JPEG.datatype

Steve Goddard

COLLABORATORS

	TITLE : JPEG.datatype		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Steve Goddard	April 25, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

JPEG.datatype

1.1   JPEG.datatype Documentation

```
Table of Contents
Introduction
Requirements
How to Install
What is JPEG?
What is a datatype?
Implementation Details (Release Notes)
Known Faults
Future Enhancements
Acknowledgements
```

1.2   introduction

This is V39.1 of the JPEG datatype. The version information can be
verified by using the AmigaDOS version command on the datatype file
SYS:classes/datatypes/JPEG.datatype after installation.

Once installed, the JPEG datatype means that any datatype-aware programs
can read JPEG image files.

Examples of this are that a JPEG image can be used as a WorkBench screen
backdrop, and Multiview can be used to display them.

The JPEG datatype is freely distributable, but you may not charge for it,
other than for reasonable media and distribution costs such as those
charged by Fred Fish and other reputable PD libraries.

If distributing the datatype as a standalone program, I ask that you
keep the complete archive together, so that users can benefit from these
instructions.

Permission is also given to distribute this datatype as part of a
product, either commercial or PD. In this case, the datatype will
presumably be considered as part of the product, and not as separate
item, and will be installed/set up as part of the product installation.

Under these circumstances, the action of the datatype is hidden, and
rest of the archive need not be supplied. I would ask however, that
some acknowledgement be given to myself and the Independent JPEG Group
in the product documentation. It would be nice if you also let me know
what it is being used for.

Please be aware that since is the first release of the program, no
attempt has been made to do any speed up/optimisation. That will be
for a future version.

1.3   requirements

Datatypes were introduced with WorkBench 3.0, so the main requirement is
that you have this version of the OS. Presumably, any future versions of
the OS will also support them.

Do not try to install the datatype if you are using an OS older than 3.0
(i.e. 1.3, 2.04, or 2.1). The correct directories do not exist, and
nothing will work even if you create them. I'm afraid you'll have to
find some other program to view your JPEGs with.

In view of the fact that every machine that Commodore have released that
has WorkBench 3.0 installed has been supplied with at least a 68020, the
datatype has been compiled for this processor. It will therefore not
work on machines with a 68000 or 68010, but a 68020 and above will be OK.

1.4   installation

Installation can be done in one of two ways:

1) Using Installer
2) Manual Installation

If you have a copy of Installer, you can simply double click on the
JPEG.datatype.install icon and follow the onscreen instructions.
The Installer script follows the manual sequence below, so you are
advised to read it quickly so that you know what to expect. In
particular, you are advised to choose a location for JPEGTMP:.

If you do not have Installer, follow the procedure below:

1) Copy the file JPEG.datatype to the directory SYS:classes/datatypes
(SYS: is a logical name for the disk you boot from). You can perform
the copy either from the WorkBench or the CLI. WorkBench users should
note that no icon exists for either the file or the directory, so you
will have to use the WorkBench Window->Show->All Files menu item to
make them visible.

2) Decide whether you want the datatype to be permanently enabled, or
manually run whenever you want to use it. Proceed to step 3 for a
permanently enabled installation, otherwise proceed to step 4.

3) Copy the file JPEG to the directory DEVS:Datatypes (DEVS: is a
logical directory on the disk you boot from. It is the same as the
DEVS drawer on the boot disk). If using the CLI, you will also have
to manually copy the icon file JPEG.info. When you reboot your system,
the JPEG datatype will be avaliable for use. Proceed to step 5.

4) Copy the file JPEG to the directory SYS:Storage/Datatypes. If using
the CLI, you will also have to manually copy the icon file JPEG.info.
The JPEG will not be available until you double click on the JPEG
icon, at which point it will remain installed until a reboot. Proceed
to step 5.

5) Modify your user-startup file to include the line
assign JPEGTMP: xxxx
You should replace the line xxxx with the location of somewhere big
and fast. A hard disk directory would be ideal, or possibly RAM disk
if you have enough RAM. If you do not make this assignment, the
datatype will ask for it if you ask it to decode certain JPEG images.

This completes installation.

1.5 jpeg

JPEG is a method of storing still images designed by the Joint Photographic
Experts Group in the late 1980s. It is quite a wide-ranging standard,
but most images only use a subset of the full specification.

The unusual thing about JPEG images is that they are lossy. This contrasts
with the lossless methods used by most other image storage formats
(including IFF). What this means is that some of the information in the
image is deliberately thrown away in order to reduce the storage
requirements. The result is that a JPEG is not an exact copy of the
original image, but it only takes up a fraction of the space. Technically,
this means that the quality of a decompressed JPEG image is not as good
as an IFF, for example.

Fortunately, the compression algorithm is very selective about what
information is discarded, and thus the average person would have some
difficulty in noticing the difference, particularly if the original image
is not available for reference. The JPEGs I have been using to test this
datatype have been of outstanding quality, with 256 colour images in a
file less than 20K in some cases.

1.6 datatypes

Datatypes were introduced with WorkBench 3.0, and provide a way for
applications to read data files without having to learn all the messy
details about how the data in the files are stored.

As an example, consider storing pictures on computer. As Amiga owners,
we are relatively lucky in that all images are stored using the IFF ILBM
format. Other computers have a bewildering array of incompatible formats,

and there exist commercial packages with the sole purpose of converting
images between them.

If I wanted to write a paint program, it would certainly be more useful
if it could read and maybe write out images in some of these other formats.
However, implementing such feature would be a bit of a nightmare, as I
would have to obtain the specifications of these formats, plus some
test images, and write (and support) a significant amount of code to
read and write these various formats.

A datatype is a little program that performs this conversion for me, and
completely hides all the messy implementation details about what file
headers exist, how they are structured, etc. My application can open the
datatypes library, pass a file to it and ask if it recognises the format.
The datatypes library has a look at the file, and then consults its list
of installed datatypes to see if any of them recognise it. It then comes
back to the application with a reply saying "Yup, its a JPEG picture".

My application can then ask for the data to be read in, and after a
short while the library will return the information in an Amiga standard
format, in this case, an IFF image. Of course, someone has to write the
datatype, but once written and installed, any other program that uses
datatypes can take advantage of it at no extra cost or expense.

This datatype decodes JPEG images, and any application that wishes to
read this format of images can do so without having to know anything about
them.

1.7   implementation

These are the release notes for V39.1 of the JPEG datatype.

This implementation will only decode JPEGs in up to 256 colours. If you
have a JPEG with more than this number of colours, it will be quantised
down to 256. In theory, the datatype could be enhanced to use HAM8 for
JPEGs with more colours.

If you are using Multiview to display JPEGs without the SCREEN tooltype/
parameter, the JPEG will be displayed in a window on the WorkBench,
it will be quantised down to however many colours you have set up on your
WorkBench screen.

If you are using Multiview with the SCREEN tooltype/parameter, the
datatype will choose the screen resolution that it thinks is most
appropriate for displaying the image from the selection available in the
display database. This descision is made using the BestModeID function,
and is based on the size, colour count and aspect ratio of the image.

The JPEG datatype is based around the JPEG decompression code available
from the Independent JPEG Group. There are many advantages is using an
off-the-shelf item, such as this, not least of which being that I didn't
have to spend time poring over the JPEG specification trying to figure
out what it all meant. The flip side of the coin is that in certain
regards, the datatype is constrained by the code I have used.

The JPEG specification is a complicated thing, and the JPEG engine does
not purport to decode every possible implementation. In addition, some
aspects of the spec may be covered by copyright and until the legal
aspects are clarified, any images that use these aspects cannot be
decoded.

If the JPEG engine detects any errors during decoding, it will display
an error requester containing the reason for aborting.

Certain JPEGs require a large amount of temporary space. If this exceeds
about 1/2 meg, the JPEG engine will attempt to store the information
on disk. It will expect the logical name JPEGTMP: to be set up. If this
has not been done, a requester will ask for this volume to be inserted.
At this point, a CLI can be opened, and by performing the assignment
and clicking on Retry, the decode can proceed.

Due to the number and structure of global variables used by the JPEG
engine, the datatype is not reentrant. It protects itself by only
allowing one JPEG to be decoded at a time. If you set off two
simulataneous decodes, one will wait for the other to complete.

1.8 faults

Firstly, the JPEG datatype will not decode every JPEG that exists. The
underlying JPEG engine is still in development by the Independent JPEG
Group, and from time to time they may release enhanced versions that
decode a wider range of JPEGs. This version seems to cope quite well
with the range of JPEGs I have access to, but if you see an error message
from the JPEG engine (such as 'invalid color conversion'), then you have
a JPEG it can't manage.

The following list describes faults that have been noted during testing:

1 Occasionally, when used with Multiview, a requester from Multiview
bearing the cryptic message 'Error -1' appears.

2 There may be a memory hole, such that memory disappears when images
are decoded. This has defied all my efforts to track down, if indeed it
is still there.

1.9 enhancements

1 The JPEG datatype is based on Release 4 (10-Dec-92) of the JPEG
software by the Independent JPEG group. As newer versions are released,
I will try and incorporate them into the JPEG datatype. Future versions
may well improve the speed of decode, and increase the range of JPEGs
that can be decoded.

2 The rest of the JPEG datatype is based on version 39.1 of the GIF
datatype I have released. There are a number of significant speed
improvements that were subsequently made to the GIF datatype which have
not yet been incorporated into this one.

1.10 acknowledgements