MF1198-01b

EPSON

CMOS 32-BIT SINGLE CHIP MICROCOMPUTER **E0C33 Family**

JPEG33 MIDDLEWARE MANUAL

ENERGY SAVING EPSON

SEIKO EPSON CORPORATION

Preface

This manual is written for those who develop applications using the E0C33 Family of microcomputers. This manual explains the configuration and functionality of the JPEG33 image compression middleware for the E0C33 Family and how to use the software.

Table of Contents

1 Outline of the JPEG33 Middleware

JPEG33 is image compression middleware for the E0C33 Family. It is used to compress and expand images on the E0C33 Family chip. Its functions are supplied as library functions, which can be referenced after linking to the target program. High-level functions calling these functions are available as C source code, to facilitate image processing-related programming.
Tools supplied with the product allow you to create image ROM data for PCs, as well as Windows GUI tools for evaluation and examination of image compression and expansion on PCs.
The JPEG33 middleware is ideal for developing applications and devices such as PDAs, electronic stationery and toys.

1.1 Contents of the JPEG33 Package

The following provides the contents of the JPEG33 package. Confirm that all listed items are supplied with your package.

(1) Tool disk (CD-ROM for PC/AT)	1 disk
(2) E0C33 Family JPEG33 Middleware Manual (this manual)	1 copy each in English and Japanese
(3) Warranty card	1 card each in English and Japanese

1.2 Basic Configuration of the Image Input/Output System

The basic hardware configuration of the E0C33 system is shown in Figure 1.2.1 below. It consists mainly of the E0C33 chip, plus external memory and other peripheral circuits.

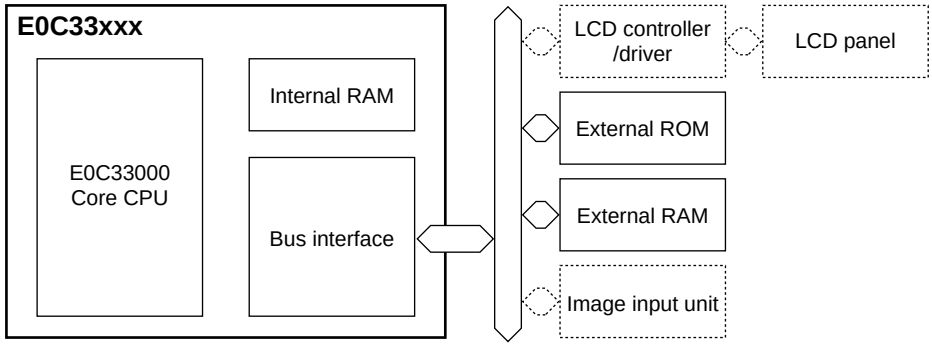


Figure 1.2.1 Hardware Configuration of the Image Input/Output System

The JPEG33 library is middleware located between the E0C33 hardware and the user program, with the primary purpose of performing image conversion-related processing. Loading the high-level functions supplied as C source files into or linking them to the user program allows image processing to be performed without having to call JPEG33 library functions directly from the user program.

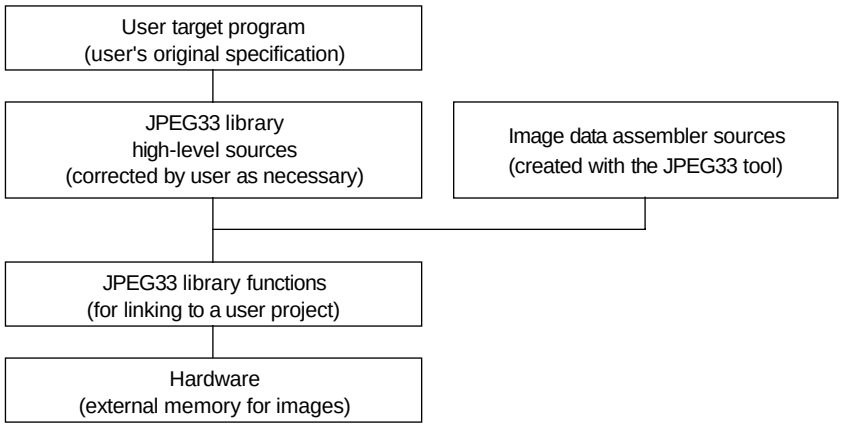


Figure 1.2.2 Software Configuration of the Image Input/Output System

For details on the JPEG33 library functions and high-level functions, see Chapter 5, "JPEG33 Library Reference".

1.3 JPEG33 Tools

JPEG33 tools create image ROM data to be written into the E0C33 chip, and evaluates image compression and expansion. All of these tools run in Windows 95 and Windows NT 4.0, as well as more recent Windows versions.

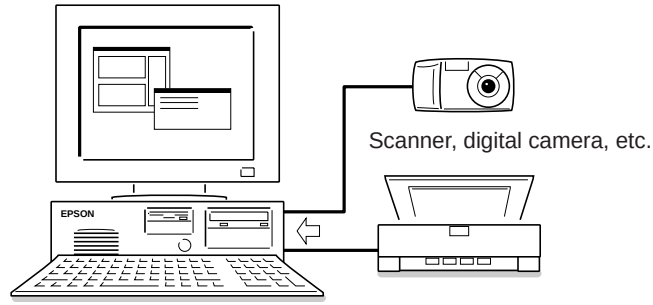


Figure 1.3.1 Hardware Configuration of the Image ROM Data Creation and Evaluation System

Image ROM data creation tools

The image ROM data creation tools consist of a series of programs that convert image files (from BMP to JPEG, RGB, GRY, or YUV) and generate assembly source files for the E0C33. These tools are used to create the display-only image data that is written into ROM. All of these programs are 32-bit applications that may be executed from the DOS prompt.

The image ROM data creation tools are listed in Table 1.3.1 below.

Table 1.3.1 Image ROM Data Creating Tools

Tools	Functions
bin2s.exe	Converts binary files (JPEG, RGB, YUV, or GRY files) into assembly source files.
bmp2gry.exe	Converts 24-bit BMP file into GRY files.
bmp2jpeg.exe	Converts 24-bit BMP file into JPEG files.
bmp2rgb.exe	Converts 24-bit BMP file into RGB files.
bmp2yuv.exe	Converts 24-bit BMP file into YUV files.
col2gry.exe	Converts 24-bit BMP file into grayscale 24-bit BMP files.
gry2bmp.exe	Converts GRY file into 24-bit BMP files.
jpeg2bmp.exe	Converts JPEG file into 24-bit BMP files.
rgb2bmp.exe	Converts RGB file into 24-bit BMP files.
yuv2bmp.exe	Converts YUV file into 24-bit BMP files.

Image compression/expansion evaluation tool (jb.exe)

The "JPEG33 Bench (jb.exe)" image compression/expansion evaluation tool is a 32-bit Windows GUI application, which allows you to set compression parameters, convert between BMP and JPEG files, convert images into grayscale BMP format files, and display original and the converted images, all in one window. Conversion of image files is done by calling the executable applications "bmp2jpeg.exe", "jpeg2bmp.exe", and "col2gry.exe".

For detailed information on the JPEG tools, see Chapter 4, "JPEG33 Tool Reference".

2 Installation

This chapter describes the operating environment of the JPEG33 tools and explains how to install the JPEG33 middleware.

2.1 Operating Environment

Software development with JPEG33 and image ROM data creation and evaluation tools requires the operating environment detailed below.

Personal computer

An IBM PC/AT or compatible hardware is required. We recommend a computer with at least a Pentium 90 MHz processor and 32MB of RAM.

Display

An SVGA display with 800 × 600 dot resolution is required. Choose the "small font" display option from the control panel.

Hard disk

The JPEG33 tools and JPEG33 library together require approximately 10MB of hard disk space.

CD-ROM drive

A CD-ROM drive is required to install the JPEG33 middleware.

Mouse

A mouse is necessary to perform actions within the compression/expansion evaluation application.

Video card

A video display card capable of 24-bit full color or 32-bit true color is recommended.

System software

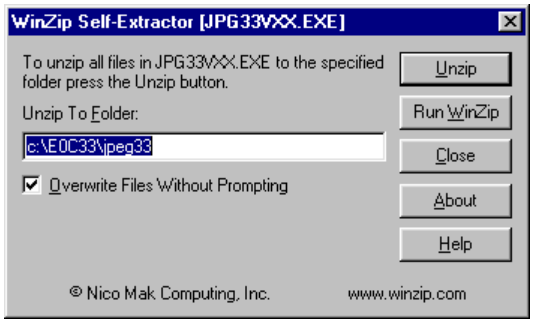
The JPEG33 tools support Microsoft® Windows® 95 and Windows NT® 4.0, or later versions of each (in Japanese and English).

Other

The "E0C33 Family C Compiler Package" is required for software development.

2.2 Installing the JPEG33 Middleware

The JPEG33 library and JPEG33 tools are supplied on a CD-ROM. Run the self-extracting file "jpg33vXX.exe" on the CD-ROM to install the JPEG33 library and JPEG33 tools. (The XX in the file name denotes a version number. For Version 1.0, for example, the file name is "jpg33v10.exe".)
When you double-click "jpg33v10.exe" to start it, the dialog box shown below appears.



In the text box, enter the path/folder in which to install the software, then click the [Unzip] button. The specified folder is created, and all files are copied to it. If the folder already exists in the specified path and [Overwrite Files Without Prompting] is checked (enabled), the folder is overwritten without prompting for confirmation.

The following shows the directory and file configuration after all files have been copied from the CD-ROM to your computer.

(root)\ (default: C:\E0C33\JPEG33\)

jpegtool		JPEG33 tool directory
	readme.txt	Supplementary explanations, etc. (in English)
	readmeja.txt	Supplementary explanations, etc. (in Japanese)
	bin	JPEG33 executable binary files
	jb.exe	JPEG33 Bench (GUI tool)
	bin2s.exe	Binary→CC33-assembler source conversion tool
	bmp2gry.exe	BMP→GRY format conversion tool
	bmp2jpeg.exe	BMP→JPEG format conversion tool
	bmp2rgb.exe	BMP→RGB format conversion tool
	bmp2yuv.exe	BMP→YUV format conversion tool
	col2gry.exe	BMP format grayscale conversion tool
	jpeg2bmp.exe	JPEG→BMP format conversion tool
	gry2bmp.exe	GRY→BMP format conversion tool
	rgb2bmp.exe	RGB→BMP format conversion tool
	yuv2bmp.exe	YUV→BMP format conversion tool
	gray16.tbl	16-level (4 bits per pixel) grayscale conversion table for the BMP format grayscale conversion tool
	gray256.tbl	256-level (8 bits per pixel) grayscale conversion table for the BMP format grayscale conversion tool
	vb40032.dll, olepro32.dll, msvcrt40.dll	.dll for the JPEG33 Bench
	sample	Sample directory Image samples, image format conversion batch files, and make files (For details on the configuration and content of sample files, refer to "readme.txt" or "readmeja.txt" in "jpegtool\".)

jpeglib		JPEG33 library files
	readme.txt	Supplementary explanations, etc. (in English)
	readmeja.txt	Supplementary explanations, etc. (in Japanese)
	lib	JPEG33 library directory
	jpeg.lib	JPEG33 library
	conv_yuv.o	Object file (RGB↔YUV image format conversion)
	jfdctin.o	Object file (DCT conversion)
	imemin.o	Object file (internal buffer RAM declaration)
	jstripin.o	Object file (image sampling)
		(These object files may be copied to internal RAM for faster execution. See Section 5.6, "Techniques for Improving Processing Speed".)
	include	Include file directory of JPEG33 library
	jpegtop.h	High-level function header file
	src	Public source directory of high-level function
	jpegtop.c	High-level function C source file of JPEG33 library
	libsrc	Public source directory for JPEG33 library
	conv	Image format conversion function source directory
	conv_yuv.c	RGB↔YUV image format conversion C source file
	conv_yuv.h	RGB↔YUV image format conversion C source file
	conv_y.c	RGB↔Y image format conversion C source file
	conv_y.h	RGB↔Y image format conversion C source file
	conv_4g.c	Y↔4-bit grayscale image format conversion C source file
	conv_4g.h	Y↔4-bit grayscale image format conversion header file
	conv_8g.c	Y↔8-bit grayscale image format conversion C source file
	conv_8g.h	Y↔8-bit grayscale image format conversion header file
	resizer	Image resize function source directory
	resizer.c	Color pixel resize C source file
	resizer.h	Color pixel resize header file
	resizgry.c	Grayscale pixel resize C source file
	resizgry.h	Grayscale pixel resize header file
	demo1	Demonstration program directory
	demo2	Each directory (demoX) contains sample programs to help you get the
	:	most from the JPEG33 library. For configuration and contents of the
	demoN	demonstration programs, refer to "readme.txt" or "readmeja.txt" in "jpeglib\".

The directory structure may be altered, but the explanations given in this manual assume the above directory structure.

3 Software Development Procedure

This chapter describes the procedure for developing software to process images on the E0C33 Family chip. The basic flow of development is shown below.

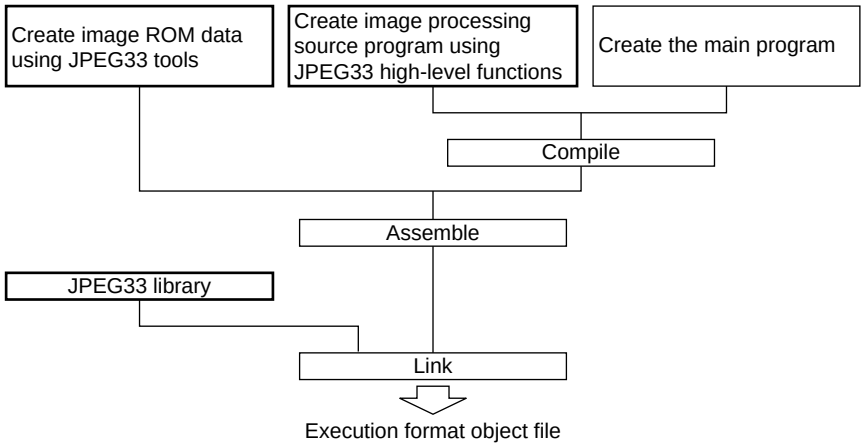


Figure 3.1 E0C33 Image Processing Software Development Procedure

1) Before writing image data into ROM, create an assembly source file for the image ROM data using the JPEG33 tools.
2) Create a user program. For image processing, use the high-level functions provided in the JPEG33 library. The source file for image ROM data created above can be included in the user program source file.
3) Compile and assemble the source program.
4) Link the object files generated in item 3 with the JPEG33 library. This creates an executable object file.

3.1 Image File Formats Handled by JPEG33 Tools and Library

The JPEG33 tools and library handle image files created in the following formats.

BMP format file

This is the standard Windows bitmap format file. The JPEG33 tools use image files in this format as source files when converting image formats. To prepare source files, create image data as 24-bit BMP files.

RGB format file

This data format provides information on color elements R (red), G (green), and B (blue) for each pixel. The RGB format files used by the JPEG33 tools and library provide 8-bit data for R, G, and B values, with one pixel represented by 3 bytes. Approximately 167,000 colors may be expressed, with (R, G, B) values of (0, 0, 0) indicating black, (255, 255, 255) indicating white, and (255, 0, 0) indicating the brightest red, for example. For details of file structure, see Section 5.4, "Image Data Structure".

YUV format file

This data format expresses the color of each pixel by values for Y (luminance), U (red chroma), and V (blue chroma). Because the human eye is insensitive to chromaticity, this format offers the advantage of easy data compression, although it requires RGB conversion for output to a PC color monitor. For details of file structure, see Section 5.4, "Image Data Structure".

GRY format file

This is the data file for a grayscale image, without color information. The JPEG33 tools and library support two types of grayscale files: a 256-level grayscale file, in which each pixel is represented by 8-bit luminance information; and a 16-level grayscale file, in which each pixel is represented by 4 bits. For file structure details, see Section 5.4, "Image Data Structure".

JPEG format file

This file consists of JPEG-compressed image data.

The JPEG33 tools convert the 24-bit BMP format into the RGB, YUV, GRY, or JPEG formats on a PC to generate an assembly source file for the image data to be written to the E0C33 system ROM. The JPEG33 library creates on the E0C33 chip JPEG compression with YUV format data interleaving and picture quality index values specified. Additionally, when expanding the compressed data, it allows you to specify the roughness of the reduced expansion and the pixel magnification values, enabling reduced display and pseudo-progressive rendering. RGB↔YUV and RGB/GRY↔Y conversion functions are also supported.

3.2 Creating Image ROM Data Using JPEG33 Tools

When the image data to be displayed on the E0C33 chip needs to be prepared in advance, create the data using the JPEG33 tools.
Figure 3.2.1 shows the procedure for creating image ROM data and the configuration of JPEG33 tools.

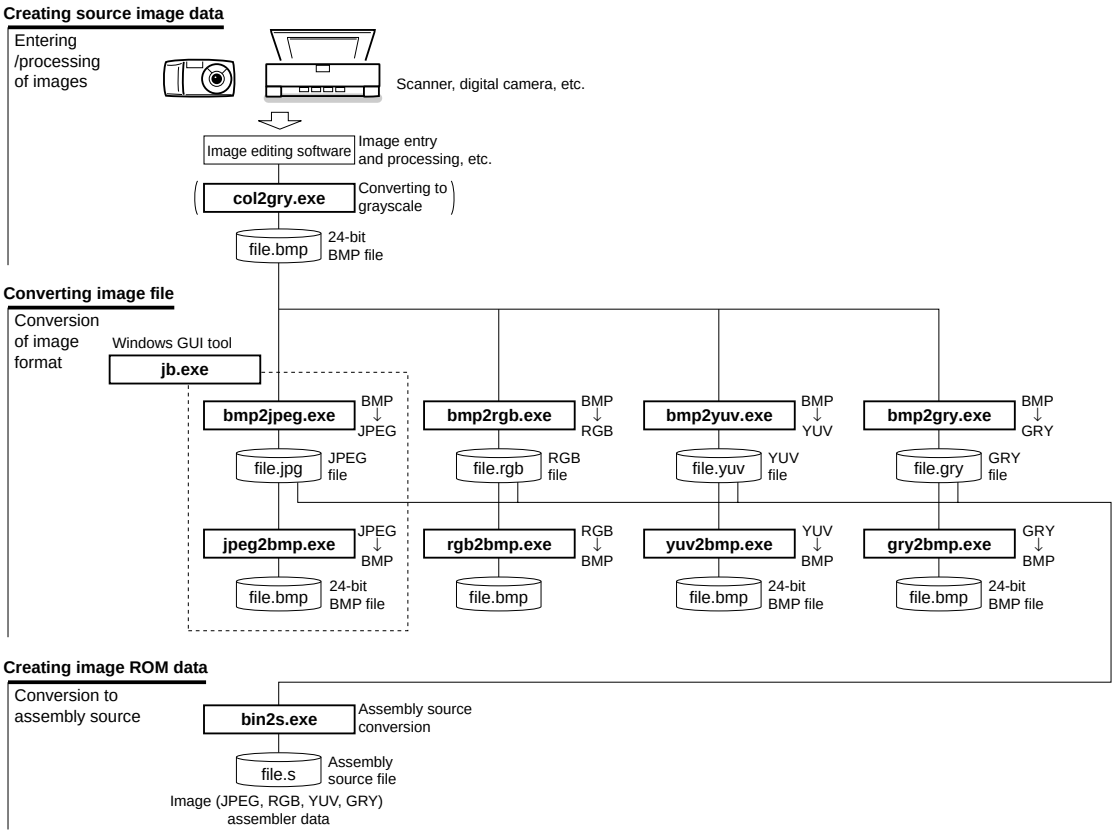


Figure 3.2.1 Flow Chart for Creating Image ROM Data

This chapter provides an outline of JPEG33 tool operations. For detailed information, please refer to Chapter 4, "JPEG33 Tool Reference".
In the following explanations, we use sample files in the "jpegtool\sample\" directory. We also assume that "jpegtool\sample\" is the current directory, and that PATH is set to the "jpegtool\bin\" directory.

Example:
```
DOS>CD c:\e0c33\jpeg33\jpegtool\sample
DOS>PATH c:\e0c33\jpeg33\jpegtool\bin
```

3.2.1 Preparing Image Data

Prepare the image you want to be written to ROM, using a scanner or digital camera or an image database. To capture images, use commercially-available image processing software and save the captured image to a 24-bit BMP file.

Notes:
- Always choose the 24-bit BMP format to save images. The JPEG33 tools cannot process 256-color or 16-color BMP format files.
- When using commercially-available photographs or images, first confirm that they are not copyrighted, or arrange for permission to use if they are copyrighted.

Sample 24-bit BMP format files are provided in the "jpegtool\sample\" directory. Use these sample files to test the JPEG33 tools.

3.2.2 Converting Image Formats

Use the JPEG33 tools to convert the prepared 24-bit BMP file. Conversion tools and batch files provided in the "jpegtool\sample\" directory are listed in the table below.

Table 3.2.1 BMP to Other Format Conversion Tools

Nature of Conversion	JPEG33 Tool	Batch File
BMP→RGB	bmp2rgb.exe	bmptorgb.bat
BMP→YUV	bmp2yuv.exe	bmptoyuv.bat
BMP→GRY	bmp2gry.exe	bmptogry.bat
BMP→JPEG	bmp2jpeg.exe	bmptojpg.bat
BMP(color)→BMP(gray)	col2gry.exe	–

After files are converted into RGB, YUV, or GRY formats, they cannot be displayed on a PC. You can also use various tools to convert each format into the BMP format to check the results of pseudo-progressive expansion of JPEG data.

Table 3.2.2 Other Formats to BMP Conversion Tools

Nature of Conversion	JPEG33 Tool	Batch File
RGB→BMP	rgb2bmp.exe	rgbtobmp.bat
YUV→BMP	yuv2bmp.exe	yuvtobmp.bat
GRY→BMP	gry2bmp.exe	grytobmp.bat
JPEG→BMP	jpeg2bmp.exe	jpgtobmp.bat

Run these tools from the DOS prompt.

Example: BMP→JPEG conversion, in which a BMP file is JPEG-compressed with no specified options

```
>bmp2jpeg sample.bmp sample.jpg
>bmptojpg sample
```

JPEG→BMP conversion, in which a JPEG image is expanded with x4 roughness (in 1/4 size) to generate a BMP format file with pixels magnified by a factor of four. Pseudo-progressive display in the same size as the original image is possible.

```
>jpeg2bmp -r 22 sample.jpg test.bmp
```

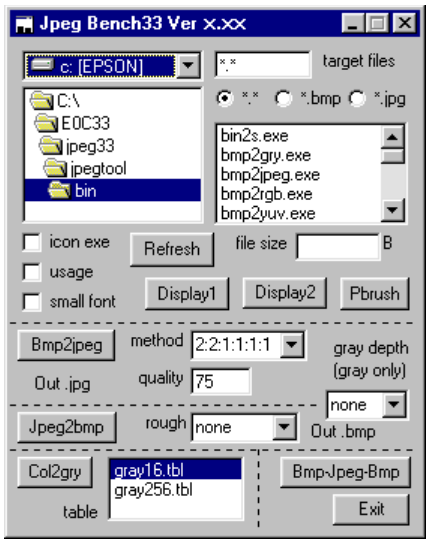
For details on each tool and batch files, see Chapter 4, "JPEG33 Tool Reference".

3.2.3 Evaluating JPEG Compression/Expansion

The "JPEG33 Bench (jb.exe)" Windows GUI tool allows you to set compression parameters, convert between BMP and JPEG files, convert images into grayscale BMP files, and display the original and the converted images, all in one window.
The following shows the basic operation procedure for compressing and expanding color images using the JPEG33 Bench. For detailed information on the tool's operation panel and grayscale conversion, see Section 4.3, "Image Compression/Expansion Evaluating Tool".

(1) Starting the JPEG33 Bench

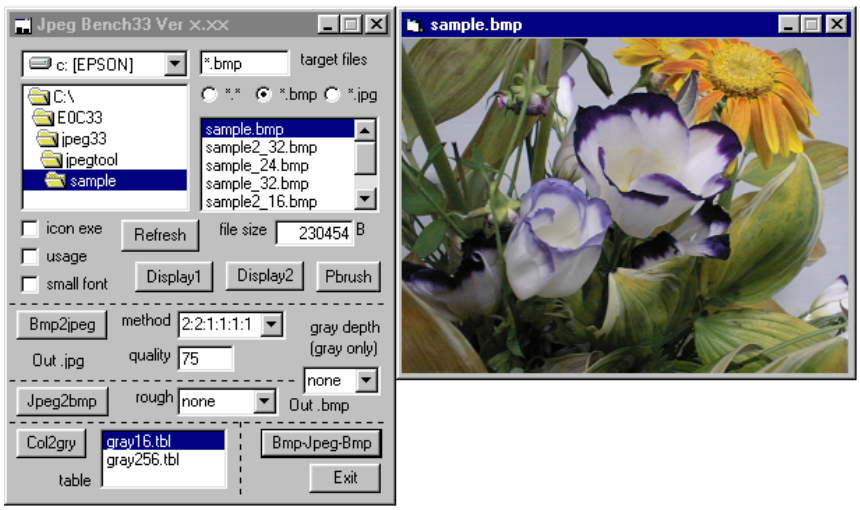
Double-click the "jb.exe" icon to start JPEG33 Bench.
When JPEG33 Bench starts, it opens the [Jpeg Bench33] window.
To quit JPEG33 Bench, click the [Exit] button in the [Jpeg Bench33] window.



[Jpeg Bench33] window

(2) Choosing image data

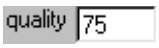
Choose the BMP format file you want to convert.
For example, choose "jpegtool\sample\" from the directory list box and double-click on "sample.bmp" in the file list box. Image display window 1 will open, displaying the selected image. Or you can click the [Display] button after choosing a file name to display the selected image in the same way.



(3) JPEG compression

First, set the interleave and picture quality index values that determine the image compression ratio.

Choose the interleave value (Yh:Yv:Uh:Uv:Vh:Vv) for UV (chroma) data. The Y (luminance) data is not interleaved.
If you select 1:1:1:1:1:1, the UV data is not interleaved during compression. If you select 2:1:1:1:1:1, the UV data is thinned out to 1/2 in the horizontal direction only. If you select 2:2:1:1:1:1, the UV data is thinned out to 1/2 in both horizontal and vertical directions.
For most cases, the default value 2:2:1:1:1:1 is recommended.
Choose "gray" for grayscale images.



Enter a decimal value from 1 to 100 for the picture quality index value. This setting significantly affects compression ratio and picture quality. Smaller values produce smaller file size following conversion to JPEG format, but produce poorer picture quality as well. The default value is 75.

After setting the above values, click the [Bmp2jpeg] button to run "bmp2jpeg.exe". The image data is JPEG-compressed. When "sample.bmp" is selected as the image source file, a JPEG file with the name "sample.jpg2" is generated within the "sample\" directory.
The compression effects of "sample.bmp" for various interleave and picture quality index values set are listed in the table below. Since compression ratios vary with each image, so the values below are shown for reference only.

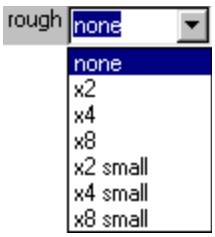
Table 3.2.3 Sizes of "sample.jpg2" ("sample.bmp" = 230,454 bytes)

Picture quality index value		1	25	50	75	100
Interleave value	2:2:1:1:1:1	2,260 (1%)	7,736 (3.4%)	11,672 (5.1%)	17,346 (7.5%)	79,666 (35%)
	2:1:1:1:1:1	2,571 (1.1%)	8,334 (3.6%)	12,662 (5.5%)	18,917 (8.2%)	99,386 (43%)
	1:1:1:1:1:1	3,186 (1.4%)	9,410 (4.1%)	14,353 (6.2%)	21,803 (9.4%)	137,482 (60%)
Picture quality		Poor←				→Good

Numeric values indicate bytes; Numbers in ( ) show approximate compression ratios.

(4) JPEG expansion

For ordinary expansion, choose the JPEG format file (sample.jpg2) and click the [Jpeg2bmp] button after setting the option "rough" to "none". "jpeg2bmp.exe" starts and expands the JPEG file. When "sample.jpg2" is expanded, a BMP format file with the name "sample.bmp2" is generated. Selecting the generated file and clicking the [Display2] (or [Display1]) button opens image display window 2 (or image display window 1) and displays the expanded image.
The following options are available for JPEG expansion:

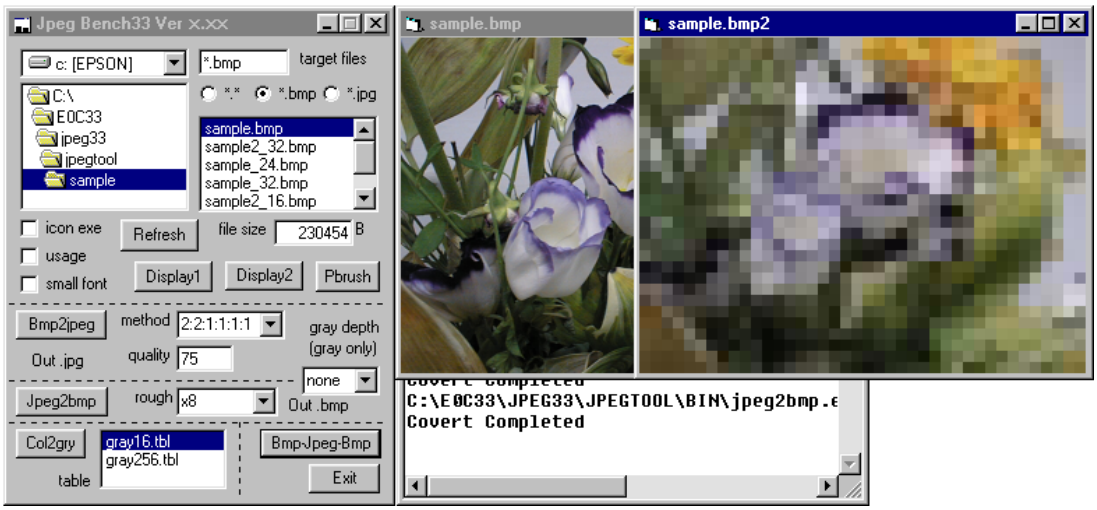


Selects the pixel magnification value and the roughness of expansion. Each current selected value is shown below.
none: Not magnified, no roughness
x2: Magnified 2-fold, x2 roughness
x4: Magnified 4-fold, x4 roughness
x8: Magnified 8-fold, x8 roughness
x2 small: Not magnified, x2 roughness
x4 small: Not magnified, x4 roughness
x8 small: Not magnified, x8 roughness
The default value is "none".

If you select the x2, x4, or x8 option, the image data is reduction-expanded with the specified roughness, resulting in the pixels being magnified by that factor. Consequently, image data is expanded to the same size as the original into a mosaic image having the roughness of the specified magnification, enabling pseudo-progressive rendering.
If you select x2 small, x4 small, or x8 small, images are reduced by thinning out pixels to 1/2, 1/4, or 1/8 so that the original image is reduced as the image data is expanded. These options can be used to display thumbnails of images.

(5) Batch processing of JPEG compression and expansion

The JPEG33 Bench can perform image batch-processing by executing "bmp2jpeg.exe" and "jpeg2bmp.exe" in succession. Choose the source image and the desired options for each tool, then click the [Bmp-Jpeg-Bmp] button. After executing image compression and expansion processing, the software displays the precompression image in image display window 1 and the post-expansion image in image display window 2.



3.2.4 Converting Image Data into Assembly Source Files

To load the created image data (JPEG, RGB, YUV, or GRY format) into or to link to the user program, generate an assembly source file for the E0C33 Assembler, using "bin2s.exe".

Example: `>bin2s sample.jpg > sample_jpg.s` (Using DOS's redirect function)

In this example, the JPEG file "sample.jpg" is converted into the assembly source file "sample_jpg.s". The file "sample_jpg.s" is generated using the input file name "sample" as a global symbol. (The symbol name can be modified using the "-l *symbol*" option of "bin2s.exe".

Contents of "sample_jpg.s"

```
        .global sample
        .align  2
sample:
        .byte   0xff 0xd8 0xff 0xdb 0x00 0x84 0x00 0x08
        .byte   0x06 0x06 0x07 0x06 0x05 0x08 0x07 0x07
        .byte   0x07 0x09 0x09 0x08 0x0a 0x0c 0x14 0x0d
                                :
; total 17346 bytes data
```

3.2.5 Precautions for Creating Image ROM Data

- When "bmp2yuv.exe" and "yuv2bmp.exe" or "bmp2jpeg.exe" and "jpeg2bmp.exe" are executed repeatedly by alternating each operation, picture quality may degrade as a result of calculation errors occurring during conversion. Avoid running these tools repeatedly.
- Always confirm that the original image data is prepared in 24-bit BMP format. No 256-color or 16-color BMP format files can be processed by the JPEG33 tools.

3.3 Creating a User Program and Linking the JPEG33 Library

Image compression and expansion on the E0C33 chip can be realized by calling JPEG33 library functions. This software package contains the high-level function C source file "jpegtop.c", as well as low-level library object files. Install these files into the user program to enable easy creation of JPEG compression and expansion routines.
For details of high-level functions and the JPEG33 library, see Chapter 5, "JPEG33 Library Reference". Sample programs are provided in the directories "jpeglib\demo1\", "jpeglib\demo2\", and "jpeglib\demo3\" for reference.
The created image ROM data also needs to be incorporated into the user program, or otherwise linked with the JPEG33 library after assembly.

Example of link command file (\jpeg33\jpeglib\demo1\demo1.cm)

```
;Map set
-code 0x0601000                          ; set relative code section start address
-code 0x0600000 {vector.o boot.o}        ; set code sections to absolute address
-data 0x0660000 {vector.o boot.o}        ; set data sections to absolute address
-bss  0x0670000 {demo1.o jpegtop.o ..\lib\jpeg.lib:jmemory.o}
                                         ; set relative bss section start address
;Library path
-l c:\cc33\lib
-l ..\lib                                ; JPEG33 libraly path

;Executable file
-o demo1.srf

;Object files
vector.o
boot.o
data1.o                                  ; Image data file
jpegtop.o
demo1.o

;Library files
jpeg.lib                                 ; JPEG33 library
string.lib
idiv.lib
```

For high-speed operation, several object files in the JPEG library can be copied to internal RAM. For details, see Section 5.6, "Techniques for Improving Processing Speed".

4 JPEG33 Tool Reference

This chapter explains the function and use of each JPEG33 tool.

4.1 Outline of JPEG33 Tools

The JPEG33 tools are software used to create image ROM data to be written to an E0C33 Family chip, and to evaluate image compression and expansion. All the tools run under Windows 95, Windows NT 4.0, as well as more recent Windows versions. (For information on the operating environment, see Section 2.1, "Operating Environment".)

The files associated with the JPEG33 tools are located in the "jpegtool" folder (directory). The configuration of JPEG33 tools and procedures for creating image ROM data are shown in Figure 4.1.1.

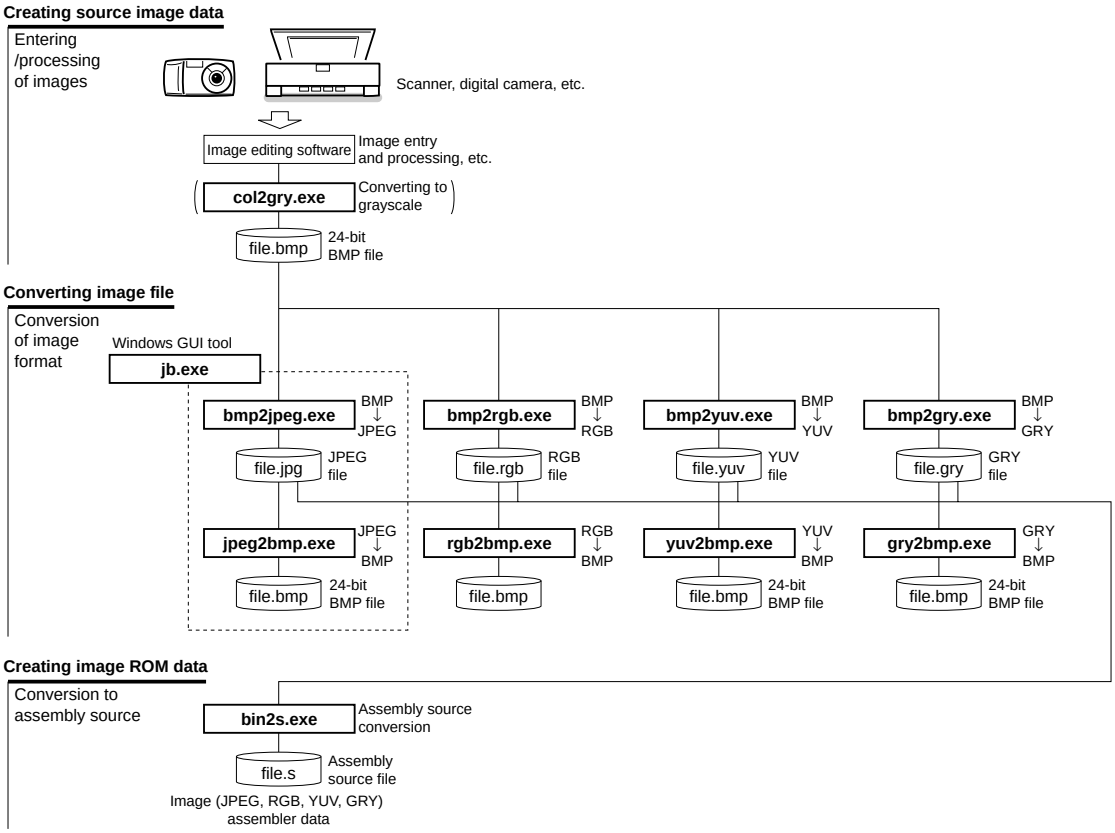


Figure 4.1.1 Flow Chart for Creating Image ROM Data

Image ROM data creation tools

The image ROM data creation tools consist of a series of programs that convert image files (from BMP to JPEG, RGB, YUV, or GRY) and generate assembly source files for the E0C33. All the programs are 32-bit applications executable from the DOS prompt. They can also be used from batch files or make files.
The image ROM data creation tools are listed in Table 4.1.1 below.

Table 4.1.1 Image ROM Data Creating Tools

Tools	Functions
bin2s.exe	Converts binary files (JPEG, RGB, YUV, or GRY files) into assembly source files.
bmp2gry.exe	Converts 24-bit BMP file into GRY files.
bmp2jpeg.exe	Converts 24-bit BMP file into JPEG files.
bmp2rgb.exe	Converts 24-bit BMP file into RGB files.
bmp2yuv.exe	Converts 24-bit BMP file into YUV files.
col2gry.exe	Converts 24-bit BMP file into grayscale 24-bit BMP files.
gry2bmp.exe	Converts GRY file into 24-bit BMP files.
jpeg2bmp.exe	Converts JPEG file into 24-bit BMP files.
rgb2bmp.exe	Converts RGB file into 24-bit BMP files.
yuv2bmp.exe	Converts YUV file into 24-bit BMP files.

The JPEG33 tools use the same algorithm used by the library to write to the E0C33 chip.
Since files in RGB, YUV, and GRY formats cannot be displayed directly on the PC screen, use "rgb2bmp.exe", "yuv2bmp.exe" and "gry2bmp.exe" to verify the image in each format.

Notes:
- BMP files converted with "yuv2bmp.exe" may not match the original BMP format files, depending on calculation results. Using "gry2bmp.exe" will result in grayscale images.
- If "bmp2yuv.exe" and "yuv2bmp.exe" or "bmp2jpeg.exe" and "jpeg2bmp.exe" are repeatedly used in alternation, picture quality may quickly degrade as a result of calculation errors made during conversion. Avoid such application of these tools.
- Only JPEG images resulting from conversion with "bmp2jpeg.exe" (the BMP→JPEG format conversion tool for the CC33) may be operated by the JPEG→BMP format conversion tool.

Image compression/expansion evaluation tool (jb.exe)

The "JPEG33 Bench (jb.exe)" image compression/expansion evaluation tool is a 32-bit Windows GUI application. It allows you to set compression parameters, convert between BMP and JPEG files, convert images into grayscale BMP format files, and display original and converted images, all in one window. Image files are converted by calls to "bmp2jpeg.exe", "jpeg2bmp.exe" and "col2gry.exe".

4.2 Image ROM Data Creation Tools

This section describes how to use each image-ROM data creation tool and their various functions.
Run each tool from the DOS prompt. If you do not specify command line parameters when launching a tool, the software displays "Usage". When a tool terminates properly, the software displays an execution message. If an error occurs, the software displays an error message and does not generate an output file.
In command line explanations, options enclosed in brackets [ ] may be omitted. The parameters indicated in *italics* require values or filenames.

Notes:
- The following limitations apply to filenames specified in each tool.
  Filename: 32 characters or less
  Valid characters: a to z, A to Z, 0 to 9, _, and .
- A space or tab is required between options or parameters.

4.2.1 bmp2jpeg.exe

Function: Converts a BMP file into a JPEG file.

Usage: `DOS>bmp2jpeg [options] infile.bmp outfile.jpg↵`

Arguments:

infile.bmp	Input filename (BMP file)
outfile.jpg	Output filename (JPEG file)

Options:

-d *value* Specifies a value for grayscale levels.
value = 4: 16 grayscale levels (4 bits per pixel)
value = 8: 256 grayscale levels (8 bits per pixel)
This option is effective only if you specify the value 0 (grayscale) in the -i option (specification of interleave value). Otherwise, it is ignored. The default value is 8. Images are converted to 256-grayscale level data when the option is left blank after grayscale is specified in the -i option.

-i *value* Specifies the interleave value.
value = 0: Grayscale (Yh:Yv = 1:1)
value = 1: Color, with UV data not interleaved.
(Yh:Yv:Uh:Uv:Vh:Vv = 1:1:1:1:1:1)
value = 2: Color, with the UV data thinned to 1/2 horizontally.
(Yh:Yv:Uh:Uv:Vh:Vv = 2:1:1:1:1:1)
value = 3: Color, with the UV data thinned 1/2 horizontally and vertically.
(Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1)
The default value assumed if this option is left blank is 3 (color, with UV data thinned 1/2 horizontally and vertically).

-q *value* Specifies the picture quality index value.
1≤value≤100 (in decimal)
Smaller correspond to smaller file sizes following conversion to JPEG format, but picture quality will be poor.
The default value is 75 when this option is left blank.

Example:
```
DOS>bmp2jpeg -i 1 -q 80 sample.bmp sample.jpg
```

Precautions:

- This tool can handle images with a maximum width and height of 32,767 pixels in each dimension.
- When compressing grayscale images, this tool first converts the RGB data for the 24-bit BMP file into grayscale data, using the equation below:
  ((R + G + B) * 21845 + 32768) / 65536
  (Although this averages R, G, and B, it converts floating-point values into fixed decimal values for integer arithmetic.)
  After conversion of this data into unsigned character type, the 4 low-order bits are truncated, and 0x08 is added for 16-level (4 bits per pixel) grayscale. The data takes on 16 values, 0xN8 (N = 0 to f). For 256-level (8 bits per pixel) grayscale, the low-order bits are not truncated, and the offset value is 0. The data assumes 256 values from 0x00 to 0xff.

4.2.2 jpeg2bmp.exe

Function: Converts a JPEG file into a BMP file.

Usage: DOS>**jpeg2bmp [options]** ***infile.jpg outfile.bmp***↵

Arguments:
- ***infile.jpg*** Input filename (JPEG file)
- ***outfile.bmp*** Output filename (BMP file)

Options:
- **-d *value*** Specifies a value for grayscale levels.
  value = 4: 16 grayscale levels (4 bits per pixel)
  value = 8: 256 grayscale levels (8 bits per pixel)
  This option affects grayscale images only. For color image conversions, this option is ignored.
  The default value is 8. If the option is left blank for a grayscale conversion, the image is converted to 256-grayscale level data.
- **-r *value*** Specifies the roughness of expansion and magnification value.
  value is a 1-byte hexadecimal number, with the 4 low-order bits specifying the roughness of expansion and the 4 high-order bits specifying the magnification value.
  value = magnification value OR roughness of expansion = 00 to 33
  Roughness of expansion: 0 = none, 1 = 2-fold, 2 = 4-fold, 3 = 8-fold
  Magnification value: 0 =none, 10 = 2-fold, 20 = 4-fold, 30 = 8-fold
  The default value is 0. If this option is left blank, the roughness of expansion and the magnification value are both assumed to be 0 as data is processed.
  Varying the roughness of expansion and magnification values while expanding images enables you to evaluate the picture quality of pseudo-progressively expanded images.

Example:
```
DOS>jpeg2bmp -r 11 sample.jpg test.bmp
```

Precautions:
- When 16-level (4 bits per pixel) grayscale images are expanded, the 4 low-order bits are truncated, and an offset 0x08 is added to the resulting value. For 256-level (8 bits per pixel) grayscale images, no truncation occurs, and the offset value is 0. Conversion to 16-level (4 bits per pixel) grayscale images occurs in the following manner.
  0x00–0x0f→0x08, 0x10–0x1f →0x18, 0x20–0x2f→0x28, 0x30–0x3f→0x38
  0x40–0x4f→0x48, 0x50–0x5f →0x58, 0x60–0x6f→0x68, 0x70–0x7f→0x78
  0x80–0x8f →0x88, 0x90–0x9f→0x98, 0xa0–0xaf→0xa8, 0xb0–0xbf→0xb8
  0xc0–0xcf→0xc8, 0xd0–0xdf→0xd8, 0xe0–0xef→0xe8, 0xf0–0xff→0xf8
  For 256-level (8 bits per pixel) grayscale images, the expanded value is output directly as obtained.
- When rough-expanding a JPEG image, the width and height of the expanded image are derived by shifting the bits right by the roughness value after first subtracting 1 from width and height, with fractions rounded up. The equations are given below:
  Width of expanded image
  = ((width of image before expansion) – 1) >> (roughness of expansion value) + 1
  Height of expanded image
  = ((height of image before expansion) – 1) >> (roughness of expansion value) + 1
  Magnification is performed simply by magnifying the expanded image. If the image is magnified after rough-expansion, the original image size and the expanded size may differ due to rounding of fractions.
- This tool can handle images with a maximum width and height of 32,767 pixels in each dimension.

4.2.3 bmp2rgb.exe

Function: Converts BMP files into RGB files.

Usage: DOS>**bmp2rgb** ***infile.bmp outfile.rgb***↵

Arguments: ***infile.bmp*** Input filename (BMP file)
outfile.rgb Output filename (RGB file)

Example:
```
DOS>bmp2rgb sample.bmp sample.rgb
```

4.2.4 rgb2bmp.exe

Function: Converts RGB files into BMP files.

Usage: DOS>**rgb2bmp** ***infile.rgb outfile.bmp***↵

Arguments: ***infile.rgb*** Input filename (RGB file)
outfile.bmp Output filename (BMP file)

Example:
```
DOS>rgb2bmp sample.rgb test.bmp
```

4.2.5 bmp2yuv.exe

Function: Converts BMP files into YUV files.

Usage: DOS>**bmp2rgb** ***infile.bmp outfile.yuv***↵

Arguments: ***infile.bmp*** Input filename (BMP file)
outfile.yuv Output filename (YUV file)

Example:
```
DOS>bmp2yuv sample.bmp sample.yuv
```

4.2.6 yuv2bmp.exe

Function: Converts YUV files into BMP files.

Usage: DOS>**yuv2bmp** ***infile.yuv outfile.bmp***↵

Arguments: ***infile.yuv*** Input filename (YUV file)
outfile.bmp Output filename (BMP file)

Example:
```
DOS>yuv2bmp sample.yuv test.bmp
```

Precautions: The BMP file which was converted into the YUV format may not match the original BMP file, depending on calculation results.

4.2.7 bmp2gry.exe

Function: Converts BMP files into GRY files.

Usage: DOS>**bmp2gry [option]** ***infile.bmp outfile.gry***↵

Arguments: ***infile.bmp*** Input filename (BMP file converted by "col2gry.exe")

outfile.gry Output filename (GRY file)

Option: **-d *value*** Specifies a value for grayscale levels.
value = 1: 2 grayscale levels (1 bit per pixel)
value = 2: 4 grayscale levels (2 bits per pixel)
value = 4: 16 grayscale levels (4 bits per pixel)
value = 8: 256 grayscale levels (8 bits per pixel)
The default value is 8. When this option is left blank, images are converted to 256-grayscale level data.

Example:
```
DOS>bmp2gry -d 4 sample.bmp sample.gry
```

Precautions: Although color BMP files can be converted directly into grayscale images using just this tool, we recommend BMP files already converted to grayscale with "col2gry.exe".
The 16-level (4 bits per pixel) grayscale data assigns one byte per two pixels, while the 256-level (8 bits per pixel) grayscale data assigns one byte per pixel. This tool can also be used to create 2-level GRY files (one byte per eight pixels) or 4-level GRY files (one byte per four pixels), although these files cannot be JPEG-compressed/expanded.

4.2.8 gry2bmp.exe

Function: Converts GRY files into grayscale BMP files.

Usage: DOS>**gry2bmp** ***infile.gry outfile.bmp***↵

Arguments: ***infile.gry*** Input filename (GRY file)

outfile.bmp Output filename (BMP file)

Example:
```
DOS>gry2bmp sample.gry test.bmp
```

Precautions: This tool converts a GRY file into a 24-bit BMP file in 16 grayscale levels (4 bits per pixel) or 256 grayscale levels (8 bits per pixel), according to the input file ID. The 16-level (4 bits per pixel) grayscale data is shifted left by 4 bits and has the offset value 0x08 added to the result, thus the RGB data in BMP file is obtained. The data has 16 values, 0xN8 (N = 0 to f).
The 256-level (8 bits per pixel) grayscale data is not shifted, and the offset value is 0. The data takes on 256 values from 0x00 to 0xff.
Note that the RGB data in the grayscale BMP file has the RGB relationship R = G = B.

4.2.9 col2gry.exe

Function: Converts a color image in BMP format into a grayscale image in BMP format.

Usage: DOS>**col2gry [option]** ***infile.bmp outfile.bmp***↵

Arguments: ***infile.bmp*** Input filename (BMP file)

outfile.bmp Output filename (BMP file)

Option: **-t *tablefile*** Specifies a grayscale conversion table file.

Example:
```
DOS>col2gry -t gray16.tbl sample.bmp gray.bmp
```

Table file: An example of a table used for 16-level (4 bits per pixel) grayscale conversion is given below.
Example: "\jpegtool\bin\gray16.tbl"

```
0f 08
1f 18
2f 28
3f 38
4f 48
5f 58
6f 68
7f 78
8f 88
9f 98
af a8
bf b8
cf c8
df d8
ef e8
ff f8
```

Write the threshold and conversion values in the same line, separated by a space or tab. Each value must be written in hexadecimal notation. Write the threshold value, then the conversion value.
The threshold value at the beginning of the file indicates the end of the grayscale range, starting from 0. In the above example, data from 0x0 to 0x0f is converted to 0x08 grayscale data (see the precautions below). In the following lines, the threshold value for each preceding line + 1 indicates the beginning of the grayscale range in that entry. For the second line in the above example, data from 0x10 to 0x1f is converted to 0x18 grayscale data.
As shown here, write the threshold values in ascending order. (Otherwise, an error is assumed.) If more than 255 entries of threshold or conversion values are written, the 256th and following entries are ignored. Data in an undefined grayscale range without a specified threshold value is not converted.
For most cases, specify an intermediate value in the range of threshold values for the conversion value.
For the 256-level grayscale conversion table, specify the same value for the threshold and conversion values from 0x0 up to 0xff. (See "\jpegtool\bin\gray256.tbl".)

Precautions: This tool first converts RGB data in the 24-bit BMP file into grayscale data, according to the equation below.
((R + G + B) * 21845 + 32768) / 65536
(Although this process averages R, G, and B, it converts floating-point values into fixed decimal values for integer arithmetic.) Following conversion to unsigned character type, if this data falls within the range of threshold values in the table file, it is converted to the specified value.

4.2.10 bin2s.exe

Function: Converts a binary file (JPEG, RGB, YUV, or GRY file) into a text file in E0C33 assembly source format.
Use the redirect function in DOS when saving to a file, because the resulting data is output into the standard output.

Usage: `DOS>bin2s [option] infile.jpg/rgb/yuv/gry > outfile.s↵`

Arguments:

infile.jpg/rgb/yuv/gry	Input filename (binary file)
outfile.s	Output filename (assembly source file)

Option:

-l *label*	Defines the assembler label name. If this option is left blank, the input filename is used as the label.

Examples: 1) When the -l option is left blank, the input filename becomes the assembler symbol name.

```
DOS>bin2s sample.jpg > sample.s
DOS>type sample.s
      .global sample
      .align 2
sample:
      .byte   0xff 0xd8 0xff 0xdb 0x00 0x84 0x00 0x08
      .byte   0x06 0x06 0x07 0x06 0x05 0x08 0x07 0x07
                                 :
DOS>
```

2) When using a symbol that differs from the filename, specify it with the -l option.

```
DOS>bin2s -l IMAGE1 sample.jpg @> sample.s
DOS>type sample.s
      .global IMAGE1
      .align 2
IMAGE1:
      .byte   0xff 0xd8 0xff 0xdb 0x00 0x84 0x00 0x08
      .byte   0x06 0x06 0x07 0x06 0x05 0x08 0x07 0x07
                                 :
DOS>
```

Precautions: The following limitations apply to symbol name specification.
- Symbol length: 32 characters or less
- Valid characters: a to z, A to Z, 0 to 9, and _

4.2.11 Execution with a Batch File

All image ROM creation tools are 32-bit applications that can be run from the DOS prompt. Thus, a series of processes can be executed using a batch file.
The following shows an example of steps executed using the batch files provided in the "jpegtool\sample\" directory. Each file assumes that "jpegtool\sample\" is the current directory, and that the image ROM creation tools in "jpegtool\" are run. Correct the files, if necessary.

Table 4.2.1 Sample Batch Files

Batch File	Content of Processing
bmptojpg.bat	Converts BMP to JPEG, creates assembly source file
bmptorgb.bat	Converts BMP to RGB, creates assembly source file
bmptoyuv.bat	Converts BMP to YUV, creates assembly source file
bmptogry.bat	Converts BMP to GRY, creates assembly source file
jpgtobmp.bat	Converts JPEG to BMP
rgbtobmp.bat	Converts RGB to BMP
yuvtobmp.bat	Converts YUV to BMP
grytobmp.bat	Converts GRY to BMP

bmptojpg.bat

Function: After converting a BMP file into a JPEG file, it generates an assembly source file. For the JPEG compression performed, it uses the -i option (interleave value) in "bmp2jpeg.exe" = 3 (Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1) and the -q option (picture quality index value) = 75.

Input file: *filename*.bmp BMP file

Output files: *filename*.jpg JPEG file
*filename*_jpg.s Assembly source file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo bmp2jpeg %1.bmp
..\bin\bmp2jpeg -i 3 -q 75 %1.bmp %1.jpg
echo bin2s %1.jpg
..\bin\bin2s -l %1_jpg %1.jpg > %1_jpg.s
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>bmptojpg sample
```
Converts a BMP file "sample.bmp" into the JPEG file "sample.jpg", then creates an assembly source file "sample_jpg.s". The file "sample_jpg.s" has the global label "sample_jpg" defined in it.

```
    .global sample_jpg
    .align  2
sample_jpg:
    .byte    0xff 0xd8 0xff 0xdb 0x00 0x84 0x00 0x08
    .byte    0x06 0x06 0x07 0x06 0x05 0x08 0x07 0x07
    .byte    0x07 0x09 0x09 0x08 0x0a 0x0c 0x14 0x0d
                                    :
; total 17346 bytes data
```

Reference: "4.2.1 bmp2jpeg.exe", "4.2.10 bin2s.exe"

jpgtobmp.bat

Function: Converts a JPEG file into a BMP file. For the JPEG expansion performed, it uses the -r option in "jpeg2bmp.exe" = 01 (not magnified, roughness of expansion = x2).

Input file: *filename*.jpg JPEG file

Output file: *filename*_jpg.bmp BMP file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo jpeg2bmp %1.jpg
..\bin\jpeg2bmp -r 01 %1.jpg %1_jpg.bmp
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>jpegtobmp sample
```
Converts a JPEG file "sample.jpg" into the BMP file "sample_jpg.bmp".

Reference: "4.2.2 jpeg2bmp.exe"

bmptorgb.bat

Function: Generates an assembly source file after converting a BMP file into an RGB file.

Input file: *filename*.bmp BMP file

Output files: *filename*.rgb RGB file
*filename*_rgb.s Assembly source file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo bmp2rgb %1.bmp
..\bin\bmp2rgb %1.bmp %1.rgb
echo bin2s %1.rgb
..\bin\bin2s -l %1_rgb %1.rgb > %1_rgb.s
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>bmptorgb sample
```
Converts a BMP file "sample.bmp" into the RGB file "sample.rgb", then creates an assembly source file "sample_rgb.s". The file "sample_rgb.s" has the global label "sample_rgb" defined in it.

```
    .global sample_rgb
    .align  2
sample_rgb:
    .byte    0x00 0x00 0x00 0x00 0x40 0x01 0x00 0x00
    .byte    0xf0 0x00 0x00 0x00 0x8e 0x98 0xa7 0x92
    .byte    0x96 0xa5 0x94 0x98 0xa8 0x95 0x97 0xa6
                                    :
; total 230412 bytes data
```

Reference: "4.2.3 bmp2rgb.exe", "4.2.10 bin2s.exe"

rgbtobmp.bat

Function: Converts an RGB file into a BMP file.

Input file: *filename*.rgb RGB file

Output file: *filename*_rgb.bmp BMP file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo rgb2bmp %1.rgb
..\bin\rgb2bmp %1.rgb %1_rgb.bmp
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>rgbtobmp sample
```
Converts an RGB file "sample.rgb" into the BMP file "sample_rgb.bmp".

Reference: "4.2.4 rgb2bmp.exe"

bmptoyuv.bat

Function: Generates an assembly source file after converting a BMP file into a YUV file.

Input file: *filename*.bmp BMP file

Output files: *filename*.yuv YUV file
*filename*_yuv.s Assembly source file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo bmp2yuv %1.bmp
..\bin\bmp2yuv %1.bmp %1.yuv
echo bin2s %1.yuv
..\bin\bin2s -l %1_yuv %1.yuv > %1_yuv.s
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>bmptorgb sample
```
Converts a BMP file "sample.bmp" into the YUV file "sample.yuv", then creates an assembly source file "sample_yuv.s". The file "sample_yuv.s" has a global label "sample_yuv" defined in it.

```
    .global sample_yuv
    .align  2
sample_yuv:
    .byte    0x01 0x00 0x00 0x00 0x40 0x01 0x00 0x00
    .byte    0xf0 0x00 0x00 0x00 0x97 0x89 0x7a 0x96
    .byte    0x88 0x7d 0x99 0x89 0x7d 0x98 0x88 0x7e
                                    :
; total 230412 bytes data
```

Reference: "4.2.5 bmp2yuv.exe", "4.2.10 bin2s.exe"

yuvtobmp.bat

Function: Converts a YUV file into a BMP file.

Input file: *filename*.rgb YUV file

Output file: *filename*_yuv.bmp BMP file

Contents of file:
```
@echo off
if "%1"=="" goto ERROR
echo yuv2bmp %1.yuv
..\bin\yuv2bmp %1.yuv %1_yuv.bmp
goto END
:ERROR
echo Please input filename
:END
```

Example:
```
>yuvtobmp sample
```
Converts a YUV file "sample.yuv" into the BMP file "sample_yuv.bmp".

Reference: "4.2.6 yuv2bmp.exe"

bmptogry.bat

Function: Generates an assembly source file after converting a BMP file into a GRY file. It first converts a color image in the BMP file into a 16-level grayscale image, using "col2gry.exe" and the conversion table file "gray16.tbl", then creates a GRY file from the resulting file and converts it into an assembly source file.

Input file:

filename.bmp	BMP file (color)

Output files:

*filename*_c2g.bmp	BMP file (16-level grayscale)
filename.gry	GRY file
*filename*_gry.s	Assembly source file

Contents of file:

```
@echo off
if "%1"=="" goto ERROR
echo col2gry %1.bmp
..\bin\col2gry -t ..\bin\gray16.tbl %1.bmp %1_c2g.bmp
echo bmp2gry %1_c2g.bmp
..\bin\bmp2gry -d 4 %1_c2g.bmp %1.gry
echo bin2s %1.gry
..\bin\bin2s -l %1_gry %1.gry > %1_gry.s
goto END
:ERROR
echo Please input filename
:END
```

Example:

```
>bmptogry sample
```

Converts a BMP file "sample.bmp" into the GRY file "sample.gry", then creates the assembly source file "sample_gry.s". The file "sample_gry.s" has the global label "sample_gry" defined in it.

```
    .global sample_gry
    .align  2
sample_gry:
    .byte    0x03 0x00 0x00 0x00 0x40 0x01 0x00 0x00
    .byte    0xf0 0x00 0x00 0x00 0x99 0x99 0x99 0x99
    .byte    0x99 0x99 0x97 0x43 0x44 0x44 0x44 0x34
                                      :
; total 38412 bytes data
```

Reference: "4.2.7 bmp2gry.exe", "4.2.10 bin2s.exe"

grytobmp.bat

Function: Converts a GRY file into a BMP file.

Input file:

filename.gry	GRY file

Output file:

*filename*_gry.bmp	BMP file

Contents of file:

```
@echo off
if "%1"=="" goto ERROR
echo gry2bmp %1.gry
..\bin\gry2bmp %1.gry %1_gry.bmp
goto END
:ERROR
echo Please input filename
:END
```

Example:

```
>grytobmp sample
```

Converts a GRY file "sample.gry" into the BMP file "sample_gry.bmp".

Reference: "4.2.8 gry2bmp.exe"

4.2.12 Execution by a make File

A make file allows you to create image ROM data more efficiently. A sample make file, "sample.mak", is provided in the "jpegtool\sample\" directory, for your reference as follows.
For details of make file syntax and make functionality, refer to the "E0C33 Family C Compiler Package Manual".

sample.mak

This make file creates the image ROM data shown below from the sample BMP file "sample.bmp" as image source data for the demonstration programs (demo1, demo2, demo3) found in the "jpeg33\jpeglib\" directory.

1) Image ROM data "data1.s" for demo1
   sample_gry1: 16-level (4 bits per pixel) grayscale data (GRY format)
   sample_gry2: 256-level (8 bits per pixel) grayscale data (GRY format)
   sample_yuv: YUV data

2) Image ROM data "data2.s" for demo2

sample_jpg1:	16-level (4 bits per pixel) grayscale data (JPEG format)		Quality index value = 75
sample_jpg2:	256-level (8 bits per pixel) grayscale data (JPEG format)		Quality index value = 75
sample_jpg3:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 75
sample_jpg4:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:1:1:1:1:1	Quality index value = 75
sample_jpg5:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 1:1:1:1:1:1	Quality index value = 75
sample_jpg6:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 1
sample_jpg7:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 25
sample_jpg8:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 50
sample_jpg9:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 75
sample_jpg10:	Color data (JPEG format)	Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1	Quality index value = 100

3) Image ROM data "data3.s" for demo3
   sample_rgb: Color data (RGB format)

To create an image ROM data file, enter the command line shown below to start make.

```
DOS>make -f sample.mak
```

The file "sample.mak" has been created to be executed from the "jpegtool\sample\" current directory. Specify the "make.exe" directory with the PATH command or copy "make.exe" into the "jpegtool\sample\" directory before running.

The file "sample.mak" also contains a command to delete all generated files except the original (sample.bmp). To execute this function, enter the command line shown below as you launch make.

```
DOS>make -f sample.mak clean
```

The contents of the file are shown below.

```
# macro definitions for tools & dir

TOOL_DIR = ..\bin
BIN2S = $(TOOL_DIR)\bin2s.exe
BMP2GRY = $(TOOL_DIR)\bmp2gry.exe
BMP2JPEG = $(TOOL_DIR)\bmp2jpeg.exe
BMP2RGB = $(TOOL_DIR)\bmp2rgb.exe
BMP2YUV = $(TOOL_DIR)\bmp2yuv.exe
COL2GRY = $(TOOL_DIR)\col2gry.exe

# suffix & rule definitions

.SUFFIXES : .bmp .gry1 .gry2 .jpg1 .jpg2 .jpg3 .jpg4 .jpg5 .jpg6 .jpg7 .jpg8 .jpg9 \
.jpg10 .rgb .yuv

.bmp.gry1 :
        $(COL2GRY) -t ..\bin\gray16.tbl $*.bmp $*_c2g1.bmp
        $(BMP2GRY) -d 4 $*_c2g1.bmp $*.gry1
        $(BIN2S) -l $*_gry1 $*.gry1 > $*_gry1.s
```

```
.bmp.gry2 :
        $(COL2GRY) -t ..\bin\gray256.tbl $*.bmp $*_c2g2.bmp
        $(BMP2GRY) -d 8 $*_c2g2.bmp $*.gry2
        $(BIN2S) -l $*_gry2 $*.gry2 > $*_gry2.s

.bmp.jpg1 :
        $(COL2GRY) -t ..\bin\gray16.tbl $*.bmp $*_c2g3.bmp
        $(BMP2JPEG) -i 0 -q 75 -d 4 $*_c2g3.bmp $*.jpg1
        $(BIN2S) -l $*_jpg1 $*.jpg1 > $*_jpg1.s

.bmp.jpg2 :
        $(COL2GRY) -t ..\bin\gray256.tbl $*.bmp $*_c2g4.bmp
        $(BMP2JPEG) -i 0 -q 75 $*_c2g4.bmp $*.jpg2
        $(BIN2S) -l $*_jpg2 $*.jpg2 > $*_jpg2.s

.bmp.jpg3 :
        $(BMP2JPEG) -i 3 -q 75 $*.bmp $*.jpg3
        $(BIN2S) -l $*_jpg3 $*.jpg3 > $*_jpg3.s

.bmp.jpg4 :
        $(BMP2JPEG) -i 2 -q 75 $*.bmp $*.jpg4
        $(BIN2S) -l $*_jpg4 $*.jpg4 > $*_jpg4.s

.bmp.jpg5 :
        $(BMP2JPEG) -i 1 -q 75 $*.bmp $*.jpg5
        $(BIN2S) -l $*_jpg5 $*.jpg5 > $*_jpg5.s

.bmp.jpg6 :
        $(BMP2JPEG) -i 3 -q 1 $*.bmp $*.jpg6
        $(BIN2S) -l $*_jpg6 $*.jpg6 > $*_jpg6.s

.bmp.jpg7 :
        $(BMP2JPEG) -i 3 -q 25 $*.bmp $*.jpg7
        $(BIN2S) -l $*_jpg7 $*.jpg7 > $*_jpg7.s

.bmp.jpg8 :
        $(BMP2JPEG) -i 3 -q 50 $*.bmp $*.jpg8
        $(BIN2S) -l $*_jpg8 $*.jpg8 > $*_jpg8.s

.bmp.jpg9 :
        $(BMP2JPEG) -i 3 -q 75 $*.bmp $*.jpg9
        $(BIN2S) -l $*_jpg9 $*.jpg9 > $*_jpg9.s

.bmp.jpg10 :
        $(BMP2JPEG) -i 3 -q 100 $*.bmp $*.jpg10
        $(BIN2S) -l $*_jpg10 $*.jpg10 > $*_jpg10.s

.bmp.rgb :
        $(BMP2RGB) $*.bmp $*.rgb
        $(BIN2S) -l $*_rgb $*.rgb > $*_rgb.s

.bmp.yuv :
        $(BMP2YUV) $*.bmp $*.yuv
        $(BIN2S) -l $*_yuv $*.yuv > $*_yuv.s

# dependency list

all : data1.s data2.s data3.s
# Demo1 input file (4bit depth GRY file, 8bit depth GRY file, YUV file)
data1.s : sample.gry1 sample.gry2 sample.yuv
        type sample_gry1.s > data1.s
        type sample_gry2.s >> data1.s
        type sample_yuv.s >> data1.s

# Demo2 input file (JPEG file)
data2.s : sample.jpg1 sample.jpg2 sample.jpg3 sample.jpg4 sample.jpg5 sample.jpg6 \
sample.jpg7 sample.jpg8 sample.jpg9 sample.jpg10
        type sample_jpg1.s > data2.s
        type sample_jpg2.s >> data2.s
        type sample_jpg3.s >> data2.s
        type sample_jpg4.s >> data2.s
        type sample_jpg5.s >> data2.s
```

```
        type sample_jpg6.s >> data2.s
        type sample_jpg7.s >> data2.s
        type sample_jpg8.s >> data2.s
        type sample_jpg9.s >> data2.s
        type sample_jpg10.s >> data2.s

# Demo3 input file (RGB file)
data3.s : sample.rgb
        type sample_rgb.s > data3.s

# BMP->GRY (4bit depth grayscale)
sample.gry1 : sample.bmp

# BMP->GRY (8bit depth grayscale)
sample.gry2 : sample.bmp

# BMP->JPEG (4bit depth grayscale mode (Yh:Yv=1:1) and quality 75)
sample.jpg1 : sample.bmp

# BMP->JPEG (8bit depth grayscale mode (Yh:Yv=1:1) and quality 75)
sample.jpg2 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 75)
sample.jpg3 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:1:1:1:1:1) and quality 75)
sample.jpg4 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=1:1:1:1:1:1) and quality 75)
sample.jpg5 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 1)
sample.jpg6 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 25)
sample.jpg7 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 50)
sample.jpg8 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 75)
sample.jpg9 : sample.bmp

# BMP->JPEG (Color mode (Yh:Yv:Uh:Uv:Vh:Vv=2:2:1:1:1:1) and quality 100)
sample.jpg10 : sample.bmp

# BMP->RGB
sample.rgb : sample.bmp

# BMP->YUV
sample.yuv : sample.bmp

clean:
        del sample_c2g*.bmp
        del *.gry
        del *.jpg
        del *.rgb
        del *.yuv
        del *.s
```

4.3 Image Compression/Expansion Evaluating Tool

This section describes how to use the image compression/expansion evaluating tool "JPEG33 Bench (jb.exe)" and its various functions.

Outline

JPEG33 Bench (jb.exe) is a GUI tool for JPEG33 compression/expansion and grayscale BMP file conversion. It allows you to select a file to convert, set compression parameters, and start conversion tools ("bmp2jpeg.exe", "jpeg2bmp.exe" or "col2gry.exe"), all in one window. You can also display the original and converted images.

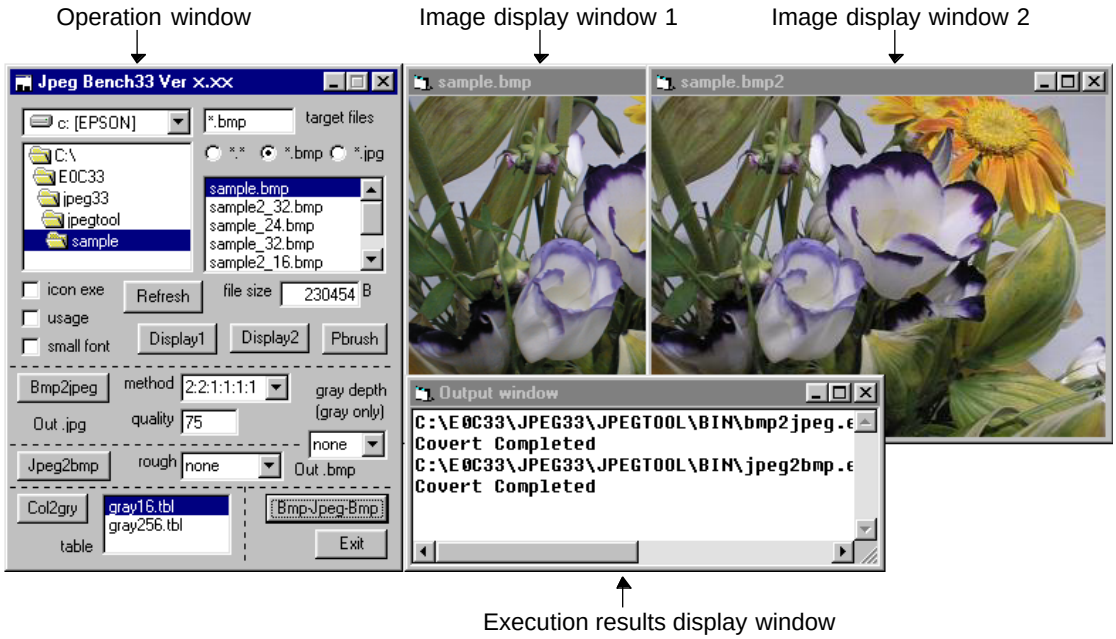
Starting and closing

Double-click the "jb.exe" icon to start JPEG33 Bench.

To close JPEG33 Bench, click the [Exit] button in the [Jpeg Bench33] window.

Window structure

JPEG33 Bench has five windows.



Operation window ([Jpeg Bench33] window)

The window displayed when you start JPEG33 Bench. All operations are performed in this window.

Image display windows 1 and 2

Select the image file (BMP format only) you want to display from the file list box and click the [Display 1] button. Image display window 1 opens, displaying the selected image. Similarly, image display window 2 will appear when you click the [Display 2] button. Or you can double-click the desired image filename to display it in Image display window 1.

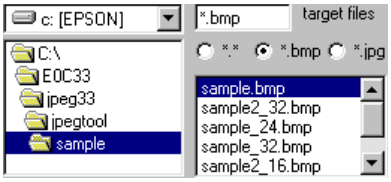
When you choose a BMP file and double-click the [Bmp-Jpeg-Bmp] button, the software displays the BMP image before compression in Image display window 1 and the BMP image expanded after JPEG compression in Image display window 2. If you specify conversion to grayscale before clicking on the [Bmp-Jpeg-Bmp] button, the software displays the BMP format image converted into grayscale before JPEG-compression in Image display window 1.

Execution results display window

This window displays the execution results of the tool launched by pressing the execution button. The contents displayed in this window are the execution and error messages and Usage that are output to STDOUT by "bmp2jpeg.exe", "jpeg2bmp.exe" and "col2gry.exe". The messages are recorded in the log file "jb.err" by "ccap.exe" (refer to the "E0C33 Family C Compiler Package Manual"), the contents of which are displayed in this window.

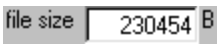
Controls on the operation window

File selection



Select the image file (BMP format) you want to convert or display. Select multiple files by holding down the [Shift] or [Ctrl] key as you click multiple filenames. Multiple files are processed one after another. ([Display 1/2] and [Bmp-Jpeg-Bmp] buttons work only for the first file among those selected.)

Updates the contents displayed in the file list box. The contents of the file list box are not automatically updated when you create or update files with a tool.



Displays the size of the file selected in the file list box. When multiple files are selected, the size of the first file is displayed.

Image display

Displays the image file (BMP format) selected in the file list box in Image display window 1. When multiple files are selected, the first file is displayed.

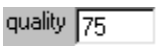
Displays the image file (BMP format) selected in the file list box in Image display window 2. When multiple files are selected, the first file is displayed.

Starts the Windows "Paint" application for the image file (BMP format) selected in the file list box. When multiple files are selected, "Paint" is launched for all selected files.

Image compression (BMP→JPEG)

Starts "bmp2jpeg.exe" to compress the BMP file selected in the file list box, according to the parameters set for conversion to JPEG format. Multiple files are processed one after another.
An output filename is automatically assigned, based on the input filename and the extension ".jpg2". It cannot be changed in JPEG33 Bench.

Selects the -i option in "bmp2jpeg.exe" (interleave value, Yh:Yv:Uh:Uv:Vh:Vv). If you select 1:1:1:1:1:1, the UV data is not interleaved when compressed. If you select 2:1:1:1:1:1, the UV data is thinned 1/2 in only the horizontal direction. If you select 2:2:1:1:1:1, the UV data is thinned 1/2 horizontally and vertically.
Choose "gray" for grayscale images.
The default value is 2:2:1:1:1:1.

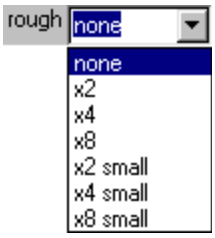


Sets the -q option in "bmp2jpeg.exe" (picture quality index value).
Enter a decimal value from 1 to 100. Smaller values correspond to smaller file size following conversion to JPEG format, but with correspondingly poor picture quality.
The default value is 75.

Selects the -d option in "bmp2jpeg.exe" (depth of grayscale value).
Selecting "8bit" converts input data to 256-level (8 bits per pixel) grayscale format.
Selecting "4bit" converts input data to 16-level (4 bits per pixel) grayscale format.
This specification is enabled only when "gray" is selected in the [interleave] list box. It is otherwise ignored.
This list box is also used to set options in "jpeg2bmp.exe".

Image expansion (JPEG→BMP)

Starts "jpeg2bmp.exe" to expand the JPEG file selected in the file list in BMP format, according to the set conversion parameters. Multiple files are processed one after another.
An output filename is automatically assigned, based on the input filename and the extension ".bmp2". It cannot be changed in JPEG33 Bench.



Selects the -r option in "jpeg2bmp.exe" (pixel magnification value and the roughness of reduced expansion). The specific actions are shown below.

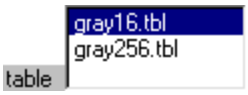
none:	Not magnified, no roughness	(Set value = -r0)
x2:	Magnified 2-fold, x2 roughness	(Set value = -r11)
x4:	Magnified 4-fold, x4 roughness	(Set value = -r22)
x8:	Magnified 8-fold, x8 roughness	(Set value = -r33)
x2 small:	Not magnified, x2 roughness	(Set value = -r1)
x4 small:	Not magnified, x4 roughness	(Set value = -r2)
x8 small:	Not magnified, x8 roughness	(Set value = -r3)

The default value is "none".
If you select the x2, x4, or x8 option, the image data is reduction-expanded with the specified roughness, resulting in magnification of pixels by that factor. Consequently, the image data is expanded to original size into a mosaic image with the roughness of the specified magnification, enabling pseudo-progressive rendering. Selecting x2 small, x4 small, or x8 small reduces images by thinning out pixels to 1/2, 1/4, or 1/8, so that the original image is reduced as image data is expanded. These options can be used to display image thumbnails.

Selects the -d option in "jpeg2bmp.exe" (depth of grayscale value).
If you select "8bit", input data is converted to 256-level (8 bits per pixel) grayscale format. If you select "4bit," input data is converted to 16-level (4 bits per pixel) grayscale format. This specification applies only for grayscale expansion.
This list box is also used to set options in "bmp2jpeg.exe".

Grayscale conversion (color BMP→grayscale BMP)

Starts "col2gry.exe" to convert the BMP format color file selected in the file list box to BMP grayscale, using the selected conversion table file. Multiple files are processed one after another.
An output filename is automatically assigned, based on the input filename and the extension "g.gry". It cannot be changed in JPEG33 Bench.



Selects the conversion table file to be used for grayscale conversion by "col2gry.exe". This list box displays the files existing in the same directory as "jb.exe" having the extension ".tbl". The conversion table files "gray16.tbl" and "gray256.tbl" have been preinstalled for 16-level (4 bits per pixel) and 256-level (8 bits per pixel) grayscale conversion, respectively. To use your own conversion table files, place them in the same directory as "jb.exe" after adding the ".tbl" extension.

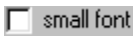
[Bmp-Jpeg-Bmp] button

Executes image data compression by "bmp2jpeg.exe" and image data expansion by "jpeg2bmp.exe" in succession. It also displays the image before compression in Image display window 1 and the expanded image in Image display window 2, allowing you to compare the original image and the image following compression and expansion. The options for these tools are set just as when running each application individually. If grayscale conversion is specified in the option for JPEG compression, the converted image is displayed as the original image in Image display window 1.
When multiple files are selected, the first selected file is processed.

Tool execution options

Selecting this option runs each tool minimized, as an icon. If this option is not selected, a DOS prompt window is displayed as the applications runs.

Selecting this option before running a tool displays Usage, the information output by the tool, in the Execution results display window. In this case, no compression, expansion, or conversion processing is performed.



Selecting this option displays the execution results in the Execution results display window in a smaller font size. (This specification applies only when the Terminal 10-point font is installed in the OS.)

[Exit] button

Terminates JPEG33 Bench.

Precautions

- When using JPEG33 Bench to expand JPEG images, note that only the following seven combinations of expansion roughness and magnification values are supported.
  none: Not magnified, no roughness (Set value = -r0)
  x2: Magnified 2-fold, x2 roughness (Set value = -r11)
  x4: Magnified 4-fold, x4 roughness (Set value = -r22)
  x8: Magnified 8-fold, x8 roughness (Set value = -r33)
  x2 small: Not magnified, x2 roughness (Set value = -r1)
  x4 small: Not magnified, x4 roughness (Set value = -r2)
  x8 small: Not magnified, x8 roughness (Set value = -r3)
  To use any other combination, run "jpeg2bmp.exe" from the DOS prompt.
- You cannot specify the output filename in JPEG33 Bench. Any existing file having the same name as the output file will be overwritten without prompting. To compare results for the same image after multiple processing involving different parameters, modify the created output filename before executing the next processing.
- JPEG33 Bench does not support the following functions. Use individual tools to execute the unsupported functions.
  - Conversion to non-BMP or JPEG formats
  - Creation of assembly source files

5 JPEG33 Library Reference

This chapter describes various precautions related to JPEG33 library functions.

5.1 Outline of the JPEG33 Library

Functional outline

The JPEG33 library consists of a group of image processing functions in srf33 library format. Before use, this library must be linked to the target program. Calling the required functions from a target program permits you to execute the functions listed below on the E0C33 chip.

- JPEG compression/expansion (including reduced expansion)
- RGB↔YUV conversion
- RGB↔Y conversion
- CC33 grayscale data↔Y conversion
- Pixel magnification (used for pseudo-progressive display)

The JPEG33 software package also offers C source code for high-level functions based on JPEG33 library functions, as well as assembly source code for demonstration programs. Any of the source code may be copied to the target program and used.
The functions allow easy implementation of image processing functions into your system.

Program structure

The structure of the application programs is shown in Figure 5.1.1.

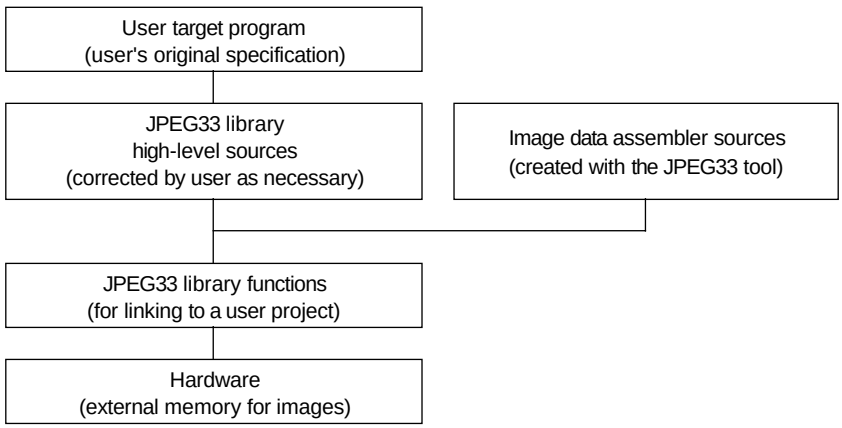


Figure 5.1.1 Program Structure

Configuration of the JPEG library

The JPEG library and its associated files are all found in the "jpeglib" folder (directory). The contents of the "jpeglib" folder are listed below.

jpeglib	JPEG33 library files
readme.txt	Supplementary explanations, etc. (in English)
readmeja.txt	Supplementary explanations, etc. (in Japanese)
lib	JPEG33 library directory
jpeg.lib	JPEG33 library
conv_yuv.o	Object file (RGB↔YUV image format conversion)
jfdctin.o	Object file (DCT conversion)
imemin.o	Object file (internal buffer RAM declaration)
jstripin.o	Object file (image sampling)
	(These object files may be copied to internal RAM for faster execution. See Section 5.6, "Techniques for Improving Processing Speed".)
include	Include file directory of JPEG33 library
jpegtop.h	High-level function header file
src	Public source directory of high-level function
jpegtop.c	High-level function C source file of JPEG33 library
libsrc	Public source directory for JPEG33 library
conv	Image format conversion function source directory
conv_yuv.c	RGB↔YUV image format conversion C source file
conv_yuv.h	RGB↔YUV image format conversion C source file
conv_y.c	RGB↔Y image format conversion C source file
conv_y.h	RGB↔Y image format conversion C source file
conv_4g.c	Y↔4-bit grayscale image format conversion C source file
conv_4g.h	Y↔4-bit grayscale image format conversion header file
conv_8g.c	Y↔8-bit grayscale image format conversion C source file
conv_8g.h	Y↔8-bit grayscale image format conversion header file
resizer	Image resize function source directory
resizer.c	Color pixel resize C source file
resizer.h	Color pixel resize header file
resizgry.c	Grayscale pixel resize C source file
resizgry.h	Grayscale pixel resize header file
demo1	Demonstration program directory
demo2	Each directory (demoX) contains sample programs to help you get the
:	most from the JPEG33 library. For configuration and contents of the
demoN	demonstration programs, refer to "readme.txt" or "readmeja.txt" in "jpeglib\".

∗ Configuration of functions in "jpegtop.c" and "jpeg.lib" are described later.

5.2 High-level Functions

High-level functions are provided in a C source code file (jpegtop.c) to facilitate implementation of JPEG compression and image expansion. They are implemented using JPEG library functions.
"jpegtop.c" functions are listed in Table 5.2.1.

Table 5.2.1 High-level Functions

Function Name	Description
jpgDataEncode( )	JPEG compresses color images
jpgGryDataEncode( )	JPEG compresses grayscale images
jpgData Decode( )	JPEG expands color images
jpgData DecodeRough( )	JPEG expands color images (with expansion roughness and magnification specified)
jpgGryDataDecode( )	JPEG expands grayscale images
jpgGryDataDecodeRough( )	JPEG expands grayscale images (with expansion roughness and magnification specified)

Required functions are copied from "jpegtop.c" and pasted into user program source code files. External variables must be defined (see Section 5.2.1). You must also include "jpegtop.h" in the user program. When using "jpegtop.c" directly, rather than copying functions from it, external variables do not need to be defined. Make sure to include "jpegtop.h" in the user program.
If you want more detailed control of the JPEG 33 library, call the library functions directly instead of using "jpegtop.c". The C source code file "jpegtop.c" may then be used as a sample program.

5.2.1 Defining External Variables

To use high-level functions after copying them one by one into the user program, include "jpegtop.h" in the user program and define the external variables (compression data line buffers) listed below.
You do not need to define external variables when using "jpegtop.c" directly, since they are already defined in it. However, be sure to include "jpegtop.h" in the user program.

```
unsigned int jmemory_common_block1[JMEMORY_IMAGE_WIDTH * 16 / sizeof(unsigned int)];
unsigned int jmemory_common_block2[JMEMORY_IMAGE_WIDTH * 16 / sizeof(unsigned int)];
unsigned int jmemory_common_block3[JMEMORY_IMAGE_WIDTH * 16 / sizeof(unsigned int)];

unsigned int *jmemory_block_ptr[3] = {
        jmemory_common_block1,
        jmemory_common_block2,
        jmemory_common_block3
};
```

For JMEMORY_IMAGE_WIDTH, see the next section.

5.2.2 Defining Horizontal Image Size

By default, high-level functions are set up for processing of images in sizes of horizontal 320 pixels × vertical 32,767 pixels (max). Although vertical size is fixed at 32,767 pixels, horizontal size may be increased up to 32,767 pixels in increments of 16 pixels (fractions rounded up) by changing the parameters defined in "jpegtop.h".
The parameters defined in "jpegtop.h" are listed below.

```
JMEMORY_IMAGE_
/*
 *      JPEG memory common buffer size definition
 */
#define JMEMORY_IMAGE_WIDTH     320             /* Width 320dot */
```

To change the horizontal image size to 160 dots, for example, modify the parameter as below.

```
#define JMEMORY_IMAGE_WIDTH     160             /* Width 160dot */
```

5.2.3 Description of Each Function

jpgDataEncode( )

Function: JPEG-compresses color images.

Format:
```
int jpgDataEncode(unsigned char *pucJpgBuf, IMAGE *srcImage,
unsigned short uwWidth, unsigned short uwHeight, int iInterleave,
int iQuality, unsigned long *pulSize);
```

Arguments:

unsigned char	***pucJpgBuf**	Pointer to memory area where JPEG data is held
IMAGE	***srcImage**	Pointer to image data (YUV data) to be compressed
unsigned short	**uwWidth**	Number of horizontal pixels in image data
unsigned short	**uwHeight**	Number of vertical pixels in image data
int	**iInterleave**	Interleave value (0 to 3) INTER_GRY (=0): Grayscale, Yh:Yv = 1:1 INTER_COLOR1 (=1): Color, Yh:Yv:Uh:Uv:Vh:Vv = 1:1:1:1:1:1 INTER_COLOR2 (=2): Color, Yh:Yv:Uh:Uv:Vh:Vv = 2:1:1:1:1:1 INTER_COLOR3 (=3): Color, Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1
int	**iQuality**	Picture quality index value (1 to 100) Larger values correspond to lower picture quality degradation (and hence better pictures), but at the cost of larger JPEG image files. Smaller values correspond to lower picture quality, with more compact JPEG files.
unsigned long	***pulSize**	Pointer to compressed JPEG image size (in bytes) The pointer to the variable for the JPEG-compressed image size is received.

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function compresses YUV format data according to the interleave and picture quality index values specified for conversion to JPEG format.

Example: Interleave value = 3, picture quality index value = 75

```
unsigned char     JpgBuf[0x40000];
unsigned short    uwWidth, uwHeight;
unsigned long     ulSize;
IMAGEHEAD         *pImageHead;
IMAGE             *srcImage;

pImageHead = (IMAGEHEAD *)sample_yuv;
srcImage = (IMAGE *)((unsigned char *)pImageHead + sizeof(IMAGEHEAD));
uwWidth = (unsigned short)pImageHead->ulWidth;
uwHeight = (unsigned short)pImageHead->ulHeight;
jpgDataEncode(JpgBuf, srcImage, uwWidth, uwHeight, INTER_COLOR3, 75, &ulSize);
```

jpgGryDataEncode( )

Function: JPEG-compresses grayscale images

Format:
```
int jpgGryDataEncode(unsigned char *pucJpgBuf, unsinged char *pucSrcBuf,
unsigned short uwWidth, unsigned short uwHeight, int iInterleave,
int iQuality, int iDepth, unsigned long *pulSize);
```

Arguments:

unsigned char	***pucJpgBuf**	Pointer to memory area where JPEG data is held
unsigned char	***pucSrcBuf**	Pointer to image data (GRY data) to be compressed
unsigned short	**uwWidth**	Number of horizontal pixels in image data
unsigned short	**uwHeight**	Number of vertical pixels in image data
int	**iInterleave**	Interleave value (0) INTER_GRY (=0): Grayscale, Yh:Yv = 1:1
int	**iQuality**	Picture quality index value (1 to 100) Larger values correspond to lower picture quality degradation (and hence better pictures), but at the cost of larger JPEG files. Smaller values correspond to lower picture quality, with more compact JPEG files.
int	**iDepth**	Grayscale depth (4, 8) 4: 16-level (4 bits per pixel) grayscale 8: 256-level (8 bits per pixel) grayscale
unsigned long	***pulSize**	Pointer to compressed JPEG image size (in bytes) The pointer to the variable for the JPEG-compressed image size is received.

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function compresses GRY-format grayscale data according to the interleave and picture quality index values specified for conversion to JPEG format data.

Example: 16-level (4 bits per pixel) grayscale, picture quality index value = 75

```
unsigned char     JpgBuf[0x40000];
unsigned short    uwWidth, uwHeight;
unsigned long     ulSize;
IMAGEHEAD         *pImageHead;
unsigned char     *pucImageBuf;

pImageHead = (IMAGEHEAD *)sample_gry1;
pucImageBuf = ((unsigned char *)pImageHead + sizeof(IMAGEHEAD));
uwWidth = (unsigned short)pImageHead->ulWidth;
uwHeight = (unsigned short)pImageHead->ulHeight;
jpgGryDataEncode(JpgBuf, pucImageBuf, uwWidth, uwHeight, INTER_GRY, 75, 4,
&ulSize);
```

jpgDataDecode( )

Function: JPEG-expands color images.

Format: `int jpgDataDecode(unsigned char *pucJpgBuf, IMAGE *destImage, unsigned short *puwWidth, unsigned short *puwHeight);`

Arguments:

unsigned char	***pucJpgBuf**	Pointer to JPEG data
IMAGE	***destImage**	Pointer to memory area where expanded image data (YUV data) is held
unsigned short	***puwWidth**	Pointer to number of horizontal pixels in image data
unsigned short	***puwHeight**	Pointer to number of vertical pixels in image data

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function expands color JPEG data into YUV files.

Example:

```
struct   tIMAGEDATA
{
IMAGEHEAD         stHeader;                 /* Image header */
IMAGE             stImage[IMAGE_SIZE];      /* YUV raw data */
}stImageData;
unsigned char     *pucJpgBuf;
unsigned short    uwWidth, uwHeight;
IMAGE             *destImage;

pucJpgBuf = sample_jpg3;
ulId = ID_YUV;
destImage = stImageData.stImage;
jpgDataDecode(pucJpgBuf, destImage, &uwWidth, &uwHeight);

stImageData.stHeader.ulId = ulId;
stImageData.stHeader.ulWidth = (unsigned long)uwWidth;
stImageData.stHeader.ulHeight = (unsigned long)uwHeight;
```

jpgDataDecodeRough( )

Function: JPEG-expands color images (with expansion roughness and magnification specified)

Format:
```
int jpgDataDecodeRough(unsigned char *pucJpgBuf, IMAGE *destImage,
unsigned long *pulWidth, unsigned long *pulHeight, int iRough, int iMag);
```

Arguments:

unsigned char	***pucJpgBuf**	Pointer to JPEG data
IMAGE	***destImage**	Pointer to memory area where expanded image data (YUV data) is held
unsigned long	***pulWidth**	Pointer to number of horizontal pixels in image data
unsigned long	***pulHeight**	Pointer to number of vertical pixels in image data
int	**iRough**	Roughness of expansion value (0 to 3)
		ROUGH_NONE (=0): No roughness of expansion
		ROUGH_MIN (=1): Expanded with x2 roughness
		ROUGH_MID (=2): Expanded with x4 roughness
		ROUGH_MAX (=3): Expanded with x8 roughness
int	**iMag**	Magnification value (0 to 3)
		MAG_NONE (=0): Not magnified
		MAG_MIN (=1): Magnified 2-fold
		MAG_MID (=2): Magnified 4-fold
		MAG_MAX (=3): Magnified 8-fold

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function expands JPEG color files into YUV files. Roughness of expansion and the magnification value may be specified. If 1 to 3 are specified for roughness of expansion and magnification value, images are rough-expanded, allowing for pseudo-progressive expansion.

Example: Roughness of expansion = 1, magnification value = 1

```
struct   tIMAGEDATA
{
IMAGEHEAD         stHeader;                  /* Image header */
IMAGE             stImage[IMAGE_SIZE];       /* YUV raw data */
}stImageData;
unsigned char     *pucJpgBuf;
unsigned long     ulWidth, ulHeight;
IMAGE             *destImage;

pucJpgBuf = sample_jpg3;
ulId = ID_YUV;
destImage = stImageData.stImage;
jpgDataDecodeRough(pucJpgBuf, destImage, &ulWidth, &ulHeight, ROUGH_MIN,
MAG_MIN);

stImageData.stHeader.ulId = ulId;
stImageData.stHeader.ulWidth = ulWidth;
stImageData.stHeader.ulHeight = ulHeight;
```

jpgGryDataDecode( )

Function: JPEG-expands grayscale images.

Format: `int` **`jpgGryDataDecode`**`(unsigned char *pucJpgBuf, unsigned char *pucDestBuf, unsigned short *puwWidth, unsigned short *puwHeight, int iDepth);`

Arguments:

unsigned char	***pucJpgBuf**	Pointer to JPEG data
unsigned char	***pucDestBuf**	Pointer to memory area where expanded image data (GRY data) is held
unsigned short	***puwWidth**	Pointer to number of horizontal pixels in image data
unsigned short	***puwHeight**	Pointer to number of vertical pixels in image data
int	**iDepth**	Grayscale depth (4, 8) 4: 16-level (4 bits per pixel) grayscale 8: 256-level (8 bits per pixel) grayscale

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function expands grayscale JPEG files into GRY-format files.

Example: 16-level (4 bits per pixel) grayscale

```
struct   tIMAGEDATA
{
IMAGEHEAD         stHeader;                  /* Image header */
IMAGE             stImage[IMAGE_SIZE];       /* YUV raw data */
}stImageData;
unsigned char     *pucJpgBuf;
unsigned char     *pucImageBuf;
unsigned short    uwWidth, uwHeight;

pucJpgBuf = sample_jpg2;
ulId = ID_GRY;
pucImageBuf = (unsigned char *)stImageData.stImage;
jpgGryDataDecode(pucJpgBuf, pucImageBuf, &uwWidth, &uwHeight, 4);

stImageData.stHeader.ulId = ulId;
stImageData.stHeader.ulWidth = (unsigned long)uwWidth;
stImageData.stHeader.ulHeight = (unsigned long)uwHeight;
```

jpgGryDataDecodeRough( )

Function: JPEG-expands grayscale images (with expansion roughness and magnification specified).

Format:
```
int jpgGryDataDecodeRough(unsigned char *pucJpgBuf,
unsigned char *pucDestBuf, unsigned long *pulWidth,
unsigned long *pulHeight, int iRough, int iMag, int iDepth);
```

Arguments:

unsigned char ***pucJpgBuf**	Pointer to JPEG data
unsigned char ***pucDestBuf**	Pointer to memory area where expanded image data (GRY data) is held
unsigned long ***pulWidth**	Pointer to number of horizontal pixels in image data
unsigned long ***pulHeight**	Pointer to number of vertical pixels in image data
int **iRough**	Roughness of expansion value (0 to 3)
	ROUGH_NONE (=0): No roughness of expansion
	ROUGH_MIN (=1): Expanded with x2 roughness
	ROUGH_MID (=2): Expanded with x4 roughness
	ROUGH_MAX (=3): Expanded with x8 roughness
int **iMag**	Magnification value (0 to 3)
	MAG_NONE (=0): Not magnified
	MAG_MIN (=1): Magnified 2-fold
	MAG_MID (=2): Magnified 4-fold
	MAG_MAX (=3): Magnified 8-fold
int **iDepth**	Grayscale depth (4, 8)
	4: 16-level (4 bits per pixel) grayscale
	8: 256-level (8 bits per pixel) grayscale

Return value: 1 when terminated normally
0 when terminated improperly

Explanation: This function expands grayscale JPEG files into GRY format. The roughness of expansion and the magnification value may be specified. If 1 to 3 are specified for roughness of expansion and magnification value, images are rough-expanded, allowing for pseudo-progressive expansion.

Example: Roughness of expansion = 1, magnification value = 1, 16-level (4 bits per pixel) grayscale

```
struct   tIMAGEDATA
{
IMAGEHEAD         stHeader;                /* Image header */
IMAGE             stImage[IMAGE_SIZE];     /* YUV raw data */
}stImageData;
unsigned char     *pucJpgBuf;
unsigned char     *pucImageBuf;
unsigned long     ulWidth, ulHeight;

pucJpgBuf = sample_jpg2;
ulId = ID_GRY;
pucImageBuf = (unsigned char *)stImageData.stImage;
jpgGryDataDecodeRough(pucJpgBuf, pucImageBuf, &ulWidth, &ulHeight, ROUGH_MIN,
MAG_MIN, 4);

stImageData.stHeader.ulId = ulId;
stImageData.stHeader.ulWidth = ulWidth;
stImageData.stHeader.ulHeight = ulHeight;
```

5.3 JPEG33 Library Functions

Object files containing all the necessary functions for expanding, compressing, and converting images on the E0C33 chip are registered in the "jpeg.lib" JPEG33 library file. Link these functions to the user program to implement various image processing functions.
JPEG33 library functions are listed in Table 5.3.1.

Table 5.3.1 JPEG33 Library Functions

Type	Function Name	Description	Header File	Source File
Image format conversion functions	convRgbToYuv( )	Converts RGB to YUV	conv_yuv.h	conv_yuv.c
	convYuvToRgb( )	Converts YUV to RGB		
	convRgbToY( )	Converts RGB to Y	conv_y.h	conv_y.c
	convYToRgb( )	Converts Y to RGB		
	conv4bGryToY( )	Converts 16-level (4 bits per pixel) grayscale to Y	conv_4g.h	conv_4g.c
	convYTo4bGry( )	Converts Y to 16-level (4 bits per pixel) grayscale		
	conv8bGryToY( )	Converts 256-level (8 bits per pixel) grayscale to Y	conv_8g.h	conv_8g.c
	convYTo8bGry( )	Converts Y to 256-level (8 bits per pixel) grayscale		
Pixel magnification functions for pseudo-progressive expansion	resizerOneLine2( )	Magnifies pixels in one color line 2-fold	resizer.h	resizer.c
	resizerOneLine4( )	Magnifies pixels in one color line 4-fold		
	resizerOneLine8( )	Magnifies pixels in one color line 8-fold		
	resizerOneGryLine2( )	Magnifies pixels in one grayscale line 2-fold	resizgry.h	resizgry.c
	resizerOneGryLine4( )	Magnifies pixels in one grayscale line 4-fold		
	resizerOneGryLine8( )	Magnifies pixels in one grayscale line 8-fold		
	resizerLineCopy16( )	Copies color lines (in size of 16 × n bytes)	resizer.h	resizer.c
	resizerLineCopy1( )	Copies color lines (in size of 1 × n bytes)		
	resizerGryLineCopy16( )	Copies grayscale lines (in size of 16 × n bytes)	resizgry.h	resizgry.c
	resizerGryLineCopy1( )	Copies grayscale lines (in size of 1 × n bytes)		
JPEG compression and expansion functions	jpgEncodeInit( )	Initializes JPEG compression	jpegtop.h	–
	jpgPutData( )	Compresses JPEG data		
	jpgEecodeClose( )	Terminates JPEG compression		
	jpgDecodeInit( )	Initializes JPEG data expansion		
	jpgGetData( )	Expands JPEG data		
	jpgDecodeClose( )	Terminates JPEG data expansion		
	jpgMemoryInit( )	Initializes JPEG library memory		

Note: Avoid repeatedly alternating RGB→YUV and YUV→RGB conversions, since such processing will eventually degrade picture quality, owing to calculation errors that occur during conversion.

Specification of each function is explained in the following pages. For sample applications of these functions, see the "jpegtop.c" source file.

5.3.1 Image Data Format Converting Functions

convRgbToYuv( )

Function: Converts RGB to YUV.

Format:
```
void convRgbToYuv(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of RGB data
IMAGE	***destImage**	Pointer to buffer for one line of YUV data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of RGB data (srcImage) into YUV data (destImage). The same buffer may be used for srcImage and destImage.

convYuvToRgb( )

Function: Converts YUV to RGB.

Format:
```
void convYuvToRgb(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of YUV data
IMAGE	***destImage**	Pointer to buffer for one line of RGB data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of YUV data (srcImage) into RGB data (destImage). The same buffer may be used for srcImage and destImage.

convRgbToY( )

Function: Converts RGB to Y.

Format:
```
void convRgbToY(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of RGB data
IMAGE	***destImage**	Pointer to buffer for one line of Y data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of RGB data (srcImage) into Y data (destImage). The same buffer may be used for srcImage and destImage.

convYToRgb( )

Function: Converts Y to RGB.

Format:
```
void convYToRgb(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data
IMAGE	***destImage**	ointer to buffer for one line of RGB data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of Y data (srcImage) into RGB data (destImage). The same buffer may be used for srcImage and destImage.

conv4bGryToY( )

Function: Converts 16-level (4 bits per pixel) grayscale to Y.

Format:
```
void conv4bGryToY(unsigned char *pucSrcBuf, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

unsigned char	***pucSrcBuf**	Pointer to buffer for one line of 16-level (4 bits per pixel) grayscale data
IMAGE	***destImage**	Pointer to buffer for one line of Y data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of 16-level (4 bits per pixel) grayscale data (pucSrcBuf) into Y data (destImage).

convYTo4bGry( )

Function: Converts Y to 16-level (4 bits per pixel) grayscale.

Format:
```
void convYTo4bGry(IMAGE *srcImage, unsigned char *pucDestBuf,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data
unsigned char	***pucDestBuf**	Pointer to buffer for one line of 16-level (4 bits per pixel) grayscale data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of Y data (srcImage) into 16-level (4 bits per pixel) grayscale data (pucDestBuf).

conv8bGryToY( )

Function: Converts 256-level (8 bits per pixel) grayscale to Y.

Format:
```
void conv8bGryToY(unsigned char *pucSrcBuf, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

unsigned char	***pucSrcBuf**	Pointer to buffer for one line of 256-level (8 bits per pixel) grayscale data
IMAGE	***destImage**	Pointer to buffer for one line of Y data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of 256-level (8 bits per pixel) grayscale data (pucSrcBuf) into Y data (destImage).

convYTo8bGry( )

Function: Converts Y to 256-level (8 bits per pixel) grayscale.

Format:
```
void convYTo8bGry(IMAGE *srcImage, unsigned char *pucDestBuf,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data
unsigned char	***pucDestBuf**	Pointer to buffer for one line of 256-level (8 bits per pixel) grayscale data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function converts one line of Y data (srcImage) into 256-level (8 bits per pixel) grayscale data (pucDestBuf).

5.3.2 Pixel Magnifying Functions for Pseudo-progressive Expansion

resizerOneLine2( )

Function: Magnifies pixels in one line of color image 2-fold.

Format:
```
void resizerOneLine2(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of YUV data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of YUV data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of color image (YUV data) 2-fold.

resizerOneLine4( )

Function: Magnifies pixels in one line of color image 4-fold.

Format:
```
void resizerOneLine4(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of YUV data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of YUV data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of color image (YUV data) 4-fold.

resizerOneLine8( )

Function: Magnifies pixels in one line of color image 8-fold.

Format:
```
void resizerOneLine8(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of YUV data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of YUV data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of color image (YUV data) 8-fold.

resizerOneGryLine2( )

Function: Magnifies pixels in one line of grayscale image 2-fold.

Format:
```
void resizerOneLine2(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of Y data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of grayscale image (Y data) 2-fold.

resizerOneGryLine4( )

Function: Magnifies pixels in one line of grayscale image 4-fold.

Format:
```
void resizerOneLine4(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of Y data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of grayscale image (Y data) 4-fold.

resizerOneGryLine8( )

Function: Magnifies pixels in one line of grayscale image 8-fold.

Format:
```
void resizerOneLine8(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer for one line of Y data before magnification
IMAGE	***destImage**	Pointer to buffer for one line of Y data after magnification
unsigned long	**ulWidth**	Length of one line (in pixels) before magnification

Return value: None

Explanation: This function magnifies one line of grayscale image (Y data) 8-fold.

resizerLineCopy16( )

Function: Copies color lines (in 16 × n byte segments).

Format:
```
void resizerLineCopy16(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer from which to copy one line of YUV data
IMAGE	***destImage**	Pointer to buffer to which to copy one line of YUV data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function copies one line of YUV data in 16 × n byte segments from srcImage to destImage.

resizerLineCopy1( )

Function: Copies color lines (in 1 × n byte segments).

Format:
```
void resizerLineCopy1(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer from which to copy one line of YUV data
IMAGE	***destImage**	Pointer to buffer to which to copy one line of YUV data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function copies one line of YUV data in 1 × n byte segments from srcImage to destImage.

resizerGryLineCopy16( )

Function: Copies grayscale lines (in 16 × n byte segments).

Format:
```
void resizerGryLineCopy16(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer from which to copy one line of Y data
IMAGE	***destImage**	Pointer to buffer to which to copy one line of Y data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function copies one line of Y data in 16 × n byte segments from srcImage to destImage.

resizerGryLineCopy1( )

Function: Copies grayscale lines (in 1 × n byte segments).

Format:
```
void resizerGryLineCopy1(IMAGE *srcImage, IMAGE *destImage,
unsigned long ulWidth);
```

Arguments:

IMAGE	***srcImage**	Pointer to buffer from which to copy one line of Y data
IMAGE	***destImage**	Pointer to buffer to which to copy one line of Y data
unsigned long	**ulWidth**	Length of one line (in pixels)

Return value: None

Explanation: This function copies one line of Y data in 1 × n byte segments from srcImage to destImage.

5.3.3 JPEG33 Compression/Expansion Functions

jpgEncodeInit( )

Function: Initializes internal settings to prepare for JPEG compression.

Format:
```
void *jpgEncodeInit(unsigned char *pucJpgBuf, unsigned short uwWidth,
unsigned short uwHeight, int iInterleave, int iQuality, int iDepth);
```

Arguments:

unsigned char	***pucJpgBuf**	Pointer to memory area where JPEG data is held
unsigned short	**uwWidth**	Number of horizontal pixels in image data
unsigned short	**uwHeight**	Number of vertical pixels in image data
int	**iInterleave**	Interleave value (0 to 3) INTER_GRY (=0): Grayscale, Yh:Yv = 1:1 INTER_COLOR1 (=1): Color, Yh:Yv:Uh:Uv:Vh:Vv = 1:1:1:1:1:1 INTER_COLOR2 (=2): Color, Yh:Yv:Uh:Uv:Vh:Vv = 2:1:1:1:1:1 INTER_COLOR3 (=3): Color, Yh:Yv:Uh:Uv:Vh:Vv = 2:2:1:1:1:1
int	**iQuality**	Picture quality index value (1 to 100) Larger values correspond to lower picture quality degradation (and hence better pictures), but at the cost of larger JPEG files. Smaller values correspond to lower picture quality, with more compact JPEG files.
int	**iDepth**	Grayscale depth (4, 8) 4: 16-level (4 bits per pixel) grayscale 8: 256-level (8 bits per pixel) grayscale

Return value: Pointer to JPEG information structure

Explanation: This function initializes internal settings to prepare for JPEG compression.
Specifying a value below 0 for interleave value sets that value to 0. Specifying a value over 3 sets the value to 3. Specifying a value smaller than 1 for the picture quality index value sets it to 1. Specifying a value over 100 sets the value to 100. Specifying a value other than 4 or 8 for grayscale depth sets the value to 8. Specifying grayscale depth is enabled only when the interleave value is set to 0.

jpgPutData( )

Function: Compresses JPEG data.

Format:
```
void jpgPutData(void *pJpeg, IMAGE *srcImage);
```

Arguments:

void	***pJpeg**	Pointer to JPEG information structure
IMAGE	***srcImage**	Pointer to buffer for one line of YUV data to be compressed

Return value: None

Explanation: This function compresses one line of YUV data into JPEG format and outputs JPEG data specified by pJpeg to the memory area where it is to be held.

jpgEncodeClose( )

Function: Terminates JPEG compression.

Format:
```
unsigend long jpgEncodeClose(void *pJpeg);
```

Arguments:

void	***pJpeg**	Pointer to JPEG information structure

Return value: JPEG data size

Explanation: This function performs termination processing to close JPEG data and free memory after compression.

jpgDecodeInit( )

Function: Initializes internal settings to prepare for JPEG data expansion.

Format:
```
void *jpgDecodeInit(unsigned char *pucJpgBuf, unsigned short *puwWidth,
unsigned short *puwHeight, int *piColor, int iRough, int iDepth);
```

Arguments:

unsigned char	***pucJpgBuf**	Pointer to memory area where JPEG data is held
unsigned short	***puwWidth**	Number of horizontal pixels in image data
unsigned short	***puwHeight**	Number of vertical pixels in image data
int	***piColor**	Pointer to specified grayscale depth or color (1, 3) 1: Grayscale 3: Color
int	**iRough**	Roughness of expansion value (0 to 3) ROUGH_NONE (=0): No roughness of expansion (1:1) ROUGH_MIN (=1): Expanded with x2 roughness (1:2) ROUGH_MID (=2): Expanded with x4 roughness (1:4) ROUGH_MAX (=3): Expanded with x8 roughness (1:8)
int	**iDepth**	Grayscale depth (4, 8) 4: 16-level (4 bits per pixel) grayscale 8: 256-level (8 bits per pixel) grayscale

Return value: Pointer to JPEG information structure (0 is returned for illegal JPEG formats)

Explanation: This function initializes internal settings to prepare for JPEG data expansion.
Specifying a value below 0 for roughness of expansion sets the value to 0. Specifying a number over 3 sets the value to 3. For ordinary expansion, set roughness of expansion to 0. Specifying a value other than 4 or 8 for grayscale depth sets it to 8. Grayscale depth may be specified only when piColor = 1.

jpgGetData( )

Function: Expands JPEG data.

Format:
```
void jpgGetData(void *pJpeg, IMAGE *destImage);
```

Arguments:

void	***pJpeg**	Pointer to JPEG information structure
IMAGE	***destImage**	Pointer to buffer in which one line of expanded YUV data is held

Return value: None

Explanation: This function expands one line of JPEG data specified by pJpeg and outputs the expanded YUV data to the memory area specified by destImage.

jpgDecodeClose( )

Function: Terminates JPEG data expansion.

Format:
```
unsigend long jpgDecodeClose(void *pJpeg);
```

Arguments:

void	***pJpeg**	Pointer to JPEG information structure

Return value: None

Explanation: This function performs termination processing to close JPEG data and free memory after expansion.

jpgMemoryInit( )

Function: Initializes the memory used for the JPEG33 library.

Format: `void jpgMemoryInit(void);`

Arguments: None

Return value: None

Explanation: This function initializes to 0 the memory locations in the static memory area used by the JPEG33 library.

5.4 Image Data Structure

The following describes the structure of various image data files for each format handled by the JPEG33 tools and library. Each format begins with a 12-byte header, followed by image data. The header consists of a 4-byte ID that indicates the image format, 4-byte information indicating the number of horizontal pixels (1 to 65,535), and 4-byte information indicating the number of vertical pixels (1 to 65,535).

RGB file

ID (=0)									
Number of horizontal pixels									
Number of vertical pixels									
R(1,1)	G(1,1)	B(1,1)	R(2,1)	G(2,1)	B(2,1)	. . .	R(w,1)	G(w,1)	B(w,1)
R(1,2)	G(1,2)	B(1,2)	R(2,2)	G(2,2)	B(2,2)	. . .	R(w,2)	G(w,2)	B(w,2)
				:		. . .			
R(1,h)	G(1,h)	B(1,h)	R(2,h)	G(2,h)	B(2,h)	. . .	R(w,h)	G(w,h)	B(w,h)

The RGB format ID is 0. This value is defined in "jpegtop.h" as ID_RGB. RGB data consists of R, G, and B values, each assigned one byte, together comprising 3 bytes per pixel. Image data begins with the upper left-most pixel and is comprised of a repeated sequence of R, G, and B values, until the end of the line. This sequence is then repeated for each line until the last line.

YUV file

ID (=1)									
Number of horizontal pixels									
Number of vertical pixels									
Y(1,1)	U(1,1)	V(1,1)	Y(2,1)	U(2,1)	V(2,1)	. . .	Y(w,1)	U(w,1)	V(w,1)
Y(1,2)	U(1,2)	V(1,2)	Y(2,2)	U(2,2)	V(2,2)	. . .	Y(w,2)	U(w,2)	V(w,2)
				:		. . .			
Y(1,h)	U(1,h)	V(1,h)	Y(2,h)	U(2,h)	V(2,h)	. . .	Y(w,h)	U(w,h)	V(w,h)

The YUV format ID is 1. This value is defined in "jpegtop.h" as ID_YUV. YUV data consists of Y, U, and V, each assigned one byte, together comprising 3 bytes per pixel. Image data begins with the upper left-most pixel and is comprised of a repeated sequence of Y, U, and V values, until the end of the line. This sequence is then repeated for each line until the last line.

256-grayscale level GRY file

ID (=2)			
Number of horizontal pixels			
Number of vertical pixels			
GRY(1,1)	GRY(2,1)	. . .	GRY(w,1)
GRY(1,2)	GRY(2,2)	. . .	GRY(w,2)
		. . .	
GRY(1,h)	GRY(2,h)	. . .	GRY(w,h)

The ID for the 256-grayscale level GRY format is 2. This value is defined in "jpegtop.h" as ID_GRY8. 256-level grayscale data has one byte assigned per pixel. Image data begins with the upper left-most pixel, proceeding to the end of the first line, then resuming with the following line, until the last line is reached.

16-grayscale level GRY file

ID (=3)						
Number of horizontal pixels						
Number of vertical pixels						
GRY(1,1)	GRY(2,1)	GRY(3,1)	GRY(4,1)	. . .	GRY(w-1,1)	GRY(w,1)
GRY(1,2)	GRY(2,2)	GRY(3,2)	GRY(4,2)	. . .	GRY(w-1,2)	GRY(w,2)
				. . .		
GRY(1,h)	GRY(2,h)	GRY(3,h)	GRY(4,h)	. . .	GRY(w-1,h)	GRY(w,h)

The ID for the 16-grayscale level GRY format is 3. This value is defined in "jpegtop.h" as ID_GRY4. 16-level grayscale data has four bits per pixel; each byte carries data for two pixels. Within this byte, the 4 high-order bits constitute the left side and the 4 low-order bits constitute the right side of two consecutive pixels. Only when the last byte for each line codes for a single pixel, the 4 low-order bits are 0b0000.

File header structure

The structure for handling file headers is declared in "jpegtop.h," as follows:

```
typedef struct  tIMAGEHEAD
{
        unsigned long    ulId;
        unsigned long    ulWidth;
        unsigned long    ulHeight;
}IMAGEHEAD;
```

RGB/YUV data union

The union processing for handling RGB and YUV format data structures is declared in "jpegtop.h" as follows:

```
typedef union   tIMAGE
{
        struct {
                unsigned char    bY, bU, bV;
        }stYUV;
        struct {
                unsigned char    bR, bG, bB;
        }stRGB;
}IMAGE;
```

5.5 Processing Speed

The processing speed of the JPEG33 library varies according to the images, parameters, and hardware environment involved. The following shows how to calculate an approximate processing speed. The calculated speed must eventually be verified by actual measurement on your system.

<System condition example>
- Operating speed of the EOC33 chip: 20 MHz
- Working RAM area: External RAM (16 bits wide, 1 wait cycle)
- Program area: External ROM (16 bits wide, 1 wait cycle)
- Stack area: Internal RAM

JPEG compression and expansion

(1) The JPEG compression and expansion speeds are approximately the same.

(2) When measured under the above system conditions and using JPEG33 library default conditions (horizontal/vertical interleave = 1/2, picture quality index = 75), the time required for compressing and expanding a 320 × 240-dot color image is approximately 2.0 seconds each.

(3) The processing speed is proportional to the number of pixels. For example, a 640 × 480-dot image requires approximately four times the above time.

(4) The processing speed varies with the parameters. If the image is thinned out by 1/2 in only the horizontal direction when compressed, the processing speed is about 1.3 times faster than that when the image is not thinned out. Furthermore, if the image is thinned out in also the vertical direction, the processing speed is about 1.5 times faster than that when the image is not thinned out. Similarly, if the picture quality index is set to 100, the processing speed is about 1.6 times slower than when the index is set to 75. When the index is set to 25, the processing speed is about 1.1 times faster than when the index is set to 75.
When the image is expanded after setting the roughness and scaling of expansion to the same multiplying factor, the processing speed is about 2.1 times as fast with the ×2 multiplying factor, about 3.3 times as fast with the ×4 multiplying factor, and about 4.1 times as fast with the ×8 multiplying factor. Furthermore, if the image is not enlarged, the processing speed is about 10% as fast.

(5) Gray-scale images are processed faster than color images by about 1.5 times when compressed, and by about 1.3 times when expanded.

(6) When the number of wait cycles in external RAM is increased by 1 under the above system conditions, the processing speed increases by 0.15 seconds.

(7) When the number of wait cycles in external ROM is increased by 1 under the above system conditions, the processing speed increases by 0.75 seconds.

(8) When "conv_yuv.o", "jfdictin.o", and "jstripin.o" are executed in the internal RAM, the processing time in (2), i.e., 2.0 seconds, is reduced to about 1.5 seconds. When all libraries are placed into the internal ROM, the time is further reduced to about 1.1 seconds. Moreover, when the BSS section of "jmemin.o" is mapped into the internal RAM, the processing time is reduced to about 1.0 second. (Refer to Section 5.6, "Techniques for Improving Processing Speed".)

(9) When the operating clock frequency is raised without changing the number of wait cycles, the processing speed increases in proportion to the clock frequency.

YUV-RGB conversion

(1) The YUV→RGB and RGB→YUV conversion times in the JPEG33 library under the above system conditions are about 0.6 seconds each.

(2) When the number of wait cycles in external RAM is increased by 1 under the above system conditions, the conversion time increases by 0.02 seconds.

(3) When the number of wait cycles in external ROM is increased by 1 under the above system conditions, the conversion time increases by 0.28 seconds.

(4) When all programs are placed into the internal ROM, the conversion time in (1) is reduced to about 0.3 seconds.

5.6 Techniques for Improving Processing Speed

By executing library functions after mapping several object files from "jpeg.lib" into internal RAM, you can increase library processing speed by up to 30%. Use the linker's U-section function to map to internal RAM. The processing required is described below.

1. Extract the necessary objects from the library.
   They can be restored to the object files using the librarian lib33 -x option.
   Example: `lib33 -x jpeg.lib conv_yuv.o`

   * The "lib" directory contains several selected sample objects. These objects may be used directly.

2. Write the following in the linker command file.
   -objsym ; Create object symbol
   -section <*name*> ; Create section symbol
   -ucode <*name*> { <*object file*> [<*object file*>....] } ; Map to U section
   Example:
   ```
   -objsym
   -section CACHE3
   -ucode CACHE3 {..\lib\conv_yuv.o}
   ```
   Look at the linked map file. You will find that the execution addresses for "conv_yuv.o" have been mapped to internal RAM.
   Example: Map file
   ```
   Address     Vaddress    Size        File
   00c19c14    00000f8c    000005b0    ..\lib\conv_yuv.o
               ~~~~~~~~
   ```

3. Modify the source statements so that the necessary object code is transferred to internal RAM at system boot or before calls are made to library functions.
   Example:
   ```
   xld.w %r12, __START_CACHE3
   xld.w %r13, __START_conv_yuv_code
   xld.w %r14, __SIZEOF_conv_yuv_code
   call HCOPY_LOOP
   ```

Compact objects that can improve processing speed are listed in Table 5.6.1. Choose the objects you want to map to RAM, according to the amount of available internal RAM.

Table 5.6.1 Objects Used to Improve Processing Speed

Object	Size	Description
conv_yuv.o	Approx. 1.5 KB	Converts RGB↔YUV image formats
jfdctin.o	Approx. 1.2 KB	Converts DCT
jstripin.o	Approx. 1.1 KB	Samples image data

These object files are found in the "jpeglib\lib\" directory.

In addition to the files above, "jpeglib\jmemin.o" in which the internal buffer RAM is declared is provided. Mapping this object in the internal RAM slightly improves processing speed.
Example: Write the following in the linker command file. (demo4.cm)

```
-bss 0x0000730 {..\lib\jmemin.o}
```

The RAM size used for this mapping is approximately 2800 bytes.

5.7 Library Memory Requirements

The JPEG33 library uses memory for JPEG compression/expansion and conversion in the following way:

CODE section: Approx. 24.5K bytes

BSS section: Approx. 10K bytes[*1] + number of horizontal pixels[*2] × 3 × (16 + 1)[*3]
*1: Used inside jpeg.lib
*2: Defined in jpegtop.h (JMEMORY_IMAGE_WIDTH)
*3: Required when working with grayscale images
This is approximately 25K bytes when working with an image 320 pixels wide, and approximately 40K bytes for an image 640 pixels wide.

Stack: Approx. 400 bytes

5.8 Precautions

Avoid repeatedly alternating RGB→YUV and YUV→RGB conversions, since such processing will eventually degrade picture quality, owing to calculation errors that occur during conversion.

EPSON International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.

- HEADQUARTERS -
1960 E. Grand Avenue
El Segundo, CA 90245, U.S.A.
Phone: +1-310-955-5300 Fax: +1-310-955-5400

- SALES OFFICES -

West
150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-408-922-0200 Fax: +1-408-922-0238

Central
101 Virginia Street, Suite 290
Crystal Lake, IL 60014, U.S.A.
Phone: +1-815-455-7630 Fax: +1-815-455-7633

Northeast
301 Edgewater Place, Suite 120
Wakefield, MA 01880, U.S.A.
Phone: +1-781-246-3600 Fax: +1-781-246-5443

Southeast
3010 Royal Blvd. South, Suite 170
Alpharetta, GA 30005, U.S.A.
Phone: +1-877-EEA-0020 Fax: +1-770-777-2637

EUROPE

EPSON EUROPE ELECTRONICS GmbH

- HEADQUARTERS -
Riesstrasse 15
80992 Munich, GERMANY
Phone: +49-(0)89-14005-0 Fax: +49-(0)89-14005-110

- GERMANY -
SALES OFFICE
Altstadtstrasse 176
51379 Leverkusen, GERMANY
Phone: +49-(0)2171-5045-0 Fax: +49-(0)2171-5045-10

- UNITED KINGDOM -
UK BRANCH OFFICE
Unit 2.4, Doncastle House, Doncastle Road
Bracknell, Berkshire RG12 8PE, ENGLAND
Phone: +44-(0)1344-381700 Fax: +44-(0)1344-381701

- FRANCE -
FRENCH BRANCH OFFICE
1 Avenue de l' Atlantique, LP 915 Les Conquerants
Z.A. de Courtaboeuf 2, F-91976 Les Ulis Cedex, FRANCE
Phone: +33-(0)1-64862350 Fax: +33-(0)1-64862355

ASIA

- CHINA -
EPSON (CHINA) CO., LTD.
28F, Beijing Silver Tower 2# North RD DongSanHuan
ChaoYang District, Beijing, CHINA
Phone: 64106655 Fax: 64107319

SHANGHAI BRANCH
4F, Bldg., 27, No. 69, Gui Jing Road
Caohejing, Shanghai, CHINA
Phone: 21-6485-5552 Fax: 21-6485-0775

- HONG KONG, CHINA -
EPSON HONG KONG LTD.
20/F., Harbour Centre, 25 Harbour Road
Wanchai, HONG KONG
Phone: +852-2585-4600 Fax: +852-2827-4346
Telex: 65542 EPSCO HX

- TAIWAN -
EPSON TAIWAN TECHNOLOGY & TRADING LTD.
10F, No. 287, Nanking East Road, Sec. 3
Taipei, TAIWAN
Phone: 02-2717-7360 Fax: 02-2712-9164
Telex: 24444 EPSONTB

HSINCHU OFFICE
13F-3, No. 295, Kuang-Fu Road, Sec. 2
HsinChu 300, TAIWAN
Phone: 03-573-9900 Fax: 03-573-9169

- SINGAPORE -
EPSON SINGAPORE PTE., LTD.
No. 1 Temasek Avenue, #36-00
Millenia Tower, SINGAPORE 039192
Phone: +65-337-7911 Fax: +65-334-2716

- KOREA -
SEIKO EPSON CORPORATION KOREA OFFICE
50F, KLI 63 Bldg., 60 Yoido-dong
Youngdeungpo-Ku, Seoul, 150-763, KOREA
Phone: 02-784-6027 Fax: 02-767-3677

- JAPAN -
SEIKO EPSON CORPORATION
ELECTRONIC DEVICES MARKETING DIVISION

Electronic Device Marketing Department
IC Marketing & Engineering Group
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5816 Fax: +81-(0)42-587-5624

ED International Marketing Department Europe & U.S.A.
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5812 Fax: +81-(0)42-587-5564

ED International Marketing Department Asia
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5814 Fax: +81-(0)42-587-5110

In pursuit of **"Saving" Technology**, Epson electronic devices.
Our lineup of semiconductors, liquid crystal displays and quartz devices
assists in creating the products of our customers' dreams.
Epson IS energy savings.

EPSON

SEIKO EPSON CORPORATION
ELECTRONIC DEVICES MARKETING DIVISION

■ EPSON Electronic Devices Website

http://www.epson.co.jp/device/

Issue APRIL 1999, Printed in Japan Ⓜ A