お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

ユーザーズ・マニュアル

保守／廃止

V55PI™

16 ビット・マイクロプロセッサ

命令編

μPD70433

資料番号　U10231JJ4V1UMJ1（第４版）
発行年月　December 2000 N　CP(K)

保守／廃止

〔メ　モ〕

目次要約

保守／廃止

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。

保守／廃止

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **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財産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。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**
- **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。

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
p. 333	**5.6.3　MH符号化命令に注意を追加**
p. 335	**5.6.4　MR符号化命令に注意を追加**

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

はじめに

対象者　このマニュアルは，μPD70433（別名称V55PI）の機能を理解し，それを用いたアプリケーション・システムを設計するユーザのエンジニアを対象とします。

目的　このマニュアルは，μPD70433の持つ各種の命令機能をユーザに理解していただくことを目的とします。

構成　μPD70433のユーザーズ・マニュアルは，ハードウエア編と命令編（このマニュアル）に分かれています。

ハードウエア編	命令編
●概説	●概説
●端子機能	●命令の説明
●CPU機能	●V20，V30またはV25，V35に対する追加命令
●内蔵周辺ハードウエア	●キュー操作命令
●スタンバイ機能	●コーデック命令
●リセット機能	

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。なお，このマニュアルでは，μPD70433という製品名をV55PIの名称に統一して説明してあります。

μPD70108，70116（別名称V20™，V30™）をすでに経験しているユーザ

→V55PIはV20，V30のネイティブ・モードの命令と互換性があります。

第3章　V20，V30またはV25，V35に対する追加命令でソフトウエアの違いを確認し，それらの説明を中心にお読みください。

μPD70320，70330（別名称V25™，V35™）をすでに経験しているユーザ

→**第3章　V20，V30またはV25，V35に対する追加命令**でソフトウエアの違いを確認し，それらの説明を中心にお読みください。

ニモニックが分かっていて，命令機能の詳細を確認するとき

→**第2章　命令**をお読みください。

一通り命令機能の詳細を理解しようとするとき

→目次に従ってお読みください。

V55PIのハードウエア機能を理解しようとするとき

→別冊の**V55PI　ユーザーズ・マニュアル　ハードウエア編**を参照してください。

電気的特性を知りたいとき

→別冊の**V55PI　データ・シート**を参照してください。

各種機能の応用例を知りたいとき

→別冊の**アプリケーション・ノート**を参照してください。

凡　例

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

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

- ●パンフレット（U10244J）
- ●データ・シート（U11775J）
- ●ユーザーズ・マニュアル　ハードウエア編（U10514J）
- ●　〃　命令編（このマニュアル）
- ●アプリケーション・ノート　ソフトウエア編（U13215J）
- ●　〃　ファクシミリ編（U13256J）
- ●　〃　ハードウエア設計編（IEA-750）

保守／廃止

開発ツール関連資料一覧

	資　料　名		資料番号
ハードウエア	IE-70433-BX　ユーザーズ・マニュアル		U10608J
ソフトウエア	RA70116-I, SP70116-I		
	ユーザーズ・マニュアル	アセンブリ言語編	EEU-861
		操作編	EEU-869

保守／廃止

〔メ　モ〕

保守／廃止

目　　次

図　の　目　次（1/2）

図　の　目　次（2/2）

表　の　目　次

保守／廃止

〔メ　モ〕

保守／廃止

第1章　概　　説

16ビットVシリーズ™には，完全なソフトウエア互換性を持つ100種類の共通命令があり，ソフトウエア資産の有効活用ができます。

この共通命令に対してV55PIに追加された専用命令については，**第3章　V20，V30またはV25，V35に対する追加命令**を参照してください。

1.1　機能別命令一覧

V55PIの命令を機能別に大別すると次の31種類になります。

表1－1　機能別命令一覧（1/2）

命　令　群	ニモニック（アルファベット順）
データ転送命令	LDEA，MOV，TRANS，TRANSB，XCH，MOVSPA，MOVSPB
リピート・プリフィクス	REP，REPC，REPE，REPNC，REPNE，REPNZ，REPZ
プリミティブ・ブロック転送命令	CMPBK，CMPBKB，CMPBKW，CMPM，CMPMB，CMPMW，LDM，LDMB，LDMW，MOVBK，MOVBKB，MOVBKW，STM，STMB，STMW
ビット・フィールド操作命令	EXT，INS
入出力命令	IN，OUT
プリミティブ入出力命令	INM，OUTM
加減算命令	ADD，ADDC，SUB，SUBC
BCD演算命令	ADD4S，CMP4S，ROL4，ROR4，SUB4S
増減命令	DEC，INC
乗除算命令	DIV，DIVU，MUL，MULU
BCD補正命令	ADJ4A，ADJ4S，ADJBA，ADJBS
データ変換命令	CVTBD，CVTBW，CVTDB，CVTWL
比較命令	CMP
補数演算命令	NEG，NOT
論理演算命令	AND，OR，TEST，XOR
ビット操作命令	CLR1，NOT1，SET1，TEST1，BSCH
シフト命令	SHL，SHR，SHRA
ローテート命令	ROL，ROLC，ROR，RORC
サブルーチン制御命令	CALL，RET
スタック操作命令	DISPOSE，POP，PREPARE，PUSH
ブランチ命令	BR

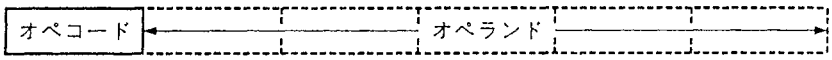
表 1－1　機能別命令一覧（2/2）

命　令　群	ニモニック（アルファベット順）
条件付きブランチ命令	BC, BCWZ, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BPE, BPO, BZ, BV, DBNZ, DBNZE, DBNZNE, BTCLR, BTCLRL
割り込み命令	BRK, BRKV, CHKIND, RETI, RETRBI, FINT
CPU制御命令	BUSLOCK, DI, EI, FPO1, FPO2, HALT, NOP, POLL, STOP
セグメント・オーバライド・プリフィクス	DS0：, DS1：, PS：, SS：
拡張セグメント・オーバライド・プリフィクス	DS2：, DS3：
レジスタ・ファイル空間アクセス用オーバライド・プリフィクス	IRAM：
レジスタ・バンク切り替え命令	BRKCS, TSKSW
ウォッチドッグ・タイマ操作命令	RSTWDT
キュー操作命令	QHOUT, QOUT, QTIN
コーデック命令	ALBIT, COLTRP, MHENC, MRENC, SCHEOL, GETBIT, MHDEC, MRDEC, CNVTRP

1.2　命令語の形式

命令語（オブジェクト・コード）は基本的に次の形式で表されます。

図 1－1　命令語形式



備考　オペコード：命令の種類を表す８ビットのコードです。

オペランド：命令の処理の対象となるレジスタ，メモリ・アドレスを示すフィールドです。

0－5バイトのフィールドで表されます。

1.3　命令の概要

1.3.1　データ転送

データ転送命令により，データの操作なしで，レジスタとレジスタ間またはレジスタとメモリ間のデータ転送ができます。次の５種の命令に分類できます。

汎用データの転送（MOV）	：第２オペランドから第１オペランドへのバイト／ワード転送をします。また，レジスタやメモリに直接数値を転送することもできます。
レジスタ・バンク内データの転送（MOVSPA，MOVSPB）	：レジスタ・バンク切り替えの前後でSS，SP間の内容を転送します。
実効アドレスの転送（LDEA）	：第２オペランドのオフセット・アドレス（実効アドレス）を第１オペランドへ転送します。
変換テーブルの転送（TRANS）	：変換テーブルの１バイトを転送します。
汎用データの交換（XCH）	：第１オペランドと第２オペランドの内容を交換します。

1.3.2　ブロック操作

リピート・プリフィクスとプリミティブ・ブロック転送命令によりバイトまたはワードのブロック（連続したデータ）に対する転送や比較ができます。

プリミティブ・ブロック転送命令には，アキュームレータとの間で行うブロック単位の転送に対する命令と同様，転送，ある値との比較，走査に対しての命令があります。また，１バイトのリピート・プリフィクスを前置するとハードウエアによる反復処理が可能で，連続したデータ操作ができます。

1.3.3 ビット・フィールド操作

ビット・フィールド操作命令により連続するメモリをビット・フィールドとみなして，指定した長さのデータを（指定した）ビット・フィールド領域とAWレジスタの間で転送することができます。

この命令は，命令実行終了後にワード・オフセット（IXレジスタまたはIYレジスタ）およびビット・オフセット（8ビット汎用レジスタ）を更新して，連続したビット・フィールド・データを自動的に指定します。コンピュータ・グラフィクスや高級言語に有効で，たとえば，Pascalのパックト・アレイやレコード・タイプのデータ構造に対応できます。

1.3.4 入 出 力

入出力命令，プリミティブ入出力命令によりI/Oデバイスに対してリード／ライトができます。

I/Oデバイスは，この命令によりデータ・バスを通じて，CPUとのデータ転送をします。

1.3.5 演　　算

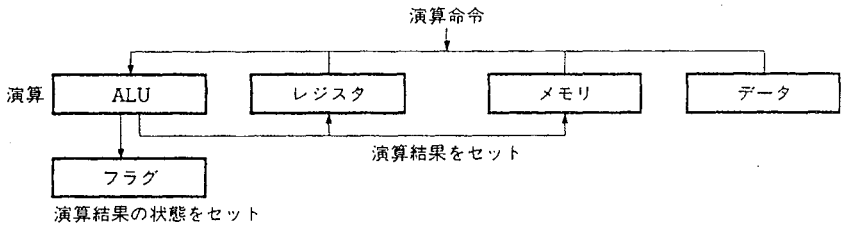
次の命令により8/16ビット・データの演算ができます。

加減算命令，増減命令，乗算命令，除算命令，比較命令，補数演算命令，論理演算命令

また，増減命令により汎用レジスタまたはメモリの8/16ビット・データをインクリメント（+1）／デクリメント（−1）することもできます。

各演算命令は対象となるレジスタやメモリの中で実行されるのではなく，実際は内部のALUで実行され，プログラム・ステータス・ワード（PSW）のフラグに，演算の結果がセット(1)，リセット(0)されます。

図1−2　演算命令でのALUの働き



1.3.6 BCD演算

BCD演算命令により16進数を用いて10進数を表現し，計算することができます。

この命令により，メモリ上のBCDストリングに対して算術演算，比較もできます。

また，BCDストリングに対してのローテートをサポートする命令も含まれています。

演算，比較命令は，使用するレジスタが決まっているため，パックトBCDストリングを指定するオペランドはありません。

ソース・ストリングの先頭アドレス（LSDを含むバイト・データのアドレス）は，データ・セグメント0（DS0）内でIXレジスタの内容で指定します。

デスティネーション・ストリングの先頭アドレス（LSDを含むバイト・データのアドレス）は，データ・セグメント1（DS1）内でIYレジスタの内容で指定します。

桁数は，CLレジスタの内容で指定します。

デスティネーション・ストリングとソース・ストリングは，同一の長さ（桁数）でなければならないので，長さが異なる場合は長い方に合わせて0を拡張します。

1.3.7 BCD補正

算術演算の実行前あるいは実行後にBCD補正命令を実行することで，BCD演算をサポートします。

BCD補正命令はALレジスタに対して行うので，オペランドはありません。加算，減算の場合，パックトBCDおよびアンパックトBCDのどちらでも補正できますが，乗除算の場合，アンパックトBCD表現の演算でしか補正できません。

1.3.8 データ変換

データ変換命令により2進数と10進数の型変換，語長の変換ができます。

CVTBD命令とCVTDB命令は，2進数と2桁のアンパックトBCDの型変換をする命令です。

CVTBW命令とCVTWL命令は，レジスタ内の符号拡張をする命令です。

1.3.9 ビット操作

ビット操作命令により汎用レジスタあるいはメモリのビット・データに対して，論理演算ができます。

命令形式のオペランドは，“reg，bit”，“mem，bit”，“reg”または“mem”となります。

第1オペランドのregまたはmemは，ビット・データを含んでいる8/16ビット・データを指定するもので，汎用レジスタまたは実効アドレスを記述します。

第2オペランドのbitは，ビット・データのバイト／ワード内アドレスを示すもので，CLの内容または8ビット・イミーディエト・データを使用します。

ただし，regまたはmemが8ビット・データの場合は，下位3ビットのみが有効なビット・アドレスとなり，16ビット・データの場合は，下位4ビットのみが有効なビット・アドレスで，上位ビットは無視されます。

1.3.10 シフト，ローテート

シフト命令，ローテート命令により汎用レジスタあるいはメモリの8/16ビット・データを1ビットまたは複数ビット（0-255），シフトまたはローテートできます。

シフトには，算術シフトと論理シフトがあります。シフトする桁数は通常では1ですが，命令のカウント・オペランドで指定すれば，CLレジスタの値によって実行するたびに変更できます（最大255）。算術シフトは，左に1ビットずらすときにLSBに0が入り，右に1ビットずらすとMSBに0が入ります。論理シフトの場合は，1ビットずらしてもLSBビットまたはMSBの値は変化しません。

ローテートもシフトと同様，ローテートする桁数はカウント・オペランドでCLレジスタに入れた値を指定します。この命令では，CYフラグとVフラグだけが変化します。CYフラグには，常にローテートして最後に押し出されたビットが入ります。Vフラグは，2桁以上のローテートの場合は必ず不定となり，1桁の場合はデスティネーションのMSB（延長）が変化するとセット(1)，変化しなければリセット(0)されます。CYフラグは，ROLC命令とRORC命令でデスティネーションの延長として使用できます。

1.3.11　スタック操作

スタック操作命令によりメモリ上のスタック操作ができます。

スタック操作命令には次の4種の命令があります。

PUSH　：スタックへデータを退避させる命令です。

POP　：スタックからデータを復帰させる命令です。

PREPARE：スタック・フレームを生成する命令で，ローカル変数の領域確保と，グローバル変数への参照を可能とするためにフレーム・ポインタをコピーします。

DISPOSE：スタック・ポインタ（SP）およびベース・ポインタ（BP）をPREPARE命令を実行する直前の状態に戻す命令です。

1.3.12　プログラム分岐

指定されたアドレスに分岐できます。

次の4種に大別できます。

サブルーチン制御命令：プログラム・カウンタ（PC）の内容をスタックに退避させる命令（CALL）と，スタックからPCの内容をリストアする命令（RET）があります。

ブランチ命令　：命令の流れを他のアドレスに移します。

条件付きブランチ命令：フラグの値によって命令の実行の流れを他のアドレスに移します。

割り込み命令　：外部デバイスからの割り込み要求や演算エラーが発生した際に，プログラムの実行を一時中止し，ソフトウエア割り込みによりプログラム実行の流れを制御します。

1.3.13　CPU制御

CPU制御命令により，フラグの操作や，外部デバイスとの同期，データ転送ができます。また，CPUに何もさせない命令（NOP）もあります。

1.3.14　レジスタ・バンク切り替え

レジスタ・バンクを切り替えることができます。

レジスタ・バンク切り替え命令には次の2種の命令があります。

BRKCS：高速なサブルーチン・コールを実現します。

TSKSW：レジスタ・バンクを使用したマルチタスク制御に使用します。

1.3.15 ウォッチドッグ・タイマ操作

ウォッチドッグ・タイマを操作するための命令です。

この命令を定期的に実行することにより，ウォッチドッグ・タイマのオーバフローを防ぎます。

プログラムの暴走などで，誤ってウォッチドッグ・タイマがクリアされないように特殊なオペランド・パターンを採用しています。

1.3.16 キュー操作

リアルタイムOS用の命令です。

リアルタイムOSにおけるマルチタスク制御のためのタスク管理用のキューを，高速に操作することができます。

この命令は，ある限定されたキュー構造に対してだけ有効です。

キュー操作命令には次の３種の命令があります。

QHOUT：キューの先頭ブロックを外します。

QOUT　：キューの任意のブロックを外します。

QTIN　：キューにブロックをキューイングします。

1.3.17 コーデック命令

２値画情報のMH符号化だけでなく，従来ACEE（アドバンスト・コンプレッション・エクスパンション・エンジン）などの専用デバイスが必要だったMR符号化を，小規模で高速なソフトウエア・コーデックにより実現できます。

コーデック命令には次の９種の命令があります。

ALBIT　：データを送信バッファへ出力します。

COLTRP：画素データの変化点情報を生成します。

MHENC ：MH符号化を行います。

MRENC ：MR符号化を行います。

SCHEOL：符号化処理においてEOLを検出します。

GETBIT ：復合化処理において１ビット（タグ）を検出します。

MHDEC ：MH復合化を行います。

MRDEC ：MR復合化を行います。

CNVTRP：画素データを生成します。

保守／廃止

〔メ　モ〕

第2章　命　　令

2.1　命令の説明（ニモニックのアルファベット順）

この節では各命令を次の項目に分けて説明します。

【命　令　形　式】
【オペレーション】
【オ　ペ　ラ　ン　ド】
【有効オーバライド・プリフィクス】
【フ　　ラ　　グ】
【説　　明】
【記　　述　　例】
【バ　イ　ト　数】
【命　令　語　形　式】

【命令形式】，【オペレーション】，【オペランド】，【有効オーバライド・プリフィクス】の各項目では説明上いくつもの識別子を用いています。表２－２から表２－４に識別子とその意味を，表２－５から表２－８にメモリ・アドレシング，汎用レジスタ／セグメント・レジスタ／拡張セグメント・レジスタの選択について示します。

【フラグ】の項目は命令実行により変化するフラグの動作を識別子により示します。各フラグの動作の凡例を表２－１に示します。

表２－１　フラグ動作の凡例

識別子	説　　明
空白	変化なし
0	リセット(0)される
1	セット(1)される
×	結果に従ってセット(1)またはリセット(0)される
U	不定
R	以前に退避した値がリストアされる

保守／廃止

表2-2 オペランド・タイプの凡例（1/2）

識別子	説明
reg	8／16ビット汎用レジスタ （8／16ビット汎用レジスタを2つ用いる命令における，デスティネーション側レジスタ）
reg'	8／16ビット汎用レジスタを2つ用いる命令における，ソース側レジスタ
reg8	8ビット汎用レジスタ （8ビット汎用レジスタを2つ用いる命令における，デスティネーション側レジスタ）
reg8'	8ビット汎用レジスタを2つ用いる命令における，ソース側レジスタ
reg16	16ビット汎用レジスタ （16ビット汎用レジスタを2つ用いる命令における，デスティネーション側レジスタ）
reg16'	16ビット汎用レジスタを2つ用いる命令における，ソース側レジスタ
mem	8／16ビット・メモリ・アドレス
mem8	8ビット・メモリ・アドレス
mem16	16ビット・メモリ・アドレス
mem32	32ビット・メモリ・アドレス
sfr	特殊機能レジスタ・ロケーション：FFF00H-FFFEFH
sfrl	特殊機能レジスタ・ロケーション：FFE00H-FFEFFH
dmem	16ビット・ダイレクト・メモリ・アドレス
imm	8／16ビット・イミーディエト・データ
imm3	3ビット・イミーディエト・データ
imm4	4ビット・イミーディエト・データ
imm8	8ビット・イミーディエト・データ
imm8'	8ビット・イミーディエト・データ（imm8の1の補数）
imm16	16ビット・イミーディエト・データ
acc	アキュームレータ（AWまたはAL）
sreg	セグメント・レジスタ
xsreg	拡張セグメント・レジスタ
src-table	256バイト変換テーブルの名称
src-block	IXレジスタでアドレスされるソース・ブロックの名称
dst-block	IYレジスタでアドレスされるデスティネーション・ブロックの名称
src-string	IXレジスタでアドレスされるソース・ストリングの名称
dst-string	IYレジスタでアドレスされるデスティネーション・ストリングの名称
near-proc	現在のプログラム・セグメント内のプロシージャの先頭アドレス
far-proc	別のプログラム・セグメント内のプロシージャの先頭アドレス
near-label	現在のプログラム・セグメント内の絶対アドレス
short-label	命令の終わりから－128～＋127バイトの範囲のメモリの相対アドレス
far-label	別のプログラム・セグメント内の絶対アドレス
regptr16	現在のプログラム・セグメント内のコール・アドレスのオフセットを持つ16ビット汎用レジスタ

保守／廃止

表2-2 オペランド・タイプの凡例（2/2）

識 別 子	説　　明
memptr16	現在のプログラム・セグメント内のコール・アドレスのオフセットを持つ16ビット・メモリ・アドレス
memptr32	別のプログラム・セグメント内のコール・アドレスのオフセットとセグメント・データを持つ32ビット・メモリ・アドレス
pop-value	スタックから捨てるバイト数（0-64 K，通常は偶数）
fp-op	外部の浮動小数点演算用コプロセッサの命令コードを判別するイミーディエト値
repeat	リピート・プリフィクス命令
IRAM：	レジスタ・ファイル空間アクセス用オーバライド・プリフィクス命令
R	レジスタ・セット（AW，BW，CW，DW，SP，BP，IX，IY）
〔　〕，（　）	省略可能
or，／	または

保守／廃止

表2-3 命令語の凡例

識別子	説明
W	バイト／ワード・フィールド(0：バイト，1：ワード)。ただしs＝1のときは，W＝1であってもサイン拡張のバイト・データを16ビット・オペランドとします。
reg	レジスタ・フィールド（000-111）
reg′	レジスタ・フィールド(000-111)(レジスタを2つ用いる命令における，ソース側レジスタ)
mod，mem	メモリ・アドレシング指定ビット（mod：00-10，mem：000-111）
(disp-low)	オプション16ビット・ディスプレースメント下位バイト
(disp-high)	〃　上位バイト
disp-low	PCレラティブ加算用16ビット・ディスプレースメント下位バイト
disp-high	〃　上位バイト
imm3	3ビット・イミーディエト・データ
imm4	4ビット　〃
imm8	8ビット　〃
imm8′	8ビット　〃　(imm8の1の補数)
imm16-low	16ビット・イミーディエト・データの下位バイト
imm16-high	〃　上位バイト
addr-low	16ビット・ダイレクト・アドレスの下位バイト
addr-high	〃　上位バイト
sreg	セグメント・レジスタ指定ビット（00-11）
xsreg	拡張セグメント・レジスタ指定ビット（10-11）
s	サイン拡張指定ビット（1：サイン拡張あり，0：サイン拡張なし）
offset-low	PCにロードされる16ビット・オフセット・データの下位バイト
offset-high	〃　上位バイト
seg-low	PSにロードされる16ビット・セグメント・データの下位バイト
seg-high	〃　上位バイト
pop-value-low	スタックの捨てるバイト数を指定する16ビット・データの下位バイト
pop-value-high	〃　上位バイト
disp8	PCにレラティブ加算される8ビット・ディスプレースメント
X	外部浮動小数点演算用コプロセッサのオペレーション・コード
XXX	〃
YYY	〃
ZZZ	〃

保守／廃止

表2-4 命令形式またはオペレーション説明上の凡例（1/2）

識別子	説明
dst	デスティネーション・オペランド
dst1	〃
dst2	〃
src	ソース・オペランド
src1	〃
src2	〃
target	ターゲット・オペランド
AW	アキュームレータ（16ビット）
AH	〃　（上位バイト）
AL	〃　（下位バイト）
BW	BWレジスタ（16ビット）
CW	CWレジスタ（　〃　）
CL	〃　（下位バイト）
DW	DWレジスタ（16ビット）
BP	ベース・ポインタ（16ビット）
SP	スタック・ポインタ（16ビット）
PC	プログラム・カウンタ（16ビット）
PSW	プログラム・ステータス・ワード（16ビット）
IX	インデクス・レジスタ（ソース）（16ビット）
IY	〃　（デスティネーション）（16ビット）
PS	プログラム・セグメント・レジスタ（16ビット）
SS	スタック・セグメント・レジスタ（16ビット）
DS0	データ・セグメント0レジスタ（16ビット）
DS1	データ・セグメント1レジスタ（16ビット）
DS2	拡張データ・セグメント2レジスタ（16ビット）
DS3	拡張データ・セグメント3レジスタ（16ビット）
AC	補助キャリー・フラグ
CY	キャリー・フラグ
P	パリティ・フラグ
S	サイン・フラグ
Z	ゼロ・フラグ
DIR	方向フラグ
IE	割り込み許可フラグ
V	オーバフロー・フラグ
$\overline{\text{IBRK}}$	I/Oブレーク・フラグ
BRK	ブレーク・フラグ
RB0	レジスタ・バンク0フラグ

保守／廃止

表2－4　命令形式またはオペレーション説明上の凡例（2/2）

識　別　子	説　　明
RB1	レジスタ・バンク1フラグ
RB2	レジスタ・バンク2フラグ
RB3	レジスタ・バンク3フラグ
VPC	ベクタPC
(…)	（ ）内で示されるメモリの内容
disp	ディスプレースメント（8/16ビット）
temp	テンポラリ・レジスタ（8/16/32ビット）
ext-disp8	8ビット・ディスプレースメントをサイン拡張した16ビット
seg	イミーディエト・セグメント・データ（16ビット）
offset	イミーディエト・オフセット・データ（16ビット）
←	転送方向
+	加　算
−	減　算
×	乗　算
÷	除　算
%	モジュロ
∧	論理積（AND）
∨	論理和（OR）
⊻	排他的論理和（XOR）
××H	16進数2けたの数値
××××H	16進数4けたの数値
／	兼用，または

表2-5 メモリ・アドレシング

mem \ mod	00	01	10
000	BW＋IX	BW＋IX＋disp8	BW＋IX＋disp16
001	BW＋IY	BW＋IY＋disp8	BW＋IY＋disp16
010	BP＋IX	BP＋IX＋disp8	BP＋IX＋disp16
011	BP＋IY	BP＋IY＋disp8	BP＋IY＋disp16
100	IX	IX＋disp8	IX＋disp16
101	IY	IY＋disp8	IY＋disp16
110	ダイレクト・アドレス	BP＋disp8	BP＋disp16
111	BW	BW＋disp8	BW＋disp16

注意　プリミティブ命令以外のメモリ・アドレシングでBPを使用する場合，ディフォールトのセグメント・レジスタはSSになります。また，BPを使用しない場合，ディフォールトのセグメント・レジスタはDS0になります。

プリミティブ命令のメモリ・アドレシングにおいて，デスティネーション・ブロックのディフォールトのセグメント・レジスタはDS1になります。また，メモリ・アドレシングにおいてソース・ブロックのディフォールトのセグメント・レジスタはDS0になります。

表2-6 8/16ビット汎用レジスタの選択

reg, reg'	W＝0	W＝1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

表2-7 セグメント・レジスタの選択

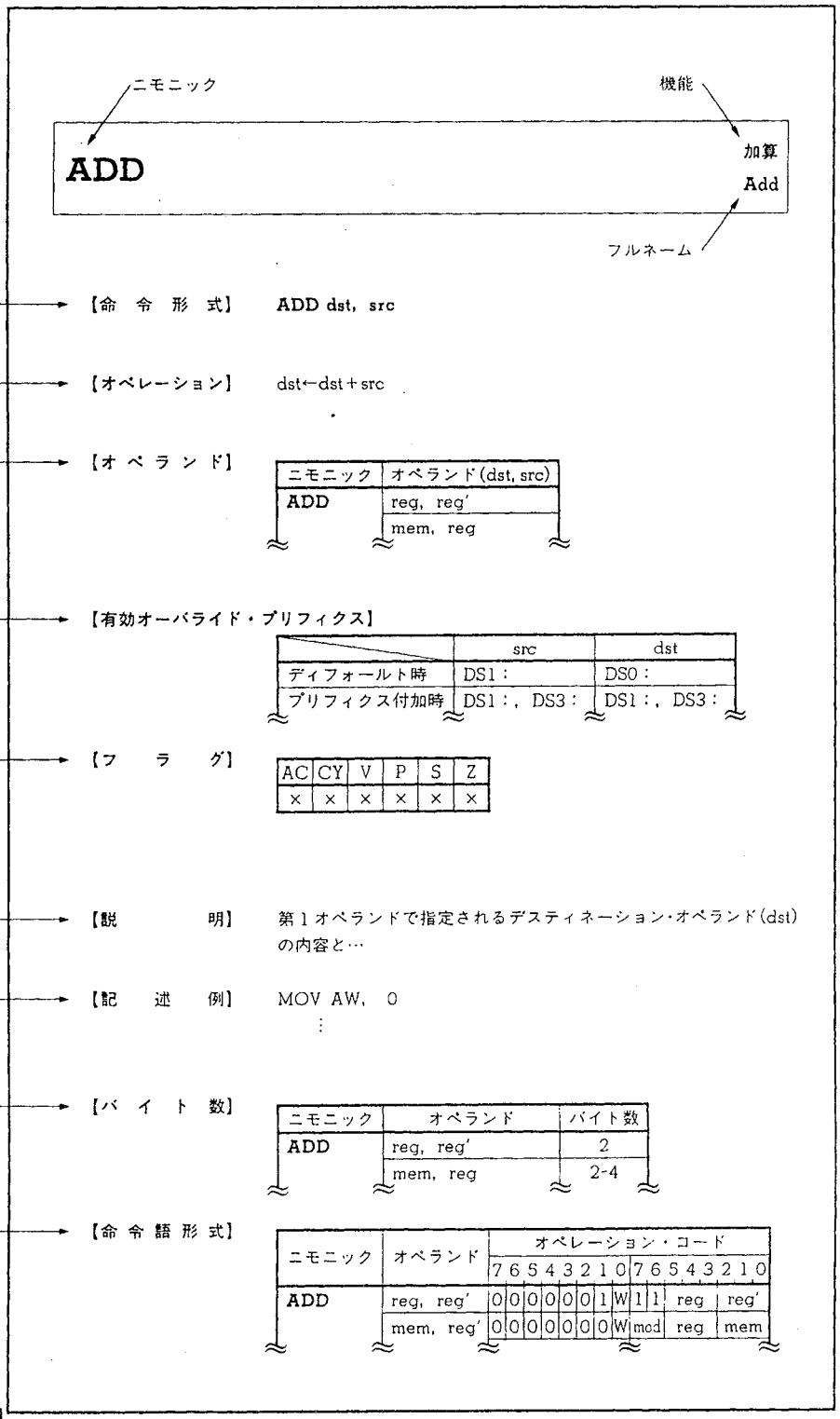
sreg	
00	DS1
01	PS
10	SS
11	DS0

表2-8 拡張セグメント・レジスタの選択

xsreg	
10	DS3/VPC
11	DS2

保守／廃止

図2-1 記述例



オペレーション・コード	
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
1バイト目	2バイト目
3バイト目	4バイト目
5バイト目	6バイト目

保守／廃止

ADD

加算
Add

【命　令　形　式】　**ADD dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
ADD	reg, reg'	dst←dst＋src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W＝0のとき］AL←AL＋imm8 ［W＝1のとき］AW←AW＋imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：，DS1：，PS：，SS：， DS2：，DS3：，IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）の内容と第２オペランドで指定されるソース・オペランド(src)の内容を加算し，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　メモリ０：50Hの内容（ワード・データ）とDWレジスタの内容を加算して0：50Hへストアする。

```
MOV  AW, 0
MOV  DS1, AW
MOV  IY, 50H
ADD  DS1 : WORD  PTR [IY], DW
```

保守／廃止

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
ADD	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADD	reg, reg'	0	0	0	0	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	0	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	0	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	0	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

ADD4S

10進加算
Add Nibble String

【命 令 形 式】 **ADD4S dst-string, src-string**
ADD4S

【オペレーション】 BCDストリング(IY, CL)←BCDストリング(IY, CL)+BCDストリング(IX, CL)

【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
ADD4S	dst-string, src-string
	な し

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	DS1：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：	DS1：, DS3：, IRAM：

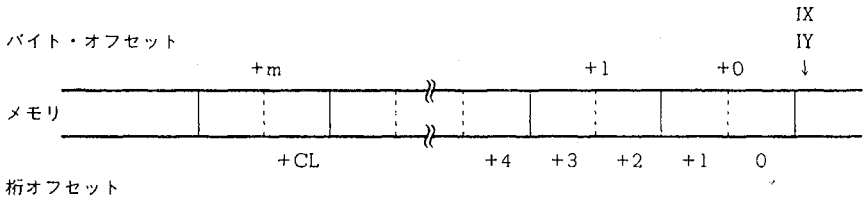
【フ ラ グ】

AC	CY	V	P	S	Z
U	×	U	U	U	×

【説　　明】 IXレジスタでアドレスされるパックトBCDストリングとIYレジスタでアドレスされるパックトBCDストリングを加算し，結果をIYレジスタでアドレスされるストリングにストアします。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され1-254桁まで可能です。

デスティネーション・ストリングのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指されるセグメント内にもロケートできます。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。

パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋1の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示します。

したがって，桁数が奇数のときのBCD加算命令は，必ず最上位バイトの上位4ビットを“0”にしてから実行してください。この結果，キャリーは最上位バイトのビット4に示され，フラグには反映されません。

【記　述　例】

```
MOV    IX, OFFSET  VAR_1
MOV    IY, OFFSET  VAR_2
MOV    CL, 4
ADD4S
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADD4S	dst-string, src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0
	な　し																

保守／廃止

ADDC

キャリーを含む加算

Add with Carry

【命　令　形　式】　**ADDC dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
ADDC	reg, reg'	dst←dst＋src＋CY
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき]AL←AL＋imm8＋CY [W＝1のとき]AW←AW＋imm16＋CY

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：，DS1：，PS：，SS：， DS2：，DS3：，IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）の内容と第２オペランドで指定されるソース・オペランド（src）の内容をCYフラグの内容も含めて加算し，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】

```
SET1   CY        ；CYフラグをセット（1）する。
XOR    AW, AW    ；AW＝0
MOV    BW, 0FFH  ；BW＝0FFH
ADDC   AW, BW    ；AWレジスタの内容＝100Hとなる。
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ADDC	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADDC	reg, reg'	0	0	0	1	0	0	1	W	1	1	reg				reg'	
	mem, reg	0	0	0	1	0	0	0	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	1	0	0	1	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	1	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	1	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

ADJ4A

加算結果のパックト10進補正

Adjust Nibble Add

【命 令 形 式】　**ADJ4A**

【オペレーション】　AL∧0FH＞9またはAC＝1のとき

AL←AL＋6

AC←1

AL＞9FHまたはCY＝1のとき

AL←AL＋60H

CY←1

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJ4A	な し

【フ ラ グ】

AC	CY	V	P	S	Z
×	×	U	×	×	×

【説　　明】　2つのパックト10進数の加算のALレジスタ内の結果を1つのパックト10進数に補正します。

【記 述 例】　ADJ4A

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJ4A	な し	0	0	1	0	0	1	1	1

保守／廃止

ADJ4S

減算結果のパックト10進補正

Adjust Nibble Subtract

【命 令 形 式】　**ADJ4S**

【オペレーション】　AL∧0FH>9またはAC=1のとき

AL←AL−6

AC←1

AL>9FHまたはCY=1のとき

AL←AL−60H

CY←1

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJ4S	な し

【フ ラ グ】

AC	CY	V	P	S	Z
×	×	U	×	×	×

【説　　明】　2つのパックト10進数の減算のALレジスタ内の結果を1つのパックト10進数に補正します。

【記 述 例】

```
SUB  AW, BW
ADJ4S
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJ4S	な し	0	0	1	0	1	1	1	1

保守／廃止

ADJBA

加算結果のアンパック10進補正

Adjust Byte Add

【命 令 形 式】　**ADJBA**

【オペレーション】　AL∧0FH>9またはAC=1のとき

AL←AL+6

AH←AH+1

AC←1

CY←AC

AL←AL∧0FH

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJBA	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	U	U	U	U

【説　　明】　２つのアンパック10進数の加算のALレジスタ内の結果を１つのアンパック10進数に補正します。上位４ビットは０になります。

【記　述　例】　ADJBA

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJBA	な　し	0	0	1	1	0	1	1	1

保守／廃止

ADJBS

減算結果のアンパック10進補正

Adjust Byte Subtract

【命 令 形 式】　**ADJBS**

【オペレーション】　AL∧0FH＞9またはAC＝1のとき

AL←AL－6

AH←AH－1

AC←1

CY←AC

AL←AL∧0FH

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJBS	な し

【フ ラ グ】

AC	CY	V	P	S	Z
×	×	U	U	U	U

【説　　明】　2つのアンパック10進数の減算のALレジスタ内の結果を1つのアンパック10進数に補正します。上位4ビットは0になります。

【記 述 例】　SUB, AW, BW

ADJBS

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJBS	な し	0	0	1	1	1	1	1	1

保守／廃止

ALBIT

送信バッファへのデータ出力

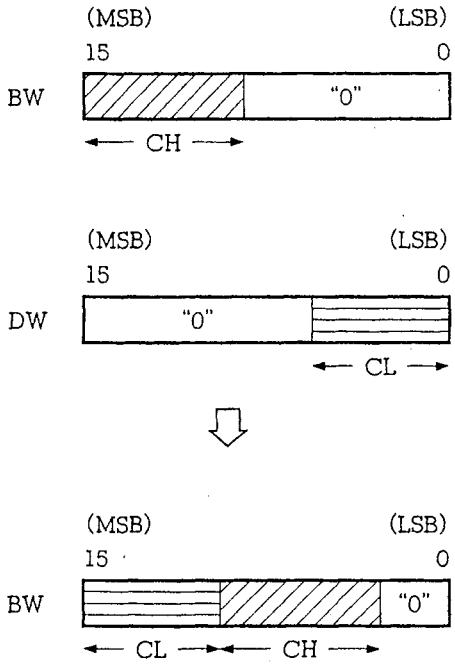
Add Last Bit

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**ALBIT**

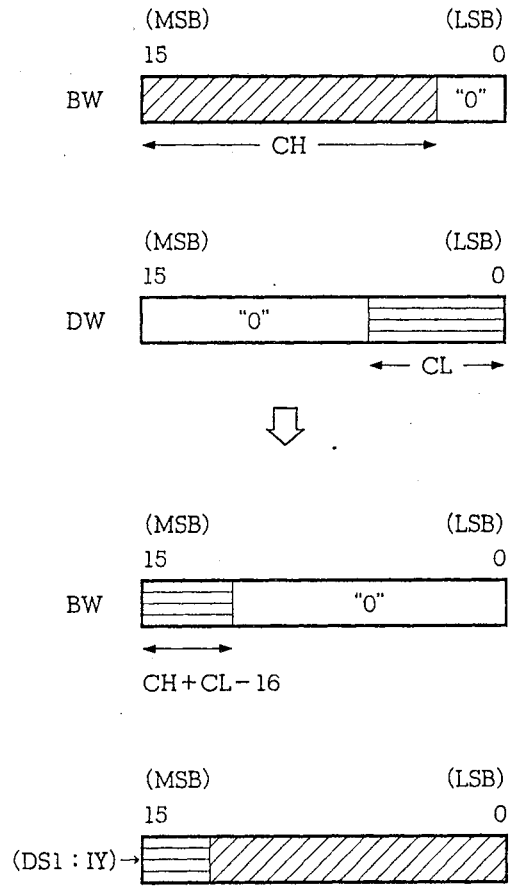
【オペレーション】

［CH＋CL＜16のとき］



CH←CL＋CH

[CH+CL≧16のとき]



(IY, IY+1) ←temp

IY ←IY+2

BP ←BP−2

CH ←CH+CL−16

【オ ペ ラ ン ド】

ニモニック	オペランド
ALBIT	な し

【フ　ラ　グ】

Z	CY	送信バッファ残り	データ残り
0	U	あり	—
1	0	なし	なし
1	1	なし	あり

保守／廃止

【説　　明】　符号化処理においてEOL，FILLなど16ビット以下のデータを送信バッファへ出力するための命令です。データの先頭はLSBに入ります。

指定された送出したいデータが，処理中の16ビットに満たない半端符号と合わせて16ビット以上にまとまったところで指定された送信バッファへ出力します。16ビットを越える分，あるいは合わせて16ビットに満たないデータはパラメータとしてBW（処理中半端符号データ）に，そのデータのビット数はパラメータとしてCH（処理中半端符号ビット数）に保持します。

割り込み要求のサンプリングを行いません。

＜入力パラメータ＞

BW ：処理中半端符号データ
CH ：処理中半端符号ビット数
DW：送出符号データ
CL ：送出符号データ・ビット数
BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）
DS1：送信バッファ・セグメント
IY ：送信バッファ・オフセット

＜出力パラメータ＞

BW ：処理中半端符号データ
CH ：処理中半端符号ビット数
BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）
DS1：送信バッファ・セグメント
IY ：送信バッファ・オフセット

【記　述　例】　ALBIT

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ALBIT	な　し	0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0

保守／廃止

AND

論理積
And

【命　令　形　式】　**AND dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
AND	reg, reg'	dst←dst∧src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL←AL∧imm8 [W＝1のとき] AW←AW∧imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	0	0	×	×	×

【説　　明】　第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）の論理積（AND）をとり，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】
```
MOV  DW, IY
AND  DW, 7FFFH
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
AND	reg, reg′	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
AND	reg, reg′	0	0	1	0	0	0	1	W	1	1	reg				reg′	
	mem, reg	0	0	1	0	0	0	0	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	0	0	0	1	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	1	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	0	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注　アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	W	1	1	1	0	0	reg		
imm8								—							

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

BC
BL

CY＝1による条件分岐
Branch if Carry
Branch if Lower

【命　令　形　式】　**BC　short-label**
BL　short-label

【オペレーション】　CY＝1のとき：PC←PC＋ext-disp8

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
BC	short-label
BL	

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CYフラグが1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        TEST  AL, BL
        BC    SHORT  LP4   ; LP4＝レーベル
        ⋮
        TEST  AL, BL
        BL    SHORT  LP5   ; LP5＝レーベル
        ⋮
LP4 :
```

【バ　イ　ト　数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BC	short-label	0	1	1	1	0	0	1	0	disp8							
BL																	

保守／廃止

BCWZ

CW＝0による条件分岐
Branch if CW equals Zero

【命 令 形 式】　**BCWZ short-label**

【オペレーション】　CW＝0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BCWZ	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　CWレジスタの値が０ならば，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記 述 例】
```
LP22：
      ⋮
      ADD  AL, BL
      BCWZ  SHORT  LP22  ；LP22＝レーベル
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BCWZ	short-label	1	1	1	0	0	0	1	1	disp8							

保守／廃止

BE
BZ

Z＝1による条件分岐
Branch if Equal
Branch if Zero

【命 令 形 式】　**BE short-label**
BZ short-label

【オペレーション】　Z＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BE	short-label
BZ	

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　Zフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記 述 例】

```
        AND  AL, 2
        BE   SHORT  LOOP   ;LOOP＝レーベル
        :
        OR   AH, BH
        BZ   SHORT  LOOP1  ;LOOP1＝レーベル
        :
LOOP:
```

【バ イ ト 数】　2

保守／廃止

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BE	short-label	0	1	1	1	0	1	0	0	disp8							
BZ																	

保守／廃止

BGE

S⊻V=0による条件分岐
Branch if Greater Than or Equal

【命 令 形 式】　**BGE short-label**

【オペレーション】　S⊻V=0のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BGE	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　SフラグとVフラグの排他的論理和(XOR)が0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
      SHL  AL, 1
      BGE  SHORT  LP16   ;LP16=レーベル
      :
LP16:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BGE	short-label	0	1	1	1	1	1	0	1	disp8							

保守／廃止

BGT

(S∀V)∨Z=0による条件分岐
Branch if Greater Than

【命 令 形 式】　**BGT short-label**

【オペレーション】　(S∀V)∨Z=0のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BGT	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　SフラグとVフラグの排他的論理和（XOR）をとった結果とZフラグの論理和（OR）が0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP18:
      ⋮
      SHL AL, 1
      BGT LP18
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BGT	short-label	0	1	1	1	1	1	1	1	disp8							

保守／廃止

BH

CY∨Z=0による条件分岐
Branch if Higher

【命 令 形 式】 **BH short-label**

【オペレーション】 CY∨Z=0のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BH	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 CYフラグとZフラグの論理和（OR）が0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。

この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記 述 例】

```
        ROL AL, 1
        BH SHORT LP10 ;LP10=レーベル
        :
LP10:
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BH	short-label	0	1	1	1	0	1	1	1	disp8							

保守／廃止

BLE

(S∀V)∨Z=1による条件分岐
Branch if Less than or Equal

【命 令 形 式】 **BLE short-label**

【オペレーション】 (S∀V)∨Z=1のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BLE	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 Sフラグ と Vフラグの排他的論理和（XOR）をとった結果とZフラグの論理和（OR）が1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記 述 例】
```
LP17:
        :
        SHR AL, 1
        BLE SHORT LP17
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BLE	short-label	0	1	1	1	1	1	1	0	disp8							

保守／廃止

BLT

S∀V＝1による条件分岐
Branch if Less Than

【命 令 形 式】 **BLT short-label**

【オペレーション】 S∀V＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BLT	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 SフラグとVフラグの排他的論理和(XOR)が1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        ADD AL, BL
        BLT SHORT LP15  ;LP15＝レーベル
        ⋮
LP15:
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BLT	short-label	0	1	1	1	1	1	0	0	disp8							

保守／廃止

BN

S＝1による条件分岐
Branch if Negative

【命 令 形 式】　**BN short-label**

【オペレーション】　S＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BN	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　Sフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
      ADD AL, BL
      BN LP11  ;LP11＝レーベル
      ⋮
LP11:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BN	short-label	0	1	1	1	1	0	0	0	disp8							

保守／廃止

BNC
BNL

CY＝0による条件分岐
Branch if Not Carry
Branch if Not Lower

【命 令 形 式】　**BNC short-label**
BNL short-label

【オペレーション】　CY＝0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNC	short-label
BNL	

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　CYフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記 述 例】

```
        ROR  AL, 1
        BNC  SHORT LP6   ;LP6＝レーベル
        ⋮
        ROR  AL, 1
        BNL  SHORT LP7   ;LP7＝レーベル
        ⋮
LP6:
```

【バ イ ト 数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNC	short-label	0	1	1	1	0	0	1	1	disp8							
BNL																	

保守／廃止

BNE
BNZ

Z=0による条件分岐
Branch if Not Equal
Branch if Not Zero

【命 令 形 式】 **BNE short-label**
BNZ short-label

【オペレーション】 Z=0のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNE	short-label
BNZ	

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 Zフラグが0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記 述 例】
```
      OR   AL, BL
      BNE  SHORT LP8   ;LP8=レーベル
      :
      AND  SH, BH
      BNZ  SHORT LP9   ;LP9=レーベル
      :
LP8:
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNE	short-label	0	1	1	1	0	1	0	1	disp8							
BNZ																	

保守／廃止

BNH

CY∨Z＝1による条件分岐
Branch if Not Higher

【命 令 形 式】　**BNH short-label**

【オペレーション】　CY∨Z＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNH	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CYフラグとZフラグの論理和（OR）が1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。

この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記　述　例】

```
        ROR AL, 1
        BNH SHORT LP9  ;LP9＝レーベル
        ⋮
LP9:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNH	short-label	0	1	1	1	0	1	1	0	disp8							

保守／廃止

BNV

V=0による条件分岐
Branch if Not Overflow

【命 令 形 式】　**BNV short-label**

【オペレーション】　V=0のとき：PC←PC+ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNV	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　Vフラグが0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜+127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
    ROR AL, 1
    BNV LP3
    :
LP3:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNV	short-label	0	1	1	1	0	0	0	1	disp8							

保守／廃止

BP

S＝0による条件分岐
Branch if Positive

【命 令 形 式】 **BP short-label**

【オペレーション】 S＝0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BP	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 Sフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        SHR AL, 1
        BP SHORT LP12  ;LP12＝レーベル
        :
LP12:
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BP	short-label	0	1	1	1	1	0	0	1	disp8							

保守／廃止

BPE

P＝1による条件分岐
Branch if Parity Even

【命 令 形 式】　**BPE short-label**

【オペレーション】　P＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BPE	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　Pフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        ADD AL, BL
        BPE SHORT LP13  ;LP13＝レーベル
        ⋮
LP13:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BPE	short-label	0	1	1	1	1	0	1	0	disp8							

保守／廃止

BPO

P＝0による条件分岐
Branch if Parity Odd

【命 令 形 式】　**BPO short-label**

【オペレーション】　P＝0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BPO	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　Pフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        ADD AL, BL
        BPO SHORT LP14  ;LP14＝レーベル
        :
LP14:
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BPO	short-label	0	1	1	1	1	0	1	1	disp8							

保守／廃止

BR

無条件分岐
Branch

【命 令 形 式】　**BR target**

【オペランド，オペレーション】

ニモニック	オペランド（target）	オペレーション
BR	near-label	PC←PC＋disp
	short-label	PC←PC＋ext-disp8
	regptr16	PC←target
	memptr16	
	far-label	PS←seg PC←offset
	memptr32	PS←（memptr32＋3，memptr32＋2） PC←（memptr32＋1，memptr32）

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　○target＝near-labelのとき

現在のPCの値に16ビット・ディスプレースメント（disp）を加えた値をPCに転送します。

ブランチ・アドレスがこの命令の置かれているセグメント内にあれば，アセンブラが自動的にこの命令を実行します。

○target＝short-labelのとき

現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット（ext-disp8））を加えた値をPCに転送します。

ブランチ・アドレスがこの命令の置かれているセグメント内でかつ±127バイト内にあれば，アセンブラが自動的にこの命令を実行します。

○target＝regptr16またはtarget＝memptr16のとき

ターゲット・オペランド（target）の内容をPCに転送します。

この命令の置かれているセグメント内の任意のアドレスにブランチできます。

○target＝far-labelのとき

PCに命令の2，3バイト目の16ビット・オフセット・データを，PSに命令の4，5バイト目の16ビット・セグメント・データを転送します。

基本メモリ空間の任意のアドレスにブランチできます。

○target＝memptr32のとき

32ビット・メモリの上位2バイトをPSに，下位2バイトをPCにロードします。

任意のセグメントの任意のアドレスにブランチできます。

【記　述　例】　BR $－8

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
BR	near-label	3
	short-label	2
	regptr16	2
	memptr16	2-4
	far-label	5
	memptr32	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BR	near-label	1	1	1	0	1	0	0	1	disp-low							
		disp-high								—							
	short-label	1	1	1	0	1	0	1	1	disp8							
	regptr16	1	1	1	1	1	1	1	1	1	1	1	0	0	reg		
	memptr16	1	1	1	1	1	1	1	1	mod	1	0	0	mem			
		(disp-low)								(disp-high)							
	far-label	1	1	1	0	1	0	1	0	offset-low							
		offset-high								seg-low							
		seg-high								—							
	memptr32	1	1	1	1	1	1	1	1	mod	1	0	1	mem			
		(disp-low)								(disp-high)							

保守／廃止

BRK

ソフトウエア・トラップ

Break

【命 令 形 式】　**BRK target**

【オペランド，オペレーション】

ニモニック	オペランド（target）	オペレーション
BRK	3	(SP－1，SP－2) ←PSW (SP－3，SP－4) ←PS (SP－5，SP－6) ←PC SP←SP－6 IE←0，BRK←0 PC← (00DH，00CH) PS← (00FH，00EH)
	imm8（≠3）	(SP－1，SP－2) ←PSW (SP－3，SP－4) ←PS (SP－5，SP－6) ←PC SP←SP－6 IE←0，BRK←0 PC← (imm8×4＋1，imm8×4) PS← (imm8×4＋3，imm8×4＋2)

【フ ラ グ】

AC	CY	V	P	S	Z	IE	BRK
						0	0

【説　　明】　PSW，PS，PCの値をスタックに退避し，IEフラグとBRKフラグをリセット(0)します。

続いて，target＝3の場合は割り込みベクタ・テーブルのベクタ３の下位２バイトをPCに，上位２バイトをPSにロードします。

target＝imm8の場合は８ビット・イミーディエト・データで指定される割り込みベクタ・テーブル（４ビット）の下位２バイトをPCに，上位２バイトをPSにロードします。

保守／廃止

【記　述　例】

- BRK 3
- BRK 5

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
BRK	3	1
	imm8	2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BRK	3	1	1	0	0	1	1	0	0	—							
	imm8	1	1	0	0	1	1	0	1	imm8							

保守／廃止

BRKCS

レジスタ・バンク切り替え

Break Context Switch

（V20，V30に対する追加命令，V25，V35に対する拡張命令）

【命 令 形 式】　**BRKCS reg16**

【オペレーション】

temp←PSW

RB3-RB0←reg16の下位４ビット

IE←0，BRK←0

新たに選択されたレジスタ・バンク上のPSW退避領域←temp

新たに選択されたレジスタ・バンク上のPC退避領域←PC

PC←新たに選択されたレジスタ・バンク上のベクタPC

【オ ペ ラ ン ド】

ニモニック	オペランド
BRKCS	reg16

【フ ラ グ】

AC	CY	V	P	S	Z	RB0	RB1	RB2	RB3	DIR	IE	BRK	$\overline{\text{IBRK}}$
						×	×	×	×		0	0	

【説　　明】

レジスタ・バンクを，オペランドに記述した16ビット・レジスタの内容の下位４ビットで示されるレジスタ・バンクに切り替えます。また，新しいレジスタ・バンク内にあらかじめストアしておいたPSとベクタPCから得られるアドレスに分岐します。

高速なサブルーチン・コールなどに使用します。

新しいレジスタ・バンクからの復帰にRETRBI命令を使用します。

【記 述 例】　BRKCS AW

【バ イ ト 数】　3

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BRKCS	なし	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1
		1	1	0	0	0	reg			—							

保守／廃止

BRKV

オーバフロー例外
Break if Overflow

【命 令 形 式】 **BRKV**

【オペレーション】 V＝1のとき (SP－1, SP－2) ←PSW
(SP－3, SP－4) ←PS
(SP－5, SP－6) ←PC
SP←SP－6
IE←0, BRK←0
PC← (011H, 010H)
PS← (013H, 012H)

【オ ペ ラ ン ド】

ニモニック	オペランド
BRKV	なし

【フ ラ グ】

AC	CY	V	P	S	Z	IE	BRK
						0	0

【説　　明】 Vフラグがセット(1)されていれば，PSW，PS，PCの値をスタックに退避し，IEフラグとBRKフラグをリセット(0)します。
次に，割り込みベクタ・テーブルのベクタ４の下位２バイトをPCに，上位２バイトをPSにロードします。
Vフラグがリセット(0)されていれば次の命令に進みます。

【記 述 例】 BRKV

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
BRKV	なし	1	1	0	0	1	1	1	0

保守／廃止

BSCH

ビット操作
Bit Search

（V20，V30に対する追加命令）

【命 令 形 式】　**BSCH dst**

【オペランド，オペレーション】

ニモニック	オペランド	オペレーション
BSCH	reg8	① CL←0 ② [dstのビットNo.CL=0のとき] ●CL<7のとき CL←CL+1，②から再実行 ●CL=7のとき Z←1 ③ [dstのビットNo.CL=1のとき] Z←0
	mem8	
	reg16	① CL←0 ② [dstのビットNo.CL=0のとき] ●CL<15のとき CL←CL+1，②から再実行 ●CL=15のとき Z←1 ③ [dstのビットNo.CL=1のとき] Z←0
	mem16	

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	U	U	×

【説　　明】　デスティネーション・オペランド（dst）で指定されるビット0から順にビット7またはビット15まで“1”をサーチし，最初にサーチしたビット番号をCLに返します。サーチした結果“1”がない場合（dst=0），Zフラグ（PSWのビット6）をセットします。

サーチした結果“1”があった場合，Zフラグをリセットします。

【記 述 例】　●BSCH BYTE

●BSCH WORD

保守／廃止

【バ イ ト 数】　5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BSCH	reg	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	W
		1	1	0	0	0	reg			—							
	mem	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	W
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							

保守／廃止

BTCLR

特殊機能レジスタの条件付き分岐

Branch if True and Clear

（V20，V30に対する追加命令）

【命 令 形 式】　**BTCLR sfr, imm3, short-label**

【オペレーション】　(sfr) のビットNo. imm3＝1のとき：(sfr) のビットNo. imm3←0

PC←PC＋ext-disp8

(sfr) のビットNo. imm3＝0のとき：PC←PC＋5

【オ ペ ラ ン ド】

ニモニック	オペランド
BTCLR	sfr, imm3, short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　特殊機能レジスタ領域の上位240バイト（FFF00H-FFFEFH）内の指定された特殊機能レジスタのimm3で示されるビットが1のとき，そのビットをクリアして現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加え，PCにロードします。

この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記　述　例】　BTCLR UARTS1, 6, 45

【バ イ ト 数】　5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BTCLR	sfr, imm3, short-label	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0
		sfr								imm3							
		disp8								—							

保守／廃止

BTCLRL

特殊機能レジスタの条件付き分岐

Branch if True and Clear Low

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**BTCLRL sfr，imm3，short-label**

【オペレーション】　(sfr) のビットNo.imm3＝1のとき：(sfr) のビットNo.imm3←0

PC←PC＋ext-disp8

(sfr) のビットNo.imm3＝0のとき：PC←PC＋5

【オ ペ ラ ン ド】

ニモニック	オペランド
BTCLRL	sfr，imm3，short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　特殊機能レジスタ領域の下位256バイト（FFE00H-FFEFFH）内の指定された特殊機能レジスタのimm3で示されるビットが1のとき，そのビットをクリアして現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加え，PCにロードします。

この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

この命令はBTCLR命令と対になっており，BTCLR命令が特殊機能レジスタ空間の上位240バイト（FFF00H-FFFEFH）を対象とするのに対し，BTCLRL命令は同空間の下位256バイト（FFE00H-FFEFFH）を対象とします。これ以外の機能は同じです。

【記　述　例】　BTCLRL IC09，7，45

【バ イ ト 数】　5

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BTCLRL	sfr, imm3,	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	1
	short-label	sfr								imm3							
		disp8								—							

保守／廃止

BUSLOCK

バス・ロック・プリフィクス
Bus Lock Prefix

【命 令 形 式】 **BUSLOCK**

【オペレーション】 Bus Lock Prefix

【オ ペ ラ ン ド】

ニモニック	オペランド
BUSLOCK	なし

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 この命令に続く1命令を実行している間，バス・ロック信号（$\overline{\text{BUSLOCK}}$）を出力し，ホールド要求を禁止します。
リピート・プリフィクスの付いたブロック処理命令に対してこの命令が用いられれば，ブロック処理が終了するまで$\overline{\text{BUSLOCK}}$信号が出力され続けます。

注意1．この命令はPOLL命令の直前に置かないでください。
2．この命令と次の命令の間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記 述 例】 BUSLOCK REP MOVBKB

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
BUSLOCK	なし	1	1	1	1	0	0	0	0

保守／廃止

BV

V＝1による条件分岐
Branch if Overflow

【命 令 形 式】　**BV short-label**

【オペレーション】　V＝1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BV	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　Vフラグが1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP2:
      ⋮
      SHL AL, 1
      BV SHORT LP2
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BV	short-label	0	1	1	1	0	0	0	0	disp8							

保守／廃止

CALL

サブルーチン・コール
Call

【命 令 形 式】　**CALL target**

【オペランド，オペレーション】

ニモニック	オペランド (target)	オペレーション
CALL	near-proc	SP←SP－2 (SP＋1, SP) ←PC PC←PC＋disp
	regptr16	SP←SP－2 (SP＋1, SP) ←PC PC←regptr16
	memptr16	(SP－1, SP－2) ←PC SP←SP－2 PC← (memptr16)
	far-proc	SP←SP－2 (SP＋1, SP) ←PS PS←seg SP←SP－2 (SP＋1, SP) ←PC PS←offset
	memptr32	(SP－1, SP－2) ←PS (SP－3, SP－4) ←PC SP←SP－4 PS← (memptr32＋3, memptr32＋2) PC← (memptr32＋1, memptr32)

【フ ラ グ】

AC	CY	V	P	S	Z

保守／廃止

【説　　明】　○target＝near-procまたはtarget＝regptr16のとき

PCの値がスタックに退避されたあと，ターゲット・オペランド（target）の次の内容がPCに転送されます。

target＝near-procのとき：16ビット相対アドレス

target＝regptr16のとき　：16ビット・レジスタの値（オフセット）

○target＝memptr16のとき

PCの値がスタックに退避されたあと，ターゲット・オペランド（target）でアドレスされる16ビット・メモリの内容（オフセット）がPCに転送されます。

この命令の置かれているセグメント内の任意のアドレスを呼ぶことが可能です。

○target＝far-procのとき

PCとPSの値がスタックに退避されたあと，PCには命令の2，3バイト目が，PSには命令の4，5バイト目が転送されます。

この命令によって基本メモリ空間の任意のアドレスを呼ぶことが可能です。

○target＝memptr32のとき

PCとPSの値がスタックに退避されたあと，ターゲット・オペランド（target）でアドレスされる32ビット・メモリの上位2バイトがPSに，下位2バイトがPCに転送されます。

この命令によって基本メモリ空間の任意のアドレスを呼ぶことが可能です。

【記　述　例】

- CALL $+10
- CALL SUB1 ；SUB1はレーベル

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
CALL	near-proc	3
	regptr16	2
	memptr16	2-4
	far-proc	5
	memptr32	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CALL	near-proc	1	1	1	0	1	0	0	0	disp-low							
		disp-high								—							
	regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg		
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	far-proc	1	0	0	1	1	0	1	0	offset-low							
		offset-high								seg-low							
		seg-high								—							
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem		
		(disp-low)								(disp-high)							

保守／廃止

CHKIND

インデクス値チェック
Check Index

【命 令 形 式】　**CHKIND reg16, mem32**

【オペレーション】　(mem32) >reg16または (mem32+2) <reg16のとき

(SP−1, SP−2) ←PSW

(SP−3, SP−4) ←PS

(SP−5, SP−6) ←PC

SP←SP−6

IE←0

BRK←0

PS← (017H, 016H)

PC← (015H, 014H)

【オ ペ ラ ン ド】

ニモニック	オペランド
CHKIND	reg16, mem32

【有効オーバライド・プリフィクス】

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】　割り込み条件成立のとき

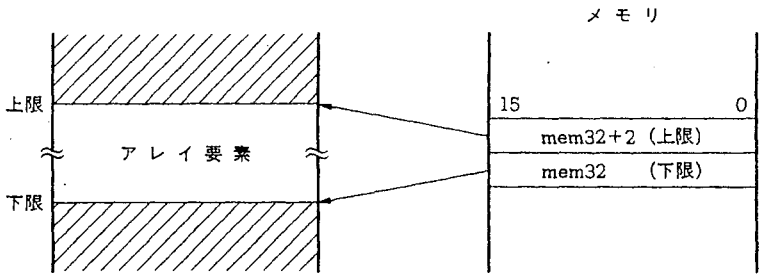
AC	CY	V	P	S	Z	IE	BRK
						0	0

割り込み条件不成立のとき

AC	CY	V	P	S	Z	IE	BRK

保守／廃止

【説　　明】　アレイ・タイプのデータ構造において，要素を指定するインデクス値が定義域内にあるか否かをチェックするための命令です。インデクスが定義域を越える場合には，BRK 5命令を起動します。定義域値は，あらかじめメモリ中の２ワード（１ワード目は下限値，２ワード目は上限値）にセットしておきます。
インデクス値は，アレイ操作プログラムが使用しているレジスタ(任意の16ビット・レジスタ）を対象とします。



【記　述　例】　CHKIND AW, DWORD_VAR

【バ　イ　ト　数】　2-4

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem		
		(disp-low)								(disp-high)							

保守／廃止

CLR1

ビットのリセット

Clear Bit

【命　令　形　式】　① **CLR1 dst, src**

② **CLR1 dst**

【オペレーション】　命令形式①：dstのビットn（nはsrcで指定）←0

命令形式②：dst←0

【オ ペ ラ ン ド】　命令形式①

ニモニック	オペランド（dst, src）
CLR1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
CLR1	CY
	DIR

【有効オーバライド・プリフィクス】

命令形式①

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　命令形式①

AC	CY	V	P	S	Z

命令形式②（dst＝CYの場合）

AC	CY	V	P	S	Z
	0				

命令形式②（dst＝DIRの場合）

AC	CY	V	P	S	Z	DIR
						0

【説　　明】　命令形式①：第１オペランドで指定されるデスティネーション・オペランド（dst）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）をリセット（0）し，結果をデスティネーション・オペランド（dst）に格納します。

オペランドがreg8，CLまたはmem8，CLのとき，CLの値は下位３ビット（0-7）のみ有効です。

オペランドがreg16，CLまたはmem16，CLのとき，CLの値は下位４ビット（0-15）のみ有効です。

オペランドがreg8，imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。

オペランドがmem8，imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。

オペランドがreg16，imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。

オペランドがmem16，imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

命令形式②：dst＝CYの場合，CYフラグをリセット（0）します。

dst＝DIRの場合，DIRフラグをリセット（0）します。また，MOVBK，CMPBK，CMPM，LDM，STM，INM，OUTMの各命令の実行時にインデクス・レジスタ（IX，IY）をオートインクリメントするように設定します。

保守／廃止

【記 述 例】

```
CLR1 CY
SHL  AL, 1
BC   $+6
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
CLR1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1
	DIR	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CLR1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	1	0	0	0	—							
	DIR	1	1	1	1	1	1	0	0	—							

保守／廃止

CMP

比較

Compare

【命 令 形 式】　**CMP dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
CMP	reg, reg'	dst－src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL－imm8 [W＝1のとき] AW－imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ ラ グ】

AC	CY	V	P	S	Z'
×	×	×	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）から第２オペランドで指定されるソース・オペランド（src）を減算します。

減算の結果はどこへも格納せず，フラグを変化させます。

【記 述 例】
- CMP BL, BYTE PTR [IX]
- CMP CW, [BP+4]

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
CMP	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CMP	reg, reg'	0	0	1	1	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	1	1	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	1	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

CMP4S

10進比較

Compare Nibble String

【命 令 形 式】 **CMP4S dst-string, src-string**

CMP4S

【オペレーション】 BCDストリング（IY, CL）−BCDストリング（IX, CL）

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
CMP4S	dst-string, src-string
	な し

【有効オーバライド・プリフィクス】

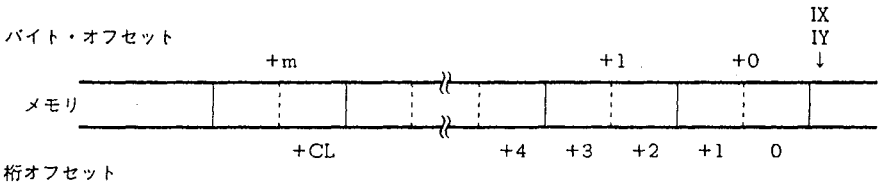
	src	dst
ディフォールト時	DS0：	DS1：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：,	DS1：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
U	×	U	U	U	×

【説　　明】 IYレジスタでアドレスされるパックトBCDストリングからIXレジスタでアドレスされるパックトBCDストリングを減算し，結果はストアされずフラグのみ影響を受けます。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され，1-254桁まで可能です。

デスティネーション・ストリングのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋1の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示します。
したがって，桁数が奇数のときのBCD比較命令は，最上位バイトの上位4ビットを“0”にしてから実行してください。

【記　述　例】

```
MOV IX, OFFSET VAR_1
MOV IY, OFFSET VAR_2
MOV CL, 4
CMP4S
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CMP4S	dst-string, src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0
	な　し																

保守／廃止

CMPBK
CMPBKB
CMPBKW

ブロック比較
Compare Block
Compare Block Byte
Compare Block Word

【命　令　形　式】　(repeat) **CMPBK** src-block, dst-block
(repeat) **CMPBKB**
(repeat) **CMPBKW**

【オペレーション】　[W＝0のとき] (IX)－(IY)
DIR＝0：IX←IX＋1, IY←IY＋1
DIR＝1：IX←IX－1, IY←IY－1
[W＝1のとき] (IX＋1, IX)－(IY＋1, IY)
DIR＝0：IX←IX＋2, IY←IY＋2
DIR＝1：IX←IX－2, IY←IY－2

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
CMPBK	src-block, dst-block
CMPBKB	な　し
CMPBKW	

【有効オーバライド・プリフィクス】(CMPBKだけ)

	src	dst
ディフォールト時	DS0：	DS1：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：,	DS1：, DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

保守／廃止

【説　　明】　IXレジスタでアドレスされるブロックからIYレジスタでアドレスされるブロックをバイトまたはワード・データ単位に繰り返し減算を行い，結果をフラグに反映させます。

IXレジスタ，IYレジスタは，次のバイト／ワード処理のために1バイト／ワード・データが処理されるごとに自動的にインクリメント（+1/+2）またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。

バイトかワードかの指定は，CMPBK命令を用いる場合はオペランドの属性によって行われ，CMPBKB命令またはCMPBKW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。

デスティネーション・ブロックのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。一方，ソース・ブロックは，ディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。

【記　述　例】　CMPBK BYTE_VAR1, BYTE_VAR2

【バ　イ　ト　数】　1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CMPBK	src-block, dst-block	1	0	1	0	0	1	1	W
CMPBKB	な　し								
CMPBKW									

保守／廃止

CMPM
CMPMB
CMPMW

アキュームレータとのブロック比較

Compare Multiple

Compare Multiple Byte

Compare Multiple Word

【命　令　形　式】　(repeat) **CMPM** dst-block

(repeat) **CMPMB**

(repeat) **CMPMW**

【オペレーション】　[W＝0のとき] AL－（IY）

DIR＝0：IY←IY＋1

DIR＝1：IY←IY－1

[W＝1のとき] AW－（IY＋1，IY）

DIR＝0：IY←IY＋2

DIR＝1：IY←IY－2

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
CMPM	dst-block
CMPMB	な　し
CMPMW	

【有効オーバライド・プリフィクス】（CMPMだけ）

	dst
ディフォールト時	DS1：
プリフィクス付加時	DS1：，DS3：，IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

【説　　明】　アキュームレータ（AL/AW）の値からIYレジスタでアドレスされるブロックをバイトまたはワード・データ単位に繰り返し減算を行い，結果をフラグに反映させます。IYレジスタは，次のバイト／ワード処理のために1バイト／ワード・データが処理

されるごとに自動的にインクリメント（+1/+2）またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，CMPM命令を用いる場合はオペランドの属性によって行われ，CMPMB命令またはCMPMW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。

【記　述　例】

```
●MOV    AW, 5555H
 MOV    BW, 1000H
 MOV    IY, BW
 REPC   CMPM WORD PTR [IY]
●REPNC  CMPMW
●REPZ   CMPMB
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CMPM	dst-block	1	0	1	0	1	1	1	W
CMPMB	な　し								
CMPMW									

保守／廃止

CNVTRP

画素データの生成

Convert Turning Point

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**CNVTRP**

【オペレーション】　変化点テーブルの１ライン分の変化点情報を画素データに変換

【オ ペ ラ ン ド】

ニモニック	オペランド
CNVTRP	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に１ワード（16ビット）単位で送出する命令です。データの先頭はLSBに入ります。

各色（白または黒）の変化点情報（変化点テーブルの１ワード）を画素データに変換する処理ごとに割り込み要求のサンプリングを行います。

なお，16の倍数以外の画素データ数を示す変化点テーブルを使用した場合，最後の16ビットに満たない画素データは送出されません。

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）	（DS2）	拡張セグメント・オーバライド・プリフィクス[注1]
02H	（プリント・バッファ・セグメント）	（DS3）	
04H	ワーク・エリア		
06H	ワーク・エリア		
08H	変化点テーブル・セグメント	（DS0）	
0AH	変化点テーブル・オフセット	（ポインタ）	
0CH	ワーク・エリア		
0EH	プリント・バッファ・セグメント	（DS1）	
10H	プリント・バッファ・オフセット	（ポインタ）	
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	処理中半端画素データ		
1CH	0クリア[注2]/処理中半端画素ビット数		0クリア必要パラメータ
1EH	0クリア[注2]/ワーク・エリア		

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. 新規命令処理時に0クリアを行います。

注意　中断処理はありません。

【記　述　例】　CNVTRP

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CVNVTRP	な　し	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	0

保守／廃止

COLTRP

画素データの変化点情報を生成
Collect Turning Point

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　COLTRP

【オペレーション】　1ライン分の画素データの変化点情報（ラン・レングス）を変化点テーブル（1ワード／変化点）に格納（先頭白ラン・レングス）

【オ ペ ラ ン ド】

ニモニック	オペランド
COLTRP	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　符号化処理において，1ライン分の画素データの変化点情報（白／黒のラン・レングス）を指定されたメモリ（変化点テーブル）上に生成する命令です。データの先頭はLSBに入ります。

変化点テーブルは，白のラン・レングスを先頭に1ワードごとに各色（白／黒）のラン・レングスを順次格納します。変化点テーブルの最終ワードには，1ライン分の終了情報として“FFFFH”が格納されます。

中断処理はありません。1ワード分の画素データ処理ごとに割り込み要求のサンプリングを行います。

（画素データ）

●●●●○●○○○○○●●●○○●○…

↓　COLTRP命令発行

（変化点テーブル）

白 黒 白 黒 白 黒 白 黒

0	4	1	1	5	3	2	1	・・・	FFFFH

└ 1ワード

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号

ALで指定されるレジスタ・バンク

00H	(画素データ・バッファ・セグメント)	(DS2)	拡張セグメント・オーバライド・プリフィクス[注1]
02H	(変化点テーブル・セグメント)	(DS3)	
04H	ワーク・エリア		
06H	ワーク・エリア		
08H	画素データ・バッファ・セグメント	(DS0)	入力データ処理情報
0AH	画素データ・バッファ・オフセット	(ポインタ)	
0CH	画素データ・バッファ・サイズ[注2]	(カウンタ)	
0EH	変化点テーブル・セグメント	(DS1)	出力データ処理情報
10H	変化点テーブル・オフセット	(ポインタ)	
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	0クリア[注3]/ワーク・エリア		0クリア必要パラメータ
1EH	0クリア[注3]/ワーク・エリア		

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. バイト単位。偶数で指定してください。

3. 新規命令処理時に0クリアを行います。

注意　各命令で使用されるパラメータ用のレジスタ・バンク内のポインタ，カウンタ値は，命令実行によって更新されます。

【記　述　例】　COLTRP

【バ　イ　ト　数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
COLTRP	な　し	0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	1

保守／廃止

CVTBD

2進→アンパック10進変換

Convert Binary to Decimal

【命 令 形 式】 **CVTBD**

【オペレーション】 AH←AL÷0AH

AL←AL%0AH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTBD	な し

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	×	×	×

【説　　明】 ALレジスタの2進数8ビットを2桁のアンパック10進数に変換します。

AHレジスタはALレジスタの値を10で割った商で置き換えられ，次にALレジスタがその乗算の剰余で置き換えられます。

【記 述 例】

```
MOV AL, 30H
CVTBD
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CVTBD	な し	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0

保守／廃止

CVTBW

ワード符号拡張
Convert Byte to Word

【命 令 形 式】　**CVTBW**

【オペレーション】　AL<80Hのとき：AH←0
AL≧80Hのとき：AH←FFH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTBW	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　ALレジスタ内のバイトの符号をAHレジスタに拡張します。バイト除算を実行する前に，あるバイトから倍長（ワード）の被除数を得るのに有効です。

【記 述 例】
```
MOV    AL, BUF1 ; BUF1はバイト変数
CVTBW
MOV    DL, 60
DIV    DL
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CVTBW	な し	1	0	0	1	1	0	0	0

保守／廃止

CVTDB

アンパック10進→2進変換

Convert Decimal to Binary

【命 令 形 式】 **CVTDB**

【オペレーション】 AL←AH×0AH+AL

AH←0

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTDB	な し

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	×	×	×

【説　　明】 AHレジスタおよびALレジスタの2桁のアンパック10進数を16ビットの2進数に変換します。

ALレジスタはAHレジスタの値に10を掛けた結果にALレジスタの値を加えたもので置き換えられ，AHレジスタは0で置き換えられます。

【記 述 例】

```
MOV AW, [BW]
CVTDB
```

【バ イ ト 数】 2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CVTDB	な し	1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0

保守／廃止

CVTWL

ダブル・ワード符号拡張
Convert Word to Long Word

【命 令 形 式】 **CVTWL**

【オペレーション】 AW＜8000Hのとき：DW←0
AW≧8000Hのとき：DW←FFFFH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTWL	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 AWレジスタのワードの符号をDWレジスタに拡張します。ワード除算を実行する前に，あるワードから倍長（ダブル・ワード）の被除数を得るのに有効です。

【記 述 例】
```
MOV    AW, BUFFER
CVTWL
DIV    CW
```

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CVTWL	な し	1	0	0	1	1	0	0	1

保守／廃止

DBNZ

CW≠0による条件ループ

Decrement and Branch if Not Zero

【命 令 形 式】　**DBNZ short-label**

【オペレーション】　CW←CW－1

CW≠0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
DBNZ	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CWレジスタの値をデクリメント（－1）し，その結果CWレジスタの値が０でなければ，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。

この命令に置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記　述　例】

```
LP21：
        ⋮
        SHL   AL, 1
        DBNZ LP21   ；LP21＝レーベル
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZ	short-label	1	1	1	0	0	0	1	0	disp8							

保守／廃止

DBNZE

CW≠0かつZ＝1による条件ループ

Decrement and Branch if Not Zero and Equal

【命　令　形　式】　**DBNZE short-label**

【オペレーション】　CW←CW－1

CW≠0でZ＝1のとき：PC←PC＋ext-disp8

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
DBNZE	short-label

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CWレジスタの値をデクリメント（－1）し，その結果CWレジスタの値が0でなく，かつZフラグがセット（1）されていれば，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記　述　例】

```
LP20 :
        ⋮
        AND   AL, BL
        DBNZE LP20   ; LP20＝レーベル
```

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZE	short-label	1	1	1	0	0	0	0	1	disp8							

保守／廃止

DBNZNE

CW≠0かつZ=0による条件ループ

Decrement and Branch if Not Zero and Not Equal

【命 令 形 式】　**DBNZNE short-label**

【オペレーション】　CW←CW－1

CW≠0でZ=0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
DBNZNE	short-label

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　CWレジスタの値をデクリメント（－1）したあと，CWレジスタの値が0でなくZフラグがクリアされていれば，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。

この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

条件不成立のときは次の命令に進みます。

【記　述　例】

```
LP19:
        ⋮
        AND     AL, 0FFH
        DBNZNE SHORT LP19   ; LP19=レーベル
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZNE	short-label	1	1	1	0	0	0	0	0	disp8							

保守／廃止

DEC

デクリメント
Decrement

【命 令 形 式】 **DEC dst**

【オペレーション】 dst←dst−1

【オ ペ ラ ン ド】

ニモニック	オペランド
DEC	reg8
	mem
	reg16

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
×		×	×	×	×

【説　　明】 デスティネーション・オペランド（dst）の内容をデクリメント（−1）します。

【記 述 例】
- DEC BW
- DEC BP
- DEC IX
- DEC IY

【バ イ ト 数】

ニモニック	オペランド	バイト数
DEC	reg8	2
	mem	2-4
	reg16	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1	reg		
	mem	1	1	1	1	1	1	1	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	0	1	reg			—							

保守／廃止

DI

マスカブル割り込みの禁止
Disable Interrupt

【命 令 形 式】　**DI**

【オペレーション】　IE←0

【オ ペ ラ ン ド】

ニモニック	オペランド
DI	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z	IE
						0

【説　　明】　IEフラグをリセット（0）し，マスカブル割り込み要求を禁止します。
この命令によって，ノンマスカブル割り込み入力（NMI）およびソフトウエア割り込み要求は禁止されません。

【記　述　例】
```
DI
PUSH  R
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DI	な　し	1	1	1	1	1	0	1	0

保守／廃止

DISPOSE

スタック・フレームの削除
Dispose a Stack Frame

【命 令 形 式】 **DISPOSE**

【オペレーション】 SP←BP
BP← (SP+1, SP)
SP←SP+2

【オ ペ ラ ン ド】

ニモニック	オペランド
DISPOSE	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 この命令は，PREPARE命令で生成されたスタック・フレームを1フレーム分リリースします。
BPには，1つ前のフレームを指すポインタ値をロードし，SPには，フレームの最下位を示すポインタ値をロードします。

【記 述 例】 DISPOSE

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DISPOSE	な し	1	1	0	0	1	0	0	1

保守／廃止

DIV

符号付き除算

Divide Signed

【命 令 形 式】　DIV dst

【オペランド，オペレーション】

ニモニック	オペランド（dst）	オペレーション
DIV	reg8	temp←AW temp÷dst＞0でtemp÷dst≦7FHまたは temp÷dst＜0でtemp÷dst＞0−7FH−1のとき 　AH←temp%dst 　AL←temp÷dst temp÷dst＞0でtemp÷dst＞7FHまたは temp÷dst＜0でtemp÷dst≦0−7FH−1のとき 　商と剰余は不定
	mem8	(SP−1，SP−2) ←PSW 　(SP−3，SP−4) ←PS 　(SP−5，SP−6) ←PC 　SP←SP−6 　IE←0，BRK←0 　PS←（003H，002H） 　PC←（001H，000H）
	reg16	temp←DW，AW temp÷dst＞0でtemp÷dst≦7FFFHまたは temp÷dst＜0でtemp÷dst＞0−7FFFH−1のとき 　DW←temp%dst 　AW←temp÷dst temp÷dst＞0でtemp÷dst＞7FFFHまたは temp÷dst＜0でtemp÷dst≦0−7FFFH−1のとき 　商と剰余は不定
	mem16	(SP−1，SP−2) ←PSW 　(SP−3，SP−4) ←PS 　(SP−5，SP−6) ←PC 　SP←SP−6 　IE←0，BRK←0 　PS←（003H，002H） 　PC←（001H，000H）

保守／廃止

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	U	U	U	U	U

【説　　明】　○dst＝reg8またはsrc＝mem8のとき

AWレジスタの値をデスティネーション・オペランド（dst）の内容で符号付き除算します。商はALレジスタに格納し，余りはAHレジスタに格納します。

正の商の最大値は＋127（7FH）で，負の商の最小値は－127（81H）です。商が正で，最大値を越えたとき，または商が負で最小値より小さくなったときは，ベクタ0割り込みが発生し，商と余りは不定となります（特にsrc＝00Hのときに発生します）。また，整数でない商は整数に切り縮められ，余りは被除数と同じ符号を持ちます。

○dst＝reg16またはsrc＝mem16のとき

AWレジスタの値とDWレジスタの値をデスティネーション・オペランド（dst）の内容で符号付き除算します。商はAWレジスタに格納し，余りはDWレジスタに格納します。

正の商の最大値は＋32767（7FFFH）で，負の商の最小値は－32767（8001H）です。商が正で，最大値を越えたとき，または商が負で最小値より小さくなったときは，ベクタ0割り込みが発生し，商と余りは不定となります（特にsrc＝0000Hのときに発生します）。また，整数でない商は整数に切り縮められ，余りは被除数と同じ符号を持ちます。

【記　述　例】　32ビット・データDW：AWをメモリ0：50の内容で割る。

```
MOV   BW, 0
MOV   DS0, BW
MOV   IX, 50H
DIV   DS0：WORD  PTR [IX]
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
DIV	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DIV	reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg		
	mem8	1	1	1	1	0	1	1	0	mod			1	1	1	mem	
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	1	reg		
	mem16	1	1	1	1	0	1	1	1	mod			1	1	1	mem	
		(disp-low)								(disp-high)							

保守／廃止

DIVU

符号なし除算
Divide Unsigned

【命 令 形 式】　DIVU dst

【オペランド，オペレーション】

ニモニック	オペランド（dst）	オペレーション
DIVU	reg8	temp←AW temp÷dst≦FFHのとき： AH←temp%dst AL←temp÷dst temp÷dst>FFHのとき： (SP−1, SP−2) ←PSW (SP−3, SP−4) ←PS (SP−5, SP−6) ←PC SP←SP−6 IE←0, BRK←0 PS← (003H, 002H) PC← (001H, 000H)
	mem8	
	reg16	temp←DW, AW temp÷dst≦FFFFHのとき： DW←temp%dst AW←temp÷dst temp÷dst>FFFFHのとき： (SP−1, SP−2) ←PSW (SP−3, SP−4) ←PS (SP−5, SP−6) ←PC SP←SP−6 IE←0, BRK←0 PS← (003H, 002H) PC← (001H, 000H)
	mem16	

保守／廃止

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	U	U	U	U	U

【説　　明】

○src＝reg8またはsrc＝mem8のとき

AWレジスタの値をデスティネーション・オペランド（dst）の内容で符号なし除算します。商はALレジスタに格納し，余りはAHレジスタに格納します。

商がALレジスタの容量（FFH）を越えた場合は，ベクタ0割り込みが発生し，商と余りは不定となります(特にsrc＝00Hのときに発生します)。また，整数でない商は整数に切り縮められます。

○src＝reg16またはsrc＝mem16のとき

AWレジスタの値とDWレジスタの値をデスティネーション・オペランド（dst）の内容で符号なし除算します。商はAWレジスタに格納し，余りはDWレジスタに格納します。

商がAWレジスタの容量(FFFFH)を越えた場合，ベクタ0割り込みが発生し，商と余りは不定となります(特にsrc＝0000Hのときに発生します)。また，整数でない商は整数に切り縮められます。

【記　述　例】　5÷3の除算をする。

```
MOV     AW, 5
MOV     DL, 3
DIVU    DL
;AH=2  AL=1
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
DIVU	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg		
	mem8	1	1	1	1	0	1	1	0	mod			1	1	0	mem	
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg		
	mem16	1	1	1	1	0	1	1	1	mod			1	1	0	mem	
		(disp-low)								(disp-high)							

保守／廃止

DS0：
DS1：
PS：
SS：

セグメント・オーバライド・プリフィクス
Data Segment 0
Data Segment 1
Program Segment
Stack Segment

【命 令 形 式】　DS0：
DS1：
PS：
SS：

【オペレーション】　セグメント・オーバライド・プリフィクス

【オ ペ ラ ン ド】

ニモニック	オペランド
DS0：	な し
DS1：	
PS：	
SS：	

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　セグメント・オーバライド可能なメモリ・オペランドをアクセスする際に，オペランドに付記してそのとき使用されるセグメント・レジスタを指定します。この命令を直接記述しなくとも，ASSUME（アセンブラ疑似命令）を使用することによって，セグメント・オーバライド指定をアセンブラに行わせることができます。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　MOV　DW，DS1：［BW］；ディフォールトのセグメント・レジスタはDS0

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DS0：	な　し	0	0	1	sreg		1	1	0
DS1：									
PS：									
SS：									

保守／廃止

DS2：
DS3：

拡張セグメント・オーバライド・プリフィクス

Data Segment2：

Data Segment3：

（V20，V30とV25，V35に対する追加命令）

【命令形式】　**DS2**：

DS3：

【オペレーション】　拡張セグメント・オーバライド・プリフィクス

【オペランド】

ニモニック	オペランド
DS2：	な　し
DS3：	

【フラグ】

AC	CY	V	P	S	Z

【説明】　セグメント・オーバライド可能なメモリ・オペランドをアクセスする際に，オペランドに付記してそのとき使用される拡張セグメント・レジスタを指定します。この命令を直接記述しなくても，ASSUME(アセンブラ疑似命令)を使用することによってセグメント・オーバライド指定をアセンブラに行わせることができます。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記述例】　MOV DW，DS2：〔BW〕；ディフォールトのセグメント・レジスタはDS0

【バイト数】　1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DS2：	な　し	0	1	1	0	0	0	1	1
DS3：									

保守／廃止

EI

マスカブル割り込みの許可

Enable Interrupt

【命 令 形 式】　**EI**

【オペレーション】　IE←1

【オ ペ ラ ン ド】

ニモニック	オペランド
EI	な し

【フ ラ グ】

AC	CY	V	P	S	Z	IE
						1

【説　　明】　IEフラグをセット（1）し，マスカブル割り込み要求を許可します。
ただし，実際に割り込み許可状態になるのはEI命令に続く1命令を実行したあとです。

【記 述 例】
```
POP  R
EI
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
EI	な し	1	1	1	1	1	0	1	1

保守／廃止

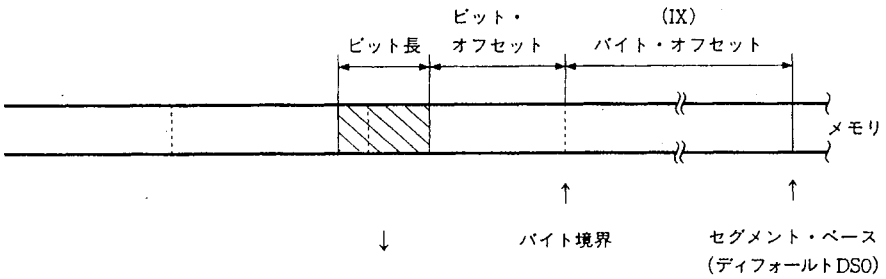
EXT

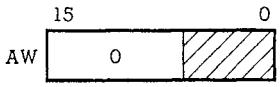
ビット・フィールドの抽出

Extract Bit Field

【命 令 形 式】　**EXT dst, src**

【オペレーション】　AW←16ビット・フィールド





【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
EXT	reg8, reg8'
	reg8, imm4

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS2：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	U	U	U

保守／廃止

【説　　明】　デスティネーションのディフォールト・セグメント・レジスタはDS0となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS2で指定されるセグメント内にもロケートできます。

セグメント・レジスタDS0または拡張セグメント・レジスタDS2およびインデクス・レジスタ（IX）でアドレスされるバイト・オフセットおよび第1オペランドの8ビット・レジスタで指定されるビット・オフセットによって決定されるメモリ領域から，ソース・オペランド（src）によって指定されるビット長のビット・フィールド・データをAWレジスタへロードします。このときAWレジスタの余りの上位ビットには0がロードされます。

転送終了後，IXレジスタおよび第1オペランドで指定される8ビット・レジスタは次のビット・フィールドを示すために自動的に次のように更新されます。

```
reg8←reg8+src+1
if reg8>15 then
        {
        reg8←reg8−16
        IX←IX+2
        }
```

ビット・オフセット（最長15ビット）を指定する第1オペランドの8ビット・レジスタの値は0-15のみ有効です。また、ビット長(最長16ビット)を指定するソース・オペランド（src）の値は0-15のみが有効で，0が1ビット長を，15が16ビット長を指定します。

ビット・フィールド・データはメモリのバイト境界にまたがることができます。

注意　reg8またはreg8'の上位4ビットは0にしてください。

【記　述　例】
- EXT CL, DL
- EXT CL, 8

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
EXT	reg8, reg8'	3
	reg8, imm4	4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
EXT	reg8, reg8′	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1
		1	1	reg′			reg			—							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1
		1	1	0	0	0	reg			imm4							

保守／廃止

FINT

割り込みコントローラに対する割り込み処理の終了

Finish Interrupt

（V20，V30に対する追加命令）

【命 令 形 式】　**FINT**

【オペレーション】　CPUに内蔵される割り込みコントローラへ，割り込み処理ルーチンが終了したことを示します。

【オ ペ ラ ン ド】

ニモニック	オペランド
FINT	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　内蔵する割り込みコントローラに対して，NMIとWDT，およびソフトウエア割り込み（例外トラップを含む）を除く割り込み処理が終了したことを知らせます（実際にはISPRレジスタをビット0から順にサーチし，最初に検索した1をクリア(0)します)。必ず，割り込み処理プログラムが終了する直前(RETI命令またはRETRBI命令の直前）に使用してください。それ以外では使用しないでください。

【記　述　例】　FINT

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FINT	な　し	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	0

保守／廃止

FPO1

浮動小数点演算用コプロセッサ制御

Floating Point Operation 1

【命　令　形　式】　① **FPO1 fp-op**

② **FPO1 fp-op, mem**

【オペランド，オペレーション】

命令形式①，②

ニモニック	オペランド	オペレーション
FPO1	fp-op	(SP−1, SP−2) ←PSW (SP−3, SP−4) ←PS (SP−5, SP−6) ←PC−x[注] SP←SP−6 IE←0, BRK←0 PC← (01DH, 01CH) PS← (01FH, 01EH)
	fp-op, mem	

注　xは命令のバイト数＋プリフィクスの数

【フ　ラ　グ】

AC	CY	V	P	S	Z	IE	BRK
						0	0

【説　　明】　PSW，PS，PC−xをスタックに退避し，IEとBRKフラグをリセット(0)します。次に割り込みベクタ・テーブルのベクタ7の下位2バイトをPCに，上位2バイトをPSにロードします。

この命令はV20，V30の互換性を保つように用意された命令です。V55PIとしては特に機能的に意味を持たず，割り込みを発生するだけです。

【記　述　例】
- FPO1　010101010B
- FPO1　0FFH
- FPO1　6, BYTE PTR [IX]
- FPO1　4, WORD_VAR

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
FPO1	fp-op	2
	fp-op, mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FPO1	fp-op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z
	fp-op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem		
		(disp-low)								(disp-high)							

保守／廃止

FPO2

浮動小数点演算用コプロセッサ制御
Floating Point Operation 2

【命　令　形　式】　① **FPO2 fp-op**
② **FPO2 fp-op, mem**

【オペランド，オペレーション】

命令形式①，②

ニモニック	オペランド	オペレーション
FPO2	fp-op	(SP−1, SP−2) ←PSW (SP−3, SP−4) ←PS (SP−5, SP−6) ←PC−x[注] SP←SP−6 IE←0, BRK←0 PC← (01DH, 01CH) PS← (01FH, 01EH)
	fp-op, mem	

注　xは命令のバイト数＋プリフィクスの数

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　PSW，PS，PC−xをスタックに退避し，IEとBRKフラグをリセット(0)します。
次に割り込みベクタ・テーブルのベクタ7の下位2バイトをPCに，上位2バイトをPSにロードします。
この命令はV20，V30の互換性を保つために用意された命令です。V55PIとしては特に機能的に意味を持たず，割り込みを発生するだけです。

【記　述　例】
- FPO2　010101010B
- FPO2　0FFH
- FPO2　0101B, BYTE　PTR [IY]
- FPO2　1010B, WORD_VAR

【バ イ ト 数】

ニモニック	オペランド	バイト数
FPO2	fp-op	2
	fp-op, mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FPO2	fp-op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z
	fp-op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem		
		(disp-low)								(disp-high)							

保守／廃止

GETBIT

1ビット検出
GET BIT

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**GETBIT**

【オペレーション】　画素データの先頭ビットを取り出し，CYフラグに設定

【オ ペ ラ ン ド】

ニモニック	オペランド
GETBIT	な　し

【フ　ラ　グ】

Z	CY	受信バッファ残り	受信データ・バッファ・サイズ
0	注	あ　り	受信データ・バッファ・サイズ>0
1	注	な　し	受信データ・バッファ・サイズ=0

注　検出した1ビット・データ

注意　半端符号データの有無にかかわらず，受信データ・バッファ・サイズのカウンタ（レジスタ・バンクのOCHの内容）が“0”のとき，Zフラグ＝1になります。

【説　　明】　復号化処理において，タグなどの1ビットを検出するための命令です。データの先頭と処理中の半端符号はLSBから入ります。

メモリ上（受信バッファ）の指定された位置から1ビットを取り出し，CYフラグに設定します。

割り込み要求のサンプリングを行いません。

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	ワーク・エリア		
04H	処理中半端符号データ		
06H	処理中半端符号ビット数		
08H	受信バッファ・セグメント	(DS0)	
0AH	受信バッファ・オフセット	（ポインタ）	
0CH	受信バッファ・サイズ注2	（カウンタ）	
0EH	ワーク・エリア		
10H	ワーク・エリア		
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	ワーク・エリア		
1EH	ワーク・エリア		

注1．DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2．バイト単位。0以外の偶数値を設定してください。

【記　述　例】　GETBIT

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
GETBIT	な　し	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	1

保守／廃止

HALT

ホールト
Halt

【命 令 形 式】 **HALT**

【オペレーション】 CPU Halt

【オ ペ ラ ン ド】

ニモニック	オペランド
HALT	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 CPUへのクロック供給を停止し，ホールト状態にします。
ホールト状態は次の４つの要因によって解除されます。

- リセット入力
- マスカブル割り込み要求入力
- NMI
- ウォッチドッグ・タイマ割り込み要求入力

【記 述 例】 HALT

【バ イ ト 数】 1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
HALT	な し	1	1	1	1	0	1	0	0

保守／廃止

IN

I/Oデバイスからのデータ入力

Input

【命　令　形　式】　**IN dst, src**

【オペランド，オペレーション】

○$\overline{IBRK}$＝1のとき

ニモニック	オペランド（dst, src）	オペレーション
IN	acc, imm8	[W＝0のとき]AL←(imm8) [W＝1のとき]AH←(imm8＋1), AL←(imm8)
	acc, DW	[W＝0のとき]AL←(DW) [W＝1のとき]AH←(DW＋1), AL←(DW)

○$\overline{IBRK}$＝0のとき

ニモニック	オペランド（dst, src）	オペレーション
IN	acc, imm8	(SP－1, SP－2) ←PSW (SP－3, SP－4) ←PS (SP－5, SP－6) ←PC－x[注]
	acc, DW	IE←0, BRK←0, $\overline{IBRK}$←1 PC← (019H, 018H) PS← (01BH, 01AH)

注　xは命令のバイト数＋プリフィクスの数

【フ　ラ　グ】　○$\overline{IBRK}$＝1のとき

AC	CY	V	P	S	Z

○$\overline{IBRK}$＝0のとき

AC	CY	V	P	S	Z	IE	BRK	$\overline{IBRK}$
						0	0	1

保守／廃止

【説　　明】　○$\overline{IBRK}$＝1のとき

デスティネーション・オペランド（dst）で指定されるアキュームレータ（AL/AW）に，ソース・オペランド（src）で指定されるI/Oデバイスのレジスタ内容を転送します。

○$\overline{IBRK}$＝0のとき

PSW，PS，PC－xをスタックに退避し，IEフラグとBRKフラグをリセット(0)し，$\overline{IBRK}$フラグをセットします。

次に割り込みベクタ・テーブルのベクタ６の下位２バイトをPCに，上位２バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするための機能です。

【記　述　例】　ポート・アドレス0DAHの内容をALレジスタに転送する。

```
MOV  DW, 0DAH
IN   AL, DW
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
IN	acc, imm8	2
	acc, DW	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
IN	acc, imm8	1	1	1	0	0	1	0	W	imm8							
	acc, DW	1	1	1	0	1	1	0	W	—							

保守／廃止

INC

インクリメント
Increment

【命 令 形 式】　**INC dst**

【オペレーション】　dst←dst+1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
INC	reg8
	mem
	reg16

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
×		×	×	×	×

【説　　明】　デスティネーション・オペランド（dst）の内容をインクリメント（+1）します。

【記 述 例】
- INC DW
- INC BP
- INC SP

【バ イ ト 数】

ニモニック	オペランド	バイト数
INC	reg8	2
	mem	2-4
	reg16	1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0	reg		
	mem	1	1	1	1	1	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	0	0	reg			—							

保守／廃止

INM

I/O-メモリのブロック転送

Input Multiple

【命　令　形　式】　(repeat) INM dst-block, DW

【オペレーション】　○$\overline{IBRK}$＝1のとき

[W＝0のとき] (IY) ← (DW)

DIR＝0：IY←IY＋1

DIR＝1：IY←IY－1

[W＝1のとき] (IY＋1, IY) ← (DW＋1, DW)

DIR＝0：IY←IY＋2

DIR＝1：IY←IY－2

○$\overline{IBRK}$＝0のとき

(SP－1, SP－2) ←PSW

(SP－3, SP－4) ←PS

(SP－5, SP－6) ←PC－x[注]

IE←0, BRK←0, $\overline{IBRK}$←1

PC← (019H, 018H)

PS← (01BH, 01AH)

注　xは命令のバイト数＋プリフィクスの数

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
INM	dst-block, DW

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS1：
プリフィクス付加時	DS1：, DS3：, IRAM：

保守／廃止

【フ　ラ　グ】　○$\overline{IBRK}$＝1のとき

AC	CY	V	P	S	Z

○$\overline{IBRK}$＝0のとき

AC	CY	V	P	S	Z	IE	BRK	$\overline{IBRK}$
						0	0	1

【説　　明】　○$\overline{IBRK}$＝1のとき

DWレジスタでアドレスされるI/Oデバイスのレジスタ内容を，IYレジスタでアドレスされるメモリに転送します。繰り返し転送回数は，ペアで使用されるリピート・プリフィクスのREP命令によって制御されます。繰り返し転送の際，DWレジスタの内容（I/Oデバイスのアドレス）は固定ですが，IYレジスタは次のバイト／ワード転送のために1バイト／ワード・データが転送されるごとに自動的にインクリメント（+1/+2）またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。

バイトかワードかの指定は，オペランドの属性によって行われます。

INM命令は，リピート・プリフィクスのREP命令とともに使用されます。

デスティネーション・ブロックのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。

○$\overline{IBRK}$＝0のとき

PSW，PS，PC−xをスタックに退避し，IEフラグとBRKフラグをリセット(0)し，$\overline{IBRK}$フラグをセットします。

次に割り込みベクタ・テーブルのベクタ6の下位2バイトをPCに，上位2バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするための機能です。

保守／廃止

【記　述　例】　●メモリ・ワーク・エリアにポート・アドレス0DAHの内容（バイト・データ）をロードする。

```
MOV  AW, 0
MOV  DS1, AW
MOV  IY, 50H
MOV  DW, 0DAH
INM  DS1 : BYTE  PTR [IY], DW
```

●メモリ0：0H～0：FFHにポート・アドレス0DAHの内容（バイト・データ）をロードする。

```
MOV  AW, 0
MOV  DS1, AW
MOV  IY, 0
MOV  DW, 0DAH
MOV  CW, 0FFH
REP  INM  DS1 : BYTE  PTR [IY], DW
```

【バ　イ　ト　数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
INM	dst-block, DW	0	1	1	0	1	1	0	W

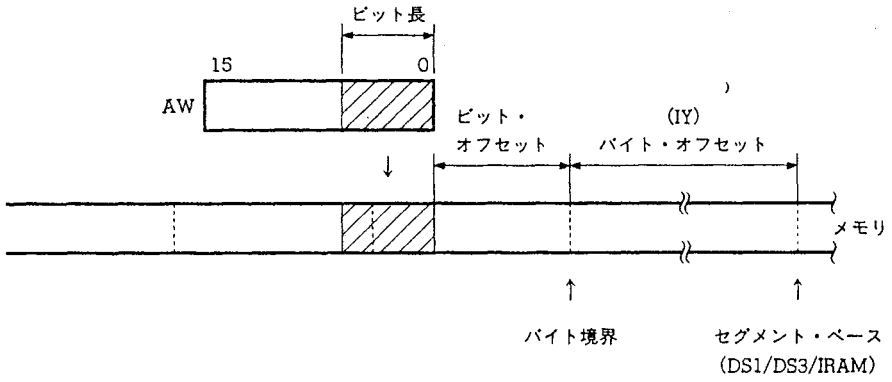
保守／廃止

INS

ビット・フィールドの挿入
Insert Bit Field

【命 令 形 式】　**INS dst, src**

【オペレーション】　16ビット・フィールド←AW



【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
INS	reg8, reg8'
	reg8, imm4

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS1：
プリフィクス付加時	DS1：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	U	U	U

【説　　明】　デスティネーションのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。

AWレジスタの16ビット・データのうち，ソース・オペランド（src）で指定される

保守／廃止

長さの下位ビット・データをDS1レジスタまたは拡張セグメント・レジスタDS3およびIYレジスタでアドレスされるバイト・オフセット，および第１オペランドの８ビット・レジスタで指定されるビット・オフセットによって決定されるメモリ領域へ転送します。
転送終了後，IYレジスタおよび第１オペランドで指定される８ビット・レジスタは，次のビット・フィールドを示すために自動的に次のように更新されます。

```
reg8←reg8+src+1
if reg8>15 then
        {
        reg8←reg8−16
        IY←IY+2
        }
```

ビット・オフセット（最長15ビット）を指定する第１オペランドの８ビット・レジスタの値は0-15だけが有効です。また，ビット長（最長16ビット）を指定するソース・オペランド（src）の値は0-15だけが有効で，０が１ビット長を，15が16ビット長を指定します。
ビット・フィールド・データはメモリのバイト境界にまたがることができます。

注意　reg8またはreg8'の上位４ビットは０にしてください。

【記　述　例】

- INS DL, CL
- INS DL, 12

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
INS	reg8, reg8'	3
	reg8, imm4	4

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
INS	reg8, reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1
		1	1	reg'			reg			—							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1
		1	1	0	0	0	reg			imm4							

保守／廃止

IRAM：

レジスタ・ファイル空間アクセス用オーバライド・プリフィクス

Internal RAM：

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　IRAM：

【オペレーション】　レジスタ・ファイル用オーバライド・プリフィクス

【オ ペ ラ ン ド】

ニモニック	オペランド
IRAM：	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　メモリ操作命令に付加することによりメイン・メモリ空間と切り離して（セグメント値に関係なく），512バイトのレジスタ・ファイル空間に対しデータ・アクセスします。レジスタ・ファイル空間内のアドレスは，メモリ指定オペランドのオフセット値の下位９ビットとなります。

【記　述　例】　MOV IRAM：〔0100H〕，AW

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
IRAM：	な　し	1	1	1	1	0	0	0	1

保守／廃止

LDEA

実効アドレスのロード
Load Effective Address

【命 令 形 式】　**LDEA reg16, mem16**

【オペレーション】　reg16←mem16

【オ ペ ラ ン ド】

ニモニック	オペランド
LDEA	reg16, mem16

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　第1オペランドで指定される16ビット汎用レジスタに，第2オペランドで生成される実効アドレス（オフセット）をロードします。
TRANS命令やプリミティブ・ブロック転送命令でオペランドを指定するために自動的に用いられるレジスタ等にオペランド・アドレスの先頭値をセット（1）するときなどに用いられます。

【記　述　例】　AWレジスタにINT_PROCという手続きの実行アドレスのオフセットをロードします。

```
LDEA  AW, INT_PROC
LDEA  AW, [BP] VAR01+2
```

【バ イ ト 数】　2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod			reg			mem	
		(disp-low)								(disp-high)							

保守／廃止

LDM
LDMB
LDMW

ブロック・ロード
Load Multiple
Load Multiple Byte
Load Multiple Word

【命　令　形　式】　(repeat) LDM src-block
(repeat) LDMB
(repeat) LDMW

【オペレーション】　[W＝0のとき] AL← (IX)
DIR＝0：IX←IX＋1
DIR＝1：IX←IX－1
[W＝1のとき] AW← (IX＋1, IX)
DIR＝0：IX←IX＋2
DIR＝1：IX←IX－2

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
LDM	src-block
LDMB	な　し
LDMW	

【有効オーバライド・プリフィクス】(LDMだけ)

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　IXレジスタでアドレスされるブロックをバイトまたはワード単位にアキュームレータ（AL/AW）に繰り返し転送します。

IXレジスタは次のバイト／ワード処理のために1バイト／ワード・データが処理されるごとに自動的にインクリメント（+1/+2）またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。

バイトかワードかの指定は，LDM命令を用いる場合はオペランドの属性によって行われ，LDMB命令またはLDMW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。

ソース・ブロックのディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。

【記　述　例】

```
●REP  LDM   DS1 : BYTE_VAR ; DS1セグメント
●REP  LDMB                  ; DS0セグメント
```

【バ　イ　ト　数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
LDM	src-block	1	0	1	0	1	1	0	W
LDMB	な　し								
LDMW									

保守／廃止

MHDEC

MH復号化を実行

MH Decode

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**MHDEC**

【オペレーション】　MH符号から変化点テーブルを生成

【オ ペ ラ ン ド】

ニモニック	オペランド
MHDEC	な　し

【フ　ラ　グ】

Z	CY	受信バッファ残り	終了状態	AHステータス
0	0	あ　り	正常終了	変化しない
0	1	あ　り	終了エラー	**表2－9参照**
1	0	な　し	正常終了	変化しない
1	1	な　し	終了エラー	**表2－9参照**

AH内の数値によって終了状態をチェックできます。数値と終了状態を次に示します。

なお，AH＝01H，02Hの場合は，ALレジスタで指定されるレジスタ・バンクのオフセット・アドレス1AH番地のワーク・エリアの値により，終了状態がそれぞれ2通りあります。

表2－9　MHDEC命令の終了状態

AH	終了状態	
00H	中断	
01H	ワーク・エリア＝0	先頭にEOLを検出
	ワーク・エリア≠0	先頭にメークアップ符号＋EOLを検出
02H	ワーク・エリア＝0	先頭にFILL付きEOLを検出
	ワーク・エリア≠0	先頭にメークアップ符号＋FILL付きEOLを検出
03H	1ライン分の画素数に満たないデータのあと，EOLを検出	
04H	1ライン分の画素数以上に復号化するデータが存在	
05H	異常なコードを検出	

備考　異常コードを検出して終了した場合，複号化ライン変化点テーブルのオフセットは，

最後に正常出力された変化点テーブル・データの次のワード領域のアドレスを指しています。

【説　　明】　指定されたメモリ（受信バッファ）上の符号データを入力し，復号化変換テーブルによりMH復号化を行い，生成された変化点情報を指定されたメモリ(変化点テーブル）上に送出する命令です。データの先頭と処理中の半端符号はLSBから入ります。変化点情報を生成することにより，１ライン分の画素データ・ビット数に達したことを認識し命令を終了とします。
MH復号化の符号データから変化点テーブルを作成する処理において,次のような場合が考えられます。

① 指定された受信バッファ内に１ライン分の符号データがあり，変化点情報への変換を終了する場合（正常終了)。
② 指定された受信バッファ内の符号が１ライン分に満たなくて，変化点情報への変換が途中で終了する場合（中断)。
③ １番先頭にEOLを検出する場合。
④ FILL付きのEOLを先頭に検出する場合。
⑤ １ライン分の画素数に満たないデータのあと，EOLを検出する場合。
⑥ １ライン分の画素数以上に復号化するデータがある場合。
⑦ 異常なコードを検出する場合。

正常終了のときだけ，変化点テーブルに“FFFFH”（１ライン分の終了コード）を出力します。
各色（白または黒）の変化点を生成する復号化処理ごとに，割り込み要求のサンプリングを行います。
MHDEC命令において，復号化できる１ラインの最大画素数は合計65535画素です。
MHDEC命令実行中に“0”の連続する異常符号を検出すると，次に“1”を検出するまで割り込み要求が保留されます。この場合，受信バッファ・サイズを小さく設定してMHDEC命令が一定の時間で中断されるようにすることにより,長時間にわたる割り込み要求の保留を回避することができます。MHDEC命令の実行時間(最長となる場合)の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません)。

- 16ビット・バス幅のとき：40+2T+（59+T）N/16（クロック）
- ８ビット・バス幅のとき：44+4T+（37+2T）N/8（クロック）

保守／廃止

N：“0”の連続するビット数

T：変化点テーブルのウエイト数

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	(受信バッファ・セグメント) (DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	(変化点テーブル・セグメント) (DS3)	
04H	処理中半端符号データ	
06H	処理中半端符号ビット数	
08H	受信バッファ・セグメント (DS0)	
0AH	受信バッファ・オフセット (ポインタ)	
0CH	受信バッファ・サイズ注2 (カウンタ)	
0EH	変化点テーブル・セグメント (DS1)	
10H	復号化ライン変化点テーブル・オフセット	
12H	1ライン分の総画素ビット数	
14H	復号化変換テーブル・セグメント	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	0クリア注3/ワーク・エリア	0クリア必要パラメータ
1EH	0クリア注3/ワーク・エリア	

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. バイト単位。0以外の偶数値を設定してください。

3. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

【記　述　例】　MHDEC

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MHDEC	な　し	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0

保守／廃止

MHENC

MH符号化を実行
MH Encode

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**MHENC**

【オペレーション】　変化点テーブルからMH符号を生成

【オ ペ ラ ン ド】

ニモニック	オペランド
MHENC	な し

【フ ラ グ】

Z	CY	送信バッファ残り	データ残り	処理
0	0	あ り	CHの値	終了
0	1	—	—	—
1	0	な し	な し	終了
1	1	な し	あ り	中断

【説　　明】　1ライン分の画素データの変化点情報を入力し，所定の符号化変換テーブルによりMH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出する命令です。データの先頭はLSBに入ります。

各色（白または黒）の変化点情報（変化点テーブルの1ワード）の符号化処理ごとに割り込み要求のサンプリングを行います。

MHENC命令において，連続した色（白または黒）で符号化できる最大画素数は合計5183画素です。

〈入力パラメータ〉

AL ：レジスタ・バンク番号指定
BW ：処置中半端符号データ
CH ：処理中半端符号ビット数
BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）
DS1：送信バッファ・セグメント
IY ：送信バッファ・オフセット

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス[注1]
02H	ワーク・エリア		
04H	ワーク・エリア		
06H	ワーク・エリア		
08H	変化点テーブル・セグメント	(DS0)	入力データ処理情報
0AH	符号化ライン変化点テーブル・オフセット	（ポインタ）	
0CH	ワーク・エリア		
0EH	符号化変換テーブル・セグメント	(DS1)	出力データ処理情報
10H	ワーク・エリア		
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	0クリア[注2]/ワーク・エリア		0クリア必要パラメータ
1EH	0クリア[注2]/ワーク・エリア		

注1. DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

【記　述　例】　MHENC

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MHENC	な　し	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	1

保守／廃止

MOV

データ転送
Move

【命 令 形 式】　① **MOV dst, src**
② **MOV dst1, dst2, src**

【オペランド，オペレーション】

命令形式①

ニモニック	オペランド(dst, src)	オペレーション
MOV	reg, reg'	dst←src
	mem, reg	
	reg, mem	
	mem, imm	
	reg, imm	
	acc, dmem	[W=0のとき] AL← (dmem) [W=1のとき] AH← (dmem+1), AL← (dmem)
	dmem, acc	[W=0のとき] (dmem) ←AL [W=1のとき] (dmem+1)←AH, (dmem)←AL
	sreg, reg16	dst←src
	xsreg, reg16[注] VPC, reg16	
	sreg, mem16	
	xsreg, mem16[注] VPC, mem16	
	reg16, sreg	
	reg16, xsreg[注] reg16, VPC	
	mem16, sreg	
	mem16, xsreg[注] mem16, VPC	
	AH, PSW	AH←S, Z, ×, AC, ×, P, ×, CY
	PSW, AH	S, Z, ×, AC, ×, P, ×, CY←AH

保守／廃止

命令形式②

ニモニック	オペランド（dst1, dst2, src）	オペレーション
MOV	DS0, reg16, mem32	reg16← (mem32)
	DS1, reg16, mem32	dst1← (mem32＋2)
	DS2, reg16, mem32[注]	
	DS3, reg16, mem32[注]	

注　V20, V30とV25, V35に対して新しく追加した命令

【有効オーバライド・プリフィクス】

命令形式①

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

命令形式②

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　オペランドがPSW, AHの場合　左記以外

AC	CY	V	P	S	Z	$\overline{\text{IBRK}}$
×	×		×	×	×	×

AC	CY	V	P	S	Z

【説　　明】　命令形式①：第1オペランドで指定されるデスティネーション・オペランド（dst）に，第2オペランドで指定されるソース・オペランド（src）の内容を転送します。

オペランドがAH, PSWの場合は，S, Z, AC, P, CYの各フラグをAHレジスタに転送します。AHレジスタのビット1，ビット3，ビット5は不定となります。

オペランドがPSW, AHの場合は，AHレジスタのビット2，ビット4，

ビット6，ビット7をそれぞれPSWのS，Z，AC，P，CY，$\overline{\text{IBRK}}$の各フラグに転送します。

注意　dst＝sregまたはsrc＝sregの場合，次の命令との間にハードウエア割り込み(マスカブル割り込み，ノンマスカブル割り込み)要求およびシングルステップ・ブレークは受け付けられません。

命令形式②：ソース・オペランド（src）でアドレスされる32ビット・メモリの下位16ビット（32ビット・ポインタ変数のオフセット・ワード）を，デスティネーション・オペランド2（dst2）の16ビット・レジスタに，上位16ビット（セグメント・ワード）をデスティネーション・オペランド1（dst1）のセグメント・レジスタ（DS0レジスタまたはDS1レジスタ），または拡張セグメント・レジスタ（DS2レジスタまたはDS3レジスタ）に転送します。

【記　述　例】　メモリ0：50Hに55Hを書く動作

```
MOV AW, 0
MOV DS1, AW
MOV IY, 50H
MOV DL, 55H
MOV DS1 : [IY], DL
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
MOV	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	mem, imm	3-6
	reg, imm	2, 3
	acc, dmem	3
	dmem, acc	
	sreg, reg16	2
	xsreg, reg16	2
	VPC, reg16	
	sreg, mem16	2-4
	xsreg, mem16	2-4
	VPC, mem16	
	reg16, sreg	2
	reg16, xsreg	2
	reg16, VPC	
	mem16, sreg	2-4
	mem16, xsreg	2-4
	mem16, VPC	
	DS0, reg16, mem32	2-4
	DS1, reg16, mem32	
	DS2, reg16, mem32	3-5
	DS3, reg16, mem32	
	AH, PSW	1
	PSW, AH	

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOV	reg, reg'	1	0	0	0	1	0	1	W	1	1	reg			reg'		
	mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	mem, imm	1	1	0	0	0	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	reg, imm	1	0	1	1	W	reg			imm8 or imm16-low							
		imm16-high								—							
	acc, dmem	1	0	1	0	0	0	0	W	addr-low							
		addr-high								—							
	dmem, acc	1	0	1	0	0	0	1	W	addr-low							
		addr-high								—							
	sreg, reg16	1	0	0	0	1	1	1	0	1	1	0	sreg		reg		
	xsreg, reg16 VPC, reg16	1	0	0	0	1	1	1	0	1	1	1	xsreg		reg		
	sreg, mem16	1	0	0	0	1	1	1	0	mod		0	sreg		mem		
		(disp-low)								(disp-high)							
	xsreg, mem16 VPC, mem16	1	0	0	0	1	1	1	0	mod		1	xsreg		mem		
		(disp-low)								(disp-high)							
	reg16, sreg	1	0	0	0	1	1	0	0	1	1	0	sreg		reg		
	reg16, xsreg reg16, VPC	1	0	0	0	1	1	0	0	1	1	1	xsreg		reg		
	mem16, sreg	1	0	0	0	1	1	0	0	mod		0	sreg		mem		
		(disp-low)								(disp-high)							
	mem16, xsreg mem16, VPC	1	0	0	0	1	1	0	0	mod		1	xsreg		mem		
		(disp-low)								(disp-high)							
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem		
		(disp-low)								(disp-high)							
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem		
		(disp-low)								(disp-high)							
	DS2, reg16, mem32	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	0
		mod		reg			mem			(disp-low)							
		(disp-high)								—							
	DS3, reg16, mem32	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0
		mod		reg			mem			(disp-low)							
		(disp-high)								—							
	AH, PSW	1	0	0	1	1	1	1	1	—							
	PSW, AH	1	0	0	1	1	1	1	0	—							

保守／廃止

MOVBK
MOVBKB
MOVBKW

ブロック転送
Move Block
Move Block Byte
Move Block Word

【命　令　形　式】　(repeat) **MOVBK** dst-block, src-block
(repeat) **MOVBKB**
(repeat) **MOVBKW**

【オペレーション】　[W＝0のとき] (IY) ← (IX)
DIR＝0：IX←IX＋1, IY←IY＋1
DIR＝1：IX←IX－1, IY←IY－1
[W＝1のとき] (IY＋1, IY) ← (IX＋1, IX)
DIR＝0：IX←IX＋2, IY←IY＋2
DIR＝1：IX←IX－2, IY←IY－2

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
MOVBK	dst-block, src-block
MOVBKB	な　し
MOVBKW	

【有効オーバライド・プリフィクス】(MOVBKだけ)

	src	dst
ディフォールト時	DS0：	DS1：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：,	DS1：, DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z

保守／廃止

【説　　明】　IXレジスタでアドレスされるブロックを，IYレジスタでアドレスされるブロックに，バイトまたはワード・データの繰り返しで転送します。

IXレジスタ，IYレジスタは，次のバイト／ワード転送のために1バイト／ワード転送ごとに自動的にインクリメント（+1/+2），またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。

バイトかワードかの指定は，MOVBK命令を用いる場合はオペランドの属性によって行われ，MOVBKB命令またはMOVBKW命令を用いる場合は，それぞれバイト，ワード・タイプに直接指定されます。

デスティネーション・ブロックのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。

一方，ソース・ブロックは，ディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。

【記　述　例】　MOVBK BYTE_VAR1, BYTE_VAR2
MOVBK WORD_VAR1, WORD_VAR2

【バ　イ　ト　数】　1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
MOVBK	dst-block, src-block	1	0	1	0	0	1	0	W
MOVBKB	な　し								
MOVBKW									

保守／廃止

MOVSPA

レジスタ・バンク切り替え後のデータ転送

Move Stack Pointer After Context Switch

（V20，V30に対する追加命令）

【命 令 形 式】　**MOVSPA**

【オペレーション】　新レジスタ・バンクのSS，SP←旧レジスタ・バンクのSS，SP

【オ ペ ラ ン ド】

ニモニック	オペランド
MOVSPA	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　レジスタ・バンクの切り替わる前のSSの値を現在（切り替え後）のレジスタ・バンクのSSに転送します。同じようにSPの値も転送します。
BRKCS命令または割り込み要求でレジスタ・バンクを切り替えた場合，切り替えの前後でスタックを連続させるときに使用します。

【記　述　例】　MOVSPA

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOVSPA	な　し	0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1

保守／廃止

MOVSPB

レジスタ・バンク切り替え先のデータ転送

Move Stack Pointer Before Task Switch

（V20，V30に対する追加命令V25，V35に対する拡張命令）

【命 令 形 式】　**MOVSPB reg16**

【オペレーション】　reg16で示されるレジスタ・バンクのSS，SP←現在のレジスタ・バンクのSS，SP

【オ ペ ラ ン ド】

ニモニック	オペランド
MOVSPB	reg16

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　現在のレジスタ・バンク内のSSの値を，オペランドに記述した16ビット汎用レジスタの内容の下位４ビットで示される切り替え先のレジスタ・バンク内のSSに転送します。同じようにSPの値も転送します。

TSKSW命令でレジスタ・バンクを切り替える場合，切り替えの前後でスタックを連続させるときに使用します。

【記　述　例】　MOVSPB AW

【バ イ ト 数】　3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOVSPB	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1
		1	1	1	1	1	reg			—							

保守／廃止

MRDEC

MR復号化を実行
MR Decode

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**MRDEC**

【オペレーション】　MR符号から変化点テーブルを生成

【オ ペ ラ ン ド】

ニモニック	オペランド
MRDEC	な し

【フ ラ グ】

Z	CY	受信バッファ残り	終了状態	AHステータス
0	0	あ り	正常終了	変化しない
0	1	あ り	終了エラー	**表2-10参照**
1	0	な し	正常終了	変化しない
1	1	な し	終了エラー	**表2-10参照**

AH内の数値によって終了状態をチェックできます。数値と終了状態を次に示します。

表2-10 MRDEC命令の終了状態

AH	終了状態
00H	中断
01H	先頭にEOLを検出
02H	FILL付きのEOLを先頭に検出
03H	1ライン分の画素数に満たないデータのあと，EOLを検出
04H	1ライン分の画素数以上に復号するデータが存在
05H	異常なコードを検出

備考 異常コードを検出して終了した場合，複号化ライン変化点テーブルのオフセットは，最後に正常出力された変化点テーブル・データの次のワード領域のアドレスを指しています。

【説　　明】　指定されたメモリ（受信バッファ）上の復号ラインの符号データと参照ラインの変化点情報を入力し，復号化変換テーブルによりMR復号化を行い，生成された復号化

保守／廃止

ラインの変化点情報を指定されたメモリ（変化点テーブル）上に送出する命令です。
データの先頭と処理中の半端符号はLSBから入ります。
変化点情報を生成することにより，１ライン分の画素データ・ビット数に達したことを認識し命令を終了とします。
MR復号化の符号データから変化点テーブルを作成する処理において，次のような場合が考えられます。

① 指定された受信バッファ内に１ライン分の符号データがあり，変化点情報への変換を終了する場合（正常終了）。
② 指定された受信バッファ内の符号が１ライン分に満たなくて，変化点情報への変換が途中で終了する場合（中断）。
③ １番先頭にEOLを検出する場合。
④ FILL付きのEOLを先頭に検出する場合。
⑤ １ライン分の画素数に満たないデータのあと，EOLを検出する場合。
⑥ １ライン分の画素数以上に復号化するデータがある場合。
⑦ 異常なコードを検出する場合。

正常終了のときだけ，変化点テーブルに“FFFFH”（１ライン分の終了コード）を出力します。
各モード（パス・モード，垂直モード，水平モード）単位の復号化処理ごとに，割り込み要求のサンプリングを行います。
MRDEC命令において，MR復号化できる最大画素数は合計32767画素です。
MRDEC命令実行中に“0”の連続する異常符号を検出すると，次に“1”を検出するまで割り込み要求が保留されます。この場合，受信バッファ・サイズを小さく設定してMRDEC命令が一定の時間で中断されるようにすることにより，長時間にわたる割り込み要求の保留を回避することができます。MRDEC命令の実行時間（最長となる場合）の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません）。

- 16ビット・バス幅のとき：211＋6T＋（59＋T）N/16（クロック）
- ８ビット・バス幅のとき：227＋12T＋（37＋2T）N/8（クロック）

N：“0”の連続するビット数
T：変化点テーブルのウエイト数

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	（変化点テーブル・セグメント）	(DS3)	
04H	処理中半端符号データ		
06H	処理中半端符号ビット数		
08H	受信バッファ・セグメント	(DS0)	
0AH	受信バッファ・オフセット	(ポインタ)	
0CH	受信バッファ・サイズ注2	(カウンタ)	
0EH	変化点テーブル・セグメント	(DS1)	
10H	復号化ライン変化点テーブル・オフセット		
12H	参照ライン変化点テーブル・オフセット		
14H	復号化変換テーブル・セグメント		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	0クリア注3/ワーク・エリア		0クリア必要パラメータ
1EH	0クリア注3/ワーク・エリア		

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. バイト単位。0以外の偶数値を設定してください。

3. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

【記　述　例】　MRDEC

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MRDEC	な　し	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	1

保守／廃止

MRENC

MR符号化を実行
MR Encode

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**MRENC**

【オペレーション】　変化点テーブルからMR符号を生成

【オ ペ ラ ン ド】

ニモニック	オペランド
MRENC	な し

【フ ラ グ】

Z	CY	送信バッファ残り	データ残り	処理
0	0	あ り	CHの値	終了
0	1	—	—	—
1	0	な し	な し	終了
1	1	な し	あ り	中断

【説　　明】　参照ラインの変化点情報と符号化ラインの変化点情報とを入力し，1ライン分のMR符号化を行い，その符号を指定されたメモリ（送信バッファ）上に送出する命令です。データの先頭はLSBに入ります。

MR符号化の1モード単位（パス・モード，垂直モード，水平モードのいずれか）の符号化処理ごとに割り込み要求のサンプリングを行います。

MRENC命令の水平モードのMH符号化において，連続した色（白または黒）で符号化できる最大画素数は合計5183画素です。

〈入力パラメータ〉

AL ：レジスタ・バンク番号指定

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）　(DS2)	拡張セグメント・オーバライド・プリフィクス[注1]
02H	ワーク・エリア	
04H	ワーク・エリア	
06H	ワーク・エリア	
08H	変化点テーブル・セグメント　(DS0)	
0AH	符号化ライン変化点テーブル・オフセット　(ポインタ)	
0CH	参照ライン変化点テーブル・オフセット（ポインタ）	
0EH	符号化変換テーブル・セグメント　(DS1)	
10H	ワーク・エリア	
12H	ワーク・エリア	
14H	ワーク・エリア	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	0クリア[注2]/ワーク・エリア	0クリア必要パラメータ
1EH	0クリア[注2]/ワーク・エリア	

注1. DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

【記　述　例】　MRENC

【バ　イ　ト　数】　2

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MRENC	な　し	0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1

保守／廃止

MUL

符号付き乗算
Multiply Signed

【命　令　形　式】　① **MUL src**

② **MUL dst, src**

③ **MUL dst, src1, src2**

【オペランド，オペレーション】

命令形式①

ニモニック	オペランド	オペレーション
MUL	reg8	AW←AL×src AH＝ALのサイン拡張：CY←0, V←0 AH≠ALのサイン拡張：CY←1, V←1
	mem8	
	reg16	DW, AW←AW×src DW＝AWのサイン拡張：CY←0, V←0 DW≠AWのサイン拡張：CY←1, V←1
	mem16	

命令形式②

ニモニック	オペランド	オペレーション
MUL	reg16, imm8	dst←dst×src 積≦16ビット：CY←0, V←0 積＞16ビット：CY←1, V←1
	reg16, imm16	

命令形式③

ニモニック	オペランド	オペレーション
MUL	reg16, reg16', imm8	dst←src1×src2 積≦16ビット：CY←0, V←0 積＞16ビット：CY←1, V←1
	reg16, mem16, imm8	
	reg16, reg16', imm16	
	reg16, mem16, imm16	

【有効オーバライド・プリフィクス】

命令形式①，③

	src/src1
ディフォールト時	DS0：
プリフィクス付加時	DS0：，DS1：， PS：，SS：， DS2：，DS3：， IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	×	×	U	U	U

【説　　明】

命令形式①：○src＝reg8またはsrc＝mem8のとき

ALレジスタの値とソース・オペランド（src）の符号付き乗算を行い，倍長結果をAWレジスタに格納します。このとき結果の上位半分（AHレジスタ）が下位半分（ALレジスタ）の符号拡張でなければCYフラグとVフラグがセット(1)されます。AHレジスタは拡張レジスタです。

○src＝reg16またはsrc＝mem16のとき

AWレジスタの値とソース・オペランド（src）の符号付き乗算を行い，倍長結果をAWレジスタとDWレジスタに格納します。このとき結果の上位半分（DWレジスタ）が下位半分（AWレジスタ）の符号拡張でなければCYフラグとVフラグがセット(1)されます。DWレジスタは拡張レジスタです。

命令形式②：デスティネーション・オペランド（dst）とソース・オペランド（src）の符号付き乗算を行い，結果をデスティネーション・オペランド(dst)に格納します。

命令形式③：第1ソース・オペランド（src1）と第2ソース・オペランド（src2）の符号付き乗算を行い，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　AWレジスタの値とメモリ0：50Hの内容（ワード・データ）を乗算する。

```
MOV BW, 0
MOV DS0, BW
MOV IX, 50H
MUL WORD PTR [IX]
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
MUL	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4
	reg16, imm8	3
	reg16, imm16	4
	reg16, reg16', imm8	3
	reg16, mem16, imm8	3-5
	reg16, reg16', imm16	4
	reg16, mem16, imm16	4-6

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1	reg		
	mem8	1	1	1	1	0	1	1	0	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	1	reg		
	mem16	1	1	1	1	0	1	1	1	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg16, imm8	0	1	1	0	1	0	1	1	1	1	reg			reg′		
		imm8								—							
	reg16, imm16	0	1	1	0	1	0	0	1	1	1	reg			reg′		
		imm16-low								imm16-high							
	reg16, reg16′, imm8	0	1	1	0	1	0	1	1	1	1	reg			reg′		
		imm8								—							
	reg16, mem16, imm8	0	1	1	0	1	0	1	1	mod		reg			mem		
		(disp-low)								(disp-high)							
		imm8								—							
	reg16, reg16′, imm16	0	1	1	0	1	0	0	1	1	1	reg			reg′		
		imm16-low								imm16-high							
	reg16, mem16, imm16	0	1	1	0	1	0	0	1	mod		reg			mem		
		(disp-low)								(disp-high)							
		imm16-low								imm16-high							

保守／廃止

MULU

符号なし乗算
Multiply Unsigned

【命 令 形 式】　**MULU src**

【オペランド，オペレーション】

ニモニック	オペランド（src）	オペレーション
MULU	reg8	AW←AL×src
	mem8	
	reg16	DW，AW←AW×src DW=0：CY←0，V←0 DW=0：CY←1，V←1
	mem16	

【有効オーバライド・プリフィクス】

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：，DS1：， PS：，SS：， DS2：，IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
U	×	×	U	U	U

【説　　明】

○src＝reg8またはsrc＝mem8のとき

ALレジスタの値とソース・オペランド（src）の符号なし乗算を行い，倍長結果をAWレジスタに格納します。このとき結果の上位半分（AHレジスタ）がゼロでなければCYフラグとVフラグがセット(1)されます。AHレジスタは拡張レジスタです。

○src＝reg16またはsrc＝mem16のとき

AWレジスタの値とソース・オペランド（src）の符号なし乗算を行い，倍長結果をAWレジスタとDWレジスタに格納します。このとき結果の上位半分（DWレジスタ）がゼロでなければCYフラグとVフラグがセット(1)されます。DWレジスタは拡張レジスタです。

保守／廃止

【記　述　例】　ALレジスタの内容とCLレジスタの内容を乗算する。

MULU CL

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
MULU	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MULU	reg8	1	1	1	1	0	1	1	0	1	1	1	0	0	reg		
	mem8	1	1	1	1	0	1	1	0	mod			1	0	0	mem	
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	0	reg		
	mem16	1	1	1	1	0	1	1	1	mod			1	0	0	mem	
		(disp-low)								(disp-high)							

保守／廃止

NEG

2の補数演算

Negate

【命　令　形　式】　**NEG dst**

【オペレーション】　dst←$\overline{dst}$+1

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst）
NEG	reg
	mem

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：,DS1：, PS：,SS：, DS2：,DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	注	×	×	×	×

注　CY＝1。ただし実行前のdstが0のときはCY＝0。

【説　　明】　デスティネーション・オペランド（dst）の内容の2の補数をとります。

【記　述　例】
- NEG DL
- NEG CW
- NEG IX
- NEG BP

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
NEG	reg	2
	mem	2-4

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg		
	mem	1	1	1	1	0	1	1	W	mod			0	1	1	mem	
		(disp-low)								(disp-high)							

保守／廃止

NOP

ノー・オペレーション
No Operation

【命 令 形 式】　**NOP**

【オペレーション】　ノー・オペレーション

【オ ペ ラ ン ド】

ニモニック	オペランド
NOP	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　何も処理せず，4クロック費やします。

【記　述　例】　NOP

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
NOP	な　し	1	0	0	1	0	0	0	0

保守／廃止

NOT

論理否定
Not

【命 令 形 式】 **NOT dst**

【オペレーション】 dst←$\overline{dst}$

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
NOT	reg
	mem

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 デスティネーション・オペランド（dst）で指定されるビットを反転し（論理否定），結果をデスティネーション・オペランド（dst）へ格納します。

【記 述 例】
- NOT AL
- NOT CW
- NOT IX

【バ イ ト 数】

ニモニック	オペランド	バイト数
NOT	reg	2
	mem	2-4

保守／廃止

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg		
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							

保守／廃止

NOT1

ビットの反転
Not Bit

【命　令　形　式】　① **NOT1 dst, src**
② **NOT1 dst**

【オペレーション】　命令形式①：dstのビットn（nはsrcで指定）←$\overline{\text{dstのビットn（nはsrcで指定）}}$
命令形式②：dst←$\overline{\text{dst}}$

【オ　ペ　ラ　ン　ド】　命令形式①

ニモニック	オペランド（dst, src）
NOT1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
NOT1	CY

【有効オーバライド・プリフィクス】

命令形式①

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　命令形式①

AC	CY	V	P	S	Z

命令形式②

AC	CY	V	P	S	Z
	×				

保守／廃止

【説　　明】　命令形式①：第１オペランドで指定されるデスティネーション・オペランド（dst）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）の論理否定をとり，結果をデスティネーション・オペランド（dst）に格納します。

オペランドがreg8, CLまたはmem8, CLのとき，CLの値は下位３ビット（0-7）のみ有効です。

オペランドがreg16, CLまたはmem16, CLのとき，CLの値は下位４ビット（0-15）のみ有効です。

オペランドがreg8, imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。

オペランドがmem8, imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。

オペランドがreg16, imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。

オペランドがmem16, imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

命令形式②：CYフラグの内容の論理否定をとり，CYフラグに格納します。

【記　述　例】

```
IN    AL, 0
NOT1 AL, 7
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
NOT1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NOT1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	0	1	0	1	—							

保守／廃止

OR

論理和

Or

【命　令　形　式】　**OR dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
OR	reg, reg'	dst←dst∨src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL←AL∨imm8 [W＝1のとき] AW←AW∨imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	0	0	×	×	×

【説　　明】　第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）の論理和（OR）をとり，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　OR AW, WORD PTR [IX]

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
OR	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
OR	reg, reg'	0	0	0	0	1	0	1	W	1	1	reg				reg'	
	mem, reg	0	0	0	0	1	0	0	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	0	1	0	1	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	0	0	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	0	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注　アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	W	1	1	0	0	1	reg		
imm8								—							

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

OUT

I/Oデバイスへのデータ出力

Output

【命　令　形　式】　**OUT dst, src**

【オペランド，オペレーション】

○$\overline{\text{IBRK}}$＝1のとき

ニモニック	オペランド（dst, src）	オペレーション
OUT	imm8, acc	[W＝0のとき] (imm8) ←AL [W＝1のとき] (imm8＋1) ←AH, (imm8) ←AL
	DW, acc	[W＝0のとき] (DW) ←AL [W＝1のとき] (DW＋1) ←AH, (DW) ←AL

○$\overline{\text{IBRK}}$＝0のとき

ニモニック	オペランド（dst, src）	オペレーション
OUT	imm8, acc	(SP－1, SP－2) ←PSW (SP－3, SP－4) ←PS (SP－5, SP－6) ←PC－x[注]
	DW, acc	IE←0, BRK←0, $\overline{\text{IBRK}}$←1, PC← (019H, 018H) PS← (01BH, 01AH)

注　xは命令のバイト数＋プリフィクスの数

【フ　ラ　グ】　○$\overline{\text{IBRK}}$＝1のとき

AC	CY	V	P	S	Z

○$\overline{\text{IBRK}}$＝0のとき

AC	CY	V	P	S	Z	IE	BRK	$\overline{\text{IBRK}}$
						0	0	1

保守／廃止

【説　　明】　○$\overline{IBRK}$＝1のとき

デスティネーション・オペランド（dst）で指定されるI/Oデバイスに，ソース・オペランド（src）で指定されるアキュームレータ（AL/AW）の内容を転送します。

○$\overline{IBRK}$＝0のとき

PSW，PS，PC－xをスタックに退避し，IEフラグとBRKフラグをリセット(0)し，$\overline{IBRK}$フラグをセットします。

次に割り込みベクタ・テーブルのベクタ６の下位２バイトをPCに，上位２バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするための機能です。

【記　述　例】　ポート・アドレス0D8HにALレジスタの内容を転送する。

```
MOV DW, 0D8H
OUT DW, AL
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
OUT	imm8, acc	2
	DW, acc	1

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
OUT	imm8, acc	1	1	1	0	0	1	1	W	imm8							
	DW, acc	1	1	1	0	1	1	1	W	—							

保守／廃止

OUTM

メモリ-I/Oのブロック転送

Output Multiple

【命　令　形　式】　(repeat) OUTM DW, src-block

【オペレーション】

○$\overline{IBRK}$＝1のとき

[W＝0のとき]　(DW) ← (IX)

DIR＝0：IX←IX＋1

DIR＝1：IX←IX－1

[W＝1のとき]　(DW＋1, DW) ← (IX＋1, IX)

DIR＝0：IX←IX＋2

DIR＝1：IX←IX－2

○$\overline{IBRK}$＝0のとき

(SP－1, SP－2) ←PSW

(SP－3, SP－4) ←PS

(SP－5, SP－6) ←PC－x[注]

IE←0, BRK←0, $\overline{IBRK}$←1

PC← (019H, 018H)

PS← (01BH, 01AH)

注　xは命令のバイト数＋プリフィクスの数

【有効オーバライド・プリフィクス】

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, IRAM：

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
OUTM	DW, src-block

保守／廃止

【フ ラ グ】 ○$\overline{IBRK}$＝1のとき

AC	CY	V	P	S	Z

○$\overline{IBRK}$＝0のとき

AC	CY	V	P	S	Z	IE	BRK	$\overline{IBRK}$
						0	0	1

【説 明】 ○$\overline{IBRK}$＝1のとき

IXレジスタでアドレスされるメモリの内容を，DWレジスタでアドレスされるI/Oデバイスのレジスタに転送します。繰り返し転送回数は，ペアで使用されるリピート・プリフィクスのREP命令によって制御されます。繰り返し転送の際，DWレジスタの内容（I/Oデバイスのアドレス）は固定ですが，IXレジスタは次のバイト／ワード転送のために1バイト／ワード・データが転送されるごとに自動的にインクリメント（+1/+2）またはデクリメント（−1/−2）されます。ブロックの方向は，DIRフラグの内容によって決定されます。

バイトかワードかの指定は，オペランド属性によって行われます。

OUTM命令は，リピート・プリフィクスのREP命令とともに使用されます。

ソース・ブロックは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にローケートできます。

○$\overline{IBRK}$＝0のとき

PSW，PS，PC−xをスタックに退避し，IEフラグとBRKフラグをリセット(0)し，$\overline{IBRK}$フラグをセットします。

次に割り込みベクタ・テーブルのベクタ6の下位2バイトをPCに，上位2バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするための機能です。

【記 述 例】 ●メモリ0：50Hの内容をポート・アドレス0D8H（バイト・データ）に転送する。

```
MOV   AW, 0
MOV   DS0, AW
MOV   IX, 50H
MOV   DW, 0D8H
OUTM  DW, DS0 : WORD PTR [IX]
```

●メモリ0：0H～0FFHの内容をポート・アドレス0D8H（バイト・データ）に転送する。

```
MOV  AW, 0
MOV  DS0, AW
MOV  IX, 0H
MOV  DW, 0D8H
MOV  CW, 0FFH
REP  OUTM DW, DS0 : PTR [IX]
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
OUTM	DW, src-block	0	1	1	0	1	1	1	W

保守／廃止

POLL

浮動小数点演算用コプロセッサ待ち

Poll and wait

【命 令 形 式】 **POLL**

【オペレーション】 POLL and wait

【オ ペ ラ ン ド】

ニモニック	オペランド
POLL	な し

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 $\overline{\text{POLL}}$端子がアクティブ(ロウ・レベル)になるまでCPUをウエイト状態にします。

注意 この命令の直前にBUSLOCK命令を置くことはできません。

【記 述 例】 POLL

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
POLL	な し	1	0	0	1	1	0	1	1

保守／廃止

POP

スタックからの復帰

Pop

【命 令 形 式】 **POP dst**

【オペランド，オペレーション】

ニモニック	オペランド (dst)	オペレーション
POP	mem16	SP←SP+2 (mem16) ← (SP−1, SP−2)
	reg16	SP←SP+2 dst← (SP−1, SP−2)
	sreg	
	xsreg[注] VPC	
	PSW	
	R	IY← (SP+1, SP) IX← (SP+3, SP+2) BP← (SP+5, SP+4) BW← (SP+9, SP+8) DW← (SP+11, SP+10) CW← (SP+13, SP+12) AW← (SP+15, SP+14) SP←SP+16

注 V20，V30とV25，V35に対して新しく追加した命令

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：，DS1：， PS：，SS：， DS2：，DS3：， IRAM：

【フ ラ グ】 ○dst＝PSWの場合

AC	CY	V	P	S	Z	DIR	IE	BRK	$\overline{\text{IBRK}}$
R	R	R	R	R	R	R	R	R	R

○dst＝PSW以外の場合

AC	CY	V	P	S	Z

【説　　明】　デスティネーション・オペランド（dst）の内容にスタックの内容を転送します（ただし，dst＝sregの場合，PSへは転送されません）。

注意　dst＝sregの場合，次の命令との間にハードウエア割り込み(マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】
- POP　AW
- POP　BW
- POP　IY
- POP　SP

MOV BP, SP

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
POP	mem16	2-4
	reg16	1
	sreg	
	xsreg	2
	VPC	
	PSW	1
	R	

保守／廃止

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	1	1	reg			—							
	sreg	0	0	0	sreg		1	1	1	—							
	xsreg VPC	0	0	0	0	1	1	1	1	0	1	1	xsreg		1	1	1
	PSW	1	0	0	1	1	1	0	1	—							
	R	0	1	1	0	0	0	0	1	—							

保守／廃止

PREPARE

スタック・フレームの生成
Prepare New Stack Frame

【命 令 形 式】　**PREPARE imm16, imm8**

【オペレーション】

(SP－1, SP－2)←BP
SP←SP－2
temp←SPを行ったあと
imm8＞0のとき次の動作を,
(SP－1, SP－2) ← (BP－1, BP－2)
SP←SP－2
BP←BP－2
"imm8－1" 回繰り返したあとに,
＊1
(SP－1, SP－2) ←temp
SP←SP－2
＊2
を実行します。
このあと次の処理を行います。
BP←temp
SP←SP－imm16
imm8＝1のとき, ＊1の繰り返し動作を行いません。
imm8＝0のとき, ＊1および＊2の動作を行いません。

【オ ペ ラ ン ド】

ニモニック	オペランド
PREPARE	imm16, imm8

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　この命令は, ブロック構造の高級言語 (たとえばPascal, Adaなど) において必要な "スタック・フレーム" を生成するために用いられます。スタック・フレームには, そのプロシージャから参照してもよい変数を含むフレームを指すポインタ群とローカル変数の領域が含まれます。
この命令は, ローカル変数の領域確保と, グローバル変数への参照を可能とするため, フレーム・ポインタのコピーを行います。第1オペランドに記述された16ビッ

ト・イミーディエト・データでローカル変数用として確保する領域の大きさ（バイト単位）が指定され，第２オペランドに記述された８ビット・イミーディエト・データでプロシージャ・ブロックの深さ（この深さをレキシカル・レベルといいます）が示されます。

この命令によって生成されるフレームのベース・アドレスは，BPにセットされます。

まずBPをスタックへ退避します。これは，プロシージャの終了時点で，コールした側のプロシージャのBPを復帰するためです。次に，コールされたプロシージャから参照可能な範囲のフレーム・ポインタ（退避されたBP）をスタックに積みます。参照可能な範囲は，そのプロシージャのレキシカル・レベルより１減じた値となります。

レキシカル・レベルが１以上の場合には，自分自身のフレーム・ポインタもスタックへ積みます。これは，このプロシージャからコールされたプロシージャにおいてフレーム・ポインタのコピーを行う場合に，コールしたプロシージャのフレーム・ポインタもコピーするためです。

次にBPに新しいフレーム・ポインタの値をセットし，そのプロシージャで使用されるローカル変数の領域をスタックに確保します。つまり，SPをローカル変数分だけ減らします。

【記　述　例】

```
MOV     SP, 60H
MOV     BP, SP
CALL    CHK
PREPARE 0006, 04
MOV     AW, [BP+0FAH]
ADD     AW, [BP+0F8A]
MOV     [BP+0FCH], AW
```

【バ イ ト 数】　4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0	imm16-low							
		imm16-high								imm8							

保守／廃止

PUSH

スタックへの退避

Push

【命　令　形　式】　**PUSH src**

【オペランド，オペレーション】

ニモニック	オペランド（src）	オペレーション
PUSH	mem16	SP←SP－2 (SP＋1,SP)←(mem16＋1，mem16)
	reg16	SP←SP－2 (SP＋1，SP)←src
	sreg	
	xsreg[注] VPC	
	PSW	
	R	temp←SP (SP－1，SP－2)←AW (SP－3，SP－4)←CW (SP－5，SP－6)←DW (SP－7，SP－8)←BW (SP－9，SP－10)←temp (SP－11，SP－12)←BP (SP－13，SP－14)←IX (SP－15，SP－16)←IY SP←SP－16
	imm8	(SP－1，SP－2)←imm8のサイン拡張 SP←SP－2
	imm16	(SP－1，SP－2)←imm16 SP←SP－2

注　V20，V30とV25，V35に対して新しく追加した命令

保守／廃止

【有効オーバライド・プリフィクス】

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 ソース・オペランド（src）の内容をスタックに退避します。
オペランドに8ビット・イミーディエト・データ（imm8）を記述した場合は，imm8はサイン拡張され，16ビット・データとしてSPでアドレスされるスタックに退避されます。

【記 述 例】
- PUSH DS0
- PUSH SS
- PUSH DS1

【バ イ ト 数】

ニモニック	オペランド	バイト数
PUSH	mem16	2-4
	reg16	1
	sreg	
	xsreg VPC	2
	PSW	1
	R	
	imm8	2
	imm16	3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	1	0	reg			—							
	sreg	0	0	0	sreg		1	1	0	—							
	xsreg VPC	0	0	0	0	1	1	1	1	0	1	1	xsreg		1	1	0
	PSW	1	0	0	1	1	1	0	0	—							
	R	0	1	1	0	0	0	0	0	—							
	imm8	0	1	1	0	1	0	1	0	imm8							
	imm16	0	1	1	0	1	0	0	0	imm16-low							
		imm16-high								—							

保守／廃止

QHOUT

キューの先頭ブロックを外す

Queue Head Out

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**QHOUT dst**

【オペレーション】　キューの先頭ブロックを外します。

【オ ペ ラ ン ド】

ニモニック	オペランド
QHOUT	imm16

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	U	U	×

【説　　明】　**4.2.1 QHOUT（Queue Head Out）命令**を参照してください。

【記 述 例】　QHOUT 20H

【バ イ ト 数】　4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
QHOUT	imm16	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0
		imm16-low								imm16-high							

保守／廃止

QOUT

キューの任意のブロックを外す

Queue Out

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】 **QOUT dst**

【オペレーション】 キューの任意のブロックを外します。

【オ ペ ラ ン ド】

ニモニック	オペランド
QOUT	imm16

【フ ラ グ】

AC	CY	V	P	S	Z
U	U	U	U	U	×

【説　　明】 **4.2.2 QOUT (Queue Out)** 命令を参照してください。

【記 述 例】 QOUT 20H

【バ イ ト 数】 4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
QOUT	imm16	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	1
		imm16-low								imm16-high							

保守／廃止

QTIN

キューにブロックをキューイングする
Queue Tail In

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】 **QTIN dst**

【オペレーション】 キューにブロックをキューイングします。

【オ ペ ラ ン ド】

ニモニック	オペランド
QTIN	imm16

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 **4.2.3 QTIN (Queue Tail In)** 命令を参照してください。

【記 述 例】 QTIN 20H

【バ イ ト 数】 4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
QTIN	imm16	0	0	0	0	1	1	1	1	0	1	1	1	0	0	1	0
		imm16-low								imm16-high							

保守／廃止

REP
REPE
REPZ

Z＝1によるリピート・プリフィクス

Repeat

Repet while Equal

Repeat while Zero

【命　令　形　式】　**REP**

REPE

REPZ

【オペレーション】　［CW≠0のとき］PS：PC＋1のバイト命令実行

CW←CW－1

Z≠1のとき：PC←PC＋2

Z＝1のとき：再実行

［CW＝0のとき］PC←PC＋2

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
REP	なし
REPE	
REPZ	

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CW≠0の間，続くバイトのブロック転送／比較／入出力命令を実行し，CWレジスタの値をデクリメント（－1）します。

REPはMOVBK，LDM，STM，OUTM，INMの各命令とともに用いられ，Zフラグの値に関係なくCW≠0の間繰り返し処理を行います。

REPZ命令およびREPE命令はCMPBK，CMPMの各命令とともに用いられ，各ブロック命令による比較の結果Z≠1になるか，またはCW＝0になるとループを抜けます。

CWレジスタのチェックは，ブロック命令の実行前，すなわちREP/REPE/REPZ命令実行直前の状態に対して行われます。したがって，最初CW＝0でREP/REPE/REPZ命令実行に入ると，続くブロック命令は１回も実行されずにその次の命令に移ります。

Zフラグのチェックは，あくまで続くブロック命令の結果に対して行われ，最初REPE/REPZ命令に入る直前の内容は関係ありません。

保守／廃止

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】
- REP MOVBKW
- REPZ CMPBKW

【バ　イ　ト　数】　1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REP	な　し	1	1	1	1	0	0	1	1
REPE									
REPZ									

保守／廃止

REPC

CY＝1 によるリピート・プリフィクス

Repeat while Carry

【命 令 形 式】　**REPC**

【オペレーション】　[CW≠0のとき] PS：PC＋1のバイト命令実行

CW←CW－1

CY≠1のとき：PC←PC＋2

CY＝1のとき：再実行

[CW＝0のとき] PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPC	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK，CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。

ブロック比較命令の結果，CY≠1になるとループを抜けます。

CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPC命令実行の直前の状態に対して行われます。したがって，最初CW＝0でREPC命令実行に入ると，続くブロック比較命令は１回も実行されずにその次の命令に移ります。

CYフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPC命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　REPC CMPBKW

【バ イ ト 数】　1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPC	な　し	0	1	1	0	0	1	0	1

保守／廃止

REPNC

CY＝0によるリピート・プリフィクス
Repeat while Not Carry

【命 令 形 式】　**REPNC**

【オペレーション】　[CW≠0のとき] PS：PC＋1のバイト命令実行
CW←CW－1
CY≠1のとき：再実行
CY＝1のとき：PC←PC＋2
[CW＝0のとき] PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPNC	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK，CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。
ブロック比較命令の結果，CY＝1になるとループを抜けます。
CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPNC命令実行の直前の状態に対して行われます。したがって，最初CW＝0でREPNC命令実行に入ると，続くブロック比較命令は1回も実行されずにその次の命令に移ります。
CYフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPNC命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　REPNC CMPMB

【バ イ ト 数】　1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPNC	な　し	0	1	1	0	0	1	0	0

保守／廃止

REPNE
REPNZ

Z=0によるリピート・プリフィクス
Repeat while Not Equal
Repeat while Not Zero

【命 令 形 式】　**REPNE**
REPNZ

【オペレーション】　[CW≠0のとき] PS：PC＋1のバイト命令実行
CW←CW－1
Z≠1のとき：再実行
Z＝1のとき：PC←PC＋2
[CW＝0のとき] PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPNE	な　し
REPNZ	

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK，CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。
ブロック比較命令の結果，Z≠0になるか，またはCW＝0になるとループを抜けます。
CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPNE/REPNZ命令実行直前に対して行われます。したがって，最初CW＝0でREPNE/REPNZ命令実行に入ると，続くブロック比較命令は1回も実行されずにその次の命令に移ります。
Zフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPNE/REPNZ命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

保守／廃止

【記　述　例】
- REPNE CMPMB
- REPNZ CMPBKW

【バ　イ　ト　数】 1

【命令語形式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPNE	な　し	1	1	1	1	0	0	1	0
REPNZ									

保守／廃止

RET

サブルーチンからの復帰
Return from Procedure

【命 令 形 式】　① **RET**
② **RET pop-value**

【オペランド，オペレーション】

○セグメント内コールからのリターンのとき

ニモニック	オペランド	オペレーション
RET	な　し	PC← (SP+1, SP) SP←SP+2
	pop-value	PC← (SP+1, SP) SP←SP+2 SP←SP+pop-value

○セグメント外コールからのリターンのとき

ニモニック	オペランド	オペレーション
RET	な　し	PC← (SP+1, SP) PS← (SP+3, SP+2) SP←SP+4
	pop-value	PC← (SP+1, SP) PS← (SP+3, SP+2) SP←SP+4 SP←SP+pop-value

【フ ラ グ】

AC	CY	V	P	S	Z

保守／廃止

【説　　明】　○セグメント内コールからのリターンのとき

PCをスタックからリストアします。オペランドにpop-valueを記述したときは16ビットのpop-valueからSPに加算されます(PCに続いて退避してあったパラメータが不要になったとき，不要なパラメータの分だけSPの値を飛ばすのに有効です)。

セグメント外コールからのRET命令との区別はアセンブラが自動的に行います。

○セグメント外コールからのリターンのとき

PCとPSをスタックからリストアします。オペランドにpop-valueを記述したときは16ビットのpop-valueがSPに加算されます(PCとPSに続いて退避してあったパラメータが不要になったとき，不要なパラメータの分だけSPの値を飛ばすのに有効です)。

セグメント内コールからのRET命令との区別はアセンブラが自動的に行います。

【記　述　例】　POP R

RET

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
RET	な　し	1
	pop-value	3

【命 令 語 形 式】　○セグメント内コールからのリターンのとき

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RET	な　し	1	1	0	0	0	0	1	1	—							
	pop-value	1	1	0	0	0	0	1	0	pop-value-low							
		pop-value-high								—							

○セグメント外コールからのリターンのとき

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RET	な　し	1	1	0	0	1	0	1	1	—							
	pop-value	1	1	0	0	1	0	1	0	pop-value-low							
		pop-value-high								—							

保守／廃止

RETI

割り込みからの復帰
Return from Interrupt

【命 令 形 式】 **RETI**

【オペレーション】
PC← (SP+1, SP)
PS← (SP+3, SP+2)
PSW← (SP+5, SP+4)
SP←SP+6

【オ ペ ラ ン ド】

ニモニック	オペランド
RETI	な し

【フ ラ グ】

AC	CY	V	P	S	Z	DIR	IE	BRK	RB0	RB1	RB2	RB3	$\overline{\text{BRK}}$
R	R	R	R	R	R	R	R	R	R	R	R	R	R

【説 明】 PC, PS, PSWにスタックの内容をリストアします。割り込み処理から戻るときに用いられます。

【記 述 例】
```
POP R
RETI
```

【バ イ ト 数】 1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
RETI	な し	1	1	0	0	1	1	1	1

保守／廃止

RETRBI

レジスタ・バンクの復帰

Return from Register Bank Switching Interrupt

（V20，V30に対する追加命令）

【命 令 形 式】　**RETRBI**

【オペレーション】　PC←現在選択されているレジスタ・バンク上のPC退避領域の値
PSW←現在選択されているレジスタ・バンク上のPSW退避領域の値

【オ ペ ラ ン ド】

ニモニック	オペランド
RETRBI	な　し

【フ　ラ　グ】

AC	CY	V	P	S	Z	RB0	RB1	RB2	RB3	DIR	IE	BRK	$\overline{IBRK}$
R	R	R	R	R	R	R	R	R	R	R	R	R	R

【説　　明】　レジスタ・バンク切り替え機能を使用した割り込みからの復帰，またはBRKCS命令からの復帰に使用します。ほかの命令では正常にこの割り込みから復帰できません。また，ほかの割り込みからの復帰にこの命令を使用することはできません。
レジスタ・バンク切り替え機能を使用した割り込みからの復帰では，この命令の実行前にFINT命令を実行する必要があります。

【記　述　例】　RETRBI

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RETRBI	な　し	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1

保守／廃止

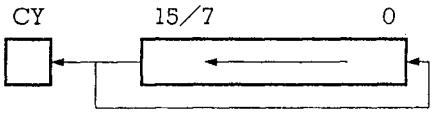
ROL

左方向のローテート

Rotate Left

【命 令 形 式】　**ROL dst, src**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
ROL	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】　src＝1の場合

AC	CY	V	P	S	Z
	×	×			

左記以外

AC	CY	V	P	S	Z
	×	U			

保守／廃止

【説　　明】　○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1ビットだけ左シフトします。dstの内容のMSB（ビット7またはビット15）のデータはLSB（ビット0）へシフトされると同時に，CYフラグへも転送されます。また，MSBが変化したときはVフラグがセット(1)され，元のままのときはリセット(0)されます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけ左シフトします。dstの内容のMSB（ビット7またはビット15）のデータはLSB（ビット0）へシフトされると同時に，CYフラグへも転送されます。

【記　述　例】

```
MOV  [IX], BL
ROL BYTE PTR [IX], 1
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROL	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

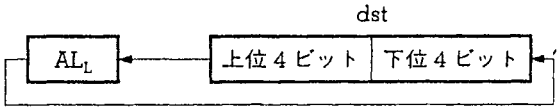
ROL4

左方向のニブル・ローテート

Rotate Left Nibble

【命　令　形　式】　**ROL4 dst**

【オペレーション】



【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst）
ROL4	reg8
	mem8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　デスティネーション・オペランド（dst）の内容を２桁のパックトBCDとして扱い，ALレジスタの下位４ビット（AL$_L$）を介して，１桁分左に回転します。

この命令の結果，ALレジスタの上位４ビットは保証されません。

【記　述　例】

●MOV AL, 24H
　ROL4 AL

●MOV AL, BYTE PTR [IX]
　ROL4 AL

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROL4	reg8	3
	mem8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0
		1	1	0	0	0	reg			—							
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							

保守／廃止

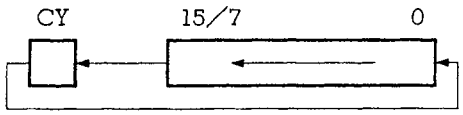
ROLC

キャリーを含む左方向のローテート

Rotate Left with Carry

【命 令 形 式】 **ROLC dst, src**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
ROLC	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】 src＝1の場合

AC	CY	V	P	S	Z
	×	×			

左記以外

AC	CY	V	P	S	Z
	×	U			

【説 明】 ○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド (dst) の内容をCYフラグを通して1ビットだけ左シフトします。dstの内容のMSB(ビット7またはビット15) のデータはCYフラグへ転送されCYフラグのデータはLSB (ビット0) へ転送されます。

また，MSBが変化したときはVフラグがセット(1)され，元のままのときはリセット(0)されます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけCYフラグを通して左シフトします。dstの内容のMSB(ビット7またはビット15)のデータはCYフラグへ転送されCYフラグのデータはLSB(ビット0)へ転送されます。

【記　述　例】

- ROLC AL, 1
- ROLC CL, 1
- ROLC DW, 1
- ROLC AW, 1

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
ROLC	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

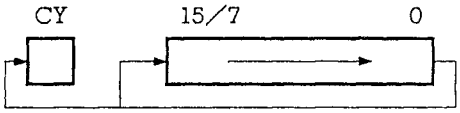
ROR

右方向のローテート

Rotate Right

【命 令 形 式】　**ROR dst, src**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
ROR	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】　src＝1の場合

AC	CY	V	P	S	Z
	×	×			

左記以外

AC	CY	V	P	S	Z
	×	U			

【説　　明】　○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1ビットだけ右シフトします。dstの内容のLSB（ビット0）のデータはMSB（ビット7またはビット15）へシフトされると同時に，CYフラグへも転送されます。また，MSBが変化したときはVフラグがセット(1)され，元のままのときはリセッ

ト(0)されます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけ右シフトします。dstの内容のLSB(ビット0)のデータはMSB(ビット7またはビット15）へシフトされると同時に，CYフラグへも転送されます。

【記　述　例】

- ROR AL, 3
- ROR CW, 6
- ROR IY, 2

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
ROR	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

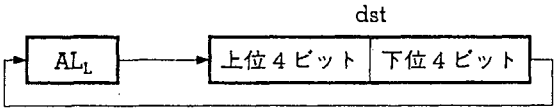
ROR4

右方向のニブル・ローテート

Rotate Right Nibble

【命 令 形 式】　**ROR4 dst**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
ROR4	reg8
	mem8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：,DS1：, PS：,SS：, DS2：,DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　デスティネーション・オペランド（dst）の内容を2桁のパックトBCDとして扱い，ALレジスタの下位4ビット（AL_L）を介して，1桁分右に回転します。

この命令の結果，ALレジスタの上位4ビットは保証されません。

【記 述 例】

●MOV AL, 24H
　ROR4 AL

●MOV AL, BYTE PTR [IX]
　ROR4 AL

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROR4	reg8	3
	mem8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0
		1	1	0	0	0	reg			—							
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							

保守／廃止

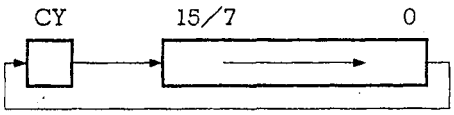
RORC

キャリーを含む右方向のローテート

Rotate Right with Carry

【命　令　形　式】　**RORC dst, src**

【オペレーション】



【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst, src）
RORC	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　src＝1の場合

AC	CY	V	P	S	Z
	×	×			

左記以外

AC	CY	V	P	S	Z
	×	U			

【説　　明】　○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容をCYフラグを通して1ビットだけ右シフトします。dstの内容のLSB（ビット0）のデータはCYフラグへ転送されCYフラグのデータはMSB（ビット7またはビット15）へ転送されます。

また，MSBが変化したときはVフラグがセット(1)され，元のままのときはリセット(0)されます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけCYフラグを通して右シフトします。dstの内容のLSB（ビット0）のデータはCYフラグへ転送されCYフラグのデータはMSB(ビット7またはビット15)へ転送されます。

【記　述　例】

- RORC AL, 1
- RORC BL, 1
- RORC CW, 1
- RORC IX, 1

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
RORC	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

RSTWDT

ウォッチドッグ・タイマ操作命令
Reset Watchdog Timer

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**RSTWDT　imm8，imm8'**

【オペレーション】

[imm8＝$\overline{\text{imm8'}}$のとき]

ウォッチドッグ・タイマのクリア

[imm8≠$\overline{\text{imm8'}}$のとき]

(SP－1，SP－2) ←PSW

(SP－3，SP－4) ←PS

(SP－5，SP－6) ←PC－x[注]

SP←SP－6

IE←0，BRK←0，

PC←（20H，21H）

PS←（22H，23H）

注　xは命令のバイト数＋プリフィクスの数

【オ ペ ラ ン ド】

ニモニック	オペランド
RSTWDT	imm8，imm8'

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　4バイト目の命令コードと3バイト目の命令コードの比較を行い，ウォッチドッグ・タイマを操作します。この命令はプログラムの暴走などで誤ってレジスタの書き換えが行われないようにするため，特殊な命令コード構成（4バイト）を採用しています。したがって，3バイト目と4バイト目のオペコードが1の補数の関係でなければ書き込みが行われません。

【記 述 例】　RSTWDT　05H，0FAH

【バ イ ト 数】　4

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RSTWDT	imm8, imm8'	0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	0

保守／廃止

SET1

ビットのセット
Set Bit

【命　令　形　式】　① **SET1 dst, src**

② **SET1 dst**

【オペレーション】　命令形式①：dstのビットn（nはsrcで指定）←1

命令形式②：dst←1

【オ　ペ　ラ　ン　ド】　命令形式①

ニモニック	オペランド（dst, src）
SET1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
SET1	CY
	DIR

【有効オーバライド・プリフィクス】

命令形式①

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　命令形式①

AC	CY	V	P	S	Z

命令形式②（dst＝CYの場合）

AC	CY	V	P	S	Z
	1				

命令形式②（dst＝DIRの場合）

AC	CY	V	P	S	Z	DIR
						1

【説　　明】　命令形式①：第１オペランドで指定されるデスティネーション・オペランド（dst）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）をセット（1）し，結果をデスティネーション・オペランド（dst）に格納します。

オペランドがreg8, CLまたはmem8, CLのとき，CLの値は下位３ビット（0-7）のみ有効です。

オペランドがreg16, CLまたはmem16, CLのとき，CLの値は下位４ビット（0-15）のみ有効です。

オペランドがreg8, imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。

オペランドがmem8, imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。

オペランドがreg16, imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。

オペランドがmem16, imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

命令形式②：dst＝CYの場合，CYフラグをセット（1）します。

dst＝DIRの場合，DIRフラグをセット（1）します。また，MOVBK, CMPBK, CMPM, LDM, STM, INM, OUTMの各命令の実行時にインデクス・レジスタ（IX, IY）をオートデクリメントするように設定します。

【記　述　例】

```
MOV CL, 4
SET1 AL, CL
OUT 0DAH, AL
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SET1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1
	DIR	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SET1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	1	0	0	1	—							
	DIR	1	1	1	1	1	1	0	1	—							

保守／廃止

SCHEOL

EOLの検出
Search EOL

（V20，V30とV25，V35に対する追加命令）

【命 令 形 式】　**SCHEOL**

【オペレーション】　MH/MR符号中の“EOL”（End of line）を検出

【オ ペ ラ ン ド】

ニモニック	オペランド
SCHEOL	なし

【フ　ラ　グ】

Z	CY	受信バッファ残り	終了状態	AHステータス
0	0	あり	正常終了	01，02
0	1	あり	終了エラー	03
1	0	なし	正常終了	01，02
1	1	なし	終了エラー	00，03

AH内の数値によって終了状態をチェックできます。数値と終了状態を次に示します。

AH	終了状態
00H	中断
01H	先頭にEOLを検出
02H	FILL付きのEOLを先頭に検出
03H	EOL以外の符号を検出したあと，EOLを検出

【説　　明】　復号化処理において，“EOL”（000000000001）を検出するための命令です。メモリ上（受信バッファ）の指定された位置から最初のEOLを検出します。データの先頭はLSBに入ります。

割り込み要求のサンプリングを行いません。したがって，割り込み要求のサンプリングを確実に行うには，SCHEOL命令が一定時間ごとに中断されるように，受信バッファ・サイズを小さく設定してください。これにより，長時間にわたる割り込み要求の保留を回避することができます。SCHEOL命令の実行時間（最長となる場合）の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません）。

保守／廃止

● 16ビット・バス幅のとき：（214+T）N/16+74+T（クロック）

● 8ビット・バス幅のとき：（110+T）N/8+74+T（クロック）

N：EOL検出までのビット数

T：受信バッファのウエイト数

復号化処理のEOLを検出する処理において，次のような場合が考えられます。

① 指定された受信バッファ内が，すべて“0”の場合（EOL検出処理の中断）。

② 一番先頭にEOLを検出する場合。

③ FILL付きEOLを先頭に検出する場合。

④ FILL以外の符号を検出したあと，EOLを検出する場合。

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント） (DS2)	拡張セグメント・オーバライド・プリフィクス[注1]
02H	ワーク・エリア	
04H	処理中半端符号データ	
06H	処理中半端符号ビット数	
08H	受信バッファ・セグメント (DS0)	
0AH	受信バッファ・オフセット （ポインタ）	
0CH	受信バッファ・サイズ[注2] （カウンタ）	
0EH	ワーク・エリア	
10H	ワーク・エリア	
12H	ワーク・エリア	
14H	ワーク・エリア	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	ワーク・エリア	
1EH	0クリア[注3]/ワーク・エリア	…0クリア必要パラメータ

注1. DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2. バイト単位。偶数で指定してください。

3. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

保守／廃止

【記　述　例】　SCHEOL

【バ　イ　ト　数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SCHEOL	なし	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0

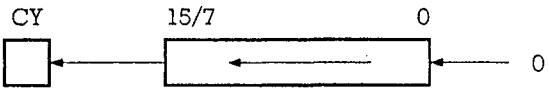
保守／廃止

SHL

左方向のシフト
Shift Left

【命 令 形 式】 **SHL dst, src**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SHL	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】 src＝1の場合

AC	CY	V	P	S	Z
U	×	×	×	×	×

左記以外

AC	CY	V	P	S	Z
U	×	U	×	×	×

【説 明】 ○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1ビットだけ左シフトします。

dstの内容のLSB（ビット0）には0がシフト・インされ，MSB（ビット7またはビット15）のデータはCYフラグにセットされます。

シフト後，符号ビット（ビット7またはビット15）が元のままであれば，Vフラグ

がクリアされます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけ左シフトします。1ビット，シフトするごとにdstの内容のLSB(ビット0)には0がシフト・インされ，MSB（ビット7またはビット15）のデータはCYフラグにセットされます。

【記　述　例】

```
IN    AW, 0C8H
MOV   [IY], AW
SHL   WORD PTR [IY], 12
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
SHL	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

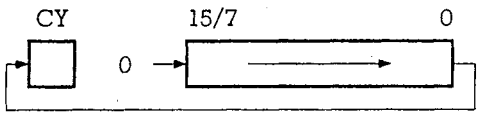
保守／廃止

SHR

右方向のシフト
Shift Right

【命 令 形 式】 **SHR dst, src**

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド (dst, src)
SHR	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】 src＝1の場合

AC	CY	V	P	S	Z
U	×	×	×	×	×

左記以外

AC	CY	V	P	S	Z
U	×	U	×	×	×

【説 明】 ○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1ビットだけ右シフトします。

dstの内容のMSB(ビット7またはビット15)には0がシフト・インされ,LSB(ビット0）のデータはCYフラグにセットされます。

保守／廃止

シフト後，符号ビット（ビット７またはビット15）が元のままであれば，Vフラグがクリアされます。

○src＝CLまたはsrc＝imm8のとき

第１オペランドで指定されるデスティネーション・オペランド（dst）の内容を第２オペランドで指定されるソース・オペランド（src）の内容のビット数分だけ右シフトします。1ビット，シフトするごとにdstの内容のMSB（ビット７またはビット15）には０がシフト・インされ，LSB（ビット０）のデータはCYフラグにセットされます。

【記 述 例】

●RCV：IN AL，0DAH
SHR AL，3
BC RCV
●SHR CW，8

【バ イ ト 数】

ニモニック	オペランド	バイト数
SHR	reg，1	2
	mem，1	2-4
	reg，CL	2
	mem，CL	2-4
	reg，imm8	3
	mem，imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHR	reg，1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg		
	mem，1	1	1	0	1	0	0	0	W	mod			1	0	1	mem	
		(disp-low)								(disp-high)							
	reg，CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg		
	mem，CL	1	1	0	1	0	0	1	W	mod			1	0	1	mem	
		(disp-low)								(disp-high)							
	reg，imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg		
		imm8								—							
	mem，imm8	1	1	0	0	0	0	0	W	mod			1	0	1	mem	
		(disp-low)								(disp-high)							
		imm8								—							

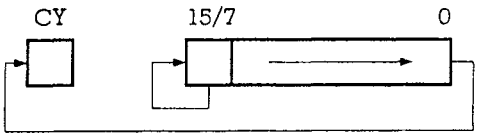
保守／廃止

SHRA

右方向の算術シフト
Shift Right Arithmetic

【命　令　形　式】　**SHRA dst, src**

【オペレーション】



【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst, src）
SHRA	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ　ラ　グ】　src＝1の場合

AC	CY	V	P	S	Z
U	×	0	×	×	×

左記以外

AC	CY	V	P	S	Z
U	×	U	×	×	×

【説　　明】　○src＝1のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1ビットだけ算術的に右シフトします。

dstの内容のMSB（ビット7またはビット15）には元と同じ値がシフト・インされ，シフト後も符号は変化しません。LSB(ビット0)のデータはCYフラグにセッ

保守／廃止

トされます。

○src＝CLまたはsrc＝imm8のとき

第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を第2オペランドで指定されるソース・オペランド（src）の内容のビット数分だけ右シフトします。dstの内容のMSB（ビット7またはビット15）には元と同じ値がシフト・インされ，シフト後も符号は変化しません。LSB（ビット0）のデータはCYフラグにセットされます。

【記 述 例】

```
●MOV   CL, 2
 SHRA  BL, CL
●MOV   CL, 9
 SHRA  DW, CL
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
SHRA	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

STM
STMB
STMW

ブロック・ストア
Store Multiple
Store Multiple Byte
Store Multiple Word

【命 令 形 式】　(repeat) STM dst-block
(repeat) STMB
(repeat) STMW

【オペレーション】　[W＝0のとき] (IX) ←AL
DIR＝0：IY←IY＋1
DIR＝1：IY←IY－1
[W＝1のとき] (IY＋1, IY) ←AW
DIR＝0：IY←IY＋2
DIR＝1：IY←IY－2

【オ ペ ラ ン ド】

ニモニック	オペランド
STM	dst-block
STMB	な　し
STMW	

【有効オーバライド・プリフィクス】　(STMだけ)

	dst
ディフォールト時	DS1：
プリフィクス付加時	DS1：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】　ALレジスタの値，またはAWレジスタの値をIYレジスタでアドレスされるブロックにバイト／ワード・データ単位に繰り返し転送します。
IYレジスタの値は次のバイト／ワード処理のために1バイト／ワード・データが処

理されるごとに自動的にインクリメント(+1／+2)またはデクリメント（−1／−2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，STM命令を用いる場合はオペランドの属性によって，STMB命令またはSTMW命令を用いる場合は，それぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。

【記　述　例】
●REP STM DS1：WORD_VAR：DS1セグメント
●REP STMB　　；DS1セグメント

【バ　イ　ト　数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
STM	dst-block	1	0	1	0	1	0	1	W
STMB	な　し								
STMW									

保守／廃止

STOP

STOP状態への移行

Stop

（V20，V30に対する追加命令）

【命 令 形 式】　**STOP**

【オペレーション】　CPUストップ

【オ ペ ラ ン ド】

ニモニック	オペランド
STOP	なし

【フ　ラ　グ】

AC	CY	V	P	S	Z

【説　　明】　ストップ状態にします。

【記 述 例】　STOP

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
STOP	なし	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1

保守／廃止

SUB

減算
Subtract

【命 令 形 式】　**SUB dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
SUB	reg, reg′	dst← dst−src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL← AL−imm8 [W＝1のとき] AW← AW−imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ ラ グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

【説　　明】　第1オペランドで指定されるデスティネーション・オペランド（dst）の内容から第2オペランドで指定されるソース・オペランド（src）の内容を減算し，結果をデスティネーション・オペランド（dst）に格納します。

【記 述 例】　DLレジスタの内容からメモリ0：50Hの内容を減算してDLレジスタへストアする。

```
MOV AW, 0
MOV DS0, AW
MOV IX, 50H
SUB DL, DS0 : BYTE PTR [IX]
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SUB	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUB	reg, reg'	0	0	1	0	1	0	1	W	1	1	reg				reg'	
	mem, reg	0	0	1	0	1	0	0	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	0	1	0	1	W	mod		reg				mem	
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	1	0	1		reg	
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		1	0	1		mem	
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	0	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

SUB4S

10進減算
Subtract Nibble String

【命 令 形 式】 **SUB4S dst-string, src-string**
SUB4S

【オペレーション】 BCDストリング（IY，CL）← BCDストリング（IY，CL）−BCDストリング（IX，CL）

【オ ペ ラ ン ド】

ニモニック	オペランド（dst，src）
SUB4S	dst-string，src-string
	な し

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	DS1：
プリフィクス付加時	DS0：，DS1：，PS：，SS：，DS2：，	DS1：，DS3：，IRAM：

【フ ラ グ】

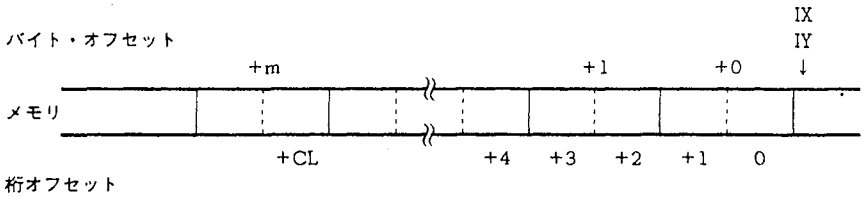
AC	CY	V	P	S	Z
U	×	U	U	U	×

【説 明】 IYレジスタでアドレスされるパックトBCDストリングからIXレジスタでアドレスされるパックトBCDストリングを減算し，結果をIYレジスタでアドレスされるストリングにストアします。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され，1-254桁まで可能です。

デスティネーション・ストリングのディフォールト・セグメント・レジスタはDS1となっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3で指定されるセグメント内にもロケートできます。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，拡張セグメント・レジスタDS3以外のセグメント・レジスタで指定されるセグメント内にロケートできます。

保守／廃止

パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋1の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示します。
したがって，桁数が奇数のときのBCD減算命令は，最上位バイトの上位４ビットを“0”にしてから実行してください。この結果ボローが生じた場合は最上位バイトの上位４ビットは“9”になります。

【記　述　例】

```
MOV IX, OFFSET VAR_1
MOV IY, OFFSET VAR_2
MOV CL, 4
SUB4S
```

【バ　イ　ト　数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUB4S	dst-string, src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0
	な　し																

保守／廃止

SUBC

キャリーを含む減算
Subtract with Carry

【命　令　形　式】　**SUBC dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
SUBC	reg, reg'	dst← dst−src−CY
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき]AL← AL−imm8−CY [W＝1のとき]AW← AW−imm16−CY

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
×	×	×	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）の内容から第２オペランドで指定されるソース・オペランド（src）の内容をCYフラグの内容も含めて減算し，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　SUBC DL, BYTE PTR [IX]

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SUBC	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUBC	reg, reg'	0	0	0	1	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	1	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	1	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	1	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	1	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

TEST

テスト
Test

【命　令　形　式】　**TEST dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
TEST	reg, reg'	dst∧src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL∧imm8 [W＝1のとき] AW∧imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	0	0	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）と第２オペランドで指定されるソース・オペランド（src）の論理積（AND）をとります。
結果はどこへも格納せず，フラグを変化させます。

【記　述　例】
```
IN    AL, 0D8H
TEST  AL, 'A'
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
TEST	reg, reg′	2
	mem, reg	2-4
	reg, mem	
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TEST	reg, reg′	1	0	0	0	0	1	0	W	1	1	reg′			reg		
	mem, reg	1	0	0	0	0	1	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	0	1	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	1	0	1	0	1	0	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

TEST1

ビットのテスト
Test Bit

【命 令 形 式】　**TEST1 dst, src**

【オペレーション】　dstのビットn＝0（nはsrcで指定）のとき：Z← 1
dstのビットn＝1（nはsrcで指定）のとき：Z← 0

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
TEST1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

【有効オーバライド・プリフィクス】

	dst
ディフォールト時	DS0：
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z
U	0	0	U	U	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）が０のときはZフラグがセット(1)され，ビットnが１のときはリセット(0)されます。
オペランドがreg8，CLまたはmem8，CLのとき，CLの値は下位３ビット(0-7)のみ有効です。

オペランドがreg16，CLまたはmem16，CLのとき，CLの値は下位４ビット(0-15)のみ有効です。

オペランドがreg8，imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。

オペランドがmem8，imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。

オペランドがreg16，imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。

オペランドがmem16，imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

【記　述　例】

```
MOV  CL, 01
IN   AL, 0DAH
TEST1 AL, CL ; ビット１をテスト
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
TEST1	reg8，CL	3
	mem8，CL	3-5
	reg16，CL	3
	mem16，CL	3-5
	reg8，imm3	4
	mem8，imm3	4-6
	reg16，imm4	4
	mem16，imm4	4-6

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TEST1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							

保守／廃止

TRANS
TRANSB

変換テーブルの転送
Translate
Translate Byte

【命 令 形 式】 **TRANS src-table**
TRANS
TRANSB

【オペレーション】 AL← (BW+AL)

【オ ペ ラ ン ド】

ニモニック	オペランド
TRANS	src-table
	な し
TRANSB	な し

【有効オーバライド・プリフィクス】

	src
ディフォールト時	DS0：
プリフィクス付加時	DS0：,DS1：, PS：,SS：, DS2：,DS3：, IRAM：

【フ ラ グ】

AC	CY	V	P	S	Z

【説 明】 BWレジスタとALレジスタによってアドレスされる256バイトの変換テーブルの1バイトをALレジスタに転送します。このときBWレジスタはテーブルの先頭アドレスを指し，先頭アドレスから256バイト内のオフセット値をALレジスタが指定します。

【記 述 例】 TRANS SIN_TBL

【バ イ ト 数】 1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
TRANS	src-table	1	1	0	1	0	1	1	1
	な　し								
TRANSB	な　し								

保守／廃止

TSKSW

レジスタ・バンク切り替え

Task Switch

（V20，V30に対する追加命令，V25，V35に対する拡張命令）

【命　令　形　式】　**TSKSW reg16**

【オペレーション】　現在選択されているレジスタ・バンク上のPSW退避領域←PSW
現在選択されているレジスタ・バンク上のPC退避領域←PC
RB3-RB0←reg16の下位４ビット
PSW←新たに選択されたレジスタ・バンク上のPSW退避領域の値
PC←新たに選択されたレジスタ・バンク上のPC退避領域の値

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
TSKSW	reg16

【フ　ラ　グ】

AC	CY	V	P	S	Z	RB0	RB1	RB2	RB3	DIR	IE	BRK	$\overline{\text{IBRK}}$
×	×	×	×	×	×	×	×	×	×	×	×	×	×

【説　　明】　レジスタ・バンクを，オペランドに記述した16ビット・レジスタの内容の下位４ビットで示されるレジスタ・バンクに切り替えます。また，新しいレジスタ・バンク内にあらかじめストアしておいたPSとPCから得られるアドレスに分岐します。
高速なタスク切り替えなどに使用します。

【記　述　例】　TSKSW BP

【バ　イ　ト　数】　3

【命令語形式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TSKSW	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0
		1	1	1	1	1	reg			—							

保守／廃止

XCH

データ交換
Exchange

【命 令 形 式】 **XCH dst, src**

【オペレーション】 dst↔src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XCH	reg, reg'
	mem, reg
	reg, mem
	AW, reg16
	reg16, AW

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ ラ グ】

AC	CY	V	P	S	Z

【説　　明】 第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）の内容を交換します。

【記 述 例】
```
MOV AW, 100H
MOV BW, 50H
XCH AW, BW
；AW＝50H, BW＝100Hとなる。
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
XCH	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	AW, reg16	1
	reg16, AW	

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
XCH	reg, reg'	1	0	0	0	0	1	1	W	1	1	reg			reg'		
	mem, reg	1	0	0	0	0	1	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	0	1	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	AW, reg16	1	0	0	1	0	reg			—							
	reg16, AW	1	0	0	1	0	reg			—							

保守／廃止

XOR

排他的論理和
Exclusive Or

【命　令　形　式】　**XOR dst, src**

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
XOR	reg, reg'	dst← dst∀src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	[W＝0のとき] AL← AL∀imm8 [W＝1のとき]AW← AW∀imm16

【有効オーバライド・プリフィクス】

	src	dst
ディフォールト時	DS0：	
プリフィクス付加時	DS0：, DS1：, PS：, SS：, DS2：, DS3：, IRAM：	

【フ　ラ　グ】

AC	CY	V	P	S	Z
U	0	0	×	×	×

【説　　明】　第１オペランドで指定されるデスティネーション・オペランド（dst）と第２オペランドで指定されるソース・オペランド（src）の排他的論理和（XOR）をとり，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】
- XOR CL, DL
- XOR CW, CW；CWレジスタのクリア
- XOR AW, DW

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
XOR	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
XOR	reg, reg'	0	0	1	1	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	1	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	1	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	1	1	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	1	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注　アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	W	1	1	1	1	0	reg		
imm8								—							

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

2.2 命令実行クロック数

各命令の実行クロック数をニモニックのアルファベット順に示します。ただし，キュー操作命令，コーデック命令については，他の命令と分けて説明しています。

また，表中の数値は実行ユニットが命令実行に必要とする時間です。プリフェッチ時間，バス調停の待ち時間などは含まれません。

(1) クロック欄の分類

命令のクロック数は，処理方法（バイト／ワード）やバス幅（8/16），命令によってアクセスまたは分岐するメモリやレジスタ（内蔵RAM／内蔵RAM以外），および各種の条件により異なります。

表中の外部データ・バス幅の表記は次のとおりです。

8：8ビット・バス幅のときを示します。

16：16ビット・バス幅のときを示します。

－：8ビット・バス幅と16ビット・バス幅で共通です。

(2) クロック欄の記号

クロック数は，メモリ・オペランドの場合アドレシング・モードによって異なります。このため，次の記号を使って表しています。

EA：次の表で求めた数値を用いてください。

T　：ウエイト数を示します。"0"（ウエイトなし）または任意のウエイト・ステート数を適用してください。

t　：ウエイト数Tの値により次の式で求めた数値を用いてください。

$T \geqq 2$のとき　$t = T - 1$

$T < 2$のとき　$t = 0$

mod / mem	00	EA	01	EA	10	EA
000	BW＋IX	3	BW＋IX＋disp8	3	BW＋IX＋disp16	3
001	BW＋IY	3	BW＋IY＋disp8	3	BW＋IY＋disp16	3
010	BP＋IX	3	BP＋IX＋disp8	3	BP＋IX＋disp16	3
011	BP＋IY	3	BP＋IY＋disp8	3	BP＋IY＋disp16	3
100	IX	2	IX＋disp8	2	IX＋disp16	2
101	IY	2	IY＋disp8	2	IY＋disp16	2
110	ダイレクト・アドレス	2	BP＋disp8	2	BP＋disp16	2
111	BW	2	BW＋disp8	2	BW＋disp16	2

保守／廃止

また，プリミティブ・ブロック転送命令とプリミティブ入出力命令については，次の３つの場合に分けてクロック数を算出しています。

① リピートなしの場合

② リピートあり（CW≠0）の場合

③ 〃　（CW＝0）の場合

保守／廃止

2.2.1 命令実行クロック数（キュー操作命令，コーデック命令以外）

表2-11 命令実行クロック数一覧（1/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
ADD	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	acc, imm	—	2	2	2	2
ADD4S	dst-string, src-string	8	6+(15+T)n	6+(19+3T)n	—	—
		16				
ADDC	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	acc, imm	—	2	2	2	2
ADJ4A		—	3	3	—	—
ADJ4S		—	3	3	—	—
ADJBA		8	6	9	—	—
		16	9			
ADJBS		8	6	6	—	—
		16	9	9		

備考　n：BCD桁数の1/2

保守／廃止

表2－11 命令実行クロック数一覧（2/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
AND	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	acc, imm	—	2	2	2	2
BC/BL[注]	short-label	—	—	—	9/3	9/3
BCWZ[注]	short-label	—	—	—	10/5	10/5
BE/BZ[注]	short-label	—	—	—	9/3	9/3
BGE[注]	short-label	—	—	—	9/3	9/3
BGT[注]	short-label	—	—	—	9/3	9/3
BH[注]	short-label	—	—	—	9/3	9/3
BLE[注]	short-label	—	—	—	9/3	9/3
BLT[注]	short-label	—	—	—	9/3	9/3
BN[注]	short-label	—	—	—	9/3	9/3
BNC/BNL[注]	short-label	—	—	—	9/3	9/3
BNE/BNZ[注]	short-label	—	—	—	9/3	9/3
BNH[注]	short-label	—	—	—	9/3	9/3
BNV[注]	short-label	—	—	—	9/3	9/3
BP[注]	short-label	—	—	—	9/3	9/3
BPE[注]	short-label	—	—	—	9/3	9/3
BPO[注]	short-label	—	—	—	9/3	9/3
BR	near-label	—	—	—	—	9
	short-label	—	—	—	—	9
	regptr16	—	—	—	—	8
	memptr16	8	—	—	EA+9	EA+14+2T
		16				EA+11+T
	far-label	—	—	—	—	9
	memptr32	8	—	—	EA+12	EA+24+4T
		16				EA+18+2T

注 クロック数で，／の左側は分岐するとき（条件成立），右側は分岐しないとき

保守／廃止

表2-11 命令実行クロック数一覧 (3/18)

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
BRK[注1]	3	8	—	—	—	50+10T
		16				36+4T+t
	imm8(≠3)	8	—	—	—	52+10T
		16				38+4T+t
BRKCS	reg16	—	—	—	12	12
BRKV[注1]		8	—	—	—	51+10T
		16				37+4T+t
BSCH	mem	8	EA+8+3m+T	EA+8+3m+T	EA+11+3m+2T	EA+11+3m+2T
		16			EA+8+3m+T	EA+8+3m+T
	reg	—	4+3m	4+3m	4+3m	4+3m
BTCLR	sfr, imm3, short-label	8	—	21/14	—	—
		16				
BTCLRL	sfr, imm3, short-label	8	—	20/13	—	—
		16				
BUSLOCK		—	0, 1	0, 1	0, 1	0, 1
BV	short-label	—	—	—	9/3	9/3
CALL	near-proc	8	—	—	—	19+2T
		16				16+T
	regptr16	8	—	—	—	18+2T
		16				15+T
	memptr16	8	—	—	EA+19+2T	EA+24+4T
		16			EA+16+T	EA+18+2T
	far-proc	8	—	—	—	29+4T
		16				23+2T
	memptr32	8	—	—	EA+32+4T	EA+44+8T
		16			EA+26+2T	EA+32+4T
CHKIND[注2]		8	—	—	EA+11	EA+21+4T
		16				EA+15+2T

注1. BRK=1において，8ビット・バス幅のとき50+10T，16ビット・バス幅のとき34+4Tを加算します。

2. (mem32)>reg16または(mem32+2)<reg16において，8ビット・バス幅のとき50+10T，16ビット・バス幅のとき34+4T+tを加算します。

備考 m：サーチしたビット番号

保守／廃止

表2－11 命令実行クロック数一覧（4/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
CLR1	reg8, CL	—	3	3	3	3
	mem8, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, CL	—	3	3	3	3
	mem16, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg8, imm3	—	2	2	2	2
	mem8, imm3	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, imm4	—	2	2	2	2
	mem16, imm4	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	CY	—	2	2	2	2
	DIR	—	2	2	2	2
CMP	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	acc, imm	—	2	2	2	2
CMP4S	dst-string,	8	6+(15+T)a	6+(18+2T)a	—	—
	src-string	16				
CMPBK	scr-block,	8	①20+T	①22+2T	①23+2T	①28+4T
	dst-block		②9+(13+T)a	②9+(15+2T)a	②9+(16+2T)a	②9+(21+4T)a
CMPBKB/			③5	③5	③5	③5
CMPBKW		16			①20+T	①22+2T
					②9+(13+T)a	②9+(15+2T)a
					③5	③5

備考 a：リピート回数

保守／廃止

表2-11　命令実行クロック数一覧（5/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
CMPM	dst-block	8	①15 ②10+7a ③5	①17+T ②10+(9+T)a ③5	①15 ②10+7a ③5	①20+T ②10+(12+2T)a ③5
CMPMB/ CMPMW		16				①17+T ②10+(9+T)a ③5
CVTBD		—	18	18	—	—
CVTBW		—	3	3	—	—
CVTDB		—	8	8	—	—
CVTWL		—	—	—	3	3
DBNZ[注1]	short-label	—	—	—	10/5	10/5
DBNZE[注1]	short-label	—	—	—	10/5	10/5
DBNZNE[注1]	short-label	—	—	—	10/5	10/5
DEC	reg8	—	2	2	—	—
	mem	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg16	—	—	—	2	2
DI		—	3	3	3	3
DISPOSE		8	—	—	—	10+2T
		16				7+T
DIV[注2]	reg8	8	17/64+10T	17/64+10T	25/59+10T	25/59+10T
		16	17/44+5T	17/44+5T	25/44+5T	25/44+5T
	mem8	8	EA+18/65+10T	EA+20+T/65+10T	EA+26/60+10T	EA+31+2T/60+10T
		16	EA+18/45+5T	EA+20+T/45+5T	EA+26/45+5T	EA+28+T/45+5T
	reg16	8	17/64+10T	17/64+10T	25/59+10T	25/59+10T
		16	17/44+5T	17/44+5T	25/44+5T	25/44+5T
	mem16	8	EA+18/65+10T	EA+20+T/65+10T	EA+26/60+10T	EA+31+2T/60+10T
		16	EA+18/45+5T	EA+20+T/45+5T	EA+26/45+5T	EA+28+T/45+5T

注1. クロック数で，／の左側は分岐するとき（条件成立），／の右側は分岐しないとき

2. クロック数で，／の右側はディバイド・エラーのとき

備考　a：リピート回数

保守／廃止

表2-11 命令実行クロック数一覧（6/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
DIVU[注1]	reg8	8	15/62+10T	15/62+10T	23/57+10T	23/57+10T
		16	15/42+5T	15/42+5T	23/42+5T	23/42+5T
	mem8	8	EA+16/63+10T	EA+18+T/63+10T	EA+24/58+10T	EA+30+2T/58+10T
		16	EA+16/43+5T	EA+18+T/43+5T	EA+24/43+5T	EA+26+T/43+5T
	reg16	8	15/62+10T	15/62+10T	23/57+10T	23/57+10T
		16	15/42+5T	15/42+5T	23/42+5T	23/42+5T
	mem16	8	EA+16/63+10T	EA+18+T/63+10T	EA+24/58+10T	EA+30+2T/58+10T
		16	EA+16/43+5T	EA+18+T/43+5T	EA+24/43+5T	EA+26+T/43+5T
DS0：, DS1：, PS：, SS：		—	0, 1	0, 1	0, 1	0, 1
DS2：, DS3：		—	0, 1	0, 1	0, 1	0, 1
EI		—	3	3	3	3
EXT	reg8, reg8'	8	—	—	19～41	19+2T～48+4T
		16				19～42+2T
	reg8, imm4	8	—	—	19～41	19+2T～48+4T
		16				19～42+2T
FINT		—	3	3	3	3
FPO1	fp-op	8	—	—	—	50+10T
		16				36+4T+t
	fp-op, mem	8	—	—	—	EA+50+10T
		16				EA+36+4T+t
FPO2	fp-op	8	—	—	—	50+10T
		16				36+4T+t
	fp-op, mem	8	—	—	—	EA+50+10T
		16				EA+36+4T+t
HALT		—	—	—	—	—
IN[注2]	acc, imm8	8	—	60+10T	—	60+10T
		16		40+5T		40+5T
	acc, DW	8	—	60+10T	—	60+10T
		16		40+5T		40+5T

注1. クロック数で，／の右側はディバイド・エラーのとき

2. $\overline{IBRK}$=0のとき

保守／廃止

表2－11　命令実行クロック数一覧（7/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
IN[注1]	acc, imm8	8	—	7+T	—	10+2T
		16				7+T
	acc, DW	8	—	7+T	—	10+2T
		16				7+T
INC	reg8	—	2	2	—	—
	mem	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg16	—	—	—	2	2
INM[注2]	dst-block, DW	8	—	60+10T	—	60+10T
		16		40+5T		40+5T
INM[注1]	dst-block, DW	8	①17+T ②9+(10+T)a ③5	①18+T ②9+(11+2T)a ③5	①20+2T ②9+(13+2T)a ③5	①21+2T ②9+(14+4T)a ③5
		16			①17+T ②9+(10+T)a ③5	①18+T ②9+(11+2T)a ③5
INS	reg8, reg8'	8	—	—	22～63	31～72
		16				23～64
	reg8, imm4	8	—	—	22～63	31～72
		16				23～64
IRAM：		—	0, 1	0, 1	0, 1	0, 1
LDEA	reg16, mem16	—	—	—	EA+2	EA+2
LDM	src-block	8	①10 ②9+3a ③5	①13+T ②9+(6+T)a ③5	①10 ②9+3a ③5	①16+T ②9+(9+2T)a ③5
LDMB/ LDMW		16				①13+T ②9+(6+T)a ③5

注1. $\overline{IBRK}$＝1のとき

2. $\overline{IBRK}$＝0のとき

備考　a：リピート回数

保守／廃止

表2-11 命令実行クロック数一覧 (8/18)

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
MOV	reg, reg'	—	2	2	2	2
	mem, reg	—	EA+2	EA+3	EA+2	EA+3
	reg, mem	8	EA+2	EA+5+T	EA+2	EA+8+2T
		16				EA+5+T
	mem, imm	—	EA+2	EA+3	EA+2	EA+3
	reg, imm	—	2	2	2	2
	acc, dmem	8	4	7+T	4	10+2T
		16				7+T
	dmem, acc	—	4	5	4	5
	sreg, reg16	—	—	—	2	2
	xsreg, reg16	8	—	—	2	2
	VPC, reg16	16				
	sreg, mem16	8	—	—	EA+2	EA+8+2T
		16				EA+5+T
	xsreg, mem16	8	—	—	EA+2	EA+8+2T
	VPC, mem16	16				EA+5+T
	reg16, sreg	—	—	—	2	2
	reg16, xsreg	8	—	—	2	2
	reg16, VPC	16				
	mem16, sreg	—	—	—	EA+2	EA+3
	mem16, xsreg	8	—	—	EA+2	EA+3
	mem16, VPC	16				
	DS0,	8	—	—	EA+5	EA+17+4T
	reg16, mem32	16				EA+11+2T
	DS2,	8	—	—	EA+5	EA+17+4T
	reg16, mem32	16				EA+11+2T
	DS1,	8	—	—	EA+5	EA+17+4T
	reg16, mem32	16				EA+11+2T
	DS3,	8	—	—	EA+5	EA+17+4T
	reg16, mem32	16				EA+11+2T
	AH, PSW	—	2	2	—	—
	PSW, AH	8	3	3	—	—
		16	2	2		

保守／廃止

表2－11 命令実行クロック数一覧（9/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
MOVBK	dst-block, src-block	8	①18+T ②9+(11+T)a ③5	①19+T ②9+(12+2T)a ③5	①21+2T ②9+(14+2T)a ③5	①22+2T ②9+(18+4T)a ③5
MOVBKB/MOVBKW		16			①18+T ②9+(11+T)a ③5	①19+T ②9+(12+2T)a ③5
MOVSPA		—	—	—	8	8
MOVSPB	reg16	—	—	—	9	9
MUL	reg8	—	10	10	14	14
	mem8	8	EA+11	EA+13+T	EA+15	EA+20+2T
		16				EA+17+T
	reg16	—	10	10	14	14
	mem16	8	EA+11	EA+13+T	EA+15	EA+20+2T
		16				EA+17+T
	reg16, reg16', imm8/reg16, imm8	—	—	—	14	14
	reg16, mem16, imm8	8	—	—	EA+15	EA+20+2T
		16				EA+17+T
	reg16, reg16', imm16/reg16, imm16	—	—	—	14	14
	reg16, mem16, imm16	8	—	—	EA+15	EA+20+2T
		16				EA+17+T
MULU	reg8	—	11	11	15	15
	mem8	8	EA+12	EA+14+T	EA+16	EA+21+2T
		16				EA+18+T
	reg16	—	11	11	15	15
	mem16	8	EA+12	EA+14+T	EA+16	EA+21+2T
		16				EA+18+T
NEG	reg	—	2	2	2	2
	mem	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T

備考 a：リピート回数

保守／廃止

表2-11 命令実行クロック数一覧（10/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
NOP		—	4	4	4	4
NOT	reg	—	2	2	2	2
	mem	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
NOT1	reg8, CL	—	3	3	3	3
	mem8, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, CL	—	3	3	3	3
	mem16, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg8, imm3	—	2	2	2	2
	mem8, imm3	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, imm4	—	2	2	2	2
	mem16, imm4	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	CY	—	2	2	2	2
OR	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	acc, imm	—	2	2	2	2
OUT[注]	imm8, acc	8	—	5	—	5
		16				
	DW, acc	8	—	5	—	5
		16				

注 $\overline{IBRK}$=0のとき

保守／廃止

表2－11　命令実行クロック数一覧（11/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
OUT[注1]	imm8, acc	8	—	60+10T	—	60+10T
		16		40+5T		40+5T
	DW, acc	8	—	60+10T	—	60+10T
		16		40+5T		40+5T
OUTM[注2]	DW, src-block	8	—	60+10T	—	60+10T
		16		40+5T		40+5T
OUTM[注1]	DW, src-block	8	①14+T ②9+(7+T)a ③5	①17+2T ②9+(10+2T)a ③5	①17+2T ②9+(10+2T)a ③5	①23+4T ②9+(16+4T)a ③5
		16			①14+T ②9+(7+T)a ③5	①17+2T ②9+(10+2T)a ③5
POLL		—	—	—	—	—
POP	mem16	8	—	—	EA+13+2T	EA+14+2T
		16			EA+10+T	EA+11+T
	reg16	8	—	—	—	10+2T
		16				7+T
	sreg	8	—	—	—	10+2T
		16				7+T
	xsreg/VPC	8	—	—	—	10+2T
		16				7+T
	PSW	8	—	—	—	11+2T
		16				8+T
	R	8	—	—	—	76+16T
		16				52+8T
PREPARE[注3]	imm16, imm8	8	—	—	—	15+2T+(16+4T)b
		16				14+(12+T)b

注1. $\overline{IBRK}$＝1のとき

2. $\overline{IBRK}$＝0のとき

3. imm8≧1のとき

備考　a：リピート回数

b：imm8の値

保守／廃止

表2-11 命令実行クロック数一覧（12/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
PREPARE[注1]	imm16, imm8	—	—	—	—	9
PUSH	mem16	8	—	—	EA+7	EA+13+2T
		16				EA+10+T
	reg16	—	—	—	—	7
	sreg	—	—	—	—	7
	xsreg/VPC	—	—	—	—	7
	PSW	—	—	—	—	6
	R	8	—	—	—	57+14T
		16				36+7T
	imm8	—	—	—	—	6
	imm16	—	—	—	—	6
REP/ REPE/ REPZ		—	0, 1	0, 1	0, 1	0, 1
REPC		—	0, 1	0, 1	0, 1	0, 1
REPNC		—	0, 1	0, 1	0, 1	0, 1
REPNE/ REPNZ		—	0, 1	0, 1	0, 1	0, 1
RET		8	—	—	—	18+2T
		16				15+T
	pop-value	8	—	—	—	19+2T
		16				16+T
	[注2]	8	—	—	—	26+4T
		16				20+2T
	pop-value[注2]	8	—	—	—	27+4T
		16				21+2T
RETI		8	—	—	—	28+4T
		16				22+2T
RETRBI		—	—	—	—	9

注1. imm8＝0のとき

2. セグメント外を示します。

保守／廃止

表2－11　命令実行クロック数一覧（13/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
ROL	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
ROL4	reg8	8	5	5	—	—
	mem8	16	EA+5	EA+8+T	—	—
ROLC	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
ROR	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
ROR4	reg8	8	5	5	—	—
	mem8	16	EA+5	EA+8+T	—	—

備考　c：シフト数

保守／廃止

表2-11　命令実行クロック数一覧（14/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
RORC	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
RSTWDT[注]	imm8, imm8'	8	—	9/54+10T	—	—
		16		9/40+4T+t		
SET1	reg8, CL	—	3	3	3	3
	mem8, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, CL	—	3	3	3	3
	mem16, CL	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg8, imm3	—	2	2	2	2
	mem8, imm3	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg16, imm4	—	2	2	2	2
	mem16, imm4	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	CY	—	2	2	2	2
	DIR	—	2	2	2	2

注　クロック数で，／の右側はデータ・エラーのときにワード処理を行う場合です。

備考　c：シフト数

保守／廃止

表2－11　命令実行クロック数一覧（15/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
SHL	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
SHR	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
SHRA	reg, 1	—	3	3	3	3
	mem, 1	8	EA+3	EA+7+T	EA+3	EA+10+2T
		16				EA+7+T
	reg, CL	—	5+c	5+c	5+c	5+c
	mem, CL	8	EA+5+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c
	reg, imm8	—	5+c	5+c	5+c	5+c
	mem, imm8	8	EA+6+c	EA+8+T+c	EA+6+c	EA+11+2T+c
		16				EA+8+T+c

備考　c：シフト数

保守／廃止

表2－11　命令実行クロック数一覧（16/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
STM	dst-block	8	①12 ②9＋5a ③5	①13 ②9＋(6＋T)a ③5	①12 ②9＋5a ③5	①13 ②9＋(9＋2T)a ③5
STMB/ STMW		16				①13 ②9＋(6＋T)a ③5
STOP		—	—	—	—	—
SUB	reg, reg'	—	3	3	3	3
	mem, reg	8	EA＋4	EA＋7＋T	EA＋4	EA＋10＋2T
		16				EA＋7＋T
	reg, mem	8	EA＋2	EA＋6＋T	EA＋2	EA＋9＋2T
		16				EA＋6＋T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA＋4	EA＋7＋T	EA＋4	EA＋10＋2T
		16				EA＋7＋T
	acc, imm	—	2	2	2	2
SUB4S	dst-string, src-string	8	6＋(16＋T)n	6＋(20＋3T)n	—	—
		16				
SUBC	reg, reg'	—	3	3	3	3
	mem, reg	8	EA＋4	EA＋7＋T	EA＋4	EA＋10＋2T
		16				EA＋7＋T
	reg, mem	8	EA＋2	EA＋6＋T	EA＋2	EA＋9＋2T
		16				EA＋6＋T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA＋4	EA＋7＋T	EA＋4	EA＋10＋2T
		16				EA＋7＋T
	acc, imm	—	2	2	2	2

備考　a：リピート回数

n：BCD桁数の1/2

保守／廃止

表2-11 命令実行クロック数一覧（17/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
TEST	reg, reg'	—	3	3	3	3
	mem, reg/	8	EA+4	EA+6+T	EA+4	EA+9+2T
	reg, mem	16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	acc, imm	—	2	2	2	2
TEST1	reg8, CL	—	3	3	3	3
	mem8, CL	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	reg16, CL	—	3	3	3	3
	mem16, CL	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	reg8, imm3	—	2	2	2	2
	mem8, imm3	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
	reg16, imm4	—	2	2	2	2
	mem16, imm4	8	EA+4	EA+6+T	EA+4	EA+9+2T
		16				EA+6+T
TRANS/ TRANSB	src-table	—	6	9+T	—	—
TSKSW	reg16	—	—	—	13	13
XCH	reg, reg'	—	4	4	4	4
	mem, reg/		EA+4	EA+7+T	EA+4	EA+10+2T
	reg, mem					EA+7+T
	AW, reg16/ reg16, AW	—	—	—	4	4

保守／廃止

表2－11　命令実行クロック数一覧（18/18）

ニモニック	オペランド	バス幅	バイト処理		ワード処理	
			内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
XOR	reg, reg'	—	3	3	3	3
	mem, reg	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	reg, mem	8	EA+2	EA+6+T	EA+2	EA+9+2T
		16				EA+6+T
	reg, imm	—	2	2	2	2
	mem, imm	8	EA+4	EA+7+T	EA+4	EA+10+2T
		16				EA+7+T
	acc, imm	—	2	2	2	2

保守／廃止

2.2.2 命令実行クロック数（キュー操作命令）

表2－12 命令実行クロック数一覧（キュー操作命令）

ニモニック	オペランド	条　件	バス幅	バイト処理		ワード処理	
				内蔵RAMアクセス	内蔵RAM以外アクセス	内蔵RAMアクセス	内蔵RAM以外アクセス
QHOUT	なし	タスク数>1	8	—	—	—	41+6T
			16				32+3T
		タスク数=1	8	—	—	—	33+4T
			16				27+2T
		キュー数=0	8	—	—	—	19+2T
			16				16+T
QOUT	なし	タスク数>1[注1]	8	—	—	—	44+6T
			16				35+3T
		タスク数>1[注2]	8	—	—	—	59+10T
			16				44+5T
		タスク数>1[注3]	8	—	—	—	43+6T
			16				34+3T
		タスク数=1	8	—	—	—	35+4T
			16				29+2T
		キュー数=0	8	—	—	—	19+2T
			16				16+T
		パラメータ・エラーのとき	8	—	—	—	62+10T
			16				47+5T
QTIN	なし	タスク数>0	8	—	—	—	46+8T
			16				34+4T
		キュー数=0	8	—	—	—	29+4T
			16				23+2T

注1. 先頭ブロックを指定した場合

2. 中間ブロックを指定した場合

3. 最後尾ブロックを指定した場合

保守／廃止

2.2.3 命令実行クロック数（コーデック命令）

(1) ALBIT命令

外部バス幅	条　件	クロック数
8ビット・バス幅	CL+CH<16のとき	14+n
	CL+CH=16 かつCH=0のとき	18
	上記以外	43−m+n
16ビット・バス幅	CL+CH<16のとき	14+n
	CL+CH=16 かつCH=0のとき	18
	上記以外	43−m+n

m：CHレジスタの設定値

n ：CLレジスタの設定値

(2) CNVTRP命令

CNVTRP命令実行中に割り込みによる中断がない場合のクロック数を示します。

なお，割り込みによる中断があった場合は，どの変化点情報まで画素データに変換する処理を終了したかによってクロック数が異なってくるため，数値は算出していません。

外部バス幅	条　件	クロック数
8ビット・バス幅	変化点情報の個数（n）が偶数の場合 （変化点が黒で終わる場合）	(34+53n)/2+14m+8k+ 2（m+k）T+2（n+1）t
	変化点情報の個数（n）が奇数の場合 （変化点が白で終わる場合）	(33+53n)/2+14m+8k+ 2（m+k）T+2（n+1）t
16ビット・バス幅	変化点情報の個数（n）が偶数の場合 （変化点が黒で終わる場合）	(28+47n)/2+11m+5k+ （m+k）T+（n+1）t
	変化点情報の個数（n）が奇数の場合 （変化点が白で終わる場合）	(27+47n)/2+11m+5k+ （m+k）T+（n+1）t

k ：全白／全黒の画素データを送出する回数

m：全白／全黒以外の画素データを送出する回数

n ：変化点情報の個数（白の変化点情報の個数と黒の変化点情報の個数の和）

T ：画素データ・バッファが置かれている領域のウエイト数

t ：変化点テーブルが置かれている領域のウエイト数

保守／廃止

(3) COLTRP命令

COLTRP命令実行中に割り込みによる中断がない場合のクロック数を示します。

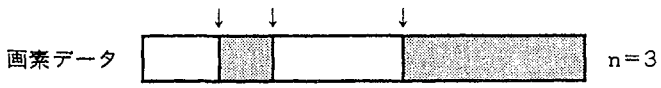
なお，割り込みによる中断があった場合は，どの画素データまで処理が終了しているかによってクロック数が異なってくるため，数値は算出していません。

外部バス幅	条　件	クロック数
8ビット・バス幅	画素データが白から始まる場合	17+60m+7n+2mT
	画素データが黒から始まる場合	24+60m+7n+2mT
16ビット・バス幅	画素データが白から始まる場合	17+57m+7n+mT
	画素データが黒から始まる場合	24+57m+7n+mT

m：画素データ・バッファ・サイズ（バイト）

n：画素データの色が白→黒へ，または黒→白へ変化する回数

例



T：画素データ・バッファが置かれている領域のウエイト数

(4) GETBIT命令

外部バス幅	条　件	クロック数
8ビット・バス幅	処理中の半端符号ビット数≠0のとき	13
	処理中の半端符号ビット数＝0のとき	18+2T
16ビット・バス幅	処理中の半端符号ビット数≠0のとき	13
	処理中の半端符号ビット数＝0のとき	15+T

T：受信データ・バッファが置かれている領域のウエイト数

(5) SCHEOL命令

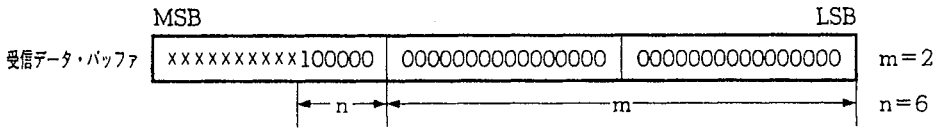
次の表の条件を満たさない場合，つまりFILL以外のコードを検出したあとにEOLを検出した場合については，FILL以外のコードの数値によってクロック数が異なってくるため，数値は算出していません。

外部バス幅	条　件	クロック数
8ビット・バス幅	先頭にEOLを検出した場合	24＋59m＋3n＋2mT
	先頭にFILL付きのEOLを検出した場合	
16ビット・バス幅	先頭にEOLを検出した場合	24＋56m＋3n＋mT
	先頭にFILL付きのEOLを検出した場合	

m：受信データ・バッファの指定された位置から最初に“1”を検出するまでのデータが，すべて“0”であるワード数

n：受信データ・バッファの指定された位置から最初に“1”を検出したワードの，“0”の連続ビット数＋1

例



T：受信データ・バッファが置かれている領域のウエイト数

(6) MHENC，MHDEC，MRENC，MRDEC命令

これらの命令については，処理パターンにより数値が異なるためクロック数は不定です。

〔メ　モ〕

第３章　V20，V30またはV25，V35に対する追加命令

V55PIの命令セットは，V20，V30またはV25，V35の命令セットと上位互換を持っています。

V20，V30またはV25，V35に対し，追加された命令，および適用範囲が拡張された命令を次に示します。

なお，追加された命令のうち，キュー操作命令，コーデック命令については，**第４章　キュー操作命令**と**第５章　コーデック命令**を参照してください。

○**IRAM**：……**レジスタ・ファイル空間アクセス用オーバライド・プリフィクス命令（追加）**

メモリ操作命令に付加することによりメイン・メモリ空間と切り離して，512バイトのレジスタ・ファイル空間に対しデータ・アクセスします。

レジスタ・ファイル空間のアドレスは，メモリ指定オペランドのオフセット値の下位９ビットとなります。

ニモニック	オペランド
IRAM：	なし

○**DS2**　：……**拡張セグメント・オーバライド・プリフィクス命令（追加）**

16Mバイトの拡張メモリ空間に対しデータ・アクセスする場合に，セグメント・オーバライド・プリフィクス可能なメモリ・オペランドに付加してセグメント・レジスタDS2を指定します。

ニモニック	オペランド
DS2：	なし

○**DS3**　：……**拡張セグメント・オーバライド・プリフィクス命令（追加）**

16Mバイトの拡張メモリ空間に対しデータ・アクセスする場合に，セグメント・オーバライド・プリフィクス可能なメモリ・オペランドに付加してセグメント・レジスタDS3を指定します。

ニモニック	オペランド
DS3：	なし

○**MOV DS2，reg16，mem32**

……32ビット・メモリから16ビット・レジスタとDS2への転送命令（追加）

第3オペランドで示される32ビット・メモリの下位16ビットを第2オペランドで指定される16ビット・レジスタに，上位16ビットを拡張セグメント・レジスタDS2に転送します。

ニモニック	オペランド		
MOV	DS2	reg16	mem32

○**MOV DS3，reg16，mem32**

……32ビット・メモリから16ビット・レジスタとDS3への転送命令（追加）

第3オペランドで示される32ビット・メモリの下位16ビットを第2オペランドで指定される16ビット・レジスタに，上位16ビットを拡張セグメント・レジスタDS3に転送します。

ニモニック	オペランド		
MOV	DS3	reg16	mem32

○**PUSH DS2**

PUSH DS3

PUSH VPC[注]

……拡張セグメント・レジスタのスタック操作命令（追加）

(SP－1，SP－2) ← DS2/DS3/VPC

SP ← SP－2

オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）をスタックに退避します。

ニモニック	オペランド
PUSH	DS2
PUSH	DS3
PUSH	VPC

注　VPCはベクタPCを意味します。VPCはDS3と同一のアドレスです。

保守／廃止

○**POP DS2**

POP DS3

POP VPC

……拡張セグメント・レジスタのスタック操作命令（追加）

DS2/DS3/VPC ← （SP＋1，SP）

SP ← SP＋2

オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）にスタックの内容を転送します。

なお，この命令と次の命令との間では，割り込みは受け付けません。

ニモニック	オペランド
POP	DS2
POP	DS3
POP	VPC

○**RSTWDT imm8，imm8'……ウォッチドッグ・タイマの操作命令（追加）**

4バイト目の命令コードと3バイト目の命令コードの比較を行い，ウォッチドッグ・タイマを操作します。この命令はプログラムの暴走などで誤ってレジスタの書き換えが行われないようにするため，特殊な命令コード構成（4バイト）を採用しており，3バイト目と4バイト目のオペコードが1の補数の関係でなければ書き込みが行われません。

ニモニック	オペランド	
RSTWDT	imm8	imm8'

○**BTCLRL sfr1，imm3，short-label……条件付きブランチ命令（追加）**

特殊機能レジスタ空間の下位256バイト（0FFE00H-0FFEFFH）内にある，指定されたレジスタのimm3で示されるビットがセット(1)されているとき，そのビットをリセット(0)して，現在のPCの値にshort-labelの値を加え，PCにロードします。

この命令の置かれているセグメント内で－128～＋127バイトのアドレス内にブランチできます。

この命令はBTCLR命令と対になっており，BTCLR命令が特殊機能レジスタ空間の上位240バイト（0FFF00H-0FFFEFH）を対象とするのに対し，BTCLRL命令は同空間の下位256バイトを対象とします。これ以外の機能は，BTCLR命令と同じです。

ニモニック	オペランド		
BTCLRL	sfr1	imm3	short-label

保守／廃止

○**MOV xsreg，reg16**

MOV VPC，reg16

……レジスタから，拡張セグメント・レジスタまたはベクタPCへのデータ転送命令(追加)

拡張セグメント・レジスタ（DS2，DS3/VPC）が追加されたため，オペランドに指定できるレジスタが追加されています。

第1オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）に第2オペランドで指定される16ビット・レジスタの内容を転送します。

ニモニック	オペランド	
MOV	DS2	reg16
MOV	DS3	reg16
MOV	VPC	reg16

○**MOV xsreg，mem16**

MOV VPC，mem16

……メモリから，拡張セグメント・レジスタまたはベクタPCへのデータ転送命令（追加）

拡張セグメント・レジスタ（DS2，DS3/VPC）が追加されたため，オペランドに指定できるレジスタが追加されています。

第1オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）に第2オペランドで指定される16ビット・メモリの内容を転送します。

ニモニック	オペランド	
MOV	DS2	mem16
MOV	DS3	mem16
MOV	VPC	mem16

○**MOV reg16，xsreg**

MOV reg16，VPC

……拡張セグメント・レジスタまたはベクタPCからレジスタへのデータ転送命令（追加）

拡張セグメント・レジスタ（DS2，DS3/VPC）が追加されたため，オペランドに指定できるレジスタが追加されています。

第1オペランドで指定される16ビット・レジスタに第2オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）の内容を転送します。

ニモニック	オペランド	
MOV	reg16	DS2
MOV	reg16	DS3
MOV	reg16	VPC

○**MOV mem16，xsreg**

MOV mem16，VPC

……拡張セグメント・レジスタまたはベクタPCから，メモリへのデータ転送命令（追加）

拡張セグメント・レジスタ（DS2，DS3/VPC）が追加されたため，オペランドに指定できるレジスタが追加されています。

第１オペランドで指定される16ビット・メモリに第２オペランドで指定される拡張セグメント・レジスタ（DS2またはDS3/VPC）の内容を転送します。

ニモニック	オペランド	
MOV	mem16	DS2
MOV	mem16	DS3
MOV	mem16	VPC

○**BRKCS reg16……レジスタ・バンク切り替え命令（拡張）**

BRKCS命令の実行により，レジスタ・バンクが，オペランドに記述した16ビット・レジスタの内容の下位４ビットで示されるレジスタ・バンクに切り替わります。また，新しいレジスタ・バンク内にあらかじめストアしておいたPSとベクタPCから得られるアドレスに分岐します。

新しいレジスタ・バンクからの復帰にはRETRBI命令を使用します。

V25に比べ，切り替えられるレジスタ・バンクの数が増えています。

（V25，V35の同命令に対し，適用範囲が拡張されました。V20，V30に対しては追加命令です）

ニモニック	オペランド
BRKCS	reg16

○**TSKSW reg16……レジスタ・バンク切り替え命令（拡張）**

BRKCS命令と同様にレジスタ・バンクを切り替え，新しいレジスタ・バンク内にあらかじめストアしておいたPSとPC退避エリアから得られるアドレスに分岐します。V25に比べ，切り替えられるレジスタ・バンクの数が増えています。

（V25，V35の同命令に対し，適用範囲が拡張されました。V20，V30に対しては追加命令です）

ニモニック	オペランド
TSKSW	reg16

保守／廃止

○**MOVSPB reg16……SS，SPの転送命令（拡張）**

現在（切り替え前）のレジスタ・バンクのSS，SPの値を，オペランドに記述した16ビット・レジスタの内容の下位４ビットで示される切り替え先のレジスタ・バンクのSS，SPに転送します。V25に比べ，切り替え先のレジスタ・バンクの数が増えています。

（V25，V35の同命令に対し，適用範囲が拡張されました。V20，V30に対しては追加命令です）

ニモニック	オペランド
MOVSPB	reg16

○**BSCH　reg/mem**

……reg/memのビット０から順にビット７あるいはビット15まで“1”をサーチし，最初にサーチしたビット番号をCLに返します。サーチした結果“1”がない場合（reg/mem＝0の場合），“ZF”（Zero Flag：PSWのビット６）をセット（1）します。

ニモニック	オペランド
BSCH	reg/mem

備考　サーチした結果“1”があった場合，“ZF”はリセット（0）されます。

そのほかV20，V30に対して追加された命令として，次の５命令があります。

○BTCLR sfr，imm3，short-label

……特殊機能レジスタのビット・テスト命令（特殊機能レジスタ空間の上位240バイト（0FFF00H-0FFFEFH）を対象）

○MOVSPA……レジスタ・バンク切り替え前のSS，SPの値を，現在（切り替え後）のレジスタ・バンクのSS，SPへ転送する命令

○RETRBI……レジスタ・バンク切り替え割り込みからの復帰命令

○FINT ……割り込みコントローラに対し，割り込み処理の終了を示す命令

○STOP　……STOP状態への移行命令

詳細な機能説明については，**第２章　命　　令**の各命令説明を参照してください。

第4章　キュー操作命令

4.1　概　　要

V55PIは，３種類のキュー操作用の命令を持っています。

これらの命令は，それ自体がリアルタイムOS（RTOS）のシステム・コールではありません。システム・コールはユーザがこれらの命令を使用して独自に作成するものです。

キュー操作用命令は，キュー構造を限定するのみで，ほかの部分についてはすべてユーザが決定するものです。

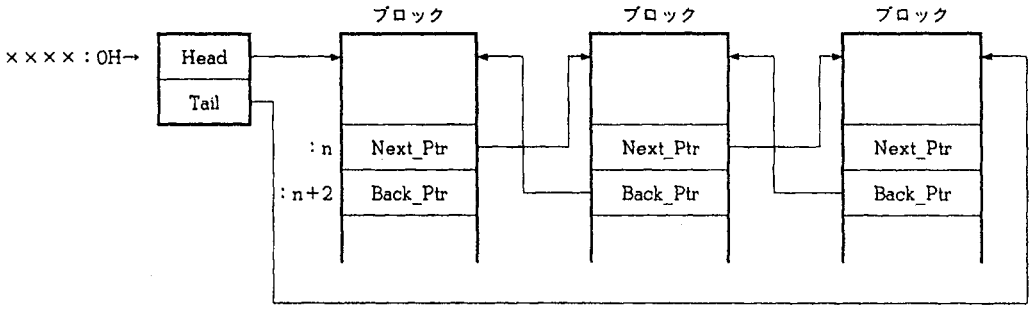
4.1.1　キュー構造

双方向のリンク・ポインタによるキュー構造をとります。

また，各ポインタにはセグメント値のみが格納されています。

キューイングされているブロックのリンク・ポインタの位置は自由に設定可能です。ただし，ブロック内にある，前後にリンクされるブロックを示すためのリンク・ポインタは，連続して置かれる必要があります。

図４－１　キュー構造例



●Head

キューイングされているブロックの先頭を示します(先頭ブロックのセグメント値)。キューが空の場合には，“0”を格納します。

●Tail

キューイングされているブロックの最後尾を示します（最後尾ブロックのセグメント値)。

●Next_Ptr

ブロックのリンク・ポインタで，次にリンクされているブロックのセグメント値を格納します。

●Back_Ptr

ブロックのリンク・ポインタで，前にリンクされているブロックのセグメント値を格納します。

保守／廃止

4.2　各命令の機能説明

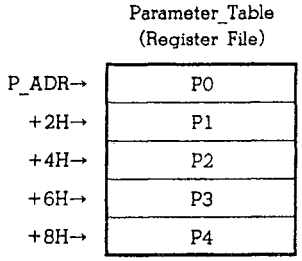
キュー操作用命令には，次の３種類があります。

○QHOUT
○QOUT
○QTIN

上記命令のうち，キュー操作命令のパラメータは，レジスタ・ファイル上の'P_ADR'でアドレスされるパラメータ・テーブルに設定します。

P_ADRにP0，P_ADR＋2HにP1，P_ADR＋4HにP2，P_ADR＋6HにP3，P_ADR＋8HにP4を設定し，必要なパラメータの設定後，命令を発行します(命令によってはP0-P4すべてのパラメータを設定する必要はありません)。

図４－２　パラメータ・テーブル



保守／廃止

4.2.1 QHOUT（Queue Head Out）命令

<記　述　形　式>

QHOUT　imm16

imm16：パラメータ・テーブル・アドレス（P_ADR）を指定

P_ADR→

P0
P1
P2
P3

P0：キュー・アドレスのオフセット値

P1：キュー・アドレスのセグメント値

P2：外したブロックのセグメント値を返す

P3：リンク・ポインタのオフセット値

<命　令　語　形　式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
imm16-low								imm16-high							

<バ　イ　ト　数>

4

<機　　能>

キューの先頭にキューイングされているブロックを外して，P2にそのセグメント・アドレスを格納します。指定されたキューが空の場合には，キューに対しては何の操作も行わず"ZF"（Zero Flag：PSWのビット6）をセット(1)します。それ以外の場合は，"ZF"をリセット(0)します。

<フラグの動作>

RB3	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	0	AC	0	P	$\overline{\text{IBRK}}$	CY
				U				U	×		U		U		U

図4－3　QHOUT命令実行例1

指定したキューにブロックが複数ある場合

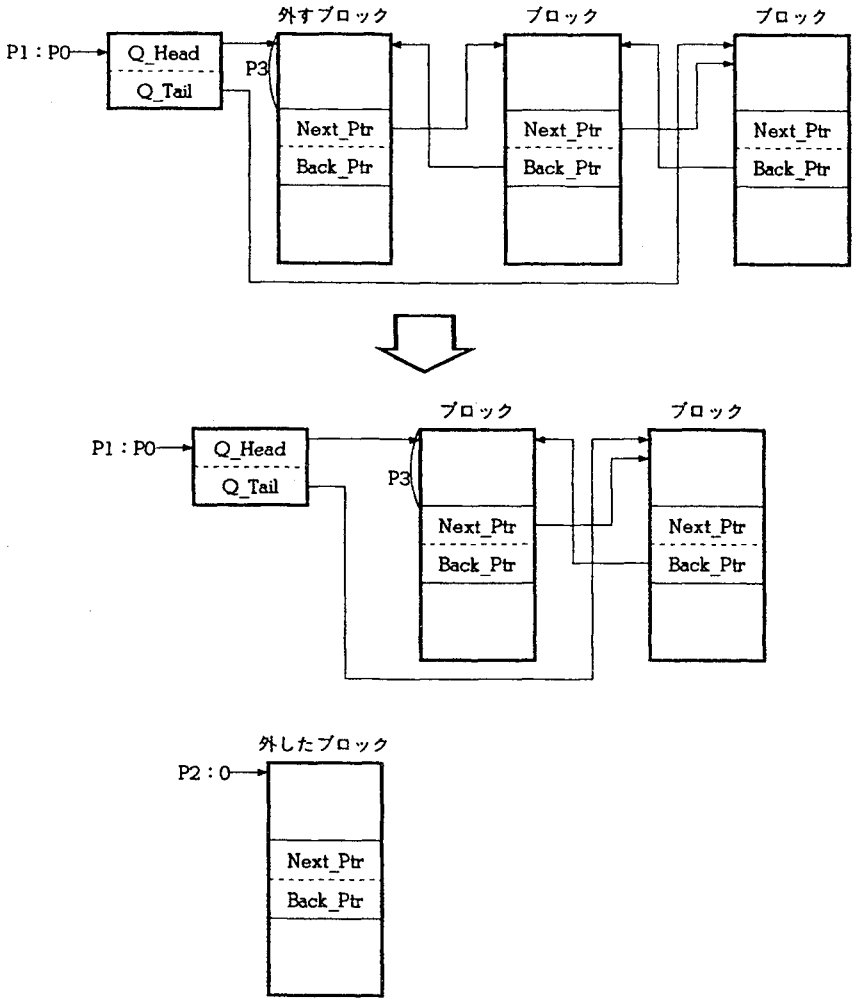
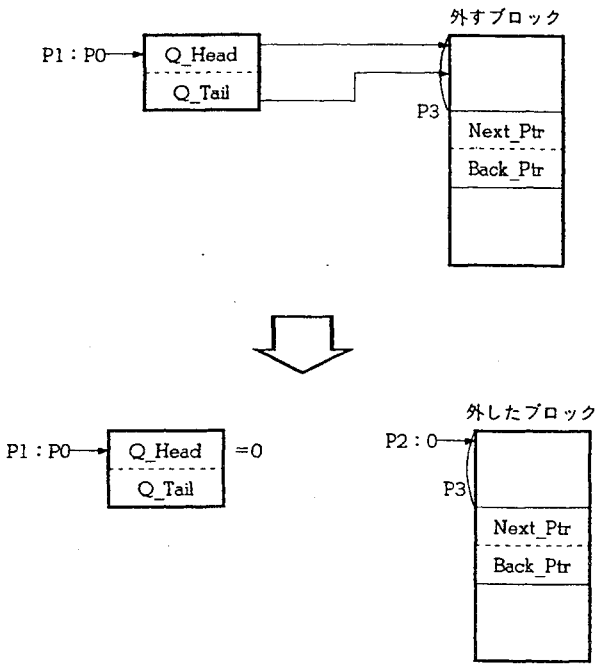


図4－4　QHOUT命令実行例2

指定したキューにブロックが1つだけある場合



保守／廃止

4.2.2 QOUT (Queue Out) 命令

<記 述 形 式>

QOUT　　imm16　　imm16：パラメータ・テーブル・アドレス（P_ADR）を指定

P_ADR→

P0
P1
P2
P3

P0：キュー・アドレスのオフセット値

P1：キュー・アドレスのセグメント値

P2：外すブロックのセグメント値設定

P3：リンク・ポインタのオフセット値

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	1

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
imm16-low								imm16-high							

<バ イ ト 数>

4

<機　　能>

キューのP2で示されるブロックを外します。

P0, P1で指定されたキューが空の場合には，キューに対しては何の操作も行わず，"ZF"(Zero Flag：PSWのビット６）をセット(1)します。それ以外の場合は，"ZF" をリセット(0)します。

<フラグの動作>

RB3	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	0	AC	0	P	$\overline{\text{IBRK}}$	CY
				U				U	×		U		U		U

図４－５　QOUT命令実行例１

指定したキューに複数ブロックがあり，先頭ブロックを外すように指定した場合（QHOUT命令と同様に処理されます）

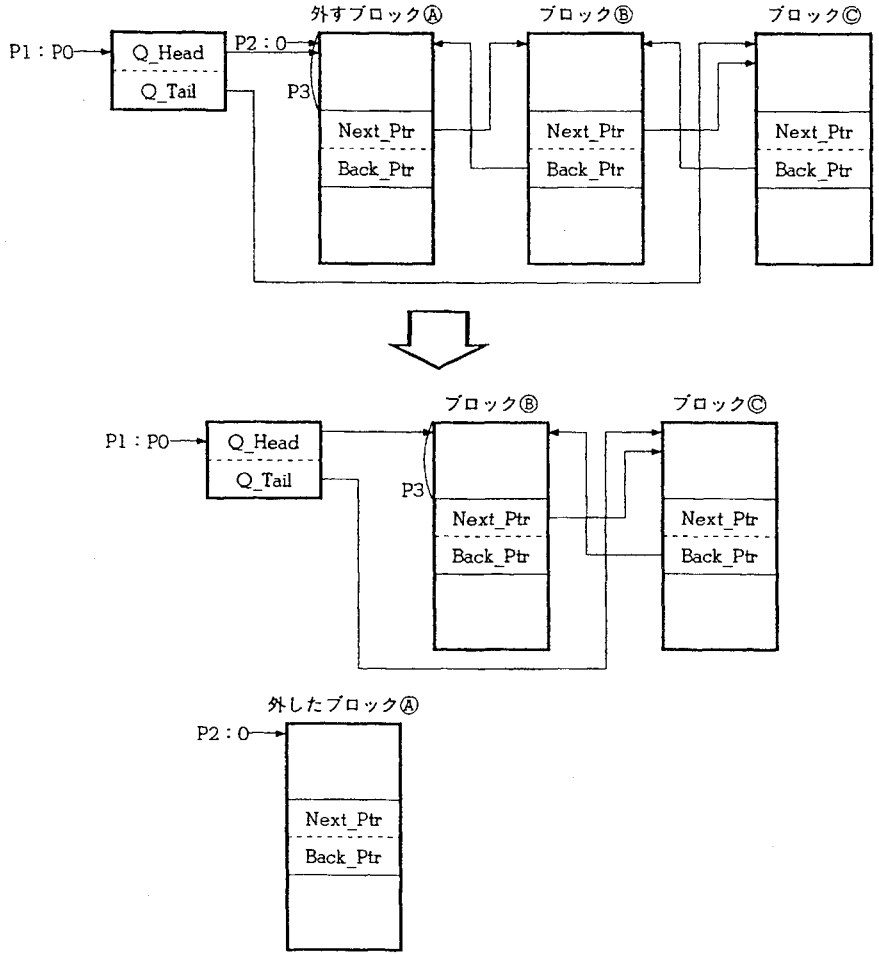


図４－６　QOUT命令実行例２

指定したキューに複数ブロックがあり，途中のブロックを外すように指定した場合

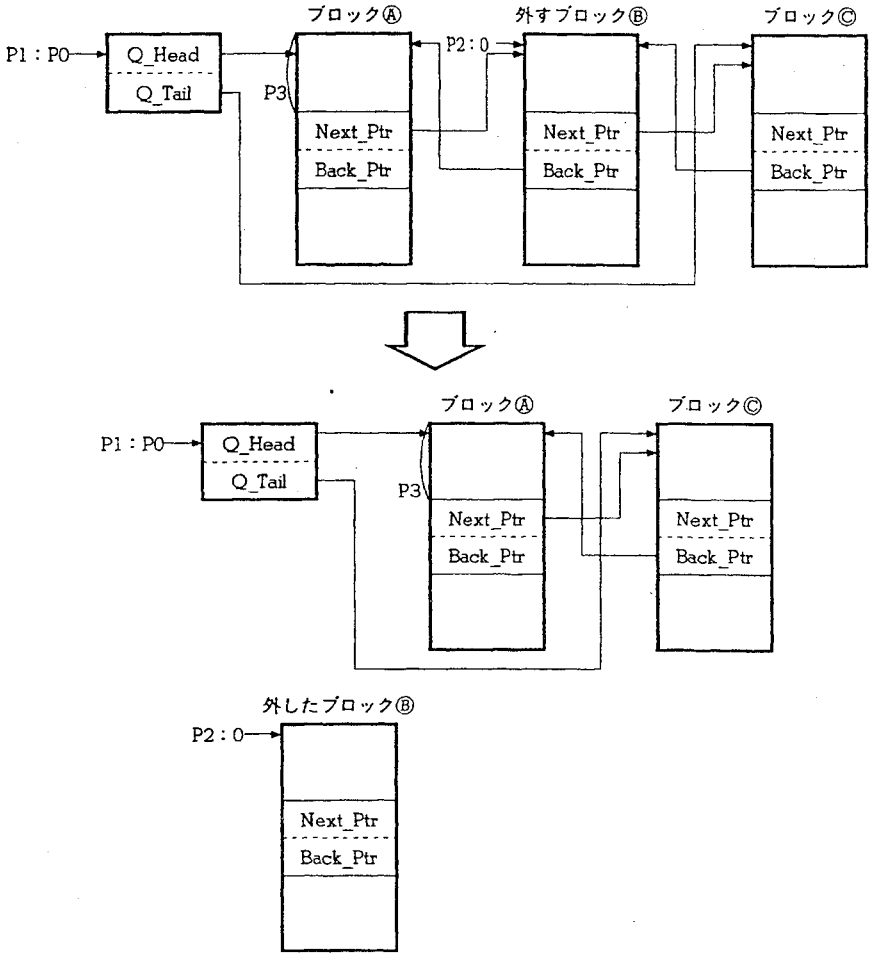


図４－７　QOUT命令実行例３

指定したキューに複数ブロックがあり，最後尾のブロックを外すように指定した場合

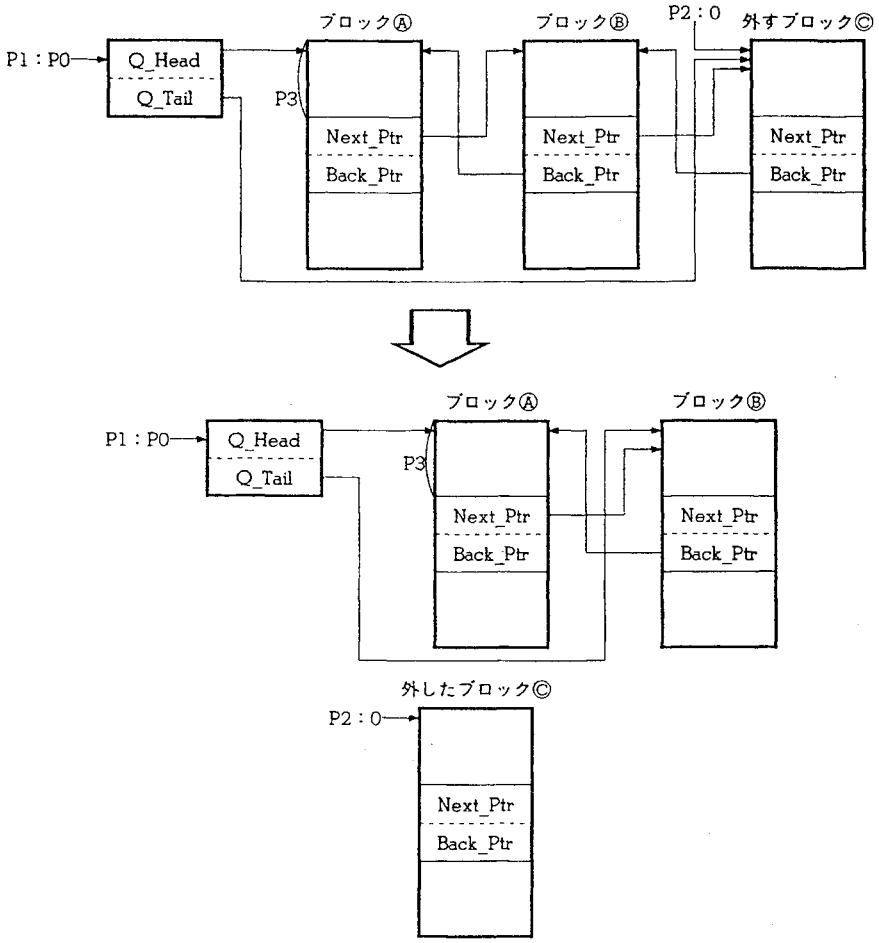
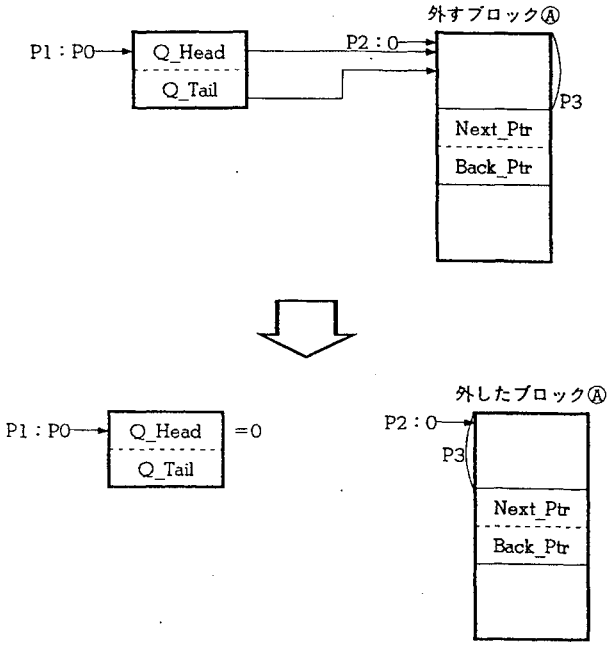


図４－８　QOUT命令実行例４

指定したキューにブロックが１つだけあり，そのブロックを外すように指定した場合



保守／廃止

4.2.3 QTIN（Queue Tail In）命令

<記　述　形　式>

QTIN　　imm16

imm16：パラメータ・テーブル・アドレス（P_ADR）を指定

P_ADR→

P0
P1
P2
P3

P0：キュー・アドレスのオフセット値

P1：キュー・アドレスのセグメント値

P2：キューイングするブロックのセグメント値設定

P3：リンク・ポインタのオフセット値

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	0	0	1	0

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
imm16-low								imm16-high							

<バ　イ　ト　数>

4

<機　　能>

P2で示されるブロックをキューの最後尾にキューイングします。

<フラグの動作>

なし

図４－９　QTIN命令実行例１

指定したキューに複数ブロックがある場合

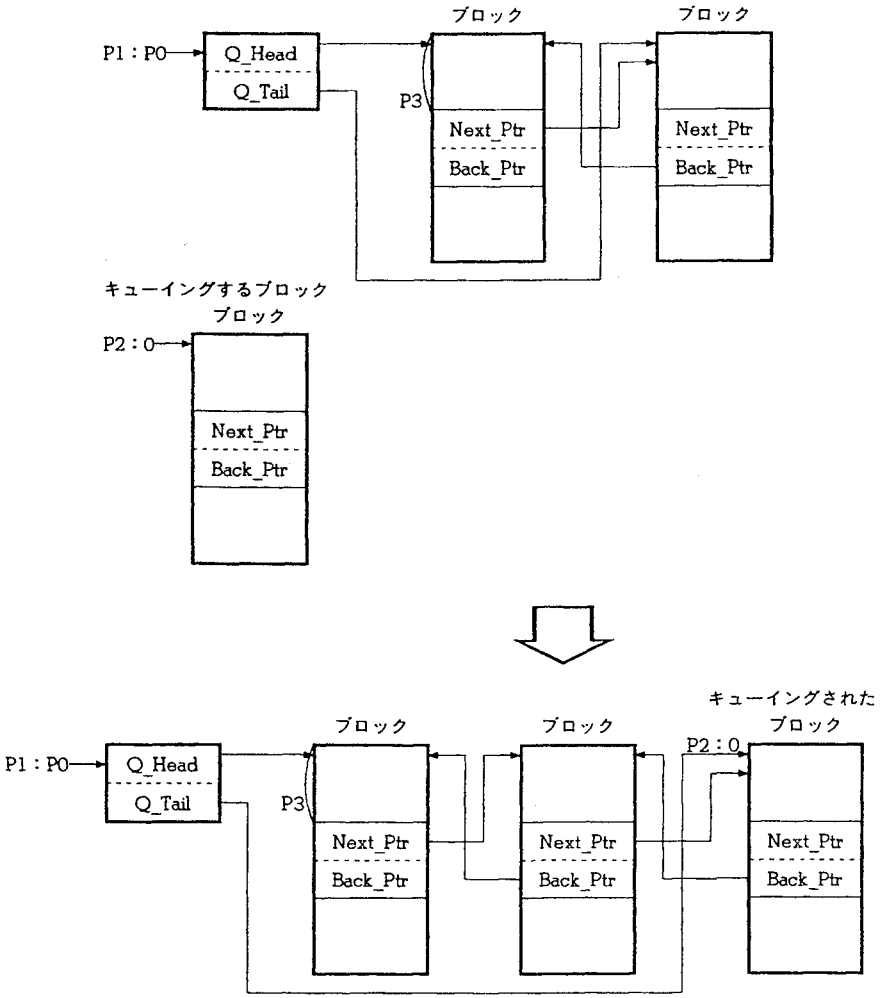
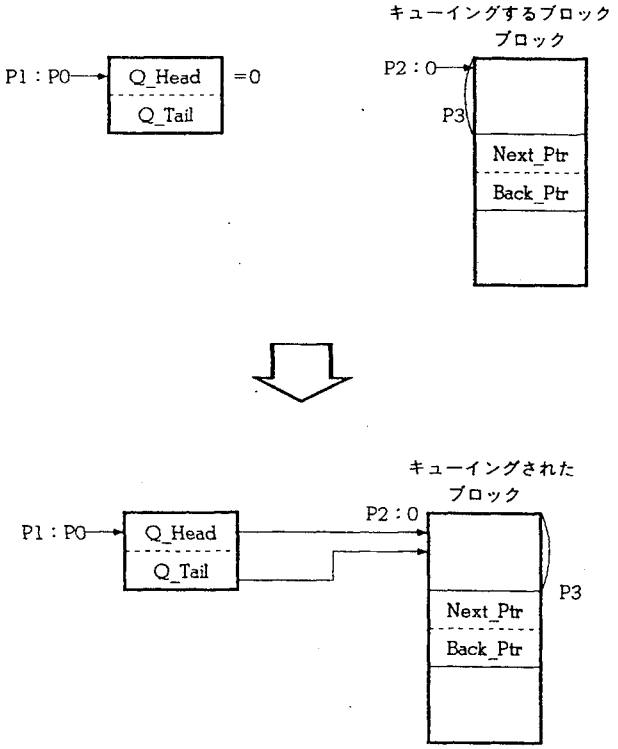


図４－１０　QTIN命令実行例２

指定したキューが空の場合



保守／廃止

〔メ　モ〕

第5章　コーデック命令

V55PIは，9種類のコーデック命令を持っています。

V55PIではこれらの命令を用いることで，2値画情報のMH[注1]符号化だけでなく，従来ACEE（アドバンスト・コンプレッション・エクスパンション・エンジン）などの専用デバイスを用いる必要のあったMR[注2]符号化を，小規模かつ高速なソフトウエア・コーデックにより実現できます。

注1. Modified Huffman
2. Modified Relative element designate

5.1 特　　徴

V55PIはコーデック命令として次に示す9命令（圧縮系：4，伸長系：5）を持っています。

○圧縮系命令
(1) 変化点テーブル作成命令：COLTRP
(2) データ送出命令（EOL[注1]，FILL，RTC[注2]などを送出）：ALBIT
(3) MH符号化命令：MHENC
(4) MR符号化命令：MRENC

○伸長系命令
(5) EOL検出命令：SCHEOL
(6) 1ビット（タグ）検出命令：GETBIT
(7) MH復号化変化点テーブル作成命令：MHDEC
(8) MR復号化変化点テーブル作成命令：MRDEC
(9) 画素データ作成命令：CNVTRP

このような命令を使用し，MH，MR符号化は図5-1のように，MH，MR復号化は図5-2のように行われます。

注1. EOL：End of Line
2. RTC：Return To Control

注意　V55PIのコーデック命令を用いて圧縮／伸長処理を行う場合，次のことを前提に指定してください。

保守／廃止

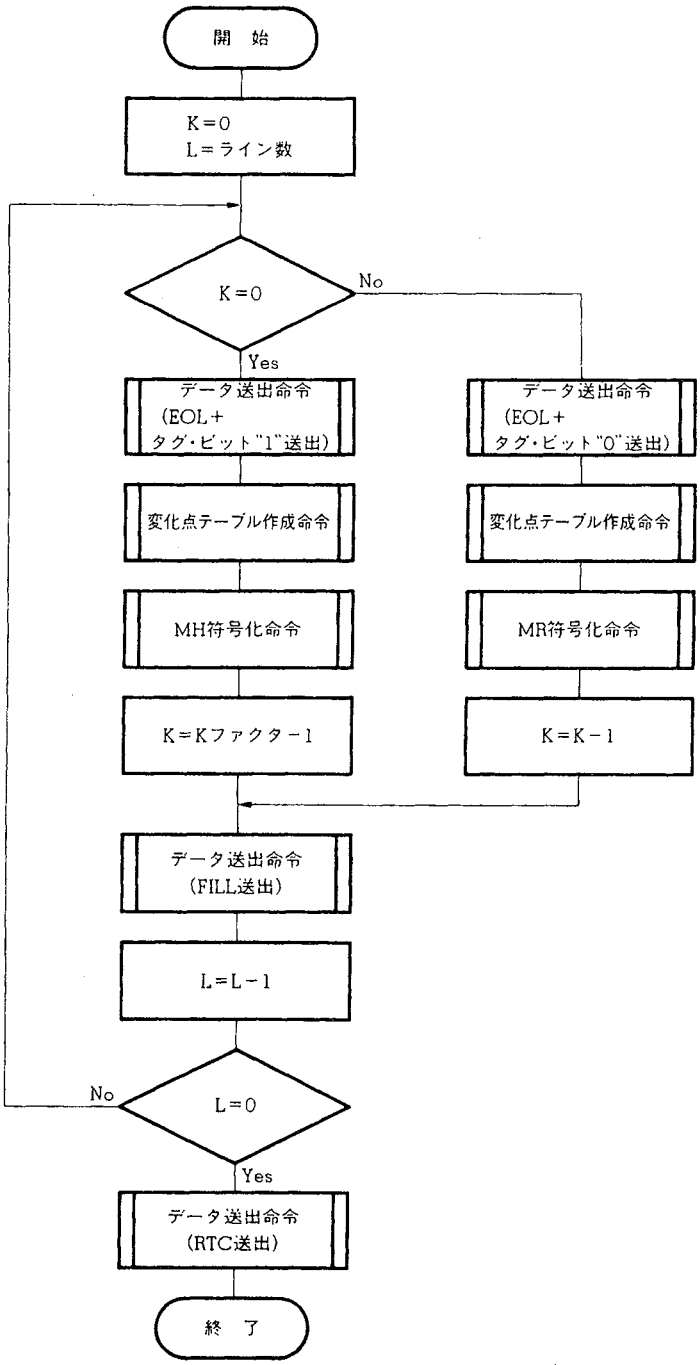
● 1ライン単位ごとに圧縮／伸長を行う。

●圧縮処理中にタスク切り替えおよび割り込みが発生することを考慮する。

● 1ライン処理ビット数が1ページ処理中変わらないようにする。

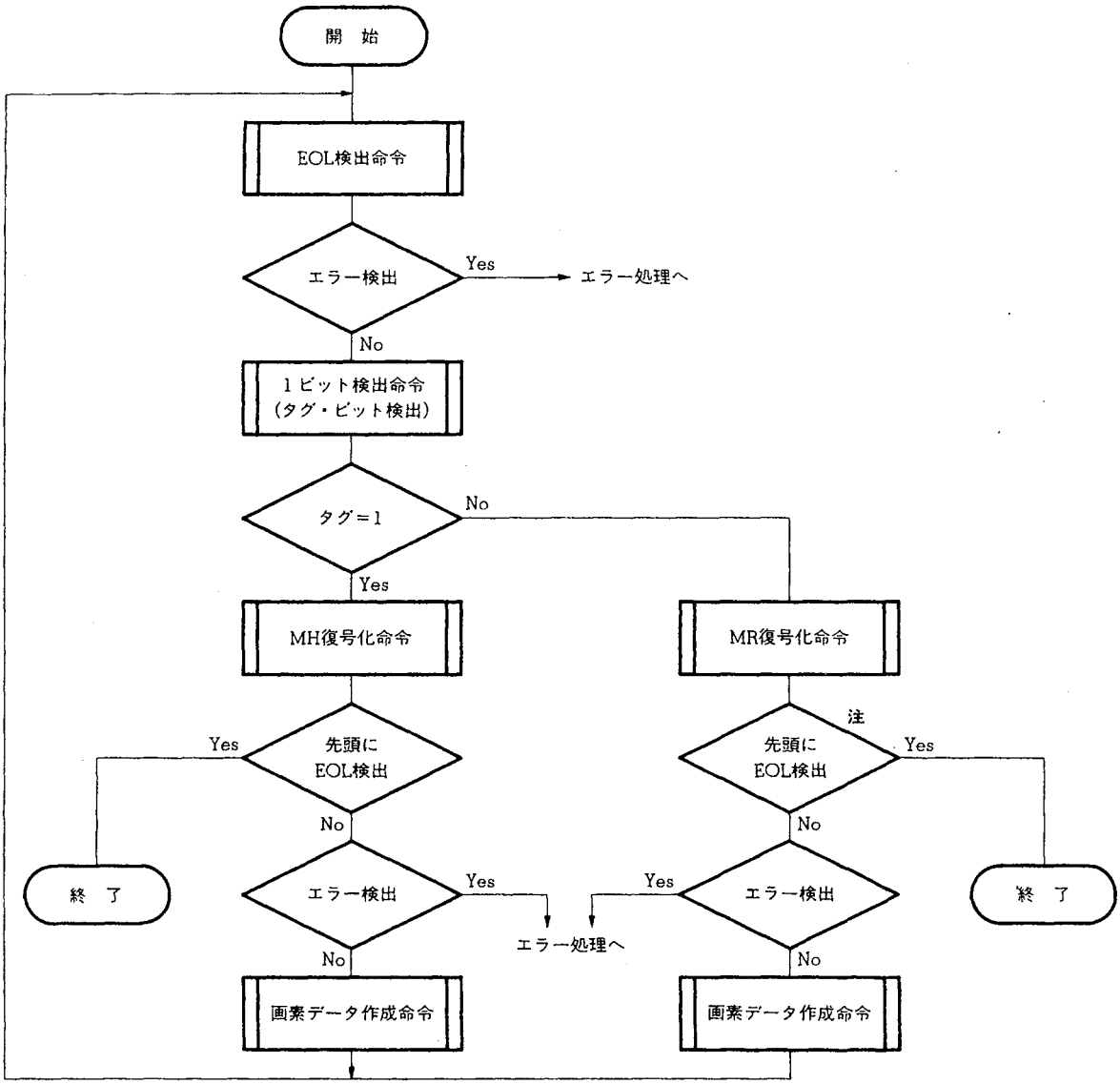
●セグメントをまたぐ64Kバイト以上のデータに対しては，途中でセグメント値を変更する。

図5－1　MH，MR符号化処理フロー



保守／廃止

図5-2　MH，MR復号化処理フロー



注　RTCは2個のEOLで検出しています。

保守／廃止

5.2　2値画情報の符号化／復号化方式

ここでは，2値画情報の符号化(圧縮)方式と復号化(伸長)方式について説明します。

2値画情報の符号化（圧縮）方式としては，ITU-T（国際電気通信連合）勧告に基づいて，MH方式，MR方式とMMR[注]方式が標準的に採用されています。復号化（伸長）は符号化と逆の手順で実現します。

ITU-T勧告についての詳細は「ITU-T　ホワイト・ブック　G3/G4ファクシミリ関連Tシリーズ勧告」（財団法人　新日本ITU協会刊）を参照してください。

注　Modified MR

各方式の説明に入る前に，その基本的な考え方であるラン・レングス符号化について説明します。

5.2.1　ラン・レングス符号化

いま，図5-3のような絵をスキャナなどで2値画情報として取り込んだ場合，メモリ上には図5-4のようなビット・マップ・データとして取り込まれます。つまり，白の画素は0で，黒の画素は1で表されます。

保守／廃止

図5-3　サンプル図

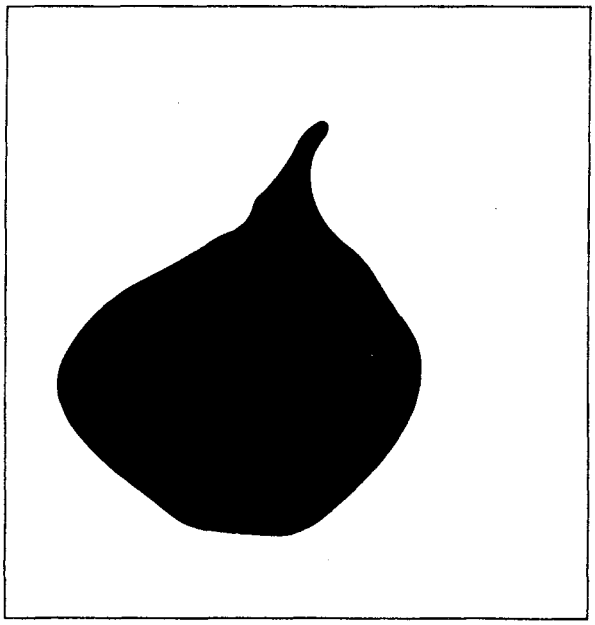
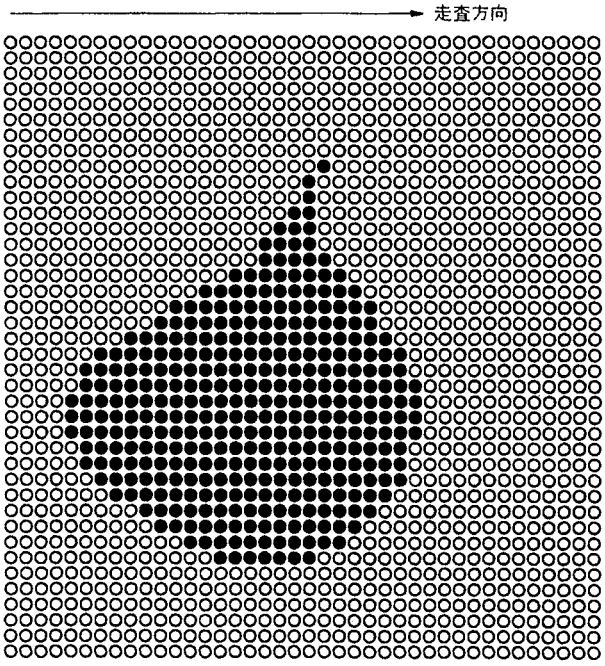


図5-4　ビット・マップ・データ



備考　○：ビット0

●：ビット1

これを図５－４の矢印の方向に走査していくと，初めは０のビットが連続していますが，あるところで０→１へ変化する点（変化点）にぶつかります。さらに，走査を進めると今度は１→０への変化点にぶつかります。ラン・レングス符号化とは，ひとつの変化点から次の変化点までの連続する同じ色（白か黒）の画素数（これをラン・レングスといいます）を何らかの方法で符号に置き換えるものです。

たとえば，符号としてラン・レングスそのものを使えば，図５－４のビット・マップ・データは図５－５のような数値列に置き換わります。元のデータ量は40×40＝1600ビット＝200バイトですが，符号化後は数十バイトに圧縮できます。スキャナの解像度を上げれば，この差はさらに大きくなります。

図５－５　ラン・レングス符号列

341 1 38 1 39 1 38 2 37 3 36 4 36 5 37 8 30 11 27 13 25 15 23 18 20 21 18
22 18 23 16 24 16 24 16 24 17 22 18 22 20 19 23 16 25 14 28 11 31 7 259

符号を復号して元の画像を再生するには，走査の開始点の色と画像の横サイズが分かれば符号化の逆の手順によって容易に実現できます。しかも，この再生画像は元の画像と完全に同じものです。

以上がラン・レングス符号化の考え方です。次項より説明するITU-T勧告の符号化方式は，このラン・レングス符号化が基礎となっています。

保守／廃止

5.2.2　MH方式

MH方式は一次元符号化方式とも呼ばれます。

この方式では，ラン・レングスを画像データの各水平ラインごとに計数します。各ラン・レングスに対する符号としては，表5-1のものを採用しています。符号にはメークアップ符号とターミネート符号の2種類があります。64個未満のラン・レングスに対してはターミネート符号が割り当てられ，64個以上のラン・レングスについてはメークアップ符号とターミネート符号の組み合わせが割り当てられます（例1，例2　参照）。

例1.　ラン・レングスが231の場合

231＝64×3＋39＝192＋39

192：メークアップ符号
39：ターミネート符号

2.　ラン・レングスが128の場合

128＝64×2＋0＝128＋0

128：メークアップ符号
0：ターミネート符号

保守／廃止

表5-1　ラン・レングス符号

(a)　ターミネート符号

ラン・レングス	白ラン用符号語	黒ラン用符号語	ラン・レングス	白ラン用符号語	黒ラン用符号語
0	00110101	0000110111	32	00011011	0000001101010
1	000111	010	33	00010010	0000001101011
2	0111	11	34	00010011	0000011010010
3	1000	10	35	00010100	0000011010011
4	1011	011	36	00010101	0000011010100
5	1100	0011	37	00010110	0000011010101
6	1110	0010	38	00010111	0000011010110
7	1111	00011	39	00101000	0000011010111
8	10011	000101	40	00101001	0000001101100
9	10100	000100	41	00101010	0000001101101
10	00111	0000100	42	00101011	0000011011010
11	01000	0000101	43	00101100	0000011011011
12	001000	0000111	44	00101101	0000010100100
13	000011	00000100	45	00000100	0000010101101
14	110100	00000111	46	00000101	0000010101110
15	110101	000011000	47	00001010	0000010101111
16	101010	0000010111	48	00001011	0000011001100
17	101011	0000011000	49	01010010	0000011001101
18	0100111	0000001000	50	01010011	0000010100010
19	0001100	00001100111	51	01010100	0000010100011
20	0001000	00001101000	52	01010101	0000001001000
21	0010111	00001101100	53	00100100	0000001101111
22	0000011	00000110111	54	00100101	0000001110000
23	0000100	00000101000	55	01011000	0000001001111
24	0101000	00000010111	56	01011001	0000001010000
25	0101011	00000011000	57	01011010	0000010110000
26	0010011	0000011001010	58	01011011	0000010110001
27	0100100	0000011001011	59	01001010	0000001010111
28	0011000	0000011001100	60	01001011	0000001011000
29	00000010	0000011001101	61	00110010	0000010110010
30	00000011	0000011010000	62	00110011	0000011001110
31	00011010	0000011010001	63	00110100	0000011001111

(b)　メークアップ符号 (1/2)

ラン・レングス	白ラン用符号語	黒ラン用符号語	ラン・レングス	白ラン用符号語	黒ラン用符号語
64	11011	0000001111	960	011010100	0000001110011
128	10010	0000011001000	1024	011010101	0000001110100
192	010111	0000011001001	1088	011010110	0000001110101
256	0110111	0000001011011	1152	011010111	0000001110110
320	00110110	0000000110011	1216	011011000	0000001110111
384	00110111	0000000110100	1280	011011001	0000001010010
448	01100100	0000000110101	1344	011011010	0000001010011
512	01100101	0000001101100	1408	011011011	0000001010100
576	01101000	0000001101101	1472	010011000	0000001010101
640	01100111	0000001001010	1536	010011001	0000001011010
704	011001100	0000001001011	1600	010011010	0000001011011
768	011001101	0000001001100	1664	011000	0000001100100
832	011010010	0000001001101	1728	010011011	0000001100101
896	011010011	0000001110010	EOL	000000000001	000000000001

保守／廃止

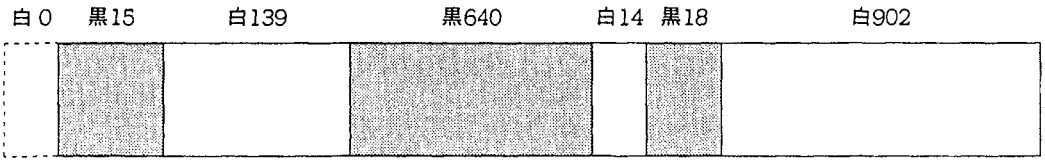
(b)　メークアップ符号 (2/2)

ラン・レングス	符号語（白ラン，黒ラン共通）	ラン・レングス	符号語（白ラン，黒ラン共通）
1792	00000001000	2240	000000010110
1856	00000001100	2304	000000010111
1920	00000001101	2368	000000011100
1984	000000010010	2432	000000011101
2048	000000010011	2496	000000011110
2112	000000010100	2560	000000011111
2176	000000010101		

図5-6に1ライン分の符号化の例を示します。ラインの先頭画素が黒で始まる場合には，その直前に0個の仮想的な白画素があるものとして符号化します。

図5-6　1ライン分の符号化例

画像データ



計 1728ビット

符号データ

白0	黒15	白128	白11	黒640	黒0	白14	黒18	白896	白6
00110101	000011000	10010	01000	0000001001010	000110111	110100	0000001000	011010011	1110

計 78ビット

1ページ分の符号化は次の手順で行います。

① 最初に符号表中のEOL（End Of Line）符号を出力します。

② 1ラインを符号化します。

③ 必要ならば，最小伝送時間[注]を保証するために複数個の0から成るフィル・ビットを付加します。

④ EOL符号を付加します。

⑤ ①から④を繰り返します。

⑥ 1ページの最終ラインの符号（またはそれに続くフィル・ビット）の後ろにRTC（Return To Control）符号を付加してページの終端を示します。ただし，

RTC＝EOL×6

とします。

保守／廃止

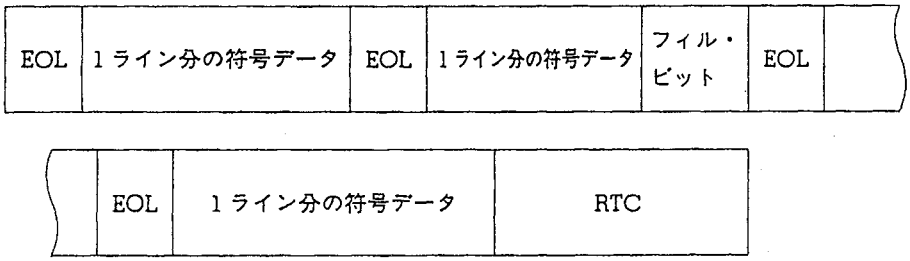
注　最小伝送時間

ファクシミリなどにおいては，受信側の受信許容速度がまちまちです。受信許容速度は主として，受信バッファの大きさやプリント・アウトの速度に依存して決まります。符号データの量は画像データのパターンに依存し，送信速度は符号データの量に依存します。したがって，この受信許容速度が遅いにもかかわらず短いライン符号が次々と送られてきた場合には，1ライン分の画像を出力し終わらないうちに次のライン符号を受信しなければならなくなり，データの取りこぼしが生じることになります。

このため，1ライン分の符号データを受信できる最小の時間（最小伝送時間）が機種ごとに決まっています。送信側が1ライン分の符号データを送信する場合に，この時間を守れないときはフィル・ビットを挿入して調整してください。また，送り先の最小伝送時間は符号データの送信を開始する前に，それぞれの機器間で連絡をとることになっています。

この手順で符号化することにより，1ページ分の画像データは図5-7に示すようになります。

図5-7　1ページ分のMH符号



備考　RTC＝EOL×6

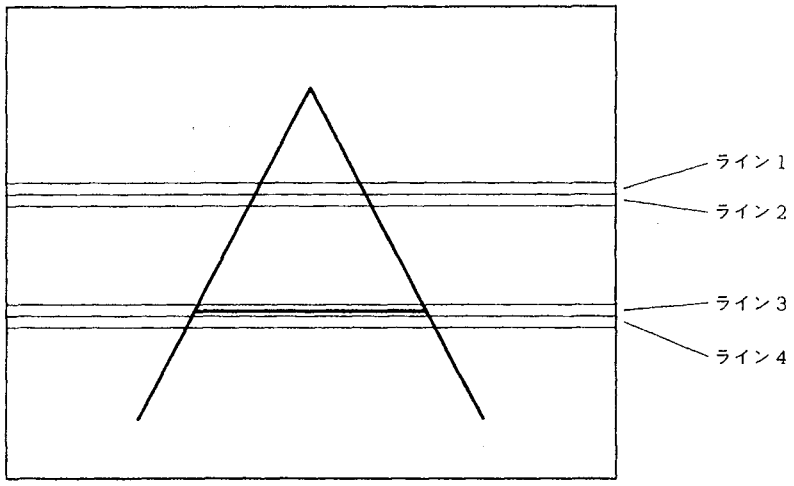
保守／廃止

5.2.3　MR方式

MR方式は，画像データの垂直方向の相関に着目して，符号化するラインを１つ手前のラインと比較しながら符号化を進める方式です。

文字や図形などが描かれた文書画像では基本的に縦，横，斜めの線で画像が構成されています。したがって，図５-８のライン１とライン２のように，あるラインについて特定の画像パターンが現れたならば，その上下のラインについても同じようなパターンが現れる頻度が高いと言えます。同様に，図５-８のライン３とライン４のように，上下のラインでまったく違う画像パターンが現れる頻度も高いと言えます。そこで，このような２つの場合に対して短い符号を割り当てることによって，MH方式以上の圧縮率が期待できます。

図５-８　画像の垂直方向の相関



この方式の説明をする前に説明で使用する用語と記号の定義をします。

符号化するラインの１つ手前のラインを参照ラインと呼びます。符号化するライン上と参照ライン上の変化点に対して，図５-９のようにa1，a2，b1，b2のレーベルを付けます。この４個の画素の位置関係から３つのモードを考えて符号化します。a0は基準点と呼ばれる走査を開始する画素です。5個の画素は次のように定義されています。

① a0：各ラインを符号化する際に，ラインの最初の画素の直前に仮想的な白画素があるものとして，ここに置かれます。a0を開始点として１ラインのある範囲の符号化が終了すると，符号化された領域の後ろに新たにa0の位置が決定されます。

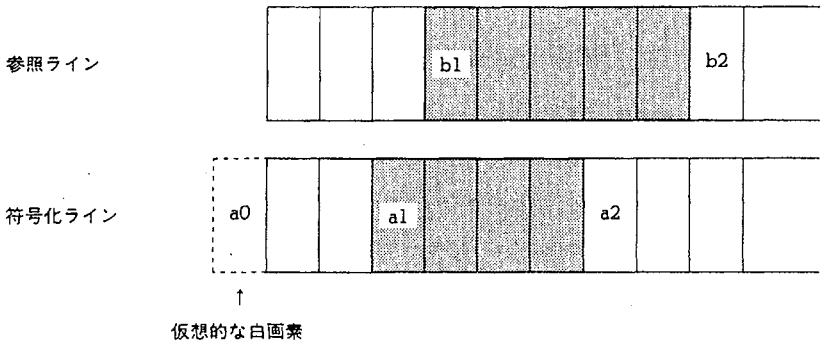
② a1：符号化するライン上で，a0の次に現れる変化点に置かれます。

③ a2：符号化するライン上で，a1の次に現れる変化点に置かれます。

④ b1：参照ライン上で，a0の右に現れる，a0の画素の色と反対色の最初の画素に置かれます。

⑤ b2：参照ライン上で，b1の次に現れる変化点に置かれます。

図５－９　符号化の基準となる点



符号化のモードとしては，次の３つがあります。

(1)　パス・モード

(2)　垂直モード

(3)　水平モード

それぞれのモードの詳細は次のとおりです。

(1)　パス・モード

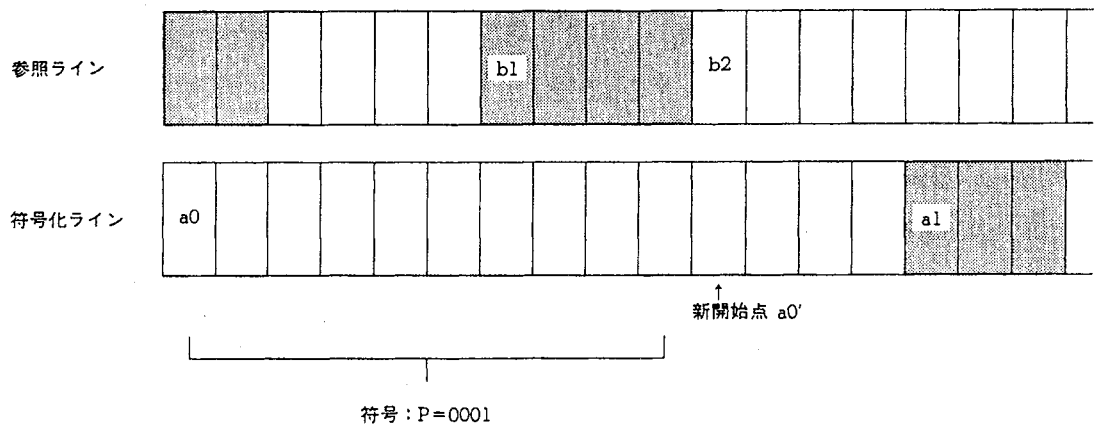
パス・モードは，b2がa1の左側にある場合に対応します。このことは，符号化するライン上でのa0の次の変化点が，b2に対応する位置までは存在しないことを意味します（図５－10）。図５－８のライン３とライン４のような組み合わせで起こります。

具体的な符号化は，次の手順で行います。

①　符号としてパス・モード符号P（＝0001）を出力します。

②　b2に対応する位置に新基準点a0′を設定します。a0′は以後の符号化において，a0となります。

図５－10　パス・モードの例



(2) 垂直モード

垂直モードは，a1がb1の左右計7画素以内の範囲にある場合に対応します（図5-11）。図5-8のライン1とライン2のような組み合わせで起こります。

具体的な符号化は，次の手順で行います。

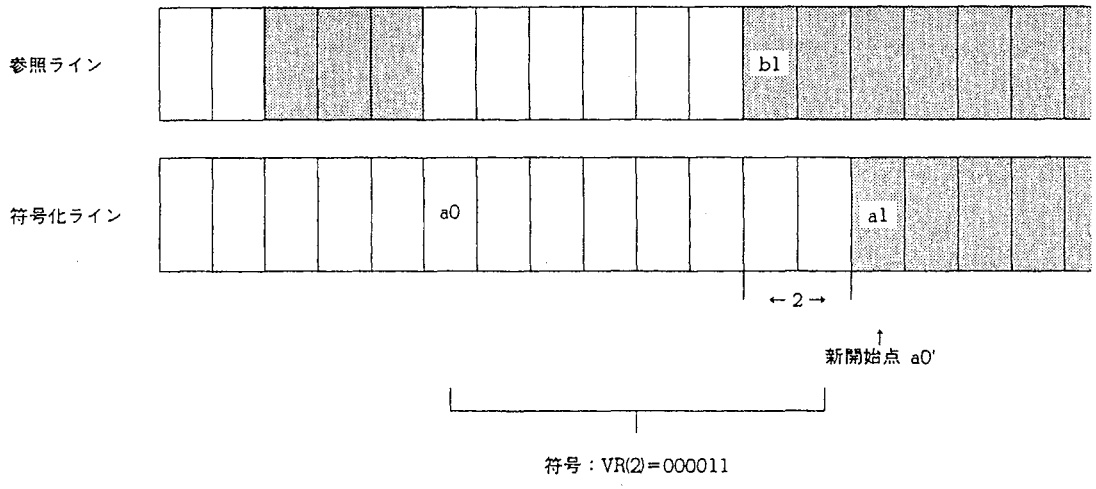
① 符号として，b1に対するa1の相対的な位置（右側か左側）および距離（画素数）に着目して表に従った符号を出力します（表5-2）。

② a1の位置に新基準点a0'を設定します。a0'は以後の符号化において，a0となります。

表5-2　垂直モードの符号

種類	a1とb1との位置関係	符号
V(0)	a1がb1の真下にある	1
VL(1)	a1がb1から1画素分左にある	010
VL(2)	a1がb1から2画素分左にある	000010
VL(3)	a1がb1から3画素分左にある	0000010
VR(1)	a1がb1から1画素分右にある	011
VR(2)	a1がb1から2画素分右にある	000011
VR(3)	a1がb1から3画素分右にある	0000011

図5-11　垂直モードの例（a1がb1の右側2画素目にある場合）



保守／廃止

(3) 水平モード

パス・モードまたは垂直モード以外の場合，つまりb2がa1の右側にあり，かつa1とb1の距離が3画素を越える場合に対応します（図5-12）。

具体的な符号化は，次の手順で行います。

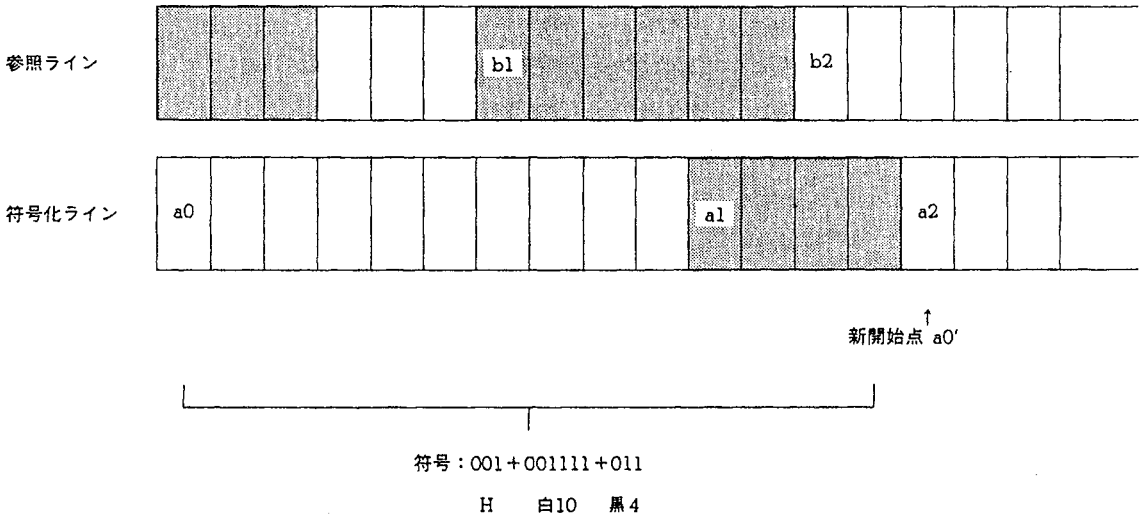
① 符号として，まず水平モード符号H（=001）を出力します。

② a0からa1までのラン・レングスをMH方式の符号表（表5-1）に従って符号化します。

③ a1からa2までのラン・レングスをMH方式の符号表（表5-1）に従って符号化します。

④ a2の位置に新基準点a0'を設定します。a0'は以後の符号化において，a0となります。

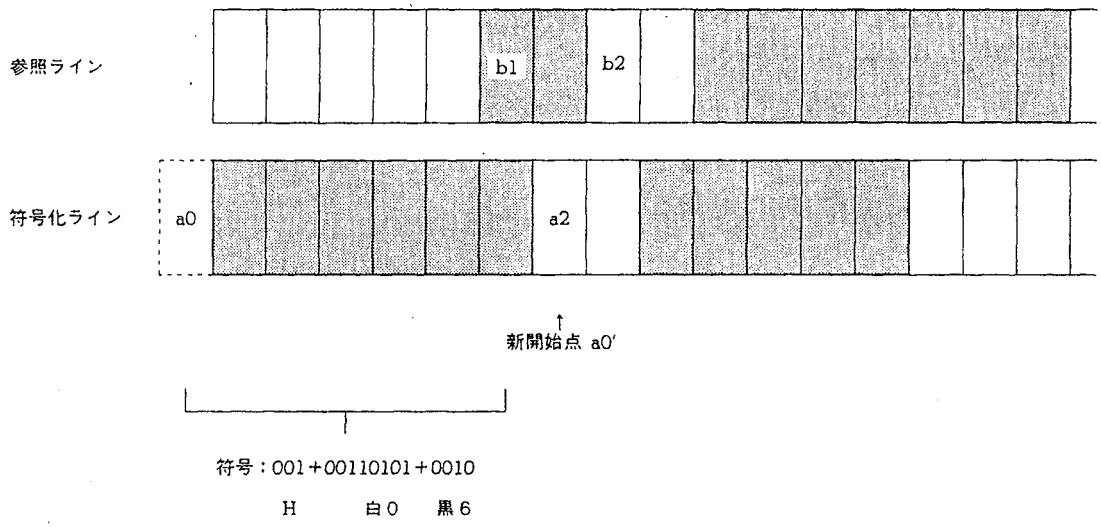
図5-12　水平モードの例



各ラインの始端と終端ではa0-a2，b1-b2の関係が特殊なものとなるため特別な処理が必要です。

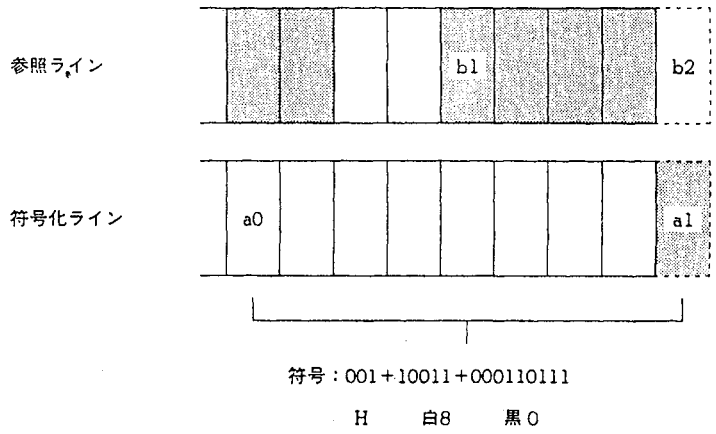
ライン始端では，先頭の画素が黒の場合，ラインの最初の画素の直前に０個の仮想的な白画素があるものとして，a0がここに置かれます。したがって，図５-13のようにラインが黒画素で始まっているような場合，符号はHに続いてラン・レングス０の符号が来ることになります。

図５-13　ライン始端の処理例



ライン終端では，最後の画素の直後にそれと反対色の仮想的な画素が０個続くものとして，仮想画素までを符号化します（図５-14）。

図５-14　ライン終端の処理例



以上の手順を二次元符号化方式と呼びます。

１ページ分の符号化はKというパラメータを使って一次元符号化と二次元符号化の組み合わせで行います（図５-15）。

具体的な符号化は，次の手順で行います。

① 第１ライン目を一次元符号化します。

② 第２ライン目以降を二次元符号化します。

③ K＋１ライン目を一次元符号化します。

④ ①から③の手順を繰り返します。

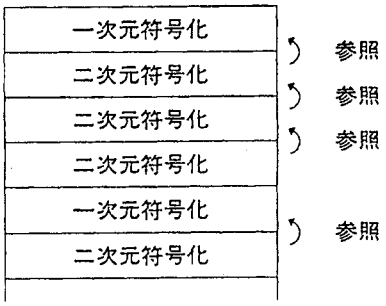
つまり，Kラインごとに一次元符号化方式で符号化します。これは次の理由によります。二次元符号化方式は直前のラインを参照して符号化します。このため，あるラインの符号が伝送誤りなどで正常な符号でなくなった場合，このライン以降のすべての復元画像が正常なものでなくなります。これを防ぐために，Kラインごとに一次元符号化を挿入します。

Kの値は，垂直方向の解像度により，一般的に次のようになります。

・100LPI（Lines Per Inch）の場合，K＝2

・200LPI（Lines Per Inch）の場合，K＝4

図５-15　Kパラメータ（K＝4の場合）



一次元符号化したラインと二次元符号化したラインとを区別するため，ライン符号の区切り符号を付加します。

・一次元符号の直前：EOL＋1

・二次元符号の直前：EOL＋0

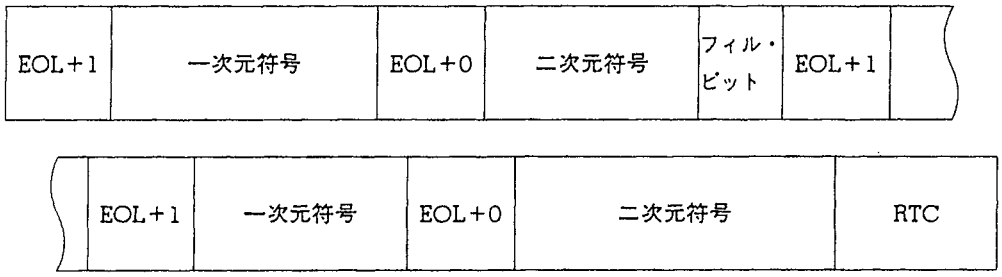
また，ページ終端には次の符号を付加します。

RTC＝(EOL＋1)×6

この手順で符号化することにより，１ページ分の画像データは図５-16に示すようになります。

保守／廃止

図5-16　1ページ分のMR符号



備考　RTC＝(EOL+1)×6

保守／廃止

5.2.4　MMR方式

MMR方式では，すべてのラインを二次元符号化します。ページの第１ライン目はその直前に仮想的な全白のラインがあるものとして，二次元符号化されます。

MR方式では，符号データを通常の電話回線で伝送することを前提として規定されているため，伝送誤りの伝播を制限するためにKパラメータが用いられています。これに対して，MMR方式はデジタル回線を用いた誤りのない伝送を前提としているため，Kパラメータを無限大に設定しています。

１ページ分の符号化は次の手順で行います。

① ページの第１ライン目はその直前に仮想的な全白のラインがあるものとして，第１ライン目からすべてのラインを二次元符号化します。各ライン符号間を区切る特別な符号は用いません。

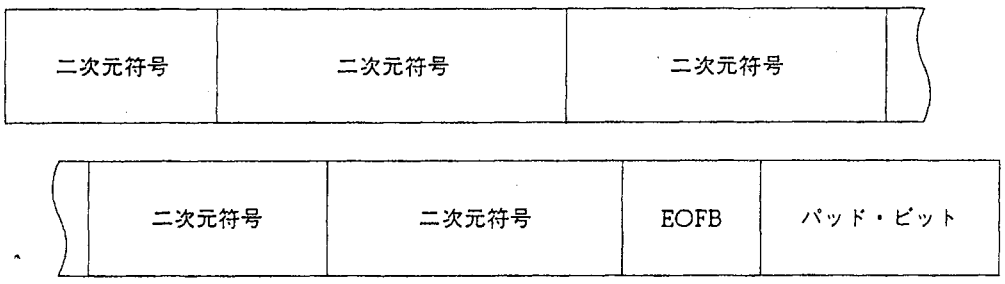
② ページ終端を示す符号として，最終ライン符号の直後にEOFB（End Of Facsimile Block）符号を付加します。ただし，EOFB符号は次の値とします。

EOFB＝EOL×2

③ １ページ分の符号データ量をバイト単位にそろえる必要がある場合，またはある大きさ以上にそろえる必要がある場合には，EOFB符号の後ろにパッド・ビットとして０のビット列を付加します。

この手順で符号化することにより，１ページ分の画像データは図５-17に示すようになります。

図５-17　１ページ分のMMR符号



備考　EOFB＝EOL×２

保守／廃止

5.3　メモリ・マップ

V55PIのコーデック命令が必要とするデータ用メモリ領域は次のとおりです。

(1) レジスタ・ファイル空間

パラメータ設定用レジスタ・バンクです。

パラメータの設定についての詳細は，**5.6　各命令の機能説明**を参照してください。

(2) ユーザRAM

符号化ライン変化点テーブル：符号化を行うために必要な変化点情報格納領域
（nビット／ラインの場合，最大2n＋4バイトの領域が必要）

参照ライン・テーブル　：参照ラインの変化点情報格納領域

画素データ・バッファ　：符号化の場合には，スキャナから読み込まれた画素データ，復号化の場合には，モデムから受信された符号化データの格納領域

送信／受信バッファ　：モデム／スキャナに，符号化されたデータを引き渡すためのバッファ

プリント・バッファ　：記録系に，復号化された画素データを引き渡すためのバッファ

(3) ユーザROM

符号化変換テーブル　：MH，MR符号化のための変換テーブル

復号化変換テーブル　：MH，MR復号化のための変換テーブル

(4) 拡張メモリ空間へのアクセス

拡張セグメント・オーバライド・プリフィクス命令（DS2：またはDS3：）を使用して16Mバイト拡張メモリ空間をアクセスできます。

ただし，命令の実行時に使用するセグメント・レジスタDS2，DS3は命令ごとのパラメータ設定用レジスタ・バンク内のDS2，DS3を使用して行います。

保守／廃止

表5－3　拡張セグメント・オーバライド・プリフィクスの付加可能なコーデック命令

DS2：	DS3：	コーデック命令
可	可	COLTRP
可	不可	MHENC
可	可	MHDEC
可	不可	MRENC
可	可	MRDEC
可	不可	SCHEOL
可	不可	GETBIT
可	可	CNVTRP

記述例

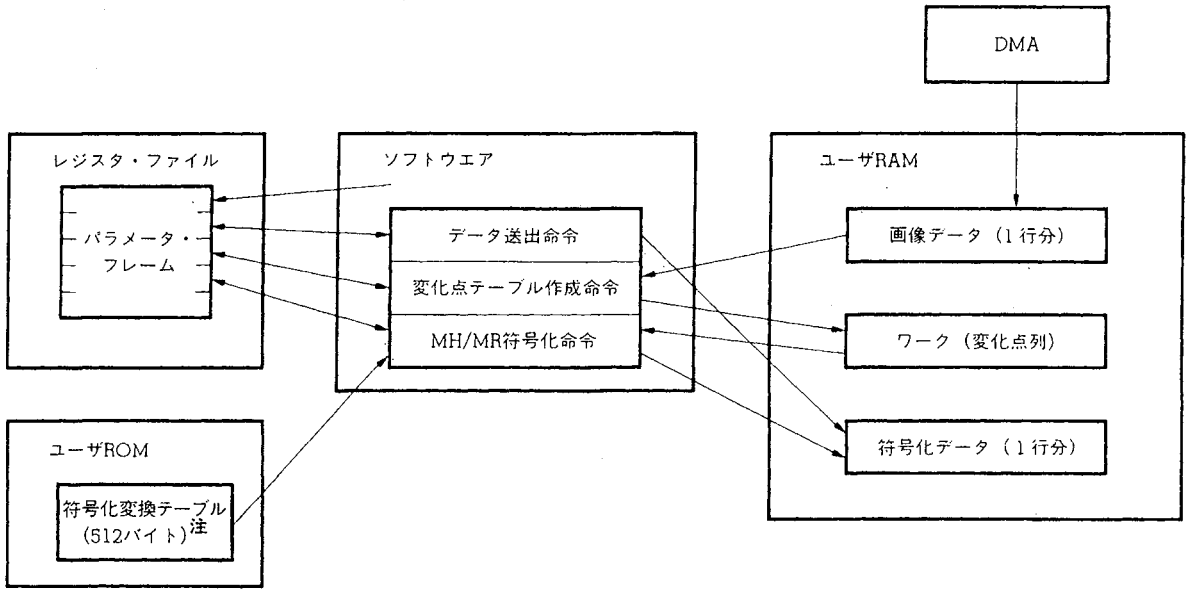
DS2：DS3：COLTRP

DS2：SCHEOL

図5－18に符号化命令と各メモリ上のデータの関係を，図5－19に復号化命令と各メモリ上のデータの関係を示します。

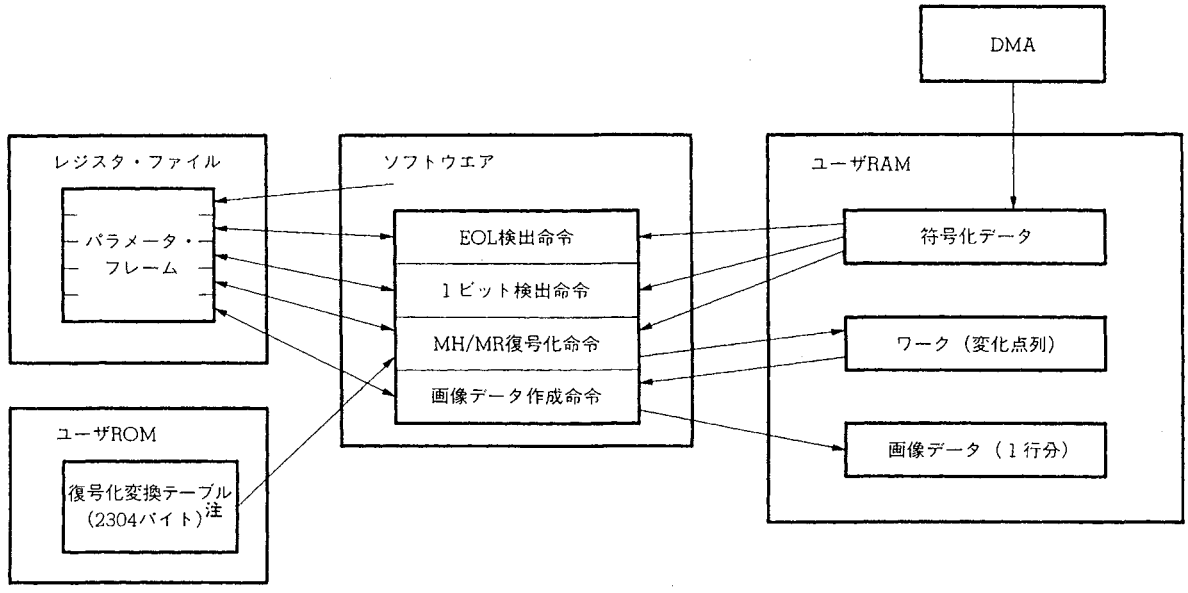
保守／廃止

図5－18　符号化命令と各メモリ上のデータ



注　MH，MR符号化命令の場合です。

図5－19　復号化命令と各メモリ上のデータ



注　MH，MR復号化命令の場合です。

保守／廃止

5.4　符号化／復号化変換テーブル

符号化変換テーブルは，画素データ（ラン・レングス）をMH符号に変換するためのテーブルです。MH符号化命令“MHENC”，MR符号化命令“MRENC”を使用する場合には，**5.4.1　符号化変換テーブル**で規定されるテーブルを指定してください（**5.4.3　復号化変換テーブル（MH復号化命令用）**参照）。

復号化変換テーブルは，符号を画素データ（ラン・レングス）に変換するためのテーブルです。MH復号化命令“MHDEC”，MR復号化命令“MRDEC”を使用する場合には，**5.4.2　復号化変換テーブル**で規定されるテーブルを指定してください（**5.4.4　符号化変換テーブル（MH符号化命令用）**参照）。

符号化／復号化変換テーブルの開始オフセットは，それぞれ“0000H”番地に配置してください。

5.4.1　符号化変換テーブル

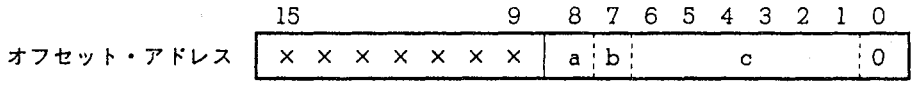
符号化変換テーブルは，次のような規則で構成されます。

ラン・レングス＝Rの画素データ符号（MH）は，符号化変換テーブルの，次に示すようなオフセット・アドレスの指定先に格納されています。

（1）オフセット・アドレス

オフセット・アドレスは，次のように生成されます。

例　R＝64M＋Tの場合



a：0＝白，1＝黒

b：0＝ターミネート符号，1＝メークアップ符号

c：ターミネート符号の場合（b＝0）

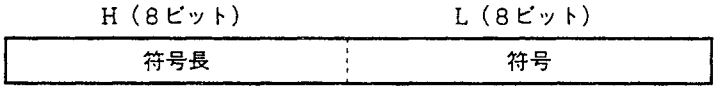
T（2進数）の値

メークアップ符号の場合（b＝1）

M（2進数）の値

（2）オフセット・アドレスの指定先

オフセット・アドレスが指定するワード領域には，次のような符号が格納されます。



保守／廃止

●符号が８ビット未満の場合，符号は右詰めにされます。８ビットに満たない分は“0”となります。

●符号が８ビット以上の場合，符号は左詰めにされます。

●符号はLSB先頭で示されます。

例　白ラン・レングスR＝708の場合

R＝64×11＋4　→　M＝11，T＝4

●メークアップ符号

符号は“**０１１００１１００**”です。

オフセット・アドレス＝××××××××010010110の示すワード領域に次のように格納されます。この領域に格納されない分についてはすべて“0”です。

00001001	**００１１００１１**
符号長	符号

●ターミネート符号

符号は“**１０１１**”です。

オフセット・アドレス＝××××××××000001000の示すワード領域に次のように格納されます。８ビットに満たない分についてはすべて“0”です。

00000100	**００００１１０１**
符号長	符号

保守／廃止

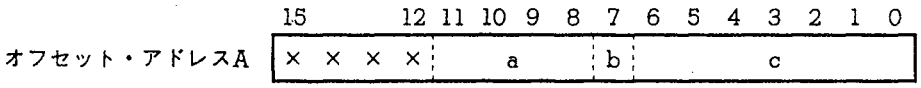
5.4.2　復号化変換テーブル

復号化変換テーブルは，次のような規則で生成されます。

変換の対象となる符号データの，連続する“0”の数をカウントします。

(1) オフセット・アドレスA

オフセット・アドレスAは，次のように生成されます。

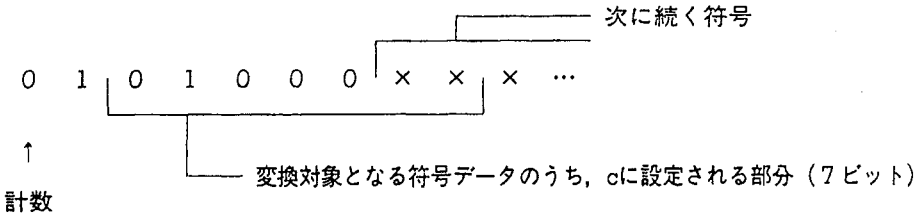


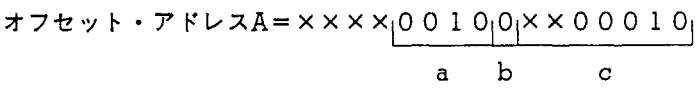
a：（カウントした“0”の数）＋1

b：0＝白，1＝黒

c：変換対象となる符号データ

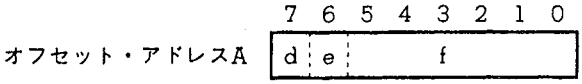
例　変換対象となる符号＝0101000（白ラン・レングス＝24）の場合





(2) オフセット・アドレスAの指定先

オフセット・アドレスAが指定するバイト領域には，次のようなデータが格納されます。



d：0＝白，1＝黒

e：0＝ターミネート符号，1＝メークアップ符号

f：ラン・レングス

(3) オフセット・アドレスB

(2)で格納された８ビットのデータに“000H”を付加し，12ビットにしたものをオフセット・アドレスBとします。オフセット・アドレスBが指定するバイト領域には，変換対象となる符号の“符号長－（カウントした“0”の数＋1)”が格納されます。

例　オフセット・アドレスB＝×××０００００００１１０００（×０１８H）

（変換対象となる符号が“０１０１０００”の場合）

	7	6	5	4	3	2	1	0	
オフセット・アドレスB	0	0	0	0	0	1	0	1	7－(1＋1)＝5

保守／廃止

5.4.3　復号化変換テーブル（MH復号化命令用）

```
;*****************************************
;*      MH decode (expansion) table      *
;*         compression code -> data      *
;*****************************************
;
;      * EOL
;         Number of '0' = 11
;
;      * EOL + '1'
;       -----4bit---+-1-+-----------7bit--------------
;       +---+---+---+---+---+---+---+---+---+---+---+---+
;       |num of'0'+1|col|           search data         | ---+
;       +---+---+---+---+---+---+---+---+---+---+---+---+    |
;           num of '0'+1: number of '0'+1                    |
;           col: color                                       |
;           search data: 7 bits                              |
;           ex) 0001 1011xxxx                                |
;               num of '0' = 3                               |
;               search data = 1011xxx                        |
;                                                            |
;       --1-+-1-+---------6bit-----------                    |
;       +---+---+---+---+---+---+---+---+                    |
;       |col|t/m|       run length      | <------------------+
;       +---+---+---+---+---+---+---+---+ ------------+
;           col: color                                 |
;           t/m: 0 = terminate code                    |
;                1 = makeup code                       |
;           run length: 6 bits                         |
;                                                      |
;       ------4bit------+-----4bit-------              |
;       +---+---+---+---+---+---+---+---+              |
;       | 0   0   0   0 |sch data length| <------------+
;       +---+---+---+---+---+---+---+---+
;        data length
;           terminate:  number of '0'+1 + seach data length
;           makeup    : (number of '0'+1 + seach data length)*64
```

保守／廃止

```
;
;+++++++++++++ Decode Table +++++++++++++
code    segment at      xxxxh                   ;       offset
        ORG     0h
;------ White Terminal Data Table -------
        db      05h,02h,02h,03h,03h,03h,03h,03h   ; W.T=000h-007h
        db      04h,04h,02h,03h,03h,01h,05h,05h   ; W.T=008h-00Fh
        db      05h,05h,05h,03h,03h,04h,01h,02h   ; W.T=010h-017h
        db      05h,05h,04h,05h,04h,01h,01h,04h   ; W.T=018h-01Fh
        db      04h,04h,04h,04h,04h,04h,04h,05h   ; W.T=020h-
        db      05h,05h,05h,05h,05h,02h,02h,03h
        db      03h,06h,06h,06h,06h,05h,05h,06h   ; W.T=030h-
        db      06h,06h,06h,06h,06h,05h,05h,05h
;-------- White Makeup Data Table --------
        db      00h,04h,04h,04h,05h,05h,05h,06h   ; W.M=040h-
        db      06h,06h,06h,07h,07h,07h,07h,07h
        db      07h,07h,07h,07h,07h,07h,07h,07h   ; W.M=050h-057h
        db      07h,07h,04h,07h                   ; W.M=058h-05Bh
;---- White/Black Addition Data Table ---
        db      03h,03h,03h,04h                   ; W/B.M=05Ch-05Fh
        db      04h,04h,04h,04h,04h,04h,04h,04h   ; W/B.M=060h-067h
        db      04h,00h,00h,00h,00h,00h,00h,00h   ; W/B.M=068h dummy=069h-
        db      00h,00h,00h,00h,00h,00h,00h,00h   ; dummy
        db      00h,00h,00h,00h,00h,00h,00h,00h
;------ Black Terminate Data Table ------
        db      05h,01h,01h,01h,01h,01h,01h,01h   ; B.T=080h-087h
        db      02h,02h,02h,02h,02h,02h,02h,04h   ; B.T=088h-08Fh
        db      04h,04h,03h,06h,06h,06h,05h,05h   ; B.T=090h-
        db      04h,04h,07h,07h,07h,07h,06h,06h
        db      06h,06h,07h,07h,07h,07h,07h,07h   ; B.T=0A0h-
        db      06h,06h,07h,07h,06h,06h,06h,06h
        db      06h,06h,06h,06h,05h,05h,05h,05h   ; B.T=0B0h-
        db      05h,06h,06h,05h,05h,06h,06h,06h
;------- Black Makeup Data Table --------
        db      00h,03h,07h,07h,06h,05h,05h,05h   ; B.M=0C0h-
        db      06h,06h,06h,06h,06h,06h,06h,06h
        db      06h,06h,06h,06h,06h,06h,06h,06h   ; B.M=0D0h-0D7h
        db      06h,06h,06h,06h                   ; B.M=0D8h-0DBh
```

保守／廃止

```
        ORG     0100h
;++++++++++++++ Decode Table ++++++++++++++
        db      003h,005h,009h,006h,042h,00Eh,004h,007h   ; address=0100h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h   ; address=0110h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h   ; address=0120h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h   ; address=0130h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h   ; address=0140h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h   ; address=0150h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h   ; address=0160h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h   ; address=0170h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
;
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=0180h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=0190h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01A0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01B0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01C0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01D0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01E0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h   ; address=01F0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
```

```
        ORG     0200h
;++++++++++++++ Decode Table ++++++++++++++
        db      00Bh,05Ah,018h,002h,01Bh,049h,037h,002h   ; address=0200h-
        db      00Bh,047h,033h,002h,057h,053h,043h,002h
        db      00Bh,05Ah,031h,002h,03Bh,04Fh,039h,002h   ; address=0210h-
        db      00Bh,04Bh,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,04Dh,038h,002h   ; address=0220h-
        db      00Bh,048h,034h,002h,059h,055h,043h,002h
        db      00Bh,05Ah,032h,002h,03Ch,051h,03Ah,002h   ; address=0230h-
        db      00Bh,04Ah,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,049h,037h,002h   ; address=0240h-
        db      00Bh,047h,033h,002h,058h,054h,043h,002h
        db      00Bh,05Ah,031h,002h,03Bh,050h,039h,002h   ; address=0250h-
        db      00Bh,04Ch,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,04Eh,038h,002h   ; address=0260h-
        db      00Bh,048h,034h,002h,05Bh,056h,043h,002h
        db      00Bh,05Ah,032h,002h,03Ch,052h,03Ah,002h   ; address=0270h-
        db      00Bh,04Ah,019h,002h,012h,044h,043h,002h
;
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=0280h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=0290h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02A0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02B0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02C0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02D0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02E0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h   ; address=02F0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
```

保守／廃止

```
        ORG     0300h
;++++++++++++++ Decode Table ++++++++++++++
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah   ; address=0300h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah   ; address=0310h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah   ; address=0320h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah   ; address=0330h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah   ; address=0340h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah   ; address=0350h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah   ; address=0360h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah   ; address=0370h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
;
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=0380h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=0390h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03A0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03B0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03C0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03D0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03E0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h   ; address=03F0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
```

```
        ORG     0400h
;++++++++++++++ Decode Table ++++++++++++++
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0400h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0410h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0420h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0430h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0440h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0450h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0460h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h   ; address=0470h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
;
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=0480h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=0490h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04A0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04B0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04C0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04D0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04E0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h   ; address=04F0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
```

保守／廃止

```
        ORG     0500h
;++++++++++++++ Decode Table ++++++++++++++
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0500h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0510h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0520h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0530h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0540h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0550h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0560h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh   ; address=0570h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
;
        db      08Ah,08Fh,08Bh,08Ch,08Ah,094h,08Bh,08Ch   ; address=0580h-
        db      08Ah,0C2h,08Bh,08Ch,08Ah,095h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A4h,08Bh,08Ch   ; address=0590h-
        db      08Ah,09Ch,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A2h,08Bh,08Ch   ; address=05A0h-
        db      08Ah,09Ah,08Bh,08Ch,08Ah,0AAh,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A6h,08Bh,08Ch   ; address=05B0h-
        db      08Ah,093h,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,094h,08Bh,08Ch   ; address=05C0h-
        db      08Ah,0C3h,08Bh,08Ch,08Ah,095h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A5h,08Bh,08Ch   ; address=05D0h-
        db      08Ah,09Dh,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A3h,08Bh,08Ch   ; address=05E0h-
        db      08Ah,09Bh,08Bh,08Ch,08Ah,0ABh,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A7h,08Bh,08Ch   ; address=05F0h-
        db      08Ah,093h,08Bh,08Ch,08Ah,080h,08Bh,08Ch
```

```
        ORG     0600h
;++++++++++++++ Decode Table ++++++++++++++
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0600h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0610h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0620h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=062Eh-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0640h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0650h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0660h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h   ; address=0670h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
;
        db      08Dh,091h,097h,08Eh,08Dh,09Eh,0B9h,08Eh   ; address=0680h-
        db      08Dh,0B0h,0ACh,08Eh,08Dh,0A8h,090h,08Eh
        db      08Dh,091h,0B2h,08Eh,08Dh,0A0h,0BDh,08Eh   ; address=0690h-
        db      08Dh,0BEh,0AEh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Fh,0BAh,08Eh   ; address=06A0h-
        db      08Dh,0B1h,0ADh,08Eh,08Dh,0A9h,090h,08Eh
        db      08Dh,091h,0B3h,08Eh,08Dh,0A1h,0C4h,08Eh   ; address=06B0h-
        db      08Dh,0BFh,0AFh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Eh,0B9h,08Eh   ; address=06C0h-
        db      08Dh,0B0h,0ACh,08Eh,08Dh,0A8h,090h,08Eh
        db      08Dh,091h,0B2h,08Eh,08Dh,0A0h,0BDh,08Eh   ; address=06D0h-
        db      08Dh,0BEh,0AEh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Fh,0BAh,08Eh   ; address=06E0h-
        db      08Dh,0B1h,0ADh,08Eh,08Dh,0A9h,090h,08Eh
        db      08Dh,091h,0B3h,08Eh,08Dh,0A1h,0C4h,08Eh   ; address=06F0h-
        db      08Dh,0BFh,0AFh,08Eh,08Dh,096h,090h,08Eh
```

保守／廃止

```
        ORG     0700h
;++++++++++++++ Decode Table ++++++++++++++
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0700h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0710h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0720h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=072Eh-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0740h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0750h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0760h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh   ; address=0770h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
;
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h   ; address=0780h-
        db      092h,0DAh,0D6h,0D0h,0CCh,0C8h,098h,0C1h
        db      092h,099h,0D4h,0CEh,0CAh,0C7h,0D8h,0C1h   ; address=0790h-
        db      092h,0C5h,0BBh,0D2h,0B7h,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h   ; address=07A0h-
        db      092h,0DBh,0D7h,0D1h,0CDh,0C9h,098h,0C1h
        db      092h,099h,0D5h,0CFh,0CBh,0C7h,0D9h,0C1h   ; address=07B0h-
        db      092h,0C5h,0BBh,0D3h,0B7h,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h   ; address=07C0h-
        db      092h,0DAh,0D6h,0D0h,0CCh,0C8h,098h,0C1h
        db      092h,099h,0D4h,0CEh,0CAh,0C7h,0D8h,0C1h   ; address=07D0h-
        db      092h,0C5h,0BBh,0D2h,0B7h,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h   ; address=07E0h-
        db      092h,0DBh,0D7h,0D1h,0CDh,0C9h,098h,0C1h
        db      092h,099h,0D5h,0CFh,0CBh,0C7h,0D9h,0C1h   ; address=07F0h-
        db      092h,0C5h,0BBh,0D3h,0B7h,0B5h,098h,0C1h
```

保守／廃止

```
        ORG     0800h
;++++++++++++++ Decode Table ++++++++++++++
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0800h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0810h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0820h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0830h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0840h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0850h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0860h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0870h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
;
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0880h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=0890h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08A0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08B0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08C0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08D0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08E0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h   ; address=08F0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
code    ends

        end
```

保守／廃止

5.4.4　符号化変換テーブル（MH符号化命令用）

```
;*******************************************
;*      MH encode (compression) table      *
;*         data -> compression code        *
;*******************************************
;
;       address= +XXXXh
;
;       -1+1+----6bit---+1-      -----8bit(H)----+----8bit(L)-----
;       +-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;       |a|b|     c     |0| ===> |  code length  |               |
;       +-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;
;       * Run Length >= 64
;        Run_Length = 64T +T
;        a: 0 = white
;           1 = black
;        b: 0 = terminate --- c = T
;           1 = makeup    --- c = M
;
;       * Run Length < 64
;        Only change the terminate code
;
code    segment at      xxxxh
        ORG     0h
;++++++++++ White Encode Table ++++++++++
;-------------- Terminate ---------------
        db      0ACh, 008h              ; address= +0000h
        db      038h, 006h
        db      00Eh, 004h
        db      001h, 004h
        db      00Dh, 004h
        db      003h, 004h
        db      007h, 004h
        db      00Fh, 004h
        db      019h, 005h              ; address= +0010h
        db      005h, 005h
        db      01Ch, 005h
        db      002h, 005h
        db      004h, 006h
        db      030h, 006h
```

保守／廃止

```
db      00Bh, 006h
db      02Bh, 006h
db      015h, 006h              ; address= +0020h
db      035h, 006h
db      072h, 007h
db      018h, 007h
db      008h, 007h
db      074h, 007h
db      060h, 007h
db      010h, 007h
db      00Ah, 007h              ; address= +0030h
db      06Ah, 007h
db      064h, 007h
db      012h, 007h
db      00Ch, 007h
db      040h, 008h
db      0C0h, 008h
db      058h, 008h
db      0D8h, 008h              ; address= +0040h
db      048h, 008h
db      0C8h, 008h
db      028h, 008h
db      0A8h, 008h
db      068h, 008h
db      0E8h, 008h
db      014h, 008h
db      094h, 008h              ; address= +0050h
db      054h, 008h
db      0D4h, 008h
db      034h, 008h
db      0B4h, 008h
db      020h, 008h
db      0A0h, 008h
db      050h, 008h
db      0D0h, 008h              ; address= +0060h
db      04Ah, 008h
db      0CAh, 008h
db      02Ah, 008h
db      0AAh, 008h
```

```
        db      024h, 008h
        db      0A4h, 008h
        db      01Ah, 008h
        db      09Ah, 008h              ; address= +0070h
        db      05Ah, 008h
        db      0DAh, 008h
        db      052h, 008h
        db      0D2h, 008h
        db      04Ch, 008h
        db      0CCh, 008h
        db      02Ch, 008h
;-------------- Terminate ---------------
        db      000h, 000h              ; address= +0080h / no data
        db      01Bh, 005h
        db      009h, 005h
        db      03Ah, 006h
        db      076h, 007h
        db      06Ch, 008h
        db      0ECh, 008h
        db      026h, 008h
        db      0A6h, 008h              ; address= +0090h
        db      016h, 008h
        db      0E6h, 008h
        db      033h, 009h
        db      0B3h, 009h
        db      04Bh, 009h
        db      0CBh, 009h
        db      02Bh, 009h
        db      0ABh, 009h              ; address= +00A0h
        db      06Bh, 009h
        db      0EBh, 009h
        db      01Bh, 009h
        db      09Bh, 009h
        db      05Bh, 009h
        db      0DBh, 009h
        db      019h, 009h
        db      099h, 009h              ; address= +00B0h
        db      059h, 009h
        db      006h, 006h
```

```
        db      0D9h, 009h
        db      010h, 00Bh
        db      030h, 00Bh
        db      0B0h, 00Bh
        db      048h, 00Ch
        db      0C8h, 00Ch              ; address= +00C0h
        db      028h, 00Ch
        db      0A8h, 00Ch
        db      068h, 00Ch
        db      0E8h, 00Ch
        db      038h, 00Ch
        db      0B8h, 00Ch
        db      078h, 00Ch
        db      0F8h, 00Ch              ; address= +00D0h

;++++++++++ Black Encode Table ++++++++++
        ORG     0100h
;-------------- Terminate ---------------
        db      0ECh, 00Ah              ; address= +0100h
        db      002h, 003h
        db      003h, 002h
        db      001h, 002h
        db      006h, 003h
        db      00Ch, 004h
        db      004h, 004h
        db      018h, 005h
        db      028h, 006h              ; address= +0110h
        db      008h, 006h
        db      010h, 007h
        db      050h, 007h
        db      070h, 007h
        db      020h, 008h
        db      0E0h, 008h
        db      018h, 009h
        db      0E8h, 00Ah              ; address= +0120h
        db      018h, 00Ah
        db      010h, 00Ah
        db      0E6h, 00Bh
        db      016h, 00Bh
```

保守／廃止

```
db      036h, 00Bh
db      0ECh, 00Bh
db      014h, 00Bh
db      0E8h, 00Bh              ; address= +0130h
db      018h, 00Bh
db      053h, 00Ch
db      0D3h, 00Ch
db      033h, 00Ch
db      0B3h, 00Ch
db      016h, 00Ch
db      096h, 00Ch
db      056h, 00Ch              ; address= +0140h
db      0D6h, 00Ch
db      04Bh, 00Ch
db      0CBh, 00Ch
db      02Bh, 00Ch
db      0ABh, 00Ch
db      06Bh, 00Ch
db      0EBh, 00Ch
db      036h, 00Ch              ; address= +0150h
db      0B6h, 00Ch
db      05Bh, 00Ch
db      0DBh, 00Ch
db      02Ah, 00Ch
db      0AAh, 00Ch
db      06Ah, 00Ch
db      0EAh, 00Ch
db      026h, 00Ch              ; address= +0160h
db      0A6h, 00Ch
db      04Ah, 00Ch
db      0CAh, 00Ch
db      024h, 00Ch
db      0ECh, 00Ch
db      01Ch, 00Ch
db      0E4h, 00Ch
db      014h, 00Ch              ; address= +0170h
db      01Ah, 00Ch
db      09Ah, 00Ch
db      0D4h, 00Ch
```

保守／廃止

```
        db      034h, 00Ch
        db      05Ah, 00Ch
        db      066h, 00Ch
        db      0E6h, 00Ch
;-------------- Makeup ------------------
        db      000h, 000h              ; address= +0180h / no data
        db      0F0h, 00Ah
        db      013h, 00Ch
        db      093h, 00Ch
        db      0DAh, 00Ch
        db      0CCh, 00Ch
        db      02Ch, 00Ch
        db      0ACh, 00Ch
        db      036h, 00Dh              ; address= +0190h
        db      0B6h, 00Dh
        db      052h, 00Dh
        db      0D2h, 00Dh
        db      032h, 00Dh
        db      0B2h, 00Dh
        db      04Eh, 00Dh
        db      0CEh, 00Dh
        db      02Eh, 00Dh              ; address= +01A0h
        db      0AEh, 00Dh
        db      06Eh, 00Dh
        db      0EEh, 00Dh
        db      04Ah, 00Dh
        db      0CAh, 00Dh
        db      02Ah, 00Dh
        db      0AAh, 00Dh
        db      05Ah, 00Dh              ; address= +01B0h
        db      0DAh, 00Dh
        db      026h, 00Dh
        db      0A6h, 00Dh
        db      010h, 00Bh
        db      030h, 00Bh
        db      0B0h, 00Bh
        db      048h, 00Ch
        db      0C8h, 00Ch              ; address= +01C0h
        db      028h, 00Ch
```

```
        db      0A8h, 00Ch
        db      068h, 00Ch
        db      0E8h, 00Ch
        db      038h, 00Ch
        db      0B8h, 00Ch
        db      078h, 00Ch
        db      0F8h, 00Ch              ; address= +01D0h

code    ends
        end
```

5.5　処理フロー

符号化／復号化の処理において，**5.1　特徴**に示した各専用命令は図5－20，図5－21に示すような順に使用されます。

図5－20　1ライン分の符号化処理の流れ

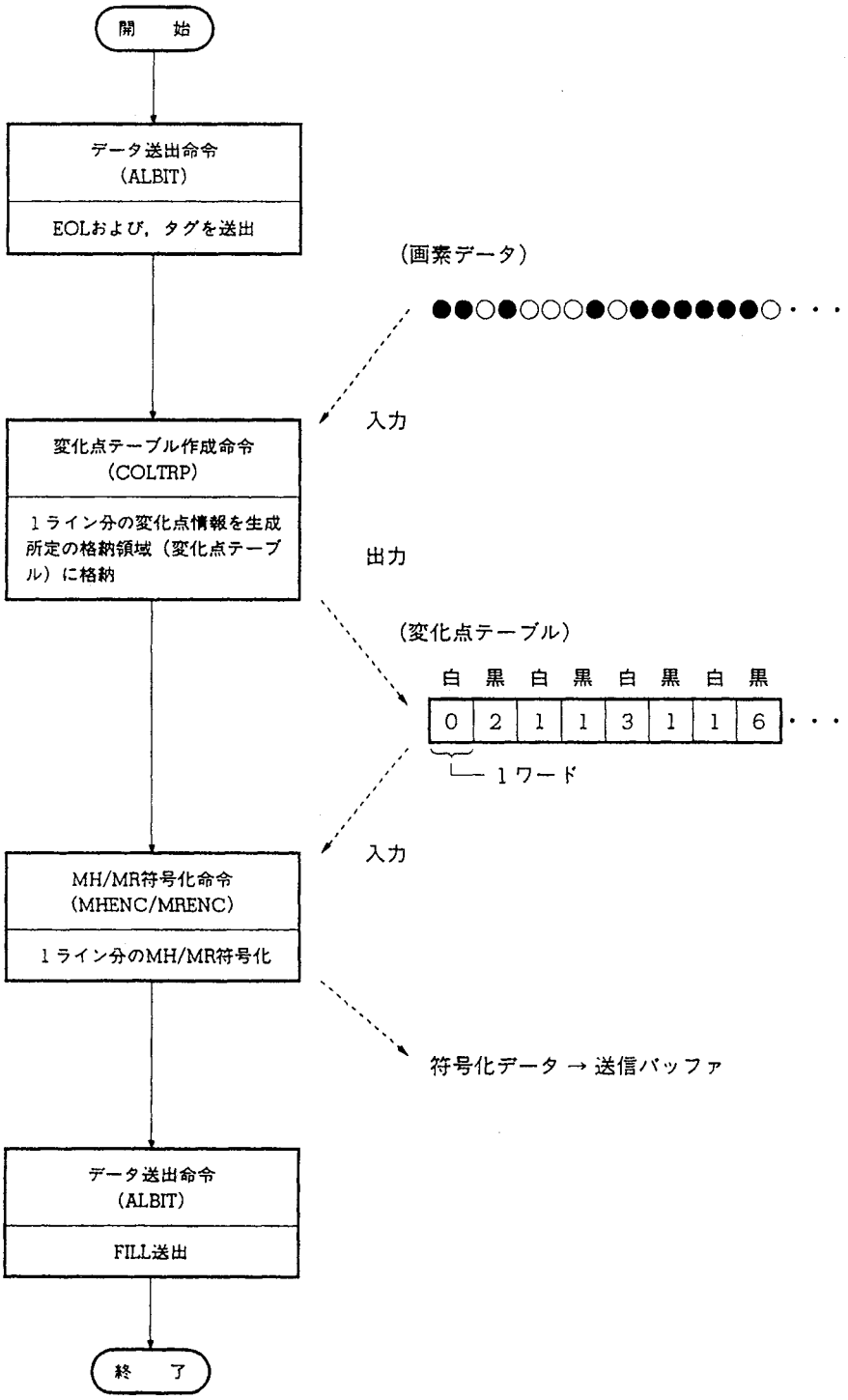
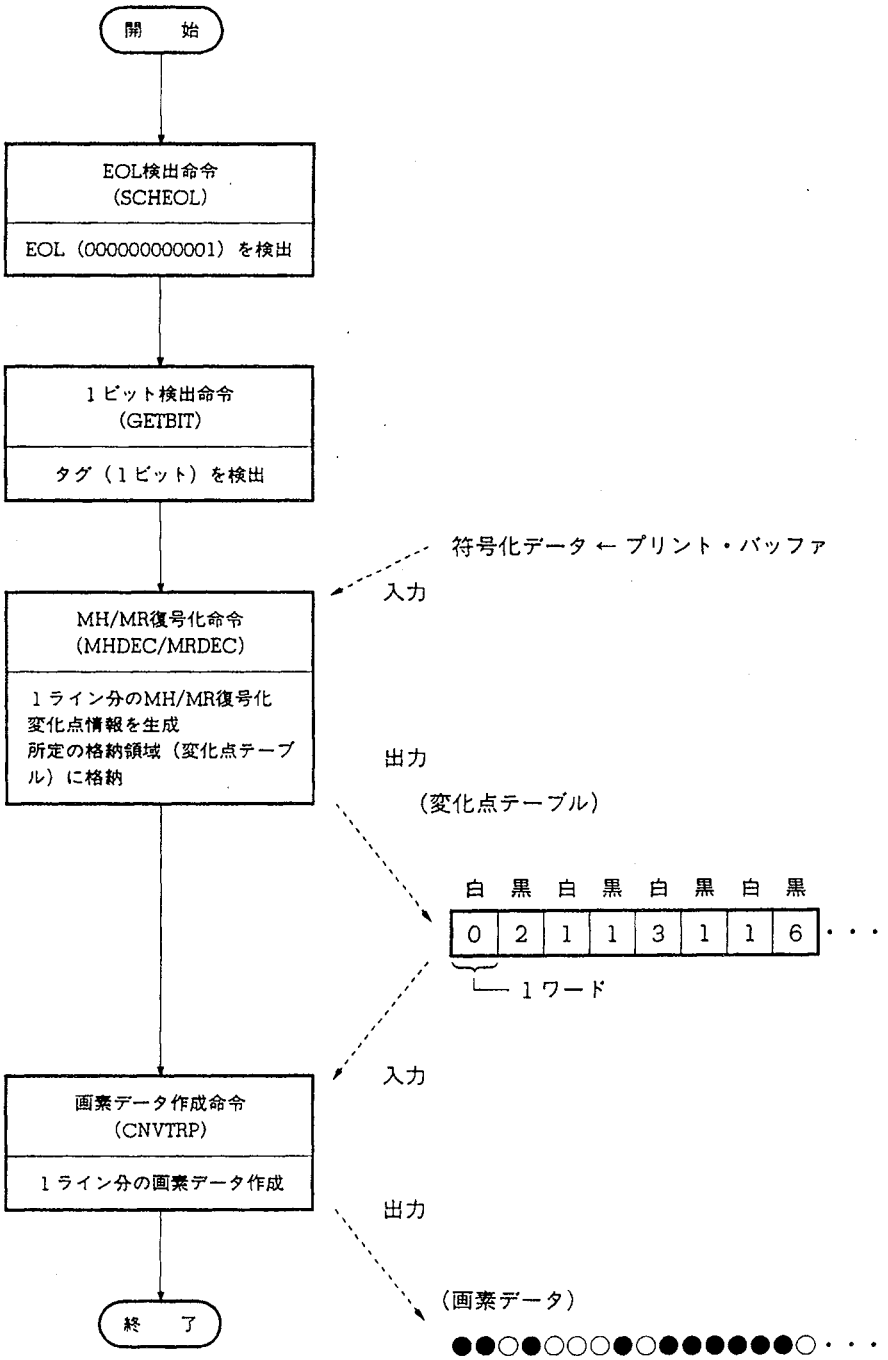


図5－21　1ライン分の復号化処理の流れ



保守／廃止

5.6　各命令の機能説明

5.6.1　データ送出命令

<記　述　形　式>　ALBIT（Add Last BIT）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0

<バ　イ　ト　数>　2

<機　　能>

符号化処理においてEOL，FILLなど16ビット以下のデータを送信バッファへ出力するための命令です。データの先頭はLSBに入ります。

指定された送出したいデータが，処理中の16ビットに満たない半端符号と合わせて16ビット以上にまとまったところで指定された送信バッファへ出力します。16ビットを越える分，あるいは合わせて16ビットに満たないデータはパラメータとしてBW（処理中半端符号データ）に，そのデータのビット数はパラメータとしてCH（処理中半端符号ビット数）に保持します。

割り込み要求のサンプリングを行いません。

<入力パラメータ>

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

DW：送出符号データ

CL ：送出符号データ・ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

保守／廃止

＜出力パラメータ＞

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

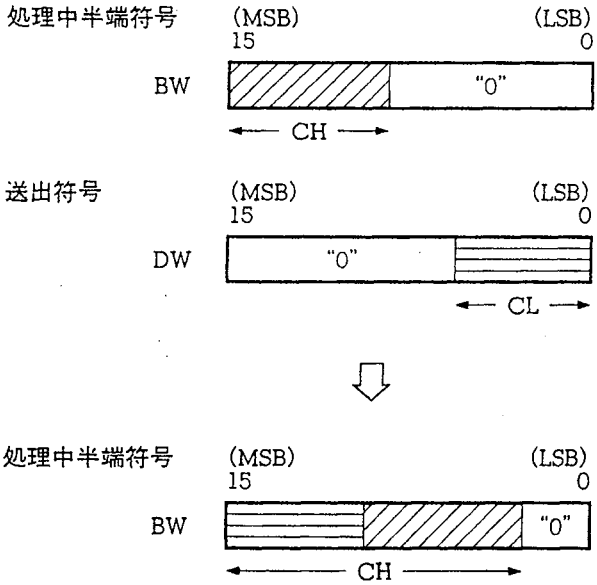
DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

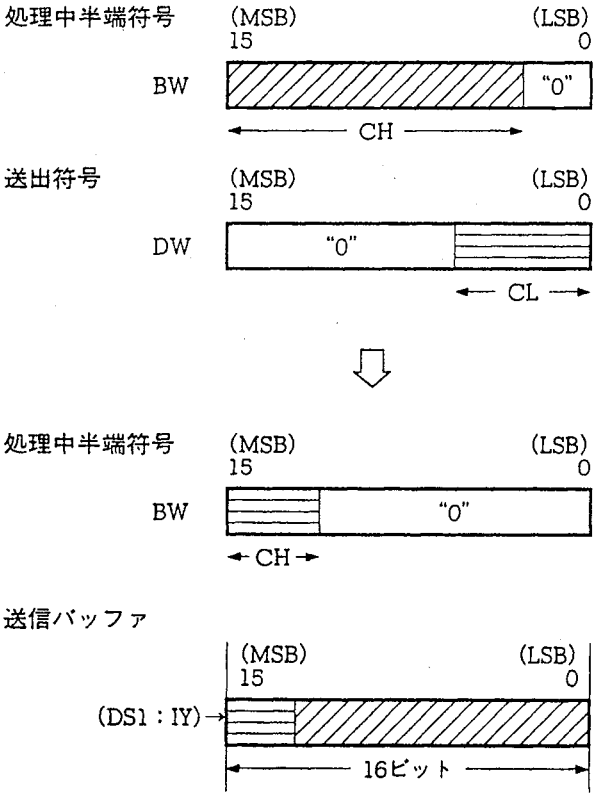
Zフラグ	CYフラグ	送信バッファ残り	データ残り
0	不定	あり	—
1	0	なし	なし
1	1	なし	あり

保守／廃止

例1．処理中半端符号と送出符号が合わせて16ビットに満たない場合



2．処理中半端符号と送出符号が合わせて16ビット以上の場合



IY ← IY+2

BP ← BP−2

保守／廃止

5.6.2　変化点テーブル作成命令

<記　述　形　式>　COLTRP（Collect Turning Point）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	1

<バ　イ　ト　数>　2

<機　　能>

符号化処理において，１ライン分の画素データの変化点情報（白／黒のラン・レングス）を指定されたメモリ（変化点テーブル）上に生成する命令です。データの先頭はLSBに入ります。

変化点テーブルは，白のラン・レングスを先頭に１ワードごとに各色（白／黒）のラン・レングスを順次格納します。変化点テーブルの最終ワードには，１ライン分の終了情報として“FFFFH”が格納されます。

中断処理はありません。１ワード分の画素データ処理ごとに割り込み要求のサンプリングを行います。

（画素データ）

●●●●○●○○○○○●●●○○●○…

↓　COLTRP命令発行

（変化点テーブル）

白	黒	白	黒	白	黒	白	黒		
0	4	1	1	5	3	2	1	・・・	FFFFH

└ １ワード（２バイト）

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号

ALで指定されるレジスタ・バンク

00H	（画素データ・バッファ・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	（変化点テーブル・セグメント）	(DS3)	
04H	ワーク・エリア		
06H	ワーク・エリア		
08H	画素データ・バッファ・セグメント	(DS0)	入力データ処理情報
0AH	画素データ・バッファ・オフセット	（ポインタ）	
0CH	画素データ・バッファ・サイズ注2	（カウンタ）	
0EH	変化点テーブル・セグメント	(DS1)	出力データ処理情報
10H	変化点テーブル・オフセット	（ポインタ）	
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	0クリア注3/ワーク・エリア		0クリア必要パラメータ
1EH	0クリア注3/ワーク・エリア		

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. バイト単位。偶数で指定してください。

3. 新規命令処理時に0クリアを行います。

注意　各命令で使用されるパラメータ用のレジスタ・バンク内のポインタ，カウンタ値は，命令実行によって更新されます。

保守／廃止

5.6.3　MH符号化命令

<記　述　形　式>　MHENC（MH Encode）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	1

<バ　イ　ト　数>　2

<機　　能>

1ライン分の画素データの変化点情報を入力し，所定の符号化変換テーブルによりMH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出する命令です。データの先頭はLSBに入ります。

各色（白または黒）の変化点情報（変化点テーブルの1ワード）の符号化処理ごとに割り込み要求のサンプリングを行います。

MHENC命令において，連続した色（白または黒）で符号化できる最大画素数は合計5183画素です。

<入力パラメータ>

AL　：レジスタ・バンク番号指定

BW：処理中半端符号データ

CH：処理中半端符号ビット数

BP　：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY　：送信バッファ・オフセット

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）　（DS2）	拡張セグメント・オーバライド・プリフィクス注1
02H	ワーク・エリア	
04H	ワーク・エリア	
06H	ワーク・エリア	
08H	変化点テーブル・セグメント　（DS0）	入力データ処理情報
0AH	符号化ライン変化点テーブル・オフセット　（ポインタ）	
0CH	ワーク・エリア	
0EH	符号化変換テーブル・セグメント　（DS1）	出力データ処理情報
10H	ワーク・エリア	
12H	ワーク・エリア	
14H	ワーク・エリア	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	0クリア注2/ワーク・エリア	0クリア必要パラメータ
1EH	0クリア注2/ワーク・エリア	

注1.　DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2.　新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

BW：処理中半端符号データ

CH：処理中半端符号ビット数

BP：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY：送信バッファ・オフセット

Zフラグ	CYフラグ	送信バッファ残り	データ残り	処理
0	0	あり	CHの値	終了
0	1	—	—	—
1	0	なし	なし	終了
1	1	なし	あり	中断

★　注意　最後の符号化データを送信バッファに送出した際に，送信バッファの残りバイト・サイズがちょうど"0"となった場合，MHENC命令実行後のフラグ内容は，Z＝1，CY＝1（送信バッファ残りなし，データ残りあり，処理中断）となります。したがって，MHENC命令を再実行してください。

保守／廃止

5.6.4　MR符号化命令

<記　述　形　式>　MRENC（MR Encode）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1

<バ　イ　ト　数>　2

<機　　能>

参照ラインの変化点情報と符号化ラインの変化点情報とを入力し，１ライン分のMR符号化を行い，その符号を指定されたメモリ（送信バッファ）上に送出する命令です。データの先頭はLSBに入ります。

MR符号化の１モード単位（パス・モード，垂直モード，水平モードのいずれか）の符号化処理ごとに割り込み要求のサンプリングを行います。

MRENC命令の水平モードのMH符号化において，連続した色（白または黒）で符号化できる最大画素数は合計5183画素です。

<入力パラメータ>

AL　：レジスタ・バンク番号指定

BW　：処理中半端符号データ

CH　：処理中半端符号ビット数

BP　：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY　：送信バッファ・オフセット

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）　(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	ワーク・エリア	
04H	ワーク・エリア	
06H	ワーク・エリア	
08H	変化点テーブル・セグメント　(DS0)	
0AH	符号化ライン変化点テーブル・オフセット　（ポインタ）	
0CH	参照ライン変化点テーブル・オフセット（ポインタ）	
0EH	符号化変換テーブル・セグメント　(DS1)	
10H	ワーク・エリア	
12H	ワーク・エリア	
14H	ワーク・エリア	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	0クリア注2/ワーク・エリア	0クリア必要パラメータ
1EH	0クリア注2/ワーク・エリア	

注1. DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

BW ：処理中半端符号データ

CH ：処理中半端符号ビット数

BP ：バッファ・サイズ（送信バッファの残りバイト・サイズ）

DS1：送信バッファ・セグメント

IY ：送信バッファ・オフセット

Zフラグ	CYフラグ	送信バッファ残り	データ残り	処理
0	0	あり	CHの値	終了
0	1	—	—	—
1	0	なし	なし	終了
1	1	なし	あり	中断

★ **注意** 最後の符号化データを送信バッファに送出した際に，送信バッファの残りバイト・サイズがちょうど“0”となった場合，MRENC命令実行後のフラグ内容は，Z＝1，CY＝1（送信バッファ残りなし，データ残りあり，処理中断）となります。したがって，MRENC命令を再実行してください。

保守／廃止

5.6.5　EOL検出命令

<記　述　形　式>　SCHEOL（Search EOL）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0

<バ　イ　ト　数>　2

<機　　能>

復号化処理において，“EOL”（000000000001）を検出するための命令です。メモリ上（受信バッファ）の指定された位置から最初のEOLを検出します。データの先頭はLSBに入ります。

割り込み要求のサンプリングを行いません。したがって，割り込み要求のサンプリングを確実に行うには，SCHEOL命令が一定時間ごとに中断されるように，受信バッファ・サイズを小さく設定してください。これにより，長時間にわたる割り込み要求の保留を回避することができます。SCHEOL命令の実行時間（最長となる場合）の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません）。

- 16ビット・バス幅のとき：（214＋T）N/16＋74＋T（クロック）
- ８ビット・バス幅のとき：（110＋T）N/8＋74＋T（クロック）

N：EOL検出までのビット数

T：受信バッファのウエイト数

復号化処理のEOLを検出する処理において，次のような場合が考えられます。

① 指定された受信バッファ内が，すべて“0”の場合（EOL検出処理の中断）。
② 一番先頭にEOLを検出する場合。
③ FILL付きEOLを先頭に検出する場合。
④ FILL以外の符号を検出したあと，EOLを検出する場合。

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）	（DS2）	拡張セグメント・オーバライド・プリフィクス注1
02H	ワーク・エリア		
04H	処理中半端符号データ		
06H	処理中半端符号ビット数		
08H	受信バッファ・セグメント	（DS0）	
0AH	受信バッファ・オフセット	（ポインタ）	
0CH	受信バッファ・サイズ注2	（カウンタ）	
0EH	ワーク・エリア		
10H	ワーク・エリア		
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	ワーク・エリア		
1EH	0クリア注3/ワーク・エリア		…0クリア必要パラメータ

注1. DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2. バイト単位。偶数で指定してください。

3. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

Zフラグ	CYフラグ	受信バッファ残り	終了状態	AHステータス
0	0	あり	正常終了	01，02
0	1	あり	終了エラー	03
1	0	なし	正常終了	01，02
1	1	なし	終了エラー	00，03

保守／廃止

AH内の数値によって終了状態をチェックすることができます。数値と終了状態を次に示します。

AH	終　了　状　態
00H	中断
01H	先頭にEOLを検出
02H	FILL付きのEOLを先頭に検出
03H	EOL以外の符号を検出したあと，EOLを検出

保守／廃止

5.6.6　1ビット検出命令

<記　述　形　式>　GETBIT (GET BIT)

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	1

<バ　イ　ト　数>　2

<機　　能>

復号化処理において，タグなどの1ビットを検出するための命令です。データの先頭と処理中の半端符号はLSBから入ります。

メモリ上（受信バッファ）の指定された位置から1ビットを取り出し，CYフラグに設定します。

割り込み要求のサンプリングを行いません。

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）	（DS2）	拡張セグメント・オーバライド・プリフィクス注1
02H	ワーク・エリア		
04H	処理中半端符号データ		
06H	処理中半端符号ビット数		
08H	受信バッファ・セグメント	（DS0）	
0AH	受信バッファ・オフセット	（ポインタ）	
0CH	受信バッファ・サイズ注2	（カウンタ）	
0EH	ワーク・エリア		
10H	ワーク・エリア		
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	ワーク・エリア		
1EH	ワーク・エリア		

注1．DS2：を付加した場合，オフセット値は08Hの代わりに00Hが参照されます。

2．バイト単位。0以外の偶数値を設定してください。

＜出力パラメータ＞

CYフラグ：検出した1ビット・データ

Zフラグ	受信バッファ残り	
0	あり	…受信データ・バッファ・サイズ>0
1	なし	…受信データ・バッファ・サイズ=0

注意　半端符号データの有無にかかわらず，受信データ・バッファ・サイズのカウンタ（レジスタ・バンクの0CHの内容）が"0"のとき，Zフラグ＝1となります。

保守／廃止

5.6.7　MH復号化変化点テーブル作成命令

<記　述　形　式>　MHDEC（MH Decode）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0

<バ　イ　ト　数>　2

<機　　能>

指定されたメモリ（受信バッファ）上の符号データを入力し，復号化変換テーブルによりMH復号化を行い，生成された変化点情報を指定されたメモリ（変化点テーブル）上に送出する命令です。データの先頭と処理中の半端符号はLSBから入ります。

変化点情報を生成することにより，１ライン分の画素データ・ビット数に達したことを認識し命令を終了とします。

MH復号化の符号データから変化点テーブルを作成する処理において，次のような場合が考えられます。

① 指定された受信バッファ内に１ライン分の符号データがあり，変化点情報への変換を終了する場合（正常終了）。
② 指定された受信バッファ内の符号が１ライン分に満たなくて，変化点情報への変換が途中で終了する場合（中断）。
③ １番先頭にEOLを検出する場合。
④ FILL付きのEOLを先頭に検出する場合。
⑤ １ライン分の画素数に満たないデータのあと，EOLを検出する場合。
⑥ １ライン分の画素数以上に復号化するデータがある場合。
⑦ 異常なコードを検出する場合。

正常終了のときだけ，変化点テーブルに“FFFFH”（1ライン分の終了コード）を出力します。

各色（白または黒）の変化点を生成する復号化処理ごとに，割り込み要求のサンプリングを行います。

MHDEC命令において，復号化できる１ラインの最大画素数は合計65535画素です。

MHDEC命令実行中に“0”の連続する異常符号を検出すると，次に“1”を検出するまで割り込み

保守／廃止

要求が保留されます。この場合，受信バッファ・サイズを小さく設定してMHDEC命令が一定の時間で中断されるようにすることにより，長時間にわたる割り込み要求の保留を回避することができます。MHDEC命令の実行時間（最長となる場合）の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません）。

- 16ビット・バス幅のとき：40+2T+（59+T）N/16（クロック）
- 8ビット・バス幅のとき：44+4T+（37+2T）N/8（クロック）

N：“0”の連続するビット数

T：変化点テーブルのウエイト数

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	（変化点テーブル・セグメント）	(DS3)	
04H	処理中半端符号データ		
06H	処理中半端符号ビット数		
08H	受信バッファ・セグメント	(DS0)	
0AH	受信バッファ・オフセット	（ポインタ）	
0CH	受信バッファ・サイズ注2	（カウンタ）	
0EH	変化点テーブル・セグメント	(DS1)	
10H	復号化ライン変化点テーブル・オフセット		
12H	1ライン分の総画素ビット数		
14H	復号化変換テーブル・セグメント		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	ワーク・エリア		
1CH	0クリア注3/ワーク・エリア		0クリア必要パラメータ
1EH	0クリア注3/ワーク・エリア		

注1. DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2. バイト単位。0以外の偶数値を設定してください。

3. 新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

Zフラグ	CYフラグ	受信バッファ残り	終了状態	AHステータス
0	0	あり	正常終了	変化しない
0	1	あり	終了エラー	**表5－4参照**
1	0	なし	正常終了	変化しない
1	1	なし	終了エラー	**表5－4参照**

AH内の数値によって終了状態をチェックすることができます。数値と終了状態を次に示します。

なお，AH＝01H，02Hの場合は，ALレジスタで指定されるレジスタ・バンクのオフセット・アドレス1AH番地のワーク・エリアの値により，終了状態がそれぞれ2通りあります。

表5－4　MHDEC命令の終了状態

AH	終了状態	
00H	中断	
01H	ワーク・エリア＝0	先頭にEOLを検出
	ワーク・エリア≠0	先頭にメークアップ符号＋EOLを検出
02H	ワーク・エリア＝0	先頭にFILL付きEOLを検出
	ワーク・エリア≠0	先頭にメークアップ符号＋FILL付きEOLを検出
03H	1ライン分の画素数に満たないデータのあと，EOLを検出	
04H	1ライン分の画素数以上に復号化するデータが存在	
05H	異常なコードを検出	

備考　異常コードを検出して終了した場合，復合化ライン変化点テーブルのオフセットは，最後に正常出力された変化点テーブル・データの次のワード領域のアドレスを指しています。

保守／廃止

5.6.8　MR復号化変化点テーブル作成命令

<記　述　形　式>　MRDEC (MR Decode)

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	1

<バ　イ　ト　数>　2

<機　　能>

指定されたメモリ（受信バッファ）上の復号ラインの符号データと参照ラインの変化点情報を入力し，復号化変換テーブルによりMR復号化を行い，生成された復号化ラインの変化点情報を指定されたメモリ(変化点テーブル)上に送出する命令です。データの先頭と処理中の半端符号はLSBから入ります。

変化点情報を生成することにより，１ライン分の画素データ・ビット数に達したことを認識し命令を終了とします。

MR復号化の符号データから変化点テーブルを作成する処理において,次のような場合が考えられます。

① 指定された受信バッファ内に１ライン分の符号データがあり，変化点情報への変換を終了する場合（正常終了)。
② 指定された受信バッファ内の符号が１ライン分に満たなくて，変化点情報への変換が途中で終了する場合（中断)。
③ １番先頭にEOLを検出する場合。
④ FILL付きのEOLを先頭に検出する場合。
⑤ １ライン分の画素数に満たないデータのあと，EOLを検出する場合。
⑥ １ライン分の画素数以上に復号化するデータがある場合。
⑦ 異常なコードを検出する場合。

正常終了のときだけ，変化点テーブルに“FFFFH”（１ライン分の終了コード）を出力します。

各モード（パス・モード，垂直モード，水平モード）単位の復号化処理ごとに，割り込み要求のサンプリングを行います。

保守／廃止

MRDEC命令において，MR復号化できる最大画素数は合計32767画素です。

MRDEC命令実行中に“0”の連続する異常符号を検出すると，次に“1”を検出するまで割り込み要求が保留されます。この場合，受信バッファ・サイズを小さく設定してMRDEC命令が一定の時間で中断されるようにすることにより，長時間にわたる割り込み要求の保留を回避することができます。MRDEC命令の実行時間（最長となる場合）の算出式を次に示します（リフレッシュ・サイクル，DMAサイクル，バス・ホールド要求は考慮していません）。

- 16ビット・バス幅のとき：211+6T+（59+T）N/16（クロック）
- 8ビット・バス幅のとき：227+12T+（37+2T）N/8（クロック）

N：“0”の連続するビット数

T：変化点テーブルのウエイト数

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（受信バッファ・セグメント）（DS2）	拡張セグメント・オーバライド・プリフィクス注1
02H	（変化点テーブル・セグメント）（DS3）	
04H	処理中半端符号データ	
06H	処理中半端符号ビット数	
08H	受信バッファ・セグメント（DS0）	
0AH	受信バッファ・オフセット（ポインタ）	
0CH	受信バッファ・サイズ注2（カウンタ）	
0EH	変化点テーブル・セグメント（DS1）	
10H	復号化ライン変化点テーブル・オフセット	
12H	参照ライン変化点テーブル・オフセット	
14H	復号化変換テーブル・セグメント	
16H	ワーク・エリア	
18H	ワーク・エリア	
1AH	ワーク・エリア	
1CH	0クリア注3/ワーク・エリア	0クリア必要パラメータ
1EH	0クリア注3/ワーク・エリア	

注1．DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2．バイト単位。0以外の偶数値を設定してください。

3．新規命令処理時に0クリアを行います。
中断した処理を再開する場合，0クリアが必要なパラメータは中断時の状態のままにしておいてください。

＜出力パラメータ＞

Zフラグ	CYフラグ	受信バッファ残り	終了状態	AHステータス
0	0	あり	正常終了	変化しない
0	1	あり	終了エラー	表5-5参照
1	0	なし	正常終了	変化しない
1	1	なし	終了エラー	表5-5参照

AH内の数値によって終了状態をチェックすることができます。数値と終了状態を次に示します。

表5-5　MRDEC命令の終了状態

AH	終　了　状　態
00H	中断
01H	先頭にEOLを検出
02H	FILL付きのEOLを先頭に検出
03H	1ライン分の画素数に満たないデータのあと，EOLを検出
04H	1ライン分の画素数以上に復号するデータが存在
05H	異常なコードを検出

備考　異常コードを検出して終了した場合，復号化ライン変化点テーブルのオフセットは，最後に正常出力された変化点テーブル・データの次のワード領域のアドレスを指しています。

保守／廃止

5.6.9　画素データ作成命令

<記　述　形　式>　CNVTRP（Convert Turning Point）

<命 令 語 形 式>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	0

<バ　イ　ト　数>　2

<機　　能>

指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に１ワード（16ビット）単位で送出する命令です。データの先頭はLSBに入ります。

各色（白または黒）の変化点情報（変化点テーブルの１ワード）を画素データに変換する処理ごとに割り込み要求のサンプリングを行います。

なお，16の倍数以外の画素データ数を示す変化点テーブルを使用した場合，最後の16ビットに満たない画素データは送出されません。

保守／廃止

＜入力パラメータ＞

AL：レジスタ・バンク番号指定

ALで指定されるレジスタ・バンク

00H	（変化点テーブル・セグメント）	(DS2)	拡張セグメント・オーバライド・プリフィクス注1
02H	（プリント・バッファ・セグメント）	(DS3)	
04H	ワーク・エリア		
06H	ワーク・エリア		
08H	変化点テーブル・セグメント	(DS0)	
0AH	変化点テーブル・オフセット	（ポインタ）	
0CH	ワーク・エリア		
0EH	プリント・バッファ・セグメント	(DS1)	
10H	プリント・バッファ・オフセット	（ポインタ）	
12H	ワーク・エリア		
14H	ワーク・エリア		
16H	ワーク・エリア		
18H	ワーク・エリア		
1AH	処理中半端画素データ		
1CH	0クリア注2／処理中半端画素ビット数		0クリア必要パラメータ
1EH	0クリア注2／ワーク・エリア		

注1．DS2：，DS3：を付加した場合，オフセット値は08H，0EHの代わりにそれぞれ00H，02Hが参照されます。

2．新規命令処理時に0クリアを行います。

注意　中断処理はありません。

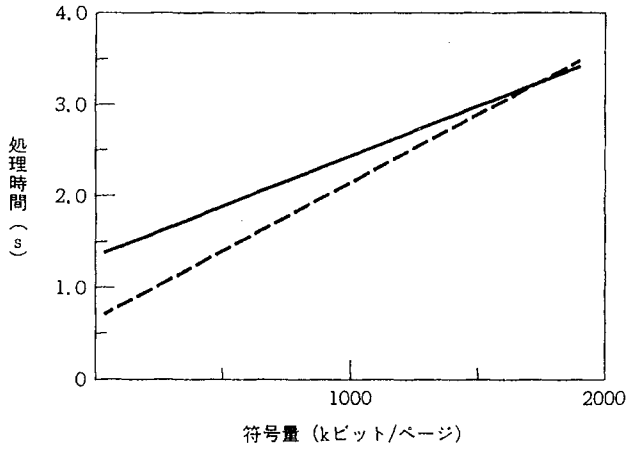
保守／廃止

5.7　処理時間

コーデック命令を使用した場合の，符号化／復号化命令の符号量当たりの処理時間を次に示します。

図5－22　MH符号化／復号化命令の処理時間

(1) 8ビット・バス幅，1ウエイトの場合

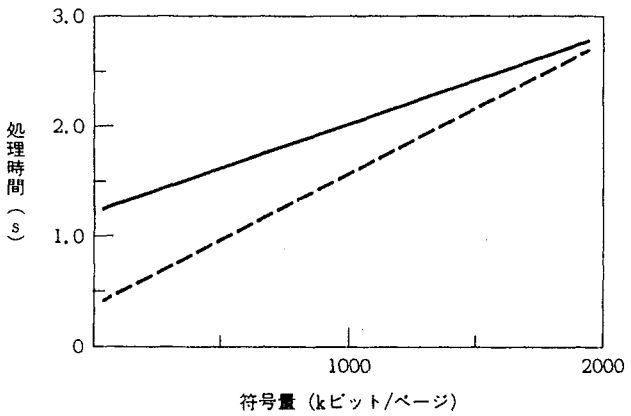


備考1. ——— 符号化命令

----- 復号化命令

2. 測定条件は12.5 MHzです。

(2) 16ビット・バス幅，ノー・ウエイトの場合



備考1. ——— 符号化命令

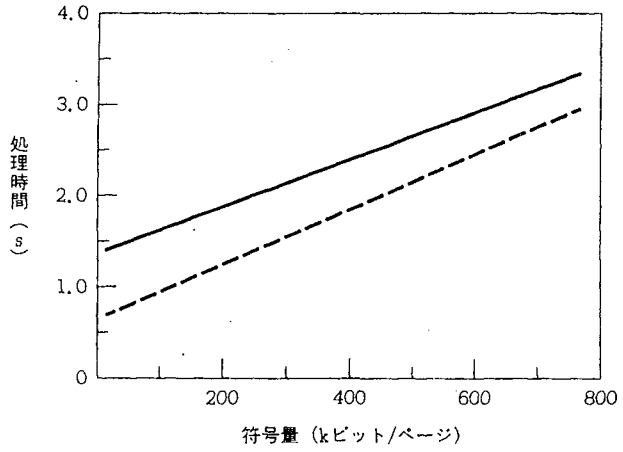
----- 復号化命令

2. 測定条件は12.5 MHzです。

保守／廃止

図5－23　MR符号化／復号化命令の処理時間

(1) 8ビット・バス幅，1ウエイトの場合

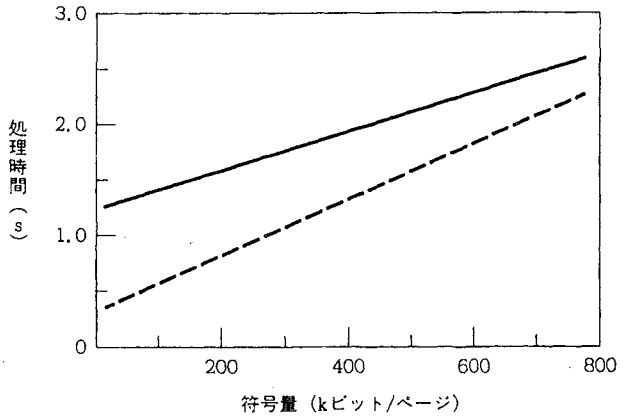


備考1. ——— 符号化命令

----- 復号化命令

2. 測定条件は12.5 MHz，Kファクタ＝4 です。

(2) 16ビット・バス幅，ノー・ウエイトの場合



備考1. ——— 符号化命令

----- 復号化命令

2. 測定条件は12.5 MHz，Kファクタ＝4 です。

付録A　新規プリフィクス命令を付加可能な命令

表A－1に新規プリフィクス命令（DS2：，DS3：，IRAM：）を付加可能な命令を示します。
表内の記号の意味は，次のとおりです。

(1) セグメント

セグメント，拡張セグメント・プリフィクスの付加可能な命令です。

Src：オペランドのソース側にプリフィクスを付加可能です。
Dst：オペランドのデスティネーション側にプリフィクスを付加可能です。

0 ：DS1，DS0，PS，SS，DS2，DS3
　　ディフォールト＝DS0
1 ：DS2，DS1，DS0，PS，SS
　　ディフォールト＝DS0
2 ：DS2，DS0
　　ディフォールト＝DS0
3 ：DS3，DS1
　　ディフォールト＝DS1

備考　第2章内の各命令説明では，【有効オーバライド・プリフィクス】として表しています。

(2) IRAM

IRAMプリフィクスを付加可能な命令です。

表A-1　新規プリフィクス命令を付加可能な命令（1/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
データ転送命令	MOV	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		mem, imm	0		○
		reg, imm			
		acc, dmem		0	○
		dmem, acc	0		○
		sreg, reg16			
		sreg, mem16		0	○
		reg16, sreg			
		mem16, sreg	0		○
		DS0, reg16, mem32		0	○
		DS1, reg16, mem32		0	○
		AH, PSW			
		PSW, AH			
	LDEA	reg16, mem16			
	TRANS TRANSB[注1]	(src-table)		0	○
	XCH	reg, reg'			
		mem, reg/reg, mem		0	○
		AW, reg16/reg16, AW			
	MOVSPA[注2]				
	MOVSPB[注2]	reg16			
リピート・プリフィクス	REPC				
	REPNC				
	REP/ REPE/ REPZ				
	REPNE/ REPNZ				

注1.　ニモニックでB/W（バイト／ワード）で指定している場合は，オペランドなしとなります。

2.　V20，V30に対して，新しく追加された命令です。

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（2/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
プリミティブ・ブロック転送命令	MOVBK/ MOVBKB[注1] MOVBKW[注1]	dst-block, src-block	3	1	○ (Dst)
	CMPBK/ CMPBKB[注1] CMPBKW[注1]	src-block, dst-block	3	1	○ (Dst)
	CMPM/ CMPMB[注1] CMPMW[注1]	dst-block	3		○
	LDM/ LDMB[注1] LDMW[注1]	src-block		1	○
	STM/ STMB[注1] STMW[注1]	dst-block	3		○
ビット・フィールド操作命令	INS[注2]	reg8, reg8'	3		○
		reg8, imm4	3		○
	EXT[注2]	reg8, reg8'	2		○
		reg8, imm4	2		○
入出力命令	IN[注3]	acc, imm8			
		acc, DW			
	OUT[注3]	imm8, acc			
		DW, acc			

注1. ニモニックでB/W（バイト／ワード）で指定している場合は，オペランドなしとなります。

2. ビット・フィールド操作命令にプリフィクスを付加する場合は，オペランドの先頭に付加記述します。

例）INS　　DS3：CH, AH

3. $\overline{IBRK}$＝0のとき，ソフトウエア割り込みが自動的に発生し，命令は実行されません。

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（3/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
入出力命令プリミティブ	INM[注]	dst-block, DW	3		○ (Dst)
	OUTM[注]	DW, src-block		1	○ (Src)
加減算命令	ADD	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
	ADDC	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
	SUB	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			

注　$\overline{\text{IBRK}}$＝0のとき，ソフトウエア割り込みが自動的に発生し，命令は実行されません。

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（4/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
加減算命令	SUBC	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
BCD演算命令	ADD4S[注1]	(dst-string, src-string)	3	1	○ (Dst)
	SUB4S[注1]	(dst-string, src-string)	3	1	○ (Dst)
	CMP4S[注1]	(dst-string, src-string)	3	1	○ (Dst)
	ROL4	reg8			
		mem8		0	○
	ROR4	reg8			
		mem8		0	○
増減命令	INC	reg8			
		mem		0	○
		reg16			
	DEC	reg8			
		mem		0	○
		reg16			
乗算命令	MULU	reg8			
		mem8		0	○
		reg16			
		mem16		0	○
	MUL	reg8			
		mem8		0	○
		reg16			
		mem16		0	○
		reg16, (reg16,)[注2] imm8			
		reg16, mem16, imm8		0	○
		reg16, (reg16,)[注2] imm16			
		reg16, mem16, imm16		0	○

注1．オペランドの省略可能。BCD桁数はCLレジスタで与えられ，1から254の値が設定可能です。

2．第2オペランドは省略可。省略した場合は第1オペランドと同じレジスタを指定したことになります。

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（5/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
符号なし除算命令	DIVU	reg8			
		mem8		0	○
		reg16			
		mem16		0	○
符号つき除算命令	DIV	reg8			
		mem8		0	○
		reg16			
		mem16		0	○
BCD補正命令	ADJBA				
	ADJ4A				
	ADJBS				
	ADJ4S				
データ変換命令	CVTBD				
	CVTDB				
	CVTBW				
	CVTWL				
比較命令	CMP	reg, reg′			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
補数演算命令	NOT	reg			
		mem		0	○
	NEG	reg			
		mem		0	○

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（6/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
論理演算命令	TEST	reg, reg'			
		mem, reg/reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
	AND	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
	OR	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
	XOR	reg, reg'			
		mem, reg	0		○
		reg, mem		0	○
		reg, imm			
		mem, imm	0		○
		acc, imm			
ビット操作命令	TEST1	reg8, CL			
		mem8, CL		0	○
		reg16, CL			
		mem16, CL		0	○
		reg8, imm3			
		mem8, imm3		0	○
		reg16, imm4			
		mem16, imm4		0	○

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（7/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
ビット操作命令	NOT1	reg8, CL			
		mem8, CL		0	○
		reg16, CL			
		mem16, CL		0	○
		reg8, imm3			
		mem8, imm3		0	○
		reg16, imm4			
		mem16, imm4		0	○
	CLR1	reg8, CL			
		mem8, CL		0	○
		reg16, CL			
		mem16, CL		0	○
		reg8, imm3			
		mem8, imm3		0	○
		reg16, imm4			
		mem16, imm4		0	○
	SET1	reg8, CL			
		mem8, CL		0	○
		reg16, CL			
		mem16, CL		0	○
		reg8, imm3			
		mem8, imm3		0	○
		reg16, imm4			
		mem16, imm4		0	○
	NOT1	CY			
	CLR1	CY			
		DIR			
	SET1	CY			
		DIR			
	BSCH	reg8			
		mem8			
		reg16			
		mem16			

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（8/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
シフト命令	SHL	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
	SHR	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
	SHRA	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
ローテート命令	ROL	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
	ROR	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
	ROLC	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○

保守／廃止

表A－1　新規プリフィクス命令を付加可能な命令（9/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
ローテート命令	RORC	reg, 1			
		mem, 1		0	○
		reg, CL			
		mem, CL		0	○
		reg, imm8			
		mem, imm8		0	○
サブルーチン制御命令	CALL	near-proc			
		regptr16			
		memptr16		0	○ (Src)
		far-proc			
		memptr32		0	○ (Src)
	RET	セグメント内			
		pop_value セグメント内			
		セグメント外			
		pop_value セグメント外			
スタック操作命令	PUSH	mem16		0	○ (Src)
		reg16			
		sreg			
		PSW			
		R			
		imm8/imm16			
	POP	mem16		0	○
		reg16			
		sreg			
		PSW			
		R			
	PREPARE	imm16, imm8			
	DISPOSE				

表A－1　新規プリフィクス命令を付加可能な命令（10/11）

命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
ブランチ命令	BR	near-label			
		short-label			
		regptr16			
		memptr16		0	○ (Src)
		far-label			
		memptr32		0	○ (Src)
条件付きブランチ命令	BV	short-label			
	BNV	short-label			
	BC/BL	short-label			
	BNC/BNL	short-label			
	BE/BZ	short-label			
	BNE/BNZ	short-label			
	BNH	short-label			
	BH	short-label			
	BN	short-label			
	BP	short-label			
	BPE	short-label			
	BPO	short-label			
	BLT	short-label			
	BGE	short-label			
	BLE	short-label			
	BGT	short-label			
	DBNZNE	short-label			
	DBNZE	short-label			
	DBNZ	short-label			
	BCWZ	short-label			
	BTCLR[注]	sfr, imm3, short-label			

注　V20, V30に対して，新しく追加された命令です。

表A－1　新規プリフィクス命令を付加可能な命令（11/11）

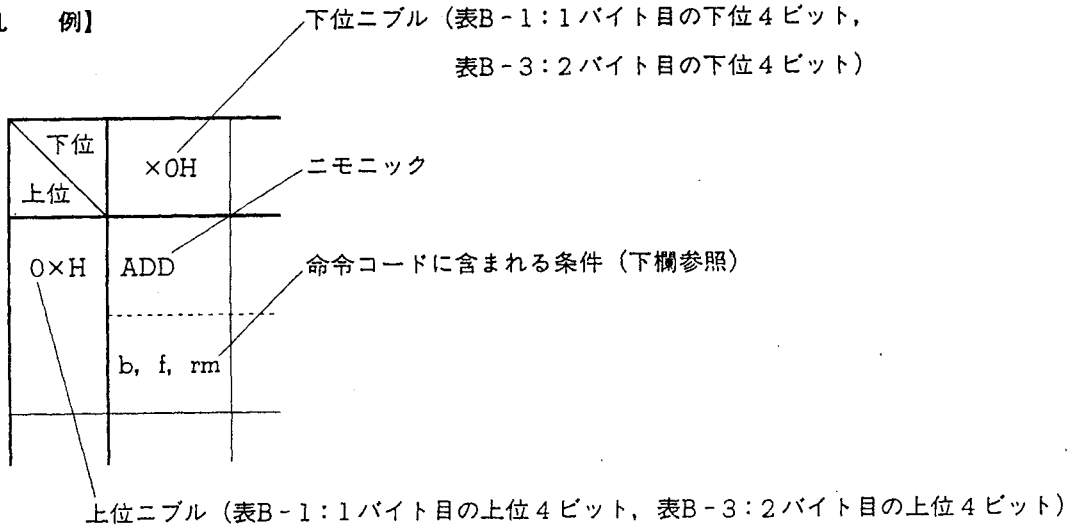
命令群	ニモニック	オペランド	セグメント		IRAM
			Dst	Src	
割り込み命令	BRK	3			
		imm8（≠3）			
	BRKV				
	RETI				
	RETRBI[注1]				
	FINT[注1]				
	CHKIND	reg16, mem32		0	○ (Src)
バンク切替え	BRKCS[注1]	reg16			
	TSKSW[注1]	reg16			
CPU制御命令	HALT				
	STOP[注1]				
	POLL				
	DI				
	EI				
	BUSLOCK				
	FPO1	fp-op			
		fp-op, mem			
	FPO2	fp-op			
		fp-op, mem			
	NOP				
注2 追加命令	PUSH	DS2			
		DS3/VPC			
	POP	DS2			
		DS3/VPC			
	MOV	xsreg, reg16			
		xsreg, mem16		0	○
		reg16, xsreg			
		mem16, xsreg	0		○
		DS2, reg16, mem32		0	○
		DS3, reg16, mem32		0	○
	RSTWDT	imm8, imm8′			
	BTCLRL	sfr1, imm3, short-label			

注1．V20，V30に対して，新しく追加された命令です。

2．V25，V35に対して，新しく追加された命令です。

付録B　命令マップ

【凡　例】



【命令コードに含まれる条件】

b　：バイト動作を行う
d　：ダイレクト・アドレシングを用いる
f　：CPU内のレジスタからのリード動作を伴う
i　：イミーディエト・データを用いる
ia　：イミーディエト・データを用い，アキュムレータへの書き戻しがある
id　：インダイレクト・アドレシングを用いる
l　：セグメント間の制御を伴う
m　：メモリ・データを用いる
reg8：8ビット・レジスタを用いる
rm　：2バイト目に実効アドレス・フィールドを持つ
s　：符号拡張した16ビット・イミーディエト・データを用いる
sr　：セグメント・レジスタを使用する
t　：CPU内のレジスタへのライト動作
v　：インダイレクトでポート番号を指定する
w　：ワード動作を行う

上記以外の記号については**表2－4　命令形式またはオペレーション説明上の凡例**を参照してください。

保守／廃止

表B－1　命令マップ

上位＼下位	×0H	×1H	×2H	×3H	×4H	×5H	×6H	×7H	×8H	×9H	×AH	×BH	×CH	×DH	×EH	×FH
0×H	ADD	ADD	ADD	ADD	ADD	ADD	PUSH	POP	OR	OR	OR	OR	OR	OR	PUSH	Group3
	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia	DS1	DS1	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia	PS	
1×H	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	PUSH	POP	SUBC	SUBC	SUBC	SUBC	SUBC	SUBC	PUSH	POP
	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia	SS	SS	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia	DS0	DS0
2×H	AND	AND	AND	AND	AND	AND	DS1 :	ADJ4A	SUB	SUB	SUB	SUB	SUB	SUB	PS :	ADJ4S
	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia			b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia		
3×H	XOR	XOR	XOR	XOR	XOR	XOR	SS :	ADJBA	CMP	CMP	CMP	CMP	CMP	CMP	DS0 :	ADJBS
	b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia			b, f, rm	w, f, rm	b, t, rm	w, t, rm	b, ia	w, ia		
4×H	INC	INC	INC	INC	INC	INC	INC	INC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	AW	CW	DW	BW	SP	BP	IX	IY	AW	CW	DW	BW	SP	BP	IX	IY
5×H	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	POP	POP	POP	POP	POP	POP	POP	POP
	AW	CW	DW	BW	SP	BP	IX	IY	AW	CW	DW	BW	SP	BP	IX	IY
6×H	PUSH	POP	CHKIND	DS2 :	REPNC	REPC	FPO2	FPO2	PUSH	MUL	PUSH	MUL	INM	INM	OUTM	OUTM
	R	R					0	1	w, i	w, i	s, i	s, i	b	w	b	w
7×H	BV	BNV	BC BL	BNC BNL	BE BZ	BNE BNZ	BNH	BH	BN	BP	BPE	BPO	BLT	BGE	BLE	BGT
8×H	Imm	Imm	Imm	Imm	TEST	TEST	XCH	XCH	MOV	MOV	MOV	MOV	MOV	LDEA	MOV	POP
	b, rm	w, rm	b, s, rm	w, s, rm	b, rm	w, rm	b, rm	w, rm	b, f, rm	w, f, rm	b, t, rm	w, t, rm	sr, f, rm		sr, t, rm	rm
9×H	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	CVTBW	CVTWL	CALL	POLL	PUSH	POP	MOV	MOV
	AW	CW	DW	BW	SP	BP	IX	IY			I, d		PSW	PSW	PSW, AH	AH, PSW
A×H	MOV	MOV	MOV	MOV	MOVBK MOVBKB MOVBKW	MOVBK MOVBKB MOVBKW	CMPBK CMPBKB CMPBKW	CMPBK CMPBKB CMPBKW	TEST	TEST	STM STMB STMW	STM STMB STMW	LDM LDMB LDMW	LDM LDMB LDMW	CMPM CMPMB CMPMW	CMPM CMPMB CMPMW
	AL, m	AW, m	m, AL	m, AW	b	w	b	w	b, ia	w, ia	b	w	b	w	b	w
B×H	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
	AL, i	CL, i	DL, i	BL, i	AH, i	CH, i	DH, i	BH, i	AW, i	CW, i	DW, i	BW, i	SP, i	BP, i	IX, i	IY, i
C×H	Shift	Shift	RET	RET	MOV	MOV	MOV	MOV	PREPARE	DISPOSE	RET	RET	BRK	BRK	BRKV	RETI
	b, i	w, i	(SP)		DS1	DS0	b, i, rm	w, i, rm			I, (SP)	I	3	i		
D×H	Shift	Shift	Shift	Shift	CVTBD	CVTDB	DS3 :	TRANS TRANSB	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1
	b	w	b, v	w, v					0	1	2	3	4	5	6	7
E×H	DBNZE	DBNZE	DBNZ	BCWZ	IN	IN	OUT	OUT	CALL	BR	BR	BR	IN	IN	OUT	OUT
					b	w	b	w	d	d	I, d	sI, d	b, v	w, v	b, v	w, v
F×H	BUSLOCK	IRAM :	REPNE REPNZ	REP REPE REPZ	HALT	NOT1	Group1	Group1	CLR1	SET1	DI	EI	CLR1	SET1	Group2	Group2
						CY	b	w	CY	CY			DIR	DIR	b	w

注意　□：Group1, Group2, Imm, Shiftは命令コードの２バイト目のビット３－ビット５によって命令が定まります（表B－２参照）。

Group3は命令コードの２バイト目によって命令が定まります（表B－３参照）。

表B－2　Group1，Group2，Imm，Shiftコード表

注	000	001	010	011	100	101	110	111
Imm	ADD	OR	ADDC	SUBC	AND	SUB	XOR	CMP
Shift	ROL	ROR	ROLC	RORC	SHL	SHR	未定義	SHRA
Group1	TEST	未定義	NOT	NEG	MULU	MUL	DIVU	DIV
	rm		rm	rm	rm	rm	rm	rm
Group2	INC	DEC	CALL	CALL	BR	BR	PUSH	未定義
	rm	rm	id	I, id	id	I, id	rm	

注　2バイト目のビット5－ビット3

表B－3　Group3コード表

下位 / 上位	×0H	×1H	×2H	×3H	×4H	×5H	×6H	×7H	×8H	×9H	×AH	×BH	×CH	×DH	×EH	×FH
0×H																
1×H	TEST1	TEST1	CLR1	CLR1	SET1	SET1	NOT1	NOT1	TEST1	TEST1	CLR1	CLR1	SET1	SET1	NOT1	NOT1
	b	w	b	w	b	w	b	w	i, b	i, w	i, b	i, w	i, b	i, w	i, b	i, w
2×H	ADD4S		SUB4S				CMP4S		ROL4		ROR4			BRKCS		
3×H		INS		EXT						INS		EXT	BSCH	BSCH		
		reg8		reg8						i		i	b	w		
7×H	QHOUT	QOUT	QTIN						SCHEOL	GETBIT	CNVTRP		MHDEC	MRDEC		
8×H																
9×H		RETRBI	FINT	MHENC	TSKSW		RSTWDT	MRENC			ALBIT	COLTRP	BTCLR	BTCLRL	STOP	
F×H																

備考　空白の欄は未定義コードです。

保守／廃止

〔メ　モ〕

保守／廃止

付録C　μPD8086，8088とのニモニック対応表

V55PIの命令セットは，オブジェクト・コード・レベルで，μPD8086, 8088に対して上位互換性があります。

表C－1にμPD8086，8088とV55PIのニモニックの対応を示します。

なお，以下の命令については，μPD8086，8088には対応する命令がありません。

ALBIT，BRKCS，BSCH，BTCLR，BTCLRL，CHKIND，CMP4S，CNVTRP，COLTRP，DS2：，DS3：，EXT，FINT，FPO2，GETBIT，INM，INS，IRAM：，MHDEC，MHENC，MOVSPA，MOVSPB，MRDEC，MRENC，OUTM，PREPARE，QHOUT，QOUT，QTIN，REPC，REPNC，RETRBI，ROL4，ROR4，RSTWDT，SCHEOL，STOP，SUB4S，TEST1，TSKSW

表C－1　μPD8086，8088とのニモニック対応表

μPD8086, 8088	V55PI	μPD8086, 8088	V55PI	μPD8086, 8088	V55PI	μPD8086, 8088	V55PI
AAA	ADJBA	JB	BC/BL	LOOP	DBNZ	SHR	SHR
AAD	CVTDB	JBE	BNH	LOOPE	DBNZE	SS :	SS :
AAM	CVTBD	JC	BC/BL	LOOPNE	DBNZNE	STC	SET1 CY
AAS	ADJBS	JCXZ	BCWZ	LOOPNZ	DBNZNE	STD	SET1 DIR
ADC	ADDC	JE	BE/BZ	LOOPZ	DBNZE	STI	EI
ADD	ADD	JG	BGT	MOV	MOV	STOS	STM/STMB/
AND	AND	JGE	BGE	MOVS	MOVBK		STMW
CALL	CALL	JL	BLT	MOVSB	MOVBKB	SUB	SUB
CBW	CVTBW	JLE	BLE	MOVSW	MOVBKW	TEST	TEST
CLC	CLR1 CY	JMP	BR	MUL	MULU	WAIT	POLL
CLD	CLR1 DIR	JNA	BNH	NEG	NEG	XCHG	XCH
CLI	DI	JNAE	BC/BL	NOP	NOP	XLAT	TRANS
CMC	NOT1 CY	JNB	BNC/BNL	NOT	NOT	XLATB	TRANSB
CMP	CMP	JNBE	BH	OR	OR	XOR	XOR
CMPS	CMPBK/	JNC	BNC/BNL	OUT	OUT		
	CMPBKB/	JNE	BNE/BNZ	POP	POP		
	CMPBKW	JNG	BLE	POPF	POP PSW		
CS :	PS :	JNGE	BLT	PUSH	PUSH		
CWD	CVTWL	JNL	BGE	PUSHF	PUSH PSW		
DAA	ADJ4A	JNLE	BGT	RCL	ROLC		
DAS	ADJ4S	JNO	BNV	RCR	RORC		
DEC	DEC	JNP	BPO	REP	REP		
DIV	DIVU	JNS	BP	REPE	REPE		
DS :	DS0 :	JNZ	BNE/BNZ	REPNE	REPNE		
ES :	DS1 :	JO	BV	REPNZ	REPNZ		
ESC	FPO1	JP	BPE	REPZ	REPZ		
HLT	HALT	JPE	BPE	RET	RET		
IDIV	DIV	JPO	BPO	ROL	ROL		
IMUL	MUL	JS	BN	ROR	ROR		
IN	IN	JZ	BE/BZ	SAHF	MOV PSW, AH		
INC	INC	LAHF	MOV AH, PSW	SAL	SHL		
INT	BRK	LDS	MOV DS0,	SAR	SHRA		
INT 3	BRK 3	LEA	LDEA	SBB	SUBC		
INTO	BRKV	LES	MOV DS1,	SCAS	CMPM/		
IRET	RETI	LOCK	BUSLOCK		CMPMB/		
JA	BH	LODS	LDM/LDMB/		CMPMW		
JAE	BNC/BNL		LDMW	SHL	SHL		

付録D　他社アセンブラ，Cコンパイラでの開発手法

ここでは，他社アセンブラ，Cコンパイラを使用して，V55PIの応用ソフトウエア開発を行うための手法について説明します。

D.1　アセンブラでの開発

(1) 対象アセンブラ

ここでは，パソコン上のアセンブラとして最も普及している次のアセンブラを対象とします。

MASM V5.1（米国Microsoft Corp.）

その他のアセンブラを使用する場合でも，アセンブラのマクロ機能を使うことにより同様に開発することができます。ただし，アセンブラでサポートしているマクロ機能により制限が異なります。

(2) 使用方法

ソース・プログラムの先頭で，**D.3　添付リスト(1) V55PI.MAC**の内容をインクルードしてください。

(3) 制限事項

① 次の命令は，メモリのアドレシング・モードに対応してニモニックが異なります。必要に応じて，V55PI.MACにマクロを追加してください。

- MOV xsreg, mem16
- MOV mem16, xsreg
- MOV xsreg, reg16, mem32
- BSCH mem

例　MOV xsreg, mem16

アドレシング・モード［BW+IX］の場合

```
mov_bw_ix   MACRO  para1, para2
            IFIDN   <para1>,  <ds2>
                    DB      8eH, 38H
            ENDIF
            IFIDN   <para1>,  <ds3>
                    DB      8eH, 30H
            ENDIF
            ENDM
```

② 次の命令は，ニモニックを変更する必要があります。

- MOV xsreg, reg16 → MOV55 xsreg, reg16
- MOV reg16, xsreg → MOV55 reg16, xsreg
- PUSH xsreg → PUSH55 xsreg
- POP xsreg → POP55 xsreg
- ds2： → ds2fx
- ds3： → ds3fx
- iram： ← iram

③ 次の命令は，ほかの命令中に記述できません。

- ds2：
- ds3：
- iram：

例 MOV ds2：a, reg → ds2fx
MOV a, reg

D.2 Cコンパイラでの開発

(1) V55PIの専用命令について

V55PIの命令はV20，V30に対して上位互換性があるため，他社86系Cコンパイラを使用できます。V55PIの専用命令を使う場合は，アセンブラで作成した関数を呼び出します。

(2) 特殊機能レジスタについて

特殊機能レジスタ群(SFR)は，メモリ上にマッピングされています。したがって，ポインタ変数を使用して参照／代入することで操作できます。

保守／廃止

(3) 特殊機能レジスタの使用方法

ソース・プログラムの先頭で，**D.3　添付リスト(2) SFR_PI.H**の内容をインクルードしてください。

例　#include <SFR_PI.H>

```
struct SFR *sfradr;
sfradr= (struct SFR*) 0xffe00000;      /*SFR先頭アドレスの設定  */
PMC2='A';
```

注意　以降に示すリストは，ご使用するコンパイラで十分評価のうえお使いください。
特に，最適化を行うコンパイラでは，**SFR**へ値を代入してもそれを参照しない場合は，Cコンパイラが冗長な命令と解釈し，コード生成を行わない場合などがあります。

保守／廃止

D.3　添付リスト

(1) V55PI.MAC

```
;*********************************************************************
;*      Sample for V55PI programing                                  *
;*                              for MASM V5.1                        *
;*                                      (C) NEC Corp. 1993           *
;*********************************************************************

;***********************************
;       SFR MACRO FOR V55PI
;***********************************

SFR_433 struc
        adcr0           DB      ?
        dummy1          DB      ?
        adcr1           DB      ?
        dummy2          DB      ?
        adcr2           DB      ?
        dummy3          DB      ?
        adcr3           DB      ?
        dummy4          DB  9   DUP(?)
        pab             DB      ?
        dummy5          DB  7   DUP(?)
        pac0            DB      ?
        pac1            DB      ?
        pas             DB      ?
        dummy6          DB      ?
        pai1            DB      ?
        pai2            DB      ?
        dummy7          DB  2   DUP(?)
        adm             DB      ?
        dummy8          DB  159 DUP(?)
        mk01            DB      ?
        mk0h            DB      ?
        mk11            DB      ?
        mk1h            DB      ?
        ispr            DB      ?
        imc             DB      ?
        dummy9          DB  3   DUP(?)
        ic09            DB      ?
        ic10            DB      ?
        ic11            DB      ?

        ic12            DB      ?
        ic13            DB      ?
        ic14            DB      ?
        dummy10         DB      ?
        ic16            DB      ?
        ic17            DB      ?
        ic18            DB      ?
        ic19            DB      ?
        ic20            DB      ?
        ic21            DB      ?
        ic22            DB      ?
        ic23            DB      ?
```

```
ic24            DB      ?
ic25            DB      ?
ic26            DB      ?
ic27            DB      ?
ic28            DB      ?
ic29            DB      ?
ic30            DB      ?
ic31            DB      ?
ic32            DB      ?
dummy11         DB  3   DUP(?)
ic36            DB      ?
ic37            DB      ?
dummy12         DB  26  DUP(?)

p0              DB      ?
p1              DB      ?
p2              DB      ?
p3              DB      ?
p4              DB      ?
p5              DB      ?
p6              DB      ?
p7              DB      ?
p8              DB      ?
dummy13         DB  3   DUP(?)
prdc            DB      ?
dummy14         DB      ?
rtp             DB      ?
dummy15         DB      ?
pm0             DB      ?
dummy16         DB      ?
pm2             DB      ?
pm3             DB      ?
pm4             DB      ?
pm5             DB      ?
dummy17         DB      ?
pm7             DB      ?
pm8             DB      ?
dummy18         DB  9   DUP(?)
pmc2            DB      ?
pmc3            DB      ?
pmc4            DB      ?
pmc5            DB      ?
dummy19         DB      ?

pmc7            DB      ?
pmc8            DB      ?
dummy20         DB  3   DUP(?)
rtpc            DB      ?
rtpd            DB      ?
p71             DB      ?
p7h             DB      ?
tmc0            DB      ?
tmc1            DB      ?
```

```
toc0            DB      ?
toc1            DB      ?
intm0           DB      ?
intm1           DB      ?
dummy21         DB  10  DUP(?)
tm0             DW      ?
tm1             DW      ?
tm2             DW      ?
tm3             DW      ?
ct00            DW      ?
ct01            DW      ?
cm00            DW      ?
cm01            DW      ?
ct10            DW      ?
cm10            DW      ?

cm11            DW      ?
dummy22         DB  2   DUP(?)
cm20            DW      ?
cm21            DW      ?
cm22            DW      ?
cm23            DW      ?
wdm             DB      ?
dummy23         DB  3   DUP(?)
cm30            DW      ?
cm31            DW      ?
dummy24         DB  4   DUP(?)
pwm             DB      ?
pwmc            DB      ?
dummy25         DB  2   DUP(?)
txbrg0          DB      ?
rxbrg0          DB      ?
prs0            DB      ?
uartm0          DB      ?
uarts0          DB      ?
txb0            DB      ?
rxb0            DB      ?
dummy26         DB      ?

txbrg1          DB      ?
rxbrg1          DB      ?
prs1            DB      ?
uartm1          DB      ?
uarts1          DB      ?
txb1            DB      ?
rxb1            DB      ?
asp             DB      ?
tc0l            DW      ?
tc0h            DW      ?
tcm0l           DW      ?
tcm0h           DW      ?
udc0l           DW      ?
udc0h           DW      ?
```

```
        dcm0l           DW      ?
        dcm0h           DW      ?
        mar0l           DW      ?
        mar0h           DW      ?

        dptc0l          DW      ?
        dptc0h          DW      ?
        dummy27         DB  4   DUP(?)
        dmam0           DB      ?
        dmac0           DB      ?
        dmas            DB      ?
        dummy28         DB      ?
        tc1l            DW      ?
        tc1h            DW      ?
        tcm1l           DW      ?
        tcm1h           DW      ?
        udc1l           DW      ?
        udc1h           DW      ?
        dcm1l           DW      ?
        dcm1h           DW      ?
        mar1l           DW      ?
        mar1h           DW      ?
        dptc1l          DW      ?
        dptc1h          DW      ?
        dummy29         DB  4   DUP(?)
        dmam1           DB      ?
        dmac1           DB      ?
        dummy30         DB  34  DUP(?)
        stc_            DW      ?
        stmc            DW      ?
        dummy31         DB  4   DUP(?)

        pwc0            DB      ?
        pwc1            DB      ?
        mbc             DB      ?
        dummy32         DB      ?
        rfm             DB      ?
        dummy33         DB      ?
        stbc            DB      ?
        prc             DB      ?
SFR_433 ends


;*********************************************
;         V55PI Extend instructions
;*********************************************

mov55   MACRO   para1,para2
        IFIDN   <para1>,<ds2>
                IFIDN   <para2>,<ax>
                        DB      8eH,0f8H
                ENDIF
                IFIDN   <para2>,<cx>
```

```
                DB      8eH,0f9H
        ENDIF
        IFIDN   <para2>,<dx>
                DB      8eH,0faH
        ENDIF
        IFIDN   <para2>,<bx>
                DB      8eH,0fbH
        ENDIF
        IFIDN   <para2>,<sp>
                DB      8eH,0fcH
        ENDIF
        IFIDN   <para2>,<bp>
                DB      8eH,0fdH
        ENDIF
        IFIDN   <para2>,<si>
                DB      8eH,0feH
        ENDIF
        IFIDN   <para2>,<di>
                DB      8eH,0ffH
        ENDIF
ENDIF
IFIDN   <para1>,<ds3>
        IFIDN   <para2>,<ax>
                DB      8eH,0f0H
        ENDIF
        IFIDN   <para2>,<cx>
                DB      8eH,0f1H
        ENDIF
        IFIDN   <para2>,<dx>
                DB      8eH,0f2H
        ENDIF
        IFIDN   <para2>,<bx>
                DB      8eH,0f3H
        ENDIF
        IFIDN   <para2>,<sp>
                DB      8eH,0f4H
        ENDIF
        IFIDN   <para2>,<bp>
                DB      8eH,0f5H
        ENDIF
        IFIDN   <para2>,<si>
                DB      8eH,0f6H
        ENDIF
        IFIDN   <para2>,<di>
                DB      8eH,0f7H
        ENDIF
ENDIF
IFIDN   <para2>,<ds2>
        IFIDN   <para1>,<ax>
                DB      8cH,0f8H
        ENDIF
        IFIDN   <para1>,<cx>
                DB      8cH,0f9H
```

```
                ENDIF
                IFIDN   <para1>,<dx>
                        DB      8cH,0faH
                ENDIF
                IFIDN   <para1>,<bx>
                        DB      8cH,0fbH
                ENDIF
                IFIDN   <para1>,<sp>
                        DB      8cH,0fcH
                ENDIF
                IFIDN   <para1>,<bp>
                        DB      8cH,0fdH
                ENDIF
                IFIDN   <para1>,<si>
                        DB      8cH,0feH
                ENDIF
                IFIDN   <para1>,<di>
                        DB      8cH,0ffH
                ENDIF
        ENDIF
        IFIDN   <para2>,<ds3>
                IFIDN   <para1>,<ax>
                        DB      8cH,0f0H
                ENDIF
                IFIDN   <para1>,<cx>
                        DB      8cH,0f1H
                ENDIF
                IFIDN   <para1>,<dx>
                        DB      8cH,0f2H
                ENDIF
                IFIDN   <para1>,<bx>
                        DB      8cH,0f3H
                ENDIF
                IFIDN   <para1>,<sp>
                        DB      8cH,0f4H
                ENDIF
                IFIDN   <para1>,<bp>
                        DB      8cH,0f5H
                ENDIF
                IFIDN   <para1>,<si>
                        DB      8cH,0f6H
                ENDIF
                IFIDN   <para1>,<di>
                        DB      8cH,0f7H
                ENDIF
        ENDIF
        ENDM
movspa  MACRO
        DB      0fH,25H
        ENDM
movspb  MACRO   para1
        IFIDN   <para1>,<ax>
                DB      0fH,95H,0f8H
```

```
        ENDIF
        IFIDN   <para1>,<cx>
                DB      0fH,95H,0f9H
        ENDIF
        IFIDN   <para1>,<dx>
                DB      0fH,95H,0faH
        ENDIF
        IFIDN   <para1>,<bx>
                DB      0fH,95H,0fbH
        ENDIF
        IFIDN   <para1>,<sp>
                DB      0fH,95H,0fcH
        ENDIF
        IFIDN   <para1>,<bp>
                DB      0fH,95H,0fdH
        ENDIF
        IFIDN   <para1>,<si>
                DB      0fH,95H,0feH
        ENDIF
        IFIDN   <para1>,<di>
                DB      0fH,95H,0ffH
        ENDIF
        ENDM
push55  MACRO   para1
        IFIDN   <para1>,<ds2>
                DB      0fH,7eH
        ENDIF
        IFIDN   <para1>,<ds3>
                DB      0fH,76H
        ENDIF
        ENDM
pop55   MACRO   para1
        IFIDN   <para1>,<ds2>
                DB      0fH,7fH
        ENDIF
        IFIDN   <para1>,<ds3>
                DB      0fH,77H
        ENDIF
        ENDM
brkcs   MACRO   para1
        IFIDN   <para1>,<ax>
                DB      0fH,2dH,0c0H
        ENDIF
        IFIDN   <para1>,<cx>
                DB      0fH,2dH,0c1H
        ENDIF
        IFIDN   <para1>,<dx>
                DB      0fH,2dH,0c2H
        ENDIF
        IFIDN   <para1>,<bx>
                DB      0fH,2dH,0c3H
        ENDIF
        IFIDN   <para1>,<sp>
```

```
                DB      0fH,2dH,0c4H
        ENDIF
        IFIDN   <para1>,<bp>
                DB      0fH,2dH,0c5H
        ENDIF
        IFIDN   <para1>,<si>
                DB      0fH,2dH,0c6H
        ENDIF
        IFIDN   <para1>,<di>
                DB      0fH,2dH,0c7H
        ENDIF
        ENDM
tsksw   MACRO   para1
        IFIDN   <para1>,<ax>
                DB      0fH,94H,0f8H
        ENDIF
        IFIDN   <para1>,<cx>
                DB      0fH,94H,0f9H
        ENDIF
        IFIDN   <para1>,<dx>
                DB      0fH,94H,0faH
        ENDIF
        IFIDN   <para1>,<bx>
                DB      0fH,94H,0fbH
        ENDIF
        IFIDN   <para1>,<sp>
                DB      0fH,94H,0fcH
        ENDIF
        IFIDN   <para1>,<bp>
                DB      0fH,94H,0fdH
        ENDIF
        IFIDN   <para1>,<si>
                DB      0fH,94H,0feH
        ENDIF
        IFIDN   <para1>,<di>
                DB      0fH,94H,0ffH
        ENDIF
        ENDM
retrbi  MACRO
        DB      0fH,91H
        ENDM
fint    MACRO
        DB      0fH,92H
        ENDM
stop    MACRO
        DB      0fH,9eH
        ENDM
rstwdt  MACRO   para1,para2
        DB      0fH,96H,para1,para2
        ENDM
```

保守／廃止

```
ds2fx   MACRO
        DB      63H
        ENDM
ds3fx   MACRO
        DB      0d6H
        ENDM
iram    MACRO
        DB      0f1H
        ENDM
albit   MACRO           ; only 70433
        DB      0fH,9aH
        ENDM
coltrp  MACRO           ; only 70433
        DB      0fH,9bH
        ENDM
mhenc   MACRO           ; only 70433
        DB      0fH,93H
        ENDM
mrenc   MACRO           ; only 70433
        DB      0fH,97H
        ENDM
scheol  MACRO           ; only 70433
        DB      0fH,78H
        ENDM
getbit  MACRO           ; only 70433
        DB      0fH,79H
        ENDM
mhdec   MACRO           ; only 70433
        DB      0fH,7cH
        ENDM
mrdec   MACRO           ; only 70433
        DB      0fH,7dH
        ENDM
cnvtrp  MACRO           ; only 70433
        DB      0fH,7aH
        ENDM
qhout   MACRO   para1
        DB      0fH,70H
        DW      para1
        ENDM
qout    MACRO   para1
        DB      0fH,71H
        DW      para1
        ENDM
qtin    MACRO   para1
        DB      0fH,72H
        DW      para1
        ENDM
bsch    MACRO   para1
        IFIDN   <para1>,<ax>
                DB      0fH,3dH,0c0H
        ENDIF
        IFIDN   <para1>,<cx>
```

```
        DB      0fH,3dH,0c1H
    ENDIF
    IFIDN   <para1>,<dx>
        DB      0fH,3dH,0c2H
    ENDIF
    IFIDN   <para1>,<bx>
        DB      0fH,3dH,0c3H
    ENDIF
    IFIDN   <para1>,<sp>
        DB      0fH,3dH,0c4H
    ENDIF
    IFIDN   <para1>,<bp>
        DB      0fH,3dH,0c5H
    ENDIF
    IFIDN   <para1>,<si>
        DB      0fH,3dH,0c6H
    ENDIF
    IFIDN   <para1>,<di>
        DB      0fH,3dH,0c7H
    ENDIF
    IFIDN   <para1>,<al>
        DB      0fH,3cH,0c0H
    ENDIF
    IFIDN   <para1>,<cl>
        DB      0fH,3cH,0c1H
    ENDIF
    IFIDN   <para1>,<dl>
        DB      0fH,3cH,0c2H
    ENDIF
    IFIDN   <para1>,<bl>
        DB      0fH,3cH,0c3H
    ENDIF
    IFIDN   <para1>,<ah>
        DB      0fH,3cH,0c4H
    ENDIF
    IFIDN   <para1>,<ch>
        DB      0fH,3cH,0c5H
    ENDIF
    IFIDN   <para1>,<dh>
        DB      0fH,3cH,0c6H
    ENDIF
    IFIDN   <para1>,<bh>
        DB      0fH,3cH,0c7H
    ENDIF
    ENDM

btclr1  MACRO   para1,para2,para3
        DB      0fH,9dH,offset para1,para2,offset para3-$-1
        ENDM
```

保守／廃止

(2) SFR_PI.H

```
/*
        This structure is defined for the special function registor of V55PI.

        Date 08 July 93
        Copyright(C) NEC Corporation 1993
*/
struct  SFR {
        char    adcr0;
        char    dummy1[1];
        char    adcr1;
        char    dummy2[1];
        char    adcr2;
        char    dummy3[1];
        char    adcr3;
        char    dummy4[9];
        char    pab;
        char    dummy5[7];
        char    pac0;
        char    pac1;
        char    pas;
        char    dummy6[1];
        char    pai1;
        char    pai2;
        char    dummy7[2];
        char    adm;
        char    dummy8[159];
        char    mk0l;
        char    mk0h;
        char    mk1l;
        char    mk1h;
        char    ispr;
        char    imc;
        char    dummy9[3];
        char    ic09;
        char    ic10;
        char    ic11;

        char    ic12;
        char    ic13;
        char    ic14;
        char    dummy10[1];
        char    ic16;
        char    ic17;
        char    ic18;
        char    ic19;
        char    ic20;
        char    ic21;
        char    ic22;
        char    ic23;
        char    ic24;
        char    ic25;
        char    ic26;
        char    ic27;
        char    ic28;
        char    ic29;
```

```
char    ic30;
char    ic31;
char    ic32;
char    dummy11[3];
char    ic36;
char    ic37;
char    dummy12[26];

char    p0;
char    p1;
char    p2;
char    p3;
char    p4;
char    p5;
char    p6;
char    p7;
char    p8;
char    dummy13[3];
char    prdc;
char    dummy14[1];
char    rtp;
char    dummy15[1];
char    pm0;
char    dummy16[1];
char    pm2;
char    pm3;
char    pm4;
char    pm5;
char    dummy17[1];
char    pm7;
char    pm8;
char    dummy18[9];
char    pmc2;
char    pmc3;
char    pmc4;
char    pmc5;
char    dummy19[1];

char    pmc7;
char    pmc8;
char    dummy20[3];
char    rtpc;
char    rtpd;
char    p7l;
char    p7h;
char    tmc0;
char    tmc1;
char    toc0;
char    toc1;
char    intm0;
char    intm1;
char    dummy21[10];
int     tm0;
int     tm1;
int     tm2;
```

保守／廃止

```
int     tm3;
int     ct00;
int     ct01;
int     cm00;
int     cm01;
int     ct10;
int     cm10;

int     cm11;
char    dummy22[2];
int     cm20;
int     cm21;
int     cm22;
int     cm23;
char    wdm;
char    dummy23[3];
int     cm30;
int     cm31;
char    dummy24[4];
char    pwm;
char    pwmc;
char    dummy25[2];
char    txbrg0;
char    rxbrg0;
char    prs0;
char    uartm0;         /**/
char    uarts0;         /**/
char    txb0;           /**/
char    rxb0;
char    dummy26[1];

char    txbrg1;
char    rxbrg1;
char    prs1;
char    uartm1;         /**/
char    uarts1;
char    txb1;           /**/
char    rxb1;
char    asp;
int     tc01;
int     tc0h;
int     tcm01;
int     tcm0h;
int     udc01;
int     udc0h;
int     dcm01;
int     dcm0h;
int     mar01;
int     mar0h;

int     dptc01;
int     dptc0h;
char    dummy27[4];
char    dmam0;
char    dmac0;
```

```
        char    dmas;
        char    dummy28[1];
        int     tc1l;
        int     tc1h;
        int     tcm1l;
        int     tcm1h;
        int     udc1l;
        int     udc1h;
        int     dcm1l;
        int     dcm1h;
        int     mar1l;
        int     mar1h;
        int     dptc1l;
        int     dptc1h;
        char    dummy29[4];
        char    dmam1;
        char    dmac1;
        char    dummy30[34];
        int     stc;
        int     stmc;
        char    dummy31[4];

        char    pwc0;
        char    pwc1;
        char    mbc;
        char    dummy32[1];
        char    rfm;
        char    dummy33[1];
        char    stbc;
        char    prc;
};


/*
        This define name is special function registor name.

        Date 08 July 93
        Copyright(C) NEC Corporation 1993
*/


#define         ADCR0   SFR.adcr0;
#define         ADCR1   SFR.adcr1;
#define         ADCR2   SFR.adcr2;
#define         ADCR3   SFR.adcr3;
#define         PAB     SFR.pab;
#define         PAC0    SFR.pac0;
#define         PAC1    SFR.pac1;
#define         PAS     SFR.pas;
#define         PAI1    SFR.pai1;
#define         PAI2    SFR.pai2;
#define         ADM     SFR.adm;
#define         MK0L    SFR.mk0l;
#define         MK0H    SFR.mk0h;
#define         MK1L    SFR.mk1l;
```

```
#define         MK1H    SFR.mk1h;
#define         ISPR    SFR.ispr;
#define         IMC     SFR.imc;
#define         IC09    SFR.ic09;
#define         IC10    SFR.ic10;
#define         IC11    SFR.ic11;

#define         IC12    SFR.ic12;
#define         IC13    SFR.ic13;
#define         IC14    SFR.ic14;
#define         IC16    SFR.ic16;
#define         IC17    SFR.ic17;
#define         IC18    SFR.ic18;
#define         IC19    SFR.ic19;
#define         IC20    SFR.ic20;
#define         IC21    SFR.ic21;
#define         IC22    SFR.ic22;
#define         IC23    SFR.ic23;
#define         IC24    SFR.ic24;
#define         IC25    SFR.ic25;
#define         IC26    SFR.ic26;
#define         IC27    SFR.ic27;
#define         IC28    SFR.ic28;
#define         IC29    SFR.ic29;
#define         IC30    SFR.ic30;
#define         IC31    SFR.ic31;
#define         IC32    SFR.ic32;
#define         IC36    SFR.ic36;
#define         IC37    SFR.ic37;

#define         P0      SFR.p0;
#define         P1      SFR.p1;
#define         P2      SFR.p2;
#define         P3      SFR.p3;
#define         P4      SFR.p4;
#define         P5      SFR.p5;
#define         P6      SFR.p6;
#define         P7      SFR.p7;
#define         P8      SFR.p8;
#define         PRDC    SFR.prdc;
#define         RTP     SFR.rtp;
#define         PM0     SFR.pm0;
#define         PM2     SFR.pm2;
#define         PM3     SFR.pm3;
#define         PM4     SFR.pm4;
#define         PM5     SFR.pm5;
#define         PM7     SFR.pm7;
#define         PM8     SFR.pm8;
#define         PMC2    SFR.pmc2;
#define         PMC3    SFR.pmc3;
#define         PMC4    SFR.pmc4;
#define         PMC5    SFR.pmc5;

#define         PMC7    SFR.pmc7;
#define         PMC8    SFR.pmc8;
```

保守／廃止

```
#define         RTPC    SFR.rtpc;
#define         RTPD    SFR.rtpd;
#define         P7L     SFR.p7l;
#define         P7H     SFR.p7h;
#define         TMC0    SFR.tmc0;
#define         TMC1    SFR.tmc1;
#define         TOC0    SFR.toc0;
#define         TOC1    SFR.toc1;
#define         INTM0   SFR.intm0;
#define         INTM1   SFR.intm1;
#define         TM0     SFR.tm0;
#define         TM1     SFR.tm1;
#define         TM2     SFR.tm2;
#define         TM3     SFR.tm3;
#define         CT00    SFR.ct00;
#define         CT01    SFR.ct01;
#define         CM00    SFR.cm00;
#define         CM01    SFR.cm01;
#define         CT10    SFR.ct10;
#define         CM10    SFR.cm10;

#define         CM11    SFR.cm11;
#define         CM20    SFR.cm20;
#define         CM21    SFR.cm21;
#define         CM22    SFR.cm22;
#define         CM23    SFR.cm23;
#define         WDM     SFR.wdm;
#define         CM30    SFR.cm30;
#define         CM31    SFR.cm31;
#define         PWM     SFR.pwm;
#define         PWMC    SFR.pwmc;
#define         TXBRG0  SFR.txbrg0;
#define         RXBRG0  SFR.rxbrg0;
#define         PRS0    SFR.prs0;
#define         UARTM0  SFR.uartm0;     /**/
#define         CSIM0   SFR.uartm0;     /**/
#define         UARTS0  SFR.uarts0;     /**/
#define         SBIC0   SFR.uarts0;     /**/
#define         TXB0    SFR.txb0;       /**/
#define         SIO0    SFR.txb0;       /**/
#define         RXB0    SFR.rxb0;

#define         TXBRG1  SFR.txbrg1;
#define         RXBRG1  SFR.rxbrg1;
#define         PRS1    SFR.prs1;
#define         UARTM1  SFR.uartm1;     /**/
#define         CSM1    SFR.uartm1;     /**/
#define         UARTS1  SFR.uarts1;
#define         TXB1    SFR.txb1;       /**/
#define         SIO1    SFR.txb1;       /**/
#define         RXB1    SFR.rxb1;
#define         ASP     SFR.asp;
#define         TC0L    SFR.tc0l;
#define         TC0H    SFR.tc0h;
#define         TCM0L   SFR.tcm0l;
```

```
#define         TCM0H   SFR.tcm0h;
#define         UDC0L   SFR.udc0l;
#define         UDC0H   SFR.udc0h;
#define         DCM0L   SFR.dcm0l;
#define         DCM0H   SFR.dcm0h;
#define         MAR0L   SFR.mar0l;
#define         MAR0H   SFR.mar0h;

#define         DPTC0L  SFR.dptc0l;
#define         DPTC0H  SFR.dptc0h;
#define         DMAM0   SFR.dmam0;
#define         DMAC0   SFR.dmac0;
#define         DMAS    SFR.dmas;
#define         TC1L    SFR.tc1l;
#define         TC1H    SFR.tc1h;
#define         TCM1L   SFR.tcm1l;
#define         TCM1H   SFR.tcm1h;
#define         UDC1L   SFR.udc1l;
#define         UDC1H   SFR.udc1h;
#define         DCM1L   SFR.dcm1l;
#define         DCM1H   SFR.dcm1h;
#define         MAR1L   SFR.mar1l;
#define         MAR1H   SFR.mar1h;
#define         DPTC1L  SFR.dptc1l;
#define         DPTC1H  SFR.dptc1h;
#define         DMAM1   SFR.dmam1;
#define         DMAC1   SFR.dmac1;
#define         STC     SFR.stc;
#define         STMC    SFR.stmc;

#define         PWC0    SFR.pwc0;
#define         PWC1    SFR.pwc1;
#define         MBC     SFR.mbc;
#define         RFM     SFR.rfm;
#define         STBC    SFR.stbc;
#define         PRC     SFR.prc;
```

保守／廃止

付録E　命令索引

E.1　機能別索引

保守／廃止

保守／廃止

E.2　アルファベット順索引

【S】

【T】

【X】

〔メ　モ〕

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
名古屋　(052)222−2375
大　阪　(06)6945−3178, 3200, 3208, 3212
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
広　島　(082)242−5504
高　崎　(027)326−1303
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【インターネット電子デバイス・ニュース】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL(アドレス)　http://www.ic.nec.co.jp/

保守／廃止

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］V55PI　ユーザーズ・マニュアル　命令編
（U10231JJ4V1UM00（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