

WELCOME TO EZIP

Welcome the Easy Print Engine. Designed to give 32 or 64 bit Freebasic Windows programs an easy and versatile way to quickly add printing capabilities to a program. EZIP is actual has 2 functionalities; it can create documents in its proprietary format and then it can print preview those documents before printing. Though EZIP is not a designer, its use of a grid greatly smooths the way to nice looking documents quickly. EZIP is based on wide strings and readily accepts inches or centimeters as the measurement format to place objects on a page and then view them on a print preview dialog or immediately print. EZIP though, does store all measurements internally as inches. The print commands are saved to a file which can be saved for future use or simply be printed and then automatically deleted. The print engine comes as an include file so it should be seamless to add to your application. Plus the engine's dimensions can be modified by changing the program **#defined** attributes such the maximum number of bookmarked pages etc.

If someone want to dig into the EZIP control, you should be able to modify EZIP to your own needs. Have fun using EZIP.

PRINT ENGINE WORKFLOW

To use EZIP simply:

1) include the EZIP.inc after the Windows include file

```
#INCLUDE ONCE "windows.bi"
```

```
#INCLUDE ONCE "EZIP.inc"
```

2) If you want to make a new document then it is done with a number of commands that

a) Begin the document which starts a new file

Describe the document : paper, orientation, inches or cm, tray etc. and give it a file name

b) Add commands to the first page with a font command and grid command being your first.

c) Make a new page (which can be bookmarked)

d) Add commands to that page and so on

e) End the document which closes the file and is now saved for future use

You can add end attributes to automatically view the document, print the document and to delete the document immediately after viewing or printing

If you want to print an existing file then call the DoDoc routine.

```
hFile = EZIP_DocBegin("MyDocTest.ezp",EZIP_INCH,EZIP_PAPER_LETTER,EZIP_PORTRAIT,EZIP_BIN_AUTO)
IF hFile <> INVALID_HANDLE_VALUE THEN

    EZIP_DocDateStamp(hFile,"June 8th, 2017")
    EZIP_DocFont(hFile,EZIP_ITALIC OR EZIP_UNDER,12,EZIP_RED,0,"Arial")
    MyBuffer = "Sample Text"
    EZIP_DocText(hFile,EZIP_G(5,2,EZIP_r),EZIP_G(5,2,EZIP_t),VARPTR(MyBuffer))

    MyBuffer = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quae cum praeponun
    MyBuffer = MyBuffer & "Duo Reges: constructio interrete. Ex rebus enim timiditas, non e
    MyBuffer = MyBuffer & "Nos quidem Virtutes sic natae sumus, ut tibi serviremus, aliud n
    EZIP_DocTextExRead(hFile,VARPTR(MyBuffer))

    EZIP_DocFont(hFile,EZIP_NORMAL,12,EZIP_BLACK,0,"Arial")
    EZIP_DocTextEx(hFile,3,0.5,6)

    EZIP_DocFont(hFile,EZIP_ITALIC OR EZIP_UNDER,12,EZIP_RED,0,"Arial")
    MyBuffer = "Hello World"
    EZIP_DocText(hFile,2,2,VARPTR(MyBuffer))
    EZIP_DocLine(hFile,1,3,2,4,1,EZIP_PDASH,EZIP_BLACK)
    EZIP_DocRect(hFile,5,5,6,6,EZIP_BHATCHED,EZIP_RED,1,EZIP_BLACK)
    EZIP_DocEllipse(hFile,5,3,6,4,EZIP_BSOLID,EZIP_BLUE,1,EZIP_BLACK)
    EZIP_DocFont(hFile,EZIP_NORMAL,24,EZIP_BLACK,0,"Times New Roman")
    EZIP_DocHeader(hFile,LEFT(*EZIP.TextPtr,11),0.5,1,8,2,EZIP_PAGENUMLEFT + EZIP_DATERIGHT,8)
    EZIP_DocFont(hFile,EZIP_NORMAL,8,EZIP_BLACK,0,"Times New Roman")
    EZIP_DocFooter(hFile,"",0.5,10,8,10.5,EZIP_PAGENUMRIGHT,12)

    EZIP_DocNewPage(hFile,"MyPage2") ' Bookmark the page
    EZIP_DocFont(hFile,EZIP_ITALIC OR EZIP_UNDER,18,EZIP_BLUE,0,"Arial")
    MyBuffer = "My Second Page"
    EZIP_DocText(hFile,3,3,VARPTR(MyBuffer))

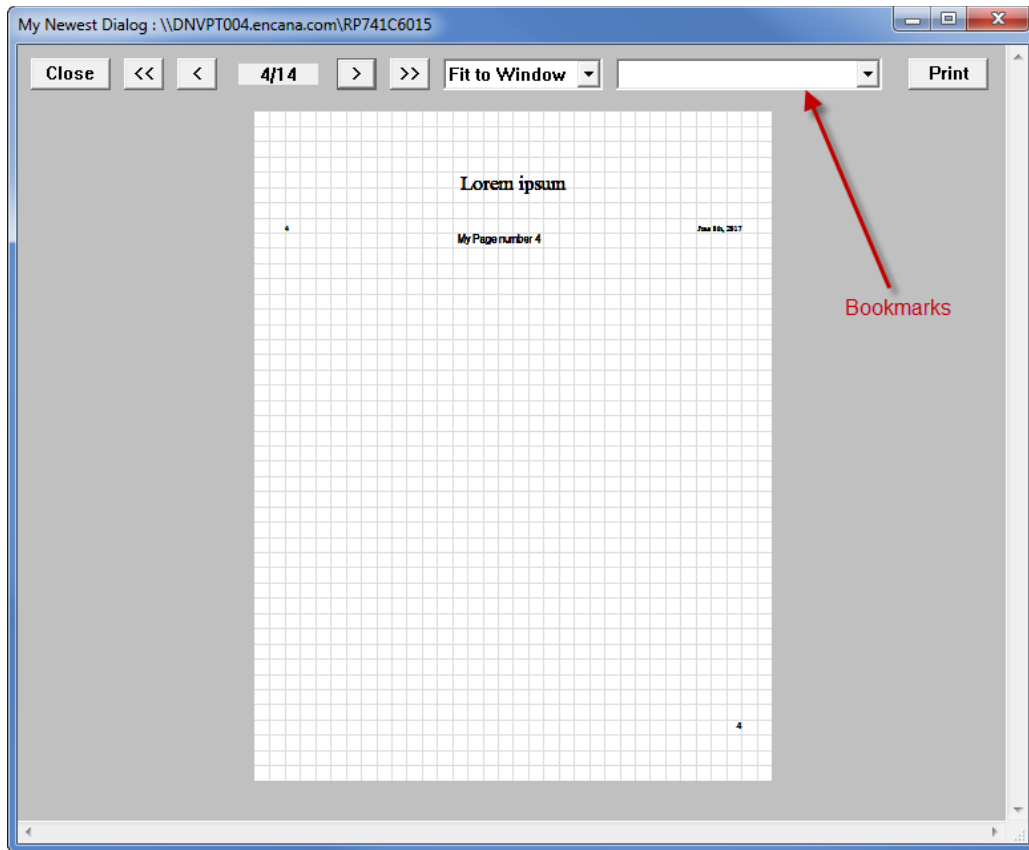
    'Insert and draw a graphic file
    bm = EZIP_DocAddImage(hFile,"Test.png")
    EZIP_DocDrawImage(hFile,bm,0,2,2,2,2,1)

    EZIP_DocEnd(hFile,EZIP_END_VIEW + EZIP_END_DELETE)
END IF

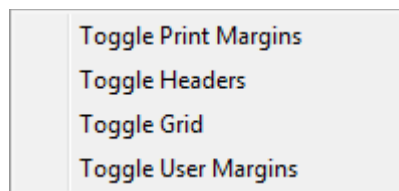
EZIP_DoDoc (HWND,0,"MyDocTest.ezp",EZIP_VIEWER_GRID + EZIP_VIEWER_ZOOMFIT)
```

PRINT PREVIEW DIALOG

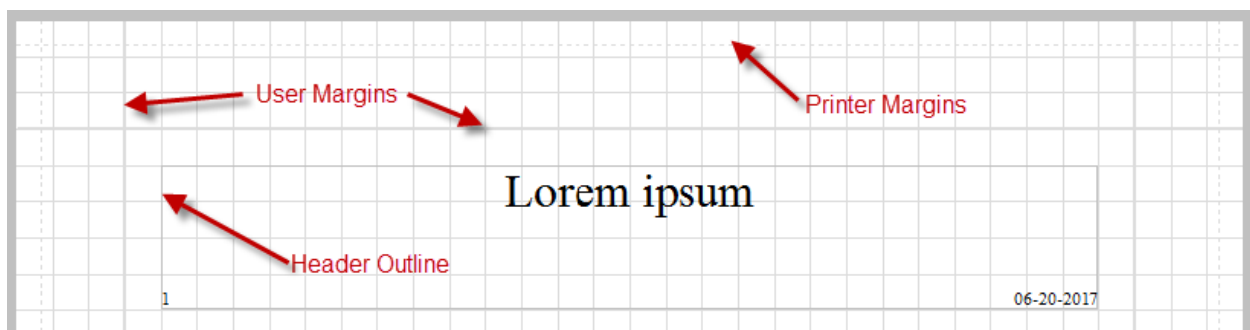
When DoDoc is called, the routine looks at the document's end statement to see what needs to be done. These end statement attribute can be over-ridden with the EZIP_IGNOREFILEEND switch. If the file name is not given (""), you will be presented with an Open File Dialog. Normally, the document file will be print previewed and the document will be shown as seen below. Each print can be viewed using the transport buttons, the document can be resized, if bookmarks exists, that page can easily be found, and if the document wishes to be printed then the Print button can be pushed to display the print dialog. In the screen shot below, DoDoc was set to automatically show the grid and fit the document to the window. The Print Preview dialog is set to resize to the monitor size.



Right Click Button Menu to control visual outlines:



```
EZP_DocFont(hFile,EZP_NORMAL,24,EZP_BLACK,0,"Times New Roman")
EZP_DocHeader(hFile,LEFT(*EZP.TextPtr,11),1.00,1,7.5,2,EZP_PAGENUMLEFT + EZP_DATERIGHT,8)
```



PRINT ENGINE GRID

EZP can place objects on a page by directly defining the x and y coordinates from the upper-left corner of the page, in inches or centimeters (as dictated in the EZP_DocBegin command) or to use the corners of cells in the grid to find a location. The grid is initialized with 0.25 inch cell heights and widths. This initial dimensions can be changed by editing the following define statements which need to be in inches since EZP stores all numbers internally as inches. If you are working in centimeters then convert your desired default numbers to inches.

```
#define EZP_GRIDCOLDEFAULT 0.25
```

```
#define EZP_GRIDROWDEFAULT 0.25
```

The predefined grid should be big enough for all be the finest gridded applications but can be changed:

```
#define EZP_MAXNUMCOLS 300
```

```
#define EZP_MAXNUMROWS 1000
```

One of the first commands may be to resize the grid to your liking just like you would do with a spreadsheet. This is done with the EZP_DocSetGrid command with the EZP_GRIDNEW action. If you are working in inches and desire the base grid to be sized on 1 inch by 1 inch cells, then the command would be:

```
EZP_DocGridSet(hFile, EZP_GRIDNEW,1,1)
```

Individual row and columns can be resized using the EZP_GRIDMOD action. The following command sets row 2 to 3.75 inches.

```
EZP_DocGridSet(hFile, EZP_GRIDMOD,2,0,3.75,0)
```

See the EZPrintDemoLabels.bas file to see an application of resizing the grid.

The corners of each cell can be accessed using the EZP_G statement. EZP_G need the row, column and which corner. Here is the statement in action

```
EZP_DocText(hFile,EZP_G(MyRow,MyCol,EZP_l) + 1.5,EZP_G(MyRow,MyCol,EZP_t) +  
0.9,VARPTR(MyBuffer))
```

Some macros have been predefined to cut down on the amount of typing

```
#define ezt(r,c) EZP_G(r,c,0) top of the cell
```

```
#define ezl(r,c) EZP_G(r,c,1) left of the cell
```

```
#define ezb(r,c) EZP_G(r,c,2) bottom of the cell
```

```
#define ezr(r,c) E郑_G(r,c,3) right of the cell
```

Unlike most E郑 commands which use the x,y format order, the grid location commands are in a row, column order like Excel commands. Just something to keep in mind.

PRINT DIALOG

The print dialog can be called from a directive from the document end command action and from the Print Preview Dialog. E郑 defaults to printing to the current default printer on your system. The print dialog box allows for selection of a printer besides the default printer. Also from the print dialog the user can also select a specific print range. E郑 is set up for up to 10 print ranges. The paper, orientation, tray (bin) , double size printing options are all defined in the E郑_DocBegin statement and are a part of the document. It is possible for the user to change these parameters but not encouraged.

PRINT ENGINE KEYWORDS

These keywords are directives and place objects on a page.

E郑_DocAddImage

```
FUNCTION E郑_DocAddImage(hFile as HANDLE, MyBitmapFile as WSTRING) as LONG
```

This function adds an image to the file buffer and returns a number which reference the image for use with E郑_DocDrawImage. The maximum number of graphic images are set the the define statement

```
#define E郑_MAX_GRAPHICS 50
```

This function supports the following file types - BMP, GIF, JPEG, PNG, TIFF, Exif, WMF, and EMF - and stores them internally as a bitmap.

Return Value	Description
bm	Number of the bitmap stored in the document files

```
DIM bm AS LONG
```

```
bm = E郑_DocAddImage(hFile, "Test.png")
```

E郑_DocArc

```
SUB E郑_DocArc(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, x2 AS
SINGLE, y2 AS SINGLE, X1Arc AS SINGLE, Y1Arc AS SINGLE, x2Arc AS
SINGLE ,Y2Arc AS SINGLE, MyWidth AS SINGLE, MyStyle AS LONG, MyColor
AS LONG)
```

The Arc subroutine draws an elliptical arc which is part of the Windows SDK. The width of the pen is MyWidth. MyStyle is the pen style and MyColor is the color of the brush.

```
E郑_DocArc(hFile,4,6,5,7,4.4,6.2,4.9,6.9,2,0,E郑_BLUE,E郑_BLUE)
```

E郑_DocBegin

```
FUNCTION E郑_DocBegin(StrFile AS WSTRING,InchOrCm AS LONG, _
WhichPaper AS LONG, WhichOrient AS LONG, WhichBin AS LONG =
0,DoDuplex AS LONG = 0) AS HANDLE
```

This function starts the document and returns a file handle which will be needed by all the commands when constructing the document. Be careful of defining DoDuplex due to forcing a non duplexing printer and giving non desired results.

Parameter	Description
StrFile	The name of the file in the form of a Wstring. A suffix .ezp is recommended.
InchOrCm	Whether Inches or Centimeters will be used to locate objects on a page. E郑_INCH or E郑_CM
WhichPaper	The coordinates of the lower-right corner of the bounding rectangle in which to print the bitmap. See Pre-defined paper constants below
WhichOrient	The name of the graphic to embed and display. E郑_PORTRAIN or E郑_LANDSCAPE
WhichBin	Which tray. Usually needs to be defined for printing envelopes. Zero is printer default.
DoDuplex	E郑_DUPLEX_NONE E郑_DUPLEX_HORIZONTAL E郑_DUPLEX_VERTICAL

Return Value	Description
hFile	Valid File Handle– Success. INVALID_HANDLE_VALUE - Failure

```
hFile = E郑_DocBegin("MyDocTest.ezp", E郑_INCH, E郑_PAPER_LETTER,
E郑_PORTRAIT, E郑_BIN_AUTO)
```

```
IF hFile <> INVALID_HANDLE_VALUE THEN
```

```
    `Put Commands here
```

```
END IF
```

EZP_DocChangeNames

```
SUB EZP_DocChangeNames(hFile AS HANDLE, MyPrintName AS  
WSTRING, MyCloseName AS WSTRING, MyWindowFitName AS  
WSTRING, MyWidthFitName AS WSTRING)
```

This routine changes the names in the Print Preview dialog for non-English speaking applications. The “Print Button” is changed to MyPrintName, “Close Button” is changed to MyCloseName. In the document sizing dropdown control the “Fit to Window” selection is changed to MyWindowFitName and the “Fit to Width” selection is changed to MyWidthFitName. This routine easily does the task versus changing the source code.

```
EZP_DocChangeNames(hFile, "Printer", "Closer", "FitWin", "FitWidth")
```

EZP_DocDateStamp

```
SUB EZP_DocDateStamp(hFile AS HANDLE, MyText AS WSTRING)
```

This routine allow for finer control of how the date is displayed in the Header or Footer. By default the date is displayed in the mm-dd-yyyy format.

```
EZP_DocDateStamp(hFile, "June 8th, 2017")
```

EZP_DocDrawImage

```
SUB EZP_DocDrawImage(hFile AS HANDLE, WhichBitmap AS LONG, MyAngle AS  
LONG, x AS SINGLE, y AS SINGLE, x2 AS SINGLE = 0, y2 AS SINGLE = 0,  
AspectFlag AS LONG = 0)
```

This routine tells the display engine to draw a bitmap loaded by EZP_DocAddImage at location x,y on the current page. If x2 and y2 are defined, then the image will be scaled to fit the rectangle. If the AspectFlag is non zero then the height of the image will be changed to keep the aspect ratio of the image intact. The MyAngle of the image is currently ignored since I was unable to rotate an image around the image center successfully. This may be fixed in a later version.

```
DIM bm AS LONG
```

```
EZP_DocDrawImage(hFile, bm, 0, 2, 7, 2, 2, 0)
```

EZP_DocEllipse

```
SUB EZP_DocEllipse(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, x2 AS SINGLE, y2 AS SINGLE, MyFillStyle AS SINGLE, MyFillColor AS INTEGER, MyWidth AS LONG, MyBorderColor AS INTEGER)
```

This routine tells the display engine to draw an ellipse at location x,y to x2,y2. The ellipse border will be drawn with a pen of MyWidth width and MyBorderColor color. The ellipse will be fill with a brush of MyFillColor color and of MyFillStyle style. If the MyFillStyle brush is of **EZP_BHOLLOW** then the fill color is ignored.

```
EZP_DocEllipse(hFile, 5, 3, 6, 4, EZP_BSOLID, EZP_BLUE, 1, EZP_BLACK)
```

EZP_DocEnd

```
SUB EZP_DocEnd(hFile AS HANDLE, MyStyle AS LONG = 0)
```

This routine tells the display engine to close the document. The optional MyStyle parameter is a set of options directing the DoDoc routine on what to do with the document when opened. If MyStyle is not used, then the document will be viewed in the Print Preview Dialog as a default. Other options include to Print the document to the default printer, and to Display the Print dialog to allow the user options such as set a print range. If desired the EZP_END_DELETE switch can be OR'ed in MyStyle to delete the file after being processed.

```
EZP_DocEnd(hFile, EZP_END_PRINT + EZP_END_DELETE)
```

MyStyle Switches

CONST EZP_END_VIEW	=	0	
CONST EZP_END_PRINT	=	1	'With Dialogs
CONST EZP_END_PRINT_NODIALOG	=	2	
CONST EZP_END_DELETE	=	4	'Delete File

EZP_DocFont

```
Function EZP_DocFont(hFile AS HANDLE, MyStyle AS LONG, MySize AS LONG, MyColor AS INTEGER, MyFontName AS WSTRING PTR) AS SINGLE
```

This routine tells the display engine which font to make the current font. Any following command that uses a font in displaying text will use this font. The return value will be the height of the font in inches or centimeters depending on which was chosen in the EZP_Doc_Begin directive.

Font styles can be OR'ed together using a plus sign from:

```
CONST E郑_NORMAL      = 0
CONST E郑_BOLD        = 1
CONST E郑_ITALIC      = 2
CONST E郑_UNDER       = 4
CONST E郑_STRIKE      = 8
```

Such as:

```
E郑_DocFont(hFile,E郑_ITALIC OR E郑_UNDER,12,E郑_RED,"Arial")
```

E郑_DocFooter

```
SUB E郑_DocFooter(hFile AS HANDLE,MyText AS WSTRING PTR,x AS SINGLE,y
AS SINGLE,x2 AS SINGLE,y2 AS SINGLE, MyStyle AS LONG = 0, MySize AS
SINGLE = 0)
```

This routine tells the display engine to insert a footer into the document. Each page will have this footer which comprises of centered text (which can be blank or a string of zero length). This centered text uses the current font. Optional date and page numbers can also be displayed in the upper-left or upper-right corners of the prescribed rectangle by defining a MyStyle which is OR'ed with the following constants:

```
CONST E郑_PAGENUMLEFT   = 1
CONST E郑_PAGENUMRIGHT = 2
CONST E郑_DATELEFT     = 4
CONST E郑_DATERIGHT    = 8
```

The font will be the current font but the size of the footer can be resized using the optional MySize parameter. Usually the page numbers and date will be a smaller size than the centered text (if any).

Then date will be the current date at the time of the document creation if mm-dd-yyyy format unless altered with the E郑_DocDateStamp command prior to the E郑_DocFooter command.

```
E郑_DocFooter(hFile," ",0.5,10,8,10.5,E郑_PAGENUMRIGHT,12)
```

E郑_DocGridSet

```
SUB E郑_DocGridSet(hFile AS HANDLE, Action AS LONG,x AS SINGLE,y AS
SINGLE,w AS SINGLE = 0, h AS SINGLE = 0)
```

This routine tells the display engine how to display the grid and user margins. A grid can be total redefined from the default, or just some specified rows or columns can have the respective heights and widths changed. The user margins are used for reference only in EZIP

```
CONST EZIP_GRIDNEW          = 0
CONST EZIP_GRIDMOD          = 1
CONST EZIP_GRIDMARGINS      = 2
```

Assume the document is set to inches

```
EZIP_DocGridSet(hFile, EZIP_GRIDNEW,1,1)
```

All cells are set to 1 inch height and 1 inch width

Cell Widths = x

Cell Heights = y

The size of the grid is set up in the define section of the include file

```
#define EZIP_MAXNUMCOLS 300
```

```
#define EZIP_MAXNUMROWS 1000
```

```
EZIP_DocGridSet(hFile, EZIP_GRIDMOD,5,0,3.75,0)
```

Row 5 is now 3.75 inches in height

Zeros are ignored

```
EZIP_DocGridSet(hFile, EZIP_GRIDMARGINS,.88,.88,.88,.88)
```

All user margins are set to 0.88 inches from the default values.

Zeros are ignored and default values will remain unchanged.

Left Margin = x

Right Margin = w

Top Margin = y

Bottom Margin = h

Default margins are set in the define section of the include file

```
#define EZIP_GRIDMARGINSDEFAULT 0.75
```

EZIP_DocHeader

```
SUB EZIP_DocHeader(hFile AS HANDLE, MyText AS WSTRING PTR,x AS
SINGLE,y AS SINGLE,x2 AS SINGLE,y2 AS SINGLE, MyStyle AS LONG = 0,
MySize AS SINGLE = 0)
```

This routine is a partner to the E郑_DocFooter command described early with the exception that dates and page numbers will be displayed in the LOWER corners of the prescribed rectangle. See E郑_DocFooter for parameter descriptions.

```
E郑_DocHeader(hFile, "This Header", 0.5, 1, 8, 2, E郑_PAGENUMLEFT +  
E郑_DATERIGHT, 8)
```

E郑_DocLine

```
SUB E郑_DocLine(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, x2 AS  
SINGLE, y2 AS SINGLE, MyWidth AS SINGLE, MyStyle AS LONG, MyColor AS  
LONG)
```

This routine tells the display engine to draw a line from x,y to x2,y2 with pen width MyWidth, pen style of MyStyle and of MyColor.

```
E郑_DocLine(hFile, 6, 1, 6, 7, 3, E郑_PSOLID, E郑_BLUE)
```

E郑_DocNewPage

```
SUB E郑_DocNewPage(hFile AS Handle, MyBookmark AS WSTRING = "")
```

This routine tells the display engine to create a new page. A bookmark is optional but if included, E郑 will create a bookmark which will allow for going to that page from the Print Preview Dialog bookmark dropdown control.

```
E郑_DocNewPage(hFile, "MyPage2")  
Start a new page and create a bookmark named "MyPage2"  
Bookmarks can be up to 49 characters long
```

The number of available bookmarks is set in the define section.

```
#define E郑_MAX_BOOKMARKS 50
```

E郑_DocPie

```
SUB E郑_DocPie(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, x2 AS  
SINGLE, y2 AS SINGLE, X1Arc AS SINGLE, Y1Arc AS SINGLE, x2Arc AS  
SINGLE, Y2Arc AS SINGLE, MyWidth AS SINGLE, MyStyle AS LONG, MyColor AS  
LONG, MyFill AS LONG)
```

The Pie function draws a pie-shaped wedge bounded by the intersection of an ellipse and two radials. The pie is outlined by a line of MyWidth of MyStyle and filled with MyColor. This function uses the Windows SDK Pie function.

```
EZP_DocPie(hFile,4,6,5,7,4.4,6.2,4.9,6.9,2,0,EZP_BLUE,EZP_BLUE)
```

EZP_DocPolygon

```
SUB EZP_DocPolygon(hFile AS HANDLE,NumPoints AS LONG,MyPoints() AS PolyPoints,lineColor AS INTEGER, fillColor AS INTEGER)
```

This routine tells the display engine to draw a polygon using the array MyPoints of type PolyPoints and of NumPoints indices. It will have an outline pen color of lineColor and a brush fill color of fillColor.

DIM pp(2) as PolyPoints

```
'Make a Polygon
pp(0).x = 5.3 : pp(0).y = 4
pp(1).x = 6.3 : pp(1).y = 5
pp(2).x = 4.3 : pp(2).y = 5
EZP_DocPolygon(hFile,3,pp(),EZP_YELLOW,EZP_YELLOW)
```

EZP_DocRect

```
SUB EZP_DocRect(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, x2 AS SINGLE, y2 AS SINGLE, MyFillStyle AS SINGLE, MyFill AS INTEGER, MyWidth AS LONG, MyBorderColor AS INTEGER ,RoundRectPix AS INTEGER = 0)
```

This routine tells the display engine to draw a rectangle at x,y to x2,y2. It will have a pen width of MyWidth, and a pen color of MyBorderColor. It will have a brush fill color of MyFill and a brush fill style of MyFillStyle. If the fill style is EZP_BHOLLOW, then the fill color will be ignored. Optionally, a rectangle can be rounded by adding the RoundRectPix parameter which is

In this example of a rounded rectangle, the fill color of EZP_RED is ignored since this is a hollow brush filled rectangle.

```
EZP_DocRect(hFile,5,5,6,6,EZP_BHOLLOW,EZP_RED,1,EZP_BLACK,30)
```

EZP_DocShape

```
SUB EZP_DocShape(hFile AS HANDLE, WhichShape AS LONG,x AS SINGLE,y AS SINGLE,x2 AS SINGLE, y2 AS SINGLE, fillColor AS INTEGER=0, MySize AS LONG = 2)
```

This routine tells the display engine to draw a shape of type WhichShape at x,y to x2,y2 of color fillColor (default is black) and of MySize (default = 2). MySize controls the thickness of the shaft and size of the cap. If WhichShape is EYP_ARROWAUTO then MySize is ignored and the shaft and cap are proportional to the arrow length. Currently only 1 general shape is current supported which is an arrow.

```
CONST EYP_ARROWEND    = 1
CONST EYP_ARROWSTART  = 2
CONST EYP_ARROWBOTH   = 3
CONST EYP_ARROWAUTO   = 4
```

```
EYP_DocShape(hFile, EYP_ARROWBOTH,1,5,4,6,EYP_RED,2)
```

EYP_DocSymbol

```
SUB EYP_DocSymbol(hFile AS HANDLE, WhichSymbol AS LONG, MySize AS
LONG,x AS SINGLE,y AS SINGLE,lineColor AS INTEGER, fillColor AS
INTEGER = -123)
```

This routine tells the display engine to draw a shape which is centered on x,y. The main purpose of a symbol would be for use in building a plot. The symbol would be of pen color lineColor and be filled with a brush of fillColor. It would be of point size MySize. The default fillColor is a hollow brush. The types of symbols currently supported are:

```
CONST EYP_SYMBOL_RECT        = 1
CONST EYP_SYMBOL_DIAMOND     = 2
CONST EYP_SYMBOL_TRIUP       = 3  'Triangle pointing up
CONST EYP_SYMBOL_TRIDOWN     = 4  'Triangle pointing down
CONST EYP_SYMBOL_TRILEFT     = 5  'Triangle pointing left
CONST EYP_SYMBOL_TRIRIGHT    = 6  'Triangle pointing right
CONST EYP_SYMBOL_CIRCLE      = 7
CONST EYP_SYMBOL_STAR        = 8
```

```
EYP_DocSymbol(hFile,EYP_SYMBOL_STAR,30,2,6,EYP_RED,EYP_RED)
```

EYP_DocTable

```
SUB EYP_DocTable(hFile as HANDLE,MyTable() AS TableCell,x AS SINGLE, y AS SINGLE, CellCharWidth
AS LONG = 8, SuppressGridLines AS LONG = 0)
```

This routine tells the display engine to draw a table defined by the MyTable array at x,y. The table will use the current font and calculate the average character width and height. The cells will have a width of CellCharWidth (defaults to 8 character wide) . A column width can be increased by entering the

number of characters in any cell in that column for the delta character width. The cells will have a height to accommodate 1 line of text. The height of a row of cells can be increased by entering 1 (delta height) in any cell on that row to bring the height to 2 characters high and so forth. Type will wrap if the cell height is large enough.

```
TYPE TableCell
    DIM h AS LONG 'Height Adjustment in Pixels stored in Column 0
    DIM w AS LONG 'Width Adjustment in Pixels stored in Row 0
    DIM MyText AS WSTRING * 100
    DIM FontStyle AS LONG 'Adjusted from the current font
    DIM MyCellColor AS INTEGER
    DIM MyJustify AS LONG
END TYPE
DIM tt(1,1) as TableCell
```

Justification Constants

```
CONST EBP_LEFT      = 0
CONST EBP_CENTER    = 1
CONST EBP_RIGHT     = 2
```

Font Styles

```
CONST EBP_NORMAL    = 0
CONST EBP_BOLD      = 1
CONST EBP_ITALIC     = 2
CONST EBP_UNDER      = 4
CONST EBP_STRIKE     = 8
```

'Make a table

```
EBP_DocFont(hFile,EBP_NORMAL,14,EBP_BLACK,0,"Arial")
tt(0,0).MyText = "The rain in spain"
tt(0,1).MyText = "Hello01"
tt(1,0).MyText = "Hello10"
tt(1,1).MyText = "Hello11"

tt(0,0).MyJustify = EBP_CENTER
tt(0,0).FontStyle = EBP_BOLD
tt(1,1).MyCellColor = EBP_RED
tt(0,0).w = 5
tt(0,0).h = 1
EBP_DocTable(hFile,tt(),1,4)
```

EZP_DocText

```
SUB EZP_DocText(hFile AS HANDLE, x AS SINGLE, y AS SINGLE, MyText AS
WSTRING PTR, MyAngle AS LONG = 0, x2 AS SINGLE = 0, y2 AS SINGLE = 0,
MyJust AS LONG = 0, MyBorderColor AS INTEGER = 0, MyFillColor AS
INTEGER = 0)
```

This routine tells the display engine to draw the text at location x,y using the current font. In its simplest form, the command would be:

```
EZP_DocText(hFile,4,7,"Hello World") at 4 inches x and 7 inches y
```

Angled text at 90 degrees

```
EZP_DocText(hFile,4,7,"Hello World",90)
```

Text can be in an enclosed rectangle which x,y,x2,y2. The height of the rectangle will be adusted to fit the text. The text of the can be justified with MyJust. The angle of the text in rectangled text must be zero. The color of the pen used to outline is determined by MyBorderColor. If you want to frame the rectangle then enter in a negative MyJust. That is the switch. The color of the outline rectangle is MyBorderColor which defaults to black and the color of the fill is MyFillColor which defaults to white.

Justification Constants

```
CONST EZP_LEFT           = 0
CONST EZP_CENTER         = 1
CONST EZP_RIGHT          = 2
```

The same below enclosed centered text in a blue outline and filled by red.

```
MyBuffer = "The rain in spain stays mainly on the plains. The rain
in spain stays mainly on the plains."
EZP_DocText(hFile,2.33,3,VARPTR(MyBuffer),0,4,5,-EZP_CENTER,
EZP_BLUE,EZP_RED)
```

EZP_DocTextEx

```
FUNCTION EZP_DocTextEx(hFile AS HANDLE, TxtLen AS SINGLE, x AS
SINGLE, y AS SINGLE) AS LONG
```

This routine tells the display engine to draw the text with the programmer having more control how it flows. Once text is loaded into the buffer using EZP_DocTextExRead it is read in one line at a time of

TxtLen inches or centimeters, and placed at x,y. Then remaining number of characters is returned by the function. It may be useful to place text in relation to the grid with the height of the grid cells being the desired space between lines.

See EZPrintDemoLorem.bas for an example of how to use this command.

EZP_DocTextExRead

```
FUNCTION EZP_DocTextExRead(hFile AS HANDLE, MyText AS WSTRING PTR) AS LONG
```

This routine loads a longer text string into a buffer and returns the length of the text in characters. This text can be then read by EZP_DocTextEx in chucks to control how it is printed. The text can be flowed around pictures for example. As stated before, the length of the text is returned by this function.

See EZPrintDemoLorem.bas for an example of how to use this command.

EZP_DoDoc

```
FUNCTION EZP_DoDoc(byval HWND AS HWND, MyOptions AS LONG, TheDoc AS WSTRING, ViewerOptions AS LONG = 0) AS LONG
```

This routine processes a EZP document. HWND is the parent of the Print Preview Dialog child window. If the dialog is a print only then the Print Preview Dialog will not open. MyOptions is not used at this time but is a placeholder for future use. TheDoc is the document file to be processed. It is recommended that all EZP files have a .ezp file type suffix. If it is of zero length, then an open file dialog is presented to select a file with an .ezp suffix. The optional ViewerOptions toggles on different grid outlines and can set the Previewer to Zoom to Fit. Also, the EZP_IGNOREFILEEND switch can be used to force the opening in the Previewer of a document even though it is a print only file.

```
EZP_DoDoc(HWND, 0, "MyDocTest.ezp", EZP_VIEWER_GRID + EZP_VIEWER_ZOOMFIT)
```

ViewerOptions

CONST EZP_VIEWER_GRID	=	1
CONST EZP_VIEWER_MARGINS	=	2
CONST EZP_VIEWER_HEADERFOOTER	=	4
CONST EZP_VIEWER_ZOOMFIT	=	8
CONST EZP_IGNOREFILEEND	=	16

A version number has been predefined and stored with the document. If the version number of a document does not match the current engine, then DoDoc will not run it.

```
#define EZP_VERSION 1
```

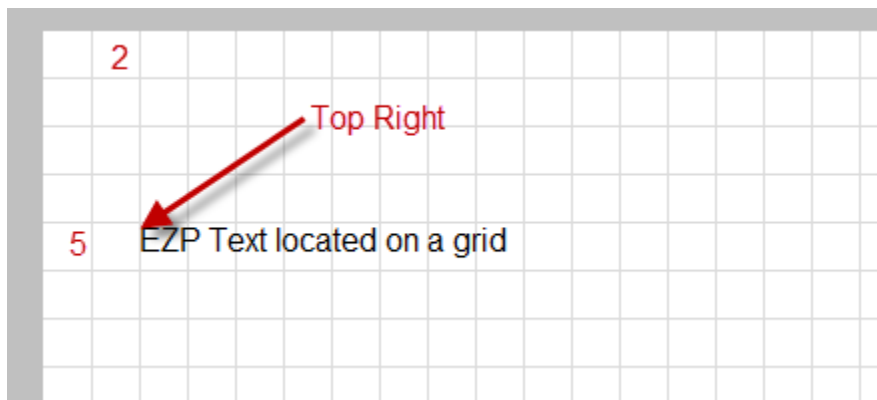

EZP_G

```
FUNCTION EZP_G(row AS LONG, col AS LONG, pt AS LONG = 0) AS SINGLE
```

This routine is helpful in telling the display engine where to draw the page which is gridded. One of 4 corners of a cell can be used to locate an x or y value. EZG_G returns either inches or centimeters depending on how the document is set up. A point can be “tweaked” by adding a little to it if necessary such as: `EZP_G(5,2,EZP_1) + 0.125`

```
CONST EZP_t      = 0
CONST EZP_l      = 1
CONST EZP_b      = 2
CONST EZP_r      = 3
```

```
EZP_DocText(hFile,EZP_G(5,2,EZP_l),EZP_G(5,2,EZP_t),"EZP Text located  
on a grid")
```



Some macros have been predefined to cut down on the amount of typing

```
EZP_DocText(hFile,ezl(5,2),ezt(5,2),"EZP Text located on a grid")
```

```
#define ezt(r,c) EZP_G(r,c,0) top of the cell
```

```
#define ezl(r,c) EZP_G(r,c,1) left of the cell
```

```
#define ezb(r,c) EZP_G(r,c,2) bottom of the cell
```

```
#define ezc(r,c) EZP_G(r,c,3) right of the cell
```

PRINT ENGINE CONSTANTS

~~~~~  
Unit of measure constants AS passed to EZP\_DocBegin  
~~~~~

CONST EZP_INCH = 0
CONST EZP_CM = 1

~~~~~  
EZP\_DoDocEnd Option constants  
~~~~~

CONST EZP_END_VIEW = 0
CONST EZP_END_PRINT = 1 'With Dialogs
CONST EZP_END_PRINT_NODIALOG = 2
CONST EZP_END_DELETE = 4 'Delete File

~~~~~  
EZP\_DoDoc View Option constants  
~~~~~

CONST EZP_VIEWER_GRID = 1
CONST EZP_VIEWER_MARGINS = 2
CONST EZP_VIEWER_HEADERFOOTER = 4 '
CONST EZP_VIEWER_ZOOMFIT = 8
const EZP_IGNOREFILEEND = 16

CONST EZP_ZOOM100 = 0
CONST EZP_ZOOMFIT = 1
CONST EZP_ZOOMWIDTH = 2
CONST EZP_ZOOM125 = 3
CONST EZP_ZOOM75 = 4

~~~~~  
Orientation Constants for use with EZP\_DocBegin  
~~~~~

CONST EZP_PORTRAIT = 0 'DMORIENT_PORTRAIT = 1
CONST EZP_LANDSCAPE = 1 'DMORIENT_LANDSCAPE = 2

CONST EZP_DUPLEX_NONE = 0 ' Normal nonduplex Printer default is actually
DMDUP_SIMPLEX = 1

```

CONST EZP_DUPLEX_VERTICAL    = 2 ' LONG Edge binding - the LONG edge of the page is
vertical
CONST EZP_DUPLEX_HORIZONTAL = 3 ' Short Edge binding - the LONG edge of the page is
horizontal

```

~~~~~

Paper Bin Constants used in EZP\_DocBegin and EZP\_DocNewPage

~~~~~

```

CONST EZP_BIN_UPPER = 1
CONST EZP_BIN_FIRST = EZP_BIN_UPPER
CONST EZP_BIN_ONLYONE = 1
CONST EZP_BIN_LOWER = 2
CONST EZP_BIN_MIDDLE = 3
CONST EZP_BIN_MANUAL = 4
CONST EZP_BIN_ENVELOPE = 5
CONST EZP_BIN_ENVMANUAL = 6
CONST EZP_BIN_AUTO = 7
CONST EZP_BIN_TRACTOR = 8
CONST EZP_BIN_SMALLFMT = 9
CONST EZP_BIN_LARGEfmt = 10
CONST EZP_BIN_LARGEcapacity = 11
CONST EZP_BIN_CASSETTE = 14
CONST EZP_BIN_FORMSOURCE = 15

```

~~~~~

Paper Size Constants used in EZP\_DocBegin and EZP\_DocNewPage

~~~~~

```

CONST EZP_PAPER_LETTER = 1           ' Letter 8 1/2 x 11 in
CONST EZP_PAPER_LETTERSMALL = 2       ' Letter Small 8 1/2 x 11 in
CONST EZP_PAPER_TABLOID = 3           ' Tabloid 11 x 17 in
CONST EZP_PAPER_LEDGER = 4            ' Ledger 17 x 11 in
CONST EZP_PAPER_LEGAL = 5             ' Legal 8 1/2 x 14 in
CONST EZP_PAPER_STATEMENT = 6         ' Statement 5 1/2 x 8 1/2 in
CONST EZP_PAPER_EXECUTIVE = 7         ' Executive"7 1/2 x 10 in
CONST EZP_PAPER_A3 = 8                ' A3 297 x 420 mm
CONST EZP_PAPER_A4 = 9                ' A4 210 x 297 mm
CONST EZP_PAPER_A4SMALL = 10          ' A4 Small 210 x 297 mm
CONST EZP_PAPER_A5 = 11               ' A5 148 x 210 mm
CONST EZP_PAPER_B4 = 12               ' B4 250 x 354
CONST EZP_PAPER_B5 = 13               ' B5 182 x 257 mm
CONST EZP_PAPER_FOLIO = 14            ' Folio 8 1/2 x 13 in
CONST EZP_PAPER_QUARTO = 15           ' Quarto 215 x 275 mm
CONST EZP_PAPER_10x14 = 16            ' 10x14 in
CONST EZP_PAPER_11x17 = 17            ' 11x17 in
CONST EZP_PAPER_NOTE = 18             ' Note 8 1/2 x 11 in
CONST EZP_ENV_9 = 19                  ' Envelope #9 3 7/8 x 8 7/8
CONST EZP_ENV_10 = 20                 ' Envelope #10 4 1/8 x 9 1/2
CONST EZP_ENV_11 = 21                 ' Envelope #11 4 1/2 x 10 3/8

```

CONST EZP_ENV_12 = 22	' Envelope #12 4 \276 x 11
CONST EZP_ENV_14 = 23	' Envelope #14 5 x 11 1/2
CONST EZP_ENV_DL = 27	' Envelope DL 110 x 220mm
CONST EZP_ENV_C5 = 28	' Envelope C5 162 x 229 mm
CONST EZP_ENV_C3 = 29	' Envelope C3 324 x 458 mm
CONST EZP_ENV_C4 = 30	' Envelope C4 229 x 324 mm
CONST EZP_ENV_C6 = 31	' Envelope C6 114 x 162 mm
CONST EZP_ENV_C65 = 32	' Envelope C65 114 x 229 mm
CONST EZP_ENV_B4 = 33	' Envelope B4 250 x 353 mm
CONST EZP_ENV_B5 = 34	' Envelope B5 176 x 250 mm
CONST EZP_ENV_B6 = 35	' Envelope B6 176 x 125 mm

~~~~~  
Text alignment constants for EZP\_DocText in the style for multi-line  
text  
~~~~~

CONST EZP_LEFT	= 0
CONST EZP_CENTER	= 1
CONST EZP_RIGHT	= 2

~~~~~  
Grid Positioning  
~~~~~

CONST EZP_GRIDNEW	= 0
CONST EZP_GRIDMOD	= 1
CONST EZP_GRIDMARGINS	= 2

~~~~~  
Grid Positioning  
~~~~~

CONST EZP_t	= 0
CONST EZP_l	= 1
CONST EZP_b	= 2
CONST EZP_r	= 3

~~~~~  
Font Styles Constants  
~~~~~

CONST EZP_NORMAL	= 0
CONST EZP_BOLD	= 1
CONST EZP_ITALIC	= 2
CONST EZP_UNDER	= 4
CONST EZP_STRIKE	= 8

~~~~~  
Polygon Pre-Define Shapes  
~~~~~

```
CONST EZP_SHAPE_ARROW    = 1
`CONST EZP_SHAPE_STAR    = 2
`CONST EZP_SHAPE_BALLOON = 3
`CONST EZP_SHAPE_OTHER   = 4
```

~~~~~  
Symbol Shapes that are centered on a point  
~~~~~

```
CONST EZP_SYMBOL_RECT      = 1
CONST EZP_SYMBOL_DIAMOND   = 2
CONST EZP_SYMBOL_TRIUP     = 3  'Triangle pointing up
CONST EZP_SYMBOL_TRIDOWN   = 4  'Triangle pointing down
CONST EZP_SYMBOL_TRILEFT   = 5  'Triangle pointing left
CONST EZP_SYMBOL_TRIRIGHT  = 6  'Triangle pointing right
CONST EZP_SYMBOL_CIRCLE    = 7
CONST EZP_SYMBOL_STAR      = 8
```

~~~~~  
Pre-Defined Color Constants  
~~~~~

```
CONST EZP_WHITE      = BGR(255,255,255) ' Default White
CONST EZP_BLACK       = BGR(0,0,0) 'Black
CONST EZP_LTRED       = BGR(250,128,114) 'Light Red Salmon
CONST EZP_GREEN       = BGR(0,128,0) 'Green
CONST EZP_LTGREEN     = BGR(0,255,127) 'Light Green
CONST EZP_BLUE        = BGR(0,0,128) 'Blue
CONST EZP_LTBLUE      = BGR(175,238,238) 'Light Blue
CONST EZP_MAGENTA     = BGR(255,0,255) 'Magenta
CONST EZP_BEIGE       = BGR(139,0,139) 'Dark white
CONST EZP_CYAN        = BGR(0,255,255) 'Cyan
CONST EZP_AQUA        = BGR(127,255,212) 'Light Cyan Aquamarine
CONST EZP_RED         = BGR(255,0,0) 'Red
CONST EZP_BROWN      = BGR(244,164,96) 'Brown
CONST EZP_YELLOW      = BGR(255,255,0) 'Yellow
CONST EZP_ORANGE      = BGR(255,127,80) 'Orange
CONST EZP_GAINSBORO   = &HDCDCDC
CONST EZP_LIGHTGRAY   = &HD3D3D3
CONST EZP_SILVER      = &HC0C0C0
CONST EZP_DARKGRAY    = &HA9A9A9
CONST EZP_GRAY        = &H808080
CONST EZP_DIMGRAY     = &H696969
CONST EZP_LIGHTSLATEGRAY = &H998877
CONST EZP_SLATEGRAY   = &H908070
CONST EZP_DARKSLATEGRAY = &H4F4F2F
```

```

~~~~~
Pen Styles Constants
~~~~~
CONST EZP_PSOLID      = PS_SOLID
CONST EZP_PDASH       = PS_DASH
CONST EZP_PDOT        = PS_DOT
CONST EZP_PDASHDOT    = PS_DASHDOT
CONST EZP_PDASHDOTDOT = PS_DASHDOTDOT

~~~~~
Brush Styles Constants
~~~~~
CONST EZP_BHATCHED    = BS_HATCHED
CONST EZP_BHOLLOW     = BS_HOLLOW
CONST EZP_BSOLID      = BS_SOLID

~~~~~
Arrow Styles Constants
~~~~~
CONST EZP_ARROWNONE   = 0
CONST EZP_ARROWEND    = 1
CONST EZP_ARROWSTART  = 2
CONST EZP_ARROWBOTH   = 3
CONST EZP_ARROWAUTO   = 4

~~~~~
Control Words Contants
~~~~~
CONST EZP_CLOSE       = 0
CONST EZP_PRINT       = 1
CONST EZP_FITWIN      = 2
CONST EZP_FITWIDTH    = 3

~~~~~
Header and Footer Contants
~~~~~

CONST EZP_PAGENUMLEFT = 1
CONST EZP_PAGENUMRIGHT = 2
CONST EZP_DATELEFT    = 4
CONST EZP_DATERIGHT   = 8

```

PRINT ENGINE DEFINES

This routine tells the display engine to draw the

```
#define RGBA_R( c ) ( CUInt( c ) Shr 16 And 255 )
#define RGBA_G( c ) ( CUInt( c ) Shr  8 And 255 )
#define RGBA_B( c ) ( CUInt( c )           And 255 )
#define RGBA_A( c ) ( CUInt( c ) Shr 24           )
```

```
#define ezt(r,c) EZP_G(r,c,0)
#define ezl(r,c) EZP_G(r,c,1)
#define ezb(r,c) EZP_G(r,c,2)
#define ezr(r,c) EZP_G(r,c,3)
```

```
#define GetFilePointer(hFile) SetFilePointer(hFile, 0, NULL,
FILE_CURRENT)
#define EZP_VERSION 1
#define EZP_MAX_PAGES 1000
#define EZP_MAX_BOOKMARKS 50
#define EZP_MAX_GRAPHICS 50
#define EZP_DEFAULTPAPERSIZE 1 'Letter
#define EZP_MAX_PAPERS 35
#define EZP_MAXTEXTLEN 1024
#define EZP_DIAGVIEWTOP 50
#define EZP_FONT_NAME "Arial" 'Default Font Name
#define EZP_FONT_SIZE 10 'Default Font Size4
#define EZP_MAXNUMCOLS 300
#define EZP_MAXNUMROWS 1000
#define EZP_GRIDCOLDEFAULT 0.25 'or 0.63 cm
#define EZP_GRIDROWDEFAULT 0.25 'or 0.63 cm
#define EZP_GRIDMARGINSDEFAULT 0.75 'or 1.91 cm
```

