BASIC
Language Reference Booklet

RT PC Graphics Development Toolkit

Programming Family

Personal
Computer
Software

59X8615

IBM RT PC BASIC Language Reference Booklet

This reference booklet describes the IBM RT PC BASIC syntax required to write an application program using the Graphics Development Toolkit. Any general or special considerations for the BASIC language are described in this booklet. It is important that you keep this booklet in a safe place. This booklet is the only source of information that specifically describes the IBM RT PC BASIC language interface to the Toolkit.

Only native mode of the IBM RT PC BASIC language is supported by the Graphics Development Toolkit.

Note: The Graphics Development Toolkit, as shipped, works only with the IBM RT PC BASIC Compiler, not the BASIC Interpreter. See the *IBM RT PC BASIC Language Handbook* for binding routines to the Interpreter.

Writing in BASIC

Writing in the BASIC language requires a few special considerations. They are as follows:

- **Array numbering.** Arrays are numbered from 0 to n. When calling a Toolkit routine with an array as a parameter, you must reference the first element in an array as **VARPTR(array%(0)).** If OPTION BASE 1 is set, you must reference the first element in an array as **VARPTR(array%(1)).** For more information on OPTION BASE see *IBM RT PC BASIC Language Reference.*

  For some routines the size of an array for one parameter is based on the value of another parameter. This is shown in the "Data Types:" by using the parameter name in the array size. This notation is used only to show the relationship between the parameters and does not imply actual coded values or refer to storage allocation.

- **Data Types.** In this booklet, variable names followed by a % are signed 32-bit integers and variable names followed by a $ are 8-bit character strings. You need not dimension (DIM) character strings.

  CHANGE must be used to convert an integer array to a BASIC character string. **VARPTR** must be used to pass a pointer to an array, and **CCHAR** converts a string to the proper form for passing to the Toolkit routine.

If your application program receives a character or character string from a Toolkit routine, you must have the following lines in that program:

```
DIM s%(n)
funct%(VARPTR(s%(0))...)
CHANGE s%() TO s$
```

(where s$ is the name of a character string)

Note: You must DIMension s%() to be (n/4)+1, where n is the number of characters.

If your application program sends a character string to some Toolkit routine, you must have either the following two lines in that program:

```
s$="xxxxxx"
status%=vname%(CCHAR(s$,s%()))
```

(where s$ is the name of a character string and xxxxxx is the string itself) or one line as follows:

```
status%=vname%(CCHAR("xxxxxx",s%())
```

The include statement for the file **extrnvdi.bas**

```
REM $INCLUDE: "extrnvdi"
```

must appear in each program which uses Toolkit routines. This resolves external reference calls to the routines. If there is not a copy of **extrnvdi.bas** in the directory with the program, then specify the pathname **/usr/include/extrnvdi**.

- **Routine Calls.** The call to a BASIC Toolkit routine takes the following form:

  status% = vname%(a,b,c)

  Where: status = status returned
  vname = subroutine name
  a, b, and c = parameters

  All Toolkit routines return an integer value that is the status assigned by the routine. Unless otherwise indicated, a value of zero indicates successful completion and a value of minus one indicates an error has occurred.

- **Compiling and Linking.** Use the following command line to compile your BASIC application program and link it to the Toolkit library of subroutines:

```
basicnc filename -n --
 /usr/lpp/vdi/lib/basvdi.a
```

Programming Considerations

In a situation where both Toolkit routines and BASIC statements are available to perform the same operations, use the Toolkit routine to ensure you get correct results. For example:

- **Clear Screen.** Do not use the BASIC Clear Screen Command CLS to clear the DISPLAY device. Use the Clear Workstation routine instead.

- **Screen Scroll.** Do not allow BASIC to scroll the DISPLAY device with the BASIC PRINT command. Use Output Alpha Text or Output Graphic Text instead.

BASIC Language Syntax

If the number of parameters passed to a routine is not the same as the number of parameters expected by that routine, a compiler error will occur.

A double asterisk (**) following a generic function name indicates that the routine is device-dependent. Using these routines in an application program makes that application device-dependent.

If a parameter is not shown in "Data Types:" that parameter is an integer.

Throughout the "Routines" section of this booklet, input parameters are italicized and output parameters are shown in regular type.

Application Data**
status% = vappl%
(*handle%, CCHAR(functn$, functn%()), datcnt%, VARPTR(appdat%(0))*)

Data Types: DIM appdat%(n)

Clear Workstation
status% = vclrwk%
(*handle%*)

Close Workstation
status% = vclswk%
(*handle%*)

Copy Page**
status% = vcpage%
(*handle%, source%, destination%*)

Copy Pels**
status% = vcppel%
(*handle%, VARPTR(xy%(0))*)

Data Types: DIM xy%(6)

Cursor Down**
status% = vcurdn%
(*handle%*)

Cursor Home**
status% = vcurhm%
(*handle%*)

Cursor Left**
status% = vcurlf%
(*handle%*)

Cursor Right**
status% = vcurrt%
(*handle%*)

Cursor Up**
status% = vcurup%
(*handle%*)

Direct Cursor Address**
status% = vcurad%
(*handle%, row%, column%*)

Display Graphic Input Cursor**

status% = vdspcr%
(*handle%, x%, y%*)

Enter Cursor Addressing Mode**

status% = vencur%
(*handle%*)

Erase to End of Line**

status% = vereol%
(*handle%*)

Erase to End of Screen**

status% = vereos%
(*handle%*)

Exit Cursor Addressing Mode**

status% = vexcur%
(*handle%*)

Get Pels**

status% = vgtpel%
(*handle%, VARPTR(xy%(0))*, VARPTR(parray%(0)))

Data Types: DIM xy%(4)
DIM parray%(n)

Hardcopy**

status% = vhdcpy%
(*handle%*)

Input Choice (request mode)

status% = vrqchc%
(*handle%, initial.choice%*, VARPTR(final.choice%))

status% = 0 request unsuccessful
> 0 request successful
= -1 an error has occurred

Input Choice (sample mode)

status% = vsmchc%
(*handle%*, VARPTR(final.choice%))

status% = 0 sample unsuccessful
> 0 sample successful
= -1 an error has occurred

Input Locator (request mode)

```
status% = vrqloc%
    (handle%, VARPTR(xy%(0)), ink%, rubberband%,
     echo.handle%, VARPTR(final.xy%(0)),
     VARPTR(term%(0)))

    Data Types:    DIM xy%(2)
                   DIM final.xy%(2)
                   DIM term%(1)

    CHANGE term%() TO term$

    status% =  0 request unsuccessful
            > 0 request successful
           = -1 an error has occurred
```

Input Locator (sample mode)

```
status% = vsmloc%
    (handle%, VARPTR(xyin%(0)), VARPTR(xyout%(0)),
     VARPTR(pressed%(0)), VARPTR(released%(0)),
     VARPTR(key.state%(0)))

    Data Types:    DIM xyin%(2)
                   DIM xyout%(2)

    status% =  0 sample unsuccessful
            > 0 sample successful
           = -1 an error has occurred
```

Input String (request mode)

```
status% = vrqstr%
    (handle%, maximum.length%, echo.mode%,
     VARPTR(echo.xy%(0)), VARPTR(chstr%(0)))

    Data Types:    DIM echo.xy%(2)
                   DIM chstr%(n)

    CHANGE chstr%() TO chstr$

    status% =  0 request unsuccessful
            > 0 number of characters
           = -1 an error has occurred
```

Input String (sample mode)

```
status% = vsmstr%
    (handle%, maximum length, echo.mode%,
     VARPTR(echo.xy%(0)), VARPTR(chstr%(0)))

    Data Types:    DIM echo.xy%(2)
                   DIM chstr%(n)

    CHANGE chstr%() TO chstr$

    status% =  0 sample unsuccessful
            > 0 number of characters
           = -1 an error has occurred
```

Input Valuator (request mode)

status% = vrqval%
(*handle%, initial.value%, echo.handle%,*
VARPTR(final.value%))

status% = 0 request unsuccessful
> 0 request successful
= -1 an error has occurred

Input Valuator (sample mode)

status% = vsmval%
(*handle%,* VARPTR(final.value%))

status% = 0 sample unsuccessful
> 0 sample successful
= -1 an error has occurred

Inquire Addressable Character Cells

status% = vqchcl%
(*handle%,* VARPTR(rows%), VARPTR(columns%))

Inquire Alpha Text Capabilities

status% = vqacap%
(*handle%,* VARPTR(capabilities%(0)))

Data Types: DIM capabilities%(15)

Inquire Alpha Text Cell Location

status% = vqacel%
(*handle%, row%, column%,* VARPTR(proportion%),
VARPTR(x.out%), VARPTR(y.out%))

Inquire Alpha Text Font Capability

status% = vqafnt%
(*handle%, font.requested%, size.requested%,*
VARPTR(capabilities%(0)))

Data Types: DIM capabilities%(7)

status% = 0 font unavailable
> 0 font available
= -1 an error has occurred

Inquire Alpha Text Position

status% = vqapos%
(*handle%,* VARPTR(x.out%), VARPTR(y.out%))

Inquire Alpha Text String Length

status% = vqalen%
(*handle%, CCHAR(chstr$, chstr%())*)

status% ≥ 0 length
= -1 an error has occurred

Inquire Cell Array

```
status% = vqclry%
    (handle%, VARPTR(xy%(0)), row.length%,
     number.rows%, VARPTR(elements.per.row%),
     VARPTR(rows.used%), VARPTR(flag%),
     VARPTR(colors%(0)))

    Data Types:    DIM xy%(4)
                   DIM colors%(n)
```

Inquire Color Representation

```
status% = vqcolr%
    (handle%, index.requested%, set.flag%,
     VARPTR(rgb.returned%(0)))

    Data Types:    DIM rgb.returned%(3)

    status% ≥ 0 actual index selected
             = -1 an error has occurred
```

Inquire Current Cursor Text Address**

```
status% = vqcura%
    (handle%, VARPTR(row%), VARPTR(column%))
```

Inquire Current Fill Area Attributes

```
status% = vqfatt%
    (handle%, VARPTR(attributes%(0)))

    Data Types:    DIM attributes%(4)
```

Inquire Current Graphic Text Attributes

```
status% = vqtatt%
    (handle%, VARPTR(attributes%(0)))

    Data Types:    DIM attributes%(10)
```

Inquire Current Polyline Attributes

```
status% = vqlatt%
    (handle%, VARPTR(attributes%(0)))

    Data Types:    DIM attributes%(4)
```

Inquire Current Polymarker Attributes

```
status% = vqmatt%
    (handle%, VARPTR(attributes%(0)))

    Data Types:    DIM attributes%(4)
```

Inquire Cursor Text Mode**

```
status% = vqcurm%
    (handle%)

    status% ≥ 0 current mode
             = -1 an error has occurred
```

Inquire Error

status% = vqerr%()

Inquire Graphic Color Burst Mode**

status% = vqgcm%
(*handle%*)

status% ≥ 0 actual mode selected
= −1 an error has occurred

Inquire Page**

status% = vqpage%
(*handle%*, VARPTR(gr.mode%(0)),
VARPTR(cur.mode%(0)))

Data Types: DIM gr.mode%(3)
DIM cur.mode%(3)

Message**

status% = vmsg%
(*handle%*, *CCHAR(msg$, msg%()*), *wait%*)

Open Workstation

status% = vopnwk%
(*VARPTR(workin%(0))*, VARPTR(handle%),
VARPTR(workout%(0)))

Data Types: DIM workin%(19)
DIM workout%(66)

Output Alpha Text

status% = vatext%
(*handle%*, *CCHAR(chars$, chars%())*,
VARPTR(xout%), VARPTR(yout%))

Output Arc

status% = varc%
(*handle%*, *x%*, *y%*, *radius%*, *start.angle%*,
end.angle%)

Output Bar

status% = vbar%
(*handle%*, *VARPTR(xy%(0))*)

Data Types: DIM xy%(4)

Output Cell Array

status% = vclary%
(*handle%*, *VARPTR(xy%(0))*, *row.length%*,
elements.per.row%, *number.rows%*, *writing.mode%*,
VARPTR(colors%(0)))

Data Types: DIM xy%(4)
DIM colors%(n)

Output Circle

status% = vcircl%
(*handle%, x%, y%, radius%*)

Output Cursor Addressable Text**

status% = vctext%
(*handle%, CCHAR(chstr$, chstr%())*)

Output Filled Area

status% = vfarea%
(*handle%, count%, VARPTR(xy%(0))*)

Data Types: DIM xy%(2*COUNT%)

Output Graphic Text

status% = vgtext%
(*handle%, x%, y%, CCHAR(chstr$, chstr%())*)

Output Pie Slice

status% = vpiesl%
(*handle%, x%, y%, radius%, start.angle%, end.angle%*)

Output Polyline

status% = vpline%
(*handle%, count%, VARPTR(xy%(0))*)

Data Types: DIM xy%(2*COUNT%)

Output Polymarker

status% = vpmark%
(*handle%, count%, VARPTR(xy%(0))*)

Data Types: DIM xy%(2*COUNT%)

Put Pels**

status% = vptpel%
(*handle%, VARPTR(xy%(0)), VARPTR(parray%(0))*)

Data Types: DIM xy%(2)
DIM parray%(n)

Read Cursor Movement Keys**

status% = vrdcky%
(*handle%, input.mode%*, VARPTR(direction%), VARPTR(keyin%(0))

CHANGE keyin%() TO keyin$

Remove Graphic Input Cursor**

status% = vremcr%
(*handle%*)

Reverse Video Off**

status% = vrvoff%
(*handle%*)

Reverse Video On**

status% = vrvon%
(*handle%*)

Set Alpha Text Color Index

status% = vsacol%
(*handle%*, *index.requested%*)

status% ≥ 0 index selected
=-1 an error has occurred

Set Alpha Text Font and Size

status% = vsafnt%
(*handle%*, *font.requested%*, *size.requested%*,
VARPTR(capabilities%(0)))

Data Types: DIM capabilities%(8)

status% = 0 font unavailable
> 0 font selected
=-1 an error has occurred

Set Alpha Text Line Spacing

status% = vsaspc%
(*handle%*, *spacing.requested%*)

status% ≥ 0 spacing selected
=-1 an error has occurred

Set Alpha Text Overstrike Mode

status% = vsaovr%
(*handle%*, *mode.requested%*)

status% ≥ 0 mode selected
=-1 an error has occurred

Set Alpha Text Pass Through Mode

status% = vsapas%
(*handle%*, *mode.requested%*)

status% ≥ 0 mode selected
=-1 an error has occurred

Set Alpha Text Position

status% = vsapos%
(*handle%*, *x.in%*, *y.in%*, VARPTR(x.out%(0)),
VARPTR(y.out%(0)))

Set Alpha Text Quality

status% = vsaql%
(*handle%, mode.requested%*)

status% ≥ 0 mode selected
= -1 an error has occurred

Set Alpha Text Subscript Superscript Mode

status% = vsasub%
(*handle%, mode.requested%*)

status% ≥ 0 mode selected
= -1 an error has occurred

Set Alpha Text Underline Mode

status% = vsaund%
(*handle%, mode.requested%*)

status% ≥ 0 mode selected
= -1 an error has occurred

Set Background Color Index

status% = vsbcol%
(*handle%, index.requested%*)

status% ≥ 0 index selected
= -1 an error has occurred

Set Character Height

status% = vsthgt%
(*handle%, height.requested%,*
VARPTR(char.width%), VARPTR(cell.width%),
VARPTR(cell.height%))

status% ≥ 0 height selected
= -1 an error has occurred

Set Color Representation

status% = vscolr%
(*handle%, index.requested%,*
VARPTR(rgb.requested%(0)),
VARPTR(rgb.selected%(0)))

Data Types: DIM rgb.requested%(3)
DIM rgb.selected%(3)

status% ≥ 0 index selected
= -1 an error has occurred

Set Cursor Text Attributes**

status% = vcratt%
(*handle%, VARPTR(req.att%(0))*,
VARPTR(sel.att%(0)))

Data Types: DIM req.att%(4)
DIM sel.att%(4)

Set Cursor Text Color Index**

status% = vcrcol%
(*handle%, fore.requested%, back.requested%,*
VARPTR(fore.selected%),
VARPTR(back.selected%))

Set Cursor Text Mode**

status% = vscurm%
(*handle%, mode.requested%*)

status% ≥ 0 actual mode selected
=-1 an error has occurred

Set Fill Color Index

status% = vsfcol%
(*handle%, index.requested%*)

status% ≥ 0 actual index selected
=-1 an error has occurred

Set Fill Interior Style

status% = vsfint%
(*handle%, style.requested%*)

status% ≥ 0 style selected
=-1 an error has occurred

Set Fill Style Index

status% = vsfstl%
(*handle%, index.requested%*)

status% ≥ 0 index selected
=-1 an error has occurred

Set Graphic Color Burst Mode**

status% = vsgcm%
(*handle%, mode.requested%*)

status% ≥ 0 actual mode selected
=-1 an error has occurred

Set Graphic Text Alignment

status% = vstaln%
(*handle%, horizontal.requested%,*
vertical.requested%, VARPTR(horizontal.realized%),
VARPTR(vertical.realized%))

Set Graphic Text Color Index

status% = vstcol%
(*handle%, index.requested%*)

status% ≥ 0 index selected
=-1 an error has occurred

Set Graphic Text Font

status% = vstfnt%
(*handle%, font.requested%*)

status% ≥ 0 font type selected
= –1 an error has occurred

Set Graphic Text String Baseline Rotation

status% = vstrot%
(*handle%, angle.requested%*)

status% ≥ 0 angle selected
= –1 an error has occurred

Set Line Edit Characters

status% = vsedch%
(*handle%, CCHAR(line.del$, line.del%()),*
CCHAR(char.del$, char.del%()))

Set Page**

status% = vspage%
(*handle%, VARPTR(gr.in%(0)), VARPTR(cur.in%(0)),*
VARPTR(gr.out%(0)), VARPTR(cur.out%(0)))

Data Types: DIM gr.in%(2)
DIM cur.in%(2)
DIM gr.out%(2)
DIM cur.out%(2)

Set Pen Speed**

status% = vpnspd%
(*handle%, speed%*)

status% ≥ 0 actual pen speed selected
= –1 an error has occurred

Set Polyline Color Index

status% = vslcol%
(*handle%, index.requested%*)

status% ≥ 0 index selected
= –1 an error has occurred

Set Polyline Line Type

status% = vsltyp%
(*handle%, type.requested%*)

status% ≥ 0 type selected
= –1 an error has occurred

Set Polyline Line Width

status% = vslwid%
(*handle%, width.requested%*)

status% ≥ 0 width selected
= –1 an error has occurred

Set Polymarker Color Index
status% = vsmcol%
(*handle%, index.requested%*)

status% ≥ 0 index selected
= −1 an error has occurred

Set Polymarker Height
status% = vsmhgt%
(*handle%, height.requested%*)

status% ≥ 0 height selected
= −1 an error has occurred

Set Polymarker Type
status% = vsmtyp%
(*handle%, type.requested%*)

status% ≥ 0 type selected
= −1 an error has occurred

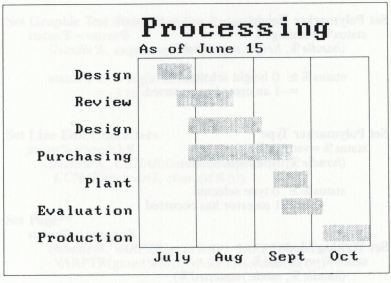
Set Writing Mode
status% = vswrmd%
(*handle%, mode.requested%*)

status% ≥ 0 actual mode selected
= −1 an error has occurred

Update Workstation
status% = vupdwk%
(*handle%*)

Program Example

This program demonstrates how to create and display a Gantt chart. The output from this program should appear as follows:




```
REM     This is a program to use the Graphics
REM     Development Toolkit BASIC binding
REM     to draw a Gantt chart
REM
REM $include: "extrnvdi"
REM
DIM workin%(18), savary%(65)
DIM xy%(10), echo.xy%(1)
DIM tasks$(6), start.dates%(6)
DIM end.dates%(6), ticks$(3)
DIM taskay%(50), tickay%(50), dummy%(2)
REM
REM     create the open device array
REM
DATA 0, 1, 1, 3, 1, 1, 1, 0, 0, 1, 1, 68, _
     73, 83, 80, 76, 65, 89, 32
FOR i = 0 TO 18 STEP 1
   READ workin%(i)
NEXT i
REM
REM     set the echo location for request work
REM
echo.xy%(0) = 0
echo.xy%(0) = 0
REM
REM     create the array of task names
REM
DATA "Production", "Evaluation", "Plant"
DATA "Purchasing", "Design", "Review", "Design"
FOR i = 0 TO 6 STEP 1 : READ tasks$(i) : NEXT i
```

```
REM
REM    create the array of start dates
REM    for the bars
REM
DATA 83, 72, 70, 48, 48, 45, 40
FOR i = 0 TO 6 STEP 1
   READ start.dates%(i)
NEXT i
REM
REM    do the same for end dates
REM
DATA 95, 83, 79, 75, 67, 60, 49
FOR i = 0 TO 6 STEP 1
   READ end.dates%(i)
NEXT i
REM
REM    create the array of dates for the
REM    horizontal axis
REM
DATA "July", "Aug", "Sept", "Oct"
FOR i = 0 TO 3 STEP 1 : READ ticks$(i) : NEXT i
REM
REM    define a function that transforms
REM    percentages into NDC units.
REM
DEF fnxtr%(a) = (a / 100.0) * savary51
DEF fnytr%(a) = (a / 100.0) * savary52
REM
REM    open the workstation
REM
status% =  vopnwk%(VARPTR(workin%(0)),_
                   VARPTR(dev.handle%), _
                   VARPTR(savary%(0)))
savary51 = savary%(51)
savary52 = savary%(52)
REM
REM    set the constants for the grid
REM
xy%(1) = fnytr%(10.0)
xy%(3) = fnytr%(80.0)
REM
REM for the points along the axis for grid lines
REM
FOR i = 50 TO 80 STEP 15
REM
REM    set the rest of the elements
REM    in the array for the grid line
REM    and draw the line
REM
  xy%(0) = fnxtr%(i)
  xy%(2) = xy%(0)
  status%= vpline%(dev.handle%,2,VARPTR(xy%(0)))
NEXT i
REM
REM    set text alignment to top center
REM
```

```
status% =  vstaln%(dev.handle%,1,2,_
                   VARPTR(hor.out%),_
                   VARPTR(vert.out%))
REM
REM set the character height for the tick labels
REM
work1% = fnytr%(4.0)
status% =  vsthgt%(dev.handle%,work1%, _
                   VARPTR(xheight%), _
                   VARPTR(xwidth%), _
                   VARPTR(cwidth%), _
                   VARPTR(cheight%))
REM
REM    set an index into the array of months
REM
j = 0
REM
REM    for the points along the horizontal axis
REM
work2% = fnytr%(10.0)
FOR i = 43 TO 88 STEP 15
REM
REM    write out a month label
REM
  work1% = fnxtr%(i)
  status% =  vgtext%(dev.handle%,work1%, _
                     work2%, _
                     CCHAR(ticks$(j),tickay%()))
  j = j + 1
NEXT i
REM
REM    set text alignment to middle right
REM
status% =  vstaln%(dev.handle%,2,1, _
                   VARPTR(hor.out%), _
                   VARPTR(vert.out%))
REM
REM    set an index into the arrays of tasks
REM
j = 0
REM
REM    write out the lables for the points along
REM     the vertical axis
REM
work1% = fnxtr%(33.0)
FOR i = 15 TO 75 STEP 10
  work2% = fnytr%(i)
  status% =  vgtext%(dev.handle%,work1%,_
                     work2%, _
                     CCHAR(tasks$(j),taskay%()))
  j = j + 1
NEXT i
REM
REM    set text alignment back to
REM    the default lower left
REM
```

```
status% =  vstaln%(dev.handle%,0,0, _
                   VARPTR(hor.out%), _
                   VARPTR(vert.out%))
REM
REM    write out the subtitle to the chart
REM
work1% = fnxtr%(35.0)
work2% = fnytr%(82.0)
status% =  vgtext%(dev.handle%, _
                   work1%,work2%, _
                 CCHAR("As of June 15",WORK%()))
REM
REM    set a new character height for the title
REM
work1% = fnytr%(9.0)
status% =  vsthgt%(dev.handle%,work1%, _
                   VARPTR(xheight%), _
                   VARPTR(xwidth%), _
                   VARPTR(cwidth%), _
                   VARPTR(cheight%))
REM
REM    write out the title
REM
work1% = fnxtr%(35.0)
work2% = fnytr%(88.0)
status% =  vgtext%(dev.handle%,work1%,work2%, _
        CCHAR("Processing", WORK%()))
REM
REM    set the fill pattern to a 45 degree hatch
REM
status% =  vsfst1%(dev.handle%,2)
status% =  vsfint%(dev.handle%,3)
REM
REM    set an index into the arrays of dates
REM
j = 0
REM
REM    for the location of the bars
REM    create the array for drawing,
REM    and output them
REM
REM
FOR i% = 12 TO 72 STEP 10
  xy%(1) = fnytr%(i%)
  xy%(3) = fnytr%(i% + 6)
  xy%(0) = fnxtr%(start.dates%(j))
  xy%(2) = fnxtr%(end.dates%(j))
  status% =  vbar%(dev.handle%,VARPTR(xy%(0)))
  j = j + 1
NEXT i%
REM
REM    call routine to create box coordinates
REM
x.min = 35.0
x.max = 95.0
y.min = 10.0
y.max = 80.0
gosub 10000
```

```
        REM
        REM     draw a frame around the chart
        REM
        status% =  vpline%(dev.handle%,5,VARPTR(xy%(0)))
        REM
        REM     call routine to create box coordinates
        REM
        x.min = 0.0
        x.max = 100.0
        y.min = 0.0
        y.max = 100.0
        gosub 10000
        REM
        REM     draw a border around the page
        REM
        status% =  vpline%(dev.handle%,5,VARPTR(xy%(0)))
        REM
        REM     wait for the viewer before we close the
        REM     workstation
        REM
        status% =  vrqstr%(dev.handle%,1,0, _
                           VARPTR(echo.xy%(0)), _
                           VARPTR(dummy%(0)))
        REM
        REM     close down the workstation
        REM
        status% =  vclswk%(dev.handle%)
        end
        REM
10000   REM
        REM     a subroutine to create box coordinates
        REM     from mins and maxs
        REM
        REM
        REM     set values for all the corners
        REM
        xy%(0) = fnxtr%(x.min)
        xy%(1) = fnytr%(y.min)
        xy%(2) = fnxtr%(x.max)
        xy%(3) = xy%(1)
        xy%(4) = xy%(2)
        xy%(5) = fnytr%(y.max)
        xy%(6) = xy%(0)
        xy%(7) = xy%(5)
        xy%(8) = xy%(0)
        xy%(9) = xy%(1)
        return
```

International Business
Machines Corporation
Dept. 997, Bldg. 998
11400 Burnet Rd.
Austin, Texas 78758

Printed in the
United States of America

59X8615