WarpJPEG

Oliver Roberts

COLLABORATORS

	TITLE : WarpJPEG	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Oliver Roberts	July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

WarpJPEG

1.1 WarpJPEG.datatype 44.23

```
WarpJPEG.datatype 44.23 - the fastest 24-bit JPEG picture datatype!
                            (for 68k, PPC/WarpOS and PPC/MorphOS)
```


1.2 Description

As you've probably guessed, WarpJPEG.datatype is a yet another JFIF/JPEG datatype. The original difference being that it was for owners of PPC cards, and was targetted specifically at WarpUp (not PowerUp). Even better, it is fast, compact, clean, well behaved and fast - a true plug'n'play PPC datatype. Now though, 68k optimized and MorphOS native versions have been added, which also benefit from my WarpDT engine. One of the key features is its superior speed, hence the name WarpJPEG.

I originally decided to write this datatype because no other decent WarpOS native JPEG datatype existed for the PPC, and I thought this would be the ideal chance to finally learn how to start coding PPC stuff on the Amiga.

1.3 Features

```
 - Supports regular JFIF files, and also many JPEG variant formats such
   as Exif, Adobe, Photoshop, Mavi, Windows, JCCK/CMYK colorspace files
   and other less common formats

 - Supports both normal and progressive JPEG files

 - Full true colour (24-bit) and high quality reduced colour (256 colours
   or less) and greyscale output modes, for graphics cards and native
Amiga
   displays respectively

 - Highly optimized datatype dispatch engine, and fast JPEG decoder,
   resulting in a very efficient, compact and quick JPEG 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)
```

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 WarpJPEG.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 JPEG 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

WarpJPEG can be configured using the "Datatypes/WarpJPEG.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,OUTPUT_MODE/K,DCT_METHOD/K,FANCY_UPSAMPLING/T,DITHER/K,
QUANTIZATION/K,NUM_COLOURS=NUM_COLORS/K/N
```

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 noticable, but for computer generated images containing smooth colour gradients dithering produces a noticably 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 <= 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 otherride 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

DECODER OPTIONS

The following options define the internal settings of the JPEG decoder:

OUTPUT_MODE

Switch between full colour (24-bit colour / 8-bit greyscale) output or reduced colour (256 colours or less) output.

FULL - output a full colour image, which may be dithered down by picture.datatype to satisfy the target display requirements, if necessary (default)
REDUCED - output a reduced colour image, which results in much higher quality images if you only have 8-bit display modes

```
            available (e.g. if you don't have a graphics card).  The
            quality of this mode can be altered using the colour
            reduction options below.
  GREY    - always output a greyscale image.  The quality of this mode
            can be altered using the colour reduction options below.

Selects which decode method to use.  Two methods are available:

  SLOW - high quality, accurate integer algorithm (default)
  FAST - slightly faster (on 68K) integer algorithm, but sacrifices the
         output quality.

DCT_METHOD (DCT Quality)

Selects which decode method to use.  Two methods are available:

  SLOW - high quality, accurate integer algorithm (default)
  FAST - slightly faster (on 68K) integer algorithm, but sacrifices the
         output quality.

Note: PPC users should probably stick with the slow algorithm since the
fast methods is only fractionally faster on a PPC (on a 603e, at least).

FANCY_UPSAMPLING

  ON  - do careful upsampling of chroma components
  OFF - use a faster, but sloppier method.  The visual impact is usually
        very small, and the speed gains can be significent on some images
        (default)


COLOUR REDUCTION OPTIONS

These options are only relevant when using the reduced colour output mode:

DITHER

There are three dithering modes available:

  NONE    - no dithering, which is extremely fast, but the image quality
            will not be very good
  ORDERED - ordered dithering gives satisfactory output image quality,
            at a high speed
  FS      - floyd-steinberg dithering gives the very best output image
            quality, but at the expense of speed (default)

QUANTIZATION (Colour Quality)

This option changes the way in which colours are chosen for the palette of
a reduced colour image.  Note: this option does not have any affect when
ordered dithering is chosen.

  POOR - do a single pass quantization in order to create a satisfactory
         palette
  BEST - use two-pass quantization, which is slower, but creates an
         optimum image palette, and therefore a higher quality output
         image (default)
```

NUM_COLOURS

Select the maximum number of colours that the output image should use. This may be anything between 8 and 256 colours (defaults to 256).

1.7 Speed

The fastest JPEG datatype?

The table below shows the time (in seconds) it took to decode 10 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 WarpJPEG's internal settings. However, I should emphasize that the figures above 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 the WarpOS version of WarpJPEG is noticably faster than all alternative PPC datatypes.

The figures speak for themselves... WarpJPEG for WarpOS is around 70-90% faster on most images (twice as fast on greyscale and small colour images) than other PPC datatypes and around 3-4 times faster than 68k datatypes 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 WarpJPEG for yourself and you will see the difference! :)

	PowerPC/68K (dual-cpu)		
	WarpOS		PowerUp
	WarpJPEG 44.23	akJFIF 44.112 PPCLibEmu	PowerUp native
2008x1597 col, 225570 bytes	2.02	3.86	3.60
1280x1012 col, 148387 bytes	0.90	1.74	1.59
804x1040 col, 306821 bytes	0.91	1.56	1.46
1024x 766 col, 244522 bytes	0.86	1.54	1.41
779x 767 col, 162752 bytes	0.64	1.11	1.02
779x 767 col p, 86016 bytes	1.15	1.63	1.47
450x 450 col, 50845 bytes	0.26	0.48	0.41
370x 502 col, 24480 bytes	0.19	0.38	0.32
506x1007 gry, 88115 bytes	0.25	0.51	0.45
761x 495 gry p, 68593 bytes	0.40	0.61	0.54

(p = progressive)

The results for the MorphOS version are interesting. It's actually on par with the WarpOS version (above), and actually even faster in some cases, especially with larger images. It's also faster than the PowerUp version of akJFIF running under PowerUp emulation in MorphOS, where it's at least 30% faster. For smaller images, it's twice as fast, and for greyscale images it can be upto 5 times faster than akJFIF! No benchmarks for 68k

versions are included because they were found to be many times slower.

	PowerPC only (no 68K)	
	MorphOS	
	WarpJPEG 44.23	akJFIF 44.112 PowerUp emulation
2008x1597 col, 225570 bytes	1.55	2.03
1280x1012 col, 148387 bytes	0.74	1.01
804x1040 col, 306821 bytes	0.87	1.13
1024x 766 col, 244522 bytes	0.83	1.13
779x 767 col, 162752 bytes	0.60	0.82
779x 767 col p, 86016 bytes	0.77	1.05
450x 450 col, 50845 bytes	0.23	0.42
370x 502 col, 24480 bytes	0.15	0.34
506x1007 gry, 88115 bytes	0.23	1.14
761x 495 gry p, 68593 bytes	0.32	1.07

(p = progressive)

The results for the 68k version are much tighter, although overall, WarpJPEG comes out on top. Not by that much, but faster nonetheless:

	M68K			
	WarpJPEG 44.23	akJFIF 44.112	JFIFdt G.Nikl	OS 3.9
2008x1597 col, 225570 bytes	5.57	6.21	8.29	8.77
1280x1012 col, 148387 bytes	2.63	3.02	3.79	3.96
804x1040 col, 306821 bytes	3.29	3.65	3.47	3.50
1024x 766 col, 244522 bytes	3.11	3.47	3.33	3.31
779x 767 col, 162752 bytes	2.21	2.52	2.30	2.33
779x 767 col p, 86016 bytes	2.60	3.20	2.95	3.03
450x 450 col, 50845 bytes	0.72	0.90	0.76	0.76
370x 502 col, 24480 bytes	0.44	0.58	0.60	0.63
506x1007 gry, 88115 bytes	0.73	1.05	1.18	0.76
761x 495 gry p, 68593 bytes	1.01	1.28	1.26	1.05

(p = progressive)

Note: all tests were performed, multiple times, using Visage, with the following command line: "visage test.jpg nojpeg time test".

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 (approximately only 2-3 times faster). 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 negligable as WarpJPEG 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 WarpJPEG 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 WarpJPEG, 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 WarpJPEG 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

WarpJPEG.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, Amiga Format and Amigactive have been granted permission to distribute WarpJPEG.datatype on their CDs.

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

Classes/Datatypes/WarpJPEG.datatype.020
Classes/Datatypes/WarpJPEG.datatype.030.pch
Classes/Datatypes/WarpJPEG.datatype.040.pch
Classes/Datatypes/WarpJPEG.datatype.060.pch
Classes/Datatypes/WarpJPEG.datatype.wos
Classes/Datatypes/WarpJPEG.datatype.elf

```
Devs/Datatypes/JPEG
Devs/Datatypes/JPEG.info
WarpJPEG.guide
WarpJPEG.guide.info
Install_WarpJPEG
Install_WarpJPEG.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!

1.10 Acknowledgements

This software is based in part on the work of the Independent JPEG Group, using the libjpeg link library.

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:

- If I can squeeze any more speed out of the datatype, in general, I’ll

```
   do so :)

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

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

  http://www.nanunanu.org/~oliver/warpjpeg.html

If you would you like to know when WarpJPEG 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

  futaura-announce-subscribe@yahoogroups.com

or go to

  http://groups.yahoo.com/subscribe/futaura-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@futaura.co.uk
   www: http://www.nanunanu.org/~oliver/
   icq: 34640231
```

1.13 Program History

```
44.23 (17.8.2001)

 - Added greyscale output mode option.
 - Fixed bug in memory manager, which could have caused crashes under
   certain low memory conditions.
 - Now supports the PDTA_GetNumPictures and PDTA_WhichPicture tags.
 - Discovered a bug in the CyberGraphX picture.datatype, concerning
   error numbers which would have resulted in an unknown error being
   reported by application software if bitmap allocation failed - added
   workaround.

44.22 (24.7.2001)

 - Added a workaround to the WarpOS version to stop the Exec memory list
   getting trashed if/when the datatype is flushed from memory, due to a
   bug in VBCC 0.8.
 - Added French strings to the installer.

44.21 (17.7.2001)
```

```
 - Fixed bug in the JPEG descriptor file, which caused some types of
   file not to be recognized.
 - Added support for various webcam images that weren't recognized before,
   including images from the NASA site.
 - Further optimizations to the memory management code.
 - PowerPC versions recompiled with VBCC 0.8.

44.20 (12.4.2001)

 - Improved YCCK to RGB colour conversion routine (Adobe images using
   the YCCK colour space should now look much closer to the original
   image).
 - Added RGB colour conversion support for Adobe images using the CMYK
   colour space.

44.19 (4.4.2001)

 - Added an optional 8-bit output mode for non-graphics card users, with a
   number of dithering and quality settings, which produces a much better
   image than the dithering method currently offered by picture.datatype.
 - Added options to control the precision/quality of pen allocations for
   when an image is displayed on a pallette-based display (8-bit).
 - Added Spanish strings to the installer.

44.18 (15.3.2001)

 - Made fast integer DCT method available (approx 10% faster on 68k), and
   added a switch to toggle fancy upsampling.
 - Added preferences options to configure the OS3.5/3.9 dithering control
   feature.
 - Improved the memory management routines in all versions, bringing a
   small performance advantage for larger images on 68k (about 2% faster)
 - The 68k versions could crash if the JPEG stream was corrupt (bug
   introduced in v44.15) - fixed.

44.17 (19.2.2001)

 - The dithering disabler feature (for when the OS3.5/3.9 picture.datatype
   is in use on 15/16-bit displays) was flawed, since it only worked with
   images that were eventually attached to windows (e.g. Multiview). Now
   dithering will be disabled in all other cases too (e.g. Workbench /
   DOpus backdrops).

44.16 (12.1.2001)

 - Fixed a bug in the new startup code, which caused images not to be
   loaded in some cases - e.g. IPrefs failing to load WB backdrops.
 - The 68020-040 versions were completely broken in the previous release -
   forgot to rebuild one of the link libraries.
 - The 68k versions are faster again - moreorless back to the speed they
   were in 44.14.

44.15 (9.1.2001)

 - Recoded much of the startup code, with libraries now not being opened
   on the ramlib context (after advice from H&P)
 - Modified internal structure of the WarpOS version, to totally safeguard
```

```
   against cache conflicts (may have been causing rare random crashes)
 - All WarpOS benchmarks recomputed, since WarpOS V5 is a little bit
   faster than V4 (this datatype is upto 5% faster as a result)
 - Added benchmarks for MorphOS
 - Unfortunately, due to all the internal changes, the 68k version is now
   a little bit slower (only about 1%) than before

44.14 (7.9.2000)

 - 68k version is now a little bit faster (1-4% depending on the image).
 - Fixed bug in the previous release which caused all versions not to work
   with the P96 picture.datatype.

44.13 (2.9.2000)

 - Added a MorphOS native version and 68k version (in 020, 030, 040 and
   060 flavours) to accompany the original WarpOS version
 - Fixed bug which could case crashes under low memory conditions.

44.12 (2.8.2000)

 - Around 10% faster decoding of most images!
 - Recompiled with VBCC 0.7d beta, which gives a slight performance
   increase and smaller program binaries.
 - Tweaked compiler optimization options and increased disk i/o buffer for
   enhanced performance.
 - Improved the datafile descriptor file recognition code - it's faster
   now, and is able to detect more of the rarer types on JPEG file.
 - Recomputed all benchmarks, to ensure correctness after a change in my
   test environment (test files are now on a different HD, due to a
   semi-fatal HD crash on the old partition).

44.11 (7.5.2000)

 - Made the descriptor less pedantic, so it will now match JFIF files
   starting with any standard marker type - i.e. less common,
   unidentifiable, types of JPEG file should now be recognized too.
 - Minor optimizations and code changes.

44.10 (5.3.2000)

 - Descriptor now recognizes JPEG's saved by Graphic Workshop for Windows.
 - Very minor code changes.

44.9 (8.2.2000)

 - Fixed a problem in the error cleanup code, which caused a PPC crash
   after decoding some corrupt JPEG files.

44.8 (4.2.2000)

 - Added support for images using the Adobe YCCK format.
 - Descriptor now recognizes Photoshop 3.0/5.2 JPEG file variations.

44.7 (22.1.2000)

 - Fixed possible deadlock case (caused everything to freeze/lock-up)
```

```
   which usually only occurred when lots of images were being decoded
   simultaneously.

44.6 (15.1.2000)

 - Rewrote parts of the descriptor code, so that it still works if
   other datatypes have messed with the current file position.  Should
   also be faster and more efficient.
 - Now able to decode EXIF thumbnail images.
 - Minor code changes.
 - The installer will now remove any JFIF descriptor files.

44.5 (9.1.2000)

 - When the OS3.5 picture.datatype is in use, dithering is now switched
   off (only) if the image is to be rendered to a 15/16-bit screen,
   resulting in much higher performance, with negligable quality loss.
 - Enhanced descriptor file to detect EXIF and Photoshop JPEG file
   variations.
 - Fixed bug in error handling.
 - Dispatcher now performs some extra functions, which may quash the
   stability problems that some users have experienced.

44.4 (5.1.2000)

 - Fixed silly bug new to v44.3 which caused a lot of images to be
   rendered with half the vertical resolution (i.e. every other line was
   left out and the rest were doubled), resulting in blocky images.
 - Ironically, fixing the above bug enabled the optimizations that I had
   originally intended: 30-50% faster for colour images, and around 100%
   faster for greyscale images!
 - Reverted to my simpler and less hungry asynchronous engine, as the
   new engine in 44.3 no longer offers an advantage with the above
   optimizations in place. [after release this was found to solve major
   stability problems that some users experienced with v44.3]
 - Corrected akJFIF WarpUp benchmarks.

44.3 (3.1.2000)

 - At least 35% faster overall (upto 60% faster in many cases!), with
   no loss in the output image quality.
 - Enhanced descriptor file to detect more JPEG file formats.
 - Now uses asynchronous i/o routines which has made things a little bit
   faster overall, but will be most noticable on large (low compression)
   files and/or slower media.
 - Improved error handling.
 - Added Speed section to this documentation.

44.2 (18.12.1999)

 - Now at least 10-25% faster (depending on the JPEG file) after
   optimizations and tweaking libjpeg settings.
 - Greyscale jpegs are now displayed if using the OS 3.5 or P96
   picture.datatype (only worked with the CGX one in v44.1).
 - Fixed little bug in the lib init (wasn't setting the revision number).

44.1 (15.12.1999)
```

- Initial release.