お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

ユーザーズ・マニュアル

78K/0Sシリーズ

8ビット・シングルチップ・マイクロコンピュータ

命令編

78K/0Sシリーズ共通

資料番号　U11047JJ4V0UM00（第4版）
発行年月　April 2006 N　CP(K)

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

① 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力がノイズなどに起因して，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

② 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，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，未使用の入出力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 静電気対策

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，当社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。

④ 初期化以前の状態

電源投入時，MOSデバイスの初期状態は不定です。

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。

⑤ 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

⑥ 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

EEPROMは，NECエレクトロニクス株式会社の登録商標です。

本製品のうち，外国為替及び外国貿易法の規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

- 本資料に記載されている内容は2006年4月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品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，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，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，ソフトウエアおよびこれらに関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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 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いただく「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ください。

  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，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

  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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対 象 者　このマニュアルは，78K/0Sシリーズ製品の機能を理解し，その応用システムを設計するユーザのエンジニアを対象としています。

目　　的　このマニュアルは，78K/0Sシリーズ製品の持つ各種命令機能を理解していただくことを目的とします。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

- ●CPU機能
- ●命令セット
- ●命令の説明

読 み 方　このマニュアルを読むにあたっては，電気，論理回路およびマイクロコンピュータの一通りの知識を必要とします。

○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
→**付録**A, B　**命令索引**を利用してください。

○ニモニックは分からないが，大体の機能が分かっている命令を確認するとき
→**第4章　命令セット**でそのニモニックを調べ，そのあと**第5章　命令の説明**で機能を調べてください。

○一通り78K/0Sシリーズ製品の各種命令を理解しようとするとき
→目次に従って読んでください。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。この“★”をPDF上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

○78K/0Sシリーズ製品のハードウエア機能について知りたいとき
→別冊のユーザーズ・マニュアルを参照してください。

凡　　例

データ表記の重み	：左側が上位桁，右側が下位桁
注	：本文中につけた注の説明
注意	：特に気をつけていただきたい内容
備考	：本文の補足説明
数の表記	：2進数…××××Bまたは××××
	10進数…××××
	16進数…××××H

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。

○78K/0S**シリーズ共通資料**

資料名	和文資料番号	英文資料番号
ユーザーズ・マニュアル　命令編	このマニュアル	U11047E

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには必ず最新の資料をご使用ください。

目　次

第1章　メモリ空間

1. 1　メモリ空間

78K/0Sシリーズの製品は,内蔵するメモリの容量などによってプログラム・メモリのマッピングが異なります。メモリ・マップのアドレス領域の詳細については，**各製品のユーザーズ・マニュアル**を参照してください。

1. 2　内部プログラム・メモリ（内部ROM）空間

78K/0Sシリーズの製品は，次に示すアドレス空間にそれぞれROMを内蔵しており，プログラムやテーブル・データなどを格納します。通常，プログラム・カウンタ（PC）でアドレスされます。内部ROM空間については，**各製品のユーザーズ・マニュアル**を参照してください。

表1－1　78K/0Sシリーズ製品の内部ROM空間の例

容量 / アドレス空間 / サブシリーズ名	2Kバイト	4Kバイト	8Kバイト	12Kバイト	16Kバイト	24Kバイト	32Kバイト
	0000H-07FFH	0000H-0FFFH	0000H-1FFFH	0000H-2FFFH	0000H-3FFFH	0000H-5FFFH	0000H-7FFFH
μPD789177 サブシリーズ					μPD789176	μPD789177 μPD78F9177 μPD78F9177A	
78K0S/KA1+ サブシリーズ	μPD78F9221	μPD78F9222					

1. 3　ベクタ・テーブル領域

ベクタ・テーブル領域には，$\overline{\text{RESET}}$入力，各割り込み要求発生により分岐するときのプログラム・スタート・アドレスを格納しておきます。16ビット・アドレスのうち下位8ビットが偶数アドレスに，上位8ビットが奇数アドレスに格納されます。ベクタ・テーブル領域については，**各製品のユーザーズ・マニュアル**を参照してください。

表1－2　ベクタ・テーブル（0000H-0023H）（μPD789177サブシリーズの例）

ベクタ・テーブル・アドレス	割り込み要求	ベクタ・テーブル・アドレス	割り込み要求
0000H	$\overline{\text{RESET}}$入力	0012H	INTWT
0004H	INTWDT	0014H	INTWTI
0006H	INTP0	0016H	INTTM80
0008H	INTP1	0018H	INTTM81
000AH	INTP2	001AH	INTTM82
000CH	INTP3	001CH	INTTM90
000EH	INTSR20/INTCSI20	0022H	INTAD0
0010H	INTST20		

表1－3　ベクタ・テーブル（0000H-0020H）（78K0S/KA1+の例）

ベクタ・テーブル・アドレス	割り込み要求	ベクタ・テーブル・アドレス	割り込み要求
0000H	$\overline{\text{RESET}}$入力	0012H	INTAD
0006H	INTLVI	0016H	INTP2
0008H	INTP0	0018H	INTP3
000AH	INTP1	001AH	INTTM80
000CH	INTTM1	001CH	INTSRE6
000EH	INTTM000	001EH	INTSR6
0010H	INTTM010	0020H	INTST6

1.4　CALLT命令テーブル領域

0040H-007FHの64バイトの領域には，1バイト・コール命令（CALLT）のサブルーチン・エントリ・アドレスを格納することができます。

1.5　内部データ・メモリ空間

78K/0Sシリーズの製品は，次に示すデータ・メモリを内蔵しています。内蔵しているデータ・メモリについては，各製品のユーザーズ・マニュアルを参照してください。

（1）内部高速RAM

78K/0Sシリーズの製品は，それぞれ内部高速RAMを内蔵しています。

内部高速RAMはスタックとしても使用します。

（2）LCD表示用RAM

78K/0Sシリーズには，LCD表示用RAMが割り付けられている製品があります。

LCD表示用RAMは，通常のRAMとしても使用できます。

（3）EEPROM®

78K/0Sシリーズには，EEPROM（Electrically Erasable PROM）が割り付けられている製品があります。

EEPROMは，通常のRAMと異なり，電源を切ってもその内容を保持できます。また，EPROMとは異なり，紫外線を用いずに電気的に内容を消去することができます。

表1－4　78K/0Sシリーズ製品の内部データ・メモリ空間の例

サブシリーズ名	製品名	高速RAM	LCD表示用RAM	EEPROM
μPD789177サブシリーズ	μPD789176	FD00H-FEFFH（512バイト）	－	－
	μPD789177			
	μPD78F9177			
	μPD78F9177A			
78K0S/KA1+	μPD78F9221	FE80H-FEFFH（128バイト）	－	－
	μPD78F9222	FE00H-FEFFH（256バイト）		

1. 6　特殊機能レジスタ（SFR：Special Function Register）領域

FF00H-FFFFHの256バイトの領域には，オンチップ周辺ハードウエアの特殊機能レジスタ（SFR）が割り付けられています。特殊機能レジスタについては，各製品のユーザーズ・マニュアルを参照してください。

第2章　レジスタ

2. 1　制御レジスタ

プログラム・シーケンス，ステータス，スタック・メモリの制御など専用の機能を持ったレジスタです。制御レジスタには，プログラム・カウンタ，プログラム・ステータス・ワード，スタック・ポインタがあります。

2. 1. 1　プログラム・カウンタ（PC）

プログラム・カウンタは，次に実行するプログラムのアドレス情報を保持する16ビット・レジスタです。

通常動作時には，フェッチする命令のバイト数に応じて，自動的にインクリメントされます。分岐命令実行時には，イミーディエト・データやレジスタの内容がセットされます。

$\overline{\text{RESET}}$入力により，0000Hと0001H番地のリセット・ベクタ・テーブルの値がプログラム・カウンタにセットされます。

図2－1　プログラム・カウンタの構成

	15															0
PC	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

2. 1. 2　プログラム・ステータス・ワード（PSW）

プログラム・ステータス・ワードは，命令の実行によってセット，リセットされる各種フラグで構成される8ビット・レジスタです。

プログラム・ステータス・ワードの内容は，割り込み要求発生時およびPUSH PSW命令の実行時に自動的にスタックされ，RETI命令およびPOP PSW命令の実行時に自動的に復帰されます。

$\overline{\text{RESET}}$入力により，02Hになります。

図2－2　プログラム・ステータス・ワードの構成

7							0
IE	Z	0	AC	0	0	1	CY

(1) 割り込み許可フラグ（IE）

CPUの割り込み要求受け付け動作を制御するフラグです。

IE = 0のときは割り込み禁止（DI）状態となり，ノンマスカブル割り込み以外の割り込みはすべて禁止されます。

IE = 1のときは割り込み許可（EI）状態となります。このときの割り込み要求の受け付けは，各割り込み要因に対する割り込みマスク・フラグにより制御されます。

このフラグは，DI命令実行または割り込みの受け付けでリセット（0）され，EI命令実行によりセット（1）されます。

(2) ゼロ・フラグ（Z）

演算結果がゼロのときセット（1）され，それ以外のときにリセット（0）されるフラグです。

(3) 補助キャリー・フラグ（AC）

演算結果が，ビット3からキャリーがあったとき，またはビット3へのボローがあったときセット（1）され，それ以外のときリセット（0）されるフラグです。

(4) キャリー・フラグ（CY）

加減算命令実行時のオーバフロー，アンダフローを記憶するフラグです。また，ローテート命令実行時はシフト・アウトされた値を記憶し，ビット演算命令実行時には，ビット・アキュームレータとして機能します。

2. 1. 3　スタック・ポインタ（SP）

メモリのスタック領域の先頭アドレスを保持する16ビットのレジスタです。スタック領域としては内部高速RAM領域のみ設定可能です。

図2－3　スタック・ポインタの構成

	15															0
SP	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

スタック・メモリへの書き込み（退避）動作に先立ってデクリメントされ，スタック・メモリからの読み取り（復帰）動作のあとインクリメントされます。

各スタック動作によって退避／復帰されるデータは図2－4，2－5のようになります。

注意　SPの内容は$\overline{\text{RESET}}$入力により，不定になりますので，必ず命令実行前にイニシャライズしてください。

図2－4　スタック・メモリへ退避されるデータ

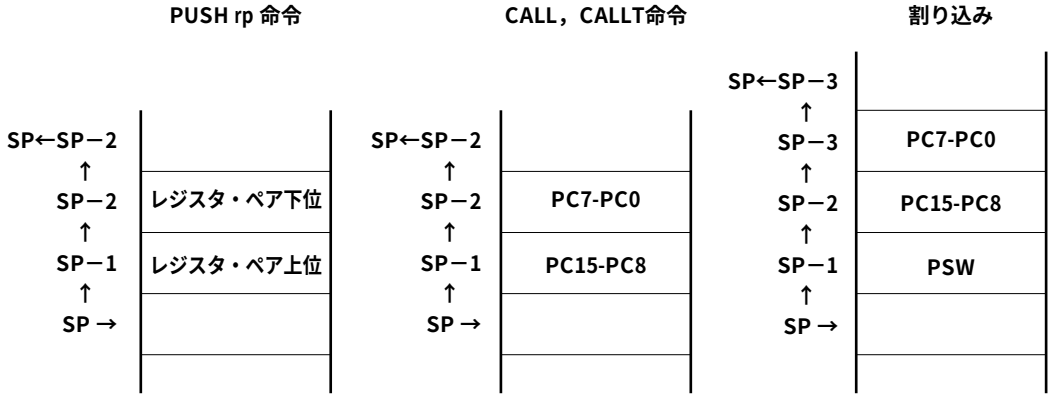
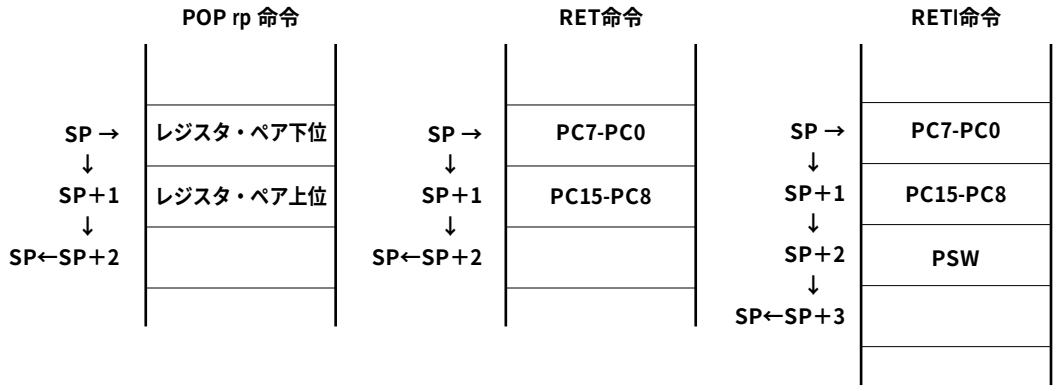


図2－5　スタック・メモリから復帰されるデータ



2. 2　汎用レジスタ

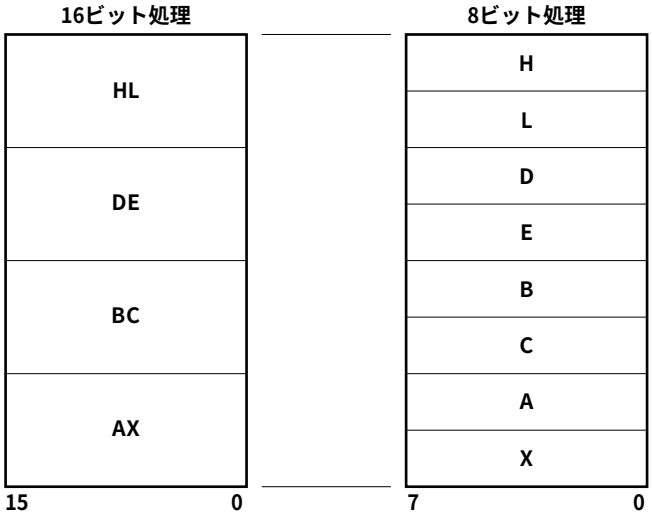
汎用レジスタは，8ビット・レジスタ8個（X, A, C, B, E, D, L, H）で構成されています。

各レジスタは，それぞれ8ビット・レジスタとして使用できるほか，2個の8ビット・レジスタをペアとして16ビット・レジスタとしても使用できます（AX, BC, DE, HL）。

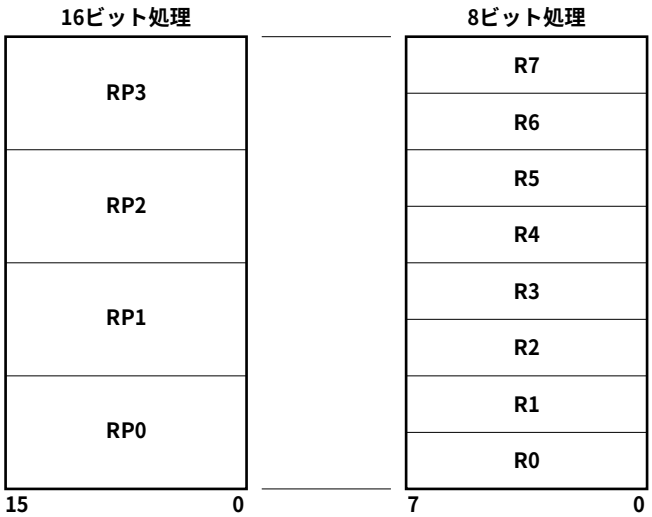
また，機能名称（X, A, C, B, E, D, L, H, AX, BC, DE, HL）のほか，絶対名称（R0-R7, RP0-RP3）ででも記述できます。

図2－6　汎用レジスタの構成

（a）機能名称



（b）絶対名称



2. 3　特殊機能レジスタ（SFR）

特殊機能レジスタは，汎用レジスタとは異なり，それぞれ特別な機能を持つレジスタです。

FF00H-FFFFHの256バイトの空間に割り付けられています。

特殊機能レジスタは，演算命令，転送命令，ビット操作命令などにより，汎用レジスタと同じように操作できます。操作可能なビット単位（1, 8, 16）は，各特殊機能レジスタで異なります。

各操作ビット単位ごとの指定方法を次に示します。

- 1ビット操作

  1ビット操作命令のオペランド（sfr.bit）にアセンブラで予約されている略号を記述します。アドレスでも指定できます。
- 8ビット操作

  8ビット操作命令のオペランド（sfr）にアセンブラで予約されている略号を記述します。アドレスでも指定できます。
- 16ビット操作

  16ビット操作命令のオペランドにアセンブラで予約されている略号を記述します。アドレスを指定するときは偶数アドレスを記述してください。

特殊機能レジスタについては，**各製品のユーザーズ・マニュアル**を参照してください。

第3章　アドレシング

3. 1　命令アドレスのアドレシング

命令アドレスは，プログラム・カウンタ（PC）の内容によって決定されます。PCの内容は，通常，命令を1つ実行するごとにフェッチする命令のバイト数に応じて自動的にインクリメント（1バイトに対して＋1）されます。しかし，分岐を伴う命令を実行する際には，次に示すようなアドレシングにより分岐先アドレス情報がPCにセットされて分岐します（各命令についての詳細は**第5章　命令の説明**を参照してください）。

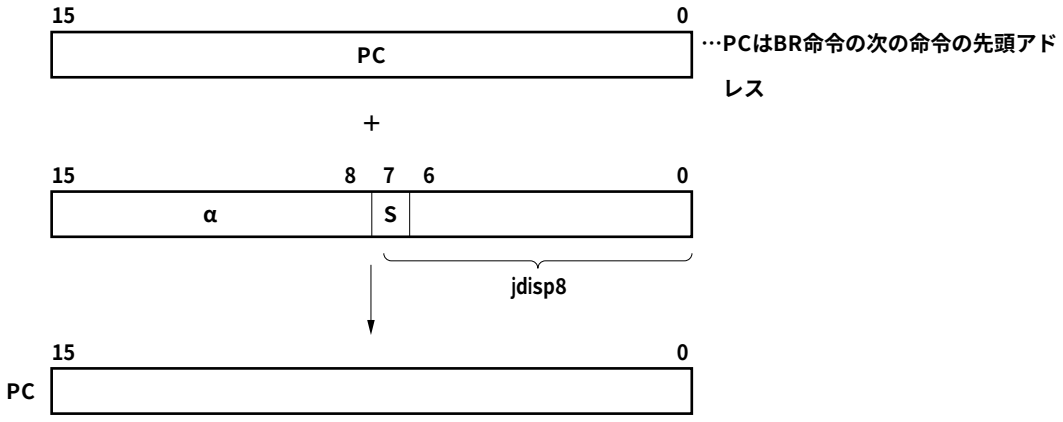
3. 1. 1　レラティブ・アドレシング

【機　　能】

次に続く命令の先頭アドレスに命令コードの8ビット・イミーディエト・データ（ディスプレースメント値：jdisp8）を加算した値が，プログラム・カウンタ（PC）に転送されて分岐します。ディスプレースメント値は，符号付きの2の補数データ（－128～＋127）として扱われ，ビット7が符号ビットとなります。つまり，レラティブ・アドレシングでは，次に続く命令の先頭アドレスから相対的に－128～＋127の範囲に分岐するということです。

BR　$addr16命令および条件付き分岐命令を実行する際に行われます。

【図　　解】



S=0のとき，αは全ビット0

S=1のとき，αは全ビット1

3. 1. 2　イミーディエト・アドレシング

【機　　能】

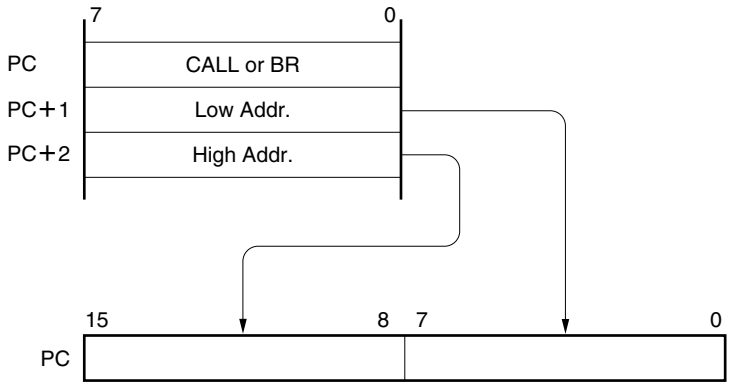
命令語中のイミーディエト・データがプログラム・カウンタ（PC）に転送され，分岐します。

CALL !addr16, BR !addr16**命令を実行する際に行われます。**

CALL !addr16, BR !addr16**命令は，全メモリ空間に分岐できます。**

【図　　解】

CALL !addr16, BR !addr16**命令の場合**



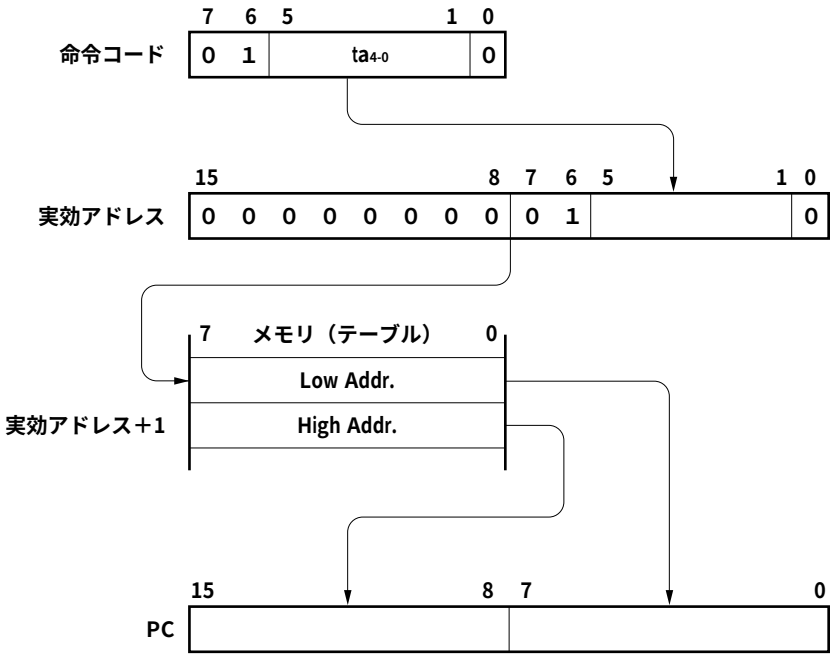
3. 1. 3　テーブル・インダイレクト・アドレシング

【機　　能】

命令コードのビット1からビット5のイミーディエト・データによりアドレスされる特定ロケーションのテーブルの内容（分岐先アドレス）がプログラム・カウンタ（PC）に転送され，分岐します。

CALLT［addr5］**命令を実行する際にテーブル・インダイレクト・アドレシングが行われます。この命令では** 40H-7FH**のメモリ・テーブルに格納されたアドレスを参照し，全メモリ空間に分岐できます。**

【図　　解】



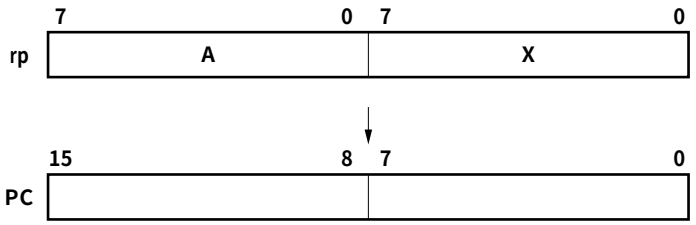
3. 1. 4　レジスタ・アドレシング

【機　　能】

命令語によって指定されるレジスタ・ペア（AX**）の内容がプログラム・カウンタ（**PC**）に転送され，分岐します。**

BR AX**命令を実行する際に行われます。**

【図　　解】



3. 2　オペランド・アドレスのアドレシング

命令を実行する際に操作対象となるレジスタやメモリなどを指定する方法（アドレシング）として次に示すいくつかの方法があります。

3. 2. 1　ダイレクト・アドレシング

【機　　能】

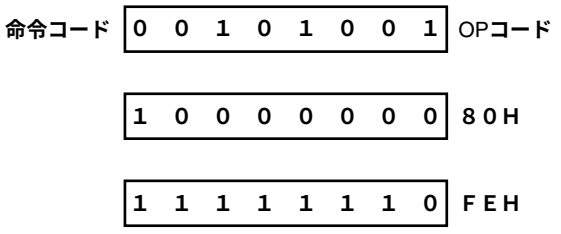
命令語中のイミーディエト・データが示すメモリを直接アドレスするアドレシングです。

【オペランド形式】

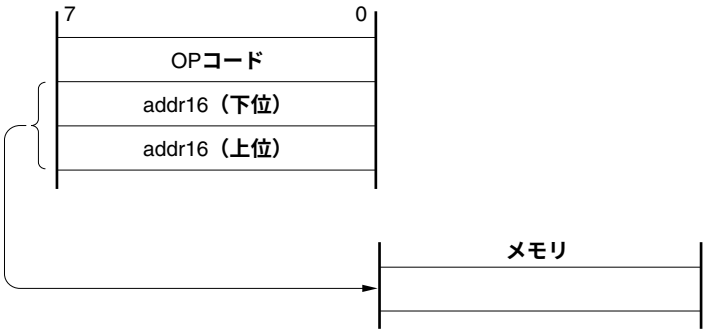
表現形式	記述方法
addr16	レーベルまたは16ビット・イミーディエト・データ

【記　述　例】

MOV A, !0FE80H；!addr16をFE80Hとする場合



【図　　解】



3. 2. 2　ショート・ダイレクト・アドレシング

【機　　能】

命令語中の8ビット・データで，固定空間の操作対象メモリを直接アドレスするアドレシングです。

このアドレシングが適用される固定空間とは，FE20H-FF1FHの256バイト空間です。FE20H-FEFFHには内部高速RAMが，FF00H-FF1FHには特殊機能レジスタ（SFR）がマッピングされています。

ショート・ダイレクト・アドレシングが適用されるSFR領域（FF00H-FF1FH）は，全SFR領域の一部です。この領域には，プログラム上でひんぱんにアクセスされるポートや，タイマ／イベント・カウンタのコンペア・レジスタがマッピングされており，短いバイト数，短いクロック数でこれらのSFRを操作することができます。

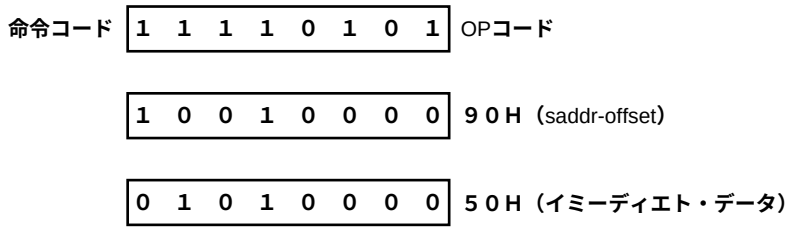
実効アドレスのビット8には，8ビット・イミーディエト・データが20H-FFHの場合は0になり，00H-1FHの場合は1になります。**次頁の【図解】**を参照してください。

【オペランド形式】

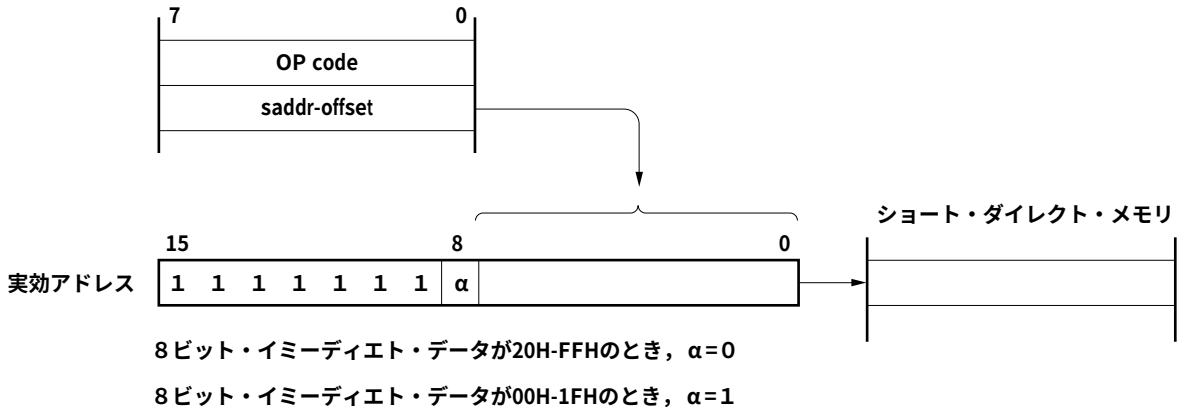
表現形式	記述方法
saddr	レーベルまたはFE20H-FF1FHのイミーディエト・データ
saddrp	レーベルまたはFE20H-FF1FHのイミーディエト・データ（偶数アドレスのみ）

【記　述　例】

```
EQU DATA1 0FE90H    ；DATA1はsaddr領域のFE90Hを示し，
MOV DATA1, #50H     ；イミーディエト・データを50Hとする場合
```



【図　　解】



8ビット・イミーディエト・データが20H-FFHのとき，α=0

8ビット・イミーディエト・データが00H-1FHのとき，α=1

3. 2. 3　特殊機能レジスタ（SFR）アドレシング

【機　　能】

命令語中の8ビット・イミーディエト・データでメモリ・マッピングされている特殊機能レジスタ（SFR）をアドレスするアドレシングです。

このアドレシングが適用されるのはFF00H-FFCFH，FFE0H-FFFFHの240バイト空間です。ただし，FF00H-FF1FHにマッピングされているSFRは，ショート・ダイレクト・アドレシングでもアクセスできます。

【オペランド形式】

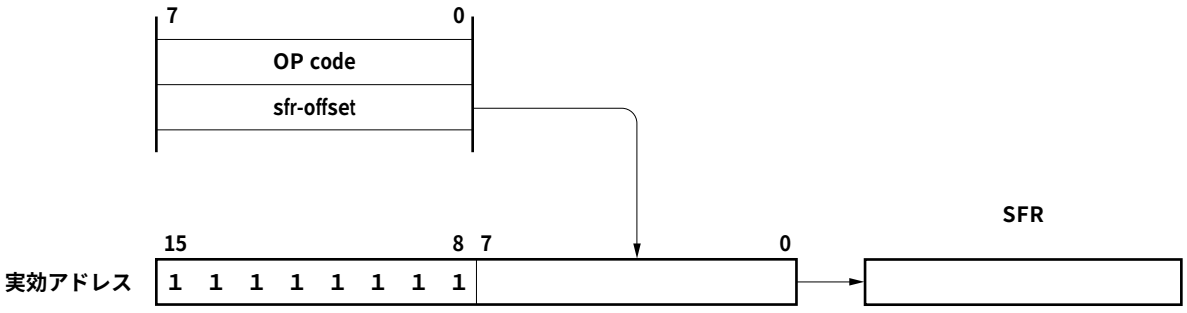
表現形式	記述方法
sfr	特殊機能レジスタ名

【記　述　例】

MOV PM0, A；sfrにPM0を選択する場合

命令コード	1	1	1	0	0	1	1	1
	0	0	1	0	0	0	0	0

【図　　解】



3. 2. 4　レジスタ・アドレシング

【機　　能】

オペランドとして汎用レジスタをアクセスするアドレシングです。アクセスされる汎用レジスタは，命令コード中のレジスタ指定コードや機能名称で指定されます。

レジスタ・アドレシングは，次に示すオペランド形式を持つ命令を実行する際に行われ，8ビット・レジスタを指定する場合は命令コード中の3ビット（レジスタ指定コード）により8本中の1本を指定します。

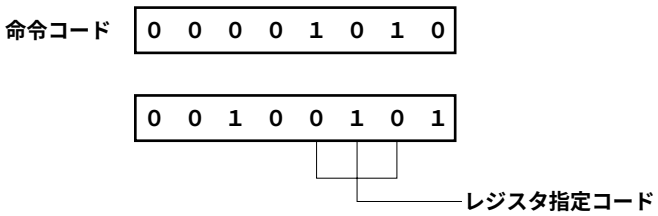
【オペランド形式】

表現形式	記述方法
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

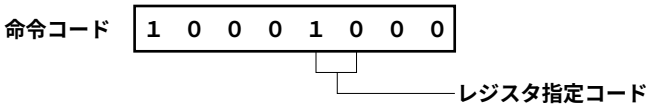
r, rp**は，機能名称（**X, A, C, B, E, D, L, H, AX, BC, DE, HL**）のほかに絶対名称（**R0-R7, RP0-RP3**）で記述できます。**

【記　述　例】

MOV A, C**；r**にC**レジスタを選択する場合**



INCW DE**；rp**にDE**レジスタ・ペアを選択する場合**



3. 2. 5　レジスタ・インダイレクト・アドレシング

【機　　能】

オペランドとして指定されるレジスタ・ペアの内容でメモリをアドレスするアドレシングです。アクセスされるレジスタ・ペアは，命令コード中のレジスタ・ペア指定コードにより指定されます。すべてのメモリ空間に対してアドレシングできます。

【オペランド形式】

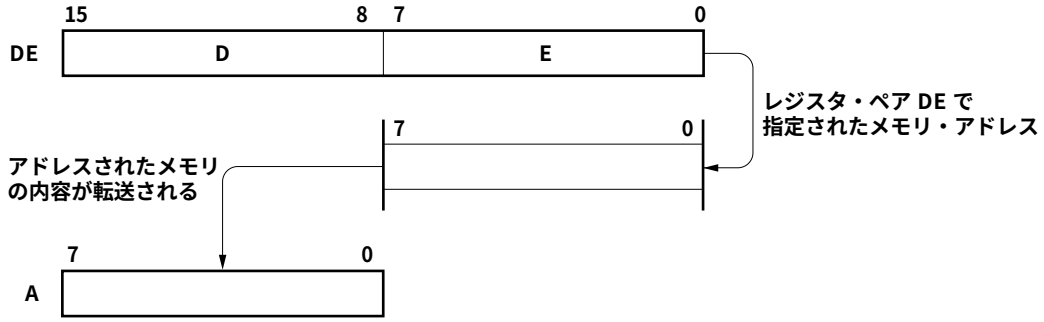
表現形式	記述方法
−	[DE]，[HL]

【記　述　例】

MOV A, ［DE］**；レジスタ・ペア**［DE］**を選択する場合**

命令コード	0	0	1	0	1	0	1	1

【図　　解】



3. 2. 6　ベースト・アドレシング

【機　　能】

HLレジスタ・ペアをベース・レジスタとし，この内容に8ビットのイミーディエト・データを加算した結果でメモリをアドレスするアドレシングです。加算は，オフセット・データを正の数として16ビットに拡張して行います。16ビット目からの桁上がりは無視します。すべてのメモリ空間に対してアドレシングできます。

【オペランド形式】

表現形式	記述方法
－	[HL＋byte]

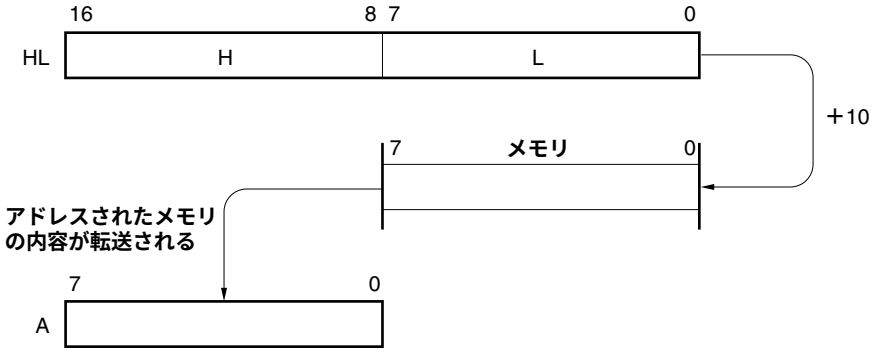
【記　述　例】

MOV A, ［HL＋10H］；byteを10Hとする場合

命令コード　0 0 1 0 1 1 0 1

0 0 0 1 0 0 0 0

【図　　解】



3. 2. 7　スタック・アドレシング

【機　　能】

スタック・ポインタ（SP）の内容により，スタック領域を間接的にアドレスするアドレシングです。

PUSH, POP**，サブルーチン・コール，リターン命令の実行時および割り込み要求発生によるレジスタの退避／復帰時に自動的に用いられます。**

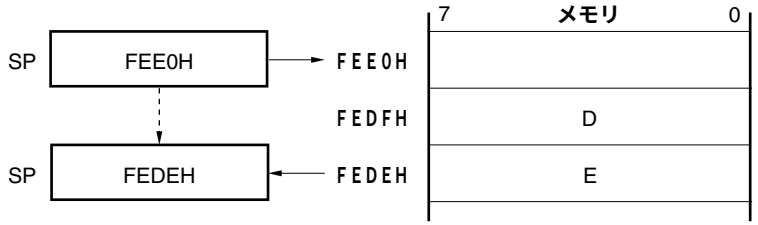
スタック・アドレシングは，内部高速RAM**領域のみアクセスすることができます。**

【記　述　例】

PUSH DE**の場合**

命令コード	1	0	1	0	1	0	1	0

【図　　解】



第4章　命令セット

この章では，78K/0S**シリーズの命令セットを一覧表にして示します。**

78K/0S**シリーズ製品の命令は，すべて共通です。**

4.1　オペレーション

4.1.1　オペランドの表現形式と記述方法

各命令のオペランド欄には，その命令のオペランド表現形式に対する記述方法に従ってオペランドを記述しています（詳細は，アセンブラ仕様による）。記述方法の中で複数個あるものは，それらの要素の1つを選択します。大文字で書かれた英字および#，!，$，［　］の記号はキー・ワードであり，そのまま記述します。記号の説明は，次のとおりです。

- #：イミーディエト・データ指定
- $　：相対アドレス指定
- !：絶対アドレス指定
- ［　］：間接アドレス指定

イミーディエト・データのときは，適当な数値またはレーベルを記述します。レーベルで記述する際も#，!，$，［　］記号は必ず記述してください。

また，オペランドのレジスタの記述形式r, rpには，機能名称（X, A, Cなど），絶対名称（下表の中のカッコ内の名称，R0, R1, R2など）のいずれの形式でも記述可能です。

表4－1　オペランドの表現形式と記述方法

表現形式	記述方法
r	X（R0），A（R1），C（R2），B（R3），E（R4），D（R5），L（R6），H（R7）
rp	AX（RP0），BC（RP1），DE（RP2），HL（RP3）
sfr	特殊機能レジスタ略号
saddr	FE20H-FF1FH　イミーディエト・データまたはレーベル
saddrp	FE20H-FF1FH　イミーディエト・データまたはレーベル（偶数アドレスのみ）
addr16	0000H-FFFFH　イミーディエト・データまたはレーベル （16ビット・データ転送命令時は偶数アドレスのみ）
addr5	0040H-007FH　イミーディエト・データまたはレーベル（偶数アドレスのみ）
word	16ビット・イミーディエト・データまたはレーベル
byte	8ビット・イミーディエト・データまたはレーベル
bit	3ビット・イミーディエト・データまたはレーベル

備考　特殊機能レジスタの略号は**各製品のユーザーズ・マニュアル**を参照してください。

4.1.2　オペレーション欄の説明

A	：Aレジスタ；8ビット・アキュームレータ
X	：Xレジスタ
B	：Bレジスタ
C	：Cレジスタ
D	：Dレジスタ
E	：Eレジスタ
H	：Hレジスタ
L	：Lレジスタ
AX	：AXレジスタ・ペア；16ビット・アキュームレータ
BC	：BCレジスタ・ペア
DE	：DEレジスタ・ペア
HL	：HLレジスタ・ペア
PC	：プログラム・カウンタ
SP	：スタック・ポインタ
PSW	：プログラム・ステータス・ワード
CY	：キャリー・フラグ
AC	：補助キャリー：フラグ
Z	：ゼロ・フラグ
IE	：割り込み要求許可フラグ
( )	：( ) 内のアドレスまたはレジスタの内容で示されるメモリの内容
$\times_H$，$\times_L$	：16ビット・レジスタの上位8ビット，下位8ビット
∧	：論理積（AND）
∨	：論理和（OR）
∀	：排他的論理和（exclusive OR）
$\overline{\quad}$	：反転データ
addr16	：16ビット・イミーディエト・データまたはレーベル
jdisp8	：符号付き8ビット・データ（ディスプレースメント値）

4.1.3　フラグ動作欄の説明

（ブランク）	：変化なし
0	：0にクリアされる
1	：1にセットされる
×	：結果に従ってセット／クリアされる
R	：以前に退避した値がストアされる

4.1.4　クロック欄の説明

命令実行時のクロック数の概要を示します。

命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

オペレーション一覧を次に示します。

4.1.5　オペレーション一覧

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ Z	AC	CY
MOV	r, #byte	3	6	r←byte			
	saddr, #byte	3	6	(saddr) ←byte			
	sfr, #byte	3	6	sfr←byte			
	A, r 注1	2	4	A←r			
	r, A 注1	2	4	r←A			
	A, saddr	2	4	A← (saddr)			
	saddr, A	2	4	(saddr) ←A			
	A, sfr	2	4	A←sfr			
	sfr, A	2	4	sfr←A			
	A, !addr16	3	8	A← (addr16)			
	!addr16, A	3	8	(addr16) ←A			
	PSW, #byte	3	6	PSW←byte	×	×	×
	A, PSW	2	4	A←PSW			
	PSW, A	2	4	PSW←A	×	×	×
	A, [DE]	1	6	A← (DE)			
	[DE], A	1	6	(DE) ←A			
	A, [HL]	1	6	A← (HL)			
	[HL], A	1	6	(HL) ←A			
	A, [HL+byte]	2	6	A← (HL+byte)			
	[HL+byte], A	2	6	(HL+byte) ←A			
XCH	A, X	1	4	A⟷X			
	A, r 注2	2	6	A⟷r			
	A, saddr	2	6	A⟷ (saddr)			
	A, sfr	2	6	A⟷sfr			
	A, [DE]	1	8	A⟷ (DE)			
	A, [HL]	1	8	A⟷ (HL)			
	A, [HL+byte]	2	8	A⟷ (HL+byte)			

注1． r = Aを除く。

2．r = A, Xを除く。

備考　命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ Z	AC	CY
MOVW	rp, #word	3	6	rp←word			
	AX, saddrp	2	6	AX←（saddrp）			
	saddrp, AX	2	8	（saddrp）←AX			
	AX, rp 注	1	4	AX←rp			
	rp, AX 注	1	4	rp←AX			
XCHW	AX, rp 注	1	8	AX⟷rp			
ADD	A, #byte	2	4	A, CY←A＋byte	×	×	×
	saddr, #byte	3	6	（saddr），CY←（saddr）＋byte	×	×	×
	A, r	2	4	A, CY←A＋r	×	×	×
	A, saddr	2	4	A, CY←A＋（saddr）	×	×	×
	A, !addr16	3	8	A, CY←A＋（addr16）	×	×	×
	A，［HL］	1	6	A, CY←A＋（HL）	×	×	×
	A，［HL＋byte］	2	6	A, CY←A＋（HL＋byte）	×	×	×
ADDC	A, #byte	2	4	A, CY←A＋byte＋CY	×	×	×
	saddr, #byte	3	6	（saddr），CY←（saddr）＋byte＋CY	×	×	×
	A, r	2	4	A, CY←A＋r＋CY	×	×	×
	A, saddr	2	4	A, CY←A＋（saddr）＋CY	×	×	×
	A, !addr16	3	8	A, CY←A＋（addr16）＋CY	×	×	×
	A，［HL］	1	6	A, CY←A＋（HL）＋CY	×	×	×
	A，［HL＋byte］	2	6	A, CY←A＋（HL＋byte）＋CY	×	×	×
SUB	A, #byte	2	4	A, CY←A－byte	×	×	×
	saddr, #byte	3	6	（saddr），CY←（saddr）－byte	×	×	×
	A, r	2	4	A, CY←A－r	×	×	×
	A, saddr	2	4	A, CY←A－（saddr）	×	×	×
	A, !addr16	3	8	A, CY←A－（addr16）	×	×	×
	A，［HL］	1	6	A, CY←A－（HL）	×	×	×
	A，［HL＋byte］	2	6	A, CY←A－（HL＋byte）	×	×	×

注　rp = BC, DE, HL**のときのみ。**

備考　**命令の**1**クロックはプロセッサ・クロック・コントロール・レジスタ（**PCC**）で選択した**CPU**クロック（**f_{CPU}**）の**1**クロック分です。**

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ		
					Z	AC	CY
SUBC	A, #byte	2	4	A, CY←A－byte－CY	×	×	×
	saddr, #byte	3	6	(saddr)，CY←(saddr)－byte－CY	×	×	×
	A, r	2	4	A, CY←A－r－CY	×	×	×
	A, saddr	2	4	A, CY←A－(saddr)－CY	×	×	×
	A, !addr16	3	8	A, CY←A－(addr16)－CY	×	×	×
	A, [HL]	1	6	A, CY←A－(HL)－CY	×	×	×
	A, [HL＋byte]	2	6	A, CY←A－(HL＋byte)－CY	×	×	×
AND	A, #byte	2	4	A←A ∧ byte	×		
	saddr, #byte	3	6	(saddr)←(saddr) ∧ byte	×		
	A, r	2	4	A←A ∧ r	×		
	A, saddr	2	4	A←A ∧ (saddr)	×		
	A, !addr16	3	8	A←A ∧ (addr16)	×		
	A, [HL]	1	6	A←A ∧ (HL)	×		
	A, [HL＋byte]	2	6	A←A ∧ (HL＋byte)	×		
OR	A, #byte	2	4	A←A ∨ byte	×		
	saddr, #byte	3	6	(saddr)←(saddr) ∨ byte	×		
	A, r	2	4	A←A ∨ r	×		
	A, saddr	2	4	A←A ∨ (saddr)	×		
	A, !addr16	3	8	A←A ∨ (addr16)	×		
	A, [HL]	1	6	A←A ∨ (HL)	×		
	A, [HL＋byte]	2	6	A←A ∨ (HL＋byte)	×		
XOR	A, #byte	2	4	A←A ⊻ byte	×		
	saddr, #byte	3	6	(saddr)←(saddr) ⊻ byte	×		
	A, r	2	4	A←A ⊻ r	×		
	A, saddr	2	4	A←A ⊻ (saddr)	×		
	A, !addr16	3	8	A←A ⊻ (addr16)	×		
	A, [HL]	1	6	A←A ⊻ (HL)	×		
	A, [HL＋byte]	2	6	A←A ⊻ (HL＋byte)	×		

備考　命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ		
					Z	AC	CY
CMP	A, #byte	2	4	A－byte	×	×	×
	saddr, #byte	3	6	(saddr) －byte	×	×	×
	A, r	2	4	A－r	×	×	×
	A, saddr	2	4	A－ (saddr)	×	×	×
	A, !addr16	3	8	A－ (addr16)	×	×	×
	A, ［HL］	1	6	A－ (HL)	×	×	×
	A, ［HL＋byte］	2	6	A－ (HL＋byte)	×	×	×
ADDW	AX, #word	3	6	AX, CY←AX＋word	×	×	×
SUBW	AX, #word	3	6	AX, CY←AX－word	×	×	×
CMPW	AX, #word	3	6	AX－word	×	×	×
INC	r	2	4	r←r＋1	×	×	
	saddr	2	4	(saddr) ← (saddr) ＋1	×	×	
DEC	r	2	4	r←r－1	×	×	
	saddr	2	4	(saddr) ← (saddr) －1	×	×	
INCW	rp	1	4	rp←rp＋1			
DECW	rp	1	4	rp←rp－1			
ROR	A, 1	1	2	(CY,A_7←A_0, A_{m-1}←A_m) ×1回			×
ROL	A, 1	1	2	(CY,A_0←A_7, A_{m+1}←A_m) ×1回			×
RORC	A, 1	1	2	(CY←A_0, A_7←CY, A_{m-1}←A_m) ×1回			×
ROLC	A, 1	1	2	(CY←A_7, A_0←CY, A_{m+1}←A_m) ×1回			×
SET1	saddr.bit	3	6	(saddr.bit) ←1			
	sfr.bit	3	6	sfr.bit←1			
	A.bit	2	4	A.bit←1			
	PSW.bit	3	6	PSW.bit←1	×	×	×
	［HL］.bit	2	10	(HL) .bit←1			
CLR1	saddr.bit	3	6	(saddr.bit) ←0			
	sfr.bit	3	6	sfr.bit←0			
	A.bit	2	4	A.bit←0			
	PSW.bit	3	6	PSW.bit←0	×	×	×
	［HL］.bit	2	10	(HL) .bit←0			
SET1	CY	1	2	CY←1			1
CLR1	CY	1	2	CY←0			0

備考　命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ Z	AC	CY
NOT1	CY	1	2	CY←$\overline{CY}$			×
CALL	!addr16	3	6	(SP−1) ← (PC+3)$_H$, (SP−2) ← (PC+3)$_L$, PC←addr16, SP←SP−2			
CALLT	[addr5]	1	8	(SP−1) ← (PC+1)$_H$, (SP−2) ← (PC+1)$_L$, PC$_H$← (00000000, addr5+1), PC$_L$← (00000000, addr5), SP←SP−2			
RET		1	6	PC$_H$← (SP+1), PC$_L$← (SP), SP←SP+2			
RETI		1	8	PC$_H$← (SP+1), PC$_L$← (SP), PSW← (SP+2), SP←SP+3	R	R	R
PUSH	PSW	1	2	(SP−1) ←PSW, SP←SP−1			
	rp	1	4	(SP−1) ←rp$_H$, (SP−2) rp$_L$, SP←SP−2			
POP	PSW	1	4	PSW← (SP), SP←SP+1	R	R	R
	rp	1	6	rp$_H$← (SP+1), rp$_L$← (SP), SP←SP+2			
MOVW	SP, AX	2	8	SP←AX			
	AX, SP	2	6	AX←SP			
BR	!addr16	3	6	PC←addr16			
	$addr16	2	6	PC←PC+2+jdisp8			
	AX	1	6	PC$_H$←A, PC$_L$←X			
BC	$addr16	2	6	PC←PC+2+jdisp8 if CY = 1			
BNC	$addr16	2	6	PC←PC+2+jdisp8 if CY = 0			
BZ	$addr16	2	6	PC←PC+2+jdisp8 if Z = 1			
BNZ	$addr16	2	6	PC←PC+2+jdisp8 if Z = 0			
BT	saddr.bit, $addr16	4	10	PC←PC+4+jdisp8 if (saddr.bit) = 1			
	sfr.bit, $addr16	4	10	PC←PC+4+jdisp8 if sfr.bit = 1			
	A.bit, $addr16	3	8	PC←PC+3+jdisp8 if A.bit = 1			
	PSW.bit, $addr16	4	10	PC←PC+4+jdisp8 if PSW.bit = 1			

備考　命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

ニモニック	オペランド	バイト	クロック	オペレーション	フラグ Z AC CY
BF	saddr.bit, $addr16	4	10	PC←PC＋4＋jdisp8 if（saddr.bit）= 0	
	sfr.bit, $addr16	4	10	PC←PC＋4＋jdisp8 if sfr.bit = 0	
	A.bit, $addr16	3	8	PC←PC＋3＋jdisp8 if A.bit = 0	
	PSW.bit, $addr16	4	10	PC←PC＋4＋jdisp8 if PSW.bit = 0	
DBNZ	B, $addr16	2	6	B←B－1, then PC←PC＋2＋jdisp8 if B≠0	
	C, $addr16	2	6	C←C－1, then PC←PC＋2＋jdisp8 if C≠0	
	saddr, $addr16	3	8	（saddr）←（saddr）－1, then PC←PC＋3＋jdisp8 if（saddr）≠0	
NOP		1	2	No Operation	
EI		3	6	IE←1（Enable Interrupt）	
DI		3	6	IE←0（Disable Interrupt）	
HALT		1	2	Set HALT Mode	
STOP		1	2	Set STOP Mode	

備考　命令の1クロックはプロセッサ・クロック・コントロール・レジスタ（PCC）で選択したCPUクロック（f_{CPU}）の1クロック分です。

4.1.6　アドレシング別命令一覧

(1) 8ビット命令

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, INC, DEC, ROR, ROL, RORC, ROLC, PUSH, POP, DBNZ

第2オペランド / 第1オペランド	#byte	A	r	sfr	saddr	!addr16	PSW	[DE]	[HL]	[HL+byte]	$addr16	1	なし
A	ADD ADDC SUB SUBC AND OR XOR CMP		MOV[注] XCH[注] ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP		ROR ROL RORC ROLC	
r	MOV	MOV[注]											INC DEC
B, C											DBNZ		
sfr	MOV	MOV											
saddr	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV									DBNZ		INC DEC
!addr16		MOV											
PSW	MOV	MOV											PUSH POP
[DE]		MOV											
[HL]		MOV											
[HL+byte]		MOV											

注　r = Aは除く。

(2) 16ビット命令

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

第1オペランド ＼ 第2オペランド	#word	AX	rp[注]	saddrp	SP	なし
AX	ADDW SUBW CMPW		MOVW XCHW	MOVW	MOVW	
rp	MOVW	MOVW[注]				INCW DECW PUSH POP
saddrp		MOVW				
SP		MOVW				

注　rp = BC, DE, HLのときのみ。

(3) ビット操作命令

SET1, CLR1, NOT1, BT, BF

第1オペランド ＼ 第2オペランド	$addr16	なし
A.bit	BT BF	SET1 CLR1
sfr.bit	BT BF	SET1 CLR1
saddr.bit	BT BF	SET1 CLR1
PSW.bit	BT BF	SET1 CLR1
[HL].bit		SET1 CLR1
CY		SET1 CLR1 NOT1

(4) コール命令／分岐命令

CALL, CALLT, BR, BC, BNC, BZ, BNZ, BT, BF, DBNZ

第1オペランド ＼ 第2オペランド	AX	!addr16	[addr5]	$addr16
基本命令	BR	CALL BR	CALLT	BR BC BNC BZ BNZ
複合命令				DBNZ

(5) その他の命令

RET, RETI, NOP, EI, DI, HALT, STOP

4. 2　命令コード

4. 2. 1　命令コード表の説明

r

R_2	R_1	R_0	reg	
0	0	0	R0	X
0	0	1	R1	A
0	1	0	R2	C
0	1	1	R3	B
1	0	0	R4	E
1	0	1	R5	D
1	1	0	R6	L
1	1	1	R7	H

rp

P_1	P_0	reg-pair	
0	0	RP0	AX
0	1	RP1	BC
1	0	RP2	DE
1	1	RP3	HL

Bn	：bitに対応するイミーディエト・データ
Data	：byteに対応する8ビット・イミーディエト・データ
Low/High byte	：wordに対応する16ビット・イミーディエト・データ
Saddr-offset	：saddrに対応する16ビット・アドレスの下位8ビット・オフセット・データ
Sfr-offset	：sfrの16ビット・アドレスの下位8ビット・オフセット・データ
Low/High addr	：addr16に対応する16ビット・イミーディエト・データ
jdisp	：次の命令の先頭アドレスと分岐アドレスとの相対アドレス距離の符号付き2の補数データ（8ビット）
ta_{4-0}	：addr5に対応するイミーディエト・データの5ビット

4.2.2 命令コード一覧

ニモニック	オペランド	命令コード			
		B1	B2	B3	B4
MOV	r, #byte	0 0 0 0　1 0 1 0	1 1 1 1　R_2 R_1 R_0 1	Data	
	saddr, #byte	1 1 1 1　0 1 0 1	Saddr-offset	Data	
	sfr, #byte	1 1 1 1　0 1 1 1	Sfr-offset	Data	
	A, r 注1	0 0 0 0　1 0 1 0	0 0 1 0　R_2 R_1 R_0 1		
	r, A 注1	0 0 0 0　1 0 1 0	1 1 1 0　R_2 R_1 R_0 1		
	A, saddr	0 0 1 0　0 1 0 1	Saddr-offset		
	saddr, A	1 1 1 0　0 1 0 1	Saddr-offset		
	A, sfr	0 0 1 0　0 1 1 1	Sfr-offset		
	sfr, A	1 1 1 0　0 1 1 1	Sfr-offset		
	A, !addr16	0 0 1 0　1 0 0 1	Low addr	High addr	
	!addr16, A	1 1 1 0　1 0 0 1	Low addr	High addr	
	PSW, #byte	1 1 1 1　0 1 0 1	0 0 0 1　1 1 1 0	Data	
	A, PSW	0 0 1 0　0 1 0 1	0 0 0 1　1 1 1 0		
	PSW, A	1 1 1 0　0 1 0 1	0 0 0 1　1 1 1 0		
	A, ［DE］	0 0 1 0　1 0 1 1			
	［DE］, A	1 1 1 0　1 0 1 1			
	A, ［HL］	0 0 1 0　1 1 1 1			
	［HL］, A	1 1 1 0　1 1 1 1			
	A, ［HL＋byte］	0 0 1 0　1 1 0 1	Data		
	［HL＋byte］, A	1 1 1 0　1 1 0 1	Data		
XCH	A, X	1 1 0 0　0 0 0 0			
	A, r 注2	0 0 0 0　1 0 1 0	0 0 0 0　R_2 R_1 R_0 1		
	A, saddr	0 0 0 0　0 1 0 1	Saddr-offset		
	A, sfr	0 0 0 0　0 1 1 1	Sfr-offset		
	A, ［DE］	0 0 0 0　1 0 1 1			
	A, ［HL］	0 0 0 0　1 1 1 1			
	A, ［HL＋byte］	0 0 0 0　1 1 0 1	Data		
MOVW	rp, #word	1 1 1 1　P_1 P_0 0 0	Low byte	High byte	
	AX, saddrp	1 1 0 1　0 1 1 0	Saddr-offset		
	saddrp, AX	1 1 1 0　0 1 1 0	Saddr-offset		
	AX, rp 注3	1 1 0 1　P_1 P_0 0 0			
	rp, AX 注3	1 1 1 0　P_1 P_0 0 0			
XCHW	AX, rp 注3	1 1 0 0　P_1 P_0 0 0			

注1．r = A**を除く。**

2．r = A, X**を除く。**

3．rp = BC, DE, HL**のときのみ。**

ニモニック	オペランド	命令コード			
		B1	B2	B3	B4
ADD	A, #byte	1 0 0 0　0 0 1 1	Data		
	saddr, #byte	1 0 0 0　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	1 0 0 0　R_2 R_1 R_0 1		
	A, saddr	1 0 0 0　0 1 0 1	Saddr-offset		
	A, !addr16	1 0 0 0　1 0 0 1	Low addr	High addr	
	A,　[HL]	1 0 0 0　1 1 1 1			
	A,　[HL＋byte]	1 0 0 0　1 1 0 1	Data		
ADDC	A, #byte	1 0 1 0　0 0 1 1	Data		
	saddr, #byte	1 0 1 0　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	1 0 1 0　R_2 R_1 R_0 1		
	A, saddr	1 0 1 0　0 1 0 1	Saddr-offset		
	A, !addr16	1 0 1 0　1 0 0 1	Low addr	High addr	
	A,　[HL]	1 0 1 0　1 1 1 1			
	A,　[HL＋byte]	1 0 1 0　1 1 0 1	Data		
SUB	A, #byte	1 0 0 1　0 0 1 1	Data		
	saddr, #byte	1 0 0 1　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	1 0 0 1　R_2 R_1 R_0 1		
	A, saddr	1 0 0 1　0 1 0 1	Saddr-offset		
	A, !addr16	1 0 0 1　1 0 0 1	Low addr	High addr	
	A,　[HL]	1 0 0 1　1 1 1 1			
	A,　[HL＋byte]	1 0 0 1　1 1 0 1	Data		
SUBC	A, #byte	1 0 1 1　0 0 1 1	Data		
	saddr, #byte	1 0 1 1　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	1 0 1 1　R_2 R_1 R_0 1		
	A, saddr	1 0 1 1　0 1 0 1	Saddr-offset		
	A, !addr16	1 0 1 1　1 0 0 1	Low addr	High addr	
	A,　[HL]	1 0 1 1　1 1 1 1			
	A,　[HL＋byte]	1 0 1 1　1 1 0 1	Data		
AND	A, #byte	0 1 1 0　0 0 1 1	Data		
	saddr, #byte	0 1 1 0　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	0 1 1 0　R_2 R_1 R_0 1		
	A, saddr	0 1 1 0　0 1 0 1	Saddr-offset		
	A, !addr16	0 1 1 0　1 0 0 1	Low addr	High addr	
	A,　[HL]	0 1 1 0　1 1 1 1			
	A,　[HL＋byte]	0 1 1 0　1 1 0 1	Data		

ニモニック	オペランド	命令コード			
		B1	B2	B3	B4
OR	A, #byte	0 1 1 1　0 0 1 1	Data		
	saddr, #byte	0 1 1 1　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	0 1 1 1　R_2 R_1 R_0 1		
	A, saddr	0 1 1 1　0 1 0 1	Saddr-offset		
	A, !addr16	0 1 1 1　1 0 0 1	Low addr	High addr	
	A,　[HL]	0 1 1 1　1 1 1 1			
	A,　[HL＋byte]	0 1 1 1　1 1 0 1	Data		
XOR	A, #byte	0 1 0 0　0 0 1 1	Data		
	saddr, #byte	0 1 0 0　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	0 1 0 0　R_2 R_1 R_0 1		
	A, saddr	0 1 0 0　0 1 0 1	Saddr-offset		
	A, !addr16	0 1 0 0　1 0 0 1	Low addr	High addr	
	A,　[HL]	0 1 0 0　1 1 1 1			
	A,　[HL＋byte]	0 1 0 0　1 1 0 1	Data		
CMP	A, #byte	0 0 0 1　0 0 1 1	Data		
	saddr, #byte	0 0 0 1　0 0 0 1	Saddr-offset	Data	
	A, r	0 0 0 0　1 0 1 0	0 0 0 1　R_2 R_1 R_0 1		
	A, saddr	0 0 0 1　0 1 0 1	Saddr-offset		
	A, !addr16	0 0 0 1　1 0 0 1	Low addr	High addr	
	A,　[HL]	0 0 0 1　1 1 1 1			
	A,　[HL＋byte]	0 0 0 1　1 1 0 1	Data		
ADDW	AX, #word	1 1 0 1　0 0 1 0	Low byte	High byte	
SUBW	AX, #word	1 1 0 0　0 0 1 0	Low byte	High byte	
CMPW	AX, #word	1 1 1 0　0 0 1 0	Low byte	High byte	
INC	r	0 0 0 0　1 0 1 0	1 1 0 0　R_2 R_1 R_0 1		
	saddr	1 1 0 0　0 1 0 1	Saddr-offset		
DEC	r	0 0 0 0　1 0 1 0	1 1 0 1　R_2 R_1 R_0 1		
	saddr	1 1 0 1　0 1 0 1	Saddr-offset		
INCW	rp	1 0 0 0　P_1 P_0 0 0			
DECW	rp	1 0 0 1　P_1 P_0 0 0			
ROR	A, 1	0 0 0 0　0 0 0 0			
ROL	A, 1	0 0 0 1　0 0 0 0			
RORC	A, 1	0 0 0 0　0 0 1 0			
ROLC	A, 1	0 0 0 1　0 0 1 0			

ニモニック	オペランド	命令コード			
		B1	B2	B3	B4
SET1	saddr.bit	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　1 0 1 0	Saddr-offset	
	sfr.bit	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　0 1 1 0	Sfr-offset	
	A.bit	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　0 0 1 0		
	PSW.bit	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　1 0 1 0	0 0 0 1　1 1 1 0	
	［HL］.bit	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　1 1 1 0		
CLR1	saddr.bit	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　1 0 1 0	Saddr-offset	
	sfr.bit	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　0 1 1 0	Sfr-offset	
	A.bit	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　0 0 1 0		
	PSW.bit	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　1 0 1 0	0 0 0 1　1 1 1 0	
	［HL］.bit	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　1 1 1 0		
SET1	CY	0 0 0 1　0 1 0 0			
CLR1	CY	0 0 0 0　0 1 0 0			
NOT1	CY	0 0 0 0　0 1 1 0			
CALL	!addr16	0 0 1 0　0 0 1 0	Low addr	High addr	
CALLT	［addr5］	0 1　ta_{4-0}　0			
RET		0 0 1 0　0 0 0 0			
RETI		0 0 1 0　0 1 0 0			
PUSH	PSW	0 0 1 0　1 1 1 0			
	rp	1 0 1 0　P_1 P_0 1 0			
POP	PSW	0 0 1 0　1 1 0 0			
	rp	1 0 1 0　P_1 P_0 0 0			
MOVW	SP, AX	1 1 1 0　0 1 1 0	0 0 0 1　1 1 0 0		
	AX, SP	1 1 0 1　0 1 1 0	0 0 0 1　1 1 0 0		
BR	!addr16	1 0 1 1　0 0 1 0	Low addr	High addr	
	$addr16	0 0 1 1　0 0 0 0	jdisp		
	AX	1 0 1 1　0 0 0 0			
BC	$addr16	0 0 1 1　1 0 0 0	jdisp		
BNC	$addr16	0 0 1 1　1 0 1 0	jdisp		
BZ	$addr16	0 0 1 1　1 1 0 0	jdisp		
BNZ	$addr16	0 0 1 1　1 1 1 0	jdisp		
BT	saddr.bit, $addr16	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　1 0 0 0	Saddr-offset	jdisp
	sfr.bit, $addr16	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　0 1 0 0	Sfr-offset	jdisp
	A.bit, $addr16	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　0 0 0 0	jdisp	
	PSW.bit, $addr16	0 0 0 0　1 0 1 0	1 B_2 B_1 B_0　1 0 0 0	0 0 0 1　1 1 1 0	jdisp

ニモニック	オペランド	命令コード			
		B1	B2	B3	B4
BF	saddr.bit, $addr16	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　1 0 0 0	Saddr-offset	jdisp
	sfr.bit, $addr16	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　0 1 0 0	Sfr-offset	jdisp
	A.bit, $addr16	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　0 0 0 0	jdisp	
	PSW.bit, $addr16	0 0 0 0　1 0 1 0	0 B_2 B_1 B_0　1 0 0 0	0 0 0 1　1 1 1 0	jdisp
DBNZ	B, $addr16	0 0 1 1　0 1 1 0	jdisp		
	C, $addr16	0 0 1 1　0 1 0 0	jdisp		
	saddr, $addr16	0 0 1 1　0 0 1 0	Saddr-offset	jdisp	
NOP		0 0 0 0　1 0 0 0			
EI		0 0 0 0　1 0 1 0	0 1 1 1　1 0 1 0	0 0 0 1　1 1 1 0	
DI		0 0 0 0　1 0 1 0	1 1 1 1　1 0 1 0	0 0 0 1　1 1 1 0	
HALT		0 0 0 0　1 1 0 0			
STOP		0 0 0 0　1 1 1 0			

第5章　命令の説明

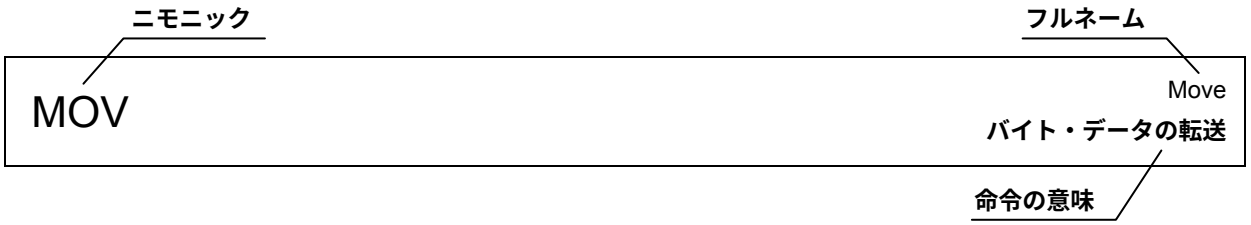
この文章では，78K/0S**シリーズ製品の命令の説明をします。各命令は，ニモニック単位で，複数のオペランドをまとめて説明します。**

命令の説明の基本構成を次ページに示します。

なお，命令のバイト数，命令コードは，第4**章　命令セット**を参照してください。

78K/0S**シリーズ製品の命令は，すべて共通です。**

記　述　例



【命　令　形　式】　MOV dst, src　**：命令の基本記述形式を示します。**

【オペレーション】　dst←src　　**：命令のオペレーションを略号を用いて示します。**

【オ ペ ラ ン ド】　**：この命令で指定できるオペランドを示します。各オペランドの略号の説明は，**4. 1　**オペレーションを参照してください。**

ニモニック	オペランド（dst, src）
MOV	r, #byte
	A, saddr
	saddr, A
	PSW, #byte

ニモニック	オペランド（dst, src）
MOV	A, PSW
	[HL] , A
	A, [HL＋byte]
	[HL＋C] , A

【フ　ラ　グ】　**：命令実行により変化するフラグの動作を示します。**
各フラグの動作記号を凡例に示します。

Z	AC	CY

凡　例

記号	解　説
ブランク	変化なし
0	0にクリアされる
1	1にセットされる
×	結果に従ってセットまたはクリアされる
R	以前に退避した値がリストアされる

【説　明】　**：命令のオペレーションの詳細を解説します。**

●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。

【記　述　例】

MOV A, #4DH**；Aレジスタに**4DH**を転送**

5. 1　8ビット・データ転送命令

8ビット・データ転送命令には，次の命令があります。

MOV

Move
バイト・データの転送

【命　令　形　式】　MOV 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）
MOV	r, #byte
	saddr, #byte
	sfr, #byte
	A, r 注
	r, A 注
	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, !addr16

ニモニック	オペランド（dst, src）
MOV	!addr16, A
	PSW, #byte
	A, PSW
	PSW, A
	A, ［DE］
	［DE］,A
	A, ［HL］
	［HL］,A
	A, ［HL＋byte］
	［HL＋byte］,A

注　r = Aを除く。

【フ　ラ　グ】

PSW, #byteとPSW, A

のオペランドの場合

Z	AC	CY
×	×	×

左記以外

Z	AC	CY

【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）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。

●MOV PSW, #byte命令，MOV PSW, A命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

MOV A, #4DH；Aレジスタに4DHを転送

XCH

Exchange
バイト・データの交換

【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	A, X
	A, r 注
	A, saddr
	A, sfr
	A, ［DE］
	A, ［HL］
	A, ［HL＋byte］

注　r = A, Xを除く。

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランドと第2オペランドの内容を交換します。

【記　述　例】

XCH A, 0FEBCH；Aレジスタの内容とFEBCH番地の内容を交換

5. 2　16ビット・データ転送命令

16ビット・データ転送命令には，次の命令があります。

MOVW

Move Word
ワード・データの転送

【命　令　形　式】　MOVW 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）
MOVW	rp, #word
	AX, saddrp
	saddrp, AX
	AX, rp 注
	rp, AX 注

注　rp = BC, DE, HL**のときのみ。**

【フ　ラ　グ】

Z	AC	CY

【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）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。

【記　述　例】

MOVW AX, HL **；HLレジスタの内容を**AX**レジスタへ転送**

【注　　意】

saddrp**に対するアドレスの指定は偶数アドレスのみです。奇数アドレスは指定できません。**

XCHW

Exchange Word
ワード・データの交換

【命　令　形　式】　XCHW 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）
XCHW	AX, rp 注

注　rp = BC, DE, HL**のときのみ。**

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランドと第2オペランドの内容を交換します。

【記　述　例】

XCHW AX,BC **；AXレジスタと**BC**レジスタの内容を交換**

5. 3　8ビット演算命令

8ビット演算命令には，次の命令があります。

ADD

Add
バイト・データの加算

【命　令　形　式】　ADD dst, src

【オペレーション】　dst, CY←dst＋src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADD	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
ADD	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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）を加算し，その結果をCYフラグとデスティネーション・オペランド（dst）へ格納します。**
- **加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。**
- **加算の結果，ビット7からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。**
- **加算の結果，ビット3からビット4へのキャリーが発生した場合は，ACフラグはセット（1），その他の場合はACフラグはクリア（0）されます。**

【記　述　例】

ADD CR10, #56H；CR10**レジスタに**56H**を加算し，結果を**CR10**レジスタへ格納**

ADDC

Add with Carry
キャリーを含むバイト・データの加算

【命　令　形　式】　ADDC dst, src

【オペレーション】　dst, CY←dst＋src＋CY

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADDC	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
ADDC	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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）とCYフラグを加算して，結果をデスティネーション・オペランド（dst）とCYフラグに格納します。

CYフラグは最下位ビットへ加算されます。

この命令は，主として複数バイトの加算を行うときに使用します。

●加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。

●加算の結果，ビット7からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。

●加算の結果，ビット3からビット4へのキャリーが発生した場合は，ACフラグはセット（1），その他の場合はACフラグがクリア（0）されます。

【記　述　例】

ADDC A,［HL］；**Aレジスタと（HLレジスタ）番地の内容とCYフラグを加算し，結果をAレジスタに格納**

SUB

Subtract
バイト・データの減算

【命　令　形　式】　SUB dst, src

【オペレーション】　dst, CY←dst－src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUB	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
SUB	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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，結果をデスティネーション・オペランド（dst）とCYフラグへ格納します。

ソース・オペランド（src）とデスティネーション・オペランド（dst）を同一のものとすることにより，デスティネーション・オペランドの0クリアが可能です。

●減算の結果，dstが0なら，Zフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット7でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

SUB A, D；**Aレジスタから**D**レジスタを減算し，結果を**A**レジスタへ格納**

SUBC

Subtract with Carry
キャリーを含むバイト・データの減算

【命　令　形　式】　SUBC dst, src

【オペレーション】　dst, CY←dst－src－CY

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUBC	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
SUBC	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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）とCYフラグを減算し，結果をデスティネーション・オペランド（dst）へ格納します。

CYフラグは最下位ビットから減算します。

この命令は，主として複数バイトの減算を行うときに使用します。

●減算の結果，dstが0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット7でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合は，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

SUBC A, ［HL］；**Aレジスタから（HLレジスタ）番地の内容とCYフラグを減算し，結果をAレジスタへ格納**

AND

And
バイト・データの論理積

【命　令　形　式】　AND dst, src

【オペレーション】　dst←dst ∧ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
AND	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
AND	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×		

【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）のビットごとの論理積をとり，結果をデスティネーション・オペランド（dst）へ格納します。**
- **論理積をとった結果，全ビットが0であればZフラグはセット（1），その他の場合は，Zフラグはクリア（0）されます。**

【記　述　例】

AND 0FEBAH, #11011100B；FEBAHの内容と11011100Bのビットごとの論理積をとり，結果をFEBAHへ格納

OR

Or
バイト・データの論理和

【命　令　形　式】　OR dst, src

【オペレーション】　dst←dst ∨ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
OR	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
OR	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×		

【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）のビットごとの論理和をとり，結果をデスティネーション・オペランド（dst）へ格納します。

●論理和をとった結果，全ビットが0であればZフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

OR A, 0FE98H；Aレジスタと FE98Hのビットごとの論理和をとり，結果をAレジスタへ格納

XOR

Exclusive Or
バイト・データの排他的論理和

【命　令　形　式】　XOR dst, src

【オペレーション】　dst←dst ∀ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XOR	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
XOR	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×		

【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）のビットごとの排他的論理和をとり，結果をデスティネーション・オペランド（dst）へ格納します。

この命令でソース・オペランド（src）に#0FFHを選択することにより，デスティネーション・オペランド（dst）の全ビットの論理否定がとれます。

●排他的論理和の結果，全ビットが0であればZフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

XOR A, L；**Aレジスタと**L**レジスタのビットごとの排他的論理和をとり，結果を**A**レジスタへ格納**

CMP

Compare
バイト・データの比較

【命　令　形　式】　CMP 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）
CMP	A, #byte
	saddr, #byte
	A, r
	A, saddr

ニモニック	オペランド（dst, src）
CMP	A, !addr16
	A, ［HL］
	A, ［HL＋byte］

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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にZ, AC, CY**の各フラグだけを変化させます。**

●減算の結果，0ならZ**フラグはセット（1），その他の場合は**Z**フラグはクリア（0）されます。**

●減算の結果，ビット7でボローが発生した場合，CY**フラグはセット（1），その他の場合はクリア（0）されます。**

●減算の結果，ビット4からビット3へのボローが発生した場合，AC**フラグはセット（1），その他の場合はクリア（0）されます。**

【記　述　例】

CMP 0FE38H, #38H；FE38H**番地の内容から**38H**を減算し，フラグだけを変化させる（**FE38H**番地の内容とイミーディエト・データの比較）**

5. 4　16ビット演算命令

16ビット演算命令には，次の命令があります。

ADDW

Add Word
ワード・データの加算

【命　令　形　式】　ADDW dst, src

【オペレーション】　dst,CY←dst＋src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADDW	AX, #word

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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）の加算を行い，結果をデスティネーション・オペランド（dst）へ格納します。**
- **加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。**
- **加算の結果，ビット15からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。**
- **加算の結果，ACフラグは不定となります。**

【記　述　例】

ADDW AX, #0ABCDH；AXレジスタとABCDHを加算し，結果をAXレジスタへ格納

SUBW

Subtract Word
ワード・データの減算

【命　令　形　式】　SUBW dst, src

【オペレーション】　dst, CY←dst－src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUBW	AX, #word

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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，結果をデスティネーション・オペランド（dst）とCYフラグへ格納します。

ソース・オペランド（src）とデスティネーション・オペランド（dst）を同一のものとすることにより，デスティネーション・オペランドの0クリアが可能です。

●減算の結果，dstが0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット15でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ACフラグは不定となります。

【記　述　例】

SUBW AX, #0ABCDH；AXレジスタの内容からABCDHを減算し，結果をAXレジスタへ格納

CMPW

Compare Word
ワード・データの比較

【命　令　形　式】　CMPW 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）
CMPW	AX, #word

【フ　ラ　グ】

Z	AC	CY
×	×	×

【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にZ, AC, CY**の各フラグだけを変化させます。**
- **減算の結果，0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。**
- **減算の結果，ビット**15**でボローが発生した場合，**CY**フラグはセット（**1**），その他の場合はクリア（**0**）されます。**
- **減算の結果，**AC**フラグは不定となります。**

【記　述　例】

CMPW AX, #0ABCDH；AX**レジスタから**ABCDH**を減算し，フラグだけを変化させる（**AX**レジスタとイミーディエト・データとの比較）**

5. 5　増減命令

増減命令には，次の命令があります。

INC

Increment
バイト・データのインクリメント

【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	r
	saddr

【フ　ラ　グ】

Z	AC	CY
×	×	

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけインクリメントします。

●インクリメントした結果が0になればZフラグはセット（1），その他の場合はクリア（0）されます。

●インクリメントした結果，ビット3からビット4へのキャリーがあれば，ACフラグはセット（1），その他の場合はクリア（0）されます。

●繰り返し処理のカウンタに使用することが多いため，CYフラグの内容は変化させません（複数バイトの演算時に，CYフラグの内容を保持させるため）。

【記　述　例】

INC B；Bレジスタをインクリメント

DEC

Decrement
バイト・データのデクリメント

【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（dst）
DEC	r
	saddr

【フ　ラ　グ】

Z	AC	CY
×	×	

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけデクリメントします。

●デクリメントした結果が0であれば，Zフラグはセット（1），その他の場合はクリア（0）されます。

●デクリメントした結果がビット4からビット3へのキャリーがあれば，ACフラグはセット（1），その他の場合はクリア（0）されます。

●繰り返し処理のカウンタに使用することが多いため，CYフラグの内容は変化させません（複数バイトの演算時にCYフラグを保持させるため）。

●dstがBレジスタ，Cレジスタ，またはsaddrの場合でAC, CYの各フラグを変化させたくない場合，DBNZ命令を使用することができます。

【記　述　例】

DEC 0FE92H；FE92H番地の内容をデクリメント

INCW

Increment Word
ワード・データのインクリメント

【命　令　形　式】　INCW 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）
INCW	rp

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）**の内容を**1**だけインクリメントします。**

●レジスタを使用するアドレシングで，使用するレジスタ（ポインタ）のインクリメントに使用することが多いため，Z, AC, CY**の各フラグを変化させません。**

【記　　述　　例】

INCW HL；HL**レジスタをインクリメント**

DECW

Decrement Word
ワード・データのデクリメント

【命　令　形　式】　DECW 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）
DECW	rp

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）**の内容を**1**だけデクリメントします。**

●レジスタを使用するアドレシングで，使用するレジスタ（ポインタ）のデクリメントに使用することが多いため，Z, AC, CY**の各フラグを変化させません。**

【記　　述　　例】

DECW DE；DE**レジスタをデクリメント**

5. 6　ローテート命令

ローテート命令には，次の命令があります。

ROR

Rotate Right
バイト・データの右方向のローテート

【命　令　形　式】　ROR dst, cnt

【オペレーション】　（CY, dst_7←dst_0, dst_{m-1}←dst_m）×1回

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
ROR	A, 1

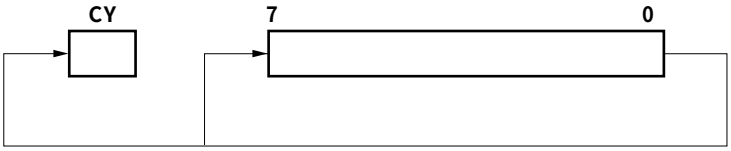
【フ　ラ　グ】

Z	AC	CY
		×

【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）の内容を，1回だけ右方向へ回転させます。

●LSB（ビット0）の内容はMSB（ビット7）へ回転されると同時にCYフラグへも転送されます。



【記　述　例】

ROR A, 1；Aレジスタの内容を右へ1ビット回転

ROL

Rotate Left
バイト・データの左方向のローテート

【命　令　形　式】　ROL dst, cnt

【オペレーション】　（CY, dst_0←dst_7, dst_{m+1}←dst_m）×1回

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
ROL	A, 1

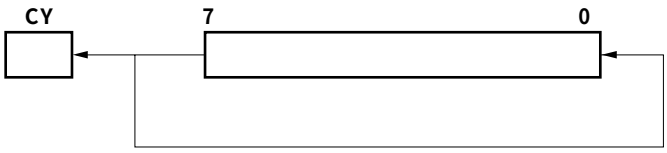
【フ　ラ　グ】

Z	AC	CY
		×

【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）の内容を1回だけ左方向へ回転させます。

●MSB（ビット7）の内容は，LSB（ビット0）へ回転されると同時にCYフラグへも転送されます。



【記　述　例】

ROL A, 1；Aレジスタの内容を左へ1ビット回転

RORC

Rotate Right with Carry
キャリーを含むバイト・データの右方向のローテート

【命　令　形　式】　RORC dst, cnt

【オペレーション】　（CY←dst_0, dst_7←CY, dst_{m-1}←dst_m）×1**回**

【オ ペ ラ ン ド】

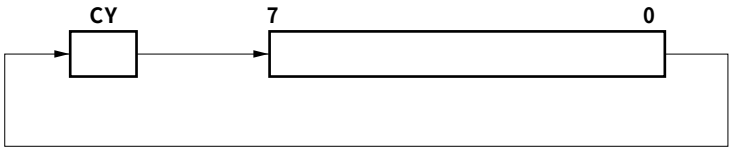
ニモニック	オペランド（dst, cnt）
RORC	A, 1

【フ　ラ　グ】

Z	AC	CY
		×

【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）の内容を，CYフラグを含め，1回だけ右方向へ回転します。



【記　述　例】

RORC A, 1；**Aレジスタの内容を，CYフラグを含めて1ビット右方向へ回転**

ROLC

Rotate Left with Carry
キャリーを含むバイト・データの左方向のローテート

【命　令　形　式】　ROLC dst, cnt

【オペレーション】　（CY←dst_7, dst_0←CY, dst_{m+1}←dst_m）×1**回**

【オ ペ ラ ン ド】

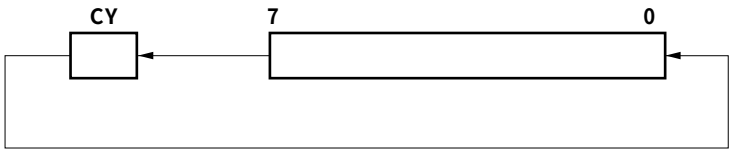
ニモニック	オペランド（dst, cnt）
ROLC	A, 1

【フ　ラ　グ】

Z	AC	CY
		×

【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）の内容を，CYフラグを含め，1回だけ左方向へ回転させます。



【記　述　例】

ROLC A, 1；**Aレジスタの内容を，CYフラグを含めて1ビット左へ回転**

5. 7　ビット操作命令

ビット操作命令には，次の命令があります。

SET1

Set Single Bit（Carry Flag）
1ビット・データのセット

【命　令　形　式】　SET1 dst

【オペレーション】　dst←1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
SET1	saddr.bit
	sfr.bit
	A.bit
	PSW.bit
	［HL］.bit
	CY

【フ　ラ　グ】

dstがPSW.bit

Z	AC	CY
×	×	×

dstがCY

Z	AC	CY
		1

左記以外

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）をセット（1）します。

●デスティネーション・オペランド（dst）がCY，またはPSW.bitの場合，該当するフラグのみがセット（1）されます。

【記　述　例】

SET1 0FE55H.1；FE55Hのビット1をセット（1）する。

CLR1

Clear Single Bit（Carry Flag）
1ビット・データのクリア

【命　令　形　式】　CLR1 dst

【オペレーション】　dst←0

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
CLR1	saddr.bit
	sfr.bit
	A.bit
	PSW.bit
	［HL］.bit
	CY

【フ　ラ　グ】

dstがPSW.bit

Z	AC	CY
×	×	×

dstがCY

Z	AC	CY
		0

左記以外

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）をクリア（0）します。

●デスティネーション・オペランド（dst）がCY，またはPSW.bitの場合，該当するフラグのみがクリア（0）されます。

【記　述　例】

CLR1 P3.7；ポート3のビット7をクリア（0）する。

NOT1

Not Single Bit（Carry Flag）
1ビット・データの論理否定

【命　令　形　式】　NOT1 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）
NOT1	CY

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●CY**フラグを反転します。**

【記　述　例】

NOT1 CY；CY**フラグを反転します。**

5. 8　コール・リターン命令

コール・リターン命令には，次の命令があります。

CALL

Call
サブルーチン・コール（16ビット直接）

【命　令　形　式】　CALL target

【オペレーション】　（SP－1）　←（PC＋3）$_H$，
（SP－2）　←（PC＋3）$_L$，
SP　←SP－2，
PC　←target

【オ ペ ラ ン ド】

ニモニック	オペランド（target）
CALL	!addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●16ビットの絶対アドレスまたは，レジスタ間接アドレスによるサブルーチン・コールです。

●次の命令の先頭アドレス（PC＋3）をスタックに退避し，ターゲット・オペランド（target）で指定されるアドレスへ分岐します。

【記　述　例】

CALL　!3059H；3059H**へサブルーチン・コールする**

CALLT

Call Table
サブルーチン・コール（コール・テーブル参照）

【命　令　形　式】　CALLT［addr5］

【オペレーション】　（SP－1）　←（PC＋1）$_H$，
（SP－2）　←（PC＋1）$_L$，
SP　←　SP－2，
PC_H　←（00000000, addr5＋1）
PC_L　←（00000000, addr5）

【オ ペ ラ ン ド】

ニモニック	オペランド（［addr5］）
CALLT	[addr5]

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●コール・テーブル参照のサブルーチン・コールです。

●次の命令の先頭アドレス（PC＋1）をスタックに退避し，コール・テーブル（アドレスの上位8ビットは00000000Bに固定で，次の5ビットをaddr5で指定します）のワード・データで示されるアドレスへ分岐します。

【記　述　例】

CALLT［40H］；0040H, 0041H番地にあるワード・データをアドレスとして，そのアドレスへサブルーチン・コール

RET

Return
サブルーチンからの復帰

【命　令　形　式】　RET

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

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●CALL, CALLT命令でコールされたサブルーチン・コールからのリターン命令です。

●スタックに退避されているワード・データをPCに復帰し，サブルーチンからリターンします。

RETI

Return from Interrupt
ハードウエア・ベクタ割り込みからの復帰

【命　令　形　式】　RETI

【オペレーション】　PC_L　←（SP），
PC_H　←（SP＋1），
PSW　←（SP＋2），
SP　←　SP＋3，

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY
R	R	R

【説　　明】

●ベクタ割り込みからの復帰命令です。

●スタック退避されているデータをPCとPSWに復帰し，割り込み処理ルーチンからリターンします。

●この命令と次に実行する命令の間では，すべての割り込みを受け付けません。

●NMISフラグはノンマスカブル割り込み受け付けにより1にセットされ，RETI命令により0にクリアされます。

【注　　意】

ノンマスカブル割り込み処理からの復帰を，RETI命令以外の命令で行うと，NMISフラグが0にクリアされないためすべての割り込み（ノンマスカブル割り込みを含む）を受け付けなくなります。

5. 9　スタック操作命令

スタック操作命令には，次の命令があります。

PUSH

Push
プッシュ

【命　令　形　式】　PUSH src

【オペレーション】

srcがrpの場合	srcがPSWの場合
(SP－1)　←src_H,	(SP－1)　←src
(SP－2)　←src_L,	SP　←SP－1
SP　←SP－2	

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
PUSH	PSW
	rp

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●ソース・オペランド（src）で指定されたレジスタのデータをスタックに退避します。

【記　述　例】

PUSH AX；AX**レジスタの内容をスタックへ退避**

POP

Pop
ポップ

【命　令　形　式】　POP dst

【オペレーション】

dstがrpの場合	dstがPSWの場合
dst_L ←（SP），	dst ←（SP）
dst_H ←（SP＋1），	SP ←　SP＋1
SP　←　SP＋2	

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
POP	PSW
	rp

【フ　ラ　グ】

dstがrp

Z	AC	CY

PSW

Z	AC	CY
R	R	R

【説　　明】

●デスティネーション・オペランド（dst）で指定されたレジスタへ，データをスタックから復帰します。

●オペランドがPSWの場合，各フラグはスタックのデータで置き換わります。

●POP PSW命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

POP AX；AXレジスタへスタックのデータを復帰

MOVW SP, AX
MOVW AX, SP

Move Word
スタック・ポインタとのワード・データの転送

【命　令　形　式】　MOVW 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）
MOVW	SP, AX
	AX, SP

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●スタック・ポインタの内容を操作するための命令です。

●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）へ第2オペランドで指定されるソース・オペランド（src）を格納します。

【記　述　例】

MOVW SP, AX**；スタック・ポインタへ**AX**レジスタの内容を格納**

5. 10　無条件分岐命令

無条件分岐命令には，次の命令があります。

BR

Branch
無条件分岐

【命　令　形　式】　BR target

【オペレーション】　PC←target

【オ ペ ラ ン ド】

ニモニック	オペランド（target）
BR	!addr16
	AX
	$addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●無条件に分岐を行う命令です。

●ターゲット・アドレス・オペランド（target）のワード・データをPC**に転送し，分岐します。**

【記　述　例】

BR AX；AX**レジスタの内容をアドレスとして分岐**

5. 11　条件付き分岐命令

条件付き分岐命令には，次の命令があります。

BC

Branch if Carry

キャリー・フラグによる条件分岐（CY = 1）

【命　令　形　式】　BC $addr16

【オペレーション】　PC←PC＋2＋jdisp8 if CY = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr16）
BC	$addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●CY = 1**の場合に，オペランドで指定されたアドレスへ分岐します。**

CY = 0**の場合，何も処理を行わず，次に続く命令を実行します。**

【記　述　例】

BC $300H；CY = 1**なら**0300H**へ分岐（ただし，この命令の先頭は**027FH-037EH**番地内にある）。**

BNC

Branch if Not Carry

キャリー・フラグによる条件分岐（CY = 0）

【命　令　形　式】　BNC $addr16

【オペレーション】　PC←PC＋2＋jdisp8 if CY = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr16）
BNC	$addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●CY = 0の場合に，オペランドで指定されたアドレスへ分岐します。

CY = 1の場合，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BNC $300H；CY = 0なら0300Hへ分岐（ただし，この命令の先頭は027FH-037EH番地内にある）。

BZ

Branch if Zero
ゼロ・フラグによる条件分岐（Z = 1）

【命　令　形　式】　BZ $addr16

【オペレーション】　PC←PC＋2＋jdisp8 if Z = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr16）
BZ	$addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●Z = 1の場合に，オペランドで指定されたアドレスへ分岐します。

Z = 0の場合は，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

```
DEC B
BZ $3C5H；Bレジスタが0なら03C5Hに分岐（ただし，この命令の先頭は，0344H-0443H番地内にある）。
```

BNZ

Branch if Not Zero
ゼロ・フラグによる条件分岐（Z = 0）

【命　令　形　式】　BNZ $addr16

【オペレーション】　PC←PC＋2＋jdisp8 if Z = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr16）
BNZ	$addr16

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●Z = 0**の場合に，オペランドで指定されたアドレスへ分岐します。**

Z = 1**の場合は，何も処理を行わず，次に続く命令を実行します。**

【記　述　例】

```
CMP A, #55H
BNZ  $0A39H ; Aレジスタが0055Hでないとき，0A39Hへ分岐（ただし，この命令の先頭は09B8H-0AB7H番
             地内にある）。
```

BT

Branch if True

ビット・テストによる条件分岐（バイト・データのビット= 1）

【命　令　形　式】　BT bit, $addr16

【オペレーション】　PC←PC＋b＋jdisp8 if bit = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（bit, $addr16）	b（バイト数）
BT	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	4

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランド（bit）の内容がセット（1）されているとき，第2オペランド（$addr16）で指定されるアドレスへ分岐します。

第1オペランド（bit）の内容がセット（1）されていないときは，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BT　0FE47H.3,　$55CH **；FE47H番地のビット3が1のとき，055CHへ分岐（ただし，この命令の先頭は，04DAH-05D9H番地内にある）。**

BF

Branch if False

ビット・テストによる条件分岐（バイト・データのビット= 0**）**

【命　令　形　式】　BF bit, $addr16

【オペレーション】　PC←PC＋b＋jdisp8 if bit = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（bit, $addr16）	b（バイト数）
BF	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	4

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1**オペランド（**bit**）の内容がクリア（**0**）されているとき，第**2**オペランド（**$addr16**）で指定されるアドレスへ分岐します。**

第1**オペランド（**bit**）の内容がクリア（**0**）されていないときは，何も処理を行わず，次に続く命令を実行します。**

【記　述　例】

BF P2.2, $1549H**；ポート**2**のビット**2**が**0**のとき，**1549H**番地へ分岐（ただし，この命令の先頭は，**14C6H-15C5H**番地内にある）。**

DBNZ

Decrement and Branch if Not Zero
条件ループ（R1≠0）

【命　令　形　式】　DBNZ dst, $addr16

【オペレーション】　dst←dst－1,
then PC←PC＋b＋jdisp16 if dst R1≠0

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, $addr16）	b（バイト数）
DBNZ	B, $addr16	2
	C, $addr16	2
	saddr, $addr16	3

【フ　ラ　グ】

Z	AC	CY

【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）の内容を－1して，デスティネーション・オペランド（dst）へ格納します。**
- **デスティネーション・オペランド（dst）を－1した結果が0でなかった場合，第2オペランド（$addr16）で示されるアドレスへ分岐します。**
  デスティネーション・オペランド（dst）を－1した結果が0のときは，何も処理を行わず，次に続く命令を実行します。
- **フラグは変化しません。**

【記　述　例】

DBNZ B, $1215H；**Bレジスタの内容をデクリメントし，0にならなければ**1215H**へ分岐（ただし，この命令の先頭は，**1194H-1293H**番地内にある）。**

5. 12　CPU制御命令

CPU制御命令には，次の命令があります。

NOP

No Operation
ノー・オペレーション

【命　令　形　式】　NOP

【オペレーション】　no operation

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●何も処理をせずに時間だけを消費します。

EI

Enable Interrupt
割り込みの許可

【命　令　形　式】　EI

【オペレーション】　IE←1

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●マスカブル割り込みの受け付け可能な状態にします（割り込み許可フラグ（IE）をセット（1）します）。

●この命令実行後，すぐに割り込みを受け付けます。

●この命令を実行しても，他の要因によりベクタ割り込みの受け付けを行わないようにすることができます。
詳細については，各製品のユーザーズ・マニュアルの割り込み機能を参照してください。

DI

Disable Interrupt
割り込みの禁止

【命　令　形　式】　DI

【オペレーション】　IE←0

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

- **マスカブル割り込みのベクタ割り込みによる受け付けを禁止します（割り込み許可フラグ（IE）をクリア（0）します）。**
- **この命令と次に続く1命令の間では，すべての割り込みを受け付けません。**
- **割り込み処理の詳細については，各製品のユーザーズ・マニュアルの割り込み機能を参照してください。**

HALT

Halt
ホルト・モードの設定

【命　令　形　式】　HALT

【オペレーション】　Set HALT Mode

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●HALT**モードになります。**CPU**の動作クロックを停止させるモードです。通常動作モードとの組み合わせによる間欠動作により，システムのトータル消費電力を低下させることができます。**

STOP

Stop
ストップ・モードの設定

【命　令　形　式】　STOP

【オペレーション】　Set STOP Mode

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●STOPモードになります。メイン・システム・クロック発振回路を停止させ，システム全体が停止するモードです。リーク電流だけの超低消費電力にすることができます。

付録A　命令索引（ニモニック：機能別）

【無条件分岐命令】

【条件付き分岐命令】

【CPU制御命令】

付録B　命令索引（ニモニック：アルファベット順）

【P】

【R】

【S】

【X】

付録C　改版履歴

C. 1　本版で改訂された主な箇所

箇　所	内　容
全般	78K/0Sシリーズ共通の情報以外は削除（製品個別の情報については，各製品のユーザーズ・マニュアルを参照）。

C. 2　前版までの改版履歴

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版　数	前版からの主な改版内容	適用箇所
第2版	次の製品を追加 μPD789026, 789407, 789417, 789800, 789806Yサブシリーズ	全般
	78K/0Sシリーズ製品別内部データ・メモリ空間一覧のフォーマットを変更	第1章　メモリ空間
第3版	次の製品を追加 μPD789046, 789104, 789114, 789124, 789134, 789146, 789156, 789167, 789177, 789197AY, 789217AY, 789407A, 789417A, 789842サブシリーズ	全般
	次の製品を削除 μPD789407, 7894717, 789806Yサブシリーズ	
	MOV PSW, #byteの命令コードを変更	第4章　命令セット
	XOR A, rの命令コードを変更	
	CMP A, rの命令コードを変更	
	CLR1［HL］. bitの命令コードを変更	

〔メ　モ〕

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニク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.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。