NAKAMOZU TINY BASIC

連載第3回　プログラミングテクニックとグラフィックエディタ

山下春生

グラフィック用ソフトウエア

以上の改造で使用できるようになったグラフィックをグラフィック記号としてPOKEやPRINT文中で用いる場合は，英字やカナ文字のアスキー・コードと同様に割りあてられたコードでV－RAMに書き込むだけですが，64×48のフル・グラフィックとして使用する（！W，！B，！R，！Pなど）には，かなり複雑なソフトのサポートが必要です．

図1のフローは，I／Oルーチンの後半にあるグラフィック・サポート・ルーチンで，SET，RESET，REV，およびPICKは，上記のグラフィック制御命令に対応するサブルーチンです．

これらの処理のすべてで使用するBITPATというサブルーチンは，各ルーチンで共通なすべての処理を行ないます．

64×48のグラフィック座標での，X座標をACCA，Y座標をACCBにセットしてコールします．

V－RAM上のADRSは，9ビットですので，Y座標を細工して8ビット演算で済むようにします．X方向の除算は，除数が2ですので，問題ありませんがY方向は3で割る必要があります．X座標，Y座標の除算共に商と余りの両方を利用してV－RAM上のADRSとビット・パターンを計算します．同時に，現在そのADRSにあるキャラクタを読

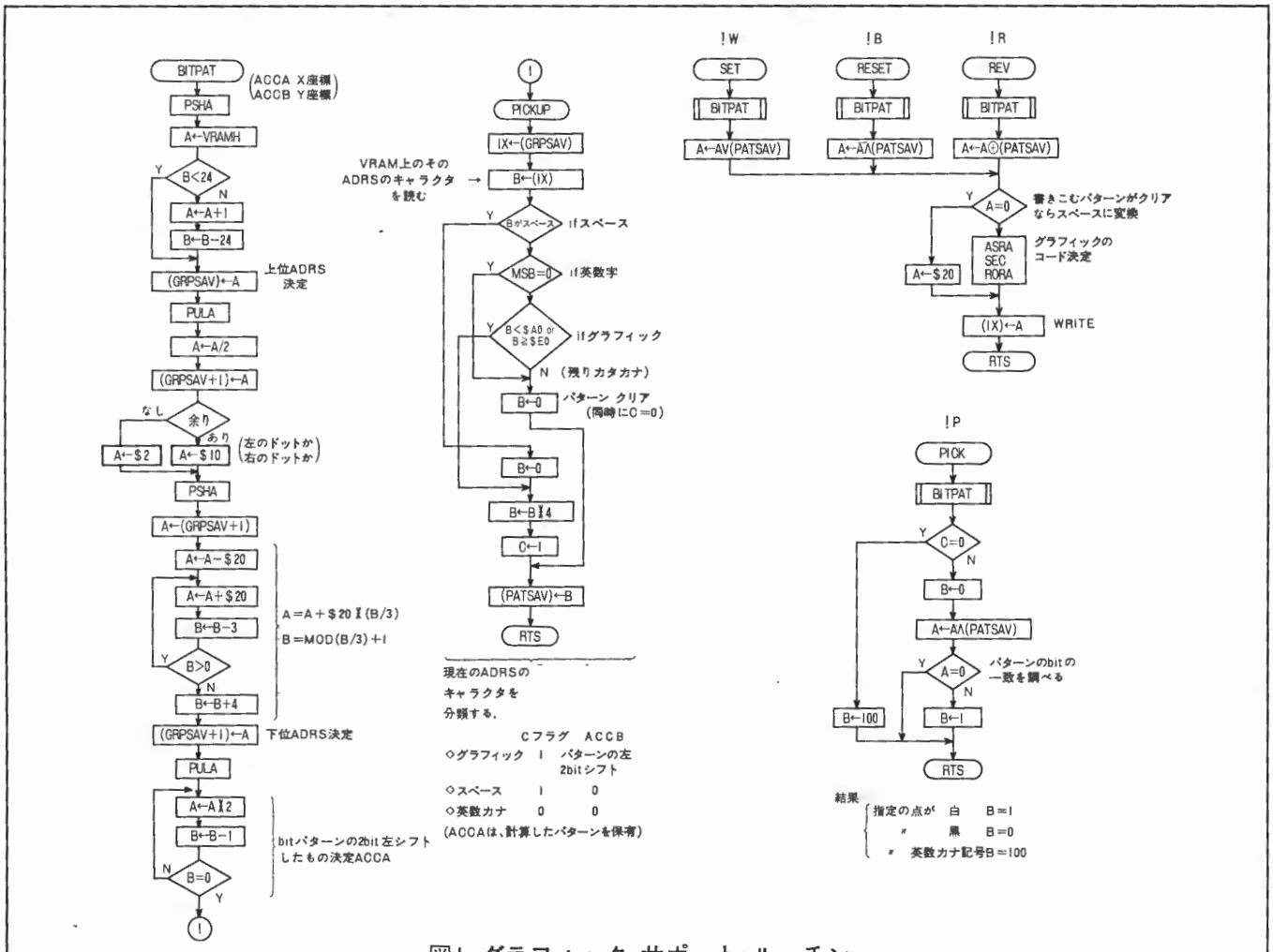
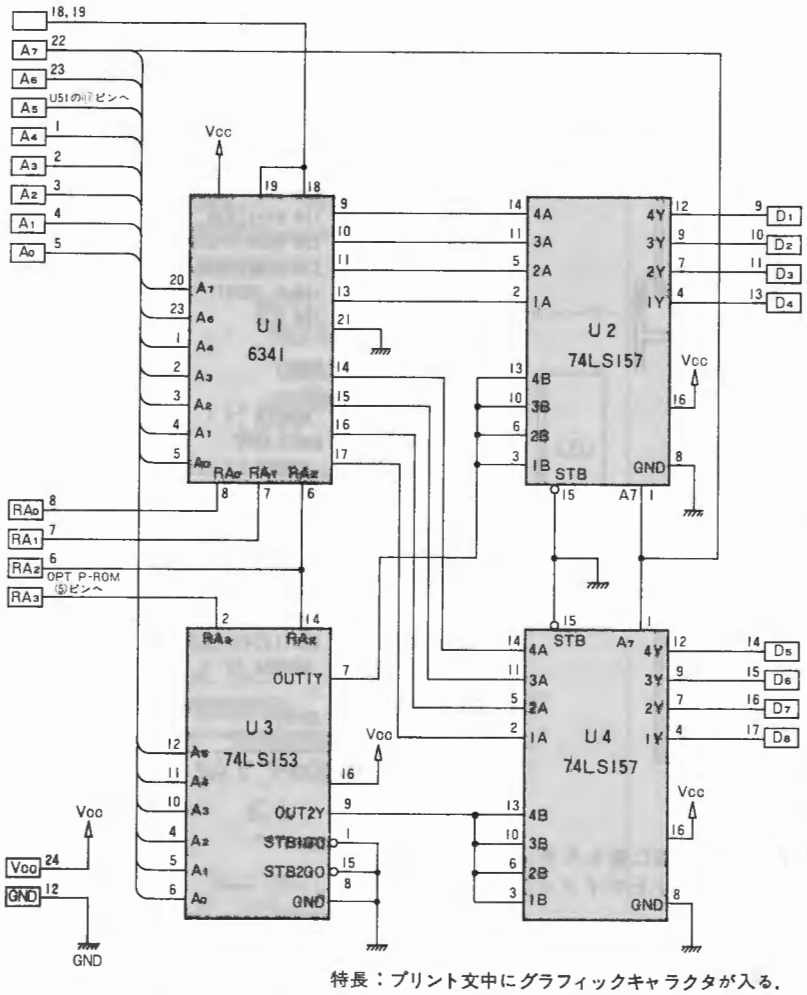


図1　グラフィック・サポート・ルーチン

H68TVのグラフィック改造(キャラクタモードでグラフィックを！)



特長：プリント文中にグラフィックキャラクタが入る．
48×64，48×128，96×64，96×128が可能．

んで分類を行なっています．注意が必要なのは，英数記号の内のスペースだけは，特別扱いでグラフィックとして分類し，グラフィック記号としてのスペース（$80）は，使用禁止にしています．（カタカナのスペースは，文字として分類する．）

以上の全サブルーチンは，かなり複雑ですが，125バイト程度にまとまっていますので，グラフィック改造をした人は，利用して下さい．

H68－TVなどを使用の人は，グラフィック制御命令の引数範囲を拡大して付属のサブルーチンとリンクし，オプション・ステートメントを利用してグラフィック・モードへの切り換えをすれば，簡単に済みますが，PEEK，POKEやPRINT文でのグラフィック使用は，不可能になります．

私の友人は，H68－TVのグラフィックモードを使用せずに，TVD02と同様の改造（ついでに，V－RAMアクセスをCPU優先に改造）して好結果を得ています．せっかくの高分解能グラフィックを使用しないのは惜しいですが，ゲームなどでキャラクタと同時にグラフィックが使用できる点は，非常に有利です．蛇足ですが，H68－TVを64×32キャラクタにイニシャライズすると，128×96になり，TVのグラフィックモードと同じ分解能になります．

NTBによるプログラミング

高級言語でプログラムする事は，アセンブラ・レベルでのプログラミングに比べて容易である事は確かですが，大きくて複雑なそして速いプログラムを書くには，それなりのトレーニングが必要です．

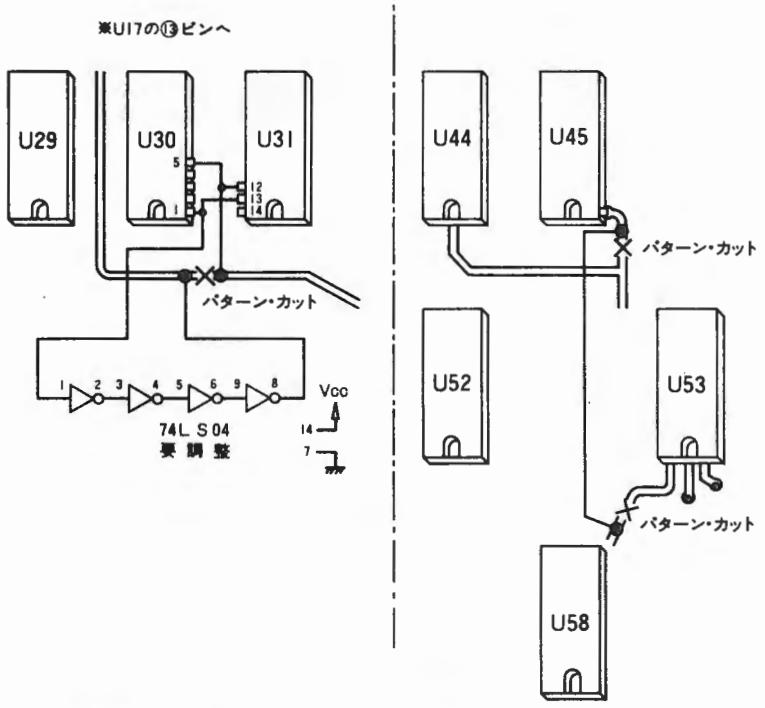
NTBの命令は，マニュアルの説明だけでは有効な使用法が明確にならないと思いますので，サンプル・プログラムを紹介しておきます．

大きなプログラムを作る時に重要なことは，一目見ただけで，プログラムの構造がわかるという点で，そうするための手法としては，

(1) REM文をブロックごとにつける
(2) ループ構造が明確にわかるDO－UNTIL，FOR－NEXTを用い，IF－

（H68ＴＶは現在新型と旧型があり，パターンが異なっています．この改造は旧型のみ可能です．）

H68TVをCPU優先に改造



改造方法　①プリント・パターンを３個所カットする
②図のように結線
③74ＬＳ04のインバーターを２，４，６個に変えてディレイ時間を調整し，リフレッシュメモリ上でイメージが出ないようにする．

特　　長　○リフレッシュメモリをＣＰＵ優先に変える．
○従来のH68/ＴＶ用ソフトはそのまま使用可能
○スクロールの時間短縮(従来の方式では，カセットロードができない)
○フラグ待ちなしで画面上を直接PEEK，POKEできる．

ＧＯＴＯによるループをなくす．

(3) (2)を実現する時，ＤＯ，ＦＯＲ，ＵＮＴＩＬ，ＮＥＸＴそしてＲＥＴなどは，マルチ・ステートメント中に置かず各行の先頭に置く．

(4) 発表する時は，他の人の移植の事を考えて変更箇所をメイン・ルーチン中の一箇所に集めておく．

などです．

プログラム１は，整数のルートを計算するサンプルであり，ゲームなどで距離の計算などで使用できます．ＲＥＭの代りに，＊を使用していますが，これはプログラム中でＲＥＡＤを使用しない時のみ使用できます．（ＲＥＡＤがない時は，ＤＡＴＡもＲＥＭも同じ．）

プログラム２は，ＩＸ配列を用いて画面操作を行なう例です．ＢＡＳＩＣのプログラムで画面を上にスクロールするもので，上に消えた行は下から現れてきます．実行速度は，一行シフト・アップするのに一秒ですみ，１キャラクタずつの動きは，目で見えず行ごとの転送に見えます．一度他のＢＡＳＩＣでもやってみて下さい．

プログラム３は，プリント文のみのサンプルでゲームなどで一番効果的に使用できるものです．こういうプリント文をキーインするのは，想像以上に大変な作業であり，私の根性では，不可能に近いと言えます．このプリント文を作った方法は，１ドット単位で絵を描いたり修正したりする数種のコマンドを持ったＧＲＡＰＨＩＣ－ＥＤＩＴＯＲで自動的にプリント文を作製しテープ出力した訳です．

プログラム４は，自然対数の底の値を小数点以下100ケタまで計算するもので，

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots$$

$$= 1 + \frac{1}{1} + \left(1 + \frac{1}{2} + \left(1 + \frac{1}{3} + \left(1 + \frac{1}{4} + \left(1 + \frac{1}{5} + (\cdots)\right)\right)\right)\right)$$

のマクローリン展開を用いています．

小数以下100ケタまでの精度を得るには，70！まで必要ですので余裕をもって71！まで72項を整数部＋小数部102ケタの固定小数小演算で行ないます．

104ケタのアキュムレータとしてＩＸ配列を使用し，１つの配列に２ケタの２進化10 進数で計算します．

実行時間は，104ケタの割り算を71回くり返すアルゴリズムの割に35秒という短時間で済み驚いています．

実際の計算は，行番号150までで，大半の時間は割り算で使用しているため，ケタ数を少なくすると非常に実効時間が短くなります．

試しに，小数第10位までにすると0.7秒で終ります．（Ｌ＝６，Ｎ＝12に変更）

●プログラム1

```
READY
>L.
10 ********************
20 *  INTEGER ROOT    *
30 * SAMPLE PROGRAM  *
40 ********************
80 DO
100 IN." NUMBER :"N
110 R=-1:W=N
120 DO:R=R+2:W=W-R
130 U.W<0:R=R/2
140 P."ROOT(";N;")=";R
150 U.0

READY
>R.
 NUMBER :? 1
ROOT(1)=1
 NUMBER :? 123
ROOT(123)=11
 NUMBER :? 8241
ROOT(8241)=90
 NUMBER :? 30120
ROOT(30120)=173
 NUMBER :? 234
ROOT(234)=15
 NUMBER :?
>
READY
```

●プログラム2

```
READY

>L.
10 ********************
20 * 2 BYTE PEEK POKE *
30 * SAMPLE PROGRAM   *
40 * ENDLESS SCROLL   *
50 ********************
60 * VRAM ADRS EQU $C000
70 V=$C000:%9=¥+1
80 %0=V:%1=V+$20:%2=V+$1E0
100 DO
110 F.L=0TO 15:%9(L)=%0(L):N.L
120 F.L=0TO239:%0(L)=%1(L):N.L
130 F.L=0TO 15:%2(L)=%9(L):N.L
140 U.0

READY
>
READY
```

●プログラム3

```
READY
L.
3500 P."
3510 P."
3520 P."
3530 P."
3540 P."
3550 P."
3560 P."
3570 P."
3580 P."
3590 P."
3600 P."
3610 P."
3620 P."
3630 P."
3650 P.
4000 P."
4010 P."
4020 P."
4030 P."
4040 P."
4050 P."
4060 P."
4070 P."
4080 P."
4090 P."
4100 P."
4110 P."
4120 P."
4130 P."
4150 P.
4500 P."
4510 P."
4520 P."
4530 P."
4540 P."
4550 P."
4560 P."
4570 P."
4580 P."
4590 P."
4600 P."
4610 P."
4620 P."
4630 P."
```

●プログラム4

```
READY
>L.
10 ***************************
20 *  EXP ﾉ ｹｲｻﾝ  100 ｹﾀ    *
30 *    BY ﾉﾘｵ ﾔﾏｼﾀ          *
40 ***************************
50 %0=¥+1:K=100:L=51:N=71
60 F.I=1TOL:%0(I)=0
70 N.I
80 %0(0)=1
100 DO
110 F.I=0/1 TOL
120 %0(I)=(%0(I)+K*MOD)/N
130 N.I
140 N=N-1:%0(0)=%0(0)+1
150 U.N=0
200 REM TYPE OUT
210 P."  EXP=";%0(0);"."
220 F.I=1TOL-1
230 P.%0(I)/10;" ";MOD;" ";
240 IFMOD(I,5)=0 P."  ";
250 IFMOD(I,10)=0 P.
260 N.I

READY
>R.
  EXP=2.
7 1 8 2 8 1 8 2 8 4   5 9 0 4 5 2 3 5 3 6
0 2 8 7 4 7 1 3 5 2   6 6 2 4 9 7 7 5 7 2
4 7 0 9 3 6 9 9 9 5   9 5 7 4 9 6 6 9 6 7
6 2 7 7 2 4 0 7 6 6   3 0 3 5 3 5 4 7 5 9
4 5 7 1 3 8 2 1 7 8   5 2 5 1 6 6 4 2 7 4

READY
```

行番号110は，割り算ルーチンで，Ｉに０を代入すると同時にＭＯＤの初期値をクリアしています．

GRAPHIC EDITOR

この種のシステム・プログラムは，アセンブラで書かれることが多いのですが，インタプリターで書くとテストをしながら完成できるため，この様なシステム・プログラムも短期間で製作できます．

このＧＲＡＰＨＩＣ－ＥＤＩＴＯＲは，グラフィック改造をしたＶ－ＲＡＭの特徴（フル・グラフィックとして使用できると同時に，グラフィック記号としてプリント文中で使用できる．）を生かして作られています．

グラフィック制御命令を用いＶ－ＲＡＭ上をドット単位（64×48）でポインタを８方向に動かしながら絵や図を描きます．

このプログラムは，Ｈ68用ＫＥＹ関数を使用しており，**図２**は，各種コマンドのＫＥＹ配置を示しています．

画面構成は，上２行をＬＯＣＫ状態の表示や，コンパイルしたプリント文をテープ出力する時のエコーバックに使用するので，14行分（Ｘ方向64ドット，Ｙ方向42ドット）がエ

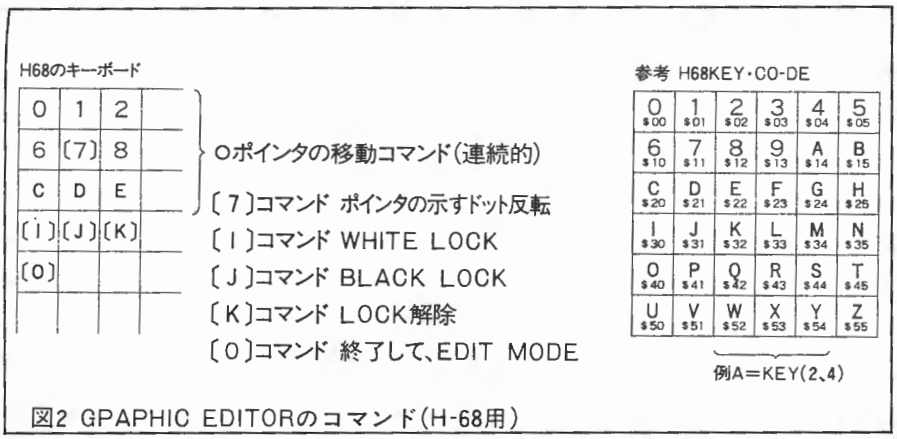


図2 GPAPHIC EDITORのコマンド(H-68用)

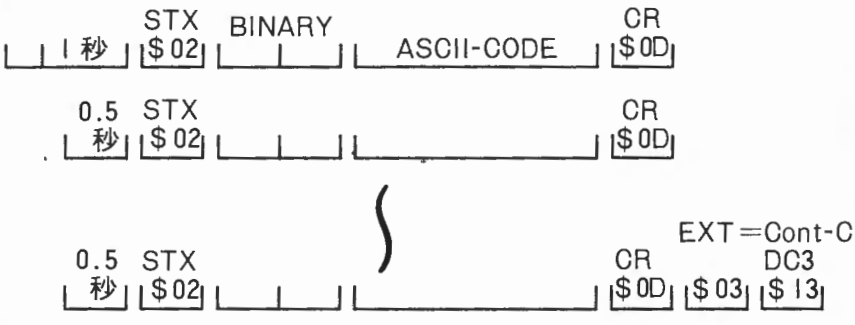


図4 N.T.B TAPE FORMAT

ディタ用バッファになりました.

RUNすると画面がクリアされ中央で1ドットが点滅します.(これがポインタ)

ポインタ移動コマンドでポインタを動かし(KEYを押す間, 連続作動する),〔7〕コマンドでそのドットを反転します.

続けて描いてゆく時は,〔I〕コマンドで, WHITE−LOCKモードにすると, ポインタの通った後の軌跡が残ります.

同様にBLACK−LOCKは, 通った後が消されてゆき, LOCKの解除は,〔K〕コマンドで行ないます.(各コマンドを投入する時発振音が出て確認できます.)

絵や図が完成し,〔O〕コマンドでEDITモードから脱出すると, 行番号とステップを聞いてきますので, カセットを録音状態で走らせキー・インしますと, その行番号から指定したステップおきのPRINT文を作ってカセットにSAVEします.

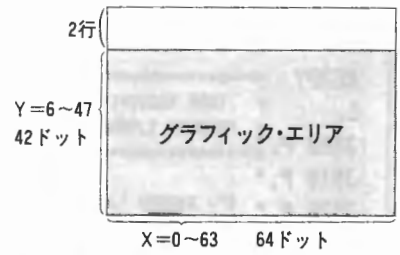


図3 画面構成

録音されるフォーマットは, 図4のようにNTBのLOAD, SAVEと全く同じですので, ゲーム等のプログラムにAPPENDして使用できます.

メイン・ルーチンは, 300行から後であり, 360行はKEY関数の値からX, Yを変更しています. 370行と380行は, Xを0〜63にYを6〜48に制限するルーチンです.

510行と550行で有効なエリアを探索し, 580行で行数をRに, 各行のグラフィック・キャラクタ数を%0(R)に入れています.

後は, 510行で出力をカセットに設定しているので, 単にPRINTするだけでSAVEでき, 図4のフォーマットを忠実に実現するだけです.

ただ, 最後の$03は, コントロールCのコードのため, プリント文では出力できないので, 770行では, フラグ待ちをしてACIAに直接出力しています.

フル・キーボードを使っている人は, KEY関数のデコードをアスキー・コードで行なうように変更するだけで使用できます.

●グラフィックエディタ プログラム・リスト

```
READY
>L.
10 ****************************
15 * N.T.B. GRAPHIC-EDITOR    *
20 *   BY H.YAMASHITA         *
25 *     H68-KEYS             *
30 *  0  1  2     :           *
35 *  6 [7] 8     : MOVE KEYS*
40 *  C  D  E     :           *
45 *  I  J  K  / I:WHITE LOCK*
50 *  O        / J:BLACK LOCK*
55 *           / K:UNLOCK     *
60 *           / O:EDIT END   *
65 *            & CASET SAVE  *
70 ****************************
80 %0=¥+1:V=$C000:S=$FF-" "
90 G.300
100 REM COMMAND
110 CU.(0,0)
120 IFM=$30 P."WHITE LOCK";C.7:F=1
130 IFM=$31 P."BLACK LOCK";C.7:F=2
140 IFM=$32 P."           ";C.6:F=0
150 REM DELAY SHORT
160 F.J=0 TO20:N.J
170 RET
200 REM TAPE
210 GOS.250:CU.(0,0):P.C.2;
220 RET
250 REM DELAY LONG
260 F.J=0 TO600:N.J
270 RET
300 REM GRAPHIC
310 CLR
315 X=31:Y=26:F=0:G=0
320 DO:GOS.160:!R(X,Y):GOS.160:!R(X,Y)
330 M=KEY(0,2)
```

```
340 IFM=$40 G=1:M=0
350 IFM)$22 GOS.100:U.0
360 Y=Y+M/16-1:X=X+MOD-1
370 X=(X)0)*X+(X)62)*(62-X)
380 Y=(Y)47)*(47-Y)+(Y)6)*(Y-6)+6
400 IFF=1 !W(X,Y)
410 IFF=2 !B(X,Y)
420 IFM=$11 !R(X,Y):P.C.7
430 U.G
450 CU.(0,0):IN."LINE NUMBER "L:IN."LINE  STEP
"U
500 REM
510 P.C.18;:T=V+$3F
520 DO:T=T+1
530 U.#T()S
540 T=AND(T,$FFE0)
550 E=V+$1FF
560 DO:E=E-1
570 U.#E()S
580 R=-1
590 DO:R=R+1:J=$20
600 DO:J=J-1:P=T+$20*R+J
610 U.#P()S
620 %0(R)=J
630 U.E=P
700 REM TAPE OUT
710 F.I=0 TOR
720 GOS.200:C.(0,1):P.L;' P."';
730 F.P=T+$20*I TO P+%0(I):P.C.$FF-#P;
740 N.P:P.'"':L=L+U
750 N.I:GOS.250
770 DO:U.AND(2,##E010):##E011=3
780 GOS.250:P.C.19;C.20
800 CU.(0,0):CLR(0,1):P." RE-EDIT ( Y OR N ) ? "
;
810 IFGET$="Y" CLR(0,1):G.315

READY
```

●グラフィックエディタ サンプル・プログラム

```
READY
)L.
800 P."
810 P."
820 P."
830 P."
840 P."
850 P."
860 P."
870 P."
880 P."
890 REM
```