



WIMPBASIC 2

User Guide

ProAction

3QD
DEVELOPMENTS LTD

APDL

WIMPBASIC 2

© 3QD Developments Ltd 2026
All rights reserved.

No part of this product may be reproduced in whole or part by any means without written permission of the publisher. Unauthorised hiring, renting, lending, public performance or broadcasting of this product or its parts is prohibited.

While every care is taken, the publisher cannot be held responsible for any errors in this product, or for loss of data or consequential effects from the use of this package.

Software and Documentation supplied by
APDL
a part of
3QD Developments Ltd
Single User version downloaded from the free of charge software section of
www.riscos.com

Please note that any free of charge downloads from RISCOS.com are supplied without any technical support package as per the licence agreement.

This document last edited on 29/09/26



If you have found this application and this manual useful please consider making a donation to help cover the costs of cataloguing, documenting, updating and hosting the APDL applications. Scanning the QR code will take you to our donations page.

Contents

WimpBasic User Guide and Tutorial	7	
About this manual	7	
Distributing an Application written with WimpBasic	7	
WimpBasic Version 2	7	
Introduction	8	
Conventions used in this guide	8	
Installation	8	
!WBModules	8	
Distributing your applications	8	
Installing !WBModules	8	
Getting Started	9	
The iconbar menu	9	
Library directories	9	
The Designer	10	
Main Window	10	
Program name	10	
Memory needed	10	I
conbar sprite	10	
Iconbar menu	10	
Select procedure	11	
Adjust procedure	11	
Quit function	11	
Load procedure	11	
Startup procedure	11	
Filetypes	11	
Include Files	11	
Window Viewer	12	
Add window	12	
Rename	12	
Export	12	
Delete	12	
Clear selected	12	
Display	12	
The Window Designer	13	
Create icon	13	
Sprite	13	
Radio	13	
Option	13	
Normal	13	
Default	13	
Menu Request	14	
Writable	14	
Left arrow	14	
Right arrow	14	
Up arrow	14	
Down arrow	14	
Modify Icons window	15	
Icon name	15	
Descrip.	15	
Text	15	
Size	15	
Sprite	15	
Border	15	
Colours	15	
Button type	15	
Procedure	16	

Allow	16
Disallow	16
Pointer	16
Other	16
ESG group	17
Menu	17
H centred	17
V centred	17
R justified	17
Shaded	17
Filled	17
Needs help	17
Keyboard	17
Multiple icons	17
Copying, Renumbering and Moving icons	18
Renumber	18
Moving icons	18
Editing the window	19
Title text	19
Title sprite	19
Size	19
Input focus	19
Title foregr	19
Title backgr	19
Work foregr	19
Work backgr	19
Scrol. Outer	19
Scrol. Inner	19
Button type	19
Procedure	19
Menu	20
H scroll	20
V scroll	20
Adjust	20
Pane	20
Title bar	20
Back icon	20
Close	20
Toggle	20
Movable	20
No bounds	20
Hot keys	20
Back window	20
Setting the window size	20
Window grid	20
Show Grid	21
Grid colour	21
Grid offset	21
Column width	21
Column gap	21
Row height	21
Row gap	21
Align position to grid	21
Align size to grid	21
Test window	21
Save window	21
Close window	21

The Event handler	22
What is an event ?	22
Event Handler window	22
Null event	22
Redraw event	22
Open window event	22
Close window event	22
Pointer leaving	22
Pointer entering	23
End drag event	23
Keyboard event	23
Menus Viewer	24
Menu Designer	25
Menu name	25
Description	25
Menu maker procedure	25
Menu selection function	25
Extend menu	25
Save	25
S-cut	26
Ticked	26
Fade	26
Div	26
Submenu/window	26
Procedure	26
Shortcut window	26
How shortcuts work	26
Variables viewer	27
About WimpBasic variables	27
The Variables menu	28
Add Group	28
Add variable	28
Rename	28
Export	29
Delete	29
Clear selection	29
Display	29
Code viewer	30
Add Code	30
Rename	30
Export	30
Delete	30
Clear selection	30
Select all	30
Display	30
Code Editor	31
Using WB Edit	31
WB Edit menu	31
Save code	31
Select	31
Edit	31
Display	31
Enter	31
Using other editors	31
WimpBasic interactive help	32

Compiling & Debugging	33
Sprites viewer	33
Raw handlers	34
Configuration	35
Close editor window after save	35
Double precision floating point	35
Save design before compiling	35
Blinking caret in editor	35
Text colouring in editor	35
Guidelines in window designer	35
Fancy fonts in help	35
Project vs Default configuration	35
Tutorials	38
HelloWorld	38
Reverse	43
Counter	45
Calculator	46
Calculator 2	49

WimpBasic User Guide and Tutorial

About this manual

This Manual is divided into three parts. The User Guide, the Tutorials, and finally the Reference Manual.

This first part covers the general operation of WimpBasic and its user Interface, and, with the Tutorials, is designed to be read and used until you are familiar with the way that WimpBasic works.

The third section describes the WimpBasic language and deals with the various keywords and components. In particular it details the differences with standard BBC Basic.

Distributing an application written using WimpBasic

You may freely distribute the directory !WBModules and the files contained within it ie !Boot, !Run, Copyright and WBModule as long as you acknowledge that they remain the copyright of APDL/ProAction by including the copyright file.

You may not remove any references made to WimpBasic in any of the files contained in your application or the !WBModules directory

You must advise users of how to make use of the !WBModules directory and take account of the fact that they may already have a copy of the !WBModules already on their system.

WimpBasic Version 2

WimpBasic 2 has various enhancements. There are:

- Modifications to the environment making it easier to create icons and windows. There is also an additional, integrated help system.
- New keywords split into various groups.
- Optimisation of applications enabling some applications to go faster.

Introduction

WimpBasic is a programming environment that lets you create RISC OS applications quickly and easily. The program interface, menus, windows and structure are created in the WimpBasic designer. Procedures or Functions are assigned to menus or buttons so the only code you have to write is the specific procedures and functions related to the application you wish to create. The structured way WimpBasic programs are created makes it very easy to reuse components in future applications.

We would appreciate you acknowledging fact that you used WimpBasic to create your application (there are some references in the !Run file). A reference on any banner that you put up at load time would be nice and/or a comment in the Info box as provided in the supplied Info files.

Please note that this is a request and we do not insist that you do it as part of the licence conditions. Our hope is that the more WimpBasic is seen the more widespread its use will become.

Note: Everything in WimpBasic is case insensitive ie

a\$ is the same as A\$

PROCDEMO is the same as PROCdemo or PROCDemo

Conventions used in this guide

We use the standard Acorn conventions for the mouse buttons:

SELECT is the left button, and the most commonly used.

MENU is the middle button and is used primarily for opening menus.

ADJUST is the right mouse button.

Unless otherwise stated **Select** or **Click** means move over something and Click on it with SELECT or ADJUST.

It is assumed that you are familiar with the basic concepts of using the RISC OS desktop environment. If not you may need to refer to your computer manual from time to time.

Installation

To install WimpBasic run the !WBINSTALL application on the CD and just drag the icon in the window which will appear to a suitable directory on your hard disc. This will install WimpBasic and the Wimp Basic run-time library (!WBModules).

The Tutorials and other example files can be accessed by double-clicking on !SUPPORT. You can load them directly from here, but it is probably best if you copy them to your hard drive so that they can be altered.

!SUPPORT also contains another copy of !WBModules

Note that every copy of WimpBasic contains an embedded serial number. This number is displayed in the Info box and on the 'start-up' banner. We can therefore trace unauthorised copies of WimpBasic and will take action against any person or organisation who distributes illegal copies.

!WBModules

The directory !WBModules is copied to the directory containing WimpBasic. These modules are required when running a compiled WimpBasic application. Note that only a single copy of !WBModules is required no matter how many WimpBasic applications you create.

Distributing your applications

You may (in fact you MUST) distribute a copy of !WBModules with any applications that you distribute, either freely or as a commercial activity. There is no royalty to pay for distributing !WBModules but you must comply with the Licence conditions referred to in this document

Installing !WBModules

By default the installer places !WBModules in the same directory as WimpBasic. When !WBModules is seen by the filer all WimpBasic applications then know where to locate it. There are three strategies that you (or your customers) can adopt:

1. Keep all WimpBasic applications in one directory that also contains !WBModules
2. Store !WBModules wherever you wish and ensure that it is Filer_Booted in your Boot sequence eg
Filer_Boot ADFS::HardDisc4.\$\$.WimpBasic.!WBModules
3. On a RiscPC put !WBModules in the \$.!Boot.Choices.Boot.Tasks directory.

Note: In all cases the modules are only loaded when a WimpBasic application is run.

Getting Started

Although this chapter is called 'Getting Started' you are strongly advised to read it in conjunction with the later **Tutorial** section.

The iconbar menu

This is accessed by clicking MENU on the iconbar icon.

Info leads to the usual info box about the program. This also shows the name of the Registered User and the program's Serial Number.

Configure opens the Configure window.

Library leads to the Library menu. Each item opens the relevant library directory. If you click on the Library item on the main menu the root Library directory opens showing all sub directories.

Windows opens the Window library directory. To load a file drag it into the Windows Viewer window.

Icons opens the Icons library directory. To load a file drag it into the Windows Designer window.

Menus opens the Menus library directory. To load a file drag it into the Menus Viewer window.

Code opens the Code library directory. To load a file drag it into a Code editor window.

Sprites opens the Sprites library directory. To load a file drag it into the Sprites Viewer window.

Remove project removes the current project from memory.

Quit quits the Wimp designer.

Library directories

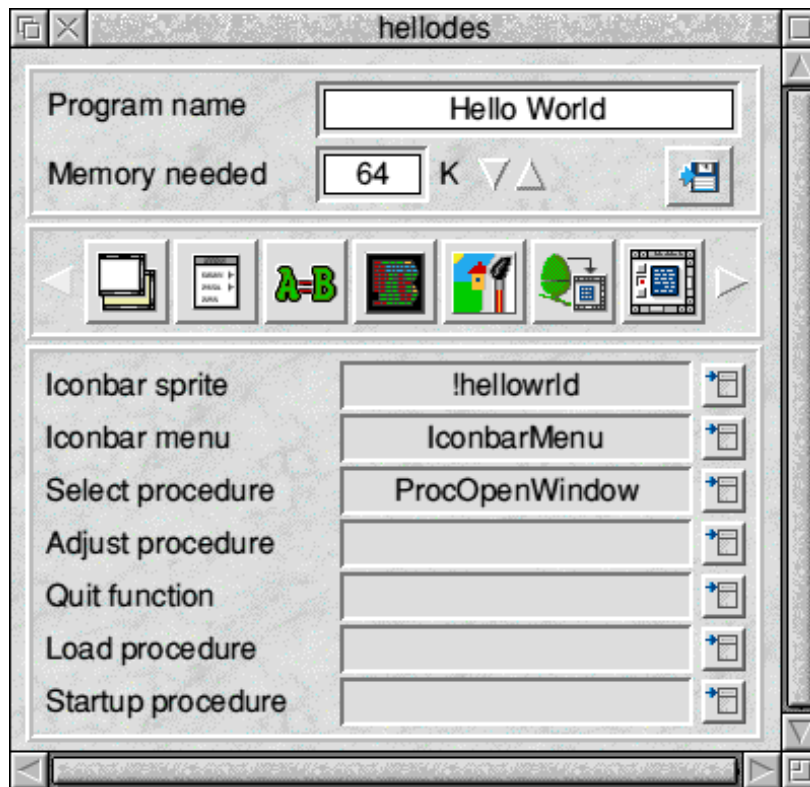
A library directory is simply a directory where you can store useful code, sprites, menus and windows that could be reused in other projects. This is normally inside the !WimpBasic application directory. The location of the library directories is set in the !Run file of the !WimpBasic application. On a network it could point to the same directory for all the computers.

The Designer

This is where you create the framework and structure of your application. The designer handles the tedious Wimp programming for you, allowing you to concentrate on the important aspects of the application. The Designer is accessed via the Main Window.

Main Window

Clicking on the WimpBasic icon opens the Main Window. Through this and the Main Window Menu all other parts of the designer are accessed.



The various icons in the main Window will now be described.

Program name

Enter the program name here. This appears in the Task Manager window when you run your program.

Memory needed

This is the memory allocated to your program when it starts. WimpBasic will warn you during compiling if this is too small for your program. If your program needs more memory when running, WimpBasic will try to increase the memory allocated to it as needed.

Iconbar sprite

Use the menu icon to open a sub menu showing Application icons and Wimp icons. Application contains any icons that you have defined for use with the application. Wimp contains all icons known to the Wimp at the current time ie all of those 'seen' by the system. Select the required sprite from the menu and its name appears in the main window. This icon appears on the iconbar when your application is run.

Iconbar menu

The menu icon lets you select the menu to attach to your application's icon on the iconbar. When you have selected the menu you can go directly into the **Menu Designer** by just clicking on the menu name. This applies everywhere in WimpBasic, if you see a Code, Window or Menu name in an icon like this, clicking on it will take you to the appropriate editor for the item. If you click on a Code icon with no text in it the editor will pop up with a 'template' code item to show the parameters and return conditions required.

Select procedure

This lets you choose the procedure to call when the user clicks on the iconbar icon of your application using SELECT.

Adjust procedure

This lets you choose the procedure to call when the user clicks on the iconbar icon of your application using ADJUST.

Quit function

This lets you choose the function to call when the Task Manager wants to quit the program.

When the user asks the Task Manager to quit the program or is quitting the desktop, the Task Manager warns all programs about it. Programs are expected to use this message to ask the user to save any unsaved data before quitting. The Quit function will be called when the Task manager is giving this warning. If your program has any unsaved data it should ask the user to save it. If the user does save it or the data hasn't been modified or doesn't need saving then you should return TRUE. Return FALSE to tell the Task manager that your program shouldn't be terminated. TRUE and FALSE are constants, if you are unfamiliar with the concept look in the BASIC manual that came with your computer.

Load procedure

This is called whenever the user drags data into your program. You can, in the procedure, decide to load the data or ignore it.

If you decide to load the file you should tell WimpBasic if you loaded it successfully. You do this with the **LoadOK** and **LoadFailed** statements.

Startup procedure

This will be called when the program starts running. It is useful for setting up variables and displaying startup windows.

The main window also has a set of scrolling icon buttons which are, from the left; open Windows Viewer, open Menus Viewer, open Variables Viewer, open Code Viewer, open Sprites Viewer, setup Raw Events and Configure Project.

These buttons have corresponding entries in the Main Window Menu and will be described fully later.

The last two icons on the right do not have corresponding menu entries and are:

Filetypes

This lets you select which filetypes your application will accept for loading. It also modifies the !Boot and !Run files to set the appropriate alias so if you double click on a file of that type your application will load it automatically.

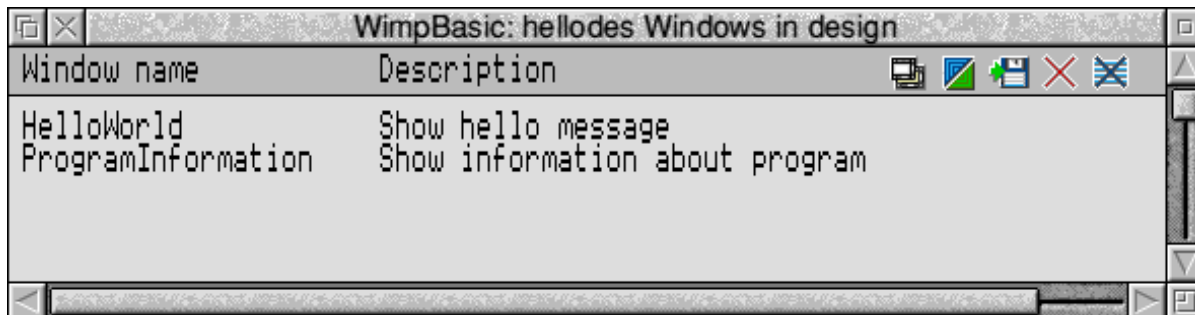
Include Files

This opens a window where you can drag files and directories which will be automatically placed inside the application's main directory. This saves having to constantly place these files there manually each time you recompile your project. The files are stored inside the application's Design File and can also be exported if required.



Window Viewer

The Windows Viewer is accessed from 'Windows' in the Main Menu or by clicking on this icon. In the Windows Viewer you will see a list of all the windows in the design along with their descriptions.



The Viewer Menu is accessed by clicking MENU over the window.

Some menu items have corresponding icons on the Viewer Toolbar, which are shown for each item.



Add window

Opens a 'bare' window for you to begin your design. (See *Window Designer* later).



Rename

Renames a window.



Export

Exports all tagged windows or, if none are tagged, all windows are exported. When saving you are given the option to also save the variables and/or code associated with the window(s). This is useful when creating stand-alone windows which can then be imported into any design without any change to code.



Delete

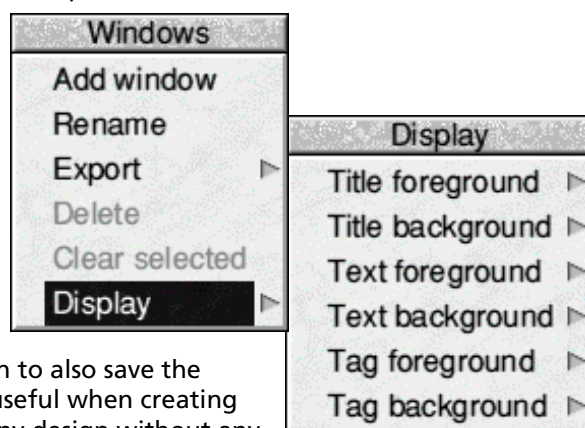
Deletes all tagged windows, prompting first.

Clear selected

Untags all tagged windows. *Tagged* means selected with the mouse. The first is selected with SELECT and additional selections made with ADJUST.

Display

This is only available via the menu. It has a submenu which lets you select colours for the viewer. The selected colours are also used by the Window, Menu, Code and Sprites viewers and the Throwback Window. To edit a window definition double click on its name.

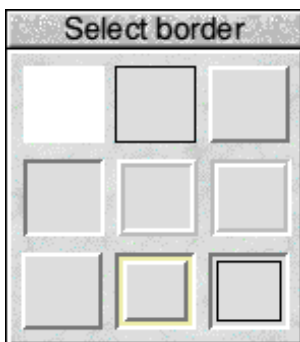


The Window Designer

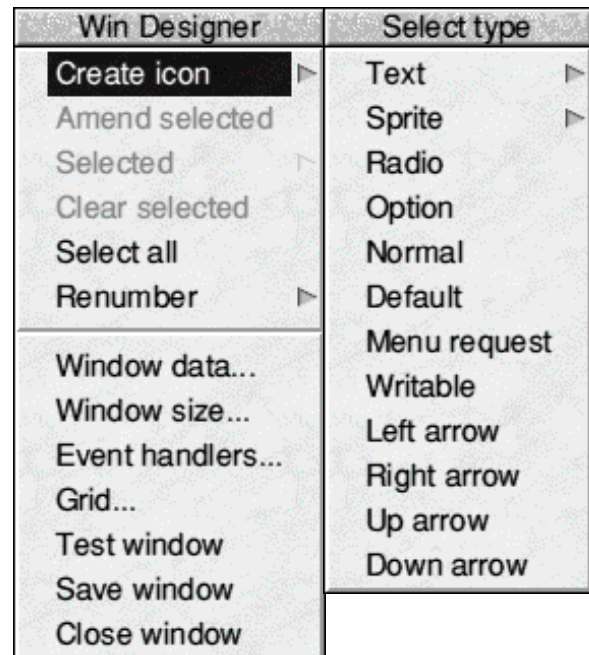
When you select Add Window an empty window appears. This is where you define a new window. Click MENU over this window to open the Window designer menu.

Create icon

This sub menu provides a range of different icon types that can be added to your window. Alternatively clicking on the 'Create icon' menu item itself will add a text icon with no border (the default) or, if you have previously selected something else then that is used as the default. The icon is added to the window near to the location the mouse pointer was when you opened the menu.



The 'Text' and 'Sprite' items on this menu lead to a window from which you can select the border type. Clicking on an icon type, or on one of the borders in the Text or Sprite sub menus, open a window asking you for the icon name and description.



Enter the name you want to use for referring to this icon and a description, to remind you of its function. The icon name must be a valid string variable name eg *Mylcon\$*. See your BASIC manual for information about variable names. When you click on OK the icon appears in the window. Remember that all variables in WimpBasic are case insensitive.

Note: The first item in the Text and Sprite sub menus is 'no border at all' and this is followed by icon type 0, 1, 2 etc as detailed in the PRMs. Icons 5 and 6 both highlight when clicked on. These are the same as Normal and Default respectively. You can move the icon in the window by dragging with SELECT. You can alter the icon size by dragging with the ADJUST near its edge.

Sprite

Creating a sprite icon is the same as the Text icon above except that the created icon has a default *Happy face* sprite assigned to it. You change the sprite using the *Amend selected* menu item.

Radio

This produces a Radio button which can be altered using *Amend selected*.

Option

This creates an Option button which can be altered using *Amend selected*

Normal

Select this to create a 'Normal' icon. This is a standard slabbed icon which can be altered using *Amend selected*.

Default

Creating a Default icon produces a button with yellow border. This is the button that should indicate the default action taken in a dialogue box if RETURN is pressed. The text can be altered using *Amend selected*.

Menu Request

Creates a Menu Request icon which can be altered using *Amend selected*.

Writable

This item creates a Writable icon which can be altered using *Amend selected*. Writable icons allow text to be entered by the user.

It is best to place as many characters in the writable icon as you are likely to need, otherwise its size is set to one. eg for a numeric field that needs numbers up to 6 figures enter '999999'. The correct length is then set automatically. You will have to clear the icon when you run the program.

Left arrow

This creates a Left Arrow icon which can be altered using *Amend selected*. Arrows are used to alter values in icons or move through a list of entries.

Right arrow

This is the same as Left arrow except for the direction.

Up arrow

This is the same as Left arrow except for the direction.

Down arrow

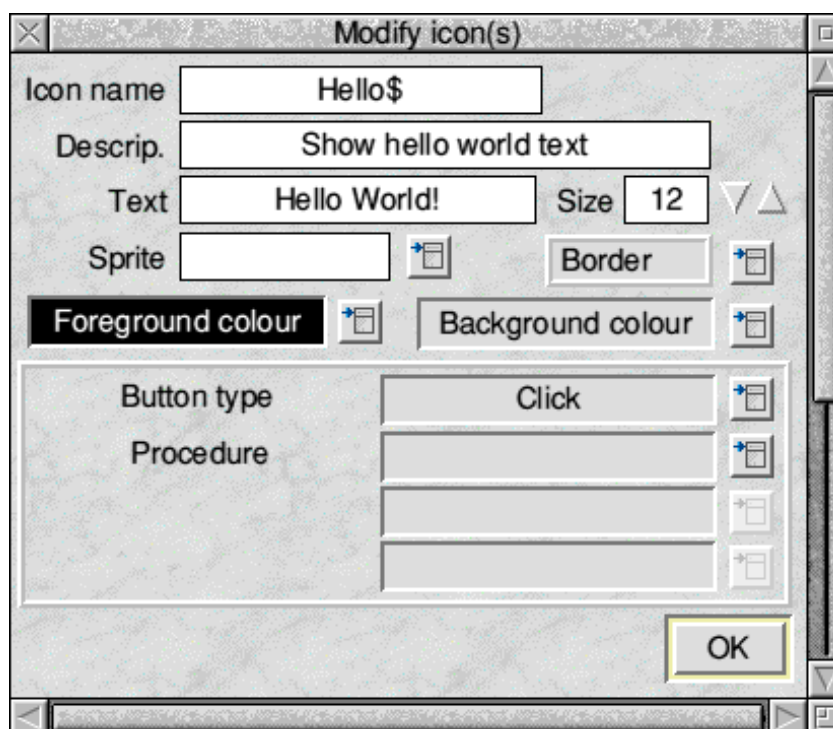
This is the same as Left arrow except for the direction.

Modify Icons window

This opens when you click on 'Amend' on the menu. It has two uses.

First, if you have no icons selected and you click MENU over an icon the menu item shows 'Amend Icon'. If you now click on this the Modify Icons window will open showing the icon's name and attributes. You can now modify this icon.

Secondly, if you select an icon or icons (multiple selections are done using ADJUST) the menu item shows 'Amend selected'. Clicking on this opens the Modify Icons window but this time the Name and Description fields are greyed out. Changes made will now be applied to all selected icons.



Icon name

This is the variable name you will use to refer to this icon in the code. If you are modifying a selection of icons (Amend selected), this entry is shaded as icon names must be unique.

Descrip.

Here you enter the icon description. This is shaded if you are amending a selection of icons. Description is like a comment in code - just to remind you what the purpose of the icon is.

Text

Enter text that will appear in the icon here.

Size

Enter the maximum number of characters that can be displayed or entered in a text icon.

Sprite

Enter the sprite name that is to appear in a sprite icon.

For some button types it is possible to have two sprites. In this case you separate them with a ',' For example. radio buttons have the sprites optoff,opton. You can select sprites from the Wimp sub menu or from the Application sub menu. In the case of the Application sub menu you must have already loaded the sprites into the design. Sprites are loaded in the Sprite viewer window by dropping the file onto the window. The file can be a single sprite or a multiple sprite file.

Border

To the right of the Sprite icon is the Border icon. Click on the menu icon to open a window from which you can select from all available border types.

Colours

Below are the Foreground and Background colour items. Click on the menu icon to open a menu with colours to select from. Note that the colours are those suggested by Acorn for use in all modes.

Button type

Click on the menu icon to open a menu of possible button types. 12 (Undefined) and 13 (Undefined) are shaded and cannot be selected.

The button types are;

- **Ignore** tells RISC OS to ignore clicks on the icon - you will never get an 'event' from that icon. Used for labels and decoration icons.
- **Always** This button type sends an 'event' whenever the pointer is over it. You don't have to click, it's enough to be over the icon to send an 'event' continuously. Not often used in normal programs.
- **Auto repeat** When the icon is clicked on an 'event' is sent. If the mouse button is held down the program will receive 'events' until it is released. Useful for changing numbers with up/down arrow icons.
- **Click** An event is sent when the user clicks on this icon.
- **Release** An event is sent when the user releases the mouse button over this icon.
- **Double click** Although this is just called double click it also sends a Click event. That is, when the user first clicks the mouse the program gets a Click event but with the second click a Double click event.
- **Click & Drag** This event is sent when the user clicks on the icon and starts a drag action (pressing the mouse button and holding it while moving the mouse).
- **Release & Drag** The same as Click & Drag except the first event comes when the user releases the mouse button.
- **Double Click & Drag** The program receives a Double Click event and a Drag event.
- **Menu** The program receives an event when the user clicks on the icon. The icon is also 'selected' by the wimp while the mouse is over it, similar to the action seen when moving the pointer over a menu.
- **Double/Click & Drag** The program receives a Click event, Double Click event and a Drag event.
- **Radio** No events are generated by these icons. However the state of the icon is changed from Selected to Unselected or vice-versa. Used for Option buttons.
- **Write/Click/Drag** The program receives a Click event when the user clicks on the icon and a Drag event when the user starts a drag action over the icon. The caret is placed in the icon when clicked on.
- **Writable** No events are generated by this icon. The caret is placed into the icon when the user clicks on it.

Procedure

The menu con lets you select which procedure is called by each event.

If you scroll the window down or click on its Toggle Size icon some more icons will appear below the OK button.

Allow

Here you can enter a list of characters allowed in a writable icon. You can specify a range, such as all numeric characters 0-9, or a range and a single character by separating them with a semi colon. For example 0-9;a-z would allow all numeric characters and a-z whilst 0-9;a would allow all numeric characters and only the letter a.

Disallow

The same as Allow except these are the characters that are *not* allowed in a writable icon.

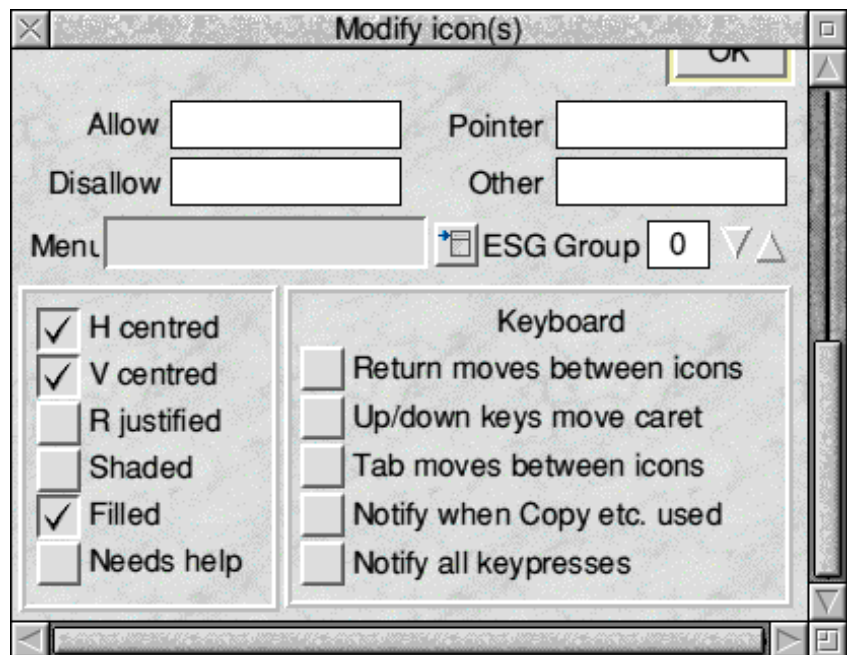
Pointer

You can enter the name of a pointer sprite here and each time the pointer moves over the icon the pointer will change to the named sprite. Pointer sprite names usually start with *ptr*. Some standard pointer sprites are: *ptr_menu*, *ptr_write*. You can also define your own sprites in !Paint.

Other

Lets you enter any other information for the icon validation string. Note: the acceptable values can be found in the Acorn reference manuals. Some useful ones are listed here:

- L Allow multiple lines in non writable icons eg Program information box. The text is automatically word wrapped.
- K Give certain keys special significance. K is followed by one or more of the following.



- R Pressing RETURN moves the caret to the next writable icon.
- A Allow Up and Down arrow keys to move between writable icons.
- T Pressing Tab moves forwards to the next writable icon and Shift Tab moves back to the previous writable icon.
- D Follow this by a '*' to hide from the user, anything typed in. The data is replaced by the star character. This is usually used for password entry as the data is still passed to the application.

There are some additional commands but you should refer to the PRMs for more information. A useful combination is KAR, which is the the most commonly used sequence and is easy to remember. The order of the writable icons is dictated by the icon numbers. If necessary you can re-number them. See *Renumber* for more information.

ESG group

This stands for Exclusive Selection Group. You can group radio button icons together so that only one of them can be set at a time. You do this by giving them the same ESG group number. ESG group 0 is taken as no group. As one button is selected all other buttons in the same group are automatically de-selected.

Menu

Use the menu icon to select a menu which should appear when user clicks with menu button on the icon.

H centred

Ticking this centres the text horizontally in the icon.

V centred

Ticking this item centres the text vertically in the icon.

R justified

Ticking this item right justifies the text in the icon and, if it is a sprite+text icon, eg. an Option button, moves the 'button' to the right.

Shaded

Ticking this item makes the icon shaded ie it can't be selected.

Filled

Ticking this item gives the icon a background.

Needs help

If there are user drawn graphics in the icon and this flag is set then changing the icon generates a Redraw event.

Keyboard

This section enables you to enter keyboard validation strings using tick boxes. When you select the boxes the relevant string is entered into the Other writable icon.

Multiple icons

If you are changing a selection of icons, a window appears for you to indicate which changes should affect the icons.

Hint: You can 'copy' attributes from an icon to a selection of icons. You do this by making sure no icon is selected, then select the icon you want to copy attributes from, then select all the icons you want to copy the attributes to. Select 'Amend selected' from the menu and click on OK in the window. Select the attributes you want to change and click on OK. All selected icons should now have the same attributes as the first icon.

Copying, Renumbering and Moving icons

From the main Window Designer menu you will find that 'Selected' leads to a sub-menu.

The first item, **Copy**, leads to its own sub menu. When you select one of the items from here all selected icons are copied in the direction you select.

When you select **Delete** all selected icons are deleted. Be aware **there is no undelete!**

Selected	Copy
Copy ▶	Up ^U ▶
Delete ^X	Down ^D ▶
Align ▶	Left ^L ▶
Save ▶	Right ^R ▶

Align again leads to a sub-menu. When you select one of these, all selected icons are aligned to the relevant position, relative to the most distant icon in the relevant direction. That is, when selecting Left, all selected icons align their left border to the position of the left most icon.

Align
Top
Bottom
Left
Right

Save leads to a 'Save as' window from which you can save the currently selected icon or icons so that they can be used in other applications. It is advisable to save common icons in *!WimpBasic.Library.Icons*.

Renumber

This entry has two sub menus. **New number** lets you enter a new icon number for the icon that the pointer was over when you opened the menu. **By clicking** allows you to quickly renumber icons by pointing at them in turn and clicking. The mouse pointer changes to the number used for renumbering and will count as you renumber icons.

There are various reasons why you might need to renumber icons. If you have writable icons, the caret will be placed in the icon with the lowest icon number when the window opens. When you use arrow keys to go between icons the icon number is used to decide which icon the caret enters next.

Higher numbered icons will be 'on top' of those with lower numbers, and so might obscure them. Also if a higher numbered icon is over one with a lower number it will not be possible to select the low numbered icon as the high numbered icon will prevent mouse click from reaching it.

The icon number also controls the order in which the icons are redrawn; for example in frames and backgrounds.

Moving icons

You can select icons using the SELECT and ADJUST mouse buttons. If you have selected icons and the window has the input focus you can move the icon(s) using the arrow keys on the keyboard. Holding down SHIFT while you press the arrow keys makes the icons move faster.

Editing the window

Selecting 'Window data' from the Window Designer menu opens a window from which you can set and alter the window data.

Title text

Enter the text you want to appear on the window's Title bar

Title sprite

Enter the name of the sprite you want to appear in the window title or select a name from the menu by clicking on the menu button to the right of the writable icon

Size

Enter the maximum size for the Title bar text. You need only do this if you intend to change the text from within the program.

Input focus

This is the colour you want the Title bar to be when the window has the input focus. Usually this is a cream yellow colour.

Title foregr

This is the Title bar foreground (text) colour - usually black.

Title backgr

This is the Title bar background colour - usually colour number 3.

Work foregr

This is the Work area foreground colour - usually black.

Work backgr

Work area background colour. Note it is possible to set the background colour to None. This means that the Wimp will not clear the background of the window when it is redrawn so you will need to do it yourself.

Scrol. Outer

This defines the Scrollbar outer colour - usually colour 3.

Scrol. Inner

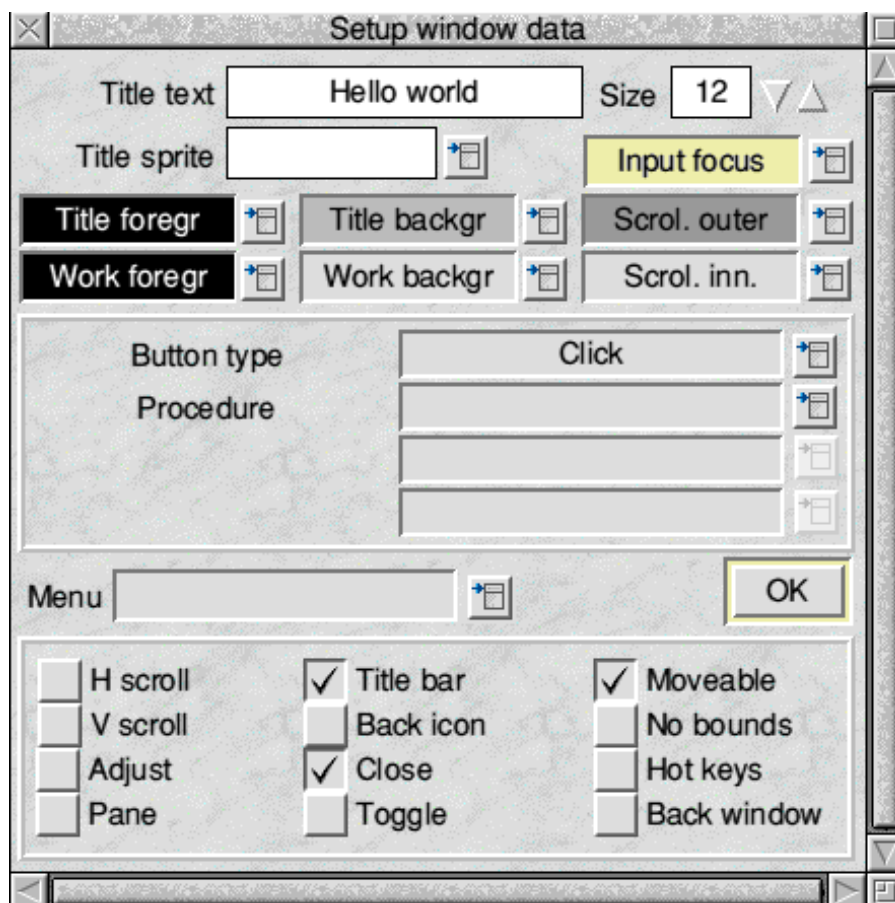
This defines the Scrollbar inner colour - usually colour 1.

Button type

This sets the 'button type' of the window's background. If this is set then clicking on a window's background returns the same type of event as clicking on an icon.

Procedure

This identifies the Procedure that you want to call when the user takes the appropriate action on a button.



Menu

This indicates the Menu you want to be opened if the user clicks the mouse MENU button inside the window.

H scroll

This indicates that the window is to have a Horizontal scroll bar.

V scroll

This indicates that the window is to have a Vertical scroll bar.

Adjust

This indicates that the window is to have an Adjust size button.

Pane

This indicates that the window is a Pane window. This is a window that follows another window's movement, like the Tool window in Draw.

Title bar

This indicates that the Window has a title bar.

Back icon

This indicates that the Window has a Back (send window to back) icon.

Close

This indicates that the Window has a Close icon.

Toggle

This indicates that the Window has a Toggle size icon. this toggles between the last two sizes of the window.

Movable

This indicates that the Window is movable.

No bounds

This indicates that the Window can be moved off the screen.

Hot keys

This indicates that key events will be sent to this window. This is used to implement shortcuts that should function while the program is working. Normally this type of window is opened off screen so it isn't visible.

Back window

This sets the Window as a background window. Like backdrop in a RISC OS 3 window it will always stay in the background.

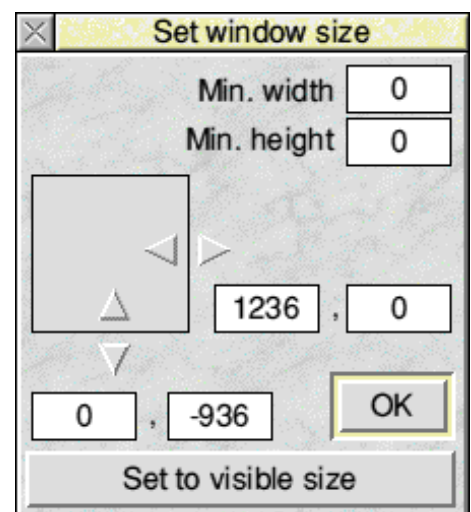
Setting the window size

Selecting 'Window size' from the Window Designer menu opens a window from which you can set and alter the window's size. Use the arrow icons to change the window size. 'Min. width' and 'Min. height' can be used if you want to limit the window's minimum size. 'Set to visible size' sets the values to the area that is currently visible in the window.

Click on OK to set the size.

Window grid

Select 'Grid' from the Window Designer menu to open the Grid dialogue.



Show Grid

Tick this box if you want the grid visible in the Window Designer.

Grid colour

Use the menu icon to open a menu from which you can select the grid colour.

Grid offset

Sets the X and Y offsets of the grid. This has the effect of adding the offset to the left and top of the grid eg, using the default grid of 128 * 64 you can add a half grid offset with the figures 64 & 32. This enables you to set an icon position near to the edge very accurately.

Column width

Sets the column width in pixels.

Column gap

Sets a gap between columns in pixels. The gap area is ignored when using the Align options below.

Row height

Sets the row height in pixels.

Row gap

Sets a gap between rows in pixels. The gap area is ignored when using the Align options below.

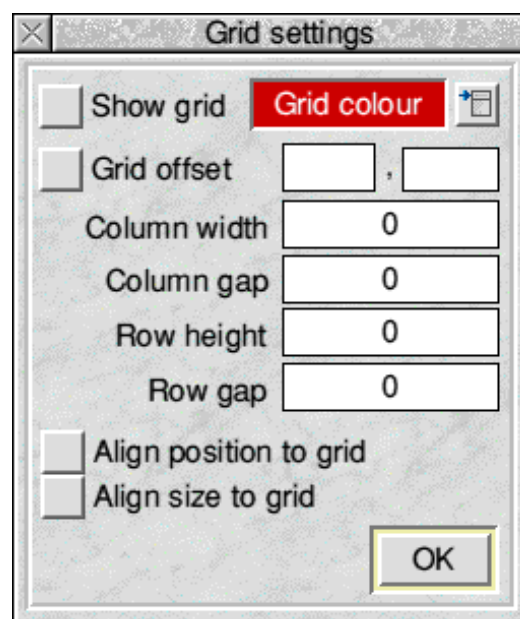
Align position to grid

Tick this to align the icon(s) with the grid intersections. Alignment only takes effect when an icon is moved.

Align size to grid

Tick this to adjust the icon(s) with the grid intersections when you re-scale them. Alignment only takes effect when an icon is rescaled or moved.

When you have finished Click on OK to confirm the settings.



Test window

Normally, when you are designing, icons don't work as they will in the finished program. They are all writable and can be dragged around. If you want to make the icons work as they will in a real program select 'Test window' from the Window Designer menu. This will make all icons behave correctly. Click 'Test window' again to return to editing the icons.

Save window

This is used when you have finished defining your window. A window pops up asking you for the window's name and description.

Note: The position that the window is in when you save it is the position that it will open at by default.

Close window

Closes the window you are over without saving it.

The Event handler

What is an event ?

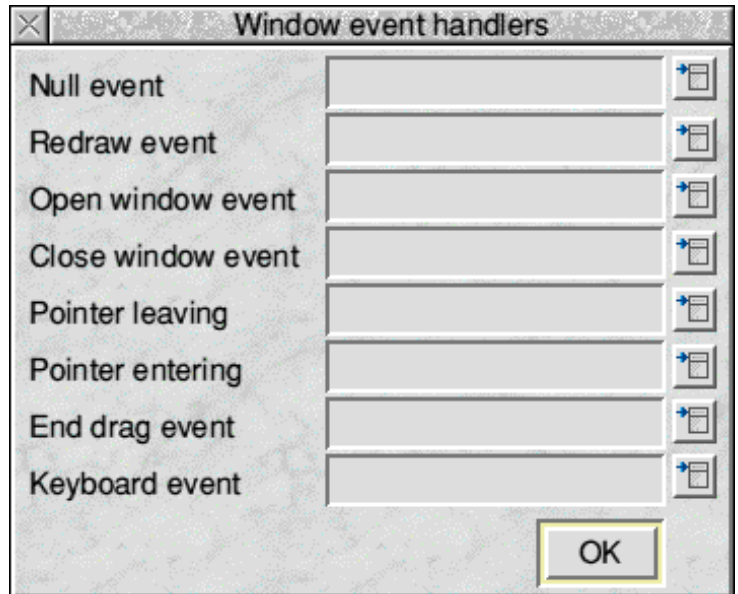
If you have an icon with the button type 'Click', RISC OS will inform the application when the user has clicked on the button. This is called an 'event'. If you change the button types you will see that, in some cases, the text below the Button Type box changes. This allows you to attach a procedure to an 'event'. There can be up to three procedures attached to a single icon. For example in Double/Click & Drag you can attach procedures to (a) Double click, (b) Drag and (c) Click. When the events are received by your application the appropriate procedure is called.

Event Handler window

Selecting 'Event handlers' from the Window Designer menu will open the Event Handler window.

Here you can set the procedures to handle wimp events. This is a very easy way to deal with wimp events. In each window you can assign a procedure that will be called if a particular wimp event takes place. In many cases WimpBasic handles situations automatically, but you have the ability to do something that is customised to your requirements.

There is a file in the Tutorial directory, called *Events*, that makes things a little clearer. Do not confuse these Events, which are specific to a window, with the Raw Handler Events, which are general.



Null event

This is an event that tells your program that nothing is happening. Not very useful you might think, but it is. This event is connected to code that has to be run even though nothing is happening. For example in a clock application, to update the clock display or to provide a limited time for a window to be on screen - see the tutorial application, *Warn*. This event is also used for tracking the pointer's location when the user is dragging.

Redraw event

This event tells your program to redraw a window. Normally you don't need this in WimpBasic as you can use the *Output* statement to make the window remember what is in it, but in some cases you may find this useful.

For example in a clock application it would be possible to plot the background graphics via the *Output* statement but plot the actual clock hands in the redraw routine. That way the background graphics code would only have to be executed once instead of at every redraw event.

Open window event

This event occurs when the user moves or resizes a window. If you don't have any procedure connected to this event WimpBasic takes care of opening the window in the correct location.

Close window event

This event occurs when the user clicks on the Close icon in the window. If you don't have any procedure connected to this event WimpBasic closes the window for you.

Pointer leaving

This event occurs when the pointer moves outside the window. It is used to modify the mouse pointer back to the original if you have changed it.

Pointer entering

This event occurs when the pointer moves into the window. It is used to modify the mouse pointer to a special pointer (crosshair for example) in your application.

End drag event

This event occurs when the user stops a drag action (releases all mouse buttons).

Keyboard event

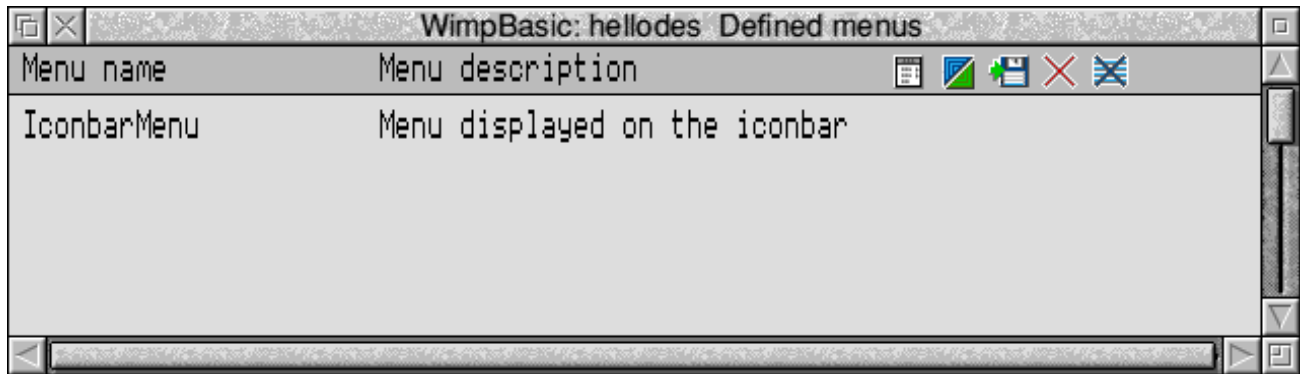
This event occurs when the user presses a key on the keyboard and the window has input focus. The attached procedure then processes the keyboard input.



Menu Viewer

Click on this icon in the Main Window or select 'Menus' from the Main Window menu to open the Menu Viewer.

You will see a list of all menus in the design with their description. You will notice that this window and the way it works is similar to the Window Viewer previously described.



The buttons on the toolbar along the top are described below. All these functions are duplicated on the Menu Viewer menu.

 The first button opens a window in which you design the menu in. See the section on the Menu Designer for more information. This is duplicated by Add menu on the menu.

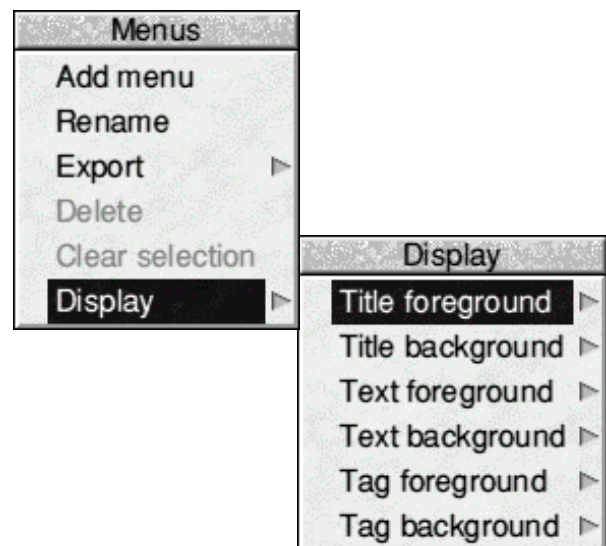
 The second item is used to rename menus. Again this is duplicated on the menu.

 This item lets you export all tagged menus or, if no menu is tagged, all the menus in the design.

 This deletes all tagged menus.

 This untags all tagged menus.

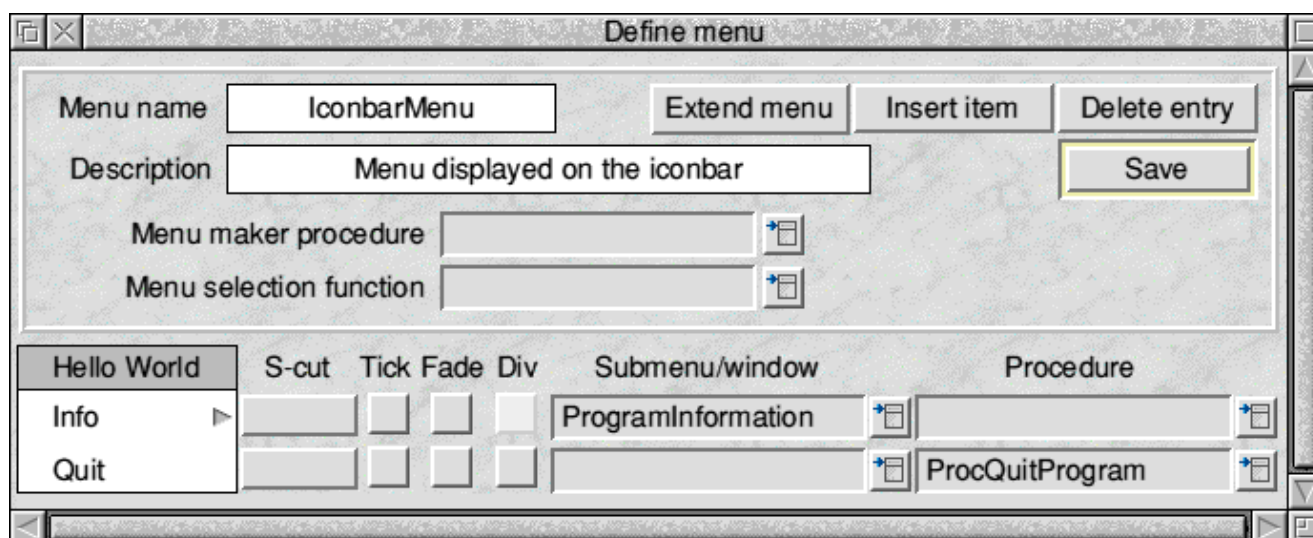
The final item on the menu, **Display**, has no equivalent button. It has a submenu which lets you set the colours used in the viewer. The Window, Menu, Code and Sprites viewers and the throwback window all use these same colours.



Menu Designer

If you select 'Add menu' from the Menus Viewer menu or click on the first button on its button bar the Menu Designer window will open.

To edit or modify an existing menu double-click on it in the Menu Viewer window and the Menu Designer window will also open, but this time with all the data from the selected menu. The illustration below shows this with the menu from the Hello World program



Menu name

Here you enter the name of the menu. This is the name you will use to reference this menu. This is not the menu title that appears in the menu header bar.

Description

Here you can enter a description of the menu. This is for reference only. It's main purpose is to act as a reminder so that, in programs with many menus, you will know what each is used for.

Menu maker procedure

This is a procedure that may be called before the menu is opened. This could be used to set up the menu, for example ticked or faded menu entries. For example;

```
DefProcmenumaker
if Global:headings% then
    MenuTickEntry menu,4,TRUE
else
    MenuTickEntry menu,4,FALSE
endif
EndProc
```

Menu selection function

This is a procedure that may be called when the user makes a selection from this menu. It is usually not used because it's easier to make WimpBasic handle the menu selection, but if you are using a menu which is extended when the program runs then this routine is used to handle the selection.

Extend menu

Adds one empty entry to the menu to add a new menu item.

Save

Saves the menu in the Defined menus window.

The lower section of the window has a dummy menu header (coloured grey) and, initially, space for one menu

item. You can type directly into the grey menu title to give it your required name, which should have a maximum of 12 characters. You can also type entries directly into the menu to create your menu. A maximum of ten entries per menu is allowed and a menu entry can be up to 40 characters. The icons on the same line to the right of each menu entry define the attributes of the menu item.

S-cut

This opens the Shortcut window (see later). This is used to attach keyboard shortcuts to menu entries.

Ticked

This indicates if a menu item should be ticked or not.

Fade

This indicates whether the menu item should be faded or not. To edit a faded menu item you must first de-select Faded.

Div

Select this item to place a divider line in the menu. The line will be above the item, which is why you cannot select this for the first entry.

Submenu/window

Here you can select a sub menu or window to the menu item. This will automatically be opened by WimpBasic.

Procedure

This is the procedure that will be called when the user selects this entry from the menu.

Shortcut window

If you click on one of the S-cut icons in the Menu Designer window the Shortcut window will open. This is used to associate a keyboard shortcut with the menu item.

You will see a display of all the normal keys similar to the usual keyboard layout, and below it a display of the Function keys. At the bottom are two option buttons' Control and Shift.

To set a keyboard shortcut just select the key(s) you want to use. For example, to set a shortcut using Ctrl-A select the A icon by clicking on it and select the Control option.



How shortcuts work

If a key is pressed and it is not used by any of the Key Event Handlers in your program, WimpBasic will check to see if a shortcut is set for the window with input focus.

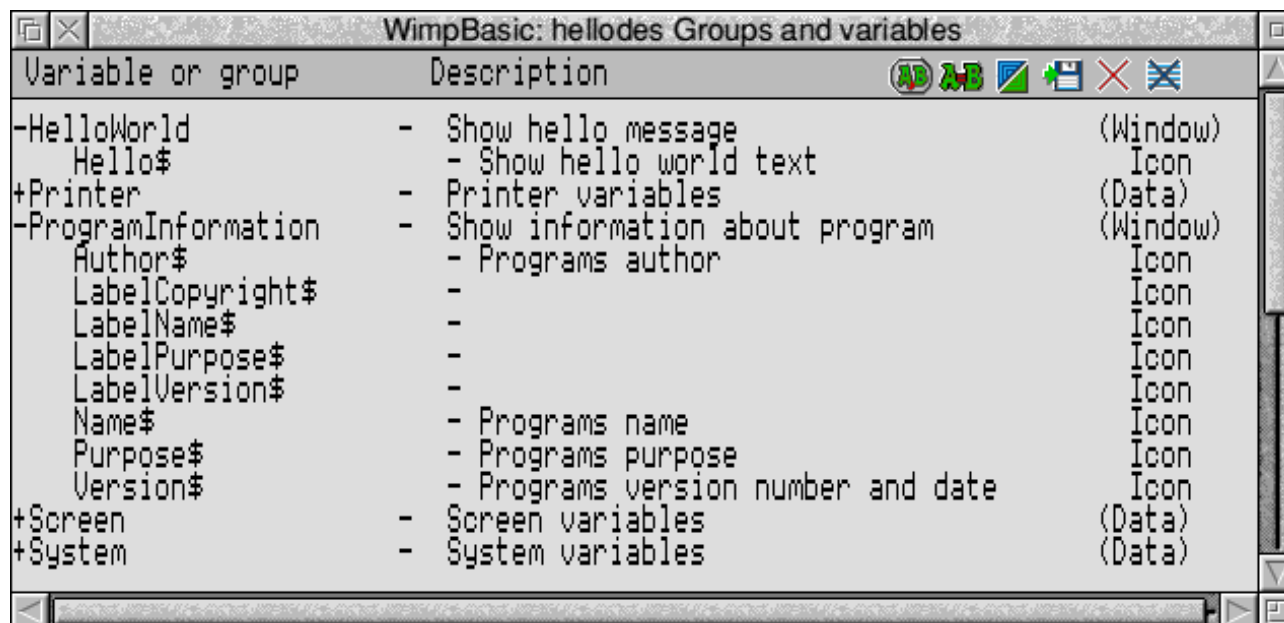
If a shortcut is defined WimpBasic executes the appropriate action by calling the click routine for that icon (or background). If no icon or background uses the key as a shortcut, WimpBasic looks at the menu attached to the window. If it uses the shortcut then the appropriate procedure is called. If the menu attached to the window doesn't use the shortcut, WimpBasic looks at all the icons in the window to see if any of them has a menu attached to it which uses the shortcut. If there is such an icon, the appropriate procedure is called. If no procedure is found the key is passed back to the Wimp in case some other programs uses it.

Variables viewer



This is opened by clicking on the button shown in the Main Window or by selecting Variables from the Main Window menu.

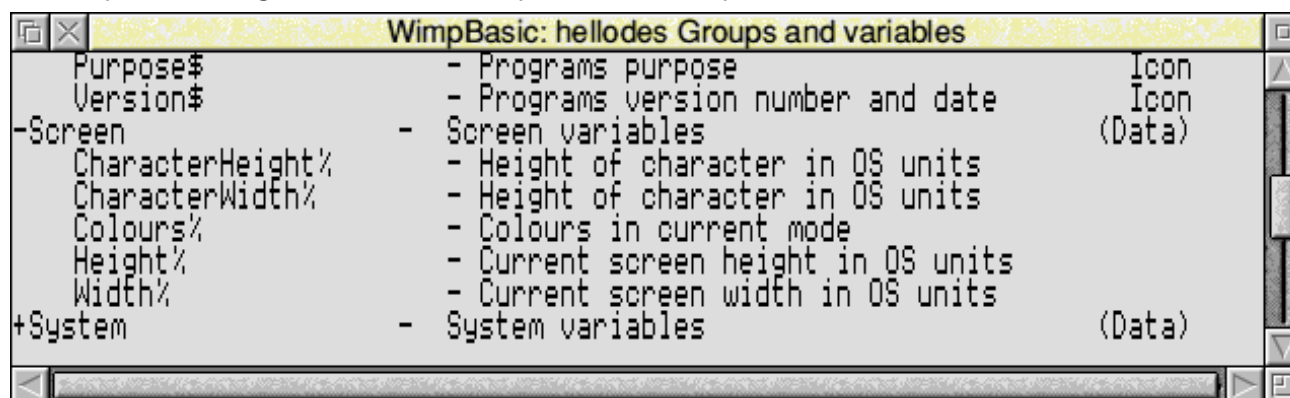
The Variables viewer lists all global variables in the design together with their description. The Printer, Screen and System variables are provided by WimpBasic and cannot be edited. They provide information about the current printer driver and screen mode.



For more information on the Printer variables refer to the Programmers Reference Manual.

Most users will not need to access these variables unless writing low level redraw code or printer code. Each Group has either a Plus (+) sign or a Minus (-) sign in front of the name. Clicking on the Plus sign *unfolds* the group to show all the variables in that group. Clicking on the Minus sign *folds* the variables back up so that just the Group name is visible. this makes it much easier to concentrate on a set of variables.

For example, unfolding 'Screen' in the example above would produce;



About WimpBasic variables

This is a brief explanation of the way WimpBasic handles variables.

In WimpBasic global variables belong to a group. This makes the code easier to read and more structured. When WimpBasic starts up, you will see three groups already in the Variables viewer.

- The *System* group which contains some information about the system such as the program path.
- The *Screen* group which contains information about the screen mode
- The *Printer* group which contains information about printing. (For more information on the variables refer to the RISC OS PRMs).

These groups cannot be edited and are provided for the experienced programmer who wishes to write low level code for window redrawing and print handling. Most users can just ignore them.

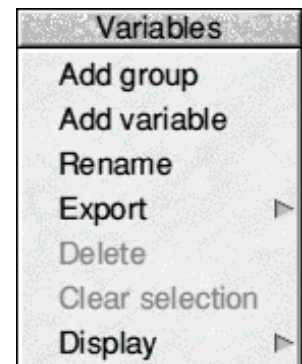
When you design a window, *it becomes a group in the variables section*. You can then attach variables to that window. These variables will only exist when the window has been created. Closing a window removes the variables. Using **Hide** to close a window (see reference section) hides the window (ie. it is removed from screen) but the variables continue to exist.

If you create an icon in a window it becomes a variable in that group and can then be referred to by that name eg

Main:Clock\$

where *Main* is the name of the window and *Clock\$* the name of the icon

Remember that all variables in WimpBasic are case insensitive.



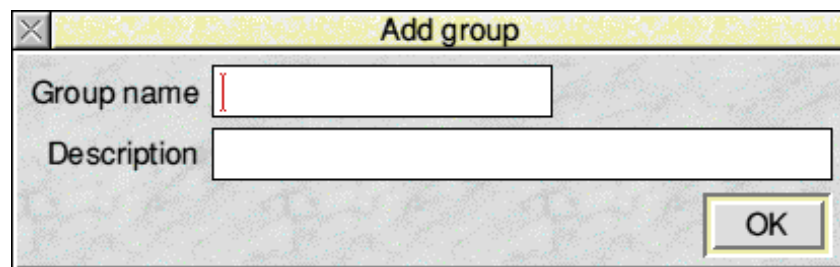
The Variables menu

This is opened by clicking MENU anywhere over the Variables window. Most of these items can also be accessed from the buttons in the Variable viewer window, and these are shown with the menu entry in the following section.



Add Group

This opens the Add group window. Enter the group name and a description. A useful convention to adopt is to use the group *Global* for variables that will be used globally throughout your program and *Constant* for variables that will have a constant value (eg. VAT rate).



Example:

Global:Destination\$="Untitled"



First create the Group, *Global* and then create the variable *Destination\$*. Now you can assign data to the variable in a ProcSetUp or similar.

Constant:TEXT=&FFF

First create the Group *Constant* and then create the variable *TEXT*. Now you can assign a value to the variable *TEXT* in a ProcSetUp or similar eg Constant:TEXT=&FFF. In this case you can now use the variable TEXT to check a filetype is for a Text file by using

IF FileType%\<\>Constant:TEXT THEN



To set up a constant for a DrawFile you would use

Constant:DrawFile=&AFF

Add variable



This opens the Add variable window. First you should select the group to which you wish to add this variable. Then enter the variable name and description. Now click on 'OK' to create the variable.



Rename

This is used to rename variables.

Export

Export is used to export variables. This can be useful if you need a similar or identical variable structure in another program.

Delete

Deletes all selected variables.

Clear selection

Untags all tagged variables and groups.

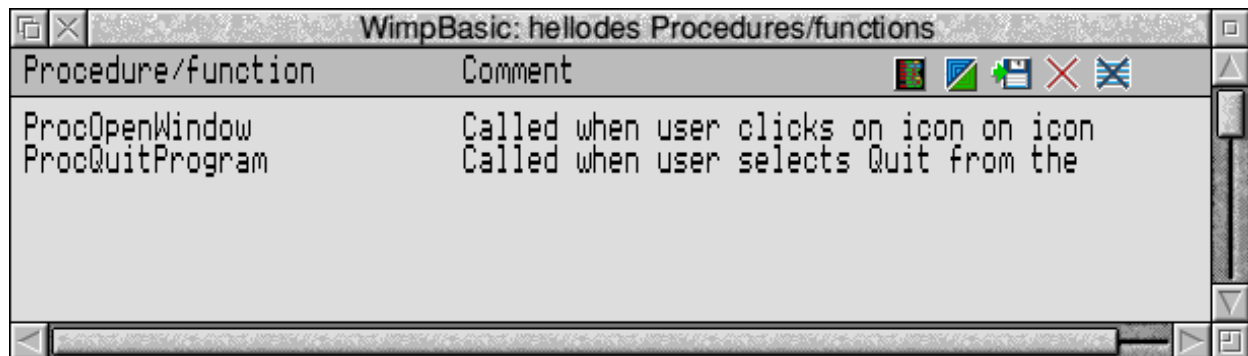
Display

Display is only available from the menu. It has a submenu which allows you to set the colour for the viewer. The Window, Menu, Code and Sprites viewers and the throwback window all use the same colour setup.



Code viewer

Click on the button shown in the Main Window or select 'Code' from the main menu to open the Code Viewer. Here you see a list of all procedures and functions in the design with their descriptions



Again you will see the similarity with the previous windows, and again the buttons duplicate the functions on the menu.



Add Code

Opens the WimpBasic Code Editor which is explained elsewhere.



Rename

Opens the rename window where you can rename the code segment.



Export

Exports the selected procedures and functions for use in other programs.



Delete

Deletes all selected procedures and functions.



Clear selection

Untags all tagged procedures and functions

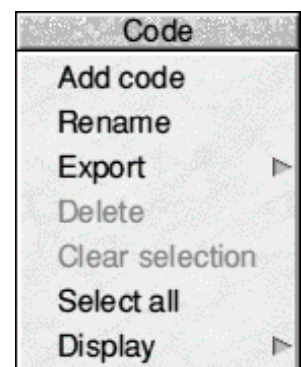
Select all

Only available from the menu. This selects all procedures. You could use this if you wanted to export all the code.

Display

Leads to a submenu from you can set the colours for the viewers.

Double-clicking on any of the code entries in the Code Viewer window will load that procedure into the Code Editor and open a window for you to edit the code.

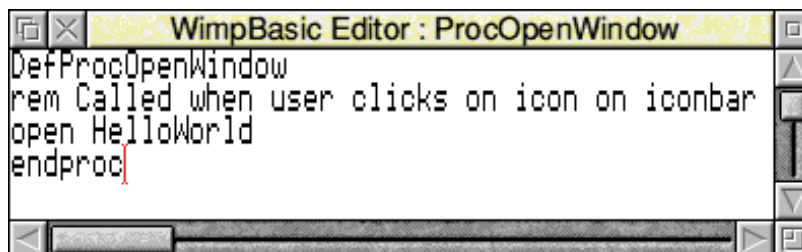


Code Editor

The WimpBasic code editor (WB Edit) is a very simple editor. It has a line length limit of about 200 characters and starts complaining if routines are longer than 100 lines. These constraints help ensure that program segments do not become over complex. If you reach the limit of 100 lines it would probably be better to split up the procedure into sub procedures

Using WB Edit

WB Edit is similar in many ways to !Edit. We assume that you are familiar with !Edit and this section describes WB Edit very briefly and concentrates on its special features. Most normal editing functions, cursor movement, selecting, copying, search and replace etc. work almost exactly as in !Edit. The most obvious difference is that WB Edit can display statements, variables etc. in different colours which makes code much easier to read.



To start a new procedure click on the left button on the Code Viewer window button bar or select 'Add code' from the menu. To edit an existing procedure double-click on it in the Code Viewer window. If you had double-clicked on ProcOpenWindow in the Hello World program the WB Edit window should appear as shown.

WB Edit menu

Save code

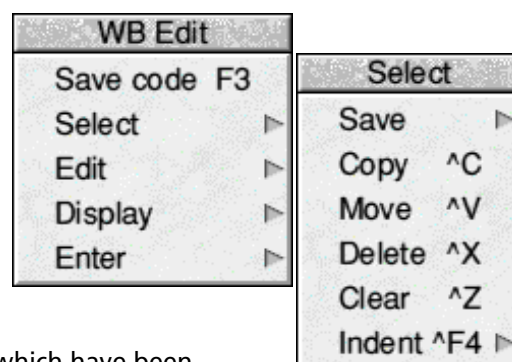
Clicking on Save code or pressing F3 will save the code. With a new procedure it will appear in the Code Viewer window with a default name of the first line of the code.

Select

This menu will be inactive unless an area is selected.

This leads to a sub-menu and performs actions on sections of code which have been 'selected'. These should all be self evident. The items on this menu all have keyboard shortcuts as shown.

Hint. To REM out a section of code, mark the relevant lines and use *Select => Indent* (Ctrl F4) with 'REM' as the indent. To remove the REMs mark the block again and use an indent of -3 to remove three characters at the start of each line.



Edit

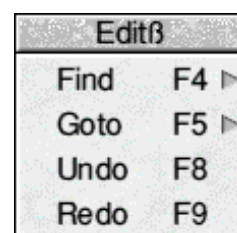
This leads to another sub-menu.

Find leads to a window which enables you to search the text. **Goto** lets you to jump to a particular line number.

Display

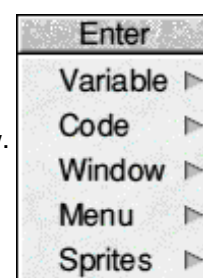
This leads to a sub-menu where you can set the editor display options.

All the items on this menu except the first lead to a colour selector menu from which you set the colour of the item. The first item sets the line spacing, the number of pixels between each line in the display. Normally this would be '0', but you can increase it to space out the lines if you wish



Enter

This leads to another sub menu with five further menus. By selecting from these you can directly enter the names of various variables and components directly into the editor window. Not only does this save typing it also avoids errors.



Using other editors

You may prefer to use your own favourite editor, such as !Zap, StrongED or even !Edit, to write and modify these files. This can very easily be done.

Write the code in your preferred editor and then open a blank WB Edit window and drag the text file to it. The

code will appear there and can be Saved as part of the program in the usual way.

To edit an existing procedure, select the one you want to edit in the Code Viewer window and use 'Export' from the menu or click on the equivalent icon on the Code Viewer button bar. This will open a SaveAs window from which you can save the selected item(s) as a plain text file

WimpBasic interactive help

WB Edit has a built in Interactive Help system.

When you are editing code, if you place the caret in any keyword or Basic function and press **F1** a window will open giving information about that keyword, it's correct syntax, parameter, etc.

This system works in a similar manner to HTML, with a link back to an index from which you can examine any keyword.

Note that the Interactive Help system is still fairly crude. This is one area we hope to improve in a later release of WimpBasic.

Compiling & Debugging

Compiling a WimpBasic application is achieved by saving the application from the Main Window menu or using SHIFT F3. As compilation progresses a window informs you as each stage is completed.

Any errors generated are shown in a window and show the line number and a suggestion as to the cause of the error.

A full list of Compiler errors can be found in Appendix A.

A list of RunTime errors is provided in Appendix B. These errors should only occur when running your application.

WimpBasic provides a 'throwback' facility for correcting errors. Clicking with SELECT, on the error in the errors window opens the editor with the cursor at the line that caused the error. You can correct the error, save the code and attempt to recompile.

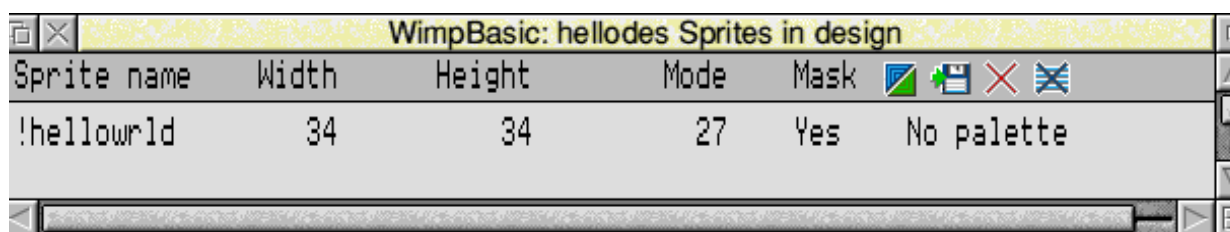
If you click on the error with ADJUST the line is removed from the list. This allows you to correct an error and remove it from the list.

Note: You can use the appropriate configure option to ensure that the application design is always saved prior to compilation. It is best to do this to ensure that you will always have the latest application's design file saved to disc.

Sprites viewer



Click on this in the Main Window or select 'Sprites' from the Main Window menu to open the Sprite Window.



This window shows details of all the sprites used by the program.

The Sprites Viewer menu is similar to those previously described and from it you can Delete and Export selected sprites.

Unlike the other viewer windows there is no means of editing sprites from within WimpBasic. Sprites would normally be created in !Paint or some other bitmap editing program. However, !Paint is perfectly adequate for creating the icons and small sprites used in programs.

A good selection of sprites is supplied with WimpBasic and these can be seen by opening the Sprites Library directory from the iconbar menu.



Raw handlers

Click on this in the Main Window or select 'Raw handlers' from the Main Window menu to open the Raw Event Handler window.

When you select this item a window opens where you can set up *Raw Event Handlers*.

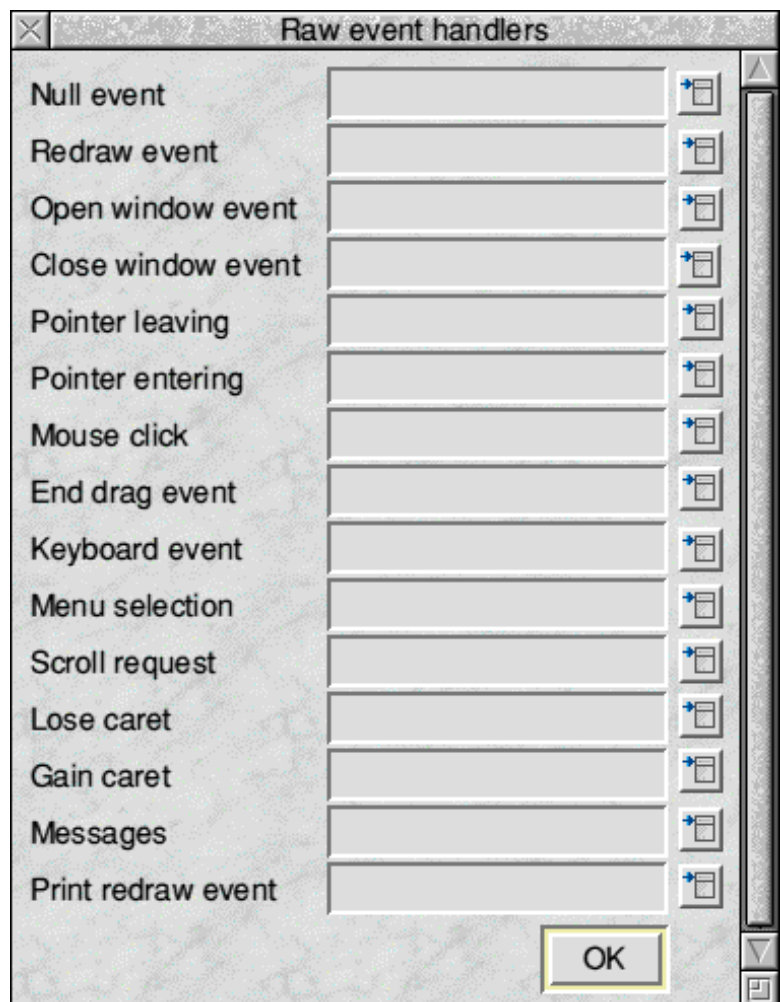
When your program receives an Event the Raw handler is called first, before any other procedure.

It's not likely that you will need to employ Events, and it is beyond the scope of this manual to describe their use. If you want to know more about them you are advised to read the relevant section on the Programmer's Reference Manual. Using Raw Events in a wimp program is normally discouraged but it is possible that in some cases they will be needed, so if they are Wimp Basic can use them.

The syntax for all raw handler routines is the same:

```
DefFnRawHandler(EventNumber%,PollBlock%)
REM Return True if you handle the event and other
REM routines should NOT receive the event. Return False
REM if other routines should receive the event =Handled
```

To write a Raw Event procedure click on the required event to open the editor. It should be noted that for window and printer redraw, the EventNumber% parameter is unused (usually 0) and the pollblock% is actually the block returned by Wimp_GetRectangle.



Configuration

Select Configure from the iconbar menu or from the Main Window menu to open the Configure window. See the note at the end on the different actions of these two methods.

Close editor window after save

If selected this automatically closes the editor window when you select Save code from the menu.

Double precision floating point

This selects more precise, but slower, floating point maths. This provides the same precision as Basic64, that is, you can have a larger number of digits to handle larger (or smaller) numbers. BASIC64 is provided in the Modules directory of your !System.

Save design before compiling

This will save the design every time you compile. This can be useful as it ensures that you always have an up to date version of the design file in case of a computer crash.

Blinking caret in editor

This makes the caret in the editor blink to aid location.

Text colouring in editor

This turns keyword colouring in the editor on or off. It can be used instead of resetting the colours of individual items in the editor.

Guidelines in window designer

If selected then guidelines are available to assist in object placement.

Fancy fonts in help

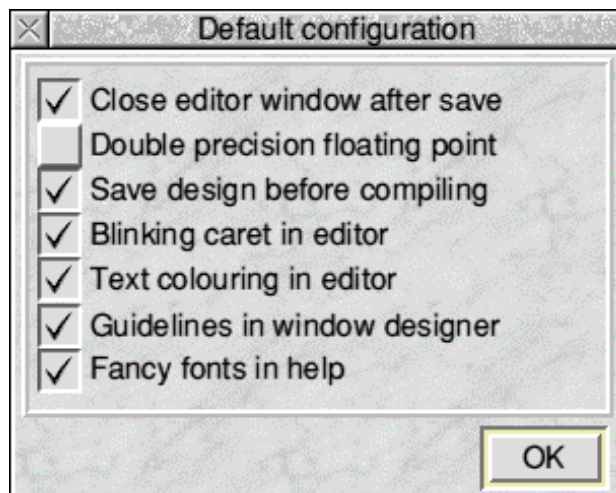
If selected then outline fonts are used in the Interactive Help System.

Project vs Default configuration

If you open the Configure window from the iconbar menu you are setting the default options that will be used by all new projects that you start.

If you open the Configure window from the Main Window menu you are only setting the options for that particular project. This allows you to have different settings for different projects.

The configuration in a saved project overrides the default configuration, so after loading a project your default configuration may be changed.



Tutorials

This section describes the example programs and takes you through the creation of some of these step by step. There follows a list of the tutorial programs and what they do. This may help you to select an application to study that is of interest to you.

The construction of the first tutorial, HelloWorld, is described step by step, with illustrations. The following tutorials are less detailed as repeating every step in this way would soon become tedious. Only new concepts will be explained in detail.

HelloWorld Demonstrates what is needed for a minimal WimpBasic program and how to open a window.

Reverse How to read information from icons, store data in icons and how to attach code to an icon.

Counter How to attach an event handler to a window and demonstrates the use of array windows. Array windows are like array variables and are used if you need several identical windows.

Calculator A small calculator application.

Calc2 Another calculator application based on the previous one. Adds the ability to enter numbers from the keyboard.

The following are provided as commented design files in the Tutorial directory. You can load them in to WimpBasic and view the code routines.

Events Provides examples of handling Events in WimpBasic.

InOutDes Simple program that shows how to transfer data into and out of a WimpBasic program.

Clock Simple clock application using characters.

DigClock More sophisticated clock application using sprites to provide LCD type characters.

FrameWork1 Provides a framework for loading a file and doing some processing on it. This program analyses a text file and shows how many times each character appears.

Framework2 Provides a framework for loading a file, doing some processing to the file and then saving the file.

Graphics Shows how to draw some simple graphics into a window. Note the absence of any window handling or clipping routines.

Numbers A simple number guessing game.

Rock The old Scissors, Paper, Rock game.

Slid Shows how to implement a simple slider bar.

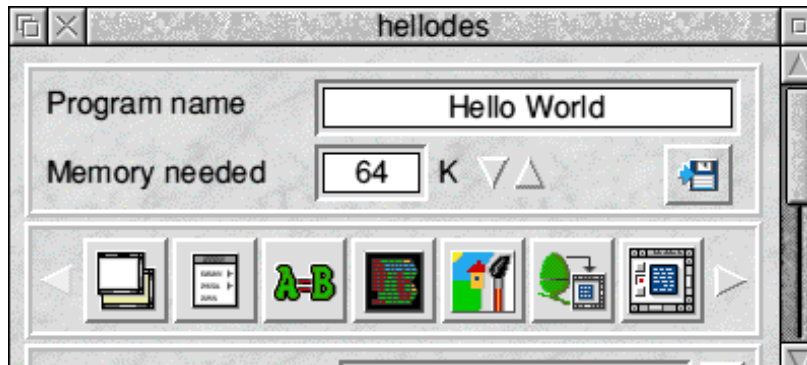
Slid2 Shows how to implement a more complex slider bar including a numeric indicator.

ToDo This is quite a useful little application and provides a ToDo list application that enables you to prioritise your jobs.

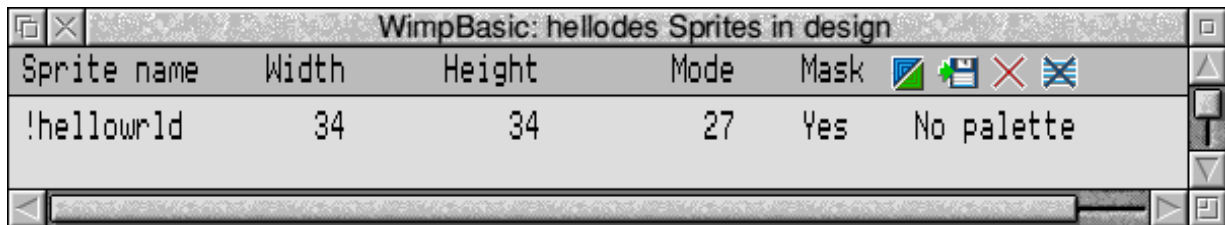
Warn Shows how to display a Warning box which disappears after a set time. This uses the null event handler

HelloWorld

1. Start WimpBasic and open the Main Window by clicking on the icon.
2. Enter *Hello World* into the 'Program name' field.
3. Enter 64 in the 'Memory needed' box



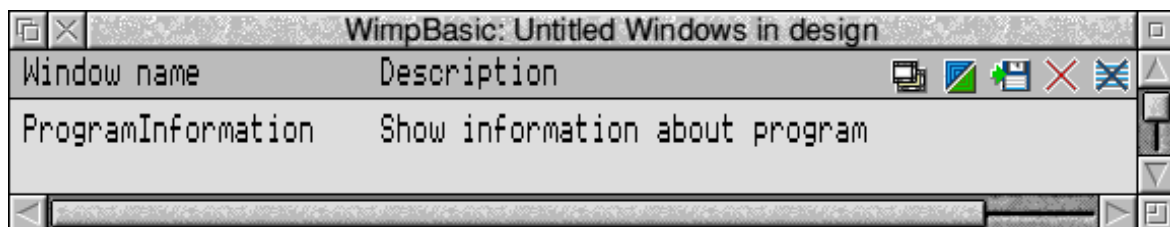
4. Open the Sprites viewer by clicking on this icon
5. Open the *Tutorials.Sprites* directory on your hard disc.
6. Drag the sprite file *HelloSpr* into the Sprites viewer window.
7. You should now see the sprite '!helloworld' displayed in the viewer.



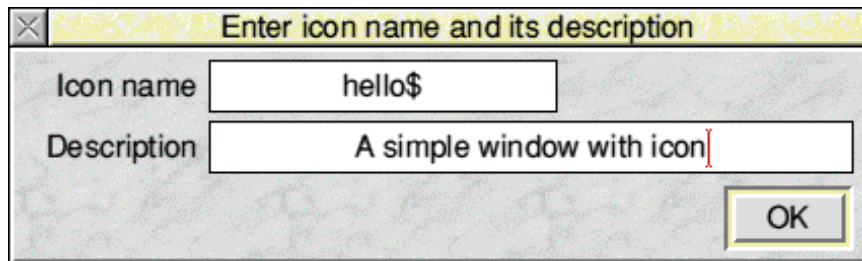
8. Close the Sprites viewer window.
9. Open the Sprites menu by clicking on the Iconbar sprite menu button in the Main window.



10. Select *Application => !helloworld* from the menu. The sprite name should now appear in the Iconbar sprite icon.
11. Open the Windows viewer by clicking on this icon.
12. Select *Library => Windows* from the iconbar menu to open the Windows directory viewer.
13. Drag the *ProgrInfo4* data file into the Windows viewer.
14. You can now close the Windows Library



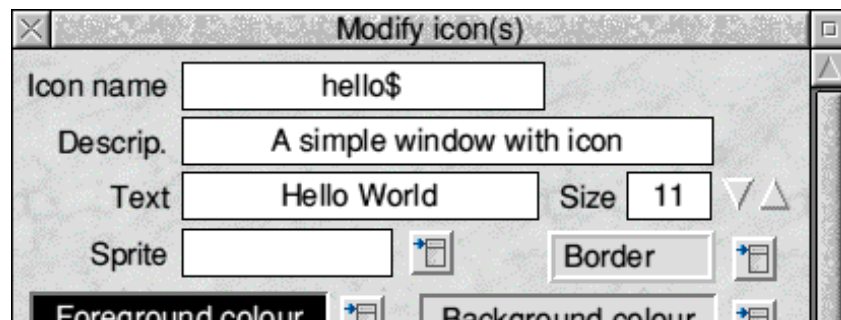
15. Create the Hello world window as follows.
 - a. Select *Add window* from the Window Viewer menu.
 - b. Click MENU over the new Window and choose *Create icon => Text*. Choose the raised edge border.
 - c. A dialogue will open for you to enter the icon name; enter *hello\$* and add a description of your own before clicking on *OK*.



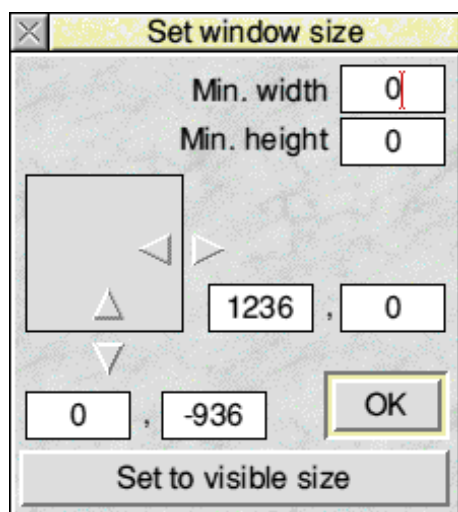
- d. The icon will appear in the window. Click MENU over the icon and select *Amend icon* to open the Modify icon window.

Note: No icons must be selected when doing the above as this opens the dialogue in a different way.

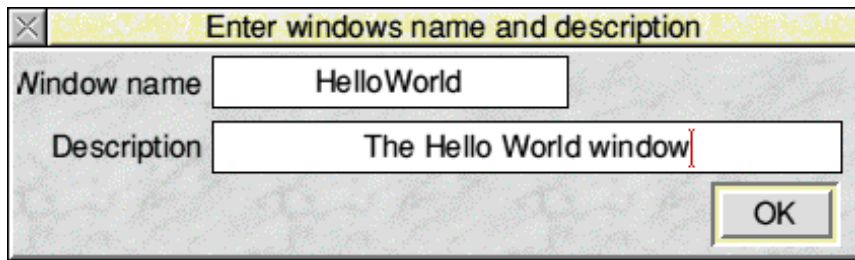
- e. Click on the 'Text' writable icon, delete <Untitled> and enter *Hello World* then click on OK.



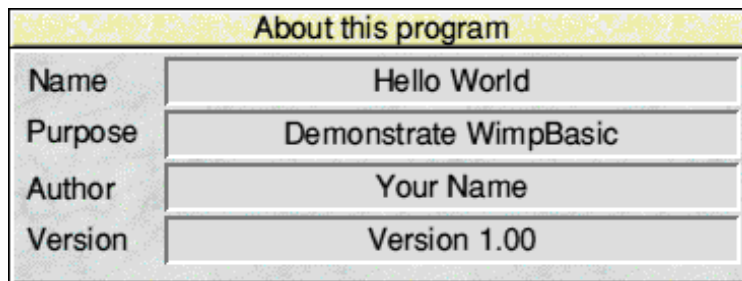
- f. Drag the icon so it is about 2 cm. from the top and 2 cm. from the left and adjust the window by dragging its adjust size icon so there is about 2 cm. all around the 'Hello World' icon.
- g. Click MENU over the window again and select Window size from the menu to open the 'Set window size' window.
- h. Click on Set to visible size and then OK
- i. Select Window data from the Window Viewer menu. Enter WimpBasic Tutorial into the 'Title' icon. Deselect the H scroll, V scroll and Adjust buttons and click on OK. You now have a window, with no scroll bars, containing an icon



- j. Select *Save window* from the Window Viewer menu. A dialogue will appear for you to enter the window name. Call it *HelloWorld* and add a description. Click on *OK* and the window name and description will appear in the Windows viewer. At this stage the window data is not saved to disc, it is just stored within WimpBasic



16. Double click on *ProgramInformation* in the Windows Viewer to edit this window.
17. Change the Name field to *Hello World* by clicking MENU over the field and selecting *Amend icon #n* (*n* being the number of the icon). This opens the Modify icon window where you can enter *Hello World* into the Text icon and click on *OK*.
18. Do the same for *Purpose* (enter *Demonstrate WimpBasic*), *Author* (enter your name) and *Version* (enter *Version 1.00*).
19. Select *Save window* from the Window Designer menu and click *OK*.



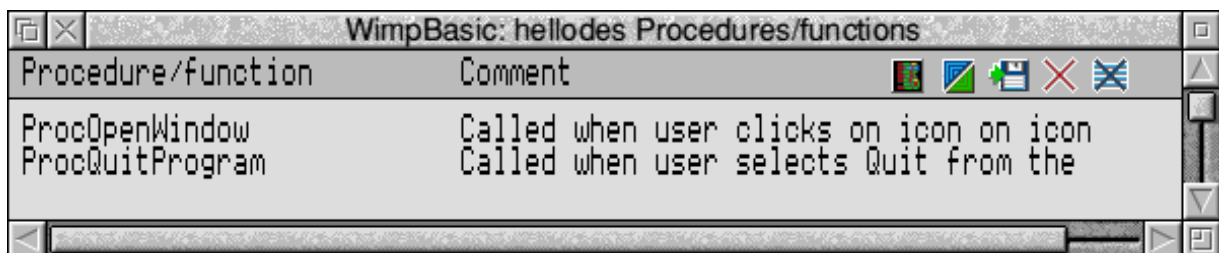
20. Close the Windows viewer.
21. Open the Code Viewer by clicking on this icon.
22. Select *Add code* from the Code viewer menu. A blank editor window will open.
23. Enter this procedure:

```
DefProcQuitProgram
Rem Called when user selects Quit from menu
Quit
EndProc
```

24. Save the code by selecting *Save code* from the editor menu. The Procedure name is extracted from the name after **DefProc** and the comment is extracted from the text after the **Rem** statement.
25. Select *Add code* from the Code Viewer menu and enter the procedure:

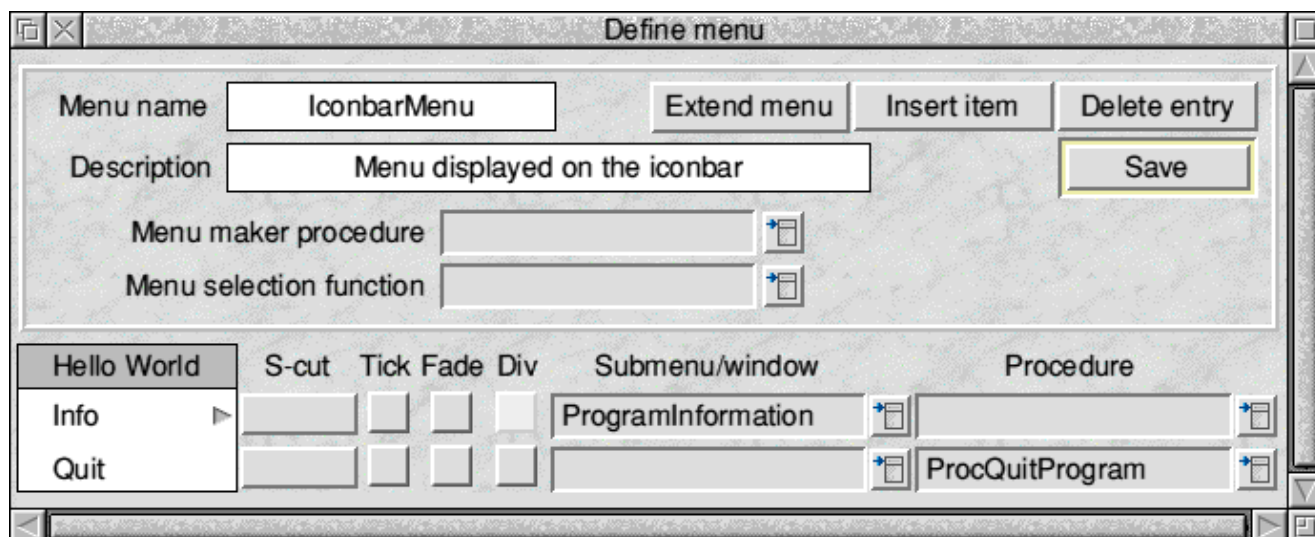
```
DefProcOpenWindow
Rem Called when user clicks on iconbar icon
Open HelloWorld
EndProc
```

26. Save the code by selecting *Save code* from the editor menu. The Code Viewer window should now show the two new procedures.

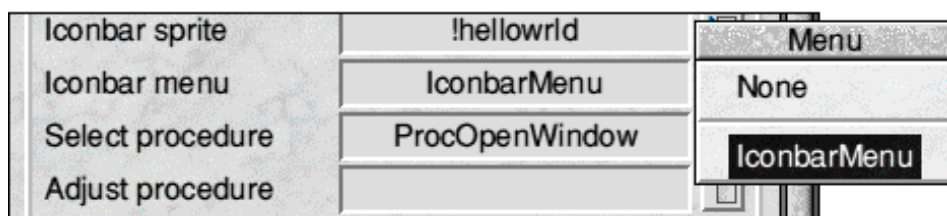


27. Close the Code viewer.
28. Open the Menus Viewer by clicking on this icon.

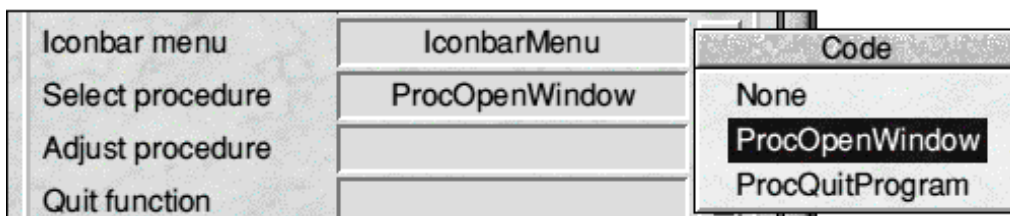
29. Select *Add menu* from the *Menus Viewer* menu.
 30. In the *Define Menu* window enter *IconbarMenu* as the Menu name. Enter something like *Menu displayed on the icon bar* in *Description*. Enter *Hello World* into the grey Menu title area. The first menu entry below *Hello World* needs to be *Info*. Click on the topmost part of the white menu area and enter *Info*. Click on *Extend menu* to add another line to the menu and enter a second menu item, *Quit*.
 31. Open the *Submenu/window* menu for the *Info* entry and select *Windows => ProgramInformation*. This will appear in the box and attaches the *ProgramInformation* window to the *Info* menu item.
 32. Open the *Procedure* menu for the *Quit* entry and select *PROCQuitProgram*. This attaches the procedure to the menu item *Quit* and the procedure is called when the user clicks on *Quit*. The
33. Click on the *Save* button to save the menu to the *Menus Viewer*. The *Define Menu* window should now look like this.



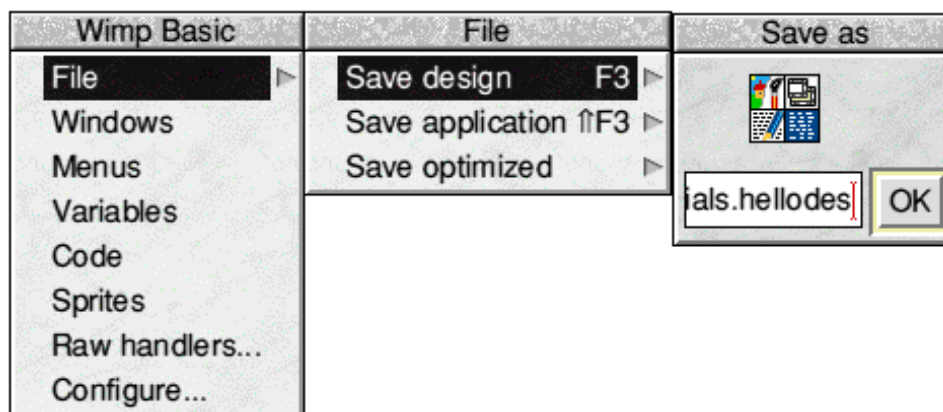
34. Close the *Menus viewer*.
35. Open the *Iconbar* menu in the *Main* window and select *IconbarMenu*. This attaches the menu you have created to the iconbar icon of your application.



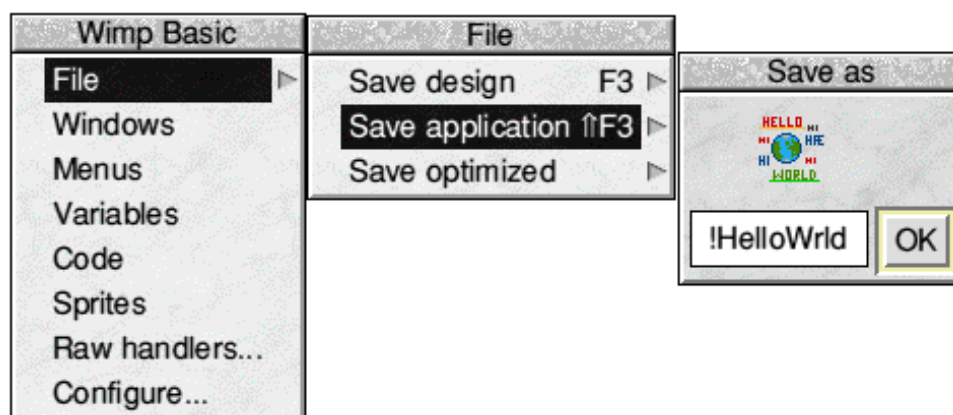
36. Open the *Select procedure* menu in the *Main* window and select *ProcOpenWindow*. This tells the application to call this procedure when the user clicks on the iconbar icon with *SELECT*.



37. Open the Main menu and save the design file as *HelloDes* using the SaveAs window found by following *File => Save design*.



38. Save the application as *!HelloWrld* using the SavaAs window found by following *File => Save application*. This compiles your application and saves it to disc.



If you get an error during compilation see the notes on Debugging.

Your Hello World application is now ready and can be run by double clicking on the icon in the filer. Click on the icon on the icon bar to open the window and try out the icon bar menu.

Okay, so it's not very exciting but it is your first WimpBasic program and if you try to create it again you will do it very much faster this time.

Reverse

1. Start up WimpBasic and open the Main Window.
2. Enter *Reverse* into *Program name* field.
3. Enter *64* in *Memory needed*.
4. Open the Sprites Viewer.
5. Open the *Tutorials.Sprites* directory on your disc.
6. Drag the sprite *ReverseSpr* into the Sprites Viewer window.
7. Close the Sprites Viewer window.
8. Clicking on the *Iconbar sprite* menu button in the Main Window.
9. Select *Application => !Reverse* from the menu.
10. Open the Code Viewer.
11. Select *Add code* from the Code Viewer menu to open a blank Editor window.
12. Enter the following procedures:

```

DefProcQuitProgram
Quit
EndProc

DefProcOpenWindow
Open ReverseWindow
EndProc

DefProcReverseString
local value$,new$,counter%
value$=ReverseWindow:Source$
new$=""
For counter%=Len(value$) TO 1 Step -1
new$=new$+Mid$(value$,counter%,1)
Next counter%
ReverseWindow:Destination$=new$
EndProc

```

You may want to add a *Rem* statement on the lines following the *DefProcs*. The *Rem* statement is extracted and gives information in the Code Viewer window.

13. Save the code by selecting *Save code* from the Editor Menu. Note that you don't have to save each procedure separately as WimpBasic extracts each one and puts it into the Procedure list individually.
14. Close the Code Viewer.
15. Open the Windows Viewer.
16. Select *Library => Windows* from the iconbar menu to open the Windows Library directory.
17. Drag the *ProgrInfo4* window data file into the Windows Viewer.
18. You can now close the Windows filer window.
19. Select *Add window* from the Windows Viewer menu.
20. Add a Writable icon to the window using *Create icon => Writable*. Name it *Source\$*. Use *Amend #n* to ensure the Button type is *Writable* and set Size to 20.
Note: You will need to alter the size of the writable icon to display the full 20 characters. This depends upon the desktop font that you are using. To alter the size drag one of the corners with ADJUST.
21. Add another Writable icon, called *Destination\$*. Use *Amend #n* to change the Button type to *Ignore*, set Background colour to 1 and set Size to 20. Again you will need to change the size of the icon by dragging a corner with ADJUST.
22. Finally create a Text icon called *Reverse\$* and, using *Amend #n*, make it Button type *Click*. Change the Text field to *Reverse string*.
23. Before clicking on the OK button, click on the menu icon for *Procedure* and select *PROCReverseString*. This is the procedure that will be called when you click on the button.
24. Now click on OK.
Note: You may need to alter the size of the *Reverse string* icon to display the full text. This depends upon the desktop font that you are using. To alter the size drag one of the corners with ADJUST.
25. Save the window as *ReverseWindow*.
26. Double click on *ProgramInformation* to edit the info window.
27. Change the *Name*, *Purpose*, *Author* and *Version* fields to *Reverse*, *Demonstrate WimpBasic*, *Your name* and *1.00* by selecting each field in turn and using *Amend Selected*.
28. Select *Save window* from the menu.
29. Close the Windows viewer.
30. Open the Menus Viewer window.
31. Select *Add menu* from the Menu Viewer menu to open the Menu Designer

32. Enter *IconbarMenu* as Menu name, *Any text* as Description, *Reverse* in the grey Menu title. *Info* for first menu entry and *Quit* for the second entry after clicking on the *Extend Menu* button to add the second blank item to the menu.
33. Click on the *Submenu/window* menu icon for the *Info* menu item and select *Windows => ProgramInformation* to link it to the window.
34. Open the *Procedure* menu for the *Quit* menu item and select *ProcQuitProgram*.
35. Click on *Save* button and then close the Menus Viewer.
36. Click on the *Iconbar* menu icon in the Main Window and select *IconbarMenu* from the menu.
37. Click on the *Select procedure* menu icon in the Main window and select *ProcOpenWindow* from the menu.
38. Save the design file as *ReverseDes*.
39. Save the application as *!Reverse*

Note: If you get an error during compilation see the notes on Debugging.

The **Reverse** application is now ready. Run it and type a string into Text to change box and click on the Reverse string button. The text is reversed.

The supplied **Reverse** design file also has a *Copy* button in the window.

Counter

For the Counter program you can re use some of your earlier work. Create a directory called Counter, if you do not already have one. Copy the file HelloDes to the Counter directory and rename it CounterDes.

1. Start WimpBasic, load the *CounterDes* file and open the Main Window
2. Open the Windows Viewer and edit the *ProgramInformation* window's *Name* field to *Counter*. Save the window using the menu.
3. Using *Amend #n* edit the *HelloWorld* window to contain 0 instead of *Hello world*. Also change the icon name to *Counter\$*.
4. Change the window title to *Counter* with the *Window data* dialogue
5. Save the window as *CounterWindow(10)*. This allows the application to open to 10 copies of this window at a time.
6. Delete the old *HelloWorld* window and Close the Windows Viewer.
7. Change the program name in the Main Window to *Counter Demo*.
8. Open the Menus Viewer and edit the *IconbarMenu* so the menu title in the grey box is *Counter*.
9. Save the menu and close the Menus Viewer.
10. Open the Sprites Viewer and delete the *!helloworld* sprite.
11. Open the *Sprites* directory in the *Tutorial* directory and drag the *CounterSpr* file into the Sprites Viewer. Now close the Sprites viewer.
12. Change the Iconbar sprite in the Main Window to *!counter*.
13. Open the Code Viewer.
14. Use *Add code* to add this procedure (and don't forget to save it):

```
DefProcNullEvent
Local value%
value%=Val(CounterWindow(WindowIndex):Counter$)
value%=value%+1
CounterWindow(WindowIndex):Counter$=Str$(value%)
EndProc
```

15. Rewrite the *ProcOpenWindow* procedure like this:

```
DefProcOpenWindow
local counter%
For counter%=0 to 9
If Not IsOpen(CounterWindow(counter%)) then
Open CounterWindow(counter%)
counter%=99
EndIf
Next counter%
EndProc
```

Close the Code viewer window.

16. Open the Windows Viewer. Double click on *Counterwindow(10)*, Click MENU over the window and select *Event handlers...*
17. In the Window Event Handlers window open the *Null event* procedure menu and select *ProcNullEvent*. Now click on OK.
18. Save the window.
19. Close the window and the Windows Viewer.
20. Save the design file using F3.
21. Save the application as *!Counter*.

If you get an error during compilation see the notes on Debugging.

Note: The name of the application must be the same as the name of the sprite used for the filer sprite, otherwise no icon will appear in the filer. The sprite must be present in the Sprite Viewer or be provided in system sprites. Normally you would use the same sprite for the iconbar sprite.

The counter program is now ready. Load the program and click on the icon bar icon to open a window that counts up in steps of one. You can open up to ten windows and each one starts from 0. You have created your first multi document application in WimpBasic.

Calculator

Begin by copying the *HelloWorld* program into a directory named *Calc1* and rename it *CalcDes*. Double click on *CalcDes* to load WimpBasic with the file and modify the design as before to have *Calculator* instead of *HelloWorld* as the program name. You should use the sprite *CalcSpr* instead of the *!helloworld* sprite. Items that need to be changed are:

Windows - HelloWorld (see 2 below) and ProgramInformation; Menu - IconBarMenu (the menu title); Main window - Iconbar sprite to !Calc1.

You should also delete the procedure *ProcOpenWindow*. When you have done all that, start on the list below.

1. Enter the following code using Add code from the Code Viewer:

```
DefProcDigit0
ProcAddNumber(0)
EndProc

DefProcDigit1
ProcAddNumber(1)
EndProc

DefProcDigit2
ProcAddNumber(2)
EndProc

DefProcDigit3
ProcAddNumber(3)
EndProc

DefProcDigit4
ProcAddNumber(4)
EndProc

DefProcDigit5
ProcAddNumber(5)
EndProc

DefProcDigit6
ProcAddNumber(6)
EndProc

DefProcDigit7
ProcAddNumber(7)
EndProc

DefProcDigit8
ProcAddNumber(8)
EndProc

DefProcDigit9
ProcAddNumber(9)
EndProc

DefProcAddNumber(Number%)
Local n%
If Calculator:ClearNext% Then
    Calculator:Display$=""
    Calculator:ClearNext%=False
EndIf
n%=Val(Calculator:Display$)
If n%<99999999 Then
    n%=n%*10
    n%=n%+Number%
    Calculator:Display$=Str$(n%)
EndIf
EndProc
```

```

DefProcClear
Calculator:Display$="0"
Calculator:Action$=""
Calculator:Value%=0
Calculator:ClearNext%=False
EndProc

DefProcClearEntry
Calculator:Display$="0"
EndProc

DefProcDivide
ProcAction("/")
EndProc

DefProcEqual
ProcAction("=")
EndProc

DefProcMinus
ProcAction("-")
EndProc

DefProcMultiply
ProcAction("*")
EndProc

DefProcPlus
ProcAction("+")
EndProc

DefProcInitCalcWindow
If Global:Initialise% Then
    Calculator:Action$=""
    Calculator:Value%=0
    Calculator:ClearNext%=True
    Calculator:Display$=""
    Global:Initialise%=false
Endif
EndProc

DefProcOpenWindow
Open Calculator
ProcInitCalcWindow
EndProc

DefProcStartupProcedure
Global:Initialise%=True
EndProc

DefProcAction(Action$)
Local n%
n%=Val(Calculator:Display$)
Case Calculator:Action$ Of
When "+"
    Calculator:Value%=Calculator:Value%+n%
When "-"
    Calculator:Value%=Calculator:Value%-n%
When "*"
    Calculator:Value%=Calculator:Value%*n%
When "/"
    Calculator:Value%=Calculator:Value%/n%
Otherwise
    Calculator:Value%=n%
EndCase
Calculator:ClearNext%=True
If Calculator:Action$\<>" " Then
    Calculator:Display$=Str$(Calculator:Value%)
EndIf

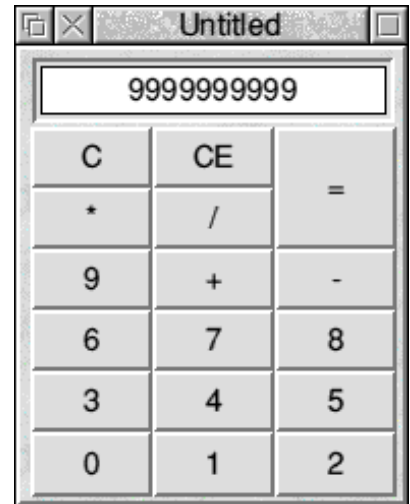
```

```

Calculator:Action$=Action$
EndProc

```

2. Delete the *Hello world* window and create a new window that looks like a calculator. Use *Create icon => Normal* to create button icons for 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, =, C, and CE, all with button type *Click*. The icons should be named, respectively, *Digit0\$, Digit1\$, Digit2\$, Digit3\$, Digit4\$, Digit5\$, Digit6\$, Digit7\$, Digit8\$, Digit9\$, Plus\$, Minus\$, Multiply\$, Divide\$, Equal\$, C\$* and *CE\$*. The window from the finished Tutorial calculator is shown to give you some idea of the layout but you can use a different design.
3. Create a Writable display icon called *Display\$* with Button type *Ignore*. You must also set the Size to 10 and enter the digit 9 ten times. This ensures that the writable icon will hold the required data. The display will be cleared in the *ProcnitCalcWindow* procedure.
4. Attach the relevant procedures to the icons using *Amend*. It should be obvious which one should be connected to which icon. eg icon *Digit0\$* has *ProcDigit0* attached to it.
5. Use *Window size* to adjust the window and Window data to remove the scroll bars. Save the window as *Calculator*.
6. Attach *ProcStartUpProcedure* to *Startup procedure* in the Main Window
7. Go to the Variables Viewer, choose *Add group* and enter *Global* as the Group name.
8. Go to *Add variable*, select *Global* as the Group and enter *Initialise%* as the variable. This creates *Global:Initialise%* which is used in *ProcStartUpProcedure*.
9. Go to *Add variable*, select *Calculator* as the Group and enter *Action\$* as the variable. Repeat this to create *ClearNext%* and *Value%*.
10. Save the design using F3.
11. Save the application as *!Calc1* using SHIFT F3.



Note: If you get an error during compilation see the notes on Debugging.

Note: The name of the application must be the same as the name of the sprite used for the filer sprite or no icon will appear in the filer. The sprite must also be present in the Sprite Viewer or be provided in the system sprites. In most cases you would use the same sprite for the iconbar sprite. By convention the sprite should be 34 by 34 pixels in Mode 27.

The calculator is now ready. Run the application and click on the buttons to perform calculations.

Calculator 2

Copy the design file for *Calculator* as *Calc2Des* and copy it into a directory called *Calc2*.

1. Enter these procedures using *Add code* from the Code Viewer menu:

```

DefProcClaimInputFocus
InputFocus Calculator
EndProc

DefFnKeyHandler(Key%)
Local used%,char%
used%=False
If Key%>=Asc("0") And Key%<=Asc("9") Then
    char%=Key%-Asc("0")
    ProcAddNumber(char%)
    used%=True
Else
    Case Key% Of
        When 8,127
            Rem Delete and backspace
            Calculator:Display$=Left$(Calculator:Display$)
used%=True
        When Asc("+")
            ProcPlus
            used%=True
        When Asc("-")
            ProcMinus
            used%=True
        When Asc("/")
            ProcDivide
            used%=True
        When Asc("*")
            ProcMultiply
            used%=True
        When 13
            ProcEqual
            used%=True
    EndCase
EndIf
=used%

```

2. Open the Window Viewer, double click on *Calculator*, bring up the menu and click on *Window data....* Ensure the button type is *Click*. Attach the *ProcClaimInputFocus* procedure to the *Procedure* field and click on OK.
3. Click MENU over the Calculator window and select *Event handlers...* Attach the function *FnKeyHandler* to the *Keyboard Event* field and click on OK. Close the Calculator window to save the changes.
4. Open the Sprites Viewer and use *Rename* to change the sprite from *!Calc1* to *!Calc2*.
5. Save the design.
5. Save the application as *!Calc2*.

You now have the *Calculator2* program ready. Input can be by mouse or by using the keyboard (use RETURN for equals). To use the keyboard you must gain the Input Focus by clicking on the window background. You might like to try adding a button to gain the input focus.

Note: To use the key pad you must have Num Lock on.

Here are just a few of the other products you will be able to
download from RISCOS.com free of charge
(You can make a donation to help support future product releases but this is entirely voluntary)



This Documentation is © 3QD Developments Ltd 2026
Some of the products shown above might not be available immediately.
Not all products will run on all versions of RISC OS, check the product information page for details.

Other Products list last updated 29/2/26