NEC

ユーザーズ・マニュアル

μPD172××サブシリーズ

4ビット・シングルチップ・マイクロコンピュータ

共通事項

μPD17201A
μPD17203A
μPD17204
μPD17207
μPD17225
μPD17226
μPD17227
μPD17228
μPD17P203A
μPD17P204
μPD17P207
μPD17P218

資料番号　U12795JJ4V0UM00（第4版）
（旧資料番号　IEU−762B）
発行年月　November 1997　N

〔メ　モ〕

目　次　要　約

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	μPD17225, 17226, 17227, 17228を追加
全　般	μPD17202A, 17215, 17216, 17217, 17218, 17P202Aを削除
全　般	アセンブラを変更（AS17K→RA17K）
p.	1.1　機能一覧に一部追加
p.	15.4　アセンブラ（RA17K）組み込みマクロ命令の凡例の表に拡張命令を追加
p.	A.1　ハードウエアー覧を変更
p.	A.2　ソフトウエアー覧を変更

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，μPD172××サブシリーズの機能を理解し，それを用いたアプリケーション・システムを設計するユーザを対象としています。

目　的　このマニュアルはμPD172××サブシリーズの共通部分を抜き出したもので，プログラムを作る際の参考資料としていただくことを目的としています。

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

●一通りμPD172××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C　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
→**15.3　命令セット一覧**でその命令のニモニックを調べ，**15.5　命令の個別説明**でその機能を調べてください。

●μPD172××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	：上部ー下位，下部ー上位
注	：本文中につけた注の説明
注　意	：気をつけて読んでいただきたい内容
備　考	：本文の補足説明
数の表記	：2進数…××××または××××B 10進数…××××または××××D 16進数…××××H

関連資料　次の資料とあわせてご利用ください（表中の番号は資料番号です）。

●4ビット・シングルチップ・マイクロコントローラ（μPD172××サブシリーズ）（1/2）

品名 / 項目	μPD17201A	μPD17207	μPD17P207	μPD17203A	μPD17P203A	μPD17204	μPD17P204
データ・シート	U11778J [U11778E]		U11777J [U11777E]	IC-8089 [U10334E]	IC-8303 [IC-2851]	IC-8089 [U10334E]	IC-8303 [IC-2851]
インストラクション活用表	IEM-5537［IEM-1213］			IEM-5544			
ユーザーズ・マニュアル	U12795J（このマニュアル）						
アプリケーション・ノート	IEA-707［IEA-1285］			—	—	—	—
	IEA-757［IEA-1306］（浮動小数点演算パッケージ）			—	—	—	—
SEボード ユーザーズ・マニュアル	EEU-763 [EEU-1372]			EEU-762 [EEU-1371]			
デバイス・ファイル ユーザーズ・マニュアル	EEU-736 [EEU-1373]	EEU-746 [EEU-1360]		EEU-738 [EEU-1350]		EEU-751 [EEU-1361]	
リモコン用IC セレクション・ガイド	X10088J [X10088E]						
17Kシリーズ/DTS標準品 セレクション・ガイド	U10317J [U10317E]						
RA17K ユーザーズ・マニュアル	U10305J [U10305E]						
IE-17K/IE-17K-ET CLICE/CLICE-ET ユーザーズ・マニュアル	U10063J [U10063E]						
SIMPLEHOST® ユーザーズ・マニュアル	EEU-723：入門編［EEU-1336］ EEU-724：レファレンス編［EEU-1337］						
SIMPLEHOST emIC-17K®/RA17K対応版 ユーザーズ・マニュアル	EEU-5009：入門編［U10445E］ EEU-5007：レファレンス編［U10496E］						
プロジェクト・マネージャ ユーザーズ・マニュアル	U12810J [EEU-1527]						
MAKE/CNV17K ユーザーズ・マニュアル	U10596J [U10596E]						
emIC-17K ユーザーズ・マニュアル	EEU-876 [EEU-1511]						
LK17K ユーザーズ・マニュアル	U12518J [U12518E]						
DOC17K ユーザーズ・マニュアル	EEU-5006 [EEU-1536]						

備考　［　］内は英文の資料番号です。

● 4ビット・シングルチップ・マイクロコントローラ（μPD172××サブシリーズ）（2/2）

項目＼品名	μPD17225	μPD17226	μPD17227	μPD17228	μPD17P218
データ・シート	U12643J [U12643E]				U12217J [IC-3252]
インストラクション活用表	—	—	—	—	—
ユーザーズ・マニュアル	U12795J（このマニュアル）				
アプリケーション・ノート	—	—	—	—	—
SEボード ユーザーズ・マニュアル	U12372J (U12372E)				—
デバイス・ファイル ユーザーズ・マニュアル	U12136J (U12136E)				EEU-925 [EEU-1461]
リモコン用IC セレクション・ガイド	X10088J [X10088E]				
17Kシリーズ/DTS標準品 セレクション・ガイド	U10317J [U10317E]				
RA17K ユーザーズ・マニュアル	U10305J [U10305E]				
IE-17K/IE-17K-ET CLICE/CLICE-ET ユーザーズ・マニュアル	U10063J [U10063E]				
SIMPLEHOST ユーザーズ・マニュアル	EEU-723：入門編［EEU-1336］ EEU-724：レファレンス編［EEU-1337］				
SIMPLEHOST *emIC-17K*/RA17K対応版 ユーザーズ・マニュアル	EEU-5009：入門編［U10445E］ EEU-5007：レファレンス編［U10496E］				
プロジェクト・マネージャ ユーザーズ・マニュアル	U12810J [EEU-1527]				
MAKE/CNV17K ユーザーズ・マニュアル	U10596J [U10596E]				
emIC-17K ユーザーズ・マニュアル	EEU-876 [EEU-1511]				
LK17K ユーザーズ・マニュアル	U12518J [U12518E]				
DOC17K ユーザーズ・マニュアル	EEU-5006 [EEU-1536]				

備考 ［　］内は英文の資料番号です。

〔メ　モ〕

目　　次

図 の 目 次 (1/2)

図 の 目 次 (2/2)

表 の 目 次

〔メ　モ〕

第1章　概　　説

★　μPD17201A, 17203A, 17204, 17207, 17225, 17226, 17227, 17228はCPUとROM，RAM，入出力ポートのほか，タイマ，リモコン・キャリア・ジェネレータなどを1チップに集積した赤外線リモコン送信機用マイクロコントローラです。

μPD17P203A, 17P204, 17P207, 17P218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。

★ 1.1 機能一覧

●4ビット・シングルチップ・マイクロコントローラ（μPD172××サブシリーズ）（1/2）

品名 / 項目	μPD17201A	μPD17207	μPD17203A	μPD17204
ROM容量	3072×16ビット	4096×16ビット	4096×16ビット	7936×16ビット
RAM容量	336×4ビット		336×4ビット	
スタティックRAM	なし		4096×4ビット	2048×4ビット
LCDコントローラ／ドライバ	最大136セグメント		なし	
赤外線リモコン用キャリア発生回路（REM）	内蔵（LED出力はハイ・アクティブ）		内蔵（LED出力はロウ・アクティブ）	
赤外線リモコン受信プリアンプ	なし		内蔵	
入出力ポート数	19本		28本	
外部割り込み（INT）	1本（立ち上がりエッジ検出）		1本（立ち上がり，立ち下がりエッジ検出）	
A/Dコンバータ	4チャネル（8ビットA/D）		なし	
タイマ	2チャネル { 8ビット・タイマ 時計用タイマ		4チャネル { 8ビット・タイマ：3チャネル 時計用タイマ	
ウォッチドッグ・タイマ	内蔵（$\overline{\text{WDOUT}}$出力）			
低電圧検出回路	なし			
シリアル・インタフェース	1チャネル		1チャネル	
スタック・レベル	5レベル			7レベル
最小命令実行時間	4 μs（4 MHz動作時）			
動作電源電圧	2.2～5.5 V（4 MHz動作時） 2.0～5.5 V（32 kHz動作時）			
パッケージ	80ピン・プラスチックQFP（14×20 mm）		52ピン・プラスチックQFP（□14 mm）	
ワン・タイムPROM製品	μPD17P207		μPD17P203A	μPD17P204

注意　NEC送信フォーマットを使用する場合は，当社にカスタム・コードを申請してください。

●4ビット・シングルチップ・マイクロコントローラ（μPD172××サブシリーズ）（2/2）

品名 / 項目	μPD17225	μPD17226	μPD17227	μPD17228
ROM容量	2048×16ビット	4096×16ビット	6144×16ビット	8192×16ビット
RAM容量	111×4ビット		223×4ビット	
スタティックRAM	なし			
LCDコントローラ／ドライバ	なし			
赤外線リモコン用キャリア発生回路（REM）	内蔵（LED出力はなし）			
赤外線リモコン受信プリアンプ	なし			
入出力ポート数	20本			
外部割り込み（INT）	1本（立ち上がり，立ち下がりエッジ検出）			
A/Dコンバータ	なし			
タイマ	2チャネル｛8ビット・タイマ／ベーシック・インターバル・タイマ			
ウォッチドッグ・タイマ	内蔵（$\overline{\text{WDOUT}}$出力）			
低電圧検出回路[注]	内蔵（$\overline{\text{WDOUT}}$出力：マスク・オプション）			
シリアル・インタフェース	なし			
スタック・レベル	5レベル			
最小命令実行時間	2 μs（8 MHz動作時：高速モード） 4 μs（〃：通常モード）			
動作電源電圧	2.2～3.6 V（8 MHz動作時：高速モード） 2.0～3.6 V（〃：通常モード）			
パッケージ	28ピン・プラスチックSOP（375 mil） 28ピン・プラスチック・シュリンクDIP（400 mil）			
ワン・タイムPROM製品	μPD17P218			

注　回路構成上は同じですが，電気的特性が製品によって異なりますので注意してください。

注意　NEC送信フォーマットを使用する場合は，当社にカスタム・コードを申請してください。

〔メ　モ〕

第2章　プログラム・カウンタ（PC）

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

2.1　プログラム・カウンタの構成

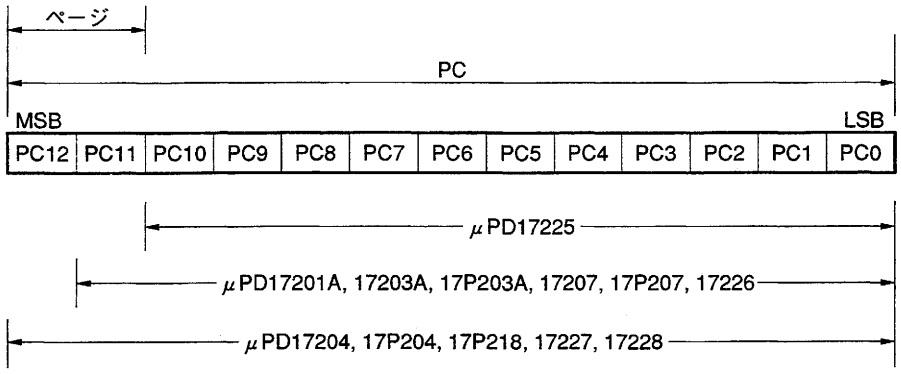
プログラム・カウンタは図2－1に示すように最大13ビットのバイナリ・カウンタで構成されています。

13ビットのバイナリ・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。

プログラム・カウンタは，アドレス・スタックおよびアドレス・レジスタとのデータ転送を16ビット単位で行います。

このとき，プログラム・カウンタのうち，プログラム・メモリ・アドレス範囲を越えるビットは0に固定されています。

★　図2－1　プログラム・カウンタ



2.2　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，プログラム・カウンタに指定された値が設定されます。

2.2.1～2.2.7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。

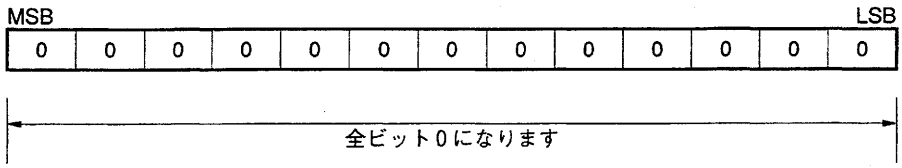
図2－2　命令実行後のプログラム・カウンタの値

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												
	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
リセット時	0	0	0	0	0	0	0	0	0	0	0	0	0
BR addr	命令で指定した値												
CALL addr	0	0	命令で指定した値										
BR @AR CALL @AR （MOVT DBF, @AR）	アドレス・レジスタの内容												
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												

2.2.1　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0000Hになります。

図2－3　リセット時のプログラム・カウンタの値



2.2.2　分岐命令（BR）実行時

分岐命令には，分岐先を命令のオペランドに記述する直接分岐命令（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。

図2－4では，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ています。

図2－4　直接分岐命令実行時のプログラム・カウンタの値

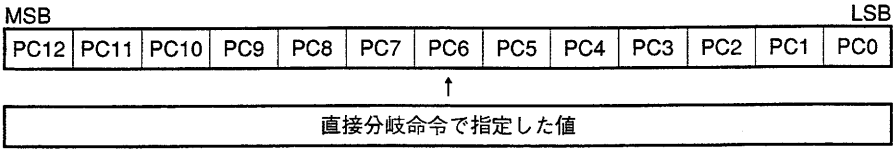
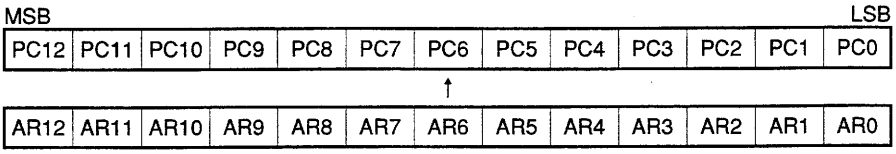


図2－5では，間接分岐命令によりアドレス・レジスタの値がプログラム・カウンタに設定されています。

図2－5　間接分岐命令実行時のプログラム・カウンタの値

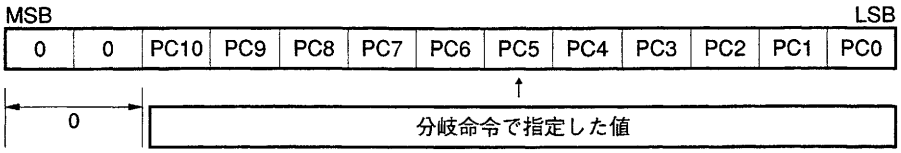


2.2.3　サブルーチン・コール命令（CALL）実行時

サブルーチン・コール命令には，分岐先を命令のオペランドに記述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。

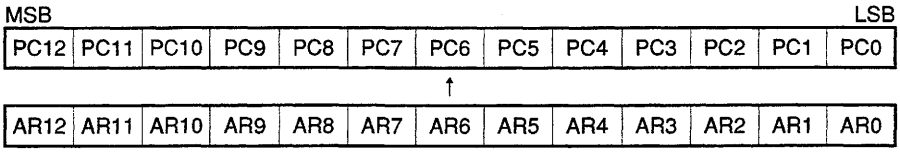
直接サブルーチン・コール命令により，プログラム・カウンタの値がアドレス・スタックにプッシュされたあと，オペランドに記述したアドレスがプログラム・カウンタに設定されます。直接サブルーチン・コール命令で指定できるアドレスの範囲はプログラム・メモリのページ0，すなわち0000H-07FFHです。

図2－6　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・スタックにプッシュされたあと，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2－7　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



2.2.4　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・スタックの値がプログラム・カウンタに設定され，そのアドレスの命令を実行します。

2.2.5　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。

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

2.2.6　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，スキップ命令を実行したとき，スキップ条件が成立してもしなくても，実行する命令数および命令実行時間は変わりません。

2.2.7　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，各割り込みに対するベクタ・アドレスがプログラム・カウンタに設定されます。

〔メ　モ〕

第3章　プログラム・メモリ（ROM）

μPD172××サブシリーズのプログラム・メモリ構成を次に示します。

★ 表3－1　μPD172××サブシリーズのプログラム・メモリ

品　名	プログラム・メモリ容量	プログラム・メモリ番地
μPD17225	2048×16ビット	0000H-07FFH
μPD17201A	3072×16ビット	0000H-0BFFH
μPD17203A	4096×16ビット	0000H-0FFFH
μPD17207		
μPD17226		
μPD17P203A		
μPD17P207		
μPD17227	6144×16ビット	0000H-17FFH
μPD17204	7936×16ビット	0000H-1EFFH
μPD17P204		
μPD17228	8192×16ビット	0000H-1FFFH
μPD17P218		

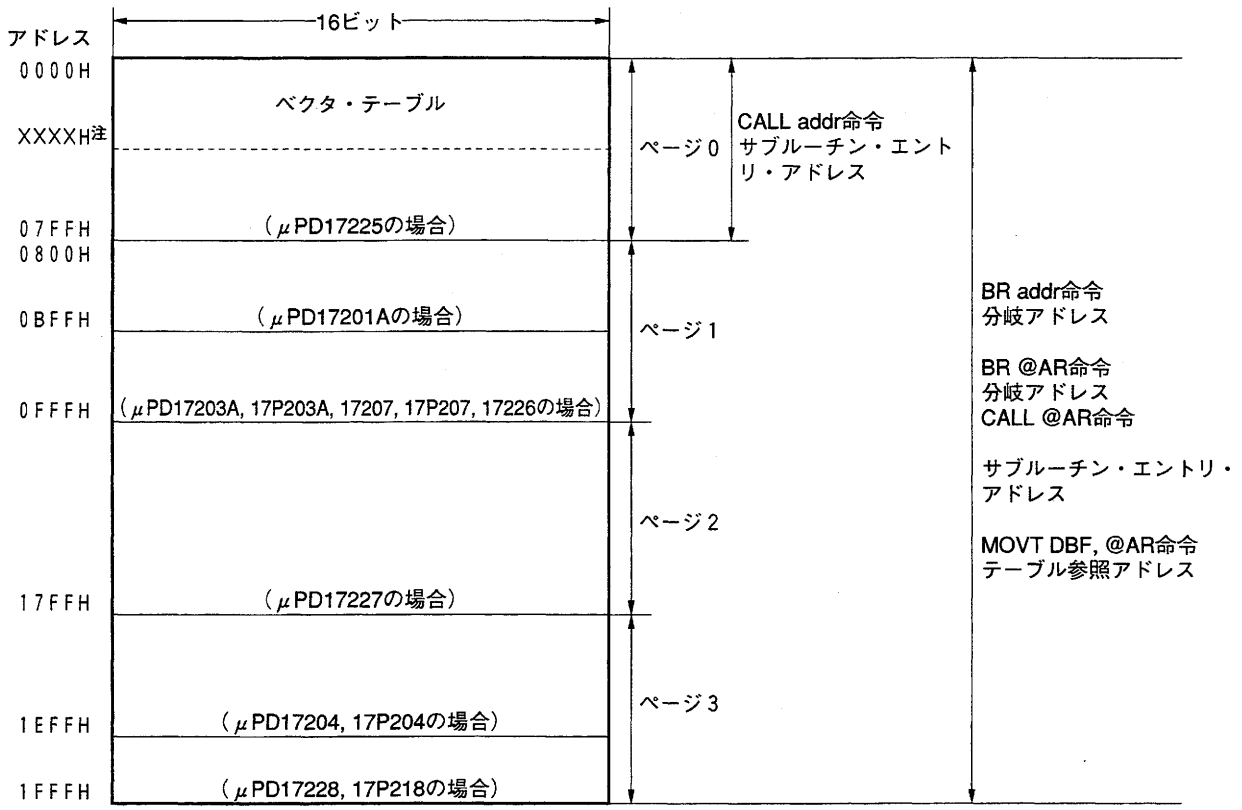
プログラム・メモリには，プログラム，割り込みベクタ・テーブル，および固定データ・テーブルなどを格納します。

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。

3.1　プログラム・メモリの構成

図3－1にプログラム・メモリ・マップを示します。BR addr, BR @AR, CALL @AR, MOVT DBF, @ARの各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。ただし，CALL addr命令のサブルーチン・エントリ・アドレスは，0000H-07FFHまでです。

図3－1　プログラム・メモリ・マップ



注　ベクタ・アドレスは，製品によって異なります。

3.2　プログラム・メモリの使い方

プログラム・メモリは大別して以下の2つの機能を持っています。

（1）プログラムの格納
（2）定数データの格納

プログラムとはCPU（Central Processing Unit）を動作させる“命令”の集まりであり，プログラムに書かれた命令に従い順次CPUが処理を実行しています。すなわち，CPUはプログラム・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，16ビット長のプログラム・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，定数データ読み出し専用命令であるMOVT命令を使用することによりプログラム・メモリの内容をデータ・メモリ上のデータ・バッファ（DBF）に読み出すことができます。このようにプログラム・メモリ上の定数データを読み出すことを“テーブル参照”と呼びます。

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

3.2.1　プログラムの格納

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。ところが，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，そのたび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，専用命令であるCALL命令により一箇所にまとめたプログラムを呼び出すことにより，何度でも1つのプログラムを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ことにより，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）ベクタ・テーブル

リセットあるいは割り込み発生時に分岐するアドレス（ベクタ・アドレス）を表3－2～表3－5に示します。

表3－2　μPD17201A, 17207, 17P207のベクタ・テーブル

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	時計用タイマ割り込み
0003H	外部入力（INT）割り込み
0004H	8ビット・タイマ割り込み

表3－3　μPD17203A, 17P203A, 17204, 17P204のベクタ・テーブル

ベクタ・アドレス	割り込み要因
0000H	リセット
0001H	XRAMアドレス割り込み
0002H	シリアル・インタフェース割り込み
0003H	時計用タイマ割り込み
0004H	外部入力（INT）割り込み
0005H	10ビット・タイマ割り込み（TM1）
0006H	8ビット・タイマ割り込み（TM0）
0007H	16ビット・タイマ割り込み（TM2）
0008H	エンベロープ回路出力割り込み

★ 表3－4　μPD17225,17226,17227,17228,17P218のベクタ・テーブル

ベクタ・アドレス	割り込み要因
0000H	リセット
0001H	ベーシック・インターバル・タイマ割り込み
0002H	外部入力（INT）割り込み
0003H	8ビット・タイマ割り込み

（2）直接分岐

直接分岐命令（BR addr）では命令のオペレーション・コードの下位2ビットと命令のオペランド11ビットの合計13ビットで分岐先のプログラム・メモリ・アドレスを指定します。したがって，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。

6.2　アドレス・レジスタの項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。したがって，直接サブルーチン・コール命令を使用する場合は，その呼び出し番地つまりサブルーチンの先頭番地はページ0（0000H-07FFH番地）に置く必要があります。ページ0以外（0800H-1FFFH番地）に先頭番地のあるサブルーチンは呼び出すことができません。

しかし，直接サブルーチン・コール命令やサブルーチン・リターン命令（RET, RETSK）はページ0以外にあってもかまいません。

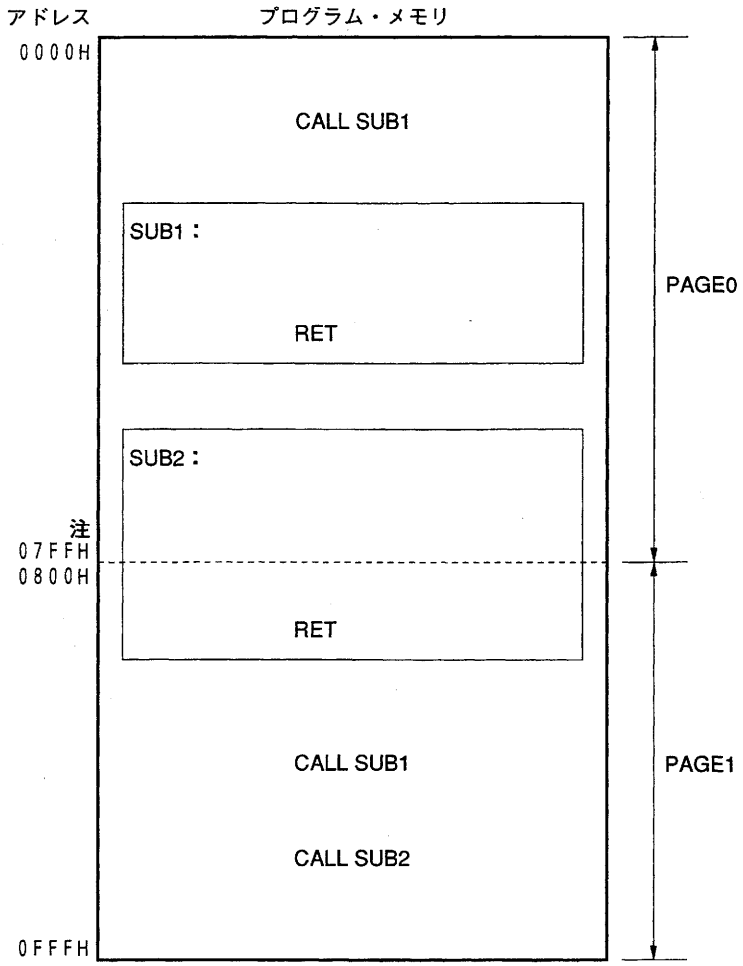
例1．サブルーチンからのリターン命令がページ0内にある場合

図3－2に示すように，サブルーチンの先頭番地がページ0内にあれば，このサブルーチンを呼び出すコール命令，リターン命令とともにどのページにあってもかまいません。

サブルーチンの先頭番地がページ0にあるかぎり，CALL命令はページの概念なく使用することができます。

しかしプログラム作成上，サブルーチンの先頭番地をページ0内に置くことができない場合は，例2に示す手法が有効となります。

図3－2　直接サブルーチン・コール（CALL addr）

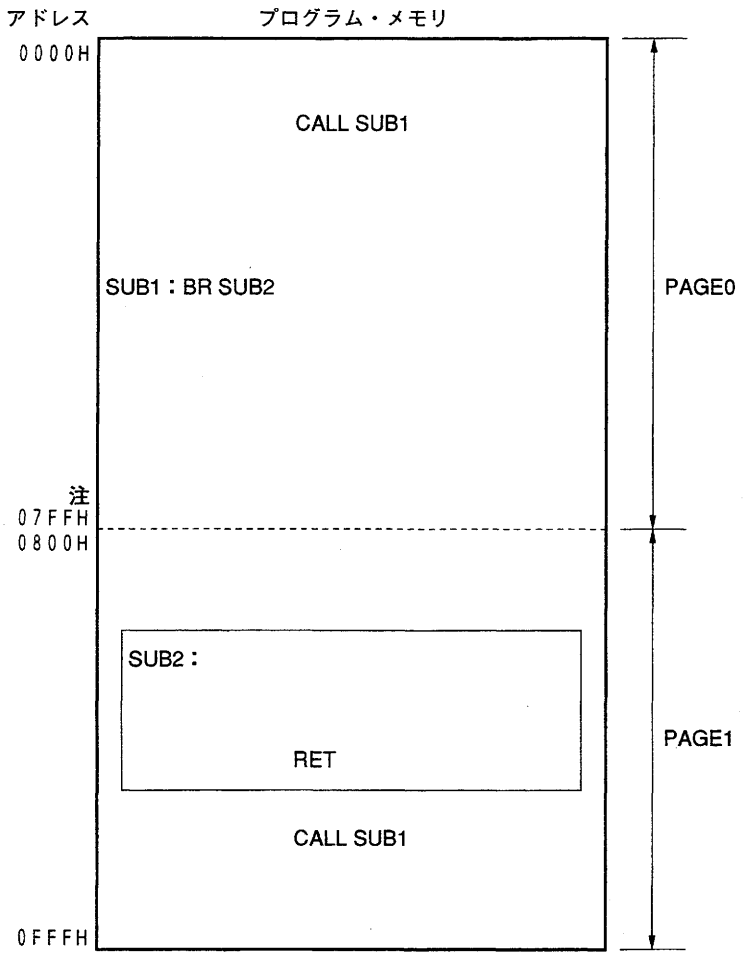


★　注　μPD17225のプログラム・メモリは0000H-07FFH番地です。

例2．サブルーチンの先頭番地がページ1内にある場合

図3－3に示すように，ページ0内に分岐命令（BR）を設け，この分岐命令を介してページ1にあるサブルーチン（SUB1）を呼び出す手法です。

図3－3　サブルーチンの先頭番地がページ1にあるとき



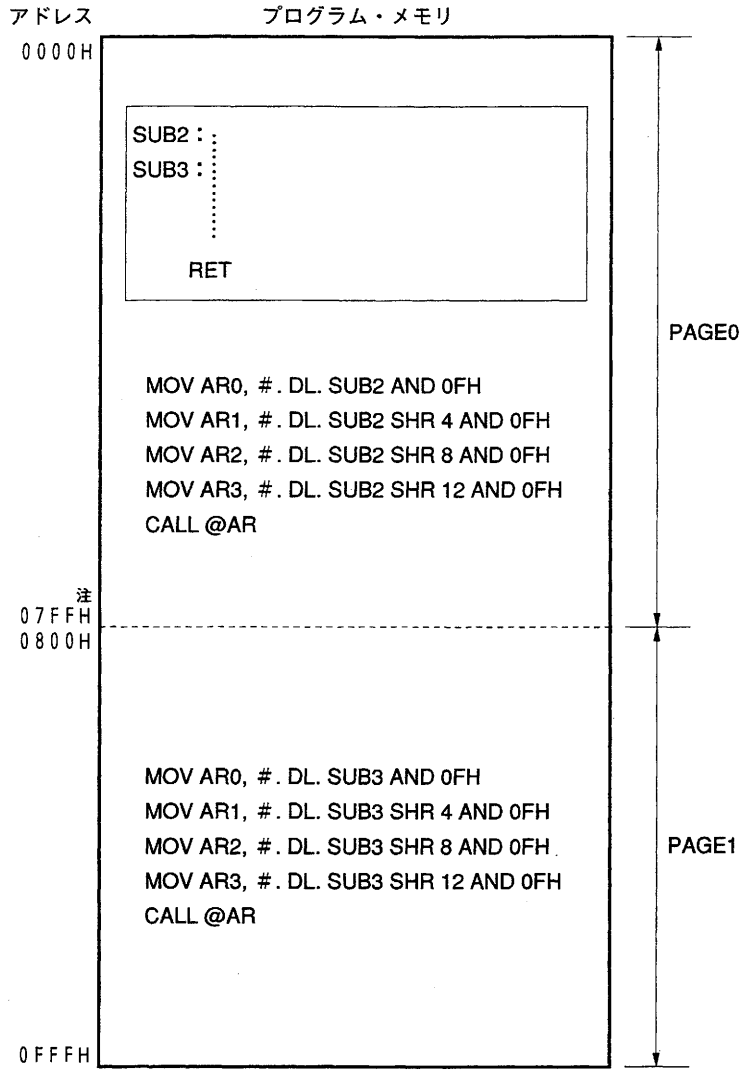
★　注　μPD17225のプログラム・メモリは0000H-07FFH番地です。

（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。

6.2　アドレス・レジスタ（AR）の項も参照してください。

図3－4　間接サブルーチン・コール（CALL @AR）



★　注　μPD17225のプログラム・メモリは0000H-07FFH番地です。

3.2.2　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。

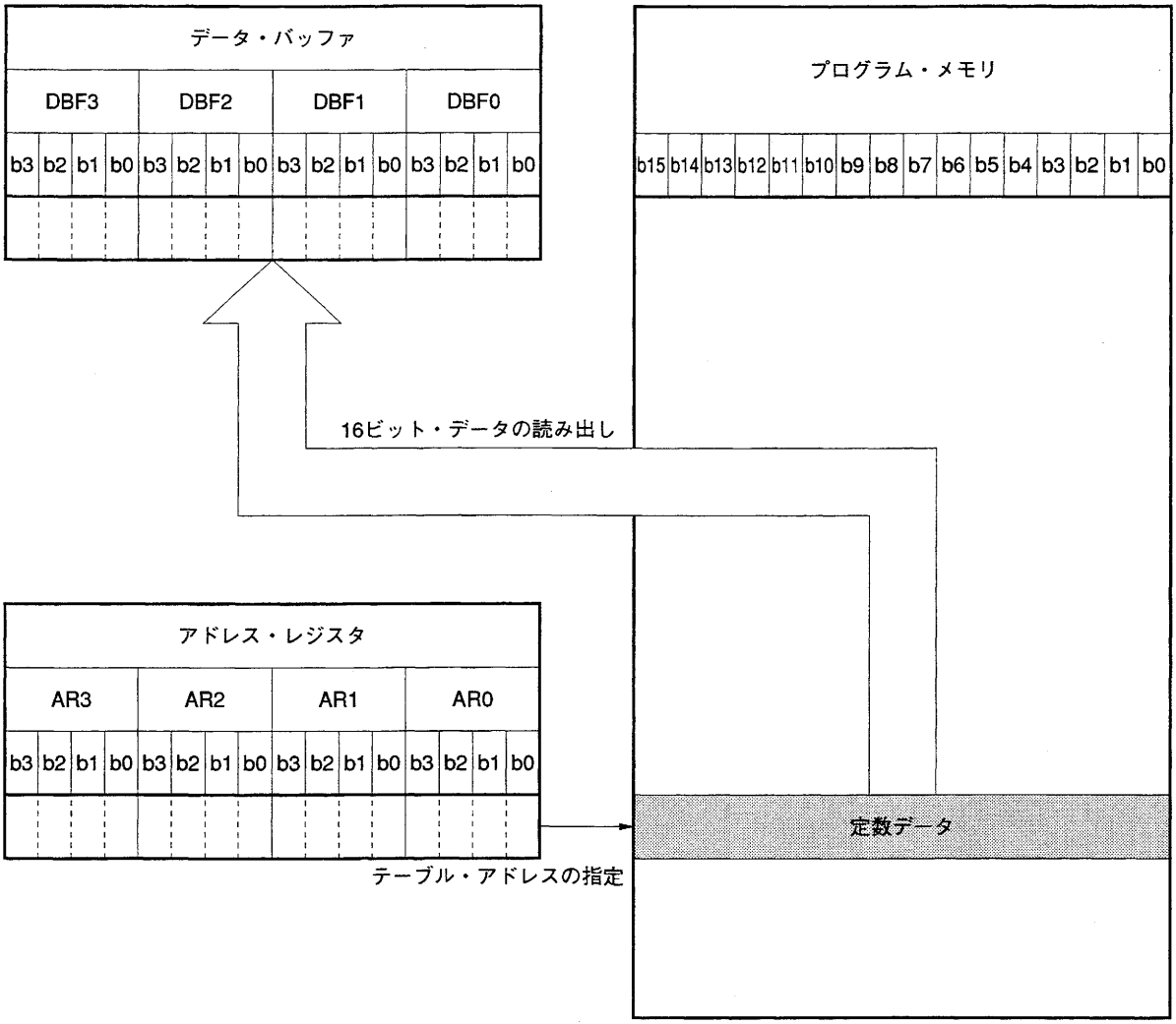
テーブル参照命令（MOVT DBF, @AR）を実行すると，アドレス・レジスタで指定されるプログラム・メモリ・アドレスの内容がデータ・バッファに格納されます。

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。

テーブル参照を行うときは，一時的にスタックが1レベル使用されるので，スタック・レベルに注意してください。

6.2　アドレス・レジスタ（AR），および4.1.2　データ・バッファ（DBF）の項も参照してください。

図3－5　テーブル参照（MOVT DBF, @AR）



（1）定数テーブル

以下に定数テーブルのテーブル参照プログラム例を示します。

例　定数データ・テーブルのOFFSET番目の値を読み出すプログラム

```
OFFSET   MEM    0.00H          ;オフセット・アドレスを格納しておく
         BANK0
         MOV    RPH, # 0       ;レジスタ・ポインタをロウ・アドレス7にする
         MOV    RPL, # 7 SHL 1
ROMREF:
         BANK0                 :テーブルの先頭アドレスを設定する
         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・アドレスを加算する
         ADDC   AR1, # 0
         ADDC   AR2, # 0
         ADDC   AR3, # 0
         MOVT   DBF, @AR       ;テーブル参照の実行

          ∫     プログラム

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

以下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　分岐先テーブルのOFFSET番目の値が示すアドレスに分岐するプログラム

```
OFFSET  MEM   0.00H          ;オフセット・アドレスを格納しておく
        BANK0
        MOV   RPH, #0        ;レジスタ・ポインタをロウ・アドレス7にする
        MOV   RPL, #7 SHL 1
ROMREF:
        BANK0                ;テーブルの先頭アドレスを設定する
        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・アドレスを加算する
        ADDC  AR1, #0
        ADDC  AR2, #0
        ADDC  AR3, #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
        DW    0400H
        DW    0800H
        DW    1000H
        DW    2000H
        DW    4000H
        DW    8000H
        END
```

〔メ　モ〕

第4章　データ・メモリ（RAM）

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

次にμPD172××サブシリーズのデータ・メモリ容量の一覧を示します。

★ 表4－1　μPD172××サブシリーズのデータ・メモリ

品　名	データ・メモリ容量
μPD17225	111×4ビット（BANK0）
μPD17226	
μPD17227	223×4ビット（BANK0, BANK1）
μPD17228	
μPD17P218	
μPD17201A	336×4ビット（BANK0-BANK2）
μPD17203A	
μPD17P203A	
μPD17204	
μPD17P204	
μPD17207	
μPD17P207	

4.1　データ・メモリの構成

図4－1にデータ・メモリの構成を示します。

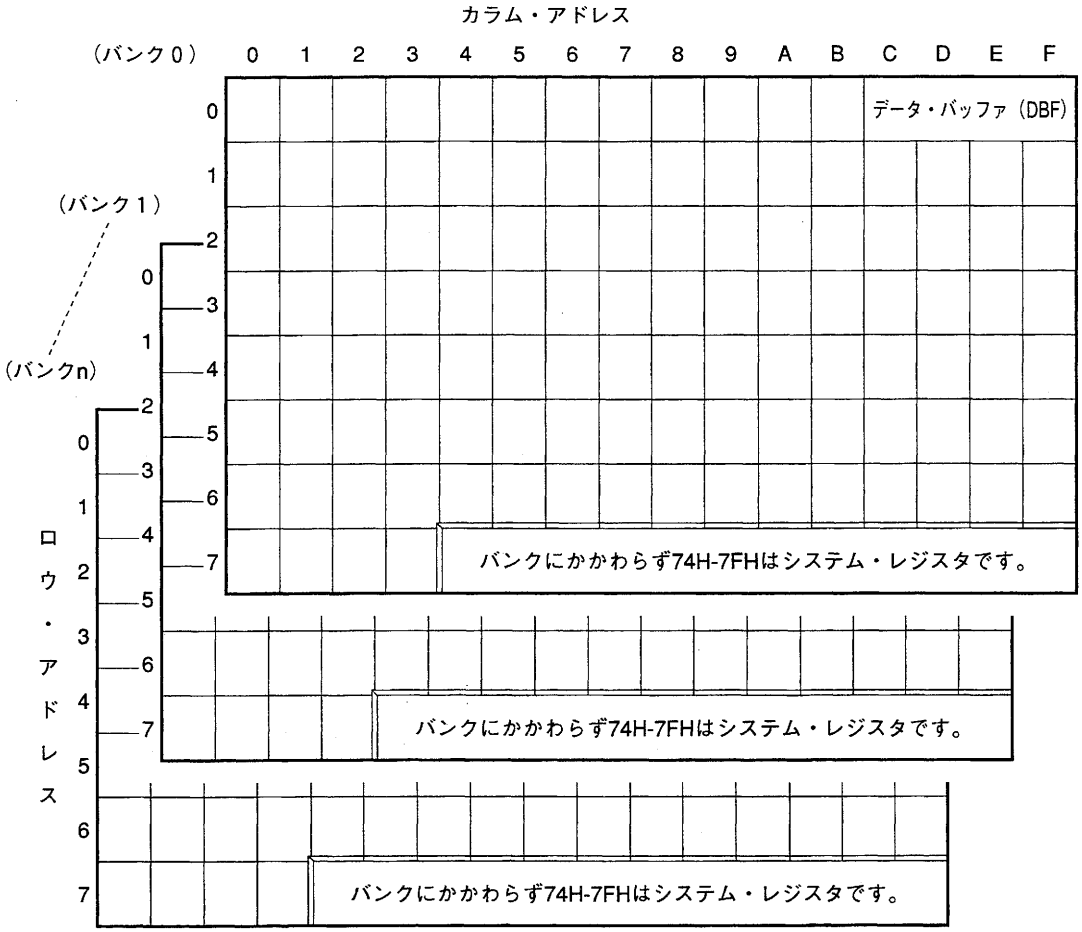
データ・メモリは“バンク”と呼ぶ概念で管理されています。

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。

データ・メモリは機能別に次の4.1.1-4.1.4に示すブロックに分けられます。

図4－1　データ・メモリの構成



4.1.1　システム・レジスタの構成

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4－2に構成を示します。

図4－2　システム・レジスタの構成

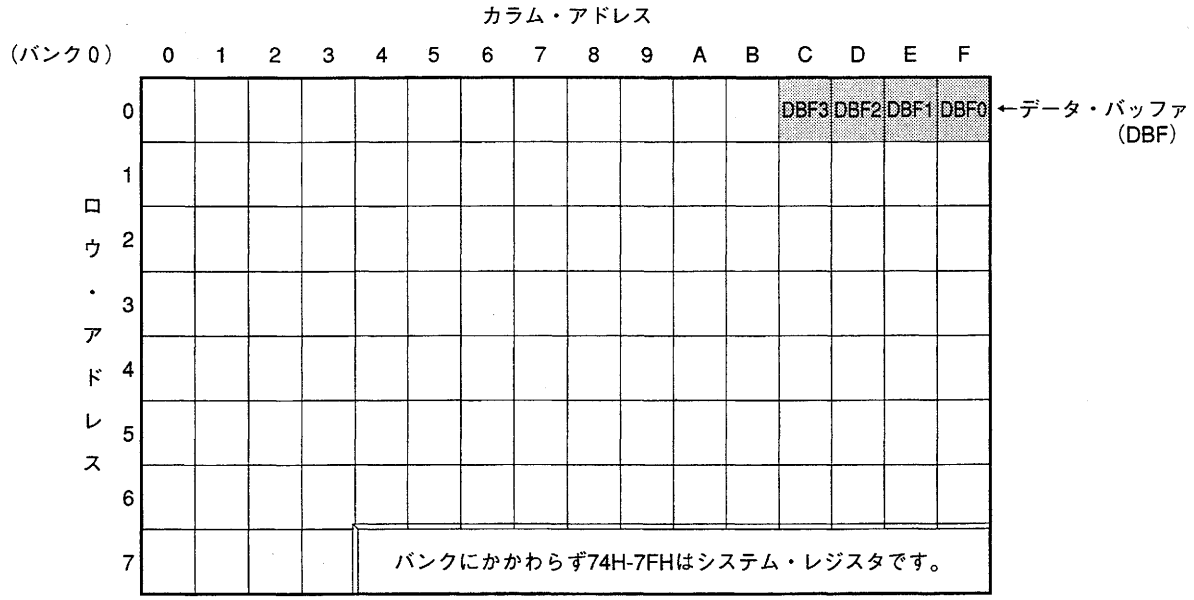
<table>
<tr><td colspan="12">システム・レジスタ（SYSREG）</td></tr>
<tr><td>アドレス</td><td>74H</td><td>75H</td><td>76H</td><td>77H</td><td>78H</td><td>79H</td><td>7AH</td><td>7BH</td><td>7CH</td><td>7DH</td><td>7EH</td><td>7FH</td></tr>
<tr><td>名　称
（記号）</td><td colspan="4">アドレス・レジスタ（AR）</td><td>ウインドウ・
レジスタ
（WR）</td><td>バンク・
レジスタ
（BANK）</td><td colspan="3">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）</td><td colspan="2">ジェネラル・レジスタ・
ポインタ（RP）</td><td>プログラム・
ステータス・
ワード
（PSWORD）</td></tr>
</table>

4.1.2　データ・バッファ（DBF）

データ・バッファ（DBF）はデータ・メモリのバンク0のアドレス0CH-0FHに割り当てられています。

データ・バッファは周辺ハードウエアとのデータ転送（PUT命令，GET命令）およびテーブル参照（MOVT命令）に使用するバッファです。

図4－3　データ・バッファ



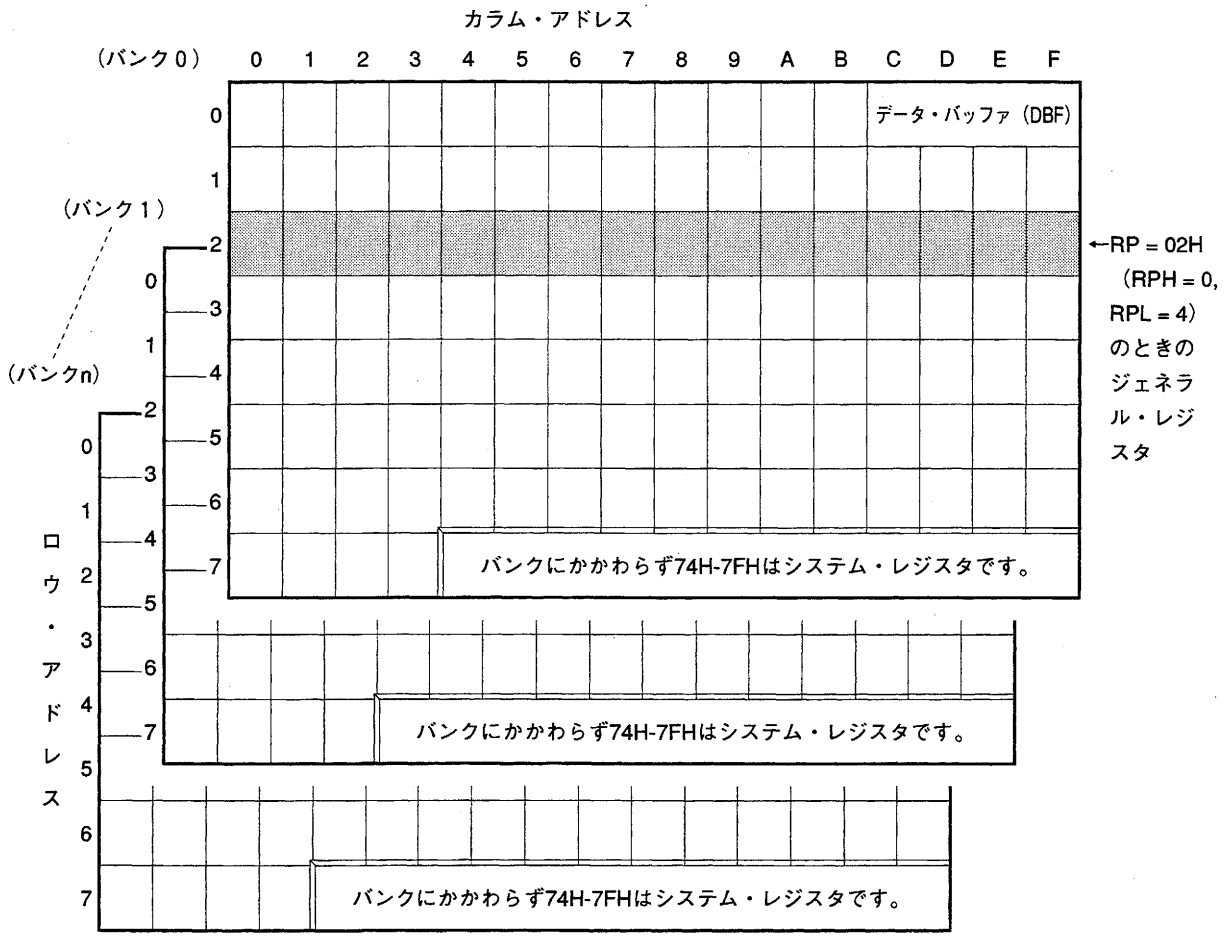
4.1.3　ジェネラル・レジスタ

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。

図4－4に構成を示します。

図4－4　ジェネラル・レジスタ



4.1.4　ポート・レジスタ

データ・メモリの各バンクのアドレス6FH-73Hに割り当てられた5ニブル（5×4ビット）で構成されます。

図4－5にμPD17225を例に構成を示します。

図4－5　ポート・レジスタの構成（μPD17225）

アドレス	6FH				70H				71H				72H				73H			
	P0E				P0A				P0B				P0C				P0D			
記　号	P0E3	P0E2	P0E1	P0E0	P0A3	P0A2	P0A1	P0A0	P0B3	P0B2	P0B1	P0B0	P0C3	P0C2	P0C1	P0C0	P0D3	P0D2	P0D1	P0D0

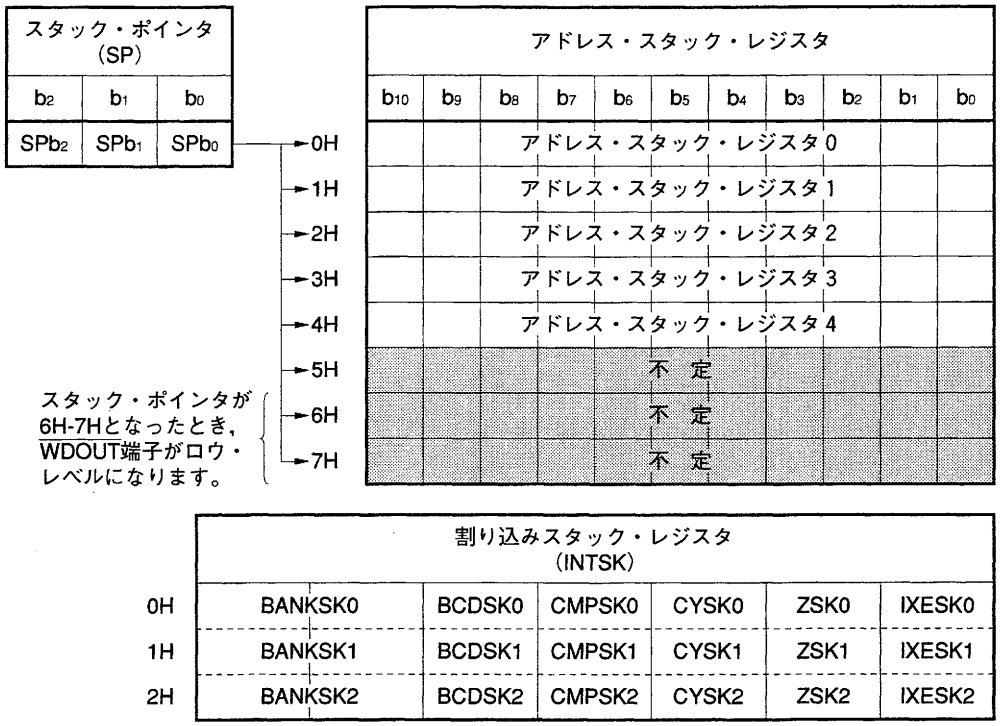
第5章　スタック

スタックはプログラム・カウンタを格納するアドレス・スタックと，システム・レジスタの内容を格納する割り込みスタックの2種類あります。アドレス・スタックは，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を退避するためのレジスタです。また，割り込みスタックは，割り込み受け付け時にシステム・レジスタの一部またはすべてを退避するためのレジスタです。

5.1　スタックの構成

図5－1に，μPD17207を例に，スタックの構成を示します。スタックは3ビットのバイナリ・カウンタであるスタック・ポインタ（SP）を1個と11ビットのアドレス・スタック・レジスタを5個と，7ビットの割り込みスタック・レジスタ3個より構成されています。

図5－1　スタックの構成（μPD17207）



備考　μPD17204，17P204のアドレス・スタック・レジスタは，0H-6Hまでの7個です。

5.2　スタックの機能

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

5.3　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，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[注]を越えるサブルーチン・コールや割り込みを実行すると，**$\overline{\text{WDOUT}}$端子がロウ・レベルになります。**

注　μPD17204,17P204は7レベルです。

5.4　割り込みスタック・レジスタ

μPD17207を例に，割り込みスタック・レジスタを説明します。

割り込みスタック・レジスタは，図5－1に示すように3×7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）およびバンク・レジスタ，計7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，3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　割り込みスタック・レジスタに退避後，PSWORD, BANKのすべてのビットは自動的に“0”にクリアされます。

5.5　スタック・ポインタ（SP）と割り込みスタック・レジスタ

スタック・ポインタ（SP）は図5－1に示したように5個のアドレス・スタック・レジスタのアドレスを指定する3ビットのバイナリ・カウンタです。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。

スタック・ポインタは表5－1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5－1　スタック・ポインタの動作

命　令	スタック・ポインタ（SP）の値	割り込みスタック・レジスタのカウンタ
CALL addr CALL @AR MOVT DBF, @AR （第1命令サイクル） PUSH AR	－1	0
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	0
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以上になったり，スタック・ポインタの値が0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・ポインタの値が－1されて7になると**$\overline{\text{WDOUT}}$端子がロウ・レベルになります。**

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。

リセット時にはスタック・ポインタは5になります。

備考　μPD17204,17P204のアドレス・スタック・レジスタは7個です。リセット時にはスタック・ポインタは7になります。

5.6　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

5.6.1-5.6.3におのおののスタック動作を示します。

5.6.1　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5－2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5－2　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プログラム・カウンタ（PC）の値を＋1 ②スタック・ポインタ（SP）の値を－1 ③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 ④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。

5.6.2　テーブル参照命令（MOVT DBF, @AR命令）

表5－3にテーブル参照命令実行時の動作を示します。

表5－3　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プログラム・カウンタ（PC）の値を＋1 ②スタック・ポインタ（SP）の値を－1 ③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 ④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	⑤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 ⑥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 ⑦スタック・ポインタ（SP）の値を＋1

5.6.3　割り込み受け付け時とリターン命令（RETI命令）

表5－4にμPD17207を例に割り込み受け付け時とリターン命令実行時のスタック動作を示します。

表5－4　割り込み受け付け時とリターン命令の動作（μPD17207）

命　令	動　　作
割り込み受け付け時	①プログラム・カウンタ（PC）の値を＋1 ただし，割り込み受け付け時の命令が分岐命令（BR命令）およびサブルーチン・コール命令（CALL命令）であるときは，分岐またはコールするプログラム・メモリ（ROM）のアドレスになります。 ②スタック・ポインタ（SP）の値を－1 ③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 ④割り込みスタック・レジスタへPSWORD（BCD, CMP, CY, Z, IXE）の各フラグとBANKの値を退避 ⑤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）とBANK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）の値を＋1

5.7　スタックのネスティング・レベルとPUSH命令およびPOP命令

スタック・ポインタ（SP）はサブルーチン・コール命令やリターン命令などで単純に＋1，－1される3ビットのカウンタとして動作しています。したがってスタック・ポインタの値が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，**$\overline{\text{WDOUT}}$端子がロウ・レベルになります。**

このような事態を避けるため，アドレス・スタック・レジスタを多く使用する場合は，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することにより対処します。

表5－5にPUSH命令およびPOP命令の動作を示します。

表5－5　PUSH命令およびPOP命令の動作

命　令	動　　作
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
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

〔メ　モ〕

第6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。

6.1　システム・レジスタの構成

図6－1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6－1に示すようにシステム・レジスタは，データ・メモリの74H-7FH番地にバンクに無関係に配置されています。つまり，どのバンクであっても74H-7FH番地には同一のシステム・レジスタが存在しています。

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6－1　システム・レジスタのデータ・メモリ上の配置

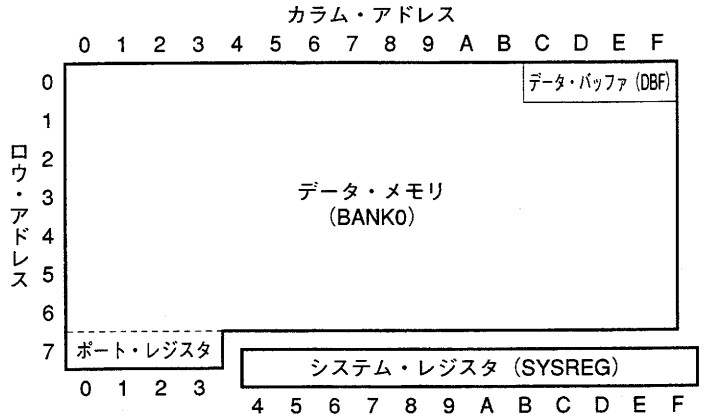


図6－2にμPD17204を例にシステム・レジスタの構成を示します。図6－2に示すようにシステム・レジスタは，次の7個のレジスタで構成されています。

- 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6－2　システム・レジスタの構成（μPD17204）

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)												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)			
																			データ・メモリ・ロウ・アドレス・ポインタ (MP)																									
記　号	AR3				AR2				AR1				AR0				WR	BANK	IXH				IXM				IXL				RPH				RPL				PSW			
																			MPH				MPL																			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	0	0	0														←	0	0	MPE	0	0	(IX)								0	0							BCD	CMP	CY	Z	IXE
	← (AR)															→		← (BANK)					← (MP)				→				← (RP)				→							
リセット時の初期値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	不定	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.2　アドレス・レジスタ（AR）

6.2.1　アドレス・レジスタの構成

図6－3にμPD17226を例にアドレス・レジスタの構成を示します。

図6－3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4ビットは常に0に固定されているために実際は12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6－3　アドレス・レジスタの構成（μPD17226）

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												
リセット	0				0				0				0			

★　図6－4　μPD172××サブシリーズのアドレス・レジスタ

品　名	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
μPD17225	←						(AR)						→
	0	0											
μPD17201A	←						(AR)						→
μPD17203A μPD17207 μPD17226	0												
μPD17204	←						(AR)						→
μPD17227 μPD17228													

6.2.2　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5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）

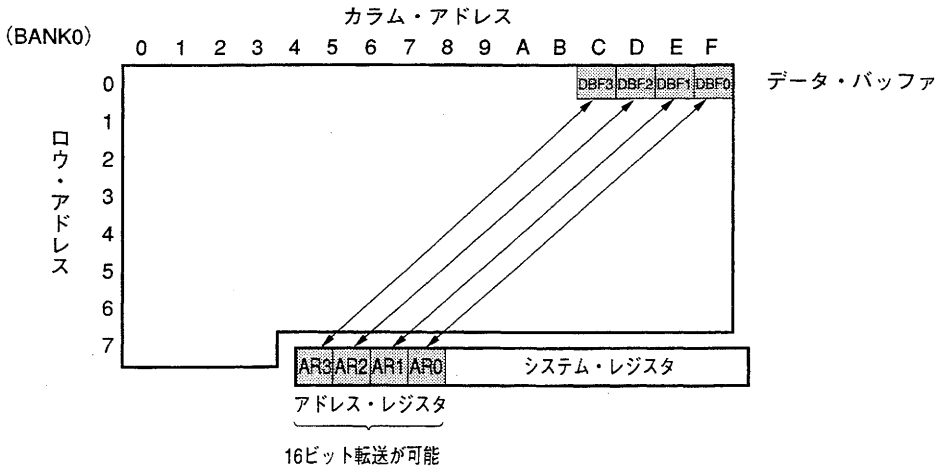
CALL @AR命令を実行することにより，アドレス・レジスタの値をプログラム・メモリ・アドレスとしてサブルーチンを呼び出すことができます。

（5）周辺ハードウエア・レジスタとしてのアドレス・レジスタ

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

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

図6－5　周辺回路としてのアドレス・レジスタ



6.3　ウインドウ・レジスタ（WR）

6.3.1　ウインドウ・レジスタの構成

図6－6にウインドウ・レジスタの構成を示します。

図6－6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$\overline{\text{RESET}}$端子を用いたときは以前の状態を保持しています。

図6－6　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			

6.3.2　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専用命令である“PEEK WR，rf”および“POKE rf，WR”で行います。

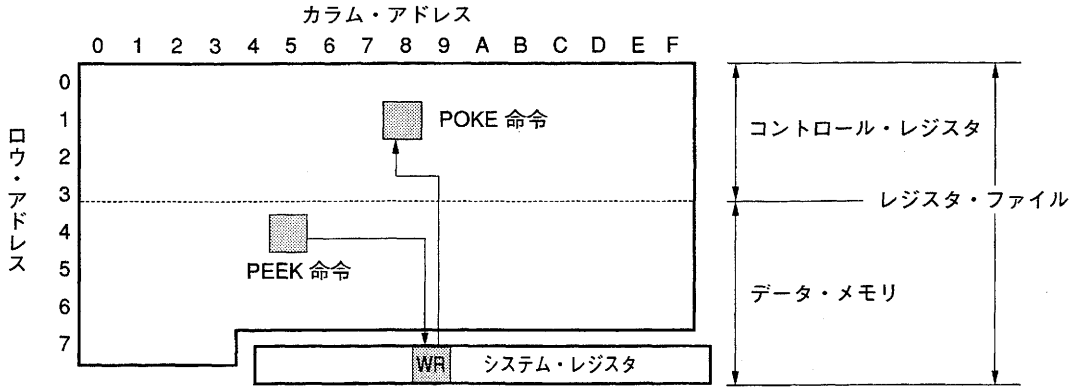
（1）PEEK WR，rf命令

図6－7に示すように，“PEEK WR，rf”命令により，rfで指定されたレジスタ・ファイルの内容をウインドウ・レジスタに転送します。

（2）POKE rf，WR命令

図6－7に示すように，“POKE rf，WR”命令により，ウインドウ・レジスタの内容をrfで指定されるレジスタ・ファイルに転送します。

図6－7　ウインドウ・レジスタの操作例



6.4　バンク・レジスタ（BANK）

図6－8にμPD17207を例に，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。μPD17207はバンクが3つあるため，上位2ビットが0に固定されています。

割り込みが受け付けられると，割り込みスタック・レジスタに退避されます。なお，退避後BANKは“0”にクリアされます。

図6－8　バンク・レジスタの構成（μPD17207）

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 (BANK)		
リセット	0			

★ 図6－9　μPD172××サブシリーズのバンク・レジスタ

品　名	b_3	b_2	b_1	b_0
μPD17225	←	(BANK)		→
μPD17226	0	0	0	0
μPD17201A				
μPD17203A	0	0		
μPD17P203A				
μPD17204	←	(BANK)		→
μPD17P204				
μPD17207				
μPD17P207				
μPD17227	0	0	0	
μPD17228	←	(BANK)		→
μPD17P218				

6.5　インデクス・レジスタ（IX）

インデクス・レジスタは，データ・メモリの操作命令に対してデータ・メモリのアドレス修飾を行うときに使用します。

インデクス・レジスタの構成をμPD17201Aを例に，図6－10に示します。

図6－10に示すようにインデクス・レジスタはシステム・レジスタのIXH（7AH），IXM（7BH），IXL（7CH）の計12ビットから構成されています。なお，7AHのb_2, b_1は0に固定されています。また，7AHの最上位ビットはメモリ・ポインタ・イネーブル・フラグ（MPE）になっています。

メモリ・ポインタ・イネーブル・フラグは，MOV @r, m命令などの命令において，オペランドが@rで表されているレジスタのアドレスをデータ・メモリ・ロウ・アドレス・ポインタ（MP：メモリ・ポインタ（MPHの下位3ビットとMPLの4ビット））の内容でアドレス修飾するときに使用します。

インデクス・イネーブル・フラグ（IXE）はPSWの最下位ビットに割り当てられています。これはADD r, m命令などのデータ・メモリ操作命令において，オペランドがmで表されているデータ・メモリのアドレスとインデクス・レジスタ（IX）の内容とをOR演算することによって，mで表されるデータ・メモリのアドレスをアドレス修飾するときに使用します。また，MPE＝0のときは，ジェネラル・レジスタ間接転送命令（MOV @r, m命令など）のオペランドが@rで表されている側のレジスタ・アドレスもIXHおよびIXMの内容でアドレス修飾されます。

図6－10　インデクス・レジスタの構成（μPD17201A）

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）															
シンボル名	IXH				IXM				IXL				PSW			
	MPH				MPL											
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													
リセット時の値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.1　インデクス・レジスタとデータ・メモリ・ロウ・アドレス・ポインタの機能

（1），（2）にインデクス・レジスタとデータ・メモリ・ロウ・アドレス・ポインタの機能を示します。

（1）インデクス・レジスタ

インデクス・レジスタは，データ・メモリ操作命令を実行すると命令で指定されたデータ・メモリのバンクとアドレスをインデクス・レジスタの内容によりアドレス修飾します。

ただし，インデクス・レジスタによるアドレス修飾はインデクス・イネーブル・フラグ（IXE）がセットされているときのみ有効になります。

アドレス修飾の方法はデータ・メモリのバンクとアドレスをインデクス・レジスタの内容とOR演算し，その演算結果で指定されるアドレス（実アドレスと呼ぶ）のデータ・メモリに対して命令を実行します。

インデクス・レジスタによるアドレス修飾は，すべてのデータ・メモリ操作命令を対象にできます。

また，以下の命令はインデクス・レジスタの修飾対象となりません。

MOVT	DBF, @AR	BR	addr	INC	AR	EI
PEEK	WR, rf	BR	@AR	INC	IX	DI
POKE	rf, WR	CALL	addr	RORC	r	
GET	DBF, p	CALL	@AR	STOP	s	
PUT	p, DBF	RET		HALT	h	
PUSH	AR	RETSK		NOP		
POP	AR	RETI				

（2）データ・メモリ・ロウ・アドレス・ポインタ

データ・メモリ・ロウ・アドレス・ポインタはジェネラル・レジスタ間接転送命令（MOV @r, mおよびMOV m, @r）を実行すると，間接転送先のアドレスをデータ・メモリ・ロウ・アドレス・ポインタの内容でアドレス修飾します。

ただし，データ・メモリ・ロウ・アドレス・ポインタによるアドレス修飾はデータ・メモリ・ロウ・アドレス・ポインタ・イネーブル・フラグ（メモリ・ポインタ・イネーブル・フラグ；MPE）がセット（1）されているときのみ有効になります。

また，ジェネラル・レジスタ間接転送命令以外ではアドレス修飾の対象になりません。

アドレス修飾の方法は間接転送先のバンクとロウ・アドレスをデータ・メモリ・ロウ・アドレス・ポインタの内容で置き換えます。

表6－1にインデクス・レジスタとデータ・メモリ・ロウ・アドレス・ポインタによるデータ・メモリ・アドレスの修飾および間接転送アドレスの修飾を示します。

また，6.5.2-6.5.4にインデクス・レジスタとデータ・メモリ・ロウ・アドレス・ポインタによるデータ・メモリのアドレス修飾の各動作について説明します。

表6－1　インデクス・レジスタとデータ・メモリ・ロウ・アドレス・ポインタによるデータ・メモリ・アドレスの修飾

IXE	MPE	rで指定されるジェネラル・レジスタ・アドレス			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	バンク	ロウ・アドレス	カラム・アドレス
		b3 b2 b1 b0	b2 b1 b0	b3 b2 b1 b0	b3 b2 b1 b0	b2 b1 b0	b3 b2 b1 b0	b3 b2 b1 b0	b2 b1 b0	b3 b2 b1 b0
0	0	RP		r	BANK		m	BANK	m_R	(r)
0	1		同上			同上		MP		(r)
1	0		同上		BANK		m	BANK	m_R	(r)
					Logical OR			Logical OR		
					IX			IXH	IXM	
1	1	設定禁止								

	アドレス修飾される命令群			
加減算	ADD ADDC SUB SUBC	r	m	
			m, #n4	
論理	AND OR XOR	r	m	
			m, #n4	
比較	SKE SKGE SKLT SKNE		m, #n4	
判断	SKT SKF		m, #n	
転送	LD ST	r	m	
	MOV		m, #n4	
		@r	m	間接転送アドレス

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-ビット8
　IXM：インデクス・レジスタのビット7-ビット4
　IXL：インデクス・レジスタのビット3-ビット0
m　：m_R,mc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）
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
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
（X）：Xでアドレスされる内容
　　X：m, rなどのダイレクト・アドレス
　　：BANKなどのレジスタ

6.5.2　MPE＝0，IXE＝0のとき（データ・メモリ修飾なし）

表6－1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6－11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

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6－11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
```

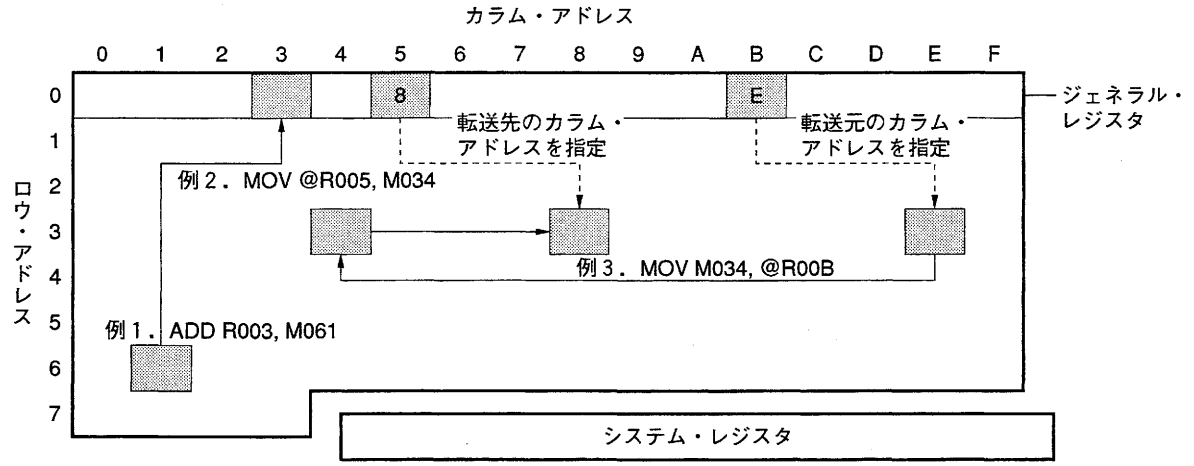
上の命令を実行すると，図6－11に示すようにデータ・メモリM034へ，アドレス3EH番地のデータ・メモリの内容を転送します。

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6－11　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

6.5.3　MPE＝1，IXE＝0のとき（ななめ間接転送）

表6－1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     MPH, #1000B     ; MPE←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6－12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6.5.2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（6.5.2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）。

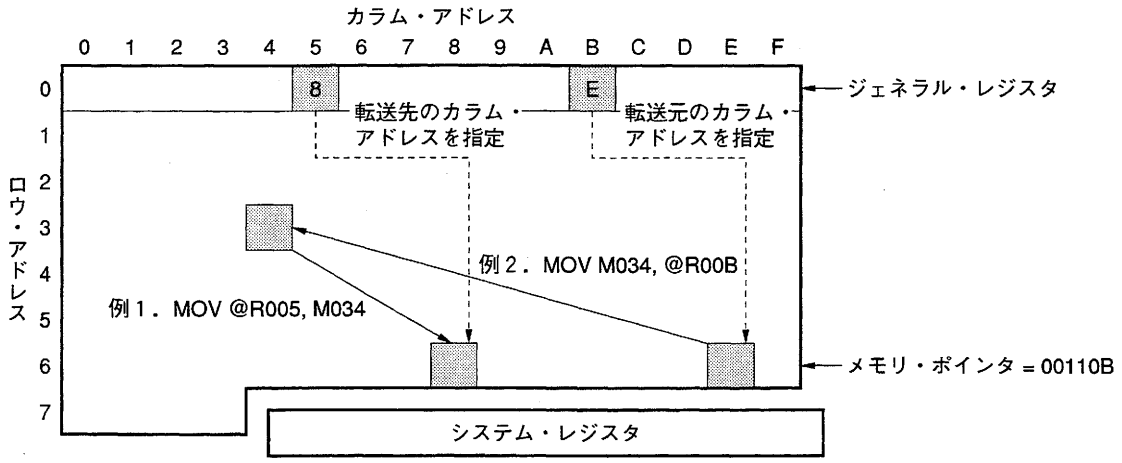
したがって，MPE＝1とすることによりジェネラル・レジスタ間接転送をななめに行うことが可能になります。

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6－12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6－12　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

6.5.4　MPE＝0，IXE＝1のとき（データ・メモリ・アドレス・インデクス修飾）

表6－1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
```

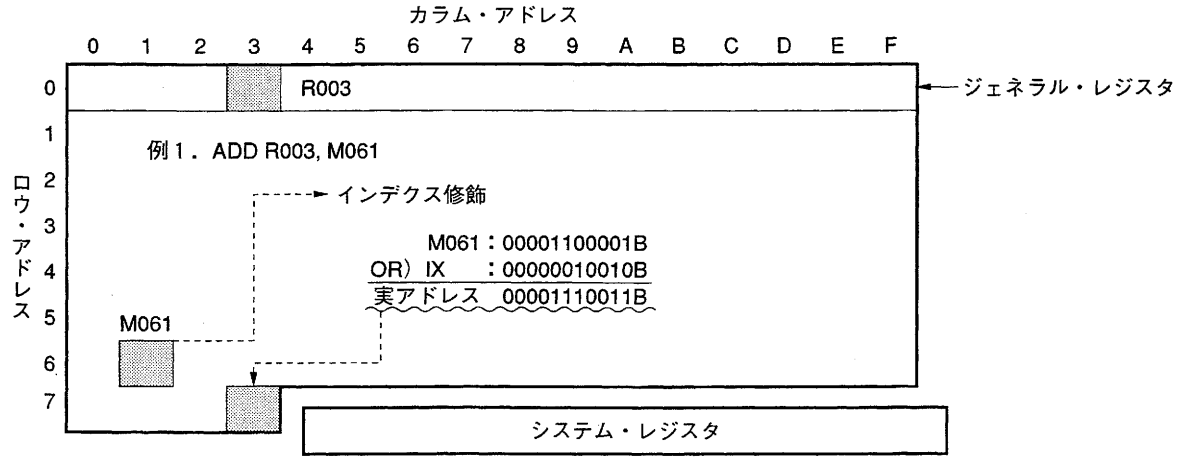
例1の命令を実行すると，図6－13に示すようにアドレスが73H番地のデータ・メモリ（実アドレス）の内容とジェネラル・レジスタR003（アドレスの03H番地）の内容を加算し，結果をジェネラル・レジスタR003へ格納します。

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（6.5.2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6－13　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

BANK0でジェネラル・レジスタがロウ・アドレス0であるとする。

```
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6－14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。

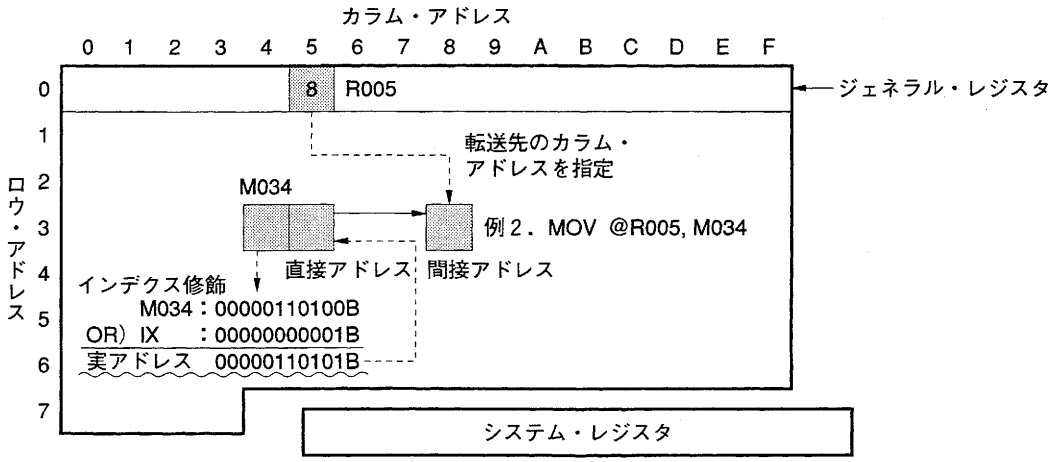
すなわち，IXE＝1のときに“MOV @r, m”命令を実行すると，“m”で指定されるデータ・メモリ・アドレス（直接アドレス）をインデクス・レジスタの内容でアドレス修飾し，“@r”で指定される間接アドレスのバンクとロウ・アドレスもインデクス・レジスタで修飾されます。

“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6.5.2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（6.5.2例3では直接アドレスは修飾されない）。

図6－14　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6－15に示すように，いまAという8ビットのデータを1次元で定義したとする。

このとき

A（N）＝A（N）＋4　（0≦N≦15）

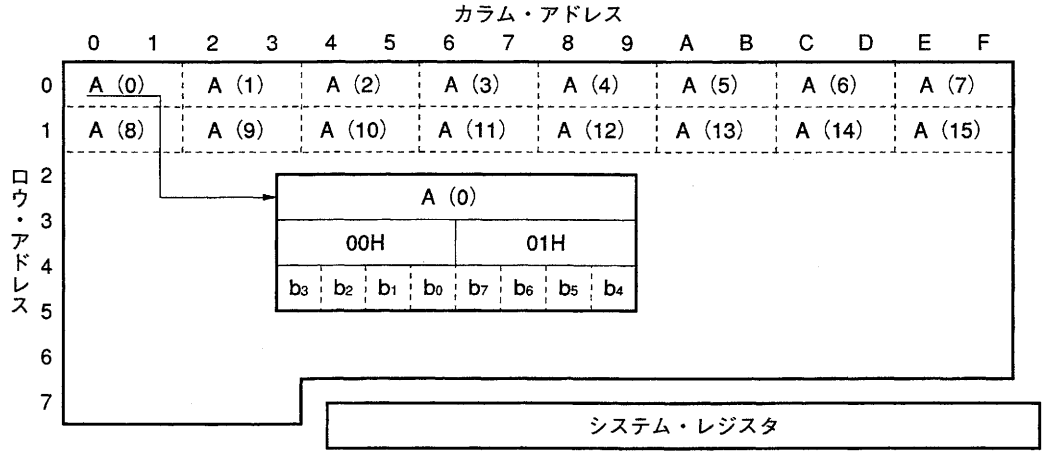
という演算を行うためには以下の命令を実行すればよい。

ジェネラル・レジスタはロウ・アドレス7にあるとする。

```
M000    MEM 0.00H
M001    MEM 0.01H
MOV     IXH, # 0                ; IX←2N
MOV     IXM, #N SHR 3           ; なぜなら配列要素は8ビットであるためインデクス修飾するデ
MOV     IXL, #N SHL 1 AND 0FH   ; ータ・メモリ・アドレスをシフトする。
OR      PSW, #.DF.IXE AND 0FH   ; IXE← 1
ADD     M000, # 4               ; IXでインデクス修飾されるデータ・メモリM000およびM001に
ADDC    M001, # 0               ; 4を加える。すなわちA（N）で指定される8ビットの配列に4
                                ; を加える。
```

上記の例のように配列A（N）のNの指定をするには，インデクス・レジスタにNの2倍の値を指定すればよいことになります。

図6－15　MPE＝0，IXE＝1のときの動作例（配列の処理）



6.6　ジェネラル・レジスタ・ポインタ（RP）

6.6.1　ジェネラル・レジスタ・ポインタの構成

図6－16にμPD17201Aを例に，ジェネラル・レジスタ・ポインタの構成を示します。

図6－16　ジェネラル・レジスタ・ポインタの構成（μPD17201A）

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		(RP)				
リセット時	0				0			

図6－16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番地の上位3ビットは常に0に固定されているため実際には7DHの最下位ビットと7EH番地の上位3ビットの計4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。

6.6.2　ジェネラル・レジスタ・ポインタの機能

μPD17201Aを例に，ジェネラル・レジスタ・ポインタの機能を説明します。

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6－17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5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BANK0-2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。

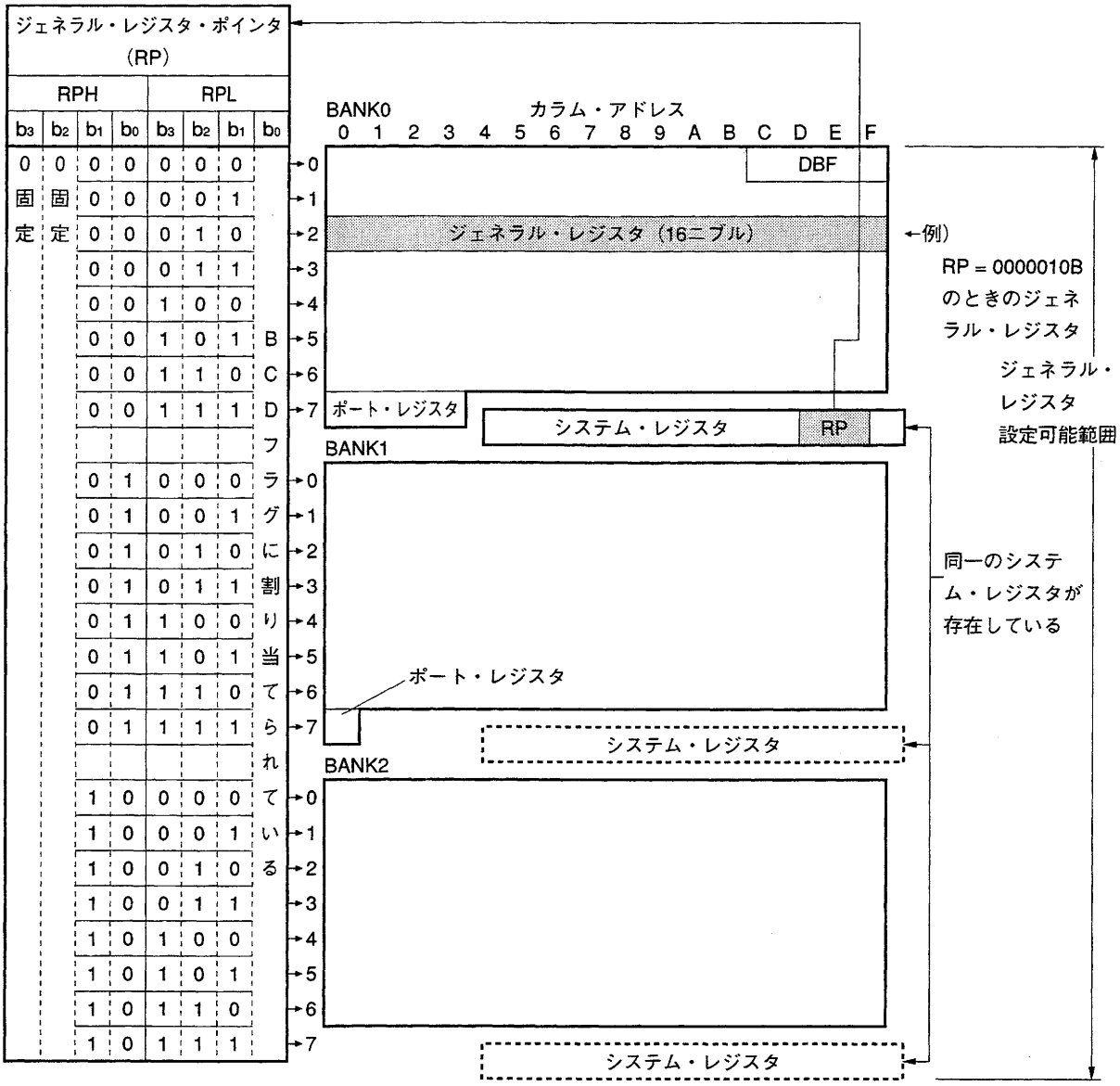
たとえば，

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。

詳しくは**第7章　ジェネラル・レジスタ（GR）**を参照してください。

図6－17　ジェネラル・レジスタの構成（μPD17201A）



6.7　プログラム・ステータス・ワード（PSWORD）

6.7.1　プログラム・ステータス・ワードの構成

図6－18にプログラム・ステータス・ワードの構成を示します。

図6－18　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6－18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。

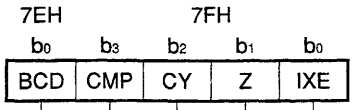
リセット時はすべて0にクリアされます。

割り込みが受け付けられると，割り込みスタック・レジスタに退避されます。なお，退避後，PSWORDはすべて“0”にクリアされます。

6.7.2　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6－19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6－19　プログラム・ステータス・ワードの機能概要



インデクス・イネーブル・フラグ	このフラグがセットされるとインデクス修飾が可能です。
ゼロ・フラグ	算術演算の結果が“0”以外のときにリセットされる。セットの条件はCMPフラグの内容により異なる。 （1）CMP＝0のとき 算術演算の結果が“0”のときにセットされる。 （2）CMP＝1のとき Z＝1でかつ実行した算術演算の結果が“0”のときのみセットされる。
キャリー・フラグ	加算命令実行時のキャリー発生，または減算命令実行時のボロー発生でセットされ，それらが発生しないときにリセットされる。 またRORC命令が実行されるジェネラル・レジスタの最下位ビットが“1”のときセットされ，“0”のときにはリセットされる。
コンペア・フラグ	このフラグがセットされているとき，算術演算の結果は，データ・メモリには格納されない。CMPフラグは，SKT, SKF命令を実行すると自動的にリセットされる。
BCDフラグ	このフラグがセットされるとすべての算術演算は10進（BCD）で行われ，リセットされると2進で行われる。

6.7.3　インデクス・イネーブル・フラグ（IXE）

IXEフラグは，データ・メモリ操作命令実行時にデータ・メモリのアドレスをインデクス修飾するために使用します。

IXEフラグをセット（1）すると，命令で指定されたデータ・メモリ・アドレスとインデクス・レジスタ（IX）の内容をOR演算し，そのOR演算結果を実アドレスとするデータ・メモリに対して命令を実行します。

詳しくは6.5　インデクス・レジスタ（IX）を参照してください。

6.7.4　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6－2のようになります。

表6－2　コンペア・フラグ（CMP）の状態とゼロ・フラグ（Z）のセット，リセット条件

条　　件	Zフラグの状態	
	CMPフラグが0のとき	CMPフラグが1のとき
リセット時	リセット	CMPフラグとともにリセット
データ・メモリ操作命令でZフラグに直接“0”が書き込まれたとき	リセット	リセット
データ・メモリ操作命令でZフラグに直接“1”が書き込まれたとき	セット	セット
算術演算の結果が“0”以外になったとき	リセット	リセット
算術演算の結果が“0”になったとき	セット	以前のZフラグの状態保持

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。

6.7.5　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。

6.7.6　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。

6.7.7　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。

6.8　システム・レジスタ使用時の注意

6.8.1　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（RA17K）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（RA17K）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6－2　システム・レジスタの構成（μPD17204）**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。

```
例1．MOV    34H, #0101B  ；オペランドにデータ・メモリ・アドレスを34Hや76Hと記述すると，アセ
     MOV    76H, #1010B  ；ンブラでエラーとなる。
     M037   MEM 0.37H    ；汎用データ・メモリは，MEM疑似命令でデータ・メモリ・アドレスをシ
     MOV    M037, #0101B ；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6H）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・ファイルにて定義さ
                         ；れている。
```

また，アセンブラ（RA17K）を使用する場合はアセンブラ内部に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が定義されています。この予約語はMPE, BCD, CMP, CY, Z, IXEとなります。

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）
```

6.8.2　“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6－2　システム・レジスタの構成（μPD17204）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”で固定されているデータに“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（RA17K）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・エラーは発生ぜず，インサーキット・エミュレータ上で命令が実行されたときに初めてエミュレータ・エラーを発生します。

★　アセンブラ（RA17K）でエラーが発生しない理由としては，レジスタ間接転送等が行われるときは，データ・メモリ操作命令の対象となるデータ・メモリ・アドレスを判断できない等の理由によります。

したがって，アセンブラ・エラーを発生するのは次の場合のみです。

“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，“BANKn”命令を使用するときは，明らかにシステム・レジスタのバンク・レジスタを操作したいという意志のもとで使用されていると判断するためです。

第7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。

7.1　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μPD17201Aを例に，図7－1に示します。

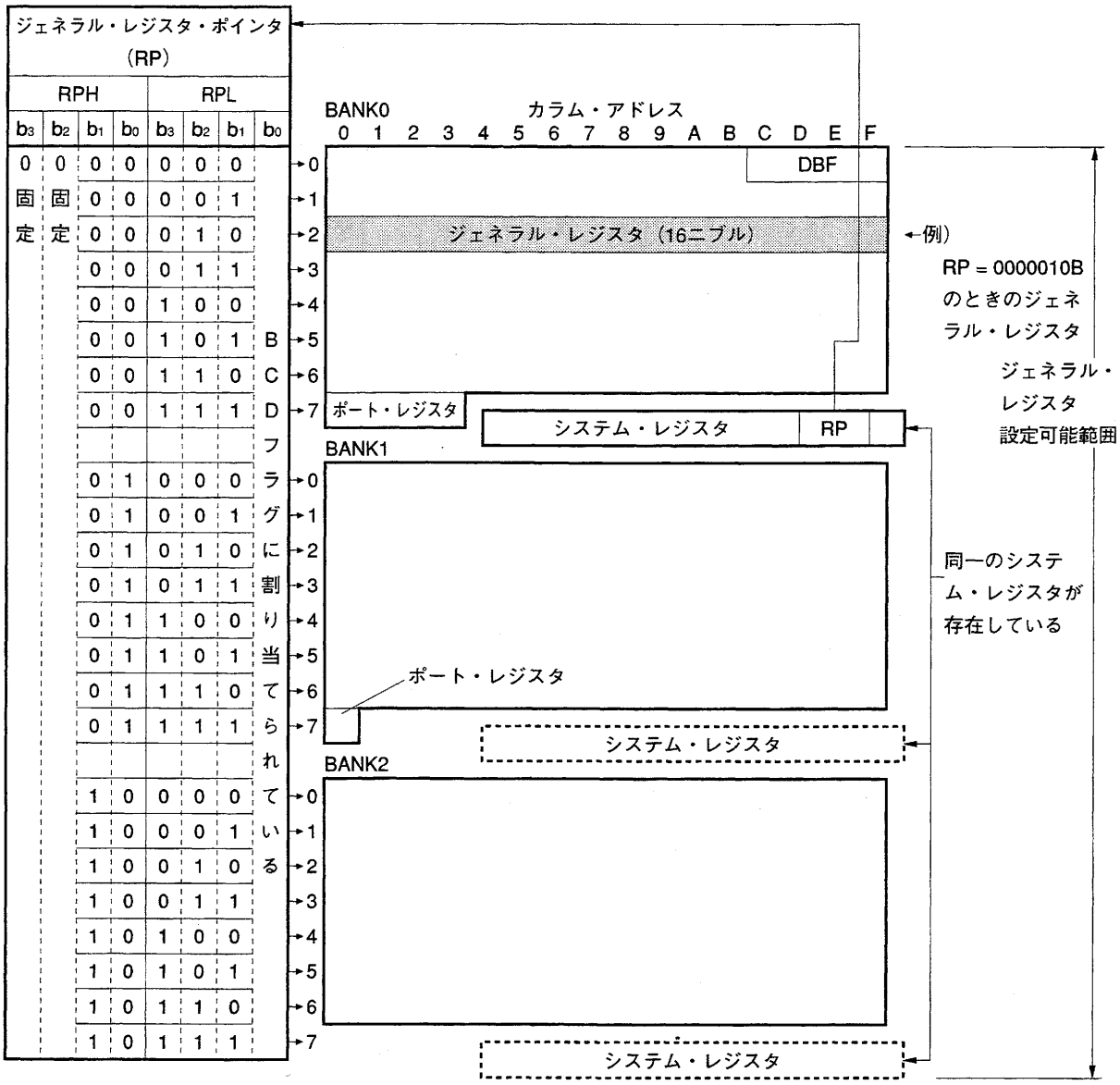
図7－1に示すように，データ・メモリ上で同一ロウ・アドレスである16ニブル（16×4ビット）をジェネラル・レジスタとして使用できます。

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5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。

7.2　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7－1　ジェネラル・レジスタの構成（μPD17201A）



第8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。

8.1　レジスタ・ファイルの構成

8.1.1　レジスタ・ファイルの構成

図8－1にレジスタ・ファイルの構成を示します。

図8－1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。

8.1.2　レジスタ・ファイルとデータ・メモリ

図8－2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8－2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8－1　レジスタ・ファイルの構成

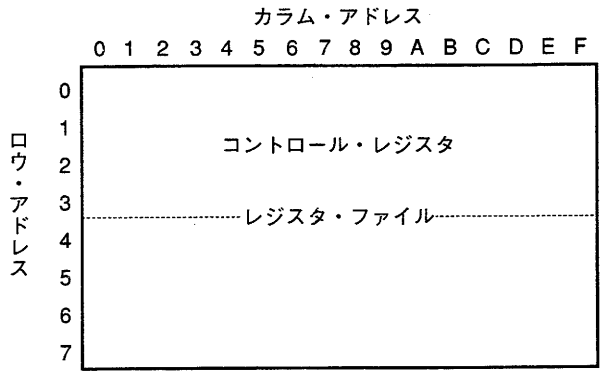
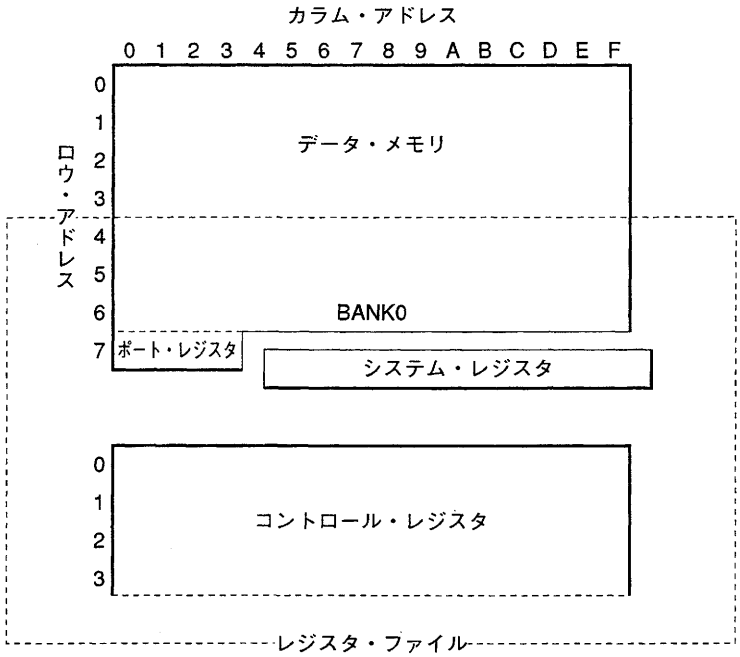


図8－2　レジスタ・ファイルとデータ・メモリの関係



8.2　レジスタ・ファイルの機能

8.2.1　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，8.2.3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。

8.2.2　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。

★ 表8－1　μPD172××サブシリーズの内蔵周辺ハードウエア一覧

製品名 / 周辺ハードウエア	17201A 17207	17203A 17204	17225 17226 17227 17228
スタック	○	○	○
タイマ	○	○	○
割り込み	○	○	○
キャリア・ジェネレータ	○	○	○
リモコン受信アンプ		○	
汎用ポート	○	○	○
A/Dコンバータ	○		
シリアル・インタフェース	○	○	
LCDドライバ	○		

注意　周辺ハードウエアの中には，データ転送をデータ・バッファ（DBF）を介して行うものもあります（第9章　データ・バッファ（DBF）参照）。

8.2.3　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はBANK0の80H-BFH
      RF70    MEM   0.70H   ；として定義する必要があります。詳細については8.4　レジスタ・ファ
      RF73    MEM   0.73H   ；イル使用時の注意を参照してください。
      BANK0
    ①PEEK     WR, RF11      ;

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

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

    ⑥ST       WR, M032      ;
```

図8－3に動作例を示します。

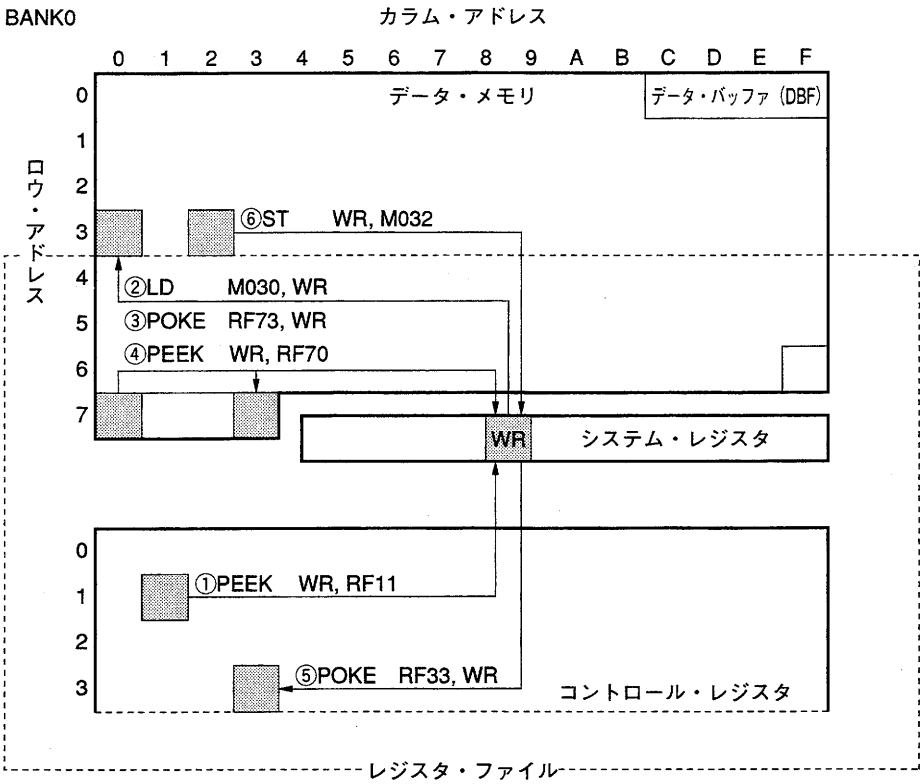
図8－3に示すように，コントロール・レジスタ（00H-3FH番地）は“PEEK WR, rf”および“POKE rf, WR”命令が実行されると，“rf”でアドレス指定されるレジスタ・ファイルの内容をウインドウ・レジスタに対してのみ読み込みまたは書き込みを行います。

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（**8.4.2　レジスタ・ファイルのシンボル定義と予約語**参照）。

図8－3　PEEK，POKE命令によるレジスタ・ファイルのアクセス



8.3　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。

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。

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。

8.4　レジスタ・ファイル使用時の注意

8.4.1　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上，17Kシリーズのアセンブラ（RA17K）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（RA17K）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（バッチ処理等で操作したとき）

読み出し専用レジスタに書き込みを行っても何も変化しません。“エラー”は発生しません。

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しません。“エラー”は発生しません。

8.4.2　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（RA17K）を使用するうえでは，“PEEK WR, rf”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する必要があります。

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    MEM0.71H   ; 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    MEM1.71H   ; データ・メモリと重なっているレジスタ・ファイル
RF02    MEM0.82H   ; 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（RA17K）を使用する場合は，フラグ型シンボル操作命令として次のマクロ命令があらかじめアセンブラに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
★　INITFLGX：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，次の例3に示すようにレジスタ・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（RA17K）上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唯一データ・メモリ型として“SP”で定義されています。このため，予約語によるフラグの操作命令は，使用できません。

例3．INITFLG　WDTRES　；イニシャライズ
（SET1　WDTRES　；フラグのセット）

マクロ展開

```
PEEK   WR, .MF.WDTRES SHR4
OR     WR, #.DF.WDTRES AND 0FH
POKE   .MF.WDTRES SHR4, WR
```

第9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・バッファ上に読み出すことができます。

9.1　データ・バッファの構成

図9－1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9－1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9－1　データ・バッファの配置

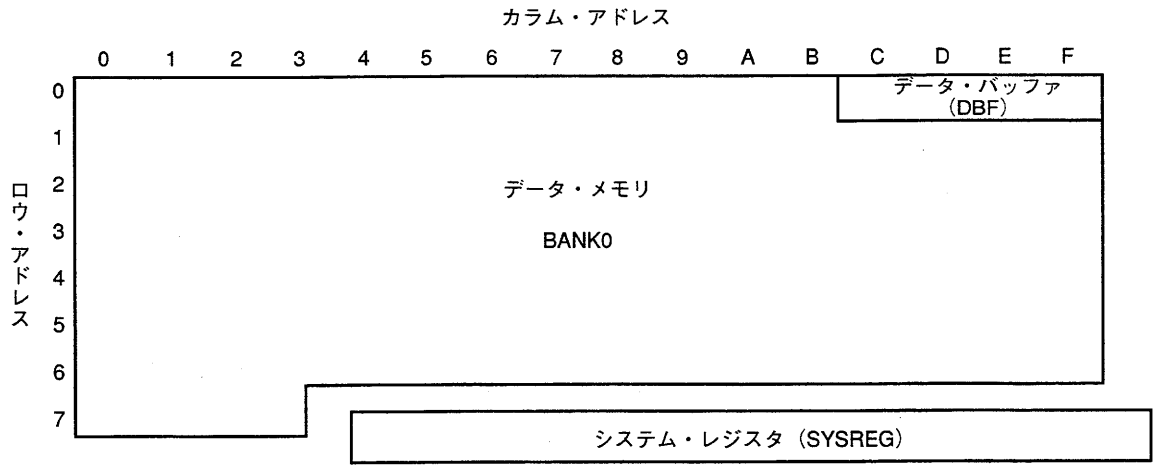


図9－2にデータ・バッファの構成を示します。図9－2に示すようにデータ・バッファはデータ・メモリの0FH番地のビット0をLSBとし，0CH番地のビット3をMSBとする16ビットで構成されています。

図9－2　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

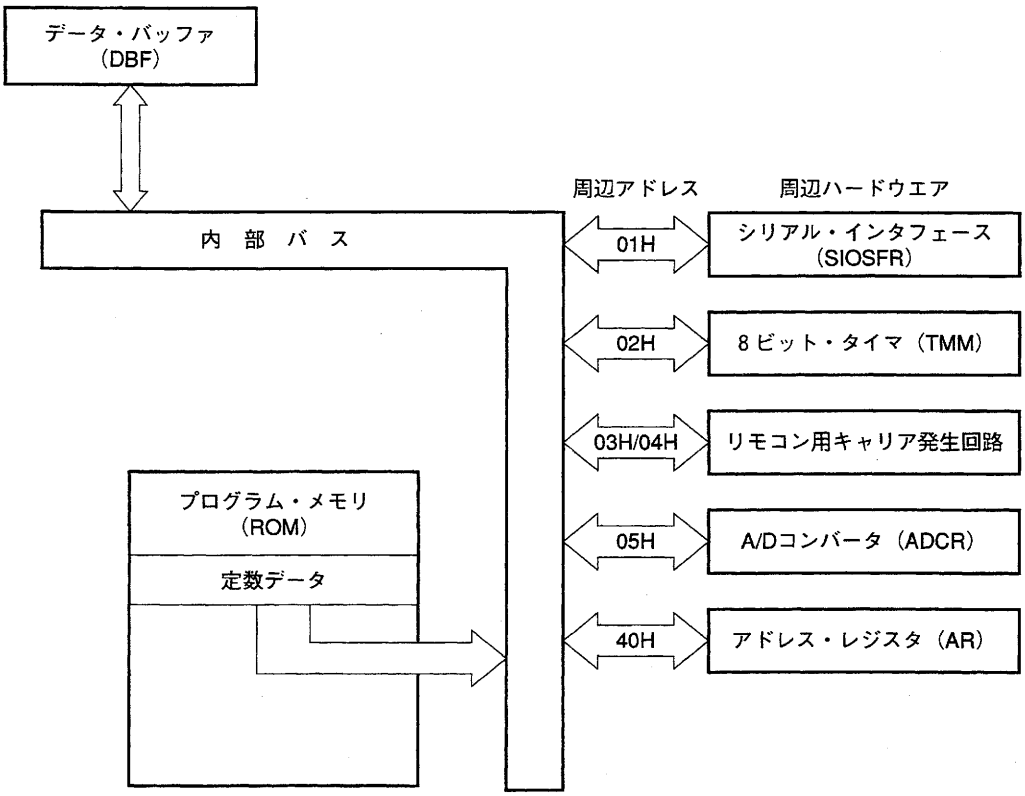
データ・バッファは，データ・メモリ上に配置されているため，すべてのデータ・メモリ操作命令で操作できます。

9.2　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9－3にμPD17201Aを例にデータ・バッファと周辺ハードウエアの関係を示します。

図9－3　データ・バッファと周辺ハードウエア（μPD17201A）



9.2.1　データ・バッファと周辺ハードウエア

表9－1に，μPD17201Aを例に，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と同じ）。

表9－1　周辺ハードウエア（μPD17201A）

（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		実用ビット長
			PUT	GET	
01H	SIOSFR	シリアル・インタフェース	○	○	8ビット
02H	TMM	8ビット・タイマ	○	○	8ビット
03H	NRZLTMM	NRZモジュロ・レジスタ（ロウ・レベル）	○	○	6ビット
04H	NRZHTMM	NRZモジュロ・レジスタ（ハイ・レベル）	○	○	6ビット
05H	ADCR	A/Dコンバータ	○	○	8ビット

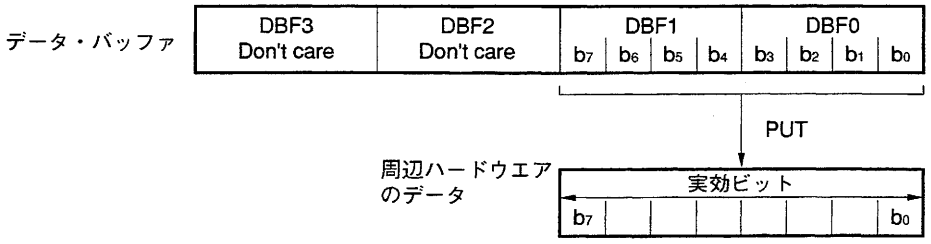
（2）16ビット単位で入出力を行う周辺ハードウエア

周辺アドレス	名　称	周辺ハードウエア	データ方向		実用ビット長
			PUT	GET	
40H	AR	アドレス・レジスタ	○	○	12ビット

9.2.2　周辺ハードウエアとのデータ転送

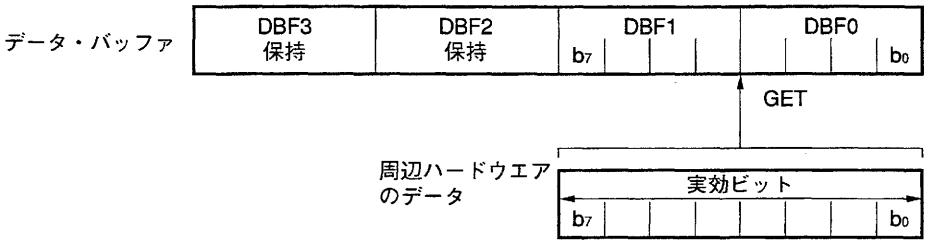
データ・バッファと各周辺ハードウエアとのデータ転送時は8ビットまたは16ビット単位で行います。このとき，PUTおよびGET命令は，データ・ビットが16ビットであっても1命令サイクルで実行できます。

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）



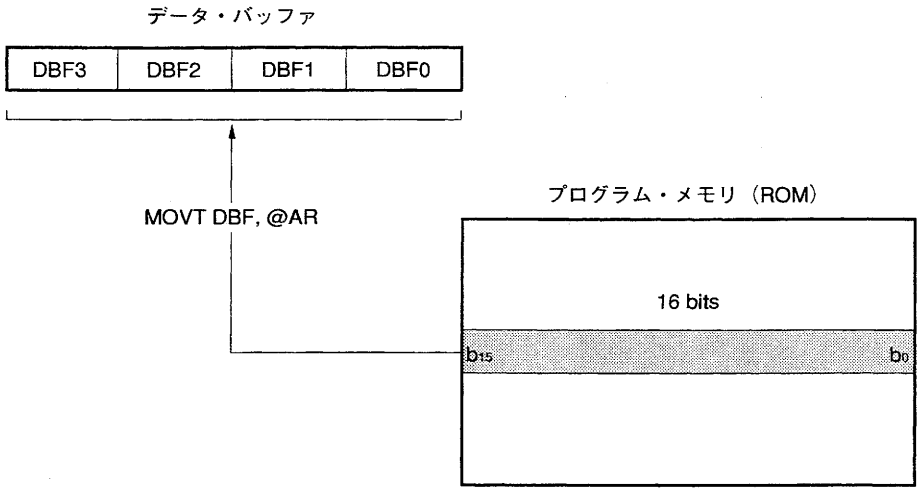
8ビット・データ読み込み時にはデータ・バッファの上位8ビット（DBF3，DBF2）の値は変化しません。

9.2.3　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。



注意　テーブル参照を行うときは，一時的にスタックが1レベル使用されるので，スタック・レベルに注意してください。

〔メ　モ〕

第10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。

10.1　ALUブロックの構成

図10－1にALUブロックの構成を示します。

図10－1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10－1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。

10.2　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10－1に各演算，判断，および回転処理命令の一覧を示します。

表10－1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。

10.2.1　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。

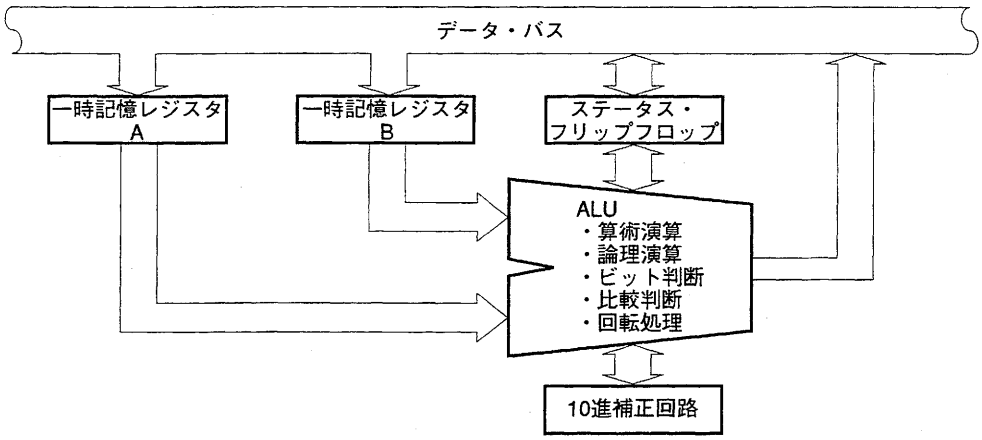
論理演算には論理積（AND），論理和（OR）および排他的論理和（XOR）があります。論理演算は，ジェネラル・レジスタの内容とデータ・メモリの内容との演算，またはデータ・メモリの内容とイミーディエト・データとの演算が行えます。

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10－1　ALUブロックの構成



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

ステータス・フリップフロップ			
BCD フラグ用 FF	CMP フラグ用 FF	CY フラグ用 FF	Z フラグ用 FF

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

〔メ　モ〕

表10－1　ALU処理命令一覧（1/2）

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

表10－1　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

10.2.2　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。

10.2.3　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2進4ビットで演算されます。

論理演算，ビット判断，比較判断，回転処理には影響を与えません。

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

10.2.4　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。

10.2.5　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10－2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10－2の網掛け部分）。

表10－2　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

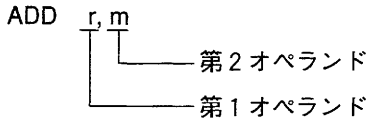
演算結果	2進4ビット減算		BCD減算	
	CY	演算結果	CY	演算結果
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	1100
11	0	1011	1	1101
12	0	1100	1	1110
13	0	1101	1	1111
14	0	1110	1	1100
15	0	1111	1	1101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

10.2.6　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転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NOP命令として実行）して動作を終了します。

ALUブロック動作については次の点に注意が必要です。

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

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

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

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

10.3　算術演算（2進4ビット加減算およびBCD加減算）

表10－3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。

10.3.1-10.3.4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10－3　算術演算の種類

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

10.3.1　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。

10.3.2　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。

10.3.3　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10－2　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

$\overset{\sim}{\text{CY}}$ 0, 0000B～$\overset{\sim}{\text{CY}}$ 1, 0011Bのことです

－10～－1Dとは同様に

$\overset{\sim}{\text{CY}}$ 1, 0110B～$\overset{\sim}{\text{CY}}$ 1, 1111Bのことです

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。

10.3.4　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。

10.3.5　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。

10.4　論理演算

表10－4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10－4　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10－5　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

10.5　ビット判断

表10－6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6章　システム・レジスタ（SYSREG）**を参照してください。

10.5.1，**10.5.2**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10－6　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

10.5.1　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。

10.5.2　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。

10.6　比較判断

表10－7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（**10.3　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。

10.6.1-**10.6.4**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10－7　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

10.6.3　“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。

10.6.4　“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。

10.7　回転処理

回転処理は，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。

10.7.1，10.7.2で，回転処理について説明します。

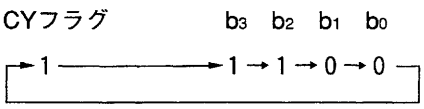
10.7.1　右回転処理

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；最上位
MOV   R2,    #0100B
MOV   R3,    #0010B；最下位
RORC  R1
RORC  R2
RORC  R3
```

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

10.7.2　左回転処理

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

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

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

〔メ　モ〕

第11章　割り込み機能

μPD172××サブシリーズの割り込み制御回路には以下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。

（1）割り込みイネーブル・フラグ（INTEF）と割り込み許可フラグ（IP×××）により各割り込みごとの受け付け可否を制御可能
（2）割り込み処理開始アドレスを割り込みベクタ・テーブルによって任意に設定可能
（3）多重割り込みが可能
（4）割り込み要求フラグ（IRQ×××）のリード／ライトが可能
（5）割り込み要求によるスタンバイ・モード（STOP, HALT）の解除が可能（割り込み許可フラグによる解除条件の選択も可能）

注意　割り込み処理において，ハードウエアにより自動的にスタックに退避されるのは，BCD, CMP, CY, Z, IXEの各フラグとBANKで，最大3レベルまでです。また，割り込み処理の内容において，周辺ハードウエア（タイマ，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。

11.1　割り込み制御回路の構成

11.1.1　割り込み制御（EI, DI）

割り込み要求により割り込み処理を行う場合，EI命令（INTEフラグのセット）により割り込み可能状態にしておく必要があります。

DI命令（INTEフラグのクリア）を実行し，割り込み禁止状態になっている場合は，すべての割り込みが保留されます。

11.1.2　割り込み許可フラグ（IP×××）

割り込み許可フラグ（IP×××）は，各割り込み要求フラグと1対1に対応しており，割り込み許可フラグがセットされているとき割り込みが許可され，割り込み許可フラグがリセットされているとき割り込みが保留されます。

保留されている割り込みは，割り込み要求を読み出すとセットされています。このセットされている割り込み要求をリセットすることにより割り込みの解除ができます。

11.1.3　割り込み要求フラグ（IRQ×××）

割り込み要求フラグ（IRQ×××）は，割り込み要求の発生でセットされ，割り込み処理が実行されると自動的にリセットされます。

割り込み要求フラグを命令によりセットすると，割り込みが発生していなくても発生した場合と同様にベクタ割り込みが実行されます（ソフトウエア割り込み）。

11.2　割り込みシーケンス

11.2.1　割り込みの受け付け

図11－1に割り込み受け付け時のタイミング・チャートを示します。

図11－1の（1）は1種類の割り込みによるタイミング・チャートです。

割り込みの受け付けは，割り込み要求フラグ（IRQ×××），INTEフラグおよび割り込み許可フラグ（IP××
×）のすべてがセットされた時点で行われます。

（1）の（a）は割り込み要求フラグ（IRQ×××）が最後にセット（1）された場合，（b）は割り込み許可フラグ（IP×××）が最後にセット（1）された場合のタイミング・チャートです。

最後のフラグをセットするタイミングがMOVT DBF, @AR命令の第1命令サイクルまたはスキップ条件を満たした命令の場合は，それぞれMOVT DBF, @AR命令の第2命令サイクルおよびスキップした命令（NOPになる）の実行後に割り込みを受け付けます。

INTEフラグは，EI命令を実行した次の命令サイクルでセットされます。

図11－1の（2）は複数の割り込みによるタイミング・チャートです。

複数の割り込みを用いるときは，割り込み許可フラグ（IP×××）がすべてセットされていれば，ハードウエアで優先されている割り込みを先に受け付けますが，プログラムで割り込み許可フラグを操作することにより，ハードウエアの優先度を変えることができます。

なお，図11－1の“割り込みサイクル”とは，割り込みが受け付けられてから割り込み要求フラグのリセット（0），ベクタ・アドレスの指定，プログラム・カウンタの退避などを行うための特別なサイクルです。割り込みサイクルは1命令実行時間分です。

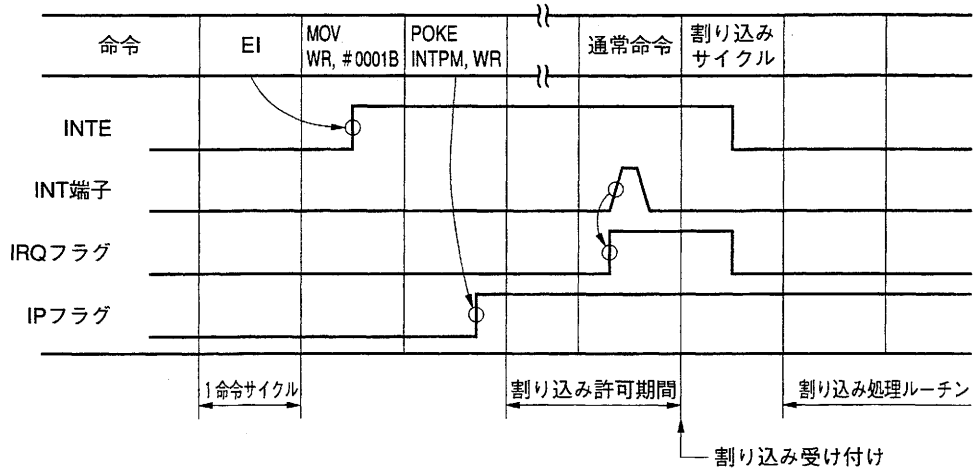
割り込み処理を開始すると，プログラムの戻り番地を格納するためアドレス・スタック・レジスタを1レベル消費し，さらにシステム・レジスタ中のPSWORDとBANKを退避するため割り込みスタック・レジスタを1レベル消費します。

図11－1　割り込み受け付けタイミング・チャート（1/3）

（1）1種類の割り込み（例：INT端子の立ち上がり）を使用時

（a）割り込み要求フラグ（IRQ××）が最後にセットされた場合

① 割り込み受け付け時が通常命令の場合



② 割り込み受け付け時がMOVT命令またはスキップ条件を満たした命令の場合

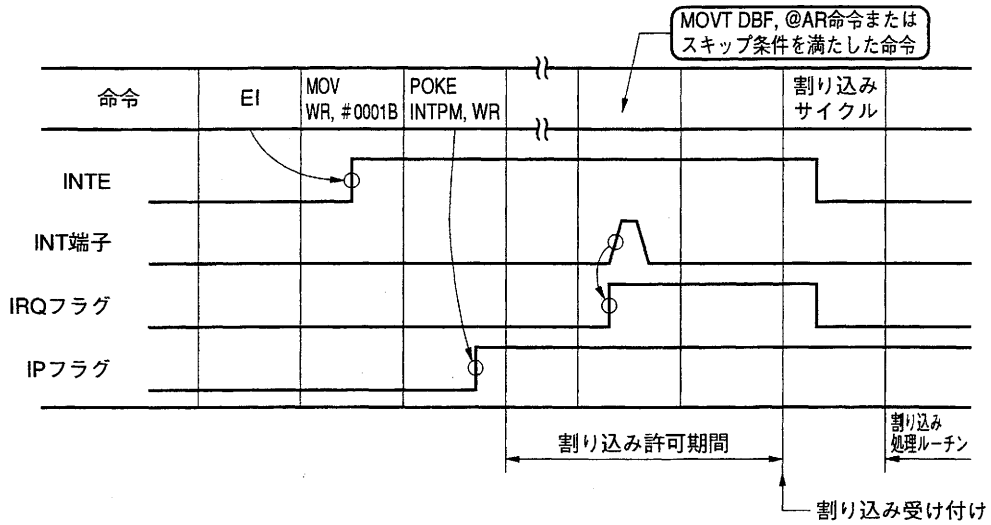


図11－1　割り込み受け付けタイミング・チャート（2/3）

（b）割り込み許可フラグ（IP××X）が最後にセットされた場合

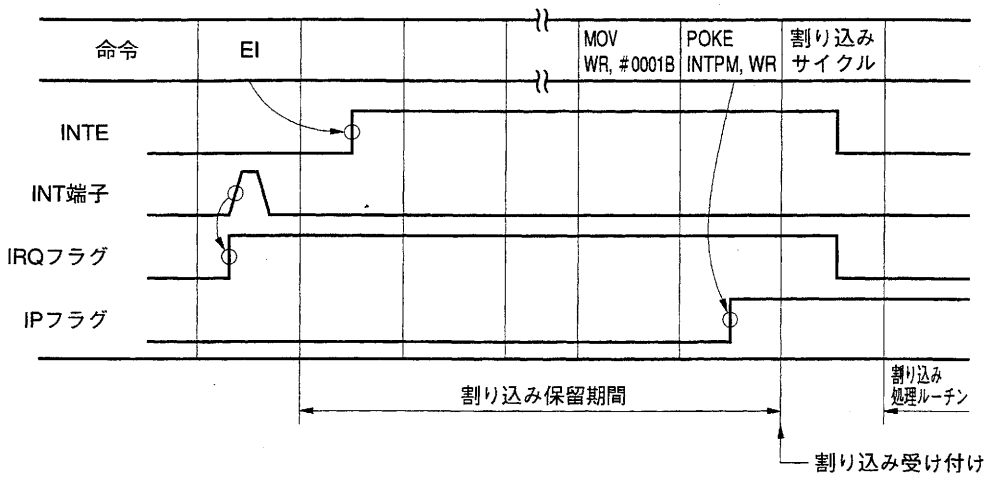
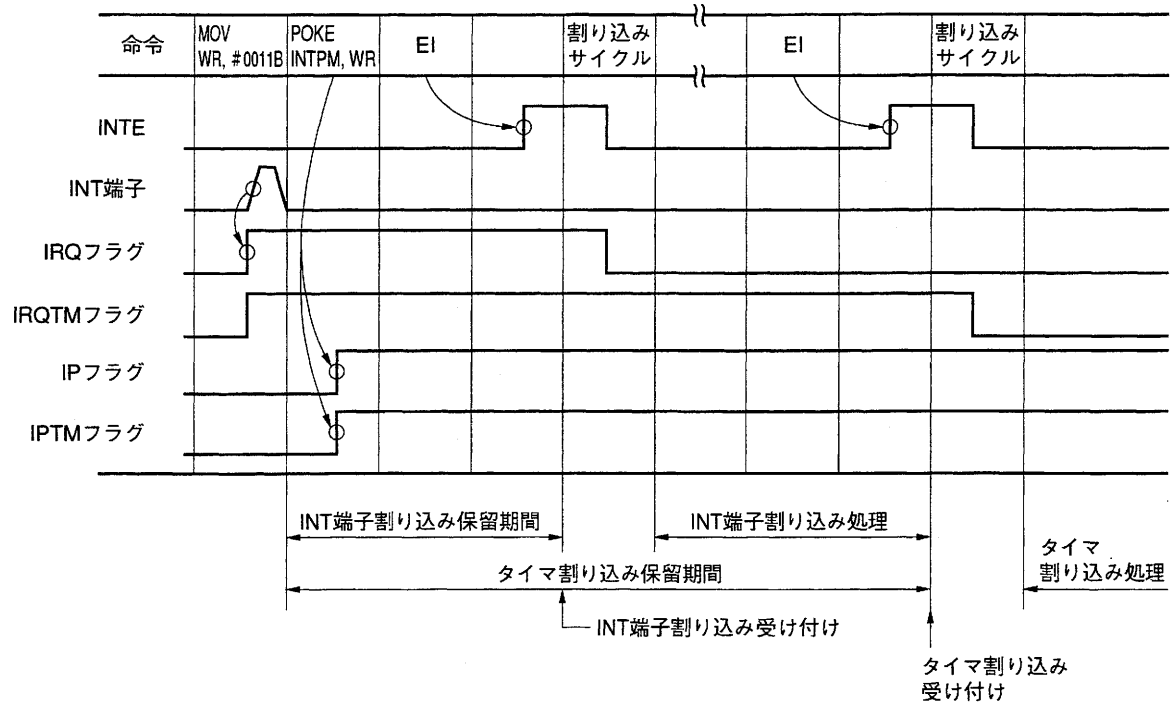


図11－1　割り込み受け付けタイミング・チャート（3/3）

（2）複数の割り込み（例：INT端子，タイマの2種類）を使用時

（a）ハードウエア優先



（b）ソフトウエア優先

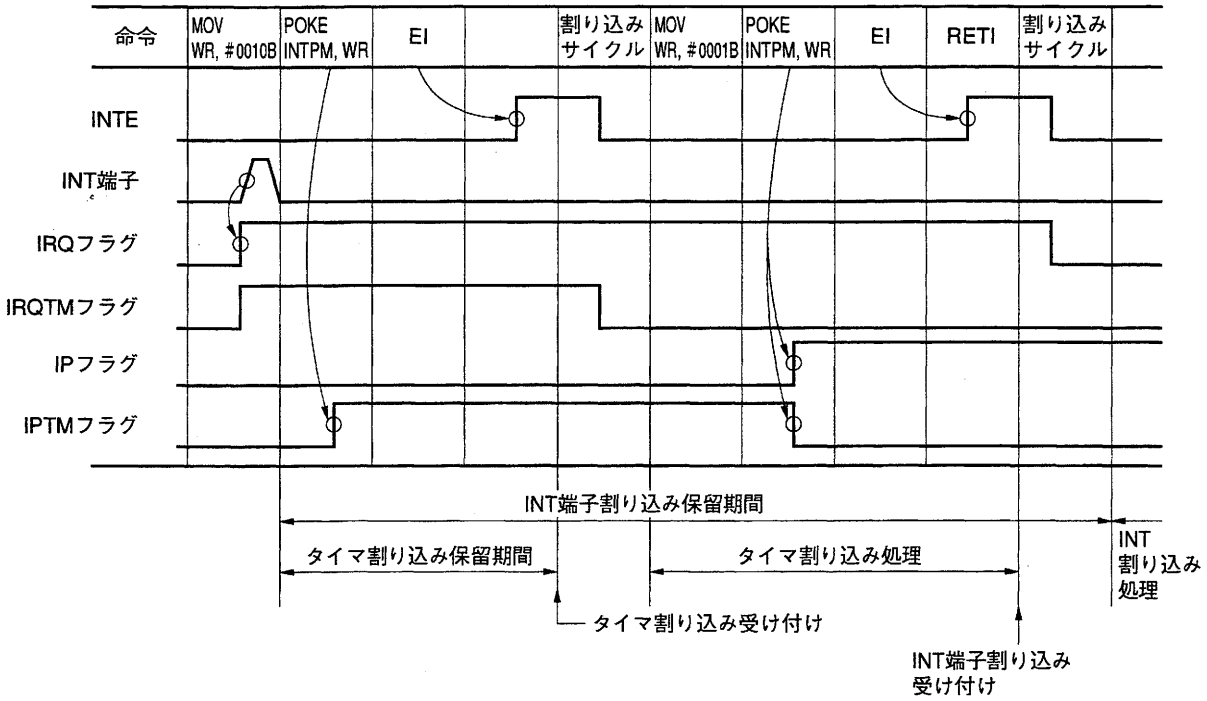
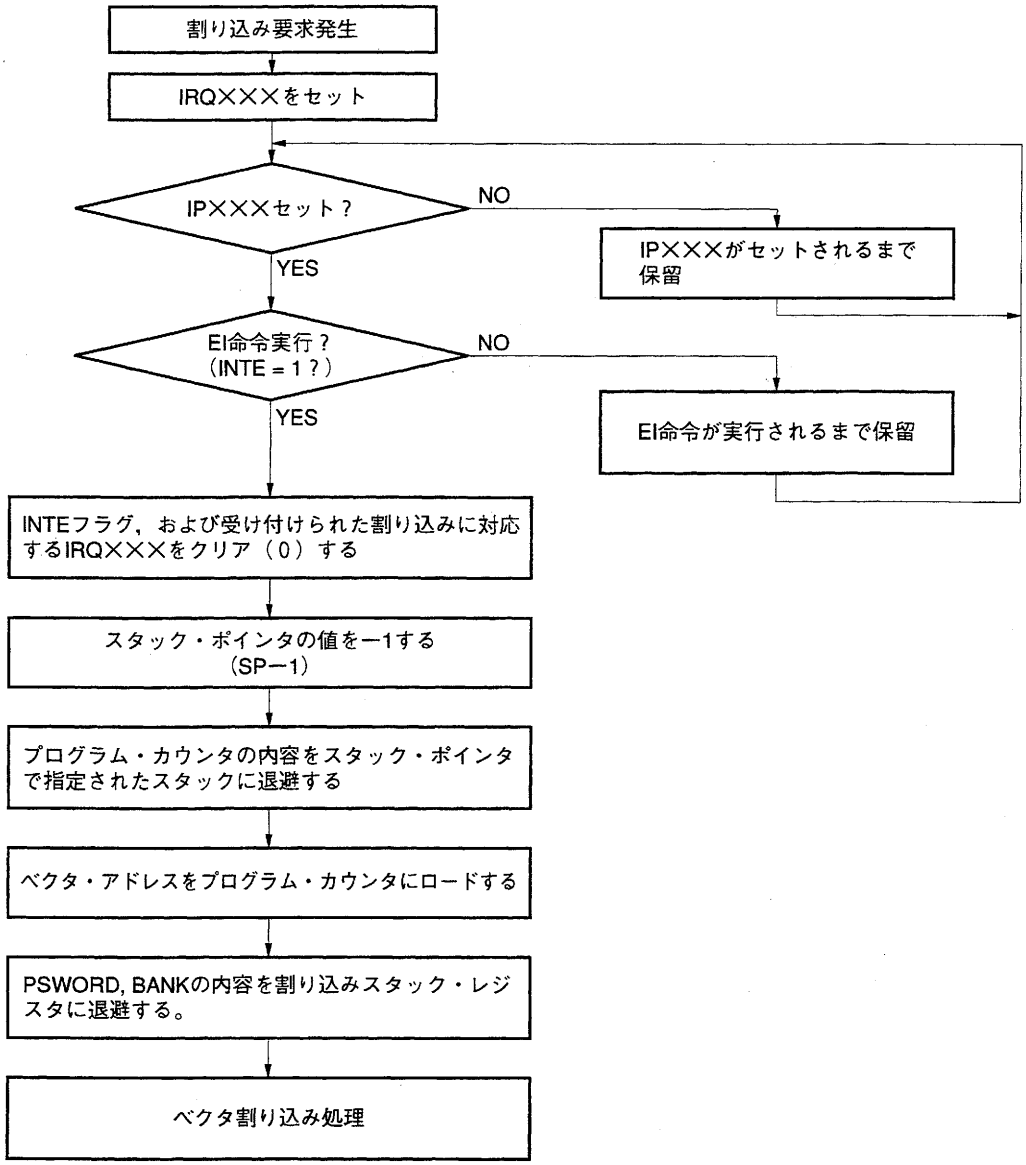


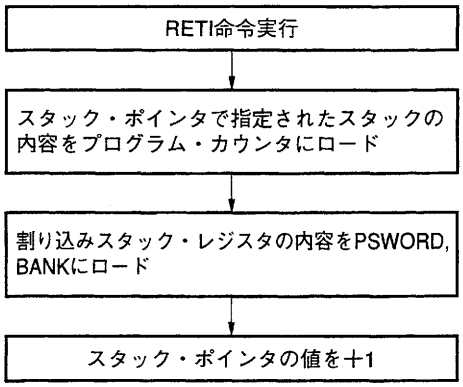
図11－2　割り込み処理手順



11.2.2　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11－3　割り込み処理からの復帰

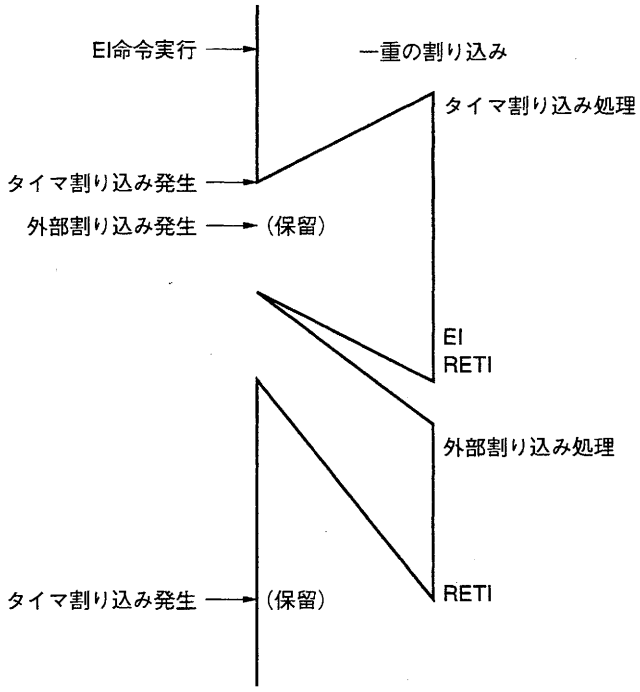


注意　RETI命令ではINTEフラグはセットされません。

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。

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



〔メ　モ〕

第12章　スタンバイ機能

μPD172××サブシリーズはCMOSプロセスの特徴である低消費電流をさらに低減するために，スタンバイ機能を内蔵しています。

12.1　機能概要

μPD172××サブシリーズのスタンバイ機能は，STOPモードと，HALTモードの2つのモードがあり，用途に応じて使い分けることができます。

STOPモードは，メイン・クロック発振回路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，メイン・クロック発振回路の発振は接続しますが，システム・クロックの供給が停止するためCPUの動作が停止するモードです。HALTモードは，STOPモードほどの消費電流低減はできませんが，メイン・クロック発振回路が動作しているため，割り込み要求によりすぐ動作を開始したい場合に有効です。

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。したがって，システム全体の消費電流を抑えることができるように，あらかじめ入出力ポートの状態を設定してください。

表12－1　スタンバイ・モード時の各動作状態

	STOPモード	HALTモード
設定命令	STOP命令	HALT命令
メイン・クロック発振回路	発振停止	発振継続
サブクロック発振回路	発振継続	
時計用タイマ	サブクロックが発振していれば動作可能	動作可能
シリアル・インタフェース	シリアル・クロックに外部$\overline{SCK}$入力を指定すれば動作可能	動作可能
タイマ・カウンタ	外部クロック入力を指定すれば動作可能	動作可能
CPU	動作停止	
データ・メモリ	データ保持	
コントロール・レジスタ	データ保持	
出力ポート	出力データ保持	

注意　STOP命令は，サブシステム・クロックのみのシステムでは無効です。

12.2　STOPモードの設定と解除

12.2.1　STOPモードの設定

STOPモードの設定はSTOP命令を使用します。STOP命令はメイン・クロックをシステム・クロックとして使用しているときのみ有効です。サブクロックをシステム・クロックとして使用しているときSTOP命令を実行した場合，STOP命令はNOP命令として実行されSTOPモードにはなりません。

また，STOPモード解除条件が成立した状態でSTOP命令を実行した場合もNOPとして実行されSTOPモードにはなりません。

STOPモードの解除条件は，STOP命令のオペランド指定できます。STOP命令のオペランドと解除条件の関係については各デバイスのデータ・シートを参照してください。

12.2.2　STOPモード解除時の動作

STOP命令のオペランドで指定したスタンバイ解除条件が成立したとき，解除後の動作が以下のようになります。

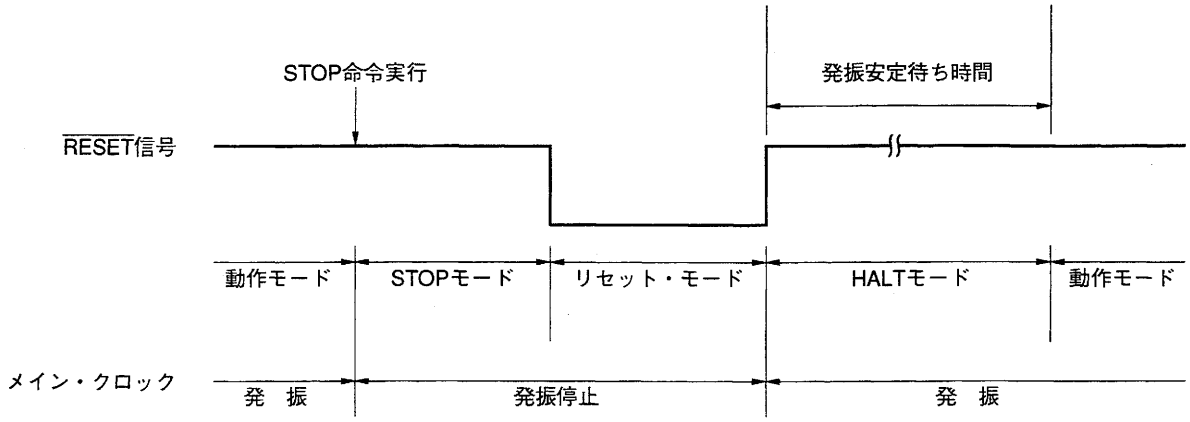
① IRQTMのリセット
② ベーシック・インターバル・タイマ（または時計用タイマ・カウンタ）とウォッチドッグ・タイマのスタート（リセットされません）
③ 8ビット・タイマ／カウンタのリセット，スタート
④ 8ビット・カウンタの値がモジュロ・レジスタの値と一致（IRQTMのセット）したとき，「STOP 8H」の次または割り込みベクタ・アドレス分岐の命令を実行します。

発振回路は，STOP命令が実行された場合に発振停止となります（STOPモード）。STOPモードが解除されるまで発振を再開しません。STOPモード解除後はHALTモードとなりますので，モジュロ機能付きタイマによってHALTモード解除の時間を設定します。

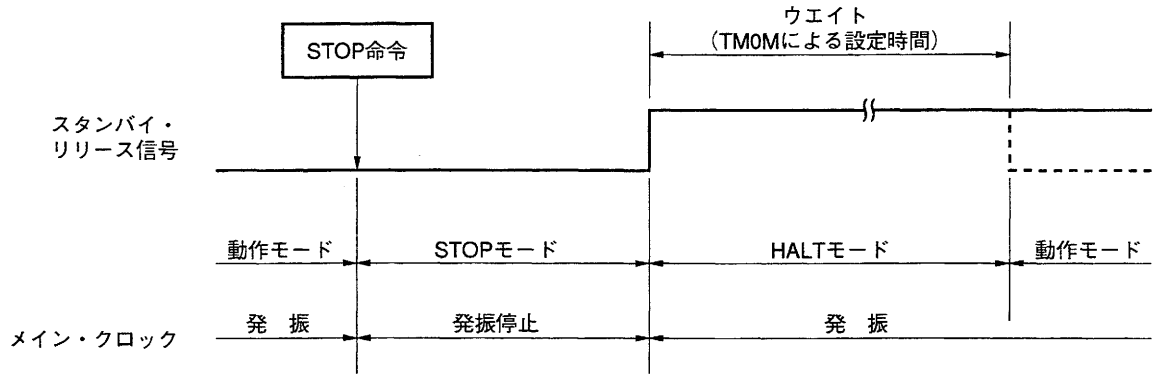
注意　8ビット・モジュロ・レジスタの設定はSTOP命令を実行する前に行ってください。

図12－1　STOPモード解除後の動作

（1）$\overline{\text{RESET}}$入力による解除



（2）$\overline{\text{RESET}}$入力以外による解除



備考　破線は，スタンバイを解除した割り込み要求が受け付けられた場合です。

12.3　HALTモードの設定と解除

12.3.1　HALTモードの設定

HALTモードの設定はHALT命令を使用します。

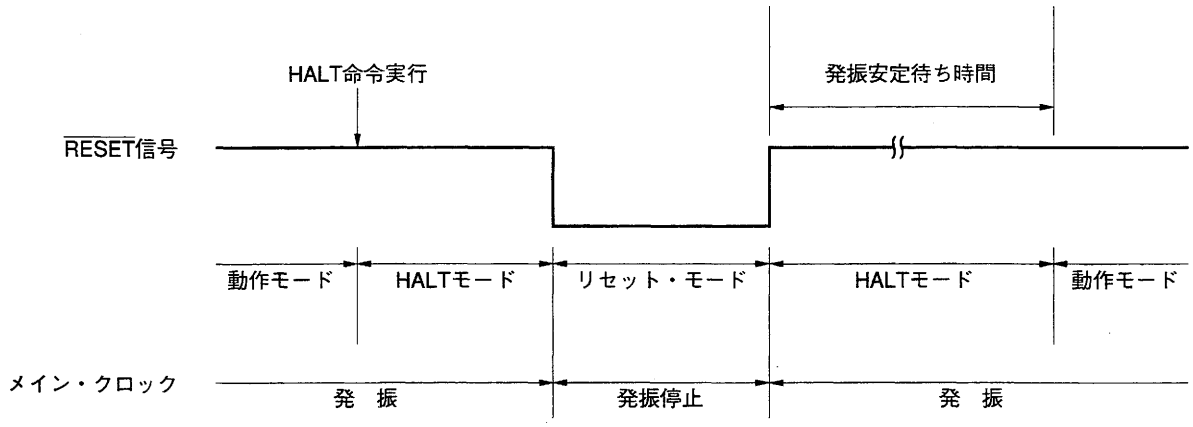
HALTモードの解除条件は，HALT命令のオペランドで指定できます。HALT命令のオペランドと解除条件の関係については各デバイスのデータ・シートを参照してください。

12.3.2　HALTモード解除時の動作

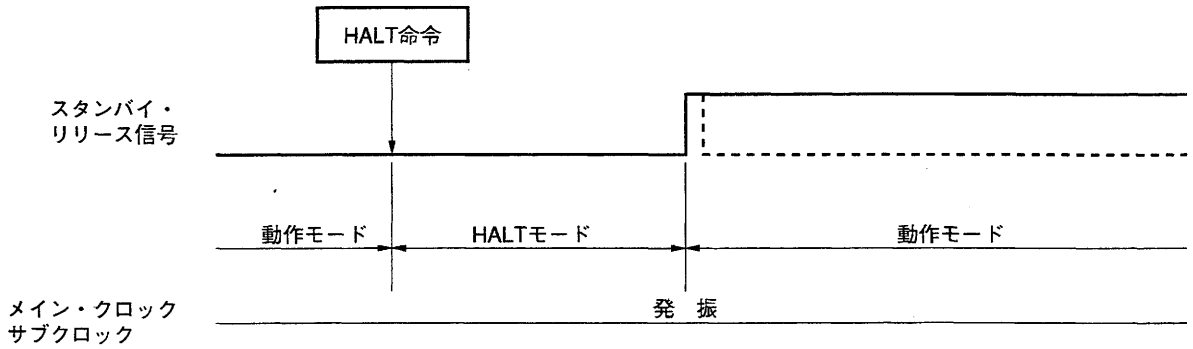
HALT命令のオペランドで指定したスタンバイ解除条件が成立したとき，解除後の動作は以下のようになります。

図12－2　HALTモード解除後の動作

（1）$\overline{\text{RESET}}$入力による解除



（2）$\overline{\text{RESET}}$入力以外による解除



備考　破線は，スタンバイを解除した割り込み要求が受け付けられた場合です。

第13章　リセット機能

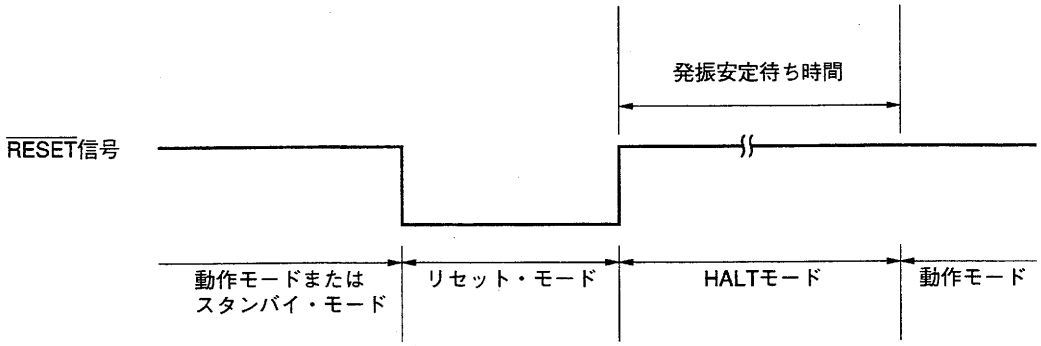
13.1　$\overline{\text{RESET}}$端子によるリセット

$\overline{\text{RESET}}$端子にロウ・レベル信号を入力するとリセットがかかります。
電源投入時には内部回路の動作が不定となるため，少なくとも1回はリセットをかけてください。
リセットがかかると以下の回路が初期化されます。

（1）プログラム・カウンタが0000Hにリセットされます。
（2）コントロール・レジスタが初期化されます。
　　初期値は各デバイスごとに異なりますので，各デバイスのデータ・シートを参照してください。
（3）データ・バッファ（DBF）が初期化されます。
（4）周辺ハードウエアが初期化されます。

$\overline{\text{RESET}}$端子をロウ・レベルからハイ・レベルに立ち上げると，メイン・クロックの発振を開始し，発振安定持ち時間後に0番地からプログラムの実行を開始します。

図13－1　$\overline{\text{RESET}}$入力によるリセット動作



13.2　ウォッチドッグ機能（$\overline{\text{WDOUT}}$出力）

μPD172××サブシリーズは，$\overline{\text{RESET}}$端子と$\overline{\text{WDOUT}}$端子を接続することにより，プログラムの誤動作による暴走を未然に防ぐウォッチドッグ機能として，ウォッチドッグ・タイマおよびスタック・レベル・チェックを行うことができます。必ず，$\overline{\text{RESET}}$端子と$\overline{\text{WDOUT}}$端子を接続して使用してください。

13.2.1　ウォッチドッグ・タイマによるリセット（$\overline{\text{RESET}}$端子と$\overline{\text{WDOUT}}$端子を接続）

プログラム実行中にウォッチドッグ・タイマが働くと$\overline{\text{WDOUT}}$端子にロウ・レベルを出力し，プログラム・カウンタを0にリセットします。

すなわち，一定時間以上ウォッチドッグ・タイマのリセットが行われなくなった場合，プログラムを0H番地からスタートさせることができます。

プログラム作成の際は340 ms（fx = 4 MHzのとき）以内の間隔でウォッチドッグ・タイマをリセット（WDTRESフラグをセット）してください。

13.2.2　スタック・ポインタによるリセット（$\overline{\text{RESET}}$端子と$\overline{\text{WDOUT}}$端子を接続）

プログラム実行中にスタック・ポインタが実装されていないアドレス・スタックを示す値になると，$\overline{\text{WDOUT}}$端子にロウ・レベルを出力し，プログラム・カウンタが0000Hにリセットされます。

13.3　低電圧検出回路（$\overline{\text{RESET}}$端子と$\overline{\text{WDOUT}}$端子を接続）

低電圧検出回路は，電池交換時のプログラム暴走を防ぐため，低電圧を検出した場合$\overline{\text{WDOUT}}$端子よりロウ・レベルを出力して初期化状態（リセット状態）としています。

★　μPD17225, 17226, 17227, 17228は，低電圧検出回路をマスク・オプションで任意に設定できます。詳しくは，それぞれのデータ・シートを参照してください。

13.4　INT端子および$\overline{\text{RESET}}$端子の使用上の注意

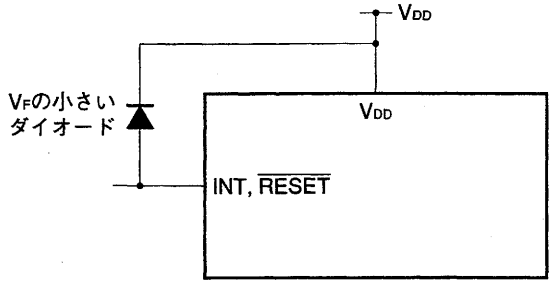
INT端子および$\overline{\text{RESET}}$端子は，端子そのものの機能のほかに，マイコンの内部動作をテストするテスト・モードを設定する機能（ICテスト専用）を持っています。

これらの端子のいずれかにV_{DD}を越えた電圧を印加すると，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。

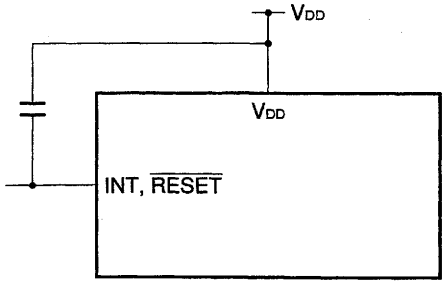
たとえば，INT端子および$\overline{\text{RESET}}$端子の配線の引き回しが長い場合などでは，これらの端子に布線間ノイズが加わって上記の問題を起こしてしまうことがあります。

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14章　ワン・タイムPROMの書き込みとベリファイ

μPD172××サブシリーズのワン・タイムPROM製品として，μPD17P203A, 17P204, 17P207, 17P218があります。これらに内蔵されているプログラム・メモリは電気的書き込み可能なワン・タイムPROMで構成されています。

ワン・タイムPROMの書き込み／ベリファイには，表14－1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}＋0.3 V以上の高電圧が印加されると，マイコンが暴走する可能性がありますので，端子の保護には十分注意してください。

表14－1　プログラム・メモリ書き込み／ベリファイ時の使用端子

端子名	機　　能
V_{PP}	プログラム電圧印加用端子です。＋12.5 Vを印加します。
V_{DD}	正電源です。＋6 Vを印加します。
CLK	アドレス更新用クロック入力です。パルスを4回入力することにより，プログラム・メモリのアドレスを更新します。
MD_0-MD_3	動作モード選択用入力です。
D_0-D_7	8ビット・データ入出力端子です。

14.1　マスクROM製品とワン・タイムPROM製品との違い

μPD17P2××は，マスクROM内蔵製品μPD172××サブシリーズのプログラム・メモリをワン・タイムPROMに置き換えた製品です。

表14－3～14－7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は同じです。したがって，μPD172××サブシリーズのワン・タイムPROM製品と，その対象になるマスクROM製品の組み合わせは次のとおりです。

なお，サブクロック発振回路のある製品では，**メイン・クロック発振回路のみの使用はできません。**必ず，サブクロック発振回路も使ってください。

★ 表14－2　ワン・タイムPROM製品とマスクROM製品の組み合わせ

ワン・タイムPROM製品	マスクROM製品
μPD17P203A	μPD17P203A
μPD17P204	μPD17P204
μPD17P207	μPD17201A,17207
μPD17P218	μPD17225, 17226, 17227, 17228

表14－3　μPD17P203AとμPD17203Aの違い

項　　目	μPD17P203A-001	μPD17P203A-002	μPD17P203A-003	μPD17203A
ROM	ワン・タイムPROM			マスクROM
	4096×16ビット			
$\overline{\text{RESET}}$端子のプルアップ抵抗	あり	なし	なし	マスク・オプション
P0A, P0B端子のプルアップ抵抗		あり		
メイン・クロック発振回路使用の有無				
サブクロック発振回路使用の有無		なし	あり	
V_{PP}端子，PROMプログラム用端子	あり			なし

表14－4　μPD17P204とμPD17204の違い

項　　目	μPD17P204-001	μPD17P204-002	μPD17P204-003	μPD17204
ROM	ワン・タイムPROM			マスクROM
	7936×16ビット			
$\overline{\text{RESET}}$端子のプルアップ抵抗	あり	なし	なし	マスク・オプション
P0A, P0B端子のプルアップ抵抗		あり		
メイン・クロック発振回路使用の有無				
サブクロック発振回路使用の有無		なし	あり	
V_{PP}端子，PROMプログラム用端子	あり			なし

表14－5　μPD17P207とμPD17201A, 17207の違い

項　　目	μPD17P207-001	μPD17P207-002	μPD17P207-003	μPD17207	μPD17201A
ROM	ワン・タイムPROM			マスクROM	
	4096×16ビット				3072×16ビット
$\overline{\text{RESET}}$端子のプルアップ抵抗	あり	なし	なし	マスク・オプション	
メイン・クロック発振回路使用の有無		あり			
サブクロック発振回路使用の有無		なし	あり		
V_{PP}端子，PROMプログラム用端子	あり			なし	

★ 表14－6　μPD17P218とμPD17225, 17226, 17227, 17228の違い

項　　目	μPD17P218	μPD17225	μPD17226	μPD17227	μPD17228
ROM	ワン・タイムPROM	マスクROM			
	8192×16ビット	2048×16ビット	4096×16ビット	6144×16ビット	8192×16ビット
$\overline{\text{RESET}}$端子のプルアップ抵抗	あり	マスク・オプション			
低電圧検出回路					
V_{PP}端子，PROMプログラム用端子	あり	なし			

14.2　プログラム・メモリ書き込み／ベリファイ時の動作モード

μPD17P2××は，ある一定時間のリセット状態（V_{DD} = 5 V, $\overline{RESET}$ = 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，プログラム・メモリ書き込み／ベリファイ時に使用しない端子は，プルダウン抵抗（470 Ω）を介してGNDに接続してください。

表14－7　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）

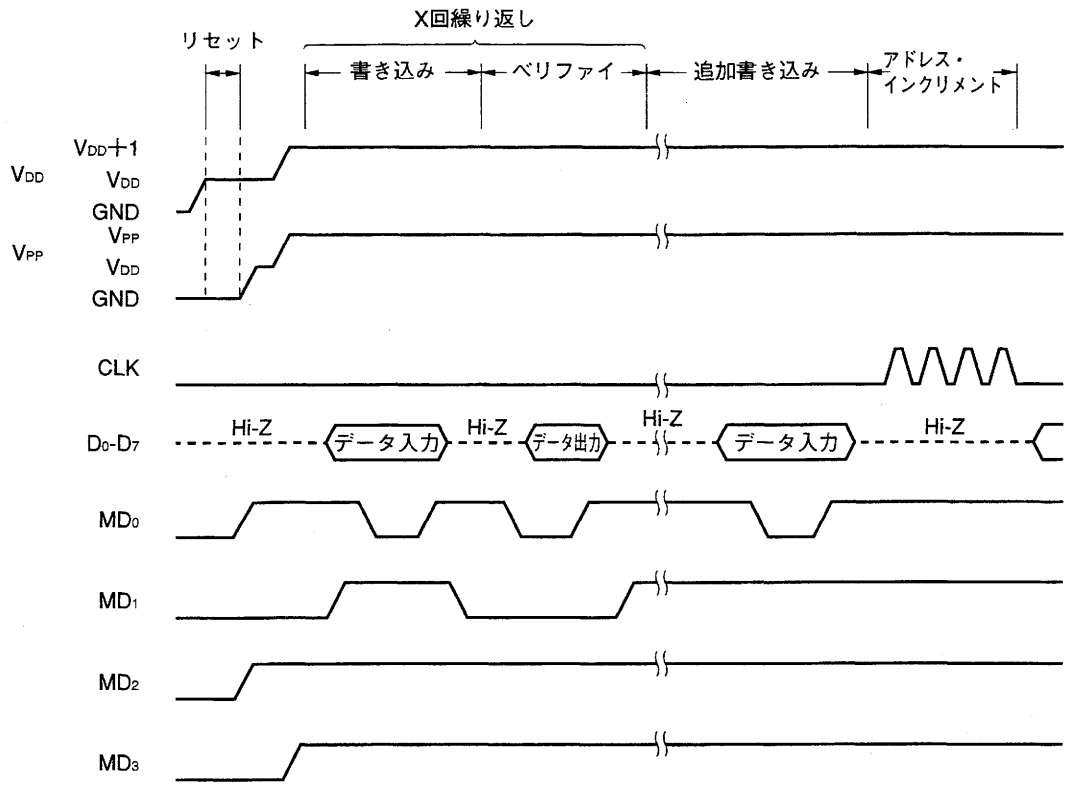
14.3　プログラム・メモリ書き込み手順

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5 Vを供給。V_{PP}端子はロウ・レベル。
（3）10 μsウエイト後，V_{PP}端子に5 Vを供給。
（4）モード設定端子をプログラム・メモリ・アドレスの0クリア・モードに設定。
（5）V_{DD}に6 V，V_{PP}に12.5 Vを供給。
（6）プログラム・インヒビット・モード。
（7）1 msの書き込みモードでデータを書き込む。
（8）プログラム・インヒビット・モード。
（9）ベリファイ・モード。書き込めていれば（10）へ，書き込めていなければ（7）-（9）を繰り返す。
（10）（（7）-（9）で書き込んだ回数：X）×1 msの追加書き込み。
（11）プログラム・インヒビット・モード。
（12）CLK端子にパルスを4回入力することにより，プログラム・メモリのアドレスを更新（＋1）。
（13）（7）-（12）を最終アドレスまで繰り返す。
（14）プログラム・メモリ・アドレスの0クリア・モード。
（15）V_{DD}, V_{PP}端子の電圧を5 Vに変更。
（16）電源オフ。

この（2）-（12）の手順を図14－1に示します。

図14－1　プログラム・メモリ書き込み手順

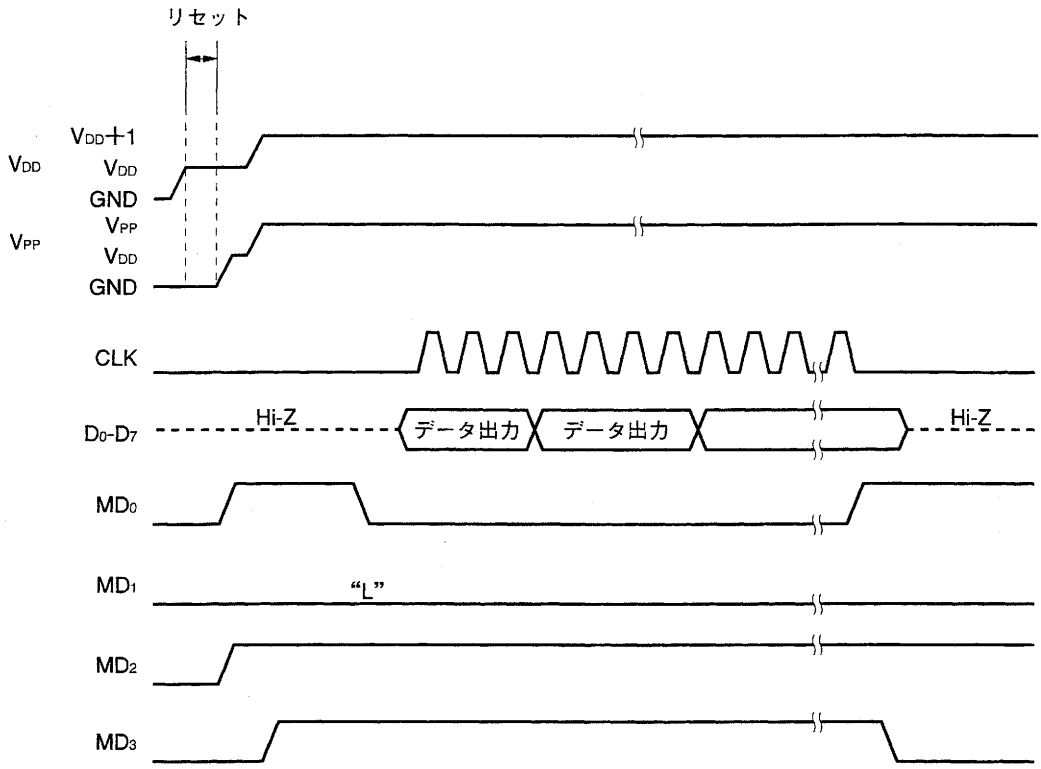


14.4　プログラム・メモリ読み出し手順

（1）使用しない端子を抵抗を介してGNDにプルダウン。CLK端子はロウ・レベル。
（2）V_{DD}端子に5 Vを供給。V_{PP}端子はロウ・レベル。
（3）10 μsウエイト後，V_{PP}端子に5 Vを供給。
（4）モード設定端子をプログラム・メモリ・アドレスの0クリア・モードに設定。
（5）V_{DD}端子に6 V，VPPに12.5 Vを供給。
（6）プログラム・インヒビット・モード。
（7）ベリファイ・モード。CLK端子にクロック・パルスを入力すると，4回入力するごとにデータを1アドレスずつ順次出力。
（8）プログラム・インヒビット・モード。
（9）プログラム・メモリ・アドレスの0クリア・モード。
（10）V_{DD}, V_{PP}端子の電圧を5 Vに変更。
（11）電源オフ。

プログラム・メモリ読み出し手順の（2）-（9）を図14－2に示します。

図14－2　プログラム・メモリ読み出し手順



第15章　命令セット

15.1　命令セット概要

b_{14}-b_{11} \ b_{15}		0		1	
BIN	HEX				
0000	0	ADD	r, m	ADD	m, #n4
0001	1	SUB	r, m	SUB	m, #n4
0010	2	ADDC	r, m	ADDC	m, #n4
0011	3	SUBC	r, m	SUBC	m, #n4
0100	4	AND	r, m	AND	m, #n4
0101	5	XOR	r, m	XOR	m, #n4
0110	6	OR	r, m	OR	m, #n4
0111	7	INC	AR		
		INC	IX		
		MOVT	DBF, @AR		
		BR	@AR		
		CALL	@AR		
		RET			
		RETSK			
		EI			
		DI			
		RETI			
		PUSH	AR		
		POP	AR		
		GET	DBF, p		
		PUT	p, DBF		
		PEEK	WR, rf		
		POKE	rf, WR		
		RORC	r		
		STOP	s		
		HALT	h		
		NOP			
1000	8	LD	r, m	ST	m, r
1001	9	SKE	m, #n4	SKGE	m, #n4
1010	A	MOV	@r, m	MOV	m, @r
1011	B	SKNE	m, #n4	SKLT	m, #n4
1100	C	BR	addr（ページ0）	CALL	addr
1101	D	BR	addr（ページ1）	MOV	m, #n4
1110	E			SKT	m, #n
1111	F			SKF	m, #n

15.2　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

15.3　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	mc	r
		m, #n4	(m) ← (m) ＋n4	10000	m_R	mc	n4
	ADDC	r, m	(r) ← (r) ＋ (m) ＋CY	00010	m_R	mc	r
		m, #n4	(m) ← (m) ＋n4＋CY	10010	m_R	mc	n4
	INC	AR	AR←AR＋1	00111	000	1001	0000
		IX	IX←IX＋1	00111	000	1000	0000
減算	SUB	r, m	(r) ← (r) － (m)	00001	m_R	mc	r
		m, #n4	(m) ← (m) －n4	10001	m_R	mc	n4
	SUBC	r, m	(r) ← (r) － (m) －CY	00011	m_R	mc	r
		m, #n4	(m) ← (m) －n4－CY	10011	m_R	mc	n4
論理演算	OR	r, m	(r) ← (r) ∨ (m)	00110	m_R	mc	r
		m, #n4	(m) ← (m) ∨n4	10110	m_R	mc	n4
	AND	r, m	(r) ← (r) ∧ (m)	00100	m_R	mc	r
		m, #n4	(m) ← (m) ∧n4	10100	m_R	mc	n4
	XOR	r, m	(r) ← (r) ⊻ (m)	00101	m_R	mc	r
		m, #n4	(m) ← (m) ⊻ n4	10101	m_R	mc	n4
判断	SKT	m, #n	CMP←0, if (m) ∧n = n, then skip	11110	m_R	mc	n
	SKF	m, #n	CMP←0, if (m) ∧n = 0, then skip	11111	m_R	mc	n
比較	SKE	m, #n4	(m) －n4, skip if zero	01001	m_R	mc	n4
	SKNE	m, #n4	(m) －n4, skip if not zero	01011	m_R	mc	n4
	SKGE	m, #n4	(m) －n4, skip if not borrow	11001	m_R	mc	n4
	SKLT	m, #n4	(m) －n4, skip if borrow	11011	m_R	mc	n4
回転	RORC	r	┌► CY→(r)$_{b3}$→(r)$_{b2}$→(r)$_{b1}$→(r)$_{b0}$ ┐	00111	000	0111	r
転送	LD	r, m	(r) ← (m)	01000	m_R	mc	r
	ST	m, r	(m) ← (r)	11000	m_R	mc	r
	MOV	@r, m	if MPE = 1 : (MP, (r)) ← (m) if MPE = 0 : (BANK, m_R, (r)) ← (m)	01010	m_R	mc	r
		m, @r	if MPE = 1 : (m) ← (MP, (r)) if MPE = 0 : (m) ← (BANK, m_R, (r))	11010	m_R	mc	r
		m, #n4	(m) ←n4	11101	m_R	mc	n4
	MOVT[注]	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	注		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

注　“BR addr”のオペレーションとオペ・コードは，各製品のROMサイズによって，それぞれ次のようになります。

★　（a）μPD17225

オペランド	オペレーション	オペ・コード
addr	PC_{10-0}←addr	01100

★　（b）μPD17201A, 17203A, 17P203A, 17207, 17P207, 17226

オペランド	オペレーション	オペ・コード
addr	PC_{10-0}←addr, PAGE←0	01100
	PC_{10-0}←addr, PAGE←1	01101

★　（c）μPD17227

オペランド	オペレーション	オペ・コード
addr	PC_{10-0}←addr, PAGE←0	01100
	PC_{10-0}←addr, PAGE←1	01101
	PC_{10-0}←addr, PAGE←2	01110

★　（d）μPD17204, 17P218, 17228

オペランド	オペレーション	オペ・コード
addr	PC_{10-0}←addr, PAGE←0	01100
	PC_{10-0}←addr, PAGE←1	01101
	PC_{10-0}←addr, PAGE←2	01110
	PC_{10-0}←addr, PAGE←3	01111

★ 15.4　アセンブラ（RA17K）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	注
拡張命令	BRX	Label	Jump Label	—
	CALLX	function-name	CALL sub-routine	—
	INITFLGX	〈NOT/INV〉flag 1, …〈NOT/INV〉flag n	if description = NOT（or INV）←0 flag,（flag）←0 if description = flag,（flag）←1	n≦4

注　n = 0　：μPD17225, μPD17226

n = 0, 1　：μPD17P218, μPD17227, μPD17228

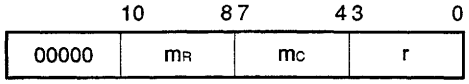
n = 0 - 2　：μPD17201A, μPD17203A, μPD17P203A, μPD17204, μPD17P204, μPD17207, μPD17P207

15.5　命令の個別説明

15.5.1　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, ME02F
```

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							B
	0	0	0	0	ロウ・アドレス			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）の割り当ては上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←00000010000B (0.10H) 注
        MOV   IXM, #01H
        MOV   IXL, #00H
        SET1  IXE                ; IXEフラグ←1
        ADD   MEM003, MEM02F     ; IX                    00000010000B (0.10H)
                                 ; バンク・オペランド OR) 00000101111B (0.2FH)
                                 ; 指定アドレス          00100111111B (0.3FH)
```

注

レジスタ	IX											
	IXH				IXM				IXL			
ビット	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
データ	MPE	0	0	0	0	ロウ・アドレス			カラム・アドレス			
		バンク										

システム・レジスタ中のIX（インデクス・レジスタ）の割り当ては前頁の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

10	8 7	4 3	0
10000	mR	mC	n4

② 機　能

CMP＝0のとき　　(m) ← (m) ＋n4

データ・メモリの内容にイミーディエト・データを加算し，結果をデータ・メモリへ格納します。

CMP＝1のとき　　(m) ＋n4

結果はデータ・メモリに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

加算の結果，桁上げがあればキャリー・フラグCYをセットし，桁上げがなければキャリー・フラグCYをリセットします。

加算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で加算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で加算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

加算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをPSWORDのBCDフラグによって指定します。

③ 例1

0.2FH番地の内容に5を加算し，結果を0.2FH番地に格納します。

(0.2FH) ← (0.2FH) +5

```
MEM02F  MEM   0.2FH
        ADD   MEM02F, #05H
```

例2

0.6FH番地の内容に5を加算した結果を0.6FH番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

(0.6FH) ← (0.6FH) +05H
　　└インデクス・レジスタの内容0.40Hとデータ・メモリのアドレス0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ; データ・メモリのバンクを0
        MOV   IXH, #00H       ; IX←00001000000B (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE             ; IXEフラグ←1
        ADD   MEM02F, #05H    ; IX                    00001000000B (0.40H)
                              ; バンク・オペランド OR) 00000101111B (0.2FH)
                              ; 指定アドレス           00001101111B (0.6FH)
```

例3

0.2FH番地の内容に5を加算した結果を0.2FH番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 0, IXL = 0すなわちIX = 0.00Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.2FH番地を指定することができます。

(0.2FH) ← (0.2FH) +05H
　　└インデクス・レジスタの内容0.00Hとデータ・メモリのアドレス0.2FHをOR演算したアドレス

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ; データ・メモリのバンクを0
        MOV   IXH, #00H       ; IX←00000000000B
        MOV   IXM, #00H
        MOV   IXL, #00H
        SET1  IXE             ; IXEフラグ←1
        ADD   MEM02F, #05H    ; IX                    00000000000B (0.00H)
                              ; バンク・オペランド OR) 00000101111B (0.2FH)
                              ; 指定アドレス          00000101111B (0.2FH)
```

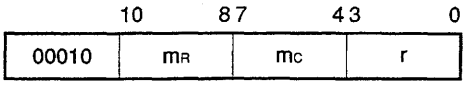
④ 注　意

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



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

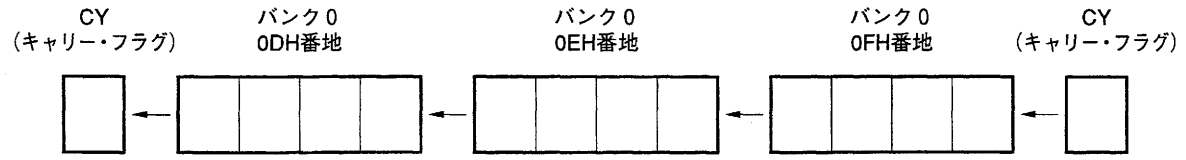
ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H-0.0FH）が指定されているとき，0.0DH番地から0.0FH番地の12ビットの内容に0.2DH番地から0.2FH番地の12ビットの内容を加算した結果を，0.0DH番地から0.0FH番地の12ビットに格納します。

（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

10	8 7	4 3	0
10010	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進演算とBCD演算の2種類があり，どちらの演算を行うかをPSWORDのBCDフラグによって指定します。

③　例1

0.0DH番地から0.0FH番地の12ビットの内容に5を加算した結果を0.0DH番地から0.0FH番地に格納します。

```
                (0.0FH) ← (0.0FH) +05H
                (0.0EH) ← (0.0EH) +CY
                (0.0DH) ← (0.0DH) +CY
MEM00D  MEM     0.0DH
MEM00E  MEM     0.0EH
MEM00F  MEM     0.0FH
        MOV     BANK, #00H          ;データ・メモリのバンクを0
        ADD     MEM00F, #05H
        ADDC    MEM00E, #00H
        ADDC    MEM00D, #00H
```

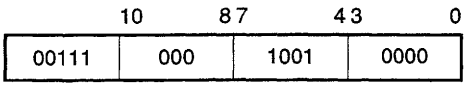
例2

0.4DH番地から0.4FH番地の12ビットの内容に5を加算した結果を0.4DH番地から0.4FH番地に格納します。

```
                (0.4FH) ← (0.4FH) +05H
                (0.4EH) ← (0.4EH) +CY
                (0.4DH) ← (0.4DH) +CY
MEM00D  MEM     0.0DH
MEM00E  MEM     0.0EH
MEM00F  MEM     0.0FH
        MOV     BANK, #00H          ;データ・メモリのバンクを0
        MOV     IXH, #00H           ;IX←00001000000B（0.40H）
        MOV     IXM, #04H
        MOV     IXL, #00H
        SET1    IXE                 ;IXEフラグ←1
        ADD     MEM00F, #5          ; (0.4FH) ← (0.4FH) +5H
        ADDC    MEM00E, #0          ; (0.4EH) ← (0.4EH) +CY
        ADDC    MEM00D, #0          ; (0.4DH) ← (0.4DH) +CY
```

（5）INC AR　　Increment address register

① 命令コード



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，**9.2.3　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
        ORG     10H
        DW      0F3FFH
        DW      0A123H
        DW      0FFF1H
        DW      0FFF5H
        DW      0FF11H
        ⋮
        MOV     AR3, #0H        ;テーブル・データのアドレス
```

```
        MOV     AR2, #0H        ;0010Hをアドレス・レジスタに設
        MOV     AR1, #1H        ;定します
        MOV     AR0, #0H
LOOP:
        MOVT    DBF, @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（AR0-AR3）は，製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

（6）INC IX　　Increment indexregister

① 命令コード

10	87	43	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

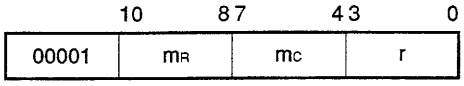
インデクス・レジスタを用いてデータ・メモリ0.00H-0.73Hの内容をすべて“0”にします。

```
                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
                SUBC    IXM, #07H       ;なったかど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
                BR      RAMクリア       ;るまでループします
```

15.5.2　減算命令

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

① 命令コード



② 機　能

CMP = 0のとき　　(r) ← (r) − (m)

ジェネラル・レジスタの内容からデータ・メモリの内容を減算し，結果をジェネラル・レジスタへ格納します。

CMP = 1のとき　　(r) − (m)

結果はレジスタに格納されず，キャリー・フラグCYとゼロ・フラグZが結果によって変化します。

減算の結果，ボローがあればキャリー・フラグCYをセットし，ボローがなければキャリー・フラグCYをリセットします。

減算の結果がゼロ以外になったときは，コンペア・フラグCMPに関係なくゼロ・フラグZはリセットされます。

コンペア・フラグがリセットされた状態（CMP = 0）で減算の結果がゼロになったときは，ゼロ・フラグZがセットされます。

コンペア・フラグがセットされた状態（CMP = 1）で減算の結果がゼロになったときは，ゼロ・フラグZは変化しません。

減算には2進4ビット演算とBCD演算の2種類があり，どちらの演算を行うかをプログラム・ステータス・ワードPSWORDのBCDフラグによって指定します。

③ 例1

ジェネラル・レジスタとしてバンク0のロウ・アドレス0（0.00H-0.0FH）が指定されているとき（RPH = 0, RPL = 0），0.03H番地の内容から0.2FH番地の内容を減算した結果を0.03H番地に格納します。

(0.03H) ← (0.03H) + (0.2FH)

```
MEM003   MEM    0.03H
MEM02F   MEM    0.2FH
         SUB    MEM003, MEM02F
```

例2

ジェネラル・レジスタとしてバンク0のロウ・アドレス2（0.20H-0.2FH）が指定されているとき（RPH = 0, RPL = 4），0.23H番地の内容から0.2FH番地の内容を減算した結果を0.23H番地に格納します。

（0.23H）←（0.23H）－（0.2FH）

```
MEM023  MEM   0.23H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを0
        MOV   RPL, #04H       ;ジェネラル・レジスタのロウ・アドレスを2
        SUB   MEM023, MEM02F
```

例3

0.03H番地の内容から0.6FH番地の内容を減算した結果を0.03H番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.6FH番地を指定することができます。

（0.03H）←（0.03H）＋（0.6FH）

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ;データ・メモリのバンクを0
        MOV   RPH, #00H       ;ジェネラル・レジスタのバンクを0
        MOV   RPL, #00H       ;ジェネラル・レジスタのロウ・アドレスを0
        MOV   IXH, #00H       ;IX←00001000000B（0.40H）
        MOV   IXM, #04H       ;
        MOV   IXL, #00H       ;
        SET1  IXE             ;IXEフラグ←1
        SUB   MEM003, MEM02F  ;IX                    00001000000B（0.40H）
                              ;バンク・オペランド OR) 00000101111B（0.2FH）
                              ;指定アドレス          00001101111B（0.6FH）
```

例4

0.03H番地の内容から0.3FH番地の内容を減算した結果を0.03H番地に格納します。このとき，IXE = 1, IXH = 0, IXM = 1, IXL = 0すなわちIX = 0.10Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリの0.3FH番地を指定することができます。

（0.03H）←（0.03H）＋（0.3FH）

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
```

```
MOV    BANK, #00H        ;データ・メモリのバンクを0
MOV    RPH, #00H         ;ジェネラル・レジスタのバンクを0
MOV    RPL, #00H         ;ジェネラル・レジスタのロウ・アドレスを0
MOV    IXH, #00H         ;IX←00000010000B（0.10H）
MOV    IXM, #01H         ;
MOV    IXL, #00H         ;
SET1   IXE               ;IXEフラグ←1
SUB    MEM003, MEM02F    ;IX                    00000010000B（0.10H）
                         ;バンク・オペランド  OR) 00000101111B（0.2FH）
                         ;指定アドレス          00000111111B（0.3FH）
```

④ 注　意

SUB r, m命令の第1オペランドはジェネラル・レジスタ・アドレスでなければなりません。したがって，次のように書くと03H番地がレジスタとして指定されます。

```
MEM013   MEM    0.13H
MEM02F   MEM    0.2FH
         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	mB	mC	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)
```

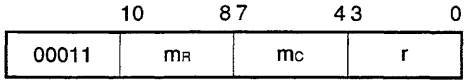
④ 注　意

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



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）
```

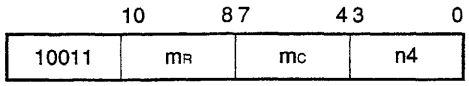
例3

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



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

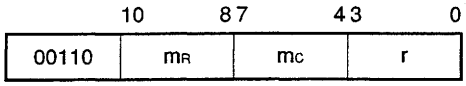
0.00H番地から0.03H番地の12ビットの内容とイミーディエト・データの0A3FHを比較し，同一であればLAB1へジャンプし，異なればLAB2へジャンプします。

```
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：        ⋮
```

15.5.3　論理演算命令

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

① 命令コード



② 機　能

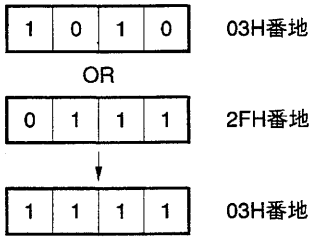
(r) ← (r) ∨ (m)

ジェネラル・レジスタの内容とデータ・メモリの内容との論理和（OR）の結果をジェネラル・レジスタへ格納します。

③ 例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。

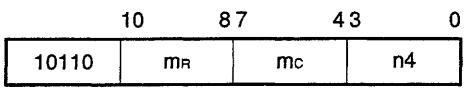
(0.03H) ← (0.03H) ∨ (0.2FH)




```
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

0.03H番地

1	×	×	×

×：don't care

```
MEM003   MEM    0.03H
         OR     MEM003, #1000B
```

例2

0.03H番地のすべてのビットをセットします。

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

または

```
MEM003   MEM    0.03H
         MOV    MEM003, #0FH
```

(3) AND r, m　　AND between general register and data memory

① 命令コード

10	87	43	0
00100	mR	mC	r

② 機　　能

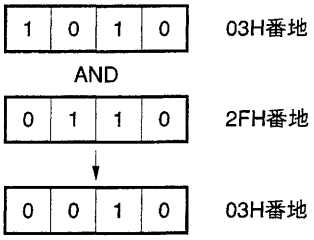
(r) ← (r) ∧ (m)

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。

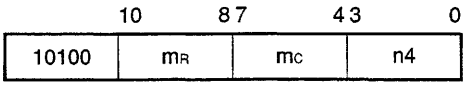
(0.03H) ← (0.03H) ∧ (0.2FH)




```
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



② 機　能

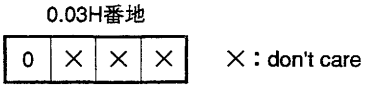
（m）←（m）∧n4

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




```
MEM003   MEM    0.03H
         AND    MEM003, #0111B
```

例2

0.03H番地のすべてのビットをリセットします。

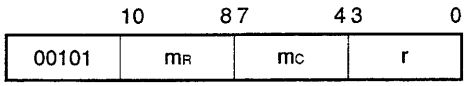
```
MEM003   MEM    0.03H
AND      MEM003, #0000B
```

または

```
MEM003   MEM    0.03H
         MOV    MEM003, #00H
```

（5）XOR r, m　　Exclusive OR between general register and data memory

① 命令コード



② 機　能

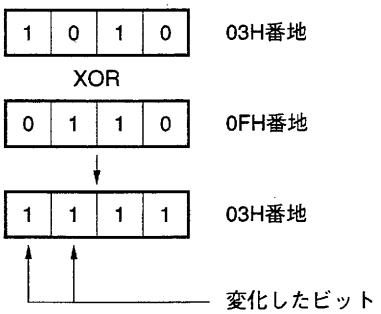
(r) ← (r)　(m)

ジェネラル・レジスタの内容とデータ・メモリの内容との排他的論理和（XOR）の結果をジェネラル・レジスタへ格納します。

③ 例1

0.03H番地の内容と0.0FH番地の内容を比較した結果，異なるビットをセットして0.03H番地へ格納し，0.03H番地がすべてリセット（0.03H番地と0.0FH番地の内容が同じ）されていればLBL1へジャンプし，それ以外であればLBL2へジャンプします。

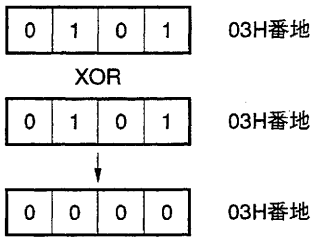
この例はオルタネート・スイッチの状態（0.03H番地の内容）と内部状態（0.0FH番地の内容）を比較し，変化したスイッチの処理へと分岐するときなどの例です。




```
MEM003  MEM   0.03H
MEM00F  MEM   0.0FH
        XOR   MEM003, MEM00F
        SKNE  MEM003, #00H
        BR    LBL1
        BR    LBL2
```

例 2

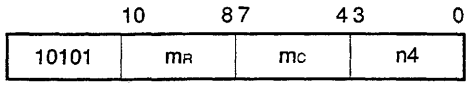
0.03H番地の内容をクリアします。




```
MEM003   MEM    0.03H
         XOR    MEM003, MEM003
```

（6）XOR m, #n4　　Exclusive OR between data memory and immediate data

① 命令コード



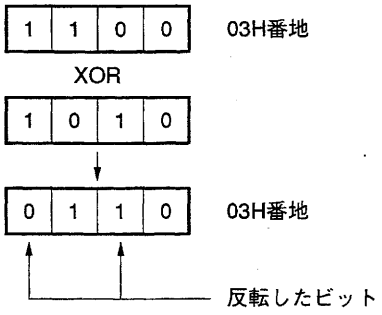
② 機　能

(m) ← (m) ∀ n4

データ・メモリの内容とイミーディエト・データとの排他的論理和（XOR）の結果をデータ・メモリへ格納します。

③ 例

0.03H番地のビット 1 とビット 3 を反転して03H番地へ格納します。

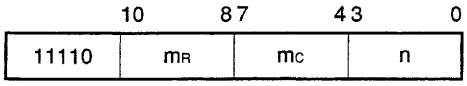



```
MEM003   MEM    0.03H
         XOR    MEM003, #1010B
```

15.5.4　判断命令

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

① 命令コード



② 機　能

CMP←0, if（m）∧n = n, then skip

データ・メモリの内容とイミーディエト・データnの論理積の結果がnと等しければ次の1命令をスキップします（NOP命令として実行します）。

③ 例1

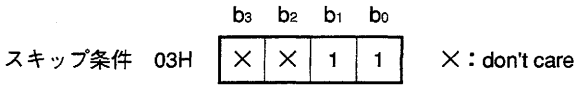
03H番地のビット0が“1”ならばAAAへジャンプし，“0”ならばBBBへジャンプします。

```
SKT     03H, #0001B
BR      BBB
BR      AAA
```

例2

03H番地のビット0とビット1がともに“1”ならば次の命令をスキップします。

```
SKT     03H, #0011B
```



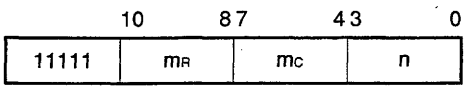
例3

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



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

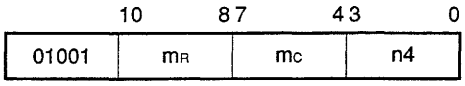
次の2つの命令の実行結果は同じです。

```
SKF     34H, #1111B
SKE     34H, #00H
```

15.5.5　比較命令

（1）SKE m, #n4　　Skip if data memory equal to immediate data

① 命令コード



② 機　能

（m）−n4, skip if zero

データ・メモリの内容がイミーディエト・データの値と等しいとき，次に続く1命令をスキップします（NOP命令として実行します）。

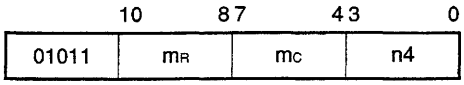
③ 例

24H番地の内容が0ならば24H番地に0FHを転送し，0でなければOPE1へジャンプします。

```
MEM024  MEM   0.24H
        SKE   MEM024, #00H
        BR    OPE1
        MOV   MEM024, #0FH
OPE1：
```

（2）SKNE m, #n4　　Skip if data memory not equal to immediate data

① 命令コード



② 機　能

（m）−n4, skip if not zero

データ・メモリの内容がイミーディエト・データの値と異なるとき，次に続く1命令をスキップします（NOP命令として実行します）。

③ 例

1FH番地の内容が1で，かつ1EH番地の内容が3ならばXYZへジャンプし，そうでなければABCへジャンプします。

8ビットの比較をする場合は以下のように組み合わせて使います。

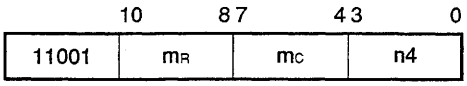

```
MEM01E  MEM   0.1EH
MEM01F  MEM   0.1FH
        SKNE  MEM01F, #01H
        SKE   MEM01E, #03H
        BR    ABC
        BR    XYZ
```

また，コンペア・フラグ，ゼロ・フラグを用いて上記の内容と同じことを行う場合は，以下のようになります。

```
MEM01E  MEM   0.1EH
MEM01F  MEM   0.1FH
        SET2  CMP, Z            ; CMPフラグ←1，Zフラグ←1
        SUB   MEM01F, #01H
        SUB   MEM01E, #03H
        SKT1  Z
        BR    ABC
        BR    XYZ
```

（3）SKGE m, #n4　　Skip if data memory greater than or equal to immediate data

① 命令コード



② 機　能

（m）−n4, skip if not borrow

データ・メモリの内容がイミーディエト・データの値以上のとき，次に続く1命令をスキップします（NOP命令として実行します）。

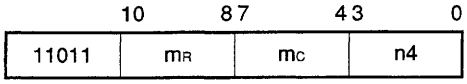
③ 例

1FH番地（上位）および2FH番地（下位）に格納されている8ビット・データがイミーディエト・データ17Hより大きければRETし，そうでなければRETSKするプログラム例。

```
MEM01F  MEM   0.1FH
MEM02F  MEM   0.2FH
        SKGE  MEM01F, #1
        RETSK
        SKNE  MEM01F, #1
        SKLT  MEM02F, #8        ; 7+1
        RET
        RETSK
```

（4）SKLT m, #n4　　Skip if data memory less than immediate data

①　命令コード



②　機　　能

（m）−n4, skip if borrow

データ・メモリの内容がイミーディエト・データの値未満のとき，次に続く1命令をスキップします（NOP命令として実行します）。

③　例

10H番地の内容がイミーディエト・データ“6”以上であれば，0FH番地の内容に01Hを格納し，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
```

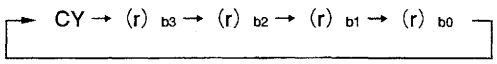
15.5.6　回転命令

（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

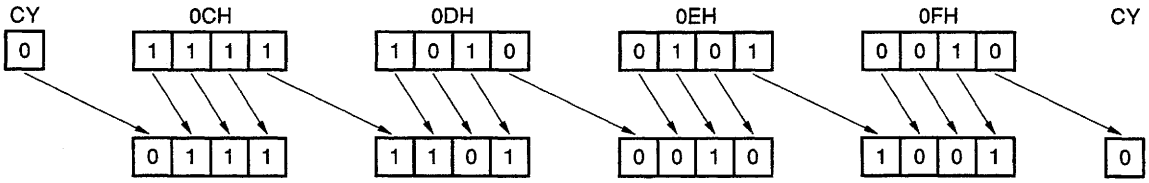
ジェネラル・レジスタとしてバンク 0 のロウ・アドレス 0（0.00H-0.0FH）が指定されているとき（RPH = 0, RPL = 0），0.00H番地の値（1000B）を右に 1 ビット回転し，0100Bにします。

（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。

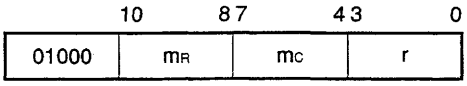


```
MEM00C  MEM   0.0CH
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   RPH, #00H       ; ジェネラル・レジスタのバンクを 0
        MOV   RPL, #00H       ; ジェネラル・レジスタのロウ・アドレスを 0
        CLR1  CY              ; CYフラグ← 0
        RORC  MEM00C
        RORC  MEM00D
        RORC  MEM00E
        RORC  MEM00F
```

15.5.7　転送命令

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

① 命令コード



② 機　能

(r) ← (m)

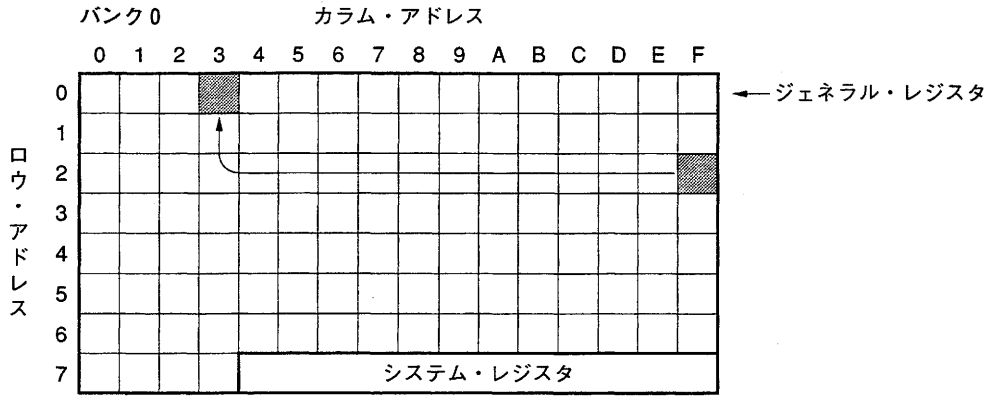
データ・メモリの内容をジェネラル・レジスタへ格納します

③ 例1

0.03H番地に0.2FH番地の内容を格納します。

(0.03H) ← (0.2FH)

```
MEM003   MEM   0.03H
MEM02F   MEM   0.2FH
         MOV   RPH, #00H        ;ジェネラル・レジスタのバンクを0
         MOV   RPL, #00H        ;ジェネラル・レジスタのロウ・アドレスを0
         LD    MEM003, MEM02F
```



例2

0.6FH番地の内容を0.03H番地に格納します。このときIXE = 1, IXH = 0, IXM = 4, IXL = 0すなわちIX = 0.40Hなら，データ・メモリ・アドレスを2FHとすることでデータ・メモリ0.6FH番地を指定することができます。

IXH←00H

IXM←04H

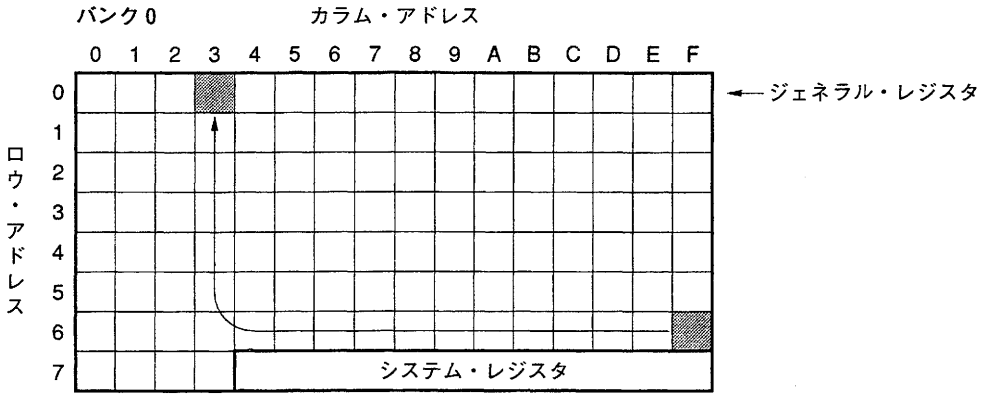
IXL←00H

IXEフラグ←1

(0.03H) ← (0.6FH)

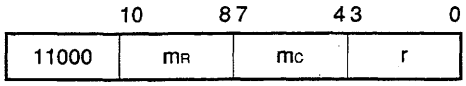
└── インデクス・レジスタの内容040Hとデータ・メモリの0.2FHをOR演算したアドレス

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   IXH, #00H          ; IX←00001000000B (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE                ; IXEフラグ←1
        LD    MEM003, MEM02F
```



(2) ST m, r　　Store general register to data memory

① 命令コード



② 機　能

(m) ← (r)

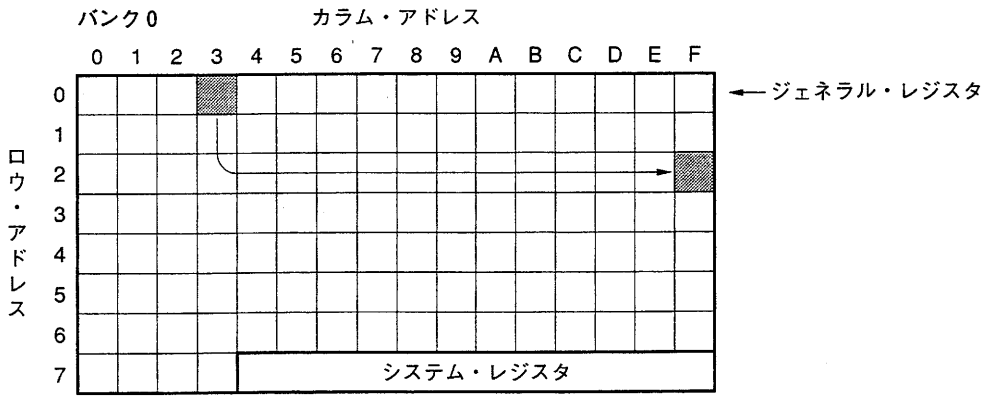
ジェネラル・レジスタの内容をデータ・メモリへ格納します。

③ 例1

0.2FH番地に0.03H番地の内容を格納します。

(0.2FH) ← (0.03H)

```
MOV     RPH, #00H        ; ジェネラル・レジスタのバンクを0
MOV     RPL, #00H        ; ジェネラル・レジスタのロウ・アドレスを0
ST      2FH, 03H         ; データ・メモリにジェネラル・レジスタの内容を転送
```

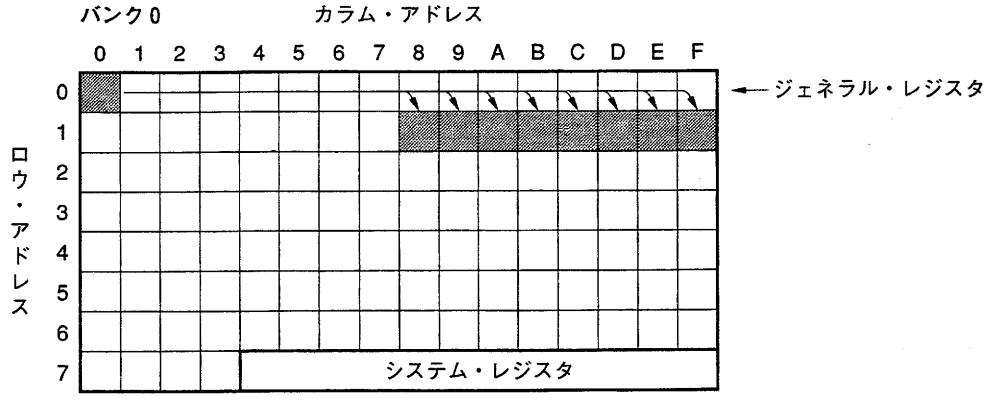


例2

0.18H番地から0.1FH番地に0.00H番地の内容を格納します。インデクス・レジスタでデータ・メモリ（18H-1FH番地）を指定します。

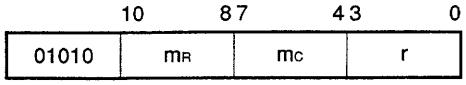
(0.18H) ← (0.00H)
(0.19H) ← (0.00H)
⋮
(0.1FH) ← (0.00H)

```
        MOV   IXH, #00H      ; IX←00000000000B (0.00H)
        MOV   IXM, #00H
        MOV   IXL, #00H      ; データ・メモリに0.00H番地を指定
MEM018  MEM   0.18H
MEM000  MEM   0.00H
LOOP1 :
        SET1  IXE            ; IXEフラグ← 1
        ST    MEM018, MEM000 ; (0.1×H) ← (0.00H)
        CLR1  IXE            ; IXEフラグ← 0
        INC   IX             ; IX←IX＋1
        SKGE  IXL, #08H
        BR    LOOP1
```



(3) MOV @r, m　　Move data memory to destination indirect

① 命令コード



② 機　能

MPE = 1のとき

(MP, (r)) ← (m)

MPE = 0のとき

(BANK, mR, (r)) ← (m)

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

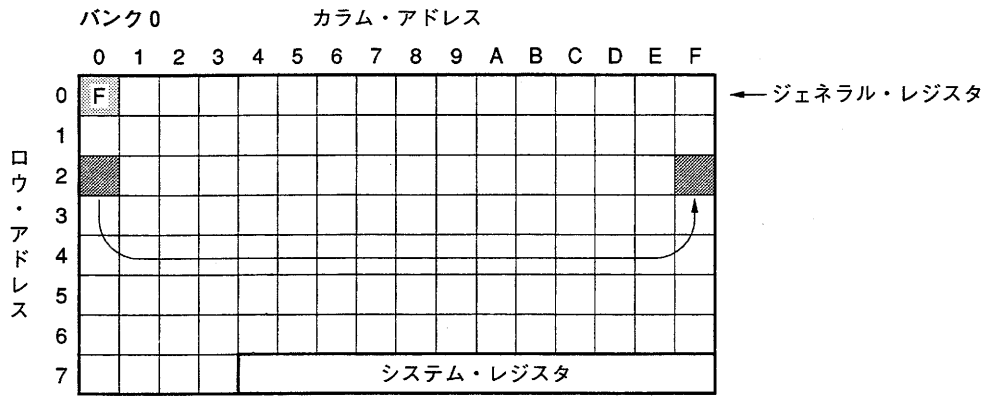
MPE = 0のときには，同一バンクの同一ロウ・アドレス内での転送となります。

③ 例1

MPEフラグを0にして，0.2FH番地に0.20H番地の内容を格納します。格納先のデータ・メモリは，転送元と同一のロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

(0.2FH) ← (0.20H)

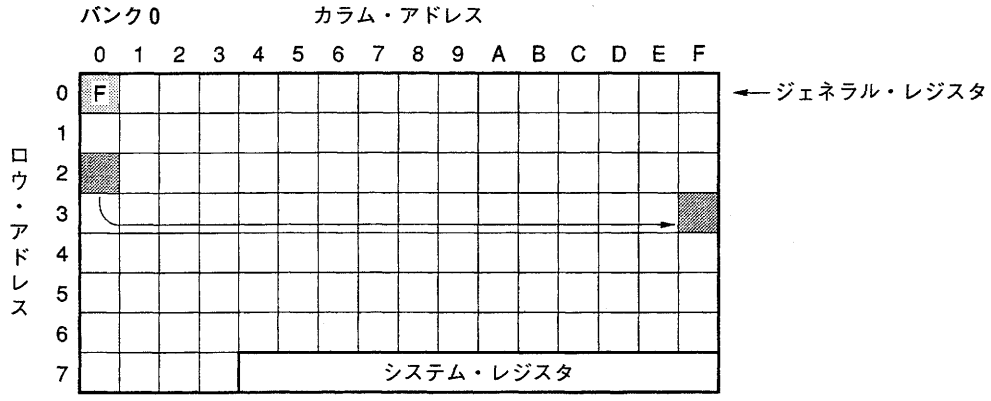
```
MEM000  MEM   0.00H
MEM020  MEM   0.20H
        CLR1  MPE              ; MPEフラグ←0
        MOV   MEM000, #0FH     ; ジェネラル・レジスタにカラム・アドレス設定
        MOV   @MEM000, MEM020  ; 格納
```



例2

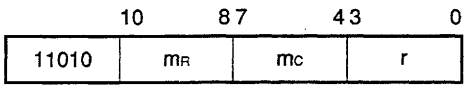
MPEフラグを1にして，0.3FH番地に0.20H番地の内容を格納します。格納先のデータ・メモリは，メモリ・ポインタMPの内容がロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

```
                (0.3FH) ← (0.20H)
MEM000  MEM     0.00H
MEM020  MEM     0.20H
        MOV     RPH, #00H          ; ジェネラル・レジスタのバンクを0
        MOV     RPL, #00H          ; ジェネラル・レジスタのロウ・アドレスを0
        MOV     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     @MEM000, MEM020    ; 格納
```



(4) MOV m, @r　　Move data memory to destination indirect

① 命令コード



② 機　能

MPE = 1のとき

(m) ← (MP, (r))

MPE = 0のとき

(m) ← (BANK, mR, (r))

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

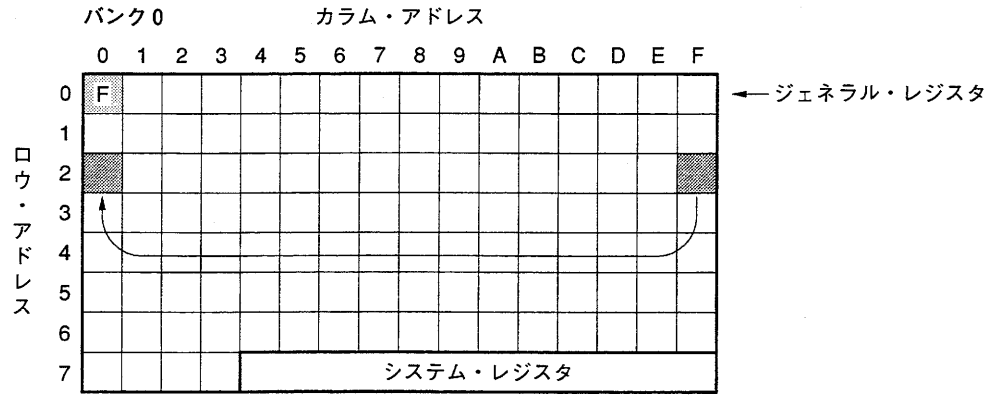
MPE = 0のときには，同一バンクの同一ロウ・アドレス内での転送となります。

③ 例1

MPEフラグを0にして，0.20H番地に0.2FH番地の内容を格納します。転送元のデータ・メモリは，格納先と同一のロウ・アドレスで，ジェネラル・レジスタの0.00H番地の内容がカラム・アドレスです。

（0.20H）←（0.2FH）

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

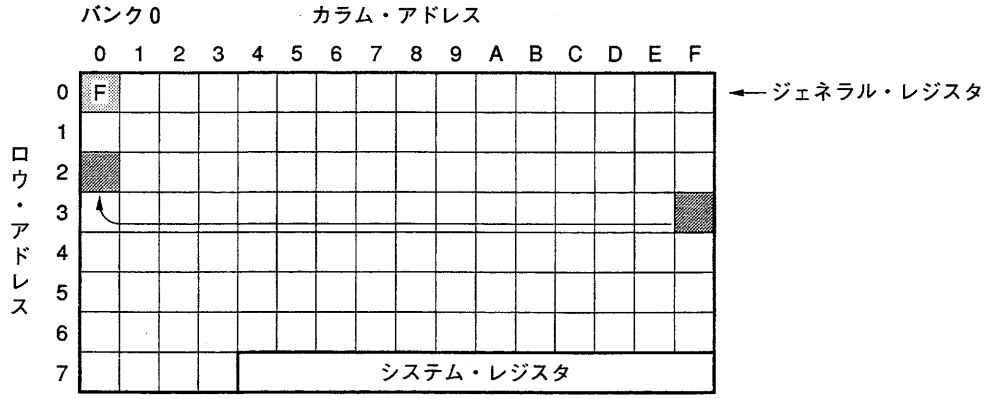


例2

MPEフラグを1にして，0.20H番地に0.3FH番地の内容を格納します。転送元のデータ・メモリは，メモリ・ポインタMPの内容がロウ・アドレスで，ジェネラル・レジスタの0.00番地の内容がカラム・アドレスです。

（0.20H）←（0.3FH）

```
MEM000  MEM   0.00H
MEM020  MEM   0.20H
        MOV   MEM000, #0FH     ; ジェネラル・レジスタにカラム・アドレス設定
        MOV   MPH, #00H        ; メモリ・ポインタにロウ・アドレスを設定
        MOV   MPL, #03H        ;
        SET1  MPE              ; MPEフラグ←1
        MOV   MEM020, @MEM000  ; 格納
```



（5）MOV m, #n4　　Move immediate data to data memory

① 命令コード

10	87	43	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	87	43	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0011Hを設定
                MOV   AR2, #00H          ;AR2←00H
                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

10	87	43	0
00111	000	1101	0000

② 機　能

SP←SP−1,

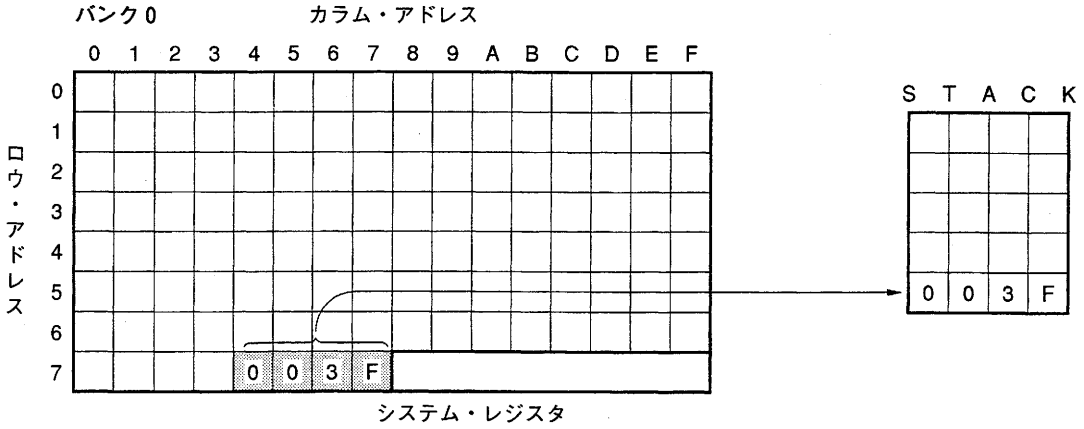
ASR←AR

スタック・ポインタSPをデクリメントしたあと，スタック・ポインタで指定されるアドレス・スタック・レジスタにアドレス・レジスタARの値を格納します。

③ 例1

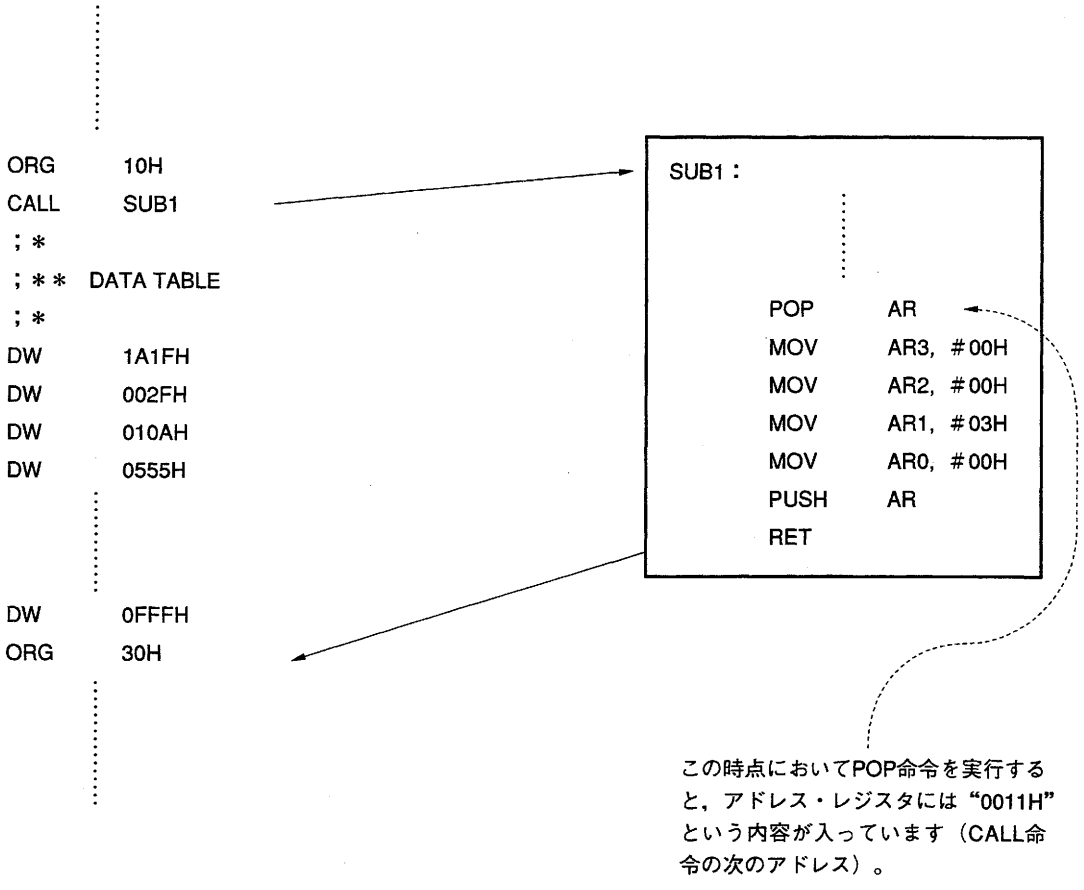
アドレス・レジスタに003FHを設定し，スタックに格納します。

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



例2

CALL命令に続いてデータ・テーブルがある場合にサブルーチンからの戻り番地（データ・テーブルの次の番地）をアドレス・レジスタに設定しリターンします。



（8）POP AR　　Pop address register

① 命令コード

00111	000	1100	0000

② 機　　能

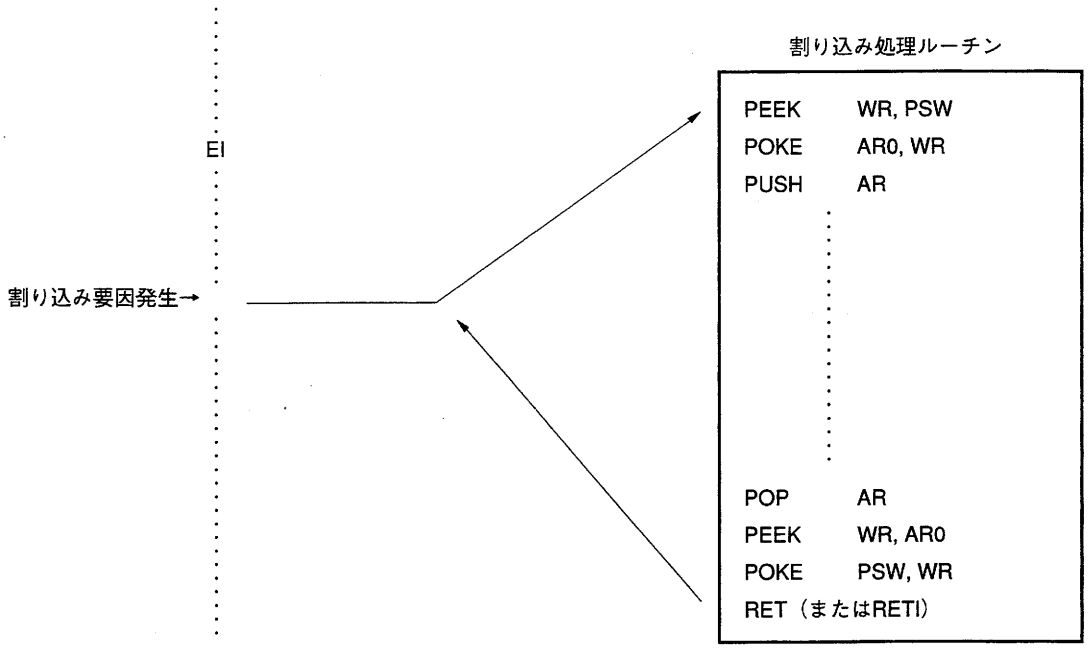
AR←ASR,

SP←SP＋1

スタック・ポインタで示されるアドレス・スタック・レジスタの内容をアドレス・レジスタARに取り出したあと，スタック・ポインタSPをインクリメントします。

③ 例

割り込み処理を行う場合，割り込み処理ルーチン内にてPSWが変化してしまう場合に，割り込み処理の始めでPSWの内容をWRを介してアドレス・レジスタに転送し，PUSH命令によりアドレス・スタック・レジスタに退避し，リターンする前にPOP命令によりアドレス・レジスタに復帰させWRを介してPSWに転送します。



（9）PEEK WR, rf　　Peek register file to window register

① 命令コード

00111	rf_R	0011	rf_C

② 機　　能

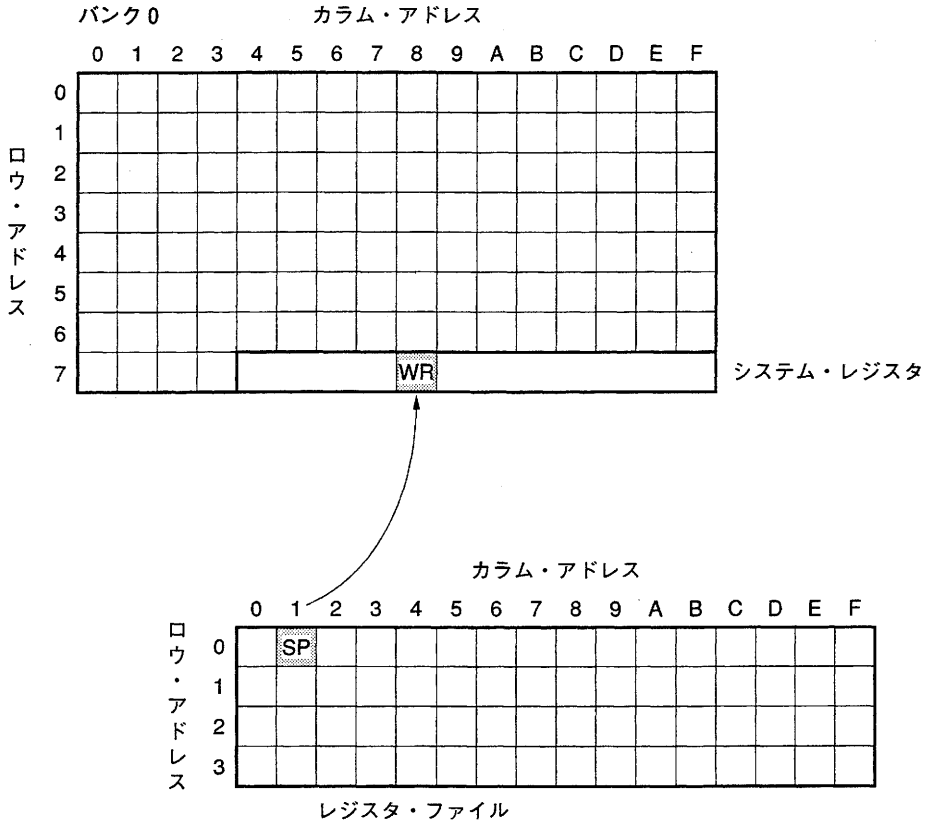
WR← (rf)

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

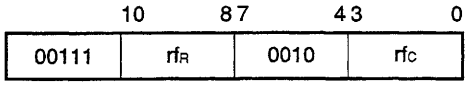
レジスタ・ファイル内の01H番地のスタック・ポインタSPの内容をウインドウ・レジスタに格納します。

PEEK　WR, SP



(10) POKE rf, WR　　Poke window register to register file

① 命令コード



② 機　　能

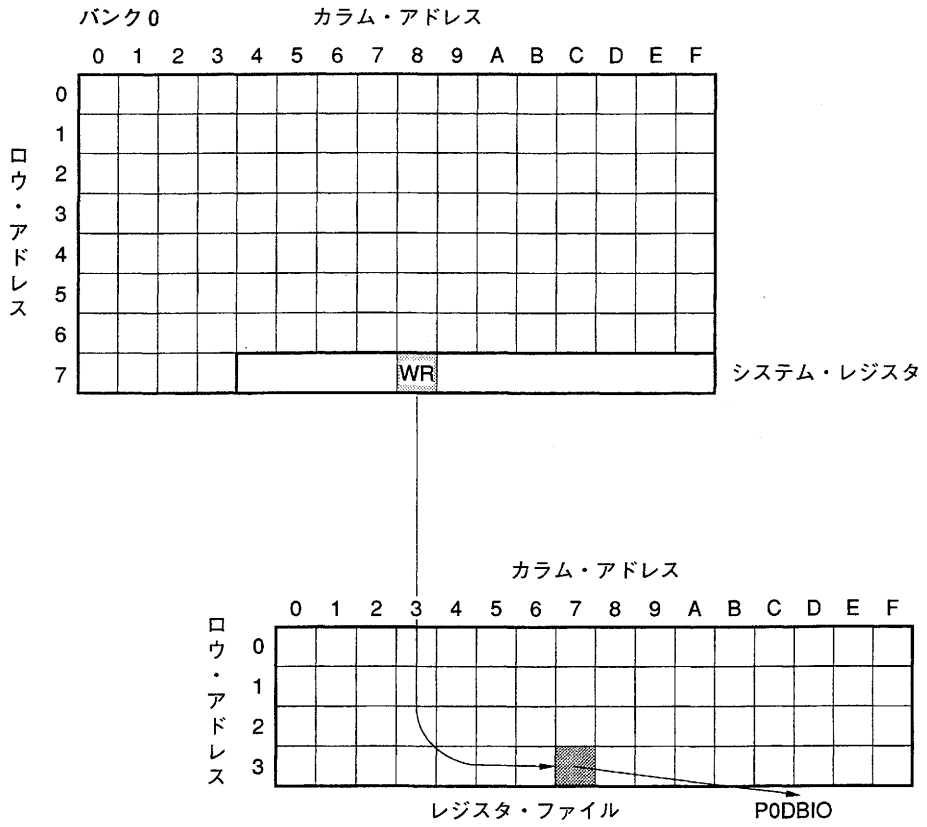
(rf) ←WR

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。

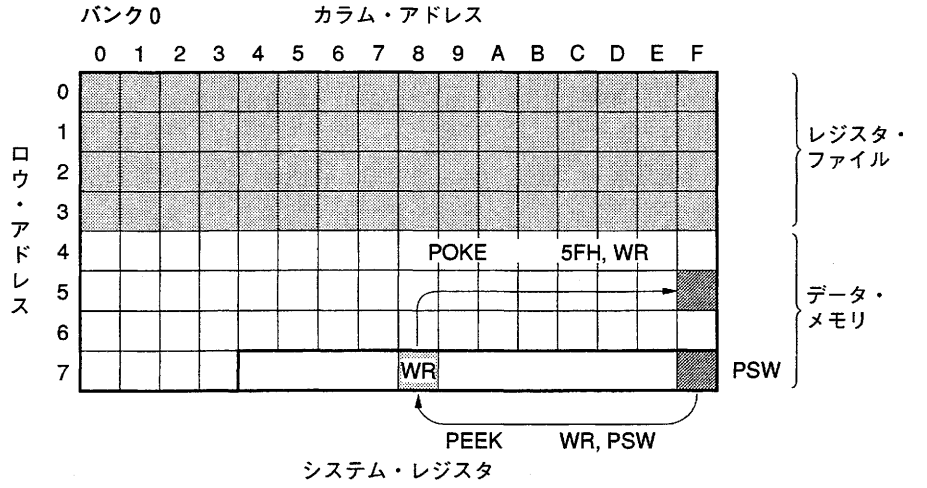
```
MOV     WR, #0FH
POKE    P0DBIO, WR          ; P0D0, P0D1, P0D2, P0D3をすべて出力モードに設定
```



④ 注　意

プログラム上は，レジスタ・ファイルの40Hから7FH番地に，データ・メモリのアドレス40Hから7FH番地と同じメモリがあるように見えます。したがって，PEEK, POKE命令ではレジスタ・ファイル以外にデータ・メモリの各バンクの40H番地から7FH番地までをアクセスすることができます。たとえば次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

10	8 7	4 3	0
00111	pH	1011	pL

② 機　能

DBF←（p）

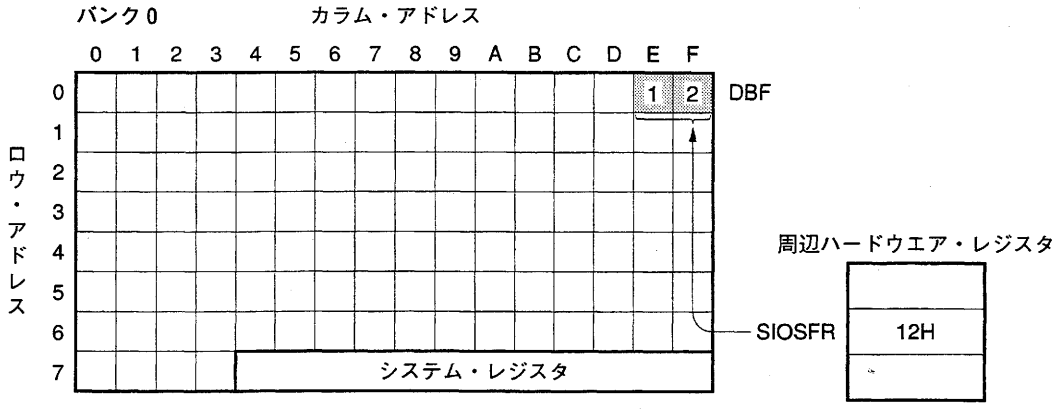
周辺レジスタの内容をデータ・バッファDBFに格納します。

DBFはバンク・レジスタの値に関係なく，データ・メモリのBANK0の0CHから0FH番地の16ビットの領域となります。

③ 例

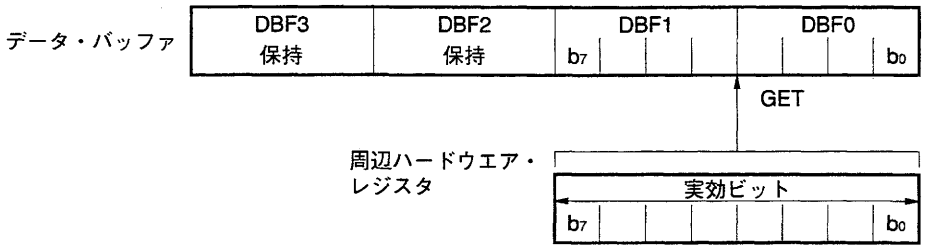
シリアル・インタフェースのシフト・レジスタSIOSFRの内容（8ビット）をデータ・バッファのDBF0, DBF1に格納します。

GET DBF, SIOSFR



④ 注　意

データ・バッファは16ビットで構成されています。ただし，周辺ハードウエアによってアクセスされるビット数は異なります。たとえば実効ビット長が8ビットの周辺ハードウエア・レジスタに対してGET命令を実行した場合は，データ・バッファDBFの下位8ビット（DBF1, DBF0）に周辺ハードウエア・レジスタの内容が格納されます。



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

① 命令コード

10	87	43	0
00111	p_H	1010	p_L

② 機　能

(p) ←DBF

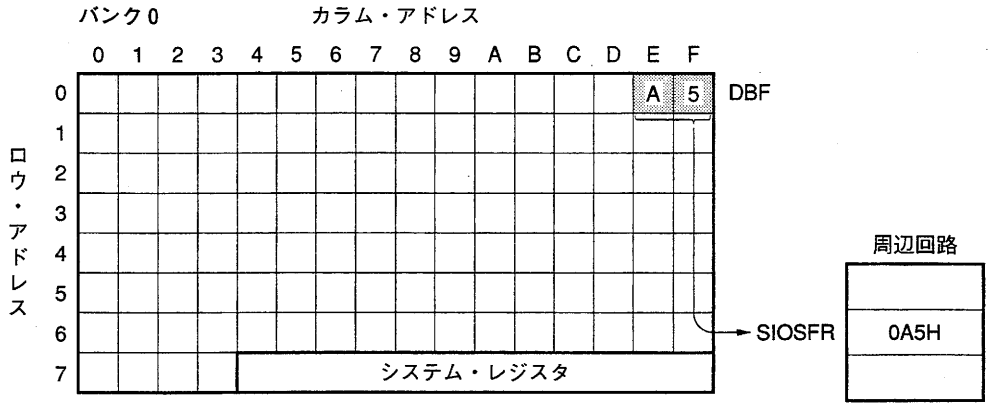
データ・バッファDBFの内容を周辺ハードウエア・レジスタに格納します。

DBFはバンク・レジスタの値に関係なく，データ・メモリのBANK0の0CHから0FH番地の番地の16ビットの領域となります。

③ 例

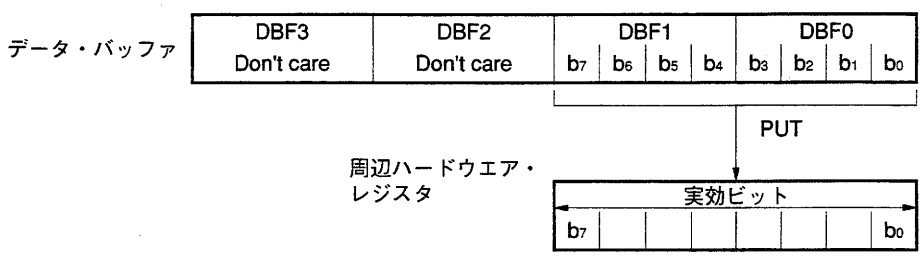
データ・バッファのDBF1とDBF0にそれぞれ0AH, 05Hを設定し，周辺レジスタの1つであるシリアル・インタフェース用のシフト・レジスタ（SIOSFR）に転送します。

```
MOV     BANK, #00H          ；データ・メモリのバンクを0
MOV     DBF0, #05H
MOV     DBF1, #0AH
PUT     SIOSFR, DBF
```



④ 注　意

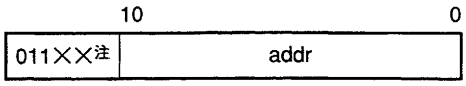
データ・バッファは，16ビットで構成されています。ただし，周辺ハードウエアによってアクセスされるビット数は異なります。たとえば実効ビット長が8ビットの周辺ハードウエア・レジスタに対してPUT命令を実行した場合は，データ・バッファDBFの下位8ビット（DBF1, DBF0）の内容を周辺ハードウエア・レジスタに格納します（DBF3, DBF2の内容は無効です）。



15.5.8　分岐命令

（1）BR addr　　Branch to the address

① 命令コード



注　④注意参照。

② 機　能

$PC_{10-0} \leftarrow addr$

addrで指定するアドレスに分岐します。

③ 例

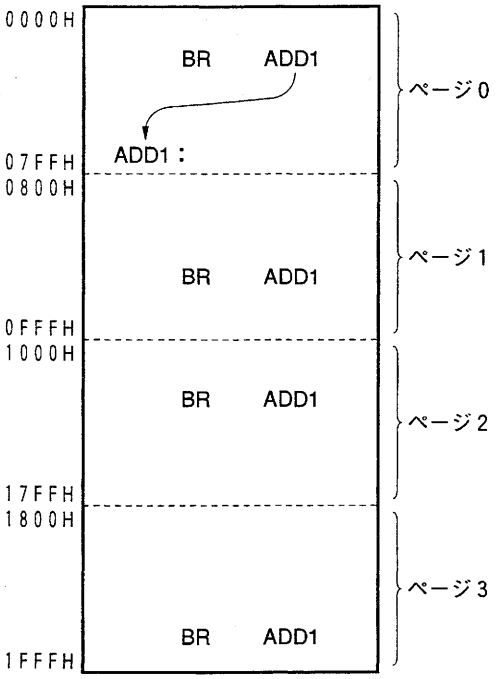
```
FLY     LAB   0FH          ; FLY = 0FHを定義
          :
        BR    FLY          ; 0FH番地にジャンプします
          :
        BR    LOOP1        ; LOOP1にジャンプします
          :
        BR    $+2          ; 現在番地より2つ下の番地へジャンプします
          :
        BR    $-3          ; 現在番地より3つ上の番地へジャンプします
          :
LOOP1：
```

④ 注　意

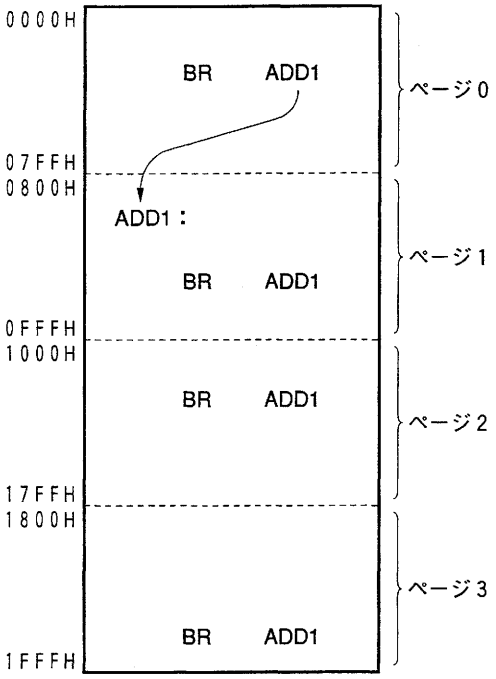
BR命令はアセンブラで記述する上ではページの概念はなく，ROMアドレス0000H～1FFFH番地の間で，同一記述で使用することができます。ただし，ページ0内（0000H～07FFH番地）へのBR命令と，ページ1内（07FFH～0FFFH番地）へのBR命令と，ページ2（1000H～17FFH番地）へのBR命令と，ページ3（17FFH～1FFFH番地）へのBR命令とでは，オペレーション・コードがそれぞれ異なります。

ページ0内でのオペレーション・コードは‘0C’，ページ1内でのオペレーション・コードは‘0D’，ページ2内でのオペレーション・コードは‘0E’，ページ3内でのオペレーション・コードは‘0F’です。これらはアセンブルする際，17Kシリーズのアセンブラを使用すればアセンブラがジャンプ先を参照して自動的に変換します。

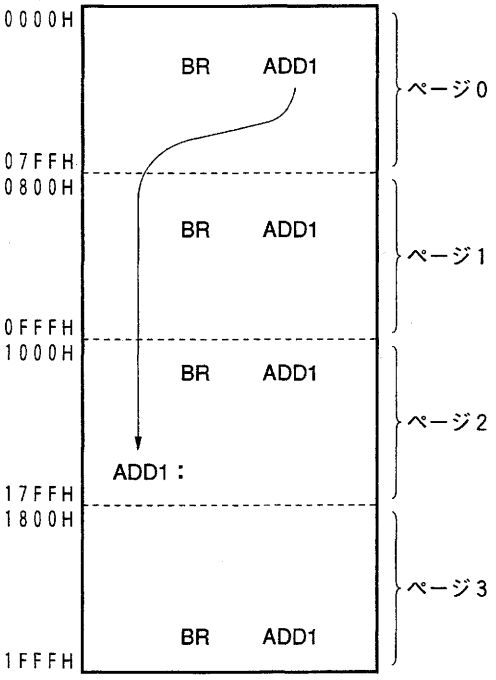
オペレーション・コードが0Cとなる場合
（ジャンプ先のアドレスがページ0内にある場合）



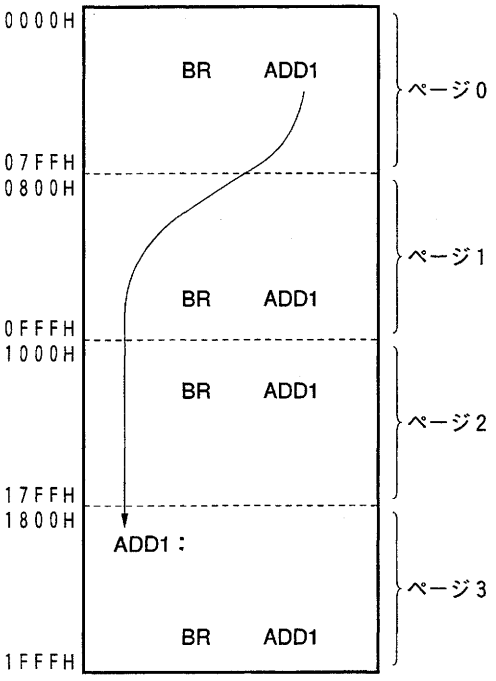
オペレーション・コードが0Dとなる場合
（ジャンプ先のアドレスがページ1内にある場合）



オペレーション・コードが0Eとなる場合
（ジャンプ先のアドレスがページ2内にある場合）

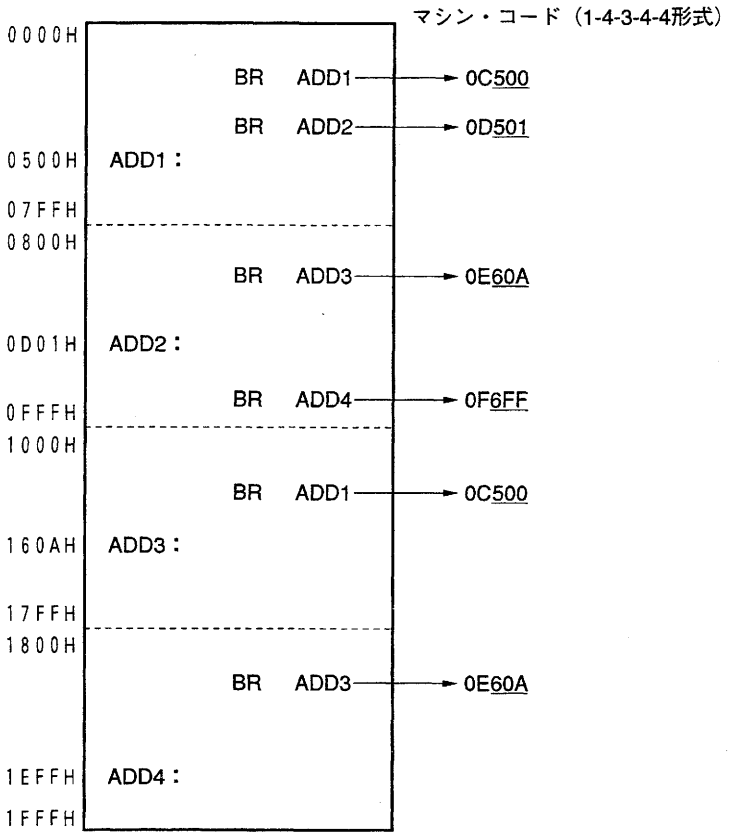


オペレーション・コードが0Fとなる場合
（ジャンプ先のアドレスがページ3内にある場合）



ディバグ時，パッチ修正を行う場合は，プログラム自身で‘0C’，‘0D’，‘0E’，‘0F’のそれぞれの変換を行う必要があります。

また，BR命令のジャンプ先がそれぞれ0000H番地～07FFH番地，0800H番地～0FFFH番地，1000H番地～17FFH番地，1800H番地～1FFFH番地の場合に対してアドレスの変換も必要です。つまり，0000H番地，0800H番地，1000H番地，1800H番地を000H番地として以降1番地ずつインクリメントされたアドレスとなります。



注意　μPD172××サブシリーズの各製品ごとでページ数が異なりますので，使用する各製品ごとのデータ・シートを参照してください。

(2) BR @AR　　Branch to the address specified by address register

① 命令コード

00111	000	0100	0000

② 機　能

PC←AR

アドレス・レジスタARが指定するプログラム・アドレスに分岐します。

③ 例1

アドレス・レジスタAR（AR0-AR3）に003FHを設定し，BR @AR命令により003FH番地へジャンプします。

```
MOV     AR3, #00H       ; AR3←00H
MOV     AR2, #00H       ; AR2←00H
MOV     AR1, #03H       ; AR1←03H
MOV     AR0, #0FH       ; AR0←0FH
BR      @AR             ; 003FH番地へジャンプ
```

例2

データ・メモリの0.10H番地の内容によって以下のように分岐先を変えます。

```
0.10Hの内容          分岐先のレーベル
  00H          →     AAA
  01H          →     BBB
  02H          →     CCC
  03H          →     DDD
  04H          →     EEE
  05H          →     FFF
  06H          →     GGG
  07H          →     HHH
08H-0FH        →     ZZZ
```

```
; *
; **ジャンプ・テーブル
; *
ORG     10H
BR      AAA
BR      BBB
BR      CCC
BR      DDD
BR      EEE
BR      FFF
```

```
        BR      GGG
        BR      HHH
        BR      ZZZ
                :
MEM010  MEM     0.10H
        MOV     AR3, #00H     ; AR3←00H ARを001×Hにする
        MOV     AR2, #00H     ; AR2←00H
        MOV     AR1, #01H     ; AR1←01H
        MOV     RPH, #00H     ; ジェネラル・レジスタのバンクを0
        MOV     RPL, #02H     ; ジェネラル・レジスタのロウ・アドレスを1
        ST      AR0, MEM010   ; AR0←0.10H
        SKLT    AR0, #08H
        MOV     AR0, #08H     ; AR0の内容が08Hよりも大きい場合は，AR0の内容を08Hにする
        BR      @AR
```

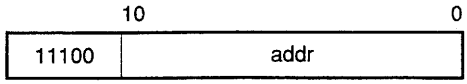
④ 注　意

アドレス・レジスタ（AR0-AR3）は製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

15.5.9　サブルーチン命令

（1）CALL addr　　Call subroutine

① 命令コード



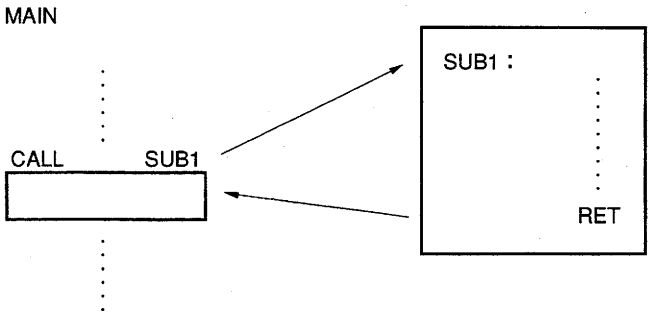
② 機　　能

SP←SP−1, ASR←PC,

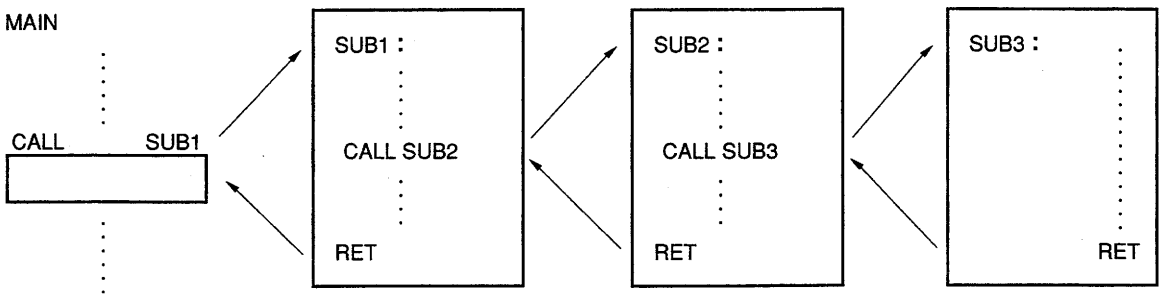
PC_{10-0}←addr, PAGE←0

プログラム・カウンタPCの値をインクリメントし，スタックに格納したのち，addrで指定するサブルーチンに分岐します。

③ 例1



例2



（2）CALL @AR　　Call subroutine specified by address register

① 命令コード

00111	000	0101	0000

② 機　　能

SP←SP−1,

ASR←PC,

PC←AR

プログラム・カウンタPCの値を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

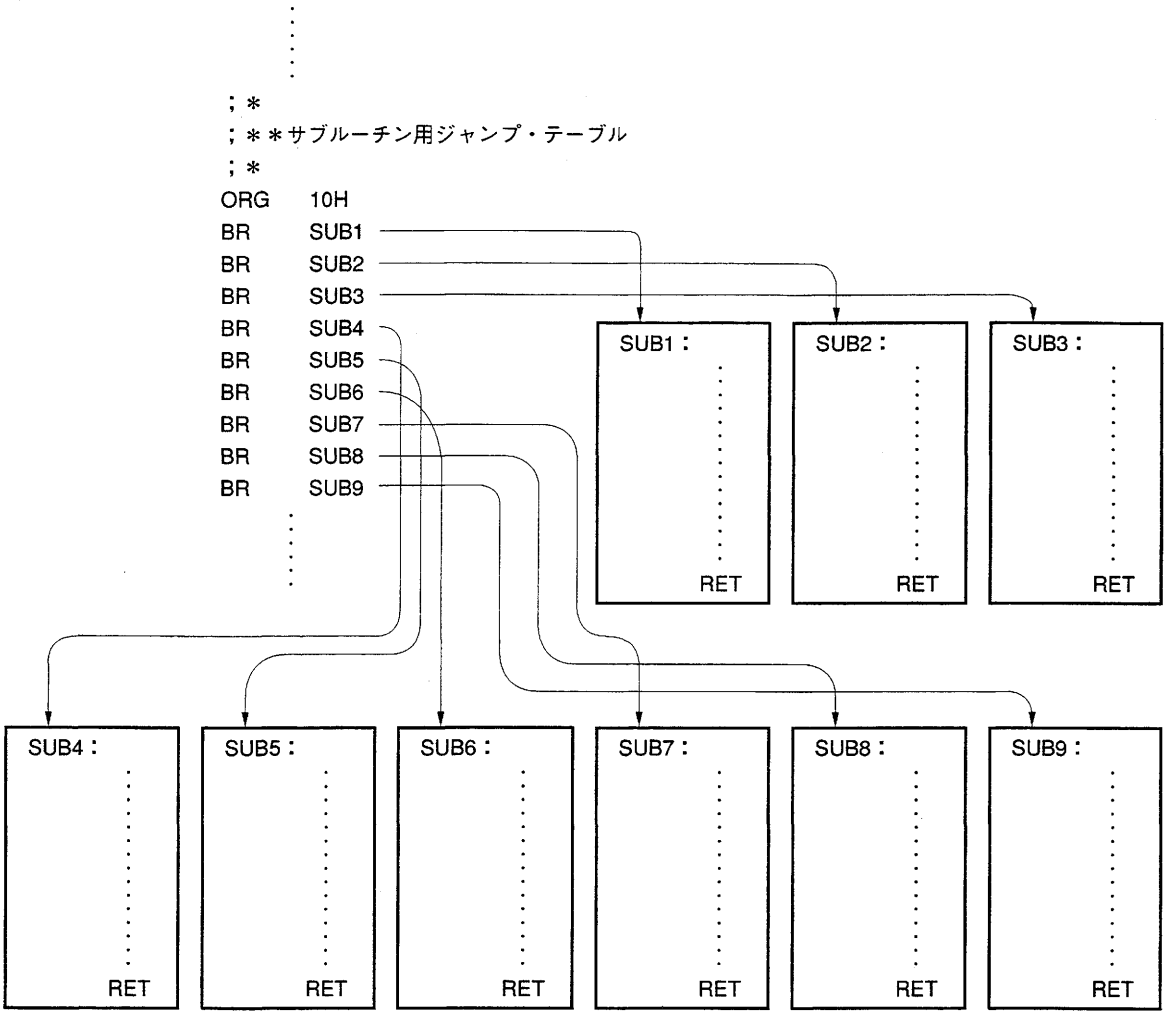
アドレス・レジスタAR（AR0-AR3）に0020Hを設定し，CALL @AR命令により0020H番地のサブルーチンをコールします。

```
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   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
MOV   RPH,   #00H  ;ジェネラル・レジスタのバンクを0
MOV   RPL,   #02H  ;ジェネラル・レジスタのロウ・アドレスを1
ST    AR0,   10H   ;AR0←0.10H
SKLT  AR0,   #08H  ;AR0の内容が08Hより大きい場合は,
MOV   AR0,   #08H  ;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（AR0-AR3）は製品により使用できるビット数が異なりますので，使用する際は，各製品ごとのデータ・シートを参照してください。

（3）RET　　Return to the main program from subroutine

① 命令コード

10	8 7	4 3	0
00111	000	1110	0000

② 機　能

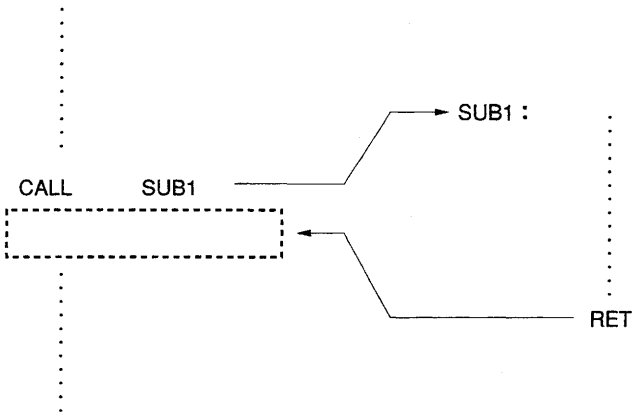
PC←ASR,

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

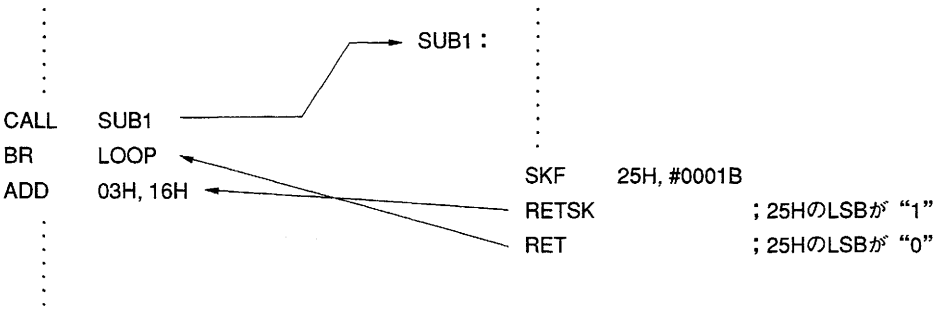
サブルーチンからメイン・プログラムへ戻るための命令です。

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。

15.5.10　割り込み命令

（1）EI　　Enable Interrupt

① 命令コード

00111	000	1111	0000

② 機　　能

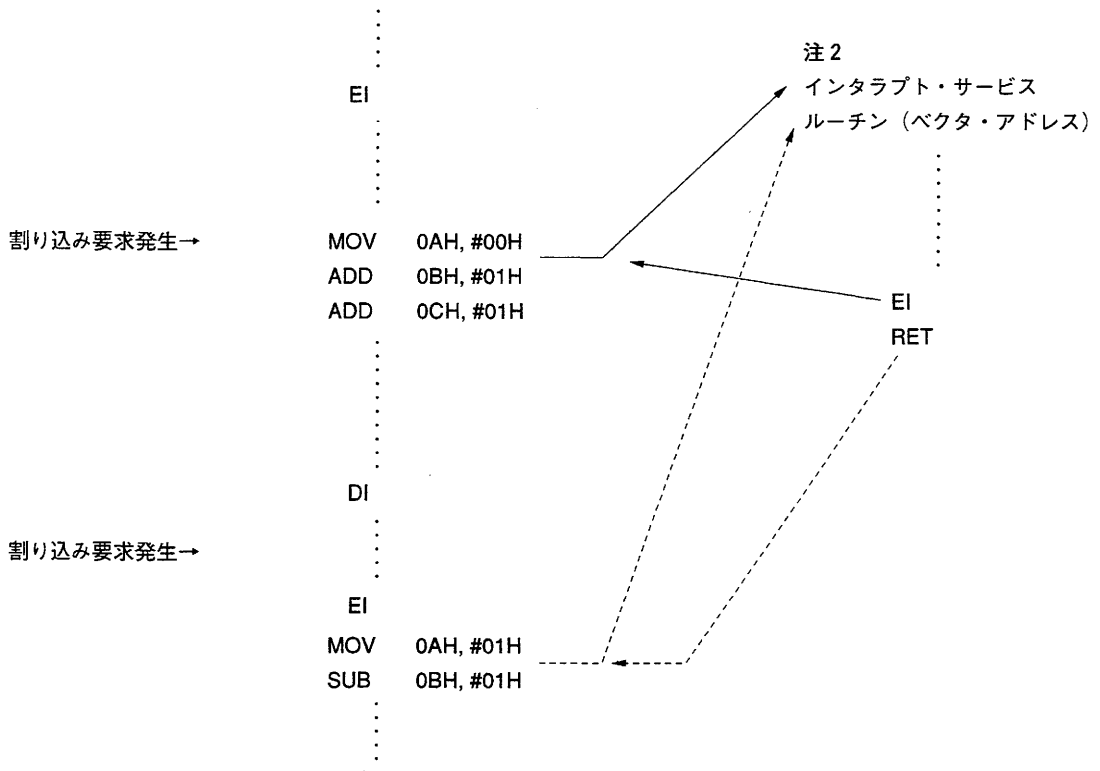
INTEF←1

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]

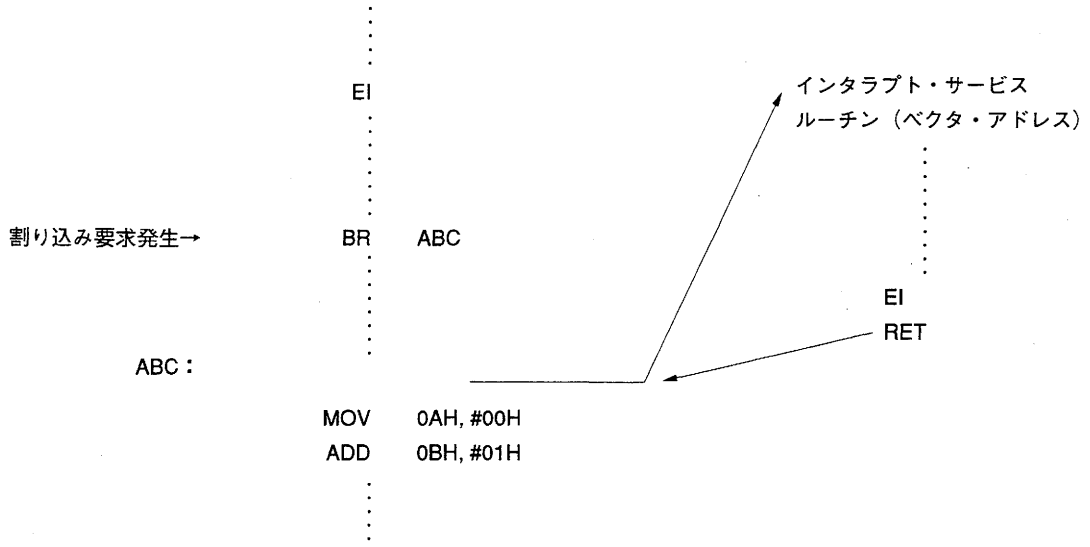


注1．ベクタ・アドレスは，受け付けられる割り込みにより異なります。

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××X）がセットされているものとします。各割り込み許可フラグがセットされていない状態で，EI命令実行後，割り込み要求が発生してもプログラムの流れは変化しません（割り込みは受け付けられません）。ただし，割り込み要求フラグ（IRQ××X）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。

15.5.11　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**12.2　STOPモードの設定と解除**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**12.3　HALTモードの設定と解除**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。

付録A　開発ツール

★ A.1　ハードウエアー覧

対象デバイス		インサーキット・エミュレータ	SEボード	エミュレーション・プローブ	変換ソケット／アダプタ[注1]
デバイス名	形　状				
μPD17201AGF	80ピンQFP	IE-17K	SE-17207	EP-17201GF	EV-9200G-80
μPD17203AGC	52ピンQFP	IE-17K-ET	SE-17204	EP-17203GC	EV-9200G-52
μPD17204GC	52ピンQFP	EMU-17K[注2]			
μPD17207GF	80ピンQFP		SE-17207	EP-17201GF	EV-9200G-80
μPD17225CT	28ピンSDIP		SE-17225	EP-17K28CT	EV-9500GT-28
μPD17225GT	28ピンSOP			EP-17K28GT	（EP-17K28GT用）
μPD17226CT					
μPD17226GT					
μPD17227CT					
μPD17227GT					
μPD17228CT					
μPD17228GT					

注1．EV-9200××-××は変換ソケット，EV-9500××-××はフレキシブル基盤です。

2．内藤電誠町田製作所の製品です。ホスト・マシンは，PC-9800シリーズにのみ対応しています。詳細につきましては，（株）内藤電誠町田製作所（TEL 044-822-3813）までお問い合わせください。

★ A.2　ソフトウエアー覧

●PC-9800シリーズ（日本語版Windows™）

デバイス	アセンブラ	Cコンパイラ	デバイス・ファイル	*SIMPLEHOST*
17201A	μSAA13RA17K	μSAA13CC17K	μSAA13AS17201	μSAA13ID17K
17203A			μSAA13AS17203	
17204			μSAA13AS17204	
17207			μSAA13AS17207	
17225			μSAA13AS17225	
17226				
17227				
17228				

●IBM PC/AT™（日本語版Windows）

デバイス	アセンブラ	Cコンパイラ	デバイス・ファイル	*SIMPLEHOST*
17201A	μSAB13RA17K	μSAB13CC17K	μSAB13AS17201	μSAB13ID17K
17203A			μSAB13AS17203	
17204			μSAB13AS17204	
17207			μSAB13AS17207	
17225			μSAB13AS17225	
17226				
17227				
17228				

●IBM PC/AT（英語版Windows）

デバイス	アセンブラ	Cコンパイラ	デバイス・ファイル	*SIMPLEHOST*
17201A	μSBB13RA17K	μSBB13CC17K	μSBB13AS17201	μSBB13ID17K
17203A			μSBB13AS17203	
17204			μSBB13AS17204	
17207			μSBB13AS17207	
17225			μSBB13AS17225	
17226				
17227				
17228				

A.3　PROMプログラマ

対象PROM	プログラムアダプタ	PROMプログラマ本体
μPD17P203AGC	AF-9808B	AF-9703
μPD17P204GC		AF-9704
μPD17P207GF	AF-9808A	AF-9705
μPD17P218CT	AF-9808J	AF-9706
μPD17P218GT	AF-9808H	

備考　17Kシリーズ用PROMプログラマおよびプログラムアダプタは，安藤電気株式会社の製品です。

〔メ　モ〕

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（RA17K）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。

注意　マスクROM発注用ヘキサ・ファイルの拡張子が.ICEでは発注できません。

（3）必要書類の作成

マスクROM発注にあたって，下記の書類を記入してください。

・マスク式ROM発注書

・マスク式ROM発注チェック・シート

（4）発　　注

（2）で作成した媒体と（3）で記入した書類とをまとめて，発注予約日までに当社特約店または当社販売部門に渡してください。

★ **備考**　詳しくはインフォメーション資料「ROMコードの発注方法」（資料番号 C10302J）をご覧ください。

〔メ　モ〕

付録C　命令索引

C.1　命令索引（機能別）

C.2　命令索引（アルファベット順）

★

付録D　改版履歴

これまでの改版箇所を次に示します。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	μPD17211, 17215, 17216, 17P218を追加	全般
第3版	μPD17217, 17218を追加 μPD17211を削除 μPD17P218開発中→開発済み	全般
	9.2.3　テーブル参照 テーブル参照命令の注意文を追加	第9章　データバッファ（DBF）
	第11章　割り込み機能に注意文を追加	第11章　割り込み機能
	14.3　プログラム・メモリ書き込み手順，14.4　プログラム・メモリ読み出し手順を修正	第14章　ワン・タイムPROMの書き込みとベリファイ
第4版	μPD17225, 17226, 17227, 17228を追加	全般
	μPD17202A, 17215, 17216, 17217, 17218, 17P202Aを削除	全般
	アセンブラを変更（AS17K→RA17K）	全般
	1.1　機能一覧に一部追加	第1章　概　説
	15.4　アセンブラ（RA17K）組み込みマクロ命令の凡例の表に拡張命令を追加	第15章　命令セット
	A.1　ハードウエアー覧を変更	付録A　開発ツール
	A.2　ソフトウエアー覧を変更	付録A　開発ツール

—— お問い合わせは，最寄りのNECへ ——

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01 東京都港区芝五丁目7番1号（NEC本社ビル）	東京	(03)3454-1111 (大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460 名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540 大阪市中央区城見一丁目4番24号（NEC関西ビル）	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社 札幌	(011)251-5599	太田支店 太田	(0276)46-4011	福井支店 福井	(0776)22-1866
東北支社 仙台	(022)267-8740	宇都宮支店 宇都宮	(028)621-2281	富山支店 富山	(0764)31-8461
岩手支店 盛岡	(019)651-4344	小山支店 小山	(0285)24-5011	三重支店 津	(0592)25-7341
郡山支店 郡山	(0249)23-5511	長野支社 松本	(0263)35-1662	京都支社 京都	(075)344-7824
いわき支店 いわき	(0246)21-5511	甲府支店 甲府	(0552)24-4141	神戸支社 神戸	(078)333-3854
長岡支店 長岡	(0258)36-2155	埼玉支社 大宮	(048)649-1415	中国支社 広島	(082)242-5504
土浦支店 土浦	(0298)23-6161	立川支社 立川	(0425)26-5981	鳥取支店 鳥取	(0857)27-5311
水戸支店 水戸	(029)226-1717	千葉支社 千葉	(043)238-8116	岡山支店 岡山	(086)225-4455
神奈川支社 横浜	(045)682-4524	静岡支社 静岡	(054)254-4794	松山支店 松山	(089)945-4149
群馬支店 高崎	(0273)26-1255	北陸支社 金沢	(076)232-7303	九州支社 福岡	(092)261-2806

【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 名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋	(052)222-2125
半導体販売技術本部 西日本販売技術部	〒540 大阪市中央区城見一丁目4番24号（NEC関西ビル）	大阪	(06) 945-3383

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

C97.8

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］ μPD172××サブシリーズ　ユーザーズ・マニュアル
（U12795JJ4V0UM00（第4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