岡山大学経済学会雑誌28(1),1996, 73～ 90

MS-DOS 時代の思い出

藤　本　喬　雄

1　はじめに

ＰＣの世界が，Windows 95 (Microsoft Corp.) によってついに，MS-DOS (Microsoft Corp.) から離陸しようとしている。Intel の8080上で，ＣＰ/Mを使い，それから8086上で，MS-DOS ver.1.25, 2.11 へと移行していった典型的な日本俗衆の一人として，MS-DOS，あるいはその上で走ったソフトに関する思い出を書き残しておきたい。感傷的な要素は少しもない。ただ，人生の莫大な時間を消費（浪費？）したので，せめてメモ程度は残しておきたいのである。また，小さい経済モデルをシミュレーションする場合，パラメータの変更のみならず，次々とプログラムをもいじりたいときが，多々ある。MS-DOS は，このようなとき手軽でよい。Windows の Interface は，使いやすく見栄えがするが，そのために余計な気配りが必要である。(もっとも，この問題は，自分のスタイルを完成させてしまえば，後は慣れの進行で解決できる。) Multi-Task と，Event-Driven のために，自分が本来分析したいプログラム部分は，たとえ小さくても，compile，link に，より多くの時間がかかる。

授業においても，はでな Windows に気を引かれる多くの学生の中で，軽小迅速な MS-DOS 上のプログラミングを志向するものが，ここ２，３年はいても不自然ではない。このメモは，こういう学生向けの注意事項として読んでもらいたい。

2　リダイレクトによるデバッグ

デバッグの最初の部分（フローが解析できた部分）に関する一連の命令をファイルにしておいて，それをレダイレクトして焦点の場所まで自動的に進行させることは，よくやった。この方法の問題点は，繰り返しているうちに内部 FCB 番号が，段々大きくなっていくことである。そして，キー入力を INT 21h (AH＝06h, DL＝FFh) で行おうとすると，ハングアップとなる。これにあてはまる重要なソフトは，Turbo PASCAL ver. 3.0 (Borland International Inc.) 本体，およびそれが作成する COM ファイルである。この問題は次のように解決できる。すなわち，デバッグファイルの先頭で，PSP ＋ 18h のアドレスに，01h を入れてやるのである。ファイルハンドル 0 の内部 FCB 番号を強制的に，01h にするわけである。

3　EXEPACK. EXE

旧いプログラムで，EXEPACK. EXE (Microsoft Corp.) で圧縮されたものは，マシンが速い場合，ハングすることがある。“Packed file is corrupt.” のエラーがでる。このエラーは，圧縮されているものを改造したときにもでるので注意。解決策は UNPACK. COM で，元に戻す。

圧縮ソフトは何種類かあるが，不明の場合，次のようにして元に戻せる。デバッグで追跡して，元のプログラムの先頭に来たとき，適当な大きさで全体を保存する。同じことをマウス・ドライバを常駐させた後，もう一度行う。ファイル比較して，一定の数だけ異なっているセグメントの値の所を抽出する。適当な大きさのリロケーション・テーブルを含むプログラム・ヘッダを作成する。筆者は，以上のことを半自動的に行うプログラムを，PASCAL で書いた。

4 グラフィックス画面が3個に分割

NEC の PC-9800 機で，GDC を 5MHz にしている場合，グラフィックス画面が小型化，3個程に分解してしまうソフトがあった。GDC を 2.5MHz にするのは，もったいない。次のようにすれば，うまく行った。OUT Port A2h に，まず，70h を送る。次に，OUT Port A0h に，00h を3回送る。それから，やはり A0h に，59h を送る。この問題は，TEX の DVI2.EXE や Power C（Mix Software：日本語版）で生じた。

5 ＳＩ，ＤＩレジスタが，FFFFh で演算するとハング

80486系の石で，ＳＩ，ＤＩが，FFFFh の状態である演算を実行するとハングするという経験をした。Power C (Mix Software) が作成する EXE ファイルに，このような演算を行うルーティンが，リンカーから埋め込まれていた。解決策は，プログラムの改造しかない。セグメント・レジスタＤＳ，ＥＳを 100h 加算して，ＳＩ，ＤＩが，EFFFh でよいように，リンカ内部を書き換えた。

この問題が，80846系に普遍的なものかどうかは，わからない。

6 Turbo PASCAL ver. 5.0 までの一問題

Turbo PASCAL ver. 4.0 と ver. 5.0 (Borland International Inc.) では，コンパイラ指令に，{＄Ｎ＋} というのがあり，数値演算コプロセッサがあれば，extended という型（10バイト）の浮動点小数計算ができることになっている。しかし，SIN，COS，SQRT などの関数は，real 型（6バイト）のままである。よって，三角関数などを使った場合，精度は芳しくない。この問題は，Turbo PASCAL ver. 6.0，Turbo PASCAL for Windows ver. 1.0 で

は，解決されていた。

7　Microsoft C ver. 5.1 までの一問題

Microsoft C ver.5.1 までの，オプティマイザは，（ｉ＊（ｊ＊ｒ））という乗算の括弧をはずして，ｉ＊ｊ＊ｒにしてしまう。ｉ，ｊが整数型で，ｒが実数型のとき，後者の順序では，ｉ＊ｊが負数になってしまうことがある。プログラマが付けた括弧をはずすことは，オプティマイザとはいえタブーである。この問題は，ver.6.0 では解決されていた。デバッグするまえに，まずオプティマイザをはずせ。

8　MS-DOS の version チェック

旧いプログラムで，MS-DOS の version を調べた後，“DOS のバージョンがちがいます。”というエラー・メッセージを出して終了してしまうものがある。MS-DOS ver. 3.x 以降を使ってでもである。これは，MS-DOS の version が，2かどうか調べて，等しくなければ強制終了するものである。すぐに，ver. 3.x などが出てくると，予想していなかったのであろう。解決策は，2より小さければ終了するというように改造することである。ジャンプ命令の1バイトを書き換えるだけである。この問題は，例えば，Pop Up Menu (ASCII) やEGR98 (Canopus) にあった。

9　旧い FORMAT. COM の改造

MS-DOS ver. 2. 11 にある FORMAT. COM (1988-10) を改造した。FORMAT されるドライブに，カレント・ドライブにあるディスク上の IO. SYS，MSDOS. SYS，COMMAND. COM をコピーするようにした。

MS-DOS ver. 2. 11 から ver. 5.0 まで使える。ワーク・エリアが狭かったので，IO. SYS，MSDOS. SYS，COMMAND. COM 全て別々のセグメントに読み込むようにした。FCB+18h の情報が各バージョンで異なることが，わかった。MS-DOS ver. 5.0 では，IO. SYS が，64KB を越えているが，64KB に削っても問題はない。

10 旧い XCOPY の改造

MS-DOS ver. 3.1 にある XCOPY. EXE を改造した。MS-DOS ver. 2. 11 でも使えるようにした。ver. 3.1 では，subdirectory がなければ作りながらコピーしてくれるが，ver. 2.11 では，INT 21h (AH=3Dh) の機能が落ちるので一工夫必要であった。改造したものは，MS-DOS ver. 2. 11 から ver. 5.0 まで使える。

11 Zortech C++ ver. 1.06 の浮動小数点計算

マイクロソフトウェア・アソシエイツが販売した Zortech C++ (Zortech Limited) の浮動小数点計算は，コントロール・ワードに不適切なものを書き込んでいたので，有効桁数が 6 桁ぐらいに落ちていた。正しい値を書き込むようにライブラリを書き換えた。旧い計量経済学ソフトで，数値演算コプロセッサを用いているうちにハングする場合，正しい値をコントロール・ワードに書き込んでやれば，正常に走るようになるものがある。

12 グラフィックス・ライブラリの作成

いろいろな旧い言語ソフトに，自己流のグラフィックス・ライブラリをアセンブラで作成した。全て，NEC の PC-9800 用である。Turbo PASCAL

ver. 3.0, Zortech C++, Lattice C (Lattice Inc.), MS-Fortran などである。サウンド関係のルーティンも入れたりした。

旧いグラフィックス・ソフトが図形塗りつぶしでハングするのは，GPAINT 1 と GPAINT 2 で，ワーク・エリアが狭すぎること，およびワーク・エリアの先頭アドレスが小さすぎる場合に起こる。適当な値に書き換えてやればよい。

付録1に Zortech C++ 用のアセンブラ・リストを載せておく。付録2は使用例のCプログラムである。両者とも，MS-C で変更なしで使える。

13 そ の 他

IO. SYS や MSDOS. SYS を改造して，いろいろな実験をした。GRPH+のキー操作を（ビープ音なしで）有効にしたり，CTRL+function key の機能も変更した。背景色・背景のパターン，ボーダー色を変えるプログラムは人並みに作成した。

年月日を，1900年から2000年まで表せるようにも改造した。これは，COMMAND. COM も変更する必要があった。また，各ＡＰも該当個所を改造する必要があり，使用中止にした。

市販 FORTRAN 用のライブラリを，C用に改造したこともあった。根気がいった。

14 システム構成

思い出記念に，私のシステム構成を書くことを許されたい。本体は，

PC-9801RX (NEC) + IBM486SLC2 (+487)

ハードは，ICM Cb-200 (MB)，ディスプレイは，Sharp の CU-14FD であるが，ＩＯデータ機器の GA-1024 で，640×480にしている。増設メモリは，16

MBで，ＩＯデータ機器の PC34R (8MB) ＋ EX34F (8MB) である。ＣＤは，メルコの２倍速 CDO-E ＋ SRB-G (Sound) である。プリンタは，Canon の BJ-10v となっている。3.5"ディスク，特に，1.44MB のフロッピを読むために，九十九電機の TS-3NR を付けている。マウスは，NEC の PC-9872R である。

では，MS-DOS がもうしばらくのあいだ活躍せんことを！

付録１　Zortech C++(ver.1.06)用グラフィックス・ライブラリ

```
CGROUP    GROUP          _TEXT
; Takao Fujimoto: for Zortech C++
;  graph.h = prototypes.
; 1991-3-10(Sun):Yashima
;*_sound added: TF: 1991-5-5(Sun).
;
; A>MASM /mx GRAPH;  アセンブル
;
  PUBLIC _ginit,_gscreen,_gcolor
  PUBLIC _gcolor2,_gcls,_gclr
  PUBLIC _gpset,_setpat
  PUBLIC _circle,_gput2,_gwrite
  PUBLIC _point,_line,_gpaint
  PUBLIC _gpaint2,_gget,_gput1
  PUBLIC _dline
  PUBLIC _tclr,_tcls,_gotoxy
  PUBLIC _hitkey
  PUBLIC _sound
  PUBLIC _csreg,_dsreg
  PUBLIC _esreg,_ssreg
  PUBLIC _rindex,_intcall
  PUBLIC _smove,_movedata
  PUBLIC _cursoff, _curson
  PUBLIC _sysloff, _syslon

LIO_SEG SEGMENT PUBLIC 'CODE'
        DB  1400h dup(?)
        DB  '*** Takao Fujimoto :'
        DB  '1991-3-10(Sun):'
        DB  'Yashima ***'
LIO_SEG ENDS
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
        ASSUME CS:CGROUP,DS:CGROUP
TF      PROC    near
_ginit:
        push    bp
        push    di
        push    si
        push    ds
        push    es
        xor     ax,ax
        mov     es,ax
        mov     ax,0F990h
        mov     ds,ax
        mov     si,6
        mov     di,0A0h*4
        mov     cx,16
        cld
v_set:
        movsw
        stosw
        add     si,2
        loop    v_set
        mov     di,0C5h*4
        lea     ax,INT_C5
        mov     es:[di],ax
        mov     es:[di+2],cs
        mov     ax,LIO_SEG
        mov     ds,ax
        assume  ds:LIO_SEG
        INT     0A0h
        pop     es
        pop     ds
        pop     si
        pop     di
        pop     bp
        RET
INT_C5: IRET
_gscreen:
        push  bp
        mov   bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  al,[bp+4]
        mov  ds:[0000],AL
        mov  al,[bp+6]
        mov  ds:[0001],al
        mov  byte ptr ds:[0002],00
        mov  byte ptr ds:[0003],01
        INT  0A1h
        mov  byte ptr ds:[0001],00
        mov  byte ptr ds:[0002],01
        mov  byte ptr ds:[0003],07
        ; 16 colours
        mov  byte ptr ds:[0004],02
        INT  0A3h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gcolor:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  al,[bp+4]
```

```
        mov  ds:[0001],AL
        mov  al,[bp+6]
        mov  ds:[0002],al
        mov  al,[bp+8]
        mov  ds:[0003],al
        ; 16 colours
        mov  byte ptr ds:[0004],02
        INT  0A3h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gcolor2:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  al,[bp+4]
        mov  ds:[0000],al
        mov  ax,[bp+6]
        ; density 0x0GRB.
        mov  ds:[0001],ax
        INT  0A4h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gclr:
_gcls:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        INT  0A5h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gpset:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0000],AX
        mov  ax,[bp+6]
        mov  ds:[0002],ax
        mov  al,[bp+8]
        mov  ds:[0004],al
        INT  0A6h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_setpat:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0100h],ax
        mov  ds:[0102h],es
        ; length
        mov  al,[bp+6]
        mov  ds:[00104h],AL
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_circle:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
```

```
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0000],Ax
        mov  ax,[bp+6]
        mov  ds:[0002],Ax
        mov  ax,[bp+8]
        mov  ds:[0004],Ax
        mov  ds:[0006],ax
        ; mode
        mov  al,[bp+12]
        cmp  al,03h
        jnz  l1
        mov  ax,ds:[0100h]
        mov  ds:[0013h],ax
        mov  ax,ds:[0102h]
        mov  ds:[0015h],ax
        mov  al,ds:[0104h]
        mov  ds:[0012h],al
        mov  al,60h
        jmp  l3
l1:
        cmp  al,00h
        jz   l2
        mov  al,20h
        jmp  l3
l2:
        mov  al,00h
l3:
        mov  ds:[0009],AL
        ; colour
        mov  al,[bp+10]
        mov  ds:[0008],AL
       ; painting colour of the incide
        mov  ds:[0012h],AL
        INT  0A8h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gpaint:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0000],Ax
        mov  ax,[bp+6]
        mov  ds:[0002],Ax
        mov  al,[bp+8]
        ; color for painting
        mov  ds:[0004],AL
        mov  al,[bp+10]
        ; color of the border
        mov  ds:[0005],AL
           ; work area: top
      mov  word ptr ds:[0008],0A40h
           ; work area: tail
      mov  word ptr ds:[0006],0FFFh
        INT  0A9h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_gpaint2:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0000],Ax
        mov  ax,[bp+6]
        mov  ds:[0002],Ax
        mov  ax,[bp+8]
        ; color of the border
        mov  ds:[000Ah],AL

        mov  ax,ds:[0100h]
        mov  ds:[0006h],ax
        mov  ax,ds:[0102h]
        mov  ds:[0008h],ax
        mov  al,ds:[0104h]
        mov  ds:[0005h],al

        ; work area: top
      mov  word ptr ds:[0012h],0A40h
        ; work area: tail
      mov  word ptr ds:[0010h],0FFFh
        INT  0AAh
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
```

```
_gput2:
     push bp
     mov  bp,sp
     push di
     push si
     push ds
     push es
     push ds
     pop  es
     mov  ax,LIO_SEG
     mov  ds,ax
     assume  ds:LIO_SEG
     xor  ax,ax
     mov  bx,ax
     mov  ax,[bp+4]
     mov  ds:[0000],Ax
     mov  ax,[bp+6]
     mov  ds:[0002],Ax
     mov  ax,[bp+8]
     mov  ds:[0004],Ax
     ; mode
     mov  al,[bp+10]
     mov  ds:[0006],AL
     ; colour
     mov  byte ptr ds:[0007],01
     ; colour
     mov  al,[bp+12]
     mov  ds:[0008],AL
     mov  al,[bp+14]
     mov  ds:[0009],AL
     INT  0ADh
     pop  es
     pop  ds
     pop  si
     pop  di
     pop  bp
     RET
_gwrite:
     push bp
     mov  bp,sp
     push di
     push si
     push ds
     push es
     push cx

     push ds
     pop  es
     mov  ax,LIO_SEG
     mov  ds,ax
     assume  ds:LIO_SEG
     xor  ax,ax
     mov  bx,ax
     mov  ax,[bp+4]
     mov  ds:[0000],Ax
     mov  ax,[bp+6]
     mov  ds:[0002],Ax

     ; mode
     mov  al,[bp+10]
     mov  ds:[0006],AL
     ; colour
     mov  byte ptr ds:[0007],01
     ; colour
     mov  al,[bp+12]
     mov  ds:[0008],AL
     mov  al,[bp+14]
     mov  ds:[0009],AL

     mov  si,[bp+8]

nc:
     ; ES: segment prefix
     db   26h
     lodsw
     cmp  al,00h
     jnz  n_nc
     jmp  nce
n_nc:
    ; Shift JIS code ==> JIS code
    ;  AH=BF, AL=8A 漢.
     cmp  al,80h
     jnb  nc_z0
     jmp  nc_h
nc_z0:
     cmp  al,0E0h
     jnb  nc_z
     cmp  al,0A0h
     jb   nc_z
nc_h:
        ; flag 半角
     mov  byte ptr ds:[0010h],01h
     dec  si
     cmp  al,80h
     ja   nc_h_1
     ; xchg の必要なし.
     mov  ah,29h
     jmp  nc7
nc_h_1:
     mov  ah,2Ah ; ｶﾀｶﾅ
     jmp  nc7
nc_z:
       ; flag 全角
   mov  byte ptr ds:[0010h],00h
     cmp  al,0a0h
     jnb  nc2
     sub  al,70h
     jmp  nc3
nc2:
     cmp  al,0F0h
     jnb  nc1
     sub  al,0B0h
nc3:
     or   ah,ah
     jns  nc4
     dec  ah
nc4:
     add  al,al
     cmp  ah,9Eh
     jb   nc5
```

```
        sub  ah,5Eh
        jmp  nc6
nc5:
        dec  al
nc6:
        sub  ah,1Fh
        xchg ah,al
nc7:
        mov  ds:[0004],Ax
        push es
        push si
        push cx

wa1:    in   al,0A0h
        test al,04h
        jz   wa1
wa2:    in   al,0A0h
        test al,08h
        jnz  wa2

        INT  0ADh
        pop  cx
        pop  si
        pop  es
ncl:
        mov  di,0
         cmp  byte ptr ds:[0010h],00h
        jnz  ncl_h
        mov  ax,0010h
        jmp  ncl_z
ncl_h:
        mov  ax,0008h
ncl_z:
        add  [di],ax
        jmp  nc
nce:
        pop  cx
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_point:
        RET
_line:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
        pop  es
        mov  ax,LIO_SEG
        mov  ds,ax
        assume  ds:LIO_SEG
        xor  ax,ax
        mov  bx,ax
        mov  ax,[bp+4]
        mov  ds:[0000],Ax
        mov  ax,[bp+6]
        mov  ds:[0002],Ax
        mov  ax,[bp+8]
        mov  ds:[0004],Ax
        mov  ax,[bp+10]
        mov  ds:[0006],Ax
        ; mode
        mov  al,[bp+14]

        cmp  al,03h
        jnz  11L
        mov  ax,ds:[0100h]
        mov  ds:[000Eh],ax
        mov  ax,ds:[0102h]
        mov  ds:[0010h],ax
        mov  al,ds:[0104h]
        ; length
        mov  ds:[000Dh],al
        mov  byte ptr ds:[0009],02
        mov  byte ptr ds:[000Ah],02
        jmp  14L
11L:
        cmp  al,02h
        jnz  12L
        mov  byte ptr ds:[0009],02
        mov  byte ptr ds:[000Ah],01
        jmp  14L
12L:
        cmp  al,01
        jnz  13L
        mov  byte ptr ds:[0009],01
        mov  byte ptr ds:[000Ah],00
        jmp  14L
13L:
        mov  byte ptr ds:[0009],00
        mov  byte ptr ds:[000Ah],00

14L:
        ; colour
        mov  al,[bp+12]
        mov  ds:[0008],AL
; painting colour of the incide
        mov  ds:[000Bh],AL
        INT  0A7h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET

_dline:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push ds
```

```
     pop  es
     mov  ax,LIO_SEG
     mov  ds,ax
     assume  ds:LIO_SEG
     xor  ax,ax
     mov  bx,ax
     mov  ax,[bp+4]
     mov  ds:[0000],Ax
     mov  ax,[bp+6]
     mov  ds:[0002],Ax
     mov  ax,[bp+8]
     mov  ds:[0004],Ax
     mov  ax,[bp+10]
     mov  ds:[0006],Ax
     ; mode
     mov  al,[bp+14]

     cmp  al,01
     jnz  13LD
     mov  byte ptr ds:[0009],01
     mov  byte ptr ds:[000Ah],01
     jmp  14LD
13LD:
     mov  byte ptr ds:[0009],00
     mov  byte ptr ds:[000Ah],01

14LD:
     ; colour
     mov  al,[bp+12]
     mov  ds:[0008],AL
     ; Line style
     mov  ds:word ptr[000Bh],0AAAAh
     INT  0A7h
     pop  es
     pop  ds
     pop  si
     pop  di
     pop  bp
     RET

_gget:
     push bp
     mov  bp,sp
     push di
     push si
     push ds
     push es
     push ds
     pop  es
     mov  ax,LIO_SEG
     mov  ds,ax
     assume  ds:LIO_SEG
     xor  ax,ax
     mov  bx,ax
     mov  ax,[bp+4]
     mov  ds:[0000],Ax
     mov  ax,[bp+6]
     mov  ds:[0002],Ax
     mov  ax,[bp+8]
     mov  ds:[0004],Ax
     mov  ax,[bp+10]
     mov  ds:[0006],Ax
     ; offset of the array
     mov  ax,[bp+12]
     mov  ds:[0008],Ax
     ; segment
     mov  ds:[000Ah],es
     mov  ax,[bp+14]
     ; length
     mov  ds:[000Ch],Ax

     INT  0ABh
     pop  es
     pop  ds
     pop  si
     pop  di
     pop  bp
     RET

_gput1:
     push bp
     mov  bp,sp
     push di
     push si
     push ds
     push es
     push ds
     pop  es
     mov  ax,LIO_SEG
     mov  ds,ax
     assume  ds:LIO_SEG
     xor  ax,ax
     mov  bx,ax
     mov  ax,[bp+4]
     mov  ds:[0000],Ax
     mov  ax,[bp+6]
     mov  ds:[0002],Ax
     ; offset of the array
     mov  ax,[bp+8]
     mov  ds:[0004],Ax
     ; segment
     mov  ds:[0006],es
     mov  ax,[bp+10]
     ; length
     mov  ds:[0008],Ax
     ; mode = set
    mov  byte ptr ds:[000Ah],00h
     ; color switch
    mov  byte ptr ds:[000Bh],00h

     INT  0ACh
     pop  es
     pop  ds
     pop  si
     pop  di
     pop  bp
     RET

_tclr:
_tcls:
```

```
        push bp
        push di
        push si
        push ds
        push es
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset esc1
        mov  ah,09h
        int  21h
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
esc1    db   1Bh,'[2J$'
        RET
_gotoxy:
        push bp
        mov  bp,sp
        push di
        push si
        push cx
        push dx
        push ds
        push es
        mov  dl,[bp+4]
        mov  dh,[bp+6]
        mov  ah,03h
        mov  cl,10h
        int  0DCh
        pop  es
        pop  ds
        pop  dx
        pop  cx
        pop  si
        pop  di
        pop  bp
        RET
_sound:
        push bp
        mov  bp,sp
        push di
        push si
        push ds
        push es
        push cx
        push dx
        mov  ax,[bp+4]
        mov  cx,[bp+6]
        mov  dx,03FDBh
        out  dx,al
        xchg ah,al
        out  dx,al
        mov  al,06
        out  37h,al
sol:
        push cx
        mov  cx,0400h
so2:
        loop so2
        pop  cx
        loop sol
        mov  al,07
        out  37h,al
        pop  dx
        pop  cx
        pop  es
        pop  ds
        pop  si
        pop  di
        pop  bp
        RET
_hitkey:
        push bp
        push ds
        push es
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset mes1
        mov  ah,09h
        int  21h
           ; cursor normalized
        mov  al,4Bh
        out  62h,al
        mov  al,8Fh
        out  60h,al
        mov  al,00h
        out  60h,al
        mov  al,7Bh
        out  60h,al
        mov  ah,00h
        int  18h
        pop  es
        pop  ds
        pop  bp
        RET
mes1    db   '*** ',1Bh,'[43;5m
        db   1Bh,'[>5l'
        db   'Hit any key !'
        db   1Bh,'[m***','$'
_intcall:
        push bp
        mov  bp,sp
        push ds
        push es
        push si
        push di
        mov  ax,[bp+4]
        mov  di,offset ic1
        inc  di
        push cs
        pop  es
        stosb
        mov  bx,[bp+6]
        mov  ax,[bx]
        mov  cx,[bx+4]
```

```
        mov  dx,[bx+6]
        mov  si,[bx+8]
        mov  di,[bx+0Ah]
        mov  es,[bx+0Eh]
        push ds
        mov  ds,[bx+0Ch]
        push bp
        mov  bp,bx
        mov  bx,[bp+02]
        mov  bp,[bp+10h]
        cli
ic1:
        int  21h
        sti
        pop  bp
        pop  ds
        mov  bx,[bp+8]
        mov  [bx],ax
        lahf
        xchg ah,al
        mov  [bx+12h],ax
        mov  [bx+4],cx
        mov  [bx+6],dx
        mov  [bx+8],si
        mov  [bx+0Ah],di
        mov  [bx+0Ch],ds
        mov  [bx+0Eh],es
        mov  [bx+2],bx
        pop  di
        pop  si
        pop  es
        pop  ds
        mov  sp,bp
        pop  bp
        ret
_rindex:
        push si
        push di
        push bp
        mov  bp,sp
        sub  sp,02
        mov  si,[BP+08h]
        mov  di,[BP+0Ah]
        mov  [BP-02],SI
ri1:
        mov  bx,si
        inc  si
        cmp  byte ptr[bx],00
        jnz  ri1
ri2:
        mov  ax,[bp-02]
        cmp  ax,si
        jnb  ri3
        dec  si
        mov  bx,si
        mov  al,[bx]
        sub  ah,ah
        mov  cx,ax
        cmp  di,cx
        jnz  ri2
        mov  ax,si
        jmp  ri4
ri3:
        sub  ax,ax
ri4:
        mov  sp,bp
        pop  bp
        pop  di
        pop  si
        ret
_smove:
        push bp
        mov  bp,sp
        push   es
        push di
        push si
        push cx
        push ds
        pop  es
        mov  di,[bp+4]
        mov  si,[bp+6]
        mov  cx,[bp+8]
        repz movsb
        pop  cx
        pop  si
        pop  di
        pop  es
        mov  sp,bp
        pop  bp
        ret
_movedata:
        push bp
        mov  bp,sp
        push ds
        push es
        push di
        push si
        push cx
        mov  ds,[bp+4]
        mov  si,[bp+6]
        mov  es,[bp+8]
        mov  di,[bp+10]
        mov  cx,[bp+12]
        repz movsb
        pop  cx
        pop  si
        pop  di
        pop  es
        pop  ds
        mov  sp,bp
        pop  bp
        ret
_csreg:
        mov  ax,cs
        ret
_dsreg:
        mov  ax,ds
        ret
_esreg:
        mov  ax,es
```

```
        ret
_ssreg:
        mov  ax,ss
        ret
_cursoff:
        push bp
        push ds
        push dx
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset cmes0
        mov  ah,09h
        int  21h
        pop  dx
        pop  ds
        pop  bp
        ret
cmes0:
        db   1Bh,'[>5h$','$'
_curson:
        push bp
        push ds
        push dx
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset cmes1
        mov  ah,09h
        int  21h
        pop  dx
        pop  ds
        pop  bp
        ret
cmes1:
        db   1Bh,'[>5l$','$'
_sysloff:
        push bp
        push ds
        push dx
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset smes0
        mov  ah,09h
        int  21h
        pop  dx
        pop  ds
        pop  bp
        ret
smes0:
        db   1Bh,'[>1h$','$'
_syslon:
        push bp
        push ds
        push dx
        push cs
        pop  ds
        assume  ds:CGROUP
        mov  dx,offset smes1
        mov  ah,09h
        int  21h
        pop  dx
        pop  ds
        pop  bp
        ret
smes1:
        db   1Bh,'[>1l$','$
TF      ENDP
_TEXT   ENDS
        END
```

付録 2　グラフィックス・ライブラリ使用例

```
/* GTEST.C for Zortech C++ v.1.06. */
/* (adapted from the one for Pro-Fortran 77) */
/*  TF: 1991-7-7 */
# include <stdio.h>
# include <graph.h>
# include <process.h>

main(){
/* Variables for SOUND below */
       int do1,re,mi,fa,so,la,si,do2,dur;
/* ************************ */

 char *PAT;
 int  iarray[500];
 int i;
 unsigned kanjic;
       PAT="¥xaa¥x00¥x00¥xFF"; /* Blue, Red, Gree, Grey */
       kanjic=0x3441;
    do1=6400; re=5700; mi=5000; fa=4700;
    so=4200; la=3800; si=3400; do2=3200;

       ginit();            /* 必要 */
       gscreen(3,0);       /* 必要 */
       gcolor(0,0,7);      /*  back, border, fore */
       gcolor2(0,0x0060); /* palette 0..15 ==> density 0x0GRB. */
        tclr();            /* テキスト画面をクリア */
       gclr();             /* グラフィックス画面をクリア */

       gpset(50,351,2);    /* 点を描く */ /*  x, y, color */
       gpset(50,352,2);
       gpset(50,353,2);

/*             *** Set a pattern for CIRCLE *** */
       setpat(PAT,4);  /* 円 */ /* x, y, radius, color, mode */
       circle(100, 100, 100, 4, 3); /* ﾀｲﾙ     */
       circle(550, 300,  50, 9, 2); /* filled */
       circle(320, 199, 180, 6, 0);

       gwrite(300,160,"文字列を GWRITE で",0,5,10);
            /* x, y, code, mode, fore_colour, back_colour */
       gput2(340, 180, 0x3441, 0, 6, 9);
       gput2(360, 180, 0x3441, 0, 5, 10);
       gput2(460, 260, kanjic, 2, 6, 15);

       PAT="¥x00¥x33¥xDD¥x00";
       setpat(PAT,4);
          /* 直線 */ /* x1, y1, x2, y2, color, mode */
       line(400, 50, 500, 150, 4, 0);  /* 0=line */
       line(500, 150, 550, 200, 2, 1); /* 1=box 箱      */
       line(550, 200, 600, 250, 14, 2);/* 2=box filled */
       line(600, 250, 639, 289, 7, 3); /* 3=box tiled  */
       line(550, 10, 600, 60, 4, 2);
       line(600, 60, 639, 110,12, 2);

       gpaint(520,170,6,2); /* x, y, paint color, border color */

       PAT="¥x10¥xA3¥x2D¥x01¥x12¥x02";
       setpat(PAT,6);
```

```
        line(50,250,250,350, 2, 1); /* 1=box  */
    gpaint2(100,300,2); /* x, y, border color: Tile-Pattern used.*/

        gget(300,180,349,209,(int)iarray,850);
             /* x1,y1,x2,y2,array,length */
        gput1(100,300,(int)iarray,850);
             /* x,y,array(offset),length(ｵｵｷﾒﾆ) */

        gotoxy(30,0);
        hitkey();
/*************************************************/
/* SOUND(freq, duration);                        */
/*    do ﾊ ﾂｶｴﾅｲ. reserved.                       */
/*   frequencyの逆数 = 1ｵｸﾀｰﾌﾞ低 = 全て 2 倍     */
/*       = 1ｵｸﾀｰﾌﾞ高 = 全て 2 分の 1              */
/*                   = 100 = ｷｺｴﾅｲ             */
/*   duration=300= 300/1000 seconds on RX(12MHz). */
/*************************************************/
      dur=300;
      sound(do1,dur);
      sound(re,dur);
      sound(mi,dur);
      sound(fa,dur);
      sound(so,dur);
      sound(la,dur);
      sound(si,dur);
      sound(do2,dur*2);
gcolor2(0,0);
gclr();
    spawnl(0,"c16bb.com","c16bb.com",0);
    system("dir");
}
```