Compiler

COLLABORATORS			
	TITLE : Compiler		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Compiler

1.1 Commander Compiler

```
                              Commander Compiler

   Configuration

   Tutorials

   Language Overview

   Format
   Conventions
   Root Command Order

   Command Index


  History
   New in this version
   Previous versions
```

1.2 Configuration

```
Configuration
-------------

Configuration is supported through use of Workbench ToolTypes or CLI
parameters. If any options are not included, the stated defaults will be
used.

Configuration options are not case sensitive.

Invalid options will be ignored. In this case, the default values will
be used.


The following configuration commands are supported:
```

```
Definition
  Description: Compiles the given definition upon startup. No file
               requester is presented unless the module can not be found.

               The definition name is referenced as the path and filename
               within the 'definitions' directory.

  Default: NONE : FileRequester

  Example: Definition=Test.def

ErrorLog
  Description: Outputs error messages to the given file rather using
               requesters. This option is recommed for batch file
               processing.

  Default: Requesters

  Example: ErrorLog=RAM:Compiler.errors


Font
  Description: Sets an 8 point font for the window text. This should
  be a non-proportioanl font.

  Default: Topaz2.font

  Example: Font=Courier.font


Last
  Description: Compiles the last compiled definition. No file
               requester is presented unless the last used definition
               file can not be found.

  Valid values: Last

  Default: NO

  Example: Last


WindowL
  Description: Sets the position of the left edge of the window.

  Valid values: >= 0

  Default: 0

  Example: WindowL=50


WindowT
  Description: Sets the position of the top edge of the window.

  Valid values: >= 0
```

```
    Default: 0

    Example: WindowT=100
```

1.3 Tutorials

```
Tutorials
----------

The Commander definition language may look a bit daunting at first
glance, so here's some simple tutorials to get you started. Most users
will only want to modify the included definitions to suit there needs, so
that's the approach we'll take here for these mini-tutorials. It's not a
difficult task as you'll soon see.

You may wish to back up the included definition files, before proceeding
with these tutorials.

It is recommended that you do each tutorial in order, as assumptions may
be made that you have done so.

     Modifying the included definitions

     1.  Adding ARexx scripts
     2.  Removing panels

     Creating new definitions

     3.  Building your first module
```

1.4 Tutorial 1

```
Tutorials
---------

1. Adding ARexx scripts

Let's say you just want to add an ARexx script to the ARexx page of the
"AllInOne_OnePanel.cmndr" module. Load "AllInOne_OnePanel.def" into your
text editor and we'll begin.

There are two parts to this process. Gadgets are defined and then linked
to a special function, in this case an ARexx file. Locate the following
section of the definition. If you can, search/find ->   Window "ARexx"

    Window "ARexx" (
      Gadget (
        Position 0 2
        Type List (
          Size 145 313
          Data (
```

```
            "Border.ifx"
            "Designs.ifx"
            "MapToWB.ifx"
          )
          Link "ARexx"
        )
      )
    )
```

This describes the ARexx page of the module, and includes the List gadget with all the names of the ARexx scripts. We want to add the script "Rexx/AnalyzeJPEG.ifx" to our module. The gadget is already tall enough to accomodate more entries without having to scroll, so we don't need to alter the Size info. Insert the name "AnalyzeJPEG.ifx" at the top of the Data list, so that it's alphabetical. Note that you could call this "My Favourite Script" if you wanted, Commander will still be able to find the file, as the file name is defined elsewhere. You're just defining the text that you see in the gadget list here. It should look like this now:

```
    Data (
      "AnalyzeJPEG.ifx"
      "Border.ifx"
      "Designs.ifx"
      "MapToWB.ifx"
    )
```

Okay, we're half done already. Note the Link "ARexx" line used here. That tells Commander what this gadget is linked to. Now locate the following section:

```
    Define Link "ARexx" (
      Type ARexx (
        Data (
          File "Rexx/Border.ifx"
          File "Rexx/Designs.ifx"
          File "Rexx/MapToWB.ifx"
        )
      )
    )
```

This is the link definition for the ARexx page, and currently includes three external (File) ARexx scripts. Since we are adding an external script we want to use the File option, so our definition line is as follows:

```
    File "Rexx/AnalyzeJPEG.ifx"
```

As we did with the gadget section, we want it to appear at the top of the list. It should look like this now:

```
    Data (
      File "Rexx/AnalyzeJPEG.ifx"
      File "Rexx/Border.ifx"
      File "Rexx/Designs.ifx"
      File "Rexx/MapToWB.ifx"
    )
```

Note that it's important to get the exact file name, and the path is important too. Since this script is in ImageFX's "Rexx" directory that's all we need for the path. If you wanted to link to a script on another hard drive, for example, you would include the full path. Perhaps something like "AnotherHarddrive:ImageFXScripts/MyScript.rexx".

Okay, were done. Now, save the definition and let's go to the compiler. Double click on the Compiler icon and you should get a file requester. Select the file "AllInOne_OnePanel.def" and, if all went well, you shouldn't get any errors. If you goofed something up, you'll be told soon enough.

That's it. Now go Open the module in Commander and try it out.

1.5 Tutorial 2

```
Tutorials
---------

2. Removing panels
```

Okay, maybe you like the "AllInOne_MovePanels.cmndr" but it has too many panels for your needs (or screen space). Let's butcher it a bit.

Maybe you don't want to have the "Analysis" panel because you don't use it much and it just gets in the way. Load the "AllInOne_MovePanels.def" file into your text editor and let's get going.

There's three things we need to do: Delete the window, delete the associated links, and delete the associated gadgets. Locate the following section:

```
    Define Window "Alpha" (
      Position 0 27
      Size 153 136
      Title
    )
```

This defines the actual window. To get rid of it, just delete all those lines. Now locate this section:

```
     ' ********************************************** Alpha **

      Window "Alpha" (
        Gadget (
           Position 4 14
           Type Button (
            Label "Copy From Alpha"
            Size 145 15
            Link "Alpha_CopyFromAlpha"
          )
        )

        ... + 6 more Gadgets
```

```
        Gadget (
          Position 4 119
          Type Button (
            Label "Swap"
            Size 145 15
            Link "Alpha_Swap"
          )
        )
      )
```

This section defines all the gadgets in the "Alpha" window (which we just deleted). There are 8 gadgets in total (only two shown here). You want to delete everything you see here, right down to the end of the window structure. Make sure you delete the final closing structure bracket, too. Each section of this module is divided with the '******* comment lines, so it should be very obvious where the end is.

You might have wanted to note each of the Links used before deleting each gadget, because we want to delete them too. But, I've made your life simple by naming them all with the page as part of the title. So, it'll be obvious. Now locate this section:

```
    ' ************************************************ Alpha **

    Define Link "Alpha_CopyFromAlpha" (
      Type ARexx (
        Data ( String "Alpha2Buffer" )
      )
    )

    ... + 6 more Link definitions

    Define Link "Alpha_Swap" (
      Type ARexx (
        Data ( String "SwapAlpha" )
      )
    )
```

Delete those 8 Link definitions and you're all set.

Now save the definition, and recompile it. There, no more Alpha panel.

1.6 Tutorial 3

```
Tutorials
---------

3. Building your first module

In the following tutorial you do not need to actually create a
definition file, we have already included them to save you the time.
Use them for reference as you proceed through the tutorial, if
necessary.  They are a good guide for proper stucture and ordering.
```

First things first

The first step is to tell the compiler what fonts we want to use in this module. We just need one right now. The compiler also needs to know how we want to refer to this font later, we will just use the label "Font1". We want to use the Topaz2 font with a size of 8. Here is how it would look:

```
Define Font "Font1" (
  Name "Topaz2.font"
  Size 8
)
```

Notice the brackets that enclose the 'Define Font' structure. Several commands have parameters with the same name, such as 'Position'. These brackets ensure that the compiler can tell where one command ends and the next one begins. Brackets must always appear in pairs '(' for starting (opening) and ')' for ending (closing) a structure. We have also indented the two parameters, so that it is easier to see that they are part of the 'Define Font' structure.

'Define Font' has two requirements, a label "Font1" and the 'Name' parameter. The 'Size' parameter is actually optional. The compiler will default to a size of 8 if it is not told specifically to use a different size. We have included it here to make things more understandable.

The next thing to do is to define a window for our gadgets and text to appear in. Windows are defined like this:

```
Define Window "Main" (
  Position 10 20
  Size 150 200
  Title
)
```

We have given this one the label "Main", so that we can refer to it later. The two required parameters, 'Position' and 'Size' tell the compiler where we want the window to appear on the screen. In this case, the upper left hand corner of the window should be at the coordinates: x=10, y=20 and it should be 150 pixels wide and 200 pixels tall. The optional parameter 'Title' indicates that we want the window to have a title in it. The compiler is smart enough to also enable the 'DragBar' option when 'Title' is used. This allows the window to be moved around at will.

Finally, we need to reference the defined window or the compiler will complain that it hasn't been used. For now, we will just use an empty window structure which we will explain later.

```
Window "Main" (
)
```

Compiling a module

At this point, we have enough information to compile a module. We have included the file "Tutorial_3-A.def" which includes the commands we have used so far.

Start the compiler by double clicking the Compiler icon. When you are prompted with the file requester, select the file "Tutorial_3-A.def".

Once you have selected the definition file to be compiled the compiler begins its job. This is not a very big definition yet, so you likely will not get to see any of the information the compiler displays as it compiles. Basically, all you missed is the compiler counting lines, fonts, windows, etc. If there had been any errors, the compiler would have told you and stopped. Note that you do not have to quit the compiler, it quits when it is finished.

Viewing the module

In that couple of seconds the compiler generated our tutorial module. Let's have a look and see what we have so far. Start Commander by clicking on its icon. When you are prompted with the file requester, select the file "Tutorial_3-A.cmndr". Commander will load the module and display the defined window. Our module does not do anything yet, so just enjoy dragging the little window around for a moment.

Displaying text

We have not used the font we defined, so let's dislay some text with it. First, we need to the tell the compiler what window we want to display the text in. All of the commands that apply to windows, 'Text', 'Gadget', 'Border', etc. are contained within a window structure. Simply, a window structure looks like this:

```
Window "Main" (
)
```

That will not actually do anything on its own. It simply tells the compiler which window we are going to work with. We need to put something inside that structure for it to be of any real use.

```
Window "Main" (
  Text "ABCDEFGHIJK" (
    Position 10 15
  )
)
```

The text command has two required parameters: a text string and a 'Position'. Here we have told the compiler to place the text "ABCDEFGHIJK" at x=10, y=15. 'x' refers to the left edge of the text and 'y' refers to the baseline of the font. The baseline is the point at which a letter meets the ground, so to speak. Letters such as "abcdefhiklmnorstuvwxz" all sit on the baseline. But "gjpqy" have descenders, which drop below the baseline. With a standard size 8 font you have something like this:

```
      01234567     01234567

    0 **         0
    1 **         1
    2 ******     2 **    **
    3 **   **    3  **  **
    4 **   **    4   ****
    5 **   **    5    **
    6 o*****     6 o  **
    7            7  ***
```

Where the baseline is at row 6. When defining the position text, you have to keep the baseline in mind when indicating coordinates. If you wanted the upper left hand corner of the letter 'b' to appear at x=5, y=10 you would use 'Position 5 16'. In the above examples, 'o' indicates the point at which coordinates refer (at 0,6).

We will display two different texts in our previously defined window:

```
  Window "Main" (
    Text "Plain Text" (
      Position 20 25
    )
    Text "Shadow Text" (
      Position 20 40
      Color 2
      Shadow
    )
  )
```

Refer to the Command Index section for more information about the Text command and the extra parameters, 'Color' and 'Shadow', used here and others that can also be used.

Compile "Tutorial_3-B.def" and open the 'Tutorial_3-B.cmndr' module in Commander to see the results.

Adding borders

Let's make things a little prettier and add a border around the text. Borders are defined like this:

```
  Border (
    Position 10 10
    Size 107 40
    Fill 3
  )
```

Remember that, although we have not shown it this way, this command and others like it must be contained within a 'Window' structure.

The 'Position' parameter, as with 'Define Window', refers to the

coordinates of the upper left hand corner of the border. Again, 'Size' refers to the width and height in pixels. Note that there is an alternative to using the 'Size' command anywhere it is used. 'Position2' sets the position of the lower right hand corner. In the above example, we could have used 'Position2 117 50' instead of 'Size 107 40' and would have gotten the same result. When using a paint program to do intial layout designs, the 'Position2' alternative can save you some calculation since most paint programs only give x/y coordinates.

The 'Fill' option tells the compiler that we want the border filled with color 3. It is important to realize that sometimes the order that commands appear within the definition file needs to be considered. If we added this 'Border' definition in AFTER the 'Text' commands, the text would be overdrawn by the fill option we have used. Because we want the text to appear over the filled border, we have to order it BEFORE. Something like this:

```
  Window "Main" (
    Border (
      Position 10 10
      Size 107 40
      Fill 3
    )
    Text "Plain Text" (
      Position 20 25
    )
    Text "Shadow Text" (
      Position 20 40
      Color 2
      Shadow
    )
  )
```

Compile "Tutorial_3-C.def" and open the 'Tutorial_3-C.cmndr' module in Commander to see the results.

Adding gadgets

To interact with Commander we have to add gadgets. Although Commander supports several different kinds, we will begin with a simple button gadget example:

```
  Gadget (
    Position 10 75
    Type Button (
      Label "Border.ifx"
      Size 107 12
      Link "ARexx"
    )
  )
```

The gadget command has two required parameters: 'Position' and 'Type'. As with other commands, 'Position' defines the upper left hand coordinate of the gadget. 'Type' is used to define the kind of gadget and its parameters. Note that 'Type' is also a structure imbedded (or nested)

within the Gadget command with its own structure bracketing pair.

The 'Type Button' structure has three required parameters. 'Label' sets the text that will appear on the gadget. 'Size' sets the width (107) and height (12) of the gadget. 'Link' directs the gadget to a link function, which we'll add next.

Note that 'Gadget' is a window command and like others must be placed within a window structure.

Gadgets need to be told what to do when you use them. In Commander this is done with the 'Define Link' command. For example:

```
Define Link "ARexx" (
  Type ARexx (
    Data ( File "Rexx/Border.ifx" )
  )
)
```

A link has to have a name, in this case "ARexx", so that it can be connected to a gadget. Rather than just including the function within the 'Gadget' command itself, this modular approach has the advantage that you can use the same function in multiple places within the same definition, without wasting extra memory. It also makes it easier to move functions from one module to another.

This link is of type 'ARexx', which, in this case, instructs the gadget to execute the external script "Border.ifx" located in the "Rexx" directory.

Compile "Tutorial_3-D.def" and open the 'Tutorial_3-D.cmndr' module in Commander to see the results. If you click on the button gadget it will execute the defined script.

1.7 Language Overview

Language Overview

All commands, parameters, options, values, etc. must be separated by a SPACE (032), LF-line feed (010), or TAB (007) character. LF-line feed + CR-carriage return (010+013), such as used on DOS systems, is supported.

A command structure is a group of parameters applicable to a command. Command structures must be enclosed within brackets (). The end of structure indicator ')', must always be followed by a LF character (or LF+CR).

Except for TAB (007), ASCII characters from (000) to (031) are not permitted.

All commands, parameters and options are case sensitive. Therefore, 'Define Window' is accepted, but 'DEFine WinDOW' is not.

In most cases, it is permissable to order commands and parameters in any

manner wished. But, there are some restrictions, as noted in the reference section. Regardless of the semi-freeform capability, it is suggested that the recommended ordering and format be used.

1.8 Format

```
Format
------

There are several ways to format a definition (.def) file, but for
consistency, one is recommended. For example, the 'Define Font'
parameter, 'Style' could be formatted as:

The recommended format:

     Style (
       UNDERLINED
       BOLD
     )

is equivalent to (sometimes preferred):

     Style ( UNDERLINED BOLD )

and, is equivalent to:

     Style
     (
     UNDERLINED
     BOLD
     )

or even this format will work, but is not recommended:

     Style
       (        UNDERLINED
 BOLD
           )
```

1.9 Conventions

```
Conventions
-----------

bold Denotes required
plain denotes optional
bold+italic denotes optional-required.

[...] Denotes

<> Denotes required items. The '<' and '>' characters should not be
   included.
```

```
[] Denotes optional items. The '[' and ']' characters should not be
   included.

{} Denotes optional-required items. Only one of these options may be used
   within a structure but one of them is required. The '{' and '}'
   characters should not be included.

|  Denotes OR. Only one of these options may be used with this item.
   The '|' character should not be included.


<#>  Denotes an integer numeric value in decimal format (BASE 10). The
     range is determined by the parameter it is used with.

     Examples: 10,   9,   5,   127

<left>   Denote integer numeric values (BASE 10) in pixels, in the range
<top>    of 0 to n, where n is the maximum width of the screen/window.
<width>
<height>

<name> Denotes text information enclosed within quotation marks. For
       compatibility with ARexx support, spaces are not permitted.

       NOTE: <name>s are only used by the Module Compiler for
       reference, and are not stored in the output module.

<string> Denotes any text information enclosed within quotation marks.

<type> Denotes a type reference, as supported by the command it is used
       with. Only one type may be selected from those supported.
```

1.10 Root Command Order

```
Root Command Order
------------------

    Define Palette ()
    Define Font <name> ()

    Set Font <name> ]
    Set Color <#> ]

    Define Window <name> ()
    Define Link <name> ()

    Window <name> ()
```

1.11 Command Index

```
  Commands
   '

  Root Commands
   Define Font
   Define Link
   Define Palette
   Define Window
   Set Color
   Set Font
   Window

  Parameter Commands
   Border
   Gadget
   Image
   Text

  Gadget Types
   Type Button
   Type Cycle
   Type List
   Type MX

  Link Types
   Type ARexx
   Type Windows
```

1.12 ' Comment

```
'                            ' Comment                                  OPTIONAL
=============================================================================

Occurance: MULTIPLE

Description: The comment marker (apostrophe) can appear anywhere within a
definition file. All text after the comment marker, on the same line, is
ignored.

Comments can be very handy as reminders of how or why you did something a
certain way. Also, the comment marker can be very useful when debugging a
definition file, by providing a method of temporarly ignoring commands or
parameters, without having to delete the text.
```

1.13 Define Font

```
Define Font                                                             REQUIRED
=============================================================================

  Define Font <name> (
    Name <string>
```

```
    Size <#>
    Style ( [BOLD] [ITALIC] [UNDERLINED] )
  )


Occurance: MULTIPLE
Placement: before Window

Description: This command is used to open Amiga fonts for use with other
commands having text based parameters. The defined font becomes the
GLOBAL DEFAULT from this point on. At least one Define Font must be used.

PARAMETERS

Name                                                                REQUIRED
------------------------------------------------------------------------------
<string>

  This is the filename of a font contained within the FONTS: directory.
  The full name should be used.

  Example: Name "Topaz2p.font"

Size                                                                OPTIONAL
------------------------------------------------------------------------------
<#>

  This sets the pointsize, in pixels, to be used with the font. If this
  pointsize is not found, it will be created by scaling another size.
  Generally, scaled fonts look really bad. Therefore, it is recommended
  that existing font sizes be used.

  Default: 8

Style                                                               OPTIONAL
------------------------------------------------------------------------------
BOLD
ITALIC
UNDERLINED

  This is used to apply style(s) to a font. BOLD, ITALIC and UNDERLINED
  can be used in any combination. These styles are only applied to text
  displayed using the Text command. Other commands, such as Gadget, will
  ignore style definitions. At least one style must be set for this
  parameter.

  Examples: Style ( BOLD ITALIC )
            Style ( UNDERLINED )

  Default: Plain
```

1.14 Define Link

```
Define Link                                                         OPTIONAL
==============================================================================
```

```
  Define Link <name> (
    Type <type> ()
  )


Occurance: MULTIPLE
Placement: before Window

Description: This command is used to define links for use with other
definitions, as a way of supporting additional information/functions.

See Link Type descriptions for more information.
```

1.15 Define Palette

```
Define Palette                                                    OPTIONAL
=========================================================================

  Define Palette (
    <#> <r> <g> <b>
    [...]
  )

Occurance: SINGLE
Placement: before Window

Description: This command is used to modify colors within the palette.
Colors 0-3 are used for gadget and window rendering and can not be modified.


PARAMETERS

<#>

  Selects the color # within the palette. Range 4-31.

<r>

  Sets the RED component level from 0-15.

<g>

  Sets the GREEN component level from 0-15.

<b>

  Sets the BLUE component level from 0-15.
```

1.16 Define Window

```
Define Window                                                        REQUIRED
=============================================================================

  Define Window <name> (
    Position <left> <top>
    Position2 <right> <bottom>
    Size <width> <height>
    Font <name>
    Color <#>
    Border
    DragBar
    Title
  )


Occurance: MULTIPLE
Placement: before Window

Description: This command is used to define a window to which gadgets,
text, etc. can be attached. Multiple windows can be defined for
applications where one window is not enough. Using the Link Window
command, windows can be grouped together so that only one window in a
group is displayed at a time. For this reason, it is permissable for
windows to overlap each other. The Show Window command can be used to set
the front window at startup.

PARAMETERS

Position                                                             REQUIRED
-----------------------------------------------------------------------------

This sets the co-ordinates, in pixels, of the upper left hand corner
of the window. The values for <left> and <top> must be within the range
of the screen.

To avoid overlapping the screen's titlebar, it is suggested that a minimum
value of 11 be used for <top> when using an 8 point screen font.

Size
Position2                                                 OPTIONAL REQUIRED
-----------------------------------------------------------------------------
'Size' sets the dimensions of the window, in pixels. The values of
<width> + <left> and <height> + <top> must be within the range of
the screen. This is an alternative to 'Position2'. Use one or the other,
but not both.

'Position2' sets the co-ordinates, in pixels, of the lower right hand
corner of the window. The values for <right> and <bottom> must
be within the range of the screen and greater than the <left> and <top>
values of 'Position'. This is an alternative to 'Size'. Use one or the
other, but not both.

Font                                                                 OPTIONAL
-----------------------------------------------------------------------------

This sets the default font for this window. Unless otherwise specified by
```

the Font parameter within a Window command, this font will override the GLOBAL DEFAULT within any Window command that references this defined window.

If not included, the GLOBAL DEFAULT will be used.

Color OPTIONAL

This sets the default color for this window. Unless otherwise specified by the Color parameter within a Window command, this color will override the GLOBAL DEFAULT within any Window command that references this defined window.

If not included, the GLOBAL DEFAULT will be used.

Border OPTIONAL

This gives the window standard intuition borders.

DragBar OPTIONAL

This gives the window an intuition dragbar and allows it to be moved. This options forces the Border option to be in effect.

Title OPTIONAL

This gives the window the title <name>. This options forces the DragBar option to be in effect and allows the window to be moved.

1.17 Set Color

Set Color OPTIONAL
==========

Set Color <#>

Occurance: SINGLE
Placement: before Define Window

Description: This command is used to define the GLOBAL DEFAULT color for commands not including an optional Color parameter. If not included, an internal default will be used.

1.18 Set Font

Set Font OPTIONAL
==========

Set Font <name>

```
Occurance: SINGLE
Placement: before Define Window

Description: This command is used to define the GLOBAL DEFAULT font for
commands not including an optional Font parameter. If not included, an
internal default will be used.
```

1.19 Window

```
Window                                                          REQUIRED
========================================================================

  Window <name> (
    Gadget <name> ()
    Image <string> ()
    Text <string> ()
    Border ()
  )


Occurance: MULTIPLE
Placement: after Define Window / after Define Link

Description: This command is used to attach Gadget, Image, Text and
Border definitions to a previously defined window.

PARAMETERS

See PARAMETER COMMANDS.
```

1.20 Border

```
Border                                                          OPTIONAL
========================================================================

  Border (
    Position <left> <top>
    Position2 <right> <bottom>
    Size <width> <height>
    Fill <#>
    Frame 3D|3D-Recess|NONE
  )

Occurance: MULTIPLE

Description: This command is used to define a border to be drawn in a
window.

A border can be used as a method of visually grouping gadgets together, or
just to spice up the look of the display.

PARAMETERS
```

```
Position                                                          REQUIRED
-------------------------------------------------------------------------
This sets the co-ordinates, in pixels, of the upper left hand corner of
the border. The values for <left> and <top> must be within the range of
the referenced window.

Size
Position2                                                OPTIONAL REQUIRED
-------------------------------------------------------------------------
'Size' sets the dimensions of the border, in pixels. The values of
<width> + <left> and <height> + <top> must be within the range of the
referenced window. This is an alternative to 'Position2'. Use one or the
other, but not both.

'Position2' sets the co-ordinates, in pixels, of the lower right hand
corner of the border. The values for <right> and <bottom> must be within
the range of the referenced window. and greater than the <left> and <top>
values of 'Position'. This is an alternative to 'Size'. Use one or the
other, but not both.

Fill                                                              OPTIONAL
-------------------------------------------------------------------------
This is used to apply a fill color to a border. A value of 0, the default
background color, will have no visual effect.

NOTE: Some gadgets do not render properly when displayed over a
non-background (0) color. For this reason, fill may not always produce
expected or pleasing results.

If not included, no fill will be applied to the border.

Frame                                                             OPTIONAL
-------------------------------------------------------------------------
This is used to set the visual rendering of the border's edges.
```

1.21 Gadget

```
Gadget                                                            OPTIONAL
=========================================================================

  Gadget <name> (
    Position <left> <top>
    Font <name>
    Type <type> ()
  )


Occurance: MULTIPLE

Description: This command is used to define a gadget and its placement in
a window.

PARAMETERS
```

```
Position                                                          REQUIRED
-------------------------------------------------------------------------
This sets the co-ordinates, in pixels, of the upper left hand corner of
the gadget. The values for <left> and <top> must be within the range of
the referenced window.

Font                                                              OPTIONAL
-------------------------------------------------------------------------
This sets the font to be used when rendering the current gadget. This
value overrides the Window's Default and the GLOBAL DEFAULT.

Style definitions, within the referenced font definition, are ignored.

If not included, the Window's Default (if defined) or the GLOBAL DEFAULT
will be used.

Placement: before Type

Type                                                              REQUIRED
-------------------------------------------------------------------------
This sets the type of Gadget being defined. Type parameters are included
within this structure. See Gadget Types.

Placement: after Font
```

1.22 Image

```
Image                                                             OPTIONAL
=========================================================================

  Image <string> (
    Position <left> <top>
  )


Occurance: MULTIPLE

Description: This command is used to display an image within a window.

<string>                                                          REQUIRED
-------------------------------------------------------------------------
Defines the file name. This must include the full path.

Position                                                          REQUIRED
-------------------------------------------------------------------------
This sets the co-ordinates, in pixels, of the image. The values for
<left> and <top> must be within the range of the referenced window.
```

1.23 Text

```
Text                                                              OPTIONAL
=========================================================================
```

```
  Text <string> (
    Position <left> <baseline>
    Font <name>
    Color <#>
    Shadow
  )
```

Occurance: MULTIPLE

Description: This command is used to display text within a window.

Position REQUIRED

This sets the co-ordinates, in pixels, of the text. The values for <left> and <baseline> must be within the range of the referenced window.

Font OPTIONAL

This sets the font to be used when rendering this text. This value overrides the Window's Default and the GLOBAL DEFAULT.

Style definitions, within the referenced font definition, are supported.

If not included, the Window's Default (if defined) or the GLOBAL DEFAULT will be used.

Color OPTIONAL

This sets the color to be used when rendering this text. This value overrides the Window's Default and the GLOBAL DEFAULT.

If not included, the Window's Default (if defined) or the GLOBAL DEFAULT will be used.

Shadow OPTIONAL

Applies a shadow in color 1 (Black) offset 2 pixels to the right and 1 pixel down from the coordinates defined with 'Position'.

1.24 Type Button

Type Button
==========

```
  Type Button (
    Label <string>
    Link <name>
    Size <width> <height>
  )
```

Occurance: MULTIPLE

```
Description: BUTTON gadgets are used to execute internal macros or
external scripts.

PARAMETERS

Label                                                          REQUIRED
-------------------------------------------------------------------------
This sets the text that appears on the button gadget.

Link                                                           REQUIRED
-------------------------------------------------------------------------
Only 'ARexx' type links with one data may be used with 'Button' type gadgets.

Size                                                           REQUIRED
-------------------------------------------------------------------------
This sets the dimensions of the gadget, in pixels. The value of <width>
and <height> must be within the range of the referenced window.
```

1.25 Type Cycle

```
Type Cycle
=========================================================================

  Type Cycle (
    Size <width> <height>
    Data (
      <string>
      ...
    )
    Default <#>
    Link <name>
  )


Occurance: MULTIPLE

Description: CYCLE gadgets are used when a set number of options are
allowed (such as ON/OFF, 1/2/3/etc.)

PARAMETERS

Data                                                 OPTIONAL REQUIRED
-------------------------------------------------------------------------
This structure is used to define possible options for the gadget. There
must be at least two strings within the data structure.

      Example:
                              Data (
                                "Sawtooth"
                                "Pulse"
                                "Triangle"
                                "Square"
                              )
```

```
Default                                                              OPTIONAL
-------------------------------------------------------------------------------
This defines the default state of the gadget. The range of possible
values is 0 to n, where n is the number of data.

      Example: To set the default to "Triangle", in the above example,
          use:
                              Default 2

      Default: 0
    Placement: after Data

Link                                                                 OPTIONAL
-------------------------------------------------------------------------------

    Placement: after Data


Size                                                                 REQUIRED
-------------------------------------------------------------------------------
This sets the dimensions of the gadget, in pixels. The value of <width>
and <height> must be within the range of the referenced window.
```

1.26 Type List

```
Type List
===============================================================================

  Type List (
    Size <width> <height>
    Data (
      <string>
      ...
    )
    Default <#>
    Link <name>
  )


Occurance: MULTIPLE

Description: LIST gadgets are used when a set number of options are
allowed (such as ON/OFF, PULSE/SAWTOOTH/TRIANGLE/etc.). It is similar in
use to CYCLE gadgets but preferred when there are many options.


PARAMETERS

Data                                                  OPTIONAL REQUIRED
-------------------------------------------------------------------------------
This structure is used to define possible options for the gadget. There
must be at least two strings within the data structure.

      Example:
```

```
                              Data (
                                "Sawtooth"
                                "Pulse"
                                "Triangle"
                                "Square"
                              )
```

```
Default                                                          OPTIONAL
------------------------------------------------------------------------
This defines the default state of the gadget. The range of possible
values is 0 to n, where n is the number of data.

      Example: To set the default to "Triangle", in the above example,
          use:
                              Default 2

      Default: 0
    Placement: after Data

Link                                                             OPTIONAL
------------------------------------------------------------------------

    Placement: after Data


Size                                                             REQUIRED
------------------------------------------------------------------------
This sets the dimensions of the gadget, in pixels. The value of <width>
and <height> must be within the range of the referenced window.
```

1.27 Type MX

```
Type MX
========================================================================

  Type MX (
    Data (
      <string>
      ...
    )
    Default <#>
    LabelPlace LEFT|RIGHT
    Link <name>
    Spacing <#>
  )


Occurance: MULTIPLE

Description: MX (Mutually Exclusive) gadgets are used when a set number
of options are allowed (such as ON/OFF, PULSE/SAWTOOTH/TRIANGLE/etc.).
They operate similar to 'Cycle'gadgets, but display all possible options
at once.

PARAMETERS
```

```
Data                                                        OPTIONAL REQUIRED
-------------------------------------------------------------------------------
This structure is used to define possible options for the gadget. There
must be at least two strings within the data structure.

      Example:
                              Data (
                                "Sawtooth"
                                "Pulse"
                                "Triangle"
                                "Square"
                              )


Default                                                             OPTIONAL
-------------------------------------------------------------------------------
This defines the default state of the gadget. The range of possible
values is 0 to n, where n is the number of strings.

      Example: To set the default to "Triangle", in the above example,
          use:
                              Default 2

      Default: 0
    Placement: after Data

LabelPlace                                                          OPTIONAL
-------------------------------------------------------------------------------
      Default: LEFT

Link                                                                OPTIONAL
-------------------------------------------------------------------------------

    Placement: after Data

Spacing                                                             OPTIONAL
-------------------------------------------------------------------------------
The number of pixels between each row.

      Default: 1
```

1.28 Type ARexx

```
Type ARexx
===============================================================================

  Type ARexx (
    Data (
      File <string>
      String <string>
      [...]
    )
  )
```

Occurance: MULTIPLE

Description: AREXX link types are the way to control ImageFX, or any other ARexx capable program. Or can be used to provide additional functionality not capable with GUIF's own command set. Simple ARexx commands can be internal to a module (String), or larger external scripts can be used (File).

If there is only one data defined that ARexx File/String will be executed regardless of the value of the gadget that is linked to it.

Alternatively, the number of defined data must match the gadget which has linked to it. For example, if a 'Cycle' type gadget has 5 values, there must be 5 data defined here.

PARAMETERS

File	
String	OPTIONAL REQUIRED

'File' is used to link an external ARexx script to a gadget.

'String' is used to link an ARexx command string to a gadget.

1.29 Type Windows

Type Windows

```
  Type Windows (
    Data (
      <name>
      <name>
      [...]
    )
  )
```

Occurance: MULTIPLE

Description: WINDOW type links are used for grouping two or more windows together, so that only one window is displayed at a time. This is useful when all the parameters for a device can not be displayed in one window. This link type is commonly linked with a CYCLE or BUTTON type gadget.

1.30 Previous versions

Previous versions

```
- First release.
```

1.31 New in this version

```
00.Mar.25
---------

- First release.
```