

WarpTIFF

Oliver Roberts

COLLABORATORS

	<i>TITLE :</i> WarpTIFF		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Oliver Roberts	July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	WarpTIFF	1
1.1	WarpTIFF.datatype 44.1	1
1.2	Description	1
1.3	Features	2
1.4	System Requirements	2
1.5	Installation	3
1.6	Preferences Options	3
1.7	Speed	4
1.8	Distribution Conditions	7
1.9	Disclaimer	8
1.10	Acknowledgements	8
1.11	Support & the Future	9
1.12	About the author	9
1.13	Program History	9

Chapter 1

WarpTIFF

1.1 WarpTIFF.datatype 44.1

WarpTIFF.datatype 44.1 - the fastest TIFF picture datatype!
(for 68k, PPC/WarpOS and PPC/MorphOS)

Copyright © 2000-2001 Oliver Roberts, All Rights Reserved.

Description	what is this datatype for?
Features	list of features
System requirements	what you need to use this software
Installation	installing this software
Preferences	descriptions of configurable settings
Speed	information regarding speed issues
Distribution	distribution conditions
Disclaimer	important notices
Acknowledgements	thankyous and credits
Support & the Future	support + improvements I intend to make
About the author	how to contact the author
History	program history

1.2 Description

WarpTIFF.datatype is a powerful, fast datatype capable of reading images using the TIFF image format that adhere to the 5.0 or 6.0 TIFF spec. In terms of encoding support, it offers the widest selection of any TIFF datatype on the Amiga. However, LZW support is not enabled due to legal restrictions concerning the LZW patent owned by Unisys.

As users of other datatypes in my WarpDT series will be familiar with, WarpTIFF comes optimized in a variety of cpu flavours, with PPC versions (native WarpOS and native MorphOS) and separate versions for 68020, 030, 040 and 060. Even better, it is fast, compact, clean, well behaved and fast - a true plug'n'play PPC datatype. One of the key features of my WarpDT engine is its superior speed, and WarpTIFF inherits that just like

the other datatypes in the series.

1.3 Features

- Supports 5.0 and 6.0 TIFF spec, including Packbits, Thunderscan, CCITT (fax), Pixar, LogLuv (CIE and SGILog) and Deflate (zip) encodings, and uncompressed files. LZW and JPEG encodings are currently unsupported
- Image types supported include RGB (24-bit and 48-bit), palette based and greyscale (1-bit, 4-bit, 8-bit and 16-bit)
- Supports image data organized as strips, tiles or separate components
- Highly optimized datatype dispatch engine, and fast TIFF decoder, resulting in a very efficient, compact and quick TIFF datatype
- Asynchronous file i/o and double buffering techniques (WarpOS only), which speeds up image decoding
- Optimized versions for 68020, 030, 040 and 060
- PowerPC support with native WarpOS and native MorphOS versions
- Alter the pen allocation precision when images are remapped to an 8-bit display
- Specific support for the OS 3.5/3.9 picture.datatype, when available
- The dithering feature of the OS 3.5/3.9 picture.datatype can be configured to your liking (e.g. disabled for 15/16-bit displays)
- Based on libtiff 3.5.6-beta and zlib 1.1.3

1.4 System Requirements

This datatype needs the following in order to work:

- Kickstart 3.0 or higher
- picture.datatype v43 or higher
(i.e. either of the ones supplied with AmigaOS 3.5/3.9, P96 or CGraphX)

68k version

- 68020 processor or higher (optimized versions included)

WarpOS version

- PPC accelerator card + 68040/060
- WarpUp Release 4 (powerpc.library V15) or higher

MorphOS version

- PPC accelerator card
- MorphOS beta release 1 or higher

Note that a graphics card is not necessary.

1.5 Installation

To install WarpTIFF.datatype, simply run the provided installer script by double-clicking the icon. This will create a CPU specific version of the library for you, and install it to SYS:Classes/Datatypes and also installs the TIFF descriptor file to DEVS:Datatypes. It does nothing else, so you needn't worry about it messing about with your system :)

1.6 Preferences Options

WarpTIFF can be configured using the "Datatypes/WarpTIFF.prefs" environment variable. The preferred way of altering the settings is via the WarpDTPrefs graphical user interface (requires OS3.5 or higher), available from <http://www.nanunanu.org/~oliver/warpdtprefs.html> or Aminet (util/dtype/WarpDTPrefs.lha).

However, it is also possible to alter the settings manually by changing the environment variable with setenv, according to the following template:

```
DITHER_OVERRIDE/K,DITHER_QUALITY/K,DITHER_DEPTH/K/N,PENS_OVERRIDE/K,
PENS_QUALITY/K
```

OS3.5 DITHER CONTROL OPTIONS

The three dither options all interact with each other and apply only when at least picture.datatype V44 (OS3.5/3.9) is in use.

By default picture.datatype V44+ dithers images for all target displays less than 24-bit. It is debatable whether it's worth using dithering on 15/16-bit displays - for photographic images, the difference is usually not noticeable, but for computer generated images containing smooth colour gradients dithering produces a noticeably better output image.

DITHER_OVERRIDE

Selects what our dither quality setting should override:

- | | |
|------------|--|
| NOTHING | - do not change the dither quality in any circumstances |
| DEFAULTS | - only override picture.datatype's default dither quality, which is used when the application does not specify it's own dither quality value |
| APPS | - override the dither quality that an application may set |
| EVERYTHING | - always use your custom dithering settings |

DITHER_QUALITY

Select the dither quality to use when this datatype decides that changes to the dither quality are required, according to your dither override and depth settings:

- POOR - simple colour remapping and no dithering (fastest)
- GOOD - slower, higher quality dithered output (default)
- BEST - similar to GOOD, with maybe slightly higher quality output

DITHER_DEPTH (Screen Bit Depth Filter)

Use this option to select which display depths should use the above quality setting. When the target display is deeper than the given depth value, the above dither quality will be applied to that image. For example setting this to 8 will cause the dithering settings to be used for 15/16-bit target displays, but left untouched when the target display is \leq 8-bit.

PEN SELECTION OPTIONS

The pen selection options are only relevant to 8-bit displays, and adjust the precision to which pens/colours are allocated, which affects the image quality and number of pens that will get used.

PENS_OVERRIDE

Selects what our pen/colour quality setting should override:

- NOTHING - do not change the pen precision under any circumstances
- DEFAULTS - only override picture.datatype's default pen precision quality, which is used when the application does not specify it's own pen precision value
- APPS - only override the pen precision that an application has set
- EVERYTHING - always use your pen precision settings

PENS_QUALITY

Changes the pen precision quality to use when remapping an image to an 8-bit display, in situations according to your pens override setting:

- POOR - use only a small amount of unique colours, at the expense of image quality
- GOOD - allocate enough pens to ensure a reasonable output image quality, whilst not hogging all pens for a single image (default)
- BEST - allocate as many pens as possible/necessary, resulting in the best image quality, but when displaying multiple images on the same screen, the quality of every image might not be so good

1.7 Speed

The fastest TIFF datatype?

The table below shows the time (in seconds) it took to decode 12 different images on my A1200 603e/240MHz 060/50MHz, with BVision and CGX picture.datatype, with other tested datatypes configured as close as

possible to WarpTIFF's internal settings. However, I should emphasize that the figures below are for comparison purposes only - the actual values are likely to be slightly different (faster or slower) on your system, but I'm confident that WarpTIFF is noticeably faster than other TIFF datatypes, whilst supporting more encoding types.

The figures speak for themselves... WarpTIFF for WarpOS is clearly much faster than other PPC datatypes where most image types are concerned, with WarpTIFF generally around 2-4 times faster depending on image type. For cpu-intensive codings the WarpOS version is about twice as fast as WarpTIFF 68k on a 060 (the speed difference will be even greater on 040 systems with a fast PPC). Maybe you don't believe these claims - my answer to that is simple... Try WarpTIFF for yourself and you will see the difference! :)

		PowerPC/68K (dual-cpu)		
		WarpOS	PowerUp	
		WarpTIFF	akTIFF 44.99	
		44.1	PPCLibEmu	PowerUp native
584x768x24	RGB uncompressed, 1.28Mb	0.56	1.81	1.84
584x768x24	RGB packbits, 1.29Mb	0.86	1.07	0.90
584x768x24	RGB deflated, 910K	0.88	*	*
800x607x8	Grey packbits, 190K	0.21	*	*
800x607x8	Grey tiled, 768K	0.37	*	*
333x225x8	Grey CIE LogL, 71K	0.48	*	*
333x225x24	RGB CIE LogLuv, 220K	1.25	*	*
487x414x8	Colour uncompressed, 205K	0.14	0.47	0.38
256x192x8	Colour packbits, 46K	0.07	0.28	0.19
1728x1028x1	B&W Fax, 32K	0.41	0.90	0.75
664x813x1	B&W uncompressed, 66K	0.19	0.47	0.37
1512x359x4	Grey thunderscan, 151K	0.19	*	*

(* = not supported, causes crash or corrupt output image)

The results for the MorphOS version are even more impressive, outperforming the WarpOS version, and outperforming akTIFF to a far greater extent with WarpTIFF being 3-7 times faster depending on image type.

		PowerPC only (no 68K)		
		MorphOS		
		WarpTIFF	akTIFF 44.99	
		44.1	PowerUp emulation	
584x768x24	RGB uncompressed, 1.28Mb	0.50	1.61	
584x768x24	RGB packbits, 1.29Mb	0.61	0.68	
584x768x24	RGB deflated, 910K	0.78	*	
800x607x8	Grey packbits, 190K	0.17	*	

800x607x8	Grey tiled, 768K	0.35	*	
333x225x8	Grey CIE LogL, 71K	0.21	*	
333x225x24	RGB CIE LogLuv, 220K	0.48	*	
487x414x8	Colour uncompressed, 205K	0.13	0.73	
256x192x8	Colour packbits, 46K	0.07	0.43	
1728x1028x1	B&W Fax, 32K	0.23	1.57	
664x813x1	B&W uncompressed, 66K	0.12	0.74	
1512x359x4	Grey thunderscan, 151K	0.14	*	
+-----+				

(* = not supported, causes crash or corrupt output image)

The results for the 68k version are much the same, relatively speaking. WarpTIFF is by far the fastest, able to decode most types of images 3-4 times faster than aktIFF.

		+-----+		
		M68K		
		+-----+		
		WarpTIFF	aktIFF	TIFFDT
		44.1	44.99	43.2
		+-----+		
584x768x24	RGB uncompressed, 1.28Mb	0.55	1.53	14.94
584x768x24	RGB packbits, 1.29Mb	0.87	1.07	15.35
584x768x24	RGB deflated, 910K	1.74	*	*
800x607x8	Grey packbits, 190K	0.22	*	2.68
800x607x8	Grey tiled, 768K	0.34	*	*
333x225x8	Grey CIE LogL, 71K	1.08	*	*
333x225x24	RGB CIE LogLuv, 210K	2.27	*	*
487x414x8	Colour uncompressed, 205K	0.13	0.44	2.33
256x192x8	Colour packbits, 46K	0.07	0.29	0.59
1728x1028x1	B&W Fax, 32K	0.55	2.09	*
664x813x1	B&W uncompressed, 66K	0.17	0.82	0.82
1512x359x4	Grey thunderscan, 151K	0.29	*	*
		+-----+		

(* = not supported or corrupt output image)

Note: all tests were performed, multiple times, using Visage, with the following command line: "visage test.tif time test". Most of the test images were taken from ftp://ftp.onshore.com/pub/libtiff/v3.4pics.tar.Z

WarpOS version still too slow!

Despite these facts, I'm still disappointed with the relatively slow performance advantage offered by my 603e over my 060. The main problem is that PPC datatypes still have to use the 68k for reading the data from disk and for creating / writing to the bitmap structures that the datatypes system requires. As far as the former goes, time lost for file i/o is negligible as WarpTIFF uses double buffered asynchronous i/o (supports DMA controllers).

The largest bottleneck is that the DTM_WRITEPIXELARRAY method of the picture.datatype has to be used to write the image data from WarpTIFF into the image bitmap. As this process is done via picture.datatype, it can only currently be performed by the 68k. To give you some idea of how much of a problem this is for WarpTIFF, typically, half of the overall decode time is used by the PPC to decode the whole image, and the other half is used by DTM_WRITEPIXELARRAY on the 68k. And that's on a graphics card - the time used by DTM_WRITEPIXELARRAY will probably be even greater on

systems using native Amiga graphics. It doesn't take a genius to see that this is slowing the datatype down, and is the main reason why WarpTIFF will still be faster on a 060 than a 040.

How to make the datatype faster

Is there anything that can be done about this? Well, yes, there are a few patches that you can install which should make things faster:

- NewWPA8 (util/boot/NewWPA8.lha on Aminet) should provide a notable speed increase on native Amiga graphics - probably won't make any difference if you use a graphics card.
- If you use a graphics card and CyberGraphX, you may want to make sure you are using the supplied v43 picture.datatype, as this will be faster than the P96 and OS3.5/3.9 picture.datatype on your system.

Of course, any other general speed-up patches should help too.

1.8 Distribution Conditions

WarpTIFF.datatype is public domain with the copyright remaining with the author and may be freely distributed legally providing:

- (1) None of the distributed files are changed in any way
- (2) It is not sold for profit and it is not included on any disks that are sold solely for profit (includes magazine coverdisks)
- (3) The distribution contents remain complete (see list below)

If this software is to be sold for profit, permission must be obtained from me, the author.

Aminet and Amigactive have been granted permission to distribute WarpTIFF.datatype on their CDs.

The following files must be present in their original and unchanged form in any copies of this software:

```
Classes/Datatypes/WarpTIFF.datatype.020
Classes/Datatypes/WarpTIFF.datatype.030.pch
Classes/Datatypes/WarpTIFF.datatype.040.pch
Classes/Datatypes/WarpTIFF.datatype.060.pch
Classes/Datatypes/WarpTIFF.datatype.wos
Classes/Datatypes/WarpTIFF.datatype.elf
Devs/Datatypes/TIFF
Devs/Datatypes/TIFF.info
WarpTIFF.guide
WarpTIFF.guide.info
Install_WarpTIFF
Install_WarpTIFF.info
spatch
```

1.9 Disclaimer

This software is provided "as is", without warranty of any kind, either expressed or implied, statutory or otherwise. By using the archive and its contents, you accept the entire risk as to its quality and performance.

Neither Oliver Roberts nor any other party involved in the creation, production or delivery of the archive and its contents shall be liable for any direct, indirect, special, consequential or incidental damages, including without limitation damages for loss of profits, loss of use or loss of anticipated costs, expenses or damages, and any data or information which may be lost or rendered inaccurate, even if Oliver Roberts is advised of the possibility of such damages.

Do not attempt to tamper with the supplied files. Doing so will cause problems and you may find things start going wrong!

libtiff

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

1.10 Acknowledgements

TIFF support based on the libtiff link library...

Copyright (c) 1988-1997 Sam Leffler
Copyright (c) 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

Deflate decompression support provided by the zlib link library by Jean-loup Gailly and Mark Adler.

The WarpOS version was made possible by VBCC, which was used to build and compile the datatype. Thanks to Volker Barthelmann and Frank Wille, for their support and help.

The Spanish installer translation is by Dámaso D. Estévez.

The French installer translation is by Philippe Bovier.

Thanks also to Sam Jordan for WarpOS and helping me out with various queries regarding it.

Finally, thanks to the OS 3.5 development team - now everyone has access to a 24-bit picture.datatype, I don't need to bother messing about adding dithering routines :)

1.11 Support & the Future

Some things that may appear in the future:

- I am undecided whether to include JPEG encoding support, since I have no idea how widely used it is. So, do let me know if you need it and I'll consider adding JPEG encoding support to WarpTIFF.
- If I can squeeze any more speed out of the datatype, in general, I'll do so :) And since this is the first release, I'm sure there's some more speed to come.

If you have any other suggestions, please let me know.

Future releases of WarpTIFF.datatype will be available from either Aminet (util/dtype/WarpTIFFdt.lha) or its webpage:

<http://www.nanunanu.org/~oliver/warptiff.html>

If you would you like to know when WarpTIFF is next updated, then you may want to subscribe to my announcement list to receive an e-mail informing you of the changes as soon as a new versions of any of my products are released. To subscribe, send a blank e-mail to

futura-announce-subscribe@yahoogroups.com

or go to

<http://groups.yahoo.com/subscribe/futura-announce>

1.12 About the author

If you have any problems with this software, or if you have any suggestions/queries, please contact me and I will do my best to sort any bugs out as soon as possible:

e-mail: oliver@futura.co.uk

www: <http://www.nanunanu.org/~oliver/>

icq: 34640231

1.13 Program History

44.1 (31.7.2001)

- Initial release.