ZBKシステムプログラム操作説明書

(有)中日電工

目次

〒463-0067 名古屋市守山区守山2－8－14
パレス守山305
有限会社中日電工
TEL052-791-6254 Fax052-791-1391
E-mail thisida@alles.or.jp
Homepage http://www.alles.or.jp/~thisida/

[memo]

1章　準備／システムプログラムの起動

1．準備

1）CDROM取扱説明書にしたがってZBKNフォルダを作成して、必要なプログラムをコピーしておいてください。
2）ND80KL／86とWindowsパソコンをUSB接続します。
3）デスクトップのZBKNアイコンをクリックしてMSDOSプロンプト（DOSプロンプト、DOS窓）を開きます。
4）ND80KL／86の電源をONにして7セグメントLEDの表示がオール0になっていることを確認してください。

以上で準備は完了です。

2．ZBKシステムプログラムの起動

ZBKN[Enter]と入力するとシステムプログラムが起動します。

C:￥ZBKN＞ZBKN[Enter]

DEBUG　TOOL　FOR　KL5C8012
(C)Copyright　CHUNICHIDENKO
ZBKボードをサーチ中です．．
＜1＞

と表示されます。

[注記]
先にND80KL／86ボードの電源をONにしておかないと、
ND80KL／86が接続されていないかボードの電源が入っていません。
のメッセージが表示されます。

ここでND80KL／86の5×5キーボードから[I／O][8]とキー入力します。
7セグメントLEDに12345678と表示されたあと、USB通信が開始され、接続に成功すると7セグメントLEDには42434445と表示され、WindowsのDOSプロンプト画面には

＜1＞[41][42][43][44][45]
ZBKボードとの接続に成功しました
）

と表示されます。
これ以後コマンドの入力が可能になります。
ZBKシステムのコマンドやBASIC命令は英大文字（半角）ですが、パソコンのキーボードから入力された文字はZBKシステムプログラムが英小文字から大文字に変換してバッファに格納しますから、コマンドもBASIC命令も小文字で入力ができます。

)Z[Enter]

と入力します。
プロンプトマークが＞になってBASICモードになります。
ZBKシステムの起動前のDOSプロンプトの表示と同じですが、両者には次の違いがあります。

C:￥ZBKN＞　　…ZBKシステム起動前
＞　　…ZBK－V3BASIC起動後

これでZBK－V3BASICシステムが起動してBASICが使える状態になりました。
このあとBASICを使うためには、ZBK－V3BASIC操作説明書を参照してください。
BASICプログラムについて一通り理解できたら、マシン語モニタコマンドやアセンブラも使ってみてください。
参考までにZBK－V3BASICを使う上での基本的な事柄の概要を下に記します。下記概要とその次の「3．システムの終了」「4．ログファイル」を読んでから、ZBK－V3BASIC操作説明書に移ってください。

[ZBK－V3BASIC基本操作の概要]
行番号つきのプログラムを入力作成することができます。

スクリーンエディタ機能が働いています。
マウスは使えません。
カーソルの移動は[←][↓][↑][→]と[Backspace]、[Delete]および[Insert]を使います。[Enter]を押すことでその行が入力確定されます。
[PageUp][PageDown][Home][End]キーによってPageUp、PageDownができます。どんどん入力して画面の上から消えてしまった行も[PageUp][PageDown][Home][End]キーによって再表示できます。
AUTOコマンドで行番号を自動表示できます。
Z80アセンブラはここでは使用しません。
ZBKシステムプログラムを起動しないで、MSDOSプロンプト(コマンドプロンプト)上でZASM. COM、ZDAS. COMを使用します(なおラインアセンブラ、ライン逆アセンブラはZBK－V3BASIC上で使用します)。

3. システムの終了

／EXITコマンドを使います。
システムがハングUPしたときは、[Ctrl]CでMSDOSに戻ります。
WindowsパソコンとND80KL／86間での通信が食い違ってしまいハングUPした場合には[Ctrl]Cが受けつけられないことがあります。
そのときはDOSプロンプト(コマンドプロンプト)画面の右上の×ボタンをクリックして強制終了してください。[！]メッセージが出ますが[はい]をクリックすれば終了できます。
終了後にND80KL／86をリセットするか電源を切ってください。

4. ログファイル

ZBKシステムプログラムが起動すると、その時の月日時分をファイル名とするログファイルがOPENします。
システム起動中に入力、表示された内容はすべてログファイルに記録されます。
／EXITコマンドで終了したときは完全なログが保存されます。
それ以外の方法で終了した場合には最後の1ページは保存されないことがあります。
ログファイルはZBKN. EXEが置かれたフォルダ上のZBKLOGフォルダ(存在しなければ自動的に作成される)に12031453. TXTのように名前がつけられて作成されます(システムが起動した日時を名前にします。
この例では12月3日14時53分)。
ログファイルはテキストエディタで開いて参照、またはプリンタ出力したり、さかのぼって表示を確認するなどしてデバッグの助けとすることができます。
ログファイルはサイズが大きくなるので適宜削除してください。

4．1 ／CLOSE

ZBKシステムプログラムが起動中に作成されつつあるログファイルはそのZBKシステムプログラムを終了するまではテキストエディタなどで参照することができません。
／CLOSEコマンドが用意されており、任意の時点で／CLOSE[Enter]と入力すると一旦ログファイルがクローズされ、その日時をファイルネームとするログファイルが新たに作られます。
クローズされたログファイルはZBKシステムプログラムを終了しなくてもテキストエディタなどで参照することができます。

2章　マシン語プログラムの作成

ZBKシステムのメリットは8ビットCPUの制御をBASICプログラムで行うことができる点にあります。
しかし処理によってはマシン語サブルーチンと併用したいときも出て来ます。
ZBKシステムはマシン語プログラムの開発にも適しています。

1．Z80アセンブラ

マシン語プログラムの作成にはアセンブラが便利です。
KL5C80A12CPUボードにはWindowsパソコン上で使えるZ80アセンブラ（ZASM．COM）とZ80逆アセンブラ（ZDAS．COM）が含まれています。
マシン語サブルーチンの作成にはZASM．COMを使えば簡単にZ80マシン語プログラムを作成することができます。
ZASM．COMはZBK－V3BASICを起動して使うのではなく、MSDOSプロンプト（コマンドプロンプト）上で単独に使います。
ZBK－V3BASICを起動させた状態で別のMSDOSプロンプト（コマンドプロンプト）を開いてそこでZASM．COMを実行し、マシン語プログラムを作成してからZBK－V3BASICのMSDOSプロンプト（コマンドプロンプト）に戻って、いま作成したマシン語プログラムのバイナリファイルをロードする、といった使い方もできます。
Z80アセンブラの使い方については 「ND80Z3．5アセンブラ・逆アセンブラ操作説明書」を参照してください。

2．ラインアセンブラ

小さなマシン語プログラムで今ちょっと一回だけ使うだけで、特に保存しておく必要はない、という場合に便利なのがラインアセンブラです。
ZBK－V3BASICにエントリした状態で適当なユーザー用アドレス（テキストエリアのアドレス）を指定して、ASコマンドを入力するだけですぐに使うことができます。

```
＞AS　8000［Enter］
8000　_　……ここでZ80ニモニックを入力すればただちにマシン語コードに翻訳される
```

手軽に使えるアセンブラとして重宝します。
詳細は5章を参照してください。

3．マシン語モニタコマンド

アセンブラ、ラインアセンブラが使えるシステムですからマシン語コードをそのまま入力するCMコマンドはプログラム作成の手段としては、すでに使命を終えた感があります。
しかしメモリの中身を確認したり、少し値を変更したりするには、CMコマンドは非常に有効な手段になります。
数バイトではなくて数十バイトのメモリの内容を確認するにはDMコマンドが便利です。
アセンブラを利用したプログラムをデバッグするにはマシン語モニタコマンドの助けが必要です。
BP／RT（ブレークポイント／リターンコマンド）はマシン語プログラムのデバッグには欠かせない機能です。
マシン語モニタコマンドの詳細については4章を参照してください。

4．逆アセンブラ

これも有りがたい機能です。
マシン語プログラムのソースプログラムがどこかにいってしまって見つからない、とかあるいはもともと昔に作ったプログラムでROMだけしか残っていない、という場合に威力を発揮します。
ROMならばROM　WRITERにセットしてR256コマンドでRAMに読みこみます。
CMコマンドやDMコマンドを使えば書いてある内容をマシン語コードで確認することはできます。
しかしその一部を変更するとか、別のシステムに移植するとかになると、マシン語コマンドでそれを行うのは至難の技です。
内容を判りやすいZ80ニーモニックに直して読みたいというのでしたら、即席で使えるライン逆アセンブラが便利です。
DAコマンドを使って、解読したいメモリアドレスを指定すれば1ステップずつZ80ニーモニックに翻訳して表示します。
ライン逆アセンブラについては6章を参照してください。
マシン語のバイナリファイルを読みこんで、それを再アセンブル可能なZ80アセンブラソースプログラムにまで作り上げてしまう、本格的な逆アセンブラも利用できます。
MSDOS上で使用できるZ80逆アセンブラについては 「ND80Z3．5アセンブラ・逆アセンブラ操作説明書」を参照してください。

5. （参考）KL5C8012（Z80A）のレジスタ

5. 1 レジスタ

KL5C8012（Z80A）は内部に下記のレジスタを持っていて、これらのレジスタはプログラムの中で色々な処理に利用されます。

← 8ビット→ ← 8ビット→

A	F
B	C
D	E
H	L

← 8ビット→ ← 8ビット→

A’	F’
B’	C’
D’	E’
H’	L’

← 16ビット →

IX	
IY	
SP	
PC	
I	R

（裏レジスタ）
A～Lは全く同じレジスタがもう1組ある。
通常裏レジスタと呼んでいる。EXX’やEX AF，AF’命令で表裏を切り換えるが厳密な意味での表裏の区別は無い（裏も表に呼び出せばその時点から表レジスタになってしまう）

［A］
一般にアキュムレータ（加算器）と呼ばれているように、演算命令はこのレジスタを中心に行われる。
［F］
フラグレジスタ。命令の実行により現れる色々な状態を1ビットずつに記録して保持する。各ビットの意味は5．2で説明する。
［B］［C］
共に8ビットのレジスタとして、独立して使うこともできるが、つないで16ビットのレジスタ［BC］として使うこともできる。その場合には［B］が上位8ビット、［C］が下位8ビットになる。［B］［BC］をカウンタとして使っている命令がいくつかある。
［D］［E］
B、Cと同じだがカウンタとして使う命令はない。
［H］［L］
B、Cと同じだがカウンタとして使う命令はない。16ビットレジスタ［HL］はメモリアドレスを入れて使うことが多い。また［HL］は16ビットの加減算命令で加算器（アキュムレータ）としても使われる。
［IX］［IY］
インデックスレジスタとして使うが、16ビットのワークレジスタとして使うこともできる。［HL］と同様に16ビットの加減算命令で加算器（アキュムレータ）としても使われる。
［SP］
スタックポインタ。現在のスタックのトップ・アドレスを示している。スタックについては、5．3で説明する。
［PC］
プログラムカウンタ。現在実行中のアドレスを管理している。（ 正しくは、次のアドレスを示している）
［I］
インタラプト・ベクタ。モード2の割り込みで使用される。Iレジスタはシステムモニタでも使っているのでユーザーはIレジスタの値を変更してはならない。
［R］
リフレッシュレジスタ。ダイナミックRAMのリフレッシュアドレス出力用カウンタ。プログラムでは使わない。

5. 2 KL5C8012（Z80A）のフラグ

フラグは8ビットのフラグレジスタに、下のように割りつけられています。

S	Z		H		P／V	N	C
7	6	5	4	3	2	1	0

（ビット位置）

各記号の意味は下の通りです。（ビット3とビット5は使用されない）
なお、フラグがセットされたときは、そのビットが1になり、リセットされたときは0になります。

C
キャリ・フラグ。計算結果がオーバーフローしたときなどにセットされる。ローテイト命令でもセット、リセットされる。
N
加減算フラグ。加算命令のときリセット、減算命令のときセットされる。
P／V
パリティ・オーバーフロー・フラグ。論理演算のときはパリティ・フラグとして用いられ、演算の結果1のビットが偶数個あるときにセッ

トされ、奇数個のときはリセットされる。算術演算のときはオーバーフロー・フラグとして用いられ結果がオーバーフローしたときセットされる。

H

ハーフ・キャリ・フラグ。算術演算でビット3からビット4へのキャリーや、ビット4からビット3へのボローがあったときセットされる。これはNフラグとともに、BCD演算後のDAA命令で利用される。

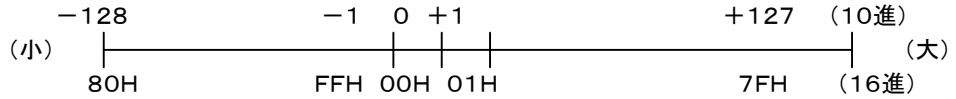
Z

ゼロ・フラグ。結果がゼロのときセットされる。

S

サイン・フラグ。結果が負のときセット、正またはゼロのときリセットされる。8ビットの数00H～FFHは符号なしでは10進の0～255として扱われるが、符号付の数として扱ったときには、－128～＋127の数になり、これは16進では80H～7FHになる。(ビット7が0のときはその数は正で、ビット7が1のときは負になる)

●符号付8ビットの数の大小



5．3 スタック

大きなプログラムになると、レジスタもたくさん必要で、とても5．1で説明したレジスタだけでは足りません。そこでレジスタの値をひとまずメモリのワークエリアにしまっておいて、そのレジスタを次の用途に使う、ということが簡単にできると便利になります。

ところがレジスタの値をしまうときに、一々異なるメモリアドレスに割りつけていくのでは大変です。

そんなときにスタックを使えば、一々メモリアドレスを指定しなくても簡単な操作でレジスタの値を保存することができます。

スタックとは積み重ねるという意味です。

ちょうど本などを積み重ねるように、メモリの中にレジスタの値を順番にしまうことができます(PUSH命令を使います)。

取り出すときは、入れたときと逆の順番で取り出します(POP命令を使います)。

スタックの現在の位置を管理しているのがSP(スタックポインタ)です。

(例) SP＝F800H、BC＝1234H、DE＝5678Hのとき、

```
PUSH  BC
PUSH  DE
```

を実行すると、メモリ内容は下のようになります。

F800H		←命令実行前のSPの位置
F7FFH	12	
F7FEH	34	
F7FDH	56	
F7FCH	78	←命令実行後のSPの位置(SP＝F7FCH。BC、DEは変化しない)
F7FBH		

この後で POP HL を実行すると、下のようになります。

F700H		
F7FFH	12	
F7FEH	34	←命令実行後のSPの位置(SP＝F7FEH。HL＝5678Hになる。BC、DEは
F7FDH	56	変化しない)
F7FCH	78	
F7FBH		

こうなってしまった後で、POP　DEを実行してもDEにはもとの値は戻りません(1234Hが入る)。

PUSH、POPは常に順番を覚えておいて、間違わないように使う必要があります。

〔注意1〕

スタックの操作はPUSH、POPだけではなくCALL、RET命令や割り込み処理でも使用されます(アドレスがスタックに入れられる)。

3章 プログラムの保存

ZBK－V3BASICにエントリして作成したマシン語のプログラム、データやBASICのプログラムはZBK－V3BASICを終了すれば消えてしまいます。
実際にはボタン電池でRAMのバックアップをしているので消えてしまうことはありません。
しかしその上から別のプログラムで上書きすれば無くなってしまいます。
マシン語プログラム、データやBASICプログラムはファイル名をつけてハードディスクに保存することができます。
作成されるファイルはMSDOSの規則にしたがって作られますから、MSDOSやWindowsのツールで参照したり編集することが可能です。
マシン語プログラム、データはバイナリ形式のファイルになります。
テキストエディタで見ることはできませんが、MSDOSのDEBUGコマンドで編集することができます。
BASICプログラムはテキスト形式で保存されますから、Notepadなどのテキストエディタで参照、修正、プリンタへの出力などが行えます。
各コマンドはZBK－V3BASICにエントリした状態で使用します。

1. マシン語プログラム（およびデータ）の保存

／SVコマンドを使います。

[書式]／SV ファイルネーム, aaaa, bbbb

指定アドレス範囲（aaaa～bbbb）のマシン語プログラム（またはデータ）をファイル名をつけてディスクに保存します。
aaaa, bbbbは4桁の16進数でaaaa＜＝bbbbです。
ファイル名はMSDOSの規則に従います。
／SVコマンドで作成したバイナリファイルを逆アセンブラ（ZDAS. COM）にかけてアセンブラソースプログラムを作成することもできます。

[使用例]
＞／SV TEST__SUB. BIN, 4004, 4365[Enter]

[注記]
ファイル名は名前部が8桁以内の英数字および－__で拡張子は3桁以内です。
拡張子は任意です。
付けなくても構いません。
ファイル名にはドライブ名やディレクトリ名も付加することができます。
例: ／SV C:￥BASIC￥TESTPRO. ROM, 4000, 7FFF

2. マシン語プログラム（およびデータ）のファイルからの読出し

1. で作成したバイナリファイルをRAMの指定アドレスに読みこみます。
／LDコマンドを使います。

[書式]／LD ファイルネーム, aaaa

ファイル名にはドライブ名やディレクトリ名も付加することができます。
aaaaは4桁の16進数でLOAD先のRAMアドレスを示します。
aaaaを省略することはできません。
ファイルがみつからないときは FILE NOT FOUND と表示されます。

[使用例1]
＞／LD TEST__SUB. BIN, 4004[Enter]

3. BASICプログラムの保存

ZBK－V3BASICにエントリして作成、編集したBASICプログラムは／SAVEコマンドでディスクに保存することができます。
／SAVEコマンドについてはZBK－V3BASIC操作説明書でBASICコマンドとして説明していますのでそちらも参照してください。

[書式]／SAVE ファイルネーム

ファイルネームがファイル名のみの場合にはZBK－V3BASICが存在するディレクトリにSAVEされます。
同じファイル名があると上書きされます。
ファイル名にドライブ名やディレクトリ名を含めて記述することで、別のドライブやディレクトリにSAVEすることができます（使用例③）。
現在のテキストエリアのアドレス情報は保存されません。
／LOAD時に新しく決定されます。
／SAVEコマンドではBASICシステムによって1行ずつBASIC内部コードをテキスト形式に変換していきます。
このときプログラムは1行ずつ画面にリスト表示されます。

[使用例①]
＞／SAVE　TEST．TXT[Enter]
[使用例②]
＞／SAVE　MIHON[Enter]　　……拡張子はなくてもよい。
[使用例③]
＞／SAVE　C：￥BASIC￥TESTPRO．BAS[Enter]
[注記1]
使用例③のように別のドライブや別のディレクトリにSAVEすることもできます（上の①②例ではZBK－V3BASICの存在するディレクトリにSAVEされます）。

4.　BASICプログラムのファイルからの読出し

テキスト形式で保存されたファイルをBASICプログラムとしてND80KL／86のRAMエリアに読みこむことができます。
／LOADコマンドを使います。

[書式]／LOAD　ファイルネーム，aaaa

ファイルネームがファイル名のみの場合にはZBK－V3BASICが存在するディレクトリからLOADします。
ファイル名にドライブ名やディレクトリ名を含めて記述することで、別のドライブやディレクトリにあるファイルをLOADすることができます（使用例③）。
ファイルがみつからないときは　FILE　NOT　FOUND　と表示されます。
aaaaは4桁の16進数で、省略することもできます。
省略した場合には現在のテキストエリアの先頭からLOADされます。
aaaaをつけるとaaaa番地からLOADされます。
／SAVEでファイルを作成したときのテキストエリアのアドレス情報は保存されません。
／LOAD時に新しく決定されます。
／LOADコマンドでテキスト形式のファイルが読み込まれるとき、BASICシステムによって1行ずつBASIC内部コードに変換してRAMに格納していきます。
このときプログラムは1行ずつ画面にリスト表示されます。
内部コードに変換するときに文法エラーが見つかるとそこでエラーコードが表示されLOAD作業は打ち切られます。
メモ帳などのテキストエディタで作成したプログラムも文法的に正しければ／SAVEで保存されたファイルと同じようにLOADすることができます。

[注意1]
LOADコマンドの動作は、NEW　＋　／LOAD、またはNEW　aaaa　＋　／LOAD、というように必ずNEWコマンドの動作を伴っています（LOAD前にRAMに存在したプログラムは失われます）。
[注意2]
Windowsのメモ帳（Notepad）はWindowsのバージョンによってはピュアなテキストファイルが作成されずにファイルエンドにゴミが混じる場合があります。
フリーソフトのTeraPadがお勧めです。
全角英数モードで作成されたファイルは読みこめません（異常な表示になります）。
必ず半角英数モードで作成してください。

[使用例①]
＞／LOAD　TEST．TXT[Enter]
[使用例②]
＞／LOAD　MIHON[Enter]　　……テキスト形式のファイルなら拡張子のついていないファイルでもよい
[使用例③]
＞／LOAD　C：￥BASIC￥TESTPRO．BAS，5000[Enter]

この例のように別のドライブや別のディレクトリにあるファイルをLOADすることもできます。
上の①②例ではプログラムは現在のテキストエリアの先頭からLOADされます。
③では5000番地からLOADされます。

[注意3]
ファイル名は名前部分が半角英数8文字以内で拡張子は3文字以内に限られます。
[注意4]
テキスト形式（拡張子の種類に関わらず内容がテキスト形式になっている）以外のファイルをLOADするとZBK－V3BASICシステムが暴走してハングアップすることがあります。
テキスト形式のファイルでも半角英数字以外の文字が使用してあると、やはり暴走してしまいます。

5. マシン語サブルーチンとBASICプログラムの一括保存

マシン語プログラムはアセンブラで作成して、ソースプログラムは□□□. TXT、マシン語コードは□□□. BINなどというファイル名で保存します。
BASICプログラムは△△△. TXTのようにそれらとはまた別に保存します。
テキストエディタで編集を行うためにはこの形でなければいけませんが、プログラムとして完成した時点で保存を考えると、BASICプログラムだけではなくてマシン語サブルーチンと一緒に働くプログラムの場合には、別々に保存するのは面倒でもあります。
この場合、BASICブログラムとマシン語プログラムを一括してバイナリ形式で保存できると便利です。
マシン語プログラムだけの場合と同じように／SVコマントを使いますが、そのままではあとでBASICプログラムをLIST表示可能な状態に戻すことはできません。／SVコマンドに先だってBROMコマンドでBASICテキストの情報を作成します。

[書式]BROM
　　／SV　ファイルネーム，4000，bbbb

[使用例]
＞BROM[Enter]
4500－5732
＞／SV　TESTPRO. ROM，4000，5732[Enter]

[注記]
ファイル名にはドライブ名やディレクトリ名も付加することができます。
例：　／SV　C:￥BASIC￥TESTPRO. ROM，4000，7FFF
[注意1]
／SVの先頭アドレスは必ず4000にします。
先にBROMを実行しておなかいと、／LD作業でBASICプログラムを再生させることができません。
bbbbはBROMの実行により表示される2番目のアドレス値にします。
使用例で／SVのアドレス指定は4500，5732ではなくて4000，5732になる点に注意してください。
[注意2]
このようにして保存したファイルにはBASICプログラムも含まれていますがバイナリイメージで保存しているため、Notepadなどのテキストエディタで参照することはできません。

6. マシン語サブルーチンとBASICプログラムを一括保存したファイルからの読みこみ

この場合もマシン語プログラムだけの場合と同様に／LDコマントを使いますが、LOAD後にBASICプログラムをLIST表示可能にするため、BSSETコマンドの実行が必要です。
ファイルにBASICプログラムが含まれていない場合や／LDで4000以外のアドレスを指定した場合、または／SVでBROMを実行しなかったか、SAVE開始アドレスを4000にしなかった場合には、BSSETコマンド入力時にHOW？が表示されます。

[書式]／LD　ファイルネーム，4000
　　BSSET

[使用例]
＞／LD　A:￥BASIC￥TESTPRO. ROM，4000[Enter]
2543BYTES　LOAD
＞BSSET[Enter]
TEXT　4500－6543
ﾍﾝｽｳ DBC9－DFFF

4章　マシン語モニタコマンドおよびシステム操作コマンド

メモリの内容をチェックしたり、部分的にマシン語のデータを変更したり、簡単なマシン語プログラムを書いて、すぐに実行してみたい、というようなことがよくあります。
BASICの命令をダイレクトモードで実行してもそのようなことはできるのですが、ZBK－V3BASICに含まれている、マシン語モニタコマンドを利用すると、BASICとはまた異なった細かい操作が、簡単に行えます。
またマシン語モニタコマンドのほかに、ZBK．EXEにはファイル操作や終了処理などのシステムコマンドも用意されています。
これらのコマンドはND80KL／86単独で使用することはできません。
Windowsパソコンと接続しているときのみ、パソコンのキーボードから入力して実行することができます。

●コマンドとパラメータの区切りについて

ZBK－V3BASICのもとになった以前の旧システムではコマンドとパラメータの区切りとして ,(カンマ)を使っていました。
しかしBASICでは命令の後ろの区切り(セパレータ)としてスペースを使っており、またアセンブラニーモニックも命令とオペランドとの区切りはスペースを使っている点を考えて現在のZBK－V3BASICではコマンドの後ろの区切りマークとしてスペースを使うように改めました。
そのような経緯があって区切りマークとしてカンマも使えますが、この説明書の書式としては、スペースを使っています。

／BOOT ［省略形］無し

ブートプログラムに戻ります。通常の使い方では必要ありません。
ブートプログラムについては8章を参照してください。

BP／RT／CR ［省略形］B.（RT、CRの省略形は無し）

[書式1]BP aaaa
[書式2]BP 0
[書式3]BP D

ZBKシステムではマシン語プログラムの開発、デバッグに役立つようRAM上に書かれたプログラムなら、どこでも実行中にブレイクできるようブレイクポイントが1カ所設定可能です(BPはbreak pointの、またRTはreturnの略です)。
aaaaは4桁の16進数で、ブレイクしたいアドレスを指定します。
例えば、下のようなプログラムを実行する場合を考えます。

```
C000  2100C1          LD  HL,＄C100
C003  75        LOOP:LD  (HL),L
C004  2C               INC  L
C005  C203C0           JP  NZ,LOOP
C008  C33310           JP  ＄1033
```

このプログラムは＄C100～＄C1FFにデータ＄00～＄FFを書きこむものです。
CMコマンドでC000～C00Aまでに上のマシン語プログラムを1バイトずつ入力します。
これを実行するにはJP C000でよいのですが、この動作を確認したいと仮定します。
下のように入力してみます。

```
＞BP C003[Enter]
＞JP C000[Enter]
```

すると下のような表示が得られます。

```
A F  B C  D E  H L  A'F' B'C' D'E' H'L'  PC   SP   IX   IY  I  SZ H PNC
xx44 xxxx xxxx C100 xxxx xxxx xxxx xxxx C003 F800 xxxx xxxx xx 01000100
```

BPコマンドでアドレスを指定したうえで、マシン語のプログラムを実行させると、そのアドレスまで実行が進んだところで、そのアドレスの命令を実行する直前でブレイクしてその時の全レジスタの内容を表示してコマンド待ちになります。
このプログラム例ではフラグレジスタ(F)、HLレジスタ、プログラムカウンタ(PC)、スタックポインタ(SP)以外は使用していないため、その他のレジスタの内容には意味はありません(誤解を避けるため、xxxx にしてあります)。
一番最後の8桁はフラグレジスタ(F)の内容をビット毎に表示したものです。
この状態は普通のコマンドモードと変わりないので、他のコマンドを受け付けることができます。例えばDMコマンドでメモリの内容を確認したりすることもできます。

必要な確認が全て済んで、以後の実行を継続したい時は

＞RT[Enter]

と入力します。以後は普通に処理が終了します。
なおBPは続けてセットすることもできます。
上の状態でブレイクしているときに、下のように入力してみて下さい。

＞BP C005[Enter]
＞RT[Enter]

再び、ブレイクしますが今度は下のようにF、HL、PCが変化していることが確認できるはずです。

```
A F  B C  D E  H L  A'F' B'C' D'E' H'L'  PC   SP   IX   IY  I  SZ H PNC
xx00 xxxx xxxx C101 xxxx xxxx xxxx xxxx C005 F800 xxxx xxxx xx 00000000
```

＄C005ではフラグの内容がチェックされ、ZフラグがOFFならば再び＄C003に戻って処理が続けられます。フラグの内容を見てみると、右のフラグのビット表示は全て0になっていて、どのフラグもOFFになっていることがわかります。
そこで再びBP C003[Enter]とセットしたうえで、RT[Enter]を入力すると、今度はC003でブレイクします。
ブレイクポイントを設定するときは、必ず命令の1バイト目のアドレスを指定する必要があります。
上の例で、C000、C003、C004、C005、C008は指定できますが、C001、C002、C006、C007、C009、C00Aを指定すると正しく実行されません。
正しく指定しているにもかかわらず、指定アドレスによってはブレイクがかからないときがあります。
上のプログラム例では全部の命令が少なくとも一回は実行されますが、プログラムによっては、条件付の分岐命令があって、条件によっては全く実行されない部分がでてきます。
そのような場合にはブレイクポイントを設定したにもかかわらず、ブレイクしないで処理が終了してしまいます。
このような場合には次のことに注意して下さい。
ブレイク動作は指定アドレスの命令を、FF（RST 7）で置きかえることで実現しています。CPUはFFコードを見つけると＄0038からの命令を実行しますが、このシステムでは、その＄0038にブレイク処理ルーチンが置いてあります。
ブレイクすると、ブレイク処理が行われると共に、ブレイクアドレスに、FFのかわりにもとの命令コードをもどします。ところがブレイクしないとこのFFが残ってしまいます。
例えば上の例で、BP C003[Enter]を入力したあと、すぐにCMコマンドかDMコマンドでC000～C00Aの内容を確認してみて下さい。C003にFFが入っているのが確認できます。
ブレイクしないために残ってしまったFFコードをもとの命令コードに戻すには、

＞BP 0[Enter] ……（書式2）

と入力します（もう一度C000～C00Aの内容を確認してみて下さい。C003にFFが入っていたのが元の75に戻っているはずです）。
ブレイクポイントがセットされたままになっているかどうかはっきりしない時は、下のようにBP D[Enter]と入力すればわかります。セットされているときはそのアドレスが表示されます。

＞BP D[Enter] ……（書式3）
C003 ……BPアドレスが表示される。BPがセットされていない時は何も表示されない

[注意]
リセットすると、BPがセットされたままになっていても、FFコードがプログラム内に残されたままで、BP関係の制御情報はクリアされてしまいます。そうなった後で、BP 0[Enter]を実行してもFFはもとに戻りませんし、BP D[Enter]を実行しても何も表示されません。

[**CR**コマンド]
プレイクしたアドレスから続きを実行するときにレジスタの値を変更すると都合がよい場合があります。ループしているプログラムの動作をデバッグしていて、ループカウンタの途中のある値の近くを詳しく調べたいときなどがあります。1000回のループの終りの20回くらいをチェックしたいときに980回もBP／RTを繰り返すのでは疲れてしまいます。このときループカウンタとしてメモリのワークエリアを使っていれば、ループで一旦ブレイクしループカウンタの値をCMコマンドで変更しておいて（1000を20にしてしまいます。同時にループ回数に伴って変化するワークエリアがあれば、それも合理的な値に直します。）、ループ内の次のプレイクポイントを設定してRTコマンドを実行すれば、980回を一度で済ますことができます。
CRコマンドはA、B、CなどのCPUレジスタもメモリのワークエリアと同じように変更できる機能です。
ブレイクしているときに、CR[Enter]と入力します。
ブレイク直後と同じレジスタ表示が行われます。

```
>CR[Enter]
A F  B C  D E  H L  A'F' B'C' D'E' H'L'  PC   SP   IX   IY  I  SZ H PNC
3785 0004 0000 C6BE 502C 0000 0000 0000 25C5 F7EA 204B 0182 FE 10000101

_
```

↑カーソルが表示されます。[↑][→]で変更したい値を書き換えます。スクリーンエディタが働いていますからレジスタ全部でも1レジスタでもフラグのみでも変更できます。変更する必要のないレジスタはそのままにしておきます。例としてA＝50、HL＝1234、キャリーフラグOFFに書き換えてみます。

```
A F  B C  D E  H L  A'F' B'C' D'E' H'L'  PC   SP   IX   IY  I  SZ H PNC
5085 0004 0000 1234 502C 0000 0000 0000 25C5 F7EA 204B 0182 FE 10000100_
```

必要な変更が済んだら、[Enter]を入力します。カーソルはこの行の中ならどこにあっても構いません。[Enter]入力によってレジスタが更新されます。そして更新結果がすぐ下に表示されます。

```
A F  B C  D E  H L  A'F' B'C' D'E' H'L'  PC   SP   IX   IY  I  SZ H PNC
5084 0004 0000 1234 502C 0000 0000 0000 25C5 F7EA 204B 0182 FE 10000100
>
```

[注意1]

フラグレジスタの変更は右のビット表示の部分で行います。左のAFの部分を書き換えても更新はできません。ビット部分を書きかえることで、結果のAFレジスタ部分が更新されます（上例でFレジスタが85から84に変化していることを確認してください）。

[注意2]

カーソルを他の行に移動してはいけません。通常の＞表示によるコマンド入力待ちのスクリーンエディタとは違うルーチンで入力しますから、他の行で[Enter]を入力するとエラーになります。

[注意3]

CRはブレイク中にレジスタの値を変更し、その後にRTで中断中のプログラムを再開することを前提にしています。その他のタイミングで使用しても無意味です（システムには影響を与えません）。

[注意4]

CRは繰り返して実行できますが、値の更新はCRコマンド入力直後のレジスタ表示の時のみ有効です。普通のブレイク表示の時やCRでの値の更新後の[Enter]入力により表示されたレジスタ表示行に対して更新作業はできません。もう一度更新したいときはあらためて、CRコマンドを入力します。

●以上のコマンドを下にまとめて整理しておきます。

①BP aaaa	ブレイクしたいアドレスをセットする。命令コードの1バイト目のアドレスを指定する。先の例でBP C003はよいが、BP C002やBP C001は不可。また、ROM上のプログラムにはセットできない
②BP 0	ブレイクポイントを解除する。
③BP D	ブレイクポイントがセットされているときはそのアドレスを表示する
④RT	ブレイクしていたアドレスから以後を続けて実行する。
⑤CR	ブレイク後にレジスタの値を更新してRTによる実行再開に反映させる

[注記1]

すでに説明したように、このブレイクの機能は、RST 7（FF）を利用しているため、ブレイクポイントがセットされているときに、マシン語プログラムのミスなどで暴走した結果、別のアドレスでたまたまFFコードに行きつくと、ブレイクポイント以外の部分でもブレイクしてしまうことがあります。

[注記2]

ユーザープログラム内でSPに新しい値を設定しない場合には、システムスタック（＄F490～＄F7FF）がそのまま使用されることになります。

BPの設定により、実行途中でブレイクするといままでユーザーが使用していたスタックをシステムが使用することになります。そのままではスタック内容が変化してしまうため、RTコマンドでユーザープログラムの実行を再開した場合に、正しく実行されなくなります。

そこでこのシステムでは、通常使われるスタックを768バイトと仮定し、＄F500～＄F7FFの内容を、ブレイク時に別のエリア（＄E500～＄E7FF）に一時待避します。そしてRTコマンドが入力されると、この待避データをもとの＄F500～に戻した上で、ユーザープログラムにリターンします。

ブレイク時のスタックの状態を確認したい場合に、もしシステムスタックをそのまま利用している場合には、この＄E500～＄E7FFを見るようにします（DM E500, E7FF[Enter]を入力する）。

BROM ［省略形］BR. BRO.

BASICプログラムの開始アドレスと終了アドレスを＄4000～＄4003の制御エリアに書込みます。

／SVコマンドでBASICプログラムとマシン語プログラムをバイナリファイルとして一括保存することができますが、その場合には、／SVにさきだってBROMの実行が必要です。

BSSET ［省略形］BS.　BSS.　BSSE.

　バイナリイメージでロードしたBASICプログラムをLIST表示可能な形にします。
／SVコマンドでBASICプログラムとマシン語プログラムを一括保存したバイナリファイルもマシン語のみのバイナリファイルと同様に／LDコマンドでRAMに読み込むことができますが、そのままではBASICプログラムもバイナリのままなのでLIST表示させることはできません。BSSETコマンドによってLIST表示が可能になり、編集や追加、削除などができるようになります。

［使用例］
＞／LD　TESTPRO. ROM, 4000［Enter］
＞BSSET［Enter］
TEXT　4500－6543
ﾍﾝｽｳ DBC9－DFFF

／CLOSE［省略形］なし

ZBKシステムプログラムが起動中に作成されつつあるログファイルはZBKシステムプログラムを終了するまではテキストエディタなどで参照することができません。
／CLOSEコマンドが用意されており、任意の時点で／CLOSE［Enter］と入力すると一旦ログファイルがクローズされ、その日時をファイルネームとするログファイルが新たに作られます。クローズされたログファイルはZBKシステムプログラムを終了しなくてもテキストエディタなどで参照することができます。

CM　［省略形］C.

［書式］CM　aaaa

指定アドレス（aaaa）のメモリの内容を1バイトずつ16進数で表示し、キーボードから入力する16進数と置き換えます。（aaaaは4桁の16進数）
マシン語で簡単なプログラムを書きたい時や、数バイト～数十バイト程度のメモリ内容を書き換えたい時などに使うと便利です。（CMはchange　memoryの略です）

［使用例］
①内容を確認する
＞CM　8100［Enter］
8100　2A－［Space］　指定したアドレスとそのメモリの内容が表示される。このままの内容でよい場合には
8101　87－　　スペースキーを押すと、次のアドレスが同じように続けて下に表示される

②内容を更新する
8100　2A－31　下線部のように入力する。［Enter］は必要無い。0～F以外のキーを押すと？が出てもう一
8101　87－　　度同じアドレスが表示される。データ（2桁の16進数）を入力すると、次のアドレスが同じように続けて表示される

③前のアドレスに戻る
8100　2A－31　　入力ミスなどでもう一度前のアドレスに戻りたいときは［←］を押す。前のアドレスとその
8101　87－［←］　　内容が下に表示される
8100　31－

④処理の終了
この作業を打ち切りたい時は［Enter］を押すか又は［Ctrl］B を入力します。するとプロンプトマーク（＞）が出てコマンド入力待ちに戻り、カーソルマークが表示されます。
この時までに行った書き換え作業はすでに実行済みなので、処理を打ち切っても取り消しはできません。

［注意］
スペースキー、［←］、［Enter］、［Ctrl］B　は1桁目のデータ入力では有効ですが、2桁目の入力では受け付けられません。？マークが表示されます。

8105　3A－_　　ここでのスペースキー、［←］、［Enter］、［Ctrl］B［の入力は有効
8105　3A－5_　　ここでスペースキー、［←］、［Enter］、［Ctrl］Bを入力すると？が出る

CP　［省略形］なし

[書式]CP　aaaa,bbbb,cccc

　指定アドレス範囲（aaaa～bbbb）のメモリの内容を、指定アドレス（cccc）からはじまるメモリの内容と1バイトずつ比較し、一致しない場合はそのアドレスとデータを表示します。一致した箇所は表示されません。（CPはcompareの略です）
　比較作業が完了すると、END＝bbbbと表示して、コマンドモードに戻ります。
　aaaa、bbbb，ccccは4桁の16進数で、ccccに制限はありませんが、aaaa≦bbbbでなければいけません。aaaa＞bbbbのときはHOW？メッセージが出ます。

CS　[省略形]なし

　指定範囲のメモリの内容を1バイトずつ加算して、そのトータル値を16進4桁で表示します（CS＝Check　Sum）。

[書式]CS　aaaa，bbbb

　aaaaは開始アドレス、bbbbは終了アドレスです。

DM　[省略形]D.

[書式]DM　aaaa,bbbb

　CMが1バイトずつの表示であったのに対し、DMコマンドは指定した範囲（aaaa～bbbb）のメモリ内容を1行16バイトずつ表示します。（DMはdump　memory　&　restoreの略です）
　aaaa、bbbbは4桁の16進数でaaaa≦bbbbでなければいけません。aaaa＞bbbbのときはHOW？メッセージが出ます。
　1行16バイトで表示する結果、最後の行が16バイトに満たなくなった場合でも、指定範囲を越えて16バイト分の表示が行われます。
　メモリ内容の表示はCMと同じように16進数2桁で行い、さらに各行の右側にその16進数を文字コード（JIS、ASCII）とみなして、対応する文字を一緒に表示します。（＄00～＄1F、＄80～＄9F、＄E0～＄FFのコードに対しては、ピリオド（　.　）が表示されます）

／EXIT　[省略形]なし

　ZBKシステムを終了します。ログファイルもクローズされます。
　ZBKシステムの終了は／EXITのほか[Ctrl]＋[C]入力でも行えますが、[Ctrl]＋[C]で終了するとログファイルが正しく保存されないことがあります。

JP　[省略形]J.

[書式]JP　aaaa

　指定する16進アドレス（aaaa）にジャンプします。（JPはjumpの略です）
　ユーザーが書いたマシン語のプログラムを実行する場合に使います。そのプログラムの先頭アドレスを指定することによって、これ以後はCPUの制御はユーザープログラムに移ります。

●ユーザープログラムの終りの処理
　ユーザープログラムの最後の命令は重要です。BASICプログラムなら終りっぱなしでも構いませんが、マシン語プログラムを終りっぱなしにすると大抵は暴走して止まらなくなったりハングUPしてしまいます。
　ユーザープログラムの最後にはシステムのリエントリアドレスへのジャンプ命令を書いてください。リエントリアドレスは＄1033です。
　JP　＄1033（コード　C33310）と書いてください。
　ユーザープログラムの最後にJP　＄1033があるとユーザープログラム終了後そのままシステムのリエントリプログラムが実行されます。そこではスタックの値やその他システムの処理を継続するのに必要なワークアドレスの値が再設定されるため、ユーザーは終りっぱなしに近い感覚でプログラムを終ることができます。

[参考]USR命令
　マシン語のユーザープログラムを実行するには、このJPコマンドの他にBASICのUSR（＄aaaa）命令も使えます。
　USR（＄aaaa）もコマンドモードで実行すれば、動作はこのJPコマンドと同じになります。
　終わりの処理は、JPの場合には既に説明したように、＄1033へのジャンプ命令でなければいけませんが、このUSR命令の場合には、RET命令で終わることもできます（USR命令をBASICプログラム中で使う場合には、終わりは必ずRET命令でなければいけ

ません)。
ただし終わりがRET命令の場合には、プログラム中でSPの値を不用意に変更してはいけません。PUSH、POPの使い方にも注意して、ユーザープログラムのスタート時のSPの値と、最後のRET命令の実行直前のSPの値が同じになるように注意する必要があります。

／LD [省略形]なし

[書式]／LD ファイルネーム, aaaa

バイナリファイルをRAMの指定アドレスにロードします。ファイル名にはドライブ名やディレクトリ名も付加することができます。aaaaは4桁の16進数でLOAD先のRAMアドレスを示します。aaaaを省略することはできません。ファイルがみつからないときは FILE NOT FOUND と表示されます。
／SVでBASICプログラムとマシン語サブルーチンを一括して保存作成したファイルの場合にはロード後にBSSETコマンドでBASICプログラムをLIST表示可能な形にすることができます(使用例2)。

[使用例1]
＞／LD TEST__SUB. BIN, 4004[Enter]

[使用例2]
＞／LD C:￥BASIC￥TESTPRO. ROM, 4000[Enter]
＞BSSET[Enter]
TEXT 4500－6543
ﾍﾝｽｳ DBC9－DFFF

MV [省略形]M.

[書式]MV aaaa,bbbb,cccc

指定アドレス範囲(aaaa～bbbb)のメモリの内容を、指定アドレス(cccc)からはじまるメモリエリアにコピーします。(MVはmoveの略です)
もとのメモリエリア(aaaa～bbbb)の内容は変化しません。
ただしもとのメモリエリアと転送先のメモリエリアに重なりがある場合には、重なった部分のメモリ内容は転送後の内容に置き代わります(メモリエリアの重なり方に制限はありません。どういう重なり方でも、正しく転送されます)。
aaaa、bbbb, ccccは4桁の16進数で、ccccに制限はありませんが、aaaa≦bbbbでなければいけません。aaaa＞bbbbのときはHOW？メッセージが出ます。

[注意]
このコマンドは、指定範囲のメモリ内容がプログラムであっても、ただのデータとしてそのまま転送します。
したがって例えばC000から実行される形で書かれたマシン語のプログラムをこのコマンドで他のアドレス(例えばB000)に移動させた場合、そのアドレスではそのプログラムは実行できないことに注意して下さい。
そのプログラムを他のアドレスで実行できるように移動したい場合には、一度逆アセンブラ(ZDAS. COM)にかけて、ソースプログラムを再生した後、アセンブラ(ZASM. COM)で、そのアドレスにオブジェクトプログラムを作るようにします(詳しくは 「ND80Z3. 5アセンブラ・逆アセンブラ操作説明書」を参照してください。

R256 [省略形]なし

別売の27C256 WRITERを実装して、書き込み用ソケットに書込み済みのROMをセットして、このコマンドを実行するとROMからRAMのテキストエリア(＄4000～)にROMの内容がそのままCOPYされます。
＞R256[Enter] 27C256の内容が4000～BFFFに読み込まれる。ROMの前半16KBが8000～BFFFに、後半16KBが4000～7FFFに入る。

[注意]
書き込み用ソケットは通電状態でROMをセットするため、逆差しをするとその瞬間にROMが破損することになります。十分注意して作業してください。

SD [省略形]なし

[書式1]SD aaaa, bbbb, hhhhhh……
[書式2]SD aaaa, bbbb, ”文字列”

指定アドレス範囲（aaaa～bbbb）のメモリをサーチし、指定データと同じ内容をみつけると、その前後の32バイトを表示します（SD＝Search Data）。
［書式1］は指定データを16進数で示したもので、データの組合せに制限はありません。
［書式2］は指定データを文字列で示します。したがってサーチできるデータは文字コードに限られます。

［使用例］
>SD 5000,57FF,C33310[Enter]

```
51C3  11 55 1F C3 CC 11 11 5B 1F CD 61 11 AF CD 18 10  .U.ﾃﾌ..[.ﾍ a.ｯﾍ..
51D3  C3 33 10 CD E0 11 7C FE FF CA BD 11 C9 21 00 00  ﾃ3.ﾍ..:..ﾊｽ.ﾉ!..

5289  00 DB 01 E6 50 C9 CD 86 12 C0 CD A2 15 CD 03 1A  .ﾛ..Pﾉﾍ .ﾀﾍ｢.ﾍ..
5299  C3 33 10 CD 6F 11 0E 06 1A 6F 13 1A 67 13 CD 5D  ﾃ3.ﾍo....o..g.ﾍ]
```

>SD 4000,4FFF,"ERR" [Enter]

```
4641  55 54 D2 45 41 44 D2 45 53 54 4F 52 45 CF 4E 20  UTﾒEADﾒESTOREﾏN
4651  45 52 52 4F 52 20 47 4F 54 4F D2 45 53 55 4D 45  ERROR GOTOﾒESUME
```

／SV　［省略形］C.

［書式］／SV　ファイルネーム，aaaa，bbbb

指定アドレス範囲（aaaa～bbbb）のマシン語プログラム（データ）をファイル名をつけてハードディスクに保存します。aaaa，bbbbは4桁の16進数でaaaa＜＝bbbbです。ファイル名はMSDOSの規則に従います。
BASICプログラムとマシン語サブルーチンを一括して保存する目的で使うと効果的です（使用例2）。その場合には／SVコマンドにさきだってBROMコマンドの実行が必要です。［使用例2］ではユーザープログラムのROMイメージを保存することと同じです。この場合も拡張子は任意です。

［使用例1］
＞／SV　TEST_SUB. BIN，4004，4365［Enter］

［使用例2］
＞BROM［Enter］
4500－5732
＞／SV　TESTPRO. ROM，4000，5732［Enter］

［注記］
ファイル名は名前部が8桁以内の英数字および－＿で拡張子は3桁以内です。拡張子は任意です。付けなくても構いません。ファイル名にはドライブ名やディレクトリ名も付加することができます。
例：　／SV　C:￥BASIC￥TESTPRO. ROM，4000，7FFF
［注意1］
使用例2ではaaaaは必ず4000にします。先にBROMを実行しておなかいと、／LD作業でBASICプログラムを再生させることができません。bbbbはBROMの実行により表示される2番目のアドレス値にします。使用例でaaaa，bbbbは4500，5732ではなくて4000，5732になる点に注意してください
［注意2］
使用例2で保存したファイルにはBASICプログラムも含まれていますがバイナリイメージで保存しているため、Notepadなどのテキストエディタで参照することはできません。

W256　［省略形］なし

［書式］W256　aaaa,bbbb

27C256用の書き込みコマンドです。
aaaa、bbbbは4桁の16進数でそれぞれもとになるメモリの、書き込みを開始したいアドレスと、書き込みを終了したいアドレスを示します。
＄4000～＄7FFFが書き込み用ROMの後半16kBに対応し、＄8000～＄BFFFが前半16kBに対応します。しかしコマンドパラメータとしては必ずaaaa＜＝bbbbです。

W256 4000, BFFF （正）
W256 8000, 7FFF （誤）

W256 4000, BFFF[Enter]と入力すると、＊が左から右に表示されていきます。＊1個が256バイト書込み済みを示します。27C256は32KBですから4000～BFFFまで書く場合には＊が128個表示されることになります。
書き込み中にエラーが発生すると書き込みを打ち切り、そのときのアドレスとその後ろにもとのデータとそれに対するエラーデータを表示したあと、その下にERR, Wと表示してコマンドモードに戻ります。

[注記]
部分的に書き込むことも可能です。
ROMの先頭から書き込まなくても、途中からでも書き込みを開始できます。 たとえばW256 4000,435Fとか、W256 5300, 56EFというような指定をしても構いません。
途中のアドレスを指定した場合は、ROMの先頭から書き込まれるのではなく、対応するアドレス位置から書き込みが開始されます。

XD [省略形]なし

[書式1]XD aaaa, bbbb, nnnn……, mmmm……

指定アドレス範囲（aaaa～bbbb）のメモリをサーチし、指定データ（nnnn……）と同じ内容をみつけるとその前後の32バイトを表示したのち、mmmm……で置きかえます（XD＝Exchange Data）。
nnnn……, mmmm……は16進2桁を単位として複数バイト記述できます。nnnn……, mmmm……は同じ長さでなければいけません。

[使用例]
```
>XD 5000,57FF,C33310,C33010[Enter]

51C3  11 55 1F C3 CC 11 11 5B 1F CD 61 11 AF CD 18 10  .U.ﾃﾌ..[.ﾍ a.ｯﾍ..
51D3  C3 33 10 CD E0 11 7C FE FF CA BD 11 C9 21 00 00  ﾃ3.ﾍ..ﾆ..ﾊｽ.ﾉ!..

5289  00 DB 01 E6 50 C9 CD 86 12 C0 CD A2 15 CD 03 1A  .ﾛ..Pﾉﾍ .ﾀﾍ｢.ﾍ..
5299  C3 33 10 CD 6F 11 0E 06 1A 6F 13 1A 67 13 CD 5D  ﾃ3.ﾍo....o..g.ﾍ]

>SD 5000,57FF, C33010[Enter]

51C3  11 55 1F C3 CC 11 11 5B 1F CD 61 11 AF CD 18 10  .U.ﾃﾌ..[.ﾍ a.ｯﾍ..
51D3  C3 30 10 CD E0 11 7C FE FF CA BD 11 C9 21 00 00  ﾃ3.ﾍ..ﾆ..ﾊｽ.ﾉ!..

5289  00 DB 01 E6 50 C9 CD 86 12 C0 CD A2 15 CD 03 1A  .ﾛ..Pﾉﾍ .ﾀﾍ｢.ﾍ..
5299  C3 30 10 CD 6F 11 0E 06 1A 6F 13 1A 67 13 CD 5D  ﾃ3.ﾍo....o..g.ﾍ]
```

5章　ラインアセンブラ

マシン語の短いプログラムを書くのにはCMコマンドが便利です。
しかしそれにしてもマシン語の命令をマシン語コードで書くのは根気の要る作業です。
ジャンプ命令がC3でOUTがD3、とこのくらいはすぐに覚えられますが、LD命令になると種類が多くてとても覚えられません。一つ一つの命令毎に命令表を引きながらマシン語コードに置き換えるのは、面倒で大変根気の要る作業です。
この章で説明するラインアセンブラの機能は、その退屈な置き換え作業を一気にゼロにしてしまいます。
ニーモニックコードをキーから入力すると、ただちにマシン語コードに置き換えて、メモリに書き込んでくれます。

1.　ラインアセンブラの起動

下のようにキー入力します。

＞AS　aaaa[Enter]

aaaaはマシン語プログラムの先頭のメモリアドレスで4桁の16進数です。

[使用例]
>AS 8000[Enter]　　……このように入力すると
8000 __　　……アドレスに続いてカーソルが表示される。そこで
8000 LD A,($8100)[Enter]　　……と入力すると
8000 3A0081　　LD A,($8100)　　……すぐにマシン語コードが表示されて次のアドレスとカーソルが表示される。
8003 __
8003 LD B,A[Enter]　　……続けて次の命令を入力すると
8003 47　　LD B,A　　……マシン語コードが表示されて次のアドレスとカーソルが表示される。
8004 __

処理を終了（中止）したいときは[Ctrl]Bを入力します。
この機能はニーモニックコードをマシン語コードに翻訳し画面に表示すると同時にそのメモリアドレスにマシン語コードを書き込む作業も行っています。
したがってどの時点で処理を中止しても、それ以前に翻訳された部分はマシン語コードでメモリに書き込まれています。
本当に書き込まれているかどうか、CMコマンドかDMコマンドで確認してみて下さい。
なおニーモニックコードを入力している途中で、キーの押し間違いに気がついたら、[Enter]を押す前ならば、［←］［→］[INST][DEL] などの機能を使って、書き直すことができます。
[Enter]を入れてしまったあとでは、取り消しはできません。
済んでしまったアドレスに戻ってやりなおしたい場合には、[Ctrl]Bを入力して作業を打ち切ってから、もう一度希望するアドレスを指定してASコマンドを入力します。
またORG命令を書くことでも、アドレスを指定しなおすことができます。

8050 ORG $8035[Enter]　　……このように入力すると
8050　　ORG $8035
8035 __　　……次のアドレスが変更されて表示される。

2.　ラベル・変数の使用

ライン・アセンブラは、疑似命令の機能なども一定の条件付で使用することができます。
ラベル・変数も使用することは可能ですが、ライン・アセンブラの性質上、その値がそれ以前に確定している場合に限ります。
たとえば次のようなプログラムを考えます。

```
        LD B,A
LOOP1:INC C
        JP LOOP1
```

ここで2行目のINC命令の前に書いてあるLOOP1というのがラベルです。その下のJP命令のジャンプ先を示しています。この例ではJP命令でそのジャンプ先としてLOOP1というラベルを使用した時点では、そのアドレスはすでに確定しています。
というのは、LOOP1:INC C という行を入力した時点で、システムによってその時のアドレスがLOOP1として記憶されるからです。
したがってそれからあとで、JP LOOP1 という行を入力したとき、システムによって、このLOOP1として記憶されていたアドレスが読み出されて使用されます。
しかし、もしジャンプ先がそのJP命令よりもさらに先にあった場合に、そのジャンプ先がラベルで表現されていたとすると、このJP命

令を翻訳する時点では、まだそのジャンプ先のラベルはどのアドレスかわかりませんから、エラーになります。
したがってこの場合には、直接16進数で入力するしか方法はありません。
なおこれはあくまでライン・アセンブラに限っての制約で、Z80アセンブラ(ZASM. COM)ではラベルがJP命令の後にあっても全く支障ありません。
同様に、このアセンブラでは、LD HL,DATA1 というように16進4桁の数値の代わりに変数を用いることができますが、この場合にもライン・アセンブラに限り、それ以前に変数の値が確定している必要があります。
変数の値を確定するには、=(イコール)を使います(「ND80Z3. 5アセンブラ・逆アセンブラ操作説明書」を参照してください)。

[注意1]
ラベルも変数も、使用する以前に確定していなければなりませんが、それは一続きのASコマンドの作業内に限ります。
例えば一度[Ctrl]Bで処理を打ち切ると、それ以前に定義されていたラベルや変数の値はクリアされてしまうため、再びASコマンドで作業を再開しても、前のASコマンドの作業中に定義したラベルや変数を利用することはできません。
[注意2]
ラインアセンブラの作業エリアとしてB000～のメモリを使用します。B000より大きいアドレスを指定してASコマンドを実行すると、ニーモニック入力時に、SORRY と表示して処理が中止されます。
[注意3]
BASICプログラムの書かれているメモリアドレスにマシン語のプログラムを書くと、BASICプログラムは正しく実行されなくなりますから、アドレスには注意が必要です。

6章　ライン逆アセンブラ

5章では、ラインアセンブラについて説明しました。
マシン語のプログラムを作成するのに、アセンブラの機能がとても便利だということが充分理解できたと思います。
ところで時にはプログラムを作るのではなくて、「読む」ことも必要になる場合があります。
例えば、以前に作成したプログラムだが、ROMは残っているが、ノートがどこかへ行ってしまって、どういうプログラムだったかもう一度調べなければならない、というような場合です。
また時には他人の作ったプログラム（マシン語コードのリストかROMしか無い）を解析しなければならない、という場合もあると思います。
すでに説明したCMコマンドやDMコマンドは、ROMまたはRAMの中味を読むのに便利な機能です。
しかしマシン語コードで表示されるため、そのコードを命令説明書をもとに逆にニーモニックに翻訳していかなければ、何が書いてあるか理解することができません。
そんな時ライン逆アセンブラを使えば、一命令毎にニーモニックコードに逆翻訳してくれます（逆アセンブラという名前は、アセンブラの逆の働きをするところから来ています）。

1．ライン逆アセンブラの使い方

下のようにキー入力します。

＞DA　aaaa[Enter]

aaaaは読み出したいメモリの先頭アドレスで、4桁の16進数です。

[使用例]
```
>DA 8000[Enter]                  ……このように入力すると
8000 3A0081     LD A,($8100)  ……すぐに命令が翻訳されニーモニックが表示されます。ここで適当なキーを
                                  押すと
8003 47         LD B,A        ……次の命令が翻訳されて表示されます
```

何かキーを押す度に次の命令がマシン語コードと共に表示されて行きます。
実行を中止したいときは[Ctrl]B を押します。

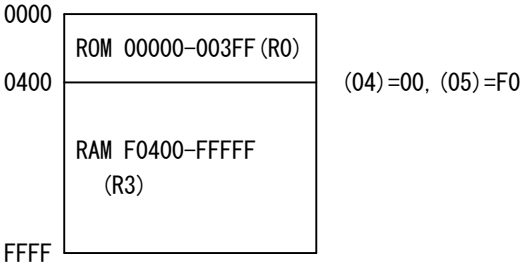
[注意]
この機能は、指定するメモリアドレスの内容がZ－80のマシン語プログラムである、という前提で翻訳作業を行います。したがって命令以外のデータなどを読んだ場合にも、その16進数に該当する命令コードがあれば、そのニーモニックに変換して表示を行います。
また2バイト以上の長さの命令の場合には、その第一バイトを指定して、逆アセンブラを開始しないと、別の命令と解釈して誤訳してしまいます。
例えば上の例で、DA　8001[Enter]と入力して、スタートしたらどうなるでしょうか？下にその場合の結果を示します。

```
>DA 8001[Enter]
8001 00         NOP
8002 81         AD A,C
8003 47         LD B,A                ……ここでやっと正しくなりました
```

逆アセンブラはオールマイティ（万能）ではない、ということをよく認識して使用して下さい。
なお、命令コードとしては存在しない16進数にぶつかったときは、ニーモニックを表示する代わりに、？が表示されます。

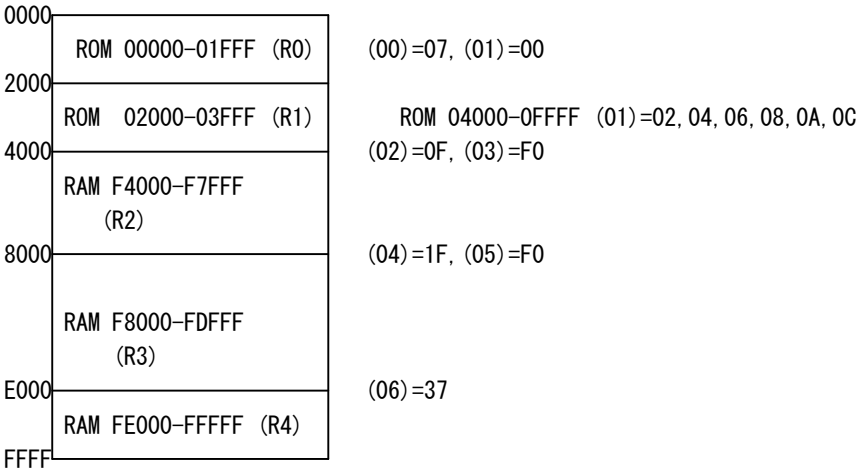
7章 メモリマップ

ZBKNで起動するとブートプログラムが実行されます。そのときのメモリマップは次の通りです。



0000～03FFがROMで残りはRAMになります。メモリバンクはR0とR3のみが有効です。メモリバンク用レジスタには（04）＝00、（05）＝F0がセットされます。メモリバンクについては、「KL5C80A12ハードウェア説明書」を参照してください。

Zコマンドを入力するか、メモリバンク用レジスタ（00）～（06）を書きかえるまでは、このメモリ構成になっています。 ZコマンドによってZBK－V3BASICが起動すると次のメモリ構成になります。



2000～3FFFはシステムがROMの02000～0FFFFをバンク切り替えによって割り当てて使用します。この部分にユーザーが関与することはできません。

8章 ブートプログラム

［1章2. システムプログラムの起動］で説明した、起動時の ） プロンプト表示がブートプログラムが実行されている状態です。ZBK－V3BASICにエントリするために入力する Z はブートプログラムのコマンドです。

ブートプログラムはZBKシステムプログラムが起動するとスタートする、小さなプログラムです（0000～03FFの1KB）。BASICを使う上では何の役にも立ちません。普通のユーザーはZコマンドでZBK－V3BASICを起動させてしまうとあとは終了するまでここに戻ってくることはありません。

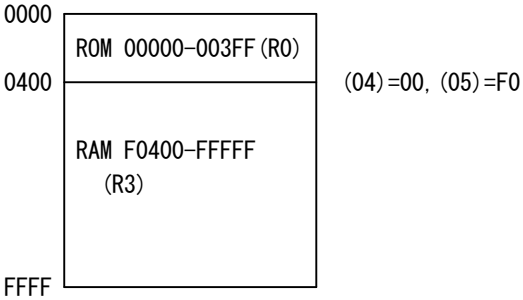
ブートプログラムは、MSDOSのDEBUGコマンドに似たマシン語デバッガです。

マシン語プログラムのためのデバッグ機能はZBK－V3BASICの中にもあります（4章）が、ブートプログラムはそれよりももっと下層で働くマシン語プログラムのためのツールです。

ブートプログラムの実行中もZBK－V3BASICと同様、スクリーンエディタが機能しています。またログファイルも作成されます。

1. 起動時（ブートプログラムエントリ時）のメモリマップ

ZBKシステムプログラムが起動すると、下のメモリマップになります。



0000～03FFがROMで残りはRAMになります。メモリバンクはR0とR3のみが有効です。メモリバンク用レジスタには（04）＝00、（05）＝F0がセットされます。メモリバンクについては、「KL5C80A12ハードウェア説明書」を参照してください。

Zコマンドを入力するか、メモリバンク用レジスタ（00）～（06）を書きかえるまでは、このメモリ構成になっています。ブートプログラムのコマンドは0400～FFFFのRAMに対して使用します。

［注意］

ブートプログラム自身のワークやスタックのためにF×××番地のメモリを使用します。この範囲のメモリを不用意に書きかえると、ブートプログラムが機能しなくなりますから、F000～FFFFは書き換えないようにしてください。

2. コマンド

次のコマンドがあります。

C （Change memory）
D （Dump memory）
G （Go）
I （In）
L （Load）
N （ND80K JUMP）
O （Out）
Q （Quit）
S （Save）
T （Trace）
Z （ZBK－V3BASIC entry）

各コマンドについて簡単に説明します。コマンド（およびパラメータ）入力後［Enter］入力により実行されます。

①C （Change memory）

［書式］C aaaa

指定メモリアドレスの内容を書き換えます。ZBK－V3BASICマシン語モニタのCMコマンドとほぼ同じ動作をします。aaaaは4桁の16進数でアドレスを示します。

②D （Dump memory）

[書式]D aaaa

指定メモリアドレスからはじまる128バイトの内容を表示します。ZBK－V3BASICマシン語モニタのDMコマンドとほぼ同じ動作をします。aaaaは4桁の16進数でアドレスを示します。

③G （Go）

[書式1]G＝aaaa bbbb
[書式2]G bbbb

指定メモリアドレスにジャンプしてユーザープログラムを実行します。[書式1]でaaaaはジャンプ先ユーザーアドレスで、bbbbはブレイクアドレスです。bbbbは省略できます。
[書式2]はブレイク後に次のブレイクアドレス（bbbb）を指定してユーザープログラムにリターンします。
aaaa、bbbbは4桁の16進数で命令の第一バイトが存在するアドレスでなければいけません。
ZBK－V3BASICマシン語モニタのJP、BP、RTコマンドとほぼ同じ動作をします。

④ I （In）

[書式] I aa

I／Oアドレス aa に対してIN命令を実行し、入力した値を16進数で表示します。ZBK－V3BASICマシン語モニタのINコマンドとほぼ同じ動作をします。aaは2桁の16進数でI／Oアドレスを示します。

⑤L （Load）

[書式] L ファイルネーム, aaaa

バイナリファイルをアドレスaaaaからロードします。ファイルネームはMSDOSのルールに従います。aaaaは4桁の16進数で省略はできません。ZBK－V3BASICマシン語モニタの／LDコマンドとほぼ同じ動作をします。ロードできるファイルサイズはB000バイト以下でなければいけません。

⑥N （ND80K JUMP）

N[Enter]でND80Kモニタに戻ることができます。
ND80KL／86ボードの7セグメントLEDの表示はオールゼロになってリセット後の状態になりますが、Windows側は改行後 ）を表示して入力待ちの状態になりますが、この時点では何も入力してはいけません（ND80KL／86ボードとの通信が途絶えています）。
ND80Kモニタからブートプログラムに戻るには、ND80KL／86ボードのキーボードから0200[ADRSSET][RUN]と操作します。通信が接続状態になるので、これ以後、再びコマンドの入力を続けることができるようになります。
ZBK－V3BASICからND80Kモニタに戻ることはできません。ZBK－V3BASICからは、まず／BOOTコマンドでブートプログラムに戻ってから、NコマンドでND80Kモニタに戻ります。
なおNコマンドでND80Kモニタに戻ったあと、ふたたび0200[ADRSSET][RUN]でブートプログラムにリターンすると、レジスタダンプが表示されますが、それ以後は普通にブートコマンドの入力ができます。

⑦O （Out）

[書式] O aa, nn

I／Oアドレス aa に2桁の16進数nnを出力します。ZBK－V3BASICマシン語モニタのOTコマンドとほぼ同じ動作をします。aaは2桁の16進数でI／Oアドレスを示します。

⑧Q （Quit）

処理を終了してMSDOSに戻ります。ブートプログラムの実行中もログファイルの機能が働いていて全ての表示内容が保存されます。Qコマンドで終了すると完全なログファイルが作成されます。[CTRL]CでMSDOSに戻った場合、最後の1ページは保存されません。

⑨**S**（Save）

［書式］ S ファイルネーム, aaaa, bbbb

バイナリファイルをアドレスaaaa～bbbbの範囲のマシン語プログラム（データ）LOADします。ファイルネームはMSDOSのルールに従います。aaaa、bbbbは4桁の16進数で省略はできません。ZBK－V3BASICマシン語モニタの／SVコマンドとほぼ同じ動作をします。

⑩**T**（Trace）

［書式1］ T＝aaaa n
［書式2］ T n

ブレイクポイントから指定ステップだけトレース実行し、各ステップ毎にレジスタの値をダンプ表示します。書式1は G＝aaaa ＋ T n の動作になります。アドレスaaaaからスタートしてnステップ、トレース実行します。
書式2はブレイクしているポイントからnステップ、トレース実行します。
トレースはKL5C8012のタイマ／カウンタBチャネル0を割り込み使用しています。
ステップ回数nは1桁以上の16進数で、nを省略するとn＝1を指定したのと同じになります。

［注意1］
トレース機能はタイマーの割り込みを利用しています。ユーザープログラムに割り込み禁止命令DIがあると実行できなくなります。そのためトレース中にDI命令をみつけると！を表示してトレースを中止します。
［注意2］
トレースはブレイク機能と異なり、ROMに書かれたプログラムに対しても実行できますが、ブートプログラム自身をトレースすると、トレースそのもので使用している機能をトレースすることになり正しく実行されません。

⑪**Z**（ZBK－V3BASIC entry）

ZBK－V3BASICにエントリします。なおZBK－V3BASICからブートプログラムに戻るには／BOOTコマンドを使います。

[memo]