XiJet
OEM Inkjet Print Engines

8 Lunar Drive, Woodbridge, CT 06525

HP45 API

USB Interface Specification

Dec 23, 2008

DOCUMENT REVISION 2.6

API v5.5

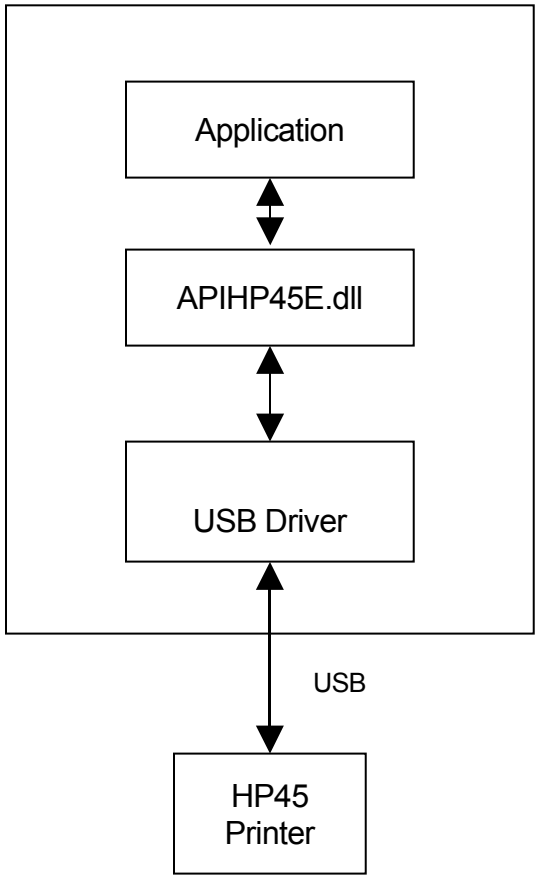
Tel: (203) 397-2800 Fax: (203) 389-1129

Table of Contents

Overview

The HP45 API has been developed to simplify the interface between an application program and the underlying USB driver and printer drive electronics.

The API is supplied in the form of a Windows DLL.



The API accepts bit-mapped image data at a depth of 1-bit per pixel. Two printing modes are provided: Page mode expects that the entire page has been pre-processed, while Canvas mode allows a page to be built up in rectangular sections.

Functions

HP45E_ProbePrinter

```
BOOL HP45E_ProbePrinter(  USHORT index,
                          CHAR *PrinterName );
```

Returns: TRUE if printer *index* exists.

HP45E_ProbePrinter will return the printer name associated with a given index. An application uses this function to build a list of all attached HP45 printers by starting *index* at 0 and iterating as long as the function returns TRUE. Each HP45 printer can be composed of a single head or a collection of heads. Up to 4 pens may be attached to each head.

HP45E_OpenPrinter

```
HANDLE HP45E_OpenPrinter( CHAR *PrinterName,                // as returned from HP45E_ProbePrinter
                          HP45E_CONFIGURATION *pPrinterConfig);
```

Returns: A handle to the printer if successful, else returns INVALID_HANDLE_VALUE.

HP45E_OpenPrinter will open the HP45 printer specified by PrinterName. On success a handle is returned for use in subsequent API calls. Also, the HP45E_CONFIGURATION structure pointed to by pPrinterConfig will be populated with information associated with the printer.

```
typedef struct {
        USHORT          VerticalResolutionDPI;          //  printhead resolution
        USHORT  HorizontalResolutionDPI;                //  transport resolution
        USHORT          NumberOfPrintheads;             //  up to 4 printheads may be attached
        USHORT          Head1Height;                    //  height of each printhead in pixels
        USHORT          Head2Height;
        USHORT          Head3Height;
        USHORT          Head4Height;
} HP45E_CONFIGURATION;
```

HP45E_ClosePrinter

```
void HP45E_ClosePrinter( HANDLE PrinterHandle)  ;          //  as returned from open call
```

HP45E_ClosePrinter closes the printer port and associated resources.

HP45E_GetPrinterParameter

```
int HP45E_GetPrinterParameter (  HANDLE PrinterHandle,          //  as returned from open call
                                 USHORT ParameterIndex,
                                 PVOID pParameter,
                                 USHORT HeadNumber = 0);        //  when applicable
```

Returns: 1 = success
0 = error

HP45E_GetPrinterParameter will return a single printer configuration parameter. *ParameterIndex* dictates which parameter will be returned, while pParameter points to an appropriately sized memory field. Defined constants for ParameterIndex are located in APIHP45E.h. The optional parameter *HeadNumber* should only be included for head specific parameters, such as trigger offset distance.

For example to read the printers current resolution:

```
int Success;
USHORT ResolutionIndex;

Success = HP45E_GetPrinterParameter( printerHandle,  HP45E_RESOLUTION, &ResolutionIndex);
```

See Appendix A for list of adjustable printer parameters.

HP45E_SetPrinterParameter

```
int HP45E_SetPrinterParameter (  HANDLE PrinterHandle,          //  as returned from open call
                                 USHORT ParameterIndex,
                                 PVOID pParameter,
                                 USHORT HeadNumber = 0);        //  when applicable
```

Returns: 1 = success
0 = error

HP45E_SetPrinterParameter will set a single printer configuration parameter. *ParameterIndex* dictates which parameter will be modified, while pParameter points to an appropriately sized memory field, preloaded with the new parameter value. The optional parameter *HeadNumber* should only be included for head specific parameters, such as trigger offset distance.

NOTE: Parameter changes made are temporary, NOT permanent. Stored configuration will re-loaded on the next call to HP45E_OpenPrinter.

For example to set the printers resolution to 600x300 mode:

```
int Success;
USHORT ResolutionIndex = HP45E_RES_600_300;

Success = HP45E_SetPrinterParameter( printerHandle,  HP45E_RESOLUTION, &ResolutionIndex);
```

See Appendix A for list of adjustable printer parameters.

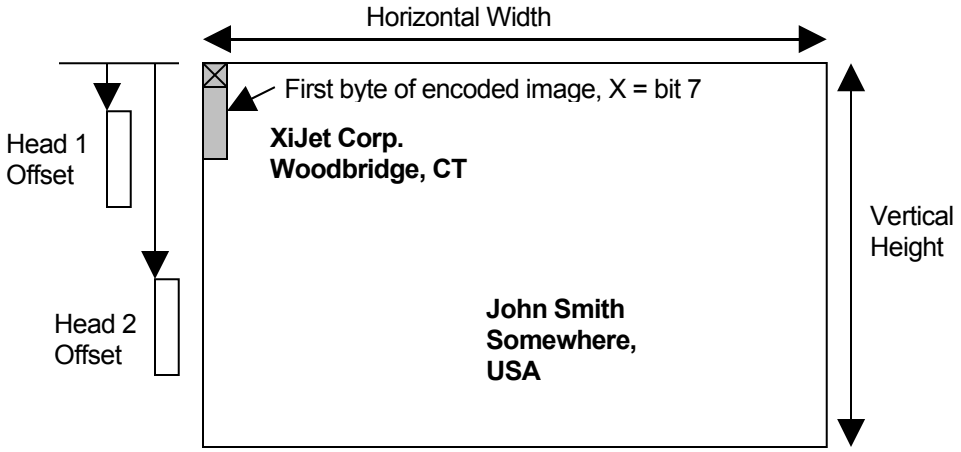
HP45E_PrintDocumentPage

```
int HP45E_PrintDocumentPage( HANDLE PrinterHandle,        //  as returned from open call
                             ULONG VerticalHeight,        //  in pixels
                             ULONG HorizontalWidth,       //  in pixels
                             UCHAR *pBitmappedBuffer,     //  1-bit per pixel image
                             ULONG Head1Offset,           //  Vertical offset in pixels
                             ULONG Head2Offset,           //  Vertical offset in pixels
                             ULONG Head3Offset,           //  Vertical offset in pixels
                             ULONG Head4Offset,           //  Vertical offset in pixels
                             ULONG Timeout );             //  timeout in msecs
```

Returns: Value = 1 indicates success.
Value = 0 indicates timeout occurred.
Value = -1 indicates the printer is in error state, call HP45E_GetStatus to retrieve error message.

HP45E_PrintDocumentPage copies a single document page into HP45 memory in preparation for printing. The HP45 uses a double buffering scheme and therefore the first call to HP45E_PrintDocumentPage will return as soon as the image has been transferred to the HP45 and is ready to print. Subsequent calls will not return until the previous page has been printed or until a timeout occurs. In the event that a timeout occurs, HP45E_PrintDocumentPage will return a value of 0 and the document *will not* enter the queue. The calling application may choose to wait and retry, however a timeout here is likely the result of an interruption in document feeding.

- Raster data is encoded from top to bottom, left to right across the image.
- For this mode, encode the entire page including whitespace above, below and between multiple printheads.
- High order bit is the most significant (similar to 1-bit per pixel BMP). X in the picture below represents bit 7 of the first byte of the buffered image.
- Each new line of encoded raster shall begin on a 32 bit “long word” boundary (also similar to BMP file format).
- Specify a Vertical Offset for each attached head. This need not be on a word or byte boundary.



HP45E_CanvasBegin

```
int HP45E_CanvasBegin(  HANDLE PrinterHandle,          //  as returned from open call
                        ULONG VerticalHeight,          //  in pixels
                        ULONG HorizontalWidth);        //  in pixels
```

Returns: 1 = success
0 = error

HP45E_CanvasBegin is the first call in canvas mode image composition. It will reserve and clear an area of image memory as dictated by VerticalHeight and HorizontalWidth. If the API cannot reserve enough memory a value of 0 will be returned, 0 will also be returned for an invalid parameter.

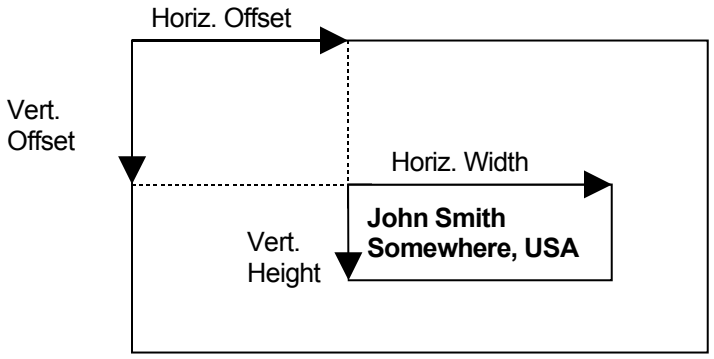
HP45E_CanvasWrite

```
int HP45E_CanvasWrite(  HANDLE PrinterHandle,          //  as returned from open call
                        ULONG VerticalOffset,          //  in pixels
                        ULONG HorizontalOffset,//  in pixels
                        ULONG VerticalHeight,          //  in pixels
                        ULONG HorizontalWidth,//  in pixels
                        UCHAR *pBitmappedBuffer);      //  1-bit per pixel image
```

Returns: 1 = success
0 = error

HP45E_CanvasWrite copies a 1-bit per pixel image onto HP45 canvas. Like HP45E_PrintDocumentPage raster data is encoded from top to bottom, left to right across the image, with bit 7 being the most significant bit (see HP45E_PrintDocumentPage). Return value of 0 indicates invalid parameter.

Parameters *VerticalHeight* and *HorizontalWidth* are specified in pixels. Each new line of encoded raster shall begin on a long word boundary.



HP45E_LoadFontXFT

```
HANDLE HP45E_LoadFontXFT(  HANDLE PrinterHandle,  //  as returned from open call
                           const char *filename);     // full pathname to XFT font file
```

Returns: A handle to the font if successful, else returns INVALID_HANDLE_VALUE.

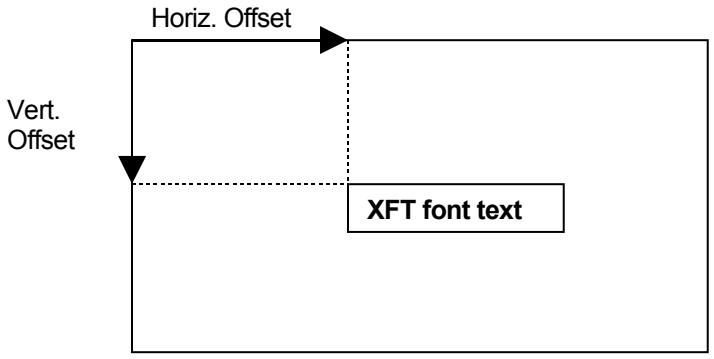
HP45E_LoadFontXFT is used to preload a XiJet format font file. The handle returned may then be used in calls to HP45E_CanvasWriteStr.

HP45E_CanvasWriteStr

```
int HP45E_CanvasWriteStr(   HANDLE PrinterHandle,  //  as returned from open call
                            HANDLE FontHandle,     // as returned from HP45E_LoadFont
                            const char *printString,     // pointer to string to print
                            ULONG VerticalOffset,    //  in pixels
                            ULONG HorizontalOffset);          //  in pixels
```

Returns: 1 = success
          0 = error

HP45E_CanvasWriteStr inserts text into a canvas. The font must be previously loaded via HP45E_LoadFontXFT. Parameters *VerticalOffset* and *HorizontalOffset* are specified in pixels.



HP45E_CanvasPrint

```
int HP45E_CanvasPrint (  HANDLE PrinterHandle,        //  as returned from open call
                         ULONG Head1Offset,           //  Vertical offset in pixels
                         ULONG Head2Offset,           //  Vertical offset in pixels
                         ULONG Head3Offset,           //  Vertical offset in pixels
                         ULONG Head4Offset,           //  Vertical offset in pixels
                         ULONG Timeout );             //  timeout in msecs
```

Returns: 1 = success
0 = indicates timeout occurred
-1 = indicates the printer is in error state, call HP45E_GetStatus to retrieve error message.

HP45E_CanvasPrint transfers image data to the HP45 for printing. The HP45 uses an image buffering scheme and therefore the first call to HP45E_CanvasPrint will return as soon as the image has been transferred to the HP45 and is ready to print. Subsequent calls may block while the image queue is full or until a timeout occurs. In the event that a timeout occurs, HP45E_CanvasPrint will return a value of 0 and the document *will not* enter the queue. The calling application may choose to wait and retry.

HP45E_CanvasPrintExt2

```
int HP45E_CanvasPrintExt2 (  HANDLE PrinterHandle,          //  as returned from open call
                             ULONG Head1Offset,             //  Vertical offset in pixels
                             ULONG Head2Offset,             //  Vertical offset in pixels
                             ULONG Head3Offset,             //  Vertical offset in pixels
                             ULONG Head4Offset,             //  Vertical offset in pixels
                             ULONG Timeout,                         //  timeout in msecs
                             float *pTransportSpeedIPS);  //  Returned transport speed
```

Returns: 1 = success
0 = indicates timeout occurred
-1 = indicates the printer is in error state, call HP45E_GetStatus to retrieve error message.

HP45E_CanvasPrintExt2 functions the same as HP45ECanvasPrint, but will also return the current transport speed in inches/sec. This call should be used when an application requires the display of transport speed while printing.

HP45E_WaitForPrintComplete

```
int HP45E_WaitForPrintComplete(  HANDLE PrinterHandle,  //  as returned from open call
                                 ULONG Timeout);                //  Timeout in msecs
```

Returns: 1 = success
0 = indicates timeout occurred
-1 = indicates the printer is in error state, call HP45E_GetStatus to retrieve error message.

HP45E_WaitForPrintComplete may be called following a print request. Function will return when printer has completed printing of the current document or canvas page. In the event that a timeout occurs, HP45E_WaitForPrintComplete will return a value of 0.

HP45E_Reset

```
void HP45E_Reset(   HANDLE PrinterHandle );
```

HP45E_Reset will cause the HP45 printer to clear image memory and any error condition. This command will also fully reset the pen cartridges.

HP45E_ResetPrintData

```
void HP45E_ResetPrintData(HANDLE PrinterHandle );
```

HP45E_ResetPrintData will cause the HP45 printer to clear all queued print requests as well as any print in progress.

HP45E_ResetPrintQueue

void HP45E_ResetPrintQueue(HANDLE PrinterHandle);

HP45E_ResetPrintQueue will cause the HP45 printer to clear all queued print requests, but will not effect any print already in progress.

HP45E_GetStatus

int HP45E_GetStatus(HANDLE PrinterHandle, // as returned from open call
CHAR *pStatusMessage); // returned text message

Returns: errorStatus code

HP45E_GetStatus retrieves status of the printer. It may be called at any time, but particularly following an error return of HP45E_PrintDocumentPage or HP45E_PrintCanvas. A text status message will describe any error state.

HP45E_ActivatePens

int HP45E_ActivatePens(HANDLE PrinterHandle); // as returned from open call

Returns: 1 = success
0 = error

HP45E_ActivatePens manually activates all print stalls associated with the printer. *Note that this is not a function that needs to be called when using the recommended configuration*. *XiJet recommends setting and using the automatic pen activation and deactivation option (set via XiJet Control Panel software).*

HP45E_DeactivatePens

int HP45E_DeactivatePens(HANDLE PrinterHandle); // as returned from open call

Returns: 1 = success
0 = error

HP45E_DeactivatePens manually deactivates all print stalls associated with the printer. *Note that this is not a function that needs to be called when using the recommended configuration*. *XiJet recommends setting and using the automatic pen activation and deactivation option (set via XiJet Control Panel software).* If using manual pen activation it is very important to call HP45_DeactivatePens before allowing operator to remove a print cartridge, as removing an active cartridge risks damage to the print stall and HP45 printer board.

HP45E_SelectPens

```
int HP45E_SelectPens(  HANDLE PrinterHandle,          //  as returned from open call
                       ULONG bmPensSelect );          // bit mapped pens selector
```

Returns: 1 = success
0 = error

HP45E_SelectPens may be used to selectively enable individual pen(s) for printing. By default all pens are activated and available for printing. Use this function only in special circumstances under directions from XiJet. The parameter bmPensSelect is defined as follows:

Bit	Head	Pen
0	1	1
1	1	2
2	1	3
3	1	4
4	2	1
5	2	2
6	2	3
7	2	4
8	3	1
9	3	2
10	3	3
11	3	4
12	4	1
13	4	2
14	4	3
15	4	4

Note: when using manual pen activation, call HP45_SelectPens *before* calling HP45_ActivatePens.

HP45E_GetInkRemaining

```
int HP45E_GetInkRemaining(  HANDLE PrinterHandle,          //  as returned from open call
                            USHORT headIndex,              // zero-based index of head number
                            SHORT   *remainingPen1,                // returned SHORT
                            SHORT   *remainingPen2,
                            SHORT   *remainingPen3,
                            SHORT   *remainingPen4 );
```

Returns: 1 = success
0 = error

HP45E_GetInkRemaining returns the approximate ink level on a scale of 0 to 100 (100 being full). This is based solely on ink usage, not any hardware feedback.

HP45E_ResetInkCartridge

```
int HP45E_ResetInkCartridge( HANDLE PrinterHandle,      //  as returned from open call
                             USHORT headIndex,          // zero-based index of head number
                             USHORT penIndex);          // zero-based index of pen number
```

Returns: 1 = success
0 = error

Call HP45E_ResetInkCartridge when a new cartridge is installed. HP45 will reset it's ink level to 100 percent.

HP45E_QueueOutputToggle

```
int HP45E_QueueOutputToggle(    HANDLE PrinterHandle,       //  as returned from open call
                                USHORT headIndex,           // zero-based index of head number
                                USHORT outputBitmask,       // bit 0 = new state of output
                                USHORT toggleTimeMS,        // time in milliseconds (0=permanent)
                                UCHAR  syncOption = 0,      // 0 = immediate
                                USHORT syncDelay = 0 );     // synchronize delay (time or distance)
```

Returns: 1 = success
0 = error

Call HP45E_QueueOutputToggle is used to synchronize a change in output state with documents being printed. The output will not change state until any currently queued documents are finished printing. Current HP45E print electronics provides for 1 auxiliary output per attached print head. Use bit 0 of the outputBitmask parameter to set or clear the output, but please reserve additional bits for future use. The output will be toggled for a period of time equal to the value specified in the toggleTimeMS parameter and then returned to the previous state. If toggleTimeMS is set to 0, the change will be permanent.

By default, the output will be toggled as soon as the most recently transferred document page has been completely printed. However, the syncOption parameter may be set to delay the action of the output for a specific amount of time or distance as follows:

syncOption	syncDelay
0	Don't Care
1	Delay in milliseconds
2	Delay in tenths of inches (1=0.1")

HP45E_OpenLogFile

```
FILE* HP45E_OpenLogFile(  CHAR *filename                    // filename including full path
                          );
```

Returns: FILE pointer if successful, else returns NULL.

HP45E_OpenLogFile may be used to cause diagnostic information to be written to a log file. This function may be called at any time (before or after HP45E_OpenPrinter).

HP45E_CloseLogFile

```
void HP45E_CloseLogFile( );
```

Returns: void

HP45E_CloseLogFile will close the diagnostic log file (if opened).

HP45E_TestUSB

```
ULONG HP45E_TestUSB(  HANDLE PrinterHandle,  //  as returned from open call
                      );
```

Returns: ULONG if successful, else returns 0.

HP45E_TestUSB will transmit a small amount of data to each print head in order to gauge the USB transfer rate. The returned number is the bandwidth in Bytes / second. This function should be called before image data is transferred to the printer

HP45E_GetVersionAPI

```
double HP45E_GetVersionAPI( );
```

Returns: double

HP45E_GetVersionAPI returns a version number as a double precision floating point number.

Appendix A - Printer Parameters

Parameters readable / adjustable by HP45E_GetPrinterParameter / HP45E_SetPrinterParameter are as follows:

Parameter	Size	Description
HP45E_RESOLUTION	USHORT	Index from printer resolutions table
HP45E_TRIGGER_OFFSET	USHORT	Photocell offset in mils (1 mil = 0.001")
HP45E_AUX_OUTPUT	USHORT	Controls aux output on a per head basis
HP45E_QUEUE_DEPTH	USHORT	Number document pages buffered by HP45
HP45E_SUB_SAMPLE	USHORT	Allows faster print speeds by skipping data
HP45E_JET_MASKING	USHORT	Turns off half of the ink jets
HP45E_HEAD_HEIGHT	USHORT	Returns head height (GetPrinterParameter only)
HP45E_CARTRIDGE_PROFILE	STRUCT	Cartridge electrical settings
HP45E_PEN_WARMING	USHORT	Enable or Disable Pen Warming Option

The C header file APIHP45E.h contains defined constants for each of these parameters.

HP45E_RESOLUTION

Printer resolutions are expressed in DPI (Dots / Inch). The following resolutions are valid:

Resolution Index	Vert. x Horiz. DPI	Max Transport Speed
1	600x600	25" / sec
2	600x300	50" / sec
3	300x300 Fast	100" / sec
4	300x300 Dark	50" / sec
5	300x150 Fast	200" / sec
6	300x200 Fast	150" / sec
7	300x250 Fast	120" / sec
8	300x100 Fast	300" / sec
9	300x300	50" / sec
10	300x600	25" / sec
11	600x250	60" / sec
12	600x200	75" / sec
13	600x150	100" / sec
14	600x100	150" / sec

Programmers note: This is a 1 based index; setting resolution to an index of 0 will result in an invalid parameter status. The C header file APIHP45E.h contains defined constants for each of these parameters.

HP45E_TRIGGER_OFFSET

Specifies the distance between the photocell and the leading edge of the print head. This is a head specific parameter and, on multi-headed printers, the *HeadNumber* parameter should be supplied in the call to HP45E_SetPrinterParameter or HP45E_GetPrinterParameter.

HP45E_AUX_OUTPUT

Current HP45E print electronics provides for 1 auxiliary output per attached print head. Use bit 0 of the supplied USHORT to set or clear the output, but please reserve additional bits for future use. This is a head specific parameter and, on multi-headed printers, the *HeadNumber* parameter should be supplied in the call to HP45E_SetPrinterParameter or HP45E_GetPrinterParameter.

HP45E_QUEUE_DEPTH

Specifies the depth of the HP45 image queue. By default the HP45 runs in a double buffered mode (queue depth = 2). By increasing HP45E_QUEUE_DEPTH, the printer will accept more images into internal memory. A multiple head printer that is printing small pieces (such as labels) may require increasing HP45E_QUEUE_DEPTH.

HP45E_SUB_SAMPLE

Allows for faster printing speeds by sub-sampling (or skipping) horizontal print lines. The default value of 1 results in all data printing. A value of 2 instructs the electronics to print every other line, allowing the transport speed to double. Note that print quality (and density) are sacrificed for the increased speed, thus this should be used only in special circumstances.

HP45E_JET_MASKING

Under normal operating conditions, the HP45 ejects ink from two columns of jets separated by a distance of 0.016 inches. In unusual circumstances, this can results in a ghosting, or double image being seen. This could occur when using simulated encoder mode and an incorrect or varying transport speed. Setting HP45E_JET_MASING will cause 1 column of jets to not print. Note that print quality (and density) are sacrificed in this mode, thus this should be used only in special circumstances.

HP45E_HEAD HEIGHT

Returns the head height in mils. This was added in order to support printers with more than 4 print heads attached. This call is only compatible with HP45E_GetPrinterParameter.

HP45E_CARTRIDGE_PROFILE

Allows for the adjustment of ink cartridge electrical characteristics.

```
typedef struct {
        USHORT      voltage;        // cartridge voltage, 0.1 volt units, Range 70 to 112 (7.0 to 11.2)
        USHORT      pulseWidth;     // pulse width, 0.1 microsecond units, Range 19 to 25 (1.9 to 2.5)
        USHORT      temperature;    // degrees C, Range 30 to 70
} HP45E_CARTRIDGE_PROFILE_STRUCT;
```

HP45E_PEN_WARMING

Allows an application to enable or disable Pen Warming option.

Document Revision History

Revision	Notes
2.6	Expanded HP45E_GetPrinterParameter, HP45E_SetPrinterParameter to include HP45E_PEN_WARMING.
2.5	HP45E_CanvasPrintExt2 call
2.4.1	Corrected Resolution Table – Resolutions 6, 7
2.4	Extended Resolutions Table
2.3	HP45E_OpenLogFile, HP45E_CloseLogFile, HP45E_TestUSB, HP45E_GetVersionAPI added. Expanded HP45E_GetPrinterParameter, HP45E_SetPrinterParameter to include HP45E_HEAD_HEIGHT, HP45E_CARTRIDGE_PROFILE.
2.2	HP45E_QueueOutputToggle changed units of delay distance
2.1	HP45E_QueueOutputToggle added
2.0	HP45E_ResetPrintQueue added
1.9	HP45E_ResetPrintData added
1.8	Expanded HP45E_GetPrinterParameter, HP45E_SetPrinterParameter to include control of I/O board aux outputs. Removed calls HP45E_GetPrinterConfiguration, HP45E_SetPrinterConfiguration, HP45E_GetHeadConfiguration, HP45E_SetHeadConfiguration.
1.7	Added calls HP45E_GetPrinterParameter, HP45E_SetPrinterParameter.
1.6	Added calls HP45E_LoadFontXFT, HP45E_CanvasWriteStr
1.5	Added calls HP45E_GetPrinterConfiguration, HP45E_SetPrinterConfiguration, HP45E_GetHeadConfiguration, HP45E_SetHeadConfiguration.
1.4	Added call HP45E_SelectPens
1.3	Added calls HP45E_ActivatePens, HP45E_DeactivatePens, HP45E_GetInkRemaining, HP45E_ResetInkCartridge
1.2	Refined canvas mode calls.
1.0	Initial Release for HP45E