CP/M pages

Home -> CP/M -> GSX

# The Graphics System eXtension

The GSX is a graphics library, for CP/M and CP/M-86, designed to be portable. This document gives details of the system calls it uses.

A conventional GSX system comes in three parts:

- `GSX.SYS / GENGRAF.COM` contain the device-independent portion of GSX (the GDOS). This is normally added to GSX-aware programs when they are distributed.
- `ASSIGN.SYS` is a text file listing the device drivers on a system;
- `DD*.PRL` are the actual device drivers (the GIOS).

Under CP/M-86, the files are:

- `GRAPHICS.CMD` is the resident portion of GSX, including the GDOS.
- `ASSIGN.SYS`, as for the CP/M-80 version, lists the device drivers in the system.
- `DD*.SYS` are the device drivers, in CP/M-86 CMD format.

GSX went on to become the Virtual Device Interface of the GEM environment. GEM versions 1 and 2 support the GSX API as well as the GEM one.

JOYCE incorporates a GSX driver allowing CP/M programs to use 800x600 colour graphics.

GSX is accessed using one BDOS call - number 115. It is entered with C=73h (115) and DE=address of parameter block. On exit, values in the arrays indicated by the parameter block are changed.

In CP/M-86, GSX should be invoked with CX=0473h and DS:DX giving the address of the parameter block. If CH has a value other than 4, it will invoke one of the GDOS internal functions.

The parameter block format is:

```
        DEFW    CONTRL  ;Address of control array
        DEFW    INTIN   ;Address of integer input array
        DEFW    PTSIN   ;Address of pixel input array
        DEFW    INTOUT  ;Address of integer output array
        DEFW    PTSOUT  ;Address of pixel output array
```

(in CP/M-86, these are 4-byte segment/offset addresses).

The control array is:

```
CONTRL: DEFW    function ;Input:  GSX function, 1-33
        DEFW    #ptsin   ;Input:  Number of points in PTSIN array.
        DEFW    #ptsout  ;Output: Number of points in PTSOUT array.
        DEFW    #intin   ;Input:  Number of integers in INTIN array.
        DEFW    #intout  ;Output: Number of integers in INTOUT array.
XCTRL:  DEFW    special  ;Input:  For special uses.
```

## Function 1 - Open workstation

### Entered with:

- *function*=1;
- *#ptsin*=0;
- *#intin*=10;
- INTIN contains initial settings.

### Returns:

- *#ptsout*=6;
- *#intout*=45;
- INTOUT contains device data.

This loads a device driver and initialises it. Valid device driver numbers are found in the ASSIGN.SYS file, and follow this pattern:

```
 1- 9: Screen
11-19: Plotter
21-29: Printer
31-39: GKS metafile.
```

The format of the INTIN array is:

```
INTIN:  DEFW    device_number
        DEFW    line_style
        DEFW    line_colour
        DEFW    marker_style
        DEFW    marker_colour
        DEFW    text_style
        DEFW    text_colour
        DEFW    fill_style
        DEFW    fill_index
        DEFW    fill_colour
```

The INTOUT return array gives:

```
INTOUT: DEFW    Screen width, device units
        DEFW    Screen height, device units
        DEFW    0 if device is capable of continuous scaling (eg a printer),
                1 if it is not (eg a CRT)
        DEFW    Width of a pixel, in thousandths of a millimetre.
        DEFW    Height of a pixel, in thousandths of a millimetre.
        DEFW    Number of character sizes, 0 for continuous sizing.
        DEFW    Number of line styles.
        DEFW    Number of line widths.
        DEFW    Number of marker styles.
        DEFW    Number of marker sizes.
        DEFW    Number of fonts.
        DEFW    Number of patterns.
        DEFW    Number of hatch styles.
        DEFW    Number of colours displayable at once.
        DEFW    Number of General Drawing Primitives
```

```
                        ; 3 => Pie slice
                        ; 4 => Circle
                        ; 5 => Ruling characters
        DEFS    20      ;General Drawing Primitive attributes
                        ;-1 => End of list
                        ; 0 => Polyline
                        ; 1 => Polymarker
                        ; 2 => Text
                        ; 3 => Filled area
                        ; 4 => None
        DEFW    0 for black/white, 1 for colour.
        DEFW    0 if text rotation is not possible, 1 if it is.
        DEFW    0 if filled areas are not possible, 1 if they are.
        DEFW    0 if cannot read cell array, 1 if can.
        DEFW    Number of colours in the palette.
                        ; 0 => More than 32767
                        ; 2 => Black and white
        DEFW    Number of locator devices (mice, tablets, lightpens)
        DEFW    Number of valuator devices
        DEFW    Number of choice devices
        DEFW    Workstation type
                        ; 0 => Output only
                        ; 1 => Input only
                        ; 2 => Input and output
                        ; 3 => Segment storage
                        ; 4 => GKS metafile output.
```

The PTSOUT return array gives:

```
        DEFW    ?, minimum character height
        DEFW    ?, maximum character height
        DEFW    minimum line width,?
        DEFW    maximum line width,?
        DEFW    ?, minimum marker height
        DEFW    ?, maximum marker height.
```

## Function 2 - Close workstation

### Entered with:

- *function*=2;
- *#intin*=0.

### Returns:

- *#intout*=0.

GSX can only use one device at a time. When you have finished with a device, close it.

## Function 3 - Clear picture

### Entered with:

- *function*=3;

- *#intout*=0.

Empties the buffer containing the current picture. This may be the screen, or buffered data for something like a printer.

## Function 4 - Output graphics

**Entered with:**

- *function*=4;
- *#intin*=0.

**Returns:**

- *#intout*=0.

Ensures that all graphics have been displayed which should be displayed. On a device such as a printer, this will cause the picture to be printed.

## Function 5 - Escape

**Entered with:**

- *function*=5;
- *special*=escape number;
- Other values as required by the escape.

**Returns:** values vary.

The escapes are:

1. Get text screen size in characters. The first two words of INTOUT will hold the height and the width respectively, and this will therefore return *#intout*=2.
2. Enter graphics mode.
3. Enter text mode.
4. Text cursor up.
5. Text cursor down.
6. Text cursor left.
7. Text cursor right.
8. Clear text screen.
9. Clear from text cursor to end of screen.
10. Clear from text cursor to end of line.
11. Move text cursor to coordinates given in INTIN. Therefore this should be entered with *#intin*=2.
12. Print (to the text screen) a string whose characters are stored in INTIN (one character to each word). *#intin* = length of string.
13. Select reverse video.

yes.

17. Dump the current screen to the printer.
18. Place a graphic cursor (a mouse pointer or similar). Its coordinates are given in PTSIN, so enter with *#ptsin*=1.
19. Remove the graphic cursor.

Most of these escapes are not supported by the drivers supplied with GEM.

## Function 6 - Draw a polyline

**Entered with:**

- *function*=6;
- *#ptsin*=number of points;
- Coordinates of points in PTSIN.

**Returns:** Nothing.

## Function 7 - Plot a group of markers

**Entered with:**

- *function*=7;
- *#ptsin*=number of markers;
- Coordinates of markers in PTSIN.

**Returns:** Nothing.

## Function 8 - Draw text

**Entered with:**

- *function*=8;
- *#intin*=number of characters;
- *#ptsin*=1.
- Coordinates of text in PTSIN.
- Text in INTIN.

**Returns:** Nothing.

## Function 9 - Draw a filled polygon.

**Entered with:**

- *function*=9;
- *#ptsin*=number of vertices;

Not all drivers can support filled polygons; usually, the fallback position is to draw the outline of the polygon.

## Function 10 - Output colour index array.

**Entered with:**

- *function*=10;
- *#ptsin*=2;
- *#intin*=length of array;
- Array in INTIN;
- Coordinates in PTSIN for bottom left and top right corners;
- Additional parameters in *contrl* as follows:

```
XCTRL: DEFW    length of each row
                DEFW    number of elements used in each row
                DEFW    number of rows
                DEFW    mode
```

*Mode* is:

1. replace
2. OR
3. XOR
4. erase

**Returns:** Nothing.

## Function 11 - General Drawing Primitive

**Entered with:**

- *function*=11;
- *special*=primitive ID
- *#ptsin, #intin*, PTSIN, INTIN vary.

General Drawing Primitives may not be present on all systems; they are normally provided if the hardware can do one of these operations faster than the generic GSX functions. For example, many raster graphic systems can draw a filled bar very quickly since it aligns with the scan lines.

**ID=1**
Filled bar. *#ptsin*=2; PTSIN gives diagonally opposite corners.

**ID=2**
Arc. *#ptsin*=4, *#intin*=2. INTIN holds start angle and end angle in 360ths of a degree; PTSIN holds coords of centre, start point, end point and (radius,0).

**ID=3**
Pie slice. As for arc.

**ID=5**

Draw text. *#ptsin*=1; *#intin*=no. chars. PTSIN holds text coordinates; INTIN holds 16-bit characters.

**ID=6-7**

Reserved.

**ID=8-10**

Available to implementor (used in GEM).

## Function 12 - Set text size

**Entered with:**

- *function*=12;
- *#ptsin*=1
- PTSIN holds (0, height)

Returns:

- PTSOUT[0]=(width,height) of a W
- PTSOUT[1]=(width,height) of a cell

## Function 13 - Set text direction

**Entered with:**

- *function*=13;
- *#intin*=3
- INTIN holds:
  - INTIN[0] = Angle to turn through in tenths of a degree (anticlockwise, 0 = normal)
  - INTIN[1] = 100 cos (angle)
  - INTIN[2] = 100 sin (angle)

In practice many drivers only support text rotations of 0, 90, 180 and 270 degrees.

## Function 14 - Set colour index (palette registers)

**Entered with:**

- *function*=14;
- *#intin*=4;
- INTIN holds:
  - INTIN[0] = Colour number
  - INTIN[1] = Red 0-1000
  - INTIN[2] = Green 0-1000
  - INTIN[3] = Blue 0-1000

**Entered with:**

- *function*=15;
- *#intin*=1;
- INTIN[0]=style.

## Function 16 - Set line width

**Entered with:**

- *function*=16;
- *#ptsin*=1;
- PTSIN[0]=(width,0)

## Function 17 - Set line colour

**Entered with:**

- *function*=17;
- *#intin*=1;
- INTIN[0]=colour.

## Function 18 - Set marker type

**Entered with:**

- *function*=18;
- *#intin*=1;
- INTIN[0]=type.

## Function 19 - Set marker height

**Entered with:**

- *function*=19;
- *#ptsin*=1;
- PTSIN[0]=(0, height)

## Function 20 - Set marker colour

**Entered with:**

- *function*=20;

## Function 21 - Set text font

**Entered with:**

- *function*=21;
- *#intin*=1;
- INTIN[0]=font.

## Function 22 - Set text colour

**Entered with:**

- *function*=22;
- *#intin*=1;
- INTIN[0]=colour.

## Function 23 - Set fill style

**Entered with:**

- *function*=23;
- *#intin*=1;
- INTIN[0]=style.

Fill style is 0-3: 0=transparent, 1=solid, 2=Pattern, 3=Hatch.

## Function 24 - Set fill index

**Entered with:**

- *function*=24;
- *#intin*=1;
- INTIN[0]=fill index.

The fill index is used only with styles 2 & 3.

## Function 25 - Set fill colour

**Entered with:**

- *function*=25;
- *#intin*=1;
- INTIN[0]=colour.

- *function*=26;
- *#intin*=2;
- INTIN[0]=style;
    - INTIN[1]=0 to request return values (values requested when palette was set).
    - INTIN[1]=1 to request real values (values actually in use at the moment).

Returns INTOUT[0]=Colour value, INTOUT[1-3]=RGB values 0-1000.

## Function 27 - Inquire cell array

**Entered with:**

- *function*=27;
- *#ptsin*=2;
- *#intin*=maximum length of colour index array;
- CONTRL[5]=length of each row;
- CONTRL[6]=number of rows.
- PTSIN holds coordinates of bottom LH and top RH corners of array.

Returns:

- CONTRL[7]=no. elements used in each row.
- CONTRL[8]=no. rows used.
- CONTRL[9]=error flag (0=OK 1=error).
- Array in INTOUT.

If INTOUT[x]=-1, the corresponding pixel could not be read.

*The next few functions can be operated in Request mode or Sample mode*:

## Function 28 - Read locator (eg tablet or mouse)

### In Request mode:

**Entered with:**

- *function*=28;
- *#ptsin*=1;
- *#intin*=1;
- INTIN[0]=locator number (normally 1 for keyboard, 2 for mouse or tablet);
- PTSIN gives initial coordinates.

Returns:

- PTSOUT=coordinates when key/button pressed;
- INTOUT[0]=terminator (key pressed, or mouse button +20h); *#intout*=0 if error.

- *function*=28;
- *#intin*=1;
- INTIN[0]=locator number (normally 1 for keyboard, 2 for mouse etc.)

Returns:

**If coordinates changed:**
*#ptsout*=1 to indicate coordinates changed; new coordinates in PTSOUT[0];
**If key or button pressed:**
*#intout=1* to indicate key/button pressed; and key or button in INTOUT[0].

If you are developing a GSX driver, note that some programs (including DR DRAW) only check for button presses if there are no new coordinates.

## Function 29 - Read valuator

### In Request mode:

**Entered with:**

- *function*=29;
- *#intin*=2;
- INTIN[0]=valuator number.
- INTIN[1]=initial value.

Returns:

- INTOUT[0]=final value when key/button pressed;
- INTOUT[1]=terminator (key pressed, or button + 20h); *#intout*=0 if error.

### In Sample mode:

**Entered with:**

- *function*=29;
- *#intin*=1;
- INTIN[0]=valuator number.

Returns:

**If value changed:**
*#intout*=1; new value in INTOUT[0].
**If key or button pressed:**
*#intout=2* if key or button pressed; final value in INTOUT[0] and key or button in INTOUT[1].

## Function 30 - Read choice

### In Request mode:

**Entered with:**

- *function*=30;
- *#intin*=1;
- INTIN[0]=choice number (1=function keys on keyboard).

Returns *#intout*=1, INTOUT[0]=choice (1-n).

### In Sample mode:

**Entered with:**

- *function*=30;
- *#intin*=1;
- INTIN[0]=choice number (1=function keys on keyboard).

Returns:

- *#intout*=0 if nothing happened;
- *#intout*=1 if choice (choice in INTOUT[0]);
- *#intout*=2 if non-choice key (choice in INTOUI[0], key in INTOUT[1]).

## Function 31 - Read string

### In Request mode:

**Entered with:**

- *function*=31;
- *#ptsin*=INTIN[2];
- *#intin*=3;
- INTIN[0]=device number;
- INTIN[1]=maximum length;
- INTIN[2]=1 to echo, 0 not to.
- If echoing is on, PTSIN[0] gives coordinates of where to echo the string.

Returns *#intout*=length of string returned, string in INTOUT.

On a two-head system with separate text and graphics screens, the input text may be echoed on the text screen, not the graphics one.

### In Sample mode:

- *#ptsin*=0;
- *#intin*=2;
- INTIN[0]=device number;
- INTIN[1]=maximum length.

Returns *#intout*=length of string returned, string in INTOUT.

### Function 32 - Set writing mode

**Entered with:**

- *function*=32;
- *#intin*=1;
- INTIN[0]=mode.

Note: "background" is the second colour used in dashed lines etc. When such a line is being drawn, the dashes are drawn as "foreground" areas and the gaps as "background" areas.

Modes are:

1. Replace. "Foreground" and "background" areas are replaced ("background" areas with GSX colour 0).
2. Transparent. "Foreground" areas are replaced but "background" areas stay the same.
3. XOR. "Foreground" areas are XOR'ed with previous colour; "background" areas stay the same.
4. Erase. "Foreground" areas are written in GSX colour 0; "background" areas stay the same.

### Function 33 - Set input mode

**Entered with:**

- *function*=33;
- *#intin*=2;
- INTIN[0]=device type (1=locator, 2=valuator, 3=choice, 4=string).
- INTIN[1]=mode (1=Request, 2=Sample).

## Activity extensions

The 'Activity' GUI on the Apricot PC range uses GSX video drivers, with the admixture of some extensions. The numbering scheme used by the extra functions seems to match the GEM VDI, but the parameters often differ.

### Function 104 - Set Perimeter

- #intin=1;
- INTIN[0]=perimeter (1 to give filled shapes a perimeter, 0 not to).

## GDOS internal functions

In GSX-86, the function passed in CX to INT 0E0h is interpreted as follows:

```
CL = 73h
CH low nibble  = function, 0-7
CH high nibble = layer number, 0-14
```

('Layer' is the index into the internal driver table).

Functions are:

**Function=0: KillLayer**

Unbind any driver in the specified layer.

**Function=1: BindIndirect**

Load a driver into the specified layer by ASSIGN.SYS ID (1-9 for screen, 10-19 for plotter, etc.)

Pass DX = ASSIGN.SYS ID. If succeeded returns AL=0. If failed returns AL=0FFh, AH = error number.

**Function=2: BindFCB**

Load a driver into the specified layer by FCB. DS:DX = FCB address.

**Function=3: BindDirect**

Map an already-loaded driver into the specified layer. DS:DX = address of entry point.

**Function=4: InvokeDriver**

The public GSX API, as described above.

**Function=5: PassThrough**

Call the entry point of the driver in the specified layer with all caller registers as passed to GSX-86. No coordinate scaling will be done on entry or exit.

**Function=6: ReturnPtr**

Return ES:BX = address within the GDOS of a table describing the specified layer:

```
00:     DB      HowBound
                        0FFh: Driver not loaded
                        000h: Driver loaded by BindIndirect
                        001h: Driver loaded by BindFCB
```

```
                                Segment of driver's base page [data segment]
      07:     DW      ModuleSize
                                Number of paragraphs allocated to this driver
      09:     DW      ProcId
                                ASSIGN.SYS device ID of this driver
      0B:     DW      XPixels
                                Device resolution, horizontal
      0D:     DW      YPixels
                                Device resolution, vertical
      0F:     DB      0
                                Filler byte
```

### Function=7: ReturnMaxProc

> Returns AX = 0Fh (maximum number of layers)

The API design suggests that GSX-86 was designed to be able to cope with multiple concurrently-loaded drivers, using the layer number to distinguish them (ie, calling GSX with CX=0473h, CX=1473h, CX=2473h etc.). In the current GSX-86, only layers 0 and 1 have meaning. Layer 0 is for the current graphics driver, and layer 1 is for character I/O.

## GDOS character I/O

GSX-86 device drivers are expected to do keyboard and screen I/O using layer 1 functions. The example in the GDOS source suggests calling INT 0E0h / CX=1573h (PassThrough to layer 1). The GSX 1.1 drivers do it a bit differently, by using INT 0E0h / CX=1673h to get the address of the driver definition, and extracting its entry point (ProcPtr).

The entry point is called with:

```
BH = function:
        0 = Reset channel
        1 = Get input status, return status in AX
        2 = Wait for input, return input status in AX, input in DX
        3 = Poll input, return input status in AX, input (if any) in DX
        4 = Reserved
        5 = Get output status, return status in AX
        6 = Wait for output ready, then send character in DX. Returns
            status in AX.
        7 = If output is ready, send character in DX. Returns status in AX.
        8 = Line input, DS:DX as for BDOS function 0Ah.
BL = channel number:
        0 = CON: (screen / keyboard)
        1 = AUX: (serial)
        2 = LST: (printer)
        3 = CON: (graphics)
        4 = AUX: (sgraphics)
        5 = LST: (graphics)
        6,7 not assigned
DX = data, if appropriate
```

Status returned in AX:

```
Bit 0:  On receive: Parity error
Bit 1:  On receive: Overrun error
Bit 2:  On receive: Framing error
Bit 13: On receive: End of file. On send: Channel has hardware queue, data
```

Return to archive listing