

GX Reference Manual

Passport Reader

```
Enter rows and columns for second matrix: ";
// Enter elements of first matrix.
cout << endl << "Enter elements of matrix 1:" << endl;
for(i = 0; i < r1; ++i)
    for(j = 0; j < c1; ++j)
    {
        cout << "Enter element a" << i + 1 << j + 1 << " : ";
        cin >> a[i][j];
    }

// Enter elements of second matrix.
cout << endl << "Enter elements of matrix 2:" << endl;
for(i = 0; i < r2; ++i)
    for(j = 0; j < c2; ++j)
    {
        cout << "Enter element b" << i + 1 << j + 1 << " : ";
        cin >> b[i][j];
    }

// Enter elements of m
```

PASSPORT READER GX

REFERENCE MANUAL

v. 7.2.10

Document version: 02.10.2023

Table of Contents

1. IMAGE HANDLING.....	3
1.1. MODULE PROPERTIES.....	5
1.2. PROPERTIES OF THE 'GXBMP' SUBMODULE	5
1.3. PROPERTIES OF THE 'GXJPEG8' AND 'GXJPEG12' SUBMODULES	6
1.4. PROPERTIES OF THE 'QXJPEG2K' SUBMODULE	9
1.5. PROPERTIES OF THE 'GXWSQ' MODULE.....	12
2. LOGGING.....	13
2.1. MODULE PROPERTIES.....	13
3. ABBREVIATIONS.....	16
CONTACT INFORMATION.....	17

1. IMAGE HANDLING

The gximage module provides an easy-to-use image manipulating library.

The GX system implements functions for:

- loading/saving images (from/to file or memory)
- converting images into different image formats
- mirroring an image (horizontal and/or vertical)
- rotating an image (by 0, 90, 180 and 270 degrees)
- handling 8/12 bit grayscale and 8/12 bit [RGB](#)/ [BGR](#) images
- zooming in/out an image (nearest neighborhood and linear interpolation algorithms)

The supported image formats are:

- [BMP](#): stores images using lossless or no compression. Supported BMP formats: MS Windows 3.0 BMP file format handling: 2 colors, 16 colors (RLE4 and no compression), 256 colors (RLE8 and no compression), 24 bits per pixel (RGB without compression) and read-only 16 and 32 bits per pixel modes.
- [JPEG](#): stores images using lossy compression. The gxjpeg module uses the IJG JPEG library (<http://www.jpeg.org>). The quality of the image can be set on a scale from 0 to 100, where 0 stands for the worst and 100 stands for the best quality. The GX system handles 8/12 bit grayscale and 8/12 bit (per color components) RGB / BGR images. Small messages/data (markers) can be included in the JPEG images. Every message has a type set by the application that created it (e.g., these kinds of markers are included in digital cameras to store information about lighting). The universally used marker code is GX_JPEGMARKER_COM. Other applications work in the range of GX_JPEGMARKER_APP0 to GX_JPEGMARKER_APP15. More than one marker with the same type can be placed in an image.
- JPEG-2000: stores pictures using lossless or lossy compression. The codec of the JPEG-2000 is based on wavelet/sub-band coding techniques. It handles both lossy and lossless compression of single-component and multicomponent image representations.
- The gxjpeg2k module uses the JasPer (<http://www.jpeg.org>) library and handles RGB and [YCbCr](#) color space with unsigned components. The module can write and read files in JP2 (JPEG-2000 file format) and JPC (JPEG-2000 code stream) formats. You can use it as an input and output method for the GX image with arbitrary pixel format.

- If you prefer the BGR order to RGB and you use the JPC format then the components of the image will be stored in reversed order. As the JPEG-2000 standard does not specify any means for encoding color space information in a JPEG-2000 code stream (JPC), the decoder has to make certain assumptions about the color space of an image. If accurate color representation is important, the use of the JPEG-2000 code stream (JPC) format is not advisable. The JPEG-2000 JP2 file format is recommended instead of it.
- [PNG](#): Stores images using lossless compression. The `gxpng` module uses the `libpng` (<http://www.libpng.org>) library. PNG provides a patent-free replacement for [GIF](#) and can also replace many common uses of [TIFF](#). Indexed-color, grayscale and true color images are supported, plus an optional alpha channel.
- [WSQ](#): Stores grayscale images using a lossy compression. This format is developed to store fingerprint images. The `gxwsq` module uses the Free FingerPrint Imaging Software Math Library (<http://ffpis.sourceforge.net>).



1.1. MODULE PROPERTIES

Zoom mode – Zoom type (integer)

```
<default>
  <zoom mode value="0"/>
</default>
```

It can be:

- 0: the zoom function uses the nearest neighborhood algorithm (faster than the linear interpolation algorithm)
- 1: the zoom function uses the linear interpolation algorithm (better than nearest neighborhood algorithm)

1.2. PROPERTIES OF THE 'GXBMP' SUBMODULE

The **gxbmp** module properties are located under the **bmp** node in the property tree.

```
<default>
  <bmp>
    <force24bpp value="0"/>
    <enablerle value="0"/>
  </default>
```

bmp/force24bpp - forces the 24 bit per pixel mode

It can be:

- 0: the indexed mode is used
- 1: the function uses 24 bit per pixel (RGB) mode

bmp/enablerle - enables the RLE compression

It can be:

- 0: disables the RLE compression
- 1: enables the RLE compression

1.3. PROPERTIES OF THE 'GXJPEG8' AND 'GXJPEG12' SUBMODULES

The GX system has two JPEG submodules:

- The **gxjpeg8** module handles the 8-bit (per color components) images.
- The **gxjpeg12** module handles the 12-bit (per color components) images.

The submodule properties are located under the **jpeg**, **jpeg8** or **jpeg12** nodes in the property tree.

If a property is not set in the jpeg8 or jpeg12 branches its value is set to the value specified in the jpeg branch.

```
<default>
  <jpeg>
    <decompress>
      <scale_num value="1"/>
      <scale_denom value="1"/>
      <dct_method value="0"/>
      <do_fancy_upsampling value="0"/>
      <do_block_smoothing value="0"/>
    <compress>
      <quality value="80"/>
      <progressive value="0"/>
      <dct_method value="0"/>
      <optimize_coding value="1"/>
      <restart_interval value="0"/>
      <restart_in_rows value="0"/>
      <smoothing_factor value="0"/>
    <jpeg8>
      ...
    <jpeg12>
      ...
  </default>
```

jpeg/decompress/scale_num - image scaling

According to the JPEG library documentation:

„Scale the image by the fraction scale_num/scale_denom. Default is 1/1, or no scaling. Currently, the only supported scaling ratios are 1/1. 1/2. 1/4 and 1/8. Smaller scaling ratios permit significantly faster decoding since fewer pixels need be processed and a simpler [IDCT](#) method can be used.”

jpeg/decompress/scale_denom - image scaling

See also: [jpeg/decompress/scale_num - image scaling](#)

jpeg/decompress/dct_method - DCT method

Selects the algorithm used for the [DCT](#) step.

It can be:

- 0: default method (JDCT_DEFAULT)
- 1: slow but accurate integer algorithm (JDCT_ISLOW)
- 2: faster, less accurate integer algorithm (JDCT_IFAST)
- 3: floating-point method (JDCT_FLOAT)

jpeg/decompress/do_fancy_upsampling - image sampling

It can be:

- 0: A faster but less accurate method is used.
- 1: More careful upsampling of the chromatic components.

The visible difference between the more careful and the faster method is often insignificant.

jpeg/decompress/do_block_smoothing - image smoothing

Interblock smoothing is applied or disabled.

It can be:

- 0: Disables interblock smoothing.
- 1: Interblock smoothing is applied in the early stages of decoding progressive JPEG files.

Early progression stages look 'fuzzy' with smoothing, 'blocky' without it. In any case, block smoothing ceases to be applied after the first few AC coefficients are known to full accuracy, so it is relevant only when using buffered-image mode for progressive images.

jpeg/compress/quality - compression quality

The quality of a JPEG image. 0 stands for the worst quality, 100 stands for the best. The default value is 80 in the GX system and 75 in the original JPEG library.

jpeg/compress/progressive - compression mode

It enables or disables creating progressive JPEG files.

- 0: disables progressive JPEG
- 1: enables progressive JPEG

jpeg/compress/dct_method - DCT method

Selects the algorithm used for the DCT step.

It can be:

- 0: default method (JDCT_DEFAULT)
- 1: slow but accurate integer algorithm (JDCT_ISLOW)
- 2: faster, less accurate integer algorithm (JDCT_IFAST)
- 3: floating-point method (JDCT_FLOAT)

jpeg/compress/optimize_coding - code optimization

If the value is true then the compressor generates optimal Huffman coding tables for the image.

It requires an extra pass over the data, so it costs a good amount of space and time:

- 0: the compressor uses the supplied or default Huffman tables
- 1: the compressor computes optimal Huffman coding tables

jpeg/compress/restart_interval - restarting interval

According to the JPEG documentation:

„To emit restart markers in the JPEG file, set one of these nonzero. Set `restart_interval` to specify the exact interval in MCU blocks. Set `restart_in_rows` to specify the interval in MCU rows. (If `restart_in_rows` is not 0, then `restart_interval` is set after the image width in MCUs is computed.) Defaults are zero (no restarts). One restart marker per MCU row is often a good choice.”

Note

„The overhead of restart markers is higher in grayscale JPEG files than in color files, and much higher in progressive JPEGs. If you use restarts, you may want to use larger intervals in those cases.”

jpeg/compress/restart_in_rows - restarting interval in rows

See also: [jpeg/compress/restart_interval - restarting interval](#)

jpeg/compress/smoothing_factor - compression smoothing factor

If non-zero, the input image is smoothed; the value varies between 1 and 100 (minimal and maximal smoothing).

1.4. PROPERTIES OF THE 'QXJPEG2K' SUBMODULE

The property section of the module is `jpeg2k` in the property tree.

```
<default>
  <jpeg2k>
    <imgareatlx value="-1"/>
    <imgareatly value="-1"/>
    <tilegrdtlx value="-1"/>
    <tilegrdtly value="-1"/>
    <tilewidth value="-1"/>
    <tileheight value="-1"/>
    <prcwidth value="32768"/>
    <prcheight value="32768"/>
    <cblkwidth value="64"/>
    <cblkheight value="64"/>
    <mode value="int"/>
    <rate value="-1"/>
    <ilyrrates value=""/>
    <prg value="0"/>
    <nomct value="0"/>
    <numrlvls value="6"/>
    <sop value="0"/>
    <eph value="0"/>
    <lazy value="0"/>
    <termall value="0"/>
    <segsym value="0"/>
    <vcausal value="0"/>
    <pterm value="0"/>
    <resetprob value="0"/>
    <numgbits value="2"/>
  </default>
```

`jpeg2k/imgareatlx` - X-coordinate of the image area.

Sets the x-coordinate of the top-left corner in the image area. If you use a negative value, the property will be ignored.

`jpeg2k/imgareatly` - Y-coordinate of the image area.

Sets the y-coordinate of the top-left corner in the image area. If you use a negative value, the property will be ignored.

`jpeg2k/tilegrdtlx` - X-coordinate of the tiling grid.

Sets the x-coordinate of the tiling grid's top-left corner. If you use a negative value, the property will be ignored.

jpeg2k/tilegrdtly - Y-coordinate of the tiling grid.

Sets the y-coordinate of the tiling grid's top-left corner. If you use a negative value, the property will be ignored.

jpeg2k/tilewidth - The nominal tile width.

Sets the nominal tile width. If you use a negative value, the property will be ignored.

jpeg2k/tileheight - The nominal tile height.

Sets the nominal tile height. If you use a negative value, the property will be ignored.

jpeg2k/prcwidth - The precinct width.

Sets the precinct width. The property must be an integer power of two. The default value is 32768. If you use a negative value, the property will be ignored.

jpeg2k/prcheight - The precinct height.

Sets the precinct height. The property must be an integer power of two. The default value is 32768. If you use a negative value, the property will be ignored.

peg2k/cblkwidth - The nominal code block width.

Sets the nominal code block width. The property must be an integer power of two. The default value is 64. If you use a negative value, the property will be ignored.

jpeg2k/cblkheight - The nominal code block height.

Sets the nominal code block height. The property must be an integer power of two. The default value is 64. If you use a negative value, the property will be ignored.

jpeg2k/mode - The coding mode.

Sets the coding mode. This property must have one of the following values:

- **int,0**: integer mode
- **real,1**: real mode

In case of lossless coding, the integer mode has to be used. By default, the integer mode is employed. The choice of mode also determines which multicomponent and wavelet transforms (if any) are used.

jpeg2k/rate - The target rate.

Specifies the target rate. The property is a positive real number. As the rate of one corresponds to no compression, explicit specification of a rate greater than one is never needed. By default, the target rate is considered to be infinite.

jpeg2k/ilyrrates - The rates for any intermediate layers.

Specifies the rates for any intermediate layers. The **ilyrrates** property to this option is a comma separated list of N rates. Each rate is a positive real number. The rates must increase monotonically. The last rate in the list should be less than or equal to the overall rate (as specified with the rate property).

jpeg2k/prg - Progression order.

Sets the progression order. This property must have one of the following values:

- **lrp,0**: layer-resolution-component-position progressive (e.g., adjustable rate)
- **rlcp,1**: resolution-layer-component-position progressive (e.g., adjustable resolution)
- **rpcl,2**: resolution-position-component-layer progressive
- **pcrl,3**: position-component-resolution-layer progressive
- **cpri,4**: component-position-resolution-layer progressive

By default, LRCP progressive ordering is employed.

Note

The RPCL and PCRL progressions are not valid for all possible image geometries.

jpeg2k/nomct - Disallows the use of any multicomponent transform.

Set to 1 to disallow the use of any multicomponent transform.

jpeg2k/numrlvs - Number of resolution levels.

Sets the number of resolution levels. It must be an integer that is greater than or equal to one. The default value is 6.

jpeg2k/sop - Generates SOP marker segments.

Set to 1 to generate SOP marker segments.

jpeg2k/eph - Generates EPH marker segments.

Set to 1 to generate EPH marker segments.

jpeg2k/lazy - Enables lazy coding mode.

Set to 1 to enable lazy coding mode (a.k.a. arithmetic coding bypass).

jpeg2k/termall - Terminates all coding passes.

Set to 1 to terminate all coding passes.

jpeg2k/segSYM - Uses segmentation symbols.

Set to 1 to use segmentation symbols.

jpeg2k/vcausal - Uses vertically stripe causal contexts.

Set to 1 to use vertically stripe causal contexts.

jpeg2k/pTerm - Uses predictable termination.

Set to 1 to use predictable termination.

jpeg2k/resetprob - Resets the probability models.

Set to 1 to reset the probability models after each coding pass.

jpeg2k/numgbits - Number of guard bits.

Set the number of guard bits (1 to 8). Default value is 2.

1.5. PROPERTIES OF THE 'GXWSQ' MODULE

wsq/bitrate – bitrate parameter of the wsq algorithm

Suggested values: 0.75 – 3.0

2. LOGGING

The gxlog module helps to use and to create log files for your GX based applications.

2.1. MODULE PROPERTIES

```
<default>
  <log>
    <ident value="my_application"/>
    <file value="mylogfile.txt"/>
    <format value="$Y-$o-$d $h:$m:$s ($l:$L) [$i] {$F:$I:$P} > $M\n"/>
    <filter value="5"/>
    <levels>
      <L0 value="NONE"/>
      <L1 value="EMER"/>
      <L2 value="FATL"/>
      <L3 value="ERRO"/>
      <L4 value="WARN"/>
      <L5 value="INFO"/>
      <L6 value="INF2"/>
      <L7 value="INF3"/>
      <L8 value="DBG1"/>
      <L9 value="DBG2"/>
    </levels>

  <maxfilesize value="10000"/>
  <maxfilenum value="5"/>
</log>
</default>
```

ident - identification string

Identification string of the application (e.g., 'MYAPPLICATION').

file - name of the log file

This property contains the name of the property file (e.g., 'mylog.txt'). If it is empty, the log module forwards the log messages to the system logger (syslogd on Linux and event logging system on Windows).

format - format string

You can define the line format of the log file. The logger reads it and replaces the specified arguments with the specified values.

The arguments can be:

- **\$y**: year (4 digits)
- **\$r**: year (2 digits)
- **\$o**: month (2 digits)
- **\$d**: day (2 digits)
- **\$h**: hour (2 digits)
- **\$m**: minute (2 digits)
- **\$s**: second (2 digits)
- **\$z**: milliseconds (3 digits)
- **\$l**: level (number)
- **\$L**: level (string)
- **\$i**: identification (string)
- **\$F**: name of the file (string)
- **\$l**: line in the file (number)
- **\$P**: name of the function (string)
- **\$M**: message (string)
- **\$p**: process id (8 digits hex number)
- **\$t**: thread id (8 digits hex number)
- **\$\$**: one dollar character (1 character)

filter - message filter

Messages are arranged into levels according to their importance (see the example below). The message filter value determines the maximum level of messages to be written to the log file. If it is 0 (GX_NONE), it disables logging.

levels - strings for level codes

Every message level has a description string. The logger writes it into the log file when the \$L is specified in the format string.

Example: Log levels.

```
<levels>
  <L0 value="NONE"/>
  <L1 value="EMER"/>
  <L2 value="FATL"/>
  <L3 value="ERRO"/>
  <L4 value="WARN"/>
  <L5 value="INFO"/>
  <L6 value="INF2"/>
  <L7 value="INF3"/>
  <L8 value="DBG1"/>
  <L9 value="DBG2"/>
</levels>
```

maxfilesize – maximal length of a log file

maxfilenum – maximum number of log files

The logger removes the last files when there are more files than the specified maxfilesize value.

Example:

```
logfile.txt, logfile.txt.1, logfile.txt.2, logfile.txt.3
```

If the **maxfilenum** value is 2 then the logger removes the `logfile.txt.2` and `logfile.txt.3` files and creates a new `logfile.txt`.

3. ABBREVIATIONS

BGR – Image pixel format where each pixel is composed by blue (B), green (G) and red (R) components

BMP – Image file format (Device Independent BitMaP)

DCT – Direct Cosine Transformation

GIF – Image file format (Graphics Interchange Format)

IDCT – Inverse Direct Cosine Transformation

JPEG – Image file format (Joint Photographic Expert Group)

PNG – Image file format (Portable Network Graphics)

RGB – Image pixel format where each pixel is composed by red (R), green (G) and blue (B) components

TIFF – Image file format (Tagged Image File Format)

WSQ – Image file format (Wavelet Scalar Quantization)

YCbCr – Image pixel format where each pixel is composed by the luminance (Y), and the chrominance (Cb), (Cr) components.

CONTACT INFORMATION

Headquarters:

Adaptive Recognition, Hungary Inc.
Alkotás utca 41 HU
1123 Budapest Hungary
Web: adaptiverecognition.com

Service Address:

Adaptive Recognition, Hungary Inc.
Ipari Park HRSZ1113/1 HU
2074 Perbál Hungary
Web: adaptiverecognition.com/support/

Adaptive Recognition Hungary Technical Support System (ATSS) is designed to provide you the fastest and most proficient assistance, so you can quickly get back to business.

Information regarding your hardware, latest software updates and manuals are easily accessible for customers via our [Documents Site \(www.adaptiverecognition.com/doc\)](http://www.adaptiverecognition.com/doc) after a quick registration.

New User

If this is your first online support request, please contact your sales representative to register you in our Support System. More help [here \(www.adaptiverecognition.com/support\)](http://www.adaptiverecognition.com/support)!

Returning User

All registered ATSS customers receive a personal access link via e-mail. If you previously received a confirmation message from ATSS, it contains the embedded link that allows you to securely enter the support site.

