USER'S MANUAL

µSAP703000-B03

MIDDLEWARE FOR JPEG
(PRELIMINARY)

APPLICABLE DEVICE
V850 FAMILY™

Document No. U10684EJ1V0UM00 (1st edition)
Date Published March 1996 P
Printed in Japan

[illegible]

PREFACE

Users This manual is aimed at those users involved in the design and development of application systems based on the V800 series.

Purpose The purpose of this manual is to help users to understand the functions of the µSAP703000-B03.

Organization This manual includes the following:

- Overview
- Library specifications
- Source lists of sample programs

Reading this manual The “µSAP703000-B03” is referred to as the “AP703000-B03” throughout this manual.

Notation

Note : Explanation of item indicated in the text
Caution : Information to which the user should afford special attention
Remark : Supplementary information
Numeric values : Binary : xxxx or xxxxB
Decimal : xxxx
Hexadecimal : 0xXXXX
Units for representing powers of 2 (address space or memory space):
K (kilo) : $2^{10} = 1024$
M (mega) : $2^{20} = 1024^{2}$

Related documents The following tables list related documents. Note that some documents may be preliminary editions, although this is not indicated in this manual.

- **Documents related to the V850 family**

Document name Product name	Data sheet	User’s manual	
		Hardware	Architecture
V851™	IP-8971	IEU-1411	U10243E
V852™	To be determined	U10038E	

- **Documents related to development tools**

Document name		Document No.
IE-703000-MC User's Manual		Scheduled for release
CA850 User's Manual	Operation (Windows^{TM}-based)	Scheduled for release
	Operation (UNIX^{TM}-based)	U10553EJ
	Assembly language	U10543EJ
	C	EEU-1545
ID850 User's Manual	Operation (Windows-based)	Scheduled for release
	Installation (Windows-based)	Scheduled for release
SM850 User's Manual	Operation (Windows-based)	To be determined
	Installation (Windows-based)	To be determined
RX850 User's Manual		To be determined
AZ850 User's Manual		To be determined

- **Documents related to tools produced by Green Hills Software^{TM}, Inc. (GHS)**
  For more information about GHS tools and related documents, contact:

Green Hills Software, Inc.
One Cramberry Hill Telephone (617) 862-2002
Lexington, MA02173 Fax (617) 863-2633
USA

CONTENTS

LIST OF FIGURES (1/2)

LIST OF FIGURES (2/2)

LIST OF TABLES (1/2)

LIST OF TABLES (2/2)

CHAPTER 1 OVERVIEW

1.1 MIDDLEWARE

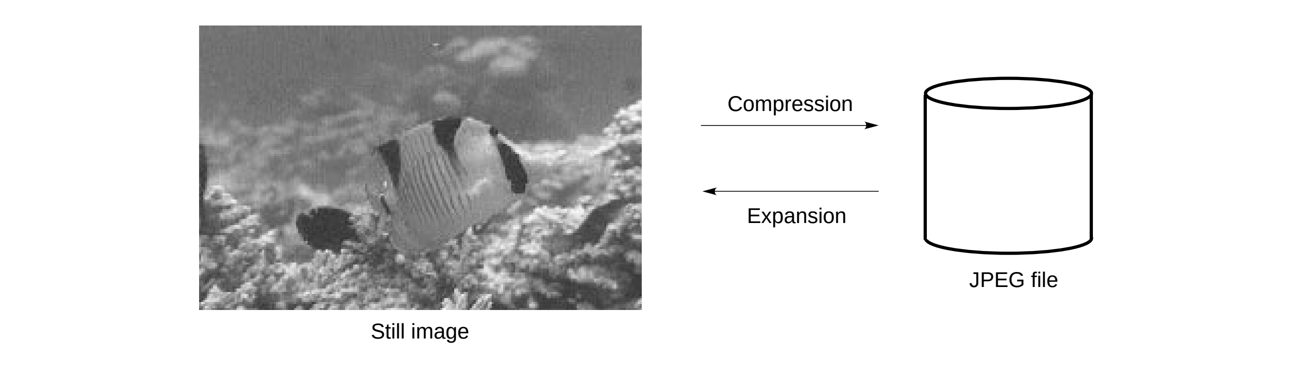
Middleware is a software group that has been tuned to fully exploit the performance of a processor. The software implements processing that is conventionally performed by hardware. The advent of high-performance RISC (reduced instruction set computer) processors has spawned the concept of middleware, with which processing can be realized with ROM/RAM alone, without the need for dedicated hardware.

NEC supplies system solutions that support a wide range of user needs by providing human-machine interface and signal processing technologies in the form of middleware.

1.2 JPEG

JPEG stands for Joint Photographic Experts Group, an international still image compression/expansion standard, established in 1991. This standard is laid down in documents ISO/IEC 10918-1 and 2.

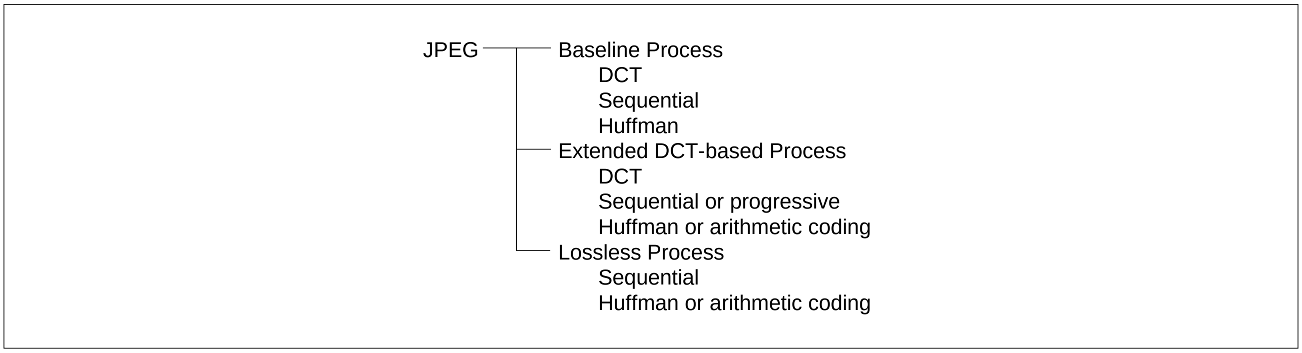
Figure 1-1. Image Compression/Expansion



1.2.1 Overview

There are several versions of the JPEG standard, such as progressive JPEG, in which an outline of the image appears first, detail being added subsequently. Lossless JPEG can completely restore an image to the state existing before compression. These versions, however, are not in general use. The AP703000-B03 supports only the most commonly used version, called Baseline DCT.

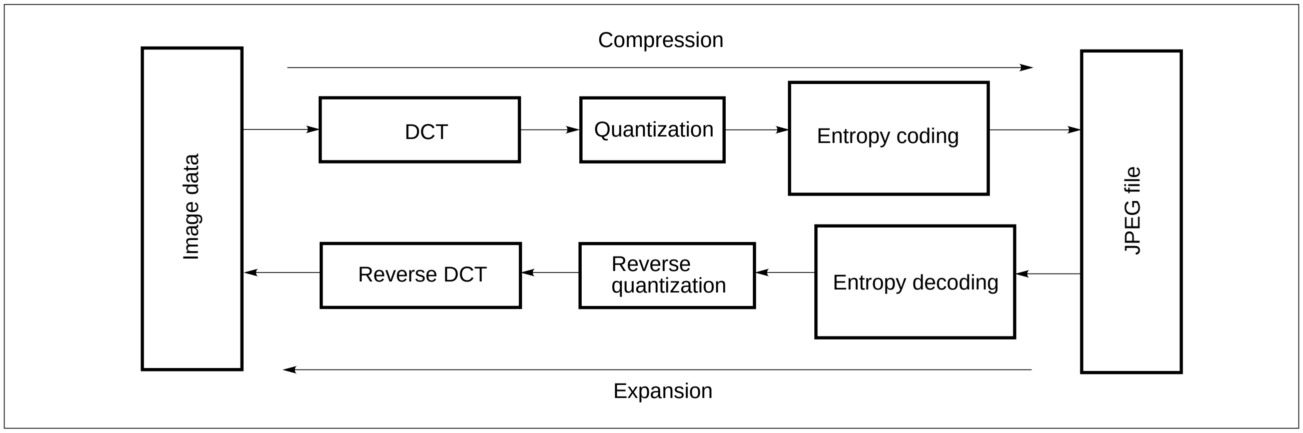
Figure 1-2. JPEG Versions


JPEG
Baseline Process
DCT
Sequential
Huffman
Extended DCT-based Process
DCT
Sequential or progressive
Huffman or arithmetic coding
Lossless Process
Sequential
Huffman or arithmetic coding

(1) Flow of JPEG processing

JPEG compression involves compressing data in three steps: **<1>** DCT, **<2>** quantization, and **<3>** entropy compression. JPEG expansion involves reproducing a compressed image by applying the reverse of the above procedure: **<1>** entropy expansion, **<2>** reverse quantization, and **<3>** reverse DCT.

Figure 1-3. JPEG Processing



DCT (discrete cosine transform) processing involves the disassembly of frequencies. Quantization reduces the volume of information by eliminating, from the data obtained as a result of DCT (i.e., data whose frequency has been disassembled), those frequency components that humans cannot sense. Entropy encoding is generally known as reversible compression/expansion, while baseline DCT uses a technology based on Huffman encoding.

The AP703000-B03 performs DCT and quantization as part of the same function. Similarly, entropy decoding and reverse quantization are performed as part of the same function. This increases the processing speed.

(2) YCbCr/RGB

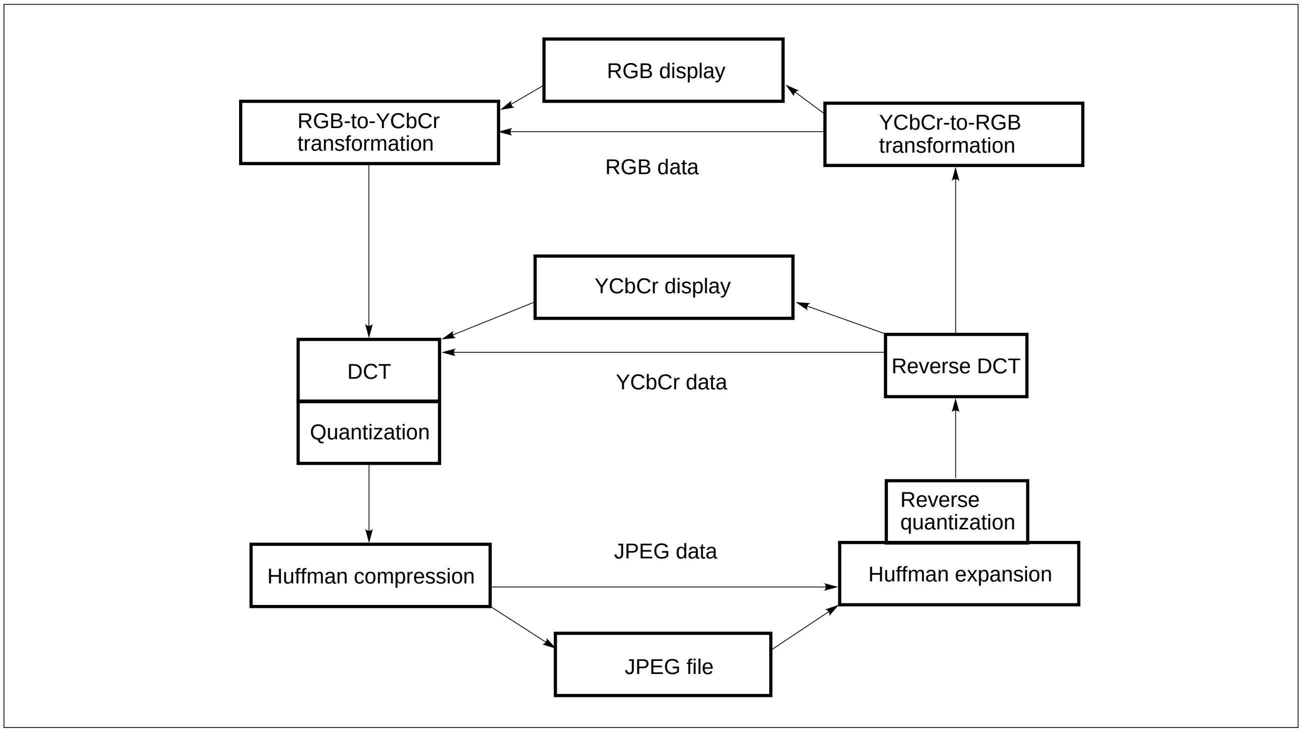
JPEG compresses/expands images in YCbCr color space. If the image data is not YCbCr but RGB, processing to transform the RGB data into YCbCr for compression, or that to transform YCbCr data into RGB before displaying the result of expansion, is added.

The Y of YCbCr is luminance (brightness index), and Cb/Cr is chrominance, a color difference (Cb is the difference in color tone between green and blue, while Cr is the difference in color tone between green and red). Transformation between YCbCr and RGB can be illustrated as follows:

$$\begin{bmatrix} Y + 0x80 \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.29900 & 0.58700 & 0.11400 \\ -0.16874 & -0.33126 & 0.50000 \\ 0.50000 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.40200 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.77200 & 0 \end{bmatrix} \begin{bmatrix} Y + 0x80 \\ Cb \\ Cr \end{bmatrix}$$

Figure 1-4. Outline of JPEG Processing



(3) Sampling and MCU

The minimum unit in which JPEG processing is performed is called an MCU (minimum coded unit). The MCU is separated into Y/Cb/Cr in units of 8 x 8 pixels, each of which is called a block.

Obtaining four blocks of Y, one block of Cb, and one block of Cr from one MCU can be expressed as a “sampling ratio of 4:1:1.” Similarly, when obtaining two blocks of Y, one block of Cb, and one block of Cr from one MCU, the sampling ratio is said to be 4:2:2. When obtaining one block each of Y, Cb, and Cr from one MCU, the sampling ratio is 4:4:4.

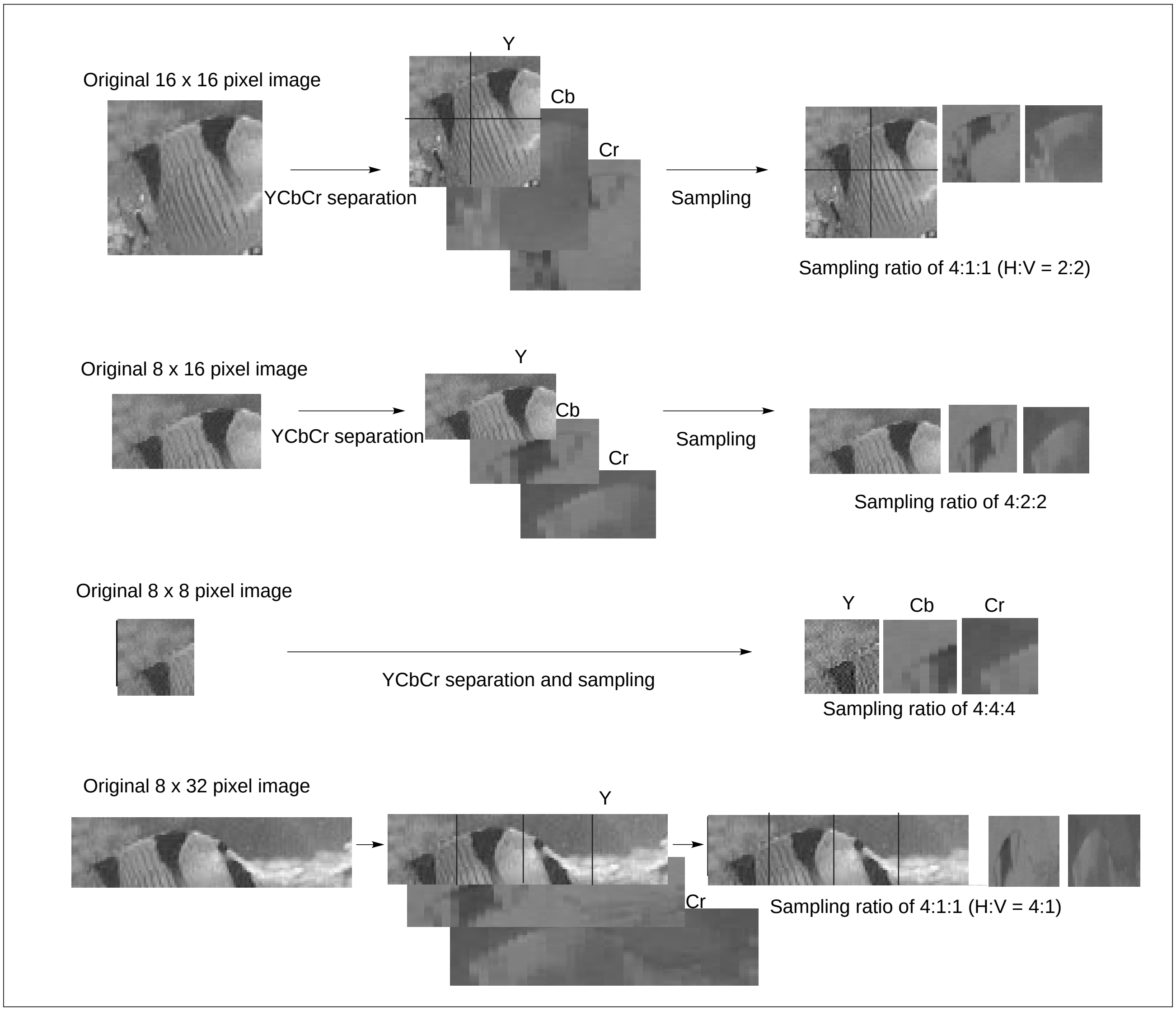
Table 1-1. Sampling Ratio and MCU

MCU	Sampling ratio	Block
Vertical 16 pixels Horizontal 16 pixels	4:1:1 (H:V = 2:2)	Y: 4 blocks Cb: 1 block, Cr: 1 block
Vertical 8 pixels Horizontal 32 pixels	4:1:1 (H:V = 4:1)	Y: 4 blocks Cb: 1 block, Cr: 1 block
Vertical 8 pixels Horizontal 16 pixels	4:2:2	Y: 2 blocks Cb: 1 block, Cr: 1 block
Vertical 8 pixels Horizontal 8 pixels	4:4:4	Y: 1 block Cb: 1 block, Cr: 1 block

Although the JPEG standard covers sampling ratios other than those listed in Table 1-1, the AP703000-B03 supports only those sampling ratios listed in the table.

JPEG compression starts by dividing the image in this MCU units into grids. Conversely, JPEG expansion involves arranging the processing result for each MCU in a manner exactly like paving a floor with tiles. For example, an image is vertically and horizontally divided into 16-pixel units, each at a sampling ratio of 4:1:1 (H:V = 2:2). Next, the 16 x 16 pixel image is separated into Y, Cb, and Cr components, and the Y component is divided into four blocks, each block consisting of 8 x 8 pixels. For the Cb and Cr components, an 8 x 8 pixel image is created from the 16 x 16 pixel image. At this time, the vertically and horizontally adjacent 4 pixels are averaged. This is called “thinning out.”

Figure 1-5. Sampling of Image


Original 16 x 16 pixel image
Y
Cb
Cr
YCbCr separation
Sampling
Sampling ratio of 4:1:1 (H:V = 2:2)
Original 8 x 16 pixel image
Y
Cb
Cr
YCbCr separation
Sampling
Sampling ratio of 4:2:2
Original 8 x 8 pixel image
Y
Cb
Cr
YCbCr separation and sampling
Sampling ratio of 4:4:4
Original 8 x 32 pixel image
Y
Cr
Sampling ratio of 4:1:1 (H:V = 4:1)

With JPEG compression, a sampling ratio of 4:1:1 is used more often than 4:4:4.
At a sampling ratio of 4:1:1, the chrominance component is subjected to less processing than the luminance component. This is because the human eye is more sensitive to changes in brightness than changes in color, such that a high compression ratio can be realized by omitting that information which is difficult for the human eye to detect.

As an example, let’s consider the case in which an image consisting of 640 x 480 pixels is compressed. To compress this image at a sampling ratio of 4:1:1 (H:V = 2:2), it is divided by 16 pixels both horizontally and vertically, giving 40 horizontal segments and 30 vertical segments. Six blocks are extracted from each MCU: four blocks of the Y component, one block of the Cb component, and one block of Cr component. Consequently, 7200 blocks (= 40 x 30 x 6) are obtained from the entire image. To these 7200 blocks, DCT, quantization, and Huffman compression are applied in sequence.

Table 1-2. Sampling Ratio and Block

Sampling ratio	640 x 480 pixels		Number of blocks per MCU	Total number of blocks
	Horizontal	Vertical		
4:1:1 (H:V = 2:2)	40 segments	30 segments	6	7200
4:1:1 (H:V = 4:1)	20 segments	60 segments	6	7200
4:2:2	40 segments	60 segments	4	9600
4:4:4	80 segments	60 segments	3	14400

As is evident from the above table, more blocks are needed at a sampling ratio of 4:4:4 than at 4:1:1. The greater the number of blocks, the more processing time is required. Moreover, the size of the resulting JPEG file also increases.

In JPEG compression, processing is performed on a block-by-block basis after sampling. DCT, quantization, and entropy encoding are performed based on the information to which of Y or Cb/Cr a given block belongs.

In JPEG expansion, the result is obtained in units of blocks once entropy decoding, reverse quantization, and reverse DCT have been completed.

(4) DCT

DCT transformation uses the following expression:

DCT

$$F(u, v) = \frac{2C(u)C(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \, cos \left\{ \frac{(2i+1)u\pi}{2N} \right\} \, cos \left\{ \frac{(2j+1)v\pi}{2N} \right\}$$

Reverse DCT

$$f(i, j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \, cos \left\{ \frac{(2i+1)u\pi}{2N} \right\} \, cos \left\{ \frac{(2j+1)v\pi}{2N} \right\}$$

$$\begin{aligned} C(w) &= \frac{1}{\sqrt{2}} \quad (w = 0) \\ &= 1 \quad\quad (w \neq 0) \end{aligned}$$

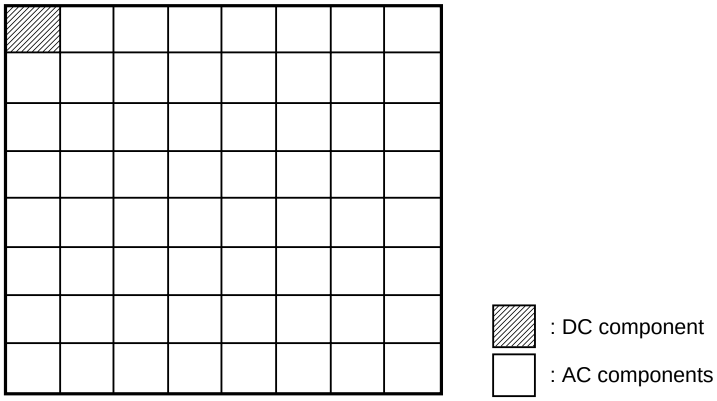
Generally, this DCT is applied to 8 x 8 elements with signal processing techniques such as JPEG and MPEG.

DCT disassembles a frequency of *cos* ($n\pi$/16) (where n = 0, 1, 2, ... 7) in both the vertical and horizontal directions.

Generally, relatively few elements of a natural image, such as a photograph, have values, the other elements tending to have values close to zero when the frequency is disassembled in this way. Even by approximating those elements having a value close to zero with zero, an image close to the original can be produced by using the remaining elements. However, the differences between the original image and an image created in this way are barely visible to the human eye.

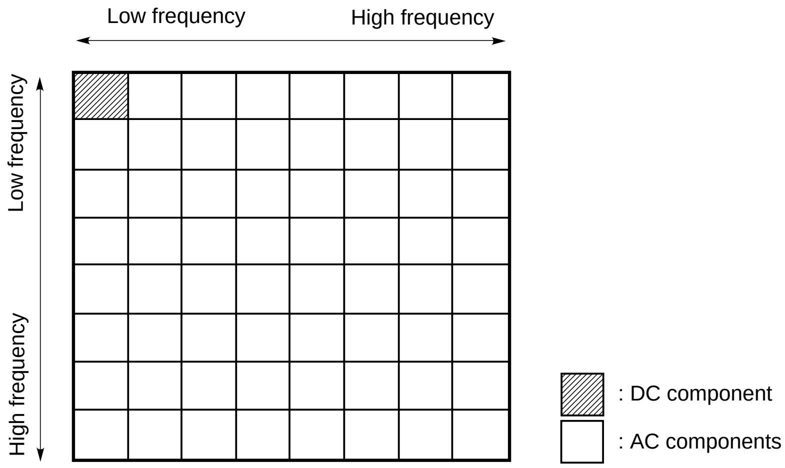
When image data of 8 x 8 pixels is transformed by means of DCT, the first element indicates the average value for the entire matrix, while the other 63 elements indicate the distortion of color in the matrix. Because of the difference in nature between the first element in the matrix and the other 63 elements, the first element is called a DC (direct current) component, while the other 63 elements are called AC (alternating current) components.

Figure 1-6. Matrix Components



In an 8 x 8 matrix after the application of DCT, the low-frequency components are concentrated at the left and top edges, while the high-frequency components are concentrated at the right and bottom edges. If the original image exhibits few changes in tone, such as those that approach monochrome, a matrix of only low-frequency components (with almost all the high-frequency values being 0) can be obtained. Conversely, with a delicate image such as a diced pattern, a matrix with several high frequencies can be obtained.

Figure 1-7. Distribution of Frequency Components



(5) Quantization and zigzag scan

It is said that the human eye can barely recognize changes in high-frequency components but can easily recognize the most subtle changes in low-frequency components. To increase the compression ratio, JPEG compression divides low-frequency components by a small value and high-frequency components by a greater value. This processing is called quantization. To expand compressed data, the data is multiplied by the same value by which it was divided (reverse quantization). However, the data cannot be fully restored by applying quantization and reverse quantization (cannot be reversed). This is because, when data is quantized, only the quotient resulting from division is used as information, the remainder being ignored. In this way, the JPEG standard enables an increase in the compression ratio without visibly degrading the image.

Figure 1-8. Quantized Matrix and Quantization

Example of quantized matrix

$$Q(i, j) = \begin{pmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{pmatrix}$$

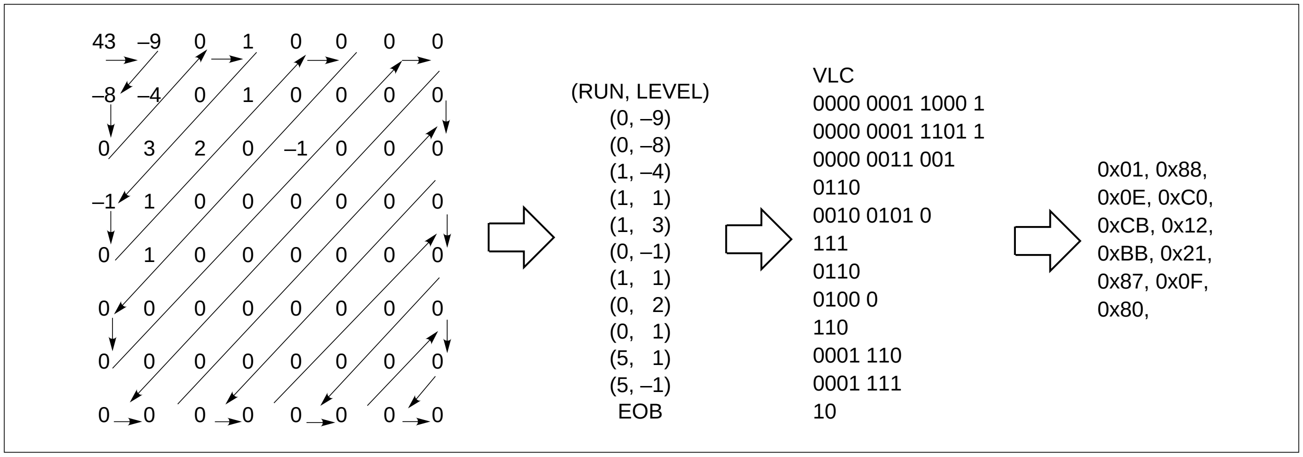
Example of quantized 8 x 8 matrix

$$\begin{pmatrix} 43 & -9 & 0 & 1 & 0 & 0 & 0 & 0 \\ -8 & -4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Data obtained by applying DCT to a block of the original image is notable in that the data of the Y component differs from that of the Cb/Cr component. Therefore, JPEG uses two types of quantized matrixes for the Y and Cb/Cr components, respectively (in some cases, only one quantized matrix is used). These quantized matrixes can be defined independently for each image (JPEG file). Information relating to these quantized matrixes is stored as a DQT segment in the header of the JPEG file.

As shown by the example in Figure 1-8, if most of the values in the obtained matrix are 0, the information that “there is a sequence of n zeroes followed by a value that is not zero” is interpreted to increase the compression rate. The JPEG standard refers to this “sequence of zeroes” as “the length of zero run.” The non-zero values in the matrix obtained as a result of quantization gather in the upper left part of the matrix most of the time. For this reason, the length of the zero run is counted by JPEG in the sequence illustrated below (zigzag scan).

Figure 1-9. Zigzag Scan and Coding



43	–9	0	1	0	0	0	0
–8	–4	0	1	0	0	0	0
0	3	2	0	–1	0	0	0
–1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(RUN, LEVEL)	VLC
(0, –9)	0000 0001 1000 1
(0, –8)	0000 0001 1101 1
(1, –4)	0000 0011 001
(1, 1)	0110
(1, 3)	0010 0101 0
(0, –1)	111
(1, 1)	0110
(0, 2)	0100 0
(0, 1)	110
(5, 1)	0001 110
(5, –1)	0001 111
EOB	10

0x01, 0x88,
0x0E, 0xC0,
0xCB, 0x12,
0xBB, 0x21,
0x87, 0x0F,
0x80,

(6) Entropy encoding

With Baseline DCT version of JPEG, entropy encoding that uses Huffman coding is performed. In entropy encoding, the absolute values and distribution of the DC and AC components differ.

While the absolute value of an AC component is relatively low, the absolute value of the DC component tends to be great. This is because the DC component is the average value of a given block. With JPEG, a difference between the DC component of the current block and the DC component of the preceding block is calculated for each of the Y, Cb, and Cr components, and this difference is compressed by means of entropy when the DC component is compressed. For the AC coefficients, the combination of the length of the zero run and the value of a non-zero coefficient (LEVEL value) is compressed by means of entropy. The compressed code is called a VLC (Variable Length Code).

In JPEG compression, the DC and AC components are compressed in accordance with different Huffman encoding conventions. This is referred to as “the DC and AC components using different Huffman tables.” Moreover, like quantization, because the distribution of values differs between the Y and Cb/Cr components, separate Huffman tables are usually used for the Y and Cb/Cr components. Consequently, four Huffman tables are used for JPEG compression. Information relating to these Huffman tables can be defined by each JPEG file, and is stored as a DHT segment in the JPEG file header.

For entropy encoding of a certain value, an absolute value of n bits can only contain n bits of information. In other words, a value whose absolute value is n bits can be expressed using n bits. In signal processing, values are usually defined as follows:

Positive number consisting of n bits: lower n bits of value
Negative number consisting of n bits: lower n bits of value, with the sign inverted

In JPEG compression, entropy encoding follows the above scheme.

Table 1-3. Values of DC/AC Components and Bit Length

Value of component	Category
0	0
–1, 1	1
–3, –2, 2, 3	2
–7 to –4, 4 to 7	3
–15 to –8, 8 to 15	4
–31 to –16, 16 to 31	5
–63 to –32, 32 to 63	6
–127 to –64, 64 to 127	7
–255 to –128, 128 to 255	8
–511 to –256, 256 to 511	9
–1023 to –512, 512 to 1023	10
–2047 to –1024, 1024 to 2047	11

In JPEG compression, entropy compression of the values in this category is performed.
For example, suppose the Huffman table for the DC component for luminance (Y) follows the convention shown below:

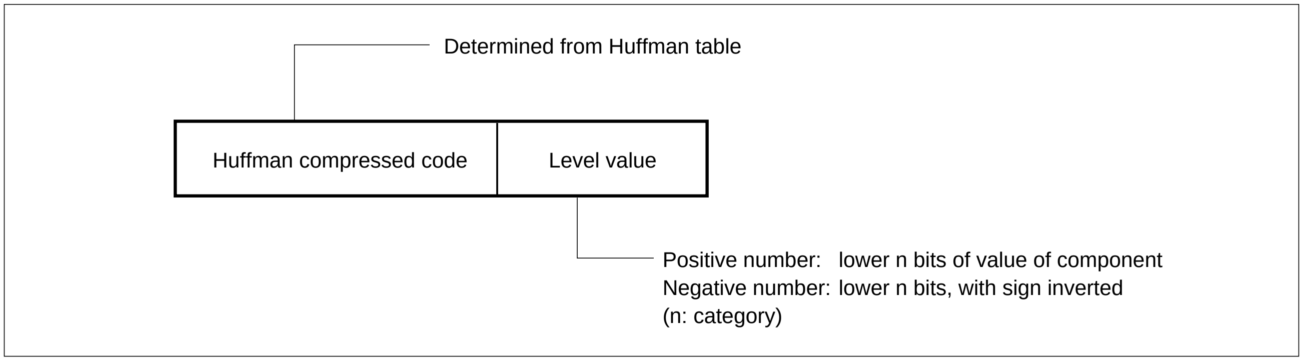
Huffman compressed code 00 (2 bits) is allocated to a value 0 bits long.
Huffman compressed code 010 (3 bits) is allocated to a value 1 bit long.
Huffman compressed code 011 (3 bits) is allocated to a value 2 bits long.
Huffman compressed code 100 (3 bits) is allocated to a value 3 bits long.
Huffman compressed code 001 (3 bits) is allocated to a value 4 bits long.
.
.
.

If the difference in the DC component of the block of the Y component (difference from the DC component of the block of the preceding Y component) is “–3,” “–3” is encoded as follows, because it belongs to category 2.

Huffman compressed code of category 2:	011 (3 bits)
Lower 2 bits of “–3” with sign inverted:	00

3 + 2 = 5 bits

Figure 1-10. Huffman Encoding

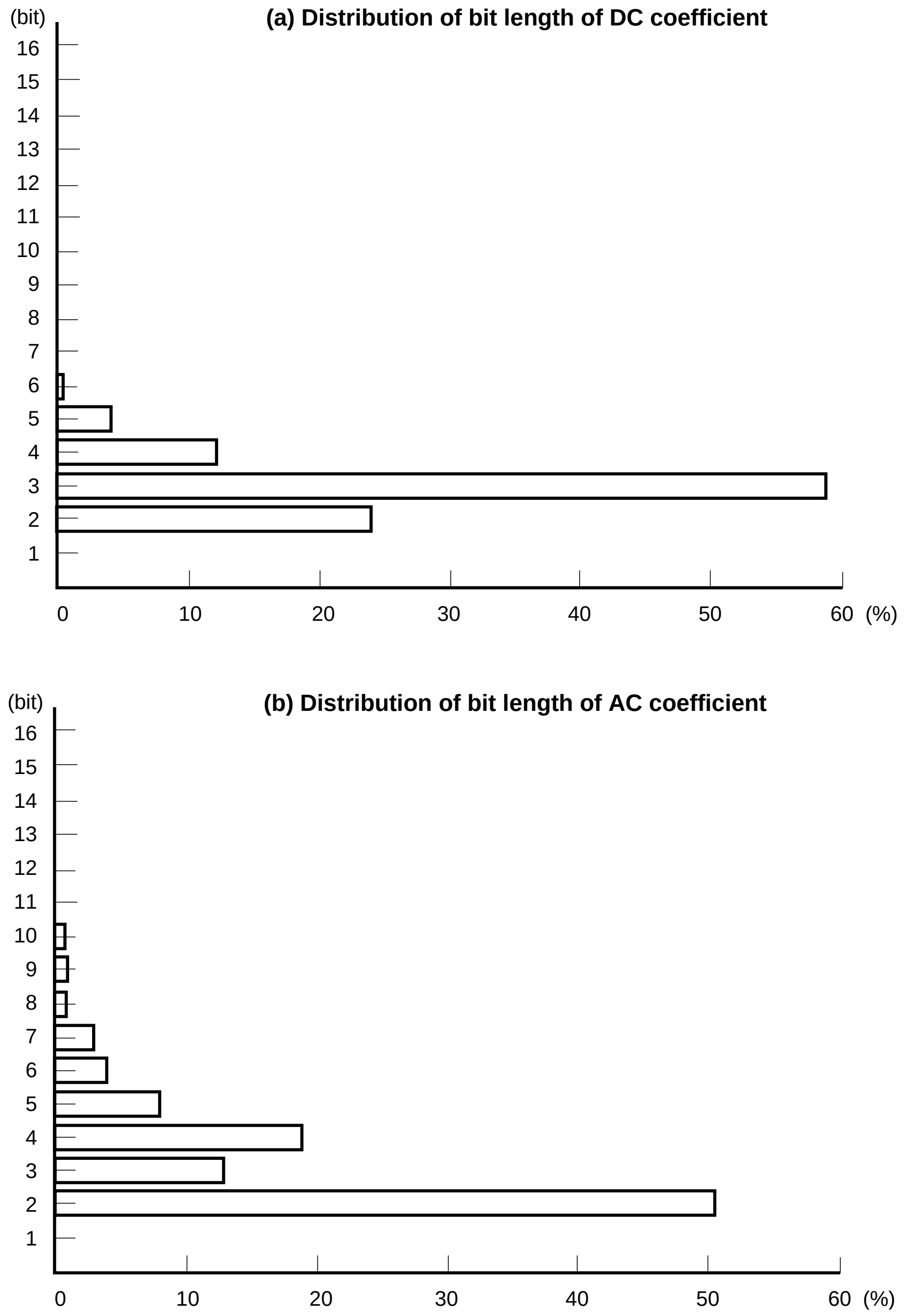

Determined from Huffman table
Huffman compressed code
Level value
Positive number: lower n bits of value of component
Negative number: lower n bits, with sign inverted
(n: category)

For the AC components, the Huffman table follows the convention shown below:

Compressed code 00 (2 bits) is allocated to a 1-bit value with a zero run of 0.
Compressed code 01 (2 bits) is allocated to a 2-bit value with a zero run of 0.
Compressed code 100 (3 bits) is allocated to a 3-bit value with a zero run of 0.
Compressed code 1010 (4 bits) is allocated to a 4-bit value with a zero run of 0.
Compressed code 1011 (4 bits) is allocated to a 1-bit value with a zero run of 1.
Compressed code 1100 (4 bits) is allocated to a 5-bit value with a zero run of 0.
Compressed code 11010 (5 bits) is allocated to a 2-bit value with a zero run of 1.

.
.
.

Figure 1-11. Example of Distribution of Bit Length of DC/AC Coefficients



(7) Restart marker

In JPEG compression, a 2-byte marker (restart marker) is inserted in a code for compressing MCU. The restart marker can be used to expand only the lower part of a JPEG image. If a bit error occurs while a JPEG file is being transferred, and if that file uses restart markers, expansion can be correctly resumed from the next restart marker. With a JPEG file that does not use restart markers, the data cannot be correctly expanded if a bit error occurs.

Figure 1-12. Correct Expansion Cannot Be Performed Because of Bit Error in JPEG File

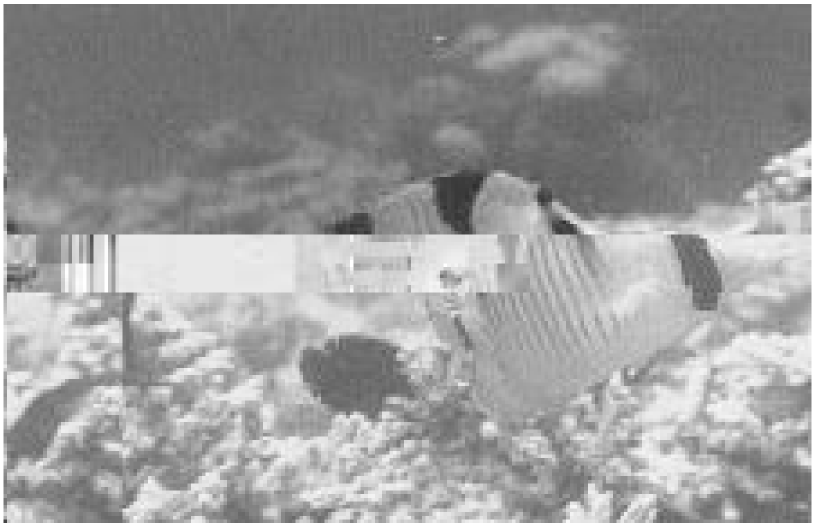
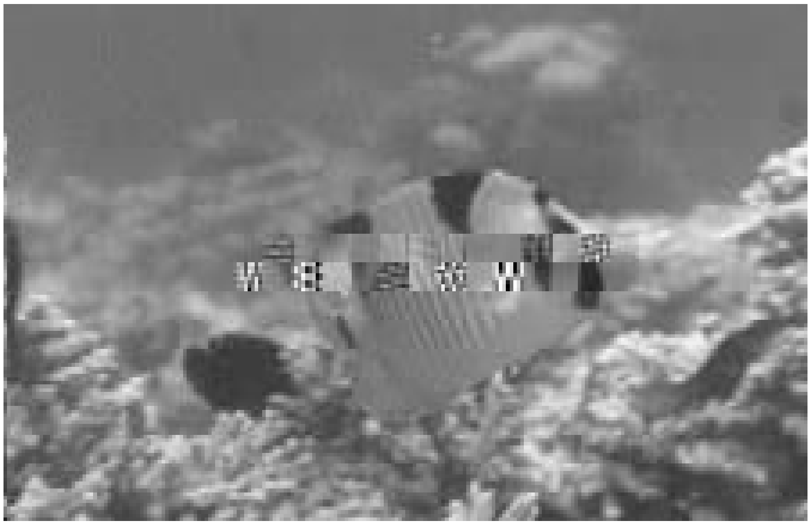
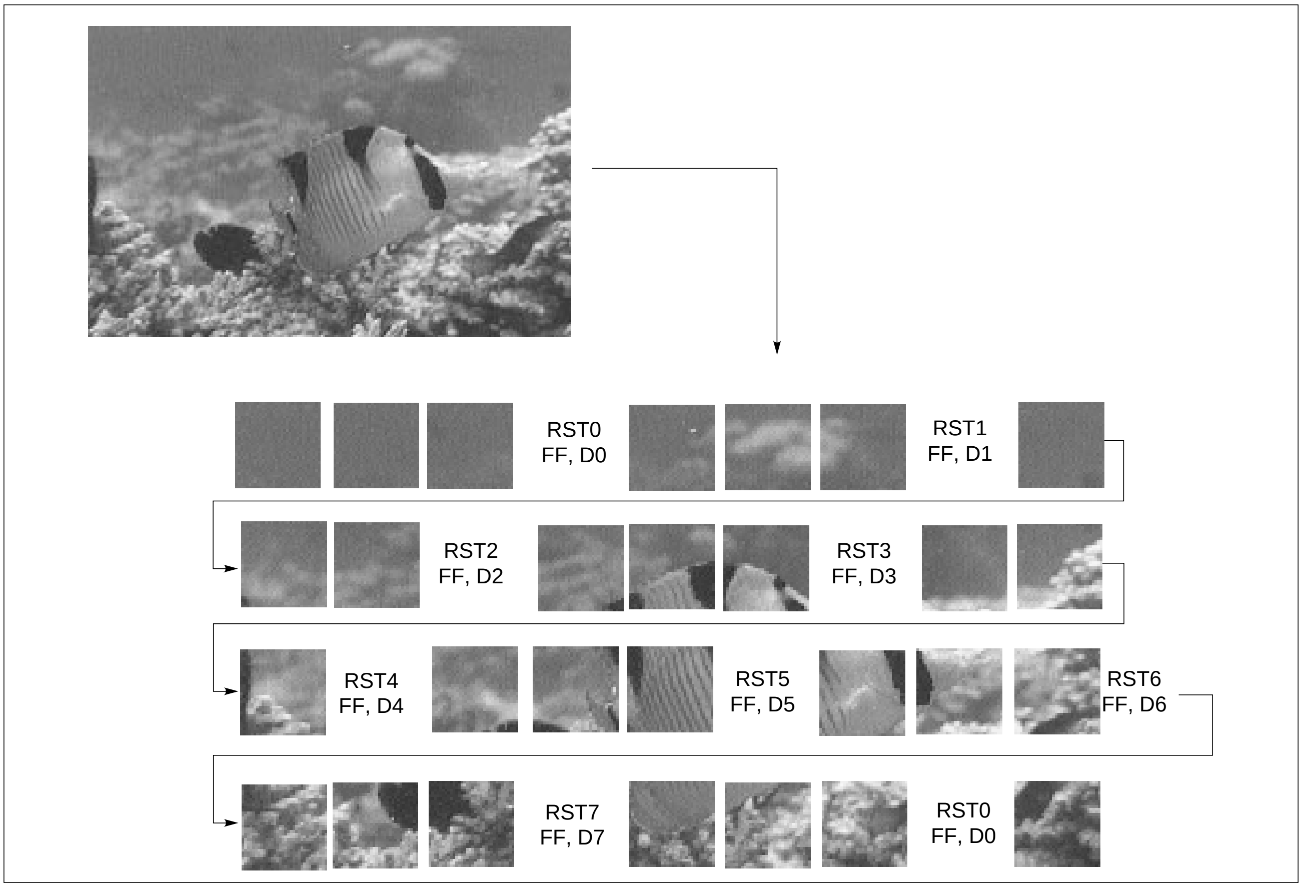


Figure 1-13. Correct Expansion Can Be Performed Due to Use of Restart Markers



There are eight types of restart markers, in the value range of 0xFF,0xD0 to 0xFF,0xD7. A restart marker is inserted in a compressed code every m MCUs, and used in the order of RST0, RST1, and RST2 to RST7. Following RST7, RST0 is used. The value of m is called the restart interval.
If the restart interval is 3, the image will be as shown in the figure below.

Figure 1-14. Restart Marker



The number of restart markers to be inserted is determined by the size of the image. For example, the number of restart markers for an image measuring 640 x 480 pixels, for a sampling ratio of 4:2:2 and a restart interval of 2, is calculated as follows:

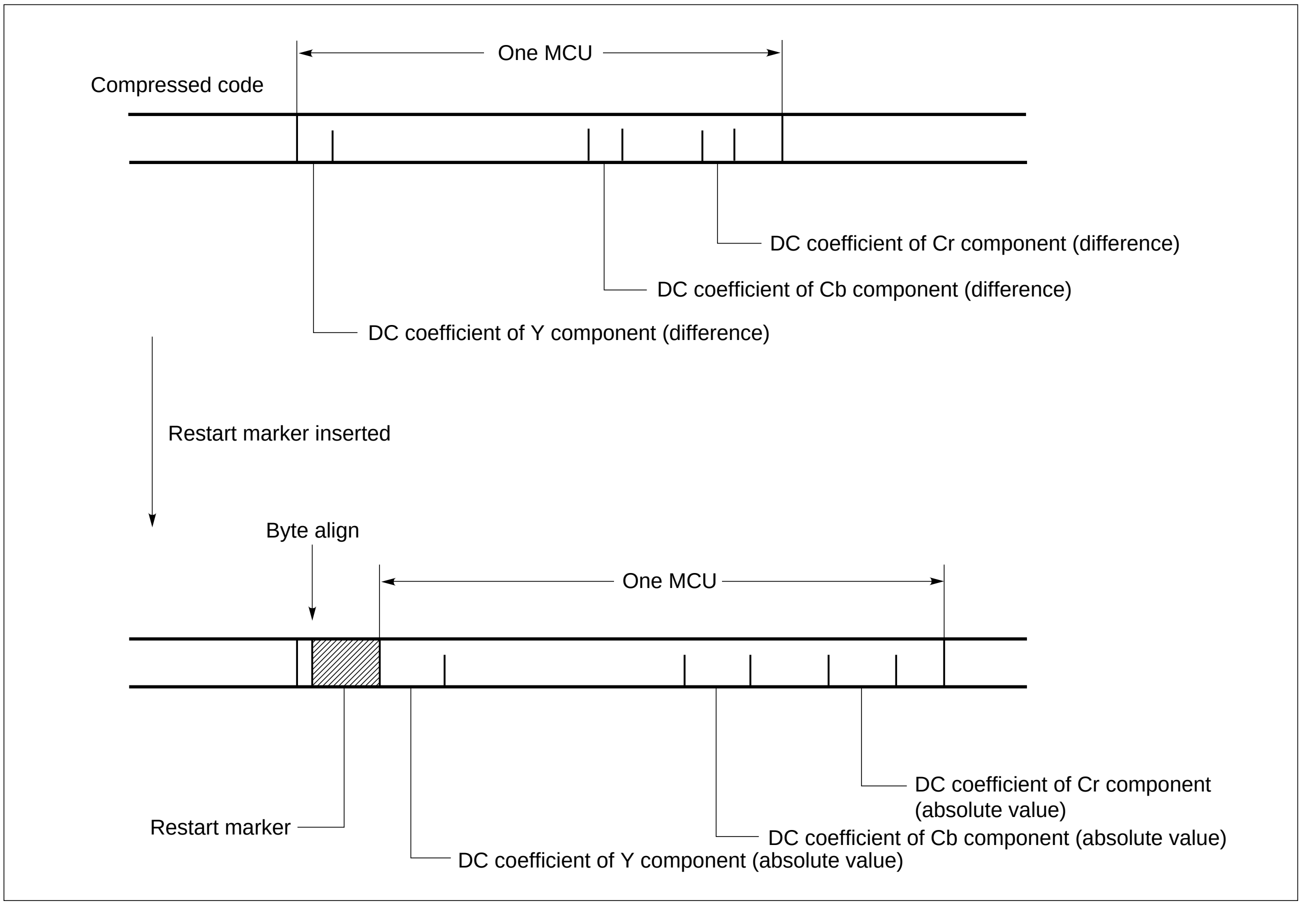
MCU (minimum compression unit):	16 x 8 pixels
Restart marker:	every 2 MCUs

Therefore,

(640 x 480) ÷ (16 x 8) ÷ 2 = 1200 restart markers

A restart marker is located on a byte boundary. On the other hand, compressed code is located in bit units. If one restart marker is inserted, therefore, the data quantity increases to a value equal to the marker, plus 2 bytes. The number of bytes per restart marker is usually less than 4 bytes, although it tends to vary slightly. The DC component immediately after a restart marker is compressed not as the difference from the preceding DC component, but as the value of the DC component itself.
For example, the size of the file for an image measuring 640 x 480 pixels, where the sampling ratio is 4:2:2 and the restart interval is 2, increases by about 4800 bytes (1200 markers x about 4 bytes) relative to when no restart marker is used.

Figure 1-15. Increase in File Size Caused by Use of Restart Marker


One MCU
Compressed code
DC coefficient of Cr component (difference)
DC coefficient of Cb component (difference)
DC coefficient of Y component (difference)
Restart marker inserted
Byte align
One MCU
DC coefficient of Cr component (absolute value)
Restart marker
DC coefficient of Cb component (absolute value)
DC coefficient of Y component (absolute value)

(8) APPn marker

In JPEG compression, an application data segment (APPn segment) can be used so that data not directly related to JPEG compression/expansion can be embedded in or extracted from the header of a JPEG file.

There are 16 types of APPn segments. The contents of these segments can be defined by the user.

Figure 1-16. Structure of APPn Segment

APPn marker (2 bytes)	Data length (2 bytes)	Data (1 to 65533 bytes)

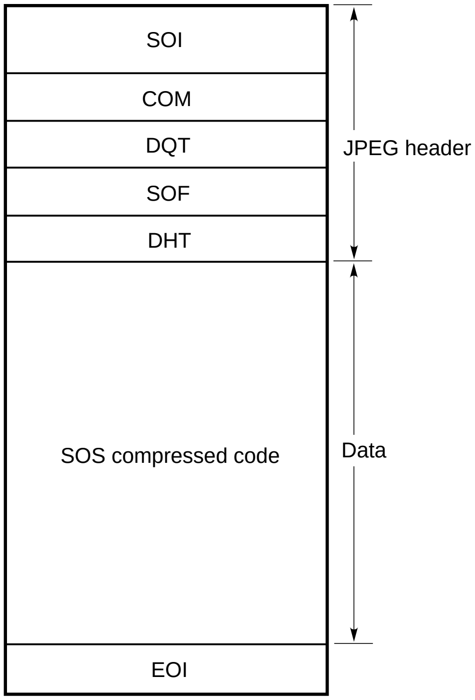
There are 16 types of APPn markers, from 0xFF,0xE0 to 0xFF,0xEF, each corresponding to an APPn segment.

The AP703000-B03 determines whether an APPn segment is to be used during compression. When an APPn segment is to be used, which of the segments is to be used is specified by selecting the corresponding APPn marker. An analysis routine that detects the position of an APPn segment in the JPEG file is also provided.

1.2.2 JPEG File Format

A JPEG file consists of a header that contains several pieces of information necessary for expanding the file, and data obtained by means of DCT, quantization, and entropy compression of an image. All the header data is in byte units (when information is analyzed, however, 1 byte is processed as "4 bits + 4 bits"). Data is in bit units. All data is accommodated on a byte boundary.

Figure 1-17. JPEG File Format



(1) Header

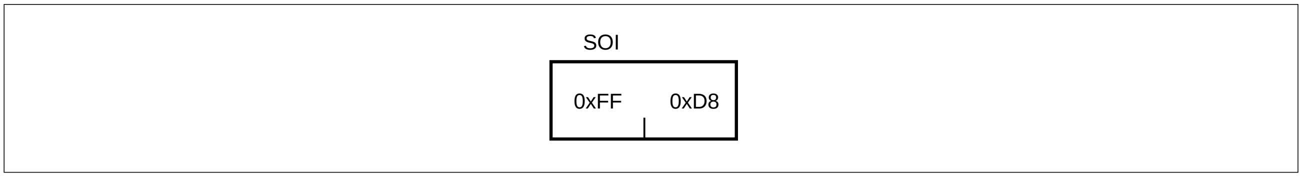
In JPEG compression, tables are managed in units called "segments" that start with a "marker." A marker always consists of 2 bytes, a combination of 0xFF and 1 byte unique to each marker. If a JPEG file is searched for all occurrences of 0xFF, all the markers used in the file can be detected. However, 0xFF is also used in the compressed data, not only in the header. To distinguish between the markers and data, therefore, 0xFF in the compressed data is immediately followed by 0x00, which is meaningless as compressed data. "0xFF,0x00" is not a marker, instead being compressed data "0xFF."

The sequence of each segment (such as COM, DQT, SOF, and DHT) of the JPEG header is arbitrary. The following table lists the JPEG markers.

Table 1-4. JPEG Markers

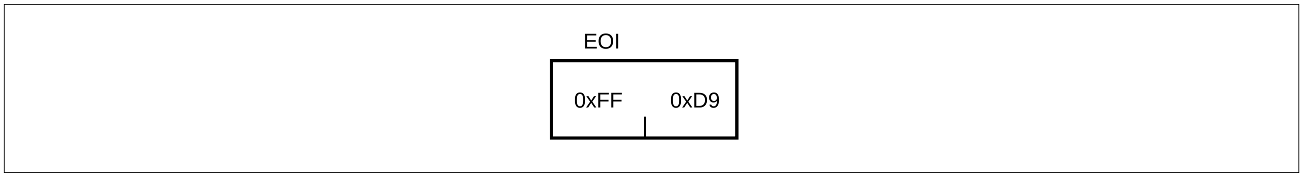
Value	Contents
0xFF 0x00	Non-marker (compressed data 0xFF)
0xFF 0x01	TEM (temporary marker for arithmetic compression)
0xFF 0x02 to 0xFF 0xBF	RES (reserved)
0xFF 0xC0	SOF0 marker (Baseline DCT (Huffman))
0xFF 0xC1	SOF1 marker (Extended sequential DCT (Huffman))
0xFF 0xC2	SOF2 marker (Progressive DCT (Huffman))
0xFF 0xC3	SOF3 marker (Spatial (sequential) lossless (Huffman))
0xFF 0xC4	DHT marker (Huffman table definition segment)
0xFF 0xC5	SOF5 marker (Differential sequential DCT (Huffman))
0xFF 0xC6	SOF6 marker (Differential progressive DCT (Huffman))
0xFF 0xC7	SOF7 marker (Differential spatial (Huffman))
0xFF 0xC8	JPG marker (reserved for JPEG expansion)
0xFF 0xC9	SOF9 marker (Extended sequential DCT (arithmetic))
0xFF 0xCA	SOF10 marker (Progressive DCT (arithmetic))
0xFF 0xCB	SOF11 marker (Spatial (sequential) lossless (arithmetic))
0xFF 0xCC	DAC marker (environment setting segment for arithmetic coding)
0xFF 0xCD	SOF12 marker (Differential sequential DCT (arithmetic))
0xFF 0xCE	SOF13 marker (Differential progressive DCT (arithmetic))
0xFF 0xCF	SOF14 marker (Differential spatial (arithmetic))
0xFF 0xD0 to 0xFF 0xD7	RSTn marker (restart marker)
0xFF 0xD8	SOI marker (header of JPEG file)
0xFF 0xD9	EOI marker (tail of JPEG file)
0xFF 0xDA	SOS marker (header of compressed data)
0xFF 0xDB	DQT marker (quantization table definition)
0xFF 0xDC	DNL marker (number of lines definition)
0xFF 0xDD	DRI marker (definition of restart interval)
0xFF 0xDE	DHP marker (definition of Huffman table)
0xFF 0xDF	EXP marker (expand segment)
0xFF 0xE0 to 0xFF 0xEF	APPn marker (reserved for user application)
0xFF 0xF0 to 0xFF 0xFD	JPGn marker (reserved for JPEG expansion)
0xFF 0xFE	COM marker (comment)

(a) SOI (Start of image) marker



This marker indicates the beginning of a JPEG file. A JPEG file always starts with this 2-byte marker.

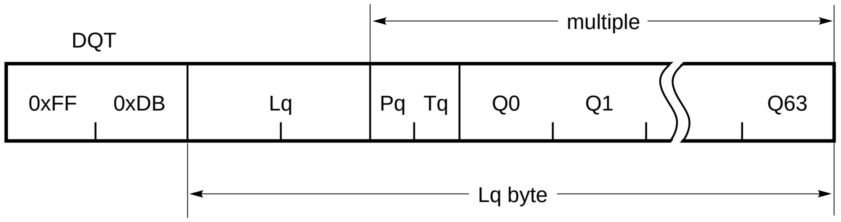
(b) EOI (End of image) marker



This marker indicates the end of a JPEG file. A JPEG file always ends with this 2-byte marker.

(c) DQT (Define quantization table(s)) marker

This marker defines a quantization table.

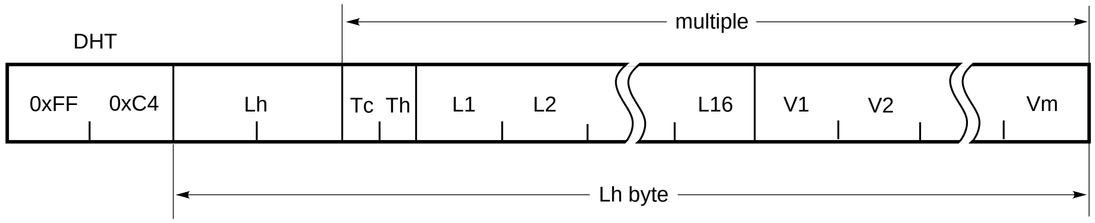


Lq: length of DQT segment
Pq: fixed to 0 with baseline DCT
Tq: quantization table number (0, 1, 2, 3)
Q0 to Q63: quantization factor (1 to 255)

Two DQT markers, one for the normal luminance component (Luminance quantization table) and the other for the chrominance component (Chrominance quantization table), are supported.

(d) DHT (Define Huffman table(s)) marker

This marker defines a Huffman table.



Lh: length of DHT segment
Tc: 0 = DC, 1 = AC
Th: Huffman table number (0, 1)
L1 to L16: indicates how many Huffman codes of i bits in length exist
V1 to Vm: corresponding Huffman code

Example

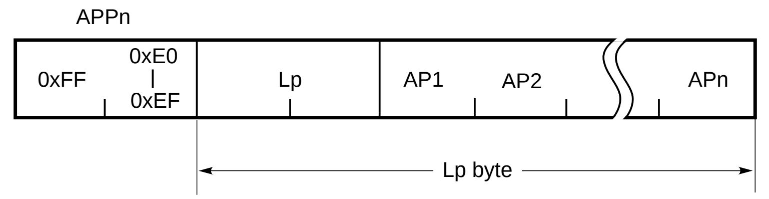
00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00

If L1 through L16 are as shown above, the meaning is as follows:

Zero 1-bit code
One 2-bit code, 00
Five 3-bit codes, 010, 011, 100, 101, and 110
One 4-bit code, 1110
One 5-bit code, 11110
One 6-bit code, 111110
One 7-bit code, 1111110
One 8-bit code, 11111110
One 9-bit code, 111111110
No other codes

V1 through Vm are the corresponding Huffman codes. For example, the Huffman code corresponding to compressed code ‘010’ is V2 (in this case, ‘010’ is the second compressed code).

(e) APPn (Reserved for application segments) marker



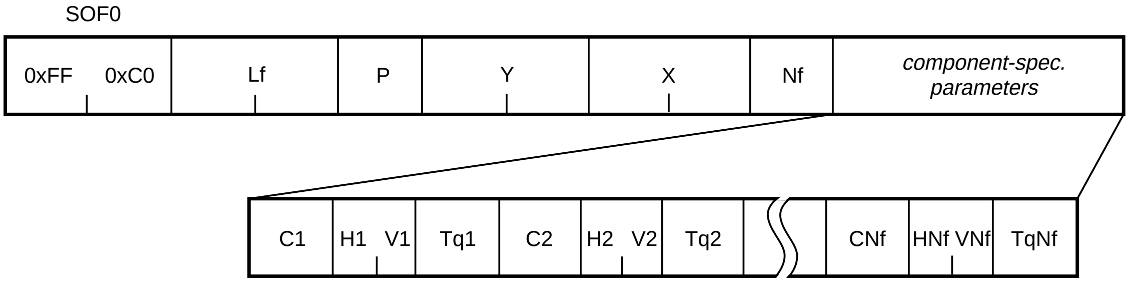
Lp: length of APPn segment
AP1 to APn: data peculiar to application

The application data segment is a segment that can be freely used by each application. Usually, this segment contains the version of the application that created a JPEG file. In some cases, a small JPEG file is contained as is. This segment can be skipped by only referring to the value of Lp.

(f) SOFn (Start of frame) marker

In JPEG compression, that portion of a JPEG file other than the SOI marker and EOI marker is called a frame. The SOFn segment specifies a quantization table number necessary for expansion. The SOFn segment is also called a frame header.

With JPEG, color elements such as Y, Cb, and Cr are called components.



Lf: length of SOF0 segment
P: number of bits constituting component
(8 bits are allocated to each color element for 24-bit full color compression. In this case, P = 8)
Y: vertical size of image (number of pixels)
X: horizontal size of image (number of pixels)
Nf: number of components (Nf = 3 in the case of YCbCr, because the number of colors is 3)
Cn: number of nth component
Hn: horizontal sampling ratio of nth component
Vn: vertical sampling ratio of nth component
Tqn: quantization table number used for nth component

Example

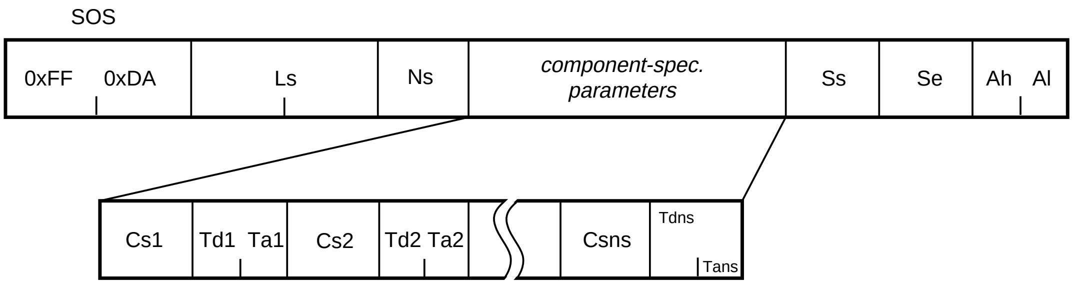
0xFF 0xC0, 0x00, 0x11, 0x08, 0x00, 0x90, 0x00, 0xE0, 0x03,
0x01, 0x22, 0x00, 0x02, 0x11, 0x01, 0x03, 0x11, 0x01

If the contents of this segment are as shown above, the meaning will be as follows:

The sampling ratio of the first component (Y) is 2 x 2 and the quantization table number is 0.
The sampling ratio of the second component (Cb) is 1 x 1 and the quantization table number is 1.
The sampling ratio of the third component (Cr) is 1 x 1 and the quantization table number is 1.

(g) SOS (Start of scan) marker

The SOS segment is also called a scan header. The data constituting a compressed image starts immediately after the scan header. The scan header specifies the Huffman table number of the compressed data.



Ls: length of SOS segment
Ns: specifies number of components
Csn: ID of nth component
Tdn: Huffman table number of DC coefficient of nth component
Tan: Huffman table number of AC coefficient of nth component
Ss: fixed to 0 with baseline DCT
Se: fixed to 63 with baseline DCT
Ah: fixed to 0 with baseline DCT
Al: fixed to 0 with baseline DCT

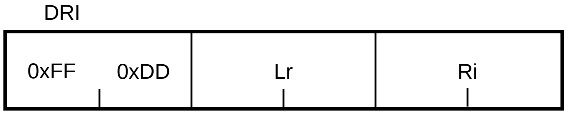
(h) DRI (Define restart interval) marker and RSTn (Restart interval termination) marker

A restart marker is used to minimize the influence of illegal data such as a communication error. A restart marker is inserted every number of MCUs, as set by the DRI marker.

For example, to insert a marker every four MCUs, restart markers are inserted sequentially, starting from RST0 and RST1 to RST7, as follows:

[MCU1][MCU2][MCU3][MCU4]RST0[MCU5][MCU6][MCU7][MCU8]RST1 ...

Because the DC coefficient is differential information with JPEG, the preceding DC coefficient is required to expand an 8 x 8 pixel block. The DC coefficient immediately after a restart marker is a differential from 0. Consequently, even if data is destroyed in MCU4, MCU5 and subsequent MCUs can be correctly expanded.



Lr: length of DRI segment (4)
Ri: restart interval

(2) Data

Although the header is stored in a file in units of bytes, data is stored in a file in units of bits (the overall data is stored in multiples of bytes).

The first point to be noted concerning data is that data 0xFF is immediately followed by 0x00 to distinguish it from a marker. For expansion, therefore, all occurrences of 0x00 following 0xFF must be ignored. To do so, data must be read from memory 1 byte at a time, and a judgment made each time to determine whether each item of byte data is 0xFF. This means that a technique to increase the speed, such as reading data from memory in blocks of 2 or 4 bytes, cannot be used.

(3) Huffman decoding

Example

```
FF C4 00 1F 00
00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B
```

The Huffman table segment shown above is interpreted as follows:

There is no 1-bit Huffman code.
There is one 2-bit Huffman code, ‘00,’ for which the corresponding value is ‘0x00.’
There are five 3-bit Huffman codes: ‘010,’ ‘011,’ ‘100,’ ‘101,’ and ‘110.’ The corresponding values are ‘0x01,’ ‘0x02,’ ‘0x03,’ ‘0x04,’ and ‘0x05.’
There is one 4-bit Huffman code, ‘1110,’ for which the corresponding value is ‘0x06.’
....

If this interpretation were to be made each time compressed image data is expanded, processing would take an excessively long time. It is also wasteful to repeat the same processing over and over again. Actually, therefore, a simple table is created by analyzing the Huffman table segment before starting the expansion of compressed image data. This table is referenced when the image data is expanded.

(a) Huffman table initialization

The JPEG standard recommends the creation of the following table:

Table 1-5. Recommended Huffman Table

Number of bits	Minimum value of code	Maximum value of code	Number of minimum code
1 bit	Undefined	–1	Undefined
2 bits	0x0	0x0	0x0
3 bits	0x2	0x6	0x1
4 bits	0xE	0xE	0x6
5 bits	0x1E	0x1E	0x7
6 bits	0x3E	0x3E	0x8
7 bits	0x7E	0x7E	0x9
8 bits	0xFE	0xFE	0xA
9 bits	0x1FE	0x1FE	0xB
10 bits	Undefined	–1	Undefined
11 bits	Undefined	–1	Undefined
12 bits	Undefined	–1	Undefined
13 bits	Undefined	–1	Undefined
14 bits	Undefined	–1	Undefined
15 bits	Undefined	–1	Undefined
16 bits	Undefined	–1	Undefined

In the above Huffman code segment, the minimum value of the 3-bit Huffman code is ‘010’ (= 0x02) while the maximum value is ‘110’ (= 0x06). The minimum code ‘010’ is the second (0x1) of those that follow ‘00.’

(b) Decoding of DC coefficient

As an example, suppose that the Huffman table segment shown in Table 1-5 is given.
Also suppose that the compressed code has the following value:

0xDA 0x49 (= 1101 1010 0100 1001)

First, the first bit ‘1’ (= 0x1) of the compressed data is compared with the maximum 1-bit value of the Huffman code table.

0x1 > –1 (maximum 1-bit value of table)

It can be determined, therefore, that the compressed code is not a 1-bit code.
Next, the first 2 bits ‘11’ (= 0x3) are compared with the maximum 2-bit value of the table.

0x3 > 0x0 (maximum 2-bit value of table)

Therefore, the compressed data is not a 2-bit code, either.
Then, the first 3 bits ‘110’ (= 0x6) are compared with the maximum 3-bit value of the table.

0x6 = 0x6 (maximum 3-bit value of table)

This determines that the compressed code is 3 bits long.
By using this 3-bit compressed code, the position of ‘110’ as a Huffman code must be obtained. To do this, the minimum 3-bit value ‘0x2’ of the Huffman table is subtracted from compressed code ‘0x6’.

0x6 – 0x2 = 0x4

This means that the 3-bit compressed code is at the 0x4th position. Because the minimum 3-bit code ‘0x2’ is at the 0x1th position (if the table is referenced), the following calculation is performed:

0x4 + 0x1 = 0x5

This indicates that the compressed code is at the 0x5th code position.
Next, the Huffman code segment is referenced.

00 01 02 03 04 05 06 07 08 09 0A 0B

The value of the 0x5th position is ‘05.’ This means that the ‘5’ bits following the compressed code are a level value.
Next, the decoded 3 bits, ‘110,’ are stripped from the compressed code and discarded.

```
0xDA 0x49 (= 1101 1010 0100 1001)
                ↓
0xD2 0x48 (=    1 1010 0100 1001)
```

The first 5 bits are ‘11010’ (= 0x1A). Because the first sign bit in this case is ‘1’ (indicating a positive number), the obtained value is ‘0x1A.’ This completes the decoding of the DC coefficient.

(c) Decoding of AC coefficient

Basically, the AC coefficient is decoded in the same manner as the DC coefficient, but with the following differences:

- The Huffman code table is different.
- The higher 4 bits of the value taken from the Huffman code segment indicate the length of the zero run, while the lower 4 bits indicate the number of bits of the level value.

(4) Reverse quantization

Reverse quantization involves multiplying the 8 x 8 pixel matrix obtained as a result of Huffman decoding, by a quantized matrix defined by the quantization segment.

Because this processing only involves calculating the product of the two values read from memory, followed by writing the result to another memory address, processing that restores the original sequence of a zigzag scan is also performed at the same time, so that the number of times memory is accessed can be reduced and the overall processing speed improved.

Almost all the elements of the matrix obtained as a result of Huffman decoding have a value of ‘0’ (zero). The normal CPU requires several clocks to execute the multiplication or memory access (read or write). The AP703000-B03 inserts one expression to identify whether the value of an element is ‘0.’ If the value is ‘0,’ that element is rejected.

(5) Actuality of DCT/reverse DCT

DCT is equivalent to repeating the following one-dimensional calculation twice (vertically and horizontally).

$$\begin{aligned} F(u,v) &= 2C(u)\mathrm{C}(v)/\mathrm{N}{*}\Sigma\Sigma f(i, j){*}\cos\{(2i+1)u\pi/2\mathrm{N}\}\ {*}\cos\{(2j+1)v\pi/2\mathrm{N}\} \\ &= 2C(u)C(v)/\mathrm{N}{*}\Sigma\cos\{(2i+1)u\pi/2\mathrm{N}\}\ {*}\Sigma f(i, j){*}\cos\{(2j+1)v\pi/2\mathrm{N}\} \end{aligned}$$

The number of constants required for two dimensions and eight orders (N = 8; i.e., 8 x 8 matrix) is 32. Actually, however, this can be decreased by applying the formulae of a trigonometric function to the subsequent seven.

$$\cos(\mathrm{n}\pi/16)\ (\mathrm{n} = 0, 1, 2, \ldots 31)$$
$$\downarrow$$
$$\cos(\mathrm{n}\pi/16)\ (\mathrm{n} = 1, 2, 3, 4, 5, 6, 7)$$

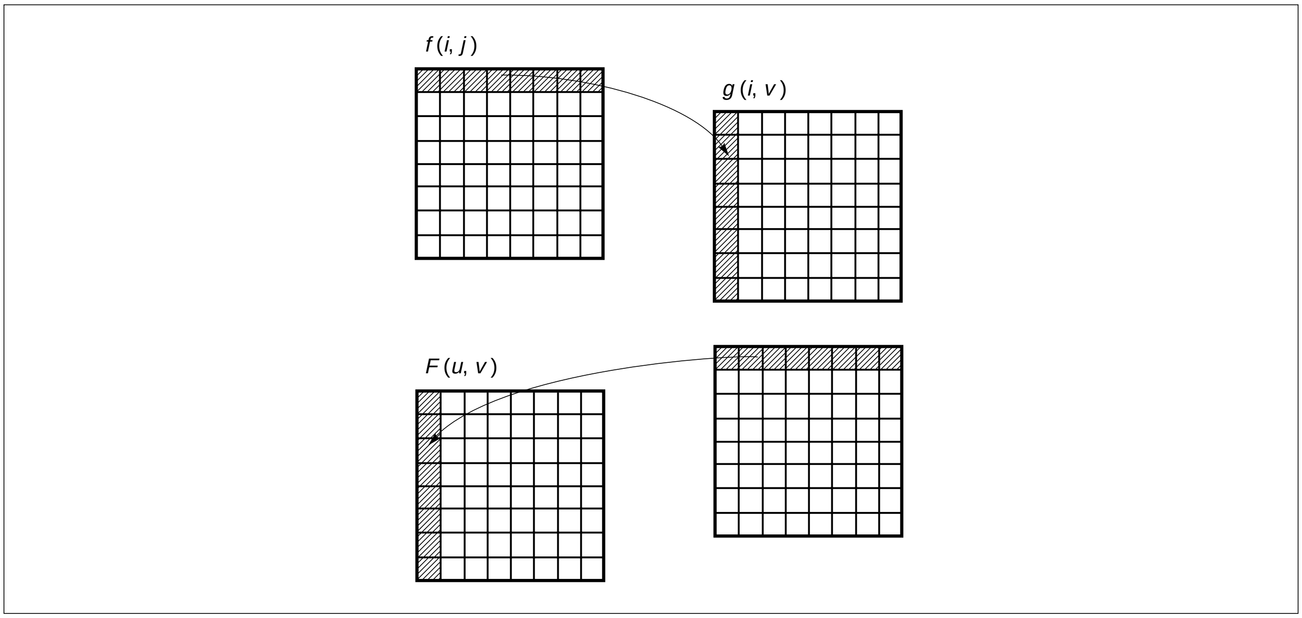
If the following value has been calculated in advance, the number of times multiplication is executed can be reduced.

$$f(0, j) \pm f(7, j),\ f(1, j) \pm f(6, j),\ \ldots$$

Further, the vertical and horizontal routines can be used commonly if the DCT expression is changed as follows:

$$\begin{aligned} g(i, v) &= \Sigma f(i, j){*}\cos\{(2j+1)v\pi/16\} \\ f(u, v) &= \Sigma g(i, v){*}\cos\{(2i+1)u\pi/16\} \end{aligned}$$

Figure 1-18. DCT/Reverse DCT Algorithm



Actually, the pointer of *f* (*i*, *j*) and the pointer of work area *g* (*i*, *v*) are passed as arguments to the DCT routine. By allocating this work area to high-speed RAM, the processing speed can be increased.

1.3 OUTLINE OF SYSTEM

1.3.1 Major Functions

(1) Sampling ratio

The following four sampling ratios are supported.

- 4:1:1 [H:V = 2:2] (The screen size is a multiple of 16 both vertically and horizontally.)
- 4:1:1 [H:V = 4:1] (The screen size is a multiple of 32 horizontally, and of 8 vertically.)
- 4:2:2 [H:V = 2:1] (The screen size is a multiple of 16 horizontally, and of 8 vertically.)
- 4:4:4 [H:V = 1:1] (The screen size is a multiple of 8 both vertically and horizontally.)

(2) Coordinates (x, y)

Assuming VRAM specification for both YCbCr and RGB, an image can be expanded at any point in VRAM and can be compressed at any point in VRAM.

(3) Quantization table

Up to two quantization tables can be set.

A default quantization table is provided for compression, but a user-defined quantization table can also be used.

The value written to the DQT header is used for expansion.

(4) Huffman table

Up to four Huffman tables can be set.

A default Huffman table is provided for compression, but a user-defined Huffman table can also be used.

The value written to the DHT header is used for expansion.

(5) Restart marker

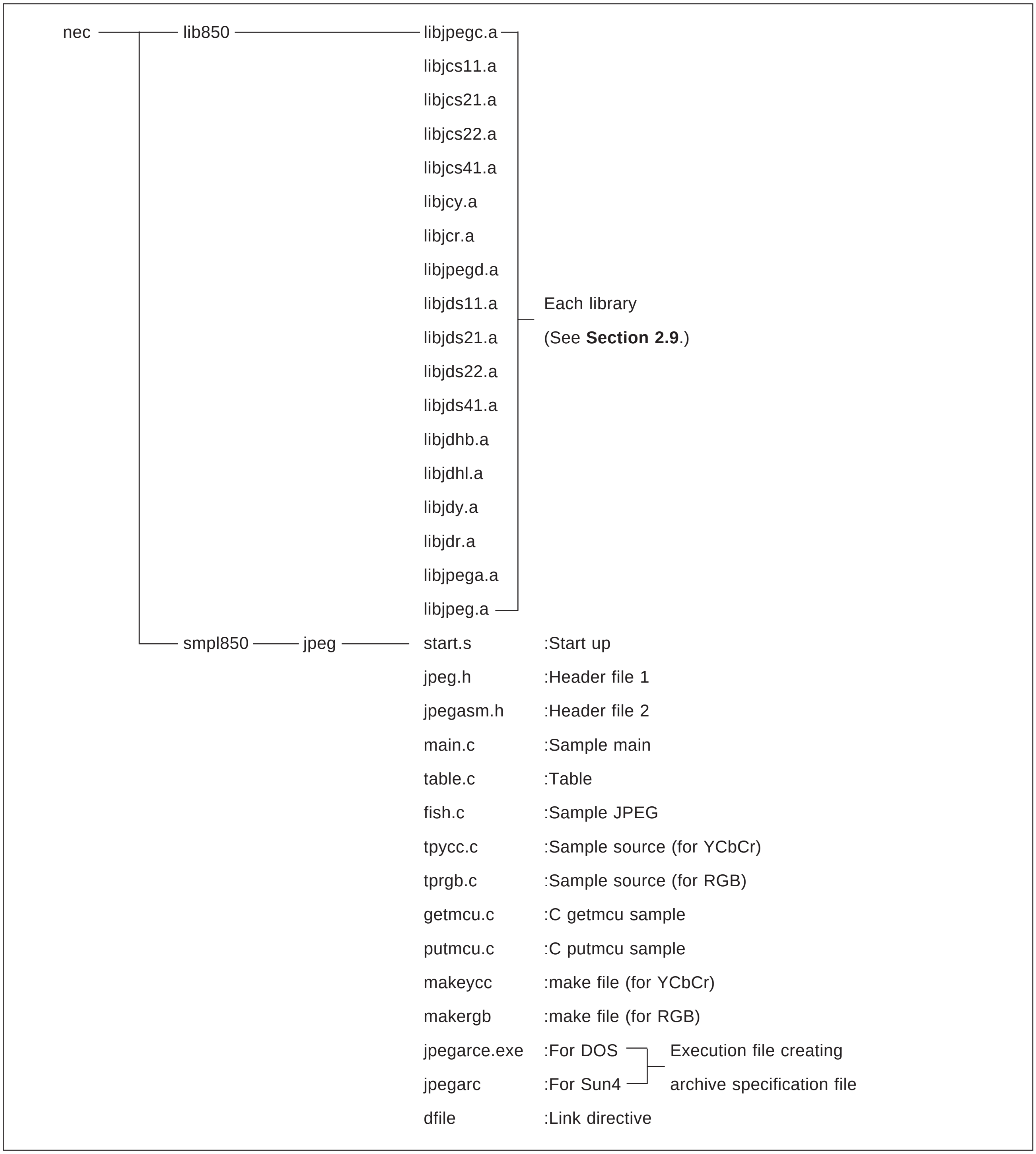
Whether restart markers are to be used can be specified for compression. If they are used, the restart interval can be changed.

The value of the DRI header is used for expansion.

1.3.2 Package Contents

The package includes the following libraries and sample source.

(1) NEC version



```
nec ──┬── lib850 ───────────── libjpegc.a ─┐
      │                        libjcs11.a  │
      │                        libjcs21.a  │
      │                        libjcs22.a  │
      │                        libjcs41.a  │
      │                        libjcy.a    │
      │                        libjcr.a    │
      │                        libjpegd.a  │
      │                        libjds11.a  ├─ Each library
      │                        libjds21.a  │  (See Section 2.9.)
      │                        libjds22.a  │
      │                        libjds41.a  │
      │                        libjdhb.a   │
      │                        libjdhl.a   │
      │                        libjdy.a    │
      │                        libjdr.a    │
      │                        libjpega.a  │
      │                        libjpeg.a ──┘
      └── smpl850 ── jpeg ──── start.s        :Start up
                               jpeg.h         :Header file 1
                               jpegasm.h      :Header file 2
                               main.c         :Sample main
                               table.c        :Table
                               fish.c         :Sample JPEG
                               tpycc.c        :Sample source (for YCbCr)
                               tprgb.c        :Sample source (for RGB)
                               getmcu.c       :C getmcu sample
                               putmcu.c       :C putmcu sample
                               makeycc        :make file (for YCbCr)
                               makergb        :make file (for RGB)
                               jpegarce.exe   :For DOS ──┬─ Execution file creating
                               jpegarc        :For Sun4 ─┘  archive specification file
                               dfile          :Link directive
```

(2) GHS version

```
ghs ──┬── lib850 ──────── libjpegc.a ─┐
      │                   libjcs11.a  │
      │                   libjcs21.a  │
      │                   libjcs22.a  │
      │                   libjcs41.a  │
      │                   libjcy.a    │
      │                   libjcr.a    │
      │                   libjpegd.a  │
      │                   libjds11.a  │   Each library
      │                   libjds21.a  ├── (See Section 2.9.)
      │                   libjds22.a  │
      │                   libjds41.a  │
      │                   libjdhb.a   │
      │                   libjdhl.a   │
      │                   libjdy.a    │
      │                   libjdr.a    │
      │                   libjpega.a  │
      │                   libjpeg.a  ─┘
      └── smpl850 ── jpeg ── start.s        :Start up
                             jpeg.h         :Header file 1
                             jpegasm.h      :Header file 2
                             main.c         :Sample main
                             table.c        :Table
                             fish.c         :Sample JPEG
                             tpycc.c        :Sample source (for YCbCr)
                             tprgb.c        :Sample source (for RGB)
                             gemcu.c        :C getmcu sample
                             putmcu.c       :C putmcu sample
                             makeycc        :make file (for YCbCr)
                             makergb        :make file (for RGB)
                             jpegarce.exe   :For DOS  ──┐   Execution file creating
                             jpegarc        :For Sun4 ──┴── archive specification file
                             make.lnk       :Section specification
```

Each library can be re-linked. For details of how to perform link, refer to the manual supplied with the linker.

1.3.3 Operating Environment

(1) Applicable CPU

V850 family

(2) Compiler package used

GHS (Green Hills Software, Inc.) compiler

C cross compiler V850 Ver.1.8.7B or later

NEC compiler package

CA850 Ver.1.00 or later

(3) Memory capacity

The amount of ROM/RAM required for each library is as follows. However, the total differs depending on which sampling ratio is selected.

When using RGB, more memory is required.

ROM size (units: bytes)

Processing \ Sampling ratio		BASE	4:4:4 [H:V = 1:1]	4:2:2 [H:V = 2:1]	4:1:1 [H:V = 2:2]	4:1:1 [H:V = 4:1]
Compression	YCbCr	5.5K	0.7K	0.7K	1.0K	1.0K
	RGB		0.8K	1.0K	1.5K	1.5K
Expansion	YCbCr	4.5K	0.5K	0.5K	0.7K	0.7K
	RGB		0.7K	0.7K	1.1K	1.1K
Analysis		1.5K				

RAM size (units: bytes)

Processing	Sampling ratio	BASE	4:4:4 [H:V = 2:2]	4:2:2 [H:V = 2:2]	4:1:1 [H:V = 2:2]	4:1:1 [H:V = 4:1]
Compression	JPEGINFO	128				
	Work1	256				
	Work2	3072				
	MCU		384	512	768	768
	Stack	128				
	Subtotal	3584	384	512	768	768
Expansion	JPEGINFO	128				
	Work1	256				
	Work2	8192				
	MCU		384	512	768	768
	Stack	128				
	Subtotal	8704	384	512	768	768
Analysis	JPEGINFO	128				
	Work1	256				
	Stack	128				
	Subtotal	512				

Example The required memory size is as shown in the table below, provided conditions are as follows:

YCbCr/RGB:	YCbCr
Selected processing:	Compression/expansion
Sampling ratio:	Only “4:1:1 [H:V = 2:2]” for both compression/expansion

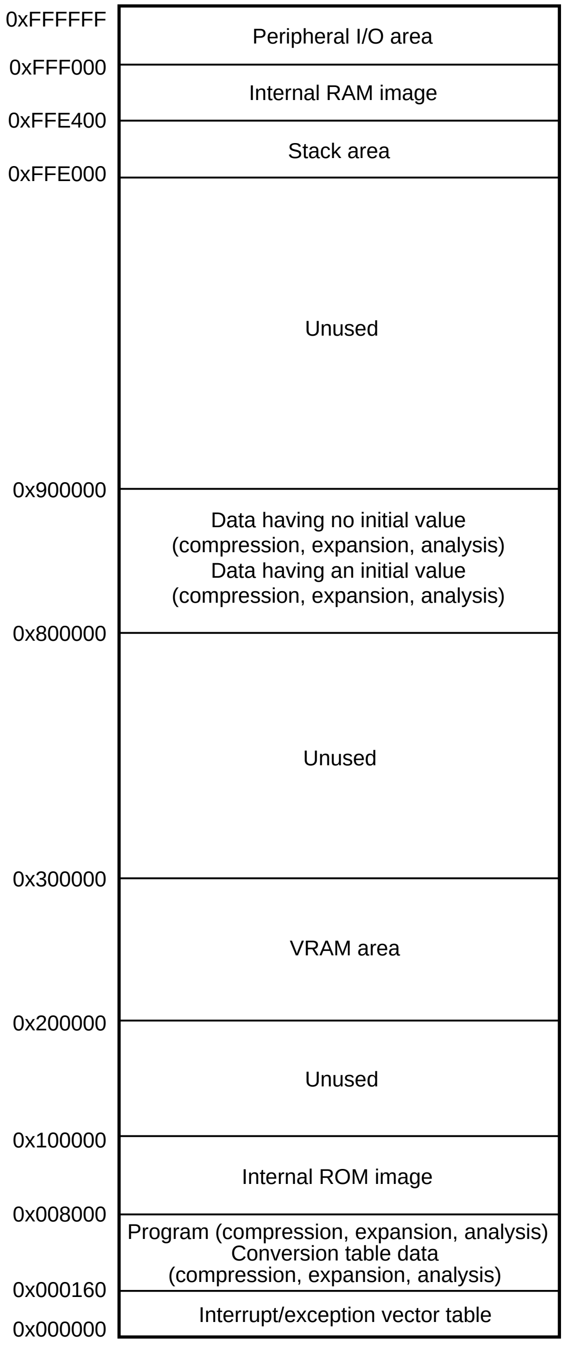
	ROM size (bytes)	RAM size (bytes)
Compression	Compression BASE + compression 4:1:1 = 5.5K + 1.0K = 6.5K	Compression BASE + compression MCU 4:1:1 = 3584 + 768 = Approx. 5K
Expansion	Expansion BASE + expansion 4:1:1 = 4.5K + 0.7K = 5.2K	Expansion BASE + expansion MCU 4:1:1 = 8704 + 768 = Approx. 10K
Total	Approx. 11.7K	Approx. 15K

1.3.4 Sample Program

The package includes the sources of sample programs for compression, expansion, and analysis. Use these sources as system examples. For details of the sources, see **APPENDIX A**.

Figure 1-19 shows an example of mapping the sample programs.

Figure 1-19. Memory Map for Sample Program (with V851)


0xFFFFFF
Peripheral I/O area
0xFFF000
Internal RAM image
0xFFE400
Stack area
0xFFE000
Unused
0x900000
Data having no initial value
(compression, expansion, analysis)
Data having an initial value
(compression, expansion, analysis)
0x800000
Unused
0x300000
VRAM area
0x200000
Unused
0x100000
Internal ROM image
0x008000
Program (compression, expansion, analysis)
Conversion table data
(compression, expansion, analysis)
0x000160
Interrupt/exception vector table
0x000000

[MEMO]

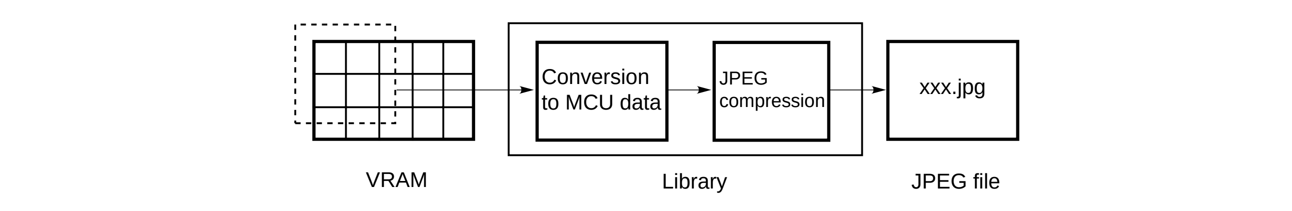
CHAPTER 2 LIBRARY SPECIFICATIONS

2.1 FUNCTION

The library group provided with the AP703000-B03 enables the following three types of processing to be performed:

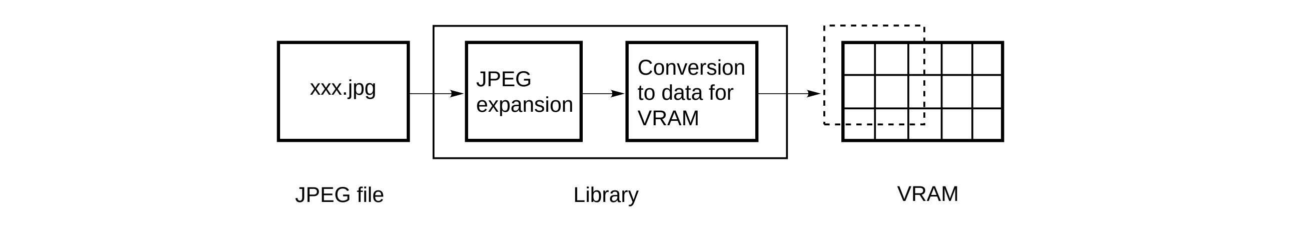
(1) Compression

Image data is compressed and a JPEG file is created.



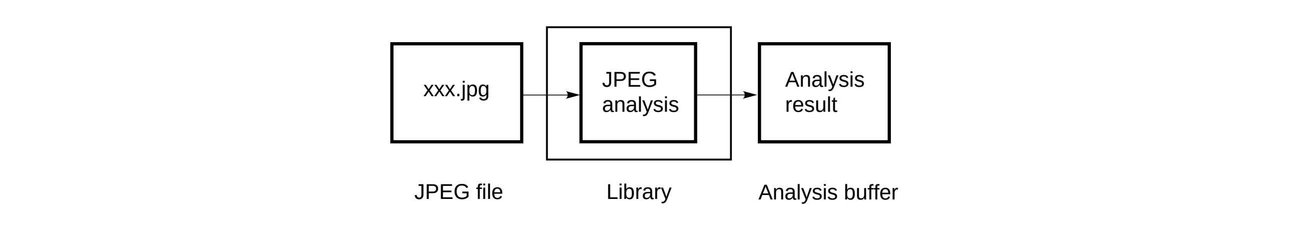
(2) Expansion

The JPEG file is expanded and displayed on the screen.



(3) Analysis

Information, such as screen size, is obtained from the JPEG file.



The following two types of processing information can be set by the libraries:

<1> Processing information that can be set by parameters

The user can set parameters in an application for the following six items:

- Huffman table
- Quantization table
- Sampling ratio (1)
- Restart marker
- Screen size (horizontal, vertical)
- Coordinates (x, y)

<2> Processing information that can be selected by the libraries

The user can select parameters for the following three items from the libraries.

- Sampling ratio (2)
- VRAM (YCbCr/RGB)
- Huffman expansion procedure

2.1.1 Processing Information for Which Parameters Can Be Set

The items for which the user can set parameters in an application are as follows. For details, see **Section 2.5**.

(1) Huffman table

Set four Huffman tables, for the DC luminance component, AC luminance component, DC chrominance component, and AC chrominance component.

Compression Although a default table is provided, a user-defined table can also be set.

Expansion The value in the DHT header is used.

(2) Quantization table

Set two quantization tables for the luminance and chrominance components.

Compression Although a default table is provided, a user-defined table can also be set.
A parameter called Quality is also provided in a library. A defined table processed with Quality specified is handled as a quantization table.

Expansion The value in the DQT header is used.

(3) Sampling ratio (1)

Compression Four sampling ratios are supported: 4:4:4 [H:V = 1:1], 4:2:2 [H:V = 2:1], 4:1:1 [H:V = 2:2], and 4:1:1 [H:V = 4:1].

This information is appended to the JPEG header.

Expansion Four sampling ratios are supported: 4:4:4 [H:V = 1:1], 4:2:2 [H:V = 2:1], 4:1:1 [H:V = 2:2], and 4:1:1 [H:V = 4:1].

The sampling ratio used is automatically identified and, therefore, can be ignored by the user.

Table 2-1. Sampling Ratios

Sampling ratio	4:1:1	4:2:2	4:4:4
Color	Normal	Fairly clear	Clear
File size	Reference value (x1)	About x4/3	About x2

(4) Restart marker

Compression Whether restart markers are used can be selected.

To use restart markers, any value can be specified.

Expansion The value of the DRI header is used.

(5) Screen size (horizontal, vertical)

Compression and expansion The sizes that can be set are as follows:

Table 2-2. Screen Sizes

Sampling ratio	Holizontal	Vertical
4:4:4 [H:V = 1:1]	Multiple of 8	Multiple of 8
4:2:2 [H:V = 2:1]	Multiple of 16	Multiple of 8
4:1:1 [H:V = 2:2]	Multiple of 16	Multiple of 16
4:1:1 [H:V = 4:1]	Multiple of 32	Multiple of 8

(6) Coordinates (x, y)

Compression and expansion Assuming that a VRAM specification for both YCbCr/RGB, image data can be expanded at any point in VRAM, and also compressed at any point in VRAM.

2.1.2 Processing Information That Can Be Selected from a Library

The items that the user can select from the libraries are as follows. For details of making the selection, see **Section 2.9**.

(1) Sampling ratio (2)

Although the sampling ratios described in Sampling ratio (1) are supported by default, libraries are provided to enable each sampling ratio to be set in detail. For example, if it is known that a sampling ratio of 4:4:4 is not necessary, selection of that library can be omitted. This function enables a slight reduction in the code size.

(2) VRAM (YCbCr/RGB)

The VRAM input/output portion (VRAM-MCU (Minimum Coded Unit) data transfer block between buffers) can be selected from the following:

Default	VRAM specifications assumed by the libraries for both YCbCr/RGB.
User-created	VRAM specifications other than the above.

For details, see **Sections 2.4 and 2.10**.

(3) Huffman expansion

Methods which perform search starting from the point having the highest probability of statistical branch, as well as those which use a loop, are provided.

2.2 DETAILS OF PROCESSING

2.2.1 Compression

Image data is compressed and a JPEG file is created.

(1) Function

Function name	Description
jpeg_CompressInit	JPEG compression library initialization
jpeg_Compress	JPEG compression

(2) Status transition

The transition of the compression status is shown below.

Figure 2-1. Compression Status Transition

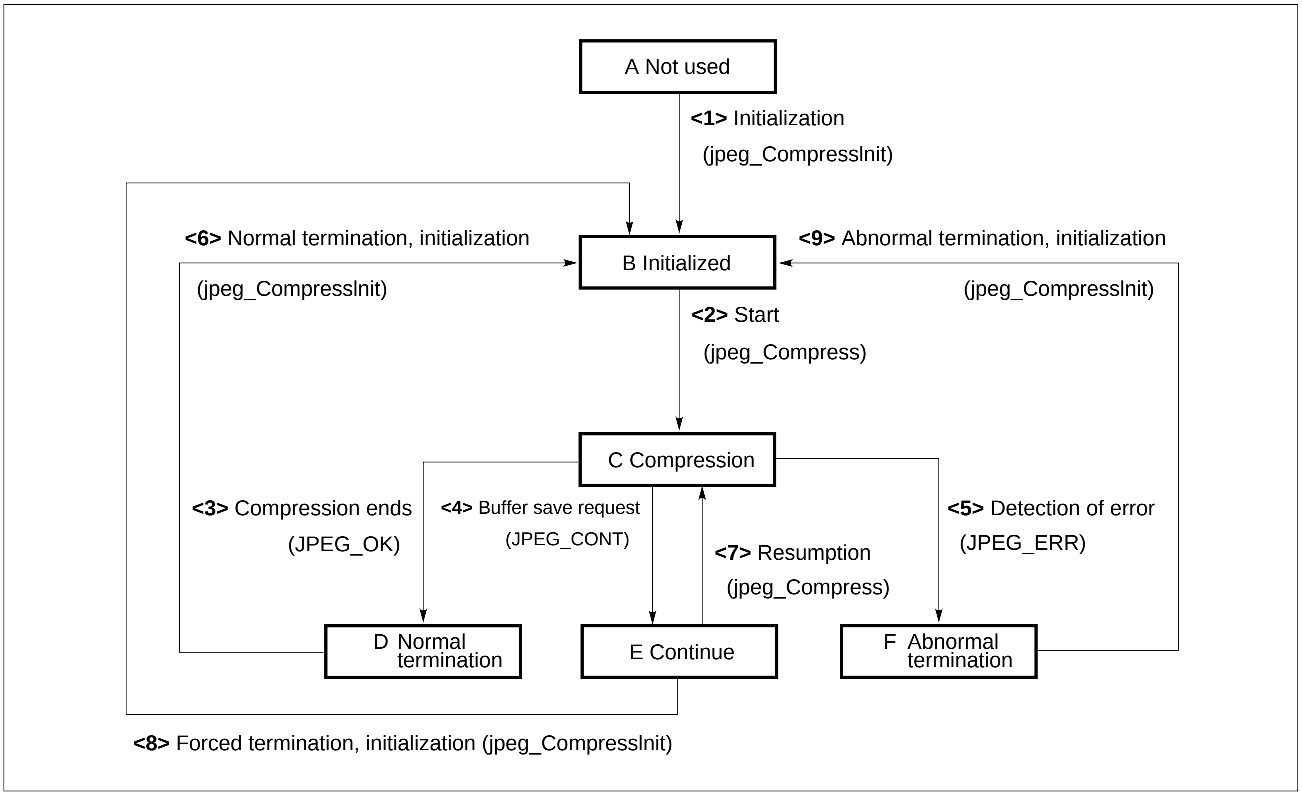


Table 2-3. Compression Status

Status	Explanation
A Not used	Status in which library has never been initialized since being started, such that compression cannot be executed
B Initialized	Status in which library has been initialized, allowing compression to be executed
C Compressed	Compression execution status
D Normal termination	Status in which compression has terminated normally
E Continue	Status in which processing has been aborted because the JPEG file storage buffer has become full during compression and a request to save the contents of the buffer has been issued. Compression can be continued.
F Abnormal termination	Status in which processing has been terminated because an error was detected during compression. Compression cannot be continued.

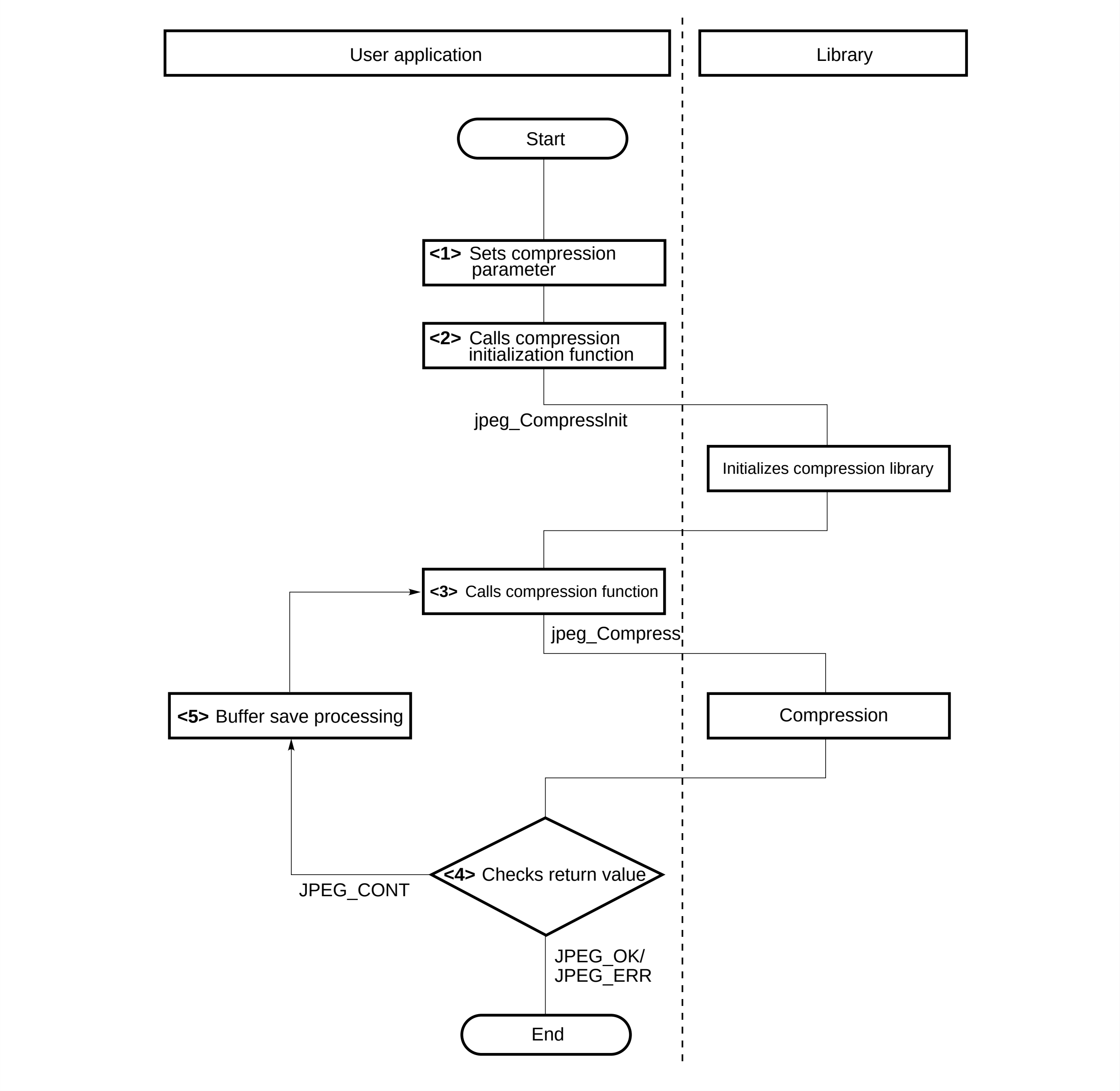
Table 2-4. Compression Status Transition

Status transition	Explanation	Function/return value
<1> A –> B	Library initialization	Function: jpeg_CompressInit
<2> B –> C	Start of compression	Function: jpeg_Compress
<3> C –> D	Normal termination	Return value: JPEG_OK
<4> C –> E	Processing aborted and request to save data in buffer is issued because the JPEG file storage buffer is full.	Return value: JPEG_CONT
<5> C –> F	Processing terminated because error was detected	Return value: JPEG_ERR
<6> D –> B	Library initialization after normal termination	Function: jpeg_CompressInit
<7> E –> C	Compression resumption	Function: jpeg_Compress
<8> E –> B	Library initialization after forced termination	Function: jpeg_CompressInit
<9> F –> B	Library initialization after abnormal termination	Function: jpeg_CompressInit
Others	Transition impossible	None

(3) Processing flow

The following figure shows the flow of application processing using the compression library.

Figure 2-2. Compression Flow


User application
Library
Start
<1> Sets compression parameter
<2> Calls compression initialization function
jpeg_CompressInit
Initializes compression library
<3> Calls compression function
jpeg_Compress
<5> Buffer save processing
Compression
<4> Checks return value
JPEG_CONT
JPEG_OK/
JPEG_ERR
End

<1> **Setting compression parameter**

Define variables having JPEGINFO type structure, as defined by header file “jpeg.h,” and set members (see **Section 2.5.1**). When executing new compression, the members must be set first. When continuing processing, the members need not be set again.

Subsequently, do not set parameters.

Table 2-5. User-Set Members

Member	Data to be set
Quality	Quantization factor
Sampling	Sampling ratio
Restart	Restart interval
Width	Horizontal size of compressed image
Height	Vertical size of compressed image
StartX	Start x coordinate of compressed image
StartY	Start y coordinate of compressed image
VRAM_Bptr	First address of VRAM
VRAM_W_Pixel	Number of pixels in horizontal direction
VRAM_H_Pixel	Number of pixels in vertical direction
VRAM_Line_Byte	Number of bytes equivalent to VRAM address difference of 1 pixel in vertical direction
VRAM_Pixel_Byte	Number of bytes equivalent to VRAM address difference of 1 pixel in horizontal direction
VRAM_Gap1_Byte	Number of bytes equivalent to VRAM address difference of Y/Cb or R/G component of same pixel
VRAM_Gap2_Byte	Number of bytes equivalent to VRAM address difference of Y/Cr or R/B component of same pixel
JPEG_Buff_Bptr	First address of JPEG file storage buffer
JPEG_Buff_Eptr	End address of JPEG file storage buffer
MCU_Buff_Bptr	First address of MCU buffer
DQT_Y_Bptr	First address of quantization table for luminance component
DQT_C_Bptr	First address of quantization table for chrominance component
DHT_DC_Y_Bptr	First address of Huffman table for DC luminance component
DHT_DC_C_Bptr	First address of Huffman table for DC chrominance component
DHT_AC_Y_Bptr	First address of Huffman table for AC luminance component
DHT_AC_C_Bptr	First address of Huffman table for AC chrominance component
APP_Info_Bptr	When APPn marker is supported: first address of APPINFO structure variable When APPn marker is not supported: 0
Work1_Bptr	First address of buffer for library work area
Work2_Bptr	First address of buffer for library work area

<2> Calling compression initialization function [jpeg_CompressInit]

Call the compression initialization function by using the pointer to the structure set in **<1>**, above, as an argument.

This function initializes the data required for compression and sets the compression library to an executable status. Call this function first when executing new compression. When continuing compression, this function need not be called.

<3> Calling compression function [jpeg_Compress]

Call the compression function by using the pointer to the structure set in **<2>**, above, as an argument.

<4> Checking return value [JPEG_OK/JPEG_ERR/JPEG_CONT]

The return value of jpeg_Compress is as follows:

Table 2-6. Return Value of jpeg_Compress

Return value	Meaning	Explanation
JPEG_OK	Normal termination	Compression terminated normally. Start from **<1>** again to execute new compression.
JPEG_ERR	Abnormal termination	Processing aborted because error was detected. Error details are stored in member “ErrorState” of JPEGINFO structure. Start from **<1>** again to execute new compression.
JPEG_CONT	Continue	Processing has been stopped because specified JPEG file storage buffer is full. Start from **<1>** again to forcibly terminate processing and execute new compression. To continue, execute **<5>** buffer save processing and start from **<3>** again.

Table 2-7. Compression Result Information Members

Member	Data to be stored
ErrorState	Error status
FileSize	JPEG file size

<5> Buffer save processing

Save the contents of the JPEG file storage buffer.

2.2.2 Expansion

Expansion involves expanding the JPEG file and displaying it on the screen.

(1) Function

Function name	Description
jpeg_DecompressInit	JPEG expansion library initialization
jpeg_Decompress	JPEG expansion

(2) Status transition

The status transition during expansion is illustrated below.

Figure 2-3. Expansion Status Transition

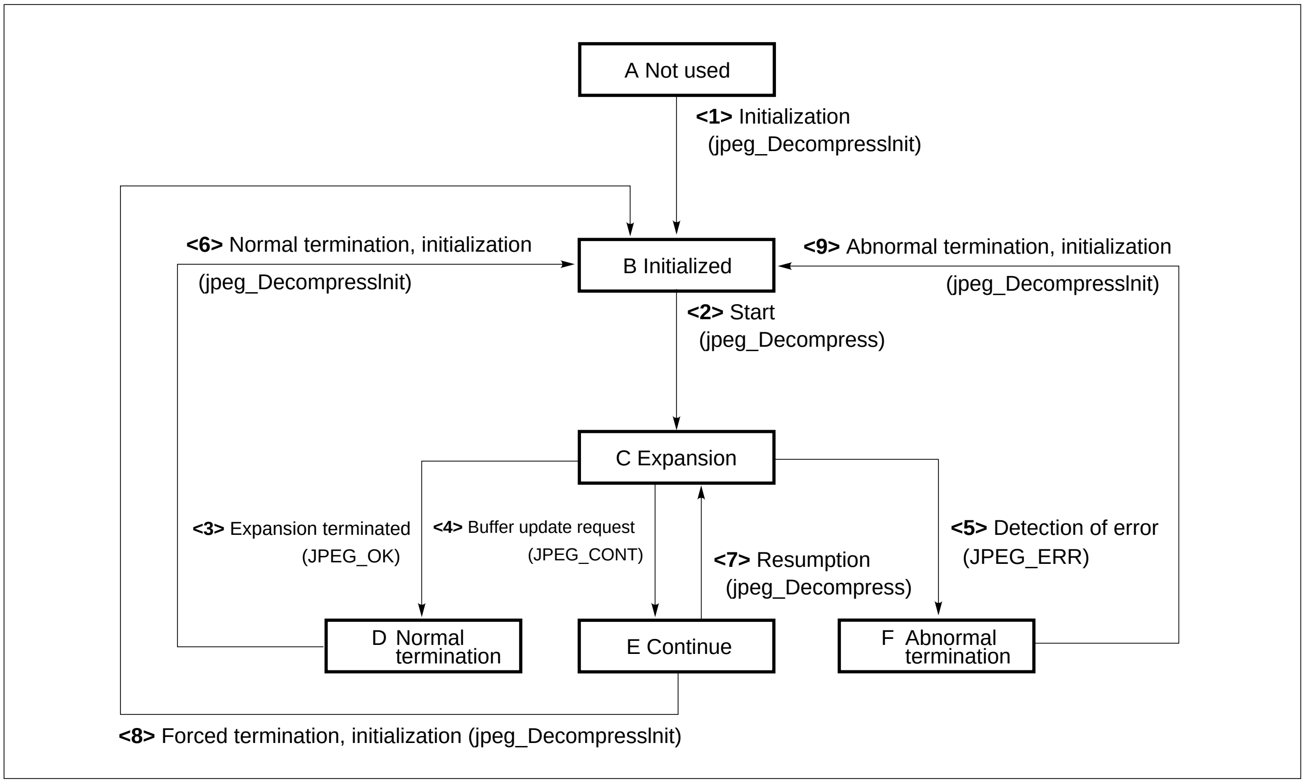

A Not used
<1> Initialization (jpeg_DecompressInit)
<6> Normal termination, initialization (jpeg_DecompressInit)
B Initialized
<9> Abnormal termination, initialization (jpeg_DecompressInit)
<2> Start (jpeg_Decompress)
C Expansion
<3> Expansion terminated (JPEG_OK)
<4> Buffer update request (JPEG_CONT)
<7> Resumption (jpeg_Decompress)
<5> Detection of error (JPEG_ERR)
D Normal termination
E Continue
F Abnormal termination
<8> Forced termination, initialization (jpeg_DecompressInit)

Table 2-8. Expansion Status

Status	Explanation
A Not used	Status in which library has never been initialized since start and expansion cannot be executed
B Initialized	Status in which library has been initialized and expansion can be executed
C Expansion	Expansion execution status
D Normal termination	Status in which expansion has been terminated normally
E Continue	Status in which expansion in buffer to which JPEG file has been stored has been terminated, but processing has been aborted because JPEG file end code cannot be found, causing buffer update request to be issued. Expansion can be continued.
F Abnormal termination	Status in which processing has been terminated because error was detected during expansion. Expansion cannot be continued.

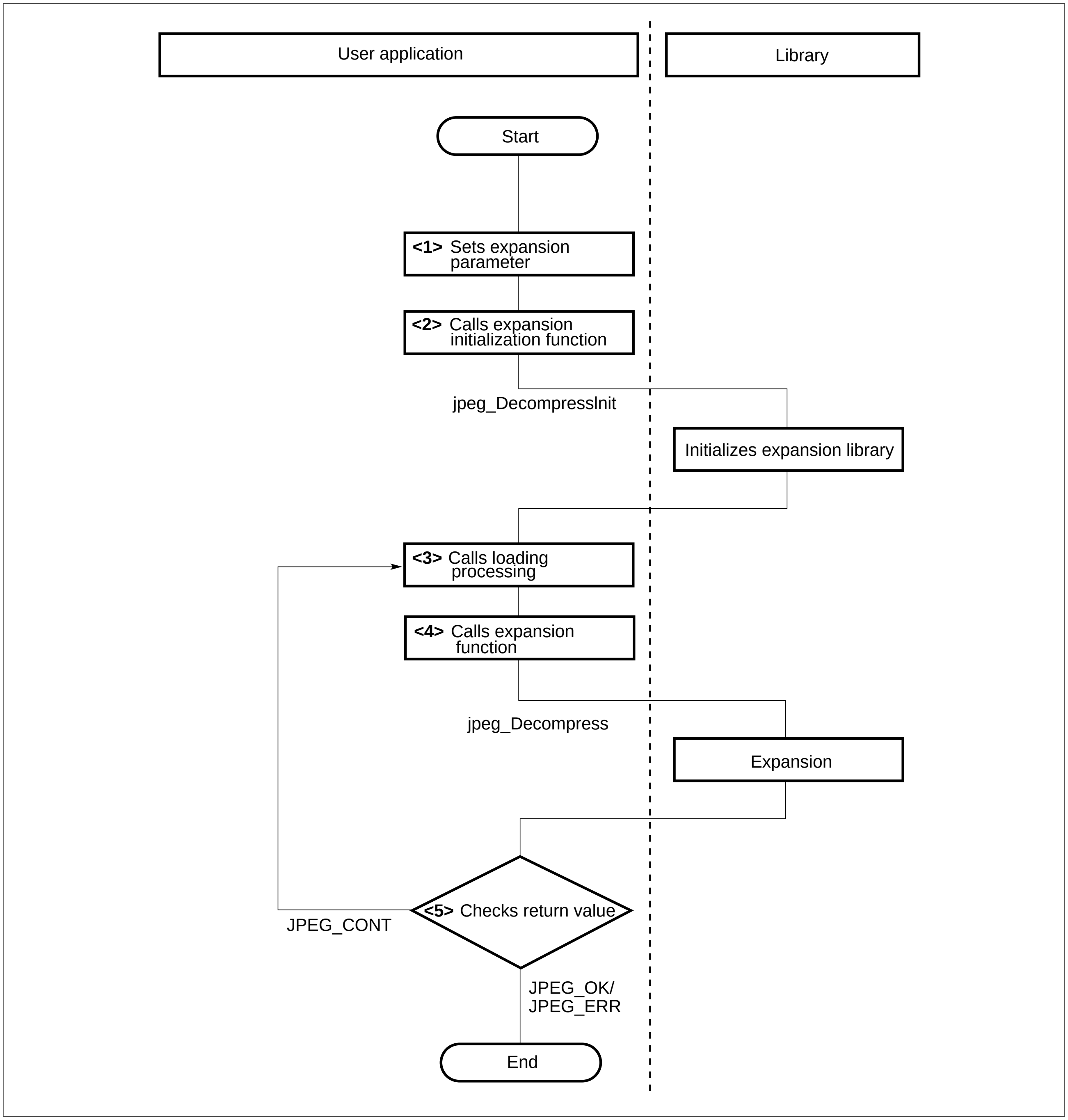
Table 2-9. Expansion Status Transition

Status transition	Explanation	Function/return value
<1> A –> B	Library initialization	Function: jpeg_DecompressInit
<2> B –> C	Start of expansion	Function: jpeg_Decompress
<3> C –> D	Normal termination	Return value: JPEG_OK
<4> C –> E	Data in JPEG file storage buffer has been expanded but JPEG file end code cannot be found. Therefore, processing has been aborted and updating of data in the buffer has been requested.	Return value: JPEG_CONT
<5> C –> F	Processing terminated because error was detected	Return value: JPEG_ERR
<6> D –> B	Library initialization after normal termination	Function: jpeg_DecompressInit
<7> E –> C	Resumption of expansion	Function: jpeg_Decompress
<8> E –> B	Library initialization after forced termination	Function: jpeg_DecompressInit
<9> F –> B	Library initialization after abnormal termination	Function: jpeg_DecompressInit
Others	Transition impossible	None

(3) Processing flow

The following figure illustrates the flow of processing of an application that uses the expansion library.

Figure 2-4. Expansion Flow


User application
Library
Start
<1> Sets expansion parameter
<2> Calls expansion initialization function
jpeg_DecompressInit
Initializes expansion library
<3> Calls loading processing
<4> Calls expansion function
jpeg_Decompress
Expansion
<5> Checks return value
JPEG_CONT
JPEG_OK/
JPEG_ERR
End

<1> Setting expansion parameters

Define the variable of the JPEGINFO structure defined by header file "jpeg.h" and set members (see **Section 2.5.1**). The members must be set when executing new expansion. When continuing expansion, the members need not be set again.

Subsequently, do not set parameters.

Table 2-10. User-Set Members

Member	Data to be set
StartX	Start x coordinate of expanded image
StartY	Start y coordinate of expanded image
VRAM_Bptr	First address in VRAM
VRAM_W_Pixel	Number of pixels in horizontal direction
VRAM_H_Pixel	Number of pixels in vertical direction
VRAM_Line_Byte	Number of bytes equivalent to VRAM address differing by one pixel in vertical direction
VRAM_Pixel_Byte	Number of bytes equivalent to VRAM address differing by one pixel in horizontal direction
VRAM_Gap1_Byte	Number of bytes equivalent to VRAM address difference of Y/Cb or R/G component for same pixel
VRAM_Gap2_Byte	Number of bytes equivalent to VRAM address difference of Y/Cr or R/B component for same pixel
JPEG_Buff_Bptr	First address of JPEG file storage buffer
JPEG_Buff_Eptr	End address of JPEG file storage buffer
MCU_Buff_Bptr	First address of MCU buffer
APP_Info_Bptr	0
Work1_Bptr	First address of library work area buffer
Work2_Bptr	First address of library work area buffer

<2> Calling expansion initialization function [jpeg_DecompressInit]

Call the expansion initialization function by using the pointer to the structure defined in **<1>**, above, as an argument.

This function initializes the data required for expansion and sets the expansion library to executable status. Call this function first when executing new expansion. When continuing expansion, this function need not be called.

<3> Buffer loading (updating) processing

Load data into the JPEG file storage buffer and update it.

<4> Calling expansion function [jpeg_Decompress]

Call the expansion function by using the pointer to the structure used in **<2>** as an argument.

<5> Checking return value [JPEG_OK/JPEG_ERR/JPEG_CONT]

The return value of jpeg_Decompress is as follows:

Table 2-11. jpeg_Decompress Return Values

Return value	Meaning	Explanation
JPEG_OK	Normal termination	Expansion has terminated normally. Start from **<1>** again to execute new expansion.
JPEG_ERR	Abnormal termination	Processing terminated because an error occurred. Details of the error are stored in the member of JPEGINFO structure “ErrorState.” Start from **<1>** again to execute new expansion.
JPEG_CONT	Continue	Processing terminated because JPEG file end code cannot be found after expansion in the specified JPEG file storage buffer. To forcibly terminate and execute new expansion, start from **<1>** again. To continue, execute **<3>** buffer update processing, then start from **<4>** again.

Table 2-12. Expansion Result Information Members

Member	Data to be stored
ErrorState	Error status
FileSize	JPEG file size
Sampling	Sampling ratio
Restart	Restart interval
Width	Horizontal size of image
Height	Vertical size of image

2.2.3 Analysis

Analysis analyzes the JPEG file and obtains the following data:

<1> Sampling ratio
<2> Restart interval
<3> Size of image (horizontal/vertical)
<4> JPEG file size
<5> Application data (APP data)

(1) Function

Function name	Description
jpeg_AnalysisInit	JPEG analysis library initialization
jpeg_Analysis	JPEG analysis

(2) Status transition

The following figure illustrates the status transition during analysis.

Figure 2-5. Analysis Status Transition

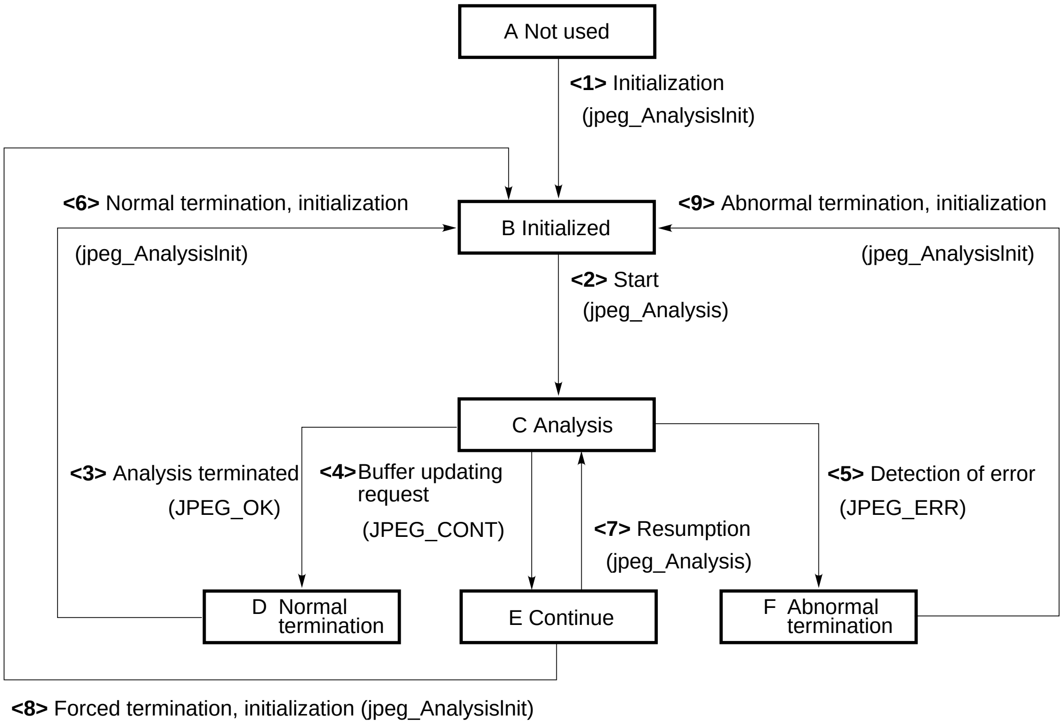


Table 2-13. Analysis Status

Status	Explanation
A Not used	Status in which library has never been initialized since start and analysis cannot be executed
B Initialized	Status in which library has been initialized and analysis can be executed
C Expansion	Analysis execution status
D Normal termination	Status in which analysis has been terminated normally
E Continue	Status in which analysis in buffer to which JPEG file has been stored has been terminated, but processing has been aborted because JPEG file end code cannot be found, causing buffer update request to be issued. Analysis can be continued.
F Abnormal termination	Status in which processing has been terminated because error was detected during analysis. Analysis cannot be continued.

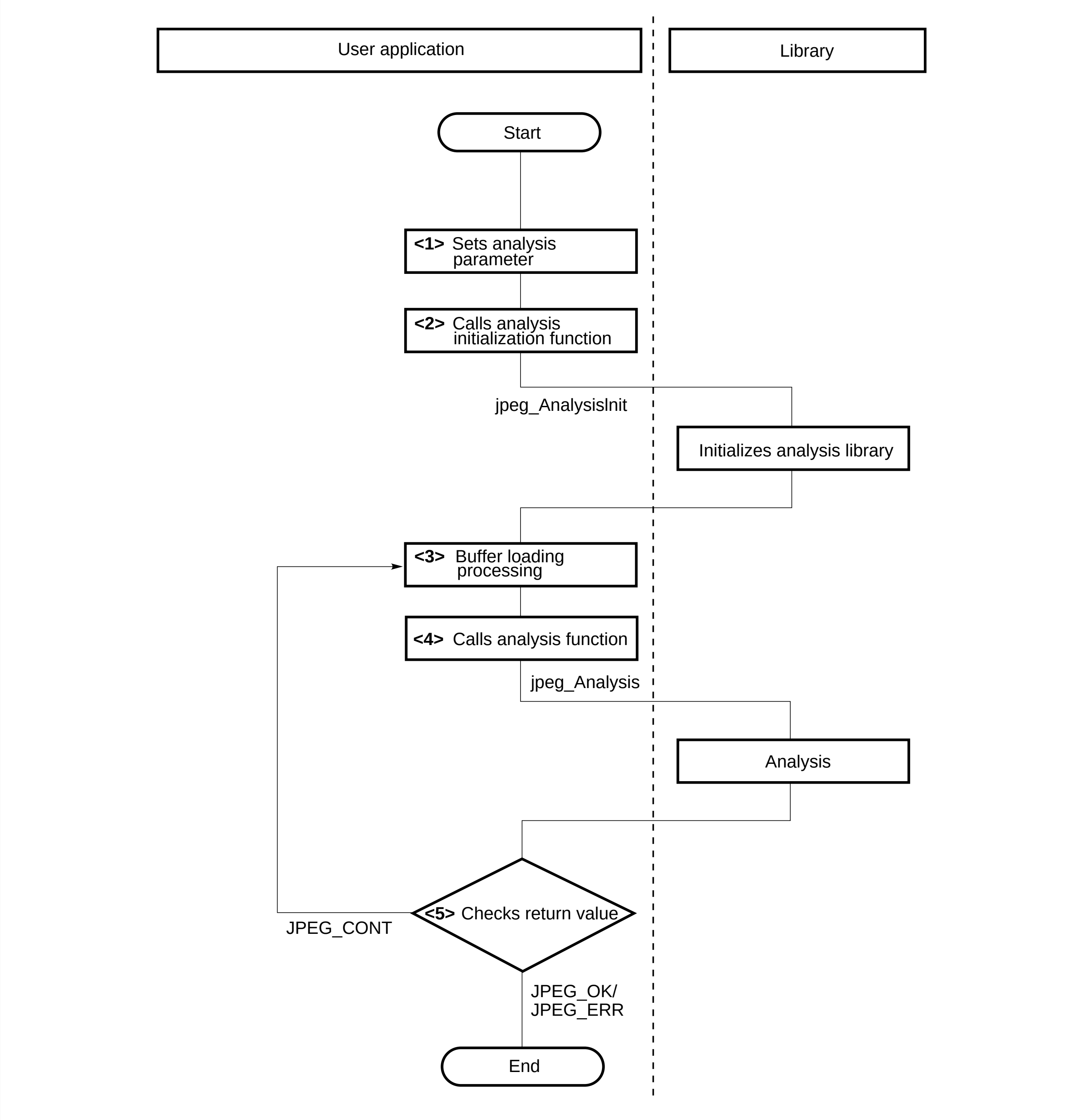
Table 2-14. Analysis Status Transition

Status transition	Explanation	Function/return value
<1> A –> B	Library initialization	Function: jpeg_AnalysisInit
<2> B –> C	Start of analysis	Function: jpeg_Analysis
<3> C –> D	Normal termination	Return value: JPEG_OK
<4> C –> E	Data in JPEG file storage buffer has been analyzed but end code of JPEG file cannot be found. Therefore, processing is aborted and updating of the data in the buffer has been requested.	Return value: JPEG_CONT
<5> C –> F	Processing terminated because error was detected	Return value: JPEG_ERR
<6> D –> B	Library initialization after normal termination	Function: jpeg_AnalysisInit
<7> E –> C	Resumption of analysis	Function: jpeg_Analysis
<8> E –> B	Library initialization after forced termination	Function: jpeg_AnalysisInit
<9> F –> B	Library initialization after abnormal termination	Function: jpeg_AnalysisInit
Others	Transition impossible	None

(3) Processing flow

The following figure illustrates the flow of processing of an application that uses the analysis library.

Figure 2-6. Analysis Flow


User application
Library
Start
<1> Sets analysis parameter
<2> Calls analysis initialization function
jpeg_AnalysisInit
Initializes analysis library
<3> Buffer loading processing
<4> Calls analysis function
jpeg_Analysis
Analysis
<5> Checks return value
JPEG_CONT
JPEG_OK/
JPEG_ERR
End

<1> Setting of analysis parameters

Define the variable of the JPEGINFO structure defined by header file “jpeg.h” and set members (see **Section 2.5.1**). The members must be set to execute new analysis. When continuing analysis, the members need not be set again.

Subsequently, do not set parameters.

Table 2-15. User-Set Members

Member	Data to be set
JPEG_Buff_Bptr	First address of JPEG file storage buffer
JPEG_Buff_Eptr	End address of JPEG file storage buffer
APP_Info_Bptr	When APPn marker is supported: first address of APPINFO structure variable When APPn marker is not supported: 0
Work1_Bptr	First address of library work area buffer

<2> Calling analysis initialization function [jpeg_AnalysisInit]

Call the analysis initialization function by using the pointer to the structure defined in **<1>**, above, as an argument.

This function initializes the data required for analysis and sets the analysis library to executable status. Call this function first when executing new analysis. When continuing analysis, this function need not be called.

<3> Buffer loading (updating) processing

Load data into the JPEG file storage buffer and update it.

<4> Calling analysis function [jpeg_Analysis]

Call the analysis function by using the pointer to the structure used in **<2>** as an argument.

<5> Checking return value [JPEG_OK/JPEG_ERR/JPEG_CONT]

The return value of jpeg_Analysis is as follows:

Table 2-16. Return Value of jpeg_Analysis

Return value	Meaning	Explanation
JPEG_OK	Normal termination	Analysis has been terminated normally. Start from **<1>** again to execute new analysis.
JPEG_ERR	Abnormal termination	Processing has been terminated because an error occurred. Details of the error are stored in JPEGINFO structure member "ErrorState." Start from **<1>** again to execute new analysis.
JPEG_CONT	Continue	Processing has been terminated because the JPEG file end code cannot be found after analysis in a specified JPEG file storage buffer was completed. To forcibly terminate and execute new analysis, start from **<1>** again. To continue, execute **<3>** buffer updating processing, then start from **<4>** again.

Table 2-17. Analysis Result Information Members

Member	Data to be stored
ErrorState	Error status
Sampling	Sampling ratio
Restart	Restart interval
Width	Horizontal size of image
Height	Vertical size of image
FileSize	JPEG file size
APP_Info_Bptr	The address and data size of an APPn marker are stored to a member of a specified APPINFO structure variable only when the address of the APPINFO structure variable is set in this member when initial parameters are set.

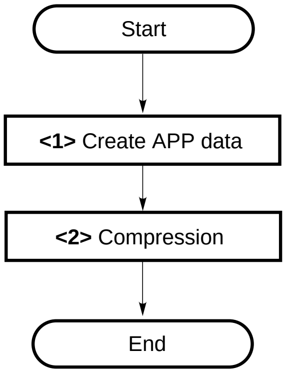
2.3 APPn Marker

The processing sequences when the APPn marker is supported are shown below.

(1) Compression

The flow of the processing of an application that uses the compression library when the APPn marker is supported is shown below.

Figure 2-7. Compression When APPn Marker Is Used



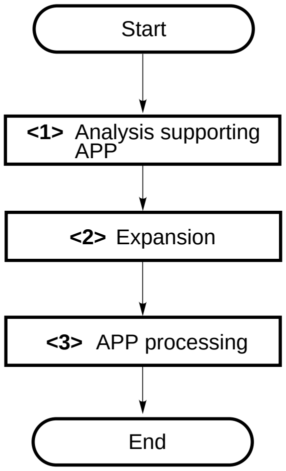
<1> Creating APP data
Define the variable of the APPINFO structure defined by header file "jpeg.h" by using the APP data and set members.
For details, see **Section 2.5.2**.

<2> Compression
For details, see **Section 2.2.1**.

(2) Expansion

The flow of the processing of an application that uses the expansion library when the APPn marker is supported is shown below.

Figure 2-8. Expansion When APPn Marker Is Used



<1> Analysis supporting APP
For details, see **Section 2.2.3**.

<2> Expansion
For details, see **Section 2.2.2**.

<3> APP processing
Perform processing supporting the APPn marker based on the data stored in the APPINFO structure variable in **<1>**, above.

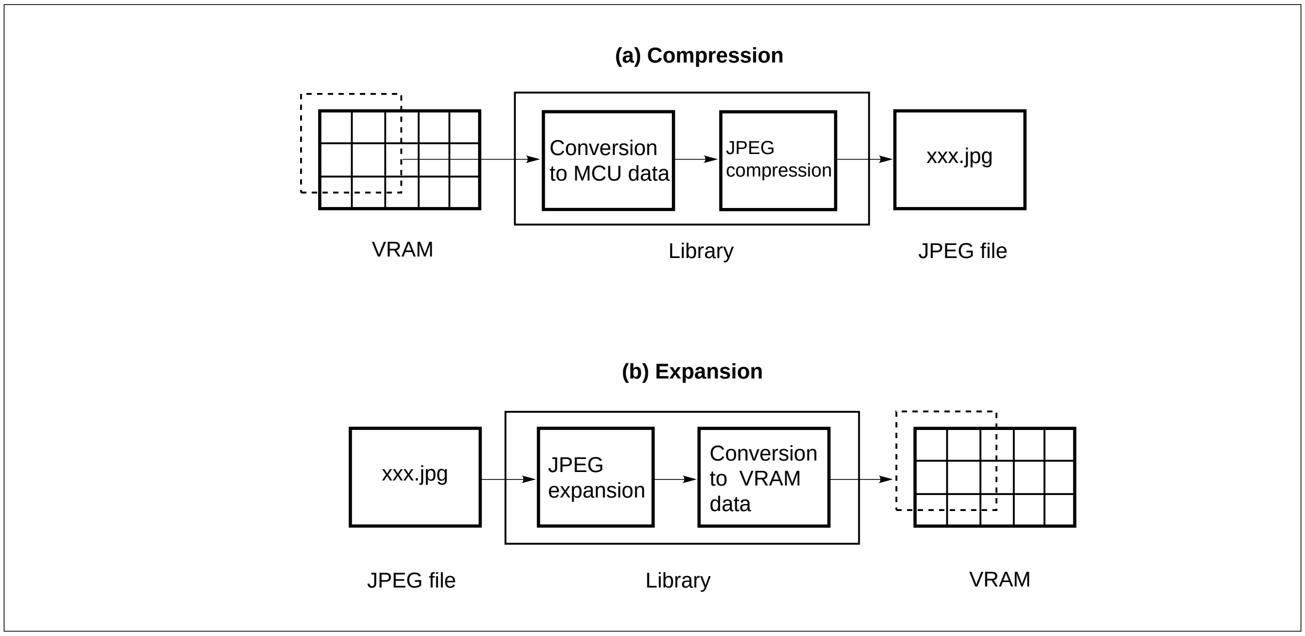
(3) Analysis

For details of how to use the analysis library when the APPn marker is supported, see **Section 2.2.3**.

2.4 DATA CONVERSION/TRANSFER BETWEEN VRAM-MCU BUFFERS

The following figure outlines the processing performed for the library.

Figure 2-9. Data Conversion/Transfer between VRAM-MCU Buffers



With JPEG compression, the MCU is the basic unit of processing (for details of the MCU, see **Section 2.4.2**). Compression involves conversion from VRAM data to MCU, while expansion involves conversion from MCU to VRAM data.

The library provides functions for converting or transferring data between the VRAM-MCU buffers in MCU units by default.

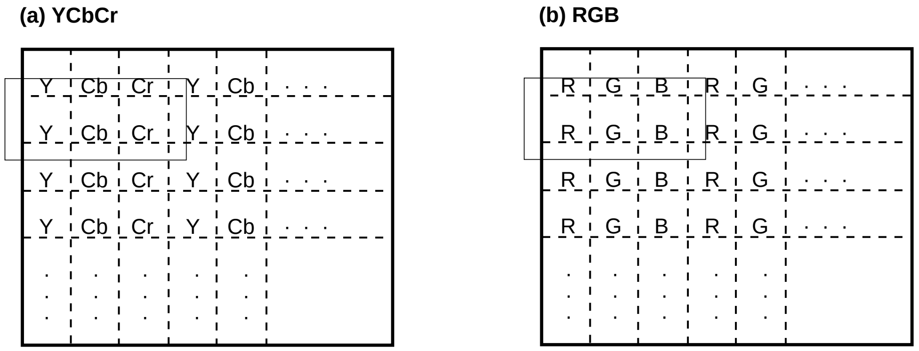
The user can create his or her own functions by using these conversion routines (such as when a monochrome image is to be displayed). For details, see **Section 2.10**.

2.4.1 VRAM Configuration

The VRAM configuration assumed by the library (default conversion routine) is shown below.

- Both Y/Cb/Cr and R/G/B are continuous
- Can be accessed in byte units
- 24-bit full color

Figure 2-10. VRAM Image



Remarks 1. Data is input/output in block (MCU) units in the horizontal direction.

2. ▭ : 1 pixel

The members of the JPEGINFO structure shown in Table 2-18 must be set. For details of the JPEGINFO structure, see **Section 2.5.1**.

Table 2-18. VRAM Setting Members

Member	Explanation
VRAM_W_Pixel	Number of pixels in horizontal direction of assumed image
VRAM_H_Pixel	Number of pixels in vertical direction of assumed image
VRAM_Line_Byte	Number of bytes equivalent to VRAM address difference of one pixel in vertical direction
VRAM_Pixel_Byte	Number of bytes equivalent to VRAM address difference of one pixel in horizontal direction
VRAM_Gap1_Byte	YCbCr: Number of bytes equivalent to VRAM address difference of Y and Cb components of same pixel RGB: Number of bytes equivalent to VRAM address difference of R and G components of same pixel
VRAM_Gap2_Byte	YCbCr: Number of bytes equivalent to VRAM address difference of Y and Cr components of same pixel RGB: Number of bytes equivalent to VRAM address difference of R and B components of same pixel

Setting Example

[VRAM specification]

Number of horizontal pixels of assumed image	: 320
Number of vertical pixels of assumed image	: 240
Number of horizontal bytes of VRAM	: 0x3C0
Mode	: YCbCr
1 pixel	: 3 bytes
Y = Cb = Cr = 1 byte/continuous	
JPEGINFO structure variable	: *x*

–>

[Setting member of JPEGINFO structure variable]

x.VRAM_W_Pixel	= 320 ;
x.VRAM_H_Pixel	= 240 ;
x.VRAM_Line_Byte	= 0x3C0 ;
x.VRAM_Pixel_Byte	= 3 ;
x.VRAM_Gap1_Byte	= 1 ;
x.VRAM_Gap2_Byte	= 2 ;

2.4.2 MCU Buffer Structure

When using the JPEG library (compression/expansion), the user must prepare an MCU buffer to store data.

The MCU buffer stores data sampled in MCU units when compression is executed. DCT processing is performed on the stored data. If the VRAM is of RGB type, data is converted to YCbCr type and then stored. When expansion is executed, the result of reverse DCT processing is output in MCU units.

The configuration of the MCU buffer is as follows:

Table 2-19. MCU Buffer Configuration

Declaration	Size (bytes)	Sampling ratio
short *name* [3] [64]	384	Only 4:4:4 supported
short *name* [4] [64]	512	Up to 4:2:2 supported
short *name* [6] [64]	768	Up to 4:1:1 supported

Configure the buffer in the following manner:

(1) Define the entity in the application and pass the pointer to the member of the JPEGINFO structure variable. For details of the JPEGINFO structure, see **Section 2.5**.

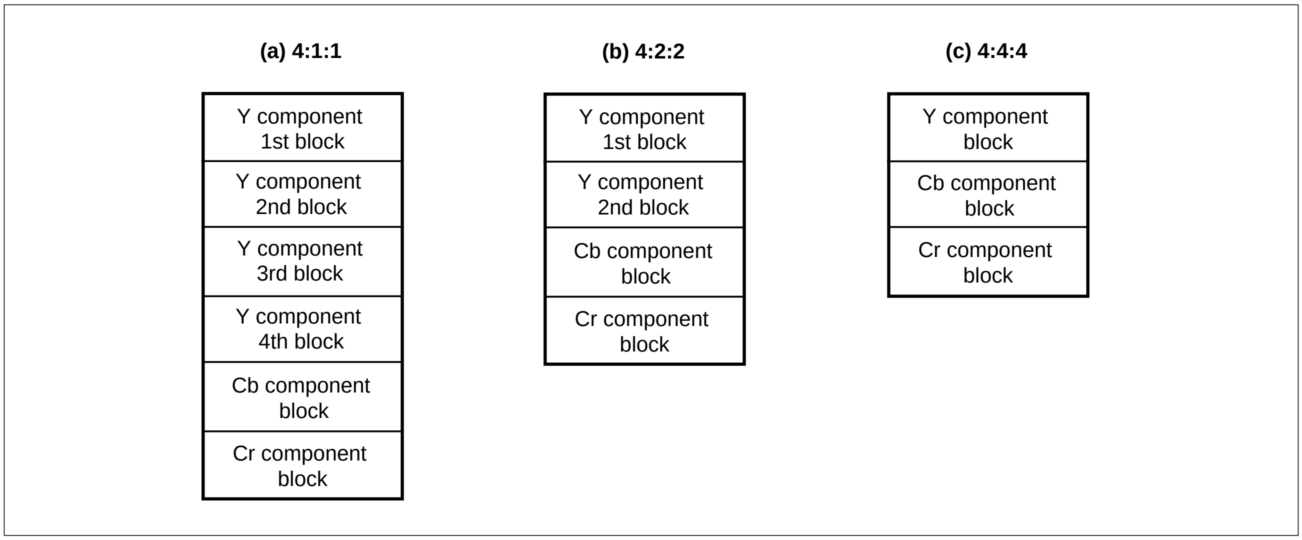
(2) The data is 8 bits long. It must be declared as being of short type (16 bits), however, because the data is used internally as a work buffer.

(3) Specify any name for *name*.

(4) A buffer must be prepared for each JPEGINFO structure variable.

Store each MCU into the buffer as shown below.

Figure 2-11. Storing MCU into Buffer



(a) 4:1:1

Y component 1st block
Y component 2nd block
Y component 3rd block
Y component 4th block
Cb component block
Cr component block

(b) 4:2:2

Y component 1st block
Y component 2nd block
Cb component block
Cr component block

(c) 4:4:4

Y component block
Cb component block
Cr component block

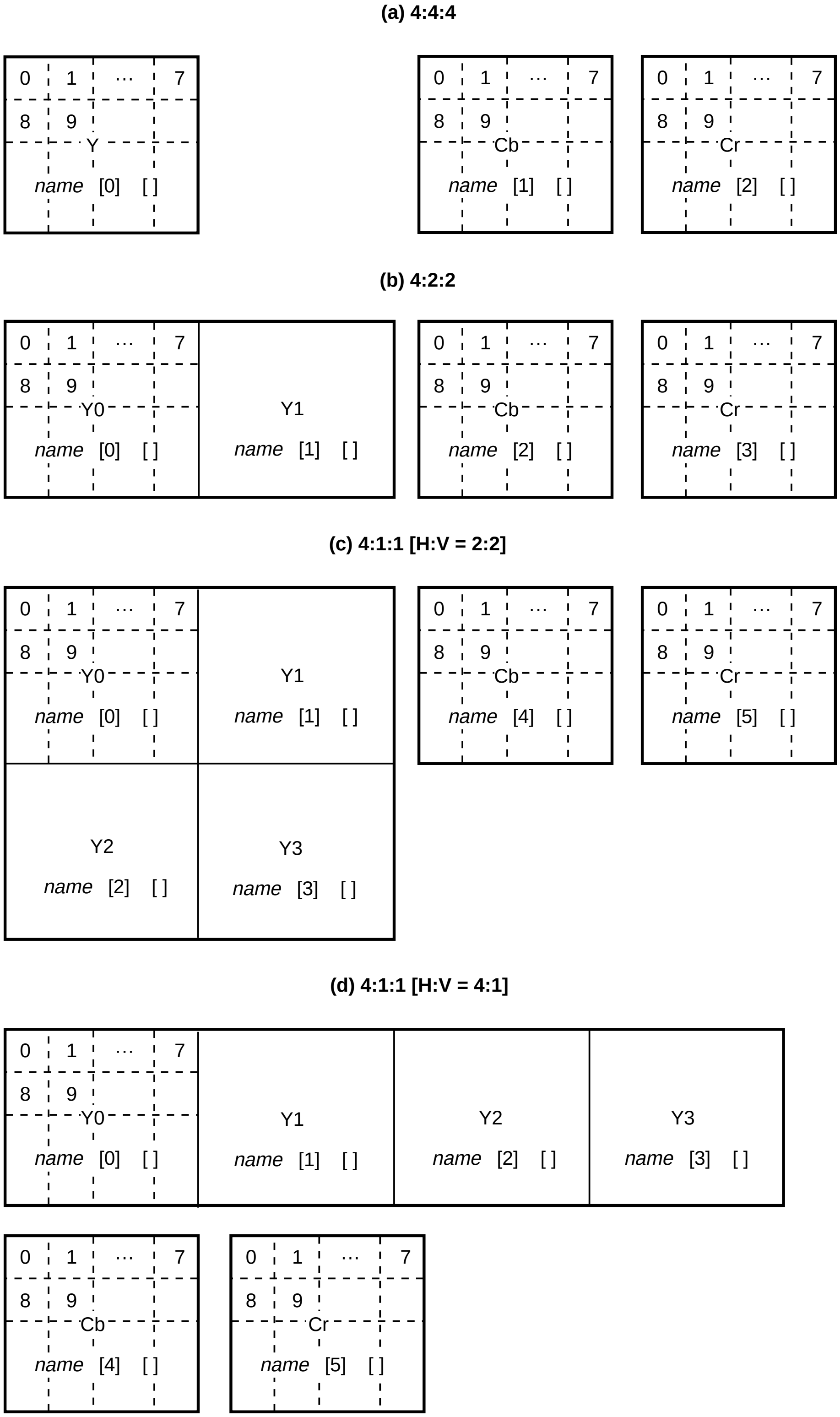
Define the following buffer.

Table 2-20. Buffer Definition

Processing	Required size
Compression	Buffer supporting compression sampling ratio. **Example** When executing compression at a sampling ratio of 4:2:2, regardless of the supported library, define *name* [4] [64] as the buffer.
Expansion, analysis	Define buffer with maximum size of supported library. **Example** If library supports sampling ratios of 4:4:4 and 4:2:2, define *name* [4] [64] as buffer.

The contents of the buffer are shown below.

Figure 2-12. Contents of MCU Buffer



(a) 4:4:4

(b) 4:2:2

(c) 4:1:1 [H:V = 2:2]

(d) 4:1:1 [H:V = 4:1]

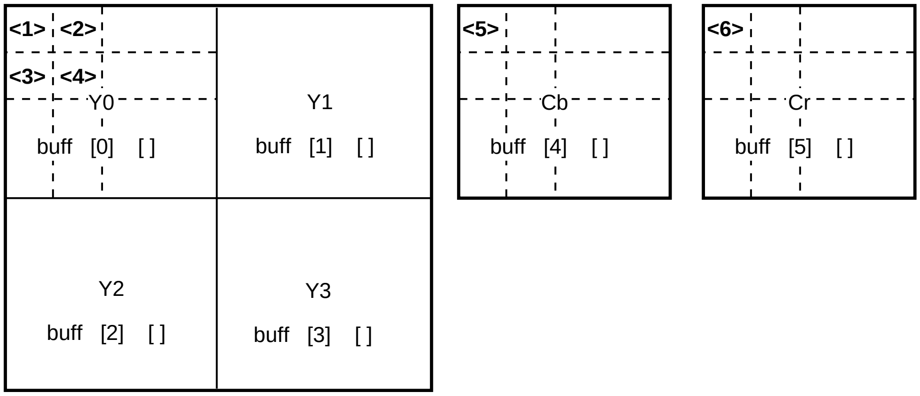
Remark 0, 1, ... : value of second element of matrix for MCU buffer

An example where the MCU buffer is defined as buff [6] [64] at a sampling ratio of 4:1:1 is shown below.

Table 2-21. Storing Data of 1 MCU (at 4:1:1)

Matrix	Data stored
buff [0] [ ]	Y component (upper left, 8 x 8 pixels)
buff [1] [ ]	Y component (upper right, 8 x 8 pixels)
buff [2] [ ]	Y component (upper left, 8 x 8 pixels)
buff [3] [ ]	Y component (lower right, 8 x 8 pixels)
buff [4] [ ]	Cb component (16 x 16 pixels: However, average of 2 x 2 pixels for one element)
buff [5] [ ]	Cr component (16 x 16 pixels: However, average of 2 x 2 pixels for one element)

Figure 2-13. Storing Data of Upper Left, 2 x 2 Pixels



Component	Storage position
Y component	**<1>** buff [0] [0], **<2>** buff [0] [1], **<3>** buff [0] [8], **<4>** buff [0] [9]
Cb component	**<5>** buff [4] [0]
Cr component	**<6>** buff [5] [0]

2.5 DETAILS OF DATA TYPE

There are two data types, as shown in Table 2-22. Declare all the variables of these structures, and the variables that pass an address to a member, as external variables.

Table 2-22. Data Type

Classification	Data type name	Data type	Description
Common	JPEGINFO	Structure	Compression, expansion, analysis parameter setting
	APPINFO	Structure	APPn marker information saving

2.5.1 JPEGINFO Structure

The member configuration of the JPEGINFO structure is shown in the table below.

This same structure is used for JPEG compression, expansion, and analysis processing.

Table 2-23. Members of JPEGINFO Structure

Member	Type	Bytes	Explanation
ErrorState	long	4	Error status
FileSize	long	4	JPEG file size
Quality	char	1	Quantization factor
Sampling	char	1	Sampling ratio
Restart	ushort	2	Restart interval
Width	ushort	2	Horizontal size of image
Height	ushort	2	Vertical size of image
StartX	ushort	2	Processing start x coordinate (for assumed image)
StartY	ushort	2	Processing start y coordinate (for assumed image)
CurrentX	ushort	2	Current x coordinate (for assumed image)
CurrentY	ushort	2	Current y coordinate (for assumed image)
VRAM_Bptr	uchar*	4	VRAM first address
VRAM_W_Pixel	ushort	2	Number of pixels in horizontal direction of assumed image
VRAM_H_Pixel	ushort	2	Number of pixels in vertical direction of assumed image
VRAM_Line_Byte	short	2	Number of bytes equivalent to VRAM address difference of one pixel in vertical direction
VRAM_Pixel_Byte	short	2	Number of bytes equivalent to VRAM address difference of one pixel in horizontal direction
VRAM_Gap1_Byte	short	2	Number of bytes equivalent to VRAM address difference of YCb or RG
VRAM_Gap2_Byte	short	2	Number of bytes equivalent to VRAM address difference of YCr or RB
JPEG_Buff_Bptr	uchar*	4	First address of JPEG file storage buffer
JPEG_Buff_Eptr	uchar*	4	End address of JPEG file storage buffer
MCU_Buff_Bptr	short*	4	First address of MCU buffer
RGB_Buff_Bptr	short*	4	First address of RGB buffer
DQT_Y_Bptr	char*	4	First address of quantization table (for luminance component)
DQT_C_Bptr	char*	4	First address of quantization table (for chrominance component)
DHT_DC_Y_Bptr	char*	4	First address of Huffman table (for DC luminance component)
DHT_DC_C_Bptr	char*	4	First address of Huffman table (for DC chrominance component)
DHT_AC_Y_Bptr	char*	4	First address of Huffman table (for AC luminance component)
DHT_AC_C_Bptr	char*	4	First address of Huffman table (for AC chrominance component)
APP_Info_Bptr	APPINFO*	4	First address of APPINFO structure variable
Work1_Bptr	long*	4	First address of library work area 1
Work2_Bptr	long*	4	First address of library work area 2
Work3	char	36	Library work area 3

Remark uchar: unsigned char, ushort: unsigned short

Table 2-24. Setting of JPEGINFO Structure Member

Member	Compression		Expansion		Analysis	
	User setting	IN/OUT	User setting	IN/OUT	User setting	IN/OUT
ErrorState	x	OUT	x	OUT	x	OUT
FileSize	x	OUT	x	OUT	x	OUT
Quality	o	IN	x	Reserved field	x	Reserved field
Sampling	o	IN	x	OUT	x	OUT
Restart	o	IN	x	OUT	x	OUT
Width	o	IN	x	OUT	x	OUT
Height	o	IN	x	OUT	x	OUT
StartX	o	IN	o	IN	x	Reserved field
StartY	o	IN	o	IN	x	Reserved field
CurrentX	x	Reserved field	x	Reserved field	x	Reserved field
CurrentY	x	Reserved field	x	Reserved field	x	Reserved field
VRAM_Bptr	o	IN	o	IN	x	Reserved field
VRAM_W_Pixel	o	IN	o	IN	x	Reserved field
VRAM_H_Pixel	o	IN	o	IN	x	Reserved field
VRAM_Line_Byte	o	IN	o	IN	x	Reserved field
VRAM_Pixel_Byte	o	IN	o	IN	x	Reserved field
VRAM_Gap1_Byte	o	IN	o	IN	x	Reserved field
VRAM_Gap2_Byte	o	IN	o	IN	x	Reserved field
JPEG_Buff_Bptr	o	IN	o	IN	o	IN
JPEG_Buff_Eptr	o	IN	o	IN	o	IN
MCU_Buff_Bptr	o	IN	o	IN	x	Reserved field
RGB_Buff_Bptr	x	Reserved field	x	Reserved field	x	Reserved field
DQT_Y_Bptr	o	IN	x	Reserved field	x	Reserved field
DQT_C_Bptr	o	IN	x	Reserved field	x	Reserved field
DHT_DC_Y_Bptr	o	IN	x	Reserved field	x	Reserved field
DHT_DC_C_Bptr	o	IN	x	Reserved field	x	Reserved field
DHT_AC_Y_Bptr	o	IN	x	Reserved field	x	Reserved field
DHT_AC_C_Bptr	o	IN	x	Reserved field	x	Reserved field
APP_Info_Bptr	o	IN	o	IN	o	IN
Work1_Bptr	o	IN	o	IN	o	IN
Work2_Bptr	o	IN	o	IN	x	Reserved field
Work3	x	Reserved field	x	Reserved field	x	Reserved field

Remark o: Must be set by the user.

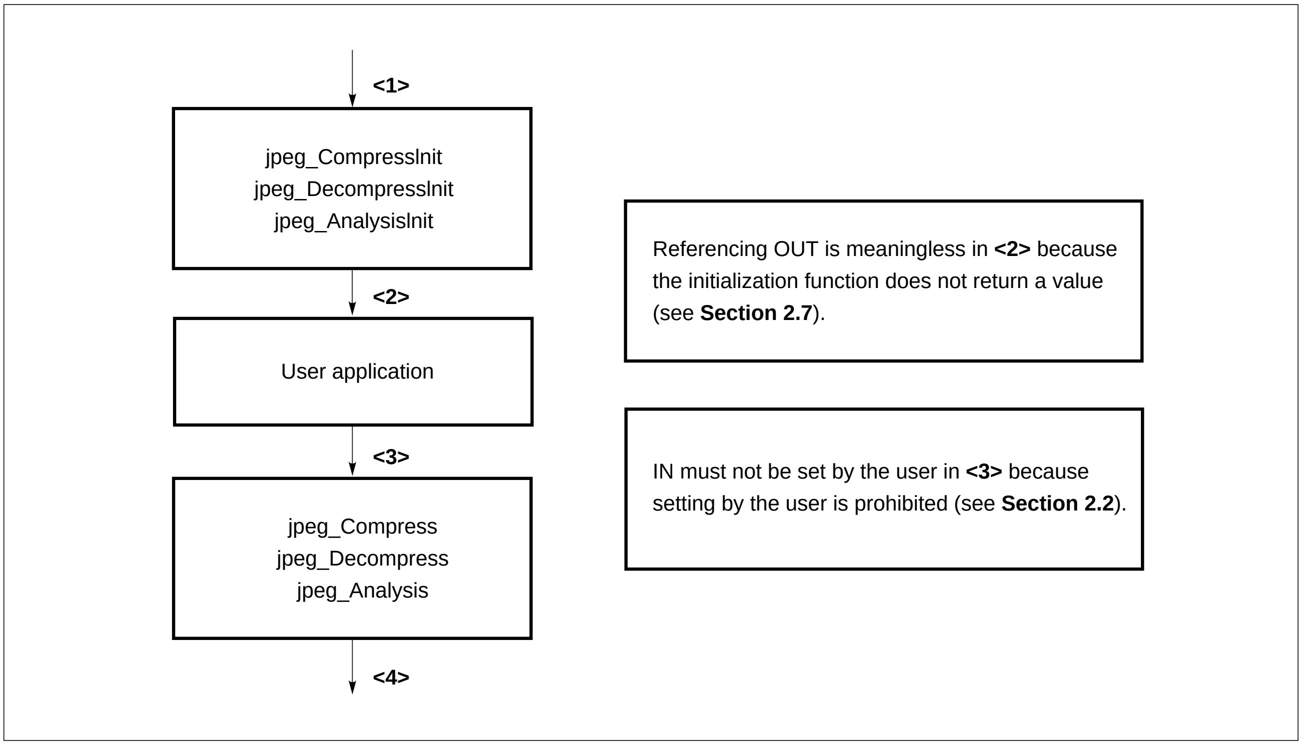
x: Must not be set by the user.
Because these members are used by the library, the user must never attempt to set them.

IN: Must be set by the user.

OUT: Library sets value (return value).

Reserved field: Reserved for library

[IN] indicates **<1>** in the figure below, while [OUT] indicates **<4>** because of the specifications of the library.



Referencing OUT is meaningless in **<2>** because the initialization function does not return a value (see **Section 2.7**).

IN must not be set by the user in **<3>** because setting by the user is prohibited (see **Section 2.2**).

Next, the details of each member are described.

(1) ErrorState

Classification	Explanation
Compression	Stores error status.
Expansion	
Analysis	

For details of the error status, see **Section 2.7**.

(2) FileSize

Classification	Explanation
Compression	Stores size of JPEG file.
Expansion	
Analysis	

(3) Quality

Classification	Value	Explanation
Compression	0 to 100	Set quantization factor.
Expansion	–	Reserved field
Analysis		

The quantization factor is used to perform processing on a defined quantization table. By setting this factor appropriately, the quality of the image can be adjusted.

Quantization factor	0	···	75	···	100
Status of image	Mosaic	···	Normal	···	Clear
Compression rate	High	···	Recommended value	···	Low

The lower the value of the factor, the smaller the size of the JPEG file created, but the more pronounced the “mosaic” effect. The higher the value, the finer the image, but the larger the size of the JPEG file created.

If the quantization factor is 75, the same table is created both before and after processing. It is therefore recommended that the coefficient be set to 75.

(4) Sampling

Classification	Value	Explanation
Compression	SAMPLE11 SAMPLE21 SAMPLE22 SAMPLE41	Set sampling ratio. Four sampling ratios are supported. This information is appended to the JPEG header.
Expansion	SAMPLE11 SAMPLE21 SAMPLE22 SAMPLE41	Sampling ratio is automatically stored from JPEG file. Four sampling ratios are supported. Because the expansion library automatically identifies the sampling ratio being used, the user does not have to consider the sampling ratio.
Analysis	Higher 4 bits: [0x1 to 0x4] Lower 4 bits: [0x1 to 0x4]	Sampling ratio is automatically stored from JPEG file.

SAMPLEn is defined by header file “jpeg.h” for compression/expansion.

Sampling (sampling ratio)	SAMPLE22 (4:1:1[H:V = 2:2])	SAMPLE21 (4:2:2[H:V = 2:1])	SAMPLE11 (4:4:4[H:V = 1:1])
Color	Normal	Fairly clear	Clear
File size	Reference value (x1)	About x4/3	About x2

(5) Restart

<table>
<tr><th>Classification</th><th>Value</th><th>Explanation</th></tr>
<tr><td>Compression</td><td>0 to 65535</td><td>Set value of restart interval.
<table>
<tr><th>Value</th><th>Restart interval</th><th>Explanation</th></tr>
<tr><td>0</td><td>None</td><td>DRI header/RSTn marker is not inserted.</td></tr>
<tr><td>Other than 0</td><td>Provided</td><td>Inserts RSTn marker for each specified MCU.</td></tr>
</table></td></tr>
<tr><td>Expansion</td><td rowspan="2">0 to 65535</td><td rowspan="2">Value of restart interval is automatically stored from JPEG file.
<table>
<tr><th>Value</th><th>Restart interval</th><th>Explanation</th></tr>
<tr><td>0</td><td>None</td><td>DRI header/RSTn marker is not included.</td></tr>
<tr><td>Other than 0</td><td>Provided</td><td>RSTn marker is inserted for each specified MCU.</td></tr>
</table></td></tr>
<tr><td>Analysis</td></tr>
</table>

(6) Width, Height

Classification	Value	Explanation
Compression	0 to 65535 (conditional)	Set horizontal/vertical size of image to be compressed.
Expansion	0 to 65535 (conditional)	Horizontal/vertical size of compressed image is automatically stored from JPEG file.
Analysis	0 to 65535	

The “value” is limited as follows, depending on the sampling ratio, for compression/expansion.

Sampling ratio	Width (horizontal value)	Height (vertical value)
4:4:4 [H:V = 1:1]	Multiple of 8	Multiple of 8
4:2:2 [H:V = 2:2]	Multiple of 16	Multiple of 8
4:1:1 [H:V = 2:2]	Multiple of 16	Multiple of 16
4:1:1 [H:V = 4:1]	Multiple of 32	Multiple of 8

(7) StartX, StartY

Classification	Value	Explanation
Compression	0 to 32767	Set start coordinates (x, y) of image to be compressed.
Expansion	0 to 32767	Set start coordinates (x, y) of image to be expanded.
Analysis	–	Reserved field

(8) CurrentX, CurrentY

Classification	Explanation
Compression	Current coordinates are stored.
Expansion	
Analysis	Reserved field

User reference is limited as follows depending on how the library is used.

Applicable library	Usage	User reference
jpeg_getMCU_*xx* jpeg_putMCU_*xx*	Default	Prohibited
	User-created	Reference is permitted in each function only.

For details of “jpeg_getMCU_*xx*” and “jpeg_putMCU_*xx*,” see **Sections 2.4 and 2.10**.

(9) VRAM_Bptr

Classification	Explanation
Compression	Specify first address of VRAM.
Expansion	
Analysis	Reserved field

(10) VRAM_W_Pixel, VRAM_H_Pixel, VRAM_Line_Byte, VRAM_Pixel_Byte, VRAM_Gap1_Byte, VRAM_Gap2_Byte

Classification	Explanation
Compression	Set specifications of image to be assumed.
Expansion	
Analysis	Reserved field

The contents to be set are as follows:

Member	Explanation
VRAM_W_Pixel	Number of pixels in horizontal direction
VRAM_H_Pixel	Number of pixels in vertical direction
VRAM_Line_Byte vertical direction	Number of bytes equivalent to VRAM address difference of one pixel in
VRAM_Pixel_Byte horizontal direction	Number of bytes equivalent to VRAM address difference of one pixel in
VRAM_Gap1_Byte	YCbCr: Number of bytes equivalent to VRAM address difference of Y and Cb components of same pixel RGB: Number of bytes equivalent to VRAM address difference of R and G components of same pixel
VRAM_Gap2_Byte	YCbCr: Number of bytes equivalent to VRAM address difference of Y and Cr components of same pixel RGB: Number of bytes equivalent to VRAM address difference of R and G components of same pixel

(11) JPEG_Buff_Bptr

Classification	Explanation
Compression	Specify first address of JPEG file storage buffer.
Expansion	
Analysis	

(12) JPEG_Buff_Eptr

Classification	Explanation
Compression	Set end address of buffer specified by member “JPEG_Buff_Bptr.” End address = first address + buffer size.
Expansion	
Analysis	

(13) MCU_Buff_Bptr

Classification	Explanation
Compression	Set first address of MCU buffer. For details, see **Section 2.4.2**.
Expansion	
Analysis	Reserved field

(14) RGB_Buff_Bptr

Classification	Explanation
Compression	Reserved field
Expansion	
Analysis	

(15) DQT_Y_Bptr, DQT_C_Bptr

Classification	Explanation
Compression	Specify first address of quantization table.
Expansion	Reserved field
Analysis	

The specified quantization table is as follows:

Member	Explanation
DQT_Y_Bptr	For luminance component
DQT_C_Bptr	For chrominance component

(16) DHT_DC_Y_Bptr, DHT_DC_C_Bptr, DHT_AC_Y_Bptr, DHT_AC_C_Bptr

Classification	Explanation
Compression	Specify first address of Huffman table.
Expansion	Reserved field
Analysis	

The specified quantization table is as follows:

Member	Explanation
DHT_DC_Y_Bptr	For DC luminance component
DHT_DC_C_Bptr	For DC chrominance component
DHT_AC_Y_Bptr	For AC luminance component
DHT_AC_C_Bptr	For AC chrominance component

(17) APP_Info_Bptr

<table>
<tr><th>Classification</th><th>Explanation</th></tr>
<tr><td>Compression</td><td>Set as follows:
<table>
<tr><th>APPn marker</th><th>Setting</th></tr>
<tr><td>Supported</td><td>Specify first address of APPINFO structure variable.</td></tr>
<tr><td>Not supported</td><td>Set 0.</td></tr>
</table></td></tr>
<tr><td>Expansion</td><td>Set 0.</td></tr>
<tr><td>Analysis</td><td>Set as follows:
<table>
<tr><th>APPn marker</th><th>Setting</th></tr>
<tr><td>Supported</td><td>Specify first address of APPINFO structure variable.</td></tr>
<tr><td>Not supported</td><td>Set 0.</td></tr>
</table></td></tr>
</table>

For the details of the APPINFO structure, see **Section 2.5.2**.

(18) Work1_Bptr

Classification	Explanation
Compression	Declare the buffer as being of “long type: 256 bytes” for the library work area, and specify the first address of the buffer.
Expansion	
Analysis	

Locating this buffer in internal RAM will result in an increase in the processing speed of the library.

(19) Work2_Bptr

<table>
<tr><th>Classification</th><th>Explanation</th></tr>
<tr><td>Compression</td><td>Declare a buffer of the following size as being of “long type” for the library work area, and specify the first address of the buffer.
<table>
<tr><th></th><th>Compression</th><th>Expansion</th></tr>
<tr><td>Size (bytes)</td><td>3072</td><td>8192</td></tr>
</table>
</td></tr>
<tr><td>Expansion</td><td></td></tr>
<tr><td>Analysis</td><td>Reserved field</td></tr>
</table>

(20) Work3

Classification	Explanation
Compression	Fixed work area of library.
Expansion	
Analysis	

2.5.2 Structure of APPINFO Structure

The member configuration of the APPINFO structure is shown below.

This structure is commonly used for JPEG compression/analysis, provided the APPn marker is supported.

Table 2-25. Members of APPINFO Structure

Member	Type	Bytes	Explanation
APP00_Buff_Bptr	uchar*	4	First address of APP0 data
. . .	. . .	. . .	. . .
APP15_Buff_Bptr	uchar*	4	First address of APP15 data
APP00_BuffSize	ushort	2	Length of APP0 data
. . .	. . .	. . .	. . .
APP15_BuffSize	ushort	2	Length of APP15 data

Each of these members is described in detail below.

(1) APPn_Buff_Bptr

	Compression	Analysis
User setting	Required	Prohibited
IN/OUT	IN	OUT

	Compression	Analysis
Used [APPn]	**Note 1**	**Note 3**
Not used [APPn]	**Note 2**	**Note 4**

Notes 1. Declare a data storage buffer, store data into the buffer, and specify the first address of the buffer in the corresponding member.

2. Specify 0 for the corresponding member.

3. The address of the APPn marker (absolute address of the JPEG file) is stored in the corresponding member.

4. 0 is stored in the corresponding member.

(2) APPn_BuffSize

	Compression	Analysis
User setting	Required	Prohibited
IN/OUT	IN	OUT

	Compression	Analysis
Used [APPn]	**Note 1**	**Note 3**
Not used [APPn]	**Note 2**	**Note 4**

Notes 1. Set a data size for the corresponding member.

2. Specify 0 for the corresponding member.

3. The length of the APPn segment is stored in the corresponding member.

4. 0 is stored in the corresponding member.

2.6 EXTERNAL VARIABLES

The following external variables are provided.

Table 2-26. External Variables

External variable	Type	Size (bytes)	Explanation
jpeg_DQT_Y	char	64	Quantization table (for luminance component)
jpeg_DQT_C	char	64	Quantization table (for chrominance component)
jpeg_DHT_DC_Y	char	About 50	Huffman table (for DC luminance component)
jpeg_DHT_DC_C	char	About 50	Huffman table (for DC chrominance component)
jpeg_DHT_AC_Y	char	About 300	Huffman table (for AC luminance component)
jpeg_DHT_AC_C	char	About 300	Huffman table (for AC chrominance component)

2.7 DETAILS OF FUNCTIONS

The following functions are provided.

Table 2-27. Functions

Classification	Function name	Description
Compression	jpeg_CompressInit	JPEG compression library initialization
	jpeg_Compress	JPEG compression, main
Expansion	jpeg_DecompressInit	JPEG expansion library initialization
	jpeg_Decompress	JPEG expansion, main
Analysis	jpeg_AnalysisInit	JPEG analysis library initialization
	jpeg_Analysis	JPEG analysis, main

2.7.1 Compression

(1) jpeg_CompressInit

Classification Compression (1/2)

Function name jpeg_CompressInit

Outline Initializes JPEG compression library.

Syntax #include “jpeg.h”
void jpeg_CompressInit (JPEGINFO**cInfo*)

Argument

Argument	Type	Explanation
cInfo	JPEGINFO*	Pointer to JPEGINFO structure[Note]

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value None

Description Initializes the JPEG compression library and sets a compression executable status. For an explanation of how to set the members of *cInfo*, see **Section 2.5.1**.

(2) jpeg_Compress

Classification Compression (2/2)

Function name jpeg_Compress

Outline Compression, main

Syntax

```
#include “jpeg.h”
long jpeg_Compress (JPEGINFO*cInfo )
```

Argument

Argument	Type	Explanation
cInfo	JPEGINFO*	Pointer to JPEGINFO structure**Note**

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value Error code

Error code	Explanation
JPEG_OK	Normal termination
JPEG_CONT	Continue
JPEG_ERR	Abnormal termination

For details, see **Figure 2-1**.

Error details are stored in member “ErrorState” of *cInfo* when processing terminates abnormally.

Description Performs JPEG compression specified by *cInfo*.

For an explanation of how to set the members of *cInfo*, see **Section 2.5.1**.

The value stored in member “ErrorState” of *cInfo* is as follows:

Value	Meaning
0x00000001	Area specification for compressed image is undefined.
0x00000002	Sampling ratio at which compression cannot be executed is specified (if compression of a JPEG file has been executed at 4:2:2 even though link was executed with library for compression at 4:2:2 not specified).
0x00000005	The values of Tc and Tp of the DHT header are undefined.

2.7.2 Expansion

(1) jpeg_DecompressInit

Classification Expansion (1/2)

Function name jpeg_DecompressInit

Outline Initializes JPEG expansion library.

Syntax #include “jpeg.h”
void jpeg_DecompressInit (JPEGINFO**dInfo*)

Argument

Argument	Type	Explanation
dInfo	JPEGINFO*	Pointer to JPEGINFO structure**Note**

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value None

Description Initializes the JPEG expansion library and sets the expansion executable status. For an explanation of how to set the members of *dInfo*, see **Section 2.5.1**.

(2) jpeg_Decompress

Classification Expansion (2/2)

Function name jpeg_Decompress

Outline Expansion, main

Syntax #include “jpeg.h”
long jpeg_Decompress (JPEGINFO**dInfo*)

Argument

Argument	Type	Explanation
dInfo	JPEGINFO*	Pointer to JPEGINFO structure**Note**

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value Error code

Error code	Explanation
JPEG_OK	Normal termination
JPEG_CONT	Continue
JPEG_ERR	Abnormal termination

For details, see **Figure 2-3**.
Error details are stored in member “ErrorState” of *dInfo* when processing terminates abnormally.

Description Executes the JPEG expansion specified by *dInfo*.

For an explanation of how to set the members of *dInfo*, see **Section 2.5.1**.

The value stored in member “ErrorState” of *dInfo* is as follows:

Value	Meaning
0x00000001	Area specification for expanded image is undefined.
0x00000002	Sampling ratio at which expansion cannot be executed has been specified (if expansion of a JPEG file has been executed at 4:2:2 even though link was executed with expansion library at 4:2:2 not specified).
0x00000003	Value of Pq of DQT header is not set to 0.
0x00000004	Quantization table number (Tp) of DQT header is other than 0, 1, 2, or 3.
0x00000005	Values of Tc and Tp of DHT header are undefined.
0x00000006	Number of components of SOS header is other than 3.
0x00000007	Huffman table number specified by SOS header is incorrect.
0x00000008	Value of Ss of SOS header is other than 0.
0x00000009	Value of Se of SOS header is other than 63.
0x0000000A	Values of Ah and Al of SOS header are other than 0.
0x0000000B	Value other than 8 is set for P of SOF header.
0x0000000C	Value of Nf of SOF header is too great.
0x0000000D	Unknown marker appears.
0x0000000E	RSTn marker is illegal.

2.7.3 Analysis

(1) jpeg_AnalysisInit

Classification Analysis (1/2)

Function name jpeg_AnalysisInit

Outline Initializes JPEG analysis library.

Syntax #include “jpeg.h”
void jpeg_AnalysisInit (JPEGINFO**aInfo*)

Argument

Argument	Type	Explanation
aInfo	JPEGINFO*	Pointer to JPEGINFO structure[Note]

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value None

Description Initializes the JPEG analysis library and sets an analysis executable status.
For an explanation of how to set the members of *aInfo*, see **Section 2.5.1**.

(2) jpeg_Analysis

Classification Analysis (2/2)

Function name jpeg_Analysis

Outline Analysis, main

Syntax #include “jpeg.h”
long jpeg_Analysis (JPEGINFO**aInfo*)

Argument

Argument	Type	Explanation
aInfo	JPEGINFO*	Pointer to JPEGINFO structure[Note]

Note For details of the JPEGINFO structure, see **Section 2.5.1**.

Return value Error code

Error code	Explanation
JPEG_OK	Normal termination
JPEG_CONT	Continue
JPEG_ERR	Abnormal termination

For details, see **Figure 2-5**.
Error details are stored in member “ErrorState” of *aInfo* when processing terminates abnormally.

Description Executes the JPEG analysis specified by *aInfo*.

For an explanation of how to set the members of *aInfo*, see **Section 2.5.1**.

The value stored in member “ErrorState” of *aInfo* is as follows:

Value	Meaning
0x00000006	Number of components of SOS header is other than 3.
0x00000007	Huffman table number specified by SOS header is incorrect.
0x00000008	Value of Ss of SOS header is other than 0.
0x00000009	Value of Se of SOS header is other than 63.
0x0000000A	Values of Ah and Al of SOS header are other than 0.
0x0000000B	Value other than 8 is set in P of SOF header.
0x0000000C	Value of Nf of SOF header is too great.
0x0000000D	Unknown marker has appeared.

2.8 DETAILS OF SECTION

Table 2-28 lists the sections defined (used) by the library.

Table 2-28. Sections Used by Library

Classification	Section name	Type	Explanation
Compression	.JPCTEXT	text	Text for compression
	.JPCTBL	rodata	Table data for compression
	.JPCDATA	data	Data having initial value for compression
	.JPCBSS	bss	Data having no initial value for compression
Expansion	.JPDTEXT	text	Text for expansion
	.JPDTBL	rodata	Table data for expansion
	.JPDDATA	data	Data having initial value for expansion
	.JPDBSS	bss	Data having no initial value for expansion
Analysis	.JPATEXT	text	Text for analysis
	.JPATBL	rodata	Table data for analysis
	.JPADATA	data	Data having initial value for analysis
	.JPABSS	bss	Data having no initial value for analysis
Common processing	.JPJTEXT	text	Text for common processing
	.JPJTBL	rodata	Table data for common processing
	.JPJDATA	data	Data having initial value for common processing
	.JPJBSS	bss	Data having no initial value for common processing

2.9 SELECTING A LIBRARY

Table 2-29 lists those files available as libraries.

Table 2-29. File Names of Libraries

Classification	File name	Explanation	Select
Compression	libjpegc.a	Compression base	**Note 1**
	libjcs11.a	Sampling ratio 4:4:4 [H:V = 1:1] supported	**Note 2**
	libjcs21.a	Sampling ratio 4:2:2 [H:V = 2:1] supported	
	libjcs22.a	Sampling ratio 4:1:1 [H:V = 2:2] supported	
	libjcs41.a	Sampling ratio 4:1:1 [H:V = 4:1] supported	
	libjcy.a	YCbCr supported	**Note 3**
	libjcr.a	RGB supported	
Expansion	libjpegd.a	Expansion base	**Note 4**
	libjds11.a	Sampling ratio 4:4:4 [H:V = 1:1] supported	**Note 2**
	libjds21.a	Sampling ratio 4:2:2 [H:V = 2:1] supported	
	libjds22.a	Sampling ratio 4:1:1 [H:V = 2:2] supported	
	libjds41.a	Sampling ratio 4:1:1 [H:V = 4:1] supported	
	libjdhb.a	Huffman “branch mode” supported	**Note 5**
	libjdhl.a	Huffman “loop mode” supported	
	libjdy.a	YCbCr supported	**Note 6**
	libjdr.a	RGB supported	
Analysis	libjpega.a	Analysis base	**Note 7**
Common processing	libjpeg.a	Common processing	**Note 8**

Notes 1. This must be selected when using the compression library.

2. Select a supported sampling ratio.

3. Either of these must be selected when using the default “jpeg_getMCU_*xx*” library.

4. This must be selected when using the expansion library.

5. One of these must be selected.

6. Either of these must be selected when using the default “jpeg_putMCU_*xx*” library.

7. This must be selected when using the analysis library.

8. This must be selected when using the JPEG library.

2.9.1 Selecting a Library During Link

The following three items allow the user to select a library during link.

- Non-link of unnecessary object
- Selection of YCbCr/RGB of VRAM
- Selection of Huffman expansion routine to be used

Use the following command to select a library:

jpegarce.exe:	for DOS
jpegarc:	for Sun4™

Caution For DOS, execute the command from the command line, not from within Windows.

When this function is executed, file “archive” is created. If a file having the same name already exists, that file is overwritten. This file is in the make file and is referenced during linking.

(1) Non-link of unnecessary object

When the command that creates file “archive” is executed, the following messages are displayed. Unnecessary objects are not linked if you enter data in response to these messages in sequence.

```
Do you need JPEG compress library? (Y/N)
```

```
    .
    .
    .
```

```
Do you need 4:2:2 compress library? (Y/N)
```

```
    .
    .
    .
```

```
Do you need 4:4:4 compress library? (Y/N)
```

```
    .
    .
    .
```

For example, where only a sampling ratio of 4:2:2 is required, the archive file related to the sampling ratios is selected as follows:

Selected file
libjcs21.a, libjds21.a
File not selected
libjcs11.a, libjcs22.a, libjcs41.a,
libjds11.a, libjds22.a, libjds41.a

(2) Whether to use the default VRAM access function

The following message is displayed. Input “Y” or “N” in response.

```
Do you need default VRAM access library? (Y/N)
```

Entering “Y” causes the following message to appear. Select the necessary item.

```
Please enter VRAM type RGB or YCbCr? (Y/R)
```

If the default VRAM access function is not used, the user must create the following functions:

- Compression: jpeg_getMCU_22, jpeg_getMCU_41, jpeg_getMCU_21, jpeg_getMCU_11
- Expansion: jpeg_putMCU_22, jpeg_putMCU_41, jpeg_putMCU_21, jpeg_putMCU_11

For details, see **Section 2.10**.

(3) Selecting the Huffman expansion routine to be used

For Huffman expansion, the following message is displayed. Select “L” or “B” as appropriate.

```
Choose which huffman routine do you use, LOOP or BRANCH? (L/B)
```

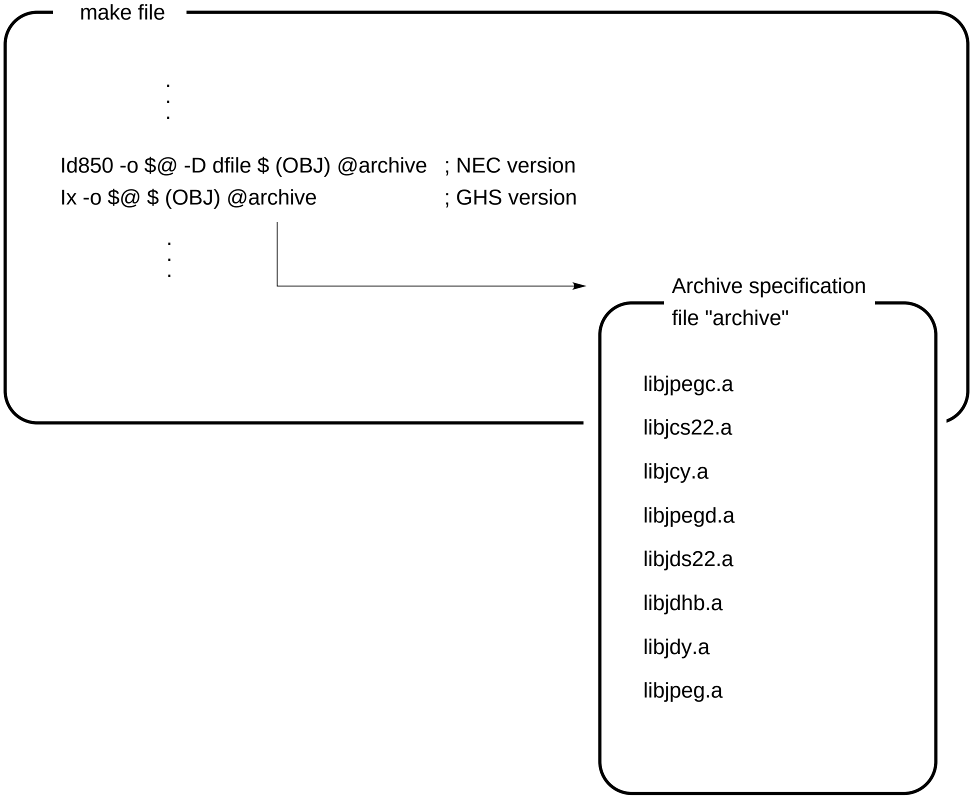
The following archive file will be selected:

- Loop: libjdhl.a
- Branch: libjdhb.a

2.9.2 Specifying an Archive File

When the command that specifies the creation of an archive file is executed, file “archive” is created. This command passes the contents of the file, in @archive format to the argument of the linker in the make file. For details of the options, refer to the manual supplied with the linker.

Figure 2-14. Specifying Archiver



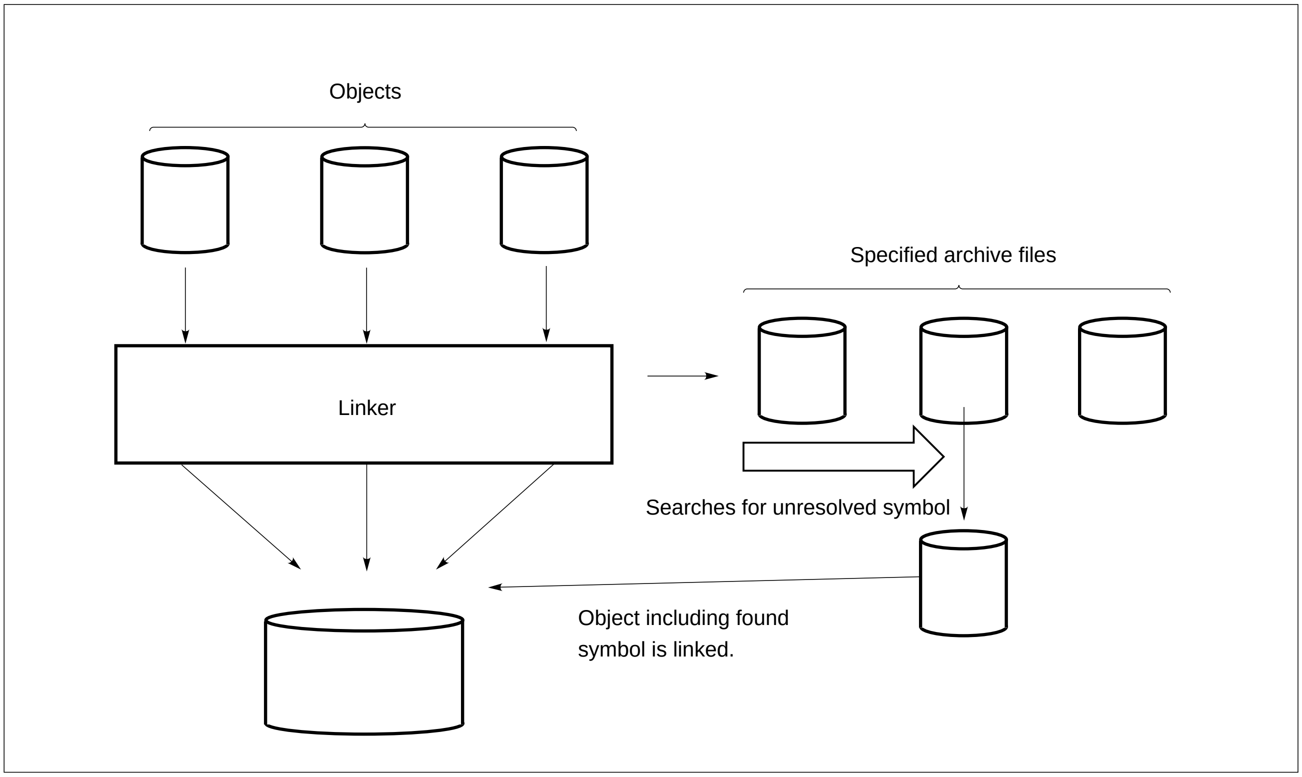
```
        .
        .
        .
ld850 -o $@ -D dfile $ (OBJ) @archive   ; NEC version
lx -o $@ $ (OBJ) @archive               ; GHS version
        .
        .
        .
```

```
libjpegc.a
libjcs22.a
libjcy.a
libjpegd.a
libjds22.a
libjdhb.a
libjdy.a
libjpeg.a
```

Archive file libjpeg.a contains a default library. Always specify libjpeg.a at the end of an archive specification file.

The linker searches for a specified archive file if an unresolved symbol exists in the object. When a symbol is found, the object file including the symbol is extracted from the archive file and linked.

Figure 2-15. Handling of Archive File by Linker



2.10 CUSTOMIZE

This section explains how the user can create VRAM access functions without using the default values.

AP703000-B03 provides getmcu.c and putmcu.c, which are used as samples when the user creates VRAM access functions.

2.10.1 VRAM Access Function Used During Compression

The functions required for VRAM access during compression are as follows:

Table 2-30. VRAM Access Functions Required During Compression

Function name	Supported sampling ratio
jpeg_getMCU_22	4:1:1 [H:V = 2:2]
jpeg_getMCU_41	4:1:1 [H:V = 4:1]
jpeg_getMCU_21	4:2:2
jpeg_getMCU_11	4:4:4

These functions read data from VRAM in MCU units, convert the format of the data into YCbCr format, and store the data into the MCU buffer in a specific format. The functions corresponding to unnecessary sampling ratios need not be created. The user may not arbitrarily specify function names. Instead, the same function names as those listed in this table must be used.

The specifications of each function are as shown below.

Function name jpeg_getMCU_xx (xx = 22, 41, 21, 11)

Outline Exchanges/transfers data from VRAM to MCU buffer.

Syntax #include “jpeg.h”
void jpeg_getMCU_22 (JPEGINFO**cInfo*)
void jpeg_getMCU_41 (JPEGINFO**cInfo*)
void jpeg_getMCU_21 (JPEGINFO**cInfo*)
void jpeg_getMCU_11 (JPEGINFO**cInfo*)

Argument First address of JPEGINFO structure

Return value None

Description Exchanges or transfers the data of the VRAM coordinates, specified by *cInfo* members “CurrentX” and “CurrentY,” to the MCU buffer specified by *cInfo* member “MCU_Buff_Bptr.”

Caution Ensure that the contents of registers r20, r21, and r29 are not lost.

The size of one MCU corresponding to each sampling ratio is as shown in Figure 2-12. If the VRAM is of RGB type each pixel, converted into YCbCr format, corresponds to the figure.

Store data in the MCU buffer in the format shown in Figure 2-11.

The following information is required by this function.

JPEGINFO->MCU_Buff_Bptr	First address of MCU buffer
JPEGINFO->CurrentX	x coordinate of corresponding MCU
JPEGINFO->CurrentY	y coordinate of corresponding MCU

If 16-bit displacement of the load/store instruction is used when this routine is described in assembler, the processing speed can be increased.

Suppose the following is executed:

```
*mcu=Y0;
*(mcu+1) = Y1;
*(mcu+2) = Y2;
*(mcu+3) = Y3;
```

At this time, the following are stored into the registers:

r7	mcu
r10	Y0
r11	Y1
r12	Y2
r13	Y3

If the following description is made, the amount of address calculation is substantially reduced, resulting in a higher processing speed.

```
st.h    r10, 0x0 [r7]
st.h    r11, 0x2 [r7]
st.h    r12, 0x4 [r7]
st.h    r13, 0x6 [r7]
```

Define the value of the symbol in an assembler file as follows (same as #define of C in the case of the GHS version).

```
VRAM_GAP1 .set 1 ··· Address difference between Y and Cb of same pixel of VRAM
VRAM_GAP2 .set 2 ··· Address difference between Y and Cr of same pixel of VRAM
```

Assuming that the address of the Y component of the pixel of VRAM to be accessed is in r8, execute the instruction as follows:

```
ld.b    0x0 [r8], r10
ld.b    VRAM_GAP1 [r8], r11
ld.b    VRAM_GAP2 [r8], r12
andi    0xFF, r10, r10
andi    0xFF, r11, r11
andi    0xFF, r12, r12
```

As a result, the following are stored into the registers.

r10	Value of Y component
r11	Value of Cb component
r12	Value of Cr component

2.10.2 VRAM Access Function Used During Expansion

The functions required for VRAM access during expansion are as follows:

Table 2-31. VRAM Access Functions Required During Expansion

Function name	Supported sampling ratio
jpeg_putMCU_22	4:1:1 [H:V = 2:2]
jpeg_putMCU_41	4:1:1 [H:V = 4:1]
jpeg_putMCU_21	4:2:2
jpeg_putMCU_11	4:4:4

These functions write YCbCr-format data, stored in the MCU buffer in a specific format, to VRAM. The functions corresponding to unnecessary sampling ratios need not be created. The user may not arbitrarily determine the function names. Instead, the same function names as those listed in this table must be used.

The specifications of each function are as shown below.

Function name jpeg_putMCU_xx (xx = 22, 41, 21, 11)

Outline Exchanges and transfers data from the MCU buffer to VRAM.

Syntax #include “jpeg.h”
void jpeg_putMCU_22 (JPEGINFO**dInfo*)
void jpeg_putMCU_41 (JPEGINFO**dInfo*)
void jpeg_putMCU_21 (JPEGINFO**dInfo*)
void jpeg_putMCU_11 (JPEGINFO**dInfo*)

Argument	First address of JPEGINFO structure
Return value	None
Description	Exchanges the data of the MCU buffer specified in *dInfo* member “MCU_Buff_Bptr” into data for VRAM, and transfers the data to the VRAM coordinates specified by *dInfo* member “CurrentX” and “CurrentY.”
Caution	Ensure that the contents of registers r20, r21, and r29 are not lost.

The format in which data is stored into the MCU is shown in Figure 2-11.

The size of one MCU corresponding to each sampling ratio is shown in Figure 2-12. If the VRAM is of RGB type each pixel, converted into YCbCr format, corresponds to the figure.

The following information is required for this function.

JPEGINFO->MCU_Buff_Bptr	First address of MCU buffer
JPEGINFO->CurrentX	x coordinate of corresponding MCU
JPEGINFO->CurrentY	y coordinate of corresponding MCU

2.11 SYMBOL NAME CONVENTION

The names of the symbols used in the JPEG library are determined according to the following convention. When you define symbols in an application, ensure that the same symbol names are not used.

Table 2-32. Symbol Name Convention

Classification	Convention
Function	Prefixes “jpeg_”.
Variable	Prefixes “jpeg_”.
Data type	JPEGINFO APPINFO
Constant	JPEG_OK JPEG_ERR JPEG_CONT SAMPLE11 SAMPLE21 SAMPLE22 SAMPLE41