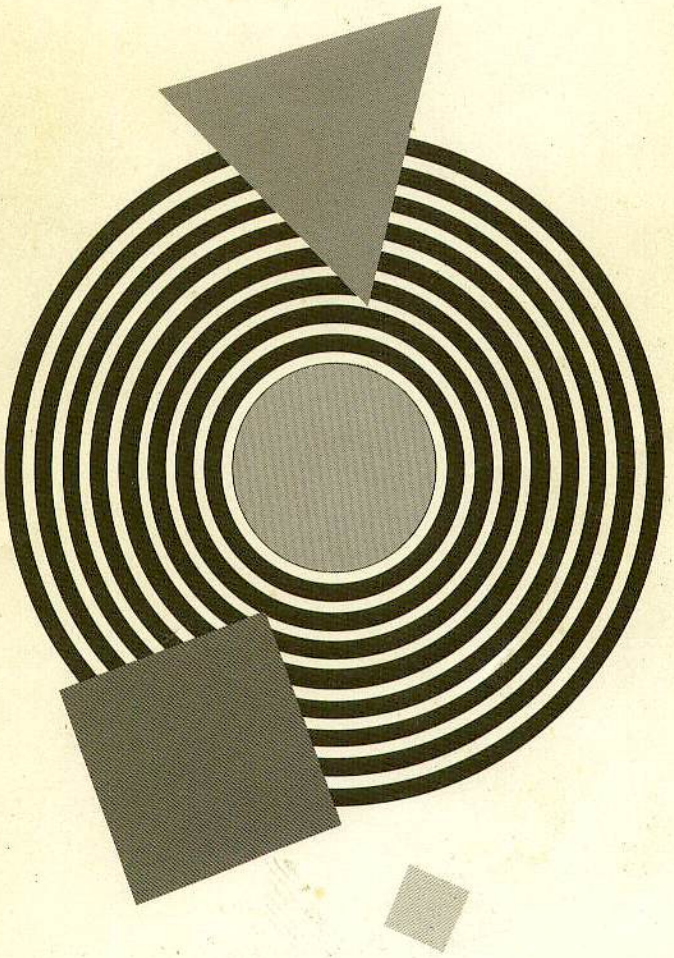
MSX 2 DOS2

日本語MSX-DOS2専用

MSX-DOS2 TOOLS

エムエスエックス ドス2 ツールズ

ユーザーズマニュアル



ASCII

はじめに

MSX-DOS2 TOOLS は，階層ディレクトリ，RAM ディスク，漢字等をサポートした日本語 MSX-DOS2 の機能を活かすための，25 のツールソフトウェア，スクリーンエディタ，アセンブラ，リンカなどのユーティリティソフトウェアをパッケージした強力な開発ツールパッケージです．

日本語 MSX-DOS2 がさらに使いやすくなる上に，生産性を高め，アセンブリ言語によるアプリケーションプログラムの開発環境を大幅に向上させるソフトウェアです．

本パッケージには以下のものが含まれています．

■ MSX-DOS2 TOOLS　システムディスク　　1 枚
（3.5-1DD フロッピーディスク）

■ MSX-DOS2 TOOLS　ユーザーズマニュアル 1 冊

ご注意

(1)このソフトウェアならびにマニュアルを賃貸業に使用することを禁じます．また，このソフトウェアやマニュアルの一部または全部を無断でコピーすることはできません．

(2)このソフトウェアは，個人使用以外の目的でコピーすることはできません．

(3)このマニュアルに記載されている事柄は，将来予告なく変更することがありますが，当社に登録されている方にはご案内をお送りします．

(4)製品の内容については万全を期しておりますが，製品の内容についてのご不審や，誤り，マニュアルの記載もれなど，お気付きのことがございましたら，マニュアルの巻末の「お問い合わせ」についての要領で下記の問い合わせ先へお送り下さい．

(5)このソフトウェアを運用した結果の影響については，(4)項にかかわらず，責任を負いかねますのでご了承下さい．

お問い合わせ先　〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル
株式会社アスキー システム事業部 MSX係

目 次

第３部　ユーティリティ　93

MSX-DOS2 TOOLS

第1章　MSX-DOS2 TOOL コマンドの概要

MSX DOS2-TOOL コマンド(以下 DOS2-TOOL コマンド)は MSX DOS-TOOLS(以下 DOS1-TOOLS)の 28 本のツールコマンド中から，日本語 MSX-DOS2(以下 DOS2)の COMMAND2.COM や DOS2 パッケージ付属のユーティリティコマンドでサポートされていないものを選び，パッケージしたものです．

ここでは，DOS2-TOOL コマンドで拡張された機能などについて，パッケージ全体を見渡しての解説をおこないます．

第2章　DOS1-TOOLSからのバージョンアップ

2.1　ファイル指定におけるパス名のサポート

DOS2の最大の魅力のひとつとして，UNIXやMS-DOSライクな階層化ディレクトリがサポートされた点があげられます．これによりファイル管理の効率が飛躍的にアップしたわけですが，DOS1-TOOLSの各コマンドでは階層化されたディレクトリに対応しておらず，カレントディレクトリ(現在自分のいるディレクトリ)のファイルしか扱うことができませんでした．これに対してDOS2-TOOLコマンドでは，ファイル名の指定にパス名を用いることができますので，いちいちオペレーションの対象となるファイルのあるディレクトリに入ることなくファイルにアクセスすることができます．

パスや階層ディレクトリの詳しい説明はDOS2のマニュアルをご参照下さい．

2.2　パイプ・リダイレクトとツールコマンド

DOS2では，OSレベルでパイプ・リダイレクションの機能をサポートしています．“パイプやリダイレクションならDOS1-TOOLSでも使えた機能では?”，と思われるかもしれませんがDOS1-TOOLコマンドでは，それぞれのコマンドが入出力時に疑似的にパイプ・リダイレクションをおこなっていたものです．言い換えれば，DOS1-TOOLSの各コマンドは実際の処理をおこなう部分以外にパイプなどの入出力の機能を持っていたのです．このためプログラムサイズがたいへん大きくなっていたわけです．

これに対してDOS2-TOOLコマンドでは，DOS2がOSレベルでパイプ・リダイレクションをサポートしたのをうけて，コマンドの入出力の部分はDOS2の標準入出力をそのまま用いることで，パイプ・リダイレクション機能をサポートしているのです．このため，各コマンドのサイズはたいへん小さくなっています．

第3章　DOS2-TOOLコマンドの注意点

DOS2ではシステムディスク付属のユーティリティコマンドが用意されており，その中のいくつかはDOS1-TOOLコマンドと重複しています．したがって，DOS2-TOOLSのTOOLコマンドはその重複したコマンドを除き，別の新たなコマンドを追加しています．

DOS2-TOOLSで削除されたコマンドは，

CHKDSK，CLS，DISKCOPY，ECHO，HELP

の5本で，追加されたコマンドは，

ADDAUX，SPEED

です．追加されたコマンドについては，第5章　コマンドリファレンスをご参照下さい．

3.1　DOS1-TOOLコマンドとの相違点

DOS2-TOOLコマンドとDOS1-TOOLコマンドでは多少仕様の異なる点があります．また，いくつかの機能はDOS2-TOOLコマンドでは用いることができなくなっています．

ここでは，DOS2-TOOLコマンドとDOS1-TOOLコマンドとの仕様の違いについて記述します．

3.1.1　パイプ・リダイレクションの仕様の違い

前述したようにファイル入出力の部分をDOS2に依存していますから，パイプ・リダイレクションなどについての詳しい仕様は，DOS2のマニュアルをご覧ください．

主な違いとしては中間ファイルを作成するディレクトリの指定方法が環境変数を用いておこなうようになったことなどがありますが，基本的にはDOS2のマニュアルを参照しなくともDOS1-TOOLコマンドと同様に用いることができるでしょう．

3.1.2　シーケンシャルプロセスの非サポート

DOS1-TOOLコマンドではコマンドを“；”でつなぐことにより複数のコマンドの実行を1行の入力でおこなうことができましたが，DOS2-TOOLコマンドではこの機能はサポートされていません．

3.1.3　ファイルスペックの指定

DOS1-TOOL コマンドではファイルの間接指定やファイルスペックを“+”でつなぐことにより複数指定することができましたが，この仕様は DOS2-TOOL コマンドでもそのままサポートしています．詳細は，3.3 コマンドの書式指定をご参照下さい．

注意が必要なのは，DOS2 コマンドや DOS2 のマスターディスクに含まれるユーティリティコマンドでサポートされる複合ファイルスペック(これも“+”を用いる)とは異なるということです．複合ファイルスペックについての詳細は DOS2 のマニュアルをご参照下さい．

3.2　各コマンドの仕様の相違点

基本的に DOS1-TOOL コマンドの仕様はまったく変わっていません．ただし，LS，PATCH，WC などいくつかのコマンドにおいてはファイル名の表示の際に画面レイアウトの都合上，パス名の先頭の何文字かが省略される場合があります．

例えば，

H：¥DOS2¥TOOLS¥KEY.COM　→　¥TOOLS¥KEY.COM

のように先頭のドライブ名及びサブディレクトリ名が省略されて表示される場合があります．

3.3　コマンドの書式指定

3.3.1　ファイルスペック

書式中で“ファイルスペック”となっているパラメータはドライブ名，ファイル名，ピリオド，拡張子名が正しい順序で並んだものを示し，ファイル名，拡張子名はワイルドカードキャラクタ(*，?)を含むことができます．

さらに上記の指定を“＋”または“,”でつなぐことにより複数指定可能です．(間のブランクは無視されます)

ドライブ名は常に省略可能であり，デフォルトはカレントドライブとなります．

また，ファイルスペックのかわりにファイルの間接指定(3.3.4 ファイルの間接指定を参照)を用いることが可能です．

例　b：*.bat ＋ *.c ＋ a：stdio.h

3.3.2　ファイル名

“ファイル名”で示されるパラメータはワイルドカードキャラクタを含むことができないという点，複数の指定がゆるされないという点でファイルスペックと区別されます．

その他については同様の指定となります．

3.3.3　ドライブ名

“ドライブ名”で示されるパラメータは接続されているドライブのA～Hまでの各ドライブ名にコロン“：”をつけたもので示します．

3.3.4　ファイルの間接指定

ファイルの間接指定とは，複数のファイルを同時に指定可能なコマンドについて，前述の“ファイルスペック”を用いず，必要なファイル名をまとめてひとつのファイルにして，コマンド行ではその“ファイル名のファイル”を間接的に指定することです．

ファイルの間接指定により複数のファイル指定を簡単なコマンド行の入力で行うことができます．

A）間接指定の方法

通常のファイルスペックを指定するかわりに，下記のフォーマットにしたがったファイル名の

リストファイルを“[ ]”で囲んで指定します．コマンド側は“[ ]”で囲まれたファイルを間接指定用のリストファイルとしてオープンしその内容を通常のファイル指定と同様に処理します．

間接指定の“[ ]”のなかでワイルドカードを用いたり複数のリストファイルを指定することはできません．また，間接指定を用いた場合はそれのみが有効であり，同時にファイルスペックやファイル名の指定をしても無効となります．

B) ファイル名のリストファイル

ファイルの間接指定のためのファイル名のリストファイルは以下のフォーマットで作ります．

●必要なファイル名を1行にひとつだけ指定します．指定できるファイル数に制限はありません．
●ファイル名にワイルドカードは用いられません．
●各行頭のスペース，タブは無視され最初のキャラクタからファイル名とみなされます．

C) 間接指定の用例

間接指定の用例としてディスク上の拡張子“.bat”のついたすべてのファイルをファイルサイズの更新日の新しい順にそのヘッダー部分を見ながら必要なら，別のディスクにコピーするという操作を以下に解説します．

(1) LS コマンドのリダイレクトによりリストファイルをつくる．

例　ls /t/c *.bat > b：bat.lst

LS コマンドの/T,/C オプションによりカレントドライブのディスク上の拡張子“.bat”をもったすべてのファイルが最新更新日の新しい順にソート(/T オプションによる)され，一行に1ファイル名のリストが(/C による)B ドライブに bat.lst という名でリダイレクトされます．

(2) bat.lst をファイル間接指定のリストファイルに用いて MENU コマンドにより必要なファイルをコピーする．

例　menu　[b：bat.lst]

これにより拡張子“.bat”をもったファイルが更新日の新しい順にプロンプト表示され，以下 MENU コマンドの T コマンドによりそのヘッダー部分を見ながら，必要なら C コマンドでコピーすることができます．

3.3.5　西暦指定

西暦年(yyyy)の入力は，4桁のうち上2桁が省略可能でありデフォルトは20世紀となります．

例　00～99 ・・・・ 1900～1999年

3.3.6　オプションスイッチ

“/”にアルファベット1文字をつけた(“/S100”のようにパラメータを伴うこともある)アーギュメントは，コマンドの各種のオプション機能を実行するためのスイッチを意味します．アルファベットは大文字でも小文字でも有効であり，また複数のオプションスイッチを指定する場合，スペース，タブ等で区切らずに続けて指定可能です．

例　1．GREP/X/N/A ・・・
　　2．grep/x /n /a ・・・
　　1.，2．どちらも有効です．

3.3.7　アーギュメントの区切り

ツールコマンドでは，書式中“ ”で表されるアーギュメントの区切りにはスペースまたはタブのみが有効です．カンマを用いることはできません．

ただし，“/X”のようなオプションスイッチを複数指定する場合，各オプション間のスペースは省略できます．

例　grep/x/n/a "---" < xxx.c

第４章　パッケージの内容

本パッケージにはDOS2用に機能拡張された25本のツールソフトウエアが含まれます．また，DOS1-TOOLコマンドに付属のアセンブラ(MSX･M-80)やリンカ(MSX･L-80)，エディタ等のユーティリティソフトウエアもDOS2対応になっています．これらのソフトウェアについては，第2部以降をご参照下さい．

以下に本パッケージに含まれるツールソフトウエアの一覧を示します．

ADDAUX	AUXデバイスのインストール	19
BEEP	BEEP音の発生	21
BIO	バイオリズムの出力	22
BODY	ファイルの一部の切り出し	23
BSAVE	HEXファイルのバイナリファイルへの変換	24
CAL	カレンダーの表示	25
CALC	簡易電卓	26
DUMP	ファイルの16進ダンプ	27
EXPAND	スペース・タブの変換	29
GREP	任意の文字列の検索	30
HEAD	ファイルの先頭部分の表示	31
KEY	ファンクションキーの設定	32
LIST	BASICの中間言語ファイルのソースファイルへの変換	33
LS	ディレクトリの詳細情報の表示	34
MENU	ディレクトリのファイルの諸操作	36
MORE	ファイルの表示	38
PATCH	バイナリファイルの更新	39
SLEEP	アラーム機能	41
SORT	ソート(クイックソート)	42
SPEED	AUXデバイスの通信パラメータの設定	43
TAIL	ファイルの末尾部分の表示	45
TR	文字列の置き換え	46
UNIQ	重複行の削除	48
VIEW	ファイルの表示(スクロール)	49
WC	語数，行数，ページ数カウント	50

以上25本

第5章　コマンドリファレンス

ADDAUX

機能　DOS2 の AUX デバイスを RS-232C 入出力用として割り当てます.

書式　ADDAUX

解説　DOS2 の AUX デバイスを RS-232C 入出力用として割り当てます. ADDAUX を実行することにより，それ以降の COMMAND2.COM の内部コマンド(COPY など)や DOS2 のシステムコールレベルで AUX デバイスを使用できるようになります.

ADDAUX を実行すると，

```
ADDAUX version 1.0
Copyright 1989 ASCII Corporation
RS232C driver is installed.
```

と表示し，ドライバを組み込み，それ以降の AUX に対する入出力は RS-232C に対して行なわれます.

この時の RS-232C の通信パラメータは

通信速度	1200 ビット/秒
キャラクタ長	8 ビット
パリティ	なし
ストップビット	1 ビット
XON/XOFF 制御	あり
CTS-RTS ハンドシェーク	なし
CR 受信時の LF の付加	しない
CR 送信後の LF の削除	しない
SI/SO 制御	なし
送信時タイムアウト	なし

に設定されます.

RS-232C がない場合は

```
ADDAUX version 1.0
Copyright 1989 ASCII Corporation
Cannot find RS232C
```

と表示し，組み込みを中止します．

また，AUX がすでに使用されている場合は

```
ADDAUX version 1.0
Copyright 1989 ASCII Corporation
Device driver is already installed.
```

と表示し，組み込みを中止します．

なお，ADDAUX は TPA(トランジェント・プログラム・エリア――ユーザーエリア)に常駐し，TPA 領域を 359 バイト使用します．そのため，ユーザーエリアや BASIC のフリーエリアが 359 バイト少なくなります．また，この時の TPA の最終アドレスが C800H より小さい場合は

```
ADDAUX version 1.0
Copyright 1989 ASCII Corporation
No enough memory.
```

と表示し，組み込みを中止します．この場合は，接続されたディスク装置の数を減らす(インターフェイスを外す，CTRL キーを押しながら立ち上げる)などの対策を行って下さい．

RS-232C を利用するアプリケーション(ターミナルソフトなど)をご使用になるときは，一度システムを立ち上げ直して ADDAUX を組み込んでいない状態でご使用下さい．

ADDAUX を組み込んだ状態ではスタック領域が 8000H 以上にないと，RS-232C が呼び出されたとき(データの受信割り込みや AUX としての呼び出し)に，スタック領域が RS-232C の ROM となってしまうため，不都合が起きます．AUX を使用したアプリケーションプログラムを作成する際はご注意下さい．

例　A>ADDAUX ↵

BEEP

機能　ビープ音を発生します．

書式　BEEP ［ビープの回数］

解説　ビープ音を発生します．

バッチ処理中などで用いることにより，処理の進行の目安とすることができます．また，BEEP コマンドでは，リダイレクトによりファイルに BEL コード(07H)を送ることも可能です．

アーギュメントとして，発生するビープ音の回数を指定することができます．

例

```
A>beep 2
    Beep!
    Beep!
A>
```

BIO

機能　バイオリズムを出力します．

書式　BIO ＜yyyy/mm/dd＞　［＜［[yyyy/］mm/］dd＞］

解説　バイオリズムを出力します．出力は標準出力が用いられますから，通常はコンソールに表示されますが，リダイレクトによりプリンタに印刷することも可能です．

生年月日を指定すると，当日と前後20日間のバイオリズムが出力されます．生年月日の指定は西暦年，月，日の順にスラッシュで区切り指定します．

オプションとしてバイオリズムを調べたい日を指定できます．この場合も，年月日を指定すると，指定の日と前後20日間のバイオリズムが出力されます．このオプションを指定しない場合のデフォルトは当年，当月です．日付省略は可能です．

指定可能な日付の範囲は1901年1月1日からＤＯＳのＤＡＴＥコマンドで設定している日付(ほとんどの場合は当日)までです．ＢＩＯコマンドが表示可能なバイオリズムの範囲は1901年1月1日から2070年12月31日までです．

例　A>bio 1962/8/7 >prn↵

BODY

機能　入力の一部を切り取って出力します．

書式　BODY　[/H] [/N] [/S<開始行>] [/E<終了行>]　[<ファイルスペック>]

解説　入出力は通常標準入出力が用いられますが，入力については<ファイルスペック>により，1つまたは複数のファイルの指定が可能です．

ファイルスペックで'*'や'?'などのワイルドカードキャラクタを指定した場合，通常不可視属性をもったファイルはマッチングの対象となりません．

出力には入力の先頭行からの行番号，ファイル名がつきます．

各行は通常，コンソール画面幅を超えた部分については出力されませんが，出力がファイルその他のデバイス(プリンタなど)にリダイレクトされている場合は行の途中でカットされることはありません．これにより行番号のついたファイルやプリントアウトを得ることができます．

/S　入力の切り出し開始を<開始行>で指定します(デフォルトは先頭行)．

/E　切り出し終了行を<終了行>で指定します(デフォルトは末尾行)．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．

/N　行番号，ファイル名等の出力が省かれ，入力行のみの出力となります．出力行が途中でカットされることはありません．

例

```
A>body ¥help¥command2.hlp >h:temp↵
A>type h:temp↵
     A:¥HELP¥COMMAND2.HLP:
    1: COMMAND2
    2:
    3: [機能]  コマンドインタプリタを起動する．
    4:
    5: [書式]  COMMAND2 [コマンド]
    6:
    7: [解説]  'コマンド'は通常プロンプトで入力できる任意のコマンドで
    8:         す．
    9:
   10:         COMMAND2はディスク上のコマンドインタプリタの名前であり，
                                  •
                                  •
                                  •
```

BSAVE

機能　HEX 形式のファイルをバイナリファイルに変換します．

書式　BSAVE ＜入力ファイル＞ ［＜出力ファイル＞］

解説　アセンブラ，リンカ等により生成したインテル HEX 形式のファイルを入力することにより，BASIC の “BLOAD” 命令でロード，実行が可能なバイナリファイルを生成します．これにより，従来の煩雑なファイル操作なしに，マシン語のルーチンが BASIC の環境下で実行可能となります．

入力ファイルは必ず HEX 形式のファイル 1 つだけ指定して下さい．HEX ファイルは MSX･L-80 の “/X” オプションにより生成することができます．MSX･L-80 でリンクする場合，以下のような方法で行います．

```
A>L80 /P：xxxx,　rel ファイル名,作成するファイル名/N/X/E：START
```

xxxx：プログラムを置くアドレス

START：スタートしたいグローバルラベル名

出力は標準出力ですが，特定のファイル名を指定することができます．この場合，標準出力へは出力されませんので，さらにリダイレクトなどをすることはできません．

例

```
A>bsave b¥usr¥work¥input.hex >b:¥usr¥bin¥out.bin
Complete
A>
```

CAL

機能　カレンダーを出力します．

書式　CAL　[/Y[<yyyy>]　|　[< [yyyy>/] <mm>

解説　出力は標準出力が用いられ，通常画面表示となります．以下オプションの指定とその機能について解説します．

月を指定することにより，指定の月とその前後１ケ月のカレンダーを出力します．月の指定は＜yyyy/mm＞で年と月をスラッシュで区切って行います．各々のデフォルトは当年当月で省略可能ですが，月のみの省略は出来ません．

/Y　年を指定した場合，<yyyy>で指定した年の１年分のカレンダーを出力します．年の指定は上２桁を省略出来ます．年の指定を省略した場合は，当年の１年分のカレンダーを出力します．

例　A>CAL 12↵

```
      Nov 1988                Dec 1988                Jan 1989
Su M Tu W Th F Sa  Su M Tu W Th F Sa  Su M Tu W Th F Sa
      1  2  3  4  5                  1  2  3   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   4  5  6  7  8  9 10   8  9 10 11 12 13 14
13 14 15 16 17 18 19  11 12 13 14 15 16 17  15 16 17 18 19 20 21
20 21 22 23 24 25 26  18 19 20 21 22 23 24  22 23 24 25 26 27 28
27 28 29 30           25 26 27 28 29 30 31  29 30 31
```

CALC

機能　簡易電卓機能です.

書式　CALC ［<式>］

解説　四則演算を通常の演算書式による計算式の入力により行うことができます.
CALC コマンドでは次のようなふたつの演算式の入力方法をサポートしています.

(1)コマンド行で式を指定しない場合(連続計算可)
コマンドラインで“CALC”と入力しコマンドを起動したのち，計算したい数式を入力しリターンキーを押すと改行し，計算結果を表示するとともに，次の計算式の入力まちの状態となり，以下<CTRL-Z>,<CTRL-C>，“Q”または“q”によりコマンドをぬけるまで連続して計算を行うことができます.
(2)コマンド行で式を指定する場合
コマンド行で計算式を入力することも可能です．ただし，この場合は1回の計算でコマンドを終了します.

CALC コマンドでは，十進十二桁の精度での浮動小数点演算をサポートし，
-9.99999999999E＋99～9.999999999999E＋99.の範囲で結果を表示します.

例
```
A>CALC 12+1222.2+345
  1579.2
A>
```

DUMP

機能　ファイルを16進及びキャラクタによりダンプします.

書式　DUMP [/H] [/N] [/B<1行のバイト数>] [/S<開始バイト>] [/E<終了バイト>] [<ファイルスペック>]

解説　入出力には通常標準入出力が用いられますが，入力には<ファイルスペック>による1つまたは複数のファイルが指定可能です.

ファイルスペックで'*'や'?'などのワイルトカードキャラクタを指定した場合，通常不可視属性をもったファイルはマッチングの対象となりません.

ダンプ出力には16進数とキャラクタが用いられます.

出力は十分な画面幅があるかぎり16バイト毎に改行し,標準出力にたいして行われます. 画面幅が16バイトの表示に足りない場合，1行に表示可能な最大数バイトが出力されます. ただし，標準出力がリダイレクトされている場合，"/B"スイッチにより行毎のバイト数をユーザーが指定しない限り，画面幅に関係なく16バイト毎に改行されます.

/S　ダンプの開始バイトを16進数で指定します(デフォルトはファイルの先頭).

/E　ダンプの終了バイトを16進数で指定します(デフォルトはファイルの末尾).

/H　不可視ファイルもワイルドカードのマッチングの対象とします.

/N　ファイルの先頭からのオフセット番地及びキャラクタダンプを省き，16進ダンプのみを出力します. "/B"オプションとの組み合せで1行のバイト数を変更することにより，例えば1行に1バイトずつ出力し，リダイレクトによりファイルにするなどの操作が可能です.

漢字モードでコマンドが起動された際には，キャラクタダンプを表示する時に，それが2バイト文字に対応する文字コードだった場合，画面上で正しく表示されるように出力が調整されることがあります. したがって，その場合には行のはじめのバイトのキャラクタ表示が前の行に付加されます.

例

```
A>dump ¥help¥xcopy.hlp
A:HELP¥XCOPY.HLP:
0000: 58 43 4F 50 59 0D 0A 0D 0A 5B 8B 40 94 5C 5D 20 XCOPY....[機能]
0010: 20 83 66 83 42 83 58 83 4E 82 CC 83 74 83 40 83  ディスクのファイ
0020: 43 83 8B 82 A8 82 E6 82 D1 83 66 83 42 83 8C 83  ルおよびディレク
0030: 4E 83 67 83 8A 82 F0 95 CA 82 CC 83 66 83 42 83  トリを別のディス
0040: 58 83 4E 82 C9 83 52 0D 0A 09 83 73 81 5B 82 B7  クにコ...ピーす
0050: 82 E9 2E 0D 0A 0D 0A 5B 8F 91 8E AE 5D 20 20 58 る.....[書式]  X
0060: 43 4F 50 59 20 81 6D 83 74 83 40 83 43 83 8B 83 COPY [ファイルス
```

EXPAND

機能　スペース/タブの変換をします.

書式　EXPAND　[/H] [/R] [/F]　[<ファイルスペック>]

解説　オプションを省略した時，タブをスペースに変換します.

/H　不可視ファイルもワイルドカードのマッチングの対象とします.
/R　スペースをタブに変換します.
/F　各行頭から最初のキャラクタ(スペース，タブ以外の)間のみで変換を行います.

“/F”により，例えば，プログラム中の文字列データ内のスペース，タブが変換されてしまい，コンパイル後，実行時に予想外の表示がなされるといったトラブルを防ぐことができます.

例　A>expand test.c > prn↵

GREP

機能　入力から任意の文字列を含む行のみを出力します．

書式　GREP　[/H] [/X] [/C] [/N] [/A]　“<文字列>”　[<ファイルスペック>]

解説　入出力は通常標準入出力が用いられますが，入力には<ファイルスペック>による1つまたは複数のファイルが指定可能です．

文字列はダブルクォーテーションで囲み必ずファイルスペックの前で指定します．入力中の検索文字列を含む行を先頭からの行番号をつけて出力します．入力がファイルスペックの場合，ファイル単位でファイル名とともに出力されます．

文字列中で大文字，小文字は区別されません．

文字列には漢字を含むことができます．

GREPコマンドには次のような4つのオプションスイッチがあり，これらを有効に組み合わせることにより，コマンドを効果的に用いることが出来ます．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．
/X　指定した文字列を含まない行を返します．
/C　指定した文字列を含む行数のみを返します．
/N　行番号，ファイル名，文字列を含む行数を省き，行のみを出力します．
/A　大文字，小文字を区別して検索します．

例

```
A>grep /n "書式" ¥help¥c*.hlp↵
[書式] CHDIR [d:] [パス] または CD [d:] [パス]

[書式] CHDIR [d:] [パス] または CD [d:] [パス]

[書式] CHKDSK [d:] [/F]

[書式] CLS

[書式] COMMAND2 [コマンド]
                 •
                 •
                 •
A>
```

HEAD

機能　入力の先頭から指定された行を切り取って出力します．

書式　HEAD　[/H] [/N] [/L＜出力行数＞]　[＜ファイルスペック＞]

解説　入出力は通常，標準入出力が用いられますが，入力については＜ファイルスペック＞で1つまたは複数のファイルが指定可能です．

入力の先頭行から指定の行数を切り取り，行番号，残り行数，入力がファイルスペックである場合，ファイル名をつけて出力します．

各行は通常，コンソール画面幅を超えた部分については出力されませが，出力がファイルその他のデバイス(プリンタなど)にリダイレクトされている場合は，行の途中でカットされることはありません．これにより行番号のついたファイルやプリントアウトを得ることができます．

出力の内容は以下のオプションスイッチにより変更可能です．

/L　先頭から何行出力するかを＜出力行数＞で指定します．指定のない場合，先頭から10行分の出力となります．

/N　スイッチを指定すると，行番号，ファイル名等すべて省かれ，行本体のみの出力となります．この場合の出力は，画面幅と無関係に全て表示されます．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．

例

```
A>head ¥help¥x*.hlp
  A:¥HELP¥XCOPY.HLP:
 1: XCOPY
 2:
 3: [機能]  ディスクのファイルおよびディレクトリを別のディスクにコ
 4:         ピーする．
 5:
 6: [書式]  XCOPY [ファイルスペック [ファイルスペック] ] [オプション]
 7:
 8: [解説]  使用できるオプションを次に示します．
 9:
10:         [/A] [/E] [/H] [/M] [/P] [/S] [/T] [/W]
  A:¥HELP¥XDIR.HLP:
 1: XDIR
 2:
 3: [機能]  ディレクトリ中のすべてのファイルのリストを表示する．
 4:
 5: [書式]  XDIR [ファイルスペック] [/H]
 6:
 7: [解説]  ファイルスペックでリストするファイルを指定します．
 8:         /Hによって隠されたファイルを表示することができます．
 9:
10:         XDIRはDIRコマンドと類似していますが，ファイルの日付と
A>
```

KEY

機能　**ファンクションキーに文字列を設定します.**

書式　KEY　[/I]　[ON | OFF]　[<ファンクションキー番号>, <設定文字列>]　・・・

解説　画面最下行に現在のファンクションキーの設定を表示(キーオンの状態)したり，消去(キーオフの状態）します.

コマンド行で変更したいファンクションキー番号(1〜10)に続いて,設定文字列を入力することにより，ファンクションキーの設定をします．キー番号と設定文字列はカンマで区切ります.

設定文字列中にキャリッジリターン，エスケープなどのコントロールコードを入力したい場合，“¥xx” のように'¥'に続けて2桁の16進数でキャラクタコードを入力します.

設定文字列の長さは15文字まで指定可能です.

KEY の ON/OFF の指定はコマンド行の最初のアーギュメントで行い，省略するとコマンド実行以前の設定となります.

また,文字列中にツールコマンドのアーギュメントの区切として使われる文字(スペース，タブ，‘,’，‘+’，‘<’，‘>’，カンマ，‘／’，‘;’，‘|’)を指定したい場合，“dir /w” などのようにダブルクォーテーションで囲みます.

/I　　ファンクションキーを BASIC の初期設定に再設定します.

例　**A>key off 1,"DIR/W ¥0D"**↵
(ファンクションキー1に DIR/W と改行コードを設定)

LIST

機能　MSX-BASIC の中間言語ファイルをアスキー文字列に変換し出力します.

書式　LIST　<入力ファイル名>

解説　入力には必ずディスク上の BASIC バイナリファイル名を指定して下さい. 出力は標準出力です.

MSX-DOS 上で BASIC のプログラムリストを表示したり, リダイレクションを用いてファイルにすることにより, DOS2 上のエディタ(KID など)をつかって編集することが可能になります.

例

```
A>list TEST.BAS
100 DEFINT A-Z:SCREEN 1,2:WIDTH 32
110 COLOR 15,1,1:KEY OFF:GOSUB 400
                •
                •
                •
800 END
A>
```

LS

機能　**ディレクトリの詳細情報を報告します.**

書式　LS [/H] [/L] [/A] [/B] [/C] [/D] [/S | /T] [/R] [<ファイルスペック> | <ドライブ名>]

解説　アーギュメントを指定しない場合，カレントドライブのカレントディレクトリ下のファイルやサブディレクトリ名をアルファベット順にソートし，スクリーン幅に合わせて改行し標準出力に出力されます.

ファイルスペックやドライブ名を指定した場合には，指定のファイルやサブディレクトリ，あるいは指定のドライブのカレントディレクトリのファイルやサブディレクトリの情報が出力されます.

/L　最新更新日時，ファイルサイズ，ファイル属性の情報を付加し，各ファイルごとに改行して表示します.
/A　1行に1ファイルずつファイル名とファイル属性を表示します.
/B　〃　ファイルサイズを表示します.
/D　〃　ファイルの日付を表示します.
/C　表示の形式を1行に1ファイルにする.
/T　最新更新日の新しい順に表示します.
/S　ファイルサイズの大きい順に表示します.
/R　上記2つのスイッチとそれぞれ組み合わせることにより，各々逆に表示します. 単独の場合，アルファベットの逆順となります.
/H　不可視ファイルもワイルドカードのマッチングの対象とします.

尚，"/T"，"/S"が共に指定された場合，"/T"のみが有効となり日付順に表示されます.

ファイル属性を表すキャラクタの意味と表示形式は以下のとおりです.

d　サブディレクトリを表す.
r　ファイルが読みだし可能であることを示す.
w　ファイルが書き込み，削除可能であることを示す.
s　ファイルがシステム属性を持つことを示す.
a　ファイルのアーカイブビットがONであることを示す.
h　ファイルまたはディレクトリが不可視属性を持つことを示す.

例えば，

```
-rw-ah
```

と表示されたときは，このファイルは読み書き可能な不可視ファイルであり，現在アーカイブビットが立っていることを示します．

例

```
A>ls /h/l/t↵                          (更新/作成の日付の新しい順にソート)
drw-a-        0 bytes  Aug-10-1988 13:49  A:TOOLS
drw-ah        0 bytes  Jul-22-1988 15:05  A:HELP
drw-ah        0 bytes  Jul-22-1988 15:05  A:UTILS
-rw-a-    14976 bytes  Jul-22-1988 14:30  A:COMMAND2.COM
-rw-ah     4480 bytes  Jul-22-1988 14:30  A:MSXDOS2.SYS
-rw-a-       57 bytes  Jul-21-1988 23:01  A:AUTOEXEC.BAT
-rw-a-       57 bytes  Jul-21-1988 23:01  A:REBOOT.BAT
A>
```

MENU

機能　**ディスク上のファイルについて，連続してデリート，リネーム，タイプ，ダンプなどの処理をおこなうことが可能です．**

書式　MENU　[/H] [/L] [/T | /S] [/R]　[<ファイルスペック> | <ドライブ名>]

解説　MENUと入力すると，カレントドライブのディスク上のファイル名がアルファベット順にプロンプト表示されます．下記のサブコマンドにより各ファイルにつきコピー，リネーム等様々な処理を行うことができます．

また，カレントドライブについてのみでなく，ドライブ名やファイルスペックの指定により他のドライブや特定ファイルの指定が可能です．ただし，サブディレクトリの指定はできません．

以下，各コマンド操作について解説します．

ESC，Q	コマンド処理の中止
SPACE，↵	現在表示されているファイルについて何も処理しません．次のファイルについて，処理命令待ちの状態になります．
E	表示中のファイルを消去します．
R	表示中のファイルをリネームします．新しいファイルネームを引き続き入力します．
T	ファイルの先頭から1画面分をタイプアウトします．
D	ファイルの先頭から1画面分のダンプリストを表示します．
C	ファイルを他のファイルにコピーします．“C”を入力後，引き続きコピー先の<ファイル名>を入力します．
BSキー	1つ前のファイルにプロンプトが戻ります．

“T”及び“D”を実行した場合，プロンプトのファイル名は実行前のものが引き続き表示されます．従って，ファイルの内容を確認した後，デリート等の処理を行うことが可能です．

ファイルの表示順序は以下のスイッチにより変更できます．

/T	ファイルの日付の新しい順に表示．
/S	ファイルサイズの大きい順に表示．
/R	“/S”，“/T”と組み合せてそれぞれの逆に表示，単独ではアルファベットの逆順に表示します．
/L	ファイル名の他にファイル属性，日付およびサイズをプロンプトとして表示

します．表示の形式はLSコマンドの場合と同様です．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．

“/T”と“/S”を同時に指定した場合，“/S”は無視され，日付順に表示されます．

例

```
A>menu
  COMMAND2.COM
  MSXDOS2.SYS
  TEST.C:R                ⇦ Rを入力して，リネームする．
  NEWFILE NAME:TEST1.C
                          ⇦ファイル名をTEST1.Cと入力し，リターンキーを押す．
  Complete
  TEST2.C:Q               ⇦ Qで終了．
A>
```

MORE

機能　**標準入力または指定されたファイルを１画面分ずつ区切りながら標準出力します．**

書式　MORE　[/H] [/P<行数>]　[<ファイルスペック>]

解説　標準入力からの内容，または指定のファイルの内容を標準出力(通常はCRTディスプレイ)に１画面分表示します．入力がさらに続く場合，画面最下部に

```
====MORE?====
```

と表示されキー入力待ちの状態となります．

さらに，スペースキーをおすことにより，次の１画面分，リターンキーを押すことにより次の１行が表示され，再度キー入力待ちとなります．

以下，入力(ファイルまたは標準入力)の終りまで同様の操作により表示することが出来ます．

Q　キー入力待ちの状態で'Q'キーを押すことによりコマンドは中止されます．

S　キーにより，現在表示中のファイルから次のファイルにスキップして出力します．入力が標準入力の場合や１ファイルのみ指定の時などはコマンドの終了となります．

/P　１画面の出力行数を任意にかえることが出来ます．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．

例

```
A>more test.doc↵
              TEST.DOC:
これは、moreコマンドのテスト用ファイルです．
                          •
                          •
                          •
                  ==== More ? ====
```

この状態でスペースキーを押すことにより，出力が再開されます．

PATCH

機能　バイナリファイルを更新します．

書式　PATCH ［/S＜ダンプ開始バイト＞］ ＜ファイル名＞

解説　コマンド行の入力により指定のファイルがスクリーンに16進で表示され，パッチモード(ファイル更新モード)となります．カーソルキーでカーソルを動かし，スクリーン上の数値を変更することにより，指定のバイナリファイルを更新することができます．

ダンプ開始バイトは“/S”オプションにより4桁以内の16進数で指定可能です．指定しない場合には，ファイルの先頭からとなります．

さらにパッチモードにおいて，ESCキーを押すことでコマンドモードに入り，ファイルの任意の位置を指定し更新することが可能です．以下，各モードで使用できるキー，コマンドについて解説します．

パッチモード

‘A’～‘F’	‘0～9’の各キー 更新したいバイトの16進表示上にカーソルを移動し，各キーを入力することにより，メモリ上のファイルイメージを変更します．
‘P’，‘N’	ダンプ表示ページを切り替えます．
カーソルキー	カーソルを任意のバイト上に移動します．
SPACEキー	カーソルを右に一つ移動します．
BSキー	カーソルを左に一つ移動します．
ESCキー	‘$’コマンドモードへ移行します．

コマンドモード

P	パッチモードに戻ります．
X	更新したメモリ上のファイルをセーブしてコマンドを終了します．
Q	更新したメモリ上のファイルをセーブせずにコマンドを終了します．

＜ダンプ開始バイト＞の指定

ダンプ開始バイトを16進数4桁以内で指定し，リターンキーを押すことにより指定のバイトからダンプ表示されパッチモードに戻ります．

PATCHコマンドでは上記の各キーの入力をリダイレクトによりファイルから行うことができます．例えば，下記のPATDATA.DATの様なファイルをつくり，これをPATCH起動時に入力リダイレクトすることにより，自動的にファイルが更新されコマンドを終了します．

例　ファイルの100Hバイト目をCDH，101Hバイト目をEFHに変更する．

```
A>body patdat.dat↵ (パッチデータファイルの内容)
      PATDATA.DAT:
    1:$
    2:100
    3:cdef
    4:$
    5:x
A>
A>patch file.bin <patdata.dat↵(ファイルからパッチデータを入力する)
                 •
                 •
                 •
A>
```

SLEEP

機能　システムを一時休止状態にします．

書式　SLEEP　[/A] [/B]　[[<hh>,]<mm>,]<ss>

解説　SLEEP コマンドではシステムの再開までの時間を時(hh)，分(mm)，秒(ss)で指定します．また，オプションによりシステム再開時刻の指定も可能です．

時，分は省略可能ですが，分のみの省略は無効です．また，この場合，時のデフォルトは当時刻となります．

/A　システム再開時刻の指定となります．時，分，両方の省略は無効です．

/B　システムの再開時にビープ音を発生し，アラームとして用いることが出来ます．

SLEEP コマンドは 1/60 秒に 1 の割り込みをカウントすることにより時間を計っています．

例

```
A>sleep  /b 20↵
   •
   •
   •

A>
```

SORT

機能　文字列データをソート(並べ替え)します.

書式　SORT　[/H] [/A] [/R] [/S<桁数>]　[<ファイルスペック>]

解説　入出力は通常標準入出力が用いられますが，入力については<ファイルスペック>で一つまたは複数の入力ファイルを指定できます.

入力されたアスキーコードのデータを各行の１文字目からを対象に判断し，各行を特殊文字，数字，アルファベット，カナの順に並べ替えを行います．アルファベットはアルファベット順にソートされた後，大文字，小文字の順に出力されます.

/R　上記の逆順にソートします.

/S　各行の何桁目からを判断の対象とするかを桁数で指定します．タブは相当数のスペースとして数えます．また，２つのオプションは同時に指定出来ます.

/A　アスキーコード順のソートをします.

/H　不可視ファイルもワイルドカードのマッチングの対象とします.

“/R”と“/A”を同時に指定した場合，アスキーコード順の逆ソートとなります.

例

```
A>sort test.dat
1 2 3
4 5 6
A B C
a b c
B B B
b b b
ｱ ｲ ｳ
ｴ ｵ ｶ
A>
```

SPEED

機能　ADDAUX によりドライバを組み込んだ後に RS-232C の通信パラメータを設定します.

書式　SPEED ＜speed＞ ＜parameter＞ ＜timeout＞

解説　各パラメータの意味は以下の通りです.

＜speed＞　通信速度を指定します. 指定できる速度は以下の通りです.

50, 75, 110, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200

＜parameter＞キャラクタ長などの通信パラメータを文字列で指定します.
各々の文字の意味は次の通りです.

1 文字目　キャラクタ長の指定です.
- 5　5 ビット
- 6　6 ビット
- 7　7 ビット
- 8　8 ビット

2 文字目　パリティビットの指定です.
- N　なし
- E　偶数パリティ
- O　奇数パリティ

3 文字目　ストップビット長の指定です.
- 1　1 ビット
- 2　1.5 ビット
- 3　3 ビット

4 文字目　XON/XOFF 制御の指定です.
- X　制御を行う
- N　制御を行わない

5 文字目　CTS-RTS 制御の指定です.
- H　制御を行う
- N　制御を行わない

6 文字目　CR を受信した時, LF を付加するかどうかの指定です.
- A　LF を付加する
- N　LF を付加しない

7文字目　CRを送信した直後のLFを送信するかどうかの指定です.

A　送信しない

N　送信する

8文字目　SI/SOの制御の指定です.

S　制御を行う

N　制御を行わない

＜timeout＞　XON/XOFF制御やCTS-RTS制御が指定されている時に文字を送信する場合，XOFFを受け取っていたり，CTSがOFFだった場合にはXONを受け取ったり，CTSがONになるまで待ちますが，この待ち時間を数字(秒数)で指定します．0が指定されると無限に待ちます．指定範囲は0から255までです．

なお，＜speed＞だけの指定や＜speed＞と＜parameter＞だけの指定もできます．この場合，省略されたパラメータはそれぞれ

＜parameter＞　8N1XNNNN

＜timeout＞　0

となります．

通信パラメータにエラーがあった場合には以下のように表示し，設定を中止します．

例

```
A>speed abc
  SPEED version 1.0
  Copyright 1989 ASCII Corporation
  Parameter error
  Usage : speed <speed> [<parameter> [<timeout>]]

        <speed>=50,75,110,300,600,1200,1800,2000,2400,3600,4800,7200,9600,19200
        <parameter>=<char><parity><stop><X-control><CTS-RTS>
                        <Receive LF><Send LF><SI/SO>
              <char>          :  5, 6, 7, 8
              <parity>        : N (none), O (odd), E (even)
              <stop>          : 1 (1bit), 2 (1.5bit), 3 (2bit)
              <X-control>     : X (do X-control), N (don't)
              <CTS-RTS>       : H (do CTS-RTS hand-shake), N (don't)
              <Receive LF>    : A (add LF when CR is received), N (no)
              <Send LF>       : A (don't send LF after CR), N (send)
              <SI/SO>         : S (do SI/SO control), N (don't)
        <timeout>=0, 1, ... , 255 (second)

        Default: speed 1200 8N1XNNNN 0
```

TAIL

機能　入力の末尾から指定された行を切り取って出力します．

書式　TAIL　[/H] [/N] [/S＜開始行＞ | /L＜出力行数＞]　[＜ファイルスペック＞]

解説　入出力は通常標準入出力が用いられますが，入力については＜ファイルスペック＞で1つまたは複数のファイルが指定可能です．

入力の末尾から指定の長さを切り取り，行番号，入力がファイルの場合ファイル名をつけて出力します．出力の開始位置は以下のオプションスイッチにより指定します．

/S　出力の開始行を＜開始行＞で指定します．
/L　末尾の何行を指定するかを＜出力行数＞で指定します．
　　以上，どちらも用いない場合，末尾10行の出力となります．
/N　行番号，ファイル名が省かれ，行本体のみの出力となります．
/H　不可視ファイルもワイルドカードのマッチングの対象とします．

“/N”オプションを用いない場合，出力はCRTのスクリーン幅にあわせて，行の終わりがカットされる場合があります．ただし，出力がファイルやプリンタなどのデバイスにリダイレクトされている場合には出力が行の途中でカットされることはありません．

例

```
A>tail b:test.doc
     B:TEST.DOC:
  22: this is line 22
  23: this is line 23
            •
            •
            •
  31: this is line 31
```

TR

機能　**文字，文字列の置き換えを行います．**

書式　**TR　[/H] [/W] [/D] [/S] [/I] [/C] [/N]　“＜文字列１＞”　[“＜文字列２＞”] [＜ファイルスペック＞]**

解説　TRコマンドでは，入出力は標準入出力ですが，＜ファイルスペック＞によりひとつまたは複数のファイルを指定可能です．

また，“/W”スイッチにより，２種類の置き換えが選択可能です．“/W”を指定しない場合，文字単位の置き換えを行います．＜文字列１＞に含まれる文字が入力中に見つかると，その文字に対応する＜文字列２＞の文字に置き換えます．

また，“/W”スイッチを指定しない時にのみ，“/C”スイッチが使用出来ます．“/C”スイッチは＜文字列１＞に含まれない入力中の文字を＜文字列２＞に置き換えます．

“/W”を指定した場合，単語単位での置き換えを行います．＜文字列１＞と完全に一致する文字列が入力中に検出されると，＜文字列２＞に置き換え出力します．

文字列の指定について

- 文字列はダブルクォーテーションで囲みます．
- ＜文字列１＞は省略出来ません．また“/D”スイッチ指定時には＜文字列２＞は指定できません．
- 文字列は255文字以下で指定し途中にヌルキャラクタを含んではいけません．
- 次の三つのキャラクタは文字列中で機能コードとして特別の意味を持ちます．

“-”（ハイフン）　二つの文字または数字を“-”でつなぐことにより文字または数字の範囲の指定となります．ハイフンの両側はANKキャラクタのみ有効です．
例えば，“i-n”と指定すると“ijklmn”の指定と同等となります．

“*”（アスタリスク）　文字*N(Nは数字)で同一文字N個の文字列を意味します．ANKキャラクタに対してのみ有効です．
例えば，“z*5”は“zzzzz”と同等となります．

“¥”(円記号)やタブ，キャリッジリターンなどの特殊コードやハイフン，アスタリスクなどを文字列の要素として指定したい場合“¥”の後に以下のような指定をすることにより可能となります．

¥n	ラインフィード(0AH)．
¥r	キャリッジリターン(0DH)．
¥t	タブコード(09H)．
¥x<16進2桁>	“x”の後に2桁の16進数でコードを指定します．
¥<8進3桁>	3桁の8進数で指定します．例えば，“¥102”は“A”となります．
¥<文字>	円記号の後の文字はそのまま文字列の要素となります． 例えば，“¥¥”のように円記号を文字列に指定可能です．

TRコマンドには以下のようなオプションスイッチが指定可能です．

- /W　単語単位での置換(デフォルトは文字単位)をします．
- /C　<文字列1>に含まれない文字を置換します(“/W”指定時は無効)．“/C”オプションは，スクリーンモードがANKモードのみで有効で，漢字モードで起動した場合には無視されます．
- /N　ファイル名を出力から省きます．
- /D　<文字列1>の文字または語を入力中から削除して出力します．
- /S　同一の置き換えが連続する場合，1回の置き換えにまとめて出力します(“/W”，“/D”のどちらかが指定されている場合，“/S”は無効)．本オプションはANKモード時のみ有効です．
- /I　複数の連続したスペース，タブを1スペースとしてマッチングを行います．本オプションはANKモード時のみ有効です．
- /H　不可視ファイルもワイルドカードのマッチングの対象とします．

例

```
A>TR "a-z" "A-Z" text.c↵

A>TR "printf(" "fprintf(stderr" CTEXT.C >CTEXT1.C↵
```

(置き換えの結果をファイルにリダイレクトする場合)

UNIQ

機能　**入力中の隣接する２行を比較し，それが同じである場合，２行目を出力しません．これにより重複する行を１行にすることができます．**

書式　UNIQ　[/H][/C][/N][〈ファイルスペック〉]

解説　入出力は通常標準入出力が用いられますが，入力については〈ファイルスペック〉により，１つまたは複数のファイルの指定が可能です．

UNIQ の機能は SORT との組み合わせで特に有効となります．SORT の結果を UNIQ フィルタにかける事により，同一行の連続出力を避けることができ，見やすい出力になるでしょう．

/C　行頭にその行の出現回数が表示されます．
/H　不可視ファイルもワイルドカードのマッチングの対象とします．
/N　ファイル名を出力から省きます．

例
```
A>sort <file1 >file2
A>uniq <file2 >list1
```
(以上のオペレーションにより file1 の各行が辞書順にソートされ同一内容の行が重複しないリストが list1 としてディスク上につくられました．)

VIEW

機能　**ファイルの内容を画面表示し，カーソルキー，その他のキーにより画面外も自由に見ることができます．**

書式　VIEW ＜ファイル名＞

解説　VIEWコマンドを実行すると，＜ファイル名＞で指定されたファイルが先頭から1画面分表示されます．以下カーソルキーなどの操作により，画面を上下自由にスクロールすることができ，1画面に収まらないファイルの内容をすべて見ることが可能です．

各キーは以下のような意味を持ちます．

カーソルアップキー　5行上を表示します．
ダウンキー　5行下を表示します．
ライトキー　8桁右を表示します．
レフトキー　8桁左を表示します．

P　前ページを表示します．
N　次ページを表示します．
H　ファイルの先頭を表示します．
T　ファイルの末尾を表示します．

?　VIEWコマンドのサブコマンドのヘルプが表示されコマンドを起動中に参照することができます．
Q，q，＜CTRL-C＞，ESCキー
VIEWを終了します．

例　A>view test.c↵

WC

機能　入力中の語数，行数，ページ数をカウントします．

書式　WC　[/H] [/P<行数>]　[<ファイルスペック>]

解説　入出力には通常標準入出力が用いられますが，入力には<ファイルスペック>により，1つまたは複数のファイルの指定が可能です．WCコマンドでは，文字，数字のブランクを含まない列(ただし，“,”“-”等が間にあっても連続とみなす)を1ワードとします．それ以外のものは無視されます．

全角文字をサポートしていないため，入力中に全角のスペースなどが含まれた場合，正しくカウントされません．

/P　1ページ毎の行数を指定出来ます．デフォルトは60行です．

/H　不可視ファイルもワイルドカードのマッチングの対象とします．

例

```
A>wc *.doc
                Pages     Lines     Words     Characters
A.DOC              12       116       441           1543
B.DOC               1        12        25             85
C.DOC              12       110       500           1834
A>
```

第6章　エラーメッセージ一覧

ツールコマンドではすべてのエラーメッセージはコンソール画面にのみ出力されます.
プリンタやファイルにエラーメッセージが出力されることはありません.

Bad argument(+usage)

アーギュメントの数に誤りがあったり，不要なアーギュメントがコマンドライン中にあった場合このメッセージとともにユーセージを表示しコマンドを終了します.

Bad field specified

BODY，DUMP コマンドで出力の開始行(バイト)の指定が終了行(バイト)より大きい場合このメッセージで処理を終了します.

Bad hexadecimal

16 進数の指定に誤りがあります.

例　tr "¥Ox" "Oa"　(x は 16 進数として不適切なためエラーとなる)

Excessive argument(+usage)

アーギュメントの数が多すぎます.
複数のファイル名を'+'または', 'でつながずに指定したような場合，このメッセージが表示されます.

例　ls *.c *.com *.rel

File cannot be copied onto itself

MENU コマンドの "C"(コピー)コマンドでコピー先のファイル名がコピー元と同一に指定された場合このメッセージを表示します.
コピーは行われずプロンプトのファイル名はそのままで次のコマンド待ちとなります.

<ファイル名> is not a hexfile

BSAVE コマンドで HEX ファイルでないファイルを入力したときにこのメッセージでコマンドを終了します.

Illegal combination of switch

オプションスイッチの指定に無効な組み合せがあります.
その後の処理はコマンドにより異なります.(BODY，MENU，LS の各解説を参照して下さい)

例　ls /s/t

Invalid month specified(+usage)

CALコマンドで1～12の範囲以外で月の指定があった場合のメッセージです.

Invalid character in '/x'(+usage)

オプションスイッチ中に不要なパラメータがあります.
例えば'/H'オプションなどのようにパラメータの不要なオプションにおいて,'/H1'などのように無効な文字や数字が指定されてる場合,ユーセージと共にこのメッセージを表示しコマンドを終了します.

Invalid key number(+usage)

KEYコマンドにおいて1～10以外のファンクションキーナンバーの指定があった場合,このメッセージでコマンドを終了します.
複数のキーの指定のなかでひとつでも誤りがあった場合このエラーとなり処理を中止します.

Invalid minute or second specified(+usage)

SLEEPコマンドで60以上の分または秒の指定があったときにこのエラーとなります.
ただし,時のパラメータ省略時の分,時,及び分のパラメータ省略時の秒の指定には60を超えた値も許されます.

<ファイル名> has insufficient lines(bytes)

BODY, DUMP等で'/S'オプションで指定された出力の開始行(ダンプは開始バイト)が指定のファイルの総行数(バイト数)を超えている場合このメッセージが表示されます.
複数のファイルが指定されている場合コマンドを終了せず次のファイルを出力します.

Invalid time, missing parameters (+usage)

SLEEPコマンドの時刻指定(/A指定時)で時,分のパラメータが共に省略されたときこのエラーとなります.

'x'is not digit (+usage)

数字でパラメータを指定しなければならないオプションスイッチで数字以外を指定した場合,このエラーメッセージが表示されます.

例　HEAD /S12FG

この場合,"'F'is not digit"と表示されます.

Missing parameter (+usage)

HEADコマンドの'/S'オプションのように必ずパラメータが必要なところで指定がないと，このメッセージでエラー終了となります．

例　cal /y

この場合，“Missing parameter in '/Y'” と表示されます．

Not enough memory for buffer

コマンドの実行に十分なメモリを確保できなかった場合にこのメッセージを表示しコマンドを終了します．

＜ファイル名＞ not found

指定のファイルがディレクトリ中に見つかりません．

Rename error

MENUコマンドの'R'(リネーム)コマンドにおいて指定された新しいファイル名が，すでにディスク上に存在するファイルと同一である場合に，リネームエラーとなりプロンプトのファイル名はそのままで次のコマンド入力待ちとなります．

Syntax error (+usage)

KEYコマンドで文字列設定の書式にまちがいがある場合のエラーメッセージです．

例　KEY F1.DIR

この場合，ファンクションキーナンバーと設定文字列の区切りにカンマでなくピリオドが使われているためこのエラーとなります．

Unknown switch (+usage)

無効なオプションスイッチが指定された場合のエラーメッセージです．

例　WC /K

この場合，WCコマンドのオプションに/Kは存在しないためこのエラーとなりコマンドを終了します．

スクリーンエディタ

第1章　MSX-DOS2エディタ KID，AKIDの概略

KID，AKIDはMSX-DOS2上で動作するスクリーンエディタです．

KID，AKIDの主な用途はプログラムファイルの作成ですが，DOS2対応による階層ディレクトリのサポートや，漢字対応により通常の文書作成にも十分に対応できるものです．また，VRAMをデータ領域として用いることにより非常に大きなファイルを編集することができます．

以下に本エディタの特徴を簡単に述べてみます．

1.1　MSX-DOS2への対応

本エディタはMSX-DOS2の階層ディレクトリに対応しています．これにより，編集対象となるファイルの管理を効率的に行うことができます．

1.2　漢字対応

MSX-DOS2や漢字BASICによりMSXの世界でも日本語を扱えるようになってきました．これに対応して，KIDでは日本語ファイルの編集が可能になりました．これにより，プログラム中に日本語のコメントを付けたり，漢字データを扱うプログラムを容易に作成することができるようになりました．

1.3　ユーザー独自のキー設定が可能

本エディタでは，カーソル移動や，文字の削除などエディタの基本的機能のほとんどを，CTRLキーによる操作で行うことができるため，効率的な編集が可能です．

さらに，これらのキーの割り当てをユーザー自身が独自に設定しなおすことができるようになっています．つまり，ほかのマシン上のエディタに慣れ親しんだ方が使い慣れたキー設定でKID，AKIDを使うことができるのです．

1.4　テキストエリアとしてVRAMをサポート

本エディタのもう1つの大きな特徴は，テキストのデータエリアとしてVRAMを用いていることがあげられます．

VRAMにはANKモードで約120Kバイト，漢字モードで約64Kバイトの空き領域がありますが，これらがほぼフルにテキストエリアに割り当てられます．これにより，メインRAMをデータエリアに使うよりずっと多くの容量を確保できます(メインRAMはトータルで64Kバイト，実際にデータエリアに使えるのはせいぜい30Kバイト)．

実際に編集できるファイルの大きさはAKIDで約110Kバイト前後，KIDで約60Kバイト前後ですが，これは通常のプログラムや文書の作成には十分な大きさです．

第2章　エディタって何?

エディタを初めて使う人の中には，ワープロなら使ったことがあるけど，エディタって何のことだろう？　ワープロとどこが違うんだろう？　という人もいるかもしれません．

既に，エディタを使っている人や，使ったことのある人など，エディタがどういうものか知っている人は，「第3章　起動方法」に進んでもらってもかまいません．

この章では，エディタがどういうものなのかワープロと比べてみながら少し説明してみることにします．

ワープロは，もともと文書を作成するために作られたものです．ですから，画面に書いた文書や，表や，グラフを最後には紙に記録する(印刷する)ということを目的としています．また，そのための機能(いろいろな文字体やさまざまなバリエーションの印字等)をたくさん備えています．

では，エディタは何に使うのでしょうか？

エディタはおもにプログラムを作成したり，修正したりするための道具として使います．ですから，ワープロのように表を作ったり，図を書いたり，印刷したりすることは重要ではありません．いってみれば，“プログラム”という文章のみをいかに効率良く，早く，作成したり修正したりして，最後にはファイルに記録するということを，目的としているのです．

普通，エディタで扱われるファイルは“テキストファイル”と呼ばれるファイルです．このファイルは，簡単に言えば，“文字(スペースや改行文字も含めて)が連続的に集まったもの”です．画面に表示する時には，このまま表示したのではわかりにくいので，改行文字のところで改行したり，長い文字列は画面に入るように右端でターンして見やすいように表示されます．

エディタはワープロに比べ機能が絞られているので，一般に操作は簡単で，プログラム開発という目的のために処理は高速です．プログラムを開発するのに便利であるということは，デバッグ作業(プログラムの不具合を修正する作業)に便利であるということです．デバッグ作業では，修正したい箇所を素早く捜しだしたり(検索)，内容を新しい命令や処理に置き換えたり(置換)といった処理が頻繁に行われます．エディタは，これらの機能に優れています．

大きなプログラムでは，1つのファイルで開発を行うと効率が悪くなるので複数のファイルで構成することになります．そこで，あるファイルの一部を別のファイルとしたり(ファイルの分割)，あるファイルに別のファイルの一部を持って来たり(ファイルの統合)といった機能があれば便利です．ほとんどのエディタは，これらの機能を持っています．ワープロでは文章を“ページ”で区切りますが，エディタには“ページ”という考え方はありません．区切りが必要な時は，行番号で区切ります．

以上エディタについてワープロと比較をして簡単に説明してみました．ただ，最近ではエディタのようなワープロや，ワープロのような機能を持ったエディタもでてくるなどして，エディタとワープロの厳密な区別はつきにくくなっています．さらに，エディタの持つ操作性の良さ，処理の早さ等に注目してエディタをワープロ代わりに利用したりする人もでてきています．

第3章　起動方法

3.1　漢字エディタKIDと ANK専用エディタAKID

本パッケージには，

漢字エディタとして　　KID(キッド)
ANK専用エディタとして　AKID(エイキッド)

の2種類が用意されています．

漢字を含む日本語のテキストファイルの編集にはKID，ソースプログラムなど漢字を使用しないテキストファイルの編集にはAKID，という様に使い分けていただきます．もちろん，KIDでもアルファベットやカタカナといったANK文字を編集することは可能ですが，AKIDとKIDでは，AKIDの方が処理速度，扱えるファイルの大きさといった点で漢字処理をしない分，格段優れています．このことを踏まえて，編集するテキストファイルに応じて使い分ければ，効率よく編集作業を進めることができます．

3.1.1　KIDとAKIDの違い

- 編集できるテキストファイルの大きさは，編集するテキストファイルの内容にもよりますが，KIDで最大約60Kバイト，AKIDで最大約110Kバイトです．
- 全ての処理においてAKIDの処理速度の方が優っていますが，とくにスクロールアップ/ダウンなど画面の書き換えに関する処理速度は大きくAKIDが優っています．
- KIDでは漢字などの全角文字の編集が行えます
- 詳しい文字入力操作については「4.3.1 内容更新を行う操作」をご覧下さい．
- 両エディタとも，画面幅40桁と80桁についてサポートしています．
  起動時の画面幅が40桁以内であれば40桁モードで，そうでなければ80桁モードで起動します．また，40桁モードと80桁モードはコマンドメニュー(ファンクションキーを押して表れるメニュー)のレイアウトが違うだけで操作は同一です．

3.2　キーアサイン変更プログラム(CONKID)

本パッケージには，KID, AKID に関連するプログラムとして CONKID というプログラムが用意されています．

CONKID は，エディットモードのキーアサイン，ロールアップ／ロールダウンのスクロール行数，起動時の入力モード(挿入／上書)，検索／置換での大文字・小文字の区別の設定を行い，指定されたエディタファイルに設定内容をセーブすることができます．

デフォルトのキーアサインでは使いにくい，という方はこのアプリケーションで変更し，独自の環境を作り上げて下さい．本アプリケーションのセーブはエディタ本体のキーアサインテーブルに行いますので，設定を変更したエディタは新しいキーアサインの状態で立ち上げることができます．

また，変更したキーアサインは，簡単にデフォルト値に戻せますので，気軽に変更を行うことができます．

CONKID の詳しい使い方は，「第5章　キーアサインの変更」をご覧下さい．

3.3　とりあえず使ってみる(起動から終了まで)

MSX-DOS2 が立ち上がり，画面には，A>■が表示されています．

```
A>■
```

A>AKID ↵ と入れてみましょう．

ディスクドライブがアクセスされ，画面が変わります．

```
File name:■
```

File name: test ↵ と入れてみましょう．

再びディスクドライブがアクセスされ，画面が変わります．test というのは，これから作ろうとするファイルの名前です．入力間違いは，BS キーで消して修正できます．

画面左上にはカーソルが，左下には“New File”と表示されています．New File というのは，test というファイルが新しく編集されることを示しています．何かキーを押すと，New File の文

字は消えます．では，実際に簡単な文字を入力してみましょう．

```
Hello Hello
I am Happy
```

と入力してみて下さい．(もちろん大文字，小文字はCAPSキーで制御できます)
カーソルはIの文字の下にきていますね．

左カーソルキーかCTRL+Sを押し続けてみて下さい．カーソルは最初の文字に戻ります．次に，逆に右カーソルキーかCTRL+Dを押し続けてみて下さい．カーソルは既に入力した文字の上だけを移動することがわかります．このことは，“テキストファイル”が文字の連続した集まりであることを考えるとよく理解できると思います．

今度は文字を削除したり，挿入したりしてみましょう．
カーソルを1行目の先頭に持ってきて，DELキーかCTRL+Gを押して下さい．カーソルのあったところの文字が1文字削除されました．DELキーかCTRL+Gを押し続けてHelloの文字を1つ消してみて下さい．(ちなみにBSキーではカーソルの1つ前の文字が削除されます)

次は2行目のHappyの文字のHの上にカーソルをもってきて，Veryと入力して下さい．
文字が挿入されましたね．
AKIDは立ち上げた状態では，挿入状態になっていますので，カーソルを挿入したい位置に移動して入力すれば文字はカーソルの直前の位置に入力されていきます．
挿入/上書き状態の切り換えは，INSキー又はCTRL+Jで行うことができます．
なお，起動時の挿入/上書き状態は付属の専用プログラムCONKIDによって変更する事ができます．詳しくは，「第5章　キーアサインの変更」をご覧下さい．

それでは，1度終了させてみることにします．
今まで文字を入力したり，削除したりしていた状態がエディットモードです．
F1キーを押してみて下さい．
F1キーのメニューが画面上に表示されました．このようにコマンドメニューの表示された状態がコマンドモードです．F2からF5のファンクションキーにもコマンドが割り当てられています．
エディットモードに戻りたいとき(コマンドを実行したくないとき)はESCキーまたはコマンド以外の文字を入力して下さい．
では，今編集しているtestというファイルをセーブして終了することにします．
Eを入力して下さい．エディタが終了し，画面はMSX-DOS2のプロンプトA>に戻りました．
一応DIRでディレクトリを見てtestというファイルが新しくできていることを確認して下さい．

今度はちょっと違うやり方でAKIDを起動してみましょう.

A>AKID test ↵ と入力します.

このようにすると,1回でエディットモードに入ることができます.画面には,さきほど編集した文字が表示されています.
では,セーブしないで終了する方法です.F1キーを押してメニューが表示されたら,Qを入力して下さい.

Q(Quit)はプログラムの変更,修正でなく,ファイルの内容を確認するときに使うとよいでしょう.

3.4　起動時の注意

KID, AKID とも、起動するときのMSX-DOS2の状態が漢字モードかANKモードかに関係なく使用する事ができますが、そのときの注意事項を以下に記します。

- AKID を使用する際は、ファイル名に漢字が使用されているファイルは使用できません。また、KIDを使用するときでも、起動時の状態がANKモードであるときは同様にファイル名に漢字が使用されているファイルは使用できません。
  したがってファイル名に漢字を使用できるのは、MSX-DOS2の状態が漢字モードでKIDを使用するときだけ、ということになります。

- MSX-DOS2の状態が漢字モードで KID, AKIDを使用する際、単漢字入力モード(CTRL+SPACEキーを押して最下行が反転している状態)のまま起動すると、エディタを終了してMSX-DOS2に戻ったときファンクションキーが表示状態になります。

第４章　操作方法

4.1　コマンドモード

AKIDでは様々なコマンドを簡単に実行させるためにMSXのファンクションキーの1から5を使用しています(下記参照)．これらのファンクションキーを使用して以下に示されるコマンドを実行するわけです．

ファンクションキーを押してから編集状態(エディットモード)に戻るまでの状態をコマンドモードと呼ぶことにします．

SELECTキーはファンクションキーの1から4のメニューを押すごとに，順次表示します．(ファンクションキーの5は他と少し使い方が異なります)

なお，以下に示される各ファンクションのメニューはKIDの80桁モード(mode 80)のときのメニューです．しかし，AKIDの40桁モードのときのメニューでも，KIDのメニューでも使用するコマンドはすべて同一で，入力操作も全く同じです．

1) コマンドを実行するための操作

1.ファンクションキー又はSELECTキーを押してメニューを表示させる．
2.実行したいメニューの頭の文字を入力する．

MSX-WRITEなどの日本語フロントエンドプロセッサ(MSX-JE仕様)を使用して文字入力をしている場合は，1度日本語フロントエンドプロセッサを抜けてから，上記操作を行って下さい．

2) ファンクションキーを押したが実行したくないとき(キャンセル)

ESCキー又はそれぞれのメニューのコマンド以外の文字を入力することで取り消すことができます．コマンドがキャンセルされると元の編集状態(エディットモード)に戻ります．

キー操作	機　能
F1	編集の終了や新しいファイルのオープンに関するコマンド
F2	ファイルの分割と統合に関するコマンド
F3	カーソルの移動に関するコマンド
F4	検索と置換に関するコマンド
F5	文字列の移動や，複写に関するコマンド

では，それぞれのコマンドについて順に説明していくことにします．

4.1.1　編集を終了したい時や新しくファイルをオープンしたい時(F1)

編集の終了やファイルの更新をするときは，それぞれの場合に応じて適切なコマンドを選択します．

メニュー項目	機能
E セーブ終了	現在編集しているファイルをセーブして編集を終了します．
Q 中断	現在編集しているファイルをセーブせずに編集を終了します．
S セーブ	現在編集しているファイルをセーブして編集は継続します．
N セーブ/新ファイル	現在編集しているファイルをセーブして新しいファイルを編集します．
A 中断/新ファイル	現在編集しているファイルをセーブしないで新しいファイルを編集します．
O 再編集	現在編集しているファイルをセーブしないで最初からやり直します．

- E(Save/End)のコマンドを実行すると現在編集しているファイルをセーブしてエディタは終了し，画面はMSX-DOSのプロンプトA>に戻ります．このとき編集ファイルに変更がなければセーブは行われません．

- Q(Quit)のコマンドを実行すると現在編集しているファイルをセーブしないでエディタは終了し，画面はMSX-DOSのプロンプトA>に戻ります．このとき編集ファイルに変更があればアボートプロンプト(セーブせずに終了 Y/N ?)を画面に表示してファイルをセーブしないで終了してよいかどうかを聞いてきます．ここでY(Yes)と入力するとエディタは終了し，それ以外の入力では編集状態(エディットモード)に戻ります．

- S(Save/edit)を実行したときは，画面は編集画面のままです．編集の途中で念のためセーブしておくときなどに使います．

- N(save/New)やA(Abort/new)のコマンドを入力すると画面上に“ファイル名：”と表示されるので，新しく編集したいファイル名を入力します．なお，このときディスク上に存在するファイル名を入力すると，そのファイルに切り換えることができます．新しい(存在しない)ファイル名のときは，画面下に<New file>の表示がでます．NやAのコマンドは，いくつものファイルを続けて編集したいときに便利です．

● O(Original)のコマンドを入力すると現在編集しているファイルを最初からやり直します．このときアボートプロンプト(Abort Y/N ?)は表示しません．

S(Save/edit)，N(save/New)，A(Abort/new)のコマンドを行っても，YANKバッファの内容は保持されたままですので，別のテキストファイルに対してペースト処理を行うことができます．(ペースト処理については4.1.5を参照して下さい)

ファイルを読み込むとき(起動時，下記 I..Insert コマンドも含む)は改行までを1行(画面の右端までではない)として読み込みますが，本エディタは改行までの1行の長さに制限(画面上10行)を設けてありますので，それを越える1行は分割(改行コードを付加)されます．

4.1.2 ファイルを分割したり，統合したりしたい時(F2)

メニュー項目	機　能
I ファイル挿入	カーソルのある行の前に指定したテキストファイルを挿入します．
W ファイル書き込み	現在編集しているテキストを指定した名前のファイルで書き込みます．
Y ヤンクバッファ書き出し	YANKバッファの内容を指定した名前のファイルで書き出します．
D ディレクトリ	ディレクトリの一覧を表示します．

● I(Insert file)を入力すると，挿入するファイル名を聞いてくるので入力します．
カーソルのある前の行に，指定したファイルが挿入されます．
ファイルを統合したいときなどに使います．

● W(Write File)を入力すると，書き出すファイル名を聞いてくるので入力します．
Wコマンドは編集後のファイルをセーブしようとしたとき，ディスクに空きがなくセーブできなかった場合，パス名(主にドライブ名)を変更して別のドライブにセーブしたり，編集前のファイルの名前を変えてそのまま残しておきたいときなどに使用します．

● Y(Yank buffer write)を入力すると，ファイル名を聞いてくるので入力します．
YANKバッファに現在格納されている内容をファイルとして書き出します．
ファイルを分割したいときなどに使います．(YANKバッファについては4.1.5を参照して下さい)

● D(Directory)を入力すると，ファイル名を聞いてくるのでドライブ名，パス名，ファイル名(ワイルドカードも含む)を入力します．例えば，ここで*.*と入力するとカレントドライブのカレ

ントディレクトリのすべてのファイル一覧を表示します．また1画面に表示しきれないときは，画面いっぱいに表示したところで入力待ちになります．なにかキー入力をすると残りの表示を行います．

4.1.3　カーソルを目的の場所へ早く移動したいとき(F3)

大きなファイルでは，離れた場所にカーソルを移動するのにカーソルキーを使っていたのでは不便です．大きなファイルを効率良く編集するときに使います．

メニュー項目	機　能
G 行移動	カーソルを指定する行番号の行へ移動します．
T 第1行移動	カーソルをテキストファイルの第1行へ移動します．
B 最終行移動	カーソルをテキストファイルの最終行へ移動します．

T(第1行移動)コマンドは，CTRL+Oでも行えます．

B(最終行移動)コマンドは，CTRL+Pでも行えます．

- G(Go to line)コマンドを入力すると画面上に“行番号：”と表示されます．カーソルを移動したい行番号を入力します．

- G，T，Bのコマンドを実行するとカーソルの行，列，ファイル名を表示します．

ここでいう行は改行までのことをいいます．つまり画面上に複数行に渡って表示されていても，改行までその行は1行と勘定される訳です．画面の右端に文字がくるまでは，見た目の行数とコマンドで表示される行数は一致していますが，右端を越えて画面上，次の行に渡るような長い行がある場合は，見た目の行数とコマンドで表示される行数は一致しませんので注意してください．

編集テキストの最終行より大きな行番号を指定されると自動的に最終行に移動します．

現在のカーソル位置を知りたければ，編集状態(エディットモード)で，CTRL+Qキーを押すだけで，そのカーソルの行，列と編集テキストファイル名を表示します．

4.1.4　文字列を検索したり，置き換えたりしたいとき(F4)

メニュー項目	機　能
F 前方検索	検索したい文字列をカーソルのある位置から前方向(下方)に向かって検索します.
B 後方検索	検索したい文字列をカーソルのある位置から後方向(上方)に向かって検索します.
A 全置換	カーソルのある位置から前方向(下方)に向かって指定された旧文字列を指定された新文字列にすべて置き換えます.
R 置換	カーソルのある位置から前方向(下方)に向かって指定された旧文字列を指定された新文字列に1つずつ置き換えます.
S 設定	検索や置き換えの際，大文字・小文字の区別をするか，しないかを決めます.

文字列の前方検索は，'CTRL+^'でも行えます.

文字列の後方検索は，'CTRL+¥'でも行えます.

- F(Forward)やB(Backward)のコマンドを入力すると，"検索文字列：　"と表示されますので，検索したい文字列を入力します．入力文字列の修正は，BSキー，CTRL+Uキーで行えます．また，このとき，'CTRL+^'か'CTRL+¥'を押すと，押すごとに元の編集画面のカーソル位置から1ワードずつ入力文字列に拾ってきます．編集画面上に検索したい文字列がある場合，この機能を使えば文字列を手で入力することなく検索が行えます.
もちろん拾ってきた文字列を修正することもできます.
入力文字列は1度設定すると再び設定されるまで保持されますので，検索の再実行(CTRL+]，CTRL+ [)を行うことができます.

文字列が見つかった場合には，検索した文字列がリバース(反転)します.
文字列が見つからなかった場合には，画面上に"見つかりません"と表示されます.
TABを入力すると入力文字列には"^I"と表示します.

- A(Replace All)コマンドを入力すると"旧文字列：　"と表示されるので，置き換えられる文字列を入力します．次に，"新文字列："と表示されるので，置き換える文字列を入力します．Aコマンドでは，カーソルの位置から前方向(テキストの最後の方向)にある旧文字列に該当するものをすべて置き換えてしまうので注意が必要です.確認しながら置き換えたいときは，Rコ

マンドを使うと良いでしょう.

● R(Replace)コマンドでの置換は，カーソルのある位置から前方向(テキストの最後の方向)に向かって実行されます．テキストのすべての領域に対して行いたいときは，カーソルをテキストの最初の行に移動してから行います.
置き換える新旧文字列を入力すると，置換の対象となる文字列が見つかればその文字列をリバースし，画面にはガイドが表示されます.

リターンキーを押すと文字列は置換されます.
スペースキーを押すと置換はされず次の対象文字列にカーソルが移動します.
ESC キーを押すと編集状態(エディットモード)に戻ります.

● S(Setting)コマンドを入力すると，現在の状態が表示されます．設定を変更するときはスペースキーを押して区別する(Care)か区別しない(Don't care)かを選びリターンキーを押して下さい.
区別する(Care)にすると大文字・小文字の区別をすることになり，以降の検索や置換は大文字・小文字を区別して行われます．つまり文字列“ABC”で検索や置換を行っても，“abc”や“Abc”や“aBC”は該当文字列として扱われません．区別する(Care)にすると逆に全て該当文字列として扱われます．エディタ起動時は区別しない(Don't care)になっています.
リターンキーかESC キー(キャンセル)を押すと再びF4 のメニューが表示されて検索や置換のできる状態になります.

検索文字列や置換文字列の旧文字列中にワイルドカードの“?”を使用して任意の1文字に該当させる事ができます．例えば文字列を“TE?T”と入力したとすると，テキストの中の“TEST”や“TEXT”や“TE2T”などはこれに該当する事になり，該当する文字列全てに対して検索や置換が行われます．なお，“?”という文字を検索や置換したい場合は，“¥?”というふうに“?”文字の前に“¥”文字を付けてください.

コントロールコード(^C や^X など)に対しても検索や置換を行えますが，改行に対して検索や置換を行うときは検索文字列や置換文字列に“^J”を入力してください．本エディタでは，検索や置換を行うときの改行の意味を“^J”としています．またテキスト中の改行に対して置換を行うときは置換を行った結果，1行の長さが長くなることに十分注意して下さい．本エディタは1行を改行までとし，その長さに制限(画面上10行)を設けてあります．“^J”を入力するには，実際のキー操作として‘CTRL＋V’，‘CTRL＋J’と押します.
コントロール文字の詳しい入力の仕方のついては，「4.3.1　内容更新を行う操作」をご覧ください.

AKID の40桁モードのときは上記の文字列のリバース(反転)は行われず，それに代わって，文字列が’<’と’>’で囲まれます.

4.1.5　文字列を移動したり，複写したりしたいとき(F5)

このキーは，他のファンクションキーと違って，押してもメニューが表示されません．その代わりに，カーソルのあった行が先頭から文字列の終わりまでリバース(反転)します(空行はリバースしません)．これはこれから移動したり，複写したりしたい文字列ブロックの開始行を指定したという意味です．ですからこのキーの使い方としては，まず，編集状態で移動したり複写したりしたい文字列の開始行にカーソルを移動して，F5 を押します．すると選択された行がリバースします．次にカーソルを，移動したり複写したりしたい文字列の終了行に移動して，リターンキーを押しますと，以下のメニューが表示されます．ここで実行したいメニューの頭の文字を入力すると以下の処理を実行し編集状態に戻ります．

メニュー項目	機　能
C 複写	文字列ブロックを YANK バッファに記憶します．
A 追加複写	文字列ブロックを YANK バッファに追加記憶します．
X 削除	文字列ブロックを削除しYANKバッファへ記憶します．

終了行でリターンの代わりにメニューの頭の文字を入力すると，メニュー表示を省略して実行します．

文字列の移動や複写は YANK バッファというバッファを介して行われます．
C(Copy)や A(Append copy)は主に文字列を複写するときに，X(Cut)は文字列を移動するときに使います．YANK バッファを使った操作は，初めは多少煩わしく感じますが，慣れると大変便利です．

記憶された YANK バッファの文字列ブロックの取り出し(ペースト処理)はペーストキー(CTRL＋@)を用います．編集状態でペーストキー(CTRL＋@)を入力すると，現在カーソルのある位置に YANK バッファの文字列ブロックを挿入します．ペースト処理をするときは，まず YANK バッファの文字列ブロックを出力したい位置にカーソルを移動してから行います．
このような YANK バッファを介して文字列ブロックを格納したり取り出したりする処理のことをカット＆ペースト処理と呼ぶことにします．

- C(Copy)，A(Appendr　copy)，X(Cut)を入力するとリバースしている文字列ブロックが YANK バッファに格納されます．

- A(Append copy)コマンドの場合は，今まで YANK バッファに格納されていた文字列に加えてリバースしている文字列ブロックが追加して記憶されます．

● Xコマンドの場合は，文字列ブロックの選択が終了すると同時にその文字列ブロックは画面から削除されます．そして削除された文字列ブロックがYANKバッファに格納されます．

文字列ブロックの指定の途中でESCキーを押すとキャンセルされて編集状態(エディットモード)に戻ります．

文字列ブロックの指定で，終了行は開始行よりも前方向(下方)に対してのみ有効です．終了行を開始行よりも後方向(上方)に指定するとリバースしている行が無くなり，実行しても何も行われません．

YANKバッファに格納できる最大行数は2400行です．これを越えて格納しようとすると，エラーになります．

AKIDの40桁モードのときは上記の文字列のリバース(反転)は行われず，それに代わって，文字列が'<'と'>'で囲まれます．

CコマンドやXコマンドで新しく文字列ブロックを記憶させると前に格納されていた文字列ブロックは，失われますが，新しく記憶させない限り何度でも取り出すことができます．たくさんの箇所に複写したいとき便利です．

開始行が指定されてから，終了行を指定するまで使用できるキーは以下の通りです

CTRL+E，上カーソル	カーソルを1行上へ移動
CTRL+X，下カーソル	カーソルを1行下へ移動
CTRL+S，左カーソル	カーソルを1文字左へ移動
CTRL+D，右カーソル	カーソルを1文字右へ移動
CTRL+W，SHIFT+上カーソル	カーソル位置は変化せず，1行ロールアップ
CTRL+Z，SHIFT+下カーソル	カーソル位置は変化せず，1行ロールダウン
CTRL+C	ロールアップ(ライン数はCONFIG可能)
CTRL+R	ロールダウン(ライン数はCONFIG可能)
CTRL+A	カーソルを1ワード後方(左方向)へ移動
CTRL+F	カーソルを1ワード前方(右方向)へ移動
CTRL+O	カーソルをテキストの第1行へ移動
CTRL+P	カーソルをテキストの最終行へ移動
CTRL+^	カレント検索文字列の設定，前方検索
CTRL+¥	カレント検索文字列の設定，後方検索
CTRL+]	カレント検索文字列の前方検索(Forwardと同じ)
CTRL+[	カレント検索文字列の後方検索(Backwardと同じ)

<FUNCTION 3>

G 行移動	カーソルを指定する行番号の行へ移動
T 第1行移動	カーソルをテキストの第1行へ移動
B 最終行移動	カーソルをテキストの最終行へ移動
リターン，CTRL+M	Gと同じ

<FUNCTION 4>

F 前方検索	カレント検索文字列の前方検索
B 後方検索	カレント検索文字列の後方検索
A 全置換	テキスト内の指定された全ての文字列の置換
R 置換	テキスト内の指定された文字列を1つずつ置換
S 設定	検索/置換の際，大文字・小文字の区別の設定
リターン，CTRL+M	F と同じ

4.2 コマンドモードのエラー

以下のエラーメッセージは，KID の場合は日本語，AKID の場合は英語で出力されます

●“ファイルがオープンできません”，“Can not open file”

指定されたファイル名でファイルオープンができない．

処理　何もしないで編集状態(エディットモード)に戻る．

対策　ファイル名を正しく指定する．

●“ファイルが見つかりません”，“File not found”

ディレクトリ一覧表示コマンドで，指定されたファイル名が見つからない．

処理　何もしないで編集状態(エディットモード)に戻る．

対策　ドライブ名，パス名，ファイル名を正しく指定する．

●“無効な行番号です”，“Invalid line number”

無効な行番号が指定された．

処理　何もしないで編集状態(エディットモード)に戻る．

対策　行番号を正しく入力する．

●“ディスクがいっぱいです”，“Disk full”

ディスクに書いている途中でディスクがいっぱいになった．

処理　ディスクがいっぱいになるまで書き出す．

対策　ドライブを替えるか，別のディスクにする．

●“ヤンクバッファが空です”，“Yank buffer empty”

まだ何も Yank バッファに格納されていないのにペースト処理を行った．

処理　何もしないで編集状態(エディットモード)に戻る．

対策　不必要．

●“ヤンクバッファがいっぱいです”，“Yank buffer full”

指定された文字列ブロックの行数が大きすぎて Yank バッファに格納しきれない．

処理　Yank バッファにできるだけ格納する．

対策　このときは指定通りの格納が行われていないので，文字列ブロックを分割して，Yank バッファに格納する行数を減らす．

●“メモリが足りません”，“Out of memory”

ファイルを読み込んでいる途中，Yank バッファに格納する途中，あるいはペースト処理の途中で，メモリがいっぱいになった．

処理　ファイルの読み込みの途中，Yank バッファに格納する途中であれば，メモリがいっぱいになるまで読み込み，格納をする．
ペースト処理の途中であれば，メモリがいっぱいになるまでペーストする．

対策　ファイルの読み込みの途中で，エラーになるのであればファイルを分割するなどして読み込める大きさにする．
編集テキストか YANK バッファを削除するなどして，使用メモリを減らす．

●“総行数が多すぎます”，“Too many Lines”

ファイルを読み込んでいる途中，あるいはペースト処理の途中で，編集テキストの行数が，編集可能な最大行数を越えた．

処理　最大行数まで読み込み，あるいはペースト処理を行う．

対策　ファイルの読み込みの途中で，エラーになるのであればファイルを分割するなどして読み込める行数にする．
編集テキストを削除するなどして，総行数を減らす．

4.3 エディットモード

AKID が起動され最初にカーソルが表示された状態がエディットモードです．エディットモードでの操作はさらに次の 3 つに分けることができます．

1)キーボードからの打ち込みにより編集中のプログラムや文書の内容を更新する操作
2)内容更新を行いたい箇所にカーソルを移動するためのカーソル移動操作
3)コマンドモードに移行するための操作

上記のうち3)については前章で述べていますのでここでは,1)の入力操作と2)のカーソル移動操作について解説します．また，本章では1)，2)を合わせて編集操作と呼ぶことにします．

4.3.1　内容更新を行う操作

1)文字やコントロールコードの入力

AKIDではひらがな，カタカナを含む全ての1バイト文字(グラフィック文字は入力できません)および，一部を除くほとんどのコントロールコード(00Hから1fH)を入力することができます．入力はAKIDとKIDでは異なりますし，また文字とコントロールコードでも入力方法が異なります．

AKIDの文字入力

文字の入力方法は，MSX-DOSのコマンド入力と同じです．キーボードから直接入力したい文字や数字を打ち込んで下さい．CAPSキーや，かなキーの機能も通常のMSX-DOSやMSX-BASICのコマンド入力時と同じです．また，シフトキーを押しながら，かなキーを押すことによりローマ字仮名入力となります．

改行(CR，LF)の入力はRETURNキーにより行います．

タブの入力はTABキーまたはCTRL+Iにより行います．

KIDの文字入力

MSX-WRITEなどの日本語フロントエンドがある場合には漢字などの全角文字が入力できます．フロントエンドの起動法や具体的な入力方法はフロントエンドのマニュアルをご覧下さい．また，MSX-DOS2の単漢字変換の機能は用いることができません．

フロントエンドが無い場合の入力はAKIDと同じですが，ひらがなは全角文字で入力されます．つまり，フロントエンドがないと漢字を含むファイルを読み込むことはできますが，ひらがな以外の全角文字(漢字など)の入力を行うことができないことになります．

コントロールコードの入力

コントロールコードの入力方法はAKIDもKIDも同じです．

コントロールコードはCTRL+V(CTRLキーを押しながらVキーを押す)を入力した後,実際に入力したいコントロールコードを与えることにより入力することができます．

例えば，CTRL+@(00H)を入力したい場合まずCTRL+Vを入力します．この時点ではエディット中の画面は何も変わりません．次に入力したいCTRL+@(CTRLキーを押しながら@キーを押す)を押すことによりデータとして00Hが入力されます．つまりCTRL+Vはこれからコントロールコードの入力をすることをエディタに教えるための前準備ということができます．またCTRL+Vのあとにコントロールコード以外を入力してもその入力は無視され，CTRL+Vの効力は解除されます．

こうしてコントロールコードの入力ができたわけですが，コントロールコードには画面上表示

可能な文字が無いため，便宜上“^@”や“^C”のように文字の前に“^”をつけてその存在を表示します．これもMSX-DOSなどと同じですので見慣れた表示方法でしょう．
ただし，タブコード(08H)は次のタブ位置までの空白文字として表示されます．従って，CTRL+I(08H)を入力しても画面に“^I”と表示されることはありません．

2)インサートモードとオーバーライトモード

本エディタにはインサート(挿入)モードとオーバーライト(上書き)モードの二つの入力モードがあります．二つのモードはINSキーを押すことにより交互に切り替えることができ，モードの区別はカーソルの形状により区別することができます．

すなわち，カーソルが■のように縦ながの時はオーバライトモードで，この状態からINSキーを押すことにより，カーソルの形状が_のように変わりインサートモードになります．エディタ起動時の状態はインサートモードです．

次にインサートモードとオーバーライトモードの違いを解説します．

A)インサートモード

インサートモードでは入力された全ての文字やコントロールコードは現在のカーソル位置に挿入されます．

例えば，

あいえお

の状態で“う”を入力すると“い”と“え”の間に文字“う”の2バイトコードが挿入され，画面上では

あいうえお

のようになります．同様にCTRL+C(CTRL+Vを押してからCTRL+C)を入力すれば，

あい^Cえお

となり，タブを入力すると

あい　　えお

のようになります．

インサートモードで注意すべきなのは改行(RETURN)の入力です．行の先頭で改行を入力す

ると，カーソル位置に1行空行がつくられ，元の行以降はその分下にスクロールします．カーソル挿入された空行の次の行の先頭に移動します．

すなわち，

あいうえお
かきくけこ
さしすせそ

の状態でRETURNキーを押すと，

あいうえお

かきくけこ
さしすせそ

のようになります．また，行の末尾で改行を入力すると，現在のカーソル行の下に1行空行が挿入され，以降の行は1行ずつ下にスクロールします．カーソルは挿入された空行の先頭に移動します．

あいうえお
かきくけこ

ここでRETURNキーを押すと，

あいうえお

かきくけこ

となるわけです．されに行の途中で改行を入力すると，その行はカーソル位置を境にカーソルより前とカーソル以降の2行に分割されカーソルは後の行の先頭に移動します．たとえば

あいうえお

の状態でRETURNキーを押すと

あい
うえお

のようになります．

B)オーバーライトモード

オーバーライトモードでは現在のカーソル位置にある文字やコントロールコードが入力した文字やコントロールコードに置き換えられます．新しいデータによって上書きされ元のデータが失われることになります．

たとえば，

あいうえお

の状態でCTRL+C(CTRL+V を押してからCTRL+C)を入力すると

あい^Cえお

のようになり，
また，半角文字“u”を入力すれば

あいuえお

のように“えお”の2文字が半角分だけ右に移動します．

オーバーライトモードでは改行やタブを入力してもデータとして書き込まれずカーソルの移動のみが行われます．すなわち，改行では，空行の挿入や行の分割は行われず，単純にカーソルが次の行の先頭に移動します．またタブもタブコードがデータとして入力されることなくカーソルが次のタブ位置に移動します．

オーバーライトモードで注意すべきなのは，行末での入力です．

カーソルが行の末尾にある場合，改行以外の入力はインサートモード同様の動作を行います．すなわち，行の最後の文字と改行の間に入力された文字が挿入されていくことになります．これはタブの場合も同じです．これによりオーバーライトモードでも現在の行末に新しい文字を書き足していくことができます．

3)文字やコントロールコードの削除

入力に対応して必要なのが，入力した文字やコントロールコードの削除機能です．これにはデリートとバックスペースの二つの機能があります．

デリート

デリートの機能はDEL キーを押すことにより行います．デリートは現在のカーソル位置の文字やコントロールコードを削除します．これによりカーソルより後ろが1文字分だけ(コントロー

ルコードの場合半角2個分)前に移動します．カーソルの位置は移動しません．

デリートはDELキーの他にCTRL+Gによっても行うことができます．

バックスペース

バックスペースの機能はBSキーを押すことにより行います．バックスペースは現在のカーソル位置の1つ前のデータを削除します．これによりカーソル及び，カーソル以降が前に移動します．行の先頭でバックスペースを行うと，前の行の改行が削除されて，前の行と結合して1つの行となります．

バックスペースはBSキーの他にCTRL+Hによっても行うことができます．

デリートやバックスペースで注意すべきなのは改行の削除です．すなわち行の末尾でのデリートや行の先頭でのバックスペースです．改行が削除されることにより，前後の2行が1行に結合されます．このことは，場合によっては期せずして長い1行を作り出すことになります．AKIDでは1行の長さに制限があるため(画面上で10行以内)，これを超える行をつくるとエラーとなり行の一部が失われてしまうことがあります．詳しくは，4.2　コマンドモードのエラーをご覧下さい．

3) TINYカット&ペースト

TINYカット&ペーストとは，コマンドモードで紹介したカット&ペーストの縮小機能版とでも言えばよいでしょうか，1行の中での手軽なカット&ペースト機能でエディットモードで実行できるものです．

TINYカット

TINYカットには以下のように削除する領域の異なる3種類の機能があります．

a) 1ラインのカット(CTRL+Y)

カーソル行の画面幅分の領域を削除します．削除されたデータはTINYペーストのためのバッファ(以降TINYカットバッファ)に格納されます．この機能はCTRL+Yを押すことのより行います．

b) カーソル位置までのカット(CTRL+U)

カーソル行の先頭(画面の左端ではありません)からカーソル位置の前までを削除します．削除されたデータはTINYカットバッファに格納されます．これはCTRL+Uにより行います．

c) カーソル行以降のカット(CTRL+K)

カーソル位置以降，行末まで(画面の右端ではありません)を削除します．改行は削除されないため，行の先頭で行っても空行が残ります．削除されたデータはTINYカットバッファに格納されます．この機能はCTRL+Kにより行います．

TINY ペースト(CTRL+L)

TINY ペーストは TINY カットで TINY カットバッファに格納されたデータを現在のカーソル位置に挿入します．オーバーライトモードの場合でもデータは上書きではなく挿入となります．TINY カットバッファのデータは最後に行った TINY カットにより格納されたものだけが有効です．TINY ペーストは CTRL+L により行います．

4)その他

a) 1行の挿入(CTRL+N)

現在のカーソル位置に空行を1行挿入します．これのより現在のカーソル行以降の行は1行ずつ下にスクロールします．

b) 1ワードの削除(CTRL+T)

カーソル位置から次のワード(単語)の前までを削除します．改行は削除されず，削除されたデータは TINY カットバッファに格納されません．

4.3.2 カーソルを移動する操作

ここまでの編集作業はすべてカーソルのある位置を基準に行われるものでした．そこでカーソルを移動する機能が必要になります．これによりプログラムや文書の中を自由に動き回り，任意の箇所を更新することが可能になるのです．

まず，本エディタのカーソル移動機能全般に渡っての解説を行います．

1) AKID のカーソル移動の規則

まず注意しなければならないのは，AKID ではデータのない場所にはカーソルを移動できないということです．このことは画面上のイメージと実際のデータの内容がほぼ一致しているということであり，プログラムや通常の文書の編集ではむしろこのほうが好都合であるといえます．

次にやはりこのことと関連しているのですが，AKID にはカーソルの横方向の位置を一定に保とうとする機能があります．これだけではなんのことかわからないと思いますから，具体的に説明してみましょう．

```
        Quality of mercy is not strain'd
The quality of mercy is  not strain'd,  It  droppeth  as  the
gentle rain from heaven, Upon the place beneath: it  is twice
blest; it blesseth him that gives and him takes;
'Tis mightiest in the mightiest.
```

いまカーソルは2行目の“a”の文字の上にあります．ここでカーソル下キーによりカーソルを3行目に移動します．カーソルは3行目の“s”の文字の上つまり真下に移動しました．

```
                    :y is not strain'd
The qua             : not strain'd,  It  droppeth  as  the
gentle r            ), Upon the place beneath: it  is twice
blest;               that gives and him takes;
'Tis mightiest in the mightiest.
```

もう1度カーソルを下に移動します．今度はカーソルは真下には移動せず，4行目の改行の位置に移動しました．これはデータのない位置にはカーソルは移動しないという規則ですから予想できる結果です．

```
        Quality of mercy is not strain'd
The quality of mercy is  not strain'd,  It  droppeth  as  the
gentle rain from heaven, Upon the place beneath: it  is twice
blest; it blesseth him that gives and him takes;
'Tis mightiest in the mightiest.
```

ここでカーソル上キーによりカーソルを3行目に戻してみます．カーソルは真上に移動するのではなく，“s”の文字にむかって右斜め上に移動します．すなわちエディタがまえのカーソルの横方向の位置を記憶していて，そこにデータがあってカーソル移動が可能な限りその位置をキープしようとするのです．

```
        Quality of mercy is not strain'd
The quality of mercy is  not strain'd,  It  droppeth  as  the
gentle rain from heaven, Upon the place beneath: it  is twice
blest; it blesseth him that gives and him takes;
'Tis mightiest in the mightiest.
```

もともと3行目から4行目への移動ではユーザー自身が左に移動する操作をしないにもかかわらず，エディタ自身の都合で左斜め下に移動してしまったわけですからその後のカーソルの上下でできる限りもとの横位置に戻そうという発想からこのような動作をするわけです．

この機能は行と行の間に空行がある場合などに特に威力を発揮します．従来の多くのエディタではカーソルの上下により空行にぶつかるとカーソルは左端に移動してしまい，次に長い行にもどってもカーソルは行の先頭にとどまったままでした．これに対してKID，AKIDではユーザー自身がカーソルを左右に移動する操作をしない限りカーソルの横位置は可能な限り元の位置に近付けようとします．

カーソルの左右方向の移動操作により，AKIDは新しい桁位置を記憶することになります．つまり上の例では，4行目でカーソル右キーにより“；”の位置にカーソルを移動してから3行目に戻すと，カーソルは真上の“：”の位置に移動するわけです．

2)カーソル移動のための操作

つぎにカーソルを移動する操作について個々の機能を具体的に解説していきます.

a) カーソルキー

最も基本的なカーソルの左右上下の移動はカーソルキーにより行います.それぞれカーソルキーの矢印の方向にカーソルが移動します.

行の先頭でカーソル左キーを押した場合は上の行の末尾に,行の末尾でカーソル右キーが押されればカーソルは次の行の先頭に移動します.

また画面をはみ出す場合には行がある限りスクロールがおこります.

b) CTRL キーや SHIFT キーを活用したカーソル移動操作

AKID ではより効率の良い編集作業のために CTRL キーや SHIFT キーを押しながら他のキーを押すことで,カーソル移動を行うことができます.

特に,キーボードを見ずに入力のできる方は,ホームポジションから手を離すことなく操作ができ,より効率を上げることができるでしょう.

CTRL＋E	カーソルを1行上へ移動します(カーソル上と同じ).
CTRL＋X	カーソルを1行下へ移動します(カーソル下と同じ).
CTRL＋S	カーソルを1文字左へ移動します(カーソル左と同じ).
CTRL＋D	カーソルを1文字右へ移動します(カーソル右と同じ).
SHIFT＋カーソル上 または,CTRL＋W	カーソルを移動しないで1行下にスクロールします.スクロールする行が無い場合,カーソルは1行上に移動します.
SHIFT＋カーソル下 または,CTRL＋Z	カーソルを移動しないで1行分上へスクロールします.スクロールする行が無い場合,カーソルは1行下に移動します.
CTRL＋C	1画面分スクロールアップします.
CTRL＋R	1画面分ロールダウンします.
SHIFT＋カーソル右	カーソルを行の右端へ移動します.
SHIFT＋カーソル左	カーソルを行の左端へ移動します.
CTRL＋A	カーソルを1ワード後方(左方向)へ移動します.
CTRL＋F	カーソルを1ワード前方(右方向)へ移動します.
CTRL＋O	カーソルをテキストの第1行へ移動します.
CTRL＋P	カーソルをテキストの最終行へ移動します.

4.3.3 エディットモードのエラー

1) 1行が長過ぎる場合のエラー

入力中に1行の長さが，画面上で10行を超えると次のエラーメッセージが表示されます．

AKIDの場合

```
Line too long
```

KIDの場合

行が長過ぎます

この場合画面上10行の範囲におさまるように行の末尾が切り捨てられてしまいます．通常のプログラムや文書では1行の長さがこれほど長くなることはめったにありませんが，改行を削除することにより比較的簡単に長すぎる行が作られることがありますので注意が必要です．

2)行数が大き過ぎる場合のエラー

エディットできる行数は画面上の行数で，2000から5000行です．これを超えると次のメッセージが表示されます．

AKIDの場合

```
Too many lines
```

KIDの場合

総行数が多すぎます

この場合行数を減らさない限り，行を増やすような操作(改行や空行の挿入など)を行うことができなくなります．

3)テキストのためのメモリが足りない場合のエラー

本エディタでは高速化のためにテキストのためのバッファはすべてVRAM上にとられています．MAIN-RAMにとるよりははるかに大きな領域が確保できるのですが，テキストがあまりに長大になるとメモリが足りなくなる場合があります．この場合次のエラーメッセージが表示されます．

AKIDの場合

```
Out of memory
```

KIDの場合

メモリが足りません

このエラーがおこると最後に入力した1行が失われることがあります．メモリ不足は本エディタの物理的な限界であるため，行を減らしたり，カット&ペーストでカットやコピーされているデータを捨てるなどしてメモリの空き領域を増やさない限りエディットを続けることはできません．

第５章　キーアサインの変更

5.1　KID/AKID 用キーアサイン変更プログラム CONKID

例えばエディットモードでカーソルを上下左右に１つずつ移動したいとき，カーソル移動キーを使うとどうしてもホームポジションから手を離さなければなりません．こんなときにカーソル移動機能を[CTRL]＋文字キーなどにセットできれば，手をホームポジションにおいたままカーソルの操作することができるようになります．また，カーソル移動に限らず様々な機能をセットすることにより，入力の効率もよくなることでしょう．このように任意のキーに，様々な操作を割付ることをキーアサイン(KEY ASSIGN)といい，本エディタは[CTRL]＋文字キーに設定することができます．

CONKID は，上記のようなエディットモードのキーアサイン，スクロール行数，起動時の入力モード(挿入／上書)，検索／置換での大文字・小文字の区別の設定を行い，エディタ本体にセーブすることができます．上記のような機能を変更するときに使用して下さい．

5.2　起動方法

MSX-DOS2 のコマンド入力行より次に示すように入力して下さい．

```
A>CONKID ［エディタファイル名］
```

引数としてエディタ本体のファイル名が必要です．エディタのファイル名は省略することができますが，その場合，立ち上げ時の画面が漢字モードならば“KID.COM”を，ANK モードならば“AKID.COM”を指定したことになります．複数のファイルを指定することはできません．なお，CONKID は１行 40 桁以上にして使用して下さい．また，ANK モードで立ち上げるときはスクリーンモード 0 でなければなりません．起動の際にエラーが発生したときはエラーメッセージを表示し，MSX-DOS2 のコマンド入力行に戻ります．

＜起動時の画面例＞

```
漢字モード O

  1. カーソル 1行上へ移動              ^E
  2. カーソル 1行下へ移動              ^X
  3. カーソル 1文字右へ移動            ^D
  4. カーソル 1文字左へ移動            ^S
  5. カーソル 1ワード前方へ移動        ^F
  6. カーソル 1ワード後方へ移動        ^A
```

7. カーソル 行の端へ移動　^B
8. カーソル テキストのトップへ移動　^O
9. カーソル テキストのエンドへ移動　^P
10. 1行ロールアップ　^Z
11. 1行ロールダウン　^W

ANK モード

1. ｶｰｿﾙ 1ｷﾞｮｳｳｴﾍｲﾄﾞｳ　^E
2. ｶｰｿﾙ 1ｷﾞｮｳｼﾀﾍｲﾄﾞｳ　^X
3. ｶｰｿﾙ 1ﾓｼﾞﾐｷﾞﾍｲﾄﾞｳ　^D
4. ｶｰｿﾙ 1ﾓｼﾞﾋﾀﾞﾘﾍｲﾄﾞｳ　^S
5. ｶｰｿﾙ 1ﾜｰﾄﾞｾﾞﾝﾎﾟｳﾍｲﾄﾞｳ　^F
6. ｶｰｿﾙ 1ﾜｰﾄﾞｺｳﾎｳﾍｲﾄﾞｳ　^A
7. ｶｰｿﾙ ｷﾞｮｳﾉﾊｼﾍｲﾄﾞｳ　^B
8. ｶｰｿﾙ ﾃｷｽﾄﾉﾄｯﾌﾟﾍｲﾄﾞｳ　^O
9. ｶｰｿﾙ ﾃｷｽﾄﾉｴﾝﾄﾞﾍｲﾄﾞｳ　^P
10. 1ｷﾞｮｳﾛｰﾙｱｯﾌﾟ　^Z
11. 1ｷﾞｮｳﾛｰﾙﾀﾞｳﾝ　^W
12. ﾛｰﾙｱｯﾌﾟ　^C
13. ﾛｰﾙﾀﾞｳﾝ　^R
14. ｹﾝｻｸﾓｼﾞﾚﾝﾉｾｯﾃｲ，ｾﾞﾝﾎﾟｳｹﾝｻｸ　^^
15. ｹﾝｻｸﾓｼﾞﾚﾝﾉｾｯﾃｲ，ｺｳﾎｳｹﾝｻｸ　^¥
16. ｶﾚﾝﾄｹﾝｻｸﾓｼﾞﾚﾝﾉｾﾞﾝﾎﾟｳｹﾝｻｸ　^]
17. ｶﾚﾝﾄｹﾝｻｸﾓｼﾞﾚﾝﾉｺｳﾎｳｹﾝｻｸ　^[
18. 1ﾓｼﾞｻｸｼﾞｮ　^G
19. ﾊﾞｯｸｽﾍﾟｰｽ　^H
20. ｶｰｿﾙﾉｱﾙｷﾞｮｳﾜｻｸｼﾞｮ　^Y
21. ｾﾝﾄｳｶﾗｶｰｿﾙﾁｮｸｾﾞﾝﾏﾃﾞｻｸｼﾞｮ　^U
22. ｶｰｿﾙｶﾗｷﾞｮｳﾏﾂﾏﾃﾞｻｸｼﾞｮ　^K

5.2.1　起動時のエラー

●“<ファイル名>ファイルが見つかりません”，“<ファイル名>ファイルガミツカリマセン”

指定したファイルが見つからない

対策　ドライブ名，パス名，ファイル名を正しくする

●“パラメータを正しくして下さい”，“パラメータヲタダシクシテクダサイ”

複数のファイル名を指定した

対策　指定するファイルを1つにする

●“エディタファイルを指定して下さい”，“シテイサレタファイルハ，エディタファイルデハアリマセン”

指定したファイルがエディタファイルと認められない

対策　エディタファイルを指定する

●“1行の幅を40桁以上にして下さい”，“1ギョウノハバヲ 40ケタ イジョウニシテクダサイ”

1行の幅が40桁未満である

対策　DOS の MODE コマンドまたは BASIC の WIDTH コマンドで1行の幅を変更する

●“スクリーンモード 0 (1 ギョウ 40 ケタ イジョウ) デシヨウシテクダサイ”

スクリーンモードが0以外である(ANK モードでのみ発生するエラー)

対策　DOS からは MODE コマンドで，BASIC からは SCREEN コマンドと WIDTH コマンドで画面を再設定する

5.3 操作方法

5.3.1 設定方法

キーアサインに対応しているキーは，キャラクタコード 00H～1FH([DEL] キーを除く全てのコントロールキャラクタ)です．再設定をするときは，キーアサインを変更したい操作までカーソルを移動し，[CTRL] キーを押しながら文字キーを押して下さい．例えば [CTRL]+ [E] と入力した場合，画面には「^ E」と表示されます．キャラクタコード 00H～1FH は以下の通りです．

コード (10 進)	コード (16 進)	対応キー
0	00H	[CTRL]+[@]
1	01H	[CTRL]+[A]
2	02H	[CTRL]+[B]
3	03H	[CTRL]+[C]
4	04H	[CTRL]+[D]
5	05H	[CTRL]+[E]
6	06H	[CTRL]+[F]
7	07H	[CTRL]+[G]
8	08H	[CTRL]+[H]
9	09H	[CTRL]+[I]
10	0AH	[CTRL]+[J]
11	0BH	[CTRL]+[K]
12	0CH	[CTRL]+[L]
13	0DH	[CTRL]+[M]
14	0EH	[CTRL]+[N]
15	0FH	[CTRL]+[O]
16	10H	[CTRL]+[P]
17	11H	[CTRL]+[Q]
18	12H	[CTRL]+[R]
19	13H	[CTRL]+[S]
20	14H	[CTRL]+[T]

21	15H	[CTRL]+[U]
22	16H	[CTRL]+[V]
23	17H	[CTRL]+[W]
24	18H	[CTRL]+[X]
25	19H	[CTRL]+[Y]
26	1AH	[CTRL]+[Z]
27	1BH	[CTRL]+[[ ]
28	1CH	[CTRL]+[¥]
29	1DH	[CTRL]+[ ]]
30	1EH	[CTRL]+[^]
31	1FH	[CTRL]+[_]

キーアサインの変更において無効なキーが入力された場合，そのキーについては無視され，すでに他のアサインに使用されているキーであれば，「同一コードがあります」(漢字モード)または「ドウイツコードガアリマス」(ANK モード)とメッセージが返されます．この場合，すでに使用されているキーをあらかじめ[BS]キーまたは[DEL]キーで解除しておかなければなりません．アサインされていない操作は，「未使用」(漢字モード)または「ミシヨウ」(ANK モード)と表示されます．[SEL]キーにより，カーソル上のキーアサインをデフォルト値にすることができ，[HOME]キーにより，キーアサインを変更前の値に戻すことができます．

ロールアップ／ロールダウンのスクロール行数は，1 行から 30 行まで設定できます．入力できる数値は 0 から 30 までですが，0 は(現在の画面の行数-1)を表します．

起動時の入力モードの設定は，[INS]キーまたは[SPACE]キーで行います．共にトグルスイッチになっていますので，1 回押すと変更し，もう 1 回押すと元に戻ります．画面表示は，挿入モードを選択すると「挿入」(漢字モード)または「ソウニュウ」(ANK モード)，上書きモードを選択すると「上書」(漢字モード)または「ウワガキ」(ANK モード)となります．

検索／置換処理での大文字・小文字の区別の設定は，[SPACE]キーで行います．入力モードの設定と同じようにトグルスイッチになっています．画面表示は，区別する方を選択すると「する」(漢字モード)または「スル」(ANK モード)，区別しない方を選択すると「しない」(漢字モード)または「シナイ」(ANK モード)となります．

5.3.2 キー操作

5.3.2.1 カーソルの移動,設定

カーソルの移動,設定は次のようなキー操作で行います.

キー	機　能
↑	カーソルを1行上へ移動する.
↓	カーソルを1行下へ移動する.
RETURN	カーソルを1行下へ移動する.
ESC	メニューを表示する.
BS	カーソル上のキーアサインを解除し,「未使用」(漢字モード)または,「ミシヨウ」(ANKモード)と表示する.
DEL	カーソル上のキーアサインを解除し,「未使用」(漢字モード)または,「ミシヨウ」(ANKモード)と表示する.
SEL	カーソル上のキーアサインをデフォルト値にする.
HOME	カーソル上のキーアサインを変更前の値に戻す.

5.3.2.2 ESCキー

[ESC]キーを押すと画面の1行目にメニューが表示されます.

<メニュー表示画面例>

漢字モード0

```
End/Quit/Save/Default/ESC
 1. カーソル 1行上へ移動                ^E
 2. カーソル 1行下へ移動                ^X
 3. カーソル 1文字右へ移動              ^D
 4. カーソル 1文字左へ移動              ^S
 5. カーソル 1ワード前方へ移動          ^F
 6. カーソル 1ワード後方へ移動          ^A
 7. カーソル 行の端へ移動               ^B
 8. カーソル テキストのトップへ移動     ^O
 9. カーソル テキストのエンドへ移動     ^P
10. 1行ロールアップ                     ^Z
11. 1行ロールダウン                     ^W
```

ANKモード

```
End/Quit/Save/Default/ESC
```

```
 1. ｶｰｿﾙ 1ｷﾞｮｳｳｴﾍｲﾄﾞｳ                ^E
 2. ｶｰｿﾙ 1ｷﾞｮｳｼﾀﾍｲﾄﾞｳ                ^X
 3. ｶｰｿﾙ 1ﾓｼﾞﾐｷﾞﾍｲﾄﾞｳ                ^D
 4. ｶｰｿﾙ 1ﾓｼﾞﾋﾀﾞﾘﾍｲﾄﾞｳ              ^S
 5. ｶｰｿﾙ 1ﾜｰﾄﾞｾﾞﾝﾎﾞｳﾍｲﾄﾞｳ            ^F
 6. ｶｰｿﾙ 1ﾜｰﾄﾞｺｳﾎｳﾍｲﾄﾞｳ              ^A
 7. ｶｰｿﾙ ｷﾞｮｳﾉﾊｼﾍｲﾄﾞｳ                ^B
 8. ｶｰｿﾙ ﾃｷｽﾄﾉﾄｯﾌﾟﾍｲﾄﾞｳ             ^O
 9. ｶｰｿﾙ ﾃｷｽﾄﾉｴﾝﾄﾞﾍｲﾄﾞｳ             ^P
10. 1ｷﾞｮｳﾛｰﾙｱｯﾌﾟ                     ^Z
11. 1ｷﾞｮｳﾛｰﾙﾀﾞｳﾝ                     ^W
12. ﾛｰﾙｱｯﾌﾟ                           ^C
13. ﾛｰﾙﾀﾞｳﾝ                           ^R
14. ｹﾝｻｸﾓｼﾞﾚｼﾞﾉｾｯﾃｲ, ｾﾞﾝﾎﾞｳｹﾝｻｸ      ^^
15. ｹﾝｻｸﾓｼﾞﾚｼﾞﾉｾｯﾃｲ, ｺｳﾎｳｹﾝｻｸ        ^¥
16. ｶﾚﾝﾄｹﾝｻｸﾓｼﾞﾚｼﾞﾉｾﾞﾝﾎﾞｳｹﾝｻｸ       ^]
17. ｶﾚﾝﾄｹﾝｻｸﾓｼﾞﾚｼﾞﾉｺｳﾎｳｹﾝｻｸ         ^[
18. 1ﾓｼﾞｻｸｼﾞｮ                         ^G
19. ﾊﾞｯｸｽﾍﾟｰｽ                         ^H
20. ｶｰｿﾙﾉｱﾙｷﾞｮｳﾇｹｻｸｼﾞｮ               ^Y
21. ｾﾝﾄｳｶﾗｶｰｿﾙﾁｮｸｾﾞﾝﾏﾃﾞｻｸｼﾞｮ         ^U
22. ｶｰｿﾙｶﾗｷﾞｮｳﾏﾂﾏﾃﾞｻｸｼﾞｮ             ^K
```

この状態で[E], [Q], [S], [D], [ESC]を選択すると以下のようになります.

End　　設定内容をエディタ本体にセーブ後,CONKID を終了します.
セーブが正常に終了した場合は「書き込みは正常に終了しました」(漢字モード)または「カキコミハ セイジョウニシュウリョウシマシタ」(ANK モード),エラーが発生した場合は「エラーが発生しました」(漢字モード)または「ERROR ガハッセイシマシタ」(ANK モード)とメッセージを表示し,DOS に戻ります.

Quit　　セーブせずに CONKID を終了します.

Save　　設定内容をエディタ本体にセーブし,設定モードに戻ります.
エラーが発生した場合は「エラーが発生しました」(漢字モード)または「ERROR ガハッセイシマシタ」(ANK モード)とメッセージを表示し,DOS に戻ります.

Default　　すべてのキーアサインをデフォルト値にします.

ESC　　設定モードに戻ります.

5.3.3 デフォルト値

1. カーソルを1行上へ移動	^E
2. カーソルを1行下へ移動	^X
3. カーソルを1文字右へ移動	^D
4. カーソルを1文字左へ移動	^S
5. カーソルを1ワード前方へ移動	^F
6. カーソルを1ワード後方へ移動	^A
7. カーソルを行の端へ移動	^B
8. カーソルをテキストのトップへ移動	^O
9. カーソルをテキストのエンドへ移動	^P
10. 1行ロールアップ	^Z
11. 1行ロールダウン	^W
12. ロールアップ	^C
13. ロールダウン	^R
14. 検索文字列設定,前方検索	^^
15. 検索文字列設定,後方検索	^¥
16. カレント検索文字列の前方検索	^]
17. カレント検索文字列の後方検索	^[
18. 1文字削除	^G
19. バックスペース	^H
20. カーソルのある行を削除	^Y
21. 先頭からカーソル直前まで削除	^U
22. カーソルから行末まで削除	^K
23. 1ワードの文字列削除	^T
24. 改行	^M
25. カーソル行の上に1行挿入	^N
26. タブ	^I
27. 行,列,ファイル名の表示	^Q
28. 挿入,上書き状態の切り替え	^J
29. コントロール文字入力	^V
30. FUNCTION 1〜4のクリック	未使用
31. 削除文字列バッファの内容挿入	^L
32. YANKバッファの内容挿入	^@

スクロール行数(0：現在の画面の行数-1)	0
起動時の入力モード(挿入／上書)	挿入
検索/置換の大文字・小文字の区別	しない

第6章　コマンド一覧

以下に,KID,AKIDのコマンドとその意味を説明します.

6.1　コマンドモード

<FUNCTION 1>

E セーブ終了	現在の編集テキストをセーブし編集を終了.
Q 中断	現在の編集テキストをセーブせずに編集を終了.
S セーブ	現在の編集テキストをセーブ.編集は継続.
N セーブ/新ファイル	現在の編集テキストをセーブ.新しいテキストファイルを編集.
A 中断/新ファイル	現在の編集テキストをセーブせずに新しいテキストファイルを編集.
O 再編集	現在の編集テキストの編集を最初からやり直す.

<FUNCTION 2>

I ファイル挿入	カーソルのある行の前に指定したテキストファイルを挿入.
W ファイル書き込み	現在編集しているテキストを指定した名前のファイルで書き込む.
Y ヤンクバッファ書き出し	YANKバッファの内容を指定した名前のファイルで書き出す.
D ディレクトリ	ディレクトリの一覧を表示.

<FUNCTION 3>

G 行移動	カーソルを指定する行番号の行へ移動.
T 第1行移動	カーソルをテキストの第1行へ移動.
B 最終行移動	カーソルをテキストの最終行へ移動.
リターン,CTRL+M	Gと同じ

このときカーソルの行,列,編集テキストファイル名を表示します.

<FUNCTION 4>

F 前方検索	カレント検索文字列の前方検索(テキストの終わり方向).
B 後方検索	カレント検索文字列の後方検索.
A 全置換	テキスト内の指定された全ての文字列の置換.
R 置換	テキスト内の指定された文字列を1つずつ置換.
S 設定	検索/置換の際,大文字・小文字の区別をする,しないの設定.
リターン,CTRL+M	Fと同じ.

<FUNCTION 5>

カーソル移動キーにより,文字列ブロックを指定した後,以下の機能を実行します.

C 複写	文字列ブロックをYANKバッファに記憶.
A 追加複写	文字列ブロックをYANKバッファに追加記憶.
X 削除	文字列ブロックの削除とYANKバッファへの記憶.
リターン,CTRL+M	NEMU表示後,実行.

<SELECT>

FUNCTION 1～4 のクリック.

コマンドの実行

F.n キー,SELECT (F1～F4)	ファンクションの表示.
各MENUの最初の文字入力	実行.
該当なし文字入力,ESC	キャンセル

6.2 エディットモード

CTRL＋E, 上カーソル	カーソルを1行上へ移動.
CTRL＋X, 下カーソル	カーソルを1行下へ移動.
CTRL＋S, 左カーソル	カーソルを1文字左へ移動.
CTRL＋D, 右カーソル	カーソルを1文字右へ移動.
CTRL＋W, SHIFT＋上カーソル	カーソル位置は変化せず, 1行ロールアップ.
CTRL＋Z, SHIFT＋下カーソル	カーソル位置は変化せず, 1行ロールダウン.
CTRL＋C	ロールアップ(ライン数はCONFIG可能).
CTRL＋R	ロールダウン(ライン数はCONFIG可能).
CTRL＋B, (SHIFT＋左カーソル)	カーソルを行の左端へ移動(CRL＋Bはトグルスイッチ).
CTRL＋B, (SHIFT＋右カーソル)	カーソルを行の右端へ移動(CRL＋Bはトグルスイッチ).
CTRL＋A	カーソルを1ワード後方(左方向)へ移動.
CTRL＋F	カーソルを1ワード前方(右方向)へ移動.
CTRL＋O	カーソルをテキストの第1行へ移動.
CTRL＋P	カーソルをテキストの最終行へ移動.
CTRL＋Q	カーソルの行, 列, 編集テキストファイル名の表示.
CTRL＋M, リターンキー	改行. **挿入状態**　　行分割とカーソル位置の改行. **上書き状態**　　カーソル位置の改行.
CTRL＋N	上に1行挿入. カーソル行の上に1行挿入し, カーソルを挿入した行の左端に移動.
BS, CTRL＋H	バックスペース.
INS, CTRL＋J	文字の挿入状態と上書き状態の切り替え.
DEL, CTRL＋G	カーソル位置1文字削除.
CTRL＋V	コントロール文字入力指示.
CTRL＋T	1ワードの文字列削除.
CTRL＋Y	カーソルのある行を削除.
CTRL＋K	行末までの削除.
CTRL＋U	先頭からカーソル直前までの削除.
CTRL＋L	削除文字列バッファの内容を挿入.
CTRL＋@	YANKバッファの内容をテキストのカーソル位置に出力する(ペースト処理).
CTRL＋^	カレント検索文字列の設定, 前方検索.

CTRL＋¥	カレント検索文字列の設定，後方検索．
CTRL＋]	カレント検索文字列の前方検索（Forward と同じ）．
CTRL＋[	カレント検索文字列の後方検索（Backward と同じ）．

6.3　コンフィギュレーション

デフォルト値

キーアサイン（CTRL キー）	上記エディットモードの CTRL キーのアサイン．
スクロール行数	エディタを起動したときの画面の行数-1．
起動時の入力モード（挿入／上書）	挿入モード．
検索/置換の大文字・小文字の区別	区別しない．

ユーティリティ

第１章　はじめに

1.1　本書の構成

本書はMSX-DOS2ユーティリティ・ソフトウェア・パッケージについての解説です．
ただし本書は読者が8080/Z80アセンブリ言語によるプログラミングを理解しており，アセンブラに関する経験があることを前提にして書かれており，アセンブリ言語の解説書ではありません．従って，8080/Z80アセンブリ言語でプログラムをした経験がない場合，別の解説書等でアセンブリ言語に関する知識を得た上で本書を読むことをおすすめします．

本書の構成は次のとおりです．

第１章　はじめに

ユーティリティ・ソフトウェア・パッケージの概略を解説します．

第２章　ユーティリティ・ソフトウェア・パッケージを使ったプログラム開発

ユーティリティ・ソフトウェア・パッケージを使ったプログラム開発について説明します．

第３章　MSX･M-80 マクロアセンブリ言語

MSX･M-80 マクロアセンブリ言語について説明します．

第４章　MSX･M-80 の実行

MSX･M-80 マクロアセンブラの使用方法(アセンブル方法)について説明します．

第５章　MSX･L-80 リンクローダ

MSX･L-80 リンクローダの働きと使用方法を説明します．

第６章　CREF-80 クロスリファレンサ

CREF-80 クロスリファレンサの働きと使用方法を説明します．

第７章　LIB-80 ライブラリマネージャ

LIB-80 ライブラリマネージャの働きと使用方法を説明します．

付録

ASCIIキャラクタコード，オブジェクトファイルのフォーマット，アセンブラ疑似命令，命令コード一覧表を記載します．

1.2　パッケージの内容

1.2.1　ソフトウェア

本パッケージでは，ソフトウェアとして以下の4つのファイルを提供します．

M80.COM	MSX･M-80 マクロアセンブラ
L80.COM	MSX･L-80 リンクローダ
CREF80.COM	CREF-80 クロスリファレンサ
LIB80.COM	LIB-80 ライブラリマネージャ

1.2.2　ドキュメント

ドキュメントは，本書「ユーティリティ・ソフトウェア・マニュアル」です．

1.3　システムの環境

MSX-DOS2 ユーティリティ・ソフトウェア・パッケージの各プログラムは，MSX-DOS2 の環境下で動作することを前提にしています．
従ってこのユーティリティ・ソフトウェア・パッケージを使うためには日本語 MSX-DOS2 が必要です．
また，各プログラムは1台のディスクドライブでも動作するように作られていますが，2台のディスクドライブを使用することをおすすめします．

1.4　本書での表記法

本書では，コマンドおよびステートメントの書式を表わすために，以下の表記法を使います．

英文字　　ステートメントやコマンドのうち，指定されたとおりに入力しなければならないところを表わします．

＜＞(山形かっこ)　山形かっこは入力するデータを表わします.
かっこ内に項目名がある場合，指定された項目の規則に従ったデータを入力します.
例えば,＜ファイル名＞では実際のファイル名を入力します.また，かっこ内に,＜CTRL+C＞,＜RETURN＞のような英文字がある場合は，ここに書かれている名称のキーを押さなければなりません.

[](角形かっこ)　このかっこ内の項目は省略できることを表わします.

{}(波かっこ)　波かっこは，いくつかの項目からどれか１つを選択することを示します.
波かっこで囲まれている項目は，さらに角型かっこで囲まれていない限り，少なくとも１つは入力しなければなりません.

...（ピリオド３つ）　この直前の項目を必要な回数だけ繰り返すことができることを表わします.
その他の記号，カンマ，コロン，スラッシュ等は示されたとおりに入力します.例えば,＜ファイル名＞[,＜ファイル名＞]... は，ファイル名を１つ，あるいは２つ以上のファイル名をカンマで区切って入力することを表わします.

第2章　ユーティリティ・ソフトウェア・パッケージを使ったプログラム開発

ユーティリティ・ソフトウェア・パッケージはアセンブリ言語でプログラム開発を行うためのパッケージです.
このパッケージを使ったプログラム開発の説明の前に，プログラム開発についての解説で一般に使用されている用語を説明します.

2.1　用語の説明

ソースプログラム

汎用テキストエディタで作成したプログラムテキストステートメントで構成されています.

ソースファイル

ソースプログラムが入っているファイル．ソースファイルはアセンブラ，またはコンパイラの入力ファイルであり，ASCII フォーマットでなければなりません.
マクロアセンブラのソースファイルの拡張子の省略値は.MAC です.

オブジェクトプログラム（オブジェクトモジュール）

ソースプログラムをアセンブラまたはコンパイラで翻訳した結果，リロケータブルな機械語に変換されています.

オブジェクトファイル

オブジェクトプログラムが入っているファイル．オブジェクトファイルはコンパイラの出力ファイルであり，また，リンクローダの入力ファイルでもあります.
オブジェクトファイルの標準的な拡張子は.REL です.

モジュール

アセンブラ，コンパイラまたはリンクローダが扱うことのできる基本的な命令語コード群の単位です．モジュールには開発の段階や性質に応じていくつかの種類があります．アセンブラやコンパイラの翻訳結果であるオブジェクトプログラムはオブジェクトモジュールと呼びますが，その性質からはリロケータブルモジュールとも呼びます．このモジュールがリンクローダによって処理された最終的な実行可能プログラムは実行可能モジュールと呼びます.

外部参照（グローバルリファレンス）

異なるモジュール間でシンボルを参照することを外部参照(グローバルリファレンス)といいます.リンクローダは参照関係のあるモジュールを連結し，グローバルリファレンスの解決をします．また，グローバルリファレンスは他のモジュールから参照されるシンボルのことを指すこともあります.

エントリポイント　　他のモジュールが呼び出すことのできるラベル．パブリックシンボルはエントリポイントとして使うことができます．

リロケータブル　　あるモジュールをメモリ内の別の場所にリロケート(再配置)して実行することができる場合，そのモジュールはリロケータブル(再配置可能)であると言います．
リロケータブルモジュールでは，ラベルまたは変数がそのモジュールの先頭アドレスからの相対アドレスで表されています．これらのラベルや変数を“コードリラティブ”であると言い，モジュールがリンクローダによってロードされる時に絶対アドレスに変換されます．

リンクロード　　リンクローダ(MSX･L-80)がリロケータブルモジュール内のラベルや変数の絶対アドレスを計算したり，他のモジュールやランタイムライブラリを検索してグローバルリファレンスを解決したりする間の処理を言います．ロードとリンクを行なったあと，リンクローダはメモリ内にロードしたモジュールを単一のCOMファイルとして，ディスクに保存します．

2.2　ユーティリティ・ソフトウェア・パッケージを使ったプログラム開発

ユーティリティ・ソフトウェア・パッケージは，アセンブラとリンクローダの2つの重要なプログラムが中心になっています．
コンピュータ上で動作するプログラムを開発するためには，アセンブラとリンクローダの両方が必要です．
アセンブリ言語プログラムの開発の手順を図2-1に示します．
図についている番号は以下の説明にある番号と対応しています．

(1) MSX-DOS2 TOOLSパッケージ付属のエディタ，あるいは他の汎用エディタを使用してアセンブリ言語のソースプログラムファイルを作成します．

(2)ソースファイルをアセンブラを使ってアセンブルし，中間オブジェクトコードからなるオブジェクトファイルを作ります．
この中間オブジェクトコードはソースコードにくらべると機械語に近いものですが，実行することはできません．

(3)別々にアセンブルされた(1つまたは複数の)オブジェクトファイルをリンクローダを使って1本のプログラムファイルにリンクロードします.
リンクローダは中間オブジェクト・コードを，オペレーティングシステムで実行可能な実際の機械語から成るファイルに変換します.

(4)デバッグ時など必要に応じてCREF-80クロスリファレンサを使い，クロスリファレンス・リスティングファイルを出力します.
クロスリファレンス・リスティングファイルにはMSX・M-80のリスティング情報に加え，ラベルとその定義位置と参照している行のクロスリファレンス情報が出力され，デバッグに役立ちます.

(5)サブルーチンがたくさんあるプログラムなどの場合には，サブルーチンのオブジェクトモジュールをLIB-80ライブラリマネージャでライブラリファイルに集め，MSX・L-80でのコマンド入力を簡易にすることもできます.

(6)MSX・L-80で作成した実行可能プログラムはMSX-S BUG2(別売)を使ってデバッグすることができます.

(7)プログラムはオペレーティングシステムのコマンドと同じように名前をキーインするだけで実行することができます.

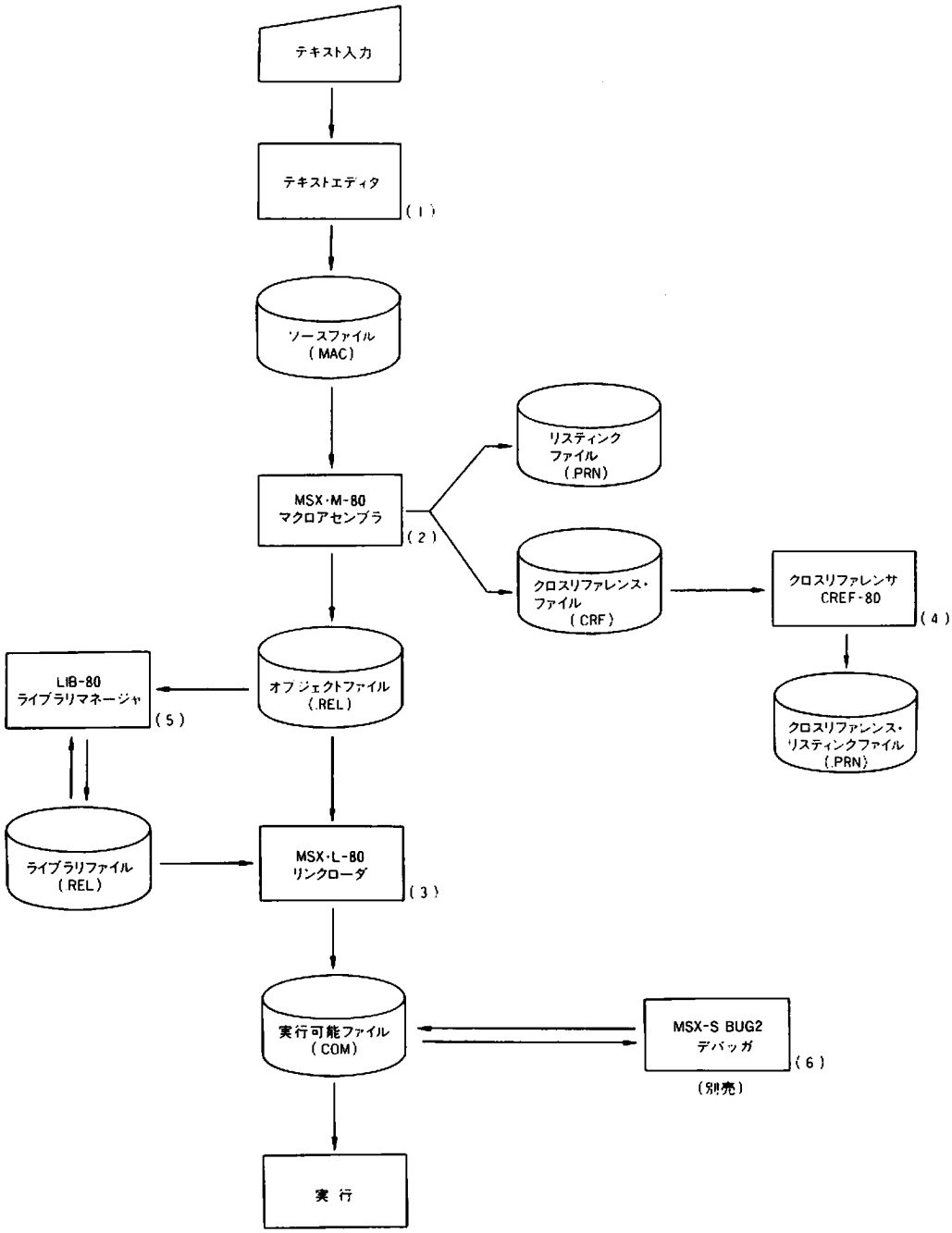


図2-1 アセンブリ言語のプログラム開発

このうち，最初の3つが全体の手順の基本的な部分です．
ソースファイルを実行可能なプログラムに変換する手順を3段階に分けることで，以下のようなかたちで，無駄な時間をなくしたりプログラムの有用性を高めることができます．

●プログラムをモジュールと呼ぶ適当な大きさに分割することができます．モジュールはそれぞれ別にアセンブルすることができるので，正しく動作しない場合，関係するモジュールだけを修正し再アセンブルします．このようにして開発時間を短縮することができます．

●モジュールのメモリ上の配置を(ある種の制限を除き)自由に行なうことができます．
あるいはリンクローダに配置をまかせることも可能です．

●モジュール間には決まったつながりはないので，アセンブルしたモジュールを他のプログラムで使ったり，もとのプログラムの変更版で使ったりすることができます．
汎用性のある使用頻度の高い処理を行なう部分を(独立したモジュールにしておけば)再コーディングする必要がなくなります．

アセンブルしたモジュールをまとめて実行可能なプログラムにするにはリンクローダを使います．まとめたいモジュールを単にリンクローダに指示すれば，リンクローダはモジュールをメモリ上に次々にロードします．さらに加えて，リンクローダではあるモジュールの命令と，別のモジュールの命令との間を直接結びつけることができます．
この結びつける(リンクする)というのは，あるモジュールの中のある値(普通はプログラムのアドレス)を，別のモジュールの中の，ちょうどそれが必要なところで使えるようにするということです．

リンクローダはモジュール同士のリンクを行ないますが，そのためにリンクローダにリンクが必要なことを知らせる指示を与えなければなりません．この指示のことをシンボル，正確にはエクスターナル(外部参照)シンボルおよびパブリックシンボルと呼びます．
外部参照シンボルは，リンクローダに対してプログラムのその位置に他のモジュール中の値をリンクしたいことをつたえます．リンクされる値はパブリックシンボルで与えます．
これはリンクローダを正しいモジュールと命令に導くための指示です．リンクローダはそこでパブリックシンボルの値を外部参照シンボルにリンクし，それが終わるとモジュールのロードを続行します．

●アセンブラではモジュールそれぞれに異なったモードを与えることができます．さらに1つのモジュール中でもモードを使い分けてアセンブルすることができます．
モードには絶対(Absolute)モード，データ相対(Data-relative)モード，コード相対(Code-relative)モード，コモン相対(Common-relative)モードの4つがあります．

絶対モードは単純なアセンブラが生成するコードのように，メモリ上の固定アドレスに配

置されるコードを生成します．それ以外の3つのモードは，これとはまったく違って，モジュールをメモリ上の任意の位置に配置できるようにするためのものです．3つの相対モードはそれぞれ異なったセグメントにアセンブルされます．それぞれのセグメント内でのアドレスは相対アドレスです．
これは，セグメント中の最初のバイトには相対アドレス0が与えられ，次のバイトには1が与えられる，という意味です．

リンクローダはモジュールをロードする時に，セグメント内での相対アドレスをメモリ上の絶対アドレスに変換します．相対アドレスはリンクローダが決める固定アドレスからのオフセットです．MSX-DOSの場合，最初にロードされるモジュールではこのアドレスは103Hです．
従って，最初のモジュールの相対アドレスは103Hからのオフセットです．その次のモジュールは最初のモジュールのすぐ後ろにロードされ，相対アドレスは最初のモジュールの直後のアドレスからのオフセットになります．後続のモジュールも同様にロード(およびオフセット)されます．
最初のモジュールに対するこのデフォルトの開始アドレスは，リンク時に変更することができます．その場合，相対アドレスは，指定した固定アドレスからのオフセットになります．

相対アドレス方式の影響の1つは，ORGステートメントが(他の多くのアセンブラと)異なった性質を持つということです．相対セグメントでは，ORGステートメントはオフセットを指定し，(たいていのアセンブラや，絶対モード中でのように)固定アドレスを指定するわけではありません．
従って，相対セグメントにORGがあると，それに続くコードをロードする前に，指定された数だけのアドレスをとばしてしまいます．3.6.3のORGステートメントの解説を注意して読んで下さい．

アセンブラが絶対アドレスではなく相対アドレスを出力しているため，モジュールのメモリ上の配置を，ある種の制限(第5章参照)を除いて調整できるわけです．「モジュールのメモリ上の配置を調整できる」ことを再配置可能(リロケータブル Relocatable)であるといいます．
3つの相対セグメントに対してアセンブラが生成する中間コードは再配置可能コードと呼びます．
アセンブルされたモジュールの拡張子が.RELであり，そのファイルがRELファイルと呼ばれているのはこれに由来しています．

各モードはそれぞれ違った用途に使います．絶対モードには特定のメモリアドレスに置きたいコードを入れます．各相対モードは分かれたセグメントとしてメモリ上にロードされます．
データ相対セグメントはデータ項目やたびたび変更されるコードなどのRAM上にしか置

けないコードを含めます．コード相対セグメントは変更することのないコードを含め，ROMやPROMに適しています．コモン相対セグメントは1つ以上のモジュールで共有することのできるデータ項目を含めます．

これらのモード中にあるソースステートメントはそのモードの性質を受け継ぎます．
ステートメント中のシンボルや式は，アセンブラによって，そのシンボルが位置するステートメントのモードや，データのタイプやシンボルを定義したり式の一部を構成したりするその他のエントリに応じて評価されます．シンボルや式に与えられたモードの属性もまたそれのモードと呼びます．すなわち，シンボルあるいは式は，絶対，データ相対，コード相対，あるいはコモン相対のモードを属性として持ちます．モードのこういった概念は柔軟さとともに複雑さの原因ともなっているので重要です．もし，ソースステートメントをすべて絶対モードでアセンブルしたら，シンボルや式はすべて絶対値を持ち，このようなシンボルや式を扱うのは比較的容易です．
絶対モードでの問題は，再配置が複雑な手間暇のかかるやり方でしか可能でないことです．絶対モードは，コードを再使用する可能性を確実に減らします．

相対モード(コード，データ，コモン)は再配置を可能にし，つまりはモジュールの取り扱いを柔軟にするための基礎となります．問題は，相対シンボルや相対式を評価するのがずっと難しいことです．実際，アセンブラはモジュールをアセンブルするためにソースステートメントを2回のパスで処理しなければなりません．最初のパスでは，ステートメントを評価しマクロの展開を行なって，それが生成するコードの大きさを計算し，シンボルテーブルを作成します．
シンボルテーブル内では現われた全てのシンボルに値(と属性と)が割り当てられます．
次のパスでは，アセンブラは，シンボルや式の値をシンボルテーブルを参照して評価し，またマクロの展開も行なって，実際のコードを生成し，RELファイルに中間コードをはきだします．

RELファイルがリンクローダに渡されたとき，複数のセグメントは一緒にリンクされ固定のメモリアドレスにロードされます．相対アドレスは(ロードアドレスにオフセットを加算して)絶対アドレスに変換されます．相対セグメントは，コモン相対，データ相対，コード相対の順番で固定アドレスからロードされます．この固定アドレスのデフォルトは103Hです．(アドレスの100Hから102Hは，最初に実行する命令へのジャンプのために使われます．通常これはコードセグメントの先頭です．)

リンクローダでモジュールのリンクとアドレスの割当が終わると，その結果はオペレーティングシステムで実行可能なファイルとしてセーブすることができるようになります．プログラムの実行は，オペレーティングシステムのコマンドと同じように簡単に行えます．そのため，このリンクロードの終わったファイルはコマンドファイルと呼ばれます．

以上が，ユーティリティ・ソフトウェア・パッケージの全般的な概念です．パッケージ全体の解説は2.3に概略が，また，各章に詳細な解説があります．

2.3　ユーティリティ・ソフトウェア・パッケージの特徴

2.3.1　2つのアセンブリ言語

MSX・M-80は8080/Z80の両方のニモニックをサポートします．

2.3.2　リロケータビリティ

MSX・M-80はリロケータブルなコードのモジュールを作成することができます．また，多くのアセンブラと同様に絶対コードを作成することもできます．リロケータビリティの利点は，実行時の配置位置を意識せずにプログラムをモジュールごとにアセンブルすることができるということです．さらに，第5章に記述されているいくつかの制約を除いてそのモジュールをメモリ内の任意な位置におくことができます．

リロケータブルモジュールはコーディングが容易で，テスト，デバッグ，および修正が早いという利点を持っています．また，あとでRAMまたはROM/PROMにロードするオブジェクトコードのセグメントを指定することも可能です．リロケータビリティについては3.2.4　モード属性の項で詳しく述べます．

2.3.3　マクロ機能

MSX・M-80はインテル社の標準マクロ機能をサポートします．マクロ機能によって，頻繁に使用する命令セットのブロックに名前を定義し，命令セットの代わりに名前を書くことができます．この名前を書くことによって，同じ命令セットを何回もくり返しコーディングする必要がなくなります．

この命令セットのブロックをマクロと呼び，名前をつけることをマクロ定義と呼びます．
マクロ定義はマクロを使用する前にソースプログラム上で行いますが，ディスク上の別のソースファイルにマクロ定義をしておき，INCLUDE疑似命令で挿入するようにすると，マクロ定義を複数のモジュールで共用できるので便利です．プログラム中で命令セットが必要な場合はマクロ名だけを指定して，マクロを呼び出します．MSX・M-80はマクロ名の指定があると，マクロ定義しておいた命令セットをソースプログラム上に展開した形でアセンブルを行います．マクロの使用によりソースプログラムのサイズを減少することができます．

また，マクロの呼び出し時に，マクロの展開に使用するパラメータをアセンブラに対して指定することもできます．

マクロはネストする(重ねる)ことができます．

すなわち，マクロは他のマクロ内から呼び出すこともできます．
マクロのネスティングはメモリのサイズによってのみ制限を受けます．

2.3.4　条件付きアセンブル

MSX・M-80 は条件付きアセンブルをサポートします．プログラマはプログラムのどの部分をアセンブルするか，あるいはアセンブルしないかの条件を指定することができます．条件付きアセンブル機能はアセンブルパスのテスト，シンボル定義，およびマクロへのパラメータなどの条件付き疑似命令によって拡張されています．条件は 255 レベルまでネストが可能です．

2.3.5　リンクローダ

MSX・L-80 リンクローダは，アセンブルされたモジュール(REL ファイル)を実行可能なモジュール(COM ファイル)に変換するために使います．REL ファイルは実行可能ではありません．
MSX・L-80 はまた，次のことを行なうのにも使うことができます．

●1 つまたは複数のモジュールのロード，リンク，及び実行．

●再配置可能なモジュールの指定のアドレスへのロード．

●プログラム領域とデータ領域の分離．

これらの処理を行なう際に，MSX・L-80 はモジュール間の外部参照を解決し(別のプログラムあるいはモジュールで定義されている値を参照，あるいは呼び出しいるプログラムはリンク時にしか解決できない外部参照を行なっています)，オペレーティングシステムで実行可能な(COM)ファイルをディスクにセーブします．

これらの機能は，アセンブルしたモジュールをライブラリと一緒にリンクして高級言語の実行時ライブラリにルーチンを追加したり，アセンブル言語のプログラムでライブラリを使用したりすることを可能にします．

2.3.6　クロスリファレンサ

CREF-80 クロスリファレンサはアセンブラの作成したクロスリファレンスファイルを処理します．
その結果は，プログラムのデバッグに役立つクロスリファレンスリストです．

2.3.7　ライブラリマネージャ

LIB-80 はユーティリティ・ソフトウェア・パッケージおよび高級言語の実行時ライブラリの管理を目的に作られています．LIB-80 を使って，専用のアセンブリ言語のサブルーチンライブラリを作ることもできます．

LIB-80 は，高級言語や他のアセンブリ言語のサブルーチンとなる，アセンブリ言語のプログラムからライブラリを作成します．

LIB-80 では，個別に作ったモジュールや，既存のライブラリ中のモジュールを扱うことができ，目的にあった特別のライブラリを作成することができます．

第３章　MSX・M-80マクロアセンブリ言語

この章ではMSX・M-80アセンブラのソースファイルを作成するために知っておかなければならないことについて解説します.
ソースファイルはテキストエディタを使用して作成します.
ユーティリティ・ソフトウェア・パッケージにはテキストエディタのプログラムは含まれていません.

ソースファイルは第4章「MSX・M-80の実行」で解説する方法でアセンブルします.

3.1　ソース・ファイルの構成

3.1.1　ファイル構成

MSX・M-80アセンブラのソースファイルはアセンブリ言語で書かれた一連のステートメント行です. ファイルの最後の行は, ENDステートメントでなければなりません. 対応しているステートメント(たとえば, IF...ENDIFなど)は, 正しい順序で入力しなければなりません.
それ以外の行は, プログラマが考えたとおりの, 任意の順序で書くことができます.

●各行は最大132文字の長さまで受け入れることができます.

●小文字でタイプされた全てのシンボル, 命令コード, 疑似命令コードは大文字に変換されます. ただし引用符で囲まれた文字列や注釈は小文字から大文字への変換はされません.
またMSBを0にするようなことはしないため, カナ文字, 漢字等も使用できます.

3.1.2　ステートメント行の形式

ステートメント行はフィールドに分けられます.
その形式は次のとおりです.

シンボルフィールド	命令フィールド	アーギュメントフィールド	コメントフィールド

例	BUF:	DEFS	1000H	; create a buffer
	↓	↓	↓	↓
	シンボルフィールド	命令フィールド	アーギュメントフィールド	コメントフィールド

シンボルフィールド　　このフィールドにはシンボルを記述します.
シンボルの直後にはEQU, SET, DEFL, ASETまたはMACROステートメントを除いてコロン(:)を付けます.

命令フィールド　　このフィールドには，命令コード，疑似命令コード，マクロ名，あるいは式を記述します．

アーギュメントフィールド　このフィールドには式(定数，変数，レジスタ名，オペランドおよび演算子)を記述します．

コメントフィールド　　このフィールドの先頭にはセミコロン(；)を置き，以降にコメントテキストを記入します．

各フィールドはいずれも省略することができます．完全な空行を入力することもできます．

ステートメントは任意の桁から開始することができます．空白またはタブをフィールド間に適当に挿入してステートメントを読みやすくすることができますが，各フィールド間には少なくとも１つの空白またはタブが必要です．

3.1.3　コメント

MSX･M-80 アセンブラのソース行は基本的には命令とその引数(アーギュメント)です．そのため，コメントは常にセミコロンで始める必要があります．コメントは行末で終ります．

長いコメントの場合，各行ごとにセミコロンを入力するかわりに.COMMENT 疑似命令を使用することができます．.COMMENT については，3.6.4　ファイル関連の疑似命令を参照してください．

3.2　シンボル

シンボルは特定の機能や値につけた名前です．シンボル名はプログラマが決め，定義します．

MSX･M-80 のシンボルはその機能に応じてラベル，パブリック，エクスターナルの３つの種類に分けられます．３種類のシンボルはいずれも，対応するモード属性(3.2.4 参照)を持っています．シンボルは次の規則に従います．

●シンボルの長さは任意です．ただし，有効文字数はシンボルの種類によって次のように異なります．

a.ラベルでは先頭の 16 文字までが有効です．

b.パブリックおよびエクスターナルシンボルでは，先頭の６文字までがリンクローダに渡されます．

余った文字は内部で切り捨てられます.

●シンボルに使える文字は次のとおりです.

A～Z　0～9　$　.　?　@　_

●シンボルは数字で始まっていてはいけません.

●シンボルを読みとるときには，小文字は大文字に変換します．名前は大文字，小文字あるいはその両方で入力することができます.

3.2.1 ラベル

ラベルは，それを定義しているステートメントをプログラム中の別のステートメントから参照するために使います．機械語命令やデータ定義命令のシンボルフィールドに名前を書くと，その名前に領域のアドレスが値として与えられます.

例
```
BUF:   DS   1000H
```

BUF は 1000H バイト確保された領域の先頭アドレスを値として持ちます.

ラベルを定義すると，そのラベルはアーギュメントフィールドの項目として使用することができるようになります．アーギュメント中にラベルのあるステートメントは，そのラベルがシンボルフィールド中にある，つまりラベルが定義されているステートメントを参照し，定義された値でアーギュメントを置き換えます.

例
```
LD   (BUF),A
```

この例ではアキュームレータの値を BUF というラベルで表される領域にストアします.

ラベルはシンボルの規則に従う 16 文字までの名前です．ラベルを定義する場合，それはステートメントの最初の項目で，直後に(間を開けずに)コロンを 1 つ付けなければなりません．(コロンを 2 つ付けるとパブリックシンボルとなります．3.2.2 を参照して下さい.)

またラベルは，EQU 疑似命令や SET 疑似命令を使っても定義することができます．3.6.2　データ定義およびシンボル定義の項を参照して下さい.

3.2.2　パブリックシンボル

パブリックシンボルはラベルとほとんど同じように定義/参照することができます．
ラベルとの違いは，パブリックシンボルが他のプログラムからも参照できるところにあります．
パブリックシンボルの宣言方法は次の2通りあります．

●シンボルの直後にコロンを2つ付ける．

例
```
FOO::   RET
```

●疑似命令のPUBLIC，ENTRY，またはGLOBALで宣言する．

例
```
PUBLIC  FOO
```

これらの疑似命令については3.6.2　データ定義およびシンボル定義の項を参照してください．

上記2つの宣言方法の結果は同じです．従って，次の2つの例は同じ意味です．

例
```
FOO::   RET
```

例
```
PUBLIC  FOO
FOO:    RET
```

3.2.3　外部参照シンボル

外部参照シンボルはそれを宣言しているモジュールの外部で定義されているシンボルです．外部参照シンボルは別のモジュールではパブリックシンボルとして定義されています．外部参照シンボルには，リンク時に，他のモジュールのパブリックシンボルの値が与えられます．

例　モジュール1
```
        EXTRN   FOO     ; EXTERNAL FOO
        CALL    FOO
```

モジュール2
```
FOO::   RET     ; PUBLIC FOO
```

リンクローダはリンク時に，モジュール1の外部参照シンボルFOOの値としてモジュール2のパブリックシンボルFOOの値(ステートメントのアドレス)を使用します．

外部参照シンボルの宣言方法は次の3通りです.

●シンボルの参照の直後にハッシュ記号2つ(##)を入れる.

例 CALL　FOO ##

FOO は他のプログラムで定義された2バイトの値を持つシンボル(通常はプログラムアドレス)であることを宣言します.

●疑似命令 EXT, EXTRN, または EXTERNAL のいずれかで, 2バイトの値であることを宣言する.

例 EXTRN　FOO

FOO は他のプログラムで定義された2バイトの値を持つことを宣言します.

●疑似命令 BYTE EXT, BYTE EXTRN, または BYTE EXTERNAL のいずれかで, 1バイトの値であることを宣言する.

例 BYTE EXT FOO

FOO は他のプログラムで定義された1バイトの値として宣言します.

これらの疑似命令については, 3.6.2　データ定義およびシンボル定義の項を参照してください.

これらの宣言の結果は同じです. 従って, 次の2つの例は同じ意味です.

```
例 CALL      FOO ##

例 EXT       FOO
   CALL      FOO
```

3.2.4 モード属性

シンボルは, アーギュメントフィールドに名前を書くことで参照します. シンボルを参照すると, シンボルの名前に代わって(シンボルを定義したステートメントから得られる)シンボルの値が演算で使われます.

シンボルはプログラムカウンタ(PC)モードに従って評価されます.
PCモードが絶対モードならば絶対アドレスで,コード相対モード,データ相対モードならば相対アドレスで,また,コモン相対モードならば他のモジュールと共用するアドレスがそれぞれ設定されます.モードの指定がない場合はコード相対モードを仮定します.

MSX・M-80では1つのモジュールを必要に応じて次の4つのプログラムカウンタ・モード(PCモード)に分けてアセンブルすることができます.疑似命令の詳細は3.6.3　PCモードの項を参照して下さい.

●絶対モード(Absolute)

疑似命令：ASEG
絶対モードは実行時のメモリ上の配置位置を特定のアドレスに固定するプログラム部分について指定します.絶対モードのシンボルおよび式は絶対値を持ち,非リロケータブルです.
絶対モードのセグメント中のコードは非リロケータブルコードです.

●データ相対モード(Data-relative)

疑似命令：DSEG
データ相対モードは実行時にメモリに領域を書き換える可能性があるプログラム部分について指定します.したがって,データ相対モードのセグメントはRAM上にロードします.
データ相対モードはプログラムのデータ領域に適しています.
データ相対モードのシンボルはリロケータブルです.

●コード相対モード(Code-relative)

疑似命令：CSEG
コード相対モードは実行時にメモリ領域の書き換えの可能性がないプログラム部分について指定します.したがってROM/PROM上に置くこともできます.
コード相対モードはプログラムのコード領域に適しています.
コード相対モードのシンボルはリロケータブルです.

●コモン相対モード(Common-relative)

疑似命令：COMMON
コモン相対モードは他のプログラムとの共通データ領域の部分について指定します.
コモン相対モードのセグメントは他のプログラムと共用することができます.

相対モードのプログラム部分はリロケータブルであり,メモリの配置について柔軟なプログラ

ミングをすることができます.

アセンブル時には4つの各モードに対応したセグメントごとにオブジェクトコードを生成します.

また,リンクロードでは,指定がないかぎりデフォルトのアドレス103Hからコモン相対セグメント,データ相対セグメント,コード相対セグメントを順に配置し,相対セグメント内に含まれている相対アドレスを絶対アドレスに変換します.アドレス100H～102Hにはプログラム開始アドレスへのジャンプ命令がストアされます.

複数のモジュールをリンクする場合には各モジュールごとに同じモードのセグメント同士を統合します.また,コード相対セグメントとデータ相対セグメントはMSX・L－80のスイッチ/Pおよび/Dを使ってセグメントを任意のアドレスに配置することもできます.

3.3　命令コードおよび疑似命令

命令コード(オペコード)は機械語命令に対応するニモニック名です.疑似命令は,マイクロプロセッサではなく,アセンブラに対する指示です.

MSX・M-80は8080/Z80の2つの機械語命令セットをサポートします.命令コードおよびその機能の簡単な要約が付録Cに記載してあります.8080/Z80のうちのどちらかの命令セットでプログラムを作成するかは,疑似命令で選択することができます.また,プログラムの途中から,異なる命令語セットにすることができます.詳細については3.6.1　命令セットの選択を参照して下さい.

MSX・M-80は,また,アセンブラに種々の機能を指示するたくさんの疑似命令をサポートします.

疑似命令の詳細は3.6に記しており,また,付録Bに要約したものを記載しています.

マクロ名はマクロ定義ブロックに付けた名前です.頻繁に使用する命令セットのブロックをマクロ定義しておきマクロ名を指定するだけで呼び出すことができます.

命令フィールドの項目は次の順番で評価されます.

●マクロ名
●命令コード/疑似命令
●式

命令フィールドの項目は上記の順序で評価されますので,命令コードや疑似命令と同じ名前のマクロ定義を作成するとマクロ定義の方が優先され,そのマクロ定義以降では同じ名前の命令コードや疑似命令を使うことができません.たとえば,マクロ名がADDというマクロ定義をする

と，命令コードのADDを使用することができません．

ステートメント行に有効なマクロ名あるいは命令コード/疑似命令がなく，式のみがある場合でもアセンブラはエラーを表示しません．この場合，アセンブラは式の前にバイト定義疑似命令があるものと仮定し，その行を処理します．

3.4　アーギュメント(式)

命令コードと疑似命令のアーギュメントには式を記述します．式は，たとえば5+4*3のような数式に似ています．式はオペランド(たとえば数式内の5，4，3)および演算子(オペレータ，たとえば数式内の+，*)から構成されます．式は1つ以上のオペランドを含み，2つ以上のオペランドを含む場合はオペランドは演算子によってたがいに関係づけます．

例　ADDR　7*2　6-3　8/7　ADDR+3

以下にオペランドとオペレータの形式を説明します．

3.4.1　オペランド

数値

数値のデフォルトの基数は，10進数です．基数は.RADIX疑似命令によって2進から16進までの任意のベースに変更できます．基数が10より大きい場合には9の次の数にはA～Fを使用します．

数値の最初の文字が数字でない場合には，その数の前に0がなければなりません．

数値は次の表記法が使用されていない限り，その時点で設定されている基数を使用します．

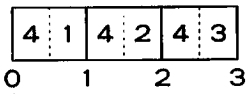
nnnnB　　2進
nnnnD　10進
nnnnO　　8進
nnnnH　16進
X'nnnn'　16進

数値は16ビットの符号なしの2進数です．65535(16ビットで表わすことができる最大値)を超える数のオーバフローは無視され，その結果は16ビットで表される範囲内の数になります．

ASCIIストリング

ストリング(文字列)はゼロまたは1個以上の文字により構成され，引用符で囲みます．引用符は，シングル(')またはダブル(")のいずれかを使用します．引用符で囲まれたストリングがアーギュメントとして入力されたとき，引用符内の文字が順にメモリにストアされます．

例　DB　　"ABC"



上の例では先頭のアドレスにAのASCIIコード(41H),2番目のアドレスにB(42H),3番目のアドレスにC(43H)がストアされます.

また,シングルクォーテーション(')で囲んでいるストリング中には,ダブルクォーテーション(")をストリングに一部として入れておくことができ,ダブルクォーテーション(")で囲んでいるストリング中には,シングルクォーテーション(')をストリングの一部として入れておくことができます.

例　"I am 'great' today"
　　'I am "great" today'

区切りとして使われている引用符も,それが必要な部分に2つの引用符を並べて文字として用いることが出来ます.

例　"I am ""great"" today"

上の例は次のようにストアされます.

I am "great" today

引用符に文字がない場合,ストリングはヌルストリングとして評価されます.

文字定数

文字定数はストリングに似ていて,0～2文字のASCII文字より構成され引用符で囲みます.引用符はシングルでもダブルでも使用できます.区切りの引用符も,2つ並べると文字として用いることが出来ます.

ストリングとの違いは次のとおりです.

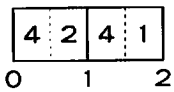
●文字定数は文字数が0～2個です.

●式が2つ以上のオペランドをもっている場合,引用符で囲まれた文字は文字定数として扱われます.引用符で囲まれた文字が式中唯一のオペランドである場合,その文字は,ストリングとして扱われます.

例 'A'+1 ・・・・ 文字定数
'A' ・・・・ ストリング

●文字定数の値は計算され，結果は最初のアドレスに下位バイトが，2番目のアドレスに上位バイトがストアされます．

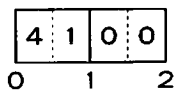
例 DEFW 'AB'+0



上の例では文字定数が4142Hと評価され，最初のアドレスに42Hを，2番目のアドレスに41Hをストアします．

1文字の文字定数はその文字のASCIIコードに対応する値を持っています．文字定数の上位バイトはゼロであり，下位バイトは文字のASCIIコードに対応する値です．たとえば文字定数'A'の値は41Hです．

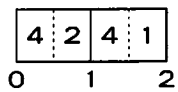
例 DEFW 'A'



上の例では，メモリ上に図のような2バイトの値が確保されます．なお，アセンブルリストのオブジェクトコード欄では，上位バイト，下位バイトの順に表示されます．

2つの文字の文字定数は，上位バイトに最初の文字のASCIIコードを持っています．たとえば文字定数'AB'+0の値は41H*256+42H+0です．

例 DEFW 'AB'+0



ASCII文字の10進数及び16進数の値を付録Aに記載しています．

式内のシンボル

シンボルは式内のオペランドとして使用することができます.
シンボルは評価され，シンボルの持つ値がシンボルに代わって置き換えられます.

●オペレーションはシンボルの値を使用して行われます.
オペランドとしてシンボルを使用する利点は，ジャンプ先やデータ領域のアドレス値を覚えておく必要がなく，シンボルの名前を覚えていればよいということです.
シンボルの名前付けは覚えやすく意味を持った綴りが便利です.
プログラム内の参照ポイントの数が多い場合，オペランドとしてシンボル名を使えることは大きな魅力です.

●式中に外部参照シンボルを使用するための規則

a.外部参照シンボルは次の演算子を式中に使用することができます.

＋　－　＊　／　MOD　HIGH　LOW

b.外部参照シンボルが式中で使用されていた場合は，その式の結果は外部参照となります.

●式中のシンボルに関するモードの規則

a. オペレーションが AND，OR，または XOR を除いて，オペランドは任意のモードでもかまいません.

b. AND，OR，XOR，および SHR の場合，両オペランドは絶対モードかつ内部参照でなくてはなりません.

c. 絶対モードのオペランドと別のモードのオペランドを含む場合，その式は(絶対モードでない)もう一方のモードとなります.

d. 異なるモードで2つのオペランドを減算した場合，式は絶対モードとなり，おなじモードの場合，式はそのオペランドのモードとなります.

e. データ相対シンボル及びコード相対シンボルを加えた場合，式の値は未定です.
MSX･M-80 はその式を未解決のものとして MSX･L-80 に渡し，MSX･L-80 が解決します.

カレント・プログラムカウンタ・シンボル

アーギュメントフィールドにしか書けないシンボルとして，カレント・プログラムカウンタ・シンボルというものがあります．カレント・プログラムカウンタは，次にアセンブルする命令のアドレスです．カレント・プログラムカウンタは次の命令の参照点として便利なことがよくあります．カレント・プログラムアドレスを記憶して置くか，計算するかわりに$というシンボルを使って現在のプログラムのアドレスの値を使用することができます．

例
```
JP    $+6
A     EQU $
```

オペランドとしての8080命令コード

8080命令コードは，8080モードでのみ有効な1バイトのオペランドとなります．命令コードはアセンブル中に命令コードのバイト値として評価されます．

オペランドとして8080命令コードを使用するためには，まず．8080疑似命令をセットします．命令コードの先頭1のバイトのみがオペランドとして有効です．オペランドとして用いる命令コードが2バイト以上を生成する命令の場合，かっこで囲んで命令コードの先頭1バイトを生成するようアセンブラに指示して下さい．

例
```
MVI     A,(JMP)
ADI     (CPI)
MVI     B,(RNZ)
CPI     (INX H)
ACI     (LXI B)
MVI     C,(MOV A,B)
```

オペランド内に(かっこ内に)2バイト以上のコードが生成されるような命令が含まれると，たとえば，(CPI 5)，(LXI B,LABEL1)，または(JMP LABEL2)などの場合はエラーとなります．

通常1バイトを生成する命令コードはかっこで囲まなくてもオペランドとして使用することができます．

3.4.2　演算子(オペレータ)

MSX・M-80 では算術演算子および論理演算子の両方を使うことができます．論理演算子の計算結果は，その値が 0 以外の数値のときに真(True)と，また 0 の時に偽(False)として取り扱われます．
論理値のみを返す論理演算子は真の値に-1 を偽の値に 0 を返します．次のオペレータを式中で使用することができます．

オペレータ定義

NUL

アーギュメント(パラメータ)がヌルの場合真を返します．NUL の後の，同一行の残り部分が NUL のアーギュメントとして使用されます．

例　IF　NUL　<アーギュメント>

この条件はアーギュメントの最初の文字がセミコロンまたは行末以外のものであった場合に偽となります．上の条件は IFB や IFNB の機能と同じですが，IFB の方が使用法が簡単です．
(3.8 の条件アセンブル機能の項を参照してください．)

TYPE

TYPE オペレータはアーギュメントのモードと，外部参照であるかそうでないかの情報を含むバイトを返します．TYPE のアーギュメントは任意の式(ストリング，数値，論理)です．その式が無効の場合 TYPE はゼロを返します．返されるバイトは次のとおりです．

●下位 2 ビットはモードを表わします．

00(0)　絶対モード
01(1)　コード相対モード
10(2)　データ相対モード
11(3)　コモン相対モード

●最上位ビット(80H)は外部参照ビットです．このビットがオンの場合式には外部参照が含まれます．また上位ビットがオフの場合式は内部参照となります．

●式がモジュール内で定義されている場合は 20H のビットがオンです．
このビットは式が内部で定義されている場合はオンであり，式が定義されていないか外部参照の場合はオフです．全部のビットがオフの場合その式は無効です．

TYPE は，通常マクロの中で，アーギュメントの種別を調べる必要のあるようなときに使用します．例えば，条件アセンブリなどでプログラムの流れを変えることができます．

例
```
FOO     MACRO   X
        LOCAL   Z
Z       SET     TYPE X
        IF      Z...
```

TYPE は X のモードおよびタイプをテストします．X の評価によって IF Z... で始まるコードのブロックをアセンブルする場合と，アセンブルしない場合とに分けることができます．

LOW

16 ビット値の下位 8 ビットを分離します．

HIGH

16 ビット値の上位 8 ビットを分離します．

*****

乗算を行います．

/

除算を行います．

MOD

剰余の計算をします．左オペランドを右オペランドで割り，その残り(剰余)の値を返します．

SHR

右シフトを行います．
SHR に続く整数の数だけ右へシフトが行われます．

SHL

左シフトを行います．SHL に続く整数の数だけ左へシフトが行われます．

-(負号)

負号を表わすオペレータ演算子です．このオペレータに続く整数は負の値であることを示します．

+

加算を行います．

－

左オペランドから右オペランドを減算したものを返します．

EQ

等号の条件判断．両オペランドが等しい場合に真を返します．

NE

不等号の条件判断．両オペランドが等しくない場合に真を返します．

LT

左オペランドが右オペランドより小さい場合に真を返します．

LE

左オペランドが右オペランドより小さいかあるいは等しい場合に真を返します．

GT

左オペランドが右オペランドより大きい場合に真を返します．

GE

左オペランドが右オペランドより大きいかあるいは等しい場合に真を返します．

NOT

1)否定．オペランドが偽である場合，真を返します．オペランドが真である場合，偽を返します．

2)オペランドの全ビットを反転した値を返します．

AND

1)論理積．両オペランドが真の場合に返します．いずれかのオペランドが偽であるか，両オペランドが偽の場合に偽を返します．両オペランド共に絶対値でなければなりません．

2)左オペランドと右オペランドの論理積を返します．

OR

1)論理和．いずれかのオペランドが真であるか，両オペランドが真である場合に真を返します．両オペランド共に絶対値でなければなりません．

2)左オペランドと右オペランドの論理和を返します．

XOR

1)排他的論理和．いずれかのオペランドが真であり他方が偽である場合に真を返します．

両オペランドが共に真であるか，両オペランドが共に偽である場合に偽を返します．両オペランドは共に絶対値でなければなりません．

2)左オペランドと右オペランドの排他的論理和を返します．

オペレータ演算子の優先順位は以下のとおりです．

NUL，TYPE
LOW，HIGH
*，/，MOD，SHR，SHL
負号
＋，－
EQ，NE，LT，LE，GT，GE
NOT
AND
OR，XOR

優先順位の高い演算子を含む部分式が最初に計算されます．優先順位は優先したいと望む式の部分をかっこで囲むことによって変更することができます．

＋，－，*，および/を除くすべてのオペレータは，1つ以上の空白を入れてオペランドと分離しなければなりません．

3.5　アセンブラの機能

MSX・M-80マクロアセンブラは単一機能疑似命令，マクロ機能，および条件アセンブル機能の3つの一般的な機能を備えています．

3.6　単一機能疑似命令

単一機能疑似命令は，それ自身のステートメント行だけに関係し，アセンブラに一つの機能だけを実行するよう指示します．(マクロおよび条件付きアセンブルは，これに対し複数の命令行が関係してきますので，ブロック疑似命令と考えることができます．)

単一機能疑似命令は，次の5つのタイプに分類することができます．

●命令セット選択
●データ定義/シンボル定義
●PCモード指定
●ファイル関連
●リスティング制御

3.6.1　命令セットの選択

MSX・M-80 ではアセンブルしようとするプログラムが Z80 命令セットを使用するか，8080 命令セットを使用するかを選択することができます．

命令セットのデフォルトは Z80 です．8080 の命令セットを使用するプログラムでは，プログラムの先頭で 8080 命令セットを使用することを疑似命令によって宣言しなければなりません．
本章に記述のある疑似命令はどちらの命令セットでも使用できることに留意して下さい．

命令セット選択疑似命令に続いて入力されるすべての命令コードは，別の命令セット選択疑似命令に出会うまで指定した命令セットの命令コードとしてアセンブルされます．選択した命令セットにない命令コードを入力するとエラーとなります．

命令セット選択疑似命令は次のとおりです．

.8080
.Z80

.8080(8080 モード選択)

機　能　8080 命令セットモードにします．

書　式　.8080

文　例　.8080

解　説　●8080 命令セットモードでアセンブルするよう MSX・M-80 に指示します．
●アーギュメントはありません．

.Z80(Z80 モード選択)

機　能　Z80 命令セットモード(デフォルト)にします．

書　式　.Z80

文　例　.Z80

解　説　●Z80 命令セットモードでアセンブルするよう MSX・M-80 に指示します．
●アーギュメントはありません．

3.6.2　データ定義およびシンボル定義

データ定義およびシンボル定義疑似命令はSET疑似命令を除いて，8080/Z80のどちらの命令セットモードでもサポートされます．SET疑似命令はZ80モードでは使用できません．データ定義およびシンボル定義の疑似命令は次のとおりです．

DB/DEFB/DEFM
DC
DS/DEFS
DW/DEFW
EQU
EXT/EXTRN/EXTERNAL
BYTE EXT/BYTE EXTRN/BYTE EXTERNAL
ENTRY/GLOBAL/PUBLIC
SET/DEFL/ASET

参考のために，それぞれの命令セットでどの疑似命令が普通使われるかを示すために，疑似命令の前に以下の印がつけてあります．

* Z80の疑似命令

アスタリスク(*)のない場合は，インテル8080の疑似命令．

DB/DEFB/DEFM(バイト定義)

機　能　アーギュメントで指定したストリングまたは式の値を持つ1バイトずつのデータ領域を確保し，初期値を設定します．

書　式
```
DB      <式> [,<式>]...
*DEFB   <式> [,<式>]...
DB      <ストリング> [,<ストリング>]...
*DEFM   <ストリング> [,<ストリング>]...
```

文　例
```
DEFB    'AB'
DEFB    'AB' AND 0FFH
DEFB    'ABC'
```

アセンブル結果は次のとおりです．

```
0000'  41 42          DEFB    'AB'
0002'  42             DEFB    'AB' AND 0FFH
0003'  41 42 43       DEFB    'ABC'
```

解　説
- DEFBはアーギュメントとして式を指定する場合に使用します．また，DEFMはアーギュメントとしてストリングを指定する場合に使用します．
- アーギュメントで指定した値は現在のロケーションカウンタの位置から1バイトずつストアされます．
- <式>の値は1バイトで表わすことができる値(-256～255，上位バイトが0または255)でなければなりません．それ以外は“A”エラーとなります．
- 3文字以上のストリングは<式>の中で使うことはできません．(即ち，カンマ(,)で次に続けるか，行末で終わっていなければなりません．)
- ストリング中の文字はその順序に従って1バイトずつストアされます．

DC(文字定義)

機　能　**アーギュメントで指定したストリングのデータ領域を確保して初期値を設定します.**

書　式　DC　<ストリング>

文　例　FOO:　　DC　　"ABC"

アセンブル結果は次の通りです.

0000'　41 42 C3　　FOO:　　DC　　'ABC'

解　説
- ●アーギュメントで指定した<ストリング>がカレント・ロケーションカウンタの位置からの連続した領域に用意されます.
- ●ストリング中の各文字はその順序に従ってストアされますが，ストリングの最後の文字は最上位ビットが1になります.
- ●アーギュメントがヌルストリングの場合はエラーとなります.

DS/DEFS(領域定義)

機　能　メモリ領域を予約(確保)します.

書　式　DS　　<式> [,<値>]
*DEFS　<式> [,<値>]

文　例　DEFS　　100H　・・・・ 100Hバイトのメモリ領域を予約します.

DEFS　　100H,2・・・・ 100Hバイトのメモリ領域を予約し，各バイトの値を2にします.

解　説
- DS/DEFSは<式>の値の大きさのメモリ領域を確保します.
- <値>は確保した領域の初期設定値です.<値>を省略すると領域の初期設定はされず,実行時の内容は保証されません.ただし,領域をゼロに初期設定したい場合,<値>に0を指定するかわりに，アセンブル時にMSX･M-80の/Mスイッチを使用することができます./Mスイッチの詳細は4.5　スイッチを参照して下さい.
- <式>中に使用するシンボルは，前もって定義されて(すなわち，パス1のこの時点で値が決って)いなければなりません．そうでないとパス1で“V”エラー，そして多分，パス2では“U”エラーが発生します．パス2で“U”エラーが発生しない(後で定義されている)場合，領域定義疑似命令はパス1でコードを生成していないので，フェーズエラーが発生します.

DW/DEFW(ワード定義)

機　能　２バイトのワードを確保し，初期値を設定します．

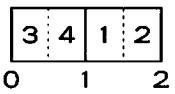
書　式　DW　　<式> [,<式>]...
*DEFW　<式> [,<式>]...

文　例　FOO：　DEFW　1234H

アセンブル結果は次のとおり

0000'　1234　　FOO：　DEFW　1234H

解　説
●カレント・ロケーションカウンタからの連続した領域に<式>の数だけ２バイトのワード領域を確保し，式の値をストアします．
●<式>は２バイトの値(ワード)として評価され，メモリ上に下位バイト，上位バイトの順でストアされます．



●アセンブルリスト上では，上の例のように，ストアされた順ではなく評価された値がそのまま表示されます．

EQU

機　能　＜シンボル＞にアーギュメントで指定した＜式＞の値を割り当てます.

書　式　＜シンボル＞　EQU　＜式＞

文　例　BUF　　EQU　　0F3H

解　説
- ●＜シンボル＞はラベルあるいは変数で，以後式の中で使用することができます.
- ●＜シンボル＞の直後にコロン(：)を付けることは出来ません.
- ●＜式＞が外部参照の場合はエラーとなります.また,＜シンボル＞がすでに＜式＞以外の値を持っている場合は“M”エラーとなります.
- ●＜シンボル＞をプログラム中であとから再定義する場合には，EQUの代わりにSET疑似命令を使用してください.

EXT/EXTRN/EXTERNAL
BYTE EXT/BYTE EXTRN/BYTE EXTERNAL
(外部参照シンボル)

機　能　アーギュメント中のシンボルが外部参照シンボル(他のモジュールで定義されているパブリックシンボル)であることを宣言します.

書　式

	EXT	<シンボル> [,<シンボル>]...
	EXTRN	<シンボル> [,<シンボル>]...
*	EXTERNAL	<シンボル> [,<シンボル>]...
	BYTE EXT	<シンボル> [,<シンボル>]...
	BYTE EXTRN	<シンボル> [,<シンボル>]...
	BYTE EXTERNAL	<シンボル> [,<シンボル>]...

文　例

```
EXTERNAL ITRAN    ; tranf init rtn
```

解　説

- アーギュメント中のシンボルが同一モジュール内で定義してあるシンボルの場合,“M”エラーとなります.
- EXT, EXTRN, EXTERNAL は2バイトの値として, また, BYTE EXT, BYTE EXTERNAL, BYTEEXTRN は1バイトの値として評価されます.
- EXTRNとEXT, BYTE EXTRNとBYTE EXTとはそれぞれ同じ働きをします.
- アーギュメント中のシンボルは, 同一モジュール中で定義したシンボルとおなじように使用することができます.
- シンボルを参照するときに直後にハッシュ記号2つ(##)を付けることで, 同様にシンボルを外部参照として定義することができます.

例

```
EXTERNAL    SUB1
CALL        SUB1
```

例

```
CALL        SUB1 ##
```

- 外部参照シンボルは先頭6文字のみがリンクローダに渡され, 余分な文字は内部で切り捨てられます.
- MSX・M-80 はこの疑似命令についてのコードは生成しません. 文例の EXTRN を含むモジュールでは,

```
CALL  ITRAN
```

のCALLについてコードを生成するとき, ITRAN というシンボルの値をゼロ(0000*)で生成します.
リンクロード時にロードされた他のモジュールの中から PUBLIC ITRAN を含むモジュールを検索し, ITRAN が定義された値(アドレス)に置き換わります.

ENTRY/GLOBAL/PUBLIC(パブリックシンボル)

機　能　同一モジュール内で定義しているシンボルを他のモジュールから参照できるように宣言します.

書　式　ENTRY　<シンボル> [,<シンボル>]...
GLOBAL　<シンボル> [,<シンボル>]...
PUBLIC　<シンボル> [,<シンボル>]...

文　例　PUBLIC　LABEL1, LABEL2, DATA

解　説
- この疑似命令で宣言したパブリックシンボルはMSX･L-80で一緒にリンクロードする他のモジュールで参照することができます.
- ENTRY, GLOBAL, PUBLICの機能は同じです.
- シンボルはモジュール内で定義されていなければなりません. シンボルがモジュール内で定義されていない場合"U"エラーになります. また, シンボル名が外部参照名またはコモンブロック名の場合"M"エラーとなります.
- パブリックシンボル名は先頭6文字のみがリンクローダに渡され, 余分な文字は内部で切り捨てられます.
- MSX･M-80はこの疑似命令についてのコードは生成しません. 次の例のモジュール1とモジュール2のリンクロード時に, モジュール2のITRANシンボルの値(アドレス)がモジュール1のCALLステートメントのITRANの値として使用されます.
- ラベルの直後にコロンを2つ(::)付けることで, パブリックシンボル疑似命令で宣言したのと同じように, 他のモジュールから参照することができます.

サンプルプログラム　モジュール1

```
        EXTRN   ITRAN
                .
                .
                .
        CALL    ITRAN
```

モジュール2

```
        PUBLIC  ITRAN
                .
                .
                .
ITRAN:  LD      HL,PASSA        ;store addr of
                                ; reg pass area
```

SET/DEFL/ASET(セット)

機　能　＜シンボル＞にアーギュメントで指定した＜式＞の値を割り当てます.

書　式
```
＜シンボル＞  SET    ＜式＞      (.Z80 モード以外)
*＜シンボル＞  DEFL   ＜式＞
 ＜シンボル＞  ASET   ＜式＞
```

文　例
```
FOO     ASET    BAZ+1000H
          .
          .
          .
FOO     ASET    3000H
          .
          .
          .
FOO     DEFL    6CDEH
```

解　説

●＜シンボル＞はラベルあるいは変数であり，これ以降式の中で使用することができます.
また,＜シンボル＞の直後にコロン(:)を付けることは出来ません.式が外部参照の場合はエラーとなります.

● SET 疑似命令は Z80 の命令コードに SET 命令があるため，Z80 命令セットモードでは使用できません．ASET と DEFL はどちらの命令セットモードでも使用が可能であり，機能は同じです.

●あとで＜シンボル＞の再定義をしたい場合は,EQU ではなくセット疑似命令を使用します.
いずれかのセット疑似命令で一旦定義した＜シンボル＞は，前に使用したセット疑似命令にかかわりなくどのセット疑似命令でも再定義することができます.(ただし Z80 命令セットモードでは SET を使用できないのは変わりありません.)

3.6.3 PCモード

コードの生成はプログラムカウンタ(略してPC)とよばれるカレント・ロケーションカウンタのアドレスを基準として行います．PCは次のステートメントのコード生成アドレスを指しています．

MSX・M-80ではPCは絶対モード，データ相対モード，コード相対モード，またはコモン相対モードのいずれか1つのモードをもっています．PCモードの指定は以降のプログラムをそれぞれ絶対セグメント，コード相対セグメント，データ相対セグメント，コモン相対セグメントの対応するセグメントとしてアセンブルします．プログラム中のシンボルや式のモードは，PCモードとおなじモード属性になります．詳細については3.2.4を参照ください．PCモード疑似命令はPCモードを設定し，PCモード疑似命令以降のプログラムをどのモードのセグメントにアセンブルするかを決定します．PCモードの疑似命令は次のとおりです．

```
ASEG
CSEG
DSEG
COMMON
ORG
.PHASE/.DEPHASE
```

ASEG(絶対モード)

機　能　PC モードを絶対モードにします.

書　式　ASEG

文　例

```
ASEG
ORG　103H
```

解　説

- ASEG にはアーギュメントはありません.
- ASEG 以降のプログラムは絶対モードのセグメントにコード生成されます.
- ASEG はロケーションカウンタをメモリの絶対セグメント(実アドレス)にセットします.

  ロケーションカウンタのデフォルト値は 0 であるため, ASEG を指定しただけではオペレーティングシステムの領域にプログラムを配置することになります. ASEG ステートメントの次に ORG ステートメントでロケーションカウンタの値を 103H 以上に指定してください.

CSEG(コード相対モード)

機　能　PC モードをコード相対モードにします.

書　式　CSEG

文　例　CSEG

解　説
- CSEG にはアーギュメントはありません.
- CSEG はロケーションカウンタをコード相対セグメントに切り換えます.
- ロケーションカウンタの初期値は 0 で, ORG ステートメントで変更しない限り, 生成コードの大きさだけ増加します.
- CSEG 以降のプログラムはコード相対モードのセグメントにコード生成されます. コード相対セグメントはリロケータブルです. コード相対モードでは, ORG ステートメントで絶対アドレスを指定することはできません. ORG ステートメントでは<式>にはコード相対の値を指定します.
  <式>で絶対値(数値など)を指定すると, コード相対とみなします.
  次の例はロケーションカウンタをコード相対の 50 にします.
  CSEG ステートメントの次に ORG ステートメントを置くことにより, コード相対セグメント中のコード生成アドレスを相対的に変えることができます.

例

```
CSEG
ORG     50
        コード部分 A
DSEG
        データ部分 B
CSEG
        コード部分 C
```

- コード相対セグメントのメモリ配置位置は MSX･M-80 でリンクロードするときに, /P スイッチを使って任意のアドレスに指定することができます.
- CSEG は MSX･M-80 のデフォルトのモードです. ASEG, DSEG, または COMMON 疑似命令を指定するまでプログラムをコード相対セグメントとしてアセンブルします.

DSEG(データ相対モード)

機　能　PC モードをデータ相対モードにします.

書　式　DSEG

文　例　DSEG

解　説
- DSEG にはアーギュメントはありません.
- DSEG はロケーションカウンタをデータ相対セグメントに切り替えます.
- ロケーションカウンタの初期値は 0 で, ORG ステートメントで変更しない限り, 生成コードの大きさだけ増加します.
- DSEG 以降のプログラムはデータ相対モードのセグメントにコード生成されます. データ相対セグメントはリロケータブルです. データ相対モードでは ORG ステートメントで絶対アドレスを指定することはできません. ORG ステートメントでは, <式>にはデータ相対の値を指定します.
  <式>に絶対値(数値など)を指定すると, データ相対とみなします.
- データ相対セグメントのメモリ配置位置は MSX･L-80 でリンクロードするときに/D スイッチを使って任意のアドレスに指定することができます.

COMMON(コモン相対モード)

機　能　COMMON 以降をコモン相対モードにします.

書　式　COMMON　/<ブロック名>/

文　例

```
        COMMON    /DATAIN/
ANVIL   EQU       100H
        DEFB      0FFH
        DEFW      1234H
        DEFM      'FORGE'
        CSEG
          .
          .
          .
```

解　説

- ●コモン相対モードのセグメントは他のモジュールと共用できる共通データエリアです.
- ●<ブロック名>は6文字までです.7文字以上の<ブロック名>を記述すると,先頭6文字までが<ブロック名>として使用され,以降の文字は無効となります.
- ● COMMON は同じ<ブロック名>を持つ COMMON ブロックごとに共通データエリアを作成します.
  <ブロック名>を省略しているか,スペースである場合,<ブロック名>がブランクの共通データエリアと仮定されます.
- ● COMMON は,変数,配列,およびストリングを COMMON ストレージと呼ばれるストレージエリアに割り当て,異なるモジュール間で同一のストレージエリアを共用することができます.
- ● COMMON 以降のステートメントは<ブロック名>の COMMON エリアとしてアセンブルされます.
- ● COMMON エリアの大きさは別の PC モード疑似命令までの COMMON ブロック内の変数,配列,データ領域長の合計です.
- ● MSX･L-80 でリンクロードしようとする複数モジュールに大きさのことなる同一名の COMMON ブロックがある場合,もっとも大きな COMMON ブロックを含むモジュール名を MSX･L-80 コマンドの先頭に指定し,最初にロードしなければなりません. MSX･L-80 の詳細は第5章を参照してください.
- ● COMMON はロケーションカウンタを COMMON ブロックのポインタに切り換えます. ロケーションは FORTRAN の COMMON ステートメントとの互換性を保つために常にエリアの始めにされます.

ORG(セットオリジン)

機　能　ロケーションカウンタの値を変更します.

書　式　ORG　＜式＞

文　例　1)　DSEG
　　ORG　　50
2)　ASEG
　　ORG　　800H

解　説
- ORG はロケーションカウンタに＜式＞の値をセットします. MSX・M-80 は＜式＞の値のアドレスからコードを生成します.
- 絶対モードでは＜式＞の値は絶対値です.
- コード相対モード，データ相対モード，コモン相対モードでは＜式＞の値はそれぞれのセグメント内での相対値です. コード相対モード, データ相対モードでは＜式＞で絶対アドレスを指定することができません. コード相対セグメント，データ相対セグメントのロードアドレスはリンクロード時にフレキシブルに決まり, /P：＜アドレス＞や/D：＜アドレス＞スイッチを指定することにより，任意に指定したアドレスへロードすることができます. /P スイッチ, /D スイッチの詳細については 5.5 を参照してください.
- ＜式＞に使用するシンボルは同一モジュール内で定義しなければなりません. また, ＜式＞の値は絶対値またはロケーションカウンタで使用するエリアと同一のエリアになければなりません.
- 文例の 1) はデータ相対モードのロケーションカウンタをデータ相対セグメントの先頭＋50 に設定します. データ相対セグメントの先頭 50 バイトはとばされます.
- 文例の 2) は絶対モードのロケーションカウンタを絶対アドレスの 800H に設定します.
  2) の絶対セグメントは絶対アドレス 800H からロードされます.

.PHASE/.DEPHASE(リロケート)

機　能　.PHASE から .DEPHASE までのコードを＜式＞のエリアで実行できるようにします.

書　式

```
.PHASE  ＜式＞
   .
   .
   .
.DEPHASE
```

文　例

```
        .PHASE   100H
FOO:    CALL     BAZ
        JP       ZOO
BAZ:    RET
        .DEPHASE
ZOO:    JP       5
```

解　説

●.PHASE/.DEPHASE は特定のアドレスでプログラムの一部分を実行するために用います.

●.PHASE/.DEPHASE 間のプログラムの PC モードは＜式＞のモードと同じですが, コードは.PHASE を指定する直前のプログラムと同じセグメントに連続して配置されます.

●.PHASE/.DEPHASE 間のプログラムを実行するためには＜式＞で指定したアドレスへ移動させなければなりません.

●文例は次のようにアセンブルされます.

```
                                .PHASE  100H
0100    CD 0106         BAR:    CALL    BAZ
0103    C3 0007'                JP      ZOO
0106    C9              BAZ:    RET
                                .DEPHASE
0007'   C3 0005         ZOO:    JP      5
```

3.6.4 ファイル関連

ファイル関連の疑似命令はプログラムに長いコメントを入れたり，モジュールに名前を付けたりします．ファイル関連の疑似命令は次のとおりです．

.COMMENT
END
INCLUDE/$INCLUDE/MACLIB
NAME
.RADIX
.REQUEST

.COMMENT(コメント)

機　能　＜区切り記号＞ではさまれた＜テキスト＞をコメントとみなします.

書　式　.COMMENT　＜区切り記号＞＜テキスト＞＜区切り記号＞

文　例

```
.COMMENT  * any amount of text
entered here
    •
    •
    •     * ; return to normal assembly
```

解　説

●.COMMENT のあとの最初の空白以外の文字を区切り文字として扱います.＜区切り記号＞以降，次の＜区切り記号＞までの＜テキスト＞がコメントブロックとなります.

●コメント行のようにコメントの前のセミコロン(;)を置く必要はありません.＜テキスト＞は複数行にわたって書くことができます.

●アセンブル中コメントブロックは無視され，コードの生成は行われません.

END(プログラムの終了)

機　能　モジュールの終了を指定します．また，オプションでプログラムの実行開始点を指定します．

書　式　END　[<式>]

文　例　1）　END　PROG1
2）　END

解　説
- ENDステートメントはモジュールの終了を宣言します．また，リンクローダにプログラムの開始アドレスを指示します．ソースプログラムにENDステートメントがない場合，“%No END statement”の警告メッセージが表示されます．
- <式>はMSX・L-80がプログラムの開始点としてロードすることができるシンボル，数値，その他のアーギュメントです．<式>を指定すると，MSX・L-80は100H番地に<式>のアドレスへのジャンプ命令をセットします．<式>を指定しない場合は，MSX・L-80にプログラム開始点が渡されず，最初にロードしたモジュールの先頭からプログラムが開始します(また，<式>を指定しない場合，MSX・L-80の/Gスイッチは無効です)．
- <式>の指定は，このモジュールがメインプログラムであることを宣言します．<式>が無い場合，MSX・L-80はこのモジュールをサブルーチンの集まりとみなします．
- リンクロード時にプログラム開始点を指定したモジュールが2つ以上ある場合，MSX・L-80はどれがメインプログラムであるか区別することが出来ません．プログラム開始点を持つモジュールを2つ以上リンクロードする場合は，MSX・L-80の/G：<名前>または/E：<名前>スイッチを使用してください．
- 高級言語プログラムは，自動的に，プログラム開始点を指定したオブジェクトを作成します．
  そのため，通常は高級言語プログラムがメインプログラムとなります．MSX・L-80の/G：<名前>または/E：<名前>スイッチを使用してアセンブリ言語で作成したモジュールを高級言語で作成したモジュールの前に実行することは可能ですが，それよりも，高級言語モジュールの最初でCALLあるいはINCLUDEステートメントなどを使ってアセンブリ言語モジュールを呼び出す方法をおすすめします．

INCLUDE/$INCLUDE/MACLIB(挿入)

機　能　INCLUDE ステートメントの位置に他のソースファイルに含まれるステートメントを論理的に挿入します．

書　式　INCLUDE　＜ファイル名＞
$INCLUDE ＜ファイル名＞
MACLIB　＜ファイル名＞

文　例　INCLUDE TEXT

解　説
- INCLUDE，$INCLUDE，MACLIB の機能は同じです．
- インクルード疑似命令は，アセンブリ中に，別のソースファイルを現在のソースファイルの中に挿入します．インクルード疑似命令を使用すると，頻繁に使用する部分を繰り返しプログラミングする手間を省くことができます．
- ファイル名はオペレーティングシステムで有効な任意のファイル指定で，デフォルトの拡張子は.MAC です．INCLUDE 環境変数が(“SET INCLUDE=...” コマンドで)セットされていれば，それはパス名とみなされ，INCLUDE ステートメントで指定したファイルを，そのディレクトリから読み込みます．指定するファイル名には，ディレクトリを含むパス名を指定できます．
- インクルード疑似命令は，ステートメントの読み込みを指定されたファイルに，一時的に変更します．このファイルを読み終わると，アセンブラはインクルード疑似命令の次のステートメントからもとのソースファイルのアセンブルを再開します．
- インクルードのネスト(入れ子)は許されません．インクルードのネストがあると“O” エラーとなります．
- ＜ファイル名＞で指定したファイルがアセンブル時に見つからない場合“V”エラーとなり，インクルード疑似命令は無視されます．
- アセンブルリストには挿入したインクルードファイルのステートメントがプリントされます．
  また挿入したステートメントのオブジェクトコードとソースラインの間には，他のステートメントとの区別をするために“C”を表示します．リストフォーマットの詳細についてはリスティング制御命令を参照してください．

NAME(モジュール名定義)

機　能　モジュール名を定義します.

書　式　NAME　('<モジュール名>')

文　例　NAME　('MODUL1')

解　説
- ●<モジュール名>はかっこ()とシングルクォーテーションで囲みます. モジュール名は先頭6文字までが有効です.
- ●モジュール名はTITLE疑似命令でも定義することができます. NAMEおよびTITLE疑似命令の両方を指定していない場合, モジュール名はソースファイル名から作成されます.

.RADIX(基数指定)

機　能　定数の入力基数を指定します.

書　式　.RADIX　<式>

文　例　.RADIX　2

解　説
- ●<式>は現在の基数の指定に関わらず10進数の定数です.
- ●定数のデフォルトの入力基数は10進です..RADIXを使うと2進から16進までの範囲で入力基数を任意の底に変更することができます.
- ●.RADIXはアセンブルリストの基数を変更するわけではなく,特殊な表記法を使わずに,指定した基数の数値を入力できるようにします.
- ●.RADIXで指定した基数と異なる基数を用いる場合は3.4.1で説明している表記法を使ってください.

サンプルプログラム

```
DEC:    DEFB    20
        .RADIX  2
BIN:    DEFB    00011110
        .RADIX  16
HEX:    DEFB    0CF
        .RADIX  8
OCT:    DEFB    73
        .RADIX  10
DECI:   DEFB    16
HEXA:   DEFB    0CH
```

アセンブル結果

```
0000'   14              DEC:    DEFB    20
0002                            .RADIX  2
0001'   1E              BIN:    DEFB    00011110
0010                            .RADIX  16
0002'   CF              HEX:    DEFB    0CF
0008                            .RADIX  8
0003'   3B              OCT:    DEFB    73
000A                            .RADIX  10
0004'   10              DECI:   DEFB    16
0005'   0C              HEXA:   DEFB    0CH
```

.REQUEST(リンクロード要求)

機　能　リンクローダに，未解決の外部参照シンボルを指定したファイルで検索することを要求します．

書　式　.REQUEST　＜ファイル名＞［,＜ファイル名＞］

文　例　.REQUEST　SUBR1

解　説
- .REQUEST はリンクローダに対する要求です．リンクローダは外部参照シンボルの参照を解決するために＜ファイル名＞で指定したモジュールを検索し，対応するシンボルがあれば追加モジュールとしてロードします．
- ＜ファイル名＞はシンボルとして有効な文字列でなければならず，ドライブ指定や拡張子は指定できません．＜ファイル名＞の長さは7文字までです．
- 文例ではリンクロード時に外部参照シンボルに対応するパブリックシンボルがロード済のモジュールにない場合，SUBR1 を検索します．

3.6.5　リスティング疑似命令

リスティング疑似命令はアセンブルリストのプリントに関する疑似命令です．

フォーマット制御

アセンブルリストのページブレーク，タイトル，およびサブタイトルを指示します．
フォーマット制御疑似命令は次のとおりです．

PAGE/$EJECT
TITLE
SUBTTL/$TITLE

一般リスティング制御

アセンブルリストの出力/抑止を指定します．特に指定しなくてもプログラムの先頭で.LISTを指定したものとみなします．一般リスティング疑似命令は次のとおりです．

.LIST
.XLIST

条件リスティング制御

偽条件ブロック内に含まれるステートメントのアセンブルリストの出力/抑止を指定します．
4.5 の/X スイッチに関する記述も参照してください．条件リスティング制御命令は次のとおりです．

.SFCOND
.LFCOND
.TFCOND

マクロ拡張リスティング制御

拡張リスティング疑似命令はマクロおよびREPT, IRP, IRPCで展開するステートメントのリスティングを制御します．拡張リスティング制御疑似命令はマクロまたはリピートブロック内のステートメントに対して有効です．指定がない場合は.XALLを指定しているものとみなします．
マクロ拡張リスティング制御疑似命令は次のとおりです．

.LALL
.SALL
.XALL

CREF 情報出力制御疑似命令

CREF 情報出力制御疑似命令はクロスリファレンス情報をクロスリファレンス・ファイルへ出力するか抑止するかを指定します．この疑似命令はMSX･M-80 スイッチ/C を指定しないかぎり効果はありません．

CREF 情報出力制御疑似命令はソースプログラムの任意の位置に指定し，プログラムの必要な

部分についてのクロスリファレンス情報を出力することができます．CREF 情報出力制御疑似命令は次のとおりです．

.CREF
.XCREF

PAGE/$EJECT(フォームフィード)

機　能　アセンブルリストの改ページを行います.

書　式　PAGE　[<式>]
$EJECT [<式>]

文　例　$EJECT　58

解　説　●改ページを行いフォームフィード疑似命令以降のステートメントを次ページからプリントします. MSX・M-80 はページの終りでリスティングファイル内にフォームフィード文字を置きます.
●<式>は，指定する場合，改ページ後の1ページに出力する行数を指定します.<式>は10から255の数値です. デフォルトの行数は1ページあたり60行です.

TITLE(タイトル)

機　能　アセンブルリストにプリントするタイトルを指定します.

書　式　TITLE　<テキスト>

文　例　TITLE　PROG1

解　説
- ●アセンブルリストの各ページの最初の行にプリントするタイトルを<テキスト>で指定します. 1つのモジュールに2つ以上のTITLEを指定すると“Q”エラーとなります.
- ●同じモジュール内にNAME疑似命令を使用していない場合,<テキスト>の先頭6文字をモジュール名として設定します.(TITLE, NAME疑似命令がない場合,モジュール名はソースファイル名から作られます.)
- ●文例はアセンブルリストの各ページの最初の行にPROG1というタイトルをプリントし, NAME疑似命令がモジュール内になければモジュール名をPROG1に設定します.

SUBTL/$TITLE(サブタイトル)

機　能　アセンブルリストにプリントするサブタイトルを指定します.

書　式　SUBTTL <テキスト>
$TITLE ('<テキスト>')

文　例　SUBTTL SPECIAL I/O ROUTINE
・
・
・
SUBTTL
・
・
・

解　説
- SUBTTL はアセンブルリストの各ページのタイトルの次の行にプリントするサブタイトルを指定します. <テキスト>の 60 文字以降は無視します.
- 1つのモジュールに任意の数の SUBTTL を指定することができます. 既に SUBTTL を指定している場合, 前の SUBTTL でセットした<テキスト>は最新の SUBTTL の<テキスト>で置き換わります.
  既に指定しているサブタイトルのプリントを解除する場合は<テキスト>の指定を省略してください.
- 文例の最初の SUBTTL により, サブタイトル SPECIAL I/O ROUTINE がアセンブルリストの各ページの先頭にプリントされます. 2番目の SUBTTL はサブタイトルを解除し空白にします.

.LIST(リスト)

機　能　.LIST 以降のステートメントについてアセンブルリストを出力します.

書　式　.LIST

文　例

```
        •
        •
        •
.XLIST      ; listing suspended here
        •
        •
        •
.LIST       ; listing resumes here
```

解　説
- .LIST はアセンブル開始時のデフォルト状態です.
- MSX・M-80 コマンドでリスティングファイルを指定した場合に，指定したファイルにリストを出力します.

.XLIST(リスト抑止)

機　能　.XLIST 以降のステートメントのアセンブルリストの出力を抑止します.

書　式　.XLIST

文　例

```
        •
        •
        •
.XLIST      ; listing suspended here
        •
        •
        •
.LIST       ; listing resumes here
```

解　説

●アセンブル開始時のデフォルトは.LIST 状態です..XLIST を指定すると，ソースステートメント及び生成したオブジェクトコードはリストされません..XLIST は.LIST を指定するまでのステートメントに対して有効です.

●.XLIST は，(.LIST 疑似命令以外の)リスティング疑似命令に優先します．したがって，他のリスティング疑似命令があっても，リストには何もプリントされません.

.PRINTX(コンソールメッセージ表示)

機　能　アセンブル中，テキストをコンソールに表示します．

書　式　.PRINTX ＜区切り記号＞＜テキスト＞＜区切り記号＞

文　例　1）　.PRINTX *Assembly half done*

2）
```
IF1
.PRINTX *Pass 1 done*  ; pass 1 message only
ENDIF
IF2
.PRINTX *Pass 2 done*  ; pass 2 message only
ENDIF
```

解　説
- .PRINTX は＜区切り記号＞にはさまれた＜テキスト＞をアセンブル時にコンソールへ表示します．
- .PRINTX 以降の空白以外の最初の文字を＜区切り記号＞とみなします．また，＜区切り記号＞から次の＜区切り記号＞までが＜テキスト＞です．
- .PRINTX は長時間かかる場合にアセンブルの進行状況を表示したり，条件アセンブリのスイッチの値を表示するのに役立ちます．
- .PRINTX はパス 1，パス 2 の両パスでそれぞれ＜テキスト＞をコンソールへ表示します．
  どちらかのパスでのみ表示したい場合，IF1，または IF2 疑似命令を併用して下さい．
  IF1 及び IF2 については条件疑似命令を参照してください．

.SFCOND(偽条件抑止)

機　能　**偽条件ブロックのアセンブルリスト出力を抑止します.**

書　式　.SFCOND

文　例　.SFCOND

解　説　●.SFCOND は MSX･M-80 のスイッチ/X に関係なく，以後の偽条件のブロックのアセンブルリストの出力を抑止します.

●偽条件ブロックに対するリスティング制御疑似命令がない場合,/X スイッチの指定があれば偽条件ブロックのアセンブルリストは抑止されます.

.LFCOND(偽条件出力)

機　能　**偽条件ブロックのアセンブルリストを出力します.**

書　式　.LFCOND

文　例　.LFCOND

解　説　●.LFCOND は MSX･M-80 のスイッチ/X に関係なく，以後の偽条件ブロックのアセンブルリストが出力されます.

●偽条件ブロックに対するリスティング制御疑似命令がない場合,/X スイッチの指定がなければ偽条件ブロックのアセンブルリストは出力されます.

.TFCOND(トグルリスティング偽条件)

機　能　/X スイッチの有無により偽条件ブロックのアセンブルリストの出力を決定する．また，.TFCOND の指定をするたびに出力/抑止を反転する．

書　式　.TFCOND

文　例　.TFCOND

解　説

●.SFCOND および.LFCOND の指定とは無関係に，/X で決まるデフォルトの出力状態を反転します．デフォルトでは，/X を指定すると偽条件ブロックの出力は抑止され，指定しないと抑止されません．

●偽条件ブロックを，2 つの.TFCOND で囲んでおくことにより，次のような効果を得ることができます．

a. /x を指定する時，奇数番目の.TFCOND とその次の.TFCOND までの間の偽条件ブロックのアセンブルリストは出力され，偶数番目の.TFCOND とその次の.TFCOND までの間の偽条件ブロックの出力は抑止されます．

b. /x を指定しない時，奇数番目の.TFCOND とその次の.TFCOND までの間の偽条件ブロックのアセンブルリストの出力は抑止され，偶数番目の.TFCOND とその次の.TFCOND までの間の偽条件ブロックは出力されます．

.LALL

機　能　**マクロおよびリピートブロックの展開部分をアセンブルリストに出力します.**

書　式　.LALL

文　例　.LALL

解　説　●オブジェクトコードを生成するステートメントか否かにかかわらず，マクロおよびリピートブロックの展開部分のすべてのステートメントをアセンブルリストに出力します.

.XALL

機　能　**マクロおよびリピートブロックの展開部分のうちオブジェクトコードを生成するステートメントをアセンブルリストに出力します.**

書　式　.XALL

文　例　.XALL

解　説　●拡張リスティング制御疑似命令がない場合は，.XALL を指定しているものとみなします.

.SALL

機　能　**マクロおよびリピートブロックの展開部分のアセンブルリストへの出力を抑止します.**

書　式　.SALL

文　例　.SALL

.CREF(クロスリファレンス作成)

機　能　クロスリファレンス・ファイルを作成します.

書　式　.CREF

文　例　.CREF

解　説
- .CREF 以降, .XCREF の指定をするまでクロスリファレンス情報が出力されます.
- .CREF はデフォルトの状態ですので, .XCREF 疑似命令を使用した後でクロスリファレンス・ファイルの作成を再開するときに使用してください.
- .CREF の機能は MSX・M-80 のコマンドに/C スイッチを指定したときに有利です.

.XCREF(クロスリファレンス抑止)

機　能　クロスリファレンス・ファイルの作成を抑止します.

書　式　.XCREF

文　例　.XCREF

解　説
- .XCREF は.CREF 疑似命令の指定を解除します. .XCREF 以降, .CREF の指定をするまでの部分のクロスリファレンス情報は出力されません. . XCREF を使用することによってクロスレファレンス情報の作成を指定した部分だけ抑止することができます.
- .CREF と.XCREF は MSX・M-80 のコマンドに/C スイッチを指定したときに有効です. このため通常のリスティングファイル(クロスリファレンス・ファイルではない)を出力させる場合には, MSX・M-80 のコマンドに/C スイッチを指定しないだけでよく, .XCREF を使う必要はありません.

3.7　マクロ機能

マクロ機能は，何回も繰り返して現われる一連のコードを，再コーディングすることなくマクロブロックとして定義できるようにします．マクロブロックはマクロ定義疑似命令，あるいは反復疑似命令で始まり，ENDM 疑似命令で終わります．ブロックの中でさらに，任意の別のマクロブロックを使用することができます．ネストの深さはメモリ領域の大きさでのみ制限されます．

マクロ機能で使う疑似命令には，次のものがあります．

●マクロ定義疑似命令
　MACRO

●反復疑似命令
　REPT
　IRP
　IRPC

●マクロブロックあるいはマクロ展開の終了
　ENDM
　EXITM

●マクロ内のローカルシンボル宣言
　LOCAL

また，マクロ機能では次の特殊なマクロ演算子が使えます．

　$
　; ;
　!
　%

MACRO～ENDM(マクロ定義)

機　能　マクロブロックを定義します.

書　式

```
<マクロ名>  MACRO   [<ダミー> [,<ダミー>]...]
                .
                .
                .
            ENDM
```

解　説

- MACROステートメントから，ENDMまでのステートメントがマクロ本体，すなわちマクロの定義になります．マクロ定義に対するオブジェクトコードは生成されません.
- <マクロ名>は定義するマクロの名前で，シンボル名の規則に従います.<マクロ名>の長さは任意ですが，先頭16文字までが有効です．マクロを定義した後は,<マクロ名>を使ってマクロを呼びだします.
- <ダミー>は，マクロブロックが使用されるときに，マクロ呼び出しの<パラメータ>のテキストで一対一に置き換えられる部分を指定します.<ダミー>は32文字までの文字列です.
  また,<ダミー>の数は一行に入るまでいくつでも指定できます.<ダミー>と<ダミー>との間はカンマ(,)で区切ります.

サンプルプログラム

```
MOVE    MACRO   X,Y
        LD      HL,(Y)
        LD      (X),HL
        ENDM
```

呼び出し例

```
    MOVE    AREA2,AREA1
```

アセンブル結果

```
MOVE    MACRO   X,Y
        LD      HL,(Y)
        LD      (X),HL
        ENDM
        MOVE    AREA2,AREA1
+       LD      HL,(AREA1)
+       LD      (AREA2),HL
```

TITLE，SUBTTL疑似命令をマクロブロックが現われる部分で使いたい場合は，ステートメントの形式に注意しなければなりません．例えば，もし SUBTTL MACRO DEFINITIONS という行を入力したとすると,MSX・M-80はこのステートメントをSUBTTLをマクロ名としDEFINITIONSをダミーとするマクロ定義とし

てアセンブルします．この問題を避けるためには，MACROという単語を避け，MACROSなどを使うようにします．

マクロの呼び出し

マクロの呼び出しは，マクロを定義した後(マクロ定義の前にマクロの呼び出しをするとエラーとなります)，プログラムの任意の場所で行うことができます．
呼び出しの形式は次のとおりです．

＜マクロ名＞　［＜パラメータ＞［,＜パラメータ＞］...］

●＜マクロ名＞はマクロブロックの名前です．

●＜パラメータ＞はマクロ定義の＜ダミー＞と左から順番に対応し，マクロブロック中の＜ダミー＞を置き換えます．

●＜パラメータ＞の数は任意ですが，＜パラメータ＞の並びは一行以内に納まらなくてはなりません．
また，パラメータとパラメータとの間はカンマ(,)で区切ります．

●カンマで区切った＜パラメータ＞を山形かっこ(＜＞)で囲むと，山形かっこ内のすべての項目を単一の＜パラメータ＞として扱います．

FOO　1,2,3,4,5　・・・・５つのパラメータとして扱います．

FOO　＜1,2,3,4,5＞　・・・・１つのパラメータとして扱います．

●＜パラメータ＞の数とマクロ定義の＜ダミー＞の数は一致しなくても構いません．
＜パラメータ＞の数が＜ダミー＞の数より多い場合，余った＜パラメータ＞は無視されます．
また＜パラメータ＞の数が＜ダミー＞の数より少ない場合，不足している＜パラメータ＞をヌルとみなします．

サンプルプログラム

```
EXCHNG  MACRO   X,Y
        PUSH    X
        PUSH    Y
        POP     X
        POP     Y
        ENDM
```

呼び出し例

```
                        EXCHNG  MACRO   X,Y
                                PUSH    X
                                PUSH    Y
                                POP     X
                                POP     Y
                                ENDM
0000'  3A 002F                  LD      A,(2FH)
0003'  67                       LD      H,A
0004'  3A 003F                  LD      A,(3FH)
0007'  57                       LD      D,A
                                EXCHNG  HL,DE
0008'  E5               +       PUSH    HL
0009'  D5               +       PUSH    DE
000A'  E1               +       POP     HL
000B'  D1               +       POP     DE
```

反復疑似命令

反復疑似命令は反復ブロックを指定した回数だけ繰り返し展開します．
マクロと反復疑似命令との違いは次のとおりです．

●マクロはブロックに名前をつけ，あとで必要な場所で何回でも呼び出して展開することができます．
●マクロではマクロ呼び出し時にパラメータを指定することによって，展開するマクロを変更することができます．

反復疑似命令は反復ブロックを反復疑似命令の位置に繰り返し展開します．
他の場所から呼び出すことはできません．

REPT～ENDM(反復)

機　能　**反復ブロックを指定した数だけ繰り返し展開します.**

書　式　REPT　＜式＞

　　・

　　・

　　・

ENDM

解　説

- REPT と ENDM ステートメント間の反復ブロックを＜式＞の回数だけ繰り返し展開します.
- ＜式＞は 16 ビットの符号なし整数として評価されます.
- ＜式＞が外部参照であったり，定義していない項目がある場合にはエラーとなります.

サンプルプログラム

```
0000                    X       DEFL    0
                                REPT    10      ;Generates DEFB 1 - DEFB 10
                                DEFB    X
                        X       DEFL    X+1
                                ENDM
0000'  00               +       DEFB    X
0001'  01               +       DEFB    X
0002'  02               +       DEFB    X
0003'  03               +       DEFB    X
0004'  04               +       DEFB    X
0005'  05               +       DEFB    X
0006'  06               +       DEFB    X
0007'  07               +       DEFB    X
0008'  08               +       DEFB    X
0009'  09               +       DEFB    X
```

IRP～ENDM(不定回数反復)

機　能　反復ブロックを指定したパラメータに従ってくり返し展開します.

書　式　.IRP　＜ダミー＞,＜山形かっこで囲んだパラメータの並び＞

　　・

　　・

　　・

　　ENDM

解　説　●パラメータの並びは山形かっこ(＜＞)で囲み，カンマ(,)で区切ります.

●IRP と ENDM ステートメント間の反復ブロックを＜パラメータ＞の数だけくり返し展開します.

また，展開中に反復ブロックに含まれる＜ダミー＞を展開回数に従って左側からの＜パラメータ＞で順次置き換えます.

サンプルプログラム

```
                                    IRP     X,<1,2,3,4,5,6,7,8,9,10>
                                    DEFB    X
                                    ENDM
0000'   01              +           DEFB    1
0001'   02              +           DEFB    2
0002'   03              +           DEFB    3
0003'   04              +           DEFB    4
0004'   05              +           DEFB    5
0005'   06              +           DEFB    6
0006'   07              +           DEFB    7
0007'   08              +           DEFB    8
0008'   09              +           DEFB    9
0009'   0A              +           DEFB    10
```

IRP を MACRO 定義内で使用する場合，マクロ呼び出しの時の山形かっこはマクロの展開時に取り除かるので，前の例をマクロ呼び出しの形で使用する場合以下のようにします.

```
                        FOO         MACRO   X
                                    IRP     Y,<X>
                                    DEFB    Y
                                    ENDM
                                    ENDM
                                    FOO     <1,2,3,4,5,6,7,8,9,10>
0000'   01              +           DEFB    1
0001'   02              +           DEFB    2
0002'   03              +           DEFB    3
0003'   04              +           DEFB    4
0004'   05              +           DEFB    5
0005'   06              +           DEFB    6
```

```
0006'  07          +        DEFB    7
0007'  08          +        DEFB    8
0008'  09          +        DEFB    9
0009'  0A          +        DEFB    10
```

山形かっこはマクロの展開時に取り除かれ，単一のパラメータとしてIRPのパラメータ内のXと置き換わります．

IRPC～ENDM(不定回数文字反復)

機　能　反復ブロックを指定した文字列に従ってくり返し展開します.

書　式　IRPC　＜ダミー＞,＜文字列＞

　　・

　　・

　　・

　　ENDM

解　説　●IRPCとENDMステートメント間の反復ブロックを＜文字列＞の文字数だけくり返し展開します．また，反復ブロックに含まれる＜ダミー＞は展開中に展開回数に従って左側から＜文字列＞中の文字で順次置き換えられます.

サンプルプログラム

```
                                IRPC    X,0123456789
                                DEFB    X+1
                                ENDM
0000'  01               +       DEFB    0+1
0001'  02               +       DEFB    1+1
0002'  03               +       DEFB    2+1
0003'  04               +       DEFB    3+1
0004'  05               +       DEFB    4+1
0005'  06               +       DEFB    5+1
0006'  07               +       DEFB    6+1
0007'  08               +       DEFB    7+1
0008'  09               +       DEFB    8+1
0009'  0A               +       DEFB    9+1
```

EXITM

機　能　**マクロや反復ブロックの展開を終了します.**

書　式　EXITM

解　説

- EXITM を条件疑似命令と組み合わせて使用することにより，マクロや反復ブロックの展開を展開時の条件によって途中で終了することができます.
- EXITM をアセンブルするとマクロや反復ブロックの展開を中止し，残りの展開についてのオブジェクトコード生成を行いません.
- EXITM を含むブロックが他のブロックの中の入れ子になっている場合，外部のブロックの展開は続行します.

サンプルプログラム

```
FOO     MACRO   X
Y       DEFL    0
        REPT    X
Y       DEFL    Y+1
        IF      Y GT 255    ;test Y
        EXITM               ;if true,exit REPT
        ENDIF
        DEFB    Y
        ENDM
```

FOO マクロは呼び出しステートメントで指定した数だけバイト定義を展開します.
定義内容は'01'からの昇順です.
呼び出し時に 255 より大きい数を指定しても，255 以降の展開はしません.

LOCAL(マクロシンボル)

機　能　**マクロ展開時にマクロ定義中の<ダミー>を..0000から..FFFFの重複しないシンボルに置き換えて展開するよう指示します.**

書　式　LOCAL　<ダミー> [,<ダミー>]...

文　例　LOCAL　A, B, C

解　説
- LOCALはマクロ定義中でのみ使用できます.
- マクロ定義中にLOCALで指定した<ダミー>をシンボルとして使用しているステートメントがあれば, モジュール内で重複しない..0000から..FFFFのシンボルに置き換えて展開されます.
- LOCALステートメントはマクロ定義の最初(つまり, MACROステートメントの次のステートメント)に指定しなければなりません.
- LOCALステートメントはシンボルを自動的に作成するので, マクロ内でのラベルの定義, 参照をするときなどに便利です.
- LOCALステートメントは..0000から..FFFFまでのシンボルを定義するので, プログラミング時に..0000から..FFFFまでのシンボルを定義しないようにしてください.

サンプルプログラム

```
                                FOO     MACRO   NUM,Y
                                        LOCAL   A,B,C,D,E
                                A:      DEFB    7
                                B:      DEFB    8
                                C:      DEFB    Y
                                D:      DEFB    Y+1
                                E:      DEFW    NUM+1
                                        JP      A
                                        ENDM
                                        FOO     0C00H,0BEH
0000'   07              +..0000: DEFB    7
0001'   08              +..0001: DEFB    8
0002'   BE              +..0002: DEFB    0BEH
0003'   BF              +..0003: DEFB    0BEH+1
0004'   0C01            +..0004: DEFW    0C00H+1
0006'   C3 0000'        +        JP      ..0000
```

特殊なマクロ演算子

マクロ定義中では次のような特殊な演算子を使うことができます.

●&　・・・・ テキストやシンボルの連結
●；；　・・・・ 展開が行われないコメント
●!　・・・・ 感嘆符の次のキャラクタを文字通りアーギュメントして扱う.

マクロ呼び出しでは次のような特殊な演算子を使うことができます.

●%　・・・・ 式を現在指定されている基数で表わす数に変換する.

&

機　能　テキストまたはシンボルを連結します.

文　例

```
1)PROC&X      EQU      $
2)            LD       B,'&X'
```

解　説

●&はマクロ呼び出しステートメントでは使用できません.

●引用符で囲まれたストリング内でダミーパラメータを使う場合，また，シンボルの一部としてダミーパラメータを使う場合(文字列とダミーを連結する場合)にダミーの直前に&を置きます.&がないとき，展開時にダミーパラメータとの置き換えを行いません.

サンプルプログラム

```
ERRGEN   MACRO    X
ERROR&X: PUSH     BC
         LD       B,'&X'
         JP       ERROR
         ENDM
```

呼び出し例

```
         ERRGEN   A
+ERRORA: PUSH     BC
+        LD       B,'A'
+        JMP      ERROR
```

;;

機　能　展開を行わないコメント.

文　例　LD　　B,'&X'　　;;COMMENT

解　説　●マクロブロックの処理で,(;;)以降はブロックの一部として保存しません.(.LALL疑似命令を指定してもマクロ展開のリストにはプリントされません.)

●セミコロン1つ(;)のコメントはマクロ展開のリストにプリントされます.

サンプルプログラム

```
                        DBGEN   MACRO   X
                        Y       DEFL    0
                                REPT    X
                        Y       DEFL    Y+1
                                DEFB    Y       ;;generate each iteration
                                ENDM
                                ENDM
                                DBGEN   10
0000'   01              +       DEFB    Y
0001'   02              +       DEFB    Y
0002'   03              +       DEFB    Y
0002'   04              +       DEFB    Y
0002'   05              +       DEFB    Y
0002'   06              +       DEFB    Y
0002'   07              +       DEFB    Y
0002'   08              +       DEFB    Y
0002'   09              +       DEFB    Y
0002'   0A              +       DEFB    Y
```

!

機　能　アーギュメントやパラメータ中の感嘆符の次のキャラクタを文字通りアーギュメントとして扱います.

文　例　IRP　　X,<!;,! ,!,,!4>

解　説　●!;は<;>と同じです.

サンプルプログラム

```
                                    IRP     X,<!;,! ,!,,!4>
                                    DEFB    '&X'
                                    ENDM
0000'   3B              +           DEFB    ';'
0001'   20              +           DEFB    ' '
0002'   2C              +           DEFB    ','
0003'   34              +           DEFB    '4'
```

%

機　能　**マクロのパラメータとして指定した＜式＞を.RADIX 疑似命令でセットした基数に従って表わします.**

解　説

●%はマクロ呼び出しステートメントのパラメータにのみ使用でき，直後に続く式を展開時の基数(.RADIX 参照)で表わす数に変換します．マクロ展開時には変換後の数がダミーのかわりに使われます．

●%を使用することによりマクロに値を渡すことが出来ます．(通常は，パラメータで指定したテキストをそのままダミーと置き換えます.)

●%に続く式は，DS 領域定義疑似命令の式の規則に従います．つまり，式は(リロケータブルでない)絶対値で評価できなければなりません．

サンプルプログラム

```
PRINTE  MACRO   MSG,N
        .PRINTX * MSG N *
        ENDM
SYM1    EQU     100
SYM2    EQU     200
        PRINTE  <SYM1 + SYM2 =>,%(SYM1+SYM2)
        ;PRINTX * SYM1 + SYM2 = 300 *
```

%(SYM2+SYM2)は式の値を計算後，ダミーNと置き換わります．%(SYM1+SYM2)の代わりに(SYM1+SYM2)を指定すると次のステートメントが展開されます．

```
.PRINTX * SYM1 + SYM2 = (SYM1+SYM2) *
```

3.8　条件疑似命令

条件疑似命令はアセンブル時の特定の条件をテストし，その結果に応じてコードブロックを生成するかどうかを決めます．条件疑似命令の形式は次のとおりです．

```
IFxxxx ［アーギュメント］      COND ［アーギュメント］
  ・                            ・
  ・                            ・
  ・                            ・
 ［ELSE                        ［ELSE
  ・                            ・
  ・                            ・
  ・］                          ・］
ENDIF                         ENDC
```

● IFxxxx は条件を終了するために，ENDIF と対応していなければなりません．また，COND は条件を終了するために，ENDC と対応していなければなりません．IFxxxx や COND に対応する ENDIF や ENDC がない場合，パス 1，パス 2 の終了時に“Unterminated conditional”のメッセージが出力されます．
また，対応する IFxxxx がない ENDIF，または対応する COND がない ENDC の場合，“C”エラーとなります．

●アセンブラは条件ステートメントの真偽を評価します．

●評価結果が真である場合，ELSE ステートメントがあれば IFxxxx/COND から ELSE までのブロックをアセンブルし，ELSE 以降 ENDIF/ENDC までのブロックを無視します．また，ELSE ステートメントがない場合，IFxxxx/COND から ENDIF/ENDC までのブロックをアセンブルします．

●評価結果が偽である場合，ELSE ステートメントがあれば，IFxxxx/COND から ELSE までのブロックを無視し，ELSE 以降 ENDIF/ENDC までのブロックをアセンブルします．また，ELSE ステートメントがない場合，IFxxxx/COND から ENDC までのブロックを無視します．

●条件ステートメントの真偽の評価については次のページに述べます．

●条件は 255 レベルまでネスト(入れ子)することができます．

●条件ステートメントの中のアーギュメントはパス 1 で解決できるものでなければなりません．IF，IFT，COND 及び IFF，IFE の＜式＞はすでに定義された値を用い，かつ＜式＞の値が絶対値でなくてはなりません．また，IFDEF，または IFNDEF のアーギュメント中のシンボルが IFDEF，または IFNDEF 以降で定義されている場合，パス 1 では未定義となりますが，パス 2 で定義済みとなります．

●条件疑似命令では偽の条件が成立する場合，真の条件の場合とは別のコードを生成するために ELSE ステートメントを使うことができます．

● ELSE は IFxxxx または COND に対して 1 つだけ使用することができます．
2 つ以上の ELSE を持つ条件，または条件がない ELSE は“C”エラーを引き起こします．

IFxxxx~ELSE~ENDIF/COND~ELSE~ENDC(条件)

書　式　IF　　<式>
IFT　　<式>
COND　　<式>

解　説　●<式>がゼロ以外の値に評価された場合，条件ブロック内のステートメントをアセンブルします．

書　式　IFE　　<式>
IFF　　<式>

解　説　●<式>が0と評価された場合，条件ブロック内のステートメントをアセンブルします．

書　式　IF1

解　説　●アセンブルがパス1を実行中の場合，条件ブロック内のステートメントをアセンブルします．IF1はアーギュメントを持ちません．

書　式　IF2

解　説　●アセンブラがパス2を実行中の場合，条件ブロック内のステートメントをアセンブルします．IF2はアーギュメントを持ちません．

書　式　IFDEF　<シンボル>

解　説　●<シンボル>が定義，または外部参照宣言されている場合，条件ブロック内のステートメントをアセンブルします．

書　式　IFNDEF　<シンボル>

解　説　●<シンボル>が定義も，外部参照宣言もされていない場合，条件ブロック内のステ

ートメントをアセンブルします.

書　式　IFB　　<アーギュメント>

解　説
- ●<アーギュメント>は山形かっこで囲みます.
- ●<アーギュメント>がブランク(何も与えられていない)またはヌル(2つの山形カッコの中に何も入っていない,<>)の場合，条件ブロック内のステートメントをアセンブルします.
- ●IFB(及びIFNB)は，通常マクロブロックの内部で使用されます.
  IFB疑似命令に続く式はたいていダミーシンボルです．マクロが呼び出されると，ダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます．マクロ呼び出しでダミーに対応するパラメータを指定しない場合，そのブロックはアセンブルされます.

書　式　IFNB　　<アーギュメント>

解　説
- ●<アーギュメント>は山形かっこで囲みます.
- ●<アーギュメント>がブランクでない場合，条件ブロック内のステートメントをアセンブルします.
- ●IFNB(およびIFB)は通常マクロブロックの内部で使用されます．IFNB疑似命令に続く式はたいていダミーシンボルです．マクロが呼び出されるとダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます．マクロ呼び出しでダミーに対応するパラメータを指定した場合，そのブロックはアセンブルされます.

書　式　IFIDN　　<アーギュメント1>,<アーギュメント2>

解　説
- ●<アーギュメント1>および<アーギュメント2>は山形かっこで囲みます.
- ●ストリング<アーギュメント1>がストリング<アーギュメント2>と同一であると，条件ブロック内のステートメントをアセンブルします.
- ●IFDIF(およびIFIDN)は通常マクロブロック内部で使用されます．IFIDNに続くアーギュメントの1つはたいていダミーシンボルで，もう1つはストリングです．マクロが呼び出されるとダミーはマクロ呼び出しで渡されるパラメータに置き換えられます．マクロ呼び出しでダミーに対応するパラメータにストリングと同じものを指定すると，そのブロックがアセンブルされます.

書　式　IFDIF　<アーギュメント 1>,<アーギュメント 2>

解　説

●<アーギュメント 1>および<アーギュメント 2>は，山形かっこで囲みます．

●ストリング<アーギュメント 1>がストリング<アーギュメント 2>と異なっている場合，条件ブロック内のステートメントをアセンブルします．

● IFDIF(および IFIDN)は通常マクロブロック内部で使用されます．IFDIF に続くアーギュメントの 1 つはたいていダミーシンボルで，もう 1 つはストリングです．マクロが呼び出されるとダミーはマクロ呼び出しで渡されるパラメータに置き換えられます．マクロ呼び出しでダミーに対応するパラメータにストリングと異なるものを指定すると，そのブロックがアセンブルされます．

第４章　MSX・M-80の実行

4.1 MSX・M-80

MSX・M-80 はソースファイルに作成したソースプログラムをアセンブルし，リローケータブルな(再配置可能)なオブジェクトプログラムを作成します．オブジェクトプログラムはディスクファイルに保存することができます．また，オブジェクトプログラムを保存するディスク上のファイルをオブジェクトファイルと呼び，指定がない限りファイル名拡張子を.REL にします．

オブジェクトプログラム(.REL)は実行可能ではありません．オブジェクトプログラムはMSX・L-80 で処理後，はじめて実行可能となります．

パス

MSX・M-80 はパス 1 とパス 2 の 2 段階でソースプログラムをアセンブルします．

パス 1 ではプログラム中のステートメントの評価をし，オブジェクトコード生成量を計算します．また，プログラム中のシンボルに値を割り当て，シンボルテーブルを生成します．プログラム中にマクロ呼び出しステートメントがあれば，マクロを展開します．

パス 2 では，パス 1 で生成したシンボルテーブルを使用し，オブジェクトコード中のシンボルや式の参照箇所にシンボルや式を代入します．また，マクロを再び展開し，オブジェクトファイル(.REL)にリロケータブルなオブジェクトコードを出力します．

MSX・M-80 は，パス 1，パス 2 でシンボル，式およびマクロの値をチェックします．
パス 2 での値がパス 1 と異なる場合，フェーズエラーとなります．

4.2 MSX・M-80で使用するファイル

MSX・M-80はアセンブル時に次のファイルを使用します．
これらのファイル名は4.3，4.4で述べるMSX・M-80コマンドで指定します．

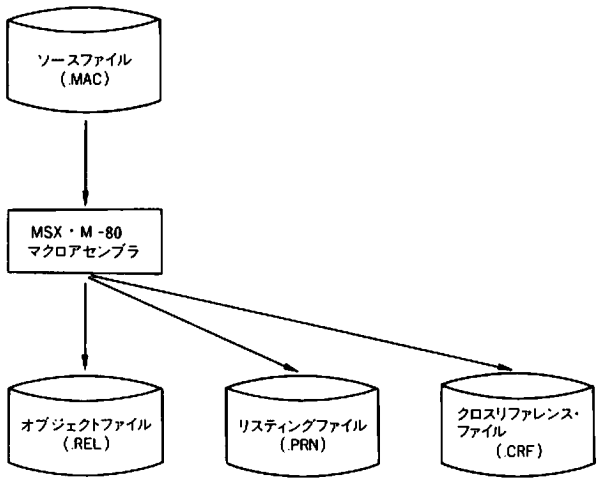


図3-1 MSX・M-80で使用するファイル

(1)ソースファイル

アセンブリ言語で書かれたソースプログラムが入っているファイル．汎用テキストエディタで作成します．

(2)オブジェクトファイル

MSX・M-80でアセンブルした結果であるオブジェクトプログラムを入れるファイル．

(3)リスティングファイル

ソースプログラムと生成したオブジェクトコードのリスティングファイルです．

(4)クロスリファレンス・ファイル

MSX・M-80では/Cスイッチを指定することにより，ディスク上へクロスリファレンス・ファイルを作成することができます．クロスリファレンス・ファイルにはリスティングファイルの情報に加えて内部ラベルのクロスリファレンス情報が含まれており，CREF-80クロスリファレンサを使ってクロスリファレンス・リストを得ることができます．

4.3 MSX・M-80の呼び出し

MSX・M-80はコマンドで呼び出します．
また，このときアセンブル中に使うファイル名やMSX・M-80に対するスイッチをMSX・M-80コマンドで指定します．
MSX・M-80の呼び出し方法は次の2通りがあります．

(1)呼び出し時のコマンドにMSX・M-80コマンドを指定する．

```
A>M80 <MSX・M-80 コマンド>
```

●この呼び出し方法はプログラムを1つだけアセンブルする場合に便利です．この呼び出し方法ではタイピングの手間が少なくてすみます．
●この呼び出し方法を使ってアセンブル処理をBATCHコマンドの一部とすることができます．
●プログラムのアセンブル後，制御はMSX・M-80からオペレーティングシステムへ自動的に戻ります．

(2)呼び出し時のコマンドにパラメータを指定せず，コンパイラのプロンプトに応じてMSX・M-80コマンドを指定する．

```
A>M80
*<MSX・M-80 コマンド>
```

●この呼び出し方法により1台のみのドライブ構成でMSX・M-80を使用することができます．
●また，MSX・M-80の再呼び出しをせずに複数のプログラムを連続してアセンブルすることができます．1つのプログラムのアセンブルが完了すると，MSX・M-80はアスタリスク(*)プロンプトを表示して次のMSX・M-80コマンドを待ちます．
●プロンプトが表示されている時にMSX・M-80を終了する場合は<CTRL+C>をキーインします．

4.4 MSX・M-80のコマンド

4.4.1 MSX・M-80のコマンドの形式

MSX・M-80のコマンドには，アセンブル時に使用するファイルとスイッチを指定します．
コマンドの形式は次のとおりです．

［<オブジェクト>］［,<リスティング>］=<ソース>/<スイッチ>

<オブジェクト>,<リスティング>,<ソース>には4.4.2で述べるファイル名を指定します．

4.4.2 ファイル名の指定

ファイル名の指定方法は次のとおりです．

<ドライブ名>：<パス名>¥<ファイル名>.<拡張子> または <論理デバイス名>

例　A：¥SRC¥SAMPLE.MAC

ドライブ名は1文字のアルファベットで，直後にコロン(：)を付けます．
ドライブを省略するとカレントドライブを仮定します．
従って，ファイルがカレントドライブ上にある場合ドライブ名の指定は必要ありません．
パス名も省略可能でその場合カレントディレクトリを仮定します．
ファイル名には下に述べるデバイスファイル(CON，PRNなど)を指定することもできます．

CON	コンソールのスクリーンまたはキーボード
PRN	プリンタ

拡張子

拡張子は1～3文字の英数字です．拡張子の前にはピリオド(.)を置きます．拡張子がないファイルの場合，ファイル名の後にピリオドを指定します．

例　A：SAMPLE.

拡張子は付けなくてもよく，また任意の名称にすることができますが，拡張子を省略すると標準的な拡張子を仮定しますので次のようなMSX-DOSでの標準的な拡張子を付けることをおすすめします．

拡張子	ファイルの種類
.MAC	MSX・M-80 ソースファイル
.REL	オブジェクトファイル
.PRN	リスティングファイル
.CRF	クロスリファレンスファイル

論理デバイス名

論理デバイス名は3文字で，直後にコロンをつけて指定します．MSX・M-80 で使用できる論理デバイスは次のとおりです．

TTY：　標準入出力
PRN：　標準プリンタ出力

MSX・M-80 スイッチはコマンドの後に任意の数だけ指定することができます．スイッチについては4.5で述べます．

例　M80 SAMPLE，SAMPLE = SAMPLE/R/C

ファイル名はその一部，あるいは全部を省略することができます．次にアセンブラで使用するファイルとその省略の方法を述べます．

(1)ソースファイル

ソースファイルの指定は省略することができません．しかし，ファイル名のうちドライブ名，パス名と拡張子を省略することができます．ドライブ名の省略時にはカレントドライブを，パス名の省略時にはカレントディレクトリを，また，拡張子の省略時には .MAC を指定したものとみなします．拡張子が .MAC でない場合拡張子を省略することはできません．カレントドライブがA：であるとき次のファイル名は同じです．

例　A：SAMPLE.MAC
A：SAMPLE
SAMPLE.MAC
SAMPLE

(2)オブジェクトファイル

オブジェクトファイルの指定はオプションです．ドライブ名を省略するとカレントドライブを指定したとみなされます．パス名を省略すると，カレントディレクトリ中のファイルとみなされます．また，拡張子を省略すると .REL を指定したとみなされます．

オブジェクトファイル名とリスティングファイル名を省略すると，拡張子が .REL であるこ

とを除いてソースファイルと同じファイル名を仮定します．オブジェクトファイルをソースファイル名と異なるファイル名にする場合，オブジェクトファイル名の指定は省略できません．
また，オブジェクトファイルとリスティングファイルを作成する場合，オブジェクトファイルの省略はできません．カレントドライブがA：のとき次のファイル名は同じです．

例　A：SAMPLE.REL
　　A：SAMPLE
　　SAMPLE

(3)リスティングファイル
リスティングファイルの指定はオプションです．リスティングファイル名の前にはカンマ(,)を置きます．リスティングファイル名を省略すると，/Lスイッチを指定していない限りリスティングファイルは作成されません．/Lスイッチについては4.5を参照下さい．
ドライブ名を省略するとカレントドライブを指定したものとみなし，パス名を省略すると，カレントディレクトリ中のファイルとみなします．また，拡張子を省略すると.PRNを指定したものとみなします．

4.4.3 MSX・M-80コマンドの例

MSX・M-80のコマンドは次のように省略することができます．

●オブジェクトファイルのみ作成する．

=<ソースファイル名>
<オブジェクトファイル名>=<ソースファイル名>

=<ソースファイル名>のみの場合，オブジェクトファイルがソースファイルと同じディスクの同じディレクトリ中に作成されます．オブジェクトファイル名は拡張子が.RELであることを除いてソースファイル名と同じです．

例　M80 = NEIL

上の例はソースファイルNEIL.MACを入力し，オブジェクトファイルNEIL.RELを出力します．

●オブジェクトファイルやリスティングファイルを作成しない．

，=<ソースファイル名>

MSX・M-80はソースプログラムのシンタックスチェックを行います．オブジェクトファイル

やリスティングファイルを作成しないのでアセンブル時間が短く，エラーをより早く知ることができます．

例　M80 , = NEIL

4.5 スイッチ

スイッチはMSX・M-80に対して追加機能や代替機能を指示します．
スイッチはMSX・M-80コマンドの最後に指定します．
スイッチはいくつでも指定することができますが，各スイッチの前にはスラッシュ(/)をいれます．

例　M80 = NEIL/L/R

MSX・M-80で使用できるスイッチは次のとおりです．

表4-1 スイッチ一覧

スイッチ	説　明
/O	リスティングファイルのアドレス表示を8進数にする．
/R	ソースファイルと同じ名前(拡張子は.REL)のオブジェクトファイルをソースファイルと同じディレクトリ上に作成します．/RはMSX・M-80コマンドで，オブジェクトファイル名を指定する代わりに使用できます．次の2つ例ではNEIL.RELというオブジェクトファイルが作成されます． 例) M80 ,NEIL = NEIL/R M80 ,= NEIL/R
/L	ソースファイルと同じ名前(拡張子は.PRN)のリスティングファイルをソースファイルと同じディレクトリ上に作成します．/LはMSX・M-80コマンドで，リスティングファイル名を指定する代わりに使用できます． 次の3つの例ではNEIL.PRNというリスティングファイルが作成されます． 例) M80 = NEIL/L M80 ,= NEIL/L M80 NEIL = NEIL/L
/C	CREF-80クロスリファレンス機能で使用する，クロスリファレンス情報ファイルをソースファイルと同じ名前(拡張子は.CRF)にして作成します．クロスレファレンス機能を使用する場合，/Cスイッチを指定してアセンブルしなければなりません．詳細は第6章　CREF-80クロスリファレンサを参照してください．
/Z	Z80命令コードのアセンブルを指示します．/Zはプログラム中で.Z80疑似命令を指定したのと同じ働きをします．

スイッチ	説　明
/I	8080 命令コードのアセンブルを指示します. /I はプログラム中で .8080 疑似命令を指定したのと同じ働きをします.
/P	/P を指定するごとに, アセンブル時に使用するスタック領域を 256 バイト追加割当します. /P はアセンブル時にスタック・オーバーフローエラーが起こった場合に使用し, それ以外の場合には必要ありません.
/M	DS(領域定義)疑似命令によって定義するデータエリアをゼロに初期設定します. /M スイッチを指定しない場合, このデータエリアはゼロに初期設定されず, プログラム開始時のデータエリアの内容は保証されません.
/X	偽条件のリスティングを抑止します. /X の指定がない場合, 偽条件ブロックをリスティングします. /X は .TFCOND 疑似命令と関連して使用される場合があります. 詳細は 3.6.5　リスティング疑似命令を参照ください.

4.6 メッセージ

アセンブル終了後に次のメッセージが出力されます.

{xx / No} Fatal Error(s)　[, yy warning(s)]

xx は重大な文法エラーの件数です.

また yy は警告エラーです.

メッセージはアセンブル終了時にコンソール及びリスティングファイルに表示されます.

プログラムにエラーがなければ“No Fatal Error(s)”が出力されます.

4.7　リスティングフォーマット

MSX・M-80 のリストは次の 2 つの部分に分かれています.

●ソース行リスト
ソースプログラムと生成したオブジェクトコードがプリントされます. また, プログラム中にエラーがあれば, エラーコードがエラーの発生した行に表示されます.

●シンボルテーブルリスト
プログラム中のすべてのマクロ名とシンボルがそれぞれアルファベット順にリストされます.

以下, リスティングフォーマットについて解説します. サンプルリストを参考にしてください.

4.7.1　ソース行リスト

各ページの先頭 3 行には見出しがプリントされます. 見出しの形式は次のとおりで, 3 行目は空白行です.

```
[タイトル]  MSX・M-80  z.zz  dd-mmm-yy  PAGE  x
[サブタイトル]
```

(1)タイトル
TITLE 疑似命令で指定したタイトルがプリントされます. TITLE 疑似命令を指定しない場合は空白となります.

(2)サブタイトル
SUBTTL 疑似命令で指定したサブタイトルがプリントされます. SUBTTL 疑似命令を指定しない場合は空白となります.

(3)バージョン番号
プログラムのバージョン番号をプリントします.

MSX・M-80　z.zz(z.zz はバージョン番号)

(4)ページ番号
PAGE x(x はページ番号). ただしシンボルテーブルリストでは PAGE S と表示されます.

各ソース行は次の形式でプリントされます.

```
[E]       XXXXm      XX XXXXm    [W] X ...  X
 ↓          ↓          ↓           ↓
[エラー] [ロケーション] [オブジェクト]  [テキスト]
```

(5)エラー

ソースステートメントにエラーがある場合のみ，1文字のエラーコードを表示します．
エラーがない場合，ここは空白です．

(6)ロケーション

XXXX はロケーションカウンタの値です．数値は4桁の16進数，または6桁の8進数で，基数はMSX･M-80コマンドの/Hまたは/Oスイッチで決められます．スイッチが指定されていない場合のデフォルトは16進数です．

m はモード属性の表示です．モード属性に応じて次のような文字がプリントされます．

'	**コード相対モード**
"	**データ相対モード**
!	**コモン相対モード**
空白	**絶対モード**
*	**外部参照**

(7)オブジェクト

XX および XXXX は生成したオブジェクトコードです．XX は1バイトの値で2桁の16進数または3桁の8進数，XXXX は2バイトの値で4桁の16進数または6桁の8進数で表示されます．2バイトの値は上位バイトが先に(つまり実際にストアされるのとは逆に)表示され，先に述べた，モード属性を表わす文字が付きます．

(8) [W]

他のファイルからインクルード疑似命令で挿入されたステートメント，あるいは(MACRO, REPT, IRP, IRPC で)展開されたステートメントを表わします．INCLUDE 疑似命令の場合には“C”が，また展開されたステートメントには“+”が表示されます．

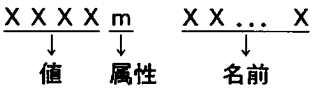
(9)テキスト

ソースプログラムのテキストがプリントされます．

4.7.2 シンボルテーブルリスト

ソース行リストと同じ形式の見出しが各ページの先頭に表示されます．ただし，ページは“PAGE S-xx”のように表示されます．

まず，マクロ名がアルファベット順に表示され，次に，シンボルが同じくアルファベット順に表示されます．シンボルは次のとおり表示されます．



(10)シンボルの値および属性

シンボルの値と属性です．値は4桁の16進数または6桁の8進数で表示されます．また，シンボルの属性によって次のような文字が付けられます．

I	PUBLIC シンボル
U	未定義シンボル
C	COMMON ブロック名．値はコモンブロックの大きさです．
空白	絶対値
'	コード相対値
"	データ相対値
!	コモン相対値
*	外部シンボル

(11)シンボル名

シンボル名の先頭16文字までが表示されます．

アセンブルリスト例

```
Some Useful Subroutines MSX･M-80 2.00   dd-mmm-yy       PAGE    1←(4)
 ↑                                (3)↑
(1)                        TITLE   Some Useful Subroutines

                           ENTRY   MUL10,DMUL10,DIV10
                         ;
(6)                      ; MUL10 - [A]=[A]*10. [B] is destroyed.
 ↓                       ;
0000'←(6)                MUL10:
0000'   47 ←(7)             (9)→ LD      B,A
0001'   87                        ADD     A,A
0002'   87                        ADD     A,A
0003'   80                        ADD     A,B
0004'   87                        ADD     A,A             ;[A]=([A]*2*2+[A])*2
0005'   C9                        RET
                         ;
                         ; DMUL10 - [HL]=[A]*10
                         ;
0006'                    DMUL10:
0006'   C5                        PUSH    BC
0007'   4F                        LD      C,A
0008'   06 00                     LD      B,0
000A'   69                        LD      L,C
000B'   60                        LD      H,B             ;Set up [BC] and [HL]
000C'   29                        ADD     HL,HL
000D'   29                        ADD     HL,HL
000E'   09                        ADD     HL,BC
000F'   29                        ADD     HL,HL           ;[HL]=([HL]*2*2+[BC])*2
0010'   C1                        POP     BC
0011'   C9                        RET
                         ;
                         ; Divide by 10
                         ;  [B]=[A]/10,[A]=[A] MOD 10
                         ;
0012'                    DIV10:
0012'   06 00                     LD      B,0             ;Clear quot.
0014'                    DVLOP:
0014'   04                        INC     B
0015'   D6 0A                     SUB     10              ;Try SUB
0017'   30 FB ←(11)               JR      NC,DVLOP        ;OK,then another try
0019'   05                        DEC     B               ;OOPS! Exceed 0,
001A'   C6 0A                     ADD     A,10            ;  then adjust stuff
001C'   C9                        RET                     ;  & return
                                  END
```

シンボルテーブル例

```
Some Useful Subroutines MSX･M-80 2.00   dd-mmm-yy       PAGE    S

Macros:
Symbols:
0012I'  DIV10           0006I'  DMUL10          0014'   DVLOP
0000I'  MUL10
(10)↑   (11)↑
No Fatal error(s)
```

4.8 エラーコードとエラーメッセージ

プログラム中にエラーがある場合，リスティングファイルとコンソールにエラーコードまたはエラーメッセージが表示されます．

エラーコードはリスティングファイルの該当ステートメントの行のカラム1に表示されます．また，エラーメッセージはリスティングファイルの終わりにプリントされます．

エラーコードの付いた行やエラーメッセージは，リスティングファイルの出力とは別にコンソールに出力されます．

表　4-2　エラーコード一覧

エラーコード	説　明
A	Argument error 疑似命令に対するアーギュメントの形式が正しくないか，指定値が範囲外である．
C	Conditional nesting error IFのないELSE，IFのないENDIF，IFに対して2つのELSEがある，CONDのないENDCなどの場合です．
D	Double Defined symbol 参照するシンボルが2箇所以上で定義されている．
E	External error エクスターナルシンボルが誤って使用されている． 例)FOO　DEFL　NAME ##
M	Multiply Defined symbol 定義しようとしたシンボルはすでに定義済みです．
N	Number error 数値の表現が間違っています． 例)8Q
O	Bad opcode or objectionable syntax ● ENDMやLOCALがマクロ定義ブロック外にある． ● SET，EQU，MACROにシンボル名やマクロ名を指定していない． ●命令コードが間違っているか，式の書式が間違っています． 例)かっこや引用符が対になっていない，演算子が連続している．
P	Phase error ラベル，またはEQUのシンボル名の値がパス1とパス2とで異なる．

エラーコード	説　明
Q	Questionable 行が正しく終了していない場合などです．これは警告エラーです． 例）　LD　A,B,
R	Relocation 式内のリロケーションを間違って使用した場合など． 例)＜アブゾリュート＞-＜リラティブ＞ データセグメント，コードセグメント，コモンセグメントの各領域はリロケータブルです．
U	Undefined symbol 式中で使用しているシンボルは定義されていません．一部の疑似命令では，パス１でＶエラーが表示され，パス２でＵエラーが表示されます．Ｖエラーの説明を参照ください．
V	Value error パス１で決まる値を持たなければならない疑似命令(たとえば.RADIX, PAGE, DS, IF, IFE など)が定義されていない値を持っている．このステートメント以降でシンボルが定義されていれば，パス２ではＵエラーは表示されません．

表　4-3　エラーメッセージ一覧

エラーメッセージ	説　明
%No END statement	END文がない．またはEND文が偽条件ブロックやIRP/IRPC/REPTブロックの中にあり，読みとられなかった．
Unterminated conditional	条件ブロックが終了していない．
Unterminated REPT/IRP /IRPC/MACRO	マクロブロックが終了していない．
Symbol table full	シンボルテーブルを作成中にメモリが不足した．一般的にマクロ定義ブロックの数が多い場合に発生する．マクロ定義ブロックは内部のコメントも含めてシンボルテーブルにストアされるので，コメントの前のセミコロン(；)をセミコロン2つ(；；)に変えるとアセンブルできることがある．
{xx | No} Fatal errors [, yy warnings]	xxは重大なエラーの数，また，yyは警告エラーの数です．このメッセージはアセンブル終了時にコンソールとリスティングファイルへ出力されます．

第５章　MSX・L-80リンクローダ

5.1 MSX・L-80

MSX・M-80や他のコンパイラで作成したオブジェクトファイル(.RELファイル)はまだ実行可能ではありません．MSX・L-80のリンクローダは.RELファイルに入っているオブジェクトモジュールをリンクロードし，実行可能なプログラムに変換してそのまま実行したり，実行可能ファイル(.COMファイル)としてディスクにセーブしたりします．

リンクローダは次の機能を持っています．

●相対アドレスを絶対アドレスに変換する．

アセンブラやコンパイラで作成したオブジェクトモジュールにはラベルや変数名がモジュール内での相対アドレスの形で含まれています．MSX・L-80はオブジェクトモジュールをメモリ上にロードし，相対アドレスへロードアドレスを加えて絶対アドレスに変換します．

●モジュール間のグローバルリファレンス(外部参照)を解決する．

あるモジュールから別のモジュールを呼び出したり，別のモジュールのラベルや変数名を参照することをグローバルリファレンス(外部参照)といいます．

リンクロードしようとするモジュールが別にアセンブルまたはコンパイルしておいたモジュールを呼び出したり参照したりする場合(モジュール中にEXTERNALシンボルがある場合)，呼び出し先や参照先の絶対アドレスが呼び出し元や参照元へ代入されます．

●高級言語で作成したプログラムにランタイムライブラリをリンクする．

MSX-Cなどのコンパイラで作成したオブジェクトモジュールにランタイムライブラリをリンクします．高級言語用のランタイムライブラリやランタイムライブラリのリンク方法については，各コンパイラのユーザーズ・マニュアルを参照してください．

●実行可能プログラムのセーブ

リンクロード後の実行可能プログラムをディスク上の実行可能ファイル(.COMファイル)へセーブすることができます．セーブした実行可能プログラムはオペレーティングシステムのコマンドレベルでファイルを指定するだけで呼び出すことができます(このとき，拡張子.COMを入力する必要はありません．)

例　B：SAMPLE

5.2 MSX・L-80で使用するファイル

リンクロード時に使用するファイルは次のとおりです.

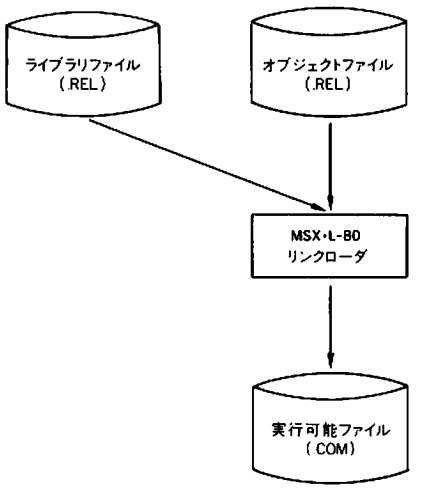


図5-1 MSX・L-80で使用するファイル

(1)オブジェクトファイル

コンパイル，あるいはアセンブル済みのオブジェクトモジュール(オブジェクトプログラム)が入っているファイル．省略時の拡張子は.REL です.

リンクロードでは複数個のオブジェクトファイルの入力が可能です．オブジェクトファイルを指定しないと“NOTHING LOADED”のメッセージが表示されます.

(2)実行可能ファイル

リンクロード後の実行可能プログラムが入るファイル.省略時の拡張子は.COM.実行可能ファイルの指定はオプションです.

(3)ライブラリファイル

オブジェクトファイルを LIB-80 ライブラリマネージャを使って集め，ライブラリ形式にしたもの．ライブラリファイルの指定があり，リンクロード時に未解決のグローバルリファレンス(外部参照)があれば，ライブラリファイルが検索され必要なオブジェクトモジュールがメモリ上にロードされます．ライブラリファイルの指定はオプションです.

5.3　MSX・L-80の呼び出し

MSX・L-80リンクローダはコマンドで呼び出します．また，リンクロード時に使うファイル名やリンクローダに対するスイッチはMSX・L-80コマンドで指定します．MSX・L-80コマンドの指定の方法は次の2通りがあります．

(1)リンクローダを呼び出すコマンドにMSX・L-80コマンド列を指定する．

```
A>L80 <MSX・L-80 コマンド列>
```

例

```
A>L80 MYPROG, SUB1, MYPROG/N/E
```

(2)リンクローダを呼び出すコマンドにはMSX・L-80コマンド列を指定せず，リンクローダのプロンプト(*)に応じて指定する．

```
A>L80
*<MSX・L-80 コマンド列>
*<MSX・L-80 コマンド列>
  ・
  ・
  ・
```

MSX・L-80コマンドをリンクローダのプロンプトに応じて指定する場合，コマンドをいくつかに分けて指定することができます．リンクロードの終了は/E,/Gスイッチを指定した時です．プロンプトはリンクロードを終了するまで表示されます．次の例は(1)と同じ結果になります．

例

```
A>L80
*MYPROG
*SUB1
*MYPROG/N
*/E
```

ディスクドライブが1台のときにリンクしようとするオブジェクトファイルが複数のディスクに入っている場合や，実行可能ファイルをオブジェクトファイルとは別のディスクに作成する場合，異なるディスクへの入出力のたびにディスクドライブのディスクを交換しなければなりません．

5.4 MSX･L-80コマンド

5.4.1 MSX･L-80コマンドの形式

MSX･L-80コマンドでは，リンクロード時に使用するファイルとリンクローダに対するスイッチを指定します．MSX･L-80コマンドの形式は次のとおりです．

＜オブジェクトファイル名＞［,＜オブジェクトファイル名＞］...

スイッチの指定方法については5.5で説明します．

5.4.2 ファイル名の指定

ファイル名の指定方法は次のとおりです．

＜ドライブ名＞：＜パス名＞¥＜ファイル名＞.＜拡張子＞

例　A：¥OBJ¥SAMPLE.REL

●ドライブ名はファイルの入出力装置の指定です．ドライブ名の直後にはコロン(：)を付けます．ドライブ名を省略するとカレントドライブが仮定されます．
●パス名を省略すると，カレント・ディレクトリが仮定されます．
●拡張子は3バイト以内の文字列です．拡張子の前にはピリオドを置きます．

例　A：SAMPLE

拡張子は任意の名称にすることも，また付けないこともできますが，省略すると次のようなMSX-DOSの標準的な拡張子が仮定されますので，これに合わせた拡張子を付けることをおすすめします．

拡張子	ファイルの種類
.REL	オブジェクトファイル
.COM	実行可能ファイル
.HEX	Intel HEX 形式実行可能ファイル

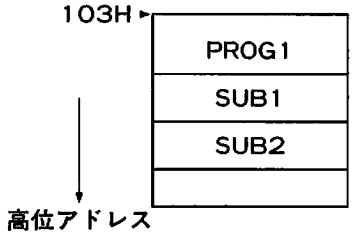
5.4.3 ローディングの順序

ローディングは次の規則に従って行われます．

- MSX･L-80 コマンドに指定した順序でオブジェクトファイルのロード，またはスイッチの機能が実行されます．ただし/N,/X,/Y スイッチの処理は/E または/G スイッチ実行時に行われます．スイッチの詳細については 5.5 を参照ください．
- オブジェクトモジュールのロード開始アドレスは/P,/D スイッチの指定をしない限り 103H からです．オブジェクトファイルを MSX･L-80 コマンドに指定するたびにオブジェクトモジュールが順次隣接してロードされます．モジュールの実行順序でファイル名を指定する必要はありません．
- /E または/G スイッチを指定すると，それまでにロードしたモジュール間のグローバルリファレンス(外部参照)の解決を行い，実行可能プログラムの実行開始アドレスへのジャンプ命令を 100H～102H へ設定します．実行開始アドレスはオブジェクトモジュールの配置とは関係ありません．

例　L80 PROG1，SUB1，SUB2，PROG1/N/E

実行可能プログラム PROG1 のローディング結果



5.4.4 MSX・L-80コマンドの例

●オブジェクトファイルのリンクロード後，実行可能プログラムを実行する．

例　*NEIL/G

NEIL.REL をリンクロードし実行します．実行可能ファイルは作成されません．

●オブジェクトファイルのリンクロード後，実行可能プログラムを実行可能ファイルへセーブし実行する．

例　*NEIL，NEIL/N/G

NEIL.REL をリンクロード後，NEIL.COM へセーブし実行する．

●複数オブジェクトファイルをリンクロード後，実行可能プログラムを実行可能ファイルへセーブし実行する．

例　*NEIL/N，BASPROG，ASMSUB1，ASMSUB2/G

ASMSUB1.REL，ASMSUB2.REL，BASPROG.REL をリンクロード後，NEIL.COM へセーブし実行します．

5.5 スイッチ

MSX・L-80 スイッチはリンクローダの動作を指示します．コマンド列には任意の数のスイッチを置くことができますが，スイッチの前にはスラッシュ(/)を入れなければなりません．

例　*NEIL，NEIL/N/G

以下の表は，各スイッチを機能別にまとめたものです．
各スイッチの詳細については表の次に解説します．

表 5-1 スイッチ一覧

機　能	スイッチ	説　明
プログラムの実行	/G /G：<名前>	ロードしたオブジェクトをリンク・実行し，次にオペレーティングシステムへ抜け出す． <名前>を指定すると，プログラム開始アドレスが<名前>で指定したパブリックシンボルのアドレスに設定されます．
MSX・L－80 の終了	/E /E：<名前>	ロードしたオブジェクトモジュールをリンクした後オペレーティングシステムへ抜け出す． <名前>を指定すると，プログラム開始アドレスが<名前>で指定したパブリックシンボルのアドレスに設定されます．
プログラムのセーブ	/N /N：P	実行可能プログラムを/N の直前のファイル名でディスクにセーブします． ：P を指定すると，プログラム領域(CSEG セグメント)のみをセーブします．：P の指定は/X を指定したときのみ有効です．
ライブラリサーチ	/S	/S の直前のファイル名のオブジェクトライブラリを検索します．
グローバルリスティング	/U	未定義の外部参照名をリスティングします．
	/M	外部参照マップをリスティングします．
基数の設定	/O	基数を 8 進数にします．
	/H	基数を 16 進(デフォルト値)にします．
ロード開始アドレスの設定	/P	オブジェクトモジュールのロード開始アドレスを設定します．/D スイッチと共に使用する場合，/P はプログラム領域(CSEG)のロード開始アドレスを設定します．
	/D	データ領域(COMMON，DSEG)のロード開始アドレスを設定します．
リセット	/R	MSX・L-80 を初期状態にリセットします．

機　能	スイッチ	説　明
特殊コード	/X	Intel ASCII16 進形式で実行可能ファイルを作成します. 実行可能ファイルの拡張子は.HEX です. /X スイッチは/N スイッチと共に指定します.
	/Y	シンボルテーブルを出力します. /Y スイッチは/N スイッチと共に指定します.

スイッチの指定位置はスイッチの種類によってつぎのように異なります.

(1)コマンド列の先頭，最後，オブジェクトファイル名の後の任意の場所に置くことができる.
また，独立して1つのコマンドとして指定することもできる.
(/E, /G, /R, /U, /M, /O, /H, /X, /Y)

例
```
*/G
*/M
```

(2)オブジェクトファイル名の後に付加し，そのファイルについて作用する.
(/N, /S)

例
```
*MYLIB/S, MYPROG
*MYPROG, MYPROG/N
```

(3)そのスイッチを作用させたい(複数の)オブジェクトファイルの前に指定する.
(/P, /D)

例
```
*/P：200, FOO
```

/G

/G スイッチはコマンド行に指定したオブジェクトファイルをロード，リンク後，プログラムを実行し，その後に制御をオペレーティングシステムのコマンドレベルへ戻すことを指示します．

●次の例は NEIL.REL をロード，リンクし，NEIL.COM というファイル名で実行可能のプログラムをディスク上にセーブします．また，リンク後のプログラムが実行され，次にオペレーティングシステムへ抜け出ます．プログラムの実行直前に 5.6 で説明するメッセージと“BEGIN EXECUTION”メッセージが出力されます．

例　*L80 NEIL, NEIL/N/G

●実行可能プログラムをセーブする必要がない場合は次のとおりになります．

例　L80 NEIL/G

上の例の場合/N スイッチを指定していないので，実行可能ファイルは作成されまん．
このプログラムをもう一度実行するためには MSX･L-80 をもう一度実行しなくてはなりません．

/G:<名前>

/G：<名前>スイッチは/G スイッチの機能に加えて，プログラムの実行開始アドレスを設定します．

●<名前>はロードしたモジュールの 1 つにあらかじめ定義されているパブリックシンボル(PUBLIC ステートメントで宣言したラベル)です．
● MSX･L-80 はラベルとして<名前>を持つ行をプログラムの実行開始位置とし，<名前>のアドレスへのジャンプ命令が 100H～102H にセットされます．
● MSX･L-80 は自動的につぎのようなプログラム実行開始アドレスを設定しますので，通常は：<名前>の指定の必要はありません．
a. 高級言語(FORTRAN, BASIC, COBOL)のモジュールをリンクロードする場合は高級言語のモジュールをメインプログラムとみなし，このモジュールの先頭の命令をプログラム開始アドレスとします．
b. アセンブリ言語のモジュールをリンクロードする場合は，END <式>ステートメントを持つモジュールをメインプログラムとみなし，<式>をプログラム開始アドレスとします．
●アセンブリ言語のモジュールをリンクロードするとき，次のような場合に：<名前>の指定をする必要があります．
a. END <式>ステートメントを含むアセンブリ言語モジュールが 2 つ以上ある場合．
b. END <式>ステートメントを含むアセンブリ言語モジュールが 1 つも無い場合．

●高級言語プログラムの実行に先だってアセンブリ言語のモジュールの実行をしたい場合，高級言語モジュールの開始点でCALL文などを使用してアセンブリ言語モジュールを呼び出して下さい．

/E

/Eスイッチはリンクロードを終了後，オペレーティングシステムへ抜け出すことを指示します．

●リンクロードを終了後，プログラムの実行をしない場合/Eスイッチを使用します．MSX・L-80の終了は/Eスイッチの他に/Gスイッチのみが使用できます．リンクロード終了直後に5.6で説明するメッセージが出力されます．
●/Gスイッチの項を参照してください．

/E:<名前>

/E：<名前>スイッチは/Eスイッチの機能に加えて，プログラムの実行開始アドレスを設定します．

●<名前>はロードしたモジュールの1つにあらかじめ定義されているパブリックシンボル(PUBLICステートメントで宣言したラベル)です．MSX・L-80はラベルとして<名前>を持つ行をプログラムの実行開始位置とし，<名前>のアドレスへのジャンプ命令が100H～102Hにセットされます．
●：<名前>の指定は/G：<名前>スイッチの項で説明しているような場合のアセンブリ言語モジュールをリンクロードするときに必要になります．

/N

/Nスイッチは，リンクロード後の実行可能プログラムを/Nの直前に指定したファイル名でディスク上にセーブします．

●<ファイル名>は出力となる実行可能ファイル名です．<ファイル名>の拡張子を省略すると，.COMが仮定されます．ただし，/Xを共に指定した場合には，.HEXが仮定されます．
このファイル(実行可能ファイル)をCOMファイルと呼びます．COMファイル名は必ず/Nスイッチの直前に付与してください．
●/Nスイッチの動作は/Eまたは/Gスイッチの機能を実行するまで据え置かれます．/Nスイッチは，/Nスイッチの指定以降に/Eまたは/Gの指定はない限り効果はありません．
●/Nスイッチを指定する場合，MSX・L-80コマンドには1つ以上のリンクロードすべきオブジェクトファイル名と，ディスクへセーブするリンクロード後の実行可能ファイル名を指定しな

ければなりません.

例　L80 NEIL, NEIL/N/G

上の例は最低限必要な項目を指定したコマンドです. 最初のファイル名 NEIL はリンクロードしようとするオブジェクトファイル名です. 2番目の/N を指定したファイル名 NEIL はディスクにセーブするリンクロードの結果の実行可能ファイル(COM ファイル)名です.

●また, コマンドに指定するファイル名の順序は任意です.

例　L80 NEIL/N, BASPROG, ASMSUB1, ASMSUB2/G

上の例はオブジェクトファイルの BASPROG, ASMSUB1, および ASMSUB2 をロード, リンクし, NEIL というファイル名で実行可能ファイルをセーブします.

●/N スイッチを指定して実行可能ファイルをセーブしないと, リンクロード終了後にプログラムを実行する場合に再度リンクロードしなければなりません.

/N:P

リンクロード後の実行可能プログラムのうちプログラム領域(CSEG エリア)のみを/N スイッチの直前のファイル名でディスク上にセーブします. :P の指定は, /X を指定して Intel ASCII 16 進形式のファイルを作成するときのみ有効です.

●<ファイル名>/N は実行可能ファイルのプログラム領域とデータ領域の両方をディスクにセーブしますが, 実行可能ファイルのサイズを小さくする場合などでは/N：P の形式にすることによりプログラム領域部分だけをディスク上にセーブすることができます.

/P:<アドレス>

/P スイッチは, /P スイッチ以降に指定するオブジェクトモジュールのロード開始アドレスを指定します. また, /D スイッチを共に指定しているとき, オブジェクトモジュールのうちプログラム領域(CSEG エリア)のロード開始アドレスを指定します.

●<アドレス>はカレント基数, つまり/O スイッチの指定があれば 8 進で指定し, /H スイッチの指定があれば 16 進で指定します. また, /O も/H も指定していない場合は, 16 進で指定します.

●<アドレス>の後はカンマ(,)で区切ります.

●/P スイッチを指定しない場合, デフォルトのアドレス 103H からオブジェクトモジュールを配置します.

●<アドレス>はロード開始アドレスです.

例　L80 /P：103，NEIL/N/E

●/P スイッチを指定し，/D スイッチを指定していない場合，オブジェクトモジュールのうちデータ領域(DSEG エリアと COMMOM エリア)が/P で指定した＜アドレス＞へロードされ，その後へ隣接してプログラム領域がロードされます．

●/P スイッチは/P スイッチの後に指定するファイルに対して有効ですが，/P スイッチの前に指定したファイルには無効です．/P スイッチは指定したアドレスでロードを開始したいファイルの前に置きます．

●/P スイッチと/D スイッチを指定している場合，オブジェクトモジュールのうちプログラム領域が/P スイッチに指定した＜アドレス＞へロードされます．また，データ領域(DSEG エリアと COMMON エリア)とが/D スイッチに指定した＜アドレス＞からロードされます．
プログラムが CSEG に加えて DSEG，あるいは COMMOM から構成されでいるとき，/P と/D スイッチを一緒に使用することができます．

● ASEG エリアのロードアドレスは/P スイッチの影響をうけません．

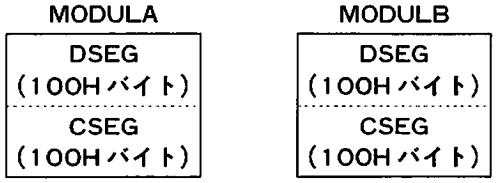
● MSX・L-80 ではセグメントが自動的に順次隣接して配置されますが，同一リンクロードセッション中に 2 つ以上の/P スイッチを指定して複数個のプログラム領域を隣接しないアドレスに配置し，モジュール間に空スペースを作ることができます．ただし空スペースは初期化されません．

●モジュールの配置については次の制約があります．

a. ロードしようとする複数のプログラム領域が重複しないようにアドレスの指定をしてください．重複すると警告エラーメッセージが表示されます．

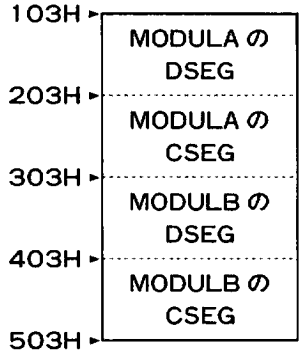
b. プログラムコードエリアがデータエリア(DSEG エリア)またはコモンエリア(COMMOM エリア)によって分割されてはいけません．たとえば 200H から CSEG を配置し，300H から DSEG を配置し，400H から CSEG を配置したりしてはいけません．このような場合エラーメッセージが表示されます．

例　リンクロード前のオブジェクトモジュール



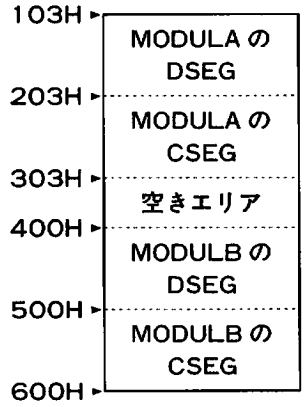
● L80 MODULA, MODULB, PROGX/N/E

PROGX のローディング結果



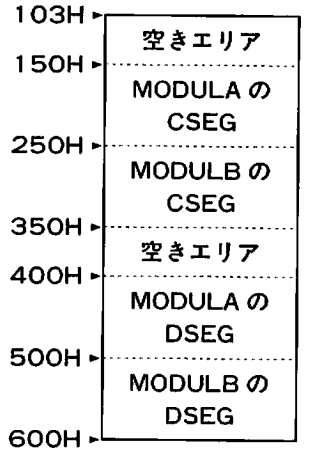
● L80 MODULA, /P : 400, MODULB, PROGX/N/E

PROGX のローディング結果



● L80 /P：150, /D：400, MODULA, MODULB, PROGX/N/E

PROGX のローディング結果



/D:<アドレス>

/D スイッチは/D スイッチの後に指定するオブジェクトファイルのうちデータ領域(DSEG エリアと COMMON エリア)を分離し，指定したアドレスへロードするように指示します．

●<アドレス>はカレント基数，つまり/O スイッチの指定があれば 8 進で指定し，また/H スイッチの指定があれば 16 進で指定します．/O も/H も指定していない場合には 16 進で指定します．
●<アドレス>の後はカンマ(,)で区切ります．

例　L80 /D：103, NEIL, NEIL/N/E

●/D スイッチの指定がない場合，/P スイッチで指定したアドレスからデータ領域のロードを開始します．また，/P の指定もない場合，103H が省略時のアドレスとなります．
●/D スイッチは/D スイッチの後に指定するファイルに対して有効です．/D スイッチの前に指定したファイルには無効です．データ領域を特定のアドレスへロードしたい場合はそのファイルの直前に/D スイッチを指定します．
●/D スイッチは/P スイッチと関連していますので，/P スイッチの項を参照してください．

注　意　リンクロード時にメモリが不足してリンクロードできないプログラムを/P，/D スイッチを併用することによりリンクロードできることがあります．

MSX・L-80 はリンクロード時に相対リファレンスごとに 5 バイトで構成するテーブルを作成しますが，/D 及び/P 指定するとこのテーブルを作成しないので，リンクロ

ード時の必要メモリ量を小さくすることができます．この時の/P,/Dスイッチに指定するアドレスは次のとおりです．

/Pスイッチで指定するアドレス=領域のサイズ＋103H
/Dスイッチで指定するアドレス= 103H

データ領域のサイズはリストを見てDSEGエリアとCOMMOMエリアのサイズを加えて求めます．

/S

/Sスイッチは/Sスイッチの直前に指定したファイルを検索し，ファイルに含まれているモジュールを参照してグローバルリファレンスの解決をはかります．

●/Sスイッチの直前のファイルはLIB-80ライブラリマネージャを使用して作成したライブラリファイルです．
●/Sスイッチの直後はカンマで区切ります．

例　L80 NEIL/N, MYLIB/S, NEIL/G

/R

/RスイッチはMSX･L-80を初期状態にリセットします．

●MSX･L-80は最初にコマンド行をスキャンし,/Rスイッチがあればロードされているすべてのファイルを無視してMSX･L-80を呼び出した直後の状態にリセットします．このときアスタリスク(*)プロンプトが表示され，コマンドの入力待ちとなります．

/U

/Uスイッチは未定義状態のグローバルシンボルをコンソールにリストします．
また，プログラム領域とデータ領域とのそれぞれの先頭アドレスと，最終アドレスおよび領域サイズを表示します．

●/Uは/Gまたは/Eスイッチのどちらも含まれていないコマンド行でのみ有効です．
●未解決状態のグローバルシンボルが多い場合，スクリーンがスクロールアップしてリストが見えなくなることがあります．スクロールアップを止めたい場合は<CTRL+S>を押します．また，これを解除する場合は<CTRL+Q>を押します．

/M

/M スイッチはすべてのグローバルシンボルをコンソールにリストします.
また，プログラム領域とデータ領域とのそれぞれの先頭アドレスと最終アドレスおよび領域サイズを表示します.

●リストはプリンタに出力することはできません.
●グローバルシンボルが定義済みであればグローバルシンボルのあとにその値が表示され，また，未定義であればアスタリスク(*)が表示されます.
●プログラム領域およびデータ領域の先頭・最終アドレスとサイズの値は/P と/D スイッチを指定していないかぎり，1 つのかたまったエリアとしてリストされます.
/P と/D スイッチを指定した場合，プログラム領域とデータ領域とがそれぞれ別々にリストされます.

/O

/O スイッチはカレント基数を 8 進にセットします.

●基数のデフォルト値は 16 進数です./O を省略すると 16 進が仮定されます.

/H

/H スイッチはカレント基数を 16 進にセットします.

●基数のデフォルト値は 16 進ですので,/O スイッチを指定していて 16 進へ変更する場合以外に使用する必要はありません.

/X

/X スイッチは実行可能ファイルをインテル ASCII HEX フォーマットにし,/N の直前のファイル名でセーブします.

●/X スイッチは/N スイッチを指定しているファイルに付与します.

例　L80 NEIL，NEIL/X/N/E

●/X のスイッチを指定している実行可能ファイルの省略時の拡張子は.HEX です.
●/X の主な用途は PROM に焼き込むプログラムを用意する場合です.
● HEX フォーマットの実行可能ファイルはあるマシンから他のマシンへのプログラムの移植を

容易にする目的で作成されました．.HEX フォーマットの形式はオブジェクトコードよりもコードのチェックが多くなります．また，.HEX ファイルはエディタで編集することができます．

/Y

シンボルテーブルを作成し/N の直前のファイル名でセーブします．

●/Y スイッチは/N スイッチを指定しているファイルに付与します．また，/E スイッチの指定を必要とします．

例　L80 NEIL, NEIL/Y/N/E

●/Y スイッチを指定している実行可能ファイルは，.COM ファイルと拡張子.SYM を持つファイルとの2通りが作成されます．ただし，/X スイッチを同時に指定していると，.SYM ファイルと.HEX ファイルとが作成されます．

例　L80 NEIL, NEIL/Y/X/N/E

5.6 メッセージ

リンクロードを終了すると次のメッセージがコンソールに表示されます.

[mmmm　nnnn　yy]

mmmm	実行可能プログラムの先頭アドレス
nnnn	実行可能プログラムの最終アドレス+1
yy	1ページ/256バイトで表わす実行可能プログラムのサイズ(ページ数)

5.7 エラーメッセージ

リンクローダでは次のようなエラーメッセージが用意されています.

表 5-2 エラーメッセージ一覧

エラーメッセージ	説　明
?No start Address	/Gスイッチが指定されたが,メインプログラムがロードされていない.
?Loading Erorr	入力として与えられた最後のファイルがMSX・L-80に対するオブジェクトファイルとして正しい形式ではなかった.
?Out of Memory	プログラムをロードするための充分なメモリがない.
?Commond Erorr	MSX・L-80のコマンドではない.
?<ファイル名> Not Found	コマンドの中で指定された<ファイル名>で示されるファイルが,どこにも存在しない.
%2nd COMMON Larger /XXXXXX/	最初に定義したCOMMONブロック/XXXXXX/が最大のものでなかった. モジュールのローディング順序を変えるか, またはCOMMOMブロックの定義を変える必要がある.
%Mult. Def. Global YYYYYY	パブリック(外部)シンボルYYYYYYが重複して定義されていることがローディング中に判明した.
%Overlaying Program Daea Area ,start = XXXX ,public =<シンボル>(XXXX) ,External =<シンボル>(XXXX)	/Dまたは/Pによって既にロードされているデータを壊す恐れがあります./Dや/Pスイッチで指定したアドレスがMSX・L-80の領域を指していたり, 既にロード済みの領域を指しているときに発生します.

エラーメッセージ	説　明
?Intersecting Program Data Area	プログラムとデータ領域とが重なっており，そのためアドレスまたはグローバルシンボルのチェインエントリも重なっている(だぶっている)．最終的な値がこの重なった領域にあるため，現在の値に変換することができない．
?Start symbol -<名前>- Undefined	/E または/G が指定されたが，その記号が定義されていない．
Origin Above Loader Below Memory, Move Anyway (Y oy N)?	/E または/G が指定されたが，データ，プログラムのいずれかの領域の先頭アドレスがローダメモリー(モジュールをロードするために利用できるメモリ)の外にある．Y↵と答えた場合 MSX･L-80 はプログラムまたはデータの領域を移動させ処理を続行する． その他の答の場合は MSX･L-80 は終了する．いずれの場合においても/N が指定してあれば，ロードイメージは常に保存されることになる．
?Can't Save Object File	ファイルをセーブしようとしたときにディスクエラーが発生した．ディスクがいっぱいであるか，書き込み禁止の場合に発生します．
?Nothing Loaded	オブジェクトファイルがロードされていないのに，<ファイル名>/S または/E または/G が与えられた． 即ち，オブジェクトファイルがロードされていない時に，ライブラリの検索，リンカの処理終了，またはプログラムの実行が要求された．

第6章 CREF-80 クロスリファレンサ

6.1 CREF-80

CREF-80 クロスリファレンサは MSX・M-80 で出力されたクロスリファレンス・ファイルを入力し，プログラム内の内部ラベルとその定義位置をクロスリファレンス・リストへ出力します．このクロスリファレンス・リストはデバッグに役立ちます．

6.2 CREF-80 で使用するファイル

クロスリファレンサの実行時に使用するファイルは次の通りです．

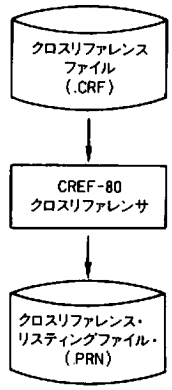


図6-1 CREF-80で使用するファイル

(1)クロスリファレンス・ファイル

MSX・M-80 でプログラムをアセンブルした時に/C スイッチを指定して出力したリスティングファイル．クロスリファレンス・ファイルには MSX・M-80 でセットされた制御文字が組み込まれています．クロスリファレンス・ファイルの省略時の拡張子は.CRF です．

(2)クロスリファレンス・リスティングファイル

クロスリファレンス・リスティングファイルは，MSX・M-80 のリスティングファイルに加えて次の追加情報が付加されています．

a. ソースステートメントの各文にはクロスリファレンス番号が付与されています．

b. リストの最後にシンボルがアルファベット順に並べてプリントされます．さらにシンボルを参照している行と，シンボルの定義をしている行の行番号がプリントされます(定義をしている行の行番号には#が付与されています．)

クロスリファレンス・リスティングファイルの省略時の拡張子は，.PRN です．

この.PRN ファイルはオペレーティングシステムのコマンド(COPY，TYPE コマンド)を使用してプリンタへ出力したり，コンソールのスクリーンへ表示させることができます．

また，CREF-80 は MSX・M-80 と同じ出力装置の指定をサポートします．

6.3　クロスリファレンス・ファイルの作成

クロスリファレンス・ファイルは，MSX・M-80 でプログラムをアセンブル時に MSX・M-80 コマンド列へ MSX・M-80 スイッチの/C を指定して作成します．

例
```
M80 = NEIL/C
```

上の例は NEIL.MAC をアセンブルし，オブジェクトファイル(NEIL.REL)とクロスリファレンス・ファイル(NEIL.CRF)を作成します．

6.4　CREF-80 の呼び出し

CREF-80 クロスリファレンサはコマンドで呼び出します．また，クロスリファレンサの実行時に使うファイル名を CREF-80 コマンドで指定します．CREF-80 コマンドの指定の方法は次の 2 通りがあります．コマンドは小文字で入力しても大文字で入力しても同じです．
小文字は大文字に変換されます．

(1)クロスリファレンサを呼び出すコマンドに CREF-80 コマンドを指定する．

```
A>CREF80 <CREF-80 コマンド>
```

例
```
A>CREF80 PRN = NEIL
```

(2)クロスリファレンサを呼び出すコマンドには CREF-80 コマンドを指定せず，クロスリファレンサのプロンプト(*)に応じて指定する．

```
A>CREF80
*<CREF-80 コマンド>
```

次の例は(1)と同じ結果になります．

例
```
A>CREF80
*PRN = NEIL
```

CREF-80 は CREF-80 コマンドを処理するとアスタリスク(*)プロンプトを表示しますので，

次のCREF-80コマンドを指定することができます.
CREF-80からオペレーティングシステムに抜け出るには<CTRL+C>をキーインします.

6.5 CREF-80コマンド

6.5.1 CREF-80コマンドの形式

コマンドの形式は次のとおりです.

<クロスリファレンス・リスティング・ファイル名>=<クロスリファレンス・ファイル名>

6.5.2 ファイル名の指定

ファイル名の指定方法は次のとおりです.

<ドライブ名>：<パス名>¥<ファイル名>.<拡張子> または <論理デバイス名>

例　A：TMP¥NEIL.CRF

(1)クロスリファレンス・ファイル
クロスリファレンス・ファイルの指定は全部を省略することはできません. ただし, ドライブ名, パス名と拡張子は省略することができます. ドライブ名の省略時にはカレントドライブを, パス名の省略時にはカレントディレクトリを, また, 拡張子の省略時には.CRFが仮定されます.

(2)クロスリファレンス・リスティングファイル
クロスリファレンス・リスティングファイルは指定を全部省略すると, クロスリファレンス・ファイルと同じディレクトリ上にクロスリファレンス・ファイルと同じファイル名(ただし拡張子は.PRN)で作成されます.
一部を指定する場合, ドライブ名を省略するとカレントドライブが, パス名を省略するとカレントディレクトリが, また, 拡張子を省略すると.PRNが仮定されます. さらに, ファイル名を省略した場合, クロスリファレンス・ファイルと同じファイル名が仮定されます. クロスリファレンス・ファイルと異なるドライブあるいはディレクトリに作成する場合, ドライブ名またはパス名を省略することはできません.

例　CREF80 B：= A：NEIL

また，.PRN 以外の拡張子で作成する場合拡張子を省略することはできません．

例　CREF80 NEIL.CRL = NEIL

クロスリファレンス・リスティングファイルでは，ファイル名に次のようなデバイスファイルを指定してプリンタやコンソールに出力することもできます．

CON	コンソール
PRN	プリンタ

論理デバイス名

論理デバイス名は3文字で，次のように直後にコロンをつけて指定します．

TTY：	標準入出力
PRN：	標準プリンタ出力

デバイスファイルとの違いは，標準入出力ではオペレーティングシステムのコマンドでリダイレクトができることです．従って，次のような応用が可能となります．

例　M80 ,TTY：= NEIL/C ｜ CREF80 NEIL=TTY：

6.5.3 CREF-80コマンドの例

●クロスリファレンス・リストをプリンタへ出力する．

例　CREF80 PRN = NEIL

NEIL.CRF を入力し，クロスリファレンス・リストをプリンタへ出力します．(ディスクファイルは作成されません)．

●クロスリファレンス・リストをコンソールに出力する．

例　CREF80 CON = NEIL

NEIL.CRF を入力し，クロスリファレンス・リストをコンソールに出力します(ディスクファイルは作成されません)．

●クロスリファレンス・リスティングファイルをディスクへ出力します.

例　CREF80 NEIL = NEIL
　　CREF80 = NEIL

NEIL.CRF を入力し，クロスリファレンス・リスティングファイルを NEIL.PRN という名前で NEIL.CRF と同じディスク上に作成します.

6.6　リスティング制御疑似命令

クロスリファレンス・リストをオプションでプログラムの全部ではなく一部だけ出力することができます．クロスリファレンス・リストの出力・抑止を制御する場合は，MSX・M-80 の入力となるソースプログラム中で次に掲げるクロスリファレンス・リスティング制御疑似命令を使用してください.

これらの疑似命令はオペレーションフィールドに入力します．他のリスティング制御疑似命令と同様にアーギュメントフィールドの指定はありません．また，これらの疑似命令はプログラム中の任意の場所に置くことができます.詳細は 6.6　リスティング制御疑似命令の項を参照してください.

.CREF

クロスリファレンスを作成します.

.CREF はデフォルト状態です..CREF は,.XCREF 疑似命令を使ってクロスリファレンスを抑止した後，抑止を解除してクロスリファレンスの作成を再開するとき使用します.

.CREF の指定は.XCREF の指定をするまで有効です．しかし，MSX・M-80 コマンドで/C スイッチを指定しないと，クロスリファレンス・ファイルは作成されません.

.XCREF

クロスリファレンスを抑止します.

.XCREF は,.CREF 疑似命令でセットした状態(デフォルト)を解除します..XCREF の効果は,.CREF の指定をするまで有効です.

プログラムの一部でクロスリファレンスの生成を抑止するときに.XCREF を使用してください.

.CREF と.XCREF は MSX・M-80 のコマンドで/C スイッチを指定しないと効果はありませんから,通常のリスティングファイル(クロスリファレンス以外のもの)を出力させたい場合でも.XCREF を使う必要はなく MSX・M-80 コマンドに/C スイッチを指定しないだけでかまいません.

クロスリファレンス・リスト例

```
Some Useful Subroutines MACRO-80 3.44  09-Dec-81       PAGE    1

(1)→1                                          TITLE   Some Useful Subroutines
     2
     3                                         ENTRY   MUL10,DMUL10,DIV10
     4                                    ;
     5                                    ; MUL10 - [A]=[A]*10. [B] is destroyed.
     6                                    ;
     7     0000'                          MUL10:
     8     0000'  47                              LD      B,A
     9     0001'  87                              ADD     A,A
    10     0002'  87                              ADD     A,A
    11     0003'  80                              ADD     A,B
    12     0004'  87                              ADD     A,A
    13     0005'  C9                              RET
    14                                    ;
    15                                    ; DMUL10 - [HL]=[A]*10
    16                                    ;
    17     0006'                          DMUL10:
    18     0006'  C5                              PUSH    BC
    19     0007'  4F                              LD      C,A
    20     0008'  06 00                           LD      B,0
    21     000A'  69                              LD      L,C
    22     000B'  60                              LD      H,B
    23     000C'  29                              ADD     HL,HL
    24     000D'  29                              ADD     HL,HL
    25     000E'  09                              ADD     HL,BC
    26     000F'  29                              ADD     HL,HL
    27     0010'  C1                              POP     BC
    28     0011'  C9                              RET
    29                                    ;
    30                                    ; Divide by 10
    31                                    ;  [B]=[A]/10,[A]=[A] MOD 10
    32                                    ;
    33     0012'                          DIV10:
    34     0012'  06 00                           LD      B,0
    35     0014'                          DVLOP:
    36     0014'  04                              INC     B
    37     0015'  D6 0A                           SUB     10
    38     0017'  30 FB                           JR      NC,DVLOP
    39     0019'  05                              DEC     B
    40     001A'  C6 0A                           ADD     A,10
    41     001C'  C9                              RET
    42
    43                                          END
```

```
    Some Useful Subroutines MACRO-80 3.44   09-Dec-81       PAGE    S

    Macros:

    Symbols:
    0012I'  DIV10           0006I'  DMUL10          0014'   DVLOP
    0000I'  MUL10

    No Fatal error(s)


(2)→DIV10       3       33#
    DMUL10      3       17#
    DVLOP      35#      38
    MUL10       3        7#
```

1) クロスリファレンス番号

2) 変数名と変数名が定義してある行と参照している行のクロスリファレンス番号

第7章 LIB-80 ライブラリマネージャ

7.1 LIB-80

LIB-80 ライブラリマネージャは，オブジェクトモジュールを集めたライブラリファイルを管理するユーティリティプログラムで，高級言語のライブラリファイルや私用のライブラリファイルを作成することができます．LIB-80 の機能は次のとおりです．

●オブジェクトファイル，あるいは既存のライブラリファイルに含まれるオブジェクトモジュールを集めて，新しくライブラリファイルを作成します．
●ライブラリファイルに含まれるオブジェクトモジュール名と，オブジェクトモジュールに含まれているパブリックシンボルの参照，被参照状態をコンソールのスクリーンに表示します．

LIB-80 を使ってオブジェクトモジュールをライブラリファイルに集めておくと，MSX･L-80 でコマンドに<ライブラリファイル名>/S を指定するだけで必要なオブジェクトモジュールのみが実行可能ファイルにリンクされます．この機能はサブルーチンが多いプログラムの場合，MSX･L-80 コマンドで１つ１つオブジェクトモジュールファイルを指定するのに較べて大変便利です．

注　意　LIB-80 は新たなライブラリファイルを作成するのに便利な反面，コマンドの指示を誤ると既存のライブラリファイルを破壊する可能性があります．
MSX･L-80 の使用前に既存のライブラリファイルのバックアップコピーを取っておいてください．

7.2 LIB-80で使用するファイル

LIB-80 ライブラリマネージャの実行時に使用するファイルは次のとおりです.

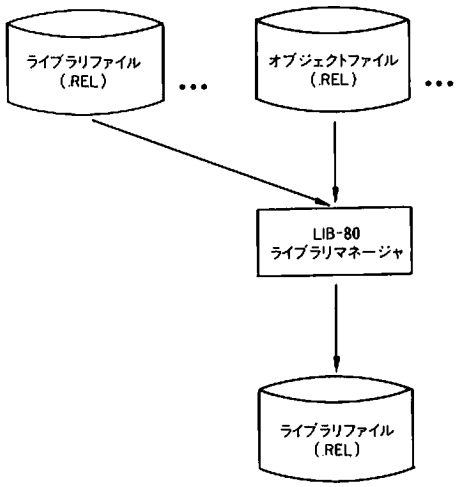


図7-1 LIB-80で使用するファイル

(1)オブジェクトファイル

アセンブリ言語のプログラムをアセンブルした結果，またはFORTRAN-80，COBOL-80，BASICなどのコンパイル結果であるオブジェクトファイルです.

(2)ライブラリファイル

オブジェクトファイルをLIB-80で集めたライブラリイメージのファイルです.

7.3 LIB-80の呼び出し

LIB-80ライブラリマネージャはコマンドで呼び出します．
また，LIB-80の実行時に使うファイル名やスイッチをLIB-80コマンドで指定します．
LIB-80コマンドの指定の方法は次の2通りあります．

(1)ライブラリマネージャを呼び出すコマンドにLIB-80コマンドを指定します．

```
A>LIB80 <LIB-80 コマンド>
```

例　A>LIB80 <LIB-80 コマンド>

(2)ライブラリマネージャを呼び出すコマンドにLIB-80コマンドを指定せず，ライブラリマネージャのプロンプト(*)に応じて指定します．

```
A>LIB80
*<LIB-80 コマンド>
        .
        .
        .
```

この方法ではLIB-80コマンドをいくつかに分けて入力することができます．
LIB-80からオペレーティングシステムへ抜け出す場合には，<CTRL+C>をキーインします．

7.4 LIB-80コマンド

7.4.1 LIB-80コマンドの形式

コマンドの形式は次のとおりです．

[<宛先フィールド>=]<ソースフィールド><スイッチ>

宛先フィールド

宛先フィールドの指定はオプションです．=は宛先フィールドの指定をする場合に必要となります．宛先フィールドにはLIB-80で作成するライブラリファイル名を指定します．

<ライブラリファイル名>

このフィールドを省略するとFORLIB.RELが仮定されます．このため，カレントドライブ上に提供されたFORLIB.RELランタイムライブラリがあるとFORLIB.RELが書き換わります．FORLIB.RELを新しく作成しない限り，宛先フィールドの指定は指定してください．

ソースフィールド

ソースフィールドの指定は省略することはできません．ソースフィールドには宛先フィールドに指定したライブラリファイルへ加えたいオブジェクトファイル名，または既に作成済みのライブラリファイルに含まれているオブジェクトモジュール名です．

＜モジュール指定＞ [,＜モジュール指定＞]...

モジュール指定は次のうちのいずれかです．

＜オブジェクトファイル名＞
＜ライブラリファイル名＞
＜ライブラリファイル名＞ '<' ＜モジュール名＞ '>'

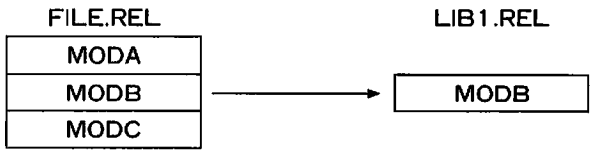
●ライブラリファイルに含まれているモジュール名は山形かっこ(＜＞)で囲みます．
●モジュール名はカンマ(,)で区切っていくつでも指定できます．

例　*FILE1, FILE2＜MODZ＞, FILE3＜MODR＞, FILE4

●ライブラリファイルに含まれているオブジェクトモジュールは次のように指定することができます．

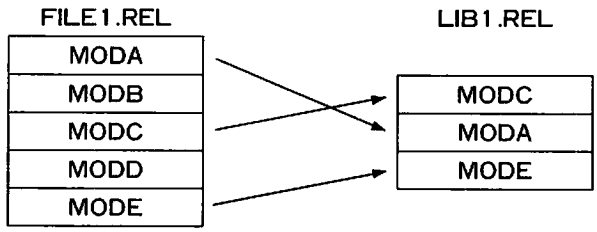
a. ライブラリファイルに含まれているオブジェクトモジュールを1つだけ抜き出す．

例　*LIB1 = FILE＜MODB＞



b. ライブラリファイルに含まれているオブジェクトモジュールをカンマで区切っていくつか抜き出す．

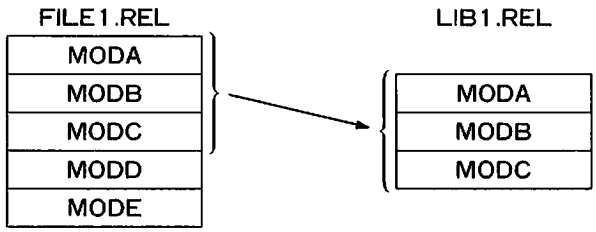
例　*LIB1 = FILE1＜MODC, MODA, MODE＞



オブジェクトモジュールは既存のライブラリファイルでの順序とは関係なく，指定した順序で新しいライブラリファイルへ入ります．

c. ライブラリファイルの先頭から，指定したオブジェクトモジュールまでを抜き出す．

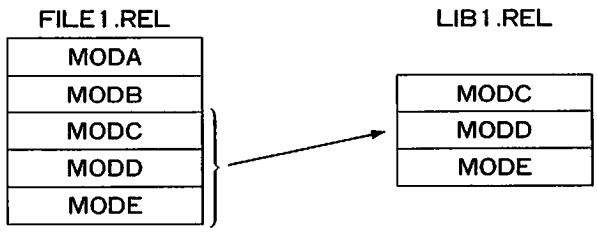
例　*LIB1 = FILE<..MODC>



抜き出す最後のオブジェクトモジュール名の前にピリオドを2つ付けます．

d. 指定したオブジェクトモジュールからライブラリファイルの最後までを抜き出す．

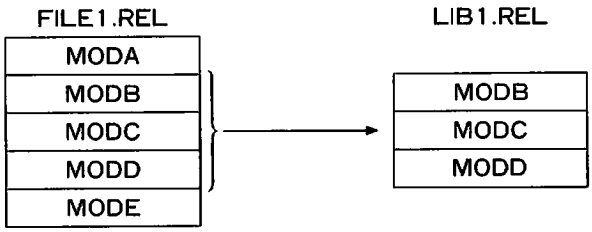
例　*LIB1 = FILE<MODC..>



抜き出す先頭のオブジェクトモジュール名の後にピリオドを2つ付けます．

e. 指定したオブジェクトモジュールから別に指定したオブジェクトモジュールまでを連続して抜き出す．

例　*LIB1 = FILE1＜MODB..MODD＞

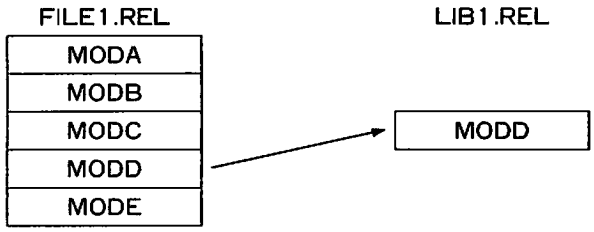


抜き出す先頭のオブジェクトモジュール名と最後のオブジェクトモジュール名の間にピリオドを2つ入れます.

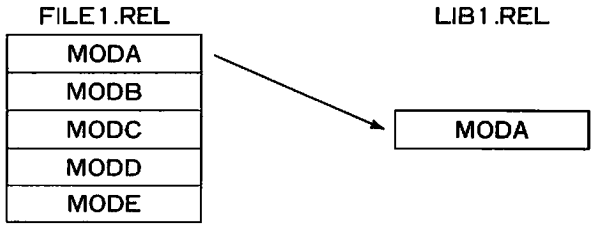
f. 抜き出すオブジェクトモジュールの指定をオブジェクトモジュール名＋オフセット番号,オブジェクトモジュール名-オフセット番号で指定する.

オフセット番号は1～255です.

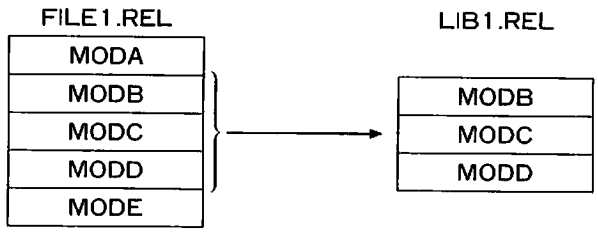
例　*LIB1 = FILE1＜MODB＋2＞



例　*LIB1 = FILE1＜MODD-3＞

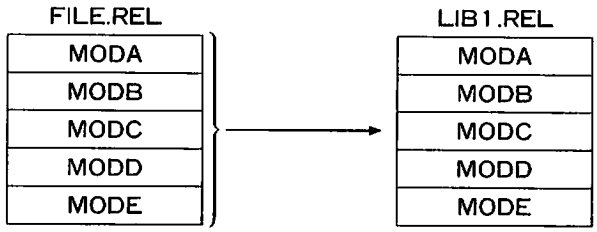


例　　*LIB1 = FILE<MODA+1..MODE-1>



g. ライブラリファイルに含まれているオブジェクトモジュールを全部抜き出す.

例　　*LIB1 = FILE1



7.4.2　ファイル名の指定

ファイル名の指定方法は次のとおりです.

＜ドライブ名＞：＜パス名＞¥＜ファイル名＞.＜拡張子＞

例　A：NEIL.REL

ファイル名の指定方法の詳細はMSX・L-80で説明したものと同じです.

(1)宛先フィールド

宛先フィールドのライブラリファイル名の指定はオプションですが，省略するとカレントドライブ上のFORLIB.RELが仮定されますので注意してください.また，ドライブ名の省略時にはカレントドライブ，パス名の省略時にはカレントディレクトリ，拡張子の省略時には.RELがそれぞれ仮定されます.

(2)ソースフィールド

ソースフィールドにはライブラリファイル名かオブジェクトファイル名を1つ以上指定します.ドライブ名の省略時にはカレントドライブ，拡張子の省略時には.RELがそれぞれ仮定されます.

7.4.3　LIB-80コマンドの例

●オブジェクトモジュールを集めてライブラリファイルを作成する.

例
```
A>LIB80
*TRANLIB=SIN,COS,TAN,ACOG
*OLDLIB<EXP>
*/E
```

SIN.REL，COS.REL，TAN.REL，ACOG.RELおよびOLDLIB.RELに含まれるEXPモジュールを集めてTRANLIB.RELというライブラリファイルを作成します.

●ライブラリファイルのリスティングを表示する.

例
```
A>LIB80
*TRANLIB.LIB/U
*TRANLIB.LIB/L
```

TRANLIB.LIBに含まれているパブリックシンボルのうちリンクロード時に未定義となる可

能性のあるシンボルのリスティングと，TRANLIB.LIB にふくまれているすべてのモジュールとパブリックシンボルのクロスリファレンス・リスティングをコンソールに表示します．

7.5 スイッチ

スイッチはLIB-80に対して追加機能の実行を指定します．スイッチはスラッシュ(/)が付いた文字で，スイッチフィールドへいくつでも指定することができます．スイッチの一覧表を次に示します．

表 7-1 スイッチ一覧

スイッチ	説　明
/E	現在作成中のライブラリを宛先フィールドで指定したファイル名でセーブしオペレーティングシステムへ抜けます．ライブラリファイルと同じ名前のファイルがあれば既存のファイルが削除されます．
/R	現在作成中のライブラリを宛先フィールドで指定したファイル名でセーブします．オペレーティングシステムには抜け出しません．ライブラリファイルと同じ名前のファイルがあれば既存のファイルが削除されます．現在作成中のライブラリファイルをセーブ後，他のライブラリファイルを作成したい場合に使用します．
/L	指定されたファイル内のオブジェクトモジュール名と，オブジェクトモジュールに含まれているグローバルシンボル定義をクロスリファレンス形式でコンソールに表示します．プリンタへ出力したい場合はLIB-80実行前に<CTRL+P>をキーインしてください．
/U	ライブラリファイル中のオブジェクトモジュールに含まれるグローバルシンボルのうち，未定義となる可能性のあるものをコンソールにリストします．
/C	現在作成中のライブラリをクリアし，LIB-80を初期状態に戻します．LIB-80のアスタリスク(*)プロンプトが表示され新しいLIB-80コマンドを待ちます．/CはLIB-80コマンドを間違えてキーインした場合などに使用してください．
/O	/Lスイッチを指定して出力するリスト中の数値の基数を8進にします． /Oは/Lスイッチと共に指定します． 例)NEWLIB/L/O
/H	/Lスイッチを指定して出力するリスト中の数値の基数を16進にします． 16進はデフォルト値ですので，/Hを指定しなくても自動的に16進基数となります．

7.6　注意事項

ライブラリファイルはリンクロード時にロード済みのオブジェクトモジュールだけでは解決できないグローバルリファレンスがあったとき，1パスだけで検索されます．このため，ライブラリファイルへオブジェクトモジュールを集める場合，他のモジュールから参照されるエントリーポイントを持つモジュールが後へ配置されるようにLIB-80コマンドで指示してください．

付 録

A. ASCII キャラクタコード表
B. MSX・M-80 疑似命令表
C. 命令コード一覧表
D. オブジェクトファイルの形成

付録 A　ASCII キャラクタコード表

10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ
000	00H	NUL	032	20H	SPACE	064	40H	@	096	60H	`
001	01H	SOH	033	21H	!	065	41H	A	097	61H	a
002	02H	STX	034	22H	"	066	42H	B	098	62H	b
003	03H	ETX	035	23H	#	067	43H	C	099	63H	c
004	04H	EOT	036	24H	$	068	44H	D	100	64H	d
005	05H	ENQ	037	25H	%	069	45H	E	101	65H	e
006	06H	ACK	038	26H	&	070	46H	F	102	66H	f
007	07H	BEL	039	27H	'	071	47H	G	103	67H	g
008	08H	BS	040	28H	(	072	48H	H	104	68H	h
009	09H	HT	041	29H	)	073	49H	I	105	69H	i
010	0AH	LF	042	2AH	*	074	4AH	J	106	6AH	j
011	0BH	VT	043	2BH	+	075	4BH	K	107	6BH	k
012	0CH	FF	044	2CH	,	076	4CH	L	108	6CH	l
013	0DH	CR	045	2DH	−	077	4DH	M	109	6DH	m
014	0EH	SO	046	2EH	.	078	4EH	N	110	6EH	n
015	0FH	SI	047	2FH	/	079	4FH	O	111	6FH	o
016	10H	DLE	048	30H	0	080	50H	P	112	70H	p
017	11H	DC1	049	31H	1	081	51H	Q	113	71H	q
018	12H	DC2	050	32H	2	082	52H	R	114	72H	r
019	13H	DC3	051	33H	3	083	53H	S	115	73H	s
020	14H	DC4	052	34H	4	084	54H	T	116	74H	t
021	15H	NAK	053	35H	5	085	55H	U	117	75H	u
022	16H	SYN	054	36H	6	086	56H	V	118	76H	v
023	17H	ETB	055	37H	7	087	57H	W	119	77H	w
024	18H	CAN	056	38H	8	088	58H	X	120	78H	x
025	19H	EM	057	39H	9	089	59H	Y	121	79H	y
026	1AH	SUB	058	3AH	:	090	5AH	Z	122	7AH	z
027	1BH	ESC	059	3BH	;	091	5BH	[	123	7BH	{
028	1CH	FS	060	3CH	<	092	5CH	¥	124	7CH	|
029	1DH	GS	061	3DH	=	093	5DH	]	125	7DH	}
030	1EH	RS	062	3EH	>	094	DEH	^	126	7EH	~
031	1FH	US	063	3FH	?	095	5FH	_	127	7FH	DEL

LF =ライン・フィード，FF =フォーム・フィード，CR =キャリッジ・リターン，DEL =ラブアウト

付録B MSX・M-80疑似命令表

MSX・M-80の疑似命令を機能ごとの一覧表に示します．なお，SET疑似命令はZ80モードでは使用できません．SET以外の疑似命令は.Z80モードでも，.8080モードでも使用できます．

命令セットの選択

疑似命令	説　明
.Z80	Z80命令セットモードにする
.8080	8080命令セットモードにする．

データ定義とシンボル定義

疑似命令			説　明
<シンボル>	ASET	<式>	シンボルに<式>の値を割り当て
[<ラベル>]	BYTE EXT	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	BYTE EXTRN	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	BYTE EXTERNAL	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	DB	<式> [,<式>]...	1バイトずつの領域を確保し，初期値をセット
[<ラベル>]	DB	<ストリング> [,<ストリング>]...	1バイトずつの領域を確保し，初期値をセット
[<ラベル>]	DC	<ストリング>	ストリングの領域を確保し，初期値をセット
[<ラベル>]	DEFB	<式> [,<式>]...	1バイトずつの領域を確保し，初期値をセット
<シンボル>	DEFL	<式>	シンボルに<式>の値を割り当て
[<ラベル>]	DEFM	<ストリング> [,<ストリング>]...	1バイトずつの領域を確保し，初期値をセット
[<ラベル>]	DEFS	<式> [,<値>]	領域を確保し，初期値をセット
[<ラベル>]	DEFW	<式> [,<式>]...	2バイトのワード領域を確保し，初期値をセット
[<ラベル>]	DS	<式> [,<値>]	領域を確保し，初期値をセット
[<ラベル>]	DW	<式> [,<式>]...	2バイトのワード領域を確保し，初期値をセット
	ENTRY	<シンボル> [,<シンボル>]...	パブリックシンボル宣言
<シンボル>	EQU	<式>	<シンボル>に<式>の値を割り当て

	EXT	<シンボル> [,<シンボル>]...	外部参照宣言
	EXTRN	<シンボル> [,<シンボル>]...	外部参照宣言
	EXTERNAL	<シンボル> [,<シンボル>]...	外部参照宣言
	GLOBAL	<シンボル> [,<シンボル>]...	パブリックシンボル宣言
	PUBLIC	<シンボル> [,<シンボル>]...	パブリックシンボル宣言
<シンボル>	SET	<式>	シンボルに<式>の値を割り当て

PC モード指定

疑似命令	説　明
ASEG	絶対モードにする
CSEG	コード相対モードにする
DSEG	データ相対モードにする
COMMON /<ブロック名>/	コモン相対モードにする
ORG　<式>	ロケーションカウンタの値を変更
.PHASE <式>	.PHASE から .DEPHASE までのプログラムを
.DEPHASE	<式>で指定したアドレスで実行

ファイル関係

疑似命令		説　明
.COMMENT	<区切り記号><テキスト> <区切り記号>	<区切り記号>ではさまれた<テキスト>をコメントとする
END	[<式>]	プログラムの終了
INCLUDE	<ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
$INCLUDE	<ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
MACLIB	<ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
NAME	('<モジュール名>')	モジュール名定義
.RADIX	<式>	基数の指定(2-16)
.REQUEST	<ファイル名> [,<ファイル名>]...	外部モジュールのリンクロードを要求

リスティングフォーマット制御

疑似命令		説　明
PAGE	[<式>]	リスティングの改ページ
SUBTTL	[<テキスト>]	サブタイトルの指定
TITLE	<テキスト>	タイトルの指定
$TITLE	('<テキスト>')	サブタイトルの指定

一般リスティング制御

疑似命令		説　明
.LIST		リスティングを出力
.XLIST		リスティングの出力を抑止
.PRINTX	<区切り記号><テキスト><区切り記号>	アセンブル中にコンソールメッセージを出力する

条件リスティング制御

疑似命令	説　明
.LFCOND	偽条件ブロックのリスティングを出力
.SFCOND	偽条件ブロックのリスティングの出力を抑止
.TFCOND	/X スイッチの有無により偽条件ブロックのリスティングの出力を決定する

マクロ拡張リスティング制御

疑似命令	説　明
.LALL	マクロ，リピートブロックの展開部分をリスティングする
.XALL	マクロ，リピートブロックの展開部分をリスティングしない
.SALL	マクロ，リピートブロックの展開部分のうち，オブジェクトコードを生成するステートメントをリスティングする

クロスリファレンス情報出力制御

疑似命令	説　明
.CREF	クロスリファレンス情報を出力
.XCREF	クロスリファレンス情報の出力を抑止

マクロ

疑似命令		説　明
<名前>MACRO	[<パラメータ> [,<パラメータ>]...]	マクロ定義を開始
ENDM		マクロ定義を終了
EXITM		マクロの展開を終了
LOCAL	<ダミー> [,<ダミー>]...	マクロ展開時にマクロ定義中のダミーパラメータをユニークなシンボルに置き換える

反復

疑似命令		説　明
PERT	<式>	反復ブロックを指定した数だけ繰り返し展開
IRP	<ダミー>,<山形カッコで囲まれたパラメータ>	反復ブロックを指定したパラメータに従って繰り返し展開
IRPC	<ダミー>,<文字列>	反復ブロックを指定した文字列に従って繰り返し展開

条件付きアセンブリ機能

疑似命令		説　明
COND	<式>	<式>がゼロ以内の値のとき条件は真とみなされ，条件ブロック内がアセンブルされる
ELSE		IFxxxx や COND の条件が偽となったときに展開するステートメントを開始する
ENDC		.COND で開始した条件ブロックを終了する
ENDIF		IFxxxx で開始した条件ブロックを終了する
IF	<式>	<式>がゼロ以外の値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFB	<アーギュメント>	<アーギュメント>がブランクのとき条件は真とみなされ条件ブロック内がアセンブルされる
IFDEF	<シンボル>	<シンボル>が定義されているとき条件は真とみなされ，条件ブロック内がアセンブルされる
IFDIF	<アーギュメント 1>, <アーギュメント 2>	<アーギュメント 1>と<アーギュメント 2>が同一のとき条件は真とみなされ，条件ブロック内がアセンブルされる
IFE	<式>	<式>がゼロの値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFF	<式>	<式>がゼロの値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFIDN	<アーギュメント 1>, <アーギュメント 2>	<アーギュメント 1>と<アーギュメント 2>が同一のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFNB	<アーギュメント>	<アーギュメント>がブランクでないとき条件は真とみなされ，条件ブロック内がアセンブルされる
IFNDEF	<シンボル>	<シンボル>が定義されていないとき条件は真とみなされ，条件ブロック内がアセンブルされる
IFT	<式>	<式>がゼロ以外の値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IF1		アセンブラがパス 1 を実行中，条件が真とみなされ条件ブロック内がアセンブルされる
IF2		アセンブラがパス 2 を実行中，条件が真とみなされ条件ブロック内がアセンブルされる

付録C　命令コード一覧表

Z80命令コード一覧表と，8080命令コード一覧表を示します．

Z80命令コード一覧

命令コード		機　　能
ADC	A	Add with Carry to Accumulator
ADC	HL,rp	Add Register Pair with Carry to HL
ADD		Add
AND		Logical AND
BIT		Test Bit
CALL	Addr	Call Subroutine
CALL	cond,addr	Call Conditional
CCF		Complement Carry Flag
CP		Compare
CPD		Compare,Decrement
CPDR		Compare,Decrement,Repeat
CPI		Compare,Increment
CPIR		Compare,Increment,Repeat
CPL		Complement Accumulator
DAA		Decimal Adjust Accumulator
DEC		Decrement
DI		Disable Interrupts
DJNZ		Decrement and Jump if Not Zero
EI		Enable Interrupt
EX		Exchange
EXX		Excange Register Pairs and Althernatives
HALT		Halt
IM	x	Set Interrupt Mode
IN		Input
INC		Increment
IND		Input,Decrement
INDR		Input,Decrement,Repeat
INI		Input,Increment
INIR		Input,Increment,Repeat
JP	addr	Jump
JP	cond,addr	Jump Conditional

JR		Jump Relative
JR	cond,addr	Jump Relative Conditional
LD	A,(addr)	Load Accumulator Direct
LD	A,(BC)or(DE)	Load Accumulator Secondary
LD	A,I	Load Accumulator from Interrupt Vector Register
LD	A,R	Load Accumulator from Refresh Register
LD	HL,(addr)	Load HL Direct
LD	data	Load Immediate
LD	xy,(addr)	Load Index Register Direct
LD	reg,(HL)	Load Register
LD	reg,(xy+disp)	Load Register Indexed
LD	rp,(addr)	Load Register Pair Direct
LD	SP,HL	Move HL to STack Pointer
LD	SP,xy	Move Index Register to Stack Pointer
LD	dst,scr	Move Register-to-Register
LD	(addr),A	Store Accumulator Direct
LD	(BC)or(DE),A	Store Accumulator Secondary
LD	I,A	Store Accumulator to Interrupt Vector Register
LD	R,A	Store Accumulator ro Refresh Register
LD	(addr),HL	Store HL Direct
LD	(HL),data	Store immediate to Memory
LD	(xy+disp),data	Store Immediate to Memory Indexed
LD	(addr),xy	Store Index Register Direct
LD	(HL),reg	Store Register
LD	(xy+disp),reg	Store Register Indexed
LD	(addr),rp	Store Register Pair Direct
LDD		Load,Decrement
LDDR		Load,Decrement,Repeat
LDI		Load,Increment
LDIR		Load,Increment,Repeat
NEG		Negate (Two's Complement) Accumulator
NOP		No Operation
OR		Logical OR
OUT		Output
OUTD		Output,Decrement
OTDR		Output,Decrement,Repeat
OUTI		Output,Increment
OTIR		Output,Increment,Repeat
POP		Pop from Stack

PUSH		Push to Stack
RES		Reset Bit
RET		Return from Subroutine
RET	cond	Retrun Conditional
RETI		Return from Interrupt
SETN		Retrun from Non-Maskable Interrupt
RL		Rotate Left Through Carry
RLA		Rotate Accumulator Left Through Carry
RLC		Rotate Left Circular
RLCA		Rotate Accumulator Left Circular
RLD		Rotate Accumulator and Memory Left Decimal
RR		Rotate Right Through Carry
RRA		Rotate Accumulator Right Through Carry
RRC		Rotate Right Circular
RRCA		Rotate Accumulator Right Circular
RRD		Rotate Accumulator and Memory Right Decimal
RST		Reset
SET		Set Bit
SBC		Subtract with Carry (Borrow)
SCF		Set Carry Flag
SLA		Shift Left Airthmetic
SRA		Shift Right Airthmetic
SRL		Shift Right Logical
SUB		Subtract
XOR		Logical Exclusive OR

8080 命令コード一覧

命令コード	機　　能
ADC,ACI	Add with Carry
ADD,ADI	Add
ANA,ANI	Logical AND
CALL	Call Subroutine
CC	Call on Carry
CM	Call on Minus
CMA	Complement Accumulator
CMC	Complement Carry
CMP,CPI	Compare
CNC	Call on No Carry
CNZ	Call on Not Zero
CP	Call on Positive
CPE	Call on Parity Even
CPO	Call on Parity Odd
CZ	Call on Zero
DAA	Decimal Adjust
DAD	16-bit Add
DCR	Decrement
DCX	16-bit Decrement
DI	Disable Interrupts
EI	Enable Interrupts
HLT	Halt
IN	Input
INR	Increment
INX	Increment 16 bits
JC	Jump on Carry
JM	Jump on Minus
JMP	Jump
JNC	Jump on Not Carry
JNZ	Jump on Not Zero
JP	Jump on Positive
JPE	Jump on Parity Even
JPO	Jump on Parity Odd
JZ	Jump on Zero
LDA	Load Accumulator

LDAX	Load Accumulator Indirect
LHLD	Load HL Direct
LXI	Load 16 bits
MOV	Move
MVI	Move Immediate
NOP	No Operation
ORA,ORI	Logical OR
OUT	Output
PCHL	HL to Program Counter
POP	Pop from Stack
PUSH	Push from Stack
RAL	Rotate with Carry Left
RAR	Rotate with Carry Rigth
RC	Retrun on Carry
RET	Return from Subroutinbe
RLC	Rotate Left
RM	Return on Minus
RNC	Retrun on No Carry
RNZ	Retunn on Not Zero
RP	Retrun on Positive
RPE	Return on Parity Even
RPO	Return on Parity Odd
RRC	Rotate Right
RST	Restart
RZ	Retrun on Zero
SBB,SBI	Subtract with Borrow
SHLD	Store HL Direct
SPHL	HL to Stack Pointer
STA	Store Accumulator
STAX	Store Accumulator Indirect
STC	Set Carry
SUB,SUI	Subtract
XCHG	Exchange D and E,H and L
XRA,XRI	Logical Exclucive OR
XTHL	Exchange Top of Stack,HL

付録D オブジェクトファイルの形式

この付録はリロケータブル・オブジェクト・ファイルのロード形式を知りたい方のための参考資料です．特に必要がなければ読み飛ばして下さい．

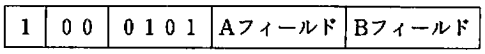
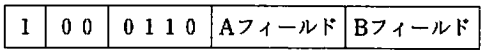
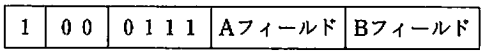
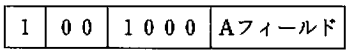
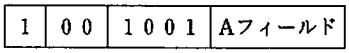
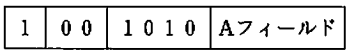
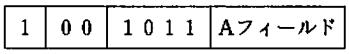
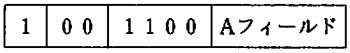
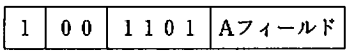
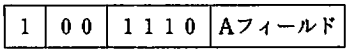
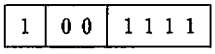
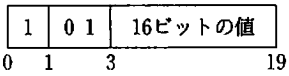
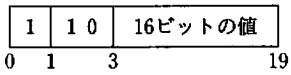
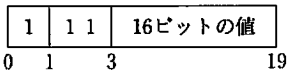
オブジェクトファイルはビットストリームより成ります．ビットストリームの中の個々のフィールドは特に述べるものを除いてバイト境界に調整されません．リロケータブルなオブジェクトファイルにビットストリームを用いることによりオブジェクトファイルのサイズが小さくなり，ディスクに対する読み出し，書き込みの回数を減らすことができます．

ロードアイテムにはアブソリュートとリロケータブルアイテムとの2つの基本的な型があります．ロードアイテムの最初のビットが0であればアブソリュートアイテムであり，また，ビットが1であればリロケータブルアイテムです．さらに，リロケータブルアイテムでは次の2ビットにより4つの型に分けられます．

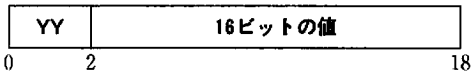
MSX･L-80ではオブジェクトファイル中のロードアイテムの種類によって次の表のような編集を行ってリンクロードします．

表D-1 ロードアイテム一覧

ロードアイテム	説 明
アブソリュートなロードアイテム 0 | アブソリュートバイト 0 1 9	2ビット目以降の8ビットはアブソリュートバイトとしてロードされる．
特別のLINKアイテム 1 | 00 | 制御フィールド 0 1 3 7	特別のLINKアイテムはビットストリーム100から始まり，4ビットの制御フィールドによって次のような種類があります．
1 | 00 | 0000 | Bフィールド	エントリ記号(検索の対象となる名前)．
1 | 00 | 0001 | Bフィールド	COMMONブロックの選択．
1 | 00 | 0010 | Bフィールド	プログラム名．
1 | 00 | 0011 | Bフィールド	ライブラリ検索の要求．
1 | 00 | 0100 | Cフィールド	拡張LINKアイテム．

ロードアイテム	説　明
	COMMON の大きさの定義
	外部参照記号のチェイン(A フィールドはチェインの先頭,B フィールドは外部参照記号).
	エントリポイントの指定(A フィールドはアドレス，B フィールドは名前).
	外部参照記号-オフセット．外部参照記号に対するジャンプ命令のために使用される.
	外部参照記号＋オフセット．A フィールドの値は実行の直前に現ロケーションカウンタが示すアドレスから始まる 2 バイトに加えられる.
	データ領域の大きさを定義. (A フィールドは大きさ).
	ローディング時のロケーションカウンタ(A フィールドはロケーションカウンタ).
	アドレスのチェイン(A フィールドはチェインの先頭番地であり,チェインで結ばれているすべてのエントリはロケーションカウンタの現在の値で置き換えられる.チェインの最後のエントリはアブソリュート番地 0 をアドレスフィールドとして持っている.)
	プログラムの大きさの定義. (A フィールドは大きさ)
	プログラムの最後. (バイト境界にされる)
	ファイルの終わり.
コード相対 	当該プログラムのベースアドレスを加えてから後に続く 16 ビットをロードする.
データ相対 	当該プログラムのベースアドレスを加えてから後に続く 16 ビットをロードする.
コモン相対 	当該プログラムのベースアドレスを加えてから後に続く 16 ビットをロードする.

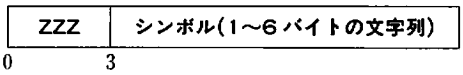
Aフィールド：



YY： 2ビットの番地型フィールド

00　絶対アドレス
01　コード相対
10　データ相対
11　コモン相対

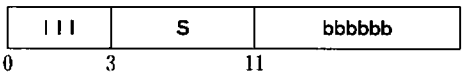
Bフィールド：



ZZZ　=シンボルの長さ(3ビット)

シンボル= 1～6文字の文字列

Cフィールド：



lll　=追加情報の長さ(3ビット)

F80はブランクコモンについて0のエントリ長を作成しますので0は1を意味します

S　=次のような8ビットのサブタイプ識別子

5 (X'35')　COBOLオーバーレイセグメント標識
A (X'41')　算術フィクスアップ(算術演算子)
B (X'42')　算術フィクスアップ(外部参照)
C (X'43')　算術フィクスアップ(領域ベース＋オフセット)

bbbbbb= 1～6バイトの追加情報

名前に対するLINKアイテムの場合追加情報は6文字だけです.

拡張LINKアイテムの説明

オーバレイセグメント標識の場合追加情報の長さは2バイト(lll = 010)であり,当該オーバレイセグメント番号が値b＋49にセットされます. このセグメントは直前のセグメント番号が0以外である場合, かつ,/Nスイッチが有効である場合にファイル名が現プログラム名であり, 拡張子が.Vnnであるディスク上のファイルに書き込まれます. (nnは数字(b＋49)10進を表わす2文字の16進数です.

サブタイプA, B, およびCによってポーランド式算術テキストの処理ができます.
各アイテムは, 逆ポーランド式として, 読み取らなければなりません. 1つ以上のサブタイプBまたはCのアイテムの次に1つ以上の算術演算子(サブタイプA)が続き, Store-Result算術演算子

(B.STBT または B.STWT)で終了します．すべての項目はオフセットがすべて最終(絶対)アドレスに変換されたあとフィックスアップ・テーブルに置かれます．ポーランド式はリンクの終わりにフィックスアップ・テーブルから取り出して実行され，その結果は LINK アイテムを読み込んだときの PC にストアされます．

索 引

お問い合わせについて

弊社では厳重に梱包した上，細心の注意を払って製品を発送しております．万一，輸送上のトラブルが起こった場合にはご一報いただければ新しいものと交換いたします．

マニュアル作成にあたり，なるべく詳細な説明をするように心がけたつもりですが，理解できないところは実際にコンピュータと向きあって納得のいくまで確かめて下さい．また，他のページを参照するのもひとつの方法です．それでも疑問点が解決できないときは，封書にて，下記の要領でお問い合わせ下さい．このパッケージの性格上，電話でのお答えは不可能と存じますので，ご了承下さいますようお願い申し上げます．記入されてない項目が一つでもありますと，回答できかねる場合があります．十分注意して下さい．

また，本製品以外に対してのご意見，ご希望がございましたら，弊社までおよせ下さい．

記

1.送付先　〒107-24　東京都港区南青山6-11-1　スリーエフ南青山ビル
株式会社アスキー システム事業部 MSX係

2.必要項目

(1)お客様の氏名，住所(郵便番号)，電話番号(市外局番も含む)

(2)製品名，製品シリアル番号，ユーザーID番号

(3)機器構成

本体装置名，メモリバイト数

CRT 装置名，フロッピーディスク装置名

プリンタ装置名

その他 I/O，I/F 装置名

(4)お問い合わせ内容

お問い合わせの内容は，できるだけ製品のマニュアルに記述されている用語を用いて，具体的かつ明確に記述して下さい．なお，障害と思われる現象については，その現象を再現可能な情報が必要です．当社で再現できないものは，調査ができません．その現象が発生するまでの操作手順，データを必ず添付して下さい．データディスクがある場合は，そのコピーも同封していただくと調査がスピーディーになります．

また，お客様固有と思われるアプリケーションの設計，作成，運用，保守については，当社のサポート範囲外ですので，お問い合わせいただいても回答できません．ご了承下さいますようお願いいたします．

MSX-DOS2 TOOLS USER'S MANUAL

1993年4月1日　第3版第3刷発行

編　集　　株式会社アスキー システム事業部

担当　北浦　訓行，舩橋　卓也

発行所　　株式会社アスキー

〒107-24　東京都港区南青山6-11-1　スリーエフ南青山ビル

制　作　　株式会社ジャパックス インターナショナル

ASCII