To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

User's Manual

Phase-out/Discontinued

μSAP705100-B03, μSAP70732-B03

JPEG Middleware

Applicable Devices

μSAP705100-B03: V830 family™
μSAP70732-B03: V810 family™

Document No. U11052EJ4V0UM00 (4th edition)
Date Published May 1999 J CP(K)

[MEMO]

SUMMARY OF CONTENTS

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Rodovia Presidente Dutra, Km 214
07210-902-Guarulhos-SP Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

J99.1

Major Revisions in This Edition

Page	Description
p.43	Addition of description to **Section 1.2.2 (1) (f)**
p.45	Addition of description to **Section 1.2.2 (1) (g)**
p.50	Addition of **Section 1.3.4**
p.51	Addition of **Section 1.3.5**
p.54	Correction of package contents in **Section 1.3.6 (3)**
p.56	Correction of package contents in **Section 1.3.6 (4)**
p.58	Addition of description of additional library to **Section 1.3.7 (3)**
p.60	Addition of description of additional library to **Section 1.3.8**
p.83	Addition of Note to **Table 2-6**
p.85	Addition of Note to **Table 2-7**
p.87	Addition of Note to **Table 2-8**
p.88	Addition of Note to **Table 2-9**
p.159	Addition of **Chapter 3**
p.227	Addition of **Section B.2**
p.229	Addition of **Appendix C**

The mark * shows major revised points.

[MEMO]

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 understand the functions of the μSAP705100-B03 and μSAP70732-B03.

Organization This manual includes the following:

- Overview
- Library specifications
- Source lists of sample programs

Reading this manual In this manual, the μSAP705100-B03 is referred to as the AP705100-B03.
The μSAP70732-B03 is referred to as the AP70732-B03.

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} = 1,024
M (mega) : 2^{20} = $1{,}024^2$

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 V810 family

Product name		Data sheet	User's manual	
Unofficial name	Part number		Hardware	Architecture
V821™	μPD70741	U11678E	U10077E	U10082E

Documents related to V830 family

Product name		Data sheet	User's manual	
Unofficial name	Part number		Hardware	Architecture
V830™	μPD705100	U11483E	U10064E	U12496E
V831™	μPD705101	U12979E	U12273E	
V832™	μPD705102	U13675E	U13577E	

Documents related to V810 family development tools (User's Manuals)

Document name		Document number
CA732 (C compiler)	Operation (UNIX™-based)	U11013E
	Operation (Windows™-based)	U11068E
	Assembly language	U11016E
	C	U11010E
	Project manager	U11991E
RX732 (real-time OS)	Basics	U10346E
	Technical	U10490E
	Nucleus installation	U10347E

Documents related to V830 family development tools (User's Manuals)

Document name			Document number
CA830 (C compiler)		Operation (UNIX-based)	U11013E
		Operation (Windows-based)	U11068E
		Assembly language	U11014E
		C	U11010E
		Project manager	U11991E
RX830 (real-time OS)	ITRON1	Basics	U11730E
		Installation	U11731E
		Technical	U11713E
	μITRON Ver. 3.0	Basics	U13152E
		Installation	U13151E
		Technical	U13150E

- **Documents related to tools produced by Green Hills Software™, 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

Phase-out/Discontinued

CONTENTS

LIST OF FIGURES (1/3)

LIST OF FIGURES (2/3)

LIST OF FIGURES (3/3)

LIST OF TABLES (1/3)

LIST OF TABLES (2/3)

LIST OF TABLES (3/3)

CHAPTER 1 OVERVIEW

1.1 MIDDLEWARE

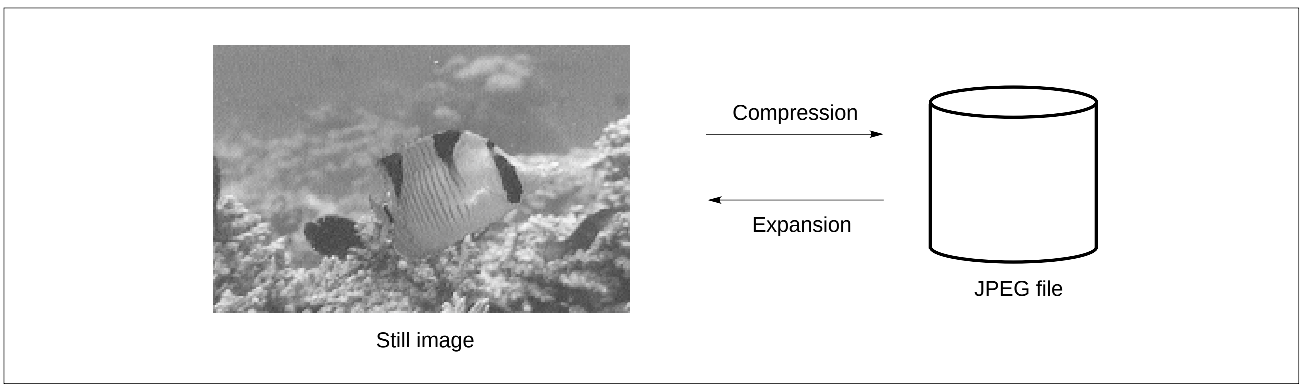
Middleware is a software group that has been tuned to fully exploit the performance of a processor. The software implements processing that is conventionally performed by hardware. The advent of high-performance RISC (reduced instruction set computer) processors has spawned the concept of middleware, 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. The AP705100-B03 and AP70732-B03 support the most fundamental baseline DCT with their basic library. The AP705100-B03 also supports the progressive format with its additional library (expansion function only).

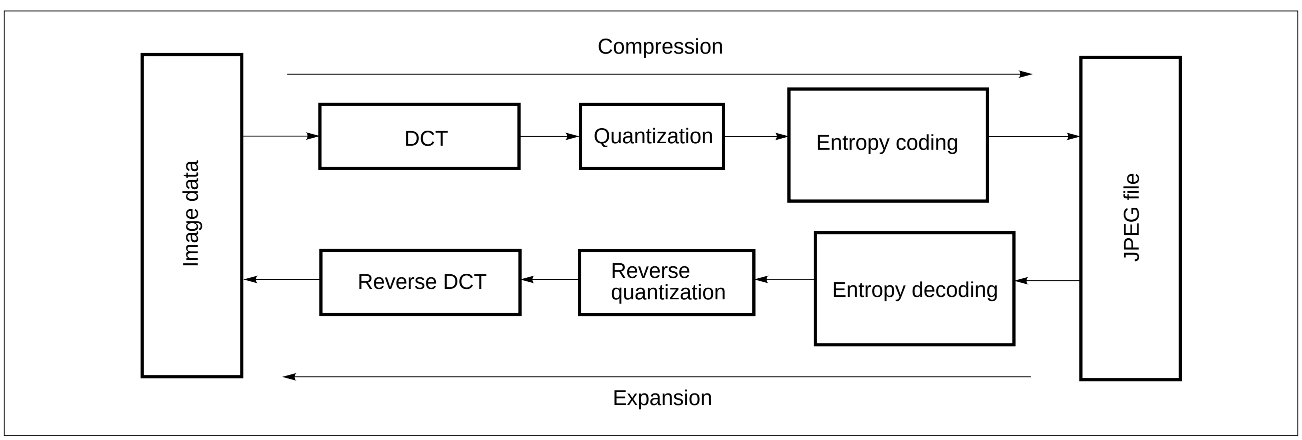
Figure 1-2. JPEG Versions

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/progressive uses a technology based on Huffman encoding.

The AP705100-B03 and AP70732-B03 perform 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

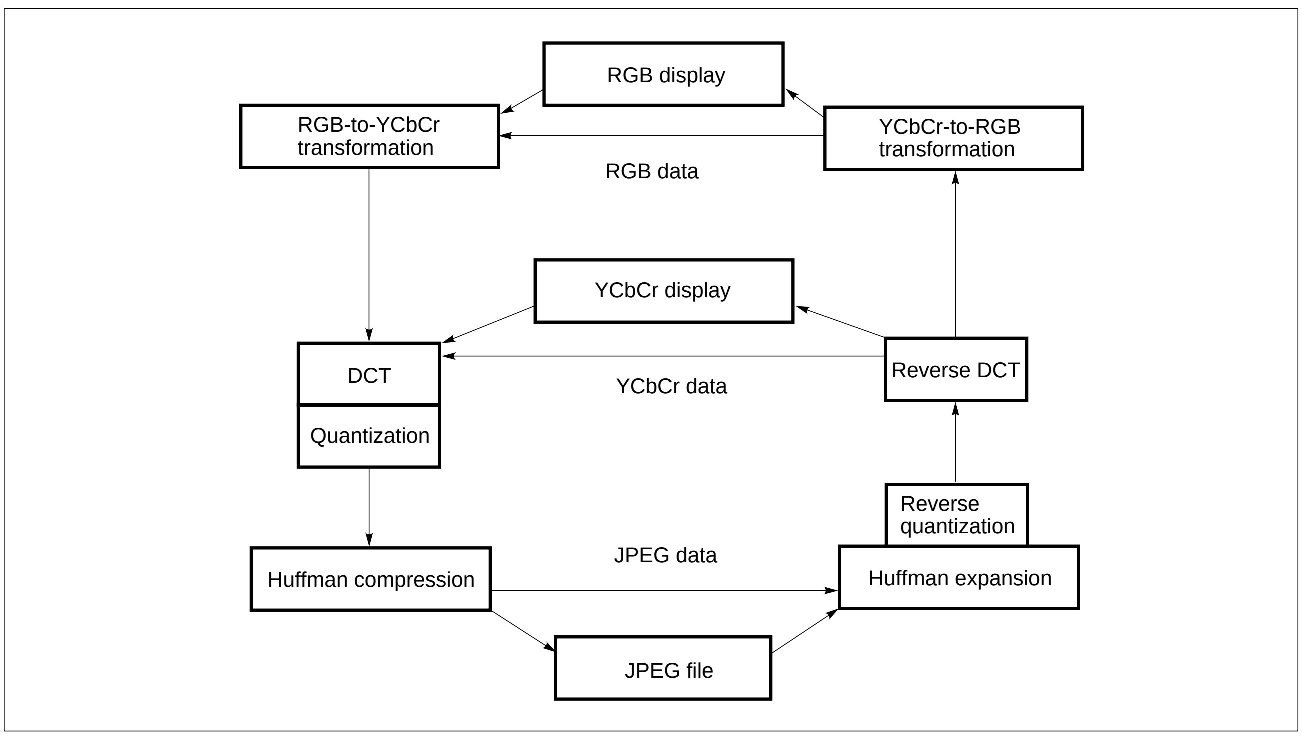
Color JPEG compresses or expands images by using three color spaces, Y, Cb, and Cr (only luminance for monochrome images). If the image data is not YCbCr but RGB, processing to transform the RGB data into YCbCr for compression, or that to transform YCbCr data into RGB before displaying the result of 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 2:1:1. When obtaining one block each of Y, Cb, and Cr from one MCU, the sampling ratio is 1:1:1.

Table 1-1. Sampling Ratio and MCU

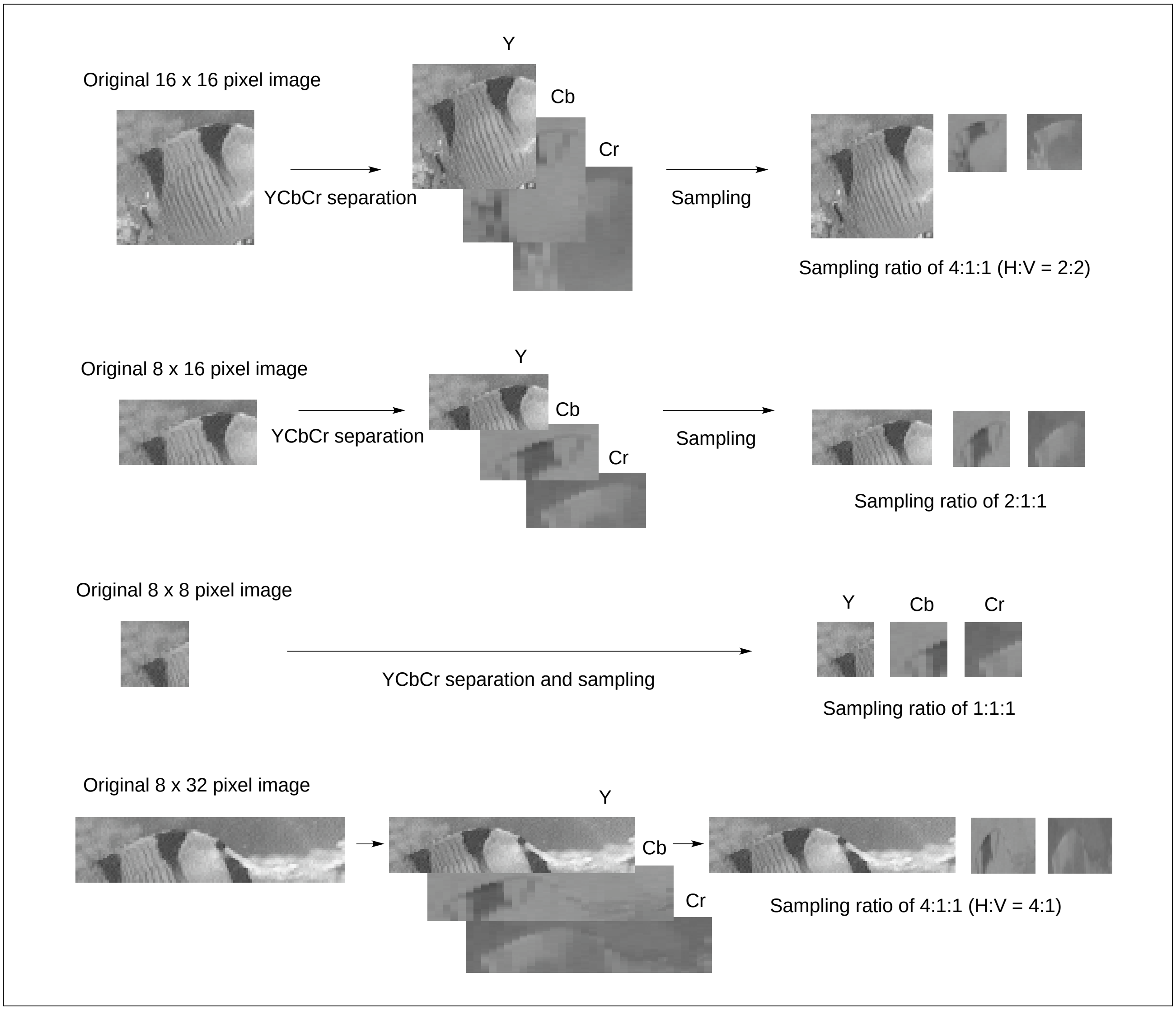
MCU	Sampling ratio	Block
Vertical 16 pixels Horizontal 16 pixels	4:1:1 (H:V = 2:2)	Y: 4 blocks Cb: 1 block, Cr: 1 block
Vertical 8 pixels Horizontal 32 pixels	4:1:1 (H:V = 4:1)	Y: 4 blocks Cb: 1 block, Cr: 1 block
Vertical 8 pixels Horizontal 16 pixels	2:1:1	Y: 2 blocks Cb: 1 block, Cr: 1 block
Vertical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

Remark H: Horizontal sampling ratio of MCU
V: Vertical sampling ratio of MCU

Although sampling ratios not listed in Table 1-1 are supported by the JPEG standard, only the sampling ratios in this table are supported by the basic library of the AP705100-B03 and AP70732-B03. The additional library of the AP705100-B03 supports all sampling ratios.

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



With JPEG compression, a sampling ratio of 4:1:1 is used more often than 1:1:1.
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 the Cr component. Consequently, 7,200 blocks (= 40 x 30 x 6) are obtained from the entire image. To these 7,200 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
2:1:1	40 segments	60 segments	4	9600
1:1:1	80 segments	60 segments	3	14400

Remark H: Horizontal sampling ratio of MCU
V: Vertical sampling ratio of MCU

As is evident from the above table, more blocks are needed at a sampling ratio of 1:1:1 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

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

$$C(w) = \frac{1}{\sqrt{2}} \quad (w = 0)$$

$$= 1 \quad (w \neq 0)$$

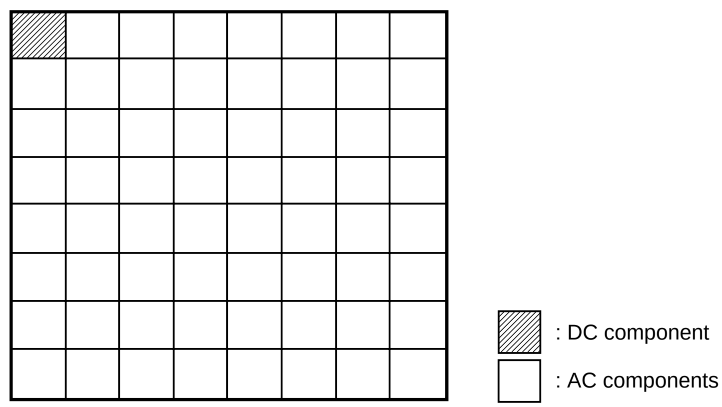
Generally, this DCT is applied to 8 x 8 elements with signal processing techniques such as JPEG and MPEG.

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.

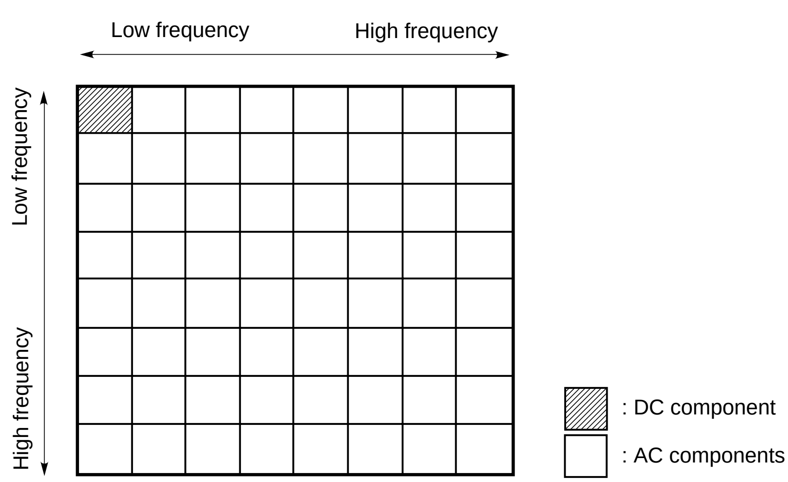
The 64 elements obtained as a result of DCT conversion of 8 x 8 pixel image data are called a DCT coefficient. The first element indicates the average color level of the entire matrix, while the other 63 elements indicate the level of distortion of the color in the matrix. Because of the difference in 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

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

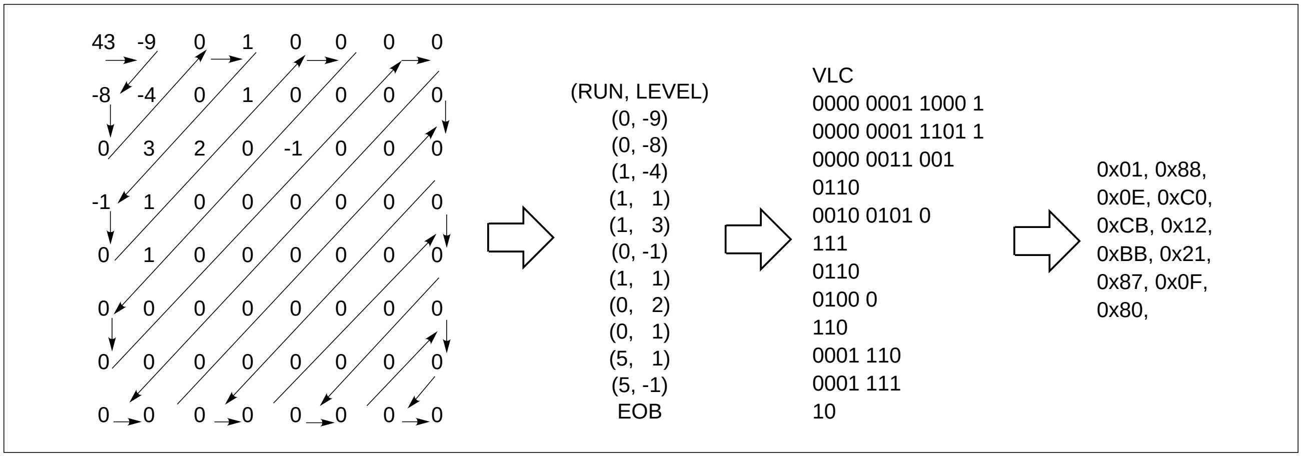
Example of quantized 8 x 8 matrix

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

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



(6) Entropy encoding

Generally, JPEG performs entropy encoding using Huffman coding. 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 components, 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
-1,023 to -512, 512 to 1,023	10
-2,047 to -1,024, 1,024 to 2,047	11

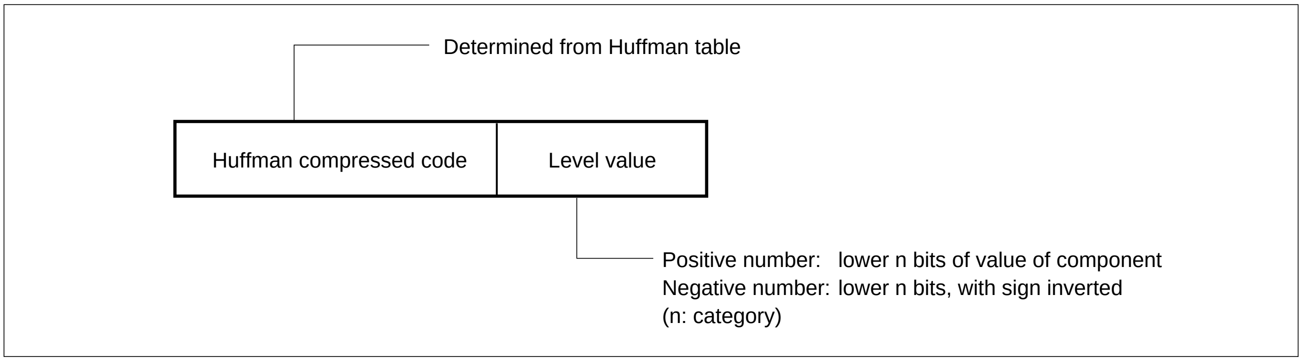
In JPEG compression, entropy compression of the values in this category is performed.
For example, suppose the Huffman table for the DC component for luminance (Y) follows the convention shown below:

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

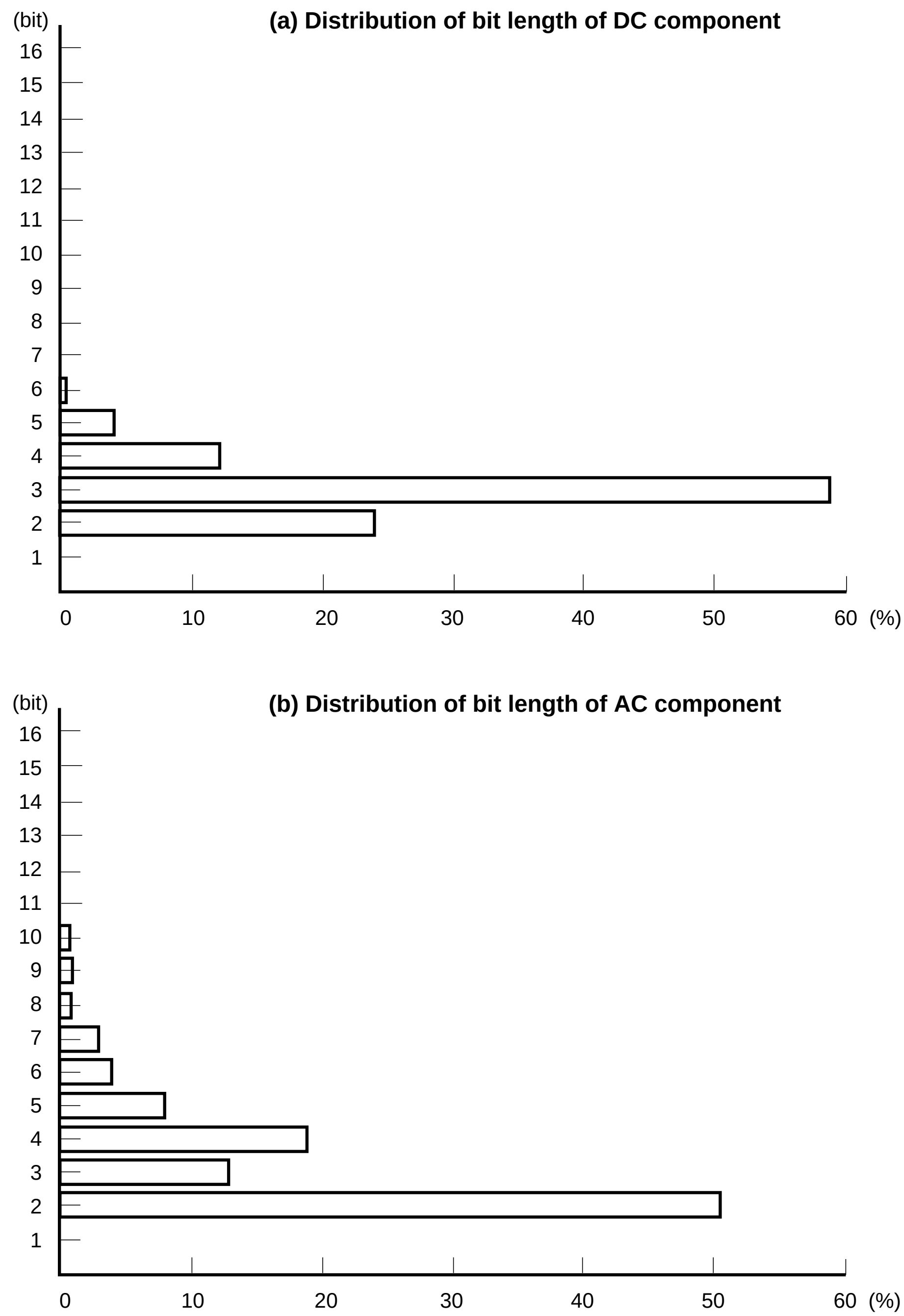


For the AC components, the Huffman table follows the convention shown below:

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 Components



(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

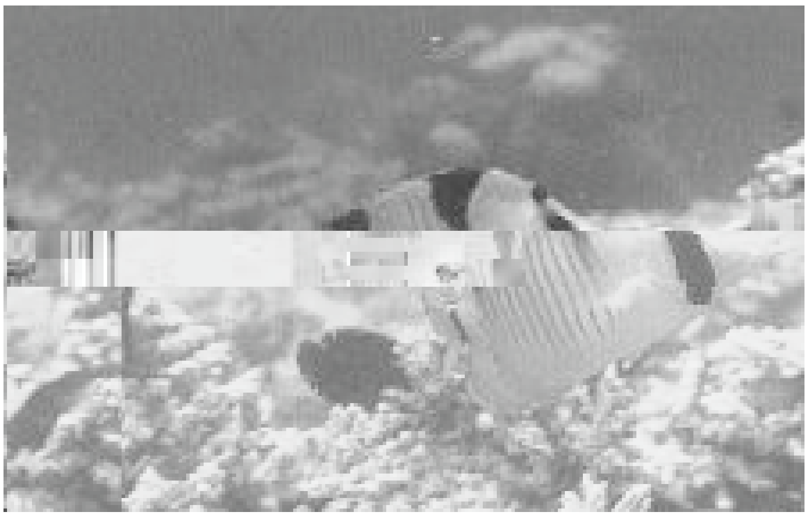
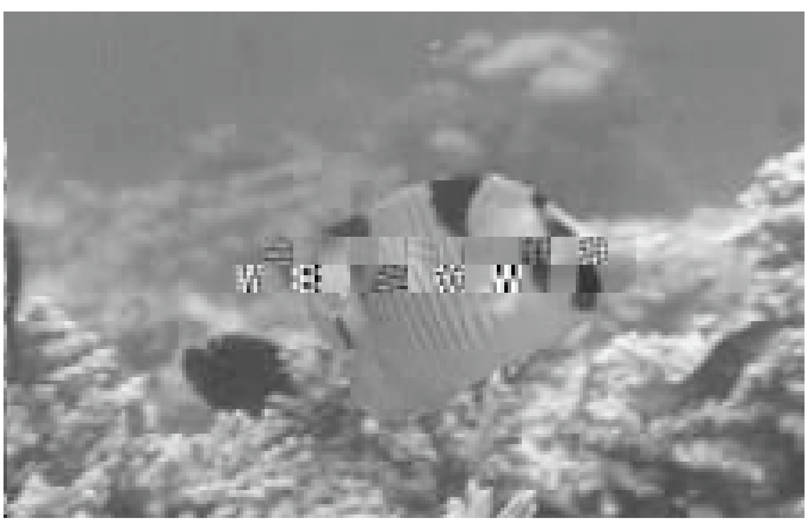
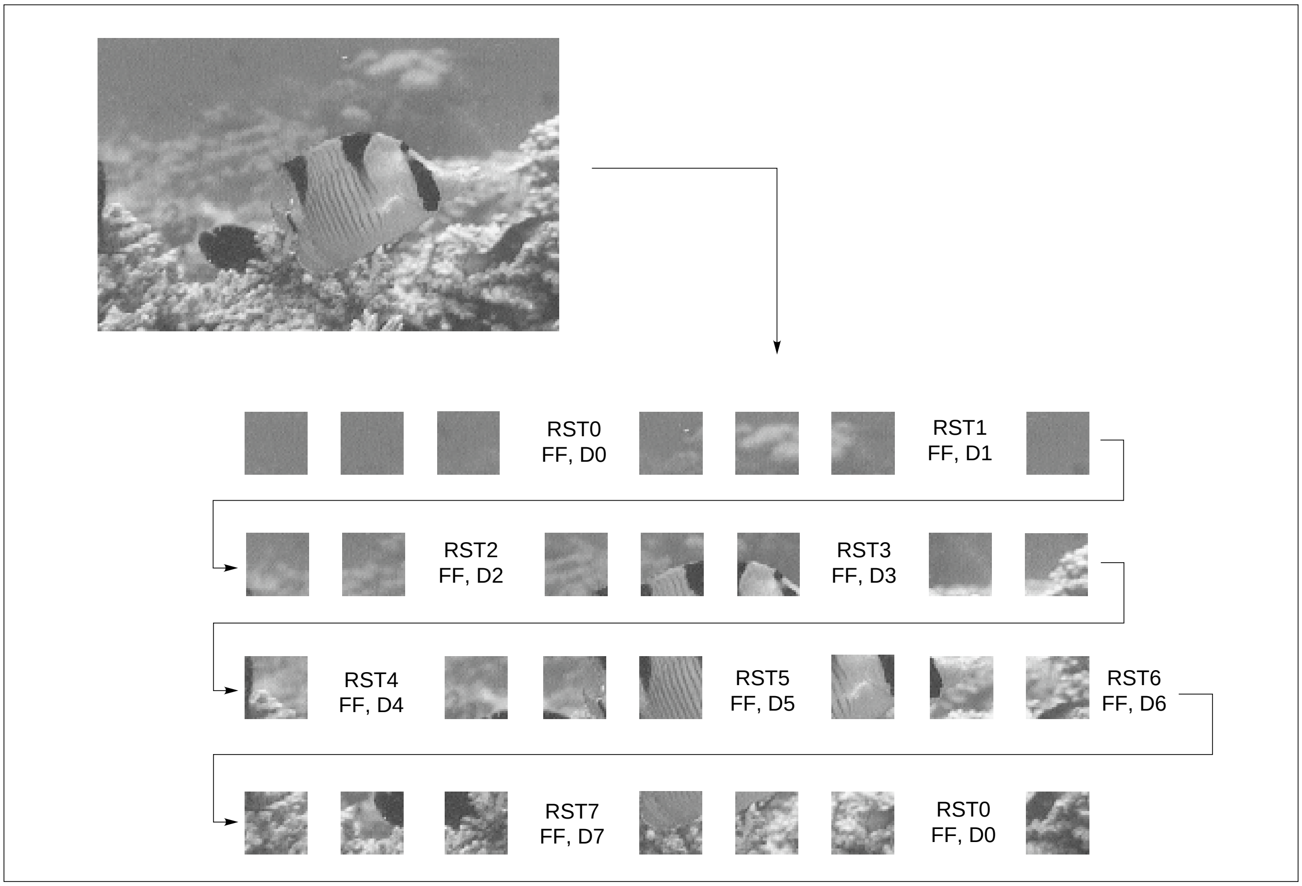


Figure 1-13. Correct Expansion Can Be Performed Due to Use of Restart Markers



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



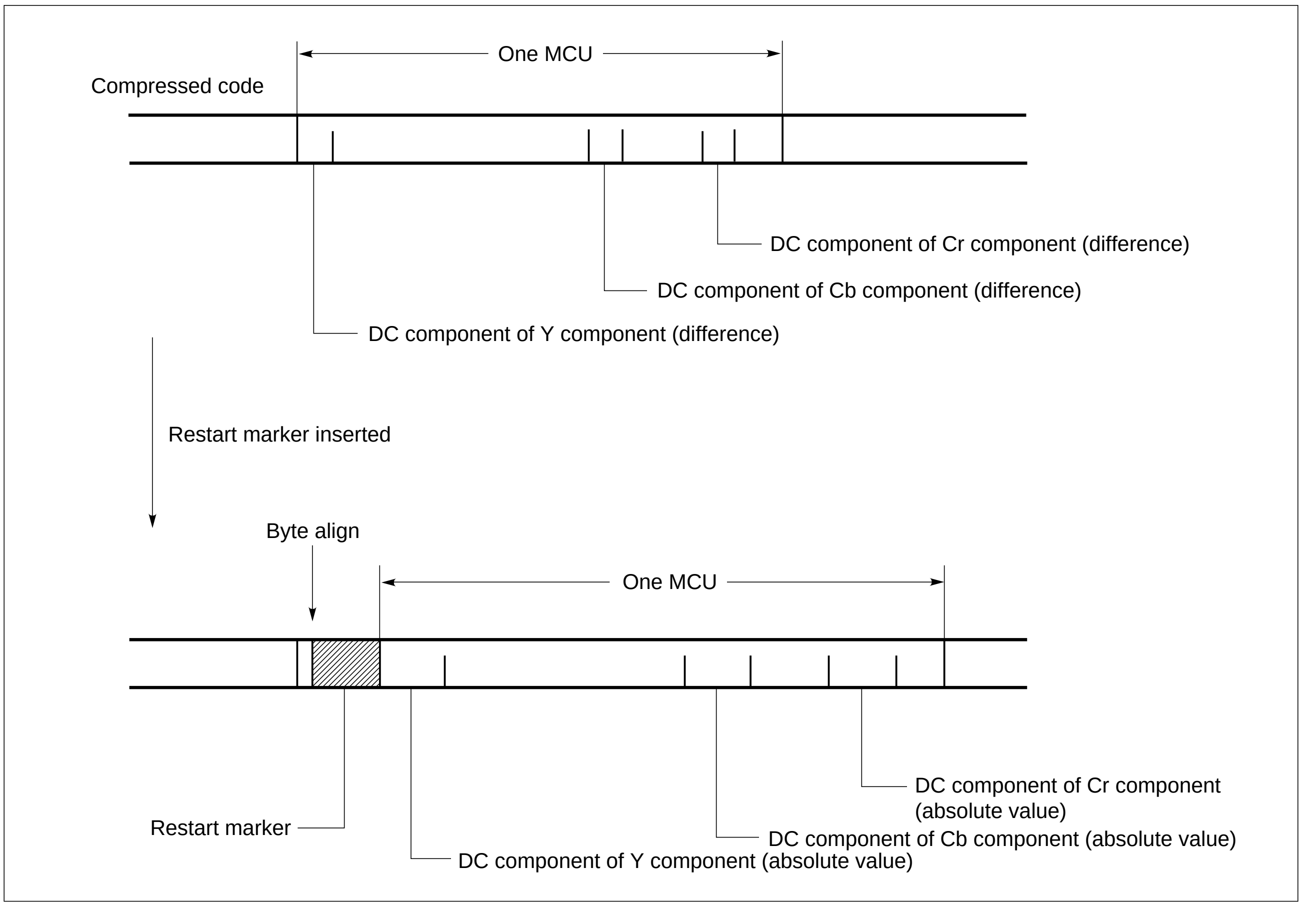
The number of restart markers to be inserted is determined by the size of the image. For example, the number of restart markers for an image measuring 640 x 480 pixels, for a sampling ratio of 2:1:1 and a restart interval of 2, is calculated as follows:

MCU (minimum compression unit): 16 x 8 pixels
Restart marker: every 2 MCUs
Therefore,
(640 x 480) / (16 x 8) / 2 = 1,200 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 2:1:1 and the restart interval is 2, increases by about 4,800 bytes (1,200 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



(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 65,533 bytes)

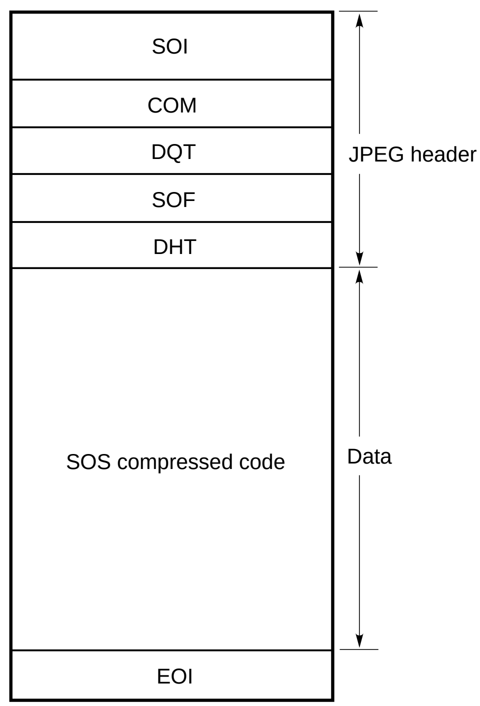
There are 16 types of APPn markers, from 0xFF,0xE0 to 0xFF,0xEF, each corresponding to an APPn segment.

The AP705100-B03 and AP70732-B03 determine 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."

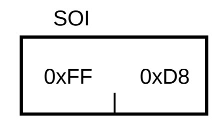
The sequence of each segment (such as COM, DQT, SOF, and DHT) of the JPEG header is arbitrary. The following table lists the JPEG markers.

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

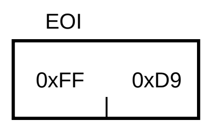
Figure 1-18. SOI Marker



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

Figure 1-19. EOI Marker

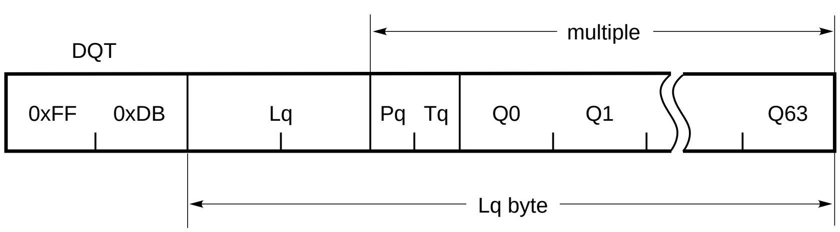


This marker indicates the end of a JPEG file. A JPEG file always ends with this 2-byte marker.

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

This marker defines a quantization table.

Figure 1-20. DQT Segment



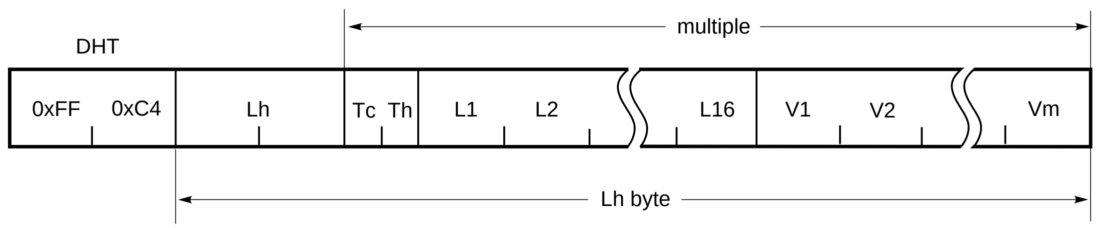
Lq: length of DQT segment
Pq: Precision of quantization factors (Q0 to Q63)
0 indicates 8-bit precision, while 1 indicates 12-bit precision (fixed to 0 on base line).
Tq: quantization table number (0, 1, 2, 3)
Q0 to Q63: quantization factor (1 to 255)
1 to 255 where Pq = 0.
1 to 65,535 where Pq = 1.

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.

Figure 1-21. DHT Segment



Lh: length of DHT segment
Tc: 0 = DC, 1 = AC
Th: Huffman table number
0 or 1 on base line
0, 1, 2, or 3 on extended base line and progressive
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

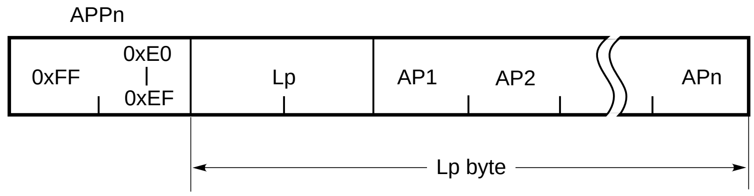
If L1 through L16 are as shown above, the meaning is as follows:

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

Figure 1-22. APPn Segment



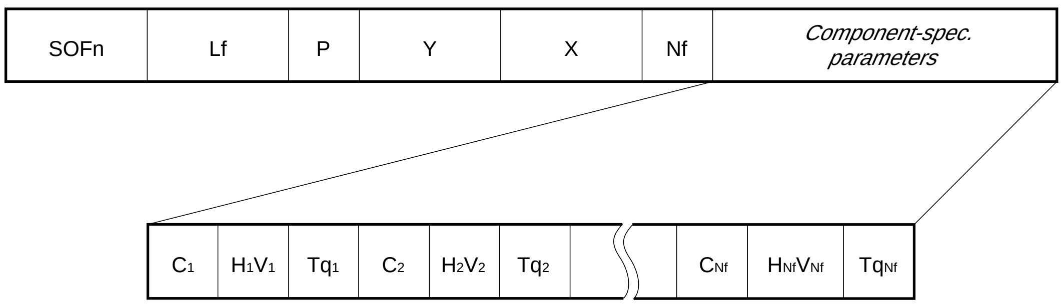
The application data segment is a segment that can be freely used by each application. Usually, this segment contains the version of the application that created a JPEG file. In some cases, a small JPEG file is contained as is. This segment can be skipped by only referring to the value of Lp.

* **(f) SOFn (Start of frame) marker**

In JPEG, the portion of a JPEG file with the SOI and EOI markers excluded is called a frame. An SOFn segment is also called a frame header and specifies the quantization table number needed for expansion.

In JPEG, color elements, such as the Y, Cb, and Cr, is called components.

Figure 1-23. SOFn Segment



SOFn	SOF0	0xFF 0xC0: Base line
	SOF1	0xFF 0xC1: Extended base line
	SOF2	0xFF 0xC2: Progressive
Lf		Length of SOFn segment (8 + 3 x Nf byte)
P		Precision of DCT coefficient 8: 8-bit 12: 12-bit
Y		Vertical size of image (number of pixels)
X		Horizontal size of image (number of pixels)
Nf		Number of components 1: Monochrome (single color) 3: Color (3 colors) 4: CMYK, YCCK (4 colors)
Ci ($0 \leq Ci \leq 255$)		Color component number definition (number of ith component)
Hi ($1 \leq Hi \leq 4$)		Horizontal sampling ratio (horizontal sampling ratio of ith component)
Vi ($1 \leq Vi \leq 4$)		Vertical sampling ratio (vertical sampling ratio of ith component)
Tqi ($0 \leq Tqi \leq 3$)		Quantization table number (quantization table number used for ith component)

Example: Suppose the contents of the SOFn segment are as follows:

SOFn		Lf		P	Y		X		Nf	
0xFF	0xC0	0x00	0x11	0x08	0x00	0x90	0x00	0xE0	0x03	*Component-spec. parameters*

C1	H1V1	Tq1	C2	H2V2	Tq2	C3	H3V3	Tq3
0x01	0x22	0x00	0x02	0x11	0x01	0x03	0x11	0x01

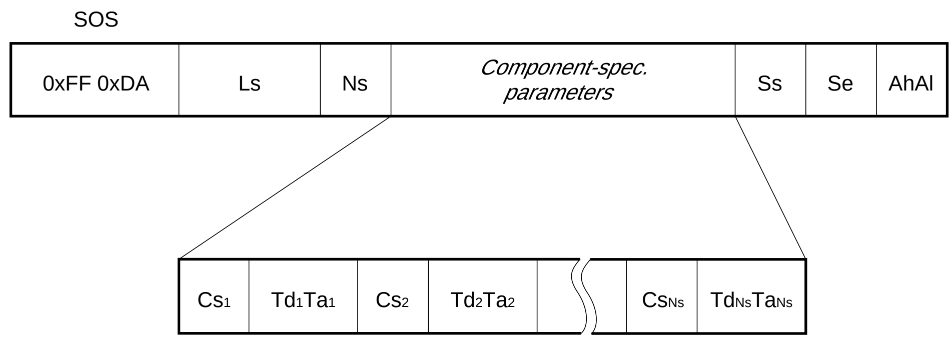
At this time, the meanings of Nf, Ci, Hi, and Vi are as follows and the sampling ratio is 4:1:1 (H:V = 2:2).

The number of components (Nf) is 3 (YCbCr).
The color component number of the Y component (C1) is 1 and the sampling ratio (H1V1) is 2 x 2 or 4 blocks.
The color component number of the Cb component (C2) is 2 and the sampling ratio (H2V2) is 1 x 1 or 1 block.
The color component number of the Cr component (C3) is 3 and the sampling ratio (H3V3) is 1 x 1 or 1 block.

* **(g) SOS (Start of scan) marker**

The compressed data of a JPEG file is divided into units called “scans” that start from an SOS segment. Therefore, the SOS segment is also called a scan header. Compressed image data starts immediately after the scan header. The scan header specifies a Huffman table number for compressed data.

Figure 1-24. SOS Segment



SOS	0xFF 0xDA
Ls	Length of SOS segment (6 + 2 x Ns byte)
Ns	Number of colors included in the scan 3: Y block, Cb block, and Cr block 1: Any of Y, Cb, and Cr
Csi	Number of ith component (number defined by SOF)
Tdi	Huffman table number of DC component of ith component Base line: 0 or 1 Extended base line and progressive: 0, 1, 2, 3
Tai	Huffman table number of AC component of ith component Base line: 0 or 1 Extended base line and progressive: 0, 1, 2, 3
Ss	Start number of DCT coefficient Base line and extended base line: Fixed to 0 Progressive: 0 to 63
Se	End number of DCT coefficient Base line and extended base line: Fixed to 63 Progressive: Ss to 63 (However, fixed to Se = 0 where Ss = 0.)
Ah	High-order bits of DCT coefficient Base line or extended base line: Fixed to 0 Progressive: 0 to 13
Al	Low-order bits of DCT coefficient Base line or extended base line: Fixed to 0 Progressive: 0 to 13

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

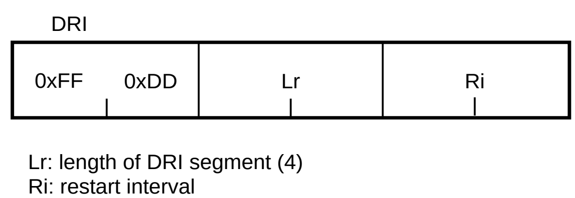
A restart marker is used to minimize the influence of illegal data such as a communication error. A restart marker is inserted every number of MCUs, as set by the DRI marker.

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 component is differential information with JPEG, the preceding DC component is required to expand an 8 x 8 pixel block. The DC component 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.

Figure 1-25. DRI Segment



1.3 OUTLINE OF SYSTEM

1.3.1 Library Configuration

The JPEG middleware library consists of the following libraries:

Table 1-5. Library Configuration of Product

Product	Library Configuration	Supported File Format
AP70732-B03 (supports V810 family)	Basic library (compression)	Base line
	Basic library (expansion)	
AP705100-B03 (supports V830 family)	Basic library (compression)	Base line
	Basic library (expansion)	
	Additional library (expansion)	Progressive Base line Extended base line

The AP705100-B03 has a basic library and an additional library. The additional library can be used either by itself or together with the basic library. The basic library is for high-speed processing and is of small memory type, while the additional library emphasizes functions and supports a range of formats.

In this User's Manual, the expansion processing of the basic library is called basic expansion, while the expansion processing of the additional library is called additional expansion.

1.3.2 Features of Basic and Additional Libraries

The features common to the basic and additional libraries are as follows:

(1) VRAM and coordinates (x, y)

The libraries do not include a VRAM access function. This function is hardware-dependent such that the user must, therefore, describe the VRAM access function according to the system (C can be used).

For VRAM that can be accessed by an LD.B or ST.B instruction, however, default routines are provided as libraries.

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.

(2) Quantization table

Specify two sides of the quantization table that can be set.

A default quantization table is provided for compression, but a user-defined quantization table can also be used. A quantization parameter (Quality) is also provided. Setting this parameter to a value of between 0 and 100 causes all values in the quantization tables to be multiplied by a constant, such that the values of the elements will be within the range of 1 to 255. (To use the quantization tables as is, specify 50 for the quantization parameter.)

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

(3) Huffman table

Specify four sides of the Huffman table that 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.

(4) 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.

(5) APPn segment

The insertion of an APPn segment can be specified for compression.
Although APPn segments are ignored during expansion, their locations can be detected.

(6) OS support

The compression, analysis, and expansion routines are reenterable. To execute a routine, specified structures are required.

1.3.3 Features of Basic Library

The features of the basic library are as follows:

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

(2) Buffer used to store JPEG files

When executing a routine, a buffer is required to store the JPEG files. Because the file size varies with each JPEG file, processing is stopped when the data reaches the end of the JPEG buffer, then restarted by re-calling the routine after buffer processing (saving the contents of the buffer during compression, or updating the buffer during expansion).

The size of the JPEG file buffer can be set to any value of 1 byte or greater. However, because register dispatch is always performed between compression or expansion and buffer processing, allocate sufficient memory to prevent register dispatch from being performed at excessively short intervals.

(3) Compression and expansion

In addition to expansion to normal size (the number of pixels written to the JPEG file header), an expansion mode reduced to 1/4, 1/16, or 1/64 of the area can be selected (in this reduced expansion mode, expansion can be implemented at high speed relative to expansion to normal size because the reverse DCT transformation is designed for reduction).

(4) Clipping of expansion (MCU unit)

When expanding the JPEG file, a rectangle can be clipped in MCU units and only the clipped part expanded.

(5) Line processing

It is possible to stop processing per line processing of the number of vertical pixels of 1 MCU in a system whose image memory cannot store the entire image.

(6) Internal RAM (AP705100-B03)

No internal instruction RAM is used.

An internal data RAM of 1,024 bytes is used.

To obtain sufficient performance, 1,024 bytes are necessary for each compression/expansion task.

★ 1.3.4 Features of Additional Library (AP705100-B03)

(1) Supports three file formats

The following three file formats can be expanded.

- Progressive file format
- Baseline file format
- Extended baseline file format

(2) Sampling ratio

All sampling ratios are supported.

(3) Support of non-interleave format

The compressed data of a JPEG file is divided into units starting from an SOS segment called a scan. The ordinary baseline format is an interleave format (single-scan format) in which only one scan exists in a file. The additional library also supports a non-interleave format (multiscan format) that has multiple scans, as well as the progressive format.

(4) Color components

In addition to the three-color formats, a monochrome format and a four-color format are also supported.

(5) DNL marker

A DNL marker that defines the number of lines is supported.

(6) Clipping of expansion (in pixel units)

Clipping can be performed in pixel units.

(7) Zooming out/in during expansion

Zooming out or in can be carried out at a ratio of n/8 (n = 1, 2, 3, ...) during expansion.

(8) JPEG file storage buffer

The additional library is not terminated even when the JPEG buffer runs short. Instead, a user-defined overwrite function (JPEG file acquisition function) is called.

(9) Memory size

Depending on the library option selected for execution, execution can be performed with a ROM/RAM size (small memory size) that closely approximates that of the basic library.

(10) Discontinuation of a library

The execution of library can be abandoned prior to its completion.

(11) Supports ISO/IEC 10918-2 Adaptive Test

The AP705100-B03 additional library conducts an adaptive test on image data, and normal expansion is confirmed with test data A, C, E, G, and K.

* 1.3.5 Differences between Basic Library and Additional Library (AP705100-B03)

Table 1-6 shows the differences between the basic library and additional library of the AP705100-B03.

Table 1-6. Differences between Basic Library and Additional Library

Item \ Library			Basic library	Additional library
File format	Encoding mode	Base line	o	o
		Extended base line	x	o
		Progressive	x	o
		Others	x	x
	Encoding sequence	Interleave	o	o
		Non-interleave	x	o
	DNL marker support		x	o
	Sampling ratio		4:1:1 (H:V = 2:2) 4:1:1 (H:V = 4:1) 2:1:1, 1:1:1	All sampling ratios supported
	Color components		Three colors	Monochrome, three colors, four colors
Interface library	Processing speed		Fast	Slow
	Support if JPEG file storage buffer runs short		Terminates library (library must be called again for resumption)	Calls JPEG buffer replenishing function
	Clipping		MCU units	Pixel units
	Zooming out/in during expansion		Zooming out/in of 1/4, 1/16, or 1/64	Zooming out/in at ratio of n/8

Remark o: Supported
x: Not supported

To expand a file in the same baseline format, the additional library is slower than the basic library because the additional library is intended for general purpose use.

When executing compression or expansion, the basic library's processing speed for 1/4 expansion is faster than expansion at a multiple of 1, while 1/16 expansion is faster than 1/4 expansion, because the basic library calculates only the pixels for the area to be expanded. In contrast, the additional library first develops an image in memory before expansion or compression. Consequently, the expansion of an area ratio at a multiple of other than 1 is slower than the speed of expansion at a multiple of 1.

In addition, how options are to be set differs between the basic library and additional library. When setting an option, refer to the description of how to set the option in question.

1.3.6 Package Contents

The package includes the following libraries and sample source.

(1) AP70732-B03 (NEC version)

Directory	File	Description
nectools — lib732	libjpegc.a	:(Compression main)
	libjcmc1.a	:(Compression mcu control) (screen units)
	libjcmc2.a	:(Compression mcu control) (MCU line units)
	libjcy.a	:(Compression YCbCr)
	libjcr.a	:(Compression RGB)
	libjpegd.a	:(Expansion main)
	libjdmc1.a	:(Expansion mcu control) (screen units)
	libjdmc2.a	:(Expansion mcu control) (MCU line units)
	libjdy.a	:(Expansion YCbCr)
	libjdr.a	:(Expansion RGB)
	libjpeg.a	:(Common)
	libjcr2.a	
nectools — smp732 — jpeg	start.s	:Startup
	jpeg.h	:Header file
	main.c	:Sample main
	fish.s	:Sample JPEG file
	fish.jpg	:Included in fish.s
	fishtga.s	:Sample image file
	fish.tga	:Included in fishtga.s
	tpycc.c	:Sample source
	tprgb.c	:Sample source
	getmcu.c	:C getmcu sample
	putmcu.c	:C putmcu sample
	makeycc	:make file
	makergb	:make file
	jparc830.exe (98)	
	jparc830 (SUN4™)	
	dfile	:Link directive

(2) AP70732-B03 (GHS version)

ghstools	lib810		libjpegc.a	:(Compression main)
			libjcmc1.a	:(Compression mcu control) (screen units)
			libjcmc2.a	:(Compression mcu control) (MCU line units)
			libjcy.a	:(Compression YCbCr)
			libjcr.a	:(Compression RGB)
			libjpegd.a	:(Expansion main)
			libjdmc1.a	:(Expansion mcu control) (screen units)
			libjdmc2.a	:(Expansion mcu control) (MCU line units)
			libjdy.a	:(Expansion YCbCr)
			libjdr.a	:(Expansion RGB)
			libjpeg.a	:(Common)
			libjcr2.a	
	smp810	jpeg	start.s	:Startup
			jpeg.h	:Header file
			main.c	:Sample main
			fish.s	:Sample JPEG file
			fishtga.s	:Sample image data
			tpycc.c	:Sample source
			tprgb.c	:Sample source
			getmcu.c	:C getmcu sample
			putmcu.c	:C putmcu sample
			makeycc	:make file
			makergb	:make file
			jparc830.exe (98)	
			jparc830 (SUN4)	
			make.lnk	:Section specification

*(3) AP705100-B03 (NEC version)

Directory	File	Description
nectools — lib830	libjpegc.a	: (Basic compression) Compression main
	libjcmc1.a	: (Basic compression) MCU control; screen unit
	libjcmc2.a	: (Basic compression) MCU control; MCU line unit
	libjcy.a	: (Basic compression) getMCU; YCbCr
	libjcr.a	: (Basic compression) getMCU; RGB1
	libjcr2.a	: (Basic compression) getMCU; RGB2
	libjpegd.a	: (Basic expansion) Expansion main
	libjpgdp.a	: (Basic expansion) Expansion main
	libjdmc1.a	: (Basic expansion) MCU control; screen unit
	libjdmc2.a	: (Basic expansion) MCU control; MCU line unit
	libjdy.a	: (Basic expansion) putMCU; YCbCr
	libjdr.a	: (Basic expansion) putMCU; RGB1
	libjdr2.a	: (Basic expansion) putMCU; RGB2
	libjpeg.a	: (Basic) Common
	libjprg.a	: (Additional expansion) Progressive support
	libjprgd.a	: (Additional expansion) Debug library
	libjprog.a	: (Additional expansion) For symbol resolution
nectools — smp830 — jpeg	start.s	: Start-up routine
	dfile	: Link directive
	jparc0, jparc1, jparc2, jparc3, jparc4	Library selection file
	makefile	: Sample program makefile
	jpeg.h, jpegcasm.h, jpegdasm.h, jpegex.h	Header file
	main0.c, main1.c, main2.c, main3.c, main4.c, main5.c, main6.c, main7.c, main8.c, getmcu.c, putmcu.c, tpycc.c, tprgb.c	Sample source file

```
└── smp830 ──── jpeg ────── (continuation of nectools/smp830/jpeg)
                            fish.s
                            fishprg.s
                            fishtga.s
                            fish.tga        : Filling image data file
                            fish.jpg        : Base line (4:1:1 (H:V = 2:2))
                            fish11.jpg      : Base line (1:1:1)
                            fish21.jpg      : Base line (2:1:1)
                            fish41.jpg      : Base line (4:1:1 (H:V = 4:1))
                            fishmono.jpg    : Monochrome
                            fishp3.jpg      : Progressive (spectral selection)
                            fishp4.jpg      : Progressive (successive approximation)
                            fishp5.jpg      : Progressive (successive approximation)
                            cmykbase.jpg    : Four-color base line
                            cmykprg3.jpg    : Four-color progressive (spectral selection)
                            cmykprg4.jpg    : Four-color (successive approximation)
                            cmykprg5.jpg    : Four-color (successive approximation)
```

*(4) AP705100-B03 (GHS version)

Directory	File	Description
ghstools — lib830	libjpegc.a	: (Basic compression) Compression main
	libjcmc1.a	: (Basic compression) MCU control; screen unit
	libjcmc2.a	: (Basic compression) MCU control; MCU line unit
	libjcy.a	: (Basic compression) getMCU; YCbCr
	libjcr.a	: (Basic compression) getMCU; RGB1
	libjcr2.a	: (Basic compression) getMCU; RGB2
	libjpegd.a	: (Basic expansion) Expansion main
	libjpgdp.a	: (Basic expansion) Expansion main
	libjdmc1.a	: (Basic expansion) MCU control; screen unit
	libjdmc2.a	: (Basic expansion) MCU control; MCU line unit
	libjdy.a	: (Basic expansion) putMCU; YCbCr
	libjdr.a	: (Basic expansion) putMCU; RGB1
	libjdr2.a	: (Basic expansion) putMCU; RGB2
	libjpeg.a	: (Basic) Common
	libjprg.a	: (Additional expansion) Progressive support
	libjprgd.a	: (Additional expansion) Debug library
	libjprog.a	: (Additional expansion) For symbol resolution
smp830 — jpeg	start.s	: Start-up routine
	make.lnk	: Section specification
	jparc0 jparc1 jparc2 jparc3 jparc4	Library selection file
	makefile	: make file of sample program
	jpeg.h jpegcasm.h jpegdasm.h jpegex.h	Header file
	main0.c main1.c main2.c main3.c main4.c main5.c main6.c main7.c main8.c getmcu.c putmcu.c tpycc.c tprgb.c	Sample source file

└─ smp830 ── jpeg ── (continuation of ghstools/smp830/jpeg)

fishtga.s	:	Filling image data file
fish.s	:	Base line (4:1:1 (H:V = 2:2))
fish11.s	:	Base line (1:1:1)
fish21.s	:	Base line (2:1:1)
fish41.s	:	Base line (4:1:1 (H:V = 4:1))
fishmono.s	:	Monochrome
fishp3.s	:	Progressive (spectral selection)
fishp4.s	:	Progressive (successive approximation)
fishp5.s	:	Progressive (successive approximation)
cmykbase.s	:	Four-color base line
cmykprg3.s	:	Four-color progressive (spectral selection)
cmykprg4.s	:	Four-color (successive approximation)
cmykprg5.s	:	Four-color (successive approximation)

1.3.7 Operating Environment

(1) Applicable CPU: V810 family (AP70732-B03)
V830 family (AP705100-B03)

(2) Compiler package

- **V810 family (AP70732-B03)**
  NEC ANSI-C compiler package
  CA732 (Windows or Sun4 version) Ver.1.00 or later
  GHS compiler package
  CC800 (Windows or Sun4 version) Ver.1.00 or later
- **V830 family (AP705100-B03)**
  NEC ANSI-C compiler package
  CA830 (Windows or Sun4 version) Ver.1.00 or later
  GHS compiler package
  CC800 (Windows or Sun4 version) Ver.1.00 or later

* **(3) Memory capacity**

Table 1-7. ROM Size (Unit: Bytes)

Basic library: Compression processing	Approx. 7 K
Basic library: Expansion processing	Approx. 14 K
Additional library: Expansion processing	Approx. 20 K

Table 1-8. RAM Size (Units: Bytes)

Processing system		Contents	Required No. of bytes		Remarks
			AP70732-B03	AP705100-B03	
Basic library	Compression	JPEG structures	1,152	128	Or less depending on corresponding sampling ratio (AP70732-B03)
		Internal RAM work area	—	1,024	Or less depending on corresponding sampling ratio (AP705100-B03)
		Other work area	2,688	2,688	
		APP structures	160	160	Necessary only when APPn segment is used
		Stack	Approx. 144	Approx. 128	
		Subtotal	Approx. 4,144	Approx. 4,128	
	Expansion	JPEG structures	1,152	128	Or less depending on corresponding sampling ratio (AP70732-B03)
		Internal RAM work area	—	1,024	Or less depending on corresponding sampling ratio (AP705100-B03)
		Other work area	3,968	3,968	
		APP structures	160	160	Necessary only when APPn segment is used
		Stack	Approx. 144	Approx. 128	
		Subtotal	Approx. 5,424	Approx. 5,408	
Additio-nal library	Expansion Two-pass setting	Work area	—	Approx. 5,000	Total of internal RAM and external RAM
		Stack	—	Approx. 500	
		Subtotal	—	Approx. 5,500	
	Expansion Single-pass setting	Work area	—	Approx. 2 M	Total of internal RAM and external RAM. This RAM size is for 640 x 480 pixels and three colors. Actually, a capacity proportional to the number of pixels and number of colors is necessary.
		Stack	—	Approx. 500	
		Subtotal	—	Approx. 2 M	

* 1.3.8 Section Name and Symbol Name Conventions

(1) Section name conventions

The sections used by the library are listed below.

Table 1-9. Sections Used by Library

Classification	Section name	Type	Description
Basic compression processing	.JPCTEXT	.text	Text (instruction code)
	.JPCTBL	.rodata	Table data (constant)
	.JPCDATA	.data	Data with initial value
	.JPCBSS	.bss	Data without initial value
Basic expansion processing	.JPDTEXT	.text	Text (instruction code)
	.JPDTBL	.rodata	Table data (constant)
	.JPDDATA	.data	Data with initial value
	.JPDBSS	.bss	Data without initial value
Basic common processing	.JPJTEXT	.text	Text (instruction code)
	.JPJTBL	.rodata	Table data (constant)
	.JPJDATA	.data	Data with initial value
	.JPJBSS	.bss	Data without initial value
Additional expansion processing	.JPDTEXT	.text	Text (instruction code)
	.JPDDATA	.data	Data with initial value

(2) Symbol name conventions

The symbols used in the JPEG library are named in compliance with the following conventions. When using these names in combination with other applications, ensure that they are not duplicated.

The global symbol name of the additional library is a character string starting with "_JPEGEX" (with underbar) with the assembler.

Table 1-10. Symbol Name Convention

Classification	Basic library	Additional library
Function/label name	Character string starting with "jpeg_"	Character string starting with "JPEGEX"
Symbol name	Character string starting with "JPEG"	
Structure name	CJINFO DJINFO APPINFO	JPEGEXINFO JPEGEXWORK JPEGEXVIDEO JPEGEXBUFF JPEGEXMCUSTR JPEGEXFrmINFO

1.3.9 Sample Program Memory Map

The memory map of the sample program included in the package is shown below.
See **Appendixes A** and **B.**

Figure 1-26. Sample Program Memory Map (AP70732-B03)

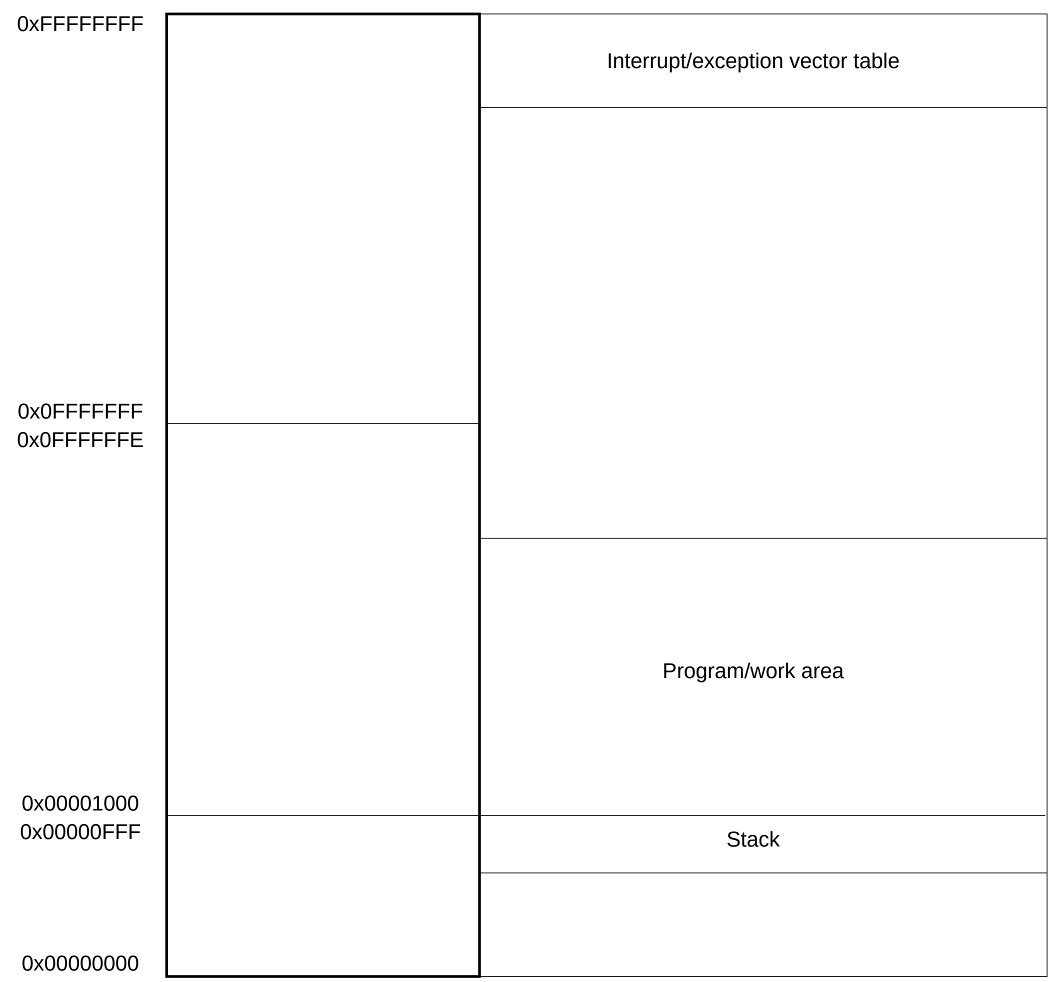
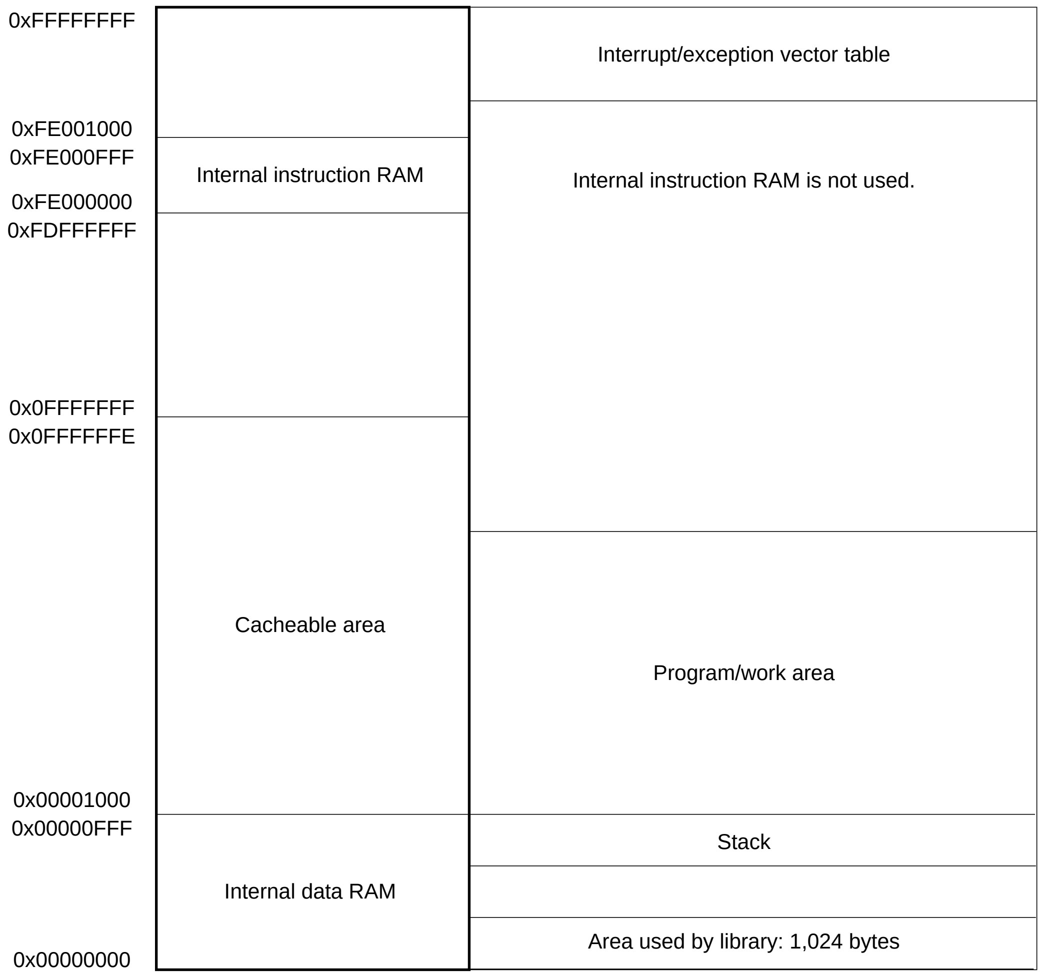


Figure 1-27. Sample Program Memory Map (AP705100-B03)



CHAPTER 2 BASIC LIBRARY SPECIFICATIONS

2.1 FUNCTION

The basic library group provided with the AP705100-B03 and AP70732-B03 enables the following two types of processing to be performed:

(1) Compression processing

A specified image is compressed into JPEG format by using a specified quantization table/Huffman table to create the JPEG file.
If the insertion of an application segment is specified, the segment is embedded into the header as an APPn segment.
A mode in which the number of bits of compressed data for 1 MCU is tested is provided.

(2) Expansion processing

A JPEG file is expanded.
Depending on the setting, expansion is not executed and only the image size and the address of an APPn segment are detected.
A rectangle can be clipped in MCU units and expanded, instead of the entire image.
An image can be reduced, relative to its normal size, such as 1/8 the length and width (1/64 of area).

2.1.1 Differences in Basic Library Operation Depending on VRAM (Image Memory) Configuration

If the VRAM (image memory) is of RGB type instead of YCbCr type, the following processing must be performed.

- Compression: RGB data must be translated to YCbCr then compressed.
- Expansion: Expanded YCbCr data must be translated to RGB before it is written into memory.

A separate object is linked depending on whether the image memory is of YCbCr or RGB type.

In addition, the basic library to be linked differs depending on whether the image memory has a sufficient capacity to store the data for the entire image or only part of the image.

Figure 2-1. Library for High-Capacity VRAM

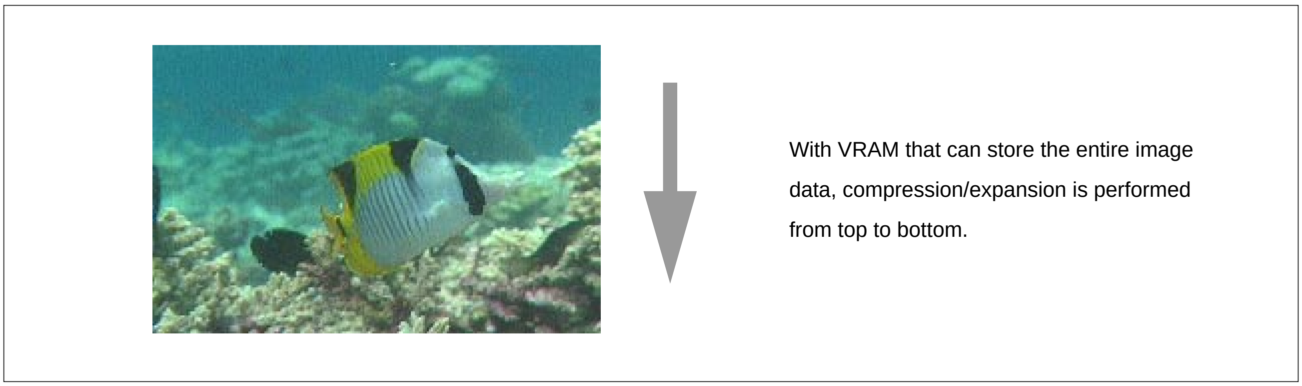


Figure 2-2. Library for Low-Capacity VRAM

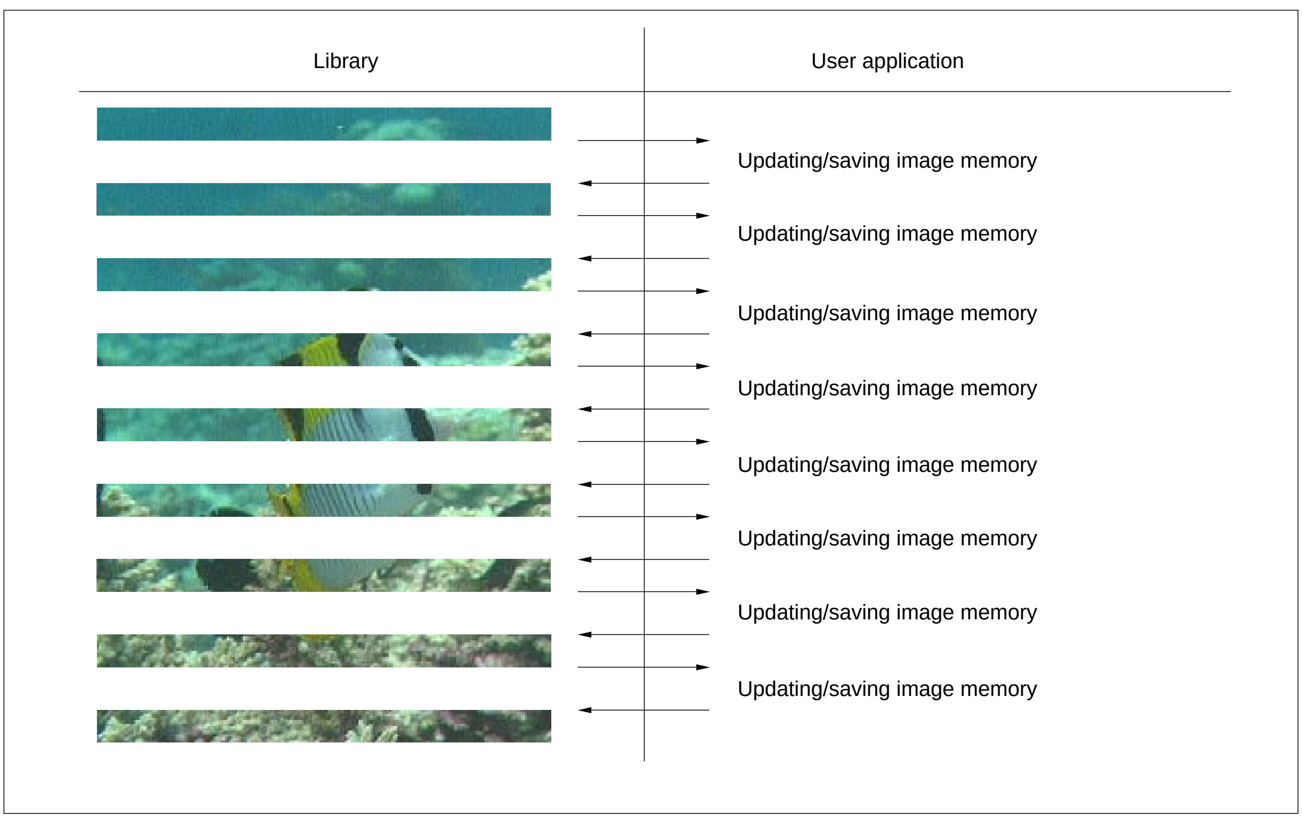


Table 2-1. Minimum Image Memory Capacity

Sampling ratio	Minimum image memory capacity
4:1:1 (H:V = 2:2)	RAM supporting random access of 16 vertical pixels
4:1:1 (H:V = 4:1)	RAM supporting random access of 8 vertical pixels
2:1:1 (H:V = 2:1)	RAM supporting random access of 8 vertical pixels
1:1:1 (H:V = 1:1)	RAM supporting random access of 8 vertical pixels

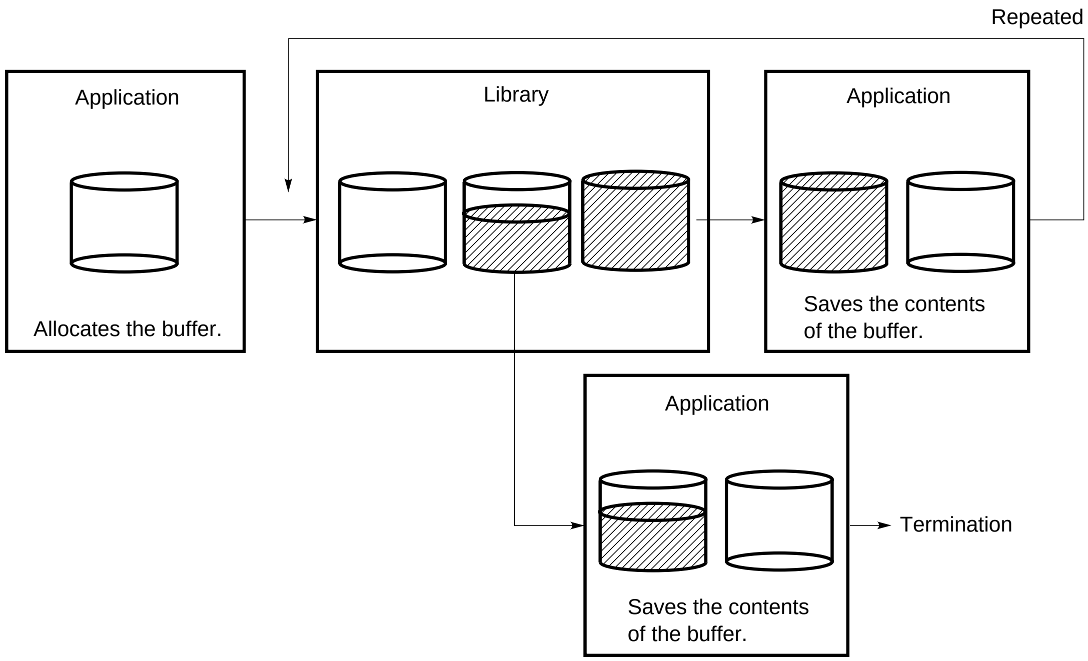
The memory capacity shown in Table 2-1 is required even for a system that does not have image memory of a size capable of storing the entire image. In a system with a relatively low memory capacity, compression/expansion of an image and update/save processing of the image memory are alternately and repeatedly executed in 16-dot-line (8-dot-line) units, as shown in Figure 2-2.

2.1.2 JPEG Buffer

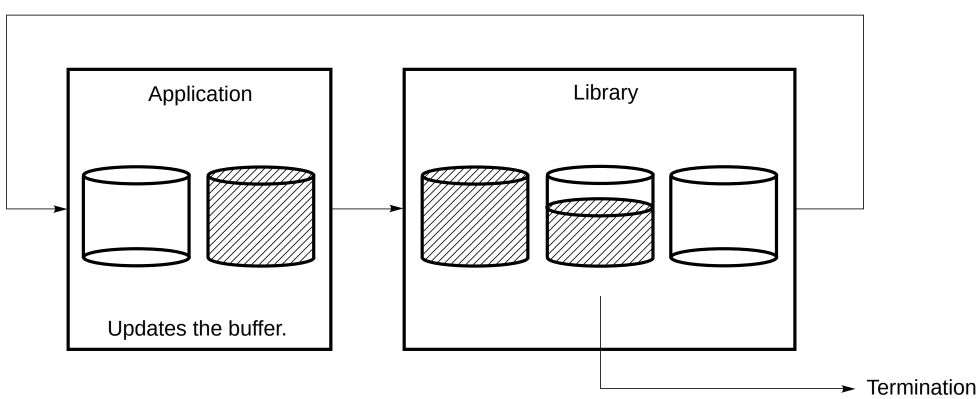
Generally, the size of the JPEG file varies considerably depending on the image or compression parameters. Moreover, the size of the file cannot be predicted easily from the image or parameters. With the AP705100-B03 and AP70732-B03 basic libraries, the processing is stopped once, the contents of the buffer are saved (compressed) or updated (expanded), then the processing is resumed if the JPEG file size is greater than that of the buffer prepared for the JPEG file.

Figure 2-3. Using the JPEG Buffer

(a) JPEG buffer during compression



(b) JPEG buffer during expansion



2.1.3 Precision of Operations

JPEG converts analog data to digital data. As a result of conversion, some information in the original data may be lost depending on the operation precision. This section describes the library quality in terms of operation precision.

Table 2-2. Information Loss Incurred by Each Type of Processing

Processing	Loss of information
Entropy encoding/decoding	No information is lost. Data which has been subjected to entropy encoding remains the same as that before entropy decoding.
Quantization/reverse quantization	Among all JPEG processes, quantization is the most likely cause of information loss. When the data obtained by a DCT operation is divided by the values in the quantization table, the remainders are discarded. If, however, the quantization parameter is set to 100 before compression, all elements in the quantization table are set to 1, so that no information is lost.
DCT conversion/reverse DCT conversion	Information is lost when: • The values output using an expression of DCT or reverse DCT conversion (frequency-disassembled factors) are treated as 16-bit integers. (The values must, however, be specified as real numbers in the expression.) • Fixed-point processing with 16-bit precision is performed to increase the processing speed.

The following explains the precision of the DCT and reverse DCT conversions.

Inspect the precision as follows:

(1) Allocate 20000 buffers for 64 short-type (2-byte) elements.

```
short BLK[20000][64];
```

(2) Using the following program for generating random numbers, generate integer image data, having values between -128 and 127, for 10000 blocks, then arrange the data such that each block consists of 8 x 8 pixels.
Generate a sign-reversed block for each of the 10000 generated blocks.
A total of 20000 blocks are used as the DCT conversion input.

```
int                     /*int is 32 bits */
rand (L, H)
int L, H;
{
            static int randx = 1;/*int is 32 bits */
            float     z = (float) 0x7FFFFFFF ;
            int       i, j ;
            float     x ;

            randx = (randx * 1103515245) + 12345 ;
            i = randx & 0x7FFFFFFF ;            /*keep 30 bits*/
            x = (  (float) i) /z ;        /*range 0 to 0.99999...*/
            x* = (L+H+1)  ;                     /*range 0 to <L+H+1*/
            j = (int) x ;                 /*truncate to integer*/
            return (j - L)  ;                   /*range -L to H*/
} ;
```

(3) Using the libraries, apply DCT then reverse DCT conversion to each block, named BLK[n] (where n is a number between 0 and 19999). The output value is specified as OUT[n][64] (where n is a number between 0 and 19999).

(4) Calculate the following errors between BLK[20000][64] and OUT[20000][64].

<1> Maximum error
<2> Mean square error for each element number
<3> Mean square error for all elements
<4> Mean error for each element number
<5> Mean error for all elements

Assume that the results of calculation are as follows:

Difference between the input data and output data:

$$\mathrm{DIFF}\,[b]\,[n] = \mathrm{BLK}\,[b]\,[n] - \mathrm{OUT}\,[b]\,[n]\ ;$$

$$(b = 0, ..., 19999\ ;\ n = 0, ..., 63)$$

<1> Maximum error:

$$\max_{b,\,n} |\,\mathrm{DIFF}\,[b]\,[n]\,| = 2$$

<2> Mean square error for each element number:

$$\max_{n} \left(\left(\sum_{b} (\mathrm{DIFF}\,[b]\,[n]\,)^2\right) /20000\right) = 0.3475$$

<3> Mean square error for all elements:

$$\left(\sum_{n} \sum_{b} (\mathrm{DIFF}\,[b]\,[n]\,)^2\right) /20000 \times 64 = 0.3313$$

<4> Mean error for each element number:

$$\max_{n} \left(\left(\sum_{b} |\,\mathrm{DIFF}\,[b]\,[n]\,|\right) /20000\right) = 0.3400$$

<5> Mean error for all elements:

$$\left(\sum_{n} \sum_{b} |\,\mathrm{DIFF}\,[b]\,[n]\,|\right) /20000 \times 64 = 0.3260$$

In the above example, if mean square error **<3>** has a value of 0.3313, the error resulting from DCT and reverse DCT conversion is approximately 0.33 gradations, for an overall range of 256 gradations.
The value of **<2>** (0.3475) is close to that of **<3>** (0.3313). This indicates that no one element in an 8 x 8 block has an excessively larger or smaller error than those of the other elements in the block, such that the entire block is equally loaded.

2.1.4 Compression Options

(1) Selecting a sampling ratio

With basic library, any of the following four sampling ratios can be selected:

- 4:1:1 (H:V = 2:2) (1 MCU consists of 16 pixels vertically and 16 pixels horizontally.)
- 4:1:1 (H:V = 4:1) (1 MCU consists of 8 pixels vertically and 32 pixels horizontally.)
- 2:1:1 (H:V = 2:1) (1 MCU consists of 8 pixels vertically and 16 pixels horizontally.)
- 1:1:1 (H:V = 1:1) (1 MCU consists of 8 pixels vertically and 8 pixels horizontally.)

Caution Sampling ratios other than those above are not supported.

(2) Huffman table and quantization table

The Huffman table and quantization table are parameters that have a significant influence on the sampling ratio. This library supports the specification of these tables.

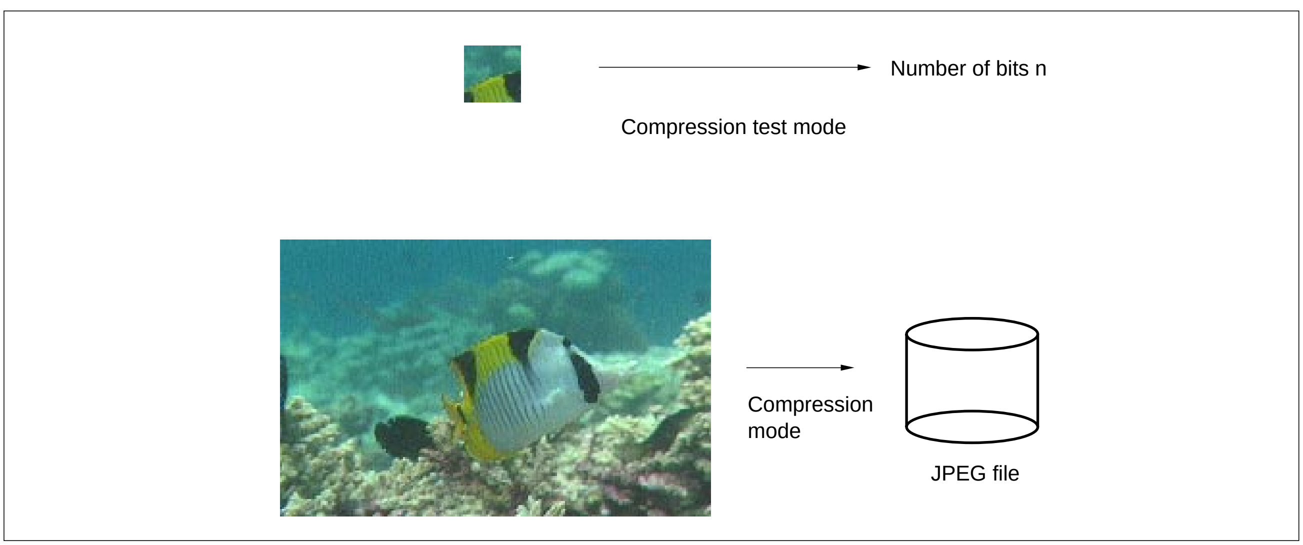
(3) Setting of quantization parameter

The quantization table is very useful for changing the compression ratio. The image quality must be traded-off against the compression ratio. This trade-off can be easily adjusted by specifying a quantization parameter.

(4) Selecting compression/compression test

A mode in which the image is actually compressed, and a mode into which the number of bits 1 MCU is compressed can be tested, are provided.

Figure 2-4. Compression Mode



(5) Restart interval

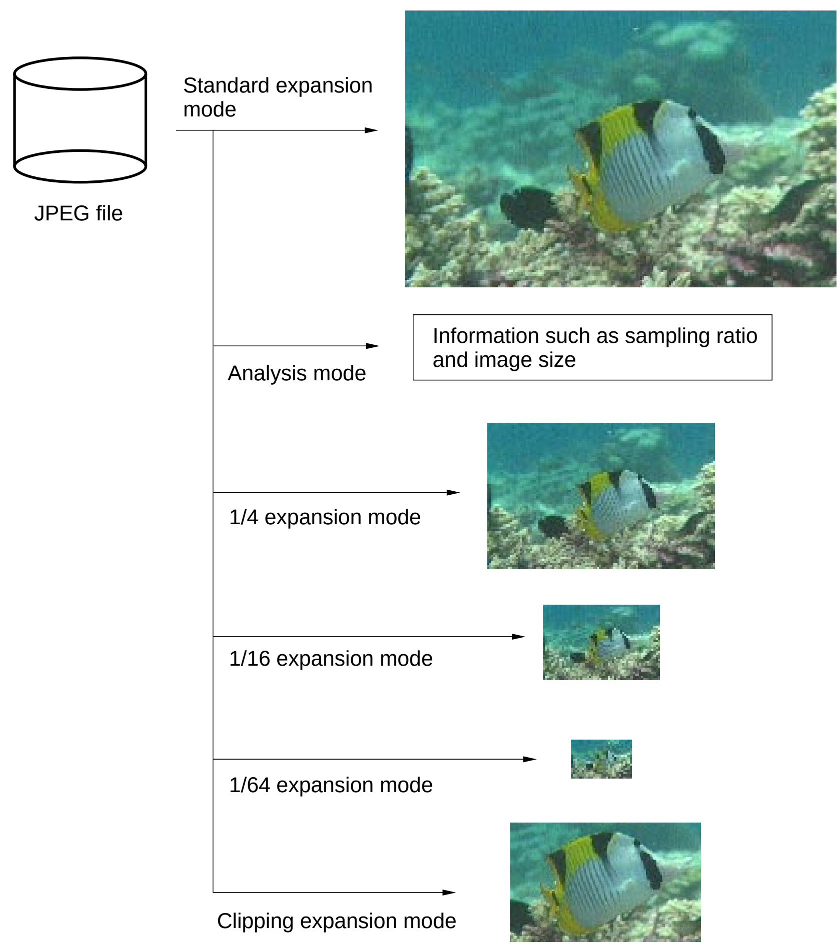
Whether a restart marker is used can be specified. When the marker is used, the interval at which the marker is inserted can also be selected.

2.1.5 Options for Basic Expansion

The main option for basic expansion is mode setting. Depending on the mode setting, whether only the JPEG header is analyzed, or whether the image is expanded normally, reduced or expanded to a Thumbnail, or clipped and expanded, is determined.

When the image is clipped, the position to be clipped is specified.

Figure 2-5. Expansion Mode



2.1.6 Notes on Compression Test Option

The number of bits n, used in compression test mode, is a calculated value because normal compression has a nature peculiar to JPEG (depends on MCU before and after) such as the differential value of a DC component (difference from the preceding block) being compressed and 0x00 being inserted to distinguish compressed data from a marker if the compressed data is 0xFF, in bytes.

Number of bytes constituting entire JPEG file

$$\fallingdotseq \{ \sum_{i=0}^{m} \sum_{j=0}^{n} \text{(Number of bits when MCU (m, n) is tested and compressed)} \} /8$$

+ Number of bytes required for header (about 300 bytes)

m: Number of MCUs in horizontal direction

n: Number of MCUs in vertical direction

2.2 LINKING BASIC LIBRARY

2.2.1 Selecting Library for Link

The user can select a library for the following three items during linking.

- Non-linking of unnecessary object
- YCbCr or RGB selection for VRAM
- Selection of processing of VRAM in image or MCU line units

To select a library, the following command is used:

jparc830.exe: for DOS
jparc830: for Sun4

Caution In DOS, execute this command from the command line.

By executing this function, file "archive" is created. If a file having the same name already exists, it is overwritten. This file is written in the make file and is referenced during linking.

(1) Do not link unnecessary objects.

When a command that creates file "archive" is executed, the following messages are displayed to set the non-linking of unnecessary codes. Respond to these messages as they are displayed.

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

.
.
.

```
Do the library must switch to the user application each 8 or 16 lines? (Y/N)
```

.
.
.

(2) Use of default VRAM access function

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

```
Do you want to use default-VRAM-access library? (Y/N)
```

If Y is selected in response to the above message, the following message is displayed. Select the desired item.

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

If the default VRAM access function is not used, create the following function.

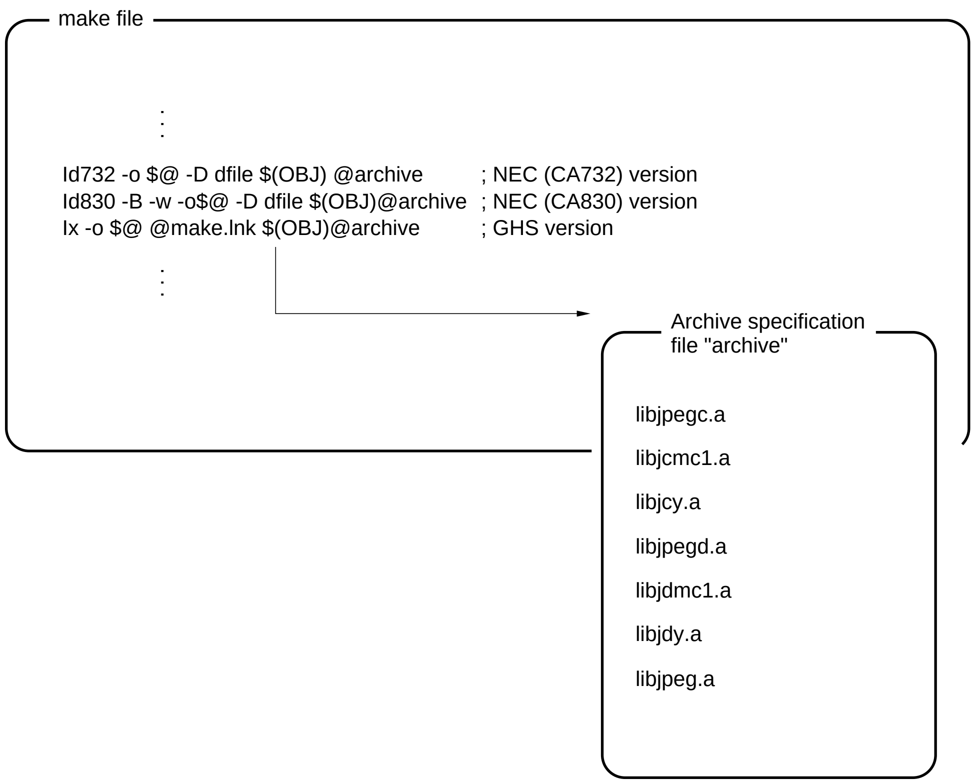
Compression: jpeg_getMCU22, jpeg_getMCU41, jpeg_getMCU21, jpeg_getMCU11
Expansion: jpeg_putMCU22x, jpeg_putMCU41x, jpeg_putMCU21x, jpeg_putMCU11x

For details, see **Section 2.6**.

2.2.2 Specifying an Archive File

When the command that specifies the creation of file "archive" 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 2-6. Specifying Archiver

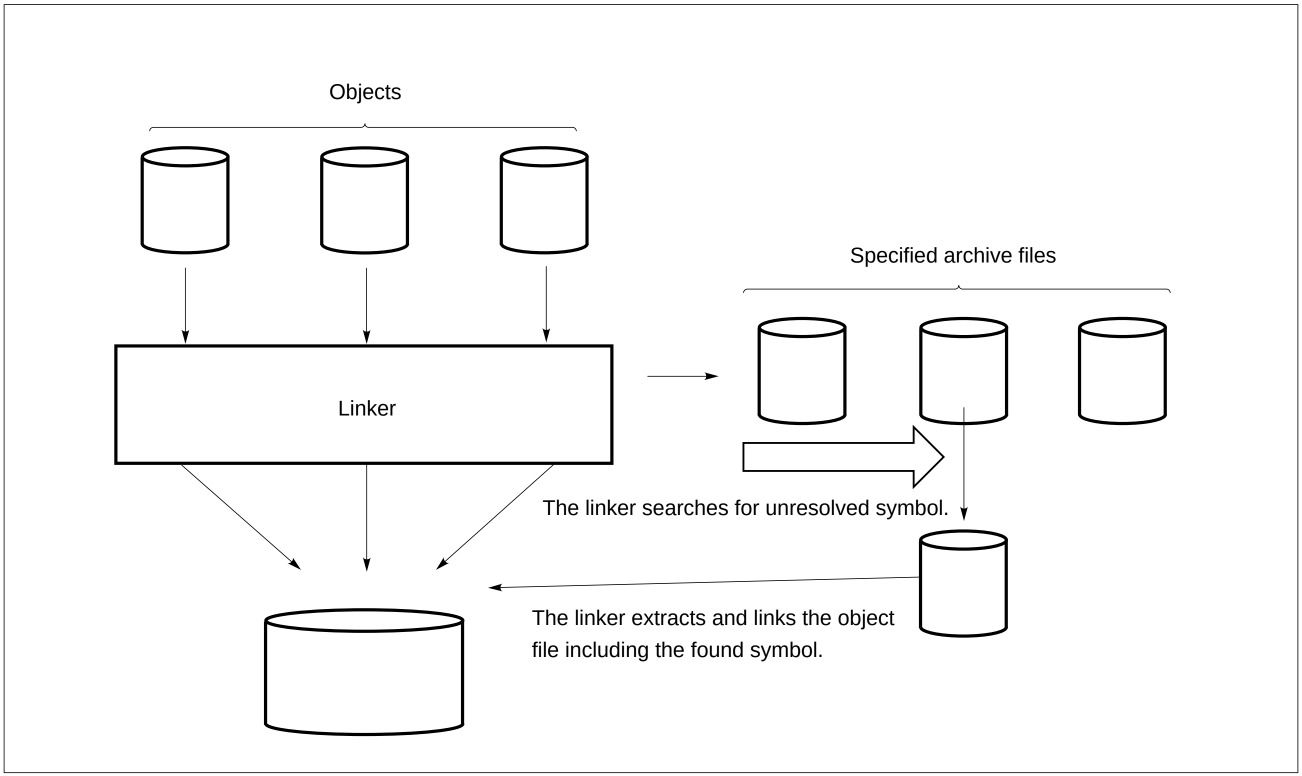


The default library is stored into archive file libjpeg.a.

To create file "archive" by using an editor, specify libjpeg.a at the end of the archive file specification.

The linker searches for specified archive files sequentially to resolve an unresolved symbol in the object. The object file including the found symbol is extracted from the archive files and linked.

Figure 2-7. Handling of Archive File by Linker



2.2.3 Advanced Library Specification

To reduce the instruction code size as much as possible to, for example, support a sampling ratio of 2:1:1 and not to link a sampling ratio of 4:1:1 or 1:1:1, directly rewrite archive file libjcmcx.a/libjdmcx.a.

```
ar732 t libjcmc1.a   (NEC CA732)
ar830 t libjcmc1.a   (NEC CA830)
ax t libjcmc1.a      (GHS)
```

When the above command is executed, the object file name included in the archive file can be displayed.

```
ar732 d libjcmc1.a jcmcu11.o   (NEC CA732)
ar830 d libjcmc1.a jcmcu11.o   (NEC CA830)
ax d libjcmc1.a jcmcu11.o      (GHS)
```

In this way, a specified object file can be deleted from the archive file. By deleting the object file for an unnecessary sampling ratio, the deleted object file is not linked.

Table 2-3. Object File Peculiar to Sampling Ratio of Compression Processing System

Sampling ratio	Object file (archive file)
4:1:1 (H:V = 2:2)	jcmcu22.o (libjcmc1.a/libjcmc2.a), gmcuyc22.o (libjcy.a), gmcurg22.o (libjcr.a)
4:1:1 (H:V = 4:1)	jcmcu41.o (libjcmc1.a/libjcmc2.a), gmcuyc41.o (libjcy.a), gmcurg41.o (libjcr.a)
2:1:1 (H:V = 2:1)	jcmcu21.o (libjcmc1.a/libjcmc2.a), gmcuyc21.o (libjcy.a), gmcurg21.o (libjcr.a)
1:1:1 (H:V = 1:1)	jcmcu11.o (libjcmc1.a/libjcmc2.a), gmcuyc11.o (libjcy.a), gmcurg11.o (libjcr.a)

Table 2-4. Object File Peculiar to Sampling Ratio of Expansion Processing System

Sampling ratio	Object file (archive file)
4:1:1 (H:V = 2:2)	jdmcu22c.o/jdmcu221.o/jdmcu222.o/jdmcu224.o/jdmcu228.o (libjdmc1.a/libjdmc2.a), pmcuy221.o/pmcuy222.o/pmcuy224.o/pmcuy228.o (libjdy.a), pmcur221.o/pmcur222.o/pmcur224.o/pmcur228.o (libjdr.a)
4:1:1 (H:V = 4:1)	jdmcu41c.o/jdmcu411.o/jdmcu412.o/jdmcu414.o/jdmcu418.o (libjdmc1.a/libjdmc2.a), pmcuy411.o/pmcuy412.o/pmcuy414.o/pmcuy418.o (libjdy.a), pmcur411.o/pmcur412.o/pmcur414.o/pmcur418.o (libjdr.a)
2:1:1 (H:V = 2:1)	jdmcu21c.o/jdmcu211.o/jdmcu212.o/jdmcu214.o/jdmcu218.o (libjdmc1.a/libjdmc2.a), pmcuy211.o/pmcuy212.o/pmcuy214.o/pmcuy218.o (libjdy.a), pmcur211.o/pmcur212.o/pmcur214.o/pmcur218.o (libjdr.a)
1:1:1 (H:V = 1:1)	jdmcu11c.o/jdmcu111.o/jdmcu112.o/jdmcu114.o/jdmcu118.o (libjdmc1.a/libjdmc2.a), pmcuy111.o/pmcuy112.o/pmcuy114.o/pmcuy118.o (libjdy.a), pmcur111.o/pmcur112.o/pmcur114.o/pmcur118.o (libjdr.a)

2.2.4 Support for ABcond Instruction

Of basic libraries, only those which do not include the ABcond instruction (conditional branch instruction with branch history function) are linked.

To link the libraries which include ABcond instruction (or whose branch instructions are partially replaced with high-speed conditional branch instructions), execute the scripts listed in Table 2-5.

Remark Using high-speed conditional branch instructions speeds up the processing, but does not affect the code size.

Table 2-5. Scripts Required for Processing Basic Libraries (1/2)

NEC compiler	GHS compiler
Ar830 d libjcr.a gmcurg22.o	ax d libjcr.a gmcurg22.o
Ar830 d libjcr.a gmcurg41.o	ax d libjcr.a gmcurg41.o
Ar830 d libjcr.a gmcurg21.o	ax d libjcr.a gmcurg21.o
Ar830 d libjcr.a gmcurg11.o	ax d libjcr.a gmcurg11.o
Ar830 d libjcr2.a g6curg22.o	ax d libjcr2.a g6curg22.o
Ar830 d libjcr2.a g6curg41.o	ax d libjcr2.a g6curg41.o
Ar830 d libjcr2.a g6curg21.o	ax d libjcr2.a g6curg21.o
Ar830 d libjcr2.a g6curg11.o	ax d libjcr2.a g6curg11.o
Ar830 d libjcy.a gmcuyc22.o	ax d libjcy.a gmcuyc22.o
Ar830 d libjcy.a gmcuyc41.o	ax d libjcy.a gmcuyc41o
Ar830 d libjcy.a gmcuyc21.o	ax d libjcy.a gmcuyc21.o
Ar830 d libjcy.a gmcuyc11.o	ax d libjcy.a gmcuyc11.o
Ar830 d libjdr.a pmcur228.o	ax d libjdr.a pmcur228.o
Ar830 d libjdr.a pmcur418.o	ax d libjdr.a pmcur418.o
ar830 d libjdr.a pmcur218.o	ax d libjdr.a pmcur218.o
ar830 d libjdr.a pmcur118.o	ax d libjdr.a pmcur118.o
ar830 d libjdr.a pmcur414.o	ax d libjdr.a pmcur414.o
ar830 d libjdr.a pmcur214.o	ax d libjdr.a pmcur214.o
ar830 d libjdr.a pmcur114.o	ax d libjdr.a pmcur114.o
ar830 d libjdr2.a p6cur228.o	ax d libjdr2.a p6cur228.o
ar830 d libjdr2.a p6cur418.o	ax d libjdr2.a p6cur418.o
ar830 d libjdr2.a p6cur218.o	ax d libjdr2.a p6cur218.o
ar830 d libjdr2.a p6cur118.o	ax d libjdr2.a p6cur118.o
ar830 d libjdr2.a p6cur414.o	ax d libjdr2.a p6cur414.o
ar830 d libjdr2.a p6cur214.o	ax d libjdr2.a p6cur214.o
ar830 d libjdr2.a p6cur114.o	ax d libjdr2.a p6cur114.o
ar830 d libjdy.a pmcuy228.o	ax d libjdr2.a p6cuy228.o
ar830 d libjdy.a pmcuy418.o	ax d libjdy.a pmcuy418.o
ar830 d libjdy.a pmcuy218.o	ax d libjdy.a pmcuy218.o
ar830 d libjdy.a pmcuy118.o	ax d libjdy.a pmcuy118.o
ar830 d libjdy.a pmcuy414.o	ax d libjdy.a pmcuy414.o
ar830 d libjdy.a pmcuy214.o	ax d libjdy.a pmcuy214.o
ar830 d libjdy.a pmcuy114.o	ax d libjdy.a pmcuy114.o
ar830 d libjpeg.a jfwddct.o	ax d libjpeg.a jfwddct.o
ar830 d libjpeg.a jchuff.o	ax d libjpeg.a jchuff.o
ar830 d libjpeg.a jdhuff.o	ax d libjpeg.a jdhuff.o

Table 2-5. Scripts Required for Processing Basic Libraries (2/2)

NEC compiler	GHS compiler
ar830 d libjpeg.a jrdct1.o ar830 d libjpeg.a jrdct2p.o ar830 d libjpeg.a jrdct4p.o ar830 d libjpeg.a jrdct8.o	ax d libjpeg.a jrdct1.o ax d libjpeg.a jrdct2p.o ax d libjpeg.a rdct4p.o ax d libjpeg.a jrdct8.o

2.2.5 Added RGB Libraries (libjcr2.a, libjdr2.a)

CCIR Recommendation 601-1 defines the expressions of transformation between RGB and YCbCr as follows:

$$\left.\begin{aligned} Y &= 0.29900 \times R + 0.58700 \times G + 0.11400 \times B \\ Cb &= -0.16874 \times R - 0.33126 \times G + 0.50000 \times B \\ Cr &= 0.50000 \times R - 0.41869 \times G - 0.08131 \times B \end{aligned}\right\}$$

... **<1>**

$$\left.\begin{aligned} R &= Y + 1.40200 \times Cr \\ G &= Y - 0.34414 \times Cb - 0.71414 \times Cr \\ B &= Y + 1.77200 \times Cb \end{aligned}\right\}$$

... **<2>**

In some cases, the following transformation expressions are used.

$$\left.\begin{aligned} Y &= 0.2990 \times R + 0.5870 \times G + 0.1140 \times B \\ Cb &= -0.1684 \times R - 0.3316 \times G + 0.5000 \times B \\ Cr &= 0.5000 \times R - 0.4187 \times G - 0.0813 \times B \end{aligned}\right\}$$

... **<3>**

$$\left.\begin{aligned} R &= Y + 1.4020 \times Cr \\ G &= Y - 0.3441 \times Cb - 0.7139 \times Cr \\ B &= Y + 1.7718 \times Cb - 0.0013 \times Cr \end{aligned}\right\}$$

... **<4>**

If the use of the default VRAM access library and the VRAM type RGB is selected using the AP705100-B03 or AP70732-B03 basic libraries, the following libraries are linked.

Compression processing : A library based on expression **<1>**.
Expansion processing : A library based on expression **<2>**.

For example, when color transformation is performed according to **<3>** and **<4>** with a Windows application or the like, the intensity of the red component is reduced a little, if the JPEG file created using **<1>** is expanded using **<4>**.

To substitute **<3>** for **<1>**, or **<4>** for **<2>** in AP705100-B03 or AP70732-B03 basic libraries, change the linker option specified as follows:

Compression processing : Change the linker option specified from libjcr.a to libjcr2.a.
Expansion processing : Change the linker option specified from libjdr.a to libjdr2.a.

2.2.6 Memory Map of Link

Mapping for each section is performed by the following files:

- NEC version: dfile (link directive file)
- GHS version: make.lnk (section specification file)

The user must rewrite these files in the same manner as the make file.
Rewrite them by referring to the file provided as a sample.
For details of these files, such as their format, refer to the following description in the manual of the linker.

- NEC version: Link directive
- GHS version: -sec option

2.2.7 Compile Option

The basic library uses all of the 32 registers. Therefore, modes other than that for 32 registers are not supported.

For details of the other compile options, refer to the manual provided with each compiler.

2.3 BASIC LIBRARY STRUCTURE AND MEMORY

With basic library, allocate memory of the specified size to each processing of compression and expansion.

With the V810 family version

Memory	Usage and size
CJINFO structure	1,152 bytes max. required for compression processing (differs with the sampling ratio)
DJINFO structure	1,152 bytes max. required for expansion processing (differs with the sampling ratio)
APPINFO structure	160 bytes max. required for embedding information in APP segment for compression, or obtaining information on APP segment for expansion. Do not allocate this structure to internal RAM.
JPEG buffer	Buffer to store completed JPEG file for compression and JPEG file to be expanded for expansion. Any number of bytes can be set. If a JPEG file is too large to be stored in a single operation, it must be divided.
External RAM work area	2,688 bytes required for compression, and 3,968 bytes required for expansion

With the V830 family version

Memory	Usage and size
CJINFO structure	128 bytes required for compression processing
DJINFO structure	128 bytes required for expansion processing
APPINFO structure	160 bytes max. required for embedding information in APP segment for compression, or obtaining information on APP segment for expansion. Do not allocate this structure to internal RAM.
Internal RAM work area	1,024 bytes max. required, depending on the sampling ratio.
JPEG buffer	Buffer to store completed JPEG file for compression and JPEG file to be expanded for expansion. Any number of bytes can be set. If a JPEG file is too large to be stored in a single operation, it must be divided.
External RAM work area	2,688 bytes required for compression, and 3,968 bytes required for expansion

2.3.1 CJINFO Structure

The CJINFO structure is used for compression processing.

The type of this structure is defined in file jpeg.h.

The first address of this structure is passed to the compression routine as an argument.

Table 2-6. CJINFO Structure (AP70732-B03) (1/2)

Member	Type	Description	In/Out
ErrorState	int	Error status	In/Out
FileSize	int	JPEG file size	Out
Restart	unsigned short	Restart interval	In
Width	unsigned short	Number of horizontal pixels of image	In
Height	unsigned short	Number of vertical pixels of image	In
Quality	char	Quantization parameter	In
Sampling	char	Sampling ratio	In
Mode	char	Compression mode	In
Reserve	char x 3	Reserved	-
JPEG_Buff_Bptr	unsigned char*	JPEG buffer first address	In
JPEG_Buff_Eptr	unsigned char*	JPEG buffer first address + JPEG buffer size	In
IRAM_Buff_Bptr	int*	Reserved	-
StartX	short	Start x position of image (number of pixels)	In
StartY	short	Start y position of image (number of pixels)	In
VRAM_Bptr	unsigned char*	VRAM first address	In**Note 1**
VRAM_W_Pixel	short	Horizontal width of VRAM (number of pixels)	In**Note 2**
VRAM_H_Pixel	short	Vertical width of VRAM (number of pixels)	In**Note 2**
VRAM_Line_Byte	int	Address difference of one vertical pixel of VRAM	In**Note 1**
VRAM_Pixel_Byte	int	Address difference of one horizontal pixel of VRAM	In**Note 1**
VRAM_Gap1_Byte	int	Address difference of /R and B between Y and Cb of VRAM	In**Note 1**
VRAM_Gap2_Byte	int	Address difference of /R and G between Y and Cr of VRAM	In**Note 1**
APP_Info_Bptr	APPINFO*	APPINFO structure first address	In
DQT_Y_Bptr	char*	Luminance component quantization table first address	In
DQT_C_Bptr	char*	Chrominance component quantization table first address	In
DHT_DC_Y_Bptr	char*	Luminance DC Huffman table first address	In
DHT_DC_C_Bptr	char*	Luminance AC Huffman table first address	In
DHT_AC_Y_Bptr	char*	Chrominance DC Huffman table first address	In

Notes 1. These members need not be set if the getmcu function is created by the user.

2. Set these members as dummies if the getmcu function is created by the user.

Table 2-6. CJINFO Structure (AP70732-B03) (2/2)

Member	Type	Description	In/Out
DHT_AC_C_Bptr	char*	Chrominance AC Huffman table first address	In
Work	int*	External RAM work area first address	In
CurrentX	short	VRAM drawing work (used internally)	-
CurrentY	short	VRAM drawing work (used internally)	-
IR	32 + 256 byte	Internally reserved (internally used work area)	**Note**
MCUbuff	0x180 unsigned short	MCU buffer	-

* **Note** Clear the area of the IR member to 0. For compression in any of the following JPEG formats, however, set a value in the IR area.

- Address specification insertion of comment marker (See **Section 2.4.4**.)
- Exit format compression (See **Sections 2.4.5** and **2.4.7**.)

Table 2-7. CJINFO Structure (AP705100-B03) (1/2)

Member	Type	Description	In/Out
ErrorState	int	Error status	In/Out
FileSize	int	JPEG file size	Out
Restart	unsigned short	Restart interval	In
Width	unsigned short	Number of horizontal pixels of image	In
Height	unsigned short	Number of vertical pixels of image	In
Quality	char	Quantization parameter	In
Sampling	char	Sampling ratio	In
Mode	char	Compression mode	In
Reserve	char x 3	Reserved	-
JPEG_Buff_Bptr	unsigned char*	JPEG buffer first address	In
JPEG_Buff_Eptr	unsigned char*	JPEG buffer first address + JPEG buffer size	In
IRAM_Buff_Bptr	int*	Internal RAM work area first address	In
StartX	short	Start x position of image (number of pixels)	In
StartY	short	Start y position of image (number of pixels)	In
VRAM_Bptr	unsigned char*	VRAM first address	In**Note 1**
VRAM_W_Pixel	short	Horizontal width of VRAM (number of pixels)	In**Note 2**
VRAM_H_Pixel	short	Vertical width of VRAM (number of pixels)	In**Note 2**
VRAM_Line_Byte	int	Address difference of one vertical pixel of VRAM	In**Note 1**
VRAM_Pixel_Byte	int	Address difference of one horizontal pixel of VRAM	In**Note 1**
VRAM_Gap1_Byte	int	Address difference of /R and B between Y and Cb of VRAM	In**Note 1**
VRAM_Gap2_Byte	int	Address difference of /R and G between Y and Cr of VRAM	In**Note 1**
APP_Info_Bptr	APPINFO*	APPINFO structure first address	In
DQT_Y_Bptr	char*	Luminance component quantization table first address	In
DQT_C_Bptr	char*	Chrominance component quantization table first address	In
DHT_DC_Y_Bptr	char*	Luminance DC Huffman table first address	In
DHT_DC_C_Bptr	char*	Luminance AC Huffman table first address	In
DHT_AC_Y_Bptr	char*	Chrominance DC Huffman table first address	In
DHT_AC_C_Bptr	char*	Chrominance AC Huffman table first address	In
Work	int*	External RAM work area first address	In

Notes 1. These members need not be set if the getmcu function is created by the user.

2. Set these members as dummies if the getmcu function is created by the user.

Table 2-7. CJINFO Structure (AP705100-B03) (2/2)

Member	Type	Description	In/Out
CurrentX	short	VRAM drawing work (used internally)	-
CurrentY	short	VRAM drawing work (used internally)	-
IR	32 byte	Internally reserved (internally used work area)	**Note**

Note Clear the area of the IR member to 0. For compression in any of the following JPEG formats, however, set a value in the IR area.

*

- Address specification insertion of comment marker (See **Section 2.4.4**.)
- Exit format compression (See **Sections 2.4.5** and **2.4.7**.)

2.3.2 DJINFO Structure

The DJINFO structure is used for basic expansion processing.
The type of this structure is defined in file jpeg.h.
The first address of this structure is passed to the expansion routine as an argument.

Table 2-8. DJINFO Structure (AP70732-B03) (1/2)

Member	Type	Description	In/Out
ErrorState	int	Error status	In/Out
FileSize	int	JPEG file size	Out
Restart	unsigned short	Restart interval	Out
Width	unsigned short	Number of horizontal pixels of image	Out
Height	unsigned short	Number of vertical pixels of image	Out
Sampling	char	Sampling ratio	Out
Mode	char	Expansion mode	In
JPEG_Buff_Bptr	unsigned char*	JPEG buffer first address	In
JPEG_Buff_Eptr	unsigned char*	JPEG buffer first address + JPEG buffer size	In
IRAM_Buff_Bptr	int*	Reserved	-
StartX	short	Start x position of image (number of pixels)	In
StartY	short	Start y position of image (number of pixels)	In
VRAM_Bptr	unsigned char*	VRAM first address	In**Note 1**
VRAM_W_Pixel	short	Horizontal width of VRAM (number of pixels)	In**Note 2**
VRAM_H_Pixel	short	Vertical width of VRAM (number of pixels)	In**Note 2**
VRAM_Line_Byte	int	Address difference of one vertical pixel of VRAM	In**Note 1**
VRAM_Pixel_Byte	int	Address difference of one horizontal pixel of VRAM	In**Note 1**
VRAM_Gap1_Byte	int	Address difference of /R and B between Y and Cb of VRAM	In**Note 1**
VRAM_Gap2_Byte	int	Address difference of /R and G between Y and Cr of VRAM	In**Note 1**
APP_Info_Bptr	APPINFO*	APPINFO structure first address	In
ClipSX	unsigned short	Clipping start position (valid only in clipping mode)	In
ClipSY	unsigned short	Clipping start position (valid only in clipping mode)	In
ClipW	unsigned short	Clipping horizontal width (valid only in clipping mode)	In
ClipH	unsigned short	Clipping vertical width (valid only in clipping mode)	In
Work	int*	External RAM work area first address	In
CurrentX	short	VRAM drawing work (used internally)	-

Notes 1. These members need not be set if the putmcu function is created by the user.
2. Set these members as dummies if the putmcu function is created by the user.

Table 2-8. DJINFO Structure (AP70732-B03) (2/2)

Member	Type	Description	In/Out
CurrentY	short	VRAM drawing work (used internally)	-
IR	52 + 256 byte	Internally reserved (internally used work area)	**Note**
MCUbuff	0x180 unsigned short	MCU buffer	-

* **Note** Clear the area of the IR member to 0. For expansion in the following JPEG formats, however, set a value in the IR area.

- Exit format expansion (See **Section 2.5.4**.)

Table 2-9. DJINFO Structure (AP705100-B03)

Member	Type	Description	In/Out
ErrorState	int	Error status	In/Out
FileSize	int	JPEG file size	Out
Restart	unsigned short	Restart interval	Out
Width	unsigned short	Number of horizontal pixels of image	Out
Height	unsigned short	Number of vertical pixels of image	Out
Sampling	char	Sampling ratio	Out
Mode	char	Expansion mode	In
JPEG_Buff_Bptr	unsigned char*	JPEG buffer first address	In
JPEG_Buff_Eptr	unsigned char*	JPEG buffer first address + JPEG buffer size	In
IRAM_Buff_Bptr	int*	Internal RAM work area first address	In
StartX	short	Start x position of image (number of pixels)	In
StartY	short	Start y position of image (number of pixels)	In
VRAM_Bptr	unsigned char*	VRAM first address	In**Note 1**
VRAM_W_Pixel	short	Horizontal width of VRAM (number of pixels)	In**Note 2**
VRAM_H_Pixel	short	Vertical width of VRAM (number of pixels)	In**Note 2**
RAM_Line_Byte	int	Address difference of one vertical pixel of VRAM	In**Note 1**
VRAM_Pixel_Byte	int	Address difference of one horizontal pixel of VRAM	In**Note 1**
VRAM_Gap1_Byte	int	Address difference of /R and B between Y and Cb of VRAM	In**Note 1**
VRAM_Gap2_Byte	int	Address difference of /R and G between Y and Cr of VRAM	In**Note 1**
APP_Info_Bptr	APPINFO*	APPINFO structure first address	In
ClipSX	unsigned short	Clipping start position (valid only in clipping mode)	In
ClipSY	unsigned short	Clipping start position (valid only in clipping mode)	In
ClipW	unsigned short	Clipping horizontal width (valid only in clipping mode)	In
ClipH	unsigned short	Clipping vertical width (valid only in clipping mode)	In
Work	int*	External RAM work area first address	In
CurrentX	short	VRAM drawing work (used internally)	-
CurrentY	short	VRAM drawing work (used internally)	-
IR	52 byte	Internally reserved (internally used work area)	**Note 3**

Notes 1. These members need not be set if the putmcu function is created by the user.

2. Set these members as dummies if the putmcu function is created by the user.

* **3.** Clear the area of the IR member to 0. For expansion in the following JPEG formats, however, set a value in the IR area.

- Exit format expansion (See **Section 2.5.4**.)

2.3.3 APPINFO Structure

The APPINFO structure is required for obtaining information on the APPn segment for basic expansion if the APPn segment is embedded in the JPEG file for compression (this structure is common to both the AP70732-B03 and AP705100-B03 basic libraries).

To support the APPn segment for compression/expansion processing, declare this APPINFO structure and register its first address in member APP of the CJINFO structure/DJINFO structure.

Table 2-10. APPINFO Structure (1/2)

Member	Type	Description
APP00_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP0 segment
APP01_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP1 segment
APP02_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP2 segment
APP03_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP3 segment
APP04_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP4 segment
APP05_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP5 segment
APP06_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP6 segment
APP07_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP7 segment
APP08_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP8 segment
APP09_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APP9 segment
APP10_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPA segment
APP11_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPB segment
APP12_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPC segment
APP13_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPD segment
APP14_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPE segment
APP15_Buff_Bptr	unsigned char*	Address of data buffer to be embedded in APPF segment
APP00_BuffSize	short	Data length to be embedded in APP0 segment (number of bytes)
APP01_BuffSize	short	Data length to be embedded in APP1 segment (number of bytes)
APP02_BuffSize	short	Data length to be embedded in APP2 segment (number of bytes)
APP03_BuffSize	short	Data length to be embedded in APP3 segment (number of bytes)
APP04_BuffSize	short	Data length to be embedded in APP4 segment (number of bytes)
APP05_BuffSize	short	Data length to be embedded in APP5 segment (number of bytes)
APP06_BuffSize	short	Data length to be embedded in APP6 segment (number of bytes)
APP07_BuffSize	short	Data length to be embedded in APP7 segment (number of bytes)
APP08_BuffSize	short	Data length to be embedded in APP8 segment (number of bytes)
APP09_BuffSize	short	Data length to be embedded in APP9 segment (number of bytes)
APP10_BuffSize	short	Data length to be embedded in APPA segment (number of bytes)

Table 2-10. APPINFO Structure (2/2)

Member	Type	Description
APP11_BuffSize	short	Data length to be embedded in APPB segment (number of bytes)
APP12_BuffSize	short	Data length to be embedded in APPC segment (number of bytes)
APP13_BuffSize	short	Data length to be embedded in APPD segment (number of bytes)
APP14_BuffSize	short	Data length to be embedded in APPE segment (number of bytes)
APP15_BuffSize	short	Data length to be embedded in APPF segment (number of bytes)

2.3.4 MCU Buffer

The minimum unit in which JPEG processing can be performed is called an MCU (Minimum Coded Unit).

The basic library requires a buffer (MCU buffer) to store the image data (from intermediate image data to final image data) when the image is compressed or expanded in this unit.

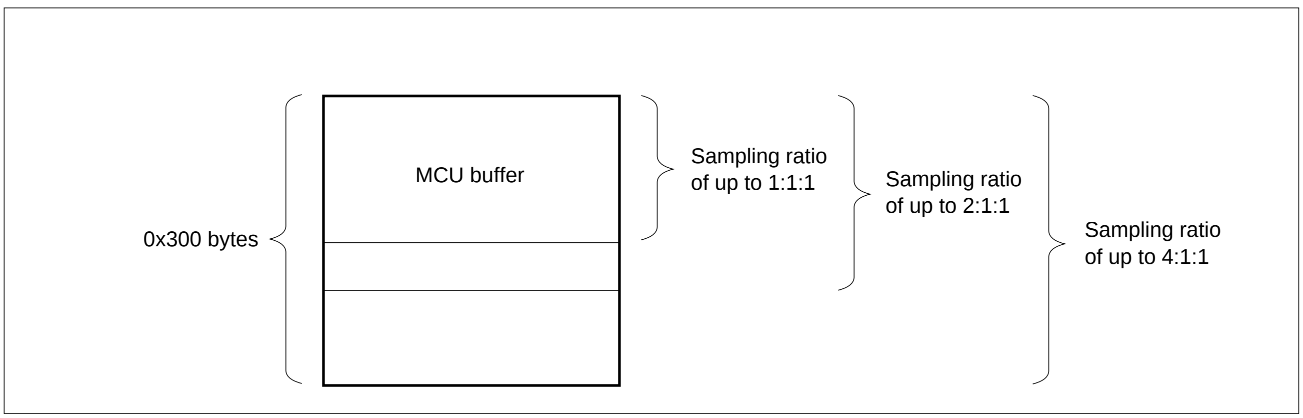
This MCU buffer is allocated to the last member of the CJINFO structure/DJINFO structure for the AP70732-B03. With the AP705100-B03, the MCU buffer is allocated to addresses following the first address of the internal data RAM work area + 0x100 bytes.

The size of the required MCU buffer is as follows:

Table 2-11. Size of MCU Buffer

Supported sampling ratio	Required size
4:1:1	0x300 bytes
2:1:1	0x200 bytes
1:1:1	0x180 bytes

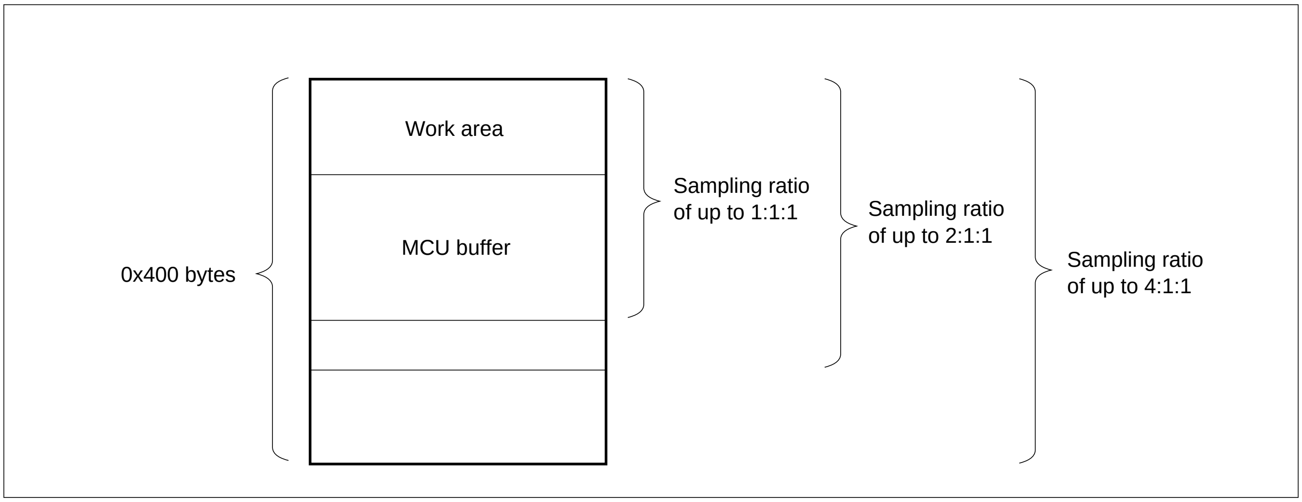
Figure 2-8. Use of MCU Buffer (AP70732-B03)



If only a sampling ratio of up to 2:1:1 is used for compression, the memory size taken up by the structure can be reduced by directly rewriting the structure definition of JPEG.H (MCU buffer size = 0x200 bytes).

If the sampling ratio of the JPEG file to be expanded is 4:1:1, and if an expansion library of 4:1:1 is linked, it is recognized that the MCU buffer has size of 0x300 bytes, and 0x300 bytes from the first address are overwritten without warning.

Figure 2-9. Use of Internal RAM Work Area (AP705100-B03)



If the sampling ratio of the JPEG file to be expanded is 4:1:1, and if an expansion library of 4:1:1 is linked, 0x400 bytes from the first address of the internal RAM work area are overwritten without warning.

2.3.5 JPEG Buffer

The JPEG buffer is an area used to store a JPEG file. The size of this buffer can be set to any number of bytes starting from 1 byte. If the buffer becomes full as a result of compression, or if the end of the buffer is reached as a result of expansion, the basic library stops processing, saves the required register contents, then restores the values of the required registers (register dispatch). If the library performs this processing too many times, the overall processing time is extended. Allocate an area of an appropriate size and use this area as the JPEG buffer.

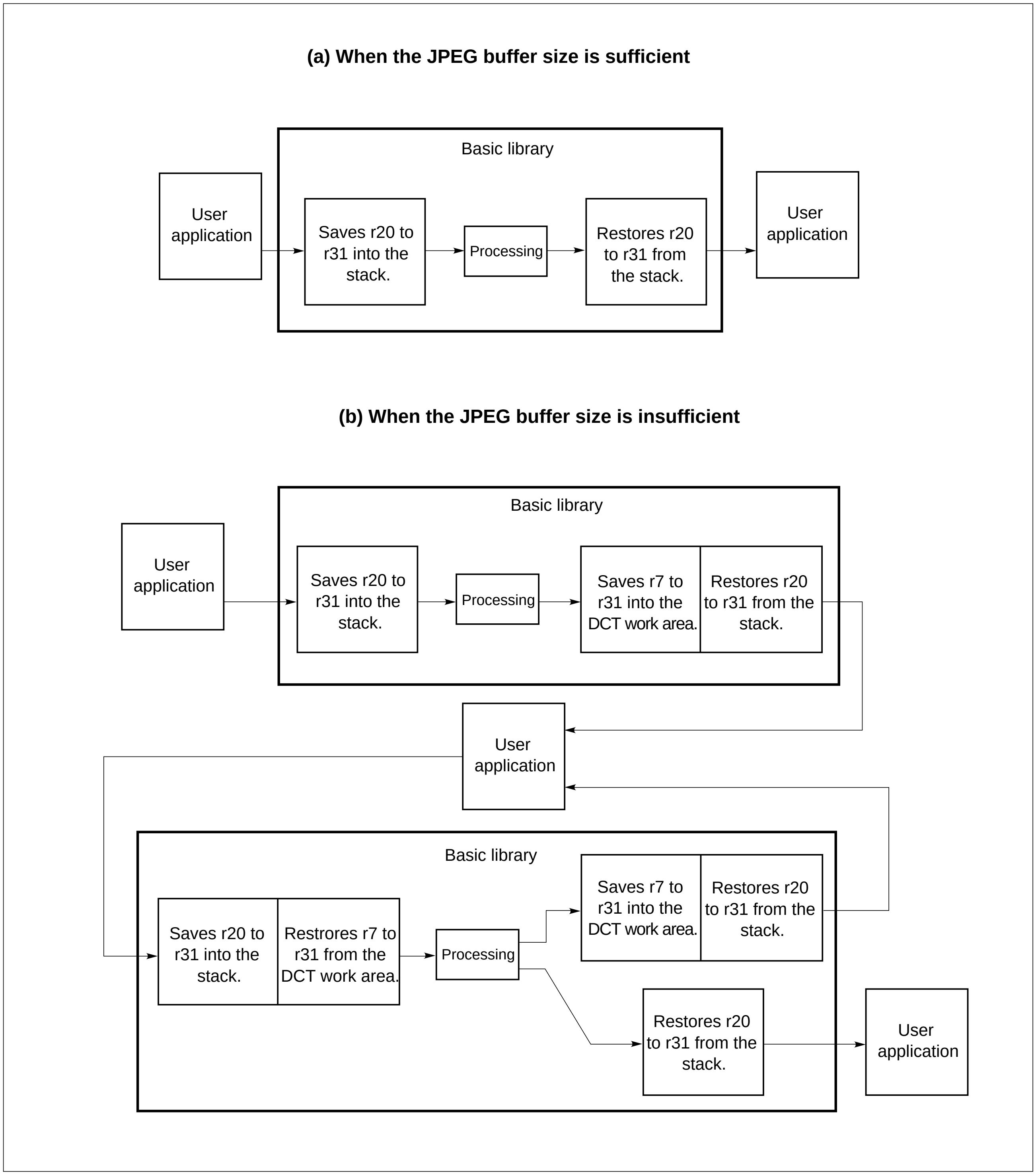
2.3.6 Register Dispatch

The basic library stops processing and transfers control to the user application in the following cases. At this time, the contents of the registers used by the basic library are saved into memory, and the contents of the registers (sp and r20 to r29) which are saved according to the C conventions, are restored.

Register dispatch takes place in the following cases:

- If the JPEG buffer becomes full as a result of compression
- If the JPEG buffer is decoded to the end as a result of expansion
- If it is specified that processing is to be stopped at each image line, and the line at which processing is to be stopped is reached

Figure 2-10. Register Dispatch



2.4 EXECUTING COMPRESSION PROCESSING

Compression processing compresses image data to create a JPEG file.

2.4.1 Compression Main Function

Classification	Compression processing system
Function name	jpeg_Compress
Feature	JPEG compression processing
Format	#include "jpeg.h"
	int jpeg_Compress (CJINFO* cJpeginfo)
Argument	First address of CJINFO structure
Return value	The return value is a numeric value like that defined as #define JPEG_OK 0 in C.

Table 2-12. Return Values for Compression Processing Function

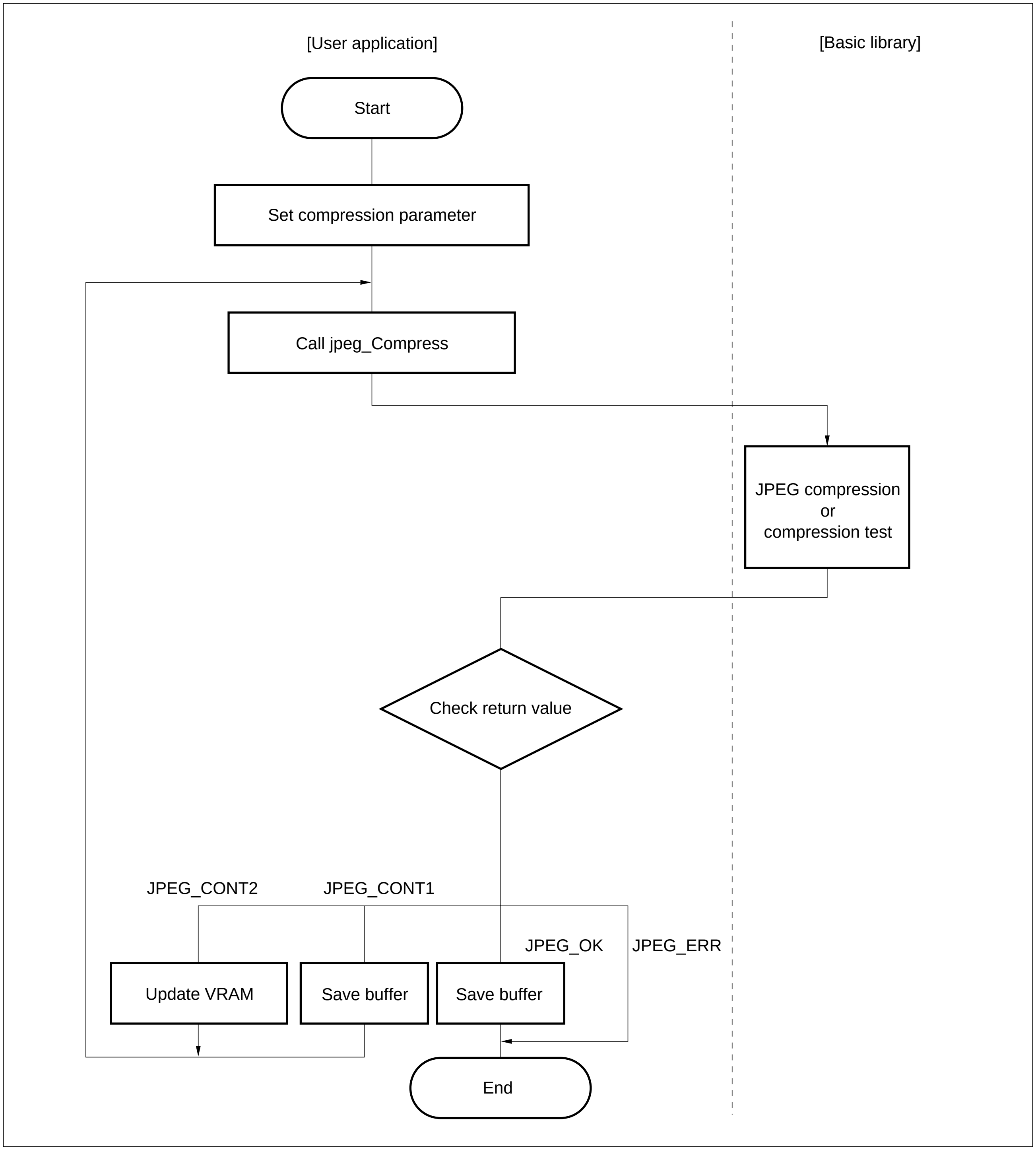
Return value	Description
JPEG_OK	Normal completion
JPEG_ERR	Error termination
JPEG_CONT1	Aborted by JPEG buffer
JPEG_CONT2	Aborted by VRAM

Remark For JPEG_ERR, an error statement is stored into member "ErrorState" of the CJINFO structure.

2.4.2 Compression Processing Flow

The flow of the compression processing is shown below.

Figure 2-11. Compression Processing Flow



2.4.3 Setting of CJINFO Structure Parameter

Figure 2-12. Setting of CJINFO Structure Parameter (AP70732-B03)

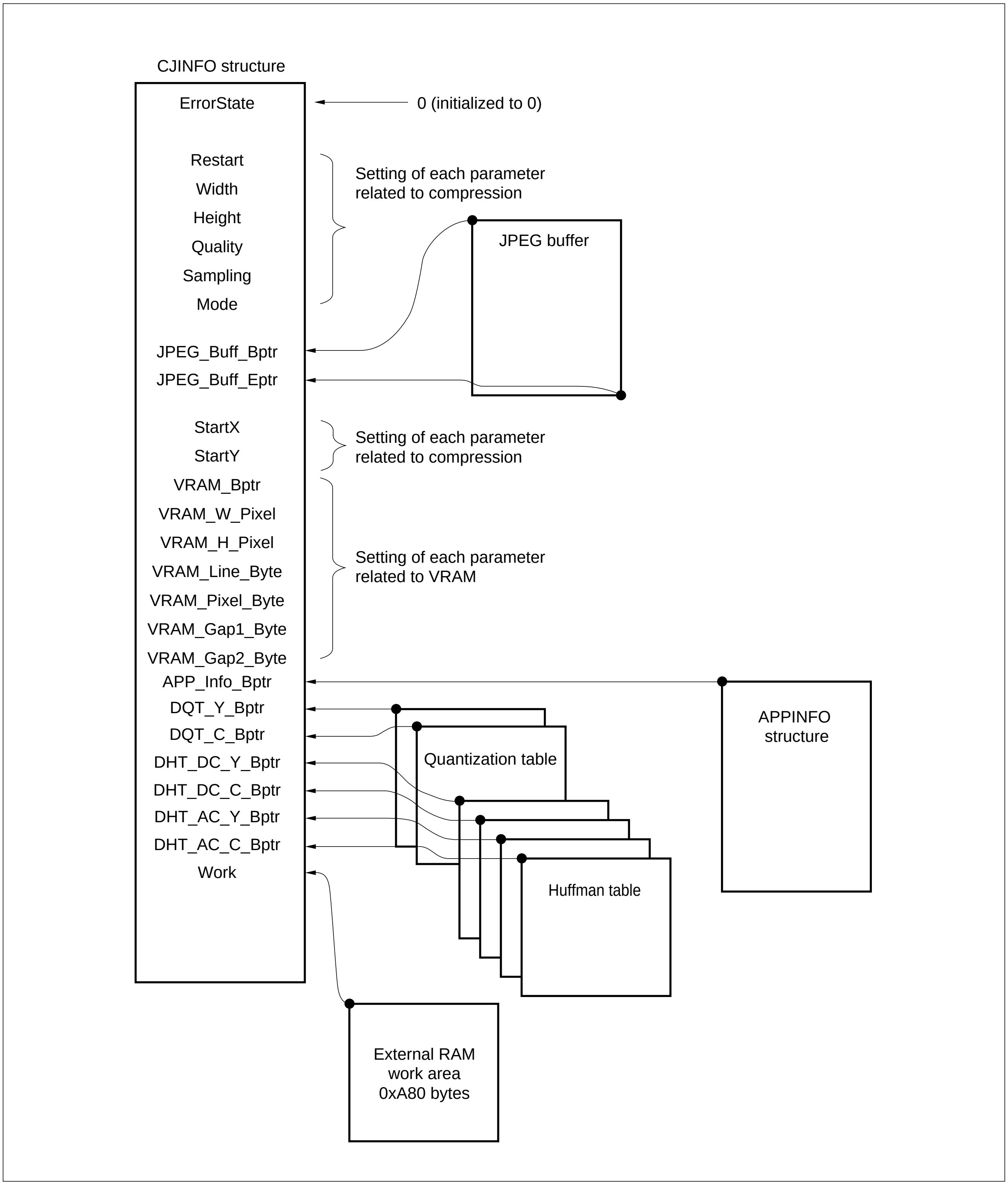
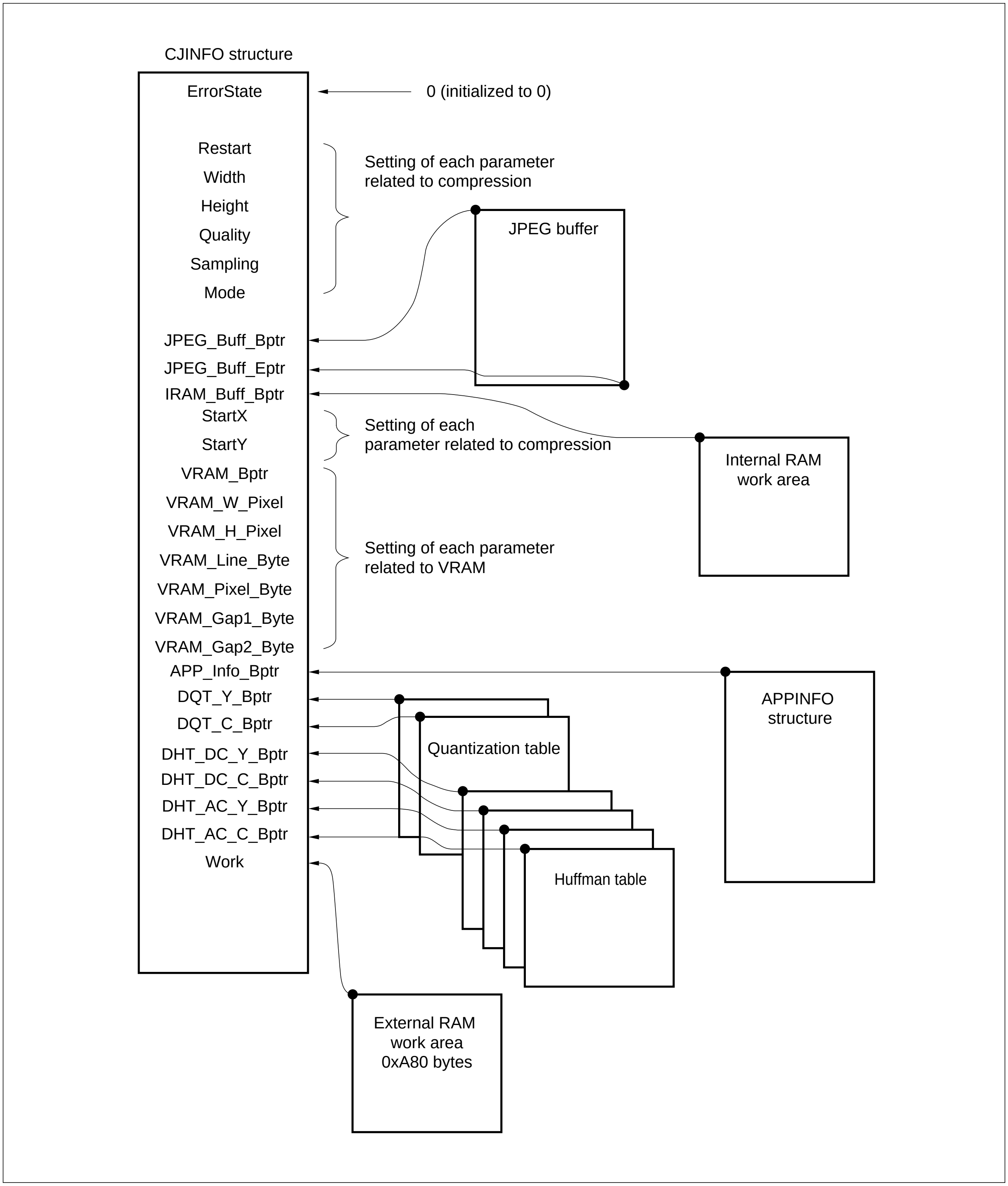


Figure 2-13. Setting of CJINFO Structure Parameter (AP705100-B03)



(1) Initialization of error status (ErrorState)

Initialize the error status to 0 once only, when compression parameters are set before the compression routine is started.

Set value: 0

Caution Do not perform any other initialization because, when processing is stopped then resumed, the basic library determines whether processing is being started for the first time or resumed by referring to this ErrorState value (if the processing is stopped, the address from which the processing is to be resumed is stored).

(2) Restart interval (Restart)

For details of the restart interval, see **(7)** in **Section 1.2.1**.

Set value: 0 to 65535

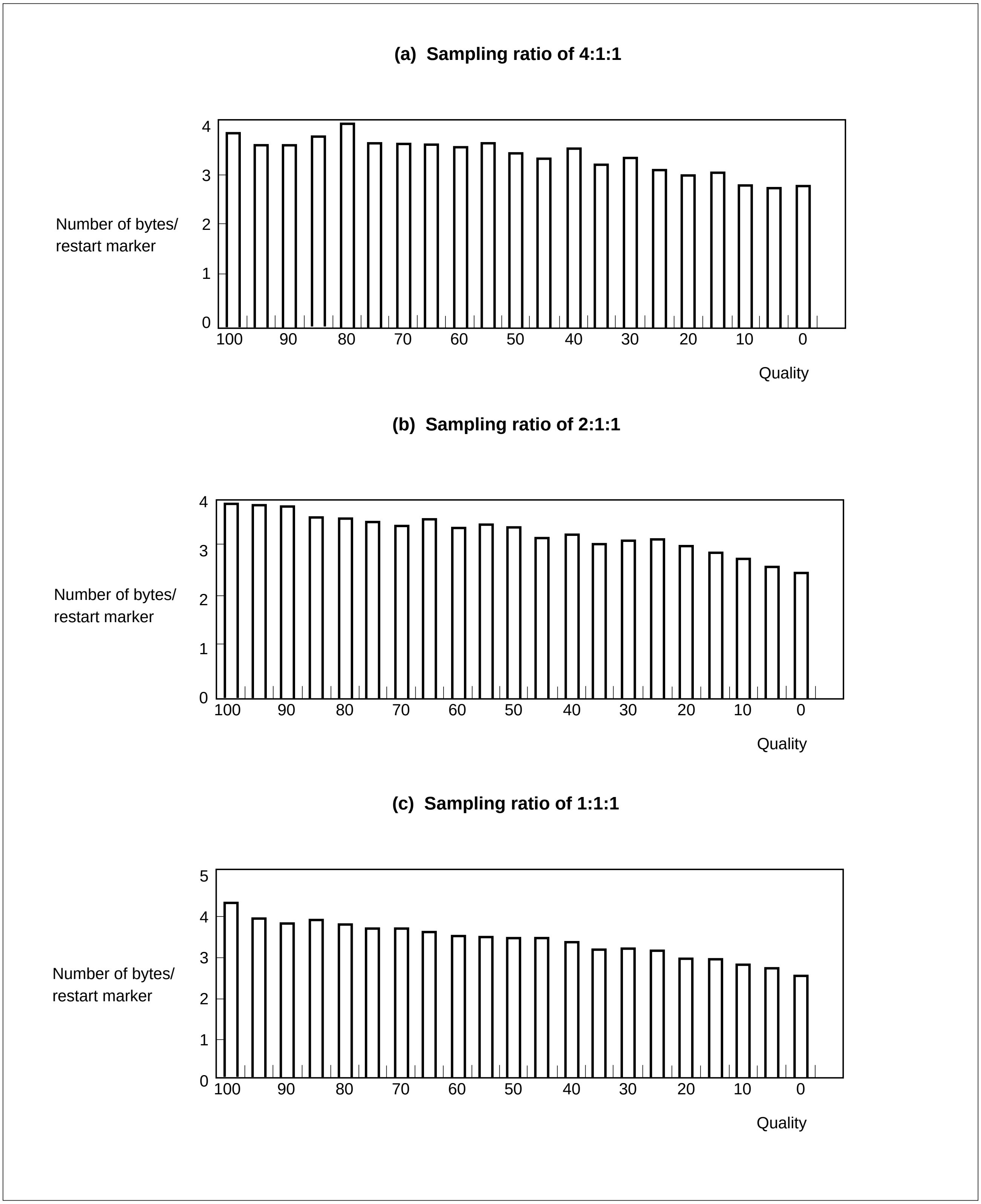
When 0 is specified, the DRI segment/RSTn marker is not inserted. If a value other than 0 is specified, that value is used as the restart interval.

Table 2-13. Setting of Restart Interval

Set value	Processing
0	DRI segment and RSTn marker are not appended to the JPEG file.
1 to 65,535	Uses the set value as the restart interval and inserts RSTn marker as many times as the number of MCUs specified by this value.

If the restart interval is valid, the size of the JPEG file is increased by the RSTn marker. One restart marker is a little less than 4K bytes. To determine the approximate value of the file size, add the file size compressed without the restart marker, multiplied by the number of RSTn included in one file, to this value. The average number of bytes per RSTn marker is shown below.

Figure 2-14. Quantization Parameter and Number of Bytes Required for Each Restart Marker



(3) Width and height of an image

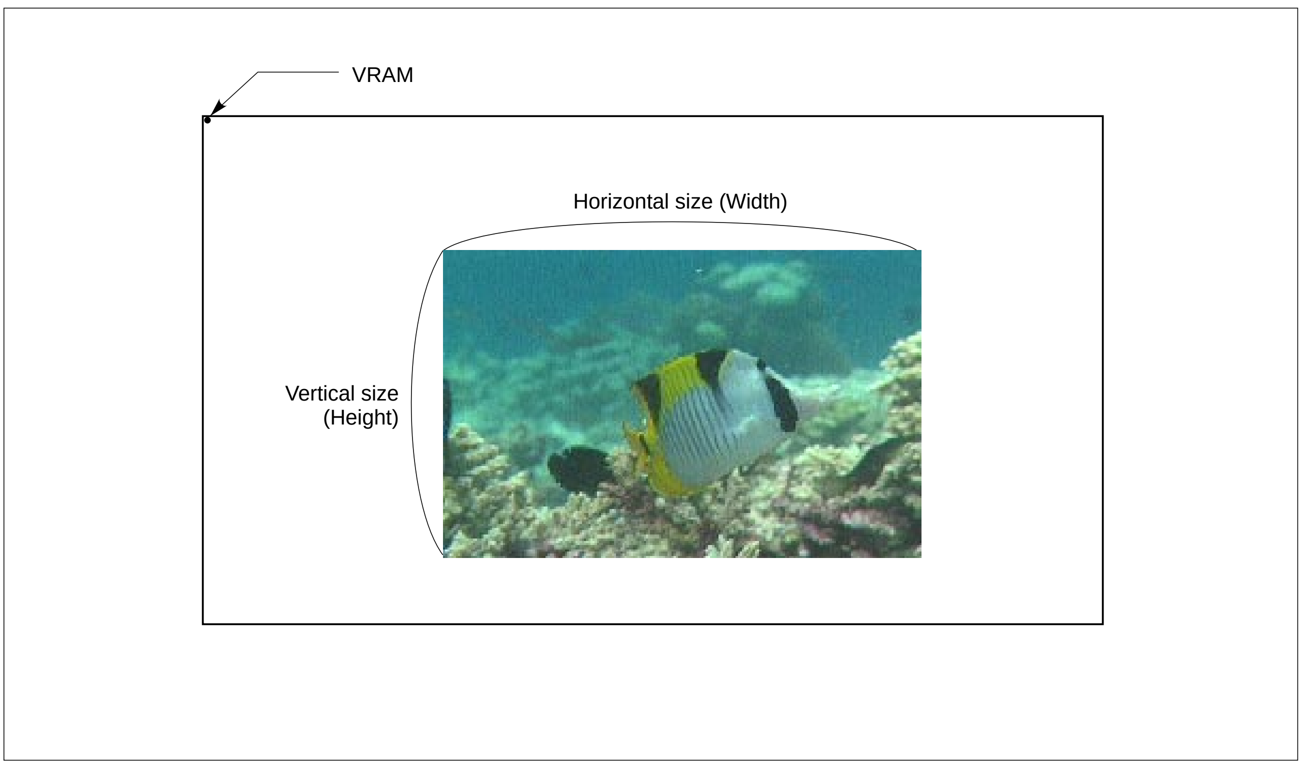
Set value: 0 to 65,535

The unit of the value is the number of pixels.
However, the value that can be set is limited as follows:

Table 2-14. Limit on Horizontal Size/Vertical Size

Sampling ratio	Horizontal size (width)	Vertical size (height)
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
2:1:1 (H:V = 2:1)	Multiple of 16	Multiple of 8
1:1:1 (H:V = 1:1)	Multiple of 8	Multiple of 8

Figure 2-15. Horizontal and Vertical Sizes of an Image



(4) Quantization parameter

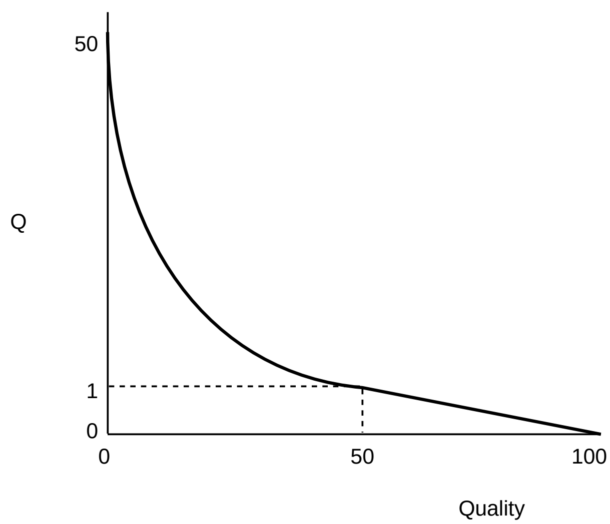
A quantization parameter (Quality) is provided in the basic libraries to enable the values in the quantization tables to be easily changed.

The basic libraries determine constant "Q" from the value of the Quality parameter according to the formulas shown below. Each element in the quantization tables, multiplied by Q (thus being rounded to between 1 and 255), is used as a quantization factor for actual quantization.

When Quality is less than 50: Q = Quality/50

When Quality is greater than or equal to 50: Q = 2 - Quality/50

Figure 2-16. Quantization Parameter "Quality" and Constant "Q"



To use the default quantization tables as is, specify 50 for the Quality parameter.

Table 2-15. Quality Parameter Settings

Quality parameter	100	…	50	…	0
Constant Q	0	…	1	…	50
Quantization table	All elements are 1.	…	Same as default	…	Most elements are 255.
Image quality	Excellent		…		Poor
JPEG file size	Large		…		Small

When Quality is set to 100 or 75 for the default quantization tables LuminanceQtbl and ChrominanceQtbl, the following quantization tables are generated and used for actual compression (quantization).

(a) When Quality is set to 100

Quantization table for luminance component

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Quantization table for chrominance component

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(b) When Quality is set to 75

Quantization table for luminance component

8	6	5	8	12	20	26	31
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	28
7	9	11	15	26	44	40	31
9	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	46	48	49	56	50	52	50

Quantization table for chrominance component

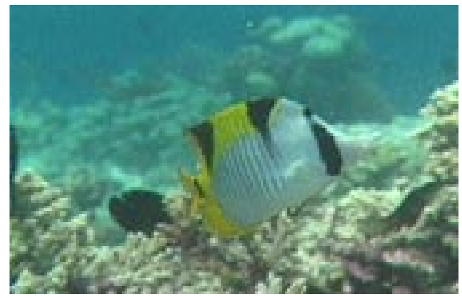
9	9	12	24	50	50	50	50
9	11	13	33	50	50	50	50
12	13	28	50	50	50	50	50
24	33	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Figure 2-17 illustrates how the appearance of the JPEG-compressed image varies depending on the value specified for the Quality parameter.

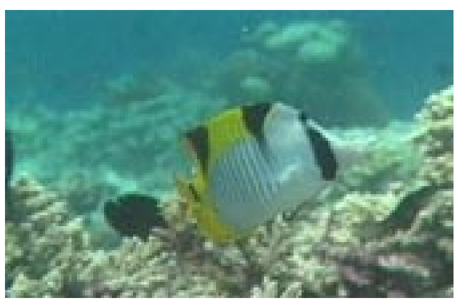
Figure 2-17. Variation in Image Quality Depending on Value of Quantization Parameter (1/2)

Source image
224 x 144 x 24

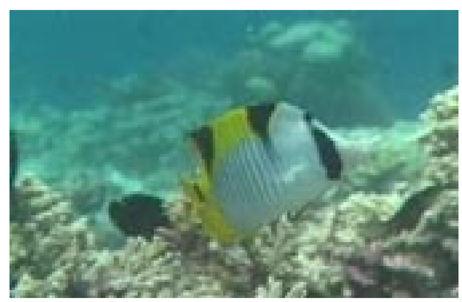
(a) Quality = 100, sampling ratio: 4:1:1,
file size: 35,714 bytes



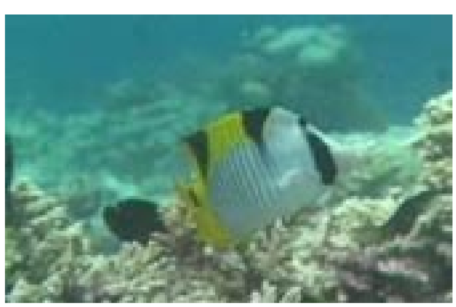
(b) Quality = 90, sampling ratio: 4:1:1,
file size: 10,424 bytes



(c) Quality = 80, sampling ratio: 4:1:1,
file size: 6,615 bytes



(d) Quality = 70, sampling ratio: 4:1:1,
file size: 5,195 bytes



(e) Quality = 60, sampling ratio: 4:1:1,
file size: 4,364 bytes

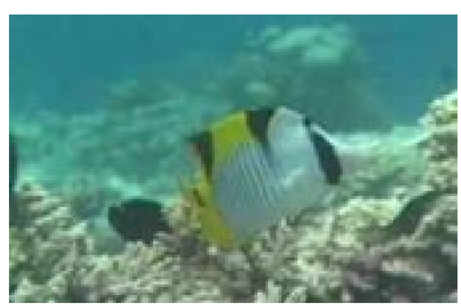
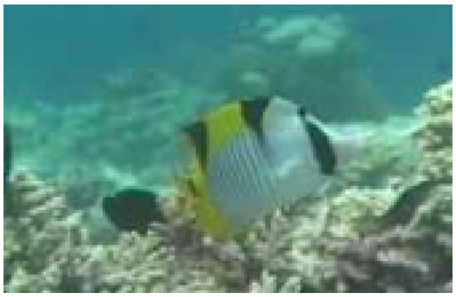
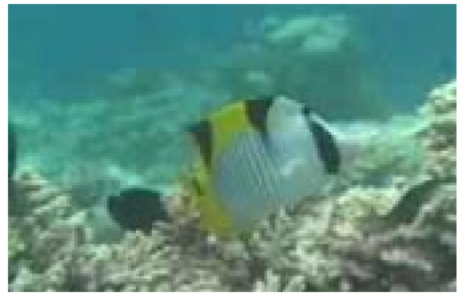


Figure 2-17. Variation in Image Quality Depending on Value of Quantization Parameter (2/2)

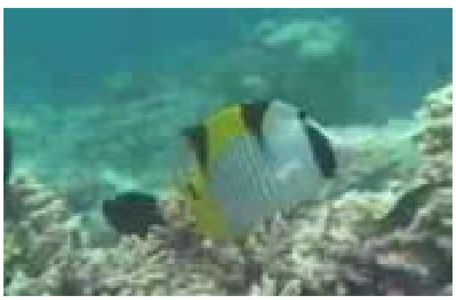
(f) Quality = 50, sampling ratio: 4:1:1,
file size: 3,869 bytes



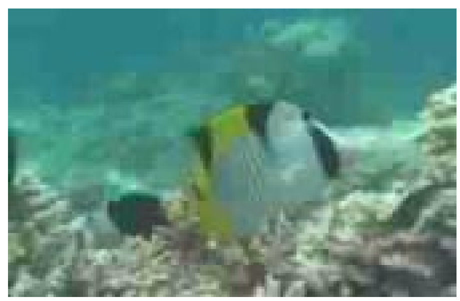
(g) Quality = 40, sampling ratio: 4:1:1,
file size: 3,388 bytes



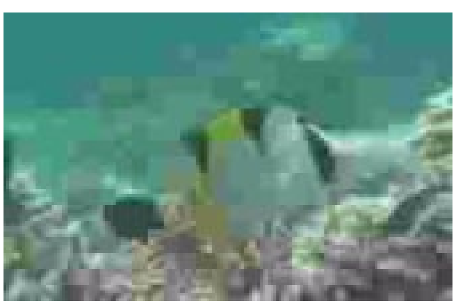
(h) Quality = 30, sampling ratio: 4:1:1,
file size: 2,915 bytes



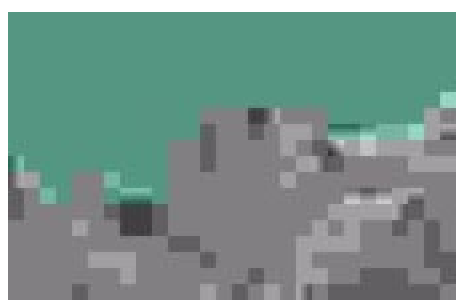
(i) Quality = 20, sampling ratio: 4:1:1,
file size: 2,335 bytes



(j) Quality = 10, sampling ratio: 4:1:1,
file size: 1,701 bytes

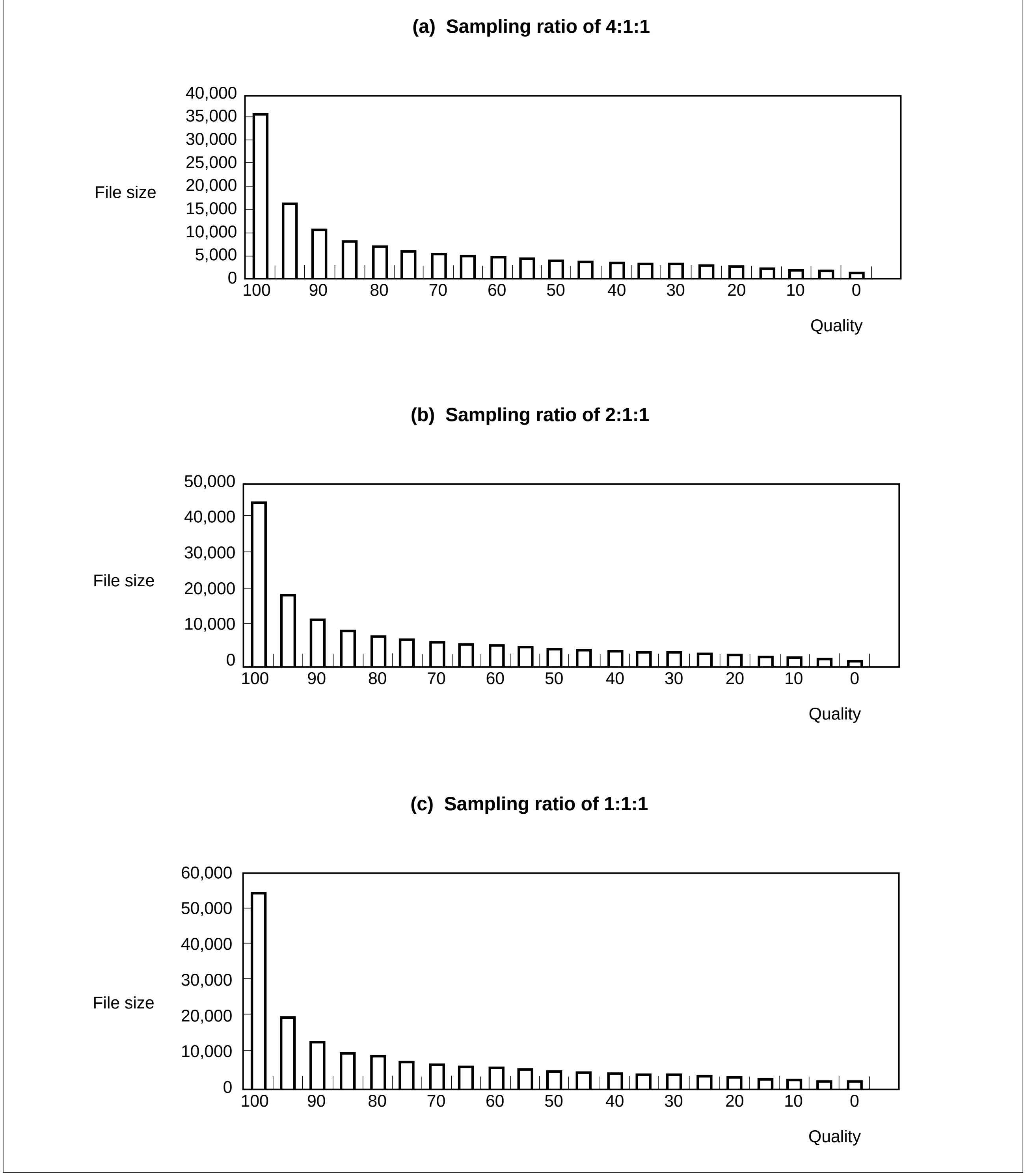


(k) Quality = 0, sampling ratio: 4:1:1,
file size: 1,228 bytes



The relationship between the quantization parameter and file size is as shown in Figure 2-18.

Figure 2-18. Quantization Parameter and File Size



(5) Selecting sampling ratio (Sampling)

For the sampling ratio, see **(3)** in **Section 1.2.1**.

The basic library supports the following four types of sampling ratios.

Table 2-16. Set Value of Member Sampling

Sampling ratio	Identification in basic library
4:1:1 (H:V = 2:2)	SAMPLE22
4:1:1 (H:V = 4:1)	SAMPLE41
2:1:1 (H:V = 2:1)	SAMPLE21
1:1:1 (H:V = 1:1)	SAMPLE11

These values are defined in file jpeg.h.

Table 2-17. Setting of Sampling Ratio

Sampling ratio	4:1:1	2:1:1	1:1:1
Color	Normal	↔	Clear
File size	Reference value	About 4/3 times	About 2 times

For the luminance component, there is no difference in the image regardless of which sampling ratio is selected. The sampling ratio influences the image quality of the chrominance component.

(6) Mode (Mode)

Set values and the corresponding operations or modes are shown below.

Table 2-18. Set Values for Member Mode

7 6	5	4	3 0
RFU	HF	QF	MD

Bit	Bit name	Description
7 to 6	RFU	Reserved field^{Note}
5	HF	Huffman table initialization flag[Note] 0: Initializes the tables. 1: Does not initialize the tables.
4	QF	Quantization table initialization flag[Note] 0: Initializes the tables. 1: Does not initialize the tables.
3 to 0	MD	Mode 0: Compression test mode 1: Normal compression mode

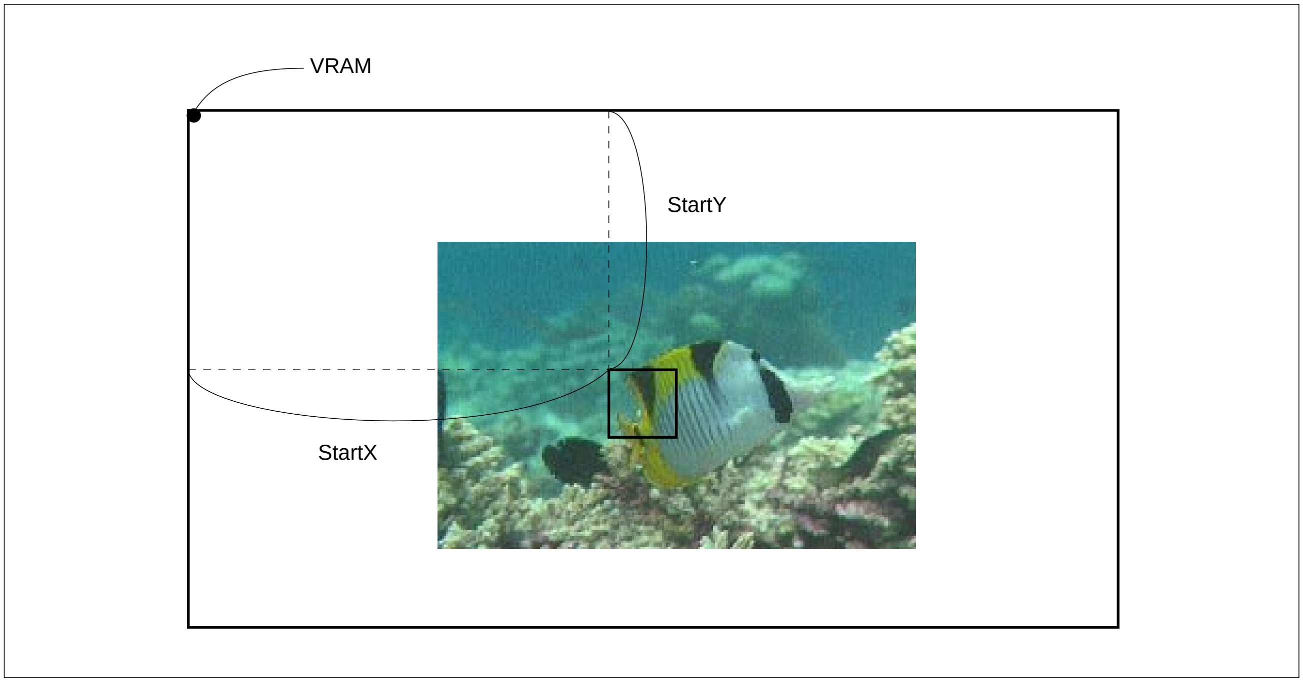
Note Added in ver. 2.10.

(a) Mode 0

This mode is used to test the number of bits of the compressed data of one MCU. The number of bits can be obtained using member FileSize.

To shift the position of an MCU, change the values of members StartX and StartY.

Figure 2-19. Adjustment of Compression Test Position



(b) Mode 1

Normal compression processing is performed.

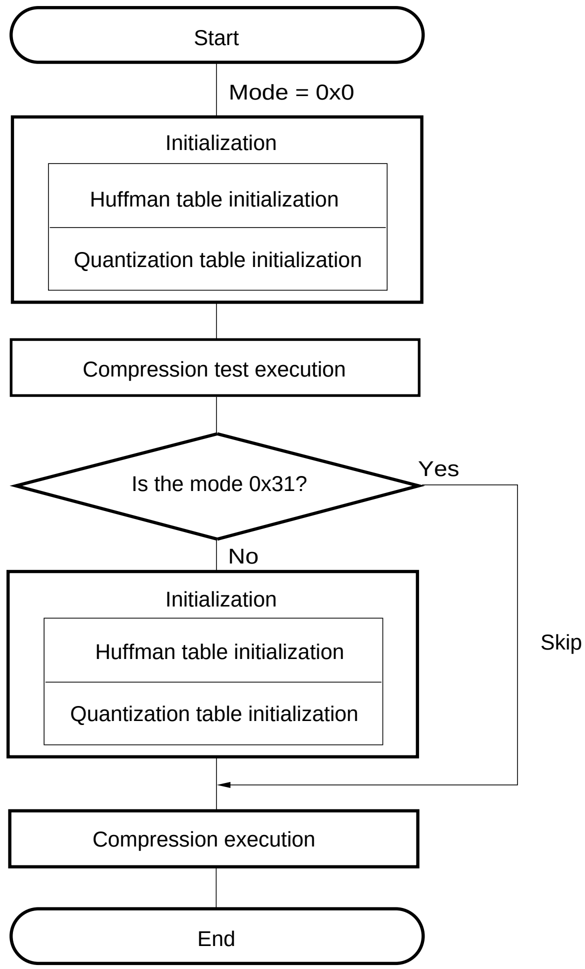
The quantization look-up table and Huffman look-up table used internally by the basic libraries are loaded into the 2,688-byte area that is specified by the member Mode in the structure.

When using a work area that has already been used for compression (when these look-up tables are initialized), the tables need not be initialized again.

If the HF bit of member Mode in the structure is set to 1, the Huffman look-up tables are not created. Similarly, the quantization look-up tables are not changed, if the QF bit is set to 1.

Figure 2-20 shows an example, in which the look-up tables are initialized in tentative compression mode, and not initialized before the normal compression is performed with the same quantization parameter and Huffman tables.

Figure 2-20. Renewed Compression Mode Setting



(7) First address (JPEG_Buff_Bptr) and end address (JPEG_Buff_Eptr) of JPEG buffer

Set the first address and end address of the JPEG buffer.

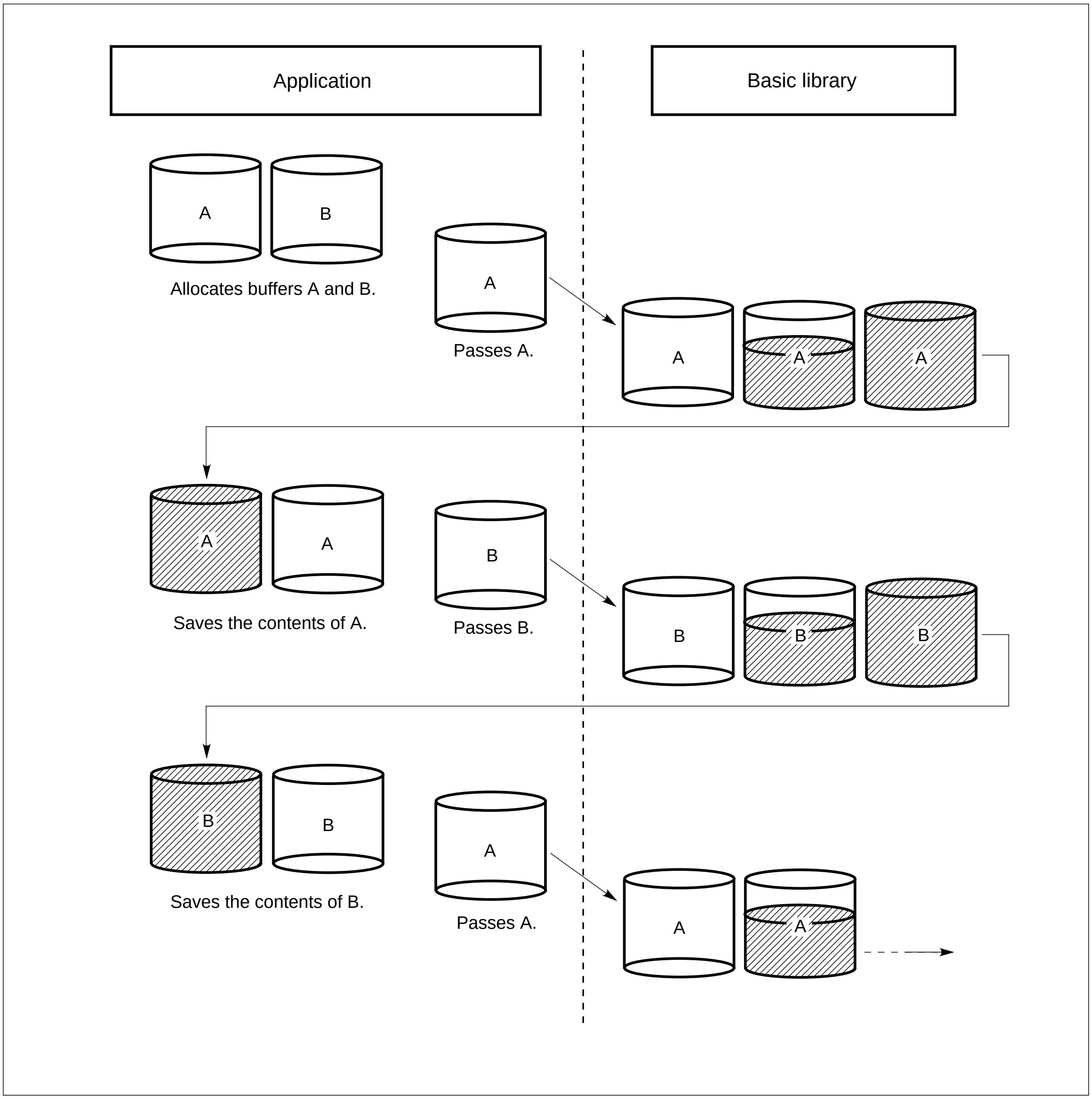
Table 2-19. Set Values for Members JPEG_Buff_Bptr/JPEG_Buff_Eptr

Member	Description
JPEG_Buff_Bptr	First address of JPEG buffer
JPEG_Buff_Eptr	First address of JPEG buffer + JPEG buffer size

If buffer save processing is performed in the middle of processing due to the limit on the JPEG buffer size, two JPEG buffers can be used alternately.

For details of the JPEG buffer, see **Sections 2.3.5** and **2.3.6**.

Figure 2-21. Switching Between Two JPEG Buffers



(8) Internal RAM work area address (IRAM_Buff_Bptr: AP705100-B03)

For details of the internal RAM work area, see **Section 2.3.4**.

Table 2-20. Set Value for Member IRAM_Buff_Bptr

Member	Description
IRAM_Buff_Bptr	First address of internal RAM work area

0x400/0x300/0x280 bytes from the first address are unconditionally overwritten at a sampling ratio of 4:1:1/2:1:1:/1:1:1.

This member need not be set with the AP70732-B03 (V810 family version).

Table 2-21. Sampling Ratio and Size of Required Internal RAM Work Area

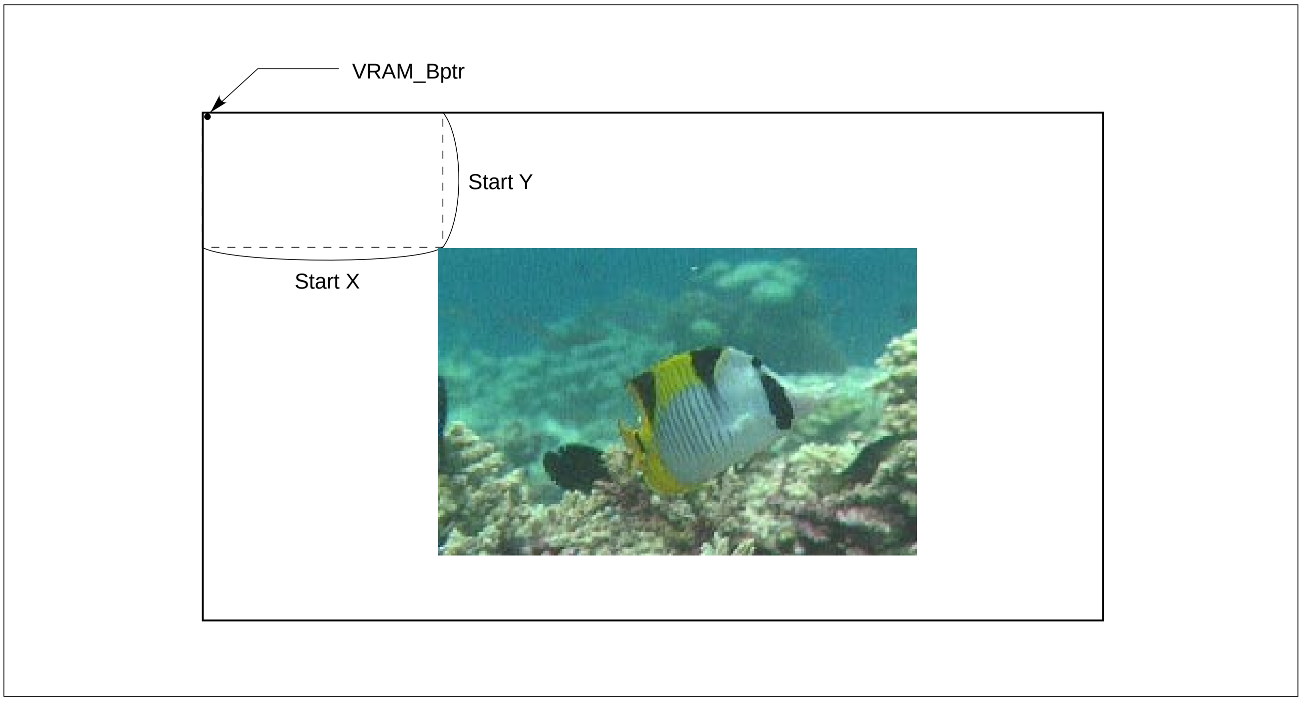
Sampling ratio	Size of required internal RAM work area
4:1:1 (H:V = 2:2)	0x400 bytes
4:1:1 (H:V = 4:1)	0x400 bytes
2:1:1 (H:V = 2:1)	0x300 bytes
1:1:1 (H:V = 1:1)	0x280 bytes

(9) Image start positions x (StartX) and y (StartY)

Set value: -32,768 to 32,767

The unit of the value is the number of pixels.

Figure 2-22. Start Point of an Image (x, y)



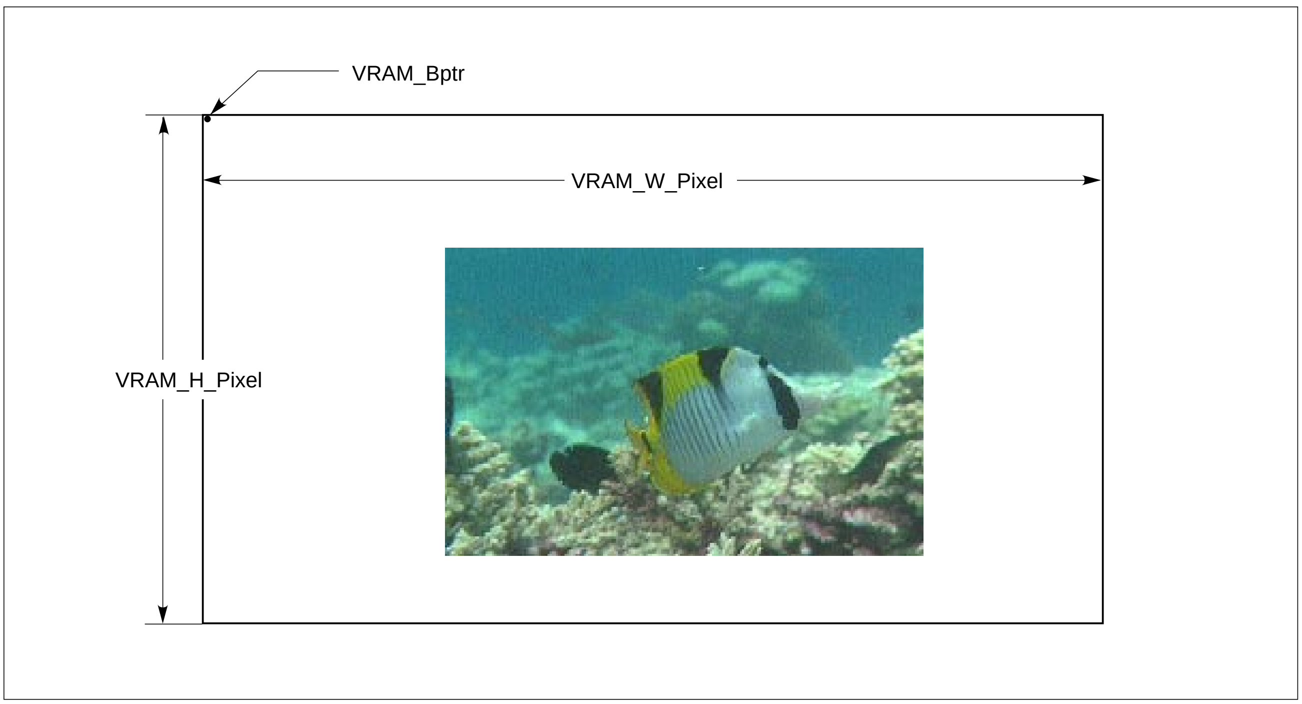
(10) VRAM size

Set the members related to VRAM.

Table 2-22. Set Values for Members Related to VRAM

Member	Description
VRAM_Bptr	VRAM first address (reference address)
VRAM_W_Pixel	Number of horizontal pixels of VRAM
VRAM_H_Pixel	Number of vertical pixels of VRAM

Figure 2-23. VRAM Size



When the compression library is executed, the following two points are checked when the header is created.

- Relation between sizes of VRAM_W_Pixel and (StartX+Width)
- Relation between sizes of VRAM_H_PIxel and (StartY+Height)

Even when customizing the VRAM access part (described below), set the values of members VRAM_W_Pixel and VRAM_H_Pixel (specify size by which the check routine is not terminated by an error). The value of VRAM_Bptr may be undefined when customizing the VRAM access part.

(11) VRAM configuration

The following values are referenced when the default VRAM access routine is used.

The default VRAM access routine assumes that VRAM has a depth of 256 tones (1 byte) of Y/Cb/Cr or R/G/B, and that the VRAM can be accessed by the LD.B/ST.B instruction.

Table 2-23. Set Values for Members Related to VRAM Configuration

Member	Description
VRAM_Line_Byte	Address difference of VRAM of vertical 1 pixel
VRAM_Pixel_Byte	Address difference of VRAM of horizontal 1 pixel
VRAM_Gap1_Byte	If VRAM is YCbCr, address difference between Y and Cb of same pixel. If VRAM is RGB, address difference between R and G of same pixel.
VRAM_Gap2_Byte	If VRAM is YCbCr, address difference between Y and Cr of same pixel. If VRAM is RGB, address difference between R and B of same pixel.

Figure 2-24. VRAM Configuration

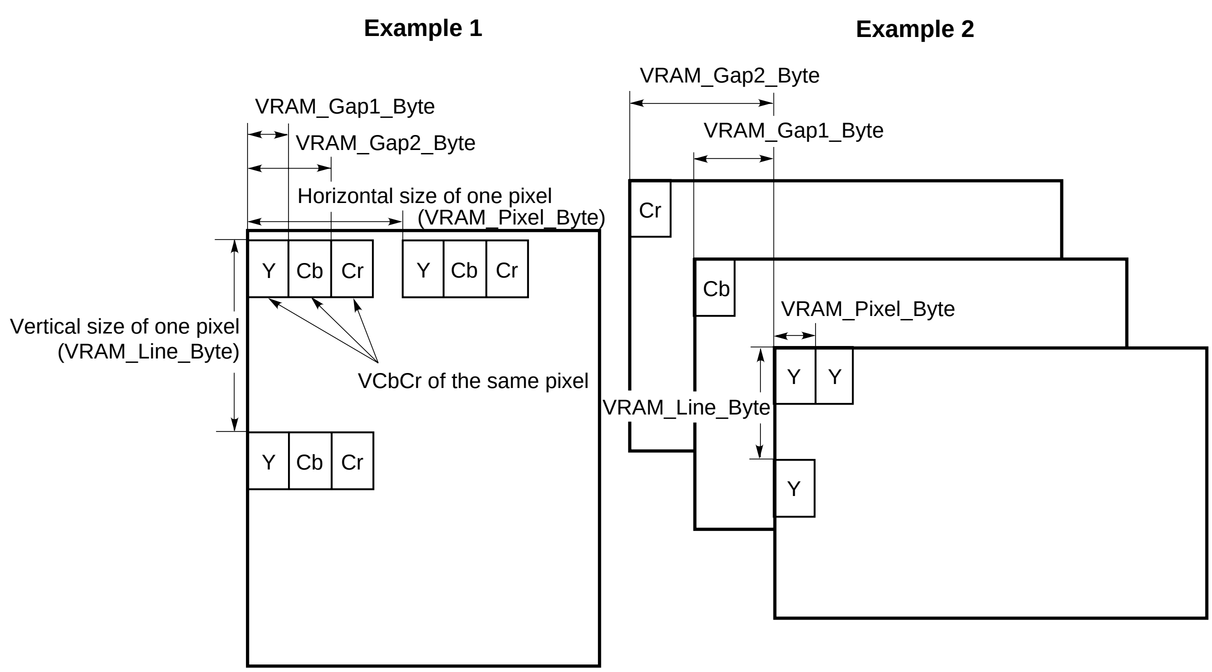
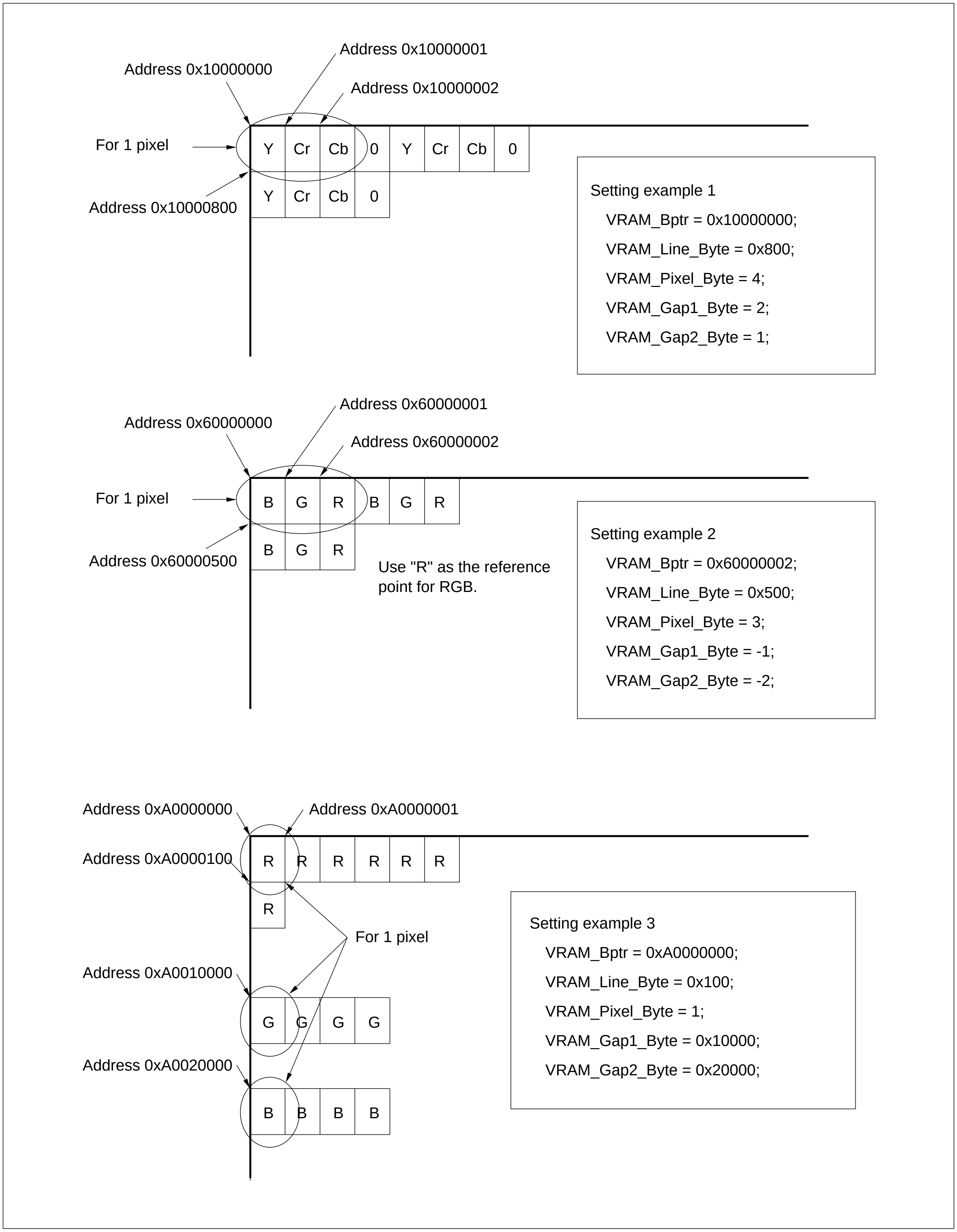


Figure 2-25. Example of Setting VRAM-Related Members of Basic Library



(12) Specification of APPINFO table (APP_Info_Bptr)

With the basic library, the embedding of an application data segment can be specified. If data is not embedded in the APPn segment, set APP_Info_Bptr to 0. At this time, the APPINFO structure is not required.

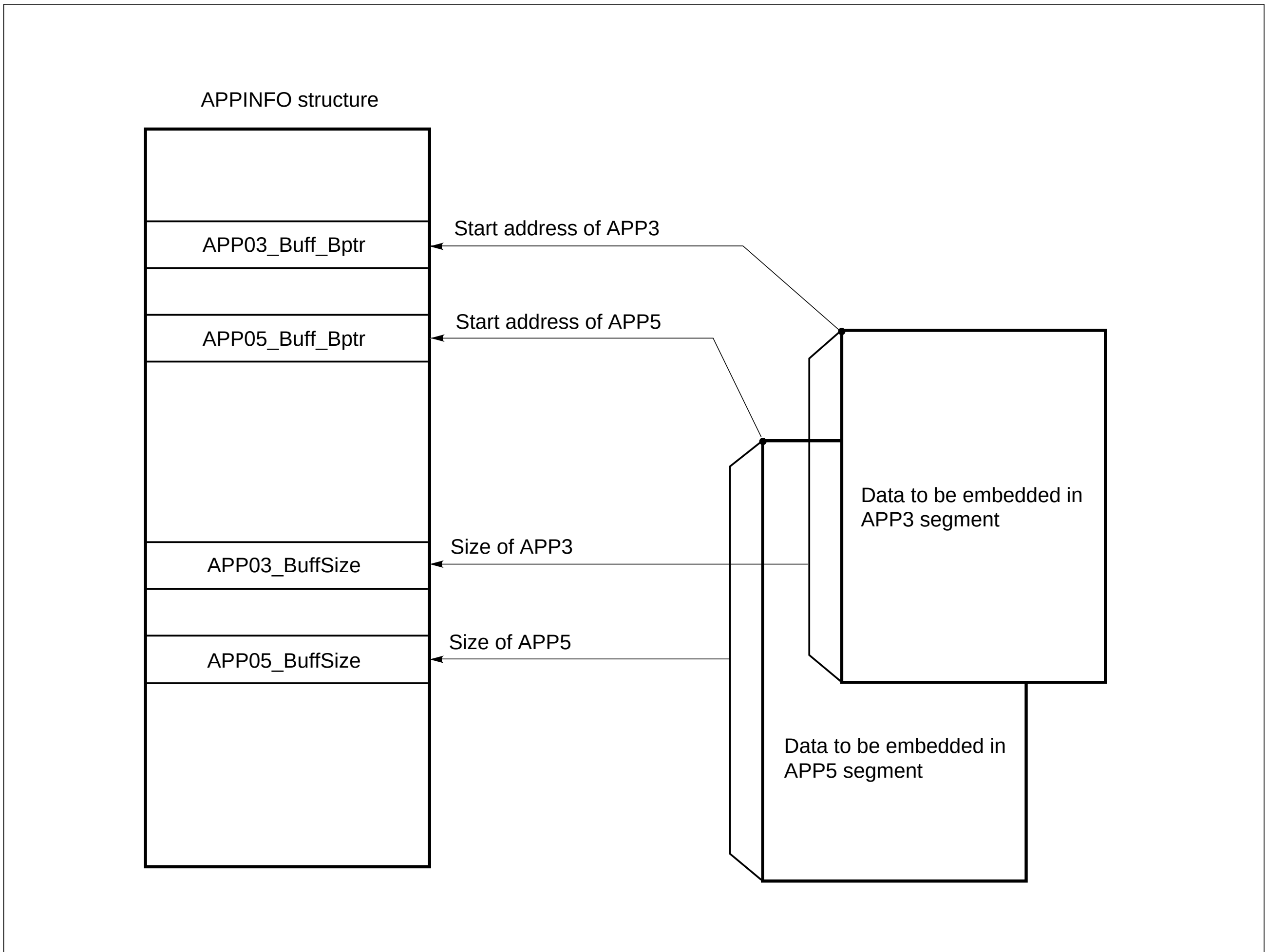
Table 2-24. Set Value for Member APP_Info_Bptr

Member	Set value
APP_Info_Bptr	0: APPn segment is not embedded
	First address of APPINFO structure: APPn segment is embedded

Caution If the APPINFO structure is placed at address 0, it is assumed that the APPINFO structure is set.

To embed the APPn segment, register the first address of the buffer storing the data to be embedded in the member corresponding to the APPn segment number used, and the size of the data in the member of the APPINFO structure.

Figure 2-26. APPINFO Structure Settings for Compression



(13) Comment marker

unsigned char jpeg_COMStr[]="This is a Comment Marker"; and the part defining a comment marker character string can be exchanged.

(14) Quantization table

Specify a 64-byte quantization table for each of the luminance and chrominance components. Each table consists of 64 elements where each element consists of 1 byte.

Table 2-25. Setting of Quantization Table

Member	Description
DQT_Y_Bptr	Quantization table for luminance component
DQT_C_Bptr	Quantization table for chrominance component

Specify the following name to use the table prepared by the library.

For luminance component:	LuminanceQtbl
For chrominance component:	ChrominanceQtbl

Default quantization table for luminance component

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Default quantization table for chrominance component

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(15) Huffman table

Specify the four Huffman tables (for DC and AC luminance components and DC and AC chrominance components) in the form of a DHT segment.

Table 2-26. Setting of Huffman Table

Member	Description
DHT_DC_Y_Bptr	Huffman table for luminance DC
DHT_DC_C_Bptr	Huffman table for chrominance DC
DHT_AC_Y_Bptr	Huffman table for luminance AC
DHT_AC_C_Bptr	Huffman table for chrominance AC

Specify the following name to use the table prepared by the library.

For luminance component DC:	DHT_markerLuminanceDC
For luminance component AC:	DHT_markerLuminanceAC
For chrominance component DC:	DHT_markerChrominanceDC
For chrominance component AC:	DHT_markerChrominanceAC

(16) External RAM work area address (Work)

Set the first address of the external RAM work area.

Table 2-27. Setting of Member Work

Member	Description
Work	First address of external RAM work area of 0xA80 bytes

2.4.4 Setting a Comment Marker

The following explains how to choose whether to embed a comment marker, and how to set a character string to be embedded.

The character string is an ASCII code string that ends with a NULL character (0x00).

(1) When a comment marker is not to be embedded

Specify either of the following descriptions on the side calling the library.

- unsigned char jpeg_COMStr [] = "\0";
- unsigned char jpeg_COMStr [] = {0};

(2) To embed a character string as the comment marker

To embed the character string "ABCDE" as the comment marker, specify the following. (When C is used, 0x00 is appended to the character string automatically.)

```
unsigned char jpeg_COMStr[] = "ABCDE";
```

When assembly language is used, the user must append a NULL character, as follows.

```
        .text
        .align 4
        .globl_jpeg_COMStr
_jpeg_COMStr:
        .str " ABCDE\0"
```

If, however, jpeg_COMStr[] = "V830" is specified, it holds a special meaning as described in (c).

(3) To embed binary code as the comment marker

To embed the following code as the comment marker, follow steps **<1>** to **<3>** below.

"This\0is\0comment\0including\0null\0character"

<1> Specify the four-letter key word "V830" for jpeg_COMStr.

<2> Cast the four bytes of CJInfo.IR [0] into int type, and specify the number of bytes of the code to be embedded.

<3> Cast the four bytes of CJInfo.IR [4] into unsigned char* type, and specify the first pointer of the code to be embedded.

An example of setting is described below.

```
unsigned char jpeg_COMStr[] = "V830";
unsigned char
  jpet_COMBuff[] = "This\0is0commernt\0including\0null\0character";

void compress_parameter_ini ()
{

Omission

        :
  :
  *(int*)&(CJinfo.IR[0]) + 40 ;/*length of COM (bytes)*/
  *(unsigned char**)&(CJinfo.IR[4]) = jpeg_COMBuff;/*address*/
}
```

When performing multiple compression tasks simultaneously using the multi-tasking function of the OS, follow steps **<1>** to **<3>** above to embed different comment markers.

2.4.5 DHT Segment, DQT Segment

In the general JPEG format, the header includes two quantization tables as the DQT segment, and four Huffman tables as the DHT segment.

The JPEG standard (ISO/IEC 10918) permits these tables to be described separately or together. Examples of the description are shown below.

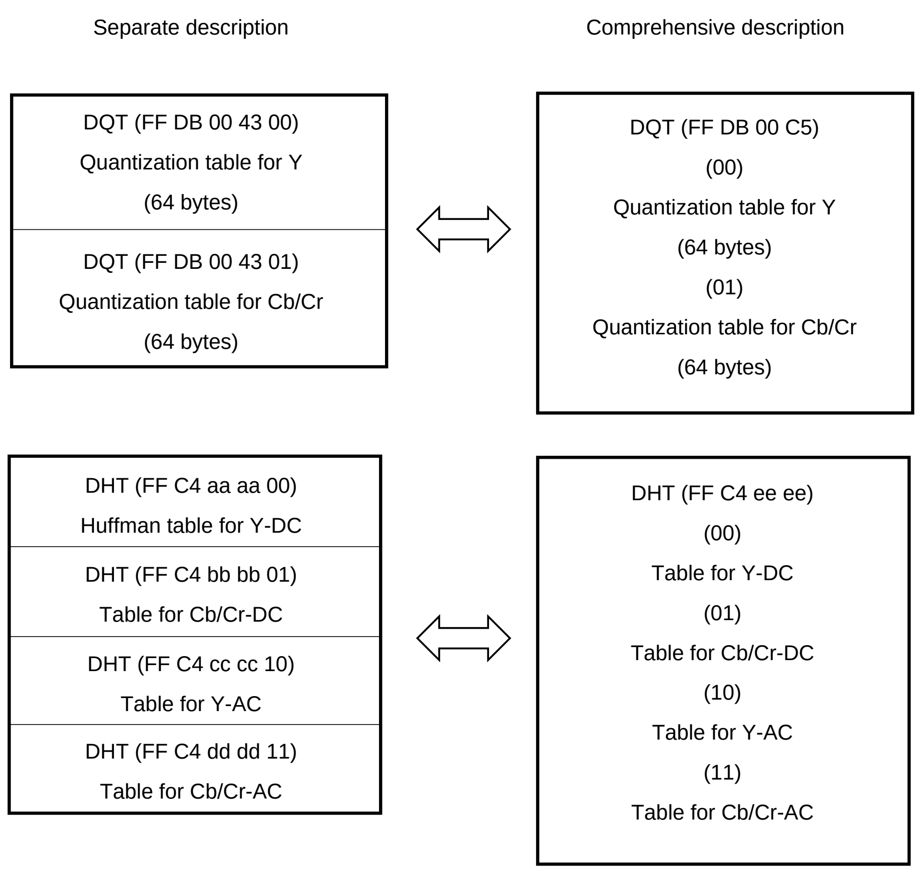
(a) Describing two 64-byte tables separately

- 0xFF, 0xDB, (segment length (2 bytes)), (table number), 64-byte table
- 0xFF, 0xDB, (segment length (2 bytes)), (table number), 64-byte table

(b) Describing two 64-byte tables together

- 0xFF, 0xDB, (segment length (2 bytes)), (table number), 64-byte table, (table number), 64-byte table

With the AP705100-B03 and AP70732-B03 basic libraries, the DQT and DHT segments are described separately, when compression is performed in a general way.



To describe the DQT and DHT segments together using the AP705100-B03 or AP70732-B03 basic libraries, follow the procedure below.

Cast the four bytes of CJInfo.IR [8] into unsigned char* type, and specify the two-letter key word "Ex."

Example of description: *(unsigned char**)&(CJInfo.IR[8]) = "Ex";

2.4.6 Limitations when Huffman Table Is Created by User

With the AP705100-B03 and AP70732-B03 basic libraries, the Huffman tables used for compression can be exchanged. However, this does not mean that any table can be used for exchange. If an inappropriate Huffman table is specified, some images may not be compressed normally. Moreover, the compression routine of the AP705100-B03 and AP70732-B03 basic libraries will terminate normally (return value: JPEG_OK) even in such a case.

To avoid this, observe the following two points when you create Huffman tables.

- The contents of L1 through L16 of the new Huffman tables must match logically.
- V1 through Vm of the new Huffman tables must contain categories up to 11 for the DC component and up to 10 for the AC component.

Figure 2-27. DHT Segment

0xFF 0xC4	Lh	id	L1	L2	···	L16	V1	V2	···	Vm

Table 2-28. Value and Bit Length of DC/AC Component

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
-1,023 to -512, 512 to 1,023	10
-2,047 to -1,024, 1,024 to 2,047	11

(1) Portions L1 through L16 of the DHT segment indicate how many i-bit Huffman codes exist.
For example, suppose L1 through L16 assume the following values:

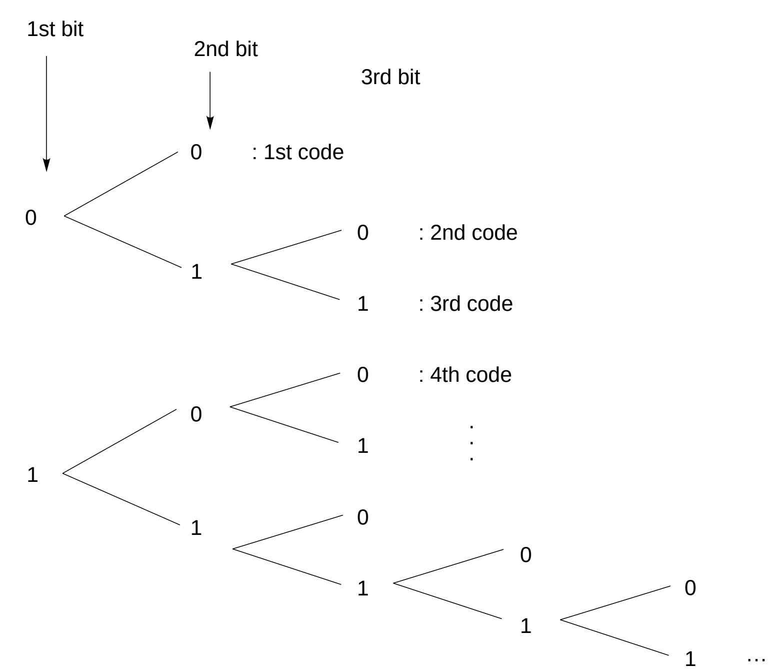
00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00

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

The values of the compressed codes are determined sequentially, starting from that having the shortest bit length, as shown in Figure 2-28.

Figure 2-28. Determining Values of Compressed Codes



L1 through L16 of the DHT segment have no meaning as data if they do not match the meaning of each element. For example, a combination that has three 1-bit values (L1 = 3) cannot exist. However, the AP705100-B03 and AP70732-B03 basic libraries do not check this. Therefore, use Huffman tables with values having logical meanings for L1 through L16.

(2) V1 through Vm of the DHT segment indicates the combination of a category and zero run to which each compressed code corresponds.
For example, suppose the values of V1 through Vm are as follows:

0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B (m = 12)

This indicates that each code is of the following category:

The first compressed code (00) is of category 0 (End of Block).
The second compressed code (010) is of category 1.
The third compressed code (011) is of category 2.
The fourth compressed code (100) is of category 3.
The fifth compressed code (101) is of category 4.
The sixth compressed code (110) is of category 5.
.
.
.
The 12th compressed code (111111110) is of category 11 (0xB).

Elements V1 through Vm are 0 through 0xB in the Huffman table for the DC component. Generally, the bit lengths of categories 2 and 1 are most widely distributed when an image is compressed. The closer to category 11, the lower the rate of appearance of the bit length. Depending on the image, bit lengths of categories 8, 9, 10, and 11 may not appear at all. In this case, the image is compressed and expanded normally even when a Huffman table from which the portions for category 8 or above are eliminated for V1 through Vm is used. If, however, an image in which the value of category 9 emerges is compressed by using a Huffman table that does not contain category 8 or above, the compression routines of the AP705100-B03 and AP70732-B03 embed 0, of 0 bits in length, into the compressed codes equivalent to category 9, and is normally terminated, interpreting that compressed codes are embedded even though no compressed codes are actually embedded. If a JPEG file created in this way is expanded, the position at which data of category 9 must appear and those that follow either cause an error or produce an image with a mosaic-like appearance.

AC coefficients have the same tendency as DC coefficients. For example, elements V1 through Vm are as follows in the Huffman table (jpeg_DHT_AC_Y) for the AC component supplied with the AP705100-B03 and AP70732-B03 basic libraries.

```
0x01, 0x02, 0x03, 0x00, 0x04, 0x11, 0x05, 0x12
0x21, 0x31, 0x41, 0x06, 0x13, 0x51, 0x61, 0x07
0x22, 0x71, 0x14, 0x32, 0x81, 0x91, 0xA1, 0x08
0x23, 0x42, 0xB1, 0xC1, 0x15, 0x52, 0xD1, 0xF0
0x24, 0x33, 0x62, 0x72, 0x82, 0x09, 0x0A, 0x16
0x17, 0x18, 0x19, 0x1A, 0x25, 0x26, 0x27, 0x28
0x29, 0x2A, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39
0x3A, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49
0x4A, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59
0x5A, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69
0x6A, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79
0x7A, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89
0x8A, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98
0x99, 0x9A, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7
0xA8, 0xA9, 0xAA, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6
0xB7, 0xB8, 0xB9, 0xBA, 0xC2, 0xC3, 0xC4, 0xC5
0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xD2, 0xD3, 0xD4
0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xE1, 0xE2
0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA
0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8
0xF9, 0xFA
```

The lower 4 bytes of each element indicate a category, while the higher 4 bytes indicate the zero run. 0x00 and 0xF0 are special codes indicating EOB (End of block) and ZRL (Zero run length), respectively. In the above example, the meanings are as follows:

The first compressed code is of zero run 0 and category 1.
The second compressed code is of zero run 0 and category 2.
The third compressed code is of zero run 0 and category 3.
The fourth compressed code is EOB.
The fifth compressed code is of zero run 0 and category 4.
The sixth compressed code is of zero run 1 and category 1.
.
.
.

As with DC coefficients, the lower the category, the higher the rate of appearance of the AC coefficient. The higher the category, the lower the rate of appearance. A zero run of 0 appears most frequently, while a zero run of 1 or more appears less frequently. Therefore, it is possible to create a Huffman table that does not have compressed codes corresponding to the portion with the higher zero run and category. With the AP705100-B03 and AP70732-B03, however, if compression is executed with such a table specified, expansion may not be executed correctly.
Therefore, use a Huffman table that has uniform values for V1 through Vm.

2.4.7 Compliance with Exif Standard

The Exif standard is an image format standard for digital still cameras, created by the Japan Electronic Industry Development Association.

The following are the features of the image format stipulated in the Exif standard (ver. 1.0).

- Data of a parameter stipulated in the Exif standard is embedded in the APP1 marker segment.
- Three quantization tables for Y, Cb, and Cr are provided, and included in one DQT segment.
- All the Huffman tables are included in one DHT segment.

(1) Setting procedure (When using the same quantization table for Cb and Cr)

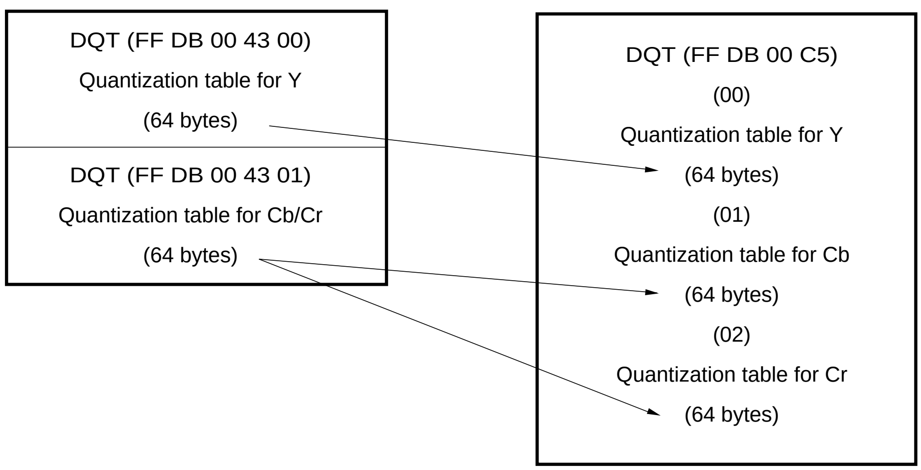
Although the AP705100-B03 and AP70732-B03 basic libraries do not support a function for creating data to be embedded in the APP1 segment, the embedding of data in the APP1 segment is possible.
To make the AP705100-B03 and AP70732-B03 basic libraries comply with the Exif standard, cast the four bytes of CJInfo.IR [8] into unsigned char* type, and specify the key word character string "Exif" as follows. An example description is shown below.

```
* (unsigned char**) & (CJInfo.IR [8] ) = "Exif" ;
```

This setting enables the following file format.

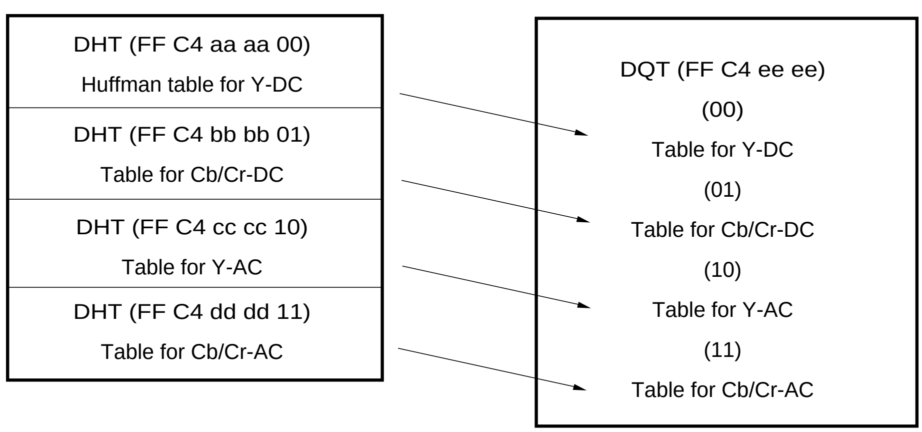
<1> Mandatory exclusion of the comment marker (This setting prevents the embedding of a comment marker.)

<2> Integration of DQT segments



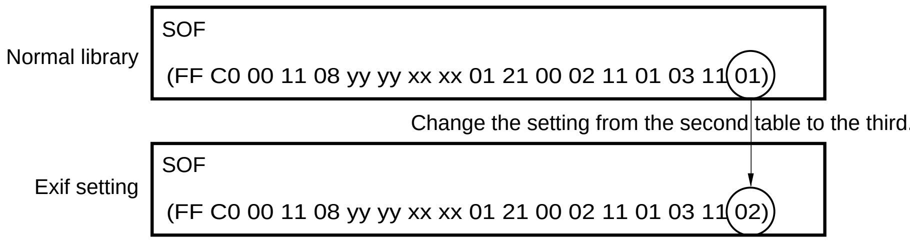
Remark In this case, the same table is copied for both the Cb table and Cr table.

<3> Integration of DHT segments



<4> Modification of SOF segment setting

Modify the SOF segment which specifies the second table for Cr such that it specifies the third table for Cr.



(2) Setting procedure (When using separate quantization tables for Cb and Cr)

The AP705100-B03 and AP70732-B03 basic libraries allow the use of two separate tables for the second (Cb) and third (Cr) quantization table, when setting is performed as follows:

<1> Cast the four bytes of CJInfo.IR [8] into unsigned char* type, and specify the five-letter key word character string "ExifQ."

<2> Cast the four bytes of CJInfo.IR [12] into char** type, and specify the result as the first address of the third quantization table (64 bytes).

<3> Allocate 2,880 bytes for the work area that is specified by member Work in the structure.

An example description is shown below.

```
char ThirdQtbl[64] = {
        1,2,3,4,5,....
};
*(unsigned char**) & (CJinfo.IR[8]) = "ExifQ";
*(char**)&(CJinfo.IR[12]) = ThirdQtbl;
```

Caution This setting requires 2,880 bytes (not 2,688 bytes) for the external RAM work area. If compression is performed on this setting with the work area set to 2,688 bytes, the next 192 bytes are overwritten without warning.

2.4.8 Error Contents during Compression

The compression routine of the basic library assigns the value of an error to member "ErrorState" of the CJINFO structure and stops processing if the processing cannot be completed normally for some reason. At this time, the routine returns JPEG_ERR as the return value.

The error contents that may be output are listed in Table 2-29.

Table 2-29. Error Contents of Compression Routine

Value	Meaning
0x00000001	Image exceeds range of VRAM (if value of (Width + StartX) of CJINFO structure exceeds value of VRAM_W_Pixel/if value of (Height + StartY) exceeds value of VRAM_H_Pixel).
0x00000002	Unsupported sampling ratio (if compression at a sampling ratio of 2:1:1 is specified even though linking is performed without a library of 2:1:1).
0x00000005	Specified Huffman table is invalid.
0xFFFFFFFF	Fatal error (error due to modification of library).

2.4.9 Output Information by Compression Routine

The compression routine of the basic library outputs the following information when the processing is completed normally.

Table 2-30. Output Information of Compression Routine

Member	Return value
FileSize	Number of bytes constituting completed JPEG file (normal compression mode)
	Number of bits of compressed data in one MCU (test mode)

2.5 BASIC EXPANSION PROCESSING

Basic expansion processing expands a JPEG file to create image data.

2.5.1 Basic Expansion Main Function

Classification Expansion processing system
Function name jpeg_Decompress
Feature JPEG expansion processing main
Format #include "jpeg.h"
int jpeg_Decompress (DJINFO* dJpeginfo)
Argument First address of DJINFO structure
Return value The return value is a numeric and is defined as #define JPEG_OK 0 in C.

Table 2-31. Return Value for Expansion Processing Function

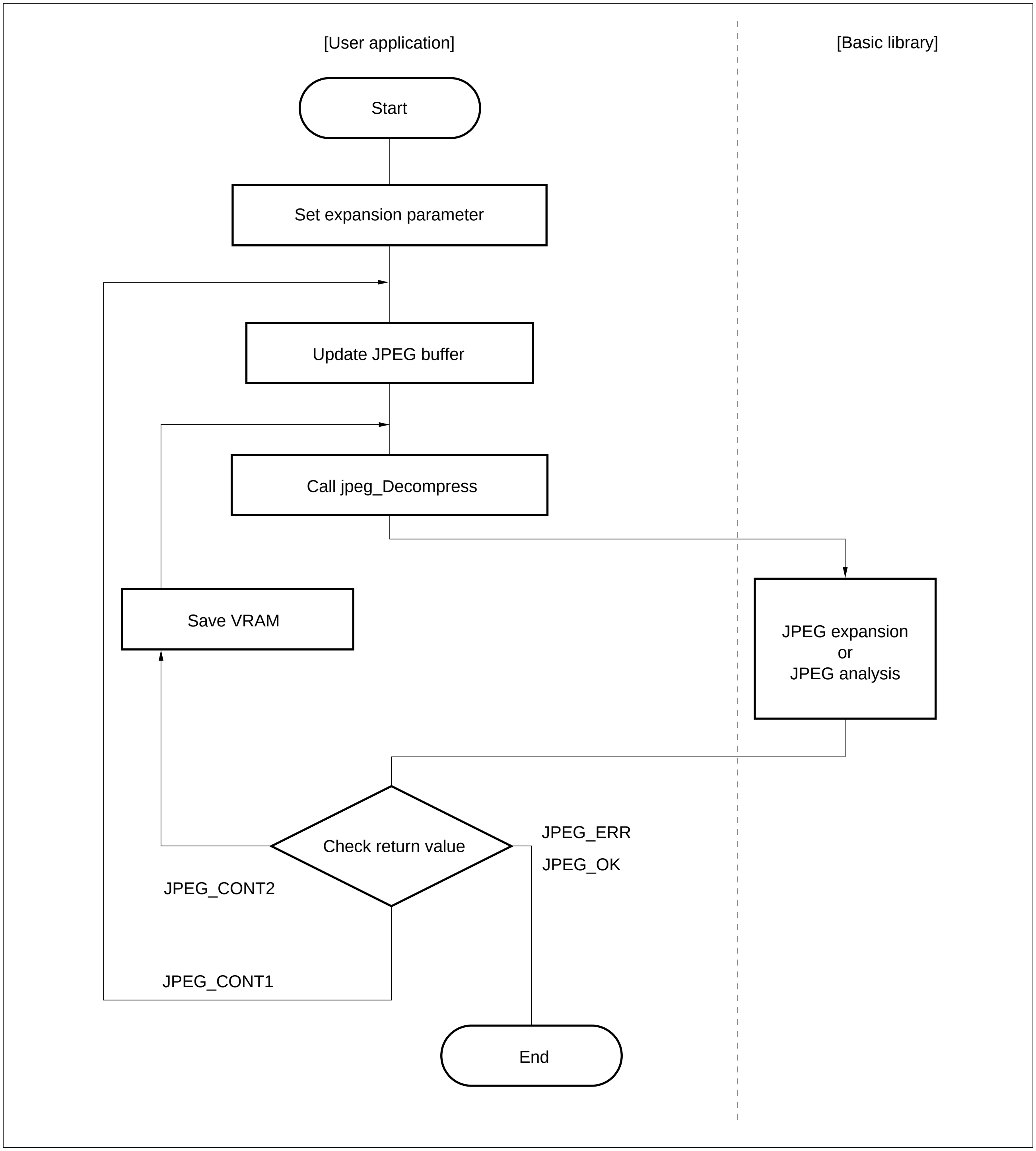
Return value	Description
JPEG_OK	Normal completion
JPEG_ERR	Error termination
JPEG_CONT1	Aborted by JPEG buffer
JPEG_CONT2	Aborted by VRAM

Remark For JPEG_ERR, an error statement is stored into member "ErrorState" of the DJINFO structure.

2.5.2 Basic Expansion Processing Flow

The flow of the basic expansion processing is shown below.

Figure 2-29. Basic Expansion Processing Flow



2.5.3 Setting of DJINFO Structure Parameter

Figure 2-30. Setting of DJINFO Structure Parameter (AP70732-B03)

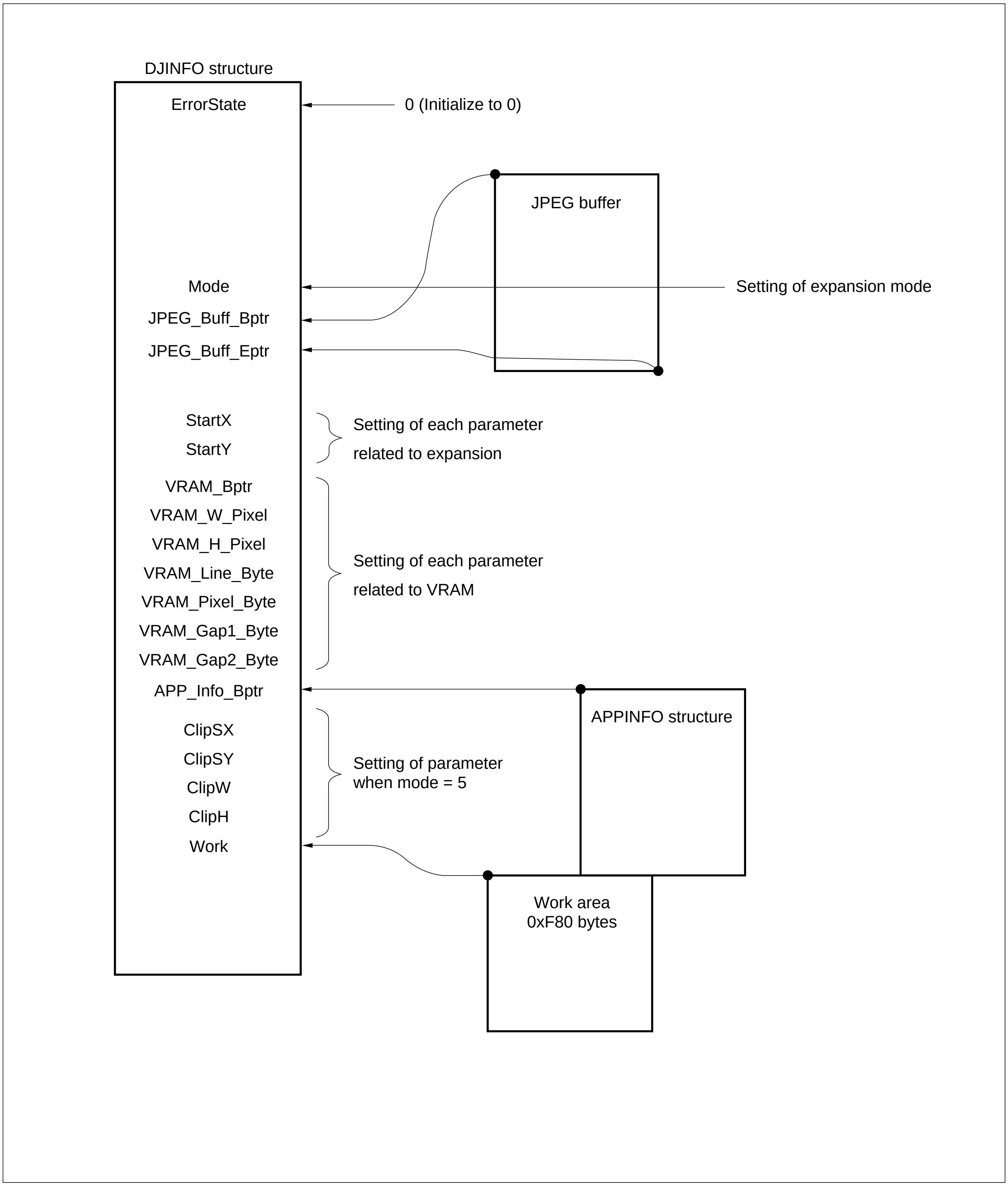
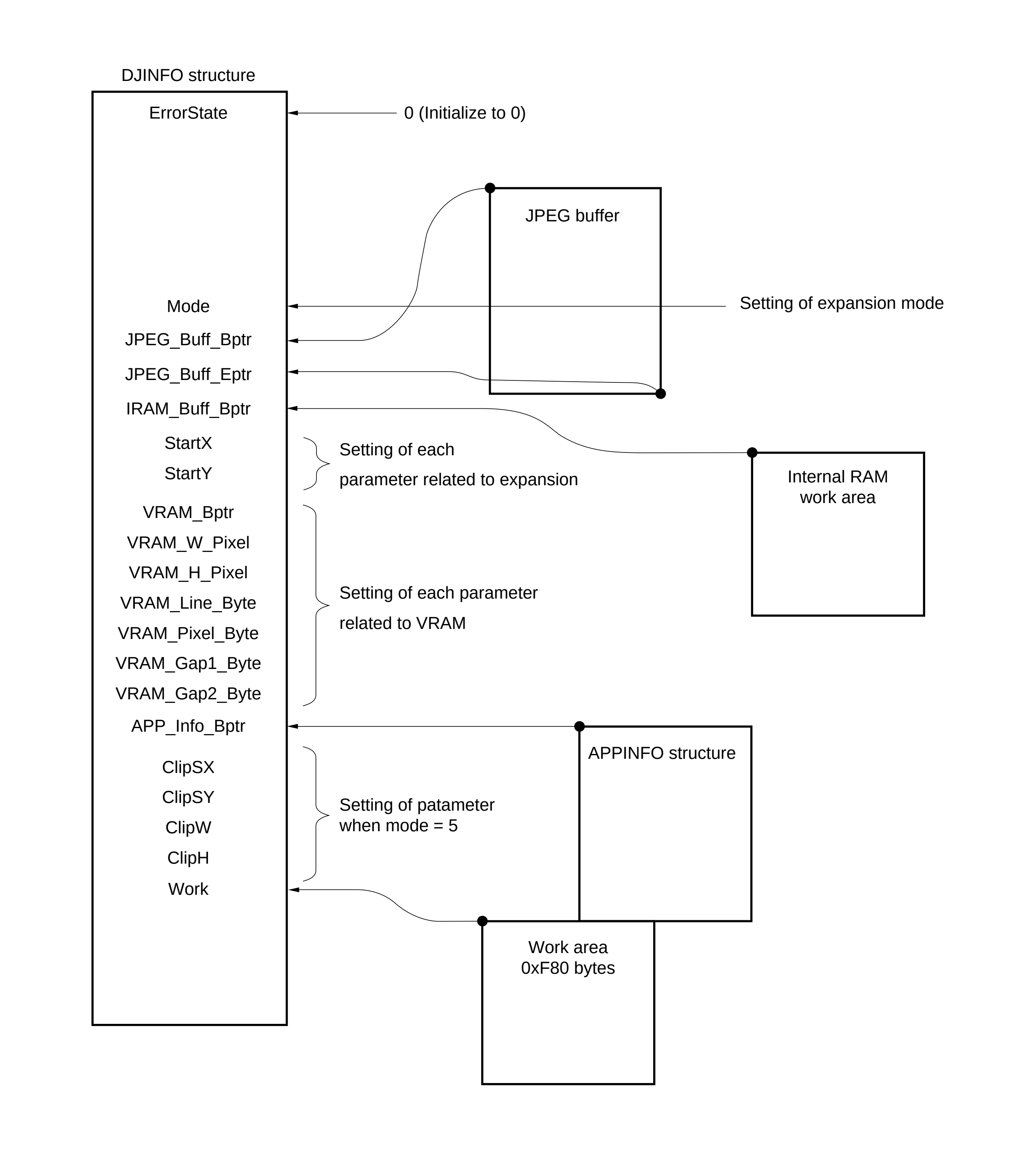


Figure 2-31. Setting of DJINFO Structure Parameter (AP705100-B03)



(1) Initialization of error status (ErrorState)

Initialize the error status to 0 once only, when expansion parameters are set before the basic expansion routine is started.

Set value: 0

Caution Do not perform any other initialization because, when processing is stopped and then resumed, the basic library determines whether the processing is being started for the first time or being resumed by referring to this ErrorState value (if the processing is stopped, the address from which the processing is to be resumed is stored).

(2) Mode (Mode)

Set values and the corresponding operations or modes are shown below.

Table 2-32. Set Values for Member Mode

7 6	5	4	3 0
RFU	HF	QF	MD

Bit	Bit name	Description
7 to 6	RFU	Reserved field**Note**
5	HF	Huffman table initialization flag**Note** 0: Initializes the tables. 1: Does not initialize the tables.
4	QF	Quantization table initialization flag**Note** 0: Initializes the tables. 1: Does not initialize the tables.
3 to 0	MD	Mode 0: Analysis mode 1: Normal expansion mode 2: 1/4 expansion mode 3: 1/16 expansion mode 4: 1/64 expansion mode 5: Clipping expansion mode (RSTn is not used.) 6: Clipping (RSTn is used. EOI is not searched for.)**Note** 7: Clipping (RSTn is used. EOI is searched for.)**Note**

Note Added in ver. 2.10.

(a) Mode 0

If a structure address is appended to the member of the JPEG structure to analyze the APPn segment, analysis of the position and size of the APPn segment is started. If an APPn segment analysis structure is not specified, analysis related to the APPn segment is not executed.

In addition to the APPn segment information, the sampling ratio, restart interval, vertical and horizontal sampling ratios of the image, and JPEG file size are analyzed.

(b) Mode 1

Normal expansion processing is performed.

Figure 2-32. Example of Expansion in Expansion Mode 1

(c) Mode 2

Expansion processing is performed at high speed by using the reverse DCT translation routine so that the part of reverse DCT is not 8 x 8, but 4 x 4.

Figure 2-33. Example of Expansion in Expansion Mode 2

(d) Mode 3

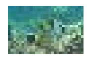
The vertical height and horizontal width are reduced to 1/4 for output.

Figure 2-34. Example of Expansion in Expansion Mode 3

(e) Mode 4

The vertical height and horizontal width are reduced to 1/8 for output.

Figure 2-35. Example of Expansion in Expansion Mode 4



(f) Mode 5

A specified rectangle is extracted from the original JPEG file and only that portion is expanded.

Figure 2-36. Example of Expansion in Expansion Mode 5

Clipping must be performed in units of MCUs.
To use this mode, the values of the following members must be set.

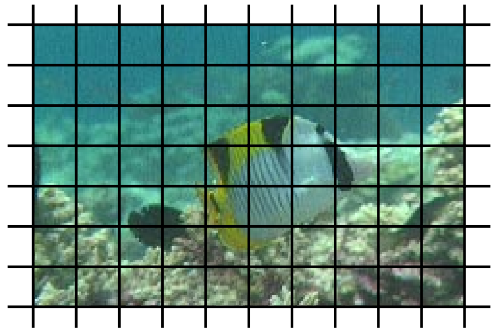
Table 2-33. Set Values for Members Related to Clipping

Member	Description
ClipSX	Sets the clipping start position in units of MCUs, starting from the left
ClipSY	Sets the clipping start position in units of MCUs, starting from the top
ClipW	Specifies horizontal width in units of MCUs
ClipH	Specifies vertical height in units of MCUs

When clipping is performed as shown above, and if the image is divided into units of MCUs, as shown below, set the above members as follows:

```
ClipSX = 3;
ClipSY = 2;
ClipW  = 6;
ClipH  = 4;
```

Figure 2-37. Example of Clipping Specification



(g) Mode 6

Huffman decoding is implemented by searching for restart markers. This helps increase the processing speed a little. In this mode, the value of Djinfo.FileSize is not defined, because decoding is terminated when the clipping area has been decoded.

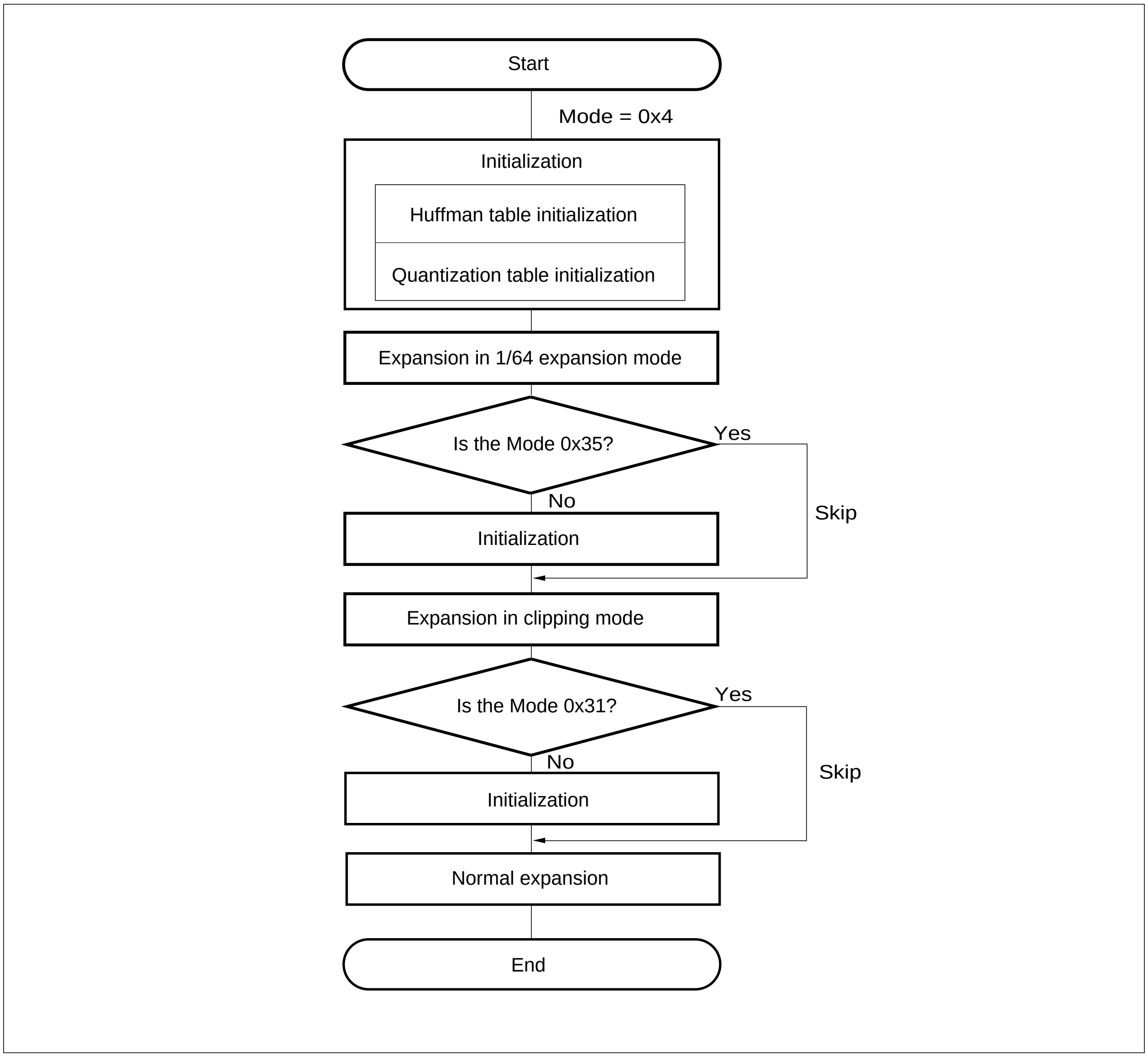
(h) Mode 7

Mode 7 maintains consistency in the values of Djinfo.FileSize as well as the functions of mode 6. The processing speed in this mode is higher than that in mode 5, but lower than that in mode 6.

Just as in compression processing, the 3,968-byte area specified by member Mode in the structure is loaded with quantization and Huffman look-up tables, and the HF and QF bits determine whether to initialize those look-up tables.

Figure 2-38 shows an example in which expansion is performed in 1/64 expansion mode before expansions with the same look-up tables.

Figure 2-38. Renewed Expansion Mode Setting



(3) Start address and end address of JPEG buffer

For details of the JPEG buffer, see **Section 2.3.5**.

Table 2-34. Set Values of Members JPEG_Buff_Bptr/JPEG_Buff_Eptr

Member	Description
JPEG_Buff_Bptr	First address of JPEG buffer
JPEG_Buff_Eptr	First address of JPEG buffer + JPEG buffer size

If buffer save processing is performed in the middle of processing due to the limit imposed on the JPEG buffer size, two JPEG buffers can be used alternately.

For details, see **Figure 2-21**.

(4) Internal RAM work area address (IRAM_Buff_Bptr: AP705100-B03)

For details of the internal RAM work area, see **Section 2.3.4**.

Table 2-35. Set Value for Member IRAM_Buff_Bptr

Member	Description
IRAM_Buff_Bptr	First address of internal RAM work area

0x400, 0x300, or 0x280 bytes from the first address are unconditionally overwritten, at a sampling ratio of the JPEG file to be expanded of 4:1:1/2:1:1/1:1:1. Unlike compression, the sampling ratio is not specified by the user in the case of expansion. Instead, the value of the SOF header of the JPEG file is used as the sampling ratio.

This member need not be set with the AP70732-B03 (V810 family version).

Caution In Mode = 0 (analysis mode), an area specified in this internal RAM work area is 0x100 bytes, regardless of the sampling ratio.

(5) Image start positions x (StartX) and y (StartY)

Set value: -32,768 to 32,767

The unit of the value is the number of pixels.

For details of StartX/StartY, see **Figure 2-22**.

(6) Setting of parameters related to VRAM

The setting of the parameters related to the VRAM is exactly the same as that for compression. See **(10)** and **(11)** in **Section 2.4.3**.

Table 2-36. Set Values for Members Related to VRAM

Member	Description	Setting when customizing VRAM output portion
VRAM_W_Pixel	Horizontal number of pixels of VRAM	Necessary
VRAM_H_Pixel	Vertical number of pixels of VRAM	
VRAM_Bptr	VRAM first address (reference address)	Unnecessary
VRAM_Line_Byte	Address difference of VRAM of one vertical pixel	
VRAM_Pixel_Byte	Address difference of VRAM of one horizontal pixel	
VRAM_Gap1_Byte	If VRAM is YCbCr, address difference between Y and Cb of same pixel. If VRAM is RGB, address difference between R and G of same pixel.	
VRAM_Gap2_Byte	If VRAM is YCbCr, address difference between Y and Cr of same pixel. If VRAM is RGB, address difference between R and B of same pixel.	

To customize the VRAM output portion, the two members that must be set (VRAM_W_Pixel and VRAM_H_Pixel) check the size when the SOF segment is analyzed.

(7) Specification of APPINFO table (APP_Info_Bptr)

If the first address of the APPINFO structure is specified in APP_Info_Bptr, the APPn segment is analyzed.

Table 2-37. Set Value for Member APP_Info_Bptr

Member	Set value
APP_Info_Bptr	0: APPn segment is not analyzed.
	First address of APPINFO structure: analyzed

If the APPn segment in which the APPINFO structure is registered is found, the first address and size of that data are written to the member corresponding to the APPn segment number.

(8) Setting of parameters related to clipping (ClipSX, ClipSY, ClipW, ClipH)

These values are referenced only in Mode = 5 (clipping mode).

For an explanation of how to set these values, see **(f)** in **Section 2.5.3 (2)**.

Caution An error occurs if clipping is specified out of the actual image.

(9) External RAM work area address (Work)

Set the first address of the external RAM work area.

Table 2-38. Set Value for Member Work

Member	Description
Work	First address of external RAM work area of 0xF80 bytes

Caution Work is not necessary in Mode = 5.

2.5.4 Compliance with Exif Standard

The AP705100-B03 and AP70732-B03 basic libraries use three quantization tables with the normal setting; therefore an Exif-standard JPEG file cannot be expanded with them.

To enable the expansion, follow the steps below.

<1> Cast the four bytes of DJInfo.IR[28] into unsigned char* type, and specify the four-letter key word character string "Exif."

<2> Allocate 4,224 bytes for the work area that is specified by member Work in the structure.

An example description is shown below.

```
*(unsigned char**)&(DJInfo.IR[28]) = "Exif"
```

Caution This setting requires 4,224 bytes (not 3,968 bytes) for the external RAM work area. If expansion is performed on this setting with the work area set to 3,968 bytes, the next 256 bytes are overwritten without warning.

2.5.5 Error Contents during Basic Expansion

The expansion routine of the basic library assigns the value of an error to member "ErrorState" of the DJINFO structure and stops processing if the processing cannot be completed normally for some reason. At this time, the routine returns JPEG_ERR as the return value.

The error contents that may be output are listed in Table 2-39.

Table 2-39. Error Contents of Basic Expansion Routine

Value	Meaning
0x00000001	Image exceeds range of VRAM. • If the value of the horizontal size of the JPEG image added to StartX of the DJINFO structure exceeds VRAM_W_Pixel value • If the value of the vertical size of the JPEG image added to StartY exceeds VRAM_H_Pixel value
0x00000002	Unsupported sampling ratio (if expansion at a sampling ratio of 2:1:1 is specified even though linking is performed without a library of 2:1:1).
0x00000003	Value other than 0 is set to Pq of DQT header.
0x00000004	Value of Tp of DQT header is other than 0, 1, 2, or 3.
0x00000005	Values of Tc and Tp of DHT header are illegal.
0x00000006	Number of components of SOS header is not 3.
0x00000007	Huffman table number specified by SOS header is wrong.
0x00000008	Value of Ss of SOS header is not 0.
0x00000009	Value of Se of SOS header is not 63.
0x0000000A	Values of Ah and Al of SOS header are not 0.
0x0000000B	Value other than 8 is set in P of SOF header.
0x0000000C	Value set in Nf of SOF header is too great.
0x0000000D	Unknown marker appears.
0x0000000E[Note]	Value of RSTn marker is illegal.
0x0000000F	Other error
0xFFFFFFFF	Fatal error (error due to modification of library)
0x00000010	SOI marker could not be found.
0x00000011	SOF0 marker could not be found.
0x00000012	DQT marker could not be found.
0x00000013	DHT marker could not be found.
0x00000014[Note]	RSTn marker could not be found.
0x00000015	EOI marker could not be found.
0x0000001F	A marker was found at an unexpected location.

Note The RSTn marker error is returned only in clipping modes (Mode 6, Mode 7).

Table 2-40. Unchecked Errors

Condition	Unchecked error
Expansion is performed in analysis mode.	SOI/SOF1/DQT/DHT/EOI
The mode value 0x10 is set.	DQT
The mode value 0x20 is set.	DHT
Expansion is performed in clipping mode (Mode 7).	EOI

2.5.6 Output Information of Basic Expansion Routine

The expansion routine of the basic library outputs the following information when the processing is completed normally.

Table 2-41. Output Information of Expansion Routine

Member of DJINFO structure	Description
FileSize	Number of bytes constituting expanded JPEG file
Sampling	Sampling ratio for expanded JPEG file 0x22 (4:1:1 (H:V = 2:2)) 0x41 (4:1:1 (H:V = 4:1)) 0x21 (2:1:1 (H:V = 2:1)) 0x11 (1:1:1 (H:V = 1:1))
Restart	Restart interval of expanded JPEG file
Width	Horizontal image size of expanded JPEG file (number of pixels)
Height	Vertical image size of expanded JPEG file (number of pixels)

Information on the JPEG file header is stored in the four bytes between the first address and 0x7C of the Djinfo structure.

An example description is shown below.

*(int**)&(DJinfo.IR[48])

The descriptions of the four bytes are listed below.

Table 2-42. Description of Information on JPEG File Header

Bit	Segment name	Description
31	SOI	This flag is set if the SOI marker has been found. In this case, all the other flags are masked with 0s.
30	EOI	This flag is set if the EOI marker has been found.
29	SOF0	This flag is set if the SOF0 marker has been found.
10	COM	This flag is set if the COM marker has been found.
9	SOS	This flag is set if the SOS marker has been found.
8	DRI	This flag is set if the DRI marker has been found.
7	DQT	Table number 3 DQT
6	DQT	Table number 2 DQT
5	DQT	Table number 1 DQT
4	DQT	Table number 0 DQT
3	DHT	Table number 1 DHT for AC
2	DHT	Table number 0 DHT for AC
1	DHT	Table number 1 DHT for DC
0	DHT	Table number 0 DHT for DC

As exceptions, the following checks are not performed.

- Check on DHT and DQT in analysis mode
- Check on DQT if Mode value 0x10 is set in a mode other than analysis mode
- Check on DHT if Mode value 0x20 is set in a mode other than analysis mode

The following information is output only when the APPINFO structure is specified for member APP_Info_Bptr of DJINFO.

Table 2-43. APPxx_Buff_Bptr/APPxx_BuffSize

Member of APPINFO structure	Description
APPxx_Buff_Bptr	Address of APPn segment (position relative to first address of JPEG file)
APPxx_Buffsize	Size of APPn segment (bytes)

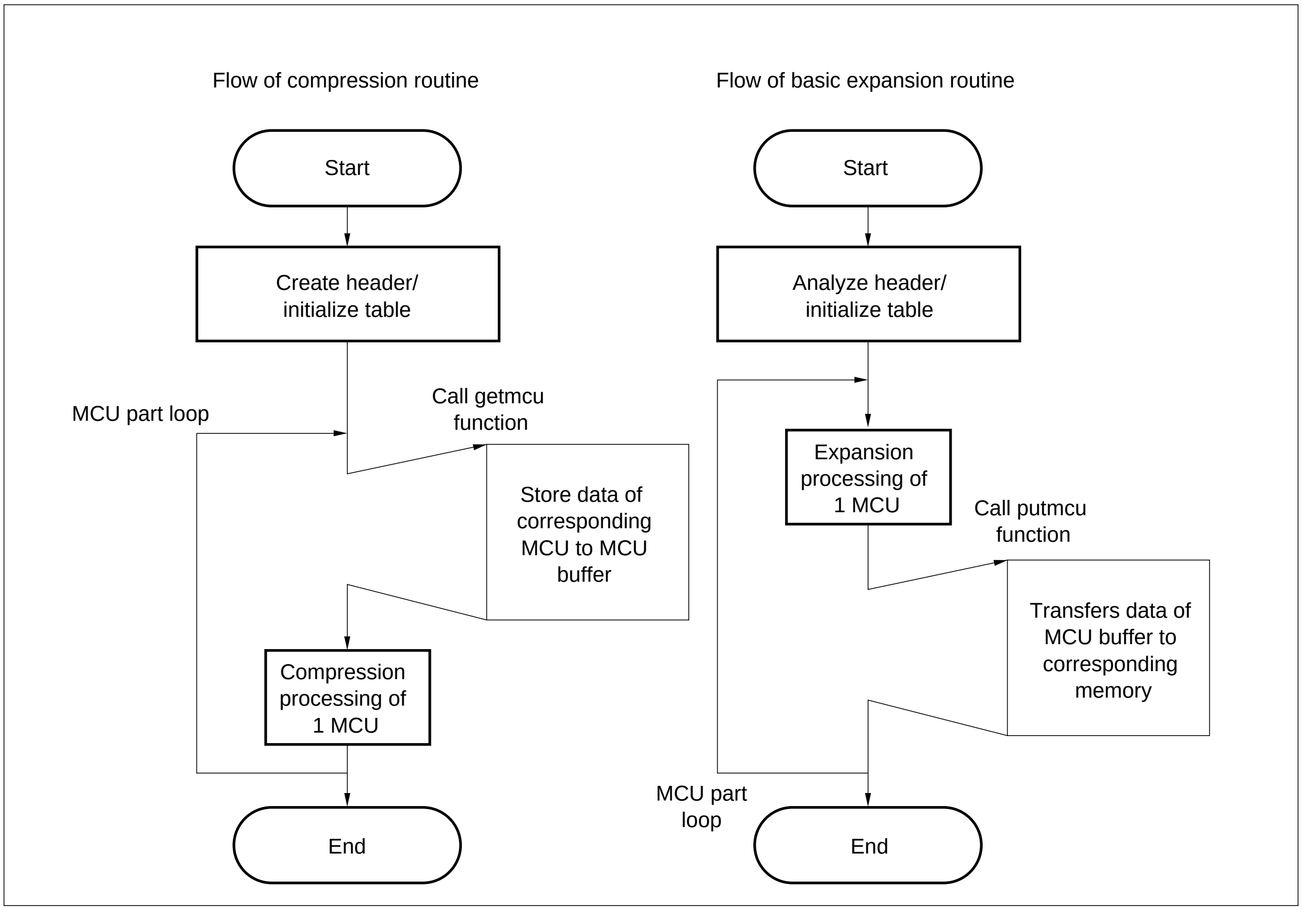
2.6 CUSTOMIZING BASIC LIBRARY

The image input/output part is mostly influenced by hardware in JPEG compression/expansion processing. The basic library allows the user to create the image input/output part (although the default VRAM access function supplied with the basic library may be used, this function does not emphasize the speed because its specifications are general-purpose).

2.6.1 Handling Image Data with Basic Library

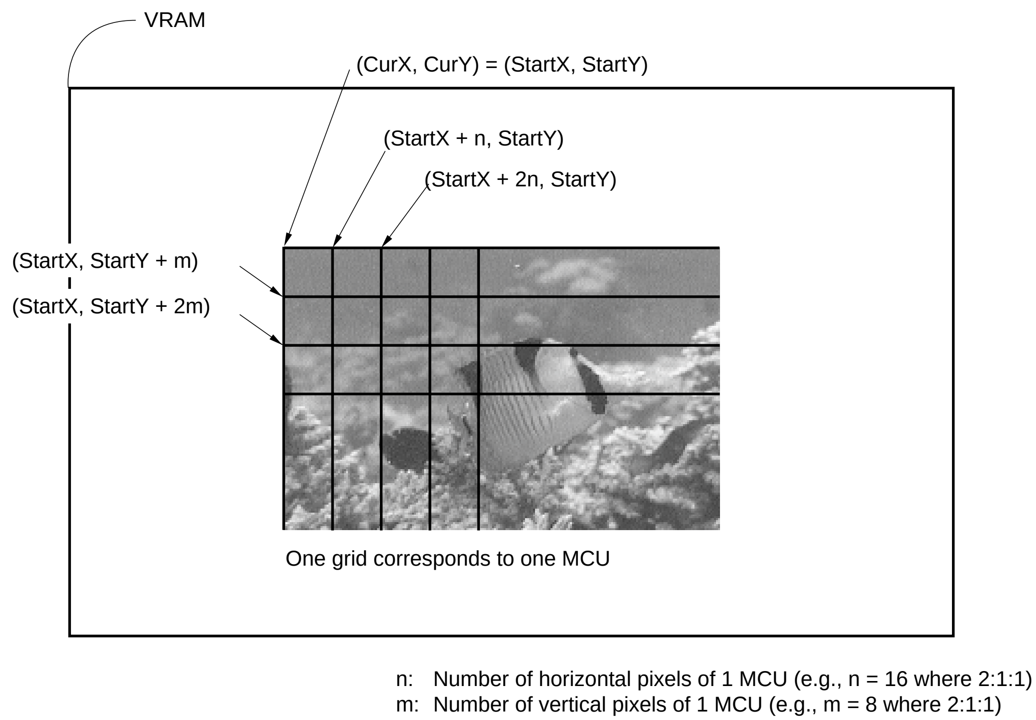
The basic library processes image data in units of MCUs (Minimum Coded Unit) (image input, DCT translation, quantization, and Huffman coding are executed in MCU units for compression, and Huffman coding, reverse quantization, reverse DCT translation, and image output are executed in MCU units for expansion).

Figure 2-39. Flow of JPEG Processing



The MCU buffer is allocated as the last member of the CJINFO and DJINFO structures.

Figure 2-40. Member CurrentX/CurrentY of Structure



To determine the part indicated by the MCU when customizing the putmcu or getmcu function, refer to the member (CurrentX or CurrentY) of the structure.

The value of member CurrentX or CurrentY is updated by the library each time the processing for one MCU has been executed. Do not change the value of the member from the size at which the function is customized by the user.

To customize the getmcu function for compression, ensure that the corresponding MCU data is stored into the MCU buffer in Y/Cb/Cr format (0 to 255 for each of Y/Cb/Cr) each time the getmcu function is called from the library. Conversely, to customize the putmcu function for expansion, make sure that the data corresponding to the MCU is transferred to VRAM because the data is stored in Y/Cb/Cr format when the putmcu function is called.

Table 2-44. Unit of MCU

Sampling ratio	MCU unit
4:1:1 (H:V = 2:2)	Vertically 16 x Horizontally 16 pixels
4:1:1 (H:V = 4:1)	Vertically 8 x Horizontally 32 pixels
2:1:1 (H:V = 2:1)	Vertically 8 x Horizontally 16 pixels
1:1:1 (H:V = 1:1)	Vertically 8 x Horizontally 8 pixels

The basic library inputs DCT translation for compression and outputs reverse DCT translation for expansion in YCbCr format, instead of RGB format. If the VRAM is of RGB type, the image data must be translated from RGB to YCbCr in order to match the data with the format of VRAM (see **Figure 1-4**).

This library handles the values of Y, Cb, and Cr as unsigned char type, to conform with CCIR Recommendation 601.

2.6.2 Sampling Ratio and Block

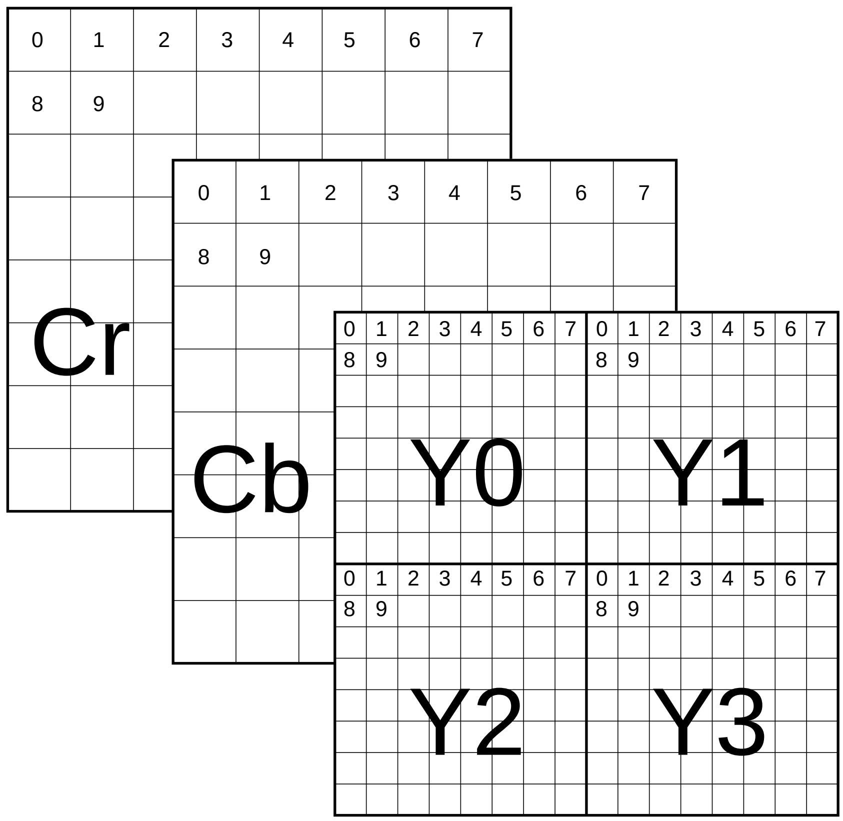
Each MCU is decomposed to YCbCr for compression and decomposed into blocks, according to the sampling ratio.

Table 2-45. MCU and Block

Sampling ratio	MCU unit	Block
4:1:1 (H:V = 2:2)	16 x 16	Y:4/Cb:1/Cr:1 block
4:1:1 (H:V = 4:1)	8 x 32	Y:4/Cb:1/Cr:1 block
2:1:1 (H:V = 2:1)	8 x 16	Y:2/Cb:1/Cr:1 block
1:1:1 (H:V = 1:1)	8 x 8	Y:1/Cb:1/Cr:1 block

Figure 2-41. Image Data of 1 MCU (1/2)

(a) Where 4:1:1 (H:V = 2:2)



(b) Where 4:1:1 (H:V = 4:1)

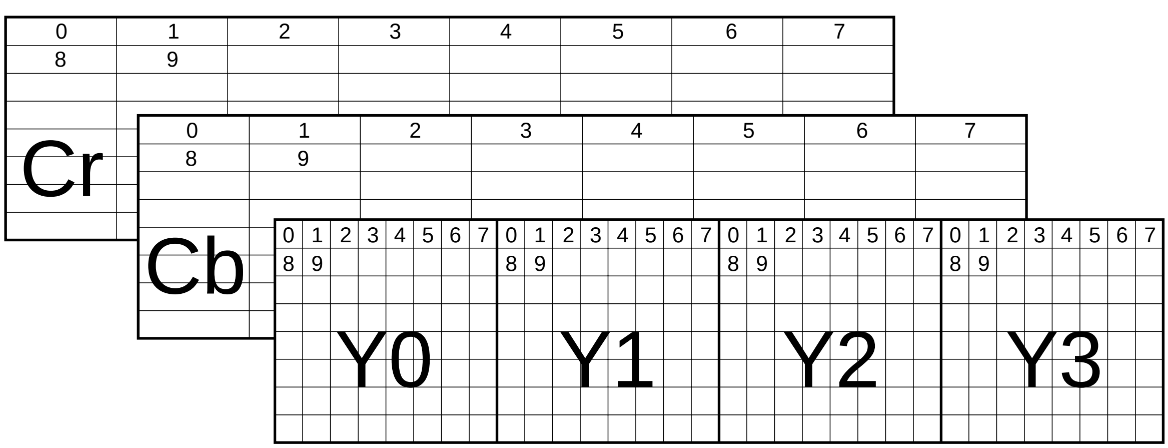
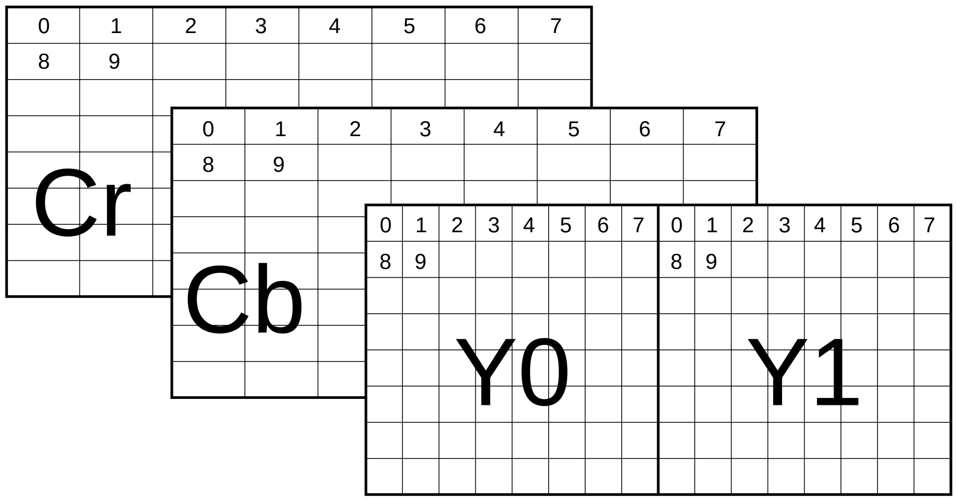
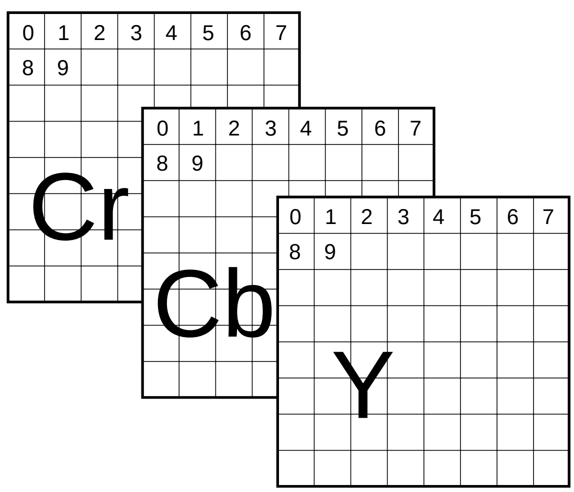


Figure 2-41. Image Data of 1 MCU (2/2)

(c) Where 2:1:1 (H:V = 2:1)



(d) Where 1:1:1 (H:V = 1:1)



For compression, the average of adjacent pixels must be calculated at sampling ratios other than 1:1:1, regarding the chrominance component (Cb/Cr). The processing required to calculate the average value is called sampling.

Table 2-46. Sampling of Chrominance Component

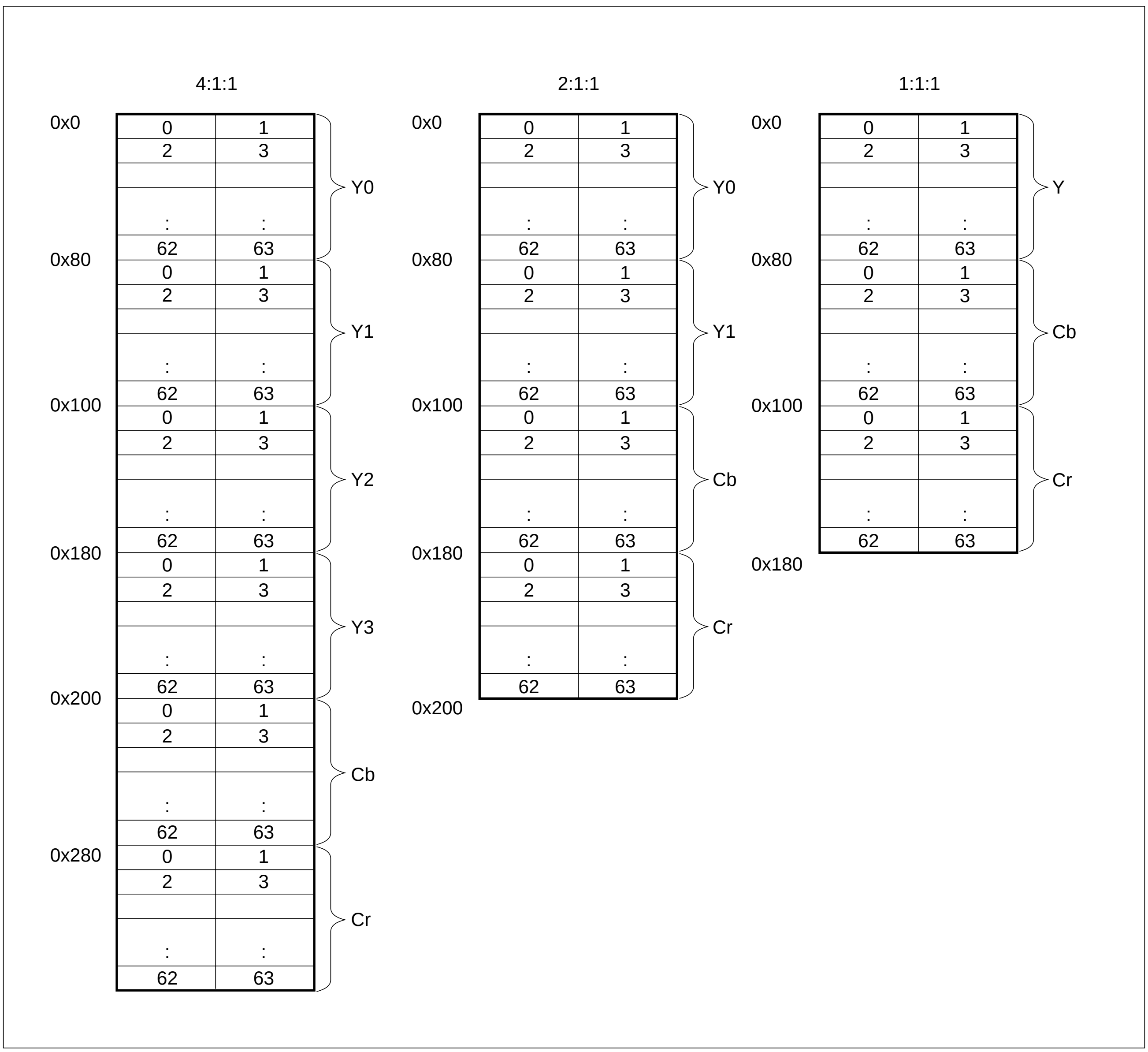
Sampling ratio	Sampling
4:1:1 (H:V = 2:2)	Averages chrominance component of 2 vertical x 2 horizontal pixels
4:1:1 (H:V = 4:1)	Averages chrominance component of 4 horizontal pixels
2:1:1 (H:V = 2:1)	Averages chrominance component of 2 horizontal pixels

2.6.3 Image Data Buffer

Image data for 1 MCU is stored into the MCU buffer (the last member of a structure in the case of the AP70732-B03, and the internal RAM work area in the case of the AP705100-B03).

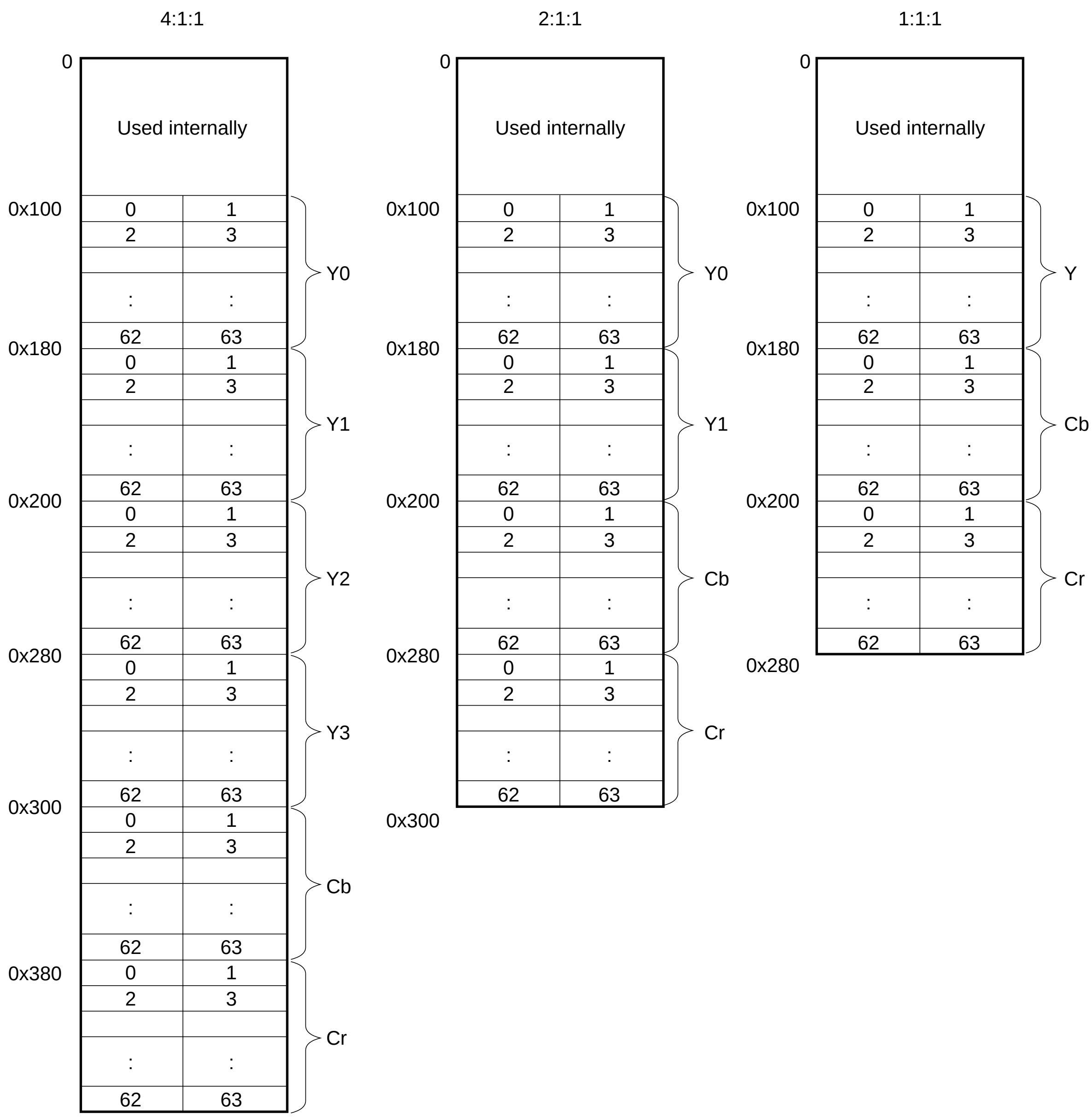
The position to which the data is to be stored is determined as shown below.

Figure 2-42. Image Data of MCU Buffer (AP70732-B03)



Zero is inserted into the high-order eight bits of the data of one chrominance element of one pixel (8 bits), and the data is handled as unsigned short type (16 bits).

Figure 2-43. Buffer for Image Data of Internal RAM (AP705100-B03)



In a mode where image data is reduced or expanded, the portion enclosed in a solid line in a the figure below (when not reduced) has a meaning.

Figure 2-44. Image Data in Reduced Expansion Mode (AP70732-B03) (1/3)

(a) 1/4 expansion (Mode = 2)

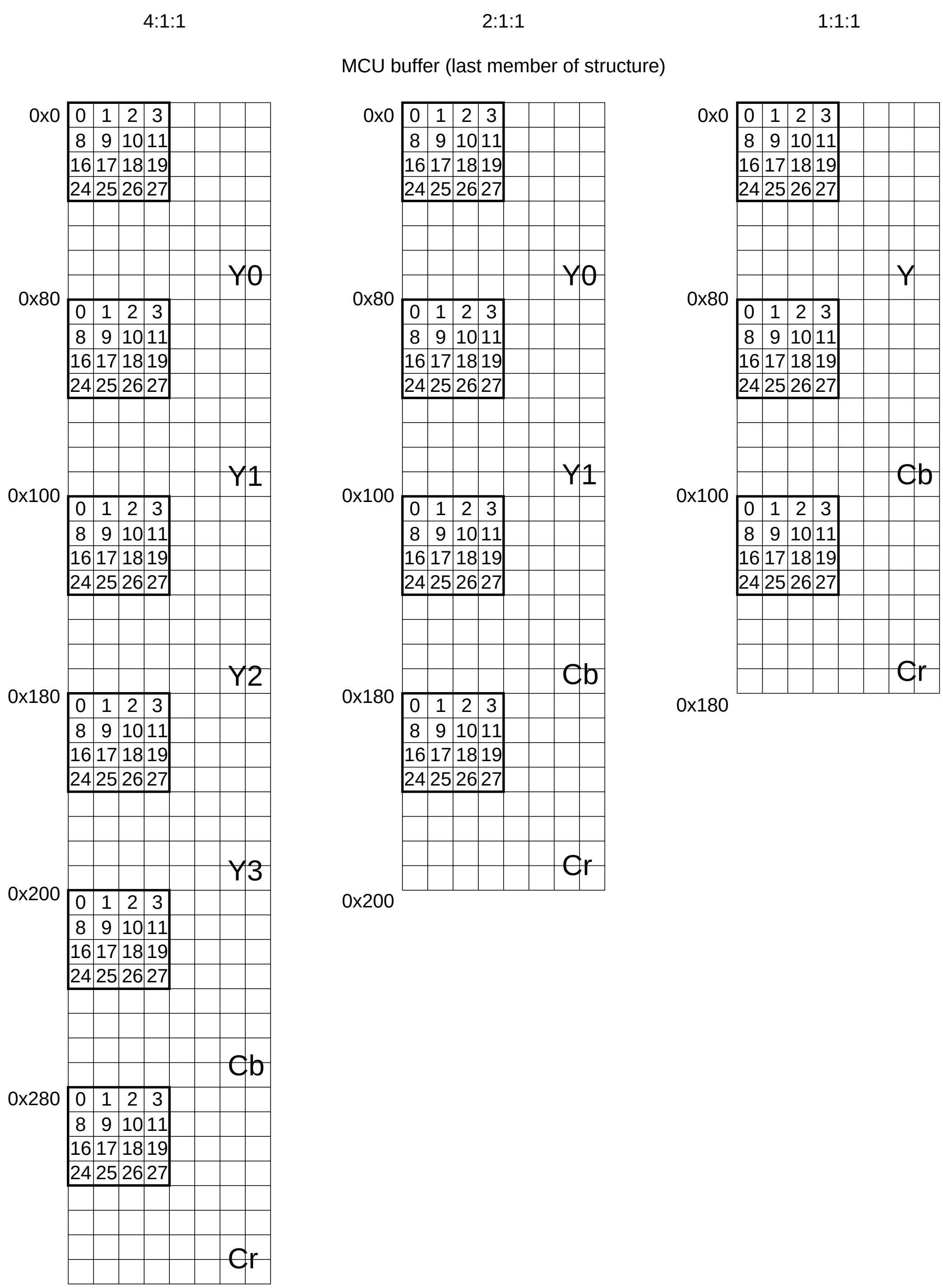


Figure 2-44. Image Data in Reduced Expansion Mode (AP70732-B03) (2/3)

(b) 1/16 expansion (Mode = 3)

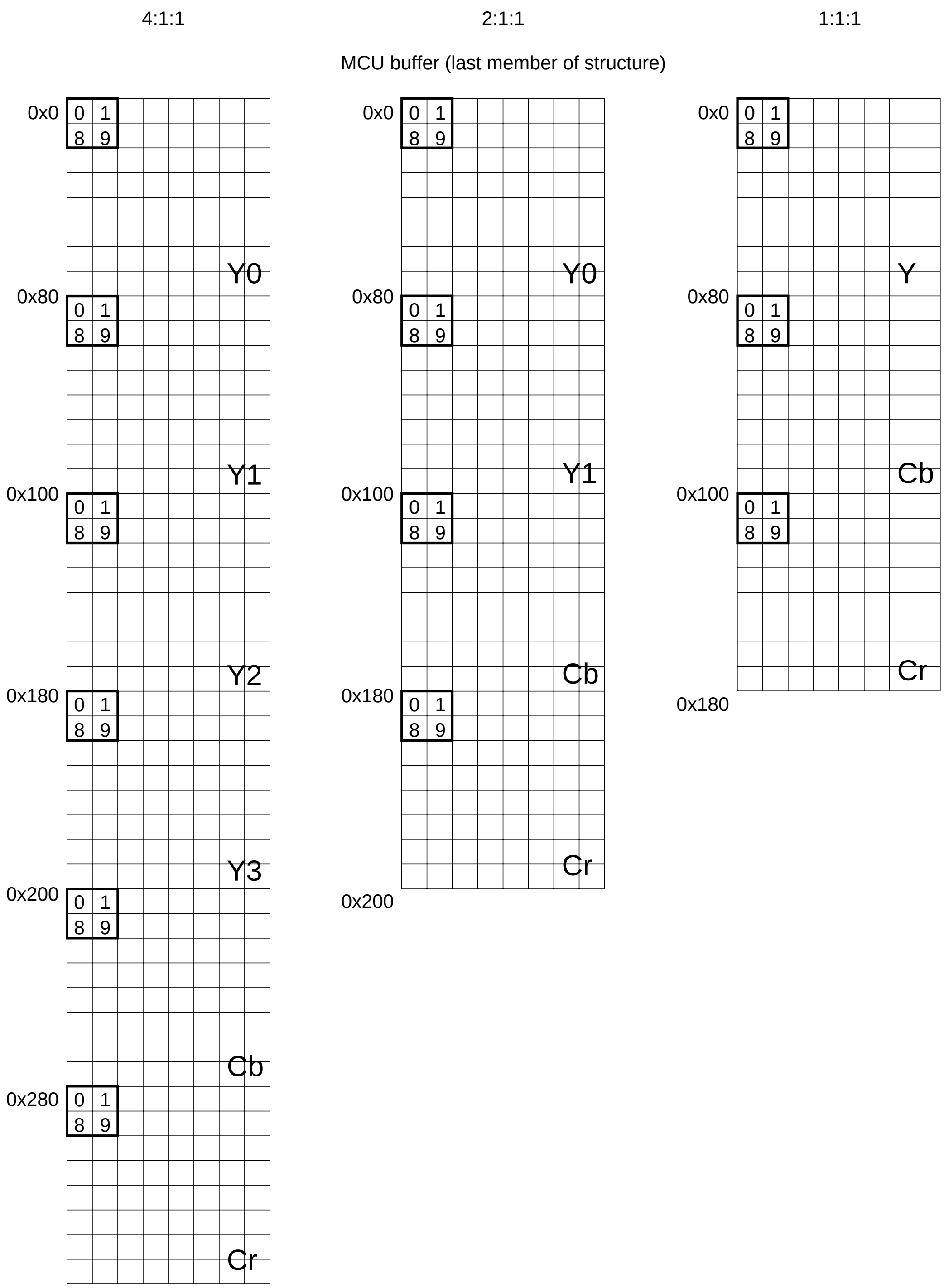


Figure 2-44. Image Data in Reduced Expansion Mode (AP70732-B03) (3/3)

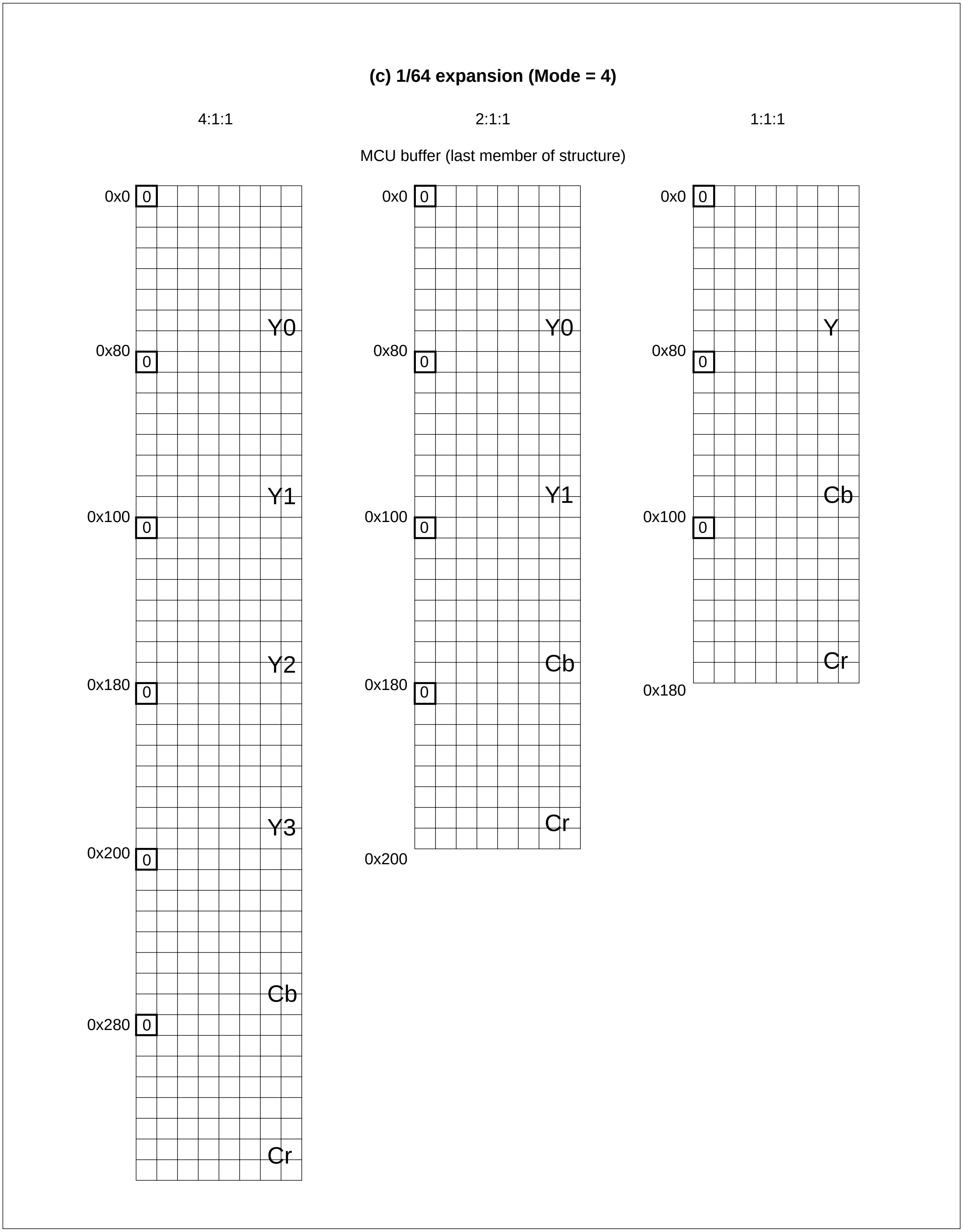


Figure 2-45. Image Data in Reduced Expansion Mode (AP705100-B03) (1/3)

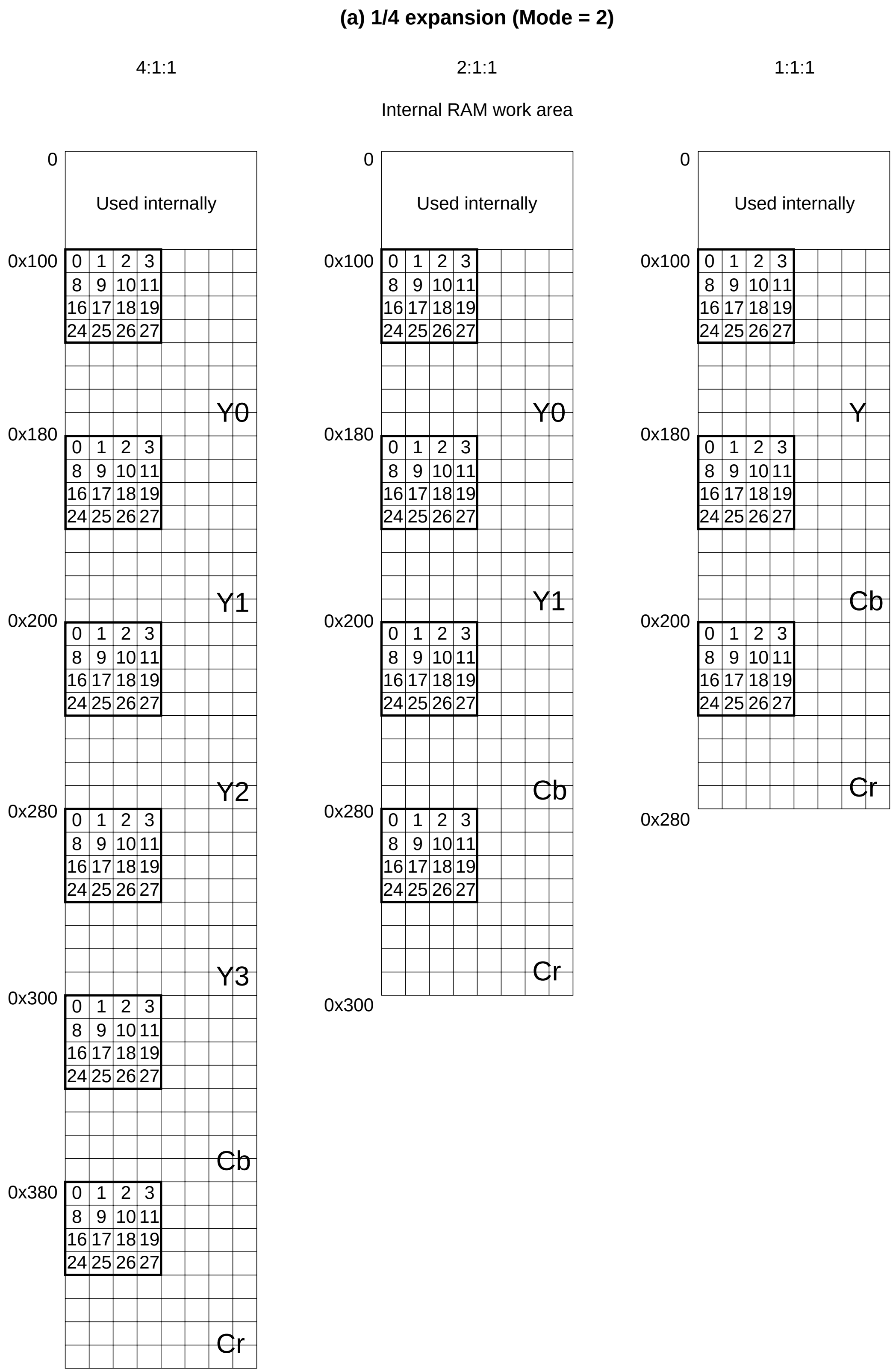


Figure 2-45. Image Data in Reduced Expansion Mode (AP705100-B03) (2/3)

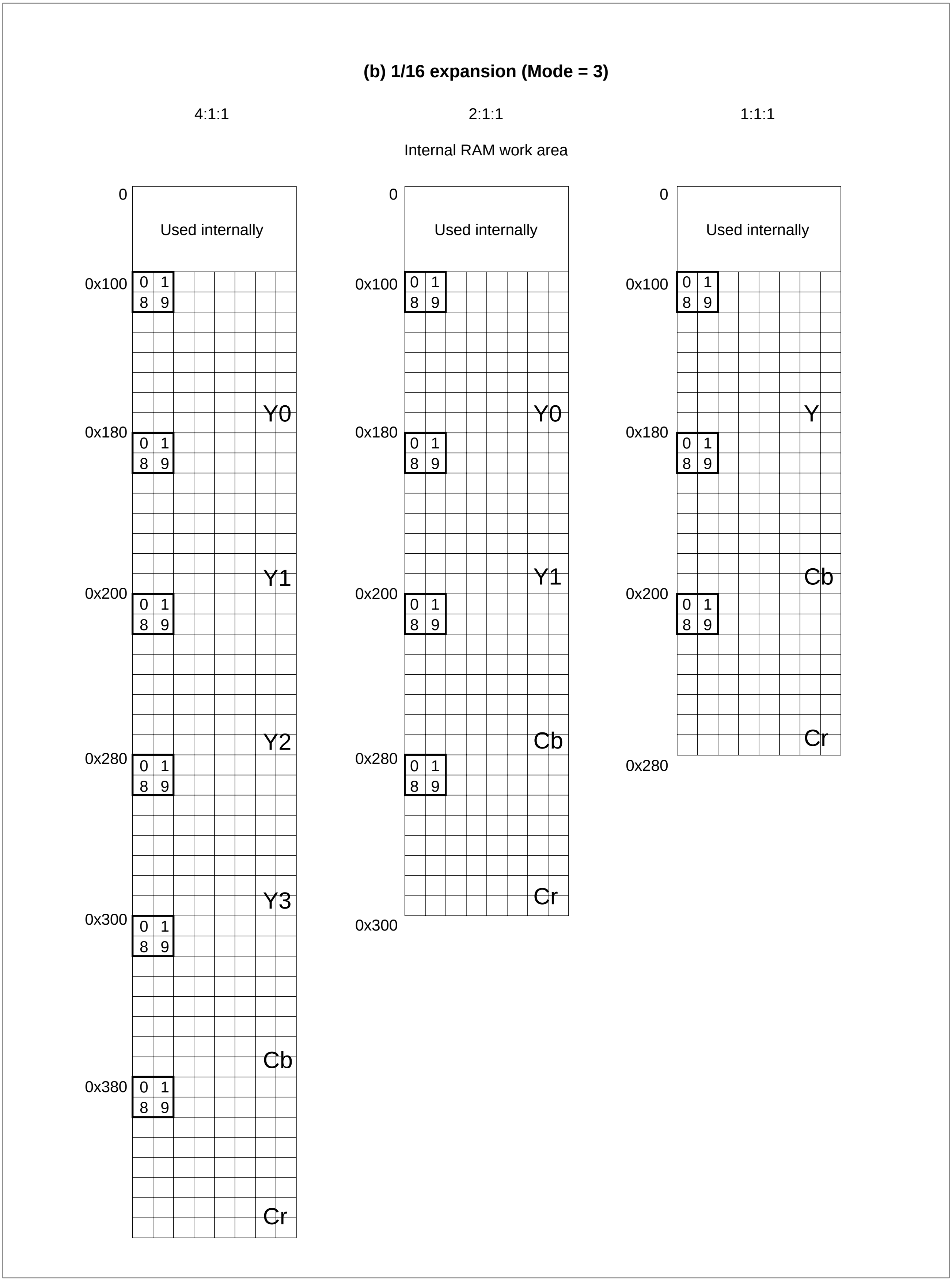
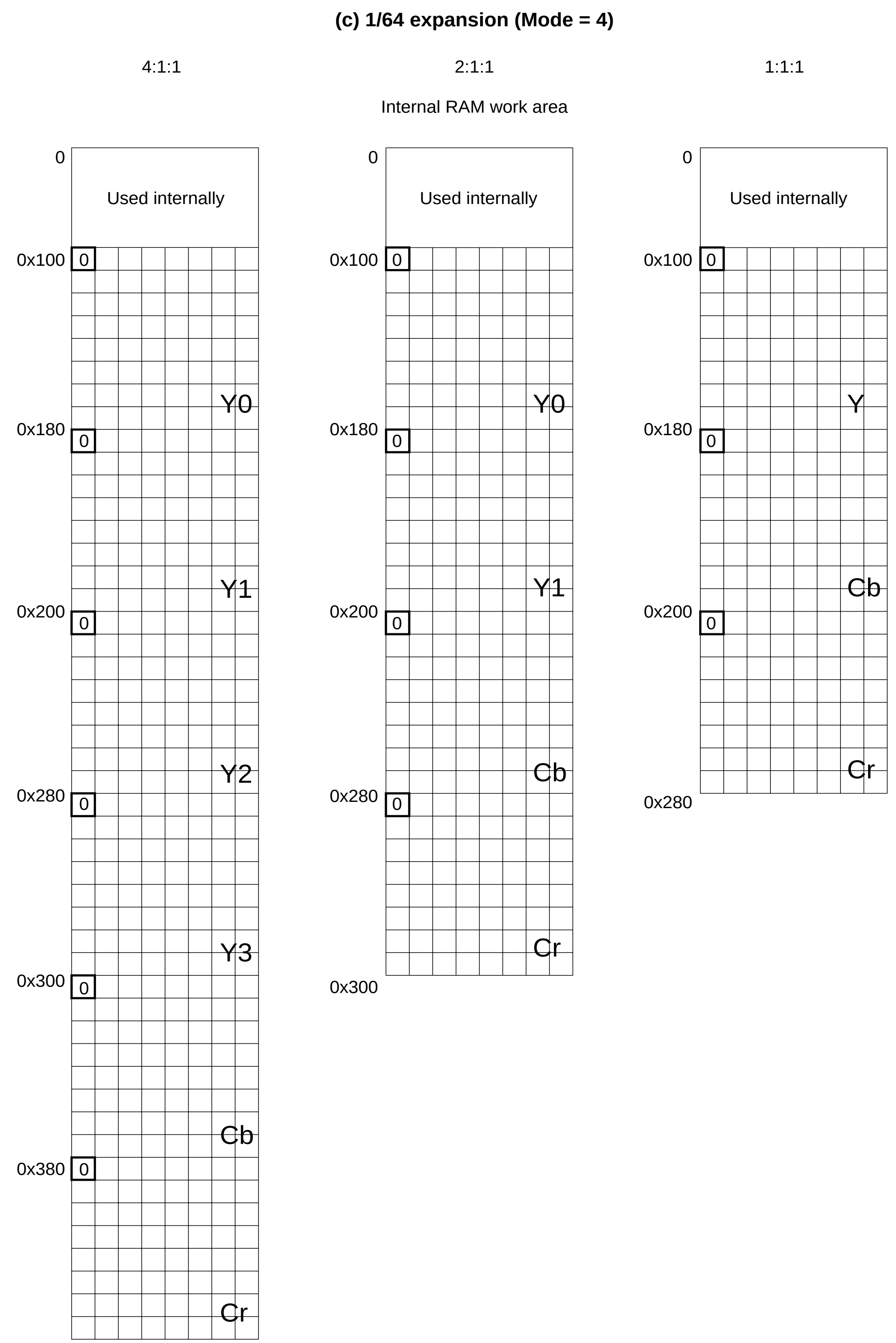


Figure 2-45. Image Data in Reduced Expansion Mode (AP705100-B03) (3/3)



2.6.4 Function Required for Customization

The functions required for compression processing are those that are stored in the locations to which the data of the internal RAM work area are assigned by 1 MCU from VRAM. Different functions must be created depending on the sampling ratio.

Table 2-47. Functions for Compression to Be Customized

Function name	Sampling ratio
void jpeg_getMCU22 (CJINFO*jinfo)	4:1:1 (H:V = 2:2)
void jpeg_getMCU41 (CJINFO*jinfo)	4:1:1 (H:V = 4:1)
void jpeg_getMCU21 (CJINFO*jinfo)	2:1:1 (H:V = 2:1)
void jpeg_getMCU11 (CJINFO*jinfo)	1:1:1 (H:V = 1:1)

For basic expansion, more functions must be created because reduction modes are used.

Table 2-48. Functions for Basic Expansion to Be Customized

Function name	Sampling ratio
void jpeg_putMCU221 (DJINFO*jinfo)	4:1:1 (H:V = 2:2)
void jpeg_putMCU411 (DJINFO*jinfo)	4:1:1 (H:V = 4:1)
void jpeg_putMCU211 (DJINFO*jinfo)	2:1:1 (H:V = 2:1)
void jpeg_putMCU111 (DJINFO*jinfo)	1:1:1 (H:V = 1:1)
void jpeg_putMCU222 (DJINFO*jinfo)	4:1:1 (H:V = 2:2) (reduced to 1/4)
void jpeg_putMCU412 (DJINFO*jinfo)	4:1:1 (H:V = 4:1) (reduced to 1/4)
void jpeg_putMCU212 (DJINFO*jinfo)	2:1:1 (reduced to 1/4)
void jpeg_putMCU112 (DJINFO*jinfo)	1:1:1 (reduced to 1/4)
void jpeg_putMCU224 (DJINFO*jinfo)	4:1:1 (H:V = 2:2) (reduced to 1/16)
void jpeg_putMCU414 (DJINFO*jinfo)	4:1:1 (H:V = 4:1) (reduced to 1/16)
void jpeg_putMCU214 (DJINFO*jinfo)	2:1:1 (reduced to 1/16)
void jpeg_putMCU114 (DJINFO*jinfo)	1:1:1 (reduced to 1/16)
void jpeg_putMCU228 (DJINFO*jinfo)	4:1:1 (H:V = 2:2) (reduced to 1/64)
void jpeg_putMCU418 (DJINFO*jinfo)	4:1:1 (H:V = 4:1) (reduced to 1/64)
void jpeg_putMCU218 (DJINFO*jinfo)	2:1:1 (reduced to 1/64)
void jpeg_putMCU118 (DJINFO*jinfo)	1:1:1 (reduced to 1/64)

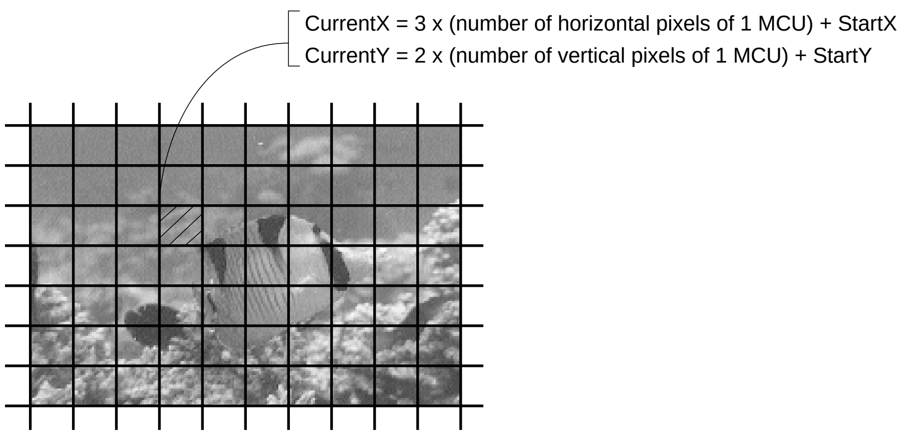
The argument of each function is only the JPEG structure (CJINFO for compression and DJINFO for expansion). Particularly, when this structure is described in assembler, save the contents of r20 to r29 and sp before the function is used, and restore these contents after the function has been executed, in compliance with the C conventions.

The following information is required to create each function.

Table 2-49. Information Required for Customization

Member	Meaning
VRAM_Bptr	First address of VRAM
CurrentX (short type)	Horizontal pixel coordinate of VRAM
CurrentY (short type)	Vertical pixel coordinate of VRAM
IRAM_Buff_Bptr	First address of internal RAM work area

Figure 2-46. CurrentX/CurrentY



[MEMO]

* CHAPTER 3 PROGRESSIVE-SUPPORTING ADDITIONAL LIBRARY SPECIFICATIONS

The additional library conducts an adaptive test on the image data of ISO/IEC 10918-2 and it confirms that the library correctly expands test data A, C, E, G, and K.

3.1 FUNCTION

This section explains the major functions of the expansion processing that can be implemented by using the additional libraries of the AP705100-B03.

The operation precision of additional expansion processing is the same as that for the basic library. See **Section 2.1.3**.

3.1.1 Sampling of Progressive Format and MCU

The minimum unit in which JPEG performs processing is called an MCU (Minimum Coded Unit). An MCU, separated into Y, Cb, and Cr in units of 8 x 8 pixels, is called a block (see **Section 1.2.1 (3)**).

If the number of color components is three, 16 x 16 pixels constitute one MCU at a sample ratio of 4:1:1 (H:V = 2:2). The MCU at this sampling ratio consists of four blocks of the Y (luminance) component, one block of the Cb (chrominance) component, and one block of the Cr (chrominance) component.

The size of the MCU and the number of blocks are determined by the Hi and Vi values included in the SOF marker segment, as follows:

Max number of horizontal pixels of MCU (H_0, H_1, ...) x 8
Max number of vertical pixels of MCU (V_0, V_1, ...) x 8
Σ_iHi x Vi $\leq$ 10 (limit by ISO/IEC 10918-1)
Σ_iHi x Vi $\leq$ 20 (limit of extended format by ISO/IEC 10918-3)

For example, the values of Hi and Vi are as follows for a sampling ratio of 4:1:1 (H:V = 2:2):
$H_0 = 2$, $V_0 = 2$
$H_1 = 1$, $V_1 = 1$
$H_2 = 1$, $V_2 = 1$

If this is substituted into the above expression as follows, then the size of the MCU is 16 x 16 pixels.
Max (H_0, H_1, H_2) x 8 = 16
Max (V_0, V_1, V_2) x 8 = 16

Next, suppose that the number of components is four and that a complicated sampling ratio, such as 1:2:3:4, is used. The values of Hi and Vi are as follows:
$H_0 = 1$, $V_0 = 1$
$H_1 = 1$, $V_1 = 2$
$H_2 = 3$, $V_2 = 1$
$H_3 = 1$, $V_3 = 4$

If this is substituted into the above expression as follows, the size of MCU is 24 x 32 pixels.

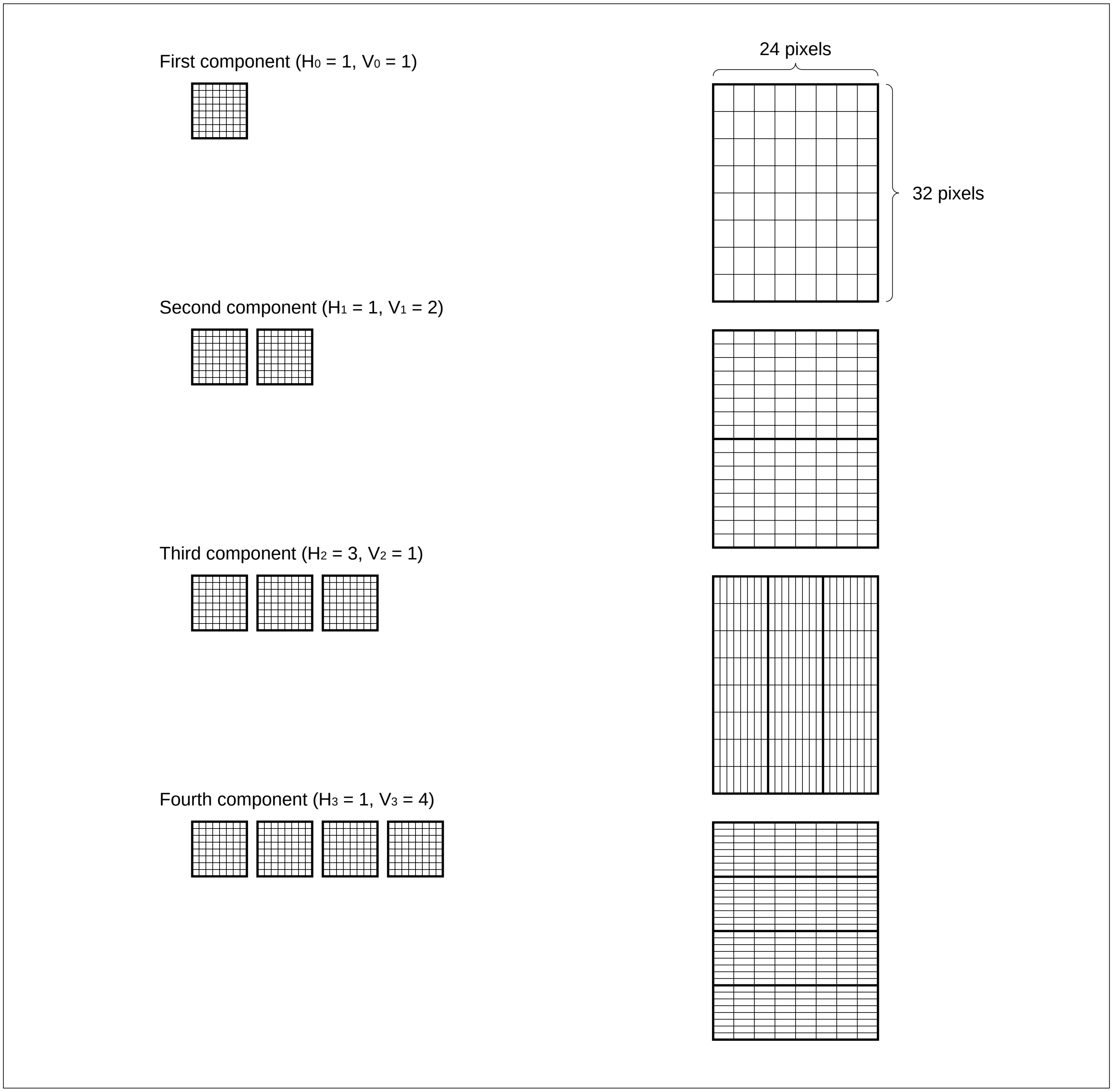
Max (H_0, H_1, H_2, H_3) x 8 = 24

Max (V_0, V_1, V_2, V_3) x 8 = 32

At this time, the first component (H_0 = 1, V_0 = 1) is expanded into 24 x 32 pixels.

Because the second component (H_1 = 1, V_1 = 2) is of 2 blocks and 24 x 32 pixels, the first block is expanded to 24 x 16 pixels. Similarly, the first block of the third component (H_2 = 3, V_2 = 1) is expanded to 8 x 32 pixels, and the first block of the fourth component (H_3 = 1, V_3 = 4) is expanded to 24 x 8 pixels.

Figure 3-1. Sampling and MCU (at a sampling ratio of 1:2:3:4)



3.1.2 Color Space

The AP705100-B03 additional library specifies the color space as follows:

- Monochrome format: Luminance (conforms to JFIF Standard)
- Three-color format: YCbCr (conforms to JFIF Standard)
- Four-color format: CMYK, YCCK

If the input JPEG file is in four-color format, the CMYK or YCCK format is automatically identified based on the information in the file header, and processing is executed with the following expressions:

(1) In CMYK format

C: First component, M: Second component, Y: Third component, K: Fourth component, R, G, B: (R:G:B) of output

```
R = C + K; if (R < 0) R = 0; if (R > 0xFF) R = 0xFF;
G = M + K; if (G < 0) G = 0; if (G > 0xFF) G = 0xFF;
B = Y + K; if (B < 0) B = 0; if (B > 0xFF) B = 0xFF;
```

Remark If the VRAM format is YCbCr instead of RGB, the values of (R:G:B), calculated by this expression, are transformed into (Y:Cb:Cr).

(2) YCCK format

Yin: First component, C1in: Second component, C2in: Third component, Kin: Fourth component , Yout, Cbout, Crout: (Y:Cb:Cr) of output

```
Yout = Kin-Yin;
Cbout = 0xFF-C1in;
Crout = 0xFF-C2in;
```

Remark If the VRAM format is RGB instead of YCbCr, the values (Y:Cb:Cr), calculated by this expression, are transformed into (R:G:B).

3.1.3 Reverse DCT Transformation of Progressive

Sixty-four elements, obtained as a result of DCT transformation of one block consisting of 8 x 8 elements, are called the DCT coefficient.

If the DCT coefficient resulting from DCT transformation is rearranged in zigzag order, low-frequency components and high-frequency components are arranged in that order. Only the first element is called a DC component and indicates the average color level of that block. The other 63 elements are called AC components (for DCT transformation, see **Section 1.2.1 (4)**).

If reverse DCT transformation is performed on the DCT coefficient of the 64 elements, the original image can be restored. If reverse DCT translation is performed on elements with all AC1 to AC63 cleared to 0 and the entire image is restored, a mosaic image in units of 8 x 8 is obtained. From data with only the DC component and AC1 to AC5 components validated and the other components being zero, a blurred image is obtained. This is the basic concept of the progressive algorithm.

3.1.4 Scan

The compressed data of a JPEG file is divided into units called "scans" that start from an SOS segment (for the SOS segment, see **Figure 1-24**).

The SOS segment, which is a scan header, has an Ss area that specifies the start number of the DCT coefficient, and an Se area that specifies the end number of the DCT coefficient.

If the DC component to AC63 component are compressed together, as for base line, Ss = 0 and Se = 63 (= 0x3F). For a scan that progressively compresses only the DC component, Ss = 0 and Se = 0.

In the progressive format, the DCT coefficient is usually divided as follows for each scan for compression.

Only DC component for first scan, AC1 to AC5 for second scan, ...

A method that does not divide each DCT coefficient, even in the same progressive format, is called a spectral section. A method that divides the value of each DCT coefficient into high-order bits and low-order bits is called successive approximation.

Second and subsequent bits of DC component for first scan
Second and subsequent bits of AC1 to AC5 for second scan

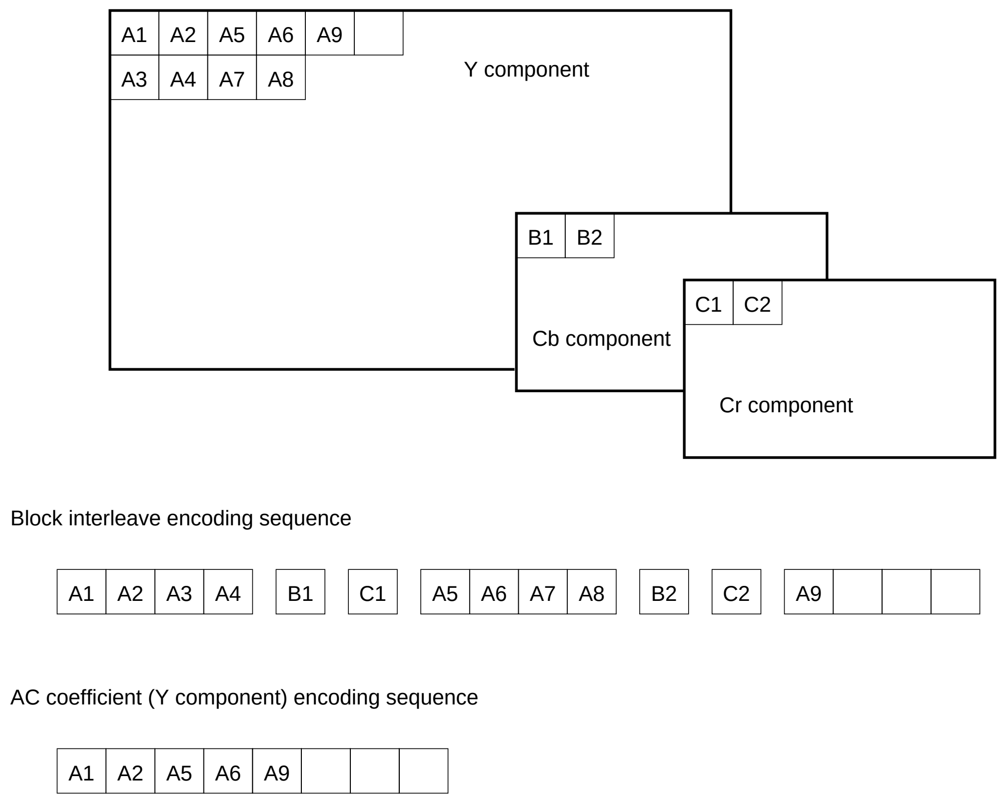
For successive approximation coding, Ah = 0 and Al = 2 is specified for the SOS segment to express 'the second and subsequent bits'. Specify Ah = 2 and Al = 0 to express 'bit 0 to first bit'.

3.1.5 MCU Encoding Sequence

In the progressive format, the DC component and AC components must be encoded in separate scan, and a scan of the DC component must be encoded before a scan of the AC components. For a DC component scan, block interleave that combines the Y, Cb, and Cr components into one scan for encoding is enabled, but each color component must be encoded for AC component scan.

For example, in the format in which block interleave is enabled at a sampling ratio of 4:1:1 (H:V = 2:2), the MCU encoding sequence is as follows:

Figure 3-2. MCU Encoding Sequence (4:1:1 (H:V = 2:2), block interleave format)



Remark ☐ indicates one block.

Only when there is only one color component included in the scan, the scan sequence is from the left to the right, and from the top to the bottom, in block units, regardless of the sampling ratio (conforms to ISO/ IEC 10918-1).

3.1.6 Options for Additional Expansion

The following options are supported for additional expansion:

(1) Forced termination of additional expansion processing

Additional expansion processing under execution can be forcibly terminated.
This option is specified by the JPEGEXINFO structure.

(2) Drawing timing

The stage at which expansion processing drawing is to be performed can be specified.
This option is specified by the JPEGEXINFO structure.

(3) Stuffing bit and stuffing byte

The value of the stuffing bit in the JPEG file can be checked. In addition, whether the stuffing byte in the JPEG file is used can be specified.
These options are specified by the JPEGEXINFO structure.

(4) Number of passes for expansion

The number of passes for additional expansion processing can be specified.
This option is specified by the JPEGEXINFO structure.

(5) DNL marker

Whether the existence of a DNL marker (redefinition of the number of lines) is approved can be specified.
This option is specified by the JPEGEXINFO structure.

(6) Zooming in/out of image

Whether an image is zoomed in or out can be specified.
This option is specified by the JPEGEXINFO structure.

(7) Clipping

Clipping for expansion can be set in pixel units.
This option is specified by the JPEGEXVIDEO structure.

3.2 LINKING ADDITIONAL LIBRARY

A library can be selected for linking.

Table 3-1 lists the libraries that can be selected by an additional library. For how to select a library, refer to makefile of the sample.

Table 3-1. Libraries That Can Be Specified for Link

Library name	Contents of library
libjprg.a	Additional library main entity
libjprgd.a	Debug version of libjprg.a
libjprog.a	Library for resolving symbols of putMCU

(1) Ordinary library specification

NEC library

```
Id830 -o hehe.elf -D dfile $(OBJ) ../../lib830/libjprg.a ../../lib830/libjprog.a
```

GHS library

```
Ix -o hehe.elf @make.lnk $(OBJ) ../../lib830/libjprg.a ../../lib830/libjprog.a
```

(2) Debug library specification

Specify libjprgd.a instead of libjprg.a when using a debug library.

NEC library

```
Id830 -o hehe.elf -D dfile $(OBJ) ../../lib830/libjprgd.a ../../lib830/libjprog.a
```

GHS library

```
Ix -o hehe.elf @make.lnk $(OBJ) ../../lib830/libjprgd.a ../../lib830/libjprog.a
```

(3) Specification when not using the JPEGEXputMCU function

The symbol used for the JPEGEXputMCU function is defined by libjprog.a.
When the JPEGEXputMCU function is not used, it is not necessary to specify libjprog.a by linker.
There are two cases in which the JPEGEXputMCU function is not used, as follows.

(a) If the JPEGEXputMCU function is not used and if the following function is defined by the source file, it is not necessary to specify libjprog.a by the linker.

- jpeg_putMCU221
- jpeg_putMCU411
- jpeg_putMCU211
- jpeg_putMCU111

(b) To use the putMCU function of the basic library, it is not necessary to specify libjprog.a by the linker. Specify the necessary options by using member Policy (see **Section 3.4.3 (3)**) of the JPEGEXINFO structure, and specify the following for link. In this example, the putMCU library of YCbCr is used.

NEC library

```
ld830 -o hehe.elf -D dfile $(OBJ) ../../lib830/libjprg.a ../../lib830/libdy.a
```

GHS library

```
lx -o hehe.elf @make.lnk $(OBJ) ../../lib830/libjprg.a ../../lib830/libjdy.a
```

3.3 STRUCTURE OF ADDITIONAL LIBRARY

This section explains the structure used for the expansion processing of the additional library.

3.3.1 JPEGEXINFO Structure

The JPEGEXINFO structure is used to set the parameters for additional expansion processing. The first address of this structure is passed to the additional expansion main function as an argument. For how to set the members of the JPEGEXINFO structure, see **Section 3.4.3**.

Table 3-2. JPEGEXINFO Structure

Member	Type	Contents	IN/OUT
TaskID	int	ID number of task	IN
Mode	int	Selection of ordinary expansion processing/forced termination of expansion processing	IN
Policy	int	Setting of options for expansion processing	IN
ratio	int	Setting image zoom in/out ratio	IN
ErrorState	int	Error status number	OUT
Work	struct JPEGEXWORK	JPEGEXWORK structure first address	IN
Video	struct JPEGEXVIDEO	JPEGEXVIDEO structure first address	IN
Inf	struct JPEGEXFrmINFO	JPEGEXFrmINFO structure first address	OUT

3.3.2 JPEGEXWORK Structure

With the JPEGEXWORK structure, specify a work area that can be used by the additional library. Set the first address of this structure in member Work of the JPEGEXINFO structure.

For how to set the members of the JPEGEXWORK structure, see **Section 3.4.4**.

Table 3-3. JPEGEXWORK Structure

Member	Type	Contents	IN/OUT
Work1	unsigned int	Work area first address	IN
Work1Len	unsigned int	Work area size (bytes)	IN
Work1Used	unsigned int	Size of work area used (bytes)	OUT
Work2	unsigned int	Work area first address	IN
Work2Len	unsigned int	Work area size (bytes)	IN
Work2Used	unsigned int	Size of work area used (bytes)	OUT

3.3.3 JPEGEXVIDEO Structure

The JPEGEXVIDEO structure performs the setting related to drawing. Specify the first address of this structure in member Video of the JPEGEXINFO structure.

Specify a value that specifies the structure of VRAM, as a member (VRAMxxx) related to VRAM.

To perform clipping during additional expansion processing, an appropriate value must be set in a member (Clipxxx) related to clipping. When clipping is not performed, set the dummy values shown in Table 3-5 in the clipping-related member (Clipxxx).

For how to set the members of the JPEGEXVIDEO structure, see **Section 3.4.5**.

To create the JPEGEXputMCU function, set the dummy values shown in Table 3-5 in each member of the JPEGEXVIDEO structure.

Table 3-4. JPEGEXVIDEO Structure

Member	Type	Contents	IN/OUT
VRAMAddress	unsigned char*	VRAM first address	IN
VRAMWidth	int	Horizontal width of VRAM	IN
VRAMHeight	int	Vertical width of VRAM	IN
VRAMPixel	int	Address difference of VRAM by one horizontal pixel	IN
VRAMLine	int	Address difference of VRAM by one vertical pixel	IN
VRAMGap0	int	Byte offset of Y pixel (or R pixel)	IN
VRAMGap1	int	Byte offset of Cb pixel (or G pixel)	IN
VRAMGap2	int	Byte offset of Cr pixel (or B pixel)	IN
ClipStartX	int	Clipping start position (X coordinate) Set dummy value 0 when not performing clipping.	IN
ClipStartY	int	Clipping start position (Y coordinate) Set dummy value 0 when not performing clipping.	IN
ClipWidth	int	Clipping horizontal size (pixel) Set dummy value 0x7FFFFFFF when not performing clipping.	IN
ClipHeight	int	Clipping vertical size (pixel) Set dummy value 0x7FFFFFFF when not performing clipping.	IN

Table 3-5. Dummy Set Value of JPEGEXVIDEO Structure

Member	Dummy value	Member	Dummy value
VRAMAddress	Need not be set	VRAMGap1	Need not be set
VRAMWidth	0x7FFFFFFF	VRAMGap2	Need not be set
VRAMHeight	0x7FFFFFFF	ClipStartX	0
VRAMPixel	Need not be set	ClipStartY	0
VRAMLine	Need not be set	ClipWidth	0x7FFFFFFF
VRAMGap0	Need not be set	ClipHeight	0x7FFFFFFF

3.3.4 JPEGEXBUFF Structure

The JPEGEXBUFF structure specifies a JPEG buffer into which a JPEG file is to be stored. The first address of this structure is passed to the JPEG file acquisition function (JPEGEXGetJpegStream) as an argument. For how to set the members of the JPEGEXBUFF structure, see **Section 3.5.1**.

Table 3-6. JPEGEXBUFF Structure

Member	Type	Contents	IN/OUT
TaskID	int	Task ID number	OUT (can be overwritten)
JPEGBUFF	unsigned char*	First address of JPEG buffer	IN
JPEGBUFFLEN	unsigned int	Size of JPEG buffer (bytes)	IN

3.3.5 JPEGEXMCUSTR Structure

The additional library sets a parameter that specifies the structure of the MCU buffer, and a parameter related to data output in the JPEGEXMCUSTR structure. Specify the first address of this structure as the fourth argument of the MCU data output function (JPEGEXputMCU) or the first argument of the JPEGEXpset function.

Table 3-7. JPEGEXMCUSTR Structure

Member	Type	Contents	IN/OUT
component	unsigned char	Number of color components 1: Luminance only 3: Three colors of Y, Cb, and Cr 4: Four colors	OUT
adobeflag	char	Output mode of four colors (valid only when four colors is specified) 0: CMYK 1: YCbCr 2: YCCK	OUT
hf [4]	unsigned char	Number of horizontal blocks of MCU buffer	OUT
vf [4]	unsigned char	Number of vertical blocks of MCU buffer	OUT
VRAMAddress	unsigned char*	The set value of the JPEGEXVIDEO structure is stored by the additional library.	OUT
VRAMWidth	int		OUT
VRAMHeight	int		OUT
VRAMPixel	int		OUT
VRAMLine	int		OUT
VRAMGap0	int		OUT
VRAMGap1	int		OUT
VRAMGap2	int		OUT
ClipStartX	int	The size to be actually clipped is stored by the additional library.	OUT
ClipStartY	int		OUT
ClipWidth	int		OUT
ClipHeight	int		OUT
hfMax	unsigned char	Horizontal width of MCU (horizontal size of MCU: hfMax x 8 pixels)	OUT
vfMax	unsigned char	Vertical width of MCU (vertical size of MCU: vfMax x 8 pixels)	OUT

The structure of the MCU buffer is determined by the values set in the JPEGEXMCUSTR structure's members component, hf [4], and vf [4]. Figure 3-3 shows an example.

Figure 3-3. Set Values of JPEGEXMCUSTR Structure Members and MCU Buffer Structure

Set values of JPEGEXMCUSTR structure members (example)
component = 3
hf [0] = vf [0] = 2
hf [1] = vf [1] = hf [2] = vf [2] = 1

MCU buffer

Y first block 128 bytes
Y second block 128 bytes
Y third block 128 bytes
Y fourth block 128 bytes
Cb block 128 bytes
Cr block 128 bytes

3.4 EXECUTING ADDITIONAL EXPANSION PROCESSING

3.4.1 Additional Expansion Main Function

Classification Additional expansion processing
Function name JPEGEXdecode
Format int JPEGEXdecode (struct JPEGEXINFO* JPInfo);
Argument First address of JPEGEXINFO structure
Return value The contents of the return value are shown in Table 3-8.

Table 3-8. Return Values for Additional Expansion Main Function

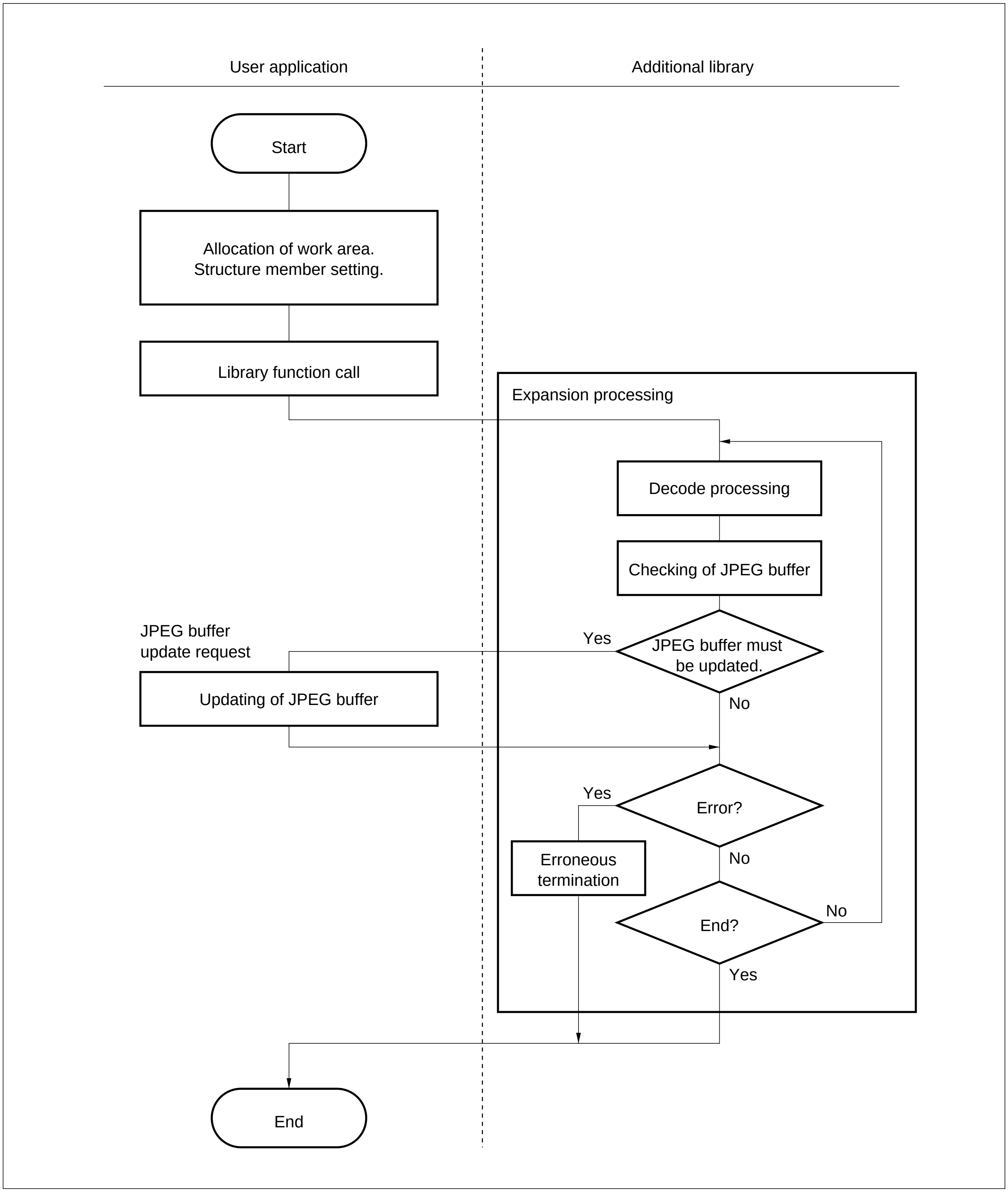
Return value		Contents
Definition name	Numeric value	
DecodeStatusComplete	1	Normal completion
DecodeStatusTerminate	2	Forced termination
DecodeStatusError	-1	Erroneous termination
DecodeStatusNotRunning	-2	Process subject to forced termination is not operating.

Before calling this function, the members of the JPEGEXINFO, JPEGEXWORK, and JPEGEXVIDEO structures must be set.

3.4.2 Additional Expansion Processing Flow

The basic flow of additional expansion processing is shown below.

Figure 3-4. Additional Expansion Processing Flow



3.4.3 Setting of JPEGEXINFO Structure Parameters

Before calling the expansion main function, set the parameters of the JPEGEXINFO structure necessary for additional expansion processing.

Table 3-9. JPEGEXINFO Structure

Member	Type	Contents	IN/OUT
TaskID	int	ID number of task	IN
Mode	int	Selection of ordinary expansion processing/forced termination of expansion processing	IN
Policy	int	Setting of options for expansion processing	IN
ratio	int	Zoom in/out rate setting	IN
ErrorState	int	Error status number	OUT
Work	struct JPEGEXWORK	JPEGEXWORK structure first address	IN
Video	struct JPEGEXVIDEO	JPEGEXVIDEO structure first address	IN
Inf	struct JPEGEXFrmINFO	JPEGEXFrmINFO structure first address	OUT

(1) TaskID

The value of TaskID is used, when two or more tasks are started in a multitask environment, to distinguish one task from another. Because an individual JPEGEXINFO structure is necessary for each task, set a different value for each TaskID. TaskID does not have to be set when a single task is used.

Note that this value is substituted into member TaskID of the JPEGEXBUFF structure.

(2) Mode

This member specifies whether expansion processing is performed normally, or is forcibly terminated.

Table 3-10. Mode Setting of Additional Expansion Processing

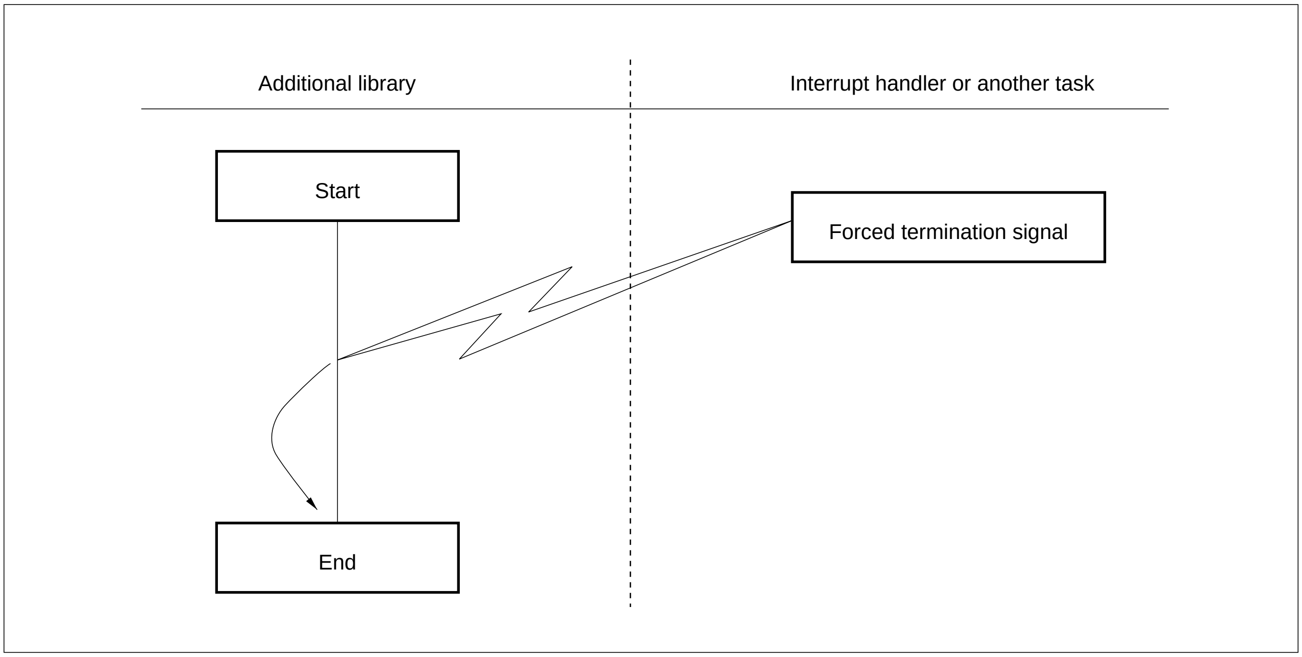
Definition name	Numeric value	Contents
ModeStart	1	Normal expansion mode
ModeTerminate	-1	Expansion processing being executed is forcibly terminated.

Specify ModeStart to start JPEG expansion normally.

```
struct JPEGINFO  JPINFO;
main ()
  {
          JPINFO.Mode = ModeStart;
          JPEGEXdecode (&JPINFO);
}
```

If ModeTerminate is specified, additional expansion processing being executed can be forcibly terminated. By calling the JPEGEXdecode function from an interrupt handler by using the same structure as the JPEGEXINFO structure specified by the library executing an expansion operation, or from another task when the OS is used, a signal that prompts forced termination is sent to the additional library being executed.

Figure 3-5. Forced Termination of Additional Expansion Processing with ModeTerminate Specified



(3) Policy

Policy has option bits in the 2-byte area shown in Figure 3-6.

Policy sets the options listed in Table 3-11.

Figure 3-6. Bit Configuration of Policy

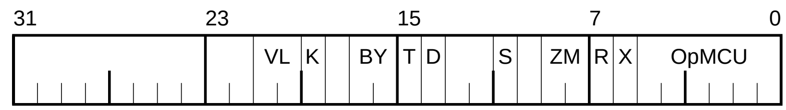


Table 3-11. Option Setting by Policy (1/2)

Option name	Bit position	Bit name	Meaning
VL	21	VideoOutLastOnly	Setting of drawing timing 1: Only end is displayed for progressive without an intermediate result. 0: Intermediate result is also displayed.
	20	LuminanceOutOnly	Setting of drawing timing 1: Only scan with the luminance component updated is displayed. 0: Display for all scan
K	19	BitStuffCheck	Checking of stuffing bit 1: Checked 0: Not checked
BY	17	ByteStuffDisable	Stuffing byte 1: Disabled 0: Enabled
	16	ByteStuffEnable	Stuffing byte (valid when ByteStuffDisable = 0) 1: Permitted up to 0x10000 0: Permitted up to four bytes between segments
T	15	2passEnable	Setting of the number of passes of expansion processing 1: Expanded with two passes 0: Expanded with one pass
D	14	DNLEnable	DNL marker 1: Enabled 0: Disabled
S	11	UsePset	JPEGEXpset function 1: JPEGEXpset function is created by user. 0: Not used
ZM	9	VideoZoomLinear	Zoom in/out of image ZM = 01: Zoomed in/out 11: Zoomed in/out 10: Zoomed in/out with liner filter 00: Not zoomed in/out (expanded at a multiple of 1)
	8	VideoZoomNormal	

Table 3-11. Option Setting by Policy (2/2)

Option name	Bit position	Bit name	Meaning
R	7	PutMCURGB	Setting of image output 1: Output as RGB 0: Output as YCbCr
X	6	UseExPutMCU	Use of JPEGEXputMCU function 1: Used 0: Not used
OpMCU	5	UsePutMCUOnly	User-created functions other than putMCU 1: Not expanded 0: Expanded
	4	UsePutMCU	User-created putMCU function 1: Used 0: Not used
	3	UsePutMCU22	User-created putMCU22 function 1: Used 0: Not used
	2	UsePutMCU41	User-created putMCU41 function 1: Used 0: Not used
	1	UsePutMCU21	User-created putMCU21 function 1: Used 0: Not used
	0	UsePutMCU11	User-created putMCU11 function 1: Used 0: Not used

Remark The JPEGEXputMCU function of the additional library can be directly rewritten and the additional library can be customized without using the UsePutMCU option (without using the putMCUxxx library created by the user with the basic library). For how to do this, see **Section 3.6**.

(a) VideoOutLastOnly/LuminanceOutOnly (VL options)

These options specify the drawing timing.

When the VideoOutLastOnly option is specified, drawing is not performed during expansion. Instead, it is performed once expansion has ended.

The LuminanceOutOnly option is valid when VideoOutLastOnly = 0 (LuminanceOutOnly is not referenced when VideoOutLastOnly = 1).

When LuminanceOutOnly = 0, drawing is performed for each scan in which the luminance component has been updated.

When LuminanceOutOnly = 1, drawing is performed in each scan.

Figure 3-7. Drawing Timing of Baseline Format

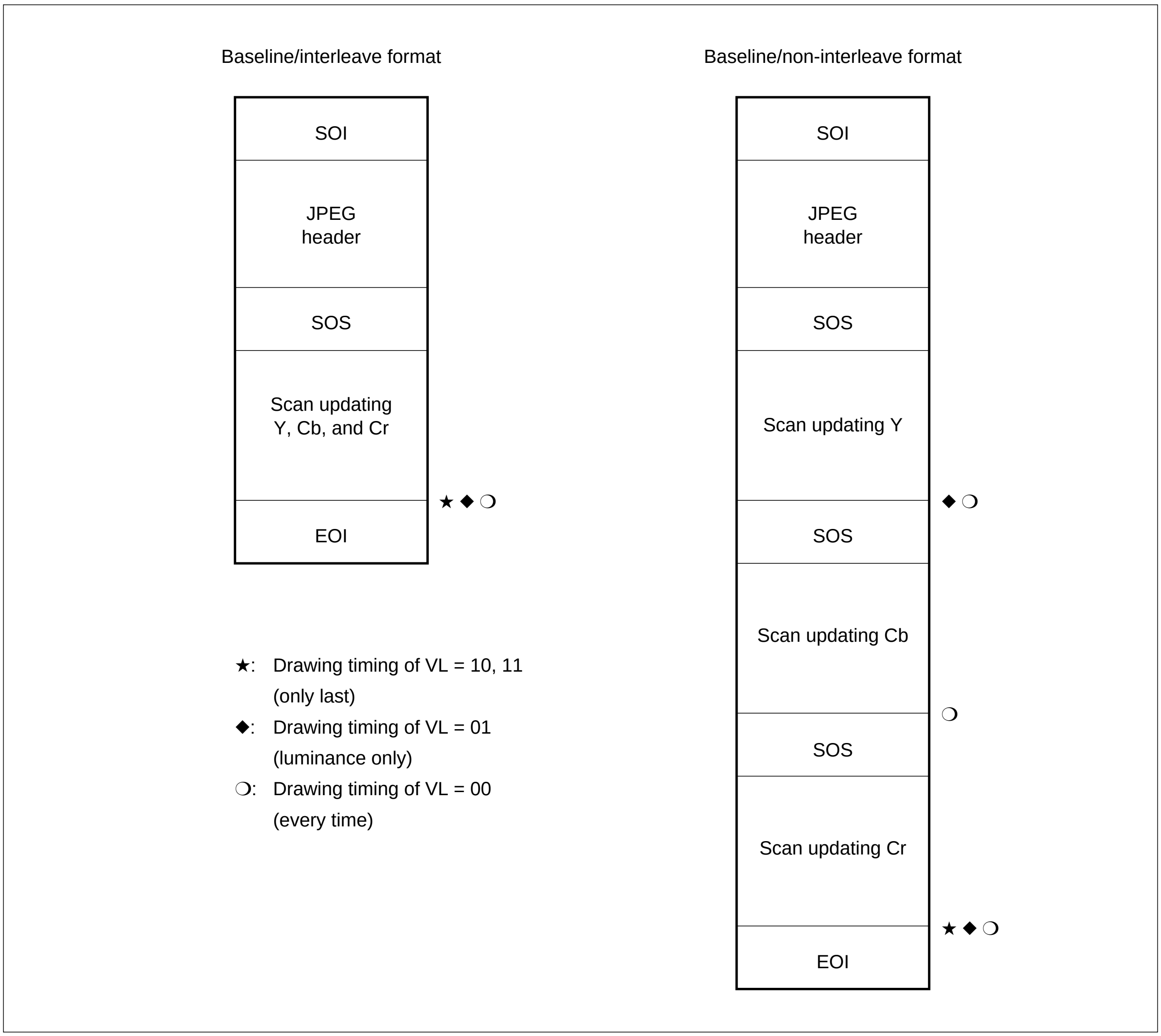
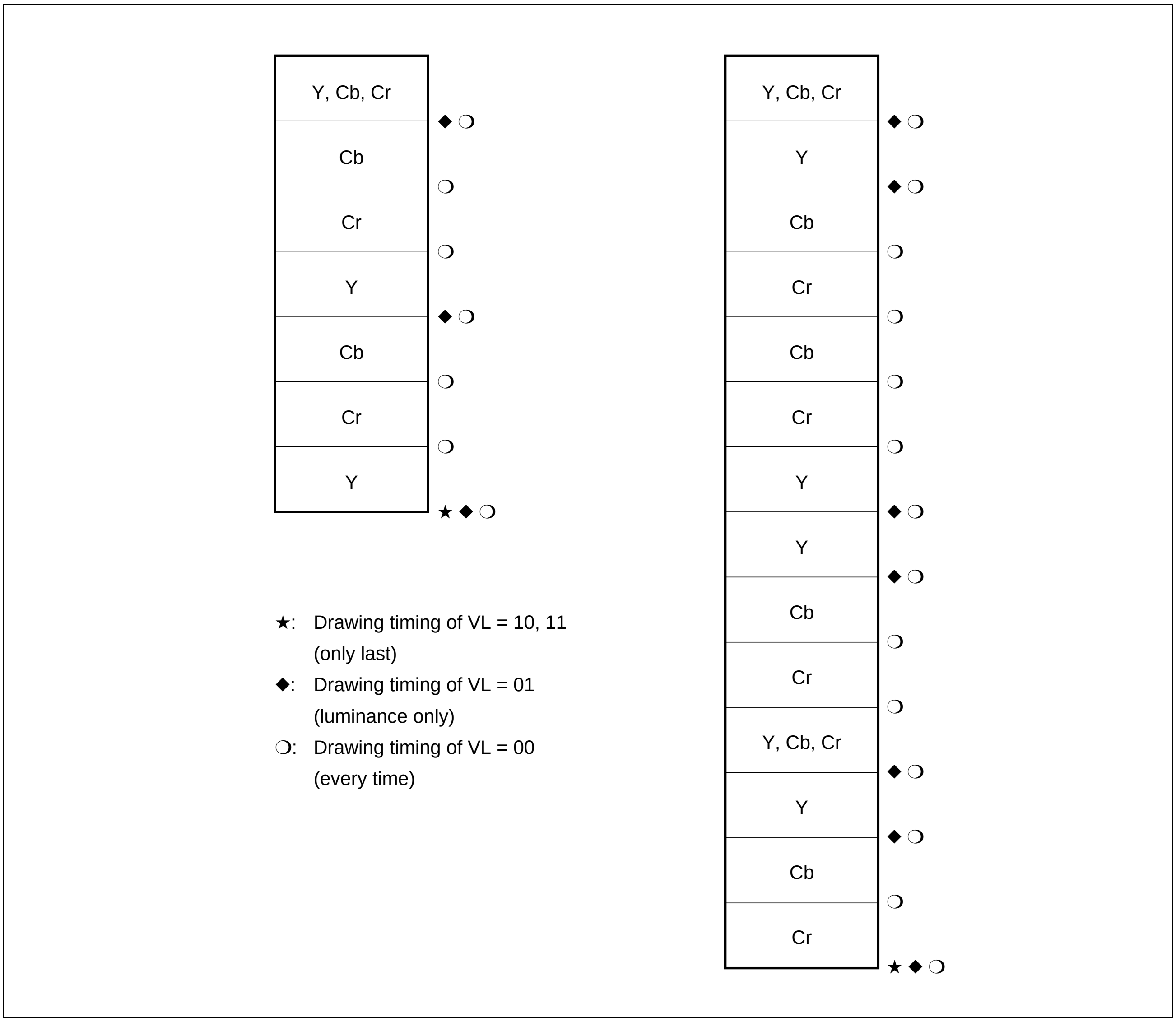


Figure 3-8. Progressive Format Drawing Timing



(b) BitStuffCheck (K option)

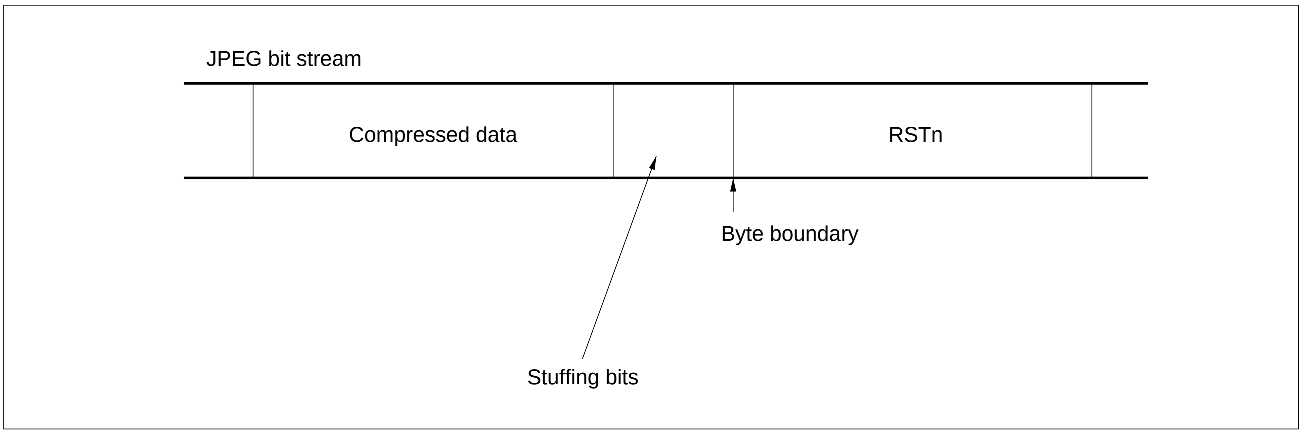
This option specifies whether the stuffing bit is checked.

The compressed data of a JPEG file is processed in bit units, and markers such as SOF, DHT segment, and RSTn are processed in byte units. Consequently, a gap of 1 to 7 bits may be created at a portion where compressed data is changed to a marker (such as EOI, SOS, RSTn, and DNL). This gap is called the stuffing bits, and ISO/IEC 10918-1 stipulates that the values of these stuffing bits be '1'.

If BitStuffCheck = 1, the additional library checks whether the stuffing bit in the JPEG file is '1' or '0'. If a bit that is '0' has been found as a result of this check, warning processing is performed.

Usually, there is no problem regardless of whether the stuffing bit of the JPEG file is '1' or '0'.

Figure 3-9. Stuffing Bit



(c) ByteStuffDisable/ByteStuffEnable (BY option)

A byte filling the gap between the segments in a JPEG file is called a stuffing byte. For example, if 1 byte of 0x00 exists between an SOI segment and the subsequent APP0 segment, this stuffing byte is meaningless for the JPEG file. However, ISO/IEC 10918-1 does not specially stipulate the existence of a stuffing byte in the JPEG file.

This option is used by the user to determine whether the existence of a stuffing byte is permitted.

Table 3-12. ByteStuffDisable/ByteStuffEnable (Stuffing Byte) Option

Set value	Meaning
BY = 10 BY = 11	Rejects the existence of a stuffing byte. In this case, if a JPEG file including a stuffing byte is expanded, a "marker error" occurs and the file is erroneously terminated in most cases.
BY = 01	Permits the existence of a stuffing byte up to 0x10000. However, care must be exercised because, if a byte string that may be taken as a marker of the JPEG file exists in the stuffing byte, it may cause a malfunction.
BY = 00	Default. Four stuffing bytes are permitted between segments.

(d) 2passEnable (T option)

This option specifies the number of passes for additional expansion processing.

When 2passEnable = 1, expansion processing is executed with two passes; when 2passEnable = 0, it is executed with one pass. However, if DNLEnable = 1 (see **(e)** in **Section 3.4.3 (3)**), 2passEnable = 1 is unconditionally assumed.

The differences between expansion processing with one pass and that with two passes are shown in Table 3-13.

Table 3-13. Differences in Expansion Processing Because of Number of Passes

Item	One pass	Two passes
Work area size	Must be large	May be small
Example) 4:1:1 (640 x 480 pixels)	Approx. 1M bytes	Approx. 5K bytes
Example) 1:1:1 (640 x 480 pixels)	Approx. 2M bytes	
Execution time	Short	Long
Function limit	None	• JPEG buffer cannot be updated (processing is stopped after data in the JPEG buffer has been expanded). • The Huffman table cannot be defined in duplicate (expansion processing is stopped if the table has been defined in duplicate).

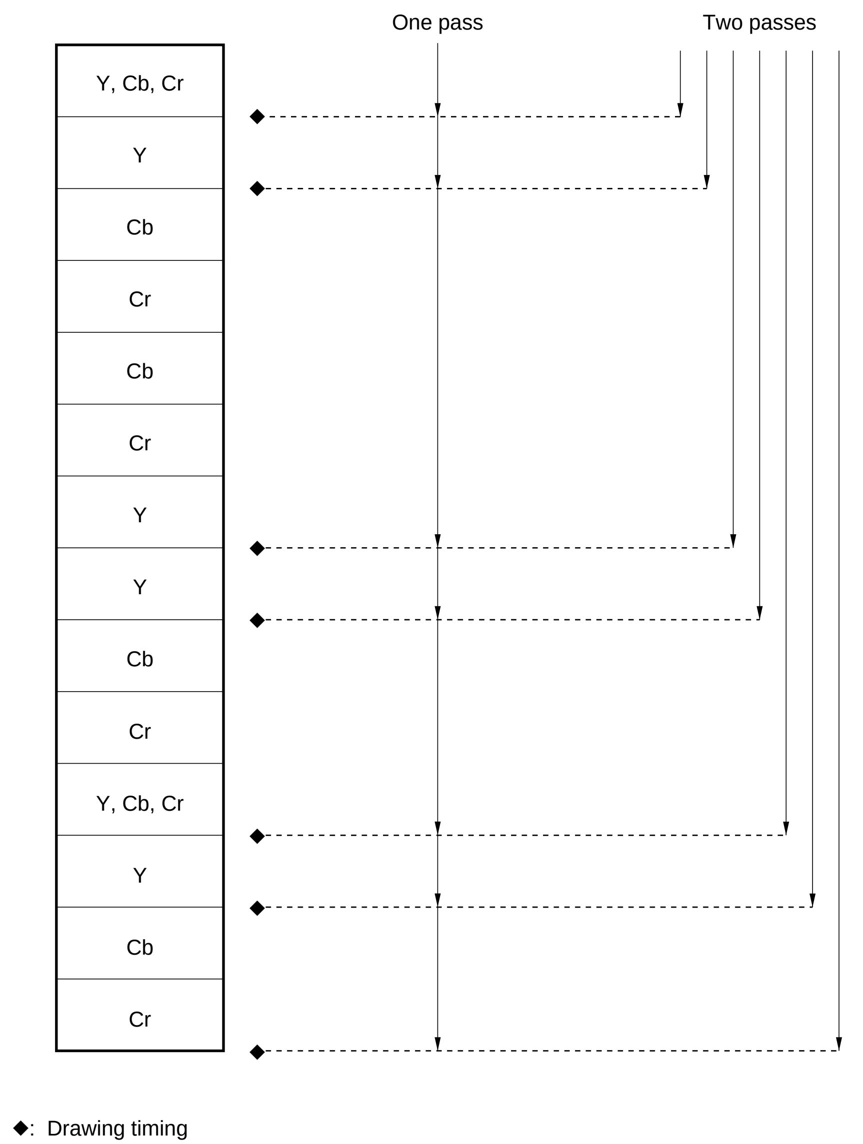
Remark The example of the work area size is for reference only.

The work area size is set by the JPEGEXWORK structure. If the number of passes is two, the work area size must be about 5K bytes regardless of the size of the image to be expanded. If only one pass is used, a very large work area is necessary because all the expanded DCT coefficients must be saved.

If a sufficient work area is not obtained when the number of passes is set to 1, the additional library automatically selects two-pass mode.

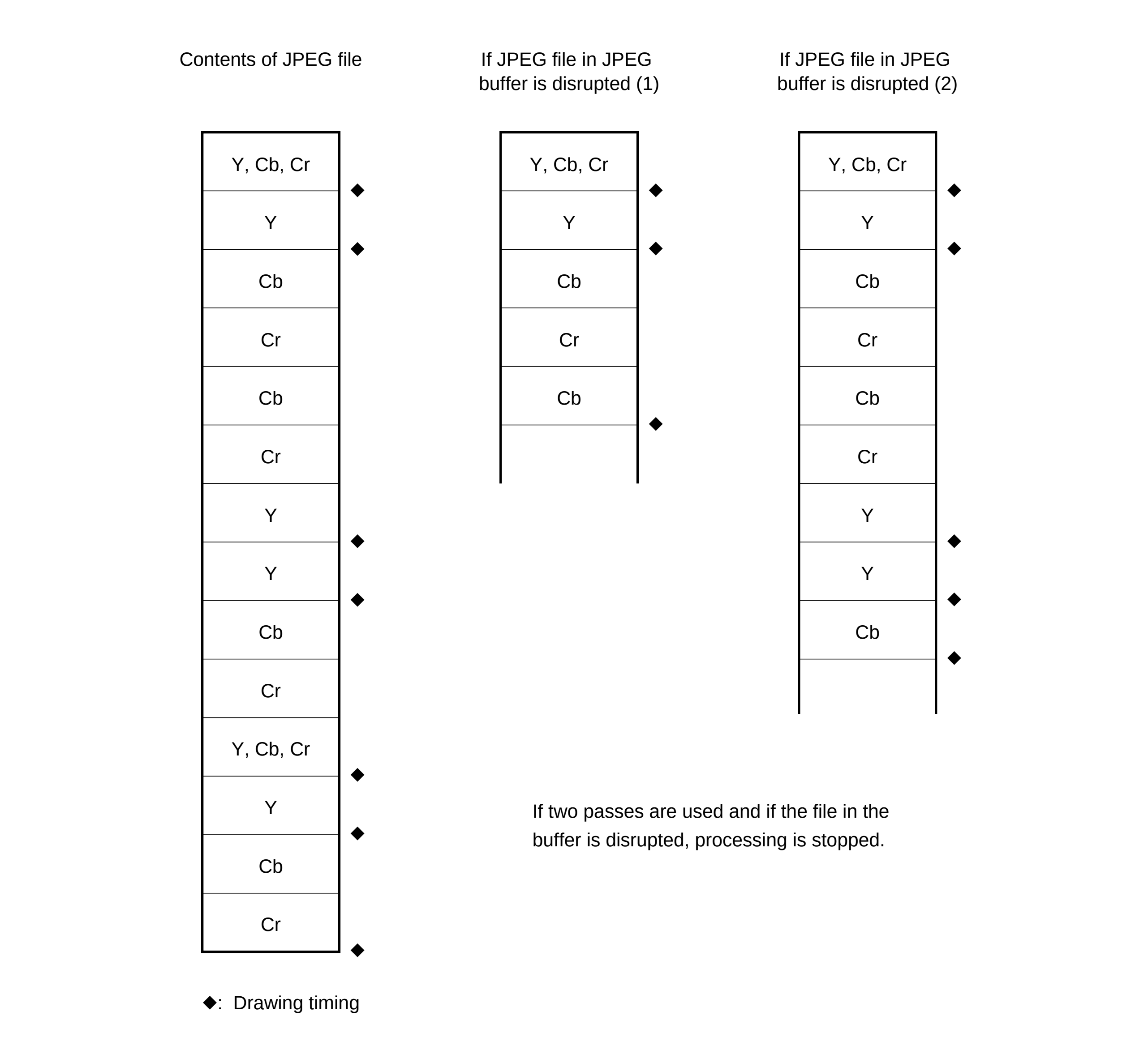
When two passes are used, the processing speed drops substantially because the compressed data are traced from the beginning of the JPEG file for drawing. In addition, the display speed of the image also drops because compressed data is decoded while drawing is performed.

Figure 3-10. Number of Passes for Additional Expansion Processing and Drawing Timing



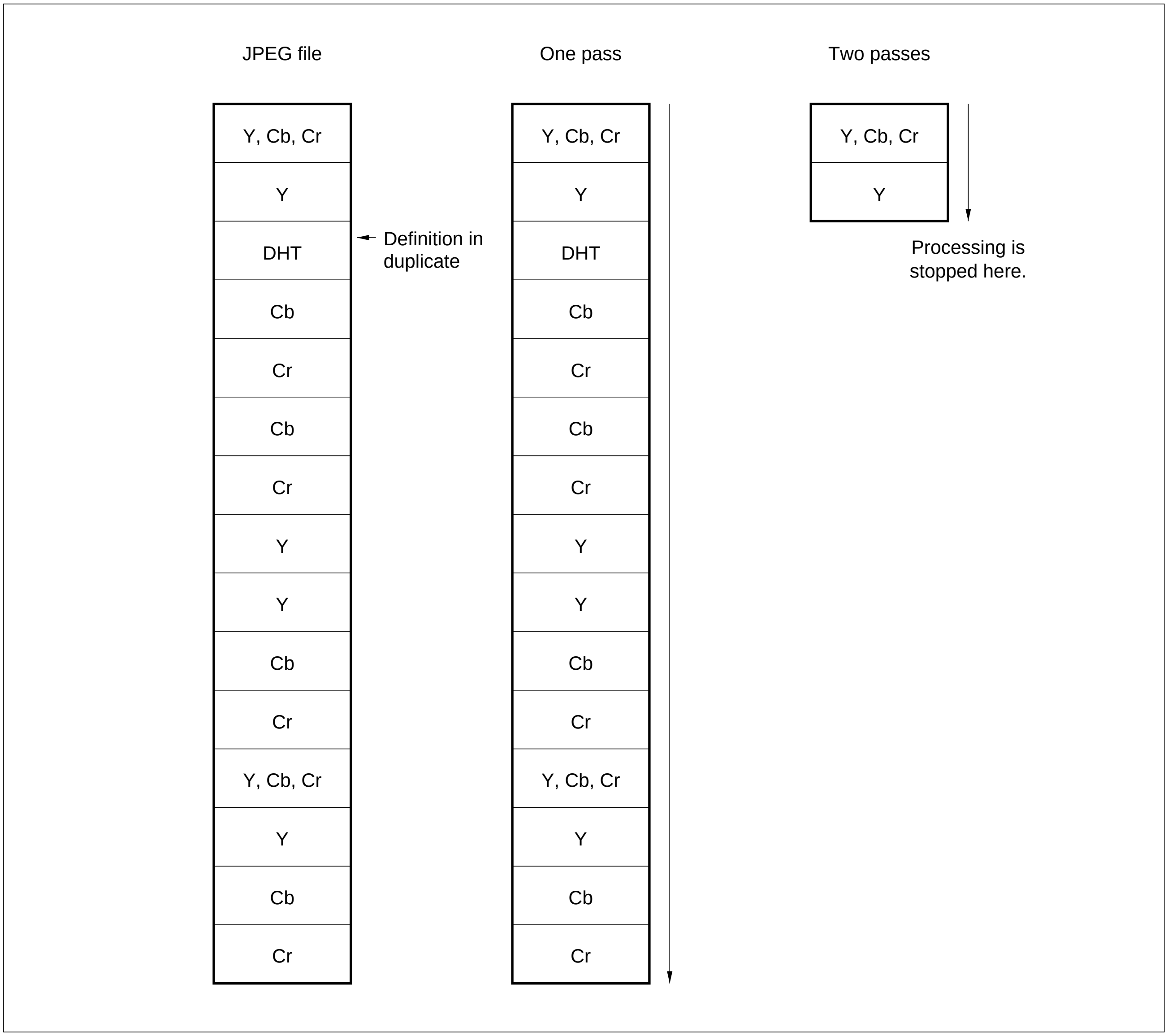
If the specified JPEG file is too large for the JPEG buffer, the contents of the JPEG buffer cannot be exchanged and expansion processing cannot be continued when two passes are used. In this case, the processing is stopped as soon as the JPEG file in the JPEG buffer specified first has been expanded.

Figure 3-11. Expansion Processing if JPEG File in JPEG Buffer is Disrupted (Two passes)



If a Huffman table having the same ID number is defined in duplicate, expansion continues, provided the size of the work area permits, when one pass is used. When two passes are used, however, expansion is performed up to the location of the duplicated definition, and no further expansion is executed.

Figure 3-12. Differences in Expansion Processing Depending on Number of Passes When Huffman Table Is Defined in Duplicate



(e) DNLEnable (D option)

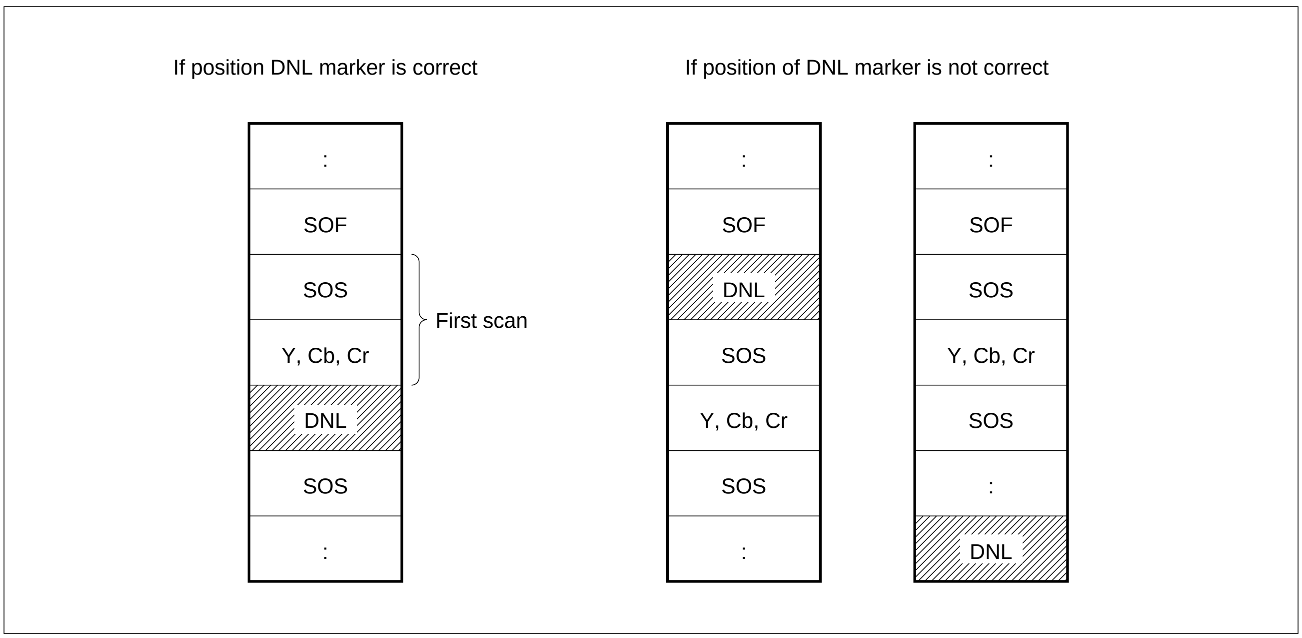
This option enables or disables the expansion of a JPEG file including a DNL marker (re-definition of the number of lines). When DNLEnable = 1, the JPEG file including a DNL marker can be expanded. When DNLEnable = 0, the JPEG file including a DNL marker cannot be expanded.

When DNLEnable = 1, expansion is forcibly performed with two passes (the setting of the 2passEnable option is ignored).

The DNL segment corrects the value of Y (number of vertical pixels of image) specified in the SOF marker. It is specified that the position of the DNL marker is immediately after the first scan (conforms to ISO/IEC 10918-1). If this is violated, an error occurs with the additional library.

The DNL marker is seldom used in an ordinary JPEG file.

Figure 3-13. Position of DNL Marker in JPEG File



(f) UsePset (S option)

This option is set when the user customizes the JPEGEXpset function (see **Section 3.6**).

To customize the JPEGEXpset function, set UsePset to 1.

When UsePset = 0, the additional library may dynamically change the pset function to a module expanded in line (dynamically selects a function according to all the set options and the type of the JPEG file to be expanded).

If use of the putMCU function of the user-customized basic library is specified (UsePutMCU = 1 (see **(j)** in **Section 3.4.3 (3)**), and if the basic library is called, the JPEGEXpset function is not called even when UsePset = 1.

(g) VideoZoomLinear/VideoZoomNormal (ZM option)

These options specify the zoomed expansion of an image. This function is implemented by the putMCU function of the AP705100-B03 additional library. The operation is not guaranteed if the putMCU function of the basic library or user-customized putMCU function is used.

VideoZoomLinear is valid only when VideoZoomNormal = 0.

Table 3-14. VideoZoomLinear/VideoZoomNormal (zoomed expansion) Options

Set Value	Meaning
ZM = 01 ZM = 11 (VideoZoomNormal)	Zooms in. As the multiple, member ratio ÷ 8 of the JPEGEXINFO structure is used.
ZM = 10 (VideoZoomLinear)	Zooms in in the form of linear primary interpolation. As the multiple, member ratio ÷ 8 of the JPEGEXINFO structure is used. Expansion in the linear primary interpolation form takes longer than the VideoZoomNormal form. Linear primary interpolation is a type of filter. Because processing is performed in MCU units, the MCU boundary may be conspicuous, though inside the MCU is smooth.
ZM = 00	Does not zoom but performs expansion with a multiple of 1. The value of the member ratio of the JPEGEXINFO structure is ignored.

(h) PutMCURGB (R option)

This option sets the output mode for an image. When PutMCURGB = 1, the image is output in RGB mode, instead of YCbCr mode. When PutMCURGB = 0, the image is output in YCbCr mode.

This function is implemented by the putMCU function of the AP705100-B03 additional library. It is not realized when the putMCU function of the basic library or user-customized putMCU function is used (not affected).

(i) UseExPutMCU (X option)

This option is set when the user customizes the JPEGEXputMCU function.

To customize the JPEGEXputMCU function, set UseExPutMCU to 1 (for customizing the JPEGEXputMCU function, see **Section 3.6**).

When UseExPutMCU = 0, the JPEGEXputMCU function and its alternate function are dynamically selected in the additional library. When UseExPutMCU = 1, the additional library does not perform dynamic selection and always calls the JPEGEXputMCU function.

(j) UsePutMCU (OpMCU option)

Specifying this option enables the additional library to use the putMCU function created by the user by using the customize function of the basic library (see **Section 2.6**).

To use the user-created putMCU function with the additional library, set either the UsePutMCUOnly or UsePutMCU bit to 1, and set the option bit corresponding to the user-created function.

To expand a JPEG file supporting putMCU22, for example, the user-created putMCU function is called by the additional library when UsePutMCUOnly = 1 or UsePutMCU = 1 and UsePutMCU22 = 1.

Table 3-15. UsePutMCU Options

Set value	Meaning
UsePutMCUOnly = 1	Functions other than user-created putMCU are not expanded.
UsePutMCU = 1	User-created putMCU function is used.
UsePutMCU22 = 1	User-created putMCU22 function is used.
UsePutMCU41 = 1	User-created putMCU41 function is used.
UsePutMCU21 = 1	User-created putMCU21 function is used.
UsePutMCU11 = 1	User-created putMCU11 function is used.

If the option (UsePutMCU22 to UsePutMCU11) corresponding to the JPEG file to be expanded is not set when UsePutMCU = 1, the putMCU function of the AP705100-B03 additional library is called. If the option (UsePutMCU22 to UsePutMCU11) corresponding to the JPEG file to be expanded is not set when UsePutMCUOnly = 1, the AP705100-B03 additional library stops expansion processing and is terminated erroneously. In this sense, the UsePutMCUOnly option takes precedence over UsePutMCU.

Table 3-16 shows examples of library operations for each set value of the OpMCU option.

Table 3-16. Set Values of OpMCU Option and Corresponding Library Operations

OpMCU option set value (example)	When JPEG file supporting putMCU22 function is input	When JPEG file supporting putMCU41 function is input
001000	Calling of putMCU function of additional library	Calling of putMCU function of additional library
011000	Calling of user-defined putMCU22 function	Calling of putMCU function of additional library
101000	Calling of user-defined putMCU22 function	Expansion processing is stopped.
111000	Calling of user-defined putMCU22 function	Expansion processing is stopped.

(4) ratio

If the image zoom in/out option (VideoZoomLinear/VideoZoomNormal) is validated by member Policy of the JPEGEXINFO structure, the zoom-in/out ratio is specified by this member ratio.
Multiply the actual rate by eight and round the result to an integer. Substitute this integer value into ratio. If a negative value or zero is specified, it is assumed that value '1' is specified.

(5) ErrorState

If an error occurs during expansion processing, an error number is written to member ErrorState. For the meaning of the error number, see **Section 3.4.6**.

(6) Work

Set the first address of the JPEGEXWORK structure to member Work. The JPEGEXWORK structure specifies a work area that can be used by the additional library (see **Section 3.4.4**).

(7) Video

Set the first address of the JPEGEXVIDEO structure in member Video. The JPEGEXVIDEO structure performs setting related to drawing (see **Section 3.4.5**).

(8) Inf

The user does not have to be aware of member Inf. The additional library itself sets the first address of the JPEGEXFrmINFO structure in member Inf. The JPEGEXFrmINFO structure is used by the additional library to store the variables needed for expansion processing, and is allocated in the work area.

3.4.4 Setting of JPEGEXWORK Structure Parameters

Before calling the expansion main function, specify a work area that can be used by the additional library, using the JPEGEXWORK structure.

Table 3-17. JPEGEXWORK Structure

Member	Type	Contents	IN/OUT
Work1	unsigned int	Work area first address	IN
Work1Len	unsigned int	Work area size (bytes)	IN
Work1Used	unsigned int	Size of work area used (bytes)	OUT
Work2	unsigned int	Work area first address	IN
Work2Len	unsigned int	Work area size (bytes)	IN
Work2Used	unsigned int	Size of work area used (bytes)	OUT

(1) Specifying work area

Specify the first address of the work area that can be used by the additional library in either Work1 or Work2, and specify the usable size (number of bytes) in Work1Len or Work2Len. The number of bytes actually used is stored in Work1Used or Work2Used after the additional library has been terminated.

If the internal data RAM can be used as a work area, specify the first address and usable size in either Work1xxx or Work2xxx, whichever is available.

(2) Work area of internal RAM

If the internal RAM can be used as a work area, the additional library tries to allocate an MCU buffer and DCT temporary buffer into internal RAM. Table 3-18 shows the size of the MCU buffer and DCT temporary buffer.

Table 3-18. Size of MCU Buffer and DCT Temporary Buffer

Buffer	Size
MCU buffer	768 bytes (with JPEG file of 4:1:1) 384 bytes (with JPEG file of 1:1:1)
DCT temporary buffer	256 bytes

3.4.5 Setting of JPEGEXVIDEO Structure Parameters

Before calling the expansion main function, set the parameters (VRAM configuration and clipping) necessary for image output of additional expansion processing, by using the JPEGEXVIDEO structure.

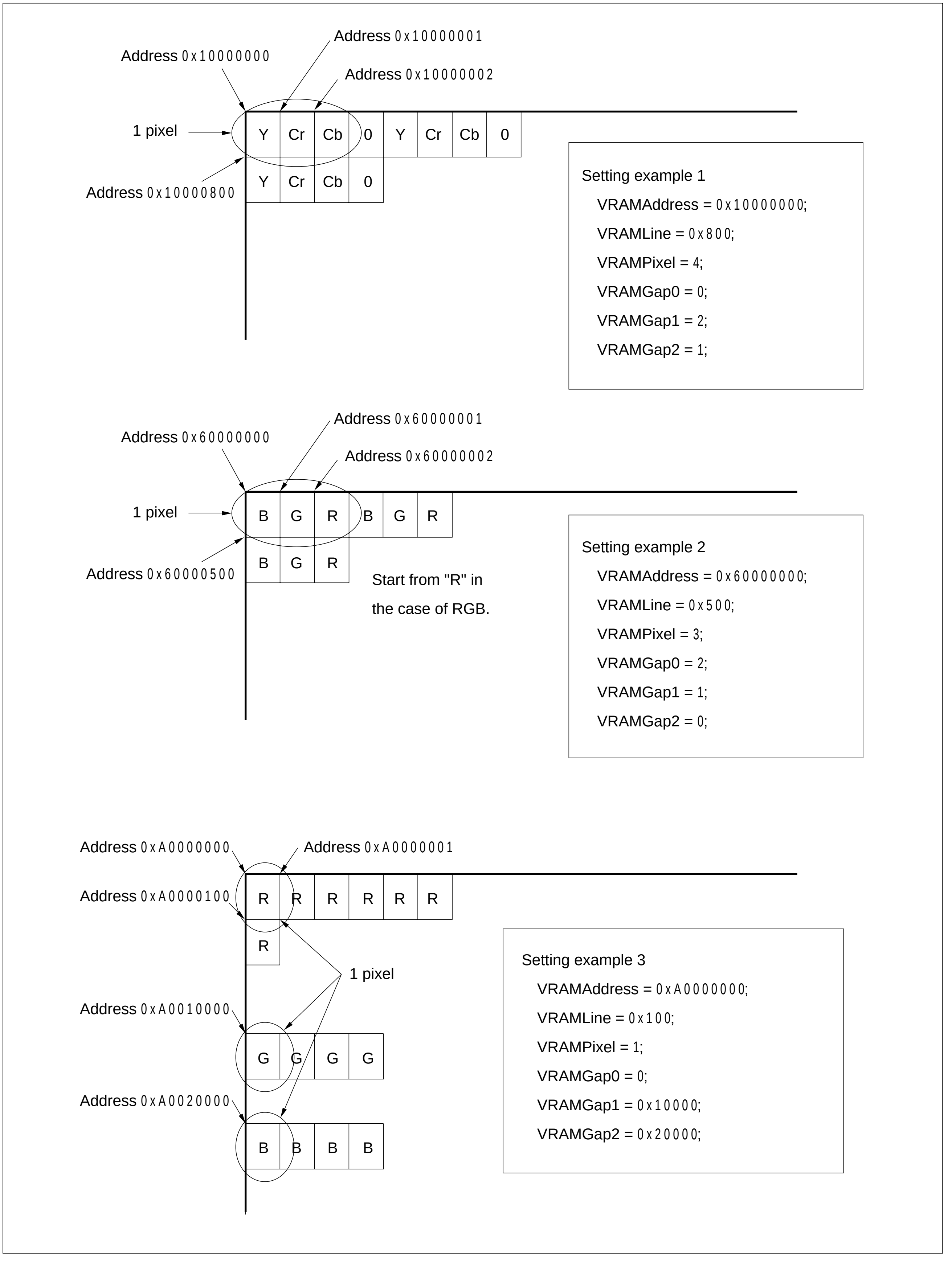
Table 3-19. JPEGEXVIDEO Structure

Member	Type	Contents	IN/OUT
VRAMAddress	unsigned char*	First address of VRAM	IN
VRAMWidth	int	Horizontal width of VRAM (pixels)	IN
VRAMHeight	int	Vertical width of VRAM (pixels)	IN
VRAMPixel	int	Address difference of VRAM by one horizontal pixel	IN
VRAMLine	int	Address difference of VRAM by one vertical pixel	IN
VRAMGap0	int	Byte offset of Y pixel (or R pixel)	IN
VRAMGap1	int	Byte offset of Cb pixel (or G pixel)	IN
VRAMGap2	int	Byte offset of Cr pixel (or B pixel)	IN
ClipStartX	int	Clipping start position (X coordinate) Set dummy value 0 when not performing clipping.	IN
ClipStartY	int	Clipping start position (Y coordinate) Set dummy value 0 when not performing clipping.	IN
ClipWidth	int	Clipping horizontal size (pixels) Set dummy value 0x7FFFFFFF when not performing clipping.	IN
ClipHeight	int	Clipping vertical size (pixels) Set dummy value 0x7FFFFFFF when not performing clipping.	IN

(1) VRAM configuration

Figure 3-14 shows an example of setting the VRAM-related members (VRAMxxx).

Figure 3-14. Example of Setting VRAM-Related Members of Additional Library



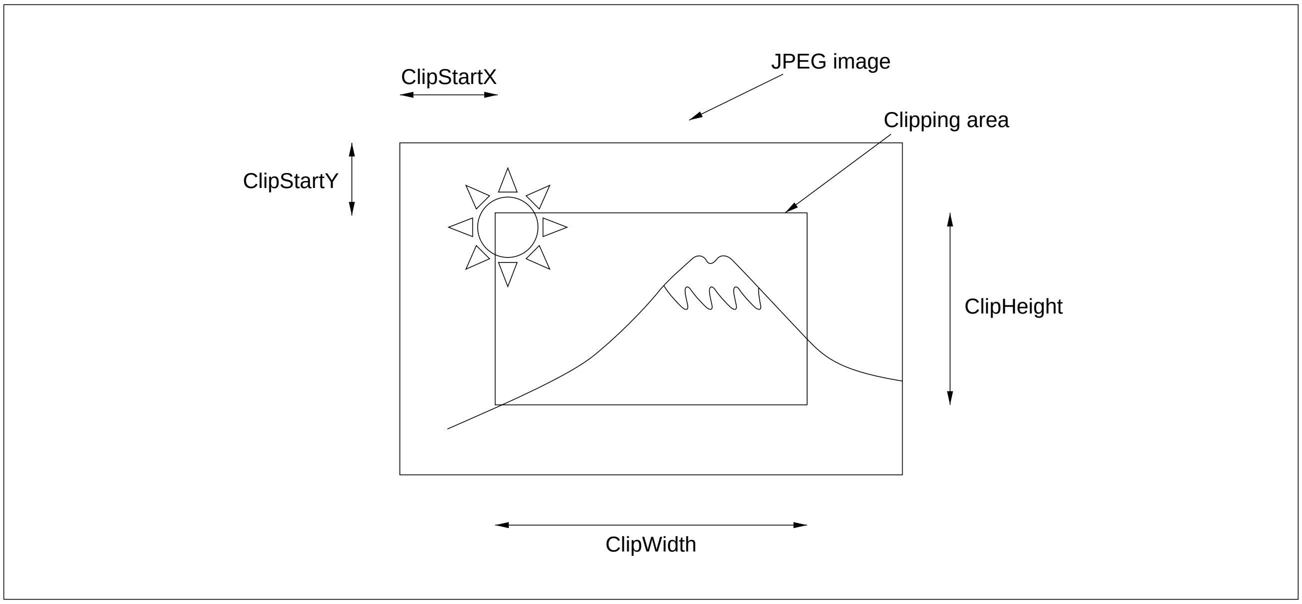
(2) Setting clipping

Clipping during additional expansion is performed when a value other than the dummy value is set in the clipping-related members (Clipxxx). When not performing clipping, substitute the dummy values shown in Table 3-19 into the clipping-related members.

The area of a JPEG image specified by the clipping-related members (Clipxxx) is shown in Figure 3-15.

To change the position at which a clipped image is to be drawn, adjust the values of the VRAMAddress members.

Figure 3-15. Clipping Area

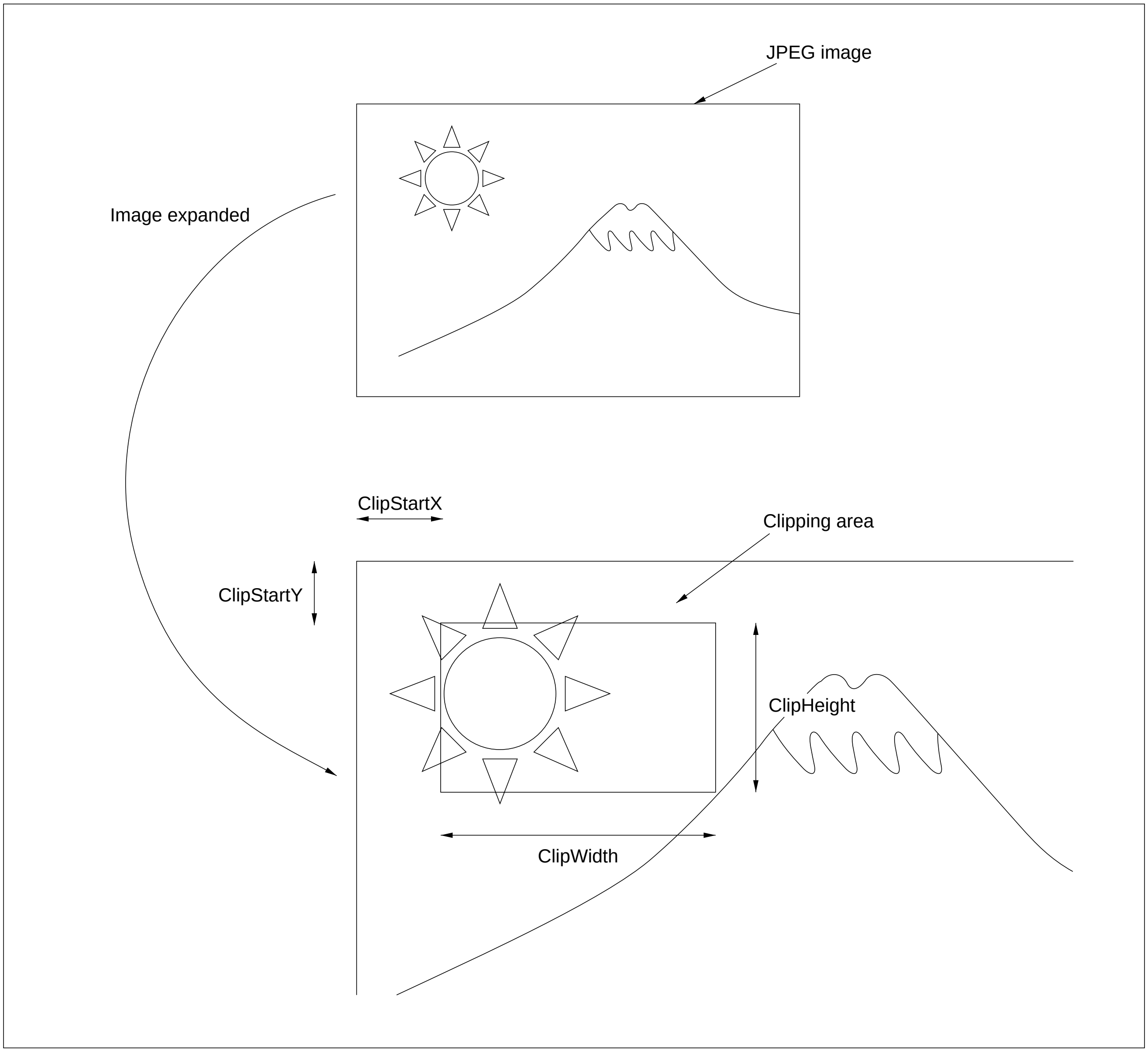


(3) Relationship between setting of clipping and zooming in/out

If zooming in or out is specified by using the option of member Policy of the JPEGEXINFO structure (see **(g)** in **Section 3.4.3 (3)**), the values of ClipStartX, ClipStartY, ClipWidth, and ClipHeight are applied to the zoomed in or out image.

Figure 3-16 shows the clipping area when the image is zoomed in or out.

Figure 3-16. Clipping Area with Image Zoomed In/Out



3.4.6 Errors during Additional Expansion

If the additional library is terminated erroneously, the numbers stored to member ErrorState of the JPEGEXINFO structure and their meanings are listed in Table 3-20.

Table 3-20. Errors of Additional Library (1/2)

Error message	Number	Meaning
ErrorMode	0x1000	An invalid value has been set in member Mode of JPEGEXINFO structure.
ErrorAllocate	0x1001	Processing cannot be performed because the work area has run short.
ErrorMultiFrame	0x1002	Multiframe format is not supported. Processing is terminated.
ErrorMultiScan	0x1003	Expansion cannot be performed because of multiscan.
ErrorMultiDQT	0x1004	Quantization tables having the same number are defined in a JPEG file in duplicate.
ErrorMultiDHT	0x1005	Huffman tables having the same number are defined in a JPEG file in duplicate.
ErrorJPEGBuffLen	0x1006	Size of JPEG buffer is too small.
ErrorJPEGMarker	0x1007	An error has been found during JPEG marker analysis. An unknown marker has been found.
ErrorSOIMarker	0x1008	Marker other than SOI is at the beginning.
ErrorDQTsegment	0x1009	DQT segment contains an error.
ErrorDQTsegmentTq	0x100A	Quantization table number written to DQT segment does not conform to JPEG standard.
ErrorDQTsegmentPq	0x100B	Value of quantization table precision written to DQT segment does not conform to JPEG standard.
ErrorSOFsegment	0x100C	SOF segment contains an error.
ErrorSOFsegmentNf	0x100D	Too many color components are specified for SOF segment.
ErrorSOFsegmentSF	0x100E	Value of sampling factor written to SOF segment does not conform to JPEG standard.
ErrorSOFsegmentTq	0x100F	Quantization table number written to SOF segment does not conform to JPEG standard.
ErrorDHTsegment	0x1010	DHT segment contains an error.
ErrorDHTsegmentTc	0x1011	Table number (Tc) written to DHT segment does not conform to JPEG standard.
ErrorDHTsegmentTh	0x1012	Table number (Th) written to DHT segment does not conform to JPEG standard.
ErrorSOSsegment	0x1014	SOS segment contains an error.
ErrorSOSsegmentCi	0x1015	Color component ID number written to SOS segment is not found in ID written to SOF segment.
ErrorSOSsegmentTq	0x1016	No quantization table is defined for expanding the specified scan.
ErrorSOSsegmentTa	0x1017	No AC component Huffman table is defined for expanding the specified scan.

Table 3-20. Errors of Additional Library (2/2)

Error message	Number	Meaning
ErrorSOSsegmentTd	0x1018	No DC component Huffman table is defined for expanding the specified scan.
ErrorDCcode	0x1019	An error is found in compressed data during DC coefficient decoding.
ErrorACcode	0x101A	An error is found in compressed data during AC coefficient decoding.
ErrorHuffcode	0x101B	A marker is found at an unexpected position. An error is found in compressed data during progressive decoding.
ErrorDNLsegment	0x101C	DNL marker is found at an unexpected position.
ErrorRSTsegment	0x101D	RSTn marker is found at an unexpected position.
ErrorDRIsegment	0x101E	DRI segment contains an error.
ErrorDNLnot1stScan	0x101F	Position of DNL segment is not immediately after the first scan.
ErrorPutMCUfunc	0x1020	UsePutMCUOnly is selected by Policy, but putMCU function corresponding to the sampling ratio of the JPEG file to be expanded is missing.

3.4.7 Warning Messages Output during Additional Expansion

If a warning message is displayed during additional library execution, the JPEGEXWarning function is called, and the warning number is passed as its first argument. The warning numbers and their meanings are listed in Table 3-21.

Table 3-21. Warning Messages for Additional Library

Warning message	Number	Meaning
WarningBitStuff	0x2000	Error is found as a result of stuffing bit check. This error is ignored and processing continues.
WarningProgACInterleave	0x2001	AC coefficient must be non-interleave in progressive format (outside JPEG standard). Processing continues because there is no problem.
WarningBlock20	0x2002	Total number of blocks in one MCU exceeds 20 (outside JPEG extended standard). Processing continues because there is no problem.
WarningBlock10	0x2003	Total number of blocks in one MCU exceeds 10 (outside JPEG standard). Processing continues because there is no problem.
WarningDQTbaselinePq	0x2004	A 16-bit precision quantization table is defined on base line (outside JPEG standard). Processing continues because there is no problem.
WarningDHTbaselineTh	0x2005	Huffman table numbers 2 and 3 are defined on base line (outside JPEG standard). Processing continues because there is no problem.
WarningAPP14	0x2006	Although APP segment (APP14) is found, the value of color space identifier in APP14 segment is other than 0, 1, or 2. Processing continues on the following assumption: Single color: Monochrome Three colors: YCbCr Four colors: YCbCr with last color ignored
WarningMultiDQT	0x2007	Quantization table is defined after one or more scan. Processing continues because there is no problem.
WarningMultiDHT	0x2008	Huffman table is defined after one or more scan. Processing continues because there is no problem.
WarningSOFYDNL	0x2009	Position of DNL segment is not immediately after the first scan but the value defined by this DNL is assumed as the number of vertical pixels of image and processing continues.

3.5 OVERWRITE FUNCTION

Of the functions offered by the additional library, those that can be overwritten by the user are called overwrite functions. Of the overwrite functions, the JPEG file acquisition function must always be overwritten and defined. The other overwrite functions are optional and do not have to be overwritten.

3.5.1 JPEG File Acquisition Function

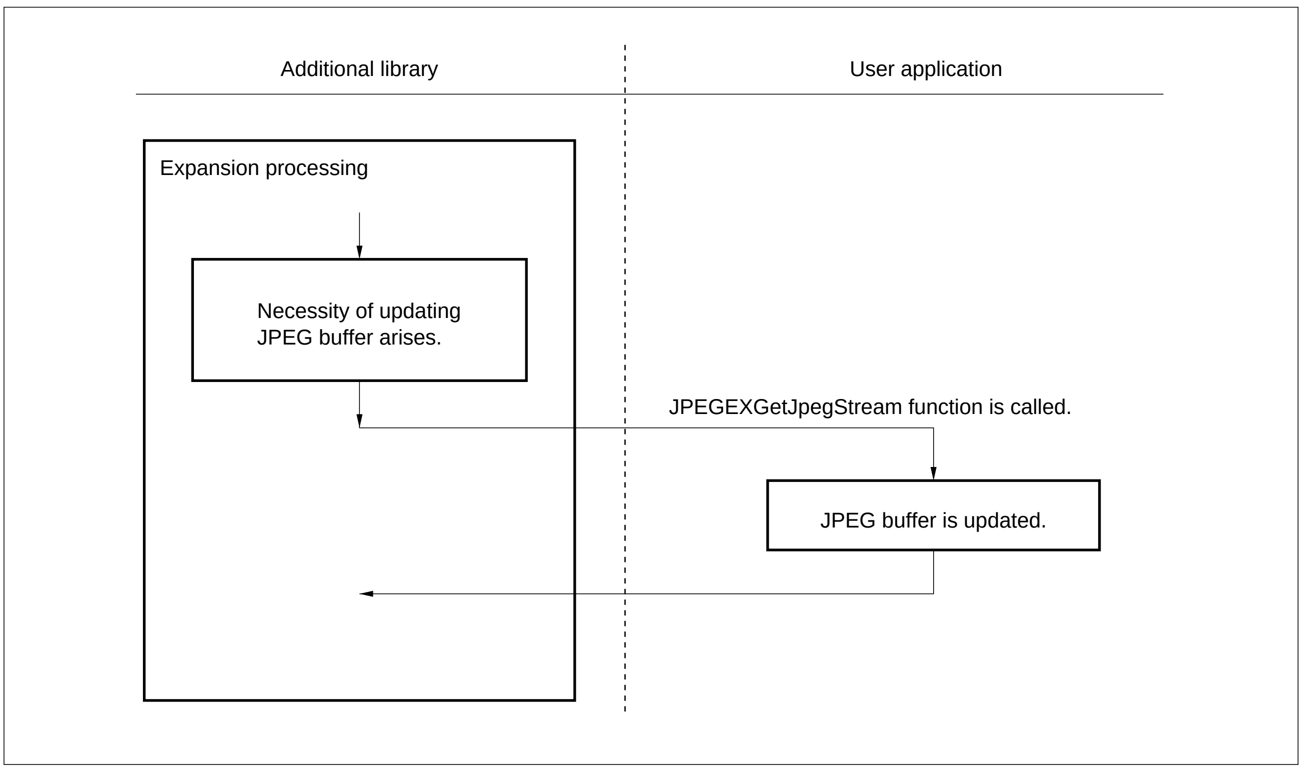
Function name	JPEGEXGetJpegStream
Format	void JPEGEXGetJpegStream (struct JPEGEXBUFF * JPBUFF);
Argument	First address of JPEGEXBUFF structure
Return value	None

To update the contents of the JPEG buffer, the members of the JPEGEXBUFF structure are set and then the JPEG file acquisition function is called. Define this function by the user application.

If the additional library requests updating of the JPEG file, this function must be called from the user application to clear the contents of the JPEG buffer.

If additional expansion processing is executed in one-pass mode, this function is called each time the additional library requests updating of the JPEG file. In two-pass mode, this function is called only once (for details of how to set the number of passes, see **(d)** in **Section 3.4.3 (3)**).

Figure 3-17. Updating of JPEG Buffer



The members of the JPEGEXBUFF structure are set as follows (the JPEGEXBUFF structure is defined in header file jpegex.h file).

Table 3-22. JPEGEXBUFF Structure

Member	Type	Contents	IN/OUT
TaskID	int	Task ID number	OUT (can be overwritten)
JPEGBUFF	unsigned char*	First address of JPEG buffer	IN
JPEGBUFFLEN	unsigned int	Size of JPEG buffer (bytes)	IN

(1) TaskID

TaskID is initialized with the value of TaskID, the member of the JPEGEXINFO structure, (see **Section 3.4.3 (1)**) when the JPEGEXBUFF structure is allocated in the work area.

The value of this TaskID is used as the ID of the task when the operation is performed in a multitask environment. When a single task is used, this TaskID may be ignored. In addition, the value of this member TaskID may be overwritten in the JPEGEXGetJpegStream function. The additional library does not reference the value of this member after initialization.

(2) JPEGBUFF

Set the first address of the JPEG buffer in JPEGBUFF.

(3) JPEGBUFFLEN

Set the size of the JPEG buffer (number of bytes) in JPEGBUFFLEN.

Allocate as great a value as possible as the size of the JPEG buffer, such that, if possible, it can accommodate all the expanded JPEG files. If the size of the JPEG buffer is 32 bytes or less, the additional library cannot run.

3.5.2 APP Marker Function

Function name	JPEGEXdecodeAPP	
Format	void JPEGEXdecodeAPP (int APPnumber, int JpegStreamOffset, int Length);	
Arguments	First argument:	Number of APPn marker
		0: APP0 (0xFF 0xE0)
		1: APP1 (0xFF 0xE1)
		2: APP2 (0xFF 0xE2)
		:
		15: APP15 (0xFF 0xEF)
		16: COM (0xFF 0xFE)
	Second argument:	Number of offset bytes of APPn segment (from the beginning of JPEG file)
	Third argument:	Length of APPn segment (number of bytes)
Return value	None	

The additional library calls this function when an APPn segment is found. This function is optional. If any processing is necessary, the user can overwrite this function. If the user does not overwrite this function, the default JPEGEXdecodeAPP function is called. The default function performs no processing.

Figure 3-18. Processing When APP Marker Is Found

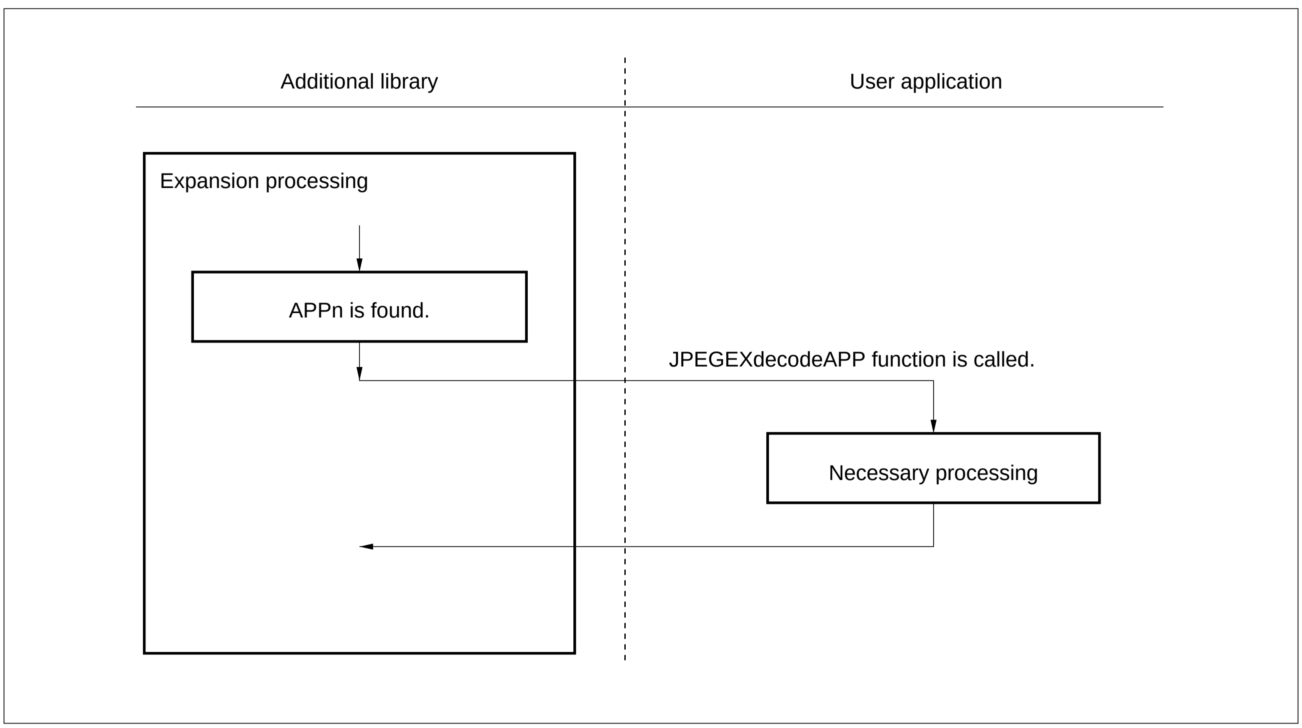
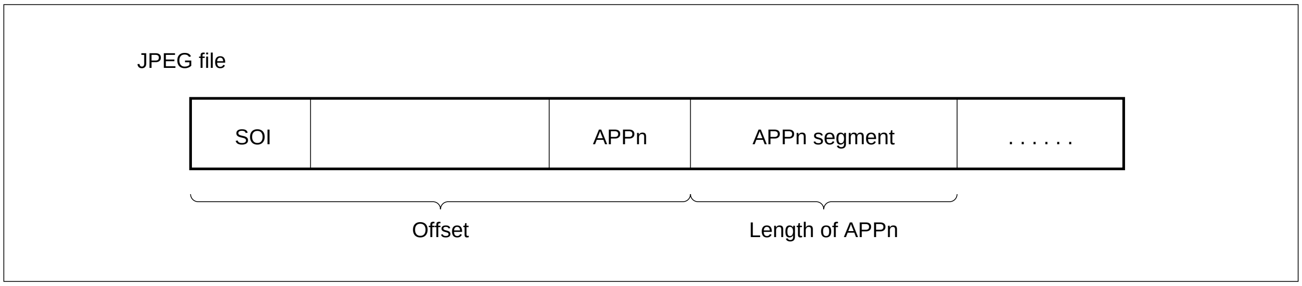


Figure 3-19. Offset and Length of APPn Segment



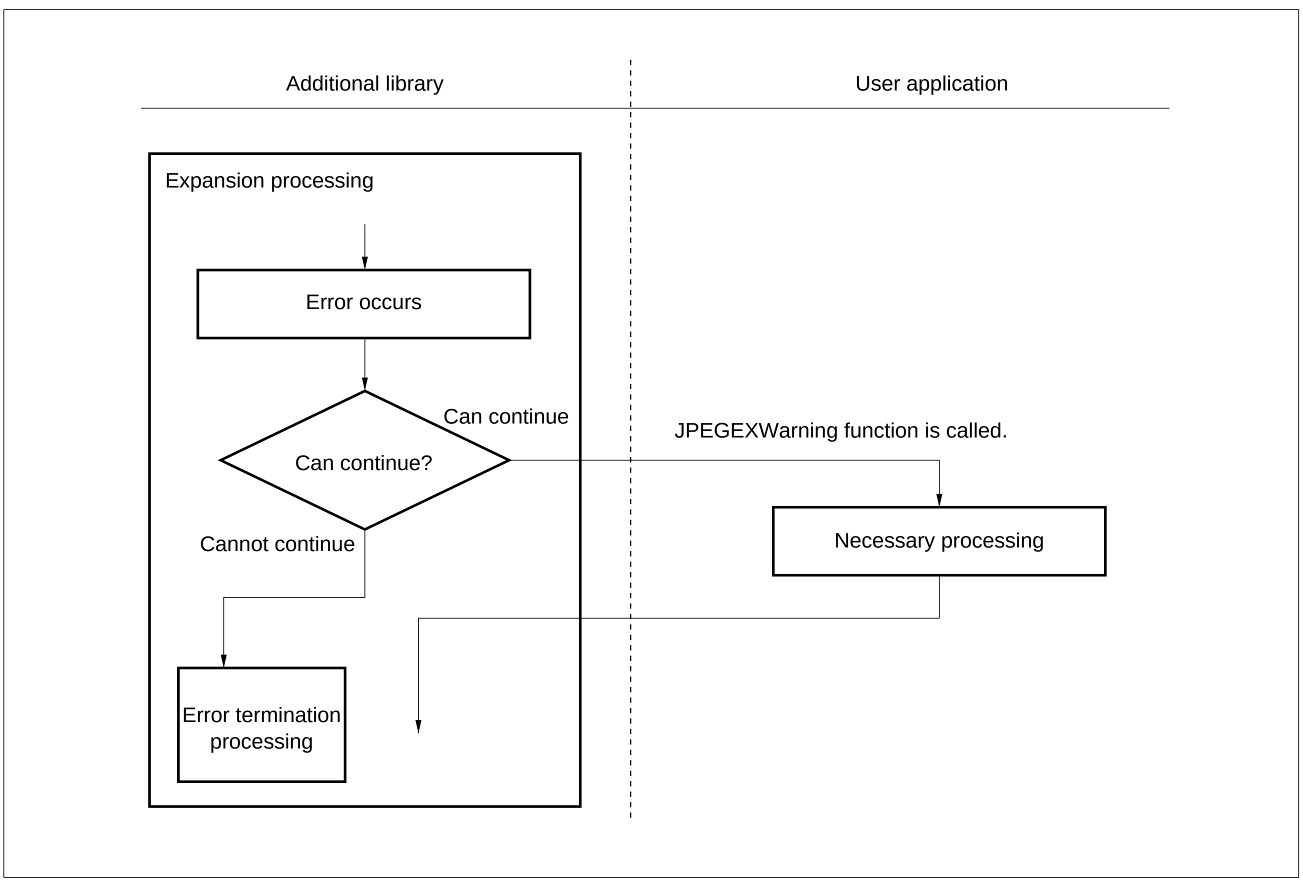
3.5.3 Warning Message Function

Function name	JPEGEXWarning
Format	void JPEGEXWarning (int WarningNumber);
Argument	Warning message number
Return value	None

If an error that is not so serious as to terminate the entire processing occurs during additional expansion processing, the additional library calls this function. This function is optional. If there is any necessary processing, the user can overwrite this function. If the user does not overwrite this function, the default JPEGEXWarning function is called. The default function performs no processing.

For a description of the warning message number that is set as an argument, see **Section 3.4.7**.

Figure 3-20. Processing If Error That Does not Disrupt Processing Occurs



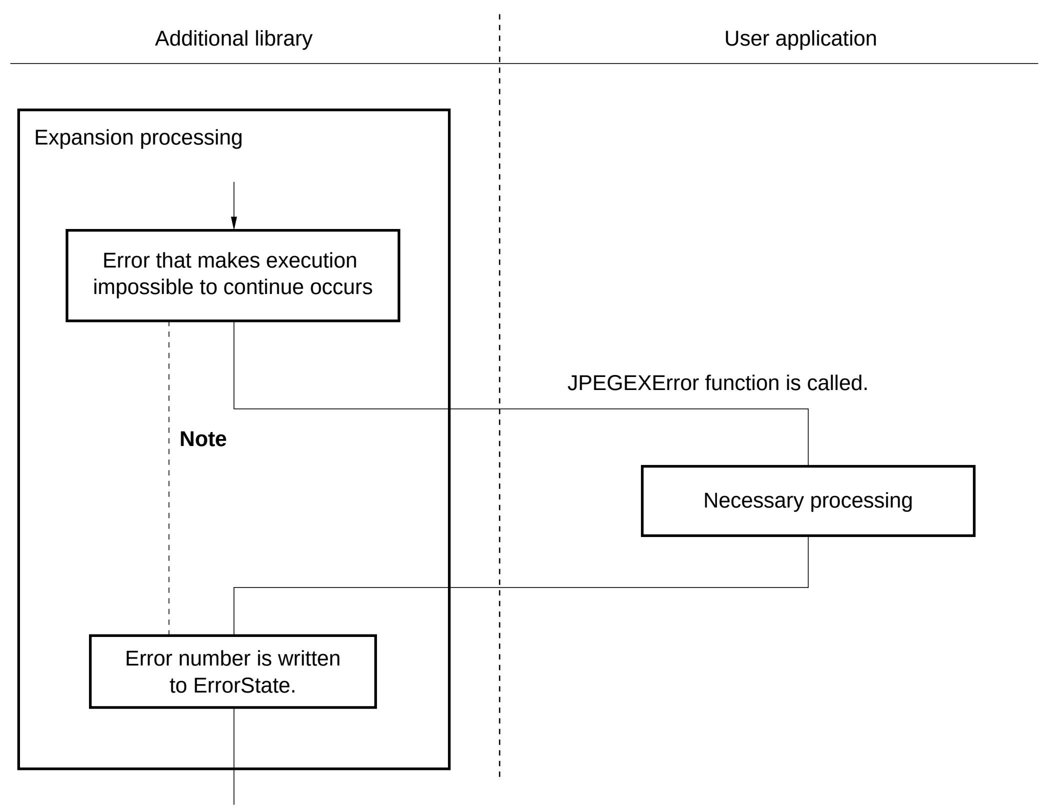
3.5.4 Error Message Function of Debug Library

Function name	JPEGEXError
Format	void JPEGEXError (char* msg);
Argument	Pointer to error message character string
Return value	None

This function is optional, and is called only when the debug library (see **Section 3.2**) is used. If an error that causes the processing to stop occurs during additional expansion processing, the additional library calls this function immediately before terminating the expansion processing. Normally, only an error number is written to member ErrorState of the JPEGEXINFO structure if an error has occurred. If this function is called, however, the nature of the error is reported by an ASCII code corresponding to the error number so that the error can be easily identified. If there is any necessary processing, the user can overwrite this function.

For the error message character string that is set as an argument, see **Section 3.4.6**.

Figure 3-21. Processing by Debug Library in Case of Error



Note This route is used when the debug library is not used. The JPEGEXError function is not called.

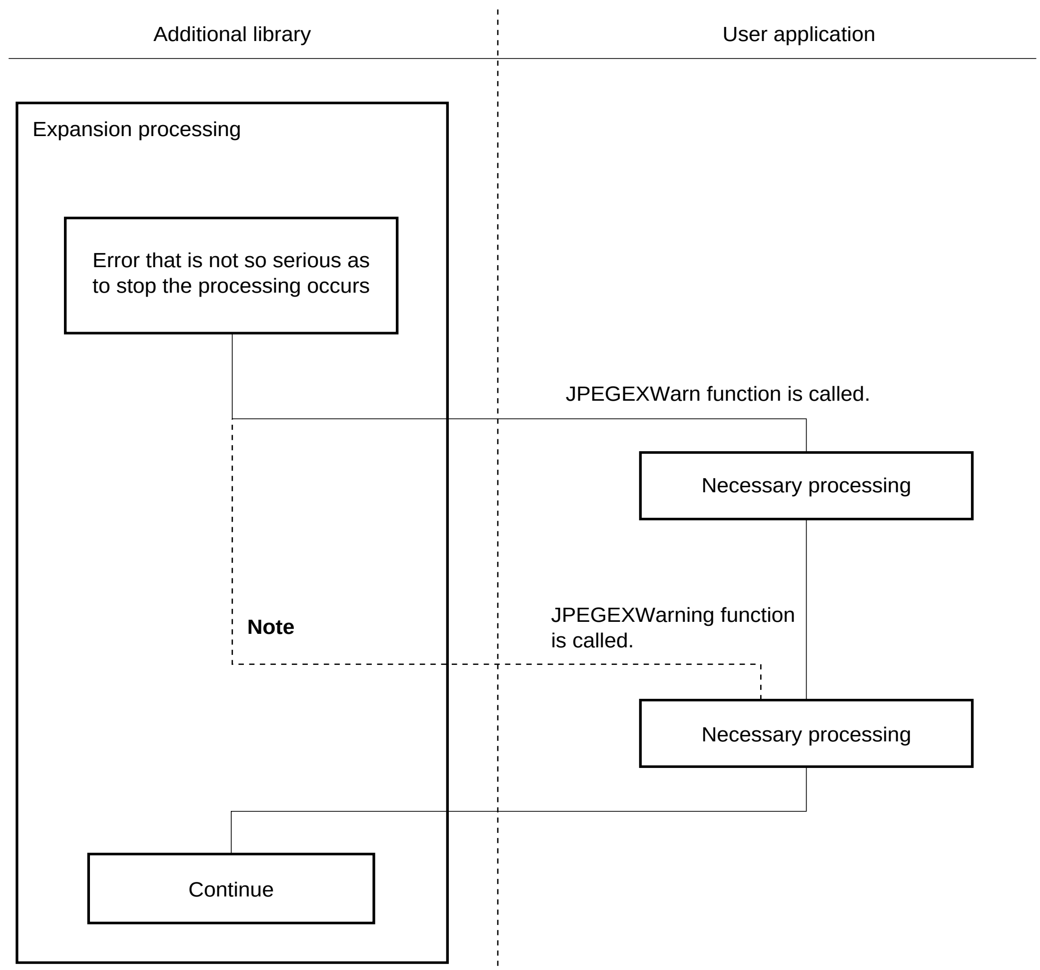
3.5.5 Warning Message Function of Debug Library

Function name	JPEGEXWarn
Format	void JPEGEXWarn (char* msg);
Argument	Pointer to warning message character string
Return value	None

This function is optional, and is called only when the debug library (see **Section 3.2**) is used. If an error (warning) that is not so serious as to stop the processing occurs during additional expansion processing, the additional library calls this function immediately before calling the JPEGEXWarning function. Normally, only a warning message number is used as the argument of the JPEGEXWarning function if a warning occurs. If this function is called, however, the nature of the warning is reported by an ASCII code corresponding to the warning number so that the warning can be easily identified. If there is any necessary processing, the user can overwrite this function.

For the warning message, see **Section 3.4.6**.

Figure 3-22. Processing by Debug Library in Case of Warning



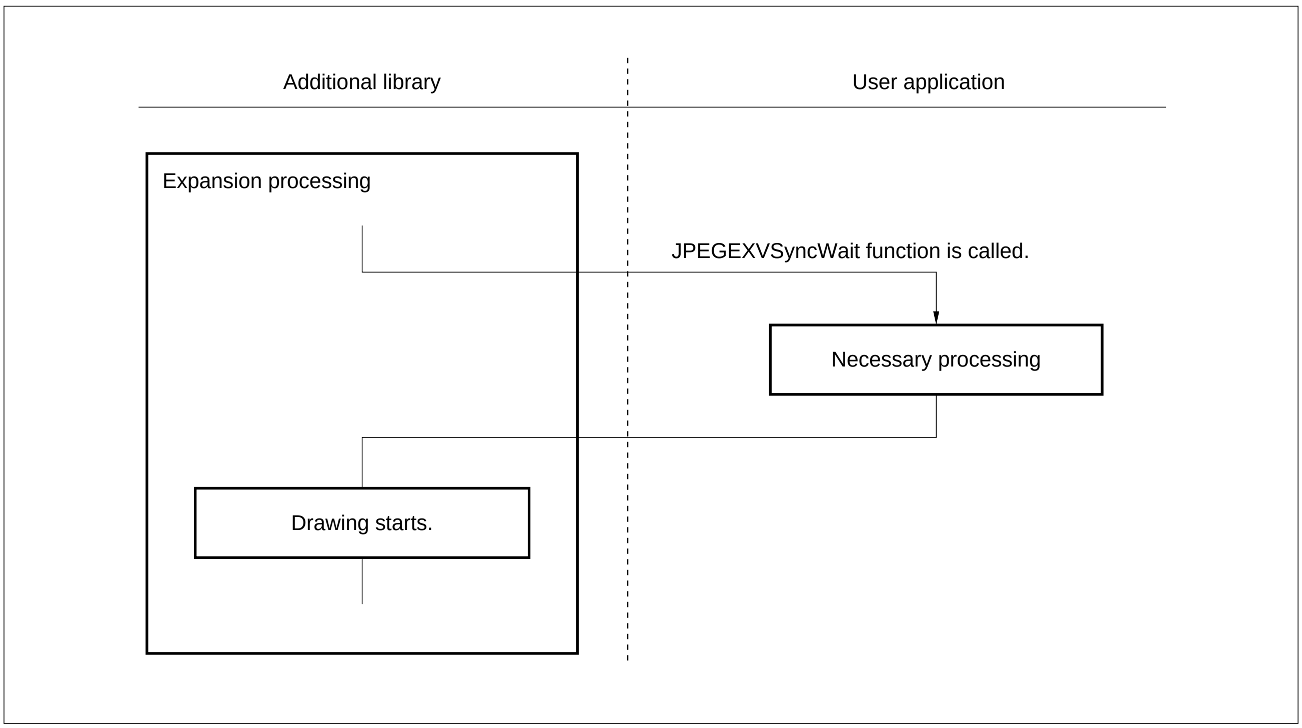
Note This route is used when the debug library is not used. The JPEGEXWarn function is not called.

3.5.6 Display Timing Adjustment Function

Function name	JPEGEXVSyncWait
Format	void JPEGEXVSyncWait ();
Argument	None
Return value	None

This function is called before the additional library starts drawing. This function is optional. The user can overwrite this function if there is a need to adjust the display timing. If the user does not overwrite this function, the default JPEGEXVSyncWait function is called. The default function does not perform any processing.

Figure 3-23. Processing before Start of Drawing



3.5.7 MCU Data Output Function

Function name	JPEGEXputMCU	
Format	void JPEGEXputMCU (short* mcubuff, int Y, int X, struct JPEGEXMCUSTR* MCUstr);	
Arguments	First argument:	Pointer to MCU buffer
	Second argument:	Upper-left corner coordinate (Y) of MCU to be drawn
	Third argument:	Upper-left corner coordinate (X) of MCU to be drawn
	Fourth argument:	First address of JPEGEXMCUSTR structure
Return value	None	

This function draws an image in VRAM. This part of the expansion processing that is most heavily affected by hardware can be overwritten and created by the user. This function is optional.

To create a JPEGEXputMCU function, the UseExPutMCU bit of member Policy of the JPEGEXINFO structure must be set (see **(i)** in **Section 3.4.3 (3)**). If this bit is not set, the additional library dynamically changes between the JPEGEXputMCU function and its alternate function.

To overwrite the JPEGEXputMCU function, see **Section 3.6**.

3.5.8 Pixel Data Output Function

Function name JPEGEXpset

Format The arguments of this function differ depending on the setting of the image output mode.

YCbCr output:

void JPEGEXpset (struct JPEGEXMCUSTR* MCUstr, int y, int x, int Cy, int Cu, int Cv);

PGB output:

void JPEGEXpset (struct JPEGEXMCUSTR* MCUstr, int y, int x, int R, int G, int B);

Arguments

First argument:	First address of JPEGEXMCUSTR structure
Second argument:	Y coordinate of pixel to be drawn
Third argument:	X coordinate of pixel to be drawn
Fourth argument:	Y color element data for YCbCr output R color element data for RGB output
Fifth argument:	Cb color element data for YCbCr output G color element data for RGB output
Sixth argument:	Cr color element data for YCbCr output B color element data for RGB output

Return value None

This function draws an image in VRAM. The part of the expansion processing that is most heavily affected by hardware can be overwritten and created by the user. This function is optional.

To create a JPEGEXpset function, the UsePset bit of the member Policy of the JPEGEXINFO structure must be set (see **(f)** in **Section 3.4.3 (3)**). If this bit is 0, the additional library may dynamically change between the JPEGEXpset function and its alternate function (pset function expanded in line).

If the UsePutMCU bit of Policy (see **(j)** in **Section 3.4.3 (3)**) is set to 1, the putMCU function of the basic library, instead of the JPEGEXputMCU function, may be called. At this time, the JPEGEXpset function is not called.

The values of variables Cy, Cu, and Cv, and R, G, and B are 0x00 to 0xFF.

An example of a C source of the pset function is shown below.

```
void JPEGEXpset( struct JPEGEXMCUSTR * MCUstr, int y, int x, int Cy, int Cu, int Cv )
{
        unsigned char * vram;

        vram  = MCUstr->VRAMAddress + y * MCUstr->VRAMLine + x * MCUstr->VRAMPixel;
        *(vram + MCUstr->VRAMGap0) = (unsigned char)Cy;
        *(vram + MCUstr->VRAMGap1) = (unsigned char)Cu;
        *(vram + MCUstr->VRAMGap2) = (unsigned char)Cv;
}
```

3.6 CUSTOMIZING ADDITIONAL LIBRARY

The image output portion of the additional library can be customized by overwriting related functions.

The image output portion must always be customized depending on the specification of the image memory to be used or on the setting of the image output options. The following cases can be cited as examples:

- If the color space of the image memory is not 24-bit RGB (24-bit YCbCr)
- If the image memory cannot be accessed with st.b/st.h/st.w

To customize the image output portion, change the JPEGEXpset or JPEGEXputMCU function. The JPEGEXpset function draws dots on the screen and is called the internal putMCU function of the additional library. The JPEGEXputMCU function outputs data to the screen in MCU units and is dedicated to customization (if this function is not customized, the internal putMCU function of the additional library is called).

3.6.1 Simple Customization

The image output portion can easily be customized by overwriting the JPEGEXpset function. For the specifications of the JPEGEXpset function, see **Section 3.5.8**.

When overwriting the JPEGEXpset function, member Policy of the JPEGEXINFO structure must be set so that the JPEGEXpset function is used (see **Section 3.6.3**).

3.6.2 Sophisticated Customization

To improve the overall performance of the system, the image output portion must be customized by overwriting the JPEGEXputMCU function, in addition to the JPEGEXpset function. By overwriting the JPEGEXputMCU function, the redundant portion such as address calculation and storing/loading arguments to/from the stack can be reduced.

For the specifications of the JPEGEXputMCU function, see **Section 3.5.7**.

3.6.3 Option Setting for Customization

Member Policy of the JPEGEXINFOR structure has options related to image output. To customize the JPEGEXpset and JPEGEXputMCU functions, set these options to the values shown in Table 3-23.

Table 3-23. Set Values of Policy Options for Customization

Priority	Option bit name	Set value when JPEGEXpset function is used	Set value when JPEGEXpset function/JPEGEXputMCU function is used
High	UsePutMCUOnly	0	0
↑	UsePutMCU	0	0
	VideoZoomNormal	Don't care	Don't care
	VideoZoomLinear	Don't care	Don't care
↓	UseExPutMCU	0	1
Low	UsePset	1	Don't care

These options have priorities. If the value of an option with the higher priority is set to 1, the image output function specified by that option is used, and the image output function specified by an option with the lower priority may not be used.

(1) UsePutMCUOnly

This option allows the use of the putMCU function created by the user with the basic library. If the value of this option is set to 1, the value of the UseExPutMCU or UsePset option is ignored, and the JPEGEXputMCU and JPEGEXpset functions are not called.

(2) UsePutMCU

This option allows the use of the putMCU function created by the user with the basic library. If all the following three conditions are satisfied, the UseExPutMCU option is ignored and the JPEGEXputMCU function is not called.

- UsePutMCU = 1
- If any of the values of the UsePutMCU22, UsePutMCU41, UsePutMCU21, and UsePutMCU11 options of Policy (see **(j)** in **Section 3.4.3 (3)**) is 1
- When the JPEG file of the sampling ratio corresponding to the UsePutMCUxx option set by Policy is to be expanded (see **Table 2-48**)

(3) VideoZoomNormal/VideoZoomLinear

If these options are set to 1, the additional library always calls the JPEGEXpset function (therefore, the JPEGEXpset function must be overwritten). In this case, the value of the UseExPutMCU option is ignored, and the JPEGEXputMCU function is not called.

3.6.4 Example of Customization

The next page shows a C source example that defines the JPEGEXputMCU and JPEGEXpset functions. In this example, the JPEGEXpset function is called repeatedly. To improve the processing speed of the additional library, it is necessary to expand the function that is often called in line as shown in this example.

```
#include "jpegex.h"

void JPEGEXpset(struct JPEGEXMCUSTR *MCUstr,int y,int x,int Cy,int Cu,int Cv)
{
        unsigned char *vram;

        vram=MCUstr->VRAMAddress+y * MCUstr->VRAMLine+x * MCUstr->VRAMPixel;
        *(vram+MCUstr->VRAMGap0)=(unsigned char)Cy;
        *(vram+MCUstr->VRAMGap1)=(unsigned char)Cu;
        *(vram+MCUstr->VRAMGap2)=(unsigned char)Cv;
}

int JPEGEXpget(short *mcubuff,int Y,int X,int vf)
{
        return (int)*(mcubuff+(((X>>3)+(Y>>3) * vf)<<6)+(Y&7)*8+(X&7));
}

void JPEGEXputMCU(short *mcubuff,int Y,int X,struct JPEGEXMCUSTR *MCUstr)
{
        int x,y,Xs,Ys,w,h;
        int Cy,Cu,Cv;
        int hf,vf,hf0,hf1,hf2,vf0,vf1,vf2;
        int x0,x1,x2,y0,y1,y2;
        short *mcubuff1;
        short *mcubuff2;

        Xs=X-MCUstr->ClipStartX;
        Ys=Y-MCUstr->ClipStartY;
        hf0=MCUstr->hf[0];vf0=MCUstr->vf[0];
        hf=MCUstr->hfMax;vf=MCUstr->vfMax;
        w=hf*8;h=vf*8;
        if(MCUstr->component==3){
                hf1=MCUstr->hf[1];vf1=MCUstr->vf[1];
                hf2=MCUstr->hf[2];vf2=MCUstr->vf[2];
                mcubuff1=mcubuff +((vf0 * hf0)<<6);
                mcubuff2=mcubuff1+((vf1 * hf1)<<6);
                for(y=0;y<h;y++){
                        y0=y * vf0/vf;
                        y1=y * vf1/vf;
                        y2=y * vf2/vf;
                        for(x=0;x<w;x++){
                                x0=x * hf0/hf;
                                x1=x * hf1/hf;
                                x2=x * hf2/hf;
                                Cy=JPEGEXpget(mcubuff,y0,x0,hf0);
                                Cu=JPEGEXpget(mcubuff1,y1,x1,hf1);
                                Cv=JPEGEXpget(mcubuff2,y2,x2,hf2);
                                JPEGEXpset(MCUstr,Ys+y,Xs+x,Cy,Cu,Cv);
                        }
                }
        }else if(MCUstr->component==1){
                for(y=0;y<h;y++){
                        y0=y * vf0/vf;
                        for(x=0;x<w;x++){
                                x0=x * hf0/hf;
                                Cy=JPEGEXpget(mcubuff,y0,x0,hf0);
                                JPEGEXpset(MCUstr,Ys+y,Xs+x,Cy,0x80,0x80);
                        }
                }
        }
}
```

CHAPTER 4 INSTALLATION

4.1 INSTALLATION PROCEDURE

(1) UNIX version

```
tar xvof  /dev/fd0
```

Copy the contents from the release medium to the hard disk by executing the above command (specify an appropriate device according to the environment).

(2) Windows version

Copy the contents from the release medium to the hard disk by using the copy command or filer (file manipulation application).
The contents are self-extracting.

Caution The configuration of the directory on the hard disk, to which the contents are copied, is arbitrary.

4.2 SAMPLE CREATING PROCEDURE

Create file "archive" that specifies the JPEG library. Execute jparc830 on the command line. For details, see **Section 2.2.1**.

The path name of the library can be specified as both a relative and absolute path name (in the example below, a relative path name is used).

```
../../lib732     (UNIX-AP70732-B03)
../../lib830     (UNIX-AP705100-B03)
..\..\lib732     (Windows-AP70732-B03)
..\..\lib830     (Windows-AP705100-B03)
```

Specify compiling as follows:

```
make    -f    makergb    (where VRAM is of RGB type)
make    -f    makeycc    (where VRAM is of YCbCr type)
```

For the RGB type, validate "#define RGB" in main.c.

If the NEC CA732/CA830 compiler package (Windows version) is used, execute compiling with VSH (V-shell) as vmake, instead of make.

When the GHS compiler is used, the address information remains in file "jpeg.map" after compiling.

If address information is required when the NEC compiler is used, execute the following dump command:

```
dump732   -h  jpeg.elf   >   jpeg.map   (AP70732-B03)
dump830   -h  jpeg.elf   >   jpeg.map   (AP705100-B03)
```

4.3 SAMPLE OPERATING PROCEDURE

Either the actual machine or a simulator is necessary.

Set the PC (program counter) for __start (global symbol, described in start.s) after downloading. Set a breakpoint for __exit.

Set two or more breakpoints as necessary.

The completed jpeg file is stored into the buffer that stores the jpeg file after the compression program has been executed. In the sample, the return value of function get_Fsize() indicates the size of the jpeg file (in bytes).

After the expansion program has been executed, the expanded image data is written into virtual VRAM. In the sample, a function that creates a BMP file from the virtual VRAM is called. When this function has been executed, the BMP file is written into the buffer that stores the BMP file. The return value of this function indicates the size of the BMP file (in bytes).

APPENDIX A SAMPLE PROGRAM SOURCE LIST (AP70732-B03)

```
/*****************************************************************
 *  Copyright (C) NEC Corporation 1995, 1996          *
 *  All rights reserved by NEC Corporation.            *
 *  Use of copyright notice does not evidence publication   *
 *****************************************************************/

/***** This file is sample usage program
         for V810 JPEG middle-ware library. *****/

#include   "jpeg.h"  /* JPEG library header file */

/*#define  RGB*/
/*** Validate this define to access VRAM in units of 8 or 16 lines ***/
/*#define  TINY_VRAM*/

extern int jpeg_Decompress();    /* Decompress main routine */
extern int jpeg_Compress();      /* Compress main routine */

extern char LuminanceQtbl[64];                  /* default Quality table */
extern char ChrominanceQtbl[64];                /* default Quality table */
extern char DHT_markerLuminanceDC[33];          /* default Huffman table */
extern char DHT_markerChrominanceDC[33];        /* default Huffman table */
extern char DHT_markerLuminanceAC[183];         /* default Huffman table */
extern char DHT_markerChrominanceAC[183];       /* default Huffman table */

/* define of numeric value related to VRAM */
#define   VRAM_ADDR  0x10000000 /* VRAM address */
#define   VRAM_WIDTH   640
#define   VRAM_HEIGHT             480
#define   VRAM_PIXEL    4
#define   VRAM_GAP1     1
#define   VRAM_GAP2     2
#define   VRAM_LINE      (VRAM_WIDTH * VRAM_PIXEL)

/* Number of vertical and horizontal pixels of image */
#define   IMAGE_WIDTH             224
#define   IMAGE_HEIGHT            144

/* Declaration of structure */
CJINFO        CJinfo;               /* structure for jpeg compress library */
DJINFO        DJinfo;               /* structure for jpeg decompress library */
APPINFO       cAppinfo, dAppinfo;   /* structure for APPn segment */
/* Declaration of external RAM work area
(In this example, the compression library and expansion library are not executed at the same time.) */
unsigned char    WorkArea[0x1000];  /* library work area */
#define     JPEGBUFFSIZE  0x10000
/*#define    JPEGBUFSIZE    1*/
unsigned char       jpegbuffer[JPEGBUFFSIZE];
/* Example of character string to be embedded in APPn segment */
unsigned char       str1[] = "JPEG middle-ware library.";
unsigned char       str2[] = "V810(uPD70732) 32-bit RISC Microcomputer";
```

Phase-out/Discontinued

```
/* Comment marker */
unsigned char jpeg_COMStr[] = "This is a Comment Marker.";

/***** Compression sample program *****/
void
move_jpeg()
{
        /* Save contents of jpegbuffer */
}

#ifdef    TINY_VRAM
void
replace_vram()
{
        /* Update VRAM */
}

int
compress()
{
        int       ret;

        CJinfo.ErrorState = 0;               /* initialize */
                /* jpeg buffer start address */
        CJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        CJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* APPINFO structure */
        CJinfo.APP_Info_Bptr = &cAppinfo;
        (CJinfo.APP_Info_Bptr)->APP00_Buff_Bptr = str1;
        (CJinfo.APP_Info_Bptr)->APP00_BuffSize = (short)(sizeof(str1) - 1);
        (CJinfo.APP_Info_Bptr)->APP01_Buff_Bptr = str2;
        (CJinfo.APP_Info_Bptr)->APP01_BuffSize = (short)(sizeof(str2) - 1);
                /* work area for this library */
        CJinfo.Work = (unsigned char *)WorkArea;
                /* compress parameter */
        CJinfo.Restart = 0;                  /* Don't use restart marker */
        CJinfo.Width = IMAGE_WIDTH;
        CJinfo.Height = IMAGE_HEIGHT;
        CJinfo.Quality = 75;
        CJinfo.Sampling = SAMPLE22;   /* 4:1:1 */
        CJinfo.Mode = 1;                     /* normal compress mode */
        CJinfo.StartX = 0;
        CJinfo.StartY = 0;
                /* VRAM information */
        CJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        CJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        CJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        CJinfo.VRAM_Line_Byte = VRAM_LINE;
        CJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
        CJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        CJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
                /* Quality table */

        CJinfo.DQT_Y_Bptr = (char *)LuminanceQtbl;
```

```
        CJinfo.DQT_C_Bptr = (char *)ChrominanceQtbl;
                /* Huffman table */
        CJinfo.DHT_DC_Y_Bptr = (char *)DHT_markerLuminanceDC;
        CJinfo.DHT_DC_C_Bptr = (char *)DHT_markerChrominanceDC;
        CJinfo.DHT_AC_Y_Bptr = (char *)DHT_markerLuminanceAC;
        CJinfo.DHT_AC_C_Bptr = (char *)DHT_markerChrominanceAC;

        replace_vram();     /* get first VRAM data */
        while( 1 ){
                ret = jpeg_Compress();
                if( ret == JPEG_OK ){
                        move_jpeg();
                        return 1;  /* complite */
                }
                if( ret == JPEG_ERR ) return 0;  /* error */
                if( ret & JPEG_CONT1 ){          /* jpegbuffer full */
                        move_jpeg();
                } else if( ret & JPEG_CONT2 ){  /* change VRAM */
                        replace_vram();
                } else {
                        return 0;   /* error ? */
                }
        }
}


#else    /* TINY_VRAM */
void
compress_parameter_ini()
{
                /* work area for this library */
        CJinfo.Work = (unsigned char *)WorkArea;
                /* APPINFO structure */
        CJinfo.APP_Info_Bptr = &cAppinfo;
        (CJinfo.APP_Info_Bptr)->APP00_Buff_Bptr = str1;
        (CJinfo.APP_Info_Bptr)->APP00_BuffSize = (short)(sizeof(str1) - 1);
        (CJinfo.APP_Info_Bptr)->APP01_Buff_Bptr = str2;
        (CJinfo.APP_Info_Bptr)->APP01_BuffSize = (short)(sizeof(str2) - 1);
                /* compress parameter */
        CJinfo.Restart = 0;                /* Don't use restart marker */
        CJinfo.Sampling = SAMPLE22;   /* 4:1:1 */
                /* VRAM information */
        CJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        CJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        CJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        CJinfo.VRAM_Line_Byte = VRAM_LINE;
        CJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
        CJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        CJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
                /* Quality table */
        CJinfo.DQT_Y_Bptr = (char *)LuminanceQtbl;
        CJinfo.DQT_C_Bptr = (char *)ChrominanceQtbl;
                /* Huffman table */
        CJinfo.DHT_DC_Y_Bptr = (char *)DHT_markerLuminanceDC;
        CJinfo.DHT_DC_C_Bptr = (char *)DHT_markerChrominanceDC;
        CJinfo.DHT_AC_Y_Bptr = (char *)DHT_markerLuminanceAC;
```

Phase-out/Discontinued

```
        CJinfo.DHT_AC_C_Bptr = (char *)DHT_markerChrominanceAC;
}

int
compress_test()
{

/*          Targeted value of number of bits of 12 MCUs*/
#define    AVR_BITS   80*12

        int       Quality;
        int       i;
        int       HEAP[3];
        short     xy[12][2]; /* 12 test point, ( x, y ) */
        short     width_tmp, height_tmp;

/*      Test 12 MCUs as follows:*/
/*      VRAM image            */
/*      +————————————————+  */
/*      |0              1|  */
/*      |       8        |  */
/*      |      4 5       |  */
/*      |   9  6 7   10  |  */
/*      |      1 1       |  */
/*      |2              3|  */
/*      +————————————————+  */
        width_tmp = (IMAGE_WIDTH >> 4);
        height_tmp = (IMAGE_HEIGHT >> 4);
        xy[0][0] = 0;                           xy[0][1] = 0;
        xy[1][0] = width_tmp - 1;               xy[1][1] = 0;
        xy[2][0] = 0;                           xy[2][1] = height_tmp - 1;
        xy[3][0] = width_tmp - 1;               xy[3][1] = height_tmp - 1;
        width_tmp >>= 1;     /* half of width */
        height_tmp >>= 1;    /* half of height */
        xy[4][0] = width_tmp - 1;               xy[4][1] = height_tmp - 1;
        xy[5][0] = width_tmp;                   xy[5][1] = height_tmp - 1;
        xy[6][0] = width_tmp - 1;               xy[6][1] = height_tmp;
        xy[7][0] = width_tmp;                   xy[7][1] = height_tmp;
        xy[8][0] = width_tmp;                   xy[8][1] = (height_tmp >> 1);
        xy[9][0] = (width_tmp >> 1);            xy[9][1] = height_tmp;
        xy[10][0] = width_tmp + (width_tmp >> 1);
                                                xy[10][1] = height_tmp;
        xy[11][0] = width_tmp;

                    xy[11][1] = height_tmp + (height_tmp >> 1);

        CJinfo.Mode = 0;              /* compress test mode */
        CJinfo.Quality = 100;
        for( i = 0, HEAP[0] = 0; i < 12; i ++ ){
                CJinfo.StartX = ( xy[i][0] << 4 );
                CJinfo.StartY = ( xy[i][1] << 4 );
                jpeg_Compress();    /* Do it! */
                HEAP[0] += CJinfo.FileSize;
        }
        CJinfo.Quality = 75;
        for( i = 0, HEAP[1] = 0; i < 12; i ++ ){
```

```
            CJinfo.StartX = ( xy[i][0] << 4 );
            CJinfo.StartY = ( xy[i][1] << 4 );
            jpeg_Compress();     /* Do it! */
            HEAP[1] += CJinfo.FileSize;
        }
        CJinfo.Quality = 50;
        for( i = 0, HEAP[2] = 0; i < 12; i ++ ){
            CJinfo.StartX = ( xy[i][0] << 4 );
            CJinfo.StartY = ( xy[i][1] << 4 );
            jpeg_Compress();     /* Do it! */
            HEAP[2] += CJinfo.FileSize;
        }

        /* Now, we got the sum:
            HEAP[0]: in case Quality = 100
            HEAP[1]: in case Quality = 75
            HEAP[2]: in case Quality = 50   */

        if( AVR_BITS >= HEAP[0] ){
            Quality = 100;
        } else if( AVR_BITS >= HEAP[1] ){
            Quality = ( HEAP[0] * 75 + AVR_BITS * 25 - HEAP[1] * 100 ) /
                            ( HEAP[0] - HEAP[1] );
        } else if( AVR_BITS >= HEAP[2] ){

            Quality = ( HEAP[1] * 50 + AVR_BITS * 25 - HEAP[2] * 75 ) /
                            ( HEAP[1] - HEAP[2] );
        } else {
            Quality = ( AVR_BITS * 50 ) / HEAP[2];
        }
        /* Returns appropriate Quality (0 to 100) */
        return     Quality;
}

int
compress_main()
{
        int        ret;

        CJinfo.ErrorState = 0;            /* initialize */
            /* jpeg buffer start address */
        CJinfo.JPEG_Buff_Bptr = jpegbuffer;
            /* jpeg buffer terminate address */
        CJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
            /* compress parameter */
        CJinfo.Width = IMAGE_WIDTH;
        CJinfo.Height = IMAGE_HEIGHT;
        CJinfo.Mode = 1;                  /* normal compress mode */
        CJinfo.StartX = 0;
        CJinfo.StartY = 0;

        while( 1 ){
            ret = jpeg_Compress();
            if( ret ==  JPEG_OK ){
                    move_jpeg();
                    return 1;     /* complite */
```

```
                }
                if( ret ==  JPEG_ERR ) return 0; /* error */
                if( ret & JPEG_CONT1 ){          /* jpegbuffer full */
                        move_jpeg();
                } else {

                        reutrn 0;     /* error ? */
                }
        }
}

int
compress()
{
        compress_parameter_ini();
        CJinfo.Quality = compress_test();
        return compress_main();
}
#endif    /* TINY_VRAM */


/***** Expansion/analysis sample program *****/
/***** Analysis sample program *****/
void
next_jpeg()
{
        /* Update jpegbuffer */
}

int
analyze()
{
        int        ret;

        DJinfo.ErrorState = 0;      /* initialize */
                /* jpeg buffer start address */
        DJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        DJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* To perform analysis related to APPn segment */
                /* APPINFO structure */
        DJinfo.APP_Info_Bptr = &dAppinfo;
                /* compress parameter */
        DJinfo.Mode = 0;            /* analyze mode */

        jpeg_next();         /* get first jpeg file data */
        while( 1 ){
                ret = jpeg_Decompress();
                if( ret == JPEG_OK ){
                        return 1;/* complite */
                }
                if( ret == JPEG_ERR ) return 0;  /* error */
                if( ret & JPEG_CONT1 ){          /* jpegbuffer come to end */
                        next_jpeg();
                } else {
                        return 0;/* error ? */
```

```
                }

        }
}

/***** Expansion sample program *****/
/*#define CLIPPING*/
#ifdef    TINY_VRAM
void
take_out_vram()
{
          /* Save contents of VRAM */
}
#endif    /* TINY_VRAM */

int
decompress()
{
        int        ret;

        DJinfo.ErrorState = 0;          /* initialize */
                /* jpeg buffer start address */
        DJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        DJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* To perform analysis related to APPn segment */
                /* APPINFO structure */
        DJinfo.APP_Info_Bptr = &dAppinfo;
                /* work area for this library */
        DJinfo.Work = (unsigned char *)WorkArea;
                /* decompress parameter */
        DJinfo.StartX = 0;
        DJinfo.StartY = 0;
                /* VRAM information */
        DJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        DJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        DJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        DJinfo.VRAM_Line_Byte = VRAM_LINE;
        DJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
        DJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        DJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
#ifndef   CLIPPING   /* if not clipping mode */
        DJinfo.Mode = 1;                /* normal mode */
/*      DJinfo.Mode = 2;*/              /* 1/4 mode */
/*      DJinfo.Mode = 3;*/              /* 1/16 mode */
/*      DJinfo.Mode = 4;*/              /* 1/64 mode */
#else   /* CLIPPING */
        DJinfo.Mode = 5;                /* clipping mode */
                /* clipping parameter */
        DJinfo.ClipSX = 0;
        DJinfo.ClipSY = 1;
        DJinfo.ClipW = 2;
        DJinfo.ClipH = 3;
#endif  /* CLIPPING */
```

```
        jpeg_next();           /* get first jpeg file data */
        while( 1 ){
                ret = jpeg_Decompress();
                if( ret == JPEG_OK ){
                        return 1;      /* complite */
                }
                if( ret == JPEG_ERR ) return 0;  /* error */
                if( ret & JPEG_CONT1 ){          /* jpegbuffer come to end */
                        next_jpeg();
                } else
#ifdef    TINY_VRAM
                if( ret & JPEG_CONT2 ){          /* VRAM come to end */
                        take_out_vram();
                } else
#endif    /* TINY_VRAM */
                {
                        return 0;      /* error ? */
                }
        }
}
```

APPENDIX B SAMPLE PROGRAM SOURCE LIST (AP705100-B03)

B.1 SAMPLE PROGRAM SOURCE LIST FOR BASIC LIBRARY

```
/*****************************************************************
 *  Copyright (C) NEC Corporation 1995, 1996             *
 *  All rights reserved by NEC Corporation.              *
 *  Use of copyright notice does not evidence publication  *
 *****************************************************************/

/***** This file is sample usage program
        for V830 JPEG middle-ware library. *****/

#include   "jpeg.h"  /* JPEG library header file */

/*#define  RGB*/
/*** Validate this define to access VRAM in units of 8 or 16 lines***/
/*#define  TINY_VRAM*/

extern int jpeg_Decompress();  /* Decompress main routine */
extern int jpeg_Compress();     /* Compress main routine */

extern char LuminanceQtbl[64];                    /* default Quality table */
extern char ChrominanceQtbl[64];                  /* default Quality table */
extern char DHT_markerLuminanceDC[33];      /* default Huffman table */
extern char DHT_markerChrominanceDC[33];   /* default Huffman table */
extern char DHT_markerLuminanceAC[183];     /* default Huffman table */
extern char DHT_markerChrominanceAC[183];  /* default Huffman table */

/* define of numeric value related to VRAM */
#define   VRAM_ADDR  0x10000000 /* VRAM address */
#define   VRAM_WIDTH 640
#define   VRAM_HEIGHT              480
#define   VRAM_PIXEL  4
#define   VRAM_GAP1   1
#define   VRAM_GAP2   2
#define   VRAM_LINE  (VRAM_WIDTH * VRAM_PIXEL)

/* Number of vertical and horizontal pixels of image */
#define   IMAGE_WIDTH              224
#define   IMAGE_HEIGHT             144

/* Declaration of structure */
CJINFO              CJinfo;                  /* structure for jpeg compress library */
DJINFO              DJinfo;                  /* structure for jpeg decompress library */
APPINFO             cAppinfo, dAppinfo;  /* structure for APPn segment */
/* Declaration of external RAM work area
(In this example, the compression library and expansion library are not executed at the same time.) */
unsigned char         WorkArea[0x1000];   /* library work area */
#define    JPEGBUFFSIZE       0x10000
/*#define  JPEGBUFSIZE          1*/
unsigned char         jpegbuffer[JPEGBUFFSIZE];
/* Example of character string to be embedded in APPn segment */
```

```
unsigned char         str1[] = "JPEG middle-ware library.";
unsigned char         str2[] = "V830(uPD705100) 32-bit RISC Microcomputer";
/* Comment marker */
unsigned char jpeg_COMStr[] = "This is a Comment Marker.";


/***** Compression sample program *****/
void
move_jpeg()
{
        /* Save contents of jpegbuffer */
}

#ifdef    TINY_VRAM
void
replace_vram()
{
        /* Update VRAM */
}

int
compress()
{
        int      ret;

        CJinfo.ErrorState = 0;              /* initialize */
                /* jpeg buffer start address */
        CJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        CJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* internal data RAM work area start address */
                /* In this case, 0x200 to 0x5FF are used. */
        CJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* APPINFO structure */
        CJinfo.APP_Info_Bptr = &cAppinfo;
        (CJinfo.APP_Info_Bptr)->APP00_Buff_Bptr = str1;
        (CJinfo.APP_Info_Bptr)->APP00_BuffSize = (short)(sizeof(str1) - 1);
        (CJinfo.APP_Info_Bptr)->APP01_Buff_Bptr = str2;
        (CJinfo.APP_Info_Bptr)->APP01_BuffSize = (short)(sizeof(str2) - 1);
                /* work area for this library */
        CJinfo.Work = (unsigned char *)WorkArea;
                /* compress parameter */
        CJinfo.Restart = 0;                 /* Don't use restart marker */
        CJinfo.Width = IMAGE_WIDTH;
        CJinfo.Height = IMAGE_HEIGHT;
        CJinfo.Quality = 75;
        CJinfo.Sampling = SAMPLE22;    /* 4:1:1 */
        CJinfo.Mode = 1;                    /* normal compress mode */
        CJinfo.StartX = 0;
        CJinfo.StartY = 0;
                /* VRAM information */
        CJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        CJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        CJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        CJinfo.VRAM_Line_Byte = VRAM_LINE;
        CJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
```

```
        CJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        CJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
                /* Quality table */
        CJinfo.DQT_Y_Bptr = (char *)LuminanceQtbl;
        CJinfo.DQT_C_Bptr = (char *)ChrominanceQtbl;
                /* Huffman table */
        CJinfo.DHT_DC_Y_Bptr = (char *)DHT_markerLuminanceDC;
        CJinfo.DHT_DC_C_Bptr = (char *)DHT_markerChrominanceDC;
        CJinfo.DHT_AC_Y_Bptr = (char *)DHT_markerLuminanceAC;
        CJinfo.DHT_AC_C_Bptr = (char *)DHT_markerChrominanceAC;

        replace_vram(); /* get first VRAM data */
        while( 1 ){
                ret = jpeg_Compress();
                if( ret == JPEG_OK ){
                        move_jpeg();
                        return 1;  /* complite */
                }
                if( ret == JPEG_ERR ) return 0;  /* error */
                if( ret & JPEG_CONT1 ){          /* jpegbuffer full */
                        move_jpeg();
                } else if( ret & JPEG_CONT2 ){  /* change VRAM */
                        replace_vram();
                } else {
                        return 0;   /* error ? */
                }
        }
}

#else     /* TINY_VRAM */
void
compress_parameter_ini()
{
                /* In this case, 0x200 to 0x5FF are used. */
                /* internal data RAM work area start address */
        CJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* work area for this library */
        CJinfo.Work = (unsigned char *)WorkArea;
                /* APPINFO structure */
        CJinfo.APP_Info_Bptr = &cAppinfo;
        (CJinfo.APP_Info_Bptr)->APP00_Buff_Bptr = str1;
        (CJinfo.APP_Info_Bptr)->APP00_BuffSize = (short)(sizeof(str1) - 1);
        (CJinfo.APP_Info_Bptr)->APP01_Buff_Bptr = str2;
        (CJinfo.APP_Info_Bptr)->APP01_BuffSize = (short)(sizeof(str2) - 1);
                /* compress parameter */
        CJinfo.Restart = 0;              /* Don't use restart marker */
        CJinfo.Sampling = SAMPLE22;   /* 4:1:1 */
                /* VRAM information */
        CJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        CJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        CJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        CJinfo.VRAM_Line_Byte = VRAM_LINE;
        CJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
        CJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        CJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
                /* Quality table */
```

```
        CJinfo.DQT_Y_Bptr = (char *)LuminanceQtbl;
        CJinfo.DQT_C_Bptr = (char *)ChrominanceQtbl;
                /* Huffman table */
        CJinfo.DHT_DC_Y_Bptr = (char *)DHT_markerLuminanceDC;
        CJinfo.DHT_DC_C_Bptr = (char *)DHT_markerChrominanceDC;
        CJinfo.DHT_AC_Y_Bptr = (char *)DHT_markerLuminanceAC;
        CJinfo.DHT_AC_C_Bptr = (char *)DHT_markerChrominanceAC;
}

int
compress_test()
{
/*      Targeted value of number of bits of 12 MCUs */
#define   AVR_BITS    80*12

        int         Quality;
        int         i;
        int         HEAP[3];
        short       xy[12][2]; /* 12 test point, ( x, y ) */
        short       width_tmp, height_tmp;

/*      Test 12 MCUs as follows: */
/*      VRAM image              */
/*      +-----------------+     */
/*      |0               1|     */
/*      |        8        |     */
/*      |       4 5       |     */
/*      |   9   6 7   10  |     */
/*      |        11       |     */
/*      |2               3|     */
/*      +-----------------+     */
        width_tmp = (IMAGE_WIDTH >> 4);
        height_tmp = (IMAGE_HEIGHT >> 4);
        xy[0][0] = 0;                       xy[0][1] = 0;
        xy[1][0] = width_tmp - 1;           xy[1][1] = 0;
        xy[2][0] = 0;                       xy[2][1] = height_tmp - 1;
        xy[3][0] = width_tmp - 1;           xy[3][1] = height_tmp - 1;
        width_tmp >>= 1;    /* half of width */
        height_tmp >>= 1;   /* half of height */
        xy[4][0] = width_tmp - 1;           xy[4][1] = height_tmp - 1;
        xy[5][0] = width_tmp;               xy[5][1] = height_tmp - 1;
        xy[6][0] = width_tmp - 1;           xy[6][1] = height_tmp;
        xy[7][0] = width_tmp;               xy[7][1] = height_tmp;
        xy[8][0] = width_tmp;               xy[8][1] = (height_tmp >> 1);
        xy[9][0] = (width_tmp >> 1);        xy[9][1] = height_tmp;

        xy[10][0] = width_tmp + (width_tmp >> 1);
                                            xy[10][1] = height_tmp;
        xy[11][0] = width_tmp;
                        xy[11][1] = height_tmp + (height_tmp >> 1);

        CJinfo.Mode = 0;            /* compress test mode */
        CJinfo.Quality = 100;
        for( i = 0, HEAP[0] = 0; i < 12; i ++ ){
                CJinfo.StartX = ( xy[i][0] << 4 );
                CJinfo.StartY = ( xy[i][1] << 4 );
```

```
            jpeg_Compress();     /* Do it! */
            HEAP[0] += CJinfo.FileSize;
        }
        CJinfo.Quality = 75;
        for( i = 0, HEAP[1] = 0; i < 12; i ++ ){
            CJinfo.StartX = ( xy[i][0] << 4 );
            CJinfo.StartY = ( xy[i][1] << 4 );
            jpeg_Compress();     /* Do it! */
            HEAP[1] += CJinfo.FileSize;
        }
        CJinfo.Quality = 50;
        for( i = 0, HEAP[2] = 0; i < 12; i ++ ){
             CJinfo.StartX = ( xy[i][0] << 4 );
             CJinfo.StartY = ( xy[i][1] << 4 );
             jpeg_Compress();    /* Do it! */
             HEAP[2] += CJinfo.FileSize;
        }

        /* Now, we got the sum:
             HEAP[0]: in case Quality = 100
             HEAP[1]: in case Quality = 75
             HEAP[2]: in case Quality = 50   */

        if( AVR_BITS >= HEAP[0] ){
             Quality = 100;
        } else if( AVR_BITS >= HEAP[1] ){

             Quality = ( HEAP[0] * 75 + AVR_BITS * 25 - HEAP[1] * 100 ) /
                            ( HEAP[0] - HEAP[1] );
        } else if( AVR_BITS >= HEAP[2] ){
             Quality = ( HEAP[1] * 50 + AVR_BITS * 25 - HEAP[2] * 75 ) /
                            ( HEAP[1] - HEAP[2] );
        } else {
             Quality = ( AVR_BITS * 50 ) / HEAP[2];
        }
        /* Returns appropriate Quality (0 to 100) */
        return     Quality;
}

int
compress_main()
{
        int        ret;

        CJinfo.ErrorState = 0;          /* initialize */
            /* jpeg buffer start address */
        CJinfo.JPEG_Buff_Bptr = jpegbuffer;
            /* jpeg buffer terminate address */
        CJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
            /* compress parameter */
        CJinfo.Width = IMAGE_WIDTH;
        CJinfo.Height = IMAGE_HEIGHT;
        CJinfo.Mode = 1;                /* normal compress mode */
        CJinfo.StartX = 0;
        CJinfo.StartY = 0;
```

```
        while( 1 ){
                ret = jpeg_Compress();
                if( ret ==  JPEG_OK ){
                        move_jpeg();
                        return 1;  /* complite */
                }
                if( ret ==  JPEG_ERR ) return 0;  /* error */
                if( ret & JPEG_CONT1 ){           /* jpegbuffer full */
                        move_jpeg();
                } else {
                        reutrn 0;    /* error ? */
                }
        }
}

int
compress()
{
   compress_parameter_ini();
   CJinfo.Quality = compress_test();
   return compress_main();
}
#endif   /* TINY_VRAM */

/***** Expansion/analysis sample program *****/
/***** Analysis sample program *****/
void
next_jpeg()
{
        /* Update jpegbuffer */
}

int
analyze()
{
        int       ret;

        DJinfo.ErrorState = 0;           /* initialize */
                /* jpeg buffer start address */
        DJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        DJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* internal data RAM work area start address */
                /* In this case, 0x200 to 0x2FF are used. */
        DJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* To perform analysis related to APPn segment */
                /* APPINFO structure */
        DJinfo.APP_Info_Bptr = &dAppinfo;
                /* compress parameter */
        DJinfo.Mode = 0;                /* analyze mode */

        jpeg_next();           /* get first jpeg file data */
        while( 1 ){
                ret = jpeg_Decompress();
                if( ret ==  JPEG_OK ){
                        return 1;     /* complite */
```

```
                }
                if( ret == JPEG_ERR ) return 0; /* error */
                if( ret & JPEG_CONT1 ){         /* jpegbuffer come to end */
                        next_jpeg();

                } else {
                        return 0;    /* error ? */
                }
        }
}


/***** Expansion sample program *****/
/*#define CLIPPING*/
#ifdef    TINY_VRAM
void
take_out_vram()
{
         /* Save contents of VRAM */
}
#endif    /* TINY_VRAM */

int
decompress()
{
        int       ret;

        DJinfo.ErrorState = 0;          /* initialize */
                /* jpeg buffer start address */
        DJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        DJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* internal data RAM work area start address */
                /* In this case, 0x200 to 0x5FF are used. */
        DJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* To perform analysis related to APPn segment */
                /* APPINFO structure */
        DJinfo.APP_Info_Bptr = &dAppinfo;
                /* work area for this library */
        DJinfo.Work = (unsigned char *)WorkArea;
                /* decompress parameter */
        DJinfo.StartX = 0;
        DJinfo.StartY = 0;
                /* VRAM information */
        DJinfo.VRAM_Bptr = (unsigned char *)VRAM_ADDR;
        DJinfo.VRAM_W_Pixel = VRAM_WIDTH;
        DJinfo.VRAM_H_Pixel = VRAM_HEIGHT;
        DJinfo.VRAM_Line_Byte = VRAM_LINE;
        DJinfo.VRAM_Pixel_Byte = VRAM_PIXEL;
        DJinfo.VRAM_Gap1_Byte = VRAM_GAP1;
        DJinfo.VRAM_Gap2_Byte = VRAM_GAP2;
#ifndef CLIPPING   /* if not clipping mode */
        DJinfo.Mode = 1;                /* normal mode */
/*      DJinfo.Mode = 2;*/              /* 1/4 mode */
/*      DJinfo.Mode = 3;*/              /* 1/16 mode */
/*      DJinfo.Mode = 4;*/              /* 1/64 mode */
```

```
#else    /* CLIPPING */
        DJinfo.Mode = 5;             /* clipping mode */
                /* clipping parameter */
        DJinfo.ClipSX = 0;
        DJinfo.ClipSY = 1;
        DJinfo.ClipW = 2;
        DJinfo.ClipH = 3;
#endif   /* CLIPPING */

        jpeg_next();        /* get first jpeg file data */
        while( 1 ){
                ret = jpeg_Decompress();
                if( ret == JPEG_OK ){
                        return 1;   /* complite */
                }
                if( ret == JPEG_ERR ) return 0; /* error */
                if( ret & JPEG_CONT1 ){         /* jpegbuffer come to end */
                        next_jpeg();
                } else

#ifdef     TINY_VRAM
                if( ret & JPEG_CONT2 ){   /* VRAM come to end */
                        take_out_vram();
                } else
#endif     /* TINY_VRAM */
                {
                        return 0;   /* error ? */
                }
        }
}
```

* B.2 SAMPLE PROGRAM SOURCE LIST FOR ADDITIONAL LIBRARY

(1) Sample in single-pass mode

```
#include  "jpegex.h"

extern unsigned char            jpegfile[100000];
struct  JPEGEXINFO JPINFO;
struct  JPEGEXWORK              JPWORK;
struct  JPEGEXVIDEO             JPVIDEO;

#define   WORKBUFFSIZE       0x1000000
unsigned int            Work[WORKBUFFSIZE/sizeof(int)];

void JPEGEXdecodeAPP( int APPnumber, int JpegStreamOffsetIdx, int SegmentLength )
{
}

unsigned char jpegtmp[0x1000];
unsigned char *jpegptr = jpegfile;

void JPEGEXGetJpegStream( struct JPEGEXBUFF *JPBUFF )
{
        int i;

        for( i = 0; i < 0x1000; i ++ ){
                jpegtmp[i] = *jpegptr++;
        }
        JPBUFF->JPEGBUFF = jpegtmp;
        JPBUFF->JPEGBUFFLEN = 0x1000;
}

void main()
{
        JPINFO.Mode = ModeStart;
        JPINFO.Policy = PolicyLuminanceOutOnly;
        JPINFO.Work = &JPWORK;
        JPWORK.Work1 = (unsigned int *)0;
        JPWORK.Work1Len = (int)0;
        JPWORK.Work2 = Work;
        JPWORK.Work2Len = (int)WORKBUFFSIZE;
        JPINFO.Video = &JPVIDEO;
        JPVIDEO.VRAMAddress = (unsigned char *)0x60000000;
        JPVIDEO.VRAMWidth = 640;
        JPVIDEO.VRAMHeight = 480;
        JPVIDEO.VRAMPixel = 4;
        JPVIDEO.VRAMLine = 640*4;
        JPVIDEO.VRAMGap0 = 0;
        JPVIDEO.VRAMGap1 = 1;
        JPVIDEO.VRAMGap2 = 2;
        JPVIDEO.ClipStartX = 13;
        JPVIDEO.ClipStartY = 15;
        JPVIDEO.ClipWidth = 321;
        JPVIDEO.ClipHeight = 311;
        JPEGEXdecode( &JPINFO );
}
```

(2) Sample in two-pass mode

```
#include  "jpegex.h"

extern unsigned char            jpegfile[100000];
struct JPEGEXINFO JPINFO;
struct JPEGEXWORK               JPWORK;
struct JPEGEXVIDEO              JPVIDEO;

#define   WORKBUFFSIZE          0x2000
unsigned int           Work[WORKBUFFSIZE/sizeof(int)];

void JPEGEXdecodeAPP( int APPnumber, int JpegStreamOffsetIdx, int SegmentLength )
{
}

void JPEGEXGetJpegStream( struct JPEGEXBUFF *JPBUFF )
{
        JPBUFF->JPEGBUFF = jpegfile;
        JPBUFF->JPEGBUFFLEN = 0x100000;
}

void main()
{
        JPINFO.Mode = ModeStart;
        JPINFO.Policy = PolicyLuminanceOutOnly|Policy2passEnable;
        JPINFO.Work = &JPWORK;
        JPWORK.Work1 = (unsigned int *)0;
        JPWORK.Work1Len = (int)0;
        JPWORK.Work2 = Work;
        JPWORK.Work2Len = (int)WORKBUFFSIZE;
        JPINFO.Video = &JPVIDEO;
        JPVIDEO.VRAMAddress = (unsigned char *)0x60000000;
        JPVIDEO.VRAMWidth = 640;
        JPVIDEO.VRAMHeight = 480;
        JPVIDEO.VRAMPixel = 4;
        JPVIDEO.VRAMLine = 640*4;
        JPVIDEO.VRAMGap0 = 0;
        JPVIDEO.VRAMGap1 = 1;
        JPVIDEO.VRAMGap2 = 2;
        JPVIDEO.ClipStartX = 13;
        JPVIDEO.ClipStartY = 15;
        JPVIDEO.ClipWidth = 321;
        JPVIDEO.ClipHeight = 311;
        JPEGEXdecode( &JPINFO );
}
```

★ APPENDIX C JPEG SAMPLE FILE (FOR AP705100-B03 ADDITIONAL LIBRARY)

C.1 fishp3.jpg (PROGRESSIVE SPECTRAL SELECTION FORMAT)

SOI	
APP0	JFIF
APPD	Thumb nail (JPEG format of base line)
COM	
APPE	
DQT	
SOF2	
DHT	
SOS	FF DA 00 0C 03 01 01 02 11 03 11 00 00 00
Compressed data	DC coefficient of Y, Cb, and Cr components
SOS	FF DA 00 08 01 02 00 01 05 00
Compressed data	AC1 to AC5 of Cb component
SOS	FF DA 00 08 01 03 00 01 05 00
Compressed data	AC1 to AC5 of Cr component
SOS	FF DA 00 08 01 01 00 01 05 00
Compressed data	AC1 to AC5 of Y component
SOS	FF DA 00 08 01 02 02 06 3F 00
Compressed data	AC6 to AC63 of Cb component
SOS	FF DA 00 08 01 03 02 06 3F 00
Compressed data	AC6 to AC63 of Cr component
SOS	FF DA 00 08 01 01 01 06 3F 00
Compressed data	AC6 to AC63 of Y component
EOI	

C.2 fishp4.jpg (PROGRESSIVE SUCCESSIVE APPROXIMATION FORMAT)

SOI	
⁀	Same as fishp3.jpg
DHT	
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 00
Compressed data	DC coefficient of Y, Cb, and Cr
SOS	FF DA 00 08 01 02 00 01 05 01
Compressed data	High-order bits except low-order one bit of AC1 to AC5 of Cb component
SOS	FF DA 00 08 01 03 00 01 05 01
Compressed data	High-order bits except low-order one bit of AC1 to AC5 of Cr component
SOS	FF DA 00 08 01 01 00 01 05 01
Compressed data	High-order bits except low-order one bit of AC1 to AC5 of Y component
SOS	FF DA 00 08 01 02 02 06 3F 01
Compressed data	High-order bits except low-order one bit of AC6 to AC63 of Cb component
SOS	FF DA 00 08 01 03 02 06 3F 01
Compressed data	High-order bits except low-order one bit of AC6 to AC63 of Cr component
SOS	FF DA 00 08 01 01 01 06 3F 01
Compressed data	High-order bits except low-order one bit of AC6 to AC63 of Y component
SOS	FF DA 00 08 01 02 03 01 3F 10
Compressed data	Low-order one bit of AC1 to AC63 of Cb component
SOS	FF DA 00 08 01 03 03 01 3F 10
Compressed data	Low-order one bit of AC1 to AC63 of Cr component
SOS	FF DA 00 08 01 01 03 01 3F 10
Compressed data	Low-order one bit of AC1 to AC63 of Y component
EOI	

C.3 fishp5.jpg (PROGRESSIVE SUCCESSIVE APPROXIMATION FORMAT)

SOI	
⁓	Same as fishp3.jpg
DHT	
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 01
Compressed data	DC coefficient of Y, Cb, and Cr components, high-order bits except low-order one bit
SOS	FF DA 00 08 01 01 00 01 05 02
Compressed data	High-order bits except low-order two bits of AC1 to AC5 of Y component
SOS	FF DA 00 08 01 02 00 01 05 02
Compressed data	High-order bits except low-order two bits of AC1 to AC5 of Cb component
SOS	FF DA 00 08 01 03 00 01 05 02
Compressed data	High-order bits except low-order two bits of AC1 to AC5 of Cr component
SOS	FF DA 00 08 01 02 02 06 3F 02
Compressed data	High-order bits except low-order two bits of AC6 to AC63 of Cb component
SOS	FF DA 00 08 01 03 02 06 3F 02
Compressed data	High-order bits except low-order two bits of AC6 to AC63 of Cr component
SOS	FF DA 00 08 01 01 01 06 3F 02
Compressed data	High-order bits except low-order two bits of AC6 to AC63 of Y component
SOS	FF DA 00 08 01 01 03 01 3F 21
Compressed data	Second lowest bit of AC1 to AC63 of Y component
SOS	FF DA 00 08 01 02 03 01 3F 21
Compressed data	Second lowest bit of AC1 to AC63 of Cb component
SOS	FF DA 00 08 01 03 03 01 3F 21
Compressed data	Second lowest bit of AC1 to AC63 of Cr component
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 10
Compressed data	Least significant bit of DC coefficient of Y, Cb, and Cr components
SOS	FF DA 00 08 01 01 03 01 3F 10
Compressed data	Least significant bit of AC1 to AC63 of Y component
SOS	FF DA 00 08 01 02 03 01 3F 10
Compressed data	Least significant bit of AC1 to AC63 of Cb component
SOS	FF DA 00 08 01 03 03 01 3F 10
Compressed data	Least significant bit of AC1 to AC63 of Cr component
EOI	

[MEMO]

APPENDIX D INDEX

[MEMO]

Phase-out/Discontinued

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❏	❏	❏	❏
Technical Accuracy	❏	❏	❏	❏
Organization	❏	❏	❏	❏