PC-9801

Moving

スプライト機能がついた アニメ用 拡張グラフィック・ツール

■武石秀男

スプライト機能は，グラフィック・キャラクタがアニメのセル画よろしく画面上を自由に，しかもバックの画面と独立した形で動き回れる機能です．

ファミコンでその力を発揮し，最近ではシャープのX68000にも取り入れられ，その名を不動のものとしました．

このスプライト機能をソフトで，しかも，ハード版よりも機能豊富に，拡張性高く，実現できたら…．

夢の始まり

九年前の夏に，LKIT-16というマイコンを購入しました．今と違って画面表示が貧弱でグラフィック機能もありません．しかも，メーカーからの後のサポートがまるっきりありませんでした．マイナー機への飽くなきチャレンジがこのときから始まることになります．そして，夢の第一夜もこのときから始まったのです．

☆ 夢・第一部 LKIT-16版 （パナファコム）	◇グラフィック画面はなく，ＰＣＧの64個のキャラクタを駆使しての動画システム
☆夢・第二部 レベル３版 （日立）	◇グラフィック画面に直接吹き付ける方法での動画システム ◇LKIT-16版を継承してのIG（PCGと同じで256個使用可）版動画システム
☆ 夢・第三部 S1版 （日立）	◇拡張BASICとしての動画システム「Moving S1」
☆ 夢・第四部 PC-9801版 （ＮＥＣ）	◇ROM・DISK-BASICの拡張BASICとしての動画システム「Moving 98」 ◇MS-DOSデバイス・ドライバとしての動画システム．拡張BASICとして，また，アセンブラ・Ｃ・Pascal言語で活用する

スプライト機能実現の夢の変遷

このLKIT-16も今では巨大化し，８インチのドライブを２台装備したシステムもとうてい我がウサギ小屋には置けず，職場の一室でビニール・シートがかけられ，決してくることのないであろう夢をひたすら待ち続けています．

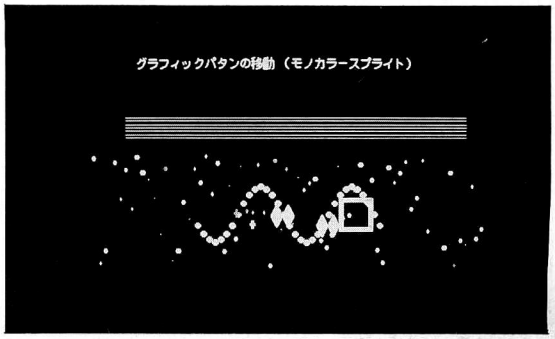
夢はレベル３，S1へと受け継がれ，そのたびに新たな夢の世界が広がっていきました．思えば，マイナー機志向が夢の大きな原動力であったようです．

スプライト&コマ送り

BASICはいろいろな欠陥を補いながら進化してきました．グラフィック機能においても，一昔前に比べれば目を見張るものがあります．

表示が速くなり，不定型のグラフィック・パターンの表示も可能になりました．しかし，動画（アニメ）に関する機能はまだまだ貧弱なようです．

たとえば，前に描いたパターンを消してしまい，立体的な表示ができません．これを最近のマイコンでは，スプラ



重ね合わせ処理が見事!!

イト機能という形でハード的に補っています．

〝Moving〟は，動画（アニメ）機能強化のための拡張グラフィック・ツールです．このツールによってグラフィック表示のための新たな機能が追加されます．

画面上のドット単位の位置に任意の色・大きさのグラフィック・パターンを瞬時に表示するもので，画面表示の高速化・鮮明さがとくに考慮されています．

コマ送り機能によってアニメ表示が簡単にできます．さらに，スプライト機能と，BG（バック・グランド）機能と呼ばれるグラフィック画面表示の際の優先順位付加機能は，奥行きのある立体的な画面の制作を，容易にしてくれます．

Movingで表示されるグラフィック・パターンは，アニメのセル画のような任意のサイズのグラフィック表示面（これをスプライト面と呼ぶ）の上に描かれたものと考えることができます．そして，その各スプライト面に番号がついています．これがパターン番号です．

●スプライトの利点

1つ1つのグラフィック・パターン表示用のスプライト面を予約・設定しておくことを「スプライト機能」といい，この機能の備った番号のグラフィック・パターンを表示することを「スプライト表示」と呼ぶことにします．

表示されたパターンは他のグラフィック・パターンの表示によって消されることはありません．ただし，表示優先順位の高いスプライト・パターンの影に隠れて見えなくなることはあります．

これに対して，スプライト機能の付加されていないグラフィック・パターンは他のパターンの表示によって消され，復元されることはありません．この特徴をうまく使い分けて有効に活用します．

●スプライトの定義

スプライト面のサイズは列（横）方向が8ドット（1ライン）単位で，行（縦）が1ドット単位で任意の大きさに定義できます．着色はモノカラースプライトを定義すると1色，RGBスプライトを定義すると，ドット単位で8色，または16色が設定できます．

スプライト面の優先順位は，パターン番号の小さいものが一番手前で，大きい番号のスプライト面ほど奥への表示となります．ただし，この優先順位を反転させることもできます．一番奥の面がBG面として設定できるので，背景表示などに利用できます．

1つのパターン番号に複数のグラフィック・パターンが設定できます．その1つ1つをコマと呼び，コマには番号がついています．表示の時にコマ番号を指定しない場合は自動的にコマ送りされます．この機能によって，簡単にアニメが実現できます．

意義と活用

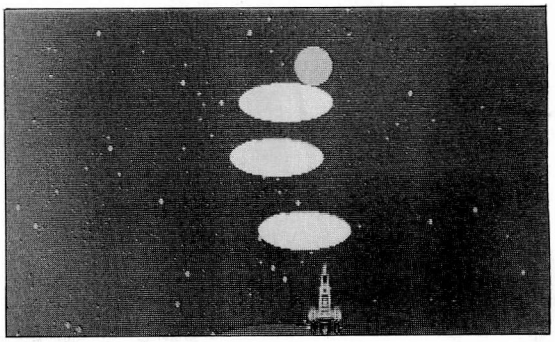
❶ Movingは「拡張BASIC」である

BASICは高級（より人間に近いという意味で）言語であり，使いやすい言語の最たるものです．人間にとってわかりやすい命令で，気楽な気持ちでプログラムが組めます．

しかし，それゆえに汎用的な命令がほとんどで，小回りのきくプログラムが作りにくいのです．そこで，ある特定の機能に力をおいたサブルーチンを用意し，それを簡単に使えるシステムが必要です．

Movingはグラフィック・アニメに重点をおいた機能を用意し，それらを拡張BASICコマンドとして利用します．

RGBスプライトのようす



❷ Movingは「システム・コール」である

機械語（アセンブラ）は低級（より機械に近いという意味で）言語です．人間にわかりにくい命令で，必要な機能はほとんど自分自身で用意しなければいけません．

本気になって取り組まなければならない言語です．幸い，マイコンの標準システムとして，ソフトウェア割り込みとしてのサブルーチンが用意され，それらが利用できるようになりました．

Movingは動画・アニメのための機能をシステム・コール（**INT40H**）で利用します．

❸ MovingはPascalやC言語用の『拡張グラフィック・ツールである』

CやPascalはアセンブラなみのハード的な機能性を持ち，システムなども書けます．

また，BASICなみの操作性を実現しています．しかし，グラフィック機能が非常に貧弱です．

ヘッダーファイルとして，C言語では〝Moving. H〟をPascalでは〝Moving. PAS〟をインクルードすることでBASICライクに動画グラフィックを活用できます．

組み込み

❶ MS-DOSのデバイス・ドライバとして組み込む

MS-DOSが起動されるとき，〝CONFIG. SYS〟ファイルの中に

DEVICE＝MOVING. SYS

を書き加える（MS-DOSのV3. 1以降では，**ADDDRV**コマンドで追加設定できる）

デバイス・ドライバ・ファイルの作り方はいろいろな記事で取り上げられているので，詳しい説明はそちらに譲ることにしますが，普通の実行ファイル（**EXE**，**COM**の拡張子のついたファイルなど）と違う点がいくつかありますので，その部分を若干説明しておきます．

ドライバとしてのいろいろな情報を持ったヘッダー部がプログラムの頭にあります．その中には以下のようないくつかのコール・ルーチンのアドレスが定義されています．

(a)ドライバが呼び出されるたびにコールされるルーチン（リスト1のENTRYルーチン）

フルシステム版ではこのルーチンにいろいろな機能を持たせていますが，コンパクト版では何もせずに復帰します．

(b)ドライバを組み込むときに一度だけコールされるルーチン（リスト1のINITIルーチン）

Movingをシステム・コール化するためのコール・ルーチ

▶8ビット機のCはどのくらい使えるのでしょうか，といっても，何年とかそういうのじゃなくて，「使いものになるか」ということです．8ビット・マシンは16ビットに比べてメモリ不足きみだし，いろいろな面で16ビット・マシン用のCに劣っていると思うのです．言語自体がどんなによくても，最も重要であるメモリ関係に絶対的な差が生じてくるのはまぁ仕方ないことですけど…．私はPC-8801mkIIFRとPC-286Vを使っているわけですが，88だったらSmall C，286だったらTurbo Cがあるわけですが，迷っているのです．ふつうの人なら286（16ビット機）

ン・アドレスを，割り込みベクタ・テーブルに登録します．フルシステム版ではパラメータによって，割り込み番号を指定できるようにしていますが，コンパクト版では40Hに定義されます．

上の組み込みによって，**INT40H** が実行されるたびに，システムのルーチン（**リスト1** の **MVCALL** ルーチン）がコールされます．

❷ N88-BASIC の拡張 BASIC として活用する

起動するとき，パラメータとして　／E：**MOVE_BAS. COM** を追加する（**N88-BASIC　／E：MOVE_BAS. COM**）と，グラフィック拡張コマンドとして新たな命令が追加されます．

MOVE_BAS. COM 実行ファイルは N88-BASIC が起動されるときに，一度だけ実行されますが，そのとき，以下のように，BASIC ワークエリア各値を設定し，BASIC の拡張を行ないます（**リスト2** 拡張 BASIC 設定の部分を参照）．

(a)$159C_H$，$15A0_H$に拡張ルーチンのオフセット・アドレスを設定

(b)$159E_H$，$15A2_H$に拡張ルーチンのセグメント・アドレスを設定

(c)アドレスの1593_Hに1を設定し，BASIC 拡張の宣言

この設定方法は，ROM，DISK 版 BASIC でも同じです．なお，この実行ファイルは，N88-BASIC が終了するまで常駐します．

ファイルの作成

●**リスト1** のアセンブラソース・テキストを以下のようにアセンブル/リンクすると，**MOVING. ASM** から **MOVING. SYS** が作成されます．

```
MASM      MOVING, MOVING;
LINK      MOVING, MOVING;
EXE2BIN   MOVING. EXE  MOVING. SYS;
```

●**リスト2** のアセンブラ・ソーステキストを以下のようにアセンブル/リンクすると，**MOVE_BAS. ASM** から **MOVE_BAS. COM** が作られます．

```
MASM      MOVE_BAS, MOVE_BAS;
LINK      MOVE_BAS. MOVE_BAS;
EXE2BIN   MOVE_BAS.  EXE  MOVE_BAS.
COM;
```

使用手順

① **MVSET** 文でパターン数，システム・メモリー領域を設定

② **GSIZE** 文で各パターンのサイズを変更

③各パターンのグラフィック・フォントを設定

- **GDSET** 文で1ライン（8ビット）単位で設定
- **GDGET** 文でグラフィック画面よりコピー
- **GDTFR** 文で他のパターンよりコピー（フルシステム版のみ）
- **KJTFR** 文で漢字フォントを設定（フルシステム版のみ）

④グラフィック画面に表示

- **GDISP** 文で1個ずつの表示
- **GZOOM** 文で拡大表示（フルシステム版のみ）
- **GPEEK**，または，**SPRITE** 文で複数個同時表示

```
10  CLEAR  &H2000          ←機械語使用領域を確保
20  MVSET  8,4,2,&H2000    ←①
30  GSIZE(2,16),0          ←②
40  CIRCLE(8,8),7,4,,,,F   ←塗り潰した円を描く
50  GDGET(0,0),0           ←③
60  FOR  X=0  TO  639      ←繰り返しルーチン
70       GDISP(X,100),0    ←④
80  NEXT  X
90  END
```

最後に

CAI，CMI などと銘打って，コンピュータ戦略の矛先が教育に向けられ，学校現場に大きな混乱が巻き起こっています．

現在勤務している学校にも，昨年の暮れ，CAI のためのマイコン47台が設置され，先生方は，研修会などに参加され，それぞれ，違和感，期待感の入り交じった複雑な気持ちで，訪れた教育革命に，対処されているようです．

私もようやく「Moving」の組み込まれたシステムをネットワーク化し CAI の授業に活用できるところまでこぎつけました．

Moving は CAI ネットワーク・システムの裏方さんとしても大活躍中で，この秋から学校関係のパソコン通信が始まるということで，ソフトの充実が急がれています．

*

現在，「Moving」は4096色表示，インターバル・タイマーによる制御，マウス対応などバージョン・アップしていますが，そのアセンブラのソース・テキストは5000行に及ぶものです．誌面の関係でとうてい掲載できないので，この記事を書くにあたりコンパクト版を作成しました．機能は大分落としてありますが，動画の醍醐味は充分に味わってもらえると思います．

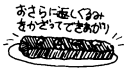
N88-BASIC の拡張コマンドに設定するためのマシン語プログラムや，C，Pascal 言語で活用するためのインクルード・ファイルもあわせて掲載しますので，これから自分のシステムをデバイス・ドライバに組み込み，C や Pascal で活用しようと思っているみなさんの参考にしていただければ幸いです．その他，理解の手助けとなるような資料，リストを掲載します．

ROM・ディスク版のシステムはコムパック（システム名：Moving98）でディスケット・サービスされていますが，MS-DOS 版のフルシステムも提供します．

CAI，CMI，ネットワーク，パソコン通信と，なかなか自分の見たい夢がみられませんが，いつか，夢の続き，または新しい夢のお話しをしたいと思っています．

◘参考文献

1) MS-DOS で4096色を，I/O'87年1月号
2) 応用 MS-DOS，ASCII
3) PC-9801解析マニュアル，秀和システムトレーディング



をとるでしょうけど，私の場合，88の方がよく知っているし，逆に286の方はあまり知らないので，使いなれた88をとるか，それとも，性能，使いやすさで286をとるか…難しいところであります．68000マシンだったら迷わずとびついてしまうんですが…．8086系はどうも気がひけてしまう私でした．8ビット機や16ビット機でCを使っている人，どうでしょうか？

（久松由実と南野陽子の好きな魔道戦士）

```
/* ------------------------------------------------------------
**          ROM版  GLIO   ヘッダー・ファイル
**
**                  File "GLIO.H"
**
**              1987 Jun. 15          By H.Matumoto
**              1987 Sep. 17   訂正      By H.Takeishi
------------------------------------------------------------ */

char    WORK[5120];
int_stack = 4096;

/*---------- define const ----------------------------------------*/

#define GINIT   0xa0    /* initialize */
#define GSCRE   0xa1    /* screen mode set */
#define GVIEW   0xa2    /* set view port */
#define GCOLO   0xa3    /* screen color set */
#define GCLS    0xa5    /* clear graphic screen */
#define GPSET   0xa6    /* pset */
#define GLINE   0xa7    /* line */
#define GCIRC   0xa8    /* circle */
#define GPAINT  0xa9    /* paint */
#define GROLL   0xae    /* scroll graphic screen */

/*------------------------------------------------------------*/
void    glio(n, point)      /* LIO call function. */
    int n,point;
 {
    union REGS ireg, oreg;
    ireg.x.bx = point;
    int86(n, &ireg, &oreg);
 }
/*------------------------------------------------------------
**              グラフィック画面の初期化 init()
*/

void    init()              /* initialize */
 {
    char    buf[20];
/*  int i;  */
    buf[0]  = 0x90;
    buf[1]  = 0xf9;
/*  for(i=0; i<16; i++)
        {
        movedata(0xf990, 6+i*4, 0x0000, 0xa0*4+i*4, 2);
        poke(0x0000, 0xa0*4+i*4+2, buf, 2);
        }
    movedata(0x0000, 0x1d*4, 0x0000, 0x0c5*4, 4);   */
    glio(GINIT);
 }

/*------------------------------------------------------------
**              スクリーン・モード設定  screen()
*/

void    screen(a, b, c, d)          /* screen set */
    char    a, b, c, d;
 {
    char    buf[20];
    buf[0]  = a;
    buf[1]  = b;
    buf[2]  = c;
    buf[3]  = d;
    glio(GSCRE,buf);
 }

/*------------------------------------------------------------
**          テキスト、または、グラフィック画面のクリア    cls()
*/

void    cls(a)
    int a;
{

    if( a <1 || a >3 ) a = 2;
    if( a & 1 )  printf("\x1b[2J");
    if( a & 2 )  glio(GCLS);
 }

/*------------------------------------------------------------
**          グラフィック画面のカラー設定  color()
*/

void    color(a, b, c, d)
    char    a,      /* back ground color */
        b,      /* boder color      */
        c,      /* for ground color  */
        d;      /* palette mode 無効 */
 {
    char    buf[20];

    buf[1]  = a;
    buf[2]  = b;
    buf[3]  = c;
    buf[4]  = d;        /* 今のところ無効    */

    glio(GCOLO,buf);
 }

/*------------------------------------------------------------
**          直線の描画  line()
*/

void    line(x0, y0, x1, y1, c, f, s, t1, t2)
    int     x0,y0,      /* start point */
    x1,y1;      /* end point   */
    char    c,          /* color       */
        f,          /* 0/line 1/box 2/box fill */
        s,          /* line style switch  0/off 1/on */
        t1,         /* line style (L) */
        t2;         /* line style (H) */
 {
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = (int)x0;
    ptr[1]  = (int)y0;
    ptr[2]  = (int)x1;
    ptr[3]  = (int)y1;
    buf[8]  = c;
    buf[9]  = f;
    buf[10] = s;
    buf[11] = t1;
    buf[12] = t2;

    glio(GLINE, buf);
 }

/*------------------------------------------------------------
**      円の描画    circle()
*/

void    circle(x0, y0, rx, ry, c, bf)
    int x0, y0;     /* center of circle. */
    int rx;         /* xhankei */
    int ry;         /* yhankei */
    char    c;          /* color */
    char    bf;         /* paint */
 {
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = (int)x0;
    ptr[1]  = (int)y0;
    ptr[2]  = rx;
    ptr[3]  = ry;
    buf[8]  = c;
    buf[9]  = bf*0x20;
    buf[18] = c;

    glio(GCIRC, buf);
 }

/*------------------------------------------------------------
**      領域の描画  paint()
*/

void    paint(x, y, c1,c2)
    int     x, y;
    char    c1,c2;
 {
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = x;
    ptr[1]  = y;
    buf[4]  = c1;
    buf[5]  = c2;
    ptr[3]  = (int)(WORK+100);
    ptr[4]  = (int)WORK;

    glio(GPAINT, buf);
 }

/*------------------------------------------------------------
**              点の描画    pset()
*/

void    pset(x, y, c)
    int     x, y;
    char    c;
 {
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = x;
    ptr[1]  = y;
    buf[4]  = c;

    glio(GPSET, buf);
 }

/*------------------------------------------------------------
**              グラフィック画面のロール    roll()
*/

void    roll(td, yd, flg)
    int td,     /*  x方向のドット数  */
        yd;         /*  y方向のドット数  */
    char    flg;
 {
```


▶最近I/Oが薄くなってきたと思いませんか？というわけで調べてみました．僕の持っている一番古い1983年5月号では600ページ，同年6，8月号で656ページを記録し減り始め，次の年1月号で584ページ，そして1987年4月号でとうとう496ページとなり，今月号(といっても今日は7月18日で都会の人には先月号なんだよな」では440ページになってしまい，約5年間で200ページ余り減ったことになります．パソコン

```
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = td;
    ptr[1]  = yd;
    buf[4]  = flg;

    glio(GROLL, buf);
 }

/*-------------------------------------------------------------------
**          ビュウポートの設定      view()
*/

void    view(x1, y1, x2, y2, c, p)
    int x1, y1,
        x2, y2;
    char    c, p;
 {
    char    buf[20];
    int *ptr;

    ptr = (int *)buf;
    ptr[0]  = x1;
    ptr[1]  = y1;
    ptr[2]  = x2;
    ptr[3]  = y2;
    buf[8]  = c;
    buf[9]  = p;

    glio(GVIEW, buf);
 }

/*-------------------------------------------------------------------
**          テキスト画面の位置指定      locate()
*/

void    locate(a,b)
    int a,b;
{
    printf (" [%d;%dH",b,a);
 }

/*-------------------------------------------------------------------*/
```

glio.pas

```
(* ------------------------------------------------------------------*)
(*          ROM版  GLIO   ヘッダー・ファイル                         *)
(*                                                                   *)
(*                          File "glio.pas"                          *)
(*                                                                   *)
(*                    1987 Jun. 15          by H.Matumoto          *)
(*                    1987 Jul. 30   訂正   by H.Takeishi          *)
(*-------------------------------------------------------------------*)

(*---------- define const ----------------------------------------*)

const
        GINIT   = $a0 ; (* initialize *)
        GSCRE   = $a1 ; (* screen mode set *)
        GVIEW   = $a2 ; (* set view port *)
        GCOLO   = $a3 ; (* screen color set *)
        GCLS    = $a5 ; (* clear graphic screen *)
        GPSET   = $a6 ; (* pset *)
        GLINE   = $a7 ; (* line *)
        GCIRC   = $a8 ; (* circle *)
        GPAINT  = $a9 ; (* paint *)
        GROLL   = $ae ; (* scroll graphic screen *)

Type

   Reg_glio = record
                ax,bx,cx,dx,bp,si,di,ds,es,flag : integer
              end;
   var
      worklio : array[0..5119] of byte ;
      hanpa   : integer ;
      GrxReg  : Reg_glio;
(*-------------------------------------------------------------------*)
procedure       reg_set( var  GrxReg : Reg_glio ) ;

     begin
        GrxReg.bx := ofs(worklio) mod $10 ;
        hanpa     := ofs(worklio)-GrxReg.bx  ;
        GrxReg.ds := dseg+ofs(worklio) div $10 ;
     end;

(*-------------------------------------------------------------------*)
(*              グラフィック画面の初期化 init()                      *)


procedure       init ;                      (* initialize *)
  var
     GrxReg : Reg_glio ;
     begin
(*      int     i;      *)
        worklio[1] := $f990;

(*      for(i:=0; i<16; i++)
                begin
                    movedata($f990, 6+i*4, $0000, $a0*4+i*4, 2);
                    poke($0000, $a0*4+i*4+2, buf, 2);
                end;
        movedata($0000, $1d*4, $0000, $0c5*4, 4);       *)
        reg_set( GrxReg );
        Intr( GINIT, GrxReg );
 end;

(*-------------------------------------------------------------------
**              スクリーン・モード設定  screen()
*)

procedure       screen(a, b, c, d : integer );          (* screen set *)
   var
     buf_chr : array[0..3] of byte absolute worklio ;
     GrxReg  : Reg_glio ;
   begin
        buf_chr[0] := a and $00ff ;
        buf_chr[1] := b and $00ff ;
        buf_chr[2] := c and $00ff ;
        buf_chr[3] := d and $00ff ;
        reg_set( GrxReg );
        Intr( GSCRE, GrxReg );
   end;

(*-------------------------------------------------------------------
**          テキスト、または、グラフィック画面のクリア    cls()
*)

procedure       cls( a : integer );                     (* cls() *)
  var
     GrxReg : Reg_glio ;
  begin
        if (a<1) or (a>15) then a := 3;
        if (a and 1)>0 then
                write(' [2J');
        if (a and 2)>0 then
          begin
             reg_set(GrxReg);
             Intr( GCLS, GrxReg );
          end;
        if (a and 4)>0 then
                write(' [>1h');
        if (a and 8)>0 then
                write(' [>1l');
   end;

(*-------------------------------------------------------------------
**          グラフィック画面のカラー設定  color()
*)

procedure       color(a, b, c, d : integer );
                (* a =  back ground color    *)
                (* b =  boder color          *)
                (* c =  for ground color     *)
                (* d =  palette mode    無効 *)
  var
     buf_chr : array[0..3] of byte absolute worklio ;
     GrxReg  : Reg_glio ;
  begin
        buf_chr[0] := a;
        buf_chr[1] := b;
        buf_chr[2] := c;
        buf_chr[3] := d;            (* 今のところ無効    *)
        reg_set( GrxReg );
        Intr( GCOLO, GrxReg );
  end;

(*-------------------------------------------------------------------
**          直線の描画  line()
*)

procedure       line(x0, y0, x1, y1, c, f, s, t1, t2 : integer );
                (* x0 , y0 =    start point *)
                (* x1 , y1 =    end point   *)
                (* c =          color       *)
                (* f =          0/line 1(box 2/box fill *)
                (* s =          line style switch  0/off 1/on *)
                (* t1=          line style (L) *)
                (* t2=          line style (H) *)
   var
     buf_lio : array[0..6] of integer absolute worklio ;
     GrxReg  : Reg_glio ;
   begin
        buf_lio[0] := x0;
        buf_lio[1] := y0;
        buf_lio[2] := x1;
        buf_lio[3] := y1;
        buf_lio[4] := ( c  and $00ff ) or (  f shl 8 );
        buf_lio[5] := ( s  and $00ff ) or ( t1 shl 8 );
        buf_lio[6] := t2 and $00ff ;
        reg_set( GrxReg );
        Intr( GLINE, GrxReg );
  end;

(*-------------------------------------------------------------------
**              円の描画        circle()
*)

procedure       circle(x0, y0, rx, ry, c, bf : integer );
                        (* x0 = center of circle. *)
```

界も冷え始めているのかと思ってしまいました。ソフトハウスの広告が少なくなっちゃったよね。P.S.最近I/Oにゲームが載らなくなってきている。どうしたんだ。

（貯金が10万円しかないのにX68Kが欲しいモロアッチ助教授）

```
                        (* y0 = xhankei *)
                        (* rx = yhankei *)
                        (*  c = color   *)
                        (*  c = paint   *)
  var
    buf_lio : array[0..9] of integer absolute worklio ;
    GrxReg  : Reg_glio ;
  begin
        buf_lio[0] := x0;
        buf_lio[1] := y0;
        buf_lio[2] := rx;
        buf_lio[3] := ry;
        buf_lio[4] := ( c  and $00ff ) or ((bf*$20) shl 8 );
        buf_lio[9] := c and $00ff ;
        reg_set( GrxReg );
        Intr( GCIRC, GrxReg );
  end;


(*-------------------------------------------------------------------
**              領域の描画       paint()
*)

procedure       paint(x, y, c1, c2 : integer );

  var
    buf_lio : array[0..4] of integer absolute worklio ;
    gp_work : array[0..255] of byte;
    GrxReg  : Reg_glio ;
  begin
        buf_lio[0] := x;
        buf_lio[1] := y;
        buf_lio[2] := ( c1 and $00ff ) or ( c2 shl 8 );
        buf_lio[3] := ofs(gp_work[255])-hanpa ;
        buf_lio[4] := ofs(gp_work[0])-hanpa ;
        reg_set( GrxReg );
        Intr( GPAINT, GrxReg );
 end;

(*-------------------------------------------------------------------
**              点の描画     pset()
*)

procedure       pset(x, y, c : integer );
   var
     buf_lio : array[0..2] of integer absolute worklio ;
     GrxReg  : Reg_glio ;
  begin
        buf_lio[0]      := x;
        buf_lio[1]      := y;
        buf_lio[2]      := c and $00ff;
        reg_set( GrxReg );
        Intr( GPSET, GrxReg );
  end;

(*-------------------------------------------------------------------
**              グラフィック画面のロール     roll()
*)

procedure       roll(td, yd, flg : integer );
                        (*  td = x方向のドット数  *)
                        (*  yd = y方向のドット数  *)
  var
     buf_lio : array[0..2] of integer absolute worklio ;
     GrxReg : Reg_glio ;
  begin
        buf_lio[0] := td;
        buf_lio[1] := yd;
        buf_lio[2] := flg;
        reg_set( GrxReg );
        Intr( GROLL, GrxReg );
  end;

(*-------------------------------------------------------------------
**           ビュウポートの設定      view()
*)

procedure       view(x1, y1, x2, y2, c, p :integer );
   var
     buf_lio : array[0..4] of integer absolute worklio ;
   begin
        buf_lio[0] := x1;
        buf_lio[1] := y1;
        buf_lio[2] := x2;
        buf_lio[3] := y2;
        buf_lio[4] := ( c and $00ff ) or ( p shl 8 );
        reg_set( GrxReg );
        Intr( GVIEW, GrxReg );
  end;

(*-------------------------------------------------------------------
**            テキスト画面の位置指定     locate()
*)

procedure       locate(a,b : integer );
  begin
        write (' [',b,';',a,'H');
  end;

(*-------------------------------------------------------------------*)
```

MOVE_BAS.ASM

```
        ; ***********************************************
        ;  ＭＳ－ＤＯＳ版Ｎ８８ＢＡＳＩＣ拡張のための
        ;  Ｍｏｖｉｎｇコンパクト版常駐ファイル
        ;  File "MOVE_BAS.ASM
        ;               1988  8/3    By  H.Takeishi
        ; ***********************************************

        MVFLG   EQU     "HT"    ;  Ｍｏｖｉｎｇフラッグ
        MVCALL  EQU     40H     ;  ＩＮＴ　４０Ｈ
        BASWK1  EQU     1590H   ;  ＢＡＳＩＣ作業領域１
        BASWK2  EQU     141AH   ;  ＢＡＳＩＣ作業領域２
        VALWK1  EQU     6E8H    ;  変数変換作業域１
        VALWK2  EQU     6EAH    ;  変数変換作業域２

        CODE    SEGMENT
                ASSUME  CS:CODE,DS:CODE,ES:CODE

        ;  ***  拡張 Ｂａｓｉｃ 設定  ****************

        EXBAS   PROC    FAR
                PUSH    DS
                PUSH    ES
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    SI
                PUSH    DI

                XOR     AX,AX
                MOV     DS,AX
                MOV     SI,MVCALL*4
                MOV     AX,[SI]+02H
                CMP     AX,0060H
                JNE     EXBASOK

                PUSH    CS
                POP     DS
                MOV     DX,OFFSET NGMSG
                MOV     AH,09H
                INT     21H
                JMP     SHORT TMWAIT

        EXBASOK:MOV     BX,CS
                MOV     AX,OFFSET CODEND
                MOV     CL,4
                ADD     AX,0FH
                SHR     AX,CL
                ADD     AX,BX
                MOV     DS,AX
                MOV     CS:DSSTAK,AX
                MOV     SI,BASWK1
                MOV     AX,OFFSET ENTRY
                MOV     [SI]+0CH,AX
                MOV     [SI]+10H,AX
                MOV     [SI]+0EH,BX
                MOV     [SI]+12H,BX
                MOV     AL,1
                MOV     [SI]+03H,AL
                PUSH    CS
                POP     DS
                MOV     DX,OFFSET SETMSG
                MOV     AH,09H
                INT     21H
                MOV     DX,CS

        TMWAIT: MOV     CX,10
        DMLOP1: XOR     AX,AX
        DMLOP2: DEC     AX
                JNE     DMLOP2
                LOOP    DMLOP1

                POP     DI
                POP     SI
                POP     CX
                POP     BX
                POP     AX
                POP     ES
                POP     DS
                RET
        EXBAS   ENDP

        SETMSG  DB      0DH,0AH,0AH,0AH,0AH
                DB      "拡張ＢＡＳＩＣとして "
                DB      "Ｍｏｖｉｎｇ が設定されました"
                DB      0AH,0DH,'$'
        NGMSG   DB      0DH,0AH,0AH,0AH,0AH
                DB      "Ｍｏｖｉｎｇ ドライバー が"
                DB      "設定されていません"
                DB      0AH,0DH,'$'
        DSOUND  DW      10000,10
                DW      6 DUP(MVFLG)


        ; ****  拡張 Ｂａｓｉｃ エントリ  ***************

        ENTRY   PROC    FAR
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX
                PUSH    SI
```


▶ある日，モニタモードで…．0000：MOV AX，1 0003：MOV SP，1 0006：と入力したのちG0でリターン・キーを押したとたんリセットがかかってしまいました．CPUが80286のパソコンを持っている人は実験してみてください(私の機種はPC-9801VX2です．FM-Rなどでは意味がないかもしれません)．またどうしてこうなるのか知っている人がいたら教えてください．V30ではかかりませんでした．

（松永夏代子と成清加奈子が好きな南翔子のファンの青い季節）

```
        PUSH    DI
        PUSH    BP
        PUSH    ES
        PUSH    DS

COMCHK: DEC     SI      ; コマンドチェック
        MOV     DX,SI
        MOV     DI,OFFSET COMTBL
        MOV     BX,CS
        MOV     ES,BX
        XOR     CH,CH
        XOR     BX,BX
CHKLOP: MOV     SI,DX
        MOV     CL,CS:[DI]
        CMP     CL,0FFH
        JE      NOEND
        INC     DI
        REP     CMPSB
        JE      EQOL
        ADD     DI,CX
        INC     BX
        INC     DI
        JMP     SHORT CHKLOP
EQOL:   PUSH    BX
        MOV     AL,CS:[DI]
        CMP     AL,0
        JNE     TOGET1
        CALL    GETDA0
        JMP     SHORT TOCALL
TOGET1: CALL    GETDA1
TOCALL: POP     AX      ; 拡張コマンドあり
        OR      AL,20H
        MOV     SI,OFFSET BATODA
        MOV     DX,CS
        INT     MVCALL          ; 実行へ

MVEND:  STC
        JMP     SHORT MVEN1
NOEND:  CLC             ; 拡張コマンドなし
MVEN1:  POP     DS
        POP     ES
        POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENTRY   ENDP

; *** 拡張 Basic コマンドテーブル ******

COMTBL: DB      6,'M',4,'VSET',0
        DB      6,'G',4,'SIZE',1
        DB      6,'G',4,'DSET',0
        DB      6,'G',4,'DISP',1
        DB      6,'G',4,'POKE',1
        DB      6,'G',4,'PEEK',1
        DB      6,'G',4,'DGET',1
        DB      6,'B',4,'GSET',0
        DB      6,'U',4,'NION',0
        DB      7,'S',5,'PRITE',0
        DB      0FFH

DSSTAK  DW      0       ; DS待避
DATASK  DW      16 DUP(0)

BATODA  DW      0       ; Basicからの
        DW      0       ; データー受け渡し領域
        DW      0
        DW      0
        DW      0
        DW      0
BADAXY  DW      0       ; (X,
        DW      0       ;    Y)
STACK   DW      0

WKDSET: MOV     DI,OFFSET BATODA
        MOV     AX,CS
        MOV     ES,AX
        MOV     AX,MVFLG
        MOV     CX,8
        REP     STOSW
MVWK:   MOV     BP,OFFSET BATODA
        RET

GETDA1: CALL    WKDSET
ALSET1: LODSB
        CMP     AL,01H
        JE      ALSET1
        CMP     AL,","
        JE      LOP3
        CMP     AL,"("
        JNE     ALSET3

XYSET:  MOV     BP,OFFSET BADAXY
        CALL    DTPEEK
        LODSB
        CMP     AL,","
        JNE     MOVEN
        CALL    DTPEEK
        LODSB
        CMP     AL,")"
        JNE     MOVEN
        MOV     BP,OFFSET BATODA
        JMP     SHORT LOP2
GETDA0: CALL    WKDSET
ALSET2: LODSB
ALSET3: CMP     AL,0
        JE      MOVEN
        CMP     AL,":"
        JE      MOVEN
        CMP     AL,","
        JE      PASSDT
        CMP     AL,1
        JE      ALSET2
LOP1:   CMP     AL,22H
        JNE     ALSET5
        MOV     CX,2020H
        LODSB
        CMP     AL,22H
        JE      LOP2
        MOV     CH,AL
        LODSB
        CMP     AL,22H
        JNE     ALSET4
        MOV     CS:[BP],CX
        INC     BP
        INC     BP
        JMP     SHORT LOP2

ALSET4: MOV     CL,AL
        MOV     CS:[BP],CX
        INC     BP
        INC     BP
        LODSB
        CMP     AL,22H
        JE      LOP2
        DEC     SI
        JMP     SHORT LOP2

ALSET5: DEC     SI
LOP0:   CMP     BP,OFFSET STACK
        JAE     MOVEN
        CALL    DTPEEK
LOP2:   LODSB
        CMP     AL,0
        JE      MOVEN
        CMP     AL,":"
        JE      MOVEN
        CMP     AL,1
        JE      LOP2
        CMP     AL,0BH
        JE      LOP2
        CMP     AL,","
        JNE     LOP1
LOP3:   LODSB
        CMP     AL,1
        JE      LOP3
        CMP     AL,","
        JNE     LOP1
PASSDT: INC     BP
        INC     BP
        JMP     SHORT LOP3

MOVEN:  DEC     SI
        MOV     BX,VALWK1
        MOV     [BX],SI
        RET

DTPEEK: PUSH    BP
        MOV     BX,VALWK2
        MOV     [BX],SI
        MOV     DI,4AH
        INT     0C4H
        MOV     BX,VALWK2
        MOV     SI,[BX]
        MOV     BX,BASWK2
        MOV     AX,[BX]
        POP     BP
        MOV     CS:[BP],AX
        INC     BP
        INC     BP
        RET
CODEND:

CODE    ENDS
        END     EXBAS
```

MOVING.ASM

```
;*********************************************
;       MS-DOSコンパクト版
;       Moving デバイス・ドライバー
;       アセンブラ・テキスト
;       File "MOVING.ASM
;       1988 8/3        By H.Takeishi
;*********************************************

MVFLG   EQU     "HT"    ; Movingフラッグ
SYSCALL EQU     40H     ; システムコールコード
COMNUM  EQU     10      ; 拡張コマンド数
STRBUF  EQU     0202H   ; BASICライン出力
```

▷SPいじられ、バッバラバー!?

(暴走ぎらいのお)

```
DIMSTA  EQU     06A2H   ; 配列開始セグメント
DIMEND  EQU     06D0H   ; 配列終了セグメント
RAMMAX  EQU     1402H   ; 実装ＲＡＭ最大値
STAADR  EQU     1404H   ; 機械語開始セグメント
VRMSEG  EQU     0A800H  ; ＶＲＡＭ開始セグメント
BIOSEG  EQU     0F990H  ; ＢＩＯＳセグメント
WIDE80  EQU     80      ; 画面水平表示カラム数
LIN400  EQU     400     ; 画面垂直表示ライン数

COMMAND EQU     2
STATUS  EQU     3
TRANS   EQU     14
BRKADR  EQU     14
COUNT   EQU     18

CODE    SEGMENT PUBLIC
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE

; *************  ヘッダー  **********************

HEADER  DD      -1
        DW      8000H
        DW      STRTGY
        DW      ENTRY
        DB      "MOV     "
CMDTBL  DW      INITI
        DW      12 DUP(EXIT)
PACKET  DD      ?
CMDCNT  DB      0
BLKCMT  DB      -1
POINTX  DW      0
POINTY  DW      0
COLOR   DW      0

; ****  拡張コマンド ジャンプ アドレス  ********

JMPTBL  DW      OFFSET MVSET
        DW      OFFSET GSIZE
        DW      OFFSET GDSET
        DW      OFFSET GDISP
        DW      OFFSET GPOKE
        DW      OFFSET GPEEK
        DW      OFFSET GDGET
        DW      OFFSET BGSET
        DW      OFFSET UNION
        DW      OFFSET SPRITE

; ************  システム・ワーク・エリア  ******

DATASK  DW      32 DUP(0)       ; [BP]-20H～-01H
BATODA  DW      0       ;  Ｄ０
        DW      0       ;  Ｄ１
        DW      0       ;  Ｄ２
        DW      0       ;  Ｄ３
        DW      0       ;  Ｄ４
        DW      0       ;  Ｄ５
BADAXY  DW      0       ;  Ｄ６（Ｘ，
        DW      0       ;  Ｄ７      Ｙ）
BPOINT  DW      132 DUP(0)
S_TOP:
MVBUSY  DB      0       ; システム使用中フラッグ
SSSTAK  DW      0       ; スタックセグメント待避
SPSTAK  DW      0       ; スタックポインタ待避１
SPSTAK2 DW      0       ; スタックポインタ待避２
SISTAK  DW      0
DXSTAK  DW      0

; ****  初期設定時のメッセージ  ***************

IPLMSG  DB      0AH,0DH
        DB      "Ｍｏｖｉｎｇが設定されました"
        DB      0AH,0DH,'$'

; ****  エラー・メッセージ  *******************

ERRMSG  DB      0AH,0DH
        DB      "《 Ｍｏｖｉｎｇ Ｅｒｒｏｒ 》"
        DB      0AH,0AH,0DH,'$'
MSGER0  DB      "範囲外の値が入力されました"
        DB      0AH,0AH,0DH,'$'
MSGER1  DB      "システムのメモリーがオーバー"
        DB      "しました",0AH,0AH,0DH,'$'
MSGER2  DB      "システムが設定されていません"
        DB      0AH,0AH,0DH,'$'

; ************  ストラテジー  *****************

STRTGY  PROC    FAR
        MOV     WORD PTR [PACKET],BX
        MOV     WORD PTR [PACKET+2],ES
        RET
STRTGY  ENDP

; ****  ドライバー・コールルーチン  ************

ENTRY   PROC    FAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
        PUSH    DS
        PUSH    ES
        LDS     BX,CS:[PACKET]
        MOV     CX,[BX+COUNT]
        MOV     AL,[BX+COMMAND]
        CBW
        CMP     AX,12
        JBE     CMDOK
CMDERR: MOV     AX,8103H
        JMP     SHORT EXIT2

CMDOK:  LES     DI,[BX+TRANS]
        SHL     AX,1
        MOV     SI,AX
        CALL    CS:CMDTBL[SI]
EXIT:   MOV     AX,0100H

EXIT2:  LDS     BX,CS:[PACKET]
        MOV     [BX+STATUS],AX
        XOR     AX,AX
        MOV     [BX+COUNT],AX
        POP     ES
        POP     DS
        POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENTRY   ENDP
; ***  ；ドライバー初期設定ルーチン  *********

INITI   PROC
        LDS     BX,CS:[PACKET]
        MOV     AX,OFFSET CODEND
        MOV     [BX+BRKADR],AX
        MOV     [BX+BRKADR+2],CS
                ;  ***  ＩＮＴ４０Ｈの設定 *****
        MOV     BX,SYSCALL*4
        XOR     AX,AX
        MOV     DS,AX
        MOV     AX,OFFSET MVCALL
        MOV     [BX],AX
        MOV     AX,CS
        MOV     [BX]+02H,AX
                ;  ***  ＧＬＩＯ  の設定  ******
        XOR     AX,AX
        MOV     ES,AX
        MOV     AX,BIOSEG
        MOV     DS,AX
        MOV     SI,6
        MOV     DI,0A0H*4
        MOV     CX,20H
BIOSLP: MOVSW
        STOSW
        LEA     SI,[SI]+2
        LOOP    BIOSLP
        XOR     AX,AX
        MOV     DS,AX
        MOV     SI,01DH*4
        MOV     DI,0C5H*4
        MOV     CX,2
        REP     MOVSW
                ; ********************************
        MOV     AX,CS
        MOV     DS,AX
        MOV     ES,AX
        MOV     BX,OFFSET CODEND+300H
        MOV     CL,4
        SHR     BX,CL
        MOV     AH,4AH
        INT     21H
        MOV     AX,CS
        MOV     DS,AX
        MOV     AH,09H
        MOV     DX,OFFSET IPLMSG
        INT     21H
        MOV     AL,0DH
        OUT     0A2H,AL
        RET
INITI   ENDP

;  ***  機械語サブルーチン・エントリ  **********

MVCALL  PROC    FAR
        TEST    BYTE PTR CS:MVBUSY,1
        JE      NOBUSY
        IRET
NOBUSY: MOV     BYTE PTR CS:MVBUSY,1
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
        PUSH    DS
        PUSH    ES
        MOV     CS:SSSTAK,SS
        MOV     CS:SPSTAK,SP
        MOV     CS:SISTAK,SI
        MOV     CS:DXSTAK,DX
        MOV     BX,CS
        MOV     SS,BX
```

```
        MOV     SP,OFFSET S_TOP
        MOV     BP,OFFSET BPOINT
        MOV     [BP]-3EH,DS
        MOV     [BP]-40H,AL
        AND     AL,1FH
        JNE     TOMVCK

        TEST    BYTE PTR [BP]-40H,20H
        JE      PASCHK
                        ;  ＢＡＳＩＣの設定
        MOV     BX,RAMMAX
        MOV     CX,[BX]
        MOV     [BP]-3CH,CX
        MOV     CX,[BX]+2
        MOV     [BP]-3AH,CX
        MOV     BX,DIMSTA
        MOV     CX,[BX]
        MOV     [BP]-38H,CX
        MOV     BX,DIMEND
        MOV     CX,[BX]
        MOV     [BP]-36H,CX
        JMP     SHORT CALLOK

PASCHK: TEST    BYTE PTR [BP]-40H,40H
        JE      CALLOK
                        ;  ＰＡＳＣＡＬの設定
        MOV     BX,[BP]-04H
        MOV     [BP]-3AH,CX
        MOV     BX,[BP]-02H
        MOV     [BP]-3CH,CX
        JMP     SHORT CALLOK

TOMVCK: CMP     AL,COMNUM
        JAE     SHORT CALLEN
        CMP     WORD PTR [BP]+2AH,MVFLG
        JNE     ERROR2

CALLOK: MOV     BX,CS
        MOV     ES,BX
        MOV     DS,DX
        MOV     CX,8
        MOV     DI,OFFSET BATODA
        REP     MOVSW
        XOR     AH,AH
        SHL     AL,1
        MOV     BX,OFFSET JMPTBL
        ADD     BX,AX
        CALL    WORD PTR CS:[BX]  ;  コマンドへ
        XOR     AX,AX
CALLEN: MOV     SS,CS:SSSTAK
        MOV     SP,CS:SPSTAK
        POP     ES
        POP     DS
        POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        MOV     BYTE PTR CS:MVBUSY,0
        IRET            ;  Ｍｏｖｉｎｇ終了
MVCALL  ENDP

;  ***  エラー処理  ****************************

ERROR0: MOV     DX,OFFSET MSGER0
        JMP     SHORT TOERR
ERROR1: MOV     DX,OFFSET MSGER1
        MOV     WORD PTR [BP]+2AH,0
        JMP     SHORT TOERR
ERROR2: MOV     DX,OFFSET MSGER2
TOERR:  PUSH    DX
        PUSH    CS
        POP     DS
        MOV     DX,OFFSET ERRMSG
        CALL    CRT_STR
        POP     DX
        CALL    CRT_STR

        MOV     AX,1
        TEST    BYTE PTR [BP]-40H,20H
        JNE     BASERR
        TEST    BYTE PTR [BP]-40H,40H
        JNE     CALLEN_
        MOV     AH,4CH
        MOV     BYTE PTR CS:MVBUSY,0
        INT     21H

BASERR: MOV     DS,[BP]-3EH
        MOV     SS,CS:SSSTAK
        MOV     SP,CS:SPSTAK
        MOV     AL,17
        XOR     DI,DI
        MOV     BYTE PTR CS:MVBUSY,0
        INT     0C4H    ;  ＢＡＳＩＣエラー処理へ

;  ****  文字列表示  ****************************

CRT_STR PROC
        TEST    BYTE PTR [BP]-40H,20H
        JNE     CRTBAS
        PUSH    DS
        PUSH    DX
        MOV     AH,09H
        INT     21H
        POP     SI
        POP     DS
STRLOP: LODSB
        CMP     AL,'$'
        JNE     STRLOP
        MOV     DX,SI
        RET

CRTBAS: PUSH    DS
        PUSH    ES
        MOV     CS:SPSTAK2,SP
        MOV     SS,CS:SSSTAK
        MOV     SP,CS:SPSTAK
        MOV     SI,DX
        PUSH    SS
        POP     ES
        MOV     DI,OFFSET STRBUF
        CLD
        XOR     CX,CX
WKDLOP: LODSB
        CMP     AL,'$'
        JE      STREND
        STOSB
        INC     CX
        JMP     SHORT WKDLOP
STREND: PUSH    SI
        PUSH    SS
        POP     DS
        MOV     DI,60    ;  文字ライン出力
        INT     0C4H
        POP     DX
        MOV     AX,CS
        MOV     SS,AX
        MOV     SP,CS:SPSTAK2
        POP     ES
        POP     DS
        RET
CRT_STR ENDP

;  ****  システムワーク  ************************

MVCHCK: CMP     WORD PTR [BP]+2AH,MVFLG
        JE      MVOK    ;  システム設定チェック
        JMP     ERROR2
MVOK:   RET

BEGIN1: CALL    MVCHCK
        CMP     AX,[BP]
        JAE     TO1ER0
BEGIN2: CALL    DMUL32
        ADD     AX,[BP]+16H
        MOV     BX,AX
        MOV     DS,[BP]+14H
        MOV     ES,[BP]+14H
        RET
BEGIN3: CALL    BEGIN1
STAKDT: LEA     BP,[BP]-30H
        MOV     CX,20H
DATALP: MOV     AL,[BX]
        MOV     [BP],AL
        INC     BX
        INC     BP
        LOOP    DATALP
        LEA     BP,[BP]+10H
        SUB     BX,20H
        RET
DMUL32: SAL     AX,1
DMUL16: SAL     AX,1
DMUL8:  SAL     AX,1
        SAL     AX,1
        SAL     AX,1
        RET
MOD16:  SAR     AX,1
MOD08:  SAR     AX,1
        SAR     AX,1
        SAR     AX,1
        RET
VRAMTR: CALL    YDCHCK
VRAMT1: MOV     DX,W1DE80
        MUL     DX
        MOV     SI,AX
        RET
YDCHCK: CMP     AX,L1N400
        JAE     TO1ER0
        RET
TO1ER0: JMP     ERROR0

;  ***  ＭＶＳＥＴ  （ムーブ・セット）  ********

MVSET   PROC
        MOV     AL,0DH  ;  ＶＲＡＭ表示  設定
        OUT     0A2H,AL
        MOV     BX,[BP]-0AH
        CMP     BX,MVFLG
        JE      WKSET2
        CMP     BX,10H
        JA      MVSET0
        MOV     AX,[BP]-04H
        MOV     [BP]+14H,AX
        MOV     BX,[BP]-02H
        ADD     BX,AX
        MOV     [BP]+24H,BX
        JMP     MVSET4
```

```
MVSET0: TEST    BYTE PTR [BP]-40H,60H
        JNE     MVSET2
WKSET1: CMP     WORD PTR [BP]+2AH,MVFLG
        JNE     MEMORY
        MOV     ES,[BP]+14H
        MOV     AH,4AH
        INT     21H
        JB      MEMNEW
        MOV     AX,[BP]+14H
        JMP     SHORT MVSET1
MEMNEW: CMP     AX,08H
        JE      TO1ER1
MEMORY: MOV     AH,48H
        INT     21H
        JB      TO1ER1
        MOV     [BP]+14H,AX
MVSET1: ADD     AX,[BP]-0AH
        MOV     [BP]+24H,AX
        JMP     SHORT MVSET4

MVSET2: MOV     AX,[BP]-3AH
        MOV     [BP]+14H,AX
        ADD     BX,AX
        CMP     BX,[BP]-3CH
        JA      TO1ER1
        MOV     [BP]+24H,BX
        JMP     SHORT MVSET4
TO1ER1: JMP     ERROR1

WKSET2: CMP     WORD PTR [BP]+2AH,MVFLG
        JE      MVSET4
        MOV     AX,[BP]-04H
        CMP     AX,MVFLG
        JNE     MVSET3
        JMP     ERROR2
MVSET3: MOV     [BP]+14H,AX
        MOV     BX,[BP]-02H
        CMP     AX,BX
        JA      TO1ER1
        MOV     [BP]+24H,BX
MVSET4: MOV     AX,40H
        MOV     [BP]+16H,AX
        MOV     AX,0
        MOV     [BP]+10H,AX
        MOV     AX,OFFSET CODEND
        MOV     [BP]+12H,AX
SET2:   MOV     AX,[BP]-10H
        CMP     AX,MVFLG
        JNE     N_SET
        CALL    MVCHCK
        MOV     AX,[BP]

N_SET:  CMP     AX,400H
        JA      TO1ER0
        CMP     AX,0
        JE      TO1ER0
        MOV     BX,AX
        MOV     AX,[BP]-0EH
        CMP     AX,MVFLG
        JNE     S_SET
        CALL    MVCHCK
        MOV     AX,[BP]+2
S_SET:  CMP     AX,BX
        JA      TO1ER0
        MOV     CX,AX
        MOV     AX,[BP]-0CH
        CMP     AX,MVFLG
        JNE     RGBSP
        CALL    MVCHCK
        MOV     AX,[BP]+4
RGBSP:  CMP     AX,CX
        JA      TO1ER0
        MOV     [BP],BX
        MOV     [BP]+02H,CX
        MOV     [BP]+04H,AX
        MOV     AX,[BP]-08H
        CMP     AX,2
        JE      FASTST
        MOV     AL,0
FASTST: MOV     [BP]+29H,AL
        MOV     DS,[BP]+14H
        MOV     ES,[BP]+14H
        MOV     AX,MVFLG
        MOV     [BP]+2AH,AX
        CMP     AX,[BP]-10H
        JE      SETDA2
        MOV     BYTE PTR [BP]+28H,0
        MOV     BX,[BP]+16H
        MOV     SI,[BP]
A06:    MOV     DI,BX
        MOV     CL,4
        SHR     DI,CL
        ADD     DI,[BP]+14H
        CALL    MEOVCK
        XOR     AX,AX
        MOV     DI,BX
        MOV     CX,10H
        REP     STOSW
        MOV     AX,0108H
        MOV     [BX]+00H,AX
        MOV     AX,1
        MOV     [BX]+1CH,AX
        LEA     BX,[BX]+20H
        DEC     SI
        JNE     A06

SETDA2: MOV     AX,[BP]
        CALL    BEGIN2
        MOV     AX,BX
        CALL    ADDSEG
        MOV     DI,DS
        ADD     DI,AX
        MOV     [BP]+18H,DI
        MOV     BX,[BP]+16H
        XOR     SI,SI
A07:    CALL    MEOVCK
        MOV     [BX]+08H,DI
        MOV     AX,[BX]
        MUL     AH
        CALL    ADDSEG
        CMP     SI,[BP]+04H
        JGE     A08
        SHL     AX,1
        SHL     AX,1
A08:    MOV     [BX]+0CH,AX
        MUL     WORD PTR [BX]+1CH
        ADD     DI,AX
        LEA     BX,[BX]+20H
        INC     SI
        CMP     SI,[BP]
        JNE     A07
        MOV     [BP]+1AH,DI

        XOR     AX,AX
        MOV     [BP]+20H,AX
        MOV     BX,[BP]+16H
        XOR     SI,SI
A09:    CALL    MEOVCK
        MOV     [BX]+0AH,DI
        MOV     AX,[BX]
        INC     AH
        MUL     AH
        CMP     AX,[BP]+20H
        JLE     A11
        MOV     [BP]+20H,AX
A11:    CALL    ADDSEG
        CMP     SI,[BP]+02H
        JGE     A12
A115:   OR      BYTE PTR [BX]+03H,2
        CMP     SI,[BP]+4
        JGE     A119
        OR      BYTE PTR [BX]+03H,4
        SHL     AX,1
        SHL     AX,1
A119:   ADD     DI,AX
A12:    LEA     BX,[BX]+20H
        INC     SI
        CMP     SI,[BP]
        JNE     A09

        MOV     AX,[BP]+20H
        CALL    ADDSEG
        ADD     DI,AX
        MOV     [BP]+1CH,DI
A125:   TEST    BYTE PTR [BP]+29H,2
        JE      A14
        MOV     [BP]+1EH,DI
        MOV     AX,1800H
        JMP     SHORT A16
A14:    ADD     DI,AX
        SHL     AX,1
        SHL     AX,1
        ADD     DI,AX
        MOV     [BP]+1EH,DI
A16:    ADD     DI,AX
        MOV     [BP]+22H,DI
        CALL    MEOVCK
        XOR     SI,SI
        MOV     AX,[BP]+22H
        MOV     DI,[BP]+24H
        TEST    BYTE PTR [BP]+28H,10H
        JE      FREEP
        MOV     SI,[BP]+08H
        MOV     AX,[BP]+0AH
        MOV     DI,[BP]+0CH
FREEP:  MOV     [BP]+08H,SI
        MOV     [BP]+0AH,AX
        MOV     [BP]+0CH,DI
        SUB     DI,AX
        SHL     DI,1
        MOV     [BP]+26H,DI
        RET
MVSET   ENDP

ADDSEG: ADD     AX,0FH
        MOV     CL,4
        SHR     AX,CL
        RET

MEOVCK: CMP     DI,[BP]+24H
        JAE     TO2ER1
        CMP     DI,[BP]+14H
        JBE     TO2ER1
        RET
TO2ER1: JMP     ERROR1

; *** GSIZE (グラフィック・サイズ) **
```

```
GSIZE   PROC
        MOV     AX,[BP]-10H
        CALL    BEGIN3
        MOV     AX,[BP]-4
        CMP     AX,0
        JLE     TO35E0
        CMP     AX,W1DE80
        JG      TO35E0
        MOV     [BX]+01H,AL
        MOV     AX,[BP]-2
        CMP     AX,0
        JLE     TO35E0
        CMP     AX,255
        JG      TO35E0
        MOV     [BX]+00H,AL
        MOV     AX,[BP]-0EH
        CMP     AX,0
        JE      KOMA1
        CMP     AX,32
        JBE     TOSZST
KOMA1:  MOV     AX,1
TOSZST: MOV     [BX]+1CH,AX
        MOV     WORD PTR [BX]+1EH,0
        MOV     AX,[BP]+1AH
        MOV     [BP]+4AH,AX
        MOV     AX,[BX]+28H
        MOV     [BP]+48H,AX
        PUSH    BX
        PUSH    DS
        CALL    SETDA2
        POP     DS
        POP     BX
        MOV     AX,[BP]-10H
        INC     AX
        SUB     AX,[BP]
        JGE     TODASE
        MOV     AX,[BX]+28H
        MOV     BX,[BP]+48H
        MOV     DX,[BP]+1AH
        CMP     AX,BX
        JE      TODASE
        JG      SIFTDW
SIFTUP: MOV     ES,AX
        MOV     DS,BX
        XOR     SI,SI
        XOR     DI,DI
        MOV     CX,8
        REP     MOVSW
        INC     AX
        INC     BX
        CMP     AX,DX
        JB      SIFTUP
        RET
SIFTDW: MOV     AX,[BP]+4AH
DWNLP:  MOV     DS,AX
        MOV     ES,DX
        XOR     SI,SI
        XOR     DI,DI
        MOV     CX,8
        REP     MOVSW
        DEC     AX
        DEC     DX
        CMP     AX,BX
        JAE     DWNLP
TODASE: RET
GSIZE   ENDP
TO35E0: JMP     ERROR0

; ***  GDSET  (ｸﾞﾗﾌｨｯｸ･ﾃﾞｰﾀｰ･ｾｯﾄ)  *****

GDSET   PROC
        MOV     AX,[BP]-0CH
        CALL    BEGIN3
        MOV     AX,[BP]-08H
        CALL    PTNAD1
SAPST:  MOV     AX,[BX]
        MOV     CL,AL
        MUL     AH
        MOV     SI,AX
        MOV     DL,[BX]+03H
        MOV     AX,[BP]-0EH
        CMP     AX,MVFLG
        JNE     DT1SET
        MOV     AX,[BP]-10H
        CMP     AX,MVFLG
        JE      FRAME
        MOV     ES,[BP]-22H
        XOR     DI,DI
        JMP     SHORT GDSSTA

DT1SET: CMP     AX,SI
        JAE     TO35E0
SALP:   DIV     CL
        MOV     CL,AL
        XOR     CH,CH
        MOV     AL,[BX]+1
        MUL     AH
        MOV     DI,AX
        ADD     DI,CX
        MOV     DS,[BP]-22H
        MOV     AX,[BP]-10H
GDSSTA: CMP     AX,255
        JA      TO35E0
        TEST    DL,4
        JE      GDSET2
        MOV     AX,[BP]-0AH
        CMP     AX,MVFLG
        JNE     RGBDCK
        MOV     AX,1FH
RGBDCK: CMP     AX,1FH
        JA      TO35E0
        MOV     DL,AL
        TEST    DL,1
        CALL    GDSET1
        TEST    DL,2
        CALL    GDSET1
        TEST    DL,4
        CALL    GDSET1
GDFRM1: TEST    DL,10H
GDSET1: JE      GDSEN
GDSET2: MOV     AL,[BP]-10H
        CMP     WORD PTR [BP]-0EH,MVFLG
        JE      DALLST
        MOV     [DI],AL
GDSEN:  ADD     DI,SI
        RET

DALLST: MOV     CX,SI
        REP     STOSB
        RET
TO38E0: JMP     ERROR0

FRAME:  TEST    DL,4    ;  外形パタンの設定
        JE      TO38E0
        MOV     AX,SI
        MOV     CX,AX
        MOV     ES,[BP]-22H
        MOV     DS,[BP]-22H
        XOR     SI,SI
        MOV     BX,AX
        PUSH    BP
        MOV     BP,BX
        ADD     BP,AX
        MOV     DI,BP
        ADD     DI,AX
RGBFLP: LODSB
        OR      AL,[BX]
        OR      AL,DS:[BP]
        STOSB
        INC     BX
        INC     BP
        LOOP    RGBFLP
        POP     BP
        RET
GDSET   ENDP
TO4ERO: JMP     ERROR0

; ***  GDISP  (グラフィック・ディスプ)

GDISP   PROC
        MOV     AX,[BP]-10H
        CALL    BEGIN1
        MOV     [BP]+3EH,BX
        CALL    DPCLCK
        MOV     AX,[BP]-0EH
        CMP     AX,MVFLG
        JNE     C01
        MOV     AL,[BX]+02H
C01:    TEST    AL,80H
        JE      C02
        OR      BYTE PTR [BX]+02H,80H
        RET

C02:    MOV     [BP]+31H,AL
        CMP     AL,20H
        JL      C09

C06:    CMP     AL,30H
        JGE     C07
        OR      BYTE PTR [BX]+03H,20H
        JMP     SHORT C09

C07:    CMP     AL,50H
        JL      C09
        CMP     AL,60H
        JNE     C08
        MOV     AL,[BX]+02H
C08:    AND     AL,0FH
        MOV     [BX]+02H,AL
        RET

C09:    AND     AL,0FH
        MOV     [BX]+02H,AL
        MOV     AX,[BP]-0CH
        CALL    PTNAD1
        CALL    XYSET
        CALL    DPWK1   ;  パタン表示ワークへ
        CALL    STAKDT
        CALL    CLWKST
        TEST    BYTE PTR [BP]-2DH,2
        JNE     SPSEDP
        CALL    SPOVR0
        CMP     WORD PTR [BP]+02H,0
        JE      TOGDP

SPSEDP: MOV     AX,[BP]+02H
        CALL    BEGIN2
```

```
                TEST    BYTE PTR [BP]+31H,10H
                JNE     SPREV1
        SPNOR1: LEA     BX,[BX]-20H
                CALL    SPOVR0
                CMP     BX,[BP]+16H
                JNE     SPNOR1
                JMP     SHORT TOGDP

        SPREV1: MOV     [BP]+3CH,BX
                MOV     BX,[BP]+16H
        SPRVL1: CALL    SPOVR0
                LEA     BX,[BX]+20H
                CMP     BX,[BP]+3CH
                JNE     SPRVL1
        TOGDP:  MOV     BX,[BP]+3EH
                CALL    GDPSUB  ;  消去・表示へ
                RET
        GDISP   ENDP

        GDPSUB: CLI
                MOV     DS,[BP]+14H
                TEST    BYTE PTR [BX]+03H,20H
                JNE     C27
                MOV     DX,[BX]+10H
                CMP     DL,0
                JLE     C27
                        ;  以前表示したものを消去
                MOV     AX,VRWSEG
                MOV     ES,AX
                MOV     DI,[BX]+12H
                MOV     SI,DI
                TEST    BYTE PTR [BP]+29H,2
                JNE     C26
                XOR     SI,SI
        C26:    MOV     DS,[BP]+1CH
                CALL    WKTPR0
                MOV     DS,[BP]+14H
        C27:    TEST    BYTE PTR [BX]+03H,8
                JNE     GDPEND
                CMP     BYTE PTR [BX]+02H,0
                JL      GDPEND
                MOV     DX,[BX]+16H
                CMP     DL,0
                JLE     GDPEND
                MOV     AX,VRWSEG    ; 新しく画面表示
                MOV     ES,AX
                MOV     DI,[BX]+18H
                MOV     SI,DI
                TEST    BYTE PTR [BP]+29H,2
                JNE     C28
                XOR     SI,SI
        C28:    MOV     DS,[BP]+1EH
                CALL    WKTPR0
        GDPEND: RET
        SPRER2: JMP     ERROR2

        ; ***  S P R I T E  (スプライト)  **********

        SPRITE  PROC
                TEST    BYTE PTR [BP]+29H,2
                JE      SPRER2
                MOV     AX,[BP]-10H
                CMP     AX,MVFLG
                JNE     C31
                XOR     AX,AX
        C31:    CALL    PADSET
                MOV     AX,[BP]-0EH
                CMP     AX,MVFLG
                JNE     C32
                MOV     AX,[BP]+00H
        C32:    CMP     AX,[BP]+00H
                JA      SPRER2
                MOV     [BP]+70H,AX
                MOV     BX,[BP]-0CH
                CMP     BX,MVFLG
                JNE     C33
                MOV     BX,1
        C33:    CMP     BX,[BP]+26H
                JA      SPRER2
                MOV     [BP]+74H,BX
                MOV     AX,[BP]-0AH
                CMP     AX,MVFLG
                JNE     C35
                XOR     AX,AX
        C35:    MOV     [BP]+7AH,AX

        SPRLP1: MOV     AX,[BP]+70H
                MOV     [BP]+72H,AX
        SPRLP2: LODSW
                MOV     [BP]-10H,AX
                LODSW
                CMP     AX,20H
                JB      C386
                CMP     AX,MVFLG
                JNE     C386
                MOV     AX,0FH
        C386:   AND     AX,0FH
                MOV     [BP]-0EH,AX
                LODSW
                MOV     [BP]-4,AX
                LODSW
                MOV     [BP]-2,AX
                PUSH    SI
                PUSH    DS
```

```
        GDISP1: MOV     AX,[BP]-10H
                CALL    BEGIN1
                CALL    DPCLCK
                MOV     AX,[BP]-0EH
                MOV     [BX]+02H,AL
                CALL    PTNAD2
                CALL    XYSET
                CALL    DPWK1
                CALL    CLWKST
                TEST    BYTE PTR [BX]+03H,2
                JNE     C39
                CALL    STAKDT
                CALL    SPOVR0
        C39:    POP     DS
                POP     SI
                DEC     WORD PTR [BP]+72H
                JNE     SPRLP2

        C40:    PUSH    SI
                PUSH    DS
        GDISP2: MOV     AX,[BP]+02H  ;  同時重ね合わせ
                CALL    BEGIN2
        SPNOR2: LEA     BX,[BX]-20H
                CMP     BYTE PTR [BX]+02H,0
                JL      TOSPNO
                CALL    SPOVR2
        TOSPNO: CMP     BX,[BP]+16H
                JNE     SPNOR2

                MOV     AX,[BP]+00H  ;  パタン同時表示
                CALL    BEGIN2
        DPDPLP: LEA     BX,[BX]-20H
                MOV     DS,[BP]+14H
                TEST    BYTE PTR [BX]+03H,10H
                JNE     NEXGDP
                OR      BYTE PTR [BX]+03H,10H
                CALL    GDPSUB
        NEXGDP: CMP     BX,[BP]+16H
                JNE     DPDPLP
                POP     DS
                POP     SI
                MOV     CX,[BP]+7AH
                CMP     CX,0
                JE      C41
        SPRLP3: XOR     AX,AX
        SPRLP4: DEC     AL
                JNE     SPRLP4
                LOOP    SPRLP3
        C41:    DEC     WORD PTR [BP]+74H
                JNE     SPRLP1
                RET
        SPRITE  ENDP
        SPRER0: JMP     ERROR0


        DPCLCK: AND     BYTE PTR [BX]+03H,0DFH
                TEST    BYTE PTR [BX]+02H,80H
                JE      DPCLEN
                OR      BYTE PTR [BX]+03H,20H
        DPCLEN: RET

        PTNAD1: CMP     AX,[BX]+1CH
                JB      PTNAD3
        PTNAD2: MOV     AX,[BX]+1EH
        PTNAD3: MOV     DX,AX
                INC     DX
                CMP     DX,[BX]+1CH
                JB      PTNAD4
                XOR     DX,DX
        PTNAD4: MOV     [BX]+1EH,DX
                MUL     WORD PTR [BX]+0CH
                ADD     AX,[BX]+08H
                MOV     [BX]+0EH,AX
                MOV     [BP]-22H,AX
                RET

        XYSET:  AND     BYTE PTR [BX]+03H,0EFH
                XOR     AX,AX
                CMP     WORD PTR [BX]+04H,0
                JL      C10
                MOV     AX,[BX]+04H
                SAR     AX,1
                SAR     AX,1
                SAR     AX,1
        C10:    MOV     [BP]+35H,AL
                XOR     AX,AX
                CMP     WORD PTR [BX]+06H,0
                JL      C11
                MOV     AX,[BX]+06H
        C11:    MOV     [BP]+36H,AX
                MOV     AX,[BX]+16H
                MUL     AH
                CMP     AX,0
                JNE     C12
                OR      BYTE PTR [BX]+03H,20H
        C12:    LEA     SI,[BX]+16H
                LEA     DI,[BX]+10H
                MOVSW
                MOVSW
                MOVSW
                XOR     DX,DX
                MOV     [BX]+16H,DX
                MOV     [BX]+18H,DX
                MOV     [BX]+1AH,DX
```

```
        MOV     AX,[BP]-04H
        MOV     [BX]+04H,AX
        MOV     DL,[BX]+01H
        MOV     CL,7
        AND     CL,AL
        JE      C14
        INC     DL
C14:    MOV     [BP]+38H,CL
        SAR     AX,1
        SAR     AX,1
        SAR     AX,1
        CMP     AX,WIDE80
        JGE     C17
        CMP     AX,0
        JGE     C15
        MOV     BYTE PTR [BP]+39H,0
        ADD     DX,AX
        JLE     C17
        SUB     [BX]+1AH,AX
        MOV     [BX]+16H,DL
        JMP     SHORT C17

C15:    MOV     [BX]+18H,AL
        MOV     [BP]+39H,AL
        NEG     AL
        ADD     AL,WIDE80
        CMP     AL,DL
        JLE     C16
        MOV     AL,DL
C16:    MOV     [BX]+16H,AL
C17:    MOV     AX,[BP]-02H
        MOV     [BX]+06H,AX
        CMP     AX,LIN400
        JGE     C195
        MOV     DL,[BX]
        CMP     AX,0
        JGE     C21
        MOV     WORD PTR [BP]+3AH,0
        ADD     DX,AX
        JG      C20
C195:   MOV     BYTE PTR [BX]+16H,0
        RET
C20:    MOV     [BX]+17H,DL
        NEG     AX
        MOV     AH,[BX]+01H
        INC     AH
        MUL     AH
        ADD     [BX]+1AH,AX
        RET
C21:    MOV     [BP]+3AH,AX
        CALL    VRAMTR
        ADD     [BX]+18H,AX
        MOV     AX,LIN400
        XOR     DH,DH
        MOV     DL,[BX]
        SUB     AX,[BP]+3AH
        CMP     AX,DX
        JLE     C22
        MOV     AX,DX
C22:    MOV     [BX]+17H,AL

C23:    AND     BYTE PTR [BX]+03H,0F7H
        TEST    BYTE PTR [BX]+03H,20H
        JNE     TOC239
        CMP     BYTE PTR [BX]+10H,0
        JLE     TOC239
        CMP     BYTE PTR [BX]+16H,0
        JLE     TOC239
        MOV     AX,[BX]+10H
        MOV     [BP]+40H,AX
        MOV     SI,[BX]+12H
        MOV     AX,[BP]+34H
        MOV     [BP]+44H,AX
        MOV     DI,[BP]+36H
        XOR     CX,CX
        MOV     DL,[BP]+39H
        SUB     DL,[BP]+45H
        JL      C231
        CMP     DL,[BX]+10H
        JL      C232
TOC239: RET
C231:   MOV     DH,[BX]+16H
        ADD     DH,DL
        JLE     TOC239
C232:   MOV     AX,[BP]+3AH
        SUB     AX,DI
        JL      C233
        CMP     AL,[BX]+11H
        JA      TOC239
        MOV     CL,[BX]+17H
        ADD     AX,CX
        CMP     AX,100H
        JGE     TOC239
        CMP     AL,[BP]+41H
        JB      C2325
        MOV     [BP]+41H,AL
C2325:  CMP     DL,0
        JL      C234
        ADD     DL,[BX]+16H
        CMP     DL,[BP]+40H
        JLE     C238
        MOV     [BP]+40H,DL
        JMP     SHORT C238
```

```
C233:   NEG     AX
        CMP     AL,[BX]+17H
        JA      TOC239
        MOV     CL,[BP]+41H
        ADD     AX,CX
        CMP     AX,100H
        JGE     C239
        MOV     [BP]+41H,AL
        MOV     DI,[BP]+3AH
        MOV     SI,[BX]+18H
        XOR     DH,DH
        CMP     DL,0
        JL      C235
        SUB     SI,DX
        ADD     DL,[BX]+16H
        CMP     DL,[BP]+40H
        JLE     C238
        MOV     [BP]+40H,DL
        JMP     SHORT C238

C234:   MOV     DH,0FFH
        ADD     SI,DX
C235:   SUB     [BP]+40H,DL
        ADD     [BP]+45H,DL

C238:   MOV     AX,[BP]+40H
        MUL     AH
        MOV     CX,AX
        MOV     AX,[BX]+00H
        INC     AH
        MUL     AH
        SHL     AX,1
        CMP     AX,CX
        JB      C239
        OR      BYTE PTR [BX]+03H,8
        MOV     AX,[BP]+40H
        MOV     [BX]+10H,AX
        MOV     [BX]+12H,SI
        MOV     AX,[BP]+44H
        MOV     [BP]+34H,AX
        MOV     [BP]+36H,DI
C239:   RET

DPWK1:  PUSH    DS      ;  パタン表示作業1
        PUSH    BX
        XOR     SI,SI
        XOR     DI,DI
        MOV     AX,[BX]
        MOV     CH,[BX]+03H
        MOV     CL,[BX]+04H
        AND     CL,7
        MOV     ES,[BX]+0AH
        MOV     DS,[BX]+0EH
        MOV.    BX,AX
        CALL    DPWK2
        TEST    CH,4
        JE      DPWKEN
        CALL    DPWK2
        CALL    DPWK2
        CALL    DPWK2
DPWKEN: POP     BX
        POP     DS
        RET
DPWK2:  MOV     DH,BL   ;  パタン表示作業2
DPLOP1: MOV     DL,BH
        XOR     AH,AH
        MOV     AL,[SI]
        JMP     SHORT DPFG1
DPLOP2: MOV     AX,[SI]
        XCHG    AH,AL
        INC     SI
DPFG1:  SHR     AX,CL
        STOSB
        DEC     DL
        JNE     DPLOP2
        MOV     AH,[SI]
        XOR     AL,AL
        SHR     AX,CL
        STOSB
        INC     SI
        DEC     DH
        JNE     DPLOP1
        RET

WKTFR0: MOV     AX,WIDE80
        SUB     AL,DL
        TEST    BYTE PTR [BP]+29H,2
        JNE     WKTFR4

WKTFR1: CALL    WKTFS0
        CALL    WKTFS0
WKTFS0: PUSH    DI
        PUSH    DX
        XOR     CH,CH
GDTFL0: MOV     CL,DL
        REP     MOVSB
        ADD     DI,AX
        DEC     DH
        JNE     GDTFL0
        POP     DX
        POP     DI
        MOV     CX,ES
        ADD     CH,8
        MOV     ES,CX
```

```
        RET

WKTFR4: CALL    WKTFS4
        CALL    WKTFS4
WKTFS4: PUSH    SI
        PUSH    DI
        PUSH    DX
        XOR     CH,CH
GDTFL4: MOV     CL,DL
        REP     MOVSB
        ADD     SI,AX
        ADD     DI,AX
        DEC     DH
        JNE     GDTFL4
        POP     DX
        POP     DI
        POP     SI
        MOV     CX,DS
        ADD     CH,8
        MOV     DS,CX
        MOV     CX,ES
        ADD     CH,8
        MOV     ES,CX
        RET

WKTFR3: CALL    WKTFS3
        CALL    WKTFS3
WKTFS3: TEST    BYTE PTR [BP]+29H,2
        JNE     WKTFS4
        PUSH    SI
        PUSH    DX
        XOR     CH,CH
GDTFL3: MOV     CL,DL
        REP     MOVSB
        ADD     SI,AX
        DEC     DH
        JNE     GDTFL3
        POP     DX
        POP     SI
        MOV     CX,DS
        ADD     CH,8
        MOV     DS,CX
        RET

CLWKST: MOV     DS,[BP]+14H
        CMP     BYTE PTR [BX]+02H,0
        JL      C63
        TEST    BYTE PTR [BX]+03H,8
        JNE     C63
        MOV     DX,[BX]+16H
        CMP     DL,0
        JLE     C63
        MOV     SI,[BX]+18H
        MOV     DI,SI
        TEST    BYTE PTR [BP]+29H,2
        JNE     C61
        XOR     DI,DI
C61:    MOV     ES,[BP]+1EH
        TEST    BYTE PTR [BP]+28H,0CH
        JE      C62
        CALL    BGTOWK
        JMP     SHORT C63
C62:    CALL    TOCLWK
C63:    MOV     DS,[BP]+14H
        TEST    BYTE PTR [BX]+03H,20H
        JNE     WKCLEN
        MOV     DX,[BX]+10H
        CMP     DL,0
        JLE     WKCLEN
        MOV     SI,[BX]+12H
        MOV     DI,SI
        TEST    BYTE PTR [BP]+29H,2
        JNE     C64
        XOR     DI,DI
C64:    MOV     ES,[BP]+1CH
        TEST    BYTE PTR [BP]+28H,0CH
        JNE     BGTOWK
        JMP     TOCLWK
WKCLEN: RET

BGTOWK: MOV     AL,1
        OUT     0A6H,AL
        MOV     AX,VRMSEG
        MOV     DS,AX
        MOV     AX,WIDE80
        SUB     AL,DL
        CALL    WKTFR3
        MOV     AL,0
        OUT     0A6H,AL
        RET

TOCLWK: TEST    BYTE PTR [BP]+29H,2
        JNE     CLWKFG
        MOV     AX,DX
        MUL     AH
        MOV     CX,AX
        ADD     CX,AX
        ADD     CX,AX
        INC     CX
        SHR     CX,1
        XOR     AX,AX
        REP     STOSW
        RET
CLWKFG: PUSH    BX
        MOV     BX,WIDE80
        SUB     BL,DL
        XOR     CX,CX
        CALL    CLWK
        CALL    CLWK
        CALL    CLWK
        POP     BX
        RET
CLWK:   PUSH    DI
        PUSH    DX
        XOR     AL,AL
CLWKL2: MOV     CL,DL
        REP     STOSB
        ADD     DI,BX
        DEC     DH
        JNE     CLWKL2
        POP     DX
        POP     DI
        MOV     AX,ES
        ADD     AH,8
        MOV     ES,AX
        RET

                        ; 重ね合わせ
SPOVR0: MOV     DS,[BP]+14H
        CMP     BYTE PTR [BX]+02H,0
        JL      SPOVE1
        TEST    BYTE PTR [BP]-2DH,20H
        JNE     DPSPOV
        MOV     CX,[BP]-20H     ; 消去チェック
        MOV     DL,[BP]+35H
        MOV     AX,[BP]+36H
        MOV     DI,[BP]-1EH
        MOV     ES,[BP]+1CH
        CALL    SPOVR1
DPSPOV: CMP     BYTE PTR [BP]-2EH,0
        JL      SPOVE1
        TEST    BYTE PTR [BP]-2DH,8
        JNE     SPOVE1
        MOV     CX,[BP]-1AH     ; 表示チェック
        MOV     DL,[BP]+39H
        MOV     AX,[BP]+3AH
        MOV     DI,[BP]-18H
        MOV     ES,[BP]+1EH
        JMP     SHORT SPOVR1
SPOVE1: RET

SPOVR1: CMP     CL,0    ; 重ね合わせチェック
        JLE     TOPOP
        CMP     BYTE PTR [BX]+16H,0
        JG      C80
TOPOP:  RET
C80:    MOV     [BP]+52H,AX
        MOV     AX,WIDE80
        TEST    BYTE PTR [BP]+29H,2
        JNE     C81
        XOR     DI,DI
        MOV     AL,CL
C81:    MOV     [BP]+54H,AX
        MOV     SI,[BX]+1AH
        MOV     AL,CH
        MOV     [BP]+50H,AX
        MOV     [BP]+30H,CL
        XOR     AX,AX
        CMP     WORD PTR [BX]+04H,0
        JL      C815
        MOV     AX,[BX]+04H
        SAR     AX,1
        SAR     AX,1
        SAR     AX,1
C815:   SUB     DL,AL
        JG      C83
        MOV     AL,DL
        ADD     DL,[BP]+30H
        JLE     DMPOP
        CMP     DL,[BX]+16H
        JLE     C82
        MOV     DL,[BX]+16H
C82:    NEG     AL
        XOR     AH,AH
        ADD     DI,AX
        JMP     SHORT C84
C83:    XOR     DH,DH
        ADD     SI,DX
        SUB     DL,[BX]+16H
        JGE     DMPOP
        NEG     DL
        CMP     DL,[BP]+30H
        JBE     C84
        MOV     DL,[BP]+30H
C84:    MOV     [BP]+40H,DL
        MOV     AX,[BP]+54H
        SUB     AL,DL
        MOV     [BP]+44H,AX
        MOV     AL,[BX]+01H
        INC     AL
        SUB     AL,DL
        MOV     [BP]+46H,AX
        XOR     CH,CH
        MOV     CL,[BX]+17H
        MOV     AX,[BP]+52H
        CMP     WORD PTR [BX]+06H,0
        JL      C842
        SUB     AX,[BX]+06H
```

```
C842:   MOV     [BP]+52H,AX
        CMP     AX,0
        JGE     C86
        ADD     AX,[BP]+50H
        JG      C845
DMPOP:  RET             ;  重なった部分なし
C845:   CMP     AX,CX
        JLE     C85
        MOV     AX,CX
C85:    MOV     [BP]+41H,AX
        MOV     AX,[BP]+52H
        NEG     AX
        MUL     WORD PTR [BP]+54H
        ADD     DI,AX
        JMP     SHORT C88

C86:    SUB     AX,CX
        JGE     DMPOP
        NEG     AX
        CMP     AX,[BP]+50H
        JLE     C87
        MOV     AX,[BP]+50H
C87:    MOV     [BP]+41H,AL
        MOV     AL,[BX]+01H
        INC     AL
        MUL     BYTE PTR [BP]+52H
        ADD     SI,AX
C88:    MOV     AL,[BP]+50H
        SUB     AL,[BP]+41H
        MUL     BYTE PTR [BP]+54H
        MOV     [BP]+4AH,AX
        JMP     SHORT SPOVR3

SPOVR2: MOV     AX,[BX]+16H
        CMP     AL,0
        JLE     DMPOP
        MOV     [BP]+40H,AX
        MOV     DI,[BX]+18H
        MOV     SI,[BX]+1AH
        MOV     AX,WIDE80
        SUB     AL,[BX]+16H
        MOV     [BP]+44H,AX
        MOV     AL,[BX]+01H
        INC     AL
        SUB     AL,[BX]+16H
        MOV     [BP]+46H,AX
        MOV     ES,[BP]+1EH
SPOVR3: PUSH    DS
        PUSH    BX
        MOV     AX,[BX]
        INC     AH
        MUL     AH
        MOV     [BP]+48H,AX
        MOV     CX,[BX]+0AH
        MOV     DX,[BP]+40H
        MOV     AX,[BX]+02H
        MOV     BX,[BP]+44H
        MOV     DS,CX
        XOR     CX,CX
        TEST    AH,4
        JNE     SPRGB
                ;  モノカラーパタンの重ね作業
MONOSP: TEST    AL,1
        CALL    SPST1
        TEST    AL,2
        CALL    SPST1
        TEST    AL,4
        CALL    SPST1
        POP     BX
        POP     DS
        RET
SPST1:  PUSH    AX
        PUSH    DI
        PUSH    SI
        MOV     AH,DH
        JNE     SPSLP3
SPSLP1: MOV     CL,DL
SPSLP2: LODSB
        NOT     AL
        AND     AL,ES:[DI]
        STOSB
        LOOP    SPSLP2
        ADD     SI,[BP]+46H
        ADD     DI,BX
        DEC     AH
        JNE     SPSLP1
        JMP     SHORT SPST2

SPSLP3: MOV     CL,DL
SPSLP4: LODSB
        OR      AL,ES:[DI]
        STOSB
        LOOP    SPSLP4
        ADD     SI,[BP]+46H
        ADD     DI,BX
        DEC     AH
        JNE     SPSLP3

SPST2:  POP     SI
        TEST    BYTE PTR [BP]+29H,2
        JNE     C92
        ADD     DI,[BP]+4AH
        POP     AX
        POP     AX
        RET
C92:    POP     DI
        MOV     AX,ES
        ADD     AH,8
        MOV     ES,AX
        POP     AX
        RET
                ;  ＲＧＢカラーパタンの重ね作業
SPRGB:  MOV     AX,[BP]+48H
        MOV     BX,SI
        ADD     SI,AX
        ADD     SI,AX
        ADD     SI,AX
        CALL    SPST4
        CALL    SPST4
        CALL    SPST4
        POP     BX
        POP     DS
        RET
SPST4:  PUSH    DI
        PUSH    SI
        PUSH    BX
        MOV     AH,DH
SPSLP6: MOV     CL,DL
SPSLP7: LODSB
        NOT     AL
        AND     AL,ES:[DI]
        OR      AL,[BX]
        INC     BX
        STOSB
        LOOP    SPSLP7
        ADD     SI,[BP]+46H
        ADD     BX,[BP]+46H
        ADD     DI,[BP]+44H
        DEC     AH
        JNE     SPSLP6
SPST5:  POP     BX
        POP     SI
        ADD     BX,[BP]+48H
        TEST    BYTE PTR [BP]+29H,2
        JNE     SPSFG7
        ADD     DI,[BP]+4AH
        POP     AX
        RET
SPSFG7: POP     DI
        MOV     AX,ES
        ADD     AH,8
        MOV     ES,AX
        RET

;  ***  ＧＰＯＫＥ  （グラフィック・ポーク）  **

GPOKE   PROC
        MOV     AX,[BP]-0CH
        CALL    PADSET
        MOV     AX,[BP]-10H
        CALL    GPOSUB
        MOV     AX,[BP]-0EH
        CALL    GPOSUB
        MOV     AX,[BP]-04H
        CALL    GPOSUB
        MOV     AX,[BP]-02H
GPOSUB: CMP     AX,MVFLG
        JE      GPOEND
        MOV     [SI],AX
GPOEND: INC     SI
        INC     SI
        RET
GPOKE   ENDP

TO6ERO: JMP     ERROR0
PADSET: CMP     AX,[BP]+26H
        JAE     TO6ERO
        MOV     SI,[BP]+08H
        TEST    AL,1
        JE      PADSE1
        LEA     SI,[SI]+8
PADSE1: SHR     AX,1
        ADD     AX,[BP]+0AH
        MOV     DS,AX
        MOV     ES,AX
        MOV     DI,SI
        RET

;  ***  ＧＰＥＥＫ（グラフィック・ピーク）  ****

GPEEK   PROC
        MOV     AX,[BP]-4
        MOV     [BP]+60H,AX
        MOV     AX,[BP]-2
        MOV     [BP]+62H,AX
        MOV     AX,[BP]-10H
        CALL    PADSET
        MOV     BX,[BP]-0EH
        MOV     AX,[BP]-10H
        ADD     AX,BX
        CMP     AX,[BP]+26H
        JAE     TO6ERO
        MOV     DX,[BP]-0CH
        CMP     DX,MVFLG
        JNE     BACKCK
        XOR     DX,DX
BACKCK: MOV     CX,[BP]-0AH
        CMP     CX,MVFLG
```

```
        JNE     FBCHCK
        XOR     CX,CX
        JMP     SHORT FBSET
FBCHCK: CMP     CL,0
        JE      FBSET
        DEC     AX
        CALL    PADSET
FBSET:  MOV     [BP]+64H,CL
        CLI
DPLOOP: LODSW
        MOV     [BP]-10H,AX
        LODSW
        MOV     [BP]-0EH,AX
        LODSW
        CMP     AX,MVFLG
        JE      XPOS
        ADD     AX,[BP]+60H
XPOS:   MOV     [BP]-4,AX
        LODSW
        ADD     AX,[BP]+62H
        MOV     [BP]-2,AX
        PUSH    DS
        PUSH    ES
        PUSH    BP
        PUSH    SI
        PUSH    DX
        PUSH    BX
        MOV     DS,[BP]+14H
        MOV     ES,[BP]+14H
        CALL    GDISP   ;  パタン表示へ
        POP     BX
        POP     DX
        POP     SI
        POP     BP
        POP     ES
        POP     DS
        MOV     CX,DX
        CMP     CX,0
        JE      NEXTDP
DMTIME: LOOP    DMTIME
NEXTDP: MOV     AX,DS
        CMP     BYTE PTR [BP]+64H,0
        JE      FORD
        CMP     SI,8
        JNE     FBCKST
        DEC     AX
        JMP     SHORT NEXTCK
FORD:   CMP     SI,8
        JLE     NEXTCK
        INC     AX
FBCKST: XOR     SI,SI
NEXTCK: MOV     DS,AX
        MOV     ES,AX
        DEC     BX
        JNE     DPLOOP
        STI
        RET
GPEEK   ENDP

;  ***  GDGET  (ｸﾞﾗﾌｨｯｸ･ﾃﾞｰﾀｰ･ｹﾞｯﾄ)  ****

GDGET   PROC
        MOV     AX,[BP]-10H
        CALL    BEGIN3
        MOV     AX,[BP]-0CH
        CALL    PTNAD1
        MOV     AX,[BP]-0EH
        CMP     AX,1FH
        JLE     GETCOL
        MOV     AX,1FH
GETCOL: MOV     [BP]+31H,AL
        XOR     AX,AX
GDGEDM: DEC     AX      ;ダミータイム
        JNE     GDGEDM
        MOV     [BP]+34H,AX
        MOV     AX,[BP]-4
        CMP     AX,WIDE80*8     ;  640
        JAE     TOIER0
        MOV     DL,AL
        CALL    MOD08
        MOV     AH,AL
        MOV     AL,DL
        AND     AL,7
        MOV     [BP]+3AH,AX
        MOV     AX,[BP]-2
        CALL    VRAMTR
        XOR     DH,DH
        MOV     DL,[BP]+3BH
        ADD     SI,DX
        MOV     ES,[BP]+1CH
        CALL    VRMGET
        MOV     AX,DI
        DEC     DI
        MOV     DL,[BP]+3AH
        CMP     DL,0
        JE      VDTFR
        MOV     DS,[BP]+1CH
GDLLP1: MOV     SI,DI
        MOV     CX,AX
        CLC
GDLLP2: RCL     BYTE PTR [SI],1
        DEC     SI
        LOOP    GDLLP2
        DEC     DL
        JNE     GDLLP1
VDTFR:  MOV     DS,[BP]+1CH
        MOV     ES,[BP]-22H
        MOV     AX,[BP]-30H
        MUL     AH
        MOV     [BP]+36H,AX
        TEST    BYTE PTR [BP]-2DH,4
        JNE     RGBDS
        XOR     DI,DI
        MOV     [BP]+34H,DI
GDOR:   XOR     SI,SI
        MOV     CX,[BP]+36H
        XOR     AL,AL
PDWKCL: STOSB
        LOOP    PDWKCL
        MOV     DI,[BP]+34H
        MOV     DH,[BP]+31H
        TEST    DH,1
        CALL    VDTFS2
        TEST    DH,2
        CALL    VDTFS2
        TEST    DH,4
VDTFS2: MOV     DI,[BP]+34H
        MOV     DL,[BP]-30H
        JE      VDNOST
MONLP2: MOV     CL,[BP]-2FH
MONLP3: LODSB
        OR      AL,ES:[DI]
        STOSB
        LOOP    MONLP3
        INC     SI
        DEC     DL
        JNE     MONLP2
        RET
VDNOST: ADD     SI,[BP]+36H
        ADD     DI,[BP]+36H
        MOV     CL,DL
        ADD     SI,CX
        RET

RGBDS:  XOR     SI,SI
        XOR     DI,DI
        MOV     DH,[BP]+31H
        TEST    DH,1
        CALL    VDTFS1
        TEST    DH,2
        CALL    VDTFS1
        TEST    DH,4
        CALL    VDTFS1
        TEST    DH,10H
        JE      RGBDEN
        MOV     [BP]+34H,DI
        MOV     AL,0FH
        MOV     [BP]+31H,AL
        JMP     SHORT GDOR
RGBDEN: RET

VDTFS1: MOV     DL,[BP]-30H
        JE      VDNOST
MONLP1: MOV     CL,[BP]-2FH
        REP     MOVSB
        INC     SI
        DEC     DL
        JNE     MONLP1
        RET
GDGET   ENDP
                ;  ＶＲＡＭのデーターを作業域へ
VRMGET: MOV     AX,WIDE80
        MOV     DX,[BP]-30H
        INC     DH
        SUB     AL,DH
        XOR     CH,CH
        MOV     BX,VRMSEG
        XOR     DI,DI
        CALL    VRMGES
        CALL    VRMGES
VRMGES: PUSH    SI
        PUSH    DX
        MOV     DS,BX
VTLP3:  MOV     CL,DH
        REP     MOVSB
        ADD     SI,AX
        DEC     DL
        JNE     VTLP3
        POP     DX
        POP     SI
        ADD     BH,8
        RET
;  ***  BGSET  (ﾊﾞｯｸ･ｸﾞﾗｳﾝﾄﾞ･ｾｯﾄ)  ******
BGSET   PROC
        CALL    MVCHCK
        MOV     AX,[BP]-10H
        CMP     AX,[BP]+2AH
        JE      VRMCOP
        CMP     AX,1
        JA      VRMCOP
        SHL     AX,1
        SHL     AX,1
        AND     BYTE PTR [BP]+28H,0F3H
        OR      BYTE PTR [BP]+28H,AL
VRMCOP: MOV     AX,[BP]-0EH
        CMP     AX,1
        JBE     VRMCOK
        RET
```

```
        VRMCOK: MOV     [BP]+66H,AL
                XOR     AX,AX
        FGLOOP: DEC     AX
                JNE     FGLOOP
                MOV     CX,[BP]+20H
                XOR     DX,DX
                MOV     AX,4000H
                DIV     CX
                MOV     [BP]+60H,AX
                MOV     [BP]+62H,DX
        VRMTS1: MOV     DX,[BP]+60H
                XOR     SI,SI
                MOV     BX,[BP]+20H
                CLI
        VRMTLP: CALL    VRMTS2
                DEC     DX
                JNE     VRMTLP
                MOV     BX,[BP]+62H
                CALL    VRMTS2
                MOV     AL,0
                OUT     0A6H,AL
                STI
                RET
        VRMTS2: MOV     ES,[BP]+1CH
                XOR     DI,DI
                MOV     AL,1
                SUB     AL,[BP]+66H
                OUT     0A6H,AL
                MOV     AX,VRMSEG
                CALL    VRMTS3
                CALL    VRMTS3
                CALL    VRMTS3
                MOV     DI,SI
                MOV     DS,[BP]+1CH
                XOR     SI,SI
                MOV     AL,[BP]+66H
                OUT     0A6H,AL
                MOV     AX,VRMSEG
                CALL    VRMTS4
                CALL    VRMTS4
                CALL    VRMTS4
                MOV     SI,DI
                ADD     SI,BX
                ADD     SI,BX
                RET
        VRMTS3: MOV     DS,AX
                PUSH    SI
                MOV     CX,BX
                REP     MOVSW
                POP     SI
                ADD     AH,08H
                RET
        VRMTS4: MOV     ES,AX
                PUSH    DI
                MOV     CX,BX
                REP     MOVSW
                POP     DI
                ADD     AH,08H
                RET
        BGSET   ENDP
        ; ###  UNION  ( ユニオン )  **********
        UNION   PROC
                MOV     AX,[BP]-10H
                CMP     AX,1
                JNE     UNIOFF
                OR      BYTE PTR [BP]+28H,10H
                MOV     AX,[BP]-0EH
                CMP     AX,MVFLG
                JNE     TOUDT1
                TEST    BYTE PTR [BP]-40H,20H
                JE      UNIOFF
                XOR     AX,AX
                MOV     BX,[BP]-38H
                MOV     DX,[BP]-36H
                JMP     SHORT TOUDT2
        TOUDT1: MOV     BX,[BP]-0CH
                MOV     DX,[BP]-0AH
                SHR     DX,1
                ADD     DX,BX
        TOUDT2: MOV     [BP]+08H,AX
                MOV     [BP]+0AH,BX
                MOV     [BP]+0CH,DX
                JMP     SHORT TOSETD
        UNIOFF: AND     BYTE PTR [BP]+28H,0EFH
        TOSETD: JMP     SETDA2
        UNION   ENDP
        CODEND:
        CODE    ENDS
                END
        ; ##############################################
```

MOVING.H

```
/*-------------------------------------------------------------
**  Moving (コンパクト版)  デバイス・ドライバー
**  Lattice_C用  ヘッダー・ファイル
**                      Version 0.0
**      File "MOVING.H"
**          1988 8 / 5          By  H.Takeishi
-------------------------------------------------------------*/

#define moving_vec  0x0040    /*  割り込みベクター・コード    */
#define mvflag      0x4854    /*  パラメーター・パス・コード  */

/*---------------------  Moving  システム・コール  ----------*/

void    mvcall(mvcode,point)
    int mvcode,point;
{
    union   REGS    reg;
    struct  SREGS   seg;
    segread(&seg);
    reg.x.si = point;
    reg.x.dx = seg.ds;
    reg.x.ax = mvcode;
    int86( moving_vec , &reg , &reg );  /*  割込み  */
}

/*--------------------  MVSET  (ムーブ・セット)  ------------*/

void    mvset(d0,d1,d2,d3,d4)
    int d0,d1,d2,d3,d4;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /*  パタン数          */
    buf[1]  = d1;       /*  スプライトパタン数  */
    buf[2]  = d2;       /*  RGBパタン数  */
    buf[3]  = d3;       /*  セグメント領域数    */
    buf[4]  = d4;       /*  高速表示モード  */
    buf[5]  = mvflag;
    buf[6]  = mvflag;
    buf[7]  = mvflag;
    mvcode  = 0;
    mvcall(mvcode,buf);
}

/*--------------------  MVSETMEM  (ムーブ・セット・メモリー)  --*/

void    mvsetmem(d0,d1,d2,d3,d4,d5)
    int d0,d1,d2,d3,d4,d5;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /*  パタン数          */
    buf[1]  = d1;       /*  スプライトパタン数  */
    buf[2]  = d2;       /*  RGBパタン数  */
    buf[3]  = 0 ;       /*  セグメント・ダミー  */
    buf[4]  = d4;       /*  高速表示モード  */
    buf[5]  = mvflag;
    buf[6]  = d5;       /*  開始セグメント  */
    buf[7]  = d3;       /*  セグメント領域数    */
    mvcode  = 0;
    mvcall(mvcode,buf);
}

/*---------------------  GSIZE  (グラフィック・サイズ)  ----*/

void    gsize(d6,d7,d0,d1)
    int d6,d7,d0,d1;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /*  パタン番号      */
    buf[1]  = d1;       /*  コマ数      */
    buf[2]  = mvflag;
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = d6;       /*  水平サイズ      */
    buf[7]  = d7;       /*  垂直サイズ      */
    mvcode  = 1;
    mvcall(mvcode,buf);
}

/*--------  GDSET  (グラフィック・データー・セット)  --------*/

void    gdset(d0,d1,d2,d3,d4)
    int d0,d1,d2,d3,d4;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /*  パタン・データー    */
    buf[1]  = d1;       /*  パタン・ポインター  */
    buf[2]  = d2;       /*  パタン番号      */
    buf[3]  = d3;       /*  RGB面指定コード  */
    buf[4]  = d4;       /*  コマ番号        */
    buf[5]  = mvflag;
    buf[6]  = mvflag;
    buf[7]  = mvflag;
    mvcode  = 2;
    mvcall(mvcode,buf);
}

/*---------------------  GDISP  (グラフィック・ディスプ)  --*/

void    gdisp(d6,d7,d0,d1,d2)
    int d6,d7,d0,d1,d2;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /*  パタン番号      */
    buf[1]  = d1;       /*  色コード        */
    buf[2]  = d2;       /*  コマ番号        */
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = d6;       /*  水平位置        */
    buf[7]  = d7;       /*  垂直位置        */
    mvcode  = 3;
```

```
    mvcall(mvcode,buf);
}

/*---------------------- GPOKE (グラフィック・ポーク) ----*/

void    gpoke(d6,d7,d0,d1,d2)
    int d6,d7,d0,d1,d2;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /* パタン番号      */
    buf[1]  = d1;       /* 色コード        */
    buf[2]  = d2;       /* ポインター      */
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = d6;       /* 水平位置        */
    buf[7]  = d7;       /* 垂直位置        */
    mvcode  = 4;
    mvcall(mvcode,buf);
}

/*---------------------- GPEEK (グラフィック・ピーク) ----*/

void    gpeek(d6,d7,d0,d1,d2)
    int d6,d7,d0,d1,d2;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /* スタートポインター  */
    buf[1]  = d1;       /* ポインター数    */
    buf[2]  = d2;       /* 待ち時間        */
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = d6;       /* 水平位置        */
    buf[7]  = d7;       /* 垂直位置        */
    mvcode  = 5;
    mvcall(mvcode,buf);
}

/*----------GDGET (グラフィック・データー・ゲット) --------*/

void    gdget(d6,d7,d0,d1,d2)
    int d6,d7,d0,d1,d2;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /* パタン番号      */
    buf[1]  = d1;       /* RGB面指定コード */
    buf[2]  = d2;       /* コマ番号        */
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = d6;       /* 水平位置        */
    buf[7]  = d7;       /* 垂直位置        */
    mvcode  = 6;
    mvcall(mvcode,buf);
}

/*------------------ BGSET (バック・グラウンド・セット) --*/

void    bgset(d0,d1)
    int d0,d1;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /* 設定モード      */
    buf[1]  = d1;       /* コピー・モード  */
    buf[2]  = mvflag;
    buf[3]  = mvflag;
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = mvflag;
    buf[7]  = mvflag;
    mvcode  = 7;
    mvcall(mvcode,buf);
}

/*------------------ UNION (ユニオン) ---------------------*/

void    gunion(d0,d1,d3)    /* UNIONは 予約語なので */
    int d0,d1,d3;   /* GUNIONに変更     */
{
    int mvcode,buf[8];

    struct  SREGS   seg;
    segread(&seg);

    buf[0]  = d0;       /* 共有設定モード */
    buf[1]  = d1;       /* スタートアドレス    */
    buf[2]  = seg.ds;   /* セグメント      */
    buf[3]  = d3;       /* 配列数      */
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = mvflag;
    buf[7]  = mvflag;
    mvcode  = 8;
    mvcall(mvcode,buf);
}

/*------------------ SPRITE (スプライト) ----------------*/

void    sprite(d0,d1,d2,d3)
    int d0,d1,d2,d3;
{
    int mvcode,buf[8];
    buf[0]  = d0;       /* ポインター          */
    buf[1]  = d1;       /* 同時表示スプライト数  */
    buf[2]  = d2;       /* アニメ数            */
    buf[3]  = d3;       /* コマ送りの間の時間    */
    buf[4]  = mvflag;
    buf[5]  = mvflag;
    buf[6]  = mvflag;
    buf[7]  = mvflag;
    mvcode  = 9;
    mvcall(mvcode,buf);
}

/*-------------------------------------------------------------*/
```

moving.pas

```
(*-------------------------------------------------------------
**  Moving (コンパクト版)  デバイス・ドライバー
**  TurboPascal用 ヘッダー・ファイル
**                                          Version 0.0
**          File "moving.pas"
**
**          1988  8/1                   by H.Takeishi
-------------------------------------------------------------*)

const
  moving_vec = $0040;    (* 割り込みベクター・コード    *)
  mvflag     = $4854;    (* パラメーター・パス・コード  *)
Type
  Register = record
               ax,bx,cx,dx,bp,si,di,ds,es,flag : integer
             end;
var
  buf : array [0..7]  of integer;
  reg : Register;
  mvcode : integer;

(*---------------------- Moving システム・コール ------------*)

procedure  Mvcall(mvcode : integer);
begin
  with reg do
    begin
      AX := mvcode+$40;
      DX := seg(buf);
      SI := ofs(buf);
      intr( moving_vec , reg );         (* 割込み  *)
      if ax =1 then halt;
    end;
end;

(*-------------------- MVSET (ムーブ・セット)  ------------*)

procedure   Mvset(d0,d1,d2,d3,d4 : integer);
begin

  with reg do
    begin
      AX := $4a00;
      BX := $2000;
      ES := Cseg;
      intr( $21 , reg );            (* 割込み  *)
    end;
    buf[0] := d0;       (* パタン数            *)
    buf[1] := d1;       (* スプライトパタン数  *)
    buf[2] := d2;       (* RGBパタン数         *)
    buf[3] := d3;       (* セグメント領域数    *)
    buf[4] := d4;       (* 高速表示モード      *)
    buf[5] := mvflag;
    buf[6] := mvflag;
    buf[7] := mvflag;
    mvcode := 0;
    Mvcall(mvcode);
end;

(*----------MvsetMem (ムーブ・セット・メモリー)  ----*)

procedure   MvsetMem(d0,d1,d2,d3,d4,d5 : integer);
begin
    buf[0] := d0;       (* パタン数            *)
    buf[1] := d1;       (* スプライトパタン数  *)
    buf[2] := d2;       (* RGBパタン数         *)
    buf[3] := 0;        (* セグメント・ダミー  *)
    buf[4] := d4;       (* 高速表示モード      *)
    buf[5] := mvflag;
    buf[6] := d5;       (* 開始セグメント      *)
    buf[7] := d3;       (* セグメント領域数    *)
    mvcode := 0;
    Mvcall(mvcode);
end;
(*------------MvsetHeap (ムーブ・セット・ヒップ)  ----*)

procedure   MvsetHeap(d0,d1,d2,d3,d4 : integer);
  Type
    arraytype = array [0..255] of byte ;
    arraypoin = ^arraytype ;
  var
    hp: array [0..4095] of arraypoin ;
```

```
    i,j,d6,d7 : integer ;
    heaptop : ^byte ;
  begin
    buf[0] := d0;       (*  パタン数            *)
    buf[1] := d1;       (*  スプライトパタン数  *)
    buf[2] := d2;       (*  ＲＧＢパタン数      *)
    buf[4] := d4;       (*  高速表示モード      *)
    buf[5] := mvflag;

    if d3 = mvflag then
      begin
        buf[3] := d3;       (*  セグメント領域数        *)
        buf[6] := mvflag;   (*  開始ヒープ・セグメント  *)
        buf[7] := mvflag;   (*  終了ヒープ・セグメント  *)
      end
    else
     begin
     mark(heaptop);
       if d3 > Memavail then
         begin
           writeln('');
           writeln(' ヒープ領域は 最大 ',Memavail,' です。');
           halt;
         end;
      j := (d3+$ff) div $10 ;
      for i:=0 to j do
          new(hp[i]);

      d6     := seg(hp[0]^[0]);
      d7     := $10*j ;
      buf[3] := 0;        (*  セグメント領域数０  *)
      buf[6] := d6;       (*  開始ヒープ・セグメント  *)
      buf[7] := d7;       (*  終了ヒープ・セグメント  *)
   end;

    mvcode := 0;
    Mvcall(mvcode);
end;

(*---------------------- ＧＳＩＺＥ （グラフィック・サイズ） ----*)

procedure Gsize(d6,d7,d0,d1: integer);
begin
    buf[0] := d0;       (*  パタン番号      *)
    buf[1] := d1;       (*  コマ数          *)
    buf[2] := mvflag;
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := d6;       (*  水平サイズ      *)
    buf[7] := d7;       (*  垂直サイズ      *)
    mvcode := 1;
    Mvcall(mvcode);
end;

(*-------- ＧＤＳＥＴ （グラフィック・データー・セット） --------*)

procedure  Gdset(d0,d1,d2,d3,d4: integer);
begin
    buf[0] := d0;       (*  パタン・データー    *)
    buf[1] := d1;       (*  パタン・ポインター  *)
    buf[2] := d2;       (*  パタン番号          *)
    buf[3] := d3;       (*  ＲＧＢ面指定コード  *)
    buf[4] := d4;       (*  コマ番号            *)
    buf[5] := mvflag;
    buf[6] := mvflag;
    buf[7] := mvflag;
    mvcode := 2;
    Mvcall(mvcode);
end;

(*---------------------- ＧＤＩＳＰ （グラフィック・ディスプ） -----*)

procedure  Gdisp(d6,d7,d0,d1,d2: integer);
begin
    buf[0] := d0;       (*  パタン番号  *)
    buf[1] := d1;       (*  色コード    *)
    buf[2] := d2;       (*  コマ番号    *)
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := d6;       (*  水平位置    *)
    buf[7] := d7;       (*  垂直位置    *)
    mvcode := 3;
    Mvcall(mvcode);
end;

(*---------------------- ＧＰＯＫＥ （グラフィック・ポーク） -----*)

procedure  Gpoke(d6,d7,d0,d1,d2: integer);
begin
    buf[0] := d0;       (*  パタン番号      *)
    buf[1] := d1;       (*  色コード        *)
    buf[2] := d2;       (*  ポインター      *)
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := d6;       (*  水平位置        *)
    buf[7] := d7;       (*  垂直位置        *)
    mvcode := 4;
    Mvcall(mvcode);
end;

(*---------------------- ＧＰＥＥＫ （グラフィック・ピーク） -----*)

procedure  Gpeek(d6,d7,d0,d1,d2: integer);
begin
    buf[0] := d0;       (*  スタートポインター  *)
    buf[1] := d1;       (*  ポインター数        *)
    buf[2] := d2;       (*  待ち時間            *)
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := d6;       (*  水平位置            *)
    buf[7] := d7;       (*  垂直位置            *)
    mvcode := 5;
    Mvcall(mvcode);
end;

(*---------------- ＧＤＧＥＴ （グラフィック・データー・ゲット） -----*)

procedure  Gdget(d6,d7,d0,d1,d2: integer);
begin
    buf[0] := d0;       (*  パタン番号          *)
    buf[1] := d1;       (*  ＲＧＢ面指定コード  *)
    buf[2] := d2;       (*  コマ番号            *)
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := d6;       (*  水平位置            *)
    buf[7] := d7;       (*  垂直位置            *)
    mvcode := 6;
    Mvcall(mvcode);
end;

(*------------------ ＢＧＳＥＴ （バック・グランド・セット） --------*)

procedure  Bgset(d0,d1: integer);
begin
    buf[0] := d0;       (*  設定モード      *)
    buf[1] := d1;       (*  コピー・モード  *)
    buf[2] := mvflag;
    buf[3] := mvflag;
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := mvflag;
    buf[7] := mvflag;
    mvcode := 7;
    Mvcall(mvcode);
end;

(*------------------ ＵＮＩＯＮ （ユニオン） ----------------------*)

(*  ＵＮＩＯＮは Ｃの予約語なのでＧＵＮＩＯＮに変更  *)

procedure  Gunion(d0,d1,d3: integer);
begin
    buf[0] := d0;       (*  共有設定モード      *)
    buf[1] := d1;       (*  スタートアドレス    *)
    buf[2] := Dseg;     (*  セグメント          *)
    buf[3] := d3;       (*  配列数              *)
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := mvflag;
    buf[7] := mvflag;
    mvcode := 8;
    Mvcall(mvcode);
end;

(*------------------ ＳＰＲＩＴＥ （スプライト） ----------------------*)

procedure  Sprite(d0,d1,d2,d3: integer);
begin
    buf[0] := d0;       (*  ポインター              *)
    buf[1] := d1;       (*  同時表示スプライト数    *)
    buf[2] := d2;       (*  アニメ・コマ数          *)
    buf[3] := d3;       (*  コマ送りの間の時間      *)
    buf[4] := mvflag;
    buf[5] := mvflag;
    buf[6] := mvflag;
    buf[7] := mvflag;
    mvcode := 9;
    Mvcall(mvcode);
end;

(*------------------------------------------------------------------*)
```

SAMPLE01.BAS

```
1000 ' ************************************************************************
1010 ' *    Ｍｏｖｉｎｇ  Ｄｅｍｏ
1020 ' *        File "SAMPLE01.BAS
1030 ' *        1988.8/1          By  H.Takeishi
1040 ' ************************************************************************
1050 '
1060 CONSOLE 0,25,0,1:SCREEN 3,0:CLS 3:COLOR 7
1070 ' GOTO *MONOSPRITE
1080 ' GOTO *RGBSPRITE
1090 ' GOTO *KANJIDSP
1100 '
1110 ' *****     グラフィックパタンの移動 （スプライト機能なし） *******
```

```
1120 '
1130 SCREEN 0:CLEAR &H2000:DEFINT X,Y
1140 DIM X(1800),Y(1800)
1150 MVSET 4,0,0,&H2000:BGSET 0
1160 GSIZE (1,5),0:GSIZE (1,5),1:GSIZE (1,5),2:GSIZE (2,16),3
1170 WIDTH 80:COLOR 6
1180 LOCATE 18,0
1190 PRINT "グラフィックパタンの移動 (スプライト機能なし)"
1200 FOR P=0 TO 4:GDSET 255,P,0:NEXT
1210 FOR P=0 TO 4:GDSET 255,P,1:NEXT
1220 LINE (100,15)-(539,70),2,BF
1230 FOR X= 100 TO 320:GDISP (X, 42),1,5:NEXT
1240 FOR T=1 TO 2000:NEXT
1250 FOR X= 100 TO 620:GDISP (X, 40),0,6:NEXT
1260 FOR T=1 TO 1000:NEXT
1270 CLS 2
1280 LINE (100,15)-(539,70),2,BF
1290 PUT(250,35),KANJI(VAL("&H"+JIS$("B")))
1300 PUT(400,35),KANJI(VAL("&H"+JIS$("G")))
1310 FOR C=1 TO 7
1320   LINE (50+C*60,100)-(110+C*60,180),C,BF
1330 NEXT
1340 BGSET 1,1
1350 FOR X= 20 TO 620:GDISP (X, 40),1,5:NEXT
1360 FOR T=1 TO 1000:NEXT
1370 GDISP (320,100),0,5:LINE (100,15)-(550,190),3,BF
1380 S=3.1416/900
1390 FOR P=0 TO 1800
1400   X(P)=320+SIN(3*P*S)*200:Y(P)=100-SIN(2*P*S)*80
1410   GDISP (X(P),Y(P)),0,5
1420 NEXT
1430 BGSET ,0
1440 FOR P=0 TO 900
1450   P2=P*2 MOD 1800:P3=P*3 MOD 1800
1460   GDISP (X(P),Y(P)),0,7
1470   GDISP (X(P2),Y(P2)),1,6
1480 NEXT
1490 FOR T=1 TO 3000:NEXT
1500 '
1510 *MONOSPRITE
1520 ' *********  グラフィックパタンの移動 (モノカラースプライト)  ******
1530 '
1540 WIDTH 80:SCREEN 0:CLS 3
1550 CLEAR &H2000:DEFINT A,X,Y
1560 DIM A(3,60),X(128),Y(128),C(128)
1570 LOCATE 12,0:PRINT "グラフィックパタンの移動 (モノカラースプライト)"
1580 MVSET 40,40,0,&H2000,2:BGSET 0:UNION 1
1590 GSIZE (1, 6),0
1600 GSIZE (6,24),1
1610 GSIZE (4,16),2
1620 GSIZE (4,16),3
1630 GSIZE (1, 6),4
1640 GSIZE (1, 4),5
1650 FOR N=6 TO 38:GSIZE (1,4),N:NEXT
1660 GSIZE (60,1),39
1670 RESTORE 3700
1680 FOR P=0 TO 5:READ D$:GDSET VAL("&H"+D$),P,0:NEXT
1690 RESTORE 3700
1700 FOR P=0 TO 5:READ D$:GDSET VAL("&H"+D$),P,4:NEXT
1710 LINE(310,90)-(357,113),4,BF
1720 LINE(316,93)-(351,110),0,BF
1730 GDGET (310,90),1,7
1740 FOR P=0 TO 31:READ D$:GDSET VAL("&H"+D$),P,2:NEXT
1750 RESTORE 3740
1760 FOR P=32 TO 63:READ D$:GDSET VAL("&H"+D$),P,2:NEXT
1770 GDISP (200,100),2,4
1780 GDGET (200,100),3,7
1790 GDISP (400,100),0,6
1800 FOR N=5 TO 38:GDGET(400,101),N:NEXT
1810 FOR P=0 TO 59:GDSET 255,P,39:NEXT
1820 CLS 2
1830 FOR I=1 TO 70
1840   CIRCLE(24+I*8,70+RND*80),RND*3,RND*6+1,,,F
1850 NEXT
1860 BGSET 1,1
1870 A(0,0)=0:A(1,0)=2:A(3,0)=108
1880 A(0,1)=1:A(1,1)=6:A(3,1)=100
1890 A(0,2)=2:A(1,2)=5:A(3,2)=105
1900 A(0,3)=3:A(1,3)=3:A(3,3)=112
1910 A(0,4)=4:A(1,4)=4:A(3,4)=116
1920 FOR I=40 TO 180
1930   A(2,0)=410-I
1940   A(2,1)=I+198
1950   A(2,2)=642-I*2
1960   A(2,3)=I*2-14
1970   A(2,4)=432-I
1980   SPRITE ,5
1990 NEXT
2000 UNION 0
2010 FOR I=0 TO 1000:NEXT
2020 '
2030 S=3.1416/8:L=33
2040 FOR P=0  TO  63:Y(P)=110+SIN(P*S)*20:NEXT
2050 FOR P=64 TO 128:Y(P)=110            :NEXT
2060 FOR P=0 TO 64
2070   IF P MOD 8=0 THEN C(P)=6 ELSE C(P)=7
2080 NEXT
2090 FOR I=0 TO 48
2100   FOR P=0 TO L-1
2110     GPOKE (176+P*8,Y(64+P-I)),P+5,C(P),P+I*L
2120   NEXT
2130   SPRITE I*L,L
2140 NEXT
2150 SPRITE 0,L,48
2160 FOR N=0 TO 5:GDISP ,N,-1:NEXT
2170 BGSET 0
2180 FOR Y=10 TO 180:GDISP (80,Y),39,7:NEXT
2190 SPRITE 0,L,48,100
2200 FOR N=0 TO 5:GDISP ,N,&H60:NEXT
2210 FOR Y=180 TO 10 STEP -1:GDISP (80,Y),39,7:NEXT
2220 SPRITE 0,L,48
2230 BGSET 1
2240 FOR Y=10 TO 199:GDISP (80,Y),39,7:NEXT
2250 FOR I=0 TO 3000:NEXT
2260 '
2270 *RGBSPRITE
2280 ' ********  グラフィックパタンの移動 (ＲＧＢスプライト)  **********
2290 '
2300 WIDTH 80:SCREEN 3,,0,1:CLS 3
2310 CLEAR &H2000:DEFINT D:DIM D(3)
2320 MVSET 9,9,6,&H2000,2:UNION 0
2330 GSIZE(14,21),0
2340 GSIZE(14,21),1
2350 GSIZE( 6,40),2
2360 GSIZE(11,18),3
2370 GSIZE(11,18),4
2380 GSIZE( 6,40),5
2390 GSIZE(14,21),6
2400 GSIZE(14,21),7
2410 LOCATE 12,0:PRINT "グラフィックパタンの移動 (ＲＧＢスプライト)"
2420 LINE (0,0)-(639,399),1,BF
2430 FOR P=1 TO 1000:PSET (RND*639,RND*399),6:NEXT
2440 FOR P=1 TO 100
2450   X=RND*639:Y=RND*199
2460   CIRCLE(X,Y),RND*2,6,,,,6,F
2470 NEXT
2480 CIRCLE(320,400),300,4,,,,7,F
2490 CIRCLE(360,50),20,6,,,,5,F
2500 BGSET 0,1
2510 SCREEN 0,,0,1
2520 FOR T=1 TO 2000:NEXT
2530 CLS 2
2540 CIRCLE(320,100),50,,,,,2,F
2550 CIRCLE(320,100),50,3,0,2,,2
2560 GDGET(270,90),0
2570 GDGET(270,90),1
2580 GDGET(270,90),6
2590 GDGET(0,0),7
2600 '
2610 '                点でグラフィックを表示、それをゲットする
2620 RESTORE 3820
2630 FOR P=0 TO 37
2640   READ D$
2650   FOR J=1 TO 24
2660     C=VAL("&H"+MID$(D$,J,1))
2670     PSET (330+J*2,160+P),C
2680     PSET (331+J*2,160+P),C
2690     PSET (200+P*2,160+J),C
2700     PSET (201+P*2,160+J),C
2710     PSET (500-P*2,160+J),C
2720     PSET (501-P*2,160+J),C
2730   NEXT
2740 NEXT
2750 GDGET (330,160),2
2760 GDGET (426,166),3
2770 GDGET (200,166),4
2780 FOR I=330 TO 300 STEP -1:GDISP(I,160),2:NEXT
2790 FOR I=0 TO 150:GDISP(300,160-I*I/10),2:NEXT
2800 FOR P=0 TO 700:GPOKE (632-P,120),4,1,P:NEXT
2810 SPRITE ,1,680
2820 FOR I=0 TO 680 STEP 3:SPRITE I,1:NEXT
2830 FOR I=0 TO 320 STEP 5:SPRITE I,1:NEXT
2840 FOR T=1 TO 2000:NEXT
2850 '
2860 BGSET 1
2870 X=-1:Y=0:S=-1
2880 GDISP (328+X*64,100+Y*20),7,0
2890 FOR I=1 TO 9
2900   M=I:IF I=9 THEN M=I-1
2910   FOR J=1 TO I:Y=Y+S:GDISP (328+X*64,100+Y*20),7,32:NEXT
2920   FOR J=1 TO M:X=X+S:GDISP (328+X*64,100+Y*20),7,32:NEXT
2930   S=-S
2940 NEXT
2950 GDISP (270, 90),0
2960 GDISP (300,130),1
2970 GDISP (342,160),2
2980 GDISP (280, 60),6,7
2990 GDISP (290,120),7,7
3000 FOR X=320 TO -100 STEP -4:GDISP(X,120),4:NEXT
3010 FOR I=0 TO 78:GDISP(342,160-I-.02*I*I),2:NEXT
3020 FOR I=300 TO -80 STEP -2:GDISP(I,20-I/10),4:NEXT
3030 FOR I=-80 TO 320 STEP  2:GDISP(I,20-I/8 ),3:NEXT
3040 FOR I=1 TO 10:GDISP (270+I*3,90-I),0:NEXT
3050 FOR I=1 TO 10:GDISP (300+I*6,130),1:NEXT
3060 FOR I=1 TO 32
3070   Y=RND
3080   GDISP (290+RND*100,50+Y*30+RND*60),6+Y,7
3090   FOR T=1 TO 150:NEXT
3100 NEXT
3110 FOR P=0 TO 6*40-1:GDSET 255,P,2,16:NEXT
3120 GDISP (-80,200),3
3130 FOR I=-50 TO 120:GDISP(340,I),2:NEXT
3140 GDSET ,,2
3150 FOR I=121 TO 130:GDISP(340,I),2:NEXT
3160 FOR I=130 TO 150:GDISP(340,I),2:NEXT
3170 SCREEN 3
3180 FOR I=151 TO 250:GDISP(340,I),2:NEXT
3190 FOR T=1 TO 3000:NEXT
3200 '
3210 *KANJIDSP
3220 ' *******  漢字フォントをグラフィックパタンとして表示  ************
3230 '
3240 SCREEN 3:CLS 3:SCREEN 0
3250 CLEAR &H2000
3260 MVSET 26,26,0,&H2000,2:BGSET 0
3270 RESTORE 4230
3280 FOR I=0 TO 25:GSIZE(2,16),I:NEXT
3290 FOR I=0 TO 25
3300   READ D$
```

```
3310   PUT(50+I*20,10),KANJI(VAL("&H"+D$))
3320   GDGET(50+I*20,10),I
3330 NEXT
3340 FOR I=3  TO 15:GDISP(I*40-60,40),I,5:NEXT
3350 FOR I=20 TO 25:GDISP(I*64-1150, 80),I,3:NEXT
3360 FOR T=1 TO 2000:NEXT
3370 P=0
3380 FOR S=32 TO 10 STEP -1
3390   FOR I= 0 TO  9 STEP  1
3400     GPOKE(300-S*8+I*S,140),I,I MOD 5+2+64,P+I
3410   NEXT
3420   FOR I=19 TO 10 STEP -1
3430     GPOKE(300-S*8+I*S,140),I,I MOD 5+2+64,P+I
3440   NEXT
3450   SPRITE P,20:P=P+20
3460 NEXT
3470 SCREEN 3
3480 I=20:D$="2334"
3490 PUT(50+I*20,10),KANJI(VAL("&H"+D$)),PSET
3500 GDGET(50+I*20,10),I
3510 FOR I=20 TO 25:GDISP(I*64-1150, 80),I,3:NEXT
3520 P=0
3530 FOR S=10 TO 32
3540   FOR I= 0 TO  9 STEP  1
3550     GPOKE(300-S*8+I*S,140+S*6),I,I MOD 5+2,P+I
3560   NEXT
3570   FOR I=19 TO 10 STEP -1
3580     GPOKE(300-S*8+I*S,140+S*6),I,I MOD 5+2,P+I
3590   NEXT
3600   SPRITE P,20:P=P+20
3610 NEXT
3620 FOR I=22 TO 0 STEP -2:SPRITE I*20,20:NEXT
3630 SPRITE 0,20,22
3640 FOR I=22 TO 0 STEP -2:SPRITE I*20,20:NEXT
3650 SPRITE 0,20,22
3660 END
3670 '
3680 '    ﾊﾟﾀﾝ ﾃﾞｰﾀｰ ( NO 0,4 )
3690 '
3700 DATA 3C,3C,FF,FF,3C,3C
3710 '
3720 '    ﾊﾟﾀﾝ ﾃﾞｰﾀｰ ( NO 2,3 )
3730 '
3740 DATA 01,03,07,0F,1F,3F,7F,FF
3750 DATA FF,7F,3F,1F,0F,07,03,01
3760 DATA 80,C0,E0,F0,F8,FC,FE,FF
3770 DATA FF,FE,FC,F8,F0,E0,C0,80
3780 '
3790 '    ﾛｹｯﾄ ﾊﾟﾀﾝ ﾃﾞｰﾀｰ ( NO 2 )   ﾖｺ(24ﾄﾞｯﾄ=3ﾗｲﾝ)*ﾀﾃ(38ﾄﾞｯﾄ)
```

```
3800 '
3810 '    01234567890123456789012345678
3820 DATA ..............55..............
3830 DATA .............66...............
3840 DATA .............66...............
3850 DATA .............66...............
3860 DATA ............3553.............
3870 DATA ............5225.............
3880 DATA ............5225.............
3890 DATA ............5225.............
3900 DATA ............5225.............
3910 DATA ............5445.............
3920 DATA ...........254452............
3930 DATA ...........577775............
3940 DATA ...........572275............
3950 DATA ...........572275............
3960 DATA ...........577775............
3970 DATA ...........514415............
3980 DATA ...........544445............
3990 DATA ...........722227............
4000 DATA ...........736637............
4010 DATA ...........737737............
4020 DATA ...........736637............
4030 DATA ...........521125............
4040 DATA .......3...577775...3.......
4050 DATA ......565..511115..565......
4060 DATA ......515.55444455.515......
4070 DATA .....751555645546555667.....
4080 DATA .....753555777775555357.....
4090 DATA .....7555.34444443.5557.....
4100 DATA ......525..222222..525......
4110 DATA .......6....6666....6.......
4120 DATA .......2....2222....2.......
4130 DATA ......212...2222...212......
4140 DATA ....77212..221122..21277....
4150 DATA ......262..266662..262......
4160 DATA ......262..226622..262......
4170 DATA .......2....2662....2.......
4180 DATA ............2662............
4190 DATA ..............22..............
4200 '
4210 ' 漢字コード
4220 '
4230 DATA 2433,246C,244F,234D,236F,2376,2369,236E,2367,244B
4240 DATA 2468,246B,3441,3B7A,493D,3C28,2447,2439,2123,2121
4250 DATA 2332,2330,2330,2569,2524,2573
```

SAMPLE02.BAS

```
100 ' ************************************************************************
110 ' *    Ｍｏｖｉｎｇ コンパクト版サンプルプログラム                        *
120 ' *          File "SAMPLE02.BAS
130 ' *                                          1988. 8/2   By  H.Takeishi    *
140 ' ************************************************************************
150 '
160 WIDTH 80:SCREEN 0,,0,1:CLS 3
170 CLEAR &H2000
180 MVSET 1,1,1,&H2000
190 GSIZE (8,32),0,4
200 C(0)=0:C(1)=1:C(2)=2:C(3)=6:C(4)=1:C(5)=2:C(6)=2:C(7)=7
210 FOR P=0 TO 15
220   READ D$
230   FOR J=1 TO 17*4-1
240     K=VAL(MID$(D$,J,1))
250     LINE (100+J*4,70+P*2)-(103+J*4,71+P*2),C(K),BF
260   NEXT
270 NEXT
280 FOR I=0 TO 3
290   GDGET (100+I*68,70),0,,I
300   GDISP (100+I*68 ,140),0,,I
310 NEXT
320 FOR X=639 TO 0 STEP -8
330   GDISP(X,120),0
340   FOR T=0 TO 200:NEXT
350 NEXT
360 END
370 '
380 '    ﾊﾟﾀﾝ ﾃﾞｰﾀｰ  ﾖｺ(16ﾄﾞｯﾄ=2ﾗｲﾝ)*ﾀﾃ(16ﾄﾞｯﾄ)
390 '
400 '    0123456789012345 0123456789012345 0123456789012345 0123456789012345
410 '
420 DATA ......222....... .....222....... .....222....... ................
430 DATA .....22222..... ....22222..... ....22222..... .....222.......
440 DATA .....3322..... .....3322..... .....3322..... ....22222.....
450 DATA ....33332..... ....33332..... ...3.33332..... .....3322.....
460 DATA .....33..... .....33..... ...3..33..... ....33332.....
470 DATA .....444..... .....444..... ..44.44444..... .....33.....
480 DATA .....4444..... .....4444..... ....44444.4..... ....444.....
490 DATA ....34444..... ...334444..... .....444.3..... ....4444.....
500 DATA .....334..... ...66.663..... .....6666.3..... ...334444.....
510 DATA .....666..... ...666.6..... ...666.6..... .....4445.....
520 DATA .....666..... ....6616..... ..666.66..... .....666.11....
530 DATA .....66..... .....1166..... ....66..666..... .....666611....
540 DATA ....1166..... ....11.66..... ...61..666..... ....66.66.1....
550 DATA .....66..... .....66..... ....11...66..... ....66.....
560 DATA .....11..... .....11..... ...11....11..... ...11.....
570 DATA .....111..... .....111..... .....11..... ..111.....
```

SAMPLE1.C

```
/*
  ****************************************************************
  *   Ｍｏｖｉｎｇ（コンパクト版）  デバイス・ドライバー
  *   Ｃ言語での  サンプルプログラム１
  *   File "SAMPLE1.C
  *              1988 8/3    By  H.Takeishi
  ****************************************************************

    Ｍｏｖｉｎｇコマンド（１０）
    MVSET   GSIZE   GDSET   GDISP   GPOKE
    GPEEK   GDGET   BGSET   GUNION  SPRITE
*/

#include <dos.h>
#include <moving.h>

/* #define  mvflag  0x4854     moving.h の中で定義されている  */

void    main()
{
    int n,s,c,m,f,i,x,y,vx,vy,p;
    int u[4][4];

    f = mvflag;
    n = 4;
    s = 3;
    c = 2;
    m = 0x2000;

    mvset (n,s,c,m,2);

    gsize(2,16,0,f,f);
    gsize(3,24,1,f,f);
    gsize(3,24,2,f,f);
    gsize(1, 4,3,f,f);

        for(i=0;i<4 ;i++)   gdget(0,0,i,f,f);

    for(p=0;p<32;p++)   gdset(255,p,0,f,f);

    for(p=0;p<72;p++)   gdset(255,p,1,16,f);
    c = 0;
    for(p=0;p<72;p++)
    {
        c = c+1;
        if( c>7 )  c=1;
        gdset(255,p,1,c,f);
    }
    for(p=0;p<72;p++)   gdset(p,p,2,f,f);

    gdset(0x7e,0,3,f,f);
    gdset(0xff,1,3,f,f);
    gdset(0xff,2,3,f,f);
    gdset(0x7e,3,3,f,f);
```

```
    bgset(1,1);

    for(x=0;x<320;x++)  gdisp(x,100,0,f,f);

    printf(" [2J");

    for(x=640;x>320;x--)    gdisp(x,100,1,f,f);

    for(y=100;y<200;y++)    gdisp(320,y,1,f,f);

    for(y=200;y>0;y--)  gdisp(320,y,1,f,f);

    for(y=0;y<100;y++)  gdisp(320,y,1,f,f);

    p = 0;
    x = 0;
    y = 0;
    vx= 2;
    vy= 1;
    for(i=0;i<2030;i++)
    {
        gdisp(x,y,3,6,f);
        gpoke(x,y,3,6,p);
        p = p+1;
        x = x+vx;
        y = y+vy;
        if(x>624)
        {
            vx = -vx;
            x  = x+vx;
        }
        if(x<0  )
        {
            vx = -vx;
            x  = x+vx;
        }
        if(y>192)
        {
            vy = -vy;
            y  = y+vy;
        }
        if(y<0  )
        {
            vy = -vy;
            y  = y+vy;
        }
    }
    gpeek(0,0,0,p,1500);

    for(x=-40;x<700;x++)
    {
        gdisp(x      ,120,0,f,f);
        gdisp(x*2-300,120,1,f,f);
        gdisp(640-x*2,120,3,5,f);
    }

    gunion (1,&u[0][0],4*4);

    for( i=0;i<3;i++ )
    {
        u[i][0]=i;
        u[i][1]=f;
        u[i][3]=120;
    }
    for(x=-40;x<700;x++)
    {
        u[0][2]=x;
        u[1][2]=x*2-300;
        u[2][2]=640-x*2;
        sprite(f,3,f,f);
    }
}
```

SAMPLE1.PAS

```
(*************************************************************)
(*   Ｍｏｖｉｎｇ デバイス・ドライバー                       *)
(*   ＰＡＳＣＡＬ言語での  サンプルプログラム１             *)
(*   File "SAMPLE1.PAS                                       *)
(*              1988 8/3    By  H.Takeishi                   *)
(*************************************************************)
(*                                                           *)
(*  Ｍｏｖｉｎｇコマンド（ 10 ）チェック                    *)
(*  MVSET   GSIZE   GDSET   GDISP   GPOKE                    *)
(*  GPEEK   GDGET   BGSET   GUNION  SPRITE                   *)

(* #define  mvflag  0x4854     moving.h の中で定義されている  *)

program sample1;

{$i moving.pas }

var
    n,s,c,m,f,i,j,x,y,vx,vy,d,p : integer;
    u : array [0..3,0..3] of integer ;
    xz,yz : integer;
    step : real;

begin
    f := mvflag;
    n := 4;
    s := 3;
    c := 3;
    m := $2000;
    mvsetheap (n,s,c,m,2);

    gsize(2,16,0,f);
    gsize(3,24,1,f);
    gsize(4,32,2,f);
    gsize(1, 4,3,f);

    for i:=0 to 3   do  gdget(0,0,i,f,f);

    for p:=0 to 31  do  gdset(255,p,0,16,f);

    for p:=0 to 127 do  gdset(255,p,2,5,f);

    gdisp(320,100,2,f,f);
    gdisp(328,108,0,f,f);
    gdget(320,100,2,f,f);
    gdisp(360,100,2,5,f);

    for p:=0 to 31 do  gdset(255,p,0,3,f);

    gdset($7e,0,3,f,f);
    gdset($ff,1,3,f,f);
    gdset($ff,2,3,f,f);
    gdset($7e,3,3,f,f);
    gdisp(332,119,3,6,f);
    for p:=0 to 71 do  gdset(255,p,1,16,f);
    c := 0;
    for p:=0 to 71 do
    begin
        c := c+1;
        if( c>7 )  then c:=1;
        gdset(255,p,1,c,f);
    end;

    bgset(1,1);

    for x:=0 to 320   do  gdisp(x,100,0,f,f);

    for x:=640 to 320 do  gdisp(x,100,1,f,f);

    for y:=100 to 200 do  gdisp(320,y,1,f,f);

    for y:=200 downto 0   do  gdisp(320,y,1,f,f);
    for y:=  0 to 100 do  gdisp(320,y,1,f,f);

    for x:=-40 to 700 do
    begin
        gdisp(x      ,120,0,f,f);
        gdisp(x*2-300,120,1,f,f);
        gdisp(640-x*2,125,2,5,f);
    end;

    step := 256;
    for i:=0  to 512 do
    begin
        X := Trunc(320+Cos(i*3/step)*200) ;
        Y := Trunc(100-Sin(i*3/step)*80) ;
        gdisp(X,Y,3,6,f);
    end;

    gunion (1,Ofs(u),4*4);

    for  i:=0 to 2 do
    begin
        u[i,0]:=i;
        u[i,1]:=f;
        u[i,3]:=120;
    end;
    for x:=-40 to 700 do
    begin
        u[0,2]:=x;
        u[1,2]:=x*2-300;
        u[2,2]:=640-x*2;
        sprite(f,3,f,f);
    end;
end.
```

SAMPLE2.C

```
/*
  ****************************************************************
  *   Ｍｏｖｉｎｇ（コンパクト版）  デバイス・ドライバー
  *   Ｃ言語での  サンプルプログラム２
  *   File "SAMPLE2.C
  *              1988 8/5    By  H.Takeishi
  ****************************************************************
```

```
    ＧＬＩＯ．Ｈ をインクルードするとＢＡＳＩＣライクに
    グラフィック命令が活用できる
*/

#include <dos.h>
#include <moving.h>
#include <glio.h>
```

```
/* #define  mvflag  0x4854      moving.h の中で定義されている  */

void    main()
{
    int f,i,x;

    f = mvflag;
    mvset (2,2,1,0x2000,f);
    init();
    screen(3,1,0,1);

    gsize(4,32,0,f);
    gsize(2,16,1,f);

    circle(115,115,15,15,6,1);
    gdget(100,100,0,f,f);

    line(100,200,115,215,3,2,0,0,0);

    gdget(100,200,1,f,f);

    cls(3);
    for(i=1;i< 16 ; i ++ )
        {
        circle(320,200,200-i*10,i*5+i*i/2,i,0);
        }
    bgset(1,1);

    for(x=-40;x<700;x++)
    {
        gdisp(x      ,190,0,f,f);
        gdisp(x*2-300,200,1,5,f);
    }

}
```

SAMPLE2.PAS

```
(*************************************************************)
(*   Moving デバイス・ドライバー                             *)
(*   PASCAL言語での  サンプルプログラム２                    *)
(*   File 'SAMPLE2.PAS                                      *)
(*              1988 8/5    By  H.Takeishi                   *)
(*************************************************************)
(*      GLIO.H をインクルードするとＢＡＳＩＣライクに       *)
(*      グラフィック命令が活用できる                         *)

(* #define  mvflag  0x4854      moving.h の中で定義されている  *)

program sample2;

{$i moving.pas }
{$i glio.pas }

var
    f,i,x,y : integer;

begin
    f := mvflag;
    mvsetheap (2,2,1,$2000,f);
    init;
    screen(3,1,0,1);

    gsize(4,32,0,f);
    gsize(2,16,1,f);

    circle(115,115,15,15,6,1);
    gdget(100,100,0,f,f);

    line(100,200,115,215,3,2,0,0,0);
    gdget(100,200,1,f,f);

    cls(3);
    for i:=1 to 15 do
    begin
        x := 200-i*10 ;
        y := Trunc(i*5+i*i/2) ;
        circle(320,200,x,y,i,0);
    end;
    bgset(1,1);

    for x:=-40 to 700 do
    begin
        gdisp(x      ,190,0,f,f);
        gdisp(x*2-300,200,1,5,f);
    end;
end.
```

COFFEE BREAK

プログラムが読みづらい理由

■村田和信

より読みやすいプログラムを作るには，逆に，「なぜ多くの人にとってプログラムが読みにくいのか」を調べるのもひとつの方法です．読みにくい理由が分かれば，よいプログラムを作る上でおおいに参考になります．

●多義性と汎用性

プログラムは数多くのステップから構成されています．これらのステップが自然言語に1対1に対応していれば，プログラムの解読は簡単なことです．残念ながら今のプログラムのステップは自然言語とは必ずしも対応していません．

それどころか，前回話したように，1つのステップにいろいろな意味付けができます．また逆に，同じ意味を表わすプログラムを1通りに決められるわけではありません．したがって，プログラムの各ステップがコンピュータに対して何を指示しているかが理解できても，プログラムの意味が分かったことにはなりません．

この対応付けのあいまいさはプログラムを読む立場からすれば誠に困った性質です．しかし，この性質はコンピュータのプログラムに汎用性がある証でもあります．プログラムの1つのステップが特定の意味としか対応しなければ，コンピュータに汎用性を持たせるには無制限に多くの命令を追加していかなければなりません．

つまり，プログラムの多義性は汎用性の代償であるといえます．限られた命令で多くの目的に利用するには，どうしても命令の意味は抽象的にならざるをえない宿命にあるのです．プログラムのステップの意味をいくら考えていてもプログラムの意味が分からないのは当然です．

プログラムを習い始めた人がいざ何かの目的でプログラムを作ろうとすると，始めにひっかかる原因もここにあります．これは現在のプログラムの構造や勉強の仕方にも問題があります．

確かにプログラムの文法や各ステップの意味は基本的な知識であることは間違いありません．これでは日本の外国語数育と同じで，試験の問題は解けても会話になるとまったく駄目という状況とひとつも変わりません．

プログラムを取り扱うには文法は当然知っていなければなりません．それをどのように応用するかはまったく別次元の問題である，という認識がなさ過ぎるのではないでしょうか．この勘違いがプログラムが読めないし作れない原因の根底にあると思います．

●処理のための処理

プログラムをどうやって応用するかは，即ちプログラムにどのような意味付けをするかということです．これはいままでに何回か話したとおり，現実の問題をプログラムにどうやって対応付けるかという問題と本質的に同じ問題であると考えられます．

プログラムと現実の問題は1対1に対応するのではなく，いくつかの対応付けのレベルがあります．さらにこの対応付けはプログラムの各ステップごとになんレベルかの対応付けがあるという具合に分解できるものではありません．

むしろ，プログラムの中のいくつかの関連したステップがひとつの事象に対応していたり，ひとつのステップがいくつかの事象に対応していることもあります．この対応付けのあいまいさが第三者にとってプログラムを理解する妨げになっています．

*

またプログラムのステップには現実の問題とまったく対応しないものもあります．たとえば，ファイルをオープンする命令はほとんどの言語にあります．しかし，このステップと対応する現実の事象はまずありません．

オープン命令はファイルを読み書きする前に必ず実行しなければならないことになっています．つまり，オープン命令はファイルを読み書きする命令を実行する前提条件を整えるためだけにある命令です．

この例は比較的単純な例です．プログラムを読むときにはオープン命令の位置にだけ気をつけておけばよいことです．ところが，プログラムの作り手がプログラムを実行するためだけに必要なステップを追加するとなるとオープン命令の例のように簡単にいきません．

つまり，プログラムの意味付けはレベルがいくつかあるだけではなく，さらに処理のための処理などという余計なものが入り込んでいることになります．これもまた初学者にとってプログラムを理解するときに大きな障壁になっているような気がします．

いかにして現実の事象に対応しない処理のためだけの処理を排除するかは，読みやすいプログラムを作るための大きな課題であると同時に，プログラミング言語そのものの問題でもあります．

●無駄な変数

この課題を克服するヒントはプログラムの中の変数の整理の仕方にあります．この問題に注目する人は，メモリの小さなコンピュータを取り扱っているなどの特殊な状況でなければ，あまりいません．

また，プログラムを習い始めた人にとってはプログラムの文法やロジックの流れなど考えなければならないことがいっぱいあります．特に文法ミスをするとコンパイラは即座にあの忌まわしいエラーメッセージをたくさん表示してきます．またロジックの流れをうまく作らないとプログラムの処理が先に進んでくれません．

こんな環境でプログラムを作るとなると，これだけで頭がいっぱいになって他のことなど考える余裕がないのは当然かも知れません．まして，最近のコンピュータはパソコンでさえかなり大きなメモリを持っています．多少の変数の増加などハードウェアが解決してくれると考えたくなるのはもっともです．

*

でもちょっと待ってください．もしまったく無駄な変数ならともかく，何らかの役に立っているなら，少なくとも1回ずつの代入と参照があるはずです．つまり変数が1つ増えると少なくとも2ステップのロジックが増えることになります．だとしたら無駄なプログラムを書かないようにするにはまず無駄な変数を減らすことが最も効果のある整理の方法なのです．

また変数は現実の問題の中の状態に対応していてある単語として表現できます．そこで現実の問題に対応しない変数があったら，それはまず処理のための処理を生み出している可能性があります．

つまり現実の問題に対応した名前の付けられない変数をいかに少なくするかは，プログラムを読みやすくする第1歩であるといえます．またプログラムの無駄なステップが減ればそれだけデバッグの手間もかけずに済みますし信頼性も向上します．

●行間の意味

最後のプログラムを読む側の問題について簡単に触れておきます．プログラムを読む作業は決して機械的な作業ではありません．それはプログラムを作った人がその問題に対してどのように考え，どのようにそれをプログラムとして表現したかを追跡する作業です．

それはちょうど文章の行間に含まれた意味を理解する作業と似ています．プログラムはあくまでもコンピュータの動作を規定するための指示書です．つまり，プログラムそのものはプログラムが何を目的とし，また何を意味するかを明示的に表現する手段ではありません．

それを理解するには自分がプログラムを作る立場になって，そのプログラムを作った人と同じイマジネーションやインスピレーションが涌かない限り，完全に理解することはできないでしょう．

よく定義されたプログラムを読み，また自らプログラムを作成する．こうした作業の繰り返しの中から経験を積むことがプログラムをよりよく理解し，またよき作成者となるための秘訣ではないでしょうか．