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

IE-78310A-R

インサーキット・エミュレータ

ソフトウエア編

資料番号 EEU-637A（第2版）
発行年月 November 1993 P

この装置は，第１種情報装置（商工業地域において使用されるべき情報装置）で商工業地域での電波障害防止を目的とした情報処理装置等電波障害自主規制協議会（ＶＣＣＩ）基準に適合しております。

従って，住宅地域またはその隣接した地域で使用すると，ラジオ，テレビジョン受信機等に受信障害を与えることがあります。

取扱説明書に従って正しく取り扱いをして下さい。

本製品は外国為替および外国貿易管理法の規定により戦略物資等（または役務）に該当しますので，日本国外に輸出する場合には，同法に基づき日本国政府の輸出許可が必要です。

本版で改訂された主な箇所

箇　所	内　　容
目次の前	使用上の注意事項を追加
ｐ．１	ＭＤシリーズ、ＰＧ－２０００の廃止製品化の記述を追加
ｐ.23,109,110,245,246	ＣＴＲＬ－Ｐ　→　ＣＴＲＬ－Ｒ　に変更

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

使用上の注意事項

ここでは，ＩＥ－７８３１０Ａ－Ｒを使用するうえでの注意事項について説明します。応用製品の開発前には必ずお読みください。なお，（　）内のページは，各注意事項の該当ページを示します。

(1) ＲＵＮ　Ｓコマンドで使用するステップ数は，実行命令のステップ数ではなく，命令のフェッチ数でカウントします。μＰＤ７８３１２Ａはプリフェッチを行うため，実行数とフェッチ数は一致しません。（ｐ．９）

(2) ＩＥ－７８３１０Ａ－Ｒでは，μＰＤ７８３１２Ａのアーキテクチャ上，以下のような使用上の注意が必要です。（ｐ．１０，１４９）

○ブランチ系命令のオペランドではブレークしません。

したがって，ブランチ系命令のオペランドにはブレーク・ポイントを置かないでください。

○ブレーク・ポイントを通過後，数命令実行してから，ブレークします（これをスリップといいます。）

スリップする命令数は，一定していませんが，ブレーク・ポイント後，１バイト命令が続いているとき最も多くスリップします。

また、同じ条件でも内部ＲＯＭ実行時と，外部メモリ実行時でも，スリップする命令数が異なります。

内部ＲＯＭ実行時は，外部メモリ実行時よりも，多くスリップします。

○命令ステップ数をカウントする場合，ＳＦＲに対する一部の命令は，２ステップと数えられます。

○ＣＡＬＬＴ，ＢＲＫ命令あるいは，割り込みによるベクタ参照では正しくブレークしません。したがって，ベクタ・エリアにブレーク・ポイントを置かないでください。

○ベクタ・テーブル領域（０Ｈ－２９Ｈ番地）にジャンプするようなプログラムでは，正しくインストラクション・トレース表示は行えません。したがって，このようなプログラムは書かないようにしてください。

(3) トレース・コマンド実行によりＳＦＲに対する一部の命令をフェッチしたときは，Ｍ１フレームが２回出力されます。（ｐ．１６）

(4) ＴＲＳコマンドは，トレース内容の設定と，クオリファイ条件の設定とを兼ねています。したがって，トレース内容に指定したポート以外でクオリファイ条件を設定することはできません。また、クオリファイ条件に指定したポート以外をトレースすることもできません。（ｐ．１７）

(5) ＰＤＡ－８８０，およびＣＲＴ内蔵タイプのＭＤ－０８６での外部コンソールでは，コマンド／データ・ライン編集機能は使用できません。（ｐ．２２）

(6) ＭＤ－１１６／０８６シリーズのＣＣＰ／Ｍで、マルチウインドウ・モードでは，コマンド／データ・ライン編集機能は使用できません。（ｐ．２２）

(7) スタンドアロン・モードでは，シンボルをローディングすることができません。（ｐ．３０）

(8) μＰＤ７８３１２ＡはＩ－Ｂｕｆｆｅｒを持ち，プリフェッチします。このため，ブランチ系の命令が多いプログラムでは，実行ステップ数とフェッチ数が大きく異なる場合があります。（ｐ．９２）

(9) ＰＧ－２０００の場合プロンプトは，ＩＥ（スタンド・アロン・モード）の場合も，ＰＧの場合も‘＊’ですので注意してください。（ｐ．１３４）

目　　次

第1章　概　　説

ＩＥ－７８３１０Ａ－Ｒは，μＰＤ７８３１２Ａ，μＰＤ７８３１０Ａ，μＰＤ７８Ｐ３１２Ａを用いたハードウエアおよびソフトウエアを，効率的にディバグするための，開発支援装置です。

本取扱説明書は，ＩＥ－７８３１０Ａ－Ｒの開包から実際のディバグ作業，さらに応用方法までについて，詳しく説明しています。

ＩＥ－７８３１０Ａ－Ｒ取扱説明書は，ハードウエア編とソフトウエア編の２冊より構成されており，本書はソフトウエア編となっています。

ハードウエア編	ソフトウエア編
第1章　概　説 *	第1章　概　説 *
第2章　機能概要	第2章　機能概要
第3章　設　置	第3章　基本的な使用方法
第4章　システムの構成方法	第4章　コマンドの説明
第5章　RS-232-Cインタフェースの機能	第5章　応用方法
第6章　ターゲットとの接続方法	
付録A *	付　録 *
付録B	

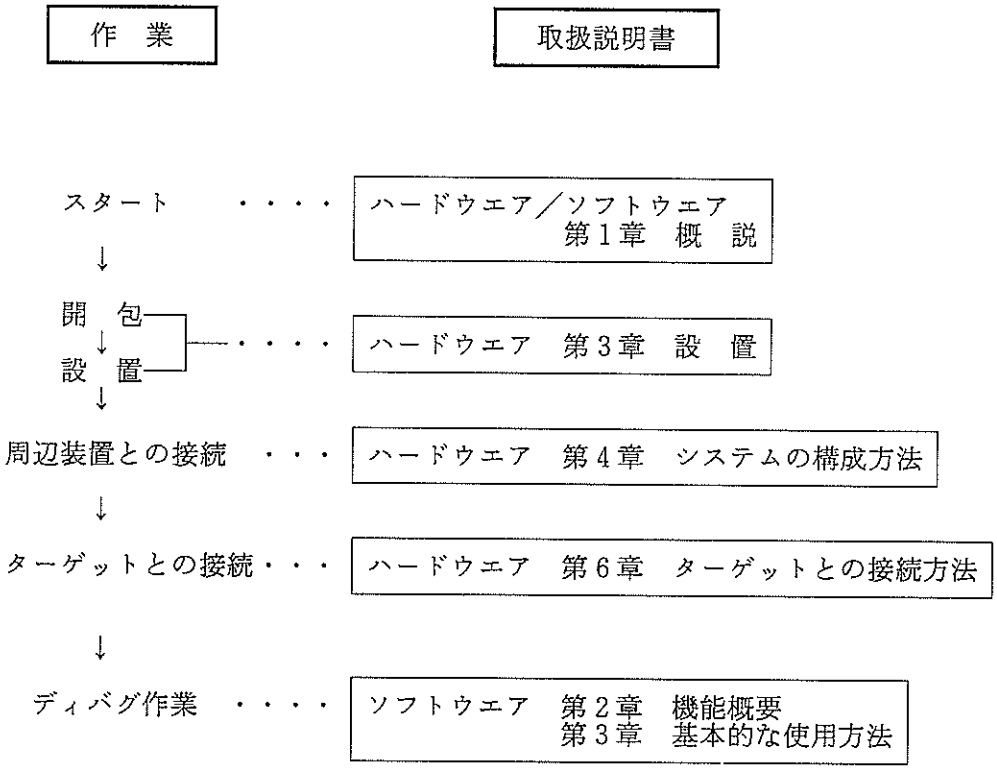
＊　第１章と付録は，ハードウエア編，ソフトウエア編とも共通の内容になっています。

このユーザーズ・マニュアルでは，ホスト・マシンとしてMDシリーズ（コンカレントＣＰ／Ｍ™）を使用して説明しています。ホスト・マシンとしてMDシリーズ以外（ＰＣ－９８００シリーズ，ＩＢＭ　ＰＣ／ＡＴ™）を使用する場合は，ハードウエア編の付録Ｂも参照してください。 ★

備考　MDシリーズ，ＰＧ－２０００は廃止製品です。 ★

1．1　設置からディバグまで

ＩＥ－７８３１０Ａ－Ｒの設置から，実際のディバグまで，本取扱説明書をどのように読み進んでいけばよいのか図示します。



・ＩＥ－７８３１０Ａ－Ｒのディバッガとしての基本仕様や外観について知りたいとき

ハードウエア　第2章　機能概要

・ＩＥ－７８３１０Ａ－ＲのRS-232-Cインタフェースの詳細について知りたいとき

ハードウエア　第5章　RS-232-Cインタフェースの機能

・ＩＥ－７８３１０Ａ－Ｒのコマンドの詳細について知りたいとき

ソフトウエア　第4章　コマンドの説明

・ひと通りの使い方を理解し，さらに効率的なディバグ方法を知りたいとき

ソフトウエア　第5章　応用方法

1．2　取扱説明書の概要

ハードウエア編

第2章　機能概要

ＩＥ－７８３１０Ａ－Ｒディバッガとしての基本仕様や，外観について述べています。

ＩＥ－７８３１０Ａ－Ｒを購入されたらひと通り目を通してください。

第3章　設　　置

ＩＥ－７８３１０Ａ－Ｒの開包方法，付属品の接続方法，本体の設定方法にいて，詳細に説明します。ＩＥ－７８３１０Ａ－Ｒを購入されたらまず最初にこの章を読んでください。

第4章　システムの構成方法

ＩＥ－７８３１０Ａ－Ｒと周辺装置（ホスト・マシン，ターミナル，ＰＲＯＭライタ）の接続方法について，詳細に説明します。

各周辺装置ごとに詳しく説明しますので，お手持ちの周辺装置の項をお読みください。

ハードウエア編の第3章を読んで設置された後は，必ずこの章を読んで周辺装置と接続してください。

第5章　ＲＳ－２３２－Ｃインタフェースの機能

ＩＥ－７８３１０Ａ－Ｒのシリアル・インタフェース（チャネル1，チャネル2）のＲＳ－２３２－Ｃインタフェースとしての機能を詳細に説明します。この章は，弊社の周辺装置を接続する限り読む必要はありません。ＩＥ－７８３１０Ａ－Ｒのシリアル・インタフェースと他社の装置を接続される場合にだけお読みください。

第6章　ターゲットとの接続方法

ＩＥ－７８３１０Ａ－Ｒとμ７８３１２Ａ，μ７８３１０Ａを使用したターゲット・システムとの接続方法について詳細に説明します。ハードウエア編の第4章を読んで周辺装置と接続された後は，必ずこの章をお読みになってターゲット・システムと接続してください。

ソフトウエア編

第２章　機能概要

スタンドアロン時と，システム・ソフトウエア使用時に分けて，ＩＥ－７８３１０Ａ－Ｒの機能を詳細に説明します。

ディバグ作業にかかる前に，ひと通りこの章に目を通してください。

この章では各機能について，実際のコマンドと対応して説明します。したがって，この章をお読みになることで，ＩＥ－７８３１０Ａ－Ｒが持つ機能とそれに対応するコマンドをつかむことができます。

第３章　基本的な使用方法

ＩＥ－７８３１０Ａ－Ｒでディバグ作業を行なうための手順や基本的なコマンドの使用方法について説明します。

ディバグ作業にかかる前に必ずこの章をお読みください。

この章では，まずディバグ作業の手順を述べ，各手順で使用するコマンドについて述べます。また，添付のサンプル・プログラムを用いた実行例についても詳細に説明します。

この章をお読みになることで，ある程度までのディバグ作業は行なえるようになります。

第４章　コマンドの説明

ＩＥ－７８３１０Ａ－Ｒのすべてのコマンドについて，詳細に説明します。最初からこの章を読まれる必要はありません。

コマンドについて詳細に知りたいときにこの章をお読みください。

第５章　応用方法

ＩＥ－７８３１０Ａ－Ｒで，基本的な機能ではないが，知っておくと便利な機能について述べます。

ひと通りのディバグ方法や，コマンドの使用方法を理解したうえで，さらに効率的なディバグを行ないたい場合は，この章をお読みください。

第２章　　機能概要

２．１　概　　要

この章では，ＩＥ－７８３１０Ａ－Ｒをスタンドアロンで使用した場合と，システム・ソフトウエアを使用した場合のそれぞれについて，機能の概要を述べます。

この章を読むことにより，ＩＥ－７８３１０Ａ－Ｒの機能と，それに対応するコマンドをつかむことができます。

２．２では，スタンドアロンで使用した場合とシステム・ソフトウエアを使用した場合の違いについて説明します。

２．３では，スタンドアロンで使用した場合の機能について実際のコマンドと対応して説明します。

２．４では，システム・ソフトウエアを使用した場合拡張される機能について実際のコマンドと対応して説明します。

２．２　スタンド・アロン時とシステム・ソフトウエア使用時の違い

ＩＥ－７８３１０Ａ－Ｒは，スタンド・アロンで使用した場合でも十分な機能をもっています。しかし，システム・ソフトウエアを使用した場合は，スタンド・アロンで使用した場合と比較してより多くの機能をもっています。

ＩＥ－７８３１０Ａ－Ｒは，スタンド・アロンで使用した場合は，以下に示すような機能をもっています。

・イニシャライズ機能
・マッピング機能　（ＭＡＰ　コマンド）
・メモリ操作機能　（ＡＳＭ，ＤＡＳ，ＭＥＭ，ＬＯＤ，ＳＡＶ　コマンド）
・レジスタ操作機能　（ＭＤＲ，ＲＥＧ，ＳＰＲ　コマンド）
・エミュレーション機能　（ＲＵＮ　コマンド）
・ブレーク機能　（ＢＲ？　コマンド）
・トレース機能　（ＴＲ？　コマンド）
・自己診断機能　（ＤＩＧ　コマンド）
・ＰＧＭモード機能　（ＰＧＭ，ＭＯＤ　コマンド）

システム・ソフトウエアを使用した場合は，スタンド・アロンで使用した場合の機能にさらに以下に示される機能が拡張されます。

・ディレクトリ表示機能　（ＤＩＲ　コマンド）
・オート・イニシャライズ機能
・コマンド／データ・ライン編集機能
・コマンド・ヒストリ機能　（ＨＩＳ　コマンド）
・コマンド・ファイル作成機能　（ＣＯＭ　コマンド）
・ファイルからの入力機能　（ＳＴＲ　コマンド）
・ファイルへの結果出力機能　（ＬＳＴ　コマンド）
・コマンド・ヘルプ機能　（ＨＬＰ　コマンド）
・シンボリック・ディバグ　（ＳＹＭ　コマンド）
・コマンド省略形入力

以上のように，ＩＥ－７８３１０Ａ－Ｒは，スタンド・アロンで使用した場合でも十分なディバグ機能をもっています。さらに，システム・ソフトウエアを使用することにより，より効率的にディバグができます。

２．３　スタンド・アロンの機能

ＩＥ－７８３１０Ａ－Ｒをスタンド・アロンで使用した場合の機能を以下に示します。

・イニシャライズ機能
・マッピング機能　（ＭＡＰ　コマンド）
・メモリ操作機能　（ＡＳＭ，ＤＡＳ，ＭＥＭ，ＬＯＤ，ＳＡＶ　コマンド）
・レジスタ操作機能　（ＭＤＲ，ＲＥＧ，ＳＰＲ　コマンド）
・エミュレーション機能　（ＲＵＮ　コマンド）
・ブレーク機能　（ＢＲ？　コマンド）
・トレース機能　（ＴＲ？　コマンド）
・自己診断機能　（ＤＩＧ　コマンド）
・ＰＧＭモード機能　（ＰＧＭ，ＭＯＤ　コマンド）

２．３．１　イニシャライズ機能

（１）オンチップＲＯＭサイズの指定

スタートアップ時に$\overline{\text{ＥＡ}}$端子をセンスし，$\overline{\text{ＥＡ}}$端子が“ハイ・レベル”の場合は，μＰＤ７８３１２Ａ，あるいは，μＰＤ７８Ｐ３１２Ａとしての使い方なので，オンチップＲＯＭのサイズを設定します。オンチップＲＯＭのサイズは，４Ｋ，８Ｋ（μＰＤ７８３１２Ａ），１６Ｋの３種類から選択することができます。

$\overline{\text{ＥＡ}}$端子が“ロウ・レベル”の場合は，μＰＤ７８３１０Ａ（ＲＯＭ　ｌｅｓｓ版）としての使い方なので，自動的にオンチップＲＯＭのサイズが０に設定されます。

（２）拡張メモリの使用／不使用の指定

シンボル情報をもつために，４８Ｋバイトのバッファをもっていますが，大容量のシンボル情報をもつ必要がある場合は，メモリを拡張することができます。

この場合は，本体の拡張スロットに，弊社のＳＢ－０５１２メモリ・ボードを差し込む必要があります。

スタートアップ時に，このメモリ・ボードを用いてメモリを拡張するかどうかを指定することができます。

２．３．２　マッピング機能（ＭＡＰコマンド）

ＩＥ－７８３１０Ａ－Ｒは，μＰＤ７８３１２Ａのオンチップ ＲＯＭ，オンチップＲＡＭ，ＳＦＲを除く，外部拡張メモリ・エリアについて２５６バイトごとに，ユーザ・システム，ＩＥ内エミュレーション・メモリのどちらにもマッピングすることができます。

ただし，ＳＦＲ中で外部アクセスのエリア（0FFB0H～0FFBFH）は，常にユーザ・システムにマッピングされます。

外部拡張メモリ・エリアのマッピング・モードは次の４種類があります。

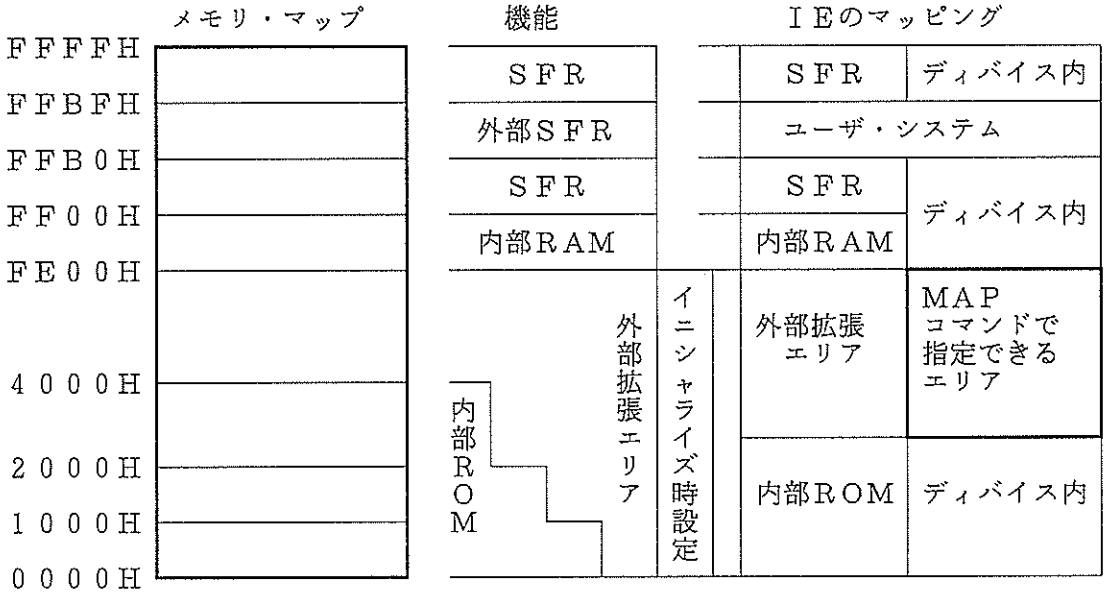
・ユーザ・マッピング（ＭＡＰ　Ｕ）

・ＩＥ内エミュレーション・メモリ・マッピング（ＭＡＰ　Ｗ）

・ＩＥ内エミュレーション・メモリ・マッピング（ライト・プロテクト）（ＭＡＰ　Ｒ）

・ノン・マッピング（ＭＡＰ　Ｋ）

μＰＤ７８３１２Ａから見た場合は，次のようなメモリ・マップとなります。



２．３．３　メモリ操作機能(ＡＳＭ，ＤＡＳ，ＭＥＭ，ＬＯＤ，ＳＡＶ　コマンド)

ＩＥ－７８３１０Ａ－Ｒは，通常の１６進数による変更（ＭＥＭ　Ｃ），表示（ＭＥＭ　Ｄ），サーチ（ＭＥＭ　Ｇ）等の他に，ニモニックによる変更（ＡＳＭ），表示（ＤＡＳ），ヘキサ形式オブジェクトのロード（ＬＯＤ），セーブ（ＳＡＶ）などのメモリ操作機能をもっています。

これらの機能によるメモリへのアクセスは，すべてマッピング状態がチェックされます。

２．３．４　レジスタ操作機能（ＭＤＲ，ＲＥＧ，ＳＰＲ　コマンド）

μＰＤ７８３１２Ａ，μＰＤ７８３１０Ａの内部レジスタを表示，変更することができます。

汎用レジスタ（ジェネラル・レジスタ，および，インプライド・レジスタ）には，ＲＥＧコマンドを使用します。

特に，ＰＳＷについては，１ビットずつの表示，変更をすることができます。

ＳＦＲのうち，モード／コントロール系のレジスタには，ＭＤＲコマンドを使用します。

ＳＦＲのうち，カウンタなどデータ・レジスタには，ＳＰＲコマンドを使用します。

２．３．５　エミュレーション機能（ＲＵＮコマンド）

エミュレーション機能には以下に示すものがあります。

（Ａ）通常リアルタイム・エミュレーション　（ＲＵＮ＿Ｎ）
（Ｂ）ブレーク条件付きリアルタイム・エミュレーション　（ＲＵＮ＿Ｂ）
（Ｃ）指定ステップ数リアルタイム・エミュレーション　（ＲＵＮ＿Ｓ）注
（Ｄ）トレース・エミュレーション　（ＲＵＮ＿Ｔ）

注　ステップ数は，実行命令のステップ数ではなく，命令のフェッチ数でカウントします。μＰＤ７８３１２Ａはプリフェッチを行なうため，実行数とフェッチ数は一致しません。

これ以外に，ワン・ステップ実行も可能です。

以上のエミュレーション機能が動作しているときのブレーク要因には，次のブレーク機能で述べられている，３種類のブレーク機能が適用されます。

ＲＵＮ＿Ｎ　に対しては，フェイルセーフ・ブレークだけがブレーク要因となります。　ＲＵＮ＿Ｂ　に対しては，フェイルセーフ・ブレークとブレーク・レジスタ・ブレークが，ブレーク要因となります。

ＲＵＮ＿Ｓ，ＲＵＮ＿Ｔ　に対しては，フェイルセーフ・ブレークと，コマンド・ブレークが，ブレーク要因となります。

注意：ＩＥ－７８３１０Ａ－Ｒでは，μＰＤ７８３１２Ａのアーキテクチャ上，以下のような使用上の注意が必要です。

- ブランチ系命令のオペランドではブレークしません。
  したがって，ブランチ系命令のオペランドにはブレーク・ポイントを置かないでください。
- ブレーク・ポイントを通過後，数命令実行してから，ブレークします（これをスリップといいます）。
  スリップする命令数は，一定していませんが，ブレーク・ポイント後，１バイト命令が続いているとき最も多くスリップします。
  また，同じ条件でも内部ＲＯＭ実行時と，外部メモリ実行時でも，スリップする命令数が異なります。
  内部ＲＯＭ実行時は，外部メモリ実行時よりも，多くスリップします。
- 命令ステップ数をカウントする場合，ＳＦＲに対する一部の命令は，２ステップと数えられます。詳しくは，第４章　４．７　オンライン・アセンブラ／逆アセンブラ仕様の命令一覧表をご覧ください。
- ＣＡＬＬＴ，ＢＲＫ命令あるいは，割込みによるベクタ参照では正しくブレークしません。したがって，ベクタ・エリアにブレーク・ポイントを置かないでください。
- ベクタ・テーブル領域（０Ｈ～２９Ｈ番地）にジャンプするようなプログラムでは，正しくインストラクション・トレース表示は行えません。したがって，このようなプログラムは書かないようにしてください。

２．３．６　ブレーク機能（ＢＲ？　コマンド）

ブレーク機能には，大きく分けて，以下の３種類があります。

・ブレーク・レジスタ・ブレーク

ブレーク・レジスタを用いて，ユーザが設定できるブレーク機能

・コマンド・ブレーク

エミュレーション・コマンドの中で，ユーザが設定できるブレーク機能

・フェイルセーフ・ブレーク

Ａ）マニュアル・ブレーク

Ｂ）ノン・マップ・ブレーク

Ｃ）ライト・プロテクト・ブレーク

Ｄ）リセット・ブレーク

（１）ブレーク・レジスタ・ブレーク

ブレーク・レジスタには，物理ブレーク・レジスタと，論理ブレーク・レジスタがあります。物理的なブレーク条件（アドレス，データ，外部データ，ステップ数，タイマ等）を物理ブレーク・レジスタに設定したのち，このレジスタの組合わせを論理ブレーク・レジスタに設定します。

このブレーク機能は，ＲＵＮ＿Ｂ時だけ有効になります。

Ａ）物理的なブレーク条件

ＢＲＡ コマンド

- アドレス　：μＰＤ７８３１２Ａのメモリ空間から，ＳＦＲ，レジスタ空間を除く 0～0FE7FHに対して合計５箇所までのアドレス，またはアドレス範囲を指定できます。
- データ　：0～0FFHまでのデータ・パターンを１種類だけ指定できます。
- ステータス　：ＯＰ（オペコード・フェッチ）
  ＲＷ（データ・リード／ライト）
  Ｒ（データ・リード）
  Ｗ（データ・ライト）
  ＲＷＰ（プログラムによるリード／ライト）
  ＲＰ（プログラムによるリード）
  ＷＰ（プログラムによるライト）
  ＲＷＭ（マクロ・サービスによるリード／ライト　）
  ＲＭ（マクロ・サービスによるリード）
  ＷＭ（マクロ・サービスによるライト）
  ＮＣ（オペコード・フェッチを含むすべてのリード／ライト）
  以上の１１種類のうち，１種類だけを指定できます。
- ループ回数　：１～２５５回のループ回数を指定できます。

ＢＲＤ
コマンド

外部データ　：ＩＥ－７８３１０Ａ－Ｒは，ターゲット・プローブの他に，外部信号センス用に８本のプローブをもっており，ＴＴＬレベルの信号をセンスすることができます。
この８本のプローブのデータ・パターンの一つをブレーク要因として指定することができます。

ＢＲＥ
コマンド

ステップ数　：１～65535 までのステップ数を指定できます。
μＰＤ７８３１２Ａが，指定ステップ数だけ命令をフェッチすることが，ブレーク条件となります。

ＢＲＴ
コマンド

タイマ　：10μsec ～655msec までの時間を指定できます。
μＰＤ７８３１２Ａが指定された時間だけユーザ・プログラムを実行することが，ブレーク条件となります。

Ｂ）物理ブレーク・レジスタ

物理ブレーク・レジスタには，ＢＲＡ，ＢＲＤ，ＢＲＥ，ＢＲＴの４種類があります。

それぞれが，物理ブレーク条件に対応します。

Ｃ）論理ブレーク・レジスタ

論理ブレーク・レジスタには，ＢＲ０，ＢＲ１，ＢＲ２，ＢＲ３，ＢＲＭの５種類があります。ＢＲＭが最終的なブレーク要因を決定します。

ＢＲ０，ＢＲ１，ＢＲ２，ＢＲ３は物理ブレーク・レジスタを組合わせて，ブレーク条件をつくることができます。

ＢＲ０～ＢＲ３で，一つの物理ブレーク・レジスタを指定した場合は，指定された物理ブレーク・レジスタのブレーク条件がＢＲ０～ＢＲ３のブレーク条件となります。二つ以上の物理ブレーク・レジスタを指定した場合は，その物理ブレーク・レジスタのブレーク条件の論理和がＢＲ０～ＢＲ３のブレーク条件となります。

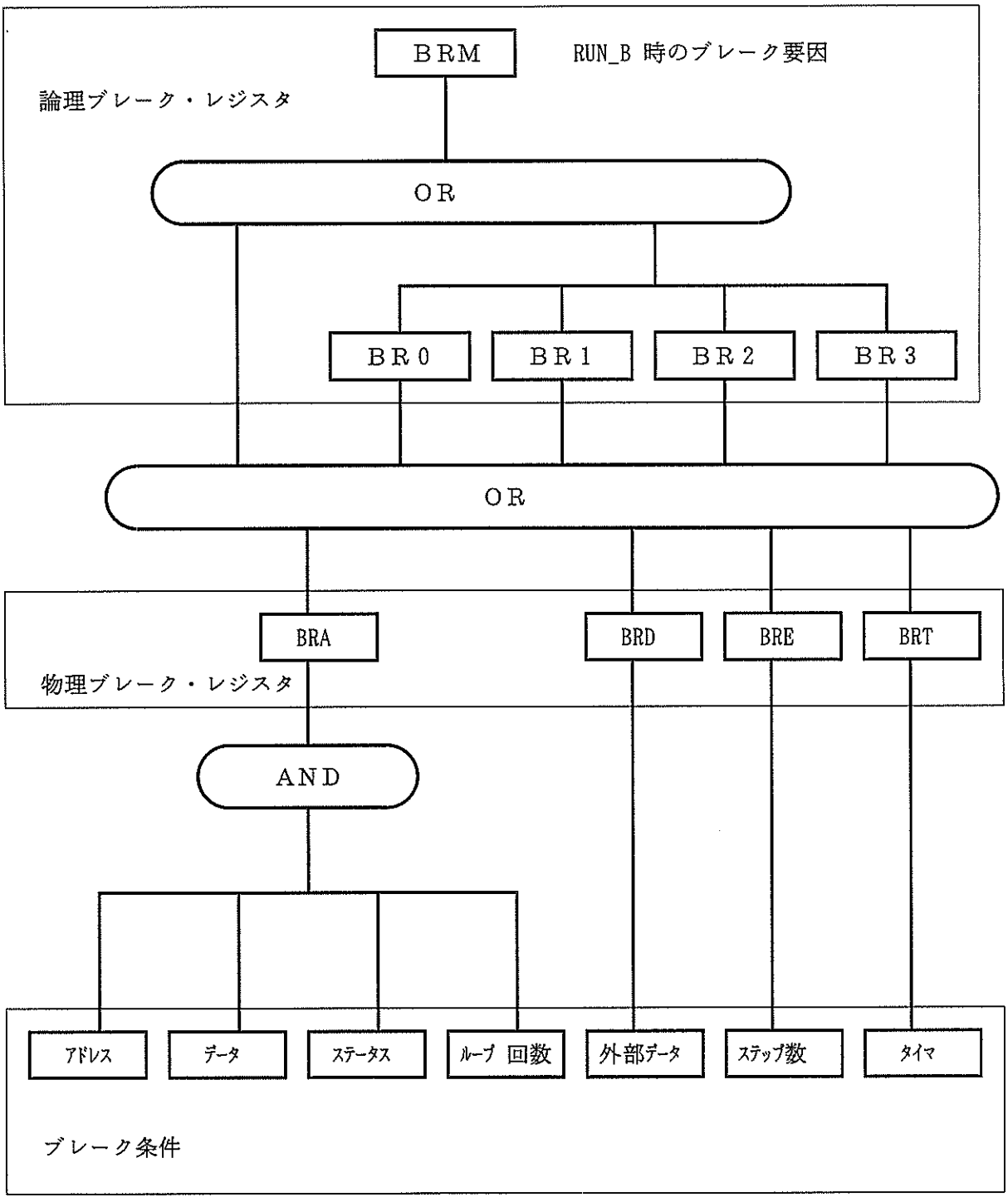
ＢＲＭは，ＢＲ０，ＢＲ１，ＢＲ２，ＢＲ３および，ＢＲＡ，ＢＲＤ，ＢＲＥ，ＢＲＴのすべてのブレーク・レジスタを組合わせて，ブレーク条件をつくることができます。

（２）コマンド・ブレーク

エミュレーション・コマンド（ＲＵＮ＿Ｓ，ＲＵＮ＿Ｔ）の中でエミュレーション時のブレーク条件を設定することができます。

ブレーク・レジスタ・ブレークとコマンド・ブレークを図２－１のように組合わせることによりブレーク条件が指定されます。

図２－１　ブレーク・レジスタ構成



（３）フェイルセーフ・ブレーク

フェイルセーフ・ブレークは，すべてのエミュレーション・コマンド（ＲＵＮ＿Ｎ，ＲＵＮ＿Ｂ，ＲＵＮ＿Ｓ，ＲＵＮ＿Ｔ）に対し有効となります。ユーザが禁止することはできません。フェイルセーフ・ブレークには，次の３種類があります。

Ａ）マニュアル・ブレーク

コンソールからＥＳＣキーが入力された場合，強制ブレークします。

Ｂ）ノン・マップ・ブレーク

マッピングされてないメモリ・エリアに対してアクセスを行なった場合，強制ブレークします。

Ｃ）ライト・プロテクト・ブレーク

内部ＲＯＭエリアあるいは，ライト・プロテクト付きＩＥ内エミュレーション・メモリ・マッピング・エリア，あるいはリード・オンリのＳＦＲに対して，ライト・アクセスを行なった場合，強制ブレークします。

Ｄ）リセット・ブレーク

ＥＶＡＣＨＩＰに対して，リセットがかかった場合，強制ブレークします。

２．３．７　トレース機能（ＴＲ？　コマンド）

（１）トレース内容（ＴＲＤ　コマンド）

アドレス・バス	：	１６ビット
データ　・バス	：	８ビット
ポート　注１	：	８ビット
外部プローブ	：	８ビット
フレーム・ステータス　注２	：	１０種類

を１フレームごとに，２０４７フレームまでトレースすることができます。

注１　ポートは，Ｐ０からＰ５までの六つのポートのうち，一つをＴＲＳコマンドで選択することができます。

注２　フレーム・ステータスは，そのフレームがどのような意味をもつかを示すステータスです。

実際には，４ビットのエンコードされたステータスを用いて，次の１０種類のフレーム・ステータスを示しています。

ＲＤ	プログラムによるリード
ＷＲ	プログラムによるライト
ＭＳＲＤ	マクロ・サービスによるリード
ＭＳＷＲ	マクロ・サービスによるライト
Ｍ１ 注３	最初のオペコードのフェッチ
ＯＰ 注４	オペランドあるいは２番目のオペコードのフェッチ
ＢＲＭ１ 注４	ＢＲステータス直後のＭ１
ＢＲＯＰ 注４	ＢＲステータス直後のＯＰ（ベクタ参照時）
（Ｍ１）	ＢＲにより，Ｉ－Ｂｕｆｆｅｒからフラッシュされた無効フレームであることを示します。
（ＯＰ）	

注３　ＳＦＲに対する一部の命令をフェッチしたときは，Ｍ１フレームが２回出力されます。

詳しくは，第４章　４．７　オンライン・アセンブラ／逆アセンブラ仕様の命令一覧表をご覧ください

注４　ＣＡＬＬＴ，ＢＲＫなどの命令あるいは割込みによるベクタ参照はオペコード・フェッチされます。したがって，ベクタ参照のフレーム・ステータスは，ＯＰあるいは，ＢＲＯＰとなります。

（２）トレース条件　（ＴＲＭ，ＴＲＸ，ＴＲＱ，ＴＲＳ　コマンド）

トレース条件には，次の３種類があります。

- 全トレース
- 部分トレース
- トレースなし

ＴＲＭコマンドで設定します。

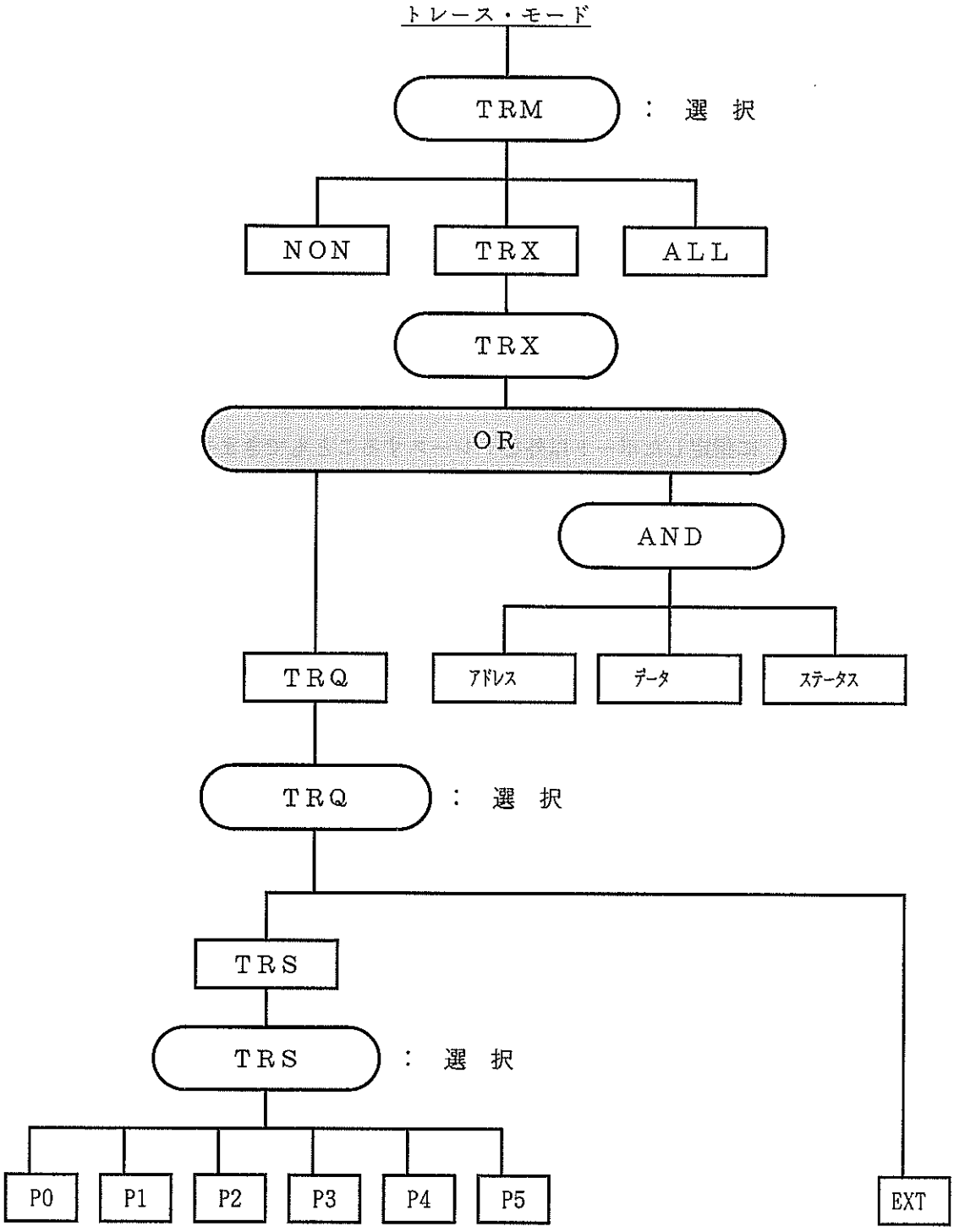
部分トレース時は，次の４種類の組合わせにより，クオリファイの条件を設定することができます。

・アドレス　：　ブレーク条件のアドレスと同じ。
・データ　　：　ブレーク条件のデータと同じ。
・ステータス：　ブレーク条件のステータスと同じ。
・ポート・データ・パターン　：
　Ｐ０，Ｐ１，　・・・　Ｐ５および外部データの７種類のポートのうち一つを選択し，そのポートのデータ・パターンを１種類だけ指定します。

ＴＲＳ，ＴＲＸ，ＴＲＱ コマンドで設定

注意　ＴＲＳコマンドは，トレース内容の設定と，クオリファイ条件の設定とを兼ねています。したがって，トレース内容に指定したポート以外でクオリファイ条件を設定することはできません。また，クオリファイ条件に指定したポート以外をトレースすることもできません。

図２－２　トレース・モード設定の構成



（３）トレース表示（ＴＲＤ　コマンド）

トレース表示には次の３種類があります。

・フレーム・モード表示（全トレース時，部分トレース時可能）（ＴＲＤ　Ｆ）・

インストラクション・モード表示（全トレース時可能，部分トレース時不可）（ＴＲＤ　Ｉ）

・マクロ・サービス・モード表示（全トレース時可能，部分トレース時不可）（ＴＲＤ　Ｍ）

フレーム・モード表示は，トレースしたフレームをすべてトレースした順番に表示するモードです。μＰＤ７８３１２Ａが実際に動作したとおりの情報を表示します。部分トレースした場合でも，全トレースした場合でもフレーム・モード表示は可能です。

ただし，μＰＤ７８３１２Ａはプリフェッチを行なっているため，このモードではプログラムの実行の流れを追うことは困難です。

したがって，このモードは，主として実際のμＰＤ７８３１２Ａの動作そのものをディバグする目的に使用されます。

一方，インストラクション・モード，および，マクロ・サービス・モードでは，トレースした情報のうち，フェッチ・フレームをプログラム実行フレームに変換し，さらに逆アセンブル・リストを付加して，プログラム実行の流れを追い易いようにしてあります。

ただし，このモードではプリフェッチを含んだ実際のターゲットＣＰＵの動作そのものを表示することはできません。

また，フェッチ・フレームをプログラム実行フレームに変換するためには，μＰＤ７８３１２Ａのすべての実行フレームをトレースしておく必要があります。このため，部分トレースした場合は，インストラクション・モード，および，マクロ・サービス・モード表示はできません。

インストラクション・モードでは，プログラムの流れをできるかぎり見易くするため，マクロ・サービスによるリード／ライト・フレームは表示しません。

マクロ・サービス・モードでは，インストラクション・モードの表示に加えてマクロ・サービスによるリード／ライト・フレームと，ポート・データ，外部プローブ・データが表示されます。

２．３．８　自己診断機能（ＤＩＧコマンド）

ＩＥ－７８３１０Ａ－Ｒは，自分自身を診断する機能，および，ハードウエアを備えています。診断項目を以下に示します。

・オルタネートＲＡＭ

・ユーザＲＡＭ

・６４ｐｉｎ　プローブ・テスト

・ＥＶＡＣＨＩＰテスト

２．３．９　ＰＧＭモード機能（ＰＧＭ，ＭＯＤコマンド）

ＰＧＭコマンドによりＰＧＭモードとなり，シリアル・チャネル２にPG-1500，PG-2000など，ＮＥＣ製ＰＲＯＭライタを接続して，オブジェクトをアップ／ダウン・ロードすることができます。

２．４　システム・ソフトウエアを使用した場合の機能拡張

システム・ソフトウエアを使用することによりスタンドアロン時のＩＥ－７８３１０Ａ－Ｒと比較して，以下の機能が拡張されます。

・ディレクトリ表示機能　（ＤＩＲ　コマンド）
・オート・イニシャライズ機能
・コマンド／データ・ライン編集機能
・コマンド・ヒストリ機能　（ＨＩＳ　コマンド）
・コマンド・ファイル作成機能　（ＣＯＭ　コマンド）
・ファイルからのコマンド入力　（ＳＴＲ　コマンド）
・ファイルへの結果出力　（ＬＳＴ　コマンド）
・シンボリック・ディバグ　（ＳＹＭ　コマンド）
・コマンド・ヘルプ機能　（ＨＬＰ　コマンド）

また，一部のコマンドに対して，

・コマンド省略形入力

が，可能となります。

２．４．１　ディレクトリ表示機能（ＤＩＲコマンド）

ＣＰ／Ｍ™のビルトイン・コマンドのＤＩＲと同等の機能をもち，指定したドライブのディレクトリを参照することができます。

２．４．２　オート・イニシャライズ機能

システム・ソフトウエア起動時の，セット・アップの内容をファイルに記憶します。次からは，ファイルの内容を自動的に設定します。

ＣＰ／Ｍ™は，米国ディジタル・リサーチ社の商標です。

２．４．３　コマンド／データ・ライン編集機能

ＭＤ－０８６／０８０（ＣＲＴ内蔵タイプ）の本体コンソールあるいは，ＭＤ－１１６／０８６／０８０－１０で，コンソールにＭＤ－９１０ＴＭを使用している場合は，コマンド，あるいは，データのキー入力時，カーソル移動，挿入，置換などの編集をすることができます。

注意　ＰＤＡ－８８０，およびＣＲＴ内蔵タイプのＭＤ－０８６での外部コンソールでは，編集機能は使用できません。

注意　ＭＤ－１１６／０８６シリーズのＣＣＰ／Ｍで，マルチウインドウ・モードでは，編集機能は使用できません。

２．４．４　コマンド・ヒストリ機能（ＨＩＳコマンド）

最新のコマンド行を２０行分記憶し，これを表示することができます。

また，コマンド入力時の最初に，“！ｎ⤶”（ｎは行番号）を入力することにより，すでに記憶しているコマンド行を呼出し、さらに“⤶”を入力することにより実行することができます。特に最新のコマンド行は，“！！⤶”と入力するだけでよく，行番号を入力する必要はありません。

２．４．５　コマンド・ファイル作成機能（ＣＯＭコマンド）

キー入力したコマンド・ラインやデータ・ラインをテキスト・ファイルに記憶することができます。このため，このテキスト・ファイルをコマンド・ファイルとしてＳＴＲコマンドで使用することができます。

テキスト・ファイルへの出力のタイミング‘Ｃｔｒｌ－Ｏ’で制御できます。

ただし，仮パラメータを指定することはできません。

２．４．６　ファイルからのコマンド入力（ＳＴＲコマンド）

ファイルからのコマンド入力機能により，キーボードから入力できるコマンド，あるいは，データなどの行単位の入力をＣＰ／Ｍベースのテキスト・ファイルから読込み，自動的にコマンドを実行することができます。

ファイル作成には，ＣＰ／Ｍベースのエディタ等を使用することができます。

また，コマンド・ファイル作成機能（ＣＯＭコマンド）を使用して作成することもできます。また，ファイル中に仮パラメータを指定することもできます。

仮パラメータは，‘$0’～‘$3’の四つを指定することができます。

仮パラメータを使用するファイルは，ＣＰ／Ｍのエディタなどで作成してください。

２．４．７　ファイルへの結果出力（ＬＳＴコマンド）

コマンド実行結果をコンソールとともに，指定された論理コンソールに対して出力できます。論理コンソールには，リスト装置，および，ファイルを指定できます。

ファイル，または，リスト装置には，コンソールに表示された文字がすべて出力されます。出力のタイミングは，‘ＣＴＲＬ－Ｒ’で制御することができます。　★

２．４．８　シンボリック・ディバグ（ＳＹＭコマンド）

コマンド・ライン入力，あるいは，データ入力の数値表現の代わりにシンボルを記述することができます。シンボルの記述は，すべてのコマンドの，すべての数値表現の代わりに記述することができます。

シンボルを記述する場合には，あらかじめ，シンボル・テーブル・ファイルをロードをしておく必要があります。

また，シンボルを追加（ＳＹＭ　Ａ）したり，追加したシンボルを変更（ＳＹＭ　Ｃ）したり，削除（ＳＹＭ　Ｅ，ＳＹＭ　Ｋ）することができます。さらに，シンボルを表示（ＳＹＭ　Ｄ）することもできます。

２．４．９　コマンド省略形

システム・ソフトウエアでは，以下に示すコマンドについては，先頭の１文字のみをコマンドとするコマンド省略形の入力も許されます。

ＡＳＭ（Ａ）：　アセンブラ・コマンド

ＣＬＫ（Ｃ）：　クロック・コマンド

ＤＡＳ（Ｄ）：　逆アセンブラ・コマンド

ＥＸＴ（Ｅ）：　システム・モード終了コマンド

ＨＬＰ（Ｈ）：　ヘルプ・コマンド
ＬＯＤ（Ｌ）：　オブジェクト／シンボル・ロード・コマンド
ＭＥＭ（Ｍ）：　メモリ・コマンド
ＲＵＮ（Ｒ）：　エミュレーション・コマンド
ＳＡＶ（Ｓ）：　オブジェクト・セーブ・コマンド
ＴＲＤ（Ｔ）：　トレース表示コマンド
ＶＲＹ（Ｖ）：　ベリファイ・コマンド

（　）内がコマンドの省略形

２．４．１０　コマンド・ヘルプ機能（ＨＬＰコマンド）

システム・ソフトウエアでは，ＩＥ－７８３１０Ａ－Ｒで使用できるコマンドの使用方法を表示することができます。

第３章　基本的な使用方法

３．１　概　　要

この章では，ＩＥ－７８３１０Ａ－Ｒでディバグをするための手順や基本的なコマンドの使用方法について説明します。

ＩＥ－７８３１０Ａ－Ｒを使用されるときは，必ず本章をお読みください。

３．２では，ＩＥ－７８３１０Ａ－Ｒでディバグをするときの操作項目と，その手順について説明します。さらに，各操作項目について，ＩＥ－７８３１０Ａ－Ｒのどのコマンドを使用したらよいか，具体的に説明します。

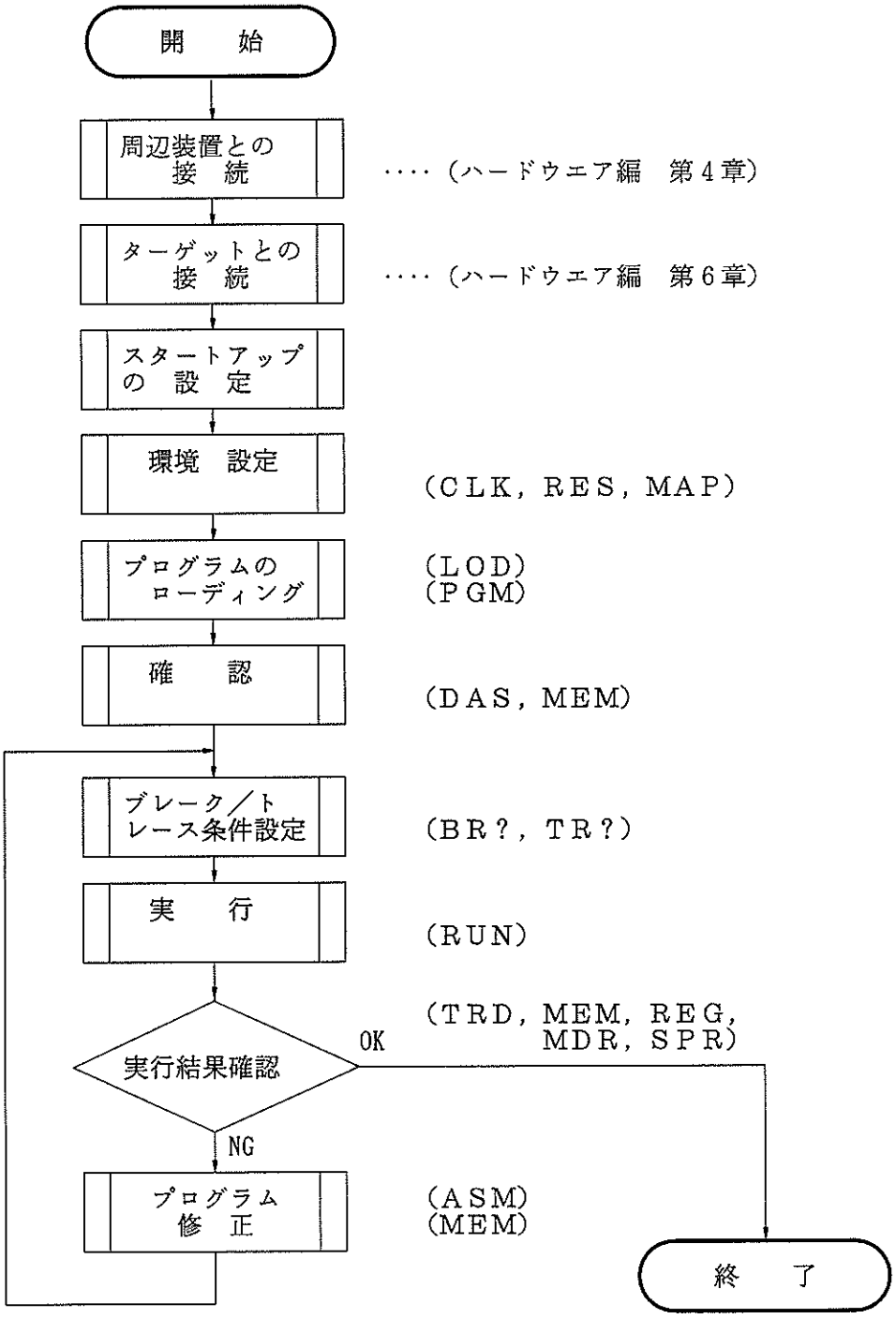
３．３では，ＩＥ－７８３１０Ａ－Ｒに添付されているフロッピィ・ディスクを，ホスト・マシンにセットし，サンプル・プログラムを用いて，実際に操作した例を示します。

本章を読むことで，第４章　コマンドの説明を読まなくても，ある程度までの操作はできるようになります。

３．２　ディバグ手順とそれに対するコマンド

一般にＩＥを用いたディバグは，図３－１のような手順で行ないます。

図３－１　ディバグ手順



３．２．１　ターゲットとの接続

ターゲット・システムのハードウエアの開発状況により，ソフトウエア単体の論理ディバグと，ハードウエアまでを含めた総合ディバグの２種類のディバグ方法が考えられます。

それぞれのディバグでターゲット・プローブの接続方法が異なります。

ハードウエア編の第６章ターゲットとの接続方法にしたがって，接続してください。

３．２．２　周辺装置との接続

ホスト・マシン，あるいは，ターミナルとＩＥ－７８３１０Ａ－Ｒを接続します。

ホスト・マシンを接続すると，ホスト・マシン上でシステム・ソフトウエアを動作させることにより，ＩＥ－７８３１０Ａ－Ｒの最大機能を使うことができます。

ターミナルと接続すると，ＩＥ－７８３１０Ａ－Ｒは，スタンドアロン動作をします。この場合，ある程度機能は制限されます。

ハードウエア編の第４章システムの構成方法にしたがって接続してください。

３．２．３　スタートアップの設定

システム・ソフト使用時

ＩＥ－７８３１０Ａ－Ｒとホスト・マシンを接続した場合，まずシステム・ソフトウエアを起動します。すると，最初に，コマンド・ライン編集機能を使うかどうか聞いてきますので特にマルチウインドウにしておく必要がない場合は“Ｙ”を，マルチウインドウを使用される場合は“Ｎ”をキー入力します。次にポート番号を聞いてきますので，ＩＥ－７８３１０Ａ－Ｒが接続されているポート番号をキー入力します。

> マルチウインドウ・モードでは，編集機能を使用できません。このため，編集機能を使用するために，マルチウインドウ・モードを解除するか，現在のウインドウ・モードのままで，編集機能を削除するかを聞いてきます。

次にターゲット・システムの電源を入れるようにメッセージが出てきますので，ターゲット・システムの電源を入れてから“Ｙ”と入力し，リターンします。次にセット・アップ・モードをアップデートするかどうかを聞いてきますので，前回のセット・アップ・モードでよければ，“Ｎ”を，あらたにセット・アップを行なうときは“Ｙ”を入力し，リターンしてください。“Ｙ”を入力した場合，スタンドアロン時と同様に，セット・アップを行なってください。

スタンドアロン時

モニタがスタートすると，最初にターゲット・システムの電源を入れるようにメッセージが出てきますので，ターゲット・システムの電源を入れてから“Ｙ”と入力し，リターンします。次に，内部ＲＯＭ容量を聞いてきますので，μＰＤ７８３１２Ａのディバグを行なうときは，“８Ｋ”と入力し，リターンします。

次に拡張メモリを使用するかどうかを聞いてきますので，スタンドアロン・モードの場合“Ｎ”を，システム・モードでも，通常は“Ｎ”を入力します。

特に大容量のシンボルをローディングする場合（２０００以上），ＩＥ－７８３１０Ａ－Ｒの空きスロットに弊社のＳＢ－０５１２を差込むことにより，大容量のシンボルをローディングすることができます。このような場合，“Ｙ”を入力します。

システム・モードのスタート・アップの表示を図３－２に，スタンドアロン・モードのスタート・アップの表示を図３－３に示します。

図３－２　システム・モードのスタート・アップ

注　アンダーラインはキーボードからの入力を表しています（以降，すべて同じです）。

```
A>IE78310 ↵
16:14:42 A:IE78310.CMD

  IE-78310 CONTROLLER    (MD-086/116 SERIES) Vx.x  [Dd Mmm Yy]
       Copyright  (C) 1985 by NEC corporation

  Do you want to use COMMAND LINE EDITOR (Y or N) : Y ↵
  Window off !
  Select port NO. (1 to 4) : 2 ↵
  IE-78310A Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1985 by NEC corporation

  Power on target system (Y/N) Y ↵
  Create new set up mode (Y or N) : Y ↵
  Internal ROM size (4K,8K,16K) = 8K ↵
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78310A? (Y/N)= Y ↵
     Symbol save area:  00000H-0BFFFH
                        10000H-1BFFFH
                        20000H-2BFFFH
                        30000H-3BFFFH
                        40000H-4BFFFH
                        50000H-5BFFFH
                        60000H-6BFFFH
                        70000H-7BFFFH
2>
```

図３－３　スタンドアロン・モードのスタート・アップ

```
  IE-78310A Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1985 by NEC Corporation

  Power on target system (Y/N) Y ↵
  Internal ROM size (4K,8K,16K) = 8K ↵
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78310A ? (Y/N)= N ↵
 *
```

３．２．４　環境設定（ＣＬＫ，ＲＥＳ，ＭＡＰ）

スタート・アップの設定が終わりますと，コマンド入力待ちになります。最初に，ＣＬＫコマンドでクロック・ソースの指定をしてください。スタート・アップ時は，内部（１２ＭＨｚ固定）となっています。内部のまま使用される場合は，特にクロック・ソースの指定をする必要はありません。

次にＲＥＳコマンドで，エバリュエーション・チップをリセットしてください。

最後にＭＡＰコマンドで外部メモリのマッピングを指定してください。

これらの設定は，ある程度固定されている処理なので，テキスト・ファイル等に登録し，ＳＴＲコマンドで自動設定すると便利です（ソフトウエア編の第５章　応用方法をお読みください）。

３．２．５　プログラム（シンボル）のローディング（ＬＯＤ，ＰＧＭ）

システム・ソフト使用時

システム・ソフト使用時は，ＬＯＤコマンドを用いて，弊社のリロケータブル・パッケージが出力するヘキサ形式オブジェクト・ファイルとシンボル・テーブル・ファイルをローディングすることができます。

スタンドアロン時

スタンドアロン時は，オブジェクトをＲＯＭに書込み，ＩＥ－７８３１０Ａ－Ｒと，弊社のＰＲＯＭプログラマ（ＰＧ－２０００，ＰＧ－１５００等）を接続して，ＰＲＯＭプログラマからオブジェクトをローディングすることができます。

このときはＰＧＭコマンドを用います。

スタンドアロン・モードでは，シンボルをローディングすることができません。

３．２．６　確　認（ＤＡＳ，ＭＥＭ　Ｄ）

ローディングしたプログラムを確認します。

この場合，ＤＡＳコマンドを用いて，逆アセンブル・リストをとります。

また，固定データの確認にはＭＥＭ　Ｄを用いてください。

３．２．７　ブレーク／トレース条件設定（ＢＲ？，ＴＲ？）

ディバグの初期段階ではプログラムを最初から最後まで実行するよりも，ポイントごとにブレーク・ポイントを設け，そこまでの実行結果を確認しながら，ステップ・

バイ・ステップでディバグする方が有効です。

このような，ブレーク・ポイントを設定するために，ＢＲＡ，ＢＲＤ，ＢＲＴ，ＢＲＭ，ＢＲ０～ＢＲ３の各コマンドを用います。

また，ブレーク・ポイントに至るまでの実行結果をリアルタイム・トレーサでトレースしておくことができます。

リアルタイム・トレーサのモード設定については，ＴＲＭ，ＴＲＸ，ＴＲＱ，ＴＲＳの各コマンドを用います。

３．２．８　実　行（ＲＵＮ）

エバリュエーション・チップで，ローディングしたオブジェクト・プログラムを実行します。

この場合，ＲＵＮコマンドを用います。

ブレーク・ポイントつきのリアルタイム実行や，トレース実行など，４種類の実行方法があります。

目的に応じた実行コマンドを入力してください。

３．２．９　実行結果確認（ＴＲＤ，ＭＥＭ　Ｄ，ＲＥＧ，ＭＤＲ，ＳＰＲ）

オブジェクト・プログラムを実行し，ブレーク・ポイントでブレークしたら，実行結果を確認します。

プログラムのフェッチ，データのリード／ライト・アクセス等の動作はリアルタイム・トレーサに格納されています。これを見るためにはＴＲＤコマンドを用います。

実行した結果，メモリの内容を見るためにはＭＥＭ　Ｄコマンドを用います。

レジスタの内容を見るためにはＲＥＧコマンドを，ＳＦＲの内容を見るためにはＭＤＲ，ＳＰＲコマンドをそれぞれ用います。

３．２．１０　プログラム修正（ＡＳＭ）

実行結果の確認をして，予想どおりの動作をしてない場合，ブレーク・ポイントまでの間にバグがあったことになります。バグを修正し，再実行し，確認しなければなりません。大規模な修正ならば，ソース・レベルから修正，再アセンブルし，ＩＥに再ロードして，確認することが望ましいのですが，小規模な修正ならば，ＡＳＭコマンドを用いて，ニモニック・レベルでのプログラム・パッチを行なうことができます。

こうして，パッチしたプログラムを再びディバグするわけです。

３．３　操作例

この操作例では，ホスト・マシンにＭＤ－０８６ＦＤ－１０を用いて，シリアル・チャネルのチャネル１にＩＥ－７８３１０Ａ－Ｒを接続しています。

また，コンカレントＣＰ／Ｍ™をスタートした後，ＩＥ－７８３１０Ａ－Ｒ添付のフロッピィ・ディスク[注]をドライブＡにセットし，システム・ソフトウエアをスタートしています。

したがって，ホスト・マシンからこの操作例にしたがって，実際にコマンドをキー入力していけば，操作例と同様な結果を得ることができます。

一通りの使用方法を理解するためには，説明と同一のことを実行したり，自分なりにコマンドのオペランドを変更して，実行するのもよいと思います。

また，操作例で用いたコマンド行やデータ行については，ＳＯＲＴ．ＳＴＲというテキスト・ファイルに登録してあります。

コマンド入力待ちになったときに，

1>STR SORT ）

とキー入力すると操作例とまったく同じことを，自動的に実行します。

ただし，コマンドの中断時のＥＳＣ等に関しては，キー入力してください。

注　ＩＥ－７８３１０Ａ－Ｒ添付のフロッピィ・ディスクには，次のファイルが入っています。

・システム・ソフトウエア・パッケージ

ＩＥ７８３１０．ＣＭＤ
（ＭＤ－０８６／１１６シリーズ用システム・ソフトウエア本体）

ＩＥ７８３１０．ＣＯＭ
（ＭＤ－０８０シリーズ用システム・ソフトウエア本体）

ＩＥ７８３１０．ＯＶ１　（省略コマンド・テーブル・ファイル）

ＩＥ７８３１０．ＯＶ２　（ヘルプ・コマンド・テーブル・ファイル）

ＩＥ７８３１０．ＨＬＰ　（ヘルプ・テキスト・ファイル）

・サンプル・ファイル

ＳＯＲＴ．ＨＥＸ　（サンプル・プログラム・オブジェクト・ファイル）

ＳＯＲＴ．ＳＹＭ
（サンプル・プログラム・シンボル・テーブル・ファイル）

ＳＯＲＴ．ＳＴＲ　（サンプル・プログラム・コマンド・ファイル）

(1) IE78310 のｼｽﾃﾑ･ｿﾌﾄ使用時の起動方法を説明します。

```
A>IE78310 ↵                カﾚﾝﾄ･ﾃﾞｨｽｸ上にある，IE78310 のｼｽﾃﾑ･ﾌｧｲﾙ を実行させます
XX:XX:XX A:IE78310.CMD

  IE-78310 CONTROLLER    (MD-086/116  SERIES) Vx.x  [Dd Mmm Yy]
       Copyright  (C) 1985 by NEC Corporation

  Do you want to use COMMAND LINE EDITOR  (Y or N) : Y ↵
  Window off !
  Select port NO. (1 to 4) :  ■ (ｶｰｿﾙ位置) ←ﾎﾟｰﾄNo.選択ﾒｯｾｰｼﾞ です。接続されてい
                                                              注1
                                                るﾎﾟｰﾄ の番号を入力してください。
                                  1 ↵   と入力  ←ﾎﾟｰﾄNo.1 を選択しました。
        IE からのﾒｯｾｰｼﾞ が次の様に表示されます。
  IE-78310A Monitor Vx.x  [Dd Mmm Yy]
  Copyright (C) 1985 by NEC Corporation
  Power on target system (Y/N)  ■ (ｶｰｿﾙ位置) ←ﾀｰｹﾞｯﾄ･ｼｽﾃﾑの起動ﾒｯｾｰｼﾞ です。ﾀｰｹﾞｯ
                                                ﾄ･ｼｽﾃﾑを起動してから"Y" を入力し
                                                てください。
                                  Y ↵
  Create new set up mode (Y or N) : ■ (ｶｰｿﾙ位置) ←新たにｾｯﾄ･ｱｯﾌﾟ をするか，しな
                                                いかのﾒｯｾｰｼﾞ です。
                                    Y ↵          ←新たにｾｯﾄ･ｱｯﾌﾟ をするので，"Y"
                                                を入力しました。
  Internal ROM size(4K,8K,16K)= ■ (ｶｰｿﾙ位置) ←内部ROMｻｲｽﾞ選択ﾒｯｾｰｼﾞ を出力し，入
                                                力待ちになります。
                                 8K ↵          ←内部ROMｻｲｽﾞ8Kを選択しました。
  Tracer  initialize                           ←ﾄﾚｰｽ･ﾒﾓﾘの初期設定ﾒｯｾｰｼﾞ
  Breaker initialize                           ←ﾌﾞﾚｰｸ条件等の初期設定ﾒｯｾｰｼﾞ
  Do you have Memory Board on IE-78310A? (Y/N)= ■     ←IEEE 796ﾊﾞｽ上にﾒﾓﾘ･ﾎﾞｰﾄﾞ が
                                (ｶｰｿﾙ位置) ┘          存在するか聞いてきます。
                                    N ↵              ﾒﾓﾘ･ﾎﾞｰﾄﾞ がないので "N"を
                                                    入力しました。
┌── ﾎﾟｰﾄNo.
1>■ (ｶｰｿﾙ位置) ←IE78310 が起動し，ｼｽﾃﾑ･ﾓｰﾄﾞを表すﾌﾟﾛﾝﾌﾟﾄ を表示します。この状態で
                ｺﾏﾝﾄﾞ入力が可能となります。
```

注1　ﾎﾟｰﾄ･ﾅﾝﾊﾞ選択における，異常ﾊﾟﾀｰﾝを以下に示します。

①指定ﾎﾟｰﾄ･ﾅﾝﾊﾞ 以外を入力した場合

```
  Select port NO. (1 to 4) : 5 ↵          ←ﾎﾟｰﾄNo.5 を選択した場合､ﾒｯｾｰｼﾞが再
                                           度表示され，入力待ちになります。
  Select port NO. (1 to 4) :  ■ (ｶｰｿﾙ位置) ←ﾎﾟｰﾄﾅﾝﾊﾞは，MD-086/116の場合は 1
                                           to，4 MD-080の場合は1 or 2です。
```

②接続状態不良の場合

```
  Select port NO. (1 to 4) : 1 ↵          ←ﾎﾟｰﾄNo.1 を選択したが，接続されて
  No connect !                             いない場合次のﾒｯｾｰｼﾞ を表示し，入
                                           力待ちとなります。
   Abort (Y or N) : ■ (ｶｰｿﾙ位置) ←IE-78310をｱﾎﾞｰﾄするか聞いてきます。
                    N ↵          "N" を入力すると，再度ﾎﾟｰﾄ･ﾅﾝﾊﾞ 選択ﾒｯｾｰｼﾞ が
                                 表示されます。
  Select port NO. (1 to 4) :  ■ (ｶｰｿﾙ位置) ←接続状態を確認し，再度ﾎﾟｰﾄ･ﾅﾝﾊﾞ を
                                           入力してください。
```

③その他のエラー

```
  指定したﾎﾟｰﾄ がｺﾝｿｰﾙ だった場合   ・・・・・・Selected port is used as CONSOLE
  指定したﾎﾟｰﾄ が他のﾀｽｸ で使用されていた場合・Selected port is used by other
                                               process
```

上記のエラー・メッセージが表示され，再度ポート設定にもどります。

(2) スタンドアロン 時での起動方法は次のようになります。

・ターミナルとIEを接続し，リセット・キーを押すと下記のようなメッセージを出力します。

```
IE-78310A Monitor Vx.x  [Dd Mmm Yy]
Copyright (C) 1985 by NEC Corporation
Power on target system (Y/N)  ■ (カーソル位置) ←ターゲット・システムの起動メッセージ です。ターゲット・システムを起動してから“Y” を入力してください。
                              Y )
Internal ROM size(4K,8K,16K)= ■ (カーソル位置) ←内部ROMサイズ選択メッセージ を出力し，入力待ちとなります。
                              8K)             ←内部ROMサイズ8Kを選択しました。
Tracer  initialize                            ←トレース・メモリの初期設定メッセージ
Breaker initialize                            ←ブレーク条件等の初期設定メッセージ
Do you have Memory Borad on IE-78310A ? (Y/N)=■←マルチバス 上にメモリ・ボード が存在するか聞いてきます。
                                                N )  スタンドアロン 動作なので “N”を入力しました。
```

* ■ (カーソル位置) ←スタンドアロン・モードを表すプロンプトを表示します。この状態でコマンド入力が可能となります。

・このあとの説明は，システム・モードでの使用例ですが，プロンプトの表示キャラクタ以外は，‘LOD’,‘SAV’,‘VRY’コマンド以外は，同一の方法で入力できます。

IE-78310A-Rが起動されたら，まずクロック・ソースの選択をします。

```
1>CLK I )        IE内のクロックを使用します。 12MHz固定です。
```

次にエバリュエーション・チップ をリセットします。

```
1>RES )
1>
```

次に，メモリ の使用目的によりマッピングします。
自由にマッピング できる範囲は，外部拡張空間だけです。
　　└── μPD78310A→ 0～FDFFH
　　└── μPD78312A→ 2000H～FDFFH

```
1>MAP )   ←マッピング 内容を表示させるコマンドです。
  0000-1FFF R/O   2000-FDFF Non ←現在のマッピング 状態です。
1>                    └── 2000 ～FDFFは，まだマッピング していないので，Nonマッピング・エリアです。
  └── 起動時に，内部ROMサイズを8Kに指定したため，0 ～1FFFはROM として使用。

1>MAP W 2000,0FDFF )←2000～FDFFまでを，RAMエリアとしてIE内にマッピング します。
1>MAP )←マッピング 内容を表示し，マッピング されたことを確認します。
  0000-1FFF R/O   2000-FDFF R/W ←  変更後のマッピング 状態です。
1>                    └── 2000 ～FDFFがIE内RAMエリアに変更されました。
```

(3) プログラム のディバグをとおして，コマンドの使用方法を説明します。

・プログラム をロード するエリア をクリア します。

```
1>MEM F 0,1FFF 00 ↵ ← 0番地から1FFF番地までを00にします。
```

・プログラム をファイルよりロード します。

```
1>LOD SORT ↵←カレント・ディスク よりSORT.HEX,SORT.SYM という二つのファイルをロード します。
  object load complete     ←オブジェクト・ロードは正常に終了しました。
  symbol table loading     ←シンボル・ロードを開始します。
  MODULE01    load complete←'MODULE01'というモジュール を正常にロード しました。
1>
```

・オブジェクト・ファイルとシンボル・ファイルを別々にロードする場合は，次のコマンドを入力してください。

```
1>LOD SORT.HEX C ↵←オブジェクトのみのロード です。
                 └── オブジェクト・ロードの指定です。
                 └──── 拡張子は省略することができます。省略時は HEXとなります。
  object load complete
1>SYM K ↵ ←シンボルを削除します。
1>LOD SORT.SYM S ↵←シンボルのみのロード です。
                 └── シンボル・ロードの指定です。
                 └──── 拡張子は省略することができます。省略時は SYMとなります。
  symbol table loading
  MODULE01    load complete
1>
```

・オブジェクト・ファイル をロード したので，メモリ 内容を確認してみます。

```
1>MEM D 100,12F ↵
        └──── メモリ・ダンプ終了アドレスです。
        └──── メモリ・ダンプ開始アドレスです。
        └──── MEMコマンドのサブ・コマンド です← "D"はメモリ・ダンプ の指定です。

  0100   3A 20 01 3A 21 00 20 4A 9F 21 80 08 6F 20 00 00     : .:!. J.!..o ..
  0110   00 00 14 FB B8 00 67 42 FE D8 88 E8 59 16 5F 83     ......gB....Y._.
  0120   07 81 05 16 34 55 26 20 26 21 14 DA 00 00 00 00     ....4U& &!......
└──────┘ └──────────────────────────────────────────────┘ └──────────────┘
  アドレス                    メモリ 内容                   メモリ 内容をASCII コード で表示
1>
```

```
1>MEM D 100 ↵ ←終了ｱﾄﾞﾚｽを省略した場合は，11行分表示します。
  0100   3A 20 01 3A 21 00 20 4A 9F 21 80 08 6F 20 00 00     : .:!. J.!..o ..
  0110   00 00 14 FB B8 00 67 42 FE D8 88 E8 59 16 5F 83     .....gB....Y._.
  0120   07 81 05 16 34 55 26 20 26 21 14 DA 00 00 00 00     ....4U& &!......
  0130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>
1>MEM D ↵ ←開始ｱﾄﾞﾚｽを省略した場合は，前回表示した次のｱﾄﾞﾚｽよりﾒﾓﾘ 内容を表示します。
    └── このﾊﾟﾀｰﾝのみ"D" を省略することができます。
  01B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0200   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0210   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0220   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0230   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0240   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0250   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>MEM ↵  ← MEM Dと同じ意味です。
  0260   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0270   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0280   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0290   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0300   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>
```

・ﾏｯﾋﾟﾝｸﾞしていないﾒﾓﾘ内容を表示させようとすると，ｴﾗｰになります。

```
1>MAP ↵ ←ﾏｯﾋﾟﾝｸﾞ 状態を確認します。
  0000-1FFF R/O   2000-FDFF R/W
1>MAP K 2000,0FDFF ↵ ←2000～FDFFをﾉﾝ･ﾏｯﾋﾟﾝｸﾞにします。
1>MAP ↵ ←ﾏｯﾋﾟﾝｸﾞ 状態を確認します。
  0000-1FFF R/O   2000-FDFF Non
1>MEM D 2000,0FDFF ↵
  Mapping error
1>
```

・プログラムをロードした メモリ内容をニモニックで表示するには，次のようなコマンドを入力します。

```
DAS 100,12B ←メモリ 内容を逆アセンブル し表示するコマンドです。
    │  └──  逆アセンブル 終了アドレス
    └─────  逆アセンブル 開始アドレス
```

```
1>DAS 100,12B ) ← 100番地から 12B番地までをニモニック で表示します。
  Addr  Object            Mnemonic
                          ORG     MODULE01\SORT
                        MODULE01\SORT:
  0100  3A 20 01          MOV     0FE20H,#1H
  0103  3A 21 00          MOV     0FE21H,#0H
                        MODULE01\COMP:
  0106  20 4A             MOV     A,0FE4AH
  0108  9F 21             CMP     A,0FE21H
  010A  80 08             BNZ     $CONT
  010C  6F 20 00          CMP     0FE20H,#0H
                        MODULE01\STOP:
  010F  00                NOP
  0110  00                NOP
  0111  00                NOP
  0112  14 FB             BR      $STOP
                        MODULE01\CONT:
  0114  B8 00             MOV     R0,#0H
  0116  67 42 FE          MOVW    RP7,#0FE42H
  0119  D8                XCH     A,R0
  011A  88 E8             ADDW    RP7,RP0
  011C  59                MOV     A,[HL+]
  011D  16 5F             CMP     A,[HL]
  011F  83 07             BC      $INC1
  0121  81 05             BZ      $INC1
  0123  16 34             XCH     A,[HL-]
  0125  55                MOV     [HL],A
  0126  26 20             INC     0FE20H
                        MODULE01\INC1:
  0128  26 21             INC     0FE21H
  012A  14 DA             BR      $COMP
                          END
1>
```

・逆アセンブル 終了アドレスを指定しないと，11行表示します。

```
1>DAS 100 ) ← 100番地より逆アセンブル・リスト を11行分表示します。
  Addr  Object            Mnemonic
                          ORG     MODULE01\SORT
                        MODULE01\SORT:
  0100  3A 20 01          MOV     0FE20H,#1H
  0103  3A 21 00          MOV     0FE21H,#0H
                        MODULE01\COMP:
  0106  20 4A             MOV     A,0FE4AH
  0108  9F 21             CMP     A,0FE21H
  010A  80 08             BNZ     $CONT
  010C  6F 20 00          CMP     0FE20H,#0H
                          END
1>
```

・逆ｱｾﾝﾌﾞﾙ の ｽﾀｰﾄ･ｱﾄﾞﾚｽを指定しないと，前回の次のｱﾄﾞﾚｽより表示します。

```
1>DAS ↵ ←前回の次のｱﾄﾞﾚｽより逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ を表示します。
  Addr  Object            Mnemonic
                          ORG     MODULE01\STOP
                        MODULE01\STOP:
  010F  00                NOP
  0110  00                NOP
  0111  00                NOP
  0112  14 FB             BR      $STOP
                        MODULE01\CONT:
  0114  B8 00             MOV     R0,#0H
  0116  67 42 FE          MOVW    RP7,#0FE42H
                          END
1>
```

・ﾃﾞｰﾀ･ｴﾘｱ へｼﾝﾎﾞﾙの定義をします。

```
1>SYM A SW OFE20 )←ｽｲｯﾁとして使用するFE20番地に，SWというｼﾝﾎﾞﾙを定義します。
        │ │ └── ｼﾝﾎﾞﾙ 値
        │ └──── ｼﾝﾎﾞﾙ 名
        └────── ｼﾝﾎﾞﾙ の追加を表すSYMｺﾏﾝﾄﾞ のｻﾌﾞｺﾏﾝﾄﾞです。

1>SYM A I OFE21 ) ←ｲﾝﾃﾞｯｸｽとして使用するFE21番地に，I というｼﾝﾎﾞﾙを定義します。
1>SYM A STACK OFE80 ) ←ｽﾀｯｸ･ｴﾘｱのﾍﾞｰｽ･ｱﾄﾞﾚｽFE80番地に，STACK というｼﾝﾎﾞﾙを定義します。
1>SYM A LIST OFE42 )←並べ替えるﾃﾞｰﾀ のﾍﾞｰｽ･ｱﾄﾞﾚｽFE42番地に，LISTというｼﾝﾎﾞﾙを定義します。
1>SYM A N OFE4A ) ←ﾃﾞｰﾀ 数を表すFE4A番地に，N というｼﾝﾎﾞﾙを定義します。
```

・ｼﾝﾎﾞﾙ名は最大 8文字まで使用可能です。
・ｼﾝﾎﾞﾙの構成文字として，A ～Z,a ～z,@,?,_(ｱﾝﾀﾞｰ･ｽｺｱ),0 ～9 の文字が許されます。
ただし，ｼﾝﾎﾞﾙの先頭文字として，0 ～9 は使用できません。
・英小文字(a～z)は，英大文字(A～Z)と同じになります。

・正しいｼﾝﾎﾞﾙ名の例
A0123456
?02ADXZb
・正しくないｼﾝﾎﾞﾙ名の例
0ABCDEFG←先頭文字が数字のためｴﾗｰ
ABC-EFGH←ｼﾝﾎﾞﾙの構成文字以外の文字 '-' のためｴﾗｰ

・ﾃﾞｰﾀ の設定をするには，次に示す方法で設定します。

```
1>MEM F OFE00,OFE7F 0 ) ← ﾃﾞｰﾀ･ｴﾘｱの内容をｸﾘｱ します。

1>MEM C OFE42 )
  FE42  00 05 )
     │   │ └── 変更ﾃﾞｰﾀ
     │   └──── ﾒﾓﾘ 内容
     └──────── 変更ｱﾄﾞﾚｽ
  FE43  00 03 )
  FE44  00 04 )
  FE45  00 0A )
  FE46  00 08 )
  FE47  00 82 )
  FE48  00 0A )
  FE49  00 04 )
  FE4A  00 08 )
  FE4B  00 .)
1>          └── 変更終了ﾃﾞｰﾀ
```

・ﾃﾞｰﾀ を設定したので，ﾃﾞｰﾀ･ｴﾘｱ のﾒﾓﾘ 内容を確認します。

```
1>MEM D OFE20,OFE4A )
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                    ...........
1>
```

・ｼﾝﾎﾞﾙを使用した場合は，次のようにｺﾏﾝﾄﾞを入力します。

```
1>MEM D SW,N ↵ ←ﾒﾓﾘ 内容表示開始ｱﾄﾞﾚｽと終了ｱﾄﾞﾚｽ両方をｼﾝﾎﾞﾙで指定したｺﾏﾝﾄﾞです。
         │└── FE4A番地に定義したｼﾝﾎﾞﾙ
         └─── FE20番地に定義したｼﾝﾎﾞﾙ
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                    ...........

1>MEM D OFE20,N ↵ ←ﾒﾓﾘ 内容表示終了ｱﾄﾞﾚｽをｼﾝﾎﾞﾙで指定したｺﾏﾝﾄﾞです。
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                    ...........

1>MEM D SW,OFE4A ↵ ←ﾒﾓﾘ 内容表示開始ｱﾄﾞﾚｽをｼﾝﾎﾞﾙで指定したｺﾏﾝﾄﾞです。
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                    ...........
```

・ﾌﾞﾚｰｸ条件の設定方法を説明します。

```
1>BRA ↵ ←対話形式でﾌﾞﾚｰｸ条件を設定します。
          A 0H,0FFFFH =  MODULE01 \STOP ↵ ←ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ の設定です。
          V 0XXXXXXXXY =  ↵ ←ﾌﾞﾚｰｸ･ﾃﾞｰﾀの設定です。
    OPcode fetch                  (OP)
    Read Write                    (RW)
    Read                          (R)
    Write                         (W)
    Read Write by Program         (RWP)
    Read by Program               (RP)
    Write by Program              (WP)
    Read Write by Macro service   (RWM)
    Read by Macro service         (RM)
    Write by Macro service        (WM)
    No Condition                  (NC)
  C NC = OP ↵ ←ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ に対するｺﾝﾃﾞｨｼｮﾝ の設定です。
  L 1H = ↵ ←ﾙｰﾌﾟ･ｶｳﾝﾀ数の設定です。
1>  └── ﾃﾞｰﾀ が設定していない場合は， -- を表示します。
```

・次にﾌﾞﾚｰｸ条件の指定をします。

```
1>BRM BRA ↵ ←ﾌﾞﾚｰｸ条件として BRAを指定します。
1>

1>BRM ↵ ←ﾌﾞﾚｰｸ条件を確認します。
          BRA    ←指定したﾌﾞﾚｰｸ条件が表示されます。
1>
```

・次にﾌﾟﾛｸﾞﾗﾑ の実行をします。

```
RUN B 100 ←ﾌﾞﾚｰｸ付きﾘｱﾙ･ﾀｲﾑ 実行ｺﾏﾝﾄﾞです。
  │ └── ﾌﾟﾛｸﾞﾗﾑ 実行開始ｱﾄﾞﾚｽ
  └──── ﾌﾞﾚｰｸ指定
```

・100番地よりﾌﾟﾛｸﾞﾗﾑを実行し，BRMｺﾏﾝﾄﾞで指定したﾌﾞﾚｰｸ条件と一致するまでﾌﾟﾛｸﾞﾗﾑを実行します。

```
1>REG C PC↵
  PC        0000 = 100 ↵ ← ﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀ を 100番地に設定します。
  SP        FE72 = 0FE80 ↵ ← ｽﾀｯｸ･ﾎﾟｲﾝﾀを FE80 番地に設定します。
1>

1>RUN B 100 ↵ ← 100番地から実行します。
  User-system Vcc-ON      Emulation start at 0100
          │                    └── '100番地よりﾌﾟﾛｸﾞﾗﾑ を実行します。' というﾒｯｾｰｼﾞ です。
          └──────────────────── 'ﾕｰｻﾞ･ｼｽﾃﾑの電源が入っています。'というﾒｯｾｰｼﾞ です。
  Standard break  terminated← ﾌﾞﾚｰｸ 条件が一致し，ﾌﾟﾛｸﾞﾗﾑ の実行を停止します。
```

ﾌﾟﾛｸﾞﾗﾑ の実行を停止すると，ﾌﾞﾚｰｸしたときのﾚｼﾞｽﾀ内容を表示します。
実際には，設定されたﾌﾞﾚｰｸ･ﾎﾟｲﾝﾄ より数命令実行してからﾌﾞﾚｰｸします。

```
 PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z    RSS  AC   UF   P/V  SUB  CY
0112  FE80        0    0    0    0    0    0    0    0    0    0    1    0
 └──   次に実行するﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀ
  R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6     RP7
   X    A     C    B                           VP       UP       DE      HL
  00   00    CB   F7   FF   FF   FF   FB      FFFF     F6FF     FFFF    FE43
```

ﾚｼﾞｽﾀ内容表示後，次のﾒｯｾｰｼﾞ を表示し，入力待ちとなります。

```
One step emulation standby■ (ｶｰｿﾙ位置) ←ﾜﾝ･ｽﾃｯﾌﾟ 実行ﾓｰﾄﾞ になったので，ｷｰ入力
                          ESC ｷｰ入力       待ちとなっています。
1>←次のｽﾃｯﾌﾟを実行する場合は，ﾘﾀｰﾝ･ｷｰ を入力します。実行しない場合は，ESCｷｰ を入
   力します。
```

・ﾌﾞﾚｰｸ･ﾎﾟｲﾝﾄ まで実行したので，ﾃﾞｰﾀ を見てみます。

```
1>MEM D SW,N↵ ←SW  から Nまでのﾒﾓﾘ 内容を表示します。
  FE20   03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
  FE40   00 00 03 05 04 0A 08 82 0A 04 00                     ...........
            └──┘  このﾃﾞｰﾀ のみ並び替わっています。
1>
```

・正しくデータ が並び替わっていませんので，もう一度，データ を設定して動作を確認します。

```
1>MEM C LIST )←LISTｴﾘｱ のﾃﾞｰﾀ を設定します。
  FE42  03 05 )
  FE43  05 03 )
  FE44  04 )
  FE45  0A )
  FE46  08 )        ﾃﾞｰﾀ が変化していないので変更しません。
  FE47  82 )
  FE48  0A )
  FE49  04 )
  FE4A  00 08 )
  FE4B  08 00 )
  FE4C  00 . )
1>
```

・次にﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀを100 に変更します。

```
1>REG C PC )←ﾚｼﾞｽﾀを変更するためのｺﾏﾝﾄﾞです。
  PC        0112 = 100 )←PCを 100に変更します。
  SP        FE80 = . )←SPは変更しません。
1>
```

・次にﾄﾚｰｽ･ｺﾏﾝﾄﾞ でﾌﾟﾛｸﾞﾗﾑ の実行を確認します。

```
  RUN T ,6 REG
        |  |  └ ﾚｼﾞｽﾀ内容表示指定
        |  └──── ﾄﾚｰｽするｽﾃｯﾌﾟ数 6ｽﾃｯﾌﾟ指定
        └─────── ﾄﾚｰｽ指定
1>RUN T ,6 REG )←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀより6ｽﾃｯﾌﾟ 実行します。

  User-system Vcc-ON     Emulation start at 0100
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0103  FE80       0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   00   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43

   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   00   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43

   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80       0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   08   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43

   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  010A  FE80       0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   08   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43
```

```
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0114  FE80       0    0    0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
   X    A    C    B                           VP       UP       DE       HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF     F6FF     FFFF     FE43

  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0116  FE80       0    0    0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
   X    A    C    B                           VP       UP       DE       HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF     F6FF     FFFF     FE43

  terminated
Frame Status Address Data Label  Mnemonic                  PO EX
                             MODULE01＼CONT:               B0 FF
0000          0114               MOV     R0,#0H
One step emulation standby ) ←ワン・ステップ 実行ﾓｰﾄﾞ で実行します。
Frame Status Address Data Label  Mnemonic                  PO EX
0000          0116               MOVW    RP7,#LIST
 PC    SP  PSW: RBS2  RBS1  RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0119  FE80       0     0     0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
   X    A    C    B                           VP       UP       DE       HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF     F6FF     FFFF     FE42
One step emulation standby E ) ←ワン・ステップ 実行ﾓｰﾄﾞ を終了します。
1>
```

・次にﾌﾟﾛｸﾞﾗﾑ にﾊﾟｯﾁ を投入します。

```
1>ASM MODULE01＼CONT ) ←CONT…114 番地へﾊﾟｯﾁ を投入します。
  0114  MODULE01＼CONT:                ←ﾚｰﾍﾞﾙが指定されている場合に表示されます。
  0114                MOV     R0,#0H←114 番地の命令が表示されます。
              = BR $12C ) ←ﾊﾟｯﾁ した命令 12C番地へｼﾞｬﾝﾌﾟさせます。
        14 16 ←生成された命令のｵﾌﾞｼﾞｪｸﾄ
  0116                MOVW    RP7,#LIST ←116 番地の命令が表示されます。
              = ORG 12CH ) ←ｱﾄﾞﾚｽを12C 番地に変更します。
  012C                NOP
    │         = MOV A,I ) ←追加する命令です。
    └ ｱﾄﾞﾚｽが変更されました
        20 21
  012E                NOP
              = MOV R0,#0H ) ← 114番地にあった命令です。
        B8 00
  0130                NOP
              = BR $116 )   ← 116番地へｼﾞｬﾝﾌﾟします。
        14 E4
  0132                NOP
              = END ) ← ASMｺﾏﾝﾄﾞを終了させます。
1>
```

・ASMｺﾏﾝﾄﾞ 実行時における警告ﾒｯｾｰｼﾞ について説明をします。

・警告ﾒｯｾｰｼﾞ には次の５種類のﾒｯｾｰｼﾞ があります。

① Caution!←指定されたｱｾﾝﾌﾞﾗ の入力ではないが正しいｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞを生成しますので，その命令を実行することができます。

② Warning!←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞは生成しますが，実行した結果は保障できません。

③ Error!　←明らかに間違いがありｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞを生成できません。

④ Assemble area over!　←ｱｾﾝﾌﾞﾙ 許容領域をｵｰﾊﾞ しました。

⑤ Non map area access!←ﾏｯﾋﾟﾝｸﾞされていないｴﾘｱ に対し，ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞを生成しました。

```
1>DAS 114,116 ↵ ←変更後ﾌﾟﾛｸﾞﾗﾑ 確認のため，逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ を表示させます。
  Addr  Object              Mnemonic
                            ORG     MODULE01\CONT
                          MODULE01\CONT:
  0114  14 16               BR      $12CH
  0116  67 42 FE            MOVW    RP7,#LIST
                            END
1>DAS 12C,131 ↵
  Addr  Object              Mnemonic
                            ORG     12CH
  012C  20 21               MOV     A,I
  012E  B8 00               MOV     R0,#0H
  0130  14 E4               BR      $116H
                            END
1>
```

・ﾊﾟｯﾁ を投入しましたので，もう一度，実行します。ﾌﾞﾚｰｸ条件の指定は前と同じなので，すぐに実行します。

```
1>RUN B 100 ↵ ← 100番地から，ﾌﾟﾛｸﾞﾗﾑ を実行させます。
  User-system Vcc-ON     Emulation start at 0100
  Non map area access break    terminated ﾏｯﾋﾟﾝｸﾞ されていないﾒﾓﾘ をｱｸｾｽしたため実
                                          行を停止しました。
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    1    0    1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
                      X    A    C    B        VP       UP       DE       HL
    BD   FE   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FD00
  One step emulation standby ESC ｷｰ 入力
        1>
```

・もう一度，ﾃﾞｰﾀ･ｴﾘｱ の内容を確認します。

```
1>MEM D SW,N ↵
  FE20   73 BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00     s...............
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 03 04 05 08 0A 0A 04 08 00                    ...........
1>
```

ﾃﾞｰﾀ の並びが正しくありません。

・RSS フラグ が “1”になっていますので “0”に変更します。

```
1>REG C RSS )
  RSS         1 = 0 )
1>
1>REG ) ← レジスタ の内容を表示します。
   PC     SP   PSW: RBS2 RBS1 RBS0 IE     S     Z    RSS   AC    UF    P/V   SUB   CY
  011D   FE80         0    0    0    0     1     0     0     0     1     0     0     1
    R0    R1    R2    R3    R4    R5    R6    R7         RP4       RP5       RP6       RP7
     X     A     C     B                                  VP        UP        DE        HL
    BD    FE    CB    F7    FF    FF    FB    FF       FFFF      FFF6      FFFF      FD00
1>
```

・再度 プログラムに パッチを投入します。

10C番地で，SW…FE20番地が0かチェックしているが，比較結果に関係なく，STOP番地にジャンプしているので，その所を変更します。

```
1>ASM 10C ) ← 10C番地 パッチ投入
  010C                              CMP       SW,#0H
                                  = BR $132 ) ← 132番地へジャンプ
       14 24
  010E                              NOP
                                  = NOP ) ← 10C番地が3バイト命令なのでバイト 数を合わせるため
       00                                   の命令です。
  010F  MODULE01＼STOP:
  010F                              NOP
                                  = END )
1>ASM 132 ) ← 132番地からパッチ 投入
  0132                              NOP
                                  = CMP SW,#0H ) ← 10C番地にあった命令
       6F 20 00
  0135                              NOP
                                  = BNZ  $MODULE01＼CONT ) ←SWが 0以外の時CONT番地(114番
       80 DD                                                地) へジャンプします。
  0137                              NOP
                                  = BR $MODULE01＼STOP )
       14 D6
  0139                              NOP
                                  = END )
1>

1>DAS 10C,10E ) ← 10C番地から 10E番地まで，逆アセンブル・リスト 表示
  Addr  Object              Mnemonic
                            ORG     10CH
  010C  14 24               BR      $132H
  010E  00                  NOP
                            END
1>DAS 132,138 ) ← 132番地から138 番地まで逆アセンブル・リスト 表示
  Addr  Object              Mnemonic
                            ORG     132H
  0132  6F 20 00            CMP     SW,#0H
  0135  80 DD               BNZ     $CONT
  0137  14 D6               BR      $STOP
                            END
1>
```

```
1>DAS MODULE01\CONT )←CONT(114番地) から逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ 表示。
  Addr  Object                Mnemonic     終了ｱﾄﾞﾚｽを指定しないと，11行表示します。
                              ORG      MODULE01\CONT
                          MODULE01\CONT:
  0114  14 16                 BR       $12CH
  0116  67 42 FE              MOVW     RP7,#LIST
  0119  D8                    XCH      A,R0
  011A  88 E8                 ADDW     RP7,RP0
  011C  59                    MOV      A,[HL+]
  011D  16 5F                 CMP      A,[HL]
  011F  83 07                 BC       $INC1
                              END

1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 )←LIST～ Nのﾒﾓﾘ 内容を初期化します。
1>

          ・ﾌﾞﾚｰｸ･ﾎﾟｲﾝﾄ を新たに設定し，ﾌﾟﾛｸﾞﾗﾑ を実行します。
1>BRA A=11F C=0P )← ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽを11F 番地に指定します。
1>RUN B 100 )← 100番地より，ﾌﾟﾛｸﾞﾗﾑ を実行します。
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated←ﾌﾞﾚｰｸ条件が一致し，ﾌﾟﾛｸﾞﾗﾑ の実行を停止しました。
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0125  FE80       0    0    0    0    0    0    0    0     1    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FE42
  One step emulation standby )ｷｰ← ﾘﾀｰﾝ･ｷｰを入力し，次のｽﾃｯﾌﾟを実行します。
  Frame Status Address Data Label  Mnemonic                  P0 EX
  0000          0125               MOV      [HL],A
  0004    WR    FE42    03                                   B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0126  FE80       0    0    0    0    0    0    0    0     1    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FE42
  One step emulation standby )ｷｰ← ﾘﾀｰﾝ･ｷｰを入力し，次のｽﾃｯﾌﾟを実行します。
  Frame Status Address Data Label  Mnemonic                  P0 EX
  0000          0126               INC      SW
  0003    RD    FE20    01                                   B0 FF
  0004    WR    FE20    02                                   B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0128  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FE42
  One step emulation standby )ｷｰ← ﾘﾀｰﾝ･ｷｰを入力し，次のｽﾃｯﾌﾟを実行します。
  Frame Status Address Data Label  Mnemonic                  P0 EX
                                   MODULE01\INC1:             B0 FF
  0000          0128               INC      I
  0003    RD    FE21    00                                   B0 FF
  0004    WR    FE21    01                                   B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FE42
  One step emulation standby ESC ｷｰ 入力
```

```
1>MEM D LIST,LIST+7 ) ← LISTｴﾘｱの ﾒﾓﾘ内容を確認します。
  FE42          03 05 04 0A 08 82 0A 04                          ........

                並び替わりました。
1>MEM D I,I ) ← Iｴﾘｱ の ﾒﾓﾘ内容表示を確認します。
  FE21        01
1>

1>RUN T ,1 )←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀより 1ｽﾃｯﾌﾟ実行します。
  User-system Vcc-ON      Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                  PO EX
  0000           012A              BR         $COMP
   PC     SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   03   CB   F7   FF   FF   FB   FF        FFFF      FFF6      FFFF      FE42
  One step emulation standby )
  Frame Status Address Data Label  Mnemonic                  PO EX
                                   MODULE01＼COMP:           BO FF
  0000           0106              MOV        A,N
  0003     RD    FE4A    08                                  BO FF
   PC     SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    00   08   CB   F7   FF   FF   FB   FF        FFFF      FFF6      FFFF      FE42
  One step emulation standby ESCｷｰ入力
1>BRA A=MODULE01＼INC1 C=OP ) ←新たなﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を指定します。
1>RUN B ) ←現在のPCの値からﾌﾟﾛｸﾞﾗﾑ を実行します。
  User-system Vcc-ON      Emulation start at 0108
  Standard break    terminated
   PC     SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    01   04   CB   F7   FF   FF   FB   FF        FFFF      FFF6      FFFF      FE43
  One step emulation standby ESC ｷｰ 入力
1>

1>MEM D I,I ) ←Iｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE21        02
1>MEM D LIST,LIST+7 ) ← LISTｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE42          03 04 05 0A 08 82 0A 04                          ........

                並び替わりました。
1>RUN T ,1 )  ←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀより1ｽﾃｯﾌﾟ 実行
            User-system Vcc-ON      Emulation start at 0106
      terminated
  Frame Status Address Data Label  Mnemonic                  PO EX
                                   MODULE01＼COMP:           BO FF
  0000           0106              MOV      A,N
  0003     RD    FE4A    08                                  BO FF
   PC     SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                             VP        UP        DE        HL
    01   08   CB   F7   FF   FF   FB   FF        FFFF      FFF6      FFFF      FE43
  One step emulation standby ESC ｷｰ 入力
1>
1>RUN B ) ←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀよりﾌﾟﾛｸﾞﾗﾑ 実行します。
  User-system Vcc-ON      Emulation start at 0108
  Standard break    terminated
   PC     SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
```

```
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    02   05   CB   F7   FF   FF   FB   FF      FFFF      FFF6      FFFF      FE45
  One step emulation standby ESC キー 入力
1>MEM D I,I ↵ ←Iｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE21       03                                                    .
1>

1>MEM D LIST,LIST+7 ↵ ← LISTｴﾘｱのﾒﾓﾘ 内容を表示します。
  FE42          03 04 05 0A 08 82 0A 04                          ........
1>

1>BRA A=MODULE01\STOP 132 C=0P ↵ ← ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽを 2箇所設定します。
1>RUN B ↵ ←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀよりﾌﾟﾛｸﾞﾗﾑ 実行します。
  User-system Vcc-ON      Emulation start at 0106
  Non map area access break     terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    1    0     1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
                      X    A    C    B         VP        UP        DE        HL
    BD   FE   CB   F7   FF   FF   FF   FF      F6FF      FFFF      FFFF      FD00
  One step emulation standby ESC キー 入力
1>MEM D LIST,N ↵←LIST～N までのﾒﾓﾘ 内容を表示します。
  FE42          03 04 05 08 0A 0A 04 08 00                       .........

        この範囲だけ昇順に並びました。　N の値が変わってしまいました。

1>REG C RSS ↵
  RSS          1 = 0 ↵
1>

1>REG ↵
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    0    0     1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
                      X    A    C    B         VP        UP        DE        HL
    BD   FE   CB   F7   FF   FF   FF   FF      F6FF      FFFF      FFFF      FD00

1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ↵ ←LIST～N の範囲の ﾒﾓﾘ内容を初期化します。
1>DAS 126,12B ↵ ←ｱﾄﾞﾚｽ確認のため, 126 番地～12B 番地の逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ を表示します。
  Addr  Object              Mnemonic
                            ORG     126H
  0126  26 20               INC     SW
                          MODULE01\INC1:
  0128  26 21               INC     I
  012A  14 DA               BR      $COMP
                            END
1>
```

```
1>BRA A=126 C=OP ↵ ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ設定, INC SW命令のところです。
1>RUN B 100 ↵ 100番地よりﾌﾟﾛｸﾞﾗﾑ 実行します。
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE42
  One step emulation standby ESC ｷｰ 入力
1>

1>MEM D I,I ↵ ←Iｴﾘｱのﾒﾓﾘ 内容確認
  FE21      01                                                    .
1>

1>MEM D LIST,LIST+7 ↵ ← LISTｴﾘｱのﾒﾓﾘ 内容表示
  FE42        03 05 04 0A 08 82 0A 04                          ........

            この範囲のみ並び替えました。
1>RUN T ,1 ↵←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀより1ｽﾃｯﾌﾟ 実行させます。
  User-system Vcc-ON     Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                PO EX
  0000          012A               BRC     $COMP
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE42
  One step emulation standby ESC ｷｰ 入力
1>
1>RUN B ↵ ←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀよりﾌﾟﾛｸﾞﾗﾑ 実行させます。
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    01   04   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE43
  One step emulation standby ESC ｷｰ 入力
1>
1>MEM D I,I ↵ ←Iｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE21      02
1>

1>MEM D LIST,LIST+7 ↵ ← LISTｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE42        03 04 05 0A 08 82 0A 04                          ........

1>          この範囲のみ並び替わりました。
```

```
1>RUN T ,1 )←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀより1ｽﾃｯﾌﾟ 実行します。
  User-system Vcc-ON     Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                PO EX
  0000          012A              BR      $COMP
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    01   04   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE43
  One step emulation standby ESC ｷｰ 入力
1>

1>RUN B ) ←現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀよりﾌﾟﾛｸﾞﾗﾑ 実行します。
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    03   08   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE45
  One step emulation standby ESC ｷｰ 入力
1>MEM D I,I ) ←Iｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE21      04                                          .
1>

1>MEM D LIST,LIST+7 ) ← LISTｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE42         03 04 05 08 0A 82 0A 04                  ........

1>          この範囲が並び替わりました。
1>MEM D SW,SW ) ← SWｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE20   04                                             .
1>
            ・ ﾌﾟﾛｸﾞﾗﾑに ﾊﾟｯﾁを投入します。

1>ASM 135 ) ← 135番地へ ﾊﾟｯﾁを投入します。
  0135                      BNZ     $COMP
                    = BNZ  $MODULE01＼SORT←ｼﾞｬﾝﾌﾟ先をCONTからSORTに変更します。
       80 C9
  0137                      BR      $STOP
                    = END )
1>DAS 135,136 ) ← ﾊﾟｯﾁ投入を逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ にて確認します。
  Addr  Object          Mnemonic
                        ORG     135H
  0135  80 C9           BNZ     $SORT
                        END
1>・ﾃﾞｰﾀ･ｴﾘｱ を初期化し，再度ﾌﾟﾛｸﾞﾗﾑ を実行します。

1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ) ←LIST～N の ﾒﾓﾘ内容を変更します。
1>
            ・実行結果をﾄﾚｰｽするために，ﾄﾚｰｽ･ﾓｰﾄﾞの設定をします。

1>TRM ALL ) ←ﾄﾚｰｽ･ﾓｰﾄﾞを全ﾄﾚｰｽ･ﾓｰﾄﾞに設定しました。
1>
```

・新たにﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ ，および，ﾙｰﾌﾟ 回数を指定します。

```
1>BRA A=MODULE01＼COMP L=9 C=OP ）
        ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ          └──  COMP… 106番地の命令を 9回まで実行させます。
1>RUN B 100 ）← 100番地よりﾌﾟﾛｸﾞﾗﾑ 実行します。
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  010A  FE80       0    0    0    0    0    0    0    1     1    1    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4      RP5      RP6      RP7
     X    A    C    B                             VP       UP       DE       HL
    07   82   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE49
  One step emulation standby ESC ｷｰ 入力
1>MEM D N,N ）←Nｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE4A                                  82
                                      LISTｴﾘｱ の値が入っています。
1>MEM D I,I ）←Iｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE21       08
1>MEM D LIST,LIST+7 ）← LISTｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE42          03 04 05 08 0A 0A 04 08                          ........
```

1>　　この範囲が並び替わりました。 Nの値が入っています。

・ﾄﾚｰｽ結果を表示してみます。

```
  TRD I ALL ← ﾄﾚｰｽ されたﾃﾞｰﾀ を表示するｺﾏﾝﾄﾞです。
       │└──  ﾄﾚｰｽ･ﾃﾞｰﾀをすべて表示します。省略した場合は，11行分のみ表示します。
       └────  ﾄﾚｰｽ･ﾓｰﾄﾞをｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ に指定します。

1>TRD I ALL ）←ﾄﾚｰｽ･ﾃﾞｰﾀをｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ ですべて出力します。
  Frame Status Address Data Label  Mnemonic
                                   MODULE01＼SORT:
  0000         0100                      MOV     SW,#1H
  0004    WR   FE20    01
  0003         0103                      MOV     I,#0H
  0008    WR   FE21    00
                                   MODULE01＼COMP:
  0007         0106                      MOV     A,N
  0011    RD   FE4A    08
  0010         0108                      CMP     A,I
  0014    RD   FE21    00
  0013         010A                      BNZ     $CONT
                                   MODULE01＼CONT:
  0018         0114                      BR      $12CH
  0022         012C                      MOV     A,I
  0025    RD   FE21    00
  0024         012E                      MOV     R0,#0H
  0027         0130                      BR      $116H
  0031         0116                      MOVW    RP7,#LIST
  0034         0119                      XCH     A,R0
  0035         011A                      ADDW    RP7,RP0
  0037         011C                      MOV     A,[HL+]
  0041    RD   FE42    05
  0038         011D                      CMP     A,[HL]
  0043    RD   FE43    03
  0040         011F                      BC      $INC1
  0044         0121                      BZ      $INC1
  0046         0123                      XCH     A,[HL-]
  0050    RD   FE43    03
  0051    WR   FE43    05
  0048         0125                      MOV     [HL],A
  0054    WR   FE42    03
  0049         0126                      INC     SW
  0055    RD   FE20    01
  0056    WR   FE20    02
```

```
                                MODULE01\INCI:
0053            0128                INC     I
0059    RD      FE21    00
0060    WR      FE21    01
0058            012A                BR      $COMP
                                MODULE01\COMP:
0064            0106                MOV     A,N
0067    RD      FE4A    08
0066            0108                CMP     A,I
0070    RD      FE21    01
0069            010A                BNZ     $CONT
                                MODULE01\CONT:
0074            0114                BR      $12CH
0078            012C                MOV     A,I
0081    RD      FE21    01
0080            012E                MOV     R0,#0H
0083            0130                BR      $116H
0087            0116                MOVW    RP7,#LIST
0090            0119                XCH     A,R0
0091            011A                ADDW    RP7,RP0
0093            011C                MOV     A,[HL+]
0097    RD      FE43    05
0094            011D                CMP     A,[HL]
0099    RD      FE44    04
0096            011F                BC      $INCI
0100            0121                BZ      $INCI
0102            0123                XCH     A,[HL-]
0106    RD      FE44    04
0107    WR      FE44    05
0104            0125                MOV     [HL],A
0110    WR      FE43    04
0105            0126                INC     SW
0111    RD      FE20    02
0112    WR      FE20    03
                                MODULE01\INCI:
0109            0128                INC     I
0115    RD      FE21    01
0116    WR      FE21    02
0114            012A                BR      $COMP
                                MODULE01\COMP:
0120            0106                MOV     A,N
0123    RD      FE4A    08
0122            0108                CMP     A,I
0126    RD      FE21    02
0125            010A                BNZ     $CONT
                                MODULE01\CONT:
0130            0114                BR      $12CH
0134            012C                MOV     A,I
0137    RD      FE21    02
0136            012E                MOV     R0,#0H
0139            0130                BR      $116H
0143            0116                MOVW    RP7,#LIST
0146            0119                XCH     A,R0
0147            011A                ADDW    RP7,RP0
0149            011C                MOV     A,[HL+]
0153    RD      FE44    05
0150            011D                CMP     A,[HL]
0155    RD      FE45    0A
0152            011F                BC      $INCI
                                MODULE01\INCI:
0158            0128                INC     I
0161    RD      FE21    02
0162    WR      FE21    03
0160            012A                BR      $COMP
                                MODULE01\COMP:
```

```
0166          0106              MOV     A,N
0169    RD    FE4A    08
0168          0108              CMP     A,I
0172    RD    FE21    03
0171          010A              BNZ     $CONT
                          MODULE01\CONT:
0176          0114              BR      $12CH
0180          012C              MOV     A,I
0183    RD    FE21    03
0182          012E              MOV     R0,#0H
0185          0130              BR      $116H
0189          0116              MOVW    RP7,#LIST
0192          0119              XCH     A,R0
0193          011A              ADDW    RP7,RP0
0195          011C              MOV     A,[HL+]
0199    RD    FE45    0A
0196          011D              CMP     A,[HL]
0201    RD    FE46    08
0198          011F              BC      $INCI
0202          0121              BZ      $INCI
0204          0123              XCH     A,[HL-]
0208    RD    FE46    08
0209    WR    FE46    0A
0206          0125              MOV     [HL],A
0212    WR    FE45    08
0207          0126              INC     SW
0213    RD    FE20    03
0214    WR    FE20    04
                          MODULE01\INCI:
0211          0128              INC     I
0217    RD    FE21    03
0218    WR    FE21    04
0216          012A              BR      $COMP
                          MODULE01\COMP:
0222          0106              MOV     A,N
0225    RD    FE4A    08
0224          0108              CMP     A,I
0228    RD    FE21    04
0227          010A              BNZ     $CONT
                          MODULE01\CONT:
0232          0114              BR      $12CH
0236          012C              MOV     A,I
0239    RD    FE21    04
0238          012E              MOV     R0,#0H
0241          0130              BR      $116H
0245          0116              MOVW    RP7,#LIST
0248          0119              XCH     A,R0
0249          011A              ADDW    RP7,RP0
0251          011C              MOV     A,[HL+]
0255    RD    FE46    0A
0252          011D              CMP     A,[HL]
0257    RD    FE47    82
0254          011F              BC      $INCI
                          MODULE01\INCI:
0260          0128              INC     I
0263    RD    FE21    04
0264    WR    FE21    05
0262          012A              BR      $COMP
                          MODULE01\COMP:
0268          0106              MOV     A,N
0271    RD    FE4A    08
0270          0108              CMP     A,I
0274    RD    FE21    05
0273          010A              BNZ     $CONT
                          MODULE01\CONT:
```

```
0278          0114              BR      $12CH
0282          012C              MOV     A,I
0285    RD    FE21    05
0284          012E              MOV     R0,#0H
0287          0130              BR      $116H
0291          0116              MOVW    RP7,#LIST
0294          0119              XCH     A,R0
0295          011A              ADDW    RP7,RP0
0297          011C              MOV     A,[HL+]
0301    RD    FE47    82
0298          011D              CMP     A,[HL]
0303    RD    FE48    0A
0300          011F              BC      $INCI
0304          0121              BZ      $INCI
0306          0123              XCH     A,[HL-]
0310    RD    FE48    0A
0311    WR    FE48    82
0308          0125              MOV     [HL],A
0314    WR    FE47    0A
0309          0126              INC     SW
0315    RD    FE20    04
0316    WR    FE20    05
                      MODULE01＼INCI:
0313          0128              INC     I
0319    RD    FE21    05
0320    WR    FE21    06
0318          012A              BR      $COMP
                      MODULE01＼COMP:
0324          0106              MOV     A,N
0327    RD    FE4A    08
0326          0108              CMP     A,I
0330    RD    FE21    06
0329          010A              BNZ     $CONT
                      MODULE01＼CONT:
0334          0114              BR      $12CH
0338          012C              MOV     A,I
0341    RD    FE21    06
0340          012E              MOV     R0,#0H
0343          0130              BR      $116H
0347          0116              MOVW    RP7,#LIST
0350          0119              XCH     A,R0
0351          011A              ADDW    RP7,RP0
0353          011C              MOV     A,[HL+]
0357    RD    FE48    82
0354          011D              CMP     A,[HL]
0359    RD    FE49    04
0356          011F              BC      $INCI
0360          0121              BZ      $INCI
0362          0123              XCH     A,[HL-]
0366    RD    FE49    04
0367    WR    FE49    82
0364          0125              MOV     [HL],A
0370    WR    FE48    04
0365          0126              INC     SW
0371    RD    FE20    05
0372    WR    FE20    06
                      MODULE01＼INCI:
0369          0128              INC     I
0375    RD    FE21    06
0376    WR    FE21    07
0374          012A              BR      $COMP
                      MODULE01＼COMP:
0380          0106              MOV     A,N
0383    RD    FE4A    08
0382          0108              CMP     A,I
```

```
0386    RD    FE21    07
0385          010A              BNZ     $CONT
                          MODULE01\CONT:
0390          0114              BR      $12CH
0394          012C              MOV     A,I
0397    RD    FE21    07
0396          012E              MOV     R0,#0H
0399          0130              BR      $116H
0403          0116              MOVW    RP7,#LIST
0406          0119              XCH     A,R0
0407          011A              ADDW    RP7,RP0
0409          011C              MOV     A,[HL+]
0413    RD    FE49    82
0410          011D              CMP     A,[HL]
0415    RD    FE4A    08
0412          011F              BC      $INC1
0416          0121              BZ      $INC1
0418          0123              XCH     A,[HL-]
0422    RD    FE4A    08  ]  Nのﾒﾓﾘ 内容を書き替えています。
0423    WR    FE4A    82  ]
0420          0125              MOV     [HL],A
0426    WR    FE49    08     Nのﾃﾞｰﾀ を LIST ｴﾘｱ の最後に書いています。
0421          0126              INC     SW
0427    RD    FE20    06
0428    WR    FE20    07
                          MODULE01\INC1:
0425          0128              INC     I
0431    RD    FE21    07
0432    WR    FE21    08
0430          012A              BR      $COMP
                          MODULE01\COMP:
0436          0106              MOV     A,N
0439    RD    FE4A    82
0438          0108              CMP     A,I
0442    RD    FE21    08
1>
```

・N のﾃﾞｰﾀ が変化するのを防ぐためﾊﾟｯﾁ を投入します。

```
1>DAS 106,109 ↵ ← Nの値を Aに入れている命令のｱﾄﾞﾚｽを，逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ により確認します。
  Addr  Object              Mnemonic
                            ORG     MODULE01＼COMP
                          MODULE01＼COMP:
  0106  20 4A               MOV     A,N
  0108  9F 21               CMP     A,I
                            END
1>ASM 106 ↵ ← 106番地にﾊﾟｯﾁ を投入します。
  0106  MODULE01＼COMP:
  0106                      MOV     A,I
                  = BR $139 ↵ ← 139番地にｼﾞｬﾝﾌﾟします。
        14 31
  0108                      CMP     A,I
                  = END ↵
1>ASM 139 ↵ ← 139番地にﾊﾟｯﾁ 投入します。
  0139                      NOP
                  = MOV A,N ↵ ← 106番地にあった命令です。
        20 4A
  013B                      NOP
                  = DEC R1 ↵←追加する命令です。A の値を1ﾃﾞｸﾘﾒﾝﾄ します。
        C9
  013C                      NOP
                  = CMP A,I ↵ ← 108番地にある命令です。
        9F 21
  013E                      NOP
                  = BNZ  $MODULE01＼CONT ↵← 10A番地にある命令です。
        80 D4
  0140                      NOP
                  = BR $10C ↵ ← 10C番地へｼﾞｬﾝﾌﾟします。
        14 CA
  0142                      NOP
                  = END ↵

1>ASM MODULE01＼SORT ↵
  0100  MODULE01＼SORT:
  0100                      MOV     SW,#1H
                  = MOV SW,#0H ↵←SWの初期値を 00 にします。
        3A 20 00
  0103                      MOV     I,#0H
                  = END ↵
1>
```

・ﾃﾞｰﾀ･ｴﾘｱ のｼﾝﾎﾞﾙ値を変更します。

```
1>SYM C LIST 0FE22 ↵←LISTのｼﾝﾎﾞﾙ値をFE22に変更します。
          │ │  └──── 新しいｼﾝﾎﾞﾙ値
          │ └─────── ｼﾝﾎﾞﾙ名
          └───────── 変更
1>SYM C N 0FE2A ↵ ← Nのｼﾝﾎﾞﾙ値をFE2Aに変更します。
1>SYM E STAC  ↵   ← STACKというｼﾝﾎﾞﾙは使用しないので削除します。
          │ └──── 削除するｼﾝﾎﾞﾙ名
          └─────── 削除
1>
```

・ｼﾝﾎﾞﾙ値を変更したため，ﾌﾟﾛｸﾞﾗﾑ を変更します。

```
1>MEM C 117 ↵ ← 117番地のﾒﾓﾘ 内容を変更します。
  0117  42 22 ↵ ←42から22にﾒﾓﾘ 内容を変更します。
  0118  FE .↵ ← ﾒﾓﾘ変更終了ｷｬﾗｸﾀ を入力しました。
1>MEM C 13A ↵ ← 13A番地のﾒﾓﾘ 内容を変更します。
  013A  4A 2A ↵ ←4Aから2Aにﾒﾓﾘ 内容を変更します。
  013B  C9 .↵
1>

1>DAS 100,141 ↵ ←変更後のﾌﾟﾛｸﾞﾗﾑ を逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ にて表示します。
  Addr  Object              Mnemonic
                            ORG     MODULE01\SORT
                        MODULE01\SORT:
  0100  3A 20 00            MOV     SW,#0H
  0103  3A 21 00            MOV     I,#0H
                        MODULE01\COMP:
  0106  14 31               BR      $139H
  0108  9F 21               CMP     A,I
  010A  80 08               BNZ     $CONT
  010C  14 24               BR      $132H
  010E  00                  NOP
                        MODULE01\STOP:
  010F  00                  NOP
  0110  00                  NOP
  0111  00                  NOP
  0112  14 FB               BR      $STOP
                        MODULE01\CONT:
  0114  14 16               BR      $12CH
  0116  67 22 FE            MOVW    RP7,#LIST
              └──  42から22に変更されています。
          0119  D8                    XCH     A,R0
  011A  88 E8               ADDW    RP7,RP0
  011C  59                  MOV     A,[HL+]
  011D  16 5F               CMP     A,[HL]
  011F  83 07               BC      $INC1
  0121  81 05               BZ      $INC1
  0123  16 34               XCH     A,[HL-]
  0125  55                  MOV     [HL],A
  0126  26 20               INC     SW
                        MODULE01\INC1:
  0128  26 21               INC     I
  012A  14 DA               BR      $COMP
  012C  20 21               MOV     A,I
  012E  B8 00               MOV     R0,#0H
  0130  14 E4               BR      $116H
  0132  6F 20 00            CMP     SW,#0H
  0135  80 C9               BNZ     $SORT
  0137  14 D6               BR      $STOP
  0139  20 2A               MOV     A,N
              └──  4Aから2Aに変更されています。
  013B  C9                  DEC     R1
  013C  9F 21               CMP     A,I
  013E  80 D4               BNZ     $CONT
  0140  14 CA               BR      $10CH
                            END
1>
```

・ﾃﾞｰﾀ･ｴﾘｱ を初期化し，再度ﾌﾟﾛｸﾞﾗﾑ を実行します。

```
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ↵←LIST～N のﾒﾓﾘ 内容を初期化します。
1>BRA A=MODULE01＼STOP C=OP ↵← ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽをSTOPに指定します。
1>RUN B 100 ↵← 100番地より，ﾌﾟﾛｸﾞﾗﾑ を実行します。
  User-system Vcc-ON      Emulation start at 0100
  Standard break     terminated
   PC     SP   PSW: RBS2 RBS1 RBS0 IE     S     Z    RSS  AC     UF    P/V  SUB  CY
  0112   FE80         0    0    0    0     0     1     0    0      1     0    1    0
    R0    R1    R2    R3    R4    R5    R6    R7         RP4       RP5       RP6       RP7
     X     A     C     B                                 VP        UP        DE        HL
    06    07    CB    F7    FF    FF    FF    FF        F6FF      FFFF      FFFF      FE29
  One step emulation standby ESC ｷｰ 入力
1>MEM D LIST,LIST+7 ↵← LISTｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE22          03 04 04 05 08 0A 0A 82                          ........
```

昇順に並び替わりました。

```
1>MEM D N,N ↵←Nｴﾘｱのﾒﾓﾘ 内容を確認します。
  FE2A                                  08                                .
1>MEM D SW,I ↵←SW～N のﾒﾓﾘ 内容を確認します。
  FE20    00 07                                                          ..
1>          │ └──  I
            └────  SW
```

・ﾌﾟﾛｸﾞﾗﾑが正常に動いたので，ﾒﾓﾘ内容をﾌｧｲﾙにｾｰﾌﾞします。

```
  SAV SORT01.HEX 100,141←  100 番地から 141番地の ﾒﾓﾘ内容をﾌｧｲﾙに ｾｰﾌﾞするための
              │          │ │  ｺﾏﾝﾄﾞです。
              │          │ └── ｾｰﾌﾞ 終了ｱﾄﾞﾚｽです。
              │          └──── ｾｰﾌﾞ 開始ｱﾄﾞﾚｽです。
              └──────────────── ｾｰﾌﾞ するﾌｧｲﾙ名です。ﾄﾞﾗｲﾌﾞ番号が省略されているので，ｶﾚﾝﾄ･ﾃﾞｨｽｸ 上にﾌｧｲﾙを作成します。
```

```
1>SAV SORT01.HEX 100,141 ↵
  object save complete  ←ｾｰﾌﾞ･ｺﾏﾝﾄﾞ正常終了ﾒｯｾｰｼﾞ です。
1>
```

・ｾｰﾌﾞ したﾌｧｲﾙ内容とﾒﾓﾘ 内容が同一かﾁｪｯｸします。

```
1>VRY SORT01.HEX ↵← ｶﾚﾝﾄ･ﾃﾞｨｽｸ上にあるSORT01.HEXとﾒﾓﾘ 内容を比較します。
  object verify complete
1>
```

第４章　コマンドの説明

４．１　概　　要

この章では，ＩＥ－７８３１０Ａ－Ｒのコマンドの詳細について述べます。

本章は，ソフトウエア編「第３章　基本的な使用方法」を読み，一通りＩＥ－７８３１０Ａ－Ｒをお使いになってからお読みください。

４．２　では，ＩＥ－７８３１０Ａ－Ｒのコマンド入力の方法について説明します。

４．３　では，コマンドに用いる数値表現，シンボル表現について説明します。

４．４　では，コマンド全体に共通することについて説明します。

４．５　では，コマンド一つ一つに対する詳細な説明をします。

４．６　では，エラー・メッセージについて説明します。

４．７　では，ＡＳＭコマンドに適用されるオンライン・アセンブラ仕様，ＤＡＳコマンド等に適用される逆アセンブラ仕様について説明します。

４．２　コマンド入力方法

４．２．１　概　　要

ＩＥ－７８３１０Ａ－Ｒでは，

①　スタンドアロン時，

②　ＰＤＡ－８８０上でシステム・ソフトウエアを動作させたとき，

③　ＭＤ－１１６／０８６／０８０上でシステム・ソフトウエアを動作させたとき，

で，コマンドの入力方法が若干異なります。①，②，③の順序でより使い易いコマンド入力ができるようになります。

①では，制御キーによる１文字削除，１行削除が可能となっています。

②では，①に加えて，

・ファイルからのコマンド入力

・一部のコマンドに対するコマンド省略形入力

・コマンド・ヒストリ機能

・コマンド・ファイル作成機能

が可能になります。

③では，さらに，

・行単位の編集機能

が可能になります。

４．２．２　制御キー

コマンド入力時の制御キーについて説明します。

制御キーは，

①スタンドアロンの場合，

②ＰＤＡ－８８０上でシステム・ソフトウエアを動作させた場合，

③ＭＤ－１１６／０８６／０８０上でシステム・ソフトウエアを動作させた場合，

で，若干異なります。

①では，１文字削除，１行削除，行入力の終了，行入力のキャンセル等の制御キーが有効です。

②では，①の制御キーに加えて，ヒストリ機能を呼出す制御キーが有効になります。

③では，②の制御キーに加えて，カーソル制御キー，インサート・モード・キーが有効になります。

表４－１　ＩＥ－７８３１０Ａ－Ｒで使用可能な制御キー

キー	機　　能	①	②	③
ＤＥＬ ＢＳ（↑Ｈ）	１文字削除：カーソルの直前のキャラクタを削除し，カーソルを一つ左に移動する	○	○	○
↑Ｘ	１行削除：カーソルのある行を削除し，カーソルをその行の先頭に置く	○	○	○
ＴＡＢ（↑Ｉ）	スペースと同じ	○	○	○
ＣＲ（↑Ｍ） ＬＦ（↑Ｊ）	行入力の終了	○	○	○
ＥＳＣ	キャンセル：プロンプトを出力して，コマンドの入力に戻る	○	○	○
；	コメント：この文字以降をコメントとし，コマンド行の一部として解釈しない	○	○	○
！	ヒストリ機能の呼出し	×	○	○
↑Ｋ	ＳＴＲコマンドの中断	×	○	○
↑Ｌ	ＳＴＲコマンドの一時停止	×	○	○
→	入力されている１行の範囲でカーソルを右へ１文字分移動する	×	×	○
←	入力されている１行の範囲でカーソルを左へ１文字分移動する	×	×	○
↑Ａ	インサート・モード・トグル・スイッチ	×	×	○

注　↑は，ＣＴＲＬキーを示します（以後，すべて同じです）。

① スタンドアロン時の制御キー

キー	コード	機　　能
ＤＥＬ ＢＳ（↑Ｈ）	７ＦＨ ０８Ｈ	１文字削除：カーソルの直前のキャラクタを削除し，カーソルを一つ左に移動する。
↑Ｘ	１８Ｈ	１行削除：カーソルのある行を削除し，カーソルをその行の先頭に置く。
ＴＡＢ（↑Ｉ）	０９Ｈ	スペース（２０Ｈ）と同じ。
ＣＲ（↑Ｍ） ＬＦ（↑Ｊ）	０ＤＨ ０ＡＨ	行入力の終了
ＥＳＣ	１ＢＨ	キャンセル：プロンプトを出力し，コマンド入力モードに戻る。
；	３ＢＨ	コメント：この文字以降をコメントとし，コマンド行の一部としては解釈しない。

ＩＥ－７８３１０Ａ－Ｒで使用できる基本的な制御キーです。これらの制御キーは，ＰＤＡ－８８０，ＭＤ－１１６／０８６／０８０上でシステム・ソフトウエアを動作させた場合でも使用できます。

② ＰＤＡ－８８０上でシステム・ソフトウエアを動作させた場合

キー	コード	機　　能
！	２１Ｈ	ヒストリ機能の呼出し
↑Ｋ	０ＢＨ	ＳＴＲコマンドの中断
↑Ｌ	０ＣＨ	ＳＴＲコマンドの一時停止

コマンド入力待ち時，最初に‘！！↵’をキー入力すると直前に入力したコマンドが画面に再表示され，キー入力待ちになります。このとき‘↵’を入力すると，再表示されたコマンドを実行することができます。

また，同様に‘！ｎ↵’（ｎは１～２０の数字）をキー入力すると，ヒストリ・メモリに記憶されているｎ行目のコマンドが，画面に再表示され，キー入力待ちになります。

ＳＴＲコマンドを中断する場合は，↑Ｋキーを使用します。ＳＴＲコマンドで，ファイルからコマンド，あるいは，データを入力しているときに↑Ｋを入力すると，これ以降ファイルからの入力は中断されます。

また，ＳＴＲコマンドを一時停止する場合は，↑Ｌキーを使用します。

ＳＴＲコマンドで，ファイルからコマンド，あるいは，データを入力していると

きに↑Ｌキーを入力すると，ファイルからの入力は一時停止されます。一時停止されている状態で，もう一度↑Ｌキーを入力すると，ファイルからのコマンド，あるいは，データの入力が再開されます。

③　ＭＤ－１１６／０８６／０８０上でシステム・ソフトウエアを動作させた場合

キー	コード	機　　能
→	＊１	入力されている１行の範囲でカーソルを右へ１文字分移動する。
←	＊１	入力されている１行の範囲でカーソルを左へ１文字分移動する。
↑Ａ	０１Ｈ	インサート・モード・トグル・スイッチ

＊１：　ＣＲＴ内蔵タイプのＭＤと，ＭＤ－９１０ＴＭで異なります。

ディフォールトでは，インサート・モードはｏｆｆになります。なお，１行入力終了ごとにディフォールトが設定されます。

インサート・モード時には，カーソル位置に‘＜’が表示され，インサート・モードであることがわかります。

４．２．３　スタンド・アロン時のコマンド入力方法

ＩＥ－７８３１０Ａ－Ｒは，スタンド・アロン時でも，１文字削除および１行削除による行編集ができます。

例１　１文字削除の例

＊ＭＥＭ　Ｄ　０，０ＦＦＧ＿

└カーソルの位置

└誤ってキー入力した文字

この場合は，ＤＥＬキーかＢＳキーをキー入力して，“Ｇ”を削除します。表示は次のようになります。

＊ＭＥＭ　Ｄ　０，０ＦＦ＿

└カーソルの位置

例２　行削除の例

＊ＭＥＭ　Ｄ　０，０ＦＦＨ＿

└カーソルの位置

この行すべてを削除し，新たにコマンドを入力する場合は‘↑Ｘ’キーをキー入力します。表示は次のようになります。

＊＿

└カーソルの位置

例３　キャンセルの例

行入力の途中で‘ＥＳＣ’キーを入力します。入力中のコマンド行はキャンセルされ，新たにコマンド入力待ちになります。

＊ＭＥＭ　Ｄ　０，０ＦＦＨ

└‘ＥＳＣ’キー入力

＊＿

└カーソル位置

４．２．４　ＰＤＡ－８８０使用時のコマンド入力方法

ＰＤＡ－８８０の５７ＫＣＰ／Ｍ上で，システム・ソフトウエアを動作させた場合は，基本的にはスタンドアロン時のコマンド入力方法と同様です。１文字削除，１行削除による行編集ができます。

また，これに加えてコマンド・ヒストリ機能が可能となります。システム・ソフトウエアはキー入力したコマンド行を，最新の２０行だけＦＩＦＯ的に記憶しています。これを，コマンド・ヒストリといいます。

このコマンド・ヒストリの任意の１行を，コマンド・ヒストリから呼出し，そのままコマンドを再実行することができます。

コマンド・ヒストリの内容は，‘ＨＩＳ’コマンドで表示することができます。今，

‘ＨＩＳ’コマンドを実行し，次のようにヒストリが表示されたとします。

```
 1    MAP  R  0，1FFFH
 2    MAP  W  2000H，2FFFH
 3    MAP
 4    MEM
  .
  .
  .
17    MEM  D  0，0FFH
18    MEM  D
19    MEM  C  START
20    HIS
```

このとき，１７番のコマンド行を再表示する場合は，コマンド入力待ちの状態で，次のようにキー入力します。

```
n＞！１７⤶
   MEM  D  0，0FFH_  ←１７番目のコマンド
                  ↑
                  └─カーソル位置
```

このまま，コマンドを実行させる場合は，この後すぐに‘⤶’を入力します。このコマンド行を修正する場合は，‘ＤＥＬ’キーか‘ＢＳ’キーを用いて，編集した後‘⤶’をキー入力します。

また，直前に入力したコマンド行（この例では，２０番のコマンド行）を，再表示する場合は，‘！’の次にコマンド行の番号を入力する代わりに，もう１回‘！’を入力します。つまり，次のように入力します。

```
n＞！！⤶
   ＨＩＳ_    ←最後に入力されたコマンド
      ↑
      └─カーソル位置
```

４．２．５　ＭＤ－１１６／０８６／０８０使用時のコマンド入力方法

ＭＤ－１１６ＦＤ－１０，ＭＤ－１１６ＨＤ－１０，ＭＤ－０８６ＨＤ－１０，ＭＤ－０８６ＦＤ－１０，ＭＤ－０８０ＦＤ－１０，および，ＭＤ－０８６ＦＤ，ＭＤ－０８０ＦＤの本体コンソールでは，カーソル制御キーを使用して行単位のスクリーン・エディト機能が可能になります。

カーソルは，コマンド行の先頭から最後までの間で移動できます。カーソル位置の文字を，キー入力した文字と置換えることができます。

また，‘↑Ａ’キーを入力することにより，インサート・モードになります。インサート・モードでは，カーソル位置に，文字列を挿入することができます。インサート・モードでは，カーソル位置に‘＜’が表示され，インサート・モードであることがわかります。

インサート・モードを解除する場合は，もう１回‘↑Ａ’キーを入力します。なお，

プロンプトが表示される度に，インサート・モードは解除されます。

コマンド入力終了の，‘⤶’は，コマンド行中のどこで入力してもかまいません。

したがって，コマンド入力終了時に，カーソルをコマンド行の最後まで移動する必要はありません。

ただし，マルチウインドウ・モードの場合および，ＭＤ－０８６／０８０ＦＤの，本体以外のコンソールでは，スクリーン・エディト機能はサポートされませんので注意してください。この場合は，ＰＤＡ－８８０使用時のコマンド入力方法と同様です。

ただし，コマンド・ヒストリ機能でコマンド行を再表示した場合のカーソル位置が異なります。ＰＤＡ－８８０使用時は，コマンド行の最後の文字の次が，ＭＤ－０８６／０８０ＦＤ使用時は，コマンド行の先頭の文字が，それぞれカーソル位置になります。

４．３　数値，シンボル，式の記述仕様

コマンドのオペランドに使用される数値，シンボル，または，式の記述規則について説明します。オペランドに使用される数値，シンボル，または，式は，次の４種類に分類されます。

（１）数値表現
（２）シンボル表現
（３）式表現
（４）特殊数値表現

数値表現は，直接数値を記述します。記述された数値は，１６ビット，または，８ビットで評価されます。また，シンボル表現，あるいは，式表現で置換えることもできます。

シンボル表現は，定義済みのシンボル（通常は，シンボル・ファイルをロードする）を記述します。ただし，ビット・タイプのシンボルは記述できません。

式表現は，複数の数値表現，あるいは，シンボル表現を演算子で結合して記述します。

特殊数値表現は，数値表現の特別な場合を表します。数値表現，シンボル表現，および，式表現はただ一つの数値を表します。これに対して，特殊数値表現は複数の数値の集まりを表します。

４．３．１　数値表現

数値は，１６ビットと８ビットの２種類があります。また，数値は，１６進数，１０進数，８進数，および，２進数の記述ができます。それぞれ数値の最後にＨ，Ｔ，Ｑ，Ｙのサフィックスを付加することが必要です。ただし，‘ＳＵＦ’コマンドでサフィックスを指定してある場合は，これを省略できます。

数値の最上位桁は，数字（０～９）でなければなりません。また，数値の前に符号（＋，または，－）を付加できます。この場合も符号の次は，数字（０～９）でなければなりません。

数値の範囲は，１６ビットの場合，

１６進数で　０～ＦＦＦＦ
１０進数で　０～６５５３５
８進数で　０～１７７７７７
２進数で　０～１１１１１１１１１１１１１１１１

です。８ビットの場合，

１６進数で　０～ＦＦ

１０進数で　０～２５５

８進数で　０～３７７

２進数で　０～１１１１１１１１

です。これより大きな数値の場合は，エラーになります。

例

```
  ０ＦＦＦＦ
－０ＦＦＦＦ      ＯＫ

  －ＦＦＦＦ      ＦＦＦＦというシンボルに符号が付いたものとみなします。

    ＦＦＦＦ      ＦＦＦＦというシンボルとみなします。

  １ＦＦＦＦ
－１ＦＦＦＦ      数値がＦＦＦＦより大きいのでエラーとなります。
０Ｆ０ＦＦＦ
```

４．３．２　シンボル表現

シンボルは，ローカル・シンボルとパブリック・シンボルの２種類があります。通常これらのシンボルは，シンボル・テーブル・ファイルをロードすることで定義されます。

定義済みのシンボルは，数値の代わりに記述できます。ただし，ビット・タイプのシンボルは記述できません。記述できるシンボルの種類は，次の３種類です。

①　コード・タイプ

②　データ・タイプ

③　ナンバ・タイプ

また，シンボル値が２５６以上のシンボルは，８ビット数値の代わりに記述することはできません。

シンボルは，

Ａ～Ｚ，ａ～ｚ，@，？，＿（アンダー・スコア），０～９

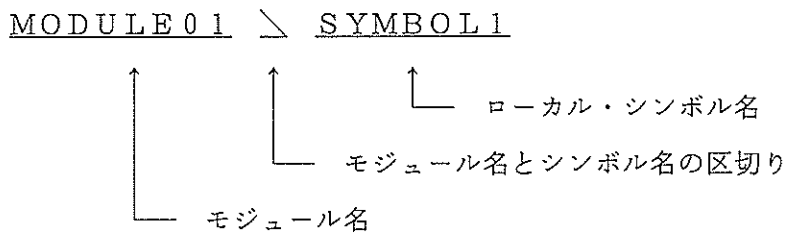
の，いずれかの文字（最大８文字）で構成されます。ただし，先頭の文字は，数字（０～９）以外の文字でなければなりません。また，英小文字（ａ～ｚ）は，英大文字（Ａ～Ｚ）と同じになります。

もし，８文字以上の文字が記述された場合は，先頭から８文字が有効になります。

例　ＡＢＣＤＥＦＧＨＩＪＫ　は，ＡＢＣＤＥＦＧＨ　が有効となります。
　　また，ａｂｃｄｅｆｇｈ　は，ＡＢＣＤＥＦＧＨ　と同じになります。

ローカル・シンボルは，モジュール名と対にして表現しなければなりません。パブリック・シンボルは，シンボル名だけを記述します。

例　・ローカル・シンボルの記述

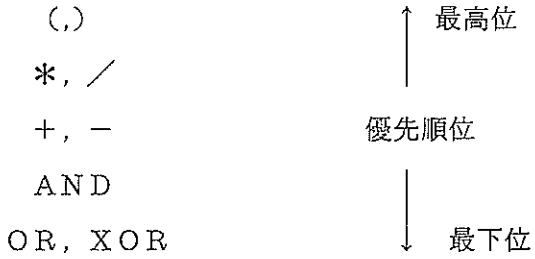


・パブリック・シンボルの記述

ＳＹＭＢＯＬ１０　――　パブリック・シンボル名

４．３．３　式表現

式表現は，数値表現の代わりに記述できます。式表現は，数値と数値，シンボルとシンボル，あるいは，数値とシンボルを演算子で結合して記述します。式には，以下の演算子が使用できます。また，カッコの使用もできます。カッコのネストは，最大３２レベルまでネストすることができます。



なお，演算はすべて１６ビットの整数で行なわれます。演算の中間結果，あるいは，最終結果が１６ビット以上になった場合は，１７ビット目より上位は切捨てます。

例

１０Ｈ＋１０Ｈ	→	２０Ｈ
１０Ｈ／３Ｈ	→	５Ｈ
０ＦＦＦＦＨ＋２Ｈ	→	１Ｈ
１Ｈ－２Ｈ	→	０ＦＦＦＦＨ

（１００Ｈ＊１００Ｈ）／（１００Ｈ＊１００Ｈ）→　エラー（ゼロ・ディバイド）

演算は，数値，および，シンボルに対してだけ有効です。予約語，あるいは，特殊数値に対しての演算はエラーになります。

なお，数値，あるいは，シンボルと（　）　＊　／　＋　－の各演算子は，連続していてもかまいません。また，数値，あるいは，シンボルとＡＮＤ　ＯＲ　ＸＯＲの各演算子は，一つ以上のスペースが演算子の前後に必要です。

式表現を，８ビット数値の代わりに記述した場合も，演算は１６ビットでします。したがって，中間結果が８ビットより大きくてもエラーにはなりません。ただし，最終結果が８ビットより大きくなった場合は，エラーとなります。

４．３．４　特殊数値表現

特殊数値表現は，１６ビット，および，８ビットの複数の数値の集まりを表します。特殊数値表現は，‘Ｘ’で表現します。ただし，１０進数の‘Ｘ’表現はできません。

‘Ｘ’は，１６進数の場合は０～Ｆに，８進数の場合は０～７に，２進数の場合は０と１にそれぞれ対応します。

ただし，８進数の場合は，１６ビットと８ビットでは最上位桁とその他の桁では異なる対応をします。１６ビットの場合の最上位桁（６桁目）は，０と１に対応します。８ビットの場合の最上位桁（３桁目）は，０～３に対応します。

例	０ＸＸＸＸＨ	→	０Ｈ～０ＦＦＦＦＨ	
	０ＸＸＸＸＸＸＱ	→	０Ｑ～１７７７７７Ｑ	最上位桁に注意 １６ビット：１ ８ビット：３
	０ＸＸＸＱ	→	０Ｑ～３７７Ｑ	
	０ＸＸＸＸＸＸＸＸＹ	→	０Ｙ～１１１１１１１１Ｙ	

特殊数値表現は，アドレス範囲，１６ビット，あるいは，８ビット・マスク・データがあります。アドレス範囲は，１６ビットの連続した数値となります。マスク・データは，連続した数値とはなりません。

アドレス範囲として記述する場合は，‘Ｘ’を下位桁から連続して記述しなければなりません。連続していない場合は，マスク・データと判断されます。

マスク・データとして記述する場合は，‘Ｘ’は連続して記述する必要はありません。ただし，‘Ｘ’が連続していてもアドレス範囲のように連続した数値にはなりません。‘Ｘ’で表現された桁が何であってもよいことを示すだけです。

例　０Ｘ００Ｈ　→　０００００Ｈ，０１００Ｈ，０２００Ｈ，０３００Ｈ，
０４００Ｈ，０５００Ｈ，０６００Ｈ，０７００Ｈ，
０８００Ｈ，０９００Ｈ，０Ａ００Ｈ，０Ｂ００Ｈ，
０Ｃ００Ｈ，０Ｄ００Ｈ，０Ｅ００Ｈ，０Ｆ００Ｈ

０１Ｘ１Ｑ　→　０１０１Ｑ，０１１１Ｑ，０１２１Ｑ，０１３１Ｑ，
０１４１Ｑ，０１５１Ｑ，０１６１Ｑ，０１７１Ｑ

１Ｘ１０１０Ｘ１Ｙ　→　１０１０１００１Ｙ，１１１０１００１Ｙ
１０１０１０１１Ｙ，１１１０１０１１Ｙ

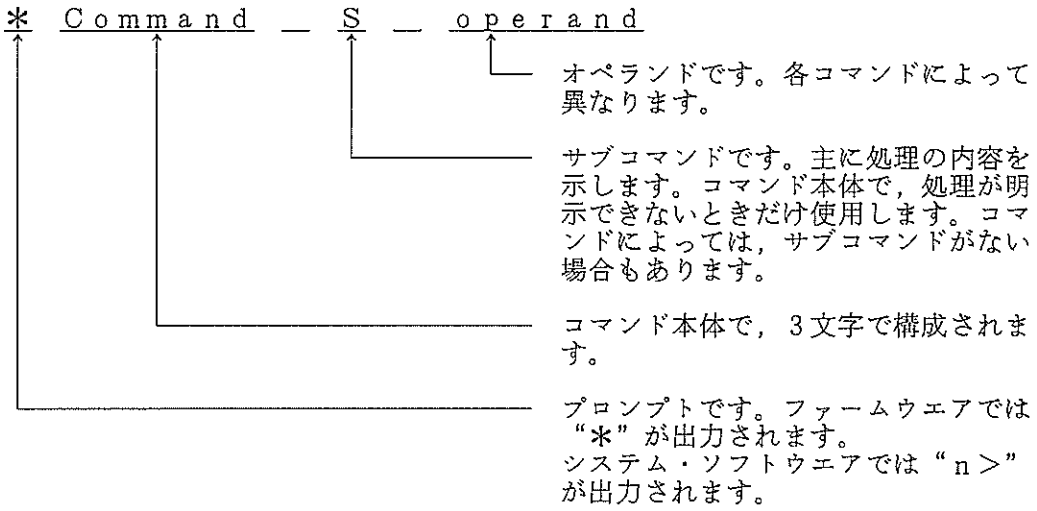
特殊数値表現が‘Ｘ’で始まる場合は，‘Ｘ’の前に数値であることを示すため‘０’を付けてください。‘Ｘ’から始まりますとシンボル表現となります。

４．４　コマンド一覧

４．４．１　コマンド形式

ＩＥ－７８３１０Ａ－Ｒのコマンド形式は，基本的には１ライン入力の形式になっています。ただし，‘ＴＲＸ’コマンド，‘ＢＲＡ’コマンド，‘ＭＯＤ’コマンドなどの設定が複雑なコマンドに関しては，１ライン入力だけでなくコマンド本体をキー入力するだけで，後のオペランドを対話形式で設定することができます。

コマンドの一般形式を以下に示します。



４．４．２　記述の説明

これ以降の説明で使用する記述を以下のように定めます。

英大文字	キー入力する文字列，または，文字そのものを表します。
英小文字	キー入力するオペランド要素名を表します。
＼	バック・スラッシュを表します。
{文字列}	｛　　｝の中に記述されている文字列のどれかを選ぶことを表します。
［文字列］	［　　］の中に記述されている文字列の入力が省略できることを表します。
_	スペース入力を表します。
（文字列）	（　　）の中に記述されている文字列は，サブコマンド，および，オペランドの説明です。

オペランド要素には以下に示すものがあります。

ｗｏｒｄ

数値表現の１６ビット数値です。通常，一つのアドレスを表現するのに使用されます。アドレス表現の他には，１６ビット・レジスタの値の表現，あるいは，タイマ値の表現などに使用されます。

ｂｙｔｅ

数値表現の８ビット数値です。８ビット・レジスタ値の表現，ループ・カウント値の表現，あるいは，ステップ数などに使用されます。

ｐａｒｔｉｔｉｏｎ

ｐａｒｔｉｔｉｏｎは，一つのアドレス範囲を示します。ｐａｒｔｉｔｉｏｎには，次の２つの表現方法があります。

ｐａｒｔｉｔｉｏｎ＝
- ｗｏｒｄ，ｗｏｒｄ（１６ビット数値を‘，’で区切って記述する。）
- 特殊数値表現

たとえば，１００Ｈ～１ＦＦＨのｐａｒｔｉｔｉｏｎを二つの表現方法で記述すると，次のようになります。

１００Ｈ～１ＦＦＨ
- １００Ｈ，１ＦＦＨ
- １ＸＸＨ

‘ｗｏｒｄ，ｗｏｒｄ’で記述する場合は，‘スタート番地，エンド番地’の形式で入力します。スタート番地≦エンド番地でなければなりません。

ｍａｓｋ

ｍａｓｋは，８ビット数値，または，８ビット・マスク・データのことです。‘ＢＲＡ’コマンドのブレーク・データ値の表現，あるいは，‘ＢＲＤ’コマンドのブレーク・データ値の表現，あるいは，‘ＲＵＮ’コマンドの８ビット・レジスタ値の表現，クオリファイ・ポート／外部データ値の表現に使用されます。

複数の８ビット・データを表現する場合は，８ビット・マスク・データを記述します。一つの８ビット・データを表現する場合は，８ビット数値を記述します。

ｗｍａｓｋ

ｗｍａｓｋは，１６ビット数値，または，１６ビット・マスク・データのことです。‘ＲＵＮ’コマンドの１６ビット・レジスタ値の表現に使用されます。

複数の１６ビット・データを表現する場合は，１６ビット・マスク・データを記述します。一つの１６ビット・データを表現する場合は，１６ビット数値を記述します。

ａｄｄｒｓ

ａｄｄｒｓは，ｗｏｒｄ，あるいは，ｐａｒｔｉｔｉｏｎを合計五つまでまとめた複数の１６ビット・データの集まりを表現します。

‘ＢＲＡ’コマンドのブレーク・アドレス表現，‘ＴＲＸ’コマンドのクオリファイ・アドレス表現に使用します。

ａｄｄｒｓは，次のような構成になっています。

addrs ＝ word(あるいは partition)[_word(あるいは partition)][・・・
合計　五つまで

ｄａｔａ　ｓｔｒｉｎｇ

ｄａｔａ　ｓｔｒｉｎｇは，複数のｂｙｔｅデータの集まりを表現します。

‘MEM_F’コマンドのイニシャライズ・データの集まりの表現と，

‘MEM_G’コマンドのサーチ・データの集まりの表現に使用されます。

ｄａｔａ　ｓｔｒｉｎｇ＝　ｂｙｔｅ，ｂｙｔｅ，・・・・・・，ｂｙｔｅ
最大１０個

ｃｏｍｍａｎｄ

‘ＨＬＰ’コマンドで，説明を表示するコマンドを指定する表現です。コマンド本体を記述します。

たとえば，

ＨＬＰ_ＲＵＮ

上記のように記述した場合は，‘ＲＵＮ’コマンドの説明が表示されます。

ｒｅｇｉｓｔｅｒ　ｎａｍｅ

レジスタ指定の表現です。‘ＲＥＧ’コマンド，あるいは，‘ＲＵＮ_Ｔ’コマンドのレジスタ表現ができます。

ｒｅｇｉｓｔｅｒ　ｎａｍｅには以下のものがあります。

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）
Ｒ０，Ｒ１，Ｒ２，Ｒ３，Ｒ４，Ｒ５，Ｒ６，Ｒ７	（８ビット・ジェネラル・レジスタ）
ＲＰ４，ＲＰ５，ＲＰ６，ＲＰ７	（１６ビット・ジェネラル・レジスタ）
Ｘ，Ａ，Ｂ，Ｃ	（８ビット・インプライド・レジスタ）
ＶＰ，ＵＰ，ＤＥ，ＨＬ	（１６ビット・インプライド・レジスタ）

これ以外に，ＰＳＷのビットを直接表現することもできます。

ＲＳＢ２，ＲＳＢ１，ＲＳＢ０	（レジスタ・バンク選択フラグ）
ＩＥ	（割込み許可フラグ）
Ｓ	（サイン・フラグ）
Ｚ	（ゼロ・フラグ）
ＲＳＳ	（レジスタ・セット選択フラグ）
ＡＣ	（ハーフ・キャリ）
ＵＦ	（ユーザ・フラグ）
Ｐ／Ｖ	（パリティ／オーバフロー・フラグ）
ＳＵＢ	（減算フラグ）
ＣＹ	（キャリ・フラグ）

ｍｏｄｅ　ｒｅｇｉｓｔｅｒ　ｎａｍｅ

‘ＭＤＲ’コマンドのモード・レジスタ指定と，‘ＢＲＡ’コマンド，あるいは，‘ＴＲＸ’コマンドのａｄｄｒｓ表現の代わりにモード・レジスタをブレーク，あるいは，クオリファイ条件に指定する場合に使用します。

ａｄｄｒｓの代わりにモード・レジスタを使用する場合は，データ条件を指定してＡＮＤをかけることはできません。このような指定をした場合は，正常なブレーク，あるいは，クオリファイをすることはできません。

ｍｏｄｅ　ｒｅｇｉｓｔｅｒ　ｎａｍｅには，以下のものがあります。

ＰＭ０，ＰＭ１，ＰＭ２，ＰＭ３，ＰＭ５，ＰＭＣ２，ＰＭＣ３，ＲＴＰＣ，ＭＭ，ＲＦＭ，ＷＤＭ，ＳＴＢＣ，ＴＢＭ，ＩＮＴＭ，ＩＳＰＲ，ＣＣＷ，ＳＣＭ，ＳＣＣ，ＦＲＣＣ，ＣＰＴＭ，ＰＷＭＭ，ＡＤＭ，ＣＵＩＭ，ＵＤＣＣ０，ＣＲＣ，ＵＤＣＣ１，ＴＭＣ０，ＴＭＣ１，ＣＲＩＣ００，ＣＲＭＳ００，ＣＲＩＣ０１，ＣＲＩＣ１０，ＣＲＭＳ１０，ＣＲＩＣ１１，ＥＸＩＣ０，ＥＸＭＳ０，ＥＸＩＣ１，ＥＸＭＳ１，ＥＸＩＣ２，ＥＸＭＳ２，ＴＭＩＣ０，ＴＭＭＳ０，ＴＭＩＣ１，ＴＭＭＳ１，ＴＭＩＣ２，ＴＭＭＳ２，ＳＥＩＣ，ＳＲＩＣ，ＳＲＭＳ，ＳＴＩＣ，ＳＴＭＳ，ＡＤＩＣ，ＡＤＭＳ，ＴＢＩＣ

これらのモード・レジスタは，すべてリード／ライト可能な８ビット・レジスタです。

ｓｐｅｃｉａｌ　ｒｅｇｉｓｔｅｒ　ｎａｍｅ

‘ＳＰＲ’コマンドのスペシャル・レジスタ指定と，‘ＢＲＡ’コマンド，および，‘ＴＲＸ’コマンドのａｄｄｒｓ表現の代わりにスペシャル・レジスタをブレーク，あるいは，クオリファイ条件に指定する場合に使用します。

ａｄｄｒｓの代わりにスペシャル・レジスタを使用する場合は，データ条件を指定してＡＮＤをかけることはできません。このような指定をした場合は，正常なブレーク，あるいは，クオリファイをすることはできません。

リード／ライト可能なｓｐｅｃｉａｌ　ｒｅｇｉｓｔｅｒ　ｎａｍｅには，以下のものがあります。＊が付いているものは１６ビット・レジスタです。

Ｐ０，Ｐ１，Ｐ２，Ｐ３，Ｐ４，Ｐ５，ＣＲ００＊，ＣＲ０１＊，ＣＲ１０＊，
ＣＲ１１＊，ＣＰＴ０＊，ＣＰＴ１＊，ＰＷＭ０＊，ＰＷＭ１＊，ＵＤＣ０＊，ＵＤＣ１＊，
Ｐ０Ｌ，Ｐ０Ｈ，ＢＲＧ，ＴＭ０＊，ＭＤ０＊，ＴＭ１＊，ＭＤ１＊
ＥＸＴＳＦＲ０，ＥＸＴＳＦＲ１，ＥＸＴＳＦＲ２，ＥＸＴＳＦＲ３，
ＥＸＲＳＦＲ４，ＥＸＴＳＦＲ５，ＥＸＴＳＦＲ６，ＥＸＴＳＦＲ７，
ＥＸＴＳＦＲ８，ＥＸＴＳＦＲ９，ＥＸＴＳＦＲ１０，ＥＸＴＳＦＲ１１，
ＥＸＴＳＦＲ１２，ＥＸＴＳＦＲ１３，ＥＸＴＳＦＲ１４，ＥＸＴＳＦＲ１５

リード・オンリのｓｐｅｃｉａｌ　ｒｅｇｉｓｔｅｒ　ｎａｍｅを以下に示します。すべて８ビット・レジスタです。

ＲＸＢ，ＡＤＣＲ

ライト・オンリのｓｐｅｃｉａｌ　ｒｅｇｉｓｔｅｒ　ｎａｍｅを以下に示します。すべて８ビット・レジスタです。

ＴＸＢ

ｓｙｍｂｏｌ

４．３のシンボル表現で述べているシンボルのことです。ｓｙｍｂｏｌの代わりに数値を入力することはできません。

ｓｙｍｂｏｌは，‘ＳＹＭ’コマンドでだけ使用されます。

ｄ：

ドライブ番号を示します。省略された場合は，カレント・ディスクが選択されます。

ｆｉｌｅ

ファイル名を示します。ファイル名は，８文字のファイル名と３文字の拡張子で構成されます。ファイル名，および，拡張子には，＜＞．，；：＝？＊［］を除く特殊記号，英数字，英文字が使用できます。ただし，‘ＤＩＲ’コマンドでは，？　＊　が使用できます。

なお，コマンドによっては拡張子が省略できる場合があります。

ｍｏｄｕｌｅ　ｎａｍｅ

シンボル・テーブル・ファイルから読込まれたモジュール名を表します。パブリック・シンボルに対しては記述することはできません。

４．４．３　コマンド一覧

コマンド一覧表を示します。

表のコマンド本体のところに＊が書いてあるコマンドは，システム・ソフトウエア使用時だけ有効です。

また，＊＊が書いてあるコマンドは，スタンドアロン動作時だけ有効なコマンドです。

コマンド種類	コマンド本体	サブコマンド	オペランド
ライン・アセンブラ	ＡＳＭ	なし	[word]（ｱｾﾝﾌﾞﾗ のｽﾀｰﾄ･ｱﾄﾞﾚｽ）
物理ブレーク条件設定 ハードウエア・ ブレーク条件設定	ＢＲＡ	なし	[A=addrs][V=mask]（ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ）（ﾌﾞﾚｰｸ･ﾃﾞｰﾀ） [C= OP（ｵﾍﾟﾗﾝﾄﾞ･ﾌｪｯﾁ） RW（ﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） R（ﾃﾞｰﾀ･ﾘｰﾄﾞ） W（ﾃﾞｰﾀ･ﾗｲﾄ） RWP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） RP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ） WP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾗｲﾄ） RWM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） RM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾘｰﾄﾞ） WM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾗｲﾄ） NC（ｵﾍﾟﾗﾝﾄﾞ･ﾌｪｯﾁを含むすべてのﾘｰﾄﾞ/ﾗｲﾄ）] （ﾌﾞﾚｰｸ･ｽﾃｰﾀｽ） [L=byte]（ﾙｰﾌﾟ･ｶｳﾝﾄ数） （各オペランドは＿で区切って入力する）
外部信号ブレーク 条件設定	ＢＲＤ	なし	[mask]（外部ｾﾝｽ 信号のﾌﾞﾚｰｸ･ﾃﾞｰﾀ）
ｲﾝｽﾄﾗｸｼｮﾝ･ｶｳﾝﾄ･ﾌﾞﾚｰｸ 条件設定	ＢＲＥ	なし	[word]（ﾌｪｯﾁ命令数）
タイマ・ブレーク 条件設定	ＢＲＴ	なし	[word]（実行時間。単位は10μs）
論理ブレーク条件設定	ＢＲＭ	なし	[BRA][BRD][BRE][BRT][BR0][BR1][BR2][BR3]（ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ名）
	ＢＲ０ ＢＲ１ ＢＲ２ ＢＲ３	なし	[BRA][BRD][BRE][BRT]（物理ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ 名）
クロック選択	ＣＬＫ	[U(ﾕｰｻﾞ) / I(IE)]	なし （U:ﾕｰｻﾞ･ｼｽﾃﾑのｸﾛｯｸ，I:IE内部のｸﾛｯｸ）

コマンド種類	コマンド本体	サブコマンド	オペランド
コマンド・ファイル作成	COM ＊	なし	[{ LST: / CON: / file(コマンド・ファイル名) }]
逆アセンブラ	DAS	なし	[{ word (逆アセンブラ のスタート・アドレス) / partition(逆アセンブラ のスタート・アドレス とエンド・アドレス) }]
自己診断	DIG	なし	なし
ディレクトリ表示	DIR ＊	なし	[file] (ファイル名)
システム・モード終了	EXT ＊	なし	なし
コマンド・ヒストリ表示	HIS ＊	なし	なし
ヘルプ	HLP ＊	なし	[command] (表示したいコマンドのコマンド本体)
オブジェクト・ロード	LOD＊＊	なし	[{ TTY1(チャネル1) / TTY2(チャネル2) }]
オブジェクト／シンボル・ロード	LOD ＊	なし	file(オブジェクト/シンボル・ファイル名)[module name(モジュール名)\……] [{ C(オブジェクト指定) / S(シンボル指定) }]
出力ディバイス・リダイレクト	LST ＊	なし	[{ LST: / CON: / file (出力ファイル名) }]
マッピング	MAP	[{ W / R / U / K }]	[partition] (マッピング範囲) (W:内部マッピング , R:ライト・プロテクト 内部マッピング , U:ユーザ・マッピング , K:マッピング 解除)
演算	MAT	なし	word (通常は式を記述する)

コマンド種類	コマンド本体	サブコマンド	オペランド
モード・レジスタ操作	MDR	[D](表示)	[mode register name]
		C(変更)	[mode register name]
メモリ操作	MEM	C(変更)	[word] (変更スタート・アドレス)
		[D](表示)	[word (表示スタート・アドレス) partition (表示スタート・アドレス と表示エンド・アドレス)]
		F(イニシャライズ)	partition_data string → (イニシャライズ・データ(8ビット) の集まり) → (イニシャライズ・スタート・アドレス とエンド・アドレス)
		G(サーチ)	partition_data string → (サーチ・データ(8ビット) の集まり) → (サーチ・スタート・アドレス とエンド・アドレス)
		M(コピー)	partition_word → (コピー 先スタート・アドレス) → (コピー 元スタート・アドレス とエンド・アドレス)
		X(交換)	partition_word → (交換先スタート・アドレス) → (交換元スタート・アドレス とエンド・アドレス)
		V(比較)	partition_word → (比較先スタート・アドレス) → (比較元スタート・アドレス とエンド・アドレス)
		E(テスト)	[partition] (テスト・スタート・アドレスとエンド・アドレス)

コマンド種類	コマンド本体	サブコマンド	オペランド
チャネル２モード設定	ＭＯＤ	なし	[MODE= { CHAR / FLOW }] [BAUD= { 19200 / 9600 / 4800 / 2400 / 1200 / 600 / 300 }] [LONG= { 7 / 8 }] [PAR= { NON / EVEN / ODD }] [STOP= { 1 / 2 }] (ﾊﾝﾄﾞｼｪｲｸ･ﾓｰﾄﾞ)　(ﾎﾞｰﾚｰﾄ)　(ｷｬﾗｸﾀ長)　(ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ)　(ｽﾄｯﾌﾟ･ﾋﾞｯﾄ 長)
内部→ユーザ／ ユーザ→内部 メモリ転送	ＭＯＶ	{ U / I }	partition_word →(ｺﾋﾟｰ先ｽﾀｰﾄ･ｱﾄﾞﾚｽ) →(ｺﾋﾟｰ元ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ) (U:内部→ﾕｰｻﾞ , I:ﾕｰｻﾞ →内部)
端末モード	ＰＧＭ	なし	なし
レジスタ操作	ＲＥＧ	C(変更)	[register name]
		[D](表示)	[register name]　ALLを指定すると全ﾚｼﾞｽﾀ･ﾊﾞﾝｸの内容を表示します。
エミュレーション操作	ＲＵＮ	N	[word] (実行ｽﾀｰﾄ･ｱﾄﾞﾚｽ)　(N:ﾌﾞﾚｰｸなしﾘｱﾙﾀｲﾑ実行)
		B	[word] (実行ｽﾀｰﾄ･ｱﾄﾞﾚｽ)　(B:ﾌﾞﾚｰｸ付きﾘｱﾙﾀｲﾑ実行)
		S	[word][,word]　(S:ｽﾃｯﾌﾟ数指定ﾘｱﾙﾀｲﾑ実行) →(ｽﾃｯﾌﾟ 数) →(実行ｽﾀｰﾄ･ｱﾄﾞﾚｽ)
		T	[word] [, { ※ / word } [_TRD][_REG]] →(ﾚｼﾞｽﾀ 表示指定) →(ﾄﾚｰｽ 表示指定) →(ﾌﾞﾚｰｸ 条件, ※はﾚｼﾞｽﾀ条件, word はｽﾃｯﾌﾟ数) ※･･･････ { = / > / < / => / >= / =< / <= / >< / <> } register name　{ mask / wmask } (T:ﾄﾚｰｽ 実行)

コマンド種類	コマンド本体	サブコマンド	オペランド
リセット	ＲＥＳ	[H]	なし (H: 省略時はエバチップ だけのリセット。指定時はIEすべてのリセット。)
オブジェクト・セーブ	ＳＡＶ＊＊	なし	{TTY1(チャネル1) / TTY2(チャネル2)} [_partition][_partition] ・・・・ [_partition] 最大五つまで
	ＳＡＶ ＊	なし	file (入力ファイル名)[_partition][_partition] ・・・・ [_partition] 最大五つまで
特殊レジスタ操作	ＳＰＲ	C(変更)	[special register name]
		[D](表示)	[special register name]
入力ディバイス・リダイレクト	ＳＴＲ ＊	なし	file (入力ファイル名)
サフィックス指定	ＳＵＦ	なし	[{H(16進) / T(10進) / Q(8進) / Y(2進)}]
アペンド・シンボル操作	ＳＹＭ	[D](表示)	なし
		K(削除)	なし(すべてのアペンド・シンボル を削除)
		A(アペンド)	symbol_word →(シンボル 値) →(アペンド・シンボル名)
		C(変更)	symbol_word →(変更シンボル値) →(アペンド・シンボル名)
		E(削除)	[symbol] →(削除するアペンド・シンボル 名)

コマンド種類	コマンド本体	サブコマンド	オペランド
アペンド・シンボル操作	ＳＹＭ　＊	L(ロード)	なし
		S(セーブ)	なし
シンボル操作	ＳＹＭ　＊	[D](表示)	[{ module＼(モジュール指定) / PUBLIC　(パブリック指定) }]
カレント・モジュールの変更	ＳＹＭ	M(変更)	なし
トレース・モード設定	ＴＲＭ	なし	[{ NON(ノン・トレース) / ALL(全トレース) / TRX(クオリファイ・トレース) }]
クオリファイ条件設定	ＴＲＸ	なし	[A=addrs](クオリファイ・アドレス)[V=mask](クオリファイ・データ) [C= { OP (オペランド・フェッチ) / RW (データ・リード/ライト) / R (データ・リード) / W (データ・ライト) / RWP(プログラムによるデータ・リード/ライト) / RP (プログラムによるデータ・リード) / WP (プログラムによるデータ・ライト) / RWM(マクロ・サービス によるデータ・リード/ライト) / RM (マクロ・サービス によるデータ・リード) / WM (マクロ・サービス によるデータ・ライト) / NC(オペランド・フェッチを含むすべてのリード/ライト) } (クオリファイ・ステータス)] [TRQ=mask](クオリファイ・ポート/外部データ)
クオリファイ・ データ選択	ＴＲＱ	なし	[{ TRS(ポート 選択) / EXT(外部信号選択) }]

コマンド種類	コマンド本体	ｻﾌﾞｺﾏﾝﾄﾞ	オペランド
トレース／クオリファイ ポート選択	ＴＲＳ	なし	[{ P0 (ﾎﾟｰﾄ0) P1 (ﾎﾟｰﾄ1) P2 (ﾎﾟｰﾄ2) P3 (ﾎﾟｰﾄ3) P4 (ﾎﾟｰﾄ4) P5 (ﾎﾟｰﾄ5) }]
トレース・ポインタ操作	ＴＲＰ	なし	[{ word (ﾎﾟｲﾝﾀ移動数) 0 (ﾎﾟｲﾝﾀ を先頭に置く) N (ﾎﾟｲﾝﾀ を最後に置く) }]
トレース表示	ＴＲＤ	[{ F I M }]	[{ word (表示ｽﾃｯﾌﾟ数) ALL (ﾄﾚｰｽ結果すべてを表示) }] (F:ﾌﾚｰﾑ･ﾓｰﾄﾞ , I:ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ , M:ﾏｸﾛ･ｻｰﾋﾞｽ付きｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ)
オブジェクト・ ベリファイ	ＶＲＹ＊＊	なし	{ TTY1(ﾁｬﾈﾙ1) TTY2(ﾁｬﾈﾙ2) }
	ＶＲＹ　＊	なし	file (入力ﾌｧｲﾙ名)

4．5　コマンドの説明

コマンドの詳細について，説明します。記述されている数値は特にことわらないかぎり16進数（H）です。また，＿＿＿＿（下線）部分は，キー・ボードからの入力を表します。

4．5．1　ライン・アセンブラ（ASM）

ASM[_word]

word：変更開始アドレス

wordで指定されたアドレスからメモリの内容をニモニックで変更することができます。wordが省略された場合は，前回アセンブルした次のアドレスが自動的に指定されます。

アセンブラ仕様について詳しくは，4．7　オンライン・アセンブラ／逆アセンブラ仕様を参照してください。

例　100H番地からのﾒﾓﾘ 内容を変更する場合

```
*ASM 100H↵
0100                    MOV    SW,#0H  ←内容が逆ｱｾﾝﾌﾞﾙ されて表示されます。
                  = MOVW SP,#STCK ↵ ←ﾒﾓﾘ 内容の変更
      0B FC 7F FE                     ←変更後のｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞ
0104                    NOP
                  = MOVW HL,#WORK1 ↵
      Caution!                        ←警告ﾒｯｾｰｼﾞ
      67 00 10
0107                    NOP
                  = MOV R0,#02 ↵
      B8 02
0109                    NOP
                  = MOV A,R0 ↵
      D0
010A                    NOP
                  = MOV [HL+],A  ↵
      51
010B                    NOP
                  = DEC R0 ↵
      C8
010C                    NOP
                  = MOVW TMOH,#1000H ↵
    Warning!   Generate code? (Y/N) Y
    0B 20 00 0F
0110                    NOP
                  = END ↵
*
```

・ASM コマンドで変更可能なメモリ範囲は，　0～0FE7FHまでです。

・Error の場合は，次のようにメッセージを表示して，もう一度入力待ちになります。

```
Error!
```

Error が表示された場合は，オブジェクト・コードを生成することができません。ニモニック，オペランド，あるいは，予約語に間違いがある場合に表示されます。

・Warning の場合は，次のように表示してオブジェクト・コードを生成するかどうかを聞いてきます（例を参照）。

```
Warning!   Generate code? (Y/N)
```

Y を入力しますと，オブジェクト・コードを表示してメモリの内容を変更します。そして，次のニモニックの入力ができます。Y 以外のキーを入力しますと，もう一度，同じアドレスの内容の変更ができます。

Warning が表示された場合は，オブジェクトの生成はできますが正しい動作はのぞめません。

・Caution の場合は，次のようなメッセージを表示し，オブジェクト・コードを生成します（例を参照）。

```
Caution!
```

Caution はジェネリックなオブジェクトが生成された場合（たとえば，ジェネリック・ブランチ，ｒｐ１とｒｐ２を置換えた場合）に，表示されます。

４．５．２　ブレーク条件指定コマンド（ＢＲＭ）

```
ＢＲＭ[_BRA][_BRD][_BRE][_BRT][_BRO][_BR1][_BR2][_BR3]
```

‘ＢＲＭ’コマンドは，物理ブレーク条件，および，論理ブレーク条件を選択し指定できます。‘ＲＵＮ＿Ｂ’コマンドは，‘ＢＲＭ’コマンドでブレーク条件を設定しなければ，ブレークしません。ブレーク条件は，スペースで区切って複数の条件を設定することができます。設定された条件は，ＯＲ条件となります。

また，オペランドが省略された場合は，現在設定されているブレーク条件を表示します。

例

```
*BRM BRE BRT BR2↵   ←ブレーク条件にBRE,BRT,BR2 を指定します。
*
*BRM↵               ←設定されているﾌﾞﾚｰｸ条件を表示します。
 BRE BRT BR2        ←設定されている条件が表示されます。
*
```

ブレーク・コマンドは，物理ブレーク条件設定コマンドと論理ブレーク条件設定コマンドの二つがあります。

物理ブレーク条件設定コマンドは，ハードウエア・ブレーク条件，外部信号ブレーク条件，インストラクション・カウント・ブレーク条件，タイマ・ブレーク条件を設定することができます。

論理ブレーク条件設定コマンドは，物理ブレーク条件設定コマンドで設定された条件を選択し組合わせてブレーク条件を設定することができます。

物理ブレーク条件設定コマンド

ＢＲＡ[_A=addrs][_V=mask][_C=ｽﾃｰﾀｽ][L=byte]	(ﾊｰﾄﾞｳｪｱ･ﾌﾞﾚｰｸ 条件設定)
ＢＲＤ[_mask]	(外部信号ﾌﾞﾚｰｸ条件設定)
ＢＲＥ[_word]	(ｲﾝｽﾄﾗｸｼｮﾝ･ｶｳﾝﾄ･ﾌﾞﾚｰｸ条件設定)
ＢＲＴ[_word]	(ﾀｲﾏ･ﾌﾞﾚｰｸ条件設定)

論理ブレーク条件設定コマンド

```
ＢＲ０[_BRA][_BRD][_BRE][_BRT]
ＢＲ１[_BRA][_BRD][_BRE][_BRT]
ＢＲ２[_BRA][_BRD][_BRE][_BRT]
ＢＲ３[_BRA][_BRD][_BRE][_BRT]
```

４．５．３　ハードウエア・ブレーク条件設定（ＢＲＡ）

```
ＢＲＡ[_A=addrs][_V=mask][_C=ｽﾃｰﾀｽ][_L=byte]
```

A=addrs ：ブレーク・アドレスを設定します。ブレーク・アドレスは，五つまでスペースで区切って設定することができます。省略された場合は，条件として設定されません。

V=mask ：ブレーク・データを設定します。省略された場合は，条件として設定されません。

C=ｽﾃｰﾀｽ ：ブレーク・アドレスと，ブレーク・データの組合わせ条件を設定します。条件を下記から選択し設定します。省略した場合は，条件として‘NC’が選択されます。

```
OP   :ｵﾍﾟｺｰﾄﾞ･ﾌｪｯﾁ
RW   :ﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ
R    :ﾃﾞｰﾀ･ﾘｰﾄﾞ
W    :ﾃﾞｰﾀ･ﾗｲﾄ
RWP  :ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ
RP   :ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ
WP   :ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾗｲﾄ
RWM  :ﾏｸﾛ･ｻｰﾋﾞｽ によるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ
```

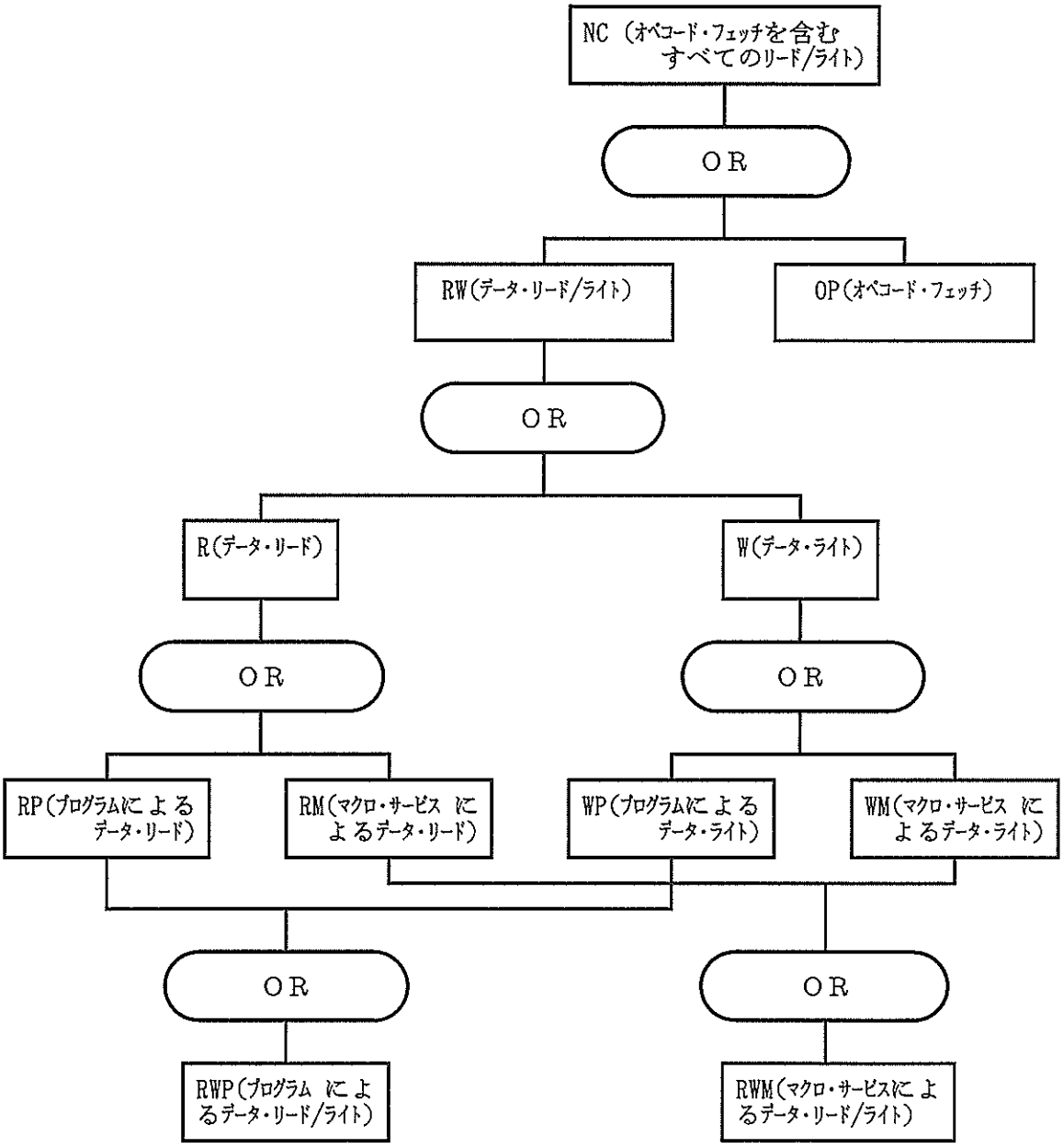
RM　：ﾏｸﾛ･ｻｰﾋﾞｽ によるﾃﾞｰﾀ･ﾘｰﾄﾞ
WM　：ﾏｸﾛ･ｻｰﾋﾞｽ によるﾃﾞｰﾀ･ﾗｲﾄ
NC　：ｵﾍﾟｺｰﾄﾞ･ﾌｪｯﾁ を含むすべてのﾘｰﾄﾞ/ﾗｲﾄ

L=byte　：ループ・カウント数を設定します。ループ・カウント数は，ブレーク・アドレス，ブレーク・データ，ブレーク・ステータスの組合わせに対する繰返し回数を設定することができます。１０進数の１～２５５までを設定することができます。省略された場合は，ループ・カウント数に１が設定されます。

‘ＢＲＡ’コマンドは，ハードウエアにブレーク条件を設定するコマンドです。ハードウエア・ブレーク条件は，‘ＢＲＭ’コマンドで指定（ＢＲ０，ＢＲ１，ＢＲ２，および，ＢＲ３コマンドでの指定も含む）されていなければ，‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

‘ＢＲＡ’コマンドのオペランドを省略しますと，ハードウエアへのブレーク条件設定を対話形式ですることができます。対話形式でブレーク条件を設定する場合は，現在設定されているブレーク条件を表示します。もし，条件が設定されていない場合は，‘－－’を表示します。

図４－１　ＢＲＡコマンド・ステータス構成



例

*BRA A=0,1FFF 3000 C=OP ↵　　0 ～1FFF番地と3000番地でオペランドをフェッチした場合にブレークする条件を設定します。

*BRA V=11 C=W L=10T ↵　　データ‘11’を１０回データ・ライトした場合にブレークする条件を設定します。

```
*BRA ↵                          ←対話形式でﾌﾞﾚｰｸ条件を設定する場合のｺﾏﾝﾄﾞ
 A --- = 0XXXH ↵                ←現在ｱﾄﾞﾚｽは設定されていないので，0XXXH を設定します。
 V 11 = ↵                       ←現在のﾃﾞｰﾀ を変更しません。
   OPcode fetch                  (OP)
   Read Write                    (RW)
   Read                          (R)
   Write                         (W)
   Read Write by program         (RWP)
   Read by program               (RP)
   Write by program              (WP)
   Read Write by Macro service   (RWM)
   Read by Macro service         (RM)
   Write by Macro service        (WM)
   No Condition                  (NC)
 C W = NC ↵                      ←現在のｽﾃｰﾀｽ を新しく‘NC’に設定します。
 L 10T = 100H ↵                 ←ﾙｰﾌﾟ･ｶｳﾝﾄ数に100Hを設定します。
 Input data error                ←ﾙｰﾌﾟ･ｶｳﾝﾄ数は 1～255 までなのでｴﾗｰ になります。
 L 10T = 0FFH ↵                 ←ｴﾗｰ になったので再度ﾙｰﾌﾟ･ｶｳﾝﾄ数を0FFHに設定します。

*
```

４．５．４　外部信号ブレーク条件設定コマンド（ＢＲＤ）

ＢＲＤ［_mask ］

mask：外部センス信号のブレーク・データを設定します。

‘ＢＲＤ’コマンドは，外部センス・クリップ８本のデータでブレークする条件を設定します。外部信号ブレーク条件は，‘ＢＲＭ’コマンドで指定（ＢＲ０，ＢＲ１，ＢＲ２，および，ＢＲ３コマンドでの指定を含む）されていなければ，‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

また，オペランドが省略された場合は，現在の設定されている条件を表示します。

例

```
*BRD 0XXXX000XY ↵     ←外部ｾﾝｽ･ｸﾘｯﾌﾟの EA1～EA3 が 000をﾌﾞﾚｰｸ条件に設定します。
*
*BRD ↵                ←現在の条件を表示する場合のｺﾏﾝﾄﾞ
 0XXXX000XY           ←現在のブレーク条件が表示されます。


*
```

４．５．５　インストラクション・カウント・ブレーク条件設定コマンド(ＢＲＥ)

```
ＢＲＥ [_word ]
```

word：フェッチ命令数を設定します。１ ～ 65535まで設定できます。

‘ＢＲＥ’コマンドは，フェッチした命令数でブレークする条件を設定します。インストラクション・カウント・ブレーク条件は，‘ＢＲＭ’コマンドで指定（ＢＲ０，ＢＲ１，ＢＲ２，および，ＢＲ３コマンドでの指定を含む）されていなければ，‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

また，オペランドが省略された場合は，現在の設定されている条件を表示します。

例

```
*BRE 10H )          ←ﾌｪｯﾁ命令数に 10Hを設定します。
*
*BRE )              ←設定されている条件を表示します。
 10H                ←現在のフェッチ命令数が表示されます。
*
```

注意　μPD78312A はI-Bufferを持ち，プリ・フェッチをします。このため，ブランチ系の命令が多いプログラムでは，実行ステップ数とフェッチ数が大きく異なる場合があります。

４．５．６　タイマ・ブレーク条件設定コマンド（ＢＲＴ）

ＢＲＴ［_word］

word：実行時間（単位は10μs)を設定します。10μs ～ 655msまで設定することができます。

‘ＢＲＴ’コマンドは，実行時間でブレークする条件を設定します。タイマ・ブレーク条件は，‘ＢＲＭ’コマンドで指定（ＢＲ０，ＢＲ１，ＢＲ２，および，ＢＲ３コマンドでの指定を含む）されていなければ，‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

また，オペランドが省略された場合は，現在の条件を表示します。

例

```
*BRT 1000T↵          ←実行時間に10msを設定します。
*
*BRT↵                 ←設定されている条件を表示します。
 1000T                ←現在の実行時間が表示されます。
*
```

４．５．７　論理ブレーク条件設定コマンド（ＢＲ０～３）

ＢＲ０[_BRA][_BRD][_BRE][_BRT]

オペランドの形式は，‘ＢＲ０’コマンド，‘ＢＲ１’コマンド，‘ＢＲ２’コマンド，および，‘ＢＲ３’コマンドとも同一です。

‘ＢＲ０’コマンド，‘ＢＲ１’コマンド，‘ＢＲ２’コマンド，および，‘ＢＲ３’コマンドは，物理ブレーク条件設定コマンドで設定された条件を指定することができます。条件の指定は，スペースで区切って複数を設定することができます。指定された条件は，ＯＲ条件となります。‘ＢＲＭ’コマンドで指定されなければ，‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

また，オペランドが省略された場合は，現在，指定されている条件を表示します。

例

```
*BR0 BRA BRD BRE BRT↵   ←ＢＲ０に物理ブレーク条件を設定します。
*
*BR0↵                   ←設定されている条件を表示します。
 BRA BRD BRE BRT        ←現在，設定されている条件が表示されます。
*
```

４．５．８　クロック選択（ＣＬＫ）

```
CLK [_ {U} ]
       {I}
```

Ｕ：ユーザ・システム側のクロックを選択

Ｉ：ＩＥ側のクロックを選択

クロック・ソースを選択します。‘U’が指定された場合は，ユーザ・システム側のクロックが選択されます。また，‘I’が指定された場合は，ＩＥ側のクロックが選択されます。クロック・ソース選択後は，必ずＲＥＳコマンドを使用してエバリュエーション・チップをリセットしてください。

なお，オペランドが省略された場合は，現在選択されているクロック・ソース名を表示します。

ＩＥ－７８３１０Ａ－Ｒの起動時は，ＩＥ側のクロックが選択されています。

例

```
*CLK U↵                ←ユーザ・システム側のクロックを選択します。
*
*CLK I↵                ←ＩＥ側のクロックを選択します。
*
*CLK↵                  ←選択されているクロックを表示します。
 IE                    ←ＩＥ側のクロックが選択されている場合の表示です。
*
*CLK↵
 User                  ←ユーザ・システム側のクロックが選択されている場合
*                        の表示です。
```

４．５．９　コマンド・ファイル作成（ＣＯＭ）

```
COM [__ {file} ]
       {LST:}
       {CON:}
```

file　：ｺﾏﾝﾄﾞ･ﾌｧｲﾙ としてﾌｧｲﾙをｵｰﾌﾟﾝします。

LST:　：ｺﾏﾝﾄﾞ･ﾌｧｲﾙ としてﾘｽﾄ 装置をｵｰﾌﾟﾝします。

CON:　：ｺﾏﾝﾄﾞ･ﾌｧｲﾙ をｸﾛｰｽﾞします。

‘ＣＯＭ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

コマンド・ファイル作成コマンドは，このコマンド以降に入力されたコマンド，および，データをオペランドで指定された装置に出力するために，ファイル，あるいは，リスト装置をオープンするためのコマンドです。

実際の出力タイミングは，コマンド入力中の‘↑Ｏ’キーが入力されたコマンド行からファイル，あるいは，リスト装置に出力されます。ファイル，あるいは，リスト装置への出力の停止は，コマンド入力中の次の‘↑Ｏ’キーが入力された場合です。

例　ﾄﾞﾗｲﾌﾞB にSAMPLE.STRというﾌｧｲﾙをｵｰﾌﾟﾝします。

```
1>COM B:SAMPLE.STR ↵
1>
```

ﾄﾞﾗｲﾌﾞ Bには，すでにSAMPLE.STRというﾌｧｲﾙが正しくｵｰﾌﾟﾝできた場合です。

```
1>COM B:SAMPLE.STR ↵
  File already exists. Delete ? (Y or N): Y ↵
1>
```

ﾄﾞﾗｲﾌﾞ Bには，すでにSAMPLE.STRというﾌｧｲﾙが存在している場合に上記のようなﾒｯｾｰｼﾞ を出力します。この場合，すでに存在しているSAMPLE.STRというﾌｧｲﾙの属性が DIR属性で，かつ，R/W 属性です。ここで，Y を入力しますとSAMPLE.STRをﾃﾞﾘｰﾄして新しくSAMPLE.STRというﾌｧｲﾙをｵｰﾌﾟﾝします。Y 以外を入力しますとﾌｧｲﾙのｵｰﾌﾟﾝはしません。つまり，すでに存在しているﾌｧｲﾙは，そのままです。

```
1>COM B:SAMPLE.STR ↵
  File already exists.
1>
```

ﾄﾞﾗｲﾌﾞ Bに，すでにSAMPLE.STRというﾌｧｲﾙが存在している場合です。ただし，前記の例と異なり，SAMPLE.STRというﾌｧｲﾙの属性が SYS属性か，あるいは，R/O 属性の場合です。この場合は，ｺﾏﾝﾄﾞは無視されます。

```
1>COM LST: ↵
1>
```

ﾘｽﾄ 装置をｵｰﾌﾟﾝします。ﾎｽﾄ･ﾏｼﾝ が MD-086 ｼﾘｰｽﾞで CCP/Mの場合で，ﾘｽﾄ装置が他の処理によって使用中であったなら，下記のようなﾒｯｾｰｼﾞ が表示されます。

この場合はﾘｽﾄ 装置は，ｵｰﾌﾟﾝされません。

```
1>COM LST: ↵
  List device is used by other process.
1>
```

・ｺﾏﾝﾄﾞ･ﾌｧｲﾙ の作成する場合の例を以下に示します。

```
1>COM B:SAMPLE.STR ↵   ←ｺﾏﾝﾄﾞ･ﾌｧｲﾙ をﾄﾞﾗｲﾌﾞ BにSAMPLE.STRというﾌｧｲﾙ名でｵｰﾌﾟﾝ
                         します。
1>MAP ↵
  0000-02FF R/W   0300-07FF R/O   0800-0FFF R/W   1000-16FF Non
  1700-1FFF R/W   2000-2FFF R/O   3000-7FFF User  8000-FDFF Non
┌─1>MEM F 0,02FF 1,2,3,4,5,6,7,8 ↵  ←このｺﾏﾝﾄﾞ行で↑0 を入力します。
│ 1>MEM D 080X ↵
│   0800   01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08    ........
│ 1>BRA A=100 ↵
└─1>BRM BRA ↵
  1>TRM ALL ↵   ←このｺﾏﾝﾄﾞ行で↑0 を入力します。
  1>
```

└──→ SAMPLE.STR に出力されます。

この場合は，ﾌｧｲﾙをｵｰﾌﾟﾝ後の↑0 が入力された ｺﾏﾝﾄﾞ (MEM F 0,02FF 1, 2･･･) から次の↑0 が入力される前のｺﾏﾝﾄﾞ(BRM BRA) までがｺﾏﾝﾄﾞ･ﾌｧｲﾙ に出力されます。

４．５．１０　逆アセンブラ（ＤＡＳ）

```
DAS [_ {word      }]
       {partition }
```

word　　：逆アセンブラ開始アドレスを設定します。

partition：逆アセンブラ開始／終了アドレスを設定します。

‘ＤＡＳ’コマンドは，指定されたアドレスからメモリ内容をニモニックで表示することができます。

開始アドレスだけが指定された場合は，指定されたアドレスから１１行分のメモリ内容が表示されます。また，開始／終了アドレスが指定された場合は，指定された開始アドレスから，終了アドレスまでのメモリ内容が表示されます。

開始アドレス，および，終了アドレスが省略された場合は，前回の逆アセンブラで表示された次のアドレスから１１行分のメモリ内容が表示されます。

逆アセンブラの仕様については，４．７　オンライン・アセンブラ／逆アセンブラ仕様を参照してください。

例

```
*DAS 10XH↵
 Addr  Object             Mnemonic
                          ORG     MODULE00\START
                        MODULE00\START::
 0100  0B FC 7F FE        MOVW    SP,#STCK
 0104  67 00 01           MOVW    RP7,#WORK1
 0107  B8 02              MOV     R0,#2H
                        MODULE00\LOOP:
 0109  D0                 MOV     A,R0
 010A  51                 MOV     [HL+],A
 010B  C8                 DEC     R0
 010C  80 FB              BNZ     $LOOP
 010E  67 00 20           MOVW    RP7,#WORK2
                          END
*
```

注　‘ＤＡＳ’コマンドで表示できるアドレス範囲は，０～ＦＥ７ＦＨまでです。

４．５．１１　自己診断（ＤＩＧ）

ＤＩＧ

‘ＤＩＧ’コマンドは，自己診断をし，その結果を表示します。診断項目は，ポート０～５のテスト，アナログ・ポートのテスト，アドレス／データ・バスのテスト，ＥＡ／Ｖｐｐ線のテスト，および，リセット線のテストをします。

自己診断は，途中で異常を検出した場合でも全項目のテストをします。

例

```
*DIG )
 Self Diagnosis program

 Port0,Port1 test  --------  Good または No-Good
 Port2,Port3 test  --------  Good または No-Good
 Port4、Port5 test  -------  Good または No-Good
 Analog port test  --------  Good または No-Good
 Address,Data bus test  --   Good または No-Good
 EA/Vpp line test  --------  Good または No-Good
 Reset line test  ---------  Good または No-Good
 Self Diagnosis Complete
*
```

診断結果が，Goodの場合は正常です。No-Good が表示された場合は，異常が検出されていますので使用を中止してください。

４．５．１２　ディレクトリ表示（ＤＩＲ）

```
ＤＩＲ [_file]
```

file　：ディレクトリ表示ファイル名を設定します。

‘ＤＩＲ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

ディレクトリ参照コマンドは，ＣＰ／Ｍのビルトイン・コマンドのＤＩＲと同等の機能をもっています。指定されたドライブのディレクトリを表示します。

例

```
1>DIR )   ←カレント・ドライブ のすべてのディレクトリを表示します。
A: CONVPM   CMD : ABORT    CMD : ATTACH   CMD : DDT      COM
A: CONSOLE  CMD : DDT86    CMD : DIR      CMD : DSKRESET CMD
A: ERA      CMD : ERAQ     CMD : FORMAT   CMD : BACKUP   CMD
A: MPMSTAT  CMD : PIP      CMD : REN      CMD : SDIR     CMD
A: OBJERR   HEX : STAT     CMD : SUBMIT   CMD : TOD      CMD
A: TYPE     CMD : UNLOCKB  CMD : CXO      TXT : POW2     COM
A: ERROR2   STR : WMA      CMD : SET      CMD : GENCMD   CMD
A: WM       COM : WM       HLP : MPM      SYS : MPM      COM
A: KEY      COM : KEY      KEY : STAT     COM : CPM      CMD
A: WM       CMD : ZSID     COM : DUMP     COM : M80      COM
A: L80      COM : LOAD     COM : PIP      COM : DUMPA    COM
A: CREF80   COM : SYSCPY   CMD : SYSCPY   COM : AFF      HEX
A: ERROR    COM : SUBMIT   COM : XSUB     COM : STR      HEX
A: LIB80    COM : LH3FF    HEX : SETUP    STR : LH1FF    HEX
A: LHFF     HEX : DIRMPM   CMD
```

４．５．１３　システム・モード終了（ＥＸＴ）

ＥＸＴ

'ＥＸＴ' コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

システム・モード終了コマンドは，システム・モードを終了し，ＣＰ／Ｍ，または，ＣＣＰ／Ｍに戻ります。このとき，オープンされているファイルは，すべてクローズされます。

例

```
1>EXT ↵
A>                    ←CP/Mあるいは CCP/Mのﾌﾟﾛﾝﾌﾟﾄ が表示されます。
```

４．５．１４　コマンド・ヒストリ表示（ＨＩＳ）

ＨＩＳ

‘ＨＩＳ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

コマンド・ヒストリ表示コマンドは，ヒストリ・メモリに登録されている最新のコマンド（最大２０行分）を表示します。ヒストリ・メモリへの登録は，コマンドを実行するたびごとに自動的に登録されます。

ヒストリ・メモリに登録されているコマンドは，コマンド入力のときに簡単に読出すことができます。コマンドを読出す場合は，！ｎ〉と入力します。このときにｎに読出すコマンドの登録番号を指定します。また，直前のコマンドの場合は，！！〉と入力するだけで読出すことができます。読出したコマンドの実行は，コマンド行で〉を入力します。このとき，カーソルの位置はコマンド行のどこにあってもかまいません。

例

```
1>HIS 〉              ←ヒストリ・メモリに登録されているコマンドを表示します。
  1 LOD TEST
  2 MEM D OFX
  3 MAP
      .
      .
      .
 18 DIR
 19 HLP HIS
 20 HIS
1>

1>!18 〉              ←ヒストリ・メモリの18番目のコマンドを読出します。
 DIR 〉               ←このコマンドを実行するには〉を入力します。
 A: CONVPM   CMD : ABORT    CMD : ATTACH   CMD : DDT      CMD
 A: MPMSTAT  CMD : PIP      CMD : REN      CMD : SDIR     CMD
 A: OBJERR   HEX : STAT     CMD : SUBMIT   CMD : TOD      CMD
 A: TYPE     CMD : UNLOCKB  CMD : CXO      TXT : POW2     COM
                              .
                              .
                              .
1>
```

４．５．１５　ヘルプ（ＨＬＰ）

```
ＨＬＰ［_ｃｏｍｍａｎｄ］
```

command　：ｺﾏﾝﾄﾞ本体を指定します。

‘ＨＬＰ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

ヘルプ・コマンドは，ＩＥ－７８３１０Ａ－Ｒで使用できるコマンドの一覧，および，コマンドの使用方法を表示します。コマンド本体が指定されている場合は，指定されたコマンドの使用方法を表示します。コマンド本体が省略された場合は，コマンドの一覧が表示されます。その後，コマンド本体の入力ができます。

例

```
1>HLP DIR ↵   ←ｵﾍﾟﾗﾝﾄﾞ にｺﾏﾝﾄﾞを指定した場合
┌
｜"DIR"ｺﾏﾝﾄﾞ の説明
└
1>

1>HLP ↵
    Command Table
  ASM          BRA          BRD          BRE          BRT  ┐
  BRM          BRO          BR1          BR2          BR3  │
  CLK          COM          DAS          DIG          DIR  │
  EXT          HIS          HLP          LOD          LST  │
  MAP          MAT          MDR          MEM          MOD  │ ｺﾏﾝﾄﾞの一覧
  MOV          PGM          REG          RUN          RES  │
  SAV          SPR          STR          SUF          SYM  │
  TRM          TRX          TRQ          TRS          TRP  │
  TRD          VRY          SYS                            ┘
HLP>DIR ↵   ←"DIR"ｺﾏﾝﾄﾞ の説明を表示します。
┌
｜"DIR"ｺﾏﾝﾄﾞ の説明
└
HLP>↵      ←ﾍﾙﾌﾟ･ｺﾏﾝﾄﾞを終了します。
1>
```

４．５．１６　オブジェクト・ロード（ＬＯＤ）

```
LOD [_ {TTY1 }]
       {TTY2 }
```

スタンドアロン・モードの形式

```
LOD_file [_module name＼・・・・] [_ {C} ]
                                    {S}
```

システム・モードの形式

TTY1　　：ｼﾘｱﾙ･ﾁｬﾈﾙ 1 からｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞをﾛｰﾄﾞ します。

TTY2　　：ｼﾘｱﾙ･ﾁｬﾈﾙ 2 からｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞをﾛｰﾄﾞ します。

file　　：ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ 名，または，ｼﾝﾎﾞﾙ･ﾌｧｲﾙ 名を設定します。

module name＼：ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾓｼﾞｭｰﾙ 名を設定します。

C　　：ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞのみをﾛｰﾄﾞ するｽｲｯﾁです。

S　　：ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のみをﾛｰﾄﾞ するｽｲｯﾁです。

オブジェクト・ロード・コマンドは，システム・モードとスタンドアロン・モードでは異なります。スタンドアロン・モードでは，シリアル・チャネル１，あるいは，シリアル・チャネル２からヘキサ形式のオブジェクト・コードをロードすることができます。

システム・モードでは，ホスト・マシンのファイルに格納されているオブジェクト・コード，あるいは，シンボル・テーブル・ファイルをロードすることができます。

オブジェクトをロードするメモリは，ユーザ・マッピング，内部マッピング，ライト・プロテクト付き内部マッピングのどれかにマッピングされていなければなりません。

マッピングされていないメモリには，オブジェクトをロードできません。

（1）スタンドアロン・モードの場合

```
LOD [__ {TTY1} ]
       {TTY2}
```

スタンドアロン・モードの場合は，オブジェクト・コードだけをロードすることができます。オブジェクト・コードをロード中にエラーを検出した場合は，最終レコードまで読捨てます。スタンドアロン・モードでは，シンボル・テーブル・ファイルをロードすることはできません。

例

```
*LOD TTY1 )          ←ｼﾘｱﾙ･ﾁｬﾈﾙ1からｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞをﾛｰﾄﾞ する。
 complete            ←正常終了ﾒｯｾｰｼﾞ
*
```

オブジェクト・ロード中にエラーを検出した場合は，以下に示すようなメッセージが出力されます。この場合は，エラーを検出したレコードから最終レコードまで読捨てられます。

```
Check sum error      ←ﾁｪｯｸｻﾑ･ｴﾗｰを検出した
Bad character        ←ﾚｺｰﾄﾞ中に許されない文字を検出した
Non map area access ←ﾏｯﾋﾟﾝｸﾞ されていないﾒﾓﾘ にﾛｰﾄﾞ しようとした
```

（2）システム・モードの場合

```
LOD_ file[_module name＼, module name＼, ……][_ {C} ]
                                                {S}
```

システム・モードでは，オブジェクト・コード，および，シンボル・テーブル・ファイルをロードすることができます。

オブジェクト・コード，および，シンボル・テーブル・ファイルは，同時にロードすることもできます。また，オブジェクト・コードだけや，シンボル・テーブル・ファイルだけをロードすることもできます。

また，シンボル・テーブル・ファイルをロードする場合は，シンボル・テーブル・ファイルの必要なモジュールだけを指定してロードすることもできます。

例　ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞ，および，ｼﾝﾎﾞﾙ･ﾌｧｲﾙ を同時にﾛｰﾄﾞ する場合

```
1>LOD SAMPLE ↵          ←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞとｼﾝﾎﾞﾙ･ﾌｧｲﾙ を同時にﾛｰﾄﾞ
  object load complete←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞのﾛｰﾄﾞ が正常終了した場合のﾒｯｾｰｼﾞ
  symbol table loading←ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾛｰﾄﾞ 開始ﾒｯｾｰｼﾞ
  PUBLIC     load complete ┐
  MOD01      load complete │
  MOD02      load complete │ ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸのﾛｰﾄﾞ 終了ﾒｯｾｰｼﾞ
  MOD03      load complete │
  MOD04      load complete │
  MOD05      load complete ┘
1>

1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ ↵ ←ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸを指定
  object load complete←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞのﾛｰﾄﾞ が正常終了した場合のﾒｯｾｰｼﾞ
  symbol table loading←ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾛｰﾄﾞ 開始ﾒｯｾｰｼﾞ
  PUBLIC     load complete ┐
  MOD01      pass          │
  MOD02      load complete │ ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸのﾛｰﾄﾞ 終了ﾒｯｾｰｼﾞ
  MOD03      pass          │  pass は，指定されていないﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸに表示
  MOD04      load complete │ されます。
  MOD05      pass          ┘
1>
```

ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞとｼﾝﾎﾞﾙ･ﾌｧｲﾙ を同時にﾛｰﾄﾞ する場合は，ﾌｧｲﾙ名に拡張子を設定しますとｴﾗｰ になります。この場合の拡張子は，ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞは，“HEX” が，ｼﾝﾎﾞﾙ･ﾌｧｲﾙ は，“SYM” が自動的に設定されます。

```
1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\ ↵
  object file not found ←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞのﾌｧｲﾙ(SAMPLE.HEX)が見つからない場合
1>                          のﾒｯｾｰｼﾞ
```

ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞがﾛｰﾄﾞ できなかった場合は，ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾛｰﾄﾞ もしません。ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞ をﾛｰﾄﾞ 中にｴﾗｰ が検出された場合は，以下に示すようなﾒｯｾｰｼﾞ を表示します。

```
    Check sum error        ←ﾁｪｯｸｻﾑ･ｴﾗｰを検出した
    Bad character          ←ﾚｺｰﾄﾞ中に許されない文字を検出した
    Non map area access    ←ﾏｯﾋﾟﾝｸﾞ されていないﾒﾓﾘ にﾛｰﾄﾞ しようとした
```

```
1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\ ↵
  object load complete
  symbol table file not found ←ｼﾝﾎﾞﾙ･ﾌｧｲﾙ(SAMPLE.SYM) が見つからなかった
1>                              場合のﾒｯｾｰｼﾞ
```

```
1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\ ↵
  object load complete
  symbol table loading
  PUBLIC      load complete
  MOD01       pass
  MOD02       load failed←ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸをﾛｰﾄﾞ 中にｴﾗｰ を検出した場合のﾒｯｾｰｼﾞ
1>
```

ｼﾝﾎﾞﾙ･ﾌｧｲﾙ をﾛｰﾄﾞ 中にｴﾗｰ を検出した場合は，ｴﾗｰ を検出したﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸから以降のﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸはﾛｰﾄﾞ されません。ｴﾗｰ を検出したﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸも無効になります。

ただし，ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞは影響されません。

・ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞだけをﾛｰﾄﾞ する場合

```
1>LOD SAMPLE.HEX C ↵
  object load complete
1>

1>LOD SAMPLE C ↵
  object load complete
1>
```

ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞだけをﾛｰﾄﾞ する場合は，ｺﾏﾝﾄﾞの最後にｽｲｯﾁ “C”を付けます。ｽｲｯﾁ “C”を付けた場合は，指定されたｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞだけをﾛｰﾄﾞ します。ﾌｧｲﾙ名の拡張子を省略した場合は，“HEX” を拡張子とします。

・ｼﾝﾎﾞﾙ･ﾌｧｲﾙ だけをﾛｰﾄﾞ する場合

```
1>LOD SAMPLE.SYM S↵
  symbol table loading
  PUBLIC      load complete
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
  MOD05       load complete
1>

1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ S↵
  symbol table loading
  PUBLIC      load complete
  MOD01       pass
  MOD02       load complete
  MOD03       pass
  MOD04       load complete
  MOD05       pass
1>
```

ｼﾝﾎﾞﾙ･ﾌｧｲﾙ だけをﾛｰﾄﾞ する場合は，ｺﾏﾝﾄﾞの最後にｽｲｯﾁ “S”を付けます。ｽｲｯﾁ “S”を付けた場合は，指定されたｼﾝﾎﾞﾙ･ﾌｧｲﾙ だけをﾛｰﾄﾞ します。ﾌｧｲﾙ名の拡張子を省略した場合は，“SYM” を拡張子とします。ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸを指定した場合は，指定されたﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸだけをﾛｰﾄﾞ します。

```
1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ S↵
  symbol table loading
  PUBLIC      loaded module
  MOD01       pass
  MOD02       loaded module
  MOD03       pass
  MOD04       loaded module
  MOD05       pass
1>
```

すでに ﾛｰﾄﾞ済みのｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ をもう一度 ﾛｰﾄﾞした場合は，上記の例のようなﾒｯｾｰｼﾞ が表示されます。この場合， ﾛｰﾄﾞ済みのｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ は影響をうけません。

４．５．１７　出力ディバイス指定（ＬＳＴ）

```
LST [_ {file  }]
       {LST:  }
       {CON:  }
```

file　　：ファイルを出力ディバイス としてオープンします。

LST:　　：リスト 装置を出力ディバイス としてオープンします。

CON:　　：オープンされている出力ディバイス をクローズします。

‘ＬＳＴ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

出力ディバイス指定コマンドは，コマンドの実行結果をコンソールとともに指定された出力ディバイスに出力するために，ファイル，あるいは，リスト装置をオープンするコマンドです。

実際にファイル，あるいは，リスト装置に出力するタイミングは，‘↑Ｒ’キーが ★
入力されたコマンドから出力されます。ファイル，あるいは，リスト装置への出力の停止は，コマンド入力中の次の‘↑Ｒ’キーが入力されるまでです。 ★

例

```
1>LST B:SAMPLE.TXT ↵
1>
```

ドライブ BにSAMPLE.TXTというファイルが正しくオープンできた場合です。

```
1>LST B:SAMPLE.TXT ↵
  File already exists. Delete ? (Y or N): Y ↵
1>
```

ドライブ Bには，すでにSAMPLE.TXTというファイルが存在している場合に上記のようなメッセージ を出力します。この場合，すでに存在しているSAMPLE.TXTというファイルの属性が DIR属性で，かつ，R/W 属性です。ここで，Y を入力しますとSAMPLE.TXTをデリートして新しくSAMPLE.TXTというファイルをオープンします。Y 以外を入力しますとファイルのオープンはしません。つまり，すでに存在しているファイルは，そのままです。

```
1>LST B:SAMPLE.TXT ↵
  File already exists.
1>
```

ドライブ Bに，すでにSAMPLE.TXTというファイルが存在している場合です。ただし，前記の例と異なり，SAMPLE.TXTというファイルの属性が SYS属性，あるいは，R/O 属性です。

この場合は，コマンドは無視されます。

```
1>LST LST: ↵
1>
```

リスト 装置をオープンします。ホスト・マシン が MD-086 シリーズで CCP/Mの場合で，リスト 装置が他の処理によって使用中であったなら，下記のようなメッセージ が表示されます。

この場合はリスト 装置は，オープンされません。

```
1>LST LST: ↵
  List device is used by other process.
1>
```

・ファイルを出力ディバイス としてオープンした場合の例

```
1>LST B:SAMPLE.TXT ↵   ←出力ディバイス としてドライブ BにSAMPLE.TXTというファイル名でオープン
1>MAP ↵
  0000-02FF R/W   0300-07FF R/O   0800-0FFF R/W   1000-16FF Non
  1700-1FFF R/W   2000-2FFF R/O   3000-7FFF User  8000-FDFF Non
★ 1>MEM F 0,02FF 1,2,3,4,5,6,7,8 ↵   ←このコマンド行で↑R を入力します。
  1>MEM D 080X ↵
    0800   01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08    ........
  1>BRA A=100 ↵
  1>BRM BRA ↵
★ 1>TRM ALL ↵   ←このコマンド行で↑R を入力します。
  1>
```

→ SAMPLE.TXTに出力されます。

★ この場合は，ファイルをオープン後の↑R が入力されたコマンド(MEM F 0,02FF 1 ,

★ 2,･･･)から次の↑R が入力される前のコマンド(BRM BRA) までにコンソール に表示されたすべてがドライブ BのSAMPLE.TXTというファイルに出力されます。

４．５．１８　マッピング（MAP）

```
MAP [_ {W|R|U|K} [_partition] ]
```

W　　：内部ﾏｯﾋﾟﾝｸﾞ に設定します。

R　　：ﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 付き内部ﾏｯﾋﾟﾝｸﾞ に設定します。

U　　：ﾕｰｻﾞ･ｴﾘｱ にﾏｯﾋﾟﾝｸﾞ します。

K　　：ﾏｯﾋﾟﾝｸﾞ を解除します。

partition　：ﾏｯﾋﾟﾝｸﾞ の範囲を設定します。

マッピング・コマンドは，２５６バイト単位にエミュレーション・メモリをマッピングすることができます。マッピング範囲は，０～０ＦＤＦＦＨまで，２５６バイト単位に自由にマッピング指定をすることができます。ただし，内部ＲＯＭが指定（スタート・アップ時）された場合は，内部ＲＯＭの範囲内はライト・プロテクト付き内部マッピングに指定されます。この場合は，内部ＲＯＭの範囲内は，他のマッピング指定をすることはできません。

また，現在のマッピング状態を表示することもできます。マッピング状態の表示は，全体のマッピング状態の表示，および，指定マッピングの範囲だけの表示をすることができます。

内部ＲＯＭの範囲とマッピング指定可能範囲は以下のようになります。

内部ＲＯＭサイズ	指定可能範囲
なし	０～ＦＤＦＦＨ
４Ｋバイト	１０００Ｈ～ＦＤＦＦＨ
８Ｋバイト	２０００Ｈ～ＦＤＦＦＨ
１６Ｋバイト	４０００Ｈ～ＦＤＦＦＨ

例　マッピングの設定

```
*MAP R 1000,2FFF ↵ ←1000番地から2FFF番地までをﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 付き内部ﾏｯﾋﾟﾝｸﾞ
                     に設定します。
*MAP W 0XXX ↵       ←0 番地からFFF 番地までを内部ﾏｯﾋﾟﾝｸﾞ に設定します。
*MAP U 3000,7FFF ↵  ←3000番地から7FFF番地までをﾕｰｻﾞ･ﾏｯﾋﾟﾝｸﾞ に設定します。
*MAP K 8000,FFFF ↵  ←8000番地からFFFF番地までをﾏｯﾋﾟﾝｸﾞ 解除します。
 Mapping error      ←ﾏｯﾋﾟﾝｸﾞ 範囲以外を指定したためｴﾗｰ が表示されます。
*MAP K 8000,FDFF ↵  ←8000番地からFDFF番地までをﾏｯﾋﾟﾝｸﾞ 解除します。
*

*MAP R 345,765 ↵    ←300 番地から7FF 番地までがﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 付き内部ﾏｯﾋﾟﾝｸﾞ
                     に設定されます(ﾏｯﾋﾟﾝｸﾞ単位が256 ﾊﾞｲﾄ 単位のため) 。
*MAP K 1XXX ↵       ←1000番地から1FFF番地までをﾏｯﾋﾟﾝｸﾞ 解除します。
*MAP W 1780,1F8F ↵  ←1700番地から1FFF番地までが内部ﾏｯﾋﾟﾝｸﾞ に設定されます。
```

マッピング状態の表示

```
*MAP ↵              ←すべてのﾏｯﾋﾟﾝｸﾞ 状態が表示されます。
 0000-02FF R/W   0300-07FF R/O   0800-0FFF R/W   1000-16FF Non
 1700-1FFF R/W   2000-2FFF R/O   3000-7FFF User  8000-FDFF Non
*
        R/W     : 内部ﾏｯﾋﾟﾝｸﾞ
        R/O     : ﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 付き内部ﾏｯﾋﾟﾝｸﾞ
        User    : ﾕｰｻﾞ･ﾏｯﾋﾟﾝｸﾞ
        Non     : ﾏｯﾋﾟﾝｸﾞ 指定なし

*MAP W ↵            ←内部ﾏｯﾋﾟﾝｸﾞ されている範囲がすべて表示されます。
 0000-02FF  0800-0FFF  1700-1FFF
*

*MAP R ↵ ←ﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 付き内部ﾏｯﾋﾟﾝｸﾞ されている範囲がすべて表示されます。
 0300-07FF  2000-2FFF
*

*MAP U ↵            ←ﾕｰｻﾞ･ﾏｯﾋﾟﾝｸﾞ されている範囲がすべて表示されます。
 3000-7FFF
*
```

マッピングは，２５６バイト単位にするため，マッピング開始アドレスの下位８ビットは００Ｈと判断されます。また，マッピング終了アドレスの下位８ビットはＦＦＨと判断されます。

‘ＭＡＰ　Ｋ’コマンドでマッピング範囲を指定しない場合は，マッピング可能な範囲すべてのマッピングが解除されます。

４．５．１９　演算（ＭＡＴ）

ＭＡＴ＿ｅｘｐｒｅｓｓｉｏｎ

expression：word，byte，または，symbolを演算子で結合して記述できます。

演算コマンドは，オペランドに記述された式表現を評価し，その結果を１６進数，１０進数，８進数，および，２進数で表示します。オペランドには，以下に示す演算子を記述することができます。

(,)	高
＊，／	↑
＋，－	優先順位
ＡＮＤ	↓
ＯＲ，ＸＯＲ	低

演算は，すべて１６ビットの整数で行なわれます。演算結果が１６ビットをオーバした場合は，下位の１６ビットだけが有効となります。

例

```
*MAT 5H+7H AND 17Q )          ←式の入力
 0CH,12T,14Q,1100Y            ←演算結果の表示
*
```

演算結果は，１６進数，１０進数，８進数，２進数の順に表示されます。

４．５．２０　モード・レジスタ操作（ＭＤＲ）

```
MDR [_D[_mode register name]]    ←モード・レジスタの表示
MDR_C[_mode register name]       ←モード・レジスタの変更
```

mode register name：以下に示すモード・レジスタ名を記述できます。

PM0,PM1,PM2,PM3,PM5,PMC2,PMC3,RTPC,MM,RFM,WDM,STBC,TBM,INTM,ISPR,CCW,
SCM,SCC,FRCC,CPTM,PWMM,ADM,CUIM,UDCC0,CRC,UDCC1,TMC0,TMC1,CRIC00,
CRMS00,CRIC01,CRIC10,CRMS10,CRIC11,EXIC0,EXMS0,EXIC1,EXMS1,EXIC2,
EXMS2,TMIC0,TMMS0,TMIC1,TMMS1,TMIC2,TMMS2,SEIC,SRIC,SRMS,STIC,STMS,
ADIC,ADMS,TBIC

モード・レジスタ操作コマンドは，モード・レジスタの表示，あるいは，変更をすることができます。モード・レジスタの表示は，すべてのモード・レジスタの表示，あるいは，指定されたモード・レジスタだけの表示をすることができます。すべてのモード・レジスタを表示したい場合は，‘ＭＤＲ↵’，あるいは，‘ＭＤＲ＿Ｄ↵’と入力します。特定のモード・レジスタだけを表示したい場合は，オペランドに表示したいモード・レジスタ名を指定します。

モード・レジスタの変更は，オペランドで指定されたモード・レジスタから順番に変更されます。オペランドのモード・レジスタ名を省略した場合は，‘ＰＭ０’から最後の‘ＴＢＩＣ’まで順番に変更することができます。

モード・レジスタの変更を途中でやめたい場合は，‘．’（ピリオド）を入力します。

もし，モード・レジスタの内容を変更したくない場合は，‘↵’（リターン・キー）を入力します。

例1　モード・レジスタの表示

```
*MOD D ↵                ←すべてのモード・レジスタを表示

 PM0     PM1     PM2     PM3     PM5     PMC2    PMC3    RTPC    MM
 10      20      30      40      50      60      70      80      90

 RFM     WDM     STBC    TBM     INTM    ISPR    CCW     SCM     SCC
 A0      B0      C0      D0      E0      F0      11      22      33

 FRCC    CPTM    PWMM    ADM     CUIM    UDCC0   CRC     UDCC1   TMC0
 44      55      CC      12      9A      89      AB      E5      F6

 TMC1    CRIC00  CRMS00  CRIC01  CRIC10  CRMS10  CRIC11  EXIC0   EXMS0
 66      BB      DD      34      BC      67      CD      D4      7A

 EXIC1   EXMS1   EXIC2   EXMS2   TMIC0   TMMS0   TMIC1   TMMS1   TMIC2
 77      AA      EE      56      DE      45      EF      C3      8B

 TMMS2   SEIC    SRIC    SRMS    STIC    STMS    ADIC    ADMS    TBIC
 88      99      FF      78      F1      23      A1      B2      9C
*

*MDR D CPTM ↵          ←'CPTM'レジスタの表示
 CPTM      55
*
```

例2　モード・レジスタ内容の変更

```
*MDR C ↵                ←すべてのモード・レジスタの変更
 PM0       10 = 22 ↵    ←'10'を'22'に変更
 PM1       20 = ↵       ←変更なし
 PM2       30 = 44 ↵    ←'30'を'44'に変更
      .
      .
      .
 ADMS      B2 = ↵       ←変更なし
 TBIC      9C = FF ↵    ←'9C'を'FF'に変更
*

*MDR C ADM ↵            ←'ADM'レジスタ から，すべて変更
 ADM       12 = 50 ↵    ←'12'を'50'に変更
 CUIM      9A = ↵       ←変更なし
 UDCC0     89 = 98 ↵    ←'89'を'98'に変更
 CRC       AB = ↵       ←変更なし
      .
      .
      .
 TMMS2     88 = . ↵     ←モード・レジスタの変更をやめる
*
```

４．５．２１　メモリ操作（ＭＥＭ）

```
MEM_C[_word]
MEM [_D[_ {word      } ]
          {partition }
MEM_E[_partition]
MEM_ {F} _partition_data string
     {G}
MEM_ {M}
     {X} _partition_word
     {V}
```

C	：ﾒﾓﾘ 内容の変更
D	：ﾒﾓﾘ 内容の表示
E	：ﾕｰｻﾞ･ﾒﾓﾘ のﾃｽﾄ
F	：ﾒﾓﾘ のｲﾆｼｬﾗｲｽﾞ
G	：ﾒﾓﾘ 内容のｻｰﾁ
M	：ﾒﾓﾘ 内容のｺﾋﾟｰ
X	：ﾒﾓﾘ 内容の交換
V	：ﾒﾓﾘ 内容の比較

メモリ操作コマンドは，メモリ内容の変更，表示，テスト，イニシャライズ，サーチ，コピー，交換，比較をすることができます。メモリ操作コマンドで操作するメモリは，ユーザ・マッピング，内部マッピング，ライト・プロテクト付き内部マッピングのどれかにマッピングされていなければなりません。

内部ＲＡＭ（ＦＥ００～ＦＥＦＦ）におけるメモリ操作は，ＦＥ００～ＦＥ７Ｆの範囲では全てのメモリ操作が可能ですが，レジスタ・バンク（ＦＥ８０～ＦＥＦＦ）ではメモリの変更および表示のみが可能となります。

マッピングされていないメモリは，メモリ操作コマンドでは操作できません。

（1）メモリ内容の変更

MEM_ C[_word]

word　　：ﾒﾓﾘ の変更開始ｱﾄﾞﾚｽ

マッピング範囲内のメモリ内容を8ビット単位で変更することができます。もし，マッピングされていないメモリ内容を，変更しようとした場合は，エラーとなります。

また，メモリの変更開始アドレスを省略した場合は，前回に変更を終了したアドレスがメモリの変更開始アドレスになります。

例

```
*MEM C 100 ↵          ←100 番地からのﾒﾓﾘ 内容を変更

 0100  00 11 ↵
 0101  11 22 ↵
 0102  22 33 ↵
 0103  33 44 ↵
 0104  44 55 ↵
 0105  55 66 ↵
 0106  66 77 ↵
 0107  77 88 ↵
 0108  88 .↵          ←ﾒﾓﾘ 内容の変更終了
*

*MEM C ↵              ←ﾒﾓﾘ 内容の変更開始ｱﾄﾞﾚｽを省略
 0108  88 99 ↵        ←前回のﾒﾓﾘ 内容の変更を終了したｱﾄﾞﾚｽ
 0109  99 0AA↵
 010A  AA 0BB↵
 010B  BB 0CC↵
 010C  CC .↵          ←ﾒﾓﾘ 内容の変更終了

*
```

（２）メモリ内容表示

```
MEM [_D[_ {word      } ]]
          {partition }
```

word　　：ﾒﾓﾘ の表示開始ｱﾄﾞﾚｽ
partition ：ﾒﾓﾘ の表示開始/ 終了ｱﾄﾞﾚｽ

マッピングされているメモリの内容を表示することができます。表示開始／終了アドレスが指定された場合は，指定された範囲が表示されます。開始アドレスだけが指定された場合は，指定されたアドレスから１１行分のメモリ内容が表示されます。また，メモリの表示アドレスが省略された場合は，前回にメモリ内容表示された次のアドレスが指定されます。この場合も１１行分のメモリ内容が表示されます。

例

```
*MEM D 100,17F ↵     ←100 ～ 17Fまでのﾒﾓﾘ 内容を表示する
 0100   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    ................
 0110   30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F    0123456789:;<=>?
 0120   40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F    @ABCDEFGHIJKLMNO
 0130   50 51 52 53 54 55 56 57 58 59 5A 61 62 63 64 65    PQRSTUVWXYZabcde
                                  .
                                  .
                                  .
 0170   20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F     !"#$%&'()*+,-./
└ｱﾄﾞﾚｽ┘ └──────────── ﾃﾞｰﾀ(16ﾊﾞｲﾄ) ────────────┘  └ ASCII 文字 ┘
*

*MEM D 108 ↵        ←108 から１１行を表示する
 0108                           08 09 0A 0B 0C 0D 0E 0F        ........
 0110   30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F    0123456789:;<=>?
                                  .
                                  .
                                  .
 01A0   A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF    ................
*

*MEM D ↵             ←ﾒﾓﾘ 内容表示ｱﾄﾞﾚｽを省略
 01B0   B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF    ................
 01C0   C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF    ................
 01D0   D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF    ................
                                  .
                                  .
                                  .
 0250   30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F    0123456789:;<=>?
```

メモリ内容表示は，メモリの内容を１６進数とＡＳＣＩＩ文字で表示します。ＡＳＣＩＩ文字は，１６進数の‘００’～‘１Ｆ’，‘７Ｅ’～‘ＦＦ’までは，ＡＳＣＩＩ文字“．”（ピリオド）が表示されます。また，‘２０’～‘７Ｄ’までは，下記によります。

下位4ビット	上位4ビット 2	3	4	5	6	7
0	(SP)	0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	¥	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	
F	/	?	O	_	o	

（3）ユーザ・メモリのテスト

MEM_E[_partition]

partition　：ﾒﾓﾘ のﾃｽﾄ 範囲

ユーザ・マッピングされたメモリのテストをすることができます。メモリのテスト範囲が指定された場合は，指定された範囲のメモリをテストします。また，メモリのテスト範囲が省略された場合は，ユーザ・マッピングされたすべてのメモリをテストします。

なお，メモリ・テスト終了後は，メモリの内容は破壊されますので注意してください。

例

```
*MEM E 0XXX ↵      ←0000～0FFFまでをﾃｽﾄ する
 complete          ←ﾒﾓﾘ は正常
*

*MEM E ↵           ←ﾕｰｻﾞ･ﾏｯﾋﾟﾝｸﾞ されているすべてのﾒﾓﾘ をﾃｽﾄ する
 1021              ←ﾒﾓﾘ･ﾃｽﾄ で異常を検出したｱﾄﾞﾚｽを表示
*
```

メモリ・テストで異常を検出した場合は，異常を検出したアドレス以後のテストはされません。

（４）メモリのイニシャライズ

```
MEM_ F_partition_data string
```

partition　　：ﾒﾓﾘ のｲﾆｼｬﾗｲｽﾞ 範囲

data string ：ｲﾆｼｬﾗｲｽﾞ･ﾃﾞｰﾀ

メモリのイニシャライズは，イニシャライズ範囲で指定された範囲にイニシャライズ・データを設定します。イニシャライズ・データに指定できるデータ列は，最大１０個までです。

イニシャライズ・データには，特殊数値表現も記述することができます。この場合の特殊数値表現は，連続したデータではなくマスク・データとなります。

例

```
*MEM F 100,1FF 1,2,3,4,5,6,7,8,9,0 ↵   ← 100～1FF までを1,2,3,4,5,6,7,8,9,0,
*                                          のﾃﾞｰﾀ 列でｲﾆｼｬﾗｲｽﾞ する

*MEM D 100,1FF ↵      ← ｲﾆｼｬﾗｲｽﾞ後のﾒﾓﾘ 内容の表示
 0100   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06     ................
 0110   07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02     ................
 0120   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08     ................
                                    .
                                    .
                                    .
 01E0   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00     ................
 01F0   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06     ................
```

ﾒﾓﾘ は，上記のようにｲﾆｼｬﾗｲｽﾞ されます。

```
*MEM F 100,11F 3X↵   ←ﾏｽｸ･ﾃﾞｰﾀ によるｲﾆｼｬﾗｲｽﾞ
*

*MEM D 100,13F ↵      ←ｲﾆｼｬﾗｲｽﾞ 後のﾒﾓﾘ 内容の表示
 0100   31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36     1234567890123456
 0110   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32     7890123456789012
 0120   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08     ................
 0130   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04     ................
```

ﾏｽｸ･ﾃﾞｰﾀ ‘3X’ でｲﾆｼｬﾗｲｽﾞ した場合は，上記のように下位 4ﾋﾞｯﾄ は変化しません。

（５）メモリ内容のサーチ

MEM_G_partition_data string

partition　　：ﾒﾓﾘ のｻｰﾁ 範囲
data string　：ｻｰﾁ･ﾃﾞｰﾀ

メモリ内容のサーチは，メモリのサーチ範囲で指定された範囲内からサーチ・データで指定されるデータ列をサーチすることができます。サーチ・データで指定できるデータ列は最大１０個までです。

サーチ・データには，特殊数値表現も記述できます。この場合の特殊数値表現は，連続したデータではなくマスク・データとなります。

例

100～1FF のﾒﾓﾘ 内容が，以下のようになっているとします。

```
0100   31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36    1234567890123456
0110   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32    7890123456789012
0120   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08    ................
0130   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04    ................
0140   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00    ................
0150   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06    ................
0160   07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02    ................
0170   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08    ................
0180   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04    ................
0190   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00    ................
01A0   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06    ................
01B0   07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02    ................
01C0   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08    ................
01D0   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04    ................
01E0   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00    ................
01F0   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06    ................
```

このとき，次のｺﾏﾝﾄﾞを実行すると，ｻｰﾁ 結果が出力されます。

```
*MEM G 100,1FF 30,31,32↵        ← 100～1FF の範囲から30,31,32の連続し
 0109                               たﾃﾞｰﾀ 列をｻｰﾁ する
 0113  ├ ﾃﾞｰﾀ列を検出したｱﾄﾞﾚｽ
 011D  ┘
*
```

（６）メモリ内容のコピー

```
MEM_M_partition_word
```

partition　：ｺﾋﾟｰ 開始範囲
word　　：ｺﾋﾟｰ 先ｱﾄﾞﾚｽ

メモリ内容のコピーは，コピー開始範囲で指定された範囲のメモリ内容をコピー先アドレスで指定されるアドレス以降にコピーします。コピー開始範囲で指定された範囲とコピー先アドレスは，重複してもかまいません。

例

```
*MEM M 100,13F 180 ↵          ←100 ～ 13Fまでを180 以降にｺﾋﾟｰ する
*
*MEM D 100,1FF ↵              ←ｺﾋﾟｰ 後のﾒﾓﾘ 内容の表示
 0100  31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36  1234567890123456
 0110  37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32  7890123456789012
 0120  03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08  ................
 0130  09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04  ................
 0140  05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00  ................
 0150  01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06  ................
 0160  07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02  ................
 0170  03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08  ................
 0180  31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36  1234567890123456
 0190  37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32  7890123456789012
 01A0  03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08  ................
 01B0  09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04  ................
 01C0  03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08  ................
 01D0  09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04  ................
 01E0  05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00  ................
 01F0  01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06  ................
*
```

（７）メモリ内容の交換

```
MEM_X_partition_word
```

partition　：ﾒﾓﾘ の交換範囲
word　：ﾒﾓﾘ の交換先ｱﾄﾞﾚｽ

メモリ内容の交換は，メモリの交換範囲で指定されたメモリ内容と，メモリの交換先アドレスで指定されたアドレス以降のメモリ内容を交換します。メモリの交換範囲で指定された範囲と，メモリの交換先アドレスで指定されたアドレスは，重複してはいけません。

例

100 ～ 17Fのﾒﾓﾘ 内容が以下のようになっているとします。

```
  0100   31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36 1234567890123456
  0110   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012
  0120   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0130   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................
  0140   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  0150   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................
  0160   07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  0170   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
```

このとき，次のｺﾏﾝﾄﾞを実行します。

```
*MEM X 100,13F 140 )          ←100 ～ 13Fまでを140 以降と交換する
 *
```

```
*MEM D 100,17F )              ←交換後のﾒﾓﾘ 内容の表示
  0100   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  0110   01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................
  0120   07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  0130   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0140   31 32 33 34 33 36 37 38 39 30 31 32 33 34 35 36 1234567890123456
  0150   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012
  0160   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0170   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................
*
```

（8）メモリ内容の比較

```
MEM_V_partition_word
```

partition　：ﾒﾓﾘ の比較範囲
word　　：ﾒﾓﾘ の比較先ｱﾄﾞﾚｽ

メモリ内容の比較は，メモリの比較範囲で指定された範囲のメモリ内容と，メモリの比較先アドレスで指定されたアドレス以降のメモリ内容を比較します。比較した結果メモリ内容が一致しない場合は，一致しないアドレスとデータがそれぞれ表示されます。メモリ内容の比較は，メモリの比較範囲が終了するまで続けます。

例

```
*MEM V 100,1FF 200 )          ←100 ～ 1FFまでを200 以降と比較する
 0140 31  0240 30
 0158 35  0258 41
 018D 34  028D 00
 01FF 06  02FF 01

*
```

４．５．２２　チャネル２モード設定（ＭＯＤ）

```
MOD[_MODE={CHAR|FLOW}][_BAUD={19200|9600|4800|2400|1200|600|300}][_LONG={7|8}]
                                [_PAR={NON|EVEN|ODD}][_STOP={1|2}]
```

MODE　　：ﾊﾝﾄﾞｼｪｲｸ･ﾓｰﾄﾞ の選択
BAUD　　：ﾎﾞｰﾚｰﾄ の選択
LONG　　：ｷｬﾗｸﾀ 長の選択
PAR　　：ﾊﾟﾘﾃｨ･ﾋﾞｯﾄの選択
STOP　　：ｽﾄｯﾌﾟ･ﾋﾞｯﾄ長の選択

チャネル２モード設定コマンドは，シリアル・チャネル２の動作状態の設定をすることができます。コマンドのオペランドが省略された場合は，動作状態の設定を対話形式で設定することができます。

なお，初期状態では，１キャラクタ・ハンドシェイク，９６００ボー，８ビット長，パリティ・ビットなし，ストップ・ビット２に設定されています。

例

```
*MOD MODE=CHAR BAUD=4800 LONG=8 PAR=NON STOP=2↵
*
```

1ｷｬﾗｸﾀ･ﾊﾝﾄﾞｼｪｲｸ，ﾎﾞｰﾚｰﾄ は4800ﾎﾞｰ，ｷｬﾗｸﾀ 長は8ﾋﾞｯﾄ，ﾊﾟﾘﾃｨ･ﾋﾞｯﾄなし，ｽﾄｯﾌﾟ･ﾋﾞｯﾄは2ﾋﾞｯﾄに設定する。

```
*MOD↵                          ←対話形式でﾁｬﾈﾙ2 の動作状態を設定する。
 Mode  CHAR = FLOW↵            ←ﾊﾞｯﾌｧ制御ﾓｰﾄﾞ に変更
 Baud  4800 = 9600↵            ←ﾎﾞｰﾚｰﾄ を9600ﾎﾞｰに変更
 Long     8 = ↵                 ←ｷｬﾗｸﾀ 長は変更しない
 Par    NON = EVEN↵             ←偶数ﾊﾟﾘﾃｨ･ﾋﾞｯﾄに変更
 Stop     2 = 1↵                ←ｽﾄｯﾌﾟ･ﾋﾞｯﾄ長を1 に変更
*
```

４．５．２３　内部→ユーザ／ユーザ→内部メモリ転送（ＭＯＶ）

```
MOV_{U}_partition_word
    {I}
```

U　　：内部→ユーザ　メモリ転送
I　　：ユーザ →内部 メモリ転送
partition：転送元メモリ 範囲
word　　：転送先メモリ・アドレス

メモリ転送コマンドは，内部メモリからユーザ・システムのメモリへ，あるいは，ユーザ・システムのメモリから内部メモリへメモリ内容の転送ができます。

内部メモリからユーザ・システムのメモリへの転送の場合は，転送元メモリ範囲で指定された範囲のメモリ内容を転送先メモリ・アドレスで指定されるユーザ・システムのメモリに転送します。転送先メモリ・アドレスで指定されたアドレスは，マッピング・コマンドでユーザ・エリアにマッピングされていなければなりません。

ユーザ・システムのメモリから内部メモリへの転送の場合は，転送元メモリ範囲で指定された範囲のユーザ・システムのメモリ内容を転送先メモリ・アドレスで指定される内部メモリに転送します。転送先メモリ・アドレスで指定されるアドレスは，マッピング・コマンドで内部マッピング，あるいは，ライト・プロテクト付き内部マッピングに指定されていなければなりません。

なお，内部ＲＯＭに指定されている範囲は，メモリ転送することはできません。

例

```
*MAP U 0,0FFF↵          ← 0～FFF 番地をユーザ・マッピング にする
*MOV U 1000,1FFF 0 ↵    ←内部メモリ の1000～1FFF番地までを，ユーザ・システム
*                         の 0番地以降に転送
*MAP R 0XXX↵            ← 0～FFF 番地をライト・プロテクト つき内部マッピング に
                          する
*MOV I 0XXX 0↵          ←ユーザ・システムの 0～0FFF番地までを，内部メモリ の
*                         0 番地以降に転送
```

４．５．２４　ＰＲＯＭプログラマ制御（ＰＧＭ）

ＰＧＭ

ＰＲＯＭプログラマ制御コマンドは，ＩＥ－７８３１０Ａ－Ｒのシリアル・チャネル２とＰＧ－１５００または，ＰＧ－２０００との間でインタフェースをとるためのコマンドです。

このコマンドを実行するとＰＧＭモードとなります。

ＰＧＭモードでは，ＰＧ－１５００または，ＰＧ－２０００との間でオブジェクト・コードのアップ／ダウン・ロードができます。

ＰＧＭモードの終了は，コンソールからｃｔｒｌ－Ｚを入力することによって行います。

（1）ＰＧ－１５００，ＰＧ－２０００の概要

ＰＧ－１５００およびＰＧ－２０００を使用して読み出し，書き込みができるＲＯＭの種類を示します。

（a）ＰＧ－１５００では，付属の２７ＸＸシリーズ用ソケット・ボードおよび４ビットＣＰＵ用インタフェース・ボードと別売のアダプタを組み合わせることで多種類のＲＯＭが使用できます。

・付属の２７ＸＸシリーズ用ソケット・ボードに直接セットすることにより次のＲＯＭが使えます。

μPD27256,μPD27C256, μPD27256A, μPD27C256A,μPD27C512,
μPD27C1000,μPD27C1001,μPD27C1024

（b）ＰＧ－２０００では次のＰＲＯＭが使用できます。

μPD2716, μPD2732, μPD2732A,μPD2764, μPD27128,
μPD27C64,μPD27256,μPD27C256

（２）ＰＧ－１５００，ＰＧ－２０００との接続

ＩＥ－７８３１０Ａ－ＲにＰＲＯＭプログラマ（ＰＧ－１５００，ＰＧ－２０００）を接続する場合ハードウエア設定，および，接続方法について説明します。

① ＩＥ－７８３１０Ａ－Ｒ，ＰＧ－１５００，ＰＧ－２０００，およびターミナルのボー・レートを必ず合わせてください。

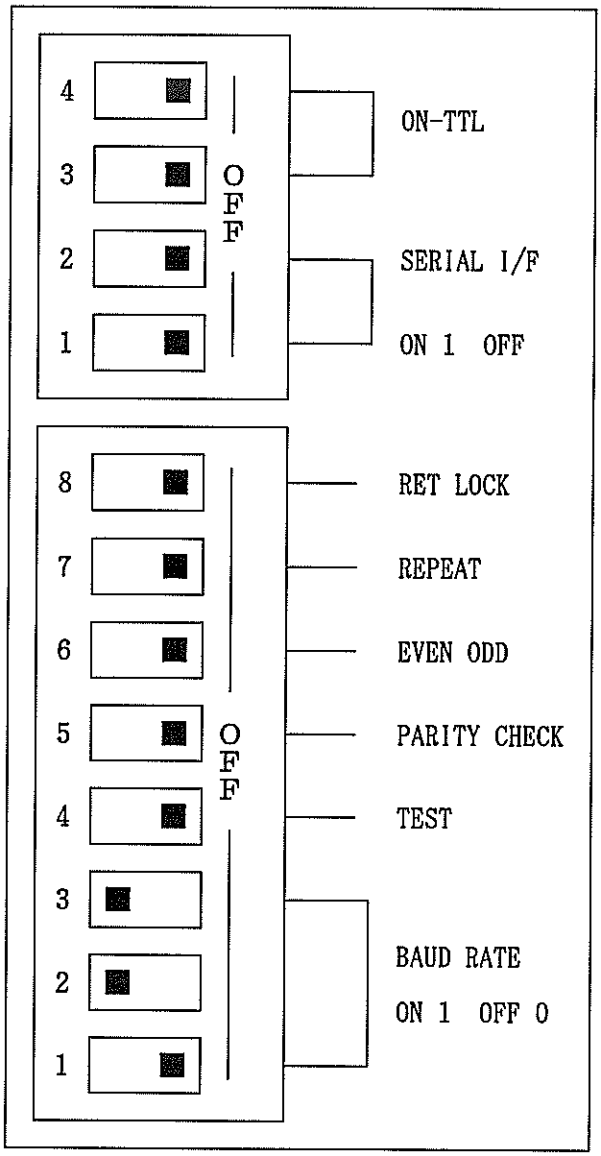
② ＰＧ－１５００を使用する場合は，キー・スイッチを使用して，ボー・レートを設定してください。

ボー・レートはＰＧ－１５００の取扱説明書を参照してください。

③ ＰＧ－２０００を使用する場合は，図４－２のように設定してください。ボー・レートはＩＥシリアル・チャネル２に合わせてください。

なお，詳細な接続方法については，「ＩＥ－７８３１０Ａ－Ｒ　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。

図４－２




ONを示します。


OFF を示します。

表４－２　PG-2000 ﾎﾞｰ･ﾚｰﾄの設定

1	2	3	BAUD RATE
0	0	0	110
1	0	0	300
0	1	0	600
1	1	0	1200
0	0	1	2400
1	0	1	4800
0	1	1	9600

（注意1）　表 4-2の ‘1　2　3’ は，8 連ﾃｨｯﾌﾟ･ｽｲｯﾁの ‘1　2　3’ です。

（注意2）　表 4-2において 1は ON ，　0は OFFを示します。

④　ＩＥ－７８３１０Ａ－Ｒの設定

まず，シリアル・チャネル２をモデム・モードにします。本体前面のスライド・スイッチでモデム・モードにしてください。

次に，ＭＯＤコマンドでシリアル・チャネル２を次のように設定します。

```
*MOD↵
 Mode  CHAR = CHAR↵
 Baud  9600 = 9600↵
 Long     8 = 8↵
 Par    Non = Non↵
 Stop     2 = 2↵
*
```

初期状態では，そのまま PG-1500，PG-2000 に接続できるようになっています。PG-1500 ，PG-2000 とＩＥを接続するときは，必ずＰＧに付属している接続ケーブルを使用してください。

⑤　ＰＧのシリアル・モード選択

（ａ）ＰＧ－１５００とＰＧ－２０００とは，取り扱いが異なるので注意してください。

（ｂ）ＩＥとＰＧ－１５００，ＰＧ－２０００のシリアル回線の通信開始方式

・ＰＧ－１５００の場合

FUNC TION キー　→　REMOTE 7 キー

・ＰＧ－２０００の場合

RESET キー　→　□REM B キー　→　START キー

以下にＰＧ－１５００，ＰＧ－２０００のキー配置を示します。

図４－３　ＰＧ－１５００のキー配置

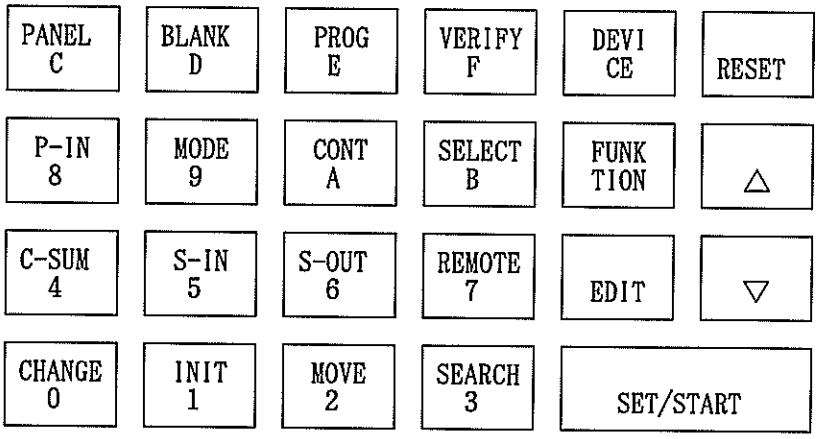
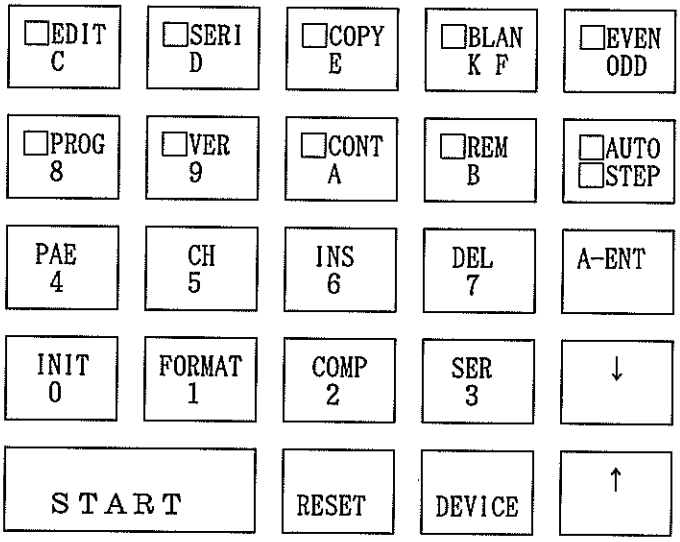


図４－４　ＰＧ－２０００のキー配置



（３）ＰＲＯＭプログラマ制御コマンドの起動と終了

①　ＰＲＯＭプログラマ制御コマンドを入力しますとメッセージが表示され，ＰＧからのプロンプト‘＊’が出力されます。

・ＰＧ－１５００の場合

```
*PGM ↵
 Beginning of PGM mode    ← PGMｺﾏﾝﾄﾞの開始ﾒｯｾｰｼﾞ
*                         ← PG から出力されたﾌﾟﾛﾝﾌﾟﾄ
PG>
```

・ＰＧ－２０００の場合

```
*PGM ↵
 Beginning of PGM mode    ← PGMｺﾏﾝﾄﾞの開始ﾒｯｾｰｼﾞ
*                         ← PG から出力されたﾌﾟﾛﾝﾌﾟﾄ
```

プロンプトが出力されない場合は，（２）の⑤（ｂ）の処理を再度行ってください。それでもプロンプトが出力されない場合は，再度設定の確認①～④を行ってください。

```
*PGM ↵
 Beginning of PGM mode
 ■                        ← ｶｰｿﾙ が停止し，ﾌﾟﾛﾝﾌﾟﾄ が出力されない状態
```

②　ＰＧ－２０００を使用してＰＲＯＭプログラマ制御コマンドをW起動したときは，トランジェント・モード（コンソールへエコー・バックしないモード）になっています。

ここで，‘Ｉ↵’と入力してください。

インテリジェント・モード（コンソールへエコー・バックするモード）へ移行します。

③　ＰＧのコマンドを操作します。

④　ＰＲＯＭプログラマ制御コマンドを終了したい場合は，コンソールからｃｔｒｌ－Ｚを入力することにより，ＩＥのコマンドの使用が可能となります。

次に，表示例を示します。

・ＰＧ－１５００の場合

```
PG>                       ← ctrl-Zｷｰ を入力

 Exit PGM mode (Y/N) Y    ← PGM ﾓｰﾄﾞの終了確認ﾒｯｾｰｼﾞ
 Termination of PGM mode  ← PGM ﾓｰﾄﾞの終了ﾒｯｾｰｼﾞ
*
```

・ＰＧ－２０００の場合

```
*                                   ← ctrl-Zｷｰ を入力

  Exit PGM mode (Y/N) Y             ← PGM ﾓｰﾄﾞの終了確認ﾒｯｾｰｼﾞ
  Termination of PGM mode           ← PGM ﾓｰﾄﾞの終了ﾒｯｾｰｼﾞ
*
```

注意　ＰＧ－２０００の場合プロンプトは，ＩＥ（スタンド・アロン・モードの場合）の場合も，ＰＧの場合も‘＊’ですので注意してください。

コマンド入力ライン途中，および，コマンド実行時にｃｔｒｌ－Ｚキーを入力することによってＰＲＯＭプログラマ制御コマンドを終了し，ＩＥのコマンド待ちの状態に復帰することができます。

ただし，ＰＧとＩＥでＨＥＸデータのアップ／ダウン・ロード実行時は，ｃｔｒｌ－Ｚキーは無視されます。

（４）ＰＲＯＭプログラマ制御コマンドの詳細

ＰＲＯＭプログラマ制御コマンドは，ＩＥ－７８３１０Ａ－Ｒで使用している端末よりＰＧのコマンドをコントロールするためのものです。

また，ＩＥ－７８３１０Ａ－Ｒ内のマッピングされたメモリの内容を，ＰＧへ転送，または，ＰＧのメモリの内容を，ＩＥ－７８３１０Ａ－Ｒのマッピングされたメモリへ転送することができます。

次に，ＰＧのコマンド一覧を示します。

（ａ）コマンド一覧

◆ＰＧ－１５００のコマンド一覧

詳細はＰＧの取扱説明書を参考にしてください。

表４－３　ＰＧ－１５００コマンド一覧

ｺﾏﾝﾄﾞ	形　　式	機　　能
ＲＲ	PG>RR P_S_ADR,R_E_ADR , PG_S_ADR,CONV ↵	PROM内容のﾘｰﾄﾞ
ＲＳ	PG>PS sub ↵　　sub=C/R/A	ﾃﾞﾊﾞｲｽの選択
ＲＶ	PG>RV P_S_ADR,R_E_ADR,PG_S_ADR,CONV ↵	PROMとPG内ﾒﾓﾘ の比較
ＲＷ	PG>PW PG_S_ADR,PG_E_ADR,R_S_ADR,CONV ↵	PROMへの書き込み
ＲＺ	PG>RZ ↵	PROM消去ﾁｪｯｸ
ＭＣ	PG>MC PG_S_ADR ↵	PG内ﾒﾓﾘ 内容の変更
ＭＤ	PG>MD PG_S_ADR,PG_E_ADR ↵	PG内ﾒﾓﾘ 内容の表示
ＭＦ	PG>MF PG_S_ADR,PG_E_ADR,INT_DATA ↵	PG内ﾒﾓﾘ の初期化
ＬＩ	PG>LI ↵	IEからPGへのﾃﾞｰﾀ 転送
ＳＩ	PG>SI PG_S_ADR,PG_E_ADR ↵	PGからIEへのﾃﾞｰﾀ 転送
？？	PG>?? ↵	ｺﾏﾝﾄﾞ・ﾍﾙﾌﾟ

備考

PG_S_ADR	PGｽﾀｰﾄ・ｱﾄﾞﾚｽ
PG_E_ADR	PGｴﾝﾄﾞ・ｱﾄﾞﾚｽ
R_S_ADR	PROMｽﾀｰﾄ・ｱﾄﾞﾚｽ
R_E_ADR	PROMｴﾝﾄﾞ・ｱﾄﾞﾚｽ
CONV	ｱﾄﾞﾚｽ分割指定
INT_DATA	初期化ﾃﾞｰﾀ

◆ＰＧ－２０００のコマンド一覧

Ｌコマンド，Ｐコマンド以外は，ＰＧのコンソール・モードのコマンドと同じですので，詳細はＰＧの取扱説明書を参考にしてください。

表４－４　ＰＧ－２０００コマンド一覧

コマンド	形　式	コマンド　内容
A	*As,e,r ↵	パラメータ の設定をします。
E	*Er ↵	PGのバッファの データを変更します。 形式は以下のとおりです。 *Er ↵ r XX- YY- XX- YY XX- ↵ 入力データ 現在設定されているデータ データ 入力形式 ・データを入力する(16進以外を入力するとその時点で '?' 表示) ・スペース・キー入力(データ変更なし)
F	*Fr,re,d ↵	PGのバッファをd で初期化します。
I	*I ↵	インテリジェント・モード(コンソールへエコーバックするモード)へ移行します。
O	*Or,re ↵	PGのバッファの内容を表示します。 表示形式　r　00 00 00 00 00 00 …… 00
R	*Rr,re ↵	PROMの内容をPGのバッファへ転送します。
S	*S ↵	PROMの選択
T	*T ↵	トランジェント・モード(コンソール へエコーバックしないモード)へ移行します。
V	*Vs,e,r ↵	PROMの内容とPGバッファの内容との比較をします。 データ がちがうとき，'?' を表示します。
W	*Ws,e,r ↵	PGのバッファの内容をPROMに書き込みます。 書き込みエラー のとき，'?' を表示します。
Y	*Y ↵	現在設定されているパラメータ の表示をします。
Z	*Z ↵	PROMにデータ が書き込まれているかどうかをチェックします。 データ が書き込まれていない場合は何も表示せず，データ が書き込まれている場合は '?' を表示します。
L	*Lbias ↵ Partition = i,ie ↵	IEのマッピング されている i番地から，ie番地までの データ をPGバッファのi+bias番地から，ie+bias 番地まで転送します。
P	*Pr,re ↵ Bias = i ↵	PGのバッファの r番地からre番地までのデータをIE-78310A-Rのマッピング されている r+i番地からre+i番地まで転送します。

備考

s：ＰＲＯＭ開始番地
e：ＰＲＯＭ終了番地
r：ＰＧバッファ開始番地
re：ＰＧバッファ終了番地
i：ＩＥ開始番地
ie：ＩＥ終了番地
d：データ

エラー条件
Ｓ＞ｅ
ｒ＞ｒｅ
ｉ＞ｉｅ
１６進以外の入力

（b）アップ／ダウン・ロード・コマンド詳細

◆ＰＧ－１５００の場合（ＬＩコマンド・ＳＩコマンド詳細）

(i) ＬＩコマンド

```
PG>LI ↵
                                        YYYY‥‥IEﾒﾓﾘ の転送開始番地
  Partition = YYYY,ZZZZ ↵      ZZZZ‥‥IEﾒﾓﾘ の転送終了番地
PG>
```

ＩＥ－７８３１０Ａ－Ｒのマッピングされたメモリ（YYYYからZZZZ）の内容をＰＧのバッファへ転送します。

途中で実行を中止したいときは，ＥＳＣキーを入力してください。

ＥＳＣキーを入力すると，ＰＧのコマンド入力待ちになります。

パーティション入力を省略しますとマッピングされている全エリアを転送します。

例

① 通常のＬＩコマンドの使用例

```
PG>LI ↵                ←IEのﾒﾓﾘの0番地からFF番地の内容をPGへﾛｰﾄﾞ
  Partition = 0,0FF ↵
PG>
```

② エラー入力時

```
PG>LI ↵                ←Partitionの入力ｴﾗｰ

  Partition = 0,0FX ↵
  Input data error
  Partition =

PG>LI ↵                ←指定されたIEのﾒﾓﾘ がﾏｯﾋﾟﾝｸﾞ されていないので
                         ｴﾗｰになります。
  Partition = 0,0FF ↵
  Mapping error
  Partition =

PG>LI ↵                ←ｼﾘｱﾙが未接続状態の場合
                         (ﾊﾟﾘﾃｨ・ｴﾗｰ等)
  ERR40 (Serial not ready)
```

エラー表示後はＰＧコマンド入力待ちになります。

(ii) ＳＩコマンド

```
PG>SI XXXX,YYYY ↵        ←XXXX‥‥PGのﾊﾞｯﾌｧの転送開始番地
                           YYYY‥‥PGのﾊﾞｯﾌｧの転送終了番地
   Bias = ZZZZ ↵           ZZZZ‥‥IEのﾛｰﾄﾞ・ﾊﾞｲｱｽ
   complete                XXXX,YYYY,ZZZZは HEX数字で有効桁は下位
PG>                          4桁です。
```

ＰＧのメモリ（XXXXからYYYY）の内容をＩＥ－７８３１０Ａ－Ｒのマッピングされたメモリ（XXXX+ZZZZ)から(YYYY+ZZZZ）までに転送します。

途中で実行を中止したいときは，ＥＳＣキーを入力してください。

ＥＳＣキーを入力しますと，ＰＧのコマンド入力待ちとなります。

Ｂｉａｓの省略入力はできません。

例

① 通常のＳＩコマンドの使用例

```
PG>SI  0,FF ↵          ←PGの0番地からFF番地の内容をﾊﾞｲｱｽ0で
                          IEのの ﾒﾓﾘへ転送します。
    Bias = 0 ↵
    complete
PG>
```

② エラー表示

```
PG>SI  0,EF ↵          ←Biasの入力ｴﾗｰ

    Bias = X ↵ .
    Input data error
    Bias =

PG>SI  0,EF ↵          ←IEのﾒﾓﾘがﾏｯﾋﾟﾝｸﾞされていないのでｴﾗｰになります。
Bias = 1000 ↵
    Non map area access
PG>

PG>SI  0,1EF ↵         ←ﾃﾞｰﾀが転送中にﾁｪｯｸ・ｻﾑ・ｴﾗｰが生じた場合

    Bias = 25 ↵
    Check sum error

PG>SI 35,FF ↵          ←ﾃﾞｰﾀ転送中に16進数字以外のｷｬﾗｸﾀが転送された
                          場合

    Bias = 0 ↵
    Bad character
PG>
```

エラー表示後はＰＧのコマンド入力待ちになります。

◆ＰＧ－２０００の場合（Ｌコマンド・Ｐコマンド詳細）

(i) Ｌコマンド

```
*LXXX ↵                    ←XXXX‥‥PGのﾛｰﾄﾞ・ﾊﾞｲｱｽ
                             YYYY‥‥IEﾒﾓﾘの転送開始番地
  Partition = YYYY,ZZZZ ↵   ZZZZ‥‥IEﾒﾓﾘの転送終了番地
*
```

ＩＥ－７８３１０Ａ－Ｒのマッピングされたメモリ（YYYYからZZZZ）の内容をＰＧのバッファの（XXXX+YYYY)から(XXXX+ZZZZ）までに転送します。

途中で実行を中止したいときは，ＥＳＣキーを入力してください。

ＥＳＣキーを入力すると，ＰＧのコマンド入力待ちになります。

パーティション入力を省略しますとマッピングされている全エリアを転送します。

例

① 通常のＬコマンドの使用例

```
*L0>  )                    ←PGのﾒﾓﾘ・ﾊﾞｲｱｽを0にｾｯﾄIE のﾒﾓﾘの0番地からFF
                             番地の内容をPGへﾛｰﾄﾞ
  Partition = 0,0FF  )
*
```

② エラー入力時

```
*L0>  )                    ←Partitionの入力ｴﾗｰ

  Partition = 0,0FX  )
  Input data error
  Partition =

*L0>  )                    ←指定されたIEのﾒﾓﾘ がﾏｯﾋﾟﾝｸﾞ されていないので
                             ｴﾗｰ になります。
  Partition = 0,0FF  )
  Mapping error
  Partition =

*L0>  )                    ←IEからPGへﾃﾞｰﾀ 転送中にﾃﾞｰﾀ がうまく送れな
                             かった場合(ﾊﾟﾘﾃｨ・ｴﾗｰ 等)
  Partition = 0,0FF  )
  ?
  *L8000  )                ←PGのﾒﾓﾘ・ｱﾄﾞﾚｽをｵｰﾊﾞ してﾃﾞｰﾀ を転送しようとし
  ?                          たとき，または，途中でﾒﾓﾘ・ｱﾄﾞﾚｽをｵｰﾊﾞ したと
  *                          き
```

エラー表示後はＰＧコマンド入力待ち‘＊’になります。

(ii)　Ｐコマンド

```
*PXXXX,YYYY  )              ←XXXX・・・・PGのﾊﾞｯﾌｧの転送開始番地
                              YYYY・・・・PGのﾊﾞｯﾌｧの転送終了番地
    Bias = ZZZZ  )            ZZZZ・・・・IEのﾛｰﾄﾞ・ﾊﾞｲｱｽ
    complete                  XXXX,YYYY,ZZZZは HEX数字で有効桁は下位
*                             4桁です。
```

ＰＧのメモリ（XXXXからYYYY）の内容をＩＥ－７８３１０Ａ－Ｒのマッピングされたメモリ（XXXX+ZZZZ)から(YYYY+ZZZZ）までに転送します。転送が正常に終了した場合は‘＊’を表示します。

途中で実行を中止したいときは，ＥＳＣキーを入力してください。ＥＳＣキーを入力しますと，ＰＧのコマンド入力待ちとなります。

Ｂｉａｓの省略入力はできません。

例

① 通常のＰコマンドの使用例

```
*P0,FF  ↵                    ←PGの0番地からFF番地の内容をﾊﾞｲｱｽ0でIE-78310
                               A-Rのﾒﾓﾘへ転送します。
    Bias = 0  ↵
    complete
*
```

② エラー表示

```
*P0,EF  ↵                    ←Biasの入力ｴﾗｰ

    Bias = X  ↵ .
    Input data error
    Bias =

*P0,EF  ↵                    ←IE-78310A-R の ﾒﾓﾘが ﾏｯﾋﾟﾝｸﾞされていないので
                               ｴﾗｰ になります。
    Bias = 100  ↵
    Non map area access
*

*P0,1EF  ↵                   ←ﾃﾞｰﾀ が転送中にﾁｪｯｸ･ｻﾑ･ｴﾗｰ が生じた場合

    Bias = 25  ↵
    Check sum error

*P0,8000 F↵                  ←PGのﾒﾓﾘ･ｱﾄﾞﾚｽより大きなｱﾄﾞﾚｽを設定しようとし
                               たとき
?
*
```

エラー表示後はＰＧのコマンド入力待ち‘＊’になります。

ＰＧＭ＿Ｃ ……カレント制御キャラクタ変更コマンド

カレント制御キャラクタ変更コマンドは，ＰＧＭモードにおける制御キャラクタを対話形式で任意のキャラクタに変更することができます。

任意のキャラクタとして以下の１６種類のキャラクタが使用できます。

使用可能キャラクタ：

ｃｔｒｌ－Ａ	（０１Ｈ）
ｃｔｒｌ－Ｂ	（０２Ｈ）
ｃｔｒｌ－Ｅ	（０５Ｈ）
ｃｔｒｌ－Ｆ	（０６Ｈ）
ｃｔｒｌ－Ｇ	（０７Ｈ）
ｃｔｒｌ－Ｎ	（０Ｅ Ｈ）
ｃｔｒｌ－Ｏ	（０Ｆ Ｈ）
ｃｔｒｌ－Ｐ	（１０Ｈ）
ｃｔｒｌ－Ｒ	（１２Ｈ）
ｃｔｒｌ－Ｔ	（１４Ｈ）
ｃｔｒｌ－Ｕ	（１５Ｈ）
ｃｔｒｌ－Ｖ	（１６Ｈ）
ｃｔｒｌ－Ｗ	（１７Ｈ）
ｃｔｒｌ－Ｘ	（１８Ｈ）
ｃｔｒｌ－Ｙ	（１９Ｈ）
ｃｔｒｌ－Ｚ	（１ＡＨ）

制御キャラクタを一通り変更するとＰＧＭモードになりますので，制御キャラクタ変更後ＰＧＭコマンドを入力する必要はありません。

同一のキャラクタを複数の機能の制御キャラクタとして設定することはできません。

ＤＥＬキーあるいはｃｔｒｌ－Ｈ入力の場合はディフォールト値に変換します。

ＥＳＣキー入力による中断の場合はそれまでに変換したキャラクタを無効とします。

例1　*PGM C ↵

```
Termination   of "PGM"        ··· ^Z
Beginning     of "HEX LOAD"   ··· ^A
Beginning     of "HEX SAVE"   ··· ^E
Beginning     of "SYM LOAD"   ··· ^N
Termination   of "LOAD"       ··· ^B
Termination   of "SAVE"       ··· ^F
Break         of "LOAD/SAVE"  ··· ^W
```

ディフォールト 値を表示

```
Termination   of "PGM"        ··· ^Z■
```

現在の値を表示し入力待ちになります。

この状態で任意のｷｬﾗｸﾀ を入力します。

```
Termination   of "PGM"        ··· ^A↵   ctrl-Zをctrl-Aに変更
Beginning     of "HEX LOAD"   ··· ^Z↵   ctrl-Aをctrl-Zに変更
Beginning     of "HEX SAVE"   ··· ^N↵   ctrl-Eをctrl-Nに変更
Beginning     of "SYM LOAD"   ··· ^E↵   ctrl-Nをctrl-Eに変更
Termination   of "LOAD"       ··· ^B↵   ﾘﾀｰﾝｷｰのみの入力で変更しません
Termination   of "SAVE"       ··· ^X↵   ctrl-Fをctrl-Xに変更
Break         of "LOAD/SAVE"  ··· ^G↵   ctrl-Wをctrl-Gに変更
```

一通り変更しますと次のﾒｯｾｰｼﾞ を表示しPGMﾓｰﾄﾞになります。

```
Beginning     of PGM mode
```

例2　DEL Key を使用し，ﾃｨﾌｫｰﾙﾄ 値に変換する場合

```
*PGM C ↵
```

```
Termination   of "PGM"        ··· ^A
Beginning     of "HEX LOAD"   ··· ^Z
Beginning     of "HEX SAVE"   ··· ^N
Beginning     of "SYM LOAD"   ··· ^E
Termination   of "LOAD"       ··· ^B
Termination   of "SAVE"       ··· ^X
Break         of "LOAD/SAVE"  ··· ^G

Termination   of "PGM"        ··· ^A■
```

この状態でDEL Key を入力すると，ﾃｨﾌｫｰﾙﾄ 値を表示します。

```
Termination   of "PGM"        ··· ^Z ↵
Beginning     of "HEX LOAD"   ··· ^A ↵
Beginning     of "HEX SAVE"   ··· ^E ↵
Beginning     of "SYM LOAD"   ··· ^N ↵
Termination   of "LOAD"       ··· ^B ↵
Termination   of "SAVE"       ··· ^F ↵
Break         of "LOAD/SAVE"  ··· ^W ↵

Beginning     of PGM mode
```

DEL Key 入力により
ﾃｨﾌｫｰﾙﾄ 値に変換

例3　ESC 入力による判断の場合

```
*PGM C ↵
```

```
Termination   of "PGM"        ··· ^Z
Beginning     of "HEX LOAD"   ··· ^A
Beginning     of "HEX SAVE"   ··· ^E
Beginning     of "SYM LOAD"   ··· ^N
Termination   of "LOAD"       ··· ^B
Termination   of "SAVE"       ··· ^F
Break         of "LOAD/SAVE"  ··· ^W

Termination   of "PGM"        ··· ^E ↵
Beginning     of "HEX LOAD"   ··· ^G ↵
Beginning     of "HEX SAVE"   ··· ^W ↵
Beginning     of "SYM LOAD"   ··· ^A■ ← ESCｷｰ入力
*
```

ESCｷｰ入力により中断をした場合，変更した部分は無効となります。

```
*PGM C ↵
```

```
Termination   of "PGM"        ··· ^Z
Beginning     of "HEX LOAD"   ··· ^A
Beginning     of "HEX SAVE"   ··· ^E
Beginning     of "SYM LOAD"   ··· ^N
Termination   of "LOAD"       ··· ^B
Termination   of "SAVE"       ··· ^F
Break         of "LOAD/SAVE"  ··· ^W

Termination   of "PGM"        ··· ^Z■
```

ESCｷｰで中断した場合
ｷｬﾗｸﾀは変更されません。

例4　同一のｷｬﾗｸﾀ を複数の制御ｷｬﾗｸﾀ として設定しようとする場合
*PGM C ↵

```
Termination   of "PGM"        ··· ^Z
Beginning     of "HEX LOAD"   ··· ^A
Beginning     of "HEX SAVE"   ··· ^E
Beginning     of "SYM LOAD"   ··· ^N
Termination   of "LOAD"       ··· ^B
Termination   of "SAVE"       ··· ^F
Break         of "LOAD/SAVE"  ··· ^W

Termination   of "PGM"        ··· ^A ↵
Beginning     of "HEX LOAD"   ··· ^Z ↵
Beginning     of "HEX SAVE"   ··· ^W ↵
Beginning     of "SYM LOAD"   ··· ^E ↵
Termination   of "LOAD"       ··· ^A ↵
Termination   of "SAVE"       ··· ^N ↵
Break         of "LOAD/SAVE"  ··· ^B ↵
```

同一のｷｬﾗｸﾀ を複数の機能に設定すると，PGMﾓｰﾄﾞにならず再度変更ｷｬﾗｸﾀ 入力待ちとなる。

```
Termination   of "PGM"        ··· ^A--- Multi define
Beginning     of "HEX LOAD"   ··· ^Z
Beginning     of "HEX SAVE"   ··· ^W
Beginning     of "SYM LOAD"   ··· ^E
Termination   of "LOAD"       ··· ^A--- Multi define
Termination   of "SAVE"       ··· ^N
Break         of "LOAD/SAVE"  ··· ^B

Termination   of "PGM"        ··· ^A■  再度ｷｬﾗｸﾀ 入力待ちとなる。
```

４．５．２５　レジスタ操作（ＲＥＧ）

```
REG_C[_register name]

REG[_D[_ {ALL           }]]
         {register name }
```

C　　：ﾚｼﾞｽﾀの変更
D　　：ﾚｼﾞｽﾀの表示
ALL　　：全ﾚｼﾞｽﾀ･ﾊﾞﾝｸの全ﾚｼﾞｽﾀ指定
register name：以下のﾚｼﾞｽﾀ名が指定できます

制御ﾚｼﾞｽﾀ：PC,SP,PSW
汎用ﾚｼﾞｽﾀ：R0,R1,R2,R3,R4,R5,R6,R7,RP4,RP5,RP6,RP7,X,A,B,C,VP,UP,DE,HL
PSWﾌﾗｸﾞ名：RBS2,RBS1,RBS0,IE,S,Z,RSS,AC,UF,P/V,SUB,CY

レジスタ操作コマンドは，モード・レジスタ，および，特殊レジスタ以外の汎用レジスタ，および，制御レジスタの変更，表示をすることができます。制御レジスタの“ＰＳＷ”（プログラム・ステータス・ワード）については，フラグ単位での変更，および，表示をすることができます。

（１）レジスタの変更

ＲＥＧ_C[_register name]

レジスタの変更は，レジスタ名で指定されたレジスタの内容を変更することができます。レジスタ名に汎用レジスタ，あるいは，制御レジスタが指定された場合は，指定されたレジスタから順番に変更することができます。また，指定されたレジスタが“ＰＳＷ”（プログラム・ステータス・ワード）の場合は，フラグ単位で変更することができます。フラグは，“ＲＢＳ２”から“ＣＹ”の順にフラグ名ごとに変更することができます。また，フラグ名を直接指定することにより，一つのフラグだけを変更することもできます。

レジスタ名が省略された場合は，汎用レジスタの“Ｒ０”が指定された場合と同じになります。この場合は，“Ｒ０”から“ＳＰ”の順番に変更することができます。ただし，“Ｒ０”から“Ｒ７”までは８ビット単位で，また，“ＲＰ４”から“ＳＰ”までは１６ビット単位で変更されます。レジスタの変更を途中で終わる場合は，‘.’（ピリオド）を入力します。また，レジスタの内容を変更しない場合は，‘↵’（リターン・キー）を入力します。

例

```
*REG C ↵                          ←すべての汎用ﾚｼﾞｽﾀの変更
 R0(X)       00 = 11 ↵
 R1(A)       11 = 22 ↵
 R2(C)       22 = 33 ↵
 R3(B)       33 = 44 ↵
 R4          44 = ↵                ←ﾚｼﾞｽﾀの変更なし
 R5          55 = 66 ↵
 R6          66 = 77 ↵
 R7          77 = 88 ↵
 RP4(VP)   1000 = 2000 ↵
 RP5(UP)   2000 = 3000 ↵
 RP6(DE)   3000 = 4000 ↵
 RP7(HL)   4000 = 5000 ↵
 PC        0123 = ↵
 SP        8000 = ↵
*

*REG C PSW ↵                       ←PSW のﾌﾗｸﾞ 名単位に変更
 RBS2         1 = 0 ↵
 RBS1         1 = 0 ↵
 RBS0         1 = ↵                ←変更なし
 IE           0 = 1 ↵
 S            1 = 0 ↵
 Z            0 = 1 ↵
 RSS          0 = 1 ↵
 AC           0 = 0 ↵
 UF           1 = 0 ↵
 P/V          0 = 1 ↵
 SUB          1 = 0 ↵
 CY           0 = 1 ↵
*
```

```
*REG C SP )                    ←制御ﾚｼﾞｽﾀ"SP"を変更
 SP        8000 = 7000 )
*

*REG C PC )                    ←制御ﾚｼﾞｽﾀ"PC"を変更
 PC        0123 = 1000 )
 SP        7000 = )
*

*REG C RP4 )                   ←汎用ﾚｼﾞｽﾀ(ﾍﾟｱ･ﾚｼﾞｽﾀ) の変更
 RP4(VP)   1000 = 2000 )
 RP5(UP)   2000 = 3000 )
 RP6(DE)   3000 = 4000 )
 RP7(HL)   4000 = 5000 )
 PC        0123 = )
 SP        7000 = )
*

*REG C R0 )                    ←汎用ﾚｼﾞｽﾀの変更
 R0          00 = 30 )
 R1          11 = 40 )
 R2          22 = 50 )
 R3          33 = . )
*
```

汎用ﾚｼﾞｽﾀの変更は，ﾍﾟｱ･ﾚｼﾞｽﾀ “ＲＰ４”から“ＲＰ７”までと，“ＰＣ”，“ＳＰ”は，16ﾋﾞｯﾄ単位で変更できます。“Ｒ０”から“Ｒ７”までは，8ﾋﾞｯﾄ単位で変更できます。

```
*REG C VP )                    ←機能ﾚｼﾞｽﾀ (ﾍﾟｱ･ﾚｼﾞｽﾀ)名で変更
 RP4(VP)   2000 = 1000 )
 RP5(UP)   3000 = . )
*

*REG C X )                     ←機能ﾚｼﾞｽﾀ名で変更
 R4(X)       10 = 20 )
 R5(A)       20 = 30 )
 R6(C)       30 = . )
*
```

汎用ﾚｼﾞｽﾀと対応する機能ﾚｼﾞｽﾀ名でも変更することができます。

```
*REG C RSS )                   ←"PSW" のﾌﾗｸﾞ 名で変更
 RSS          1 = 0 )
*
```

“PSW ” のﾌﾗｸﾞ 名が指定された場合は，指定されたﾌﾗｸﾞ だけを変更します。

（２）レジスタの表示

```
REG[_D[_ {ALL                } ]]
         {register name      }
```

レジスタの表示は，レジスタ名で指定されたレジスタの内容を表示することができます。レジスタ名が省略された場合は，現在選択されているレジスタ・バンクのすべてのレジスタの内容が表示されます。ＡＬＬを指定すると，全レジスタ・バンク（０～７）のすべてのレジスタ内容が表示されます。ただし，“ＰＳＷ”が指定された場合は，“ＰＳＷ”のすべてのフラグの内容が表示されます。

例

```
*REG D ALL ↵        ←すべてのﾚｼﾞｽﾀの内容を表示
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 1000  7000        0    0    1    1    0    1    0    0    0    1    0    1

         R0  R1  R2  R3  R4  R5  R6  R7        RP4     RP5     RP6     RP7
  BANK0  01  02  03  04  05  06  07  08       0909    0A0A    0B0B    0C0C
  BANK1  11  12  13  14  15  16  17  18       1919    1A1A    1B1B    1C1C
  BANK2  21  22  23  24  25  26  27  28       2929    2A2A    2B2B    2C2C
  BANK3  31  32  33  34  35  36  37  38       3939    3A3A    3B3B    3C3C
  BANK4  98  76  54  32  10  AA  BB  CC       DEF0    5562    1F20    3434
  BANK5  00  00  00  00  22  3F  1C  52       0006    AEFC    7000    1020
  BANK6  11  22  33  44  55  66  77  88       9999    AAAA    BBBB    CCCC
  BANK7  71  72  73  74  75  76  77  78       7979    7A7A    7B7B    7C7C

*REG D PSW ↵  ←“PSW” のすべてのﾌﾗｸﾞ の内容を表示
 PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
       0    0    1    1    0    1    0    0    0    1    0    1
*

*REG D ↵        ←ｶﾚﾝﾄ･ﾊﾞﾝｸにおける全ﾚｼﾞｽﾀ内容を表示
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 1000  7000        0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7        RP4     RP5     RP6     RP7
    X    A    C    B                           VP      UP      DE      HL
   30   10   20   30   20   20   60   70       1000    2000    3000    4000
*

*REG D PC ↵  ←“PC”の内容を表示
 PC        1000
*

*REG D R0 ↵  ←“R0”の内容を表示
 R0(X)       30
*

*REG D VP ↵  ←“VP(機能ﾚｼﾞｽﾀ名)”の内容を表示
 RP4(VP)  1000
*

*REG D RSS ↵  ←“RSS”ﾌﾗｸﾞの内容を表示
 RSS          0
*
```

４．５．２６　エミュレーション操作（ＲＵＮ）

ＲＵＮ_Ｂ[_word]　　（ブレーク付きリアルタイム実行）
ＲＵＮ_Ｎ[_word]　　（ブレークなしリアルタイム実行）
ＲＵＮ_Ｓ[_[word][,word]]（ステップ数指定リアルタイム実行）
ＲＵＮ_Ｔ[_[word][,{※ / word}][_TRD][_REG]]　（トレース実行）

※・・・・register name { = / > / < / => / >= / =< / <= / <> / >< }　{ byte data / word data }

register name { = / <> / >< }　{ mask / wmask }

word	：実行ｽﾀｰﾄ･ｱﾄﾞﾚｽ
,word	：実行ｽﾃｯﾌﾟ数
TRD	：ﾄﾚｰｽ表示指定
REG	：ﾚｼﾞｽﾀ表示指定
mask	：8ﾋﾞｯﾄ･ﾏｽｸ･ﾃﾞｰﾀ
wmask	：16ﾋﾞｯﾄ･ﾏｽｸ･ﾃﾞｰﾀ
register name	：R0,R1,R2,R3,R4,R5,R6,R7,RP4,RP5,RP6,RP7, X,A,B,C,VP,UP,DE,HL,PC,SP, RBS2,RBS1,RBS0,IE,S,Z,RSS,AC,UF,P/V,SUB,CY

エミュレーション・コマンドは，ユーザ・プログラムの実行をします。ユーザ・プログラムの実行スタート・アドレスは，オペランドによって指定することができます。ただし，０ＦＥ７ＦＨをこえたアドレスを指定することはできません。

なお，ユーザ・プログラムは，ＲＵＮ＿Ｔコマンドを除いて，リアルタイムで実行されます。

注意　ＩＥ－７８３１０Ａ－Ｒでは，μＰＤ７８３１２Ａのアーキテクチャ上，以下のように使用上の注意が必要です。

・ブランチ系命令のオペランドではブレークしません。
したがって，ブランチ系命令のオペランドにはブレーク・ポイントを置かないでください。

・ブレーク・ポイントを通過後，数命令実行してから，ブレークします（これをスリップといいます）。
スリップする命令数は，一定していませんが，ブレーク・ポイント後，１バイト命令が続いているとき最も多くスリップします。
また，同じ条件でも内部ＲＯＭ実行時と，外部メモリ実行時でも，スリップする命令数が異なります。
内部ＲＯＭ実行時は，外部メモリ実行時よりも，多くスリップします。

・命令ステップ数をカウントする場合，ＳＦＲに対する一部の命令は，２ステップと数えられます。詳しくは，第４章　４．７　オンライン・アセンブラ／逆アセンブラ仕様の命令一覧表をご覧ください。

・ＣＡＬＬＴ，ＢＲＫ命令あるいは，割込みによるベクタ参照では正しくブレークしません。したがって，ベクタ・エリアにブレーク・ポイントを置かないでください。

（１）ブレークなしリアルタイム実行

RUN_N [_word]

オペランドで指定された実行スタート・アドレスからユーザ・プログラムを実行します。オペランドの実行スタート・アドレスが省略された場合は，現在のプログラム・カウンタが実行スタート・アドレスになります。

プログラムの実行を停止させる場合は，ＥＳＣキーを入力します。ＥＳＣキーにより実行を停止した場合は，停止した時点でのレジスタの内容が表示されます。

例

```
*RUN N 200 ↵          ←200 番地からﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ を実行
 User-system Vcc-ON     Emulation start at 0200
                                          ↑
ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ 実行                          └─ｶｰｿﾙの位置
```

上記の表示をしてユーザ・プログラムが実行されます。ユーザ・プログラムの実行を停止させる場合は，カーソルの位置でＥＳＣキーを入力します。以下にその例を示します。

```
*RUN N 200 ↵
 User-system Vcc-ON     Emulation start at 0200 ←ESC ｷｰを入力
 ESC break terminated     ←停止ﾒｯｾｰｼﾞ
  PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 1000  7000        0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
    X    A    C    B                             VP        UP        DE        HL
   30   10   20   30   20   20   60   70      1000      2000      3000      4000
*
```

ＥＳＣキーによってユーザ・プログラムを停止させた場合は，上記のように停止した時点のレジスタの内容が表示されます。

（2）ブレーク付きリアルタイム実行

RUN_B [_word]

オペランドで指定された実行スタート・アドレスからブレーク条件指定コマンド(BRM コマンド)で設定されたブレーク条件と一致するまでユーザ・プログラムを実行します。ブレーク条件指定コマンドで条件が設定されていない場合は，ブレークはしません。オペランドの実行スタート・アドレスが省略された場合は，現在のプログラム・カウンタが実行スタート・アドレスになります。

ブレーク条件が一致した場合は，その時点のレジスタの内容が表示されます。

その後，ワン・ステップ実行モードとなります。

ユーザ・プログラムがブレーク条件と一致しないような場合は，プログラムの実行を強制的に停止させることができます。強制的にプログラムの実行を停止させるには，ＥＳＣキーを入力します。この場合も，停止した時点のレジスタの内容が表示されます。ただし，ＥＳＣキーで停止した場合は，ワン・ステップ実行モードにはなりません。

例

```
*RUN B 100 ↵            ←100 番地からユーザ・プログラム を実行
User-system Vcc-ON    Emulation start at 0100
Standard break  terminated  ←ブレーク条件が一致，ユーザ・プログラム の実行を停止
 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0109  7000       0    0    1    1    0    1    0    0    0    1    0    1
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
   X    A    C    B                             VP        UP        DE        HL
  30   10   20   30   20   20   60   70      1000      2000      3000      4000
One step emulation standby
                          ↑
                          └──カーソルの位置
```

上記のようにブレーク条件と一致した場合は，ユーザ・プログラムの実行を停止し，ワン・ステップ実行モードになります。カーソル位置でリターン・キーを入力しますと，ユーザ・プログラムの次のワン・ステップを実行します。

```
 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0109  7000       0    0    1    1    0    1    0    0    0    1    0    1
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
   X    A    C    B                             VP        UP        DE        HL
  30   10   20   30   20   20   60   70      1000      2000      3000      4000
One step emulation standby← ↵ キー を入力
```

```
Frame Status Address Data Label  Mnemonic                P2 EX
0000         010A               MOV     [HL+],A
0002     WR  4000    10                                  00 00
0001   MSWR  2002    7F                                  00 00
 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB CY
0109  7000       0    0    1    1    0    1    0    0    0    1    0    1
  R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
   X    A    C    B                             VP       UP       DE       HL
  30   10   20   30   20   20   60   70       1000     2000     3000     4001
One step emulation standby  ←ESC キーを入力
 *
```

ワン・ステップ実行モードは，‘One step emulation standby’の表示の後にリターン・キーを入力することによって１命令を実行します。実行内容は，トレースされ結果を逆アセンブルし表示します。また，レジスタ内容も表示されます。

ワン・ステップ実行モードを終了させるには，ＥＳＣキーを入力します。ただし，マッピングされていないエリアをアクセスした場合は，自動的にワン・ステップ実行モードを終了します。

（3）ステップ数指定リアル・タイム実行

```
RUN_S [_ [word] [, word]]
```

オペランドで指定された実行スタート・アドレスから，実行ステップ数分をリアル・タイムで実行します。実行スタート・アドレスが省略された場合は，現在のプログラム・カウンタが実行スタート・アドレスになります。実行ステップ数が省略された場合は，ワン・ステップ実行モードになります。

また，指定された実行ステップ数の実行が終了した場合も，ワン・ステップ実行モードになります。

プログラムの実行を強制的に停止する場合は，ＥＳＣキーを入力します。この場合は，ワン・ステップ実行モードにはなりません。

例

```
*RUN S 100,50T ↵          ←100 番地からﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ を50Tｽﾃｯﾌﾟ を実行
User-system Vcc-ON    Emulation start at 0100
Step break  terminated  ←指定ｽﾃｯﾌﾟ数を実行し停止
 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0109  7000       0    0    1    1    0    1    0    0    0    1    0    1
  R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
   X    A    C    B                             VP       UP       DE       HL
  30   10   20   30   20   20   60   70       1000     2000     3000     4000
One step emulation standby
                                  ↑
                                  └──ｶｰｿﾙの位置
```

上記のように実行ステップ数の実行を終了した場合は，ユーザ・プログラムの実行を停止し，ワン・ステップ実行モードになります。カーソル位置でリターン・キーを入力すると，ユーザ・プログラムの次のワン・ステップを実行します。

ワン・ステップ実行モードを終了する場合は，ＥＳＣキーを入力します。

（４）トレース実行

```
RUN_T[_[word][,{※   }][_TRD][_REG]]
               {word}
```

※・・・・register name {=, >, <, =>, >=, =<, <=, <>, ><} {byte, word}

register name {=, <>, ><} {mask, wmask}

オペランドで指定された実行スタート・アドレスから，実行ステップ数，あるいは，レジスタの条件が一致するまでユーザ・プログラムを実行します。ユーザ・プログラムの実行は，リアル・タイムには実行されません。

実行命令ごとに，レジスタの内容，トレース結果，および，実行結果の逆アセンブルを表示します。指定された実行ステップ数，あるいは，レジスタ条件が一致した場合は，ワン・ステップ実行モードになります。

実行スタート・アドレスが省略された場合は，現在のプログラム・カウンタの内容が実行スタート・アドレスになります。実行ステップ数，あるいは，レジスタの条件が省略された場合は，ワン・ステップ実行モードになります。また；トレース表示指定（TRD ）が省略された場合は，トレース結果，および，実行結果の逆アセンブルの表示はされません。レジスタ表示指定（REG ）が省略された場合は，レジスタ内容の表示はされません。レジスタ名に“ＰＳＷ”のフラグ名が指定された場合は，‘フラグ名＝０，または，１’以外の条件を設定することはできません。

プログラムの実行を強制的に停止する場合は，ＥＳＣキーを入力します。この場合は，ワン・ステップ実行モードにはなりません。

例

```
*RUN T 10A,R0=1 TRD REG↵←10A 番地からﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ を実行
 User-system Vcc-ON    Emulation start at 010A
 Frame Status Address Data Label  Mnemonic                P2 EX
 0000          010A               MOV     [HL+],A
 0003     WR   1000    02                                  00 00
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 010B  7000       0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
    X    A    C    B                           VP       UP       DE       HL
   02   02   20   30   20   20   60   70       1000     2000     3000     1001

 Frame Status Address Data Label  Mnemonic                P2 EX
 0000   MSRD   2101    FF                                  00 00
 0001          010B               DEC     R0
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 010C  7000       0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
    X    A    C    B                           VP       UP       DE       HL
   01   02   20   30   20   20   60   70       1000     2000     3000     1001

   terminated
 One step emulation standby←↵ ｷｰ を入力
 Frame Status Address Data Label  Mnemonic                P2 EX
 0000          010C               MOV     [HL+],A
 0003     WR   1001                                        00 00
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 010D  7000       0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
    X    A    C    B                           VP       UP       DE       HL
   01   02   20   30   20   20   60   70       1000     2000     3000     1002
 One step emulation standby←ESCｷｰ を入力
*

*RUN T 10A,R0=1↵          ←10A 番地からﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ を実行
 User-system Vcc-ON    Emulation start at 010A
   terminated
 One step emulation standby←↵ ｷｰ を入力
 Frame Status Address Data Label  Mnemonic                P2 EX
 0000          010C               MOV     [HL+],A
 0003     WR   1001                                        00 00
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
 010D  7000       0    0    1    1    0    1    0    0    0    1    0    1
   R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
    X    A    C    B                           VP       UP       DE       HL
   01   02   20   30   20   20   60   70       1000     2000     3000     1002
 One step emulation standby←ESCｷｰ を入力
*
```

４．５．２７　リセット（ＲＥＳ）

```
RES [_H]
```

H　　：IEすべてのﾘｾｯﾄ

リセット・コマンドは，ＥＶＡＣＨＩＰのリセット，あるいは，ＩＥシステムすべてのリセットをすることができます。オペランドを省略した場合は，ＥＶＡＣＨＩＰだけをリセットします。オペランドに“Ｈ”を指定した場合は，ＩＥシステムすべてをリセットします。

例

```
*RES H ↵          ←IE ｼｽﾃﾑ をすべてﾘｾｯﾄする
 IE-78310 Monitor Vx.x  [Dd Mmm Yy]
   Copyright (C) 1985 by NEC Corporation

 Power on target system (Y/N) Y↵
 Internal ROM size (4K,8K,16K) = 8K ↵
 Tracer  initialize
 Breaker initialize
 Do you have Memory Board on IE-78310 ? (Y/N) = N↵
*
```

ＩＥシステムのすべてをリセットします。タイトル・メッセージから表示され，もう一度，条件を設定できます。

```
*RES ↵            ←EVACHIP だけをﾘｾｯﾄする
*
```

ＥＶＡＣＨＩＰだけをリセットします。ＩＥシステムは影響を受けません。

４．５．２８　オブジェクト・セーブ（ＳＡＶ）

```
SAV [_ {TTY1} ] [_partition][_partition]·····[_partition]
       {TTY2}

SAV_file name[_partition][_partition]·····[partition]
```

TTY1　　：ｼﾘｱﾙ･ﾁｬﾈﾙ1
TTY2　　：ｼﾘｱﾙ･ﾁｬﾈﾙ2
file name　：ﾌｧｲﾙ名
partition　：ﾒﾓﾘ 範囲
partition は最大五つまで指定できる

オブジェクト・セーブ・コマンドは，マッピングされたメモリの内容を外部装置に出力することができます。スタンドアロン・モードと，システム・モードではコマンドが異なります。

スタンドアロン・モードの場合は，シリアル・チャネル１，あるいは，シリアル・チャネル２にオブジェクト・コードを出力することができます。

システム・モードの場合は，ホスト・マシンのファイルにオブジェクト・コードを出力することができます。

オブジェクト・コードは，ヘキサ形式で出力されます。

（１）システム・モードのオブジェクト・セーブ

```
ＳＡＶ_file name[_partition][_partition]・・・・[partition]
```

システム・モードのオブジェクト・セーブは，パーティションで指定されたメモリの内容をホスト・マシンに指定されたファイル名で出力します。パーティションが指定されていない場合は，マッピングされているメモリすべてが出力されます。また，この場合は，０ＦＥ００～０ＦＥ７Ｆ番地の内容も同時に出力されます。

指定するパーティションは，マッピング・コマンドでマッピングされていなければなりません。もし，マッピングされていないメモリを指定した場合は，エラーになります。

例

```
1>SAV B:SAMPLE.HEX↵          ←ﾏｯﾋﾟﾝｸﾞ されているすべてのﾒﾓﾘ 内容をｾｰﾌﾞ する
  object save complete        ←正常終了ﾒｯｾｰｼﾞ
1>

1>SAV B:SAMPLE.HEX 0XXX 2000,27FF ↵ ←出力範囲(0～0FFF，2000～27FF番地)
  object save complete                の内容をｾｰﾌﾞ する
1>

1>SAV B:SAMPLE.HEX↵
  File already exists. Delete ?(Y or N): Y↵
  object save complete
1>
```

ﾄﾞﾗｲﾌﾞ Bに SAMPLE.HEX というﾌｧｲﾙがすでに存在している場合に，上記のようなﾒｯｾｰｼﾞ が出力されます。このとき，Y を入力しますと，まず SAMPLE.HEX というすでに存在しているﾌｧｲﾙをﾃﾞﾘｰﾄします。そして，新たに SAMPLE.HEX というﾌｧｲﾙをｵｰﾌﾟﾝします。また，Y 以外が入力された場合は，ｺﾏﾝﾄﾞは無視されます。ただし，すでに存在するﾌｧｲﾙの属性が DIR属性でかつ R/W属性の場合だけです。

もし，ﾌｧｲﾙの属性が SYS属性，あるいは， R/O属性の場合は，下記のようなﾒｯｾｰｼﾞ を出力してｺﾏﾝﾄﾞは，無視されます。

```
1>SAV B:SAMPLE.HEX↵
  File already exists.
1>
```

（２）スタンドアロン・モードのオブジェクト・セーブ

```
SAV [__ {TTY1} ] [_partition][_partition]・・・・・[partition]
        {TTY2}
```

スタンドアロン・モードの場合は，指定されたシリアル・チャネルに，オブジェクト・コードを出力することができます。指定された範囲のメモリ内容をすべて出力します。

シリアル・チャネルが省略された場合は，シリアル・チャネル１が指定されます。また，メモリの範囲が省略された場合は，マッピングされているメモリすべてが指定されます。なお，この場合は，内部ＲＡＭ空間の０ＦＥ００番地から０ＦＥ７Ｆ番地の内容も同時に出力されます。

例

```
*SAV TTY1 0XXX 1800,1FFF ↵      ←ｼﾘｱﾙ･ﾁｬﾈﾙ1に 0番地から0FFF番地までと，
                                   1800番地から1FFF番地までを出力
 [ ｼﾘｱﾙ･ﾁｬﾈﾙ 1 に ﾍｷｻ形式ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞを出力
*
*SAV TTY2 0XXX 1800,1FFF ↵      ←ｼﾘｱﾙ･ﾁｬﾈﾙ2に 0番地から0FFF番地までと，
                                   1800番地から1FFF番地までを出力
 [ ｼﾘｱﾙ･ﾁｬﾈﾙ 2 に ﾍｷｻ形式ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞを出力
EOF character output(Y/N) Y ↵   ← Y ↵ を入力した場合のみEOF(1A)ｷｬﾗｸﾀが
*                                  ｼﾘｱﾙ･ﾁｬﾈﾙ 2 に出力される
```

EOF character output(Y/N) のﾒｯｾｰｼﾞ は，ｺﾝｿｰﾙ に出力されます。 Y ↵ が入力された場合のみｼﾘｱﾙ･ﾁｬﾈﾙ2にEOFｷｬﾗｸﾀが出力されます。Y 以外が入力された場合は，EOFｷｬﾗｸﾀは出力されません。

ｼﾘｱﾙ･ﾁｬﾈﾙ1(ｺﾝｿｰﾙ) が指定された場合は，EOFｷｬﾗｸﾀの出力はしません。

４．５．２９　特殊レジスタ操作（ＳＰＲ）

```
ＳＰＲ_C[_special register name]
ＳＰＲ[_D[_special register name]]
```

```
 C           : 特殊レジスタの変更
 D           : 特殊レジスタの表示
special register name:              *    *    *    *    *    *    *    *
                P0,P1,P2,P3,P4,P5,CR00,CR01,CR10,CR11,CPT0,CPT1,PWM0,PWM1,
                   *    *                                    *    *    *    *
                UDC0,UDC1,POL,POH,BRG,(RXB),<TXB>,(ADCR),TM0,MD0,TM1,MD1

                EXTSFR0,EXTSFR1,EXTSFR2,EXTSFR3,EXTSFR4,EXTSFR5,EXTSFR6,

                EXTSFR7,EXTSFR8,EXTSFR9,EXTSFR10,EXTSFR11,EXTSFR12,

                EXTSFR13,EXTSFR14,EXTSFR15

 *  で示されたﾚｼﾞｽﾀは，16ﾋﾞｯﾄ･ﾚｼﾞｽﾀ，
( ) で示されたﾚｼﾞｽﾀは，読出し専用ﾚｼﾞｽﾀ，
< > で示されたﾚｼﾞｽﾀは，書込み専用ﾚｼﾞｽﾀ，
上記以外は，8ﾋﾞｯﾄ･ﾚｼﾞｽﾀ である。
```

特殊レジスタ操作コマンドは，特殊レジスタの変更，および，表示をすることができます。８ビット・レジスタは，８ビット単位で，また，１６ビット・レジスタは，１６ビット単位でそれぞれ変更，および，表示をすることができます。

特殊レジスタを変更する場合にレジスタ名を省略した場合は，すべての特殊レジスタを順番に変更することができます。ただし，読出し専用レジスタを変更することはできません。変更する順序は，“Ｐ０”から“ＥＸＴＳＦＲＩＳ”の順番になります。レジスタ名が指定された場合は，指定されたレジスタから順番に変更することができます。

特殊レジスタの内容を表示する場合に，レジスタ名を省略しますと，すべての特殊レジスタの内容を表示することができます。ただし，書込み専用レジスタは，表示されません。レジスタ名が指定された場合は，指定されたレジスタの内容だけが表示されます。

例

・特殊ﾚｼﾞｽﾀの変更

```
*SPR C ↵                          ←特殊ﾚｼﾞｽﾀすべての変更
 P0          77 = 88 ↵
 P1          88 = ↵                ←変更なし
 P2          99 = 00 ↵
 P3          00 = 11 ↵
        .
        .
        .
 EXTSFR13    01 = 03 ↵
 EXTSFR14    55 = 30 ↵
 EXTSFR15    00 = 02 ↵
*
```

```
*SPR C BRG ↵                    ←"BRG"ﾚｼﾞｽﾀ から変更
 BRG        03 = 04 ↵
 TXB        -- = 0AA ↵          ←書込み専用ﾚｼﾞｽﾀの場合は'--'が表示される
 TM0      0200 = 0300 ↵
 MD0      1000 = 2000 ↵
 TM1      2100 = 2200 ↵
 MD1      0000 = 0100 ↵
 EXTSFR0    20 = . ↵              ←ﾋﾟﾘｵﾄﾞ入力で変更を終了します。
*
```

・特殊ﾚｼﾞｽﾀの表示

```
*SPR D ↵                        ←すべての特殊ﾚｼﾞｽﾀの表示
P0      P1      P2      P3      P4      P5      CR00    CR01    CR10    CR11
88      88      00      11      22      33      2000    3000    4000    5000

CPT0    CPT1    PWM0    PWM1    UDC0    UDC1    P0L     P0H     BRG     RXB
1000    1100    2200    3500    2500    4560    80      90      04      68

ADCR    TM0     MD0     TM1     MD1
50      0300    2000    2200    0100

EXTSFR0 EXTSFR1 EXTSFR2 EXTSFR3 EXTSFR4 EXTSFR5 EXTSFR6 EXTSFR7
20      01      02      03      04      05      06      07

EXTSFR8 EXTSFR9 EXTSFR10 EXTSFR11 EXTSFR12 EXTSFR13 EXTSFR14 EXTSFR15
08      09      0A      0B      0C      03      30      02
```

```
*SPR D PWM1↵                    ←"PWM1"の内容を表示
 PWM1     3500
*

*SPR D P0↵                      ←"P0"の内容を表示
 P0         88
*
```

４．５．３０　入力ディバイス指定（ＳＴＲ）

```
ＳＴＲ_file name_parameter list
```

file name　　：入力ﾌｧｲﾙ名
parameter list　：実ﾊﾟﾗﾒｰﾀ･ﾘｽﾄ

‘ＳＴＲ’コマンドは，システム・モードで使用している場合のみ有効なコマンドです。もし，スタンドアロン・モードで使用した場合は，エラーとなります。

入力ディバイス指定コマンドは，このコマンド以降のコマンド，および，データをホスト・マシンの指定されたファイルから入力します。また，実パラメータを指定することにより，ファイル中の仮パラメータを実パラメータに置換えられます。パラメータは，最大四つまで指定することができます。

このコマンドで使用できるファイルの形式は，コマンド・ファイル作成コマンドで作成されたファイル，あるいは，ＣＰ／Ｍなどのエディタによってコマンド，および，データの入力形式で作成されたものです。なお，仮パラメータは，＄０，＄１，＄２，＄３の四つが有効になります。仮パラメータの‘$’とアセンブラのアドレスを指定する‘$’を区別するため，アドレスを指定する場合は‘＄＄’と記述してください。

注　仮パラメータを使用するファイルは，ＣＰ／Ｍなどのエディタで作成してください。

Ｃｔｒｌ－Ｋキーはｓｔｒコマンドを中断するときに使用します。ＳＴＲコマンドで，ファイルからコマンド・データを入力しているときにＣｔｒｌ－Ｋキーを入力すると，ファイルからの入力は中断されます。これ以降，コンソールからコマンド・データを入力することができます。

Ｃｔｒｌ－Ｌキーは，ＳＴＲコマンドを一時停止するときに使用します。ＳＴＲコマンドで，ファイルからコマンド・データを入力しているときに，Ｃｔｒｌ－Ｌキーを入力すると，ファイルからの入力は一時停止されます。これ以降は，コンソールから，コマンド・データを入力することができます。この状態で，もう一度，Ｃｔｒｌ－Ｌキーを入力すると，ファイルからのコマンド・データに入力が再開されます。

例

```
1>STR B:SAMPLE.STR↵        ←ドライブ Bの SAMPLE.STR を入力ファイルに指定します。
1>MAP W 0XXX               ←このコマンドよりファイルから入力
1>MAP R 1XXX
1>MAP K 0FDXX
1>MAP U 8XXX
1>MAP
  0000-0FFF R/W   1000-1FFF R/0   2000-7FFF Non   8000-8FFF User
  9000-FDFF Non
1>LOD SAMPLE
  object load complete
  symbol table loading
  PUBLIC      load complete
  MOD00       load complete
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
  MOD05       load complete
1>MEM D 0X
  0000   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     .........
1>MEM F 0XX 00
1>MEM D 0XX                  ←このコマンドまでファイルから入力
  0000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     .........
  0010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     .........
  0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     .........
                                    .
                                    .
                                    .
  00F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     .........
1>
```

ファイルからコマンド，および，データ の入力は，ファイルの終了を検出した場合に終了します。ファイルが終了した場合は，その後の入力はコンソール からの入力になります。上記例のファイルの内容を下記に示します。

```
MAP W 0XXX
MAP R 1XXX
MAP K 0FDXX
MAP U 8XXX
MAP
LOD SAMPLE
MEM D 0X
MEM F 0XX 00
MEM D 0XX
```

```
1>STR SAMPLE.STR 0XX 0XX↵  ←パラメータ を指定した場合
1>MAP W 0XXX               ←このコマンドよりファイルから入力
1>MAP R 1XXX
1>MAP K 0FDXX
1>MAP U 8XXX
1>MAP
  0000-0FFF R/W   1000-1FFF R/0   2000-7FFF Non   8000-8FFF User
  9000-FDFF Non
1>LOD SAMPLE
  object load complete
  symbol table loading
  PUBLIC      load complete
  MOD00       load complete
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
```

```
  MOD05       load complete
1>MEM D 0X
  0000   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F      ........
1>MEM F 0XX 00  ┐ 仮ﾊﾟﾗﾒｰﾀ が展開されています。
1>MEM D 0XX     ┘
   └──  このｺﾏﾝﾄﾞまでﾌｧｲﾙから入力
  0000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ........
  0010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ........
  0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ........
                                    .
                                    .
                                    .
  00F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ........
1>
```

ﾊﾟﾗﾒｰﾀが指定された場合は，ﾌｧｲﾙ中に仮ﾊﾟﾗﾒｰﾀが指定されていなければなりません。仮ﾊﾟﾗﾒｰﾀが指定されていない場合は，実ﾊﾟﾗﾒｰﾀは無視されます。上記例のﾌｧｲﾙの内容を下記に示します。

```
  MAP W 0XXX
  MAP R 1XXX
  MAP K 0FDXX
  MAP U 8XXX
  MAP
  LOD SAMPLE
  MEM D 0X
  MEM F $0 00              ←仮ﾊﾟﾗﾒｰﾀ 1
  MEM D $1                 ←仮ﾊﾟﾗﾒｰﾀ 2

1>STR B:SAMPLE.STR↵
  file not found           ←ﾌｧｲﾙが見つからない場合のﾒｯｾｰｼﾞ
1>
```

指定したﾌｧｲﾙが見つからない場合は，このｺﾏﾝﾄﾞは，無視されます。ｺﾏﾝﾄﾞ，あるいは，ﾃﾞｰﾀの入力は，ｺﾝｿｰﾙからになります。

４．５．３１　サフィックス指定（ＳＵＦ）

```
SUF [_ {H} ]
       {T}
       {Q}
       {Y}
```

H　　：16進数
T　　：10進数
Q　　：8進数
Y　　：2進数

サフィックス指定コマンドは，入力する数値のサフィックスを設定することができます。設定できるサフィックスは，１６進数（H），１０進数（T），８進数（Q），および，２進数（Y）です。

オペランドのサフィックス指定が省略された場合は，現在設定されているサフィックスが表示されます。

例

```
*SUF H ↵          ←ｻﾌｨｯｸｽを16進数に設定
*
*SUF ↵            ←現在設定されているｻﾌｨｯｸｽの表示
 H                ←現在指定されているｻﾌｨｯｸｽは16進数(H)
*
```

４．５．３２　シンボル操作（ＳＹＭ）

```
SYM_ {A} _symbol_word
     {C}

SYM[_D[_ {PUBLIC        } ]]
         {module name \ }

SYM_E[_symbol]

SYM_ {K}
     {L}
     {M}
     {S}
```

A	：追加ｼﾝﾎﾞﾙの定義
C	：追加ｼﾝﾎﾞﾙのｼﾝﾎﾞﾙ値の変更
D	：ｼﾝﾎﾞﾙの表示
E	：追加ｼﾝﾎﾞﾙの削除
K	：ｼﾝﾎﾞﾙの削除
L	：追加ｼﾝﾎﾞﾙのﾛｰﾄﾞ
M	：ｶﾚﾝﾄ・ﾓｼﾞｭｰﾙの変更
S	：追加ｼﾝﾎﾞﾙのｾｰﾌﾞ
symbol	：ｼﾝﾎﾞﾙ名
word	：ｼﾝﾎﾞﾙ値
PUBLIC	：ﾊﾟﾌﾞﾘｯｸ・ｼﾝﾎﾞﾙすべて
module name \	：ﾓｼﾞｭｰﾙ 名

シンボル操作コマンドは，スタンドアロン・モードとシステム・モード両方で有効なコマンドと，システム・モードでのみ有効なコマンドがあります。また，追加されたシンボルでのみ有効なコマンドと，追加されたシンボルとシンボル・テーブル・ファイルで定義されたシンボルの両方で有効なコマンドがあります。

（1）追加シンボルの定義

ＳＹＭ＿Ａ＿ｓｙｍｂｏｌ＿ｗｏｒｄ

追加シンボルの定義コマンドは，オペランドで指定されたシンボルを定義することができます。このコマンドは，システム・モードとスタンドアロン・モードの両方で有効なコマンドです。

オペランドで指定したシンボルがすでに定義されているシンボルや，あるいは，予約語の場合は，このコマンドで定義することはできません。

また，このコマンドで定義されたシンボルに対しては，“ＩＥＳＹＭＢＯＬ”というモジュール名が付けられます。なお，シンボルのタイプは，“ｃｏｄｅ”タイプとなります。

例

```
*SYM A SYMBOL01 1000 ↵
*
```

シンボル値が1000で“SYMBOL01”というシンボル名を定義します。

（2）追加シンボルのシンボル値の変更

ＳＹＭ＿Ｃ＿ｓｙｍｂｏｌ＿ｗｏｒｄ

追加シンボルのシンボル値の変更コマンドは，オペランドで指定されたシンボルのシンボル値を変更することができます。このコマンドは，システム・モードとスタンドアロン・モードの両方で有効です。

このコマンドで変更できるシンボルは，追加シンボルの定義コマンドで定義されたシンボルでなければなりません。それ以外のシンボルのシンボル値の変更はできません。

例

```
*SYM C SYMBOL01 2000 ↵
*
```

“SYMBOL01”というシンボルのシンボル値を2000に変更します。“SYMBOL01”は，追加シンボルの定義コマンドで定義されたシンボルでなければなりません。

（３）シンボルの表示

```
SYM [_D]                                    ←スタンドアロン・モード のコマンド形式

SYM [_D[_ {PUBLIC      } ]]                 ←システム・モードのコマンド形式
          {module name }
```

シンボルの表示コマンドは，定義されているコマンドを表示することができます。このコマンドは，システム・モードとスタンドアロン・モードの両方で有効です。

スタンドアロン・モードでのシンボルの表示は，追加シンボルの定義コマンドで定義されたシンボルだけが表示されます。

システム・モードでのシンボルの表示は，追加シンボルの定義コマンドで定義されたシンボルとシンボル・テーブル・ファイルで定義されたシンボルが表示されます。また，オペランドにモジュール名を指定することにより，そのモジュールのローカル・シンボルだけ，あるいは，パブリック・シンボルだけを表示することができます。オペランドが省略された場合は，定義されているすべてのシンボルが表示されます。

例1 スタンドアロン・モードの場合

・追加されたシンボルがある場合

```
*SYM D ↵
 module : IESYMBOL ←追加シンボルのモジュール 名
1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
1100 SYMBOL05      2100 SYMBOL08
*
```

・追加されたシンボルがない場合

```
*SYM D ↵
 no symbol of append      ←追加シンボルがない場合のメッセージ
*
```

例2 システム・モードの場合

・追加されたシンボルがある場合

```
1>SYM D ↵
  module : IESYMBOL
  1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
  1100 SYMBOL05      2100 SYMBOL08
  module : PUBLIC
  0940 SAFF28_0 bit0 0942 SAFF28_2 bit2 0943 SAFF28_3 bit3 0947 SAFF28_7 bit7
  0948 SAFF29_0 bit0 094F SAFF29_7 bit7 0000 XSYM0001      0100 XSYM0002
  0200 XSYM0003      0300 XSYM0004      0400 XSYM0005      0500 XSYM0006
  0600 XSYM0007      0700 XSYM0008
  module : MOD00
  0008 AREG_0   bit0 000B AREG_3   bit3 000D AREG_5   bit5 000F AREG_7   bit7
```

```
  00FF IBIT7    bit7 0A00 LSYM0000      0B00 LSYM0001      0C00 LSYM0002
  0D00 LSYM0003      0E00 LSYM0004      0F00 LSYM0005      0FF0 PSWL_0   bit0
  0FF2 PSWL_2   bit2 0FF8 PSWH_0   bit0 0FFE PSWH_6   bit6 0110 RETRY
  0FF0 RETURN        0005 XREG_5   bit5
  module : MOD01
           .
           .
           .
           .
1>
```

・追加されたｼﾝﾎﾞﾙがない場合

```
1>SYM D )
  module : PUBLIC
  0940 SAFF28_0 bit0 0942 SAFF28_2 bit2 0943 SAFF28_3 bit3 0947 SAFF28_7 bit7
  0948 SAFF29_0 bit0 094F SAFF29_7 bit7 0000 XSYM0001      0100 XSYM0002
  0200 XSYM0003      0300 XSYM0004      0400 XSYM0005      0500 XSYM0006
  0600 XSYM0007      0700 XSYM0008
  module : MOD00
  0008 AREG_0   bit0 000B AREG_3   bit3 000D AREG_5   bit5 000F AREG_7   bit7
  00FF IBIT7    bit7 0A00 LSYM0000      0B00 LSYM0001      0C00 LSYM0002
  0D00 LSYM0003      0E00 LSYM0004      0F00 LSYM0005      0FF0 PSWL_0   bit0
  0FF2 PSWL_2   bit2 0FF8 PSWH_0   bit0 0FFE PSWH_6   bit6 0110 RETRY
  0FF0 RETURN        0005 XREG_5   bit5
  module : MOD01
           .
           .
           .
           .
1>
```

・ﾊﾟﾌﾞﾘｯｸ･ｼﾝﾎﾞﾙだけを表示する場合

```
1>SYM D PUBLIC )
  module : IESYMBOL
  1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
  1100 SYMBOL05      2100 SYMBOL08
  module : PUBLIC
  0940 SAFF28_0 bit0 0942 SAFF28_2 bit2 0943 SAFF28_3 bit3 0947 SAFF28_7 bit7
  0948 SAFF29_0 bit0 094F SAFF29_7 bit7 0000 XSYM0001      0100 XSYM0002
  0200 XSYM0003      0300 XSYM0004      0400 XSYM0005      0500 XSYM0006
  0600 XSYM0007      0700 XSYM0008
1>
```

・ﾓｼﾞｭｰﾙ 名を指定した場合

```
1>SYM D MOD00 \ )
  module : IESYMBOL
  1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
  1100 SYMBOL05      2100 SYMBOL08
  module : MOD00
  0008 AREG_0   bit0 000B AREG_3   bit3 000D AREG_5   bit5 000F AREG_7   bit7
  00FF IBIT7    bit7 0A00 LSYM0000      0B00 LSYM0001      0C00 LSYM0002
  0D00 LSYM0003      0E00 LSYM0004      0F00 LSYM0005      0FF0 PSWL_0   bit0
  0FF2 PSWL_2   bit2 0FF8 PSWH_0   bit0 0FFE PSWH_6   bit6 0110 RETRY
  0FF0 RETURN        0005 XREG_5   bit5
1>
```

（４）追加シンボルの削除

ＳＹＭ＿Ｅ［＿ｓｙｍｂｏｌ］

追加シンボルの削除コマンドは，オペランドで指定されたシンボルを削除します。このコマンドは，システム・モードと，スタンドアロン・モードの両方で有効です。

オペランドで指定されたシンボルは，追加シンボルの定義コマンドで定義されたシンボルでなければなりません。それ以外のシンボルは，削除することはできません。また，オペランドにシンボルが指定されなければ，追加シンボルの定義コマンドで定義されたシンボルをすべて削除します。

例

```
*SYM E SYMBOL01          ←"SYMBOL01"を削除する
*
*SYM E                   ←すべての追加ｼﾝﾎﾞﾙの定義ｺﾏﾝﾄﾞで定義されたｼﾝ
*                          ﾎﾞﾙを削除します。
```

（５）シンボルの削除

ＳＹＭ＿Ｋ

シンボルの削除コマンドは，定義されているすべてのシンボルを削除します。このコマンドは，システム・モードとスタンドアロン・モードの両方で有効です。

スタンドアロン・モードでは，追加シンボルの定義コマンドで定義されたシンボルすべてを削除します。

システム・モードでは，追加シンボルの定義コマンドで定義されたシンボル，および，シンボル・テーブル・ファイルで定義されたシンボルをすべて削除します。

例

```
*SYM K                   ←定義されているｼﾝﾎﾞﾙすべてを削除
*
```

（６）追加シンボルのロード

ＳＹＭ＿Ｌ

追加シンボルのロード・コマンドは，追加シンボル・ファイルから追加シンボルをロードします。このコマンドは，システム・モードの場合だけ有効です。

追加シンボル・ファイルのファイル名は，“ＩＥ７８３１０．ＳＹＭ”というファイル名が自動的に設定されます。また，ドライブ・ユニットは，カレント・ドライブが指定されます。このため，カレント・ドライブに“ＩＥ７８３１０．ＳＹＭ”というファイル名をもつファイルを作成しないでください。

例

```
1>SYM L ↵                    ←正常にﾛｰﾄﾞ が終了
1>

1>SYM L ↵
  append symbol file not found  ←ｶﾚﾝﾄ･ﾄﾞﾗｲﾌﾞ に IE78310.SYMというﾌｧｲﾙが
1>                                ない場合のﾒｯｾｰｼﾞ
```

（７）カレント・モジュールの変更

ＳＹＭ＿Ｍ

カレント・モジュールの変更コマンドは，現在定義されているカレント・モジュール名の表示，および変更をすることができます。

このコマンドは，システム・モードとスタンドアロン・モードの両方で有効です。コマンド行やデータ行で，数値の代わりにローカル・シンボル名を使用する場合，シンボル名の前に，モジュール名を付けなければなりません。

しかし，モジュール名とシンボル名を続けて入力すると，長くなり，キー入力が煩わしい場合があります。

ＳＹＭ＿Ｍコマンドを使用しますと，このようなときあらかじめモジュール名を登録しておき，この後のコマンド行やデータ行では，ローカル・シンボル名だけをキー入力することができます。

ただし，パブリック・シンボル・モジュールに，該当するシンボル名が存在するならば，そのシンボルはパブリック・シンボルと判断されます（図４－５参照）。また，他のモジュールに属するローカル・シンボルを入力する場合は，ローカル・シンボル名の前に，モジュール名を指定してください。

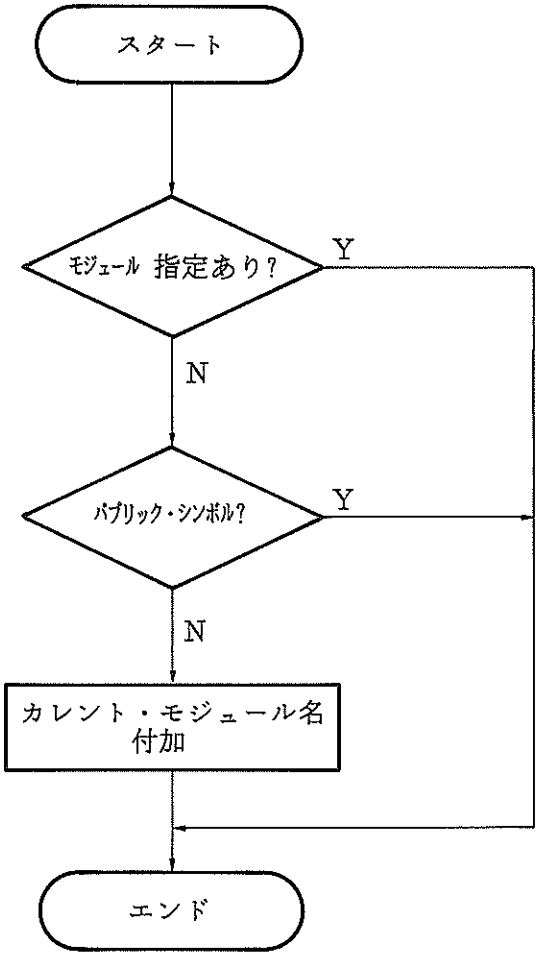
例1 現在指定されている，カレント・モジュール名の表示を行ないます。

```
*SYM M ↵
 module01＼=                ←現在のｶﾚﾝﾄ･ﾓｼﾞｭｰﾙ名を表示します。

*SYM M
 module01＼= ↵             ←ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙ名を変更しないときは，ﾘﾀｰﾝ・ｷｰ
                              を入力します
*
```

例2 カレント・モジュールを現在指定されているmodule01＼からmodule02＼へ変更します。

```
*SYM M
 module01＼= module02＼ ↵      ←ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙをmodule02＼にします。
*

*SYM M
 module02＼=                ←ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙがmodule02＼に変更され
                              ました。
```

図4－5　シンボルのモジュール判断の処理フロー



（８）追加シンボルのセーブ

ＳＹＭ＿Ｓ

追加シンボルのセーブ・コマンドは，追加シンボル・ファイルへ追加シンボルをセーブします。このコマンドは，システム・モードの場合だけ有効です。

追加シンボル・ファイルのファイル名は，“ＩＥ７８３１０．ＳＹＭ”というファイル名が自動的に設定されます。また，ドライブ・ユニットは，カレント・ドライブが指定されます。このため，カレント・ドライブに“ＩＥ７８３１０．ＳＹＭ”というファイル名をもつファイルを作成しないでください。

例

```
1>SYM S ↵                    ←ｾｰﾌﾞ が正常に終了
1>

1>SYM S ↵
  no symbol of append        ←追加ｼﾝﾎﾞﾙの定義ｺﾏﾝﾄﾞで定義されたｼﾝﾎﾞﾙがない
1>                             場合のﾒｯｾｰｼﾞ

1>SYM S ↵
  File already exists. Delete ?(Y or N): Y↵
1>
```

IE78310.SYM というﾌｧｲﾙがすでに存在している場合に，上記のようなﾒｯｾｰｼﾞが出力されます。このとき，Y を入力しますと，まず IE78310.SYMというすでに存在しているﾌｧｲﾙをﾃﾞﾘｰﾄします。そして，新たに IE78310.SYMというﾌｧｲﾙをｵｰﾌﾟﾝします。また，Y 以外が入力された場合は，ｺﾏﾝﾄﾞは無視されます。ただし，すでに存在するﾌｧｲﾙの属性が DIR属性でかつR/W 属性の場合だけです。

もし，ﾌｧｲﾙの属性が SYS属性，あるいは， R/O属性の場合は，下記のようなﾒｯｾｰｼﾞ を出力してｺﾏﾝﾄﾞは，無視されます。

```
1>SYM S ↵
  File already exists.
1>
```

４．５．３３　トレース・モード設定（ＴＲＭ）

```
TRM[_{NON}]
     {ALL}
     {TRX}
```

NON　：ノン・トレース
ALL　：全トレース
TRX　：クオリファイ・トレース

トレース・モード設定コマンドは，トレース条件を指定することができます。条件の設定が省略された場合は，現在指定されているトレース条件が表示されます。

例

```
*TRM TRX ↵          ←トレース条件にクオリファイ・トレース を指定
*

*TRM ↵              ←現在指定されているトレース条件を表示
 TRX
*
```

４．５．３４　クオリファイ条件設定（ＴＲＸ）

```
ＴＲＸ[_A=addrs][_V=mask][_C=ステータス][_TRQ=mask]
```

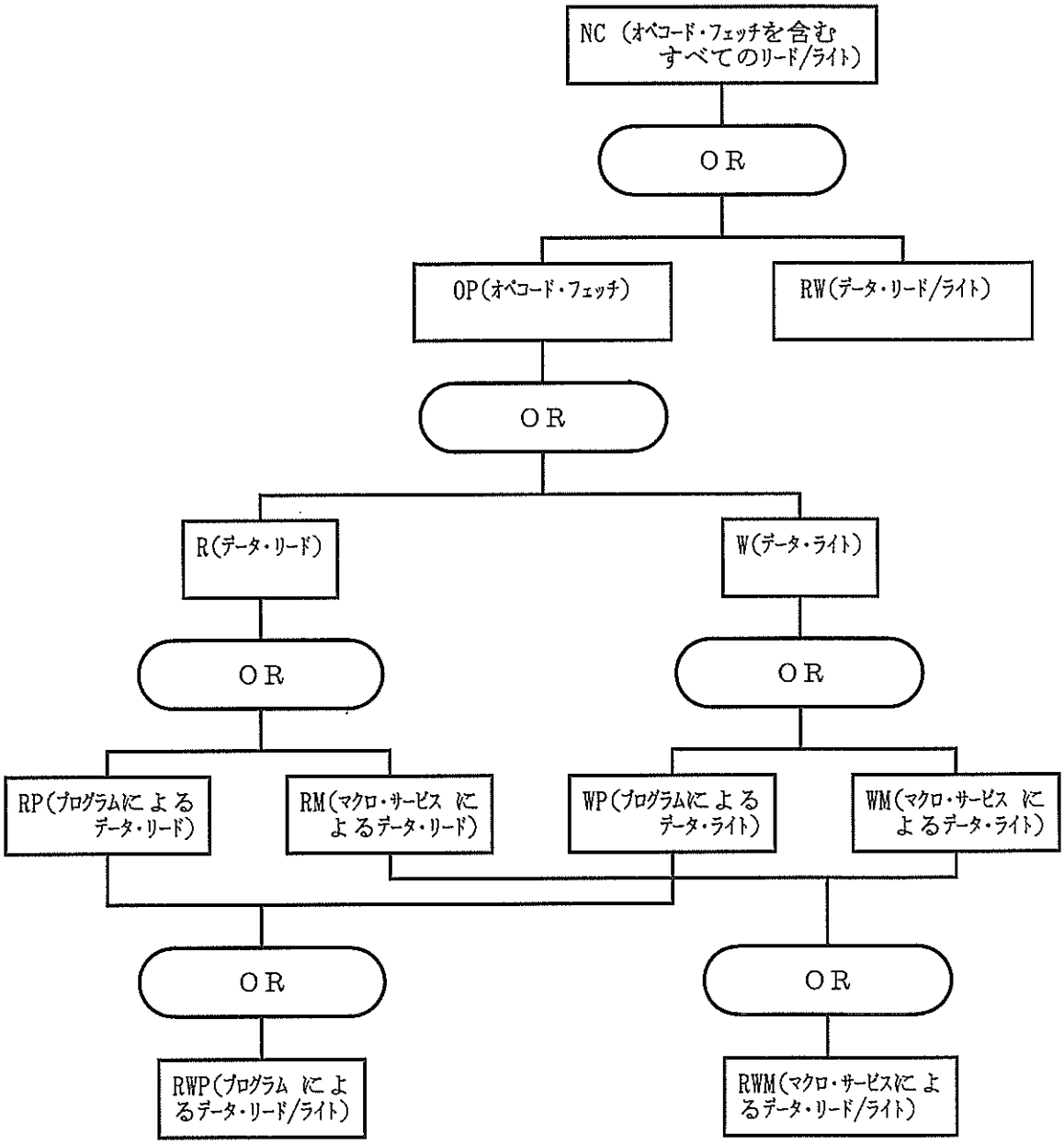
```
A=addrs    ： クオリファイ・アドレス でトレース範囲を最大５箇所まで設定できる
V=mask     ： クオリファイ・データで8ビットのマスク・データ が設定できる
C=ステータス ： クオリファイ・ステータスで下記の条件から選択できる
        OP  ：オペコード・フェッチ
        RW  ：データ・リード/ライト
        R   ：データ・リード
        W   ：データ・ライト
        RWP ：プログラムによるデータ・リード/ライト
        RP  ：プログラムによるデータ・リード
        WP  ：プログラムによるデータ・ライト
        RWM ：マクロ・サービス によるデータ・リード/ライト
        RM  ：マクロ・サービス によるデータ・リード
        WM  ：マクロ・サービス によるデータ・ライト
        NC  ：オペコード・フェッチ を含むすべてのリード/ライト
TRQ=mask   ： クオリファイ・ポート/ 外部データ で8ビットのマスク・データ が設定できる
```

クオリファイ条件設定コマンドは，ハードウエアに対するトレース条件を設定します。トレース・モード設定コマンドでクオリファイ・トレース（ＴＲＸ）が指定されていなければ無効です。

クオリファイ条件（クオリファイ・アドレス（Ａ），クオリファイ・データ（Ｖ），クオリファイ・ステータス（Ｃ））は，すべてＡＮＤ条件になります。ただし，クオリファイ・ポート／外部データとは，ＯＲ条件になります。

クオリファイ条件が省略された場合は，トレース条件の設定を対話形式ですることができます。この場合は，現在設定されているトレース条件が表示されます。もし，条件が設定されていない場合は，‘－－’が表示されます。

図４－６　ＴＲＸコマンド・データ・ステータス構成



例　　・１行で設定する場合

```
*TRX A=1H 2XXH 86H,10AH V=50H C=RW TRQ=0XX110Y )
*
```

ｸｵﾘﾌｧｲ･ｱﾄﾞﾚｽ: 1番地,200番地～ 2FF番地, 86番地～ 10A番地の 3箇所を設定
ｸｵﾘﾌｧｲ･ﾃﾞｰﾀ : 50Hを設定
ｸｵﾘﾌｧｲ･ｽﾃｰﾀｽ:ﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ(RW)を設定
ｸｵﾘﾌｧｲ･ﾎﾟｰﾄ/ 外部ﾃﾞｰﾀ:0XX110Y を設定

・対話形式で設定する場合

```
*TRX )
 A 1H 2XXH 86H,10AH = 0XXXH )   ←ｸｵﾘﾌｧｲ･ｱﾄﾞﾚｽ に新しく0XXXH を設定
 V 50H = )                      ←現在のﾃﾞｰﾀ を変更しない
   OPcode fetch                 (OP)
   Read Write                   (RW)
   Read                         (R)
   Write                        (W)
   Read Write by program        (RWP)
   Read by program              (RP)
   Write by program             (WP)
   Read Write by Macro service  (RWM)
   Read by Macro service        (RM)
   Write by Macro service       (WM)
   No Condition                 (NC)
 C RW = NC )                    ←現在のｽﾃｰﾀｽ を新しく 'NC' に設定
 TRQ 0XX110Y = )                ←現在のﾃﾞｰﾀ を変更しない
*
```

４．５．３５　クオリファイ・データ選択（ＴＲＱ）

```
ＴＲＱ[_ { TRS } ]
        { EXT }
```

TRS　：ﾎﾟｰﾄ を選択
EXT　：外部信号を選択

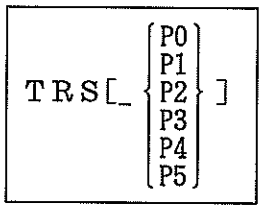
クオリファイ・データ選択コマンドは，クオリファイ条件のクオリファイ・ポート／外部データ（ＴＲＱ）のデータが外部信号，あるいは，ポートかを選択します。

また，オペランドが省略された場合は，現在選択されているデータが表示されます。

例

```
*TRQ TRS )          ←ﾎﾟｰﾄ を選択
*
*TRQ EXT )          ←外部信号を選択
*
*TRQ )              ←現在選択されているﾃﾞｰﾀ を表示
 EXT                ←現在は外部信号が選択されている
*
```

４．５．３６　トレース／クオリファイ・ポート選択（ＴＲＳ）



トレース／クオリファイ・ポート選択コマンドは，トレース／クオリファイされるポートを選択します。ただし，クオリファイ・データ選択コマンドでポート（ＴＲＳ）が選択されていなければこのコマンドは無効になります。

また，オペランドが省略された場合は，現在選択されているポートが表示されます。

例

```
*TRS P0 )          ←ポート0を選択
*
*TRS )             ←現在選択されているポート を表示
 P0
*
```

４．５．３７　トレース・ポインタ操作（ＴＲＰ）

```
TRP[_{word}]
    { 0 }
    { N }
```

word　：ポインタの移動数
0　：ポインタを先頭に置く
N　：ポインタを最後に置く

トレース・ポインタ操作コマンドは，トレース・ポインタをオペランドで指定された数だけ移動します。ポインタの移動数は，±１～２０４７までの範囲で指定できます。

オペランドに“０”が指定された場合は，トレース・ポインタを最も古いトレース・ラインに移動します。また，オペランドに“Ｎ”が指定された場合は，最も新しいトレース・ラインに移動します。

オペランドが省略された場合は，現在トレースされているデータの総数が１０進数で表示されます。

トレース実行のトレース・ポインタは，最新のトレース・ライン－９を示しています。トレース総ライン数が１０Ｔライン以下の時は，最も古いトレース・ラインを示します。

例

```
*TRP 20 ↵          ←ﾄﾚｰｽ･ﾎﾟｲﾝﾀ を20Hﾗｲﾝ進めます。
*
*TRP N ↵           ←ﾄﾚｰｽ･ﾎﾟｲﾝﾀ を最も新しいﾄﾚｰｽ･ﾗｲﾝに移動します。
*
*TRP ↵
 Total frame number     = 1200  ←ﾄﾚｰｽされたﾌﾚｰﾑの総数を表示します。
 Display frame pointer = 1012  ←表示したﾌﾚｰﾑの位置を表示します。
*
```

４．５．３８　トレース表示（ＴＲＤ）

```
TRD[_{F|I|M}[_{word|ALL}]]
```

F	：	ﾌﾚｰﾑ･ﾓｰﾄﾞ
I	：	ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ
M	：	ﾏｸﾛ･ｻｰﾋﾞｽ付きｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ
word	：	表示ｽﾃｯﾌﾟ数
ALL	：	ﾄﾚｰｽ結果すべてを表示

トレース表示コマンドは，トレースされたデータをサブコマンドによって指定された形式で表示します。表示ステップ数が指定されている場合は，指定されたステップ数だけ表示します。また，“ＡＬＬ”が指定されている場合は，トレースされているデータがすべて表示されます。なお，オペランドが省略された場合は，１１行分のデータが表示されます。

サブコマンドとオペランドが省略された場合は，インストラクション・モードで１１行分が表示されます。

例1 フレーム・モードでの表示

```
*TRD F ALL )                    ←フレーム・モードでトレース・データすべてを表示
 Frame Status Address Data     7--P2--0  7--EX--0
 0000    M1     0100   0B     00000000  00000000
 0001    OP     0101   FC     00000000  00000000
 0002    OP     0102   7F     00000000  00000000
 0003    OP     0103   FE     00000000  00000000
 0004   MSWR    2000   7D     00000000  00000000
 0005    M1     0104   67     00000000  00000000
 0006    OP     0105   00     00000000  00000000
 0007    OP     0106   10     00000000  00000000
 0008    M1     0107   B8     00000000  00000000
 0009    OP     0108   02     00000000  00000000
 0010    M1     0109   D0     00000000  00000000
 0011   MSRD    2100   FE     00000000  00000000
 0012    M1     010A   51     00000000  00000000
 0013    M1     010B   C8     00000000  00000000
 0014    M1     010C   80     00000000  00000000
 0015    WR     1000   02     00000000  00000000
 0016    OP     010D   FB     00000000  00000000
 0017   (M1)    010E   67     00000000  00000000
 0018   (OP)    010D   00     00000000  00000000
 0019   BRM1    0109   D0     00000000  00000000
 0020   MSWR    2001   7E     00000000  00000000
 0021    M1     010A   51     00000000  00000000
 0022    M1     010B   C8     00000000  00000000
 0023    WR     1001   01     00000000  00000000
 0024    M1     010C   80     00000000  00000000
 0025    OP     010D   FB     00000000  00000000
 0026    M1     010E   67     00000000  00000000
 0027    OP     010F   00     00000000  00000000
 0028   MSRD    2101   FF     00000000  00000000
 0029    OP     0110   20     00000000  00000000
 0030   (M1)    0111   00     00000000  00000000
 0031   (M1)    0112   00     00000000  00000000
 0032   (M1)    0113   00     00000000  00000000
*
```

注　意

①Status欄の“( )”で示された表示は，プリフェッチはしたが実行しなかった場合です。

②トレース・データの前に“?”が付いている場合は，データ が不確定です。

以下の場合になります。

・saddr に対する奇数番地のリード・アクセス

③トレース・データが“XX”のところは，トレースができません。

以下の場合になります。

・sfr に対するリード・アクセス

・sfrpに対する16ビット・リード・アクセス

④ベクタ・テーブル領域（0H～29H 番地）にジャンプするようなプログラム では，正しくインストラクション・トレース表示は行えません。

例2 マクロ・サービス付きインストラクション・モードでの表示

```
*TRD M ALL ↵          ←マクロ・サービス付きインストラクション・モード でトレース・データすべてを表示
 Frame Status Address Data Label  Mnemonic                    P2 EX
                             PUBLIC\START::                   00 00
 0000         0100           MOVW    SP,#STC
 0004   MSWR  2000    7D                                      00 00
 0005         0104           MOVW    RP7,#WORK1
 0008         0107           MOV     R0,#2H
                             MODULE00\LOOP:                   00 00
 0010         0109           MOV     A,R0
 0011   MSRD  2100    FE                                      00 00
 0012         010A           MOV     [HL+],A
 0015    WR   1000    02                                      00 00
 0013         010B           DEC     R0
 0014         010C           BNZ     $LOOP
                             MODULE00\LOOP:                   00 00
 0019         0109           MOV     A,R0
 0020   MSWR  2001    7E                                      00 00
 0021         010A           MOV     [HL+],A
 0023    WR   1001    01                                      00 00
 0022         010B           DEC     R0
 0024         010C           BNZ     $LOOP
 0026         010E           MOVW    RP7,#WORK2
 0028   MSRD  2101    FF                                      00 00
*
```

例3 インストラクション・モードでの表示

```
*TRD I ALL ↵          ←インストラクション・モード でトレース・データすべてを表示
 Frame Status Address Data Label  Mnemonic
                             PUBLIC\START::
 0000         0100           MOVW    SP,#STC
 0005         0104           MOVW    RP7,#WORK1
 0008         0107           MOV     R0,#2H
                             MODULE00\LOOP:
 0010         0109           MOV     A,R0
 0012         010A           MOV     [HL+],A
 0015    WR   1000    02
 0013         010B           DEC     R0
 0014         010C           BNZ     $LOOP
                             MODULE00\LOOP:
 0019         0109           MOV     A,R0
 0021         010A           MOV     [HL+],A
 0023    WR   1001    01
 0022         010B           DEC     R0
 0024         010C           BNZ     $LOOP
 0026         010E           MOVW    RP7,#WORK2
*
```

４．５．３９　オブジェクト・ベリファイ（ＶＲＹ）

```
VRY_{ TTY1 }  ........  スタンドアロン・モード
    { TTY2 }
```

```
VRY_file name  ........  システム・モード
```

TTY1　　：ｼﾘｱﾙ･ﾁｬﾈﾙ1
TTY2　　：ｼﾘｱﾙ･ﾁｬﾈﾙ2
file name　：ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ 名

オブジェクト・ベリファイ・コマンドは，オブジェクト・ファイルとメモリ内容を比較します。比較した結果がファイルとメモリで異なる場合は，そのアドレスとデータが表示されます。

スタンドアロン・モードでのオブジェクト・ベリファイは，シリアル・チャネル１，または，シリアル・チャネル２から入力されるオブジェクトとメモリ内容を比較します。オペランドのシリアル・チャネルが省略された場合は，シリアル・チャネル１が指定されます。

システム・モードでは，指定されたファイル名をもつオブジェクト・ファイルとメモリ内容を比較します。

例1 スタンドアロン・モードの場合

```
*VRY TTY2↵                ←ｼﾘｱﾙ･ﾁｬﾈﾙ 2 からのｵﾌﾞｼﾞｪｸﾄと比較する
 complete                 ←比較した結果に異常がない場合のﾒｯｾｰｼﾞ
*


*VRY TTY2↵
 Address  File  Memory    ←比較した結果に異常があった場合のﾒｯｾｰｼﾞ
  0123     00     01
  1234     FF     FE
*
```

メモリの内容
ファイル の内容
メモリのアドレス

例2 システム・モードの場合

```
1>VRY SAMPLE.HEX↵          ←SAMPLE.HEXというﾌｧｲﾙのｵﾌﾞｼﾞｪｸﾄと比較する
  object verify complete    ←比較した結果に異常がない場合のﾒｯｾｰｼﾞ
1>


1>VRY SAMPLE.HEX↵
  object verify
  Address  File  Memory     ←比較した結果に異常があった場合のﾒｯｾｰｼﾞ
   0123     00     01
   1234     FF     FE
1>
```

メモリの内容
ファイル の内容
メモリのアドレス

```
1>VRY SAMPLE.HEX↵
  file not found            ←SAMPLE.HEXというﾌｧｲﾙ名が見つからない場合
1>                            のﾒｯｾｰｼﾞ
```

4．6　エラー・メッセージ

エラー・メッセージの一覧を以下に示します。

(1) Unrecognized command

コマンド・キーワードが正しくない。

(2) Command format error

コマンド・キーワードは正しいが，オペランドが正しくない。

(3) Command/Data too long

１２８文字以上のコマンド，あるいは，データ行が入力された。

(4) Mapping error

指定されたアドレス範囲に，マッピングされていないメモリ・エリアがある。

(5) Input data error

入力したデータが正しくない。

(6) System mode command

スタンドアロン・モードでシステム・モードのコマンドを入力した。

(7) Non map area access

コマンド実行中にマッピングされていないメモリにアクセスしようとした。

(8) Check sum error

オブジェクトのロード／セーブ時にチェック・サム・エラーを検出した。

(9) Bad character

オブジェクトのロード／セーブ時に正しくない文字を検出した。

(10) aborted

オブジェクトのロード／セーブ中に中断キーを入力された。

(11) Warning double define

ＬＯＤコマンドで，同一モジュール名が複数回指定された。

(12) Bad file entry

ファイル名の記述が正しくない。

(13) File overflow

ＬＯＤコマンドで，入力可能なシンボル・ファイル数をオーバした。

(14) Illegal record

ＬＯＤコマンドで，シンボル・テーブル・ファイルのレコード形式が正しくない。

(15) load failed

ＬＯＤコマンドで，シンボルをロード中にエラーを検出した。

(16) module overflow

ＬＯＤコマンドで，入力できるモジュール数をオーバした。

(17) Not loaded symbol

シンボルがロードされていない。

(18) Not found module record

ＬＯＤコマンドで，指定されたモジュール名がシンボル・テーブル・ファイルに存在しない。

(19) file not found

指定されたファイル名が存在しない。

(20) Slave CPU communication error

チャネル２のスレーブＣＰＵ(8742)に対し，コマンドが書込めない。

(21) double define append symbol ｼﾝﾎﾞﾙ名

ＬＯＤコマンドで，アペンド・シンボルとして，すでに登録されているシンボルをロードした（アペンド・シンボルは削除されます)。

(22) double define append symbol

ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，すでに登録されているシンボルを登録しようとした。

(23) symbol table full

ＬＯＤコマンドで，シンボル・セーブ・エリアに空がない。

(24) append symbol table full

ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，アペンド・シンボル・セーブ・エリアに空がない。

(25) double define loaded symbol

ＬＯＤ，ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，すでにロードされているシンボルがロードされた。

(26) symbol record format error

ＬＯＤ，ＳＹＭ　Ｌコマンドで，シンボル・テーブル・ファイルのレコード形式が正しくなかった。

(27) reserved word symbol

ＳＹＭ　Ａコマンドで，予約語がシンボルとして定義された。

(28) double define module name ﾓｼﾞｭｰﾙ 名

表示されたモジュール名は，すでにロードされている。

(29) module buffer full

ＬＯＤコマンドで，入力できるモジュール数をオーバした。

(30) not found symbol

ＳＹＭ　Ｃ，ＳＹＭ　Ｅコマンドで，指定されたシンボルは存在しない。

(31) no symbol of append

ＳＹＭ　Ｄ，ＳＹＭ　Ｓコマンドで，アペンド・シンボルは存在しない。

(32) Can not execute HLP command !

カレント・ディスク上にヘルプ・ファイル(IE78310.HLP)，ヘルプ・オーバーレイ・ファイル(IE78310.OV2) が存在しない。

(33) No .HLP file on the default drive

ＨＬＰコマンド実行時，カレント・ディスク上にヘルプ・ファイル(IE78310.HLP)，ヘルプ・オーバーレイ・ファイル(IE78310.OV2) が見つからなかった。

(34) Keyword Error

ＨＬＰコマンドで，コマンド・キーワードが正しくない。

(35) Can not use command abbreviation !

カレント・ディスク上に省略形のオーバーレイ・ファイル(IE78310.OV2) が存在しない。

(36) File already exists.

ファイルの属性が SYS属性，あるいは，R/O 属性のファイルに対して同一名のファイルを新たにメイクしようとした。

(37) Reserved file name

システム・ソフトが使う予約されたファイル名を指定した。

(38) File name is used by other process

すでにオープン済みのファイル名を指定した。

(39) Can not close ﾌｧｲﾙ名

表示されたファイルのクローズが正常にできなかった。

(40) Disk write error ﾌｧｲﾙ 名

表示されたファイルの書込みで異常を見つけた。

(41) Disk read error ﾌｧｲﾙ名

表示されたファイルの読込みで異常を見つけた。

(42) Can not open ﾌｧｲﾙ 名

指定されたファイルがオープンできなかった。

(43) File make error ﾌｧｲﾙ名

表示されたファイルを作成できなかった。

(44) Can not close ﾌｧｲﾙ名.Cancel ××× command

×××のコマンド実行中，表示されたファイルのクローズが正常にできなかった（×××はＳＴＲ，ＬＳＴ，ＣＯＭの各コマンド）。

(45) Disk write error ﾌｧｲﾙ名.Cancel ××× command

×××のコマンド実行中，表示されたファイルの書き込みで異常を見つけた（×××は，ＬＳＴ，ＣＯＭの各コマンド）。

(46) Disk read error ﾌｧｲﾙ名.Cancel STR command

ＳＴＲコマンド実行中，表示されたファイルの読込みで異常を見つけた。

(47) List device is used by other process

他の処理がリスト装置を使っている（COM ｺﾏﾝﾄﾞとLST ｺﾏﾝﾄﾞの両方で ﾘｽﾄ装置を指定した場合，または，CCP/M　の場合にIE-78310以外の処理がﾘｽﾄ 装置を使用している場合）。

(48) Append symbol file not found

ＳＹＭ　Ｌコマンドで，アペンド・シンボル・ファイル(IE78310.SYM) がカレント・ディスク上に存在しなかった。

(49) Illegal append symbol file

ＳＹＭ　Ｌコマンドで，アペンド・シンボル・ファイルの形式が正しくない。

(50) Communication error

ＩＥとホスト・マシンの通信が正常にできなかった。

(51) Not found memories

外部メモリが指定されたのにメモリが使用できない。

(52) Non map area access!

ＡＳＭコマンド実行中，マッピングされていないメモリにアクセスしようとした。

(53) Assemble area over!

ＡＳＭコマンドで，アクセスできるメモリの範囲を越えた。

(54) Disassemble area over!

ＤＡＳコマンドで，アクセスできるメモリの範囲を越えた。

(55) Caution!

ジェネリックなオブジェクトが生成された，あるいは，注意を要する場合に表示される。

(56) Error!

オブジェクト・コードを生成できないか，あきらかにエラーである場合に表示される。

４．７　オンライン・アセンブラ／逆アセンブラ仕様

ＩＥ－７８３１０Ａ－Ｒのライン・アセンブラと逆アセンブラの仕様を規定しています。ライン・アセンブラ仕様は，ＡＳＭコマンドに適用されます。

逆アセンブラ仕様は，ＤＡＳなど逆アセンブル表示を行なうコマンドに適用されます。

４．７．１に　μＰＤ７８３１２Ａインストラクション一覧を，

４．７．２に　ＳＦＲマッピングを示します。

本アセンブラ／逆アセンブラ仕様は，基本的には，４．７．１，４．７．２で規定されますが，４．７．１，４．７．２で明確でない仕様や，４．７．１，４．７．２と若干異なる仕様につきましては，

４．７．３　オンライン・アセンブラ仕様

４．７．４　逆アセンブラ仕様

で規定しています。

４．７．１　μＰＤ７８３１２Ａ，７８３１０Ａインストラクション一覧表

μＰＤ７８３１２Ａ，７８３１０Ａのインストラクションは，次のように分類されます。

8-bit Transfer Instructions

16-bit Transfer Instructions

8-bit Arithmetic and Logic Instructions

16-bit Arithmetic and Logic Instructions

Multiply and Divide Instructions

Bit Manipulation Instructions

Branch Instructions

CPU Control Instructions

Decimal Adjust Instruction

String Manipulation Instructions

Shift and Rotate Instructions

注　一覧表中で，＊印がついている命令については，ＲＵＮ＿Ｓコマンドや ＢＲＥコマンドで，命令ステップをカウントする場合，２ステップとしてカウントされます。また，トレース時これらの命令をフェッチしますと，Ｍ１のフレームが２回続きます。

```
------ 8-bit Transfer Instructions -----

  MOV   r1,#byte       move Immediate to G-reg
  MOV   saddr,#byte    move Immediate to SDMEM
  MOV   sfr,#byte      move Immediate to SFR
  MOV   r2,r1          move G-reg to G-reg
  MOV   A,r2           move G-reg to ACC
  MOV   A,saddr        move SDMEM to ACC
  MOV   saddr,A        move ACC to SDMEM
  MOV   saddr,saddr    move SDMEM to SDMEM

  MOV   A,mem          move Memory to ACC
  MOV   mem,A          move ACC to Memory
  MOV   A,[saddrp]     move Memory to ACC
  MOV   [saddrp],A     move ACC to Memory


  MOV   A,!addr16      move Memory to ACC
  MOV   !addr16,A      move ACC to Memory
  MOV   A,sfr          move SFR to ACC
  MOV   sfr,A          move ACC to SFR
  XCH   A,r1           exchange ACC and G-reg
  XCH   r2,r1          exchange G-reg and G-reg
  XCH   A,mem          exchange ACC and Memory
  XCH   A,saddr        exchange ACC and SDMEM
* XCH   A,sfr          exchange ACC and SFR
  XCH   A,[saddrp]     exchange ACC and Memory

  XCH   saddr,saddr    exchange SDMEM and SDMEM


------ 16-bit Transfer Instructions -----

  MOVW  rp1,#word      move Immediate to G-reg Pair
  MOVW  rp1,!addr16    move Memory to G-reg Pair
  MOVW  !addr16,rp1    move G-reg Pair to Memory
  MOVW  saddrp,#word   move Immediate to SDMEM Pair
  MOVW  sfrp,#word     move Immediate to SFR Pair
  MOVW  rp1,rp1        move G-reg Pair to G-reg Pair
  MOVW  AX,saddrp      move SDMEM Pair to Extended ACC
  MOVW  saddrp,AX      move Extended ACC to SDMEM Pair
  MOVW  saddrp,saddrp  move SDMEM Pair to SDMEM Pair

  MOVW  AX,sfrp        move SFR Pair to Extended ACC
  MOVW  sfrp,AX        move Extended ACC TO SFR Pair
  XCHW  AX,saddrp      exchange Extended ACC and SDMEM Pair
* XCHW  AX,sfrp        exchange Extended ACC and SFR Pair
  XCHW  saddrp,saddrp  exchange SDMEM Pair and SDMEM Pair

  XCHW  rp1,rp1        exchange G-reg Pair and G-reg Pair
  PUSH  post           push G-reg Pair on System Stack
  POP   post           pop G-reg Pair off System Stack
  PUSHU post           push G-reg Pair on User Stack
  POPU  post           pop G-reg Pair off User Stack
  PUSH  PSW            push PSW on System Stack
  POP   PSW            pop PSW off System Stack
```

```
------ 8-bit Arithmetic and Logic Instructions -----

  ADD   A,#byte          add Immediate to ACC
  ADDC  A,#byte          add Immediate to ACC with Carry
  SUB   A,#byte          subtract Immediate from ACC
  SUBC  A,#byte          subtract Immediate from ACC with Borrow
  AND   A,#byte          and Immediate with ACC
  OR    A,#byte          or Immediate with ACC
  XOR   A,#byte          exclusive-or Immediate with ACC
  CMP   A,#byte          compare Immediate with ACC

  ADD   saddr,#byte      add Immediate to SDMEM
  ADDC  saddr,#byte      add Immediate to SDMEM with Carry
  SUB   saddr,#byte      subtract Immediate from SDMEM
  SUBC  saddr,#byte      subtract Immediate from SDMEM with Borrow
  AND   saddr,#byte      and Immediate with SDMEM
  OR    saddr,#byte      or Immediate with SDMEM
  XOR   saddr,#byte      exclusive-or Immediate with SDMEM
  CMP   saddr,#byte      compare Immediate with SDMEM

* ADD   sfr,#byte        add Immediate to SFR
* ADDC  sfr,#byte        add Immediate to SFR with Carry
* SUB   sfr,#byte        subtract Immediate from SFR
* SUBC  sfr,#byte        subtract Immediate from SFR with Borrow
* AND   sfr,#byte        and Immediate with SFR
* OR    sfr,#byte        or Immediate with SFR
* XOR   sfr,#byte        exclusive-or Immediate with SFR
* CMP   sfr,#byte        compare Immediate with SFR

  ADD   r2,r1            add G-reg to G-reg
  ADDC  r2,r1            add G-reg to G-reg with Carry
  SUB   r2,r1            subtract G-reg from G-REG
  SUBC  r2,r1            subtract G-reg from G-REG with Borrow
  AND   r2,r1            and G-reg with G-REG
  OR    r2,r1            or G-reg with G-REG
  XOR   r2,r1            exclusive-or G-reg with G-REG
  CMP   r2,r1            compare G-reg with G-REG

  ADD   A,saddr          add SDMEM to ACC
  ADDC  A,saddr          add SDMEM to ACC with Carry
  SUB   A,saddr          subtract SDMEM from ACC
  SUBC  A,saddr          subtract SDMEM from ACC with Borrow
  AND   A,saddr          and SDMEM with ACC
  OR    A,saddr          or SDMEM with ACC
  XOR   A,saddr          exclusive-or SDMEM with ACC
  CMP   A,saddr          compare SDMEM with ACC

* ADD   A,sfr            add SFR to ACC
* ADDC  A,sfr            add SFR to ACC with Carry
* SUB   A,sfr            subtract SFR from ACC
* SUBC  A,sfr            subtract SFR from ACC with Borrow
* AND   A,sfr            and SFR with ACC
* OR    A,sfr            or SFR with ACC
* XOR   A,sfr            exclusive-or SFR with ACC
* CMP   A,sfr            compare SFR with ACC
```

ADD	saddr,saddr	add SDMEM to SDMEM
ADDC	saddr,saddr	add SDMEM to SDMEM with Carry
SUB	saddr,saddr	subtract SDMEM from SDMEM
SUBC	saddr,saddr	subtract SDMEM from SDMEM with Borrow
AND	saddr,saddr	and SDMEM with SDMEM
OR	saddr,saddr	or SDMEM with SDMEM
XOR	saddr,saddr	exclusive-or SDMEM with SDMEM
CMP	saddr,saddr	compare SDMEM with SDMEM

ADD	A,mem	add Memory to ACC
ADDC	A,mem	add Memory to ACC with Carry
SUB	A,mem	subtract Memory from ACC
SUBC	A,mem	subtract Memory from ACC with Borrow
AND	A,mem	and Memory with ACC
OR	A,mem	or Memory with ACC
XOR	A,mem	exclusive-or Memory with ACC
CMP	A,mem	compare Memory with ACC
ADD	mem,A	add ACC to Memory
ADDC	mem,A	add ACC to Memory with Carry
SUB	mem,A	subtract ACC from Memory
SUBC	mem,A	subtract ACC from Memory with Borrow
AND	mem,A	and ACC with Memory
OR	mem,A	or ACC with Memory
XOR	mem,A	exclusive-or ACC with Memory
CMP	mem,A	compare ACC with Memory
INC	r1	Increment G-reg
DEC	r1	Decrement G-reg
INC	saddr	Increment SDMEM
DEC	saddr	Decrement SDMEM

```
------ 16-bit Arithmetic and Logic Instructions -----

  ADDW  AX,#word          add Immediate to Extended ACC
  SUBW  AX,#word          subtract Immediate from Extended ACC
  CMPW  AX,#word          compare Immediate with Extended ACC

  ADDW  saddrp,#word      add Immediate to SDMEM Pair
  SUBW  saddrp,#word      subtract Immediate from SDMEM Pair
  CMPW  saddrp,#word      compare Immediate with SDMEM Pair

* ADDW  sfrp,#word        add Immediate to SFR Pair
* SUBW  sfrp,#word        subtract Immediate from SFR Pair
* CMPW  sfrp,#word        compare Immediate with SFR Pair

  ADDW  rp1 ,rp1          add G-reg Pair  to G-reg Pair
  SUBW  rp1 ,rp1          subtract G-reg Pair  from G-reg Pair
  CMPW  rp1 ,rp1          compare G-reg Pair  with G-reg Pair

  ADDW  AX,saddrp         add SDMEM Pair  to Extended ACC
  SUBW  AX,saddrp         subtract SDMEM Pair  from Extended ACC
  CMPW  AX,saddrp         compare SDMEM Pair  with Extended ACC

  ADDW  saddrp,saddrp     add SDMEM Pair  to SDMEM Pair
  SUBW  saddrp,saddrp     subtract SDMEM Pair  from SDMEM Pair
  CMPW  saddrp,saddrp     compare SDMEM Pair  with SDMEM Pair



  INCW  rp2               increment G-reg Pair
  DECW  rp2               decrement G-reg Pair
  INCW  saddrp            increment SDMEM Pair
  DECW  saddrp            decrement SDMEM Pair



------ Multiply and Divide Instructions ------

  MULU  r1                multiply ACC by G-reg
  DIVUW r1                Divide Extended ACC by G-reg
  MULUW rp1               multiply Extended ACC by G-reg Pair
  DIVUX rp1               divide Extended ACC and DE by G-reg Pair
```

----- Bit Manipulation Instructions -----

```
MOV1  CY,saddr.bit      move SDMEM bit to Carry
MOV1  saddr.bit,CY      move Carry to SDMEM bit
AND1  CY,saddr.bit      and SDMEM bit with Carry
OR1   CY,saddr.bit      or SDMEM bit with Carry
AND1  CY,/saddr.bit     and complement SDMEM bit with Carry
OR1   CY,/saddr.bit     or complement SDMEM bit with Carry
XOR1  CY,saddr.bit      exclusive-or SDMEM bit with Carry
SET1  saddr.bit         set SDMEM bit
CLR1  saddr.bit         clear SDMEM bit
NOT1  saddr.bit         complement SDMEM bit

MOV1  CY,sfr.bit        move SFR bit to Carry
MOV1  sfr.bit,CY        move Carry to SFR bit
AND1  CY,sfr.bit        and SFR bit with Carry
OR1   CY,sfr.bit        or SFR bit with Carry
AND1  CY,/sfr.bit       and complement SFR bit with Carry
OR1   CY,/sfr.bit       or complement SFR bit with Carry
XOR1  CY,sfr.bit        exclusive-or SFR bit with Carry
SET1  sfr.bit           set SFR bit
CLR1  sfr.bit           clear SFR bit
NOT1  sfr.bit           complement SFR bit

MOV1  CY,A.bit          move ACC bit to Carry
MOV1  A.bit,CY          move Carry to ACC bit
AND1  CY,A.bit          and ACC bit with Carry
OR1   CY,A.bit          or ACC bit with Carry
AND1  CY,/A.bit         and complement ACC bit with Carry
OR1   CY,/A.bit         or complement ACC bit with Carry
XOR1  CY,A.bit          exclusive-or ACC bit with Carry
SET1  A.bit             set ACC bit
CLR1  A.bit             clear ACC bit
NOT1  A.bit             complement ACC bit

MOV1  CY,X.bit          move X-reg bit to Carry
MOV1  X.bit,CY          move Carry to X-reg bit
AND1  CY,X.bit          and X-reg bit with Carry
OR1   CY,X.bit          or X-reg bit with Carry
AND1  CY,/X.bit         and complement X-reg bit with Carry
OR1   CY,/X.bit         or complement X-reg bit with Carry
XOR1  CY,X.bit          exclusive-or X-reg bit with Carry
SET1  X.bit             set X-reg bit
CLR1  X.bit             clear X-reg bit
NOT1  X.bit             complement X-reg bit
```

```
MOV1  CY,PSWH .bit      move PSWH  bit to Carry
MOV1  PSWH .bit,CY      move Carry to PSWH  bit
AND1  CY,PSWH .bit      and PSWH  bit with Carry
OR1   CY,PSWH .bit      or PSWH  bit with Carry
AND1  CY,/PSWH .bit     and complement PSWH  bit with Carry
OR1   CY,/PSWH .bit     or complement PSWH  bit with Carry
XOR1  CY,PSWH .bit      exclusive-or PSWH  bit with Carry
SET1  PSWH .bit         set PSWH  bit
CLR1  PSWH .bit         clear PSWH  bit
NOT1  PSWH .bit         complement PSWH  bit

MOV1  CY,PSWL .bit      move PSWL  bit to Carry
MOV1  PSWL .bit,CY      move Carry to PSWL  bit
AND1  CY,PSWL .bit      and PSWL  bit with Carry
OR1   CY,PSWL .bit      or PSWL  bit with Carry
AND1  CY,/PSWL .bit     and complement PSWL  bit with Carry
OR1   CY,/PSWL .bit     or complement PSWL  bit with Carry
XOR1  CY,PSWL .bit      exclusive-or PSWL  bit with Carry
SET1  PSWL .bit         set PSWL  bit
CLR1  PSWL .bit         clear PSWL  bit
NOT1  PSWL .bit         complement PSWL  bit

CLR1  CY                clear Carry
NOT1  CY                complement Carry
SET1  CY                set Carry
```

```
----- Branch Instructions -----

  CALL  !addr16          call subroutine absolute
  CALLF !addr11          call subroutine in fixed area
  CALLT [addr5]          call subroutine table address
  CALL  rp1              call subroutine register
  CALL  [rp1]            call subroutine register indirect
  BRK                    software interrupt

  RET                    return from subroutine
  RETI                   return from interrupt

  BR    !addr16          branch absolute
  BR    rp1              branch register
  BR    [rp1]            branch register indirect
  BR    $addr16          branch relative
  BC    $addr16          branch if Carry
= BL    $addr16          branch if Lower
  BNC   $addr16          branch if not Carry
= BNL   $addr16          branch if not Lower
  BZ    $addr16          branch if Zero
= BE    $addr16          branch if Equal
  BNZ   $addr16          branch if not Zero
= BNE   $addr16          branch if not Equal
  BV    $addr16          branch if Overflow
= BPE   $addr16          branch if Even Parity
  BNV   $addr16          branch if not Overflow
= BPO   $addr16          branch if Odd Parity
  BN    $addr16          branch if Negative
  BP    $addr16          branch if Positive
  BGE   $addr16          branch if Greater than or Equal
  BGT   $addr16          branch if Greater than
  BLE   $addr16          branch if Less than or Equal
  BLT   $addr16          branch if Less than
  BH    $addr16          branch if Higher
  BNH   $addr16          branch if not Higher
```

```
BT    saddr.bit,$addr16 branch if SDMEM bit True
BF    saddr.bit,$addr16 branch if SDMEM bit False
BTCLR saddr.bit,$addr16 branch and clear if SDMEM bit True
BFSET saddr.bit,$addr16 branch and set if SDMEM bit False

BT    sfr.bit,$addr16   branch if SFR bit True
BF    sfr.bit,$addr16   branch if SFR bit False
BTCLR sfr.bit,$addr16   branch and clear if SFR bit True
BFSET sfr.bit,$addr16   branch and set if SFR bit False

BT    A.bit,$addr16     branch if ACC bit True
BF    A.bit,$addr16     branch if ACC bit False
BTCLR A.bit,$addr16     branch and clear if ACC bit True
BFSET A.bit,$addr16     branch and set if ACC bit False

BT    X.bit,$addr16 branch if X-reg bit True
BF    X.bit,$addr16 branch if X-reg bit False
BTCLR X.bit,$addr16 branch and clear if X-reg bit True
BFSET X.bit,$addr16 branch and set if X-reg bit False

BT    PSWH .bit,$addr16 branch if PSWH  bit True
BF    PSWH .bit,$addr16 branch if PSWH  bit False
BTCLR PSWH .bit,$addr16 branch and clear if PSWH  bit True
BFSET PSWH .bit,$addr16 branch and set if PSWH  bit False

BT    PSWL .bit,$addr16 branch if PSWL  bit True
BF    PSWL .bit,$addr16 branch if PSWL  bit False
BTCLR PSWL .bit,$addr16 branch and clear if PSWL  bit True
BFSET PSWL .bit,$addr16 branch and set if PSWL  bit False

DBNZ  r3,$addr16        decrement G-reg & branch relative if not zero
DBNZ  saddr,$addr16     decrement SDMEM & branch relative if not zero
```

```
----- CPU Control Instruction -----

NOP             no operation
EI              enable interrupt
DI              disable interrupt
BRKCS RBn       run context switch
RETCS RBn       return from context switch
SWRS            switch Register Set
SEL   RBn       select Register Bank & Main Register Set
SEL   RBn,ALT   select Register Bank & Alternate Register Set
INCW  SP        increment SP
DECW  SP        decrement SP
MOV  STBC,#byte move immediate to write protected SFR (standby)
MOV  WDM,#byte  move immediate to write protected SFR (watchdog timer)
MOV  PSWL,#byte move immediate to PSWL
MOV  PSWH,#byte move immediate to PSWH
MOV  PSWL,A     move ACC to PSWL
MOV  PSWH,A     move ACC to PSWH
MOV  A,PSWL     move PSWL to ACC
MOV  A,PSWH     move PSWH to ACC
MOVW SP,#word   move immediate to SP
MOVW SP,AX      move AX to SP
MOVW AX,SP      move SP to AX


----- Decimal Adjust Instruction -----

ADJ4            adjust decimal ACC

----- String Manipulation Instruction -----

MOVM    [DE+],A       move multiple
        [DE-],A
MOVBK   [DE+],[HL+]   move block
        [DE-],[HL-]
XCHM    [DE+],A       exchange multiple
        [DE-],A
XCHBK   [DE+],[HL+]   exchange block
        [DE-],[HL-]
CMPME   [DE+],A       compare multiple if equal
        [DE-],A
CMPBKE  [DE+],[HL+]   compare block if equal
        [DE-],[HL-]
CMPMNE  [DE+],A       compare multiple if not equal
        [DE-],A
CMPBKNE [DE+],[HL+]   compare block if not equal
        [DE-],[HL-]
CMPMNC  [DE+],A       compare multiple if not Carry
        [DE-],A             (greater than or equal)
CMPMC   [DE+],A       compare multiple if Carry
        [DE-],A             (less than)
CMPBKNC [DE+],[HL+]   compare block if not Carry
        [DE-],[HL-]
CMPBKC  [DE+],[HL+]   compare block if Carry
        [DE-],[HL-]
```

----- Shift and Rotate Instructions -----

```
SHR   r1,n          shift right G-reg n times
SHL   r1,n          shift left G-reg n times
ROR   r1,n          rotate ritht G-reg n times
ROL   r1,n          rotate left G-reg n times
RORC  r1,n          rotate right G-reg n times with Carry
ROLC  r1,n          rotate left G-reg n times with Carry
SHRW  rp1,n         shift right G-reg Pair n times
SHLW  rp1,n         shift left G-reg Pair n times
ROR4  [rp1]         rotate right digit
ROL4  [rp1]         rotate left digit
```

***** NOTE *****

SDMEM: short direct Memory

```
r1:     R0,R1,R2,R3,R4,R5,R6,R7
r2:     R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15
r3:     C,B

rp1:    RP0,RP1,RP2,RP3,RP4,RP5,RP6,RP7
rp2:    DE,HL,VP,UP

mem:    [DE+],[HL+],[DE-],[HL-],[DE],[HL],[VP],[UP] →  register indirect
        [DE+A],[HL+A],[DE+B],[HL+B],[VP+DE],[VP+HL] →  base index
        [DE+byte],[HL+byte],[VP+byte],[UP+byte],[SP+byte] →  base
        word[A],word[B],word[DE],word[HL]           →  index
```

４．７．２　ＳＦＲマッピング

8 BIT SFR MAPPING

FF00	P0	FF20	PM0	FF40	MM	FF60	FRCC
FF01	P1	FF21	PM1	FF41	RFM	FF61	
FF02	P2	FF22	PM2	FF42	WDM	FF62	
FF03	P3	FF23	PM3	FF43		FF63	
FF04	P4	FF24		FF44	STBC	FF64	CPTM
FF05	P5	FF25	PM5	FF45		FF65	
FF06		FF26		FF46	TBM	FF66	PWMM
FF07		FF27		FF47		FF67	
FF08	CR00L	FF28		FF48	INTM	FF68	ADM
FF09	CR00H	FF29		FF49		FF69	
FF0A	CR01L	FF2A		FF4A	ISPR	FF6A	ADCR
FF0B	CR01H	FF2B		FF4B		FF6B	
FF0C	CR10L	FF2C		FF4C		FF6C	
FF0D	CR10H	FF2D		FF4D		FF6D	
FF0E	CR11L	FF2E		FF4E	CCW	FF6E	
FF0F	CR11H	FF2F		FF4F		FF6F	
FF10	CPT0L	FF30		FF50	SCM	FF70	CUIM
FF11	CPT0H	FF31		FF51		FF71	
FF12	CPT1L	FF32	PMC2	FF52	SCC	FF72	UDCC0
FF13	CPT1H	FF33	PMC3	FF53	BRG	FF73	
FF14	PWM0L	FF34		FF54		FF74	CRC
FF15	PWM0H	FF35		FF55		FF75	
FF16	PWM1L	FF36		FF56	RXB	FF76	
FF17	PWM1H	FF37		FF57	TXB	FF77	
FF18		FF38	RTPC	FF58		FF78	
FF19		FF39		FF59		FF79	
FF1A		FF3A	POL	FF5A		FF7A	UDCC1
FF1B		FF3B	POH	FF5B		FF7B	
FF1C	UDC0L	FF3C		FF5C		FF7C	
FF1D	UDC0H	FF3D		FF5D		FF7D	
FF1E	UDC1L	FF3E		FF5E		FF7E	
FF1F	UDC1H	FF3F		FF5F		FF7F	

FF80	TMC0	FFA0		FFC0	CRIC00	FFE0	ADIC
FF81		FFA1		FFC1	CRMS00	FFE1	ADMS
FF82	TMC1	FFA2		FFC2	CRIC01	FFE2	TBIC
FF83		FFA3		FFC3		FFE3	
FF84		FFA4		FFC4	CRIC10	FFE4	
FF85		FFA5		FFC5	CRMS10	FFE5	
FF86		FFA6		FFC6	CRIC11	FFE6	
FF87		FFA7		FFC7		FFE7	
FF88	TMOL	FFA8		FFC8	EXIC0	FFE8	
FF89	TMOH	FFA9		FFC9	EXMS0	FFE9	
FF8A	MDOL	FFAA		FFCA	EXIC1	FFEA	
FF8B	MDOH	FFAB		FFCB	EXMS1	FFEB	
FF8C	TM1L	FFAC		FFCC	EXIC2	FFEC	
FF8D	TM1H	FFAD		FFCD	EXMS2	FFED	
FF8E	MD1L	FFAE		FFCE	TMIC0	FFEE	
FF8F	MD1H	FFAF		FFCF	TMMS0	FFEF	
FF90		FFB0	EXTSFR0	FFD0	TMIC1	FFF0	
FF91		FFB1	EXTSFR1	FFD1	TMMS1	FFF1	
FF92		FFB2	EXTSFR2	FFD2	TMIC2	FFF2	
FF93		FFB3	EXTSFR3	FFD3	TMMS2	FFF3	
FF94		FFB4	EXTSFR4	FFD4		FFF4	
FF95		FFB5	EXTSFR5	FFD5		FFF5	
FF96		FFB6	EXTSFR6	FFD6		FFF6	
FF97		FFB7	EXTSFR7	FFD7		FFF7	
FF98		FFB8	EXTSFR8	FFD8		FFF8	
FF99		FFB9	EXTSFR9	FFD9		FFF9	
FF9A		FFBA	EXTSFR10	FFDA	SEIC	FFFA	
FF9B		FFBB	EXTSFR11	FFDB		FFFB	
FF9C		FFBC	EXTSFR12	FFDC	SRIC	FFFC	
FF9D		FFBD	EXTSFR13	FFDD	SRMS	FFFD	
FF9E		FFBE	EXTSFR14	FFDE	STIC	FFFE	
FF9F		FFBF	EXTSFR15	FFDF	STMS	FFFF	

16 BIT SFR MAPPING

FF00		FF20	FF40	FF60
FF01		FF21	FF41	FF61
FF02		FF22	FF42	FF62
FF03		FF23	FF43	FF63
FF04		FF24	FF44	FF64
FF05		FF25	FF45	FF65
FF06		FF26	FF46	FF66
FF07		FF27	FF47	FF67
FF08	CR00	FF28	FF48	FF68
FF09		FF29	FF49	FF69
FF0A	CR01	FF2A	FF4A	FF6A
FF0B		FF2B	FF4B	FF6B
FF0C	CR10	FF2C	FF4C	FF6C
FF0D		FF2D	FF4D	FF6D
FF0E	CR11	FF2E	FF4E	FF6E
FF0F		FF2F	FF4F	FF6F
FF10	CPT0	FF30	FF50	FF70
FF11		FF31	FF51	FF71
FF12	CPT1	FF32	FF52	FF72
FF13		FF33	FF53	FF73
FF14	PWM0	FF34	FF54	FF74
FF15		FF35	FF55	FF75
FF16	PWM1	FF36	FF56	FF76
FF17		FF37	FF57	FF77
FF18		FF38	FF58	FF78
FF19		FF39	FF59	FF79
FF1A		FF3A	FF5A	FF7A
FF1B		FF3B	FF5B	FF7B
FF1C	UDC0	FF3C	FF5C	FF7C
FF1D		FF3D	FF5D	FF7D
FF1E	UDC1	FF3E	FF5E	FF7E
FF1F		FF3F	FF5F	FF7F

FF80		FFA0	FFC0	FFE0
FF81		FFA1	FFC1	FFE1
FF82		FFA2	FFC2	FFE2
FF83		FFA3	FFC3	FFE3
FF84		FFA4	FFC4	FFE4
FF85		FFA5	FFC5	FFE5
FF86		FFA6	FFC6	FFE6
FF87		FFA7	FFC7	FFE7
FF88	TM0	FFA8	FFC8	FFE8
FF89		FFA9	FFC9	FFE9
FF8A	MD0	FFAA	FFCA	FFEA
FF8B		FFAB	FFCB	FFEB
FF8C	TM1	FFAC	FFCC	FFEC
FF8D		FFAD	FFCD	FFED
FF8E	MD1	FFAE	FFCE	FFEE
FF8F		FFAF	FFCF	FFEF
FF90		FFB0	FFD0	FFF0
FF91		FFB1	FFD1	FFF1
FF92		FFB2	FFD2	FFF2
FF93		FFB3	FFD3	FFF3
FF94		FFB4	FFD4	FFF4
FF95		FFB5	FFD5	FFF5
FF96		FFB6	FFD6	FFF6
FF97		FFB7	FFD7	FFF7
FF98		FFB8	FFD8	FFF8
FF99		FFB9	FFD9	FFF9
FF9A		FFBA	FFDA	FFFA
FF9B		FFBB	FFDB	FFFB
FF9C		FFBC	FFDC	FFFC
FF9D		FFBD	FFDD	FFFD
FF9E		FFBE	FFDE	FFFE
FF9F		FFBF	FFDF	FFFF

４．７．３　オンライン・アセンブラ仕様

（1）文字セット

Ａ～Ｚ　ａ～ｚ　＠　？　＿（アンダー・バー）　０～９　＋　－　＊

／　＄　！　［　］　＃　（　）　；

．（ピリオド）　，（コンマ）　＼（バック・スラッシュ）

また，小文字は大文字相当として扱われます。

（2）シンボル定義

本アセンブラでは，シンボルを定義することはできません。

ただし，数値のかわりに定義済みのシンボルを用いることができます。

```
例 ＭＯＶＷ     ＨＬ，＃ｓｙｍｂｏｌ ┐
   ＯＲＧ       ｓｙｍｂｏｌ          ├── ＯＫ
   ｓｙｍｂｏｌ：  ＮＯＰ             ──── エラー
```

（3）コメント行

本アセンブラでは，；（セミコロン）以降Ｃｒまでをコメントとして扱います。

```
例 ；  ｃｏｍｍｅｎｔ          ┐
   ＮＯＰ   ；  ｃｏｍｍｅｎｔ ├── ＯＫ
```

（4）オペランドに用いる数値表現

コマンド仕様における数値と同じ表現が可能です。

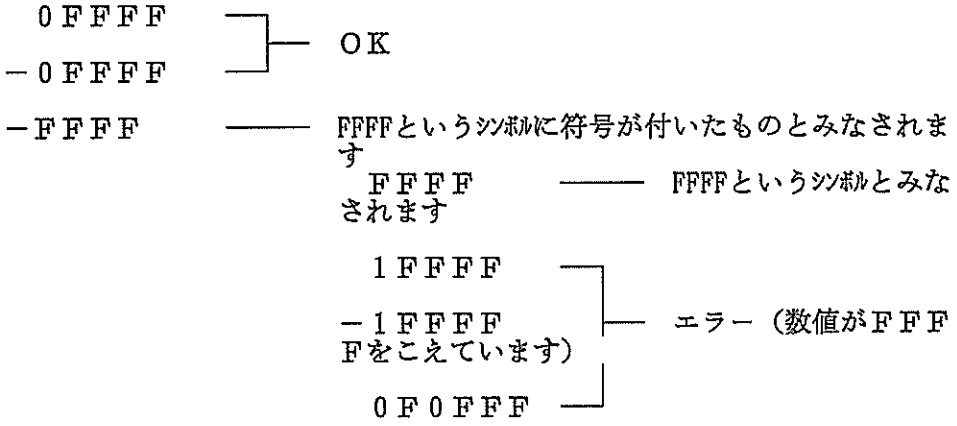
すなわち，数値は１６進，１０進，８進，２進で入力することができ，それぞれ数値の最後にＨ，Ｔ，Ｑ，Ｙを付加します。ただし，ＳＵＦコマンドであらかじめ指定されている場合，このサフィックスを省略することができます。

数値は，数字（０～９）で始まらなくてはなりません。

また，数値の前に符号（＋，－）をつけることもできます。

数値は０から１６進でＦＦＦＦ，１０進で６５５３５，８進で１７７７７７，２進で１１１１１１１１１１１１１１１１までの範囲でなければなりません。これより大きい数値が入力された場合エラーとなります。

例1 サフィックスがHのとき



また，ＡＳＣＩＩコードでの入力はできません。

すなわち，ＡＳＣＩＩ文字をクォーテーション・マークで囲むことにより，コード変換する機能は持っていません。

例2　ＭＯＶＷ　ＡＸ，＃３１３２Ｈ　――――　ＯＫ

　　ＭＯＶＷ　ＡＸ，＃”ＡＢ”　――――　エラー

（５）オペランドに用いるシンボル表現

数値の代わりに，すでに定義済みのシンボルを用いることができます。

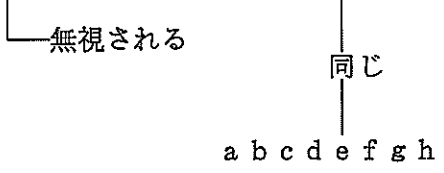
シンボルは，Ａ～Ｚ，ａ～ｚ，@，？，＿（アンダー・バー），０～９のいずれかの文字から構成されます。

ただし，シンボルは数字から始まってはいけません。

また，英小文字（ａ～ｚ）は英大文字（Ａ～Ｚ）と同じ意味に扱われます。

シンボルは，最大８文字で構成され，サフィックスを付加することはできません。付加した場合は，そこまでがシンボルと解釈されます。

例1　ＡＢＣＤＥＦＧＨＩＪＫ　――同じ――ＡＢＣＤＥＦＧＨ



ローカル・シンボルは，モジュール名と対にして表現しなければなりません。

パブリック・シンボルは，モジュール名を省略して入力します。

たとえば，“ＭＯＤＵＬＥ０１”というモジュール内で定義されている“ＳＹＭＢＯＬ０１”というローカル・シンボルは次のように記述します。

MODULE０１＼SYMBOL０１
　　└── バック・スラッシュ

同じく，“MODULE０１”というモジュール内で外部定義されている“SYMBOL１０”というパブリック・シンボルは，次のように記述します。

SYMBOL１０

パブリック・シンボルに関しては，モジュール名を省略して入力します。

シンボルは，CODE，DATA，NUMBERの３種類のタイプとＢＩＴタイプに区別されます。

例2
```
MOVW    HL, #code symbol      ┐
MOVW    HL, #data symbol      ├── OK
MOVW    HL, #number symbol    ┘
MOVW    HL, #bit symbol       ──── エラー
MOV1    CY, bit symbol        ──── OK
MOV1    CY, code(data)(number)symbol ──── エラー
MOV1    CY, code(data)(number)symbol. bit ──── OK
                  └── ただし，symbol値はFE20H ～
                      FFFFH でなければなりません
```

（６）オペランドに用いる式表現

数値の代わりに式表現を用いることができます。

式表現には，以下の演算子を用いることができます。

＋　－　＊　／　AND　OR　XOR　（　）

／── AND１，ＯＲ１命令で式の先頭にある／は，演算子としての意味をもちません（この場合はＢＩＴ反転の意味にとられます）。これ以外の場合，式の先頭に／があるとエラーとなります。

これらの演算子の優先順位は次のとおりです。

高 ←――― 優先順位 ―――→ 低					
（　）	＊　／	＋　－	AND	OR	XOR

なお，演算はすべて１６ビットの正の整数で行ない，小数点以下は切り捨てられます。

また，演算の中間結果，最終結果で，オーバフローした場合，つまり，１６ビットの正の整数をこえた場合，１７ビット目より上位は切捨てられます。

また，負になった場合，その結果に１００００Ｈを加えます。

ゼロ・ディバイドはエラーになります。

例	１０Ｈ＋１０Ｈ	→ ２０Ｈ
	１０Ｈ／３Ｈ	→ ５Ｈ
	０ＦＦＦＦＨ＋２Ｈ	→ １Ｈ
	１Ｈ－２Ｈ	→ ０ＦＦＦＦＨ
	（１００Ｈ＊１００Ｈ）／（１００Ｈ＊１００Ｈ）	→ エラー（ゼロ・ディバイド）

演算は数値，式，シンボルに対してだけ有効です。ただし，ＢＩＴタイプのシンボルに対して演算をすると，エラーになります。また，予約語に対して演算をしますとエラーになります。

なお，数値あるいは式あるいはシンボルと　（　）　＊　／　＋　－　の各演算子は，連続していてもかまいません。

数値あるいは式あるいはシンボルと，ＡＮＤ，ＯＲ，ＸＯＲの各演算子との間には，一つ以上のスペースがなければなりません。

ＴＡＢは，スペースと同じにみなされます。

（７）疑似命令

本アセンブラでは以下の疑似命令をサポートしています。

① ＯＲＧ　　ａｄｄｒ１６

ＯＲＧ命令は，ＯＲＧ命令の次の命令をａｄｄｒ１６に置きます。

ａｄｄｒ１６が，カレント・ロケーションよりも小さい値だった場合，Ｃａｕｔｉｏｎが表示されます。

また，ａｄｄｒ１６＞ＦＥ７ＦＨの場合，Ｅｒｒｏｒが表示されます。

② ＤＢ　　ｂｙｔｅ，ｂｙｔｅ‥‥

ＤＢ命令は，カレント・ロケーションにｏｐｅｒａｎｄにあるｂｙｔｅデータを置きます。複数のｂｙｔｅデータがコンマ（,）で区切られて存在する場合，カレント・ロケーション以降に順番に置かれます。このとき，データを置くロケー

ションが，ＦＥ７ＦＨをこえた場合は，Ｅｒｒｏｒが表示されます。

オペランドにｗｏｒｄデータがあった場合，Ｅｒｒｏｒが表示されます。

エラーがあった場合，データはすべて無効となります。

③　ＤＷ　　ｗｏｒｄ，ｗｏｒｄ‥‥

ＤＷ命令は，（カレント・ロケーション）にｗｏｒｄデータのＬｏｗ　ｂｙｔｅを，（カレント・ロケーション＋１）にｗｏｒｄデータのＨｉｇｈ　ｂｙｔｅを置きます。

複数のｗｏｒｄデータがコンマ（,）で区切られて存在する場合，カレント・ロケーション以降に上の規則に従って順番に置かれます。このとき，データを置くロケーションがＦＥ７ＦＨをこえた場合は，Ｅｒｒｏｒが表示されます。また，ＦＥ１ＦＨ～ＦＥ７ＤＨの範囲の奇数アドレスにｗｏｒｄデータを置こうとした場合，Ｗａｒｎｉｎｇが表示されます。

エラーがあった場合，データはすべて無効となります。

④　ＤＳ　　ｗｏｒｄ

ＤＳ命令は，ＤＳ命令の次の命令を，（カレント・ロケーション＋ｗｏｒｄ）に置きます。ただし，（カレント・ロケーション＋ｗｏｒｄ）が　ＦＥ８０Ｈ～ＦＦＦＦＨとなった場合，Ｅｒｒｏｒが表示されます。

また，（カレント・ロケーション＋ｗｏｒｄ）が　ＦＦＦＦＨをこえた場合，オーバフローした上位桁は切捨てられます。このとき，Ｃａｕｔｉｏｎが表示されます。

⑤　ＥＮＤ

ＥＮＤ命令は，ＡＳＭコマンドを終了します。

（8）同じニモニックでありながらアドレス空間により生成コードが異なる命令のコード生成規則

７８３１２／１０では，同じニモニックをもっていても，アドレス空間（FE20H ～ FF1FH = saddr ； FF00H～ FFFFH = SFR ）により，生成コードが異なる場合があります。

		命令長	実行サイクル
MOV　A,	saddr	2	3
	SFR	2	3
MOV	saddr, A	2	3
	SFR, A	2	3

MOV	{ saddr / SFR }, #byte	3	3
		3	3
MOVW	{ saddrp / SFRP }, #word	4	4
		4	3
MOV1	CY, { saddr / SFR }. bit	3	6
		3	6
MOV1	{ saddr / SFR }. bit , CY	3	7
		3	7
AND1	CY, (/) { saddr / SFR }. bit	3	6
		3	6
AND1	CY, (/) { saddr / SFR }. bit	3	6
		3	6
OR1	CY, (/) { saddr / SFR }. bit	3	6
		3	6
XOR1	CY, { saddr / SFR }. bit	3	6
		3	6
SET1	{ saddr / SFR }. bit	2	5
		3	6
CLR1	{ saddr / SFR }. bit	2	5
		3	6
NOT1	{ saddr / SFR }. bit	3	6
		3	6
BT	{ saddr / SFR }. bit , $addr16	3	9 (7)
		4	10 (7)
BF	{ saddr / SFR }. bit , $addr16	4	10 (7)
		4	10 (7)
BTCLR	{ saddr / SFR }. bit , $addr16	4	12 (7)
		4	12 (7)
BFSET	{ saddr / SFR }. bit , $addr16	4	12 (7)
		4	12 (7)

① アドレスがＳＦＲ予約語で表現された場合

・当該ＳＦＲがＦＦ２０Ｈ～ＦＦＦＦＨにマッピングされている場合，ＳＦＲを使用してコードを生成します。

・当該ＳＦＲがＦＦ００Ｈ～ＦＦ１ＦＨにマッピングされている場合，ｓａｄｄｒを使用してコードを生成します。

②　アドレスが数値あるいはシンボルあるいは式で表現された場合

・当該アドレスがＦＦ２０Ｈ～ＦＦＦＦＨにマッピングされている場合，ＳＦＲを使用してコードを生成します。

・当該アドレスがＦＥ２０Ｈ～ＦＦ１ＦＨにマッピングされている場合，ｓａｄｄｒを使用してコードを生成します。

・当該アドレスが０～ＦＥ１ＦＨにマッピングされている場合，エラーとなります。

（9）ＳＦＲ操作命令に対するエラー・チェック

ＳＦＲ空間に対する操作命令に関して，以下のエラー・チェックをします。

（ａ）アドレスがＳＦＲ予約語で表現された場合

①　ＳＦＲ，ＳＦＲＰのチェック
②　読出し専用，書込み専用の属性のチェック

をします。

①　ＳＦＲ予約語がＳＦＲかＳＦＲＰかの属性をもちます。ＳＦＲに対する１６ビット操作命令および，ＳＦＲＰに対する８ビット操作命令の場合，Ｗａｒｎｉｎｇが表示されます。

②　ＳＦＲ予約語が読出し専用（Ｒ／Ｏ），書込み専用（Ｗ／Ｏ），読み書き可能（Ｒ／Ｗ）の属性をもちます。

Ｒ／ＯのＳＦＲに対する書込み命令および，Ｗ／ＯのＳＦＲに対する読出し命令の場合，Ｗａｒｎｉｎｇが表示されます。

（ｂ）アドレスが数値あるいはシンボルあるいは式で表現された場合

①　ＳＦＲ存在のチェック
②　１６ビット・アクセス時のチェック
③　Ｒ／Ｗ属性のチェック

をします。

①　当該アドレスに実際にＳＦＲが存在しない場合，Ｗａｒｎｉｎｇが表示されます。

②　一つのアドレスに対し，ＳＦＲとＳＦＲＰが同時に存在する場合，アドレスに対してＳＦＲかＳＦＲＰかの属性をもたせることはできません。したがって，ＳＦＲＰが存在するアドレスに対する１６ビット操作命令の場合はエラー表示をしません。これ以外の場合，Ｗａｒｎｉｎｇが表示されます。

③ 当該アドレスが指示するＳＦＲはR／W属性に関しては一意的であるため，アドレスがＳＦＲ予約語で表現された場合と同様に，R／W属性のチェックをします。

(10) ｓａｄｄｒ操作命令に対するエラー・チェック

ｓａｄｄｒ空間に対する操作命令に関して，１６ビット操作時に限り，アドレスの偶数／奇数のチェックをします。

すなわち，奇数アドレスに対する１６ビット操作命令の場合，Ｗａｒｎｉｎｇが表示されます。

(11) アドレッシング・モード指定の省略

ブランチ系の命令で，absolute addressing か，relative addressing か，命令によってはっきりと決まっている場合，addressing mode の指定を省略することができます（条件付きブランチ，ＣＡＬＬ，ＣＡＬＬＦ，ＲＥＴＣＳ)。

例1

		省略		
ＢＣ	$addr16	⟶	ＢＣ	addr16
ＣＡＬＬ	!addr16	⟶	ＣＡＬＬ	addr16
ＣＡＬＬＦ	!addr16	⟶	ＣＡＬＬＦ	addr16
ＲＥＴＣＳ	!addr16	⟶	ＲＥＴＣＳ	addr16

ただし，ロケーション・カウンタの＄を用いる場合，addressing mode の指定を省略することはできません（ＢＣ　＄という表現はエラーになります)。

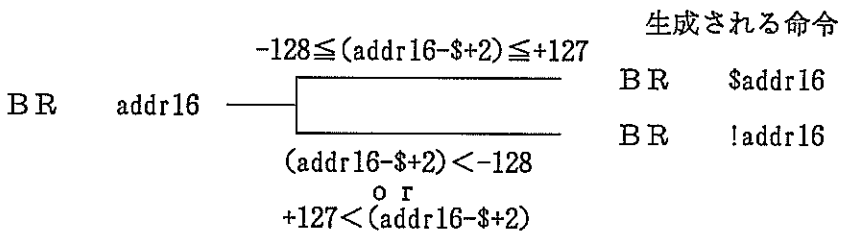
例2

addressing mode 指定の＄
｜ロケーション・カウンタの＄
｜｜

ＢＣ	＄＄±ｎ	―― ＯＫ	（＄±ｎ番地へのブランチ）
ＢＣ	＄＋ｎ	―― ＯＫ	（ｎ番地へのブランチ） （＄＋ｎ番地へのブランチではない！）
ＢＣ	＄－ｎ	―― ＯＫ	（１００００Ｈ－ｎ番地へのブランチ）

(12) ジェネリック・ブランチ

ブランチ命令（ＢＲ）でaddressing mode の指定を省略すると，ａｄｄｒ１６の値により，最短コードを生成します。このとき，Ｃａｕｔｉｏｎが表示されます。



ただし，ロケーション・カウンタの＄を用いる場合，addressing mode 指定を省略できないので，ジェネリック・ブランチは使用できません。

(13) ＶＰ，ＵＰ，ＤＥ，ＨＬ，とＲＰ４～ＲＰ７の対応

ｒｐ１（ＲＰ０～ＲＰ７）を用いる命令系に対し，ｒｐ２（ＶＰ，ＵＰ，ＤＥ，ＨＬ）の表現を用いることができます。このとき，Ｃａｕｔｉｏｎが表示されます。

ｒｐ２（ＶＰ，ＵＰ，ＤＥ，ＨＬ）を用いる命令系に対し，ｒｐ１（ＲＰ０～ＲＰ７）の表現を用いることができます。このときもＣａｕｔｉｏｎが表示されます。

対応のとり方は次のとおりです。

ｒｐ２		ｒｐ１
ＶＰ	――――	ＲＰ４
ＵＰ	――――	ＲＰ５
ＤＥ	――――	ＲＰ６
ＨＬ	――――	ＲＰ７

例　ＭＯＶＷ　ｒｐ１，＃word

	変換	
ＭＯＶＷ　ＨＬ，＃word	――――	ＭＯＶＷ　ＲＰ７，＃word
ＭＯＶＷ　ＶＰ，＃word	――――	ＭＯＶＷ　ＲＰ４，＃word

ＩＮＣＷ　ｒｐ２

ＩＮＣＷ　ＲＰ６	――――	ＩＮＣＷ　ＤＥ
ＩＮＣＷ　ＲＰ５	――――	ＩＮＣＷ　ＵＰ
ＩＮＣＷ　ＲＰ０	――――	エラー

(14) ＰＵＳＨ，ＰＯＰ，ＰＵＳＨＵ，ＰＯＰＵのオペランド記述

ＰＵＳＨ，ＰＯＰ命令のオペランドにはＲＰ０～ＲＰ７をコンマ（,）で区切って八つまで記述することができます。

ＰＵＳＨＵ，ＰＯＰＵ命令のオペランドにはＲＰ０～ＲＰ４，ＰＳＷ，ＲＰ６，ＲＰ７をコンマ（,）で区切って八つまで記述することができます。

オペランド記述の順番は規定されません。

ただし，同じオペランドが二つ以上記述された場合はＥｒｒｏｒが表示されます。

例　ＰＵＳＨ　ＲＰ７，ＲＰ５，ＲＰ６，ＲＰ０　　⟶ＯＫ

ＰＵＳＨ　ＲＰ７，ＲＰ５，ＲＰ６，ＲＰ０，ＲＰ７⟶Ｅｒｒｏｒ

ＰＵＳＨ　ＶＰ，ＵＰ，ＤＥ，ＨＬ　　⟶Ｃａｕｔｉｏｎ
（ＲＰ４，ＲＰ５，ＲＰ６，ＲＰ７）

ＰＵＳＨ　ＶＰ，ＵＰ，ＤＥ，ＨＬ，ＲＰ４　　⟶ Ｅｒｒｏｒ
（VPと RP4は同一レジスタ）

(15) エラー表示

本アセンブラは，エラー・コードとして

Ｅｒｒｏｒ

Ｗａｒｎｉｎｇ

Ｃａｕｔｉｏｎ

の３種類を表示します。

① Ｅｒｒｏｒ

オブジェクト・コードを生成できないか，あきらかにエラーである場合に表示されます。

例1 ＯＲＧ　０ＦＦ００Ｈ　（プログラム・エリアではない）

ＤＢ　０ＦＦ００Ｈ　（ワード・データである）

ＭＯＶ　Ｊ，#byte　（Ｊという予約語はない）

ＢＲ　$　（ジャンプ・アドレスがない）

② Ｗａｒｎｉｎｇ

オブジェクト・コードは生成できるが，正しい動作が望めない場合に表示されます。

例2 ＭＯＶ　　ＳＦＲＰ，＃byte　　（ＳＦＲＰに対する８ビット操作）
　　ＭＯＶＷ　ＳＦＲ，＃word　　（ＳＦＲに対する１６ビット操作）
　　ＭＯＶＷ　odd saddr,＃word　　（奇数 saddrに対する１６ビット操作）
　　ＢＲ　　０ＦＦ００Ｈ　　（ＳＦＲエリアへのブランチ）

③ Ｃａｕｔｉｏｎ

ジェネリックなオブジェクト生成が行なわれた場合，たとえば，ジェネリック・ブランチ，ｒｐ１とｒｐ２を置換した場合，あるいは注意を要する場合に表示されます。

例3 ＯＲＧ　　＄－１００　　（注意を要する）
　　ＢＲ　　１００Ｈ　　（ジェネリック・ブランチ）
　　ＭＯＶＷ　ＨＬ，＃word　　（ＭＯＶＷ　ＲＰ７，＃wordの変形）
　　ＩＮＣＷ　ＲＰ６　　（ＩＮＣＷ　ＤＥの変形）

★　(16) 予約語一覧表

(a) ニモニック

(A)	ADD	ADDC	ADDW	ADJ4	AND	AND1
(B)	BC	BE	BF	BFSET	BGE	BGT
	BH	BL	BLE	BLT	BN	BNC
	BNE	BNH	BNL	BNV	BNZ	BP
	BPE	BPO	BR	BRK	BRKCS	BT
	BTCLR	BV	BZ			
(C)	CALL	CALLF	CALLT	CLR1	CMP	CMPBKC
	CMPBKE	CMPBKNC	CMPBKNE	CMPMC	CMPME	CMPMNC
	CMPMNE	CMPW				
(D)	DBNZ	DEC	DECW	DI	DIVUW	DIVUX
(E)	EI					

(I) INC　　INCW

(M) MOV　　MOVBK　　MOVM　　MOVW　　MOV1　　MULU
　　MULUW

(N) NOP　　NOT1

(O) OR　　OR1

(P) POP　　POPU　　PUSH　　PUSHU

(R) RET　　RETCS　　RETI　　ROL　　ROLC　　ROL4
　　ROR　　RORC　　ROR4

(S) SEL　　SET1　　SHL　　SHLW　　SHR　　SHRW
　　SUB　　SUBC　　SUBW　　SWRS

(W) XCH　　XCHBK　　XCHM　　XCHW　　XOR　　XOR1

(b) 演算子

(A) AND

(E) EQ

(G) GE　　GT

(H) HIGH

(L) LE　　LOW　　LT

(M) MOD

(N) NE　　NOT

(O) OR

(S) SHL　　SHR

(X) XOR

(c) 疑似命令

(B) BR　　BSEG

(C) CSEG

(D) DB　　DBIT　　DS　　DSEG　　DW

(E) END　　ENDM　　EQU　　EXITM　　EXTBIT　　EXTRN

(I) IRP

(L) LOCAL

(M) MACRO

(N) NAME

(O) ORG

(P) PUBLIC

(R) REPT　　RSS

(S) SET

(d) 制御命令

(C) COND

(D) DEBUG　　DG

(E) EJ　　EJECT　　ELSE　　ELSEIF　　_ELSEIF　　ENDIF

(G) GEN

(I) IC　　IF　　_IF　　INCLUDE

(L) LI　　LIST

(N) NOCOND　　NODEBUG　　NODG　　NOGEN　　NOLI　　NOLIST

NOXR　　NOXREF

(P) PC　　PROCESSOR

(R) RESET

(S) SET　　SUBTITLE　　ST

(T) TITLE　　TT

(X) XR　　XREF

(e) ＳＦＲシンボル

(A) ADCR　　ADIC　　ADM　　ADMS

(B) BRG

(C) CCW　　CPT0　　CPT0H　　CPT0L　　CPT1　　CPT1H

CPT1L　　CPTM　　CR00　　CR00H　　CR00L　　CR01

CR01H　　CR01L　　CR10　　CR10H　　CR10L　　CR11

CR11H　　CR11L　　CRC　　CRIC00　　CRIC01　　CRIC10

CRIC11　　CRMS00　　CRMS10　　CUIM

(E) EXIC0　　EXIC1　　EXIC2　　EXMS0　　EXMS1　　EXMS2

(F) FRCC

(I) INTM　　ISPR

(M) MD0　　MD0H　　MD0L　　MD1　　MD1H　　MD1L

MM

(P)	P0	P0H	P0L	P1	P2	P3
	P4	P5	PM0	PM1	PM2	PM3
	PM5	PMC2	PMC3	PSW	PSWH	PSWL
	PWM0	PWM0H	PWM0L	PWM1	PWM1H	PWM1L
	PWMM					
(R)	RFM	RTPC	RXB			
(S)	SCC	SCM	SE1C	SP	SPH	SPL
	SR1C	SRMS	STBC	ST1C	STMS	
(T)	TB1C	TBM	TM0	TM0H	TM0L	TM1
	TM1H	TM1L	TMC0	TMC1	TMIC0	TMIC1
	TMIC2	TMMS0	TMMS1	TMMS2	TXB	
(U)	UDC0	UDC0H	UDC0L	UDC1	UDC1H	UDC1L
	UDCC0	UDCC1				
(W)	WDM					

(f) ＳＦＲビット・シンボル

(A)	ADCSE	ADF	ADISM	ADMK		
(C)	CRCSE00	CRCSE01	CRCSE10	CRCSE11	CRF00	CRF01
	CRF10	CRF11	CRISM00	CRISM10	CRMK00	CRMK01
	CRMK10	CRMK11	CS	CS0	CS1	
(E)	ENPWM0	ENPWM1	ENT00	ENT01	EXICSE0	EXICSE1
	EXICSE2	EXIF0	EXIF1	EXIF2	EXIISM0	EXIISM1
	EXIISM2	EXIMK0	EXIMK1	EXIMK2		
(M)	MS0					
(O)	OVF					
(R)	RFEN	RFLV	RXE			
(S)	SECSE	SEF	SEISM	SEMK	SRCSE	SRF
	SRISM	SRMK	STCSE	STF	STISM	STMK
(T)	TBCSE	TBF	TBMK	TMCSE0	TMCSE1	TMCSE2
	TMF0	TMF1	TMF2	TMISM0	TMISM1	TMISM2
	TMMK0	TMMK1	TMMK2	TS0	TS1	TSK
	TXE					

(g) レジスタ名

(A) A　　AX

(B) B　　BC

(C) C　　CY

(D) DE

(E) E

(H) H　　HL

(L) L

(R) R0　　R1　　R2　　R3　　R4　　R5

R6　　R7　　R8　　R9　　R10　　R11

R12　　R13　　R14　　R15　　RP0　　RP1

RP2　　RP3　　RP4　　RP5　　RP6　　RP7

RB0　　RB1　　RB2　　RB3　　RB4　　RB5

RB6　　RB7

(U) UP　　UPH　　UPL

(V) VP　　VPH　　VPL

(X) X

(h) セグメント属性

(A) AT

(C) CALLT0　　CALLT1

(F) FIXED

(S) SADDR　　SADDRP

(U) UNIT

４．７．４　逆アセンブラ仕様

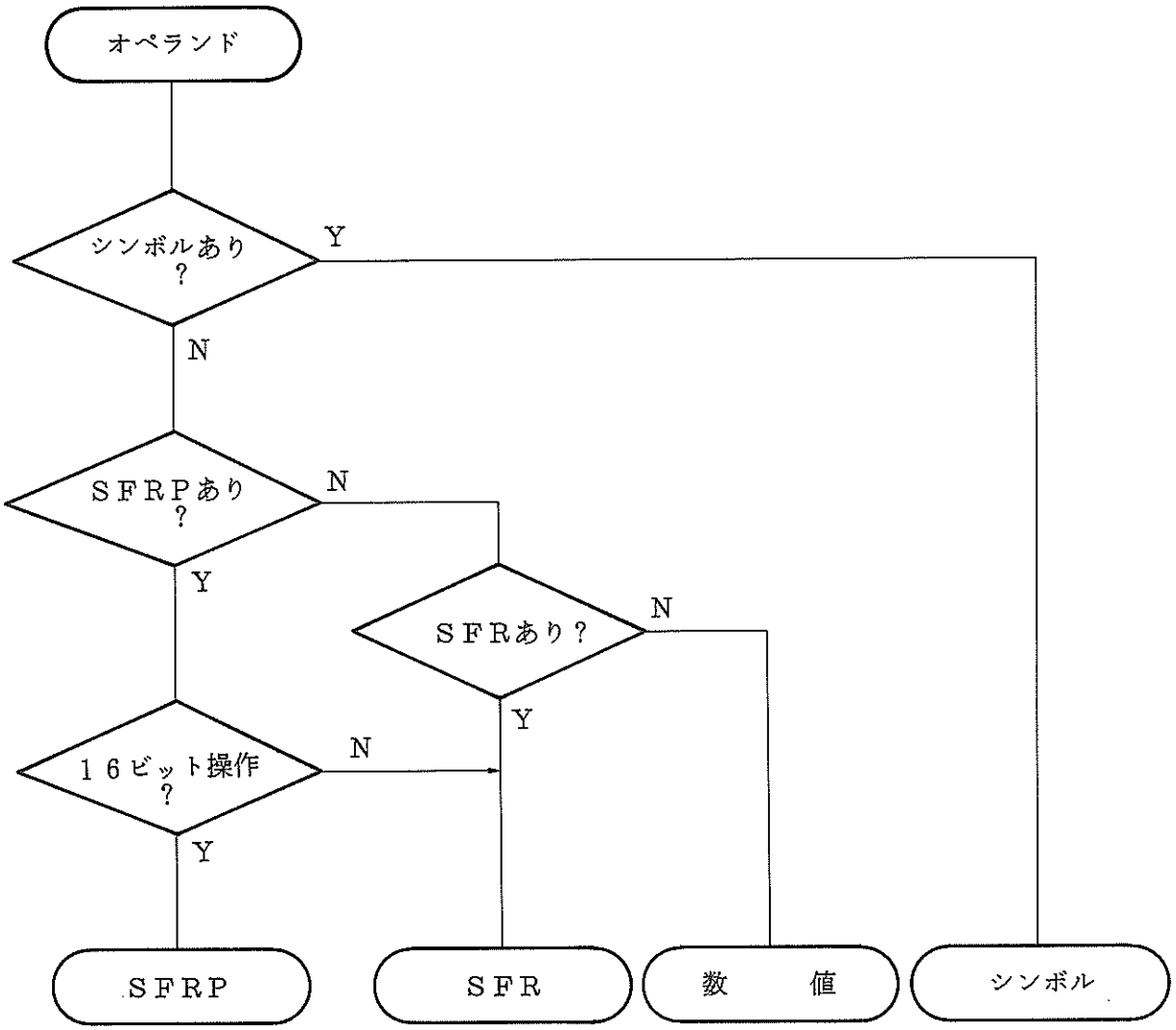
（１）疑似命令

逆アセンブル・リストの先頭に，ＯＲＧ命令を，最後にＥＮＤ命令を付加されます。

（２）オペランドの数値表示

オペランドの数値に対応するシンボルあるいはＳＦＲ，ＳＦＲＰがある場合，数値をシンボル，あるいは，ＳＦＲ，ＳＦＲＰに置換えて表示します。

数値に対して，シンボル，ＳＦＲ，ＳＦＲＰがすべて対応する場合，シンボルが表示されます。



数値に対応するシンボル，ＳＦＲ，ＳＦＲＰがない場合，１６進数で数値を表示されます。また，この場合，サフィックスHが付加されます。

数値の先頭文字は必ず数字（０～９）で始まります。

（３）オペランドのシンボル表示

オペランドのシンボル表示は，以下のようになります。

```
MOV    local symbol, A
MOV    public symbol , A
```

ローカル・シンボルであっても，モジュール名は表示されません。

（４）レーベル行の表示

カレント・ロケーションに対応するコード・シンボルあるいはデータ・シンボルがある場合，レーベル行として表示されます。

ローカル・シンボルの場合，　コロン一つ

```
module\local symbol:
```

と表示します。

パブリック・シンボルの場合，　コロン二つ

```
PUBLIC\public symbol ::
```

と表示します。

（５）ブランチ命令の表示

ブランチ系の命令では，同一のオブジェクト・コードに対して２種類のニモニックが割当てられている場合があります。このような場合，以下のように一般的に良く使われるニモニックが表示されます。

BNZ	BNZ	BZ	BZ
BNE		BE	
BNC	BNC	BC	BC
BNL		BL	
BNV	BNV	BV	BV
BPO		BPE	

（６）アドレス・チェック

ＳＦＲ空間およびｓａｄｄｒ空間に対し，アドレス・チェックを行ない，以下のような場合，警告が表示されます。

①　ＳＦＲ空間（ＦＦ００Ｈ～ＦＦＦＦＨ）

・当該アドレスにＳＦＲあるいはＳＦＲＰあるいは外部アクセス・エリア

（ＦＦＢ０Ｈ～ＦＦＢＦＨ）がない場合，

・Ｒ／ＯのＳＦＲに書込み動作を行なう場合，

・Ｗ／ＯのＳＦＲに読出し動作を行なう場合，

・ＳＦＲＰ以外の空間に対し１６ビット操作を行なう場合，

・外部アクセス・エリアに対し，１６ビット操作を行なった場合，

② ｓａｄｄｒ空間（ＦＥ２０Ｈ～ＦＥＦＦＨ）

・奇数アドレスに対し１６ビット操作を行なった場合，

（7）ＭＯＶ　ＳＴＢＣ，#byte　および　ＭＯＶ　ＷＤＭ，#byte命令（専用命令）

ＳＴＢＣ，ＷＤＭは，Ｒ／ＯのＳＦＲとして登録されています。

したがって，ＳＴＢＣ，ＷＤＭに対して書込み動作を行なうような，専用命令以外のＳＦＲ操作命令の場合，警告が表示されます。

また，専用命令において，オペランドのｉｍｍｅｄｉａｔｅデータのコンプリメントをとったものと，ｉｍｍｅｄｉａｔｅデータが一致しなかった場合，警告が表示されます。このとき，ニモニックのオペランドには，ｉｍｍｅｄｉａｔｅデータが表示されます。

（8）エラー表示

本逆アセンブラはエラー・コードとして，エラーと警告を表示します。

① エラー

逆アセンブルできない場合，ニモニックの所に？？？を表示します。

エラー時には必ず１バイト分をあけて，次から逆アセンブルは続行されます。

２バイト以上の長さをもつ命令コードで，２バイト目以降をデコードした結果，エラーであることがわかった場合でも，エラーとして表示するのは１バイト目だけであり，２バイト目を，命令コードの１バイト目としてあらためて，逆アセンブルをします。

例

```
Ａｄｄｒ　Ｏｂｊｅｃｔ          Ｍｎｅｍｏｎｉｃ
                                ＰＵＢＬＩＣ＼ＳＴＡＲＴ：：
 0100    0Ｂ　ＦＥ　７Ｆ　ＦＥ   ＭＯＶＷ    ＳＰ，#ＳＴＣＫ
 0104    ０１                    ？？？
 0105    ６７　００　１０        ＭＯＶＷ    ＲＰ７，#ＷＯＲＫ１
 0109    ５Ｆ                    ？？？
 010A    ５１                    ＭＯＶ      ［ＨＬ＋］，Ａ
```

② 警告

逆アセンブルできるが，正しい動作が望めない場合，逆アセンブルしたニモニックの直前に？を表示します。

例

Ａｄｄｒ	Ｏｂｊｅｃｔ	Ｍｎｅｍｏｎｉｃ
0100	0C 21 34 12	？　ＭＯＶＷ　　０ＦＥ２１Ｈ，＃１２３４Ｈ （奇数のsaddr に１６ビット操作を行なう。）
0104	3B 56 20	？　ＭＯＶ　　ＲＸＢ，＃２０Ｈ （Ｒ／ＯのＳＦＲに書込み動作を行なう。）
0107	0B B0 78 56	？　ＭＯＶＷ　　０ＦＦＢ０Ｈ，＃５６７８Ｈ （外部拡張エリアに16ビット操作を行なう。）

第５章　応用方法

５．１　概　　要

この章では，基本的なディバグ作業で用いる必要はないが，効率のよいディバグをするために知っておくと便利な機能（コマンド）について説明しています。

この章では，以下の機能（コマンド）について説明しています。各コマンドについては，「第４章　コマンドの説明」を読んでください。

５．２　コマンド入力時のテクニック

ＩＥ－７８３１０Ａ－Ｒに対してコマンドを入力する場合，通常であればコンソールからコマンドをキー入力します。しかし，これだけでは不便に感じることがあります。

たとえば，

（１）スタート・アップ時の環境設定（クロック設定，マッピング，プログラムのローディング，ブレーク・ポイントの設定　etc.・・・・）は，かなり多くのコマンドを入力しなければならない。しかも，スタート・アップ時には必ずしなければならない。

（２）コマンド入力時，１文字誤ったためにエラーとなったときは，また，新たにコマンドを最初からキー入力しなければならない。これが長いコマンドの場合は，煩わしい。

（３）以前に実行したコマンドを，もう一度，実行したいときでも，また，新たにコマンドを最初からキー入力しなければならない。これが長いコマンドの場合は，煩わしい。

（４）ディバグ中によく使うコマンドのキーワードを３文字も入力するのが煩わしい。

ＩＥ－７８３１０Ａ－Ｒでは，これらに対して，次のような機能（コマンド）で対処しています。

（１）ファイルからのコマンド入力(STRｺﾏﾝﾄﾞ)，コマンド・ファイル作成機能(COMｺﾏﾝﾄﾞ)

ある程度固定されている一連の手続き（コマンド）に関しては，テキストファイルからコマンドを読出し，実行します（ＳＴＲ　コマンド）。

また，わざわざエディタ等でテキスト・ファイルを作成しなくても，最初に，一連の手続きをキー入力しているときに，その手続きをテキスト・ファイルに登録す

ることができます（ＣＯＭ　コマンド）。

（２）行単位の編集機能

ＭＤ－０８０ＦＤ，ＭＤ－０８０ＦＤ－１０，ＭＤ－０８６ＦＤ－１０，ＭＤ－１１６ＦＤ－１０，ＭＤ－０８６ＨＤ－１０，ＭＤ－１１６ＨＤ－１０の本体コンソールおよび，ＭＤ－０８６ＦＤ－１０，ＭＤ－１１６ＦＤ－１０，ＭＤ－１１６ＦＤ－１０，ＭＤ－１１６ＨＤ－１０のサブコンソール上では，カーソル制御キーおよびアペンド・モード・キーを使用した行編集機能をサポートしています。

したがって，行中に誤りを見つけたときでも，カーソル制御キー，アペンド・モード・キーを使用して簡単に修正することができます（ＰＤＡ－８８０，ＭＤ－０８６ＦＤの拡張コンソールではサポートされません）。

（３）コマンド・ヒストリ機能（ＨＩＳ　コマンド）

キー入力，あるいは，ファイルから入力された，最新のコマンド行を２０行分だけ記憶しています。したがって，行番号だけを入力することにより，すでに入力されているコマンド行を呼出すことができます。呼出されたコマンド行は，（２）の行単位の編集機能を使用して修正・変更することができます。

（４）コマンドの省略機能

ディバグ時に，よく使うと思われるコマンドについては，３文字のキーワードをキー入力しなくても，１文字の省略形を入力するだけでよいようにしてあります。

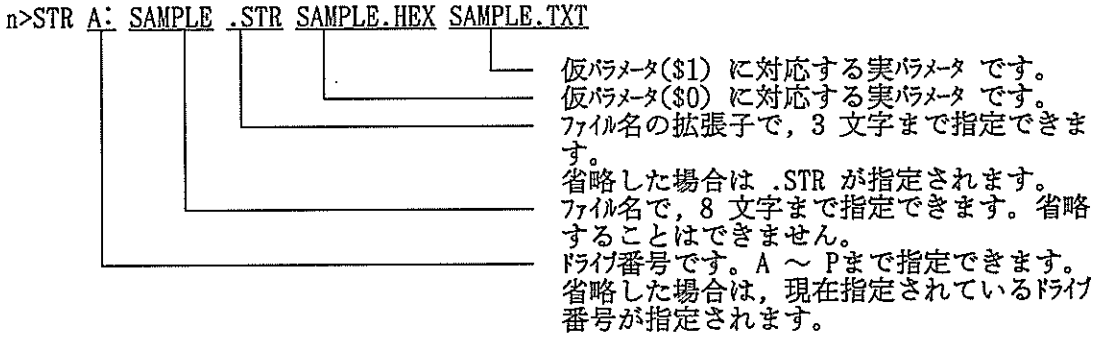
５．２．１　ファイルからのコマンド入力（ＳＴＲコマンド）

‘ＳＴＲ’コマンドを実行することにより，以降のコマンド入力を，‘ＳＴＲ’コマンドで指定したファイルより，自動的に行なうことができます。

‘ＳＴＲ’コマンドで指定するファイルには，‘ＣＯＭ’コマンドで作成したファイル，あるいは，エディタを用いて作成したファイルを用います。

エディタを用いて作成したファイルでは，ファイル内に仮パラメータ（＄０，＄１，＄２，＄３）を登録することにより，‘ＳＴＲ’コマンドのオペランドに指定したパラメータに，置換えることができます。仮パラメータは，４個まで指定することができます。

コマンド形式を次に示します。



仮ﾊﾟﾗﾒｰﾀ が指定されていて，実ﾊﾟﾗﾒｰﾀ が省略された場合は，その仮ﾊﾟﾗﾒｰﾀ は置換えられません。また，仮ﾊﾟﾗﾒｰﾀ が指定されていないのに実ﾊﾟﾗﾒｰﾀ が指定された場合は，実ﾊﾟﾗﾒｰﾀ は無視されます。

次に示す SAMPLE1.STRと，SAMPLE2.STR の二つのﾌｧｲﾙを使用した実行例を示します。

・SAMPLE1.STR の内容(ﾊﾟﾗﾒｰﾀの指定はありません。)

```
CLK U
RES
SUF H
MAP U 2XXX
MAP U 8XXX
MAP
LOD SAMPLE
```

・SAMPLE2.STR の内容(仮ﾊﾟﾗﾒｰﾀ $0, $1が指定されています。)

```
MEM D $0
RUN B $1
ESCｷｰ(1B H)
MEM D $0
```

・SAMPLE1.STR を実行しますと次のようになります。

```
n>STR SAMPLE1.STR ↵
n>CLK U ←　この行からｺﾏﾝﾄﾞがﾌｧｲﾙから入力されます。
n>RES
n>SUF H
n>MAP U 2XXX
n>MAP U 8XXX
n>MAP
  0000-1FFF R/O   2000-2FFF User  3000-7FFF Non   8000-8FFF User
  9000-FDFF Non
n>LOD SAMPLE
  object load complete
  symbol table loading
  PUBLIC      load complete
  SYM1        load complete
  SYM2        load complete
  SYM3        load complete
  SYM4        load complete
  SYM5        load complete
  SYM6        load complete
n>  ←　ここからは，ｺﾝｿｰﾙ からｺﾏﾝﾄﾞの入力ができます。
```

SAMPLE1.STR のようなﾌｧｲﾙを作成しておけば，ｼｽﾃﾑ･ﾘｾｯﾄ などの場合，簡単にｼｽﾃﾑの初期設定ができます。

次に，SAMPLE2.STR を実行しますと次のようになります。

```
n>STR SAMPLE2.STR 200X 100 ↵←SAMPLE2.STR は，ﾊﾟﾗﾒｰﾀ を指定します。
n>MEM D 200X  ←仮ﾊﾟﾗﾒｰﾀ($0) が 200X に置換えられます。
  2000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
n>RUN B 100   ←仮ﾊﾟﾗﾒｰﾀ($1) が100 に置換えられます。
  User-system Vcc-ON    Emulation start at 0100
  Standard break  terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  3000       1    0    0    0    0    0    0    0    0    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                           VP      UP      DE      HL
    07   82   00   00   00   00   00   00      0000    2000    2008    2007
  One step emulation standby← ESC(1BH) がﾌｧｲﾙから入力されます。
n>MEM D 200X  ←仮ﾊﾟﾗﾒｰﾀ($0) が 200X に置換えられます。
  2000   00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 00     ................
n>BRA A=200 ↵←ｺﾝｿｰﾙ からﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を200 番地に設定します。
n>STR SAMPLE2.STR 200X ↵←仮ﾊﾟﾗﾒｰﾀ($0) に対応する実ﾊﾟﾗﾒｰﾀ だけを設定します。
n>MEM D 200X
  2000   00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 00     ................
n>RUN B ←仮ﾊﾟﾗﾒｰﾀ($1) に対応する実ﾊﾟﾗﾒｰﾀ が指定されていませんので置換えられません。
  User-system Vcc-ON    Emulation start at 0106
                        ｴﾐｭﾚｰｼｮﾝの開始ｱﾄﾞﾚｽは，現在のﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀが指定されます。
  Standard break  terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0202  2FFA       1    0    0    0    0    1    0    1    0    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                           VP      UP      DE      HL
    07   00   01   20   00   00   00   00      0000    2100    2108    2107
  One step emulation standby← ESC(1BH) がﾌｧｲﾙから入力されます。
n>MEM D 200X  ←仮ﾊﾟﾗﾒｰﾀ($0) が 200X に置換えられます。
  2000   01 00 00 00 00 00 00 82 00 00 00 00 00 00 00 00     ................
n>
```

SAMPLE2.STR のようなﾌｧｲﾙを作成しておけば，ｴﾐｭﾚｰｼｮﾝなどを繰返し使用するような場合に，ﾊﾟﾗﾒｰﾀ を変更するだけで一連の処理ができます。

・ｺﾏﾝﾄﾞで指定したﾌｧｲﾙが存在しない場合は，次のようなﾒｯｾｰｼﾞ を表示します。

```
n>STR SAMPLE.STR↵
file not found  ←SAMPLE.STRというﾌｧｲﾙが見つからないのでﾒｯｾｰｼﾞ が表示されます。
n>
```

５．２．２　コマンド・ファイル作成機能（ＣＯＭコマンド）

ＣＯＭコマンドは，コンソールから入力されたコマンド，あるいは，データをファイル，または，リスト装置に出力するために，ファイル，または，リスト装置をオープンするためのコマンドです。このコマンドによって作成されたファイルは，ＳＴＲコマンドにおいて使用することができます。

オープンされた装置への出力の開始／終了は，↑Ｏキーの入力により制御されます。オープンできる装置には次の種類があります。

・ファイル　……　ファイルを出力装置としてオープンします。
・ＬＳＴ：　……　リスト装置を出力装置としてオープンします。
・ＣＯＮ：　……　コンソールを出力装置としてオープンします。つまり，オープンされている出力装置はクローズされます。

実行例を出力装置ごとに示します。

（１）出力装置を指定しない場合は，現在指定されている出力装置が表示されます。

```
n>COM ↵
  CON:←COMｺﾏﾝﾄﾞ で出力装置を指定していない場合や，出力装置をｸﾛｰｽﾞした場合に表示さ
n>       れます。

n>COM ↵
  LST:← ﾘｽﾄ装置が指定されている場合に表示されます。
n>

n>COM ↵
  SAMPLE.STR←ﾌｧｲﾙが指定されている場合は，そのﾌｧｲﾙ名が表示されます。
n>
```

（２）出力装置を指定した場合の実行例を示します。

```
n>COM A: SAMPLE .STR ↵ ←ﾌｧｲﾙを指定する場合の，一般形式です。
```

ﾌｧｲﾙ名の拡張子で，３ 文字まで指定できます。省略した場合は .STR が指定されます。
ﾌｧｲﾙ名です。８ 文字まで指定できます。省略できません。
ﾄﾞﾗｲﾌﾞ番号です。A ～ Pまで指定できます。省略した場合は，現在指定されているﾄﾞﾗｲﾌﾞ番号が指定されます。

```
n>COM SAMPLE.STR ↵ ←SAMPLE.STRというﾌｧｲﾙ名でﾌｧｲﾙをｵｰﾌﾟﾝします。
n>↑O (ｴｺｰ･ﾊﾞｯｸはありません←これ以降ｺﾝｿｰﾙ から入力されるｺﾏﾝﾄﾞ，および，ﾃﾞｰﾀ は，
                              SAMPLE.STRにも出力されます。

n>COM SAMPLE1.STR ↵ ←すでに存在するSAMPLE1.STR をもう一度ｵｰﾌﾟﾝします。
  File already exists.Delete ? (Y or N):■←すでに存在するﾌｧｲﾙを削除するかどうか
                          ｶｰｿﾙ位置 ┘    のﾒｯｾｰｼﾞ が表示されます。

  File already exists.Delete ? (Y or N): Y ↵
n>
```

すでに存在するﾌｧｲﾙを削除する場合は，Y を入力します。この場合は，SAMPLE1.STR が削除され，新たにSAMPLE1.STR がｵｰﾌﾟﾝされます。

```
  File already exists.Delete ? (Y or N): N↵
n>
```

N を入力しますと，すでに存在するﾌｧｲﾙは削除されません。この場合は，このｺﾏﾝﾄﾞが無効になります。

（３）ﾌｧｲﾙを指定する場合は，指定したﾌｧｲﾙがすでに存在し，そのﾌｧｲﾙの属性がR/O 属性，あるいは，SYS 属性である場合は，そのﾌｧｲﾙをｵｰﾌﾟﾝすることはできません。

```
n>COM SAMPLE.SYS↵←指定したﾌｧｲﾙが R/O属性，あるいは，SYS 属性のﾌｧｲﾙをｵｰﾌﾟﾝします。
  File already exists.←指定したﾌｧｲﾙがｵｰﾌﾟﾝできないのでﾒｯｾｰｼﾞ が表示されます。
n>                                ｺﾏﾝﾄﾞは，無効になります。
```

（４）ﾘｽﾄ 装置が他の処理によって使用されている場合は，ｵｰﾌﾟﾝすることはできません。

```
n>COM LST:↵←ﾘｽﾄ 装置をｵｰﾌﾟﾝします。必ず LST: まで入力してください。LST だけですとLST.TXT というﾌｧｲﾙが指定されたことになります。
n>↑O (ｴｺｰ･ﾊﾞｯｸはありません)
```

ｵｰﾌﾟﾝされたﾘｽﾄ 装置にｺﾝｿｰﾙ から入力されたｺﾏﾝﾄﾞ，および，ﾃﾞｰﾀ を出力するには，↑O ｷｰ を入力します。↑O ｷｰが入力されて以降にｺﾝｿｰﾙ から入力されたｺﾏﾝﾄﾞ，および，ﾃﾞｰﾀ は，ﾘｽﾄ 装置にも出力されます。

```
n>COM LST:↵
  List device is used by other process. ←ﾘｽﾄ 装置が他の処理によって使用されている場合のﾒｯｾｰｼﾞ です。この場合ｺﾏﾝﾄﾞは，無効になります。
n>
```

（５）ｺﾝｿｰﾙ を出力装置に指定しますと，現在ｵｰﾌﾟﾝされている出力装置はｸﾛｰｽﾞされます。

```
n>COM CON:↵←出力装置がｸﾛｰｽﾞされます。これ以降に入力されたｺﾏﾝﾄﾞ，および，ﾃﾞｰﾀ は出
n>          力装置には出力されません。
```

５．２．３　行単位の編集機能

システム・ソフト使用時，ＭＤ－０８０ＦＤ，ＭＤ－０８０ＦＤ－１０，ＭＤ－０８６ＦＤ，ＭＤ－０８６ＦＤ－１０，ＭＤ－０８６ＨＤ－１０，ＭＤ－１１６ＦＤ－１０，ＭＤ－１１６ＨＤ－１０の本体コンソールおよび，ＭＤ－０８６ＦＤ，ＭＤ－０８６ＦＤ－１０，ＭＤ－０８６ＨＤ－１０，ＭＤ－１１６ＦＤ－１０，ＭＤ－１１６ＨＤ－１０のサブ・コンソール上においては，カーソル制御キーを使用し，コマンド行を編集することができます。

カーソルの移動は，←，→キーを使用し，コマンド行の最初から最後までの範囲内で，自由に動かすことができます。

任意の位置へカーソルを移動し，コンソールより，新たなキャラクタを入力することにより，その位置のキャラクタを変更することができます。

例1

```
n>LOD B:SAMPLE
              └―― カーソル位置
```

‘Ｂ：’を‘Ａ：’に変更したい場合は，←キーを押し，カーソルを‘Ｂ’へ移動します。

```
n>LOD B:SAMPLE
       └―――― カーソル位置
```

この状態のとき，コンソールより‘A’を入力しますと，次のようになります。

```
n>LOD A:SAMPLE
       └―――― カーソル位置
```

このコマンドを，オブジェクトのみの指定とするため，→キーを押し，カーソルをコマンドの最後へ移動させます。

```
n>LOD A:SAMPLE
              └―― カーソル位置
```

ここで，スペースと‘C’を入力すると次のようになります。

```
n>LOD A:SAMPLE C
                └―― カーソル位置
```

この状態で，‘↵’（リターン・キー）を入力しますとコマンドが実行されます。なお，‘↵’は，カーソル位置が，行のどこにあってもかまいません。つまり，わざわざカーソルを行端にまで移動する必要はありません。

```
n>LOD A:SAMPLE C
       └──────────  カーソル位置

n>LOD A:SAMPLE C
            └─────  カーソル位置
```

このように，カーソル位置が違っていてもコマンドは実行されます。

また，編集機能には，キャラクタを追加できるインサート・モードがあります。インサート・モードにするためには，‘↑A’を入力します。

‘↑A’を入力しますとカーソル位置に‘<’が表示され，インサート・モードであることを示します。‘<’が表示された位置からキャラクタが追加されていきます。インサート・モードを終了するときは，もう一度‘↑A’を入力します。このとき表示されていた‘<’が消えて通常のコマンド入力中になります。

例2

```
n>LOD SAMPLE C
            └─────  カーソル位置
```

このコマンドを“ＬＯＤ　Ｂ：ＳＡＭＰＬＥ　Ｃ”に変更するには，まず　←キーを使用して‘S’の位置までカーソルを移動させます。

```
n>LOD SAMPLE C
       └──────────  カーソル位置
```

ここで‘↑A’キーを入力し，インサート・モードにします。

```
n>LOD <AMPLE C
       └──────────  カーソル位置（インサート・モードであることの表示）
```

そして，‘B’，‘:’を入力します。

```
n>LOD B:<AMPLE C
         └────────  カーソル位置
```

これで変更が終了しましたので，通常のコマンド入力に戻すためにもう一度‘↑A’を入力します。

```
n>LOD B:SAMPLE C

         └────────  カーソル位置
```

なお，行入力を終了するときは，特にインサート・モードを終了させる必要はありません。そのまま‘↵’を入力することでコマンドが実行されます。

インサート・モードは，コマンドが実行されますと自動的に解除されます。

５．２．４　コマンド・ヒストリ機能（ＨＩＳコマンド）

システム・ソフト使用時には，キー入力したコマンドの最新の２０行を記憶しています。この記憶しているコマンド行のうちから任意の１行を呼出して実行することができます。

記憶しているコマンドを呼出すには，コマンドを入力するときに‘！ｎ↵’と入力します（ｎは，１～２０までの行番号です）。‘ＨＩＳ↵’と入力することにより，記憶しているコマンド行を知ることができます。また，最新のコマンドを呼出すためには，‘！！↵’と入力します。

例1

```
n>HIS ↵
   1 MAP
   2 MAP W 0,1FFF
   3 MAP R 2000,0FDFF
   4 LOD TEST C
         .
         .
         .
  19 MEM D 1XXX
  20 HIS
n>
```

４番目のコマンドを呼出す場合は，

```
n>!4 ↵
  LOD TEST C
```

最新のコマンドを呼出す場合は，

```
n>!! ↵
  HIS
```

となります。

ただし，ＰＤＡ－８８０とMD－０８０／０８６／１１６では，コマンド表示後のカーソルの位置が違います。

例2

・ＰＤＡ－８８０の場合

```
n>!4 ↵
  LOD TEST C
             └──  カーソル位置
```

・MD－０８０／０８６／１１６の場合

```
n>!4 ↵
  LOD TEST C
  └──────────  カーソル位置
```

HIS コマンドは，ヒストリ・メモリに登録されているコマンドを表示します。

ヒストリ・メモリへのコマンドの登録は，コマンドを実行することにより自動的にされます。

ヒストリ・メモリへ登録できるコマンドは，最新の20行のコマンドだけです。

（１）コマンド形式を示します。

n>HIS ← オペランド の指定はできません。

（２）表示形式を示します。

```
n>HIS ↵
   1 MAP
   2 LOD TEST
          .
          .
          .
  19 MEM D
  20 HIS
n>
```

ヒストリ・メモリに登録されているコマンドです。

登録されているコマンドに付けられたコマンド番号です。

（３）実行例を示し説明します。

```
n>HIS ↵
   1 MAP
   2 MAP W 0,0FDFF
   3 MAP
   4 LOD TEST C
   5 HIS
n>LOD TEST1 C ↵
  object load complete
n>HIS ↵
   1 MAP
   2 MAP W 0,0FDFF
   3 MAP
   4 LOD TEST C
   5 HIS
   6 LOD TEST1 C
   7 HIS
n>
```

ヒストリ・メモリに登録されているコマンドを表示させます。

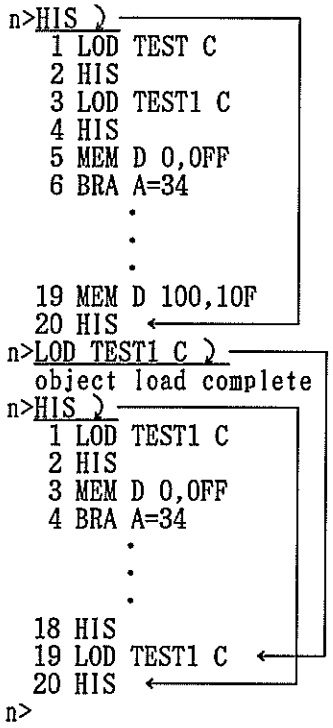
ヒストリ・メモリに登録されていたコマンドです。

この表示をするために入力したコマンドが最新のコマンドとして表示されます。

前のHISコマンド を実行後に入力したコマンドが追加されています。

このようにコマンドが20行をこえるまでは，実行したヒストリ・メモリにコマンドは追加されます。

（４）次に，ｺﾏﾝﾄﾞが20行以上になった場合について説明します。



前のHIS ｺﾏﾝﾄﾞでは，３番目に表示されたｺﾏﾝﾄﾞです。

前の HIS ｺﾏﾝﾄﾞ です。

今回の HIS ｺﾏﾝﾄﾞ です。

５．２．５　コマンド省略形入力

システム・モード時には，次に示すコマンドについては，コマンドのキーワードを１文字だけで入力することもできます。

コマンド	省略形	コマンド種類
ASM	A	アセンブラ・コマンド
CLK	C	クロック・コマンド
DAS	D	逆アセンブラ・コマンド
EXT	E	システム・モード終了コマンド
HLP	H	ヘルプ・コマンド
LOD	L	オブジェクト／シンボル・ロード・コマンド
MEM	M	メモリ操作コマンド
RUN	R	エミュレーション・コマンド
SAV	S	オブジェクト・セーブ・コマンド
TRD	T	トレース表示コマンド
VRY	V	ベリファイ・コマンド

これらのコマンドは，省略形で入力された場合は，コマンドのターミネータ（スペース，または，リターン）が入力されたときにコマンドを表示します。

例

```
n>A ↵                    ←アセンブラ・コマンドを省略形で入力します。
```

実際には，次のように表示されコマンドが実行されます。

```
n>ASM
  0000                    MOV     R0,R2
                  =
                  └──  カーソル位置
```

```
n>A                       ←‘A’，スペースと省略形で入力します。
   └─────  カーソル位置
```

実際には，スペースを入力したときに，次のような表示になります。

```
n>ASM                     ←アセンブラ開始アドレスの入力待ちになります。
   └──  カーソル位置
```

このあとは，通常のコマンドの入力と同じです。

５．３　シンボリック・ディバグ時のテクニック

５．３．１　モジュール名指定によるシンボル・ロード

シンボル・テーブル・ファイルをロードする場合は，モジュール名を指定することにより必要なモジュールだけのローカル・シンボルをロードすることができます。

このため，多数のモジュールが存在するような場合でも，必要なモジュールだけを指定してロードすることにより短時間でロードができます。

モジュールの指定は，コマンド入力においてファイル名を指定したあとにモジュール名を指定します。モジュール名を指定する場合は，モジュール名の直後に‘＼’（バック・スラッシュ）を必ず入力してください。また，複数のモジュール名を指定する場合は，モジュール名とモジュール名を‘，’（カンマ）で区切って指定します。一度に指定できるモジュール名の数は，コマンド１行の最大数（１２８文字）以内であれば，いくつでも指定できます。

パブリック・シンボルだけをロードする場合は‘ＰＵＢＬＩＣ＼’を指定してください。

例1

```
n>LOD TEST.SYM SYM1＼ ,SYM2＼ ,SYM3＼ S）
                 │      └── モジュール名の区切りです。
                 └──────── モジュール名です。
```

この場合は，TEST.SYMというシンボル・テーブル・ファイルからSYM1，SYM2，SYM3というモジュール内で定義されたローカル・シンボルだけをロードします。

複数の同じモジュール名が指定されている場合は，次のような警告メッセージが表示されます。この場合は，１回だけ指定されたモジュールのシンボルをロードします。

```
Warning double define : XXXXXXXX
                        └──── 複数指定されたﾓｼﾞｭｰﾙ 名
```

例2

```
n>LOD TEST.SYM  SYM1\, SYM2\, SYM1\ S↵
  Warning double define : SYM1
  symbol table loading
  PUBLIC      pass
  SYM1        load complete
  SYM2        load complete
  SYM3        pass
  SYM4        pass
  SYM5        pass
n>
```

指定されたモジュール名が，シンボル・テーブル・ファイルに存在しない場合は，次のようなメッセージを表示します。

```
XXXXXXXX not found module record
└──── 存在しないﾓｼﾞｭｰﾙ 名
```

例3

```
n>LOD TEST.SYM  SYM4\, SYM5\, SYM6\ S↵
  symbol table loading
  PUBLIC      pass
  SYM1        pass
  SYM2        pass
  SYM3        pass
  SYM4        load complete
  SYM5        load complete
  SYM6        not found module record
n>
```

全パブリック・シンボルと指定したモジュールのローカル・シンボルだけをロードする場合は，次のようにします。

例4この場合，全パブリック・シンボルと，ＳＹＭ４，ＳＹＭ５のローカル・シンボルだけをロードします。

```
n>LOD TEST.SYM PUBLIC \, SYM4\, SYM5\ S↵
  symbol table loading
  PUBLIC      load complete
  SYM1        pass
  SYM2        pass
  SYM3        pass
  SYM4        load complete
  SYM5        load complete
n>
```

また，すでにロードされているモジュールをロードした場合は，次のようなメッセージを表示します。この場合は，表示されたモジュールは無視されます。

```
XXXXXXXX    loaded module
└──── ロード済みの モジュール名
```

例5

```
n>LOD TEST.SYM PUBLIC \, SYM4\, SYM5\ S )
  symbol table loading
  PUBLIC      loaded module
  SYM1        pass
  SYM2        pass
  SYM3        pass
  SYM4        load complete
  SYM5        load complete
n>
```

５．４　その他

次のような場合に使用する，コマンドについて説明します。なお，これらのコマンドは，システム・ソフト使用時の場合のみ有効です。

・ＨＬＰコマンド

コマンドの使用方法を知りたい場合

・ＬＳＴコマンド

実行結果を記録にして残したい場合

・ＤＩＲコマンド

ディスクのファイル・ディレクトリを参照したい場合

５．４．１　オンライン・ヘルプ機能（ＨＬＰコマンド）

ＨＬＰコマンドは，ＩＥ－７８３１０Ａ－Ｒで使用できるコマンドの一覧，および，コマンドの使用方法を表示します。

コマンド形式を次に示します。

n>HLP　　（オペランドを指定しない場合）

n>HLP ASM
　　└── ＩＥ－７８３１０Ａ－Ｒで使用できるコマンドを指定します。

実行例を以下に示します。

```
n>HLP ↵ ←ｵﾍﾟﾗﾝﾄﾞ にｺﾏﾝﾄﾞを指定しない場合は，ｺﾏﾝﾄﾞ一覧を表示します。
    Command :
  ASM          BRA          BRD          BRE          BRT
  BRM          BRO          BR1          BR2          BR3
  CLK          COM          DAS          DIG          DIR
  EXT          HIS          HLP          LOD          LST
  MAP          MAT          MDR          MEM          MOD
  MOV          PGM          REG          RUN          RES
  SAV          SPR          STR          SUF          SYM
  TRM          TRX          TRQ          TRS          TRP
  TRD          VRY
HLP>←HLPｺﾏﾝﾄﾞ 実行中のﾌﾟﾛﾝﾌﾟﾄ を表示します。この状態でｺﾏﾝﾄﾞを入力しますと，そのｺﾏﾝﾄﾞ
      の説明が表示されます。
HLP>ASM ↵ ←ASMｺﾏﾝﾄﾞ の説明を表示します。HLPｺﾏﾝﾄﾞ では省略形の入力はできません。
      ┌
      │ ASM ｺﾏﾝﾄﾞの説明が表示されます。
      └
HLP>↵ ←HLPｺﾏﾝﾄﾞ を終了します。
n>
```

n>HLP ASM ↵ ←ｵﾍﾟﾗﾝﾄﾞ にｺﾏﾝﾄﾞを指定した場合は，そのｺﾏﾝﾄﾞの説明が表示されます。

ASM ｺﾏﾝﾄﾞの説明が表示されます。

HLP>←HLPｺﾏﾝﾄﾞ 実行中のﾌﾟﾛﾝﾌﾟﾄ を表示します。これ以降は，ｵﾍﾟﾗﾝﾄﾞ なしの場合と同じです。

・IE-78310A-R で使用できないｺﾏﾝﾄﾞを指定しますと，ｴﾗｰ・ﾒｯｾｰｼﾞ を表示して正しいｺﾏﾝﾄﾞの入力を待ちます。

```
n>HLP ABC ↵
  Keyword Error
HLP>ABC ↵
  Keyword Error
HLP>
```

５．４．２　ファイルへの結果出力（ＬＳＴコマンド）

ＬＳＴコマンドは，コンソールに出力されるキャラクタをファイル，あるいは，リスト装置に出力するために，ファイル，あるいは，リスト装置をオープンするためのコマンドです。

オープンされた装置への出力の開始／終了は，↑Ｒキーの入力により制御されます。　★

オープンできる装置には次の種類があります。

・ファイル　……　ファイルを出力装置としてオープンします。
・ＬＳＴ：　……　リスト装置を出力装置としてオープンします。
・ＣＯＮ：　……　コンソールを出力装置としてオープンします。つまり，オープンされている出力装置はクローズされます。

実行例を出力装置ごとに示します。

（１）出力装置を指定しない場合は，現在指定されている出力装置が表示されます。

```
n>LST ↵
  CON:←LSTｺﾏﾝﾄﾞ で出力装置を指定していない場合や，出力装置をｸﾛｰｽﾞした場合に表示さ
n>      れます。

n>LST ↵
  LST:←ﾘｽﾄ 装置が指定されている場合に表示されます。
n>

n>LST ↵
  SAMPLE.TXT←ﾌｧｲﾙが指定されている場合は，そのﾌｧｲﾙ名が表示されます。
n>
```

（２）出力装置を指定した場合の実行例を示します。

```
n>LST A: SAMPLE .LST←ﾌｧｲﾙを指定する場合の，一般形式です。
```

ﾌｧｲﾙ名の拡張子で，3 文字まで指定できます。省略した場合は .TXT が指定されます。
ﾌｧｲﾙ名です。8 文字まで指定できます。省略できません。
ﾄﾞﾗｲﾌﾞ番号です。A ～ Pまで指定できます。省略した場合は，現在指定されているﾄﾞﾗｲﾌﾞ番号が指定されます。

```
n>LST SAMPLE.TXT ↵←SAMPLE.TXTというﾌｧｲﾙ名でﾌｧｲﾙをｵｰﾌﾟﾝします。
n>↑R (ｴｺｰﾊﾞｯｸ はありません←これ以降ｺﾝｿｰﾙ に表示されるｷｬﾗｸﾀ は，SAMPLE.TXTにも　★
                出力されます。

n>LST SAMPLE.LST ↵←すでに存在するSAMPLE.LSTをもう一度ｵｰﾌﾟﾝします。
  File already exists.Delete ? (Y or N):■←すでに存在するﾌｧｲﾙを削除するかどうか
                ｶｰｿﾙ位置 ──┘    のﾒｯｾｰｼﾞ が表示されます。

  File already exists.Delete ? (Y or N): Y ↵
n>
```

すでに存在するﾌｧｲﾙを削除する場合は，Y を入力します。この場合は，SAMPLE.LSTが削除され，新たにSAMPLE.LSTがｵｰﾌﾟﾝされます。

```
  File already exists.Delete ? (Y or N): N↵
n>
```

N を入力しますと，すでに存在するﾌｧｲﾙは削除されません。この場合は，このｺﾏﾝﾄﾞが無効になります。

（3）ﾌｧｲﾙを指定する場合は，指定したﾌｧｲﾙがすでに存在し，そのﾌｧｲﾙの属性がR/O 属性，あるいは，SYS 属性である場合は，そのﾌｧｲﾙをｵｰﾌﾟﾝすることはできません。

```
n>LST SAMPLE.SYS↵
  File already exists.
n>
```

←指定したﾌｧｲﾙが R/O属性，あるいは，SYS 属性のﾌｧｲﾙをｵｰﾌﾟﾝします。

←指定したﾌｧｲﾙがｵｰﾌﾟﾝできないのでﾒｯｾｰｼﾞ が表示されます。ｺﾏﾝﾄﾞは，無効になります。

（4）ﾘｽﾄ 装置を指定する場合は，ﾘｽﾄ 装置が他の処理によって使用されている場合は，ｵｰﾌﾟﾝすることはできません。

```
n>LST LST:↵
```

←ﾘｽﾄ 装置をｵｰﾌﾟﾝします。必ず LST: まで入力してください。LST だけですとLST.TXT というﾌｧｲﾙが指定されたことになります。

★ n>↑R (ｴｺｰ･ﾊﾞｯｸはありません)

★ ｵｰﾌﾟﾝされたﾘｽﾄ 装置にｺﾝｿｰﾙ に表示されたｷｬﾗｸﾀ を出力するには，↑R ｷｰを入力
★ します。↑R ｷｰが入力された以降のｺﾝｿｰﾙ に表示されたｷｬﾗｸﾀ は，ﾘｽﾄ 装置にも出力されます。

```
n>LST LST:↵
  List device is used by other process.
n>
```

←ﾘｽﾄ 装置が他の処理によって使用されている場合のﾒｯｾｰｼﾞ です。この場合ｺﾏﾝﾄﾞは，無効になります。

（5）ｺﾝｿｰﾙ を出力装置に指定しますと，現在ｵｰﾌﾟﾝされている出力装置はｸﾛｰｽﾞされます。

```
n>LST CON:↵
n>
```

←出力装置がｸﾛｰｽﾞされます。これ以降にｺﾝｿｰﾙ に表示されたｷｬﾗｸﾀ は，出力装置には出力されません。

５．４．３　ディレクトリ表示機能（ＤＩＲコマンド）

ＤＩＲコマンドは，ファイル・ディレクトリを参照するためのコマンドです。指定されたドライブのディレクトリ，および，特定のファイル名のディレクトリを表示することができます。また，ファイル名にはワイルド・キャラクタ（＊，？）を使用することができます。

実行例を以下に示します。

```
n>DIR ↵
  A: CONVPM   CMD : ABORT    CMD : ATTACH   CMD : DDT      COM
  A: CONSOLE  CMD : DDT86    CMD : DIR      CMD : DSKRESET CMD
  A: ERA      CMD : ERAQ     CMD : FORMAT   CMD : BACKUP   CMD
  A: MPMSTAT  CMD : PIP      CMD : REN      CMD : SDIR     CMD
  A: OBJERR   HEX : STAT     CMD : SUBMIT   CMD : TOD      CMD
  A: TYPE     CMD : UNLOCKB  CMD : IE78310  CMD : IE78310  COM
  A: ERROR2   STR : WMA      CMD : SET      CMD : GENCMD   CMD
  A: WM       COM : WM       HLP : MPM      SYS : MPM      COM
  A: KEY      COM : KEY      KEY : STAT     COM : CPM      CMD
  A: WM       CMD : ZSID     COM : DUMP     COM : M80      COM
  A: L80      COM : LOAD     COM : PIP      COM : DUMPA    COM
  A: CREF80   COM : SYSCPY   CMD : SYSCPY   COM : AFF      HEX
  A: ERROR    COM : SUBMIT   COM : XSUB     COM : STR      HEX
  A: LIB80    COM : LH3FF    HEX : SETUP    STR : LH1FF    HEX
  A: SORT     HEX : DIRMPM   CMD : IE78310  HLP : IE78310  OV1
  A: IE78310  OV2 : SORT     SRC : SORT     STR
n>DIR *.HEX ↵      ( 拡張子が HEXというﾌｧｲﾙ名のﾃｨﾚｸﾄﾘを表示します。)
  A: OBJERR   HEX : AFF      HEX : STR      HEX : LH3FF    HEX
  A: LH1FF    HEX : SORT     HEX
n>DIR IE78310.* ↵  ( ﾌｧｲﾙ名が IE78310というﾃﾞｨﾚｸﾄﾘを表示します。)
  A: IE78310  CMD : IE78310  COM : IE78310  HLP : IE78310  OV1
  A: IE78310  OV2
n>DIR I???????.* ↵  ( ﾌｧｲﾙ名が Iで始まるﾃｨﾚｸﾄﾘをすべて表示します。)
  A: IE78310  CMD : IE78310  COM : IE78310  HLP : IE78310  OV1
  A: IE78310  OV2
n>DIR *.H?? ↵      ( 拡張子が Hで始まるﾌｧｲﾙ名のﾃｨﾚｸﾄﾘをすべて表示します。)
  A: OBJERR   HEX : WM       HLP : AFF      HEX : STR      HEX
  A: LH3FF    HEX : LH1FF    HEX : SORT     HEX : IE78310  HLP
n>
```

付　録

(1) 設置方法の概要

最初に付属品をＩＥ－７８３１０Ａ－Ｒ本体に接続します。

・ＩＥ－７８３１０Ａ－Ｒ本体にターゲット・プローブを接続する場合，ＩＥ－７８３１０Ａ－Ｒ本体の右側面からエミュレーション・ボード（上から２番目のボード）を抜きエミュレーション・ボード上のコネクタ（ＣＮ１，ＣＮ２）にターゲット・プローブのコネクタを差します。

・ＩＥ－７８３１０Ａ－Ｒ本体に外部センス・クリップを接続する場合，ＩＥ－７８３１０Ａ－Ｒ本体の右側面からブレーク・ボード（上から３番目のボード）を抜きブレーク・ボード上のコネクタ（ＣＮ１）に外部センス・クリップのコネクタを差します。

・ＩＥ－７８３１０Ａ－Ｒ本体に電源ケーブルを接続する場合，ＩＥ－７８３１０Ａ－Ｒ本体の裏側のＡＣインレットに差込みます。

・ＩＥ－７８３１０Ａ－Ｒ本体にＲＳ－２３２－Ｃインタフェース・ケーブルを接続する場合，ＩＥ－７８３１０Ａ－Ｒ本体の正面パネルのＣＨ１，あるいはＣＨ２に差します。

付属品の接続が終わったなら設置場所に設置します。

設置場所は，ゴミやチリ等の少ない場所に設置します。

また，空気取り入れ口付近には障害物をおかないようにしてください。

（2）接続可能な周辺装置

ホスト・マシン　…　MD－１１６HD－１０，MD－１１６FD－１０
MD－０８６HD－１０，MD－０８６FD－１０
MD－０８６HD，MD－０８６FD
PDA－８８０
★　PC－９８００シリーズ，IBM PC／AT
PROMプログラマ …　PG－１５００，PG－２０００
ターミナル　…　MD－９１０TM

（3）筐体に付いているスイッチ機能と設定

・電源スイッチ　　・・・・　パワー表示ＬＥＤ付きのプッシュ・スイッチを使っておりＩＥ－７８３１０Ａ－Ｒの電源をＯＮ／ＯＦＦします。

・リセット・スイッチ　・・・・　プッシュ・スイッチを押すことによりＩＥ－７８３１０にリセットがかかります。

・ターミナル／モデム・モード切替えスイッチ

・・・・　ＲＳ－２３２－Ｃインタフェースのターミナル・モードとモデム・モードとをこのスライド・スイッチにより切替えています。

・ＲＴＳの設定スイッチ

・・・・　ＲＳ－２３２－ＣインタフェースのＲＴＳのピン番号を切替えるＤＩＰスイッチです。通常はスイッチ番号の１をＯＮ，２－３をＯＦＦに設定しておきます。

・フレーム・グラウンドの設定スイッチ

・・・・　ＲＳ－２３２－Ｃインタフェースのフレーム・グラウンドとシグナル・グラウンドを共通にするか，オープンにするかのＤＩＰスイッチです。
通常はオープンに設定しておきます。スイッチ番号の４をＯＦＦにしておきます。

・ボーレート切替えスイッチ

・・・・　ＲＳ－２３２－Ｃインタフェースのチャネル１用のボーレートの設定用マイクロＤＩＰスイッチです。

（4）コントロール／トレース・ボードのジャンパの設定

コントロール／トレース・ボードの各種ジャンパは，出荷時の状態でお使いください。出荷時以外の設定をしますと，正常に動作しません。

表付－1　コントロール／トレース・ボードの出荷時のジャンパ設定

ｼﾞｬﾝﾊﾟ NO.	設　定
ＪＰ　2	1－6　ショート
ＪＰ　3	1－6　ショート
ＪＰ　4	オープン
ＪＰ　5	1－6　オープン※
	2－5　オープン※
	3－4　ショート
ＪＰ　6	1－2　ショート
ＪＰ　7	1－2　ショート

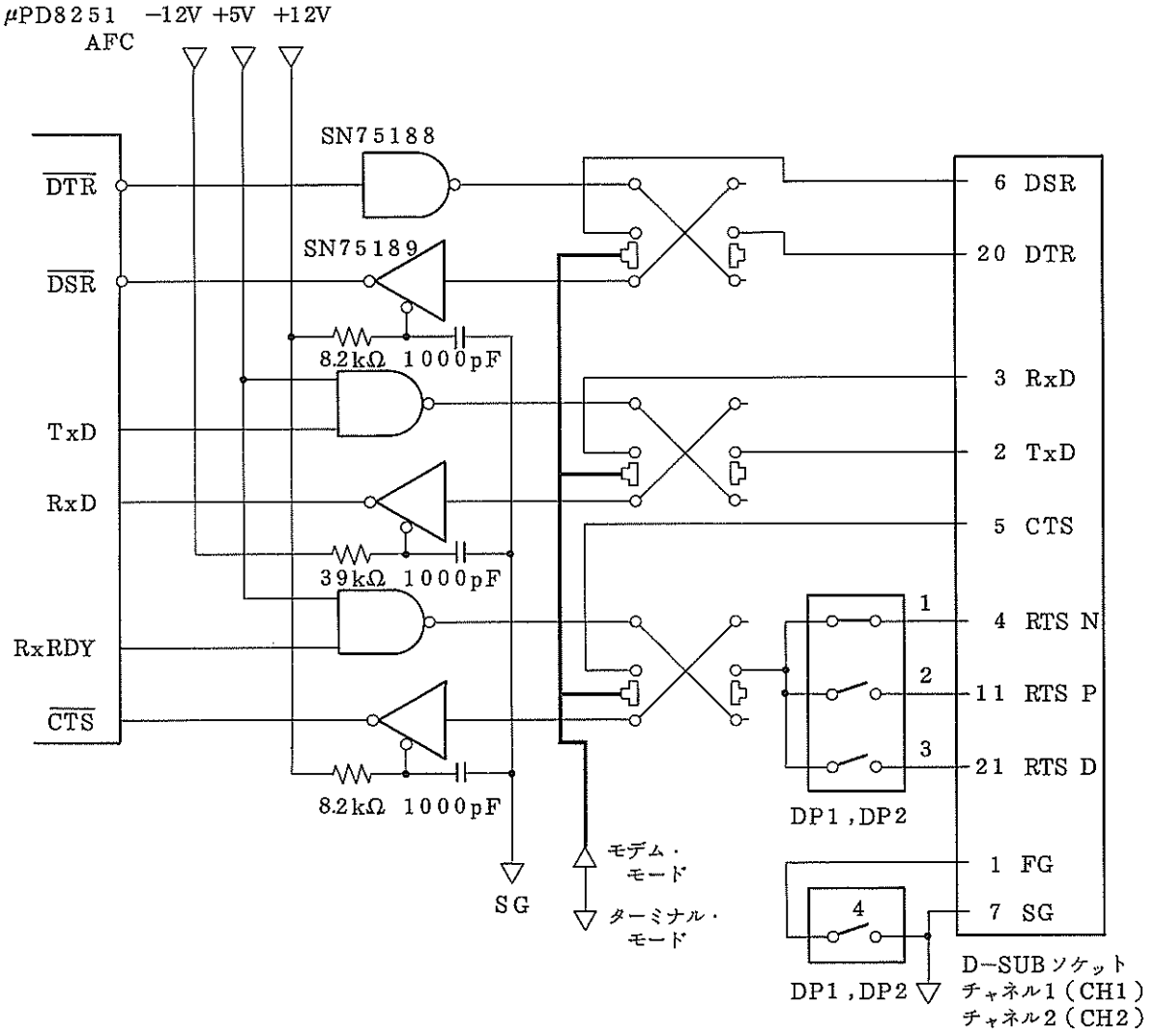
※　拡張スロットに，ＳＢ－０５１２以外のメモリ・ボードを使用される場合はＪＰ５の１－６，２－５をショートしてください。

（5）RS－232－Cインタフェース回路

IE－78310A－Rは，筐体のフロント・パネル面にRS－232－Cインタフェース用コネクタを2チャネル（チャネル1，チャネル2）内蔵しています。この2チャネルの内部の回路を図に示します。

また，インタフェース回路は，チャネル1，チャネル2とも共通です。

図付－1　RS－232－Cインタフェース回路図



（6）ターゲットとの接続方法

○ターゲット・プローブをターゲット・システムのＣＰＵソケットに差込む場合

・ターゲット・プローブのＩＣマークおよび１番ピンマークに合わせてターゲット・システムのＣＰＵソケットに差込みます。

・ターゲット・プローブのアース・クリップは必ずターゲット・システムのシグナル・グラウンド・ラインに接続してください。

○外部センス・クリップをターゲット・システムに接続する場合

・外部センス・クリップは，必ずシグナル・ラインおよびシグナル・グラウンド・ラインと接続してください。

・接続するときは，ＩＣクリップを使用してください。

○電源投入順序

①　ＩＥ－７８３１０Ａ－Ｒの電源スイッチを入れます。

②　ターゲット・システムの電源スイッチを入れます。

この順序を間違えますとＩＥ－７８３１０Ａ－Ｒが正常に動作しません。

また，ＩＥ－７８３１０Ａ－Ｒが破壊することがありますので注意してください。

○電源切断順序

①　ターゲット・システムの電源スイッチを切ります。

②　ＩＥ－７８３１０Ａ－Ｒの電源スイッチを切ります。

この順序を間違えますとＩＥ－７８３１０Ａ－Ｒが破壊することがありますので注意してください。

（7）コマンド一覧

コマンド一覧を示します。

表のコマンド本体のところに　＊が書いてあるコマンドは，システム・ソフトウエア使用時だけ有効です。

また，＊＊が書いてあるコマンドは，スタンドアロン動作時だけ有効なコマンドです。

コマンド種類		コマンド本体	サブコマンド	オペランド
ライン・アセンブラ		ＡＳＭ	なし	[word]（ｱｾﾝﾌﾞﾗ のｽﾀｰﾄ･ｱﾄﾞﾚｽ）
物理ブレーク条件設定	ブレーク条件設定 ハードウエア・ブレーク条件設定	ＢＲＡ	なし	[A=addrs][V=mask] [C= OP（ｵﾍﾟﾗﾝﾄﾞ･ﾌｪｯﾁ） / RW（ﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） / R（ﾃﾞｰﾀ･ﾘｰﾄﾞ） / W（ﾃﾞｰﾀ･ﾗｲﾄ） / RWP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） / RP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾘｰﾄﾞ） / WP（ﾌﾟﾛｸﾞﾗﾑによるﾃﾞｰﾀ･ﾗｲﾄ） / RWM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾘｰﾄﾞ/ﾗｲﾄ） / RM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾘｰﾄﾞ） / WM（ﾏｸﾛ･ｻｰﾋﾞｽによるﾃﾞｰﾀ･ﾗｲﾄ） / NC（ｵﾍﾟﾗﾝﾄﾞ･ﾌｪｯﾁを含むすべてのﾘｰﾄﾞ/ﾗｲﾄ）] [L=byte] （ｱﾄﾞﾚｽ）（ﾌﾞﾚｰｸ･ﾃﾞｰﾀ）（ﾌﾞﾚｰｸ･ｽﾃｰﾀｽ）（ﾙｰﾌﾟ･ｶｳﾝﾄ数） （各オペランドは_で区切って入力する）
	外部信号ブレーク条件設定	ＢＲＤ	なし	[mask]（外部ｾﾝｽ 信号のﾌﾞﾚｰｸ･ﾃﾞｰﾀ）
	ｲﾝｽﾄﾗｸｼｮﾝ･ｶｳﾝﾄ･ブレーク条件設定	ＢＲＥ	なし	[word]（ﾌｪｯﾁ命令数）
	タイマ・ブレーク条件設定	ＢＲＴ	なし	[word]（実行時間。単位は10μs）
論理ブレーク条件設定		ＢＲＭ	なし	[BRA][BRD][BRE][BRT][BR0][BR1][BR2][BR3]（ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ名）
		ＢＲ０ ＢＲ１ ＢＲ２ ＢＲ３	なし	[BRA][BRD][BRE][BRT]（物理ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ 名）
クロック選択		ＣＬＫ	[U / I]	なし （U:ﾕｰｻﾞ･ｼｽﾃﾑのｸﾛｯｸ，I:IE内部のｸﾛｯｸ）

コマンド種類	コマンド本体	サブコマンド	オペランド
コマンド・ファイル作成	COM *	なし	[LST: / CON: / file(コマンド・ファイル名)]
逆アセンブラ	DAS	なし	[word (逆アセンブラ のスタート・アドレス) / partition(逆アセンブラ のスタート・アドレス とエンド・アドレス)]
自己診断	DIG	なし	なし
ディレクトリ表示	DIR *	なし	[file] (ファイル名)
システム・モード終了	EXT *	なし	なし
コマンド・ヒストリ表示	HIS *	なし	なし
ヘルプ	HLP *	なし	[command] (表示したいコマンドのコマンド本体)
オブジェクト・ロード	LOD**	なし	[TTY1(チャネル1) / TTY2(チャネル2)]
オブジェクト／シンボル・ロード	LOD *	なし	file(オブジェクト/シンボル・ファイル 名)[module name＼・・・・・・] (モジュール名) [C(オブジェクト指定) / S(シンボル指定)]
出力ディバイス・リダイレクト	LST *	なし	[LST: / CON: / file (出力ファイル名)]
マッピング	MAP	[W / R / U / K]	[partition] (マッピング範囲) (W:内部マッピング , R:ライト・プロテクト 内部マッピング , U:ユーザ・マッピング , K:マッピング 解除)
演算	MAT	なし	word (通常は式を記述する)

コマンド種類	コマンド本体	サブコマンド	オペランド
モード・レジスタ操作	MDR	[D](表示)	[mode register name]
		C(変更)	[mode register name]
メモリ操作	MEM	C(変更)	[word] (変更スタート・アドレス)
		[D](表示)	[{ word (表示スタート・アドレス) / partition (表示スタート・アドレス と表示エンド・アドレス) }]
		F(イニシャライズ)	partition_data string → (イニシャライズ・データ(8ビット) の集まり) → (イニシャライズ・スタート・アドレス とエンド・アドレス)
		G(サーチ)	partition_data string → (サーチ・データ(8ビット) の集まり) → (サーチ・スタート・アドレス とエンド・アドレス)
		M(コピー)	partition_word → (コピー 先スタート・アドレス) → (コピー 元スタート・アドレス とエンド・アドレス)
		X(交換)	partition_word → (交換先スタート・アドレス) → (交換元スタート・アドレス とエンド・アドレス)
		V(比較)	partition_word → (比較先スタート・アドレス) → (比較元スタート・アドレス とエンド・アドレス)
		E(テスト)	[partition] (テスト・スタート・アドレスとエンド・アドレス)

コマンド種類	コマンド本体	サブコマンド	オペランド
チャネル２モード設定	ＭＯＤ	なし	[MODE= {CHAR / FLOW}] [BAUD= {19200 / 9600 / 4800 / 2400 / 1200 / 600 / 300}] [LONG= {7 / 8}] [PAR= {NON / EVEN / ODD}] [STOP= {1 / 2}] (ハンドシェイク・モード) (ボーレート) (キャラクタ長) (パリティ・ビット) (ストップ・ビット 長)
内部→ユーザ／ ユーザ→内部 メモリ転送	ＭＯＶ	{I / U}	partition_word →(コピー先スタート・アドレス) →(コピー元スタート・アドレス とエンド・アドレス) (U:内部→ユーザ , I:ユーザ →内部)
端末モード	ＰＧＭ	[C]	なし [省略時はディフォールト の制御キャラクタ でPROMプログラマ 制御を行う。]
レジスタ操作	ＲＥＧ	C(変更)	[register name]
		[D](表示)	[register name]　ALLを指定すると，全レジスタ・バンクの内容が表示されます。
エミュレーション操作	ＲＵＮ	N	[word] (実行スタート・アドレス)　(N:ブレークなしリアルタイム実行)
		B	[word] (実行スタート・アドレス)　(B:ブレーク付きリアルタイム実行)
		S	[word][,word]　(S:ステップ数指定リアルタイム実行) →(ステップ 数) →(実行スタート・アドレス)
		T	[word] [, {※ / word} [_TRD][_REG]] →(レジスタ 表示指定) →(トレース 表示指定) →(ブレーク 条件, ※はレジスタ条件, word はステップ数) ※…… register name {= / > / < / => / >= / =< / <= / >< / <>} {mask / wmask} (T:トレース 実行)

コマンド種類	コマンド本体	サブコマンド	オペランド
リセット	ＲＥＳ	[H]	なし (H: 省略時はｴﾊﾟﾁｯﾌﾟ だけのﾘｾｯﾄ。指定時はIEすべてのﾘｾｯﾄ。)
オブジェクト・セーブ	ＳＡＶ＊＊	なし	{TTY1(ﾁｬﾈﾙ1) / TTY2(ﾁｬﾈﾙ2)} [_partition][_partition] ・・・・ [_partition] 最大五つまで
	ＳＡＶ ＊	なし	file (入力ﾌｧｲﾙ名)[_partition][_partition] ・・・・ [_partition] 最大五つまで
特殊レジスタ操作	ＳＰＲ	C(変更)	[special register name]
		[D](表示)	[special register name]
入力ﾃﾞﾊﾞｲｽ・ﾘﾀﾞｲﾚｸﾄ	ＳＴＲ ＊	なし	file (入力ﾌｧｲﾙ名)
サフィックス指定	ＳＵＦ	なし	[{H(16進) / T(10進) / Q(8進) / Y(2進)}]
アペンド・シンボル操作	ＳＹＭ	[D](表示)	なし
		K(削除)	なし(すべてのｱﾍﾟﾝﾄﾞ・ｼﾝﾎﾞﾙ を削除)
		A(ｱﾍﾟﾝﾄﾞ)	symbol_word →(ｼﾝﾎﾞﾙ 値) →(ｱﾍﾟﾝﾄﾞ・ｼﾝﾎﾞﾙ名)
		C(変更)	symbol_word →(変更ｼﾝﾎﾞﾙ値) →(ｱﾍﾟﾝﾄﾞ・ｼﾝﾎﾞﾙ名)
		E(削除)	[symbol] →(削除するｱﾍﾟﾝﾄﾞ・ｼﾝﾎﾞﾙ 名)

コマンド種類	コマンド本体	サブコマンド	オペランド
アペンド・シンボル操作	ＳＹＭ　＊	L(ロード)	なし
		S(セーブ)	なし
シンボル操作	ＳＹＭ　＊	[D](表示)	[{module＼(モジュール指定) / PUBLIC　(パブリック指定)}]
カレント・モジュールの変更	ＳＹＭ	M(変更)	なし
トレース・モード設定	ＴＲＭ	なし	[{NON(ノン・トレース) / ALL(全トレース) / TRX(クオリファイ・トレース)}]
クオリファイ条件設定	ＴＲＸ	なし	[A=addrs](クオリファイ・アドレス)[V=mask](クオリファイ・データ) [C= {OP (オペランド・フェッチ) / RW (データ・リード/ライト) / R (データ・リード) / W (データ・ライト) / RWP(プログラムによるデータ・リード/ライト) / RP (プログラムによるデータ・リード) / WP (プログラムによるデータ・ライト) / RWM(マクロ・サービス によるデータ・リード/ライト) / RM (マクロ・サービス によるデータ・リード) / WM (マクロ・サービス によるデータ・ライト) / NC(オペランド・フェッチを含むすべてのリード/ライト)}](クオリファイ・ステータス) [TRQ=mask](クオリファイ・ポート/外部データ)
クオリファイ・データ選択	ＴＲＱ	なし	[{TRS(ポート 選択) / EXT(外部信号選択)}]

コマンド種類	コマンド本体	サブコマンド	オペランド
トレース／クオリファイ ポート選択	ＴＲＳ	なし	[{ P0 (ポート0) P1 (ポート1) P2 (ポート2) P3 (ポート3) P4 (ポート4) P5 (ポート5) }]
トレース・ポインタ操作	ＴＲＰ	なし	[{ word (ポインタ移動数) 0 (ポインタ を先頭に置く) N (ポインタ を最後に置く) }]
トレース表示	ＴＲＤ	[{ F I M }]	[{ word (表示ステップ数) ALL (トレース結果すべてを表示) }] (F:フレーム・モード , I:インストラクション・モード , M:マクロ・サービス付きインストラクション・モード)
オブジェクト・ ベリファイ	ＶＲＹ＊＊	なし	{ TTY1(チャネル1) TTY2(チャネル2) }
	ＶＲＹ ＊	なし	file (入力ファイル名)

（8）エラー・メッセージ一覧

エラー・メッセージの一覧を以下に示します。

（1）Unrecognized command

コマンド・キーワードが正しくない。

（2）Command format error

コマンド・キーワードは正しいが，オペランドが正しくない。

（3）Command/Data too long

１２８文字以上のコマンド，あるいは，データ行が入力された。

（4）Mapping error

指定されたアドレス範囲に，マッピングされていないメモリ・エリアがある。

（5）Input data error

入力したデータが正しくない。

（6）System mode command

スタンドアロン・モードでシステム・モードのコマンドを入力した。

（7）Non map area access

コマンド実行中にマッピングされていないメモリにアクセスしようとした。

（8）Check sum error

オブジェクトのロード／セーブ時にチェック・サム・エラーを検出した。

（9）Bad character

オブジェクトのロード／セーブ時に正しくない文字を検出した。

(10) aborted

オブジェクトのロード／セーブ中に中断キーを入力された。

(11) Warning double define

ＬＯＤコマンドで，同一モジュール名が複数回指定された。

(12) Bad file entry

ファイル名の記述が正しくない。

(13) File overflow

ＬＯＤコマンドで，入力可能なシンボル・ファイル数をオーバした。

(14) Illegal record

ＬＯＤコマンドで，シンボル・テーブル・ファイルのレコード形式が正しくない。

(15) load failed

ＬＯＤコマンドで，シンボルをロード中にエラーを検出した。

(16) module overflow

ＬＯＤコマンドで，入力できるモジュール数をオーバした。

(17) Not loaded symbol

シンボルがロードされていない。

(18) Not found module record

ＬＯＤコマンドで，指定されたモジュール名がシンボル・テーブル・ファイルに存在しない。

(19) file not found

指定されたファイル名が存在しない。

(20) Slave CPU communication error

チャネル２のスレーブＣＰＵ(8742)に対し，コマンドが書込めない。

(21) double define append symbol シンボル名

ＬＯＤコマンドで，アペンド・シンボルとして，すでに登録されているシンボルをロードした（アペンド・シンボルは削除されます)。

(22) double define append symbol

ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，すでに登録されているシンボルを登録しようとした。

(23) symbol table full

ＬＯＤコマンドで，シンボル・セーブ・エリアに空がない。

(24) append symbol table full

ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，アペンド・シンボル・セーブ・エリアに空がない。

(25) double define loaded symbol

ＬＯＤ，ＳＹＭ　Ａ，ＳＹＭ　Ｌコマンドで，すでにロードされているシンボルがロードされた。

(26) symbol record format error

ＬＯＤ，ＳＹＭ　Ｌコマンドで，シンボル・テーブル・ファイルのレコード形式が正しくなかった。

(27) reserved word symbol

ＳＹＭ　Ａコマンドで，予約語がシンボルとして定義された。

(28) double define module name ﾓｼﾞｭｰﾙ 名

表示されたモジュール名は，すでにロードされている。

(29) module buffer full

ＬＯＤコマンドで，入力できるモジュール数をオーバした。

(30) not found symbol

ＳＹＭ　Ｃ，ＳＹＭ　Ｅコマンドで，指定されたシンボルは存在しない。

(31) no symbol of append

ＳＹＭ　Ｄ，ＳＹＭ　Ｓコマンドで，アペンド・シンボルは存在しない。

(32) Can not execute HLP command !

カレント・ディスク上にヘルプ・ファイル(IE78310.HLP)，ヘルプ・オーバーレイ・ファイル(IE78310.OV2)が存在しない。

(33) No .HLP file on the default drive

ＨＬＰコマンド実行時，カレント・ディスク上にヘルプ・ファイル(IE78310.HLP)，ヘルプ・オーバーレイ・ファイル(IE78310.OV2)が見つからなかった。

(34) Keyword Error

ＨＬＰコマンドで，コマンド・キーワードが正しくない。

(35) Can not use command abbreviation !

カレント・ディスク上に省略形のオーバーレイ・ファイル(IE78310.OV2)が存在しない。

(36) File already exists.

ファイルの属性が SYS属性，あるいは，R/O 属性のファイルに対して同一名のファイルを新たにメイクしようとした。

(37) Reserved file name

システム・ソフトが使う予約されたファイル名を指定した。

(38) File name is used by other process

すでにオープン済みのファイル名を指定した。

(39) Can not close ﾌｧｲﾙ名

表示されたファイルのクローズが正常にできなかった。

(40) Disk write error ﾌｧｲﾙ 名

表示されたファイルの書込みで異常を見つけた。

(41) Disk read error ﾌｧｲﾙ名

表示されたファイルの読込みで異常を見つけた。

(42) Can not open ﾌｧｲﾙ 名

指定されたファイルがオープンできなかった。

(43) File make error ﾌｧｲﾙ名

表示されたファイルを作成できなかった。

(44) Can not close ﾌｧｲﾙ名.Cancel ××× command

×××のコマンド実行中，表示されたファイルのクローズが正常にできなかった（×××はＳＴＲ，ＬＳＴ，ＣＯＭの各コマンド)。

(45) Disk write error ﾌｧｲﾙ名.Cancel ××× command

×××のコマンド実行中，表示されたファイルの書込みで異常を見つけた（×××は，ＬＳＴ，ＣＯＭの各コマンド)。

(46) Disk read error ﾌｧｲﾙ名.Cancel STR command

ＳＴＲコマンド実行中，表示されたファイルの読込みで異常を見つけた。

(47) List device is used by other process

他の処理がリスト装置を使っている（COM ｺﾏﾝﾄﾞとLST ｺﾏﾝﾄﾞの両方で ﾘｽﾄ装置を指定した場合，または，CCP/M　の場合にIE-78310A-R以外の処理がﾘｽﾄ 装置を使用している場合)。

(48) Append symbol file not found

ＳＹＭ　Ｌコマンドで，アペンド・シンボル・ファイル(IE78310.SYM) がカレント・ディスク上に存在しなかった。

(49) Illegal append symbol file

ＳＹＭ　Ｌコマンドで，アペンド・シンボル・ファイルの形式が正しくない。

(50) Communication error

ＩＥとホスト・マシンの通信が正常にできなかった。

(51) Not found memories

外部メモリが指定されたのにメモリが使用できない。

(52) Non map area access!

ＡＳＭコマンド実行中，マッピングされていないメモリにアクセスしようとした。

(53) Assemble area over!

ＡＳＭコマンドで，アクセスできるメモリの範囲を越えた。

(54) Disassemble area over!

ＤＡＳコマンドで，アクセスできるメモリの範囲を越えた。

(55) Caution!

ジェネリックなオブジェクトが生成された，あるいは，注意を要する場合に表示される。

(56) Error!

オブジェクト・コードを生成できないか，あきらかにエラーである場合に表示される。

（9）実際のディバイスとの差

実際のディバイス（μＰＤ７８３１２Ａ，μＰＤ７８３１０Ａ）はＣＭＯＳの回路ですがＩＥ－７８３１０Ａ－Ｒのターゲット・インタフェース回路は，エバリュエーション・チップ（μＰＤ７８３１９Ａ）とＴＴＬ等によるエミュレーション回路で構成されています。ターゲット・インタフェースの信号線は３種類に分けられ，実際のディバイスとの差は次のようになっています。

① エバリュエーション・チップから直接取出されている信号

ポート０，ポート２，ポート３，Ａ／Ｄコンバータ関係，リフレッシュ信号は実際のディバイスとまったく同じ動作をします。

注　ただし，Ａ／Ｄコンバータ関係を除いて直列に100 Ωの抵抗が挿入されています。

② エバリュエーション・チップからゲートを通して取出されている信号

ＲＥＳＥＴ，クロック入力，ＲＤ，ＷＲ，ＡＬＥ信号は，ＴＴＬが入っていることにより実際のディバイスより信号が遅れます。

③ エミュレーション回路から取出されている信号

ＶＣＣ，$\overline{\text{ＥＡ}}$端子は，エミュレーション回路のＴＴＬでセンスしています。

エバリュエーション・チップの電源は，ＩＥ内の電源から供給しています。

ターゲット・システムのＶＣＣはエバリュエーション・チップには，接続されません。

Ｐ１，Ｐ４，Ｐ５はエミュレーション回路の各ポート・エミュレータから取出されています。ターゲットへの出力側はＬＳ　ＴＴＬ，ターゲットからの入力側はＨＣＴロジックとなっています。

したがって実際のディバイスとは，ＤＣ特性，ＡＣ特性とも若干異なります。

（１０）ディバッガ使用上の注意

○ＲＳ－２３２－Ｃインタフェース

・ターミナルとターミナル，モデムとモデムの接続は絶対しないこと。ただし，ＰＧ－１５００，ＰＧ－２０００と接続の場合，ＩＥ－７８３１０Ａ－Ｒは，モデム・モードに設定します。また接続には，ＰＧ－１５００，ＰＧ－２０００に添付されているＲＳ－２３２－Ｃインタフェース・ケーブルを使用します。

・ＲＴＳの設定はプロタイパを除いては必ずＲＴＳＮ を設定しておいてください。ハンドシェイク方式は，ハードウェア／ソフトウェア・ハンドシェイクがあります。チャネル１はハードウェア／ソフトウェア・ハンドシェイク兼用。

チャネル２は，コマンドによりハードウェア・ハンドシェイクとソフトウェア・ハンドシェイクに切替え可能です。接続する装置と合わせてください。

・ボーレートは接続した装置と同一に合わせます。

○ターゲット・プローブ

・ターゲット・プローブのアース・クリップは必ずターゲット・システムのシグナル・グラウンド・ラインに接続します。

○外部センス・クリップ

・外部センス・クリップは，ＴＴＬレベルの信号線にだけ接続します。

（11）実　行　例

第3章の実行例を以下に示します。

（本実行例では，下線部がキーボードからの入力を表します。）

```
A>IE78310 ↵
XX:XX:XX A:IE78310.CMD

  IE-78310 CONTROLLER    (MD-086/116  SERIES) Vx.x  [Dd Mmm Yy]
       Copyright  (C) 1985 by NEC Corporation

  Do you want to use COMMAND LINE EDITOR (Y or N) : Y ↵
  Window off !
  Select port NO. (1 to 4) :  1 ↵
  IE-78310A Monitor Vx.x  [Dd Mmm Yy]
             Copyright (C) 1985 by NEC Corporation

  Power on target system (Y/N) Y ↵
  Create new set up mode (Y or N) : N ↵
  Internal ROM size (4K,8K,16K) = 8K ↵
  Tracer  initialize                              (ﾄﾚｰｽ･ﾒﾓﾘの初期設定ﾒｯｾｰｼﾞ)
  Breaker initialize                              (ﾌﾞﾚｰｸ条件等の初期設定ﾒｯｾｰｼﾞ)
  Do you have Memory Board on IE-78310A ? (Y/N)= N ↵
1>CLK I ↵
1>RES ↵
1>MAP ↵
  0000-1FFF R/O   2000-FDFF Non
1>MAP W 2000,0FDFF ↵
1>MAP ↵
  0000-1FFF R/O   2000-FDFF R/W
1>MEM F 0,1FFF 00 ↵
1>LOD SORT ↵
  object load complete
  symbol table loading
  MODULE01   load complete
1>LOD SORT.HEX C ↵
  object load complete
1>SYM K ↵
1>LOD SORT.SYM S ↵
  symbol table loading
  MODULE01   load complete
1>MEM D 100,12F ↵
  0100   3A 20 01 3A 21 00 20 4A 9F 21 80 08 6F 20 00 00     : .:!. J.!..o ..
  0110   00 00 14 FB B8 00 67 42 FE D8 88 E8 59 16 5F 83     ......gB....Y._.
  0120   07 81 05 16 34 55 26 20 26 21 14 DA 00 00 00 00     ....4U& &!......
```

```
1>MEM D 100 ↵
  0100   3A 20 01 3A 21 00 20 4A 9F 21 80 08 6F 20 00 00     : .:!. J.!..o ..
  0110   00 00 14 FB B8 00 67 42 FE D8 88 E8 59 16 5F 83     ......gB....Y._.
  0120   07 81 05 16 34 55 26 20 26 21 14 DA 00 00 00 00     ....4U& &!......
  0130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>MEM D ↵
  01B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  01F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0200   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0210   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0220   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0230   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0240   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0250   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>MEM ↵
  0260   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0270   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0280   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0290   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  02F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  0300   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
1>MAP ↵
  0000-1FFF R/O   2000-FDFF R/W
1>MAP K 2000,0FDFF ↵
1>MAP ↵
  0000-1FFF R/O   2000-FDFF Non
1>MEM D 2000,0FDFF ↵
  Mapping error
```

```
1>DAS 100,12B )
  Addr  Object          Mnemonic
                        ORG     MODULE01\SORT
                      MODULE01\SORT:
  0100  3A 20 01        MOV     0FE20H,#1H
  0103  3A 21 00        MOV     0FE21H,#0H
                      MODULE01\COMP:
  0106  20 4A           MOV     A,0FE4AH
  0108  9F 21           CMP     A,0FE21H
  010A  80 08           BNZ     $CONT
  010C  6F 20 00        CMP     0FE20H,#0H
                      MODULE01\STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MODULE01\CONT:
  0114  B8 00           MOV     R0,#0H
  0116  67 42 FE        MOVW    RP7,#0FE42H
  0119  D8              XCH     A,R0
  011A  88 E8           ADDW    RP7,RP0
  011C  59              MOV     A,[HL+]
  011D  16 5F           CMP     A,[HL]
  011F  83 07           BC      $INC1
  0121  81 05           BZ      $INC1
  0123  16 34           XCH     A,[HL-]
  0125  55              MOV     [HL],A
  0126  26 20           INC     0FE20H
                      MODULE01\INC1:
  0128  26 21           INC     0FE21H
  012A  14 DA           BR      $COMP
                        END

1>DAS 100 )
  Addr  Object          Mnemonic
                        ORG     MODULE01\SORT
                      MODULE01\SORT:
  0100  3A 20 01        MOV     0FE20H,#1H
  0103  3A 21 00        MOV     0FE21H,#0H
                      MODULE01\COMP:
  0106  20 4A           MOV     A,0FE4AH
  0108  9F 21           CMP     A,0FE21H
  010A  80 08           BNZ     $CONT
  010C  6F 20 00        CMP     0FE20H,#0H
                        END

1>DAS )
  Addr   Objiect        Mnemonic
                        ORG     MODULE01\STOP
                      MODULE01\STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MODULE01\CONT:
  0114  B8 00           MOV     R0,#0H
  0116  67 42 FE        MOVW    RP7,#0FE42H
                        END
```

```
1>SYM A SW 0FE20 )
1>SYM A I 0FE21 )
1>SYM A STACK 0FE80 )
1>SYM A LIST 0FE42 )
1>SYM A N 0FE4A )
1>MEM F 0FE00,0FE7F 0 )
1>MEM C 0FE42 )
  FE42  00 05 )
  FE43  00 03 )
  FE44  00 04 )
  FE45  00 0A )
  FE46  00 08 )
  FE47  00 82 )
  FE48  00 0A )
  FE49  00 04 )
  FE4A  00 08 )
  FE4B  00 . )
1>MEM D 0FE20,0FE4A )
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                   ...........
1>MEM D SW,N )
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                   ...........
1>MEM D 0FE20,N )
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                   ...........
1>MEM D SW,0FE4A )
  FE20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE40   00 00 05 03 04 0A 08 82 0A 04 08                   ...........
1>BRA )
  A 0H,0FFFFH = MODULE01\STOP
  V 0XXXXXXXXY = __ )
    OPcode fetch                  (OP)
    Read Write                    (RW)
    Read                          (R)
    Write                         (W)
    Read Write by Program         (RWP)
    Read by Program               (RP)
    Write by Program              (WP)
    Read Write by Macro service   (RWM)
    Read by Macro service         (RM)
    Write by Macro service        (WM)
    No Condition                  (NC)
  C NC = OP )
  L 1H = __ )
1>BRM BRA )
1>BRM )
  BRA )
1>REG C PC )
  PC       0000 = 100 )
  SP       FE72 = 0FE80 )
```

```
1>RUN B 100 ↵
  User-system Vcc-ON     Emulation start at 0100
  Standard break  terminated
   PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0112  FE80        0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A     C    B                           VP       UP       DE       HL
    00   00    CB   F7   FF   FF   FF   FB     FFFF     F6FF     FFFF     FE43
  One step emulation standby ESC キー 入力
1>MEM D SW,N ↵
  FE20   03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
  FE40   00 00 03 05 04 0A 08 82 0A 04 00                     ...........
1>MEM C LIST ↵
  FE42  03 05 ↵
  FE43  05 03 ↵
  FE44  04 ↵
  FE45  0A ↵
  FE46  08 ↵
  FE47  82 ↵
  FE48  0A ↵
  FE49  04 ↵
  FE4A  00 08 ↵
  FE4B  08 00 ↵
  FE4C  00 . ↵
1>REG C PC ↵
  PC       0112 = 100 ↵
  SP       FE80 = . ↵
1>RUN T ,6 REG ↵
  User-system Vcc-ON     Emulation start at 0100
   PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0103  FE80        0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A     C    B                           VP       UP       DE       HL
    00   00    CB   F7   FF   FF   FF   FB     FFFF     F6FF     FFFF     FE43

   PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80        0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A     C    B                           VP       UP       DE       HL
    00   00    CB   F7   FF   FF   FF   FB     FFFF     F6FF     FFFF     FE43

   PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80        0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A     C    B                           VP       UP       DE       HL
    00   08    CB   F7   FF   FF   FF   FB     FFFF     F6FF     FFFF     FE43

   PC    SP   PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  010A  FE80        0    0    0    0    0    0    0    0    0    0    1    0
    R0   R1    R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A     C    B                           VP       UP       DE       HL
    00   08    CB   F7   FF   FF   FF   FB     FFFF     F6FF     FFFF     FE43
```

```
 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0114  FE80       0    0    0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
   X    A    C    B                             VP        UP        DE        HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43

 PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0116  FE80       0    0    0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
   X    A    C    B                             VP        UP        DE        HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE43

  terminated
Frame Status Address Data Label  Mnemonic                    P0 EX
                             MODULE01\CONT:                  B0 FF
0000          0114              MOV     R0,#0H
One step emulation standby  ↵キー入力
Frame Status Address Data Label  Mnemonic                    P0 EX
0000          0116                MOVW    RP7,#LIST
 PC    SP  PSW: RBS2  RBS1  RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
0119  FE80       0     0     0    0    0    0    0    0    0    0    1    0
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
   X    A    C    B                             VP        UP        DE        HL
  00   08   CB   F7   FF   FF   FF   FB      FFFF      F6FF      FFFF      FE42
One step emulation standby ESC キー 入力
1>ASM MODULE01\CONT ↵
0114  MODULE01\CONT: ↵
0114                  MOV     R0,#0H
                = BR $12C ↵
      14 16
0116                  MOVW    RP7,#LIST
                = ORG 12CH ↵
012C                  NOP
                = MOV A,I ↵
      20 21
012E                  NOP
                = MOV R0,#0H ↵
      B8 00
0130                  NOP
                = BR $116 ↵
      14 E4
0132                  NOP
                = END ↵
1>DAS 114,116 ↵
Addr  Object              Mnemonic
                          ORG     MODULE01\CONT
                        MODULE01\CONT:
0114  14 16               BR      $12CH
0116  67 42 FE            MOVW    RP7,#LIST
                          END

1>DAS 12C,131 ↵
Addr  Object              Mnemonic
                          ORG     12CH
012C  20 21               MOV     A,I
012E  B8 00               MOV     R0,#0H
0130  14 E4               BR      $116H
                          END
```

```
1>RUN B 100 ↵
  User-system Vcc-ON     Emulation start at 0100
  Non map area access break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    1    0    1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
                      X    A    C    B        VP       UP       DE       HL
    BD   FE   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FD00
  One step emulation standby ESC キー 入力
1>MEM D SW,N ↵
  FE20   73 BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00     s...............
  FE30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE40   00 00 03 04 05 08 0A 0A 04 08 00                    ...........
1>REG C RSS ↵
  RSS        1 = 0 ↵
1>REG ↵
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    0    0    1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    BD   FE   CB   F7   FF   FF   FB   FF      FFFF     FFF6     FFFF     FD00
1>ASM 10C ↵
  010C                          CMP     SW,#0H
                          = BR $132 ↵
       14 24
  010E                          NOP
                          = NOP ↵
       00
  010F  MODULE01\STOP: ↵
  010F                          NOP
                          = END ↵
1>ASM 132 ↵
  0132                          NOP
                          = CMP SW,#0H ↵
       6F 20 00
  0135                          NOP
                          = BNZ  $MODULE01\CONT ↵
       80 DD
  0137                          NOP
                          = BR $MODULE01\STOP ↵
       14 D6
  0139                          NOP
                          = END ↵
1>DAS 10C,10E ↵
  Addr  Object              Mnemonic
                            ORG     10CH
  010C  14 24               BR      $132H
  010E  00                  NOP
                            END
1>DAS 132,138 ↵
  Addr  Object              Mnemonic
                            ORG     132H
  0132  6F 20 00            CMP     SW,#0H
  0135  80 DD               BNZ     $CONT
  0137  14 D6               BR      $STOP
                            END
```

```
1>DAS MODULE01\CONT ↵
  Addr  Object              Mnemonic
                            ORG     MODULE01\CONT
                          MODULE01\CONT:
  0114  14 16               BR      $12CH
  0116  67 42 FE            MOVW    RP7,#LIST
  0119  D8                  XCH     A,R0
  011A  88 E8               ADDW    RP7,RP0
  011C  59                  MOV     A,[HL+]
  011D  16 5F               CMP     A,[HL]
  011F  83 07               BC      $INC1
                            END
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ↵
1>BRA A=11F C=0P ↵
1>RUN B 100 ↵
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0125  FE80       0    0    0    0    0    0    0    0     1    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                          VP      UP      DE      HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF    FFF6    FFFF    FE42
  One step emulation standby  ↵キー入力
  Frame Status Address Data Label  Mnemonic                    P0 EX
  0000          0125               MOV     [HL],A
  0004    WR    FE42    03                                     B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0126  FE80       0    0    0    0    0    0    0    0     1    0    1    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                          VP      UP      DE      HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF    FFF6    FFFF    FE42
  One step emulation standby  ↵キー入力
  Frame Status Address Data Label  Mnemonic                    P0 EX
  0000          0126               INC     SW
  0003    RD    FE20    01                                     B0 FF
  0004    WR    FE20    02                                     B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0128  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                          VP      UP      DE      HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF    FFF6    FFFF    FE42
  One step emulation standby  ↵キー入力
  Frame Status Address Data Label  Mnemonic                    P0 EX
                                   MODULE01\INC1:              B0 FF
  0000          0128               INC     I
  0003    RD    FE21    00                                     B0 FF
  0004    WR    FE21    01                                     B0 FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4     RP5     RP6     RP7
     X    A    C    B                          VP      UP      DE      HL
    00   03   CB   F7   FF   FF   FB   FF      FFFF    FFF6    FFFF    FE42
  One step emulation standby ESC キー 入力
```

```
1>MEM D LIST,LIST+7 ↵
  FE42        03 05 04 0A 08 82 0A 04                          ........
1>MEM D I,I ↵
  FE21      01
1>RUN T ,1 ↵
  User-system Vcc-ON     Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                  PO EX
  0000          012A              BR        $COMP
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    00   03   CB   F7   FF   FF   FB   FF       FFFF      FFF6      FFFF      FE42
  One step emulation standby ↵ キー 入力
  Frame Status Address Data Label  Mnemonic                  PO EX
                                MODULE01\COMP:               BO FF
  0000          0106              MOV       A,N
  0003    RD    FE4A    08                                   BO FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    00   08   CB   F7   FF   FF   FB   FF       FFFF      FFF6      FFFF      FE42
  One step emulation standby ESCキー入力
1>BRA A=MODULE01\INC1 C=0P ↵
1>RUN B ↵
  User-system Vcc-ON     Emulation start at 0108
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    01   04   CB   F7   FF   FF   FB   FF       FFFF      FFF6      FFFF      FE43
  One step emulation standby ESC キー 入力
1>MEM D I,I ↵
  FE21      02
1>MEM D LIST,LIST+7 ↵
  FE42        03 04 05 0A 08 82 0A 04                          ........
1>RUN T ,1 ↵
  User-system Vcc-ON     Emulation start at 0106
      terminated
  Frame Status Address Data Label  Mnemonic                   PO EX
                                MODULE01\COMP:                BO FF
  0000          0106              MOV       A,N
  0003    RD    FE4A    08                                    BO FF
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0108  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    01   08   CB   F7   FF   FF   FB   FF       FFFF      FFF6      FFFF      FE43
  One step emulation standby ESC キー 入力
```

```
1>RUN B ↵
  User-system Vcc-ON     Emulation start at 0108
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0     1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    02   05   CB   F7   FF   FF   FB   FF       FFFF     FFF6     FFFF     FE45
  One step emulation standby ESC キー 入力
1>MEM D I,I ↵
  FE21      03                                                    .
1>MEM D LIST,LIST+7 ↵
  FE42           03 04 05 0A 08 82 0A 04                          ........
1>BRA A=MODULE01\STOP 132 C=OP ↵
1>RUN B ↵
  User-system Vcc-ON     Emulation start at 0106
  Non map area access break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    1    0     1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
                      X    A    C    B           VP       UP       DE       HL
    BD   FE   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FD00
  One step emulation standby ESC キー 入力
1>MEM D LIST,N ↵
  FE42           03 04 05 08 0A 0A 04 08 00                       .........
1>REG C RSS ↵
  RSS          1 = 0
1>REG ↵
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  011D  FE80       0    0    0    0    1    0    0    0     1    0    0    1
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
                      X    A    C    B           VP       UP       DE       HL
    BD   FE   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FD00
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ↵
1>DAS 126,12B ↵
  Addr  Object             Mnemonic
                           ORG     126H
  0126  26 20              INC     SW
                         MODULE01\INCI:
  0128  26 21              INC     I
  012A  14 DA              BR      $COMP
                           END
1>BRA A=126 C=OP ↵
1>RUN B 100 ↵
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7       RP4      RP5      RP6      RP7
     X    A    C    B                           VP       UP       DE       HL
    00   03   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE42
  One step emulation standby ESC キー 入力
```

```
1>MEM D I,I )
  FE21      01
1>MEM D LIST,LIST+7 )
  FE42        03 05 04 0A 08 82 0A 04                          ........
1>RUN T ,1 )
  User-system Vcc-ON     Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                PO EX
  0000          012A                BRC     $COMP
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    00   03   CB   F7   FF   FF   FF   FF        F6FF      FFFF      FFFF      FE42
  One step emulation standby ESC キー 入力
1>RUN B )
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    01   04   CB   F7   FF   FF   FF   FF        F6FF      FFFF      FFFF      FE43
  One step emulation standby ESC キー 入力
1>MEM D I,I )
  FE21      02
1 MEM D LIST,LIST+7 )
  FE42        03 04 05 0A 08 82 0A 04                          ........
1>RUN T ,1 )
  User-system Vcc-ON     Emulation start at 012A
      terminated
  Frame Status Address Data Label  Mnemonic                PO EX
  0000          012A                BR      $COMP
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC   UF   P/V  SUB  CY
  0106  FE80       0    0    0    0    0    0    0    0    1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    01   04   CB   F7   FF   FF   FF   FF        F6FF      FFFF      FFFF      FE43
  One step emulation standby ESC キー 入力
1>RUN B )
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
  012A  FE80       0    0    0    0    0    0    0    0     1    0    0    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6       RP7
     X    A    C    B                           VP        UP        DE        HL
    03   08   CB   F7   FF   FF   FF   FF        F6FF      FFFF      FFFF      FE45
  One step emulation standby ESC キー 入力
1>MEM D I,I )
  FE21      04
1>MEM D LIST,LIST+7 )
  FE42        03 04 05 08 0A 82 0A 04                          ........
1>MEM D SW,SW )
  FE20   04
```

```
1>ASM 135 ↵
  0135                              BNZ     $COMP
                                  = BNZ  $MODULE01\SORT ↵
       80 C9
  0137                              BR      $STOP
                                  = END ↵
1>DAS 135,136 ↵
  Addr  Object              Mnemonic
                            ORG     135H
  0135  80 C9               BNZ     $SORT
                            END
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 ↵
1>TRM ALL ↵
1>BRA A=MODULE01\COMP L=9 C=OP ↵
1>RUN B 100 ↵
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF    P/V  SUB  CY
  010A  FE80       0    0    0    0    0    0    0    1     1     1    1    0
    R0   R1   R2   R3   R4   R5   R6   R7        RP4     RP5     RP6     RP7
     X    A    C    B                           VP      UP      DE      HL
    07   82   CB   F7   FF   FF   FF   FF       F6FF    FFFF    FFFF    FE49
  One step emulation standby ESC キー 入力
1>MEM D N,N ↵
  FE4A                              82                              .
1>MEM D I,I ↵
  FE21      08                                                      .
1>MEM D LIST,LIST+7 ↵
  FE42          03 04 05 08 0A 0A 04 08                            ........
1>TRD I ALL ↵
  Frame Status Address Data Label  Mnemonic
                               MODULE01\SORT:
  0000          0100                 MOV     SW,#1H
  0004    WR    FE20    01
  0003          0103                 MOV     I,#0H
  0008    WR    FE21    00
                               MODULE01\COMP:
  0007          0106                 MOV     A,N
  0011    RD    FE4A    08
  0010          0108                 CMP     A,I
  0014    RD    FE21    00
  0013          010A                 BNZ     $CONT
                               MODULE01\CONT:
  0018          0114                 BR      $12CH
  0022          012C                 MOV     A,I
  0025    RD    FE21    00
  0024          012E                 MOV     R0,#0H
  0027          0130                 BR      $116H
  0031          0116                 MOVW    RP7,#LIST
  0034          0119                 XCH     A,R0
  0035          011A                 ADDW    RP7,RP0
  0037          011C                 MOV     A,[HL+]
  0041    RD    FE42    05
  0038          011D                 CMP     A,[HL]
  0043    RD    FE43    03
  0040          011F                 BC      $INCI
  0044          0121                 BZ      $INCI
  0046          0123                 XCH     A,[HL-]
  0050    RD    FE43    03
  0051    WR    FE43    05
  0048          0125                 MOV     [HL],A
  0054    WR    FE42    03
  0049          0126                 INC     SW
  0055    RD    FE20    01
  0056    WR    FE20    02
```

```
                                MODULE01\INCI:
0053            0128                INC     I
0059    RD      FE21    00
0060    WR      FE21    01
0058            012A                BR      $COMP
                                MODULE01\COMP:
0064            0106                MOV     A,N
0067    RD      FE4A    08
0066            0108                CMP     A,I
0070    RD      FE21    01
0069            010A                BNZ     $CONT
                                MODULE01\CONT:
0074            0114                BR      $12CH
0078            012C                MOV     A,I
0081    RD      FE21    01
0080            012E                MOV     R0,#0H
0083            0130                BR      $116H
0087            0116                MOVW    RP7,#LIST
0090            0119                XCH     A,R0
0091            011A                ADDW    RP7,RP0
0093            011C                MOV     A,[HL+]
0097    RD      FE43    05
0094            011D                CMP     A,[HL]
0099    RD      FE44    04
0096            011F                BC      $INCI
0100            0121                BZ      $INCI
0102            0123                XCH     A,[HL-]
0106    RD      FE44    04
0107    WR      FE44    05
0104            0125                MOV     [HL],A
0110    WR      FE43    04
0105            0126                INC     SW
0111    RD      FE20    02
0112    WR      FE20    03
                                MODULE01\INCI:
0109            0128                INC     I
0115    RD      FE21    01
0116    WR      FE21    02
0114            012A                BR      $COMP
                                MODULE01\COMP:
0120            0106                MOV     A,N
0123    RD      FE4A    08
0122            0108                CMP     A,I
0126    RD      FE21    02
0125            010A                BNZ     $CONT
                                MODULE01\CONT:
0130            0114                BR      $12CH
0134            012C                MOV     A,I
0137    RD      FE21    02
0136            012E                MOV     R0,#0H
0139            0130                BR      $116H
0143            0116                MOVW    RP7,#LIST
0146            0119                XCH     A,R0
0147            011A                ADDW    RP7,RP0
0149            011C                MOV     A,[HL+]
0153    RD      FE44    05
0150            011D                CMP     A,[HL]
0155    RD      FE45    0A
0152            011F                BC      $INCI
                                MODULE01\INCI:
0158            0128                INC     I
0161    RD      FE21    02
0162    WR      FE21    03
0160            012A                BR      $COMP
                                MODULE01\COMP:
```

```
0166          0106              MOV     A,N
0169    RD    FE4A    08
0168          0108              CMP     A,I
0172    RD    FE21    03
0171          010A              BNZ     $CONT
                          MODULE01\CONT:
0176          0114              BR      $12CH
0180          012C              MOV     A,I
0183    RD    FE21    03
0182          012E              MOV     R0,#0H
0185          0130              BR      $116H
0189          0116              MOVW    RP7,#LIST
0192          0119              XCH     A,R0
0193          011A              ADDW    RP7,RP0
0195          011C              MOV     A,[HL+]
0199    RD    FE45    0A
0196          011D              CMP     A,[HL]
0201    RD    FE46    08
0198          011F              BC      $INCI
0202          0121              BZ      $INCI
0204          0123              XCH     A,[HL-]
0208    RD    FE46    08
0209    WR    FE46    0A
0206          0125              MOV     [HL],A
0212    WR    FE45    08
0207          0126              INC     SW
0213    RD    FE20    03
0214    WR    FE20    04
                          MODULE01\INCI:
0211          0128              INC     I
0217    RD    FE21    03
0218    WR    FE21    04
0216          012A              BR      $COMP
                          MODULE01\COMP:
0222          0106              MOV     A,N
0225    RD    FE4A    08
0224          0108              CMP     A,I
0228    RD    FE21    04
0227          010A              BNZ     $CONT
                          MODULE01\CONT:
0232          0114              BR      $12CH
0236          012C              MOV     A,I
0239    RD    FE21    04
0238          012E              MOV     R0,#0H
0241          0130              BR      $116H
0245          0116              MOVW    RP7,#LIST
0248          0119              XCH     A,R0
0249          011A              ADDW    RP7,RP0
0251          011C              MOV     A,[HL+]
0255    RD    FE46    0A
0252          011D              CMP     A,[HL]
0257    RD    FE47    82
0254          011F              BC      $INCI
                          MODULE01\INCI:
0260          0128              INC     I
0263    RD    FE21    04
0264    WR    FE21    05
0262          012A              BR      $COMP
                          MODULE01\COMP:
0268          0106              MOV     A,N
0271    RD    FE4A    08
0270          0108              CMP     A,I
0274    RD    FE21    05
0273          010A              BNZ     $CONT
                          MODULE01\CONT:
```

```
0278            0114                    BR      $12CH
0282            012C                    MOV     A,I
0285    RD      FE21    05
0284            012E                    MOV     R0,#0H
0287            0130                    BR      $116H
0291            0116                    MOVW    RP7,#LIST
0294            0119                    XCH     A,R0
0295            011A                    ADDW    RP7,RP0
0297            011C                    MOV     A,[HL+]
0301    RD      FE47    82
0298            011D                    CMP     A,[HL]
0303    RD      FE48    0A
0300            011F                    BC      $INCI
0304            0121                    BZ      $INCI
0306            0123                    XCH     A,[HL-]
0310    RD      FE48    0A
0311    WR      FE48    82
0308            0125                    MOV     [HL],A
0314    WR      FE47    0A
0309            0126                    INC     SW
0315    RD      FE20    04
0316    WR      FE20    05
                        MODULE01\INCI:
0313            0128                    INC     I
0319    RD      FE21    05
0320    WR      FE21    06
0318            012A                    BR      $COMP
                        MODULE01\COMP:
0324            0106                    MOV     A,N
0327    RD      FE4A    08
0326            0108                    CMP     A,I
0330    RD      FE21    06
0329            010A                    BNZ     $CONT
                        MODULE01\CONT:
0334            0114                    BR      $12CH
0338            012C                    MOV     A,I
0341    RD      FE21    06
0340            012E                    MOV     R0,#0H
0343            0130                    BR      $116H
0347            0116                    MOVW    RP7,#LIST
0350            0119                    XCH     A,R0
0351            011A                    ADDW    RP7,RP0
0353            011C                    MOV     A,[HL+]
0357    RD      FE48    82
0354            011D                    CMP     A,[HL]
0359    RD      FE49    04
0356            011F                    BC      $INCI
0360            0121                    BZ      $INCI
0362            0123                    XCH     A,[HL-]
0366    RD      FE49    04
0367    WR      FE49    82
0364            0125                    MOV     [HL],A
0370    WR      FE48    04
0365            0126                    INC     SW
0371    RD      FE20    05
0372    WR      FE20    06
                        MODULE01\INCI:
0369            0128                    INC     I
0375    RD      FE21    06
0376    WR      FE21    07
0374            012A                    BR      $COMP
                        MODULE01\COMP:
0380            0106                    MOV     A,N
0383    RD      FE4A    08
0382            0108                    CMP     A,I
```

```
0386    RD    FE21    07
0385          010A                BNZ     $CONT
                            MODULE01\CONT:
0390          0114                BR      $12CH
0394          012C                MOV     A,I
0397    RD    FE21    07
0396          012E                MOV     R0,#0H
0399          0130                BR      $116H
0403          0116                MOVW    RP7,#LIST
0406          0119                XCH     A,R0
0407          011A                ADDW    RP7,RP0
0409          011C                MOV     A,[HL+]
0413    RD    FE49    82
0410          011D                CMP     A,[HL]
0415    RD    FE4A    08
0412          011F                BC      $INCI
0416          0121                BZ      $INCI
0418          0123                XCH     A,[HL-]
0422    RD    FE4A    08
0423    WR    FE4A    82
0420          0125                MOV     [HL],A
0426    WR    FE49    08
0421          0126                INC     SW
0427    RD    FE20    06
0428    WR    FE20    07
                            MODULE01\INCI:
0425          0128                INC     I
0431    RD    FE21    07
0432    WR    FE21    08
0430          012A                BR      $COMP
                            MODULE01\COMP:
0436          0106                MOV     A,N
0439    RD    FE4A    82
0438          0108                CMP     A,I
0442    RD    FE21    08
1>DAS 106,109 ↵
  Addr  Object              Mnemonic
                            ORG     MODULE01\COMP
                        MODULE01\COMP:
  0106  20 4A               MOV     A,N
  0108  9F 21               CMP     A,I
                            END
1>ASM 106 ↵
  0106  MODULE01\COMP: ↵
  0106                      MOV     A,I
                    = BR $139 ↵
        14 31
  0108                      CMP     A,I
                    = END ↵
1>ASM 139 ↵
  0139                      NOP
                    = MOV A,N ↵
        20 4A
  013B                      NOP
                    = DEC R1 ↵
        C9
  013C                      NOP
                    = CMP A,I ↵
        9F 21
  013E                      NOP
                    = BNZ  $MODULE01\CONT ↵
        80 D4
  0140                      NOP
                    = BR $10C ↵
        14 CA
```

```
  0142                    NOP
                  = END )
1>ASM MODULE01\SORT )
  0100  MODULE01\SORT:
  0100                    MOV     SW,#1H
                  = MOV SW,#0H )
        3A 20 00
  0103                    MOV     I,#0H
                  = END )
1>SYM C LIST 0FE22 )
1>SYM C N 0FE2A )
1>SYM E STACK )
1>MEM C 117 )
  0117  42 22 )
  0118  FE . )
1>MEM C 13A )
  013A  4A 2A )
  013B  C9 . )
1>DAS 100,141 )
  Addr  Object            Mnemonic
                          ORG     MODULE01\SORT
                        MODULE01\SORT:
  0100  3A 20 00          MOV     SW,#0H
  0103  3A 21 00          MOV     I,#0H
                        MODULE01\COMP:
  0106  14 31             BR      $139H
  0108  9F 21             CMP     A,I
  010A  80 08             BNZ     $CONT
  010C  14 24             BR      $132H
  010E  00                NOP
                        MODULE01\STOP:
  010F  00                NOP
  0110  00                NOP
  0111  00                NOP
  0112  14 FB             BR      $STOP
                        MODULE01\CONT:
  0114  14 16             BR      $12CH
  0116  67 22 FE          MOVW    RP7,#LIST
  0119  D8                XCH     A,R0
  011A  88 E8             ADDW    RP7,RP0
  011C  59                MOV     A,[HL+]
  011D  16 5F             CMP     A,[HL]
  011F  83 07             BC      $INCI
  0121  81 05             BZ      $INCI
  0123  16 34             XCH     A,[HL-]
  0125  55                MOV     [HL],A
  0126  26 20             INC     SW
                        MODULE01\INCI:
  0128  26 21             INC     I
  012A  14 DA             BR      $COMP
  012C  20 21             MOV     A,I
  012E  B8 00             MOV     R0,#0H
  0130  14 E4             BR      $116H
  0132  6F 20 00          CMP     SW,#0H
  0135  80 C9             BNZ     $SORT
  0137  14 D6             BR      $STOP
  0139  20 2A             MOV     A,N
  013B  C9                DEC     R1
  013C  9F 21             CMP     A,I
  013E  80 D4             BNZ     $CONT
  0140  14 CA             BR      $10CH
                          END
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 )
1>BRA A=MODULE01\STOP C=0P )
1>RUN B 100 )
```

```
 User-system Vcc-ON     Emulation start at 0100
 Standard break    terminated
  PC    SP  PSW: RBS2 RBS1 RBS0 IE    S    Z   RSS  AC    UF   P/V  SUB  CY
 0112  FE80       0    0    0    0    0    1    0    0     1    0    1    0
   R0   R1   R2   R3   R4   R5   R6   R7        RP4      RP5      RP6      RP7
    X    A    C    B                             VP       UP       DE       HL
   06   07   CB   F7   FF   FF   FF   FF       F6FF     FFFF     FFFF     FE29
 One step emulation standby ESC キー 入力
1>MEM D LIST,LIST+7 ↵
 FE22         03 04 04 05 08 0A 0A 82                        ........
1>MEM D N,N ↵
 FE2A                                 08                          .
1>MEM D SW,I ↵
 FE20   00 07                                                  ..
1>SAV SORT01.HEX 100,141 ↵
 object save complete
1>VRY SORT01.HEX ↵
 object verify complete
1>
```

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］　ＩＥ－７８３１０Ａ－Ｒ　ユーザーズ・マニュアル　ソフトウエア編
（ＥＥＵ－６３７Ａ（第２版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
そ　の　他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは

NEC 販売員，特約店販売員，NEC 半応技本部員，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 半導体応用技術本部インフォメーションセンター
FAX：(044)548-7900

キリトリ

お問い合わせは，最寄りのNECへ

本社	〒108-01 東京都港区芝五丁目7番1号 (NEC本社ビル)	
コンシューマ半導体販売事業部 OA半導体販売事業部 インダストリ半導体販売事業部	〒108-01 東京都港区芝五丁目7番1号 (NEC本社ビル)	東京 (03)3454-1111
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋(052)242-2755
関西支社 半導体販売部	〒540 大阪市中央区城見一丁目4番24号 (NEC関西ビル)	大阪 (06)945-3178 大阪 (06)945-3200 大阪 (06)945-3208

北海道支社	札幌	(011)231-0161	立川支社	立川	(0425)26-5981
東北支社	仙台	(022)261-5511	千葉支社	千葉	(043)238-8116
岩手支店	盛岡	(0196)51-4344	静岡支社	静岡	(054)255-2211
山形支店	山形	(0236)23-5511	沼津支店	沼津	(0559)63-4455
郡山支店	郡山	(0249)23-5511	浜松支店	浜松	(053)452-2711
いわき支店	いわき	(0246)21-5511	北陸支社	金沢	(0762)23-1621
長岡支店	長岡	(0258)36-2155	福井支店	福井	(0776)22-1866
水戸支店	水戸	(0292)26-1717	富山支店	富山	(0764)31-8461
神奈川支社	横浜	(045)324-5511	京都支社	京都	(075)344-7824
群馬支店	高崎	(0273)26-1255	神戸支社	神戸	(078)332-3311
太田支店	太田	(0276)46-4011	中国支社	広島	(082)242-5504
宇都宮支店	宇都宮	(0286)21-2281	鳥取支店	鳥取	(0857)27-5311
小山支店	小山	(0285)24-5011	岡山支店	岡山	(086)225-4455
長野支社	長野	(0262)35-1444	四国支社	高松	(0878)36-1200
松本支店	松本	(0263)35-1666	新居浜支店	新居浜	(0897)32-5001
上諏訪支店	諏訪	(0266)53-5350	松山支店	松山	(0899)45-4111
甲府支店	甲府	(0552)24-4141	九州支社	福岡	(092)271-7700
埼玉支社	大宮	(048)641-1411	北九州支店	北九州	(093)541-2887

(技術お問い合わせ先)

半導体応用技術本部 マイクロコンピュータ技術部	〒210 川崎市川崎区駅前本町15番5号 (十五番館)	川崎	(044)246-3921
半導体応用技術本部 中部応用システム技術部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2762
半導体応用技術本部 西日本応用システム技術部	〒540 大阪市中央区城見一丁目4番24号 (NEC関西ビル)	大阪	(06)945-3383

半導体応用技術本部
インフォメーションセンター
FAX(044)548-7900
(FAXで対応させていただいております)