Appendix F : ArtWorks EPS Format

Introduction

The format is based on the Illustrator 88 format. Any procedure which only applies to the ArtWorks EPS format will start with the letter a. An '*' marks procedures that are Illustrator 88 format.

Units Used

Coordinates	: Points
Angles	: Degrees
Colour components	: Real in range 0 to 1. For tints, 0 means '100% of parent colour'.
Flags	: Boolean. 0=False, 1=True

The document is split into four sections :

- **Section 1.** The first deals with the procedures that are only used by ArtWorks when importing the file, i.e. they are not needed to render the file. For example, start and end of group procedures.
- **Section 2.** The second deals with the procedures that are also needed to render the file on a Postscript device.
- **Section 3.** The third section documents the comments at the start of an ArtWorks EPS file, for example the colour table.
- **Section 4.** The final section deals with extra procedures that can be included in the file for use only when importing. For example there is a procedure that allows you to perform clipping of one path to another.

Section 1

Procedures that are not needed to render the file.

Miscellaneous procedures

Procedure : **aoa** (Object angle).
Format : **angle aoa**
Parameters : **angle** : Angle (not zero)

Procedure : **aafs** (Attribute flag)
Format : **fill line fnone aafs**
Parameters : **fill** : True = object is filled
line : True = object has been stroked
fnone : True = Object has no fill colour

Path related procedures

Procedure : **ar** (Rectangle)
Format : **ar**
Parameters : None. Standard path and attributes follow

Procedure : **arr** (Rounded rectangle)
Format : **radius x0 y0 x1 y1 x2 y2 arr**
Parameters : **radius** : Coord
x0-y2 : 6 Coords. Define parallelogram bounding box. Standard path and attributes follow.

Procedure : **ae** (Ellipse)
Format : **x0 y0 x1 y1 x2 y2 ae**
Parameters : **x0-y2** : 6 Coords. Define parallelogram bounding box. Standard path and attributes follow.

Procedure : **apl** (Path length)
Format : **length apl**
Parameters : **length** : In bytes (used when loading the file)

Procedure : **apc**
Format : **apc**
Parameters : None. ArtWorks create next object as a child followed by a parent of it.

Procedure : **aof** (Object flags)
Format : **renderableflag mangledflag aof**
Parameters : **renderableflag** : Set if the object is renderable (If path is not renderable it will always end in a h procedure)
mangledflag : Set if the object is mangled

Text related procedures

Procedure : **asto** (Start of complex text object)
Format : **asto**
Parameters : None.

Procedure : **aeto** (End of complex text object)
Format : **aeto**
Parameters : None.

Procedure : **aco** (Character offset to next one)
Format : **x y aco**
Parameters : **x** : X offset (coord)
y : Y offset (coord)

Procedure : **atc** (Text character)
Format : **string atc**
Parameters : **string** : Text string enclosed in brackets (could be more than one character in future versions)

Procedure : **atph** (Text fitted to a path text object)
Format : **em atph**
Parameters : **em** : em above or below path. A path will follow which the text is fitted to.

Procedure : **atof** (Text object flags)
Format : **path external atof**
Parameters : **path** : True = text is pathified. If true each character has a path (except Space).
external : True = text module is not allowed to throw away paths.

Blend related procedures

Procedure : **asbd** (Start of blend object)
Format : **version expanded num asbd**
Parameters : **version** : Version of procedure def
expanded : True = Blend is expanded
num : Number of shapes in the blend

Procedure : **aebd** (End of blend object)
Format : **aebd**
Parameters : None.

Procedure : **asbr** (Start of blender object)
Format : **expanded complex ltol num en0 en1 asbr**
Parameters : **expanded** : True = blender expanded
complex : True = Blend is complex
ltol : True = One to one blend
num : Number of steps
en0 : Element for start point (1st path)
en1 : Element for start point (2nd path)

Procedure : **aebr** (ArtWorks end of blender object)
Format : **aebr**
Parameters : None.

Mould related procedures

Procedure : **asev** (Envelope object)
Format : **olx oly ohx ohy asev**
Parameters : **olx-ohy** : Original bounding box before being enveloped

Procedure : **aeev** (End of envelope object)
Format : **aeev**
Parameters : **None**

Procedure : **aspr** (Perspective object)
Format : **olx oly ohx ohy aspr**
Parameters : **olx-ohy** : Original bounding box before being perspectivised

Procedure : **aepr** (End of perspective object)
Format : **aepr**
Parameters : **None.**

Procedure : **amm** (Mould move to)
Format : **x y amm**
Parameters : **x,y** : Coords

Procedure : **aml** (Mould line to)
Format : **x y aml**
Parameters : **x,y** : Coords

Procedure : **amc** (Mould curve to)
Format : **x0 y0 xl yl x2 y2 amc**
Parameters : **x0-y2** : Standard curve coords

Procedure : **amcp** (Close path element)
Format : **amcp**
Parameters : None.

Procedure : **amep** (Endpath draw parameter)
Format : **len amep**
Parameters : None.

Group related procedures

Procedure : **u*** (Start of Group)
Format : **u**
Parameters : None.

Procedure : **anu** (Start of Group)
Format : **string anu**
Parameters : **string** : Group name (standard text string enclosed in brackets)

Procedure : **U*** (End of Group)
Format : **U**
Parameters : None.

Section 2

Procedures that are needed to render the document

Procedures that define the current attributes

Colour

Procedure :	`k`*	(Set the current fill colour)
Format :	`C M Y K k`	
Parameters :	`C`	: Cyan component
	`M`	: Magenta component
	`Y`	: Yellow component
	`K`	: Key component

Procedure :	`K`*	(Set the current line colour)
Format :	`C M Y K K`	
Parameters :	`C`	: Cyan component
	`M`	: Magenta component
	`Y`	: Yellow component
	`K`	: Key component

Procedure :	`x`*	(Set the current fill colour)
Format :	`C M Y K name tint x`	
Parameters :	`C`	: Cyan component
	`M`	: Magenta component
	`Y`	: Yellow component
	`K`	: Key component
	`name`	: Colour's name (always enclosed in open and close brackets)
	`tint`	: Value between 0 and 1 (0 is 100% of the colour defined)

Procedure :	`X`*	(Set the current line colour)
Format :	`C M Y K name tine X`	
Parameters :	`C`	: Cyan component
	`M`	: Magenta component
	`Y`	: Yellow component
	`K`	: Key component
	`name`	: Colour's name (always enclosed in open and close brackets)
	`tint`	: Value between 0 and 1 (0 is 100% of the colour defined)

Procedure : **az** (Linear and radial fill)
Format : **fillType CS MS YS KS CE ME YE KE xs ys xe ye az**
Parameters :
fillType : 1 = linear, 2 = radial
CS : Cyan component of start colour
MS : Magenta component of start colour
YS : Yellow component of start colour
KS : Key component of start colour
CE : Cyan component of end colour
ME : Magenta component of end colour
YE : Yellow component of end colour
KE : Key component of end colour
xs-ys : Coords of start of grad fill
xe-ye : Coords of end of grad fill

Procedure : **ax** (Linear and radial fill using named colours)
Format : **fillType CS MS YS KS nameS tintS CE ME YE KE nameE tintE xs ys xe ye az**
Parameters :
fillType : 1 = linear, 2 = radial
CS : Cyan component of start colour
MS : Magenta component of start colour
YS : Yellow component of start colour
KS : Key component of start colour
nameS : Name of start colour
tintS : Value 0.0 to1.0 . Zero = 100% tint
CE : Cyan component of end colour
ME : Magenta component of end colour
YE : Yellow component of end colour
KE : Key component of end colour
nameE : Name of end colour
tintE : Value 0.0 to1.0 . Zero = 100% tint
xs-ys : Coords of start of grad fill
xe-ye : Coords of end of grad fill

Procedure : **axm** (Mutated graduated fill)
Format : **xs ys xe ye axm**
Parameters :
xs-ys : Start coords of grad fill
xe-ye : End coords of grad fill

The fill type, start and end colours are taken from the previous graduated fill procedure (ie. the previous **ax** or **az**).

Overprint related procedures

Procedure : `O*`
Format : `flag O`
Parameters : `flag` :True = Overprinting will be used when a path is filled.

Procedure : `R*`
Format : `flag R`
Parameters : `flag` : True = Overprinting will be used when a path is stroked.

Procedure : `axop` (Graduated fill overprint flags)
Format : `scoflag bscopflag ecoflag becopflag axop`
Parameters :
`scoflag` : True = Start fill colour overprint set
`bscopflag`: True = Both index and colour record overprint flag for the start colour are set
`ecoflag` : True = End fill colour overprint flag set
`becopflag`: True = Both index and colour record overprint flag for the end colour are set

Others

Procedure : `awr` (Winding Rule)
Format : `winding awr`
Parameters : `winding` : 0 - None-Zero , 1 - Even odd

Procedure : `w*`
Format : `width w`
Parameters : `width` : Line width (defined in points)

Procedure : `j*`
Format : `join j`
Parameters : `join` : Line join
0 - mitred joins
1 - round joins
2 - bevelled joins

Procedure : `J*`
Format : `cap J`
Parameters : `cap` : Line cap (used if both start and end caps are the same and not triangular)
0 - butt end caps and the end of the line is squared off
1 - round end caps are used
2 - projecting square end caps are used

Procedure : **d*** (dash pattern)
Format : **[array] phase d**
Parameters : See Illustrator documentation

Procedure : **asc** (Start cap)
Format : **cap tw th asc**
Parameters : **cap** : 0 - butt caps (start of the line is squared off)
1 - round start caps are used
2 - projecting square start caps are used
3 - triangle start caps.
tw : triangle width if cap = 3 otherwise zero.
th : triangle height if cap = 3 otherwise zero.

Procedure : **aec** (End cap)
Format : **cap tw th aec**
Parameters : **cap** : 0 - butt caps (end of the line is squared off)
1 - round end caps are used
2 - projecting square end caps are used
3 - triangle end caps.
tw : triangle width if cap = 3 otherwise zero.
th : triangle height if cap = 3 otherwise zero.

Procedure : **aca** (Clear Path Attributes)
Format : **aca**
Parameters : None. : Reset all attributes that apply to paths to their default values.

Procedure : **asah** (Start arrow head)
Format : **num width height asah**
Parameters : **num** : Arrow head id
width : Width of arrow head
height : Height of arrow head

Procedure : **aeah** (End arrow head)
Format : **num width height aeah**
Parameters : **num** : Arrow head id
width : Width of arrow head
height : Height of arrow head

Procedure : **asat** (Start arrow head transformation matrix)
Format : **a b c d e f asat**
Parameters : **a-f** : Transformation matrix

Procedure : **aeat** (End arrow head transformation matrix)
Format : **a b c d e f aeat**
Parameters : **a-f** : Transformation matrix

Procedures that define a path object.

Procedure :	`m*`	(MoveTo)
Format :	`x y m`	
Parameters :	`x-y`	: MoveTo coords
Procedure :	`l*`	(LineTo)
Format :	`x y l`	
Parameters :	`x-y`	: LineTo coords
Procedure :	`c*`	(CurveTo)
Format :	`x0 y0 x1 y1 c`	
Parameters :	`x0-y2`	: CurveTo bezier definition

End of path or sub-path procedures

Procedure :	`s*`	The path is closed but not filled and stroked.
Procedure :	`S*`	The path is open and not filled and stroked.
Procedure :	`b*`	The path is closed and filled and stroked.
Procedure :	`B*`	The path is open and filled and stroked.
Procedure :	`f*`	The path is closed and filled.
Procedure :	`F*`	The path is open and filled.
Procedure :	`h*`	The path is closed and not filled or stroked (i.e. invisible).
Procedure :	`H*`	The path is open and not filled or stroked (i.e. invisible).

Paths made up of more than one sub-path (Complex path)

Procedure :	***u**	Start of complex path
Procedure :	***U**	End of complex path

For example :

```
*u
108.4 58.4 m
164.934 64.053 226.373 73.75 281.2 88.8 c
S
96.4 122.4 m
96.4 194.4 l
209.2 194.4 l
209.2 122.4 l
96.4 122.4 l
s
*U
```

This is an example of a complex path made up of two sub-paths one closed and the other open. The closed path ends with the procedure **s**. Only the case of the end procedure name in a complex path may change to signify open or closed paths. The letter used will indicate whether the path is filled, stroked etc. In our example the complex path is only stroked.

Procedures that define a text object.

Procedure :	**atp**	(Text position)
Format :	**x y atp**	
Parameters :	**x-y**	: Position coords

Procedure :	**atf**	(Text font)
Format :	**/fontname atf**	
Parameters :	**/<font name>** :	Eg. **/AvantG.Book atf** would set the text font to AvantG.Book.

Procedure :	**atxy**	(Text X,Y text size)
Format :	**x y atxy**	
Parameters :	**x**	: Horizontal font size (in points)
	y	: Vertical font size (in points)

Procedure :	**atrk**	(Tracking control)
Format :	**cx cy ax ay atrk**	
Parameters :	**cx-cy**	: Inter-character x,y values (in ems)
	ax-ay	: Inter-word x,y values (in ems)

Procedure : **akrn** (Kerning attribute)
Format : **hem vem akrn**
Parameters : **hem** : Horizontal kerning in ems
vem : Vertical kerning in ems
(this is applied to the starting position of the second character after the attribute i.e. the same as ArtWorks)

Procedure : **Tx*** (Text)
Format : **string Tx**
Parameters : **string** : Text string enclosed in brackets

Procedure : **Tm** (Transformation matrix)
Format : **a b c d e f Tm**
Parameters : **a to f** : Matrix definition (e,f, specify the x,y position of the text string)
The matrix is only applied to the next text string.

Layer procedure

Procedure : **alyr** (Layer Object)
Format : **name type visible printOveride locked alyr**
Parameters : **name** : layer name enclosed in brackets
type : layer type
0 - A background layer
1 - A foreground layer
2 - A guide layer
visible : Visible flag
0 - layer is not visible
1 - layer is visible
printOveride : PrintOveride flag
0 - do not override printing
1 - override printing
locked : Locked flag
0 - layer is not locked
1 - layer is locked

Sprite procedure

Procedure :	`ass`	(Sprite format)
Format :	`bpp width height xeigen yeigen name x0 y0 x1 y1 x2 y2 x3 y3 mode wwidth sheight fb lb maskflag palflag ass`	
Parameters :	`bpp`	: Bits per pixel
	`width`	: Width in pixels
	`height`	: Height in pixels
	`x/yeigen`	: X & Y eigen factors
	`name`	: Sprite name max 12 characters
	`x0-y3`	: Four corner x,y coordinates
	`mode`	: Sprite mode
	`wwidth`	: Width in words -1
	`sheight`	: Number of scan lines -1
	`fb`	: First bit used (left end of row)
	`lb`	: Last bit used (right end of row)
	`maskflag`	: If set a mask included with sprite definition
	`palflag`	: If set the sprite has a palette

A palette is always included if the colours are referenced via indexes
The palette will have the following format:

Number of palette entries (0 when no palette entries i.e. the pixel colours are in rgb format (not used yet))

Each palette entry has the following format:
Raw palette entry as an eight digit hex value (c m y k colour entry)

The sprite image is based on the Acorn format and is simply each word exported as an eight digit hex value.

If the sprite has a mask the corresponding mask word will follow each image word.

Procedure :	`aes`	(End of sprite definition)
Format :	`aes`	
Parameters :	None.	

Section 3

ArtWorks comments

ArtWorks EPS Header

Comment : `%%Creator:`
Description : The first word following this will always be ArtWorks if the EPS file was generated by ArtWorks.

Comment : `%%Title:`
Description : The filename of the file that was exported as an ArtWorks EPS file.

Comment : `%%DocumentPath:`
Description : The full path name of the file that was exported as an ArtWorks EPS file.

Comment : `%%DocumentFonts:`
Description : A full list of all the fonts used in the ArtWorks EPS file.

Comment : `%%DocumentPageSize:`
Description : The page width and height in the current ArtWorks page units.

Comment : `%%AWColourTable:`
Description : (This is saved out on its own when generating a Colour Table file.)
If the colour is not a tint the first letter is the model (r-RGB, c-CMYK & h-HSV) the second letter is the colour type (p-process, s-spot) this is followed by the colour name and the colour definition (*see below for information on the colour definition*). If there are any tints based on the previous colour which can be a tint itself, then the first letter is a t followed by the tint level if it is greater than zero. If there is a tint defined as the base colour of another tint the tint level is increased. This is followed by the tint name and the percentage of the base colour (0-100).

If `o` follows, the overprint flag is set and applies to all objects using this colour.

If it is an RGB colour definition then the red, green & blue components follow in the range 0-1.

If it is a CMYK colour definition then the cyan, magenta, yellow & key components follow in the range 0-1.

If it is a HSV colour definition then the hue (0-360), saturation (0-100) & value (0-100).

Comment : **%%CreationDate:**
Description : The date when the ArtWorks document was created. If there is no date then when the EPS file was generated.

Comment : **%%BoundingBox:**
Description : The bounding box of all the objects in the EPS file (four coordinates low x,y and high x,y).

An example of an ArtWorks Colour Table

```
%%AWColourTable:
%%+c p (Black) 0 0 0 1
%%+t (90% Black) 90
%%+t (80% Black) 80
%%+t 1 (A tint of a tint) 50
%%+t (70% Black) 70
%%+c p (Cyan) 1 0 0 0
%%+c p (Magenta) 0 1 0 0
%%+c p o (Yellow) 0 0 1 0
%%+h p (A HSV colour) 190 65 52
%%+r p (A RGB colour) 0.46 0 0.52
```

This is a tint of (*80% Black*) which is itself a tint of (*Black*). Hence the tint level has been incremented.

A CMYK colour definition.
"Yellow" has overprint set.
A HSV colour definition.
A RGB colour definition.

Section 4

Procedures that can be used when importing

Theses are procedures that can be found in other EPS file formats that ArtWorks can import. Hence they will never appear in an ArtWorks EPS when generated by ArtWorks, thus no procedure definitions need to be written for them at present.

There are a number of features that ArtWorks lacks at the moment, an important one is clipping. Although there is no user interface internally, paths can be clipped to paths. At least three new procedures are needed to store clipping information. The following are understood by ArtWorks.

Procedure :	**q**	Save graphic state (*gsave*)
Procedure :	**Q**	Restore graphic state (*grestore*)
Procedure :	**W**	Use the previous path as the new clip path

There are other procedures that would be useful when converting file formats. One of which is **conc**. This defines a new translation matrix that is applied to all coordinates.

Procedure :	**conc**	Concatenate matrix with the previous one.
Format :	**a b c d e f conc**	
Parameters :	**a-f**	: Matrix definition [a,b,c,d,e,f] e & f are translation coordinates.
Procedure :	**ack**	Use current attributes when creating next object.
Format :	**ack**	
Parameters :	None.	For example, the text loader module uses it when you merge a text file.
Procedure :	**aps**	Used by Draw importer for 'path style' word.
Format :	**join endcap startcap winding tricapwidth tricapheight dashflag start numofels el_0..el_n aps**	
Parameters :	**join**	: Join style
	endcap	: End cap style
	startcap	: Start cap style
	winding	: Winding rule
	tricapwidth	:
	tricapheight	:
	dashflag	: Dash pattern included if set
	start	: Offset to start of dash pattern
	numofels	: Number of elements in the dash pattern
	el_0-el_n	: A list of the elements in the dash pattern

Procedure : **actr** (File creator)
Format : **createstr actr**
Parameters : **createstr**: The string contains the creator of the file currently being loaded. This can be sent by the loader module to all the object handlers to inform them so they know what to expect. For example, the EPS loader module uses it when importing CorelDraw EPS files, since only by reading the comments does the importer know which version of Corel produced it.

Procedure : **asbk**
Format : **asbk followed by a word aligned block**
Parameters : **A block** : The first word of the block should be its length.
The procedure can used by the object handler to generate a block of data that is already converted. For example, a draw object generated when converting a graduated fill. Generally this procedure should not be used.

Procedure : **asd** (Draw sprite format)
Format : **bpp w h name a b c d e f lx ly hx hy mode ww sh fb lb maskflag palflag ass**
Parameters :
bpp : Bits per pixel
w & h : Width & height
name : Sprite name max 12 characters
a-f : Sprite transformation matrix a,b,c,d,e,f
lx-hy : Sprite bounding box
mode : Sprite mode
ww : Width in words -1
sh : Number of scan lines -1
fb : First bit used (left end of row)
lb : Last bit used (right end of row)
maskflag : If set mask is included with sprite definition
palflag : If set the sprite has a palette

The sprite data follows without being converted. Thus allowing the draw saving to export sprites with very little effort.

This is sent instead of **ass** when f_OutputDraw is set in the flags word when broadcasting Method_TranslateInitialise

Procedure and Comment Index