PNG_IO Version 4.6

User and Reference Manual

PNG_IO is an Ada 95 package for input/output of images and graphics in Portable Network Graphics (PNG) format.

January 12, 2016

Contents

1 Introduction **4**

1.1 About PNG 4

2 Detailed description **4**

2.1 Prerequisites 4

2.2 Functionality 5

2.3 Limitations 6

2.4 Implementation details and design rationale 6

3 Installation **7**

4 Comments and bug reports **7**

5 Mailing list **7**

6 Copyright **8**

A PGP and keys **9**

B Release notes **9**

B.1 Distribution 9

B.2 CVS Access 10

B.3 Changes in the current version 10

B.4 Changes in previous versions (earliest first) 11

B.5 Possible future changes 14

C Gnu Free Documentation License **16**

1 Introduction

This is the user guide and reference manual for PNG_IO, a portable Ada 95 package for reading and writing Portable Network Graphics (PNG) files. It is designed to provide Ada 95 (and of course Ada 2005 and Ada 2012[1]) programmers with direct access to images and graphics stored in PNG format files, and to be portable across architectures and operating systems[2].

1.1 About PNG

PNG is an image file format supporting greyscale and colour images with and without alpha channels. It is patent-free and offers good levels of lossless compression. The PNG format is defined by the PNG Specification which is available from the PNG website at http://www.libpng.org/pub/png/ and various mirrors. PNG is an ISO standard:

> Portable Network Graphics (PNG) Specification (Second Edition)
> Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification.
> ISO/IEC 15948:2003 (E)

(available at: `http://www.w3.org/TR/2003/REC-PNG-20031110`)

Detailed notes on changes since the previous release are given in section B.4.

2 Detailed description

2.1 Prerequisites

PNG_IO depends on the Zlib library to perform the compression and decompression of image and text data required by the PNG Specification. Zlib is written in C but an Ada 95 binding for it is available and distributed with Zlib source code. The Zlib website is easily accessed from the PNG website (see above) and the source code or a precompiled Zlib library should be downloaded if not already available on your chosen machine. Any program using PNG_IO must be linked with Zlib_Ada, and with Zlib, or have access to Zlib through a shared library or dynamic link library. The most recent version of Zlib_Ada is available from http://zlib-ada.sourceforge.net/. The earliest version of Zlib_Ada that will work with PNG_IO is version 1.3.

[1] Since Ada 95 code is legal Ada 2005 and Ada 2012 code, the package can be used with Ada 2005 and Ada 2012 code. For the moment, however, PNG_IO remains an Ada 95 package in order to work with Ada 95 compilers. This may change in the future.

[2] PNG_IO has been used by the author on PCs running Windows and Linux, and on both a Mac Powerbook and MacBook Pro running Mac OS X.

2.2 Functionality

The main package specification `png_io.ads` contains most of the functionality and has detailed comments.

A PNG file to be read must first be opened. Once this has been done, information about the content of the file is available through functions, and the pixel (and palette values, if any) may be read from the package (they are stored in internal buffers). When all the required information has been read, the file should be closed (this frees the internal storage used to hold the file information).

To write a file, the user must instantiate a generic procedure for the appropriate type of PNG file. One of the parameters supplied to instantiate the procedure is a function supplied by the user that allows PNG_IO to access the pixel values.

The child package `PNG_IO.Chromaticity_Data` provides standard data on chromaticity for use with the chromaticity chunk handling introduced with version 2. This obviates the need for users to look up this data for standard colour spaces. References are given in the source code to the sources used for the data.

The child package `PNG_IO.Standard_RGB_Encodings` provides functions to implement the encoding and decoding of sRGB samples from linear samples. When reading or writing an sRGB image, PNG_IO provides the sample data as it is found in the file, or expects the sample data as it is to be written to the file. This behaviour is consistent with that adopted for handling sample values in other cases: the user of PNG_IO must perform gamma adjustments if the gamma is not unity. For compliance with the sRGB standard the samples must be encoded according to a non-linear encoding, and the new child package provides functions to perform this encoding. It is, however, up to the user to pass the correct chunks and encoded samples to PNG_IO in order to create a compliant sRGB PNG file. PNG_IO does not check this.

The child package `PNG_IO.Gamma_Encodings` provides a similar function for mapping from one gamma value to another.

The following simple utility programs are provided:

png_test This program is meant as a simple way to test PNG_IO. It reads a PNG file and outputs the same pixel data (and palette if appropriate) to a second file. This verifies both the read and write capability of PNG_IO on the given file.

png_compare This program compares two PNG files, pixel by pixel. Any discrepancies are reported, otherwise the program reports that no differences were found. The exit status is set by this program so that it can be called from a script, to facilitate testing with large numbers of files.

png_chunks This program outputs a listing of the chunks in a PNG file, with their offsets from the start of the file, and the chunk lengths. It does not read the chunk data, nor verify the cyclic redundancy check values in the chunks.

png_properties This program outputs information about the main parameters of the given PNG file, such as height and width, PNG type, whether the file contains a gamma value, and so on.

png_dump This program outputs a dump of the pixel data in the PNG file *after decompression and filtering* but without applying gamma or other corrections. The pixel values are output in hexadecimal, but when the number of bits per pixel is less than 4, this reduces of course to the range 0 .. 1 or 0 .. 3. The

palette, if present, is also output, in hexadecimal. The dump produced by this program is much more useful than a raw dump of the file data, because the raw data in the IDAT chunks is compressed with Zlib.

2.3 Limitations

The interface to PNG_IO is not 'single-step' (*i.e.* pass an image array and a filename and go). Instead, to read a file, you have to open the file, read the dimensions and type of image, and then call functions in PNG_IO for each pixel to get the pixel values. You store them into your own image array (or process them on the fly if you wish). The reasoning behind this is that PNG_IO is stand-alone and independent of any data types for image pixels. It is intended to be used as a foundation to build the single-step image I/O packages that you might want. Thus if you use colour image pixels represented as records and use 2D arrays of these records to represent an image, then you can write a trivial image read routine that calls PNG_IO to do all the messy work and provide the pixel values. All you have to do is a type conversion to the component type of your records and store the RGB values into your records, looping over all the pixels in your image. For writing, all you need do is provide functions to access the pixel components of your image. PNG_IO does the iteration over the whole image.

At present PNG_IO does not use tasking, and therefore calls on its functions and procedures block until the actual file input/output has been done. A future version may change this to permit I/O to be overlapped with data transfer between the package and the user's code.

2.4 Implementation details and design rationale

PNG_IO consists of a main package and several child packages.

PNG_IO is independent of other packages, other than Zlib and Zlib_Ada, which are very widely available. Of course, it uses standard Ada 95 packages too.

Passing of image data to and from the user's code is done with standard integer types. This keeps the design of the package simple, but it does complicate the user interface, as explained in section 2.3. However, if you compare this to what is required to use `libpng` (assuming an Ada 95 binding existed, which is currently not the case), then PNG_IO provides a relatively straightforward means to read in or write out a PNG file.

A significant difficulty in designing a package to handle image file formats is the existence of different types of image data in the same file format. A PNG file may contain greyscale or colour images, with or without alpha information, and the bit depth can vary from 1 to 16. It is therefore hard to provide a strongly-typed interface, particularly for writing files. The solution adopted was to provide a separate generic write procedure for each type of PNG file. This means the procedure parameters may be tailored to the file type. In the case of application code that intends to handle all the legal file types, this means that a case construct is needed to cope with the different PNG types. The test program provided as part of PNG_IO illustrates how this is done.

PNG_IO was not written to be fast, although improvements have been made over the years to improve speed. It was written, above all, for correctness, and it assumes (when reading) that enough memory is available to hold the whole of the file data at once (this is no longer true on write from version 4.4 onwards). This will

work for images of reasonable size (such as may be taken with a digital camera), but it may cause problems on machines with small memory when handling high-resolution images.

PNG_IO makes some (reasonable) assumptions about the Ada 95 implementation that will compile it. These assumptions are checked by assertions in the code (one is that the default `Integer` type has at least a 32-bit range, and the other is that `Ada.Streams.Stream_Element` is a byte). These assertions are in the private part of the package `PNG_IO.Base`. When using PNG_IO for the first time it is advisable to compile with assertions enabled[3].

3 Installation

Once you have Zlib and Zlib_Ada compiled and available for linking, all that you need to do is compile the various units making up PNG_IO. (Use optimisation.) A simple test program (`png_test`) is provided, and the simplest thing to do if your compiler has a make facility is make the test program (but remember to link with Zlib). The test program reads a PNG file and outputs the same image to another PNG file. If you can view this file and it matches the image content of the file you passed to the test program you are in business! A more rigorous test is to compare the original file with the file output by `png_test` using the program `png_compare`, which performs a pixel-by-pixel comparison on two PNG files.

PNG_IO has been tested using the PngSuite of PNG test images created by Willem van Schaik and correctly reads and writes all the legal PNG images in that test suite[4] as of the January 2013 version. This may be verified by running the program `png_test` (see above) on all the images in the test suite using a batch file or script (a suitable bash script is provided with PNG_IO from version 4.5.1 onwards). The illegal files in the PngSuite are correctly rejected, with suitable exceptions and exception messages where applicable. Note that these tests verify the pixel data, but not the writing of other chunks, although the ability of PNG_IO to read other chunks is tested to some extent.

Additionally, the PNG file(s) written by PNG_IO may be compared pixel-for-pixel with PNG file(s) containing the same image written by another PNG coder using the program `png_compare`.

4 Comments and bug reports

Please send comments, suggestions or bug reports to: sangwine@users.sourceforge.net or use the facilities at Sourceforge (http://png-io.sourceforge.net/) to log a request or bug.

5 Mailing list

A low traffic mailing list exists for PNG_IO. The mailing list can be accessed from the PNG_IO page at Sourceforge, or directly at
`https://lists.sourceforge.net/lists/listinfo/png-io-announce`.
New releases and major updates to the CVS repository will be announced on this list.

[3] The assert pragma is not standard Ada 95 although it is standard in Ada 2005 and Ada 2012. If your compiler does not support this pragma, you should verify by other means that the assert conditions are met.

[4] The test suite images are available from `http://www.schaik.com/pngsuite/`.

6 Copyright

Each source code file contains a copyright notice (*q.v.*). The code is released under the Gnu General Public License, a copy of which is included with the distribution (in the file `gpl.txt`). From version 4.5 of PNG_IO onwards the license is version 3 of the Gnu General Public License. Previous versions were licensed under version 2 of the GPL.

If you wish to use PNG_IO under terms different to those of the GNU General Public License, please contact the author. A commercial license for PNG_IO version 4 may be negotiable. In addition, the author may be willing to provide consultancy services to enhance or customise the library.

A PGP and keys

PNG_IO distributed files are accompanied by detached signature files (extension `.sig`). To verify these (and it is not essential to do so) you will need a version of PGP or GPG which supports DH encryption, and my public key, which is given below and is also available from key servers and from my web page at http://privatewww.essex.ac.uk/~sjs.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 6.0.2i

mQGiBDU3GkARBADyn6tgErL8GsO/+UkVIMmMR65FgaFhsUue+/90hIRBqRJGF0Xd
RbiW9dm8MNb/2vl2XAcuGxrvokWzdt1fullFJAUuwa/tKOWX8fXuThUjdfS0GpJS
saqlJf74z1R8FT+YTJe2JCOl4vk733ctUGsSYc9hvCgFRsu9e1d9Qvhe8wCg/7Zb
Pazd/WpLuqOFy1WiOzbuN4UEAOOQqo/3Z1/bZeEoCoQNIkSnHzybQLbaZlNlvhcY
IuslKshJT6Moi4ERPpSUJHRh/B+hHcDb8a0dFYdQ89v8EvSYmFW9hL7geH2Wh27z
4giDlTpIpsj+cbdbkp9JV7OYDjuFvblvr+T+r04+D9dUOjeFnVVkwyveYTHovwHh
W4ysA/4vMojMUw2M2hI2dzL6i+eL59przRP40s62G6tQUWvLw2+wlSGw4RncG0/p
xKNCm9Sry/EZimygFU8I6ymBWcZ6FLnjON2xyaVp+VV15nSFUhdA1xcSXY2y41zU
UtEg5706gy/K1On7D1nbrDN/yzjJZ+fLK61R5h3K5Icz6n7Sc7QrU3RldmUgU2Fu
Z3dpbmUgPFMuSi5TYW5nd2luZUBSZWFkaW5nLmFjLnVrPokASwQQEQIACwUCNYpQ
zAQLAwIBAAoJEEGGEV9WsZDtMdsAn1EcgHkoNWUqC6TUZPUQ3S/Az3vGAKDgOdXj
LbvPdh3j/UD8NoDvsqzKMbQpU3RlcGhlbiBKLiBTYW5nd2luZSA8Uy5TYW5nd2lu
ZUBJRUVFLm9yZz6JAEsEEBECAAsFAjU3GkAECwMCAQAKCRBBhhFfVrGQ7QygAKDC
SBp0U+00JiQBV6WHNWA4YrR6CQCgnliLaMqUuEHbvKs8KA3qs9LPkcu0JlN0ZXZl
IFNhbmd3aW5lIDxzanNhbmd3aW5lQGllZS5vcmcudWs+iQBLBBARAgALBQI1ilCg
BAsDAgEACgkQQYYRX1axkO3zJgCg8XMgd/+0+Us2zfsMDfYXdeiY9oQAoOvr7Xk5
k2OAw0qDU5TI7/5iVSAAuQINBDU3GkAQCAD2Qle3CH8IF3KiutapQvMF6PlTETlP
tvFuuUs4INoBp1ajFOmPQFXz0AfGy0OplK33TGSGSfgMg7Il6RfUodNQ+PVZX9x2
Uk89PY3bzpnhV5JZzf24rnRPxfx2vIPFRzBhznzJZv8V+bv9kV7HAarTW56NoKVy
OtQa8L9GAFgr5fSI/VhOSdvNILSd5JEHNmszbDgNRR0PfIizHHxbLY7288kjwEPw
pVsYjY67VYy4XTjTNP18F1dDox0YbN4zISy1Kv884bEpQBgRjXyEpwpy1obEAxnI
Byl6ypUM2Zafq9AKUJsCRtMIPWakXUGfnHy9iUsiGSa6q6Jew1XpMgs7AAICB/9P
24ofRoqQVvyRv1julDbGThnmv7BJhxItoh5U1/MVksv9I6WktzgFWqMzSASoEzfs
tc2DSnnmKR9yIiX1jESFHHYkZE9ba6sPM1+de57p301isU6FaTcbcwHOv1lHXG0T
0xz4H4sBw0ZQ+3DzpMoXN248/BZWEaP96WyV1JGNEs9ijC4krDZKY8XwvGDWwc6E
t1XofqiOsR6+QkurMgdg9RrR20OlW4FEraiOn7/RMPFnlGAz/kcds7VwUuAfS1GK
zkyHDLH7NrE1Rtg4rLCHRbt++zIWN7ng7pMY0T3UwBEIQFR5+Xo2/2+MMOKpkYvX
jxu1OTEA+80diyGIGfOmiQBGBBgRAgAGBQI1NxpAAAoJEEGGEV9WsZDtLYkAnjDa
hdvZazzXkJ2ZUzNZpViY5AcqAJ988JCPvUTngaxYHvMx2V2LLdszYg==
=hhoI
-----END PGP PUBLIC KEY BLOCK-----
```

The key signature is given on my web page at http://privatewww.essex.ac.uk/~sjs.

B Release notes

PNG_IO is available from http://png-io.sourceforge.net/ and was originally developed at the University of Reading, UK. From January 2001, fixes and enhancements have been made at the University of Essex, UK.

PNG_IO is distributed in zip format[5].

B.1 Distribution

The distributed files of PNG_IO are accompanied by detached signatures so that the integrity and authenticity of the code can be verified, if desired. See section A for information about PGP or GPG and keys.

The PNG_IO distribution consists of the following 28 files, plus this manual:

[5] Earlier versions were also distributed in tar format, but since nearly all operating systems now have support or free software for the zip format, I don't intend to make PNG_IO available in tar format in future.

gpl.txt	Copy of the GNU General Public License
png_io.ads	Package spec
png_io.adb	Package body
png_io-open.adb	Separate body
png_io-chunk_ordering.adb	Separate body
png_io-adaptive_filter.adb	Separate body
png_io-write_png_type_0.adb	Separate body
png_io-write_png_type_2.adb	Separate body
png_io-write_png_type_3.adb	Separate body
png_io-write_png_type_4.adb	Separate body
png_io-write_png_type_6.adb	Separate body
png_io-adam7.ads	Child package spec
png_io-adam7.adb	Child package body
png_io-base.ads	Child package spec
png_io-base.adb	Child package body
png_io-chromaticity_data.ads	Child package spec
png_io-gamma_encoding.ads	Child package spec
png_io-gamma_encoding.adb	Child package body
png_io-generic_image_dump.ads	Child package spec
png_io-generic_image_dump.adb	Child package body
png_io-standard_rgb_encodings.ads	Child package spec
png_io-standard_rgb_encodings.adb	Child package body
png_test.adb	Simple test program
png_compare.adb	Simple pixel-for-pixel comparison
png_chunks.adb	Simple chunk listing program
png_dump.adb	Simple pixel data dump program
png_properties.adb	Simple parameter dump program
test_png_suite.sh	bash script for testing on PNG Suite

The source code files use the ISO-8859-1 Latin 1 character set, but the actual Ada code (outside comments) is restricted to the ISO-646 subset (originally known as ASCII).

The files have UNIX line endings (line feed).

B.2 CVS Access

The PNG_IO distribution site at http://png-io.sourceforge.net/ also allows access to updated versions of individual source code files through CVS. These updates eventually get incorporated into a release.

B.3 Changes in the current version

PNG_IO version 4.6 has updates to email addresses in the code as well as the manual. Two small error fixes are: at `png_io-open.adb` line 410, to add a type conversion required to fix incompatible types in a range; and at line 683 to add a `use type` clause (enclosed in a new block), required to make the =

operator for Zlib.Count visible (these errors were detected when the code was compiled with gcc 4.9.1, but had not been seen before).

Redundant type conversions were removed at line 64 of `png_io.gamma_encoding.adb` (warnings were seen with gcc 4.9.1. which had also not been seen before).

The bash script for running tests on the PngSuite of test images has been updated slightly and the code has been tested on the 2013 release of the test suite.

B.4 Changes in previous versions (earliest first)

PNG_IO Version 1.1 was the first publicly-released version.

PNG_IO Version 1.1a incorporated a fix to prevent PNG_IO rejecting a PNG file in which the compressed image data is larger than the uncompressed data.

PNG_IO Version 1.2 had a cleaner interface to Zlib (an internal change with no impact on functionality).

PNG_IO Version 1.3 incorporated a number of fixes:

Buffer allocation in write procedures. The buffers in the write procedures and the procedure to write an IDAT chunk now use dynamic allocation rather than stack variables, as has always been done when allocating buffers for read.

Exact handling of IDAT sizes on read. The fix included in Version 1.1a (see above) has been replaced by a much better solution. The buffer allocated to hold compressed data read from IDAT chunk(s) is now sized exactly by reading the IDAT chunk size(s) from the file before reading the IDAT data.

Adding this fix exposed a latent bug in which Type 4 PNG images (greyscale with alpha) were read incorrectly (the alpha information was read from the next pixel, and the alpha information for the last pixel in the image was garbage). This was because the implementation of the function `Alpha_Value` assumed a RGBA image (Type 6) and failed to check for a Type 4. This has been fixed.

Legality of Zlib compressed data. A check has been inserted on the validity of Zlib compressed data (in IDAT and zTXt chunks) by verifying the first two bytes.

IEND CRC now verified. It was noticed that the IEND chunk data was skipped, even though the length of this data is always zero. The redundant code was removed, and it was also noticed that the CRC of the IEND chunk was not verified. This has been fixed.

Calculation of compressed buffer sizes. The calculation of the size of a Zlib compressed data buffer was incorrect for uncompressed buffer sizes greater than 4,290,676 bytes. (Zlib requires the buffer to be 0.1% plus 12 bytes larger than the uncompressed data.) The calculation is now provided in a more robust manner by a new function in the package `generic_zlib`, and this function is now called wherever PNG_IO needs to calculate a buffer size. (The package `generic_zlib` was removed with PNG_IO version 4, and the calculation was moved into the main package body.)

Handling of text chunks is incorrect. The code for allocating buffers for compressed text chunks was incorrect. There is no perfect fix for this, as there is no way of knowing the size of the uncompressed text. (The size could easily have been inserted in the zTXt chunk and this would have solved the problem.) The code in version 1.3 allocates a buffer three times the size of the compressed text. Whether this is large enough is not known.

Also the keyword was not checked for legality as per the PNG Specification section 4.2.3. This has been rectified.

PNG_IO Version 2.1 incorporated several new features. The specification was changed (but it preserved backward compatibility with version 1). Support was added on output for handling ancillary chunks, including chunks like gamma and chromaticity which were previously supported for reading only. The generic write procedures were given an additional parameter for passing a list of ancillary chunks, and functions were introduced for creating the commonly used chunks and appending them to a list. Since the extra parameter appears last, and defaults to an empty list, code written for version 1.x did not need modification. Any unrecognised (by PNG_IO) chunks found in a file on read were made available to user code, using an interface similar to that

already provided for text chunks. Of course, the user's code has to interpret the array of bytes passed by PNG_IO since PNG_IO does not understand the content of the chunk. An important aspect of the way the ancillary chunks are handled is that it is possible to extend PNG_IO with child packages, which have full visibility of the ancillary chunk implementation.

The sRGB chunk was supported for both reading and writing, making use of the ancillary chunk mechanism just described, and the other chunks formerly supported on read (gamma, chromaticity, physical, text) were also supported on write.

A new exception, `Argument_Error` was introduced to represent cases where incorrect data is passed to a PNG_IO subprogram (example: invalid keyword string when creating a text chunk).

PNG_IO Versions 3.1, 3.2, 3.3 and 3.4 include enhancements and fixes as listed below (version 3.1 was used for some time without a public release and 3.2 was the first public release of version 3.x):

Interlace output is now supported Versions 3.3 onward support interlaced output for all PNG image types.

Incorrect alpha values in PNG Type 6 Version 3.3 fixed a bug in which PNG Type 6 images with 16-bit depth had incorrect alpha values written to the file in some pixel positions.

Default value for gamma The `Gamma_Chunk` function now has a default value for the parameter Gamma. This default value corresponds to a gamma of unity.

A bug in the program `PNG_Compare` was fixed This program was introduced to provide a simple way of comparing the pixel data in two PNG files, but it contained a stupid error: the first file was read twice and the second was ignored. Even Ada can't detect errors of that stupidity, and the program was not properly tested.

New child packages Two new child packages were added to handle sRGB encoding and to provide standard chromaticity data for use with chromaticity chunks. Details are given in section **??**

A bug with tEXt chunks was fixed tEXt chunks with a zero length text string raised the exception `Constraint_Error` when in fact the zero length string is legal (PNG Specification version 1.2, section 4.2.3.1). The change needed was to replace Positive by Natural to allow a value of zero to be computed and used for the string length. The rest of the code was already correct.

New function `Sample_Depth` A new function `Sample_Depth` has been added to the visible part of the specification. This function returns the number of bits per sample which is normally the number of bits stored per sample in the IDAT chunks, but in the case of palette images, the number of bits per sample is always 8, because this is the number of bits per sample in the palette. This function has been found useful in several programs written by the author which use PNG_IO and it seemed sensible to put it in the package itself as it could be generally useful.

Detection of attempt to close unopen file The procedure Close now raises `Status_Error` on an attempt to close a PNG file which has not been opened or which has already been closed. This behaviour is modelled on that of standard Ada packages like `Direct_IO`. Previously, Close would raise `Constraint_Error` in this situation.

Some variables which were not modified are now declared as constants The variables were flagged by compiler warnings in later versions of Gnat and were reported by a user. The changes were made in version 3.4 and eliminate the warnings, but obviously have no effect on the correctness of the code.

Some Inline pragmas have been removed Some functions had Inline pragmas applied alongside their bodies. Moving these to the corresponding spec was considered after a user reported an issue with them, but on review, it appeared that the functions didn't need a pragma Inline as the number of likely calls did not justify the effort to get the code inlined. From version 3.4 on the pragmas were removed.

PNG_IO version 4.0 includes the following fixes and enhancements.

The function `Palette` has been corrected This function previously returned `True` only if the image was of colour type 3. It now also returns `True` if a palette is present (this is legal for colour PNG file types, where the palette is advisory). The code already verified that the presence of the palette was legal when it was encountered in the file (`Format_Error` is raised if it is not legal).

New child package A new child package has been added for handling gamma encoding.

Zlib_Ada is now used The interface to Zlib is now through the Ada 95 binding Zlib_Ada. The former PNG_IO code for interfacing to Zlib and for computing CRCs has been removed.

IDAT data is no longer stored in an internal buffer before decompression. In previous versions all of the IDAT data in a PNG file was read into a buffer before decompression. In version 4 onwards, the data from each IDAT chunk is passed to Zlib for decompression using a temporary buffer. (If the file contains only one IDAT chunk the behaviour is similar to the previous code, but if there are multiple IDAT chunks the memory usage will be lowered compared to the previous code, since the compressed data will be read one chunk at a time, and decompressed one chunk at a time.)

Multiple IDAT chunks are now written on output. The change to Zlib_Ada has made it natural to output multiple IDAT chunks as the compressed data is passed back from Zlib to avoid the complexity of storing the data in a buffer. This results in variable lengths of IDAT chunks depending on the compression. This may result in slightly larger file sizes, but it does not result in invalid output since the PNG specification explicitly permits arbitrary division of the IDAT stream into chunks.

PNG_IO version 4.1 had only minor changes, and was the first release hosted at Sourceforge. Version 4.2 was identical – the version number in the code was set to 4.2 when it should have been 4.1. Release 4.2 simply corrects this minor error. Only the manual and the release number were changed.

PNG_IO version 4.3 included the following refinements:

Formal parameter relaxations The formal parameter `Sample` of the generic `Write_PNG_Type_x` procedures was relaxed. Previously this parameter was defined as:
`type Sample is range <>`
but this did not allow the actual to be a modular type. The new version defines the parameter as `(<>)` (discrete) thus allowing modular, integer, or enumerated types to be supplied as actuals. There is no impact on existing code, because integer types are still supported as actuals.

The formal parameter `Image_Handle` was also relaxed and now permits unconstrained formals. Again, this has no impact on existing code because a constrained formal (the previous requirement) is still permitted. Thanks to Samuel Tardieu for pointing out this possibility.

A similar change has been made to the input and output sample types in the two child packages:
`png_io.gamma_encoding` and
`png_io.standard_rgb_encodings`.
These parameters are now discrete rather than modular, permitting integer subtypes to be used (as a side-effect enumerated types are also permitted, although this is not likely to be useful). There is no impact on existing code because modular types will still be accepted and will give the same results.

Ravenscar profile A very small change to the package body has made the package compliant with the Ravenscar profile (see section D.13.1 of the LRM 2005). Thanks to Samuel Tardieu for pointing out that the package was almost Ravenscar compliant, and that a change was needed to the declaration of `Package_Identifier`.

User control over Zlib compression level An additional parameter has been added to the `Write_PNG_Type_x` procedures to permit the user control over the Zlib compression level. Of course, a default is provided that results in the same behaviour as before. A consequent change is that Zlib is now with'ed from the spec rather than the body of `png_io.adb` in order to allow visibility of some Zlib types and constants.

PNG_IO version 4.4 included the following improvements:

- When writing a PNG file, the package no longer uses a buffer to store all the uncompressed (but filtered) pixel data. Instead, each complete scan line is passed into Zlib for compression immediately after scan line filtering and any compressed data passed back from Zlib is immediately output to an IDAT chunk (with one exception - next bullet point). This improvement means that PNG file output is faster, and it uses less memory. In particular, for very large files where the buffer would have been the same size as the uncompressed pixel data, the memory usage for buffering is now drastically cut to a few kilobytes.
- The change to Zlib_Ada with version 4 introduced a small quirk - the first two bytes of Zlib compressed data (the header) were output in their own IDAT chunk. With the considerable tidying of the code resulting from using on-the-fly compression of each scan line (previous bullet point) it was simple to fix this so that the two bytes are now cached and added to

the front of the second buffer of data passed back from Zlib. The two header bytes are therefore now output in the first IDAT chunk, but this chunk no longer contains just the two header bytes.

- The program `png_compare` has been modified to set the exit status, so that if called from a script (e.g. under Unix/Linux/Mac OS X Darwin) a failure to compare exactly can be detected. This facilitates running a script to test PNG_IO on a large library of PNG test images.

PNG_IO version 4.5 was the first release for over two and a half years and included the following changes:

- The applicable license was updated to GPL v3. (See section 6).
- The license for this manual was updated to FDL v1.3. (See section C).
- Three new utility programs were included in the distribution. These are simple dump programs which output human readable information to standard output about:
  - The chunks contained in a PNG file (chunk name and size and offset from the start of the file).
  - The pixel data in the file *after decompression*, listed per pixel (not per byte of uncompressed pixel data) stored in the file, plus the palette data, if any.
  - The main properties of a PNG file, including the colour type, bit depth, presence of the main ancillary chunks such as gamma, chromaticity etc.

  These utility programs were written some years ago, but are now included because they may be useful to users, both as utilities and as examples.
- There are minor improvements such as writing the IEND chunk from a 12-byte constant rather than computing the CRC every time a file is written.
- A start has been made on rationalising functionality into a new child package `png_io.base`. This means that some low-level functionality is accessible to utility programs, and could also be accessed from extension packages in the future.
- CVS Id tags have been added to the source files so that files downloaded from the CVS repository can be identified.
- Detailed change histories have been removed from the source files: this information is kept in the CVS repository, and the use of Id tags obviates the need to edit change dates manually into the file.

PNG_IO version 4.5.1 was a minor release with minor updates to the manual including a new email address, and a small change to avoid an uninitialised variable warning at line 250 in `png_io-open.adb`. The release also included for the first time a bash script for running tests on the PngSuite of test images so that users can easily verify that PNG_IO will read all the test images correctly. This script was updated slightly with release 4.6.

B.5 Possible future changes

The current functions for reading pixel values pass only one pixel at a time, and have quite a high overhead due to the need to check the colour type. An improvement would be to pass a whole row of pixels from the image. This would require the pixel type to be known by PNG_IO (at present pixel values are passed as `Natural`). If this were to be implemented it would add additional functions, obviously retaining the current ones for backward compatibility.

Add support for the international text chunk. The package XML/Ada by Adacore now offers support for UTF-8 decoding. It is planned to add a child package to handle construction and decoding of this chunk using XML/Ada, but this is dependent on some prior cleanup of the chunk handling methods (see next paragraph).

Add code in the `Text_Chunk` function to write a compressed text chunk if the string is large, or consider adding a Boolean parameter (default False) to the spec, to control compression. This parameter should permit compression so that the default behaviour is as in previous versions, like this:

```
function Text_Chunk(Keyword, Text : String;
                    Permit_Compression : Boolean := False) return Chunk;
```

Rewrite the underlying handling of ancillary chunks so that the chunk is read from the file only when needed. At present a large amount of information is stored in the file descriptor. A more general approach would be to store only the offset to the appropriate chunk there, and read the data from the file if needed. This could apply, for example, to the gamma and physical chunks. Similarly, the current handling of unknown chunks is to store the whole chunk in a linked list, when all that is needed is the offset to the chunk in the file. Whether the CRC is verified on reading the file to open it, or only when the chunk is read, remains to be decided, but this would probably not impact user code.

The current implementation on reading a file requires the user code to read the pixels from PNG_IO. A possibly better method would be for PNG_IO to 'push' the pixels to the user code – this would require the user to instantiate a generic read procedure similar to the current generic write procedures, with a procedure parameter to be called by PNG_IO to pass pixels to the user's code. The benefit of this arrangement would be the removal of the uncompressed pixel data buffer from the internal descriptor used inside PNG_IO which would reduce the memory usage drastically. In order to support the previous method, the existing functions would be modified to read the IDAT data into a buffer on the first call. The transfer of pixel data would be initiated by the user program calling an instantiated read procedure, with pixels passed to the user code by callback of the generic procedure parameter. The existing method of 'pull' by the user would be supported by an internal buffer of pixel values, as now.

C Gnu Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple

HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution

and modification of the Modified Version to whoever possesses a copy
of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct
   from that of the Document, and from those of previous versions
   (which should, if there were any, be listed in the History section
   of the Document). You may use the same title as a previous version
   if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities
   responsible for authorship of the modifications in the Modified
   Version, together with at least five of the principal authors of the
   Document (all of its principal authors, if it has fewer than five),
   unless they release you from this requirement.
C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page. If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on. These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles. Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant. To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version. Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity. If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy. If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.3
> or any later version published by the Free Software Foundation;
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
> A copy of the license is included in the section entitled "GNU
> Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.