# BlitzOp Guide v1.0 by Paul Bowlay

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>BlitzOp Guide v1.0 by Paul Bowlay | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | August 22, 2024 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# BlitzOp Guide v1.0 by Paul Bowlay

## 1.1 CONTENTS

```
                          ***  THE BLITZOP GUIDE  ***
                            (Version 1.0 - 47 Tips)
INTRODUCTION

   Legal Information     Where you can spread the BlitzOp Guide.
   What Is This Guide?   What the BlitzOp Guide is designed for.
   You Can Help!         Got some optimizing tips of your own?
   The Author            Who I am and how to contact me.

THE OPTIMIZING TIPS

   A
B
C
-
E
F
-
-
I
-
K
L
M   Tips, Part 1.
   N
-
P
-
R
S
-
-
V
W
-
-
-   Tips, Part 2.
```

GENERAL TIPS



==========

## 1.2 LEGAL INFORMATION

>>> LEGAL INFORMATION



==========

## 1.3 PAUL BOWLAY

>>> PAUL BOWLAY

The sole author of this AmigaGuide document is Paul Bowlay. You can only contact me via E-Mail, so those of you without E-Mail access will have to rely on a friend or a public-access Internet terminal (eg: try your local library) to contact me.

My E-Mail address: cat@aic.net.au

==========

## 1.4 ABOUT THIS GUIDE

>>> WHAT IS THIS GUIDE ALL ABOUT?

This guide shows you how to re-write your Blitz2 programs so that you get a smaller executable than you normally would. Most Blitz2 commands cause

your executables to bloat up in size when you can replace such commands with sleaker ones. This guide lists all such commands and shows you how to replace them with routines that accomplish exactly the same result.

SPECIAL NOTE!
Some of the tips in this guide will only work if you have third-party libraries added to your Blitz2 system! You can tell if a tip doesn't work on your system by typing its name and pressing the Help key. If the name doesn't change colour and re-format itself, then you don't have access to it and thus can't make use of it.

==========

## 1.5 CONTRIBUTIONS

>>> YOU CAN HELP!

If you know of any optimizing tips of your own, why not consider E-Mailing them to the author for inclusion in future releases of this guide? Your name will naturally be credited to them and you'll earn the respect and gratitude of every Blitz2 programmer in existence. :)

There is only one rule for contributing a tip: it must do exactly the same thing as the command/function you're replacing, but with less executable size! I don't care if your routine is more memory-efficient or uses less system power: this guide is for reducing executable size and nothing more.

When contributing a tip, it is essential that you make your E-Mail address available to be included alongside your name. This is so programmers can contact you if they experience trouble with your tip. Thus, only submit a tip if you're willing to be asked more info about it - the author cannot always help out with a tip query if he hasn't used it himself.

Lastly, if you ever change your E-Mail address, please inform the author so he can adjust this guide accordingly.

==========

## 1.6 EXECUTABLE OPTIONS

>>> EXECUTABLE OPTIONS

In the Compiler menu of Blitz2 is an item called Compiler Options. Select it and look at the window that appears. We are interested in only two items: Runtime Error Debugger and Create Debug Info For Executable Files.

Runtime Error Debugger will add error-checking code to your executable so that it won't crash if an error occurs. If you're absolutely 100% sure that your program won't crash, then you can safely turn off this option to save on final executable size.

Create Debug Info For Executable Files is used to add debug code to your

executable for third-party software to examine. Again, if your program is fine and you don't intend to scan it with other software, then you can turn off this option to save on final executable size.

One other option may have caught your eye: Make Smallest Code. Don't be misled: this option does not have to be turned on for it to work. It is automatically turned on internally when creating an executable, so playing with it here has no affect on your final executable size.

Lastly, some of you may look at the six buffers and wonder if reducing the values will save executable size. It won't; their values get automatically changed during compilation.

==========

## 1.7 THE AMIGA LIBRARIES

>>> THE AMIGA LIBRARIES - THE BIGGEST TIP OF THEM ALL!

Okay, I'm going to tell you a little secret on how to keep your Blitz2 executables really small. Ready? Promise not to tell anyone? Here:

Use the Amiga Library Routines.

That's it. Look at your Blitz2 reference manual from pages A3-1 onwards. See all those commands that look like FindResident(name)(a1) and so on? Well, using those instead of the equivalent Blitz2 command will literally save hundreds of bytes in your executable size.

For instance, on my system, the small CLI-Only Example 1 program creates an executable of 9588 bytes. By replacing Reboot with the Amiga Library ColdReboot_ command, and Print with the Amiga Library PutStr_ command, the executable becomes a tiny 968 bytes in size! That's a massive saving of 8620 bytes (over 8K!), yet the programs do exactly the same thing!

```
EXAMPLE 1:                      EXAMPLE 2:

; All Runtime Errors On          ; All Runtime Errors Off
; Create Debug Info On           ; Create Debug Info Off
;                                ;
Print "Click LMB to reset!"      PutStr_ "Click LMB to reset!"
MouseWait                       MouseWait
Reboot                          ColdReboot_
```

The only problem with using the Amiga Libraries is that beginners won't know how to use them. They aren't described in the Blitz2 manuals (hmmm, is anything really?) and you can't just switch to them as easily as the above examples illustrate (if you do, 99% of the time you'll get a guru and your Amiga will crash). However, experiment with them as you will, and if you find something that does work, please E-Mail me for inclusion in future releases of this guide.

==========

## 1.8 VARIABLES

```
>>> VARIABLE TIPS

CONSTANTS VS VARIABLES
If you're using a numeric integer variable that never changes value during
your program, then make it a constant instead of a variable.  This means
instead of using width=640, use #width=640.  Constants save space compared
to their normal variable counterparts, but remember: they can only be used
for numbers (not strings) and integers (numbers without a decimal point).

TOGGLING A NUMERIC VARIABLE
Sometimes you may want to have a numeric variable constantly switch between
0 and 1.  Don't use If a=0 Then a=1 / If a=1 Then a=0 because it's a waste
of size.  Use this instead: a=1-a.

USELESS VARIABLES
Using variables for unnecessary work is a size-wasting idea.  For example,
don't use a variable to hold the result of something that you just want to
print - just print the result instead.  You should only ever use a variable
if you intend to reference it again later in your code; otherwise, use the
Select command for things like checking the result of a gadget hit or menu
selection.

VARIABLE SWAPPING
Use the Exchange command to swap the contents of two variables.  Don't use
a third temporary variable (temp$=one$:one$=two$:two$=temp$) to do the work
because it's a waste of size.

================================================================================
```

## 1.9 MISCELLANEOUS

```
>>> MISCELLANEOUS TIPS

ASSEMBLER
Replace as much of your code with assembler as possible, because nothing
gets smaller than pure hand-coded assembler, not even the best C compilers.

GOSUB IS YOUR FRIEND
Some people use the same routine over-and-over in their code.  This is a
waste and you should use the Gosub command to replace each routine, and
have the routine defined just once in your code.  For example, don't have
the code If AvailMem_($0)<1024 Then Print "Out of RAM!" in your program
at every place you wish to check RAM.  Instead, do a Gosub to that code so
that it becomes If AvailMem_($0)<1024 Then Print "Out of RAM!" : Return.

USELESS COMMANDS
Don't use a command that's irrelevant to your code.  For example, some
people actually use the EventWindow command when they only have one window
open!  What a waste, because it's obvious which window caused the event!

YOU ARE WHAT YOU EAT
Just as the more you eat the fatter you get, the same principle applies to
```

```
Blitz2 programs: the more commands you use, the bigger the executable gets.
Thus, think about what you're doing, and write your routines so that they
use the least amount of commands as possible.

================================================================================
```

## 1.10 UNAVAILABLE

```
>>> NO TIPS AVAILABLE!

Unfortunately, there are no tips yet for the letter you selected.  However,
the author hopes this will change, so you can help by E-Mailing him if you
have any to add.  See this section for more details.

================================================================================
```

## 1.11 'A' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "A"

--------------------------------------------------------------------------------

ACTIVATE (Paul Bowlay, cat@aic.net.au)

If you're opening a window and instantly activating it with Activate, then
you're wasting bytes.  Instead, just add the value $1000 to the window's
flags.

OLD: Window #,X,Y,Width,Height,$Flags,Title,DPen,BPen : Activate #
NEW: Window #,X,Y,Width,Height,$Flags|$1000,Title,DPen,BPen

================================================================================
```

## 1.12 'B' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "B"

--------------------------------------------------------------------------------

BEEPSCREEN (Paul Bowlay, cat@aic.net.au)

If you're using BeepScreen with screen 0 then use DisplayBeep_ instead.
Note that this tip only works for screen 0 and no others.

OLD: BeepScreen 0
NEW: DisplayBeep_0

================================================================================
```

## 1.13 'C' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "C"

------------------------------------------------------------------------------

CHANGEDIR (Paul Bowlay, cat@aic.net.au)

Use CHDir instead.

OLD: ChangeDir dir$
NEW: CHDir dir$

------------------------------------------------------------------------------

CHIPFREE (Paul Bowlay, cat@aic.net.au)

Use AvailMem_ instead.

OLD: chip=ChipFree
NEW: chip=AvailMem_($20002)

==============================================================================
```

## 1.14 'E' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "E"

------------------------------------------------------------------------------

EXEC (Paul Bowlay, cat@aic.net.au)

Use Execute_ instead.  If you don't include the run >nil: part, then your
program will halt until the DOS command ends.

OLD: Exec dos$
NEW: Execute_ "run >nil: "+dos$,0,0

==============================================================================
```

## 1.15 'F' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "F"

------------------------------------------------------------------------------

FASTFREE (Paul Bowlay, cat@aic.net.au)

Use AvailMem_ instead.

OLD: fast=FastFree
NEW: fast=AvailMem_($20004)
```

```
 ------------------------------------------------------------------------------

 FILTER OFF (Paul Bowlay, cat@aic.net.au)

 Use assembler instead.

 OLD: Filter Off
 NEW: MOVE #253,$bfe001

 ------------------------------------------------------------------------------

 FILTER ON (Paul Bowlay, cat@aic.net.au)

 Use assembler instead.

 OLD: Filter On
 NEW: MOVE #2,$bfe001

 ------------------------------------------------------------------------------

 FLOATMODE (Paul Bowlay, cat@aic.net.au)

 If you're using the Format command, then you don't need to use FloatMode
 because Format automatically invokes a floatmode of -1.

 OLD: Format "000000000" : FloatMode -1 : NPrint 9^9
 NEW: Format "000000000" : NPrint 9^9

 ------------------------------------------------------------------------------

 FORCENTSC (Paul Bowlay, cat@aic.net.au)

 Use assembler instead.

 OLD: ForceNTSC
 NEW: MOVE #0,$dff1dc

 ------------------------------------------------------------------------------

 FORCEPAL (Paul Bowlay, cat@aic.net.au)

 Use assembler instead.

 OLD: ForcePAL
 NEW: MOVE #32,$dff1dc

 ==============================================================================
```

## 1.16 'I' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "I"

 ------------------------------------------------------------------------------
```

```
IF [Tip 1] (Paul Bowlay, cat@aic.net.au)

If you're testing a numeric variable for the value 1, then don't use =1 as
part of the test.

OLD: If a=1 Then NPrint "1"
NEW: If a Then NPrint "1"

--------------------------------------------------------------------------

IF [Tip 2] (Paul Bowlay, cat@aic.net.au)

If you're testing two variables and the result will be a small block of
code with no Gotos or Gosubs or further test conditions, then use Else
instead of two Ifs.

OLD: If a=1 Then NPrint "1"
     If a=2 Then NPrint "2"

NEW: If a=1 Then NPrint "1" Else NPrint "2"

--------------------------------------------------------------------------

IF [Tip 3] (Paul Bowlay, cat@aic.net.au)

If you're testing two variables and the result will be a large block of
code, or has Gotos or Gosubs or further test conditions, then use Else
and EndIf instead of multiple Ifs.

OLD: If a=1 Then NPrint "1" : If b=1 Then Goto loop1
     If a=2 Then NPrint "2" : If b=2 Then Goto loop2

NEW: If a=1
       NPrint "1" : If b=1 Then Goto loop1
     Else
       NPrint "2" : If b=2 Then Goto loop2
     EndIf

--------------------------------------------------------------------------

IF [Tip 4] (Paul Bowlay, cat@aic.net.au)

If you're testing more than two variables, use Select and Case instead of
multiple Ifs.

OLD: If a=1 Then NPrint "1" : Goto loop1
     If a=2 Then NPrint "2" : Goto loop2
     If a=3 Then NPrint "3" : Goto loop3

NEW: Select a
       Case 1
         NPrint "1" : Goto loop1
       Case 2
         NPrint "2" : Goto loop2
       Case 3
         NPrint "3" : Goto loop3
     End Select
```

```
 ============================================================================
```

## 1.17 'K' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "K"

 ----------------------------------------------------------------------------

 KEYCODE (Paul Bowlay, cat@aic.net.au)

 Use Peek instead.

 OLD: key=KeyCode
 NEW: key=Peek($bfec00) AND $ff

 ----------------------------------------------------------------------------

 KILLFILE (Paul Bowlay, cat@aic.net.au)

 Use DeleteFile_ instead.

 OLD: KillFile filename$
 NEW: DeleteFile_ filename$

 ============================================================================
```

## 1.18 'L' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "L"

 ----------------------------------------------------------------------------

 LOADPALETTE (Paul Bowlay, cat@aic.net.au)

 The LoadScreen command can optionally load the associated palette with it,
 so don't use LoadScreen and LoadPalette for the same screen.

 OLD: LoadScreen screen#,filename$ : LoadPalette screen#,filename$
 NEW: LoadScreen screen#,filename$,palette#

 ============================================================================
```

## 1.19 'M' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "M"

 ----------------------------------------------------------------------------

 MEMFREE (Paul Bowlay, cat@aic.net.au)
```

```
 Use AvailMem_ instead.

 OLD: free=MemFree
 NEW: free=AvailMem_($0)

 ------------------------------------------------------------------------

 MENUSOFF (Paul Bowlay, cat@aic.net.au)

 If you don't intend to use menus with a window, then don't use MenusOff to
 specify this.  Instead, add the value $10000 to the window's flags.

 OLD: Window #,X,Y,Width,Height,$Flags,Title,DPen,BPen : MenusOff
 NEW: Window #,X,Y,Width,Height,$Flags|$10000,Title,DPen,BPen

 ========================================================================
```

## 1.20 'N' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "N"

 ------------------------------------------------------------------------

 NAMEFILE (Paul Bowlay, cat@aic.net.au)

 If you're using variables to specify the old/new names, then use Rename_
 instead (and note the &).  This tip does not work with literal strings.

 OLD: NameFile old$,new$
 NEW: Rename_ &old$,&new$

 ------------------------------------------------------------------------

 NPRINT (Paul Bowlay, cat@aic.net.au)

 If your executable is for CLI usage only, then use PutStr_ instead, and
 add a Chr$(10) to the end (for the carriage-return).  This tip does not
 work for anything other than CLI usage, so window usage and the like will
 get you nowhere.

 OLD: NPrint "Hello!"
 NEW: PutStr_ "Hello!"+Chr$(10)

 ========================================================================
```

## 1.21 'P' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "P"

 ------------------------------------------------------------------------
```

```
 POKE (Paul Bowlay, cat@aic.net.au)

 Use MOVE instead and swap the order of parameters.

 OLD: Poke $address,#value
 NEW: MOVE #value,$address

 ------------------------------------------------------------------------------

 PRINT (Paul Bowlay, cat@aic.net.au)

 If your executable is for CLI usage only, then use PutStr_ instead.  This
 tip does not work for anything other than CLI usage, so window usage and
 the like will get you nowhere.

 OLD: Print "Hello!"
 NEW: PutStr_ "Hello!"

 ==============================================================================
```

## 1.22 'R' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "R"

 ------------------------------------------------------------------------------

 REBOOT (Paul Bowlay, cat@aic.net.au)

 Use ColdReboot_ instead.

 OLD: Reboot
 NEW: ColdReboot_

 ------------------------------------------------------------------------------

 RENAME (Paul Bowlay, cat@aic.net.au)

 If you're using variables to specify the old/new names, then use Rename_
 instead (and note the &).  This tip does not work with literal strings.

 OLD: Rename old$,new$
 NEW: Rename_ &old$,&new$

 ------------------------------------------------------------------------------

 REPLACE$ (Paul Bowlay, cat@aic.net.au)

 Don't use Replace$ and Instr together, because Replace$ does its own check
 of the string without needing Instr.  If the string isn't found when using
 Replace$, then the string in question is left intact; but if it is found,
 then it gets replaced, thus doing what you wanted without using Instr!

 OLD: If Instr(a$,"old")<>0 Then Replace$(a$,"old","new")
 NEW: Replace$(a$,"old","new")
```

```
 ------------------------------------------------------------------------

 RUN (Paul Bowlay, cat@aic.net.au)

 Use Execute_ instead, but only if you're not using the optional stacksize
 parameter of the command.  If you don't include the run >nil: part, then
 your program will halt until the DOS command ends.

 OLD: Run filename$,args$
 NEW: Execute_ "run >nil: "+filename$+" "+args$,0,0

 ========================================================================
```

## 1.23 'S' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "S"

 ------------------------------------------------------------------------

 SAVESCREEN (Paul Bowlay, cat@aic.net.au)

 This one is a killer, and the one responsible for motivating me to write
 this guide in the first place.  It adds a whopping 8192 bytes to your final
 executable size.  That's right: this single command stacks on another 8K!

 So what can we do?  To be honest, I don't know.  This is the only tip in
 this guide to which I don't have an optimized version for.  I've tried just
 about everything, from ScreensBitMap and SaveBitMap (no luck) to grabbing
 the screen as a Shape and saving the shape (again, no luck).

 The only thing I can suggest is to IncBin a tiny third-party screen grabber
 and call it from your code.  However, there's no point in doing this if the
 external grabber's size is larger than 8K, because then SaveScreen would
 become the better option.

 OLD: SaveScreen screen#,filename$

 NEW: Call ?savescreen
      .savescreen:
      IncBin "external_grabber.exe"

 ========================================================================
```

## 1.24 'V' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "V"

 ------------------------------------------------------------------------

 VPOS (Paul Bowlay, cat@aic.net.au)

 Use VBeamPos_ instead.
```

```
OLD: pos=VPos
NEW: pos=VBeamPos_

================================================================================
```

## 1.25 'W' TIPS

```
>>> TIPS FOR COMMANDS STARTING WITH "W"

--------------------------------------------------------------------------------

WAITEVENT (Paul Bowlay, cat@aic.net.au)

If you're waiting for a single IDCMP event with a window, then don't use
the WaitEvent and related testing commands; use WaitFor instead.

OLD: Repeat : Until WaitEvent=$200
NEW: WaitFor $200

--------------------------------------------------------------------------------

WBHEIGHT (Paul Bowlay, cat@aic.net.au)

WBHeight has a severe bug, in that the maximum value it returns is 512.
Thus, Workbench screens higher than 512 pixels will give a false result.
To fix this, use the ScreenHeight command instead (assuming you've already
made Workbench the current screen).

OLD: height=WBHeight
NEW: height=ScreenHeight

--------------------------------------------------------------------------------

WBSTARTUP (Paul Bowlay, cat@aic.net.au)

This should go without saying, but... If your program is for CLI usage
only, then don't have WBStartup in your code.  WBStartup is only needed
if you want your program to be run from a Workbench icon.

--------------------------------------------------------------------------------

WBTOSCREEN (Paul Bowlay, cat@aic.net.au)

If you want to use Workbench as your screen, don't assign it to a number
and then bring that screen to the front.  Instead, bring Workbench to the
front first (with WBenchToFront_) and then FindScreen it.

OLD: WbToScreen # : ShowScreen #
NEW: WBenchToFront_ : FindScreen #

--------------------------------------------------------------------------------

WBWIDTH (Paul Bowlay, cat@aic.net.au)
```

```
WBWidth has a severe bug, in that the maximum value it returns is 640.
Thus, Workbench screens wider than 640 pixels will give a false result.
To fix this, use the ScreenWidth command instead (assuming you've already
made Workbench the current screen).

OLD: width=WBWidth
NEW: width=ScreenWidth

------------------------------------------------------------------------

WINDOWOUTPUT (Paul Bowlay, cat@aic.net.au)

If you're using WindowOutput after Use Window, then you're wasting bytes
because Use Window automatically does a WindowOutput by itself.

OLD: Use Window # : WindowOutput #
NEW: Use Window #

========================================================================
```