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。

ユーザーズ・マニュアル

RENESAS

μPD17120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

μPD17120
μPD17121
μPD17132
μPD17133
μPD17P132
μPD17P133

資料番号 IEU-835A（第2版）
発行年月 January 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	μPD17120GT, 17121GT, 17132GT, 17133GT, 17P132, 17P133 開発中→開発済み
p.7	**1.4　端子接続図（Top View）** **（2）プログラム・メモリ書き込み/ベリファイ・モードの図を修正**
p.17 p.18	**表2-1　未使用端子の処理を修正** **2.4　$\overline{\text{RESET}}$ 端子と INT 端子の使用上の注意（通常動作モード時のみ）を追加**
p.22	**3.3　プログラム・カウンタ動作時の注意を追加**
p.23 p.27	**第4章　プログラム・メモリ（ROM）** プログラム・メモリの説明文を修正 **4.2.2　テーブル参照** テーブル参照命令の備考文を追加
p.61 p.63	**7.7　プログラム・ステータス・ワード（PSWORD）** プログラム・ステータス・ワードの説明文を修正 **表7-2　ゼロ・フラグ（Z）とコンペア・フラグ（CMP）を修正**
p.89	**表11-1　ALU 処理命令一覧を修正**
p.118 p.122 p.123 p.124 p.132 p.132 p.136	**図13-1　8ビット・タイマ・カウンタの構成を修正** **13.1.5　モジュロ・レジスタへのカウント値の設定と計算方法** **図13-3　モジュロ・レジスタへのカウント値の設定に注意文を追加** **（2）インターバル時間の計算方法を修正** **13.1.6　インターバル時間の誤差を追加** **13.2.2　コンパレータの機能** コンパレート時間の記述を修正 注意文を追加 **図13-12　シリアル・インタフェースのブロック図に注意文を追加**
p.152	**14.3　割り込みシーケンスを修正**
p.161	**第15章　スタンバイ機能を修正**
p.171 p.174	**表16-1　リセット時の各ハードウエアの状態を修正** **16.3.2　パワーオン・リセット機能と動作に注意文を追加**
p.187, p.188	**18.3　命令セット一覧** オペレーションを修正
p.257, p.263	**19.2　予約シンボル** 予約シンボルの説明文を修正
p.259, p.265	**図19-1，図19-2　コントロール・レジスタの構成に注の文を追加**
p.267	**付録A　開発ツールを変更**
p.271	**付録C　システム・クロック発振回路の構成上の注意を追加**
p.275	**付録E　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，μPD17120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のエンジニアを対象としています。

目的　このマニュアルは，μPD17120サブシリーズの持つハードウエア機能をユーザに理解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μPD17120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D　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**
  →**18.3　命令セット一覧**でその命令のニモニックを調べ，**18.5　命令の個別説明**でその機能を調べてください。

- **μPD17120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－下位，下部－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

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	μPD17120	μPD17121	μPD17132	μPD17133	μPD17P132	μPD17P133
データ・シート	IC-8407 [IC-2972]	IC-8399 [IC-2976]	IC-8412 [IC-2973]	IC-8411 [IC-2974]	ID-8419 [IC-2971]	ID-8426 [IC-2983]
ユーザーズ・マニュアル	このマニュアル [IEU-1367]					
IE-17K CLICE Ver.1.6 ユーザーズ・マニュアル	EEU-929 [EEU-1467]					
IE-17K-ET CLICE-ET Ver.1.6 ユーザーズ・マニュアル	EEU-931 [EEU-1466]					
SE ボード ユーザーズ・マニュアル	EEU-847 [EEU-1412]					
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 (Ver.1.11) 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-907 [EEU-1464]					

備考　[　] 内は英文の資料番号です。

μPD17120サブシリーズは，システム・クロックの種類によって，端子名，信号名が下表のように異なります。

端子名，信号名＼システム・クロック	RC 発振 (μPD17120, μPD17132, μPD17P132)	セラミック発振 (μPD17121, μPD17133, μPD17P133)
システム・クロック発振用端子	OSC_1	X_{IN}
	OSC_0	X_{OUT}
システム・クロック周波数	f_{CC}	f_X

このマニュアルでは特に断りがないかぎり「X_{IN}」，「X_{OUT}」，「f_X」で説明しています。μPD17120，17132，17P132を使用する場合には，それぞれを「OSC_1」，「OSC_0」「f_{CC}」に読み替えてください。

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

μPD17120，17121，17132，17133は，17Kアーキテクチャを採用し，8ビット・タイマ（1チャネル），3線式シリアル・インタフェース，パワーオン/パワーダウン・リセット回路を内蔵した4ビット・シングルチップ・マイクロコントローラです。

μPD17P132，17P133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。

- コンパレータ入力内蔵（μPD17132，17133，17P132，17P133のみ）
  - 外部レファレンス電圧（V_{ref}）とのコンパレート機能
  - 15種類の内部レファレンス電圧（1/16～15/16 V_{DD}）を用い，ソフトウエアにより4ビットA/Dコンバータとして使用可能。
- 3線式シリアル・インタフェース内蔵：1チャネル
- 外付けリセット回路を削減できるパワーオン/パワーダウン・リセット回路内蔵
- μPD17P132，17P133はマスクROM製品と同じ電源電圧で動作可能
  - V_{DD}=2.7～5.5 V

μPD17120サブシリーズは，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。

- 扇風機
- ホット・プレート
- オーディオ機器
- マウス
- プリンタ
- PPC

1.1 機能一覧

品名 / 項目	μPD17120	μPD17132	μPD17P132	μPD17121	μPD17133	μPD17P133
ROM 容量	マスク ROM		ワン・タイム PROM	マスク ROM		ワン・タイム PROM
	1.5 K バイト (768×16ビット)	2 K バイト (1024×16ビット)		1.5 K バイト (768×16ビット)	2 K バイト (1024×16ビット)	
RAM 容量	64×4 ビット	111×4 ビット		64×4 ビット	111×4 ビット	
スタック	アドレス・スタック×5，割り込みスタック×1					
入出力ポート数	19本 { ● 入出力 : 18本 ● センス入力 (INT 端子[注]) : 1 本					
コンパレータ (電源電圧)	なし	4 チャネル (V_{DD}=2.7～5.5 V)		なし	4 チャネル (V_{DD}=2.7～5.5 V)	
タイマ	1 チャネル (8 ビット・タイマ)					
シリアル・インタフェース	1 チャネル (3 線式)					
割り込み	● 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：2 本 { ● タイマ (TM) / ● シリアル・インタフェース (SIO) }					
システム・クロック	RC 発振			セラミック発振		
命令実行時間	8 μs (f_{CC}=2 MHz 時)			2 μs (f_X=8 MHz 時)		
スタンバイ機能	HALT，STOP					
パワーオン/パワーダウン・リセット回路	内蔵 (V_{DD}=5 V±10 %の応用回路で使用可)			内蔵 (V_{DD}=5 V±10 %, f_X=400 kHz～4 MHzの応用回路で使用可)		
動作電源電圧	● 2.7～5.5 V ● 4.5～5.5 V (パワーオン/パワーダウン・リセット機能使用時)					
パッケージ	● 24ピン・プラスチック・シュリンク DIP (300 mil) ● 24ピン・プラスチック SOP (375 mil)					
ワン・タイム PROM 製品	μPD17P132		−	μPD17P133		−

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。

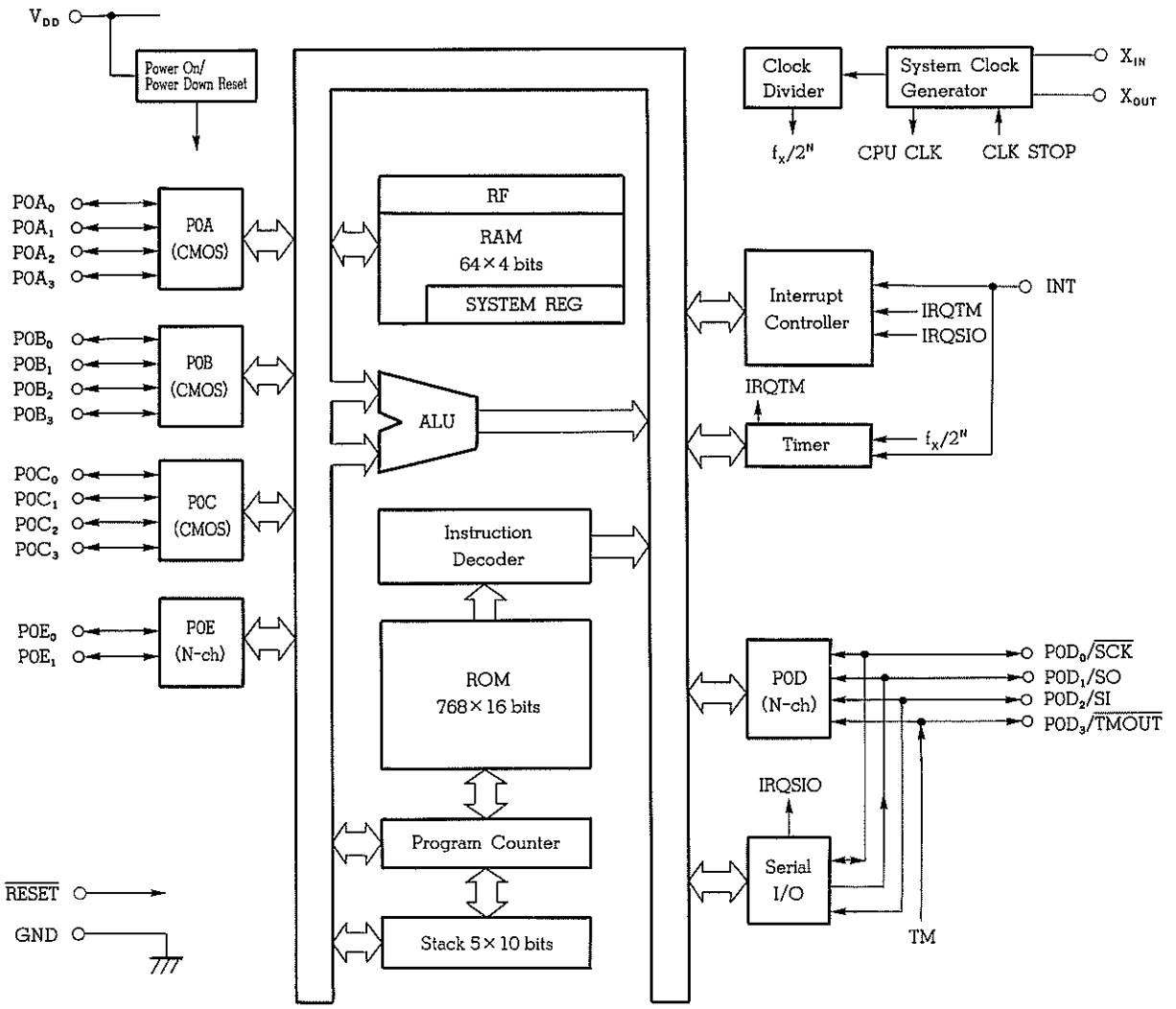
1.2 オーダ情報 ★

オーダ名称	パッケージ	内蔵 ROM
μPD17120CS-×××	24ピン・プラスチック・シュリンク DIP（300 mil）	マスク ROM
μPD17120GT-×××	24ピン・プラスチック SOP（375 mil）	〃
μPD17121CS-×××	24ピン・プラスチック・シュリンク DIP（300 mil）	〃
μPD17121GT-×××	24ピン・プラスチック SOP（375 mil）	〃
μPD17132CS-×××	24ピン・プラスチック・シュリンク DIP（300 mil）	〃
μPD17132GT-×××	24ピン・プラスチック SOP（375 mil）	〃
μPD17133CS-×××	24ピン・プラスチック・シュリンク DIP（300 mil）	〃
μPD17133GT-×××	24ピン・プラスチック SOP（375 mil）	〃
μPD17P132CS	24ピン・プラスチック・シュリンク DIP（300 mil）	ワン・タイム PROM
μPD17P132GT	24ピン・プラスチック SOP（375 mil）	〃
μPD17P133CS	24ピン・プラスチック・シュリンク DIP（300 mil）	〃
μPD17P133GT	24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

● μPD17120, 17121 ブロック図

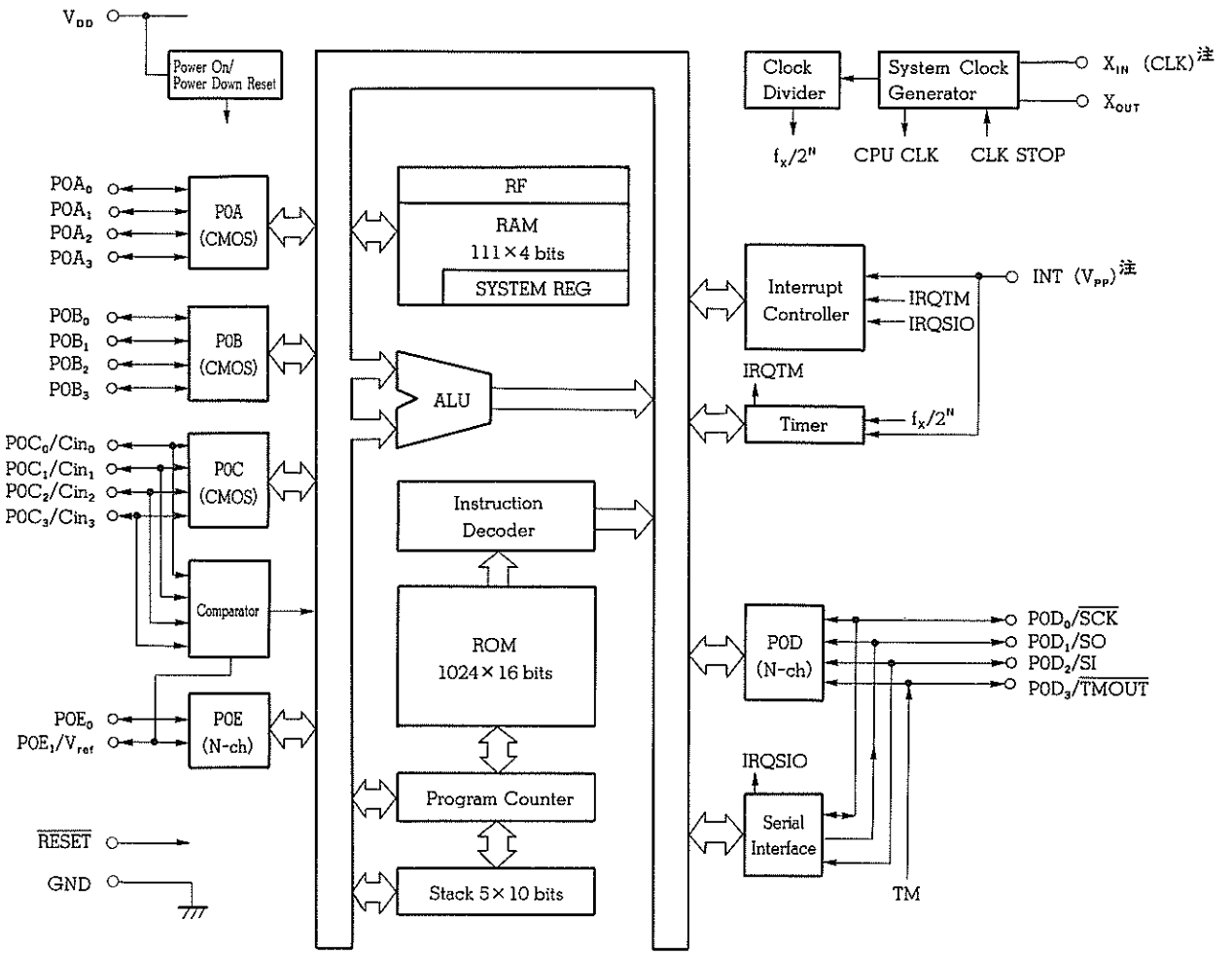


備考　（　）の CMOS, N-ch はポートの出力形式を表します。

CMOS　：CMOS プッシュプル出力

N-ch　：N-ch オープン・ドレーン出力（N-ch オープン・ドレーンの各端子はマスク・オプションによりビット単位でプルアップ抵抗を内蔵することができます）

● μPD17132, 17133, 17P132, 17P133 ブロック図



備考　（　）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（N-ch オープン・ドレーンの各端子は，マスク・オプションによりビット単位でプルアップ抵抗を内蔵することができます）

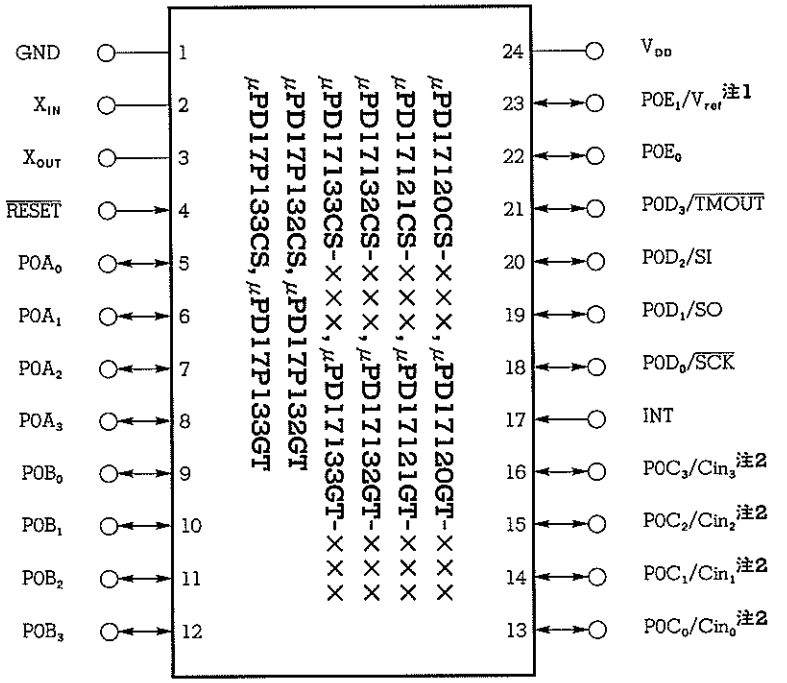
注　（　）内は μPD17P132, 17P133 のプログラム・メモリ書き込み/ベリファイ・モード時のみ有効です。

1.4 端子接続図（Top View）

(1) 通常動作モード

24ピン・プラスチック・シュリンク DIP

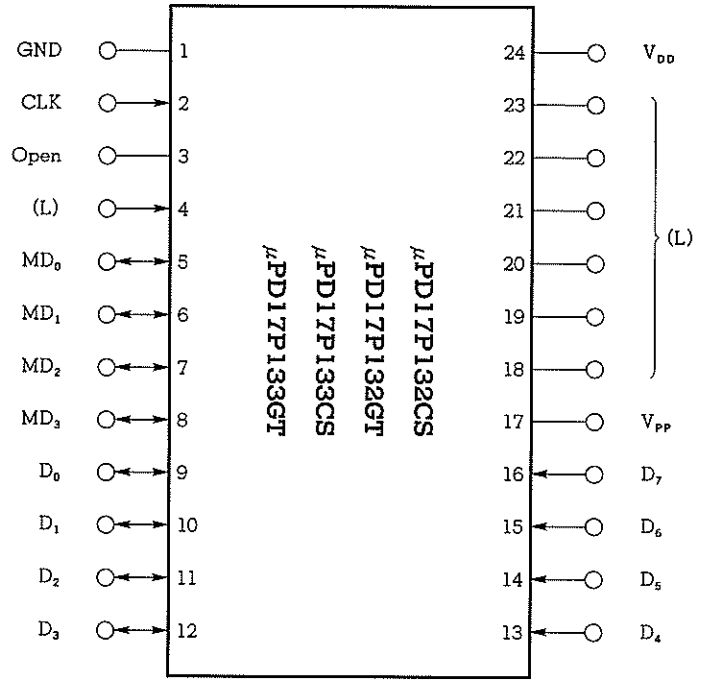
24ピン・プラスチック SOP



注 1. μPD17120, 17121 には V_{ref} 端子がありません。

2. μPD17120, 17121 には Cin_0-Cin_3 端子がありません。

(2) プログラム・メモリ書き込み/ベリファイ・モード ★



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。

(3) 端子名称

Cin_0-Cin_3	：コンパレータ入力	POE_0-POE_1	：ポート0E
CLK	：アドレス更新用クロック入力	$\overline{RESET}$	：リセット入力
D_0-D_7	：データ入出力	$\overline{SCK}$	：シリアル・クロック入出力
GND	：グランド	SI	：シリアル・データ入力
INT	：外部割り込み入力	SO	：シリアル・データ出力
MD_0-MD_3	：動作モード選択	$\overline{TMOUT}$	：タイマ出力
POA_0-POA_3	：ポート0A	V_{DD}	：電源
POB_0-POB_3	：ポート0B	V_{PP}	：プログラム電圧印加
POC_0-POC_3	：ポート0C	V_{ref}	：外部レファレンス電圧
POD_0-POD_3	：ポート0D	X_{IN}, X_{OUT}	：システム・クロック発振用

〔メ モ〕

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	パワーオン・リセット時
1	GND	GND です。	−	−
2 3	X_{IN} X_{OUT}	**μPD17121, 17133, 17P133 の場合** ● X_{IN}, X_{OUT} ・システム・クロック発振子接続用端子 ・セラミック発振子を接続	−	−
2 3	OSC_1 OSC_0	**μPD17120, 17132, 17P132 の場合** ● OSC_0, OSC_1 ・システム・クロック発振用端子 ・OSC_0, OSC_1 間に抵抗を接続		
4	$\overline{RESET}$	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内蔵可能[注]	−	入　力
5 ｜ 8	$P0A_0$ ｜ $P0A_3$	ポート 0A です。 ・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	CMOS プッシュプル	入　力
9 ｜ 12	$P0B_0$ ｜ $P0B_3$	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	CMOS プッシュプル	入　力
13 ｜ 16	$P0C_0/Cin_0$ ｜ $P0C_3/Cin_3$	ポート 0C およびコンパレータのアナログ電圧入力です。 ● $P0C_0$-$P0C_3$ ・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 ● Cin_0-Cin_3 (μPD17132, 17133, 17P132, 17P133 のみ内蔵) ・コンパレータのアナログ入力	CMOS プッシュプル	入　力 (P0C)
17	INT	外部割り込み要求信号の入力およびセンス入力です。	−	入　力

注　μPD17P132, 17P133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	パワーオン・リセット時
18	$P0D_0/\overline{SCK}$	ポート 0D，タイマ出力，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。 ● $P0D_0$-$P0D_3$ • 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 • 1 ビット単位でマスク・オプションによるプルアップ抵抗内蔵可能[注] ● $\overline{SCK}$ • シリアル・クロック入出力	N-ch オープン・ドレーン	入　力 (P0D)
19	$P0D_1/SO$	● SO • シリアル・データ出力		
20	$P0D_2/SI$	● SI • シリアル・データ入力		
21	$P0D_3/\overline{TMOUT}$	● $\overline{TMOUT}$ • タイマ出力		
22 23	$P0E_0$ $P0E_1/V_{ref}$	ポート 0E およびコンパレータの基準電圧入力です。 ● $P0E_0$，$P0E_1$ • 2 ビット入出力ポート • 1 ビット単位で入力/出力設定可能 • 1 ビット単位でマスク・オプションによるプルアップ抵抗内蔵可能[注] ● V_{ref}（μPD17132，17133，17P132，17P133のみ内蔵） • コンパレータの外部レファレンス電圧入力	N-ch オープン・ドレーン	入　力 (P0E)
24	V_{DD}	正電源です。	−	−

注　μPD17P132，17P133 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　プログラム書き込み/ベリファイ・モード時の端子

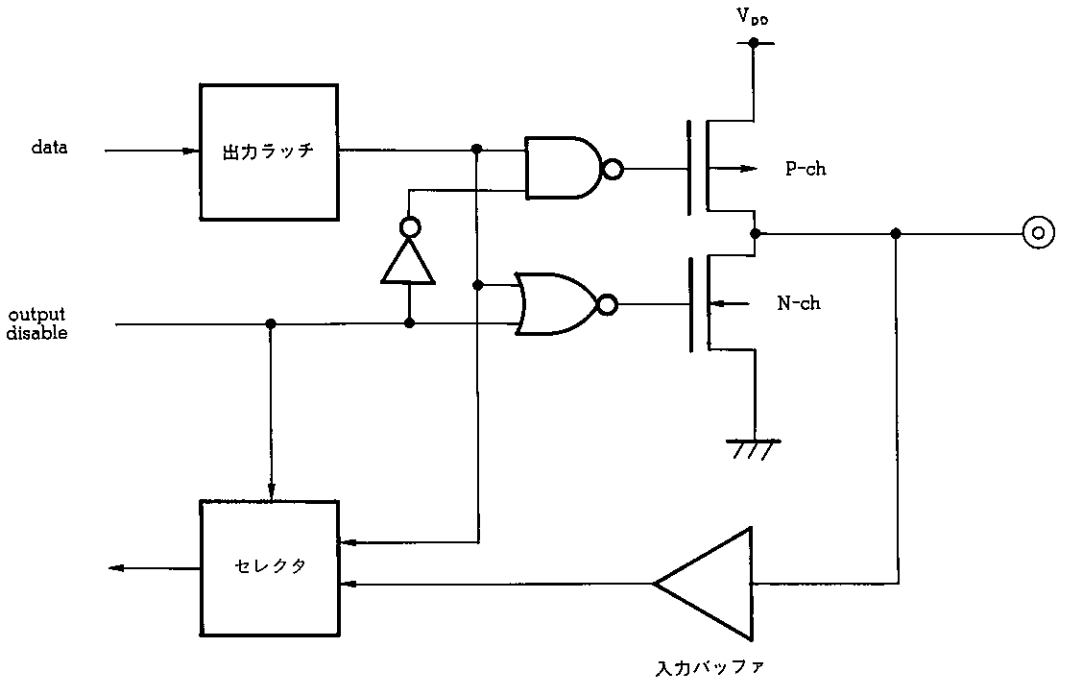
… μPD17P132, 17P133 のみ

端子番号	記　　号	機　　能	入出力
1	GND	GND です。	−
2	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
5 ｜ 8	MD_0 ｜ MD_3	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 ★
9 ｜ 16	D_0 ｜ D_7	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 ★
17	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。	−
24	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。	−

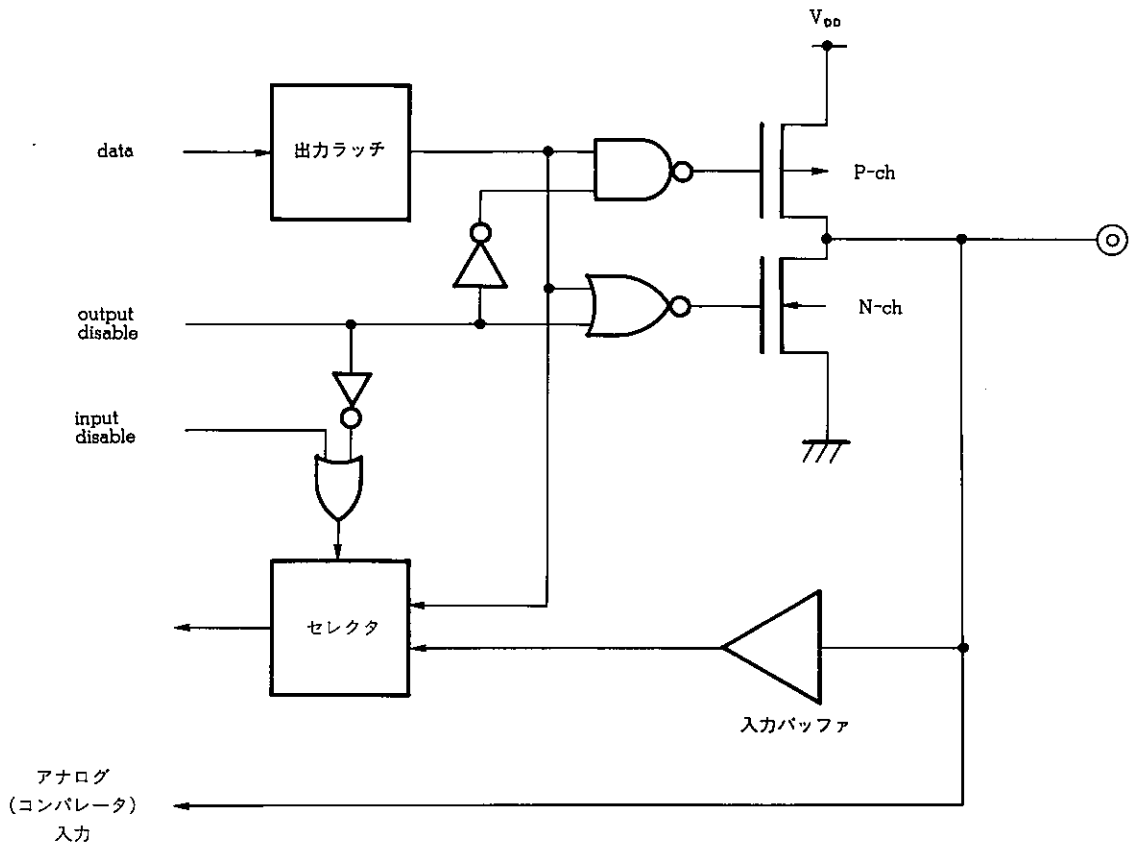
2.2　端子の入出力回路

μPD17120サブシリーズの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) POA_0-POA_3, POB_0-POB_3

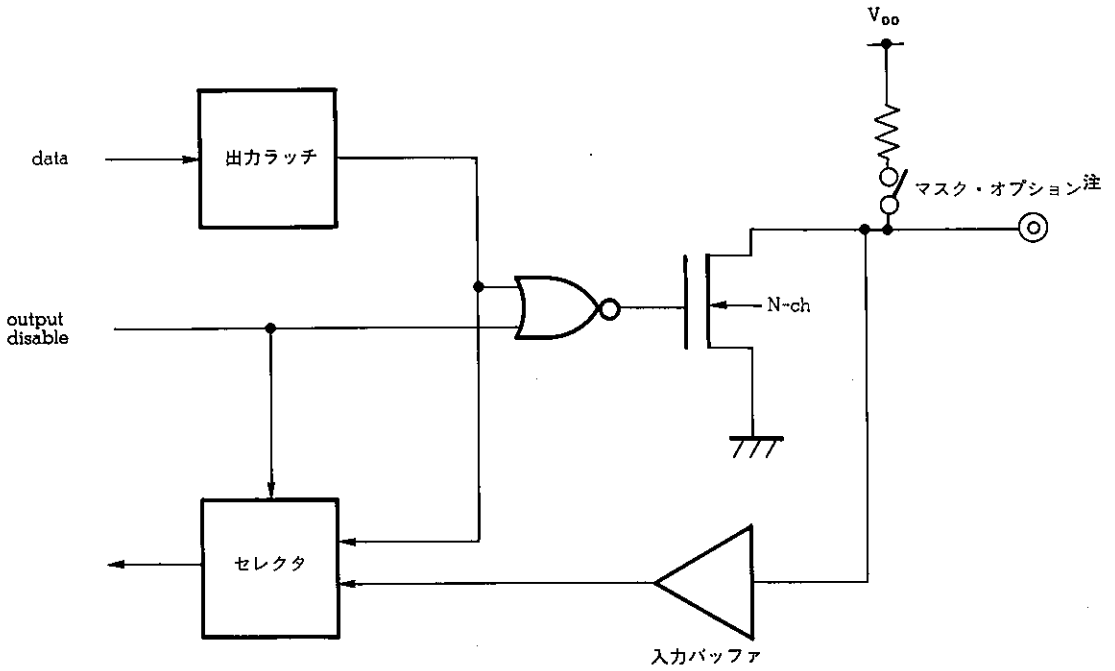


(2) $P0C_0/Cin_0$-$P0C_3/Cin_3$[注]



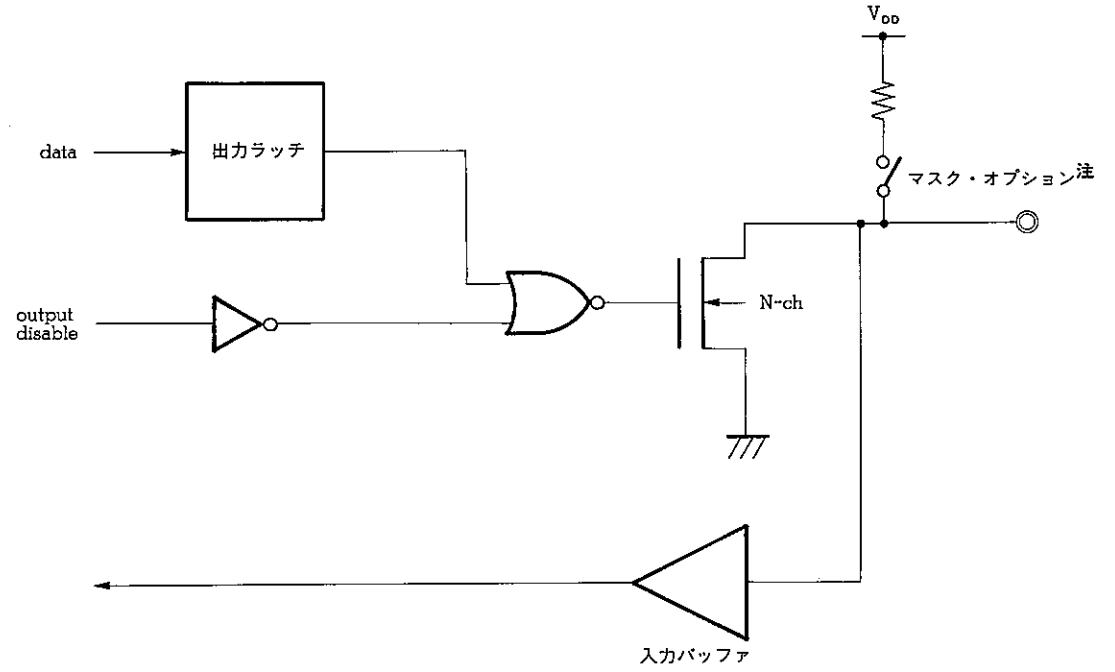
注　μPD17120, 17121 では Cin_0-Cin_3 端子がありません。

(3) POD_0-POD_3



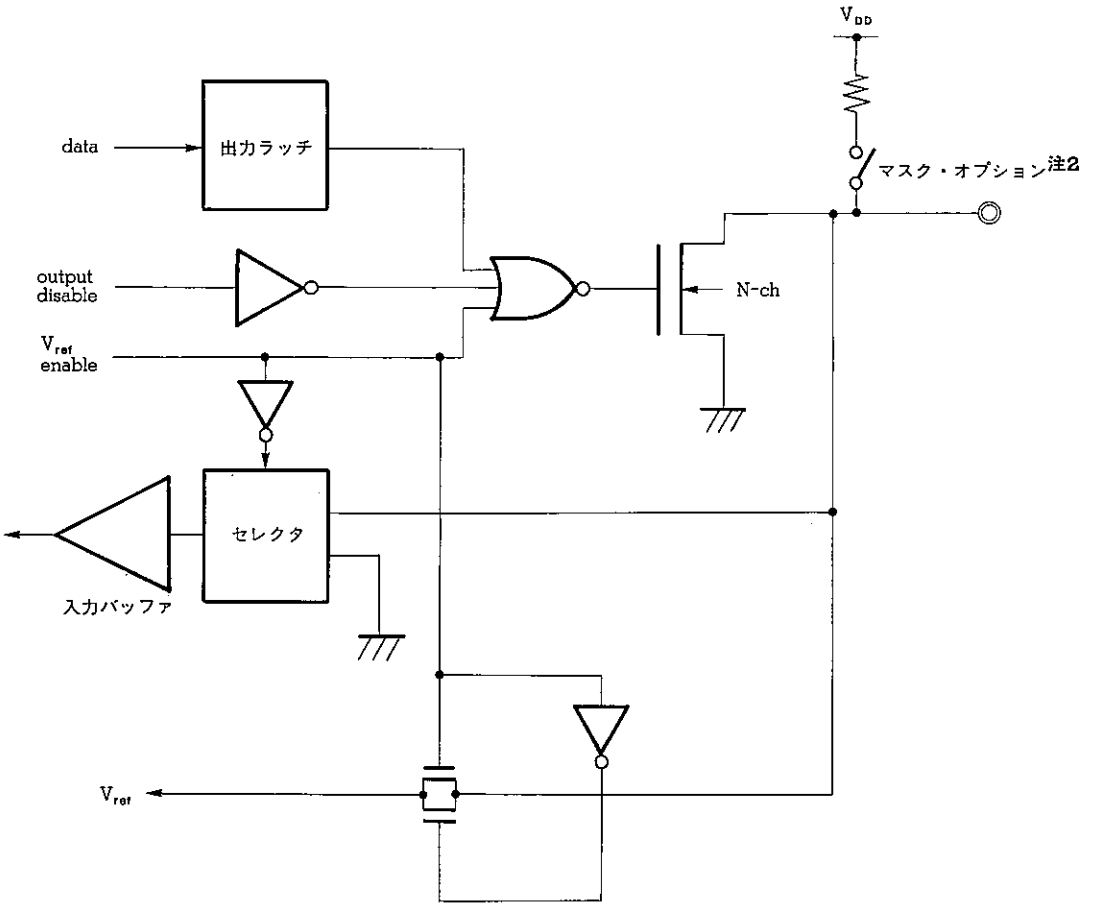
注　μPD17P132, 17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

(4) POE_0



注　μPD17P132, 17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

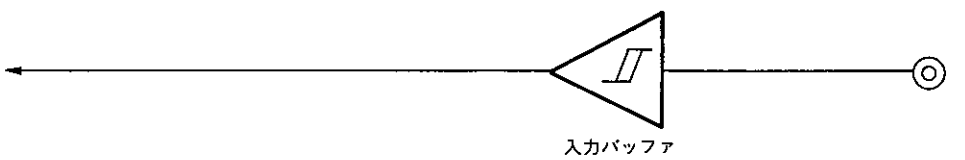
(5) POE_1/V_{ref} 注1



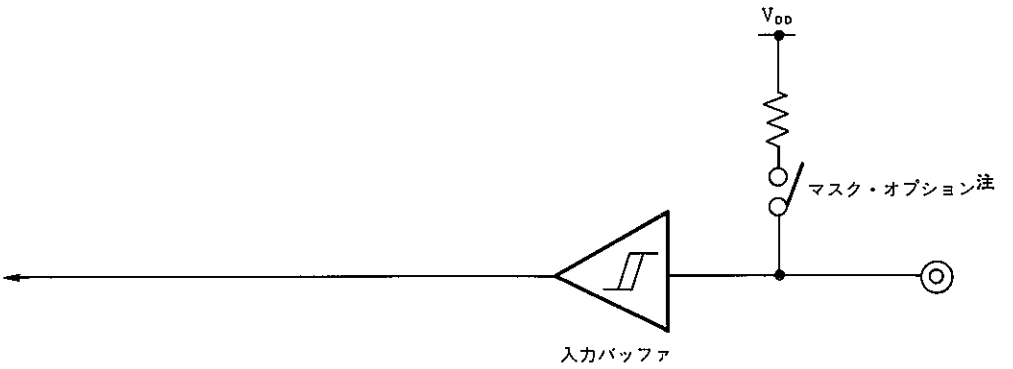
注 1. μPD17120, 17121 では V_{ref} 端子機能はありません。

2. μPD17P132, 17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

(6) INT



(7) $\overline{\text{RESET}}$



注　μPD17P132, 17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

2.3　未使用端子の処理

通常動作モード時，未使用端子には，次に示すような処置を推奨します。

表2-1　未使用端子の処理　★

端子名			推奨処理方法	
			マイコン内部	マイコン外部
ポート	入力モード	P0A，P0B，P0C	−	各端子ごとに抵抗を介して V_{DD} または GND に接続[注1]
		P0D，P0E	マスク・オプションによるプルアップ抵抗を内蔵しない	
			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
	出力モード	P0A，P0B，P0C（CMOS ポート）	−	オープン
		P0D，P0E（N-ch オープン・ドレーン・ポート）	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，ロウ・レベルを出力する	
			マスク・オプション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）[注2]			−	GND に直接接続
$\overline{RESET}$[注3]（内蔵のパワーオン/パワーダウン・リセット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（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）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程度に選ぶことが一般的です。

2．INT 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 GND に接続してください。

3．高い信頼性を必要とする応用回路では，必ず外部から $\overline{RESET}$ 信号を入力するように設計してください。また，$\overline{RESET}$ 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，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　μPD17P132，17P133 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。

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

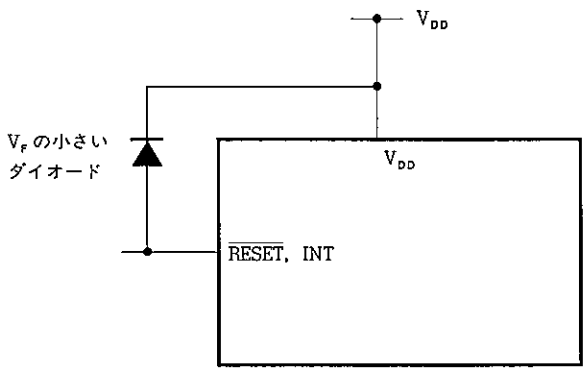
$\overline{\text{RESET}}$ 端子と INT 端子は，**2.1　端子機能説明**に示した機能のほかに，μPD17120サブシリーズの内部動作をテストする，テスト・モードを設定する機能（IC テスト専用）を持っています。

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。

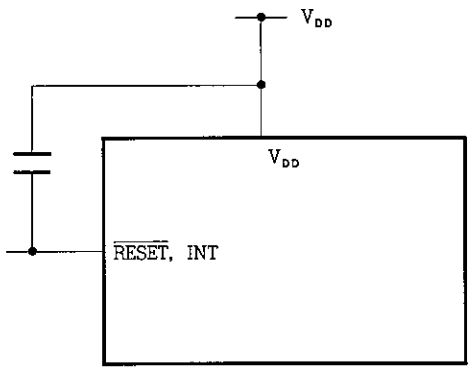
たとえば，$\overline{\text{RESET}}$ 端子または INT 端子の配線の引き回しが長い場合などでは，これらの端子に布線間ノイズが加わって上記の問題を起こしてしまうことがあります。

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・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します。

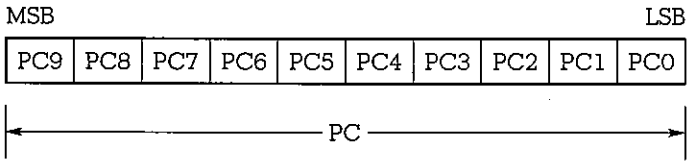
3.1　プログラム・カウンタの構成

図３－１にプログラム・カウンタの構成を示します。

プログラム・カウンタは，10ビットのバイナリ・カウンタで構成されています。

プログラム・カウンタは，命令を実行するたびにインクリメントされます。

図３－１　プログラム・カウンタの構成



3.2　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。

3.2.1-3.2.7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。

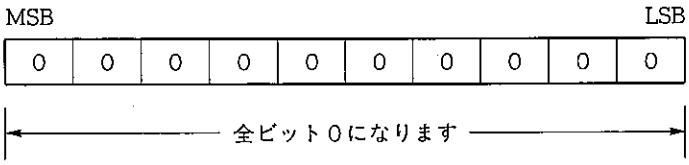
図3-2　命令実行後のプログラム・カウンタの値

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									
	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
BR addr CALL addr	addr で指定した値									
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・ポインタで指定されるアドレス・スタック・レジスタの内容 (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									

3.2.1　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3-3　リセット時のプログラム・カウンタの値

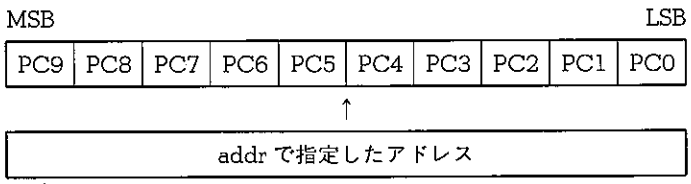


3.2.2　分岐命令（BR）実行時

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。

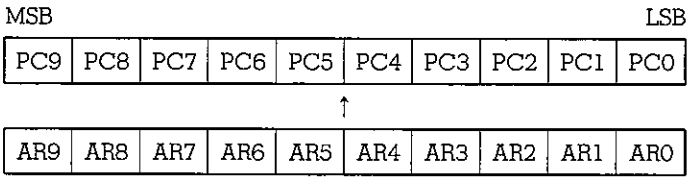
直接分岐命令により指定したアドレスがプログラム・カウンタに設定されます。

図3-4　直接分岐命令実行時のプログラム・カウンタの値



間接分岐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３－５　間接分岐命令実行時のプログラム・カウンタの値

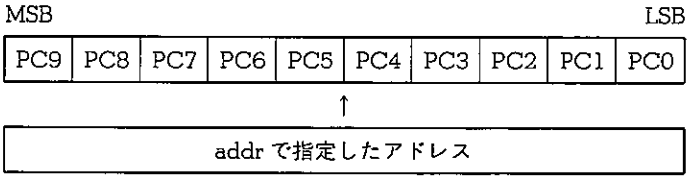


3.2.3　サブルーチン・コール命令（CALL）実行時

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。

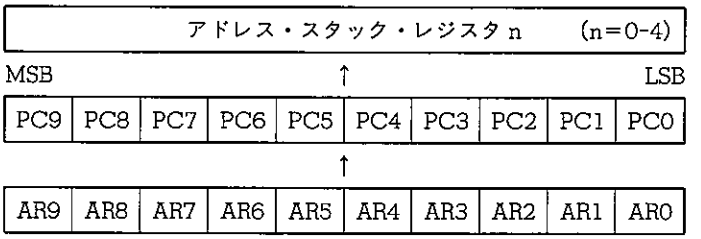
直接サブルーチン・コール命令により，プログラム・カウンタの値がアドレス・スタックに退避されたあと，オペランドに記述したアドレスがプログラム・カウンタに設定されます。直接サブルーチン・コール命令で指定できるアドレスの範囲はプログラム・メモリの全範囲です。

図３－６　直接サブルーチン・コール命令実行時のプログラム・カウンタの値



間接サブルーチン・コール命令により，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。

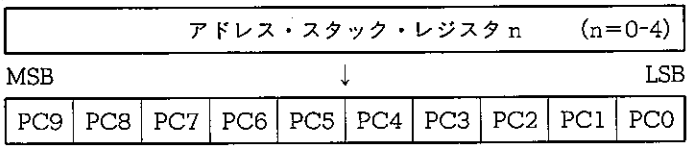
図３－７　間接サブルーチン・コール命令実行時のプログラム・カウンタの値



3.2.4　リターン命令（RET, RETSK, RETI）実行時

リターン命令（RET, RETSK, RETI）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3-8　リターン命令実行時のプログラム・カウンタの値



3.2.5　テーブル参照命令（MOVT）実行時

テーブル参照命令（MOVT DBF, @AR）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。

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

スキップ命令（SKE, SKGE, SKLT, SKNE, SKT, SKF）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ノー・オペレーション命令（NOP）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。

3.2.7　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。

★ 3.3　プログラム・カウンタ動作時の注意

μPD17120, 17121のプログラム・カウンタ（PC）は，10ビットで構成されているため，最大1024ステップ分のプログラムを指定できる構成になっています。これに対しROMサイズは768ステップ(0000H-02FFH番地）しかありません。もしプログラム・カウンタの値が300H以上になると，プログラムの内容としてはFFFFHが読み込まれ，「SKF PSW, #0FH」命令を実行していることと等価になります。

このため，次の点に注意してください。

(1) 768ステップ目（02FFH番地）の命令を実行しても，プログラム・カウンタは自動的には0000H番地になりません。768ステップ目（02FFH番地）までプログラムを使用する場合，その命令が分岐命令（BR）または復帰命令（RET）以外であると，ROMが存在しない領域をプログラム・カウンタが指定してしまいますのでご注意ください。

(2) (1)と同様に，768ステップ目(02FFH番地)以降に分岐するような命令の使用も避けてください。

第4章　プログラム・メモリ（ROM）

μPD17120サブシリーズのプログラム構成を以下に示します。

品名	プログラム・メモリ容量	プログラム・メモリ番地
μPD17120	1.5Kバイト（768×16ビット）	0000H-02FFH
μPD17121		
μPD17132	2Kバイト（1024×16ビット）	0000H-03FFH
μPD17133		
μPD17P132		
μPD17P133		

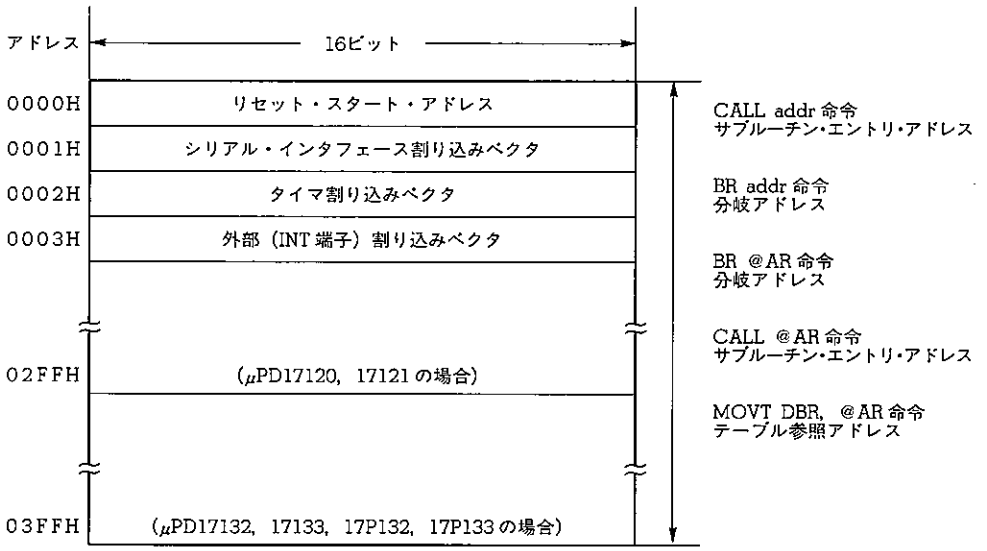
プログラム・メモリには，プログラムおよび定数データ・テーブルなどを格納します。プログラム・メモリの先頭部分は，リセット・スタート・アドレスと割り込みベクタ・アドレスに割り当てられています。 ★

プログラム・メモリはプログラム・カウンタによってアドレス指定されます。

4.1　プログラム・メモリの構成

図4-1にプログラム・メモリ・マップを示します。分岐命令，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4-1　μPD17120サブシリーズのプログラム・メモリ・マップ



4.2　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。定数データは，MOVT命令を使用することによりプログラム・メモリからデータ・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。

4.2.1　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ていると，プログラム・メモリの使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，一箇所にまとめたプログラムを呼び出すことにより，１つのプログラムを何度でも，呼び出すことができます。このプログラムのことを“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まったく関係なく，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)-(5)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4-1に示します。

表4-1　μPD17120サブシリーズのベクタ・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	タイマ割り込み
0003H	外部割り込み（INT端子）

(2) 直接分岐

直接分岐命令（BR addr）では，命令のオペランド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（ただし，最上位ビットは“0”以外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，直接分岐命令により，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。

(3) 間接分岐

間接分岐命令（BR @AR）ではアドレス・レジスタ（AR）の値をアドレスとして分岐します。したがって,間接分岐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。

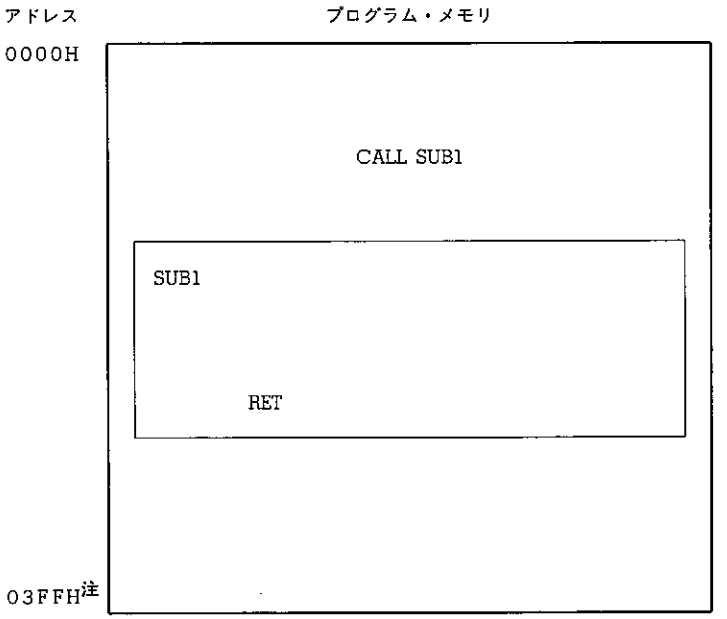
7.2　アドレス・レジスタ（AR）も参照してください。

(4) 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，命令のオペランド11ビットでコール先のプログラム・メモリ・アドレスを指定します（ただし，最上位ビットは“0”以外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)。

例

図4-2　直接サブルーチン・コール（CALL addr）



注　μPD17120, 17121のプログラム・メモリの最終アドレスは02FFH番地です。

(5) 間接サブルーチン・コール

間接サブルーチン・コール（CALL @AR）命令は，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・アドレスを呼び出すことができます。

7.2　アドレス・レジスタ（AR）も参照してください。

4.2.2　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。

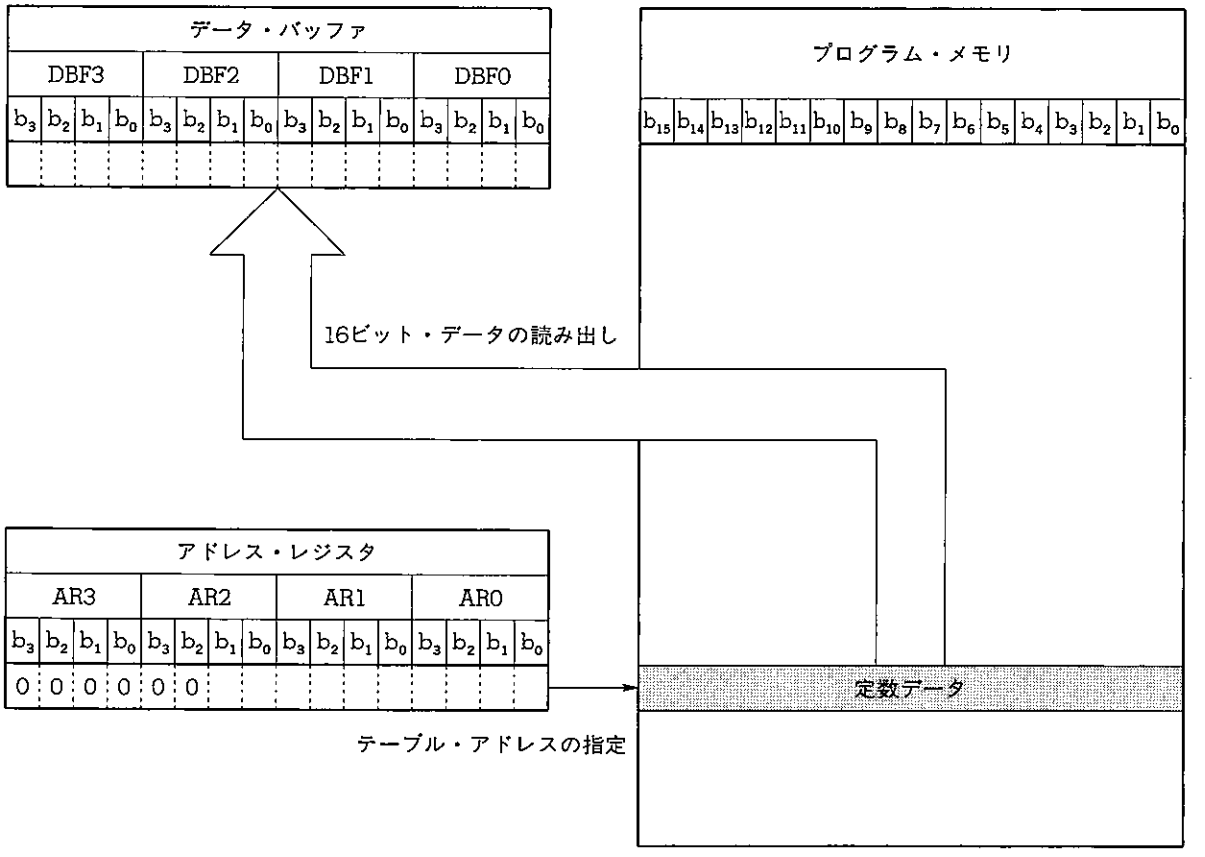
プログラム・メモリの内容は，16ビットで構成されているので，MOVT命令でデータ・バッファに格納される定数データは16ビットになります。アドレス・レジスタを使用することにより，すべてのプログラム・メモリをテーブル参照することができます。

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。

「7.2　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4-3　テーブル参照（MOVT DBF, @AR)



(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OFFSET番目に登録されている値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

          ; BANK0
          MOV    RPH, #0            ; 演算結果の格納先をロウ・アドレス7にする
          MOV    RPL, #7 SHL 1      ; ために, レジスタ・ポインタ7に設定します。
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

          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        ; テーブル参照命令の実行

TABLE :
          DW     0001H
          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

          END
```

(2) 分岐先テーブル

例２に分岐先テーブルのテーブル参照プログラム例を示します。

例 2. 分岐先テーブルの OFFSET 番目に設定した値にアドレスを分岐させるプログラム

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

          ; BANK0
            MOV    RPH, #0          ; レジスタ・ポインタをロウ・アドレス７に設
            MOV    RPL, #7 SHL 1    ; 定します。
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

            ADD      AR0, OFFSET    ; オフセット・アドレスを加算します。
            ADDC     AR1, #0
            MOVT     DBF, @AR       ; テーブル参照の実行
            PUT      AR, DBF
            BR       @AR

TABLE :
            DW     0001H
            DW     0002H
            DW     0004H
            DW     0008H
            DW     0010H
            DW     0020H
            DW     0040H
            DW     0080H
            DW     0100H
            DW     0200H

            END
```

〔メ モ〕

第5章　データ・メモリ（RAM）

データ・メモリとは，演算，制御など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。

5.1　データ・メモリの構成

図5－1にデータ・メモリの構成を示します。

データ・メモリは“バンク”と呼ぶ概念で管理されています。μPD17120サブシリーズはバンク0だけを持っています。

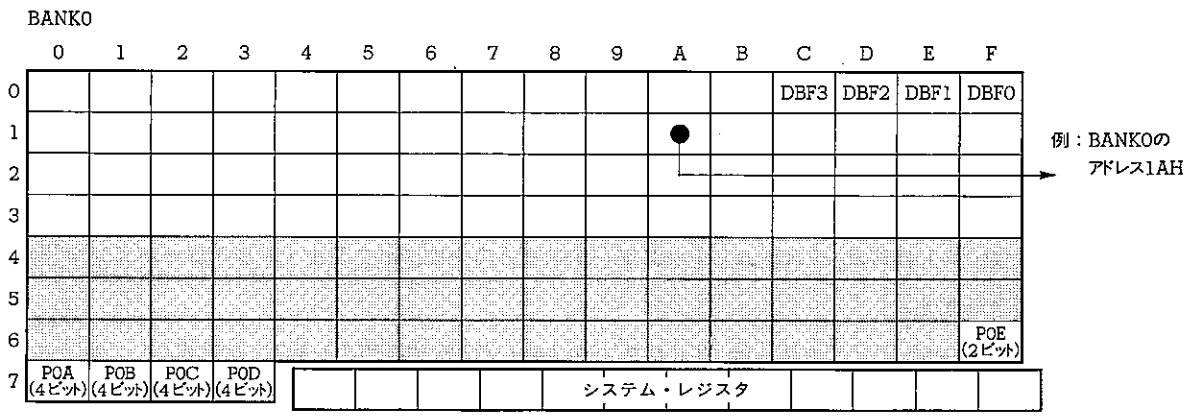
データ・メモリは各バンクごとにアドレス（番地）が割り付けられています。1つのアドレスには4ビットのメモリが対応しており，これを“1ニブル”と呼びます。

データ・メモリのアドレスは7ビットで表され，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，データ・メモリのアドレスが1AH（0011010B）番地の場合，ロウ・アドレスは1H（001B），カラム・アドレスはAH（1010B）ということになります。

なお，μPD17120，17121の場合，アドレス40Hから6EHは非実装領域となっていますので，使用を避けてください。

データ・メモリは機能別に次の**5.1.1-5.1.6**に示すブロックに分けられます。

図5－1　データ・メモリの構成



備考　□は μPD17120，17121の場合の非実装領域を示します。

5.1.1　システム・レジスタ（SYSREG）

データ・メモリのアドレス 74H－7FH に割り当てられた12ニブルで構成されています。システム・レジスタ（SYSREG）はバンクに無関係に割り当てられており，すなわちどのバンクであってもアドレス 74H－7FH には同一のシステム・レジスタ（SYSREG）が存在します。

図５－２に構成を示します。

図５－２　システム・レジスタの構成

システム・レジスタ（SYSREG）												
アドレス	74H	75H	76H	77H	78H	79H	7AH	7BH	7CH	7DH	7EH	7FH
名　称 （記号）	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）

5.1.2　データ・バッファ（DBF）

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。

図５－３に構成を示します。

図５－３　データ・バッファの構成

データ・バッファ（DBF）				
アドレス	0CH	0DH	0EH	0FH
記　号	DBF3	DBF2	DBF1	DBF0

5.1.3　ジェネラル・レジスタ（GR）

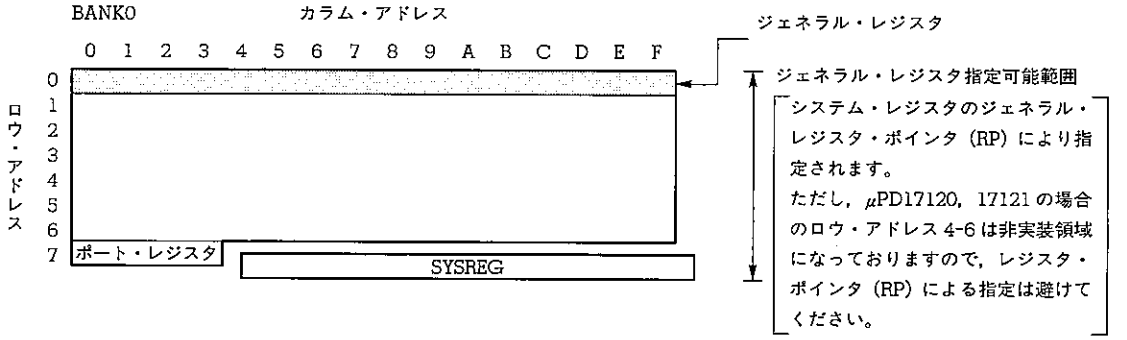
データ・メモリの任意のロウ・アドレスで指定される16ニブルで構成されています。

任意のロウ・アドレスは，システム・レジスタ(SYSREG)の中のジェネラル・レジスタ・ポインタ(RP)により指定されます。

ただし μPD17120, 17121 の場合は，アドレス 40H-6EH が非実装領域のため，この領域をジェネラル・レジスタとして指定しないでください。

図５－４に構成を示します。

図5-4　ジェネラル・レジスタ（GR）の構成



5.1.4　ポート・レジスタ

データ・メモリのBANK0のアドレス6FH－73Hに割り当てられた5ニブルで構成されています。

ただし，図5－5に示すようにアドレス6FHの上位2ビットは0に固定されています。

図5－5に構成を示します。

図5－5　ポート・レジスタの構成

ポート・レジスタ																					
アドレス		6FH				70H				71H				72H				73H			
記号	BANK0	POE				POA				POB				POC				POD			
		0	0	P0E1	P0E0	P0A3	P0A2	P0A1	P0A0	P0B3	P0B2	P0B1	P0B0	P0C3	P0C2	P0C1	P0C0	P0D3	P0D2	P0D1	P0D0

5.1.5　汎用データ・メモリ

汎用データ・メモリは，データ・メモリからシステム・レジスタ（SYSREG）とポート・レジスタを除いた部分で，μPD17120，17121の場合64ニブル，μPD17132，17133，17P132，17P133の場合111ニブルから構成されています。

5.1.6　実装されていないデータ・メモリ

μPD17120，17121のアドレス40H-6EHにはハードウエア上は何も実装されていません。このため，この領域の内容を読み出したときは，不定の値が読み出されます。また，この領域に対するデータの書き込み命令は無効になりますので使用しないでください。

〔メ モ〕

第6章　スタック

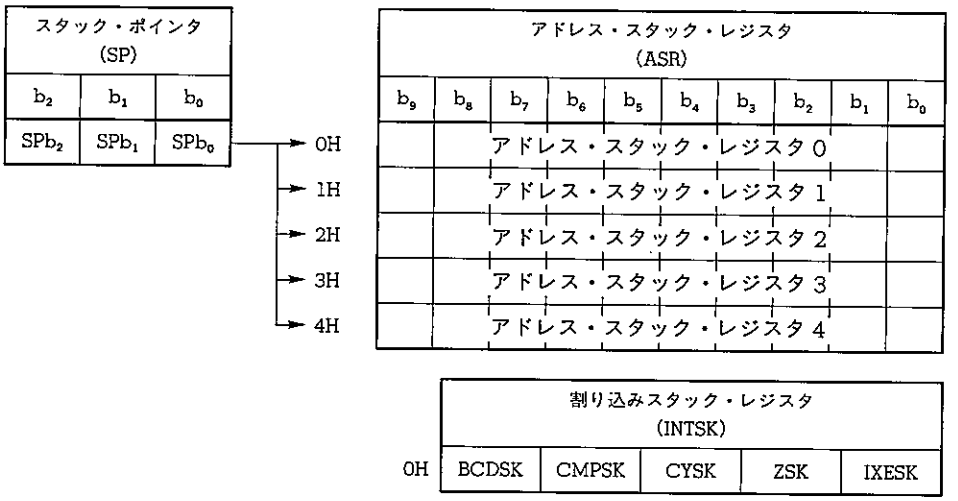
スタックとはサブルーチン・コール時や割り込み受け付け時にプログラムの戻り番地や後述するシステム・レジスタの内容を退避するためのレジスタです。

6.1　スタックの構成

スタックの構成を図6-1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1個，10ビットのアドレス・スタック・レジスタ5個および5ビットの割り込みスタック・レジスタ1個より構成されています。

図6-1　スタックの構成



6.2　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。

備考　PSWORDは割り込みスタック・レジスタに退避後，5ビットすべてが自動的に0にクリアされます。

6.3　アドレス・スタック・レジスタ

アドレス・スタック・レジスタは，図6－1に示すように5×10ビットで構成されるレジスタです。

サブルーチン・コール命令（CALL addr，CALL @AR命令），テーブル参照命令（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，プログラム・カウンタ（PC）の値に＋1した値，つまり戻り番地を格納します。またスタック操作命令（PUSH AR命令）実行時にはアドレス・レジスタ（AR）の内容が格納されます。データが格納されるアドレス・スタック・レジスタは，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。

サブルーチン・リターン命令（RET，RETSK命令），割り込みリターン命令（RETI命令）およびテーブル参照命令（MOVT DBF，@AR命令）の第2命令サイクル実行時は，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。またスタック操作命令（POP AR命令）実行時は，スタック・ポインタで指定されたアドレス・スタック・レジスタの値をアドレス・レジスタに転送して，スタック・ポインタの値を＋1します。

なお，5レベルを越えるサブルーチン・コールや割り込みを実行すると，**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**（暴走対策）。

6.4　割り込みスタック・レジスタ

割り込みスタック・レジスタは，図6－1に示すように1×5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ビットを退避します。次に割り込みリターン命令（RETI命令）が実行されると，割り込みスタック・レジスタの内容をプログラム・ステータス・ワードに復帰します。

割り込みスタック・レジスタは，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。

なお，1レベルを越える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。

備考　PSWORDは割り込みスタック・レジスタに退避後，5ビットすべてが自動的に0にクリアされます。

6.5　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6－1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・カウンタです。スタック・ポインタはレジスタ・ファイルの01H番地に配置されています。リセット時にはスタック・ポインタは5になります。

スタック・ポインタは表6－1に示すようにサブルーチン・コール命令(CALL addr, CALL @AR命令)，テーブル参照命令（MOVT DBF，@AR命令）の第1命令サイクル，スタック操作命令（PUSH AR命令）および割り込み受け付け時に－1され，サブルーチン・リターン命令（RET，RETSK命令），テーブル参照

命令（MOVT DBF, @AR 命令）の第２命令サイクル，スタック操作命令（POP AR 命令）および割り込みリターン命令（RETI 命令）により＋1 されます。また，割り込みが受け付けられるとスタック・ポインタ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 されるのは割り込みリターン命令（RETI 命令）のみです。

表６－１　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
割り込み受け付け	－1	－1
RET RETSK MOVT DBF, @AR（第２命令サイクル） POP AR	＋1	変化しない
RETI	＋1	＋1

スタック・ポインタは前述したように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・ポインタはレジスタ・ファイル上に配置されているため PEEK 命令および 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直接値の読み込み，書き込みが可能です。このときはスタック・ポインタ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。

6.6　サブルーチン命令，テーブル参照命令，割り込み実行時のスタック動作

6.6.1-6.6.3 におのおののスタック動作を示します。

6.6.1　サブルーチン・コール命令（CALL 命令）とリターン命令（RET, RETSK 命令）

表６－２にサブルーチン・コール命令とリターン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6-2　サブルーチン・コール命令とリターン命令の動作

命　令	動　作
CALL addr	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）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ノー・オペレーション命令（NOP 命令）になります。

6.6.2　テーブル参照命令（MOVT DBF, @AR 命令）

表6-3にテーブル参照命令実行時の動作を示します。

表6-3　テーブル参照命令の動作

命　令	命令サイクル	動　作
MOVT DBF, @AR	第1	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2	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

★　**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。

6.6.3　割り込み受け付け時と RETI 命令実行時

表6－4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6－4　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（BCD, CMP, CY, Z, IXE）の各フラグの値を退避 ④ベクタ・アドレスをプログラム・カウンタ（PC）へ転送
RETI	①PSWORD（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

6.7　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・ポインタの値は－1 されて 7H になると, 本製品は, 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 命令や 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6－5に PUSH 命令および POP 命令の動作を示します。

表6－5　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7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。

7.1　システム・レジスタの構成

図7－1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7－1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7－1　システム・レジスタのデータ・メモリ上の配置

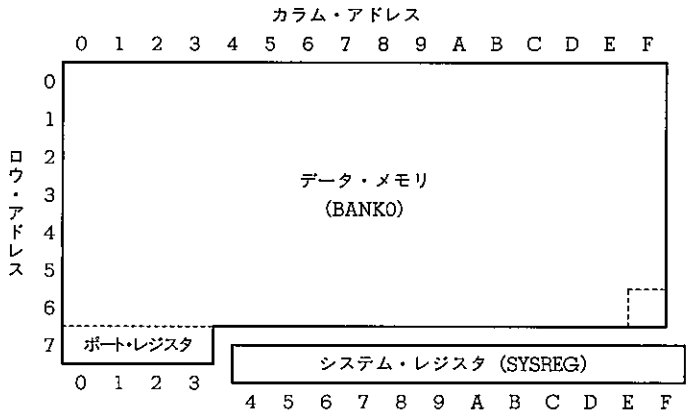


図7-2にシステム・レジスタの構成を示します。図7-2に示すようにシステム・レジスタは，次の7個のレジスタで構成されています。

- 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7-2　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	0 0 0 0 (AR)	0 0 (AR)	(AR)	(AR)	← WR →	0 0 0 0 (BANK)	MPE (IX)	0 0 0 0 (MP)	(IX) →	0 0 0 0 (RP)	(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 0

注　この欄の0が書かれている部分は“0固定”を表します。

7.2　アドレス・レジスタ（AR）

7.2.1　アドレス・レジスタの構成

図7-3にアドレス・レジスタの構成を示します。

図7-3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（AR3-AR0）の16ビットで構成されています。ただし，上位6ビットは常に0に固定されているために実際は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7-3　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	0	0										
リセット	0				0				0				0			

7.2.2　アドレス・レジスタの機能

アドレス・レジスタは，間接分岐命令（BR @AR），間接サブルーチン・コール命令（CALL @AR），テーブル参照命令（MOVT DBF, @AR）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）によりアドレス・レジスタの値をスタックに入れたり，出したりすることができます。

（1）-（4）に各命令時の動作を説明します。

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）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・レジスタの値をプログラム・メモリ・アドレスとするプログラム・メモリの値（16ビットのデータ）をデータ・バッファ（BANK0の0CH-0FH）に読み出すことができます。

(2) 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6章　スタックも参照してください。

(3) 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・レジスタの値をプログラム・メモリ・アドレスとして分岐することができます。

(4) 間接サブルーチン・コール命令（CALL @AR）

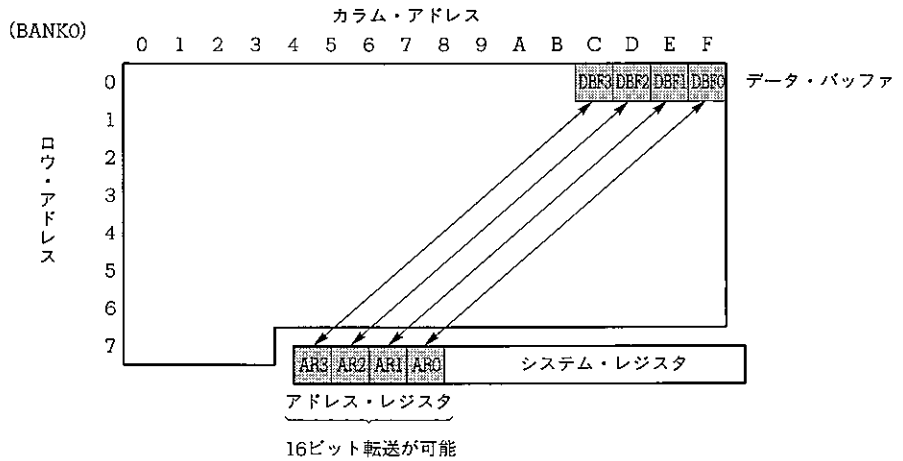
CALL @AR 命令を実行することにより，アドレス・レジスタの値をプログラム・メモリ・アドレスとしてサブルーチンを呼び出すことができます。

(5) 周辺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，データ・メモリ操作命令により，4 ビットごとに操作することができます。また，アドレス・レジスタを周辺レジスタとして扱い，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，アドレス・レジスタはデータ・メモリ操作命令のほかに 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7-4　周辺レジスタとしてのアドレス・レジスタ



7.3　ウインドウ・レジスタ（WR）

7.3.1　ウインドウ・レジスタの構成

図7-5にウインドウ・レジスタの構成を示します。

図7-5に示すようにウインドウ・レジスタ（WR）はシステム・レジスタの78H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7-5　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			

7.3.2　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。

7.4　バンク・レジスタ（BANK）

図7-6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（BANK）の4ビットで構成されています。

バンク・レジスタはRAMのバンクを切り替えるためのレジスタですが，μPD17120サブシリーズにはバンクが1つしかないため，すべて0に固定されています。

図7-6　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			

7.5　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）

7.5.1　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7-7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との論理和で示すアドレスになります。このとき指定バンクもBANKとIXHとの論理和で示すバンクになります。

備考　μPD17120サブシリーズの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）。

7.5.2　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7-7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　μPD17120サブシリーズの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7-7　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7-8　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		m	BANK	m_R	
			Logical OR			Logical OR	(r)
		IXH	IXM	IXL	IXH	IXM	
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 7－1　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

7.5.3　MPE＝0，IXE＝0のとき（データ・メモリ修飾なし）

図7－8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

```
R003    MEM     0.03H
M061    MEM     0.61H
        ADD     R003, M061
```

上の命令を実行すると，図7－9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

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     R005, #8          ; R005←8
        MOV     @R005, M034       ; レジスタ間接転送
```

上の命令を実行すると図7－9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。

すなわち“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・アドレスの@rで指定される間接アドレスのデータ・メモリへ転送します。

間接転送アドレスは，ロウ・アドレスがmと同一（上記ではロウ・アドレス3）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（上記では8）になります。すなわち上記では38Hとなります。

例3. ジェネラル・レジスタがロウ・アドレス0にあるとき

```
R00B    MEM     0.0BH
M034    MEM     0.34H
        MOV     R00B,  #0EH      ; R00B←0EH
        MOV     M034,  @R00B     ; レジスタ間接転送
```

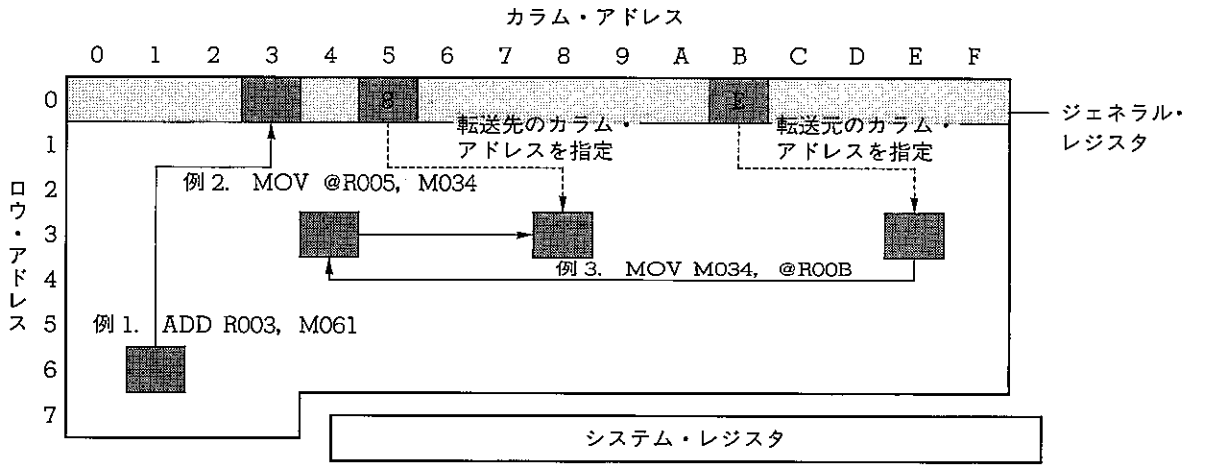
上の命令を実行すると，図7-9に示すようにデータ・メモリM034へ，アドレス3EH番地のデータ・メモリの内容を転送します。

すなわち“MOV m, @r”命令はmで指定されるデータ・メモリへmと同じロウ・アドレスの@rで指定される間接アドレスのデータ・メモリの内容を転送します。

間接転送アドレスは，ロウ・アドレスがmと同一（上記ではロウ・アドレス3）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（上記では0EH）になります。すなわち上記ではアドレス3EHとなります。

これは例2と比べると転送するデータ・メモリ・アドレスのソース(転送元)とディスティネーション（転送先）が，入れ替わったことになります。

図7-9　MPE=0，IXE=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

7.5.4　MPE＝1，IXE＝0のとき（ななめ間接転送）

図7-8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

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     MPL, #0110B     ; MP←6
        MOV     R005, #8        ; R005←8
        MOV     MPH, #1000B     ; MPE←1
        MOV     @R005, M034     ; レジスタ間接転送
```

上の命令を実行すると図7-10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**7.5.3**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（**7.5.3**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）。

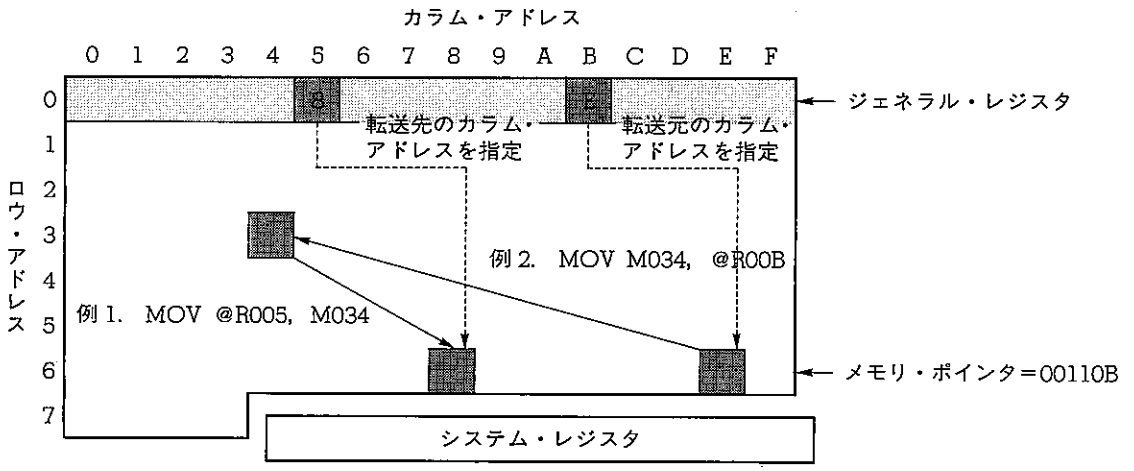
したがって，MPE＝1とすることによりジェネラル・レジスタ間接転送をななめに行うことが可能になります。

例2. ジェネラル・レジスタがロウ・アドレス0のとき

```
R00B    MEM     0.0BH
M034    MEM     0.34H
        MOV     MPL,  #0110B      ; MP←6
        MOV     MPH,  #1000B      ; MPE←1
        MOV     R00B, #0EH        ; R00B←0EH
        MOV     M034, @R00B       ; レジスタ間接転送
```

上の命令を実行すると，図7-10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7-10　MPE=1，IXE=0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

7.5.5　MPE＝0，IXE＝1 のとき（インデクス修飾）

図7－8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

```
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
```

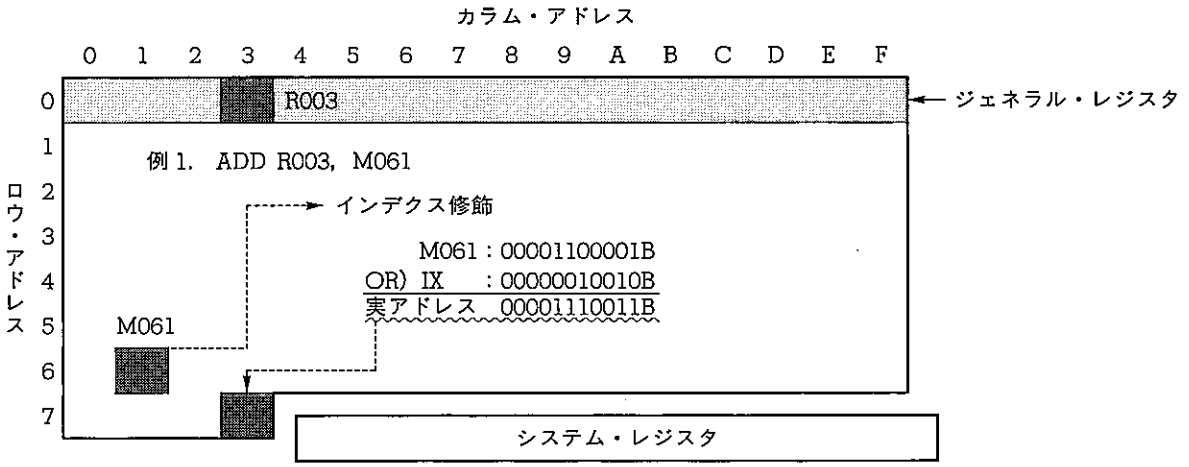
例1の命令を実行すると，図7－11に示すようにアドレスが 73H 番地のデータ・メモリ（実アドレス）の内容とジェネラル・レジスタ R003（アドレスの 03H 番地）の内容を加算し，結果をジェネラル・レジスタ R003 へ格納します。

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（**7.5.3** 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 7 - 11　MPE=0, IXE= 1 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

ジェネラル・レジスタがロウ・アドレス0にあるものとする。

```
R005    MEM     0.05H
M034    MEM     0.34H
        MOV     IXL, #0001B             ; IX← 00000000001B
        MOV     IXM, #0000B             ;
        MOV     IXH, #0000B             ; MPE← 0
        OR      PSW, #.DF.IXE AND 0FH   ; IXE←1
        MOV     R005, #8                ; R005←8
        MOV     @R005, M034             ; レジスタ間接転送
```

上の命令を実行すると図7-12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。

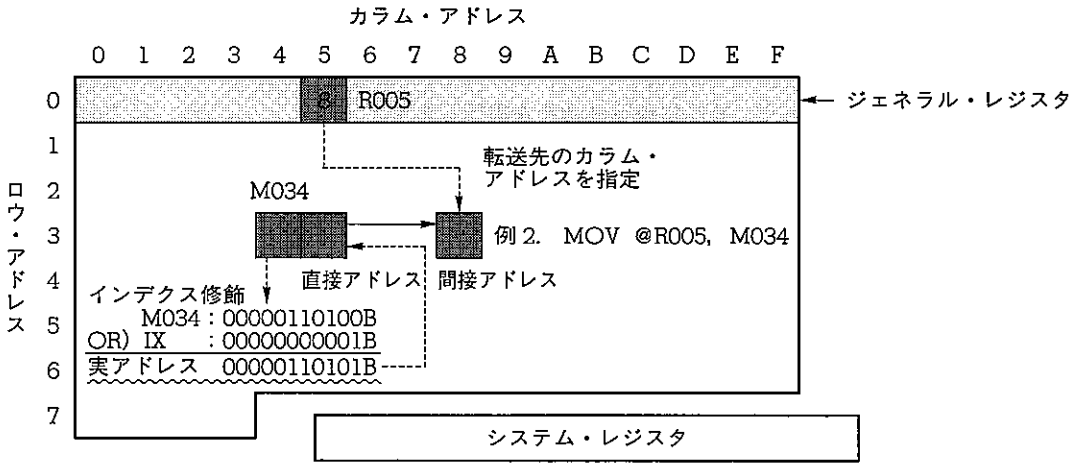
すなわち，IXE＝1のときに“MOV @r, m”命令を実行すると，“m”で指定されるデータ・メモリ・アドレス（直接アドレス）をインデクス・レジスタの内容でアドレス修飾し，“@r”で指定される間接アドレスのバンクとロウ・アドレスもインデクス・レジスタで修飾されます。

“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が35Hとなり，間接アドレスが38H番地になります。

これは**7.5.3**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（**7.5.3**例3では直接アドレスは修飾されない）。

図7-12　MPE=0，IXE=1のときのジェネラル・レジスタ間接転送動作例



例3.　すべてのデータ・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
                                      ; レス修飾されない。
        SKE   IXM,  #0111B            ; 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7－13に示すように，1つの要素が8ビットの一次元配列の要素A（N）に

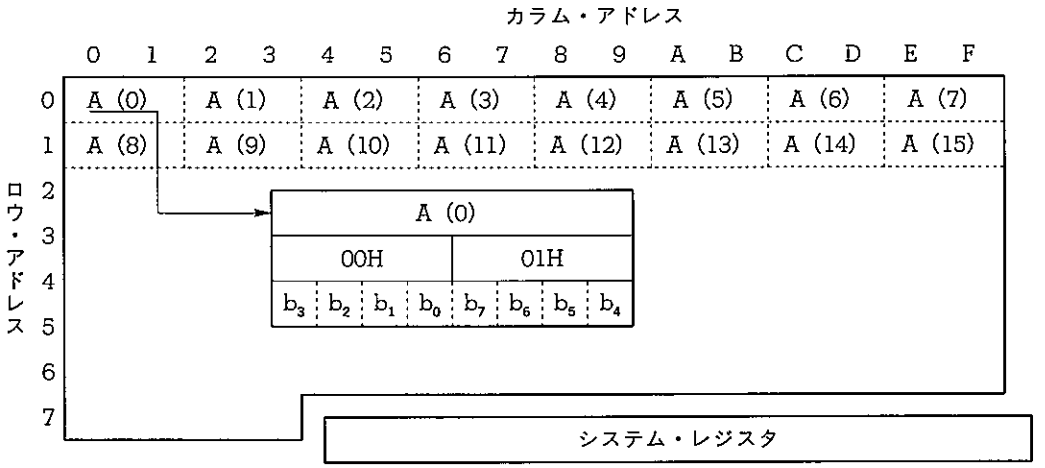
A(N)＝A(N)＋4　（0≦N≦15）

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7－13　MPE＝0，IXE＝1のときの動作例（配列の処理）



7.6　ジェネラル・レジスタ・ポインタ（RP）

7.6.1　ジェネラル・レジスタ・ポインタの構成

図7-14にジェネラル・レジスタ・ポインタの構成を示します。

図7-14　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
フラグ								BCD
データ	0	0	0	0 (RP)				
リセット時	0				0			

図7-14に示すように，ジェネラル・レジスタ・ポインタはシステム・レジスタの7DH番地（RPH）の4ビットと7EH番地（RPL）の上位3ビットの計7ビットで構成されています。ただし，7DH番地の4ビットは常に0に固定されているため実際には7EH番地の上位3ビットが有効になります。

リセット時はすべて0にクリアされます。

7.6.2　ジェネラル・レジスタ・ポインタの機能

ジェネラル・レジスタ・ポインタはデータ・メモリ上にジェネラル・レジスタを指定するために使用します（ジェネラル・レジスタについては**第8章　ジェネラル・レジスタ（GR）**を参照してください）。

ジェネラル・レジスタはデータ・メモリ上の同一ロウ・アドレスである16ニブルを指定できます。したがって図7-15に示すようにジェネラル・レジスタ・ポインタによってどのロウ・アドレスを使用するかを指定します。

ジェネラル・レジスタ・ポインタの有効ビットは3ビットであるため，ジェネラル・レジスタとして指定できるデータ・メモリのロウ・アドレスはBANK0の0H-7H番地になります。すなわちすべてのデータ・メモリがジェネラル・レジスタとして指定できます。

データ・メモリがジェネラル・レジスタに指定されるとジェネラル・レジスタとデータ・メモリの演算および転送が行えます。

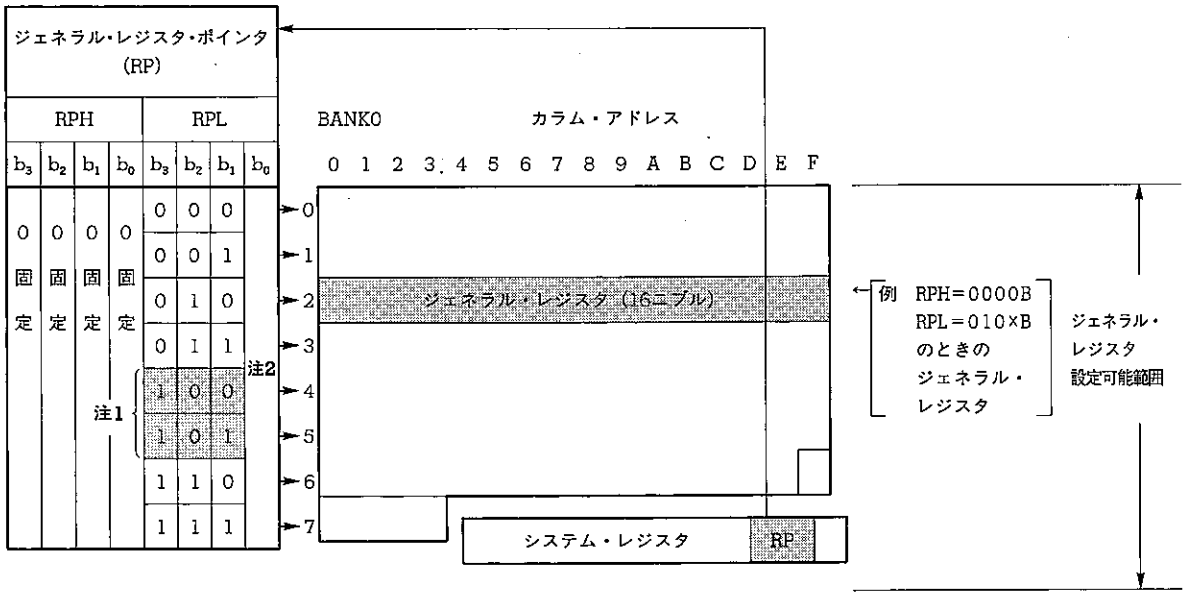
ただし，μPD17120，17121の場合，アドレス40H-6EHは非実装領域となっていますので，指定するのを避けてください。

たとえば，

ADD　r，m　または　LD　r，m

命令等を実行すると，命令のオペランド“r”でアドレス指定されるジェネラル・レジスタと“m”でアドレス指定されるデータ・メモリ間で加算や転送が行えます。

図7-15　ジェネラル・レジスタの構成



注 1．μPD17120，17121では指定を避けてください。

2．BCDフラグに割り当てられています。

7.7　プログラム・ステータス・ワード（PSWORD）

7.7.1　プログラム・ステータス・ワードの構成

図7－16にプログラム・ステータス・ワードの構成を示します。

図7－16　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7－16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。 ★

7.7.2　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7-17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7-17　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。

7.7.3　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**7.5　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)**を参照してください。

7.7.4　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7-2のようになります。

表7-2　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
        SUB     M002, #5    ; データは壊れない。
        SUB     M003, #6    ;
;       CLR1    CMP
        SKT     Z           ; CMPはビット判断命令で自動的にクリア
        BR      DIFFER      ; ≠456H
        BR      AGREE       ; ＝456H
```

7.7.5　キャリー・フラグ（CY）

CY フラグは加算命令および減算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。

7.7.6　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。

7.7.7　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。

7.8　システム・レジスタ使用時の注意

7.8.1　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（AS17K）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（AS17K）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7-2　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19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
                MOV     76H, #1010B   ; 76H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・
                MOV     M037, #0101B  ; メモリ・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・ファイルに定義されている。
```

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)
```

7.8.2 “0”に固定されているシステム・レジスタの取り扱い

システム・レジスタの中であらかじめ“0”に固定されているデータ（**図7-2　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) デバイス動作上

“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) 17Kシリーズのインサーキット・エミュレータ（IE-17K，IE-17K-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3に1を書き込む。
```

2.
```
MOV     IXL, #1111B  ;
MOV     IXM, #1111B  ;
MOV     IXH, #0001B  ;
ADD     IXL, #1      ;
ADDC    IXM, #0      ;
ADDC    IXH, #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この命令により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) 17K シリーズのアセンブラ（AS17K）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（AS17K）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（AS17K）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，μPD17120 サブシリーズでは BANK0 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。

第8章　ジェネラル・レジスタ（GR）

ジェネラル・レジスタ（GR）はデータ・メモリ上に配置されるレジスタで，データ・メモリとの直接演算や，転送を行います。

8.1　ジェネラル・レジスタの構成

ジェネラル・レジスタの構成を図8-1に示します。

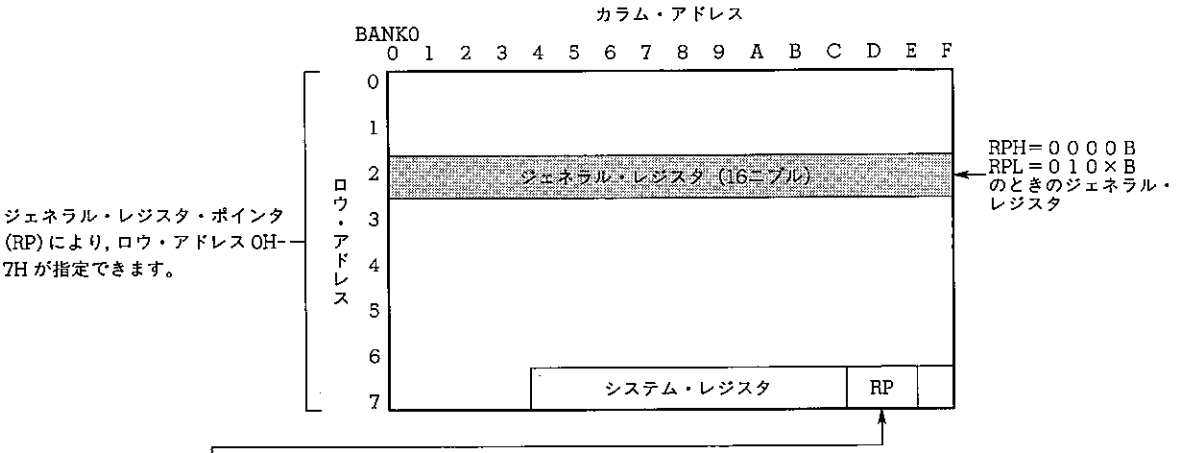
図8-1に示すように，データ・メモリ上で同一ロウ・アドレスである16ニブル（16×4ビット）をジェネラル・レジスタとして使用できます。

どのロウ・アドレスを使用するかは，システム・レジスタのジェネラル・レジスタ・ポインタ（RP）によって設定します。RPは3ビットが有効であるためジェネラル・レジスタとして指定できる範囲はデータ・メモリのロウ・アドレス 0H-7H になります。ただし，μPD17120，17121の場合，アドレス 40H-6EHは非実装領域になっていますので，指定するのは避けてください。

8.2　ジェネラル・レジスタの機能

ジェネラル・レジスタを使用することにより，データ・メモリとジェネラル・レジスタとの間で演算や転送を1命令で行うことが可能になります。ジェネラル・レジスタはすなわちデータ・メモリであるため，言い換えれば1命令でデータ・メモリ同士の演算や転送が可能になります。また，ジェネラル・レジスタは，データ・メモリ上にあるので他のデータ・メモリと同様にデータ・メモリ操作命令で制御することができます。

図8-1　ジェネラル・レジスタの構成



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

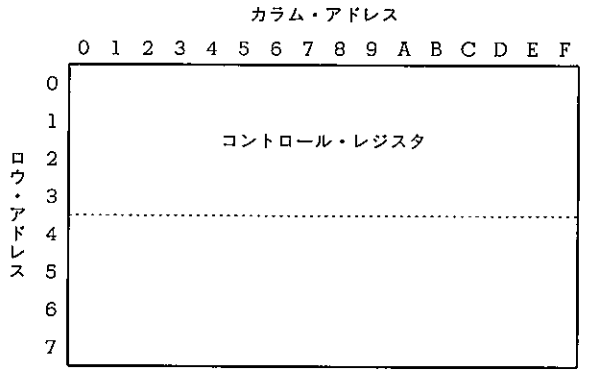
図９－１にレジスタ・ファイルの構成を示します。

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



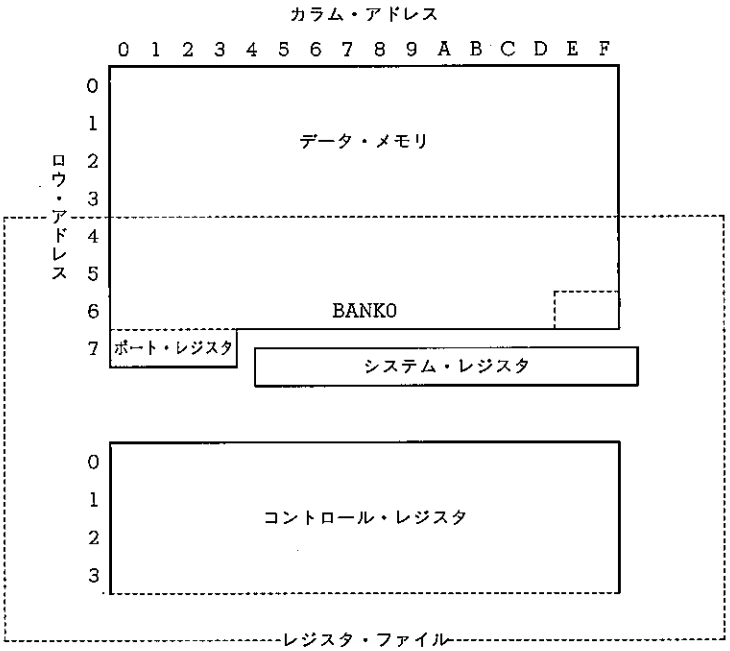
９.１.２　レジスタ・ファイルとデータ・メモリ

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。

すなわちレジスタ・ファイルの 40H-7FH 番地は，データ・メモリのそのとき選択されているバンクのアドレス 40H から 7FH 番地と同じメモリが存在しています。

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・ファイルは主として周辺ハードウエアの条件設定を行う制御レジスタです。

この制御レジスタはレジスタ・ファイルの中のコントロール・レジスタ（00H-3FH 番地）に配置されています。

残りのレジスタ・ファイル（40H-7FH 番地）はデータ・メモリと重なっているため，**9.2.3** に示すようにレジスタ・ファイル操作命令の“PEEK”および“POKE”命令により操作できる点を除けば通常のデータ・メモリと何ら変わりはありません。

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。

- スタック
- パワーオン/パワーダウン・リセット
- タイマ
- シリアル・インタフェース
- INT 端子
- コンパレータ
- 汎用ポート
- 割り込み

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込み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込む。

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。

以下に使用例を示します。

```
例    M030    MEM     0.30H       ; データ・メモリの 30H 番地を WR の退避領域として使用
      M032    MEM     0.32H       ; データ・メモリの 32H 番地を WR の操作領域として使用
      RF11    MEM     0.91H       ; シンボル定義
      RF33    MEM     0.B3H       ; レジスタ・ファイルの 00H-3FH 番地のシンボル定義
      RF70    MEM     0.70H       ; は BANK0の 80H-BFH として定義する必要がありま
      RF73    MEM     0.73H       ; す。詳細については 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。
            ; BANK0
  ①         PEEK    WR, RF11    ;

            CLR1    MPE         ; 汎用データ・メモリ（00H-3FH 番地）に WR の内容
            CLR1    IXE         ; を退避する例を示します。一例として，アドレス修
            OR      RPL, #0110B ; 飾をせず，データ・メモリの 30H 番地に退避する場
  ②         LD      M030, WR    ; 合を示します。

  ③         POKE    RF73, WR    ; 40H-7FH 番地のデータ・メモリとコントロール・レ
  ④         PEEK    WR, RF70    ; ジスタは WRと PEEK, POKE 命令で直接データのや
  ⑤         POKE    RF33, WR    ; りとりが行えます。

  ⑥         ST      WR, M032    ;
```

図９－３に動作例を示します。

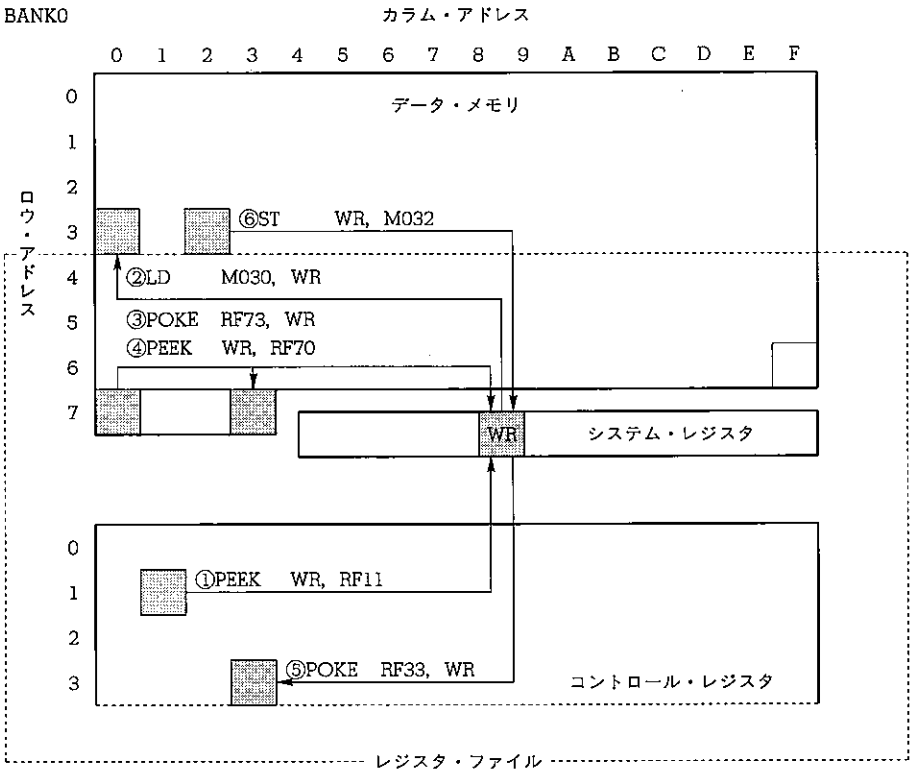
図９－３に示すように，コントロール・レジスタ（00H-3FH 番地）は“PEEK WR, rf”および“POKE rf, WR”命令が実行されると，“rf”でアドレス指定されるレジスタ・ファイルの内容をウインドウ・レジスタに対してのみ読み込みまたは書き込みを行います。

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，そのとき認定されているバンクのデータ・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。

コントロール・レジスタは，組み込みマクロ命令を使用することにより１ビット単位で操作することができます。

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

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。

μPD17120，17121 ではそのうちの17ニブル，μPD17132，17133，17P132，17P133 では20ニブルを使用しています。

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。

• TMRES　　（RF：11H，ビット 2）

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19-1，19-2　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，17K シリーズのアセンブラ（AS17K）およびインサーキット・エミュレータ（IE-17K，IE-17K-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（AS17K）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-17K, IE-17K-ET）使用時（パ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

17Kシリーズのアセンブラ（AS17K）を使用する際には，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る。

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（AS17K）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“71H番地”のデータ・メモリになる。
        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（AS17K）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，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の周辺回路（アドレス・レジスタ，シリアル・インタフェース，タイマ）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BANK0のアドレス0CH－0FHが割り当てられており，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

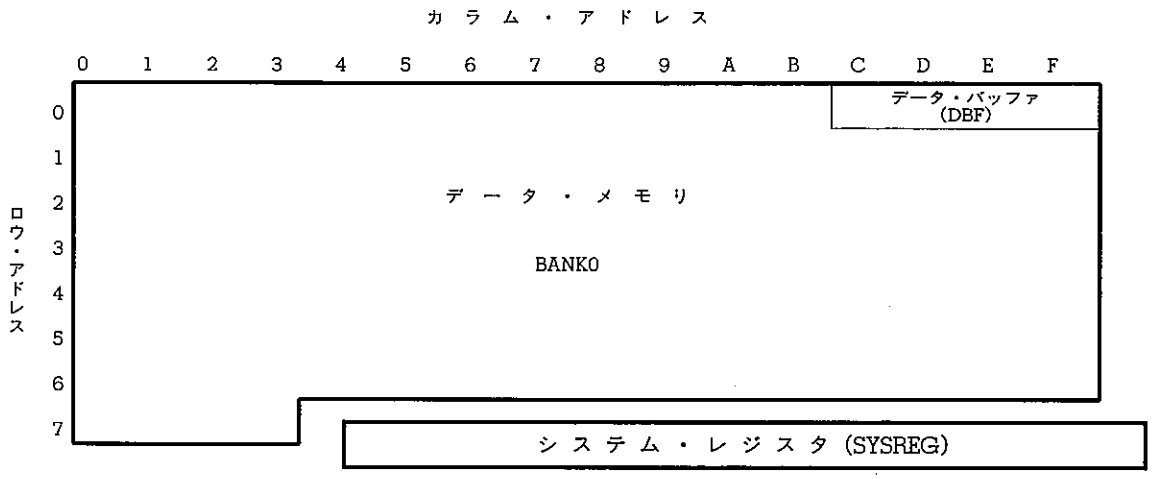


図10－2にデータ・バッファの構成を示します。図10－2に示すようにデータ・バッファはデータ・メモリの0FH番地のビット0をLSBとし，0CH番地のビット3をMSBとする16ビットで構成されています。

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	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・バッファ	ビット	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	記　号	DBF3				DBF2				DBF1				DBF0			
	データ	∧ MSB ∨				データ											∧ LSB ∨

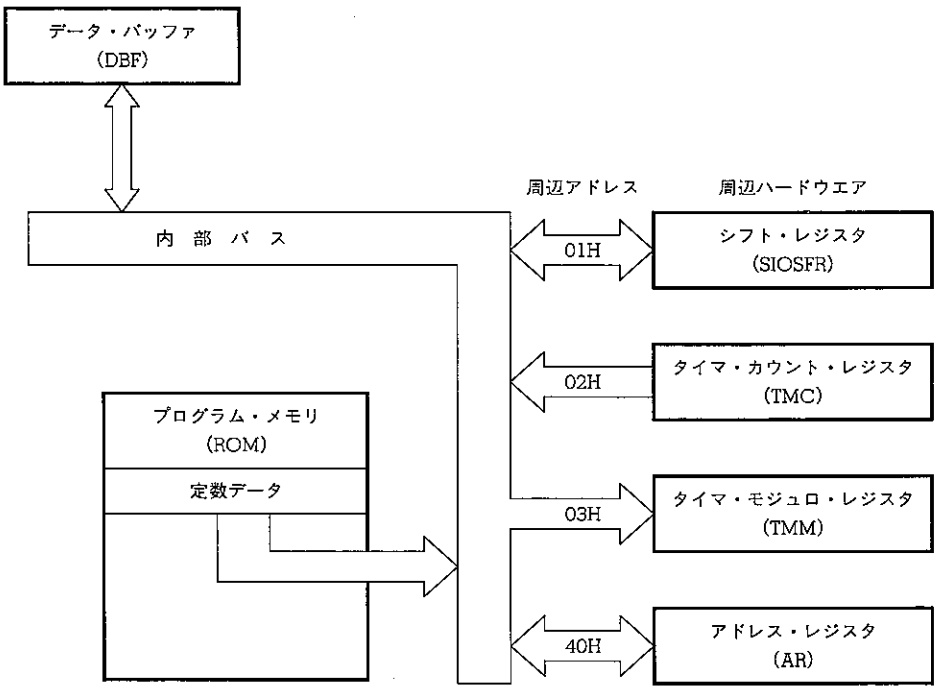
データ・バッファは，データ・メモリ上に配置されているため，すべてのデータ・メモリ操作命令で操作できます。

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，専用命令である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。

GET　　DBF，p：データ・バッファ（DBF）にpでアドレスされる周辺ハードウエアのデータを読み込む。

PUT　　p，DBF：pでアドレスされる周辺ハードウエアにデータ・バッファ(DBF)のデータを設定する。

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

このとき，書き込み専用（PUTのみ）や，読み出し専用（GETのみ）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（PUTのみ）周辺ハードウエアに対して読み出し（GET）命令実行時
  不定の値が読み出されます。
- 読み出し専用（GETのみ）周辺ハードウエアに対して書き込み（PUT）命令実行時
  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	シフト・レジスタ	○	○	8ビット
02H	TMC	タイマ・カウント・レジスタ	×	○	8ビット
03H	TMM	タイマ・モジュロ・レジスタ	○	×	8ビット

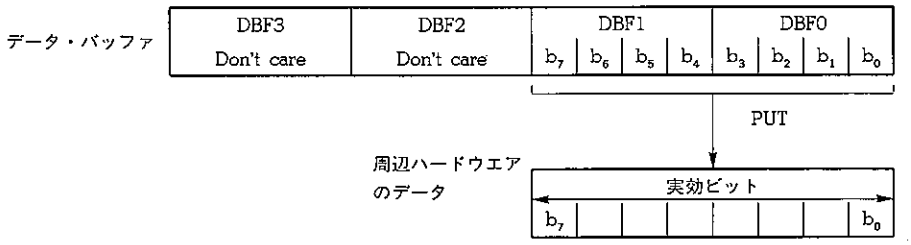
(2) 16ビット単位で入出力を行う周辺ハードウエア

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

10.2.2　周辺ハードウエアとのデータ転送

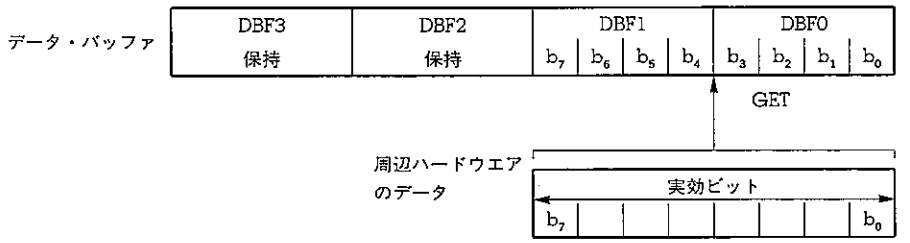
データ・バッファと各周辺ハードウエアとのデータ転送時は８ビットまたは16ビット単位で行います。このとき，PUT および GET 命令は，データ・ビットが16ビットであっても１命令サイクルで実行できます。

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。

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）



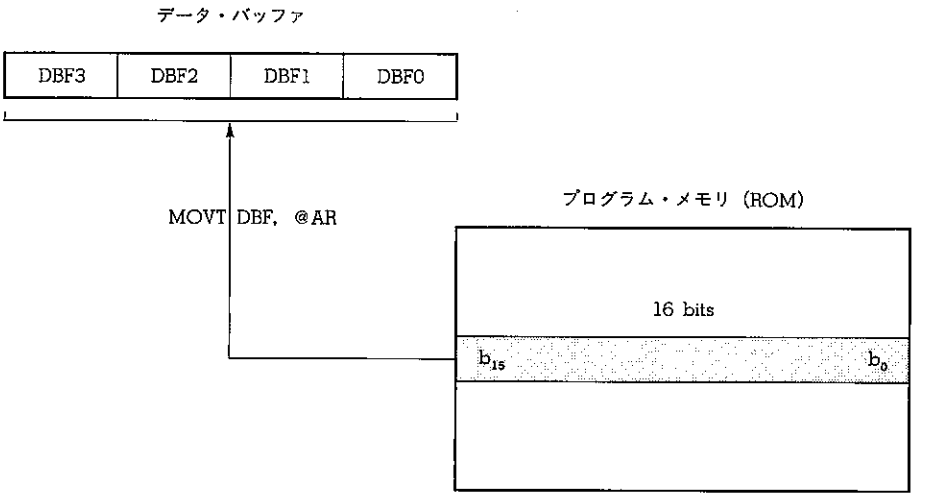
８ビット・データ読み込み時にはデータ・バッファの上位８ビット（DBF3, DBF2）の値は変化しません。

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　演算論理ユニット（ALU）

ALUは4ビット・データの算術演算，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。

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。

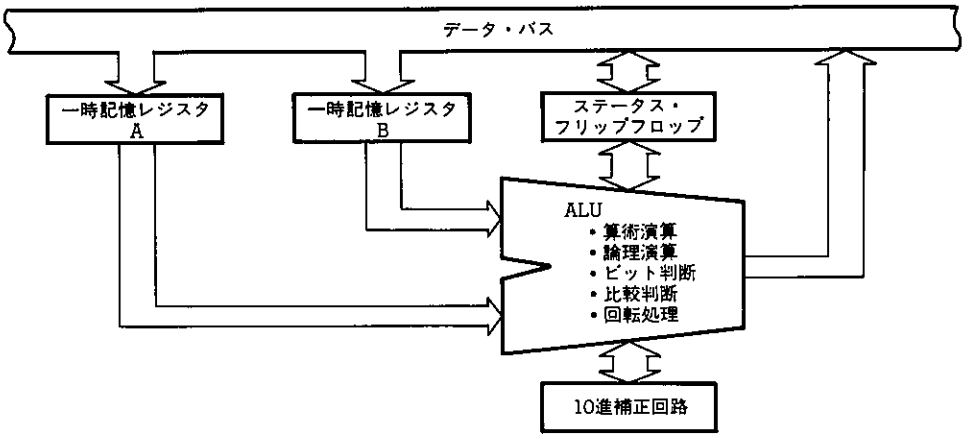
論理演算には論理積（AND），論理和（OR）および排他的論理和（XOR）があります。論理演算は，ジェネラル・レジスタの内容とデータ・メモリの内容との演算，またはデータ・メモリの内容とイミーディエト・データとの演算が行えます。

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。

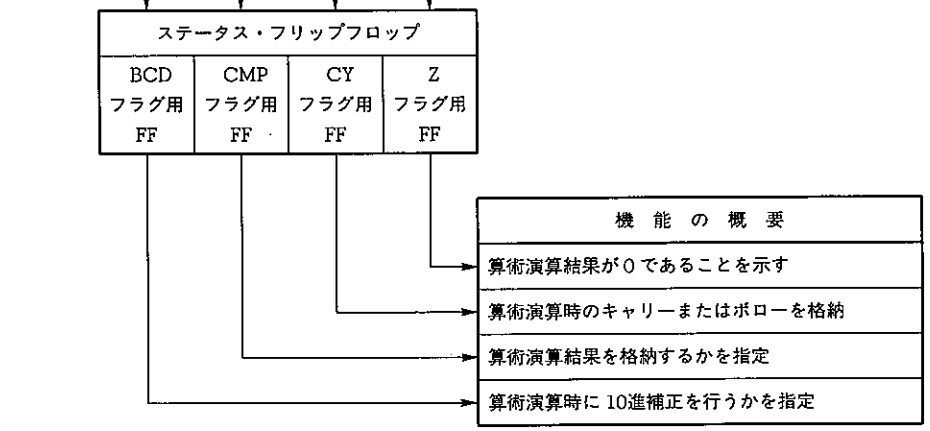
比較判断はデータ・メモリの内容とイミーディエト・データとの比較を行い，“等しい”，“等しくない”，“以上”および“未満”の判断を行います。

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)。

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



〔メ モ〕

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→$(r)_{b3}$→$(r)_{b2}$→$(r)_{b1}$→$(r)_{b0}$→CY	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

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，すべての算術演算結果が10進補正されます。

リセット(0) されているときは10進補正されません。

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

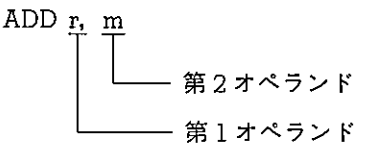
演算結果	2進4ビット減算		BCD 減算	
	CY	演算結果	CY	演算結果
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	1100
11	0	1011	1	1101
12	0	1100	1	1110
13	0	1101	1	1111
14	0	1110	1	1100
15	0	1111	1	1101
−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（次の命令はノー・オペレーション（NOP 命令）命令として実行されます）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進変換については **表 11 - 2　2進 4 ビット演算結果と BCD 演算結果** 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
CY　　CY

－10D～－1D とは同様に

1, 0110B～1, 1111B のことです
CY　　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” または “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7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 のビット 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，この命令の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。

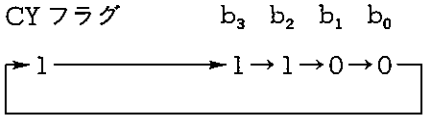
11.7.1　右回転処理

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。

例2. MOV　PSW,　#0000B ; CY フラグをリセット(0)
　　MOV　R1,　#1000B
　　MOV　R2,　#0100B
　　MOV　R3,　#0010B
　　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
　　MOV　R2,　#0100B
　　MOV　R3,　#0010B
　　ADDC　R3, R3
　　ADDC　R2, R2
　　ADDC　R1, R1

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 0A（POA_0, POA_1, POA_2, POA_3）

ポート 0A 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 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。

1 ビットごとに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 P0ABIO0-P0ABIO3（35H 番地）により行います。

P0ABIOn=0 のとき（n=0-3），ポート 0A の各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，端子の状態が読み込まれます。

P0ABIOn=1 のとき（n=0-3），ポート 0A の各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込み命令を実行すると，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 P0ABIOn は “0” になり，P0A 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

P0ABIOn RF : 35H	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 OB（POB_0，POB_1，POB_2，POB_3）

ポート OB 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（24H 番地のビット 0）により行います。

P0BGIO＝0 のとき，ポート 0B 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，端子の状態が読み込まれます。

P0BGIO＝1 のとき，ポート 0B 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込み命令を実行すると，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 P0BGIO は“0”になり，P0B 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：24H, ビット 0	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 0C（$P0C_0$，$P0C_1$，$P0C_2$，$P0C_3$）
…μPD17120，17121の場合

ポート 0C 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（34H 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込み命令を実行すると，端子の状態が読み込まれます。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込み命令を実行すると，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。

表 12-3　ポート・レジスタ（0.72H）への書き込みと読み出し（μPD17120，17121）

（n＝0-3）

P0CBIOn RF：34H	端子の入力/出力	BANK0 72H	
		書き込み	読み出し
0	入力	可能 P0C ラッチに書き込み	P0C の端子の状態
1	出力		P0C のラッチの内容

12.4　ポート 0C（$P0C_0/Cin_0$, $P0C_1/Cin_1$, $P0C_2/Cin_2$, $P0C_3/Cin_3$）…μPD17132, 17133, 17P132, 17P133 の場合

ポート 0C 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（34H 番地）により行います。

P0CBIOn＝0 のとき（n＝0-3），P0C の各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。

P0CBIOn＝1 のとき（n＝0-3），P0C の各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。

またポート 0C はコンパレータのアナログ入力端子として使用できます。ポートとアナログ入力端子の切り替えは，レジスタ・ファイル上の P0C0IDI-P0C3IDI（23H 番地）によって行います。

P0CnIDI＝0 のとき（n＝0-3），$P0C_n/Cin_n$ 端子はポートとして機能し，P0CnIDI＝1 のとき（n＝0-3），$P0C_n/Cin_n$ 端子は，コンパレータのアナログ入力端子として機能します。したがって，アナログ入力として使用する端子はプログラムの初期設定において P0CnIDI に 1 をセットしてください。

コンパレートするアナログ入力端子の切り替えは，CMPCH0 および CMPCH1（RF：1CH 番地）で行います。また，コンパレータのアナログ入力端子として使用する場合は，P0CBIOn に “0” をセットし，入力ポートに設定してください（**13.2　コンパレータ**参照）。

リセット時には P0CBIOn，P0CnIDI は “0” になり（n＝0-3），ポート 0C の端子はすべて入力ポートに設定されます。また，ポートの出力ラッチの内容も “0” になります。

表 12-4　ポート・レジスタ（0.72H）への書き込みと読み出しおよび端子の機能の選択

（n＝0-3）

P0CnIDI RF：23H	P0CBIOn RF：34H	機　　能	BANK0 72H	
			書き込み	読み出し
0	0	入力ポート	可能 P0C ラッチに 書き込み	P0C の 端子の状態
	1	出力ポート		P0C の ラッチの内容
1	0	コンパレータのアナログ入力[注1]		P0C の 端子の状態
	1	出力ポートおよびコンパレータのアナログ入力[注2]		P0C の ラッチの内容

注 1. 端子をコンパレ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。

12.5　ポート OD（$POD_0/\overline{SCK}$, POD_1/SO, POD_2/SI, $POD_3/\overline{TMOUT}$）

ポート 0D 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。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を内蔵することが指定できます注。

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（33H 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込み命令を実行すると，端子の状態が読み込まれます。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，読み込み命令を実行すると，端子の状態ではなく，出力ラッチの値が取り込まれます。

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出力として使用できます。ポート（$P0D_0$-$P0D_2$）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（0AH のビット 0）によって行います。また，ポート（$P0D_3$）とタイマ出力（$\overline{TMOUT}$）の切り替えはレジスタ・ファイル上の TMOSEL（12H のビット 0）によって行います。TMOSEL＝1を選択すると，タイマのリセット時には“1”を出力し，タイマ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。

注　μPD17P132, 17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

表 12-5　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			
TMOSEL RF：12H ビット0	SIOEN RF：0AH ビット0	P0DBIOn RF：33H ビットn	$P0D_0/\overline{SCK}$	$P0D_1/SO$	$P0D_2/SI$	$P0D_3/\overline{TMOUT}$
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			$\overline{TMOUT}$
		1	出力ポート			
	1	0	$\overline{SCK}$	SO	SI	
		1				

表 12-6　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		不定
$\overline{TMOUT}$		出力ラッチの内容

注意　シリアル・インタフェース使用後の $P0D_1/SO$ 端子の出力ラッチの内容は，SIOSFR（シフト・レジスタ）の内容に影響され不定となっています。したがって $P0D_1/SO$ 端子を出力ポートとして使用する場合は，出力ラッチの内容を再設定してください。

12.6　ポート OE（POE_0，POE_1/V_{ref}）

…V_{ref} は μPD17132，17133，17P132，17P133 のみ

ポート OE は出力ラッチ付き 2 ビットの入出力ポートです。データ・メモリの 6FH 番地のビット 0，ビット 1 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。また，マスク・オプションによりビット単位でプルアップ抵抗を内蔵することができます。

POE_1/V_{ref} 端子はコンパレータ（μPD17132，17133，17P132，17P133 のみ内蔵）の外部レファレンス電圧入力との兼用端子となっており，レファレンス電圧選択レジスタ（CMPVREF0-CMPVREF3）の値によりポートから外部レファレンス電圧入力へと機能が変わります（**13.2　コンパレータ**を参照してください）。

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 POEBIO0，POEBIO1（32H 番地のビット 0，ビット 1）により行います。

POEBIOn＝0のとき（n＝0，1），POE の各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。

POEBIOn＝1のとき（n＝0，1），POE の各端子は出力モードとなり，出力ラッチに書かれた内容が端子に出力されます。

また，入力/出力のどちらのモードにも関わらず，読み出し命令を実行すると，出力ラッチの内容ではなく，端子の状態が取り込まれます。

リセット時には POEBIOn は “0” になり（n＝0，1），POE の各端子は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。

なお，6FH 番地のビット 2 とビット 3 に対する書き込み命令は無効となり，読み出した場合には 0 が読み出されます。

備考　μPD17P132，17P133 にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

表 12-7　ポート・レジスタ（0.6FH.0，0.6FH.1）への書き込みと読み出し

（n ＝0，1）

POEBIOn RF：32H	端子の入力/出力	BANK0 6FH	
		書き込み	読み出し
0	入力	可能	POE の端子の状態
1	出力	POE の出力ラッチに書き込み	

12.6.1　ポート・レジスタの操作時の注意

μPD17120サブシリーズの入出力ポートのうちポート OE だけは，出力モードであっても読み込み時には端子の状態を読み込みます。

したがって，ポート・レジスタに組み込みマクロ命令（SETn/CLRn など）や AND/OR/XOR 命令などでビット操作すると，意図していない端子の状態も変化してしまうことがあります。

特に，ポート OE を外部で強制的にロウ・レベルにしている場合には注意が必要です。

ポート OE に CLR1 POE1 命令（AND 6FH，＃1101B 命令と同じ）を実行するとポート・レジスタおよびマイコン内部の状態が変化する例を図 12－1 に示します。

たとえばポート OE の POE_1 端子，POE_0 端子を両方とも出力として使用し，POE_1 端子と POE_0 端子からハイ・レベルが出力され，POE_0 端子を外部で強制的にロウ・レベルにしている場合を考えると，ポート OEの各状態は図 12－1①のようになります（μPD17120サブシリーズには POE_3 端子と POE_2 端子は存在しませんがプログラム上は仮想的に存在しているものとして扱います）。

POE_1 端子をロウ・レベルにするため，CLR1 POE1 命令を実行すると，ポート OE の各状態は図 12－1②のように変化します。このとき，POE_1 端子は当然ロウ・レベル出力に変化しますが，それ以外にハイ・レベルを出力していたはずの POE_0 端子 からもロウ・レベルが出力されるようにポート・レジスタの値が変化しています。これはポート・レジスタではなく端子の状態に対して CLR1 POE1 命令が実行されたために生じた結果です。

この現象を防ぐには，変化させる端子だけではなく，すべての端子の状態を MOV 命令などで設定するようにします。この例で POE_1 端子だけをロウ・レベルにするには，MOV 6FH，＃1101B命令を使用すれば問題ありません。

また，同様の理由により ポート OE を入出力混在で使用する場合には，入力として使用する端子は必ず入力モード（POEBIOn＝0）で使用してください。

図 12－1　CLR1 POE1 命令によるポート・レジスタの変化

① 命令実行前

	POE_3	POE_2	POE_1	POE_0
ポート・レジスタ	存在しない		1	1
マイコンの状態	－	－	H 出力	H 出力
端子の状態	－	－	H	L（強制）

CLR1 POE1 命令を実行
[AND 6FH, ＃1101B]

② 命令実行後

	POE_3	POE_2	POE_1	POE_0
ポート・レジスタ	存在しない		0	0
マイコンの状態	－	－	L 出力	L 出力
端子の状態	－	－	L	L

H：ハイ・レベル　　L：ロウ・レベル

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OB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12-2　グループI/Oのポート制御レジスタ

RF：24H

	ビット3	ビット2	ビット1	ビット0
	0	0	0	P0BGIO
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

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

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0A，ポート 0C，ポート 0D，ポート0E 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 12-3　ビット I/O のポート制御レジスタ（1/4）

RF：35H

	ビット3	ビット2	ビット1	ビット0
	P0ABIO3	P0ABIO2	P0ABIO1	P0ABIO0
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

P0ABIO0	機　　能
0	$P0A_0$ を入力モードに設定
1	$P0A_0$ を出力モードに設定

P0ABIO1	機　　能
0	$P0A_1$ を入力モードに設定
1	$P0A_1$ を出力モードに設定

P0ABIO2	機　　能
0	$P0A_2$ を入力モードに設定
1	$P0A_2$ を出力モードに設定

P0ABIO3	機　　能
0	$P0A_3$ を入力モードに設定
1	$P0A_3$ を出力モードに設定

図 12-3　ビット I/O のポート制御レジスタ (2/4)

RF : 34H

	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 12-3　ビット I/O のポート制御レジスタ（3/4）

RF：33H

	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

図 12-3　ビット I/O のポート制御レジスタ（4/4）

RF：32H

	ビット 3	ビット 2	ビット 1	ビット 0
	0	0	P0EBIO1	P0EBIO0
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

P0EBIO0	機　　能
0	$P0E_0$ を入力モードに設定
1	$P0E_0$ を出力モードに設定

P0EBIO1	機　　能
0	$P0E_1$ を入力モードに設定
1	$P0E_1$ を出力モードに設定

第13章　周辺ハードウエア

13.1　8ビット・タイマ・カウンタ（TM）

μPD17120サブシリーズは8ビット・タイマ・カウンタを1系統内蔵しています。

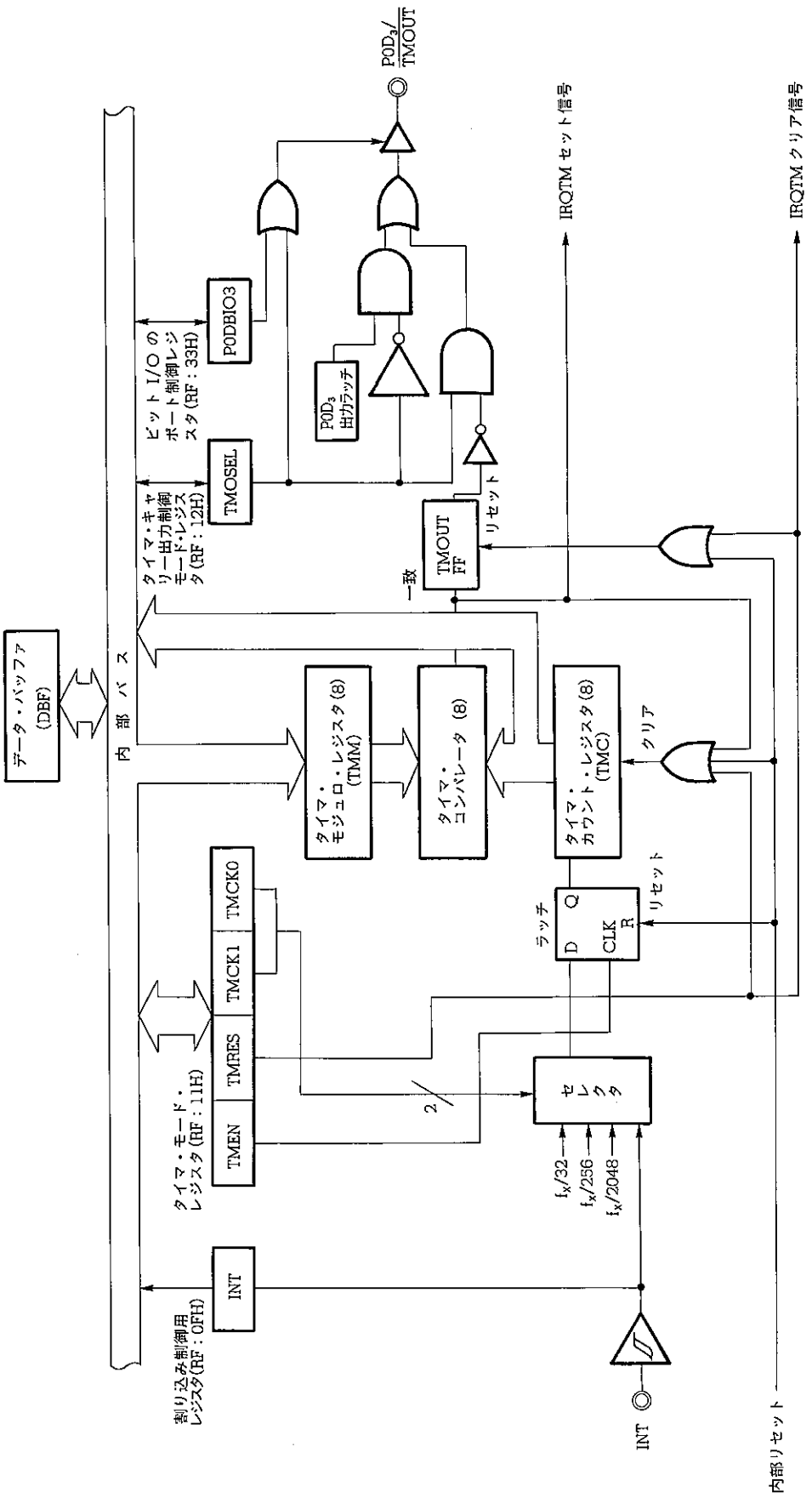
8ビット・タイマ・カウンタの制御は，PUT/GET命令を使ったハードウエアの操作とPEEK/POKE命令を使ったレジスタ・ファイル上のレジスタの操作により行います。

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　モジュロ・レジスタは書き込み専用レジスタです。
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★



13.1.2　8ビット・タイマ・カウンタの制御レジスタ

8ビット・タイマ・カウンタを制御するレジスタには，タイマ・モード・レジスタとタイマ・キャリー出力制御モード・レジスタがあります。

図13-2，13-3に8ビット・タイマ・カウンタの制御レジスタの構成を示します。

図13-2　タイマ・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
	TMEN	TMRES	TMCK1	TMCK0
リード/ライト	R/W			
リセット時の初期値	1	0	0	0

リード＝R，ライト＝W

TMCK1	TMCK0	タイマのカウント・パルスの選択
0	0	f_X/256
0	1	f_X/32
1	0	f_X/2048
1	1	INT端子からの外部クロック

TMRES	タイマのリセット
0	タイマに影響なし
1	カウント・レジスタ(TMC)とIRQTMをリセット

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

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

備考　TMENは，タイマ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（0：カウント停止状態，1：カウント動作中）。

13.1.3　8ビット・タイマ・カウンタの動作

(1) カウント・レジスタ

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。

- マイコンがリセットされたとき（**第16章　リセット**参照）
- 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
- レジスタ・ファイルのTMRESに“1”が書き込まれたとき

(2) モジュロ・レジスタ

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

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（IRQTM）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（IPTM）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（IRQTM）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。

13.1.4　カウント・パルスの選択

カウント・パルスの選択は，TMCK0およびTMCK1で選択します。

システム・クロック（f_X）を2048分周，256分周または32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TMCK0＝0，TMCK1＝0になっており，f_X/256が選択されています。

また，電源立ち上げ時およびリセット時，タイマ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TMCK0＝0，TMCK1＝0になっており，カウント・パルスには，f_X/256が選択されています。そして，TMEN＝1が初期値として設定されていますので，f_X＝8 MHz時では，リセットしてから約8 ms（f_X＝2 MHz時では約32 ms）経過後，マイコンは0000H番地スタートします（**第16章　リセット**参照）。

13.1.5　モジュロ・レジスタへのカウント値の設定と計算方法

カウント値は，データ・バッファ（DBF）を介してモジュロ・レジスタに設定します。

(1) モジュロ・レジスタへのカウント値の設定

モジュロ・レジスタへのカウント値の設定は，PUT 命令によりデータ・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，03H に割り付けられています。

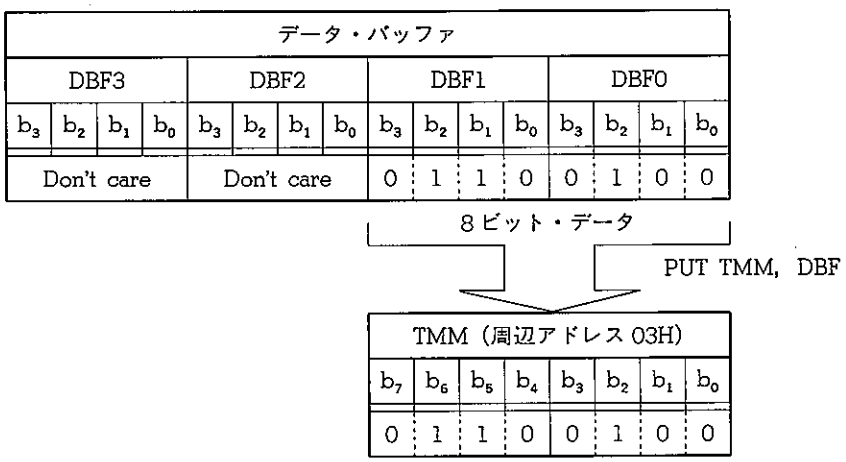
PUT 命令による値を転送する場合，データ・バッファの下位８ビット（DBF1，DBF0）のデータがモジュロ・レジスタに転送されます。

カウント値の設定例を図 13－３に示します。

図 13-3　モジュロ・レジスタへのカウント値の設定

タイマ・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   TMM, DBF    ；予約語 “TMM” 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取ることはできません。また，8 ビット・タイマ・カウンタが動作中に，PUT TMM, 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。

(2) インターバル時間の計算方法 ★

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）

(3) 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：f_X＝8 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ソース・クロック（$f_X/256$，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

$$N=\frac{T}{(\text{分解能})}-1$$

$$=\frac{7\times10^{-3}}{32\times10^{-5}}-1$$

$$=217.75\fallingdotseq218 \text{（：DAH）}$$

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 TMEN, TMRES, NOT TMCK1, NOT TMCK0
                        ; 組み込みマクロ命令 “INITFLG” を用いて TMEN, TMRES
                        ; をセット，タイマのソース・クロックを “fx/256” に設定，
                        ; カウント・スタート
```

★

13.1.6　インターバル時間の誤差

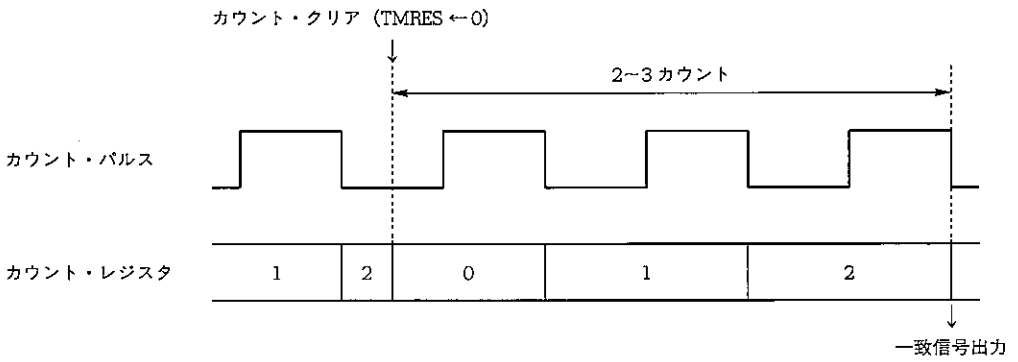
インターバル時間には，最大で−1.5カウントの誤差を生じることがあります。モジュロ・レジスタの設定値が小さい場合にはご注意ください。

(1) カウント中にカウント・レジスタを０クリアしたときの誤差（最大−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，TMRES 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 TMRES フラグをセット (1) してカウントを０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 13-4　カウント中にカウント・レジスタを０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，TMEN＝1←0と同時に，TMRES←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－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，TMRES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。

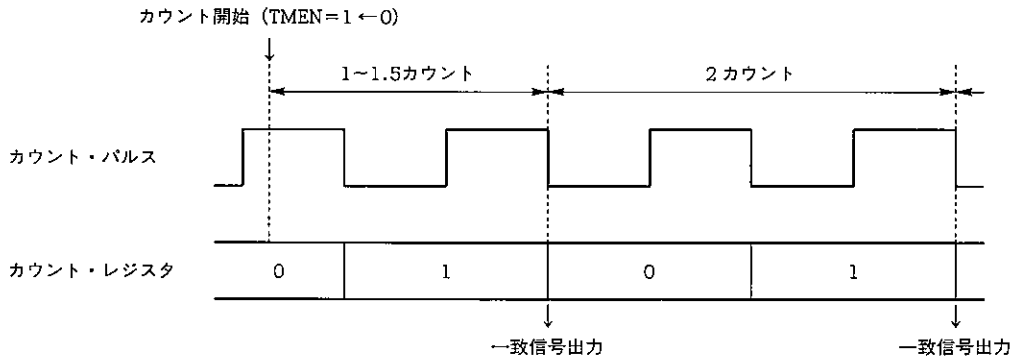
また，TMENフラグをセット(1)してカウント停止状態からカウントを開始した場合，カウント・パルスがロウ・レベルから始まるかハイ・レベルから始まるかによって，1カウント目のタイミングが次のように異なります。

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

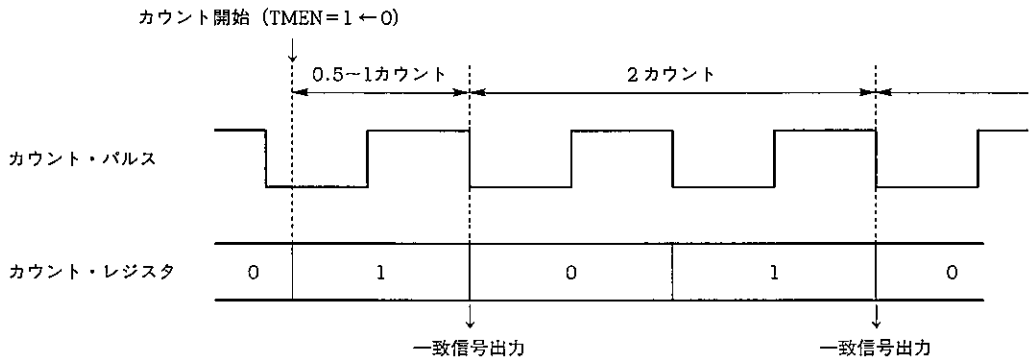
したがって，カウント開始後の最初の1回目だけは，一致信号が出るまでの時間に－0.5～－1.5カウントの誤差が生じます。次にモジュロ・レジスタに1を設定した場合のカウントの例を示します。

図13－5　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・パルスがハイ・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。

13.1.7　カウント・レジスタの値の読み取り

(1) カウント値の読み取り

カウント・レジスタのカウント値の読み取りは，GET命令によりDBF（データ・バッファ）を介して行います。

タイマのカウント・レジスタは，周辺アドレス02Hに割り付けられています。

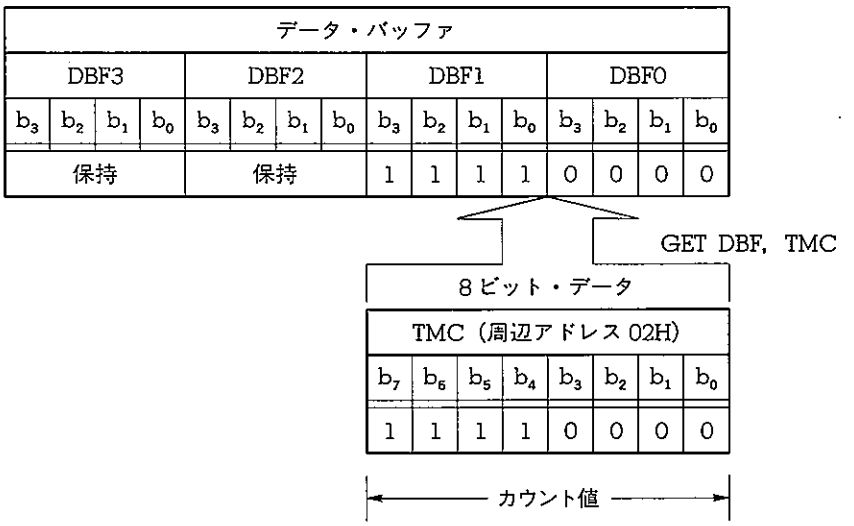
GET命令により，カウント・レジスタの値をDBFに読み取ることができます。なお，GET命令実行中には，カウント・レジスタはカウント動作を停止し，カウント値を保持している状態に入ります。なお，タイマが動作中でかつGET命令実行中にカウント・パルスが入力された場合，そのカウントを保留し，GET命令終了後カウント・レジスタを1つインクリメントしたのちカウント動作を続ける機能を備えています。

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 13-6　8 ビット・カウンタのカウント値の読み取りの例

タイマのカウンタの値が FOH のとき

GET DBF, TMC ; 予約語 DBF および TMC を使用した例



(2) プログラム例

●INT 端子から入力されるパルス幅の測定（システム・クロック：f_X=8 MHz）

INT 端子による外部割り込みの発生間隔をタイマで測定する例を次に示します。このときの INT 端子からのパルス幅は，タイマがカウント・アップする時間内でなければなりません。

（プログラム例）

```
CNTTMH  MEM     0.30H     ; カウント値の退避領域をシンボル定義
CNTTML  MEM     0.31H     ;

        ORG     00H       ; 割り込みのベクタ・アドレスを指定
        BR      MAIN      ;
        ORG     03H       ;
        BR      INTJOB    ;
        ⋮
INITIAL :
        INITFLG IP, NOT IPTM, NOT IPSIO
                          ; 外部割り込み以外の割り込みを禁止
```

```
        INITFLG  NOT IEGMD1, IEGMD0
                                    ; INT 端子からの入力を立ち下がりエッジに設定
        CLR1     IRQ                ; INT 端子からの割り込み要求信号をクリア
        INITFLG  TMEN, TMRES, NOT TMCK1, TMCK0
                                    ; タイマのソース・クロックを “fx/32” に設定,
                                    ; タイマ・カウント・レジスタとIRQTM をクリア
                                    ; してタイマ・スタート
        EI
LOOP :
        BR       LOOP

INTJOB :                            ; 割り込み受け付け直後，ベクタ割り込みは，
        GET      DBF, TMC           ; 自動的に割り込み禁止状態になる
                                    ; タイマのカウント値の読み取り
        MOV      RPH, #.DM. (CNTTMH SHR 7) AND 0FH
                                    ; シンボル定義された “CNTTMH”, “CNTTML”
                                    ; を用いてジェネラル・レジスタ・ポインタを設定
        AND      RPL, #0001B        ;
        OR       RPL, #.DM. (CNTTML SHR 3) AND 0EH
                                    ; このとき BCD フラグは以前の状態を保持

        LD       CNTTMH, DBF1       ; カウント値をカウント退避領域に格納
        LD       CNTTML, DBF0       ;
        EI                          ; メイン処理プログラム実行の際，割り込み許可
                                    ; 状態にする
        RETI                        ; メイン処理プログラムに復帰
```

13.1.8　タイマ出力

TMOSELフラグを“1”にセットすることにより，$P0D_3/\overline{TMOUT}$端子は，タイマの一致信号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。TMOSEL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$P0D_3/\overline{TMOUT}$端子に出力されます。

また，$P0D_3/\overline{TMOUT}$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，$P0D_3/\overline{TMOUT}$端子の初期状態はハイ・インピーダンスとなります。

なお，内部のタイマ出力フリップフロップは，TMEN＝1にした時点から動作を開始していますので，初期値から必ず出力を開始させるためには，TMRES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。

備考　μPD17P132，17P133にはマスク・オプションがありません。

図13-7　タイマ出力制御モード・レジスタ

RF：12H

	ビット3	ビット2	ビット1	ビット0
	0	0	0	TMOSEL
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

TMOSEL	タイマ出力の制御
0	$P0D_3/\overline{TMOUT}$端子はポートとして機能
1	$P0D_3/\overline{TMOUT}$端子はタイマの一致信号出力として機能

13.1.9　タイマの分解能と最長設定時間

タイマの各ソース・クロックにおける分解能と最長設定時間を表13-1に示します。

表13-1　タイマの分解能と最長設定時間

システム・クロック	モード・レジスタ		タイマ	
	TMCK1	TMCK0	分解能	最長設定時間
8 MHz時[注1]	0	0	32 μs	8.192 ms
	0	1	4 μs	1.024 ms
	1	0	256 μs	65.536 ms
	1	1	INT端子[注2]	
4.19 MHz時[注1]	0	0	約61.1 μs	約15.6 ms
	0	1	約7.64 μs	約1.96 ms
	1	0	約489 μs	約125 ms
	1	1	INT端子[注2]	
2 MHz時	0	0	128 μs	32.768 ms
	0	1	16 μs	4.096 ms
	1	0	1.024 ms	262.144 ms
	1	1	INT端子[注2]	
500 kHz時	0	0	512 μs	131.072 ms
	0	1	64 μs	16.384 ms
	1	0	4.096 ms	1.048576 s
	1	1	INT端子[注2]	

注 1. μPD17120, 17132, 17P132の発振保証範囲はf_{CC}＝400 kHz～2.4 MHzです。

2. INT端子のハイ，ロウ・レベル幅は，V_{DD}＝4.5～5.5 V時10 μs(MIN.)，V_{DD}＝2.7～5.5 V時50 μs(MIN.)です。詳細はデータ・シートを参照してください。

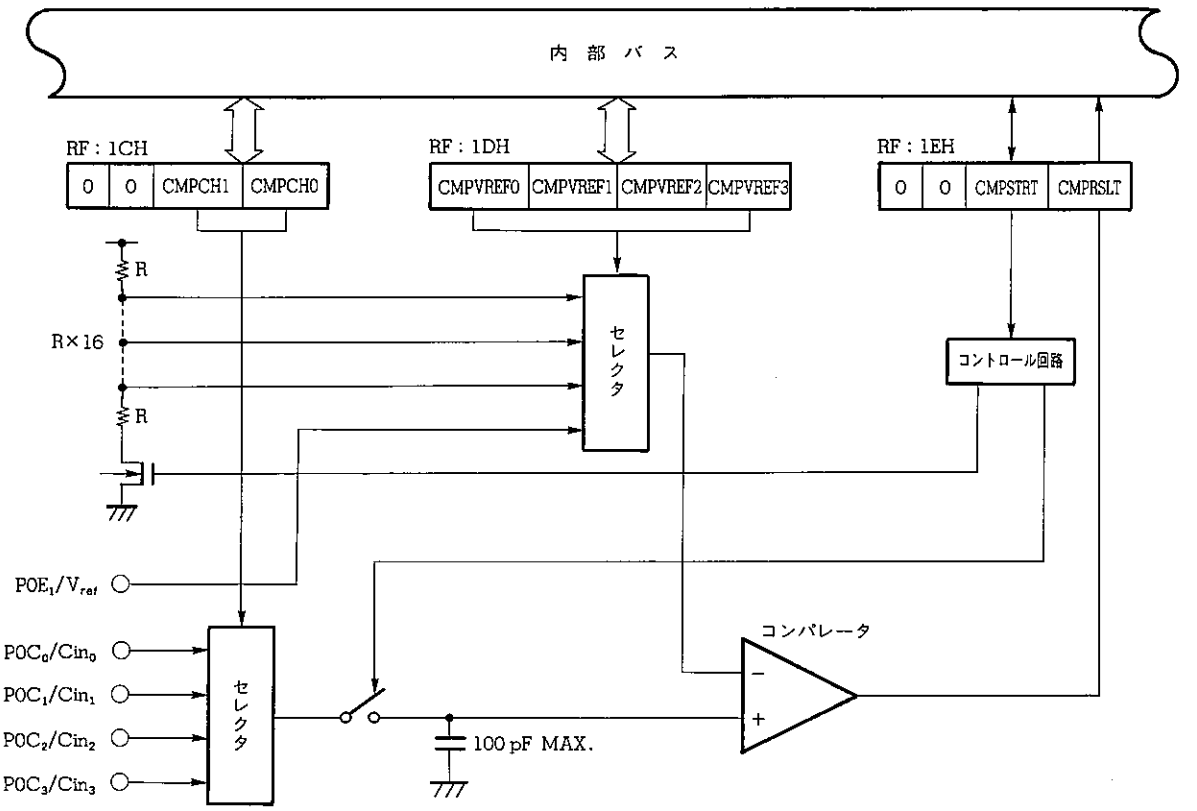
13.2　コンパレータ（μPD17132, 17133, 17P132, 17P133 のみ）

μPD17132, 17133, 17P132, 17P133のコンパレータはアナログ入力（Cin_0-Cin_3）とレファレンス電圧（外部１種類，内部15種類）の大小を比較し，比較結果を CMPRSLT（RF：1EH，ビット 0）に格納する機能を持っています。

15種類の内部レファレンス電圧を用い，ソフトウエアにより４ビットの A/D コンバータとしても使用できます。

13.2.1　コンパレータの構成

図 13-8　コンパレータの構成



備考　アナログ入力のサンプリング時間は次のとおりです。

μPD17132, 17P132：$8/f_{CC}$（4 μs，2 MHz時）

μPD17133, 17P133：$28/f_X$（3.5 μs，8 MHz時）

13.2.2　コンパレータの機能

4チャネルのアナログ入力を持つコンパレータです。

コンパレータのアナログ入力として使用する端子は，あらかじめプログラムの初期設定でP0CnIDI(n=0-3) に1をセットしてください（**第12章　ポート**参照)。

アナログ入力(Cin_0-Cin_3)をコンパレータ入力チャネル選択フラグ(CMPCH1, CMPCH0 RF：1CH, ビット1，ビット0）により選択し，レファレンス電圧をコンパレータ・レファレンス電圧選択フラグ(CMPVREF0-CMPVREF3 RF：1DH)により外部レファレンス，または内部レファレンス電圧15種類を選択することができます。

★ コンパレート時間はコンパレータ・スタート・フラグ（CMPSTRT RF：1EH，ビット1）に1をセットしたのち，μPD17132，17P132の場合は2命令実行時間を，μPD17133，17P133の場合は6命令実行時間を要します。

比較結果はコンパレータ比較結果フラグ（CMPRSLT RF：1EH，ビット0）に格納されます。

また，コンパレータ・スタート・フラグを読み出すことにより，コンパレータが動作中か比較完了かを知ることができます。

CMPSTRT＝1…コンパレータ動作中（アナログ電圧比較中）
CMPSTRT＝0…コンパレータ動作停止中（比較完了）

なお，CMPSTRT＝1のとき，すなわちコンパレータ・アナログ電圧比較中は，コンパレータ入力チャネル選択フラグ(CMPCH1, CMPCH0)およびコンパレータ・レファレンス電圧選択フラグ(CMPVREF0-CMPVREF3) の操作を行ってもデータは変化せず無効になり，コンパレータ動作モードの変更は行えません。

CMPSTRTが0にクリアされるのは，コンパレータの電圧比較動作が完了またはSTOP命令が実行された場合のみです。

★ **注意　スタンバイ機能を使用する場合には，必ずコンパレータの動作停止を待ってからスタンバイ命令（HALT/STOP）を実行するようにしてください。**
コンパレータ動作中にSTOP命令やHALT命令が実行されると，コンパレータ動作は中断されます。このとき内部レファレンス電圧を選択していると，内部の抵抗ラダーに電流が流れたままの状態になりますので，スタンバイ・モード中の消費電流が増えます。

図13-9にコンパレータ入力チャネル選択レジスタを，図13-10にレファレンス電圧選択レジスタを，図13-11にコンパレータ動作制御レジスタを示します。

図 13-9　コンパレータ入力チャネル選択レジスタ

RF：1CH

	ビット3	ビット2	ビット1	ビット0
	0	0	CMPCH1	CMPCH0
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

CMPCH1	CMPCH0	コンパレータ入力チャネル選択
0	0	Cin_0 を選択
0	1	Cin_1 を選択
1	0	Cin_2 を選択
1	1	Cin_3 を選択

図 13-10　レファレンス電圧選択レジスタ

RF：1DH

	ビット3	ビット2	ビット1	ビット0
	CMPVREF3	CMPVREF2	CMPVREF1	CMPVREF0
リード/ライト	R/W			
リセット時の初期値	1	0	0	0

リード＝R，ライト＝W

CMPVREF3	CMPVREF2	CMPVREF1	CMPVREF0	選択されたレファレンス電圧
0	0	0	0	V_{ref} 端子に印加されている電圧
0	0	0	1	1/16 V_{DD}
0	0	1	0	2/16 V_{DD} (1/8 V_{DD})
0	0	1	1	3/16 V_{DD}
0	1	0	0	4/16 V_{DD} (1/4 V_{DD})
0	1	0	1	5/16 V_{DD}
0	1	1	0	6/16 V_{DD} (3/8 V_{DD})
0	1	1	1	7/16 V_{DD}
1	0	0	0	8/16 V_{DD} (1/2 V_{DD})
1	0	0	1	9/16 V_{DD}
1	0	1	0	10/16 V_{DD} (5/8 V_{DD})
1	0	1	1	11/16 V_{DD}
1	1	0	0	12/16 V_{DD} (3/4 V_{DD})
1	1	0	1	13/16 V_{DD}
1	1	1	0	14/16 V_{DD} (7/8 V_{DD})
1	1	1	1	15/16 V_{DD}

注意　**CMPSTRT＝1** のとき，レファレンス電圧選択レジスタへの書き込み命令は，データも変化せず無効となります。

図 13-11　コンパレータ動作制御レジスタ

RF：1EH

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

リード=R，ライト=W

CMPRSLT	コンパレータ動作比較結果
0	アナログ入力（Cin_0-Cin_3）からの電圧が外部または内部レファレンス電圧より低いとき
1	アナログ入力（Cin_0-Cin_3）からの電圧が外部または内部レファレンス電圧より高いとき

CMPSTRT	コンパレータの動作確認（読み出し時）
0	コンパレータ動作停止中，またはコンパレータの電圧比較動作完了
1	コンパレータ動作中

備考　CMPSTRT が 0 にクリアされるのは，コンパレータの電圧比較動作が完了または STOP 命令が実行された場合のみです。

CMPSTRT	コンパレータ動作の開始（書き込み時）
0	無効
1	コンパレータ動作の開始

13.3　シリアル・インタフェース（SIO）

μPD17120サブシリーズのシリアル・インタフェースは，シフト・レジスタ（SIOSFR，8ビット），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。

13.3.1　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	$f_X/16$
1	0	$f_X/128$
1	1	$f_X/1024$

(2) 転送動作

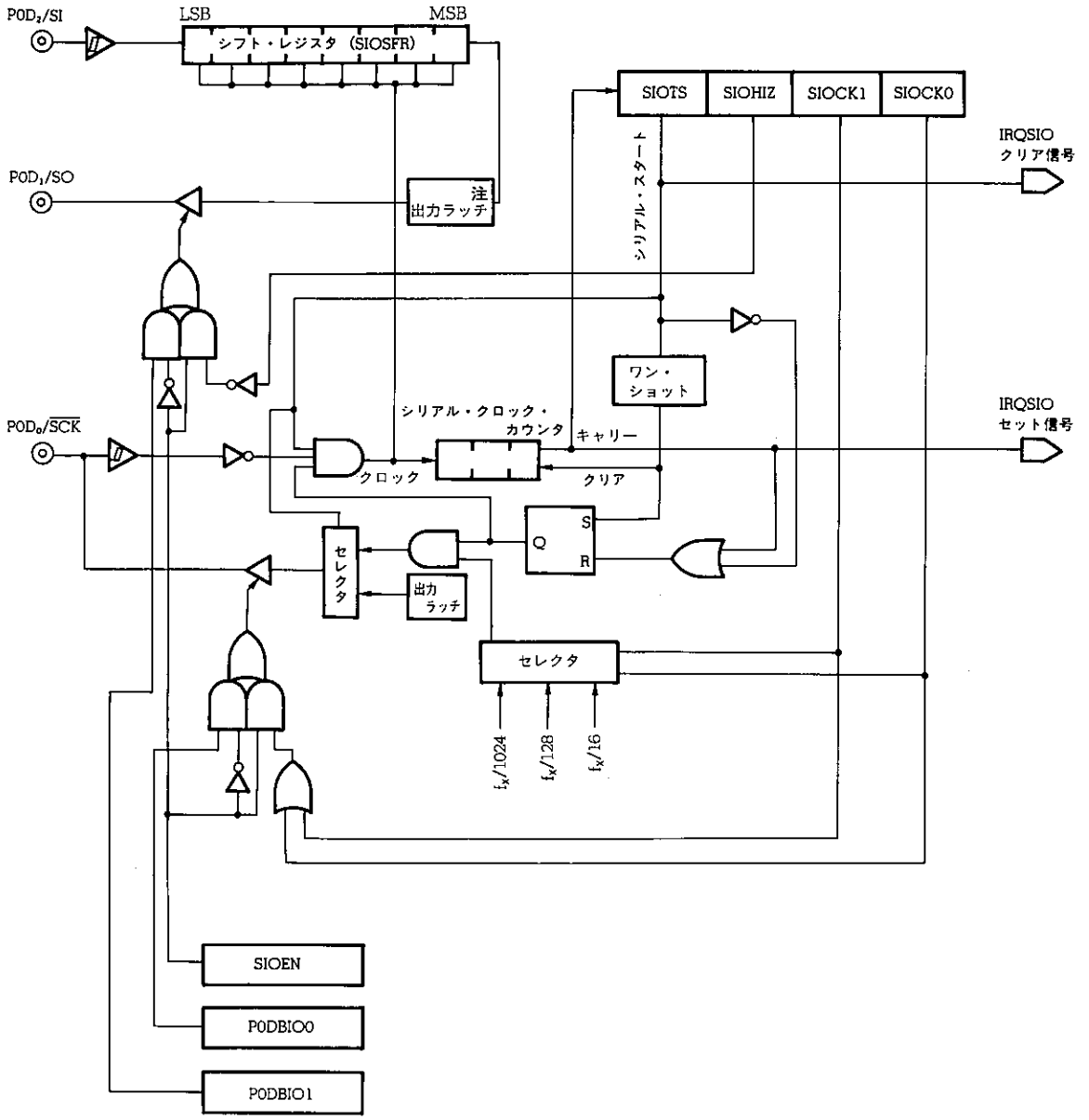
SIOENをセット（1）することにより，ポート0D（$P0D_0/\overline{SCK}$，$P0D_1$/SO，$P0D_2$/SI）の各端子は，シリアル・インタフェース用の端子として機能します。このとき，SIOTSをセット（1）すれば，シリアル・クロックの立ち下がりに同期して動作を開始します。なお，SIOTSをセットすると，IRQSIOは自動的にクリアされます。

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 - 12　シリアル・インタフェースのブロック図



★　注　シフト・レジスタの出力ラッチは，$P0D_1$ の出力ラッチと兼用になっています。したがって，$P0D_1$ に対して出力命令を実行すると，シフト・レジスタの出力ラッチの状態も変化します。

13.3.2　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，$P0D_2$/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	$P0D_2$/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	×	$P0D_2$（入出力）	$P0D_1$（入出力）	汎用ポート・モード

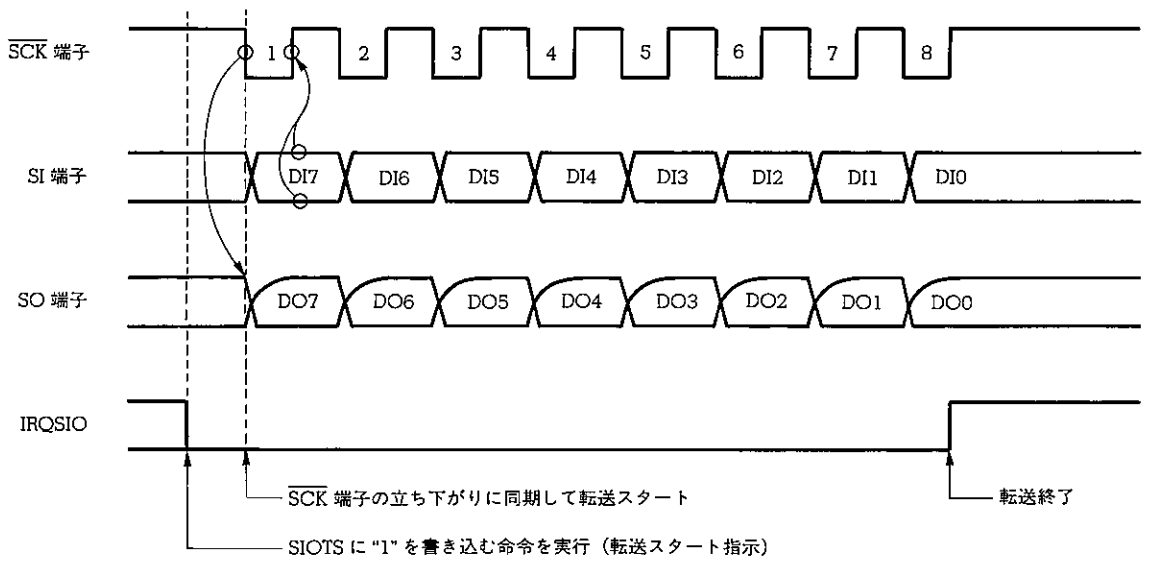
×：Don't care

(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（$\overline{SCK}$）の立ち下がりでシフト・レジスタのMSBがSOラインに出力され，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13-13　8ビット送受信モード（同時送受信）のタイミング



備考　DIn　：シリアル入力データ

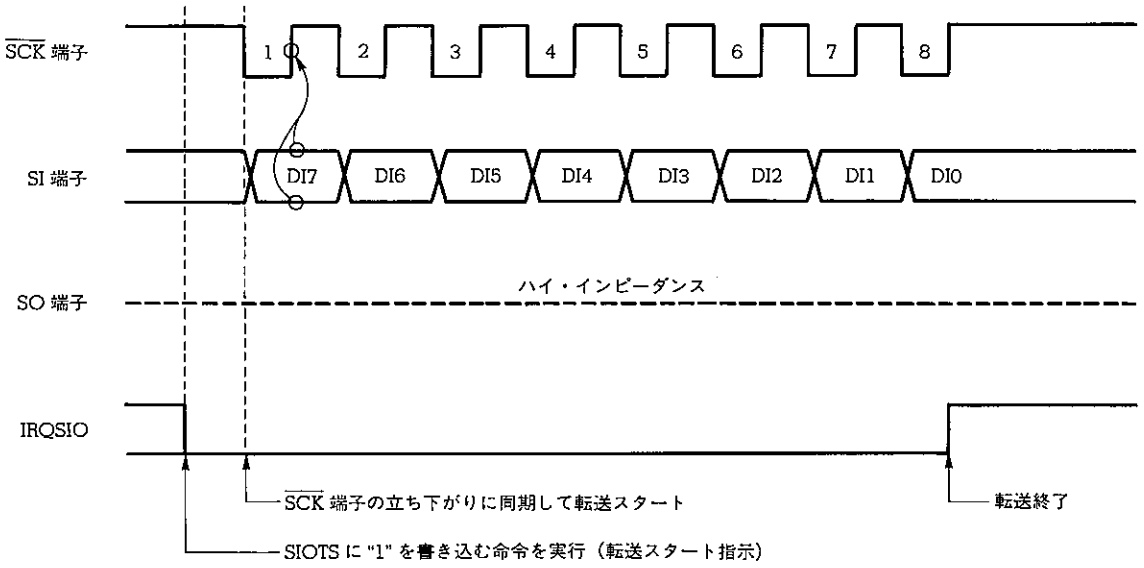
DOn　：シリアル出力データ

(2) 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，$P0D_1$/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，$P0D_1$/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（$P0D_1$）として使用することができます。

図13-14　8ビット受信モードのタイミング



備考　DIn：シリアル入力データ

(3) 動作停止モード

SIOTS (RF：1AH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 13-15　シリアル・インタフェースの制御レジスタ（1/2）

RF：1AH

	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	$f_X/16$
1	0	$f_X/128$
1	1	$f_X/1024$

SIOHIZ	$P0D_1$/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（$P0D_1$ 端子）

SIOTS	シフト・レジスタの動作状態の確認（読み出し時）
0	シフト・レジスタ停止中
1	シフト・レジスタ動作中

SIOTS	シリアル転送の開始と停止（書き込み時）
0	シフト・レジスタの強制的停止。 途中からの再開は不可。
1	シフト・レジスタの動作開始。 ・内部クロック選択時 システム・クロック（f_X）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 - 15　シリアル・インタフェースの制御レジスタ (2/2)

RF : 0AH

	ビット 3	ビット 2	ビット 1	ビット 0
	0	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	シリアル・インタフェースの許可
0	ポート 0D ($P0D_0/\overline{SCK}$, $P0D_1$/SO, $P0D_2$/SI) の各端子はポートとして機能
1	ポート 0D ($P0D_0/\overline{SCK}$, $P0D_1$/SO, $P0D_2$/SI) の各端子はシリアル･インタフェースとして機能

備考　**第12章　ポート**も参照してください。

13.3.3　シフト・レジスタへの値の設定

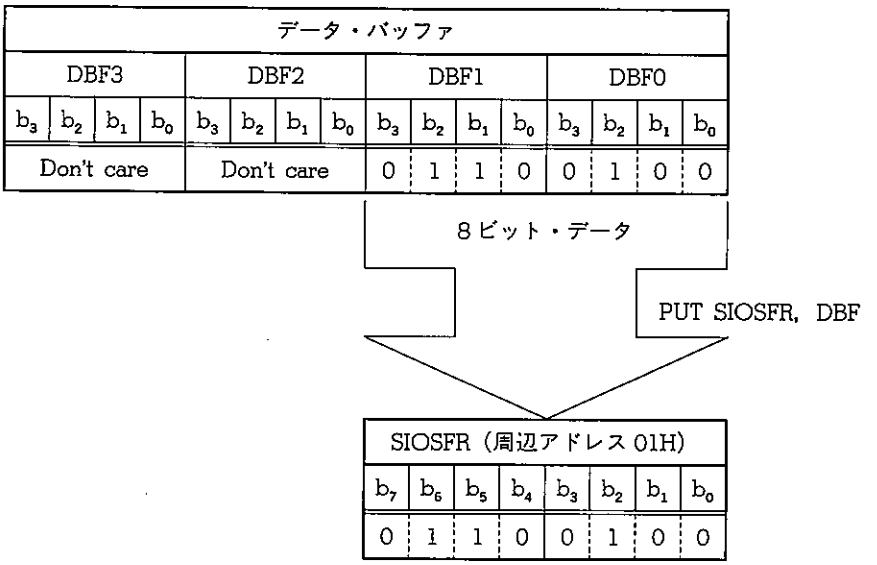
シフト・レジスタに対する値の設定は，PUT 命令により DBF（データ・バッファ）を介して行います。

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。シフト・レジスタに 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，シフト・レジスタに影響を与えません。

図 13 - 16　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。
```

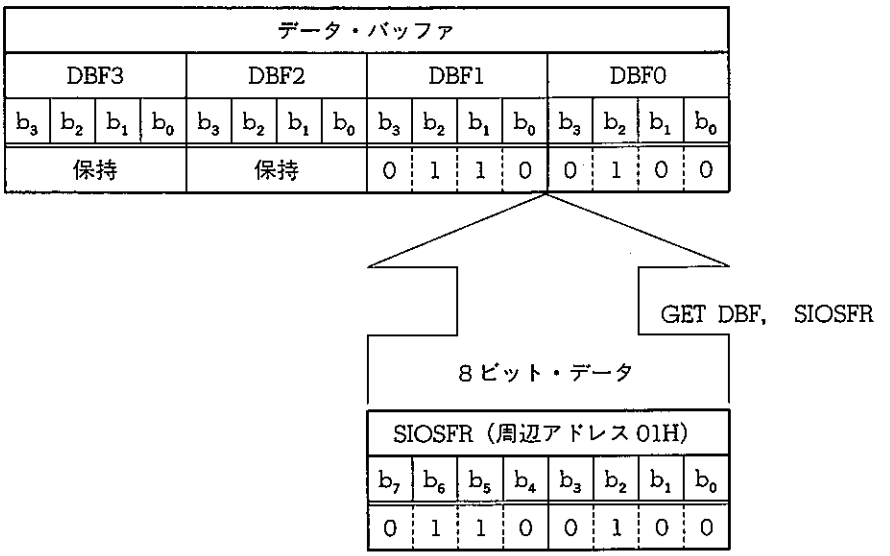


13.3.4　シフト・レジスタの値の読み取り

シフト・レジスタの値の読み取りは，GET 命令により DBF（データ・バッファ）を介して読み取ります。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 13-17　シフト・レジスタの値の読み取り

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



13.3.5　シリアル・インタフェースのプログラム例

(1) 8ビット送受信モード（同期送受信）でのデータの送受信のプログラム例

f_X/128に同期してデータの送受信を行います。シリアル・データの送受信終了判定は割り込み要求フラグ（IRQSIO）のチェックで行います。

例

```
MAIN :
                ⋮
        CALL    SIOJOB
                ⋮
        BR      MAIN

SIOJOB :
        DI                          ; SIOJOB 中の割り込みを禁止
        CLR1    IRQSIO              ; SIO の割り込み要求フラグをクリア
        SET1    SIOEN               ; SIO を許可
        MOV     DBF0, #SIODATL      ; 送信データの設定
        MOV     DBF1, #SIODATH
        PUT     SIOSFR, DBF
        INITFLG SIOTS, NOT SIOHIZ, SIOCK1, NOT SIOCK0
                                    ; シリアル・クロックを "fX/128" に設定，シフ
                                    ; ト・レジスタの動作開始，シリアル・データ
                                    ; 出力
LOOP :
        SKT1    IRQSIO              ; 送受信終了判定
        BR      LOOP                ; 送受信終了待ち
        GET     DBF, SIOSFR         ; 受信データの読み取り
        EI                          ; 割り込みを許可してメイン処理に戻る
        RET                         ;
```

(2) 8ビット受信モードでのデータ受信のプログラム例

外部クロックに同期してデータ受信を行い，割り込み処理を用いて受信データを読み取ります。

例

```
SIODATH MEM     0.50H
SIODATL MEM     0.51H
        ORG     0H
        BR      SIO_ INIT
        ORG     01H
        BR      SIOJOB

SIO_ INIT :
        MOV     SIODATH,  #0H
        MOV     SIODATL,  #0H
        CLR1    IRQSIO              ; SIO の割り込み要求フラグをクリア
        SET1    SIOEN               ; SIO を許可
        INITFLG SIOTS, SIOHIZ, NOT SIOCK1, NOT SIOCK0
                                    ; シリアル・クロックを外部クロックに設定,
                                    ; シリアル・データ受信開始, P0D1/SO 端子を
                                    ; 入力ポート(出力ハイ・インピーダンス)に設定
        EI                          ; すべての割り込みを許可
MAIN :                              ; メイン処理
        CALL    ××JOB
        CALL    ××JOB
          ⋮
        BR      MAIN

SIOJOB :
        GET     DBF, SIOSFR         ; 受信データの読み取り
        MOV     RPH,  #0000B        ; ジェネラル・レジスタを BANK0 のロウ・アド
                                    ; レス 5H に設定,
        MOV     RPL,  #1010B        ; BCD←0
        LD      SIODATH, DBF1       ; 受信データの RAM 上への格納
        LD      SIODATL, DBF0       ;
        EI
        RETI
```

第14章　割り込み機能

μPD17120サブシリーズには，2つの内部割り込み機能と1つ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) 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ソースの選択可能）

注意 1. 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 1 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，シリアル・インタフェースなど）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。

2. 割り込みスタックは 1 レベルのみなので，ハードウエアによる多重割り込みは行えません。1 レベルを越える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。

14.1　割り込み要因とベクタ・アドレス

μPD17120サブシリーズ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	0003H	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	2	0002H	IRQTM RF：3EH， ビット0	IPTM RF：2FH， ビット1	－	内部	
シリアル・インタフェース	3	0001H	IRQSIO RF：3DH， ビット0	IPSIO RF：2FH， ビット2	－	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。

14.2.1　割り込み要求フラグ(IRQ×××)と割り込み許可フラグ(IP×××)

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。

14.2.2　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 をセット（1）します。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
IRQTM	タイマからの一致信号でセット。	IPTM
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（1/4）

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，INT 端子のノイズ除去回路を通した論理状態が“0”
1	PEEK 命令実行時，INT 端子のノイズ除去回路を通した論理状態が“1”

注　INT フラグはラッチは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れば，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（2/4）

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

IRQ	INT 端子割り込み要求（読み出し時）
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

IRQ	INT 端子割り込み要求（書き込み時）
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	IRQTM
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

IRQTM	TM 割り込み要求（読み出し時）
0	タイマからの割り込み要求なし，または，タイマの割り込み処理中
1	タイマのカウント・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，またはタイマの割り込み要求を保留中

IRQTM	TM 割り込み要求（書き込み時）
0	タイマからの割り込み要求の強制的解除
1	タイマからの割り込み要求の強制的発生

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

図 14 - 1　割り込み制御用レジスタ（3/4）

RF：3DH

	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

IRQSIO	SIO 割り込み要求（読み出し時）
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

IRQSIO	SIO 割り込み要求（書き込み時）
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（4/4）

RF：2FH

	ビット 3	ビット 2	ビット 1	ビット 0
	0	IPSIO	IPTM	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

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

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。

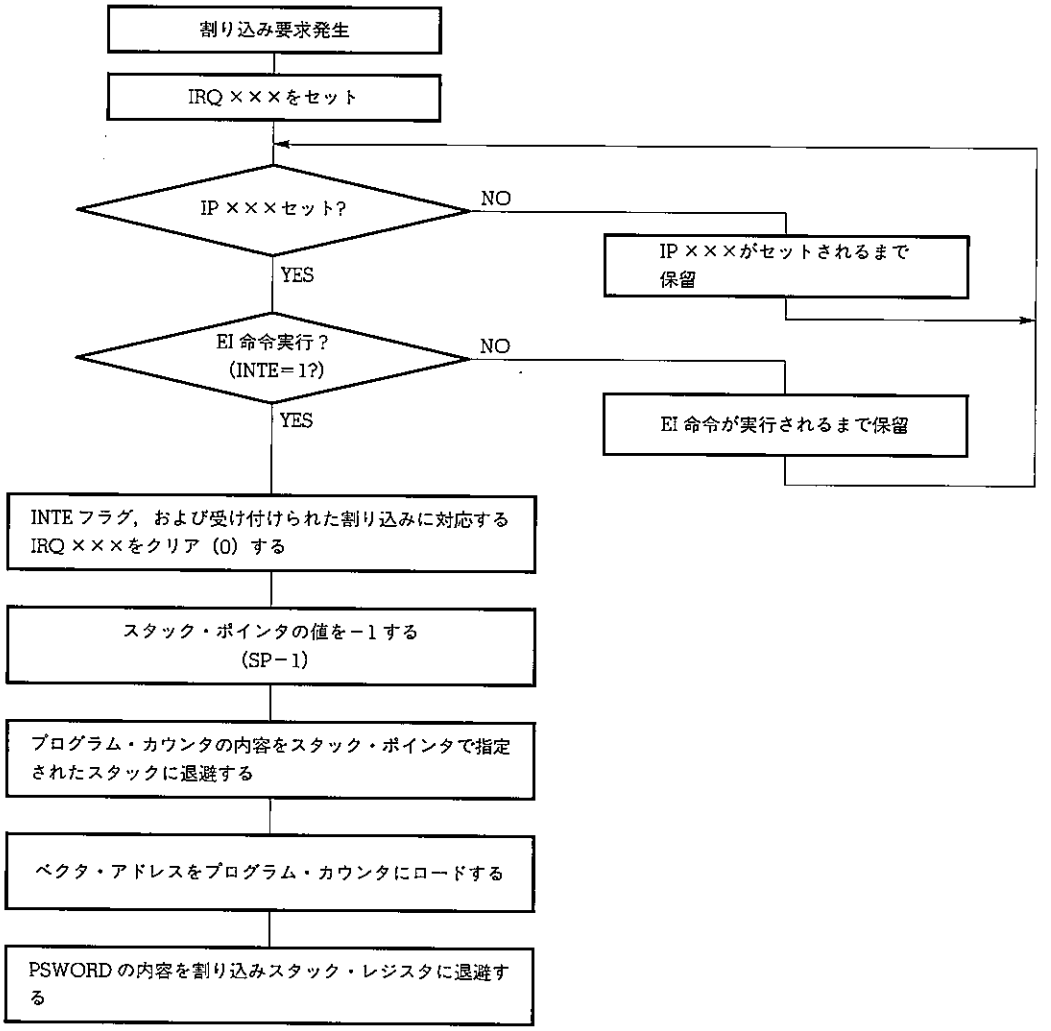
割り込み処理を開始すると，プログラムの戻り番地を記憶するためアドレス・スタック・レジスタを１レベル消費し，さらにシステム・レジスタ中の PSWORD を退避するため割り込みスタック・レジスタを１レベル消費します。

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。

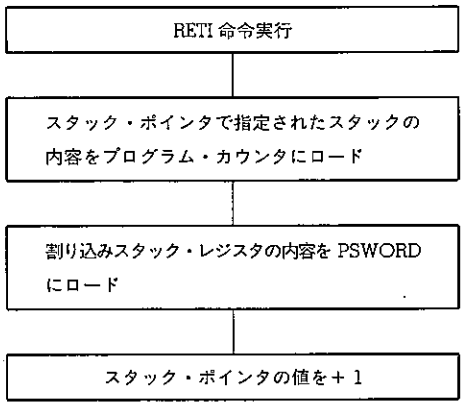
図 14－2　割り込み処理手順



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

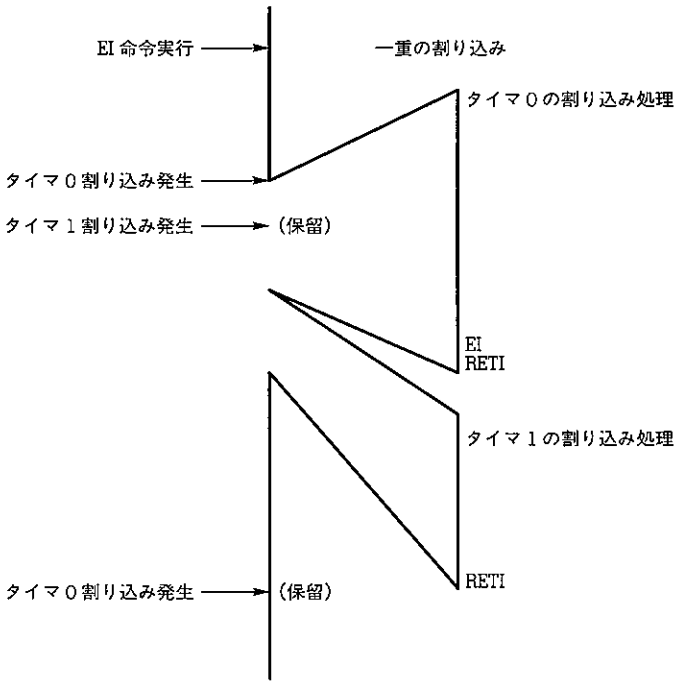


注意 1．RETI 命令では INTE フラグはセットされません。

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



14.3.3　割り込み受け付けタイミング

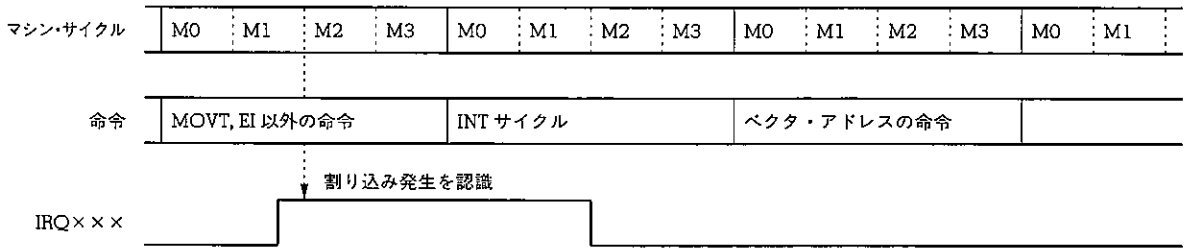
割り込み受け付けタイミング・チャートを図14-4に示します。

μPD17120サブシリーズは，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クロック単位にM0-M3に細分化されます。

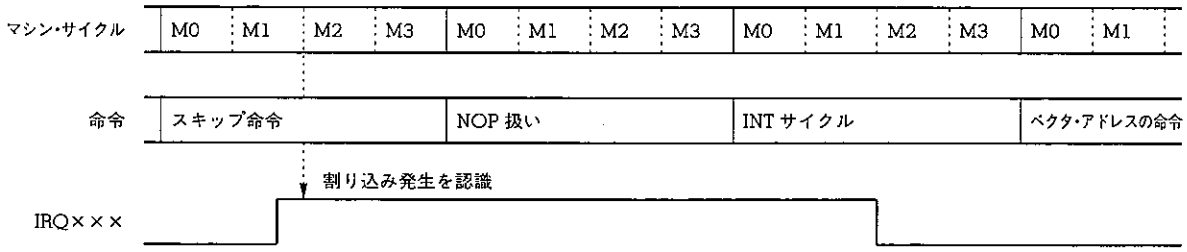
プログラムが割り込みの発生を認識するタイミングは，M2の前のエッジです。

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

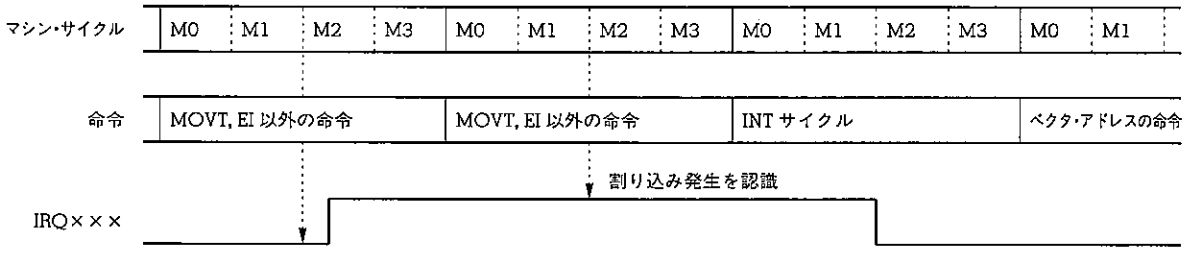
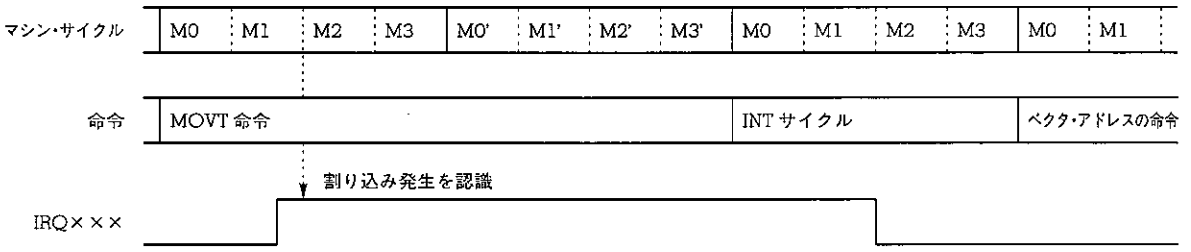
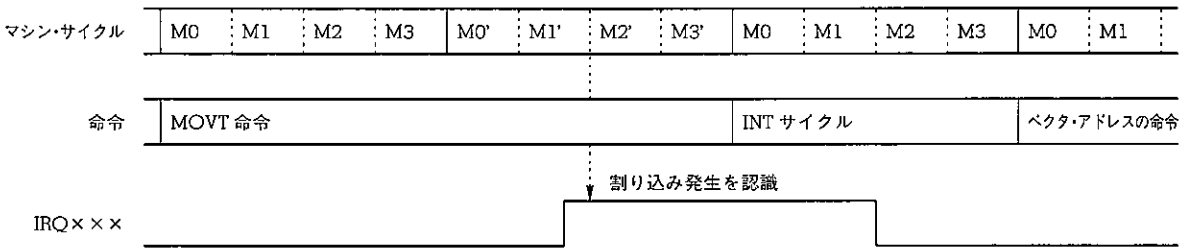


図14－4　割り込み受け付けタイミング・チャート（INTE＝1，IP×××＝1のとき）（2/3）

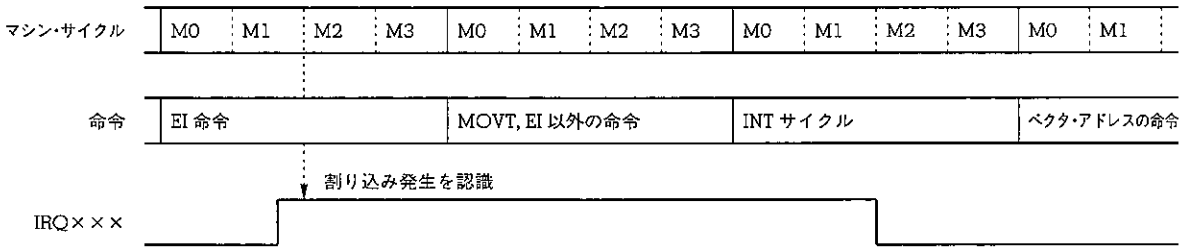
④　MOVT命令のM2以前に割り込みが発生した場合



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

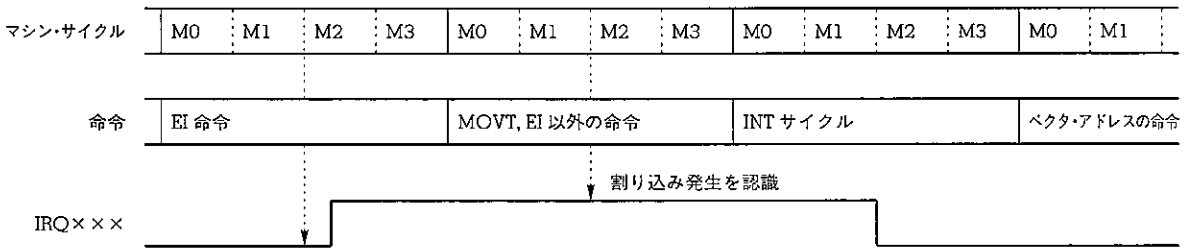
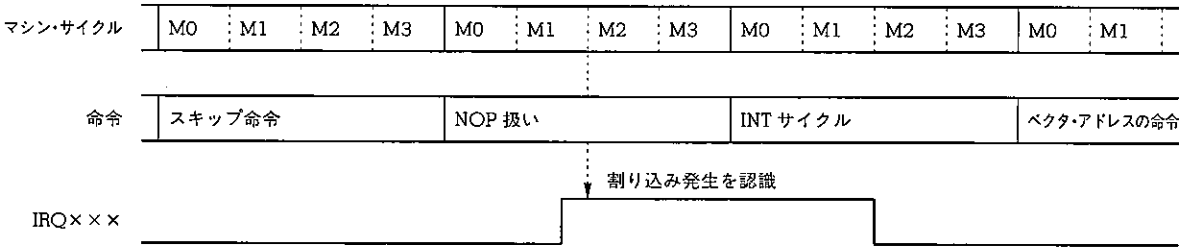


図14－4　割り込み受け付けタイミング・チャート（INTE＝1，IP××＝1 のとき）(3/3)

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。

14.4　割り込みのプログラム例

●外部割り込み（INT 端子）のノイズ除去対策のプログラム例

ここに示す例は INT 端子をキー入力などに割り当てた場合を想定しています。

キー入力処理などのスイッチ類のデータをマイコンに取り込む場合は，キーまたはスイッチを押してから入力される電圧レベルが安定するまである程度の時間を要します。したがって，キーなどから発生するノイズを除去する対策をソフトウエアで行う必要があります。

以下のプログラム例では，外部割り込み発生後，100 μs ごとに 2 回 INT 端子のレベルに変化がないことを確認してから INT 端子からの信号を有効にしています。

例

```
WAITCNT   MEM      0.00H          ; ウエイト処理のカウンタ
KEYON     FLG      0.01H.3        ; キー ON が確定すれば KEYON＝1
SECOND    FLG      0.01H.0        ; 2 回目のキー・チェック中であることを示すフラ
                                  ; グ
          ORG      0H
          BR       JOB_INIT
          ORG      3H
          BR       INT_JOB
          ⋮
JOB_INIT :
          MOV      WAITCNT, #0    ; RAM および RAM 上のフラグをクリア
          CLR2     KEYON, SECOND ;
          INITFLG  NOT IEGMD1, IEGMD0
                                  ; INT 端子からの割り込みは立ち上がりエッジ有効
          CLR1     IRQ
          SET1     IP
          EI
          ⋮
MAIN :
          CALL     ××JOB         ; 任意の処理
          CALL     ××JOB         ; 任意の処理
          ⋮
          BR       MAIN
```

```
INT_JOB :
            NOP                     ; 8 MHz 時に100 μs のウエイトを行うループ
            NOP                     ; 2 μs (1 命令) ×5 命令×10 回 (WAIT 時のカウ
                                    ; ント値)
            ADD     WAITCNT, #01    ;
            SKE     WAITCNT, #0AH   ;
            BR      INT_JOB         ;
            SKF1    INT             ; INT 端子のレベル確認
            BR      KEY_OFF         ; INT 端子がハイ・レベルなら割り込み無効でメイ
                                    ; ン処理に戻る
            SKF1    SECOND          ; ウエイトは初めてか？
            BR      WAIT_END        ; 初めてなら SECOND をセットしてもう一度ウエ
                                    ; イト，2回目ならウエイト処理終了
            SET1    SECOND          ;
            MOV     WAITCNT, #0
            BR      INT_JOB
WAIT_END :
            SET1    KEYON           ; キー入力ありと判定
            BR      INT_JOB_END
KEY_OFF :
            CLR1    SECOND          ; SECOND ← 0
INT_JOB_END :
            MOV     WAITCNT, #0
            EI
            RETI
```

〔メ モ〕

第15章　スタンバイ機能 ★

15.1　スタンバイ機能の概要

μPD17120サブシリーズは，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
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1]	
	TM	・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
	SIO	・シフト・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1]	・動作可能
	コンパレータ[注2]	・動作停止[注1]	・動作中断（再開後の結果は“不定”）
	INT	・動作可能	

注 1．STOP 0000B を実行した場合には命令実行点で，兼用端子機能で使用している場合も含めて，端子の状態は入力ポート・モードになります。

2．μPD17132，17133，17P132，17P133のみ。

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 IRQTM による強制解除を許可する。[注3, 4]
b_0	"0 固定"

注 1. HALT 0000B のときは，リセット（$\overline{\text{RESET}}$ 入力，パワーオン/パワーダウン・リセット）だけが有効です。

2. IP×××＝1 である必要があります。

3. IPTM の状態によらず，HALT モードが解除されます。

4. IRQ×××＝1 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

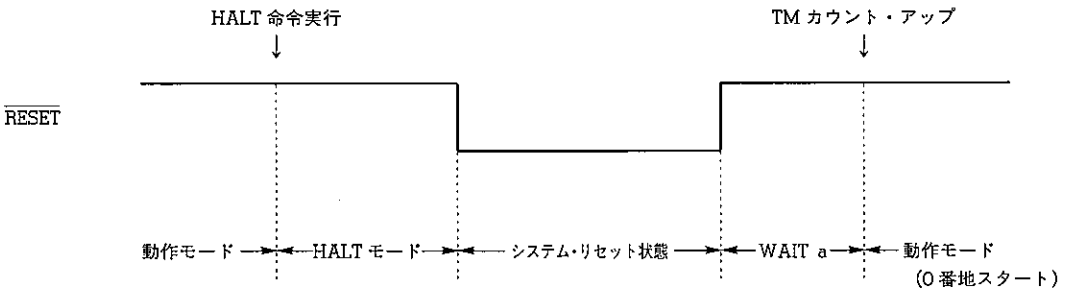
解除条件	解除後のスタート番地
リセット[注1]	0 番地
IRQ×××[注2]	DI の場合，HALT 命令の次の番地
	EI の場合，割り込みベクタ （複数の IRQ××× がセットされている場合には，優先順位の高い割り込みベクタ）

注 1. リセットは，$\overline{\text{RESET}}$ 入力，パワーオン/パワーダウン・リセットが有効です。

2. IRQTM 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。

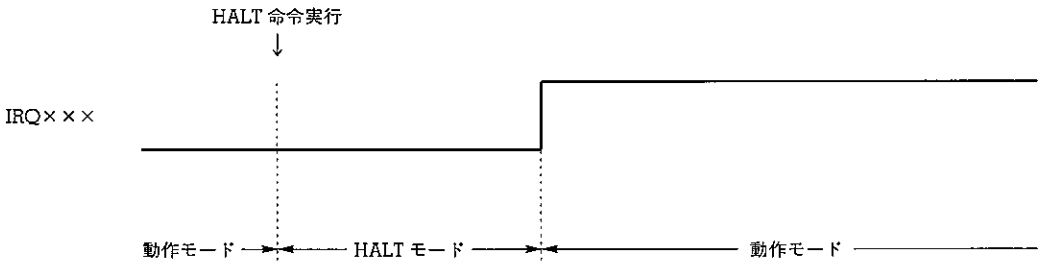
図15－1　HALT モードの解除

(a) $\overline{\text{RESET}}$ 入力による HALT 解除

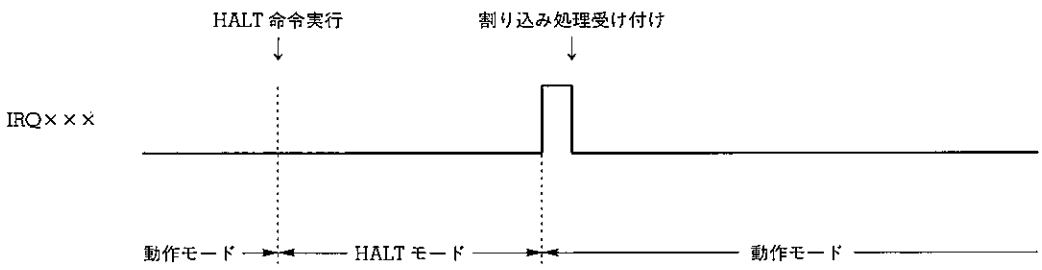


WAIT a：TM が256分周のクロックを256カウントするまでの待ち時間です。
$256 \times 256/f_X$（約32 ms，f_X＝2 MHz 時）

(b) IRQ×××による HALT 解除（DI の場合）



(c) IRQ×××による HALT 解除（EI の場合）



15.2.3　HALT の設定条件

(1) IRQTM による強制解除の場合

●タイマは動作可能状態（TMEN＝1）

●タイマの割り込み要求フラグをクリア（IRQTM＝0）

(2) 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設定。

タイマ	動作可能状態（TMEN＝1）
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

```
        ⋮
(IRQ×××のセット)
        ⋮

CLR1    IRQ×××
NOP             ; NOP 命令を HALT 命令の直前に記述
                ; (IRQ××× 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

```
        ⋮
(IRQ×××のセット)
        ⋮

CLR1    IRQ×××  ; IRQ××× 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2,3]
b_2	“0 固定”
b_1	“0 固定”
b_0	“0 固定”

注 1. STOP 0000B のときは，リセット（$\overline{\text{RESET}}$ 入力，パワーオン/パワーダウン・リセット）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，IRQTM による解除はできません。

3. IRQ×××＝1 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

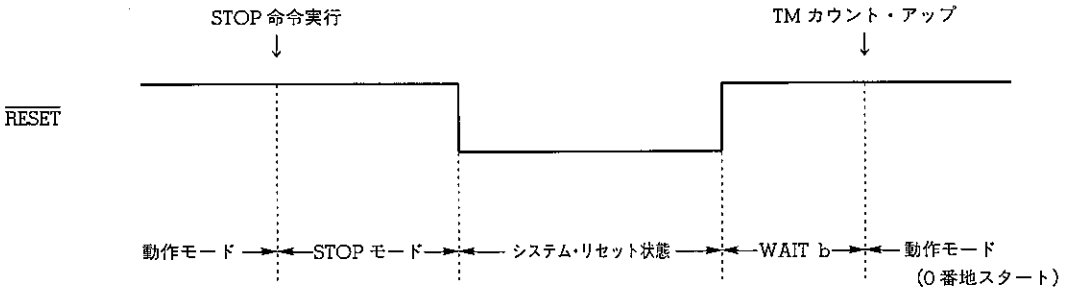
解除条件	解除後のスタート番地
リセット[注1]	0 番地
IRQ×××[注2]	DI の場合，STOP 命令の次の番地
	EI の場合，割り込みベクタ （複数の IRQ××× がセットされている場合には，優先順位の高い割り込みベクタ）

注 1. リセットは，$\overline{\text{RESET}}$ 入力，パワーオン/パワーダウン・リセットが有効です。

2. IP×××＝1 である必要があります。また，IRQTM による解除はできません。

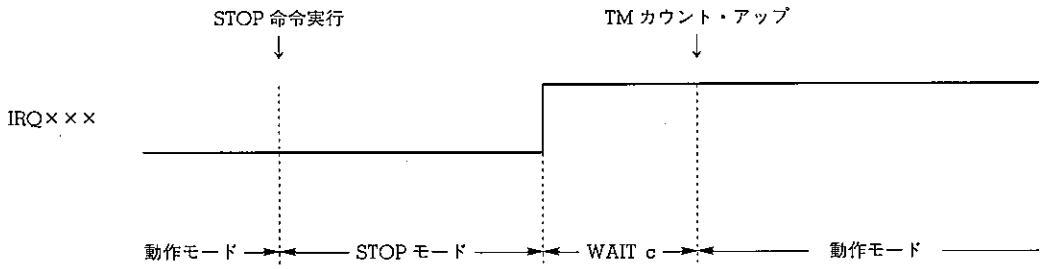
図15-2　STOP モードの解除

(a) $\overline{\text{RESET}}$ 入力による STOP 解除



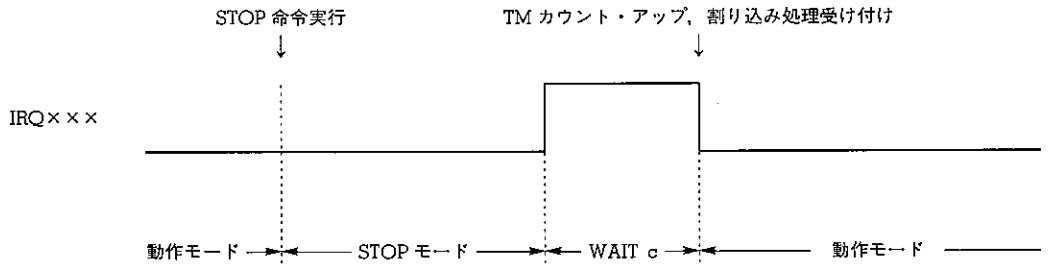
WAIT b：TM が256分周のクロックを256カウントするまでの待ち時間です。
$256\times256/f_x+\alpha$（約32 ms＋α，f_x＝2 MHz 時）
α　：発振成長時間（発振子により異なります。）

(b) IRQ×××による STOP 解除（DI の場合）



WAIT c：TM 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
$(n+1)\times m/f_x+\alpha$（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（EI の場合）



WAIT c：TM 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
$(n+1)\times m/f_x+\alpha$（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

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（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 $\overline{SCK}$ 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（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）。

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（図 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

μPD17120サブシリーズのリセットには，次の3種類があります。

① $\overline{\text{RESET}}$ 入力によるリセット

② 電源投入時および電源電圧降下時にリセットをかけるパワーオン/パワーダウン・リセット機能

③ 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機能

16.1　リセット機能概要

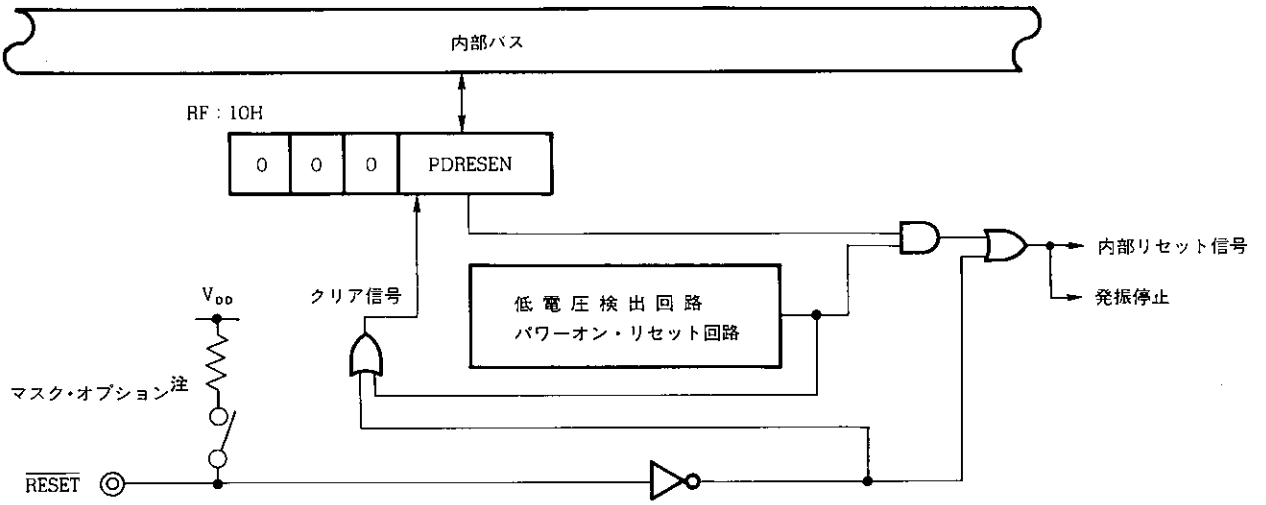
リセット機能は，デバイス動作の初期化を行うために使用します。なお，リセットの種類により，初期化される状態が異なります。

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 • 動作中の内蔵パワーオン/パワーダウン・リセット	• スタンバイ・モード中の $\overline{\text{RESET}}$ 入力 • スタンバイ・モード中の内蔵パワーオン/パワーダウン・リセット	• 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	入出力モード	入力	入力	入力
	出力ラッチ	0	0	不定
汎用データ・メモリ	DBF以外	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以外	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, TMEN＝1, CMPVREF3＝1[注], CMPRSLT＝1[注], INTはそのときのINT端子の状態，それ以外はすべて0。 **19.2　予約シンボル**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	カウント・レジスタ	00H	00H	不定
	モジュロ・レジスタ	FFH	FFH	FFH
シリアル・インタフェースのシフト・レジスタ（SIOSFR）		不定	リセット直前の状態を保持	不定

注　μPD17132, 17133, 17P132, 17P133のみ。

図16-1　リセット・ブロックの構成



注　μPD17P132, 17P133にはマスク・オプションによるプルアップ抵抗がなく，常にオープンになっています。

16.2　リセット動作

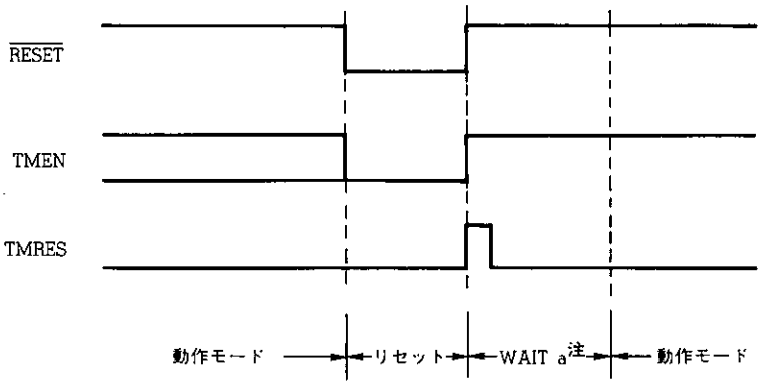
$\overline{\text{RESET}}$入力によりリセットをかけたときの動作を下記に示します。

$\overline{\text{RESET}}$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。

パワーオン・リセット機能を用いたリセットの場合も，図16-2のようなリセット信号を内部で生成し，$\overline{\text{RESET}}$入力により外部からリセットをかけたときと同様の動作をします。

なお，アドレス・スタック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。タイマによりシステム・クロックを256×256カウント（約8 ms，f_X=8 MHz時 / 約32 ms，f_{CC}=2 MHz時）すると動作モードとなります。

16.3　パワーオン/パワーダウン・リセット機能

μPD17120サブシリーズには，電源の立ち上がりおよび電源電圧の低下を監視し，マイコン内部にリセットをかけるパワーオン/パワーダウン・リセット機能があり，マイコンの誤動作防止に威力を発揮します。

この機能は，通常のマイコン・ロジック部とは動作電源電圧範囲が異なる電源監視回路と，リセットがかかると発振を停止し，マイコンを一時動作停止状態にする発振回路部により構成されています。次にパワーオン/パワーダウン・リセット機能が有効に働く条件と機能について説明します。

注意　高い信頼性が要求される応用回路を設計する際には，リセットが内蔵のパワーオン/パワーダウン・リセット機能だけに依存した設計をしないでください。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。 ★

16.3.1　パワーオン・リセット機能が有効に働く条件

パワーオン・リセット機能は，実際に使用する環境においてはパワーダウン・リセット機能とともに使用したときに初めて有効になる機能です。

パワーオン・リセット機能は，次の条件において有効です。

① 通常動作時（スタンバイ時も含む）において，電源電圧範囲が4.5～5.5 V であること。
② システム・クロック発振周波数が400 kHz～4 MHz 以内であること[注]。
③ 通常動作時（スタンバイ時も含む）において，パワーダウン・リセット機能を使用すること。
④ 電源が0 V から立ち上がること。
⑤ 0～2.7 V までの電源の立ち上がり時間が，μPD17120サブシリーズのタイマで生成される発振安定待ち時間（システム・クロック256×256カウント：約16 ms，f_X＝4 MHz 時 / 約32 ms，f_{CC}＝2 MHz 時）以内であること。

注　μPD17121，17133，17P133のみ

注意 1．上記条件が満たされない場合は，内蔵されたパワーオン・リセット回路が有効に動作しません。このため，外付けにリセット回路が必要となります。

2．スタンバイ時，パワーダウン・リセット機能が働いた場合でも V_{DD}＝2.7 V までは汎用データ・メモリ（DBF は除く）はデータを保持しています。なお外乱などにより，データが変化した場合のデータ保持については保証されていません。

16.3.2　パワーオン・リセット機能と動作

パワーオン・リセット機能は，内蔵されているハードウエアにより，ソフトウエアに関係なく電源を監視し，電源立ち上がり時に内部システムにリセットをかける機能です。

このパワーオン・リセット回路は，μPD17120サブシリーズのほかの内部回路より低電圧で動作し，発振の有無に関係なくマイコン内部を初期化します。そして，リセットが解除されると，発振子からの発振パルスをタイマによりカウントし，発振安定待ちを行います。この発振安定待ちは，発振子の発振安定待ちはもとより，マイコンに印加される電源電圧が，マイコンの動作保証電圧範囲内(V_{DD}=2.7～5.5 V，400 kHz～4 MHz[注]）になるのを待つことにも使用されています。

この発振安定待ちから解除されると，マイコンは動作状態となります。その動作例について図16－3に示します。

注　μPD17121，17133，17P133のみ

パワーオン・リセットの機能

① V_{DD}端子に印加されている電圧レベルを常に監視。

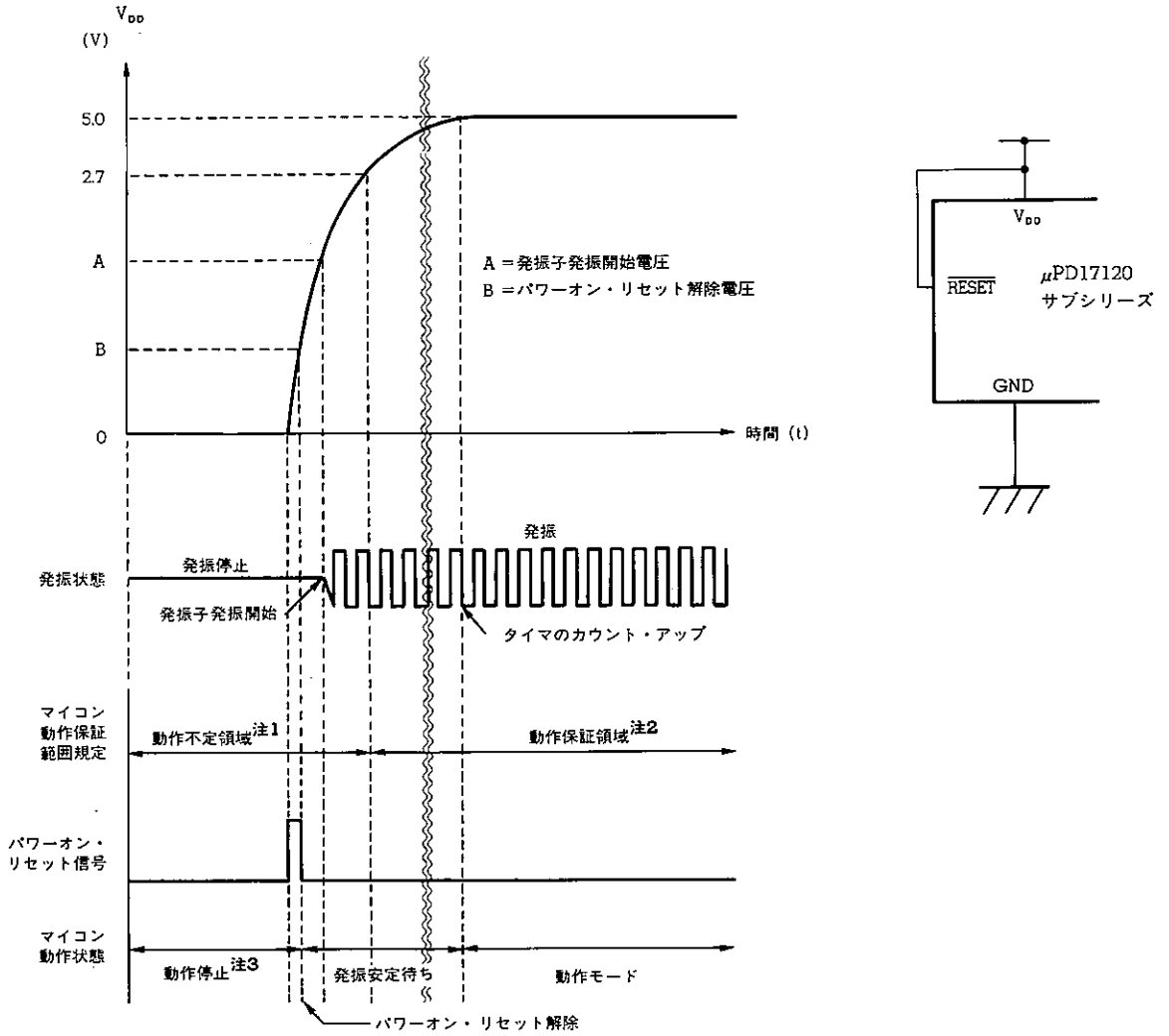
② 電源の立ち上がりにおいて，パワーオン・リセット解除電圧（V_{DD}=1.5 V TYP.）までは，発振の有無に関係なくマイコン内部にリセットをかける[注]。

③ リセットがかかっている間は発振を停止。

④ リセットが解除されると，タイマにより発振安定待ちおよび電源電圧がV_{DD}=2.7 V以上になるのを待つ。

★　注　マイコン内部にリセットがかかるのは，内部回路が動作できる（内部リセット信号を受け付けられる）電圧に電源電圧が達した時点からです。

図 16-3　内蔵パワーオン・リセット動作例



注 1. 動作不定領域とは，μPD17120サブシリーズに規定されている動作が保証されていない領域のことです。ただし，この領域においてもパワーオン・リセット機能は動作します。

2. 動作保証領域とは，μPD17120サブシリーズに規定されている動作のすべてが保証される領域のことです。

3. マイコンの動作状態において動作停止とは，マイコンのすべての機能が止まっている状態のことです。

16.3.3　パワーダウン・リセット機能が使用できる条件

パワーダウン・リセット機能は，ソフトウエアによりその使用の有無を選択することができます。使用できる条件は以下のとおりです。

○　通常動作時（スタンバイ時も含む）の電源電圧範囲が4.5～5.5Vであること。
○　システム・クロック発振周波数が400 kHz～4 MHzであること[注]。

注　μPD17121，17133，17P133のみ

注意　2.7～4.5Vの範囲で通常動作を行う場合には，内蔵されたパワーダウン・リセット機能を使用せず，リセット回路を外付けしてください。2.7～4.5Vの動作電圧範囲においてパワーダウン・リセット機能を使用すると，リセットが解除されなくなる可能性があります。

16.3.4　パワーダウン・リセット機能と動作

パワーダウン・リセット機能は，ソフトウエアによりパワーダウン・リセット・イネーブル・フラグ（PDRESEN）をセットすると機能します。

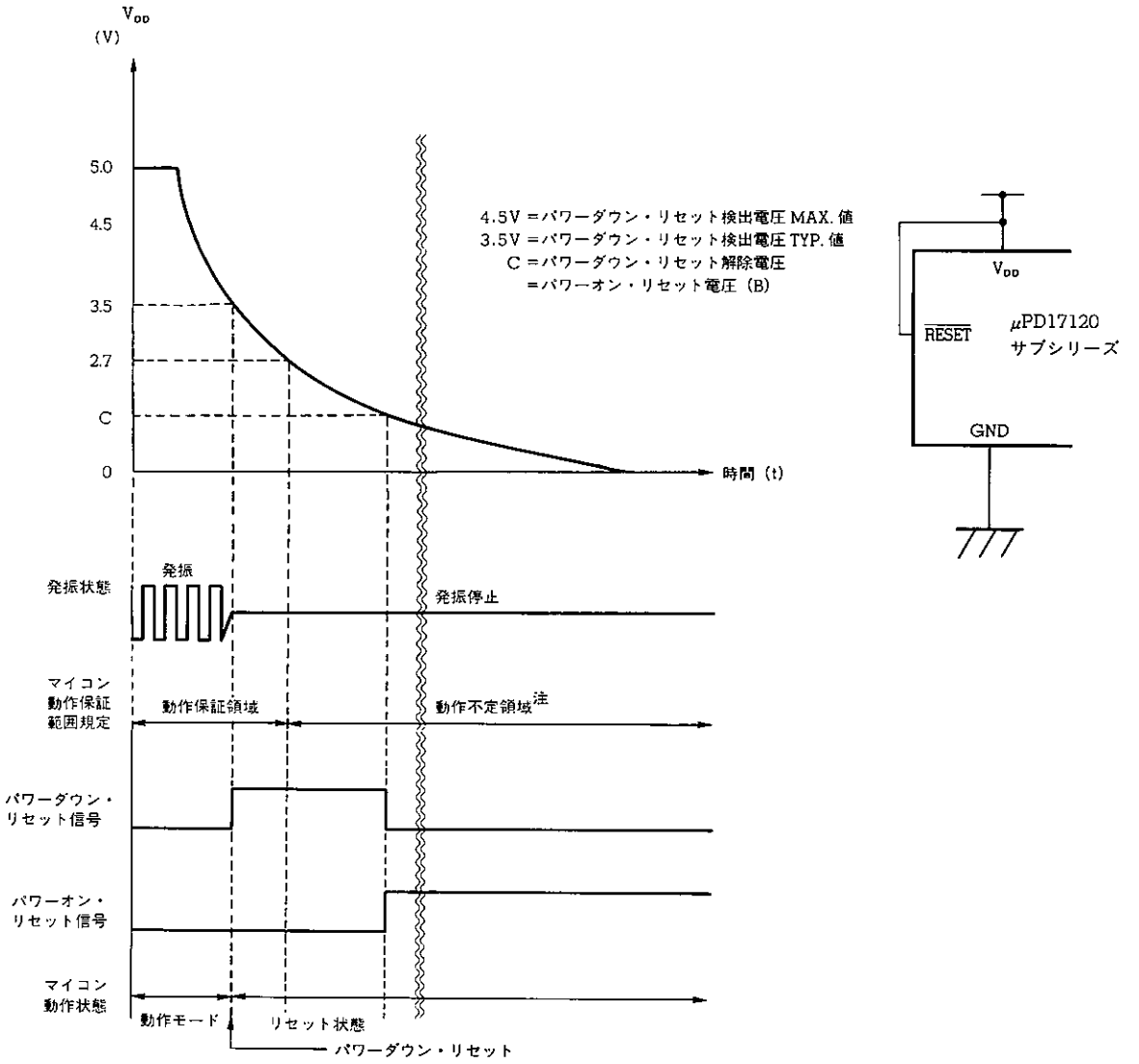
この機能が動作している間に，電源電圧の低下を検出するとマイコン内部に対しリセット信号を発生し，マイコン内部を初期化します。また，リセットがかかっている間は発振が停止しているため，マイコンが電源電圧の乱れにより暴走することを防ぐことができます。電源電圧が復帰し，パワーダウン・リセットが解除された場合は，タイマによる発振安定待ち状態を介したのち，通常の動作状態（0番地スタート）となります。

図16－4に内蔵パワーダウン・リセットの動作例，図16－5にはパワーダウン→電源復帰時のリセット動作例について示します。

パワーダウン・リセット機能

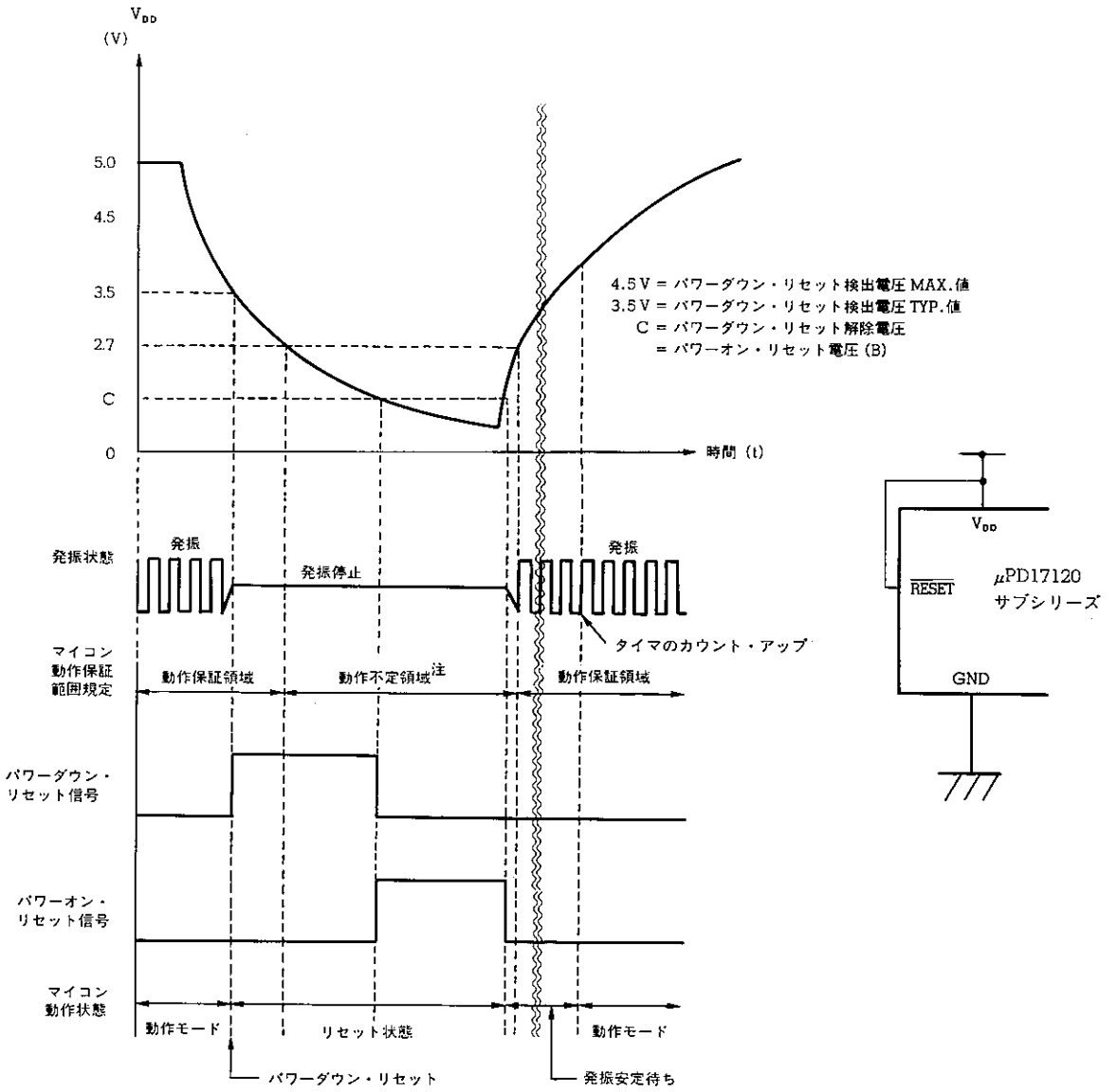
① V_{DD}端子に印加されている電圧レベルを常に監視。
② 電源電圧の低下を検出すると，リセット信号をマイコン内部に対し発生。電源電圧が復帰するか，またはマイコンのすべての機能が停止するまでリセット信号を発生し続ける。
③ リセットがかかっている間は発振を停止（暴走防止対策）。
パワーダウン・リセット機能が停止する前に電源が復帰した場合は，低電圧検出レベル（3.5V TYP.，4.5V MAX.）以上になったとき，タイマによる発振安定待ちを介したのち，通常の動作モードに移る。
④ 0Vから電源電圧が復帰した場合は，その機能をパワーオン・リセット機能に譲る。
⑤ パワーダウン・リセット機能が停止したのち，電源電圧が0Vに達する前に復帰した場合は，タイマにより発振安定待ちおよび電源電圧がV_{DD}=2.7V以上になるのを待ち，通常の動作モードに移る。

図 16－4　内蔵パワーダウン・リセット動作例



注　動作不定領域とは，μPD17120サブシリーズに規定されている動作が保証されていない領域のことです。ただし，この領域においても，パワーダウン・リセット機能は動作し，マイコン内部のそのほかの機能がすべて停止するまでリセットを発生し続けます。

図 16-5　パワーダウン→電源復帰時のリセット動作例



注　動作不定領域とは，μPD17120サブシリーズに規定されている動作が保証されていない領域のことです。ただし，この領域においても，パワーダウン・リセット機能は動作し，マイコン内部のそのほかの機能がすべて停止するまでリセットを発生し続けます。

第17章　ワン・タイムPROMの書き込みとベリファイ

μPD17P132, 17P133に内蔵されているプログラム・メモリは1024×16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17-1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　INT/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INT/V_{PP}端子にV_{DD}＋0.3V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17-1　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。＋12.5Vを印加します。
V_{DD}	正電源です。＋6V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。

17.1　マスクROM製品とワン・タイムPROM製品との違い

μPD17P132, 17P133は，マスクROM内蔵製品μPD17132, 17133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17-2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，RAM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（コンパレータは除く）は同じです。したがって，システム開発時にμPD17P132はμPD17120およびμPD17132のプログラム評価用として，μPD17P133はμPD17121およびμPD17133のプログラム評価用として使用できます。

表 17-2　マスク ROM 製品とワン・タイム PROM 製品との違い

項　　目	μPD17120	μPD17132	μPD17P132	μPD17121	μPD17133	μPD17P133
ROM	マスク ROM		ワン・タイム PROM	マスク ROM		ワン・タイム PROM
	768×16ビット (0000H-02FFH)	1024×16ビット (0000H-03FFH)		768×16ビット (0000H-02FFH)	1024×16ビット (0000H-03FFH)	
RAM	64×4 ビット	111×4ビット		64×4 ビット	111×4ビット	
POD, POE 各端子および $\overline{\text{RESET}}$ 端子のプルアップ抵抗	マスク・オプション		な　し	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	な　し		あ　り
動作周波数範囲	f_{CC} = 400 kHz ～2.4 MHz			f_X = 400 kHz ～4 MHz　(V_{DD} = 2.7～5.5 V) f_X = 400 kHz ～8 MHz　(V_{DD} = 4.5～5.5 V)		
コンパレータ	な　し	あ　り		な　し	あ　り	

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。

17.2　プログラム・メモリ書き込み/ベリファイ時の動作モード

μPD17P132, 17P133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，表 17-1 に示す以外の端子は，すべて個別にプルダウン抵抗を介して GND に接続してください（ただし，X_{OUT} 端子はオープンにしてください）。

詳しくは **1.4　(2) プログラム・メモリ書き込み/ベリファイ・モード**を参照してください。

表 17-3　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）

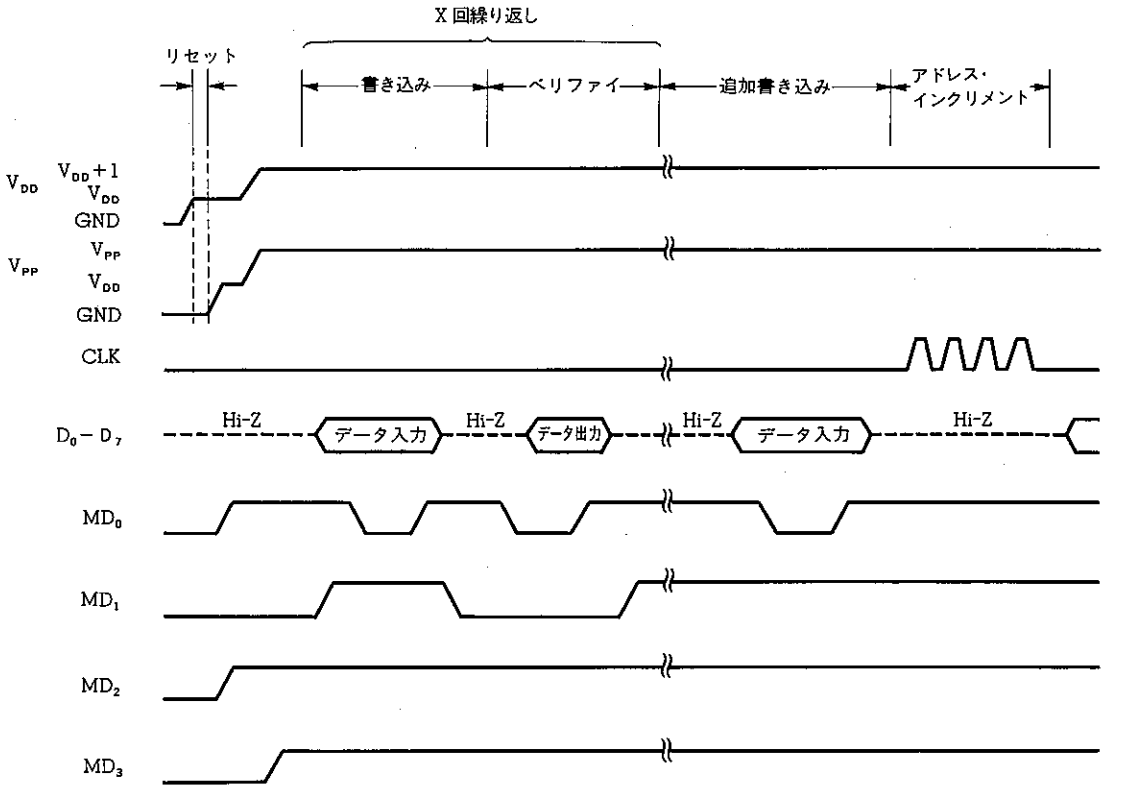
17.3　プログラム・メモリ書き込み手順

プログラム・メモリ書き込みの手順は次のようになっており，高速書き込みが可能です。

(1) 使用しない端子を抵抗を介してGNDにプルダウン（X_{OUT}端子はオープン）。CLK端子はロウ・レベル。
(2) V_{DD}端子に5V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5V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12.5V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5Vに変更。
(16) 電源オフ。

この（2)-(12）の手順を図17-1に示します。

図17-1　プログラム・メモリ書き込み手順



17.4　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プルダウン（X_{OUT}端子はオープン）。CLK端子はロウ・レベル。

(2) V_{DD}端子に5V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5V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}端子に6V，V_{PP}に12.5V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。

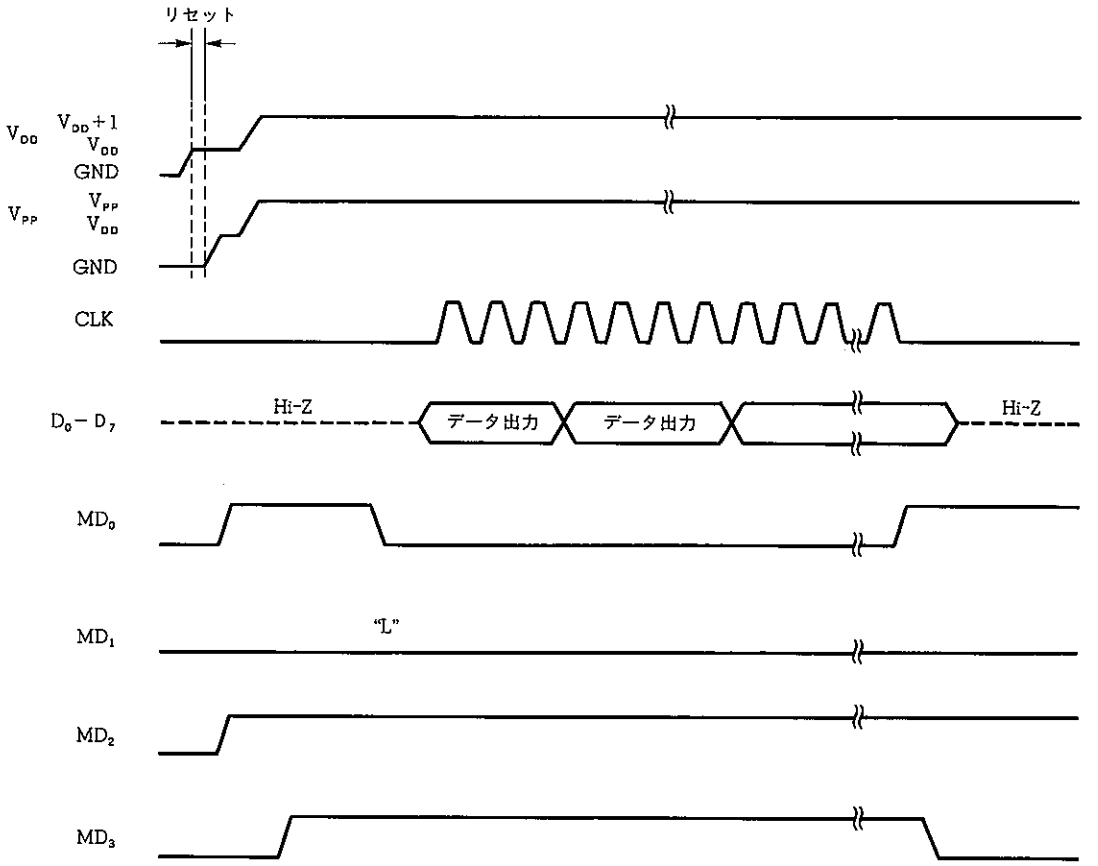
(9) プログラム・メモリ・アドレスの0クリア・モード。

(10) V_{DD}，V_{PP}端子の電圧を5V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 17 - 2 に示します。

図 17 - 2　プログラム・メモリ読み出し手順



〔メ モ〕

第18章　命令セット

18.1　命令セット概要

$b_{14}-b_{11}$ \ b_{15}		0	1
BIN	HEX		
0 0 0 0	0	ADD r, m	ADD m, #n4
0 0 0 1	1	SUB r, m	SUB m, #n4
0 0 1 0	2	ADDC r, m	ADDC m, #n4
0 0 1 1	3	SUBC r, m	SUBC m, #n4
0 1 0 0	4	AND r, m	AND m, #n4
0 1 0 1	5	XOR r, m	XOR m, #n4
0 1 1 0	6	OR r, m	OR m, #n4
0 1 1 1	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	
1 0 0 0	8	LD r, m	ST m, r
1 0 0 1	9	SKE m, #n4	SKGE m, #n4
1 0 1 0	A	MOV @r, m	MOV m, @r
1 0 1 1	B	SKNE m, #n4	SKLT m, #n4
1 1 0 0	C	BR addr	CALL addr
1 1 0 1	D		MOV m, #n4
1 1 1 0	E		SKT m, #n
1 1 1 1	F		SKF m, #n

18.2 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（11ビット，上位1ビットは0固定）
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）
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
(×)	：×でアドレスされる内容

18.3　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	マシン・コード			
				オペ・コード	オペランド		
加算	ADD	r, m	(r) ← (r) + (m)	00000	m_R	m_C	r
		m, #n4	(m) ← (m) +n4	10000	m_R	m_C	n4
	ADDC	r, m	(r) ← (r) + (m) +CY	00010	m_R	m_C	r
		m, #n4	(m) ← (m) +n4 +CY	10010	m_R	m_C	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	m_C	r
		m, #n4	(m) ← (m) −n4	10001	m_R	m_C	n4
	SUBC	r, m	(r) ← (r) − (m) −CY	00011	m_R	m_C	r
		m, #n4	(m) ← (m) −n4 −CY	10011	m_R	m_C	n4
論理演算	OR	r, m	(r) ← (r) ∨ (m)	00110	m_R	m_C	r
		m, #n4	(m) ← (m) ∨ n4	10110	m_R	m_C	n4
	AND	r, m	(r) ← (r) ∧ (m)	00100	m_R	m_C	r
		m, #n4	(m) ← (m) ∧ n4	10100	m_R	m_C	n4
	XOR	r, m	(r) ← (r) ⊻ (m)	00101	m_R	m_C	r
		m, #n4	(m) ← (m) ⊻ n4	10101	m_R	m_C	n4
判断	SKT	m, #n	CMP ←0, if (m) ∧ n=n, then skip	11110	m_R	m_C	n
	SKF	m, #n	CMP ←0, if (m) ∧ n=0, then skip	11111	m_R	m_C	n
比較	SKE	m, #n4	(m) −n4, skip if zero	01001	m_R	m_C	n4
	SKNE	m, #n4	(m) −n4, skip if not zero	01011	m_R	m_C	n4
	SKGE	m, #n4	(m) −n4, skip if not borrow	11001	m_R	m_C	n4
	SKLT	m, #n4	(m) −n4, skip if borrow	11011	m_R	m_C	n4
回転	RORC	r	→ CY → $(r)_{b3}$ → $(r)_{b2}$ → $(r)_{b1}$ → $(r)_{b0}$ →	00111	000	0111	r
転送	LD	r, m	(r) ← (m)	01000	m_R	m_C	r
	ST	m, r	(m) ← (r)	11000	m_R	m_C	r
	MOV	@r, m	if MPE=1 : (MP, (r)) ← (m) if MPE=0 : (BANK, m_R, (r)) ← (m)	01010	m_R	m_C	r
		m, @r	if MPE=1 : (m)←(MP, (r)) if MPE=0 : (m)←(BANK, m_R, (r))	11010	m_R	m_C	r
		m, #n4	(m) ←n4	11101	m_R	m_C	n4
	MOVT[注]	DBF, @AR	SP ← SP −1, ASR ← PC, PC ← AR, DBF ← (PC), PC ← ASR, SP ← SP +1	00111	000	0001	0000 ★

注　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。 ★

命令群	ニモニック	オペランド	オペレーション	マシン・コード			
				オペ・コード	オペランド		
転送	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	rf_C
	POKE	rf, WR	(rf)←WR	00111	rf_R	0010	rf_C
	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	PC←addr	01100	addr		
		@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

18.4　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

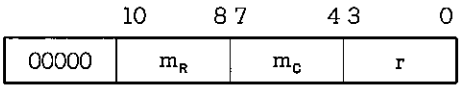
	ニモニック	オペランド	オペレーション	n
組み込みマクロ	SKTn	flag 1, …flag n	if (flag 1)～(flag n)＝all "1", then skip	1≦n≦4
	SKFn	flag 1, …flag n	if (flag 1)～(flag n)＝all "0", then skip	1≦n≦4
	SETn	flag 1, …flag n	(flag 1)～(flag n)←1	1≦n≦4
	CLRn	flag 1, …flag n	(flag 1)～(flag n)←0	1≦n≦4
	NOTn	flag 1, …flag n	if (flag n)＝"0", then (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

18.5　命令の個別説明

18.5.1　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 0	0	0	0	ロウ・アドレス			B C D

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) ＋ (0.6FH)

└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

```
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 ←000000010000B (0.10H) 注
MOV   IXM,   #01H
MOV   IXL,   #00H
SET1  IXE                  ; IXE フラグ ←1
ADD   MEM003, MEM02F       ; IX                    000000010000B (0.10H)
                           ; バンク・オペランド OR)000000101111B (0.2FH)
                           ; 指定アドレス          001001111111B (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。

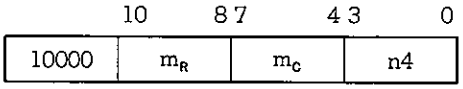
```
MEM013   MEM   0.13H
MEM02F   MEM   0.2FH
         ADD   MEM013, MEM02F
                  └─ジェネラル・レジスタのカラム・アドレスを意味し，下位4ビット
                     （この場合は03H）が有効となります。
```

CMPフラグ＝1のときは，加算結果が格納されません。

BCDフラグ＝1のときは，BCD演算した結果が格納されます。

(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。

(2.2FH) ← (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)
```

④ 注　意

CMPフラグ = 1のときは，加算結果が格納されません。

BCDフラグ = 1のときは，BCD演算した結果が格納されます。

(3) ADDC r, m　　Add data memory to general register with carry flag

① 命令コード

10	8 7	4 3	0
00010	m_R	m_C	r

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 2 ワード以上の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

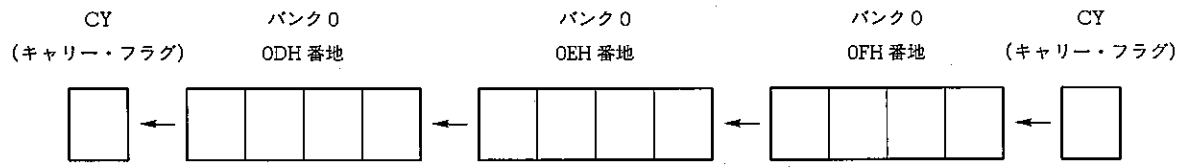
(0.0DH) ← (0.0DH) + (0.2DH) + CY

```
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
        ADDC  MEM00E, MEM02E
        ADDC  MEM00D, MEM02D
```

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

0.0FH番地の内容と0.40H番地から0.4FH番地の内容を加算し，結果を0.0FH番地へ格納します。

```
        (0.0FH) ← (0.0FH) + (0.40H) + (0.41H) +……+ (0.4FH)
MEM00F  MEM   0.0FH
MEM000  MEM   0.00H
        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
LOOP1 :
        SET1  IXE               ; IXEフラグ←1
        ADD   MEM00F, MEM000
        CLR1  IXE               ; IXEフラグ←0
        INC   IX                ; IX←IX+1
        SKE   IXL, #0
        JMP   LOOP1
```

例4

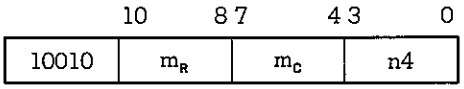
0.0DH番地から0.0FH番地の12ビットの内容に，0.40H番地から0.42H番地の12ビットの内容を加算した結果を，0.0DH番地から0.0FH番地の12ビットに格納します。

```
        (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)
```

```
ADDC  MEM00E, MEM001 ; (0.0EH) ← (0.0EH) + (0.41H)
ADDC  MEM00F, MEM002 ; (0.0FH) ← (0.0FH) + (0.42H)
```

(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 (0.40 H)
        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

10	8 7	4 3	0
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。

```
;アドレス              テーブル・データ
010H    DW            0F3FFH
011H    DW            0A123H
012H    DW            0FFF1H
013H    DW            0FFF5H
014H    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
        MOV     AR1, #1H        ;定します
        MOV     AR0, #0H
LOOP:
        MOVT    @AR             ;テーブル・データがDBFに読み込
        :                       ;まれます
        :
        :                       ;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
        BR      LOOP            ; メント
```

④ 注意

アドレス・レジスタの上位６ビットは０に固定されているため，下位10ビットのみ使用できます。

(6) INC IX　　Increment indexregister

① 命令コード

10	8 7	4 3	0
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。

```
        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：
MEM000  MEM     0.00H
        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
SUBC    IXM, #07H    ; うかをチェックします
SUBC    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
BR      RAMクリア    ; プします
```

18.5.2　減算命令

(1) SUB r, m　　**Subtract data memory from general register**

① 命令コード

10	8 7	4 3	0
00001	m_R	m_C	r

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
        SUB   MEM013, MEM02F
                  └ ジェネラル・レジスタ・アドレスは 00H-0FH の範囲（レジスタ・
                    ポインタをロウ・アドレス 1 以外に設定）でなければなりません。
```

CMP フラグ = 1 のときは，減算結果が格納されません。

BCD フラグ = 1 のときは，BCD 演算した結果が格納されます。

(2) SUB m, #n4　　**Subtract immediate data from data memory**

① 命令コード

10	8 7	4 3	0
10001	m_R	m_C	n4

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)
```

④ 注　意

CMPフラグ＝1のときは，減算結果が格納されません。

BCDフラグ＝1のときは，BCD演算した結果が格納されます。

(3) SUBC r, m　　Subtract data memory from general register with carry flag

① 命令コード

10	8 7	4 3	0
00011	m_R	m_C	r

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 2 ワード以上の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
        SUBC  MEM00E, MEM02E
        SUBC  MEM00D, MEM02D
```

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)
```

例3

0.00H番地から0.03H番地の12ビットの内容と0.0CH番地から0.0FH番地の12ビットの内容を比較し，同一であればLAB1へジャンプし，異なればLAB2へジャンプします。

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM003  MEM   0.03H
```

```
MEM00C  MEM   0.0CH
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        SET2  CMP, Z          ; CMP フラグ←1, Z フラグ←1
        SUB   MEM000, MEM00C  ; CMP フラグがセットされているため0.00H−0.03H
        SUBC  MEM001, MEM00D  ; 番地の内容は変化しません
        SUBC  MEM002, MEM00E  ;
        SUBC  MEM003, MEM00F  ;
        SKF1  Z               ; 比較した結果, 同一であればZフラグ = 1, 異なっ
        BR    LAB1            ; ていればZフラグ = 0 となります
        BR    LAB2

LAB1 :      ┆
            ┆
LAB2 :      ┆
```

(4) SUBC m, #n4　　Subtract immediate data from data memory with carry flag

① 命令コード

10	8 7	4 3	0
10011	m_R	m_C	n4

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
```

例３

0.00H 番地から 0.03H 番地の 12 ビットの内容とイミーディエト・データの 0A3FH を比較し，同一であれば LAB1 へジャンプし，異なれば LAB2 へジャンプします。

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM003  MEM   0.03H
        SET2  CMP, Z          ; CMP フラグ← 1，Z フラグ← 1
        SUB   MEM000, #0H     ; CMP フラグがセットされているため 0.00H−0.03H
        SUBC  MEM001, #0AH    ; 番地の内容は変化しません
        SUBC  MEM002, #3H     ;
        SUBC  MEM003, #0FH    ;
        SKF1  Z               ; 比較した結果，同一であれば Z フラグ ＝ 1，異なっ
        BR    LAB1            ; ていれば Z フラグ ＝ 0 となります
        BR    LAB2

LAB1 :    ┆
          ┆
LAB2 :    ┆
          ┆
```

18.5.3　論理演算命令

(1) OR r, m　　**OR between general register and data memory**

①　命令コード

10	8 7	4 3	0
00110	m_R	m_C	r

②　機　能

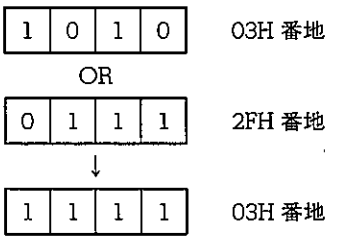
(r) ← (r) ∨ (m)

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1

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。

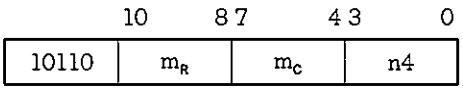
(0.03H) ← (0.03H) ∨ (0.2FH)




```
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

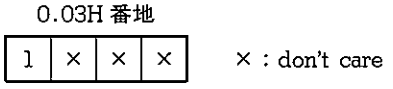
(m) ← (m) ∨ n4

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




```
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。

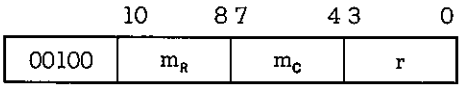
```
MEM003   MEM   0.03H
         OR    MEM003, #1111B
```

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

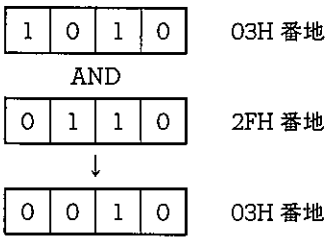
(r) ← (r) ∧ (m)

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1

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

10	8 7	4 3	0
10100	m_R	m_C	n4

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

0.03H 番地

0	×	×	×

× : 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

10	8 7	4 3	0
00101	m_R	m_C	r

② 機　能

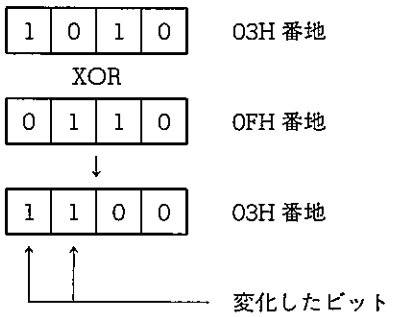
(r) ← (r) ⊻ (m)

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。

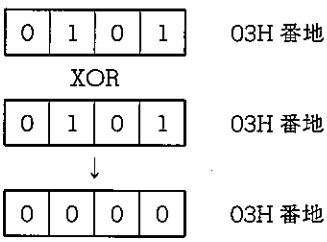
この例はオルタネート・スイッチの状態（0.03H番地の内容）と内部状態（0.0FH番地の内容）を比較し，変化したスイッチの処理へと分岐するときなどの例です。




```
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

10	8 7	4 3	0
10101	m_R	m_C	n4

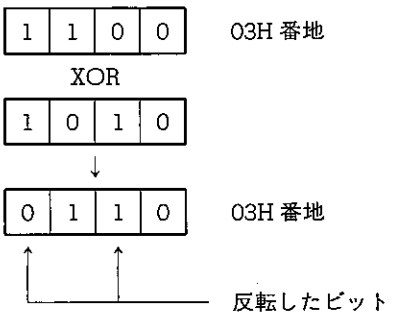
② 機　能

(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
```

18.5.4　判断命令

(1) SKT m, #n　　**Skip next instruction if data memory bits are true**

① 命令コード

10	8 7	4 3	0
11110	m_R	m_C	n

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
```

18.5.5　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	m_C	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

10	8 7	4 3	0
01011	m_R	m_C	n4

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。

	3		1
1EH	0011	1FH	0001

```
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

10	8 7	4 3	0
11001	m_R	m_C	n4

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

10	8 7	4 3	0
11011	m_R	m_C	n4

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”より大きければ，0FH番地の内容に01Hを格納し，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
```

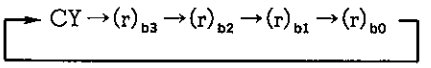
18.5.6　回転命令

(1) RORC r　　**Rotate right general register with carry flag**

① 命令コード

3			0
00111	000	0111	r

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。

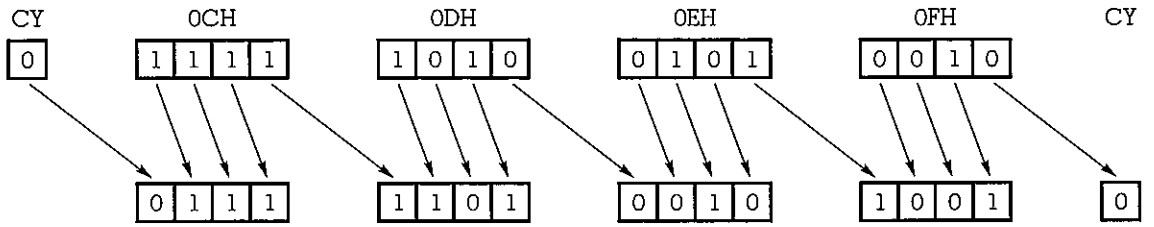
③ 例 1

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
```

18.5.7　転送命令

(1) LD r, m　　**Load data memory to general register**

① 命令コード

10	8 7	4 3	0
01000	m_R	m_C	r

② 機　能

(r) ← (m)

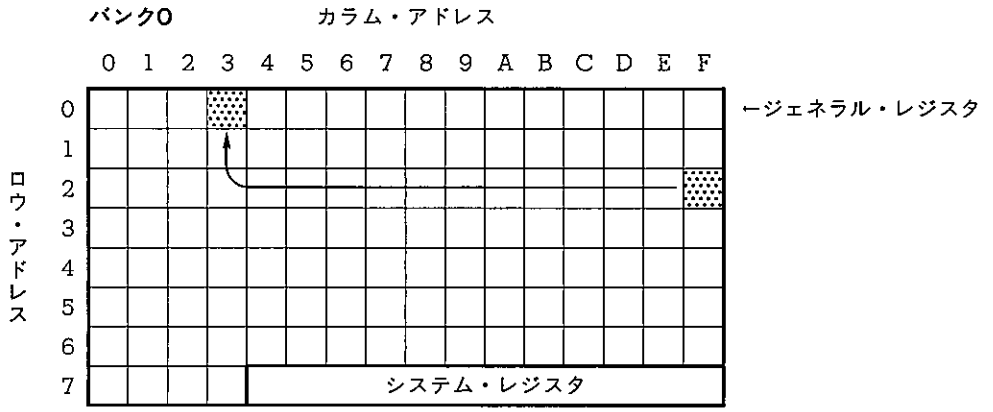
データ・メモリの内容をジェネラル・レジスタへ格納します。

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
        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

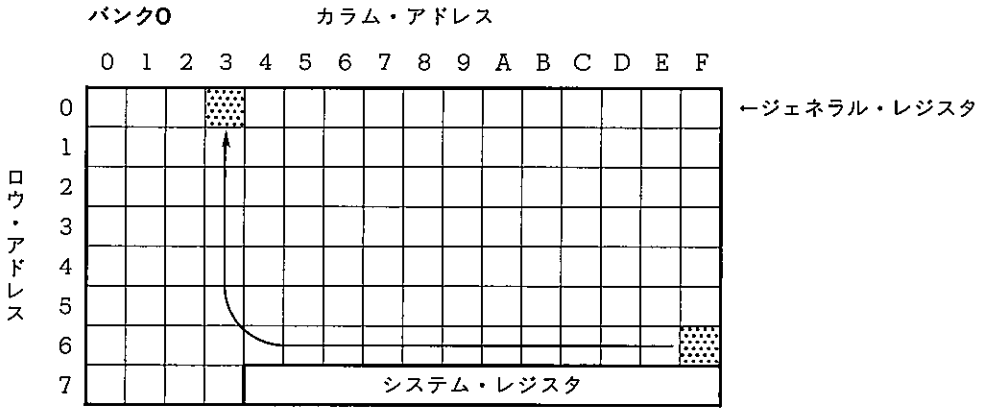
IXL←00H

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

10	8 7	4 3	0
11000	m_R	m_C	r

② 機　能

(m) ← (r)

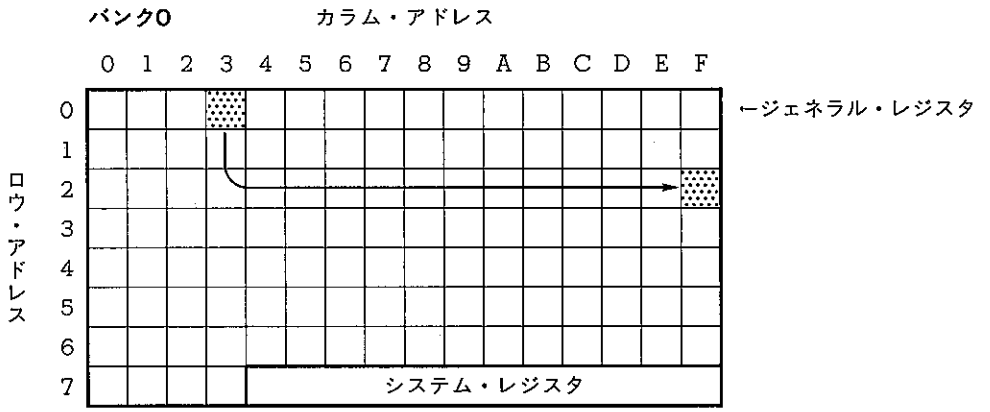
ジェネラル・レジスタの内容をデータ・メモリへ格納します。

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)

```
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)

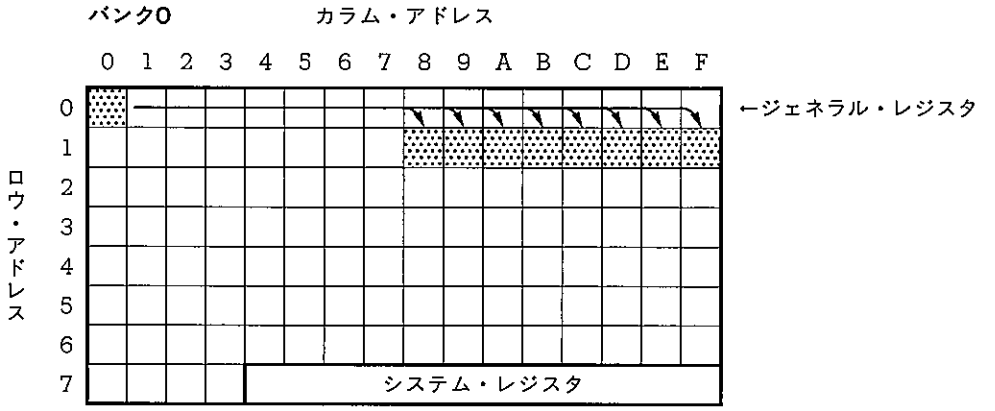
(0.19H) ← (0.00H)

⋮

(0.1FH) ← (0.00H)

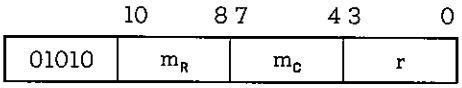
```
        MOV   IXH, #00H       ; IX ← 0000000000B (0.00H)
        MOV   IXM, #00H
        MOV   IXL, #00H       ; データ・メモリに 0.00H 番地を指定
```

```
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, m_R, (r)) ← (m)

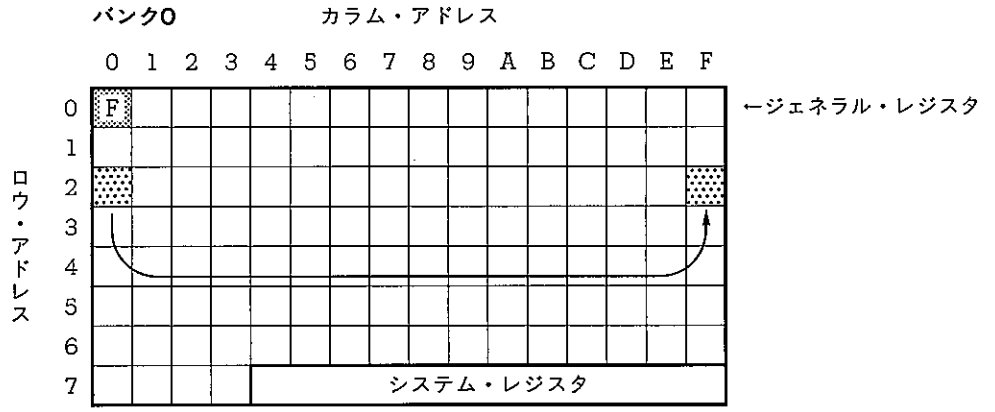
データ・メモリの内容をジェネラル・レジスタの内容でアドレスするデータ・メモリへ格納します。

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。

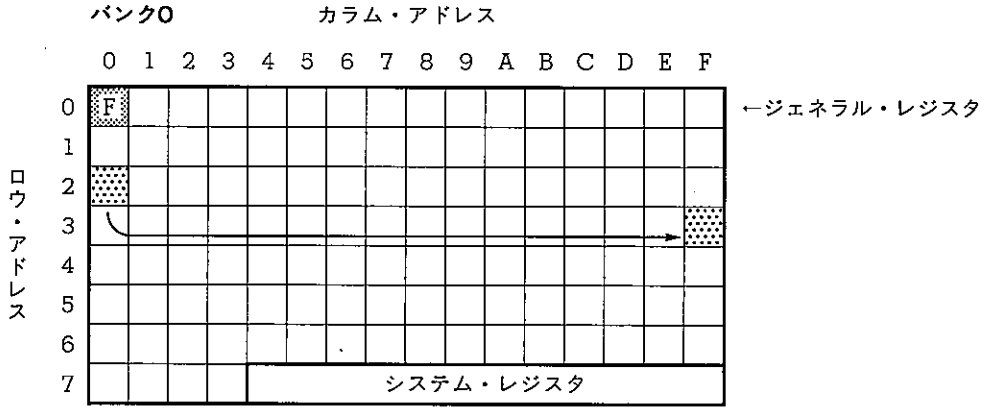
```
                (0.2FH) ← (0.20H)
MEM000   MEM    0.00H
MEM020   MEM    0.20H
         CLR1   MPE              ; MPEフラグ←0
         MOV    MEM000, #0FH     ; ジェネラル・レジスタにカラム・アドレス設定
         MOV    @MEM000, MEM020  ; 格納
```



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

10	8 7	4 3	0
11010	m_R	m_C	r

② 機　能

MPE = 1 のとき

(m) ← (MP, (r))

MPE = 0 のとき

(m) ← (BANK, m_R, (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。

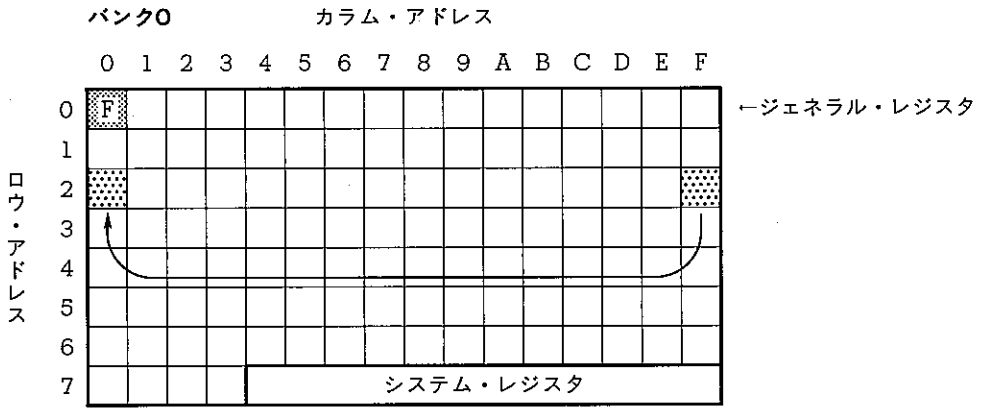
MPE = 0 のときには，同一バンクの同一ロウ・アドレス内での転送となります。

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)

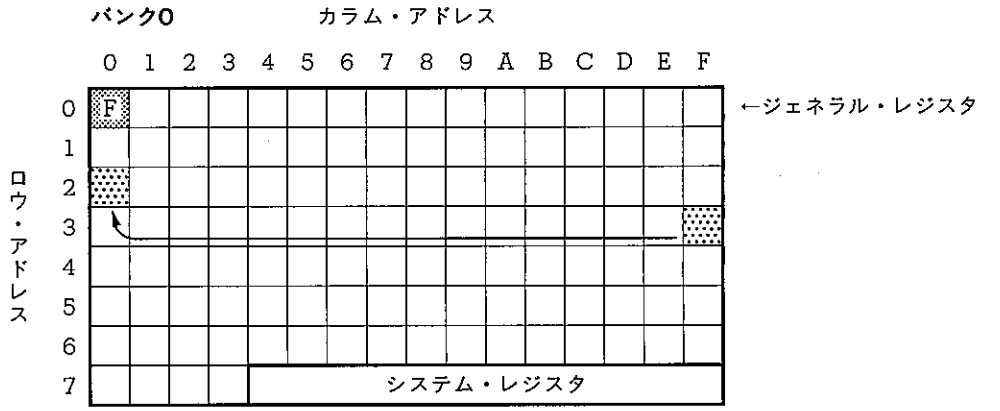
```
MEM000   MEM   0.00H
MEM020   MEM   0.20H
         CLR1  MPE                ; MPE フラグ ← 0
         MOV   MEM000, #0FH       ; ジェネラル・レジスタにカラム・アドレス設定
         MOV   MEM020, @MEM000    ; 格納
```



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	m_R	m_C	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

10	8 7	4 3	0
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
                ; *
アドレス        ORG   0010H
0010H           DW    0000000000000000B ; (0000H)
0011H           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,

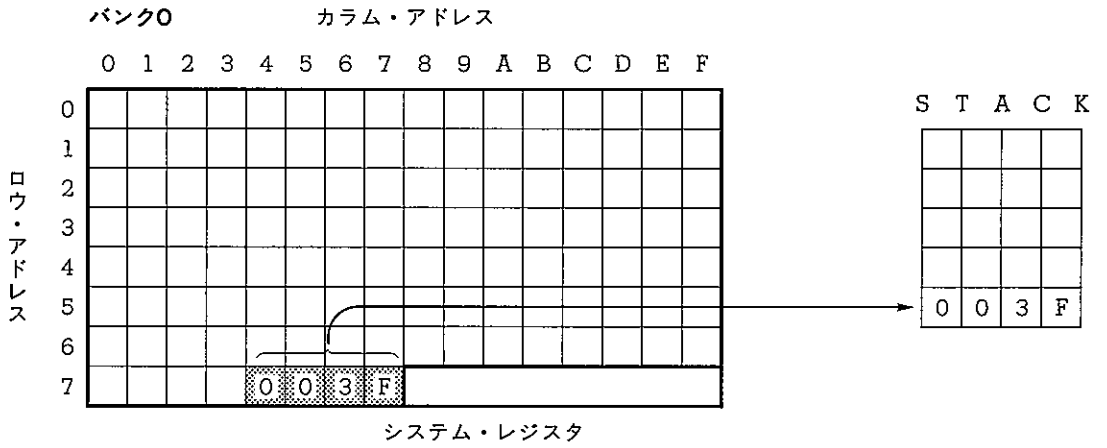
ASR ← AR

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

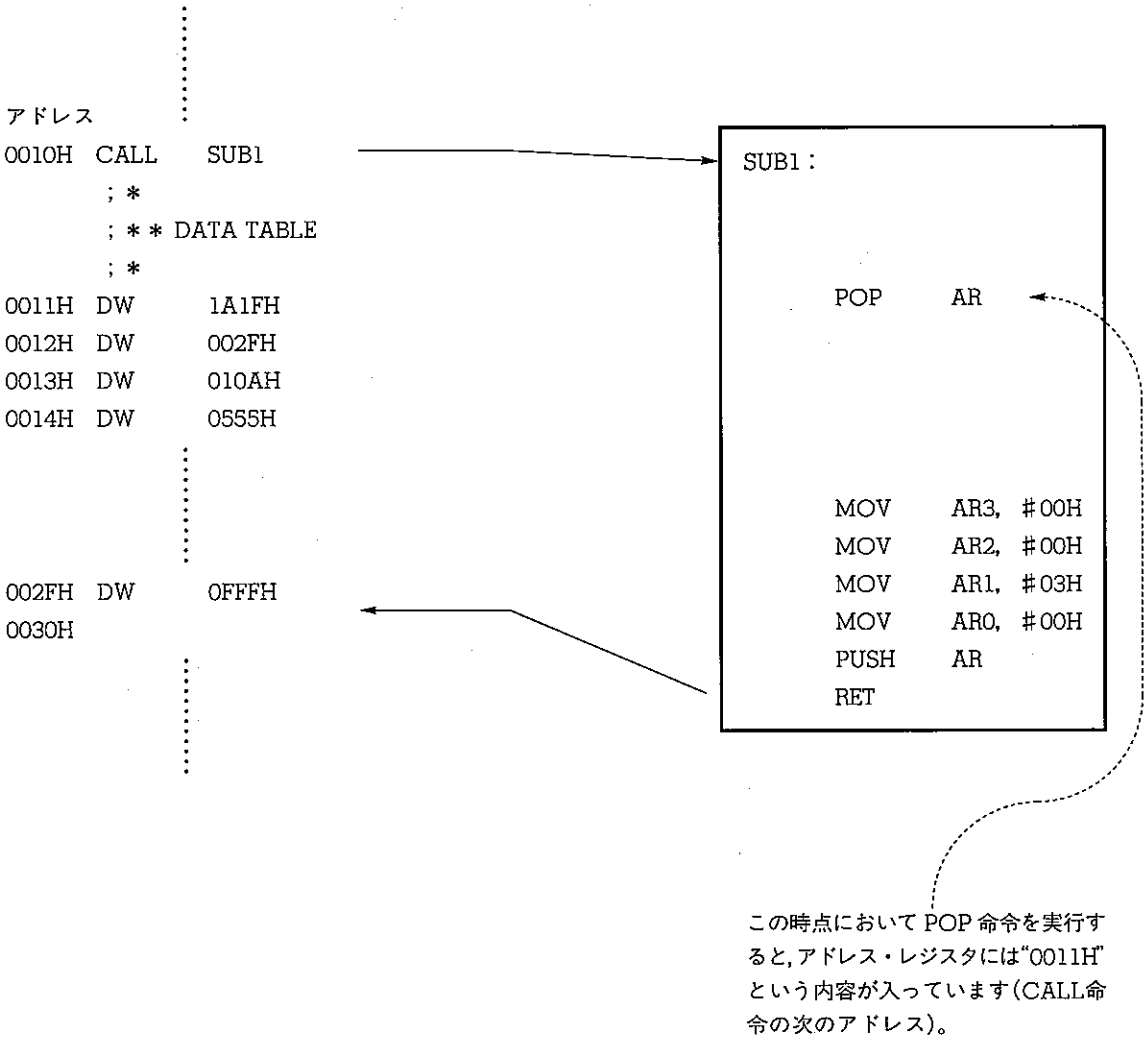
アドレス・レジスタに 003FH を設定し，スタックに格納します。

```
MOV     AR3, #00H
MOV     AR2, #00H
MOV     AR1, #03H
MOV     AR0, #0FH
PUSH    AR
```



例2

サブルーチンの後ろにデータ・テーブルがある場合にサブルーチンの戻り番地をアドレス・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

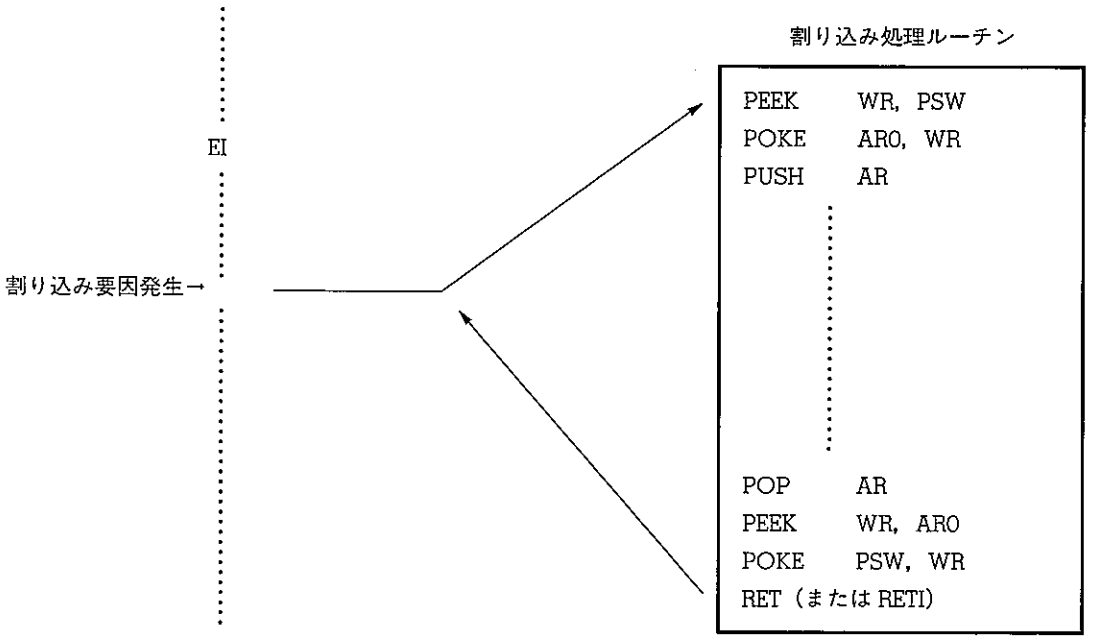
AR ←ASR,

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

00111	rf_R	0011	rf_C

② 機　能

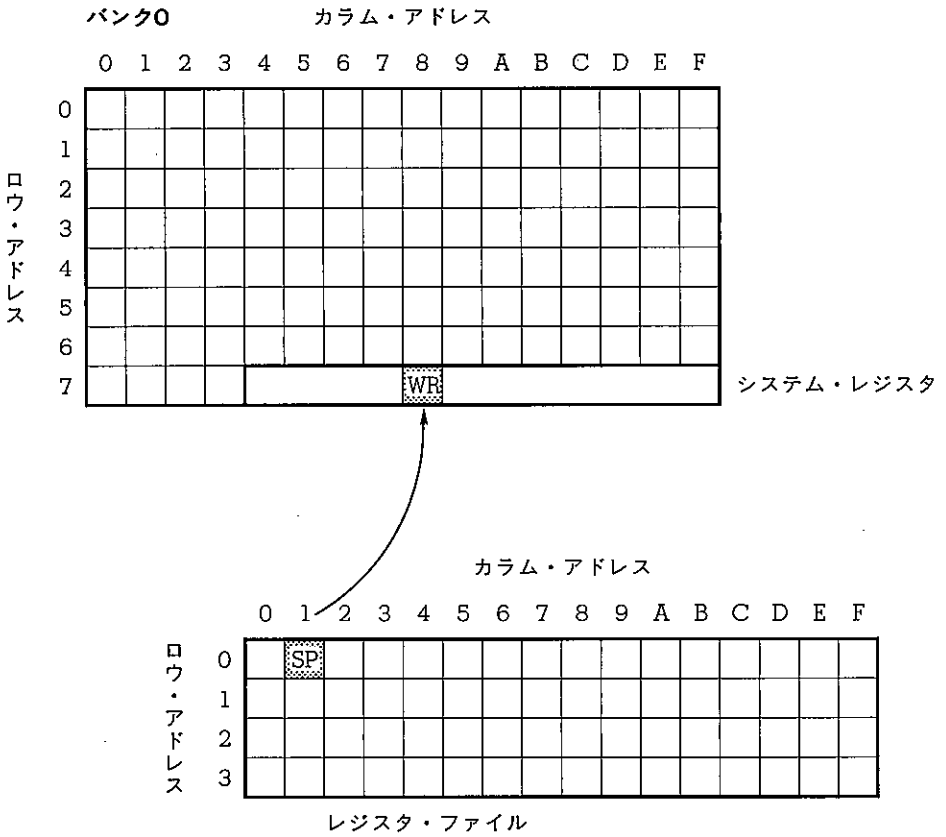
WR ← (rf)

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	rf_R	0010	rf_C

② 機　能

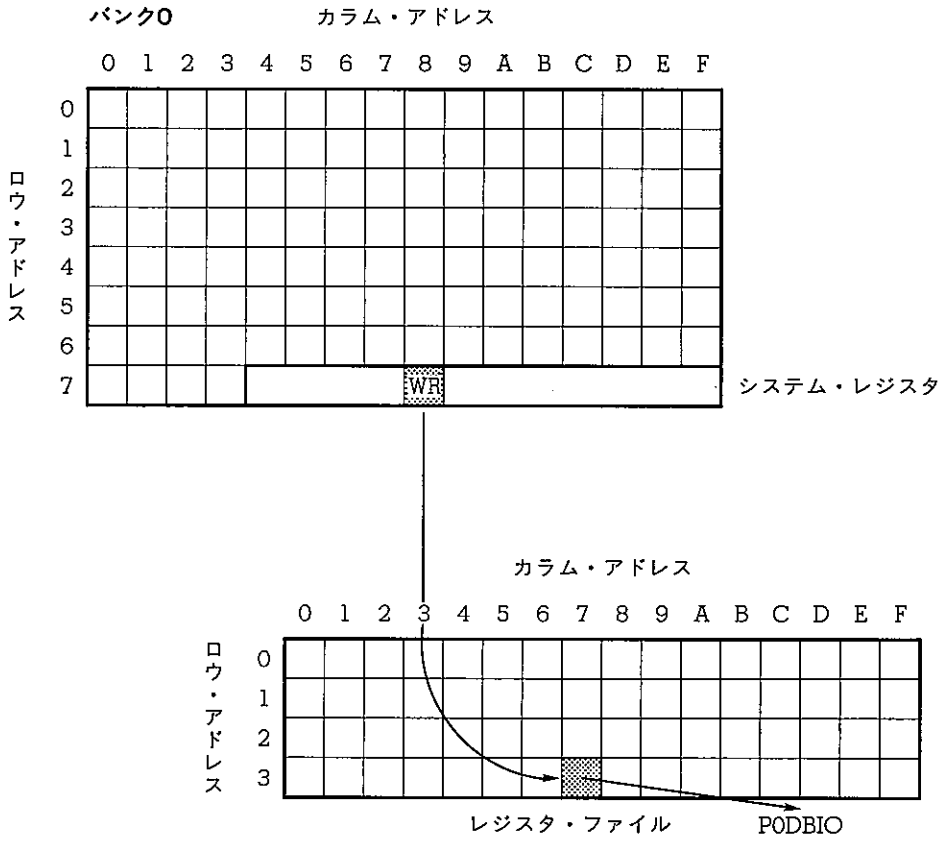
(rf) ← WR

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 1

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。

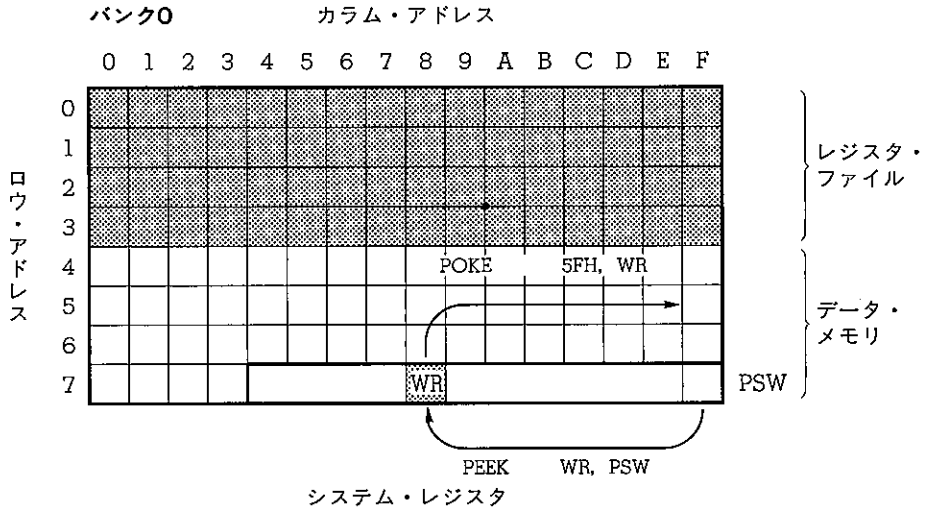
```
MOV     WR, #0FH
POKE    P0DBIO, WR    ; P0D0, P0D1, P0D2, P0D3 をすべて出力モードに設定
```



④ 注　意

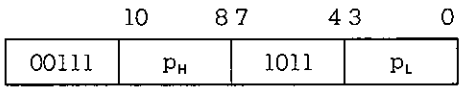
レジスタ・ファイルのうち40H-7FHはデータ・メモリ(74H-7FHはシステム・レジスタ)が見えています。したがってPEEK, POKE命令ではレジスタ・ファイル以外にデータ・メモリの各バンクの40H番地から7FH番地までをアクセスすることができます。たとえば以下のような使い方も可能です。

```
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

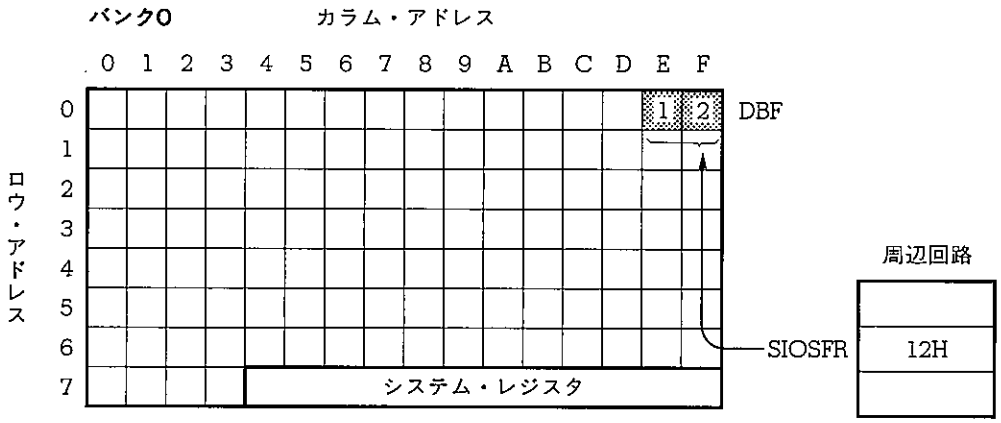
DBF ← (p)

周辺レジスタの内容をデータ・バッファDBFに格納します。

③ 例1

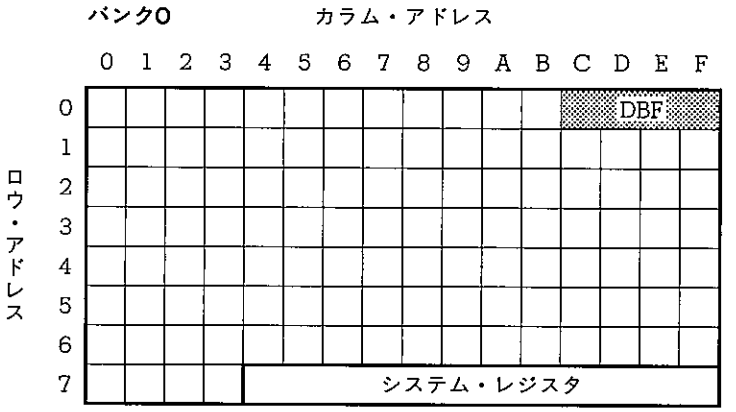
シリアル・インタフェースのシフト・レジスタSIOSFRの内容(8ビット)をデータ・バッファのDBF0, DBF1に格納します。

```
GET     DBF, SIOSFR
```



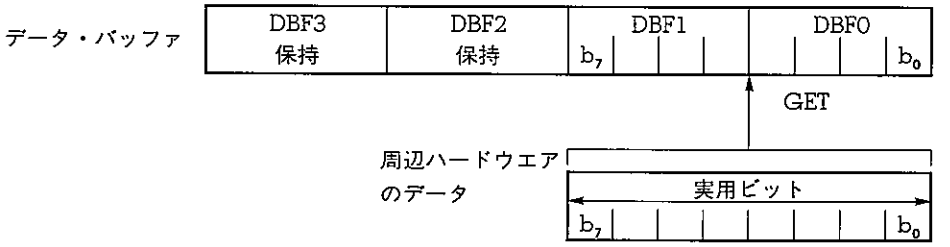
④ **注意 1**

データ・バッファは，バンク・レジスタの値に関係なくデータ・メモリのバンク 0 の 0CH, 0DH, 0EH, 0FH 番地に割り当てられます。



注意 2

データ・バッファは，全部で 16 ビットありますが，GET 命令でアクセスする周辺回路により入出力単位のビット数が異なります。たとえば 8 ビットの入出力単位となっている周辺回路に対し 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）にデータが格納されます。ご注意ください。



(12) PUT p, DBF　　Put data buffer to peripheral

① 命令コード

10	8 7	4 3	0
00111	p_H	1010	p_L

② 機　能

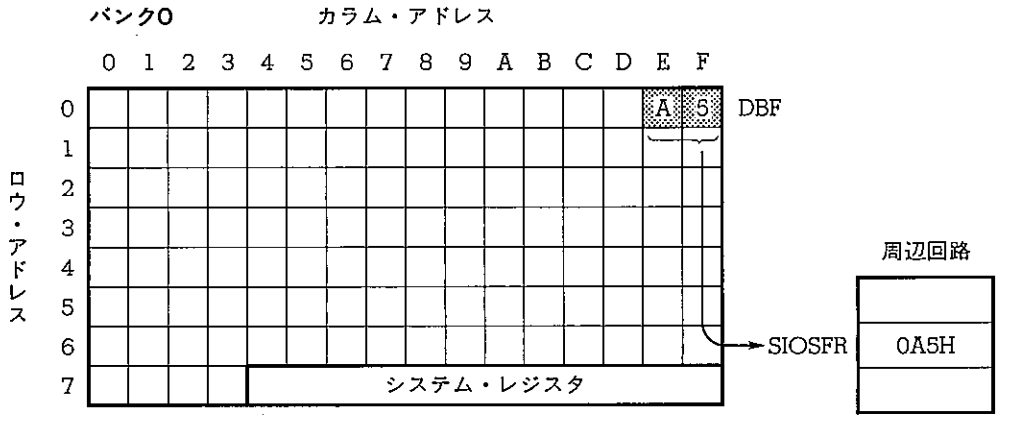
(p) ← DBF

データ・バッファ DBF の内容を周辺レジスタに格納します。

③ 例

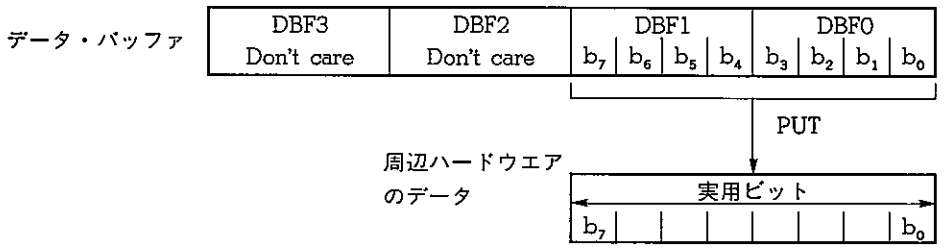
データ・バッファの DBF1 と DBF0 にそれぞれ 0AH, 05H を設定し，周辺レジスタのひとつであるシリアル・インタフェース用のシフト・レジスタ（SIOSFR）に転送します。

```
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

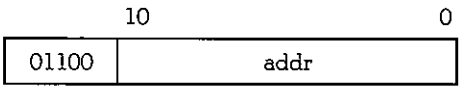
データ・バッファは，全部で 16 ビットありますが，PUT 命令でアクセスする周辺回路により入出力単位のビット数が異なります。たとえば SIO のシフト・レジスタは，8 ビットの入出力単位となっているため PUT 命令を実行した場合は，データ・バッファ DBF の下位 8 ビット（DBF1, DBF0）の内容のみ転送します（DBF3, DBF2 の内容は転送されません）。ご注意ください。



18.5.8　分岐命令

(1) BR addr　　**Branch to the address**

①　命令コード



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 :
```

(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
アドレス    ; *
0010H   BR      AAA
0011H   BR      BBB
0012H   BR      CCC
```

```
0013H     BR      DDD
0014H     BR      EEE
0015H     BR      FFF
0016H     BR      GGG
0017H     BR      HHH
0018H     BR      ZZZ
                  :
                  :
                  :
MEM010    MEM     0.10H
          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
          MOV     AR3,    #00H     ; AR3 ← 00H AR を 001 × H にする
          MOV     AR2,    #00H     ; AR2 ← 00H
          MOV     AR1,    #01H     ; AR1 ← 01H
          ST      AR0,    MEM010   ; AR0 ← 0.10H
          SKF     AR0,    #1000B   ; AR0 の内容が 08H よりも大きい場合は，AR0 の内容を 08H
          AND     AR0,    #1000B   ; にする
          BR      @AR
```

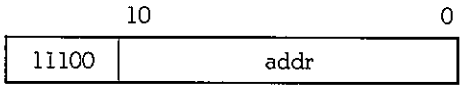
④ 注　意

アドレス・レジスタの上位６ビットは０に固定されているため，下位10ビットのみ使用できます。

18.5.9　サブルーチン命令

(1) CALL addr　　**Call subroutine**

① 命令コード



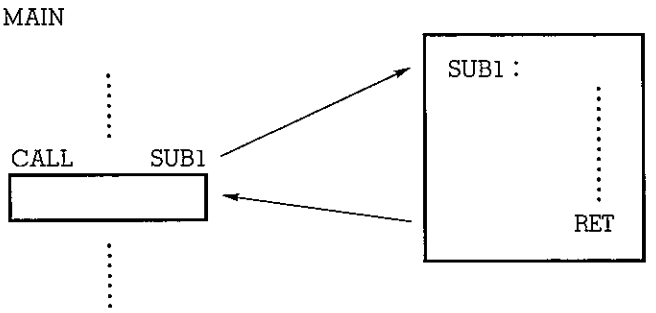
② 機　能

SP ← SP − 1, ASR ← PC,　★

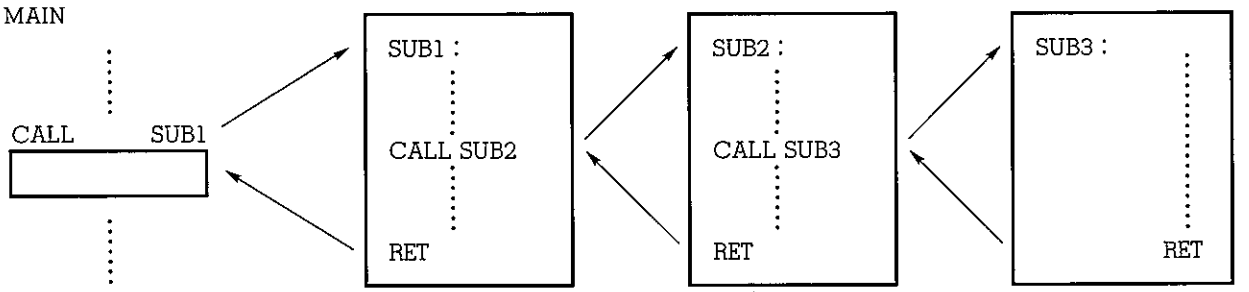
PC ← addr

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

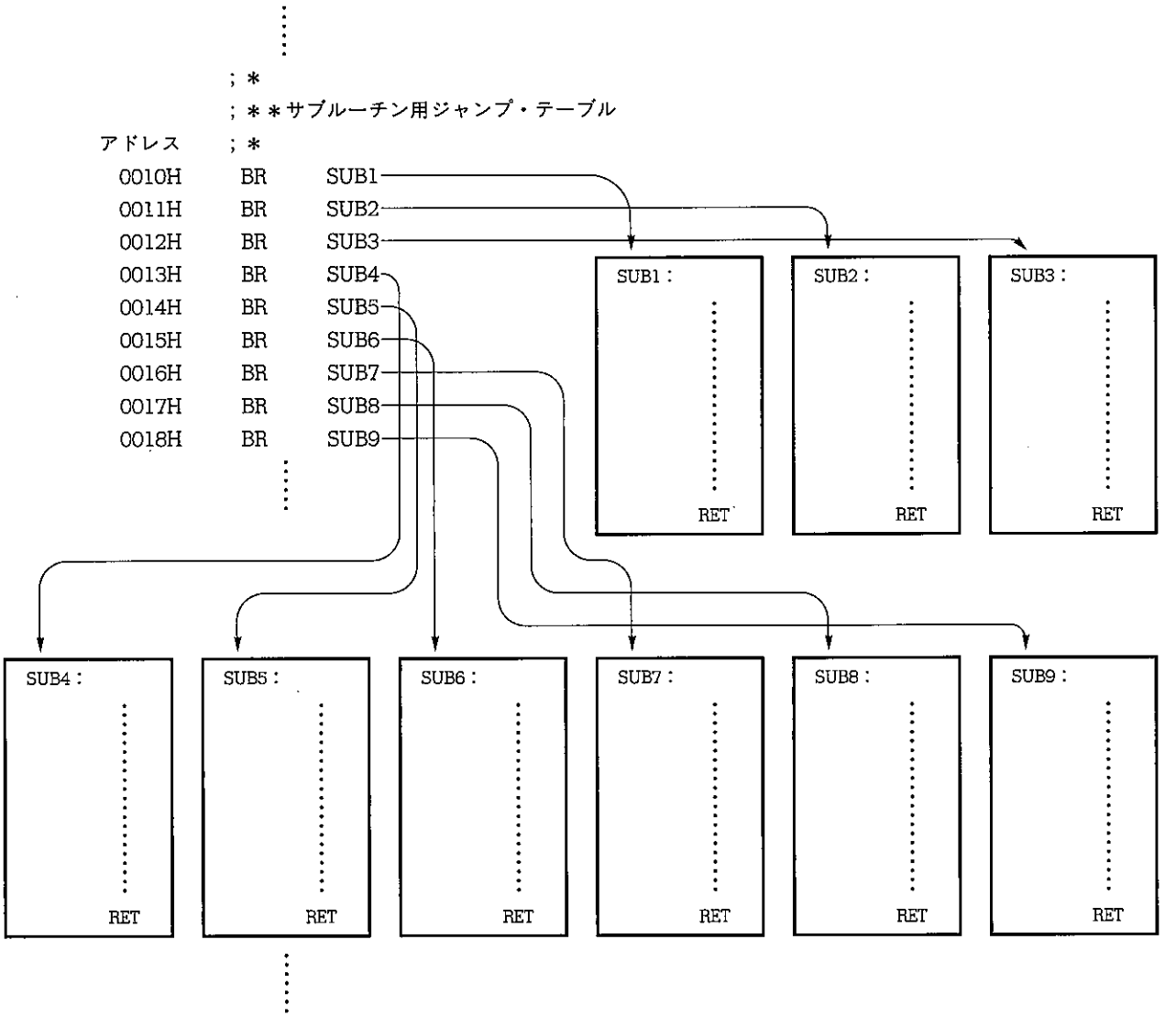
アドレス・レジスタ AR (AR0 − AR3) に 0020H を設定し, CALL @AR 命令により 0020H 番地のサブルーチンをコールします。

```
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




```
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
MOV   AR3,   #00H    ；AR3←00H　アドレス・レジスタを001×H
MOV   AR2,   #00H    ；AR2←00H
MOV   AR1,   #01H    ；AR1←01H
ST    AR0,   10H     ；AR0←0.10H
SKF   AR0,   #1000B  ；AR0の内容が08Hより大きい場合は，
AND   AR0,   #1000B  ；AR0の内容を08Hにする
CALL  @AR    ――――――――→ジャンプ・テーブルへ
                    ←――― それぞれのサブルーチンにて
                          RET命令を実行するとここに
                          戻ります。
```

④ 注　意

アドレス・レジスタの上位６ビットは０に固定されているため，下位10ビットのみ使用できます。

(3) RET　　Return to the main program from subroutine

① 命令コード

10	8 7	4 3	0
00111	000	1110	0000

② 機　能

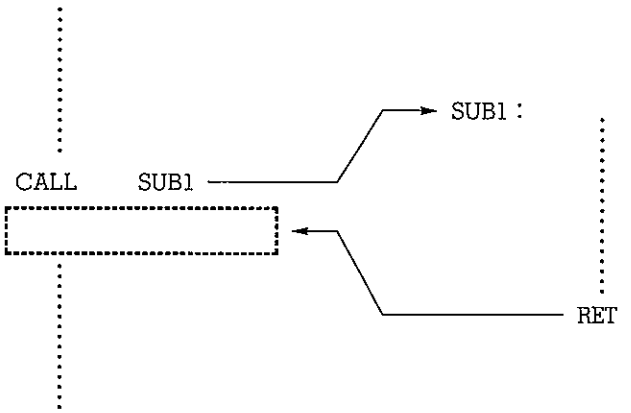
PC ← ASR,

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

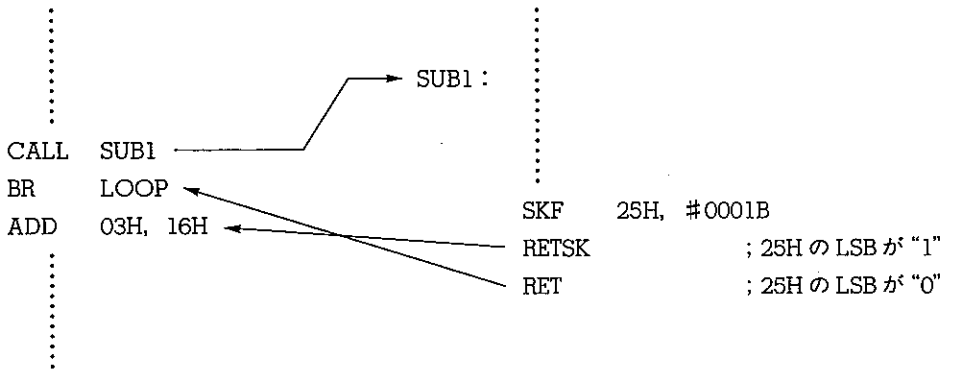
サブルーチンからメイン・プログラムへ戻るための命令です。

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・レジスタの一部もベクタ割り込み発生以前の状態に戻ります。

③ 注意1

割り込みにより自動的に退避される(RETI命令で復帰できる)システム・レジスタの内容はPSWORDです。

注意2

通常のサブルーチンにおいてRET命令の代わりにRETI命令を使用した場合，戻り番地に戻った時点でバンクなど（割り込みにより退避されるもの）が割り込みスタックの内容に書き換わってしまうため，どのような状態になるか不明の場合がありますので，サブルーチンからリターンする場合は必ずRET（またはRETSK）命令を使用してください。

18.5.10　割り込み命令

(1) EI　　Enable Interrupt

① 命令コード

00111	000	1111	0000

② 機　能

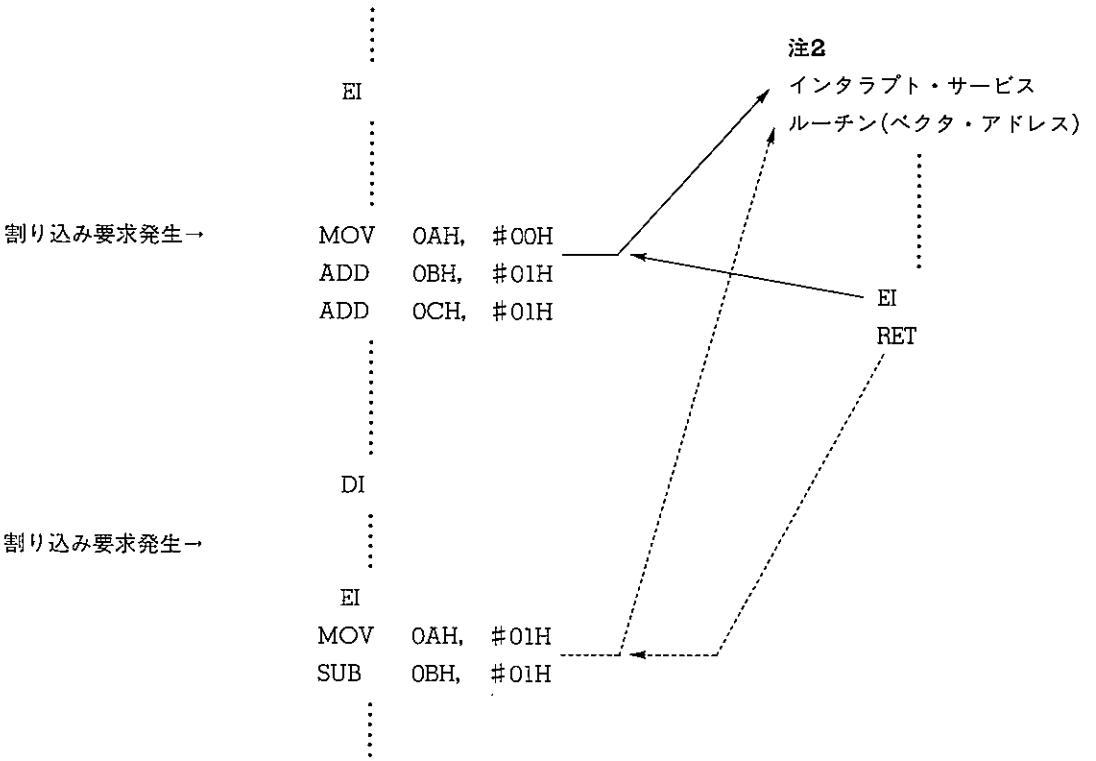
INTEF ← 1

ベクタ割り込みを許可する命令です。

割り込みはEI命令の次の命令を実行したあとに許可されます。

③ 例1

以下の例で分かるように割り込み要求が受け付けられると，受け付け後の次の命令(プログラム・カウンタを操作する命令を除く）の実行が完了してからベクタ・アドレスに流れが移ります。[注1]

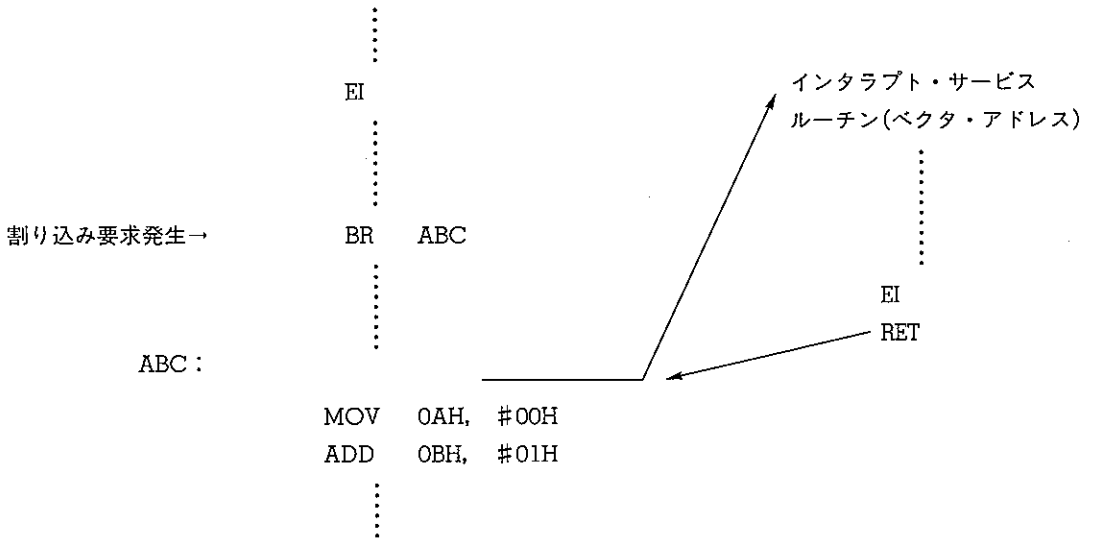


注1．ベクタ・アドレスは，受け付けられる割り込みにより異なります。表14－1　割り込み要因の種類を参照してください。

注2．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。各割り込みの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。

例2

プログラム・カウンタPCを操作する命令実行中に受け付けられた割り込み要求で発生する割り込みの例を以下に示します。



(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1を参照。

18.5.11　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19章　アセンブラ予約語

19.1　マスク・オプション疑似命令

μPD17120, 17121, 17132, 17133のプログラムを作成する場合，アセンブラのソース・プログラム中にマスク・オプション疑似命令を使用して，プルアップ抵抗が内蔵可能な端子すべてにプルアップ抵抗のあるなしを指定する必要があります。また，マスク・オプションを設定するためには，アセンブル時にAS171××(μPD171××用デバイス・ファイル)中のD171××.OPTファイルをカレント・ディレクトリに入れておかなければなりませんので注意してください。

以下の端子すべてにマスク・オプションを指定してください。

- $\overline{\text{RESET}}$ 端子
- ポートOD (POD_3, POD_2, POD_1, POD_0)
- ポートOE (POE_1, POE_0)

19.1.1　OPTION，ENDOP疑似命令

OPTION疑似命令から，ENDOP疑似命令までをマスク・オプション定義ブロックとします。

マスク・オプション定義ブロックの記述形式を以下に示します。このブロック内では，表19-1に示す3つの疑似命令だけが記述可能です。

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		

19.1.2　マスク・オプション定義疑似命令

各端子のマスク・オプションを定義する疑似命令を表19－1に示します。

表19－1　マスク・オプション定義疑似命令一覧表

端　子　名	マスク・オプション疑似命令	オペランドの数	パラメータ名
$\overline{\text{RESET}}$	OPTRES	1	OPEN　（プルアップ抵抗なし） PULLUP　（プルアップ抵抗あり）
$P0D_3$-$P0D_0$	OPTP0D	4	OPEN　（プルアップ抵抗なし） PULLUP　（プルアップ抵抗あり）
$P0E_1$, $P0E_0$	OPTP0E	2	OPEN　（プルアップ抵抗なし） PULLUP　（プルアップ抵抗あり）

OPTRESの記述形式を以下に示します。オペランド欄には$\overline{\text{RESET}}$のマスク・オプションを指定してください。

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：]	OPTRES	($\overline{\text{RESET}}$)	[；コメント]

OPTP0Dの記述形式を以下に示します。オペランド欄には第一オペランドから$P0D_3$, $P0D_2$, $P0D_1$, $P0D_0$の順にポートODすべての端子にマスク・オプションを指定してください。

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：]	OPTP0D	($P0D_3$), ($P0D_2$), ($P0D_1$), ($P0D_0$)	[；コメント]

OPTP0Eの記述形式を以下に示します。オペランド欄には第一オペランドから$P0E_1$, $P0E_0$の順にポートOEすべての端子に，マスク・オプションを指定してください。

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：]	OPTP0E	($P0E_1$), ($P0E_0$)	[；コメント]

マスク・オプションの記述例

$\overline{\text{RESET}}$ 端子 … プルアップ

$P0D_3$ … オープン，$P0D_2$ … オープン，$P0D_1$ … プルアップ，$P0D_0$ … プルアップ

$P0E_1$ … プルアップ，$P0E_0$ … オープン

シンボル欄	ニモニック欄	オペランド欄	コメント欄
; μPD17133			
マスク・オプション設定：	OPTION		
	;		
	OPTRES	PULLUP	
	OPTP0D	OPEN, OPEN, PULLUP, PULLUP	
	OPTP0E	PULLUP, OPEN	
	;		
	ENDOP		

19.2　予約シンボル

μPD17120サブシリーズのデバイス・ファイル（AS1712×，AS1713×）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。

19.2.1　予約シンボル一覧（μPD17120，17121の場合）

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

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
P0E1	FLG	0.6FH.1	R/W	ポート 0E のビット 1
P0E0	FLG	0.6FH.0	R/W	ポート 0E のビット 0
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

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
SIOEN	FLG	0.8AH.0	R/W	SIO イネーブル・フラグ
INT	FLG	0.8FH.0	R	INT 端子ステータス・フラグ
PDRESEN	FLG	0.90H.0	R/W	パワーダウン・リセット・イネーブル・フラグ
TMEN	FLG	0.91H.3	R/W	タイマ・イネーブル・フラグ
TMRES	FLG	0.91H.2	R/W	タイマ・リセット・フラグ
TMCK1	FLG	0.91H.1	R/W	タイマ・カウント・パルス選択フラグ・ビット 1
TMCK0	FLG	0.91H.0	R/W	タイマ・カウント・パルス選択フラグ・ビット 0
TMOSEL	FLG	0.92H.0	R/W	タイマ出力ポート/ポート選択フラグ
SIOTS	FLG	0.9AH.3	R/W	SIO スタート・フラグ
SIOHIZ	FLG	0.9AH.2	R/W	SO 端子の状態
SIOCK1	FLG	0.9AH.1	R/W	シリアル・クロック選択フラグ・ビット 1
SIOCK0	FLG	0.9AH.0	R/W	シリアル・クロック選択フラグ・ビット 0

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
IEGMD1	FLG	0.9FH.1	R/W	INT端子エッジ検出選択フラグ・ビット1
IEGMD0	FLG	0.9FH.0	R/W	INT端子エッジ検出選択フラグ・ビット0
P0BGIO	FLG	0.A4H.0	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）
IPSIO	FLG	0.AFH.2	R/W	SIO割り込みフラグ
IPTM	FLG	0.AFH.1	R/W	タイマ割り込み許可フラグ
IP	FLG	0.AFH.0	R/W	INT端子割り込み許可フラグ
P0EBIO1	FLG	0.B2H.1	R/W	$P0E_1$入力/出力選択フラグ（1＝出力ポート）
P0EBIO0	FLG	0.B2H.0	R/W	$P0E_0$入力/出力選択フラグ（1＝出力ポート）
P0DBIO3	FLG	0.B3H.3	R/W	$P0D_3$入力/出力選択フラグ（1＝出力ポート）
P0DBIO2	FLG	0.B3H.2	R/W	$P0D_2$入力/出力選択フラグ（1＝出力ポート）
P0DBIO1	FLG	0.B3H.1	R/W	$P0D_1$入力/出力選択フラグ（1＝出力ポート）
P0DBIO0	FLG	0.B3H.0	R/W	$P0D_0$入力/出力選択フラグ（1＝出力ポート）
P0CBIO3	FLG	0.B4H.3	R/W	$P0C_3$入力/出力選択フラグ（1＝出力ポート）
P0CBIO2	FLG	0.B4H.2	R/W	$P0C_2$入力/出力選択フラグ（1＝出力ポート）
P0CBIO1	FLG	0.B4H.1	R/W	$P0C_1$入力/出力選択フラグ（1＝出力ポート）
P0CBIO0	FLG	0.B4H.0	R/W	$P0C_0$入力/出力選択フラグ（1＝出力ポート）
P0ABIO3	FLG	0.B5H.3	R/W	$P0A_3$入力/出力選択フラグ（1＝出力ポート）
P0ABIO2	FLG	0.B5H.2	R/W	$P0A_2$入力/出力選択フラグ（1＝出力ポート）
P0ABIO1	FLG	0.B5H.1	R/W	$P0A_1$入力/出力選択フラグ（1＝出力ポート）
P0ABIO0	FLG	0.B5H.0	R/W	$P0A_0$入力/出力選択フラグ（1＝出力ポート）
IRQSIO	FLG	0.BDH.0	R/W	SIO割り込み要求フラグ
IRQTM	FLG	0.BEH.0	R/W	タイマ割り込み要求フラグ
IRQ	FLG	0.BFH.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
TMC	DAT	02H	R	タイマ・カウント・レジスタの周辺アドレス
TMM	DAT	03H	W	タイマ・モジュロ・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	PUT 命令，GET 命令，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

図 19-1　コントロール・レジスタの構成（μPD17120, 17121）(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						
	リセット時		0 1 0 1						
	Read/Write		R/W						
1 (9)	記号	0 0 0 PDRESEN	TMEN TMRES TMCK1 TMCK0	0 0 0 TMOSEL					
	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 P0BGIO			
	リセット時					0 0 0 0			
	Read/Write					R/W			
3 (B)	記号			0 0 P0EBIO1 P0EBIO0	P0DBIO3 P0DBIO2 P0DBIO1 P0DBIO0	P0CBIO3 P0CBIO2 P0CBIO1 P0CBIO0	P0ABIO3 P0ABIO2 P0ABIO1 P0ABIO0		
	リセット時			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0		
	Read/Write			R/W	R/W	R/W	R/W		

備考　（　）内はアセンブラ（AS17K）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 19-1　コントロール・レジスタの構成（μPD17120, 17121）(2/2)

8	9	A	B	C	D	E	F
		0 0 0 SIOEN					0 0 0 INT
		0 0 0 0					0 0 0 注
		R/W					R
		SIOTS SIOHIZ SIOCK1 SIOCK0					0 0 IEGMD1 IEGMD0
		0 0 0 0					0 0 0 0
		R/W					R/W
							0 IPSIO IPTM IP
							0 0 0 0
							R/W
					0 0 0 IRQSIO	0 0 0 IRQTM	0 0 0 IRQ
					0 0 0 0	0 0 0 1	0 0 0 0
					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。 ★

19.2.2　予約シンボル一覧(μPD17132, 17133, 17P132, 17P133の場合)

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

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
P0E1	FLG	0.6FH.1	R/W	ポート 0E のビット 1
P0E0	FLG	0.6FH.0	R/W	ポート 0E のビット 0
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

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
SIOEN	FLG	0.8AH.0	R/W	SIO イネーブル・フラグ
INT	FLG	0.8FH.0	R	INT 端子ステータス・フラグ
PDRESEN	FLG	0.90H.0	R/W	パワーダウン・リセット・イネーブル・フラグ
TMEN	FLG	0.91H.3	R/W	タイマ・イネーブル・フラグ
TMRES	FLG	0.91H.2	R/W	タイマ・リセット・フラグ
TMCK1	FLG	0.91H.1	R/W	タイマ・ソース・クロック選択フラグ・ビット 1
TMCK0	FLG	0.91H.0	R/W	タイマ・ソース・クロック選択フラグ・ビット 0
TMOSEL	FLG	0.92H.0	R/W	タイマ出力ポート/ポート選択フラグ
SIOTS	FLG	0.9AH.3	R/W	SIO スタート・フラグ
SIOHIZ	FLG	0.9AH.2	R/W	SO 端子の状態
SIOCK1	FLG	0.9AH.1	R/W	SIO ソース・クロック選択フラグ・ビット 1
SIOCK0	FLG	0.9AH.0	R/W	SIO ソース・クロック選択フラグ・ビット 0
CMPCH1	FLG	0.9CH.1	R/W	コンパレータ入力チャネル選択フラグ・ビット 1
CMPCH0	FLG	0.9CH.0	R/W	コンパレータ入力チャネル選択フラグ・ビット 0
CMPVREF3	FLG	0.9DH.3	R/W	コンパレータ・レファレンス電圧選択フラグ・ビット 3
CMPVREF2	FLG	0.9DH.2	R/W	コンパレータ・レファレンス電圧選択フラグ・ビット 2
CMPVREF1	FLG	0.9DH.1	R/W	コンパレータ・レファレンス電圧選択フラグ・ビット 1
CMPVREF0	FLG	0.9DH.0	R/W	コンパレータ・レファレンス電圧選択フラグ・ビット 0
CMPSTRT	FLG	0.9EH.1	R/W	コンパレータ・スタート・フラグ
CMPRSLT	FLG	0.9EH.0	R	コンパレータ比較結果フラグ
IEGMD1	FLG	0.9FH.1	R/W	INT 端子エッジ検出選択フラグ・ビット 1
IEGMD0	FLG	0.9FH.0	R/W	INT 端子エッジ検出選択フラグ・ビット 0
P0C3IDI	FLG	0.A3H.3	R/W	$P0C_3$ 入力ポート禁止フラグ（$P0C_3/Cin_3$ 選択）
P0C2IDI	FLG	0.A3H.2	R/W	$P0C_2$ 入力ポート禁止フラグ（$P0C_2/Cin_2$ 選択）
P0C1IDI	FLG	0.A3H.1	R/W	$P0C_1$ 入力ポート禁止フラグ（$P0C_1/Cin_1$ 選択）
P0C0IDI	FLG	0.A3H.0	R/W	$P0C_0$ 入力ポート禁止フラグ（$P0C_0/Cin_0$ 選択）
P0BGIO	FLG	0.A4H.0	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）
IPSIO	FLG	0.AFH.2	R/W	SIO 割り込みフラグ
IPTM	FLG	0.AFH.1	R/W	タイマ割り込み許可フラグ
IP	FLG	0.AFH.0	R/W	INT 端子割り込み許可フラグ
P0EBIO1	FLG	0.B2H.1	R/W	$P0E_1$ 入力/出力選択フラグ（1 = 出力ポート）
P0EBIO0	FLG	0.B2H.0	R/W	$P0E_0$ 入力/出力選択フラグ（1 = 出力ポート）
P0DBIO3	FLG	0.B3H.3	R/W	$P0D_3$ 入力/出力選択フラグ（1 = 出力ポート）
P0DBIO2	FLG	0.B3H.2	R/W	$P0D_2$ 入力/出力選択フラグ（1 = 出力ポート）
P0DBIO1	FLG	0.B3H.1	R/W	$P0D_1$ 入力/出力選択フラグ（1 = 出力ポート）
P0DBIO0	FLG	0.B3H.0	R/W	$P0D_0$ 入力/出力選択フラグ（1 = 出力ポート）

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.B4H.3	R/W	$P0C_3$ 入力/出力選択フラグ（1＝出力ポート）
P0CBIO2	FLG	0.B4H.2	R/W	$P0C_2$ 入力/出力選択フラグ（1＝出力ポート）
P0CBIO1	FLG	0.B4H.1	R/W	$P0C_1$ 入力/出力選択フラグ（1＝出力ポート）
P0CBIO0	FLG	0.B4H.0	R/W	$P0C_0$ 入力/出力選択フラグ（1＝出力ポート）
P0ABIO3	FLG	0.B5H.3	R/W	$P0A_3$ 入力/出力選択フラグ（1＝出力ポート）
P0ABIO2	FLG	0.B5H.2	R/W	$P0A_2$ 入力/出力選択フラグ（1＝出力ポート）
P0ABIO1	FLG	0.B5H.1	R/W	$P0A_1$ 入力/出力選択フラグ（1＝出力ポート）
P0ABIO0	FLG	0.B5H.0	R/W	$P0A_0$ 入力/出力選択フラグ（1＝出力ポート）
IRQSIO	FLG	0.BDH.0	R/W	SIO 割り込み要求フラグ
IRQTM	FLG	0.BEH.0	R/W	タイマ割り込み要求フラグ
IRQ	FLG	0.BFH.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
TMC	DAT	02H	R	タイマ・カウント・レジスタの周辺アドレス
TMM	DAT	03H	W	タイマ・モジュロ・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	PUT 命令，GET 命令，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

図 19-2　コントロール・レジスタの構成（μPD17132, 17133, 17P132, 17P133）(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						
	リセット時		0 1 0 1						
	Read/Write		R/W						
1 (9)	記号	0 0 0 PDRESEN	TMEN TMRES TMCK1 TMCK0	0 0 0 TMOSEL					
	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				P0C3IDI P0C2IDI P0C1IDI P0C0IDI	0 0 0 P0BGIO			
	リセット時				0 0 0 0	0 0 0 0			
	Read/Write				R/W	R/W			
3 (B)	記号			0 0 P0EBIO1 P0EBIO0	P0DBIO3 P0DBIO2 P0DBIO1 P0DBIO0	P0CBIO3 P0CBIO2 P0CBIO1 P0CBIO0	P0ABIO3 P0ABIO2 P0ABIO1 P0ABIO0		
	リセット時			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0		
	Read/Write			R/W	R/W	R/W	R/W		

備考　（　）内はアセンブラ（AS17K）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 19-2　コントロール・レジスタの構成（μPD17132, 17133, 17P132, 17P133）(2/2)

8	9	A	B	C	D	E	F
		0 0 0 SIOEN					0 0 0 INT
		0 0 0 0					0 0 0 注
		R/W					R
		SIOTS SIOHIZ SIOCK1 SIOCK0		0 0 CMPCH1 CMPCH0	CMPVREF3 CMPVREF2 CMPVREF1 CMPVREF0	0 0 CMPSTRT CMPRSLT	0 0 IEGMD1 IEGMD0
		0 0 0 0		0 0 0 0	1 0 0 0	0 0 0 1	0 0 0 0
		R/W		R/W	R/W	R/W R	R/W
							0 IPSIO IPTM IP
							0 0 0 0
							R/W
					0 0 0 IRQSIO	0 0 0 IRQTM	0 0 0 IRQ
					0 0 0 0	0 0 0 1	0 0 0 0
					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　開発ツール ★

μPD17120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1]）	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。 なお，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-17120）	SE-17120は，μPD17120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-17120CS）	EP-17120CSは，μPD17120サブシリーズ用のエミュレーション・プローブです。SE ボードとターゲット・システムを接続します。
PROM プログラマ （AF-9703[注3] AF-9704[注3] AF-9705[注3] AF-9706[注3]）	AF-9703, AF-9704, AF-9705, AF-9706は，μPD17P132, 17P133 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，μPD17P132, 17P133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 （AF-9808M[注3]）	AF-9808M は，μPD17P132CS, 17P132GT, 17P133CS, 17P133GT 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. 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シリーズ共通に使用できるアセンブラです。μPD17120サブシリーズのプログラム開発には，このAS17Kとデバイス・ファイル（AS17120，AS17121，AS17132，AS17133）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（AS17120 / AS17121 / AS17132 / AS17133）	AS17120，AS17121，AS17132，AS17133はμPD17120サブシリーズ用のデバイス・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	μS5A10AS17120[注]
					3.5インチ 2HD	μS5A13AS17120[注]
		IBM PC/AT	PC DOS		5インチ 2HC	μS7B10AS17120[注]
					3.5インチ 2HC	μS7B13AS17120[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××××AS17120には，AS17120，AS17121，AS17132，AS17133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 B　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 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。

(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。

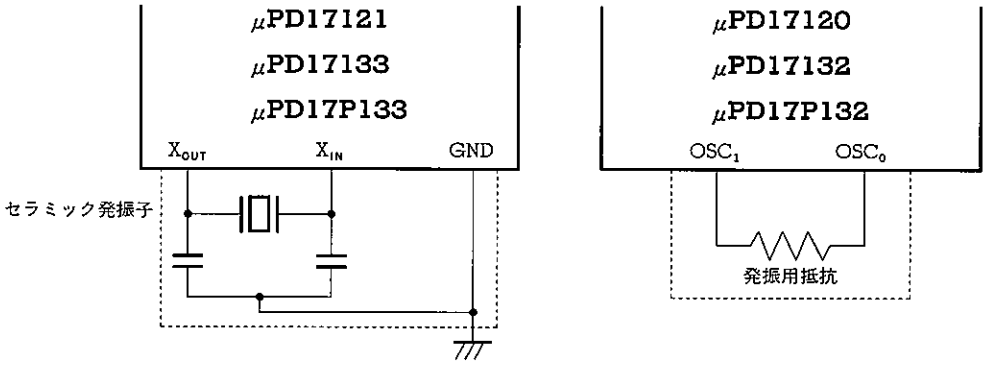
〔メ モ〕

付録C　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，X1，X2端子に接続されたセラミック発振子，またはOSC_1，OSC_0端子に接続された発振用抵抗によって発振します。

図C－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C－1　システム・クロック発振回路の外付け回路



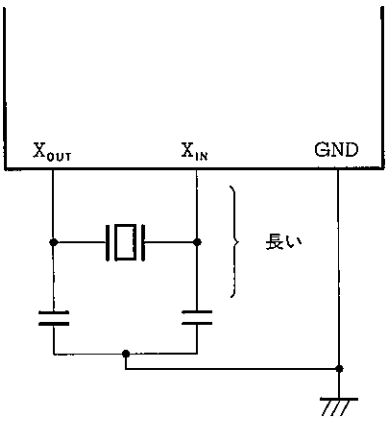
注意　システム・クロック発振回路は，グランド配線の抵抗成分やインダクタンス成分を極力小さくするように配線してください。また，配線容量などの影響を避けるために図C－1の [ ] の部分を次のように配線してください。

- ●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V_{SS}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。

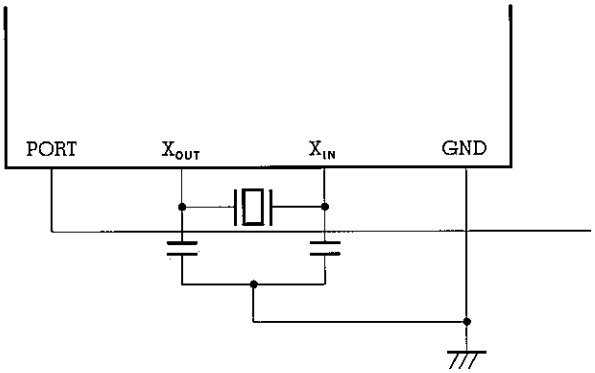
図C－2に発振回路の悪い例を示します。

図C－2　発振回路の悪い例

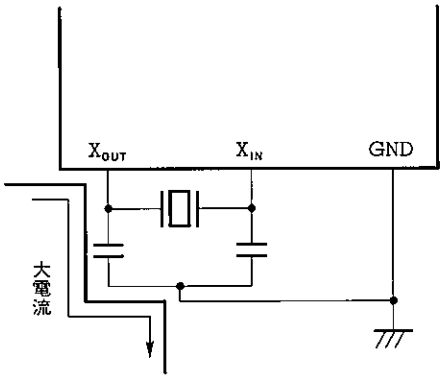
（a）接続回路の配線が長い



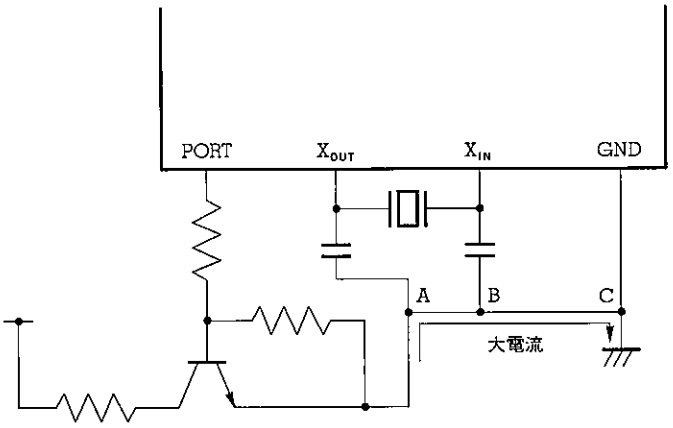
（b）信号線が交差している



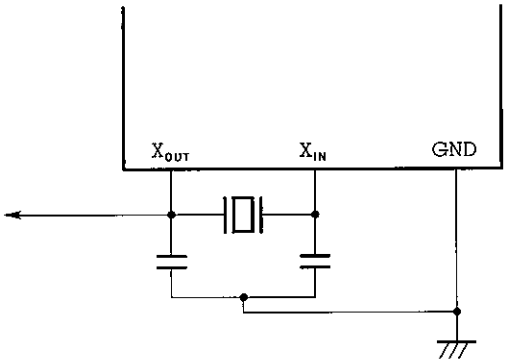
（c）変化する大電流が信号線に近接している



（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



付録 D　命令索引

D.1　命令索引（機能別）

D.2　命令索引（アルファベット順）

付録 E　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	1.4　端子接続図（Top View） (2) プログラム・メモリ書き込み/ベリファイ・モードの図を修正	第1章　概　　説
	表 2-1　未使用端子の処理を修正	第2章　端子機能
	2.4　$\overline{\text{RESET}}$ 端子と INT 端子の使用上の注意（通常動作モード時のみ）を追加	
	3.3　プログラム・カウンタ動作時の注意を追加	第3章　プログラム・カウンタ（PC）
	4.2.2　テーブル参照 テーブル参照命令の備考文を追加	第4章　プログラム・メモリ（ROM）
	7.7　プログラム・ステータス・ワード（PSWORD） プログラム・ステータス・ワードの説明文を修正	第7章　システム・レジスタ（SYSREG）
	表 7-2　ゼロ・フラグ（Z）とコンペア・フラグ（CMP）を修正	
	表 11-1　ALU 処理命令一覧を修正	第11章　演算論理ユニット（ALU）
	図 13-1　8ビット・タイマ・カウンタの構成を修正	第13章　周辺ハードウエア
	13.1.5　モジュロ・レジスタへのカウント値の設定と計算方法 図 13-3　モジュロ・レジスタへのカウント値の設定に注意文を追加 (2) インターバル時間の計算方法を修正	
	13.1.6　インターバル時間の誤差を追加	
	13.2.2　コンパレータの機能 コンパレート時間の記述を修正 注意文を追加	
	図 13-12　シリアル・インタフェースのブロック図に注意文を追加	
	14.3　割り込みシーケンスを修正	第14章　割り込み機能
	第15章　スタンバイ機能を修正	第15章　スタンバイ機能
	表 16-1　リセット時の各ハードウエアの状態を修正	第16章　リセット
	16.3.2　パワーオン・リセット機能と動作に注意文を追加	
	図 19-1，図 19-2　コントロール・レジスタの構成に注の文を追加	第19章　アセンブラ予約語
	付録 A　開発ツールを変更	付録 A　開発ツール
	付録 C　システム・クロック発振回路の構成上の注意を追加	付録 C　システム・クロック発振回路の構成上の注意

〔メ モ〕

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］ μPD17120サブシリーズ　ユーザーズ・マニュアル

（IEU-835A（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

キリトリ

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01 東京都港区芝五丁目7番1号 (NEC本社ビル)	東京	(03)3454-1111 (大代表)
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2755
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 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2762
半導体販売技術本部 西日本販売技術部	〒540 大阪市中央区城見一丁目4番24号 (NEC関西ビル)	大阪	(06) 945-3383

半導体
インフォメーションセンター
FAX(044)548-7900
(FAXにてお願い致します)

C94.11