お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

ユーザーズ・マニュアル

RENESAS

保守／廃止

IE-78130-R

ソフトウエア編

資料番号　EEU-648（第1版）
発行年月　May 1991 P

ユーザーズ・マニュアル

NEC
電子デバイス

保守／廃止

IE-78130-R

ソフトウエア編

保守／廃止

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。

したがって，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。

ユーザーズ・マニュアルに従って正しく取り扱いをしてください。

保守／廃止

目　次

保守／廃止

－ 付 録 －

保守／廃止

第１章　概　説

ＩＥ－７８１３０－Ｒは、μＰＤ７８１３Ｘを用いたハードウェア、および、ソフトウェ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－７８１３０－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。

本取扱説明書は、ハードウェア編とソフトウェア編の２部構成になっています。

ハードウェア編

第１章　概　説

第２章　機能概要

第３章　設　置

第４章　システムの構成方法

第５章　RS-232-Cインタフェースの機能

第６章　パラレルインタフェースの機能

第７章　ターゲットとの接続方法

付録

ソフトウェア編

第１章　概　説

第２章　機能概要

第３章　基本的な使用方法

第４章　コマンドの説明

第５章　応用方法

付録

保守／廃止

１ － １　設置からディバグまで

ＩＥ－７８１３０－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。

作　業	取扱説明書
スタート	ハードウェア編／ソフトウェア編　第１章　概　説
↓	
開　包	ハードウェア編　第３章　設　置
↓	
設　置	
↓	
周辺装置との接続	ハードウェア編　第４章　システムの構成方法
↓	
ターゲットとの接続	ハードウェア編　第７章　ターゲットとの接続方法
↓	
ディバグ作業	ソフトウェア編　第２章　機能概要 第３章　基本的な使用方法

・ＩＥ－７８１３０－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

ハードウェア編　第２章　機能概要

・ＩＥ－７８１３０－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

ハードウェア編　第５章　RS-232-Cインタフェースの機能

・ＩＥ－７８１３０－Ｒのパラレル・インタフェースの詳細について知りたいとき

ハードウェア編　第６章　パラレル・インタフェースの機能

・ＩＥ－７８１３０－Ｒのコマンドの詳細について知りたいとき

ソフトウェア編　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

ソフトウェア編　第５章　応用方法

１ － ２　取扱説明書の概要

ハードウェア編

第１章　概要

ＩＥ－７８１３０－Ｒのハードウェア取扱説明書の概要を述べています。

第２章　機能概要

ＩＥ－７８１３０－Ｒの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－７８１３０－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－７８１３０－Ｒの開包方法、付属品の接続方法、本体の設定方法について、詳細に説明しています。ＩＥ－７８１３０－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－７８１３０－Ｒと周辺装置（ホスト・マシン、ターミナル、Ｐ－ＲＯＭプログラマ）の接続方法について、詳細に説明しています。

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

ハードウェア編の第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。

保守／廃止

第５章　ＲＳ－２３２－Ｃインタフェースの機能

ＩＥ－７８１３０－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ています。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－７８１３０－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　パラレル･インタフェースの機能

ＩＥ－７８１３０－Ｒのパラレル･インタフェース（チャネル３、チャネル４）の機能を説明しています。

パラレル･インタフェース を使用する際は、必ずこの章を読んで周辺装置と接続してください。

第７章　ターゲットとの接続方法

ＩＥ－７８１３０－Ｒとμ ＰＤ７８１３Ｘ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ています。

ハードウェア編の第４章を読んで周辺装置と接続された後は、必ら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。

保守／廃止

ソフトウェア編

第１章　概要

ＩＥ－７８１３０－Ｒのソフトウェア取扱説明書の概要を述べています。

第２章　機能概要

スタンド・アロン時とシステム・ソフトウェア使用時に分けて、ＩＥ－７８１３０－Ｒの機能を詳細に説明しています。

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ています。したがって、この章をお読みになることで、ＩＥ－７８１３０－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－７８１３０－Ｒでディバグ作業を行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ています。

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、次に各手順で使用するコマンドについて述べています。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ています。

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えるようになります。

第４章　コマンドの説明

ＩＥ－７８１３０－Ｒのすべてのコマンドについて詳細に説明しています。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時に、この章をお読みください。

第５章　応用方法

ＩＥ－７８１３０－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ています。

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いたい場合は、この章をお読みください。

保守／廃止

第２章　機能概要

２ — １　概要

この章では、ＩＥ－７８１３０－Ｒをスタンド・アロンで使用した場合とシステム・ソフトウェアを使用した場合のそれぞれについて、機能の概要を述べています。

この章を読むことにより、ＩＥ－７８１３０－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・ソフトウェアを使用した場合の違いについて説明しています。

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させて説明しています。

２－４では、システム・ソフトウェ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させて説明しています。

注：　これ以降の記述で‘<cr>’で示されるものは、ＣＲ(0DH) の入力、または、ＣＲを示します。

２－２　スタンド・アロン時とシステム・ソフトウェア使用時の違い

ＩＥ－７８１３０－Ｒは、スタンド・アロンで使用した場合でも十分な機能を持っていますが、システム・ソフトウェアを使用した場合の方が、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持っています。

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持っています。

- イニシャライズ機能
- マッピング機能
- 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）
- ＰＧＭモード機能　（ＰＧＭ、ＭＯＤ　コマンド）

システム・ソフトウェ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の他に、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）
- コマンド・ヘルプ機能　（ＨＬＰ　コマンド）
- シンボリック・ディバグ
- 追加されたシンボルのアップ／ダウン・ロード　（ＳＹＭ　コマンド）

・コマンド省略形入力

・子プロセスの実行機能（ＤＯＳコマンド）

・ディバグ環境のアップ／ダウン・ロード（ＳＡＶ／ＬＯＤコマンド）

以上のように、ＩＥ－７８１３０－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持っています。さらに、システム・ソフトウェ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－７８１３０－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

・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）

・ＰＧＭモード機能　（ＰＧＭ、ＭＯＤ　コマンド）

２－３－１　イニシャライズ機能

◆ オン・チップＲＯＭ容量の設定

スタート・アップ時ＥＡ端子をセンスし、$\overline{\text{ＥＡ}}$＝１の場合はμＰＤ７８１３４或はＲＯＭ容量変更版としての使い方なので、オン・チップＲＯＭサイズを設定します。

オン・チップＲＯＭのサイズは、４Ｋ、８Ｋ、１２Ｋ、１６Ｋ、２０Ｋ、２４Ｋ、２８Ｋ、３２Ｋの内から選択し設定します。

$\overline{\text{ＥＡ}}$＝０の場合は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０Ｋに設定されます。

◆ オン・チップＲＡＭ容量の設定

スタート・アップ 時、内部ＲＡＭ容量を設定します。

オン・チップＲＡＭの容量は、１２８、２５６、３８４、５１２、６４０、７６８、８９６、１０２４バイトの内から選択し設定します。

◆ ダウン･ロード･モードの選択

‘ＬＯＤ’コマンドによる　オブジェクト／シンボル／ディバグ環境のダウン･ロードを、パラレル･インタフェースを用いた高速ダウンロードにするか、シリアル･インタフェースを用いたダウン･ロードにするかを以下のスタート･アップ時の設定で選択します。

‘Do you use high speed down load mode ? (Y/N)’

‘Y<cr>’入力はパラレル･インタフェースを選択し、‘N<cr>’入力はシリアル･インタフェースを選択します。

保守／廃止

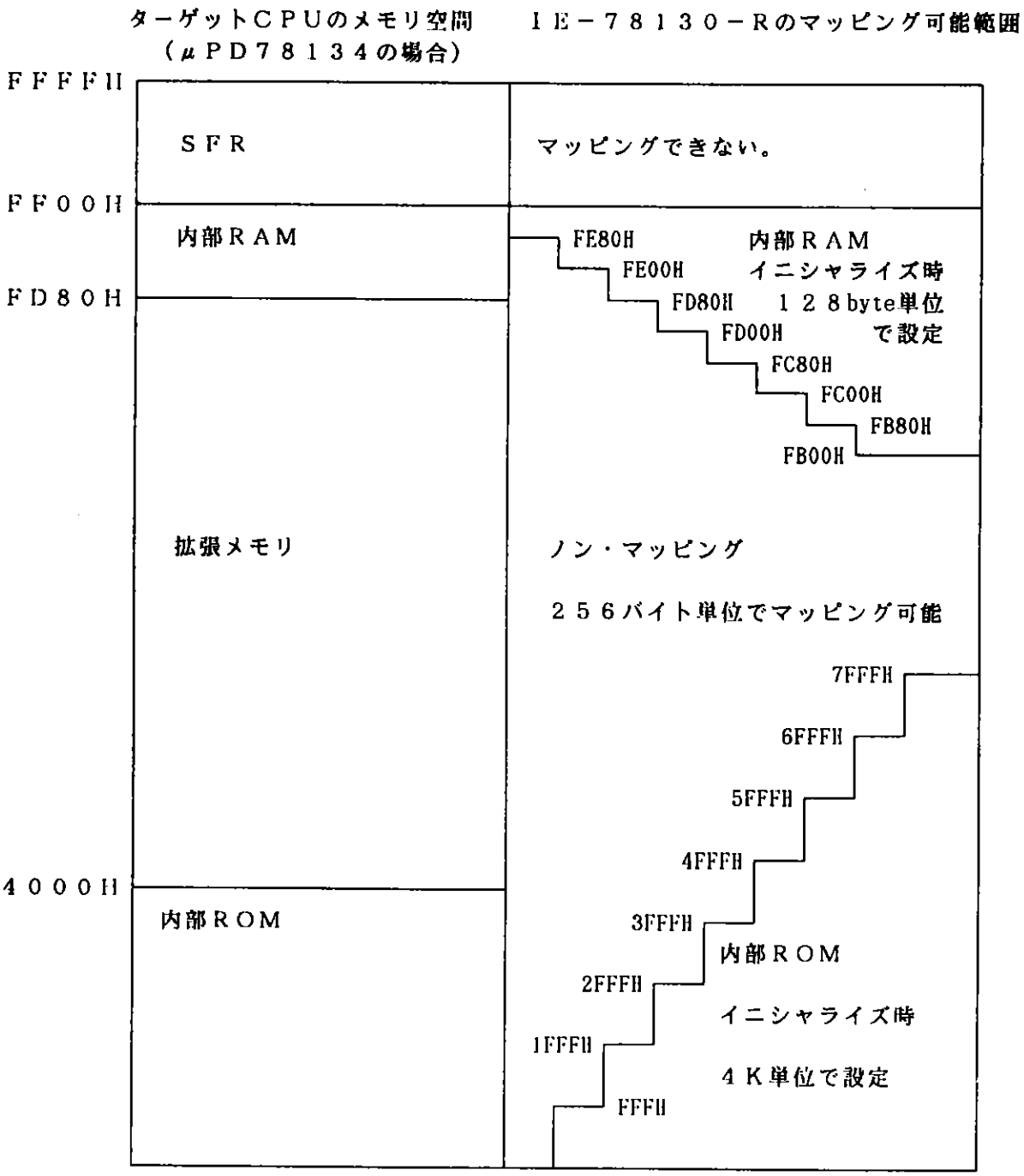
２－３－２　マッピング機能

イニシャライズで設定されたオン・チップＲＯＭ／オン・チップＲＡＭの範囲を内部ＲＯＭ／内部ＲＡＭマッピングに設定し、内部ＲＯＭ、内部ＲＡＭ、ＳＦＲを除くエリアをノン・マッピングに設定します。

ノン・マッピングに設定された範囲は、“ＭＡＰ”コマンドを使用して２５６Ｂｙｔｅ単位で、代替メモリ・マッピング（Ｗ）、ライト・プロテクト付き代替メモリ・マッピング（Ｒ）、ユーザ・メモリ・マッピング（Ｕ）に設定する事が出来ます。ノン・マッピング（Ｋ）指定で、設定したマッピングの解除をする事が出来ます。

次頁に、メモリマップを示します。

・メモリマップ


ターゲットＣＰＵのメモリ空間
（μＰＤ７８１３４の場合）
ＩＥ－７８１３０－Ｒのマッピング可能範囲
ＦＦＦＦＨ
ＳＦＲ
マッピングできない。
ＦＦ００Ｈ
内部ＲＡＭ
FE80H
FE00H
FD80H
FD00H
FC80H
FC00H
FB80H
FB00H
内部ＲＡＭ
イニシャライズ時
１２８byte単位
で設定
ＦＤ８０Ｈ
拡張メモリ
ノン・マッピング
２５６バイト単位でマッピング可能
7FFFH
6FFFH
5FFFH
4FFFH
４０００Ｈ
内部ＲＯＭ
3FFFH
2FFFH
1FFFH
FFFH
内部ＲＯＭ
イニシャライズ時
４Ｋ単位で設定

２ — ３ — ３　メモリ操作機能
（ＡＳＭ、ＤＡＳ、ＭＥＭ、ＬＯＤ、ＳＡＶ、ＶＲＹ　コマンド）

ＩＥ－７８１３０－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、ニーモニックによる変更（ＡＳＭ）、表示（ＤＡＳ）、ＨＥＸ形式オブジェクトのロード（ＬＯＤ）、セーブ（ＳＡＶ）、ベリファイ（ＶＲＹ）などのメモリ操作機能を持っています。

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）

μＰＤ７８１３Ｘ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。

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、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。

特に、ＰＳＷについては、１ビットずつの表示、変更ができます。

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）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[*1]　（ＲＵＮ＿Ｓ）

（Ｄ）ワン・ステップ・エミュレーション　（ＲＵＮ＿Ｔ）

*1　ステップ数は、実行命令数をカウントします。

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４種類のブレーク機能[*2]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・ブレークとスタンダード・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。

*2　２－３－６ブレーク機能参照

保守／廃止

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４種類があります。

・スタンダード・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

・ＲＵＮ＿Ｓにおけるコマンド・ブレーク

　ＲＵＮ＿Ｓコマンドの中で、ユーザが設定できるブレーク機能

・ＲＵＮ＿Ｔにおけるコマンド・ブレーク

　ＲＵＮ＿Ｔ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

　ユーザが禁止することができない、強制的なブレーク機能

　Ａ）マニュアル・ブレーク

　Ｂ）ノン・マップ・エリア・ブレーク

　Ｃ）ライト・プロテクト・ブレーク

　Ｄ）ＳＦＲイリーガル・アクセス・ブレーク

（1）　スタンダード・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）を物理ブレーク・レジスタに設定した後、このレジスタの組み合わせを論理ブレーク・レジスタに設定します。

物理的なブレーク条件としては、以下の条件があります。

・アクセス系ブレーク条件　（ＢＲＡ　コマンド）

・外部信号ブレーク条件　（ＢＲＤ　コマンド）

・インストラクション・カウント・ブレーク条件　（ＢＲＥ　コマンド）

・フェッチ系ブレーク条件　（ＢＲＳ　コマンド）

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　：μＰＤ７８１３Ｘのメモリ・エリアのすべての空間に対して合計５ケ所までのアドレス、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　：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・フェッチを除くすべてのリード／ライト）

以上の９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－７８１３０－Ｒは、ターゲット・プローブの他に、外部信号センス用に８本の外部センス・クリップを持っており、ＴＴＬレベルの信号をトレーサに書き込むことができます。外部センス・クリップＮｏ．１の状態をブレーク要因として、ブレーク要因のレベル（０or１）を指定することができます。

BRE
コマンド

ステップ数　：１～２５５までのステップ数を指定できます。μＰＤ７８１３Ｘ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。

BRS
コマンド

フェッチ・アドレス　：プログラムに対して、シーケンシャル、または、パラレルに合計４つまでのアドレスに対するプログラム・フェッチをブレーク要因として指定できます。

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、ＢＲＳの４種類があり、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・レジスタを組み合わせてブレーク条件をつくることができます。

ＢＲ０～ＢＲ３で１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。２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、ＢＲＳのすべてのブレーク・レジスタを組み合わせて、ブレーク条件をつくることができます。

（２）　ＲＵＮ＿Ｓにおけるブレーク

指定されたステップ数をリアルタイムで実行した後ブレークします。

（３）　ＲＵＮ＿Ｔにおけるブレーク

オペランドで設定されたブレーク条件が成立した時にブレークし、
それまでワン・ステップ実行をします。

（４）　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４種類があります。

Ａ）マニュアル・ブレーク

マニュアル・ブレークには、コンソールからＥＳＣキーが入力された場合の強制ブレークとＳＴＰコマンドによるブレークがあります。

ただし、強制ブレークは、ＲＵＮ＿Ｂ、ＲＵＮ＿Ｓ、ＲＵＮ＿Ｔの各エミュレーションに対し有効となります。ＳＴＰコマンドによるブレークは、、ＲＵＮ＿Ｎのエミュレーションに対し有効となります。

Ｂ）ノン・マップ・エリア・ブレーク

マッピングされていないメモリ・エリアの命令を実行しようとした場合、またはデータのリード／ライトをしようとした場合に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エリアに対してライト・アクセスを行った場合、強制ブレークします。

Ｄ）ＳＦＲイリーガル・アクセス・ブレーク

リード・オンリーのＳＦＲに対するライト・アクセスや、マッピングされていないＳＦＲ空間に対するフェッチ、およびアクセスをした場合、強制ブレークします。

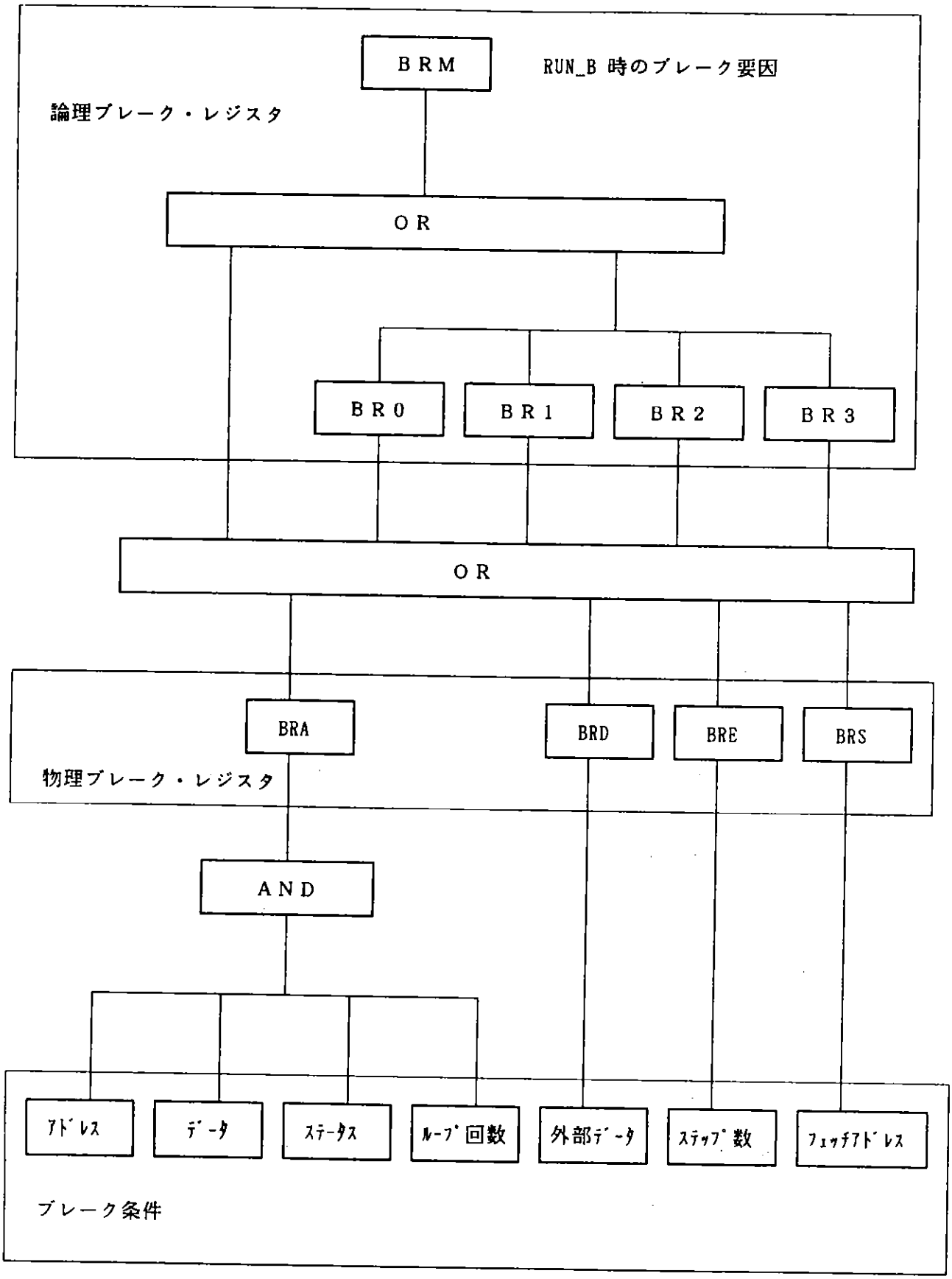


図２－１　　ブレーク・レジスタ構成

２ — ３ — ７　トレース機能（ＴＲ？ コマンド）

※トレース内容

アドレス	：	１６ビット
データ	：	８ビット
外部プローブ	：	８ビット
*1 フレーム・ステータス	：	９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。

*1）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持つかを示すステータスです。

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
*2 Ｍ１	最初のオペコードのフェッチ
*3 ＯＰ	オペランドあるいは２番目以降のオペコード・フェッチ
*3 ＢＲＭ１	ＢＲステータス直後のＭ１
ＶＥＣＴ	ベクタ参照
（Ｍ１）	無効フレーム

*2）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。

*3）ＣＡＬＬＴなどの命令、あるいは割り込みによるベクタ参照のフレーム・ステータスはＶＥＣＴとなります。

トレーサの動作

トレースの開始、終了を以下に示します。

・　ＲＵＮ＿Ｎコマンド入力後トレースを開始し、ＲＵＮ＿Ｂコマンドでブレーク条件に相当する条件が成立した時、あるいはブレークした時点でトレースを終了します。

ＲＵＮ＿Ｂコマンドでブレーク条件に相当する条件が成立した場合、実行中のコマンド入力になり、この時にＴＲＧコマンドによりトレーサを新しく起動させることができます。ＴＲＧコマンドで起動したトレーサは、ＲＵＮ＿Ｎコマンド入力時と同じ条件で終了します。

・　ＲＵＮ＿Ｂコマンド入力後トレースを開始し、ブレークした時点で、トレースを終了します。

・　ＲＵＮ＿Ｓコマンド入力後トレースを開始し、ブレークした時点で、トレースを終了します。

・　ＲＵＮ＿Ｔコマンド入力後、ワン・ステップごとにトレースを開始、実行します。

（1）　トレース条件　（ＴＲＭ，ＤＬＹ　コマンド）

トレース条件として、次の２種類があります。

ａ）トレース・モード

ｂ）トレース・ストップ・ディレイ条件

ａ）トレース・モード

トレース条件によりエミュレーションでのトレースをフェッチ系トレースかアクセス系トレースかの設定をします。

本条件は、通常実行（ＲＵＮ＿Ｎ）、ブレーク付き実行（ＲＵＮ＿Ｂ）、指定ステップ数実行（ＲＵＮ＿Ｓ）でのトレースを決定するもので、トレース付きノン・リアルタイム実行（ＲＵＮ＿Ｔ）、および、ノン・リアルタイムのワン・ステップ実行では、自動的にフェッチ系トレースが設定されます。

トレースには、次の２種類があります。

・フェッチ系トレース
・アクセス系トレース
　┗ ＴＲＭコマンドで設定

ｂ）トレース・ストップ・ディレイ条件

トレース・ストップ・ディレイ条件は、ブレーク・レジスタ・ブレークとして設定されたブレーク要因をトレース・トリガとし、ディレイ・カウンタ設定コマンド（ＤＬＹ）で設定されるフレーム数をトリガ検出以降トレースするための条件です。

本条件が有効となるエミュレーション実行はＲＵＮ＿Ｎ、ＲＵＮ＿Ｂであり、他のＲＵＮ＿Ｓ、ＲＵＮ＿Ｔ、ワン・ステップ実行に於いては無効となります。ただし、ブレーク条件付きリアルタイム・エミュレーション（ＲＵＮ＿Ｂ）では、本条件の設定により、ブレーク条件で設定した条件が生じてから、指定したディレイ分経過した後、ブレークします。

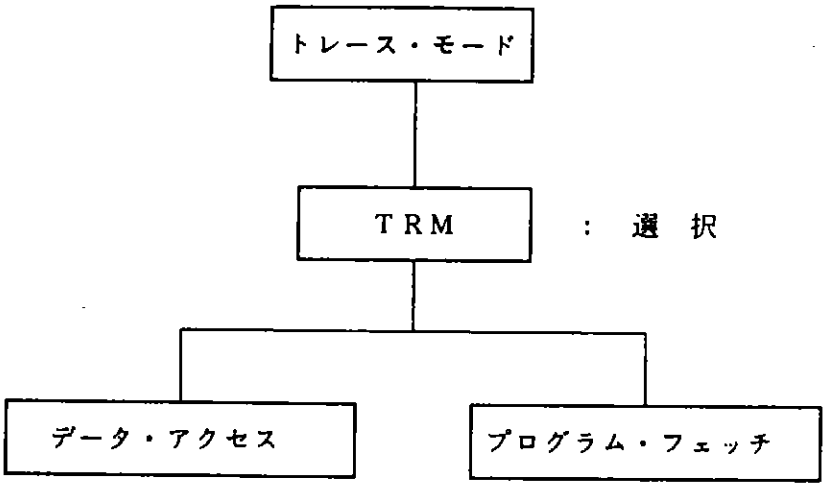


図２－２　トレース・モード設定の構成

（２）　トレース表示　（ＴＲＤ　コマンド）

トレース表示には次の２種類があり、それぞれ設定された検索条件でフレームをサーチし、そのフレームのみ、あるいはそのフレームの前後５行までを表示させることができます。

・フレーム・モード表示

フェッチ系トレース時、アクセス系トレース時可能

・インストラクション・モード表示

フェッチ系トレース時可能、アクセス系トレース時不可能

フレーム・モードは、トレースしたフレームすべてをトレースした順番に表示するモードです。μＰＤ７８１３Ｘが実際にメモリにアクセスした結果、または、実際にユーザ・プログラムをフェッチしたとおりの情報を表示します。

このモードでは、フェッチ系トレースでもアクセス系トレースでもトレース表示は可能であるため、主として実際のμＰＤ７８１３Ｘの動作（実行中でのメモリ・アクセス、または、プログラム・フェッチ）そのものをディバグする目的に用いられます。

一方、インストラクション・モードでは、実行したフェッチ・フレームをインストラクションに変換して表示し、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、このモードでのトレース表示は、フェッチ系トレースの場合のみであり、アクセス系トレースであるリード／ライト・フレームの表示は不可能です。

（３）　トレース・データ検索条件設定　（ＴＲＦコマンド）

トレースしたフレームの中から特定のフレームを検索する為の条件を設定します。

トレース表示の前にここで設定された条件でフレームを検索して、その前に‘！’を付加して表示をするので、ディバグに必要なフレームを容易に探し出すことができます。

なお、トレース・データ検索条件は、‘ＴＲＤ’コマンドで‘＄Ｑ’、‘＄Ｆ’の指定をしなければ無効です。

２ － ３ － ８　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-1000、PG-2000 など、ＮＥＣ製ＰＲＯＭプログラマを接続して、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。

（シリアル・チャネル２の動作状態は、ＰＧＭモードに状態遷移する前に、チャネル２モード設定コマンド（ＭＯＤ）により設定することができます。）

保守／廃止

２－４　システム・ソフトウェアを使用した場合の機能拡張

システム・ソフトウェアを使用することによりスタンド・アロン時のＩＥ－７８１３０－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

・追加されたシンボルのアップ／ダウン・ロード　（ＳＹＭ　コマンド）

・コマンド・ヘルプ機能　（ＨＬＰ　コマンド）

・コマンド省略形入力

・子プロセスの実行機能（ＤＯＳコマンド）

・ディバグ環境のアップ／ダウン・ロード（ＳＡＶ／ＬＯＤコマンド）

２－４－１　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指定されたディレクトリのファイル名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。

カレント・ディレクトリー以外のディレクトリーを参照する場合は、パス名の指定においてディレクトリ名を指定する必要があります。

カレントのディレクトリを参照する場合はディレクトリ名の指定は省略できます。

２－４－２　オート・イニシャライズ機能

システム・ソフトウェア起動時の、セット・アップの内容をファイルに記憶します。

次からは、ファイルの内容で自動的に設定されます。

２－４－３　コマンド／データ・ライン編集機能

コマンド／データ･ライン入力においてカーソル制御キーを使用し、カーソル移動、挿入、置換など行単位の編集をすることができます。

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<cr>”（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び出し、さらに“<cr>”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、“！！<cr>”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は‘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。

また、キー入力されたコマンド・ラインやデータ・ラインをリスト装置に対しても出力することができます。

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テキスト・ファイルから読み込み、自動的にコマンドを実行することができます。

ファイルの作成には、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。

エディタ等によるファイル作成の場合、ファイル中に仮パラメータを指定することができます。仮パラメータは、‘$0’～‘$3’の4つを指定することができます。

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、ファイルが指定できます。

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－Ｐ’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

コマンド・ライン入力、あるいはデータ・ライン入力の数値表現のかわりにシンボルを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・ファイルをロードしておく必要があります。

２－４－９　追加されたシンボルのアップ／ダウン・ロード
（ＳＹＭ　コマンド）

追加シンボル・コマンド（ＳＹＭ＿Ａ）により追加されたシンボルをアップ／ダウン・ロードすることができます。

アップ・ロードは、追加シンボル・セーブ・コマンド（ＳＹＭ＿Ｓ）で行います。

ダウン・ロードは、追加シンボル・ロード・コマンド（ＳＹＭ＿Ｌ）で行います。

保守／廃止

２－４－１０　コマンド省略形

以下に示すコマンドについては、先頭の一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

２－４－１１　コマンド・ヘルプ機能（ＨＬＰコマンド）

ＩＥ－７８１３０－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－７８１３０－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ています。

ＩＥ－７８１３０－Ｒを使用される時は、必ず本章をお読みください。

３－２では、ＩＥ－７８１３０－Ｒでディバグをする時の操作項目と、その手順について説明しています。さらに、各操作項目についてＩＥ－７８１３０－Ｒのどのコマンドを使用したらよいか具体的に説明しています。

３－３では、ＩＥ－７８１３０－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ています。

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出来るようになります。

チャネル２モード設定コマンド（ＭＯＤ）により設定することができます。）

３－２　ディバグ手順とそれに対するコマンド

ＩＥ－７８１３０－Ｒを用いたディバグの一例を、図３－１に示します。

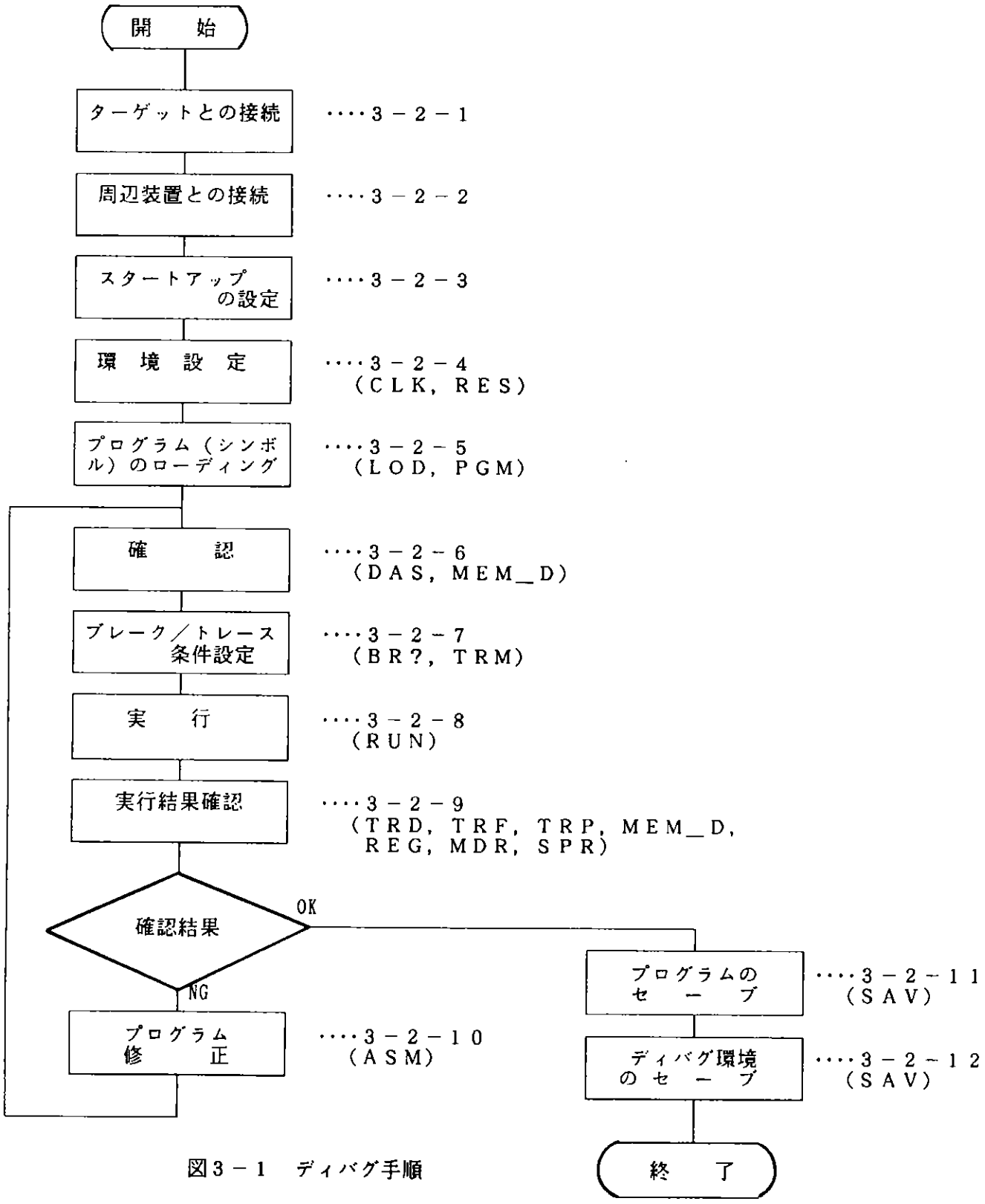


図３－１　ディバグ手順

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ェアの開発状況により、ソフトウェア単体の論理ディバグとハードウェ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。

ハードウェア編の「第６章　ターゲットとの接続方法」に従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－７８１３０－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・ソフトウェアを動作させることにより、ＩＥ－７８１３０－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－７８１３０－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。

ハードウェア編の「第４章　システムの構成方法」に従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－７８１３０－Ｒとホスト・マシンを接続した場合、まずシステム・ソフトウェアを起動します。

するとターゲット・システムの電源を入れるようにメッセージが出て来ますので、ターゲット・システムの電源を入れてから“<cr>”と入力します。　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来ますので、前回のセット・アップ・モードでよければ“N<cr>”を、あらたにセット・アップを行うときは“Y<cr>”を入力してください。“Y<cr>”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来ますので、ターゲット・システムの電源を入れてから“Y<cr>”と入力します。次に内部ＲＯＭ容量を聞いて来ますので、４Ｋ、８Ｋ、１２Ｋ、１６Ｋ、２０Ｋ、２４Ｋ、２８Ｋ、３２Ｋの内より選択し入力します。

次に内部ＲＡＭ容量を聞いてきますので、１２８、２５６、３８４、５１２、６４０、７６８、８９６、１０２４バイトの内より選択し入力します。

次にダウン・ロードをＣＨ４を使用しての高速ダウン・ロードにするか聞いてきますので、高速ダウン・ロードを使用する場合は“Y<cr>”を、使用しない場合は“N<cr>”を入力します。

“N<cr>”を入力した場合はＣＨ１を使用してダウン・ロードを行います。。

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

```
A>IE78130 <cr>
16:14:42 A:IE78130.COM

  IE-78130 CONTROLLER    (PC-9801 SERIES) Vx.x  [Dd Mmm Yy]
       Copyright  (C) 1989 by NEC corporation

  Do you want to use COMMAND LINE EDITOR (Y or N) : Y <cr>
  Window off !
  Select port NO. (1 to 4) : 2<cr>
  IE-78130 for NEW FRAME Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1989 by NEC corporation

  Power on target system (Y/N) Y<cr>
  Create new set up mode (Y or N) : N <cr>
  Internal ROM size (4K,8K,12K,16K,20K,24K,28K,32K) = 16K <cr>
  Internal RAM size (128,256,384,512,640,768,896,1024) = 128 <cr>
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78130-R ? (Y/N)= Y <cr>
     Symbol save area:  00000H-0BFFFH
                        10000H-1BFFFH
                        20000H-2BFFFH
                        30000H-3BFFFH
                        40000H-4BFFFH
                        50000H-5BFFFH
                        60000H-6BFFFH
                        70000H-7BFFFH
2>
```

図３－２－２　システム・モードのスタートアップ

```
  IE-78130 for NEW FRAME Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1989 by NEC Corporation

  Power on target system (Y/N) Y<cr>
  Internal ROM size (4K,8K,12K,16K,20K,24K,28K,32K) = 16K <cr>
  Internal RAM size (128,256,384,512,640,768,896,1024) = 128 <cr>
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78130-R ? (Y/N)= N <cr>
*
```

図３－２－３　スタンド・アロン・モードのスタートアップ

３ － ２ － ４　環境設定（ＣＬＫ，ＲＥＳ）

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。

ユーザ設定のクロックを使用する場合は、エミュレーションボード上のクロック設定用ソケット（ＯＰＣＫ）に必要とするクロックの２倍のクロックを設定しＣＬＫコマンドにより‘Ｕ’（ユーザ）を選択してください。

例えば、１０ＭＨｚでＩＥ－７８１３０－Ｒを動作させる場合は、２０ＭＨｚの発振器を使用します。（ターゲット・システムからのクロック供給はできません。）

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。

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。（ソフトウェア編「第５章　応用方法」をお読みください。）

保守／廃止

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ＨＥＸ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に書き込み、ＩＥ－７８１３０－Ｒと弊社のＰＲＯＭプログラマ（ＰＧ－１０００，ＰＧ－２０００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。

この時は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、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？、ＴＲＭ）

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，ＢＲＥ，ＢＲＳ，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。

実行方法には、ブレーク・ポイント付きのリアルタイム実行やトレース実行など、４種類の方法があります。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，ＴＲＦ，ＴＲＰ，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。トレース・ポインタの操作はＴＲＰコマンドを用います。

実行した結果のメモリ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通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、ニーモニック・レベルでのプログラム・パッチを行うことができます。

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。

３－２－１１　プログラムのセーブ（ＳＡＶ コマンド）

ディバグしたプログラムを、指定したファイルにセーブします。

３－２－１２　ディバグ環境のセーブ（ＳＡＶ コマンド）

ディバグ時の環境をファイルにセーブしておきます。

なお、次回ディバグ作業を再開する時に、このセーブした環境ファイルを再ロードする事により、前回と同様な環境を容易に再現することができます。

３ － ３　操作例

この操作例では、ホスト・マシンにＰＣ－９８０１シリーズを用いて、シリアル・チャネルのチャネル１にＩＥ－７８１３０－Ｒを接続しています。

また、＊ＭＳ－ＤＯＳをスタートした後、☆ＩＥ－７８１３０－Ｒのフロッピィ・ディスクをドライブＡにセットし、システム・ソフトウェ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従って、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<cr>
```

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。

☆ＩＥ－７８１３０－Ｒ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・ソフトウェア・パッケージ

ＩＥ７８１３０．ＣＯＭ	（ＰＣ－９８０１シリーズ用システム・ソフトウェア本体）
ＩＥ７８１３０．ＯＶ１	（省略コマンド・テーブル・ファイル）
ＩＥ７８１３０．ＯＶ２	（ヘルプ・コマンド・テーブル・ファイル）
ＩＥ７８１３０．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）

＊　ＭＳ－ＤＯＳは、マイクロ・ソフトの商標です。

・IE78130 のｼｽﾃﾑ･ｿﾌﾄ使用時の起動方法を説明します。

```
A)IE78130 〈cr〉                    (ｶﾚﾝﾄ･ﾃﾞｨｽｸ上にある IE78130の ｼｽﾃﾑ･ﾌｧｲﾙを実行させます。)
XX:XX:XX A:IE78130.COM
  IE-78130 CONTROLLER     (PC-9801 SERIES) Vx.x  [Dd Mmm Yy]
       Copyright  (C) 1989 by NEC Corporation

  Do you want to use COMMAND LINE EDITOR  (Y or N) : Y  〈cr〉
  Window off !
  Select port NO. (1 to 4) :  ■←ｶｰｿﾙ位置    (接続されているﾎﾟｰﾄ を指定します。)
                              1 〈cr〉

  IE-78130 for NEW FRAME Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1989 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 〈cr〉
  Create new set up mode (Y or N) : ■←ｶｰｿﾙ位置    (ｾｯﾄ･ｱｯﾌﾟ･ﾓｰﾄﾞ 指定のﾒｯｾｰｼﾞ です。)
                                      Y 〈cr〉

  Internal ROM size(4K,8K,12K,16K,20K,24K,28K,32K)= ■←ｶｰｿﾙ位置    (内部ROMｻｲｽﾞ選択ﾒｯｾｰｼﾞ です。)
                                  16K 〈cr〉
  Internal RAM size (128,256,384,512,640,768,896,1024) = 640〈cr〉 ←内部RAMは640ﾊﾞｲﾄで使用します。
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78130-R ? (Y/N)= ■    (IEEE-796ﾊﾞｽ上にﾒﾓﾘ･ﾎﾞｰﾄﾞ が存在する
                                  ｶｰｿﾙ位置──┘       か聞いてきます。)
                                        N 〈cr〉
┌── ﾎﾞｰﾄNo.
1)■←ｶｰｿﾙ位置    (IE78130 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。)
```

・ｽﾀﾝﾄﾞ･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 ﾒｯｾｰｼﾞを出力します。

```
  IE-78130 for NEW FRAME Monitor Vx.x  [Dd Mmm Yy]
    Copyright (C) 1989 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 〈cr〉
  Internal ROM size(4K,8K,12K,16K,20K,24K,28K,32K)= ■←ｶｰｿﾙ位置    (内部ROMｻｲｽﾞ選択ﾒｯｾｰｼﾞ です。)
                                  16K 〈cr〉
  Internal RAM size (128,256,384,512,640,768,896,1024) = 640〈cr〉 ←内部RAMは640ﾊﾞｲﾄで使用します。
  Tracer  initialize
  Breaker initialize
  Do you have Memory Borad on IE-78130-R ? (Y/N)= ■    (1-EEE796ﾊﾞｽ上に ﾒﾓﾘ･ﾎﾞｰﾄﾞが存在するか
                                                      聞いてきます。)
                                        N 〈cr〉
```

* ■←ｶｰｿﾙ位置 (ｽﾀﾝﾄﾞ･ｱﾛﾝ･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。)

・このあとの説明はｼｽﾃﾑ･ﾓｰﾄﾞでの使用例ですが、ﾌﾟﾛﾝﾌﾟﾄの表示ｷｬﾗｸﾀ と' LOD',' SAV',' VRY'ｺﾏﾝﾄﾞ 以外は、同一の方法で入力できます。

IE-78130-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 <cr>      (IE内のｸﾛｯｸを使用します。12MHz固定です。)
```

次にｴﾊﾞﾘｭｴｰｼｮﾝ･ﾁｯﾌﾟ をﾘｾｯﾄします。

```
1>RES <cr>
1>
```

・ﾌﾟﾛｸﾞﾗﾑ のﾃﾞｨﾊﾞｸﾞを通して、ｺﾏﾝﾄﾞの使用方法を説明します。

・ﾌﾟﾛｸﾞﾗﾑ を ﾛｰﾄﾞする ｴﾘｱを ｸﾘｱします。

```
1>MEM F 0,3FFF 00 <cr>          (0番地から3FFF番地までを00にします。)
```

・ﾌﾟﾛｸﾞﾗﾑ をﾌｧｲﾙより ﾛｰﾄﾞします。

```
1>LOD SORT C D<cr>          (ｶﾚﾝﾄ･ﾃﾞｨｽｸ よりSORT.HEX,SORT.SYM という二つのﾌｧｲﾙをﾛｰﾄﾞ
                             します。)
  object load complete      (ｵﾌﾞｼﾞｪｸﾄ･ﾛｰﾄﾞは正常に終了しました。)
  symbol table loading      (ｼﾝﾎﾞﾙ･ﾛｰﾄﾞを開始します。)
  MOD01      load complete  (' MOD01' という ﾓｼﾞｭｰﾙを正常に ﾛｰﾄﾞしました。)
1>
```

・ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ と ｼﾝﾎﾞﾙ･ﾌｧｲﾙを別々に ﾛｰﾄﾞする場合は、次のｺﾏﾝﾄﾞを入力してください。

```
1>LOD SORT.HEX C<cr>          (ｵﾌﾞｼﾞｪｸﾄのみの ﾛｰﾄﾞです。)
            |└ ｵﾌﾞｼﾞｪｸﾄ･ﾛｰﾄﾞ 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 <cr>                  (ｼﾝﾎﾞﾙを削除します。)
1>LOD SORT.SYM S<cr>          (ｼﾝﾎﾞﾙのみの ﾛｰﾄﾞです。)
            |└ ｼﾝﾎﾞﾙ･ﾛｰﾄﾞの指定です。
            └── 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
  MOD01      load complete
1>
```

・ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙを指定します。

```
1>SYM M <cr>                    (ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙに 'MOD01' を指定します。)
          = MOD01 ＼<cr>
1>
```

・ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙを確認します。

```
1>SYM M <cr>
  MOD01 ＼     = <cr>             (変更しないので ﾘﾀｰﾝ･ｷｰを入力します。)
1>
```

・ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ を ﾛｰﾄﾞしたので、ﾒﾓﾘ内容を確認してみます。

```
1>MEM D 100,133 <cr>
       | | |└─ ﾒﾓﾘ･ﾀﾞﾝﾌﾟ終了ｱﾄﾞﾚｽです。
       | |└─── ﾒﾓﾘ･ﾀﾞﾝﾌﾟ開始ｱﾄﾞﾚｽです。
       |└───── MEMｺﾏﾝﾄﾞの ｻﾌﾞ･ｺﾏﾝﾄﾞです。(' D'は ﾒﾓﾘ･ﾀﾞﾝﾌﾟの指定です。)
  0100   3A 80 01 3A 81 00 20 AA 9F 81 80 08 6F 80 00 00     :..:.. .....o...
  0110   00 00 14 FB B8 00 66 A2 FE D8 88 0E 24 68 5D 47     ......f.....$h]G
  0120   16 5F 83 0B 81 09 D8 5D D8 55 D8 4F 55 26 80 26     ._.....].U.OU&.&
  0130   81 14 D3 00                                         ....
  └─┘    └──────────────────────────────────────┘           └──────┘
  ｱﾄﾞﾚｽ                    ﾒﾓﾘ 内容                    ﾒﾓﾘ内容をASCIIｺｰﾄﾞで表示
1>
```

```
1>MEM D 100 <cr>        ( 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 80 01 3A 81 00 20 AA 9F 81 80 08 6F 80 00 00      :..:.. .....o...
  0110   00 00 14 FB B8 00 66 A2 FE D8 88 0E 24 68 5D 47      ......f.....Sh]G
  0120   16 5F 83 0B 81 09 D8 5D D8 55 D8 4F 55 26 80 26      ._.....].U.OU&.&
  0130   81 14 D3 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 <cr>  (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 <cr>      (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>
```

・内部ROM, 0FE40～0FEDFH以外を表示させようとすると ｴﾗｰになります。

```
1>MEM D 4000,6FFF <cr>
  Mapping error
1>
```

・ﾌﾟﾛｸﾞﾗﾑ を ﾛｰﾄﾞした ﾒﾓﾘ内容をﾆｰﾓﾆｯｸで表示するには、次のようなｺﾏﾝﾄﾞを入力します。

```
DAS 100,133                          ( ﾒﾓﾘ内容を逆ｱｾﾝﾌﾞﾙ し、表示するｺﾏﾝﾄﾞです。)
    │  └─  逆ｱｾﾝﾌﾞﾙ 終了ｱﾄﾞﾚｽ
    └────  逆ｱｾﾝﾌﾞﾙ 開始ｱﾄﾞﾚｽ
```

```
1>DAS 100,133 <cr>                   ( 100番地から 133番地までをﾆｰﾓﾆｯｸで表示します。)
  Addr  Object          Mnemonic
                        ORG     MOD01 \SORT
                      MOD01 \SORT:
  0100  3A 80 01        MOV     0FE80H,#1H
  0103  3A 81 00        MOV     0FE81H,#0H
                      MOD01 \COMP:
  0106  20 AA           MOV     A,0FEAAH
  0108  9F 81           CMP     A,0FE81H
  010A  80 08           BNZ     $CONT
  010C  6F 80 00        CMP     0FE80H,#0H
                      MOD01 \STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MOD01 \CONT:
  0114  B8 00           MOV     X,#0H
  0116  66 A2 FE        MOVW    HL,#0FEA2H
  0119  D8              XCH     A,X
  011A  88 0E           ADDW    AX,HL
  011C  24 68           MOVW    HL,AX
  011E  5D              MOV     A,[HL]
  011F  47              INCW    HL
  0120  16 5F           CMP     A,[HL]
  0122  83 0B           BC      $INC1
  0124  81 09           BZ      $INC1
  0126  D8              XCH     A,X
  0127  5D              MOV     A,[HL]
  0128  D8              XCH     A,X
  0129  55              MOV     [HL],A
  012A  D8              XCH     A,X
  012B  4F              DECW    HL
  012C  55              MOV     [HL],A
  012D  26 80           INC     0FE80H
                      MOD01 \INC1:
  012F  26 81           INC     0FE81H
  0131  14 D3           BR      $COMP
  0133  00              NOP
                        END
1>
```

・逆ｱｾﾝﾌﾞﾙ 終了ｱﾄﾞﾚｽを指定しないと、11行表示します。

```
1>DAS 100 <cr>                          ( 100番地より逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ を11行分表示します。)
  Addr  Object          Mnemonic
                        ORG     MOD01 ＼SORT
                      MOD01 ＼SORT:
  0100  3A 80 01        MOV     0FE80H,#1H
  0103  3A 81 00        MOV     0FE81H,#0H
                      MOD01 ＼COMP:
  0106  20 AA           MOV     A,0FEAAH
  0108  9F 81           CMP     A,0FE81H
  010A  80 08           BNZ     $CONT
  010C  6F 80 00        CMP     0FE80H,#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 <cr>                              (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     MOD01 ＼STOP
                      MOD01 ＼STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MOD01 ＼CONT:
  0114  B8 00           MOV     X,#0H
  0116  66 62 FE        MOVW    HL,#0FEA2H
                        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 0FE80<cr>        (ｽｲｯﾁとして使用するFE80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 0FE81 <cr>        (ｲﾝﾃﾞｯｸｽとして使用するFE81番地に Iというｼﾝﾎﾞﾙを定義します。)
1>SYM A STACK 0FEE0 <cr>    (ｽﾀｯｸ･ｴﾘｱのﾍﾞｰｽ･ｱﾄﾞﾚｽFEE0番地に STACKというｼﾝﾎﾞﾙを定義します。)
1>SYM A LIST 0FEA2<cr>      (並べ替えるﾃﾞｰﾀ のﾍﾞｰｽ･ｱﾄﾞﾚｽFEA2番地にLISTというｼﾝﾎﾞﾙを定義
                            します。)
1>SYM A N 0FEAA <cr>        (ﾃﾞｰﾀ 数を表すFEAA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 0FE80,0FEDF 0 <cr>     (ﾃﾞｰﾀ･ｴﾘｱの内容を ｸﾘｱします。)

1>MEM C 0FEA2 <cr>
  FEA2  00 05 <cr>
     |  |  └─  変更ﾃﾞｰﾀ
     |  └────  ﾒﾓﾘ 内容
     └───────  変更ｱﾄﾞﾚｽ
  FEA3  00 03 <cr>
  FEA4  00 04 <cr>
  FEA5  00 0A <cr>
  FEA6  00 08 <cr>
  FEA7  00 82 <cr>
  FEA8  00 0A <cr>
  FEA9  00 04 <cr>
  FEAA  00 08 <cr>
  FEAB  00 . <cr>
1>         └──── 変更終了
```

・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 0FE80,0FEAA <cr>
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   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<cr>          (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。)
        |  └─  FEAA番地に定義したｼﾝﾎﾞﾙ
        └────  FE80番地に定義したｼﾝﾎﾞﾙ
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   00 00 05 03 04 0A 08 82 0A 04 08                    ...........

1>MEM D 0FE80,N <cr>      (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。)
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEAA   00 00 05 03 04 0A 08 82 0A 04 08                    ...........

1>MEM D SW,0FEAA<cr>      (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。)
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   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>BRS P STOP<cr>          (ﾊﾟﾗﾚﾙ･ﾌｪｯﾁ･ﾌﾞﾚｰｸ条件を設定します。)
1>

1>BRS <cr>                (ﾌﾞﾚｰｸ条件を確認します。)
  Parallel
  STOP
1>
```

・次にﾌﾞﾚｰｸ条件の指定をします。

```
1>BRM BRS <cr>              (ﾌﾞﾚｰｸ条件として BRSを指定します。)
1>

1>BRM <cr>                  (ﾌﾞﾚｰｸ条件を確認します。)
  BRS                       (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<cr>
  PC       0000 = 100 <cr>   (ﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀ を 100番地に設定します。)
  SP       FE72 = FEE0<cr>   (ｽﾀｯｸ･ﾎﾟｲﾝﾀをFEE0番地に設定します。)
1>

1>RUN B 100 <cr>            (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。

```
 IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
  0   0   0   0   0    1   0  A2    00    00    00    FFFF    FEA3   010F  FEE0
                                                  次に実行するﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀ ──────┘
```

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           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<cr>                          (SW から Nまでの ﾒﾓﾘ内容を表示します。)
  FE80   03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
  FEA0   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 ﾃﾞｰﾀが並び替わっていませんので、もう一度ﾃﾞｰﾀを設定して動作を確認します。

```
1>MEM C LIST<cr>                  ( LISTｴﾘｱ の ﾃﾞｰﾀを設定します。)
  FEA2  03 05 <cr>
  FEA3  05 03 <cr>
  FEA4  04 <cr>  ┐
  FEA5  0A <cr>  │
  FEA6  08 <cr>  │   ﾃﾞｰﾀ が変化していないので変更しません。
  FEA7  82 <cr>  │
  FEA8  0A <cr>  │
  FEA9  04 <cr>  ┘
  FEAA  00 08 <cr>
  FEAB  08 00 <cr>
  FEAC  00 . <cr>
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<cr>                    (ﾚｼﾞｽﾀを変更するためのｺﾏﾝﾄﾞ です。)
  PC       010F = 100 <cr>        (PCを 100に変更します。)
  SP       FEE0 = . <cr>          (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<cr>                (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
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  A2    00    00    00    FFFF    FEA3   0103  FEE0

  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  A2    00    00    00    FFFF    FEA3   0106  FEE0

  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   0108  FEE0
```

```
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   010A  FEE0


  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   0114  FEE0

  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  00    08    00    00    FFFF    FEA3   0116  FEE0

     terminated
  Frame Address Label  Mnemonic
                MOD01  \CONT:
  0000   0114          MOV     X,#0H
 One step emulation standby<cr>            (ﾜﾝ･ｽﾃｯﾌﾟ 実行ﾓｰﾄﾞ で実行します。)
  Frame Address Label  Mnemonic
  0000   0116          MOVW    HL,#LIST
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  00    08    00    00    FFFF    FEA2   0119  FEE0
 One step emulation standby ESC ｷｰ         (ﾜﾝ･ｽﾃｯﾌﾟ 実行ﾓｰﾄﾞ を終了します。)
1)
```

・次に ﾌﾟﾛｸﾞﾗﾑに ﾊﾟｯﾁをします。

```
1)ASM CONT<cr>                          (CONT･･･114番地へ ﾊﾟｯﾁをします。)
  0114  MOD01 \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     X,#0H     ←114 番地の命令が表示されます。
              = BR $133 <cr>                ← ﾊﾟｯﾁした命令 (133 番地へｼﾞｬﾝﾌﾟさせます。)
        14 1D ←生成された命令のｵﾌﾞｼﾞｪｸﾄ
  0116                MOVW    HL,#LIST  ←116 番地の命令が表示されます。
              = ORG 133H<cr>            (ｱﾄﾞﾚｽを 133番地に変更します。)
  0133                NOP
  |           = MOV A,1 <cr>            (追加する命令です。)
  └─ ｱﾄﾞﾚｽが変更されました
        20 81
  0135                NOP
              = MOV X,#0H <cr>          (114 番地にあった命令です。)
        B8 00
  0137                NOP
              = BR $116 <cr>            (116 番地へｼﾞｬﾝﾌﾟします。)
        14 DD
  0139                NOP
              = END <cr>                (ASMｺﾏﾝﾄﾞ を終了させます。)
1)
```

```
1>DAS 114,116 <cr>          (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     MOD01 \CONT
                              MOD01 \CONT:
  0114  14 1D                   BR      $133H
  0116  66 A2 FE                MOVW    HL,#LIST
                                END
1>DAS 133,138 <cr>
  Addr  Object                  Mnemonic
                                ORG     133H
  0133  20 81                   MOV     A,I
  0135  B8 00                   MOV     X,#0H
  0137  14 DD                   BR      $116H
                                END
1>
```

・ﾊﾟｯﾁ をしましたので、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 <cr>            (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
  SFR illegal access break    terminated      (ﾘｰﾄﾞ･ｵﾝﾘｰ の ﾒﾓﾘをｱｸｾｽしたため実行を停止しました。)
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   1  FF    02    00    00    FFFF    FF02   012A  FEE0
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<cr>
  FE80   73 BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00     s...............
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   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。
```

・再度 ﾌﾟﾛｸﾞﾗﾑに ﾊﾟｯﾁをします。

10C番地で、SW…FE80番地が 0かﾁｪｯｸしているが、比較結果に関係なくSTOP番地にｼﾞｬﾝﾌﾟしているので、その所を変更します。

```
1>ASM 10C <cr>                          (10C 番地 ﾊﾟｯﾁ)
  010C                        CMP     SW,#0H
                          = BR $139 <cr>      (139 番地へｼﾞｬﾝﾌﾟ)
       14 2B
  010E                        NOP
                          = NOP <cr>          (10C 番地が3ﾊﾞｲﾄ命令なので ﾊﾞｲﾄ数を合わせるための命令です。)
       00
  010F  MOD01 \STOP:
```


```
  010F                          NOP
                              = END <cr>


1>ASM 139 <cr>                                  (139 番地から ﾊﾟｯﾁ)
  0139                          NOP
                              = CMP SW,#0H<cr>  (10C 番地にあった命令)
        6F 80 00
  013C                          NOP
                              = BNZ  $CONT<cr>  (SWが 0以外の時CONT番地(114番地) へ
        80 D6                                   ｼﾞｬﾝﾌﾟ します。)
  013E                          NOP
                              = BR $STOP<cr>
        14 CF
  0140                          NOP
                              = END <cr>
1>

1>DAS 10C,10E <cr>                              (10C 番地から 10E番地まで逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ 表示)
  Addr  Object          Mnemonic
                        ORG     10CH
  010C  14 2B           BR      $139H
  010E  00              NOP
                        END
1>DAS 139,13F <cr>                              (139 番地から 13F番地まで逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ 表示)
  Addr  Object          Mnemonic
                        ORG     139H
  0139  6F 80 00        CMP     SW,#0H
  013C  80 D6           BNZ     $CONT
  013E  14 CF           BR      $STOP
                        END
1>

1>DAS CONT<cr>                                  (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場合11行表示します。)
                        ORG     MOD01 \CONT
                      MOD01 \CONT:
  0114  14 1D           BR      $133H
  0116  66 A2 FE        MOVW    HL,#LIST
  0119  D8              XCH     A,X
  011A  88 0E           ADDW    AX,HL
  011C  24 68           MOVW    HL,AX
  011E  5D              MOV     A,[HL]
  011F  47              INCW    HL
                        END
1>DAS <cr>                                      (前回の次のｱﾄﾞﾚｽより逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ 表示)
  Addr  Object          Mnemonic
                        ORG     120H
  0120  16 5F           CMP     A,[HL]
  0122  83 0B           BC      $INC1
  0124  81 09           BZ      $INC1
  0126  D8              XCH     A,X
```

```
  0127  5D                  MOV     A,[HL]
  0128  D8                  XCH     A,X
  0129  55                  MOV     [HL],A
  012A  D8                  XCH     A,X
                            END

1>
1>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 <cr>      (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>BRS P 122 <cr>                          (ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を 122番地に指定します.)
1>RUN B 100 <cr>                          (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.)
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    A2    00    00    FFFF    FEA3   0127  FEE0
  One step emulation standby<cr>       (ﾘﾀｰﾝ･ｷｰ を入力し、次のｽﾃｯﾌﾟを実行します.)
   Frame Address Label  Mnemonic
   0000   0127          MOV     A,[HL]
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA3   0128  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   0128          XCH     A,X
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  03    05    00    00    FFFF    FEA3   0129  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   0129          MOV     [HL],A
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  03    05    00    00    FFFF    FEA3   012A  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   012A          XCH     A,X
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA3   012B  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   012B          DECW    HL
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012C  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   012C          MOV     [HL],A
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012D  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
   0000   012D          INC     SW
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012F  FEE0
```

```
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
                 MOD01  \INC1:
  0000   012F          INC     1
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0131  FEE0
  One step emulation standby ESCｷｰ
1>MEM D LIST,LIST+7 <cr>                (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します.)
  FEA0          03 05 04 0A 08 82 0A 04                          ........
              並び替わりました.
1>MEM D 1,1 <cr>                        (1ｴﾘｱの ﾒﾓﾘ内容表示を確認します.)
  FE80        01                                                  .
1>


1>RUN T ,1<cr>                          (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 013
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR      $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0106  FEE0
  One step emulation standby<cr>
   Frame Address Label  Mnemonic
                 MOD01  \COMP:
   0000   0106          MOV     A,N
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    08    00    00    FFFF    FEA2   0108  FEE0
  One step emulation standby ESCｷｰ
1>BRS P INC1<cr>                        (新たなﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を指定します.)
1>RUN B <cr>                            (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0106  FEE0
  One step emulation standby ESCｷｰ
1>

1>MEM D 1,1 <cr>                        (1ｴﾘｱの ﾒﾓﾘ内容を確認します.)
  FE80        02                                                  .
1>MEM D LIST,LIST+7 <cr>                (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します.)
  FEA0          03 04 05 0A 08 82 0A 04                          ........
              並び替わりました。
1>RUN T ,1<cr>                          (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 Address Label  Mnemonic
                 MOD01  \COMP:
   0000   0106          MOV     A,N
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    08    00    00    FFFF    FEA3   0108  FEE0
  One step emulation standby ESCｷｰ
```

```
1>

1>RUN B <cr>                                (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   1  A4    05    00    00    FFFF    FEA5   0106  FEE0
  One step emulation standby ESCｷｰ入力
1>MEM D I,I <cr>                            (Iｴﾘｱの ﾒﾓﾘ内容を確認します。)
  FE80      03
1>

1>MEM D LIST,LIST+7 <cr>                    (LISTｴﾘｱ の ﾒﾓﾘ内容を表示します。)
  FEA0          03 04 05 0A 08 82 0A 04                              ........
1>


1>BRS P STOP 139<cr>                        (ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ をﾊﾟﾗﾚﾙで 2箇所設定します。)
1>RUN B <cr>                                (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
  SFR illegal access break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  FF    02    00    00    FFFF    FF02   012A  FEE0
1>MEM D LIST,N<cr>                          (LIST～N までの ﾒﾓﾘ内容を表示します。)
  FEA0          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>MEM F LIST,N 5,3,4,0A,8,82,0A,4,8 <cr> (LIST～N の範囲の ﾒﾓﾘ内容を初期化します。)
1>DAS 12D,132 <cr>                          (ｱﾄﾞﾚｽ確認のため 12D番地～ 132番地の逆ｱｾﾝﾌﾞﾙ･
  Addr  Object              Mnemonic                                  ﾘｽﾄ を表示します。)
                            ORG     12DH
  012D  26 40               INC     SW
                        MOD01 \INC1:
  012F  26 41               INC     I
  0131  14 D3               BR      $COMP
                            END
1>

1>BRS P 12D <cr>                            (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 <cr>                            (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0131  FEE0
  One step emulation standby ESCｷｰ入力
1>

1>MEM D I,I <cr>                            (Iｴﾘｱの ﾒﾓﾘ内容確認)
  FE80      01
1>

1>MEM D LIST,LIST+7 <cr>                    (LISTｴﾘｱ の ﾒﾓﾘ内容表示)
```

```
  FEA0          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<cr>                          (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 0131
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR       $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0106  FEE0
  One step emulation standby ESCｷｰ
1>

1>RUN B <cr>                            (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0131  FEE0
  One step emulation standby ESCｷｰ
1>


1>MEM D 1,1 <cr>                        (1ｴﾘｱの ﾒﾓﾘ内容を確認します.)
  FE80      02
1>

1>MEM D LIST,LIST+7 <cr>                (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します.)
  FEA0          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<cr>                          (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 0131
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR       $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0106  FEE0
  One step emulation standby ESCｷｰ
1>

1>RUN B <cr>                            (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  0A    08    00    00    FFFF    FEA5   0131  FEE0
  One step emulation standby ESCｷｰ
1>MEM D 1,1 <cr>                        (1ｴﾘｱの ﾒﾓﾘ内容を確認します.)
  FE80      04
1>

1>MEM D LIST,LIST+7 <cr>                (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します.)
  FEA0          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 <cr>                    (SWｴﾘｱ の ﾒﾓﾘ内容を確認します.)
  FE80   04
1>
```

・ﾌﾟﾛｸﾞﾗﾑ に ﾊﾟｯﾁをします.

```
1>ASM 13C <cr>                        (13C 番地へ ﾊﾟｯﾁをします.)
  013C                          BNZ     $CONT
                        = BNZ  $SORT<cr>      (ｼﾞｬﾝﾌﾟ先をCONTからSORTに変更します.)
        80 C2
  013E                          BR      $STOP
                        = END <cr>
1>DAS 13C,13D <cr>                    (ﾊﾟｯﾁ を逆ｱｾﾝﾌﾞﾙ･ﾘｽﾄ にて確認します.)
  Addr  Object              Mnemonic
                            ORG     13CH
  013C  80 C2               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 <cr>    (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、ﾄﾚｰｽ条件の設定をします.

```
1>TRM D <cr>                          (ﾄﾚｰｽ条件をﾃﾞｰﾀ･ｱｸｾｽ条件に設定しました.)
1>
```

・新たにﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ 、および、ﾙｰﾌﾟ 回数を設定します.

```
1>BRA A=N C=W <cr>                    (N･･･FEAA番地へ ﾗｲﾄしたらﾌﾞﾚｰｸします.)
1>
```

・ﾌﾞﾚｰｸ条件としてｱｸｾｽ系ﾌﾞﾚｰｸ条件を指定します.

```
1>BRM BRA <cr>
1>
```

```
1>RUN B 100 <cr>                      (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  08    82    00    00    FFFF    FEA9   012E  FEE0
  One step emulation standby ESCｷｰ
1>MEM D N,N <cr>                      (Nｴﾘｱの ﾒﾓﾘ内容を確認します.)
  FEA0                                 82
                                      LISTｴﾘｱ の値が入っています.
```

```
1>MEM D LIST,N<cr>                    (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します。)
  FEA0          03 04 05 08 0A 0A 04 82 82                    .........
1>          この範囲が並び替わりました。
```

・ﾄﾚｰｽ･ﾎﾟｲﾝﾀ を最新の ﾗｲﾝに置きます。

```
1>TRP N <cr>
1>
```

・ﾄﾚｰｽ･ﾎﾟｲﾝﾀ を 9ﾗｲﾝ 前へもどします。

```
1>TRP -9T <cr>
1>
```

・ﾄﾚｰｽ結果を表示してみます。

```
  TRD F                               (ﾄﾚｰｽされた ﾃﾞｰﾀを表示するｺﾏﾝﾄﾞです。)
      │└──  省略した場合は、11行分のみ表示します。
      └───  ﾄﾚｰｽ･ﾓｰﾄﾞをﾌﾚｰﾑ･ﾓｰﾄﾞに指定します。
```

```
1>TRD F <cr>                          (ﾄﾚｰｽ･ﾃﾞｰﾀをﾌﾚｰﾑ･ﾓｰﾄﾞで11行出力します。)
   Frame  Status  Address Data    7--EX--0
   0073     WR     0FE80   06     00000000
   0074     RD     0FE81   06     00000000
   0075     WR     0FE81   07     00000000
   0076     RD     0FEAA   08     00000000
   0077     RD     0FE81   07     00000000
   0078     RD     0FE81   07     00000000
   0079     RD     0FEA9   82     00000000
   0080     RD     0FEAA   08     00000000
   0081     RD     0FEAA   08     00000000
  T0082     WR     0FEAA   82     00000000
   Total frame = 0083   (N/O/T/+/cr/Frame No./.) ? .<cr>
1>
```

・N の ﾃﾞｰﾀが変化するのを防ぐため ﾊﾟｯﾁをします。

```
1>DAS 106,109 <cr>    (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     MOD01 \COMP
                          MOD01 \COMP:
  0106  20 AA               MOV     A,N
  0108  9F 81               CMP     A,1
                            END
1>ASM 106 <cr>                        (106番地に ﾊﾟｯﾁをします。)
  0106  MOD01 \COMP:
  0106                    MOV     A,N
              = BR $140 <cr>          (140 番地にｼﾞｬﾝﾌﾟします。)
```

```
        14 38
  0108                  CMP     A,I
                = END <cr>
1>ASM 140 <cr>                                  (140 番地に ﾊﾟｯﾁします。)
  0140                  NOP
                = MOV A,N <cr>                  (106 番地にあった命令です。)
        20 AA
  0142                  NOP
                = DEC A <cr>                    (追加する命令です。Aの値を 1ﾃﾞｸﾘﾒﾝﾄします。)
        C9
  0143                  NOP
                = CMP A,I <cr>                  (108 番地にある命令です。)
        9F 81
  0145                  NOP
                = BNZ  $CONT<cr>                (10A 番地にある命令です。)
        80 CD
  0147                  NOP
                = BR $10C <cr>                  (10C 番地へｼﾞｬﾝﾌﾟします。)
        14 C3
  0149                  NOP
                = END <cr>
1>
```

```
1>ASM SORT<cr>
  0100  MOD01 \SORT:
  0100                  MOV     SW,#1H
                = MOV SW,#0H<cr>                (SWの初期値を 00 にします。)
        3A 80 00
  0103                  MOV     I,#0H
                = END <cr>
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 0FE82<cr>                          (LISTのｼﾝﾎﾞﾙ値をFE82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 0FE8A <cr>                            (N のｼﾝﾎﾞﾙ値をFE8Aに変更します。)
1>SYM E STACK <cr>                              (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 <cr>                                (117 番地の ﾒﾓﾘ内容を変更します。)
  0117  A2 82 <cr>                              (A2から82に ﾒﾓﾘ内容を変更します。)
  0118  FE .<cr>                                (ﾒﾓﾘ 変更終了ｷｬﾗｸﾀ を入力しました。)
```

```
!>MEM C 141 <cr>                    (141 番地の ﾒﾓﾘ内容を変更します.)
  0141  AA 8A <cr>                  (AAから8Aに ﾒﾓﾘ内容を変更します.)
  0142  C9 .<cr>
!>
```

```
!>DAS 100,149 <cr>                  (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     MOD01 \SORT
                      MOD01 \SORT:
  0100  3A 80 00        MOV     SW,#0H
  0103  3A 81 00        MOV     I,#0H
                      MOD01 \COMP:
  0106  14 38           BR      $140H
  0108  9F 81           CMP     A,I
  010A  80 08           BNZ     $CONT
  010C  14 2B           BR      $139H
  010E  00              NOP
                      MOD01 \STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MOD01 \CONT:
  0114  14 1D           BR      $133H
  0116  66 82 FE        MOVW    HL,#LIST
  0119  D8              XCH     A,X
  011A  88 0E           ADDW    AX,HL
  011C  24 68           MOVW    HL,AX
  011E  59              MOV     A,[HL]
  011F  47              INCW    HL
  0120  16 5F           CMP     A,[HL]
  0122  83 0B           BC      $INCI
  0124  81 09           BZ      $INCI
  0126  D8              XCH     A,X
  0127  5D              MOV     A,[HL]
  0128  D8              XCH     A,X
  0129  55              MOV     [HL],A
  012A  D8              XCH     A,X
  012B  4F              DECW    HL
  012C  55              MOV     [HL],A
  012D  26 80           INC     SW
                      MOD01 \INCI:
  012F  26 81           INC     I
  0131  14 D3           BR      $COMP
  0133  20 81           MOV     A,I
  0135  B8 00           MOV     X,#0H
  0137  14 DD           BR      $116H
  0139  6F 80 00        CMP     SW,#0H
  013C  80 C2           BNZ     $SORT
  013E  14 CF           BR      $STOP
  0140  20 8A           MOV     A,N
```

```
  0142  C9              DEC     A
  0143  9F 81           CMP     A,I
  0145  80 CD           BNZ     $CONT
  0147  14 C3           BR      $10CH
  0149  00              NOP
                        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 <cr>     (LIST～N の ﾒﾓﾘ内容を初期化します。)
1>BRM BRS <cr>                               (ﾌﾞﾚｰｸ条件をﾌｪｯﾁ系ﾌﾞﾚｰｸ条件に指定します。)
1>BRS P STOP<cr>                             (ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ をSTOPに指定します。)
1>RUN B 100 <cr>                             (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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   1   0   0   0    1   0  88    07    00    00    FFFF    FE89   010F  FEE0
  One step emulation standby ESCｷｰ
1>MEM D LIST,LIST+7 <cr>                     (LISTｴﾘｱ の ﾒﾓﾘ内容を確認します。)
  FE80          03 04 04 05 08 0A 0A 82                          ........
              昇順に並び替わりました。
1>MEM D N,N <cr>                             (Nｴﾘｱの ﾒﾓﾘ内容を確認します。)
  FE80                                  08
1>MEM D SW,I<cr>                             (SW～N の ﾒﾓﾘ内容を確認します。)
  FE80    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 100,149                (100 番地から 149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 100,149 C<cr>
  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<cr>                  (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>
```

・ディバグ環境をセーブしておきます。

```
1>SAV SORT01.DBG D<cr>
  debag condition save complete
1>
```

第４章　コマンドの説明

４ — １　概要

この章では、ＩＥ－７８１３０－Ｒのコマンドの詳細について述べています。

本章は、ソフトウェア編「第３章　基本的な使用方法」をお読みになって、一通り

ＩＥ－７８１３０－Ｒをお使いになってからお読みください。

４－２　では、ＩＥ－７８１３０－Ｒのコマンド入力の方法について説明しています。

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ています。

４－４　では、コマンド全体に共通することについて説明しています。

４－５　では、コマンド一つ一つに対する詳細な説明をしています。

４－６　では、エラー・メッセージについて説明しています。

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ています。

４－２　コマンド入力方法

４－２－１　概要

４－２－２　制御キー

４－２－３　スタンド・アロン時のコマンド入力方法

４－２－４　システム・ソフトウェア使用時のコマンド入力方法

４－３　数値、シンボル、式の記述仕様

４－３－１　数値表現

４－３－２　シンボル表現

４－３－３　式表現

４－３－４　特殊数値表現

４－４　コマンド一覧

保守／廃止

４ － ２　コマンド入力方法

４ － ２ － １　概要

ＩＥ－７８１３０－Ｒでは、下記の２種類の状態で、コマンドの入力方法が若干異なります。

① スタンド・アロン時

② システム・ソフトウェアを動作させた時

②、①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、制御キーによる一文字削除、一行削除が可能となっています。

②では、①に加えて次の機能が可能になります。

・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

・行単位の編集機能

４ — ２ — ２　制御キー

ＩＥ－７８１３０－Ｒでは、以下の制御キーを使用することができます。

制御キー	機　　能	ｽﾀﾝﾄﾞ・ｱﾛﾝ・ｱﾛﾝ・ﾓｰﾄﾞ	ｼｽﾃﾑ・ﾓｰﾄﾞ
↑Ａ	インサート・モード・トグル・スイッチ	×	○
↑Ｃ	システム・ソフトウェア強制終了	×	○
↑Ｈ	一文字削除	○	○
↑Ｉ	スペースと同じ	○	○
↑Ｊ	行入力の終了	○	○
↑Ｋ	ＳＴＲコマンドの強制終了	×	○
↑Ｌ	ＳＴＲコマンドの一時停止／再開	×	○
↑Ｍ	行入力の終了	○	○
↑Ｏ	ＣＯＭコマンドの出力スイッチ	×	○
↑Ｐ	ＬＳＴコマンドの出力スイッチ	×	○
↑Ｑ	↑Ｓによる一時停止の解除	○	○
↑Ｓ	一時停止	○	○
↑Ｘ	一行削除	○	○

表４－４　制御キー　（１／２）

制御キー	機　　能	ｽﾀﾝﾄﾞ・ｱﾛﾝ・ｱﾛﾝ・ﾓｰﾄﾞ	ｼｽﾃﾑ・ﾓｰﾄﾞ
ＢＳ	一文字削除	○	○
ＣＲ	行入力の終了	○	○
ＤＥＬ	一文字削除	○	○
ＥＳＣ	コマンド実行の終了	○	○
ＬＦ	行入力の終了	○	○
ＴＡＢ	スペースと同じ	○	○
；	コメントの開始	○	○
！	ヒストリの呼び出し	×	○
←	カーソル左移動	×	○
→	カーソル右移動	×	○

表４－５　制御キー　（２／２）

４－２－３　スタンド・アロン時のコマンド入力方法

ＩＥ－７８１３０－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文字削除

```
＊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）　１行削除

```
＊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）　キャンセル

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
```

保守／廃止

４－２－４　システム・ソフトウェア使用時のコマンド入力方法

システム・ソフトウェアを動作させた場合は、スタンド・アロン時の１文字削除、１行削除に加え、次の機能が追加されます。

（１）　ファイルからのコマンド入力

（２）　一部のコマンドに対する省略形入力

（３）　コマンド・ヒストリ機能

（４）　行単位の編集機能

（５）　コマンド入力時の制御キー一覧

（１）　ファイルからのコマンド入力

コマンド入力時、ＳＴＲコマンドによりファイルからコマンドを入力することができます。

次に示すファイルがカレント・ドライブに存在する場合の例を示します。

◇ファイル名　　ＴＥＳＴ．ＳＴＲ

◇ファイル内容　　ＭＡＰ　２０００，３ＦＦＦ　Ｒ
ＭＡＰ　４０００，５ＦＦＦ　Ｗ
ＭＥＭ　Ｆ　２０００，３ＦＦＦ　０
ＬＯＤ　ＴＥＳＴ．ＨＥＸ　Ｃ
ＭＥＭ　Ｄ　２００Ｘ
ＭＥＭ　Ｄ　３００Ｘ

例）

```
n＞ＳＴＲ　ＴＥＳＴ．ＳＴＲ<cr>
n＞ＭＡＰ　２０００，３ＦＦＦ　Ｒ
n＞ＭＡＰ　４０００，５ＦＦＦ　Ｗ
n＞ＭＥＭ　Ｆ　２０００，３ＦＦＦ　０
n＞ＬＯＤ　ＴＥＳＴ．ＨＥＸ　Ｃ
   ｏｂｊｅｃｔ　ｌｏａｄ　ｃｏｍｐｌｅｔｅ
n＞ＭＥＭ　Ｄ　２００Ｘ
   メモリ内容の表示
n＞ＭＥＭ　Ｄ　３００Ｘ
   メモリ内容の表示
n＞
```

ＴＥＳＴ．ＳＴＲより入力したコマンド

ファイルからの入力が終了すると、以降の入力はキーボードからになります。

（２）　コマンド省略形入力

表４－２で示すコマンドについては、先頭の一文字だけの省略形入力が可能となります。コマンド本体のみ入力の場合は省略形の後に‘<cr>’を入力します。

‘ＣＬＫ’を省略形で入力するには次の様にします。

n＞C<cr>

この様に入力すると表示は次の様になり、コマンドが実行されます。

```
n＞ＣＬＫ
   ＩＥ
n＞
```

オペランドを入力する場合は省略形の後にスペース･キーを入力します。

n＞Cスペース･キー

この様に入力すると表示は次の様になり、通常の入力と同様にオペランドを入力することができます。

```
n＞ＣＬＫ_■
          └──── カーソル位置
```

コマンド	省略形	コマンド	省略形	コマンド	省略形
ＡＳＭ	Ａ	ＨＬＰ	Ｈ	ＳＡＶ	Ｓ
ＣＬＫ	Ｃ	ＬＯＤ	Ｌ	ＴＲＤ	Ｔ
ＤＡＳ	Ｄ	ＭＥＭ	Ｍ	ＶＲＹ	Ｖ
ＥＸＴ	Ｅ	ＲＵＮ	Ｒ		

表４－２　省略形一覧

（３）　コマンド･ヒストリ機能

システム･ソフトウェアは、キー入力したコマンド行を最新の２０行だけＦＩＦＯ的にヒストリ･バッファに記憶しています。これを、コマンド･ヒストリ機能といいます。

このコマンド･ヒストリの任意の１行をヒストリ･バッファから呼び出し、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。

次例では、ＨＩＳコマンドを実行し、次のようにヒストリが表示された場合を示しています。

```
 1    MEM  F  0XXX  0
 2    SYM  K
 3    LOD  TEST
 4    MEM
        .
        .
        .
17    MEM  D  0, 0FFH
18    MEM  D
19    MEM  C  START
20    HIS
```

この時、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<cr>
  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‘<cr>’を入力します。このコマンド行を修正する場合は、制御キーを用いて編集した後‘<cr>’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かわりに、もう１回‘！’を入力します。つまり、次のように入力します。

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

（４）　行単位の編集機能

カーソル制御キーを使用して行単位の編集機能が可能となります。

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き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。

また、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、‘<cr>’が入力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‘<cr>’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。

（５）　コマンド入力時の制御キー一覧

コマンド入力時に使用可能な制御キーを以下に示します。

キー	機　　能
ＤＥＬ ＢＳ ↑Ｈ	一文字削除：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	一行削除：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（２０Ｈ）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、コマンド入力モードに戻る
；	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は解釈しない
！	ヒストリ機能の呼び出し
↑Ｋ	ファイルからのコマンド入力（ＳＴＲコマンド）の中断
↑Ｌ	ファイルからのコマンド入力（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右へ１文字分移動する
↑Ａ	インサート・モード・トグル・スイッチ

表４－３　コマンド入力時の制御キー

保守／廃止

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、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、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き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１つの数値を表しますが、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。

※　ビット・タイプのシンボルは、ＡＳＭコマンド中で、ビット操作命令のオペランドに使用することができます。

保守／廃止

４ － ３ － １　数値表現

数値には、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のサフィックスを指定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

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

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

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ついたものとみなす。
ＦＦＦＦ	――	ＦＦＦＦというシンボルとみなす。
１ＦＦＦＦ －１ＦＦＦＦ ０Ｆ０ＦＦＦ	┐ ├	数値がＦＦＦＦより大きいのでエラーとなります。

保守／廃止

４ － ３ － ２　シンボル表現

シンボルには、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、数値のか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ビット数値のか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）

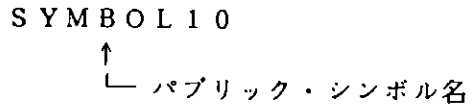
ＡＢＣＤＥＦＧＨＩＪＫ　→　ＡＢＣＤＥＦＧＨ

ａｂｃｄｅｆｇｈ　→　ＡＢＣＤＥＦＧＨ

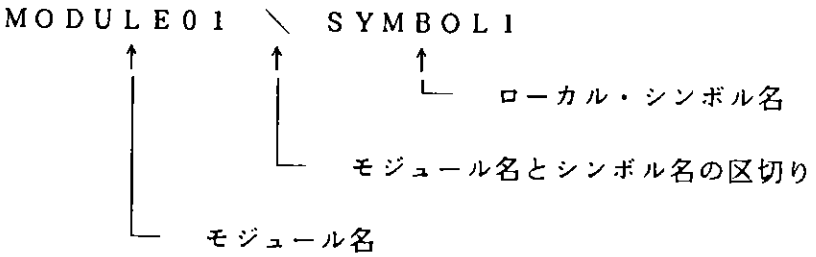
パブリック・シンボルは、シンボル名だけを記述します。しかし、ローカル・シンボルは、モジュール名と対にして表現しなければなりません。ただし、ＳＹＭ＿Ｍコマンドによりカレント・モジュールに指定されている場合は、モジュール名の記述を省略することができます。

例）

・パブリック・シンボルの記述



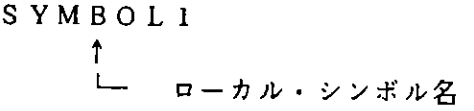
・ローカル・シンボルの記述



ＳＹＭ＿Ｍコマンドで、ＭＯＤＵＬＥ０１がカレント・モジュールに指定されている場合

ＭＯＤＵＬＥ０１ ＼ ＳＹＭＢＯＬ１

または、



４ － ３ － ３　式表現

式表現は、数値表現のか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。カッコは、最大３２レベルまでネストすることができます。

（、）	↑　最高位
＊、／	｜
＋、－	優先順位
ＡＮＤ	｜
ＯＲ、ＸＯＲ	↓　最下位

なお、演算はすべて１６ビットの整数で行われます。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ビット目から上位は切り捨てます。

例）

１０Ｈ＋１０Ｈ	→	２０Ｈ
１０Ｈ／３Ｈ	→	５Ｈ
０ＦＦＦＦＨ＋２Ｈ	→	１Ｈ
１Ｈ－２Ｈ	→	０ＦＦＦＦＨ
（１００Ｈ＊１００Ｈ）／（１００Ｈ＊１００Ｈ）	→	エラー（ｾﾞﾛ･ﾃﾞｨﾊﾞｲﾄﾞ）

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。しかし、数値、あるいはシンボルとＡＮＤ　ＯＲ　ＸＯＲの各演算子は、１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ビット数値のかわりに記述した場合も、演算は１６ビットで行います。

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わし、‘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、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
０ＸＸＸＱ	→	０Ｑ～３７７Ｑ	８ビット：３
０ＸＸＸＸＸＸＸＸＹ	→	０Ｙ～１１１１１１１１Ｙ	

特殊数値表現には、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、マスクするビット位置んに‘X’を記述してください。‘X’で表現されたビット位置が何であってもよいことを示します。

マスク・データとして記述を行った例を次頁に示します。

例）

０Ｘ００Ｈ　→　００００Ｈ、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

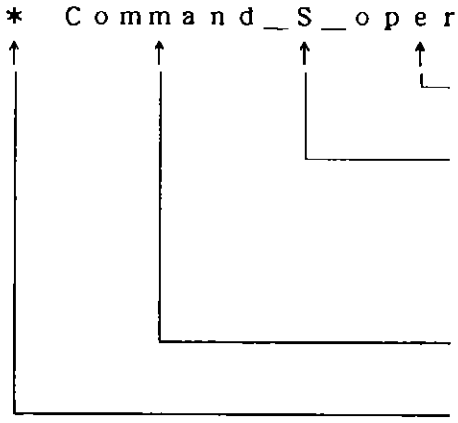
特殊数値表現が‘Ｘ’で始まる場合は、‘Ｘ’の前に数値であることを示すため‘０’をつけてください。‘Ｘ’から始まりますとシンボル表現となります。

４ — ４　コマンド一覧

４ — ４ — １　コマンド形式

ＩＥ－７８１３０－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、‘ＢＲＡ’コマンド、‘ＭＯＤ’コマンド、‘ＴＲＦ’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。



オペランド：各ｺﾏﾝﾄﾞによって異なります。

サブ・コマンド：主に処理の内容を示します。ｺﾏﾝﾄﾞ本体で処理が明示できないときにだけ使用します。ｺﾏﾝﾄﾞによっては ｻﾌﾞ･ｺﾏﾝﾄﾞがない場合もあります。

コマンド本体： 3文字で構成されます。

プロンプト： ｽﾀﾝﾄﾞ･ｱﾛﾝ･ﾓｰﾄﾞ では“＊”が出力されます。 ｼｽﾃﾑ･ﾓｰﾄﾞ では“ｎ＞”、あるいは“ｎ］”が出力されます。

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、１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、１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

例えば、１００Ｈ～１ＦＦＨのｐａｒｔｉｔｉｏｎを２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。ただし、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‘ＲＵＮ’コマンドの８ビット・レジス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。１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。１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５つまでまとめた複数の１６ビット・データの集りを表現します。

‘ＢＲＡ’コマンドのブレーク・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　５つまで

data string

data stringは、複数のbyteデータの集りを表現します。

'MEM_F'コマンドのイニシャライズ・データの集りの表現と'MEM_G'コマンドのサーチ・データの集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。

例えば、下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。

HLP_RUN

digit

digitは、4ビット数値で、バンク・ナンバを表現します。

bit

bitは、1ビット数値で、0か1を表現します。

通常はPSWフラグ値などに使用されます。

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）
ＲＰ０、ＲＰ１、ＲＰ２、ＲＰ３	（１６ビット・ジェネラル・レジスタ）
Ｘ、Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｈ、Ｌ	（８ビット・インプライド・レジスタ）
ＡＸ、ＢＣ、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。

ＩＥ	（割込み許可フラグ）
Ｚ	（ゼロ・フラグ）
ＲＢＳ１	（レジスタ・バンク選択フラグ１）
ＡＣ	（ハーフ・キャリー）
ＲＢＳ０	（レジスタ・バンク選択フラグ０）
ＩＳＰ	（割り込み優先順位ステータスフラグ）
ＣＹ	（キャリー・フラグ）

mode register name

‘MDR’コマンドのモード・レジスタ指定、および、‘BRA’コマンド、‘TRF’コマンドのaddrs表現のかわりにモード・レジスタをブレーク条件に指定する場合に使用します。

リード／ライト可能な mode register nameには以下のものがあります。（＊がついてるものは、16ビット・レジスタです。）

PRM3、TMC1、PUO、PMC3、RTPC、TOC1、ADM、PWMC、CLOM、CSIM、STBC、IF0＊（IF0L、IF0H）、MK0＊（MK0L、MK0H）、PR0＊（PR0L、PR0H）、ISM0＊（ISM0L、ISM0H）、INTM0、INTM1、IST

ライト・オンリーの mode register nameには以下のものがあります。（すべて8ビット・レジスタです。）

PM0、PM1、PM3、PM5、PM6、PM7、TMC0、CPTM、ICR、EDVC、TOM0、TOC0、TOM1、MM

保守／廃止

special register name

'SPR'コマンドのスペシャル・レジスタ指定、および、'BRA'コマンド、'TRF'コマンドのaddrs表現のかわりにスペシャル・レジスタをブレーク条件に指定する場合に使用します。

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

P0、P1、P3、P4、P5、P6、P7、*CR00、*CR01、*CR02、*CR10、*CR11、*CR20、CR30、POL、POH、SBIC、SIO、EXTSFR0、EXTSFR1、EXTSFR2、EXTSFR3、EXTSFR4、EXTSFR5、EXTSFR6、EXTSFR7、EXTSFR8、EXTSFR9、EXTSFR10、EXTSFR11、EXTSFR12、EXTSFR13、EXTSFR14、EXTSFR15

リード・オンリーの special register nameには以下のものがあります。（*がついているものは、16ビット・レジスタです。）

P2、*CR12、*CPT0、*CPT1、*CPT2H、*CPT3、*CPT2L、*TM0、*TM1、*FRC、*TM2、*TM3、CPT30、EC、ADCR

ライト・オンリーの special register nameには以下のものがあります。（16ビット・レジスタです。）

ECC1、ECC0、PWM0、PWM1

ｓｙｍｂｏｌ

「４－３　シンボル表現」で述べているシンボルのことです。ｓｙｍｂｏｌのか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。

f i l e

ファイル名を示します。ファイル名は、ドライブ番号、ディレクトリ名および、８文字以内のファイル名と３文字以内の拡張子で構成されます。ドライブ番号にはＡ～Ｐの英文字が、ディレクトリ名、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’コマンドではファイル名に？＊が使用できます。

ドライブ番号とディレクトリ名の区切りには：（コロン）を、ディレクトリ名とファイル名の区切りには￥を、ファイル名と拡張子の区切りには．（ピリオド）を使用します。

ドライブ番号やディレクトリ名は省略することができ、省略した場合はそれぞれカレント・ディスク、カレント・ディレクトリが選択されます。

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・ファイルから読み込まれたモジュール名を表します。パブリック・シンボル、あるいはカレント・モジュールに対しては省略することができます。

保守／廃止

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・ソフトウェア使用時だけ、□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。

また、△が書いてあるコマンドはＲＵＮ＿Ｎ中有効で、▲が書いてあるコマンドはＲＵＮ＿Ｎ中だけで有効なコマンドです。

保守／廃止

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
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	アクセス系ブレーク条件設定	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= [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･ﾌｪｯﾁを除くすべてのﾘｰﾄﾞ/ﾗｲﾄ)] [L=byte] ↓ (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	[bit] (外部ｾﾝｽ 信号のﾌﾞﾚｰｸ･ﾋﾞｯﾄ･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	[byte] (ﾌｪｯﾁ命令数)
	フェッチ系ブレーク条件設定	ＢＲＳ △	[S / P]	[word][_word][_word][_word]
		ＢＲＭ △	なし	[_BRA][_BRD][_BRE][_BRS][_BR0][_BR1][_BR2][_BR3] (ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ名)
論理ブレーク条件設定		ＢＲ０ △ ＢＲ１ △ ＢＲ２ △ ＢＲ３ △	なし	[_BRA][_BRD][_BRE][_BRS] (物理ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ 名)

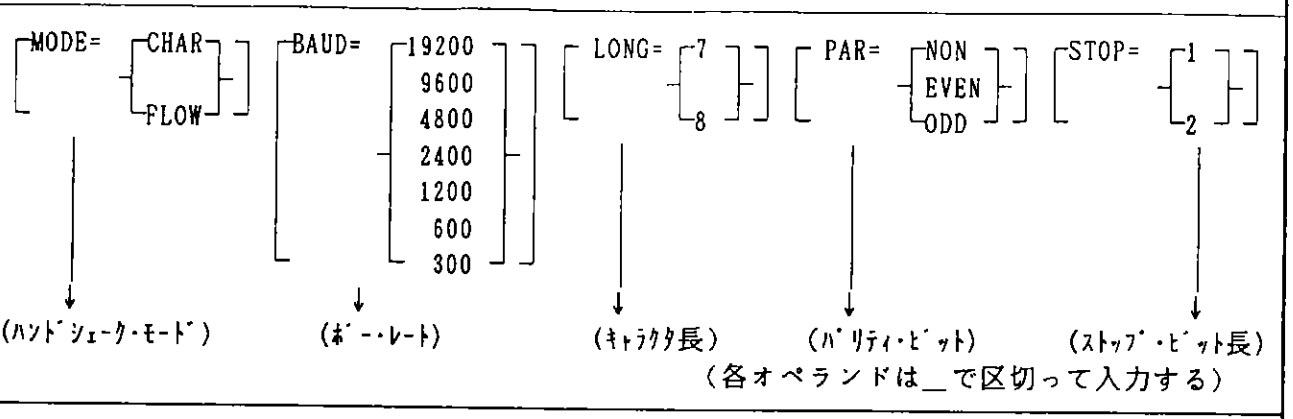
保守／廃止

コマンド種類	コマンド本体	サブ・コマンド	オペランド
クロック選択	ＣＬＫ	なし	[ ] [ ] 1(IE)
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)]]
ディレクトリ表示	ＤＩＲ△☆	なし	[file] (ファイル名)
ディレイ・カウンタ設定	ＤＬＹ △	なし	[word]
子プロセスの実行	ＤＯＳ△☆	なし	なし EXIT<ｃｒ>を入力するとシステム・ソフトウェアに戻ります。
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またはチャネル4) / TTY2(チャネル2)]] チャネル1(シリアル・インタフェース)またはチャネル4(パラレル・インタフェース)の設定はIE起動時に行います。
オブジェクト／ シンボル／ディバグ 環境のロード	ＬＯＤ ☆	なし	file [_module name＼･･･][_C][_S][_D] ↓ ↓ (オブジェクト/シンボル・ファイル 名) (モジュール名) C････オブジェクト指定 S････シンボル指定 D････ディバグ環境指定
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)]]

保守／廃止

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
マッピング設定	ＭＡＰ	[W / R / U / K]	[partition] (ﾏｯﾋﾟﾝｸﾞ範囲) (W:内部ﾏｯﾋﾟﾝｸﾞ、R:ﾗｲﾄ･ﾌﾟﾛﾃｸﾄ 内部ﾏｯﾋﾟﾝｸﾞ、U:ﾕｰｻﾞ･ﾏｯﾋﾟﾝｸﾞ、K:ﾏｯﾋﾟﾝｸﾞ 解除)
演算	ＭＡＴ △		word（通常は式を記述する）
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･ｱﾄﾞﾚｽ)
		[D](表示)	[word(表示ｽﾀｰﾄ･ｱﾄﾞﾚｽ) / partition (表示ｽﾀｰﾄ･ｱﾄﾞﾚｽ と表示ｴﾝﾄﾞ･ｱﾄﾞﾚｽ)]
		F(ｲﾆｼｬﾗｲｽﾞ)	partition_data string ←(ｲﾆｼｬﾗｲｽﾞ･ﾃﾞｰﾀ(8ﾋﾞｯﾄ) の集まり) → (ｲﾆｼｬﾗｲｽﾞ･ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		G(ｻｰﾁ)	partition_data string ←(ｻｰﾁ･ﾃﾞｰﾀ(8ﾋﾞｯﾄ) の集まり) → (ｻｰﾁ･ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		M(ｺﾋﾟｰ)	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)
		X(交換)	partition_word ←(交換先ｽﾀｰﾄ･ｱﾄﾞﾚｽ) → (交換元ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		V(比較)	partition_word ←(比較先ｽﾀｰﾄ･ｱﾄﾞﾚｽ) → (比較元ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		E(ﾃｽﾄ)	[partition] (ﾃｽﾄ･ｽﾀｰﾄ･ｱﾄﾞﾚｽとｴﾝﾄﾞ･ｱﾄﾞﾚｽ)

保守／廃止

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
チャネル２モード設定	ＭＯＤ △	なし	 （各オペランドは＿で区切って入力する）
ＩＥ代替メモリ↔ユーザ・メモリ間のデータ転送	ＭＯＶ	U / I	partition_word ←(転送先ｽﾀｰﾄ・ｱﾄﾞﾚｽ) (転送元ｽﾀｰﾄ・ｱﾄﾞﾚｽ とｴﾝﾄﾞ・ｱﾄﾞﾚｽ) (U:IE代替ﾒﾓﾘ→ﾕｰｻﾞ・ﾒﾓﾘ / I:ﾕｰｻﾞ・ﾒﾓﾘ→IE代替ﾒﾓﾘ)
端末モード	ＰＧＭ	[C]	なし
レジスタ操作	ＲＥＧ	C(変更)	[register name]
		[D](表示)	ALL / register name
レジスタ・モード設定	ＲＧＭ △	なし	I / G (I: ｲﾝﾌﾞﾗｲﾄﾞ・ﾓｰﾄﾞ , G:ｼﾞｪﾈﾗﾙ・ﾓｰﾄﾞ)

保守／廃止

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
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][,byte]（word:ｽﾀｰﾄ･ｱﾄﾞﾚｽ, byte:ｽﾃｯﾌﾟ 数）(S:ｽﾃｯﾌﾟ数指定ﾘｱﾙﾀｲﾑ実行）
		Ｔ(ﾄﾚｰｽ実行)	[word] [※1 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、 ※1 はﾚｼﾞｽﾀ条件、 word はｽﾃｯﾌﾟ数) ※1 [register name] [= > < => >= =< <= >< <>] [※2 mask wmask] PSWの ﾌﾗｸﾞ名 = bit ※2 ﾏｽｸ 表現の場合は、'=','><','<>' のみ有効
リセット	ＲＥＳ	なし	[H] (H: 省略時は ｴﾊﾞｯ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／ディバグ環境のセーブ	ＳＡＶ □	なし	[TTY1(ﾁｬﾈﾙ1) TTY2(ﾁｬﾈﾙ2)] [_partition][_partition] ・・・・ [_partition]　最大５つまで
	ＳＡＶ ☆	なし	file(入力ﾌｧｲﾙ名) [_partition][_partition] ・・・[_partition][_C][_D]　最大５つまで　C･･･ｵﾌﾞｼﾞｪｸﾄ D･･･ﾃﾞｨﾊﾞｸﾞ環境
特殊レジスタ操作	ＳＰＲ	Ｃ(変更)	[special register name]
		[D](表示)	[special register name]
ｴﾐｭﾚｰｼｮﾝCPU 停止	ＳＴＰ ▲	なし	なし
入力ﾃﾞｨﾊﾞｲｽ･ﾘﾀﾞｲﾚｸﾄ	ＳＴＲ△☆	なし	file（入力ﾌｧｲﾙ名）

保守／廃止

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
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)
	ＳＹＭ ☆	L(ﾛｰﾄﾞ)	なし
		S(ｾｰﾌﾞ)	なし
シンボル操作	ＳＹＭ△☆	[D](表示)	[module＼(ﾓｼﾞｭｰﾙ指定) / PUBLIC (ﾊﾟﾌﾞﾘｯｸ指定)]
ｶﾚﾝﾄ・ﾓｼﾞｭｰﾙ指定	ＳＹＭ	M	なし
トレーサの再起動	ＴＲＧ ▲	なし	なし
トレース・モード設定	ＴＲＭ △	なし	[P / D] (P:ﾌﾟﾛｸﾞﾗﾑのﾄﾚｰｽ / D:ﾘｰﾄﾞ・ﾗｲﾄのﾄﾚｰｽ)

保守／廃止

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・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） T（ﾎﾟｲﾝﾀをﾄﾘｶﾞ･ﾎﾟｲﾝﾄに置く）
トレース表示	ＴＲＤ △	F I	[_ALL（ﾄﾚｰｽ 結果すべてを表示）] $Q / $F （F:ﾌﾚｰﾑ･ﾓｰﾄﾞ 、I:ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ） （$Q:指定ﾌﾚｰﾑのみ表示、$F: 指定ﾌﾚｰﾑの前後5 行表示）
トレース・データ 検索条件設定	ＴＲＦ △	なし	[A=addrs][V=mask] → （検索ｱﾄﾞﾚｽ） （検索ﾃﾞｰﾀ） C= BRM1(BRM1ﾌｪｯﾁ) / M1 (M1ﾌｪｯﾁ) / OP (ｵﾍﾟｺｰﾄﾞ･ﾌｪｯﾁ) / VECT(ﾍﾞｸﾀ 参照) / 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･ﾌｪｯﾁを除くすべてのﾘｰﾄﾞ/ﾗｲﾄ) → （検索ｽﾃｰﾀｽ） [E=mask] → （外部検索ﾃﾞｰﾀ） （各オペランドは_で区切って入力する）
ｵﾌﾞｼﾞｪｸﾄ・ﾍﾞﾘﾌｧｲ	ＶＲＹ □	なし	TTY1(ﾁｬﾈﾙ1またはﾁｬﾈﾙ4) / TTY2(ﾁｬﾈﾙ2) ﾁｬﾈﾙ1(ｼﾘｱﾙ･ｲﾝﾀﾌｪｰｽ)またはﾁｬﾈﾙ4(ﾊﾟﾗﾚﾙ･ｲﾝﾀﾌｪｰｽ)の設定はIE起動時に行います。
	ＶＲＹ ☆	なし	file（入力ﾌｧｲﾙ名）

４ － ５　コマンド説明

コマンド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りのないかぎり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わします。

４－５－１　ライン・アセンブラ

ＡＳＭ［_ｗｏｒｄ］

ｗｏｒｄ：変更開始ｱﾄﾞﾚｽ

ｗｏｒｄで指定されたアドレスからメモリの内容をニー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）200H番地からの ﾒﾓﾘ内容を変更する場合

```
*ASM 200H<cr>
 0200                   NOP            ←内容が逆ｱｾﾝﾌﾞﾙ されて表示されます。
                = MOV A,#01H<cr>         ←ﾒﾓﾘ 内容の変更
       B9 01                            ←変更後のｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞ
 0202                   NOP
                = MOV [HL],A<cr>
       55
 0203                   NOP
                = MOVW AX,0FF03 <cr>
       Warning!   Generate code? (Y/N) N <cr>
                = MOVW AX,0FF03 <cr>
       Warning!   Generate code? (Y/N) Y <cr>
       11 03
 0205                   NOP
                = ORG 100H<cr>
       Caution!
 0100                   NOP
                = MOV A,FFH <cr>
       Error!
                = MOV A,0FFH<cr>
       B9 FF
 0102                   NOP
                = BR 500H <cr>
       Caution!
       2C 00 05
 0105                   NOP
                = END <cr>
*
```

・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。
（例を参照）

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。

ニーモニック、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、次のようにメッセージを表示して、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、次のニー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

一度同じアドレスで入力待ちになります。

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

望めません。

・Caution の場合は、次のようにメッセージを表示して、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に表示されます。

４－５－２　ブレーク条件設定コマンド

ＢＲＭ[_BRA][_BRD][_BRE][_BRS][_BR0][_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 BRS BR2<cr>   ←ﾌﾞﾚｰｸ条件にBRE､BRS､BR2 を指定
*

*BRM<cr>              ←ｵﾍﾟﾗﾝﾄﾞ が省略された場合
 BRE BRS BR2          ←設定されている条件の表示
*
```

ブレーク・コマンドには、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２つがあります。

物理ブレーク条件設定コマンドは、アクセス系ブレーク条件、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、フェッチ系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、組み合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]　(ｱｸｾｽ系ﾌﾞﾚｰｸ条件設定)
ＢＲＤ[_bit]　(外部信号ﾌﾞﾚｰｸ条件設定)
ＢＲＥ[_byte]　(ｲﾝｽﾄﾗｸｼｮﾝ･ｶｳﾝﾄ･ﾌﾞﾚｰｸ 条件設定)
ＢＲＳ[_ [S / P] [_word][_word][_word][_word]]　(ﾌｪｯﾁ系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][_BRS]
ＢＲ１[_BRA][_BRD][_BRE][_BRS]
ＢＲ２[_BRA][_BRD][_BRE][_BRS]
ＢＲ３[_BRA][_BRD][_BRE][_BRS]

４－５－３　アクセス系ブレーク条件設定

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

A=addrs　：ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を設定します。 ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽは、 5つまでｽﾍﾟ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　：ﾌﾞﾚｰｸ･ﾃﾞｰﾀを設定します。省略された場合は条件として設定されません。

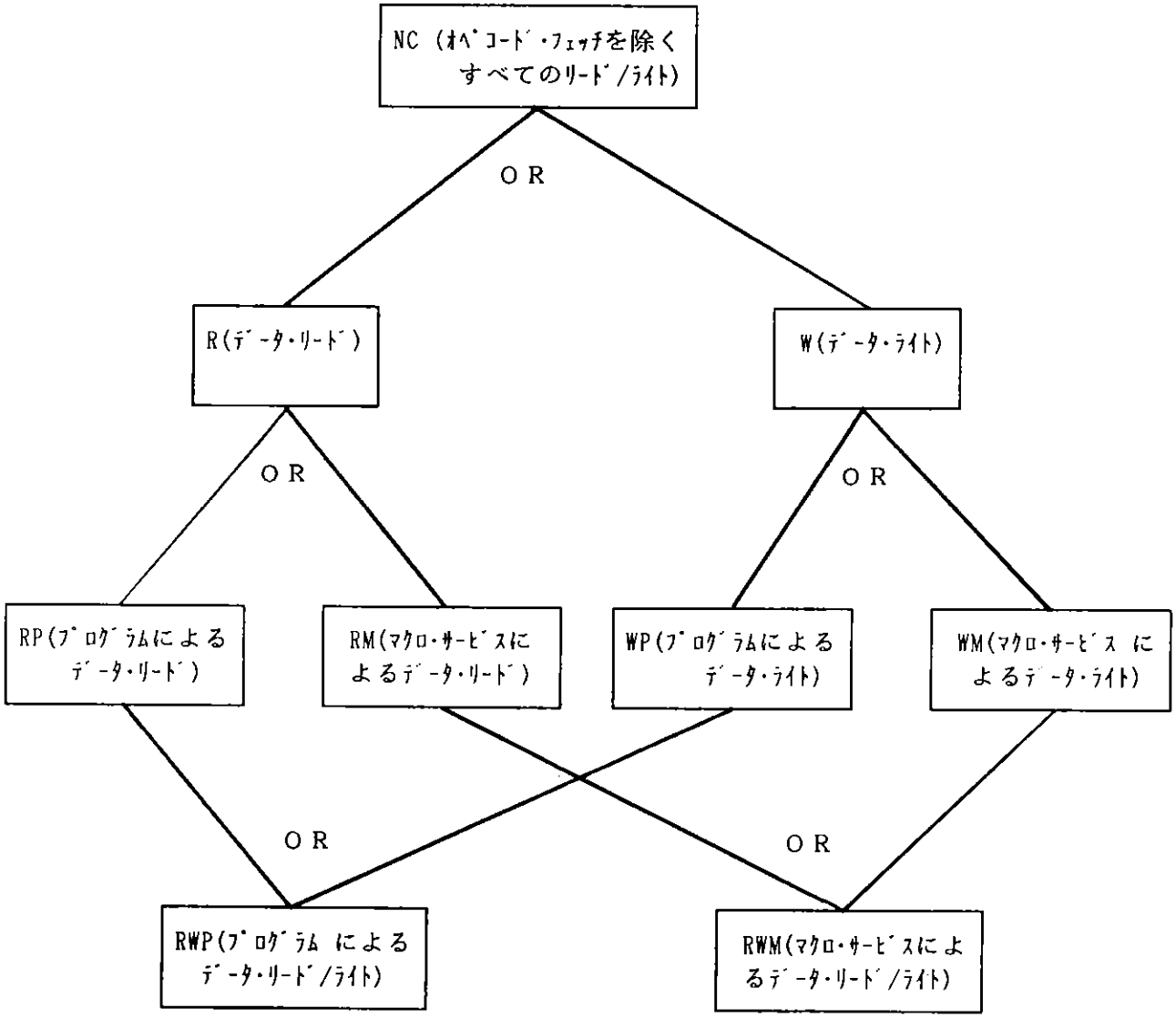
C=ｽﾃｰﾀｽ　：ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ とﾌﾞﾚｰｸ･ﾃﾞｰﾀの組み合わせ条件を下記から選択し設定します。省略した場合は条件として‘NC’が選択されます。

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 を除くすべてのﾘｰﾄﾞ/ﾗｲﾄ

L=byte　：ﾙｰﾌﾟ･ｶｳﾝﾄ数を設定します。ﾙｰﾌﾟ･ｶｳﾝﾄ数は、ﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ 、ﾌﾞﾚｰｸ･ﾃﾞｰﾀ、ﾌﾞﾚｰｸ･ｽﾃｰﾀｽの組み合わせに対する繰り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、ﾙｰﾌﾟ･ｶｳﾝﾄ数に 1 が設定されます。

‘ＢＲＡ’コマンドは、ハードウェアにアクセス系ブレーク条件を設定するコマンドです。アクセス系ブレーク条件は、‘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、ハードウェアへのアクセス系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も表示します。



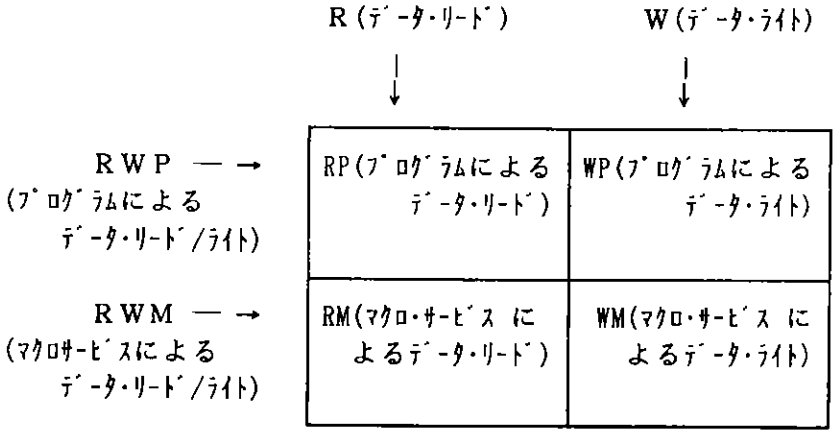


図４－５－３　　ＢＲＡコマンド　ステータス構成

例）

```
*BRA A=0FE4X 1F00,1FFF C=RWP <cr>      ←0FE40 ～ 0FE4F番地、1F00～1FFF
                                         番地の ﾒﾓﾘに対し、ﾕｰｻﾞ・ﾌﾟﾛｸﾞﾗﾑにより
                                         ﾘｰﾄﾞ/ﾗｲﾄ した場合にﾌﾞﾚｰｸします。

*BRA V=11 C=W L=10T<cr>                ←ﾃﾞｰﾀ '11' を10回 ﾃﾞｰﾀ･ﾗｲﾄした場合
                                         にﾌﾞﾚｰｸします。

*BRA <cr>                              ←対話形式でﾌﾞﾚｰｸ条件を設定する場合
 A 0,0FFFFH = 0XXXH<cr>                ←新しく 0XXXHを設定
 V 10001Y =  <cr>                      ←現在のﾃﾞｰﾀ を変更しません。
   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<cr>                          ←新しく 'NC' に設定
 L 0AH = 100H<cr>                      ←ﾙｰﾌﾟ・ｶｳﾝﾄ数に100Hを設定
 Input data error                      ←ﾙｰﾌﾟ・ｶｳﾝﾄ数は 1～FFH までなので ｴﾗｰになります。
 L 0AH = 0FFH<cr>                      ←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
 *
```

４ － ５ － ４　外部信号ブレーク条件設定コマンド

ＢＲＤ [_bit]

bit ：外部ｾﾝｽ 信号のﾌﾞﾚｰｸ･ﾃﾞｰﾀで 0か 1を設定します。

‘ＢＲＤ’コマンドは、外部センス・クリップ１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 0 <cr>          ←外部ｾﾝｽ･ｸﾘｯﾌﾟの ﾃﾞｰﾀをﾌﾞﾚｰｸ条件として 0を設定
*

*BRD <cr>            ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 0                   ←現在のﾌﾞﾚｰｸ条件を表示
*
```

保守／廃止

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 [_byte]

byte：ﾌｪｯﾁ命令数を設定します。1～255 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、現在設定されている条件を表示します。

例）

```
*BRE 10H <cr>          ←ﾌｪｯﾁ命令数に 10Hを設定
*

*BRE <cr>              ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 10H                   ←現在のフェッチ命令数を表示
*
```

４－５－６　フェッチ系ブレーク条件設定コマンド

```
BRS [_ [S|P] ][_word][_word][_word][_word]]
```

S　：設定されたﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ を ｼｰｹﾝｼｬﾙ指定とします。

P　：設定されたﾌﾞﾚｰｸ･ｱﾄﾞﾚｽ をﾊﾟﾗﾚﾙ指定とします。

word：ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ の実行ｱﾄﾞﾚｽを合計 4つまで設定します。

‘ＢＲＳ’コマンドは、ユーザ・プログラムの実行アドレスをブレーク条件として設定します。

設定できるポイント数は合計４つまでです。サブ・コマンドに‘Ｓ’を指定しますと、プログラムの流れをブレーク要因とするシーケンシャル・フェッチ・ブレーク条件となります。また、‘Ｐ’を指定しますと、設定された各ポイントをブレーク要因とするパラレル・フェッチ・ブレーク条件となります。

フェッチ系ブレーク条件設定コマンドは、‘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。ただし、条件が設定されていない場合は‘－－’が表示されます。

例）

```
*BRS S 100H 1A5H 233H 12EH <cr>     ←4ﾎﾟｲﾝﾄ の ｼｰｹﾝｼｬﾙ･ﾌｪｯﾁ･ﾌﾞﾚｰｸ条件
*                                    として設定

*BRS <cr>                           ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 Sequential                         ←現在のﾌｪｯﾁ系ﾌﾞﾚｰｸ指定および設定
 100H 1A5H 233H 12EH                  ｱﾄﾞﾚｽを表示
*

*BRS <cr>                           ←ﾌｪｯﾁ系ﾌﾞﾚｰｸ条件が設定されていない
 --                                 場合の表示
*
```

４ － ５ － ７　論理ブレーク条件設定コマンド

ＢＲ０[_BRA][_BRD][_BRE][_BRS]

※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。

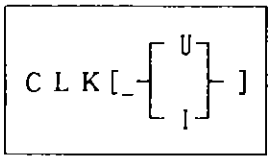
‘ＢＲ０’コマンド、‘ＢＲ１’コマンド、‘ＢＲ２’コマンド、および、‘ＢＲ３’コマンドは、物理ブレーク条件設定コマンドで設定された条件を指定することができます。条件の指定は、スペースで区切って複数を設定することができます。指定された条件はＯＲ条件となります。‘ＢＲＭ’コマンドで指定されていなければ、‘ＲＵＮ＿Ｂ’コマンドを実行してもブレークしません。

また、オペランドが省略された場合は、現在指定されている条件を表示します。

例）

```
*BR0 BRA BRD BRE BRS <cr>      ←ＢＲ０に物理ブレーク条件を設定
*

*BR0 <cr>                      ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 BRA BRD BRE BRS               ←現在設定されている条件を表示
*
```

4－5－8　クロック選択



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－７８１３０－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 <cr>          ←ユーザ・システム側のクロックを選択
*

*CLK I <cr>          ←ＩＥ側のクロックを選択
*

*CLK <cr>            ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 IE                  ←ＩＥ側のクロックが選択されている場合の表示
*

*CLK <cr>
 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
```

保守／廃止

４－５－９　コマンド・ファイル作成

```
        ┌ file ┐
COM[_┤ 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、コマンド入力時に再度‘↑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<cr>
1>
```

ﾄﾞﾗｲﾌﾞ B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<cr>
  File already exists. Delete ? (Y or N): Y <cr>
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というﾌｧｲﾙは、 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<cr>
  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というﾌｧｲﾙがR/O属性

の場合です。この場合には、ｺﾏﾝﾄﾞは無視されます。

```
1>COM LST:<cr>
1>
```

ﾘｽﾄ 装置をｵｰﾌﾟﾝします。ﾘｽﾄ 装置が他の処理によって使用中であったな

らば、下記のようなﾒｯｾｰｼﾞ が表示されます。この場合には ﾘｽﾄ装置はｵｰﾌﾟﾝ

されません。

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

・ｺﾏﾝﾄﾞ・ﾌｧｲﾙ を作成する場合の例を以下に示します。

```
1>COM B:SAMPLE.STR<cr>  ←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>MEM F 0,1FFF 1,2,3,4,5,6,7,8, <cr>  ←このｺﾏﾝﾄﾞ行で '↑0' を入力
1>MEM D 080X<cr>
  0800   01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08    ................
1>BRA A=100 <cr>
1>BRM BRA <cr>
1>TRM P <cr>              ←このｺﾏﾝﾄﾞ行で '↑0' を入力
1>
```

→ SAMPLE.STR に出力されます。

この場合には、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 0,1FFF 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。

保守／廃止

４－５－１０　逆アセンブラ

```
DAS[_┌ word      ┐ ]
     └ partition ┘
```

word　　：逆ｱｾﾝﾌﾞﾗ 開始ｱﾄﾞﾚｽを設定します。

partition：逆ｱｾﾝﾌﾞﾗ 開始／終了ｱﾄﾞﾚｽを設定します。

‘ＤＡＳ’コマンドは、指定されたアドレスからメモリ内容をニー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 100H<cr>
 Addr  Object          Mnemonic
                       ORG     100H
 0100  B8 00           MOV     X,#0H
 0102  B9 01           MOV     A,#1H
 0104  BB 02           MOV     B,#2H
 0106  BA 03           MOV     C,#3H
 0108  BD 04           MOV     D,#4H
 010A  BC 05           MOV     E,#5H
 010C  BF 06           MOV     H,#6H
 010E  BE 07           MOV     L,#7H
                       END
*
```

４－５－１１　ディレクトリ表示

```
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。

‘ＤＩＲ’コマンドは、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 <cr>  ←カレント・ディレクトリのすべてのファイルを表示します。
  Directory = A:¥
    COMMAND  COM    LC       EXE    LC1      EXE    LC2      EXE    DUMP     COM
    OML      EXE    FORMAT   EXE    DEBUG    COM    PR       EXE    ASSERT   H
    CTYPE    H      DOS      H      ERROR    H      FCNTL    H      FCTYPE   H
    FLOAT    H      IOS1     H      LIMITS   H      LOCKING  H      MATH     H
    PC       H      SETJMP   H      SIGNAL   H      STDIO    H      STDLIB   H
    STRING   H      TIME     H      UNLSTD   H      8087     MAC    KANJI    H
    PRINT    EXE    FIND     EXE    MORE     COM    SORT     EXE    MASM     EXE
    LINK     EXE    LABEL    EXE    SHARE    EXE    DISKCOPY COM    KEY      COM
    SYMDEB   EXE    AUTOEXEC BAT    AAA      HEX    LINK2    EXE    LINKS    BAK
    TEST     OBJ    README   DOC    CS       OBJ    _MAINS   OBJ    NDPS     OBJ
    NONDPS   LIB    KANJIS   LIB    SM8086   MAC    TEST     BAK    LCS      LIB
    TEST     MAP    TEST     EXE    LINKS    BAT    TEST     C      SVI      HLP
    SVI      EXE    LCS      BAT    LINKMS   BAT    GET      OVL    INI      OVL
    REC      OVL
 1>

1>DIR ¥WORK<cr>  ←カレント・ドライブ のWORKというディレクトリの中のファイルを表示します。
  Directory = A:¥WORK
    .              ..             TEST1    MAP    TEST1    EXE    TEST1    ASM
    TEST     C     SVI      HLP   HO       BAT
1>
```

４－５－１２　ディレイ・カウンタ設定

ＤＬＹ[_word]

word：ﾄﾚｰｽ･ﾄﾘｶﾞ検出後のﾄﾚｰｽ･ﾌﾚｰﾑ･ｶｳﾝﾀ数を設定します。

‘ＤＬＹ’コマンドは、リアルタイム実行に於いてのトレースで、トレース・トリガ（ブレーク要因）検出後のトレース・フレーム数を設定します。

トレース・フレーム数は、０～７ＦＦＨの範囲内で設定することができます。

また、オペランドが省略された場合は、現在の設定値を表示します。

例）

```
*DLY 1024T <cr>        ←ﾃﾞｨﾚｲ･ｶｳﾝﾀ 条件としてﾄﾚｰｽ･ﾌﾚｰﾑ･ｶｳﾝﾀ数の設定
*

*DLY <cr>              ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 400H                  ←現在設定されているｶｳﾝﾀ数の表示
*
```

保守／廃止

４－５－１３　子プロセスの実行

DOS

‘ＤＯＳ’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。

‘ＤＯＳ’コマンドを入力することにより、一時的にＯＳに制御を移します。

再びシステム・ソフトウエアに制御を移すには、‘ＥＸＩＴ<cr>’と入力します。

例）

```
1>DOS <cr>  ←DOSのプロセスが実行できるようになります。
A>COPY B:TEST.ASM <cr>  ←DOSのコマンドを入力します。
A>FORMAT C: <cr>  ←続けてDOSのコマンドを入力します。
     .
     .
     .
A>EXIT<cr>
1>          ←ｼｽﾃﾑ･ｿﾌﾄｳｪｱに戻ります。
```

保守／廃止

４－５－１４　システム・モード終了

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。

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、ＯＳに戻ります。

例）

```
1>EXT <cr>
A>                          ←ＯＳ のﾌﾟﾛﾝﾌﾟﾄが表示されます。
```

４－５－１５　コマンド・ヒストリ表示

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。

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み出すことができます。コマンドを読み出す場合は‘！ｎ<cr>’と入力します。このときｎに読み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‘！！<cr>’と入力するだけで読み出すことができます。読み出したコマンドの実行は、コマンド行で‘<cr>’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 <cr>            ←ﾋｽﾄﾘ･ﾒﾓﾘに登録されているｺﾏﾝﾄﾞを表示
   1 LOD TEST
   2 MEM D 0FX
   3 SYM
       .
       .
       .
  18 CLK I
  19 HLP HIS
  20 HIS
1>

1>!18 <cr>            ←ﾋｽﾄﾘ･ﾒﾓﾘの18番目のｺﾏﾝﾄﾞを読み出し
  CLK I               ←このｺﾏﾝﾄﾞを実行するには<cr>を入力
                              .
                              .
                              .
1>
```

４－５－１６　ヘルプ

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です。

‘ＨＬＰ’コマンドは、ＩＥ－７８１３０－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。

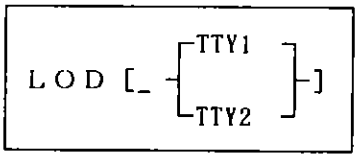
コマンド本体が省略された場合は、コマンドの一覧が表示されます。その後、コマンド本体の入力ができます。

例）

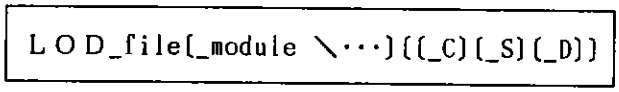
```
1>HLP DIR <cr>  ←ｵﾍﾟﾗﾝﾄﾞ にｺﾏﾝﾄﾞを指定した場合
     ┌
     | " DIR"ｺﾏﾝﾄﾞ の説明
     └
1>

1>HLP <cr>
    Command Table
  ASM          BR0          BR1          BR2          BR3        ┐
  BRA          BRD          BRE          BRS          BRM        |
  CLK          COM          DAS          DIR          DLY        |
  DOS          EXT          HIS          HLP          LOD        |
  LST          MAP          MAT          MDR          MEM        | ｺﾏﾝﾄﾞの一覧
  MOD          MOV          PGM          REG          RES        |
  RGM          RUN          SAV          SPR          STP        |
  STR          SUF          SYM          TRG          TRM        |
  TRP          TRD          TRF          VRY                     ┘
HLP>DIR <cr>  ←" 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><cr>    ←ﾍﾙﾌﾟ･ｺﾏﾝﾄﾞを終了します。
1>
```

保守／廃止

４－５－１７　オブジェクト／シンボル／ディバグ環境のロード



スタンド・アロン・モードの形式



システム・モードの形式

TTY1　　：ﾁｬﾈﾙ 1 あるいは ﾁｬﾈﾙ 4 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　　：ﾌｧｲﾙ名を設定します。

module＼　：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･ｺｰﾄﾞのﾛｰﾄﾞをするｽｲｯﾁです。

D　　：ﾃﾞｨﾊﾞｸﾞ環境のﾛｰﾄﾞをするスイッチです。

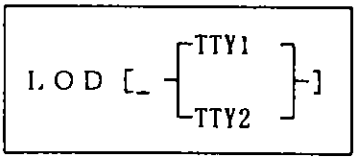
S　　：ｼﾝﾎﾞﾙ･ﾌｧｲﾙ のﾛｰﾄﾞをするｽｲｯﾁです。

'ＬＯＤ' 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、パラレル・インターフェース(ﾁｬﾈﾙ4) あるいは、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、ディバグ環境、あるいは、シンボル･ファイルをパラレル･インターフェース（CH4）あるいはシリアル･インターフェース（CH1）を使用してロードすることができます。

パラレル･インターフェースの使用は起動時の設定で、パラレル･インターフェースを使用する様にした場合に使用可能となります。又他のコマンドでリスト装置を使用している場合はパラレル･インターフェースの使用はできません。その場合は、シリアル･インターフェースを使用します。

（１）　スタンド・アロン・モードの場合



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みすてます。

オペランドを省略した場合ＴＴＹ１が指定されます。

例）

```
*LOD TTY1〈cr〉        ←ﾁｬﾈﾙ1あるいはﾁｬﾈﾙ4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。

```
Check sum error      ←ｻﾑﾁｪｯｸ･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＼,_module＼, ・・・・・][[_C][_D][_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・テーブル及びディバグ環境をロードすることができます。

オブジェクト・コード、シンボル・テーブル及びディバグ環境は、同時あるいは個別にロードすることができます。

なお、シンボル・テーブルをロードする場合は、シンボル・テーブ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。

起動時に高速ダウン・ロードを選択し、本コマンド実行時に他のコマンドで、リスト装置を使用している場合は、‘ Select serial interface ’を表示し　シリアル・チャネル１を通してダウン・ロードを行います。

例）

◆ｵﾌﾞｼﾞｪｸﾄ・ｺｰﾄﾞだけをﾛｰﾄﾞする場合

LOD_file_C

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"をつけます。

```
1>LOD SAMPLE.HEX C<cr>
  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
1>                                ﾒｯｾｰｼﾞ
```

ﾌｧｲﾙ名の拡張子を省略した場合は" HEX" が設定されます。

```
1>LOD SAMPLE C<cr>              ←ﾌｧｲﾙはSAMPLE.HEXとなります
  object load complete
1>
```

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        ←ｻﾑﾁｪｯｸ･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
```

◆ｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ だけを ﾛｰﾄﾞする場合

```
LOD_file[_module＼･･･]_S
```

ｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ 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"をつけます。

```
1>LOD SAMPLE.SYM S<cr>
  symbol table loading          ←ｼﾝﾎﾞﾙ･ﾛｰﾄﾞの開始ﾒｯｾｰｼﾞ
  PUBLIC      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
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
1>
```

ﾌｧｲﾙ名の拡張子は省略することができ、省略した場合は“SYM ”が設定

されます。

```
1>LOD SAMPLE S<cr>              ←ﾌｧｲﾙはSAMPLE.SYMとなります
  symbol table loading
  PUBLIC      load complete
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
1>
```

ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸが指定された場合は、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。

ﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸの指定は、ｺﾏﾝﾄﾞ行の範囲内でいくつでも指定することができます。

```
1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ S<cr>
  symbol table loading
  PUBLIC      load complete
  MOD01       pass              ←指定しないﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸに表示されます
  MOD02       load complete
  MOD03       pass
  MOD04       load complete
1>
```

ﾛｰﾄﾞ 済みのﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸをもう一度 ﾛｰﾄﾞした場合は、下の例のような ﾒｯｾｰｼﾞ が表示されます。この場合、 ﾛｰﾄﾞ済みのﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸは影響を受けません。

```
1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ S<cr>
  symbol table loading
  PUBLIC      loaded module
  MOD01       pass
  MOD02       loaded module
  MOD03       pass
  MOD04       loaded module
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。

```
1>LOD SAMPLE PUBLIC ＼,MOD02＼,MOD04＼ S<cr>
  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>
```

◆ﾃﾞｨﾊﾞｸﾞ環境だけをﾛｰﾄﾞする場合

```
LOD_file_D
```

ﾃﾞｨﾊﾞｸﾞ環境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 “D”をつけます。

また,ﾌｧｲﾙ名の拡張子は、DBGでなければなりません。省略した場合は “DBG”が指定されます。

```
1>LOD SAMPLE D<cr>
1>
```

ﾃﾞｨﾊﾞｸﾞ環境としてﾛｰﾄﾞされるのは、次のｺﾏﾝﾄﾞで設定された内容です。

- ･BRA
- ･BRD
- ･BRE
- ･BRS
- ･BRM
- ･BRO/1/2/3
- ･CLK
- ･DLY
- ･MAP
- ･MOD
- ･PGM
- ･SUF
- ･TRM
- ･TRF

◆ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞとｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙとﾃﾞｨﾊﾞｸﾞ環境を同時に ﾛｰﾄﾞする場合

LOD_file[_module＼･･･]

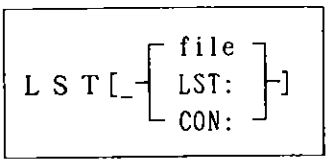
ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙとｼﾝﾎﾞﾙ･ﾌｧｲﾙ及びﾃﾞｨﾊﾞｸﾞ環境ﾌｧｲﾙは、同じﾄﾞﾗｲﾌﾞ上に存在しなければなりません。

ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞとｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ及びﾃﾞｨﾊﾞｸﾞ環境 を同時に ﾛｰﾄﾞする場合は、ﾌｧｲﾙ名に拡張子を指定することは出来ません。この場合の拡張子には、ｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ は“ HEX”、ｼﾝﾎﾞﾙ･ﾌｧｲﾙ は“ SYM”、ﾃﾞｨﾊﾞｸﾞ環境ﾌｧｲﾙには“DBG” が自動的に設定されます。


```
1>LOD SAMPLE<cr>          ←ｵﾌﾞｼﾞｪｸﾄ･ｺｰﾄﾞとｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ ﾃﾞｨﾊﾞｸﾞ環境を同時にﾛｰﾄﾞ
  object load complete    ←SAMPLE.HEXのﾛｰﾄﾞ 終了ﾒｯｾｰｼﾞ
  symbol table loading    ←SAMPLE.SYMの ﾛｰﾄﾞ開始ﾒｯｾｰｼﾞ
  PUBLIC      load complete
  MOD01       load complete
  MOD02       load complete
  MOD03       load complete
  MOD04       load complete
  Debug condition load (Y/N)? Y<cr>  ←ﾃﾞｨﾊﾞｸﾞ環境をﾛｰﾄﾞするか
  debug condition load complete       聞いてきます。 'Y' 以外の入力では
1>                                    ﾛｰﾄﾞしません。

1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\<cr>←ｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ のﾓｼﾞｭｰﾙ･ﾌﾞﾛｯｸを指定
  object load complete
  symbol table loading
  PUBLIC      load complete
  MOD01       pass
  MOD02       load complete
  MOD03       pass
  MOD04       load complete
  Debug condition load(Y/N)? N<cr>  ←ﾃﾞｨﾊﾞｸﾞ環境はﾛｰﾄﾞしません。
1>
  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･ﾃｰﾌﾞﾙ とﾃﾞｨﾊﾞｸﾞ環境の
  ﾛｰﾄﾞはしません。
1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\<cr>
  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
1>                        場合のﾒｯｾｰｼﾞ

1>LOD SAMPLE PUBLIC \,MOD02\,MOD04\<cr>
  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
```

保守／廃止

４－５－１８　出力ディバイス・リダイレクト



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。

出力ディバイス・リダイレクト・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、‘↑Ｐ’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、コマンド入力時に再度‘↑Ｐ’キーが入力されるまでです。

例）

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

ドライブ Bに、すでにSAMPLE.TXTというR/W属性の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〈cr〉
  File already exists.
1)
```

ﾄﾞﾗｲﾌﾞ Bに、すでにSAMPLE.TXTというR/O属性のﾌｧｲﾙが存在している場合です。

この場合には、ｺﾏﾝﾄﾞは無視されます。

```
1)LST LST:〈cr〉
1)
```

ﾘｽﾄ 装置をｵｰﾌﾟﾝします。ﾘｽﾄ 装置が他の処理によって使用中であったならば、下記のようなﾒｯｾｰｼﾞ が表示されます。この場合には、 ﾘｽﾄ装置はｵｰﾌﾟﾝされません。

```
1)LST LST:〈cr〉
  List device is used by other process.
1)
```

・ﾌｧｲﾙを出力ﾃﾞｨﾊﾞｲｽ としてｵｰﾌﾟﾝした場合の例

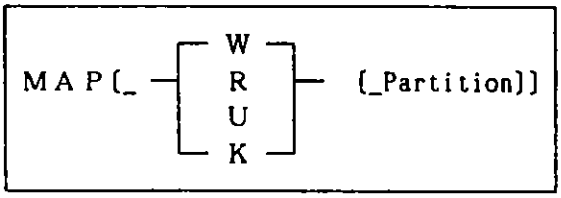
```
1)LST B:SAMPLE.TXT〈cr〉   ←出力ﾃﾞｨﾊﾞｲｽ としてﾄﾞﾗｲﾌﾞ BにSAMPLE.TXTというﾌｧｲﾙ名でｵｰﾌﾟﾝ されます。
1)MEM F 0,1FFF 1,2,3,4,5,6,7,8,〈cr〉  ←このｺﾏﾝﾄﾞ行で '↑P'を入力
1)MEM D 080X〈cr〉
  0800   01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08    ................
1)BRA A=100 〈cr〉
1)BRM BRA 〈cr〉
1)TRM P 〈cr〉              ←このｺﾏﾝﾄﾞ行で '↑P ' を入力
1)
```

→ SAMPLE.TXTに出力されます。

この場合には、ﾌｧｲﾙをｵｰﾌﾟﾝ後の '↑P ' が入力されたｺﾏﾝﾄﾞ(MEM F 0,1FFF 1,2,･･･)から次の '↑P ' が入力される前のｺﾏﾝﾄﾞ(BRM BRA) までがﾄﾞﾗｲﾌﾞ Bの SAMPLE.TXTというﾌｧｲﾙに出力されます。

保守／廃止

４－５－１９　　マッピング設定



W：代替メモリ・マッピング
R：ライト・プロテクト付き代替メモリ・マッピング
U：ユーザ・メモリ・マッピング
K：ノン・マッピング（マッピングの解除）
Partition：マッピング範囲

“ＭＡＰ”コマンドは、内部ＲＯＭ、内部ＲＡＭ、ＳＦＲを除く範囲を２５６バイト単位で自由にマッピングする事が出来ます。

ノン・マッピングの場合は、指定範囲のマッピングを解除します。

コマンド入力時サブ・コマンドのみを指定すると、サブ・コマンドで指定した種類でマッピングされている範囲を表示します。

尚、サブ・コマンドがノン・マッピングの場合は、“ＭＡＰ”コマンドで設定したすべてのマッピングを解除します。

コマンド本体のみ入力の場合は、現在のマッピング状態を表示します。

例）

マッピングの設定　（ｵﾝ･ﾁｯﾌﾟ ROM4K,ｵﾝ･ﾁｯﾌﾟ RAM256byteの場合）

```
*MAP W 1XXX〈cr〉          ←1000H～1FFFHの範囲を代替メモリ・マッピングに設定
*
*MAP R 2000,2FFF〈cr〉     ←2000H～2FFFHの範囲をライト・プロテクト付き代替
*                           メモリ・マッピングに設定
*MAP U 3000,3FFF〈cr〉     ←3000H～3FFFHの範囲をユーザ・メモリ・マッピングに
*                           設定
```

マッピング状態の表示

(ｵﾝ･ﾁｯﾌﾟROM4K,ｵﾝ･ﾁｯﾌﾟRAM256byteの場合)

```
*MAP<cr>                    ←すべてのマッピング状態の表示
 0000-0FFF R/O   1000-1FFF R/W   2000-2FFF R/O   3000-3FFF User
 4000-FDFF Non
*

*MAP W<cr>                  ←代替メモリ・マッピングの範囲を表示
 1000-1FFF
*

*MAP R<cr>                  ←ライト・プロテクト付き代替メモリ・マッピングの
 2000-2FFF                    範囲を表示
*

*MAP U<cr>                  ←ユーザ・メモリ・マッピングの範囲を表示
 3000-3FFF
*
```

マッピングの解除

```
*MAP K 1XXX<cr>             ←１０００Ｈ～１ＦＦＦＨの範囲のマッピングを解除
*

*MAP K<cr>                  ←すべてのマッピングを解除
*
```

保守／廃止

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 <cr>          ←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 － ２ １　モード・レジスタ操作

```
MDR [_D [_Mode  register  name] ]

MDR_C [_Mode  register  name]
```

D：　モード・レジスタの表示
C：　モード・レジスタの変更
Mode　register　name：　モード・レジスタ名

使用出来るモード・レジスタ名を以下に示します。

```
PRM3、<PM0>、<PM1>、<PM3>、<PM5>、<PM6>、<PM7>、<TMC0>、TMC1、<CPTM>、

PU0、PMC3、RTPC、<ICR>、<EDVC>、<TOM0>、<TOC0>、<TOM1>、TOC1、ADM、PWMC、
                        *              *              *
CLOM、CSIM、STBC、<MM>、IF0(IF0L, IF0H)、MK0(MK0L, MK0H)、PR0(PR0L、PR0H)、
   *
ISM0(ISM0L, ISM0H)、INTM0、INTM1、IST
```

＊　で示されたﾚｼﾞｽﾀは16ﾋﾞｯﾄ･ﾚｼﾞｽﾀ
　　その他は 8ﾋﾞｯﾄ･ﾚｼﾞｽﾀ
＜ ＞ で示されたﾚｼﾞｽﾀは書き込み専用ﾚｼﾞｽﾀ

モード・レジスタ操作コマンドは、モード・レジスタを８ビット単位で変更、および表示することができます。

モード・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、書き込み可能なモード・レジスタを‘ＰＲＭ３’から‘ＩＳＴ’の順番に変更することができます。

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。変更しない場合は、リターン・キーを入力します。また変更を途中で終了する場合は、‘．’（ピリオド）を入力します。

モード・レジスタの内容を表示する場合にレジスタ名を省略すると、読み出し可能なすべてのモード・レジスタを表示することができます。レジスタ名が指定された場合は、指定されたレジスタの内容だけが表示されます。

☆　チップの仕様による書き込み禁止データは自動的にエラーにします。

例）

モード・レジスタの表示

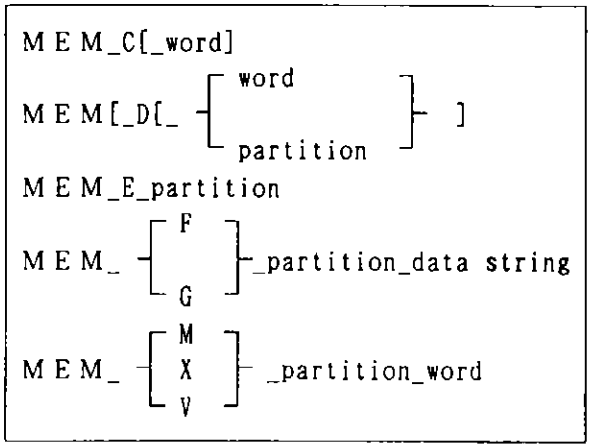
```
*MDR D<cr>                ←書き込み専用以外のモード・レジスタを表示
 PRM3   TMC1   PUO    PMC3   RTPC   TOC1   ADM    PWMC   CLOM   CSIM   STBC
 10     20     30     40     50     60     70     80     90     A0     B0

 IFOL   IFOH   MKOL   MKOH   PROL   PROH   ISMOL  ISMOH  INTM0  INTM1  IST
 C0     D0     E0     F0     01     02     03     04     05     06     07
*

*MDR D RTPC<cr>           ←ＲＴＰＣレジスタの表示
 RTPC     50
*

*MDR D PM0<cr>            ←書き込み専用レジスタの内容を表示
 PM0      --
*
```

例）

モード・レジスタの変更

```
*MDR C<cr>                      ←すべてのモード・レジスタの変更
 PRM3     10 = 20<cr>           ←10を20に変更
 PM0      -- = <cr>             ←変更なし
 PM1      -- = 66<cr>           ←66に変更
      .
      .
      .
 MKOL     E0 = 0EE<cr>
 MKOH     F0 = . <cr>           ←変更をやめる
*

*MDR C CSIM<cr>                 ←ＣＳＩＭレジスタから変更
 CSIM     A0 = 80<cr>
 STBC     B0 = <cr>
 MM       -- = 7<cr>
      .
      .
      .
 IST      07 = .<cr>
*
```

４ － ５ － ２ ２　メモリ操作



C ：ﾒﾓﾘ 内容の変更

D ：ﾒﾓﾘ 内容の表示

E ：ﾒﾓﾘ のﾃｽﾄ

F ：ﾒﾓﾘ 内容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、内部ＲＯＭ、および、内部ＲＡＭのメモリ内容の変更、表示、テスト、イニシャライズ、サーチ、コピー、交換、比較を８ビット単位でおこないます。

（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

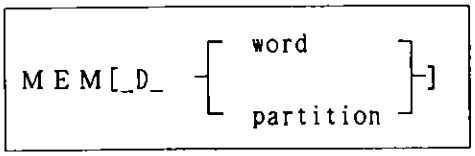
メモリ操作可能範囲のメモリを変更します。

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 <cr>        ←100 番地からのﾒﾓﾘ 内容を変更
 0100  00 11 <cr>
 0101  11 22 <cr>
 0102  22 33 <cr>
 0103  33 44 <cr>
 0104  44 55 <cr>
 0105  55 66 <cr>
 0106  66 77 <cr>
 0107  77 88 <cr>
 0108  88 .<cr>        ←ﾒﾓﾘ 内容の変更終了
*

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

（２）　メモリ内容表示



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

メモリ操作可能範囲のメモリを表示します。

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 <cr>     ←100 ～ 17Fまでの ﾒﾓﾘ内容を表示
 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 <cr>         ←108 から11行分を表示
 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 <cr>             ←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。

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。

	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
下 5	%	5	E	U	e	u
位 6	&	6	F	V	f	v
4 7	'	7	G	W	g	w
ビ 8	(	8	H	X	h	x
ッ 9	)	9	I	Y	i	y
ト A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	¥	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	
F	/	?	O	_	o	

（３）　メモリのテスト

ＭＥＭ_E_partition

partition ：ﾒﾓﾘ の ﾃｽﾄ範囲

指定された範囲のメモリのテストをすることができます。指定できる範囲は、内部ＲＯＭ、および、内部ＲＡＭの範囲内です。メモリ範囲の指定を省略した場合はエラーとなります。

例）

```
*MEM E 0XXX<cr>        ←0000～0FFFまでのﾃｽﾄ
 complete              ←ﾒﾓﾘ は正常
*

*MEM E 1000,1FFF <cr>  ←1000～1FFFH までのﾃｽﾄ
 152A                  ←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、異常を検出したｱﾄﾞﾚｽ以後のﾃｽﾄはされません。

（４）　メモリ内容のイニシャライズ

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

メモリ内容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 <cr>  ← 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

*MEM D 100,1FF <cr>     ← 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<cr>  ←ﾏｽｸ･ﾃﾞｰﾀ によるｲﾆｼｬﾗｲｽﾞ
*

*MEM D 100,13F <cr>     ←ｲﾆｼｬﾗｲｽﾞ 後のﾒﾓﾘ 内容の表示
 0100   31 32 33 34 35 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
0110   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
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
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 Q8 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<cr>        ← 100～1FF の範囲から30,31,32の連続した
 0109                                 ﾃﾞｰﾀ列を ｻｰﾁ
 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 元範囲

word　　：ｺﾋﾟｰ 先ｱﾄﾞﾚｽ

メモリ内容のコピーは、コピー元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。コピー元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 <cr>          ←100 ～ 13Fまでを 180以降にｺﾋﾟｰ
*

*MEM D 100,1FF <cr>              ←ｺﾋﾟｰ 後のﾒﾓﾘ 内容の表示
 0100   31 32 33 34 35 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 35 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 35 36 37 38 39 30 31 32 33 34 35 36
0110   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
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 <cr>          ←100 ～ 13Fまでを 140以降と交換
*
*MEM D 100,17F <cr>              ←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 35 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）

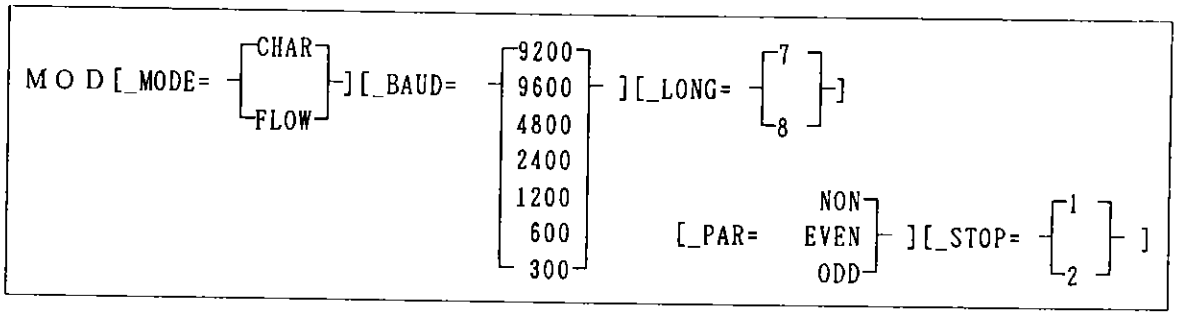
100番地以降の ﾒﾓﾘ内容が以下のようになっているとします。

```
0100   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
0110   31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36
0120   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0130   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0140   09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
0150   05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 01
0160   31 32 33 34 35 41 37 38 39 30 31 32 33 34 35 36
0170   37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0180   03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0190   09 00 01 02 2F 04 05 06 07 08 09 00 01 02 03 04
```

このとき、次のｺﾏﾝﾄﾞを実行すると、比較した結果が表示されます。

```
*MEM V 100,14F 150 <cr>          ←100 ～ 14Fまでを 150以降と比較
 010F 00  015F 01
 0115 36  0165 41
 0144 03  0194 2F
*
```

４－５－２３　チャネル２モード設定



MODE：ﾊﾝﾄﾞｼｪｰｸ･ﾓｰﾄﾞ 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の動作状態を設定します。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行うことができます。

なお、初期状態では、１キャラクタ・ハンドシェーク、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）

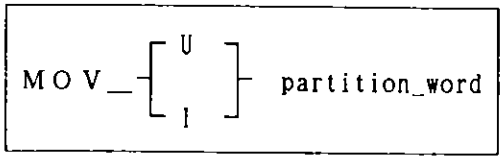
```
*MOD MODE=CHAR BAUD=4800 LONG=8 PAR=NON STOP=2<cr>
*
```

1ｷｬﾗｸﾀ･ﾊﾝﾄﾞｼｪｰｸ、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ﾋﾞｯﾄに設定します。

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

保守／廃止

４－５－２４　ＩＥ代替メモリ↔ユーザ・メモリ間のデータ転送



U　　：代替ﾒﾓﾘ→ﾕｰｻﾞ・ﾒﾓﾘ転送
I　　：ﾕｰｻﾞ・ﾒﾓﾘ →代替ﾒﾓﾘ転送
partition：転送元ﾒﾓﾘ 範囲
word　　：転送先ﾒﾓﾘ・ｱﾄﾞﾚｽ

メモリ転送コマンドは、代替メモリからユーザ・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代替メモリへメモリ内容の転送を行います。

代替メモリからユーザ・システムのメモリへ転送する場合、転送先メモリ・アドレスは、ユーザ・マッピング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代替メモリへ転送する場合、転送先メモリ・アドレスは、代替メモリ、ライト・プロテクト付き代替メモリのいずれかにマッピングされていなければなりません。

内部ＲＯＭに指定されている範囲へは、メモリ転送することはできません。

例）

```
*MAP U 0,1FFF<cr>          ← 0～1FFF 番地をﾕｰｻﾞ・ﾏｯﾋﾟﾝｸﾞ にする
*MOV U 2000,3FFF 0 <cr>    ←代替ﾒﾓﾘ の2000～3FFF番地までを、ﾕｰｻﾞ・ｼｽﾃﾑ
*                            の 0番地以降に転送

*MAP R 0,1FFF<cr>          ← 0～1FFF 番地をﾗｲﾄ・ﾌﾟﾛﾃｸﾄ付き代替ﾒﾓﾘ・ﾏｯﾋﾟﾝｸﾞ にする
*MOV I 0,1FFF<cr>          ←ﾕｰｻﾞ・ｼｽﾃﾑの 0～1FFF番地までを、内部ﾒﾓﾘ の
*                            0 番地以降に転送

*MAP I 8K<cr>              ← 0～1FFF 番地を内部ＲＯＭマッピングにする
*MOV I 0,1FFF 0<cr>          ←ﾕｰｻﾞ・ｼｽﾃﾑの 0～1FFF番地までを、内部ROM の
 Mapping error               0 番地以降に転送
*                           ( 内部ROM のためｴﾗｰ)
```

保守／廃止

４－５－２５　端末モード（ＰＲＯＭプログラマ制御）

ＰＧＭ

端末モード・コマンドは、シリアル・チャネル２とＰＧ－２０００／１０００と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しますとＰＧＭモードという状態になります。

ＰＧＭモードでは、シリアル・チャネル２とＰＧ－２０００／１０００を接続することによりＰＧ－２０００／１０００からオブジェクト・コードのアップ／ダウン・ロードができるようになります。

ＰＧＭモードの終了は、コンソールから‘↑Ｚ’キーを入力します。

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モードに於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出来ます。

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出来ます。

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事は出来ません。

ＤＥＬキーあるいはｃｔｒｌ－Ｈ入力の場合はデフォ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)

*PGM C <cr>

```
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
```

デフォルト 値を表示

```
Termination of "PGM"       ···^Z <cr>
```

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 <cr>
Beginning   of "HEX LOAD"  ···^Z <cr>
Beginning   of "HEX SAVE"  ···^N <cr>
Beginning   of "SYM LOAD"  ···^E <cr>
Termination of "LOAD"      ···^B <cr>
Termination of "SAVE"      ···^X <cr>
Break       of "LOAD/SAVE"···^G <cr>
```

ctrl-Zをctrl-Aに変更
ctrl-Aをctrl-Zに変更
ctrl-Eをctrl-Nに変更
ctrl-Nをctrl-Eに変更
リターンキーのみの入力で変更しません
ctrl-Fをctrl-Xに変更
ctrl-Wをctrl-Gに変更

一通り変更しますと次のメッセージ を表示しPGMモード になります。

```
Beginning of PGM mode
```

例） DEL Keyを使用し、ﾃﾞﾌｫﾙﾄ 値に変換する場合

```
*PGM C <cr>

 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、ﾃﾞﾌｫﾙﾄ 値を表示します。

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

 Beginning of PGM mode
```

DEL Key 入力により
ﾃﾞﾌｫﾙﾄ 値に変換

例） ESC入力による判断の場合

```
*PGM C <cr>

 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 <cr>
 Beginning   of "HEX LOAD"  ···^G <cr>
 Beginning   of "HEX SAVE"  ···^B <cr>
 Beginning   of "SYM LOAD"  ···^A ■←ESCｷｰ

*
```

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 <cr>

 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)　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 <cr>

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 <cr>
Beginning   of "HEX LOAD"  ･･･^Z <cr>
Beginning   of "HEX SAVE"  ･･･^W <cr>
Beginning   of "SYM LOAD"  ･･･^E <cr>
Termination of "LOAD"      ･･･^A <cr>
Termination of "SAVE"      ･･･^N <cr>
Break       of "LOAD/SAVE" ･･･^B <cr>
```

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。
```

［A］ＰＧ－１０００、ＰＧ－２０００との接続

ＩＥ－７８１３０－ＲにＰＲＯＭプログラマ（ＰＧ－１０００、ＰＧ－２０００）を接続する場合のハードウェア設定、および、接続方法について説明します。

① ＩＥ－７８１３０－Ｒ、ＰＧ－１０００、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。

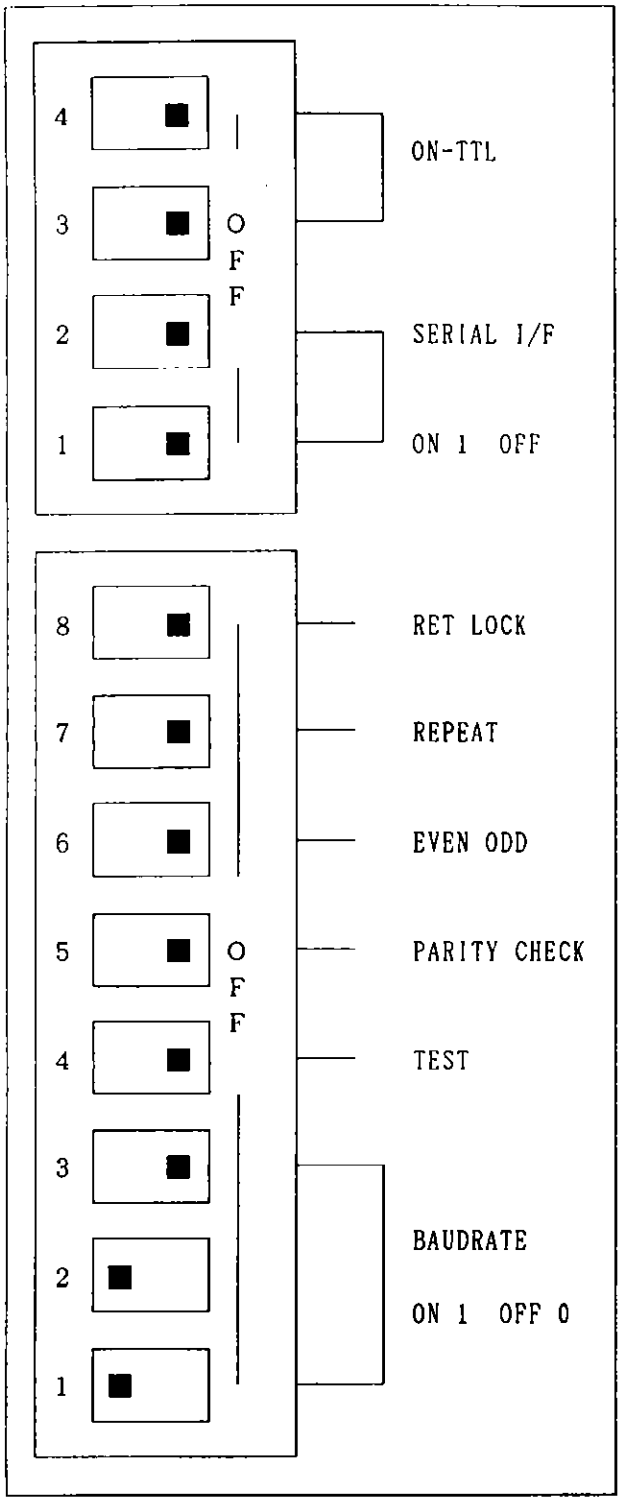
ＰＧ側のボーレートは、図4-5-1(ＰＧ－１０００の場合)、および、図4-5-2（ＰＧ－２０００の場合）を参照してください。

② ＰＧ－１０００を使用する場合は、図4-5-1 のように設定してください。

ボーレートはＩＥのシリアル・チャネル２に合わせてください。通常は9600bps にしてください。

③ ＰＧ－２０００を使用する場合は、図4-5-2 のように設定してください。

ボーレートはＩＥのシリアル・チャネル２に合わせてください。通常は9600bps にしてください。




ONを示します。


OFF を示します。

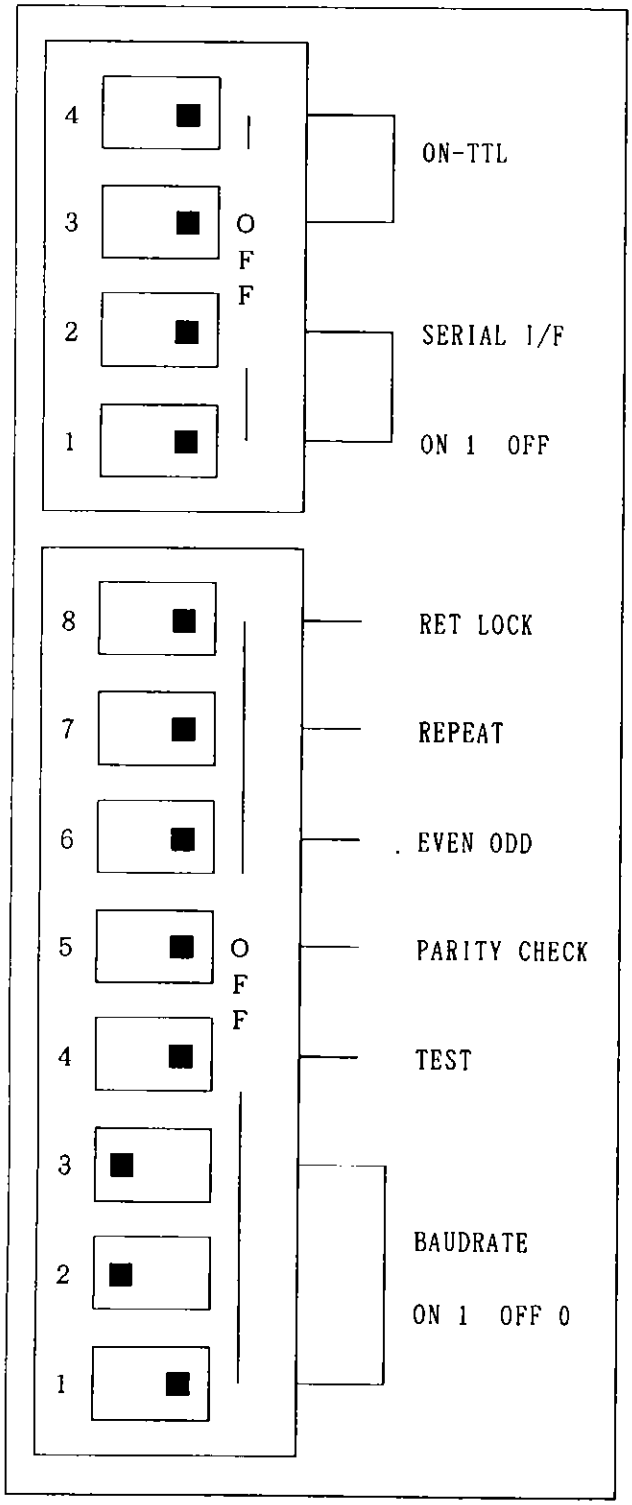
図４－５－１

表4-5-1　PG-1000 ﾎﾞｰﾚｰﾄ の設定

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

(注意1)表4-5-1 の '1　2　3' は、8 連ﾃﾞｨｯﾌﾟ･ｽｲｯﾁ
の '1　2　3' です。

(注意2)表4-5-1 に於いて 1は ON、0は OFFを示します。




ONを示します。


OFF を示します。

図４－５－２

表4-5-2　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4-5-2 の ‘1　2　3’ は、8 連ﾃﾞｨｯﾌﾟ･ｽｲｯﾁ の ‘1　2　3’ です。

(注意2)表4-5-2 に於いて 1は ON、0は OFFを示します。

④　ＩＥ－７８１３０－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<cr>
 Mode  Char = CHAR<cr>
 Baud  9600 = 9600<cr>
 Long     8 = 8 <cr>
 Par    Non = NON <cr>
 Stop     2 = 2 <cr>
*
```

初期状態では、そのまま PG-1000、PG-2000 に接続できるようになっています。PG-1000、PG-2000 とIE-78130-Rを接続する時は、必ず PG に付属している接続ｹｰﾌﾞﾙを使用してください。

⑤　ＰＧのシリアル・モード選択

(a) PG-1000 とPG-2000 とは、取扱いが異なるので注意してください。

(b) IE-78130-RとPG-1000,PG-2000 のシリアル回線の通信開始方式

・PG-1000 の場合

・PG-2000 の場合

RESET キー → □REM B キー → START キー

以下にＰＧ－１０００、ＰＧ－２０００のキー配置を示します。

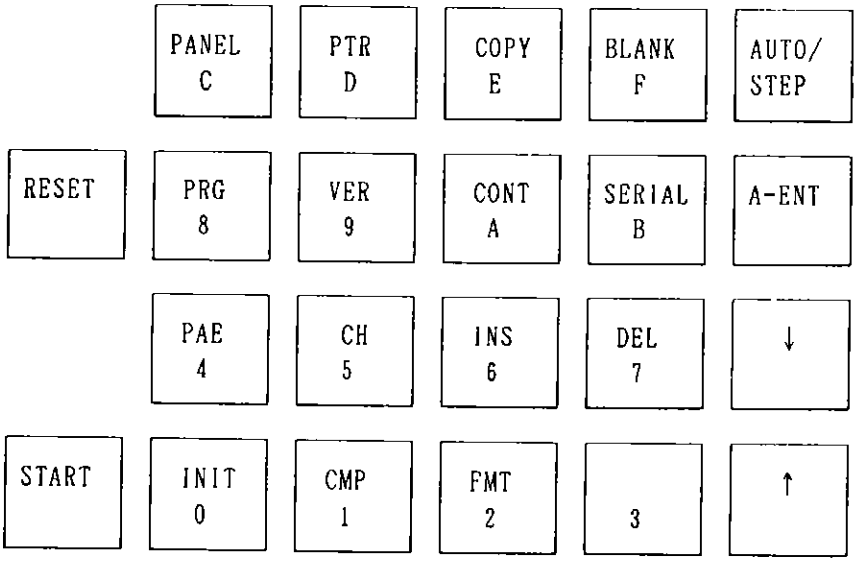


図４－５－３　ＰＧ－１０００のキー配置

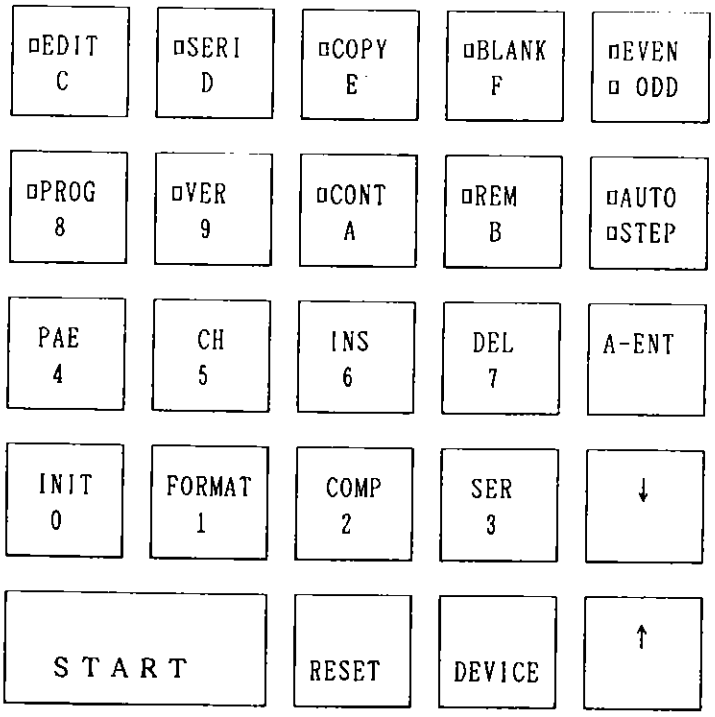


図４－５－４　ＰＧ－２０００のキー配置

［B］ＰＧ－１０００、および、ＰＧ－２０００の概要

ＰＧ－１０００、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。

ＰＧ－１０００は、ＰＧ－１００１モジュール、ＰＧ－１００２モジュール、および、ＰＧ－１００３モジュールのうち、どのモジュールを使用するかによってデータの読み出し、書き込みのできるＲＯＭがちがいます。(PG-1001,PG-1002,PG-1003 、および、PG-2000 の取扱説明書　参照)

(1)　PG-1000を使用した場合

・PG-1001 ﾓｼﾞｭｰﾙ を使用の場合

μPD8741A 、μPD8748、μPD8748H 、μPD8749H 、μPD8755A

・PG-1002 ﾓｼﾞｭｰﾙ を使用の場合

μPB403 、μPB405 、μPB406 、μPB409 、μPB417 、μPB419 、

μPB423 、μPB425 、μPB426 、μPB428 、μPB429

・PG-1003 ﾓｼﾞｭｰﾙ を使用の場合

μPD78P09R

(2)　PG-2000を使用した場合

μPD2716、μPD2732、μPD2732A 、μPD2764、μPD27128 、

μPD27C64 、μPD27256 、μPD27C256

［C］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ます。

次に、ＰＧからのプロンプト‘＊’が出力されます。

```
*PGM<cr>
 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、［A］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<cr>
 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 ＰＲＯＭプログラマ制御コマンドを起動した時は、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。（実行例は［E］を参照してください。）

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‘↑Ｚ’キーを入力することにより、通常の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。

```
*                              ← ' ctrl-Z'ｷｰ を入力
 Exit PGM mode (Y/N) Y<cr>← 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）ﾌﾟﾛﾝﾌﾟﾄ はIE（ｽﾀﾝﾄﾞ・ｱﾛﾝ・ﾓｰﾄﾞの場合）の場合も、PGの場合も' *' な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‘↑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のＬコマンド、および、Ｐコマンド（［Ｄ］②　Ｌコマンド、Ｐコマンド詳細を参照）実行中は、‘↑Ｚ’キーは無視されます。つまり、Ｌコマンド、および、Ｐコマンドを実行中にＰＧＭモードを終了することはできません。

［D］ＰＲＯＭプログラマ制御コマンド詳細

ＰＲＯＭプログラマ制御コマンドは、ＩＥ－７８１３０－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。

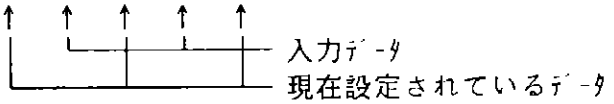
また、ＩＥ－７８１３０－Ｒ内のマッピングされたメモリの内容を、ＰＧへ転送、または、ＰＧのメモリの内容を、ＩＥ－７８１３０－Ｒのマッピングされたメモリへ転送することができます。

次に、ＰＧのコマンド一覧を示します。（表４－５－３参照）

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。

① PG－1000，PG－2000コマンド一覧

表4－5－3 PG－1000，PG－2000コマンド一覧

ｺﾏﾝﾄﾞ	形　式	コマンド　内容
A	*As,e,r <cr>	ﾊﾟﾗﾒｰﾀ の設定をします。
E	*Er <cr>	PGのﾊﾞｯﾌｧの ﾃﾞｰﾀを変更します。 形式は以下の通りです。 *Er <cr> r XX- YY XX- YY XX-<cr>  ﾃﾞｰﾀ 入力形式 ・ﾃﾞｰﾀを入力する(16進以外を入力するとその時点で ' ?' 表示) ・ｽﾍﾟｰｽ・ｷｰ入力(ﾃﾞｰﾀ変更なし)
F	*Fr,re,d<cr>	PGのﾊﾞｯﾌｧを dで初期化します。
I	*I<cr>	ｲﾝﾃﾘｼﾞｪﾝﾄ・ﾓｰﾄﾞ(ｺﾝｿｰﾙへｴｺｰﾊﾞｯｸするﾓｰﾄﾞ)へ移行します。
J	*J<cr>	PTR よりの入力 ﾊﾟﾘﾃｨ・ｴﾗｰ時、' ?' を表示します。
O	*Or,re<cr>	PGのﾊﾞ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<cr>	PROMの内容をPGのﾊﾞｯﾌｧへ転送します。
S	*S<cr>	PROMの選択
T	*T<cr>	ﾄﾗﾝｼﾞｪﾝﾄ・ﾓｰﾄﾞ(ｺﾝｿｰﾙ へｴｺｰﾊﾞｯｸしないﾓｰﾄﾞ)へ移行します。

ｺﾏﾝﾄﾞ	形　式	コマンド　内容
V	*Vs,e,r <cr>	PROMの内容とPGﾊﾞｯﾌｧの内容との比較をします。 ﾃﾞｰﾀ が違うとき、‘ ?’ を表示します。
W	*Ws,e,r <cr>	PGのﾊﾞｯﾌｧの内容をPROMに書き込みます。 書き込みｴﾗｰ 時、‘ ?’ を表示します。
Y	*Y<cr>	現在設定されているﾊﾟﾗﾒｰﾀ の表示をします。
Z	*Z<cr>	PROMにﾃﾞｰﾀ が書き込まれているかどうかをﾁｪｯｸします。 ﾃﾞｰﾀ が書き込まれていない場合はなにも表示せず、ﾃﾞｰﾀ が書き込まれている場合は ‘ ?’を表示します。
L	*Lbias<cr> Partition = i,ie<cr>	IE-78130-Rのﾏｯﾋﾟﾝｸﾞ されている i番地から、ie番地までのﾃﾞｰﾀをPGﾊﾞｯﾌｧのi+bias番地から、ie+bias 番地まで転送します。
P	*Pr,re<cr> Bias = i<cr>	PGのﾊﾞｯﾌｧの r番地からre番地までの ﾃﾞｰﾀをIE-78130-Rのﾏｯﾋﾟﾝｸﾞ されている r+i番地からre+i番地まで転送します。

注１）　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－７８１３０－Ｒ開始番地
　　ｉｅ：ＩＥ－７８１３０－Ｒ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

注２）　ＳコマンドはＰＧ－２０００のみにあります。

注３）　ＪコマンドはＰＧ－１０００のみにあります。

② Ｌコマンド・Ｐコマンド詳細

(a) Lｺﾏﾝﾄﾞ

```
*LXXXX <cr>                    ←XXXX‥‥PGのﾛｰﾄﾞ･ﾊﾞｲｱｽ
                                 YYYY‥‥IE-78130-Rﾒﾓﾘ の転送開始番地
 Partition = YYYY,ZZZZ <cr>      ZZZZ‥‥IE-78130-Rﾒﾓﾘ の転送終了番地
*                                ただしXXXX,YYYY,ZZZZは HEX数字で、
                                 下位 4桁が有効です。
```

ＩＥ－７８１３０－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ｺﾏﾝﾄﾞの使用例

```
*L0<cr>                        ←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 <cr>
*
```

② ｴﾗｰ 入力時

```
*L0<cr>                        ←Partition の入力ｴﾗｰ

 Partition = 0,0FX <cr>
 Input data error
 Partition =

*L0<cr>                        ←指定されたIEの ﾒﾓﾘが操作不可能なので
                                 ｴﾗｰ になります。
 Partition = 4000,4FFF <cr>
 Mapping error
 Partition =

*L0<cr>                        ←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 <cr>
?
*

*L8000 <cr>                    ←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時
?                                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。

(b) Pｺﾏﾝﾄﾞ

```
*PXXXX,YYYY<cr>                ←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 <cr>                ZZZZ‥‥IE-78130-R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－７８１３０－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ｺﾏﾝﾄﾞの使用例

```
*P0,FF <cr>                    ←PGの 0番地からFF番地の内容を ﾊﾞｲｱｽ0で
                                 IE-78130-Rの ﾒﾓﾘへ転送します。
 Bias = 0<cr>
 complete
*
```

② ｴﾗｰ表示

```
*P0,EF <cr>                ←Biasの入力ｴﾗｰ

 Bias = X<cr>
 Input data error
 Bias =

*P4000,4FFF<cr>            ←IE-78130-Rの ﾒﾓﾘが操作不可能なので
                             ｴﾗｰ になります。
 Bias = 100<cr>
 Non map area access
*

*P0,1EF<cr>                ←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 <cr>
 Check sum error
*

*P35,FF<cr>                ←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<cr>
 Bad character
*



*P0,8000 <cr>              ←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。

[E] PROMプログラマ制御コマンド実行例

PROMプログラマ制御コマンドの実行例を順次説明します。

（1）IE－78130－Rの電源を投入すると、次のPOWER-ONメッセージが出力されます。

```
IE-78130 for NEW FRAME Monitor Vx.x [Dd Mmm Yy]
  Copyright (C) 1989 by NEC corporation

Power on target system (Y/N) Y〈cr〉
Internal ROM size (4K,8K,12K,16K,20K,24K,28K,32K) = 16K 〈cr〉
Internal RAM size (128,256,384,512,640,768,896,1024) = 128 〈cr〉
Tracer  initialize
Breaker initialize
Do you have Memory Board on IE-78130-R ? (Y/N)= N 〈cr〉
*
```

（2）コマンド受け付け状態 '*' が出力された後、マッピングの指定を行いメモリをクリアします。

```
*MEM F 0XXX 0〈cr〉   ← 0番地から0FFF番地までのIE-78130-Rの ﾒﾓﾘを
                       0で ｸﾘｱします。
```

（3）PROMプログラマ制御コマンドを実行します。実行後、Rコマンドによりの内容をPGのバッファへ転送します。

```
*PGM 〈cr〉
 Beginning of PGM mode

*                    ←I 〈cr〉を入力
*F0,1FFF,0 〈cr〉     ←PGの ﾒﾓﾘを ｸﾘｱします。
*R 〈cr〉             ←ROM の内容をﾔｺﾏﾝﾄﾞ で設定したｱﾄﾞﾚｽ分だけPGの
                       ﾒﾓﾘへ転送します。
```

（４）読み込んだＲＯＭの内容をＩＥのメモリへ転送し、ＰＧＭコマンドを終了

します。

```
*00,3F <cr>                    ←読み込んだROM の内容を表示します。
 0000  3A 80 01 3A 81 00 20 8A 9F 81 80 06 6F 20 00 2C
 0010  00 00 B8 00 67 42 FE D8 88 E8 59 16 5F 07 FA 05
 0020  16 34 55 26 20 26 21 14 DD 00 00 00 00 00 00 00
 003F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*P0,28 <cr>                    ← PG のﾒﾓﾘ 内容をIE-78130-Rのﾒﾓﾘ に転送します。
 Bias = 0<cr>

*                              ←CTRL-Zを入力します。
 Exit PGM mode(Y/N) Y<cr>      ←PGM ﾓｰﾄﾞ を終了しますので Yを入力します。
 Termination of PGM mode       ←PGM ﾓｰﾄﾞ の終了ﾒｯｾｰｼﾞ を出力します。
*                              ←IE-78130-Rの ﾌﾟﾛﾝﾌﾟﾄが出力されます。
```

（５）ＰＲＯＭプログラマ制御コマンド終了後、データの転送されたメモリの

内容を確認します。

```
*MEM D 0,3F<cr>                ←ﾒﾓﾘ の表示
 0000   3A 80 01 3A 81 00 20 8A 9F 81 80 06 6F 20 00 2C     :..:.. .....o ..
 0010   00 00 B8 00 67 42 FE D8 88 E8 59 16 5F 07 FA 05     ....gB....Y._...
 0020   16 34 55 26 20 26 21 14 DD 00 00 00 00 00 00 00     .4U& &!.........
 0030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
*
```

（６）アセンブラ・コマンドにより、メモリの内容を変更します。

```
*ASM 0 <cr>
 0000                  MOV     0FE80H,#1H
               = MOV 0FE80H,#0H<cr>
       3A 80 00
 0003                  MOV     0FE81H,#0H
               = MOV 0FE81H,#1H<cr>
       3A 81 01
 0006                  MOV     A,0FE8AH
               = MOV A,0FEAAH<cr>
       20 AA
 0008                  CMP     A,0FE81H
               = CMP A,0FE80H<cr>
       9F 80
 000A                  BNZ     $114H
               = END <cr>
*
```

（７）変更したメモリの内容をＰＧのメモリへ転送します。

```
*PGM <cr>
 Beginning of PGM mode

*LO  <cr>                ←IE-78130-Rのﾒﾓﾘ 内容を PG に転送します。
 Partition = 0,2F<cr>
*
```

（８）未修正の箇所があったのでそれを変更後、ＰＲＯＭへ書き込んで終了します。

```
*00,3F <cr>              ←表示してみると 24 番地が未修正
 0000  3A 80 00 3A 81 01 20 AA 9F 80 80 06 6F 20 00 2C
 0010  00 00 B8 00 67 42 FE D8 88 E8 59 16 5F 07 FA 05
 0020  16 34 55 26 20 26 21 14 DD 00 00 00 00 00 00 00
 0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*E24 <cr>
 0024  20-81 26- <cr>       ← 24 番地の内容を 81Hに変更します。
*W <cr>                     ←PROMへｺﾏﾝﾄﾞで設定したｱﾄﾞﾚｽ分だけﾃﾞｰﾀ を書き込み、
*V <cr>                       および、ﾍﾞﾘﾌｧｲ をします。
*                           ←CTRL-Zを入力し、PGM ﾓｰﾄﾞ を終了します。
 Exit PGM mode(Y/N) Y<cr>
 Termination of PGM mode
*                       ←IE-78130-Rの ﾌﾟﾛﾝﾌﾟﾄが表示されます。
```

保守／廃止

４ － ５ － ２ ６　　レジスタ操作

```
REG_C [_Register  name]

REG [_D [ ┌ _ALL                    ┐ ]]
          └ _Register  name         ┘
```

Ｃ：　レジスタの変更
Ｄ：　レジスタの表示
ＡＬＬ：全レジスタバンクのレジスタ指定
Ｒｅｇｉｓｔｅｒ　ｎａｍｅ：レジスタ名を指定します

使用できるレジスタは次のとうりです。

制御レジスタ：ＰＣ，ＳＰ，ＰＳＷ
汎用レジスタ：Ｒ０，Ｒ１，Ｒ２，Ｒ３，ＲＰ２，ＲＰ３，
　　Ｘ，Ａ，Ｂ，Ｃ，ＤＥ，ＨＬ
ＰＳＷフラグ名：ＩＥ，Ｚ，ＲＢＳ１，ＡＣ，ＲＢＳ０，ＩＳＰ，ＣＹ

レジスタ操作コマンドは、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、表示をすることもできます。

（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。

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、‘ＩＥ’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、１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‘Ｘ’が指定された場合と同じになります。この場合は、‘Ｘ（Ｒ０）’から‘Ｂ（Ｒ３）’は８ビットで、‘ＤＥ（ＲＰ２）’、‘ＨＬ（ＲＰ３）’、‘ＰＣ’、‘ＳＰ’は１６ビットで変更されます。

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、‘〈cr〉’を入力します。

例）

```
*REG C <cr>                          ←すべての汎用ﾚｼﾞｽﾀの変更
 X(R0)       00 = 11<cr>
 A(R1)       11 = 22<cr>
 C(R2)       22 = 33<cr>
 B(R3)       33 = <cr>               ←ﾚｼﾞｽﾀの変更なし
 DE(RP2)   1000 = 2000<cr>
 HL(RP3)   2000 = 3000<cr>
 PC        0100 = 0130<cr>
 SP        FFFF = <cr>
*

*REG C PSW <cr>                      ←PSW のﾌﾗｸﾞ 名単位に変更
 IE           1 = 0 <cr>
 Z            1 = 0 <cr>
 RBS1         1 = <cr>                ←変更なし
 AC           0 = 1 <cr>
 RBS0         1 = 0 <cr>
 CY           0 = 1 <cr>
*


*REG C SP<cr>                        ←制御ﾚｼﾞｽﾀ'SP'を変更
 SP        FFFF = FF20<cr>
*

*REG C PC<cr>                        ←制御ﾚｼﾞｽﾀ'PC'を変更
 PC        0130 = 1000<cr>
 SP        FF20 = <cr>
*
```


```
*REG C RP2 <cr>              ←汎用ﾚｼﾞｽﾀ(ﾍﾟｱ･ﾚｼﾞｽﾀ) の変更
 DE(RP2)  2000 = 4000<cr>
 HL(RP3)  3000 = 5000<cr>
 PC       1000 = <cr>
 SP       FF20 = <cr>
*

*REG C R0<cr>                ←汎用ﾚｼﾞｽﾀの変更
 X(R0)      11 = 30<cr>
 A(R1)      22 = 40<cr>
 C(R2)      33 = 50<cr>
 B(R3)      33 = . <cr>
*
```

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 DE<cr>
 DE(RP2)  4000 = 1000<cr>
 HL(RP3)  5000 = . <cr>
*

*REG C X <cr>
 X(R0)      30 = 20<cr>
 A(R1)      40 = 30<cr>
 C(R2)      50 = . <cr>
*
```

'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できます。

```
*REG C IE<cr>                ← 'PSW ' のﾌﾗｸﾞ 名で変更
 IE          0 = 1 <cr>
*
```

（２）　レジスタの表示

```
ＲＥＧ [_D[_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、すべてのレジスタの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 <cr>              ←すべてのﾚｼﾞｽﾀの内容を表示
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   0   0   0   0    1   0  15    FF    7F    32    0000    5022   0149  FFFF

*

*REG D PSW <cr>          ← 'PSW ' のすべてのﾌﾗｸﾞ の内容を表示
  IE   Z RBS1 AC RBS0  ISP CY
   0   0   0   0   0    1   0
*

*REG D PC<cr>            ← 'PC' の内容を表示
 PC       0149
*

*REG D R0<cr>            ← 'R0' の内容を表示
 X(R0)      15
*

*REG D HL<cr>            ← 'HL' の内容を表示
 HL(RP3)  5022
*

*REG D Z <cr>            ← 'Z ' ﾌﾗｸﾞ の内容を表示
 Z          0
*
```

保守／廃止

４ — ５ — ２ ７　　リ セ ッ ト

```
RES [_H]
```

H　　：IE-78130-Rすべてのﾘｾｯﾄ

リセット・コマンドは、エバ・チップのみのリセット、あるいはＩＥ－７８１３０－Ｒ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、エバ・チップだけをリセットします。オペランドに“H”を指定した場合、ＩＥ－７８１３０－Ｒすべてをリセットします。

ただし、システム・モードかスタンド・アロン・モードかの状態とシンボル情報は保持されます。

なお、アクティブ・モード中に本コマンドを実行すると、エミュレーションをブレークして通常モードへ戻ります。

例）

```
*RES H <cr>          ←IE-78130-Rすべてﾘｾｯﾄする
 IE-78130-Rのｽﾀｰﾄ・ｱｯﾌﾟ・ﾒｯｾｰｼﾞ
*
```

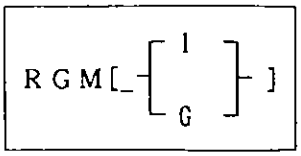
```
*RES <cr>            ←ｴﾊﾞ・ﾁｯﾌﾟだけをﾘｾｯﾄします。
*
```

アクティブ・モード時

```
#RES <cr>            ←ｴﾊﾞ・ﾁｯﾌﾟだけをﾘｾｯﾄする
*                    ←通常ﾓｰﾄﾞ に戻ります
```

エバ・チップだけをリセットします。ＩＥ－７８１３０－Ｒは影響を受けません。

４－５－２８　レジスタ・モード設定



I ：機能ﾚｼﾞｽﾀ指定

G ：汎用ﾚｼﾞｽﾀ指定

レジスタ・モード設定コマンドは、逆アセンブル・リスト中のレジスタ表示を、機能レジスタ、または、汎用レジスタのどちらで表示するのかを指定します。

また、オペランドが省略された場合、現在選択されているレジスタ・モードを表示します。

例）

```
*RGM G <cr>                 ←ﾚｼﾞｽﾀ･ﾓｰﾄﾞとして汎用ﾚｼﾞｽﾀを設定
*

*RGM <cr>                   ←ｵﾍﾟﾗﾝﾄﾞ を省略した場合
 General register mode      ←現在選択されているﾚｼﾞｽﾀ･ﾓｰﾄﾞを表示
*
```

また、現在選択されているのが機能ﾚｼﾞｽﾀ指定だった場合は、以下のようになります。

```
*RGM <cr>
 Implied register mode
*
```

４ － ５ － ２ ９　エミュレーション操作

```
RUN_B[_word]              (ブレークつ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][,byte]]     (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 { = / > / < / => / >= / =< / <= / <> / >< } { bit / mask / wmask }

ﾏｽｸ･ﾃﾞｰﾀ に特殊数値表現を使用する場合は '=' '<>'、'><' 以外の条件の指定はできません。
また、PSWﾌﾗｸﾞ名を指定した場合の ﾋﾞｯﾄ･ﾃﾞｰﾀは 0 または 1 のみです。

word	: 実行ｽﾀｰﾄ･ｱﾄﾞﾚｽ､ﾄﾚｰｽ数
byte	: 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
bit	: ﾋﾞｯﾄ･ﾃﾞｰﾀ
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, RP0, RP1, RP2, RP3, X, A, B, C, D, E, H, L, AX, BC, DE, HL, PC, SP, IE, Z, RBS1, AC, RBS0, ISP, 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。

保守／廃止

注意：ＩＥ－７８１３０－Ｒでは、μＰＤ７８１３Ｘの性格上、以下のような使用上の注意が必要です。

・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、内部ＲＯＭで２バイト命令が連続している時が最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。

（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。

プログラムが実行されると、ブレークなしリアルタイム実行中のコマンド入力要求プロンプト‘#’を表示し、エミュレーション中コマンド入力待ちになります。このとき入力可能なコマンドを示します。

BRA, BRD, BRE, BRS, BR0, BR1, BR2, BR3, BRM, <COM>, <DIR>, <DOS>, <HIS>
*
<HLP>, <LST>, MAT, MOD, <STR>, SUF, SYM, TRM, DLY, TRP, TRD, TRF,

RGM, (TRG), (STP)

* ：SYM_S, SYM_Lｺﾏﾝﾄﾞ を除く

< >：ｼｽﾃﾑ･ﾓｰﾄﾞ･ｺﾏﾝﾄﾞ であり、ｽﾀﾝﾄﾞ･ｱﾛﾝ･ﾓｰﾄﾞ に於いては入力不可能なｺﾏﾝﾄﾞです。

( )：ｴﾐｭﾚｰｼｮﾝ実行中のｺﾏﾝﾄﾞ入力でのみ入力可能なｺﾏﾝﾄﾞです。

（ａ）　トレーサ再起動コマンド

ＴＲＧ

トレーサ起動コマンドは、ブレークなしリアルタイム実行に於いて、カレントのトレース・モード、トレース・トリガ条件（ブレーク条件）、および、ディレイ・カウンタ値で新たにトレーサを起動します。

（ｂ）　Ｅ＿ＣＰＵストップ・コマンド

ＳＴＰ

エミュレーションＣＰＵストップ・コマンドは、ブレークなしリアルタイム実行に於いて、強制ブレークさせるコマンドです。
（ＥＳＣキーでのブレークはできません。）
本コマンドを入力することにより、エミュレーション中コマンド入力から通常のコマンド入力に移行します。

ＴＲＧ、ＳＴＰコマンドは、エミュレーション中コマンド入力待ち‘#’、‘ｎ]’の時のみ有効で、通常のコマンド入力待ち‘＊’、‘ｎ>’の時に実行しますと、エラー・メッセージ‘Unexecutable Command’を表示します。

スタンド・アロン・モード時

```
*RUN N 200 <cr>          ←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
#                         ←ｴﾐｭﾚｰｼｮﾝ中ｺﾏﾝﾄﾞ入力待ち
  ↑
  └─  ｶｰｿﾙの位置
```

ユーザ・プログラムを実行し、エミュレーション中コマンド入力要求プロンプトを表示します。この時点でトレース結果の表示、および、トレース・トリガ検出条件の変更が可能です。

ユーザ・プログラムの実行を停止させる場合は、‘ＳＴＰ’コマンドを入力します。以下にその例を示します。

```
*RUN N 200 <cr>
 User-system Vcc-ON    Emulation start at 0200
#SUF <cr>
 H
#STP <cr>                    ←ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ の実行停止
 ESC break terminated        ←ﾌﾞﾚｰｸ･ﾒｯｾｰｼﾞの表示
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   1   0   0   0    1   0  0F    31    00    00    0000    1400   0253  FFFF
*
```

‘ＳＴＰ’コマンドによってユーザ・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。

システム・モード時

```
 1>RUN N 200 <cr>
   User-system Vcc-ON     Emulation start at 0200
 1]DIR <cr>
```

```
   ﾃﾞｨﾚｸﾄﾘ 表示
```

```
  1]STP <cr>
    ESC break terminated
    IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
     0   1   0   0   0    1   0  0F    31    00    00    0000    1400   0253  FFFF
 1>
```

エミュレーション実行スタート・メッセージ、および、ブレーク・メッセージは、スタンド・アロン・モードでの各メッセージと同じです。

また、エミュレーション実行中に於ける各コマンド入力での結果表示等は、通常のコマンド入力での各コマンド実行と同一です。

（２）　ブレークつ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 ｺﾏﾝﾄﾞ)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、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 〈cr〉              ←100 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 0100
 Standard break  terminated  ←ﾌﾞﾚｰｸ条件が一致、ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ の実行を停止
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   1   0   0   0    1   0  00    FF    00    00    0000    2000   0107  FFFF
 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。

```
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   1   0   0   0    1   0  00    FF    00    00    0000    2000   0107  FFFF
 One step emulation standby  ←〈cr〉
  Frame Address Label  Mnemonic
 0000   010A          MOV     A,#0H
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   1   0   0   0    1   0  01    00    00    00    0000    2000   0109  FFFF
 One step emulation standby  ←ESC ｷｰ
*
```

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、実行アドレス、命令（ニーモニック）、ＰＳＷ、レジスタ内容を表示します。

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、フェイル・セーフ・ブレークが発生した場合は、ワン・ステップ実行モードを強制的に終了します。

（３）　ステップ数指定リアルタイム実行

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

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。また、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。

指定された実行ステップ数の実行が終了した場合は、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。

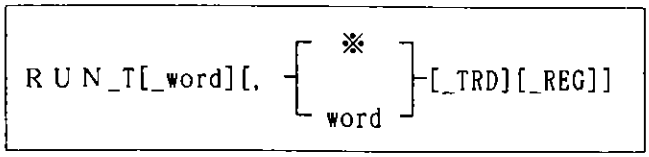
例）

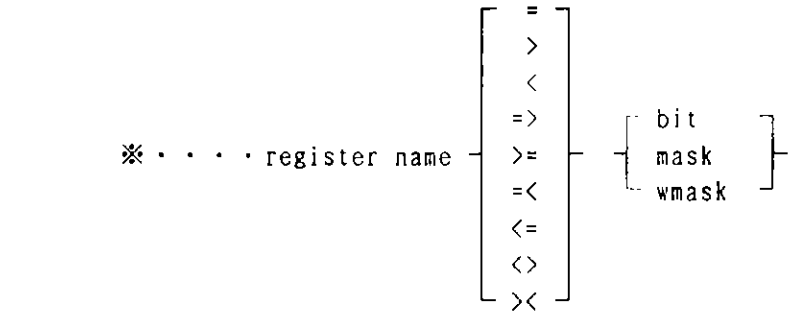
```
*RUN S 100,50T <cr>          ←100 番地からﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑ を50Tｽﾃｯﾌﾟ 実行
 User-system Vcc-ON     Emulation start at 0100
 Step break  terminated  ←指定ｽﾃｯﾌﾟ数を実行し停止
  IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0   1   0   0   0    1   0  13    24    00    F6    FFFF    1234   0137  FFFF
 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





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はできません。

マスク表現を指定した場合は、‘= ’、‘<>’、‘><’のみ有効であり、他の条件は設定できません。

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<cr>←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 Address Label   Mnemonic
  0000    010A          MOV      A,[HL]
  IE    Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0    1   0    0    0    1    0  02    02    20    30    0500    0600   010A  FFFF

  Frame Address Label   Mnemonic
  0000    010B          DEC      A
  IE    Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0    0   0    0    0    1    0  02    01    20    30    0500    0600   010B  FFFF

   terminated
 One step emulation standby←<cr>
  Frame Address Label   Mnemonic
  0000    010C          MOV      [HL],A
  IE    Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0    0   0    0    0    1    0  02    01    20    30    0500    0601   010C  FFFF
 One step emulation standby←ESCｷｰ
*


*RUN T 10A,R0=1<cr>          ←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←<cr>
  Frame Address Label   Mnemonic
  0000    010C          MOV      [HL],A
  IE    Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
   0    0   0    0    0    1    0  02    01    20    30    0500    0601   010C  FFFF
 One step emulation standby←ESCｷｰ
*
```

４－５－３０　オブジェクト／ディバグ環境のセーブ

```
                 ┌TTY1 ┐
SAV[ _┤         ├] [_partition][_partition]・・・・・[_partition]
                 └TTY2 ┘
SAV _file [_partition][_partition]・・・・・[_partition][_C][_D]
```

TTY1	: ｼﾘｱﾙ･ﾁｬﾈﾙ1
TTY2	: ｼﾘｱﾙ･ﾁｬﾈﾙ2
file	: ﾌｧｲﾙ名
partition	: ﾒﾓﾘ 範囲（partition は最大５つまで指定できます。）
C	:ｵﾌﾞｼﾞｪｸﾄ指定
D	:ﾃﾞｨﾊﾞｸﾞ環境指定

オブジェクト、ディバグ環境セーブ･コマンドは、マッピングされたメモリの内容あるいはディバグ環境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形式が異なります。

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･コードあるいはディバグ環境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／ディバグ環境セーブ

```
ＳＡＶ__file [_partition][_partition]･････[_partition][_C][_D]
```

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したメモリの内容およびディバグ環境をホスト･マシンの指定したファイルへ出力します。

コマンド行の最後に‘Ｃ’を指定した場合は、オブジェクトのセーブだけを、‘Ｄ’を指定した場合は、ディバグ環境のセーブだけを、‘Ｃ’及び‘Ｄ’の指定が無い場合は最初にオブジェクトをセーブし、次にディバグ環境をセーブします。

オブジェクト及びディバグ環境双方を一度にセーブする場合、ファイル名指定で拡張子を付加することは出来ません。

その場合、オブジェクト･ファイルは‘ＨＥＸ’　ディバグ環境は‘ＤＢＧ’となります。

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範囲のメモリ内容がすべて出力されます。

この場合は、内部ＲＡＭ（ＦＥ８０Ｈ～ＦＥＦＦＨを除く）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、マッピングされていなければなりません。

ディバグ環境としてセーブする内容は、次のコマンドで設定した内容です。

ＢＲＡ，ＢＲＤ，ＢＲＥ，ＢＲＳ，ＢＲＭ，ＢＲ０，ＢＲ１，ＢＲ２，ＢＲ３，

ＣＬＫ，ＤＬＹ，ＭＡＰ，ＭＯＤ，ＰＧＭ，ＳＵＦ，ＴＲＭ，ＴＲＦ

例）

```
1)SAV B:SAMPLE.HEX C〈cr〉          ←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 C〈cr〉  ←指定範囲(0～0FFF、2000～27FF番地）の
  object save complete                      内容をｾｰﾌﾞ する
1)
```

```
1)SAV B:SAMPLE.HEX C〈cr〉
  File already exists. Delete ?(Y or N): Y〈cr〉
  object save complete
1)
```

ﾄﾞﾗｲﾌﾞ Bに SAMPLE.HEX というR/W属性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。

```
1)SAV B:SAMPLE.DBG D〈cr〉          ←ﾃﾞｨﾊﾞｸﾞ環境だけをｾｰﾌﾞします。
  debug condition save complete
1)
```

```
1)SAV B:SAMPLE.HEX〈cr〉
  File already exists.
1)
```

ﾌｧｲﾙの属性が R/Oの場合、上記のようなﾒｯｾｰｼﾞ を表示し、ｺﾏﾝﾄﾞは無視されます。

```
1)SAV B:SAMPLE〈cr〉                ←ｵﾌﾞｼﾞｪｸﾄとﾃﾞｨﾊﾞｸﾞ環境をｾｰﾌﾞします。
  object save complete
  debug condition save complete
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、指定したシリアル・チャネルにパーティションで指定した範囲のオブジェクト・コードを出力することができます。

シリアル・チャネルが省略された場合、シリアル・チャネル１が指定されます。パーティションが省略された場合、マッピングされている範囲すべてが指定されます。

この場合は、内部ＲＡＭ空間（ＦＥ８０Ｈ～ＦＥＦＦＨを除く）の内容も同時に出力されます。

例）

```
*SAV TTY1 0XXX 1800,1FFF<cr>     ←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<cr>     ←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<cr>  ← Y<cr>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<cr>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 － ３ １　特殊レジスタ操作

```
SPR_C [_register name]

SPR [_D [_register name]]
```

C：特殊レジスタの変更
D：特殊レジスタの表示
register name：レジスタ名を指定します

使用出来る特殊レジスタ名を以下に示します。

```
                                  *    *    *    *    *    *    *    *
P0, P1, (P2), P3, P4, P5, P6, P7, CR00, CR01, CR02, CR10, CR11, (CR12), (CPT0), (CPT1),
     *    *              *    *    *    *    *
(CPT2H), (CPT3), (CPT2L), CR20, (TM0), (TM1), (FRC), (TM2), (TM3), CR30, (CPT30), P01,
                                       *    *
P0H, <ECC1>, <ECC0>, (EC), (ADCR), <PWM0>, <PWM1>, SBIC, SIO,

EXTSFR0, EXTSFR1, EXTSFR2, EXTSFR3, EXTSFR4, EXTSFR5, EXTSFR6, EXTSFR7, EXTSFR8,

EXTSFR9, EXTSFR10, EXTSFR11, EXTSFR12, EXTSFR13, EXTSFR14, EXTSFR15
```

＊で示したレジスタは、１６ビット・レジスタ
その他は、８ビット・レジスタ
（）で示したレジスタは、読み出し専用レジスタ
＜＞で示したレジスタは、書き込み専用レジスタ

特殊レジスタ操作コマンドは、８ビット・レジスタは８ビット単位で、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、書き込み可能なすべての特殊レジスタを‘Ｐ０’から‘ＥＸＴＳＦＲ１５’の順番に変更することができます。

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。変更しない場合は、リターン・キーを入力します。また、変更を途中で終了する場合は、‘.’（ピリオド）を入力します。

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、読み出し可能な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。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）

特殊レジスタの変更

```
*SPR C<cr>                          ←特殊ﾚｼﾞｽﾀすべての変更
 P0          77 = 88<cr>
 P1          66 = <cr>              ←変更なし
 P3          99 = 00<cr>
 P4          00 = 55<cr>
 P5          11 = 10<cr>
 P6          12 = 00<cr>
 P7          10 = 11<cr>
        .
        .
        .
 EXTSFR14    24 = <cr>
 EXTSFR15    25 = <cr>
*

*SPR C P6<cr>                       ← ‘P6’ ﾚｼﾞｽﾀから変更
 P6          00 = 0FH<cr>
 P7          11 = . <cr>
*

*SPR C CR12<cr>                     ←読み出し専用ﾚｼﾞｽﾀを変更
 Read only
*
```

特殊ﾚｼﾞｽﾀの表示

```
*SPR D<cr>                          ←読み出し可能な特殊ﾚｼﾞｽﾀの表示
 P0      P1      P2      P3      P4      P5      P6      P7      CR00    CR01    CR02
 88      66      22      00      55      10      FE      11      1111    2222    3333

 CR10    CR11    CR12    CPT0    CPT1    CPT2H   CPT3    CPT2L   CR20    TM0     TM1
 4444    5555    6666    7777    8888    9999    AAAA    BB      CCCC    DDDD    EEEE

 FRC     TM2     TM3     CR30    CPT30   P0L     P0H     EC      ADCR    SBIC    SIO
 FFFF    0000    11      22      30      40      50      60      70      80      90

 EXTSFR0  EXTSFR1  EXTSFR2  EXTSFR3  EXTSFR4  EXTSFR5  EXTSFR6  EXTSFR7
 01       02       03       04       05       06       07       08

 EXTSFR8  EXTSFR9  EXTSFR10 EXTSFR11 EXTSFR12 EXTSFR13 EXTSFR14 EXTSFR15
 09       0A       0B       0C       0D       0E       24       25
*


*SPR D ECC1<cr>                     ←書き込み専用ﾚｼﾞｽﾀの内容を表示
 ECC1       --
*

*SPR D P4<cr>                       ← 'P4' の内容を表示
 P4         55
*
```

保守／廃止

４－５－３２　エミュレーション・ＣＰＵ停止

ＳＴＰ

‘ＳＴＰ’コマンドは、アクティブ･モード時において、エミュレーションＣＰＵの実行を停止するコマンドです。

本コマンド入力により、アクティブ･モードから通常コマンド待ちモードへ移行します。

ＳＴＰコマンドは、エミュレーション中のコマンド入力待ち‘#’或は‘ｎ］’の時のみ有効なコマンドで、通常のコマンド入力待ち‘＊’或は、‘ｎ＞’の時に入力するとエラー･メッセージ‘Unexecutable Command’　が表示されます。

保守／廃止

４－５－３３　入力ディバイス・リダイレクト

```
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

入力ディバイス・リダイレクト・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。

入力ディバイス・リダイレクト・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き換えられます。パラメータは、最大４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４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、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時に使用します。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時に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時に使用します。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時に、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<cr>        ←ﾄﾞﾗｲﾌﾞ Bの SAMPLE.STR を入力ﾌｧｲﾙに指定
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>
```

ﾌｧｲﾙからｺﾏﾝﾄﾞ、および、ﾃﾞｰﾀ 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。

```
LOD SAMPLE
MEM D 0X
MEM F 0XX 00
MEM D 0XX
```


```
1>STR SAMPLE.STR 0XX 0XX<cr>  ←ﾊﾟﾗﾒｰﾀ を指定した場合
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
  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。

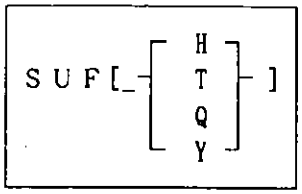
```
LOD SAMPLE
MEM D 0X
MEM F $0 00              ←仮ﾊﾟﾗﾒｰﾀ 1
MEM D $1                 ←仮ﾊﾟﾗﾒｰﾀ 2
```

```
1>STR B:SAMPLE.STR<cr>
  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。

保守／廃止

４－５－３４　サフィックス指定



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。

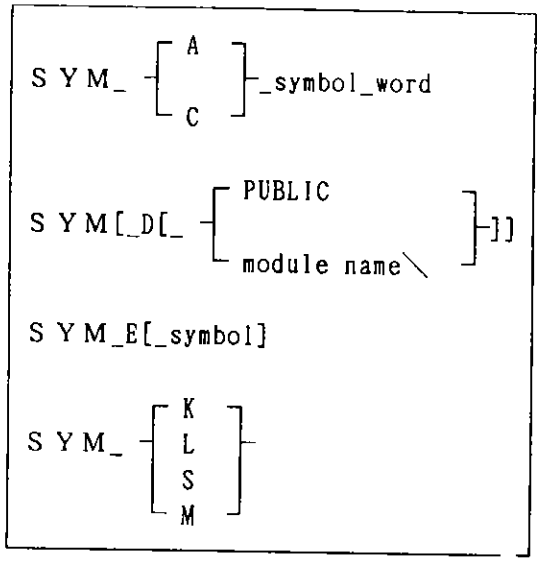
オペランドのサフィックス指定が省略された場合は、現在設定されているサフィックスが表示されます。

例）

```
*SUF H <cr>          ←ｻﾌｨｯｸｽを16進数に設定
*

*SUF <cr>            ←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)
*
```

保守／廃止

４ － ５ － ３５　シンボル操作



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
S	：追加ｼﾝﾎﾞﾙのｾｰﾌﾞ
M	：ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙの指定
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・モードの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つけられます。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 <cr>
*
```

ｼﾝﾎﾞﾙ値が1000で“SYMBOL01”というｼﾝﾎﾞﾙ名を定義します。

（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変更はできません。

例）

```
*SYM C SYMBOL01 2000 <cr>
*
```

“SYMBOL01”というｼﾝﾎﾞﾙのｼﾝﾎﾞﾙ値を2000に変更します。“SYMBOL01”は、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でなければなりません。

（３）　シンボルの表示

```
┌──────┐
│ SYM[_D] │                    ←ｽﾀﾝﾄﾞ･ｱﾛﾝ･ﾓｰﾄﾞのｺﾏﾝﾄﾞ形式
└──────┘

                  ┌ PUBLIC          ┐
SYM[_D[_ ─┤                 ├─]] ←ｼｽﾃﾑ･ﾓｰﾄﾞのｺﾏﾝﾄﾞ形式
                  └ module name＼   ┘
```

シンボルの表示コマンドは、定義されているシンボル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、または、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）スタンド・アロン・モードの場合

・追加されたｼﾝﾎﾞﾙがある場合

```
*SYM D <cr>
 module : IESYMBOL ←追加ｼﾝﾎﾞﾙのﾓｼﾞｭｰﾙ 名
    1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
    1100 SYMBOL05      2100 SYMBOL08
*
```

・追加されたｼﾝﾎﾞﾙがない場合

```
*SYM D <cr>
 no symbol of append      ←追加ｼﾝﾎﾞﾙがない場合のﾒｯｾｰｼﾞ
*
```

例）システム・モードの場合

・追加されたｼﾝﾎﾞﾙがある場合

```
1>SYM D <cr>
  module : IESYMBOL
     1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
     1100 SYMBOL05      2100 SYMBOL08
  module : PUBLIC
     0000 XSYM0001      0100 XSYM0002      0200 XSYM0003      0300 XSYM0004
     0400 XSYM0005      0500 XSYM0006      0600 XSYM0007      0700 XSYM0008
  module : MOD00
     0A00 LSYM0001      0B00 LSYM0002      0C00 LSYM0003      0D00 LSYM0004
     0E00 LSYM0005      0F00 LSYM0006
1>
```

・追加されたｼﾝﾎﾞﾙがない場合

```
1>SYM D <cr>
  module : PUBLIC
     0000 XSYM0001      0100 XSYM0002      0200 XSYM0003      0300 XSYM0004
     0400 XSYM0005      0500 XSYM0006      0600 XSYM0007      0700 XSYM0008
  module : MOD00
     0A00 LSYM0001      0B00 LSYM0002      0C00 LSYM0003      0D00 LSYM0004
     0E00 LSYM0005      0F00 LSYM0006
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<cr>
  module : IESYMBOL
     1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
     1100 SYMBOL05      2100 SYMBOL08
  module : PUBLIC
     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 \<cr>
  module : IESYMBOL
     1000 SYMBOL01      2000 SYMBOL02      3000 SYMBOL03      4000 SYMBOL04
     1100 SYMBOL05      2100 SYMBOL08
  module : MOD00
     0A00 LSYM0001      0B00 LSYM0002      0C00 LSYM0003      0D00 LSYM0004
     0E00 LSYM0005      0F00 LSYM0006
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<cr>          ←" SYMBOL01"を削除
*

*SYM E <cr>                  ←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
```

（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 <cr>              ←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、“ＩＥ７８１３０．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“ＩＥ７８１３０．ＳＹＭ”というファイル名を持つファイルを作成しないでください。

例）

```
1>SYM L <cr>  ←正常にﾛｰﾄﾞが終了
1>

1>SYM L <cr>
  append symbol file not found  ←ｶﾚﾝﾄ･ﾄﾞﾗｲﾌﾞ に IE78130.SYMというﾌｧｲﾙが
1>                                ない場合のﾒｯｾｰｼﾞ
```

（７）　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、“ＩＥ７８１３０．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“ＩＥ７８１３０．ＳＹＭ”というファイル名を持つファイルを作成しないでください。

例）

```
1>SYM S <cr>                    ←ｾｰﾌﾞ が正常に終了
1>

1>SYM S <cr>
  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 <cr>
  File already exists. Delete ?(Y or N): Y<cr>
1>
```

IE78130.SYM というR/W属性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 IE78130.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 IE78130.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。

ﾌｧｲﾙの属性が　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 <cr>
  File already exists.
1>
```

（８）　カレント・モジュールの指定

ＳＹＭ_Ｍ

カレント・モジュール指定コマンドは、モジュール名をカレントなモジュール名として指定します。

カレント・モジュールとして指定されたモジュールのモジュール内ローカル・シンボルは、このコマンド以降のコマンド／データ入力に於いて、モジュール名を省略して入力することができます。

指定するモジュールの後には‘＼’が必要です。

例）

```
1>SYM M <cr>
          = MOD01 ＼<cr>     ←ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙとして“MOD01 ”を指定
1>

1>SYM M <cr>
  MOD01 ＼  = MOD02 ＼<cr>     ←ｶﾚﾝﾄ･ﾓｼﾞｭｰﾙを“MOD01 ”から“MOD02 ”
1>                               に変更
```

保守／廃止

４－５－３６　トレース表示

```
TRD[_┌ F ┐][_ALL][_┌ $Q ┐]]
     └ I ┘          └ $F ┘
```

F　：ﾌﾚｰﾑ･ﾓｰﾄﾞ 表示

I　：ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ表示

ALL　：ﾄﾚｰｽ･ﾃﾞｰﾀ すべてを表示

$Q　：ｲﾍﾞﾝﾄ 検出の対象となったﾄﾚｰｽ･ﾃﾞｰﾀ（以後 ﾄﾘｶﾞ･ﾃﾞｰﾀという）と、TRFｺﾏﾝﾄﾞで設定された条件を満足するﾄﾚｰｽ･ﾃﾞｰﾀ（以後 !ﾃﾞｰﾀ という）のみを表示

$F　：ﾄﾘｶﾞ･ﾃﾞｰﾀと!ﾃﾞｰﾀの近傍のﾄﾚｰｽ･ﾃﾞｰﾀを表示

トレース表示コマンドは、トレース・データをフレーム・モード、あるいはインストラクション・モードで表示します。サブ・コマンド省略時は、インストラクション・モードで表示します。ただし、データ・アクセス・トレースの場合は、インストラクション・モードでの表示は出来ません。

”ＡＬＬ”が指定された場合は、表示後コマンド待ちになりますが、指定のない場合は１１行分のトレース・データ（以後、当コマンドの説明内ではページという）を表示し、表示促進入力待ちになります。（例参照）

トリガ・データを表示する場合は、行の先頭に”Ｔ”を付加し、！データを表示する場合は、行の先頭に”！”を付加します。

表示促進入力待ちに於ける入力と動作の関係は以下の通りです。

入　力	ｵﾍﾟﾗﾝﾄﾞ	動　　作
”Ｎ”	無し	最新のﾍﾟｰｼﾞ を表示します。
	ＳＱ	最新の!ﾃﾞｰﾀ･ﾍﾟｰｼﾞ を表示します。
	ＳＦ	最新の!ﾃﾞｰﾀを中心とするﾍﾟｰｼﾞ を表示します。
”Ｏ”	無し	最古のﾍﾟｰｼﾞ を表示します。
	ＳＱ	最古の!ﾃﾞｰﾀ･ﾍﾟｰｼﾞ を表示します。
	ＳＦ	最古の!ﾃﾞｰﾀを中心とするﾍﾟｰｼﾞ を表示します。
”Ｔ”	共通	ﾄﾘｶﾞ･ﾃﾞｰﾀ を中心とするﾍﾟｰｼﾞ を表示します。
”＋” 又は ｃｒ	無し	直前に表示したﾍﾟｰｼﾞ の次に新しいﾍﾟｰｼﾞ を表示します。
	ＳＱ	直前に表示した!ﾃﾞｰﾀ･ﾍﾟｰｼﾞの次に新しい!ﾃﾞｰﾀ･ﾍﾟｰｼﾞ を表示します。
	ＳＦ	直前に表示したﾍﾟｰｼﾞ の次に新しいﾄﾘｶﾞ･ﾃﾞｰﾀ 又は、!ﾃﾞｰﾀを中心とするﾍﾟｰｼﾞ を表示します。
”－”	無し	直前に表示したﾍﾟｰｼﾞ の次に古いﾍﾟｰｼﾞ を表示します。
	ＳＱ	直前に表示した!ﾃﾞｰﾀ･ﾍﾟｰｼﾞの次に古い!ﾃﾞｰﾀ･ﾍﾟｰｼﾞ を表示します。
	ＳＦ	直前に表示したﾍﾟｰｼﾞ の次に古いﾄﾘｶﾞ･ﾃﾞｰﾀ 又は、!ﾃﾞｰﾀを中心とするﾍﾟｰｼﾞ を表示します。
”．”	共通	ﾄﾚｰｽ表示ｺﾏﾝﾄﾞ を終了します。

注）

・インストラクション・モードでのトリガ・データは最新方向に最も近いＭ１に相当するインストラクションとして表示されます。このため、トリガ・データが存在しても表示できない場合があります。

・インストラクション・モードでは、ベクタ・フレームと無効フレームは！データの対象とはなりません。このため、フレーム・モードで表示可能でもインストラクション・モードでは表示できない場合があります。

例1) $Q ,$F が指定されない場合

・フレーム・モードでの表示

```
*TRD F <cr>                    ←ﾌﾚｰﾑ･ﾓｰﾄﾞでﾄﾚｰｽ･ﾃﾞｰﾀすべてを表示
  Frame  Status  Adderss  Data    7--EX--0
  0000     M1     0100     BB      00000000
  0001     OP     0101     03      00000000
  0002     M1     0102     B9      00000000
  0003     OP     0103     00      00000000
  0004     M1     0104     66      00000000
  0005     OP     0105     00      00000000
  0006     OP     0106     10      00000000
  0007     M1     0107     55      00000000
  0008     M1     0108     C1      00000000
  0009     M1     0109     CB      00000000
  0010     M1     010A     80      00000000
  Total frame = 0962   (N/O/T/+/cr/-/Frame No./.) ? _
                                                    ↑
                                                    └ｷｰ入力待ち
```

・インストラクション・モードでの表示

```
*TRD I <cr>                    ←ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ でﾄﾚｰｽ･ﾃﾞｰﾀすべてを表示
  Frame  Address  Label  Mnemonic
  0000    0100           MOV     B,#3H
  0002    0102           MOV     A,#0H
  0004    0104           MOVW    HL,#1000H
  0007    0107           MOV     [HL],A
  0008    0108           INC     A
  0009    0109           DEC     B
  0010    010A           BNZ     $107H
  0013    0107           MOV     [HL],A
  0014    0108           INC     A
  0015    0109           DEC     B
  0016    010A           BNZ     $107H
  Total frame = 0962   (N/O/T/+/cr/-/Frame No./.) ? _
                                                    ↑
                                                    └ｷｰ入力待ち
```

例2) $Q が指定された場合

・フレーム・モードでの表示

```
*TRD F $Q<cr>              ←ﾌﾚｰﾑ･ﾓｰﾄﾞでﾄﾘｶﾞ･ﾃﾞｰﾀ と!ﾃﾞｰﾀを表示
  Frame  Status  Address Data    7--EX--0
 !0013    BRM1    0107    55     00000000
 !0019    BRM1    0107    55     00000000
 !0025    BRM1    0107    55     00000000
 !0031    BRM1    0107    55     00000000
 !0037    BRM1    0107    55     00000000
 !0043    BRM1    0107    55     00000000
 !0049    BRM1    0107    55     00000000
 !0055    BRM1    0107    55     00000000
 !0061    BRM1    0107    55     00000000
 !0067    BRM1    0107    55     00000000
 !0073    BRM1    0107    55     00000000
  Total frame = 0962  !frame = 0135   (N/O/T/+/cr/-/Frame No./.) ? _
                                                    ↑
                                                    └ｷｰ入力待ち
```

・インストラクション・モードでの表示

```
*TRD I $Q<cr>              ←ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ でﾄﾘｶﾞ･ﾃﾞｰﾀ と!ﾃﾞｰﾀを表示
  Frame  Address  Label  Mnemonic
 !0013    0107          MOV [HL],A
 !0019    0107          MOV [HL],A
 !0019    0107          MOV [HL],A
 !0025    0107          MOV [HL],A
 !0031    0107          MOV [HL],A
 !0037    0107          MOV [HL],A
 !0043    0107          MOV [HL],A
 !0049    0107          MOV [HL],A
 !0055    0107          MOV [HL],A
 !0061    0107          MOV [HL],A
 !0067    0107          MOV [HL],A
 !0073    0107          MOV [HL],A
  Total frame = 0962  !frame = 0135   (N/O/T/+/cr/-/Frame No./.) ? _
                                                    ↑
                                                    └ｷｰ入力待ち
```

例3) $F が指定された場合

・フレーム・モードでの表示

```
*TRD F $F<cr>              ←ﾌﾚｰﾑ･ﾓｰﾄﾞでﾄﾘｶﾞ･ﾃﾞｰﾀ または!ﾃﾞｰﾀの前後5ﾗｲﾝ表示
  Frame  Status  Address Data    7--EX--0
  0008     M1      0108    C1     00000000
  0009     M1      0109    CB     00000000
  0010     M1      010A    80     00000000
  0011     OP      010B    FB     00000000
  0012    (M1)     010C    00     00000000
 !0013    BRM1     0107    55     00000000
  0014     M1      0108    C1     00000000
  0015     M1      0109    CB     00000000
  0016     M1      010A    80     00000000
  0017     OP      010B    FB     00000000
  0018    (M1)     010C    00     00000000
  Total frame = 0962  !frame = 0135   (N/O/T/+/cr/-/Frame No./.) ? _
                                                                  ↑
                                                                  └ｷｰ入力待ち
```

・インストラクション・モードでの表示

```
*TRD I $F<cr>              ←ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ でﾄﾘｶﾞ･ﾃﾞｰﾀ または!ﾃﾞｰﾀの前後5ﾗｲﾝ表示
  Frame  Address  Label  Mnemonic
  0004    0104          MOVW    HL,#1000H
  0007    0107          MOV     [HL],A
  0008    0108          INC     A
  0009    0109          DEC     B
  0010    010A          BNZ     $107H
 !0013    0107          MOV     [HL],A
  0014    0108          INC     A
  0015    0109          DEC     B
  0016    010A          BNZ     $107H
  0019    0107          MOV     [HL],A
  0020    0108          INC     A
  Total frame = 0962  !frame = 0135   (N/O/T/+/cr/-/Frame No./.) ? _
                                                                  ↑
                                                                  └ｷｰ入力待ち
```

例4) トリガ・データが存在しない場合

```
 .       .       .       .       .
 .       .       .       .       .
 .       .       .       .       .
 .       .       .       .       .
0010     M1     010A    80     00000000
Total frame = 0962   (N/O/T/+/cr/-/Frame No./.) ? T <cr>
Trigger frame not found
Total frame = 0962   (N/O/T/+/cr/-/Frame No./.) ? _
                                                  ↑
                                                  └キー入力待ち
```

例5) ！データが存在しない場合

```
*TRD F SQ <cr>
 Not found
*
```

保守／廃止

４－５－３７　トレース・データ検索条件設定

```
ＴＲＦ[_A=addrs][_V=mask][_C=ｽﾃｰﾀｽ][_E=mask]
```

A=addrs ：検索ｱﾄﾞﾚｽを設定します。検索ｱﾄﾞﾚｽは、5 つまでｽﾍﾟ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 ：検索ﾃﾞｰﾀ 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 ：検索ｱﾄﾞﾚｽと検索ﾃﾞｰﾀ の組み合わせ条件を下記から選択して設定します。省略した場合は条件として‘NC’が選択されます。

BRM1	:BRM1ﾌｪｯﾁ
M1	:M1ﾌｪｯﾁ
OP	:ｵﾍﾟｺｰﾄﾞ・ﾌｪｯﾁ
VECT	:ﾍﾞｸﾀ・ﾌｪｯﾁ
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・ﾌｪｯﾁを除くすべてのﾘｰﾄﾞ/ﾗｲﾄ

E=mask ：検索外部ﾃﾞｰﾀ 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。

‘ＴＲＦ’コマンドは、トレースされたデータを‘ＴＲＤ’コマンドでサーチする条件を設定するコマンドです。

‘ＴＲＦ’コマンドのオペランドを省略すると、検索する条件を対話形式で設定することができます。対話形式では現在設定されている検索条件も表示します。

例)

```
*TRF A=0FE8X 1F00,1FFF C=M1 <cr>      ←0FE80 ～ 0FE8F番地、1FF0～1FFF番地
                                        のM1ﾌｪｯﾁのﾄﾚｰｽ･ﾃﾞｰﾀを検索する設定です。

*TRF V=1 C=R E=1<cr>                  ←ﾃﾞｰﾀ '1'をﾘｰﾄﾞ して、外部ﾃﾞｰﾀ が1 の
                                        ﾄﾚｰｽ･ﾃﾞｰﾀを検する設定です。

*TRF<cr>                              ←対話形式でﾄﾚｰｽ･ﾃﾞｰﾀ検索条件を設定する場合。
 A 0,0FFFFH = 0XXXH <cr>              ←新しく0XXXH を設定します。
 V 1Y = <cr>                          ←現在のﾃﾞｰﾀ を変更しません。
   BRM1 fetch                              (BRM1)
   M1 fetch                                (M1)
   OPecode fetch                           (OP)
   VECTor fetch                            (VECT)
   Read                                    (R)
   Write                                   (W)
   Read Write by program                   (RWP)
   Read by program                         (RP)
   Write by program                        (WP)
   Read Write by Macro service             (RWP)
   Read by Macro service                   (RM)
   Write by Macro service                  (WM)
   No Condition                            (NC)
 C R = NC <cr>                        ←新しく 'NC' に設定します。
 E 1Y = 100H<cr>                      ←外部ﾃﾞｰﾀ に100Hを設定します。
 Input data error                     ←外部ﾃﾞｰﾀ は8ﾋﾞｯﾄなのでｴﾗｰ になります。
 E 1Y = 0XXH<cr>                      ←ｴﾗｰ になったので再度外部ﾃﾞｰﾀ を0XXHに設定します。
*
```

保守／廃止

４ － ５ － ３ ８　トレーサの再起動

TRG

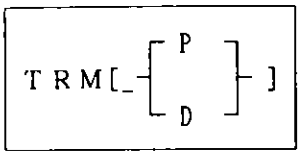
‘ＴＲＧ’コマンドは、アクティブ･モード時において、トレーサを再起動させるコマンドです。

本コマンドの入力により、アクティブ･モードから実行モード(CPU:ｴﾐｭﾚｰｼｮﾝ,ﾄﾚｰｻ:ﾄﾚｰｽ動作)へ移行します。

ＴＲＧコマンドは、エミュレーション中のコマンド入力待ち‘#’或は‘ｎ］’の時のみ有効なコマンドで通常のコマンド入力待ち‘＊’或は‘ｎ＞’の時と入力するとエラー･メッセージ　Unexcutable　Commando　が表示されます。

保守／廃止

４－５－３９　トレース・モード設定



P　：ﾌﾟﾛｸﾞﾗﾑ･ﾌｪｯﾁ

D　：ﾃﾞｰﾀ･ｱｸｾｽ

トレース・モード設定コマンドは、プログラム・フェッチかデータ・アクセスかの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）

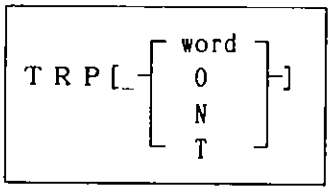
```
*TRM D <cr>                    ←ﾄﾚｰｽ条件にﾃﾞｰﾀ･ｱｸｾｽを指定
*

*TRM <cr>                      ←現在指定されているﾄﾚｰｽ条件を表示
 Data access
*
```

プログラム・フェッチが指定されている場合は、以下のようになります。

```
*TRM <cr>
 Program fetch
*
```

保守／廃止

４－５－４０　トレース・ポインタ操作



word：ﾎﾟｲﾝﾀの移動数

0　：ﾎﾟｲﾝﾀを先頭に置く

N　：ﾎﾟｲﾝﾀを最後に置く

T　：ﾎﾟｲﾝﾀをﾄﾘｶﾞ・ﾌﾚｰﾑに置く

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。オペランドに“Ｎ”が指定された場合は、トレース・ポインタを最も新しいトレース・ラインに移動します。また、オペランドに“Ｔ”が指定された場合は、トレース・ポインタをトリガ・フレームに移動します。

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。

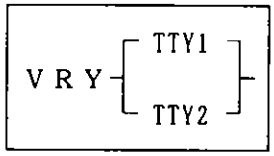
エミュレーション実行後のトレース・ポインタは、トリガ・フレームより５ポイント前のフレームを指示しています。ただし、トリガ・フレームが最も古いトレース・フレームより５行以内にある場合は、最も古いトレース・フレームを指示します。

例）

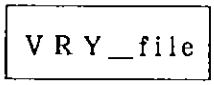
```
*TRP 20<cr>              ←ﾄﾚｰｽ･ﾎﾟｲﾝﾀ を20Hﾗｲﾝ進める
*

*TRP N <cr>              ←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
*

*TRP <cr>
 Total frame number     = 1200    ←ﾄﾚｰｽされたﾌﾚｰﾑの総数を表示
 Display frame pointer = 1012     ←表示したﾌﾚｰﾑの位置を表示
*
```

保守／廃止

４－５－４１　オブジェクト・ベリファイ



……　スタンド・アロン・モード



……　システム・モード

TTY1　　：ﾁｬﾈﾙ1（CH1）あるいはﾁｬﾈﾙ4（CH4）
TTY2　　：ｼﾘｱﾙ・ﾁｬﾈﾙ2
file　　：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。

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、あるいはセントロニクス・インターフェース 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、ＴＴＹ１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持つオブジェクト・ファイルとメモリ内容を比較します。

オブジェクト・ファイルとメモリ内容を比較した結果が異なる場合、メモリのアドレスとファイルの内容とメモリの内容を表示します。

チャネル１あるいはチャネル４の選択は、イニシャライズ時におけるダウンロード・モードの設定により行います。

例）スタンド・アロン・モードの場合

```
*VRY TTY2<cr>                ←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<cr>
 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
    └─────────────────  ﾒﾓﾘのｱﾄﾞﾚｽ
```

例）システム・モードの場合

```
1>VRY SAMPLE.HEX<cr>         ←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<cr>
  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
    └─────────────────  ﾒﾓﾘのｱﾄﾞﾚｽ


1>VRY SAMPLE.HEX<cr>
  file not found             ←SAMPLE.HEXというﾌｧｲﾙ名が見つからない場合の
1>                             ﾒｯｾｰｼﾞ
```

４－６　エラー・メッセージ

ＭＳ－ＤＯＳ版のシステム・ソフトウェアを使用している場合のみ、日本語で表示します。その他の場合は英語表示です。

（１）　aborted

失敗しました。

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。

（２）　Append symbol file not found

追加シンボル・ファイルがありません。

ＳＹＭ＿Ｌコマンドで、アペンド・シンボル・ファイルがカレント・ディスク上に存在しなかった。

（３）　append symbol table full

テーブルに空きがありません。

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。

（４）　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。

（５）　Bad character

不正なキャラクタを検出しました。

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。

（６）　Bad file entry

ファイル名の指定に間違いがあります。

ファイル名の記述が正しくない。

（７）　Can not close ﾌｧｲﾙ名

クローズできません。

表示されたファイルのクローズが正常にできなかった。

（８）　Can not close ﾌｧｲﾙ名.Cancel ×××　command

クローズできません。キャンセル×××コマンド！

×××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）

（９）　Can not execute HLP command !

ＨＬＰコマンドが使用できません！

カレント･ディスク上にヘルプ･ファイル，ヘルプ･オーバーレイ･ファイルが存在しない。

（１０）Can not open ﾌｧｲﾙ 名

オープンできません。

指定されたファイルがオープンできなかった。

（１１）Can not test

テストできるメモリがありません。

テストできるメモリがない。

（１２）Can not use command abbreviation !

省略形式でのコマンド入力はできません。

カレント・ディスク上に省略形のオーバーレイ・ファイルが存在しない。

（１３）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。

（１４）Check sum error

チェック・サム・エラー検出

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。

（１５）Command/Data too long

コマンド／データ入力文字数オーバ

１２８文字以上のコマンド、あるいはデータ行が入力された

（１６）Command format error

コマンド形式に間違いがあります。

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 。

（１７）Communication error

通信異常

ＩＥ－７８１３０－Ｒとホスト・マシンの通信が正常にできなかった 。

（１８）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。

（１９）Disk read error ﾌｧｲﾙ名

ディスク読みだしエラーを検出しました。

表示されたファイルの読み込みで異常を見つけた。

（２０）Disk read error ﾌｧｲﾙ名.Cancel STR command

ディスク読みだしエラーを検出しました。キャンセルＳＴＲコマンド

ＳＴＲコマンド実行中、表示されたファイルの読み込みで異常を見つけた。

（２１）Disk write error ﾌｧｲﾙ 名

ディスク書き込みエラーを検出しました。

表示されたファイルの書き込みで異常を見つけた。

（２２）Disk write error ﾌｧｲﾙ名.Cancel ××× command

ディスク書き込みエラーを検出しました。キャンセル×××コマンド

×××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）

（２３）double define append symbol

既に定義されたシンボルがあります。

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。

（２４）double define append symbol ｼﾝﾎﾞﾙ名

既に定義されたシンボル、シンボル名があります。

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。）

（２５）double define loaded symbol

既に定義されたシンボルをロードしました。

ＬＯＤ、ＳＹＭ＿Ａ、ＳＹＭ＿Ｌコマンドで、すでに登録されているシンボルがロードされた。

（２６）double define module name ﾓｼﾞｭｰﾙ 名

既に登録されているモジュールです。

表示されたモジュール名は、すでにロードされている。

（２７）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。

（２８）File already exists.

同じ名前のファイルが存在します。

ファイルの属性が SYSあるいは R/Oのファイルに対し、同一名のファイルを新たにメイクしようとした。

（２９）File make error ﾌｧｲﾙ名

ファイルが作成できません。

表示されたファイルを作成できなかった。

（３０）File name is used by other process

指定したファイル名は他のコマンドで使われています。

すでにオープン済みのファイル名を指定した。

（３１）file not found

ファイルがありません。

指定されたファイル名が存在しない。

（３２）File overflow

ロードできるファイル数をオーバしました。

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。

（３３）Illegal append symbol file

追加シンボル・ファイルの形式が違います。

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。

（３４）Illegal record

異常レコード

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。

（３５）Input data error

入力データに間違いがあります。

入力したデータが正しくない。

（３６）Keyword Error

キーワードに間違いがあります。

ＨＬＰコマンドで、コマンド･キーワードが正しくない。

（３７）List device is used by other process

プリンタが使用できません。

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。

（３８）load failed

ロード異常

ＬＯＤコマンドで、シンボル、あるいはオブジェクトのロード中にエラーを検出した。

（３９）Mapping error

マッピングされていない範囲があります。

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。

（４０）module buffer full

使用できるモジュール数を超過しました。

ＬＯＤコマンドで、入力できるモジュール数をオーバした。

（４１）Module not found

指定モジュールがありません。

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。

（４２）module overflow

登録できるモジュール数をオーバしました。

ＬＯＤコマンドで、入力できるモジュール数をオーバした。

（４３）Multi define

複数定義

ＰＧＭのカレント制御キャラクタ変更時に、同一キャラクタを設定した。

（４４）No appended symbol

アペンド・シンボルがありません。

ＳＹＭ＿Ｓコマンドで、アペンド・シンボルは存在しない。

（４５）No .HLP file on the default drive

ヘルプ・ファイルがありません。

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，ヘルプ･オーバーレイ･ファイルが見つからなかった。

（４６）No symbol

シンボルがありません。

シンボルがない。

（４７）Non map area access

マッピングされていないエリアをアクセスしました。

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。

（４８）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。

（４９）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。

（５０）not found module record

モジュール名レコードがありません。

ＬＯＤコマンドで指定されたモジュール名レコードがシンボル・ファイル内に存在しない。

（５１）Reserved file name

使用できないファイル名です。

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。

（５２）reserved word symbol

予約語です。

ＳＹＭ＿Ａコマンドで、予約語がシンボルとして定義された。

（５３）Slave CPU communication error

チャネル２のスレーブＣＰＵ(8742)に対し、コマンドが書き込めない。

（５４）Symbol not found

指定したシンボルがありません。

ＳＹＭ＿Ｃ、ＳＹＭ＿Ｅコマンドで、指定されたシンボルは存在しない。

（５５）Symbol record format error

シンボルレコード形式が間違っています。

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。

（５６）symbol table full

シンボル・テーブルに空きがありません。

ＬＯＤコマンドで、シンボル･セーブ･エリアに空がない。

（５７）System mode command

システム・モード専用コマンドです。

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。

（５８）Unexecutable command

このモードでは、実行できません。

エミュレーション中に実行できないコマンドを入力した、あるいはエミュレーション中にしか実行できないコマンドをブレーク中に入力した。

（５９）Unrecognized command

存在しないコマンドです。

入力したコマンドは存在しない。

（６０）Warning!

オブジェクトの生成はできるが、正しい動作は望めない。

（６１）Warning double define : ﾓｼﾞｭｰﾙ 名

複数指定

ＬＯＤコマンドで、同一モジュール名が複数回指定された。

保守／廃止

４－７　オンライン・アセンブラ／逆アセンブラ仕様

ＩＥ－７８１３０－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うコマンドに適用されます。

４－７－１　μＰＤ７８１３Ｘ　インストラクション一覧

４－７－２　ＳＦＲマッピング

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で明確とされていない仕様や、４－７－１、４－７－２と若干異なる仕様につきましては、次の項目で規定しています。

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

４－７－１　μＰＤ７８１３Ｘ　インストラクション一覧表

μＰＤ７８１３Ｘ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 Devide Instructions

Bit Manipulation Instructions

Branch Instructions

CPU Control Instructions

Decimal Adjust Instruction

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に、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   r1,r1          move G-reg to G-reg
  MOV   A,r1           move G-reg to ACC
  MOV   A,saddr        move SDMEM to ACC
  MOV   saddr,A        move ACC to SDMEM

  MOV   A,sfr          move SFR to ACC
  MOV   A,[D]          move Memory to ACC
  MOV   A,[E]          move Memory to ACC
  MOV   A,[E+]         move Memory to ACC after increment
  MOV   sfr,A          move ACC to SFR
  MOV   [D],A          move ACC to Memory
  MOV   [E],A          move ACC to Memory
  MOV   [E+],A         move ACC to Memory after increment

  MOV   A,[HL]         move Memory to ACC
  MOV   A,word[A]      move Memory to ACC
  MOV   A,word[B]      move Memory to ACC
  MOV   [HL],A         move ACC to Memory
  MOV   word[A],A      move ACC to Memory
  MOV   word[B],A      move ACC to Memory

  XCH   A,r1           exchange ACC and G-reg
  XCH   A,[E]          exchange ACC and Memory
  XCH   A,[D]          exchange ACC and Memory
  XCH   A,saddr        exchange ACC and SDMEM
* XCH   A,sfr          exchange ACC and SFR

  PUSH  PSW            push PSW on System Stack
  POP   PSW            pop PSW off System Stack
```

----- 16-bit Transfer Instructions -----

```
  MOVW  rp1,#word      move Immediate to G-reg Pair
  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  AX,sfrp        move SFR Pair to Extended ACC
  MOVW  sfrp,AX        move Extended ACC to SFR Pair
  PUSH  rp1            push G-reg Pair on System Stack
  POP   rp1            pop G-reg Pair 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	r1,r1	add G-reg to G-reg
	ADDC	r1,r1	add G-reg to G-reg with Carry
	SUB	r1,r1	subtract G-reg from G-reg
	SUBC	r1,r1	subtract G-reg from G-reg with Borrow
	AND	r1,r1	and G-reg with G-reg
	OR	r1,r1	or G-reg with G-reg
	XOR	r1,r1	exclusive-or G-reg with G-reg
	CMP	r1,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	A, [D]	add Memory to ACC
ADDC	A, [D]	add Memory to ACC with Carry
SUB	A, [D]	subtract Memory from ACC
SUBC	A, [D]	subtract Memory from ACC with Borrow
AND	A, [D]	and Memory with ACC
OR	A, [D]	or Memory with ACC
XOR	A, [D]	exclusive-or Memory with ACC
CMP	A, [D]	compare Memory with ACC
ADD	A, [E]	add Memory to ACC
ADDC	A, [E]	add Memory to ACC with Carry
SUB	A, [E]	subtract Memory from ACC
SUBC	A, [E]	subtract Memory from ACC with Borrow
AND	A, [E]	and Memory with ACC
OR	A, [E]	or Memory with ACC
XOR	A, [E]	exclusive-or Memory with ACC
CMP	A, [E]	compare Memory with ACC
ADD	A, [HL]	add Memory to ACC
ADDC	A, [HL]	add Memory to ACC with Carry
SUB	A, [HL]	subtract Memory from ACC
SUBC	A, [HL]	subtract Memory from ACC with Borrow
AND	A, [HL]	and Memory with ACC
OR	A, [HL]	or Memory with ACC
XOR	A, [HL]	exclusive-or Memory with ACC
CMP	A, [HL]	compare Memory with ACC

INC	rl	increment G-reg
DEC	rl	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   AX,rp1            add G-reg Pair to Extended ACC
SUBW   AX,rp1            subtract G-reg Pair from Extended ACC
CMPW   AX,rp1            compare G-reg Pair with Extended ACC

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   AX,sfrp           add SFR Pair to Extended ACC
SUBW   AX,sfrp           subtract SFR Pair from Extended ACC
CMPW   AX,sfrp           compare SFR Pair with Extended ACC

INCW   rp1               increment G-reg Pair
DECW   rp1               decrement G-reg Pair
```

----- Multiply and Devide Instructions -----

```
MULUW r1                 multiply Extended ACC by G-reg
DIVUW r1                 divide Extended ACC by G-reg
```

----- 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,PSW .bit	move PSW bit to Carry
MOV1	PSW .bit,CY	move Carry to PSW bit
AND1	CY,PSW .bit	and PSW bit with Carry
OR1	CY,PSW .bit	or PSW bit with Carry
AND1	CY,/PSW .bit	and complement PSW bit with Carry
OR1	CY,/PSW .bit	or complement PSW bit with Carry
XOR1	CY,PSW .bit	exclusive-or PSW bit with Carry
SET1	PSW .bit	set PSW bit
CLR1	PSW .bit	clear PSW bit
NOT1	PSW .bit	complement PSW bit
CLR1	CY	clear Carry
NOT1	CY	complement Carry
SET1	CY	set Carry


```
----- Branch Instructions -----

CALL  !addr16           call subroutine 16 bit immediate or label
CALLF !addr11           call subroutine 11 bit immediate or label
CALLT [addr5]           call subroutine 5 bit immediate or label

RET                     return from subroutine
RETI                    return from interrupt

BR    !addr16           branch absolute
BR    rp1               branch register
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

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

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

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

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

BT    PSW.bit,$addr16   branch if PSW bit True
BF    PSW.bit,$addr16   branch if PSW bit False
BTCLR PSW.bit,$addr16   branch and clear if PSW bit True

DBNZ  r2,$addr16        decrement G-reg & branch relative if not zero
DBNZ  saddr,$addr16     decrement SDMEM & branch relative if not zero
```

----- CPU Control Instruction -----

NOP		no operation
EI		enable interrupt
DI		disable interrupt
SEL	RBn	select Register Bank & Main Register Set
MOV	STBC,#byte	move Immediate to STBC
MOV	PSW,#byte	move Immediate to PSW
MOV	PSW,A	move ACC to PSW
MOV	A,PSW	move PSW to ACC
MOVW	SP,#word	move Immediate to SP
MOVW	SP,AX	move Extended Acc to SP
MOVW	AX,SP	move SP to Extended Acc

----- Decimal Adjust Instruction -----

ADJBA	adjust decimal ACC after Addition
ADJBS	adjust decimal ACC after Subtract

----- 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 right 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	[E]	rotate right digit
ROR4	[D]	rotate right digit
ROL4	[E]	rotate left digit
ROL4	[D]	rotate left digit

***** NOTE *****

SDMEM: short direct Memory

r1: X,A,C,B,E,D,L,H,R0,R1,R2,R3,R4,R5,R6,R7
r2: C,B,R3,R2

rp1: AX,BC,DE,HL,RP0,RP1,RP2,RP3

4－7－2　SFRマッピング

8bit SFR MAPPING

FF00	P0	FF20	PM0	FF40	PU0	FF60	
FF01	P1	FF21	PM1	FF41		FF61	
FF02	P2	FF22		FF42		FF62	
FF03	P3	FF23	PM3	FF43	PMC3	FF63	
FF04	P4	FF24		FF44		FF64	
FF05	P5	FF25	PM5	FF45		FF65	
FF06	P6	FF26	PM6	FF46		FF66	
FF07	P7	FF27	PM7	FF47		FF67	
FF08		FF28		FF48		FF68	ADM
FF09		FF29		FF49		FF69	
FF0A		FF2A		FF4A	POL	FF6A	ADCR
FF0B		FF2B		FF4B	POH	FF6B	
FF0C		FF2C		FF4C	RTPC	FF6C	
FF0D		FF2D		FF4D		FF6D	
FF0E		FF2E		FF4E		FF6E	
FF0F		FF2F		FF4F		FF6F	
FF10		FF30		FF50	ICR	FF70	PWMC
FF11		FF31		FF51		FF71	
FF12		FF32		FF52		FF72	
FF13		FF33		FF53	EDVC	FF73	
FF14		FF34		FF54	ECC1	FF74	
FF15		FF35		FF55	ECC0	FF75	
FF16		FF36		FF56	EC	FF76	
FF17		FF37		FF57		FF77	
FF18		FF38	TMC0	FF58	TOM0	FF78	
FF19		FF39	TMC1	FF59	TOC0	FF79	
FF1A		FF3A	CPTM	FF5A	TOM1	FF7A	
FF1B		FF3B		FF5B	TOC1	FF7B	
FF1C	CPT2L	FF3C		FF5C		FF7C	
FF1D	PRM3	FF3D	TM3	FF5D		FF7D	
FF1E		FF3E	CR30	FF5E		FF7E	
FF1F		FF3F	CPT30	FF5F		FF7F	CLOM

FF80	CSIM	FFA0		FFC0	STBC	FFE0	IF0L
FF81		FFA1		FFC1		FFE1	IF0H
FF82	SBIC	FFA2		FFC2		FFE2	
FF83		FFA3		FFC3		FFE3	
FF84		FFA4		FFC4	MM	FFE4	MK0L
FF85		FFA5		FFC5		FFE5	MK0H
FF86	SIO	FFA6		FFC6		FFE6	
FF87		FFA7		FFC7		FFE7	
FF88		FFA8		FFC8		FFE8	PR0L
FF89		FFA9		FFC9		FFE9	PR0H
FF8A		FFAA		FFCA		FFEA	
FF8B		FFAB		FFCB		FFEB	
FF8C		FFAC		FFCC		FFEC	ISM0L
FF8D		FFAD		FFCD		FFED	ISM0H
FF8E		FFAE		FFCE		FFEE	
FF8F		FFAF		FFCF		FFEF	
FF90		FFB0		FFD0	EXTSFR0	FFF0	
FF91		FFB1		FFD1	EXTSFR1	FFF1	
FF92		FFB2		FFD2	EXTSFR2	FFF2	
FF93		FFB3		FFD3	EXTSFR3	FFF3	
FF94		FFB4		FFD4	EXTSFR4	FFF4	INTM0
FF95		FFB5		FFD5	EXTSFR5	FFF5	INTM1
FF96		FFB6		FFD6	EXTSFR6	FFF6	
FF97		FFB7		FFD7	EXTSFR7	FFF7	
FF98		FFB8		FFD8	EXTSFR8	FFF8	IST
FF99		FFB9		FFD9	EXTSFR9	FFF9	
FF9A		FFBA		FFDA	EXTSFR10	FFFA	
FF9B		FFBB		FFDB	EXTSFR11	FFFB	
FF9C		FFBC		FFDC	EXTSFR12	FFFC	
FF9D		FFBD		FFDD	EXTSFR13	FFFD	
FF9E		FFBE		FFDE	EXTSFR14	FFFE	
FF9F		FFBF		FFDF	EXTSFR15	FFFF	

16bit 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	CR02	FF2C		FF4C		FF6C	
FF0D		FF2D		FF4D		FF6D	
FF0E	CR10	FF2E		FF4E		FF6E	
FF0F		FF2F		FF4F		FF6F	
FF10	CR11	FF30	TM0	FF50		FF70	
FF11		FF31		FF51		FF71	
FF12	CR12	FF32	TM1	FF52		FF72	PWM0
FF13		FF33		FF53		FF73	
FF14	CPT0	FF34	FRC	FF54		FF74	PWM1
FF15		FF35		FF55		FF75	
FF16	CPT1	FF36	TM2	FF56		FF76	
FF17		FF37		FF57		FF77	
FF18	CPT2H	FF38		FF58		FF78	
FF19		FF39		FF59		FF79	
FF1A	CPT3	FF3A		FF5A		FF7A	
FF1B		FF3B		FF5B		FF7B	
FF1C		FF3C		FF5C		FF7C	
FF1D		FF3D		FF5D		FF7D	
FF1E	CR20	FF3E		FF5E		FF7E	
FF1F		FF3F		FF5F		FF7F	

FF80	FFA0	FFC0	FFE0	IF0
FF81	FFA1	FFC1	FFE1	
FF82	FFA2	FFC2	FFE2	
FF83	FFA3	FFC3	FFE3	
FF84	FFA4	FFC4	FFE4	MK0
FF85	FFA5	FFC5	FFE5	
FF86	FFA6	FFC6	FFE6	
FF87	FFA7	FFC7	FFE7	
FF88	FFA8	FFC8	FFE8	PR0
FF89	FFA9	FFC9	FFE9	
FF8A	FFAA	FFCA	FFEA	
FF8B	FFAB	FFCB	FFEB	
FF8C	FFAC	FFCC	FFEC	ISM0
FF8D	FFAD	FFCD	FFED	
FF8E	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

本アセンブラで使用できる、すべての文字を次に示します。

Ａ～Ｚ　ａ～ｚ　＠　？　＿（アンダー・スコア）　０～９

＋　－　＊　／　＄　！　＆　［　］　＃　（　）　；

．（ピリオド）　，（コンマ）　＼（バック・スラッシュ）

ただし、＼（バック・スラッシュ）は、シンボル表現としてのみ使用可能であり、インストラクションには使用できません。

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　┐
                                    ├── ＯＫ
ＯＲＧ　　ｓｙｍｂｏｌ　　　　　　┘

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　　　　　　┐
                                ├── ＯＫ
ＮＯＰ　　；　ｃｏｍｍｅｎｔ　┘
```

（４）　オペランドに用いる数値表現

コマンド仕様に於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（＋、－）を付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）

・サフィックスがＨのとき

０ＦＦＦＦ　┐
－０ＦＦＦＦ　┘ ― ＯＫ

ＦＦＦＦ ― FFFFというｼﾝﾎﾞﾙとみなされます

－ＦＦＦＦ ― FFFFというｼﾝﾎﾞﾙに符号がついたものとみなされます

１ＦＦＦＦ　┐
－１ＦＦＦＦ　├ エラー（数値がＦＦＦＦをこえています。）
０Ｆ０ＦＦＦ　┘

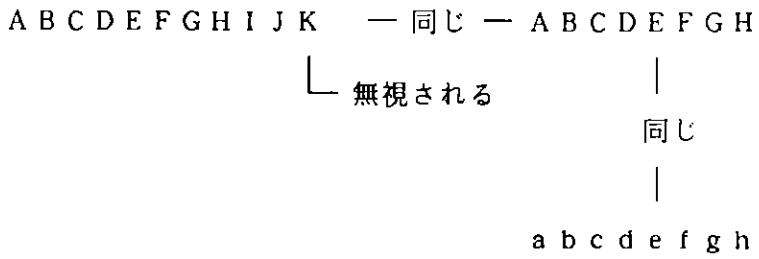
（５）　オペランドに用いるシンボル表現

数値のかわりに、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・スコア）、０～９のいずれかの文字から構成されます。ただし、数字から始まってはいけません。また、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みなされます。

英小文字（ａ～ｚ）は英大文字（Ａ～Ｚ）と同じ意味に扱われます。

例）



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。ただし、ＳＹＭ＿Ｍコマンドによりカレント・モジュールに指定されている場合、モジュール名の記述を省略することができます。

例えば、”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

ただし、ＳＹＭ＿Ｍコマンドで ＭＯＤＵＬＥ０１がカレント・モジュールに指定されている場合は次のように記述します。

ＭＯＤＵＬＥ０１＼ＳＹＭＢＯＬ０１

あるいは

ＳＹＭＢＯＬ０１

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）

```
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

数値のか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持ちません。（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うりです。

高 ←－－－－ 優先順位 －－－－→ 低
（　）　＊　／　＋　－　ＡＮＤ　ＯＲ　ＸＯＲ

演算はすべて１６ビットの正の整数で行い、小数点以下は切り捨てられます。また、演算の中間結果、最終結果でオーバー・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は、１７ビット目より上位を切り捨てられます。

負になった場合は、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(ｾﾞﾛ･ﾃﾞｨﾊﾞｲﾄﾞ)

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、および、予約語に対して演算を行いますとエラーになります。

なお、数値、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。しかし、数値、式、あるいはシンボルと AND、OR、XORの各演算子との間には、１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）　擬似命令

本アセンブラでは、以下の擬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＞ＦＥＤＦＨの場合、‘Assemble area over!’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ＦＥＤＦＨをこえた場合は、‘Assemble area over!’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記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ＦＥＤＦＨをこえた場合は、‘Assemble area over!’が表示されます。また、ＦＥ２１Ｈ～ＦＥＤＤＨ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、ＦＥＥ０Ｈ～ＦＦＦＦＨとなった場合、‘Assemble area over!’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、オーバー・フローした上位桁は切り捨てられる。この時、‘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）　同じニー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

μＰＤ７８１３Ｘでは、同じニーモニックを持っていても、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。

ｓａｄｄｒ，ＳＦＲにまたがるもの

		命令長	実行サイクル
ＭＯＶ	A, saddr	2	2
	A, ＳＦＲ	2	4
ＭＯＶ	saddr, A	2	3
	ＳＦＲ, A	2	5
ＭＯＶ	saddr, #byte	3	3
	ＳＦＲ, #byte	3	5
ＭＯＶＷ	saddrp, #word	4	4
	ＳＦＲＰ, #word	4	8
ＭＯＶ１	ＣＹ, saddr. bit	3	5
	ＣＹ, ＳＦＲ. bit	3	9
ＭＯＶ１	saddr. bit , ＣＹ	3	8
	ＳＦＲ. bit , ＣＹ	3	１２
ＡＮＤ１	ＣＹ, (／) saddr. bit	3	5
	ＣＹ, (／) ＳＦＲ. bit	3	9
ＯＲ１	ＣＹ, (／) saddr. bit	3	5
	ＣＹ, (／) ＳＦＲ. bit	3	9
ＸＯＲ１	ＣＹ, saddr. bit	3	5
	ＣＹ, ＳＦＲ. bit	3	9

SET1	saddr. bit	2	3
	SFR. bit	3	10
CLR1	saddr. bit	2	3
	SFR. bit	3	10

		命令長	実行サイクル
NOT1	saddr.bit	3	6
	SFR.bit	3	10
BT	saddr.bit,$addr16	3	4（6）
	SFR.bit,$addr16	4	6（8）
BF	saddr.bit,$addr16	4	5（7）
	SFR.bit,$addr16	4	7（9）
BTCLR	saddr.bit,$addr16	4	5（9）
	SFR.bit,$addr16	4	7（13）

実行ｻｲｸﾙの左側の数は、分岐しないで次の命令を実行するときのｻｲｸﾙ数です。ｶｯｺ 内の数は、分岐するときのｻｲｸﾙ数です。

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、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・チェックを行います。

（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

① ＳＦＲ予約語は、ＳＦＲかＳＦＲＰかの属性を持ちます。

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予約語は、読出し専用(R/O)、書込み専用(W/O)、読み書き可能(R/W)の属性を持ちます。

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、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

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 １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かの属性を持たせることはできません。したがって、ＳＦＲと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属性のチェックを行います。

（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／奇数のチェックを行います。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）

例）

		省　略		
ＢＣ	$addr16	――――	ＢＣ	addr16
ＣＡＬＬ	!addr16	――――	ＣＡＬＬ	addr16
ＣＡＬＬＦ	!addr16	――――	ＣＡＬＬＦ	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。）

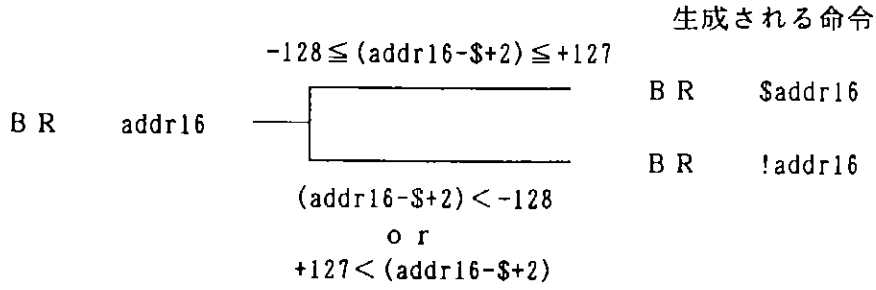
例）

addressing mode 指定の＄
｜ﾛｹｰｼｮﾝ･ｶｳﾝﾀ の＄
｜　｜

ＢＣ	＄＄±ｎ	――――	ＯＫ	（＄±ｎ番地へのﾌﾞﾗﾝﾁ）
ＢＣ	＄＋ｎ	――――	ＯＫ	（ｎ番地へのﾌﾞﾗﾝﾁ） （＄＋ｎ番地へのﾌﾞﾗﾝﾁではない！）
ＢＣ	＄－ｎ	――――	ＯＫ	（１００００Ｈ－ｎ番地へのﾌﾞﾗﾝﾁ）

（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する。この時、‘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）　機能表現と絶対表現との対応

ｒ１、ｒ２、ｒｐ、ｒｐ１を用いる命令系に対し、機能表現（Ｘ，Ａ，Ｃ，Ｂ，‥‥，ＡＸ，ＢＣ，‥‥）と絶対表現（Ｒ０，Ｒ１，Ｒ２，Ｒ３，‥‥，ＲＰ０，ＲＰ１，‥‥）の両方を記述することができます。

対応のとり方は次の通りです。

機能表現		絶対表現
Ｘ	―――	Ｒ０
Ａ	―――	Ｒ１
Ｃ	―――	Ｒ２
Ｂ	―――	Ｒ３
Ｅ	―――	Ｒ４
Ｄ	―――	Ｒ５
Ｌ	―――	Ｒ６
Ｈ	―――	Ｒ７
ＡＸ	―――	ＲＰ０
ＢＣ	―――	ＲＰ１
ＤＥ	―――	ＲＰ２
ＨＬ	―――	ＲＰ３

例）

```
MOV  r1, r1

   MOV  R2, H

   MOV  C, R7

DBNZ  r2, $addr16

   DBNZ  B, $$-5H

   DBNZ  R3, $$-5H

INCW  rp

   INCW  RP2

   INCW  DE
```

（１４）　ＰＵＳＨ，ＰＯＰのオペランド記述

ＰＵＳＨ，ＰＯＰ命令のオペランドには、ＲＰ０～ＲＰ３をコンマ（，）で区切って４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２つ以上記述された場合は‘Error!’が表示されます。

例）

ＰＵＳＨ	ＲＰ３，ＲＰ２，ＲＰ１，ＲＰ０	→ ＯＫ
ＰＵＳＨ	ＲＰ３，ＲＰ１，ＲＰ２，ＲＰ３	→ Error!
ＰＵＳＨ	ＡＸ，ＤＥ，ＨＬ，ＲＰ０	→ Error! (AXとRP0 は同一ﾚｼﾞｽﾀ)

（１５）　エラー表示

本アセンブラは、エラー・コードとして次の３種類を表示します。

Ｅｒｒｏｒ

Ｗａｒｎｉｎｇ

Ｃａｕｔｉｏｎ

① Ｅｒｒｏｒ　：　オブジェクト・コードを生成できないか、または、あきらかにエラーである場合に表示されます。

例）

ＯＲＧ	０ＦＦ００Ｈ	(ﾌﾟﾛｸﾞﾗﾑ･ｴﾘｱ ではない)
ＤＢ	０ＦＦ００Ｈ	(ﾜｰﾄﾞ･ﾃﾞｰﾀ である)
ＭＯＶ	Ｊ，＃byte	(Ｊという予約語はない)
ＢＲ	$	(ｼﾞｬﾝﾌﾟ･ｱﾄﾞﾚｽ 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）

ＭＯＶ	ＳＦＲＰ，＃byte	(ＳＦＲＰに対する 8ﾋﾞｯﾄ 操作)
ＭＯＶＷ	ＳＦＲ，＃word	(ＳＦＲに対する16ﾋﾞｯﾄ 操作)
ＭＯＶＷ	odd saddr,#word	(奇数 saddrに対する16ﾋﾞｯﾄ 操作)
ＢＲ	０ＦＦ００Ｈ	(ＳＦＲｴﾘｱ へのﾌﾞﾗﾝﾁ)

③ Ｃａｕｔｉｏｎ　：　ジェネリックなオブジェクト生成が行われた場合、例えば、ジェネリック・ブランチ、あるいは注意を要する場合。

例）

ＯＲＧ	$－１００	(注意を要する)
ＢＲ	１００Ｈ	(ｼﾞｪﾈﾘｯｸ･ﾌﾞﾗﾝﾁ)

保守／廃止

（１６）　予約語一覧表

A	AC	ADCR
ADD	ADDC	ADDW
ADJBA	ADJBS	ADM
AND	AND1	AX
B	BC	BE
BF	BL	BNC
BNE	BNL	BNZ
BR	BT	BTCLR
BZ	C	CALL
CALLF	CALLT	CLOM
CLR1	CMP	CMPW
CPT0	CPT1	CPT2H
CPT2L	CPT3	CPT30
CPTM	CR00	CR01
CR02	CR10	CR11
CR12	CR20	CR30
CSIM	CY	D
DB	DBNZ	DE
DEC	DECW	DI
DIVUW	DS	DW
E	EC	ECC0
ECC1	EDVC	EI
END	EXTSFR0	EXTSFR1
EXTSFR2	EXTSFR3	EXTSFR4
EXTSFR5	EXTSFR6	EXTSFR7
EXTSFR8	EXTSFR9	EXTSFR10
EXTSFR11	EXTSFR12	EXTSFR13
EXTSFR14	EXTSFR15	FRC
H	HL	ICR
IE	IF0	IF0H
IF0L	INC	INCW
INTM0	INTM1	ISP
IST	L	MK0
MK0H	MK0L	MM
MOV	MOV1	MOVW
MULUW	NOP	NOT1
OR	OR1	ORG
P0	P0H	P0L
P1	P2	P3
P4	P5	P6
P7	PM0	PM1
PM3	PM5	PM6
PM7	PMC3	POP
PR0	PR0H	PR0L
PRM3	PSW	PU0
PUSH	PWM0	PWM1
PWMC	R0	R1
R2	R3	R4
R5	R6	R7
RB0	RB1	RB2

RB3	RBS0	RBS1
RET	RETI	ROL
ROL4	ROLC	ROR
ROR4	RORC	RP0
RP1	RP2	RP3
RTPC	SBIC	SEL
SET1	SHL	SHLW
SHR	SHRW	SIO
SP	STBC	SUB
SUBC	SUBW	TM0
TM1	TM2	TM3
TMC0	TMC1	TOC0
TOC1	TOM0	TOM1
X	XCH	XOR
XOR1	Z	

保守／廃止

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命令が、最後にＥＮＤ命令が付加されます。

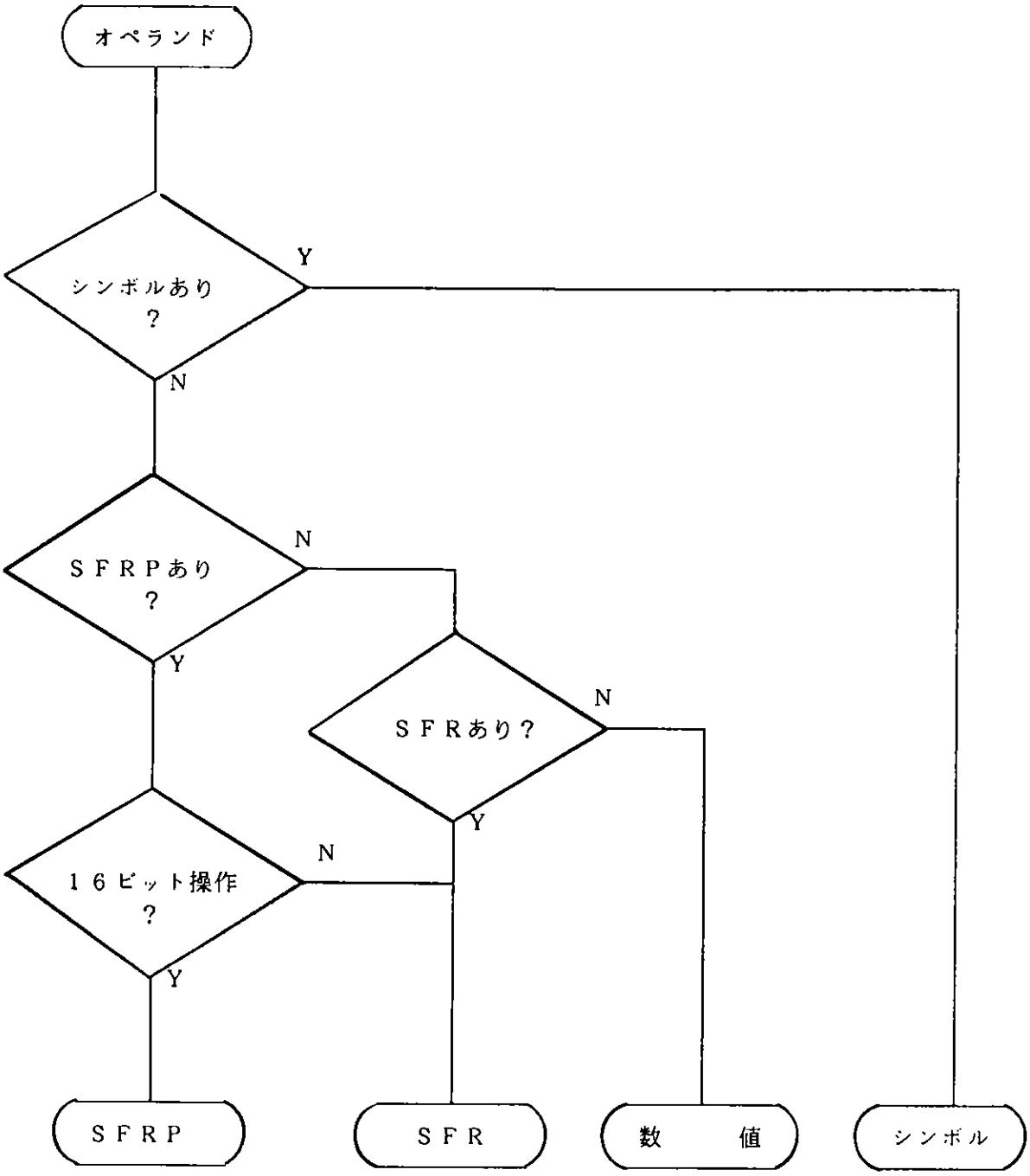
（２）　オペランドの数値表示

オペランドの数値に対応するシンボル、ＳＦＲ、あるいはＳＦＲＰがある場合、数値をシンボル、ＳＦＲ、あるいはＳＦＲＰに置き換えて表示します。

数値に対して、シンボル、ＳＦＲ、ＳＦＲＰが対応する場合、シンボルが優先して表示されます。

（３）　レジスタ表現

‘ＲＧＭ’コマンドで設定されたレジスタ表現を用います。



数値に対応するシンボル，ＳＦＲ，ＳＦＲＰがない場合、１６進数で数値を表示しサフィックスＨを付加します。

数値の先頭文字は必らず数字（０～９）で始まります。

（４）　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。

（５）　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、以下のように表示します。

```
                          ┌──  コロン１つ
module＼local symbol :
```

パブリック・シンボルの場合、以下のように表示します。

```
                          ┌┬──  コロン２つ
module＼public symbol  : :
```

（６）　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種類のニーモニックが割り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ーモニックが表示されます。

```
BNZ  ┐                    BZ  ┐
     ├─ BNZ                   ├─ BZ
BNE  ┘                    BE  ┘

BNC  ┐                    BC  ┐
     ├─ BNC                   ├─ BC
BNL  ┘                    BL  ┘
```

（７）　アドレス・チェック

ＳＦＲ空間、および、ｓａｄｄｒ空間に対しアドレス・チェックを行い、以下のような場合に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がない場合

・Ｒ／ＯのＳＦＲに書込み動作を行う場合

・Ｗ／ＯのＳＦＲに読出し動作を行う場合

・ＳＦＲＰ以外の空間に対し１６ビット操作を行う場合

②　ｓａｄｄｒ空間（ＦＥ２０Ｈ～ＦＥＦＦＨ）

・奇数アドレスに対し１６ビット操作を行った場合

（８）　ＭＯＶ　ＳＴＢＣ，＃byte命令（専用命令）

ＳＴＢＣに対して書込み動作を行うような、専用命令以外のＳＦＲ操作命令、および、ＰＳＦＲ＋ｓａｄｄｒ操作命令の場合、警告が表示されます。

また、専用命令に於いて、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、ニーモニックのオペランドには、ｉｍｍｅｄｉａｔｅデータを用います。

（９）　エラー表示

逆アセンブラはエラー・コードとして、エラー／警告を表示します。

①　エラー：逆アセンブルできない場合、ニーモニックの所に？？？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持つ命令コードで、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行います。

例）

```
ADDR    OBJECT          MNEMONIC
                    PUBLIC ＼ START::
0100    0C 40 34 12     MOVW  0FE40H,#1234H
0104    01              ???
0105    60 00 10        MOVW  AX,#1000H
0109    5F              ???
010A    7A              MOV   [D],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ーモニックの直前に？を表示します。

例）

```
ADDR    OBJECT          MNEMONIC
0100    0C 41 34 12   ?  MOVW  0FE41H,#1234H
                          (奇数のsaddr に16ﾋﾞｯﾄ 操作を行います。)
0104    3A 02 20      ?  MOV   P2,#20H
                          (R/O の SFRに書込み動作を行います。)
0107    2C 00 FF      ?  BR    !0FF00H
                          (SFRｴﾘｱへﾌﾞﾗﾝﾁします。)
```

保守／廃止

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。

各コマンドについては、「第４章　コマンド仕様」を読んでください。

５－２　コマンド入力時のテクニック

５－２－１　ファイルからのコマンド入力（ＳＴＲ　コマンド）

５－２－２　コマンド・ファイル作成機能（ＣＯＭ　コマンド）

５－２－３　行単位の編集機能

５－２－４　コマンド・ヒストリ機能（ＨＩＳ　コマンド）

５－２－５　コマンド省略形入力

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

５－４　その他

５－４－１　コマンド・ヘルプ機能（ＨＬＰ　コマンド）

５－４－２　ファイルへの結果出力（ＬＳＴ　コマンド）

５－４－３　ディレクトリ表示機能（ＤＩＲ　コマンド）

保守／廃止

５ － ２　コマンド入力時のテクニック

ＩＥ－７８１３０－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。

例えば、

１、スタート・アップ時の環境設定（クロック設定、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、ディバグ中に良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－７８１３０－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．

ａ：ファイルからのコマンド入力（ＳＴＲ　コマンド）

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み出し、実行します。

ｂ：コマンド・ファイル作成機能（ＣＯＭ　コマンド）

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。

２．行単位の編集機能

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場合でも、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。

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び出すことができます。呼び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．コマンドの省略形入力

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良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）

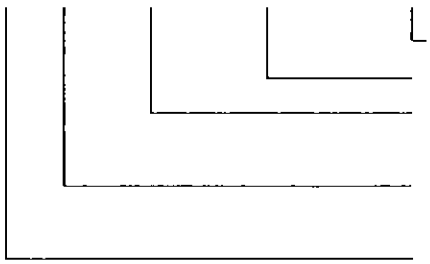
‘ＳＴＲ’コマンドを実行することにより、以降のコマンド入力を‘ＳＴＲ’コマンドで指定したファイルより、自動的に行うことができます。

‘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き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。

```
n>STR A: SAMPLE .STR SAMPLE.HEX SAMPLE.TXT
```



仮ﾊﾟﾗﾒｰﾀ($1) に対応する実ﾊﾟﾗﾒｰﾀ です。
仮ﾊﾟﾗﾒｰﾀ($0) に対応する実ﾊﾟﾗﾒｰﾀ 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、8 文字まで指定できます。省略することはできません。
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。

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 は置き換えられません。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 2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
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(1BH)
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 <cr>
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>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<cr>   (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に置き換えられます。)
  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に置き換えられます。)
  User-system Vcc-ON    Emulation start at 0100
  Standard break  terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  07    82    00    00    2000    2007   0106  8000
  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に置き換えられます。)
  2000   00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 00     ................
n>BRA A=200 <cr>                 (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<cr>       (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
                                  せんので置き換えられません。)
  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
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   1   0    1   0  07    00    01    20    2108    2107   0201  8000
  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に置き換えられます。)
  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り返し使用する

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<cr>
  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
n>                       表示されます。)
```

保守／廃止

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コマンドに於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：　……　オープンされている出力装置をクローズします。

※　オープンされている出力装置以外の装置を新たに指定しますと、オープンされている出力装置はクローズされます。

実行例を次頁に出力装置ごとに示します。

・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 <cr>
  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
n>             場合に表示されます。)

n>COM <cr>
  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 <cr>
  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>
```

・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
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 A: SAMPLE .STR<cr>      (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： 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： 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：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<cr>              (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 <cr>      (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):<cr>   (すでに存在するﾌｧｲﾙを削除するか
                          ｶｰｿﾙ位置 ┘          どうかの ﾒｯｾｰｼﾞが表示されます。)


  File already exists.Delete ? (Y or N): Y<cr>
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<cr>
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。

```
n>COM SAMPLE.SYS<cr>          (指定したﾌｧｲﾙが R/O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
n>                             されます。ｺﾏﾝﾄﾞは無効になります。)
```

・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:<cr>      (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.STR 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'ｷｰが入力された以降にｺﾝｿｰﾙ 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:<cr>
  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>
```

・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:<cr>      (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、出力装置には出力されません。)
n>
```

５ — ２ — ３　行単位の編集機能

システム・ソフト使用時、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って、その位置のキャラクタを変更することができます。

例）

```
n>LOD B:SAMPLE
          └────  カーソル位置
```

'B:' を 'A:' に変更したい場合は、←ｷｰを押してｶｰｿﾙを '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
```

このｺﾏﾝﾄﾞをｵﾌﾞｼﾞｪｸﾄのみの指定とするため、→ｷｰを押してｶｰｿﾙをｺﾏﾝﾄﾞの最後へ移動させます。

```
n>LOD A:SAMPLE
          └────  カーソル位置
```

ここで、ｽﾍﾟｰｽと '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 '<cr>' (ﾘﾀｰﾝ･ｷｰ) を入力しますとｺﾏﾝﾄﾞが実行されます。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ﾄﾞは実行されます。

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）

```
n>LOD SAMPLE C
              └──── カーソル位置
```

このｺﾏﾝﾄﾞを“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･ﾓｰﾄﾞにします。

```
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、通常のｺﾏﾝﾄﾞ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、行入力を終了する時は、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‘<cr>’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一行を呼び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び出すには、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<cr>’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び出すためには‘！！<cr>’と入力します。

例）

```
n>HIS <cr>
   1 DIR
   2 MEM F 0,1FFF C0
   3 MEM D 0XX
   4 LOD TEST C
         .
         .
         .
  19 MEM D 1XXX
  20 HIS
n>
```

4番目のｺﾏﾝﾄﾞを呼び出す場合は、

```
n>!4<cr>
  LOD TEST C
```

最新のｺﾏﾝﾄﾞを呼び出す場合は

```
n>!!<cr>
  HIS
```

となります。

・HIS ｺﾏﾝﾄﾞは、ﾋｽﾄﾘ･ﾒﾓﾘに登録されているｺﾏﾝﾄﾞを表示します。

・ﾋｽﾄﾘ･ﾒﾓﾘへのｺﾏﾝﾄﾞの登録は、ｺﾏﾝﾄﾞ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ﾄﾞは、最新の20行のｺﾏﾝﾄﾞだけです。

コマンド形式を示します。

```
n>HIS              (ｵﾍﾟﾗﾝﾄﾞ の指定はできません.)
```

表示形式を示します。

```
n>HIS <cr>
   1 MEM F 0XXX 0
   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ﾄﾞです。

登録されているｺﾏﾝﾄﾞにつけられたｺﾏﾝﾄﾞ番号です。

実行例を示し説明します。

```
n>HIS <cr>
   1 RES
   2 MEM F 0XXX 0
   3 MEM D 0XX
   4 LOD TEST C
   5 HIS  ←
n>LOD TEST1 C <cr>
  object load complete
n>HIS <cr>
   1 RES
   2 MEM F 0XXX 0
   3 MEM D 0XX
   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ﾄﾞを表示させます.)

(ﾋｽﾄﾘ･ﾒﾓﾘに登録されていたｺﾏﾝﾄﾞです.)

(この表示をするために入力したｺﾏﾝﾄﾞが最新のｺﾏﾝﾄﾞとして表示されます.)

(前のHISｺﾏﾝﾄﾞ を実行後に入力したｺﾏﾝﾄﾞが追加されています.)

このようにｺﾏﾝﾄﾞが20行をこえるまでは、実行したｺﾏﾝﾄﾞはﾋｽﾄﾘ･ﾒﾓﾘに追加されます。

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。

```
n>HIS <cr>
   1 LOD TEST C
   2 HIS
   3 LOD TEST1 C
   4 HIS
   5 MEM D 0,0FF
   6 BRA A=34
         .
         .
         .
  19 MEM D 100,10F
  20 HIS  ←
n>LOD TEST1 C <cr>
  object load complete
n>HIS <cr>
   1 LOD TEST1 C          (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。)
   2 HIS
   3 MEM D 0,0FF
   4 BRA A=34
         .
         .
         .
  18 HIS                  (前のHISｺﾏﾝﾄﾞ です。)
  19 LOD TEST1 C ←
  20 HIS  ←               (今回のHISｺﾏﾝﾄﾞ です。)
n>
```

このようにｺﾏﾝﾄﾞが20行をこえると、1 行追加する毎に古いｺﾏﾝﾄﾞ行が消去されます。

５ — ２ — ５　コマンド省略形入力

システム・モード時に、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

これらのコマンドが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）が入力された時にコマンドを表示します。

例）

```
n>A <cr>            ←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     A,#0H
                =
                  └──   カーソル位置
```

```
n>A                 ← 'A '、ｽﾍﾟ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時に、次のような表示になります。

```
n>ASM               ←ｱｾﾝﾌﾞﾗ 開始ｱﾄﾞ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。

保守／廃止

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、コマンド入力に於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）

```
n>LOD TEST.SYM  SYM1＼ ,SYM2＼ ,SYM3＼ S<cr>
                  │      └──  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というｼﾝﾎﾞﾙ･ﾃｰﾌﾞﾙ･ﾌｧｲﾙからSYM1、SYM2、SYM3というﾓｼﾞｭｰﾙ 内で定義されたﾛｰｶﾙ･ｼﾝﾎﾞﾙ だけを ﾛｰﾄﾞ 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）

```
n>LOD TEST.SYM  SYM1＼, SYM2＼, SYM1＼ S<cr>
  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）

```
n>LOD TEST.SYM  SYM4＼, SYM5＼, SYM6＼ S<cr>
  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）

この場合、全ﾊﾟﾌﾞﾘｯｸ･ｼﾝﾎﾞﾙと、SYM4,SYM5 のﾛｰｶﾙ･ｼﾝﾎﾞﾙ だけを ﾛｰﾄﾞ します。

```
n>LOD TEST.SYM  PUBLIC\, SYM4\, SYM5\ S<cr>
  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
      └──  ﾛｰﾄﾞ済みの ﾓｼﾞｭｰﾙ名
```

例）

```
n>LOD TEST.SYM  PUBLIC\, SYM3\ S<cr>
  symbol table loading
  PUBLIC      loaded module
  SYM1        pass
  SYM2        pass
  SYM3        load complete
  SYM4        pass
  SYM5        pass
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　コマンド・ヘルプ機能（ＨＬＰコマンド）

ＨＬＰコマンドは、ＩＥ－７８１３０－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                    (ｵﾍﾟﾗﾝﾄﾞを指定しない場合)

n>HLP ASM
      └──  ＩＥ－７８１３０－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 <cr>          (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 Table
  ASM          BR0          BR1          BR2          BR3
  BRA          BRD          BRE          BRS          BRM
  CLK          COM          DAS          DIR          DLY
  DOS          EXT          HIS          HLP          LOD
  LST          MAP          MAT          MDR          MEM
  MOD          MOV          PGM          REG          RES
  RGM          RUN          SAV          SPR          STP
  STR          SUF          SYM          TRG          TRM
  TRP          TRD          TRF          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 <cr>   (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><cr>       (HLPｺﾏﾝﾄﾞ を終了します。)
  n>

  n>HLP ASM <cr> (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-78130-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 TRX <cr>
  Keyword Error
HLP>TRX <cr>
    Keyword Error
HLP>
```

保守／廃止

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、‘↑Ｐ’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：　……　オープンされている出力装置をクローズします。

※　オープンされている出力装置以外の装置を新たに指定しますと、オープンされている出力装置はクローズされます。

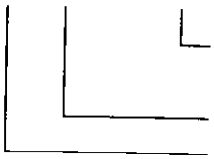
実行例を出力装置ごとに示します。

・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 <cr>
  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
n>             場合に表示されます。)

n>LST <cr>
  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 <cr>
  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>
```

・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 A: SAMPLE .LST<cr>   (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： 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： 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：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<cr>          (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>↑P (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、SYMPLE.TXT
                               にも出力されます。)

n>LST SAMPLE.LST<cr>          (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
                        ｶｰｿﾙ位置 ┘          どうかの ﾒｯｾｰｼﾞが表示されます。)


  File already exists.Delete ? (Y or N): Y<cr>
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<cr>
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。

```
n>LST SAMPLE.SYS<cr>          (指定したﾌｧｲﾙが R/O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
n>                             されます。ｺﾏﾝﾄﾞは無効になります。)
```

・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:<cr>    (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>↑P (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、‘↑P’ｷｰを入力します。‘↑P’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:<cr>
  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>
```

・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:<cr>    (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。)
n>
```

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 <cr>   (ｶﾚﾝﾄ･ﾃﾞｨﾚｸﾄﾘのすべてのﾌｧｲﾙを表示します。)
  Directory = A:¥
    COMMAND  COM    LC       EXE    LC1      EXE    LC2      EXE    DUMP     COM
    OML      EXE    FORMAT   EXE    DEBUG    COM    PR       EXE    ASSERT   H
    CTYPE    H      DOS      H      ERROR    H      FCNTL    H      FCTYPE   H
    FLOAT    H      IOS1     H      LIMITS   H      LOCKING  H      MATH     H
    PC       H      SETJMP   H      SIGNAL   H      STDIO    H      STDLIB   H
    STRING   H      TIME     H      UNLSTD   H      8087     MAC    KANJI    H
    PRINT    EXE    FIND     EXE    MORE     COM    SORT     EXE    MASM     EXE
    LINK     EXE    LABEL    EXE    SHARE    EXE    DISKCOPY COM    KEY      COM
    SYMDEB   EXE    AUTOEXEC BAT    AAA      HEX    LINK2    EXE    LINKS    BAK
    TEST     OBJ    README   DOC    CS       OBJ    _MAINS   OBJ    NDPS     OBJ
    NONDPS   LIB    KANJIS   LIB    SM8086   MAC    TEST     BAK    LCS      LIB
    TEST     MAP    TEST     EXE    LINKS    BAT    TEST     C      SVI      HLP
    SVI      EXE    LCS      BAT    LINKMS   BAT    GET      OVL    INI      OVL
    REC      OVL
n>DIR *.COM<cr>  (拡張子が COMというﾌｧｲﾙ名のﾃﾞｨﾚｸﾄﾘを表示します。)
  Directory = A:¥
    COMMAND  COM    DUMP     COM    DEBUG    COM    MORE     COM    DISKCOPY COM
    KEY      COM
n>DIR TEST.*<cr>  (ﾌｧｲﾙ名が TESTというﾃﾞｨﾚｸﾄﾘを表示します。)
  Directory = A:¥
    TEST     OBJ    TEST     BAK    TEST     MAP    TEST     EXE    TEST     C
n>DIR T*.*<cr>   (ﾌｧｲﾙ名が Tで始まるすべてのﾃﾞｨﾚｸﾄﾘを表示します。)
  Directory = A:¥
    TIME     H      TEST     OBJ    TEST     BAK    TEST     MAP    TEST     EXE
    TEST     C
n>DIR *.H??<cr>  (拡張子が ３文字でHで始まるすべてのﾃﾞｨﾚｸﾄﾘを表示します。)
  Directory = A:¥
    AAA      HEX    SVI      HLP
n>
```

付　録

（1）設置方法の概要

最初に付属品をＩＥ－７８１３０本体に接続します。

- ＩＥ－７８１３０本体にターゲット・プローブを接続する場合、ＩＥ－７８１３０本体の上面からエミュレーション・ボードのコネクタ（ＣＮ１）に、ターゲット・プローブのコネクタを差します。その際プローブ・ケーブルは正面より見て右に出るようにします。
- ＩＥ－７８１３０本体に電源ケーブルを接続する場合、ＩＥ－７８１３０本体の裏側のＡＣインレットに差し込みます。
- ＩＥ－７８１３０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－７８１３０本体の側面パネルのチャネル１、あるいはチャネル２に差します。
- ＩＥ－７８１３０本体にプリンター・ケーブルを接続する場合、ＩＥ－７８１３０本体の側面パネルのチャネル３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	……	PC－9801シリーズ
		IBM PC－XT／AT
PROMプログラマ	……	PG－1500／PG－2000
ターミナル	……	MD－910TM

（３）筐体につ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－７８１３０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－７８１３０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

（サイド・パネルのカバーの中にあります。）

・・・・　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（サイド・パネルのカバーの中にあります。）

・・・・　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（サイド・パネルのカバーの中にあります。）

・・・・　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（サイド・パネルのカバーの中にあります。）

・・・・　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）ＩＥ　Ｃ／Ｔ　ボードのジャンパの設定

ＩＥ　Ｃ／Ｔ　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。

ｼﾞｬﾝﾊﾟ NO.	設　　定
ＪＰ　１	１－２　ショート

表　付－（４）－１　ＩＥ　Ｃ／Ｔ　ボードの
出荷時のジャンパ設定

（５）ブレーク・ボードのジャンパの設定

ブレーク・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。

ｼﾞｬﾝﾊﾟ NO.	設　　定
ＪＰ　１	オープン
ＪＰ　２	１－４　ショート
ＪＰ　３	ショート

表　付－（５）－１　ブレーク・ボードの出荷時のジャンパ設定

（６）エミュレーション・ボードのジャンパの設定

エミュレーション・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。

ｼﾞｬﾝﾊﾟ NO.	設　　定
ＪＰ　４	１－２　ショート
ＪＰ　５	１－２　ショート
ＪＰ　１１	１－２　ショート
ＪＰ　１２	１－２　ショート

表　付－（６）－１　エミュレーション・ボード
の出荷時のジャンパ設定

（7） ＲＳ－２３２－Ｃインタフェース回路

ＩＥ－７８１３０は、筐体のフロント・パネル面にＲＳ－２３２－Ｃインターフェース用コネクタを２チャネル（チャネル１，チヤネル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。

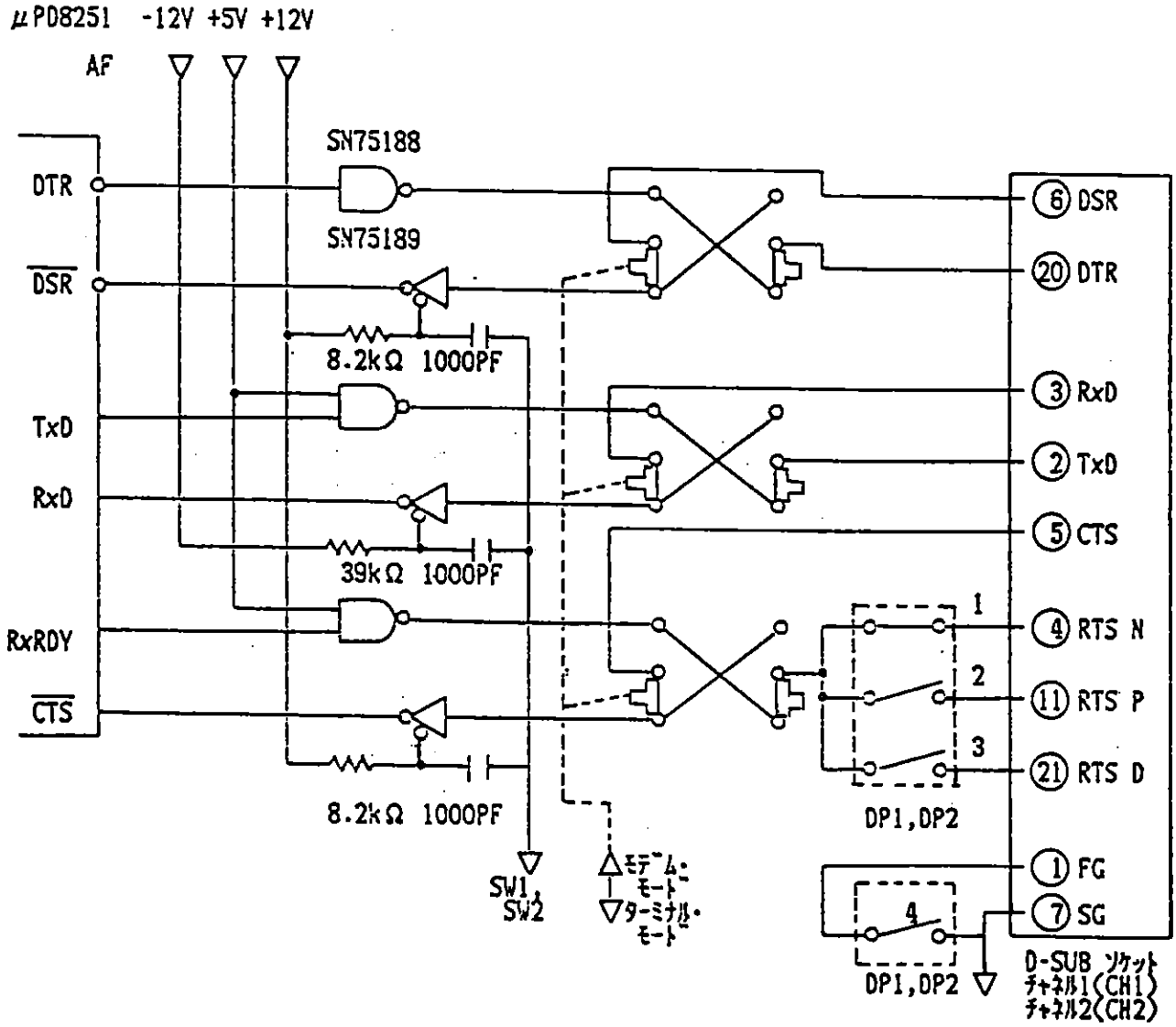


図 付－（７）－１ ＲＳ－２３２－Ｃインタフェース回路図

（8） ターゲットとの接続方法

ターゲット・プローブは別売です。

ターゲット・プローブをターゲット・システムのＣＰＵソケットに差し込む場合。

- ターゲット・プローブのアースクリップをターゲットのＧＮＤ（シグナル・グランド）に接続してください。
- 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ソケットに差し込みます。

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ＴＴＬレベル信号線だけに接続してください。
- 接続する時は、ＩＣクリップを使用してください。

電源投入順序

① ＩＥ－７８１３０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－７８１３０が正常に動作しません。

又、ＩＥ－７８１３０を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－７８１３０の電源スイッチを切ります。

この順序を間違えますとＩＥ－７８１３０を破壊することがありますので注意してください。

（9） 実際のディバイスとの差

実際のディバイス（μＰＤ７８１３４／７８１３Ｘ）はＣ－ＭＯＳの回路ですがＩＥ－７８１３０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・チップから直接取り出されている信号
ポート関係，Ａ／Ｄコンバータ関係は、実際のディバイスとまったく同じ動作をします。
注］但し、Ａ／Ｄコンバータ関係を除いて直列に１００Ω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り出されている信号
ＲＥＳＥＴは、ＣＭＯＳがはい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り出されている信号
ＶDD，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ＶDDはエバリュエーション・チップには、接続されません。

（１０） ディバッガ使用上の注意

・ＲＳ－２３２－Ｃインターフェース

- 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でください。但し、ＰＧ－２０００と接続の場合、ＩＥ－７８１３０は、モデム・モードに設定します。その際、接続にはＰＧ－２０００に添付されているＲＳ－２３２－Ｃインターフエース・ケーブルを必ず使用して下さい。
  又、ＰＧ－１５００との接続の場合、ＩＥ－７８１３０は、ターミナルモードに設定します。接続には、汎用のＲＳ－２３２－Ｃインタフェース・ケーブルを使用して下さい。

- 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。また、下線部入力後のリターン・キー”〈cr〉”入力は、本例では省略されています。）

```
A>IE78130
XX:XX:XX A:IE78130.COM
  IE-78130 CONTROLLER     (PC-9801 SERIES) Vx.x  (Dd Mmm Yy)
       Copyright  (C) 1989 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-78130 for NEW FRAME Monitor Vx.x  (Dd Mmm Yy)
    Copyright (C) 1989 by NEC Corporation

  Power on target system (Y/N)  Y
  Create new set up mode (Y or N) : N
  Internal ROM size(4K,8K,12K,16K,20K,24K,28K,32K)= 16K
  Internal RAM size (128,256,384,512,640,768,896,1024) = 128 〈cr〉
  Tracer  initialize
  Breaker initialize
  Do you have Memory Board on IE-78130-R ? (Y/N)= N
1)CLK I
1)RES
1)MEM F 0,3FFF 00
1)LOD SORT
  object load complete
  symbol table loading
  MOD01       load complete
1)LOD SORT.HEX C
  object load complete
1)SYM K
1)LOD SORT.SYM S
  symbol table loading
  MOD01       load complete
1)SYM M
              = MOD01 \
1)SYM M
  MOD01 \     =
```


```
1)MEM D 100,133
  0100   3A 80 01 3A 81 00 20 AA 9F 81 80 08 6F 80 00 00    :..:.. .....o...
  0110   00 00 14 FB B8 00 66 A2 FE D8 88 0E 24 68 5D 47    ......f.....$h]G
  0120   16 5F 83 0B 81 09 D8 5D D8 55 D8 4F 55 26 80 26    ._.....].U.OU&.&
  0130   81 14 D3 00                                        ....
1)MEM D 100
  0100   3A 80 01 3A 81 00 20 AA 9F 81 80 08 6F 80 00 00    :..:.. .....o...
  0110   00 00 14 FB B8 00 66 A2 FE D8 88 0E 24 68 5D 47    ......f.....$h]G
  0120   16 5F 83 0B 81 09 D8 5D D8 55 D8 4F 55 26 80 26    ._.....].U.OU&.&
  0130   81 14 D3 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)MEM D 4000,6FFF
  Mapping error
```


```
1)DAS 100,133
  Addr  Object          Mnemonic
                        ORG     MOD01 \SORT
                      MOD01 \SORT:
  0100  3A 80 01        MOV     0FE80H,#1H
  0103  3A 81 00        MOV     0FE81H,#0H
                      MOD01 \COMP:
  0106  20 AA           MOV     A,0FEAAH
  0108  9F 81           CMP     A,0FE81H
  010A  80 08           BNZ     $CONT
  010C  6F 80 00        CMP     0FE80H,#0H
                      MOD01 \STOP:
  010F  00              NOP
  0110  00              NOP
  0111  00              NOP
  0112  14 FB           BR      $STOP
                      MOD01 \CONT:
  0114  B8 00           MOV     X,#0H
  0116  66 A2 FE        MOVW    HL,#0FEA2H
  0119  D8              XCH     A,X
  011A  88 0E           ADDW    AX,HL
  011C  24 68           MOVW    HL,AX
  011E  5D              MOV     A,[HL]
  011F  47              INCW    HL
  0120  16 5F           CMP     A,[HL]
  0122  83 0B           BC      $INCI
  0124  81 09           BZ      $INCI
  0126  D8              XCH     A,X
  0127  5D              MOV     A,[HL]
  0128  D8              XCH     A,X
  0129  55              MOV     [HL],A
  012A  D8              XCH     A,X
  012B  4F              DECW    HL
  012C  55              MOV     [HL],A
  012D  26 80           INC     0FE80H
                      MOD01 \INCI:
  012F  26 81           INC     0FE81H
  0131  14 D3           BR      $COMP
  0133  00              NOP
                        END
```


```
1)DAS 100
  Addr  Object           Mnemonic
                         ORG     MOD01 \SORT
                       MOD01 \SORT:
  0100  3A 80 01         MOV     0FE80H,#1H
  0103  3A 81 00         MOV     0FE81H,#0H
                       MOD01 \COMP:
  0106  20 AA            MOV     A,0FEAAH
  0108  9F 81            CMP     A,0FE81H
  010A  80 08            BNZ     $CONT
  010C  6F 80 00         CMP     0FE80H,#0H
                         END
1)DAS
  Addr  Object           Mnemonic
                         ORG     MOD01 \STOP
                       MOD01 \STOP:
  010F  00               NOP
  0110  00               NOP
  0111  00               NOP
  0112  14 FB            BR      $STOP
                       MOD01 \CONT:
  0114  B8 00            MOV     X,#0H
  0116  66 62 FE         MOVW    HL,#0FEA2H
                         END
1)SYM A SW 0FE80
1)SYM A I 0FE81
1)SYM A STACK 0FEF0
1)SYM A LIST 0FEA2
1)SYM A N 0FEAA
1)MEM F 0FE80,0FEDF 0
1)MEM C 0FEA2
  FEA2  00 05
  FEA3  00 03
  FEA4  00 04
  FEA5  00 0A
  FEA6  00 08
  FEA7  00 82
  FEA8  00 0A
  FEA9  00 04
  FEAA  00 08
  FEAB  00 .
1)MEM D 0FE80,0FEAA
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  FEA0   00 00 05 03 04 0A 08 82 0A 04 08                  ...........
1)MEM D SW,N
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  FEA0   00 00 05 03 04 0A 08 82 0A 04 08                  ...........
```


```
1)MEM D 0FE80,N
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEAA   00 00 05 03 04 0A 08 82 0A 04 08                    ...........
1)MEM D SW,0FEAA
  FE80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   00 00 05 03 04 0A 08 82 0A 04 08                    ...........
1)BRS P STOP
1)BRS
  Parallel
  STOP
1)BRM BRS
1)BRM
  BRS
1)REG C PC
  PC       0000 = 100
  SP         72 = 0E0
1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  Standard break  terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0    0   0    0    1   0  A2    00    00    00    FFFF    FEA3   010F  E000
  One step emulation standby ESC ｷｰ 入力
1)MEM D SW,N
  FE80   03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   00 00 03 05 04 0A 08 82 0A 04 00                    ...........
1)MEM C LIST
  FEA2  03 05
  FEA3  05 03
  FEA4  04
  FEA5  0A
  FEA6  08
  FEA7  82
  FEA8  0A
  FEA9  04
  FEAA  00 08
  FEAB  08 00
  FEAC  00 .
1)REG C PC
  PC       010F = 100
  SP         E0 = .
```


```
1)RUN T ,6 REG
  User-system Vcc-ON     Emulation start at 0100
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  A2    00    00    00    FFFF    FEA3   0103  E000
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  A2    00    00    00    FFFF    FEA3   0106  E000
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   0108  E000
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   010A  E000
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  A2    08    00    00    FFFF    FEA3   0114  E000
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  00    08    00    00    FFFF    FEA3   0116  E000
      terminated
   Frame Address Label  Mnemonic
                 MOD01  \CONT:
   0000   0114          MOV     X,#0H
  One step emulation standby〈cr〉ｷｰ入力
   Frame Address Label  Mnemonic
   0000   0116          MOVW    HL,#LIST
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  00    08    00    00    FFFF    FEA2   0119  E000
  One step emulation standby ESC ｷｰ 入力
1)ASM CONT
  0114  MOD01 \CONT:
  0114                MOV     X,#0H
              = BR $133
        14 1D
  0116                MOVW    HL,#LIST
              = ORG 133H
  0133                NOP
              = MOV A,I
        20 81
  0135                NOP
              = MOV X,#0H
        B8 00
  0137                NOP
              = BR $116
        14 DD
  0139                NOP
              = END
1)DAS 114,116
  Addr  Object            Mnemonic
                          ORG     MOD01 \CONT
                        MOD01 \CONT:
  0114  14 1D             BR      $133H
  0116  66 A2 FE          MOVW    HL,#LIST
                          END
```


```
1)DAS 133,138
  Addr  Object              Mnemonic
                            ORG     133H
  0133  20 81               MOV     A,I
  0135  B8 00               MOV     X,#0H
  0137  14 DD               BR      $116H
                            END
1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  SFR illegal access break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   1  FF    02    00    00    FFFF    FF02   012A  E000
1)MEM D SW,N
  FE80   73 BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00     s...............
  FE90   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
  FEA0   00 00 03 04 05 08 0A 0A 04 08 00                    ...........
1)ASM 10C
  010C                           CMP     SW,#0H
                          = BR $139
        14 2B
  010E                           NOP
                          = NOP
        00
  010F  MOD01 \STOP:
  010F                           NOP
                          = END
1)ASM 139
  0139                           NOP
                          = CMP SW,#0H
        6F 80 00
  013C                           NOP
                          = BNZ  $CONT
        80 D6
  013E                           NOP
                          = BR $STOP
        14 CF
  0140                           NOP
                          = END
1)DAS 10C,10E
  Addr  Object              Mnemonic
                            ORG     10CH
  010C  14 2B               BR      $139H
  010E  00                  NOP
                            END
1)DAS 139,13F
  Addr  Object              Mnemonic
                            ORG     139H
  0139  6F 80 00            CMP     SW,#0H
  013C  80 D6               BNZ     $CONT
  013E  14 CF               BR      $STOP
                            END
```


```
1)DAS CONT
  Addr  Object              Mnemonic
                            ORG     MOD01 \CONT
                          MOD01 \CONT:
  0114  14 1D               BR      $133H
  0116  66 A2 FE            MOVW    HL,#LIST
  0119  D8                  XCH     A,X
  011A  88 0E               ADDW    AX,HL
  011C  24 68               MOVW    HL,AX
  011E  5D                  MOV     A,[HL]
  011F  47                  INCW    HL
                            END
1)DAS
  Addr  Object              Mnemonic
                            ORG     120H
  0120  16 5F               CMP     A,[HL]
  0122  83 0B               BC      $INC1
  0124  81 09               BZ      $INC1
  0126  D8                  XCH     A,X
  0127  5D                  MOV     A,[HL]
  0128  D8                  XCH     A,X
  0129  55                  MOV     [HL],A
  012A  D8                  XCH     A,X
                            END
1)MEM F LIST,N 5,3,4,0A,8,82,0A,4,8
1)BRS P 122

1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    A2    00    00    FFFF    FEA3   0127  E000
  One step emulation standby<cr>*-
   Frame Address Label  Mnemonic
   0000   0127          MOV     A,[HL]
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA3   0128  E000
  One step emulation standby<cr>*-
   Frame Address Label  Mnemonic
   0000   0128          XCH     A,X
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  03    05    00    00    FFFF    FEA3   0129  E000
  One step emulation standby<cr>*-
   Frame Address Label  Mnemonic
   0000   0129          MOV     [HL],A
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  03    05    00    00    FFFF    FEA3   012A  E000
```


```
  One step emulation standby〈cr〉ｷｰ
   Frame Address Label  Mnemonic
   0000   012A          XCH     A,X
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA3   012B  E000
  One step emulation standby〈cr〉ｷｰ
   Frame Address Label  Mnemonic
   0000   012B          DECW    HL
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012C  E000
  One step emulation standby〈cr〉ｷｰ
   Frame Address Label  Mnemonic
   0000   012C          MOV     [HL],A
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012D  E000
  One step emulation standby〈cr〉ｷｰ
   Frame Address Label  Mnemonic
   0000   012D          INC     SW
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   012F  E000
  One step emulation standby〈cr〉ｷｰ
   Frame Address Label  Mnemonic
                MOD01   ＼INC1:
  0000   012F          INC     I
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0131  E000
  One step emulation standby ESCｷｰ入力
1)MEM D LIST,LIST+7
  FEA0          03 05 04 0A 08 82 0A 04                          ........
1)MEM D I,I
  FE80       01                                                  .

1)RUN T ,1
  User-system Vcc-ON     Emulation start at 0131
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR      $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0106  E000
  One step emulation standby〈cr〉 ｷｰ 入力
   Frame Address Label  Mnemonic
                MOD01   ＼COMP:
   0000   0106          MOV     A,N
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    08    00    00    FFFF    FEA2   0108  E000
  One step emulation standby ESCｷｰ入力
1)BRS P INC1
```


```
1)RUN B
  User-system Vcc-ON      Emulation start at 0108
  Standard break     terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0106  E000
  One step emulation standby ESCキー入力
1)MEM D I,I
  FE80       02
1)MEM D LIST,LIST+7
  FEA0          03 04 05 0A 08 82 0A 04                          ........
1)RUN T ,I
  User-system Vcc-ON      Emulation start at 0106
      terminated
   Frame Address Label  Mnemonic
                 MOD01  \COMP:
   0000   0106          MOV      A,N
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    08    00    00    FFFF    FEA3   0108  E000
  One step emulation standby ESCキー入力
1)RUN B
  User-system Vcc-ON      Emulation start at 0108
  Standard break     terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   1  A4    05    00    00    FFFF    FEA5   0106  E000
  One step emulation standby ESCキー入力
1)MEM D I,I
  FE80       03
1)MEM D LIST,LIST+7
  FEA0          03 04 05 0A 08 82 0A 04                          ........
1)BRS P STOP 139
1)RUN B
  User-system Vcc-ON      Emulation start at 0106
  SFR illegal access braek     terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  FF    02    00    00    FFFF    FF02   012A  E000
1)MEM D LIST,N
  FEA0          03 04 05 08 0A 0A 04 08 00                       .........
1)MEM F LIST,N 5,3,4,0A,8,82,0A,4,8
1)DAS 12D,132
  Addr  Object              Mnemonic
                            ORG      12DH
  012D  26 40               INC      SW
                       MOD01 \INCI:
  012F  26 41               INC      I
  0131  14 D3               BR       $COMP
                            END
1)BRS P 12D
```


```
1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0131  E000
  One step emulation standby ESCｷｰ入力
1)MEM D I,I
  FE80      01                                                    .
1)MEM D LIST,LIST+7
  FEA0         03 05 04 0A 08 82 0A 04                           ........
1)RUN T ,1
  User-system Vcc-ON     Emulation start at 0131
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR      $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    03    00    00    FFFF    FEA2   0106  E000
  One step emulation standby ESCｷｰ入力
1)RUN B
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0131  E000
  One step emulation standby ESCｷｰ入力
1)MEM D I,I
  FE80      02                                                    .
1)MEM D LIST,LIST+7
  FEA0         03 04 05 0A 08 82 0A 04                           ........
1)RUN T ,1
  User-system Vcc-ON     Emulation start at 0131
      terminated
   Frame Address Label  Mnemonic
   0000   0131          BR      $COMP
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  05    04    00    00    FFFF    FEA3   0106  E000
  One step emulation standby ESCｷｰ入力
1)RUN B
  User-system Vcc-ON     Emulation start at 0106
  Standard break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0   0   0   0    1   0  0A    08    00    00    FFFF    FEA5   0131  E000
  One step emulation standby ESCｷｰ入力

1)MEM D I,I
  FE80      04                                                    .
1)MEM D LIST,LIST+7
  FEA0         03 04 05 08 0A 82 0A 04                           ........
1)MEM D SW,SW
  FE80   04                                                      .
```


```
1)ASM 13C
  013C                              BNZ     $CONT
                          = BNZ  $SORT
        80 C2
  013E                              BR      $STOP
                          = END
1)DAS 13C,13D
  Addr  Object              Mnemonic
                            ORG     13CH
  013C  80 C2               BNZ     $SORT
                            END
1)MEM F LIST,N 5,3,4,0A,8,82,0A,4,8
1)TRM D
1)BRA A=N C=W
1)BRM BRA
1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  Standard break    terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   0    0   0    0    1   0  08    82    00    00    FFFF    FEA9   012E  E000
  One step emulation standby ESCｷｰ入力
1)MEM D N,N
  FEA0                                  82                              .
1)MEM D LIST,N
  FEA0          03 04 05 08 0A 0A 04 82 82                      .........
1)TRP N
1)TRP -9T
1)TRD F
   Frame  Status  Address Data    7--EX--0
   0073     WR     0FE80   06     00000000
   0074     RD     0FE81   06     00000000
   0075     WR     0FE81   07     00000000
   0076     RD     0FEAA   08     00000000
   0077     RD     0FE81   07     00000000
   0078     RD     0FE81   07     00000000
   0079     RD     0FEA9   82     00000000
   0080     RD     0FEAA   08     00000000
   0081     RD     0FEAA   08     00000000
  T0082     WR     0FEAA   82     00000000
   Total frame = 0083   (N/O/T/+/cr/Frame No./.) ? .
```


```
1)DAS 106,109
  Addr  Object              Mnemonic
                            ORG     MOD01 \COMP
                          MOD01 \COMP:
  0106  20 AA               MOV     A,N
  0108  9F 81               CMP     A,I
                            END
1)ASM 106
  0106  MOD01 \COMP:
  0106                      MOV     A,N
                  = BR $140
        14 38
  0108                      CMP     A,I
                  = END
1)ASM 140
  0140                      NOP
                  = MOV A,N
        20 AA
  0142                      NOP
                  = DEC A
        C9
  0143                      NOP
                  = CMP A,I
        9F 81
  0145                      NOP
                  = BNZ  $CONT
        80 CD
  0147                      NOP
                  = BR $10C
        14 C3
  0149                      NOP
                  = END
1)ASM SORT
  0100  MOD01 \SORT:
  0100                      MOV     SW,#1H
                  = MOV SW,#0H
        3A 80 00
  0103                      MOV     I,#0H
                  = END
1)SYM C LIST 0FE82
1)SYM C N 0FE8A
1)SYM E STACK
1)MEM C 117
  0117  A2 82
  0118  FE .
1)MEM C 141
  0141  AA 8A
  0142  C9 .
```

1)DAS 100,149

Addr	Object	Mnemonic	
		ORG	MOD01 \SORT
		MOD01 \SORT:	
0100	3A 80 00	MOV	SW,#0H
0103	3A 81 00	MOV	I,#0H
		MOD01 \COMP:	
0106	14 38	BR	$140H
0108	9F 81	CMP	A,I
010A	80 08	BNZ	$CONT
010C	14 2B	BR	$139H
010E	00	NOP	
		MOD01 \STOP:	
010F	00	NOP	
0110	00	NOP	
0111	00	NOP	
0112	14 FB	BR	$STOP
		MOD01 \CONT:	
0114	14 1D	BR	$133H
0116	66 82 FE	MOVW	HL,#LIST
0119	D8	XCH	A,X
011A	88 0E	ADDW	AX,HL
011C	24 68	MOVW	HL,AX
011E	59	MOV	A,[HL]
011F	47	INCW	HL
0120	16 5F	CMP	A,[HL]
0122	83 0B	BC	$INC1
0124	81 09	BZ	$INC1
0126	D8	XCH	A,X
0127	5D	MOV	A,[HL]
0128	D8	XCH	A,X
0129	55	MOV	[HL],A
012A	D8	XCH	A,X
012B	4F	DECW	HL
012C	55	MOV	[HL],A
012D	26 80	INC	SW
		MOD01 \INC1:	
012F	26 81	INC	I
0131	14 D3	BR	$COMP
0133	20 81	MOV	A,I
0135	B8 00	MOV	X,#0H
0137	14 DD	BR	$116H
0139	6F 80 00	CMP	SW,#0H
013C	80 C2	BNZ	$SORT
013E	14 CF	BR	$STOP
0140	20 8A	MOV	A,N
0142	C9	DEC	A
0143	9F 81	CMP	A,I
0145	80 CD	BNZ	$CONT


```
  0147  14 C3                BR      $10CH
  0149  00                   NOP
                             END
1)MEM F LIST,N 5,3,4,0A,8,82,0A,4,8
1)BRM BRS
1)BRS P STOP
1)RUN B 100
  User-system Vcc-ON     Emulation start at 0100
  Standard break     terminated
   IE   Z RBS1 AC RBS0  ISP CY X(R0) A(R1) C(R2) B(R3) DE(RP2) HL(RP3)  PC    SP
    0   1   0   0   0    1   0  88    07    00    00    FFFF    FE89   010F  E000
  One step emulation standby ESCｷｰ入力
1)MEM D LIST,LIST+7
  FE80          03 04 04 05 08 0A 0A 82                          ........
1)MEM D N,N
  FE80                                  08                             .
1)MEM D SW,I
  FE80    00 07                                                    ..
1)SAV SORT01.HEX 100,149
  object save complete
1)VRY SORT01.HEX
  object verify complete
1)
```

（１２）　コマンド一覧

次頁　以降に　コマンド一覧を示します。

保守／廃止

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
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	アクセス系ブレーク条件設定	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= 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･ﾌｪｯﾁを除くすべてのﾘｰﾄﾞ/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	[bit] (外部ｾﾝｽ 信号のﾌﾞﾚｰｸ･ﾋﾞｯﾄ･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	[byte] (ﾌｪｯﾁ命令数)
	フェッチ系ブレーク条件設定	ＢＲＳ △	[S / P]	[word][_word][_word][_word]
		ＢＲＭ △	なし	[_BRA][_BRD][_BRE][_BRS][_BR0][_BR1][_BR2][_BR3] (ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ名)
論理ブレーク条件設定		ＢＲ０ △ ＢＲ１ △ ＢＲ２ △ ＢＲ３ △	なし	[_BRA][_BRD][_BRE][_BRS] (物理ﾌﾞﾚｰｸ･ﾚｼﾞｽﾀ 名)

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
クロック選択	ＣＬＫ	なし	[[ ] I(IE)]
コマンド・ファイル作成	ＣＯＭ△☆	なし	[[LST: CON: file(ｺﾏﾝﾄﾞ・ﾌｧｲﾙ 名)]]
逆アセンブラ	ＤＡＳ	なし	[[word(逆ｱｾﾝﾌﾞﾗ のｽﾀｰﾄ・ｱﾄﾞﾚｽ) partition (逆ｱｾﾝﾌﾞﾗ のｽﾀｰﾄ・ｱﾄﾞﾚｽ とｴﾝﾄﾞ・ｱﾄﾞﾚｽ)]]
ディレクトリ表示	ＤＩＲ△☆	なし	[file] (ﾌｧｲﾙ名)
ディレイ・カウンタ設定	ＤＬＹ △	なし	[word]
子プロセスの実行	ＤＯＳ△☆	なし	なし EXIT<ｃｒ>を入力するとｼｽﾃﾑ・ｿﾌﾄｳｪｱに戻ります。
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] (表示したいｺﾏﾝﾄﾞのｺﾏﾝﾄﾞ本体)
オブジェクト・ロード	ＬＯＤ □	なし	[[TTY1(ﾁｬﾈﾙ1またはﾁｬﾈﾙ4) TTY2(ﾁｬﾈﾙ2)]] ﾁｬﾈﾙ1(ｼﾘｱﾙ・ｲﾝﾀﾌｪｰｽ)またはﾁｬﾈﾙ4(ﾊﾟﾗﾚﾙ・ｲﾝﾀﾌｪｰｽ)の設定はIE起動時に行います。
オブジェクト／ シンボル／ディバグ 環境のロード	ＬＯＤ ☆	なし	file [_module name＼･･･][_C][_S][_D] ↓ ↓ (ｵﾌﾞｼﾞｪｸﾄ/ｼﾝﾎﾞﾙ・ﾌｧｲﾙ 名) (ﾓｼﾞｭｰﾙ名) C･･･ｵﾌﾞｼﾞｪｸﾄ指定 S･･･ｼﾝﾎﾞﾙ指定 D･･･ﾃﾞｨﾊﾞｸﾞ環境指定
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)]]

保守／廃止

保守／廃止

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
マッピング設定	ＭＡＰ	[W / R / U / K]	[partition] (ﾏｯﾋﾟﾝｸﾞ範囲) (W:内部ﾏｯﾋﾟﾝｸﾞ、R:ﾗｲﾄ・ﾌﾟﾛﾃｸﾄ 内部ﾏｯﾋﾟﾝｸﾞ、U:ﾕｰｻﾞ・ﾏｯﾋﾟﾝｸﾞ、K:ﾏｯﾋﾟﾝｸﾞ 解除)
演算	ＭＡＴ △		word (通常は式を記述する)
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･ｱﾄﾞﾚｽ)
		[D](表示)	[word(表示ｽﾀｰﾄ･ｱﾄﾞﾚｽ) / partition (表示ｽﾀｰﾄ･ｱﾄﾞﾚｽ と表示ｴﾝﾄﾞ･ｱﾄﾞﾚｽ)]
		F(ｲﾆｼｬﾗｲｽﾞ)	partition_data string ←(ｲﾆｼｬﾗｲｽﾞ･ﾃﾞｰﾀ(8ﾋﾞｯﾄ) の集まり) → (ｲﾆｼｬﾗｲｽﾞ･ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		G(ｻｰﾁ)	partition_data string ←(ｻｰﾁ･ﾃﾞｰﾀ(8ﾋﾞｯﾄ) の集まり) → (ｻｰﾁ･ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		M(ｺﾋﾟｰ)	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)
		X(交換)	partition_word ←(交換先ｽﾀｰﾄ･ｱﾄﾞﾚｽ) → (交換元ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		V(比較)	partition_word ←(比較先ｽﾀｰﾄ･ｱﾄﾞﾚｽ) → (比較元ｽﾀｰﾄ･ｱﾄﾞﾚｽ とｴﾝﾄﾞ･ｱﾄﾞﾚｽ)
		E(ﾃｽﾄ)	[partition] (ﾃｽﾄ･ｽﾀｰﾄ･ｱﾄﾞﾚｽとｴﾝﾄﾞ･ｱﾄﾞﾚｽ)

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
チャネル２モード設定	ＭＯＤ △	なし	MODE= [CHAR / FLOW] (ﾊﾝﾄﾞｼｪｰｸ・ﾓｰﾄﾞ) ／ BAUD= [19200 / 9600 / 4800 / 2400 / 1200 / 600 / 300] (ﾎﾞｰ・ﾚｰﾄ) ／ LONG= [7 / 8] (ｷｬﾗｸﾀ長) ／ PAR= [NON / EVEN / ODD] (ﾊﾟﾘﾃｨ・ﾋﾞｯﾄ) ／ STOP= [1 / 2] (ｽﾄｯﾌﾟ・ﾋﾞｯﾄ長) （各オペランドは＿で区切って入力する）
ＩＥ代替メモリ↔ユーザ・メモリ間のデータ転送	ＭＯＶ	[U / I]	partition_word ←(転送先ｽﾀｰﾄ・ｱﾄﾞﾚｽ) → (転送元ｽﾀｰﾄ・ｱﾄﾞﾚｽ とｴﾝﾄﾞ・ｱﾄﾞﾚｽ) (U:IE代替ﾒﾓﾘ→ﾕｰｻﾞ・ﾒﾓﾘ / I:ﾕｰｻﾞ・ﾒﾓﾘ→IE代替ﾒﾓﾘ)
端末モード	ＰＧＭ	[C]	なし
レジスタ操作	ＲＥＧ	C(変更)	[register name]
		[D](表示)	[ALL / register name]
レジスタ・モード設定	ＲＧＭ △	なし	[I / G] (I: ｲﾝﾌﾟﾗｲﾄﾞ・ﾓｰﾄﾞ , G:ｼﾞｪﾈﾗﾙ・ﾓｰﾄﾞ)

保守／廃止

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
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][,byte]（word:ｽﾀｰﾄ･ｱﾄﾞﾚｽ,byte:ｽﾃｯﾌﾟ 数）（S:ｽﾃｯﾌﾟ数指定ﾘｱﾙﾀｲﾑ実行）
		T(ﾄﾚｰｽ実行)	[word] [※1 / 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、※1 はﾚｼﾞｽﾀ条件、word はｽﾃｯﾌﾟ数） ※1 [register name / PSWの ﾌﾗｸﾞ名] [= > < => >= =< <= >< <>] [= bit] ※2 [mask / wmask] ※2 ﾏｽｸ 表現の場合は、'='、'><'、'<>' のみ有効
リセット	ＲＥＳ	なし	[H] (H: 省略時は ｴﾊﾞ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／ディバグ環境のセーブ	ＳＡＶ □	なし	[TTY1(ﾁｬﾈﾙ1) / TTY2(ﾁｬﾈﾙ2)] [_partition][_partition] ・・・・ [_partition]　最大５つまで
	ＳＡＶ ☆	なし	file(入力ﾌｧｲﾙ名) [_partition][_partition] ・・・[_partition][_C][_D]　最大５つまで　C･･･ｵﾌﾞｼﾞｪｸﾄ　D･･･ﾃﾞｨﾊﾞｸﾞ環境
特殊レジスタ操作	ＳＰＲ	C(変更)	[special register name]
		[D](表示)	[special register name]
ｴﾐｭﾚｰｼｮﾝCPU 停止	ＳＴＰ ▲	なし	なし
入力ﾃﾞｨﾊﾞｲｽ･ﾘﾀﾞｲﾚｸﾄ	ＳＴＲ△☆	なし	file（入力ﾌｧｲﾙ名）

保守／廃止

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
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)
	ＳＹＭ ☆	L(ﾛｰﾄﾞ)	なし
		S(ｾｰﾌﾞ)	なし
シンボル操作	ＳＹＭ△☆	[D](表示)	[module＼(ﾓｼﾞｭｰﾙ指定) / PUBLIC (ﾊﾟﾌﾞﾘｯｸ指定)]
ｶﾚﾝﾄ・ﾓｼﾞｭｰﾙ指定	ＳＹＭ	M	なし
トレーサの再起動	ＴＲＧ ▲	なし	なし
トレース・モード設定	ＴＲＭ △	なし	[P / D] (P:ﾌﾟﾛｸﾞﾗﾑのﾄﾚｰｽ / D:ﾘｰﾄﾞ・ﾗｲﾄのﾄﾚｰｽ)

保守／廃止

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・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） T（ﾎﾟｲﾝﾀをﾄﾘｶﾞ･ﾎﾟｲﾝﾄに置く）
トレース表示	ＴＲＤ　△	F I	[_ALL（ﾄﾚｰｽ 結果すべてを表示）] [$Q / $F] （F:ﾌﾚｰﾑ･ﾓｰﾄﾞ 、I:ｲﾝｽﾄﾗｸｼｮﾝ･ﾓｰﾄﾞ） （$Q:指定ﾌﾚｰﾑのみ表示、$F: 指定ﾌﾚｰﾑの前後5 行表示）
トレース・データ 検索条件設定	ＴＲＦ　△	なし	[A=addrs][V=mask] ↓ （検索ｱﾄﾞﾚｽ） ↓ （検索ﾃﾞｰﾀ） C= BRM1(BRM1ﾌｪｯﾁ) / M1 (M1ﾌｪｯﾁ) / OP (ｵﾍﾟｺｰﾄﾞ･ﾌｪｯﾁ) / VECT(ﾍﾞｸﾀ 参照) / 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･ﾌｪｯﾁを除くすべてのﾘｰﾄﾞ/ﾗｲﾄ) [E=mask] ↓ （検索ｽﾃｰﾀｽ） ↓ （外部検索ﾃﾞｰﾀ） （各オペランドは_で区切って入力する）
ｵﾌﾞｼﾞｪｸﾄ・ﾍﾞﾘﾌｧｲ	ＶＲＹ　□	なし	TTY1(ﾁｬﾈﾙ1またはﾁｬﾈﾙ4) / TTY2(ﾁｬﾈﾙ2) ﾁｬﾈﾙ1(ｼﾘｱﾙ･ｲﾝﾀﾌｪｰｽ)またはﾁｬﾈﾙ4(ﾊﾟﾗﾚﾙ･ｲﾝﾀﾌｪｰｽ)の設定はIE起動時に行います。
	ＶＲＹ　☆	なし	file（入力ﾌｧｲﾙ名）

（１３）　エラー・メッセージ一覧

エラー・メッセージ一覧を示します。

（１）　aborted

失敗しました。

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。

（２）　Append symbol file not found

追加シンボル・ファイルがありません。

ＳＹＭ＿Ｌコマンドで、アペンド･シンボル･ファイルがカレント･ディスク上に存在しなかった。

（３）　append symbol table full

テーブルに空きがありません。

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。

（４）　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。

（５）　Bad character

不正なキャラクタを検出しました。

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。

（６）　Bad file entry

ファイル名の指定に間違いがあります。

ファイル名の記述が正しくない。

（７）　Can not close ﾌｧｲﾙ名

クローズできません。

表示されたファイルのクローズが正常にできなかった。

（８）　Can not close ﾌｧｲﾙ名,Cancel ×××　command

クローズできません。キャンセル×××コマンド！

×××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）

（９）　Can not execute HLP command ！

ＨＬＰコマンドが使用できません！

カレント･ディスク上にヘルプ･ファイル，ヘルプ･オーバーレイ･ファイルが存在しない。

（１０）Can not open ﾌｧｲﾙ 名

オープンできません。

指定されたファイルがオープンできなかった。

（１１）Can not test

テストできるメモリがありません。

テストできるメモリがない。

（１２）Can not use command abbreviation !

省略形式でのコマンド入力はできません。

カレント･ディスク上に省略形のオーバーレイ･ファイルが存在しない。

（１３）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。

（１４）Check sum error

チェック・サム・エラー検出

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。

（１５）Command/Data too long

コマンド／データ入力文字数オーバ

１２８文字以上のコマンド、あるいはデータ行が入力された

（１６）Command format error

コマンド形式に間違いがあります。

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 。

（１７）Communication error

通信異常

ＩＥ－７８１３０－Ｒとホスト･マシンの通信が正常にできなかった 。

（１８）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。

（１９）Disk read error ﾌｧｲﾙ名

ディスク読みだしエラーを検出しました。

表示されたファイルの読み込みで異常を見つけた。

（２０）Disk read error ﾌｧｲﾙ名.Cancel STR command

ディスク読みだしエラーを検出しました。キャンセルＳＴＲコマンド

ＳＴＲコマンド実行中、表示されたファイルの読み込みで異常を見つけた。

（２１）Disk write error ﾌｧｲﾙ 名

ディスク書き込みエラーを検出しました。

表示されたファイルの書き込みで異常を見つけた。

（２２）Disk write error ﾌｧｲﾙ名.Cancel ×××　command

ディスク書き込みエラーを検出しました。キャンセル×××コマンド

×××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）

（２３）double define append symbol

既に定義されたシンボルがあります。

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。

（２４）double define append symbol ｼﾝﾎﾞﾙ名

既に定義されたシンボル、シンボル名があります。

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。）

（２５）double define loaded symbol

既に定義されたシンボルをロードしました。

ＬＯＤ、ＳＹＭ＿Ａ、ＳＹＭ＿Ｌコマンドで、すでに登録されているシンボルがロードされた。

（２６）double define module name ﾓｼﾞｭｰﾙ 名

既に登録されているモジュールです。

表示されたモジュール名は、すでにロードされている。

（２７）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。

（２８）File already exists.

同じ名前のファイルが存在します。

ファイルの属性が SYSあるいは R/Oのファイルに対し、同一名のファイルを新たにメイクしようとした。

（２９）File make error ﾌｧｲﾙ名

ファイルが作成できません。

表示されたファイルを作成できなかった。

（３０）File name is used by other process

指定したファイル名は他のコマンドで使われています。

すでにオープン済みのファイル名を指定した。

（３１）file not found

ファイルがありません。

指定されたファイル名が存在しない。

（３２）File overflow

ロードできるファイル数をオーバしました。

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。

（３３）Illegal append symbol file

追加シンボル・ファイルの形式が違います。

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。

（３４）Illegal record

異常レコード

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。

（３５）Input data error

入力データに間違いがあります。

入力したデータが正しくない。

（３６）Keyword Error

キーワードに間違いがあります。

ＨＬＰコマンドで、コマンド･キーワードが正しくない。

（３７）List device is used by other process

プリンタが使用できません。

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、あるいはIE-78130-R以外の処理がﾘｽﾄ 装置を使用している場合。）

（３８）load failed

ロード異常

ＬＯＤコマンドで、シンボル、あるいはオブジェクトのロード中にエラーを検出した。

（３９）Mapping error

マッピングされていない範囲があります。

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。

（４０）module buffer full

使用できるモジュール数を超過しました。

ＬＯＤコマンドで、入力できるモジュール数をオーバした。

（４１）Module not found

指定モジュールがありません。

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。

（４２）module overflow

登録できるモジュール数をオーバしました。

ＬＯＤコマンドで、入力できるモジュール数をオーバした。

（４３）Multi define

複数定義

ＰＧＭのカレント制御キャラクタ変更時に、同一キャラクタを設定した。

（４４）No appended symbol

アペンド・シンボルがありません。

ＳＹＭ＿Ｓコマンドで、アペンド･シンボルは存在しない。

（４５）No .HLP file on the default drive

ヘルプ・ファイルがありません。

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，ヘルプ･オーバーレイ･ファイルが見つからなかった。

（４６）No symbol

シンボルがありません。

シンボルがない。

（４７）Non map area access

マッピングされていないエリアをアクセスしました。

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。

（４８）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。

（４９）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。

（５０）not found module record

モジュール名レコードがありません。

ＬＯＤコマンドで指定されたモジュール名レコードがシンボル･ファイル内に存在しない。

（５１）Reserved file name

使用できないファイル名です。

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。

（５２）reserved word symbol

予約語です。

ＳＹＭ＿Ａコマンドで、予約語がシンボルとして定義された。

（５３）Slave CPU communication error

チャネル２のスレーブＣＰＵ(8742)に対し、コマンドが書き込めない。

（５４）Symbol not found

指定したシンボルがありません。

ＳＹＭ＿Ｃ、ＳＹＭ＿Ｅコマンドで、指定されたシンボルは存在しない。

（５５）Symbol record format error

シンボルレコード形式が間違っています。

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。

（５６）symbol table full

シンボル・テーブルに空きがありません。

ＬＯＤコマンドで、シンボル･セーブ･エリアに空がない。

（５７）System mode command

システム・モード専用コマンドです。

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。

（５８）Unexecutable command

このモードでは、実行できません。

エミュレーション中に実行できないコマンドを入力した、あるいはエミュレーション中にしか実行できないコマンドをブレーク中に入力した。

（５９）Unrecognized command

存在しないコマンドです。

入力したコマンドは存在しない。

（６０）Warning!

オブジェクトの生成はできるが、正しい動作は望めない。

（６１）Warning double define : ﾓｼﾞｭｰﾙ 名

複数指定

ＬＯＤコマンドで、同一モジュール名が複数回指定された。

保守／廃止

NEC 日本電気株式会社

本社	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	
半導体 第一、第二 販売事業部	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	東京(03)3454-1111
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(日本電気関西ビル)	大阪(06)945-3178 大阪(06)945-3200
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

北海道支社	札幌(011)231-0161	立川支社	立川(0425)26-0911
東北支社	仙台(022)261-5511	千葉支社	千葉(0472)27-5441
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(0534)52-2711
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)221-8511
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(0862)25-4455
長野支店	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

(技術お問い合せ先)

半導体応用技術本部 第一応用システム技術部	〒108-01 東京都港区芝五丁目7番1号(日本電気本社ビル)	東京(03)3798-6105
半導体応用技術本部 第二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(日本電気関西ビル)	大阪(06)945-3383
半導体応用技術本部 専用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)246-3922

インフォメーションセンター
FAX(044)548-7900
(24時間受付)