# The xfig file format for xfig 3.2

Ernst Reissner (rei3ner@arcor.de)

January 1, 1970

## Contents



# 1 Introduction

This document is created with `lualatex` or that like with output format pdf. The package `tex4ht` is not loaded.

This document describes the grammar of the native file format of the xfig graphic program version 3.2.9. It is based on xfig.3.2.9/Doc/FORMAT3.2 in xfig 3.2.9 source distribution. The file extension is `.fig` and the file-format is human-readable. Note that compared to version 3.2.8a, in version 3.2.9 a comment specifying the encoding has been added. Treating comments as transparent, the format is unchanged.

If a line contains `#`, the rest of the line is ignored as regards the Lines starting with `#` are comment lines and are ignored. Also, xfig does not insert comments, except the first line as described in Section 2. On the other hand, if one writes comments, e.g. with an editor, these are essentially preserved when xfig reads the file and writes it (well, if the first character of a comment is not a blank, a blank is added, but that is all what is changed). Note that xfig accepts

lines with trailing comments, but does not preserve these kinds of comments. Also, comments with leading blanks are not allowed. So comments occur line wise.

Although the format is human-readable, it is not designed to make it easy to interpret the content and to obtain an overview. Thus, comments may be valuable aid to give a notion of the intent behind the various elements contained.

As comments, also empty lines are ignored, but xfig does not preserve them when writing back the file.

Each xfig-file starts with a header as described in Section 2 followed by a (typically non-empty) sequence of object descriptions. The header specifies general attributes which apply to the whole document, whereas the object descriptions refer to individual objects.

For each type of object, there is a separate grammar, described in Section 3 and subsections. There are elements occurring for more than one type of object. To avoid repetition in the description, these elements are described separately in Section 4 and referenced from within Section 3.

Note that the header may be interpreted line wise and also both objects and their parameters are described as lines. Apart from comment lines, the entries of each line are separated by blanks. When reading a fig-file, xfig interprets a string of subsequent blanks like one blank, except for text strings in text objects (see Section 3.4). Consequently, writing back, each such maximal sequence of blanks is replaced by a single blank. Trailing blanks are not preserved.

Leading blanks do not occur in the header. Lines with leading blanks in the header are ignored. Thus, e.g. the orientation line “ `Landscape`” is replaced by “`Portrait`”, which demands the default orientation.

Whereas each attribute given in the header is described by a single line, for various object types a single line is not sufficient. Each object description may start with an indefinite number of comment lines. The first mandatory line of an object description is called the *headline*. The headline does not start with a blank. When reading subsequent lines referring to the same object xfig accepts a non-empty sequence of blanks, but when writing, such lines start with a single tab character, in some cases followed by a single blank.

There is a single exception to this rule: Compound objects as described in Section 3.6 which turns a sequence of objects described by subsequent lines into a unique object described by a start line and an end line specific for that compound. Both, start line and end line does not allow a leading blank.

### 1.1 X-Splines and backwards-compatibility

Splines are always X-splines which allows the user to mix interpolation and approximation points in a same curve. More precisely, it means that an X-spline curve is neither an interpolated spline nor an approximated one, it is BOTH (the behavior of each point is controlled by one single parameter called “shape factor”). For additional information about X-splines, see [BS95].

Caveat: Because spline models of previous versions (quadratic B-splines and Bezier with hidden points) are no longer supported, curves that are present in version 3.1 and older files are automatically converted to X-splines. This translation is only an approximation process. It means that the converted curves are not exactly the same as the original ones. Though the translation usually provides almost identical curves, some hand-fitting may be needed in some pathological cases.

## 2 Header

The very first line is comment line containing the name and version:

```
#FIG 3.2  Produced by xfig version 3.2.9
#encoding: UTF-8
```

The subsequent non-comment lines of the header are the following (in the order given):

| type | name | Possible values or explanation |
|---|---|---|
| string | Orientation | 'Landscape'\|'Portrait' |
| string | Justification | 'Center' or 'Flush Left' |
| string | Units | 'Metric' or 'Inches' |
| string | PaperSize | 'Letter'\|'Legal'\|'Ledger'\|'Tabloid'\| 'A'\|'B'\|'C'\|'D'\|'E'\| 'A4'\|'A3'\|'A2'\|'A1'\|'A0'\|'B5' |
| float | Magnification | export and print magnification, % |
| string | MultiplePage | 'Single' or 'Multiple' pages |
| int | TransparentColor | color number for transparent color for GIF export. -2=None, -1=background, 0–31 for standard colors or 32– for user colors |
| string | [Comment] | optional comment |
| int | Resolution coord-system | Fig units/inch and coordinate system: 1: origin at lower left corner (NOT USED) 2: upper left |

Fig 'resolution' is the resolution of the figure in the file. Xfig will always write the file with a resolution of 1200ppi so it will scale the figure upon reading it in if its resolution is different from 1200ppi. Pixels are assumed to be square.

Xfig will read the orientation string and change the canvas to match either the Landscape or Portrait mode of the figure file.

The specification of the units is self-explanatory.

The coordinate-system variable is ignored — the origin is ALWAYS the upper-left corner.

** Coordinates are given in 'fig_resolution' units. ** Line thicknesses are given in 80-ths of an inch ('display units'). ** dash-lengths/dot-gaps are given in 80-ths of an inch.

`https://hackage.haskell.org/package/fig-1.4.0/docs/Graphics-Fig-Syntax.html`

Orientation: ('Landscape'|'Portrait')
Justification: ('Center'|'FlushLeft')
Units: ('Metric'|'Inches')
Transparent: Background None TransparentDefault Transparent ColorSpec
CoordinateSystem: LowerLeft UpperLeft (both are integers)
Commented: Comment [String] a
Color: color_number:: Integer color_rgb_values:: String
Documentation
data Fig Source
Fig figHeader:: Header figColors:: [Commented Color] figObjects:: [Commented Object]

# 3 Objects

The rest of the file contains various objects. An object can be one of six classes (or types) with according `ObjectCode`.

0 Color pseudo-object (see Section 3.0).

1. Ellipse (see Section 3.1) which is a generalization of circle.
2. Polyline (see Section 3.2) which comprises polygon, box and also included pictures.
3. Spline (see Section 3.3) which may be closed, open approximated, interpolated or an x-spline.
4. Text (see Section 3.4).
5. Arc (see Section 3.5).
6. Compound object (see Section 3.6) which is composed of one or more objects.

In the following elaboration on object formats, every value of fig output are separated by blank characters or new line ('\n'). The value of the unused parameters will be -1.

Some fields are described as 'enumeration type' or 'bit vector'; the values which these fields can take are defined in the header file object.h. The pen_style field is unused. These values may be defined in some future version of Fig.

## 3.0 Color Pseudo-Object

Color Pseudo-objects (user-defined colors) define color numbers in terms of rgb-values. This is used to define arbitrary colors beyond the 32 standard colors. See also Section 4.4. The color objects must be defined before any other Fig objects.

Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 0 (see Section 3) |
| int | ColorNumber | color number, from 32–543 (512 total) |
| hex-string | RgbValues | red, green and blue values (e.g. #330099) |

The `Headline` ist the sole line defining a Color Pseudo-Object.

## 3.1 Ellipse including Circle

Defines an ellipse in terms of center and vertex of the bounding box or in terms of two diagonal vertices of the bounding box. Defines also a circle in terms of center and a point on the circle or in terms of two points on the circle.

The `Headline` following now is the sole line defining an ellipse.

| Type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 1 (see Section 3) |
| int | SubType | 1: ellipse defined by “radiuses” |
| | | 2: ellipse defined by “diameters” |
| | | 3: circle defined by radius |
| | | 4: circle defined by diameter |
| int | LineStyle | see Section 4.1 |
| int | LineThickness | multiples of 1/80 inch |
| int | PenColor | see Section 4.4 |
| int | FillColor | see Section 4.4 |
| int | Depth | see Section 4.1 |
| int | PenStyle | -1, not used |
| int | AreaFill | see Section 4.4 |
| float | StyleVal | multiples of 1/80 inch, see Section 4.1 |
| int | Direction | always 1 |
| float | Angle | radians, the angle of the x-axis |
| int | center-x, center-y | Fig units |
| int | radius-x, radius-y | Fig units (same for circles) |
| int | start-x, start-y | Fig units; the 1st point entered |
| int | end-x, end-y | Fig units; the last point entered |

For further explanations on start-x/y and end-x/y:

| SubType | start-x, start-y | end-x, end-y |
|---|---|---|
| 1 | the center | some vertex of the BB |
| 2 | a vertex of the BB | opposite vertex of the BB |
| 3 | the center | a point on the circle |
| 4 | a point on the circle | opposite point on the circle |

Ellipse

ellipse_common:: Common ellipse_direction:: Integer ellipse_angle:: Double ellipse_center_x:: Integer ellipse_center_y:: Integer ellipse_radius_x:: Integer ellipse_radius_y:: Integer ellipse_start_x:: Integer ellipse_start_y:: Integer ellipse_end_x:: Integer ellipse_end_y:: Integer

## 3.2 Polygonal line, Box and included Picture

Defines various kinds of polygonal lines, both open and closed. As special cases also boxes and arc-boxes, i.e. boxes with rounded vertices. As a special case also a box which serves as bounding box for a picture. In this case, this polygonal line finally defines the picture itself. See `SubType` in the Headline below.
Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 2 (see Section 3) |
| int | SubType | 1: polyline (open) |
| | | 2: box |
| | | 3: polygon (closed, regular or not) |
| | | 4: arc-box (box with round vertices) |
| | | 5: imported-picture bounding-box |
| int | LineStyle | see Section 4.1 |
| int | LineThickness | multiples of 1/80 inch |
| int | PenColor | see Section 4.4 |
| int | FillColor | see Section 4.4 |
| int | Depth | see Section 4.1 |
| int | PenStyle | -1, not used |
| int | AreaFill | see Section 4.4 |
| float | StyleVal | multiples of 1/80 inch, |
| | | see Section 4.1 |
| int | JoinStyle | 0: Miter (the default in xfig 2.1 and earlier) |
| | | 1: Bevel |
| | | 2: Round |
| int | CapStyle | see Section 4.1, |
| | | only used for subtype POLYLINE |
| int | Radius | radius of arc-boxes as multiples of 1/80 inch |
| int | ForwardArrow | 0: off, 1: on |
| int | BackwardArrow | 0: off, 1: on |
| int | NPoints | number of points in line |

After the Headline follows

- the `ForwardArrowLine` only if `ForwardArrow` is not 0 and
- the `BackwardArrowLine` only if `BackwardArrow` is not 0;

both described in Section 4.2.

Only if `SubType` in the Headline is 5, representing an included picture, follows a `PicLine` with the following form:

| type | name | Possible values or explanation |
|---|---|---|
| boolean | Orientation | 0: normal |
| | | 1: flipped at diagonal |
| string | File | name of picture file to import |

The allowed file types are eps/ps, pdf, gif, jpg, pcx, png, ppm, tiff, xmb and xpm.

Then follows a `PointsLine` with a number of coordinates given by `NPoints` in the first line. If the `SubType` is 5, the `PointsLine` defines the boundary of the picture and also rotation. The form of the `PointsLine` is given in Section 4.3. This is the same as for Polylines described in Section 3.3.

## 3.3 Spline

Defines various kinds of splines, open or closed, interpolated or approximated or something in between as x-spline.

Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 3 (see Section 3) |
| int | SubType | 0: opened approximated spline |
| | | 1: closed approximated spline |
| | | 2: opened interpolated spline |
| | | 3: closed interpolated spline |
| | | 4: opened x-spline |
| | | 5: closed x-spline |
| int | LineStyle | see Section 4.1 |
| int | LineThickness | multiples of 1/80 inch |
| int | PenColor | see Section 4.4 |
| int | FillColor | see Section 4.4 |
| int | Depth | see Section 4.1 |
| int | PenStyle | -1, not used |
| int | AreaFill | see Section 4.4 |
| float | StyleVal | multiples of 1/80 inch, see Section 4.1 |
| int | CapStyle | Section 4.1, only used for open splines |
| int | ForwardArrow | 0: off, 1: on |
| int | BackwardArrow | 0: off, 1: on |
| int | NPoints | number of control points in spline |

After the first line follows

- the `ForwardArrowLine` only if `ForwardArrow` is not 0 and
- the `BackwardArrowLine` only if `BackwardArrow` is not 0;

both described in Section 4.2.

Then follows a `PointsLine` with a number of coordinates given by `NPoints` in the first line. The form of the `PointsLine` is given in Section 4.3. This is the same as for Polylines described in Section 3.2.

Finally, and unlike for Polylines, there comes the `ControlPointsLine`:

There is one shape factor for each point in the `PointsLine`, i.e. `NPoints` shape factors. The value of this factor must be between

-1 which means that the spline is interpolated at this point and

+1 which means that the spline is approximated at this point.

The spline is always smooth in the neighbourhood of a control point, except when the value of the factor is 0 for which there is a first-order discontinuity (i.e. angular point).

## 3.4 Text

A Text object, defines a text element with text, font, font size and many other attributes. Note that horizontal justification is defined, but not vertical justification.
Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 4 (see Section 3) |
| int | SubType | 0: Left justified |
| | | 1: Center justified |
| | | 2: Right justified |
| int | PenColor | see Section 4.4 |
| int | Depth | see Section 4.1 |
| int | PenStyle | -1, not used |
| int | Font | see below |
| float | FontSize | font size in points |
| float | Angle | radians, the angle of the text |
| int | FontFlags | three bit vector (see below) |
| float | height | Fig units |
| float | length | Fig units |
| int | x, y | Fig units, coordinate of the origin of the string. |
| | | If SubType is 0/1/2, it is the |
| | | lower left/center/right corner of the string. |
| string | Text | see below |

The `Headline` ist the sole line defining a text object.
The bits of the `FontFlags` field are defined as follows:

0 Rigid text: text doesn't scale when scaling compound objects

1. Special text: interpret test as for LaTeX-code. If this is set, the following flag affects the view on the xfig GUI but not external view, e.g. when printing or exporting.

2. PostScript font: interpret `Font` above as Postscript font (otherwise as LaTeX-font) Details follow below.

3. Hidden text: whether the text is hidden in the xfig GUI. This does not affect the external view, e.g. printing and export.

The `Font` field is immaterial, if the Special flag above is set. Otherwise, its meaning depends on the PostScrpt/LaTeX-flag described above:
For `FontFlags` bit 2 = 0, `Font` is interpreted as LaTeX-font with the following encoding:

0 LaTeX Default font

1. Roman

2. Bold

3. Italic

4. Sans Serif

5. Typewriter

For `FontFlags` bit 2 = 1, `Font` is interpreted as PostScript-font with the following encoding:

-1 PostScript Default font

0 Times Roman

1. Times Italic
2. Times Bold
3. Times Bold Italic
4. AvantGarde Book
5. AvantGarde Book Oblique
6. AvantGarde Demi
7. AvantGarde Demi Oblique
8. Bookman Light
9. Bookman Light Italic
10. Bookman Demi
11. Bookman Demi Italic
12. Courier
13. Courier Oblique
14. Courier Bold
15. Courier Bold Oblique
16. Helvetica
17. Helvetica Oblique
18. Helvetica Bold
19. Helvetica Bold Oblique
20. Helvetica Narrow
21. Helvetica Narrow Oblique
22. Helvetica Narrow Bold
23. Helvetica Narrow Bold Oblique
24. New Century Schoolbook Roman
25. New Century Schoolbook Italic
26. New Century Schoolbook Bold
27. New Century Schoolbook Bold Italic
28. Palatino Roman
29. Palatino Italic
30. Palatino Bold

31. Palatino Bold Italic
32. Symbol
33. Zapf Chancery Medium Italic
34. Zapf Dingbats

The `Text` field above is a sequence of ASCII characters. It starts after a *single* blank character which separates the coordinate field `x, y` from the `Text` field. This allows to specify texts with leading blanks. The text field ends before the sequence "`\001`"; the latter sequence is not part of the text field. Characters above octal 177 are represented by "`\xxx`" where `xxx` is the octal value. This permits fig files to be edited with 7-bit editors and sent by e-mail without data loss. Note that the `Text` field may contain "`\n`". To specify a literal backslash write "`\\`". The latter is vital for including LaTeX in fig-figures.

ColorSpec: see Section 4.4 Font: Latex LatexFont Ps PsFont
LatexFont: LatexDefault Roman Bold Italic SansSerif Typewriter
FontFlags: hidden:: Bool special:: Bool rigid:: Bool

## 3.5 Arc

Defines an arc, i.e. a segment of a circle by a start point, an intermediate point and an end point.
Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 5 (see Section 3) |
| int | SubType | 0: closed (pie-wedge) |
| | | 1: open ended |
| int | LineStyle | see Section 4.1 |
| int | LineThickness | multiples of 1/80 inch |
| int | PenColor | see Section 4.4 |
| int | FillColor | see Section 4.4 |
| int | Depth | see Section 4.1 |
| int | PenStyle | -1, not used |
| int | AreaFill | see Section 4.4 |
| float | StyleVal | multiples of 1/80 inch, see Section 4.1 |
| int | CapStyle | see Section 4.1 |
| int | Direction | 0: clockwise, 1: counterclockwise |
| int | ForwardArrow | 0: off, 1: on |
| int | BackwardArrow | 0: off, 1: on |
| float | center-x, center-y | center of the arc |
| int | x1, y1 | Fig units, the 1st point the user entered |
| int | x2, y2 | Fig units, the 2nd point |
| int | x3, y3 | Fig units, the last point |

After the first line follows

- the `ForwardArrowLine` only if `ForwardArrow` is not 0 and
- the `BackwardArrowLine` only if `BackwardArrow` is not 0;

both described in Section 4.2.

Arc ArcLine (Maybe Arrow) (Maybe Arrow) Spline SplineLine (Maybe Arrow) (Maybe Arrow) [(Integer, Integer)] [Double]

SplineLine: spline_common:: Common spline_cap_style:: CapStyle

Arrow: arrowType:: ArrowType arrowStyle:: ArrowStyle arrowThickness:: Double arrowWidth:: Double arrowHeight:: Double

ArrowStyle: HollowArrow FilledArrow

ArrowType: Stick Closed Indented Pointed

### 3.6 Compound

Defines the composition of other objects, which may well be compounds themselves.

Headline:

| type | name | Possible values or explanation |
|---|---|---|
| int | ObjectCode | always 6 (see Section 3) |
| int | UpperRight-corner-x | Fig units |
| int | UpperRight-corner-y | Fig units |
| int | LowerLeft-corner-x | Fig units |
| int | LowerLeft-corner-y | Fig units |

Subsequent lines of the compound except the last one define a list of objects in the Compound These objects may well be Compounds themselves. The last line is `-6`.

CapStyle: see Section 4.1

Common: subType:: Integer lineStyle:: LineStyle lineThickness:: Integer penColor:: ColorSpec fillColor:: ColorSpec depth:: Integer penStyle:: Integer areaFill:: AreaFill styleVal:: Double

LineStyle: LineStyleDefault Solid Dashed Dotted DashDotted DashDoubleDotted DashTripleDotted

AreaFill: (NoFill|Filled Integer|Pattern Integer)

## 4 Parameters

### 4.1 Style and Depth

The `CapStyle` field is defined FOR LINES, OPEN SPLINES and ARCS only used in ArcLine, SplineLine, PolylineLine and is encoded as follows:

0 Butt (the default in xfig 2.1 and earlier)

1. Round

2. Projecting

The `LineStyle` field is encoded as follows:

-1 Default

0 Solid

1. Dashed

2. Dotted

3. Dash-dotted

4. Dash-double-dotted

5. Dash-triple-dotted

The `StyleVal` field is defined as

- the length, in 1/80 inches, of the on/off dashes for dashed lines, and
- the distance between the dots, in 1/80 inches, for dotted lines.

The `Depth` field is defined as follows:
0 …999 where larger value means object is deeper than (under) objects with smaller depth

## 4.2 Arrow lines

Forward arrow line (Optional; absent if ForwardArrow is 0):

| type | name | Possible values or explanation |
|---|---|---|
| int | ArrowType | see below |
| int | ArrowStyle | see below |
| float | ArrowThickness | multiples of 1/80 inch |
| float | ArrowWidth | Fig units |
| float | ArrowHeight | Fig units |

Backward arrow line (Optional; absent if backward-arrow is 0):

| type | name | Possible values or explanation |
|---|---|---|
| int | ArrowType | see below |
| int | ArrowStyle | see below |
| float | ArrowThickness | multiples of 1/80 inch |
| float | ArrowWidth | Fig units |
| float | ArrowHeight | Fig units |

The `ArrowType` field is defined for LINES, ARCS and OPEN SPLINES only as follows:

0 Stick-type (the default in xfig 2.1 and earlier)

1. Closed triangle:

2. Closed with 'indented' butt:

3. Closed with 'pointed' butt:

The `ArrowStyle` field is defined for LINES, ARCS and OPEN SPLINES only as follows:

0 Hollow (actually filled with white)

1. Filled with `PenColor` defined in Section 4.4

## 4.3 PointsLine

The form of the `PointsLine` is as follows:

| type | name | Possible values or explanation |
|---|---|---|
| int | x1, y1 | Fig units |
| int | x2, y2 | Fig units |
| ⋮ | ⋮ | ⋮ |
| int | xnpoints ynpoints | this will be the same as the 1st point<br>if the described line is closed<br>and in particular if it is the outline of a picture |

## 4.4 Colors and Filling

Both `PenColor` and `FillColor` are encoded as follows:

-1 Default

0 Black

1. Blue

2. Green

3. Cyan

4. Red

5. Magenta

6. Yellow

7. White

8–11 four shades of blue (dark to lighter)

12–14 three shades of green (dark to lighter)

15–17 three shades of cyan (dark to lighter)

18–20 three shades of red (dark to lighter)

21–23 three shades of magenta (dark to lighter)

24–26 three shades of brown (dark to lighter)

27–30 four shades of pink (dark to lighter)

31 Gold

32–543 (512 total) are user colors and are defined in color pseudo-objects described in Section 3.0. Only those indices may be used which are previously defined in some color pseudo-object.

The field `AreaFill` is defined depending on the value of `FillColor` (which is defined for all kinds of objects for which `AreaFill` is defined).

For all `FillColor`s except `Black` or `Default` and `White`, the `AreaFill` pattern is defined as follows:

-1 not filled

0 black

1–19 'shades' of the `FillColor`, from darker to lighter. A shade is defined as the color mixed with black

20 full saturation of the color

21–39 'tints' of the color from the color to white. A tint is defined as the color mixed with white

40 white

41 30 degree left diagonal pattern

42 30 degree right diagonal pattern

43 30 degree crosshatch

44 45 degree left diagonal pattern

45 45 degree right diagonal pattern

46 45 degree crosshatch

47 bricks

48 circles

49 horizontal lines

50 vertical lines

51 crosshatch

52 fish scales

53 small fish scales

54 octagons

55 horizontal 'tire treads'

56 vertical 'tire treads'

For `FillColor`s `Black` and `Default`, there are the following deviations:

-1 not filled

0 white

1–19 'shades' of gray, from lighter to darker.

20 (full saturation of) black

21–40 not used.

If the `FillColor` is `White` there are the following deviations from standard colors:

-1 not filled

0 black

1–19 'shades' of gray, from darker to lighter.

20 (full saturation of) white

21–40 not used.

used in Text and Common,
ColorSpec: ColorSpecDefault Black Blue Green Cyan Red Magenta Yellow White Blue4 Blue3 Blue2 LtBlue Green4 Green3 Green2 Cyan4 Cyan3 Cyan2 Red4 Red3 Red2 Magenta4 Magenta3 Magenta2 Brown4 Brown3 Brown2 Pink4 Pink3 Pink2 Pink Gold UserDefined Integer

# 5 References

[BS95] Carole Blanc and Christophe Schlick. X-splines: A spline model designed for the end-user. *Proceedings of SIGGRAPH*, pages 377–386, 8 1995. LaBRI: Laboratoire Bordelais de Recherche en Informatique (Université Bordeaux I and Centre National de la Recherche Scientifique).