジェネラル・レジスタとデータ・メモリ	SUB r, m
			データ・メモリとイミーディエト・データ	SUB m, #n4
		ボローとともに減算 SUBC	ジェネラル・レジスタとデータ・メモリ	SUBC r, m
			データ・メモリとイミーディエト・データ	SUBC m, #n4

11.3.1　CMPフラグ＝0，BCDフラグ＝0のときの加減算

2進4ビットの加減算を行い，結果をジェネラル・レジスタまたはデータ・メモリに格納します。

CYフラグは演算結果が1111Bを越えたとき（キャリーの発生）と，0000B未満（ボローの発生）になるとセット（1）され，それ以外ではリセット（0）されます。

演算結果が0000Bになるとキャリーおよびボローの発生に関係なくZフラグをセット（1）し，0000Bでなければリセット（0）します。

11.3.2　CMPフラグ＝1，BCDフラグ＝0のときの加減算

2進4ビットの加減算を行います。

ただしCMPフラグがセット（1）されているため，演算結果がジェネラル・レジスタまたはデータ・メモリに格納されません。

演算結果によりキャリーまたはボローが発生するとCYフラグがセット（1）され，発生しなければリセット（0）されます。

Zフラグは，演算結果が0000Bであれば以前の状態を保持し，0000Bでなければリセット（0）されます。

11.3.3　CMPフラグ＝0，BCDフラグ＝1のときの加減算

BCD演算を行います。

演算結果は，ジェネラル・レジスタまたはデータ・メモリに格納されます。CYフラグは演算結果が1001B（9D）を越えるか，0000B（0D）未満になるとセット（1）され，0000B（0D）～1001B（9D）であればリセット（0）されます。

Zフラグは，演算結果が0000B（0D）になるとセット（1）され，0000B（0D）以外になるとリセット（0）されます。

BCD演算は，一度2進で演算された結果を10進補正回路で10進に変換する方法を用いています。この2進-10進変換は11.2.5　BCD演算の表11－2を参照してください。

したがって，BCD演算を正しく実行するためには，次のことに注意してください。

（1）加算の結果が0D～19Dであること

（2）減算の結果が0D～9Dまたは－10D～－1Dであること

0D～19Dとは，CYフラグを考慮した値であり，2進4ビットで表すと

0,0000B～1,0011Bのことです。

$\widetilde{CY}$　　$\widetilde{CY}$

－10D～－1Dとは同様に

1,0110B～1,1111Bのことです

$\widetilde{CY}$　　$\widetilde{CY}$

上記（1），（2）以外でBCD演算を行うとCYフラグがセット（1）され，演算結果として1010B（0AH）以上のデータが出力されます。

11.3.4　CMPフラグ＝1，BCDフラグ＝1のときの加減算

BCD演算を行います。

演算結果はジェネラル・レジスタまたはデータ・メモリへ格納されません。

すなわち，CMPフラグ＝1のときと，BCDフラグ＝1のときの動作を同時に行います。

```
例  MOV    RPL,    #0001B；BCDフラグをセット（1）
    MOV    PSW,    #1010B；CMPフラグとZフラグをセット（1），CYフラグをリセット（0）
    SUB    M1,     #0001B；①
    SUBC   M2,     #0010B；②
    SUBC   M3,     #0011B；③
```

このとき，①②③によりM3, M2, M1の12ビットの内容とイミーディエト・データの321とを，10進数で比較することが可能となります。

11.3.5　算術演算使用時の注意

プログラム・ステータス・ワード（PSWORD）に対して算術演算を行うときは，プログラム・ステータス・ワードには，算術演算の結果が格納される，という点に注意する必要があります。

すなわちプログラム・ステータス・ワードの中のCYフラグおよびZフラグは，通常，算術演算結果によりセット／リセットされますが，プログラム・ステータス・ワード自身に算術演算が行われると，算術演算結果が格納されてしまい，キャリー，ボローおよびゼロの判定ができないことになります。

ただし，CMPフラグがセット（1）されているときは，算術演算結果が格納されないため，CYフラグ，Zフラグは通常どおりセット／リセットされます。

11.4　論理演算

表11－4に示すように，論理演算は論理和（OR），論理積（AND）および排他的論理和（XOR）が使用できます。

論理演算命令はこの3種類に分けられ，それぞれOR, ANDおよびXOR命令を使用します。

OR, AND, XOR命令は，さらにジェネラル・レジスタとデータ・メモリの論理演算およびデータ・メモリとイミーディエト・データの論理演算に分けられます。これは算術演算と同様に命令のオペランドに記述された値“r, m”もしくは“m, #n4”により決定されます。

論理演算は，プログラム・ステータス・ワード（PSWORD）のBCDフラグおよびCMPフラグの影響は受けません。またCYフラグおよびZフラグには何の影響も与えません。ただし，インデクス・イネーブル・フラグ（IXEフラグ）がセット（1）されているときは，インデクス・レジスタにより修飾されます。

表11－4　論理演算

論理演算	論理和 OR	ジェネラル・レジスタとデータ・メモリ	OR r, m
		データ・メモリとイミーディエト・データ	OR m, #n4
	論理積 AND	ジェネラル・レジスタとデータ・メモリ	AND r, m
		データ・メモリとイミーディエト・データ	AND m, #n4
	排他的論理和 XOR	ジェネラル・レジスタとデータ・メモリ	XOR r, m
		データ・メモリとイミーディエト・データ	XOR m, #n4

表11－5　論理演算の真理値表

論理積 C = A AND B			論理和 C = A OR B			排他的論理和 C = A XOR B		
A	B	C	A	B	C	A	B	C
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

11.5　ビット判断

表11－6に示すように，ビット判断はTrueビット（1）判断およびFalseビット（0）判断に分けられます。Trueビット（1）判断およびFalseビット（0）判断はそれぞれSKTおよびSKF命令を使用します。SKT, SKF命令はデータ・メモリに対してのみ行うことができます。

ビット判断は，プログラム・ステータス・ワード（PSWORD）のBCDフラグの影響を受けません。またCYフラグおよびZフラグには何の影響も与えません。ただし，CMPフラグはSKTおよびSKF命令が実行されるとリセット（0）されます。インデクス・イネーブル・フラグ（IXEフラグ）がセット（1）されているときは，インデクス・レジスタにより修飾されます。インデクス・レジスタによる修飾については第8章　システム・レジスタ（SYSREG）を参照してください。

11.5.1，11.5.2にTrueビット（1）判断およびFalseビット（0）判断について説明します。

表11－6　ビット判断命令

ビット判断	Trueビット（1）判断	SKT m, #n
	Falseビット（0）判断	SKF m, #n

11.5.1　Trueビット（1）判断

Trueビット（1）判断命令“SKT m, #n”は，データ・メモリの4ビットのうち，nで指定されたビットがTrue（1）であるかを判断します。nで指定されたビットがすべてTrue（1）であるとき，この命令の次の命令をスキップします。

```
例  MOV   M1,   #1011B
    SKT   M1,   #1011B ; ①
    BR    A
    BR    B
    SKT   M1,   #1101B ; ②
    BR    C
    BR    D
```

このとき，①ではM1のビット3，1，0を判断し，すべてTrue（1）であるからBへ分岐します。②では，M1のビット3，2，0を判断しますが，M1の2はFalse（0）であるため，Cへ分岐します。

11.5.2　Falseビット（0）判断

Falseビット（0）判断命令“SKF m, #n”はデータ・メモリの4ビットのうちnで指定されたビットがFalse（0）であるかを判断します。nで指定されたビットがすべてFalse（0）であるとき，この命令の次の命令をスキップします。

```
例  MOV   M1,   #1001B ;
    SKF   M1,   #0110B ; ①
    BR    A            ;
    BR    B            ;
    SKF   M1,   #1110B ; ②
    BR    C            ;
    BR    D            ;
```

このとき，①ではM1のビット2，1を判断し，すべてFalse（0）であるためBへ分岐します。②では，M1のビット3，2，1を判断しますが，M1のビット3はTrue（1）であるため，Cへ分岐します。

11.6　比較判断

表11－7に示すように，比較判断は“等しい”，“等しくない”，“以上”および“未満”の判断に分けられます。

“等しい”，“等しくない”，“以上”および“未満”の判断はそれぞれSKE, SKNE, SKGEおよびSKLT命令を使用します。

SKE, SKNE, SKGE, SKLT命令は，データ・メモリとイミーディエト・データとの比較判断のみ行うことができます。ジェネラル・レジスタとデータ・メモリとの比較判断を行うときは，プログラム・ステータス・ワード（PSWORD）のCMPフラグおよびZフラグを用いて減算命令により行えます。（**11.3　算術演算（2進4ビット加減算およびBCD加減算）**参照）。

比較判断は，プログラム・ステータス・ワード（PSWORD）のBCDフラグおよびCMPフラグの影響を受けません。またCYフラグおよびZフラグには何の影響も与えません。

11.6.1 - 11.6.4に“等しい”，“等しくない”，“以上”および“未満”の判断について説明します。

表11－7　比較判断命令

比較判断	等しい SKE m, #n4
	等しくない SKNE m, #n4
	以上 SKGE m, #n4
	未満 SKLT m, #n4

11.6.1　“等しい”の判断

“等しい”の判断命令“SKE m, #n4”はデータ・メモリとイミーディエト・データの内容が“等しい”かを判断します。

データ・メモリとイミーディエト・データの内容が“等しい”とき，この命令が次の命令をスキップします。

```
例  MOV   M1,   #1010B
    SKE   M1,   #1010B ; ①
    BR    A
    BR    B
    ;
    SKE   M1,   #1000B ; ②
    BR    C
    BR    D
```

このとき，①では，M1の内容とイミーディエト・データの1010Bが等しいためBへ分岐します。

②では，M1の内容とイミーディエト・データの1000Bが等しくないためCへ分岐します。

11.6.2　“等しくない”の判断

“等しくない”の判断命令“SKNE m, #n4”は，データ・メモリとイミーディエト・データの内容が“等しくない”かを判断します。

データ・メモリとイミーディエト・データの内容が“等しくない”とき，この命令の次の命令をスキップします。

```
例  MOV   M1,   #1010B
    SKNE  M1,   #1000B ; ①
    BR    A
    BR    B
    ;
    SKNE  M1,   #1010B ; ②
    BR    C
    BR    D
```

このとき，①では，M1の内容とイミーディエト・データの1000Bが等しくないためBへ分岐します。

②では，M1の内容とイミーディエト・データの1010Bが等しいためCへ分岐します。

11.6.3　“以上”の判断

“以上”の判断命令“SKGE m, #n4”はデータ・メモリとイミーディエト・データの内容を比較し，データ・メモリの内容が，イミーディエト・データより“大きい”か，または“等しい”ときに，この命令の次の命令をスキップします。

```
例  MOV     M1,     #1000B
    SKGE    M1,     #0111B ; ①
    BR      A
    BR      B
    ;
    SKGE    M1,     #1000B ; ②
    BR      C
    BR      D
    ;
    SKGE    M1,     #1001B ; ③
    BR      E
    BR      F
```

このとき，M1の内容は1000Bであるため①は“大きい”，②は“等しい”，③は“小さい”と判断され，それぞれB, D, Eに分岐します。

11.6.4　“未満”の判断

“未満”の判断命令“SKLT m, #n4”はデータ・メモリとイミーディエト・データの内容を比較し，データ・メモリの内容が，イミーディエト・データより“小さい”とき，この命令の次の命令をスキップします。

```
例  MOV     M1,     #1000B
    SKLT    M1,     #1001B ; ①
    BR      A
    BR      B
    ;
    SKLT    M1,     #1000B ; ②
    BR      C
    BR      D
    ;
    SKLT    M1,     #0111B ; ③
    BR      E
    BR      F
```

このとき，M1の内容は1000Bであるため，①は“小さい”，②は“等しい”，③は“大きい”と判断され，それぞれB,C,Eに分岐します。

11.7　回転処理

回転処理には，右回転処理と左回転処理に分けられます。

右回転処理にはRORC命令を使用します。

RORC命令は，ジェネラル・レジスタに対してのみ行うことができます。

RORC命令による回転処理は，プログラム・ステータス・ワード（PSWORD）のBCDフラグおよびCMPフラグの影響を受けません。またZフラグには何の影響も与えません。

左回転処理は，加算命令であるADDC命令により行うことができます。

11.7.1，11.7.2で，回転処理について説明します。

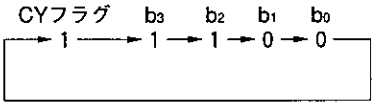
11.7.1　右回転処理

右回転処理命令RORC rはジェネラル・レジスタの内容を下位ビット方向に1ビット回転します。

このとき，ジェネラル・レジスタの内容の最上位ビット3にはCYフラグの内容が書き込まれ，最下位ビット0の内容をCYフラグに書き込みます。

例1．
```
MOV   PSW,   #0100B ; CYフラグをセット（1）
MOV   R1,    #1100B
RORC  R1
```

このとき，次のように処理されます。



すなわち，CYフラグ → b_3, b_3 → b_2, b_2 → b_1, b_1 → b_0, b_0 → CYフラグのように右に回転を行います。

2．
```
MOV   PSW,   #0000B  ; CYフラグをリセット（0）
MOV   R1,    #1000B  ; 最上位
MOV   R2,    #0100B
MOV   R3,    #0010B  ; 最下位
RORC  R1
RORC  R2
RORC  R3
```

このとき，上記プログラムはCY, R1, R2, R3の13ビット・データを右に回転します。

11.7.2　左回転処理

左回転処理は加算命令である“ADDC r, m”命令を用いることにより行えます。

```
例  MOV   PSW,   #0000B  ; CYフラグをリセット（0）
    MOV   R1,    #1000B  ; 最上位
    MOV   R2,    #0100B
    MOV   R3,    #0010B  ; 最下位
    ADDC  R3, R3
    ADDC  R2, R2
    ADDC  R1, R1
    SKF1  CY
    OR    R3,    #0001B
```

このとき、上記プログラムはCY, R1, R2 ,R3の13ビット・データを左に回転します。

〔メ　モ〕

第12章　ポ　ー　ト

12.1　ポートP0A（$P0A_0$, $P0A_1$, $P0A_2$, $P0A_3$）

ポートP0Aは，出力ラッチ付き4ビットの入出力ポートです。データ・メモリBANK0の70H番地にマッピングされています。出力形式はCMOSプッシュプル出力です。

4ビット単位で入力または出力の指定をすることができます。入力／出力の指定はレジスタ・ファイル上のP0AGIO（2CH番地のビット0）により行います。

P0AGIO＝0のとき，ポートP0Aのすべての端子は入力ポートとなり，ポート・レジスタに対しデータの読み込み命令を実行すると，端子の状態が読み出されます。

P0AGIO＝1のとき，ポートP0Aのすべての端子は出力ポートとなり，出力ラッチに書かれた内容が端子に出力されます。端子が出力ポートのとき読み出し命令を実行すると，端子の状態ではなく，出力ラッチの内容が取り込まれます。

また，ソフトウエア制御によるプルアップ抵抗を内蔵しています。プルアップ抵抗を内蔵するか，しないかの選択は，レジスタ・ファイルのP0AGPU（0CH番地のビット0）によって行います。P0AGPU＝1のとき4ビットの端子すべてがプルアップされ，P0AGPU＝0のときプルアップ抵抗は内蔵されません。

リセット時にはP0AGIOおよびP0AGPUは“0”になり，P0Aの端子はすべてプルアップ抵抗なしの入力ポートになります。また，ポートの出力ラッチの値も“0”になります。

表12－1　ポート・レジスタ（0.70H）への書き込みと読み出し

P0AGIO RF：2CH, ビット0	端子の入力／出力	BANK0 70H	
		書き込み	読み出し
0	入力	可能 P0Aラッチに書き込み	P0Aの端子の状態
1	出力		P0Aのラッチの内容

12.2　ポートP0B（$P0B_0$, $P0B_1$, $P0B_2$, $P0B_3$）

ポートP0Bは，出力ラッチ付き4ビットの入出力ポートです。データ・メモリのBANK0の71H番地にマッピングされています。出力形式はCMOSプッシュプル出力です。

4ビット単位で入力または出力の指定をすることができます。入力／出力の指定は，レジスタ・ファイル上のP0BGIO（2CH番地のビット1）により行います。

P0BGIO＝0のとき，ポートP0Bのすべての端子は入力ポートとなり，ポート・レジスタに対しデータの読み込み命令を実行すると，端子の状態が読み出されます。

P0BGIO＝1のとき，ポートP0Bのすべての端子は出力ポートとなり，出力ラッチに書かれた内容が端子に出力されます。端子が出力ポートのとき読み出し命令を実行すると，端子の状態ではなく，出力ラッチの内容が取り込まれます。

また，ソフトウエア制御によるプルアップ抵抗を内蔵しています。プルアップ抵抗を内蔵するか，しないかの選択は，レジスタ・ファイルのP0BGPU（0CH番地のビット1）によって行います。P0BGPU＝1のとき4ビットの端子すべてがプルアップされ，P0BGPU＝0のときプルアップ抵抗は内蔵されません。

リセット時にはP0BGIOおよびP0BGPUは“0”になり，P0Bの端子はすべてプルアップ抵抗なしの入力ポートになります。また，ポートの出力ラッチの値も“0”になります。

表12－2　ポート・レジスタ（0.71H）への書き込みと読み出し

P0BGIO RF：2CH, ビット1	端子の入力／出力	BANK0 71H	
		書き込み	読み出し
0	入力	可能 P0Bラッチに書き込み	P0Bの端子の状態
1	出力		P0Bのラッチの内容

12.3　ポートP0C（$P0C_0/ADC_0$, $P0C_1/ADC_1$, $P0C_2/ADC_2$, $P0C_3/ADC_3$）

ポートP0Cは，出力ラッチ付き4ビットの入出力ポートです。データ・メモリのBANK0の72H番地にマッピングされています。出力形式はCMOSプッシュプル出力です。

1ビットごとに入力または出力を指定することができます。入力／出力の指定はレジスタ・ファイル上のP0CBIO0-P0CBIO3（1CH番地）により行います。

P0CBIOn＝0のとき（n＝0-3），P0Cn端子は入力ポートとなり，ポート・レジスタに対しデータの読み出し命令を実行すると，端子の状態が読み出されます。また，P0CBIOn＝1のとき（n＝0-3），P0Cn端子は出力ポートとなり，出力ラッチに書かれた内容が端子に出力されます。端子が出力ポートのとき読み出し命令を実行すると，端子の状態ではなく，出力ラッチの内容が取り込まれます。

リセット時にはP0CBIO0-P0CBIO3は“0”になり，P0Cの端子はすべて入力ポートになります。また，ポートの出力ラッチの内容も“0”になります。

ポートP0CはA/Dコンバータのアナログ入力として使用できます。ポートまたは，アナログ入力端子としての切り替えは，レジスタ・ファイル上のP0C0IDI-P0C3IDI（1BH番地）によって行います。

P0CnIDI = 0のとき（n = 0 - 3），P0Cn/ADCn端子はポートとして機能し，P0CnIDI = 1のとき（n = 0 - 3），P0Cn/ADCn端子はA/Dコンバータのアナログ入力として機能します。なお，P0CnIDI（n = 0 - 3）がどれか1ビットでも“1”に設定されると，P0F1/VREF端子はVREF端子として設定されます。

A/Dコンバータのアナログ入力端子として使用する場合は，リセット・スタート直後に，アナログ電圧が印加される端子に対してP0CnIDIをセット（1）し，ポートとしての入力を禁止してください。また，P0CBIOn（n = 0 - 3）をクリア（0）し，入力ポートに設定してください。アナログ入力端子として使用する端子はレジスタ・ファイル上のADCCH0, ADCCH1（22H番地のビット1，0）で選択します。

リセット時，P0CBIO0 - P0CBIO3, P0C0IDI - P0C3IDI, ADCCH0, ADCCH1は“0”に設定され，入力ポートとなります。

表12－3　ポートとA/Dコンバータの切り替え

(n = 0 - 3)

P0CnIDI RF：1BH	P0CBIOn RF：1CH	機能	BANK0 72H	
			書き込み	読み出し
0	0	入力ポート	可能 P0Cラッチ	端子の状態
	1	ポート出力	可能 P0Cラッチ	P0Cの ラッチの内容
1	0	A/Dのアナログ入力[注1]	可能 P0Cラッチ	P0Cの ラッチの内容
	1	出力ポートおよび A/Dのアナログ入力[注2]	可能 P0Cラッチ	P0Cの ラッチの内容

注1．端子をA/Dコンバータのアナログ入力として使用する場合の通常の設定です。

2．出力ポートとして機能しています。このときアナログ入力電圧は，ポートからの出力の影響を受けて変化してしまいます。端子をアナログ入力として使用する場合には，必ずP0CBIOn = 0に設定してください。 ★

12.4　ポートOD（P0D0/$\overline{SCK}$, P0D1/SO, P0D2/SI, P0D3/$\overline{TM1OUT}$）

ポートODは出力ラッチ付き4ビットの入出力ポートです。データ・メモリのBANK0の73H番地にマッピングされています。出力形式はN-chオープン・ドレーン出力です。

1ビット単位で入力または出力を指定することができます。入力／出力の指定はレジスタ・ファイル上のP0DBIO0-P0DBIO3（2BH番地）により行います。

P0DBIOn = 0のとき（n = 0 - 3），P0Dn端子は入力ポートとなり，ポート・レジスタに対しデータの読み出し命令を実行すると，端子の状態が読み出されます。また，P0DBIOn = 1のとき，P0Dn端子は出力ポートとなり，出力ラッチに書かれた値が端子に出力されます。端子が出力ポートのとき，読み出し命令を実行すると，端子の状態ではなく，出力ラッチの値が取り込まれます。

また，ソフトウエア制御によるプルアップ抵抗を内蔵しています。プルアップ抵抗を内蔵するか，しないかの選択は，レジスタ・ファイルのP0DBPU0 - P0DBPU3（0DH番地）によって1ビットごとに行います。P0DBPUn = 1のときP0Dn端子がプルアップされ，P0DBPUn = 0のときプルアップ抵抗は内蔵されません。

リセット時には，P0DBIOnは“0”になり，P0Dの端子はすべて入力になり，ポートの出力ラッチの内容もすべて“0”になります。なお，P0DBIOnを“1”から“0”に変化させても出力ラッチの内容は変わりません。

また，ポートとして使用できるほかに，シリアル・インタフェース用の入出力やタイマ1の出力として使用できます。ポート（P0D0 - P0D2）とシリアル・インタフェース用入出力（$\overline{SCK}$, SO, SI）の切り替えは，レジスタ・ファイル上のSIOEN（0BHのビット0）によって行います。また，ポート（P0D3）とタイマ1出力（$\overline{TM1OUT}$）の切り替えはレジスタ・ファイル上のTM1OSEL（0BHのビット3）によって行います。TM1OSEL = 1を選択すると，タイマ1のリセット時には“1”を出力し，タイマ1のカウント値がモジュロ・レジスタの内容と一致するごとにその出力を反転します。

表12－4　レジスタ・ファイルの内容と端子の機能

(n = 0 - 3)

レジスタ・ファイルの値			端子の機能			
TM1OSEL RF：0BH ビット3	SIOEN RF：0BH ビット0	P0DBIOn RF：2BH ビットn	P0D0/$\overline{SCK}$	P0D1/SO	P0D2/SI	P0D3/$\overline{TM1OUT}$
0	0	0	入力ポート			
		1	出力ポート			
	1	0	$\overline{SCK}$	SO	SI	入力ポート
		1				出力ポート
1	0	0	入力ポート			$\overline{TM1OUT}$
		1	出力ポート			
	1	0	$\overline{SCK}$	SO	SI	
		1				

表12－5　ポート・レジスタ（0.73H）を読み出したときの内容

ポートのモード		ポート・レジスタ（0.73H）を読み出したときの内容
入力ポート		端子の状態
出力ポート		出力ラッチの内容
$\overline{SCK}$	シリアル・クロックに内部クロックを選択	出力ラッチの内容
	シリアル・クロックに外部クロックを選択	端子の状態
SI		端子の状態
SO		出力ラッチの内容
$\overline{TM1OUT}$		出力ラッチの内容

12.5　ポートOE（$P0E_0$, $P0E_1$, $P0E_2$, $P0E_3$）

ポートOEは出力ラッチ付き4ビットの入出力ポートです。データ・メモリのBANK0の6EH番地にマッピングされています。出力形式はN-chオープン・ドレーン出力です。

4ビット単位で入力または出力の指定をすることができます。入力／出力の指定はレジスタ・ファイル上のP0EGIO（2CH番地のビット2）により行います。

P0EGIO＝0のとき，ポートOEのすべての端子は入力ポートとなり，ポート・レジスタに対しデータの読み出し命令を実行すると，端子の状態が読み出されます。また，P0EGIO＝1のとき，ポートOEのすべての端子は出力ポートとなり，出力ラッチに書かれた内容が端子に出力されます。端子が出力ポートのとき読み出し命令を実行すると，端子の状態ではなく，出力ラッチの内容が取り込まれます。

また，ソフトウエア制御によるプルアップ抵抗を内蔵しています。プルアップ抵抗を内蔵するか，しないかの選択は，レジスタ・ファイルのP0EGPU（0CH番地のビット2）によって行います。P0EGPU＝1のとき4ビットの端子すべてがプルアップされ，P0EGPU＝0のときプルアップ抵抗は内蔵されません。

リセット時にはP0EGIOは“0”になり，ポートOEの端子はすべて入力ポートになります。また，ポートの出力ラッチの内容も“0”になります。

表12－6　ポート・レジスタ（0.6EH）への書き込みと読み出し

（n＝0－3）

P0EGIO RF：2CH，ビット2	端子の入力／出力	BANK0 6EH	
		書き込み	読み出し
0	入力	可能 P0Eラッチに書き込み	P0Eの端子の状態
1	出力		P0Eのラッチの内容

12.6　ポートOF（P0F0/$\overline{RLS}$, P0F1/VREF）

ポートOFは2ビットの入力専用ポートです。データ・メモリのBANK0の6FH番地にマッピングされています。また，マスク・オプションにより1ビットごとに端子にプルアップ抵抗の内蔵を指定することができます。

ポートOFの各端子が入力ポートとして機能している場合，ポート・レジスタに対して読み出し命令を実行すると，上位2ビットは0に固定され，下位2ビットに端子の状態が読み出されます。また，書き込み命令を実行した場合ポート・レジスタは何も変化せず，意味を持ちません。

P0F0/$\overline{RLS}$端子は，スタンバイ解除信号の入力端子としても使用できます。

P0F1/VREF端子は，P0CnIDI（RF：1BH番地, n = 0 - 3）のどれか1ビットでも“1”になっている場合，VREF端子（A/Dコンバータの基準電圧入力端子）として機能します。P0F1/VREF端子がVREF端子として機能している場合に，ポート・レジスタに対して読み出し命令を実行すると，6FH番地のビット1は必ず0となります。

12.7　ポート制御レジスタ

12.7.1　グループI/Oの入力／出力切り替え

4ビット単位で入力／出力を切り替えるポートをグループI/Oといいます。グループI/OとしてポートOA，ポートOB，ポートOEがあり，これらの入力／出力切り替えは次に示すレジスタで行います。

図12－1　グループI/Oのポート制御レジスタ

RF：2CH

	ビット3	ビット2	ビット1	ビット0
	0	P0EGIO	P0BGIO	P0AGIO
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

P0AGIO	機　　能
0	ポートOAを入力モードに設定
1	ポートOAを出力モードに設定

P0BGIO	機　　能
0	ポートOBを入力モードに設定
1	ポートOBを出力モードに設定

P0EGIO	機　　能
0	ポートOEを入力モードに設定
1	ポートOEを出力モードに設定

12.7.2　ビットI/Oの入力／出力切り替え

1ビット単位で入力／出力を切り替えるポートをビットI/Oといいます。ビットI/OとしてポートOC，ポートODがあり，これらの入力／出力の切り替えは次に示すレジスタで行います。

図12－2　ビットI/Oのポート制御レジスタ（1/2）

RF：1CH

	ビット3	ビット2	ビット1	ビット0
	P0CBIO3	P0CBIO2	P0CBIO1	P0CBIO0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

P0CBIO0	機　　能
0	$P0C_0$を入力モードに設定
1	$P0C_0$を出力モードに設定

P0CBIO1	機　　能
0	$P0C_1$を入力モードに設定
1	$P0C_1$を出力モードに設定

P0CBIO2	機　　能
0	$P0C_2$を入力モードに設定
1	$P0C_2$を出力モードに設定

P0CBIO3	機　　能
0	$P0C_3$を入力モードに設定
1	$P0C_3$を出力モードに設定

図12－2　ビットI/Oのポート制御レジスタ（2/2）

RF：2BH

	ビット3	ビット2	ビット1	ビット0
	P0DBIO3	P0DBIO2	P0DBIO1	P0DBIO0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

P0DBIO0	機　　能
0	$P0D_0$を入力モードに設定
1	$P0D_0$を出力モードに設定

P0DBIO1	機　　能
0	$P0D_1$を入力モードに設定
1	$P0D_1$を出力モードに設定

P0DBIO2	機　　能
0	$P0D_2$を入力モードに設定
1	$P0D_2$を出力モードに設定

P0DBIO3	機　　能
0	$P0D_3$を入力モードに設定
1	$P0D_3$を出力モードに設定

12.7.3　グループ・プルアップ・ポートのプルアップ抵抗内蔵の指定

4ビット単位でプルアップ抵抗内蔵の指定ができるポートをグループ・プルアップ・ポートと呼びます。グループ・プルアップ・ポートとしては，ポート0A，ポート0B，ポート0Eがあります。

グループ・プルアップ・ポートのプルアップ抵抗内蔵の指定は，それぞれP0AGPU, P0BGPU, P0EGPU（RF：0CH，ビット2，1，0）で行います。

図12－3　グループ・プルアップ・ポートのプルアップ抵抗内蔵指定レジスタ

RF：0CH

	ビット3	ビット2	ビット1	ビット0
	0	P0EGPU	P0BGPU	P0AGPU
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

P0AGPU	機　　能
0	ポート0Aにプルアップ抵抗を内蔵しない
1	ポート0Aにプルアップ抵抗を内蔵する

P0BGPU	機　　能
0	ポート0Bにプルアップ抵抗を内蔵しない
1	ポート0Bにプルアップ抵抗を内蔵する

P0EGPU	機　　能
0	ポート0Eにプルアップ抵抗を内蔵しない
1	ポート0Eにプルアップ抵抗を内蔵する

12.7.4　ビット・プルアップ・ポートのプルアップ抵抗内蔵の指定

1ビット単位でプルアップ抵抗内蔵の指定ができるポートをビット・プルアップ・ポートと呼びます。ビット・プルアップ・ポートとしては，ポート0Dがあります。

ビット・プルアップ・ポートのプルアップ抵抗内蔵の指定は，P0DBPU0-P0DBPU3（RF：0DH）で行います。

図12－4　ビット・プルアップ・ポートのプルアップ抵抗内蔵指定レジスタ

RF：0DH

	ビット3	ビット2	ビット1	ビット0
	P0DBPU3	P0DBPU2	P0DBPU1	P0DBPU0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

P0DBPU0	機　　能
0	$P0D_0$にプルアップ抵抗を内蔵しない
1	$P0D_0$にプルアップ抵抗を内蔵する

P0DBPU1	機　　能
0	$P0D_1$にプルアップ抵抗を内蔵しない
1	$P0D_1$にプルアップ抵抗を内蔵する

P0DBPU2	機　　能
0	$P0D_2$にプルアップ抵抗を内蔵しない
1	$P0D_2$にプルアップ抵抗を内蔵する

P0DBPU3	機　　能
0	$P0D_3$にプルアップ抵抗を内蔵しない
1	$P0D_3$にプルアップ抵抗を内蔵する

第13章　周辺ハードウエア

13.1　8ビット・タイマ・カウンタ（TM0, TM1）

μPD17149の8ビット・タイマ・カウンタには，タイマ0（TM0）とタイマ1（TM1）の2チャネルのタイマがあります。

タイマ0のカウント・アップ信号をタイマ1のカウント・パルスとして用いることにより，1チャネルの16ビット・タイマとして用いることも可能です。

各タイマの制御は，PUT/GET命令を使ったハードウエアの操作とPEEK/POKE命令を使ったレジスタ・ファイル上のレジスタの操作により行います。

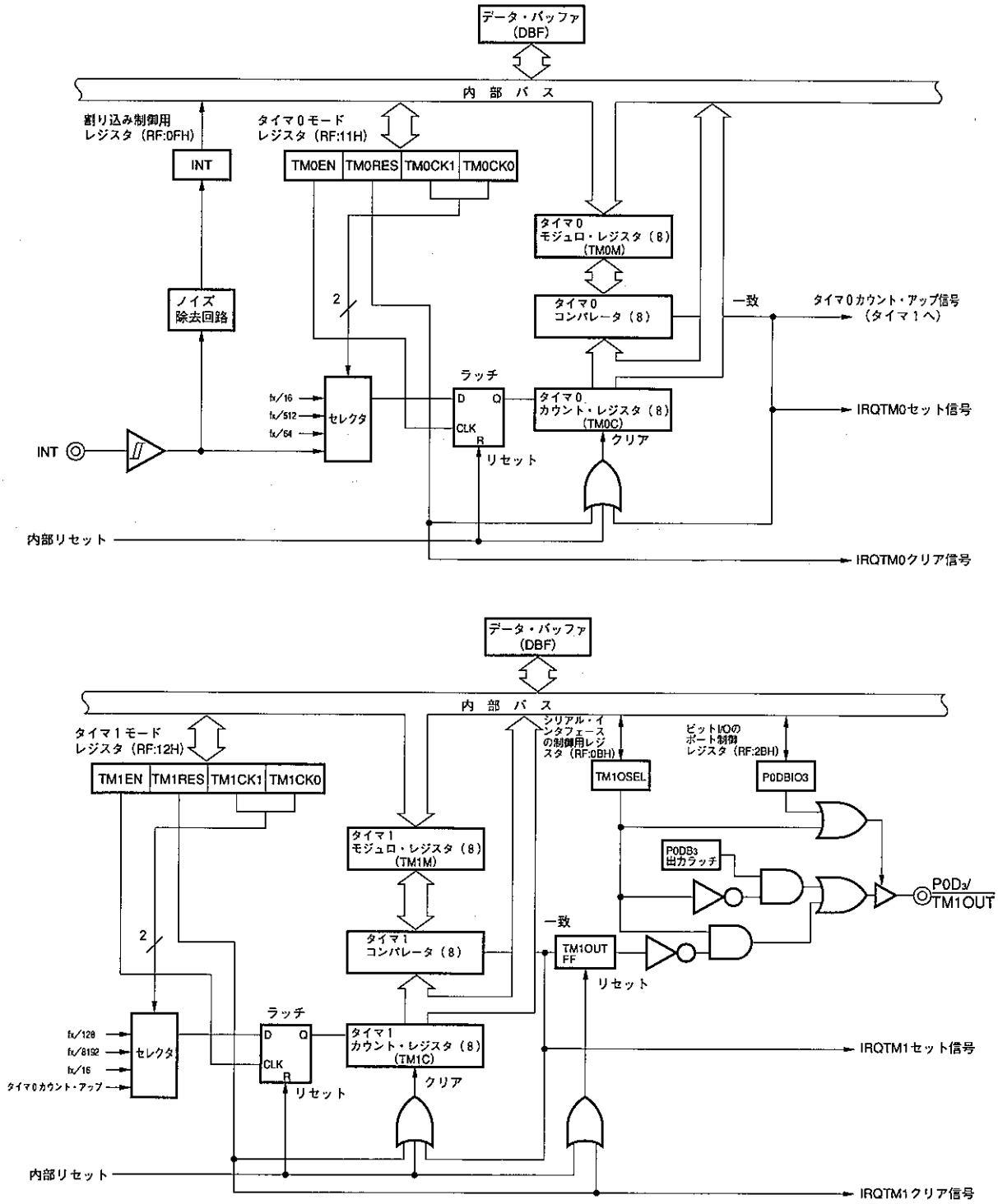
13.1.1　8ビット・タイマ・カウンタの構成

図13－1に8ビット・タイマ・カウンタの構成を示します。8ビット・タイマ・カウンタは8ビットのカウント・レジスタ，8ビットのモジュロ・レジスタ，カウント・レジスタとモジュロ・レジスタの値を比較するコンパレータおよびカウント・パルスを選択するセレクタで構成されています。

注意1．モジュロ・レジスタは，書き込み専用レジスタです。
　　2．カウント・レジスタは，読み出し専用レジスタです。

★

図13－1　8ビット・タイマ・カウンタの構成



備考　fx：システム・クロック発振周波数

図13－2　タイマ0モード・レジスタ

RF：11H

	ビット3	ビット2	ビット1	ビット0
	TM0EN	TM0RES	TM0CK1	TM0CK0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

TM0CK1	TM0CK0	タイマ0のカウント・パルスの選択
0	0	fx／16
0	1	fx／512
1	0	fx／64
1	1	INT端子からの外部クロック

TM0RES	タイマ0のリセット
0	タイマ0に影響なし
1	タイマ0カウント・レジスタとIRQTM0をリセット

備考　TM0RESは，セット（1）後，自動的にクリア（0）されます。読み出し時は常に“0”が読み出されます。

TM0EN	タイマ0のスタート指示
0	タイマ0のカウントを停止する
1	タイマ0のカウントを開始する

備考　TM0ENは，タイマ0のカウント状態を検出するステータス・フラグとして使用することができます（1：カウント動作中，0：カウント停止状態）。

図13－3　タイマ１モード・レジスタ

RF：12H

	ビット3	ビット2	ビット1	ビット0
	TM1EN	TM1RES	TM1CK1	TM1CK0
リード／ライト	R/W			
リセット時の初期値	1	0	0	0

TM1CK1	TM1CK0	タイマ１のカウント・パルスの選択
0	0	fx／128
0	1	fx／8192
1	0	fx／16
1	1	タイマ０からのカウント・アップ信号

TM1RES	タイマ１のリセット
0	タイマ１に影響なし
1	タイマ１カウント・レジスタとIRQTM1をリセット

備考　TM1RESは，セット（１）後，自動的にクリア（０）されます。読み出し時は常に“0”が読み出されます。

TM1EN	タイマ１のスタート指示
0	タイマ１のカウントを停止する
1	タイマ１のカウントを開始する

備考　TM1ENは，タイマ1のカウント状態を検出するステータス・フラグとして使用することができます（1：カウント動作中，0：カウント停止状態）。

13.1.2　８ビット・タイマ・カウンタの動作

（１）　カウント・レジスタ

タイマ０およびタイマ１のカウント・レジスタは，初期値が00Hの８ビットのアップ・カウンタで，カウント・パルスが入力されるごとにインクリメントされます。

カウント・レジスタが00Hに初期化されるのは，次のときです。

①　この製品がリセットされたとき（**第16章　リセット参照**）

②　８ビット・モジュロ・レジスタの内容とカウント・レジスタの値が一致し，コンパレータが一致信号を発生したとき

③　タイマ０の場合，レジスタ・ファイルのTM0RESに“１”が書き込まれたとき
　　タイマ１の場合，レジスタ・ファイルのTM1RESに“１”が書き込まれたとき

（2）モジュロ・レジスタ

タイマ0およびタイマ1のモジュロ・レジスタは，カウント・レジスタのカウント値を決定するレジスタで，初期値はFFHに設定されています。

モジュロ・レジスタに対する値の設定は，PUT命令によりDBF（データ・バッファ）を介して行います。

（3）コンパレータ

タイマ0およびタイマ1のコンパレータは，カウント・レジスタとモジュロ・レジスタの値が一致した次のカウント・パルスが入力された時点で，一致信号を出力します。つまり，モジュロ・レジスタの値が初期値FFHであった場合は，256カウントしたとき，一致信号を出力します。

コンパレータからの一致信号は，カウント・レジスタの内容を0にクリアするとともに，割り込み要求フラグ（IRQTM0, IRQTM1）を自動的に“1”にセットします。このときEI命令（割り込み受け付け許可命令）が実行されていて，かつ割り込み許可フラグ（IPTM0, IPTM1）がセットされている状態であれば，割り込みを受け付けます。割り込みを受け付けた場合，割り込み要求フラグ（IRQTM0またはIRQTM1）を“0”にして，プログラムの実行を割り込み処理に移します。

13.1.3　カウント・パルスの選択

タイマ0のカウント・パルスの選択は，TM0CK0およびTM0CK1で選択します。

システム・クロック（fx）を512分周，64分周または16分周したカウント・パルス，またはINT端子から入力される外部カウント・パルスの4種類の中から1つを選択することができます。

リセット時はTM0CK0 = 0, TM0CK1 = 0になっており，fx／16が選択されています。

タイマ1のカウント・パルスの選択は，TM1CK0およびTM1CK1で選択します。

fxを8192分周，128分周または16分周したカウント・パルス，または，タイマ0からのカウント・アップ信号4種類の中から1つを選択することができます。

また，電源立ち上げ時およびリセット時，タイマ1は発振安定待ち時間の生成に用いられます。このため，初期値はTM1CK0 = 0, TM1CK1 = 0になっており，カウント・パルスにはfx／128が選択されています。そして，TM1EN = 1が初期値として設定されていますので，fx = 4 MHzではリセットしてから約8 ms経過後，μPD17149は0000H番地からスタートします（**第16章　リセット**参照）。

13.1.4　モジュロ・レジスタへのカウント値の設定

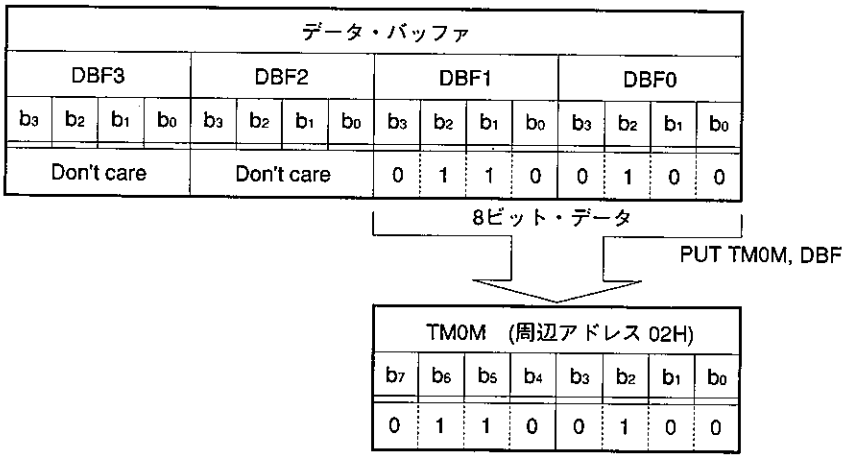
モジュロ・レジスタに対する値の設定は，PUT命令によりDBF（データ・バッファ）を介して行います。モジュロ・レジスタの周辺アドレスは，タイマ0は02H，タイマ1は03Hに割り付けられています。

PUT命令により値を転送する場合，DBFの下位8ビット（DBF1, DBF0）のデータがモジュロ・レジスタに転送されます。タイマ0の例を図13－4に示します。

図13－4　モジュロ・レジスタへのカウント値の設定

タイマ0のモジュロ・レジスタにカウント値64Hを設定した例

```
CONTDATL  DAT   4H          ；シンボル定義命令を用いCONTDATLを4Hに割り当てる
CONTDATH  DAT   6H          ；シンボル定義命令を用いCONTDATHを6Hに割り当てる
          MOV   DBF0, #CONTDATL ;
          MOV   DBF1, #CONTDATH ;
          PUT   TM0M, DBF     ；予約語“TM0M”を用い転送。
```



注意　モジュロ・レジスタに設定できる値の範囲は，01H - FFHです。00Hを設定すると，正常なカウント動作が行われません。

モジュロ・レジスタは，書き込み専用レジスタです。モジュロ・レジスタから設定値を読み出すことはできません。また，8ビット・タイマ・カウンタが動作中に，“PUT TM0M, DBF”命令または“PUT TM1M, DBF”命令を実行してもカウント動作を停止することはありません。

13.1.5　カウント・レジスタの値の読み出し

カウント・レジスタの値の読み出しは，GET命令によりDBF（データ・バッファ）を介してタイマ0とタイマ1を同時に行います。

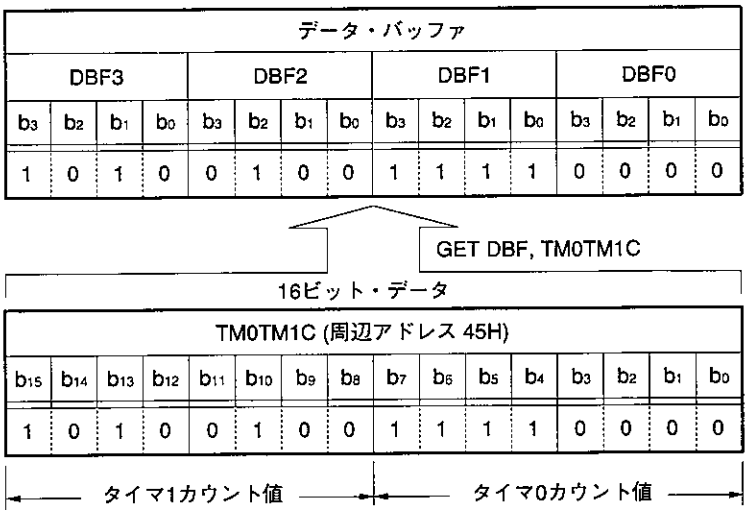
タイマ0とタイマ1のカウント・レジスタの値は，周辺アドレス45Hに割り付けられ，上位8ビットはタイマ1のカウント値，下位8ビットはタイマ0のカウント値に割り付けられています。

GET命令により，カウント・レジスタの値をDBFに読み出すことができます。なお，GET命令実行中には，カウント・レジスタはカウント動作を停止し，カウント値を保持している状態に入ります。なお，タイマが動作中でかつGET命令実行中にカウント・パルスが入力された場合，そのカウントを保留し，GET命令終了後カウント・レジスタを1つインクリメントしたのちカウント動作を続ける機能を備えています。

このため，1命令サイクル中にカウント・パルスが2つ以上入力されないかぎり，タイマの動作中にGET命令を実行しても，ミス・カウントすることはありません。

図13－5　カウント・レジスタのカウント値の読み出し

タイマ0のカウント値がF0H，タイマ1のカウント値がA4Hのとき
GET DBF, TM0TM1C；予約語DBFおよびTM0TM1Cを使用した例



★　13.1.6　インターバル時間の設定

コンパレータから一致信号が出力される時間間隔（インターバル時間）はモジュロ・レジスタに設定する値によって決まります。次にインターバル時間T［sec］からモジュロ・レジスタの設定値Nを求める計算方法を示します。

$$T = \frac{N+1}{f_{CP}} = (N+1) \times T_{CP}$$

$$N = T \times f_{CP} - 1 \text{ または } N = \frac{T}{T_{CP}} - 1 \text{（ただし，N = 1～255）}$$

f_{CP}　：カウント・パルスの周波数［Hz］

T_{CP}　：カウント・パルスの周期［sec］（$1/f_{CP}$ = 分解能）

●インターバル時間からカウント値を計算する場合の計算例とプログラム例

・タイマ1でインターバル時間に7 msを想定した例（システム・クロック：fx = 4 MHz）

インターバル時間に7 msを想定した場合，タイマの分解能から7 msちょうどのインターバル時間を設定することは不可能です。したがって最も近いインターバル時間を設定するためには，分解能が最大になるカウント・パルス（fx/128，分解能：32 μs）を選択し，カウント値を計算します。

（計算例）　T = 7 ms，分解能：32 μs

$$N = \frac{T}{\text{（分解能）}} - 1$$

$$= \frac{7 \times 10^{-3}}{32 \times 10^{-5}} - 1$$

$$= 217.75 \fallingdotseq 218 \text{（：DAH）}$$

インターバル時間が7 msに最も近くなるモジュロ・レジスタの値はDAHとなり，そのときのインターバル時間は7.008 msとなります。

（プログラム例）

```
MOV   DBF0, #0AH    ；予約語“DBF0”，“DBF1”を用いてDBFにDAHを
MOV   DBF1, #0DH    ；格納
PUT   TMM, DBF      ；予約語“TMM”を用いてDBFの内容を転送

INITFLG TM1EN, TM1RES, NOT TM1CK1, NOT TM1CK0
                    ；組み込みマクロ命令“INITFLG”を用いてTM1EN，TM1RES
                    ；をセット，タイマ1のカウント・パルスを“fX/128”に設定，
                    ；カウント・スタート
```

13.1.7　インターバル時間の誤差 ★

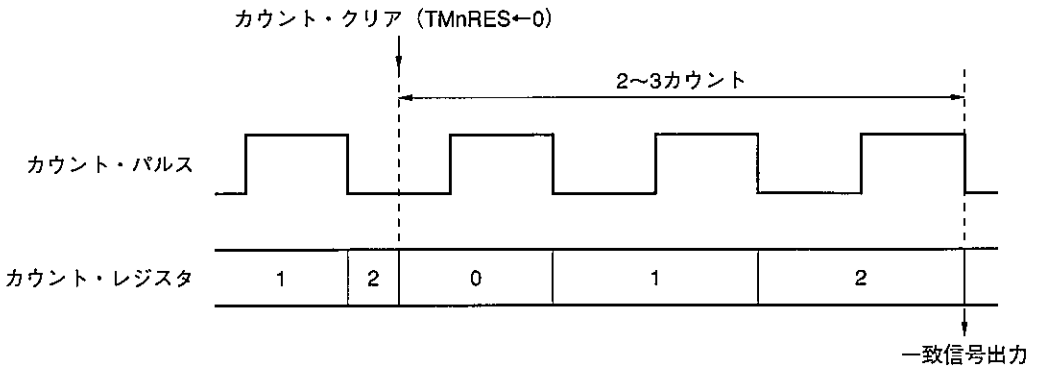
インターバル時間には，最大で−1.5カウントの誤差を生じることがあります。モジュロ・レジスタの設定値が小さい場合にはご注意ください。

（1）カウント中にカウント・レジスタを0にクリアしたときの誤差（最大誤差：−1カウント）

8ビット・タイマ・カウンタのカウント・レジスタは，TMnRESフラグをセット（1）することにより0にクリアされますが，システム・クロックからカウント・パルスを生成するための分周回路はリセットされません。

したがって，カウント中にTMnRESフラグをセット（1）してカウントを0にクリアした場合，1カウント目のタイミングにカウント・パルス1周期分の誤差が生じます。次にモジュロ・レジスタに2を設定した場合のカウント例を示します。

図13−6　カウント中にカウント・レジスタを0にクリアしたときの誤差



この例では3カウントごとに一致信号が出るはずですが，カウント・クリア後の最初の1回目だけは最小2カウントで一致信号が出力されます。

なお，TMnEN＝1←0と同時に，TMnRES←1した場合も上記の誤差が生じます。

（2）カウント停止状態からカウントを開始したときの誤差（最大誤差：−1.5カウント）

8ビット・タイマ・カウンタのカウント・レジスタは，TMnRESフラグをセット（1）することにより0にクリアされますが，システム・クロックからカウント・パルスを生成するための分周回路はリセットされません。

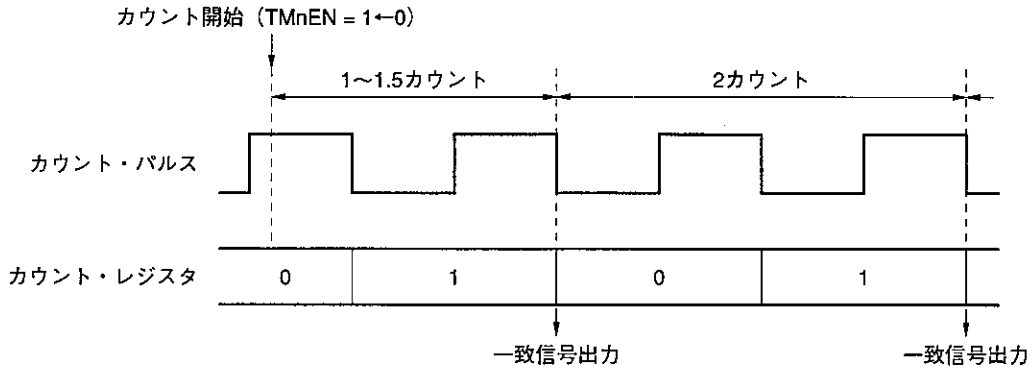
また，TMnENフラグをセット（1）してカウント停止状態からカウントを開始した場合，カウント・パルスがロウ・レベルから始まるかハイ・レベルから始まるかによって，1カウント目のタイミングが次のように異なります。

ハイ・レベルから始まる場合，次の立ち下がりが1カウント目
ロウ・レベルから始まる場合，カウント開始時点が1カウント目

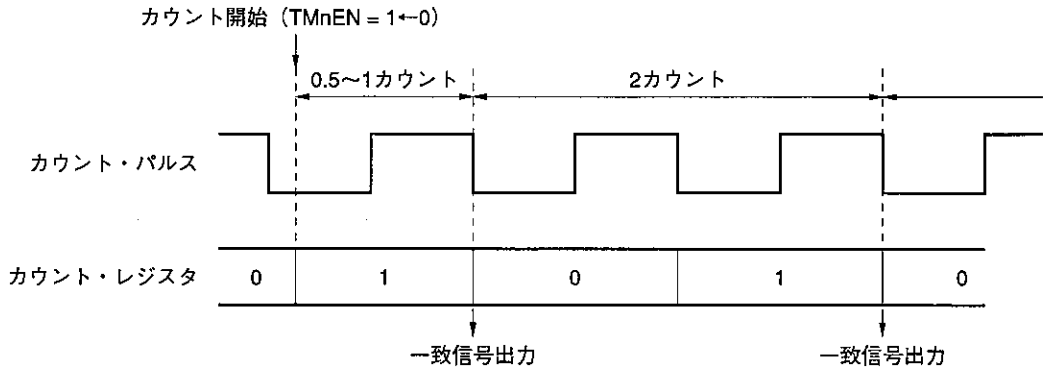
したがって，カウント開始後の最初の1回目だけは，一致信号が出るまでの時間に−0.5〜−1.5カウントの誤差が生じます。次にモジュロ・レジスタに1を設定した場合のカウントの例を示します。

図13−7　カウント停止状態からカウントを開始したときの誤差

（a）カウント・パルスからハイ・レベルから始まった場合（誤差：−0.5〜−1カウント）



（b）カウント・パルスがロウ・レベルから始まった場合（誤差：−1〜−1.5カウント）



この例では2カウントごとに一致信号が出るはずですが，最初の1回目だけは，最長で1.5カウント，最短で0.5カウント（誤差：−0.5〜−1.5カウント）で一致信号が出力されます。

なお，タイマは発振安定待ち時間の生成にも使用されているため，発振安定待ち時間にも上記の誤差が生じます。

13.1.8　タイマ1出力

TM1OSELフラグを“1”にセットすることにより，P0D3/$\overline{\text{TM1OUT}}$端子は，タイマ1出力端子として機能します。このときP0DBIO3の値は関係ありません。

タイマ1は内部に一致信号出力用のフリップフロップを持っており，コンパレータが一致信号を出力するたびに，その出力を反転させます。TM1OSELフラグを“1”にセットした場合，このフリップフロップの内容がP0D3/$\overline{\text{TM1OUT}}$端子に出力されます。

また，P0D3/$\overline{\text{TM1OUT}}$端子はN-chオープン・ドレーン出力端子で，マスク・オプションにより，プルアップ抵抗を内蔵することが可能です。プルアップ抵抗を内蔵しない場合，P0D3/$\overline{\text{TM1OUT}}$端子の初期状態はハイ・インピーダンスとなります。

なお，内部のタイマ1出力フリップフロップは，TM1EN＝1にした時点から動作を開始していますので，タイマ1出力が必ず端子の初期状態から始まるようにするためには，TM1RESに“1”をセットし，フリップフロップをリセットしてからスタートさせることが必要です。

図13−8　タイマ1出力設定用レジスタ

RF：0BH

	ビット3	ビット2	ビット1	ビット0
	TM1OSEL	0	0	SIOEN
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

SIOEN	機　能
0	P0D0/$\overline{\text{SCK}}$，P0D1/SO，P0D2/SIの各端子はポートとして機能
1	P0D0/$\overline{\text{SCK}}$，P0D1/SO，P0D2/SIの各端子はシリアル・インタフェースとして機能

注意　タイマ1の出力設定とは直接関係ありません。

TM1OSEL	機　能
0	P0D3/$\overline{\text{TM1OUT}}$端子はポートとして機能
1	P0D3/$\overline{\text{TM1OUT}}$端子はタイマ1の一致信号出力として機能

13.2　ベーシック・インターバル・タイマ（BTM）

μPD17149は，7ビットのベーシック・インターバル・タイマを内蔵しています。

ベーシック・インターバル・タイマには，次に示す機能があります。

（1）基準時間発生

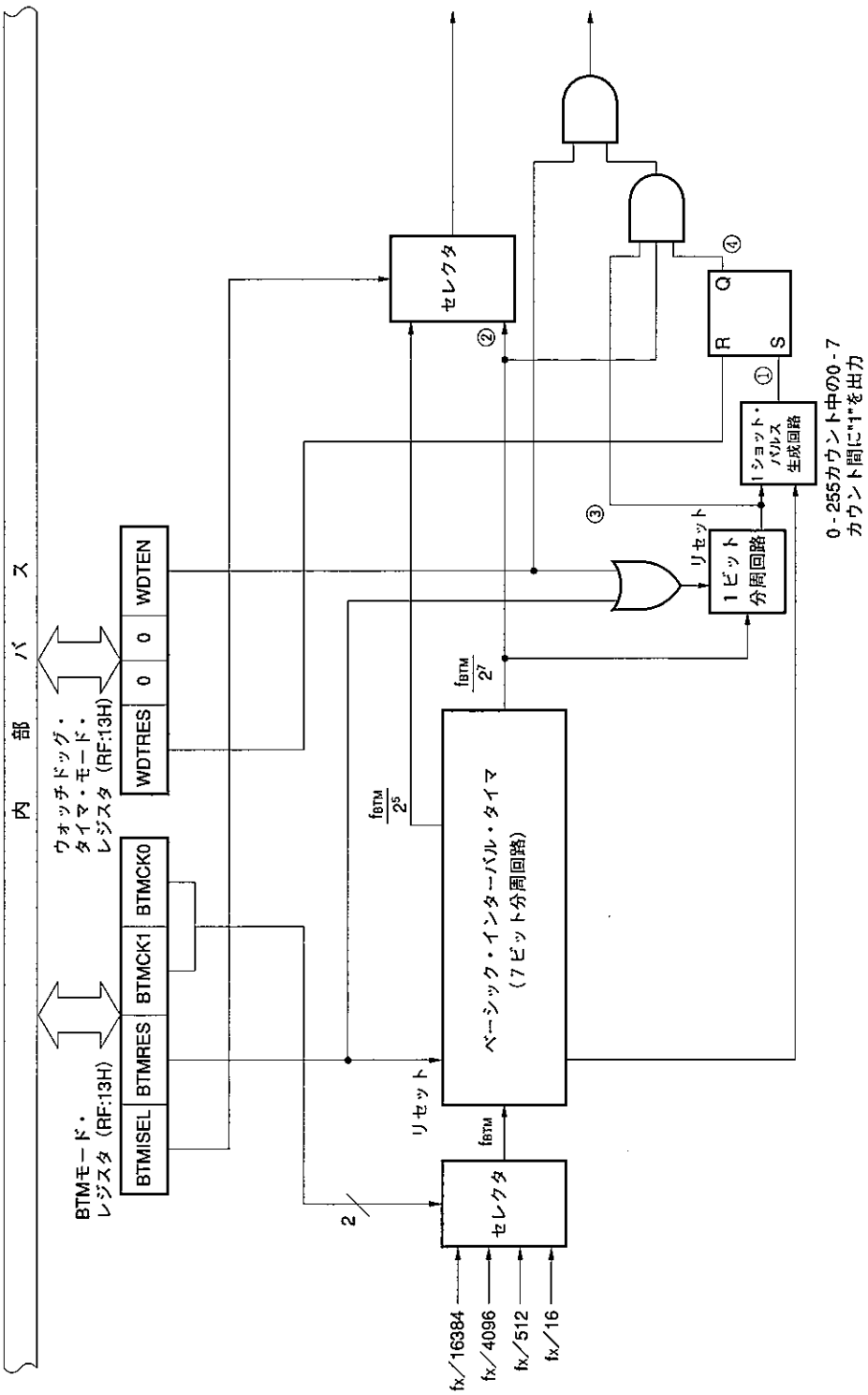
（2）スタンバイ・モード解除時のウエイト時間の選択とカウント

（3）プログラムの暴走を検出するウォッチドッグ・タイマ機能

13.2.1　ベーシック・インターバル・タイマの構成

図13－9にベーシック・インターバル・タイマの構成を示します。

図13－9　ベーシック・インターバル・タイマの構成



備考　図中の①-④は図13－12のタイミング・チャート中の信号を示します。

13.2.2　ベーシック・インターバル・タイマを制御するレジスタ

ベーシック・インターバル・タイマは，BTMモード・レジスタおよびウォッチドッグ・タイマ・モード・レジスタによって制御します。

図13－10，13－11にそれぞれのレジスタの構成を示します。

図13－10　BTMモード・レジスタ

RF：13H

	ビット3	ビット2	ビット1	ビット0
	BTMISEL	BTMRES	BTMCK1	BTMCK0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

BTMCK1	BTMCK0	BTMのカウント・パルスの選択
0	0	fx／16 （1命令実行時間）
0	1	fx／16384 （1024命令実行時間）
1	0	fx／4096 （256命令実行時間）
1	1	fx／512 （32命令実行時間）

BTMRES	BTMのリセット
0	ベーシック・インターバル・タイマ（BTM）には影響ありません。
1	ベーシック・インターバル・タイマ（BTM）のバイナリ・カウンタをリセットします。

備考　BTMRESは，セット（1）後，自動的にクリア（0）されます。読み出し時は常に“0”が読み出されます。

BTMISEL	インターバル時間の選択
0	インターバル時間をカウント・パルスの128分周に設定
1	インターバル時間をカウント・パルスの32分周に設定

図13－11　ウォッチドッグ・タイマ・モード・レジスタ

RF：03H

	ビット3	ビット2	ビット1	ビット0
	WDTRES	0	0	WDTEN
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

WDTEN	ウォッチドッグ・タイマ機能の許可
0	ウォッチドッグ・タイマが停止の状態です。
1	ウォッチドッグ・タイマの動作を開始します。

備考1．WDTENは，プログラムではクリア（0）できません。

2．WDTENは，セット（1）後，自動的にクリア（0）されます。読み出し時は常に“0”が読み出されます。

WDTRES	ウォッチドッグ・タイマのリセット
0	ウォッチドッグ・タイマに影響はありません。
1	ウォッチドッグ・タイマに用いるBTMのオーバフロー・キャリーを保持するフリップ・フロップがリセットされます。

備考　WDTRESは，セット（1）後，自動的にクリア（0）されます。読み出し時は常に“0”が読み出されます。

13.2.3　ベーシック・インターバル・タイマの動作

ベーシック・インターバル・タイマは，BTMモード・レジスタで指定されるカウント・パルスにより常にカウント・アップされている7ビットのバイナリ・カウンタです。カウント動作を停止させることはできません。

ベーシック・インターバル・タイマは，BTMISELによりインターバル時間を切り替えることができます。BTMISEL＝0のとき，インターバル時間はカウント・パルスを128分周した時間（128/f_{BTM}）となり，BTMISEL＝1のとき，カウント・パルスを32分周した時間（32/f_{BTM}）となります。

なお，インターバル時間を切り替えてもカウンタの内容はクリア（0）されません。

13.2.4　ウォッチドッグ・タイマ機能

ベーシック・インターバル・タイマは，プログラムの暴走を検出するウォッチドッグ・タイマとしても使用できます。

（1）ウォッチドッグ・タイマの機能

ウォッチドッグ・タイマは，リセット信号を一定周期で発生するカウンタです。プログラムにより毎回このリセット信号の発生を禁止することにより，外部ノイズなどによりシステムが暴走した（ウォッチドッグ・タイマが所定の時間内にリセットされなかった）場合に，システムに対してリセット（0000H番地スタート）をかけることができます。

この機能によって，プログラムが外部ノイズなどにより意図しないルーチンに跳び，無限ループに陥った場合でも，一定時間内にリセット信号が発生するため暴走状態から脱出できます。

（2）ウォッチドッグ・タイマの動作

WDTENをセット（1）すると，1ビット分周回路が動作可能状態になり，ベーシック・インターバル・タイマは8ビットのウォッチドッグ・タイマとして動作を開始します。

ウォッチドッグ・タイマはいったん動作させると，デバイスにリセットがかかりWDTENがクリア（0）されるまで止めることはできません。

ウォッチドッグ・タイマによるリセットを禁止するには，次の2つの手段があります。

① WDTRESをセットすることをプログラム中で繰り返す。
② BTMRESをセットすることをプログラム中で繰り返す。

①の場合，図13—12に示すように，ウォッチドッグ・タイマのカウント値が8から191（192になる直前）までの期間にWDTRESをセットする必要があります。したがって，ウォッチドッグ・タイマが184カウントする周期より短いタイミングで少なくとも1回は“SET1　WDTRES”が実行されるようにプログラムします。

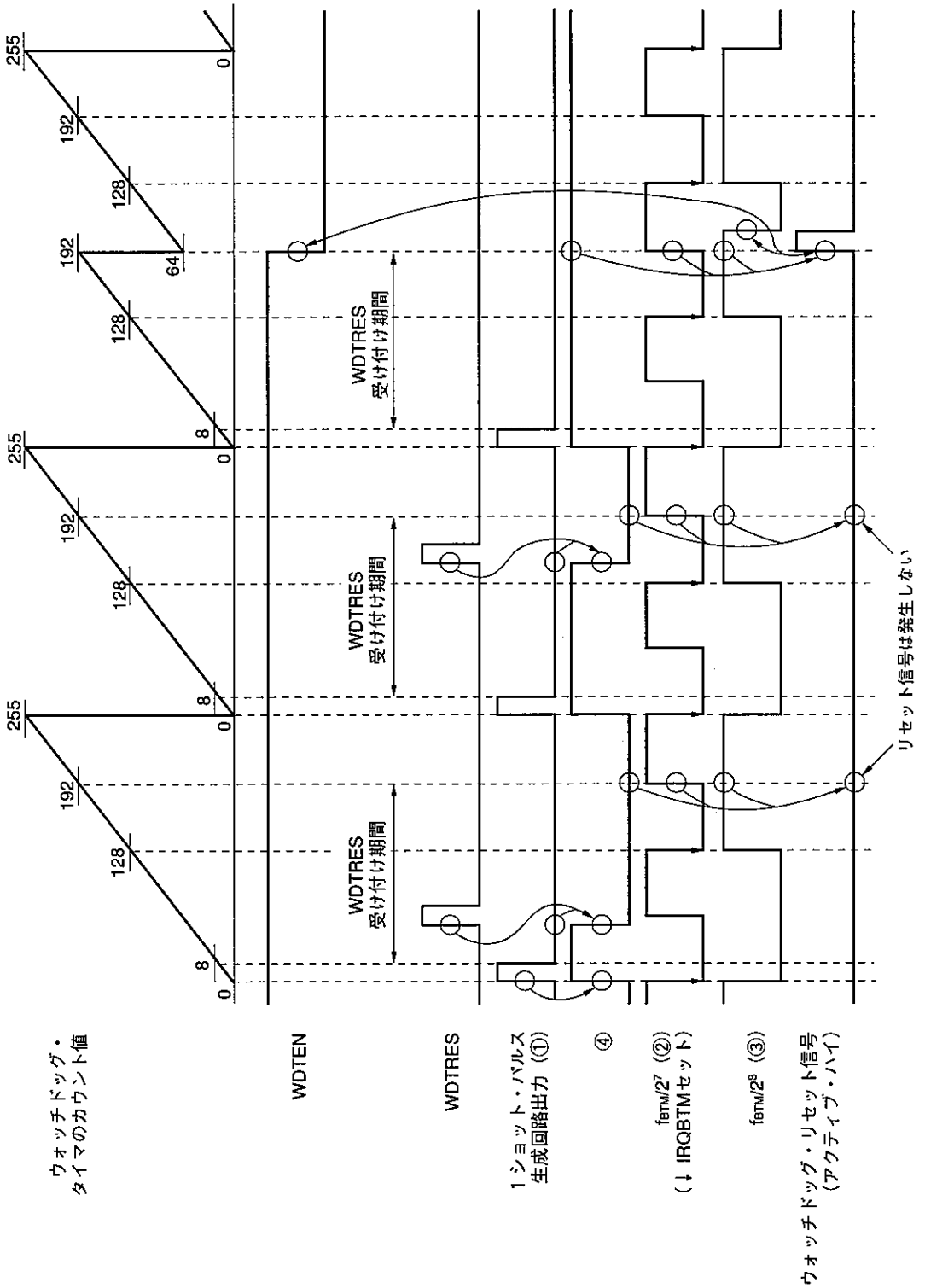
②の場合，ベーシック・インターバル・タイマ（BTM）が128カウントするまでの期間にBTMRESをセットする必要があります。したがって，BTMが128カウントする周期より短いタイミングで少なくとも1回は“SET1　BTMRES”が実行されるようにプログラムします。ただし，この方法ではBTMによる割り込み処理はできなくなります。

注意　WDTENをセットしてもBTMはリセットされません。したがって，最初にWDTENをセットする前に，必ずBTMRESをセットして，BTMをリセットするようにしてください。

例

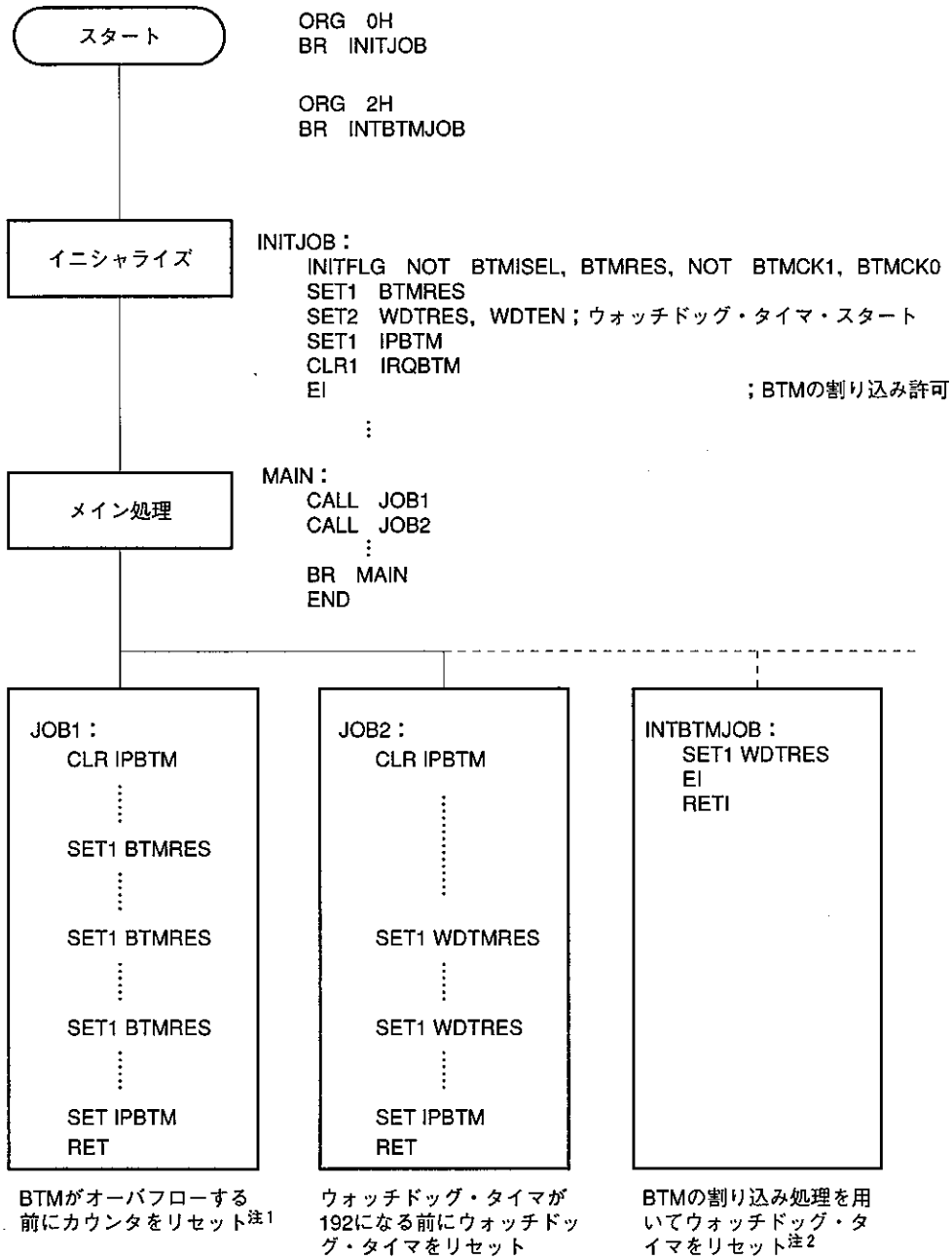
```
        ⋮
SET1    BTMRES
SET2    WDTEN, WDTRES
        ⋮
```

図13−12　ウォッチドッグ・タイマのタイミング・チャート（WDTRESフラグを利用した場合）★



★　（3）ウォッチドッグ・タイマのプログラム例

（プログラム例）



注1．BTMがオーバフローする前にカウンタをリセットする方式では，BTMによる割り込み処理はできません。

2．BTMの割り込み処理を用いてウォッチドッグ・タイマをリセットする方式は，ほかの2つの方式に比べてプログラミングは容易ですが，暴走の検出率は劣ります。

13.3　A/Dコンバータ

μPD17149には，4チャネルのアナログ入力（P0C0/ADC0 - P0C3/ADC3）を持つ8ビット分解能のA/Dコンバータを内蔵しています。

A/Dコンバータは逐次比較法を採用しています。動作モードは，次の2つです。

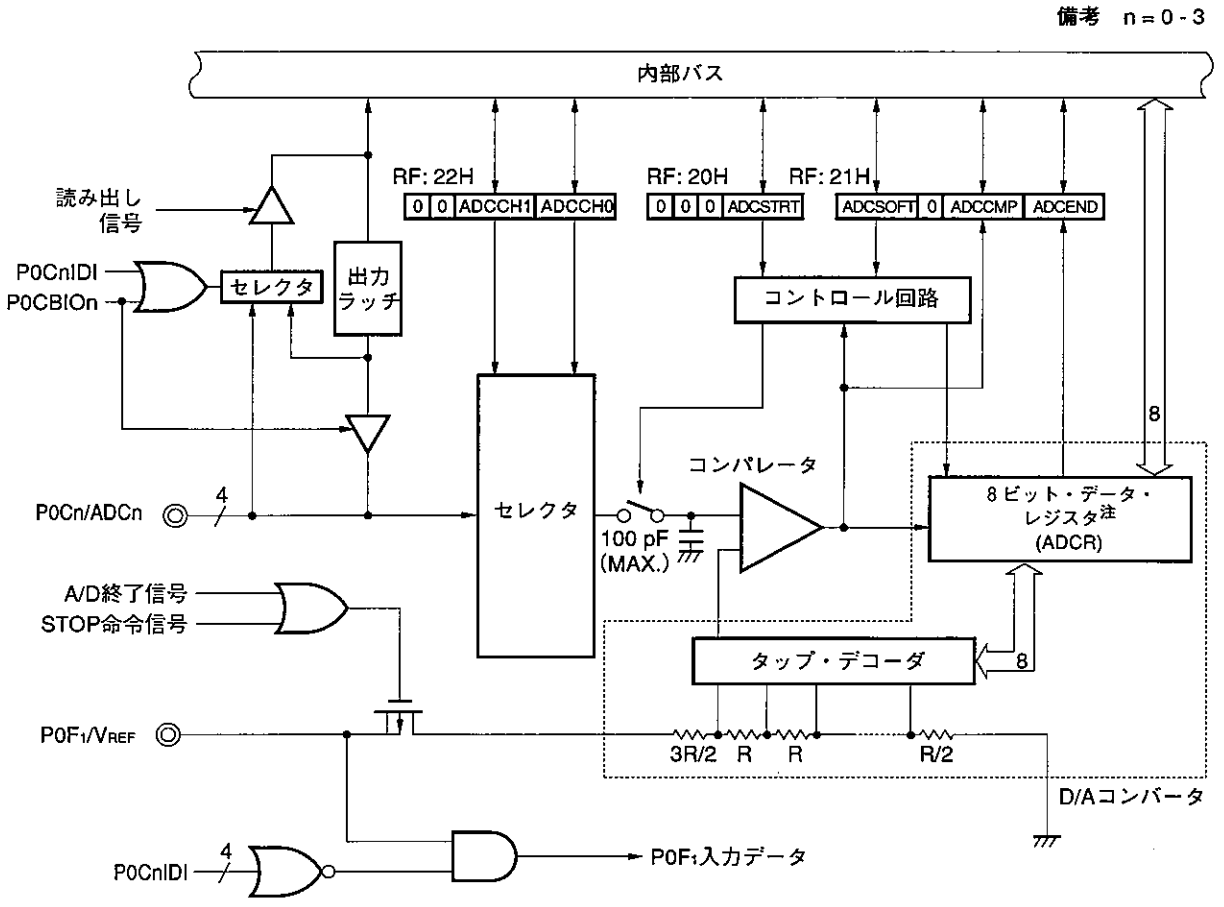
① 8ビットのA/D変換を上位ビットから順に行う連続モード

② 8ビット・データ・レジスタで設定した任意の電圧値と大小比較を行う単発モード

13.3.1　A/Dコンバータの構成

A/Dコンバータは，図13−13に示すように構成されています。

図13−13　A/Dコンバータのブロック図　★



注　STOP命令実行時，8ビット・データ・レジスタ（ADCR）は，00Hにクリアされます。　★

備考　A/D変換中にHALT命令が実行されてもA/D変換は継続し，変換が終了するとVREF端子に流れ込む電流はカットされます。　★

13.3.2　A/Dコンバータの機能

（1）ADC_0 - ADC_3端子

A/Dコンバータへの4チャネルのアナログ電圧の入力端子です。A/D変換するアナログ信号を入力します。A/Dコンバータ内部にはサンプル・ホールド回路が内蔵されており，A/D変換中のアナログ入力電圧は内部で保持されています。

（2）V_{REF}端子

A/Dコンバータの基準電圧を入力する端子です。

V_{REF} - GND間にかかる電圧に基づいて，ADC_0 - ADC_3に入力される信号をディジタル信号に変換します。なお，μPD17149のA/Dコンバータは，消費電流を抑えるため，A/Dコンバータが動作していないときは，自動的にV_{REF}端子に流れ込む電流を止める機能が内蔵されています。V_{REF}端子に電流が流れるときは，次の場合です。

① 連続モード（ADCSOFT = 0）のとき

ADCSTRTフラグがセット（1）されてからADCENDフラグがセット（1）されるまでの間

② 単発モード（ADCSOFT = 1）のとき

ADCSTRTフラグがセット（1）または8ビット・データ・レジスタの値が書き込まれてからコンパレータの比較結果がADCCMPフラグに書き込まれるまでの間

備考1．A/D変換中にHALT命令が実行されても，A/Dコンバータは連続モードではADCENDフラグがセットされるまで，また単発モードではADCCMPフラグに結果が格納されるまで，動作します。したがって，V_{REF}端子にも上記の期間，電流が流れます。

2．A/D変換中にSTOP命令が実行されると変換を中断します。また，A/Dコンバータは初期状態にイニシャライズされ，V_{REF}端子の電流もカットされます（STOPモードを解除しても，A/Dコンバータは停止したままです）。

（3）8ビット・データ・レジスタ（ADCR）

連続モード時，逐次比較のA/D変換の結果を格納する8ビットのレジスタです。GET命令により読み出します。単発モード時，8ビット・データ・レジスタの内容が内部のD/Aコンバータでアナログ電圧に変換され，コンパレータがADCn端子から入力されたアナログ信号と大小比較を行うのに使います。PUT命令により値を書き込むことができます。

（4）コンパレータ

コンパレータは，アナログ入力電圧と，D/Aコンバータから出力された電圧を比較します。アナログ入力電圧が高ければ“1”を，低ければ“0”を出力します。比較結果は，連続モード時は8ビット・データ・レジスタ（ADCR）に，単発モード時はADCCMPフラグに格納されます。

（5）A/Dコンバータ制御レジスタ

A/Dコンバータ制御レジスタを図13−14に示します。

図13−14　A/Dコンバータ制御レジスタ（1/2）

RF：21H

	ビット3	ビット2	ビット1	ビット0
	ADCSOFT	0	ADCCMP	ADCEND
リード／ライト	R/W		R	
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

ADCEND	A/D変換の終了
0	初期状態またはA/D変換中
1	連続モードでのA/D変換の終了を示します。A/D変換（連続モード，単発モード）が開始されると自動的にクリア（0）されます。

★

ADCCMP	コンペアの結果（単発モード時のみ有効）
0	アナログ入力電圧が内部のD/Aコンバータの出力電圧より低いとき。
1	アナログ入力電圧が内部のD/Aコンバータの出力電圧より高いとき。

備考1. フラグの内容は，単発モード時，ADCSTRTセット（1）実行後または，ADCRにデータをセット後，3命令目以後有効で，次のADCSTRTセット，ADCRセットまで有効です。

2. 連続モード時，A/D変換値により値は変化しますが，何ビット目の値であるかは特定できません。

ADCSOFT	A/D動作モード選択フラグ
0	連続モード
1	単発モード

図13－14　A/Dコンバータ制御レジスタ（2/2）

RF：20H

	ビット3	ビット2	ビット1	ビット0
	0	0	0	ADCSTRT
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

ADCSTRT	A/D動作スタート
0	初期状態またはA/D変換中
1	A/D変換（連続モード，単発モード）が開始されると自動的にクリア（0）されます。

備考　ADCSTRTは読み出し書き込み可能フラグですが，読み出し時には常に“0”が読まれます。

RF：22H

	ビット3	ビット2	ビット1	ビット0
	0	0	ADCCH1	ADCCH0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

ADCCH1	ADCCH0	アナログ入力チャネル選択
0	0	ADC_0を選択
0	1	ADC_1を選択
1	0	ADC_2を選択
1	1	ADC_3を選択

13.3.3　8ビット・データ・レジスタ（ADCR）への値の設定

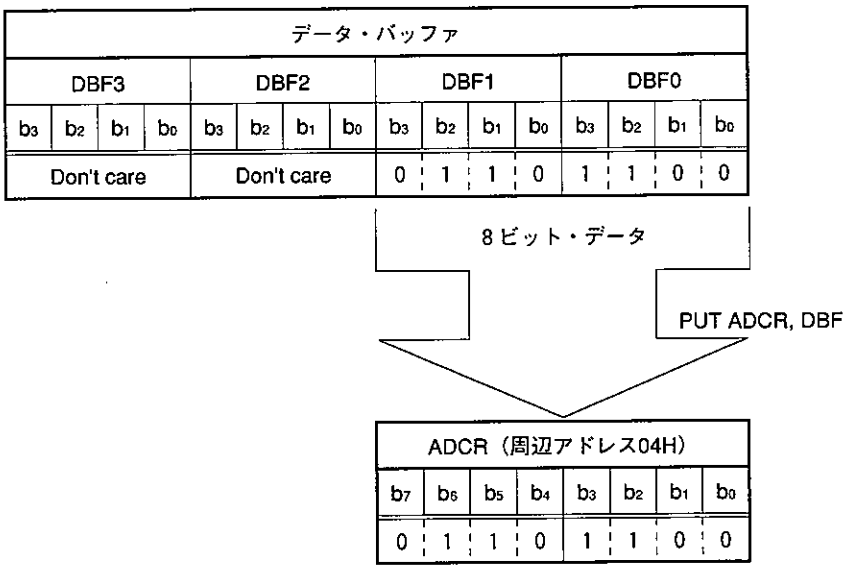
8ビット・データ・レジスタに対する値の設定は，単発モードにおける比較電圧の設定と同様に，PUT命令によりDBF（データ・バッファ）を介して行います。

A/Dコンバータの8ビット・データ・レジスタ（ADCR）の周辺アドレスは，04Hに割り付けられています。ADCRにPUT命令により値を転送する場合，DBFの下位8ビット（DBF1, DBF0）のみが有効です。DBF3およびDBF2の値はADCRには影響を与えません。

図13−15　8ビット・データ・レジスタ（ADCR）への値の設定

ADCRに6CHを設定した例

```
CONTDATL  DAT   CH          ；シンボル定義命令を用いCONTDATLをCHに割り当てる
CONTDATH  DAT   6H          ；シンボル定義命令を用いCONTDATHを6Hに割り当てる
          MOV   DBF0, #CONTDATL；
          MOV   DBF1, #CONTDATH；
          PUT   ADCR, DBF   ；予約語“ADCR” “DBF”を用い転送
```



13.3.4　8ビット・データ・レジスタ（ADCR）の値の読み出し

8ビット・データ・レジスタ（ADCR）の値の読み出しは，GET命令によりDBF（データ・バッファ）を介して行います。

A/Dコンバータの8ビット・データ・レジスタ（ADCR）は，周辺アドレス04Hに割り付けられ，下位8ビット（DBF1, DBF0）のみ有効です。GET命令を実行してもDBFの上位8ビットは影響を受けません。

図13−16　8ビット・データ・レジスタ（ADCR）の値の読み出し

8ビットA/D変換した結果がE2Hのとき

```
GET   DBF, ADCR   ；予約語DBFおよびADCRを使用した例
```

データ・バッファ															
DBF3				DBF2				DBF1				DBF0			
b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
保持				保持				1	1	1	0	0	0	1	0

GET DBF, ADCR

8ビット・データ

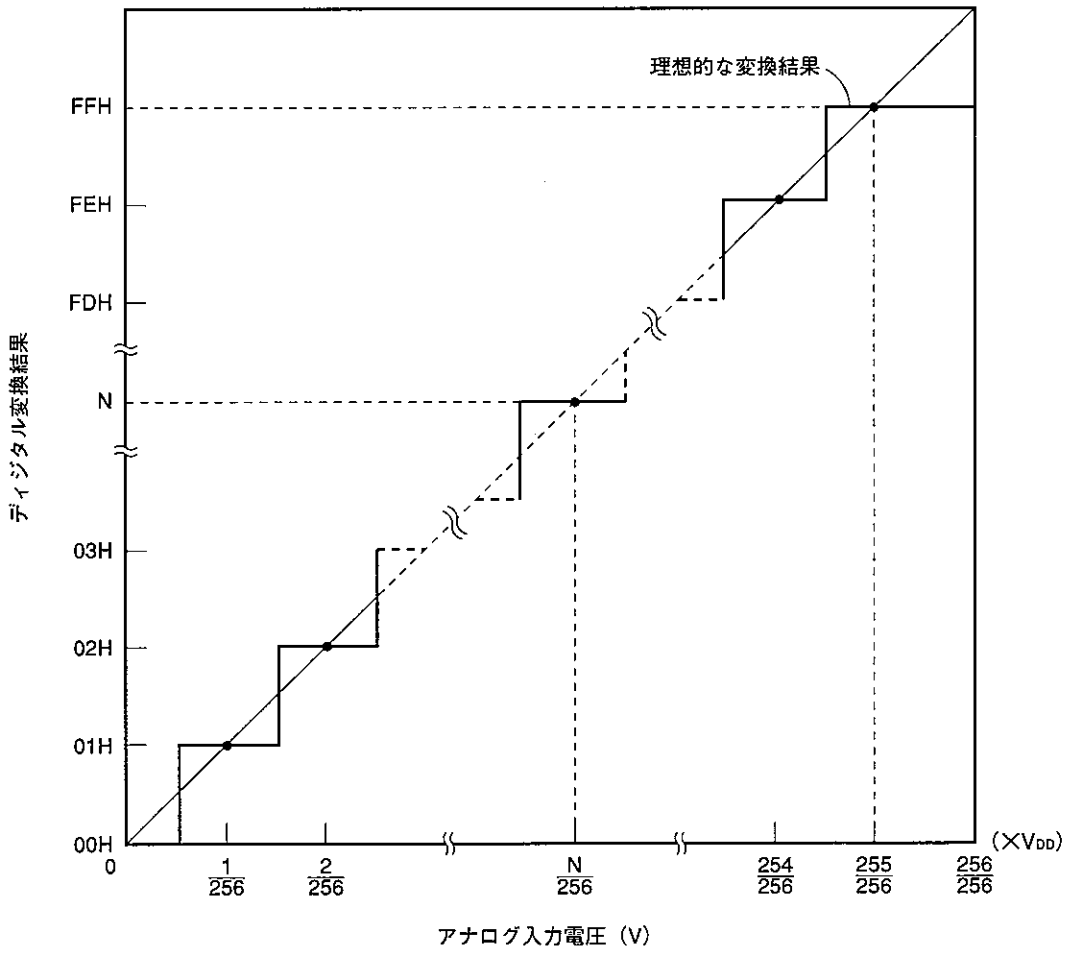
ADCR（周辺アドレス04H）							
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
1	1	1	0	0	0	1	0

13.3.5　A/Dコンバータの動作

A/Dコンバータの動作はADCSOFTフラグの設定により，連続モードと単発モードの２種類のモードに切り替えることができます。

ADCSOFT	A/Dコンバータの動作モード
0	連続モード（A/D変換）
1	単発モード（コンペア動作）

図13−17　アナログ入力電圧とディジタル変換結果との関係



（1）連続モード

（a）連続モードの概要

逐次比較法に基づく8ビットのA/D変換を行い，その変換結果を自動的に8ビット・データ・レジスタ（ADCR）に格納するモードです。

アナログ入力電圧と内部のD/Aコンバータから出力される電圧を，内部コンパレータで比較して，8ビットの上位から順に変換データを得ています。8ビットのA/D変換の完了まで25命令分の時間を必要とします。8ビットA/D変換の完了はADCENDフラグがセット（1）されることを確認することで，判断できます。

（b）連続モードの動作説明

ADCSOFT=0のとき，A/Dコンバータは連続モードに設定されます。

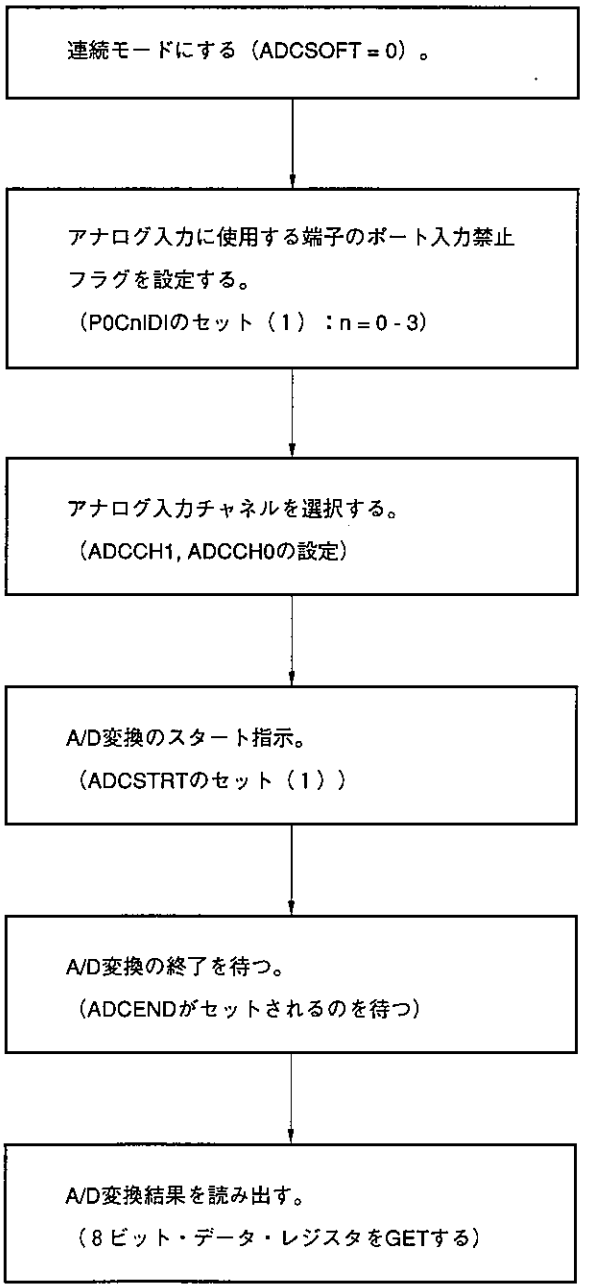
A/D変換を開始する前に，P0CnIDIをセット（1）することによりアナログ入力に使用する端子のポート入力を禁止しておきます。これは，アナログ入力に指定した端子の電圧が中間レベルになった場合，ポートの入力バッファの貫通電流の増加を防止するためです。

その後，ADCCH1，ADCCH0によりアナログ入力信号を選択します。A/D変換の開始は，ADCSTRTフラグをセット（1）することで行います。ADCSTRTフラグはA/D変換開始直後，クリア（0）されます。

A/D変換中は，内部のハードウエアによって8ビットの上位から逐次比較を行います。変換結果は8ビット・データ・レジスタに上位から1ビットずつ格納していきます。変換時間は1ビットあたり3命令分の時間を要します。このため，8ビットの分解能を必要としない場合，命令数から計算し，A/D変換途中のデータをADCENDフラグがセットされる以前に取り出すことも可能です。

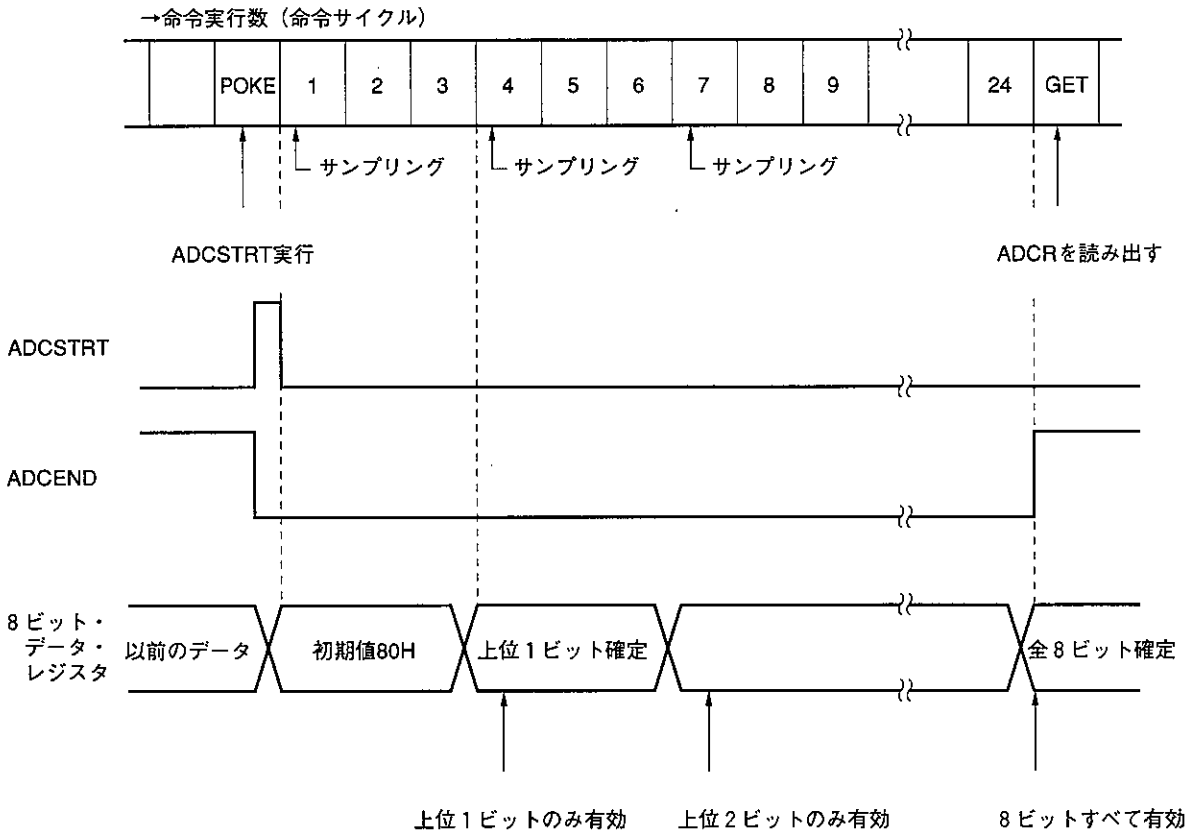
A/D変換の終了は，8ビット・データ・レジスタの最下位ビットにデータを格納したと同時に，ADCENDフラグがセット（1）されることで確認することができます。

図13−18　A/Dコンバータの連続モードの使用方法



（c）連続モード（A/D変換）のタイミング

図13−19　連続モード（A/D変換）のタイミング



注意　1回のA/D変換中に8回のサンプリングを行います。

したがって，A/D変換中にアナログ入力電圧が大きく変化すると，正確なA/D変換が行われません。正確な変換結果を得るためには，A/D変換中のアナログ入力電圧の変化ができるだけ小さくなるようにする必要があります。

1回のサンプリング時間 = 14/f_X（1.75 μs, f_X = 8 MHz時）

サンプリングの繰り返し周期 = 48/f_X（6 μs, f_X = 8 MHz時）

表13－1　A/Dコンバータのデータ変換時間

ADCSTRTをセット（1）から経過した命令実行数[注]	A/D変換が完了したビット（ADCRを読み出したときの有効ビット）
4命令	上位1ビット
7命令	上位2ビット
10命令	上位3ビット
13命令	上位4ビット
16命令	上位5ビット
19命令	上位6ビット
22命令	上位7ビット
25命令	8ビットすべて

注　ADCRからデータを読み出すためのGET命令を含みます。

（2）単発モード

（a）単発モードの概要

8ビット・データ・レジスタ（ADCR）の内容をD/A変換した電圧と，アナログ入力電圧との大小比較を行うモードです。

比較結果はADCCMPフラグに現れます。

（b）単発モードの動作説明

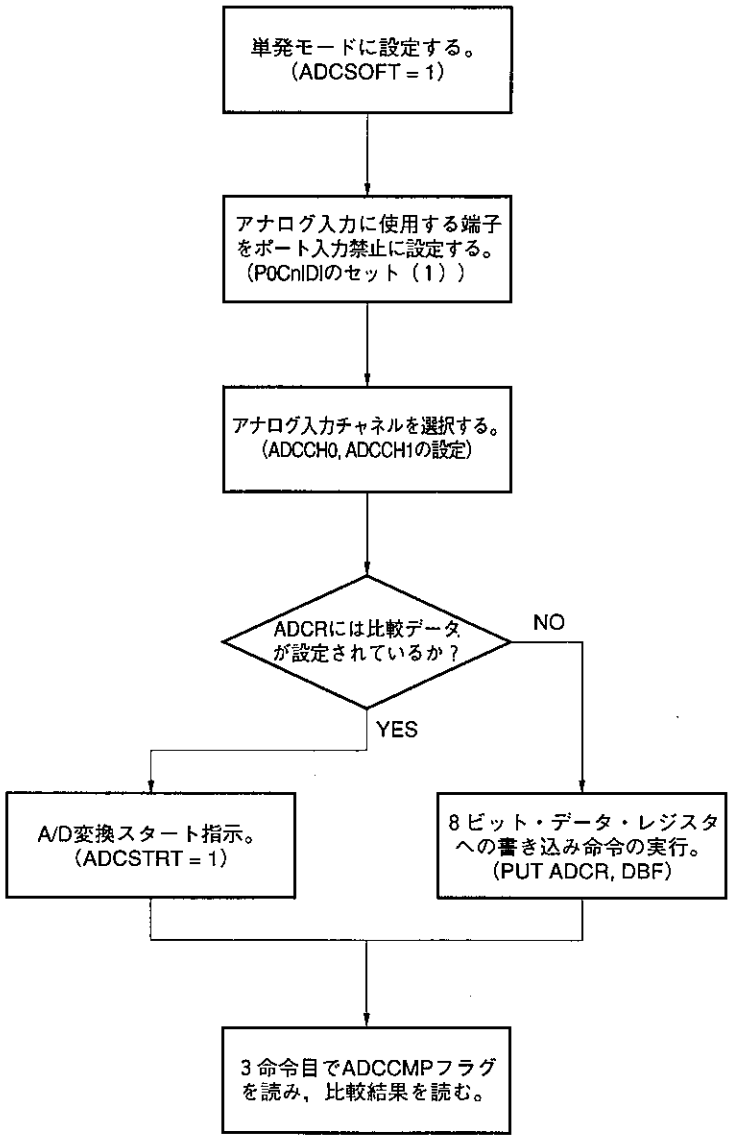
ADCSOFT＝1のとき，A/Dコンバータは単発モードに設定されます。

単発モードによる動作を開始する前にP0CnIDIをセット（1）することにより，アナログ入力に使用する端子のポート入力を禁止しておきます（連続モードの場合と同様の理由です）。その後，ADCCH1, ADCCH0によりアナログ入力信号を選択します。

単発モードの動作開始は，ADCSOFT＝1のとき8ビット・データ・レジスタ（ADCR）への書き込み命令の実行（PUT ADCR, DBF），またはADCSTRTフラグのセット（1）で行います。ADCSTRTフラグのセットにより動作を開始させる場合は，8ビット・データ・レジスタ（ADCR）にD/A変換するデータを格納したあとに行ってください。

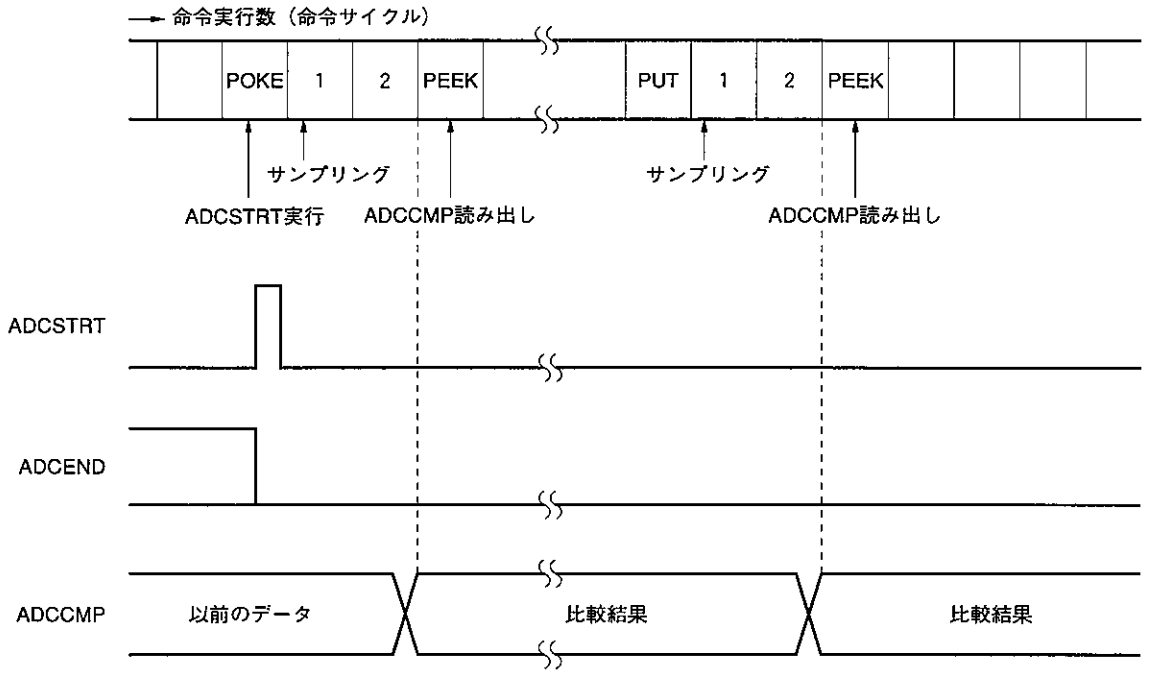
単発モードの比較結果は，PUT命令により8ビット・データ・レジスタ（ADCR）に対し，書き込み命令を実行後，3命令目で比較結果がADCCMPに現れます。なお，このときのADCENDフラグは無効となります。

図13−20　A/Dコンバータの単発モードの使用方法



（c）単発モード（コンペア動作）のタイミング

図13－21　単発モード（コンペア動作）のタイミング



単発モードではADCSTRTに1を書き込み後（POKE命令の実行），3命令目以後にADCCMPに値が格納され，PEEK命令で比較結果を読み出すことができます。また，ADCRにデータをセット（PUT命令の実行）しても，ADCSTRTと同様に比較を開始し，3命令目以後に比較結果を読み出すことができます。

ADCCMPフラグは，リセット，ADCRへの書き込み命令の実行でクリア（0）されます。

注意　ADCRに値を設定する前に，必ずADCSOFT＝1にしておいてください。ADCSOFT＝0の場合にはADCRに値を設定できません（“PUT ADCR, DBF”命令は無効になります）。

1回のサンプリング時間＝14/fx（1.75 μs，fx＝8 MHz時）

13.4　シリアル・インタフェース（SIO）

μPD17149のシリアル・インタフェースは，8ビットのシフト・レジスタ（SIOSFR），シリアル・モード・レジスタ，シリアル・クロック・カウンタで構成され，シリアル・データの入出力に使用します。

13.4.1　シリアル・インタフェースの機能

シリアル・クロック入力端子（$\overline{SCK}$），シリアル・データ出力端子（SO），シリアル・データ入力端子（SI）の3線式で，クロック同期の8ビット送受信動作が可能なシリアル・インタフェースです。μPD7500シリーズや75Xシリーズで用いられている方式とコンパチブルなモードで各種周辺I/Oデバイスと接続が可能です。

（1）シリアル・クロック

内部クロック3種類，外部クロック1種類の合計4種類選択することができます。シリアル・クロックに内部クロックを選択した場合には，P0D$_0$/$\overline{SCK}$端子にそのクロックを自動的に出力します。

★ 表13－2　シリアル・クロック一覧

SIOCK1	SIOCK0	選択されるシリアル・クロック
0	0	$\overline{SCK}$端子からの外部クロック
0	1	fx/16
1	0	fx/128
1	1	fx/1024

fx：システム・クロック発振周波数

（2）転送動作

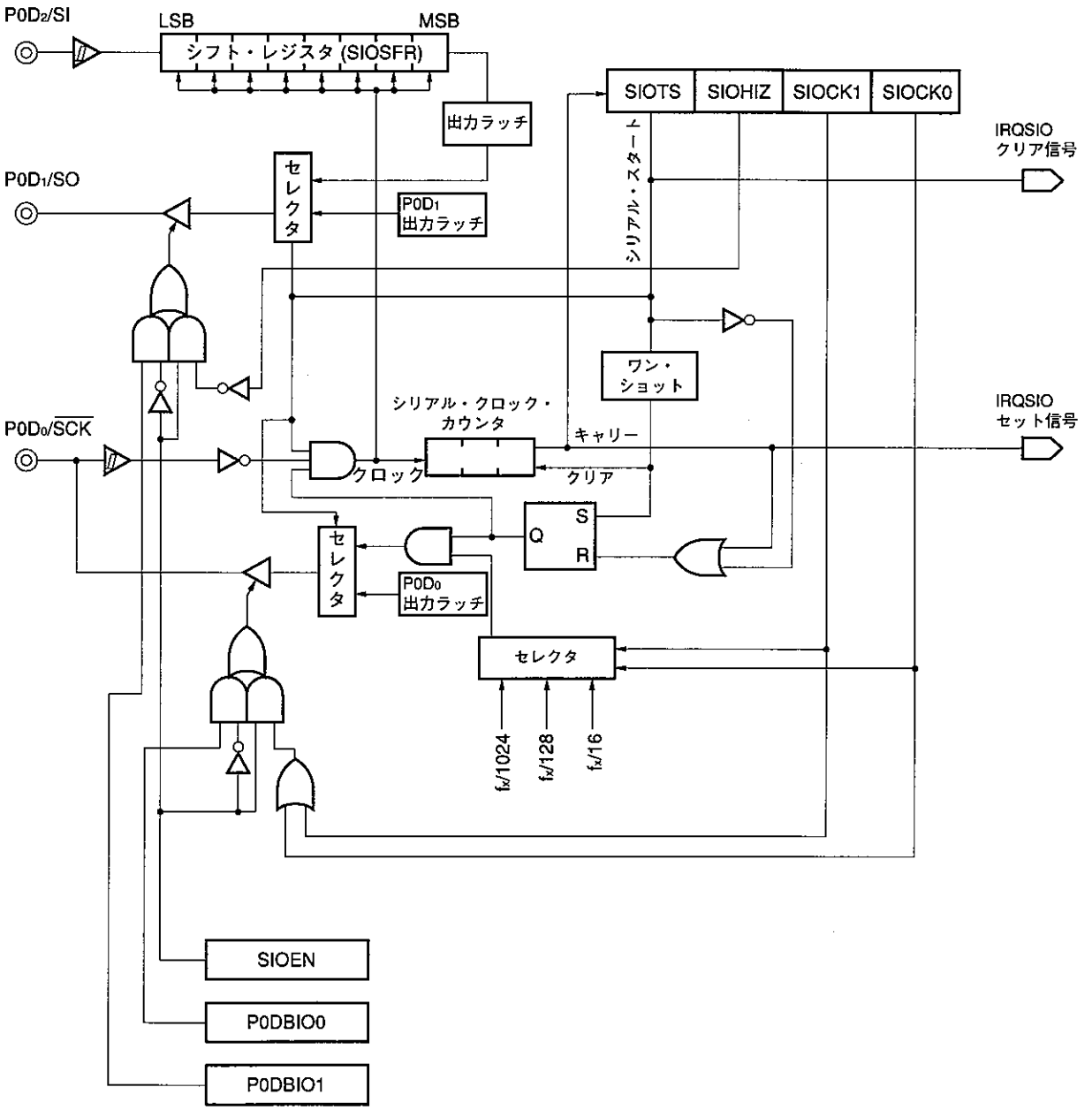
SIOENをセット（1）することにより，ポート0D（P0D$_0$/$\overline{SCK}$,P0D$_1$/SO,P0D$_2$/SI）の各端子は，シリアル・インタフェース用の端子として機能します。このとき，SIOTSをセット（1）すれば，外部クロックまたは内部クロックの立ち下がりに同期して動作を開始します。なお，SIOTSをセットすると，IRQSIOは自動的にクリアされます。

シリアル・クロックの立ち下がりに同期して，シフト・レジスタの最上位ビットから転送を開始し，シリアル・クロックの立ち上がりに同期してSI端子の情報を最下位ビットからシフト・レジスタに格納します。

8ビットのデータ転送が終了すれば，自動的にSIOTSはクリアされ，IRQSIOがセットされます。

備考　シリアル転送を行う際，シフト・レジスタの内容の最上位ビットからのみ，転送を開始します。最下位ビットからの転送は行えません。シリアル・クロックの立ち上がりに同期して，常にSI端子の状態はシフト・レジスタに取り込まれます。

図13−22　シリアル・インタフェースのブロック図



注意　シフト・レジスタの出力ラッチは，P0D1の出力ラッチと独立しています。したがって，P0D1に対して出力命令を実行しても，シフト・レジスタの出力ラッチの状態は変化しません。シフト・レジスタの出力ラッチは，リセットにより“0”にクリアされ，その後は，前回の転送データのLSBの状態を保持しています。

13.4.2　3線式シリアル・インタフェースの動作モード

シリアル・インタフェースは，2つのモードを選択することができます。シリアル・インタフェース機能を選択した場合，シリアル・クロックに同期して，P0D2/SI端子は常にデータを取り込みます。

● 8 ビット送受信モード（同時送受信）

● 8 ビット受信モード（SO端子：ハイ・インピーダンス状態）

表13－3　シリアル・インタフェースの動作モード

SIOEN	SIOHIZ	$P0D_0$/SI 端子	$P0D_1$/SO 端子	シリアル・インタフェース動作モード
1	0	SI	SO	8 ビット送受信モード
1	1	SI	$P0D_1$（入力）	8 ビット受信モード
0	×	$P0D_0$（入出力）	$P0D_1$（入出力）	汎用ポート・モード

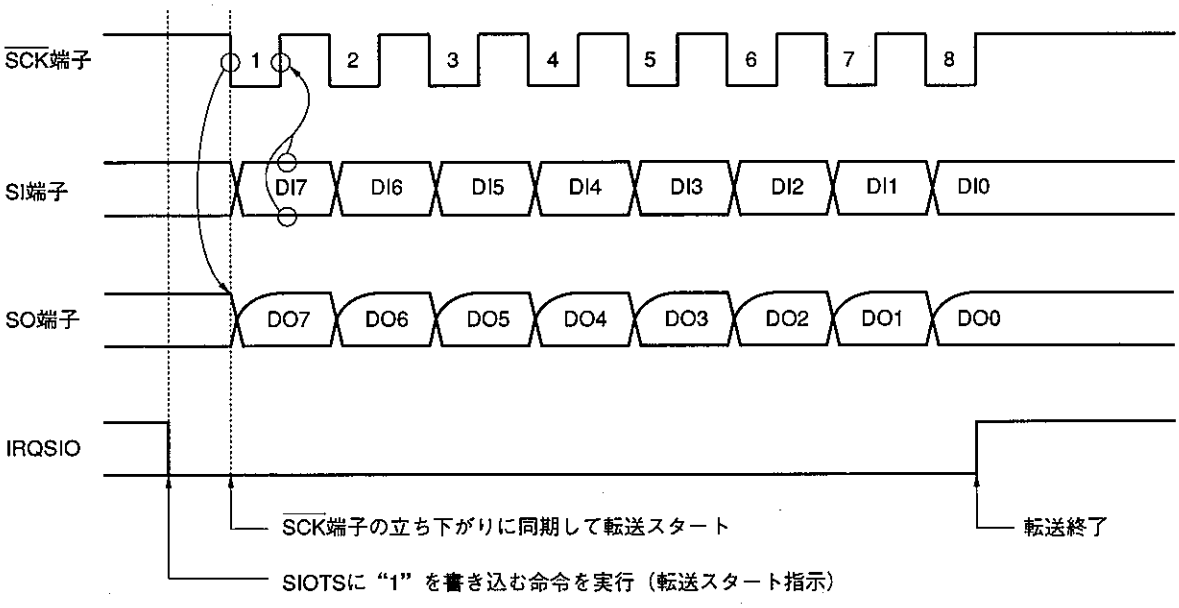
×：Don't care

（1）8 ビット送受信モード（同時送受信）

シリアル・データの入出力はシリアル・クロックによって制御されます。シリアル・クロック（$\overline{SCK}$ 端子信号）の立ち下がりでシフト・レジスタのMSB がSOラインによって出力され，立ち上がりでシフト・レジスタの内容が 1 ビット・シフトされると同時に，SIライン上のデータがシフト・レジスタのLSB にロードされます。

シリアル・クロック・カウンタ（3 ビット・カウンタ）はシリアル・クロックを 8 カウントするごとに割り込み要求フラグをセットします（IRQSIO ← 1）。

図13－23　8 ビット送受信モード（同時送受信）のタイミング



備考　DI：シリアル・データの入力

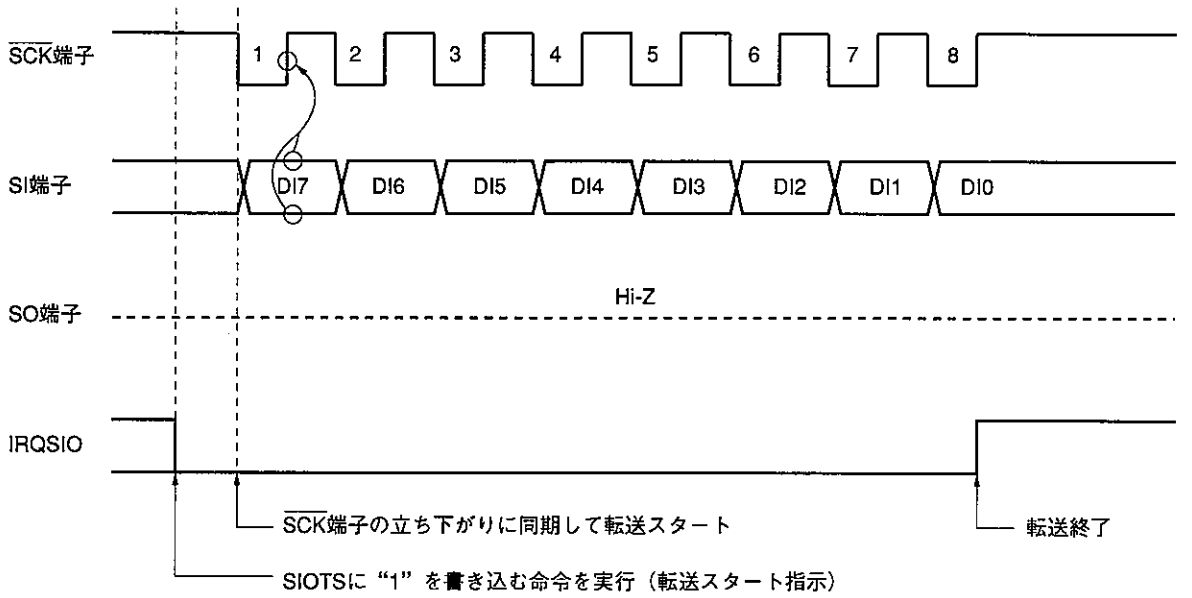
DO：シリアル・データの出力

（2）クロック同期8ビット受信モード（SO端子：ハイ・インピーダンス状態）

SIOHIZ＝1のとき，P0D1/SO端子はハイ・インピーダンス状態になります。このときSIOTSに“1”を書き込んでシリアル・クロックの供給を開始すると，シリアル・インタフェースは受信機能だけが動作します。

また，P0D1/SO端子はハイ・インピーダンス状態になっていますので，入力ポート（P0D1）として使用することができます。

図13－24　8ビット受信モードのタイミング



備考　DI：シリアル・データの入力

（3）動作停止モード

SIOTS（RF：02H番地，ビット3）の値が0のときは，シリアル・インタフェースは動作停止モードに設定されます。このモードではシリアル転送は行われません。

この動作ではシフト・レジスタはシフト動作を行いませんので，通常の8ビット・レジスタとして利用可能です。

図13－25　シリアル・インタフェースの制御用レジスタ（1/2）

RF：02H

	ビット3	ビット2	ビット1	ビット0
	SIOTS	SIOHIZ	SIOCK1	SIOCK0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

SIOCK1	SIOCK0	シリアル・クロックの選択
0	0	外部クロック（$\overline{SCK}$端子）
0	1	fx／16
1	0	fx／128
1	1	fx／1024

SIOHIZ	P0D1/SO端子の機能選択
0	シリアル・データ出力（SO端子）
1	入力ポート（P0D1端子）

★

SIOTS	シリアル転送の開始と停止
0	シリアル転送の強制的停止（途中のビットからの再開は不可）。
1	シリアル転送の動作開始。 ・内部クロック選択時 システム・クロックの内部分周信号をシリアル・クロックとして動作開始。 ・外部クロック選択時 $\overline{SCK}$端子の立ち下がりに同期して動作開始。

★

備考　シリアル転送が完了するとSIOTSは自動的にクリア（0）されます。

図13－25　シリアル・インタフェースの制御用レジスタ（2/2）

RF：0BH

	ビット3	ビット2	ビット1	ビット0
	TM1OSEL	0	0	SIOEN
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

SIOEN	SIOの動作許可
0	$P0D_0/\overline{SCK}$, $P0D_1$/SO, $P0D_2$/SIの各端子はポートとして機能
1	$P0D_0/\overline{SCK}$, $P0D_1$/SO, $P0D_2$/SIの各端子はシリアル・インタフェースとして機能

備考　**第12章**　ポートも参照してください。

TM1OSEL	$P0D_3/\overline{TM1OUT}$端子の機能選択
0	$P0D_3/\overline{TM1OUT}$端子はポートとして機能
1	$P0D_3/\overline{TM1OUT}$端子はタイマ1の出力として機能

注意　シリアル・インタフェースとは直接関係ありません。

13.4.3　シフト・レジスタへの値の設定

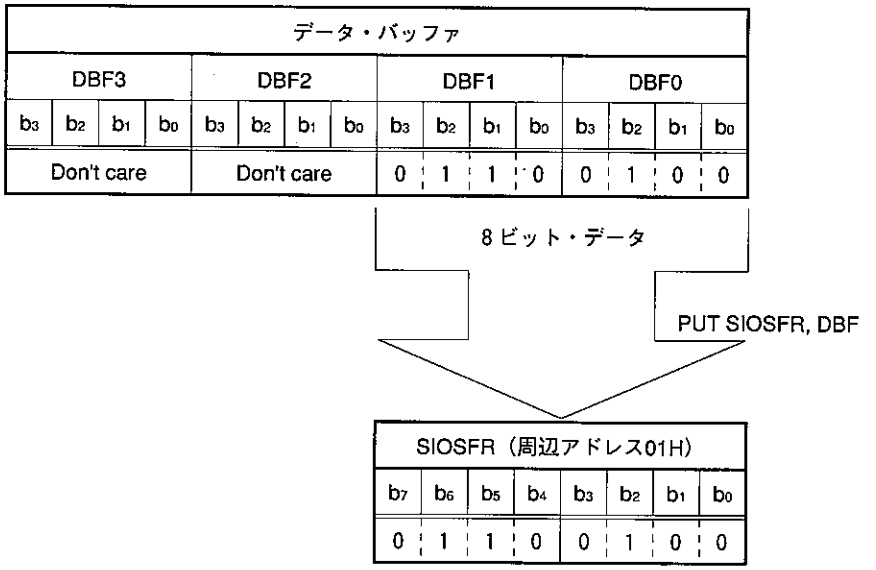
シフト・レジスタに対する値の設定は，PUT命令によりDBF（データ・バッファ）を介して行います。

シフト・レジスタの周辺アドレスは，01Hに割り付けられています。PUT命令によりシフト・レジスタに値を転送する場合，DBFの下位8ビット（DBF1, DBF0）だけが有効です。DBF3およびDBF2の値は，シフト・レジスタに影響を与えません。

図13−26　シフト・レジスタへの値の設定

シフト・レジスタに値64Hを設定した例

```
SIODATL    DAT   4H              ；シンボル定義命令を用いSIODATLを4Hに割り当てる
SIODATH    DAT   6H              ；シンボル定義命令を用いSIODATHを6Hに割り当てる
           MOV   DBF0, #SIODATL  ；
           MOV   DBF1, #SIODATH  ；
           PUT   SIOSFR, DBF     ；予約語“SIOSFR”を用い転送。
```

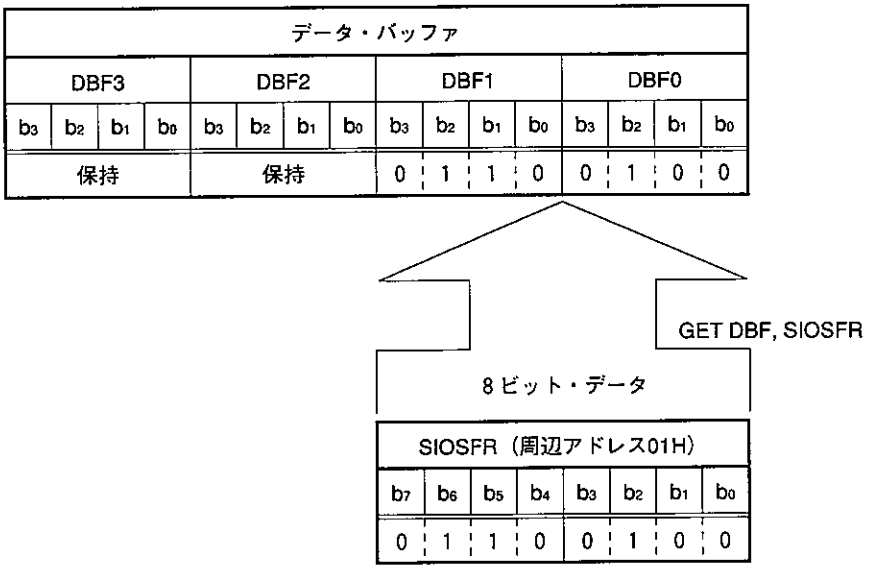


13.4.4　シフト・レジスタの値の読み出し

シフト・レジスタの値の読み出しは，GET命令によりDBF（データ・バッファ）を介して読み出します。シフト・レジスタは周辺アドレス01Hに割り付けられ，下位8ビット（DBF1, DBF0）のみ有効です。GET命令を実行してもDBFの上位8ビットには影響を与えません。

図13－27　シフト・レジスタの値の読み出し

GET DBF, SIOSFR；予約語DBFおよびSIOSFRを使用した例



〔メ　モ〕

第14章　割り込み機能

μPD17149には，４本の内部割り込み機能と１本の外部割り込み機能があり，多彩な応用が可能です。

また，この製品の割り込み制御回路には次のような特色があり，非常に高速な割り込み処理が可能となります。

（ａ）割り込みマスタ許可フラグ（INTE）と割り込み許可フラグ（IP××）により受け付けの可否を制御可能

（ｂ）割り込み要求フラグ（IRQ××）のテスト&クリア可能（ソフトウエアで割り込み発生の確認可能）

（ｃ）３レベルまでの多重割り込みが可能

（ｄ）割り込み要求によるスタンバイ・モード（STOP, HALT）の解除可能（割り込み許可フラグによる解除条件の選択可能）

注意　割り込み処理において，ハードウエアにより自動的にスタックに退避されるのは，BCD, CMP, CY, Z, IXEの各フラグのみで，最大３レベルまでです。また，割り込み処理の内容において，周辺ハードウエア（タイマ，A/Dコンバータなど）をアクセスする場合には，DBF, WRの内容はハードウエアでは退避されません。したがって，割り込み処理の最初にDBFおよびWRをソフトウエアによりRAM上に退避し，割り込み処理終了直前に退避した内容をもとに戻すことをおすすめします。

14.1　割り込み要因とベクタ・アドレス

μPD17149の割り込みはすべて，割り込みが受け付けられると，割り込み要因に対応するベクタ・アドレスへ分岐するベクタ割り込み方式となっています。割り込み要因とベクタ・アドレスは，表14－1のようになっています。

なお，複数の割り込み要求が同時に発生した場合や，保留された複数の割り込み要求が一斉に許可された場合は，表14－1の優先順位に従い，処理します。

表14－1　割り込み要因の種類

割り込み要因	優先順位	ベクタ・アドレス	IRQフラグ	IPフラグ	IEGフラグ	内部/外部	備　考
INT端子 （RF：0FH，ビット0）	1	0005H	IRQ RF：3FH， ビット0	IP RF：2FH， ビット0	IEGMD0,1 RF：1FH	外部	立ち上がり，立ち下がり，両エッジ選択可能
タイマ0	2	0004H	IRQTM0 RF：3EH， ビット0	IPTM0 RF：2FH， ビット1	—	内部	
タイマ1	3	0003H	IRQTM1 RF：3DH， ビット0	IPTM1 RF：2FH， ビット2	—	内部	
ベーシック・インターバル・タイマ	4	0002H	IRQBTM RF：3CH， ビット0	IPBTM RF：2FH， ビット3	—	内部	
シリアル・インタフェース	5	0001H	IRQSIO RF：3BH， ビット0	IPSIO RF：2EH， ビット0	—	内部	

14.2　割り込み制御回路の各種ハードウエア

次に，割り込み制御回路の各フラグについて説明します。

（1）割り込み要求フラグ，割り込み許可フラグ

割り込み要求フラグ（IRQ×××）は，割り込み要求発生でセット（1）され，割り込み処理が実行されると自動的にクリア（0）されます。

割り込み許可フラグ（IP×××）は，各割り込み要求フラグに対応して個別に備わっており，内容が“1”のとき割り込みを許可し，“0”のとき禁止します。

（2）EI/DI命令

受け付けた割り込みを実行するかどうかは，EI/DI命令によって指定します。

EI命令を実行すると，割り込みを受け付け可能とするINTE（インタラプト・イネーブル・フラグ）がセット（1）されます（割り込みが受け付けられるとINTEはクリア（0）されます）。INTEフラグは，レジスタ・ファイル上には登録されていません。このため，命令等により，フラグの状態を確認することはできません。

DI命令はINTEフラグを“0”にクリアして，すべての割り込みを禁止します。

また，リセット時にもINTEフラグはクリア（0）され，すべての割り込みは禁止状態になります。

表14－2　割り込み要求フラグと割り込み許可フラグ

割り込み要求フラグ	割り込み要求フラグのセット信号	割り込み許可フラグ
IRQ	INT端子入力信号のエッジ検出によりセット。検出エッジはIEGMD0, IEGMD1フラグにより選択。	IP
IRQTM0	タイマ0からの一致信号でセット。	IPTM0
IRQTM1	タイマ1からの一致信号でセット。	IPTM1
IRQBTM	ベーシック・インターバル・タイマからのオーバフロー（基準時間間隔信号）でセット。	IPBTM
IRQSIO	シリアル・インタフェースのシリアル・データ転送動作終了信号によりセット。	IPSIO

図14－1　割り込み制御用レジスタ（1/7）

RF：0FH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	INT
リード／ライト	R			
★ リセット時の初期値	0	0	0	注

リード＝R，ライト＝W

★ ★ INT	INT端子の状態
0	PEEK命令実行時，ノイズ除去回路を通したINT信号の論理状態が“0”
1	PEEK命令実行時，ノイズ除去回路を通したINT信号の論理状態が“1”

注　INTフラグはラッチされていないため端子の論理状態に応じて刻々と変化します。
ただし，IRQフラグは一度セットされると割り込みが受け付けられるまでセットされたままです。
また，0FH番地へのPOKE命令は無効です。

RF：1FH

	ビット3	ビット2	ビット1	ビット0
	0	0	IEGMD1	IEGMD0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

IEGMD1	IEGMD0	INT端子の割り込み検出エッジの選択
0	0	立ち上がりエッジ割り込み
0	1	立ち下がりエッジ割り込み
1	0	両エッジ割り込み
1	1	

図14－1　割り込み制御用レジスタ（2/7）

RF：3FH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IRQ
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

読み出し時

IRQ	INT端子割り込み要求
0	INT端子からの割り込み要求なし，または，INT端子による割り込み処理中
1	INT端子からの割り込み要求発生，または，INT端子からの割り込み保留中

書き込み時

IRQ	INT端子割り込み要求
0	INT端子からの割り込み要求の強制的解除
1	INT端子からの割り込み要求の強制的発生

RF：3EH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IRQTM0
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

読み出し時

IRQTM0	TM0割り込み要求
0	タイマ0からの割り込み要求なし，または，タイマ0の割り込み処理中
1	タイマ0のカウント・レジスタとモジュロ・レジスタの内容が一致し割り込み要求が発生，またはタイマ0の割り込み要求を保留中

書き込み時

IRQTM0	TM0割り込み要求
0	タイマ0からの割り込み要求の強制的解除
1	タイマ0からの割り込み要求の強制的発生

備考　TM0RESをセット（1）すると，IRQTM0もクリア（0）されます。

図14－1　割り込み制御用レジスタ（3/7）

RF：3DH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IRQTM1
リード／ライト	R/W			
リセット時の初期値	0	0	0	1

リード＝R，ライト＝W

読み出し時

IRQTM1	TM1割り込み要求
0	タイマ1からの割り込み要求なし，または，タイマ1の割り込み処理中
1	タイマ1のカウント・レジスタとモジュロ・レジスタの内容が一致し割り込み要求が発生，または，タイマ1の割り込み要求を保留中

書き込み時

IRQTM1	TM1割り込み要求
0	タイマ1からの割り込み要求の強制的解除
1	タイマ1からの割り込み要求の強制的発生

備考　TM1RESをセット（1）すると，IRQTM1もクリア（0）されます。また，STOP命令を実行した直後，IRQTM1はクリア（0）されます。

図14－1　割り込み制御用レジスタ（4/7）

RF：3CH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IRQBTM
リード／ライト	R/W			
リセット時の初期値	0	0	0	1

リード＝R，ライト＝W

★

読み出し時

IRQBTM	BTM割り込み要求
0	ベーシック・インターバル・タイマからの割り込み要求なし，または，ベーシック・インターバル・タイマの割り込みを処理中
1	ベーシック・インターバル・タイマがオーバフローし，割り込み要求が発生，または，ベーシック・インターバル・タイマからの割り込み要求を保留中

書き込み時

IRQBTM	BTM割り込み要求
0	ベーシック・インターバル・タイマからの割り込み要求を強制的解除
1	ベーシック・インターバル・タイマからの割り込み要求の強制的発生

備考　BTMRESをセット（1）するとIRQBTMもクリア（0）されます。

図14－1　割り込み制御用レジスタ（5/7）

RF：3BH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IRQSIO
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

読み出し時

IRQSIO	SIO割り込み要求
0	シリアル・インタフェースからの割り込み要求なし，または，シリアル・インタフェースからの割り込みを処理中
1	シリアル・インタフェースが転送を完了し，割り込み要求が発生，またはシリアル・インタフェースの割り込み要求を保留中

書き込み時

IRQSIO	SIO割り込み要求
0	シリアル・インタフェースからの割り込み要求を強制的解除
1	シリアル・インタフェースからの割り込み要求の強制的発生

図14－1　割り込み制御用レジスタ（6/7）

RF：2FH

	ビット3	ビット2	ビット1	ビット0
	IPBTM	IPTM1	IPTM0	IP
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

IP	INT端子割り込み許可
0	INT端子からの割り込みを禁止 IRQフラグがセット（1）されても，その割り込みを保留
1	INT端子からの割り込みを許可 EI状態でIRQフラグがセット（1）されると，その割り込み処理を実行

IPTM0	TM0割り込み許可
0	タイマ0からの割り込みを禁止 IRQTM0フラグがセット（1）されても，その割り込みを保留
1	タイマ0からの割り込みを許可 EI状態でIRQTM0フラグがセット（1）されると，その割り込み処理を実行

IPTM1	TM1割り込み許可
0	タイマ1からの割り込みを禁止 IRQTM1フラグがセット（1）されても，その割り込みを保留
1	タイマ1からの割り込みを許可 EI状態でIRQTM1フラグがセット（1）されると，その割り込み処理を実行

IPBTM	BTM割り込み許可
0	ベーシック・インターバル・タイマからの割り込みを禁止 IRQBTMフラグがセット（1）されても，その割り込みを保留
1	ベーシック・インターバル・タイマからの割り込みを許可 EI状態でIRQBTMフラグがセット（1）されると，その割り込み処理を実行

図14－1　割り込み制御用レジスタ（7/7）

RF：2EH

	ビット3	ビット2	ビット1	ビット0
	0	0	0	IPSIO
リード／ライト	R/W			
リセット時の初期値	0	0	0	0

リード＝R，ライト＝W

IPSIO	SIO割り込み許可
0	シリアル・インタフェースからの割り込みを禁止 IRQSIOフラグがセット（1）されても，その割り込みを保留
1	シリアル・インタフェースからの割り込みを許可 EI状態でIRQSIOフラグがセット（1）されると，その割り込み処理を実行

14.3　割り込みシーケンス

14.3.1　割り込みの受け付け

割り込みが受け付けられると，その時点で実行されていた命令の命令サイクル終了後，割り込み処理が開始され，ベクタ・アドレスにプログラムの流れが移ります。ただし，MOVT命令，EI命令およびスキップ条件を満たした命令中の割り込みは2命令サイクル終了後に処理を開始します。

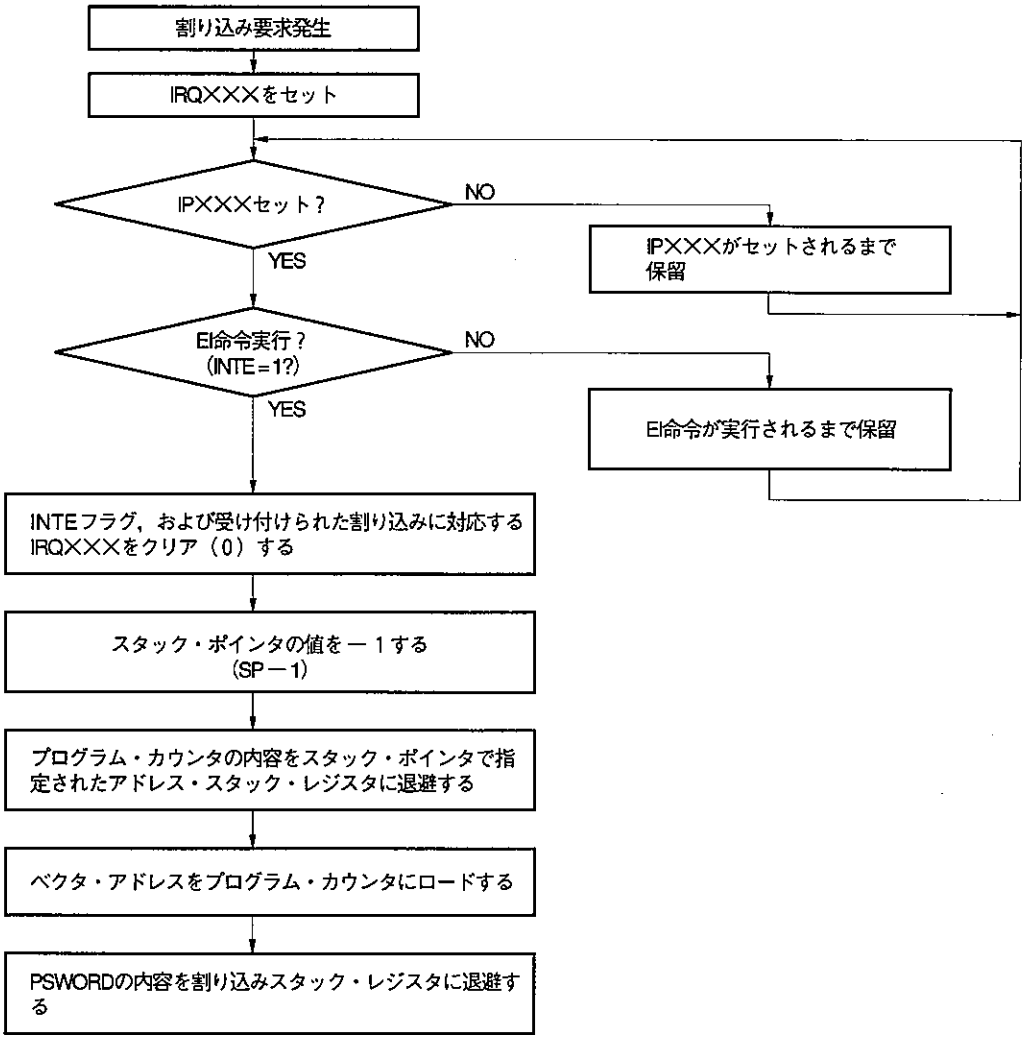
割り込みが受け付けられるとINTEフラグはクリア（0）されます。また，プログラムの戻り番地を記憶するためアドレス・スタック・レジスタを1レベル消費し，さらにシステム・レジスタ中のPSWORDを退避するため割り込みスタック・レジスタを1レベル消費します。

複数の割り込みが同時に発生または許可される状態になったときは，優先順位の高い順に割り込み処理が実行されます。優先順位の高い割り込みが処理されるまで，優先順位の低い割り込みは保留されます。

なお，優先順位は表14－1　割り込み要因の種類を参照してください。

注意　PSWORDは，割り込みスタック・レジスタに退避後，自動的に00000Bにリセットされます。

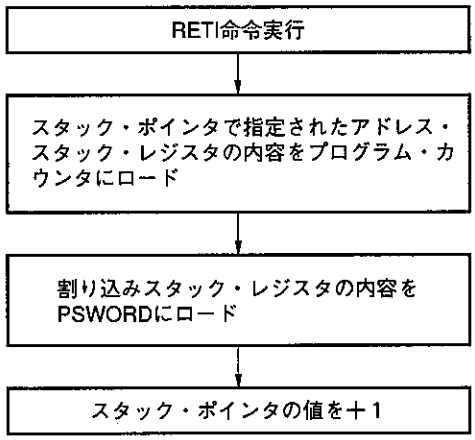
図14－2　割り込み処理手順



14.3.2　割り込みルーチンからの復帰

割り込み処理ルーチンから復帰するときは，RETI命令を実行します。このRETI命令サイクル中に以下の処理が行われます。

図14－3　割り込み処理からの復帰

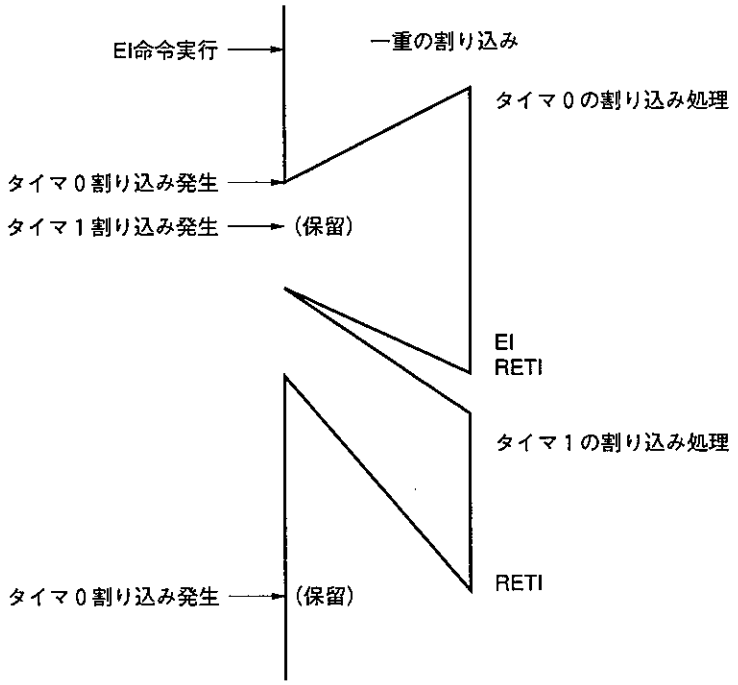


注意1．RETI命令ではINTEフラグはセットされません。

ある割り込み処理を終了後，続けて保留されている割り込みを処理したい場合は，RETI命令の直前にEI命令を実行し，INTEフラグをセット（1）してください。

2．EI命令に続けてRETI命令を実行した場合は，EI命令とRETI命令の間で割り込みが受け付けられることはありません。これはEI命令がINTEフラグをセット（1）するタイミングは次に続く命令の実行が完了したあとになる仕組みになっているためです。

例


EI命令実行
一重の割り込み
タイマ０の割り込み処理
タイマ０割り込み発生
タイマ１割り込み発生
（保留）
EI
RETI
タイマ１の割り込み処理
RETI
タイマ０割り込み発生
（保留）

14.3.3　割り込み受け付けタイミング

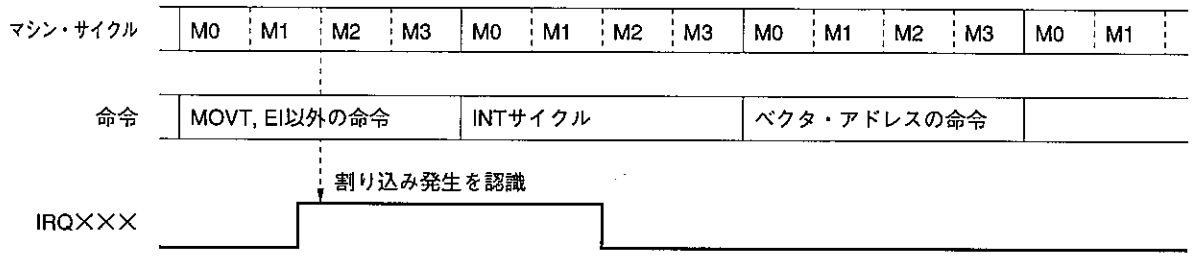
割り込み受け付けタイミング・チャートを図14－4に示します。

μPD17149は，16クロックで1命令を実行し，これを1命令サイクルとします。1命令サイクルは，4クロック単位でM0 - M3に細分化されます。

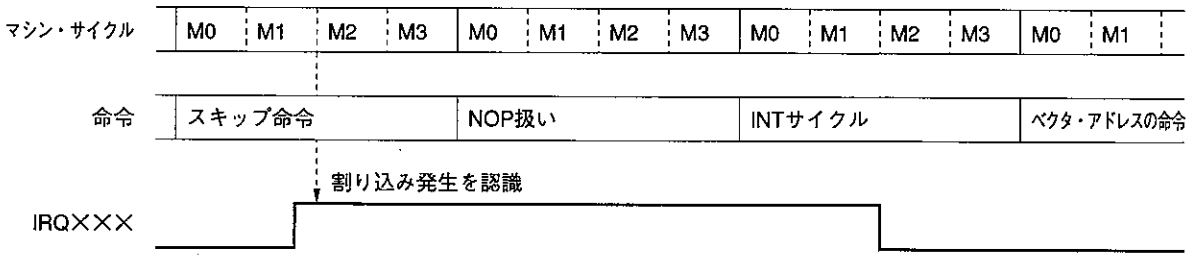
プログラムが割り込みの発生を認識するタイミングは，M2の前のエッジです。

図14－4　割り込み受け付けタイミング・チャート（INTE＝1, IP××＝1のとき）（1/3）

① MOVT, EI以外の命令のM2以前に割り込みが発生した場合



② ①でスキップ命令のスキップ条件が成立した場合



③ MOVT, EI以外の命令のM2以降に割り込みが発生した場合

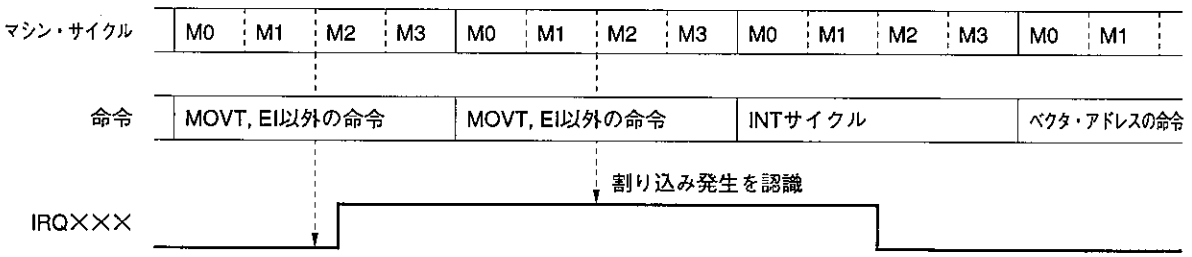
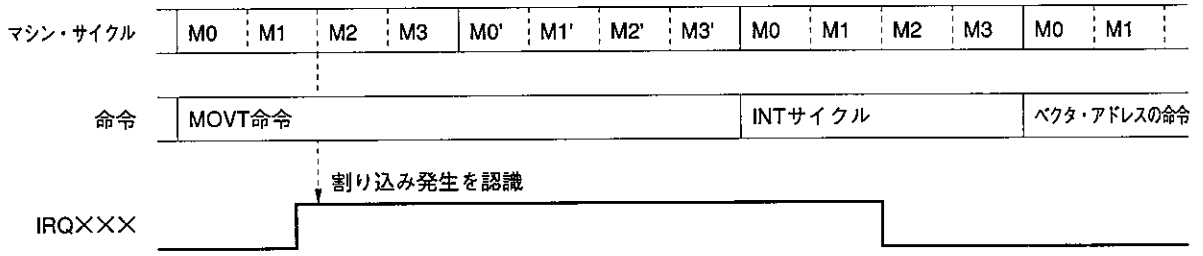
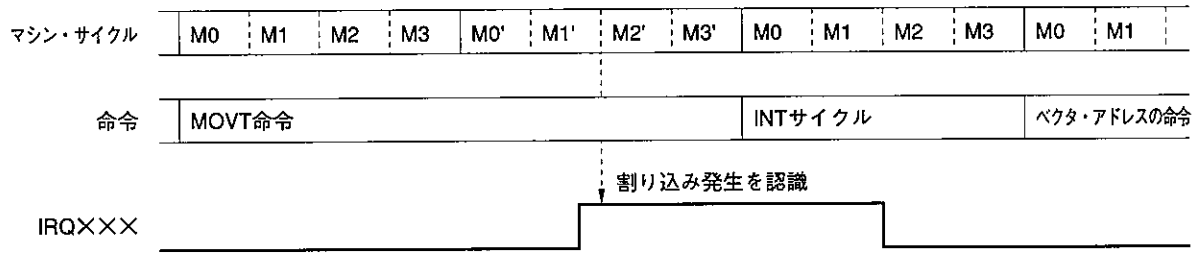


図14－4　割り込み受け付けタイミング・チャート（INTE＝1, IP××＝1のとき）（2/3）

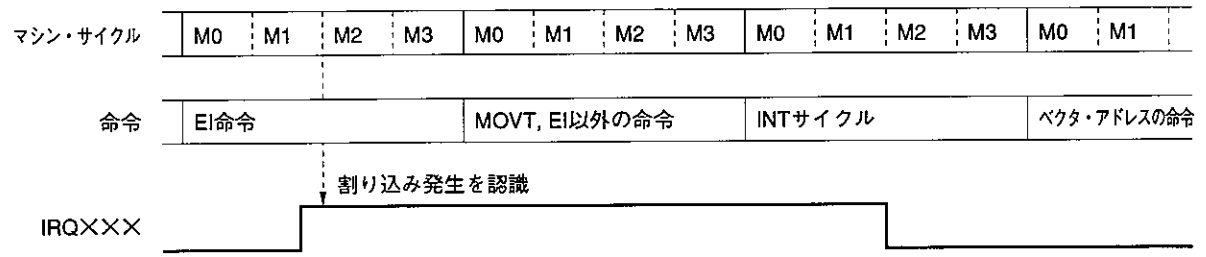
④　MOVT命令のM2以前に割り込みが発生した場合



⑤　MOVT命令のM2' 以前に割り込みが発生した場合



⑥　EI命令のM2以前に割り込みが発生した場合



⑦　EI命令のM2以降に割り込みが発生した場合

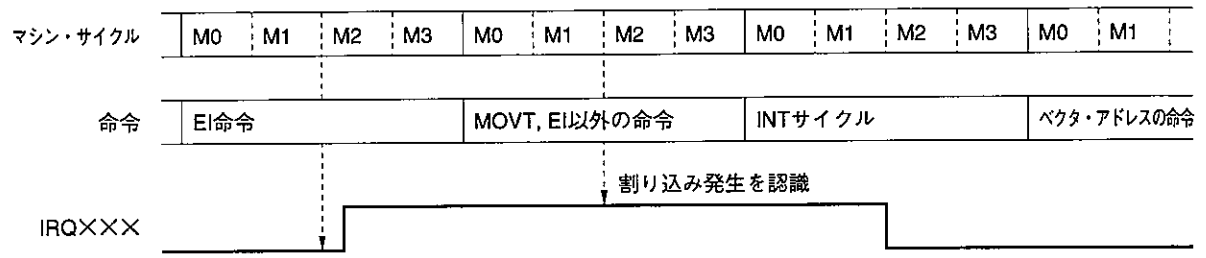
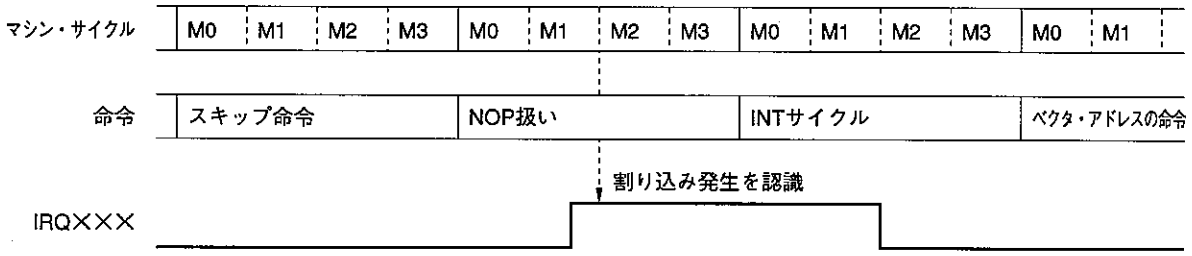


図14－4　割り込み受け付けタイミング・チャート（INTE＝1, IP××＝1のとき）（3/3）

⑧　スキップ命令によるスキップ中（NOP扱い）に割り込みが発生した場合



備考1．INTサイクルは割り込みの準備のためのサイクルです。このサイクル中に，PCとPSWORDの退避やIRQ××のクリアを行います。

2．MOVT命令の実行には例外的に2命令サイクルを必要とします。

3．EI命令は割り込み処理からの復帰時に多重割り込みが発生しないように考慮されています。

第15章　スタンバイ機能

15.1　スタンバイ機能の概要

μPD17149は，スタンバイ機能を利用することにより，消費電流を低減できます。スタンバイ・モードには用途に応じて，STOPモードとHALTモードが用意されています。

STOPモードは，システム・クロックを停止させてしまうモードです。このモードではCPUの消費電流は，ほとんどリーク電流だけとなります。したがって，CPUを動作させず，データ・メモリの内容保持を行う場合に有効です。

HALTモードはシステム・クロックの発振は継続しますが，CPUに対してクロックの供給が停止されるため，CPUの動作が停止するモードです。このモードは，STOPモードに比べて消費電流は低減できませんが，システム・クロックが発振しているため，HALT解除後にすぐ動作を開始させることができます。また，STOPモード，HALTモードどちらの場合でも，スタンバイ・モードに設定される直前のデータ・メモリ，レジスタ，出力ポートの出力ラッチなどの状態が保持されます（STOP 0000Bを除く）。したがって，スタンバイ・モードにする前にシステム全体の消費電流を抑えるように，ポートの状態を設定してください。

表15－1　スタンバイ・モード中の状態

		STOPモード	HALTモード
設定命令		STOP命令	HALT命令
システム・クロック発振回路		発振停止	発振継続
動作状態	CPU	・動作停止	
	RAM	・直前の状態を維持	
	ポート	・直前の状態を維持 注	
	TM 0	・カウント・パルスにINT 入力を選択した場合のみ動作可能 ・システム・クロックを選択した場合は停止（カウント値は保持）	・動作可能
	TM 1	・動作停止（カウント値は“0”にリセット）（カウント・アップも禁止状態）	・動作可能
	BTM	・動作停止（カウント値は保持）	・動作可能
	SIO	・シリアル・クロックに外部クロックを選択した場合のみ動作可能 注	・動作可能
	A/D	・動作停止 注（ADCR←00H）	・動作可能
	INT	・動作可能	・動作可能

★ 注　STOP 0000Bを実行した場合には，端子をポート以外の兼用端子機能で使用している場合も含めて，命令実行時点で入力ポートに切り替わります。

注意１．STOP命令，HALT命令の直前には，必ずNOP 命令を置いてください。

★ ２．割り込み要求フラグと割り込み許可フラグの両方がセットされており，その割り込みがスタンバイ・モードの解除条件に指定されている場合は，スタンバイ・モードには入りません。

15.2　HALTモード

15.2.1　HALTモードの設定

HALT命令を実行することにより，HALTモードに入ります。

HALT命令のオペランド$b_3b_2b_1b_0$は，HALTモードの解除条件です。

表15－2　HALTモードの解除条件

書式：HALT　$b_3b_2b_1b_0$B

ビット	HALTモードの解除条件 注1
b_3	1のとき IRQ×××による解除を許可する。注2，4
b_2	“0固定”
b_1	1のとき IRQTM1による強制解除を許可する。注3，4
b_0	1のとき $\overline{\text{RLS}}$入力による解除を許可する。注4

注1．HALT 0000Bのときは，リセット（$\overline{\text{RESET}}$入力，POC）だけが有効です。 ★

2．IP×××=1である必要があります。

3．IPTM1の状態によらず，HALTモードが解除されます。

4．IRQ×××=1の状態，または$\overline{\text{RLS}}$入力がロウ・レベルの状態で，HALT命令が実行されても，HALT命令は無視（NOP命令扱い）され，HALTモードには入りません。

15.2.2　HALTモード解除後のスタート番地

解除条件，割り込み許可条件によって変わります。

表15－3　HALTモード解除後のスタート番地

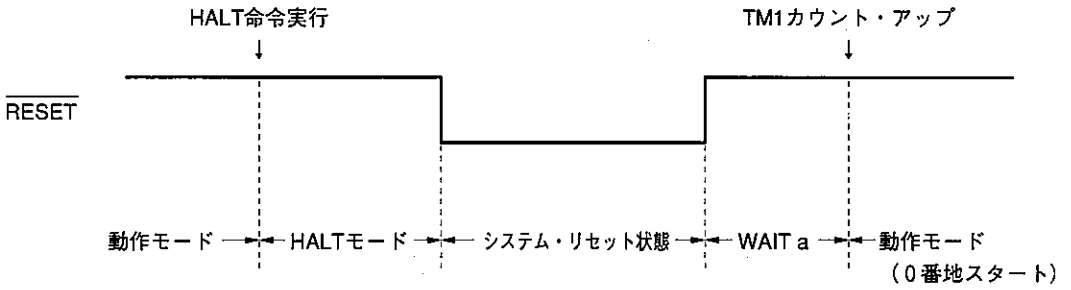
解除条件	解除後のスタート番地
リセット注1	0番地
$\overline{\text{RLS}}$	HALT命令の次の番地
IRQ××× 注2	DIの場合，HALT命令の次の番地
	EIの場合，割り込みベクタ （複数のIRQ ×××がセットされている場合には，優先順位の高い割り込みベクタ）

注1．リセットは$\overline{\text{RESET}}$入力，POCが有効です。 ★

2．IRQTM1による強制解除の場合を除き，IP×××=1である必要があります。

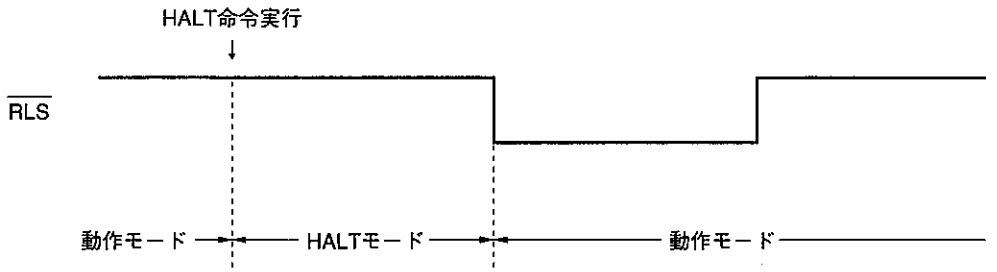
図15－1　HALTモードの解除

（a）$\overline{\text{RESET}}$入力によるHALT解除

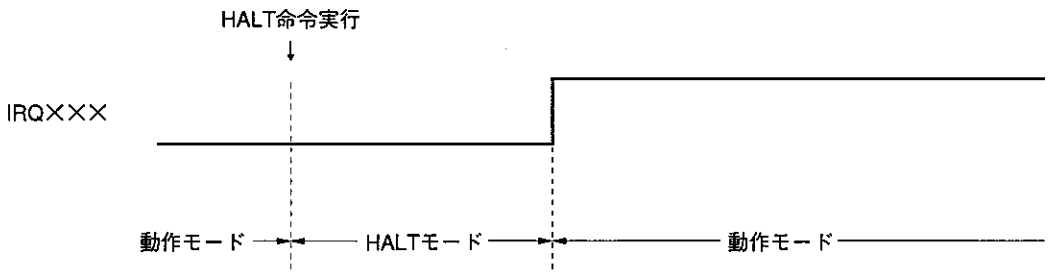


WAIT a：TM1が128分周のクロックを256カウントするまでの待ち時間です。
256×128/fx（約4 ms，fx = 8 MHz時）

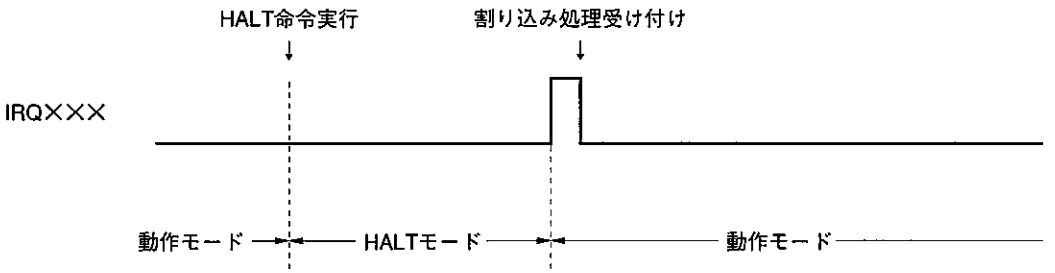
（b）$\overline{\text{RLS}}$入力によるHALT解除



（c）IRQ×××によるHALT解除（DI状態の場合）



（d）IRQ×××によるHALT解除（EI状態の場合）



15.2.3　HALTの設定条件

（1）$\overline{\text{RLS}}$入力による解除の場合

特に設定しておくレジスタはありません。

注意　HALT命令実行時にP0F0/$\overline{\text{RLS}}$端子がロウ・レベルの場合，HALT命令は無視（NOP命令扱い）され，HALTモードには入りません。

（2）IRQTM1による強制解除の場合

	設　定　条　件
外部クロックによる解除	●タイマ0＋タイマ1の16ビット・タイマとして設定（TM0CK1＝1，TM0CK0＝1，TM1CK1＝1，TM1CK0＝1）。 ●タイマ0およびタイマ1は動作可能状態（TM0EN＝1，TM1EN＝1） ●タイマ1の割り込みフラグをクリア（IRQTM1＝0）
内部クロックによる解除	●タイマ1は動作可能状態 ●タイマ1の割り込み要求フラグをクリア（IRQTM1＝0）

（3）割り込み要求フラグ（IRQ×××）による解除の場合

●HALT解除に利用する周辺ハードウエアをあらかじめ動作可能状態になるように設定。

タイマ0	動作可能状態（TM0EN＝1）
タイマ1	動作可能状態（TM1EN＝1）
タイマ0＋タイマ1	タイマ1はタイマ0からのカウント・アップを選択（TM1CK1＝1，TM1CK0＝1）。 タイマ0およびタイマ1は動作可能状態（TM0EN＝1，TM1EN＝1）。
ベーシック・インターバル・タイマ	常に動作可能状態。
シリアル・インタフェース	シリアル・インタフェース回路は動作可能状態（SIOTS＝1，SIOEN＝1）
INT端子	エッジ選択の設定

●HALT解除に利用する周辺ハードウエアの割り込み要求フラグ（IRQ×××）をクリア（0）。

●HALT解除に利用する周辺ハードウエアの割り込み許可フラグ（IP×××）をセット（1）。

注意　HALT命令の直前には必ず，NOP命令を記述してください。

NOP命令をHALT命令の直前に記述することにより，IRQ×××操作命令とHALT命令との間に1命令分の時間が生成されるため，たとえばCLR1　IRQ×××命令の場合は，IRQ×××のクリアがHALT命令に正しく反映されます（例1）。NOP命令をHALT命令の直前に記述しないと，CLR1 IRQ×××命令はHALT命令に反映されず，HALTモードには入りません（例2）。

例 1．正しいプログラム例

```
        ⋮
(IRQXXXのセット)
        ⋮

CLR1    IRQXXX
NOP             ;NOP命令をHALT命令の直前に記述
                ;（IRQXXXのクリアがHALT命令に対して正しく反映される）
HALT    1000B   ;HALT命令を正しく実行する（HALTモードに入る）

        ⋮
```

2．誤ったプログラム例

```
        ⋮
(IRQXXXのセット)
        ⋮

CLR1    IRQXXX  ;IRQXXXのクリアはHALT命令に対して反映されない
                ;（反映されるのはHALT命令の次の命令）
HALT    1000B   ;HALT命令は無視される（HALTモードに入らない）

        ⋮
```

15.3　STOPモード

15.3.1　STOPモードの設定

STOP命令を実行することにより，STOPモードに入ります。

STOP命令のオペランド$b_3b_2b_1b_0$は，STOPモードの解除条件です。

表15－4　STOPモードの解除条件

書式：STOP　$b_3b_2b_1b_0$B

ビット	STOPモードの解除条件 注1
b_3	1のとき IRQ×××による解除を許可する。注2，4
b_2	“0固定”
b_1	“0固定”
b_0	1のとき $\overline{\text{RLS}}$入力による解除を許可する。注3，4

注1．STOP 0000Bのときは，リセット（$\overline{\text{RESET}}$入力，POC）だけが有効です。また，STOP 0000B を実行した時点でマイコン内部はリセット直後の状態に初期化されます。 ★

2．IP×××＝1である必要があります。また，IRQTM1による解除はできません。

3．b_0は単独でセット（1）できません（STOP 0001Bは使用禁止）。
b_0をセット（1）する場合には，必ずb_3もセット（1）してください。

4．IRQ×××＝1の状態，または$\overline{\text{RLS}}$入力がロウ・レベルの状態で，STOP命令が実行されても，STOP命令は無視（NOP命令扱い）され，STOPモードには入りません。 ★

15.3.2　STOPモード解除後のスタート番地

解除条件，割り込み許可条件によって変わります。

表15－5　STOPモード解除後のスタート番地

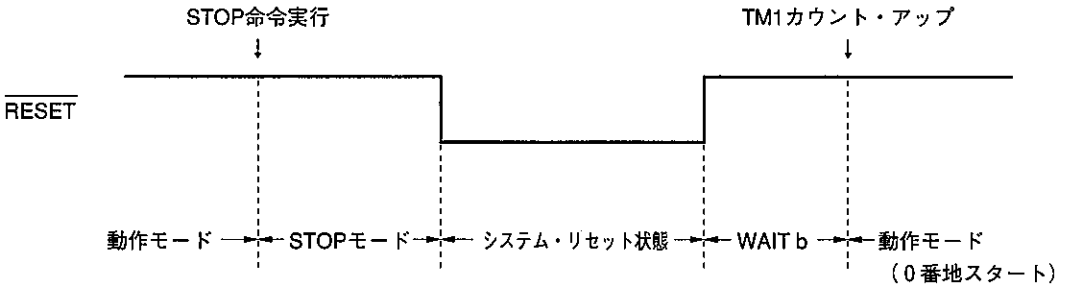
解除条件	解除後のスタート番地
リセット注1	0番地
$\overline{\text{RLS}}$	STOP命令の次の番地
IRQ××× 注2	DIの場合，STOP命令の次の番地
	EIの場合，割り込みベクタ （複数のIRQ ×××がセットされている場合には，優先順位の高い割り込みベクタ）

注1．リセットは$\overline{\text{RESET}}$入力，POCだけが有効です。 ★

2．IP×××＝1である必要があります。また，IRQTM1による解除はできません。

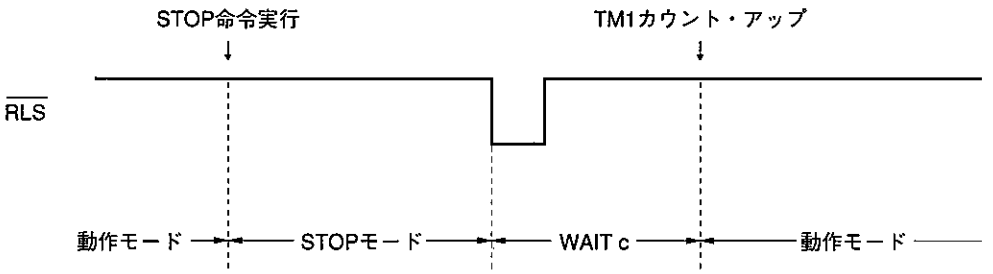
図15－2　STOPモードの解除（1/2）

（a）$\overline{\text{RESET}}$入力によるSTOP解除



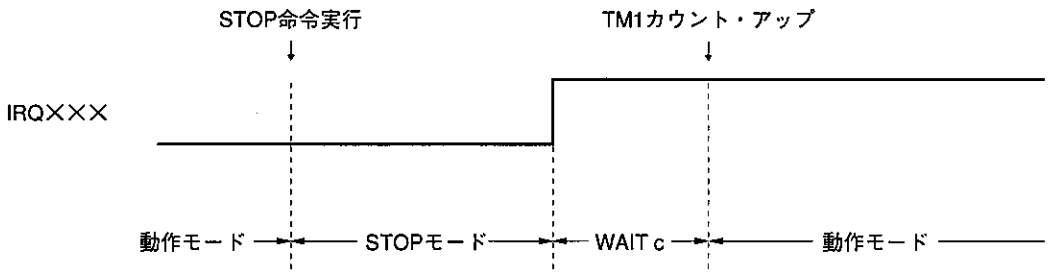
WAIT b：TM1が128分周のクロックを256カウントするまでの待ち時間
256×128/fx＋α（約4 ms＋α，fx＝8 MHz時）
α　：発振成長時間（発振子により異なります。）

（b）$\overline{\text{RLS}}$入力によるSTOP解除



★ WAIT c：TM1がm分周のクロックをn＋1カウントするまでの待ち時間
（n＋1）×m/fx＋α（n, mは，STOPモードに入る直前の値）
α　：発振成長時間（発振子により異なります。）

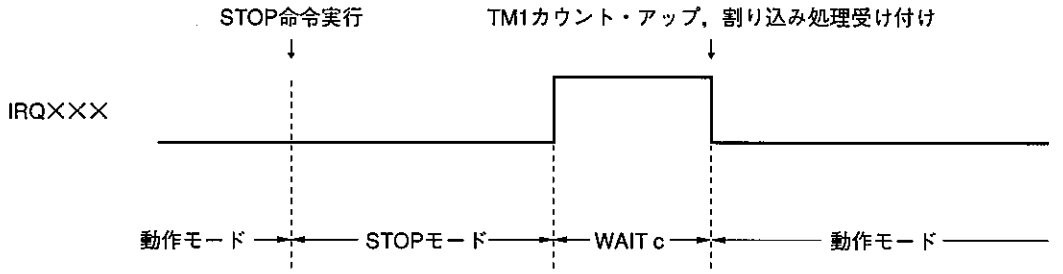
（c）IRQ×××によるSTOP解除（DI状態の場合）



★ WAIT c：TM1がm分周のクロックをn＋1カウントするまでの待ち時間
（n＋1）×m/fx＋α（n, mは，STOPモードに入る直前の値）
α　：発振成長時間（発振子により異なります。）

図15－2　STOPモードの解除（2/2）

（d）IRQ×××によるSTOP解除（EI状態の場合）



WAIT c：TM1がm分周のクロックをn＋1カウントするまでの待ち時間　★
（n＋1）×m/fx＋α（n，mは，STOPモードに入る直前の値）
α　：発振成長時間（発振子により異なります。）

15.3.3　STOPの設定条件

（1）$\overline{\text{RLS}}$入力による解除の場合

●タイマ1（発振安定待ち時間の生成）のモジュロ・レジスタ値の設定。　★

注意　STOP命令実行時にP0F0/$\overline{\text{RLS}}$端子がロウ・レベルの場合，STOP命令は無視（NOP命令扱い）され，STOPモードには入りません。　★

（2）IRQ ×××による解除の場合

IRQによる解除	●INT端子から入力される信号に対するエッジ選択（IEGMD1，IEGMD0）を設定。 ●タイマ1（発振安定待ち時間の生成）のモジュロ・レジスタ値の設定。 ●INT端子の割り込み要求フラグ（IRQ）をクリア（0）。 ●INT端子の割り込み許可フラグ（IP）をセット（1）。
IRQSIOによる解除	●ソース・クロックを$\overline{\text{SCK}}$端子から入力される外部クロック（SIOCK1＝0，SIOCK0＝0）に設定。 ●シリアル・インタフェースを動作可能状態（SIOTS＝1）に設定。 ●タイマ1（発振安定待ち時間の生成）のモジュロ・レジスタ値の設定。 ●シリアル・インタフェースの割り込み要求フラグ（IRQSIO）をクリア（0）。 ●シリアル・インタフェースの割り込み許可フラグ（IPSIO）をセット（1）。
IRQTM0による解除	●タイマ0のソース・クロックをINT端子から入力される外部クロック（TM0CK1＝1，TM0CK0＝1）に設定。 ●タイマ0のモジュロ・レジスタ値を設定。 ●タイマ1（発振安定待ち時間の生成）のモジュロ・レジスタ値とソース・クロックを設定。 ●タイマ0を動作可能状態にする（TM0EN＝1）。 ●タイマ0の割り込み要求フラグ（IRQTM0）をクリア（0）。 ●タイマ0の割り込み許可フラグ（IPTM0）をセット（1）。

注意　STOP命令の直前には必ず，NOP命令を記述してください。
NOP命令をSTOP命令の直前に記述することにより，IRQ×××操作命令とSTOP命令との間に1命令分の時間が生成されるため，たとえばCLR1　IRQ×××命令の場合は，IRQ ×××のクリアがSTOP命令に正しく反映されます（例1）。NOP命令をSTOP命令の直前に記述しないと，CLR1 IRQ ×××命令はSTOP命令に反映されず，STOPモードには入りません（例2）。

例 1．正しいプログラム例

```
        ⋮
(IRQ×××のセット)
        ⋮

CLR1      IRQ×××
NOP               ; NOP命令をSTOP命令の直前に記述
                  ; (IRQ×××のクリアがSTOP命令に対して正しく反映される)
STOP      1000B   ; STOP命令を正しく実行する（STOPモードに入る）

        ⋮
```

2．誤ったプログラム例

```
        ⋮
(IRQ×××のセット)
        ⋮

CLR1      IRQ×××  ; IRQ×××のクリアはSTOP命令に対して反映されない
                  ; (反映されるのはSTOP命令の次の命令)
STOP      1000B   ; STOP命令は無視される（STOPモードに入らない）

        ⋮
```

第16章　リセット

μPD17149のリセットには，$\overline{\text{RESET}}$入力によるリセット，電源電圧の低下を検出する内蔵POC回路によるリセット，プログラムの暴走時にリセットするためのウォッチドッグ・タイマ機能，およびアドレス・スタックのオーバフロー／アンダフローによるリセットがあります。ただし，内蔵POC回路はマスク・オプションです。

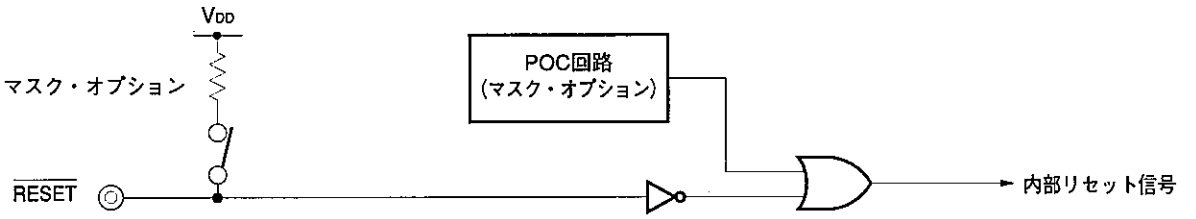
16.1　リセット機能

リセット機能は，デバイス動作の初期化を行うために使用します。なお，リセットの種類により，初期化される内容が異なります。

表16－1　リセット時の各ハードウエアの状態

ハードウエア ＼ リセットの種類		・動作中の$\overline{\text{RESET}}$入力 ・内蔵POC回路によるリセット	・スタンバイ・モード中の$\overline{\text{RESET}}$入力 ・スタンバイ・モード中の内蔵POC回路によるリセット	・ウォッチドッグ・タイマのオーバフロー ・スタックのオーバフローおよびアンダフロー
プログラム・カウンタ		0000H	0000H	0000H
ポート	入力/出力	入力	入力	入力
	出力ラッチの内容	0	0	不定
汎用データ・メモリ	汎用データ・メモリ（DBFを除く）	不定	リセット直前の状態を保持	不定 ★
	DBF	不定	不定	不定
	システム・レジスタ（WRを除く）	0	0	0
	WR	不定	リセット直前の状態を保持	不定 ★
コントロール・レジスタ		SP = 5H, IRQTM1 = 1, TM1EN = 1, IRQBTM = 1, INTはそのときのINT端子の状態，それ以外はすべて0。 第9章　レジスタ・ファイル（RF）参照		SP = 5H, INTはそのときのINT端子の状態，それ以外はすべてリセット直前の状態を保持。
タイマ0およびタイマ1	カウント・レジスタ	00H	00H	タイマ0：00H, タイマ1：不定
	モジュロ・レジスタ	FFH	FFH	FFH
ベーシック・インターバル・タイマのバイナリ・カウンタ		不定	不定	不定。ただし，ウォッチドッグ・タイマのオーバフローの場合は40H。 ★
シリアル・インタフェース	シフト・レジスタ（SIOSFR）	不定	リセット直前の状態を保持	不定
	シリアル出力ラッチ	0	0	不定
A/Dコンバータのデータ・レジスタ（ADCR）		00H	00H	00H

図16－1　リセット・ブロックの構成



16.2　リセット動作

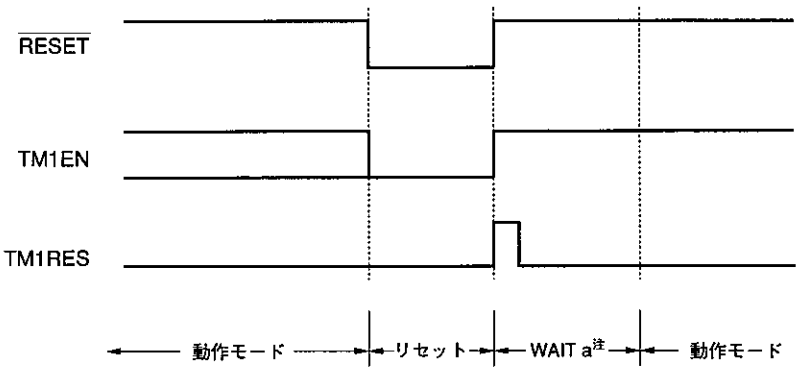
$\overline{\text{RESET}}$入力によりリセットをかけたときの動作を図16－2に示します。

$\overline{\text{RESET}}$端子をロウ・レベルからハイ・レベルに立ち上げると，システム・クロックの発振を開始し，タイマ1を用いた発振安定待ちをしたのち，0000H番地よりプログラムの実行を開始します。

以上の動作は，POC回路によりリセットがかかった場合も同様です。

なお，ウォッチドッグ・タイマのオーバフローおよびアドレス・スタック・レジスタのオーバフローとアンダフローによるリセットでは発振安定待ち時間（WAIT a）は発生せず，内部を初期状態にしたのち，0000H番地スタートとなります。

図16－2　リセット動作



注　発振安定待ち時間です。タイマ1によりシステム・クロック（fx）を128×256カウント（約4 ms，fx＝8 MHz時）すると動作モードとなります。

第17章　POC回路（マスク・オプション）

POC回路は，電源電圧を監視して，電源のON/OFF時などにマイコン内部にリセットをかけます。クロック周波数がfx = 400 kHz ～ 4 MHzの応用回路で利用できます。

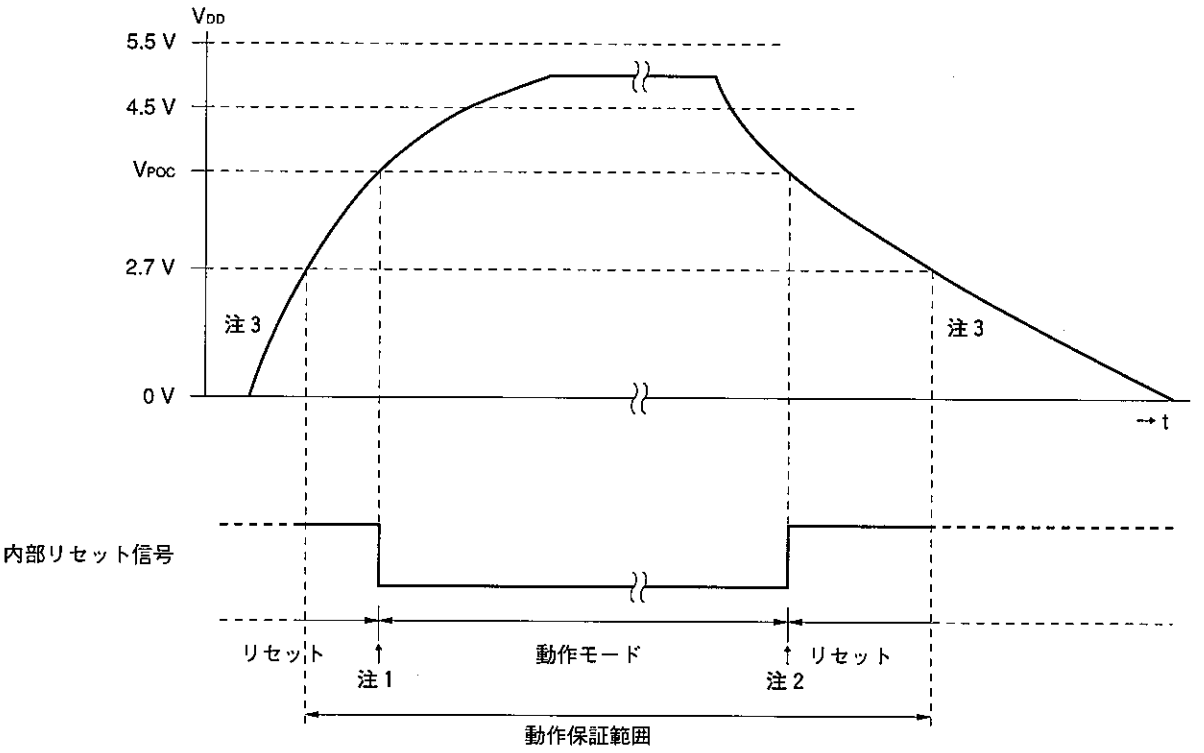
μPD17149は，POC回路をマスク・オプションによって内蔵できます。ただし，μPD17P149には内蔵されていません。

17.1　POC回路の機能

次の機能があります。

・$V_{DD} \leqq V_{POC}$のとき，内部リセット信号を発生する。
・$V_{DD} > V_{POC}$のとき，内部リセット信号を解除する。
（V_{DD}：電源電圧，V_{POC}：POC検出電圧）

図17－1　POC回路の動作



注1．実際には，動作モードに移るまでにタイマ1による発振安定待ち時間があります。発振安定待ち時間は，約2048命令実行時間（約8 ms，fx = 4 MHz時）です。

2．電源電圧が低下したとき再びリセットがかかるためには，V_{POC}以下に電圧が下がった状態が，リセット検出パルス幅t_{SAMP}以上の期間保たれる必要があります。
したがって，実際にはリセットがかかるまで，最大t_{SAMP}の時間遅れがあります。

3．電源電圧V_{DD} = 2.7 V未満の領域では，μPD17149のすべての機能の動作は保証されていません。しかしながら，POC回路は発振の有無に関係なく，可能なかぎり内部リセット信号を発生するように設計されています。したがって，内部回路が動作可能な電圧に達した時点から内部リセットがかかります。

備考　V_{POC}，t_{SAMP} の値については，データ・シートの電気的特性を参照してください。

17.2　POC回路を使用するための条件

応用回路が，次の条件を満たすとき，POC回路を使用できます。

●高度な信頼性を要求する応用回路でないこと注。
●電源電圧がV_{DD} = 4.5 ～ 5.5 Vの応用回路であること。
●システム・クロック周波数がf_X = 400 kHz ～ 4MHzの応用回路であること。
●電源電圧（V_{DD}）の特性が，POC回路の規格を満足すること（17.4 参照）。

注　高度な信頼性を要求する応用回路では，必ず外部から$\overline{\text{RESET}}$入力するように設計してください。 ★

注意　POC回路を使用すると，スタンバイ・モード時の電流が若干増加します。

備考　POC回路の機能は，V_{DD} = 2.7 ～5.5 Vで動作保証されています。

17.3　POC回路使用時の注意事項

POC回路はフェイルセーフを考慮して設計されています。したがって，電源電圧範囲（V_{DD} = 4.5～5.5 V）であっても，電源電圧の急激な変動があった場合に暴走する危険を回避するため，なるべくリセット信号を発生するという補助的な機能があります注。

そのため電源電圧が次の条件を満たしていない場合にはPOC回路によりリセットがかかることがありますので注意してください。

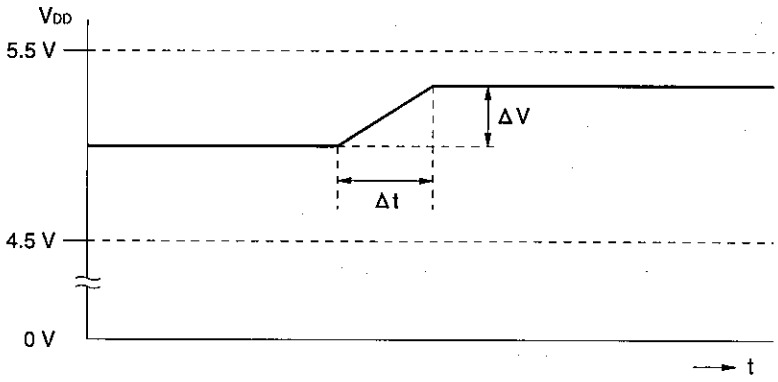
●標準水準の製品，特別水準（A）の製品の場合

・電源電圧の変動幅（ΔV）が100 mV以内であること。
・電源電圧の変動幅が100 mVを超える場合には変動の傾斜（ΔV/Δt）が3 mV/μs以内であること。

●特別水準（A 1）の製品の場合

・電源電圧の変動幅（ΔV）が80 mV以内であること。
・電源電圧の変動幅が80 mVを超える場合には変動の傾斜（ΔV/Δt）が2 mV/μs以内であること。

図17－2　電源電圧の変動



注　確実なリセットが保証されているわけではありません。電源電圧の変動によるリセット機能はあくまでも補助的に用意された機能であるため，この機能を使って確実にリセットをかけるための条件はありません。したがって，この機能に期待した設計をしないでください。

17.4　電源電圧の特性とPOC回路の規格の検討

17.4.1　電源電圧立ち下がり速度t_{POCS}の検討

今，電源電圧Vの応用回路を考えます（4.5 ≦ V ≦ 5.5 V）。

簡単にするために，電源電圧の立ち下がりが単純な減衰特性を示し，その特性v（t）が

$$v\ (t) = V\ \varepsilon^{-(t/T)}\quad (T：立ち下がりの時定数)$$

で表せるものとします。

立ち下がり開始直後（t = 0）に，立ち下がり速度（立ち下がりの傾斜）は最大になるため，この時点で，電源電圧立ち下がり速度t_{POCS}の規格を満足すればよいことになります。

したがって，電源電圧の立ち下がりが，

$$T = V/t_{POCS}$$

の時定数を持っている電源であればよいことになります。具体的には，

$$T = 5.5\ [V]\ /0.08\ [V/ms] = 68.75\ [ms]$$

となり，電源電圧の立ち下がりが，約70 ms以上の時定数を持っていればよいことになります。

注意　上記は，電源電圧の立ち下がり特性が，$v\ (t) = V\ \varepsilon^{-(t/T)}$で表されると仮定した場合の計算例です。あくまで応用回路設計の目安としてご利用ください。

17.4.2　リセット検出パルス幅t_{SAMP}の検討

電源電圧が低下したときにリセットがかかるためには，V_{POC}以下に電圧が下がった状態が，リセット検出パルス幅t_{SAMP}以上の期間保たれる必要があります。

電源の瞬停などの場合であっても，t_{SAMP}の期間を確保していればPOC回路によりリセットがかかるため，暴走状態から脱出できます。

17.5　POC回路の動作を外部から知る方法

外部でプルアップされた入出力ポートからロウ・レベルを出力するようにプログラムします。

POC回路が働きリセットがかかると，入出力ポートは入力モードになるため，ポートの状態はハイ・レベルになります。

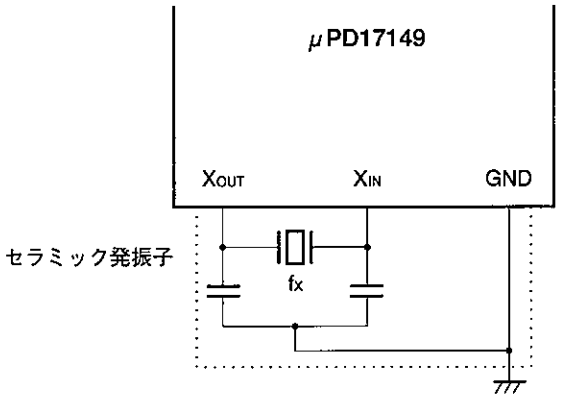
したがって，ポートの状態を外部から観測することによって，μPD17149の動作電源電圧範囲内（V_{DD} = 2.7 ～ 5.5 V）におけるPOC回路の動作を外部から知ることができます。

〔メ　モ〕

第18章　システム・クロック発振回路の構成上の注意

システム・クロック発振回路は，X_{IN}，X_{OUT}端子に接続されたセラミック発振子によって発振します。 ★
図18－1にシステム・クロック発振回路の外付け回路を示します。

図18－1　システム・クロック発振回路の外付け回路



fx：システム・クロック発振周波数

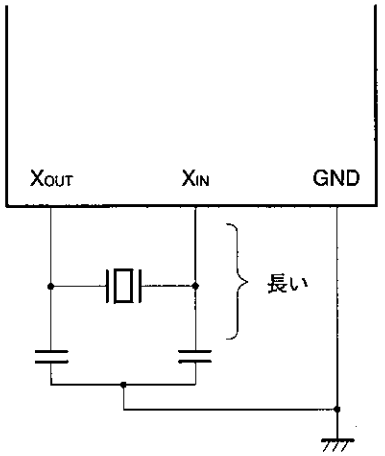
注意　システム・クロック発振回路は，グランド配線の抵抗成分やインダクタンス成分を極力小さくするように配線してください。また，配線容量などの影響を避けるために図18－1の ⋯ の部分を次のように配線してください。

●配線は極力短くする
●他の信号線と交差させない。また，変化する大電流が流れる線と接近させない。
●発振回路のコンデンサの接地点は，常にVssと同電位になるようにする。大電流が流れるグランド配線に接地しない。
●発振回路から信号を取り出さない。

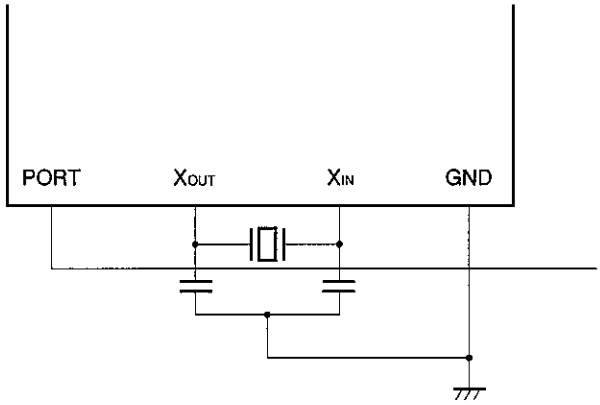
図18－2に発振回路の悪い例を示します。

図18－2　発振回路の悪い例

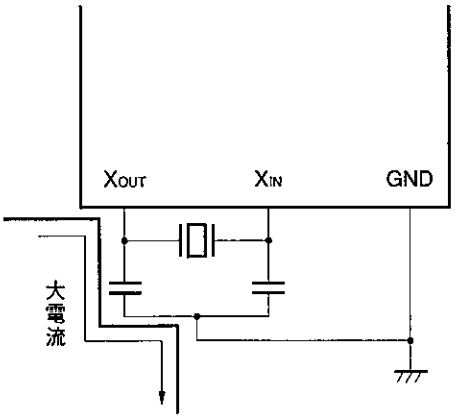
（a）接続回路の配線が長い



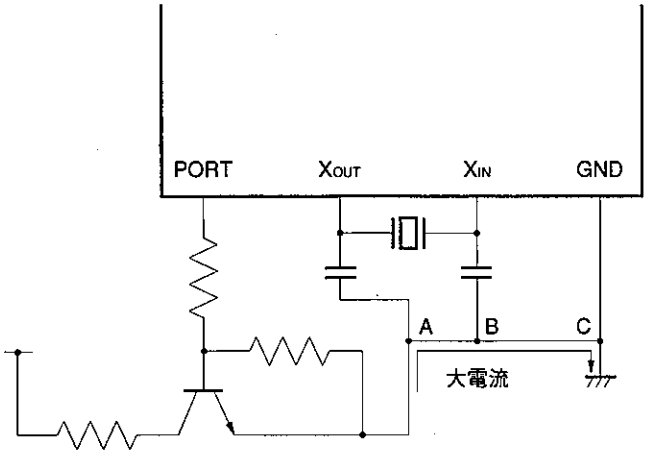
（b）信号線が交差している



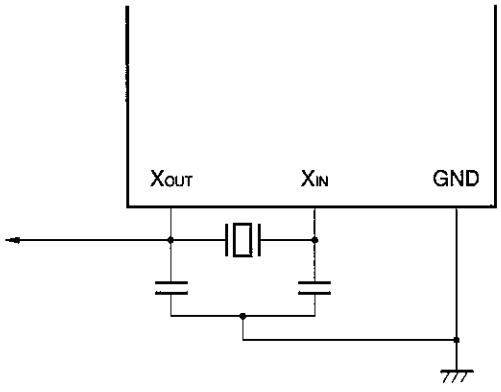
（c）変化する大電流が信号線に近接している



（d）発振回路のグランド・ライン上に電流が流れる
（C点に対してA点，B点の電位が変動する）



（e）信号を取り出している



第19章　ワン・タイムPROMの書き込みとベリファイ

μPD17P149に内蔵されているプログラム・メモリは4096×16ビットのワン・タイムPROMです。

ワン・タイムPROMの書き込み／ベリファイには，表19－1に示す端子を使用します。なお，アドレス入力はなく，代わりにCLK端子からのクロック入力によりアドレスを更新する方法をとっています。

注意　$P0F_0/\overline{RLS}/V_{PP}$端子は，プログラム書き込み／ベリファイ・モード時はV_{PP}端子として使用しています。このため，通常動作モード時において$P0F_0/\overline{RLS}$端子にV_{DD}＋0.3 V以上の高電圧が印加されると，マイコンが暴走する可能性がありますので，端子の保護には十分注意してください。

表19－1　プログラム・メモリ書き込み／ベリファイ時の使用端子

端子名	機能
V_{PP}	プログラム電圧印加用端子です。＋12.5 Vを印加します。
V_{DD}	正電源です。＋6 Vを印加します。
CLK	アドレス更新用クロック入力です。パルスを4回入力することにより，プログラム・メモリのアドレスを更新します。
MD_0 - MD_3	動作モード選択用入力です。
D_0 - D_7	8ビット・データ入出力端子です。

19.1　マスクROM製品とワン・タイムPROM製品との違い

μPD17P149は，マスクROM内蔵製品μPD17149のプログラム・メモリをワン・タイムPROMに置き換えた製品です。

表19－2にマスクROM製品とワン・タイムPROM製品との違いを示します。

各製品間の違いは，ROMの容量およびマスク・オプションの指定ができるかできないかの違いのみで，CPU機能や内蔵している周辺ハードウエアは同じです。したがって，μPD17P149はμPD17145, 17147, 17149のシステム開発時のプログラム評価用として使用できます。

表19－2　マスクROM製品とワン・タイムPROM製品との違い

項　目	μPD17145	μPD17147	μPD17149	μPD17P149
ROM	マスクROM			ワン・タイムPROM
	1024×16ビット (0000H-03FFH)	2048×16ビット (0000H-07FFH)	4096×16ビット (0000H-0FFFH)	
★ プログラム・カウンタ（PC） アドレス・レジスタ（AR） アドレス・スタック・レジスタ	10ビット	11ビット	12ビット	
P0F, $\overline{\text{RESET}}$, INT 端子のプルアップ抵抗	マスク・オプション			なし
POC回路	マスク・オプション			なし
V_{PP}端子，動作モード選択端子	なし			あり
★ 品質水準	標準水準 特別水準〔（A），（A1）〕			標準水準

★ 注意　PROM製品は，マスクROM製品と機能的には高い互換性がありますが，内部ROM回路や電気的特性の一部などに違いがあります。

PROM製品からマスクROM製品に切り替える際にはマスクROM製品のサンプルによる応用評価を十分に行ってください。

19.2　プログラム・メモリ書き込み／ベリファイ時の動作モード

μPD17P149は，ある一定時間のリセット状態（V_{DD}＝5 V，$\overline{\text{RESET}}$＝0 V）のあと，V_{DD}端子に＋6 V，V_{PP}端子に＋12.5 Vを印加すると，プログラム・メモリ書き込み／ベリファイ・モードになります。このモードはMD_0-MD_3端子設定により次のような動作モードとなります。なお，X_{OUT}端子はオープンにしてください。また，表19－1に示していない端子は（$\overline{\text{RESET}}$端子も含めて）すべてプルダウン抵抗を介してGNDに接続してください。

表19－3　動作モードの設定方法

動作モードの設定						動作モード
V_{PP}	V_{DD}	MD_0	MD_1	MD_2	MD_3	
＋12.5 V	＋6 V	H	L	H	L	プログラム・メモリ・アドレスの0クリア
		L	H	H	H	書き込みモード
		L	L	H	H	ベリファイ・モード
		H	×	H	H	プログラム・インヒビット・モード

備考　×：don't care（LまたはH）

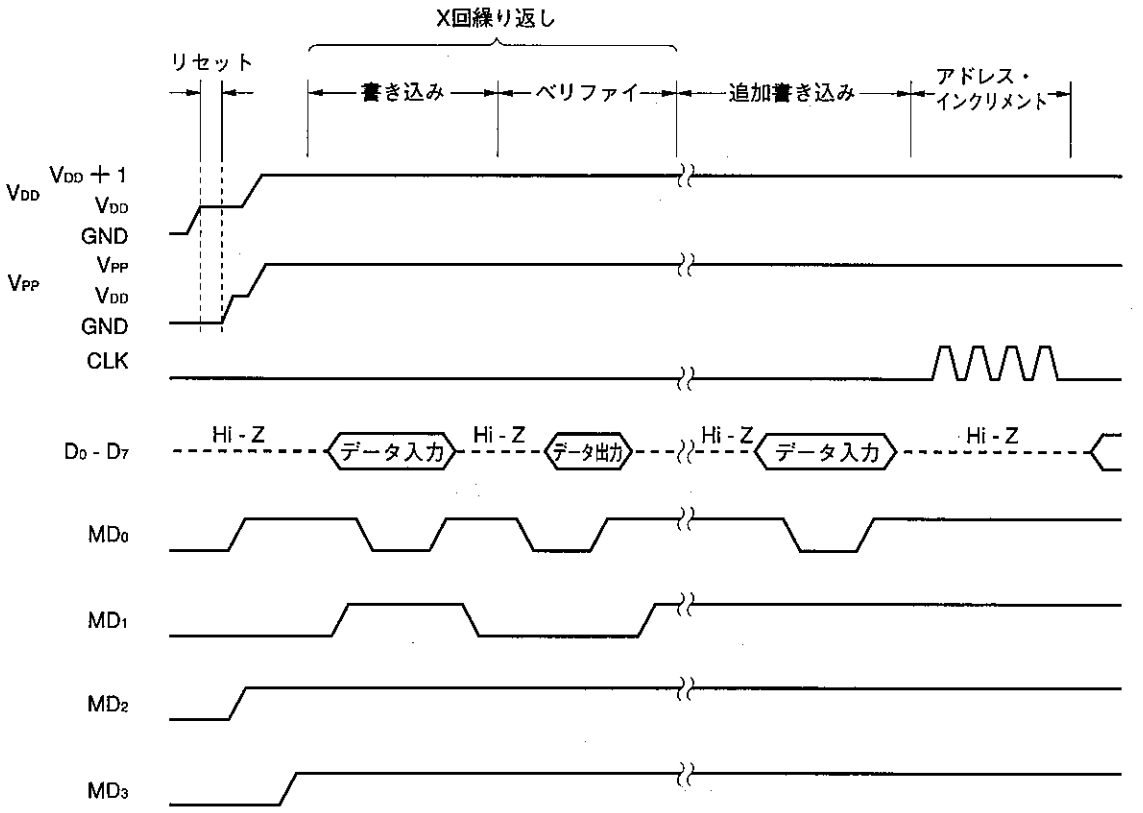
19.3　プログラム・メモリ書き込み手順

プログラム・メモリ書き込みの手順は次のようになっています。

（1）使用しない端子を抵抗を介してGNDにプルダウン（X_{OUT}はオープン）。CLK 端子はロウ・レベル。
（2）V_{DD}端子に 5 Vを供給。V_{PP}端子はロウ・レベル。
（3）10 μsウエイト後，V_{PP}端子に 5 Vを供給。
（4）モード設定端子をプログラム・メモリ・アドレスの 0 クリア・モードに設定。
（5）V_{DD}に6V，V_{PP}に12.5 Vを供給。
（6）プログラム・インヒビット・モード。
（7）1 msの書き込みモードでデータを書き込む。
（8）プログラム・インヒビット・モード。
（9）ベリファイ・モード。書き込めていれば（10）へ，書き込めていなければ（7）-（9）を繰り返す。
（10）（（7）-（9）で書き込んだ回数：X）×1 ms の追加書き込み。
（11）プログラム・インヒビット・モード。
（12）CLK端子にパルスを 4 回入力することにより，プログラム・メモリのアドレスを更新（＋1）。
（13）（7）-（12）を最終アドレスまで繰り返す。
（14）プログラム・メモリ・アドレスの 0 クリア・モード。
（15）V_{DD}，V_{PP}端子の電圧を 5 Vに変更。
（16）電源オフ。

この（2）-（12）の手順を図19－1に示します。

図19－1　プログラム・メモリ書き込み手順



19.4　プログラム・メモリ読み出し手順

（1）使用しない端子を抵抗を介してGNDにプルダウン（XOUTはオープン）。CLK 端子はロウ・レベル。

（2）VDD端子に 5 Vを供給。VPP端子はロウ・レベル。

（3）10 μsウエイト後，VPP端子に 5 Vを供給。

（4）モード設定端子をプログラム・メモリ・アドレスの 0 クリア・モードに設定。

（5）VDD端子に 6 V，VPPに12.5 Vを供給。

（6）プログラム・インヒビット・モード。

（7）ベリファイ・モード。CLK端子にクロック・パルスを入力すると，4 回入力するごとにデータを 1 アドレスずつ順次出力。

（8）プログラム・インヒビット・モード。

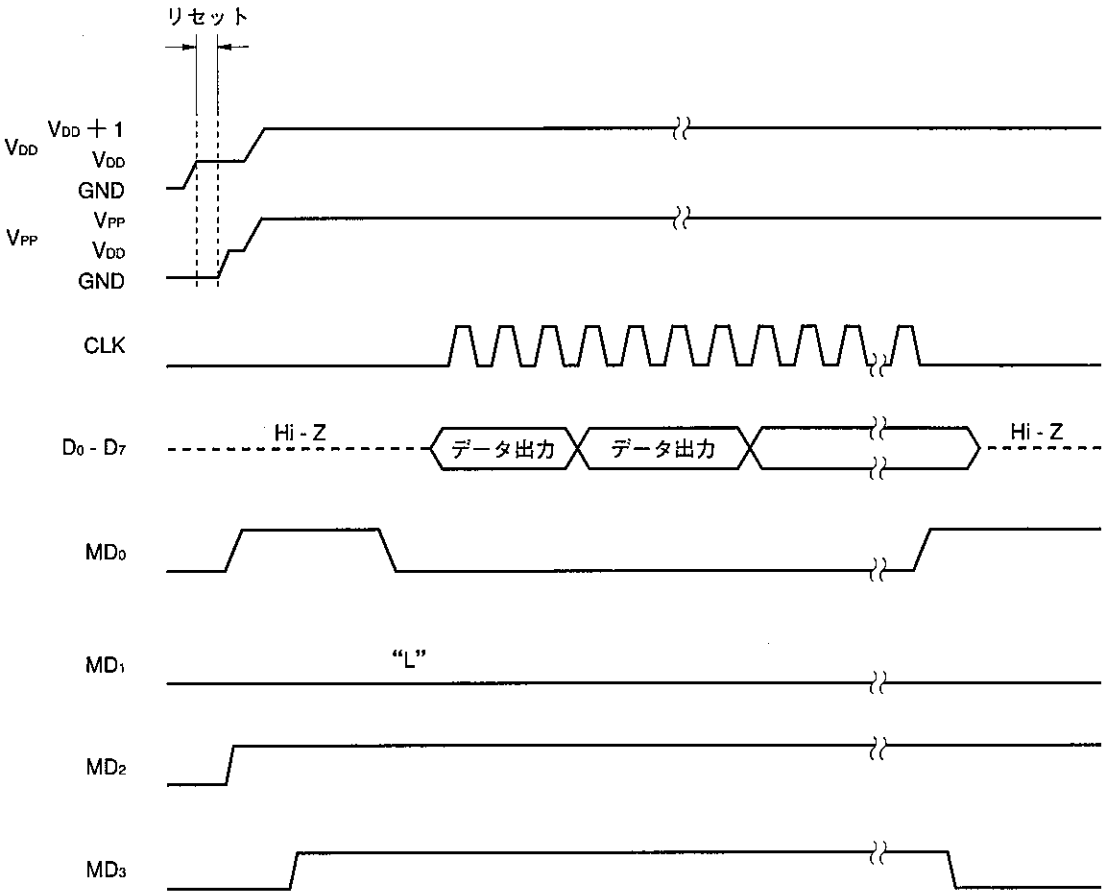
（9）プログラム・メモリ・アドレスの 0 クリア・モード。

（10）VDD，VPP端子の電圧を 5 Vに変更。

（11）電源オフ。

プログラム・メモリ読み出し手順の（2）-（9）を図19−2に示します。

図19−2　プログラム・メモリ読み出し手順



〔メ　モ〕

第20章　命令セット

20.1　命令セット概要

（1／2）

b15 / b14 - b11		0		1	
BIN	HEX				
0000	0	ADD	r, m	ADD	m, #n4
0001	1	SUB	r, m	SUB	m, #n4
0010	2	ADDC	r, m	ADDC	m, #n4
0011	3	SUBC	r, m	SUBC	m, #n4
0100	4	AND	r, m	AND	m, #n4
0101	5	XOR	r, m	XOR	m, #n4
0110	6	OR	r, m	OR	m, #n4
0111	7	INC	AR		
		INC	IX		
		MOVT	DBF, @AR		
		BR	@AR		
		CALL	@AR		
		RET			
		RETSK			
		EI			
		DI			
		RETI			
		PUSH	AR		
		POP	AR		
		GET	DBF, p		
		PUT	p, DBF		
		PEEK	WR, rf		
		POKE	rf, WR		
		RORC	r		
		STOP	s		
		HALT	h		
		NOP			

（2／2）

b15 / b14 - b11		0		1	
BIN	HEX				
1000	8	LD	r, m	ST	m, r
1001	9	SKE	m, #n4	SKGE	m, #n4
1010	A	MOV	@r, m	MOV	m, @r
1011	B	SKNE	m, #n4	SKLT	m, #n4
1100	C	BR	addr　（ページ0）	CALL	addr
1101	D	BR	addr　（ページ1）	MOV	m, #n4
1110	E			SKT	m, #n
1111	F			SKF	m, #n

20.2　凡　　例

AR	：アドレス・レジスタ
ASR	：スタック・ポインタで示されるアドレス・スタック・レジスタ
addr	：プログラム・メモリ・アドレス（下位11ビット）
BANK	：バンク・レジスタ
CMP	：コンペア・フラグ
CY	：キャリー・フラグ
DBF	：データ・バッファ
h	：ホールト解除条件
INTEF	：インタラプト・イネーブル・フラグ
INTR	：割り込み時スタックに自動退避されるレジスタ
INTSK	：割り込みスタック・レジスタ
IX	：インデクス・レジスタ
MP	：データ・メモリ・ロウ・アドレス・ポインタ
MPE	：メモリ・ポインタ・イネーブル・フラグ
m	：m_R, m_Cで示されるデータ・メモリ・アドレス
m_R	：データ・メモリ・ロウ・アドレス（上位）
m_C	：データ・メモリ・カラム・アドレス（下位）
n	：ビット・ポジション（4ビット）
n4	：イミーディエト・データ（4ビット）
PAGE	：ページ（プログラム・カウンタのビット11）
PC	：プログラム・カウンタ
p	：周辺アドレス
p_H	：周辺アドレス（上位3ビット）
p_L	：周辺アドレス（下位4ビット）
r	：ジェネラル・レジスタ・カラム・アドレス
rf	：レジスタ・ファイル・アドレス
rf_R	：レジスタ・ファイル・ロウ・アドレス（上位3ビット）
rf_C	：レジスタ・ファイル・カラム・アドレス（下位4ビット）
SP	：スタック・ポインタ
s	：ストップ解除条件
WR	：ウインドウ・レジスタ
（X）	：Xでアドレスされる内容

20.3　命令セット一覧

命令群	ニモニック	オペランド	オペレーション	命令コード			
				オペ・コード	オペランド		
加算	ADD	r, m	(r) ← (r) + (m)	00000	m_R	mc	r
		m, #n4	(m) ← (m) +n4	10000	m_R	mc	n4
	ADDC	r, m	(r) ← (r) + (m) +CY	00010	m_R	mc	r
		m, #n4	(m) ← (m) +n4 +CY	10010	m_R	mc	n4
	INC	AR	AR ← AR+1	00111	000	1001	0000
		IX	IX ← IX+1	00111	000	1000	0000
減算	SUB	r, m	(r) ← (r) − (m)	00001	m_R	mc	r
		m, #n4	(m) ← (m) −n4	10001	m_R	mc	n4
	SUBC	r, m	(r) ← (r) − (m) −CY	00011	m_R	mc	r
		m, #n4	(m) ← (m) −n4 −CY	10011	m_R	mc	n4
論理演算	OR	r, m	(r) ← (r) ∨ (m)	00110	m_R	mc	r
		m, #n4	(m) ← (m) ∨ n4	10110	m_R	mc	n4
	AND	r, m	(r) ← (r) ∧ (m)	00100	m_R	mc	r
		m, #n4	(m) ← (m) ∧ n4	10100	m_R	mc	n4
	XOR	r, m	(r) ← (r) ∀ (m)	00101	m_R	mc	r
		m, #n4	(m) ← (m) ∀ n4	10101	m_R	mc	n4
判断	SKT	m, #n	CMP ← 0, if (m) ∧ n = n, then skip	11110	m_R	mc	n
	SKF	m, #n	CMP ← 0, if (m) ∧ n = 0, then skip	11111	m_R	mc	n
比較	SKE	m, #n4	(m) −n4, skip if zero	01001	m_R	mc	n4
	SKNE	m, #n4	(m) −n4, skip if not zero	01011	m_R	mc	n4
	SKGE	m, #n4	(m) −n4, skip if not borrow	11001	m_R	mc	n4
	SKLT	m, #n4	(m) −n4, skip if borrow	11011	m_R	mc	n4
回転	RORC	r	→ CY → (r) b_3 → (r) b_2 → (r) b_1 → (r) b_0 →	00111	000	0111	r

命令群	ニモニック	オペランド	オペレーション	命令コード			
				オペ・コード	オペランド		
転送	LD	r, m	(r) ← (m)	01000	m_R	mc	r
	ST	m, r	(m) ← (r)	11000	m_R	mc	r
	MOV	@r, m	if MPE = 1：(MP, (r)) ← (m) if MPE = 0：(BANK, m_R, (r)) ← (m)	01010	m_R	mc	r
		m, @r	if MPE = 1：(m) ← (MP, (r)) if MPE = 0：(m) ← (BANK, m_R, (r))	11010	m_R	mc	r
		m, #n4	(m) ← n4	11101	m_R	mc	n4
	MOVT[注1]	DBF, @AR	SP ← SP −1, ASR ← PC, PC ← AR, DBF ← (PC), PC ← ASR, SP ← SP+1	00111	000	0001	0000
	PUSH	AR	SP ← SP−1, ASR ← AR	00111	000	1101	0000
	POP	AR	AR ← ASR, SP ← SP+1	00111	000	1100	0000
	PEEK	WR, rf	WR ← (rf)	00111	rf_R	0011	rfc
	POKE	rf, WR	(rf) ← WR	00111	rf_R	0010	rfc
	GET	DBF, p	DBF ← (p)	00111	p_H	1011	p_L
	PUT	p, DBF	(p) ← DBF	00111	p_H	1010	p_L
分岐	BR	addr	if 0000H ≦ (PC) ≦ 07FFH PC ← addr, PAGE ← 0	01100	addr		
			if 0800H ≦ (PC) ≦ 0FFFH[注2] PC ← addr, PAGE ← 1	01101			
		@AR	PC ← AR	00111	000	0100	0000
サブルーチン	CALL	addr	SP ← SP−1, ASR ← PC, PC ← addr	11100	addr		
		@AR	SP ← SP−1, ASR ← PC, PC ← AR	00111	000	0101	0000
	RET		PC ← ASR, SP ← SP+1	00111	000	1110	0000
	RETSK		PC ← ASR, SP ← SP+1 and skip	00111	001	1110	0000
	RETI		PC ← ASR, INTR ← INTSK, SP ← SP+1	00111	100	1110	0000
割り込み	EI		INTEF ← 1	00111	000	1111	0000
	DI		INTEF ← 0	00111	001	1111	0000
その他	STOP	s	STOP	00111	010	1111	s
	HALT	h	HALT	00111	011	1111	h
	NOP		No operation	00111	100	1111	0000

★

注1．MOVT命令の実行には例外的に2命令サイクルを必要とします。 ★

2．μPD17145, 17147にはありません。

20.4　アセンブラ（AS17K）組み込みマクロ命令

凡　例

flag n　：FLG 型シンボル

〈 〉：〈 〉内は省略可能

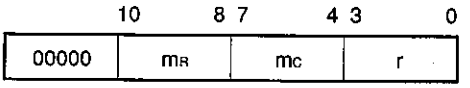
	ニモニック	オペランド	オペレーション	n
組み込みマクロ	SKTn	flag 1, …flag n	if （flag 1） ～ （flag n） = all "1" , then skip	1 ≦ n ≦ 4
	SKFn	flag 1, …flag n	if （flag 1） ～ （flag n） = all "0" , then skip	1 ≦ n ≦ 4
	SETn	flag 1, …flag n	（flag 1） ～ （flag n） ← 1	1 ≦ n ≦ 4
	CLRn	flag 1, …flag n	（flag 1） ～ （flag n） ← 0	1 ≦ n ≦ 4
	NOTn	flag 1, …flag n	if （flag n） = "0", then （flag n） ← 1 if （flag n） = "1", then （flag n） ← 0	1 ≦ n ≦ 4
	INITFLG	〈NOT〉 flag 1, … 〈 〈NOT〉 flag n〉	if description = NOT flag n, then （flag n） ← 0 if description = flag n, then （flag n） ← 1	1 ≦ n ≦ 4
	BANKn		（BANK） ← n	n = 0

20.5　命令の個別説明

20.5.1　加算命令

（1）ADD r, m　　Add data memory to general register

① 命令コード



② 機　能

CMP = 0 のとき　　（r）←（r）＋（m）

ジェネラル・レジスタの内容にデータ・メモリの内容を加算し，結果をジェネラル・レジスタへ格納します。

CMP=1 のとき　　（r）＋（m）

結果はレジスタに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

加算の結果，桁上げがあればキャリー・フラグCYをセットし，桁上げがなければキャリー・フラグCYをリセットします。

加算の結果がゼロ以外になったときは，コンペア・フラグCMP に関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で加算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で加算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

加算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをPSWORDのBCDフラグによって指定します。

③ **例1**

ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H - 0.0FH）が指定されているとき（RPH = 0, RPL = 0），0.03H番地の内容に0.2FH番地の内容を加算した結果を0.03H番地に格納します。

（0.03H）← （0.03H）＋（0.2FH）

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H   ；データ・メモリのバンクを0
        MOV   RPH, #00H    ；ジェネラル・レジスタのバンクを0
        MOV   RPL, #00H    ；ジェネラル・レジスタのロウ・アドレスを0
        ADD   MEM003, MEM02F
```

例2

ジェネラル・レジスタとしてバンク0のロウ・アドレス2（0.20H - 0.2FH）が指定されているとき（RPH = 0, RPL = 4），0.23H番地の内容に0.2FH番地の内容を加算した結果を0.23H番地に格納します。

（0.23H）← （0.23H）＋（0.2FH）

```
MEM023  MEM   0.23H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H   ；データ・メモリのバンクを0
        MOV   RPH, #00H    ；ジェネラル・レジスタのバンクを0注
        MOV   RPL, #04H    ；ジェネラル・レジスタのロウ・アドレスを2
        ADD   MEM023, MEM02F
```

注

レジスタ	RP							
	RPH				RPL			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	バンク				ロウ・アドレス			B C D
	0	0	0	0				

システム・レジスタ中のRP（ジェネラル・レジスタ・ポインタ）の割り当ては前頁の図のようになります。
したがって，ジェネラル・レジスタにバンク0とロウ・アドレス2を設定するためには，RPHに00Hを，RPLに04Hを格納しなければなりません。
この場合，BCDフラグをリセットしているので以降の算術演算は2進4ビット演算となります。

例3

0.03H番地の内容に0.6FH番地の内容を加算した結果を0.03H番地に格納します。このとき，IXE＝1，IXH＝0，IXM＝4，IXL＝0すなわちIX＝0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

（0.03H）←（0.03H）＋（0.6 FH）
　　└ インデクス・レジスタの内容0.04Hとデータ・メモリのアドレス0.2FHをOR演算したアドレス

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   RPH, #00H          ; ジェネラル・レジスタのバンクを0
         MOV   RPL, #00H          ; ジェネラル・レジスタのロウ・アドレスを0
         MOV   IXH, #00H          ; IX←00001000000B
         MOV   IXM, #04H          ;
         MOV   IXL, #00H          ;
         SET1  IXE                ; IXE フラグ←1
         ADD   MEM003, MEM02F     ; IX                    00001000000B（0.40H）
                                  ; バンク・オペランド  OR）00000101111B（0.2FH）
                                  ; 指定アドレス          00001101111B（0.6FH）
```

例4

0.03H番地の内容に0.3FH番地の内容を加算した結果を0.03H番地に格納します。このとき，IXE＝1，IXH＝0，IXM＝1，IXL＝0すなわちIX＝0.10Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリ0.3FHを指定することができます。

（0.03H）←（0.03H）＋（0.3FH）
　　└ インデクス・レジスタ0.10Hの内容とデータ・メモリのアドレス0.2FHをOR演算したアドレス

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   BANK, #00H
         MOV   RPH, #00H          ; ジェネラル・レジスタのバンクを0
```

```
MOV   RPL, #00H        ; ジェネラル・レジスタのロウ・アドレスを0
MOV   IXH, #00H        ; IX ← 00000010000B (0.10H) 注
MOV   IXM, #01H
MOV   IXL, #00H
SET1  IXE              ; IXEフラグ ←1
ADD   MEM003, MEM02F   ; IX                    00000010000B (0.10H)
                       ; バンク・オペランド  OR) 00000101111B (0.2FH)
                       ; 指定アドレス          00100111111B (0.3FH)
```

注

レジスタ	IX											
	IXH				IXM				IXL			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	MPE	0	0	0	0	ロウ・アドレス			カラム・アドレス			
		バンク										

システム・レジスタ中のIX（インデクス・レジスタ）の割り当ては上図のようになります。したがって，IX＝0.10Hにするためには，IXHに00Hを，IXMに01Hを，IXLに00Hを格納しなければなりません。

この場合，MPE（メモリ・ポインタ・イネーブル）フラグをリセットしているので，ジェネラル・レジスタ間接転送のときのMP（メモリ・ポインタ）は無効になります。

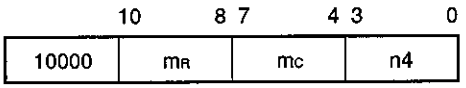
④ 注　意

ADD r, m命令の第1オペランドはジェネラル・レジスタのカラム・アドレスです。したがって，次のように書いた場合，ジェネラル・レジスタのカラム・アドレスは03Hになります。

```
MEM013   MEM   0.13H
MEM02F   MEM   0.2FH
         MOV   RPH, #00H      ; ジェネラル・レジスタのバンクを0   ★
         MOV   RPL, #00H      ; ジェネラル・レジスタのロウ・アドレスを0   ★
         ADD   MEM013, MEM02F
```

（2）ADD m, #n4　　Add immediate data to data memory

① 命令コード



② 機　能

CMP＝0のとき　　（m）←（m）＋n4

データ・メモリの内容にイミーディエト・データを加算し，結果をデータ・メモリへ格納します。

CMP＝1のとき　　（m）＋n4

結果はデータ・メモリに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

加算の結果，桁上げがあればキャリー・フラグCYをセットし，桁上げがなければキャリー・フラグCYをリセットします。

加算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP＝0）で加算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP＝1）で加算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

加算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをPSWORDのBCDフラグによって指定します。

③ 例1

0.2FH番地の内容に5を加算し，結果を0.2FH番地に格納します。

（0.2FH）←（0.2FH）＋5

```
MEM02F  MEM    0.2FH
        ADD    MEM02F, #05H
```

例2

0.6FH番地の内容に5を加算した結果を0.6FH番地に格納します。このとき，IXE＝1，IXH＝0，IXM＝4，IXL＝0すなわちIX＝0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

(0.6FH) ← (0.6FH) ＋05H
　　└インデクス・レジスタの内容0.40Hとデータ・メモリのアドレス
　　0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H     ; データ・メモリのバンクを0
        MOV   IXH, #00H      ; IX ← 00001000000B (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE            ; IXEフラグ ← 1
        ADD   MEM02F, #05H   ; IX                    00001000000B (0.40H)
                             ; バンク・オペランド  OR) 00000101111B (0.2FH)
                             ; 指定アドレス          00001101111B (0.6FH)
```

例3

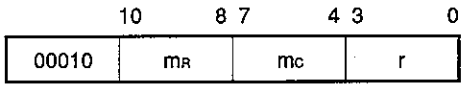
0.2FH番地の内容に5を加算した結果を0.2FH番地に格納します。このとき，IXE＝1，IXH＝0，IXM＝0，IXL＝0すなわちIX＝0.00Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.2FH番地を指定することができます。

(0.2FH) ← (0.2FH) ＋05H
　　└インデクス・レジスタの内容0.00Hとデータ・メモリのアドレス
　　0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H     ; データ・メモリのバンクを0
        MOV   IXH, #00H      ; IX ← 00000000000B
        MOV   IXM, #00H
        MOV   IXL, #00H
        SET1  IXE            ; IXEフラグ ← 1
        ADD   MEM02F, #05H   ; IX                    00000000000B (0.00H)
                             ; バンク・オペランド  OR) 00000101111B (0.2FH)
                             ; 指定アドレス          00000101111B (0.2FH)
```

（3）ADDC r, m　　Add data memory to general register with carry flag

① 命令コード



② 機　能

CMP = 0のとき　　（r）←（r）＋（m）＋CY

ジェネラル・レジスタの内容にデータ・メモリの内容とキャリー・フラグCYの値を加算し，結果をrで示すジェネラル・レジスタへ格納します。

CMP = 1のとき　　（r）＋（m）＋CY

結果はレジスタに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

★　このADDC命令を使うことにより1ニブルを越える加算が簡単にできます。

加算の結果，桁上げがあればキャリー・フラグCYをセットし，桁上げがなければキャリー・フラグCYをリセットします。

加算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で加算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で加算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

加算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H - 0.0FH）が指定されているとき，0.0DH番地から0.0FH番地の12ビットの内容に0.2DH番地から0.2FH番地の12ビットの内容を加算した結果を，0.0DH番地から0.0FH番地の12ビットに格納します。

(0.0FH) ← (0.0FH) ＋ (0.2FH)

(0.0EH) ← (0.0EH) ＋ (0.2EH) ＋CY

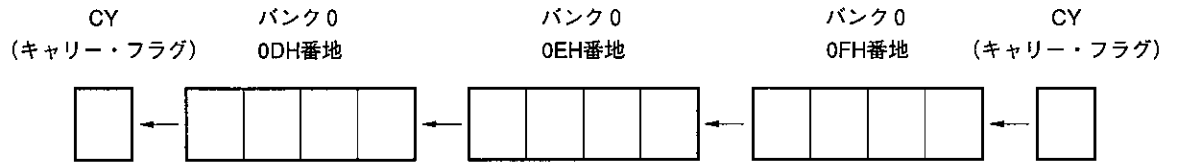
(0.0DH) ← (0.0DH) ＋ (0.2DH) ＋CY

```
MEM00D   MEM   0.0DH
MEM00E   MEM   0.0EH
MEM00F   MEM   0.0FH
```

```
MEM02D  MEM   0.2DH
MEM02E  MEM   0.2EH
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを0
        MOV   RPL, #00H       ;ジェネラル・レジスタのロウ・アドレスを0
        ADD   MEM00F, MEM02F ;下位ニブル
        ADDC  MEM00E, MEM02E
        ADDC  MEM00D, MEM02D ;上位ニブル
```

例2

ジェネラル・レジスタとしてバンク0のロウ・アドレス2（0.20H - 0.2FH）が指定されているとき，0.2DH番地から0.2FH番地の12ビットの内容をキャリー・フラグも含めて1ビット左にシフトします。




```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
MEM02D  MEM   0.2DH
MEM02E  MEM   0.2EH
MEM02F  MEM   0.2FH
        MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを0
        MOV   RPL, #04H       ;ジェネラル・レジスタのロウ・アドレスを2
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        ADDC  MEM00F, MEM02F
        ADDC  MEM00E, MEM02E
        ADDC  MEM00D, MEM02D
```

例 3

0.0DH番地から0.0FH番地の12ビットの内容に，0.40H番地から0.42H番地の12ビットの内容を加算した結果を，0.0DH番地から0.0FH番地の12ビットに格納します。

(0.0DH) ← (0.0DH) ＋ (0.40H)

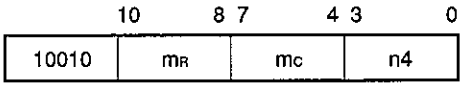
(0.0EH) ← (0.0EH) ＋ (0.41H) ＋ CY

(0.0FH) ← (0.0FH) ＋ (0.42H) ＋ CY

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H        ; データ・メモリのバンクを 0
        MOV   RPH, #00H         ; ジェネラル・レジスタのバンクを 0
        MOV   RPL, #00H         ; ジェネラル・レジスタのロウ・アドレスを 0
        MOV   IXH, #00H         ; IX ← 00001000000 (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE               ; IXEフラグ ← 1
        ADD   MEM00D, MEM000 ;  (0.0DH) ← (0.0DH) ＋ (0.40H)  ：下位ニブル
        ADDC  MEM00E, MEM001 ;  (0.0EH) ← (0.0EH) ＋ (0.41H)
        ADDC  MEM00F, MEM002 ;  (0.0FH) ← (0.0FH) ＋ (0.42H)  ：上位ニブル
```

（4）ADDC m, #n4　　Add immediate data to data memory with carry flag

① 命令コード



② 機　能

CMP = 0のとき　　（m）←（m）＋n4＋CY

データ・メモリの内容にイミーディエト・データとキャリー・フラグ（CY）の値を加算し，結果をデータ・メモリへ格納します。

CMP = 1のとき　　（m）＋n4＋CY

結果はデータ・メモリに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

加算の結果，桁上げがあればキャリー・フラグCYをセットし，桁上げがなければキャリー・フラグCYをリセットします。

加算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で加算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で加算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

加算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをPSWORDのBCDフラグによって指定します。

③ 例1

0.0DH番地から0.0FH番地の12ビットの内容に5を加算した結果を0.0DH番地から0.0FH番地に格納します。

(0.0FH) ← (0.0FH) ＋ 05H
(0.0EH) ← (0.0EH) ＋ CY
(0.0DH) ← (0.0DH) ＋ CY

```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        ADD   MEM00F, #05H
```

```
ADDC  MEM00E, #00H
ADDC  MEM00D, #00H
```

例2

0.4DH番地から0.4FH番地の12ビットの内容に5を加算した結果を0.4DH番地から0.4FH番地に格納します。

(0.4FH) ← (0.4FH) ＋ 05H
(0.4EH) ← (0.4EH) ＋ CY
(0.4DH) ← (0.4DH) ＋ CY

```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ; データ・メモリのバンクを0
        MOV   IXH, #00H       ; IX ← 00001000000B (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE             ; IXEフラグ ← 1
        ADD   MEM00F, #5      ; (0.4FH) ← (0.4FH) ＋ 5H
        ADDC  MEM00E, #0      ; (0.4EH) ← (0.4EH) ＋ CY
        ADDC  MEM00D, #0      ; (0.4DH) ← (0.4DH) ＋ CY
```

(5) INC AR　　Increment address register

① 命令コード

00111	000	1001	0000

② 機　能

AR ← AR ＋ 1

アドレス・レジスタARの内容をインクリメントします。

③ 例1

システム・レジスタ内のAR3からAR0（アドレス・レジスタ）の16ビットの内容に1を加算した結果をAR3からAR0に格納します。

AR0 ← AR0 ＋ 1
AR1 ← AR1 ＋ CY

AR2 ← AR2 ＋ CY

AR3 ← AR3 ＋ CY

```
INC AR
```

また，この命令を加算命令を用いて行うと以下のようになります。

```
ADD   AR0, #01H
ADDC  AR1, #00H
ADDC  AR2, #00H
ADDC  AR3, #00H
```

例 2

テーブル参照命令（詳細については，**10.2.3　テーブル参照**を参照してください）を用いてテーブル・データを16ビット（1アドレス）ごとにDBF（データ・バッファ）に転送します。

```
        ORG     10H
        DW      0F3FFH
        DW      0A123H
        DW      0FFF1H
        DW      0FFF5H
        DW      0FF11H
         ⋮
        MOV     AR3, #0H        ;テーブル・データのアドレス
        MOV     AR2, #0H        ;0010Hをアドレス・レジスタに設定します
        MOV     AR1, #1H
        MOV     AR0, #0H
LOOP:
        MOVT    DBF, @AR        ;テーブル・データがDBFに読み出されます  ★
         ⋮
(テーブル・データを参照する処理)
         ⋮
        INC     AR              ;アドレス・レジスタを1インクリメント
        BR      LOOP
```

④ 注　意

アドレス・レジスタ（AR0 - AR3）は，製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

（6）INC IX　　Increment indexregister

① 命令コード

00111	000	1000	0000

② 機　　能

IX ← IX ＋ 1

インデクス・レジスタIXの内容をインクリメントします。

③ 例1

システム・レジスタ内のIXH, IXM, IXL（インデクス・レジスタ）の計12ビットの内容に1を加算した結果をIXH, IXM, IXLに格納します。

IXL ← IXL ＋ 1

IXM ← IXM ＋ CY

IXH ← IXH ＋ CY

INC IX

この演算を加算命令を用いて行うと以下のようになります。

```
ADD    IXL, #01H
ADDC   IXM, #00H
ADDC   IXH, #00H
```

例2

インデクス・レジスタを用いてデータ・メモリ0.00H - 0.73Hの内容をすべて“0”にします。

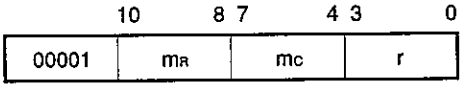
```
MEM000      MEM     0.00H
            MOV     IXH, #00H       ;インデクス・レジスタの内容をバンク0の00Hに設定します
            MOV     IXM, #00H       ;
            MOV     IXL, #00H
RAMクリア：
            SET1    IXE             ;IXEフラグ←1
            MOV     MEM000, #00H    ;インデクス・レジスタで示されるデータ・メモリに0を書き込みます
            CLR1    IXE             ;IXEフラグ← 0
            INC     IX
            SET2    CMP, Z          ;CMPフラグ←1, Zフラグ←1
            SUB     IXL, #03H       ;インデクス・レジスタの内容がバンク0の73Hになったかどうか
            SUB     IXM, #07H       ;をチェックします
```

```
SUB     IXH, #00H       ;
SKT1    Z               ;インデクス・レジスタの内容がバンク0の73Hになるまでループ
BR      RAMクリア       ;します
```

20.5.2　減算命令

（1）SUB r, m　　Subtract data memory from general register

① 命令コード



② 機　能

CMP = 0のとき　　（r）←（r）－（m）

ジェネラル・レジスタの内容からデータ・メモリの内容を減算し，結果をジェネラル・レジスタへ格納します。

CMP = 1のとき　　（r）－（m）

結果はレジスタに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

減算の結果，ボローがあればキャリー・フラグCYをセットし，ボローがなければキャリー・フラグCYをリセットします。

減算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で減算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で減算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

減算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H - 0.0FH）が指定されているとき（RPH = 0, RPL = 0），0.03H番地の内容から0.2FH番地の内容を減算した結果を0.03H番地に格納します。

（0.03H）←（0.03H）－（0.2FH）

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         SUB   MEM003, MEM02F
```

例 2

ジェネラル・レジスタとしてバンク 0 のロウ・アドレス 2（0.20H - 0.2FH）が指定されているとき（RPH = 0, RPL = 4），0.23H番地の内容から0.2FH番地の内容を減算した結果を0.23H番地に格納します。

（0.23H）←（0.23H）－（0.2FH）

```
MEM023   MEM   0.23H
MEM02F   MEM   0.2FH
         MOV   BANK, #00H      ;データ・メモリのバンクを 0
         MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを 0
         MOV   RPL, #04H       ;ジェネラル・レジスタのロウ・アドレスを 2
         SUB   MEM023, MEM02F
```

例 3

0.03H番地の内容から0.6FH番地の内容を減算した結果を0.03H番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

（0.03H）←（0.03H）－（0.6FH）

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   BANK, #00H      ;データ・メモリのバンクを 0
         MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを 0
         MOV   RPL, #00H       ;ジェネラル・レジスタのロウ・アドレスを 0
         MOV   IXH, #00H       ;IX ← 00001000000B（0.40H）
         MOV   IXM, #04H       ;
         MOV   IXL, #00H       ;
         SET1  IXE             ;IXEフラグ ← 1
         SUB   MEM003, MEM02F  ;IX                        00001000000B（0.40H）
                               ;バンク・オペランド    OR）00000101111B（0.2FH）
                               ;指定アドレス              00001101111B（0.6FH）
```

例 4

0.03H番地の内容から0.3FH番地の内容を減算した結果を0.03H番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 1, IXL = 0すなわちIX = 0.10Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.3FH番地を指定することができます。

(0.03H) ← (0.03H) − (0.3FH)

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   BANK, #00H       ; データ・メモリのバンクを 0
         MOV   RPH, #00H        ; ジェネラル・レジスタのバンクを 0
         MOV   RPL, #00H        ; ジェネラル・レジスタのロウ・アドレスを 0
         MOV   IXH, #00H        ; IX ← 00000010000B (0.10H)
         MOV   IXM, #01H        ;
         MOV   IXL, #00H        ;
         SET1  IXE              ; IXEフラグ ← 1
         SUB   MEM003, MEM02F   ; IX                    00000010000B (0.10H)
                                ; バンク・オペランド  OR) 00000101111B (0.2FH)
                                ; 指定アドレス           00000111111B (0.3FH)
```

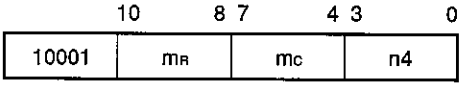
④ 注　意

SUB r, m命令の第 1 オペランドはジェネラル・レジスタ・アドレスでなければなりません。したがって，次のように書くと03H番地がレジスタとして指定されます。

```
MEM013   MEM   0.13H
MEM02F   MEM   0.2FH
         MOV   RPH, #00H        ; ジェネラル・レジスタのバンクを 0   ★
         MOV   RPL, #00H        ; ジェネラル・レジスタのロウ・アドレスを 0   ★
         SUB   MEM013, MEM02F
```

（2）SUB m, #n4　　Subtract immediate data from data memory

① 命令コード



② 機　能

CMP = 0のとき　　（m）←（m）－n4

データ・メモリの内容からイミーディエト・データを減算し，結果をデータ・メモリへ格納します。

CMP = 1のとき　　（m）－n4

結果はデータ・メモリに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

減算の結果，ボローがあればキャリー・フラグCYをセットし，ボローがなければキャリー・フラグCYをリセットします。

減算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で減算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で減算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

減算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

0.2FH番地の内容から5を減算し，結果を0.2FH番地に格納します。

（0.2FH）←（0.2FH）－5

```
MEM02F   MEM   0.2FH
         SUB   MEM02F, #05H
```

例2

0.6FH番地の内容から5を減算した結果を0.6FH番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

(0.6FH) ← (0.6FH) − 5
　　└インデクス・レジスタの内容0.40Hとデータ・メモリのアドレス
　　0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ; データ・メモリのバンクを 0
        MOV   IXH, #00H       ; IX ← 00001000000B (0.40H)
        MOV   IXM, #04H       ;
        MOV   IXL, #00H       ;
        SET1  IXE             ; IXEフラグ ← 1
        SUB   MEM02F, #05H    ; IX                    00001000000B (0.40H)
                              ; バンク・オペランド  OR) 00000101111B (0.2FH)
                              ; 指定アドレス          00001101111B (0.6FH)
```

例 3

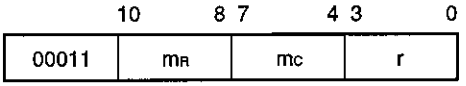
0.2FH番地の内容から 5 を減算した結果を0.2FH番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 0, IXL = 0すなわちIX = 0.00Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.2FH番地を指定することができます。

(0.2FH) ← (0.2FH) − 5
　　└インデクス・レジスタの内容0.00Hとデータ・メモリのアドレス
　　0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK0, #00H     ; データ・メモリのバンクを 0
        MOV   IXH, #00H       ; IX ← 00000000000B (0.00H)
        MOV   IXM, #00H       ;
        MOV   IXL, #00H       ;
        SET1  IXE             ; IXEフラグ ← 1
        SUB   MEM02F, #05H    ; IX                    00000000000B (0.00H)
                              ; バンク・オペランド  OR) 00000101111B (0.2FH)
                              ; 指定アドレス          00000101111B (0.2FH)
```

（3）SUBC r, m　　Subtract data memory from general register with carry flag

① 命令コード



② 機　能

CMP＝0のとき　　（r）←（r）－（m）－CY

ジェネラル・レジスタの内容からデータ・メモリの内容とキャリー・フラグCYの値を減算し，結果をジェネラル・レジスタへ格納します。このSUBC命令を使うことにより1ニブルを越える減算が簡単にできます。

CMP＝1のとき　　（r）－（m）－CY

結果はレジスタに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

減算の結果，ボローがあればキャリー・フラグCYをセットし，ボローがなければキャリー・フラグCYをリセットします。

減算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP＝0）で減算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP＝1）で減算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

減算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H - 0.0FH）が指定されているとき，0.0DH番地から0.0FH番地の12ビットの内容から，0.2DH番地から0.2FH番地の12ビットの内容を減算し，結果を0.0DH番地から0.0FH番地の12ビットに格納します。

(0.0FH) ← (0.0FH) － (0.2FH)

(0.0EH) ← (0.0EH) － (0.2EH) － CY

(0.0DH) ← (0.0DH) ＋ (0.2DH) － CY

```
MEM00D   MEM   0.0DH
MEM00E   MEM   0.0EH
MEM00F   MEM   0.0FH
```

```
MEM02D  MEM   0.2DH
MEM02E  MEM   0.2EH
MEM02F  MEM   0.2FH
        SUB   MEM00F, MEM02F ; 下位ニブル
        SUBC  MEM00E, MEM02E
        SUBC  MEM00D, MEM02D ; 上位ニブル
```

例 2

0.0DH番地から0.0FH番地の12ビットの内容から，0.40H番地から0.42H番地の12ビットの内容を減算した結果を0.0DH番地から0.0FH番地の12ビットに格納します。

(0.0DH) ← (0.0DH) − (0.40H)

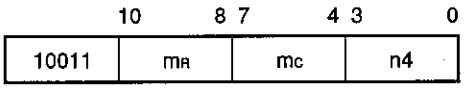
(0.0EH) ← (0.0EH) − (0.41H) − CY

(0.0FH) ← (0.0FH) ＋ (0.42H) − CY

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ; データ・メモリのバンクを 0
        MOV   RPH, #00H       ; ジェネラル・レジスタのバンクを 0
        MOV   RPL, #00H       ; ジェネラル・レジスタのロウ・アドレスを 0
        MOV   IXH, #00H       ; IX ← 00001000000B (0.40H)
        MOV   IXM, #04H       ;
        MOV   IXL, #00H       ;
        SET1  IXE             ; IXEフラグ ← 1
        SUB   MEM00D, MEM000  ; (0.0DH) ← (0.0DH) − (0.40H)
        SUBC  MEM00E, MEM001  ; (0.0EH) ← (0.0EH) − (0.41H)
        SUBC  MEM00F, MEM002  ; (0.0FH) ← (0.0FH) − (0.42H)
```

（4）SUBC m, #n4　　Subtract immediate data from data memory with carry flag

① 命令コード



② 機　能

CMP＝0のとき　　（m）←（m）－n4－CY

データ・メモリの内容からイミーディエト・データとキャリー・フラグCYの値を減算し，結果をデータ・メモリへ格納します。

CMP＝1のとき　　（m）－n4－CY

結果はデータ・メモリに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

減算の結果，ボローがあればキャリー・フラグCYをセットし，ボローがなければキャリー・フラグCYをリセットします。

減算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP＝0）で減算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP＝1）で減算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

減算には2進演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

0.0DH番地から0.0FH番地の12ビットの内容から5を減算した結果を，0.0DH番地から0.0FH番地に格納します。

(0.0FH) ← (0.0FH) − 05H
(0.0EH) ← (0.0EH) − CY
(0.0DH) ← (0.0DH) − CY

```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        SUB   MEM00F, #05H
        SUBC  MEM00E, #00H
        SUBC  MEM00D, #00H
```

例2

0.4DH番地から0.4FH番地の12ビットの内容から5を減算した結果を0.4DH番地から0.4FH番地に格納します。

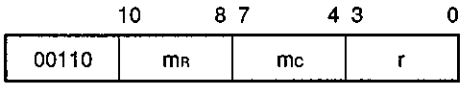
(0.4FH) ← (0.4FH) − 05H
(0.4EH) ← (0.4EH) − CY
(0.4DH) ← (0.4DH) − CY

```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        MOV   IXH, #00H       ;IX ← 00001000000B (0.40H)
        MOV   IXM, #04H       ;
        MOV   IXL, #00H       ;
        SET1  IXE             ;IXEフラグ ← 1
        SUB   MEM00F, #5      ; (0.4FH) ← (0.4FH) − 5
        SUBC  MEM00E, #0      ; (0.4EH) ← (0.4EH) − CY
        SUBC  MEM00D, #0      ; (0.4DH) ← (0.4DH) − CY
```

20.5.3　論理演算命令

（1）OR r, m　　OR between general register and data memory

① 命令コード



② 機　能

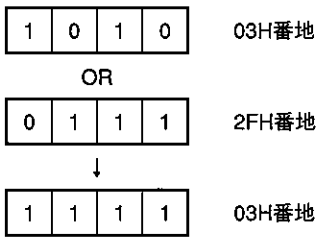
（r）←（r）∨（m）

ジェネラル・レジスタの内容とデータ・メモリの内容との論理和（OR）の結果をジェネラル・レジスタへ格納します。

③ 例

0.03H番地の内容（1010B）と0.2FH番地の内容（0111B）をOR演算した結果（1111B）を0.03H番地へ格納します。

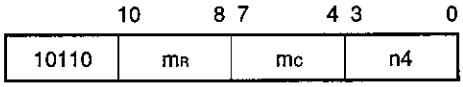
（0.03H）←（0.03H）∨（0.2FH）




```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   MEM003, #1010B
         MOV   MEM02F, #0111B
         OR    MEM003, MEM02F
```

（2）OR m, #n4　　OR between data memory and immediate data

① 命令コード



② 機　能

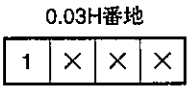
（m）←（m）∨ n4

データ・メモリの内容とイミーディエト・データとの論理和（OR）の結果をデータ・メモリへ格納します。

③ 例1

0.03H番地のビット3（MSB）をセットします。

（0.03H）←（0.03H）∨ 1000B



×：don't care

```
MEM003   MEM   0.03H
         OR    MEM003, #1000B
```

例2

0.03H番地のすべてのビットをセットします。

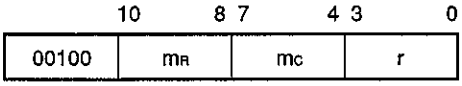
```
MEM003   MEM   0.03H
         OR    MEM003, #1111B
```

または

```
MEM003   MEM   0.03H
         MOV   MEM003, #0FH
```

（3）AND r, m　　AND between general register and data memory

① 命令コード



② 機　能

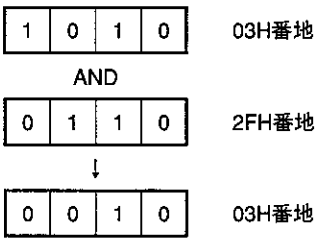
（r）←（r）∧（m）

ジェネラル・レジスタの内容とデータ・メモリの内容との論理積（AND）の結果をジェネラル・レジスタへ格納します。

③ 例

0.03H 番地の内容（1010B）と0.2FH番地の内容（0110B）をAND演算した結果（0010B）を0.03H番地へ格納します。

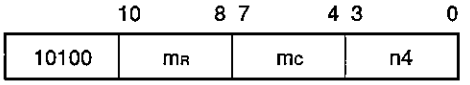
(0.03H) ← (0.03H) ∧ (0.2FH)




```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   MEM003, #1010B
         MOV   MEM02F, #0110B
         AND   MEM003, MEM02F
```

（4）AND m, #n4　　AND between data memory and immediate data

① 命令コード



② 機　能

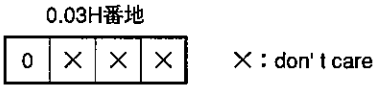
（m）←（m）∧ n4

データ・メモリの内容とイミーディエト・データとの論理積（AND）の結果をデータ・メモリへ格納します。

③ 例1

0.03H番地のビット3（MSB）をリセットします。

（0.03H）←（0.03H）∧ 0111B



×：don' t care

```
MEM003   MEM   0.03H
         AND   MEM003, #0111B
```

例2

0.03H番地のすべてのビットをリセットします。

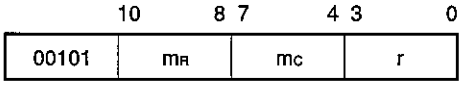
```
MEM003   MEM   0.03H
         AND   MEM003, #0000B
```

または

```
MEM003   MEM   0.03H
         MOV   MEM003, #00H
```

（5）XOR r, m　　Exclusive OR between general register and data memory

① 命令コード



② 機　　能

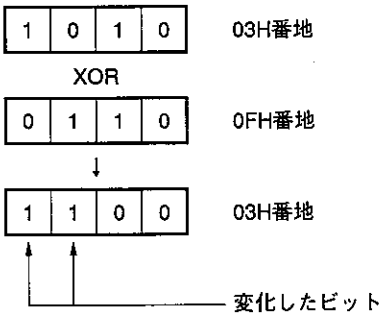
（r）←（r）∀（m）

ジェネラル・レジスタの内容とデータ・メモリの内容との排他的論理和（XOR）の結果をジェネラル・レジスタへ格納します。

③ 例1

0.03H番地の内容と0.0FH番地の内容を比較した結果，異なるビットをセットして0.03H番地へ格納し，0.03H番地がすべてリセット（0.03H番地と0.0FH番地の内容が同じ）されていればLBL1へジャンプし，それ以外であればLBL2へジャンプします。

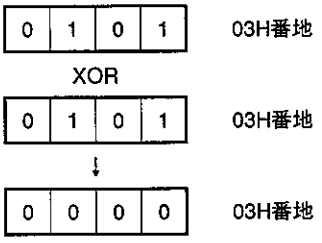
この例はオルタネート・スイッチの状態（0.03H番地の内容）と内部状態（0.0FH番地の内容）を比較し，変化したスイッチの処理へと分岐するときなどの例です。




```
MEM003   MEM   0.03H
MEM00F   MEM   0.0FH
         XOR   MEM003, MEM00F
         SKNE  MEM003, #00H
         BR    LBL1
         BR    LBL2
```

例2

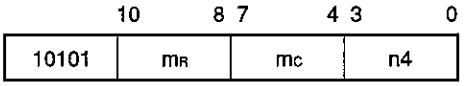
0.03H番地の内容をクリアします。




```
MEM003   MEM   0.03H
         XOR   MEM003, MEM003
```

（6）XOR m, #n4　　Exclusive OR between data memory and immediate data

① 命令コード



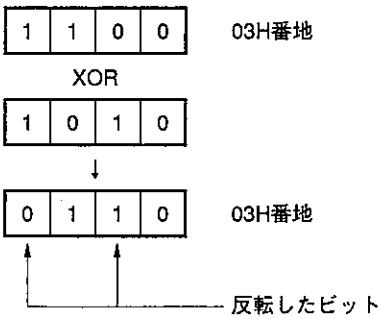
② 機　能

（m）← （m）∀ n4

データ・メモリの内容とイミーディエト・データとの排他的論理和（XOR）の結果をデータ・メモリへ格納します。

③ 例

0.03H番地のビット1とビット3を反転して03H番地へ格納します。

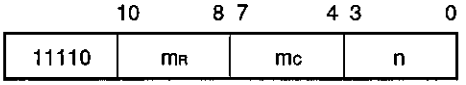



```
MEM003   MEM   0.03H
         XOR   MEM003, #1010B
```

20.5.4　判断命令

（1）SKT m, #n　　Skip next instruction if data memory bits are true

① 命令コード



② 機　能

CMP ← 0, if（m）∧ n = n, then skip

データ・メモリの内容とイミーディエト・データnの論理積の結果がnと等しければ次の1命令をスキップします（NOP命令として実行します）。 ★

③ 例1

03H番地のビット0が“1”ならばAAAへジャンプし，“0”ならばBBBへジャンプします。

```
SKT     03H, #0001B
BR      BBB
BR      AAA
```

例2

03H番地のビット0とビット1がともに“1”ならば次の命令をスキップします。

```
SKT     03H, #0011B
```

		b_3	b_2	b_1	b_0	
スキップ条件	03H	×	×	1	1	×：don't care

例3

次の2つの命令の実行結果は同じです。

```
SKT     13H, #1111B
SKE     13H, #0FH
```

(2) SKF m, #n　　Skip next instruction if data memory bits are false

① 命令コード

10	8 7	4 3	0
11111	m_R	m_C	n

② 機　能

CMP ← 0, if （m）∧ n = 0, then skip

データ・メモリの内容とイミーディエト・データnの論理積の結果が0のとき，次の1命令をスキップします（NOP命令として実行します）。

★

③ 例1

13H番地のビット2が“0”ならばデータ・メモリの0FH番地の内容にイミーディエト・データの00Hを格納し，“1”ならばABCへジャンプします。

```
MEM013  MEM   0.13H
MEM00F  MEM   0.0FH
        SKF   MEM013, #0100B
        BR    ABC
        MOV   MEM00F, #00H
```

例 2

29H 番地のビット 3 とビット 0 がともに “0” ならば次の命令をスキップします。

```
SKF     29H, #1001B
```

		b_3	b_2	b_1	b_0	
スキップ条件	29H	0	×	×	0	×；don' t care

例 3

次の 2 つの命令の実行結果は同じです。

```
SKF     34H, #1111B
SKE     34H, #00H
```

20.5.5　比較命令

（1）SKE m, #n4　　Skip if data memory equal to immediate data

① 命令コード

10	8 7	4 3	0
01001	m_R	mc	n4

② 機　能

（m）－ n4, skip if zero

データ・メモリの内容がイミーディエト・データの値と等しいとき，次に続く 1 命令をスキップします（NOP命令として実行します）。 ★

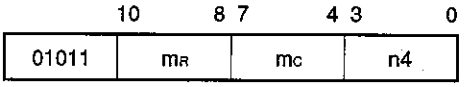
③ 例

24H番地の内容が 0 ならば24H 番地に0FHを転送し，0 でなければOPE1へジャンプします。

```
MEM024  MEM   0.24H
        SKE   MEM024, #00H
        BR    OPE1
        MOV   MEM024, #0FH
OPE1：
```

（2）SKNE m, #n4　　SKip if data memory not equal to immediate data

① 命令コード



② 機　能

（m）－n4, skip if not zero

データ・メモリの内容がイミーディエト・データの値と異なるとき，次に続く1命令をスキップします（NOP命令として実行します）。

★

③ 例

1FH番地の内容が1で，かつ1EH番地の内容が3ならばXYZへジャンプし，そうでなければABCへジャンプします。

8ビットの比較をする場合は以下のように組み合わせて使います。

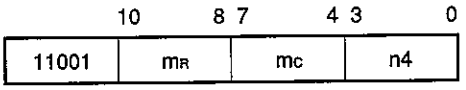

```
MEM01E   MEM    0.1EH
MEM01F   MEM    0.1FH
         SKNE   MEM01F, #01H
         SKE    MEM01E, #03H
         BR     ABC
         BR     XYZ
```

また，コンペア・フラグ，ゼロ・フラグを用いて上記の内容と同じことを行う場合は，以下のようになります。

```
MEM01E   MEM    0.1EH
MEM01F   MEM    0.1FH
         SET2   CMP, Z          ; CMPフラグ←1，Zフラグ←1
         SUB    MEM01F, #01H
         SUBC   MEM01E, #03H
         SKT1   Z
         BR     ABC
         BR     XYZ
```

（3）SKGE m, #n4　　Skip if data memory greater than or equal to immediate data

① 命令コード



② 機　能

（m）－ n4, skip if not borrow

データ・メモリの内容がイミーディエト・データの値以上のとき，次に続く1命令をスキップします（NOP命令として実行します）。 ★

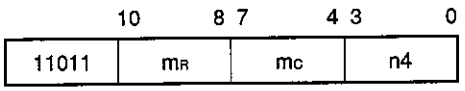
③ 例

1FH番地（上位）および2FH番地（下位）に格納されている8ビット・データがイミーディエト・データ17Hより大きければRETし，そうでなければRETSKするプログラム例。

```
MEM01F  MEM   0.1FH
MEM02F  MEM   0.2FH
        SKGE  MEM01F, #1
        RETSK
        SKNE  MEM01F, #1
        SKLT  MEM02F, #8      ; 7＋1
        RET
        RETSK
```

（4）SKLT m, #n4　　Skip if data memory less than immediate data

① 命令コード



② 機　能

（m）－ n4, skip if borrow

データ・メモリの内容がイミーディエト・データの値未満のとき，次に続く1命令をスキップします（NOP命令として実行します）。 ★

③ 例

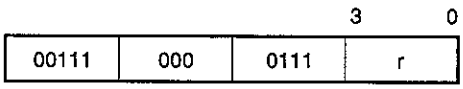
10H番地の内容がイミーディエト・データ“6”以上であれば，0FH番地の内容に01Hを格納し，“6”より小さければ0FH番地の内容に02Hを格納するプログラムです。

```
MEM00F  MEM   0.0FH
MEM010  MEM   0.10H
        MOV   MEM00F, #02H
        SKLT  MEM010, #06H
        MOV   MEM00F, #01H
```

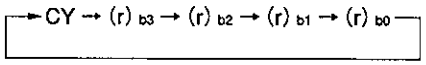
20.5.6　回転命令

（1）RORC r　　Rotate right general register with carry flag

① 命令コード



② 機　能



r で示すジェネラル・レジスタの内容をキャリー・フラグも含めて 1 ビット右に回転します。

③ 例 1

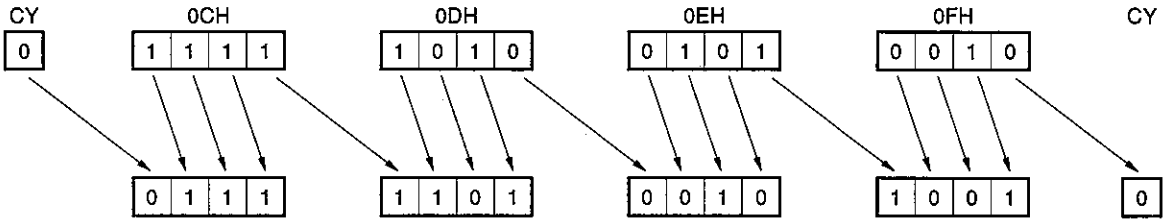
ジェネラル・レジスタとしてバンク 0 のロウ・アドレス 0（0.00H - 0.0FH）が指定されているとき（RPH = 0, RPL = 0），0.00H番地の値（1000B）を右に 1 ビット回転し，0100Bにします。

(0.00H) ← (0.00H) ÷ 2

```
MEM000  MEM   0.00H
        MOV   RPH, #00H      ; ジェネラル・レジスタのバンクを 0
        MOV   RPL, #00H      ; ジェネラル・レジスタのロウ・アドレスを 0
        CLR1  CY             ; CYフラグ ← 0
        RORC  MEM000
```

例 2

ジェネラル・レジスタとしてバンク 0 のロウ・アドレス 0（0.00H - 0.0FH）が指定されているとき（RPH = 0, RPL = 0），データ・バッファDBFの内容0FA52Hを右に 1 ビット回転し，DBFの内容を7D29Hにします。

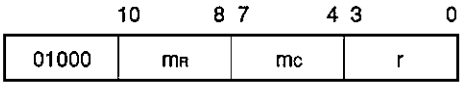



```
MEM00C  MEM   0.0CH
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   RPH, #00H      ; ジェネラル・レジスタのバンクを 0
        MOV   RPL, #00H      ; ジェネラル・レジスタのロウ・アドレスを 0
        CLR1  CY             ; CYフラグ ← 0
        RORC  MEM00C
        RORC  MEM00D
        RORC  MEM00E
        RORC  MEM00F
```

20.5.7　転送命令

（1）LD r, m　　**Load data memory to general register**

① 命令コード



② 機　能

（r）←（m）

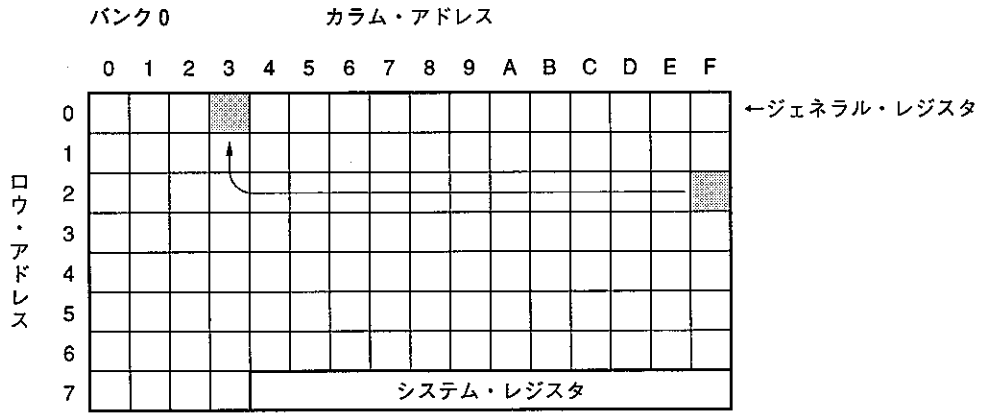
データ・メモリの内容をジェネラル・レジスタへ格納します。

③ 例1

0.03H番地に0.2FH番地の内容を格納します。

(0.03H) ← (0.2FH)

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   RPH, #00H      ; ジェネラル・レジスタのバンクを 0   ★
        MOV   RPL, #00H      ; ジェネラル・レジスタのロウ・アドレスを 0   ★
        LD    MEM003, MEM02F
```



例2.

0.6FH番地の内容を0.03H番地に格納します。このときIXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリ0.6FH番地を指定することができます。

IXH ← 00H

IXM ← 04H

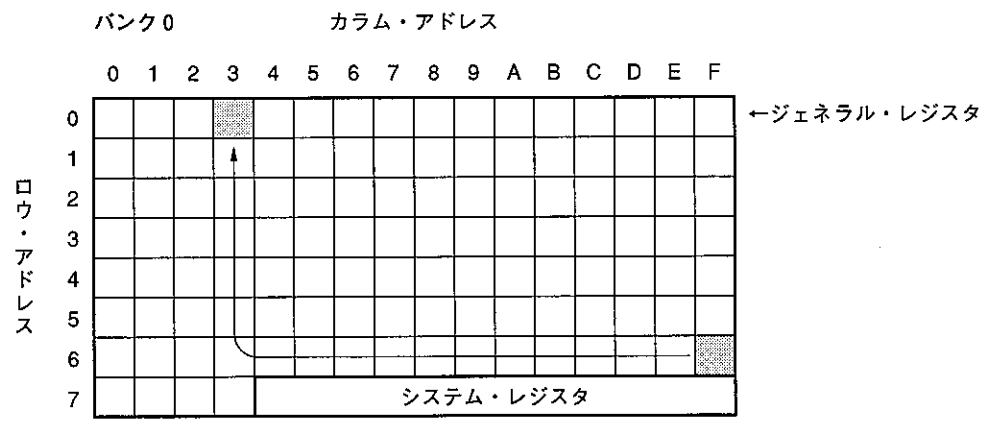
IXL ← 00H

IXEフラグ ← 1

(0.03H) ← (0.6FH)

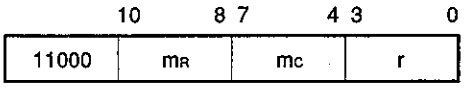
└ インデクス・レジスタの内容040Hとデータ・メモリの0.2FHをOR演算したアドレス

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   IXH, #00H          ; IX ← 00001000000B (0.40H)
         MOV   IXM, #04H
         MOV   IXL, #00H
         SET1  IXE                ; IXEフラグ ← 1
         LD    MEM003, MEM02F
```



（2）ST m, r　　Store general register to data memory

① 命令コード



② 機　能

（m）←（r）

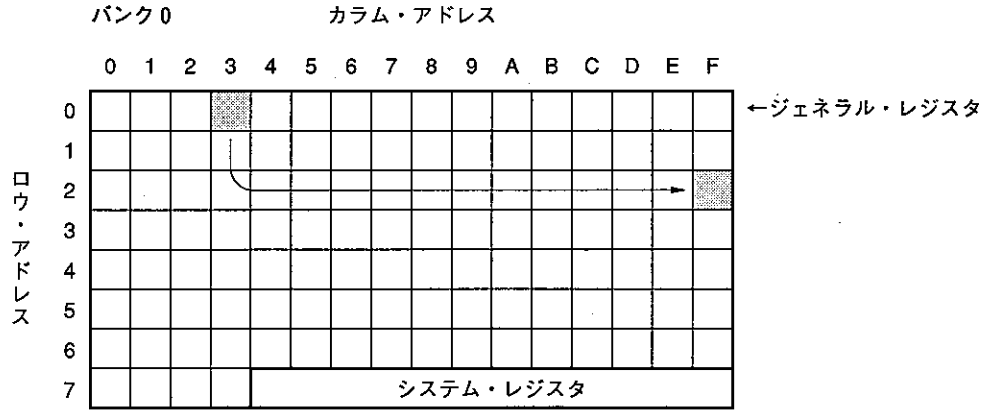
ジェネラル・レジスタの内容をデータ・メモリへ格納します。

③ 例1

0.2FH番地に0.03H番地の内容を格納します。

（0.2FH）←（0.03H）

```
MOV     RPH, #00H       ;ジェネラル・レジスタのバンクを0        ★
MOV     RPL, #00H       ;ジェネラル・レジスタのロウ・アドレスを0  ★
ST      2FH, 03H        ;データ・メモリにジェネラル・レジスタの内容を転送
```



例2

0.18H番地から0.1FH番地に0.00H番地の内容を格納します。インデクス・レジスタでデータ・メモリ（18H - 1FH番地）を指定します。

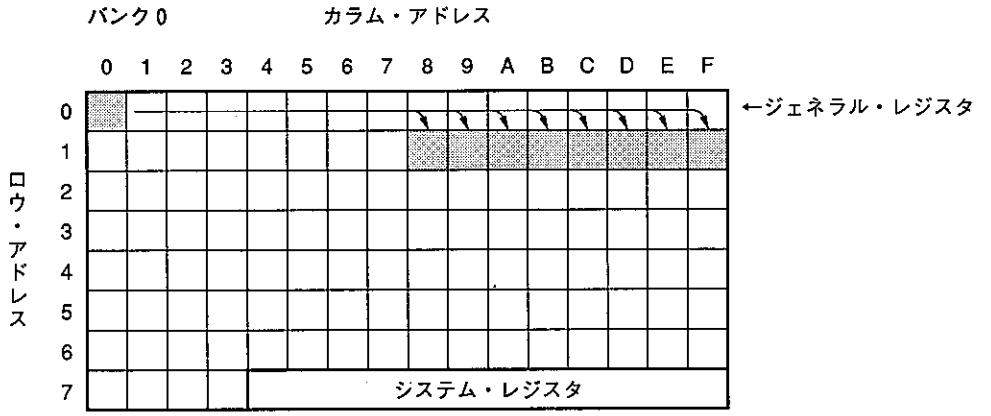
（0.18H）←（0.00H）

（0.19H）←（0.00H）

⋮

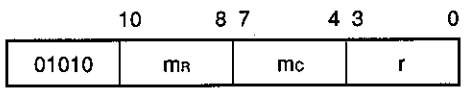
（0.1FH）←（0.00H）

```
        MOV   IXH, #00H        ; IX ← 0000000000B (0.00H)
        MOV   IXM, #00H
        MOV   IXL, #00H        ; データ・メモリに0.00H番地を指定
MEM018  MEM   0.18H
MEM000  MEM   0.00H
LOOP1 :
        SET1  IXE              ; IXEフラグ ← 1
        ST    MEM018, MEM000   ; (0.1×H) ← (0.00H)
        CLR1  IXE              ; IXEフラグ ← 0
        INC   IX               ; IX ← IX + 1
        SKGE  IXL, #08H
        BR    LOOP1
```



(3) MOV @r, m　　Move data memory to destination indirect

① 命令コード



② 機　能

MPE = 1のとき

(MP, (r)) ← (m)

MPE = 0のとき

(BANK, mR, (r)) ← (m)

データ・メモリの内容をジェネラル・レジスタの内容でアドレスするデータ・メモリへ格納します。

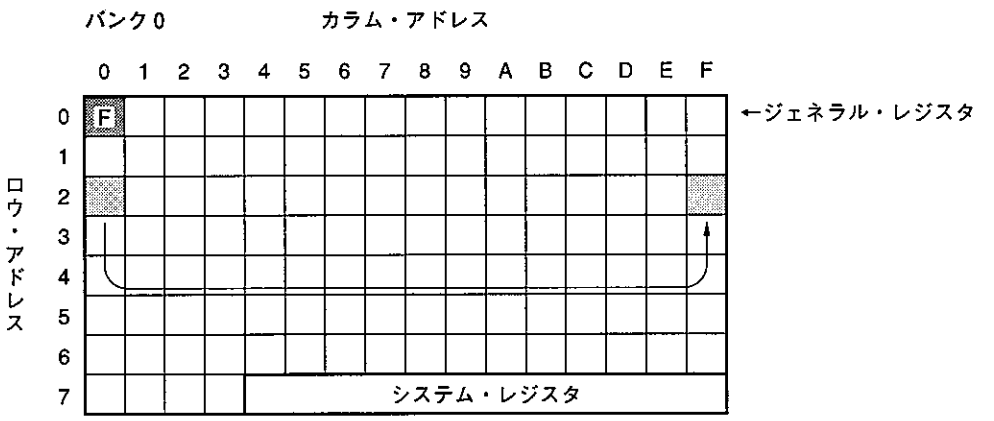
MPE = 0のときには，同一バンクの同一ロウ・アドレス内での転送となります。

③　例1

MPE フラグを0にして，0.2FH番地に0.20H番地の内容を格納します。格納先のデータ・メモリは，転送元と同一のロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

（0.2FH）←（0.20H）

```
MEM000   MEM   0.00H
MEM020   MEM   0.20H
         CLR1  MPE                ; MPEフラグ←0
         MOV   MEM000, #0FH       ; ジェネラル・レジスタにカラム・アドレス設定
         MOV   @MEM000, MEM020    ; 格納
```

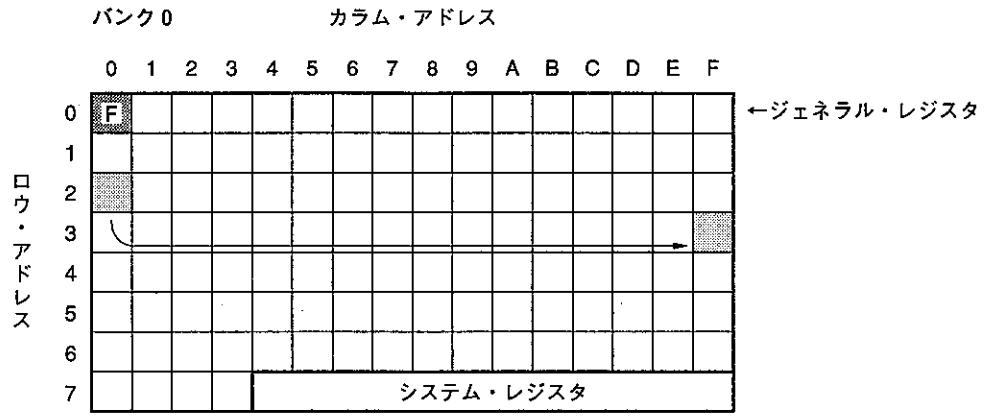


例2

MPE フラグを1にして，0.3FH番地に0.20H番地の内容を格納します。格納先のデータ・メモリは，メモリ・ポインタMPの内容がロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

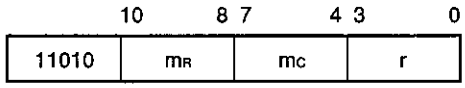
（0.3FH）←（0.20H）

```
MEM000   MEM   0.00H
MEM020   MEM   0.20H
         MOV   RPH, #00H          ; ジェネラル・レジスタのバンクを0
         MOV   RPL, #00H          ; ジェネラル・レジスタのロウ・アドレスを0
         MOV   00H, #0FH          ; ジェネラル・レジスタにカラム・アドレス設定
         MOV   MPH, #00H          ; メモリ・ポインタにロウ・アドレスを設定
         MOV   MPL, #03H          ;
         SET1  MPE                ; MPEフラグ←1
         MOV   @MEM000, MEM020    ; 格納
```



(4) MOV m, @r　　Move data memory to destination indirect

① 命令コード



② 機　能

MPE = 1のとき

(m) ← (MP, (r))

MPE = 0のとき

(m) ← (BANK, mR, (r))

ジェネラル・レジスタの内容でアドレスするデータ・メモリの内容をデータ・メモリへ格納します。

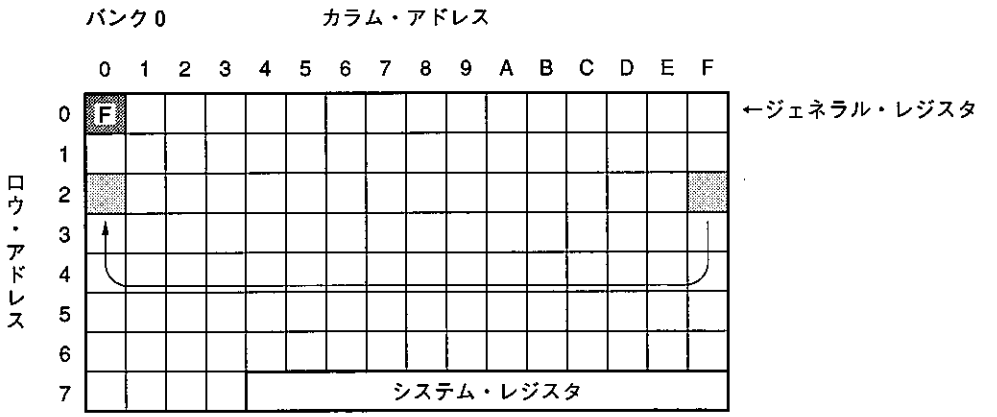
MPE = 0のときには，同一バンクの同一ロウ・アドレス内での転送となります。

③ 例1

MPEフラグを0にして，0.20H番地に0.2FH番地の内容を格納します。転送元のデータ・メモリは，格納先と同一のロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

(0.20H) ← (0.2FH)

```
MEM000   MEM    0.00H
MEM020   MEM    0.20H
         CLR1   MPE              ; MPEフラグ ← 0
         MOV    MEM000, #0FH     ; ジェネラル・レジスタにカラム・アドレス設定
         MOV    MEM020, @MEM000  ; 格納
```

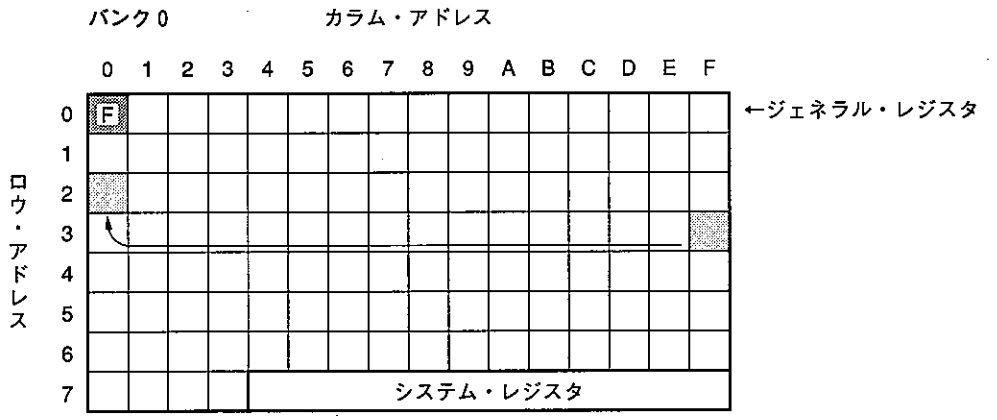


例2

MPEフラグを1にして，0.20H番地に0.3FH番地の内容を格納します。転送元のデータ・メモリは，メモリ・ポインタMPの内容がロウ・アドレスで，ジェネラル・レジスタの0.00番地の内容がカラム・アドレスです。

(0.20H) ← (0.3FH)

```
MEM000  MEM   0.00H
MEM020  MEM   0.20H
        MOV   MEM000, #0FH      ;ジェネラル・レジスタにカラム・アドレス設定
        MOV   MPH, #00H         ;メモリ・ポインタにロウ・アドレスを設定
        MOV   MPL, #03H         ;
        SET1  MPE               ;MPEフラグ←1
        MOV   MEM020, @MEM000   ;格納
```



（5）MOV m, #n4　　Move immediate data to data memory

① 命令コード

10	8 7	4 3	0
11101	mR	mC	n4

② 機　能

（m）← n4

データ・メモリにイミーディエト・データを格納します。

③ 例1

データ・メモリの0.50H 番地にイミーディエト・データの0AH を格納します。

（0.50H）← 0AH

```
MEM050  MEM   0.50H
        MOV   MEM050, #0AH
```

例2

データ・メモリとして0.00H番地が指定されているとき，IXH = 0, IXM = 3, IXL = 2, IXEフラグ = 1が設定されていると，0.32H番地にイミーディエト・データの07Hを格納します。

（0.32H）← 07H

```
MEM000  MEM   0.00H
        MOV   IXH, #00H      ; IX ← 00000110010B（0.32H）
        MOV   IXM, #03H
        MOV   IXL, #02H
        SET1  IXE            ; IXEフラグ ← 1
        MOV   MEM000, #07H
```

（6）MOVT DBF, @AR　　Move program memory data specified by AR to DBF

① 命令コード

00111	000	0001	0000

② 機　　能

SP ← SP － 1 , ASR ← PC, PC ← AR,

DBF ← (PC) , PC ← ASR, SP ← SP ＋ 1

アドレス・レジスタARでアドレスされるプログラム・メモリの内容をデータ・バッファDBFに格納します。

この命令は一時的にスタックを 1 レベル使用するため，サブルーチン，割り込みなどのネスティングに注意してください。

③ 例

システム・レジスタ内のアドレス・レジスタAR3, AR2, AR1, AR0の値によりテーブル・データの16ビットをデータ・バッファDBF3, DBF2, DBF1, DBF0に転送します。

```
;*
;**   テーブル・データ
;*
ORG     10H
DW      0000000000000000B       ; (0000H)
DW      1010101111001101B       ; (0ABCDH)
                :
;*
;**   テーブル参照プログラム
;*
MOV     AR3, #00H       ; AR3 ← 00H       アドレス・レジスタに
MOV     AR2, #00H       ; AR2 ← 00H       0011Hを設定
MOV     AR1, #01H       ; AR1 ← 01H
MOV     AR0, #01H       ; AR0 ← 01H
MOVT    DBF, @AR        ; ← アドレス0011H番地のデータをDBFに転送
```

この場合，DBFには以下のようにデータが格納されます。

DBF3 = 0AH

DBF2 = 0BH

DBF1 = 0CH

DBF0 = 0DH

（7）PUSH AR　　Push address register

①　命令コード

00111	000	1101	0000

②　機　　能

SP ← SP − 1，

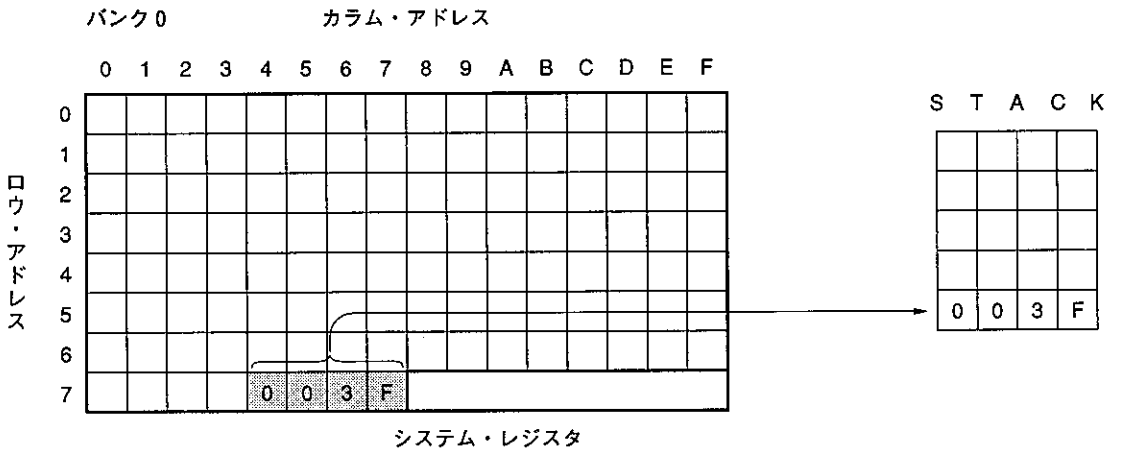
ASR ← AR

スタック・ポインタSPをデクリメントしたあと，スタック・ポインタで指定されるアドレス・スタック・レジスタにアドレス・レジスタARの値を格納します。

③　例1

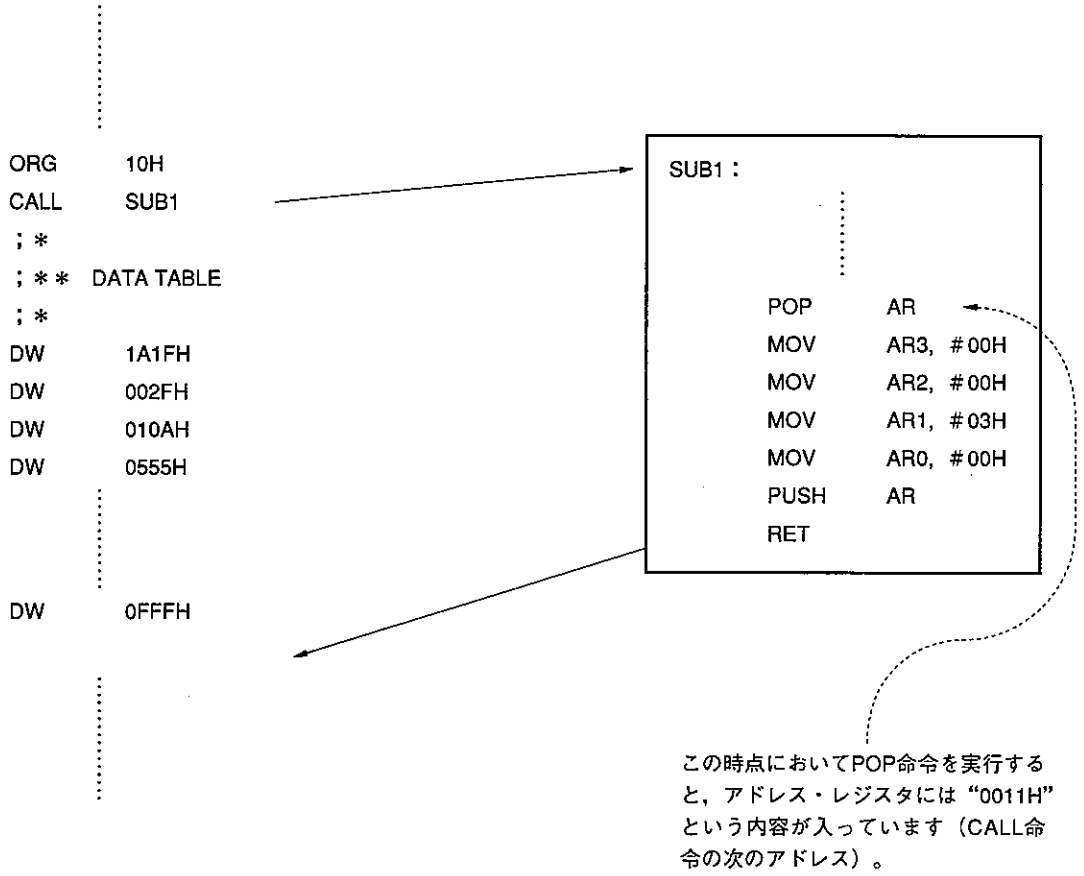
アドレス・レジスタに003FHを設定し，スタックに格納します。

```
MOV     AR3, #00H
MOV     AR2, #00H
MOV     AR1, #03H
MOV     AR0, #0FH
PUSH    AR
```



例 2

CALL命令に続いてデータ・テーブルがある場合にサブルーチンからの戻り番地（データ・テーブルの次の番地）をアドレス・レジスタに設定しリターンします。



（8）POP AR　　Pop address register

①　命令コード

00111	000	1100	0000

②　機　　能

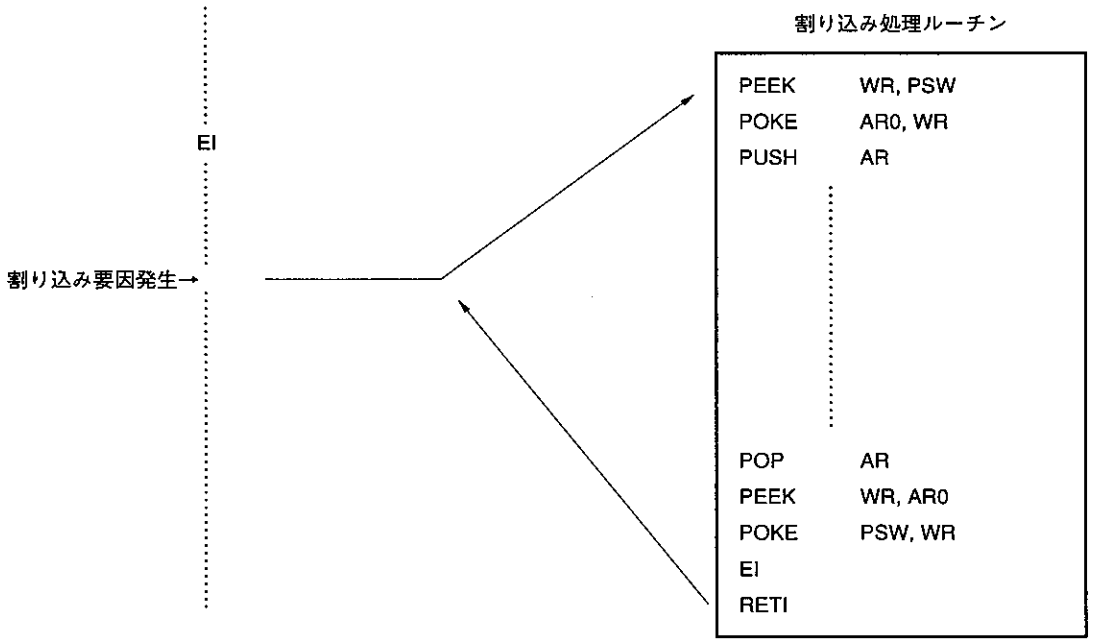
AR ← ASR,

SP ← SP ＋ 1

スタック・ポインタで示されるアドレス・スタック・レジスタの内容をアドレス・レジスタARに取り出したあと，スタック・ポインタSPをインクリメントします。

③　例

割り込み処理を行う場合，割り込み処理ルーチン内にてPSW が変化してしまう場合に，割り込み処理の始めでPSW の内容をWRを介してアドレス・レジスタに転送し，PUSH命令によりアドレス・スタック・レジスタに退避し，リターンする前にPOP命令によりアドレス・レジスタに復帰させWRを介してPSWに転送します。



(9) PEEK WR, rf　　Peek register file to window register

① 命令コード

10	8 7	4 3	0
00111	rfR	0011	rfc

② 機　　能

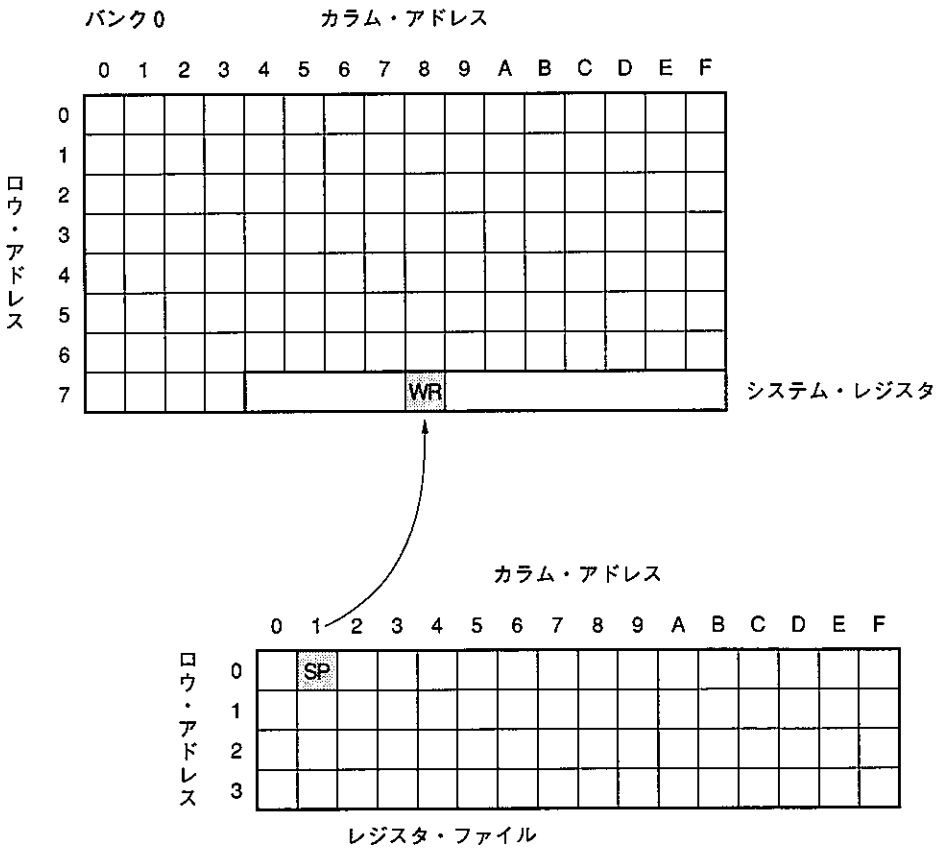
WR ← (rf)

レジスタ・ファイルの内容をウインドウ・レジスタWRに格納します。

③ 例

レジスタ・ファイル内の01H 番地のスタック・ポインタSPの内容をウインドウ・レジスタに格納します。

```
PEEK    WR, SP
```



(10) POKE rf, WR　　Poke window register to register file

① 命令コード

10	8 7	4 3	0
00111	rfR	0010	rfc

② 機　能

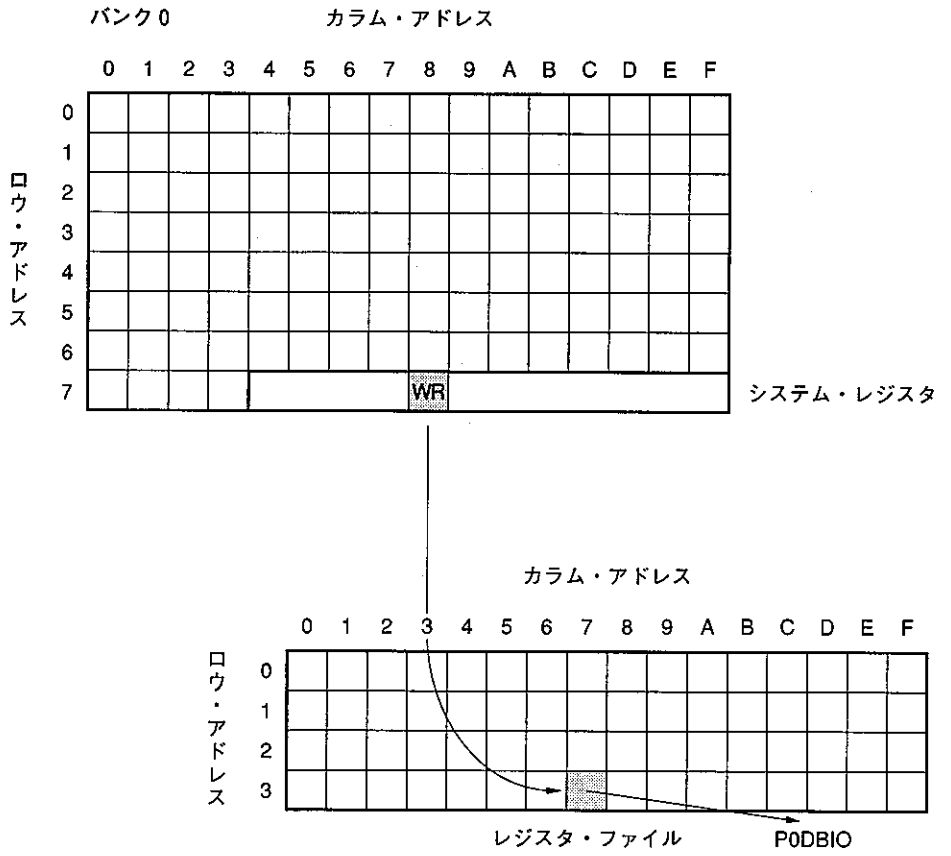
(rf) ← WR

ウインドウ・レジスタWRの内容をレジスタ・ファイルに格納します。

③ 例

ウインドウ・レジスタを介してレジスタ・ファイルのP0DBIOにイミーディエト・データの0FHを格納します。

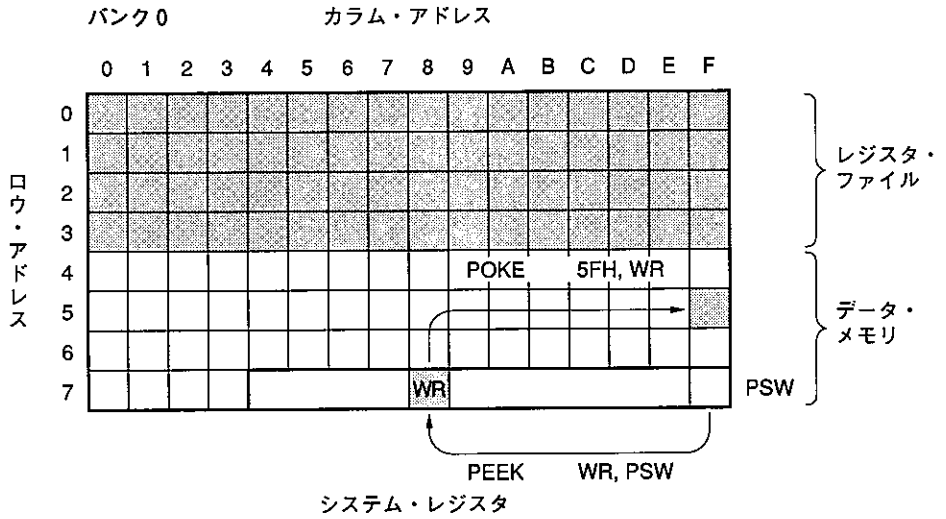
```
MOV     WR, #0FH
POKE    P0DBIO, WR      ; P0D0, P0D1, P0D2, P0D3をすべて出力モードに設定
```



④ 注　意

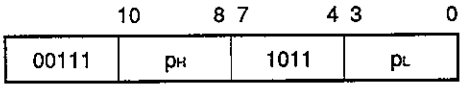
プログラム上は，レジスタ・ファイルの40Hから7FH番地に，データ・メモリのアドレス40Hから7FH番地と同じメモリがあるように見えます。したがって，PEEK,POKE命令ではレジスタ・ファイル以外にデータ・メモリの各バンクの40H番地から7FH番地までをアクセスすることができます。たとえば以下のような使い方も可能です。

```
MEM05F  MEM     0.5FH
        PEEK    WR, PSW      ；システム・レジスタ内のPSW（7FH）の内容をWRに格納
        POKE    MEM05F, WR   ；WRの内容をデータ・メモリの5FH番地に格納
```



(11) GET DBF, p　　Get peripheral data to data buffer

① 命令コード



② 機　能

DBF ← (p)

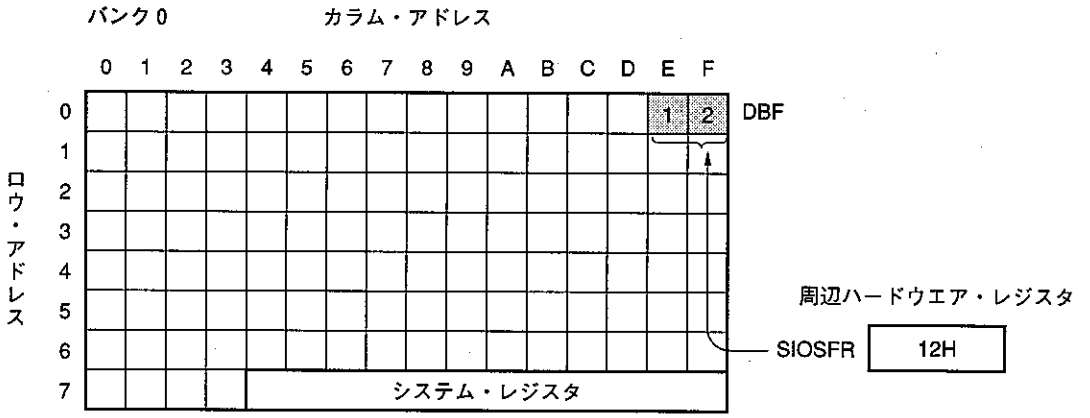
周辺ハードウエア・レジスタの内容をデータ・バッファDBFに格納します。

DBFはバンク・レジスタの値に関係なく，データ・メモリのBANK0の0CHから0FH番地の16ビットの領域となります。

③ 例

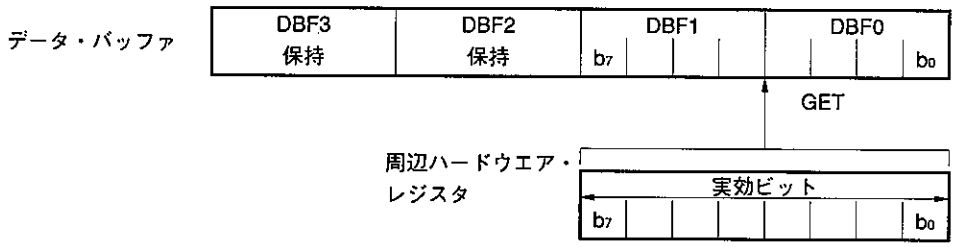
シリアル・インタフェースのシフト・レジスタSIOSFRの内容（8ビット）をデータ・バッファのDBF0, DBF1に格納します。

```
GET     DBF, SIOSFR
```



④ 注　意

データ・バッファは16ビットで構成されています。ただし，周辺ハードウエアによってアクセスされるビット数は異なります。たとえば実効ビット長が8ビットの周辺ハードウエア・レジスタに対してGET命令を実行した場合は，データ・バッファDBFの下位8ビット（DBF1, DBF0）に周辺ハードウエア・レジスタの内容が格納されます。



(12) PUT p, DBF　　Put data buffer to peripheral

① 命令コード

10	8 7	4 3	0
00111	pH	1010	pL

② 機　能

(p) ← DBF

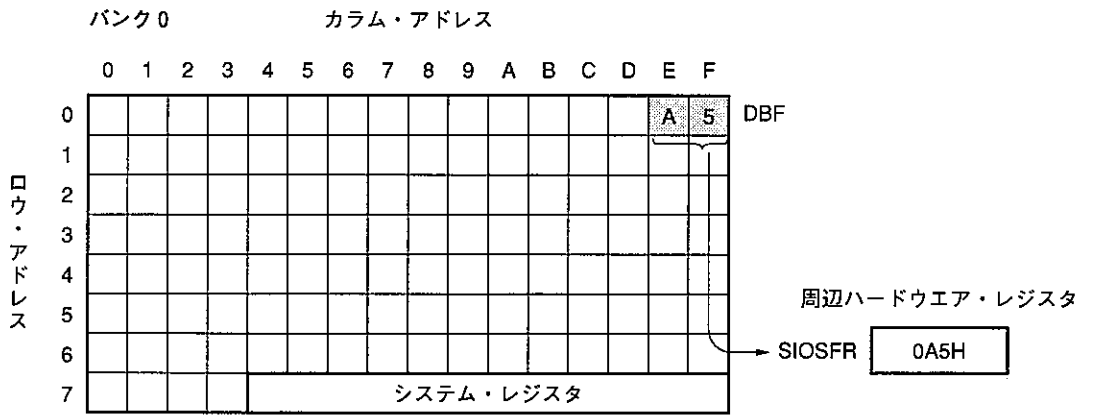
データ・バッファDBFの内容を周辺ハードウエア・レジスタに格納します。

DBFはバンク・レジスタの値に関係なく，データ・メモリのBANK0の0CHから0FH番地の16ビットの領域となります。

③ 例

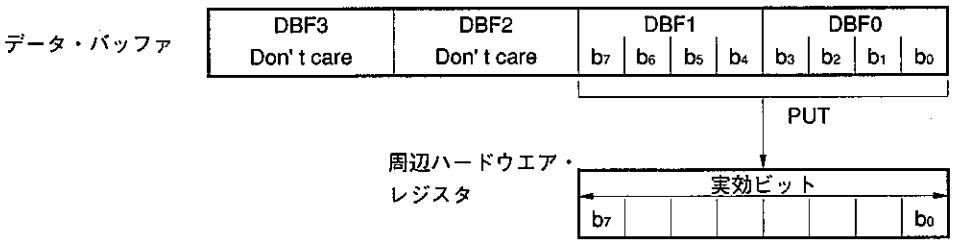
データ・バッファのDBF1とDBF0にそれぞれ0AH, 05Hを設定し，周辺レジスタのひとつであるシリアル・インタフェース用のシフト・レジスタ（SIOSFR）に転送します。

```
MOV     BANK,  #00H  ；データ・メモリのバンクを 0
MOV     DBF0,  #05H
MOV     DBF1,  #0AH
PUT     SIOSFR, DBF
```



④ 注　意

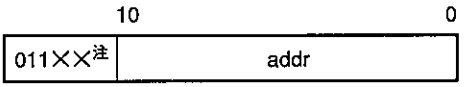
データ・バッファは，16ビットで構成されています。ただし，周辺ハードウエアによってアクセスされるビット数は異なります。たとえば実効ビット長が 8 ビットの周辺ハードウエア・レジスタに対してPUT命令を実行した場合は，データ・バッファDBFの下位 8 ビット（DBF1, DBF0）の内容を周辺ハードウエア・レジスタに格納します（DBF3, DBF2の内容は無効です）。



20.5.8　分岐命令

（1）BR addr　　Branch to the address

① 命令コード



注 ④ 注　意 参照。

② 機　能

PC ← addr

addrで指定するアドレスに分岐します。

③ 例

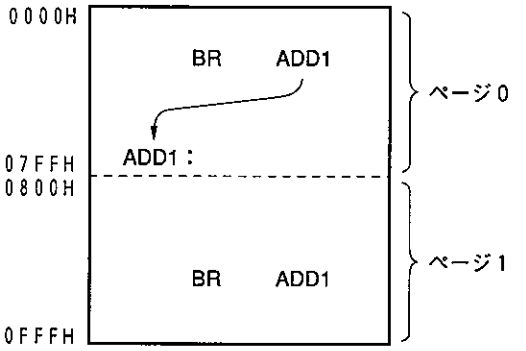
```
FLY     LAB     0FH         ; FLY = 0FHを定義
          :
        BR      FLY         ; 0F番地にジャンプします
          :
        BR      LOOP1       ; LOOP1にジャンプします
          :
        BR      $＋2        ; 現在番地より2つ下の番地へジャンプします
          :
        BR      $－3        ; 現在番地より3つ上の番地へジャンプします
          :
LOOP1：
```

④ 注　意

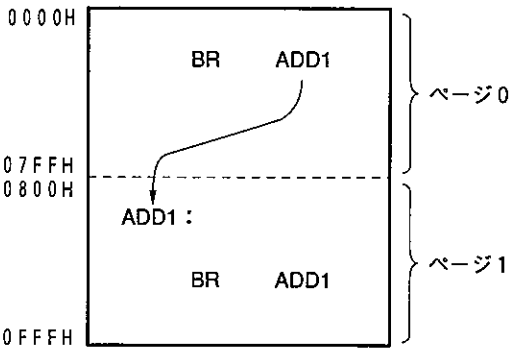
BR命令はアセンブラで記述する上ではページの概念はなく，ROMアドレス0000H - 1FFFH 番地の間で，同一記述で使用することができます。ただし，ページ0内（0000H - 07FFH番地）へのBR命令と，ページ1内（07FFH - 0FFFH番地）へのBR命令とでは，オペレーション・コードが異なります。

ページ0内でのオペレーション・コードは“0C”，ページ1内でのオペレーション・コードは“0D”です。これらはアセンブルする際，17Kシリーズのアセンブラを使用すればアセンブラがジャンプ先を参照して自動的に変換します。

オペレーション・コードが0Cとなる場合（ジャンプ先のアドレスがページ0内にある場合）

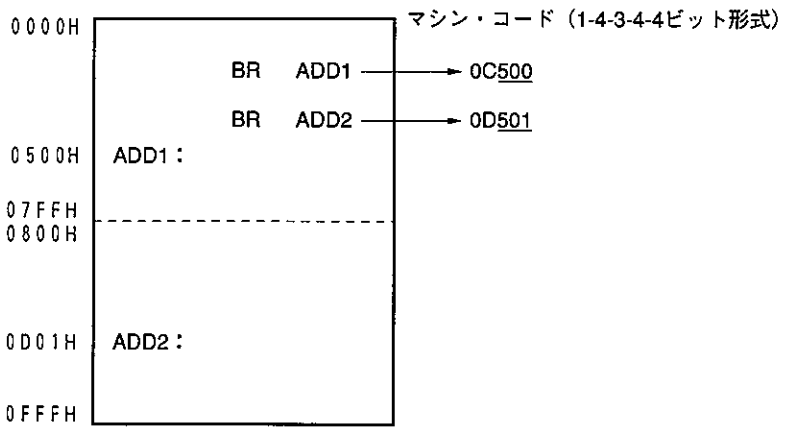


オペレーション・コードが0Dとなる場合（ジャンプ先のアドレスがページ1内にある場合）



ディバグ時，直接機械語でパッチ修正を行う場合は，オペレーション・コードに注意する必要があります（“0C”，“0D”のそれぞれの変換を行う必要があります）。

また，BR命令のジャンプ先が0800H番地 - 0FFFH番地の場合にはアドレスの変換も必要です。つまり，0800H番地を000H番地として以降1番地ずつインクリメントされたアドレスとなります。



注意　μPD17145サブシリーズの各製品ごとにページ数が異なります。

（2）BR @AR　　Branch to the address specified by address register

① 命令コード

00111	000	0100	0000

② 機　　能

PC ← AR

アドレス・レジスタARが指定するプログラム・アドレスに分岐します。

③ 例1

アドレス・レジスタAR（AR0 - AR3）に003FHを設定し，BR @AR命令により003FH番地へジャンプします。

```
MOV     AR3,  #00H    ; AR3 ← 00H
MOV     AR2,  #00H    ; AR2 ← 00H
MOV     AR1,  #03H    ; AR1 ← 03H
MOV     AR0,  #0FH    ; AR0 ← 0FH
BR      @AR           ; 003FH番地へジャンプ
```

例2

データ・メモリの0.10H番地の内容によって以下のように分岐先を変えます。

0.10Hの内容		分岐先のレーベル
00H	→	AAA
01H	→	BBB
02H	→	CCC
03H	→	DDD
04H	→	EEE
05H	→	FFF
06H	→	GGG
07H	→	HHH
08H - 0FH	→	ZZZ

```
;*
;**ジャンプ・テーブル
;*
ORG     10H
BR      AAA
BR      BBB
BR      CCC
```

```
        BR      DDD
        BR      EEE
        BR      FFF
        BR      GGG
        BR      HHH
        BR      ZZZ
                :
MEM010  MEM     0.10H
        MOV     AR3, #00H     ; AR3 ← 00H   ARを001×Hにする
        MOV     AR2, #00H     ; AR2 ← 00H
        MOV     AR1, #01H     ; AR1 ← 01H
        MOV     RPH, #00H     ; ジェネラル・レジスタのバンクを0
        MOV     RPL, #02H     ; ジェネラル・レジスタのロウ・アドレスを1
        ST      AR0, MEM010   ; AR0 ← 0.10H
        SKLT    AR0, #08H                                                    ★
        MOV     AR0, #08H     ; AR0の内容が08Hよりも大きい場合は，AR0の内容を08Hにする  ★
        BR      @AR
```

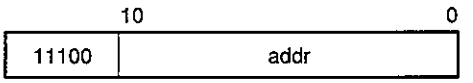
④ 注　意

アドレス・レジスタ（AR0 - AR3）は製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

20.5.9　サブルーチン命令

（1）CALL addr　　Call subroutine

① 命令コード



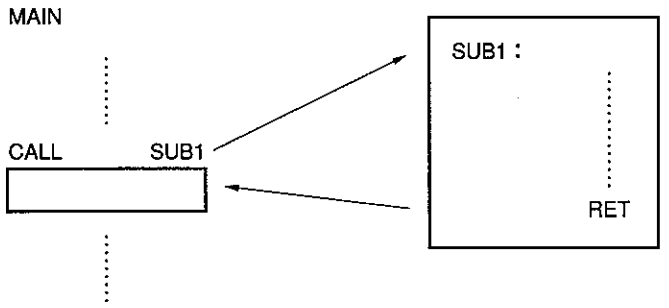
② 機　能

SP ← SP − 1 , ASR ← PC

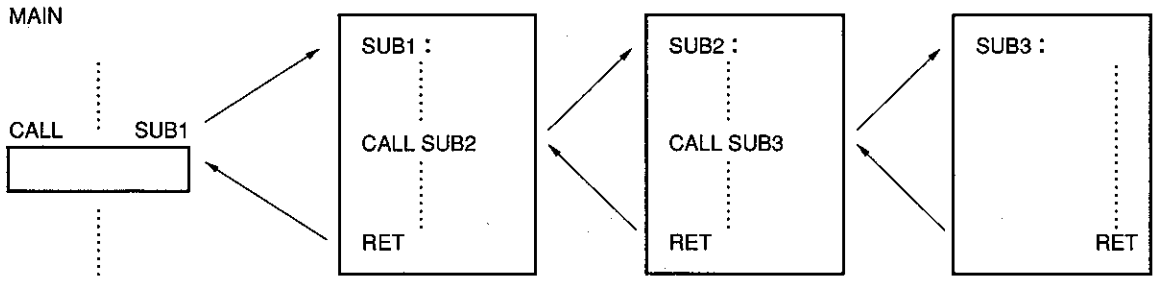
PC ← addr, PAGE ← 0

プログラム・カウンタPCの値をインクリメントし，スタックに格納したのち，addrで指定するサブルーチンに分岐します。

③ 例1



例2



（2）CALL @AR　　Call subroutine specified by address register

① 命令コード

00111	000	0101	0000

② 機　　能

SP ← SP − 1，

ASR ← PC，

PC ← AR

プログラム・カウンタPCの値をインクリメントし，スタックに格納したのち，アドレス・レジスタARが指定するアドレスから始まるサブルーチンに分岐します。

③　例1

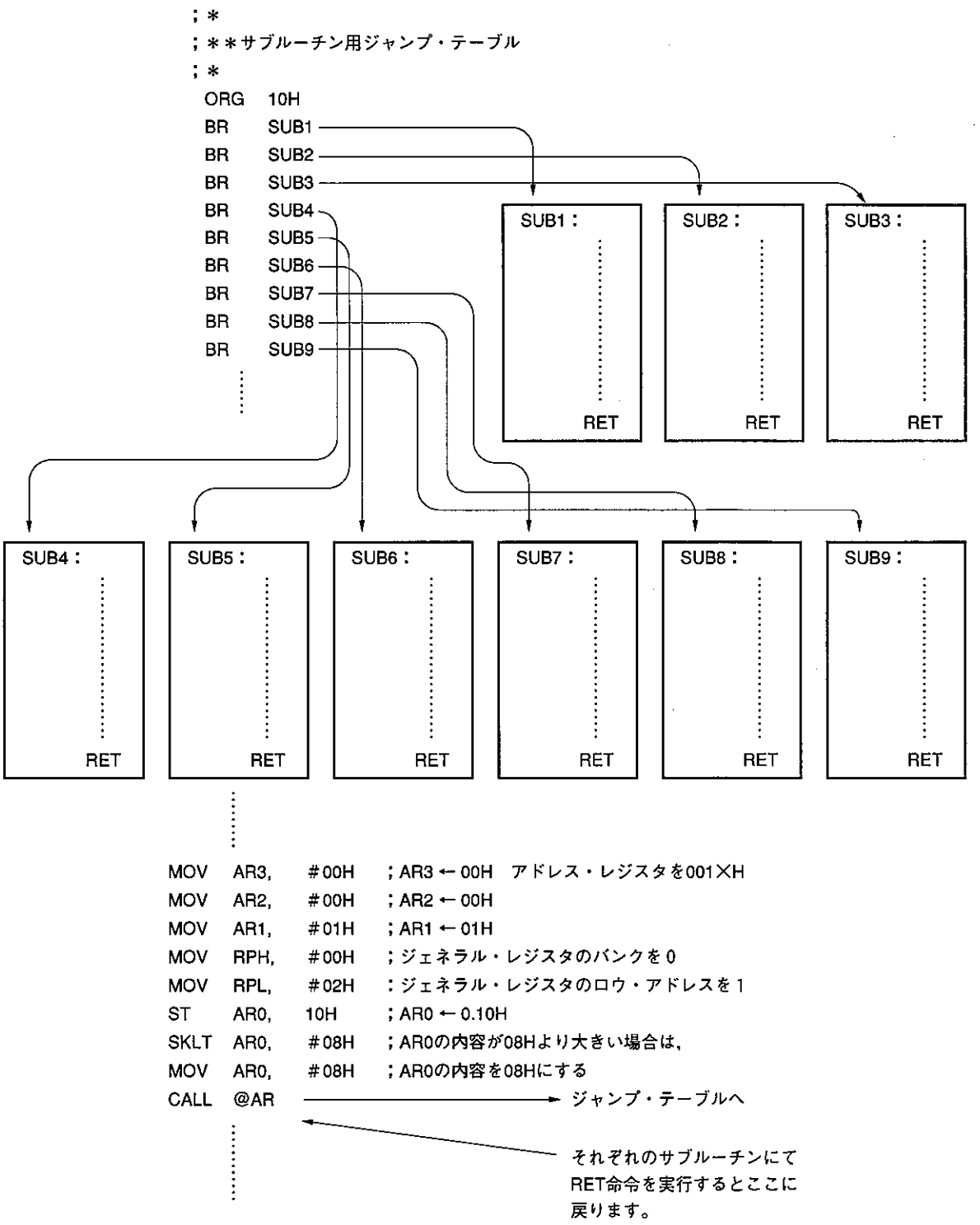
アドレス・レジスタAR（AR0 - AR3）に0020Hを設定し，CALL @AR命令により0020H番地のサブルーチンをコールします。

```
MOV     AR3,  #00H      ; AR3 ← 00H
MOV     AR2,  #00H      ; AR2 ← 00H
MOV     AR1,  #02H      ; AR1 ← 02H
MOV     AR0,  #00H      ; AR0 ← 00H
CALL    @AR             ; 0020H番地のサブルーチンをコール
```

例2

データ・メモリの0.10H番地の内容によって以下のサブルーチンをコールします。

0.10Hの内容		サブルーチン名
00H	→	SUB1
01H	→	SUB2
02H	→	SUB3
03H	→	SUB4
04H	→	SUB5
05H	→	SUB6
06H	→	SUB7
07H	→	SUB8
08H - 0FH	→	SUB9



★
★

④ 注　意

アドレス・レジスタ（AR0-AR3）は製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

（3）RET　　Return to the main program from subroutine

① 命令コード

00111	000	1110	0000

② 機　　能

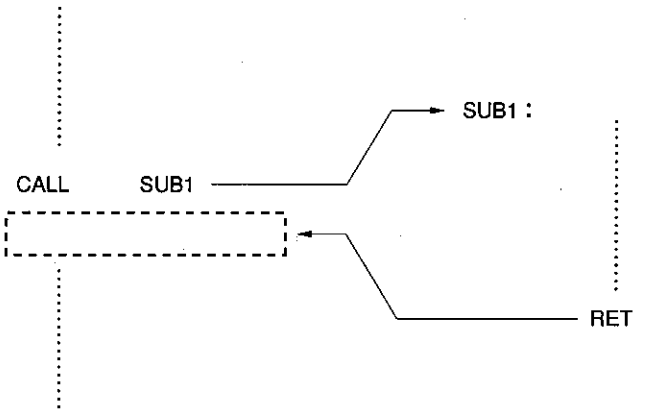
PC ← ASR,

SP ← SP ＋ 1

サブルーチンからメイン・プログラムへ戻るための命令です。

CALL命令でスタックに退避した戻り番地を，プログラム・カウンタに復帰させます。

③ 例



（4）RETSK　　Return to the main program then skip next instruction

① 命令コード

00111	001	1110	0000

② 機　　能

PC ← ASR, SP ← SP ＋ 1 and skip

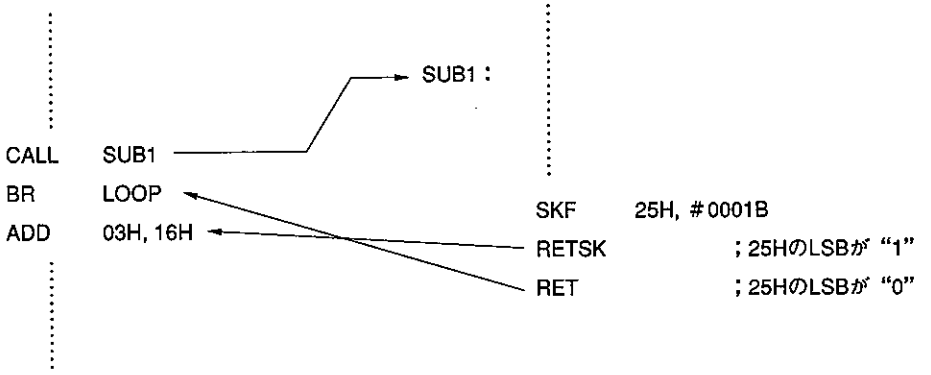
サブルーチンからメイン・プログラムへ戻るための命令です。

CALL命令の次の命令をスキップします（NOP 命令として実行します）。

すなわち，CALL命令でスタックに退避した戻り番地をプログラム・カウンタPCに復帰させたのちプログラム・カウンタをインクリメントします。

③ 例

データ・メモリ（RAM）の25H番地のLSB（最下位ビット）の内容が“0”のときはRET命令を実行し，CALL命令の次の命令に戻り，“1”のときはRETSK命令を実行し，CALL命令の次の次の命令（ここではADD 03H, 16H）に戻ります。



（5）RETI　Return to the main program from interrupt service routine

① 命令コード

00111	100	1110	0000

② 機　能

PC ← ASR, INTR ← INTSK, SP ←SP ＋ 1

割り込み処理プログラムからメイン・プログラムへ戻るための命令です。

ベクタ割り込みでスタックに退避した戻り番地をプログラム・カウンタに復帰させます。

また，システム・レジスタの一部（PSWORD）もベクタ割り込み発生以前の状態に戻ります。

20.5.10　割り込み命令

（1）EI　　Enable Interrupt

① 命令コード

00111	000	1111	0000

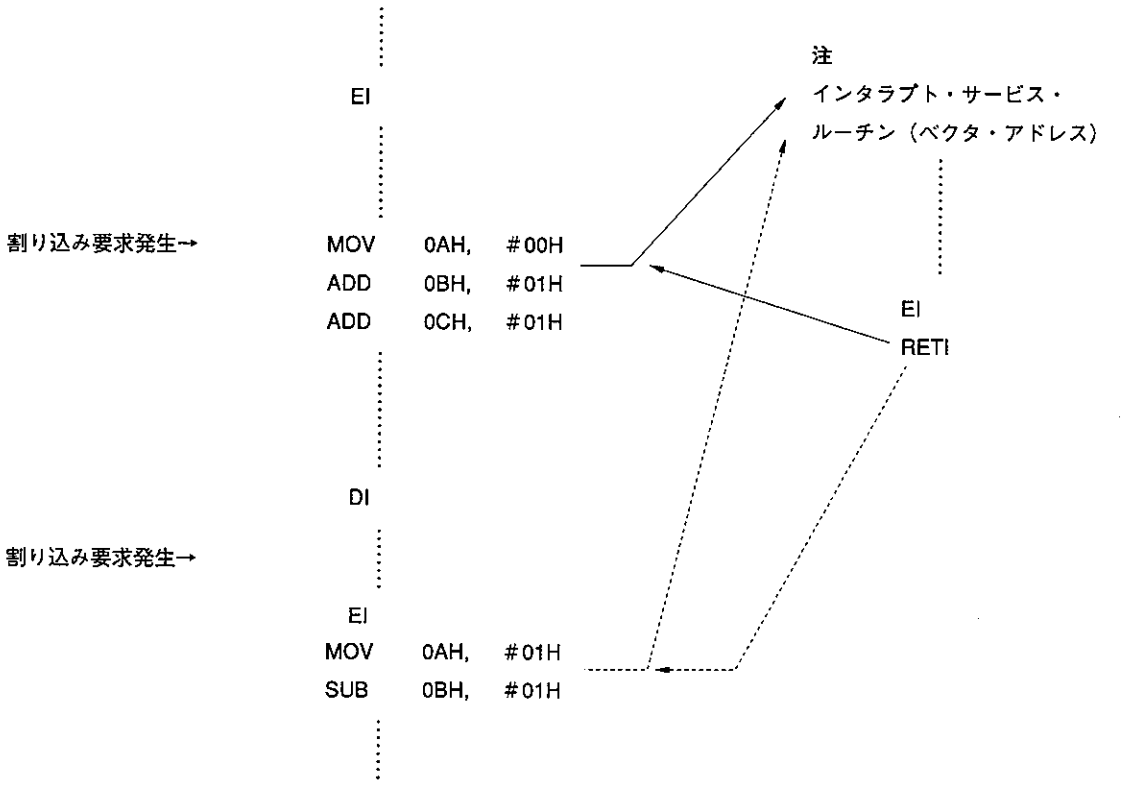
② 機　　能

INTEF ← 1

ベクタ割り込みを許可する命令です。

③ 例

割り込み処理の例を以下に示します。



注　ここで受け付けられる割り込み（EI命令実行後割り込み要求が発生しインタラプト・サービス・ルーチンに流れが移る）とはその割り込みに対する割り込み許可フラグ（IP××）がセットされているものとします。各割り込み許可フラグがセットされていない状態で，EI命令実行後，割り込み要求が発生してもプログラムの流れは変化しません（割り込みは受け付けられません）。ただし，割り込み要求フラグ（IRQ××）はセットされますので，割り込み許可フラグがセットされた時点で受け付けられます。

（2）DI　　Disable Interrupt

①　命令コード

00111	001	1111	0000

②　機　　能

INTEF ← 0

ベクタ割り込みを禁止する命令です。

③　例

（1）EIの例を参照。

20.5.11　その他の命令

（1）STOP s　　Stop CPU and release by condition s

① 命令コード

			3	0
00111	010	1111	s	

② 機　　能

システム・クロックを停止させ，デバイスをSTOPモードにします。

デバイスをSTOPモードにすることにより消費電流を最小限に抑えることができます。

STOPモードを解除する条件をオペランド（s）で指定します。

ストップ解除条件（s）については15.3　STOPモードを参照してください。

（2）HALT h　　Halt CPU and release by condition h

① 命令コード

			3	0
00111	011	1111	h	

② 機　　能

デバイスをHALTモードにします。

デバイスをHALTモードにすることにより消費電流を低減させることができます。

HALTモードを解除する条件をオペランド（h）で指定します。

ホールト解除条件（h）については15.2　HALTモードを参照してください。

（3）NOP　　No operation

① 命令コード

00111	100	1111	0000

② 機　　能

何もせず1マシン・サイクル費やす命令です。

〔メ　モ〕

第21章　アセンブラ予約語

21.1　マスク・オプション疑似命令

μPD17145, 17147, 17149には，次のマスク・オプションがあります。

- $\overline{\text{RESET}}$端子の内蔵プルアップ抵抗
- $P0F_1$端子, $P0F_0$端子の内蔵プルアップ抵抗
- INT端子の内蔵プルアップ抵抗
- 内蔵POC回路

プログラムを作成する際に，マスク・オプション定義疑似命令を使って，ソース・プログラム中で上記すべてのマスク・オプションを指定する必要があります。

21.1.1　マスク・オプションの指定方法

マスク・オプションは次の疑似命令を使ってアセンブラ・ソース・プログラム中に記述します。

- OPTION疑似命令，ENDOP疑似命令
- マスク・オプション定義疑似命令

（1）OPTION疑似命令，ENDOP疑似命令

マスク・オプションを記述する範囲（マスク・オプション定義ブロック）を指定する疑似命令です。

OPTION疑似命令とENDOP疑似命令に挟まれる領域内に，マスク・オプション定義疑似命令を記述してマスク・オプションを指定します。

記述形式

シンボル欄	ニモニック欄	オペランド欄	コメント欄
[レーベル：]	OPTION		[；コメント]
	⋮		
	ENDOP		

（2）マスク・オプション定義疑似命令

表21－1　マスク・オプション定義疑似命令一覧表

オプション	定義疑似命令と書式	オペランド	定義内容
$\overline{\text{RESET}}$端子 内蔵プルアップ抵抗	OPTRES　〈オペランド〉	OPEN	なし
		PULLUP	あり
$P0F_1$, $P0F_0$端子 内蔵プルアップ抵抗	OPTP0F　〈オペランド〉，　〈オペランド2〉注	OPEN	なし
		PULLUP	あり
INT端子 内蔵プルアップ抵抗	OPTINT　〈オペランド〉	OPEN	なし
		PULLUP	あり
内蔵POC回路	OPTPOC　〈オペランド〉	NOUSE	使用しない
		USE	使用する

注　〈オペランド1〉は$P0F_1$端子，〈オペランド2〉は$P0F_0$端子のマスク・オプション指定です。

（3）マスク・オプションの記述例

```
；μPD17149のマスク・オプションの記述例

MASK_OPTION：

    OPTION                  ；マスク・オプション定義ブロックの始まり
    OPTRES   PULLUP         ；RESET端子は内蔵プルアップ抵抗あり
    OPTP0F   PULLUP, OPEN   ；P0F1は内蔵プルアップ抵抗あり，
                            ；P0F0はオープン（外部でプルアップ）
    OPTINT   PULLUP         ；INT端子は内蔵プルアップ抵抗あり
    OPTPOC   NOUSE          ；内蔵POC回路は使用しない
    ENDOP                   ；マスク・オプション定義ブロックの終わり
```

21.2　予約シンボル

μPD17149のデバイス・ファイル（AS17149）内で定義されている予約シンボルの一覧表を次に示します。

システム・レジスタ（SYSREG）

シンボル名	属性	値	Read/Write	説　　明
AR3	MEM	0.74H	R	アドレス・レジスタのビット15 - 12
AR2	MEM	0.75H	R/W	アドレス・レジスタのビット11 - 8
AR1	MEM	0.76H	R/W	アドレス・レジスタのビット7 - 4
AR0	MEM	0.77H	R/W	アドレス・レジスタのビット3 - 0
WR	MEM	0.78H	R/W	ウインドウ・レジスタ
BANK	MEM	0.79H	R/W	バンク・レジスタ
IXH	MEM	0.7AH	R/W	インデクス・レジスタ・ハイ
MPH	MEM	0.7AH	R/W	データ・メモリ・ロウ・アドレス・ポインタ・ハイ
MPE	FLG	0.7AH.3	R/W	メモリ・ポインタ・イネーブル・フラグ
IXM	MEM	0.7BH	R/W	インデクス・レジスタ・ミドル
MPL	MEM	0.7BH	R/W	データ・メモリ・ロウ・アドレス・ポインタ・ロウ
IXL	MEM	0.7CH	R/W	インデクス・レジスタ・ロウ
RPH	MEM	0.7DH	R/W	ジェネラル・レジスタ・ポインタ・ハイ
RPL	MEM	0.7EH	R/W	ジェネラル・レジスタ・ポインタ・ロウ
PSW	MEM	0.7FH	R/W	プログラム・ステータス・ワード
BCD	FLG	0.7EH.0	R/W	BCDフラグ
CMP	FLG	0.7FH.3	R/W	コンペア・フラグ
CY	FLG	0.7FH.2	R/W	キャリー・フラグ
Z	FLG	0.7FH.1	R/W	ゼロ・フラグ
IXE	FLG	0.7FH.0	R/W	インデクス・イネーブル・フラグ

図21－1　システム・レジスタの構成

アドレス	74H	75H	76H	77H	78H	79H	7AH	7BH	7CH	7DH	7EH	7FH
名　称	アドレス・レジスタ（AR）				ウインドウ・レジスタ（WR）	バンク・レジスタ（BANK）	インデクス・レジスタ（IX） データ・メモリ・ロウ・アドレス・ポインタ（MP）			ジェネラル・レジスタ・ポインタ（RP）		プログラム・ステータス・ワード（PSWORD）
記　号	AR3	AR2	AR1	AR0	WR	BANK	IXH MPH	IXM MPL	IXL	RPH	RPL	PSW

ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ注1	0	0	0	0	注2																MPE	0	0	0	(IX)								0	0	0	0	BCD	CMP	CY	Z	IXE
	←																←																											
					(AR)												(BANK)								(MP)								(RP)											
リセット時の初期値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	不定				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

注1．この欄の0が書かれている部分は“0固定”を表します。

2．μPD17145の場合，AR2のb_3，b_2は0に固定されています。μPD17147の場合，AR2のb_3は0に固定されています。

データ・バッファ（DBF）

シンボル名	属性	値	Read/Write	説　　明
DBF3	MEM	0.0CH	R/W	DBFのビット15 - 12
DBF2	MEM	0.0DH	R/W	DBFのビット11 - 8
DBF1	MEM	0.0EH	R/W	DBFのビット7 - 4
DBF0	MEM	0.0FH	R/W	DBFのビット3 - 0

ポート・レジスタ

シンボル名	属性	値	Read/Write	説　　明
P0A3	FLG	0.70H.3	R/W	ポート0Aのビット3
P0A2	FLG	0.70H.2	R/W	ポート0Aのビット2
P0A1	FLG	0.70H.1	R/W	ポート0Aのビット1
P0A0	FLG	0.70H.0	R/W	ポート0Aのビット0
P0B3	FLG	0.71H.3	R/W	ポート0Bのビット3
P0B2	FLG	0.71H.2	R/W	ポート0Bのビット2
P0B1	FLG	0.71H.1	R/W	ポート0Bのビット1
P0B0	FLG	0.71H.0	R/W	ポート0Bのビット0
P0C3	FLG	0.72H.3	R/W	ポート0Cのビット3
P0C2	FLG	0.72H.2	R/W	ポート0Cのビット2
P0C1	FLG	0.72H.1	R/W	ポート0Cのビット1
P0C0	FLG	0.72H.0	R/W	ポート0Cのビット0
P0D3	FLG	0.73H.3	R/W	ポート0Dのビット3
P0D2	FLG	0.73H.2	R/W	ポート0Dのビット2
P0D1	FLG	0.73H.1	R/W	ポート0Dのビット1
P0D0	FLG	0.73H.0	R/W	ポート0Dのビット0
P0E3	FLG	0.6EH.3	R/W	ポート0Eのビット3
P0E2	FLG	0.6EH.2	R/W	ポート0Eのビット2
P0E1	FLG	0.6EH.1	R/W	ポート0Eのビット1
P0E0	FLG	0.6EH.0	R/W	ポート0Eのビット0
P0F1	FLG	0.6FH.1	R	ポート0Fのビット1
P0F0	FLG	0.6FH.0	R	ポート0Fのビット0

レジスタ・ファイル（コントロール・レジスタ）　（1／2）

シンボル名	属性	値	Read/Write	説　明
SP	MEM	0.81H	R/W	スタック・ポインタ
SIOTS	FLG	0.82H.3	R/W	シリアル・インタフェース・スタート・フラグ
SIOHIZ	FLG	0.82H.2	R/W	$P0D_1$/SO端子機能選択フラグ
SIOCK1	FLG	0.82H.1	R/W	シリアル・クロック選択フラグのビット1
SIOCK0	FLG	0.82H.0	R/W	シリアル・クロック選択フラグのビット0
WDTRES	FLG	0.83H.3	R/W	ウォッチドッグ・タイマ・リセット・フラグ
WDTEN	FLG	0.83H.0	R/W	ウォッチドッグ・タイマ・イネーブル・フラグ
TM1OSEL	FLG	0.8BH.3	R/W	$P0D_3$/$\overline{TM1OUT}$端子機能選択フラグ
SIOEN	FLG	0.8BH.0	R/W	シリアル・インタフェース・イネーブル・フラグ
P0EGPU	FLG	0.8CH.2	R/W	P0Eグループ・プルアップ選択フラグ（プルアップ＝1）
P0BGPU	FLG	0.8CH.1	R/W	P0Bグループ・プルアップ選択フラグ（プルアップ＝1）
P0AGPU	FLG	0.8CH.0	R/W	P0Aグループ・プルアップ選択フラグ（プルアップ＝1）
P0DBPU3	FLG	0.8DH.3	R/W	$P0D_3$プルアップ選択フラグ（プルアップ＝1）
P0DBPU2	FLG	0.8DH.2	R/W	$P0D_2$プルアップ選択フラグ（プルアップ＝1）
P0DBPU1	FLG	0.8DH.1	R/W	$P0D_1$プルアップ選択フラグ（プルアップ＝1）
P0DBPU0	FLG	0.8DH.0	R/W	$P0D_0$プルアップ選択フラグ（プルアップ＝1）
INT	FLG	0.8FH.0	R	INT端子ステータス・フラグ
TM0EN	FLG	0.91H.3	R/W	タイマ0イネーブル・フラグ
TM0RES	FLG	0.91H.2	R/W	タイマ0リセット・フラグ
TM0CK1	FLG	0.91H.1	R/W	タイマ0カウント・パルス選択フラグのビット1
TM0CK0	FLG	0.91H.0	R/W	タイマ0カウント・パルス選択フラグのビット0
TM1EN	FLG	0.92H.3	R/W	タイマ1イネーブル・フラグ
TM1RES	FLG	0.92H.2	R/W	タイマ1リセット・フラグ
TM1CK1	FLG	0.92H.1	R/W	タイマ1カウント・パルス選択フラグのビット1
TM1CK0	FLG	0.92H.0	R/W	タイマ1カウント・パルス選択フラグのビット0
BTMISEL	FLG	0.93H.3	R/W	BTMのインターバル時間選択フラグ
BTMRES	FLG	0.93H.2	R/W	ベーシック・インターバル・タイマ・リセット・フラグ
BTMCK1	FLG	0.93H.1	R/W	ベーシック・インターバル・タイマ・カウント・パルス選択フラグのビット1
BTMCK0	FLG	0.93H.0	R/W	ベーシック・インターバル・タイマ・カウント・パルス選択フラグのビット0
P0C3IDI	FLG	0.9BH.3	R/W	ADC_3/$P0C_3$端子機能選択フラグ
P0C2IDI	FLG	0.9BH.2	R/W	ADC_2/$P0C_2$端子機能選択フラグ
P0C1IDI	FLG	0.9BH.1	R/W	ADC_1/$P0C_1$端子機能選択フラグ
P0C0IDI	FLG	0.9BH.0	R/W	ADC_0/$P0C_0$端子機能選択フラグ

レジスタ・ファイル（コントロール・レジスタ）　（2／2）

シンボル名	属性	値	Read/Write	説　明
P0CBIO3	FLG	0.9CH.3	R/W	P0C$_3$入力／出力選択フラグ（1 = 出力ポート）
P0CBIO2	FLG	0.9CH.2	R/W	P0C$_2$入力／出力選択フラグ（1 = 出力ポート）
P0CBIO1	FLG	0.9CH.1	R/W	P0C$_1$入力／出力選択フラグ（1 = 出力ポート）
P0CBIO0	FLG	0.9CH.0	R/W	P0C$_0$入力／出力選択フラグ（1 = 出力ポート）
IEGMD1	FLG	0.9FH.1	R/W	INT端子エッジ検出選択フラグのビット1
IEGMD0	FLG	0.9FH.0	R/W	INT端子エッジ検出選択フラグのビット0
ADCSTRT	FLG	0.0A0H.0	R/W	A/Dコンバータ・スタート・フラグ（読み出し時：常に“0”）
ADCSOFT	FLG	0.0A1H.3	R/W	A/Dコンバータ動作モード選択フラグ（1 = 単発モード）
ADCCMP	FLG	0.0A1H.1	R	A/Dコンバータ・コンパレータ比較結果フラグ（単発モード時のみ有効）
ADCEND	FLG	0.0A1H.0	R	A/Dコンバータ変換終了フラグ
ADCCH3	FLG	0.0A2H.3	R/W	ダミー・フラグ
ADCCH2	FLG	0.0A2H.2	R/W	ダミー・フラグ
ADCCH1	FLG	0.0A2H.1	R/W	A/Dコンバータ・チャネル選択フラグのビット1
ADCCH0	FLG	0.0A2H.0	R/W	A/Dコンバータ・チャネル選択フラグのビット0
P0DBIO3	FLG	0.0ABH.3	R/W	P0D$_3$入力／出力選択フラグ（1 = 出力ポート）
P0DBIO2	FLG	0.0ABH.2	R/W	P0D$_2$入力／出力選択フラグ（1 = 出力ポート）
P0DBIO1	FLG	0.0ABH.1	R/W	P0D$_1$入力／出力選択フラグ（1 = 出力ポート）
P0DBIO0	FLG	0.0ABH.0	R/W	P0D$_0$入力／出力選択フラグ（1 = 出力ポート）
P0EGIO	FLG	0.0ACH.2	R/W	P0Eグループ入力／出力選択フラグ（1 = P0Eすべて出力ポート）
P0BGIO	FLG	0.0ACH.1	R/W	P0Bグループ入力／出力選択フラグ（1 = P0Bすべて出力ポート）
P0AGIO	FLG	0.0ACH.0	R/W	P0Aグループ入力／出力選択フラグ（1 = P0Aすべて出力ポート）
IPSIO	FLG	0.0AEH.0	R/W	シリアル・インタフェース割り込み許可フラグ
IPBTM	FLG	0.0AFH.3	R/W	ベーシック・インターバル・タイマ割り込み許可フラグ
IPTM1	FLG	0.0AFH.2	R/W	タイマ1割り込み許可フラグ
IPTM0	FLG	0.0AFH.1	R/W	タイマ0割り込み許可フラグ
IP	FLG	0.0AFH.0	R/W	INT端子割り込み許可フラグ
IRQSIO	FLG	0.0BBH.0	R/W	シリアル・インタフェース割り込み要求フラグ
IRQBTM	FLG	0.0BCH.0	R/W	ベーシック・インターバル・タイマ割り込み要求フラグ
IRQTM1	FLG	0.0BDH.0	R/W	タイマ1割り込み要求フラグ
IRQTM0	FLG	0.0BEH.0	R/W	タイマ0割り込み要求フラグ
IRQ	FLG	0.0BFH.0	R/W	INT端子割り込み要求フラグ

周辺ハードウエア・レジスタ

シンボル名	属性	値	Read/Write	説　　明
SIOSFR	DAT	01H	R/W	シフト・レジスタの周辺アドレス
TM0M	DAT	02H	W	タイマ０モジュロ・レジスタの周辺アドレス
TM1M	DAT	03H	W	タイマ１モジュロ・レジスタの周辺アドレス
ADCR	DAT	04H	R/W	A/Dコンバータ・データ・レジスタの周辺アドレス
TM0TM1C	DAT	45H	R	タイマ０タイマ１カウント・レジスタの周辺アドレス
AR	DAT	40H	R/W	GET/PUT/PUSH/CALL/BR/MOVT/INC命令用のアドレス・レジスタの周辺アドレス

そ の 他

シンボル名	属性	値	説　　明
DBF	DAT	0FH	GET/PUT/MOVT命令の固定オペランド値
IX	DAT	01H	INC命令の固定オペランド値

〔メ　モ〕

図21－2　コントロール・レジスタの構成（1／2）

カラム・アドレス / ロウ・アドレス	項目	0	1	2	3	4	5	6	7
0 (8)	記号		0 SP	SIOTS SIOHIZ SIOCK1 SIOCK0	WDTRES 0 0 WDTEN				
	リセット時		0 1 0 1	0 0 0 0	0 0 0 0				
	Read/Write		R/W	R/W	R/W				
1 (9)	記号		TM0EN TM0RES TM0CK1 TM0CK0	TM1EN TM1RES TM1CK1 TM1CK0	BTMISEL BTMRES BTMCK1 BTMCK0				
	リセット時		0 0 0 0	1 0 0 0	0 0 0 0				
	Read/Write		R/W	R/W	R/W				
2 (A)	記号	0 0 0 ADCSTRT	ADCSOFT 0 ADCCMP ADCEND	ADCCH3 ADCCH2 ADCCH1 ADCCH0					
	リセット時	0 0 0 0	0 0 0 0	0 0 0 0					
	Read/Write	R/W	R/W R	R/W					
3 (B)	記号								
	リセット時								
	Read/Write								

備考　（　）内は，アセンブラを使用する際の番地です。

なお，コントロール・レジスタのフラグはすべて，アセンブラ予約語としてデバイス・ファイルに登録されていますので，プログラム作成時には予約語を使用すると便利です。

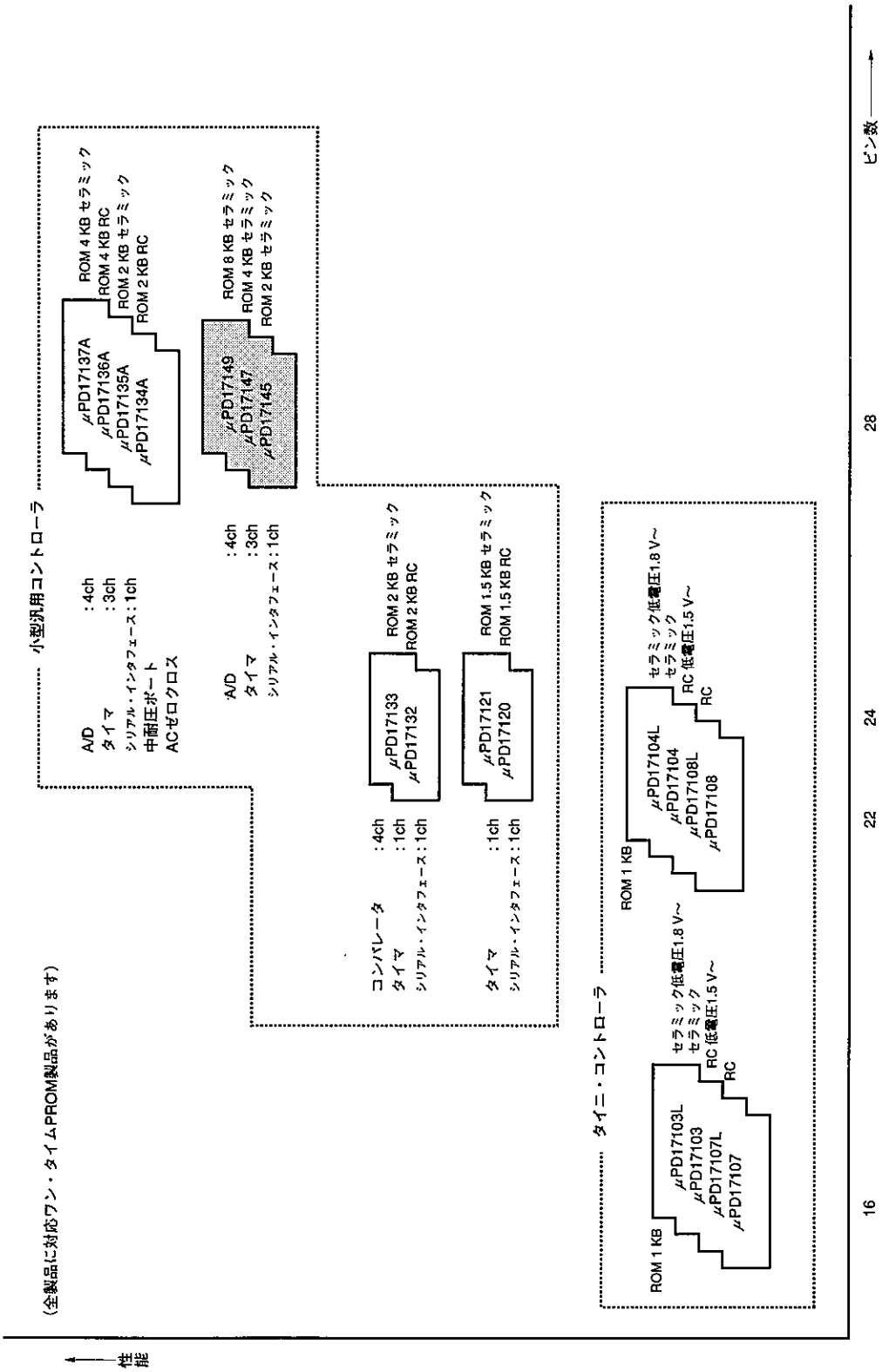
図21－2　コントロール・レジスタの構成（2／2）

8	9	A	B	C	D	E	F
			TM1OSEL	0	P0DBPU3		0
			0	P0EGPU	P0DBPU2		0
			0	P0BGPU	P0DBPU1		0
			SIOEN	P0AGPU	P0DBPU0		INT
			0	0	0		0
			0	0	0		0
			0	0	0		0
			0	0	0		注
			R/W	R/W	R/W		R
			P0C3IDI	P0CBIO3			0
			P0C2IDI	P0CBIO2			0
			P0C1IDI	P0CBIO1			IEGMD1
			P0C0IDI	P0CBIO0			IEGMD0
			0	0			0
			0	0			0
			0	0			0
			0	0			0
			R/W	R/W			R/W
			P0DBIO3	0		0	IPBTM
			P0DBIO2	P0EGIO		0	IPTM1
			P0DBIO1	P0BGIO		0	IPTM0
			P0DBIO0	P0AGIO		IPSIO	IP
			0	0		0	0
			0	0		0	0
			0	0		0	0
			0	0		0	0
			R/W	R/W		R/W	R/W
			0	0	0	0	0
			0	0	0	0	0
			0	0	0	0	0
			IRQSIO	IRQBTM	IRQTM1	IRQTM0	IRQ
			0	0	0	0	0
			0	0	0	0	0
			0	0	0	0	0
			0	1	1	0	0
			R/W	R/W	R/W	R/W	R/W

注　INTフラグは，そのときのINT端子の状態により異なります。

〔メ　モ〕

付録A　μPD171××サブシリーズの展開



〔メ　モ〕

付録B　μPD17145サブシリーズと μPD17135A, 17137Aの機能比較

（1／2）

		μPD17145	μPD17147	μPD17149	μPD17135A	μPD17137A
ROM		2Kバイト	4Kバイト	8Kバイト	2Kバイト	4Kバイト
RAM		110×4ビット			112×4ビット	
スタック		アドレス・スタック×5レベル 割り込みスタック×3レベル				
命令実行時間 （クロック，電源電圧）		2 μs（fx = 8 MHz, V_{DD} = 4.5 ～ 5.5 V） 4 μs（fx = 4 MHz, V_{DD} = 3.6 ～ 5.5 V） 8 μs（fx = 2 MHz, V_{DD} = 2.7 ～ 5.5 V）			2 μs（fx = 8 MHz, V_{DD} = 4.5 ～ 5.5 V） 4 μs（fx = 4 MHz, V_{DD} = 2.7 ～ 5.5 V）	
I/O	CMOS入出力	12（P0A, P0B, P0C）				
	入力専用	2（$P0F_0$, $P0F_1$）			1（$P1B_0$）	
	センス入力	1（INT） マスク・オプション・プルアップ可			1（INT）	
	N-chオープン・ドレーン入出力	8（P0D, P0E　耐圧：V_{DD}） P0D プルアップ：ソフトウエア P0E プルアップ：ソフトウエア			8（P0D, P1A　耐圧：9 V） P0Dプルアップ：マスク・オプション P1A プルアップ：マスク・オプション	
内蔵プルアップ抵抗		100 kΩ TYP.（P0D 以外） 10 kΩ TYP.（P0D）			100 kΩ TYP.	
A/Dコンバータ（電源電圧）		8ビット×4チャネル （V_{DD} = 4.0 ～ 5.5 V）			8ビット×4チャネル （V_{DD} = 4.5 ～ 5.5 V）	
	基準電圧端子	V_{REF}（V_{REF} = 2.5 V ～ V_{DD}）			なし（V_{REF} = V_{ADC} = V_{DD}）	
タイマ	8ビット（TM0, TM1）	2（タイマ出力：$\overline{TM1OUT}$） TM0クロック：fx/512 fx/64 fx/16 INT TM1クロック：fx/8192 fx/128 fx/16 TM0カウント・アップ			2（タイマ出力：$\overline{TM0OUT}$） TM0クロック：fx/256 fx/64 fx/16 INT TM1クロック：fx/1024 fx/512 fx/256 TM0カウント・アップ	
	ベーシック・インターバル（BTM）	1（ウォッチドッグ・タイマ兼用） カウント・パルス：fx／16384 fx／4096 fx／512 fx／16			1（ウォッチドッグ・タイマ兼用） カウント・パルス：fx／8192 fx／4096 TM0 カウント・アップ INT	

★

★

付

（2／2）

		μPD17145	μPD17147	μPD17149	μPD17135A	μPD17137A
割り込み	外部	1			1 （ACゼロクロス検出あり）	
	内部	4（TM0, TM1, BTM, SIO）				
★ SIO		1（クロック同期3線式）				
	出力ラッチ	P0D1のラッチとは独立			P0D1のラッチと兼用	
スタンバイ機能		HALT, STOP （解除用入力端子$\overline{\text{RLS}}$あり）			HALT, STOP	
発振安定待ち時間		128 ×256 カウント			512 ×256 カウント	
POC 機能		マスク・オプション			内蔵	
パッケージ		28ピン・プラスチックSDIP（400 mil） 28ピン・プラスチックSOP（375 mil）				
ワン・タイムPROM		μPD17P149			μPD17P137A	

注意　μPD17145 サブシリーズとμPD17135A, 17137Aとはピン互換製品ではありません。また，μPD17145 サブシリーズには，μPD17134A, 17136A（RC発振タイプ）に相当する製品はありません。
電気的特性については，それぞれの製品のデータ・シートを参照してください。

備考　fx：システム・クロック発振周波数

付録C　開発ツール

μPD17145サブシリーズのプログラムを開発するために，以下の開発ツールを用意しています。

ハードウエア

名　　称	概　　要
インサーキット・エミュレータ （IE-17K IE-17K-ET[注1] EMU-17K[注2]）	IE-17K, IE-17K-ET, EMU-17Kは，17Kシリーズ共通のインサーキット・エミューレータです。IE-17KおよびIE-17K-ETは，ホスト・マシンであるPC-9800 シリーズまたはIBM PC/AT™とRS-232-Cを介して接続して使用します。EMU-17Kは，ホスト・マシンであるPC-9800シリーズの拡張用スロットに実装して使用します。 各品種専用のシステム・エバリュエーション・ボード（SEボード）と組み合わせて使用することにより，その品種に対応したエミュレータとして動作します。マン・マシン・インタフェース・ソフトウエアである*SIMPLEHOST*®を使用すると，さらに高度なディバグ環境を実現できます。 また，EMU-17Kは，データ・メモリの内容をリアルタイムで確認できるという機能を備えています。
SEボード （SE-17145）	SE-17145は，μPD17145サブシリーズ用のSEボードです。単体でシステム評価に，インサーキット・エミュレータと組み合わせてディバグに使用します。
エミュレーション・プローブ （EP-17K28CT）	EP-17K28CTは，17Kシリーズ28ピン・シュリンクDIP（400 mil）用のエミュレーション・プローブです。
エミュレーション・プローブ （EP-17K28GT）	EP-17K28GTは，17Kシリーズ28ピンSOP（375 mil）用のエミュレーション・プローブです。EV-9500GT-28[注3]とともに使用することで，SEボードとターゲット・システムを接続します。
変換アダプタ （EV-9500GT-28[注3]）	EV-9500GT-28は，28ピンSOP（375 mil）用のアダプタです。EP-17K28GTとターゲット・システムを接続するために使用します。
PROMプログラマ[注4] （AF-9703, AF-9704, AF-9705またはAF-9706）	AF-9703, AF-9704, AF-9705およびAF-9706は，μPD17P149に対応したPROMプログラマです。プログラムアダプタAF-9808Mを接続することにより，μPD17P149をプログラミングすることができます。
プログラムアダプタ[注4] （AF-9808M）	AF-9808Mは，μPD17P149をプログラミングするためのアダプタです。AF-9703, AF-9704, AF-9705またはAF-9706と組み合わせて使用します。

注1．廉価版：電源外付けタイプ

2．株式会社アイ・シーの製品です。詳細につきましては，株式会社アイ・シー（東京（03）3447-3793）までお問い合わせください。

3．EP-17K28GTには，EV-9500GT-28が2個添付されています。また，EV-9500GT-28を5個1組で別売もしています。

4．安藤電気株式会社の製品です。詳細につきましては，安藤電気株式会社（東京（03）3733-1151）までお問い合わせください。

付

ソフトウエア

名　称	概　要	ホスト・マシン	OS		供給媒体	オーダ名称
17Kシリーズ アセンブラ （AS17K）	AS17Kは17Kシリーズ共通のアセンブラです。 μPD17145, 17147, 17149のプログラム開発には，このAS17Kとデバイス・ファイル（AS17145, AS17147, AS17149）を組み合わせて使用します。	PC-9800シリーズ	MS-DOS™		5インチ2HD	μS5A10AS17K
					3.5インチ2HD	μS5A13AS17K
		IBM PC/AT™	PC DOS™		5インチ2HC	μS7B10AS17K
					3.5インチ2HC	μS7B13AS17K
デバイス・ファイル （AS17145 AS17147 AS17149）	AS17145, AS17147, AS17149はμPD17145, 17147, 17149およびμPD17P149用のデバイス・ファイルです。 17Kシリーズ共通のアセンブラ（AS17K）と組み合わせて使用します。	PC-9800シリーズ	MS-DOS		5インチ2HD	μS5A10AS17145[注]
					3.5インチ2HD	μS5A13AS17145[注]
		IBM PC/AT	PC DOS		5インチ2HC	μS7B10AS17145[注]
					3.5インチ2HC	μS7B13AS17145[注]
サポート・ソフトウエア （*SIMPLEHOST*）	*SIMPLEHOST*はインサーキット・エミュレータとパーソナル・コンピュータを用いてプログラム開発を行うときにWindows™上でマン・マシン・インタフェースを行うソフトウエアです。	PC-9800シリーズ	MS-DOS	Windows	5インチ2HD	μS5A10IE17K
					3.5インチ2HD	μS5A13IE17K
		IBM PC/AT	PC DOS		5インチ2HC	μS7B10IE17K
					3.5インチ2HC	μS7B13IE17K

注　μS××××AS17145には，AS17145, AS17147, AS17149が入っています。

備考　対応しているOSのバージョンは次のとおりです。

OS	バージョン
MS-DOS	Ver.3.30～Ver.5.00A[注]
PC DOS	Ver.3.1～Ver.5.0[注]
Windows	Ver.3.0～Ver.3.1

注　MS-DOSのVer.5.00/5.00A，PC DOSのVer.5.0にはタスク・スワップ機能がありますが，このソフトウエアではタスク・スワップ機能は使用できません。

付録D　マスクROMの発注方法

プログラム開発が完了しましたら，次の手順でマスクROM品を発注してください。

（1）マスクROM 発注の予約

当社特約店あるいは当社販売部門に，マスクROM発注の予定を連絡してください。あらかじめ連絡をいただかないと納品が遅れる場合があります。

（2）発注媒体の作成

マスクROM発注用の媒体はUV-EPROMです。

まず，アセンブラ（AS17K）のアセンブル・オプションに／PROMを追加し，マスクROM発注用ヘキサ・ファイル（拡張子が　.PRO[注]になります）を作成してください。

次に，マスクROM発注用ヘキサ・ファイルをUV-EPROMに書き込んでください。

なお，UV-EPROMで発注する場合には同じ内容のUV-EPROMを3個作成してください。

注　.ICEでは発注できません。

（3）必要書類の作成

マスクROM 発注にあたって，下記の書類を記入してください。

・マスク式ROM発注書

・マスク式ROM発注チェック・シート

（4）発　　注

（2）で作成した媒体と（3）で記入した書類とをまとめて，発注予約日までに当社特約店または当社販売部門に渡してください。

注意　詳しくはインフォメーション資料「ROMコードの発注方法」（IEM-834）をご覧ください。

〔メ　モ〕

付録E　命令索引

E.1　命令索引（機能別）

付

【その他の命令】

E.2　命令索引（アルファベット順）

【X】

付録F　改版履歴 ★

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版　数	前版からの主な改版内容	適用箇所
第2版	図13－1　8ビット・タイマ・カウンタの構成を修正 13.1.6　インターバル時間の設定を追加 13.1.7　インターバル時間の誤差を追加 図13－21　単発モード（コンペア動作）のタイミングを変更	第13章　周辺ハードウエア
	表15－1　スタンバイ・モード中の状態の注意2の文を修正 15.3.3　STOPの設定条件（1）$\overline{\text{RLS}}$入力による解除の場合を変更	第15章　スタンバイ機能
	17.3　POC回路使用時の注意事項を追加	第17章　POC回路（マスク・オプション）
	20.3　命令セット一覧に注1（MOVT命令の実行サイクル数）を追加	第20章　命令セット
	付録B　μPD17145サブシリーズとμPD17135A，17137Aの機能比較 命令実行時間の修正	付録B　μPD17145サブシリーズとμPD17135A，17137Aの機能比較

付

〔メ　モ〕

キリトリ

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］ μPD17145サブシリーズ　ユーザーズ・マニュアル
（U10261JJ2V0UM00（第２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半導体ソリューション技術本部員，
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半導体インフォメーションセンター
FAX：（044）548-7900

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	(052)222-2170
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	小山支店 小山	(0285)24-5011	富山支店 富山	(0764)31-8461
東北支社 仙台	(022)261-5511	長野支社 長野	(0262)35-1444	三重支店 津	(0592)25-7341
岩手支店 盛岡	(0196)51-4344	松本支店 松本	(0263)35-1666	京都支社 京都	(075)344-7824
山形支店 山形	(0236)23-5511	上諏訪支店 諏訪	(0266)53-5350	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	(0292)26-1717	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)324-5511	沼津支店 沼津	(0559)63-4455	松山支店 松山	(0899)45-4111
群馬支店 高崎	(0273)26-1255	浜松支店 浜松	(053)452-2711	九州支社 福岡	(092)271-7700
太田支店 太田	(0276)46-4011	北陸支社 金沢	(0762)23-1621	北九州支店 北九州	(093)541-2887
宇都宮支店 宇都宮	(0286)21-2281	福井支店 福井	(0776)22-1866		

【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)548-7923	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)
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	