お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

ユーザーズ・マニュアル

μSAP705100-B03
μSAP70732-B03

JPEG ミドルウエア

保守／廃止

対象デバイス

μSAP705100-B03：V830 ファミリ™
μSAP70732-B03：V810 ファミリ™

資料番号　U11052JJ4V0UM00（第４版）
発行年月　April 1999　N　CP(K)

保守／廃止

〔メ　モ〕

保守／廃止

目次要約

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**

M7A　98.8

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
p.41	**1.2.2（1）（f）SOFn（Start of frame）マーカ**　説明追加
p.43	**1.2.2（1）（g）SOS（Start of scan）マーカ**　説明追加
p.48	**1.3.4　追加ライブラリの特徴（AP705100-B03）**　追加
p.49	**1.3.5　基本ライブラリと追加ライブラリの違い（AP705100-B03）**　追加
p.52	**1.3.6（3）AP705100-B03（NEC版）**　パッケージ内容修正
p.54	**1.3.6（4）AP705100-B03（GHS版）**　パッケージ内容修正
p.56	**1.3.7（3）メモリ容量**　追加ライブラリの記述追加
p.58	**1.3.8　セクション名，シンボル名規約**　追加ライブラリの記述追加
p.80	**表２－６　CJINFO構造体（AP70732-B03）**　注追加
p.81	**表２－７　CJINFO構造体（AP705100-B03）**　注追加
p.82	**表２－８　DJINFO構造体（AP70732-B03）**　注追加
p.83	**表２－９　DJINFO構造体（AP705100-B03）**　注追加
p.151	**第３章　プログレッシブ対応追加ライブラリ仕様**　追加
旧版p.143	**３.２　ヘルプのインストール**　削除
p.216	**B.2　追加ライブラリ用サンプル・プログラム・ソース・リスト**　追加
p.219	**付録C　JPEGサンプル・ファイル（AP705100-B03　追加ライブラリ用）**　追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

保守／廃止

はじめに

対象者　このマニュアルは，V800シリーズ™の応用システムを設計，開発するユーザを対象としています。

目　的　このマニュアルは，次の構成に示すμSAP705100-B03，70732-B03の機能をユーザに理解していただくことを目的としています。

構　成　このマニュアルは，大きく分けて次の内容で構成しています。

- 概　説
- ライブラリ仕様
- サンプル・プログラム・ソース・リスト

読み方　このマニュアルでは「μSAP705100-B03」という製品名を「AP705100-B03」に，「μSAP70732-B03」という製品名を「AP70732-B03」に置き換えて説明しています。

凡　例

注　：本文中につけた注の説明
注意　：気をつけて読んでいただきたい内容
備考　：本文の補足説明
数の表記　：２進数…××××または××××B
10進数…××××
16進数…0x××××

２のべき数を示す接頭語（アドレス空間，メモリ容量）：
K（キロ）$2^{10} = 1024$
M（メガ）$2^{20} = 1024^2$

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

V810ファミリに関する資料

製品名		データ・シート	ユーザーズ・マニュアル	
愛称	品名		ハードウエア編	アーキテクチャ編
V821™	μPD70741	U11678J	U10077J	U10082J

V830ファミリに関する資料

製品名		データ・シート	ユーザーズ・マニュアル	
愛称	品名		ハードウエア編	アーキテクチャ編
V830™	μPD705100	U11483J	U10064J	U12496J
V831™	μPD705101	U12979J	U12273J	
V832™	μPD705102	U13675J	U13577J	

保守／廃止

V810ファミリ開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
CA732（Cコンパイラ）	操作編（UNIX™ベース）	U11013J
	操作編（Windows™ベース）	U11068J
	アセンブリ言語編	U11016J
	C言語編	U11010J
	プロジェクト・マネージャ編	U11991J
RX732（リアルタイムOS）	基礎編	U10346J
	テクニカル編	U10490J
	ニュークリアス・インストレーション編	U10347J

V830ファミリ開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名			資料番号
CA830（Cコンパイラ）		操作編（UNIXベース）	U11013J
		操作編（Windowsベース）	U11068J
		アセンブリ言語編	U11014J
		C言語編	U11010J
		プロジェクト・マネージャ編	U11991J
RX830 （リアルタイムOS）	ITRON1	基礎編	U11730J
		インストレーション編	U11731J
		テクニカル編	U11713J
	μITRON Ver.3.0	基礎編	U13152J
		インストレーション編	U13151J
		テクニカル編	U13150J

・Green Hills Software™, Inc.（GHS）製ツールに関する資料

GHS製ツールは，日本国内では下記で取り扱っております。各種製品とそれに関する資料については，下記へお問い合わせください。

株式会社アドバンスド　データ　コントロールズ（ADaC）　　TEL（03）3576-5351

保守／廃止

目　　次

保守／廃止

図の目次（1/3）

保守／廃止

図の目次（2/3）

保守／廃止

図の目次（3/3）

保守／廃止

表の目次（1/3）

保守／廃止

表の目次（2/3）

保守／廃止

表の目次（3/3）

〔メ　モ〕

第1章　概　　説

1.1　ミドルウエアとは

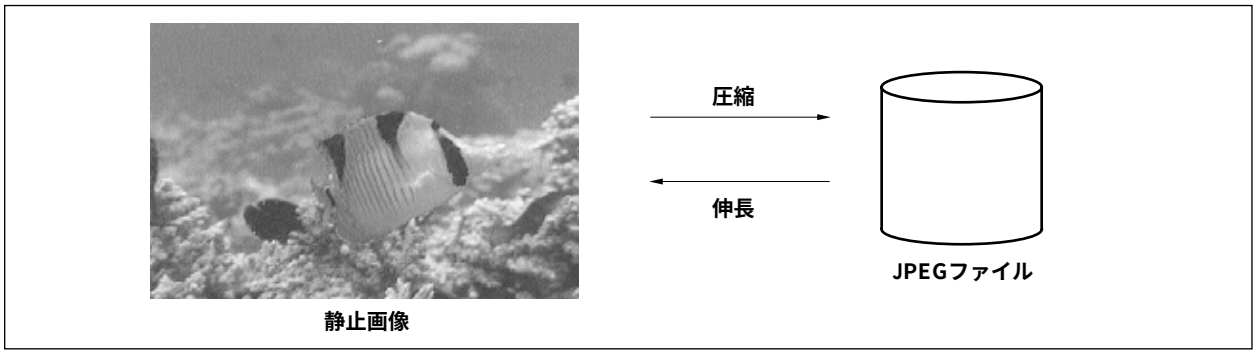
ミドルウエアとは，プロセッサの性能を最大限に引き出せるようにチューニングされたソフトウエア群で，従来ハードウエアが行っていた処理をソフトウエアで実現したものです。RISC（reduced instruction set computer）という高性能プロセッサの出現により，専用ハードウエアに頼ることなくROM/RAMのみで処理を実現するミドルウエアの発想がにわかに現実味を帯びてきました。

NECではヒューマン・マシン・インタフェースや信号処理技術をミドルウエアの形で用意し，さまざまなユーザ・ニーズに対応して優れたシステム・ソリューションを提供していきます。

1.2　JPEGとは

1991年に勧告された静止画像の圧縮／伸長の国際標準規格で，Joint Photographic Experts Groupの略です。これに関してはISO/IEC 10918-1, -2という規格書が発行されています。

図１－１　画像の圧縮／伸長

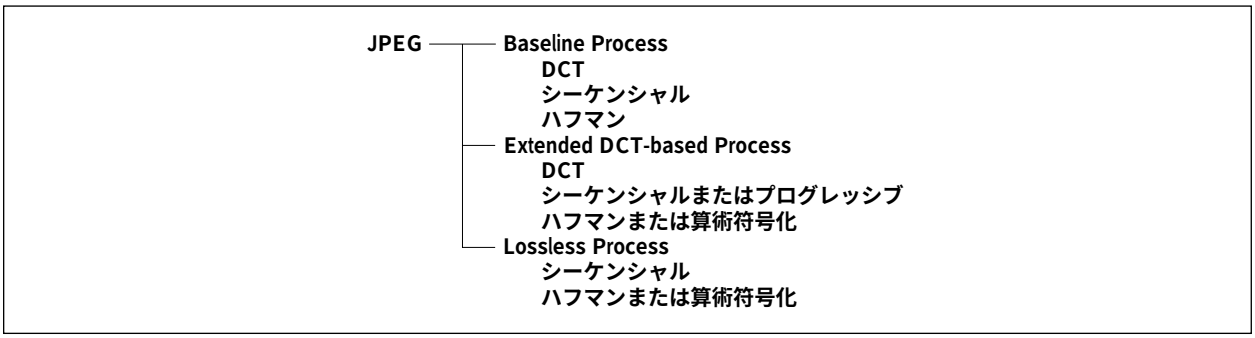


保守／廃止

1.2.1　JPEGの概要

JPEGにはいくつかのバリエーションがあります。最初に画像の概略を表示し，しだいに鮮明な表示へと変化させていくプログレッシブと呼ばれるものや，画像が圧縮前の画像に完全に復元できるロスレスと呼ばれるものなどです。AP705100-B03, AP70732-B03では最も一般的なベースラインDCTと呼ばれるものに基本ライブラリで対応しています。さらにAP705100-B03では，プログレッシブ・フォーマットにも追加ライブラリで対応しています（伸長機能のみ）。

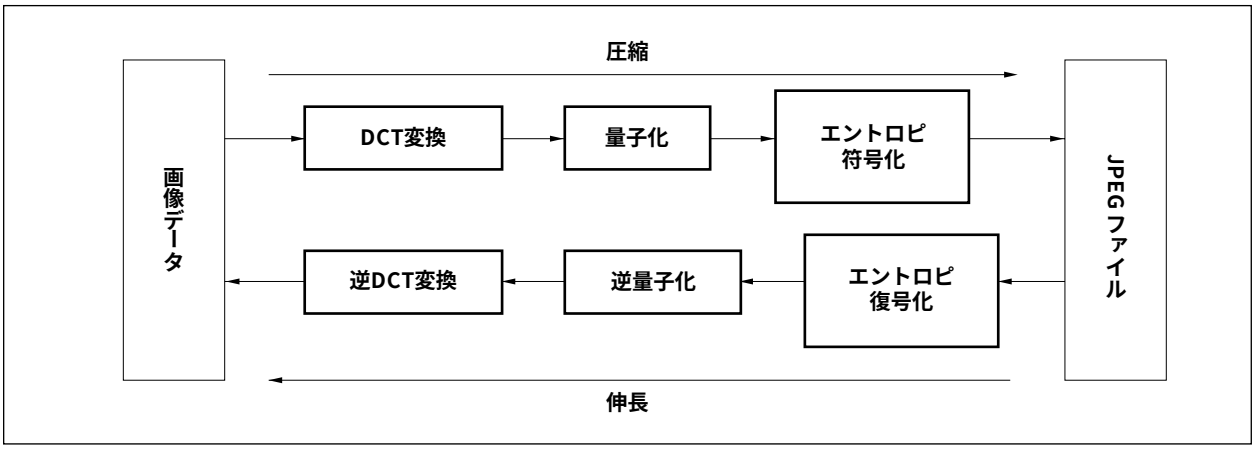
図１－２　JPEGの分類



（1）JPEG処理の流れ

JPEG圧縮では①DCT変換，②量子化，③エントロピ圧縮の３段階で情報を圧縮していきます。JPEG伸長ではその逆で①エントロピ伸長，②逆量子化，③逆DCT変換の３段階で画像を再現します。

図１－３　JPEG処理



DCT変換（discrete cosine transform：離散コサイン変換）とは周波数分解する処理です。量子化とはDCT変換で得られた（周波数分解された）データから，人間の目で見て分かりにくい周波数成分をより積極的に落とす方向で情報量を切り落とす処理です。エントロピ符号化とは，一般に知られるような可逆圧縮／伸長のことで，ベースラインDCT／プログレッシブではハフマンを応用した技術を用いています。

AP705100-B03, AP70732-B03では，処理を高速化させる目的で，DCT変換と量子化を同一関数内で，また，エントロピ複号化と逆量子化を同一関数内で処理しています。

保守／廃止

（2）YCbCr/RGB

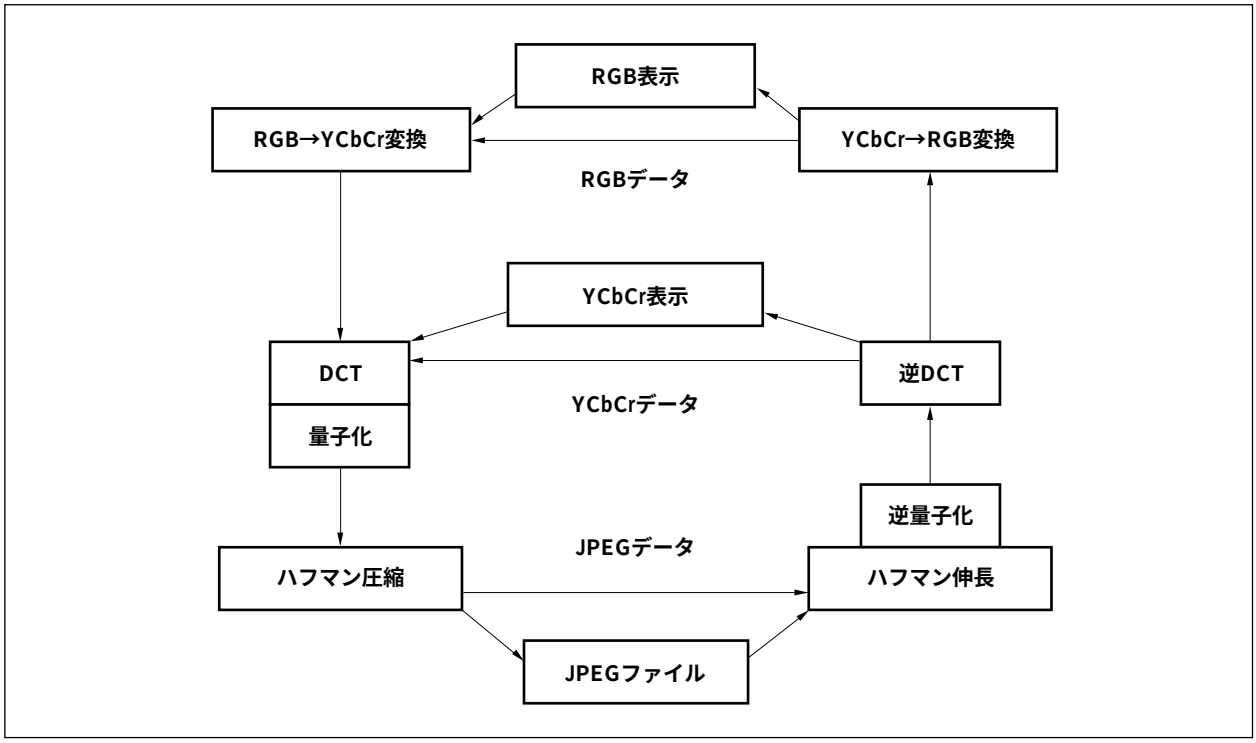
カラーのJPEGは画像をYCbCrという３色の色空間で圧縮／伸長します（モノクロは輝度のみです）。そこで，画像データがYCbCrではなくRGBであった場合には，YCbCrに変換してから圧縮し，また伸長した結果を表示する前にYCbCrからRGBに変換する処理が加わります。

YCbCrのYは輝度（明るさの指標），Cb/Crは色差（Cbは緑から青への色調の差，Crは緑から赤への色調の差）のことで，RGBとの間には次のような変換式が成り立ちます。

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

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

図１－４　JPEG処理概要



保守／廃止

（3）サンプリングとMCU

JPEGが処理を行う最小単位をMCU（minimum coded unit）と呼んでいます。またそのMCUをY/Cb/Crに分離し，８×８ピクセル単位にしたものをブロックと呼んでいます。

このとき，１つのMCUからYを４ブロック，Cbを１ブロック，Crを１ブロック取るような場合を“サンプル比４：１：１”と言います。同様に，１MCUからYを２ブロック，CbとCrをそれぞれ１ブロック取るような場合を２：１：１，１MCUからY, Cb, Crをそれぞれ１ブロックずつ取るような場合を１：１：１と呼びます。

表１－１　サンプル比とMCU

MCU	サンプル比	ブロック
縦16ピクセル 横16ピクセル	4：1：1 （H：V＝2：2）	Y：４ブロック Cb：１ブロック，Cr：１ブロック
縦８ピクセル 横32ピクセル	4：1：1 （H：V＝4：1）	Y：４ブロック Cb：１ブロック，Cr：１ブロック
縦８ピクセル 横16ピクセル	2：1：1	Y：２ブロック Cb：１ブロック，Cr：１ブロック
縦８ピクセル 横８ピクセル	1：1：1	Y：１ブロック Cb：１ブロック，Cr：１ブロック

備考　H：MCUの横方向サンプリング比率
V：MCUの縦方向サンプリング比率

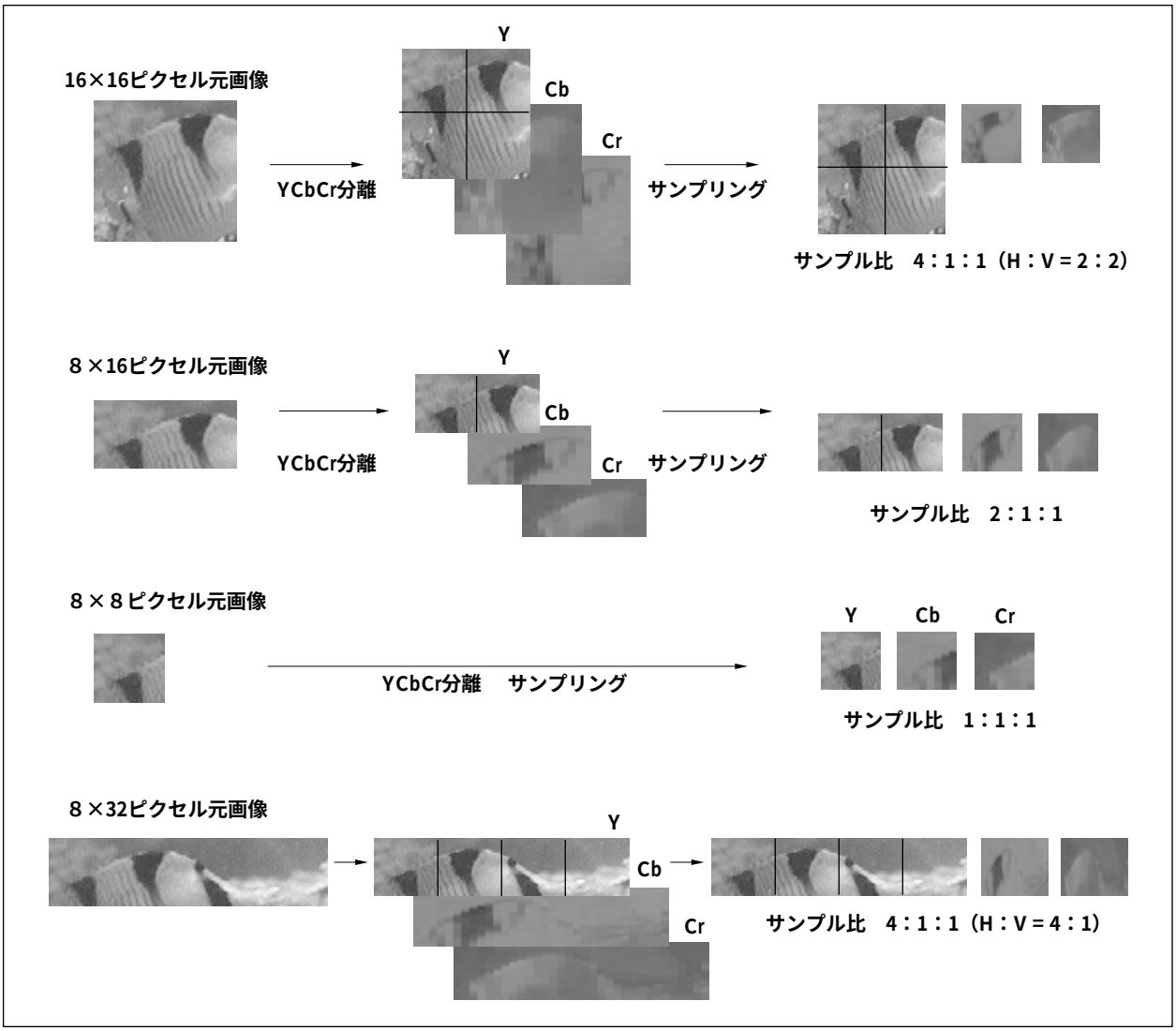
JPEGで許されているサンプル比は表１－１以外にもありますが，AP705100-B03, AP70732-B03の基本ライブラリではこの表にあるサンプル比だけに対応しています。AP705100-B03の追加ライブラリでは任意のサンプル比に対応しています。

JPEG圧縮では，画像をこのMCU単位に格子状に分割することから始まります。逆にJPEG伸長では，それぞれのMCUの処理結果をタイルを敷き詰めるように並べていきます。

たとえばH：V＝２：２のサンプル比４：１：１では画像を縦横それぞれ16ピクセル単位で分割します。次に，その16×16ピクセルをY/Cb/Crに分解し，Y成分を８×８の４つのブロックに分けます。Cb/Crについては16×16ピクセルから８×８ピクセルを作り出します。このとき隣り合う縦横の４ピクセルの平均をとります。これを“間引き”と呼びます。

保守／廃止

図１－５　画像のサンプリング



JPEGでは１：１：１に比べると４：１：１の方が一般的です。

サンプル比４：１：１では，輝度成分に比べ色差成分を軽く扱っています。これは，人間の目が明るさの変化には敏感なのに対し，色の変化には鈍感であることを利用して，人間の目では分からない部分の情報量を省略することで高圧縮を実現しようとする考え方に基づいています。

例として，640×480ピクセルの画像を圧縮することを考えてみます。

この画像をサンプル比４：１：１（H：V＝２：２）で圧縮する場合には，縦横それぞれ16ピクセルごとに区切り，横に40分割，縦に30分割することになります。それぞれのMCUからＹ成分４ブロック，Cb成分１ブロック，Cr成分１ブロックの，あわせて６ブロックを取り出すことになり，画像全体では40×30×6＝7200ブロックが得られます。この7200ブロックそれぞれに対してDCT変換，量子化，ハフマン圧縮を順次適用することになります。

表１－２　サンプル比とブロック

サンプル比	640×480ピクセルの場合		1MCUあたりブロック数	総ブロック数
	横	縦		
４：１：１（H：V＝２：２）	40分割	30分割	6	7200
４：１：１（H：V＝４：１）	20分割	60分割	6	7200
２：１：１	40分割	60分割	4	9600
１：１：１	80分割	60分割	3	14400

備考　H：MCUの横方向サンプリング比率
　　V：MCUの縦方向サンプリング比率

この表のように，サンプル比４：１：１よりも１：１：１のほうが全体としてブロック数は多くなります。ブロック数が多くなれば処理時間もそれだけかかります。また生成されるJPEGファイルのサイズも大きくなります。

JPEG圧縮では，サンプリング後はそれぞれのブロックだけに注目し，そのブロックがＹのものであるかCb/Crのものであるかの情報とともに，ブロック単位でDCT変換，量子化，エントロピ符号化の処理を行います。

JPEG伸長では，エントロピ複号化，逆量子化，逆DCT変換の処理を終了するとブロック単位で結果が得られます。

保守／廃止

（4）DCT変換

DCT変換は次の式による変換です。

DCT変換

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

逆DCT変換

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

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

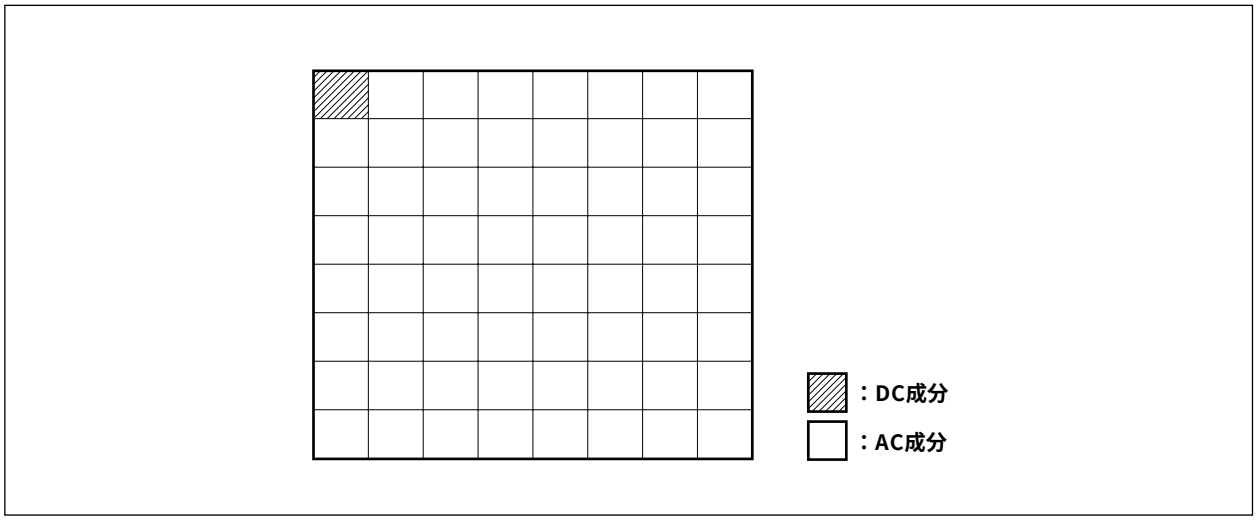
$$= 1 \qquad (w \neq 0)$$

JPEGやMPEGなどの信号処理では，このDCT変換を８×８要素に対して使用するのが一般的です。DCT変換は，縦方向，横方向それぞれを$\cos$（$n\pi/16$），（$n = 0, 1, 2, \ldots, 7$）の周波数に分解する処理です。

写真で撮った自然画などは，このように周波数分解した場合，比較的少数の要素だけに値が集中し，それ以外の要素の値はゼロに近くなる傾向があります。ゼロに近い要素をすべてゼロで近似しても，残った要素だけで元画像に近いものが再現できます。しかもその違いは人間の目ではほとんど分かりません。

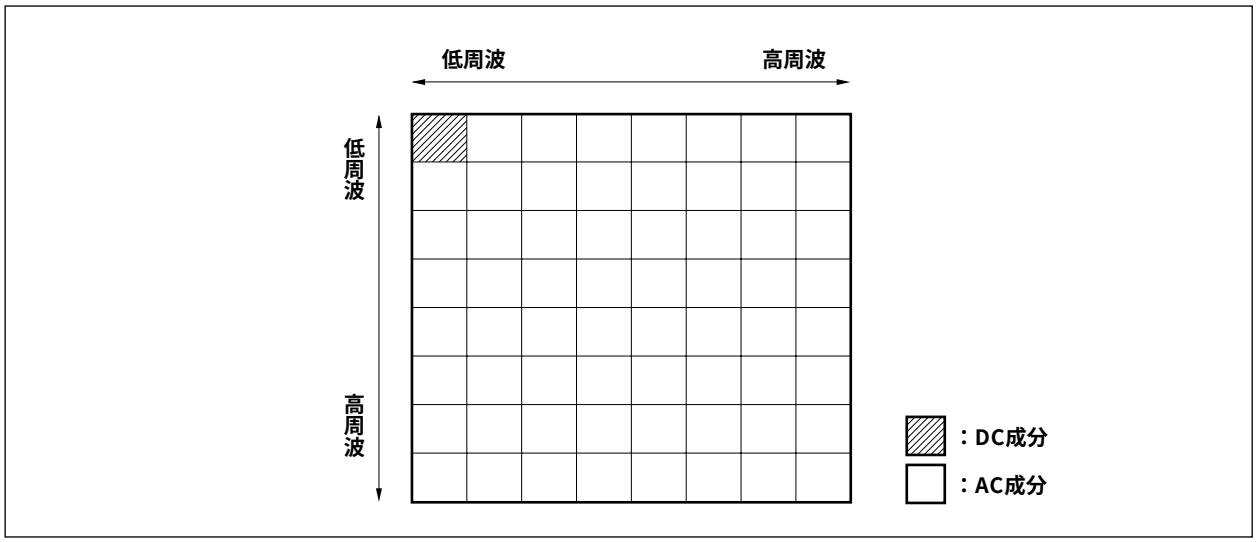
８×８ピクセルの画像データをDCT変換した結果の64要素をDCT係数と呼びます。先頭の１要素はマトリクス全体の平均の色レベルを表し，それ以外の63要素はマトリスク内の色のひずみ具合を表します。そこで信号処理では，DCT変換後のマトリクスの最初の１要素とそれ以外の63要素との特徴の違いから，最初の１要素だけをDC成分（直流成分）と呼び，残りの63要素をAC成分（交流成分）と呼んでいます。

図１－６　マトリクスの成分



DCT変換後の８×８マトリクスは，左側と上側に低周波成分が集まり，右側と下側に高周波成分が集まります。元の画像が単色に近いような変化のないものだった場合，低周波成分だけのマトリクス（高周波の値がほとんどすべて０）が得られます。逆に，市松模様や変化に富んだ細かい画像の場合には，高周波の部分の値がいくつか突出しているようなものが得られます。

図１－７　周波数成分の分布



保守／廃止

（5）量子化とジグザグ・スキャン

人間の目は高周波の成分が多少変わっても変化がよく分からない反面，低周波の成分が変わるとほんの少しの変化でも認識できると言われています。JPEGでは，少しでも圧縮率を上げるために低周波成分は小さな値で割っておいて，高周波の成分は大きな値で割るという処理を加えており，これを量子化と呼んでいます。圧縮したデータを伸長するには割った値と同じ値を掛ければよいのですが（逆量子化），量子化／逆量子化の結果，データが完全に元に戻るわけではありません（不可逆）。これは，量子化の時点で，割った商だけを情報として残し，余りは情報としては無視するためです。ここでも人間の目で見て分からないように圧縮率を上げようとしています。

図１－８　量子化行列と量子化

量子化行列の例

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

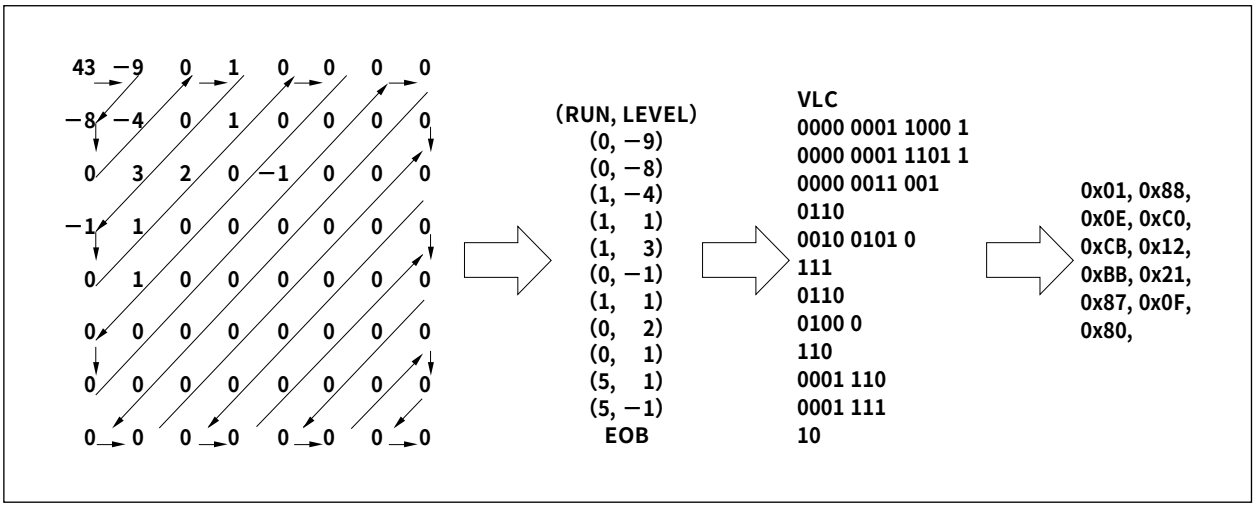
量子化された８×８行列の例

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

元画像のブロックをDCT変換して得られるデータは，Ｙ成分のものとCb/Cr成分のものでは異なった特徴があります。そこでJPEGでは，量子化行列にＹ成分用とCb/Cr成分用の２種類を用いるのが普通です（１種類で済ませる場合もあります）。また，この量子化行列は画像（JPEGファイル）ごとに独立に定義できます。この量子化行列の情報は，JPEGファイルのヘッダにDQTセグメントとして格納されます。

図１－８の例のように，得られたマトリクスのほとんどの値が０ならば，「何個のゼロが連続し，そのあとにゼロではない値が続いている」という情報の解釈をして圧縮率を高くします。JPEGではこの「連続するゼロの個数」を「ゼロランの長さ」と呼んでいます。量子化の結果得られるマトリクスのゼロでない値は，たいてい左上方に集まります。そこでJPEGでは，次の図のような順序（ジグザグ・スキャン）でゼロランの長さを数えます。

図1－9　ジグザグ・スキャンとコード化



（6）エントロピ符号化

一般のJPEGでは，ハフマンを応用したエントロピ符号化を行います。エントロピ符号化の段階では，DC成分とAC成分ではその数字の絶対値や分布が異なります。

AC成分の絶対値が比較的小さいのに比べ，DC成分の絶対値は大きくなりがちです。これはDC成分がそのブロックの平均値であるためです。JPEGではDC成分を圧縮する際に，Y 成分，Cb成分，Cr成分ごとに1 つ手前のブロックのDC成分との差分を求め，その値をエントロピ圧縮します。AC成分については，ゼロランの長さとゼロでない係数の値（LEVEL値）の組み合わせをエントロピ圧縮します。圧縮されたコードはVLC（Variable Length Code）と呼ばれます。

JPEGではDC成分とAC成分とで異なるハフマン符号化規約に基づいて圧縮します。これを“ DCとACでは異なるハフマン・テーブルを用いる ”と表現します。また量子化と同様，Y成分とCb/Cr成分では値の分布が異なるため，通常は別々のハフマン・テーブルを用います。したがって，JPEGではあわせて４種類のハフマン・テーブルを用いることになります。これらのハフマン・テーブルに関する情報はJPEGファイルごとに定義でき，JPEGファイルのヘッダにDHTセグメントとして格納されます。

ある値をエントロピ符号化する場合，絶対値がｎビットの値はｎビットの情報しか持ちません。つまり，絶対値がｎビットの値はｎビットで表すことができます。信号処理では，一般的には値を次のように定義します。

ｎビットの正の数：値の下位ｎビット
ｎビットの負の数：値を符号反転したものの下位ｎビット

JPEGではこれに基づいてエントロピ符号化を実行します。

表１－３　DC/AC成分の値とビット長

成分の値	カテゴリ
0	0
－1, 1	1
－3, －2, 2, 3	2
－7～－4, 4～7	3
－15～－8, 8～15	4
－31～－16, 16－31	5
－63～－32, 32～63	6
－127～－64, 64～127	7
－255～－128, 128～255	8
－511～－256, 256～511	9
－1023～－512, 512～1023	10
－2047～－1024,1024～2047	11

JPEGでは，このカテゴリの値をエントロピ圧縮します。

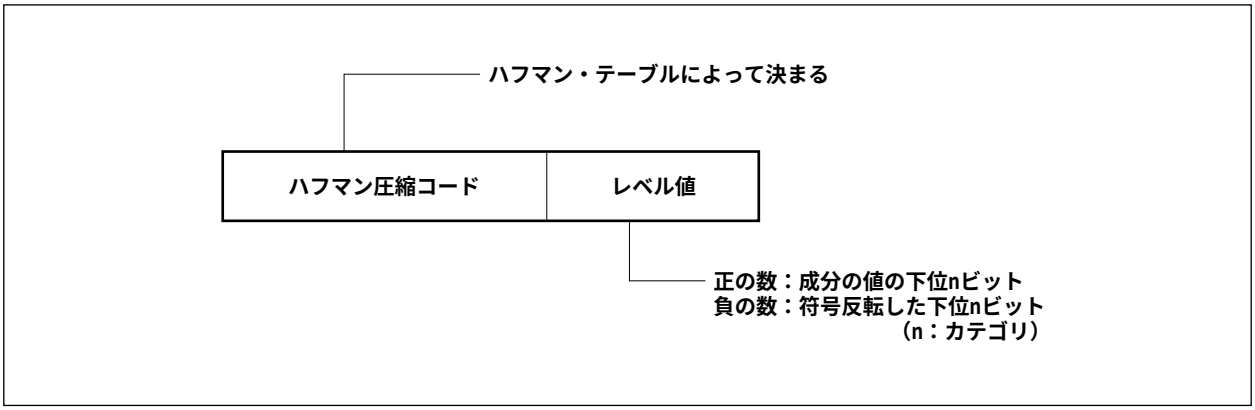
たとえば，輝度（Ｙ）のDC成分用のハフマン・テーブルが，次のような規約になっていたとします。

０ビット長の値には，ハフマン圧縮コード00（２ビット）を割り当てる
１ビット長の値には，ハフマン圧縮コード010（３ビット）を割り当てる
２ビット長の値には，ハフマン圧縮コード011（３ビット）を割り当てる
３ビット長の値には，ハフマン圧縮コード100（３ビット）を割り当てる
４ビット長の値には，ハフマン圧縮コード001（３ビット）を割り当てる
⋮

Ｙ成分のブロックのDC成分の差分値（１つ手前のＹ成分のブロックのDC成分の値との差）が“－３”であったとすると，“－３”はカテゴリ２に属するので，次のようにエントロピ符号化されます。

カテゴリ２のハフマン圧縮コード011（３ビット）
“－３”の符号反転の下位２ビット00
３＋２＝５ビット

図１－10　ハフマン符号化

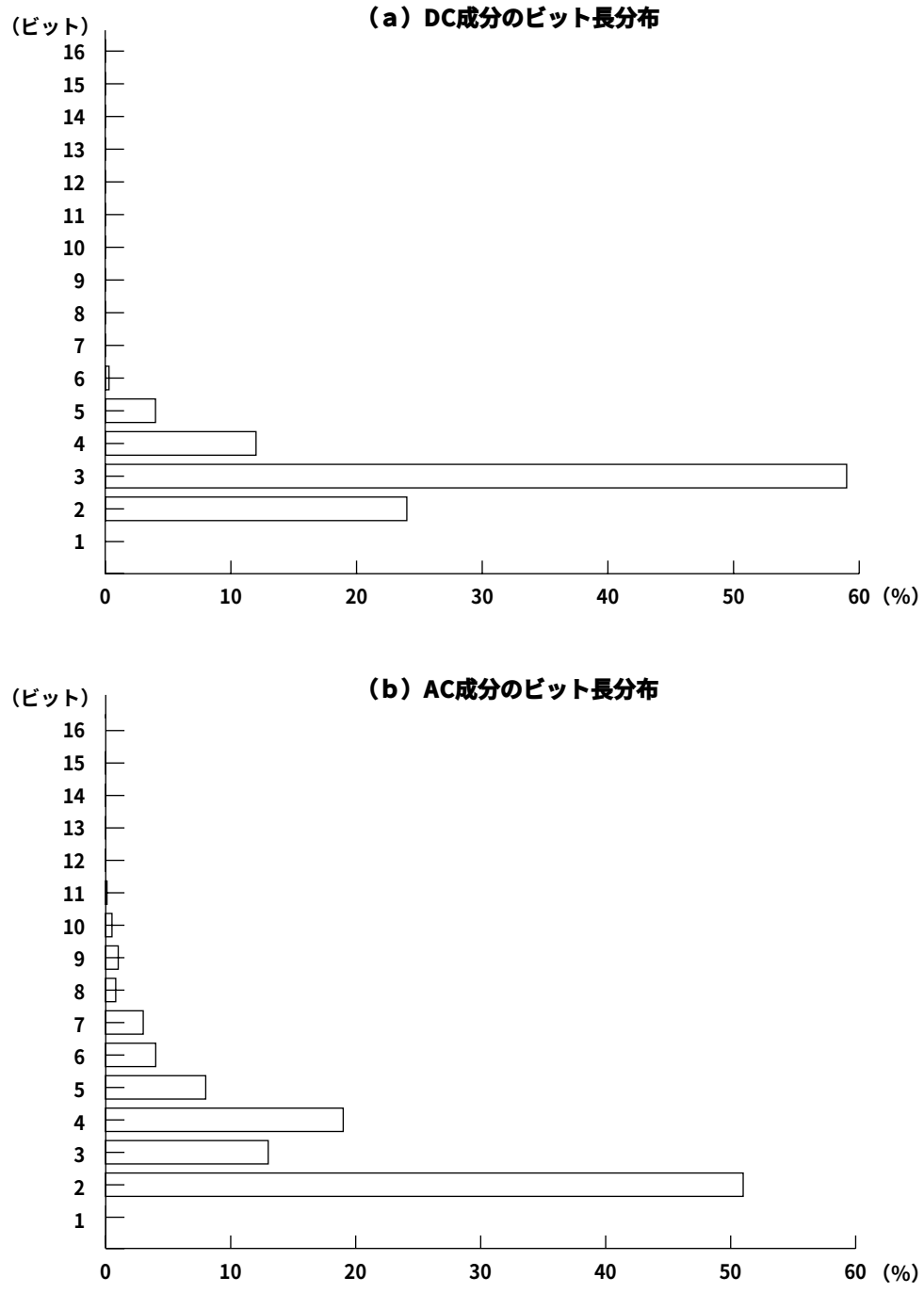


AC成分の場合は，ハフマン・テーブルが次のような規約になります。

ゼロランが０の１ビット長の値には，圧縮コード00（２ビット）を割り当てる
ゼロランが０の２ビット長の値には，圧縮コード01（２ビット）を割り当てる
ゼロランが０の３ビット長の値には，圧縮コード100（３ビット）を割り当てる
ゼロランが０の４ビット長の値には，圧縮コード1010（４ビット）を割り当てる
ゼロランが１の１ビット長の値には，圧縮コード1011（４ビット）を割り当てる
ゼロランが０の５ビット長の値には，圧縮コード1100（４ビット）を割り当てる
ゼロランが１の２ビット長の値には，圧縮コード11010（５ビット）を割り当てる
⋮

保守／廃止

図１－11　DC/AC成分のビット長分布例



保守／廃止

（7）リスタート・マーカ

JPEGでは，MCUを圧縮したコードの途中に，目印となる２バイト（リスタート・マーカ）を挿入します。

リスタート・マーカは，JPEGの画像を絵の途中から下だけ伸長したい場合などに使用できます。また，JPEGファイルの転送途中でビット誤りが生じた場合，そのJPEGファイルがリスタート・マーカを使用していれば，次のリスタート・マーカが見つかった位置から伸長を正しく再開できることがあります。リスタート・マーカを使用しないJPEGファイルでは，それ以降のデータは正しく伸長できません。

図１－12　JPEGファイルにビット誤りがあって正しく伸長できない例

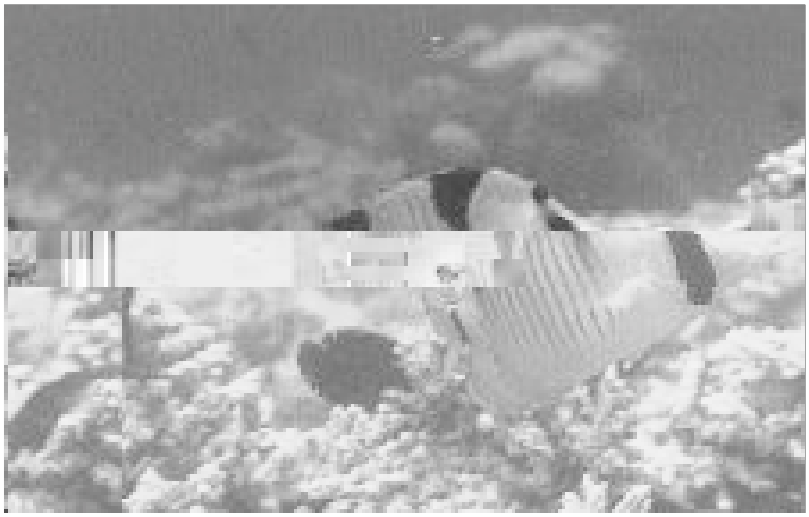
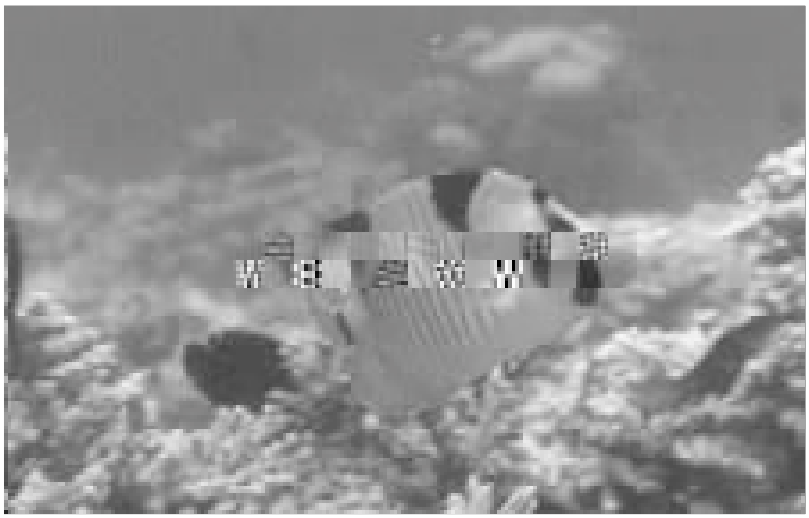


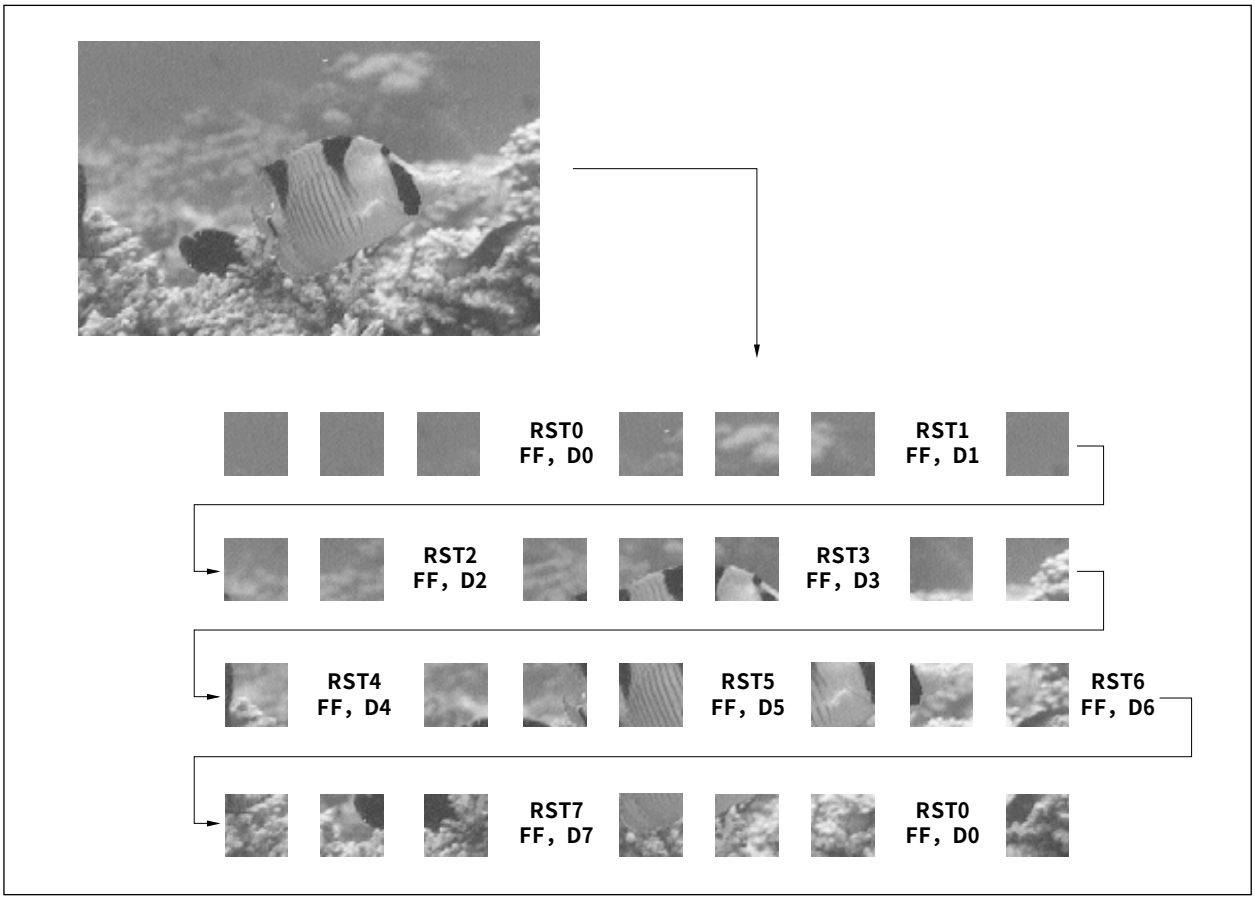
図１－13　リスタート・マーカを用いて途中から正しく伸長を再開できた例



リスタート・マーカは全部で８種類あり，値は0xFF, 0xD0から0xFF, 0xD7までです。リスタート・マーカはm個のMCUおきに圧縮コード中に挿入することになっており，RST0,RST1,RST2, ...,RST7の順に使用します。RST7の次はRST0に戻ります。このmの値をリスタート・インターバルと呼んでいます。

たとえば，リスタート・インターバルが３の場合は，次の図のようなイメージになります。

図１－14　リスタート・マーカ



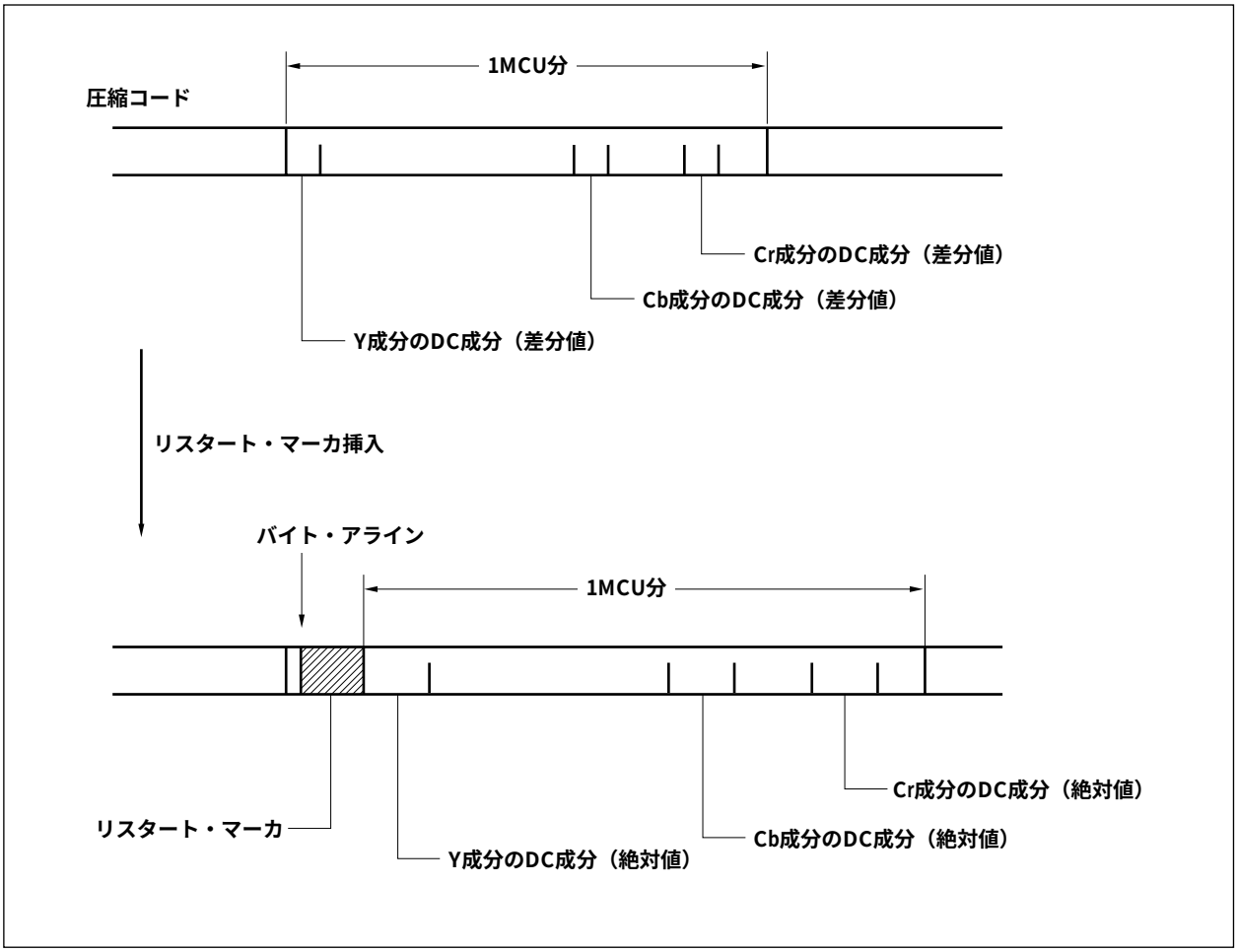
リスタート・マーカが挿入される個数は，画像のサイズから求められます。たとえば，640×480ピクセルの画像に対して，サンプル比が２：１：１でリスタート・インターバルが２の場合には，次のように計算します。

MCU（最小圧縮単位）：16×８ピクセル
リスタート・マーカ：MCU２個おき
したがって，
（640×480）÷（16×８）÷2＝1200個

リスタート・マーカはバイト境界に配置します。一方，圧縮コードはビット単位で配置します。したがって，リスタート・マーカを１個挿入すると，マーカ自身の２バイトよりもデータ量が増えます。リスタート・マーカ１個あたりのバイト数は，多少のばらつきはあるものの，ほぼ４バイト弱となります。また，リスタート・マーカ直後のDC成分は，１つ手前のDC成分との差分値ではなく，そのままの値を圧縮することになります。

たとえば，640×480ピクセルの画像に対して，サンプル比が２：１：１でリスタート・インターバルが２の場合には，リスタート・マーカなしの場合に比べて，ファイル・サイズが約4800バイト（1200個×約４バイト）増えます。

図１－15　リスタート・マーカによるファイル・サイズの増加



保守／廃止

（8）APPnマーカ

JPEGでは，JPEGの圧縮／伸長に直接関係のないデータを，JPEGファイルのヘッダ中に埋め込んだり取り出したりできるように，アプリケーション・データ・セグメント（APPnセグメント）を用意しています。

APPnセグメントは16種類あり，内容はユーザが定義できます。

図１－16　APPnセグメントの構造

APPnマーカ （２バイト）	データの長さ （２バイト）	データ （1-65533バイト）

APPnマーカは0xFF,0xE0から0xFF,0xEFまでの16種類で，各APPnセグメントに対応しています。

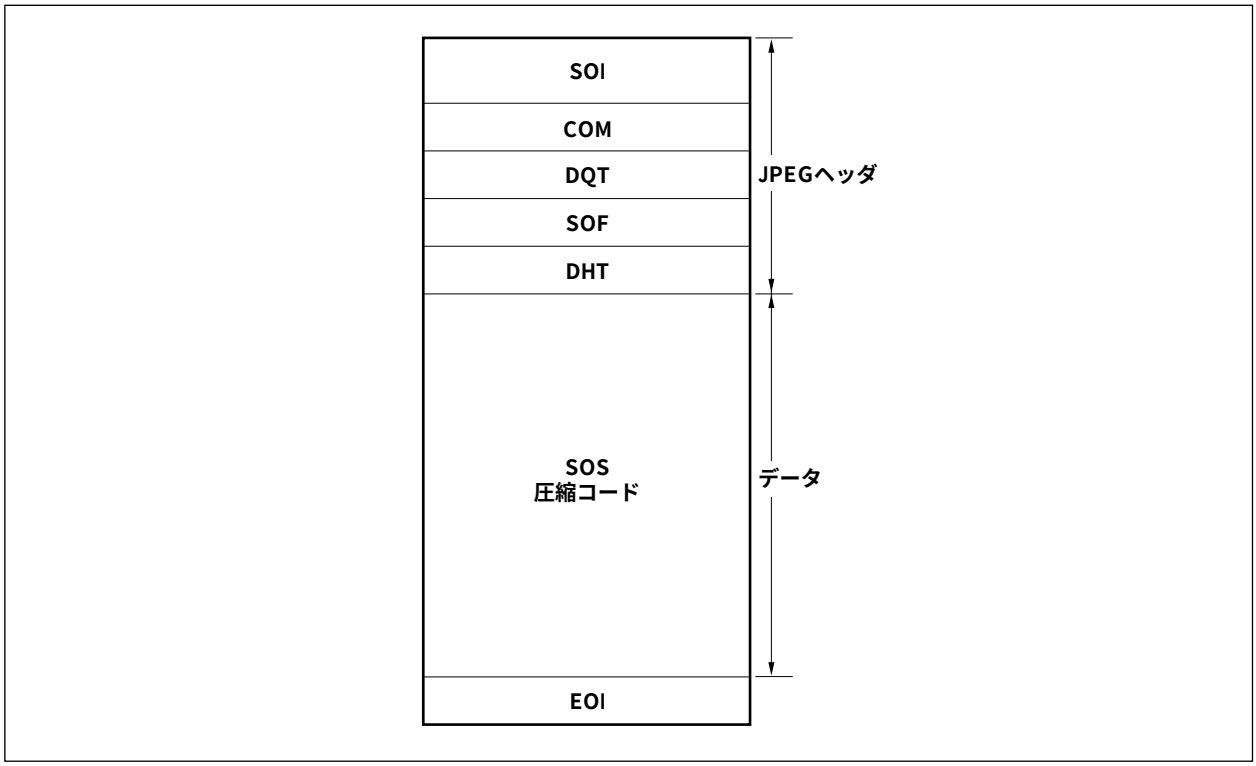
AP705100-B03, AP70732-B03では，圧縮時にAPPnセグメントを使用するかどうかを選択できます。使用する場合は，APPnマーカを選択することで，どのセグメントを使用するかを指定します。また，JPEGファイル中のAPPnセグメントの位置を検出する解析ルーチンも用意しています。

保守／廃止

1.2.2　JPEGのファイル・フォーマット

JPEGファイルは，ファイルを伸長するために必要ないくつかの情報を含んだヘッダ部分と，画像そのものをDCT変換，量子化，エントロピ圧縮して得られたデータ部分から構成されます。ヘッダ部分のデータはすべてバイト単位です（一部，情報を解釈する段階で１バイトを「４ビット＋４ビット」として処理する必要があります）。データ部分はビット単位です。データ部分全体としては，バイト境界に収まるようになっています。

図１－17　JPEGファイル・フォーマット



（1）ヘッダ

JPEGでは，いろいろなテーブルを「マーカ」から始まる「セグメント」という単位で管理しています。マーカは，必ず0xFFと，それぞれのマーカ固有の１バイトを組み合わせた２バイトから始まります。したがって，JPEGファイルを0xFFでサーチすると，その中で使用されているすべてのマーカを検出できます。ただし，0xFFはヘッダ部分だけではなく，圧縮データ中にもあります。このため，圧縮データとしての0xFFには，直後に圧縮データとしては意味のない0x00を挿入することになっています。「0xFF, 0x00」はマーカではなく，圧縮されたデータ「0xFF」のことです。

JPEGヘッダの各セグメント（COM, DQT, SOF, DHTなど）の順序に決まりはありません。

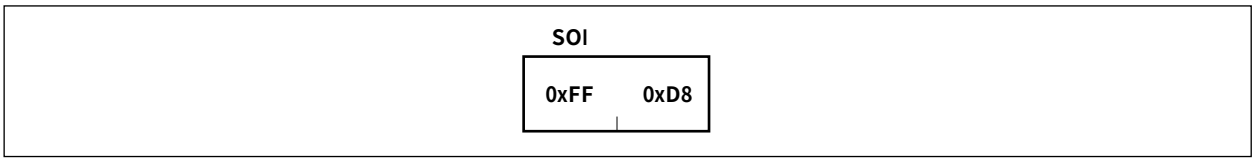
次にJPEGのマーカを示します。

表1－4　JPEGマーカ

値	内　　容
0xFF 0x00	非マーカ（圧縮データの0xFF）
0xFF 0x01	TEM（算術圧縮の場合のテンポラリ・マーカ）
0xFF 0x02-0xFF 0xBF	RES（予約）
0xFF 0xC0	SOF0マーカ（Baseline DCT（Huffman））
0xFF 0xC1	SOF1マーカ（Extended sequential DCT（Huffman））
0xFF 0xC2	SOF2マーカ（Progressive DCT（Huffman））
0xFF 0xC3	SOF3マーカ（Spatial（sequential）lossless（Huffman））
0xFF 0xC4	DHTマーカ（ハフマン・テーブル定義セグメント）
0xFF 0xC5	SOF5マーカ（Differential sequential DCT（Huffman））
0xFF 0xC6	SOF6マーカ（Differential progressive DCT（Huffman））
0xFF 0xC7	SOF7マーカ（Differential spatial（Huffman））
0xFF 0xC8	JPGマーカ（JPEG拡張用に予約）
0xFF 0xC9	SOF9マーカ（Extended sequential DCT（arithmetic））
0xFF 0xCA	SOF10マーカ（Progressive DCT（arithmetic））
0xFF 0xCB	SOF11マーカ（Spatial（sequential）lossless（arithmetic））
0xFF 0xCC	DACマーカ（算術コーディングのための環境設定セグメント）
0xFF 0xCD	SOF12マーカ（Differential sequential DCT（arithmetic））
0xFF 0xCE	SOF13マーカ（Differential progressive DCT（arithmetic））
0xFF 0xCF	SOF14マーカ（Differential spatial（arithmetic））
0xFF 0xD0-0xFF 0xD7	RSTnマーカ（リスタート・マーカ）
0xFF 0xD8	SOIマーカ（JPEGファイルの先頭）
0xFF 0xD9	EOIマーカ（JPEGファイルの末尾）
0xFF 0xDA	SOSマーカ（圧縮データの先頭）
0xFF 0xDB	DQTマーカ（量子化テーブル定義）
0xFF 0xDC	DNLマーカ（ライン数定義）
0xFF 0xDD	DRIマーカ（リスタート・インターバルの定義）
0xFF 0xDE	DHPマーカ（ハフマン・テーブルの定義）
0xFF 0xDF	EXPマーカ（エクスパンド・セグメント）
0xFF 0xE0-0xFF 0xEF	APPnマーカ（ユーザ・アプリケーション用に予約）
0xFF 0xF0-0xFF 0xFD	JPGnマーカ（JPEG拡張用に予約）
0xFF 0xFE	COMマーカ（コメント）

（a）SOI（Start of image）マーカ

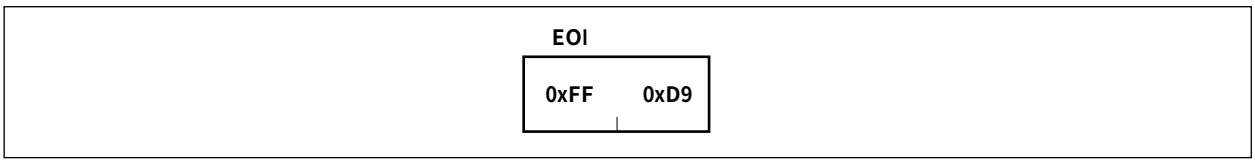
図１－18　SOIマーカ



JPEGファイルの開始を表すマーカです。JPEGファイルは必ずこの２バイトから始まります。

（b）EOI（End of image）マーカ

図１－19　EOIマーカ

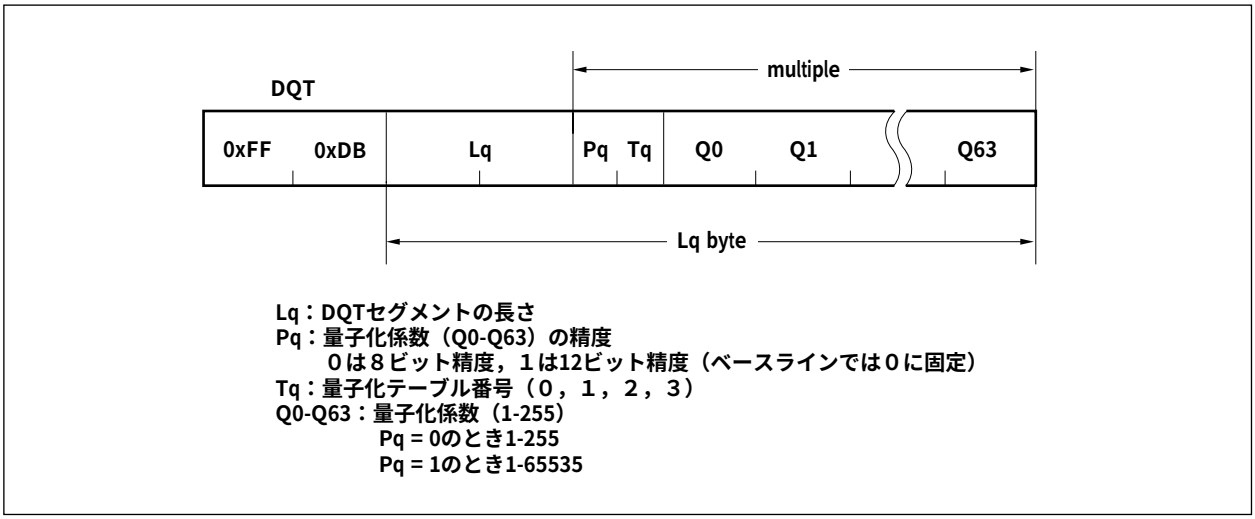


JPEGファイルの終了を表すマーカです。JPEGファイルは必ずこの２バイトで終わります。

（c）DQT（Define quantization table（ｓ））マーカ

量子化テーブルを定義します。

図１－20　DQTセグメント



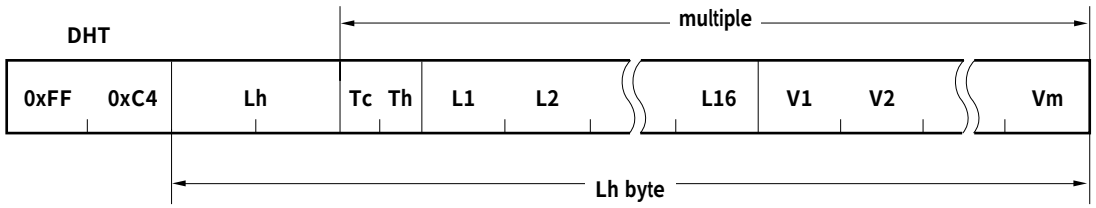
通常輝度成分用（Luminance quantization table）と色差成分用（Chrominance quantization table）の２つを持ちます。

保守／廃止

（d）DHT（Define Huffman table（s））マーカ

ハフマン・テーブルを定義します。

図1－21　DHTセグメント



Lh：DHTセグメントの長さ
Tc：0 = DC, 1 = AC
Th：ハフマン・テーブル番号
　　ベースラインでは0または1
　　拡張ベースラインおよびプログレッシブでは0，1，2，3
L1-L16：iビット長のハフマン・コードが何個あるかを表す
V1-Vm：対応するハフマン・コード

例

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

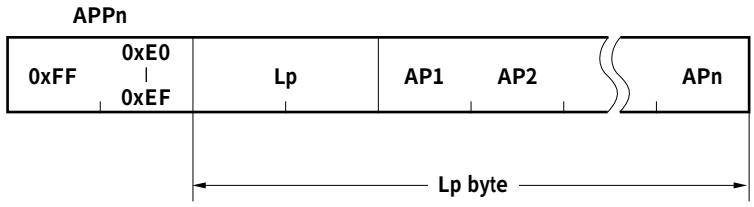
L1-L16が上記のような値の場合，次のような意味になります。

1ビット長のコードが，0個
2ビット長のコードが，00の1個
3ビット長のコードが，010,011,100,101,110の5個
4ビット長のコードが，1110の1個
5ビット長のコードが，11110の1個
6ビット長のコードが，111110の1個
7ビット長のコードが，1111110の1個
8ビット長のコードが，11111110の1個
9ビット長のコードが，111111110の1個
それ以外のコード長はなし

V1-Vmは対応するハフマン・コードです。たとえば，圧縮コード‘010’に対応するハフマン・コードは（‘010’がこの場合，2番目の圧縮コードなので）V2です。

（e）APPn（Reserved for application segments）マーカ

図１－22　APPnセグメント



Lp：APPnセグメントの長さ
AP1-APn：アプリケーション固有のデータ

アプリケーション・データ・セグメントは，それぞれのアプリケーションが自由に使うことを許されているセグメントです。たいていはJPEGファイルを作ったアプリケーションのバージョンなどが入っています。場合によっては小さなサイズのJPEGファイルがそのまま入っていることもあります。このセグメントはLpの値だけを見てスキップすることができます。

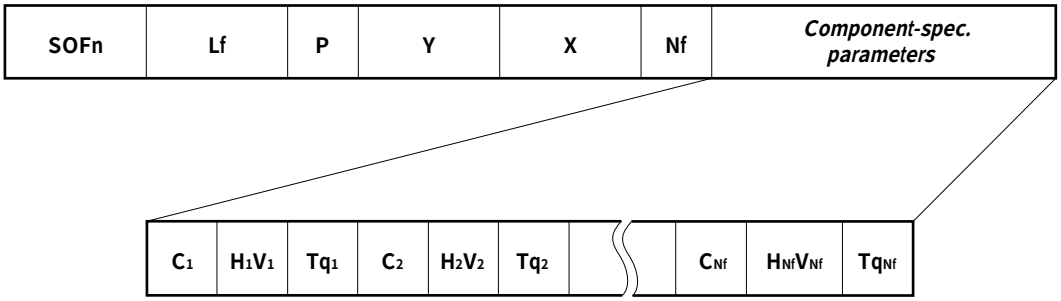
保守／廃止

★　（ｆ）SOFn（Start of frame）マーカ

JPEGでは，JPEGファイルからSOIマーカとEOIマーカを除いた部分をフレームと呼んでいます。SOFnセグメントは特にフレーム・ヘッダとも呼ばれ，伸長に必要な量子化テーブル番号などを規定しています。

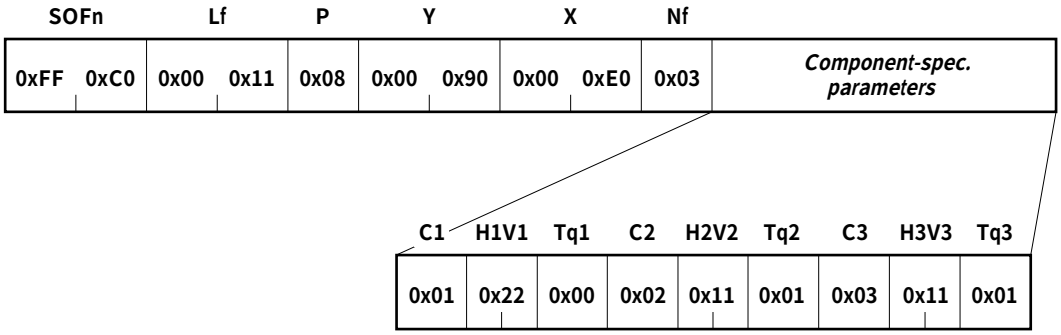
なお，JPEGではY/Cb/Crなどの色要素をコンポーネントと呼んでいます。

図１－23　SOFnセグメント



SOFn	SOF0	0xFF 0xC0：ベースライン
	SOF1	0xFF 0xC1：拡張ベースライン
	SOF2	0xFF 0xC2：プログレッシブ
Lf		SOFnセグメントの長さ（８＋３×Nfバイト）
P		DCT係数の精度　８：８ビット精度 12：12ビット精度
Y		画像の縦サイズ（ピクセル数）
X		画像の横サイズ（ピクセル数）
Nf		コンポーネントの数　１：モノクロ（単色） ３：カラー（３色） ４：CMYK, YCCK（４色）
Ci（0≦Ci≦255）		色コンポーネント番号定義（ｉ番目のコンポーネントの番号）
Hi（1≦Hi≦4）		水平サンプリング・ファクタ（ｉ番目のコンポーネントの横方向サンプリング比）
Vi（1≦Vi≦4）		垂直サンプリング・ファクタ（ｉ番目のコンポーネントの縦方向サンプリング比）
Tqi（0≦Tqi≦3）		量子化テーブル番号（ｉ番目のコンポーネントに使われる量子化テーブル番号）

例　SOFnセグメントの内容が次のような場合を考えてみます。



このとき，Nf, Ci, Hi, Viの部分の意味は次のようになり，サンプル比４：１：１（H：V＝２：２）であることを示しています。

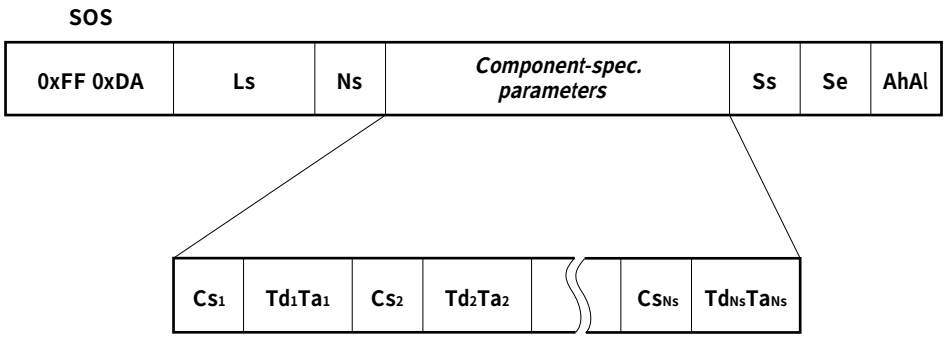
コンポーネント数（Nf）は３（YCbCr），
Y成分の色コンポーネント番号（C1）が１で，サンプリング比（H1V1）は２×２で４ブロック，
Cb成分の色コンポーネント番号（C2）が２で，サンプリング比（H2V2）は１×１で１ブロック，
Cr成分の色コンポーネント番号（C3）が３で，サンプリング比（H3V3）は１×１で１ブロック

保守／廃止

★ （g）SOS（Start of scan）マーカ

JPEGファイルの圧縮データ部分はSOSセグメントから始まる「スキャン」と呼ばれる単位で分割されています。このため，SOSセグメントはスキャン・ヘッダとも呼ばれます。スキャン・ヘッダの直後から圧縮された画像のデータが始まります。スキャン・ヘッダは圧縮データのハフマン・テーブル番号の指定などを行います。

図１－24　SOSセグメント



SOS	0xFF 0xDA
Ls	SOSセグメントの長さ（６＋２×Nsバイト）
Ns	そのスキャンに含まれる色の数 ３：Yブロック，Cbブロック，Crブロック １：Y, Cb, Crのどれか
Csi	ｉ番目のコンポーネント番号（SOFで定義した番号）
Tdi	ｉ番目のコンポーネントのDC成分のハフマン・テーブル番号 ベースライン：０または１ 拡張ベースラインおよびプログレッシブ：０，１，２，３
Tai	ｉ番目のコンポーネントのAC成分のハフマン・テーブル番号 ベースライン：０または１ 拡張ベースラインおよびプログレッシブ：０，１，２，３
Ss	DCT係数の開始番号 ベースラインおよび拡張ベースライン：０に固定 プログレッシブ：0-63
Se	DCT係数の終了番号 ベースラインおよび拡張ベースライン：63に固定 プログレッシブ：Ss-63（ただしSs = 0とのきはSe = 0に固定）
Ah	DCT係数の上位ビット ベースライン，拡張ベースライン：０に固定 プログレッシブ：0-13
Al	DCT係数の下位ビット ベースライン，拡張ベースライン：０に固定 プログレッシブ：0-13

（h）DRI（Define restart interval）マーカとRSTn（Restart interval termination）マーカ

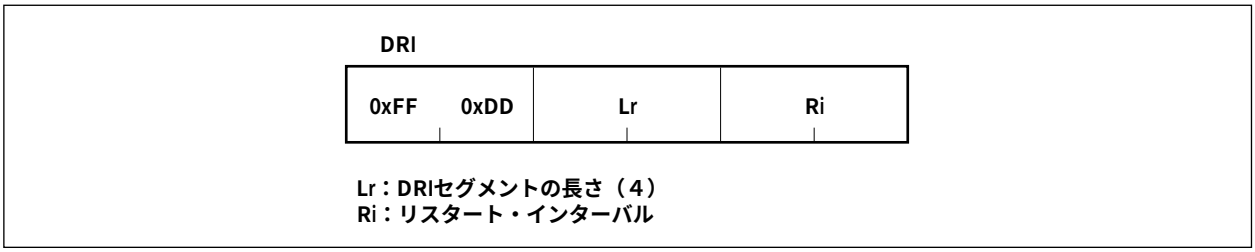
リスタート・マーカは，通信データ誤りなどの不正データによる影響を最小限に抑えるためのものです。DRIマーカで設定されたMCUの個数ごとにリスタート・マーカを挿入します。

たとえば，４個おきにマーカを入れるとした場合には，次のようにRST0, RST1,...,RST7を順番に入れていきます。

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

JPEGでは，DC成分は差分情報なので，それぞれの８×８ブロックを伸長するために１つ手前のDC成分の値が必要ですが，リスタート・マーカの直後のDC成分は０との差分になります。これにより，たまたまMCU4でデータが壊れてしまっても，MCU5以降を正確に伸長できます。

図１－25　DRIセグメント



保守／廃止

1.3　システム概要

1.3.1　ライブラリ構成

JPEGミドルウエア・ライブラリは，次に示すライブラリで構成されています。

表１－５　製品のライブラリ構成

製　　品	ライブラリ構成	対応ファイル・フォーマット
AP70732-B03 （V810ファミリ対応）	基本ライブラリ（圧縮）	ベースライン
	基本ライブラリ（伸長）	
AP705100-B03 （V830ファミリ対応）	基本ライブラリ（圧縮）	ベースライン
	基本ライブラリ（伸長）	
	追加ライブラリ（伸長）	プログレッシブ ベースライン 拡張ベースライン

AP705100-B03には，基本ライブラリと追加ライブラリがあります。追加ライブラリは単独でも，基本ライブラリとの共存でも使用可能です。基本ライブラリは高速処理，小メモリ・タイプ，追加ライブラリは機能重視，多様フォーマット対応タイプとして位置づけられます。

なお，このユーザーズ・マニュアルでは基本ライブラリの伸長処理を基本伸長，追加ライブラリの伸長処理を追加伸長と呼びます。

1.3.2　基本／追加ライブラリの特徴

基本ライブラリ／追加ライブラリに共通する特徴を示します。

（1）VRAMと座標（x, y）

VRAMアクセスはハードウエアに依存する部分のため，ライブラリとは切り分けてあります。この部分に関してはそれぞれのシステムに合わせて記述する必要があります（Ｃ言語での記述が可能です）。

ただし，LD.B命令またはST.B命令でアクセスできるVRAMに関してはディフォールトでルーチンを用意しています。

YCbCrとRGBのそれぞれに対してVRAM仕様を想定し，画像をVRAMの任意の位置に伸長，任意の位置から切り取って圧縮できます。

（2）量子化テーブル

設定できる量子化テーブルの２面を指定します。

圧縮では，量子化テーブルをディフォールトで用意していますが，ユーザが定義して設定することもできます。また，量子化パラメータ（Quality）を用意しています。このパラメータに0-100の値を設定することにより，量子化テーブルの値が一律に定数倍され，要素の値が1-255の範囲におさまるように加工されます（量子化テーブルを加工しないで用いる場合は，量子化パラメータに値50を設定してください）。

伸長では，DQTヘッダに書かれている値を使用します。

保守／廃止

（3）ハフマン・テーブル

設定できるハフマン・テーブルの４面を指定します。

圧縮では，ハフマン・テーブルをディフォールトで用意していますが，ユーザが定義して設定することもできます。

伸長では，DHTヘッダに書かれている値を使用します。

（4）リスタート・マーカ

圧縮では，リスタート・マーカを使用する／使用しないの選択ができます。使用する場合は，リスタート・インターバルは変更できます。

伸長では，DRIヘッダの値を使用します。

（5）APPnセグメント

圧縮では，APPnセグメントを挿入するように指定することができます。

伸長では，APPnセグメントは無視されますが，APPnセグメントの位置を検出します。

（6）OS対応

圧縮，解析，伸長すべてリエントラント可能です。ルーチンを実行するためにはそれぞれに指定した構造体が必要です。

保守／廃止

1.3.3　基本ライブラリの特徴

基本ライブラリの特徴を示します。

（1）サンプル比

次の４つのサンプル比をサポートします。

- 4：1：1［H：V = 2：2］（画像のサイズは縦横とも16の倍数のみ）
- 4：1：1［H：V = 4：1］（画像のサイズは横は32の倍数，縦は８の倍数）
- 2：1：1［H：V = 2：1］（画像のサイズは横は16の倍数，縦は８の倍数）
- 1：1：1［H：V = 1：1］（画像のサイズは縦横とも８の倍数のみ）

（2）JPEGファイル格納用バッファ

ルーチン実行のためにJPEGファイルを格納するバッファが必要です。JPEGファイルのサイズは一定でないため，JPEGバッファの最後までデータがきた場合に処理を中断し，バッファ処理（圧縮ではバッファの退避，伸長ではバッファの更新）後ルーチンを再度呼び出すことで処理が再開されるようになっています。

JPEGファイル用のバッファのサイズは１バイト以上で任意に設定できます。ただし，圧縮，伸長処理とバッファ処理の間には必ずレジスタ・ディスパッチ処理が入るため，この回数があまり増えないように十分なサイズを確保してください。

（3）縮小伸長

普通の大きさの伸長（JPEGファイル・ヘッダに書かれているピクセル数）のほかに，面積４分の１，16分の１，64分の１に縮小して伸長モードを選択することができます（縮小モードでは，逆DCT変換そのものを縮小用に設計してあるので普通サイズの伸長に比べて高速に伸長できます）。

（4）伸長のクリッピング（MCU単位）

JPEGファイルを伸長する際にMCU単位で矩形をクリッピングしてその部分だけを伸長することができます。

（5）ライン処理

画像メモリが１枚の画像すべてを保持できないようなシステムのために1MCUの縦ピクセル数分のライン処理ごとに処理を中断する選択ができます。

（6）内蔵RAM（AP705100-B03の場合）

内蔵命令RAMは使用しません。

内蔵データRAMを1024バイト使用します。

十分な性能を得るために圧縮／伸長それぞれ１タスクあたり1024バイトを必要とします。

保守／廃止

★ 1.3.4 追加ライブラリの特徴（AP705100-B03）

（1）３つのファイル・フォーマットに対応

次の３つのファイル・フォーマットの伸長が可能です。

・プログレッシブ・ファイル・フォーマット
・ベースライン・ファイル・フォーマット
・拡張ベースライン・ファイル・フォーマット

（2）サンプル比

任意のサンプル比に対応しています。

（3）ノンインタリーブ・フォーマット対応

JPEGファイルの圧縮データ部分はSOSセグメントで始まるスキャンという単位で分割されています。

通常のベースライン・フォーマットはファイル内に１つのスキャンのみを持つインタリーブ・フォーマット（シングルスキャン・フォーマット）ですが，追加ライブラリではプログレッシブ・フォーマットのほか，複数のスキャンをもつノンインタリーブ・フォーマット（マルチスキャン・フォーマット）にも対応しています。

（4）色要素

３色フォーマットに加え，単色，４色フォーマットにも対応しています。

（5）DNLマーカ

ライン数を定義するDNLマーカに対応しています。

（6）伸長のクリッピング（ピクセル単位）

ピクセル単位でのクリッピングが可能です。

（7）伸長時の縮小／拡大

伸長時に，n/8（n = 1, 2, 3, …）の比率で縮小／拡大が可能です。

（8）JPEGファイル格納用バッファ

追加ライブラリでは，JPEGバッファが不足した場合でもライブラリは終了せず，ユーザ側で定義されたオーバライト関数（JPEGファイル取得関数）をコールします。

（9）メモリ・サイズ

実行時のライブラリ・オプションの選択方法によっては，従来の基本ライブラリとあまり変わらないROM/RAMサイズ（小メモリ・サイズ）での実行ができます。

（10）ライブラリの中断

ライブラリの実行途中で処理を中断できます。

保守／廃止

（11）ISO/IEC 10918-2適合性検査に対応

AP705100-B03追加ライブラリは，ISO/IEC 10918-2の画像データの適合性検査を行っており，A, C, E, G, Kの検査データについて正常伸長を確認しています。

★ 1.3.5 基本ライブラリと追加ライブラリの違い（AP705100-B03）

AP705100-B03の基本ライブラリと追加ライブラリの違いを表１－６に示します。

表１－６　基本ライブラリと追加ライブラリの違い

項目＼ライブラリ			基本ライブラリ	追加ライブラリ
ファイル・フォーマット	符号化方式	ベースライン	○	○
		拡張ベースライン	×	○
		プログレッシブ	×	○
		その他	×	×
	符号化順序	インタリーブ	○	○
		ノンインタリーブ	×	○
	DNLマーカ対応		×	○
	サンプル比		４：１：１（H：V＝２：２） ４：１：１（H：V＝４：１） ２：１：１，１：１：１	任意のサンプル比に対応
	色要素		３色	単色，３色，４色
インタフェース・ライブラリ	処理速度		速い	遅い
	JPEGファイル格納用バッファ不足時の対応		ライブラリ終了（再開にはライブラリを再コール）	JPEGバッファ補充関数をコール
	クリッピング		MCU単位	ピクセル単位
	伸長時の縮小／拡大		1/4, 1/16, 1/64の縮小伸長が可能	n/8の比率で，縮小／拡大が可能

備考　○：対応している
×：対応していない

同じベースライン・フォーマット・ファイルの伸長処理をする場合，追加ライブラリは基本ライブラリよりも汎用的に作成してあるため，処理速度は遅くなります。

また，縮小伸長を行う場合，基本ライブラリでは伸長に必要な面積分のピクセルのみを計算するため，等倍伸長より1/4伸長の方が処理速度が速く，1/4伸長より，1/16伸長が速い，という特徴があります。しかし，追加ライブラリでは，一度画像をメモリ上に展開してから拡大／縮小作業を行うため，等倍以外の面積比の伸長は等倍伸長の処理速度よりも遅くなります。

なお，基本ライブラリと追加ライブラリでオプションの設定方法は異なります。オプション設定時には，それぞれのオプションの設定方法を参照してください。

保守／廃止

1.3.6　パッケージ内容

パッケージには，次に示す各ライブラリとサンプル・ソースが含まれています。

（1）AP70732-B03（NEC版）

nectools	lib732	libjegc.a	（圧縮メイン）
		libjcmc1.a	（圧縮mcu制御）（画面単位）
		libjcmc2.a	（圧縮mcu制御）（MCU行単位）
		libjcy.a	（圧縮YCbCr）
		libjcr.a	（圧縮RGB）
		libjpegd.a	（伸長メイン）
		libjdmc1.a	（伸長mcu制御）（画面単位）
		libjdmc2.a	（伸長mcu制御）（MCU行単位）
		libjdy.a	（伸長YCbCr）
		libjdr.a	（伸長RGB）
		libjpeg.a	（共通）
		libjcr2.a	
	smp732 jpeg	start.s	スタート・アップ
		jpeg.h	ヘッダ・ファイル
		main.c	サンプル・メイン
		fish.s	サンプルJPEGファイル
		fish.jpg	fish.sにインクルードされる
		fishtga.s	サンプル画像ファイル
		fish.tga	fishtga.sにインクルード
		tpycc.c	サンプル・ソース
		tprgb.c	サンプル・ソース
		getmcu.c	C言語getmcuサンプル
		putmcu.c	C言語putmcuサンプル
		makeycc	makeファイル
		makergb	makeファイル
		jparc830.exe（98）	
		jparc830（SUN4™）	
		dfile	リンク・ディレクティブ

保守／廃止

（2）AP70732-B03（GHS版）

ghstools	lib810	libjpegc.a	（圧縮メイン）
		libjcmc1.a	（圧縮mcu制御）（画面単位）
		libjcmc2.a	（圧縮mcu制御）（MCU行単位）
		libjcy.a	（圧縮YCbCr）
		libjcr.a	（圧縮RGB）
		libjpegd.a	（伸長メイン）
		libjdmc1.a	（伸長mcu制御）（画面単位）
		libjdmc2.a	（伸長mcu制御）（MCU行単位）
		libjdy.a	（伸長YCbCr）
		libjdr.a	（伸長RGB）
		libjpeg.a	（共通）
		libjcr2.a	
	smp810 — jpeg	start.s	スタート・アップ
		jpeg.h	ヘッダ・ファイル
		main.c	サンプル・メイン
		fish.s	サンプルJPEG
		fishtga.s	サンプル画像データ
		tpycc.c	サンプル・ソース
		tprgb.c	サンプル・ソース
		getmcu.c	C言語getmcuサンプル
		putmcu.c	C言語putmcuサンプル
		makeycc	makeファイル
		makergb	makeファイル
		jparc830.exe（98）	
		jparc830（SUN4）	
		make.lnk	セクション指定

保守／廃止

★ （3）AP705100-B03（NEC版）

nectools	lib830	libjpegc.a	（基圧）圧縮メイン
		libjcmc1.a	（基圧）MCU制御；画面単位
		libjcmc2.a	（基圧）MCU制御；MCU行単位
		libjcy.a	（基圧）getMCU；YCbCr
		libjcr.a	（基圧）getMCU；RGB1
		libjcr2.a	（基圧）getMCU；RGB2
		libjpegd.a	（基伸）伸長メイン
		libjpgdp.a	（基伸）伸長メイン
		libjdmc1.a	（基伸）MCU制御；画面単位
		libjdmc2.a	（基伸）MCU制御；MCU行単位
		libjdy.a	（基伸）putMCU；YCbCr
		libjdr.a	（基伸）putMCU；RGB1
		libjdr2.a	（基伸）putMCU；RGB2
		libjpeg.a	（基）共通
		libjprg.a	（追伸）プログレッシブ対応
		libjprgd.a	（追伸）ディバグ用ライブラリ
		libjprog.a	（追伸）シンボル解決用
	smp830 — jpeg	start.s	スタート・アップ・ルーチン
		dfile	リンク・ディレクティブ
		jparc0	ライブラリ選択ファイル
		jparc1	
		jparc2	
		jparc3	
		jparc4	
		makefile	サンプル・プログラムmakefile
		jpeg.h	ヘッダ・ファイル
		jpegcasm.h	
		jpegdasm.h	
		jpegex.h	
		main0.c	サンプル・ソース・ファイル
		main1.c	
		main2.c	
		main3.c	
		main4.c	
		main5.c	
		main6.c	
		main7.c	
		main8.c	
		getmcu.c	
		putmcu.c	
		tpycc.c	
		tprgb.c	

保守／廃止

smp830 ―― jpeg ―― （nectools/smp830/jpegの続き）

fish.s	
fishprg.s	
fishtga.s	
fish.tga	ベタ画像データ・ファイル
fish.jpg	ベースライン（４：１：１（H：V＝２：２））
fish11.jpg	ベースライン（１：１：１）
fish21.jpg	ベースライン（２：１：１）
fish41.jpg	ベースライン（４：１：１（H：V＝４：１））
fishmono.jpg	モノクロ
fishp3.jpg	プログレッシブ（スペクトラル・セレクション）
fishp4.jpg	プログレッシブ（サクセッシブ・アプロキシメーション）
fishp5.jpg	プログレッシブ（サクセッシブ・アプロキシメーション）
cmykbase.jpg	４色ベースライン
cmykprg3.jpg	４色プログレッシブ（スペクトラル・セレクション）
cmykprg4.jpg	４色（サクセッシブ・アプロキシメーション）
cmykprg5.jpg	４色（サクセッシブ・アプロキシメーション）

保守／廃止

★ （4）AP705100-B03（GHS版）

ghstools	lib830	libjpegc.a	（基圧）圧縮メイン
		libjcmc1.a	（基圧）MCU制御；画面単位
		libjcmc2.a	（基圧）MCU制御；MCU行単位
		libjcy.a	（基圧）getMCU；YCbCr
		libjcr.a	（基圧）getMCU；RGB1
		libjcr2.a	（基圧）getMCU；RGB2
		libjpegd.a	（基伸）伸長メイン
		libjpgdp.a	（基伸）伸長メイン
		libjdmc1.a	（基伸）MCU制御；画面単位
		libjdmc2.a	（基伸）MCU制御；MCU行単位
		libjdy.a	（基伸）putMCU；YCbCr
		libjdr.a	（基伸）putMCU；RGB1
		libjdr2.a	（基伸）putMCU；RGB2
		libjpeg.a	（基）共通
		libjprg.a	（追伸）プログレッシブ対応
		libjprgd.a	（追伸）ディバグ用ライブラリ
		libjprog.a	（追伸）シンボル解決用
	smp830 — jpeg	start.s	スタート・アップ・ルーチン
		make.lnk	セクション指定
		jparc0	ライブラリ選択ファイル
		jparc1	
		jparc2	
		jparc3	
		jparc4	
		makefile	サンプル・プログラムのmakeファイル
		jpeg.h	ヘッダ・ファイル
		jpegcasm.h	
		jpegdasm.h	
		jpegex.h	
		main0.c	サンプル・ソース・ファイル
		main1.c	
		main2.c	
		main3.c	
		main4.c	
		main5.c	
		main6.c	
		main7.c	
		main8.c	
		getmcu.c	
		putmcu.c	
		tpycc.c	
		tprgb.c	

保守／廃止

smp830 ―― jpeg ―― （ghstools/smp830/jpegの続き）

fishtga.s	ベタ画像データ・ファイル
fish.s	ベースライン（４：１：１（H：V＝２：２））
fish11.s	ベースライン（１：１：１）
fish21.s	ベースライン（２：１：１）
fish41.s	ベースライン（４：１：１（H：V＝４：１））
fishmono.s	モノクロ
fishp3.s	プログレッシブ（スペクトラル・セレクション）
fishp4.s	プログレッシブ（サクセッシブ・アプロキシメーション）
fishp5.s	プログレッシブ（サクセッシブ・アプロキシメーション）
cmykbase.s	４色ベースライン
cmykprg3.s	４色プログレッシブ（スペクトラル・セレクション）
cmykprg4.s	４色（サクセッシブ・アプロキシメーション）
cmykprg5.s	４色（サクセッシブ・アプロキシメーション）

保守／廃止

1.3.7 動作環境

（1）対象CPU：V810ファミリ（AP70732-B03）
　　V830ファミリ（AP705100-B03）

（2）使用コンパイラ・パッケージ

●V810ファミリ（AP70732-B03）
NEC製ANSI-Cコンパイラ・パッケージ
CA732（Windows版，Sun4版） Ver.1.00以降
GHS社製コンパイラ・パッケージ
CC800（Windows版，Sun4版） Ver.1.00以降
●V830ファミリ（AP705100-B03）
NEC製ANSI-Cコンパイラ・パッケージ
CA830（Windows版，Sun4版） Ver.1.00以降
GHS社製コンパイラ・パッケージ
CC800（Windows版，Sun4版） Ver.1.00以降

★ （3）メモリ容量

表１－７　ROMサイズ（単位：バイト）

基本ライブラリ：圧縮処理系	約7 K
基本ライブラリ：伸長処理系	約14 K
追加ライブラリ：伸長処理系	約20 K

保守／廃止

表１－８　RAMサイズ（単位：バイト）

処理系		内容	必要バイト数		備考
			AP70732-B03	AP705100-B03	
基本ライブラリ	圧縮処理系	JPEG構造体	1152	128	対応するサンプル比によってはこれ以下（AP70732-B03）
		内蔵RAMワーク・エリア	－	1024	対応するサンプル比によってはこれ以下（AP705100-B03）
		他ワーク・エリア	2688	2688	
		APP構造体	160	160	APPnセグメントを使用する場合だけ必要
		スタック	約144	約128	
		小計	約4144	約4128	
	伸長処理系	JPEG構造体	1152	128	対応するサンプル比によってはこれ以下（AP70732-B03）
		内蔵RAMワーク・エリア	－	1024	対応するサンプル比によってはこれ以下（AP705100-B03）
		他ワーク・エリア	3968	3968	
		APP構造体	160	160	APPnセグメントを使用する場合だけ必要
		スタック	約144	約128	
		小計	約5424	約5408	
追加ライブラリ	伸長処理系２パス設定	ワーク・エリア	－	約5000	内蔵RAMと外部RAMの合計
		スタック	－	約500	
		小計	－	約5500	
	伸長処理系１パス設定	ワーク・エリア	－	約2 M	内蔵RAMと外部RAMの合計。このRAMサイズは，640×480ピクセル，３色の場合。実際には，ピクセル数と色の数に比例した容量が必要。
		スタック	－	約500	
		小計	－	約2 M	

保守／廃止

★ 1.3.8 セクション名，シンボル名規約

（1）セクション名規約

ライブラリが使用するセクションを示します。

表１－９　ライブラリが使用するセクション

分　類	セクション名	タイプ	説　明
基本圧縮処理	.JPCTEXT	.text	テキスト（命令コード）
	.JPCTBL	.rodata	テーブル・データ（定数）
	.JPCDATA	.data	初期値ありデータ
	.JPCBSS	.bss	初期値なしデータ
基本伸長処理	.JPDTEXT	.text	テキスト（命令コード）
	.JPDTBL	.rodata	テーブル・データ（定数）
	.JPDDATA	.data	初期値ありデータ
	.JPDBSS	.bss	初期値なしデータ
基本共通処理	.JPJTEXT	.text	テキスト（命令コード）
	.JPJTBL	.rodata	テーブル・データ（定数）
	.JPJDATA	.data	初期値ありデータ
	.JPJBSS	.bss	初期値なしデータ
追加伸長処理	.JPDTEXT	.text	テキスト（命令コード）
	.JPDDATA	.data	初期値ありデータ

（2）シンボル名規約

JPEGライブラリ内で使用するシンボル名などは，次に示す規約に従って命名されています。ほかのアプリケーションと組み合わせて使用する場合は，重複などのないように注意してください。

なお，追加ライブラリのグローバル・シンボル名は，アセンブラではアンダバーをつけた「_JPEGEX」から始まる文字列になります。

表１－10　シンボル名規約

分類	基本ライブラリ	追加ライブラリ
関数／ラベル名	“ jpeg_ ”で始まる文字列	“ JPEGEX ”で始まる文字列
シンボル名	“ JPEG ”で始まる文字列	
構造体名	CJINFO DJINFO APPINFO	JPEGEXINFO JPEGEXWORK JPEGEXVIDEO JPEGEXBUFF JPEGEXMCUSTR JPEGEXFrmINFO

保守／廃止

1.3.9　サンプル・プログラムのメモリ・マップ

パッケージに含まれているサンプル・プログラムのメモリ・マップを示します。

また，**付録A　サンプル・プログラム・ソース・リスト（AP70732-B03用）**，**付録B　サンプル・プログラム・ソース・リスト（AP705100-B03用）**を参照してください。

図１－26　サンプル・プログラムのメモリ・マップ（AP70732-B03用）

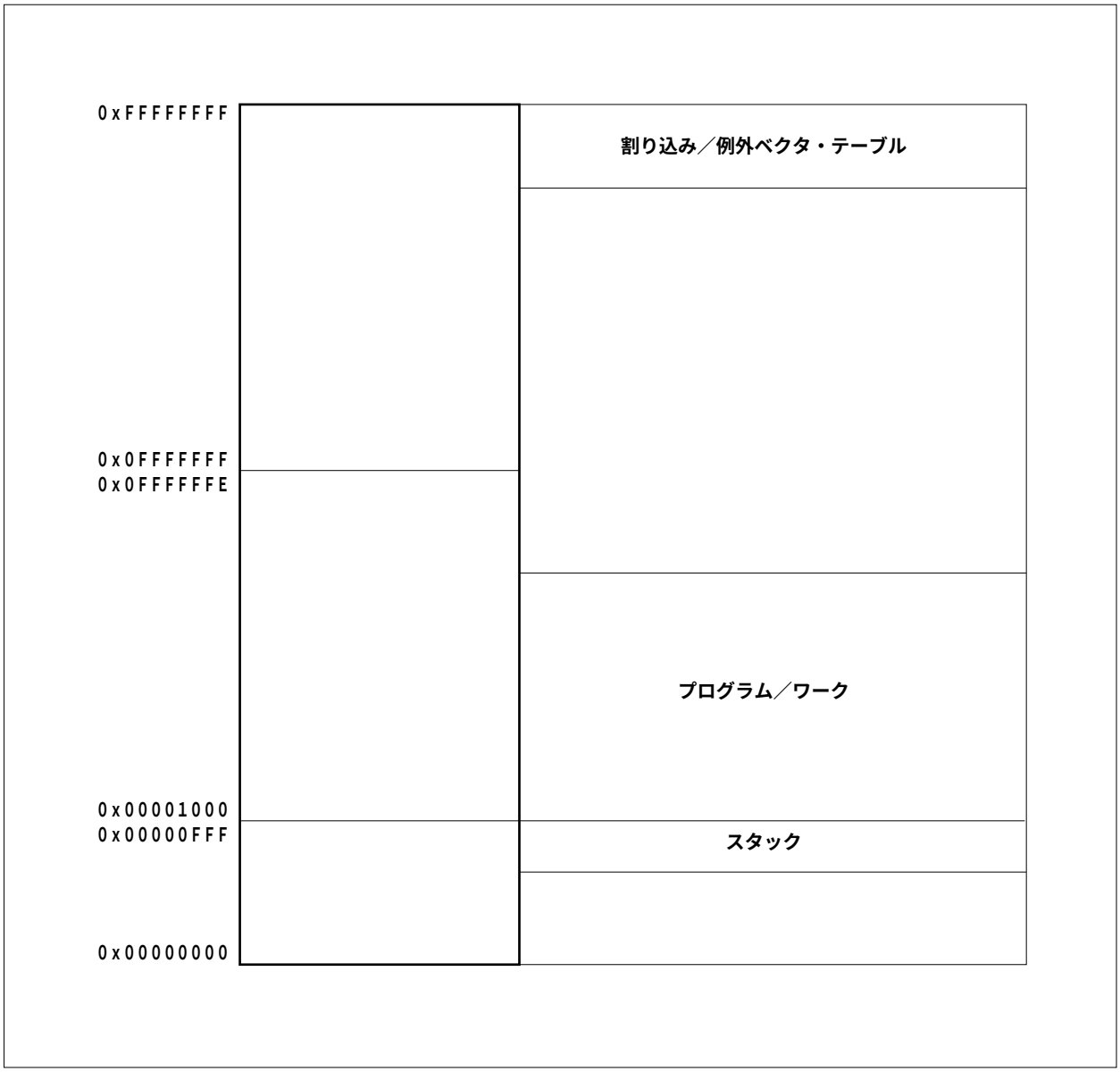
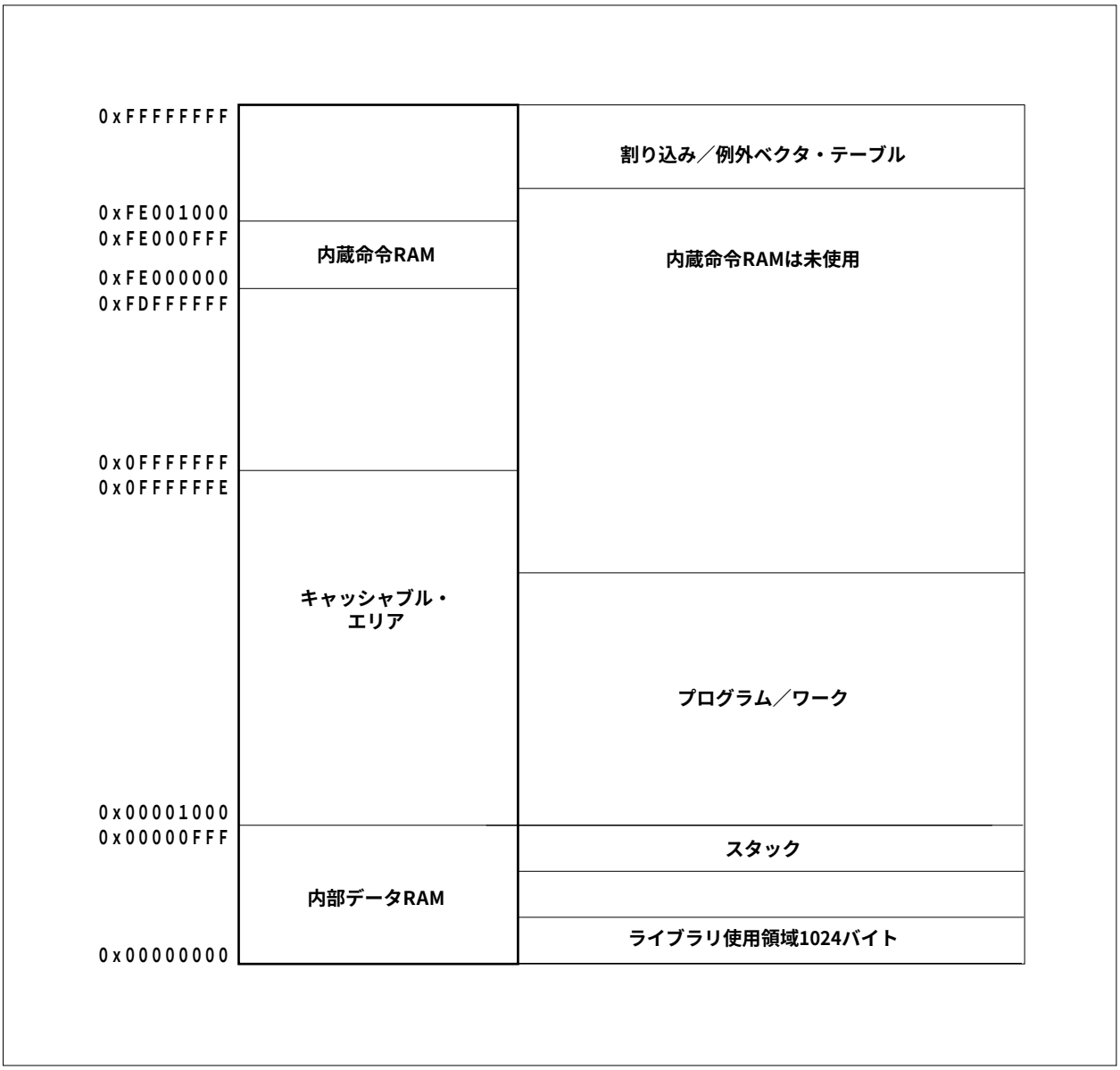


図１－27　サンプル・プログラムのメモリ・マップ（AP705100-B03用）



保守／廃止

第２章　基本ライブラリ仕様

2.1　機　　能

AP705100-B03, AP70732-B03で用意されている基本ライブラリ群を使うことにより，次の２つの処理が実現できます。

（1）圧縮処理

指定された画像を，指定された量子化テーブル／ハフマン・テーブルを用いてJPEG形式で圧縮し，JPEGファイルを生成します。

アプリケーション・セグメントを挿入するように指定すれば，ヘッダ部分にAPPnセグメントの形で埋め込みます。

1MCU分の圧縮されたデータのビット数がどのくらいになるかテストするモードを設けました。

（2）伸長処理

JPEGファイルを伸長します。

設定によっては，画像サイズやAPPnセグメントのアドレスなどを検出するだけで伸長そのものは実行しない選択ができます。

画像全体ではなく，MCU単位の矩形でクリッピングして伸長できます。

画像をそのままの大きさではなく，縦横それぞれ８分の１（面積で64分の１）など，縮小したサイズで伸長できます。

保守／廃止

2.1.1　VRAM（画像メモリ）構成による基本ライブラリの動作の違い

VRAM（画像メモリ）がYCbCr形式ではなく，RGB形式であった場合は，次に示す処理が必要です。

・圧縮：RGB形式のデータを一度YCbCr形式に変換してから圧縮

・伸長：伸長したYCbCr形式のデータをメモリに書き出す前にRGB形式に変換

画像メモリがYCbCr形式であるか，RGB形式であるかによって別々の基本ライブラリのオブジェクトがリンクされることになります。

また，画像メモリが画像全体のデータを保持できるほど大きい場合と，画像の一部しか保持できない場合とでは，リンクするライブラリが異なってきます。

図２－１　VRAMが大きい場合のライブラリ

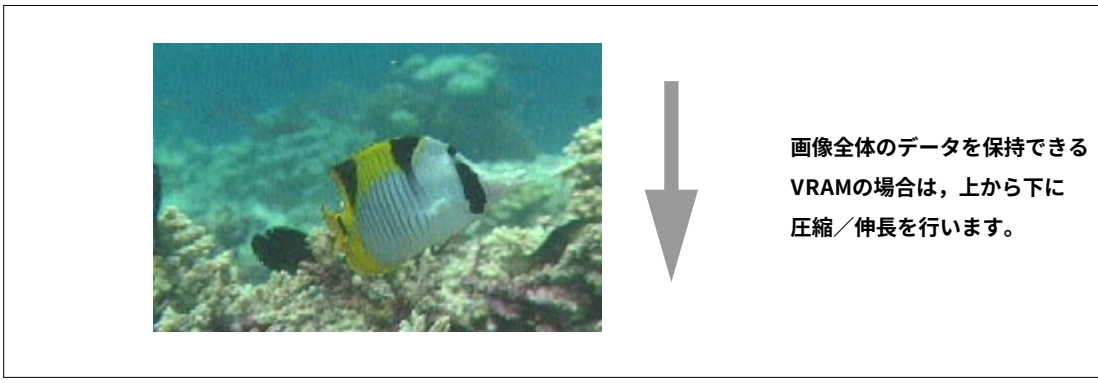
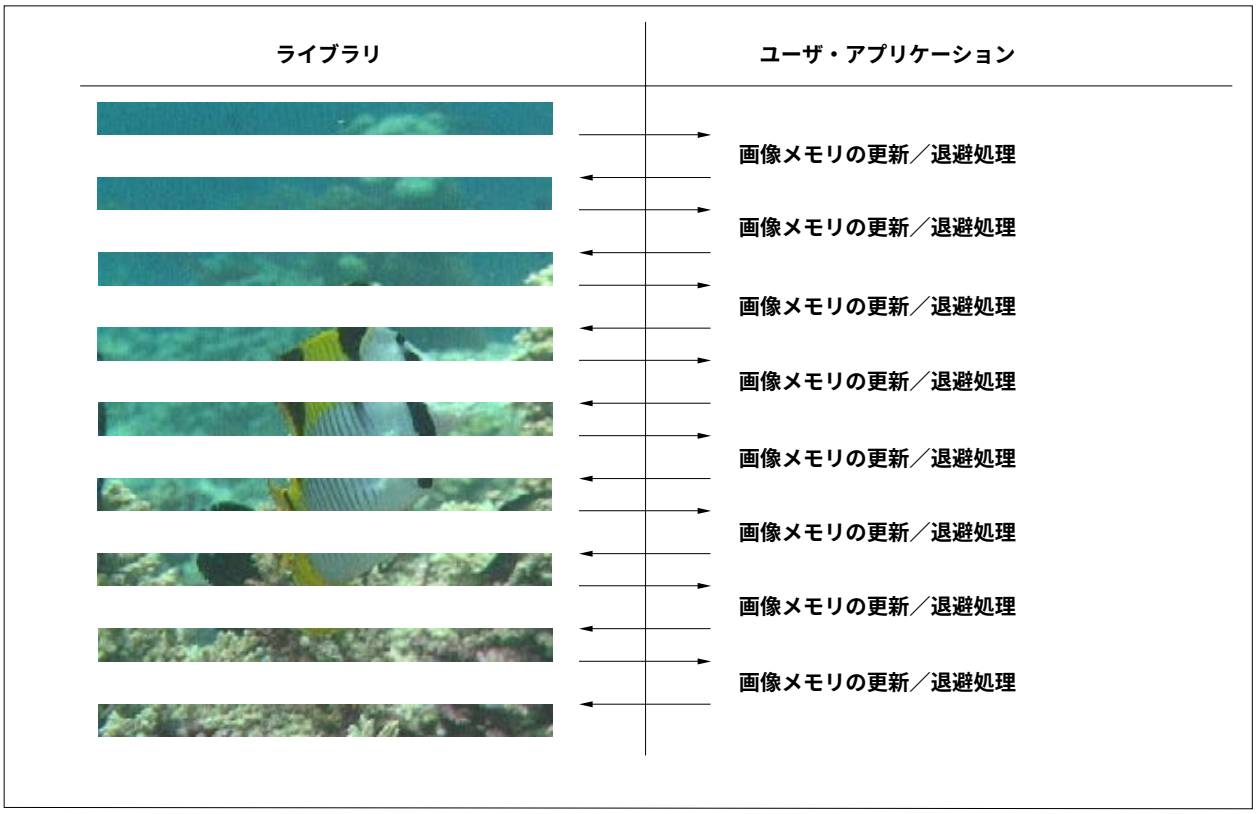


図２－２　VRAMが小さい場合のライブラリ



保守／廃止

表２－1　最低限必要な画像メモリ量

サンプル比	最低限必要な画像メモリ
4：1：1（H：V＝2：2）	縦16ピクセル分のランダム・アクセス可能なRAM
4：1：1（H：V＝4：1）	縦8ピクセル分のランダム・アクセス可能なRAM
2：1：1（H：V＝2：1）	縦8ピクセル分のランダム・アクセス可能なRAM
1：1：1（H：V＝1：1）	縦8ピクセル分のランダム・アクセス可能なRAM

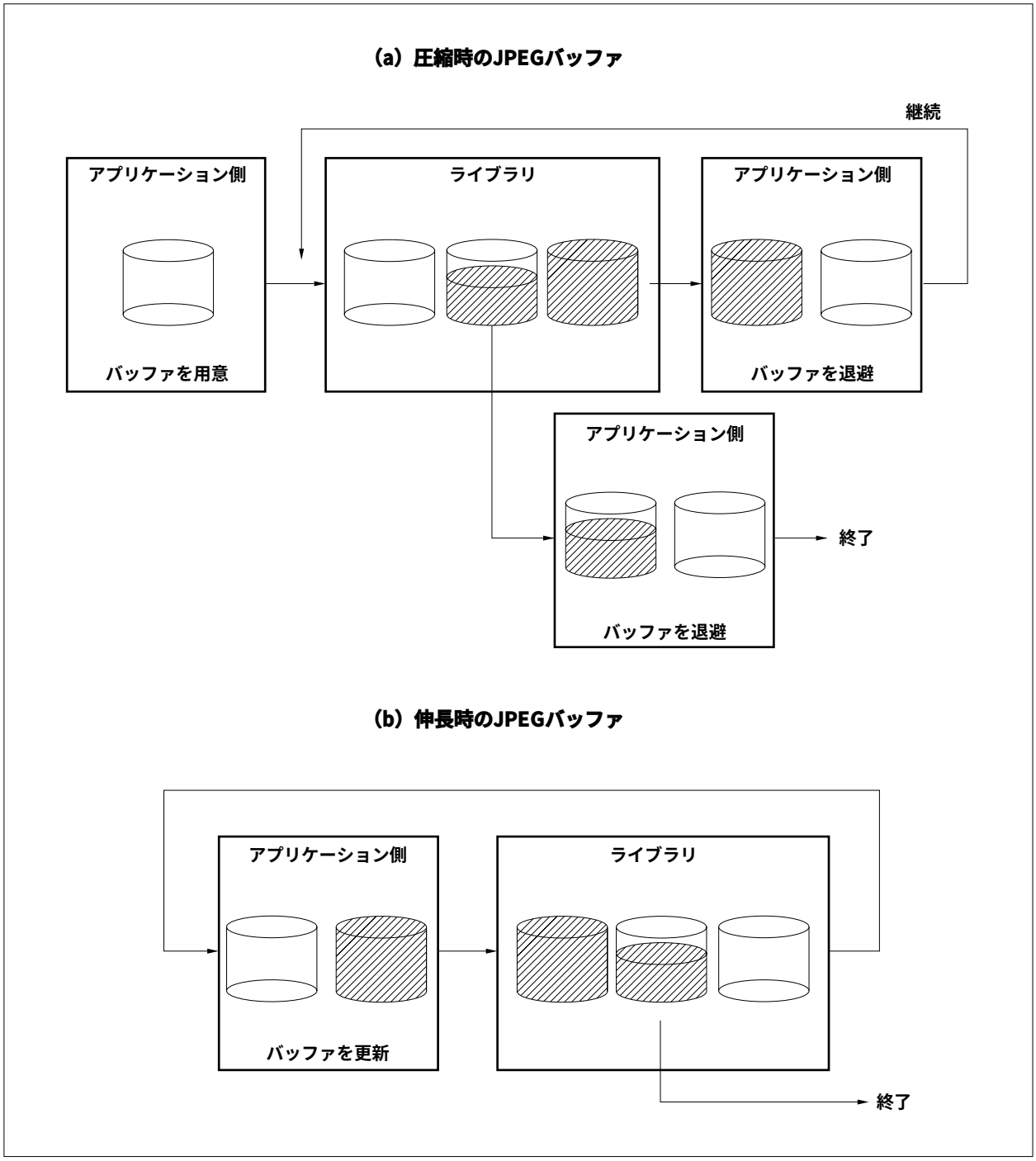
画像全体を保持するような画像メモリがないようなシステムであっても，表２－１に示すメモリは最低限必要です。メモリが少ないシステムでは，図２－２のように16ドット・ライン（８ドット・ライン）単位で画像の圧縮／伸長と画像メモリの更新／退避処理を交互に繰り返すような処理を行います。

2.1.2　JPEGバッファ

一般にJPEGファイルのサイズは画像や圧縮パラメータによって大きく変わってきます。また，画像やパラメータからサイズが容易に予測できるような種類のものでもありません。そこで，AP705100-B03，AP70732-B03の基本ライブラリではJPEGファイル用に用意したバッファよりもJPEGファイル・サイズが大きかった場合に，いったん処理を中断し，バッファの内容を退避（圧縮）／更新（伸長）してから処理を再開するような仕組みをとっています。

保守／廃止

図２－３　JPEGバッファの使用



保守／廃止

２.１.３　演算精度

JPEGはアナログのデータをディジタルのデータに変換する処理です。その処理を行う際に演算精度による情報の欠落があります。ここでは，このライブラリの品質について説明します。

表２－２　処理ごとの情報の欠落

処　　理	情報の欠落
エントロピ符号化／復号化	情報の欠落はありません。 エントロピ符号化する前のデータとエントロピ復号化したあとのデータは同じ値です。
量子化／逆量子化	JPEGの一連の処理の中で，量子化では最も多くの情報が失われます。DCT演算の結果得られたデータを量子化テーブルの値で割ると，余りは情報として失われます。 ただし，量子化パラメータの値を100に設定してから圧縮すると，量子化テーブルのすべての要素が１なので，情報の欠落はなくなります。
DCT変換／逆DCT変換	次の部分で情報の欠落があります。 ●DCT変換／逆DCT変換の数式による出力値（周波数分解した係数値）を16ビットの整数値で処理します（数式のうえでは実数でなければなりません）。 ●高速化のため16ビット精度の固定小数点で処理します。

DCT変換／逆DCT変換の精度について説明します。

次の手順で精度の検査を行います。

（１）short型（２バイト）の64要素のバッファを20000個用意する。

```
short　BLK［20000］［64］；
```

（２）次の乱数発生プログラムで，－128～127の範囲で10000ブロック分の整数画像データを発生させ，８×８のブロックにまとめる。

発生させた10000ブロックのそれぞれに対して，正負を反転したブロックを用意する。

合計20000ブロックをDCT変換の入力とする。

保守／廃止

```
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）それぞれのブロックBLK［n］（n = 0, ..., 19999）に対して，ライブラリのDCT変換，ライブラリの逆DCT変換を適用する。出力値をOUT［n］［64］（n = 0, ..., 19999）とする。

（４）BLK［20000］［64］とOUT［20000］［64］の間に生じる次の値を測定する。

① 最大誤差
② 要素ごとの平均２乗誤差
③ 全体の平均２乗誤差
④ 要素ごとの平均誤差
⑤ 全体の平均誤差

測定結果が次のようになったとします。

入力データと出力データの差：

$$DIFF[b][n] = BLK[b][n] - OUT[b][n] \quad ;$$

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

① 最大誤差：

$$\underset{b,n}{MAX} \left| DIFF[b][n] \right| = 2$$

② 要素ごとの平均２乗誤差：

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

③ 全体の平均２乗誤差

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

④ 要素ごとの平均誤差

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

⑤ 全体の平均誤差

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

たとえば，③の平均２乗誤差の値が0.3313の部分はDCT変換と逆DCT変換を実行したことによる誤差が256階調分の約0.33であることを意味しています。

②の値0.3475と③の値0.3313の間に大きな差がないことは，８×８ブロックの要素の中に著しく誤差が偏った場所はなく，ブロック全体に均一に負荷がかかっていることを示しています。

保守／廃止

2.1.4　圧縮時のオプション

（1）サンプル比の選択

基本ライブラリではサンプル比を次の4つの中から選択できます。

- 4：1：1〔H：V＝2：2〕　（1MCUは縦16ピクセル，横16ピクセル）
- 4：1：1〔H：V＝4：1〕　（1MCUは縦8ピクセル，横32ピクセル）
- 2：1：1〔H：V＝2：1〕　（1MCUは縦8ピクセル，横16ピクセル）
- 1：1：1〔H：V＝1：1〕　（1MCUは縦8ピクセル，横8ピクセル）

注意　上記以外のサンプル比には対応していません。

（2）ハフマン・テーブルと量子化テーブル

圧縮率を大きく作用するパラメータとして，ハフマン・テーブルと量子化テーブルがあります。このライブラリではこれらのテーブルを指定できるようになっています。

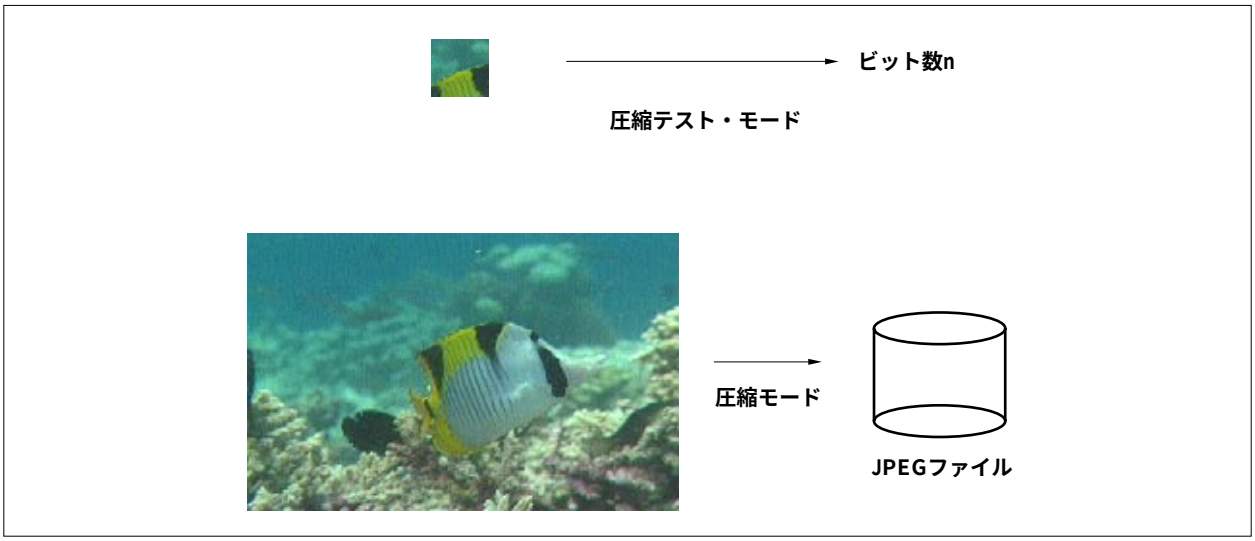
（3）量子化パラメータの設定

量子化テーブルは圧縮率を変えるのに非常に有効です。画質と圧縮率はトレード・オフの関係にあるのですが，これを簡単に調節するのが量子化パラメータです。

（4）圧縮／圧縮テストの選択

画像の圧縮を実際に行うモードと，1MCUが何ビットに圧縮されるかをテストするモードとがあります。

図２－４　圧縮モード



（5）リスタート・インターバル

リスタート・マーカを用いるか用いないかの設定ができます。用いる場合は，どの間隔で挿入するかを選択できます。

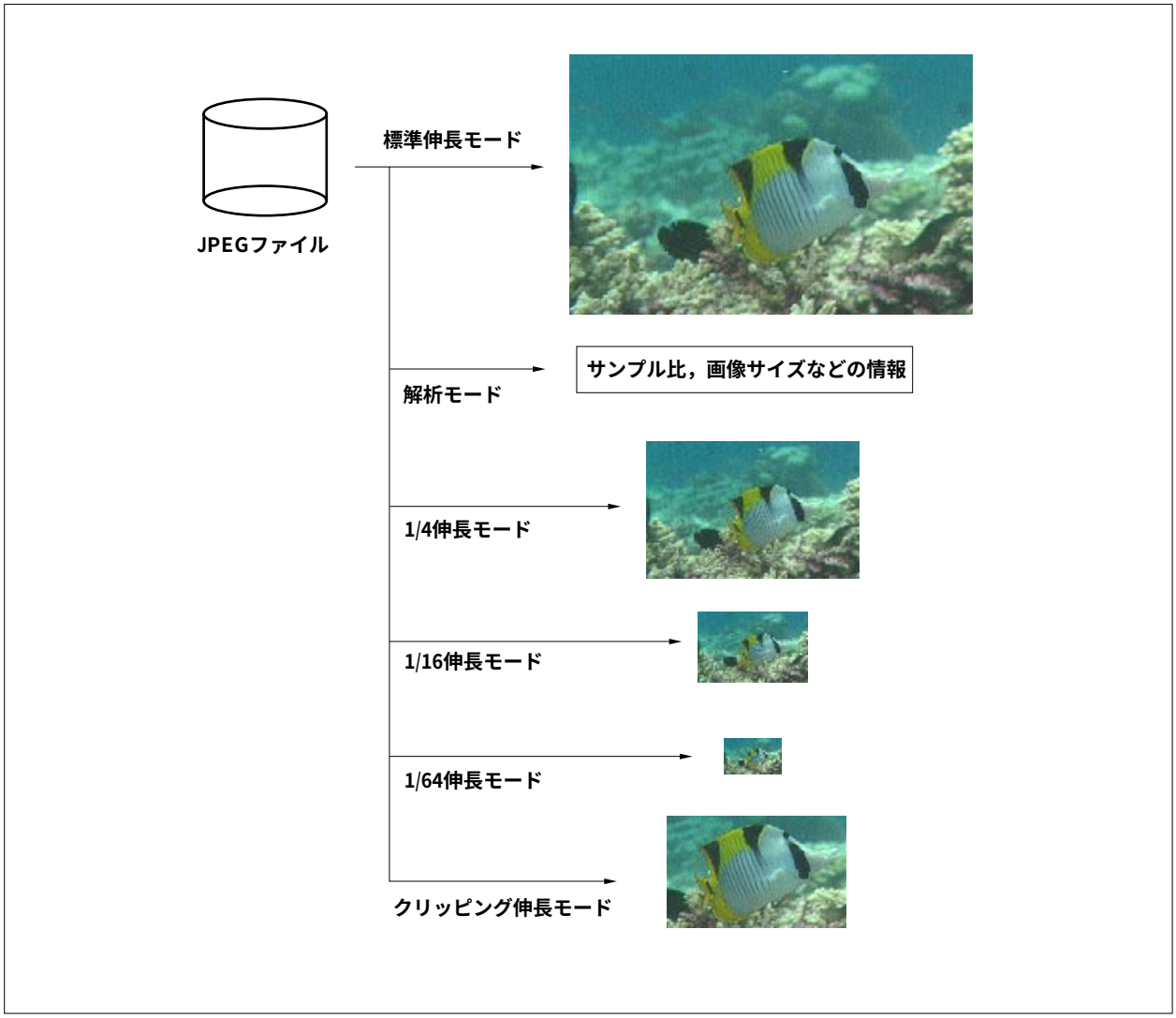
保守／廃止

２．１．５　基本伸長時のオプション

基本伸長の主なオプションは，モード設定です。モードの設定によって，JPEGヘッダの解析だけを行うか，普通に伸長するか，Thumbnail用に縮小して伸長するか，クリッピングして伸長するかが決められます。

クリッピングの場合は，どの位置をクリッピングするかを指定します。

図２－５　伸長モード



保守／廃止

2.1.6　圧縮テスト・オプションについての注意

圧縮テスト・モードで得られるビット数nは概算値です。なぜなら，通常の圧縮では，DC成分は差分値（前のブロックとの差分）を圧縮するとか，圧縮データがバイト単位で0xFFになった場合にはマーカと区別するため0x00を挿入するなどのJPEG固有の性質（前後のMCUに依存）があるからです。

JPEGファイル全体のバイト数

$$\{\sum_{i=0}^{m}\sum_{j=0}^{n}\text{（（m,n）番目のMCUをテスト圧縮したビット数）}\ \}/8$$

＋ヘッダに必要なバイト数（約300バイト）

m：横方向のMCUの個数

n：縦方向のMCUの個数

保守／廃止

2.2　基本ライブラリのリンク

2.2.1　リンク時のライブラリ選択

次の３つの項目については，ユーザがリンク時にライブラリを選択できます。

・不要なオブジェクトの非リンク
・VRAMのYCbCr/RGB選択
・VRAMの画像単位処理／MCU行単位処理の選択

ライブラリを選択するには，次のコマンドを使用します。

jparc830.exe：DOS用
jparc830：Sun4用

注意　DOSの場合，Windowsではなく，コマンド・ラインで実行してください。

この関数を実行することにより，ファイル“ archive ”が作成されます。同じ名前のファイルが存在した場合には上書きされます。このファイルはmakeファイルの中に書かれていて，リンク時に参照されます。

（1）不要なオブジェクトをリンクしない

ファイル“ archive ”作成コマンドを実行すると，不要なコードをリンクしないような設定をするために，次のようなメッセージが表示されます。順に答えてください。

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

⋮

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

⋮

保守／廃止

（２）ディフォールトのVRAMアクセス関数の使用／不使用

次のようなメッセージが表示されますので，“ Ｙ ”か“ Ｎ ”を入力してください。

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

ここで“ 使用する ”を選択した場合，次のメッセージが表示されます。該当するものを選んでください。

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

ディフォールトのVRAMアクセス関数を使用しない場合には，次のような関数を自作する必要があります。

- 圧縮：jpeg_getMCU22, jpeg_getMCU41, jpeg_getMCU21, jpeg_getMCU11
- 伸長：jpeg_putMCU22x, jpeg_putMCU41x, jpeg_putMCU21x, jpeg_putMCU11x

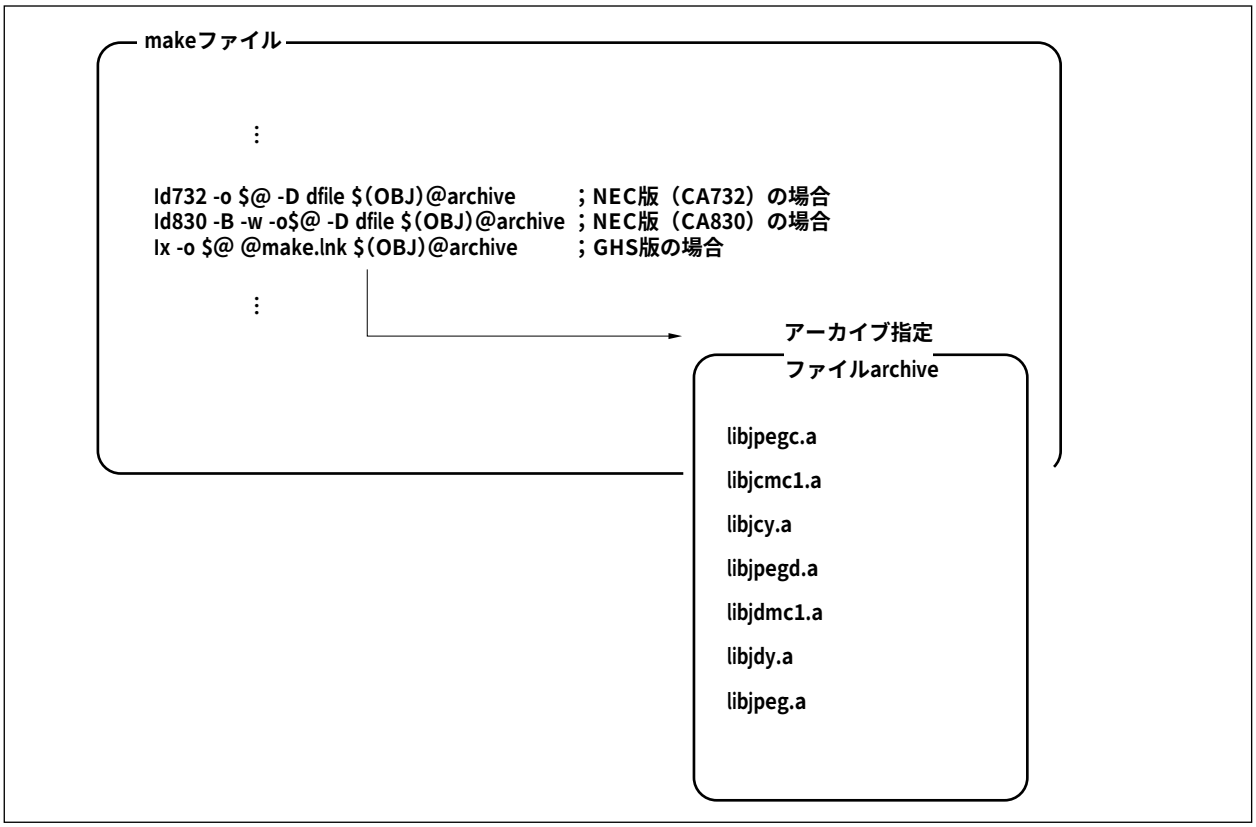
詳細は**２.６　基本ライブラリのカスタマイズ**を参照してください。

保守／廃止

２．２．２　アーカイブ・ファイルの指定

アーカイブ指定ファイル作成コマンドを実行すると，ファイル“ archive ”が作成されます。また，このコマンドはmakeファイルの中で，リンカの引き数に@archiveの形でファイルの内容を渡します。なお，オプションなどの詳細は，リンカのマニュアルを参照してください。

図２－６　アーカイバ指定方法



アーカイブ・ファイルlibjpeg.aの中にはディフォールトのライブラリが入っています。

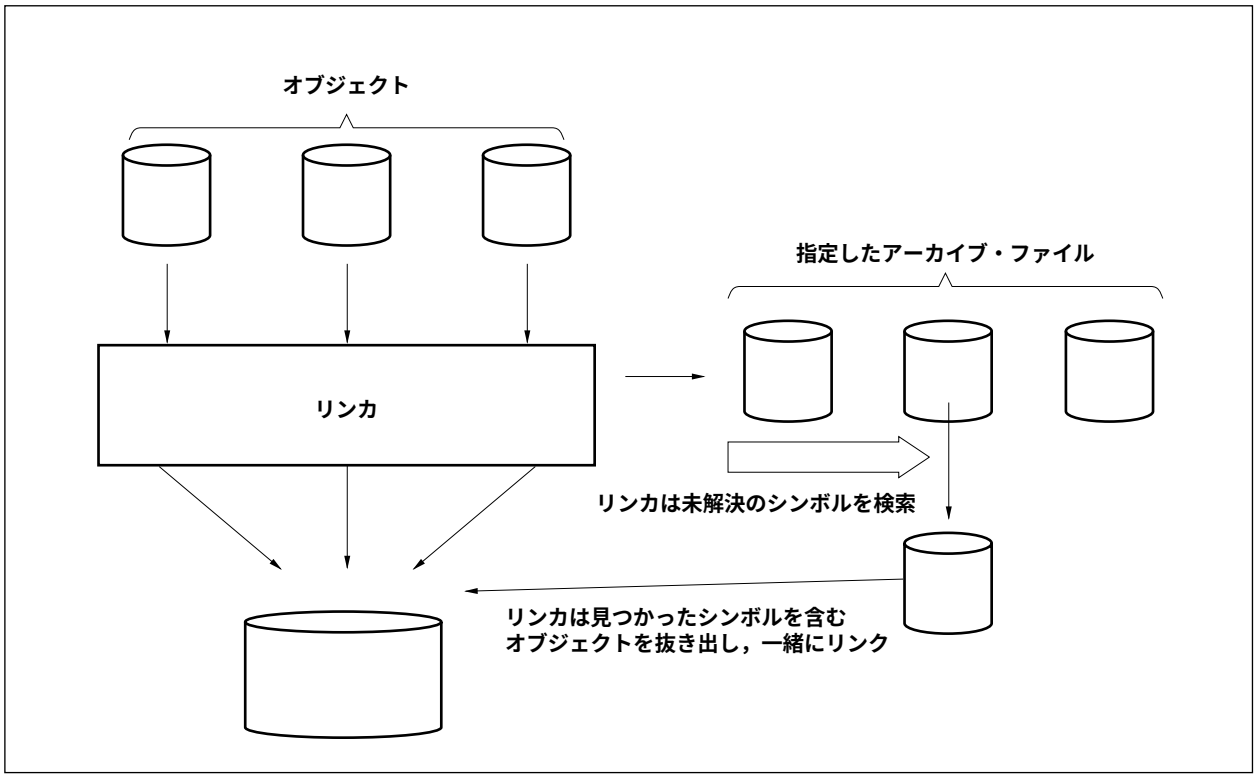
ファイル“ archive ”をエディタを使って自作する場合には必ず，アーカイブ指定ファイルの最後に，libjpeg.aを指定してください。

保守／廃止

リンカは，オブジェクト内で未解決のシンボルを解決するために，指定したアーカイブ・ファイルを順に検索します。

見つかったシンボルを含むオブジェクト・ファイルをアーカイブ・ファイルの中から抜き出して，一緒にリンクします。

図２－７　リンカによるアーカイブ・ファイルの取り扱い



保守／廃止

２.２.３　より高度なライブラリ指定

少しでも命令コード・サイズを小さくしたい場合，たとえば，サンプル比２：１：１だけ対応するようにして，４：１：１や１：１：１のものをリンクさせたくない場合には，アーカイブ・ファイルlibjcmcx.a/libjdmcx.aを直接書き換えることでそれが可能になります。

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

上記コマンドを実行すると，アーカイブ・ファイルの中に含まれるオブジェクト・ファイル名を表示させられます。

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

以上のようにすれば，アーカイブ・ファイルから指定したオブジェクト・ファイルを削除できます。不要なサンプル比のオブジェクト・ファイルを削除すれば，それらのオブジェクト・ファイルがリンクされません。

表２－３　圧縮処理系のサンプル比に固有のオブジェクト・ファイル

サンプル比	オブジェクト・ファイル（アーカイブ・ファイル）
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）

表２－４　伸長処理系のサンプル比に固有のオブジェクト・ファイル

サンプル比	オブジェクト・ファイル（アーカイブ・ファイル）
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）

保守／廃止

２.２.４　ABcond命令対応

基本ライブラリでは，ABcond命令（分岐履歴機能付き条件分岐命令）を含まないようなライブラリがリンクされるようになっています。

ABcond命令を含むような（部分的に，分岐命令を高速条件分岐に置き換えたような）ライブラリを加工するためには，表２－５に示すスクリプトを実行してください。

備考　高速条件分岐命令を取り入れることにより，若干高速化されます。コード・サイズの変更はありません。

表２－５　基本ライブラリの加工に必要なスクリプトの一覧（1/2）

NECコンパイラの場合	GHSコンパイラの場合
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 gmcuyc41.o
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 libjdy.a pmcuy228.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

保守／廃止

表２－５　基本ライブラリの加工に必要なスクリプトの一覧（2/2）

NECコンパイラの場合	GHSコンパイラの場合
ar830 d libjpeg.a jrdct1.o	ax d libjpeg.a jrdct1.o
ar830 d libjpeg.a jrdct2p.o	ax d libjpeg.a jrdct2p.o
ar830 d libjpeg.a jrdct4p.o	ax d libjpeg.a jrdct4p.o
ar830 d libjpeg.a jrdct8.o	ax d libjpeg.a jrdct8.o

2.2.5　RGBライブラリ（libjcr2.a, libjdr2.a）の追加

RGB⟷YCbCr変換式はCCIR勧告による規格CCIR601-1の中で次のように規定されています。

$$\left.\begin{array}{lcrlrlrl} Y & = & 0.29900 & \times R & +0.58700 & \times G & +0.11400 & \times B \\ Cb & = & -0.16874 & \times R & -0.33126 & \times G & +0.50000 & \times B \\ Cr & = & 0.50000 & \times R & -0.41869 & \times G & -0.08131 & \times B \end{array}\right\} \cdots ①$$

$$\left.\begin{array}{lcrlrlrl} R & = & Y & & & & +1.40200 & \times Cr \\ G & = & Y & -0.34414 & \times Cb & & -0.71414 & \times Cr \\ B & = & Y & +1.77200 & \times Cb & & & \end{array}\right\} \cdots ②$$

しかし，場合によっては，上記の式以外に次のような色変換式が使われていることもあります。

$$\left.\begin{array}{lcrlrlrl} Y & = & 0.2990 & \times R & +0.5870 & \times G & +0.1140 & \times B \\ Cb & = & -0.1684 & \times R & -0.3316 & \times G & +0.5000 & \times B \\ Cr & = & 0.5000 & \times R & -0.4187 & \times G & -0.0813 & \times B \end{array}\right\} \cdots ③$$

$$\left.\begin{array}{lcrlrlrl} R & = & Y & & & & +1.4020 & \times Cr \\ G & = & Y & -0.3441 & \times Cb & & -0.7139 & \times Cr \\ B & = & Y & +1.7718 & \times Cb & & -0.0013 & \times Cr \end{array}\right\} \cdots ④$$

AP705100-B03, AP70732-B03の基本ライブラリで，ディフォールトのRGB形式VRAMアクセス・ライブラリを使用する選択をした場合には，それぞれ次に示すライブラリがリンクされます。

圧縮処理時：①式に準拠したライブラリ
伸長処理時：②式に準拠したライブラリ

たとえば，Windowsアプリケーション等で③，④式に従う色変換を行っていた場合，①式で作成したJPEGファイルを④式で伸長すると赤色成分が若干低減されます。

AP705100-B03, AP70732-B03の基本ライブラリで，①式の代わりに③式を採用または，②式の代わりに④式を採用する場合には次に示すようにリンカのオプション指定を変更してください。

圧縮処理時：リンカのオプション指定で指定されるlibjcr.aをlibjcr2.aに変える
伸長処理時：リンカのオプション指定で指定されるlibjdr.aをlibjdr2.aに変える

保守／廃止

２．２．６　リンクのメモリ・マップ

それぞれのセクションに対するマッピングは，次に示すファイルで行います。

・NEC版：dfile（リンク・ディレクティブ・ファイル）
・GHS版：make.lnk（セクション指定ファイル）

ユーザは，makeファイルを書き換えるのと同様，これらのファイルを書き換える必要があります。サンプルで付属しているファイルを参考にして書き換えてください。

これらのファイルの書式などの詳細については，リンカのマニュアルの次に示す箇所を参照してください。

・NEC版：リンク・ディレクティブの部分
・GHS版：“ -sec ”オプションの部分

２．２．７　コンパイル・オプション

基本ライブラリでは，レジスタ32本を最大限に使用しています。そのため，レジスタ32本モード以外はサポートしていません。

これ以外のコンパイル・オプションは，それぞれのコンパイラのマニュアルを参照してください。

保守／廃止

２.３　基本ライブラリの構造体とメモリ

基本ライブラリでは，圧縮／伸長のそれぞれの処理に対して，決められた大きさのメモリを確保してください。

V810ファミリ版の場合

メモリ	用途とサイズ
CJINFO構造体	圧縮処理を行う場合に最大1152バイト必要です（サンプル比によって異なります）。
DJINFO構造体	伸長処理を行う場合に最大1152バイト必要です（サンプル比によって異なります）。
APPINFO構造体	圧縮でAPPセグメントに情報を埋め込む場合，また，伸長でAPPセグメントに関する情報を得る場合に最大160バイト必要です。 内部RAMには配置しないでください。
JPEGバッファ	圧縮ではでき上がったJPEGファイルを，伸長では伸長すべきJPEGファイルを格納するバッファです。 １バイト以上，任意のバイト数を設定できます。 JPEGファイルが一度に入らない場合は，分割して処理することになります。
外部RAMワーク・エリア	圧縮では2688バイト，伸長では3968バイト必要です。

V830ファミリ版の場合

メモリ	用途とサイズ
CJINFO構造体	圧縮処理を行う場合に最大128バイト必要です。
DJINFO構造体	伸長処理を行う場合に最大128バイト必要です。
APPINFO構造体	圧縮でAPPセグメントに情報を埋め込む場合，また，伸長でAPPセグメントに関する情報を得る場合に最大160バイト必要です。 内部RAMには配置しないでください。
内部RAMワーク・エリア	サンプル比によって最大1024バイト必要です。
JPEGバッファ	圧縮ではでき上がったJPEGファイルを，伸長では伸長すべきJPEGファイルを格納するバッファです。 １バイト以上，任意のバイト数を設定できます。 JPEGファイルが一度に入らない場合は，分割して処理することになります。
外部RAMワーク・エリア	圧縮では2688バイト，伸長では3968バイト必要です。

保守／廃止

２.３.１　CJINFO構造体

CJINFO構造体は，圧縮処理に用いられる構造体です。

この構造体の型はファイルjpeg.hの中で定義されています。

圧縮ルーチンには，この構造体の先頭アドレスを引き数として渡します。

表２－６　CJINFO構造体（AP70732-B03）

メンバ	型	内　　容	IN/OUT
ErrorState	int	エラー・ステータス	IN/OUT
FileSize	int	JPEGファイル・サイズ	OUT
Restart	unsigned short	リスタート・インターバル	IN
Width	unsigned short	画像の横ピクセル数	IN
Height	unsigned short	画像の縦ピクセル数	IN
Quality	char	量子化パラメータ	IN
Sampling	char	サンプル比	IN
Mode	char	圧縮モード	IN
Reserve	char×3	予約	－
JPEG_Buff_Bptr	unsigned char＊	JPEGバッファ先頭アドレス	IN
JPEG_Buff_Eptr	unsigned char＊	JPEGバッファ先頭アドレス＋JPEGバッファ・サイズ	IN
IRAM_Buff_Bptr	int＊	予約	－
StartX	short	画像の開始ｘ位置（ピクセル数）	IN
StartY	short	画像の開始ｙ位置（ピクセル数）	IN
VRAM_Bptr	unsigned char＊	VRAM先頭アドレス	IN注１
VRAM_W_Pixel	short	VRAMの横幅（ピクセル数）	IN注２
VRAM_H_Pixel	short	VRAMの縦幅（ピクセル数）	IN注２
VRAM_Line_Byte	int	VRAMの縦１ピクセル分のアドレス差	IN注１
VRAM_Pixel_Byte	int	VRAMの横１ピクセル分のアドレス差	IN注１
VRAM_Gap1_Byte	int	VRAMのＹとCbとの／ＲとＢとのアドレス差	IN注１
VRAM_Gap2_Byte	int	VRAMのＹとCrとの／ＲとＧとのアドレス差	IN注１
APP_Info_Bptr	APPINFO＊	APPINFO構造体先頭アドレス	IN
DQT_Y_Bptr	char＊	輝度成分用量子化テーブル先頭アドレス	IN
DQT_C_Bptr	char＊	色差成分用量子化テーブル先頭アドレス	IN
DHT_DC_Y_Bptr	char＊	輝度DC用ハフマン・テーブル先頭アドレス	IN
DHT_DC_C_Bptr	char＊	輝度AC用ハフマン・テーブル先頭アドレス	IN
DHT_AC_Y_Bptr	char＊	色差DC用ハフマン・テーブル先頭アドレス	IN
DHT_AC_C_Bptr	char＊	色差AC用ハフマン・テーブル先頭アドレス	IN
Work	int＊	外部RAMワーク・エリア先頭アドレス	IN
CurrentX	short	VRAM描画用ワーク（内部使用）	－
CurrentY	short	VRAM描画用ワーク（内部使用）	－
IR	32＋256 byte	内部予約（内部使用ワーク・エリア）	注３
MCUbuff	0x180 unsigned short	MCUバッファ	－

注１．getmcu関数を自作する場合は設定不要です。

２．getmcu関数を自作する場合はダミーで設定してください。

★ ３．IRメンバの領域は０クリアしてください。ただし，次のいずれかのJPEGフォーマットで圧縮する場合は，IR領域に値を設定してください。

・コメント・マーカのアドレス指定挿入（２.４.４　コメント・マーカ設定参照）

・Exitフォーマット圧縮（２.４.５　DHTセグメント，DQTセグメント，２.４.７　Exif対応参照）

保守／廃止

表２－７　CJINFO構造体（AP705100-B03）

メンバ	型	内　　容	IN/OUT
ErrorState	int	エラー・ステータス	IN/OUT
FileSize	int	JPEGファイル・サイズ	OUT
Restart	unsigned short	リスタート・インターバル	IN
Width	unsigned short	画像の横ピクセル数	IN
Height	unsigned short	画像の縦ピクセル数	IN
Quality	char	量子化パラメータ	IN
Sampling	char	サンプル比	IN
Mode	char	圧縮モード	IN
Reserve	char×3	予約	－
JPEG_Buff_Bptr	unsigned char＊	JPEGバッファ先頭アドレス	IN
JPEG_Buff_Eptr	unsigned char＊	JPEGバッファ先頭アドレス＋JPEGバッファ・サイズ	IN
IRAM_Buff_Bptr	int＊	内部RAMワーク・エリア先頭アドレス	IN
StartX	short	画像の開始ｘ位置（ピクセル数）	IN
StartY	short	画像の開始ｙ位置（ピクセル数）	IN
VRAM_Bptr	unsigned char＊	VRAM先頭アドレス	IN注１
VRAM_W_Pixel	short	VRAMの横幅（ピクセル数）	IN注２
VRAM_H_Pixel	short	VRAMの縦幅（ピクセル数）	IN注２
VRAM_Line_Byte	int	VRAMの縦１ピクセル分のアドレス差	IN注１
VRAM_Pixel_Byte	int	VRAMの横１ピクセル分のアドレス差	IN注１
VRAM_Gap1_Byte	int	VRAMのＹとCbとの／ＲとＢとのアドレス差	IN注１
VRAM_Gap2_Byte	int	VRAMのＹとCrとの／ＲとＧとのアドレス差	IN注１
APP_Info_Bptr	APPINFO＊	APPINFO構造体先頭アドレス	IN
DQT_Y_Bptr	char＊	輝度成分用量子化テーブル先頭アドレス	IN
DQT_C_Bptr	char＊	色差成分用量子化テーブル先頭アドレス	IN
DHT_DC_Y_Bptr	char＊	輝度DC用ハフマン・テーブル先頭アドレス	IN
DHT_DC_C_Bptr	char＊	輝度AC用ハフマン・テーブル先頭アドレス	IN
DHT_AC_Y_Bptr	char＊	色差DC用ハフマン・テーブル先頭アドレス	IN
DHT_AC_C_Bptr	char＊	色差AC用ハフマン・テーブル先頭アドレス	IN
Work	int＊	外部RAMワーク・エリア先頭アドレス	IN
CurrentX	short	VRAM描画用ワーク（内部使用）	－
CurrentY	short	VRAM描画用ワーク（内部使用）	－
IR	32 byte	内部予約（内部使用ワーク・エリア）	注３

注１．getmcu関数を自作する場合は設定不要です。

２．getmcu関数を自作する場合はダミーで設定してください。

★ ３．IRメンバの領域は０クリアしてください。ただし，次のいずれかのJPEGフォーマットで圧縮する場合は，IR領域に値を設定してください。

・コメント・マーカのアドレス指定挿入（**２．４．４　コメント・マーカ設定**参照）

・Exitフォーマット圧縮（**２．４．５　DHTセグメント，DQTセグメント，２．４．７　Exif対応**参照）

保守／廃止

２．３．２　DJINFO構造体

DJINFO構造体は，基本伸長処理に用いられる構造体です。

この構造体の型は，ファイルjpeg.hの中で定義されています。

伸長ルーチンには，この構造体の先頭アドレスを引き数として渡します。

表２－８　DJINFO構造体（AP70732-B03）

メンバ	型	内　　容	IN/OUT
ErrorState	int	エラー・ステータス	IN/OUT
FileSize	int	JPEGファイル・サイズ	OUT
Restart	unsigned short	リスタート・インターバル	OUT
Width	unsigned short	画像の横ピクセル数	OUT
Height	unsigned short	画像の縦ピクセル数	OUT
Sampling	char	サンプル比	OUT
Mode	char	伸長モード	IN
JPEG_Buff_Bptr	unsigned char＊	JPEGバッファ先頭アドレス	IN
JPEG_Buff_Eptr	unsigned char＊	JPEGバッファ先頭アドレス＋JPEGバッファ・サイズ	IN
IRAM_Buff_Bptr	int＊	予約	－
StartX	short	画像の開始ｘ位置（ピクセル数）	IN
StartY	short	画像の開始ｙ位置（ピクセル数）	IN
VRAM_Bptr	unsigned char＊	VRAM先頭アドレス	IN注１
VRAM_W_Pixel	short	VRAMの横幅（ピクセル数）	IN注２
VRAM_H_Pixel	short	VRAMの縦幅（ピクセル数）	IN注２
VRAM_Line_Byte	int	VRAMの縦１ピクセル分のアドレス差	IN注１
VRAM_Pixel_Byte	int	VRAMの横１ピクセル分のアドレス差	IN注１
VRAM_Gap1_Byte	int	VRAMのＹとCbとの／ＲとＢとのアドレス差	IN注１
VRAM_Gap2_Byte	int	VRAMのＹとCrとの／ＲとＧとのアドレス差	IN注１
APP_Info_Bptr	APPINFO＊	APPINFO構造体先頭アドレス	IN
ClipSX	unsigned short	クリッピング開始位置（クリッピング・モード時だけ有効）	IN
ClipSY	unsigned short	クリッピング開始位置（クリッピング・モード時だけ有効）	IN
ClipW	unsigned short	クリッピング横幅（クリッピング・モード時だけ有効）	IN
ClipH	unsigned short	クリッピング縦幅（クリッピング・モード時だけ有効）	IN
Work	int＊	外部RAMワーク・エリア先頭アドレス	IN
CurrentX	short	VRAM描画用ワーク（内部使用）	－
CurrentY	short	VRAM描画用ワーク（内部使用）	－
IR	52＋256 byte	内部予約（内部使用ワーク・エリア）	注３
MCUbuff	0x180 unsigned short	MCUバッファ	－

注１．putmcu関数を自作する場合は設定不要です。

２．putmcu関数を自作する場合はダミーで設定してください。

★ ３．IRメンバの領域は０クリアしてください。ただし，次のJPEGフォーマットで伸長する場合は，IR領域に値を設定してください。

・Exitフォーマット伸長（２．５．４　Exif対応参照）

保守／廃止

表２－９　DJINFO構造体（AP705100-B03）

メンバ	型	内　　容	IN/OUT
ErrorState	int	エラー・ステータス	IN/OUT
FileSize	int	JPEGファイル・サイズ	OUT
Restart	unsigned short	リスタート・インターバル	OUT
Width	unsigned short	画像の横ピクセル数	OUT
Height	unsigned short	画像の縦ピクセル数	OUT
Sampling	char	サンプル比	OUT
Mode	char	伸長モード	IN
JPEG_Buff_Bptr	unsigned char＊	JPEGバッファ先頭アドレス	IN
JPEG_Buff_Eptr	unsigned char＊	JPEGバッファ先頭アドレス＋JPEGバッファ・サイズ	IN
IRAM_Buff_Bptr	int＊	内部RAMワーク・エリア先頭アドレス	IN
StartX	short	画像の開始ｘ位置（ピクセル数）	IN
StartY	short	画像の開始ｙ位置（ピクセル数）	IN
VRAM_Bptr	unsigned char＊	VRAM先頭アドレス	IN注１
VRAM_W_Pixel	short	VRAMの横幅（ピクセル数）	IN注２
VRAM_H_Pixel	short	VRAMの縦幅（ピクセル数）	IN注２
RAM_Line_Byte	int	VRAMの縦１ピクセル分のアドレス差	IN注１
VRAM_Pixel_Byte	int	VRAMの横１ピクセル分のアドレス差	IN注１
VRAM_Gap1_Byte	int	VRAMのＹとCbとの／ＲとＢとのアドレス差	IN注１
VRAM_Gap2_Byte	int	VRAMのＹとCrとの／ＲとＧとのアドレス差	IN注１
APP_Info_Bptr	APPINFO＊	APPINFO構造体先頭アドレス	IN
ClipSX	unsigned short	クリッピング開始位置（クリッピング・モード時だけ有効）	IN
ClipSY	unsigned short	クリッピング開始位置（クリッピング・モード時だけ有効）	IN
ClipW	unsigned short	クリッピング横幅（クリッピング・モード時だけ有効）	IN
ClipH	unsigned short	クリッピング縦幅（クリッピング・モード時だけ有効）	IN
Work	int＊	外部RAMワーク・エリア先頭アドレス	IN
CurrentX	short	VRAM描画用ワーク（内部使用）	－
CurrentY	short	VRAM描画用ワーク（内部使用）	－
IR	52 byte	内部予約（内部使用ワーク・エリア）	注３

注１． putmcu関数を自作する場合は設定不要です。

２． putmcu関数を自作する場合はダミーで設定してください。

★ **３．** IRメンバの領域は０クリアしてください。ただし，次のJPEGフォーマットで伸長する場合は，IR領域に値を設定してください。

・Exitフォーマット伸長（**２．５．４　Exif対応**参照）

保守／廃止

２.３.３　APPINFO構造体

圧縮でJPEGファイルにAPPnセグメントを埋め込む設定をする場合，基本伸長でAPPnセグメントに関する情報を得るときに，APPJINFO構造体が必要です（この構造体はAP70732-B03/AP705100-B03基本ライブラリで共通です）。

圧縮処理／伸長処理でAPPnセグメントに対応する場合には，このAPPINFO構造体を宣言し，その先頭アドレスをCJINFO構造体／DJINFO構造体のメンバAPPに登録してください。

表２－10　APPINFO構造体

メンバ	型	内　容
APP00_Buff_Bptr	unsigned char＊	APP0セグメントに埋め込むデータ・バッファのアドレス
APP01_Buff_Bptr	unsigned char＊	APP1セグメントに埋め込むデータ・バッファのアドレス
APP02_Buff_Bptr	unsigned char＊	APP2セグメントに埋め込むデータ・バッファのアドレス
APP03_Buff_Bptr	unsigned char＊	APP3セグメントに埋め込むデータ・バッファのアドレス
APP04_Buff_Bptr	unsigned char＊	APP4セグメントに埋め込むデータ・バッファのアドレス
APP05_Buff_Bptr	unsigned char＊	APP5セグメントに埋め込むデータ・バッファのアドレス
APP06_Buff_Bptr	unsigned char＊	APP6セグメントに埋め込むデータ・バッファのアドレス
APP07_Buff_Bptr	unsigned char＊	APP7セグメントに埋め込むデータ・バッファのアドレス
APP08_Buff_Bptr	unsigned char＊	APP8セグメントに埋め込むデータ・バッファのアドレス
APP09_Buff_Bptr	unsigned char＊	APP9セグメントに埋め込むデータ・バッファのアドレス
APP10_Buff_Bptr	unsigned char＊	APPAセグメントに埋め込むデータ・バッファのアドレス
APP11_Buff_Bptr	unsigned char＊	APPBセグメントに埋め込むデータ・バッファのアドレス
APP12_Buff_Bptr	unsigned char＊	APPCセグメントに埋め込むデータ・バッファのアドレス
APP13_Buff_Bptr	unsigned char＊	APPDセグメントに埋め込むデータ・バッファのアドレス
APP14_Buff_Bptr	unsigned char＊	APPEセグメントに埋め込むデータ・バッファのアドレス
APP15_Buff_Bptr	unsigned char＊	APPFセグメントに埋め込むデータ・バッファのアドレス
APP00_BuffSize	short	APP0セグメントに埋め込むデータ長（バイト数）
APP01_BuffSize	short	APP1セグメントに埋め込むデータ長（バイト数）
APP02_BuffSize	short	APP2セグメントに埋め込むデータ長（バイト数）
APP03_BuffSize	short	APP3セグメントに埋め込むデータ長（バイト数）
APP04_BuffSize	short	APP4セグメントに埋め込むデータ長（バイト数）
APP05_BuffSize	short	APP5セグメントに埋め込むデータ長（バイト数）
APP06_BuffSize	short	APP6セグメントに埋め込むデータ長（バイト数）
APP07_BuffSize	short	APP7セグメントに埋め込むデータ長（バイト数）
APP08_BuffSize	short	APP8セグメントに埋め込むデータ長（バイト数）
APP09_BuffSize	short	APP9セグメントに埋め込むデータ長（バイト数）
APP10_BuffSize	short	APPAセグメントに埋め込むデータ長（バイト数）
APP11_BuffSize	short	APPBセグメントに埋め込むデータ長（バイト数）
APP12_BuffSize	short	APPCセグメントに埋め込むデータ長（バイト数）
APP13_BuffSize	short	APPDセグメントに埋め込むデータ長（バイト数）
APP14_BuffSize	short	APPEセグメントに埋め込むデータ長（バイト数）
APP15_BuffSize	short	APPFセグメントに埋め込むデータ長（バイト数）

保守／廃止

2.3.4　MCUバッファ

JPEG処理の最小単位をMCU（Minimum Coded Unit）と呼んでいます。

基本ライブラリでは，この単位ごとに画像を圧縮／伸長する際に画像データ（中間的な画像データから最終的な画像データまで）を格納するバッファ（MCUバッファ）を必要とします。

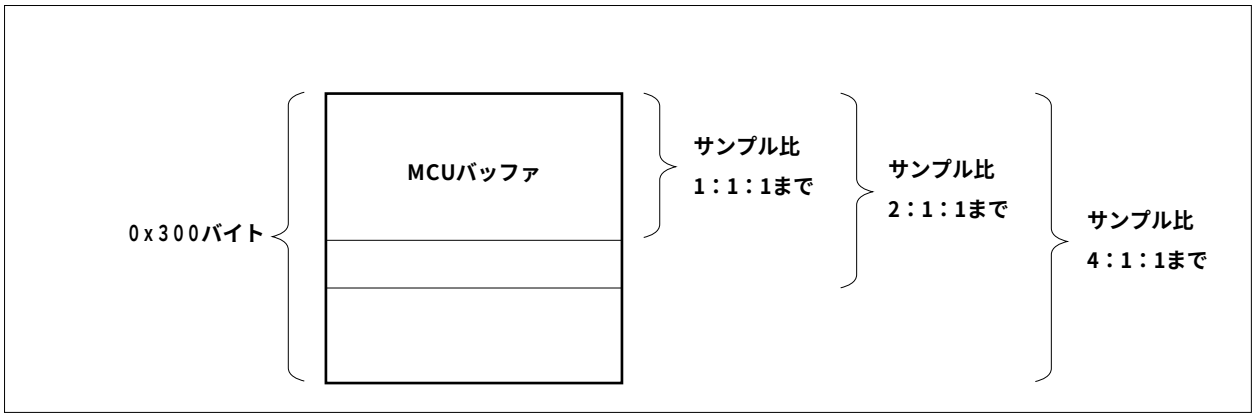
このMCUバッファは，AP70732-B03ではCJINFO構造体／DJINFO構造体の最後のメンバに割り付けてあります。AP705100-B03では内蔵データRAMワーク・エリアの先頭＋0x100バイト以降です。

必要なMCUバッファのサイズは，次に示すとおりです。

表２－11　MCUバッファのサイズ

対応させるサンプル比	必要サイズ
4：1：1	0x300バイト
2：1：1	0x200バイト
1：1：1	0x180バイト

図２－８　MCUバッファの使用方法（AP70732-B03）

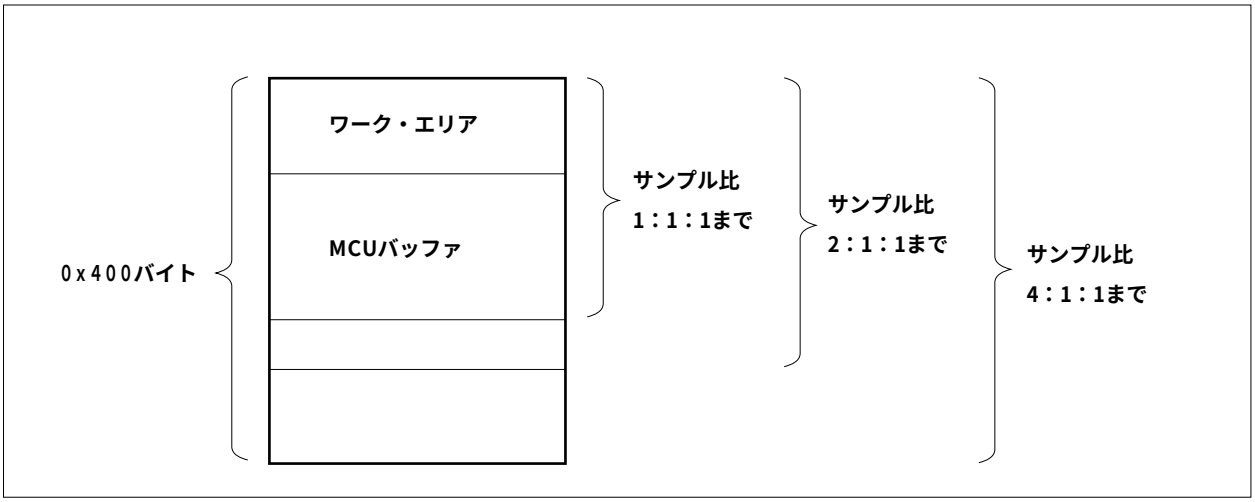


圧縮でサンプル比２：１：１までしか使わないことが保証されているような場合には，JPEG.Hの構造体定義の部分を直接書き換えることで（MCUバッファのサイズ = 0x200バイト），構造体が占めるメモリ・サイズを削減できます。

ただし，伸長しようとしたJPEGファイルが４：１：１であった場合，４：１：１の伸長ライブラリがリンクされていると，MCUバッファが0x300バイトあると認識し，警告なしに先頭から0x300バイトを上書きします。

保守／廃止

図２－９　内部RAMワーク・エリアの使用方法（AP705100-B03）



伸長しようとしたJPEGファイルが４：１：１であった場合，４：１：１の伸長ライブラリがリンクされていると，内部RAMワーク・エリアの先頭から0x400バイトを警告なしに上書きします。

2.3.5　JPEGバッファ

JPEGバッファはJPEGファイルを格納するための領域です。このバッファは１バイト以上，任意のバイト数に設定可能ですが，圧縮でバッファがいっぱいになったときや，伸長でバッファの終わりまで来たとき，基本ライブラリは処理を中断し，必要なレジスタを保存して，復帰すべきレジスタの値を元に戻す処理（レジスタ・ディスパッチ）を行います。この回数があまり多くなると，全体の処理時間に影響してきます。ある程度の領域を確保し，JPEGバッファとしてください。

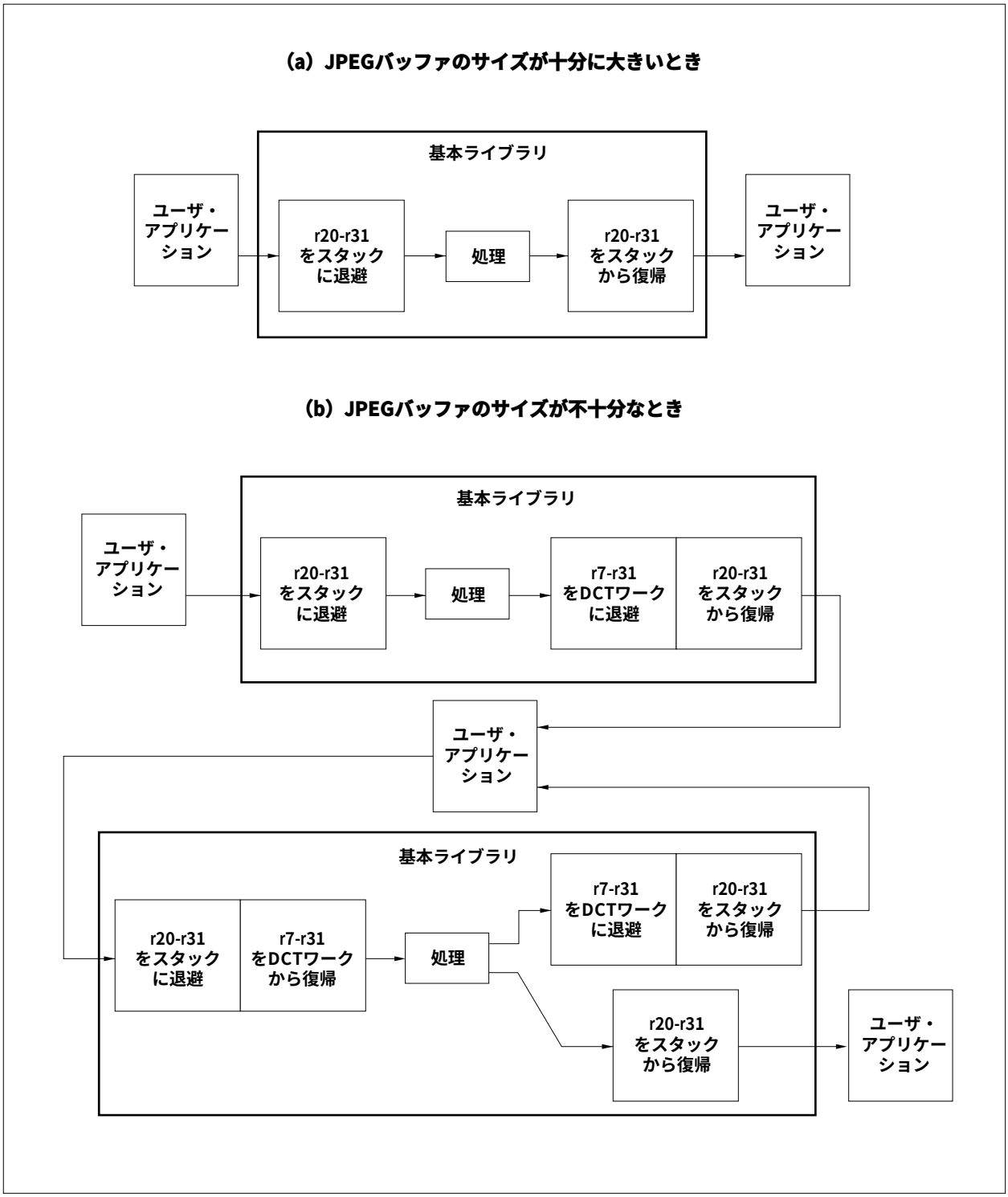
2.3.6　レジスタ・ディスパッチ

基本ライブラリでは，次の場合に処理を中断し，ユーザ・アプリケーション側に制御を移します。その際に，基本ライブラリの使用しているレジスタの内容をメモリに保管し，Ｃ言語規約で保存が約束されているレジスタ（sp, r20-r29）の内容を復帰します。

レジスタ・ディスパッチが発生する場合は，次に示すとおりです。

- 圧縮で，JPEGバッファがいっぱいになった場合
- 伸長で，JPEGバッファの最後までデコードした場合
- 画像数行ごとに処理を中断するような選択をした場合で中断すべき行に達したとき

図２－10　レジスタ・ディスパッチ



保守／廃止

２．４　圧縮処理の実行

圧縮処理では，画像データを圧縮し，JPEGファイルを生成します。

２．４．１　圧縮メイン関数

分　　類　圧縮処理系

関 数 名　jpeg_Compress

機能概要　JPEG圧縮処理

形　　式　#include “ jpeg.h ”

int jpeg_Compress（CJINFO＊　cJpeginfo）

引 き 数　CJINFO構造体の先頭アドレス

返 り 値　これらの返り値は数値です。

Ｃ言語で#define　JPEG_OK ０のように定義してあるものです。

表２－12　圧縮処理関数の返り値

返り値	内　　容
JPEG_OK	正常終了
JPEG_ERR	エラー終了
JPEG_CONT1	JPEGバッファによる中断
JPEG_CONT2	VRAMによる中断

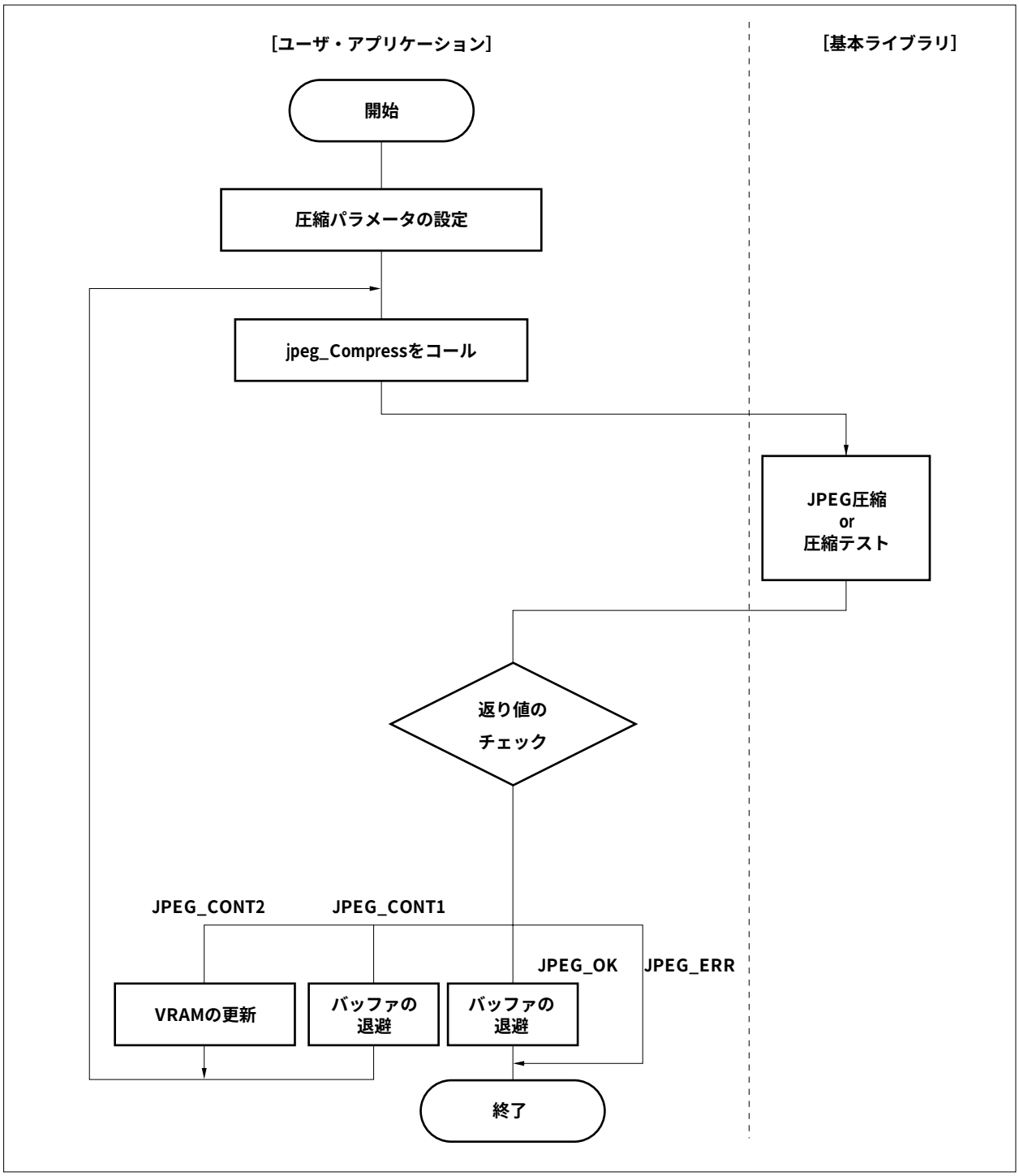
備考　JPEG_ERRの場合は，CJINFO構造体のメンバ“ ErrorState ”にエラー・ステートメントが格納されます。

保守／廃止

２．４．２　圧縮処理フロー

圧縮処理の処理の流れを次に示します。

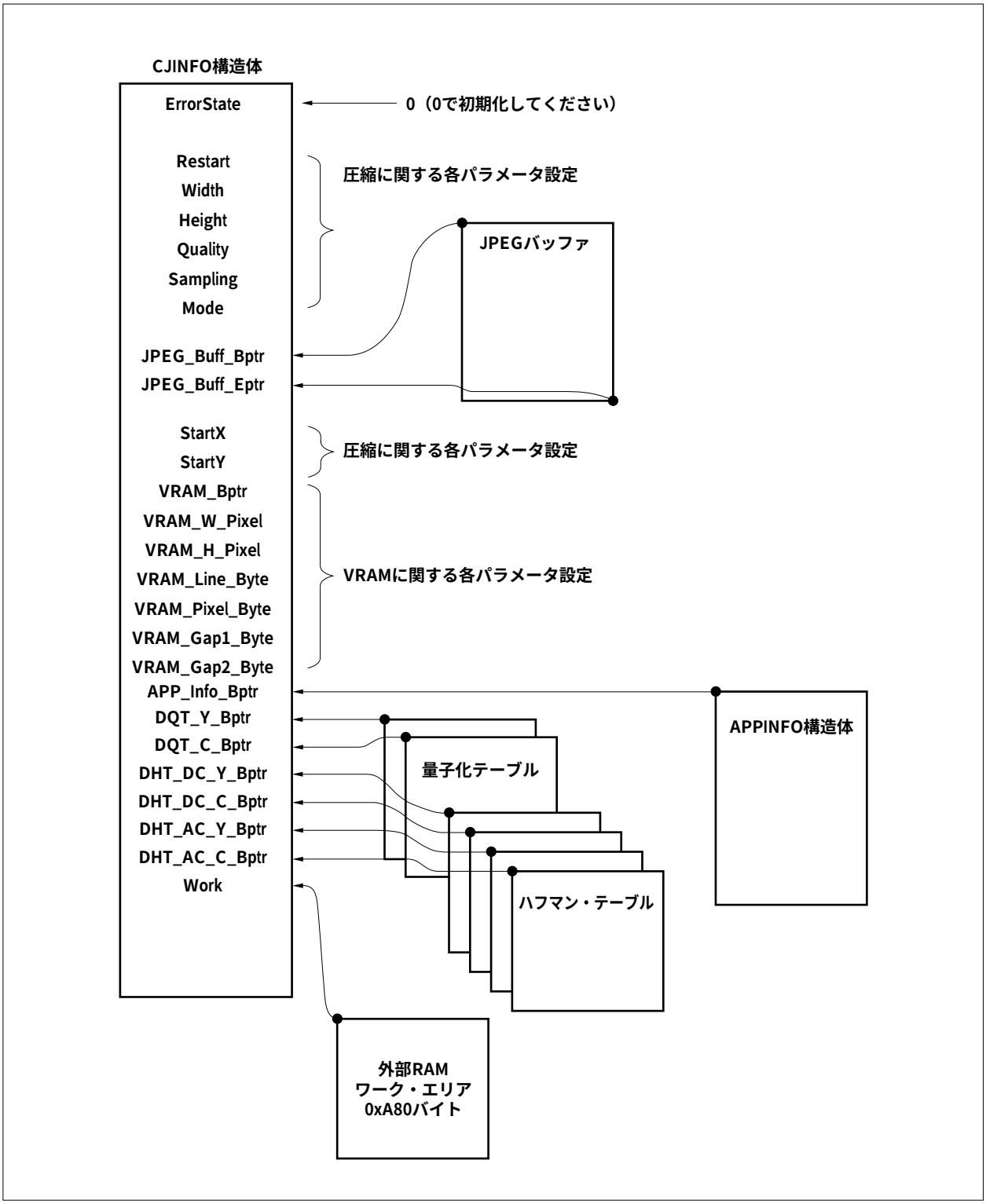
図２－11　圧縮処理フロー



保守／廃止

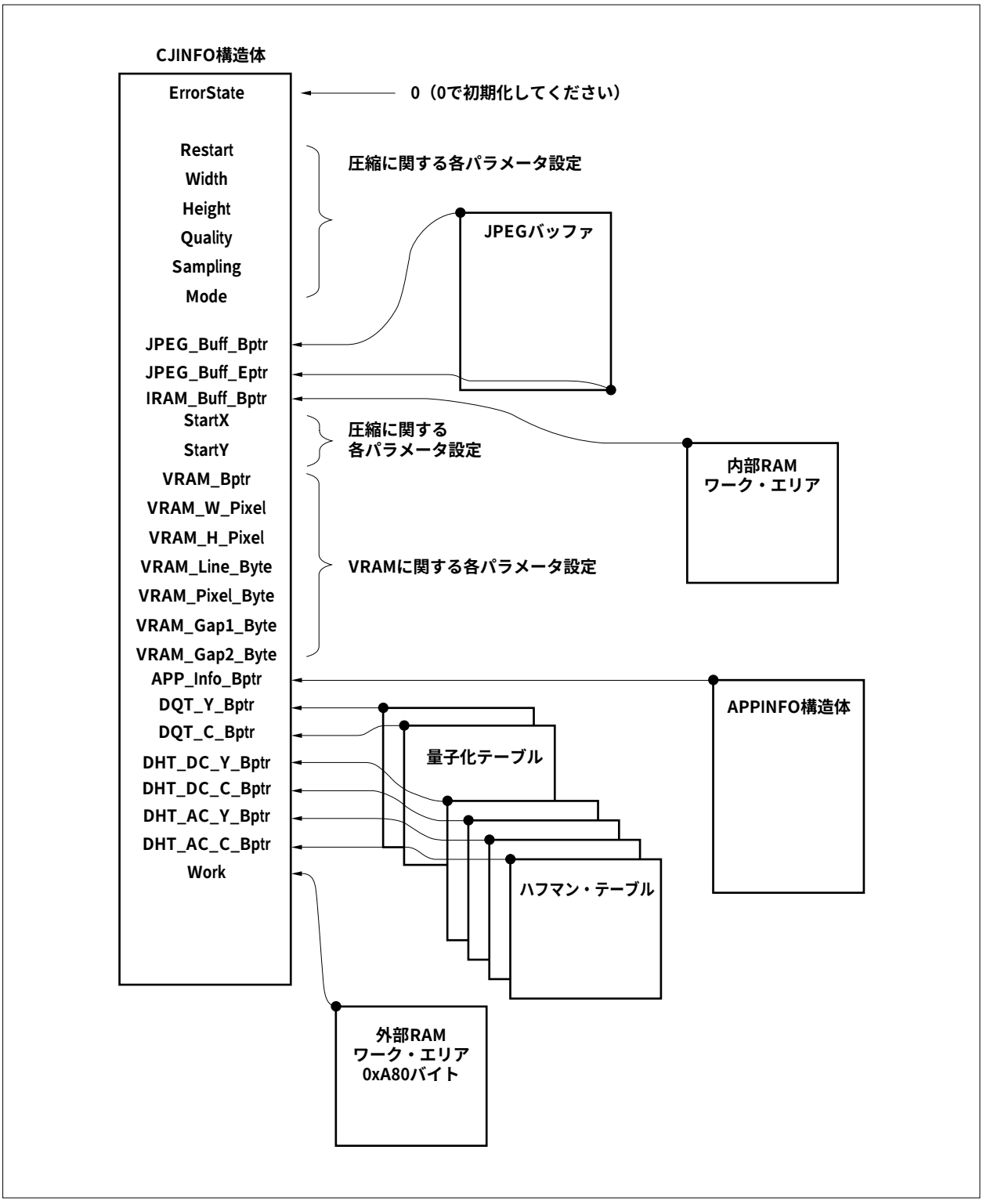
２．４．３　CJINFO構造体のパラメータ設定

図２－12　CJINFO構造体のパラメータ設定（AP70732-B03）



保守／廃止

図２－13　CJINFO構造体のパラメータ設定（AP705100-B03）



保守／廃止

（1）エラー・ステータスの初期化（ErrorState）

圧縮ルーチンを起動する前の圧縮パラメータ設定時に一度だけ０で初期化してください。

設定値：０

注意　処理を中断したあと，再開する場合には，基本ライブラリがこのErrorStateの値を見て，一番最初の起動か，それとも再開かを判別するため，これ以外の初期化はしないでください（処理が中断した場合は，再開する場所のアドレスが格納されています）。

（2）リスタート・インターバル（Restart）

リスタート・インターバルの詳細については，**1.2.1　JPEGの概要（7）リスタート・マーカ**を参照してください。

設定値：0-65535

０を指定した場合には，DRIセグメント／RSTnマーカは挿入されません。０以外の値を指定した場合は，その値がリスタート・インターバルとなります。

表２－13　リスタート・インターバルの設定

設定値	処　　理
0	DRIセグメントおよびRSTnマーカはJPEGファイルに付加されません。
1-65535	設定値をリスタート・インターバルとし，この個数のMCUごとにRSTnマーカを挿入します。

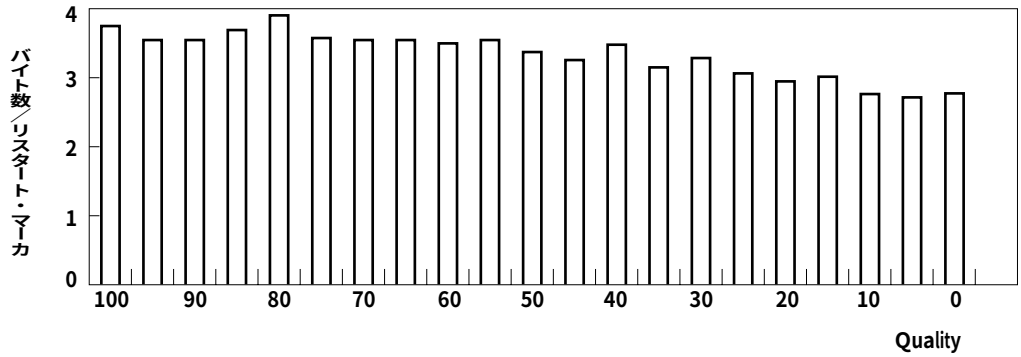
リスタート・インターバルを有効にした場合には，RSTnマーカによりJPEGファイルのサイズが増えます。リスタート・マーカ１個で約４バイト弱です。ファイル・サイズの概算値を知りたい場合には，この値に，１ファイルに含まれるRSTnの個数を掛けたものをリスタート・マーカなしで圧縮したファイル・サイズに加算してください。

次にRSTnマーカ１個あたりの平均バイト数を示します。

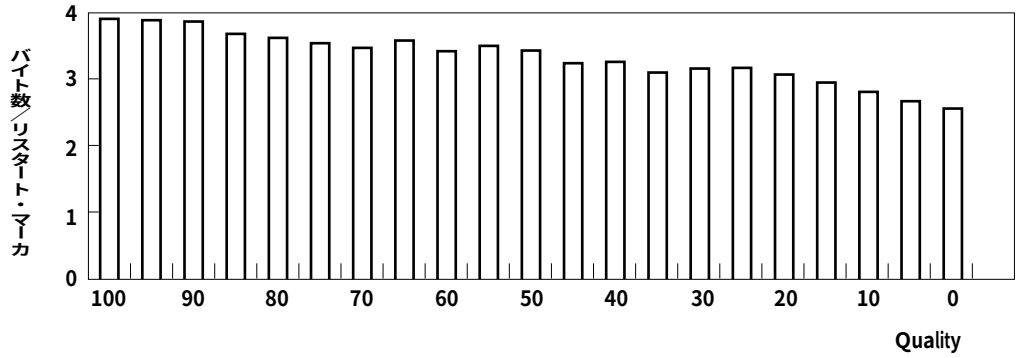
保守／廃止

図２－14　量子化パラメータとリスタート・マーカ１個あたりのバイト数

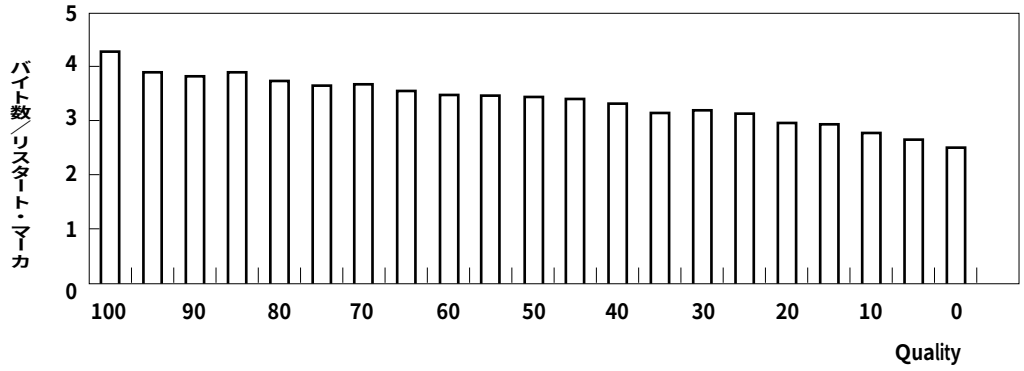
（a）サンプル比4：1：1



（b）サンプル比2：1：1



（c）サンプル比1：1：1



保守／廃止

（３）画像の横サイズ（Width）と画像の縦サイズ（Height）

設定値：0-65535

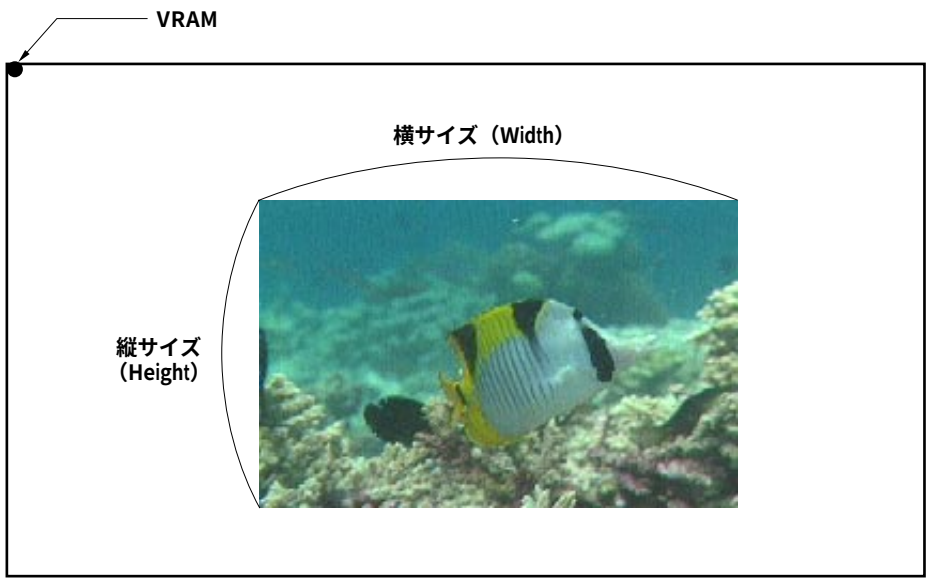
値の単位はピクセル数です。

ただし，設定できる値には次のような制限があります。

表２－14　横サイズ／縦サイズの制限

サンプル比	横サイズ（Width）	縦サイズ（Height）
4：1：1（H：V＝2：2）	16の倍数	16の倍数
4：1：1（H：V＝4：1）	32の倍数	８の倍数
2：1：1（H：V＝2：1）	16の倍数	８の倍数
1：1：1（H：V＝1：1）	８の倍数	８の倍数

図２－15　画像の横サイズ／縦サイズ



保守／廃止

（４）量子化パラメータ

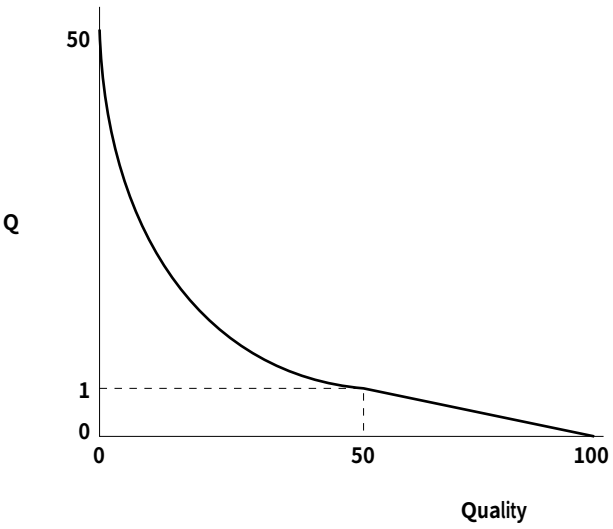
基本ライブラリでは，量子化テーブルの値を簡単に変更できるように，量子化パラメータ（Quality）を用意しています。

基本ライブラリ内部でこのQualityパラメータから定数“Ｑ”を求めます。次の式から計算し，量子化テーブルの各要素に“Ｑ”を掛けて1-255の範囲に丸めたものを量子化係数とし，実際の量子化処理に使用します。

Quality＜50の場合：Q = Quality/50

Quality≧50の場合：Q = 2－Quality/50

図２－16　量子化パラメータQualityと定数Q



ディフォールトの量子化テーブルを加工せずにそのまま使用するには，Qualityパラメータの値に“50”を設定してください。

表２－15　Qualityパラメータの設定

Qualityパラメータ	100	…	50	…	0
定数Q	0	…	1	…	50
量子化テーブル	すべての要素が１	…	ディフォールトのまま	…	ほとんどすべての要素が255
画質	高品質	…			低品質
JPEGファイルのサイズ	大	…			小

保守／廃止

ディフォールトの量子化テーブルLuminanceQtbl／ChrominanceQtblに対して，Qualityに100/75を指定した場合には次のような量子化テーブルが生成され，実際の圧縮処理（量子化）で使用されます。

（ａ）Quality = 100にした場合

輝度成分の量子化テーブル

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

色差成分の量子化テーブル

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）Quality = 75にした場合

輝度成分の量子化テーブル

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

色差成分の量子化テーブル

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

Qualityパラメータを変えてJPEG圧縮したときの見た目の違いは，図２－17のようになります。

〔メ　モ〕

保守／廃止

図２－17　量子化パラメータの違いによる画像品質（1/2）

元絵
224×144×24

（a）Quality = 100，サンプル比4：1：1
ファイル・サイズ　35714バイト

（b）Quality = 90，サンプル比4：1：1
ファイル・サイズ　10424バイト

（c）Quality = 80，サンプル比4：1：1
ファイル・サイズ　6615バイト

（d）Quality = 70，サンプル比4：1：1
ファイル・サイズ　5195バイト

（e）Quality = 60，サンプル比4：1：1
ファイル・サイズ　4364バイト

保守／廃止

図２－17　量子化パラメータの違いによる画像品質（2/2）

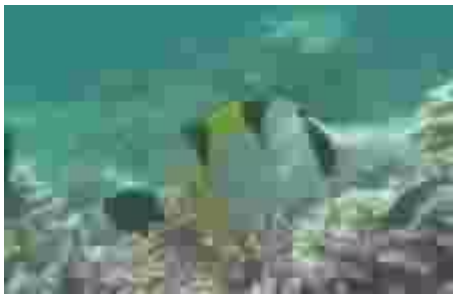
(f)　Quality = 50，サンプル比4：1：1
ファイル・サイズ　3869バイト

(g)　Quality = 40，サンプル比4：1：1
ファイル・サイズ　3388バイト

(h)　Quality = 30，サンプル比4：1：1
ファイル・サイズ　2915バイト

(i)　Quality = 20，サンプル比4：1：1
ファイル・サイズ　2335バイト

(j)　Quality = 10，サンプル比4：1：1
ファイル・サイズ　1701バイト



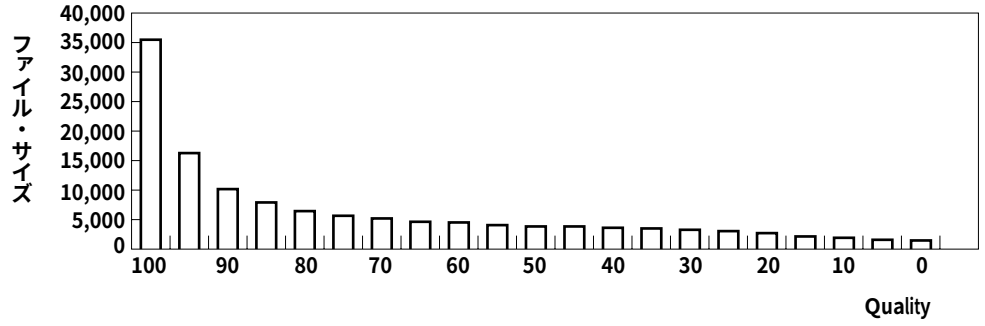
(k)　Quality = 0，サンプル比4：1：1
ファイル・サイズ　1228バイト

保守／廃止

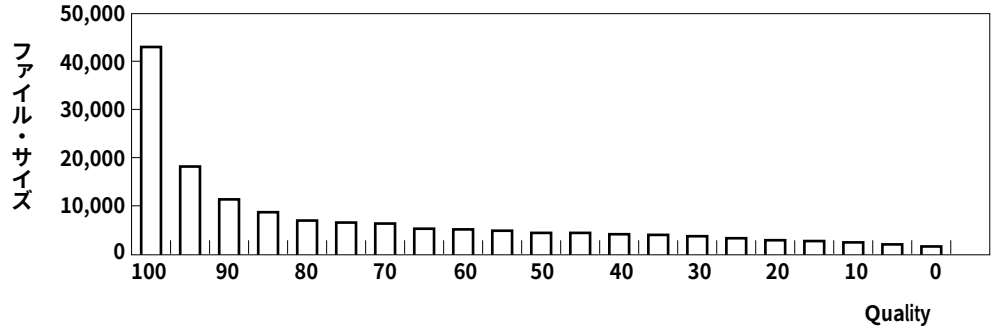
量子化パラメータとファイル・サイズの関係は図２－18のようになります。

図２－18　量子化パラメータとファイル・サイズ

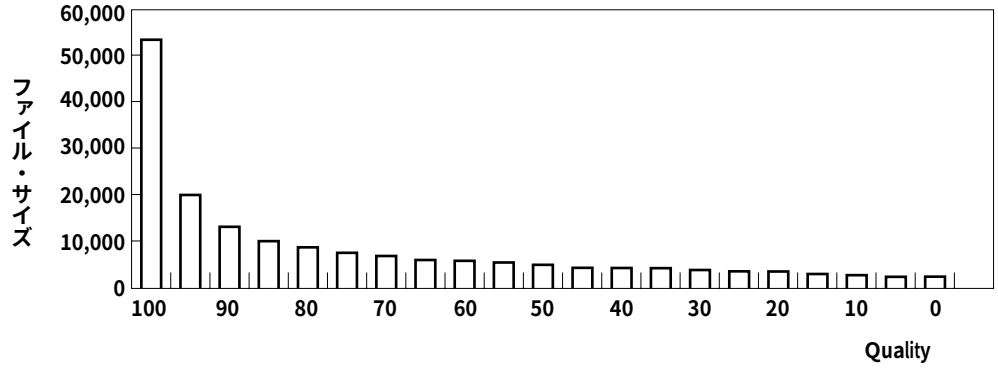
（a）サンプル比4：1：1



（b）サンプル比2：1：1



（c）サンプル比1：1：1



保守／廃止

（５）サンプル比の選択（Sampling）

サンプル比については１．２．１　JPEGの概要　（３）サンプリングとMCUのサンプル比に関する部分を参照してください。

基本ライブラリがサポートしているサンプル比は，次の４種類です。

表２－16　メンバSamplingの設定値

サンプル比	基本ライブラリ内での識別
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

これらの値は，ファイルjpeg.hの中で定義されています。

表２－17　サンプル比の設定

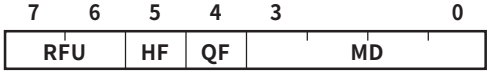
サンプル比	4：1：1	2：1：1	1：1：1
色彩	普通	⟷	鮮明
ファイル・サイズ	基準値	約4/3倍	約２倍

輝度成分についてはどのサンプル比を選択しても，サンプル比による画像の違いはありません。サンプル比は色差成分の画質に影響します。

（６）モード（Mode）

設定する値によって，次のように選択されます。

表２－18　メンバModeの設定値



ビット位置	ビット名	意　味
7-6	RFU	予約フィールド[注]
5	HF	ハフマン・テーブル・イニシャライズ用フラグ[注] ０：イニシャライズを行う １：イニシャライズを行わない
4	QF	量子化テーブル・イニシャライズ用フラグ[注] ０：イニシャライズを行う １：イニシャライズを行わない
3-0	MD	モード ０：圧縮テスト・モード １：通常圧縮モード

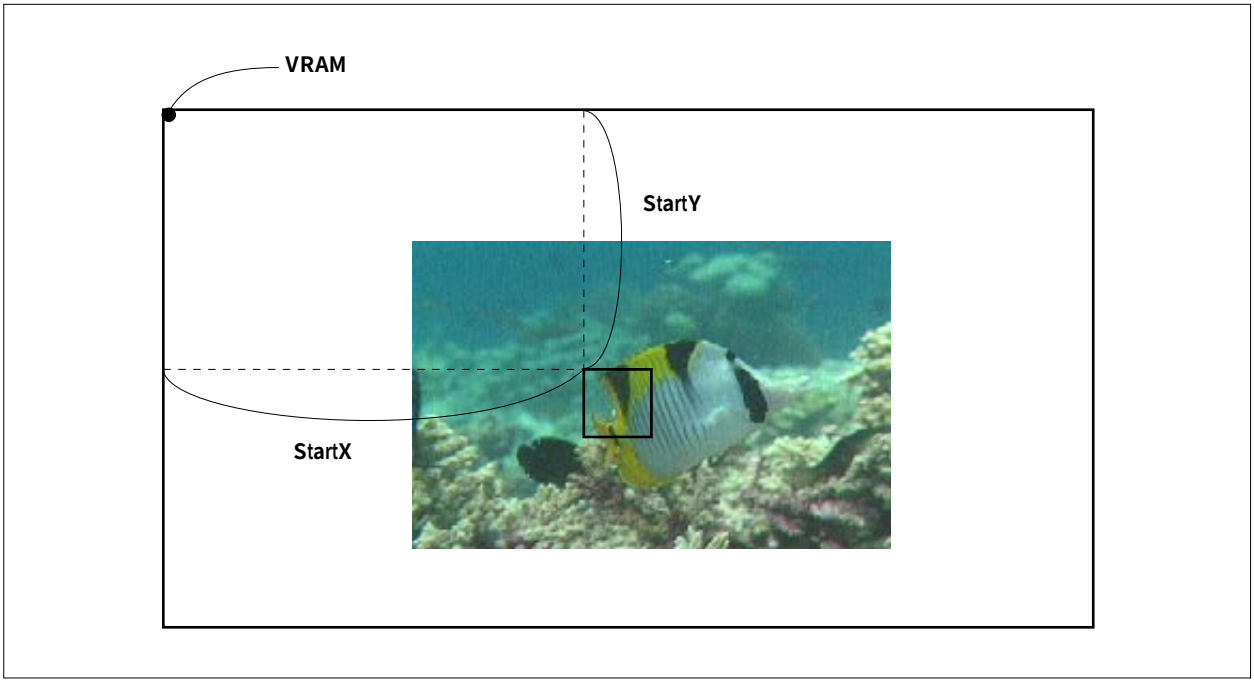
注　Ver.2.10から追加

（a）モード０

1MCU分の圧縮したデータのビット数をテストするモードです。ビット数はメンバFileSizeに得られます。

MCUの位置をずらすには，メンバStartX, StartYの値を変えてください。

図２－19　圧縮テスト位置の調整



（b）モード１

通常の圧縮処理を行います。

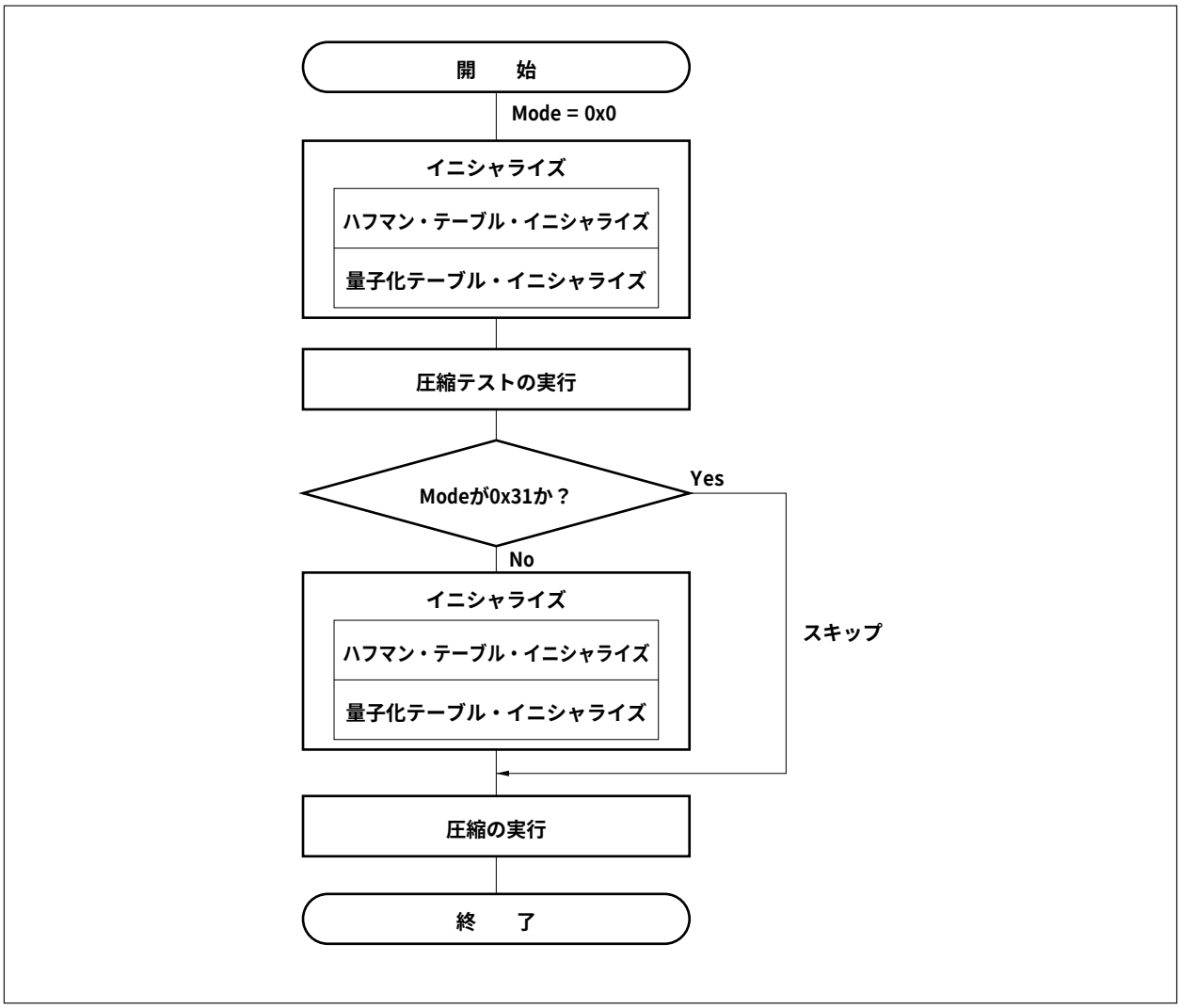
構造体のメンバModeに設定される2688バイトの領域は基本ライブラリが内部使用する量子化用のルックアップ・テーブルとハフマン用のルックアップ・テーブルで占められます。

一度圧縮（これらのルックアップ・テーブルのイニシャライズ）を行ったWorkエリアをそのまま使う場合は，二回目以降のイニシャライズを省略できます。

構造体のメンバModeのHFビットが１だった場合，ハフマン用のルックアップ・テーブル作成が抑制されます。同様にQFビットが１の場合に量子化用ルックアップ・テーブルは更新されません。

使用例として，図２－20では仮圧縮モードでルックアップ・テーブルのイニシャライズを行い，続けて同じ量子化パラメータ，ハフマン・テーブルで本圧縮をする際にイニシャライズを省略しています。

図２－20　圧縮モード設定の追加



（7）JPEGバッファの先頭アドレス（JPEG_Buff_Bptr）と終了アドレス（JPEG_Buff_Eptr）

JPEGバッファの先頭アドレスと終了アドレスを設定してください。

表２－19　メンバJPEG _Buff_Bptr/JPEG_Buff_Eptrの設定値

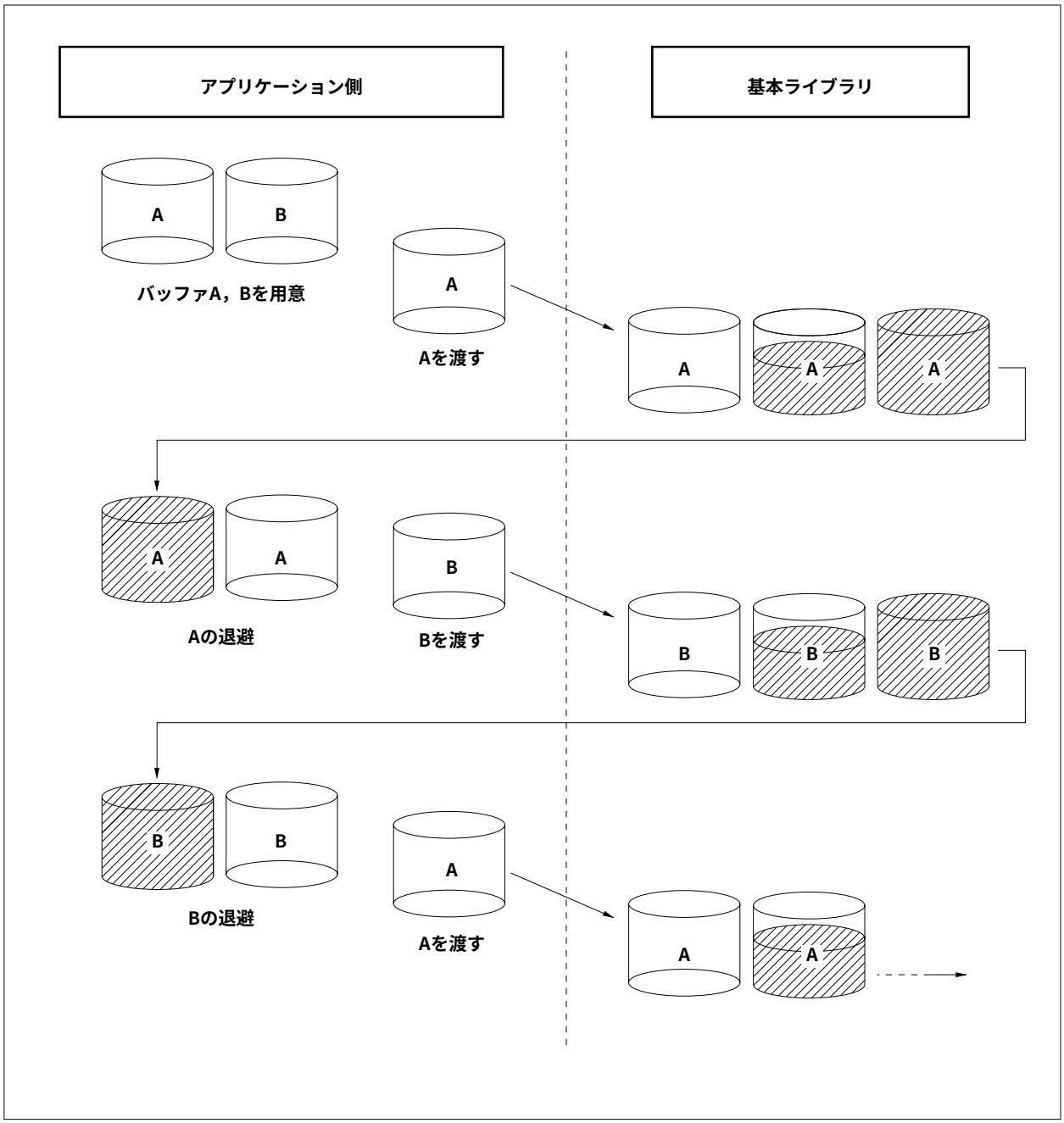
メンバ	内　容
JPEG_Buff_Bptr	JPEGバッファの先頭アドレス
JPEG_Buff_Eptr	JPEGバッファ先頭アドレス ＋JPEGバッファ・サイズ

JPEGバッファ・サイズの制限により処理途中でバッファ退避処理が入るような場合には，JPEGバッファを２面もって，それらを切り替えながら動かすこともできます。

JPEGバッファについては，**2.3.5　JPEGバッファ**，**2.3.6　レジスタ・ディスパッチ**を参照してください。

保守／廃止

図２－21　２面のJPEGバッファの切り替え



保守／廃止

（８）内蔵RAMワーク・エリア・アドレス（IRAM_Buff_Bptr：AP705100-B03）

内蔵RAMワーク・エリアの詳細については，２.３.４　MCUバッファを参照してください。

表２－20　メンバIRAM_Buff_Bptrの設定値

メンバ	内　　容
IRAM_Buff_Bptr	内蔵RAMワーク・エリアの先頭アドレス

サンプル比４：１：１／２：１：１／１：１：１によって，それぞれ先頭から0x400/0x300/0x280バイトを無条件に上書きします。

なお，AP70732-B03（V810ファミリ版）ではこのメンバの設定は必要ありません。

表２－21　サンプル比と必要な内蔵RAMワーク・エリアのサイズ

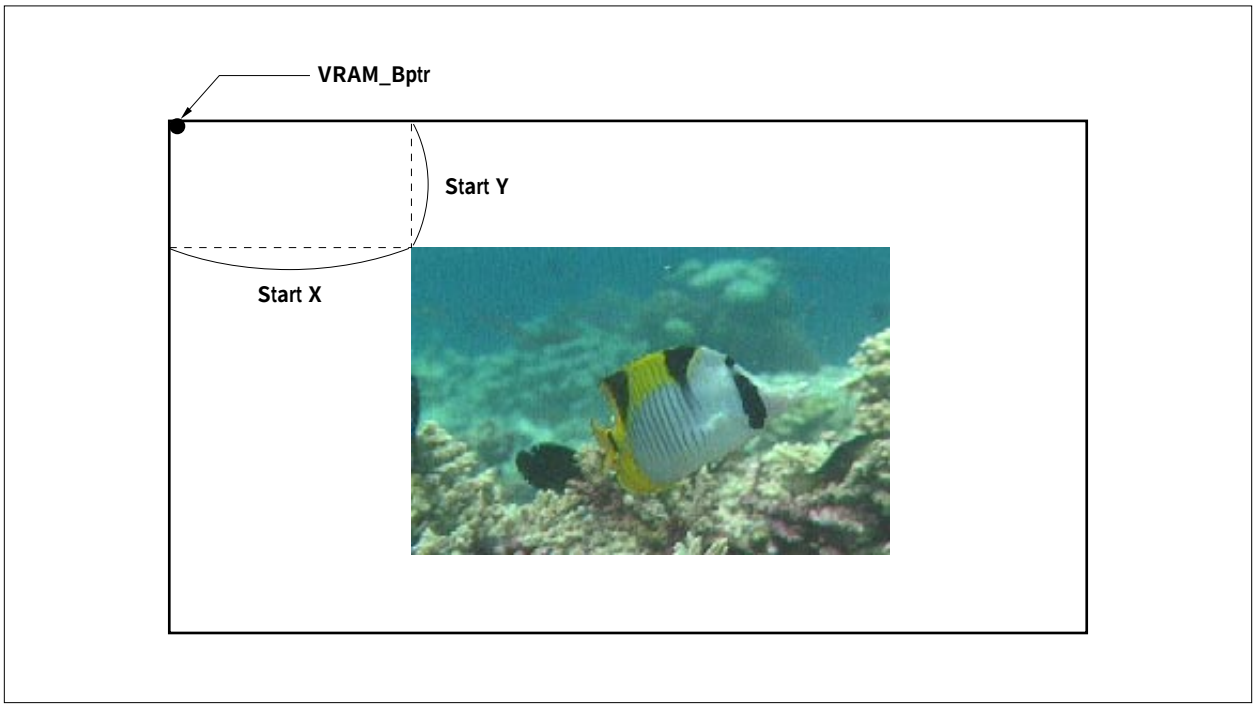
サンプル比	必要な内蔵RAMワーク・エリアのサイズ
4：1：1（H：V＝2：2）	0x400バイト
4：1：1（H：V＝4：1）	0x400バイト
2：1：1（H：V＝2：1）	0x300バイト
1：1：1（H：V＝1：1）	0x280バイト

（９）画像の開始ｘ位置（StartX）と画像の開始ｙ位置（StartY）

設定値：－32768～32767

値の単位はピクセル数です。

図２－22　画像の開始位置（ｘ，ｙ）



保守／廃止

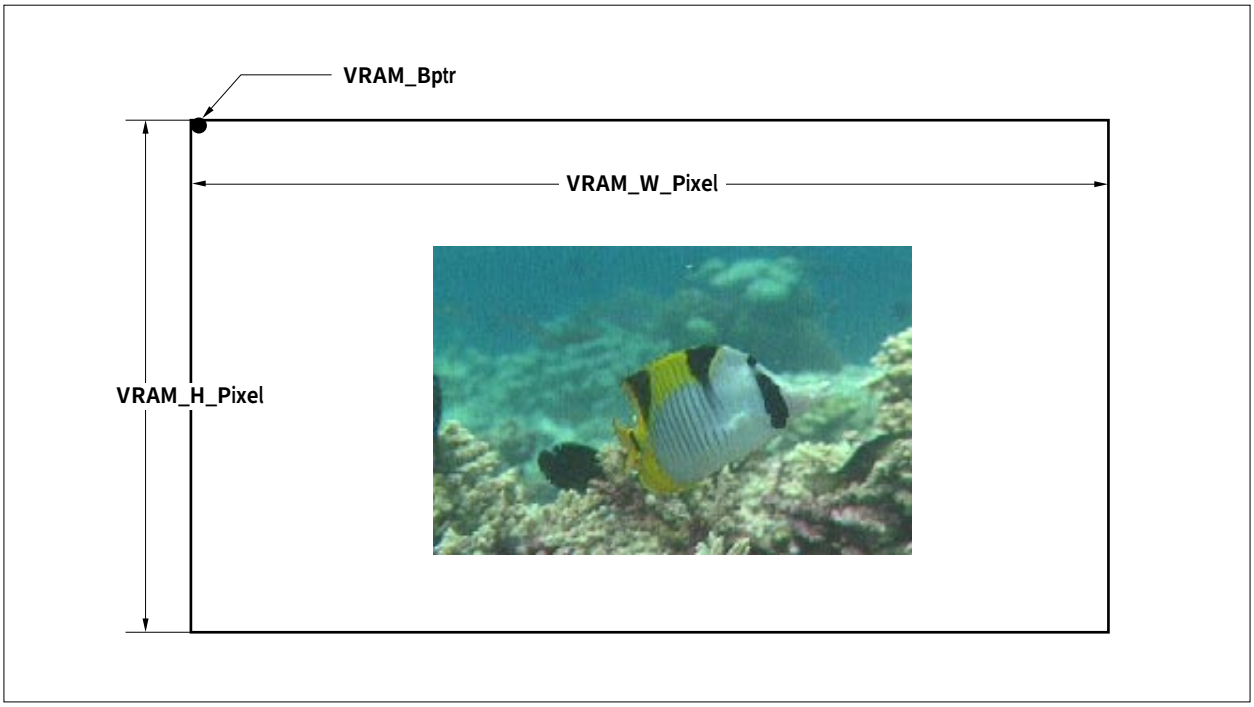
（10）VRAMサイズ

VRAM関係のメンバを設定してください。

表２－22　VRAM関係のメンバ設定値

メンバ	内　　容
VRAM_Bptr	VRAM先頭アドレス（基点アドレス）
VRAM_W_Pixel	VRAMの横ピクセル数
VRAM_H_Pixel	VRAMの縦ピクセル数

図２－23　VRAMサイズ



圧縮ライブラリを実行すると，ヘッダ部分を作成する際に次の２点をチェックします。

- VRAM_W_Pixelと（StartX＋Width）の大小関係
- VRAM_H_Pixelと（StartY＋Height）の大小関係

VRAMアクセス部分をカスタマイズ（後述）する場合でも，VRAM_W_Pixel，VRAM_H_Pixelのメンバは値を設定してください（チェック・ルーチンがError終了しないようなサイズを指定してください）。なお，カスタマイズする場合は，VRAM_Bptrの値は不定値で構いません。

保守／廃止

（11）VRAM構成

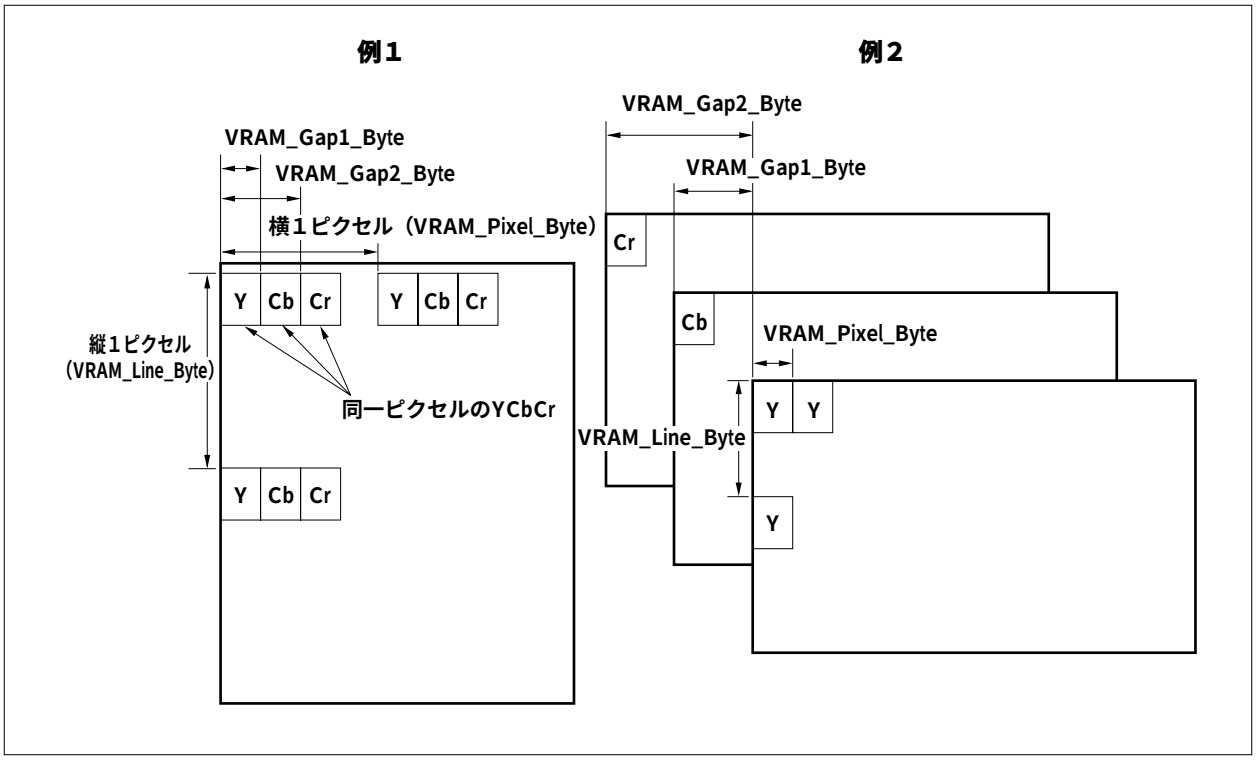
次の値は，ディフォールトのVRAMアクセス・ルーチンを使用する際に参照されます。

ディフォールトのVRAMアクセス・ルーチンでは，VRAMをY/Cb/CrあるいはR/G/Bのそれぞれが256階調（１バイト分）の深みを持つもので，LD.B命令/ST.B命令でアクセス可能なものと想定しています。

表２－23　VRAM構成に関するメンバ設定値

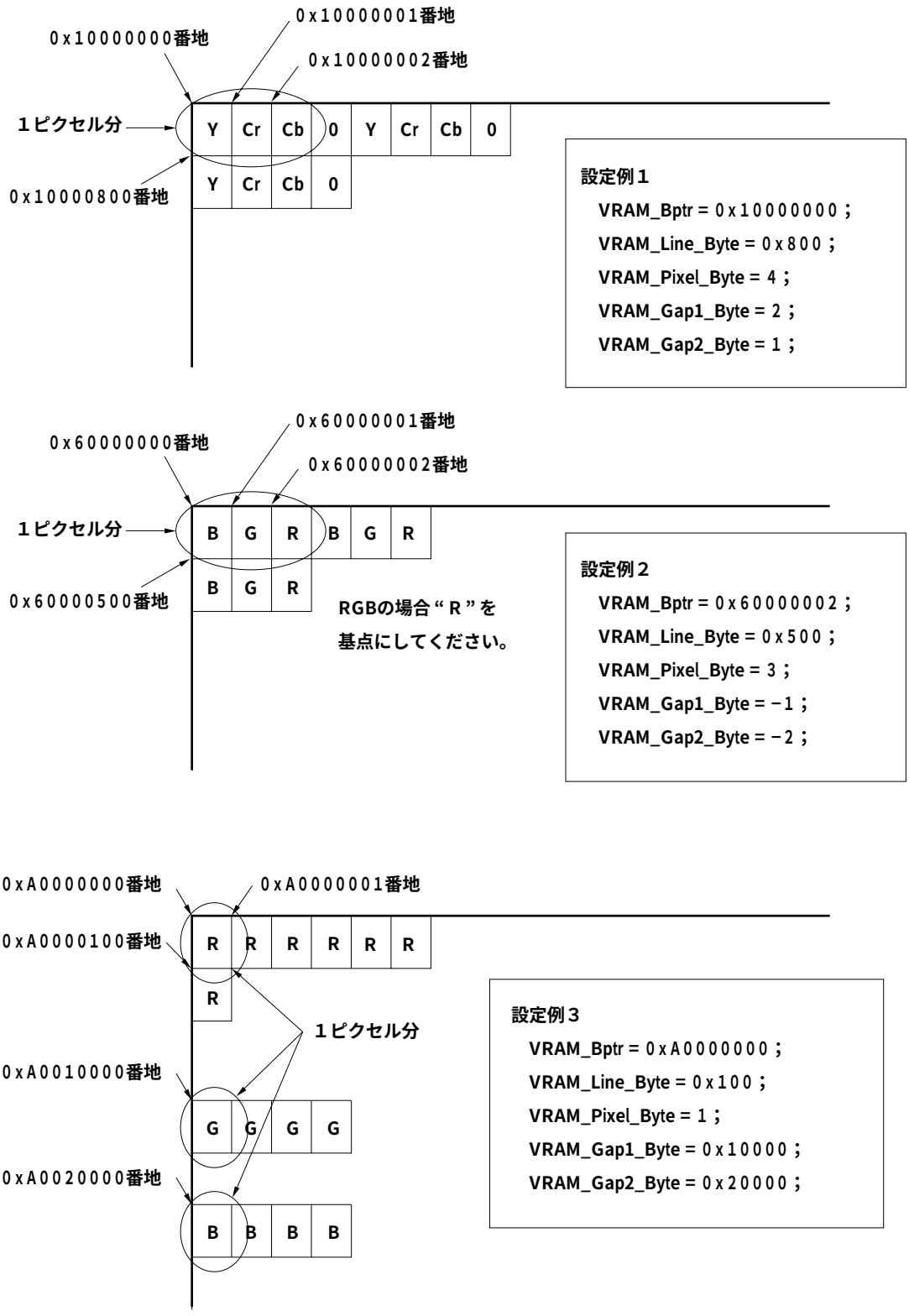
メンバ	内　　容
VRAM_Line_Byte	縦１ピクセル分のVRAMのアドレス差
VRAM_Pixel_Byte	横１ピクセル分のVRAMのアドレス差
VRAM_Gap1_Byte	VRAMがYCbCrの場合 同一ピクセルのＹとCbとのアドレス差 VRAMがRGBの場合 同一ピクセルのＲとＧとのアドレス差
VRAM_Gap2_Byte	VRAMがYCbCrの場合 同一ピクセルのＹとCrとのアドレス差 VRAMがRGBの場合 同一ピクセルのＲとＢとのアドレス差

図２－24　VRAM構成



保守／廃止

図２－25　基本ライブラリのVRAM関連メンバ設定例



保守／廃止

（12）APPINFOテーブルの指定（APP_Info_Bptr）

基本ライブラリでは，アプリケーション・データ・セグメントの埋め込みが指定できるようになっています。APPnセグメントにデータを埋め込まないときは，APP_Info_Bptrに０を設定してください。このとき，APPINFO構造体は不要です。

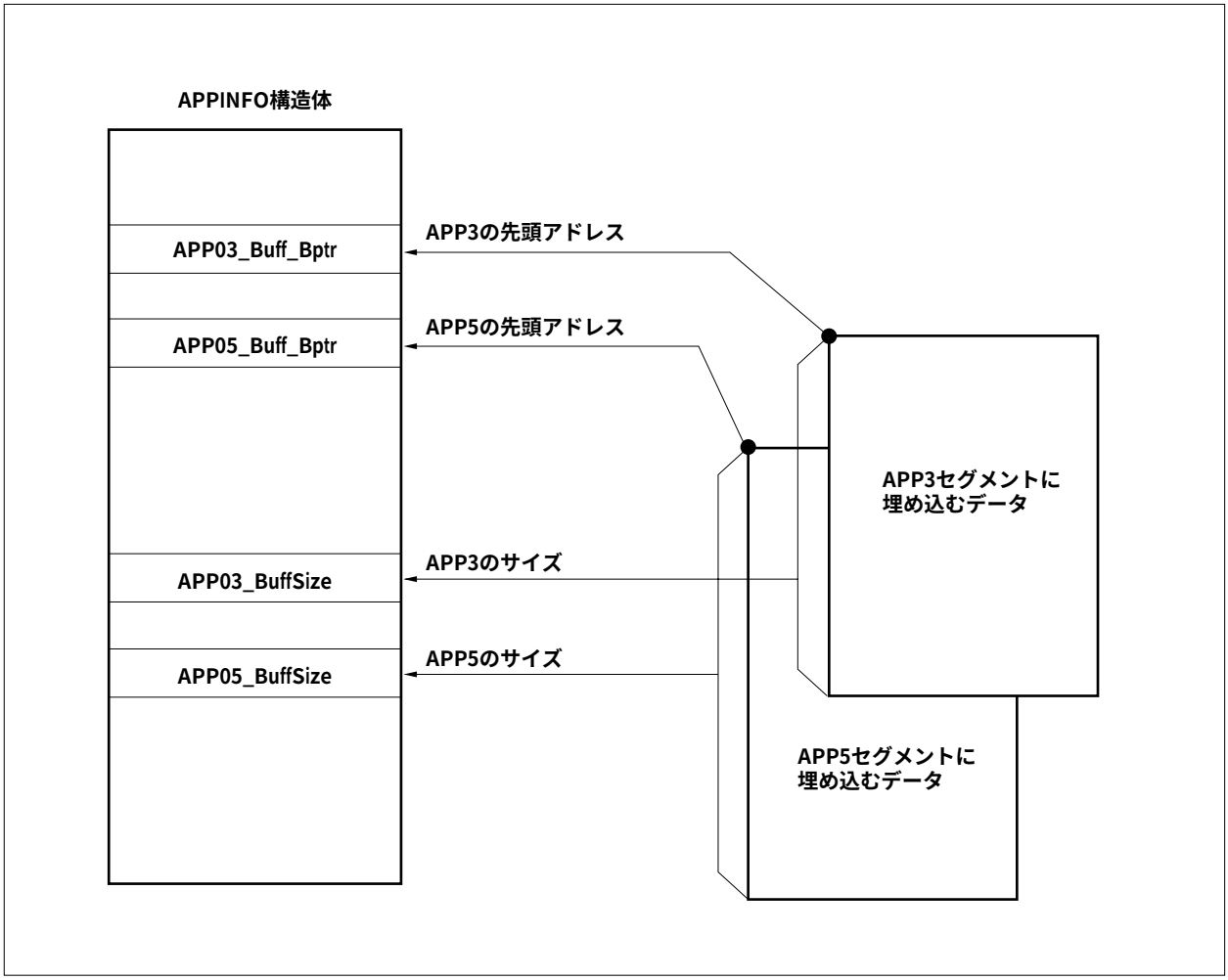
表２－24　メンバAPP_Info_Bptrの設定値

メンバ	設定値
APP_Info_Bptr	0　：APPnセグメントを埋め込まない
	APPINFO構造体の先頭アドレス　：APPnセグメントを埋め込む

注意　APPINFO構造体を０番地に置いた場合は，APPINFO構造体が設定されていないとみなします。

APPnセグメントを埋め込むような設定を行うときは，埋め込むべきデータを格納したバッファの先頭アドレスを使用するAPPnセグメント番号に相当するメンバに，また，その埋め込むデータのサイズをAPPINFO構造体のメンバに登録します。

図２－26　圧縮でのAPPINFO構造体の設定



保守／廃止

(13) コメント・マーカ

unsigned char jpeg_COMStr[ ] = “ This is a Comment Marker ”；とコメント・マーカ文字列を定義している部分を替えられます。

(14) 量子化テーブル

輝度成分用と色差成分用にそれぞれ64バイトの量子化テーブルを指定してください。それぞれのテーブルは64要素からなり，１要素１バイトです。

表２－25　量子化テーブルの設定

メンバ	内　　容
DQT_Y_Bptr	輝度成分用の量子化テーブル
DQT_C_Bptr	色差成分用の量子化テーブル

ライブラリで用意したテーブルを用いるには次の名前を指定してください。

輝度成分用：LuminanceQtbl
色差成分用：ChrominanceQtbl

輝度成分のディフォールト量子化テーブル

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

色差成分のディフォールト量子化テーブル

$$\begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

(15) ハフマン・テーブル

輝度成分DC用/AC用，色差成分DC用/AC用の４つのハフマン・テーブルをDHTセグメント形式で指定してください。

表２－26　ハフマン・テーブルの設定

メンバ	内　　容
DHT_DC_Y_Bptr	輝度DC用のハフマン・テーブル
DHT_DC_C_Bptr	色差DC用のハフマン・テーブル
DHT_AC_Y_Bptr	輝度AC用のハフマン・テーブル
DHT_AC_C_Bptr	色差AC用のハフマン・テーブル

ライブラリで用意したテーブルを用いるには次の名前を指定してください。

輝度成分DC用：DHT_markerLuminanceDC
輝度成分AC用：DHT_markerLuminanceAC
色差成分DC用：DHT_markerChrominanceDC
色差成分AC用：DHT_markerChrominanceAC

保守／廃止

（16）外部RAMワーク・エリア・アドレス（Work）

外部RAMワーク・エリアの先頭アドレスを設定してください。

表２－27　メンバWorkの設定

メンバ	内　　容
Work	外部RAMワーク・エリア0xA80バイトの先頭アドレス

２.４.４　コメント・マーカ設定

圧縮時にコメント・マーカを埋め込む／埋め込まないの設定，および埋め込む文字列の設定は次のように行います。

文字列とはNULL文字（0x00）で終わるようなASCIIコード列のことです。

（１）コメント・マーカを埋め込まない設定

次のどちらかの設定でライブラリを呼び出す側で指定します。

・unsigned char jpeg_COMStr [] = ” \0 ”;
・unsigned char jpeg_COMStr [] = {0} ;

（２）コメント・マーカに文字列を埋め込む設定

文字列“ ABCDE ”を埋め込む場合には，次のように指定します（Ｃ言語では文字列のうしろに0x00が自動的に付加されます）。

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

また，アセンブラの場合には次の例のようにNULL文字を明示的に付加する必要があります。

```
        .text
        .align 4
        .globl_ jpeg_COMStr
_jpeg_COMStr：
        .str  ” ABCDE\0 ”
```

ただし，jpeg_COMStr[] = ” V830 ”と指定した場合には次の（Ｃ）のような特別な意味になります。

（３）コメント・マーカにバイナリ・コードを埋め込む設定

たとえば次のようなコードをコメント・マーカに埋め込むには次に示す①～③の順序で設定します。

```
” This\0is\0comment\0including\0null\0character ”
```

①　jpeg_COMStrには４文字のキー・ワード“ V830 ”を指定します。

保守／廃止

② CJInfo.IR［0］からの４バイトをint型にキャストし，埋め込むコードのバイト数を指定します。

③ CJInfo.IR［4］からの４バイトをunsigned char*型にキャストし，埋め込むコードの先頭ポインタを指定します。

設定例を次に示します。

```
unsigned  char   jpeg_COMStr[] = ”V830”;
unsigned  char
      jpeg_COMBuff[] = ”This\0is\0comment\0including\0null\0character”;

void   compress_parameter_ini（）
{

省略

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

OSのマルチタスク機能を利用して２つ以上の圧縮タスクを同時に動かす場合で，異なるコメント・マーカを埋め込む場合にも①～③の順序で設定してください。

2.4.5　DHTセグメント，DQTセグメント

通常のJPEGフォーマットでは２面の量子化テーブルと４面のハフマン・テーブルをそれぞれDQTセグメント，DHTセグメントの形でヘッダの中に含んでいます。

JPEG規格（ISO/IEC 10918）ではこれらのテーブルを個別に記述することと，ひとかたまりにして記述することの両方を許可しています。具体的な記述例を次に示します。

（a）64バイトの量子化テーブル２面を個別に記述

・0xFF, 0xDB，（セグメントの長さを表す２バイト），（テーブル番号），64バイトのテーブル

・0xFF, 0xDB，（セグメントの長さを表す２バイト），（テーブル番号），64バイトのテーブル

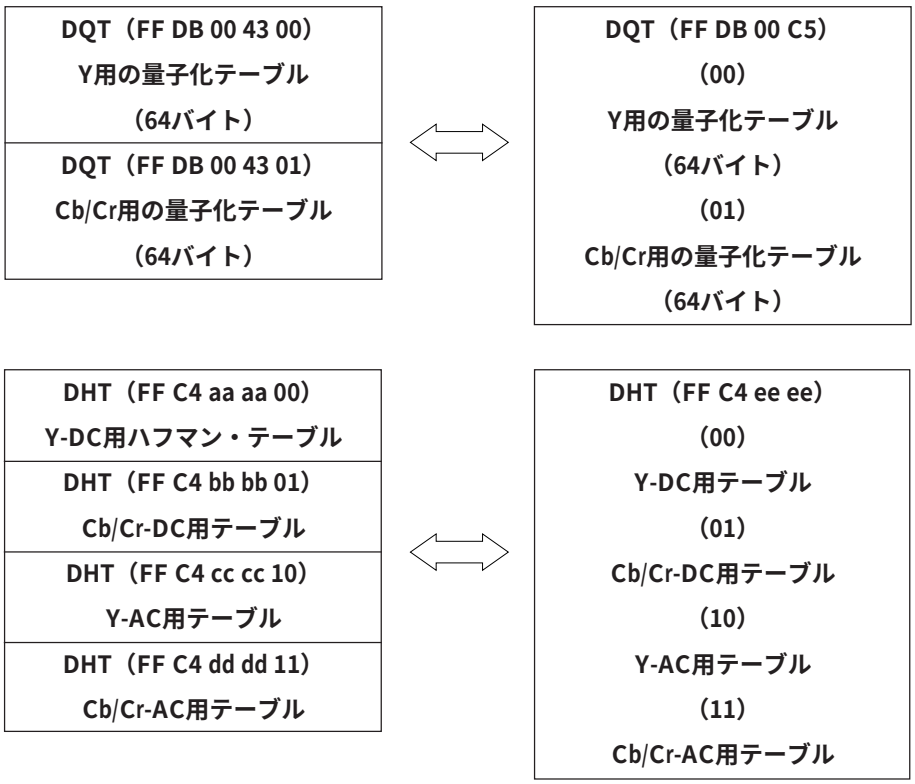
（b）64バイトの量子化テーブル２面をひとかたまりに記述

・0xFF, 0xDB，（長さを表す２バイト），（テーブル番号），64バイト，（テーブル番号），64バイト

AP705100-B03, AP70732-B03の基本ライブラリでは，通常の方法で圧縮した場合，DQT, DHTセグメントは個別に記述されます。

保守／廃止

個別に記述した例　　ひとかたまりに記述した例



AP705100-B03, AP70732-B03基本ライブラリでDQT, DHTセグメントをまとめる記述にするには次の手順で行います。

手順　：CJInfo.IR［8］からの４バイトをunsigned char*型にキャストし，２文字のキー・ワード“Ex”を指定する

記述例：*(unsigned char**)&(CJInfo.IR[8]) =　”Ex”;

2.4.6　ハフマン・テーブルを独自に用意する場合の制限事項

AP705100-B03, AP70732-B03基本ライブラリでは，圧縮時に使用するハフマン・テーブルを差し替えることができます。しかし，差し替えられたハフマン・テーブルはどのようなものでもよいとはかぎりません。適切でないハフマン・テーブルが指定された場合，画像によっては正常に圧縮されるものとされないものがある，という不具合が生じます。しかも，AP705100-B03, AP70732-B03基本ライブラリの圧縮ルーチンは，そのような場合でも正常終了（返り値JPEG_OK）します。

このような事態を避けるため，ハフマン・テーブルをユーザが作成する際は次の２点を守ってください。

・差し替えられたハフマン・テーブルのL1-L16は，理論的に内容の合ったものでなければならない
・差し替えられたハフマン・テーブルのV1-Vmには，DC成分用ではカテゴリ11のものまで，AC成分用ではカテゴリ10のものまでが含まれていなければならない

図２－27　DHTセグメント

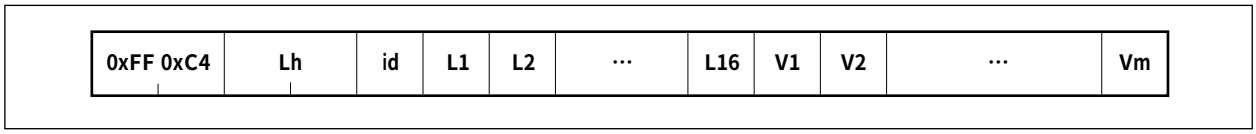


表２－28　DC/AC成分の値とビット長

成分の値	カテゴリ
0	0
－1, 1	1
－3, －2, 2, 3	2
－7～－4, 4～7	3
－15～－8, 8～15	4
－31～－16, 16～31	5
－63～－32, 32～63	6
－127～－64, 64～127	7
－255～－128, 128～255	8
－511～－256, 256～511	9
－1023～－512, 512～1023	10
－2047～－1024, 1024～2047	11

（１）DHTセグメントのL1からL16の部分は，ｉビット長のハフマン・コードが何個あるかを示しています。
たとえば，L1-L16が次のような値であるとします。

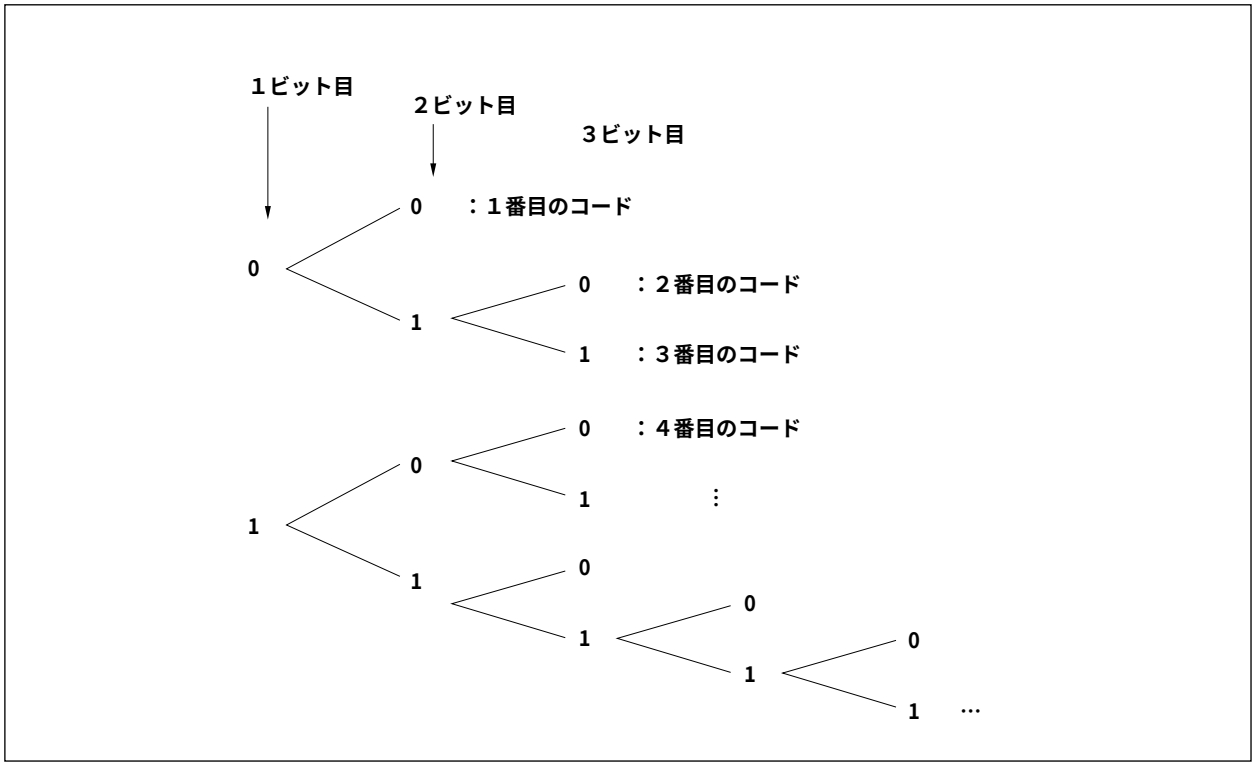
00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00

これは次のような意味になります。

１ビット長のコードが，０個
２ビット長のコードが，00の１個
３ビット長のコードが，010,011,100,101,110の５個
４ビット長のコードが，1110の１個
５ビット長のコードが，11110の１個
６ビット長のコードが，111110の１個
７ビット長のコードが，1111110の１個
８ビット長のコードが，11111110の１個
９ビット長のコードが，111111110の１個
それ以外のコード長はない

圧縮コードは，図２－28のように，ビット長の短いものから順にその値が確定していきます。

図２－28　圧縮コードの値の確定



DHTセグメントのL1-L16では，各要素の意味に合致しないものは，データとしての意味を持ちません。たとえば，１ビットの値が３つあるような組み合わせ（L1 = 3）はありえません。しかし，AP705100-B03, AP70732-B03基本ライブラリではそこまでチェックしていません。したがって，L1-L16に理論的に意味のある値を持つハフマン・テーブルを使用してください。

（２）DHTセグメントのV1-Vmの部分は，各圧縮コードがどのカテゴリとゼロランの組み合わせに対応するかを示しています。

たとえば，V1-Vmが次のような値であったとします。

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

これは，各コードが次のようなカテゴリであることを示します。

１番目の圧縮コード（00）が，カテゴリ０（End of Block）
２番目の圧縮コード（010）が，カテゴリ１
３番目の圧縮コード（011）が，カテゴリ２
４番目の圧縮コード（100）が，カテゴリ３
５番目の圧縮コード（101）が，カテゴリ４
６番目の圧縮コード（110）が，カテゴリ５
⋮
12番目の圧縮コード（111111110）が，カテゴリ11（0xB）

DC成分用のハフマン・テーブルでは，V1-Vmの各要素は0-0xBです。一般には，画像を圧縮した場合，カテゴリ２とカテゴリ１のビット長が最も多く分布し，カテゴリ11に近づくに従って出現確率が低くなります。画像によっては，全体にカテゴリ８，９，10，11のビット長が現れない場合もあります。このような場合には，V1-Vmでカテゴリ８以上の部分をなくしたハフマン・テーブルを使用しても正常に圧縮され，また正常に伸長されます。ところが，こうして作られたカテゴリ８以上のないハフマン・テーブルを使用して，カテゴリ９の値が出現するような画像を圧縮した場合，AP705100-B03, AP70732-B03の圧縮ルーチンでは，カテゴリ９に相当する圧縮コードに０ビット長の０を埋め込み，実際には圧縮コードを埋め込んでいないにもかかわらず，圧縮コードを埋め込んだものと解釈して正常終了します。このようにしてできあがったJPEGファイルを伸長すると，カテゴリ９のデータが出現するはずの場所から先は，エラーになるか，画像がモザイクのようになってしまいます。

AC係数にも，DC係数と同じような傾向があります。たとえば，V1-Vmの要素は，AP705100-B03, AP70732-B03基本ライブラリに付属のAC成分用のハフマン・テーブル（jpeg_DHT_AC_Y）では，次のようになっています。

```
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
```

それぞれの要素の下位４バイトがカテゴリを，上位４バイトがゼロランを示します。また，0x00はEOB（End of block），0xF0はZRL（Zero run length）という特別なコードです。

この例の場合には，次のような意味になります。

１番目の圧縮コードが，ゼロラン０のカテゴリ１
２番目の圧縮コードが，ゼロラン０のカテゴリ２
３番目の圧縮コードが，ゼロラン０のカテゴリ３
４番目の圧縮コードが，EOB
５番目の圧縮コードが，ゼロラン０のカテゴリ４
６番目の圧縮コードが，ゼロラン１のカテゴリ１
⋮

AC係数の場合もDC係数の場合と同様に，カテゴリの小さいものほど出現確率は高く，大きいものほど出現頻度が小さくなります。また，ゼロランは０のものが最も多く，大きくなるにつれて出現確率が低下します。そこで，ゼロランとカテゴリが大きい部分に相当する圧縮コードを持たないようなハフマン・テーブルを作ることが考えられます。しかし，AP705100-B03, AP70732-B03では，そのようなテーブルを指定して圧縮した場合，正常に伸長されなくなることがあります。

したがって，V1-Vmの値に偏りのないハフマン・テーブルを使用してください。

2.4.7　Exif対応

Exif規格は財団法人　日本電子工業振興協会が定めるディジタル・スチル・カメラの画像フォーマット規格です。

Exif規格（Ver.1.0）で定められている画像フォーマットの特徴を次に示します。

・APP1マーカ・セグメントにExif規格が定めるパラメータのデータを埋め込む
・量子化テーブルをY用Cb用Cr用の３面持ち，１つのDQTセグメントにまとめる
・DHTセグメントを１つにまとめる

保守／廃止

（１）設定方法（Cb/Cr用の量子化テーブルが同一で良い場合）

AP705100-B03, AP70732-B03の基本ライブラリでは，APP1セグメントに埋め込むデータを作成する部分は提供していませんが，できあがったデータをAPP1セグメントに埋め込む設定は可能です。

AP705100-B03, AP70732-B03の基本ライブラリをExif対応させるには，次に示すようにCJInfo.IR［8］からの４バイトをunsigned char*型にキャストし，キー・ワード文字列“ Exif ”を指定します。

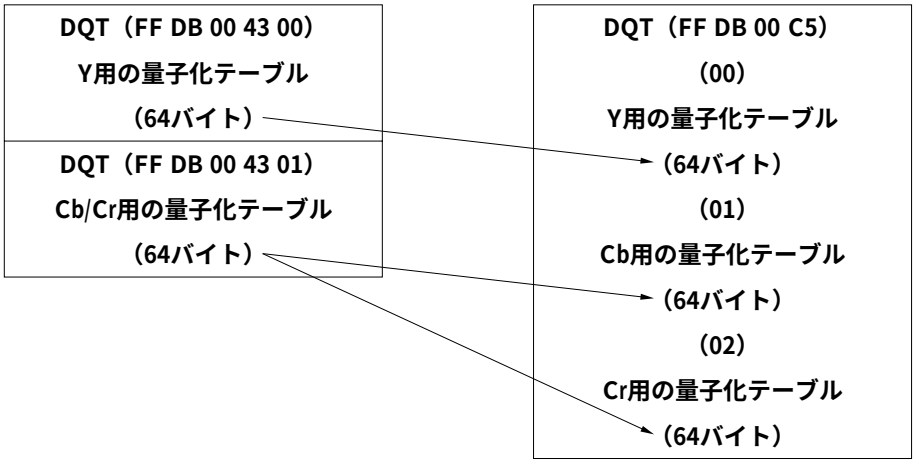
次に記述例を示します。

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

この設定により，次に示すファイル・フォーマットができるようになります。

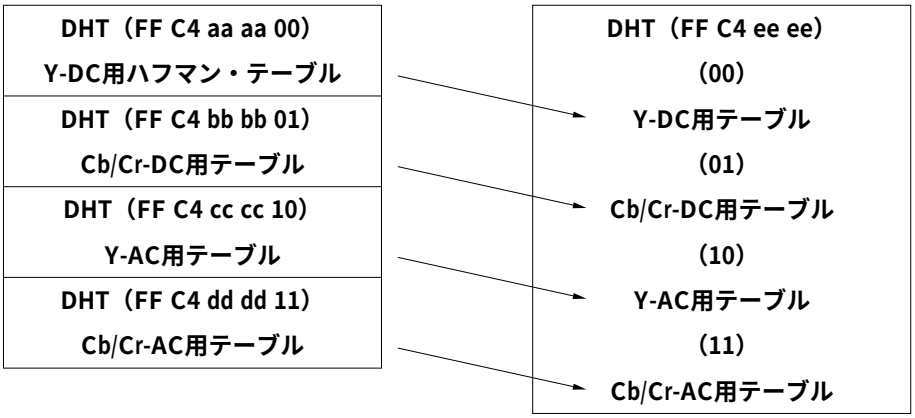
① コメント・マーカを強制的に排除する（このままの設定ではコメント・マーカを埋め込むことはできません）

② DQTセグメントのpack



備考　この場合３番目（Cr用）のテーブルには２番目（Cb用）と同じものがコピーされます。

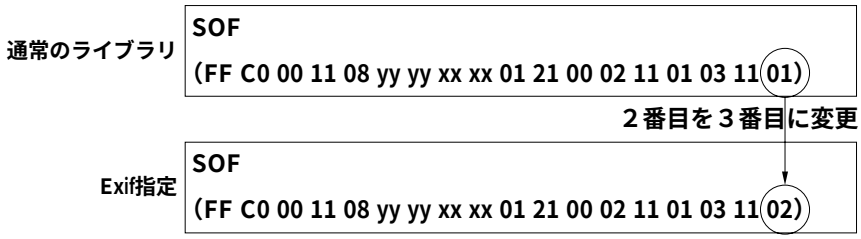
③ DHTセグメントのpack



保守／廃止

④　SOFセグメントの指定内容変更

Crに２番目の量子化テーブルが指定されているところを３番目のテーブルを指定するように変更します。



（２）設定方法（Cb/Cr用の量子化テーブルを分けたい場合）

AP705100-B03, AP70732-B03の基本ライブラリでは，次に示す順序で２番目の量子化テーブル（Cb用）と３番目の量子化テーブル（Cr用）を別々のものに設定することもできます。

①　CJInfo.IR［8］からの４バイトをunsigned char*型にキャストし，５文字のキー・ワード文字列“ ExifQ ”を指定します。

②　第３の量子化テーブル（64バイト）の先頭アドレスをCJInfo.IR［12］からの４バイトをchar**型にキャストした部分に設定します。

③　構造体のメンバWorkに設定するワーク領域は2880バイトを確保します。

次に記述例を示します。

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

注意　この設定を行う場合，外部RAMワーク・エリアは2688バイトではなく2880バイト必要です。もし，2688バイトのまま，この設定で実行した場合，続く192バイトが警告なく上書きされます。

保守／廃止

２．４．８　圧縮時のエラー内容

基本ライブラリの圧縮ルーチンは，なんらかの原因で処理が正常終了できなかった場合に，CJINFO構造体のメンバ“ ErrorState ”にエラーの値を代入し，処理を中止します。その際，返り値としてJPEG_ERRを返します。

そのときのエラー内容について表２－29に示します。

表２－29　圧縮のエラー内容

値	意　味
0x00000001	画像がVRAMの範囲を越えています（CJINFO構造体の（Width＋StartX）の値がVRAM_W_Pixelの値を越えていた場合／（Height＋StartY）の値がVRAM_H_Pixelの値を越えていた場合）。
0x00000002	サポートしていないサンプル比です（２：１：１のライブラリをはずしてリンクしたのに，サンプル比２：１：１で圧縮するように指定した場合など）。
0x00000005	指定したハフマン・テーブルに誤りがあります。
0xFFFFFFFF	致命的なエラーです（ライブラリの改造によるエラー）。

２．４．９　圧縮ルーチンの出力情報

基本ライブラリの圧縮ルーチンは，処理が正常終了した際に，次の情報を出力します。

表２－30　圧縮の出力情報

メンバ	返り値
FileSize	でき上がったJPEGファイルのバイト数（通常の圧縮モード）
	1MCU分の圧縮データのビット数（テスト・モード）

保守／廃止

２.５　基本伸長処理の実行

基本伸長処理では，JPEGファイルを伸長し，画像データを生成します。

２.５.１　基本伸長メイン関数

分　　類　伸長処理系

関 数 名　jpeg_Decompress

機能概要　JPEG伸長処理メイン

形　　式　#include “ jpeg.h"

int jpeg_Decompress（DJINFO＊　dJpeginfo）

引 き 数　DJINFO構造体の先頭アドレス

返 り 値　これらの返り値は数値です。

C言語で#define　JPEG_OK ０のように定義してあるものです。

表２－31　伸長処理関数の返り値

返り値	内　　容
JPEG_OK	正常終了
JPEG_ERR	エラー終了
JPEG_CONT1	JPEGバッファによる中断
JPEG_CONT2	VRAMによる中断

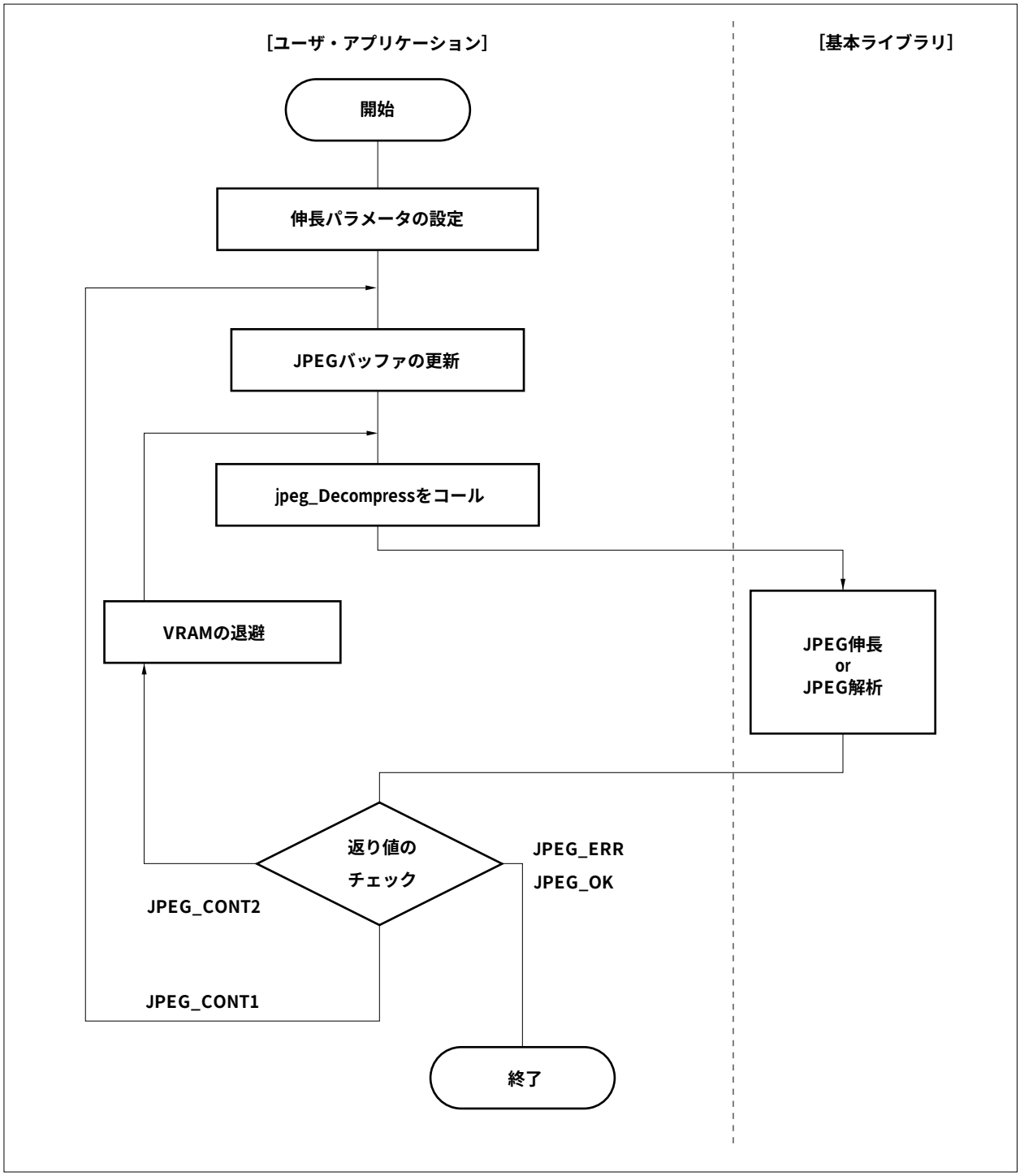
備考　JPEG_ERRの場合は，DJINFO構造体のメンバ“ ErrorState ”にエラー・ステートメントが格納されます。

保守／廃止

２．５．２　基本伸長処理フロー

基本伸長処理の処理の流れを次に示します。

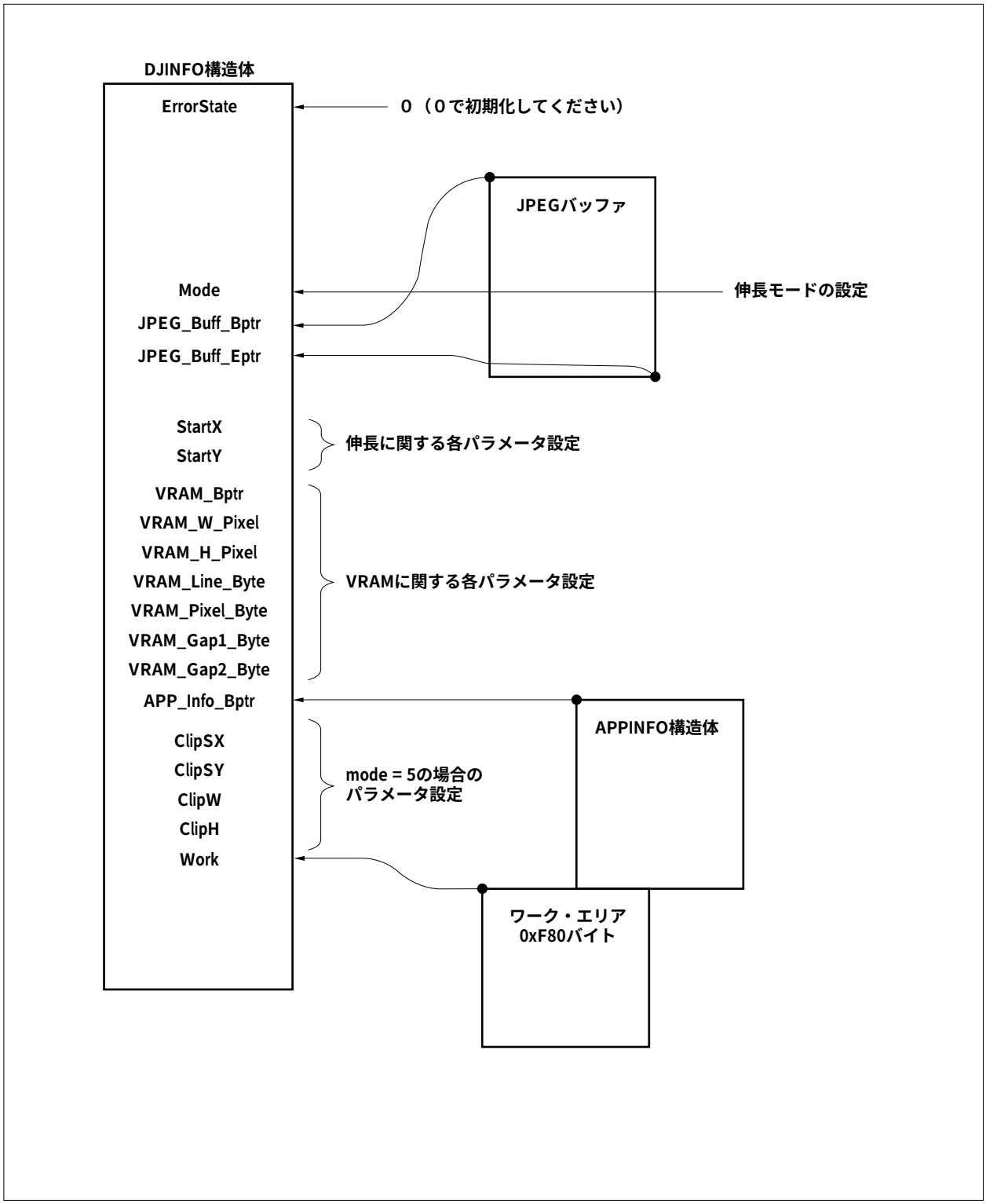
図２－29　基本伸長処理フロー



保守／廃止

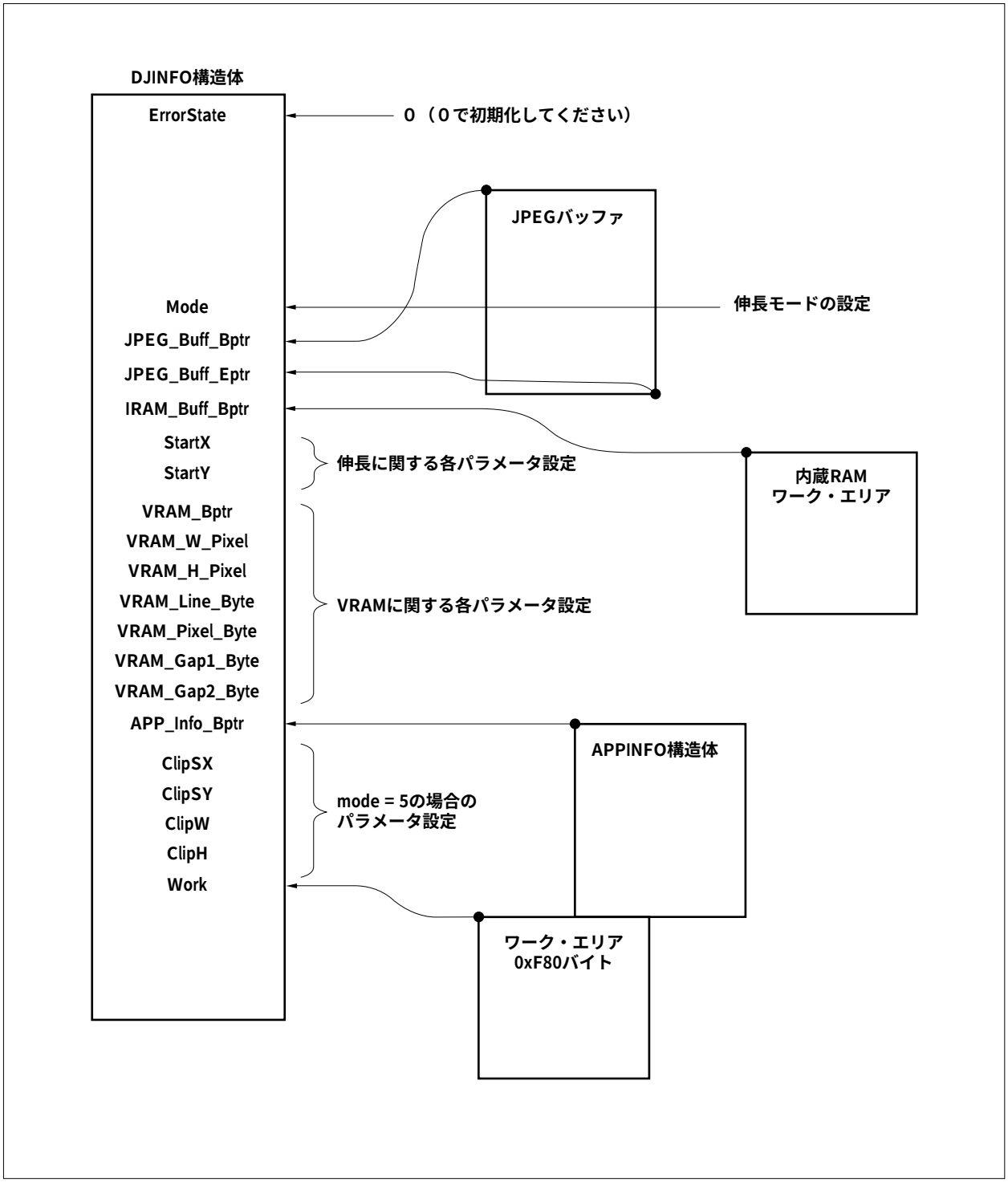
２.５.３　DJINFO構造体のパラメータ設定

図２－30　DJINFO構造体のパラメータ設定（AP70732-B03）



保守／廃止

図２－31　DJINFO構造体のパラメータ設定（AP705100-B03）



保守／廃止

（1）エラー・ステータスの初期化（ErrorState）

基本伸長ルーチンを起動する前の伸長パラメータ設定時に一度だけ０で初期化してください。

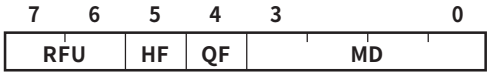
設定値：０

注意　処理を中断したあと，再開する場合には基本ライブラリがこのErrorStateの値を見て最初の起動か再開かを判別しているため，これ以外の初期化はしないでください（処理を中断した場合は，再開するアドレスを格納しています）。

（2）モード（Mode）

設定する値によって次のように選択されます。

表２－32　メンバModeの設定値



ビット位置	ビット名	意　　味
7-6	RFU	予約フィールド[注]
5	HF	ハフマン・テーブル・イニシャライズ用フラグ[注] ０：イニシャライズを行う １：イニシャライズを行わない
4	QF	量子化テーブル・イニシャライズ用フラグ[注] ０：イニシャライズを行う １：イニシャライズを行わない
3-0	MD	モード ０：解析モード １：通常伸長モード ２：1/4伸長モード ３：1/16伸長モード ４：1/64伸長モード ５：クリッピング伸長モード（RSTn未使用） ６：クリッピング（RSTn使用，EOIサーチせず）[注] ７：クリッピング（RSTn使用，EOIサーチ）[注]

注　Ver.2.10から追加

保守／廃止

（ａ）モード０

JPEG構造体のメンバにAPPnセグメントを解析するための構造体アドレスを付加すると，APPnセグメントの位置とサイズの解析を実行します。このAPPnセグメント解析用構造体が指定されない場合は，APPnに関する解析は実行しません。

APPnセグメント情報以外では，サンプル比，リスタート・インターバル，画像の縦横サンプル比とJPEGファイル・サイズを解析します。

（ｂ）モード１

通常の伸長です。

図２－32　伸長モード１の伸長例

（ｃ）モード２

逆DCTの部分の出力が８×８ではなく，４×４になるような逆DCT変換ルーチンを用いて高速に伸長処理を行います。

図２－33　伸長モード２の伸長例

（ｄ）モード３

縦横をそれぞれ４分の１にして出力します。

図２－34　伸長モード３の伸長例

保守／廃止

（ｅ）モード４

縦横をそれぞれ８分の１にして出力します。

図２－35　伸長モード４の伸長例

（ｆ）モード５

元のJPEGファイルから指定された矩形を切り出し，その部分だけを伸長します。

図２－36　伸長モード５の伸長例

クリッピングはMCUの単位でなければなりません。

このモードを使用するには次のメンバに値を設定する必要があります。

表２－33　クリッピングに関するメンバの設定値

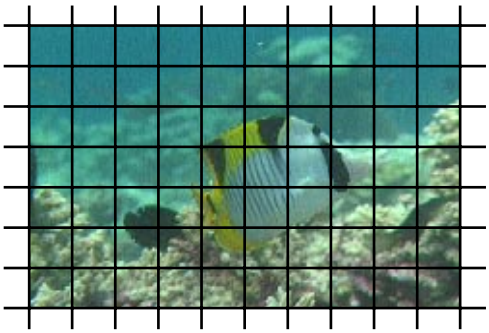
メンバ	内　　容
ClipSX	クリッピングを開始する位置が左から何番目のMCUかを設定
ClipSY	クリッピングを開始する位置が上から何番目のMCUかを設定
ClipW	横幅をMCU単位で指定
ClipH	縦幅をMCU単位で指定

上図のようにクリッピングする場合に，画像にMCUの区切りを入れたものが下図のようになったとすると，次のように指定します。

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

保守／廃止

図２－37　クリッピング指定例



（g）モード６

リスタート・マーカを活用し，「ハフマン復号化」を「リスタート・マーカのサーチ」で置き換えています。これにより処理速度が若干速くなります。このモードでは，クリッピング領域を終了した時点でデコードを終了するためDjinfo.FileSizeの値は不定になります。

（h）モード７

モード６の機能にDjinfo.FileSizeの値の整合性を保つようにしたものです。

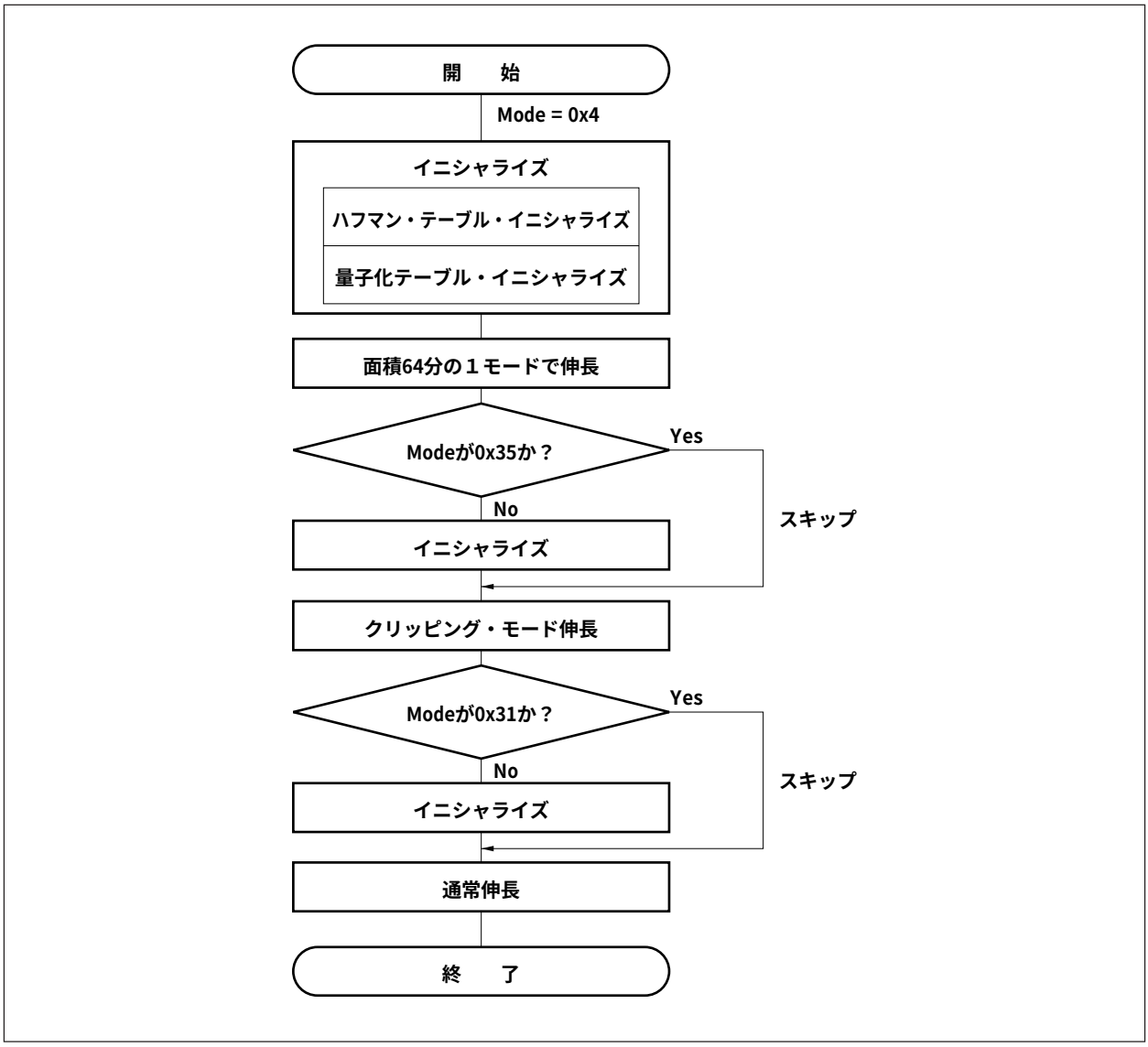
モード５の場合より処理速度が速くなりますが，モード６の場合より遅くなります。

構造体のメンバModeに指定される3968バイトが量子化とハフマンのルックアップ・テーブルに使用されていること，HFビットとQFビットでそれらのルックアップ・テーブルのイニシャライズを行う／行わないの設定ができることは圧縮の場合と同様です。

使用例として図２－38では，1/64伸長モードで伸長したあと，同じルックアップ・テーブルを使って伸長を行っています。

保守／廃止

図２－38　伸長モード設定の追加



（３）JPEGバッファの先頭アドレスと終了アドレス

JPEGバッファに関しては，**２.３.５　JPEGバッファ**を参照してください。

表２－34　メンバJPEG_Buff_Bptr/JPEG_Buff_Eptrの設定値

メンバ	内　容
JPEG_Buff_Bptr	JPEGバッファの先頭アドレス
JPEG_Buff_Eptr	JPEGバッファの先頭アドレス ＋JPEGバッファサイズ

JPEGバッファ・サイズの制限により処理途中でバッファ退避処理が入るような場合には，JPEGバッファを２面持って，それらを切り替えながら動かすことができます。

なお，詳細は**図２－21　２面のJPEGバッファの切り替え**を参照してください。

（４）内蔵RAMワーク・エリア・アドレス（IRAM_Buff_Bptr：AP705100-B03）

内蔵RAMワーク・エリアに関しては，**２.３.４　MCUバッファ**を参照してください。

保守／廃止

表２－35　メンバIRAM_Buff_Bptrの設定値

メンバ	内　　容
IRAM_Buff_Bptr	内蔵RAMワーク・エリアの先頭アドレス

伸長するJPEGファイルのサンプル比４：１：1/2：１：1/1：１：１によって，それぞれ先頭から0x400/0x300/0x280バイトが無条件に上書きされます。圧縮の場合と違って，伸長の場合はサンプル比を自分で指定しないでJPEGファイルのSOFヘッダの値を用いるため，より注意が必要です。

なお，AP70732-B03（V810ファミリ版）ではこのメンバの設定は必要ありません。

注意　Mode＝0（解析モード）の場合，この内蔵RAMワーク・エリアに指定する領域は，サンプル比に関係なく0x100バイトで十分です。

（５）画像の開始x位置（StartX）と画像の開始y位置（StartY）

設定値：－32768～32767

値の単位はピクセル数です。

StartX/StartYに関する詳細は，**図２－22　画像の開始位置（x, y）**を参照してください。

（６）VRAMに関するパラメータ設定

VRAMに関するパラメータ設定は圧縮とまったく同様です。**2.4.3（10）VRAMサイズ，（11）VRAM構成**を参照してください

表２－36　VRAMに関するメンバの設定値

メンバ	内　　容	VRAM出力部分のカスタマイズをする場合の設定の要／否
VRAM_W_Pixel	VRAMの横ピクセル数	必要
VRAM_H_Pixel	VRAMの縦ピクセル数	
VRAM_Bptr	VRAM先頭アドレス（基点アドレス）	不要
VRAM_Line_Byte	縦１ピクセル分のVRAMのアドレス差	
VRAM_Pixel_Byte	横１ピクセル分のVRAMのアドレス差	
VRAM_Gap1_Byte	VRAMがYCbCrの場合 同一ピクセルのYとCbとのアドレス差 VRAMがRGBの場合 同一ピクセルのRとGとのアドレス差	
VRAM_Gap2_Byte	VRAMがYCbCrの場合 同一ピクセルのYとCrとのアドレス差 VRAMがRGBの場合 同一ピクセルのRとBとのアドレス差	

VRAM出力部分のカスタマイズをする場合に，設定が必要な２つのメンバ（VRAM_W_Pixel/VRAM_H_Pixel）がSOFセグメント解析時に，サイズをチェックしています。

（７）APPINFOテーブルの指定（APP_Info_Bptr）

APP_Info_BptrにAPPINFO構造体の先頭アドレスを指定した場合，APPnセグメントの解析を行います。

保守／廃止

表２－37　メンバAPP_Info_Bptrの設定値

メンバ	設定値
APP_Info_Bptr	０：APPnセグメントを解析しない
	APPINFO構造体の先頭アドレス：解析する

APPINFO構造体が登録されていてAPPnセグメントが見つかった場合は，そのデータの先頭アドレスとサイズがAPPnセグメント番号に相当するメンバに書き出されます。

（８）クリッピングに関するパラメータの設定（ClipSX, ClipSY, ClipW, ClipH）

これらの値は，Mode＝5（クリッピング・モード）の場合だけ参照されます。

設定方法については２.５.３（２）（ｆ）モード５を参照してください。

注意　実際の画像からはみ出すような指定をした場合にはエラーとなります。

（９）外部RAMワーク・エリア・アドレス（Work）

外部RAMワーク・エリアの先頭アドレスを設定してください。

表２－38　メンバWorkの設定値

メンバ	内　　容
Work	外部RAMワーク・エリア0xF80バイトの先頭アドレス

注意　Mode＝5の場合，Workは必要ありません。

２.５.４　Exif対応

AP705100-B03, AP70732-B03の基本ライブラリは，通常の設定では量子化テーブルが３面あるためExif規格のJPEGファイルを伸長できません。

これを可能にするためには次の順序で設定します。

① DJInfo.IR[28]からの４バイトをunsigned char*型にキャストし，５文字のキー・ワード文字列“Exif”を指定します。

② 構造体のメンバWorkに設定するワーク領域は4224バイトを確保します。

次に記述例を示します。

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

注意　この設定を行う場合，外部RAMワーク・エリアは3968バイトではなく4224バイト必要です。もし，3968バイトのまま，この設定で実行した場合，続く128バイトが警告なく上書きされます。

２.５.５　基本伸長時のエラー内容

基本ライブラリの伸長ルーチンは，なんらかの原因で処理が正常終了できなかった場合に，DJINFO構造体のメンバ“ErrorState"にエラーの値を代入し，処理を中止します。その際，返り値としてJPEG_ERRを返します。

そのときのエラー内容について表２－39に示します。

保守／廃止

表２－39　基本伸長のエラー内容

値	意　味
0x00000001	画像がVRAMの範囲を越えています。 ・DJINFO構造体のStartXにJPEG画像の横サイズを足した値がVRAM_W_Pixelの値を越えていた場合 ・StartYにJPEG画像の縦サイズを足した値がVRAM_H_Pixelの値を越えていた場合
0x00000002	サポートしていないサンプル比です（2：1：1のライブラリをはずしてリンクしたのに，伸長しようとするJPEGがサンプル比2：1：1だった場合など）。
0x00000003	DQTヘッダのPqの値が０以外に設定されています。
0x00000004	DQTヘッダのTpの値が0,1,2,3以外の値です。
0x00000005	DHTヘッダのTc, Tpの値が不正です。
0x00000006	SOSヘッダのコンポーネント数が３ではありません。
0x00000007	SOSヘッダで指定されたハフマン・テーブル番号に誤りがあります。
0x00000008	SOSヘッダのSsの値が０ではありません。
0x00000009	SOSヘッダのSeの値が63ではありません。
0x0000000A	SOSヘッダのAh, Alの値が０ではありません。
0x0000000B	SOFヘッダのPに８以外の値が設定されています。
0x0000000C	SOFヘッダのNfの値が大き過ぎます。
0x0000000D	未知のマーカが現れました。
0x0000000E[注]	RSTnマーカの値が不正です。
0x0000000F	その他のエラー
0xFFFFFFFF	致命的なエラー（ライブラリの改造によるエラー）
0x00000010	SOIマーカが見つかりませんでした。
0x00000011	SOF0マーカが見つかりませんでした。
0x00000012	DQTマーカが見つかりませんでした。
0x00000013	DHTマーカが見つかりませんでした。
0x00000014[注]	RSTnマーカが見つかりませんでした。
0x00000015	EOIマーカが見つかりませんでした。
0x0000001F	予想されない位置にマーカが見つかりました。

注　RSTnマーカ・エラーが返る可能性のあるのはクリッピング・モード（Mode = 6, 7）の場合だけです。

表２－40　チェックしていないエラー

条　　件	エラー・チェック
解析モードで実行した場合	SOI/SOF1/DQT/DHT/EOI
モード値の0x10のビットが立っていた場合	DQT
モード値の0x20のビットが立っていた場合	DHT
クリッピング・モード（Mode = 7）で実行した場合	EOI

保守／廃止

２．５．６　基本伸長ルーチンの出力情報

基本ライブラリの伸長ルーチンは，処理が正常終了した場合に，次の情報を出力します。

表２－41　伸長の出力情報

DJINFO構造体のメンバ	内　　容
FileSize	伸長したJPEGファイルのバイト数
Sampling	伸長したJPEGファイルのサンプル比 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	伸長したJPEGファイルのリスタート・インターバル
Width	伸長したJPEGファイルの画像横サイズ（ピクセル数）
Height	伸長したJPEGファイルの画像縦サイズ（ピクセル数）

構造体Djinfoの先頭から0x7Cのアドレスにある４バイトにはJPEGファイルのヘッダの情報が生成されます。
次に記述例を示します。

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

この４バイトの意味を次の表に示します。

表２－42　JPEGファイルのヘッダ情報の意味

ビット	セグメント名	意　　味
31ビット	SOI	SOIマーカが見つかった場合にこのフラグが立てられます。 このとき，他のすべてのフラグは０でマスクされます。
30ビット	EOI	EOIマーカが見つかった場合にこのフラグが立てられます。
29ビット	SOF0	SOF0マーカが見つかった場合にこのフラグが立てられます。
10ビット	COM	COMマーカが見つかった場合にこのフラグが立てられます。
９ビット	SOS	SOSマーカが見つかった場合にこのフラグが立てられます。
８ビット	DRI	DRIマーカが見つかった場合にこのフラグが立てられます。
７ビット	DQT	テーブル番号３のDQT
６ビット	DQT	テーブル番号２のDQT
５ビット	DQT	テーブル番号１のDQT
４ビット	DQT	テーブル番号０のDQT
３ビット	DHT	AC用，テーブル番号１のDHT
２ビット	DHT	AC用，テーブル番号０のDHT
１ビット	DHT	DC用，テーブル番号１のDHT
０ビット	DHT	DC用，テーブル番号０のDHT

ただし，次に示すチェックは行われません。

- 解析モードでのDHT, DQTに関するチェック
- 解析モード以外のモードでMode値の0x10ビットが立っていた場合のDQTに関するチェック
- 解析モード以外のモードでMode値の0x20ビットが立っていた場合のDHTに関するチェック

保守／廃止

次に示す情報は，DJINFOのメンバAPP_Info_BptrにAPPINFO構造体を指定した場合だけ出力します。

表２－43　APPxx_Buff_Bptr/APPxx_BuffSize

APPINFO構造体のメンバ	内　　容
APPxx_Buff_Bptr	APPnセグメントのアドレス（JPEGファイルの先頭からの相対位置）
APPxx_Buffsize	APPnセグメントのサイズ（バイト）

保守／廃止

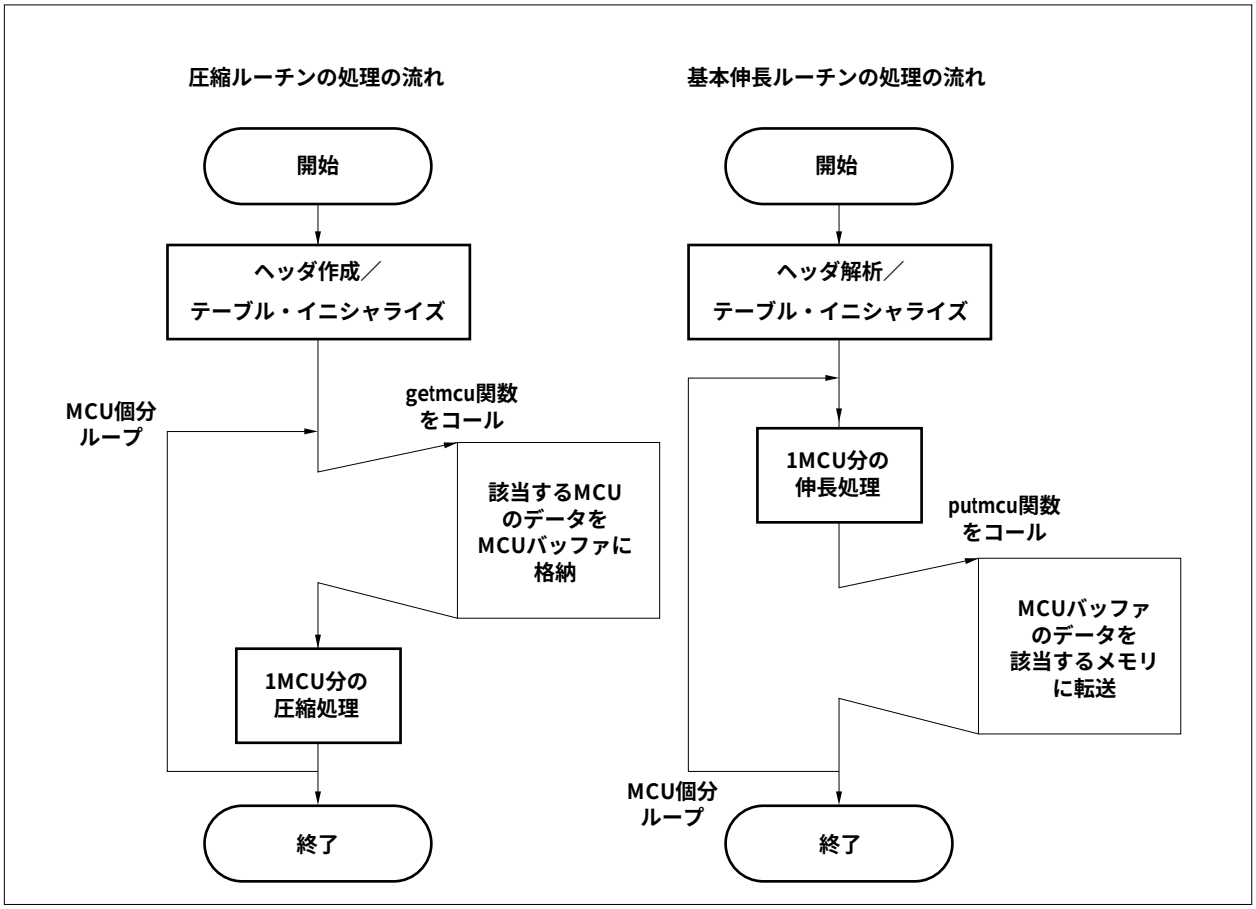
2.6　基本ライブラリのカスタマイズ

JPEG圧縮／伸長処理で最もハードウエアに左右されるのが画像入出力の部分です。基本ライブラリでは画像入出力の部分を自作できるようにしています（基本ライブラリ付属のディフォールトのVRAMアクセス関数を使用できますが，汎用的な仕様のため処理速度を重視していません）。

2.6.1　基本ライブラリでの画像データの取り扱い

基本ライブラリでは，画像データをMCU（Minimum Coded Unit）単位で処理しています（圧縮ではMCU単位で，画像入力→DCT変換→量子化→ハフマン符号化，伸長ではMCU単位で，ハフマン復号化→逆量子化→逆DCT変換→画像出力を行っています）。

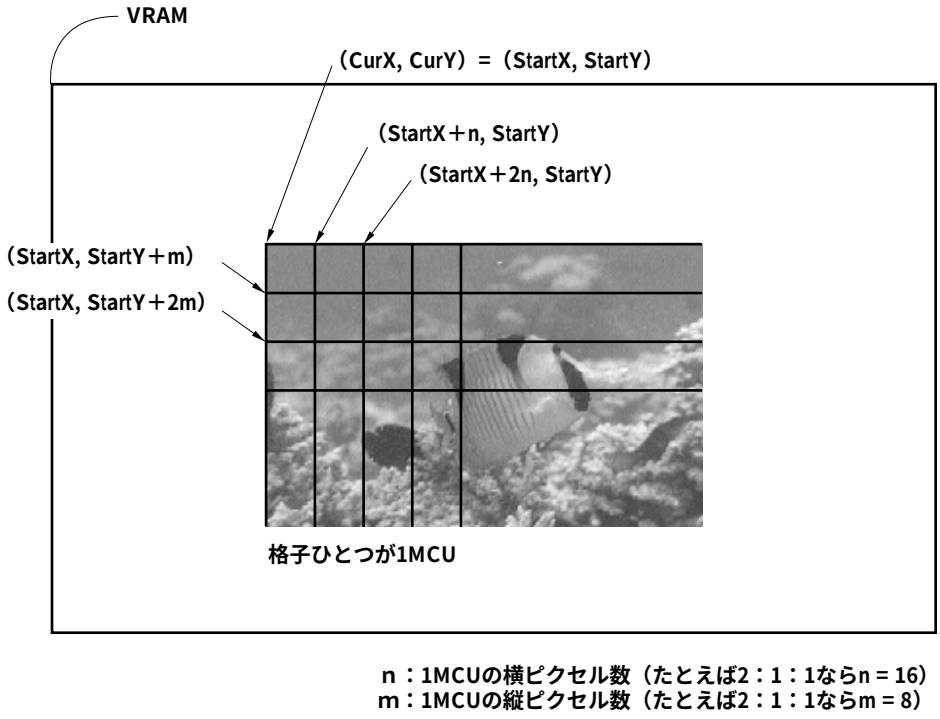
図２－39　JPEGの処理の流れ



MCUバッファはCJINFO構造体，DJINFO構造体のそれぞれ最後のメンバとして領域を確保されています。

保守／廃止

図２－40　構造体のメンバCurrentX/CurrentY



n：1MCUの横ピクセル数（たとえば2：1：1ならn = 16）
m：1MCUの縦ピクセル数（たとえば2：1：1ならm = 8）

putmcu関数/getmcu関数をカスタマイズする際に，そのMCUが画像のどの部分を指しているのかは，構造体のメンバ（CurrentX, CurrentY）を参照してください。

このメンバCurrentXとCurrentYはライブラリ側が1MCU処理ごとに値を更新します。ユーザ側でカスタマイズした側からは値を変更しないようにしてください。

圧縮のgetmcu関数をカスタマイズする場合には，getmcu関数がライブラリ側からコールされるたびに，該当するMCUのデータがY/Cb/Crの形式で（Y/Cb/Crそれぞれ0-255）MCUバッファに格納されるように作ります。伸長のputmcu関数をカスタマイズする場合は逆で，putmcu関数がコールされるときにはそのMCUに相当するデータがY/Cb/Crの形式で格納されていますから，そのデータをVRAMに転送するように作ります。

表２－44　MCUの単位

サンプル比	MCU単位
4：1：1（H：V = 2：2）	縦16×横16ピクセル
4：1：1（H：V = 4：1）	縦８×横32ピクセル
2：1：1（H：V = 2：1）	縦８×横16ピクセル
1：1：1（H：V = 1：1）	縦８×横８ピクセル

基本ライブラリで，圧縮のDCT変換の入力，伸長の逆DCT変換の出力はRGBではなく，YCbCr形式です。VRAM形式がRGBの場合には，画像データをこれに合わせるためにそれぞれのピクセル単位でRGB⟷YCbCr変換が必要です（**図１－４　JPEG処理概要**参照）。

なお，このライブラリではCCIR勧告601に準拠し，Y,Cb,Crの値をunsigned char型で扱っています。

保守／廃止

２.６.２　サンプル比とブロック

圧縮では，それぞれのMCUに対して，YCbCr分解し，サンプル比に応じてブロックに分解します。

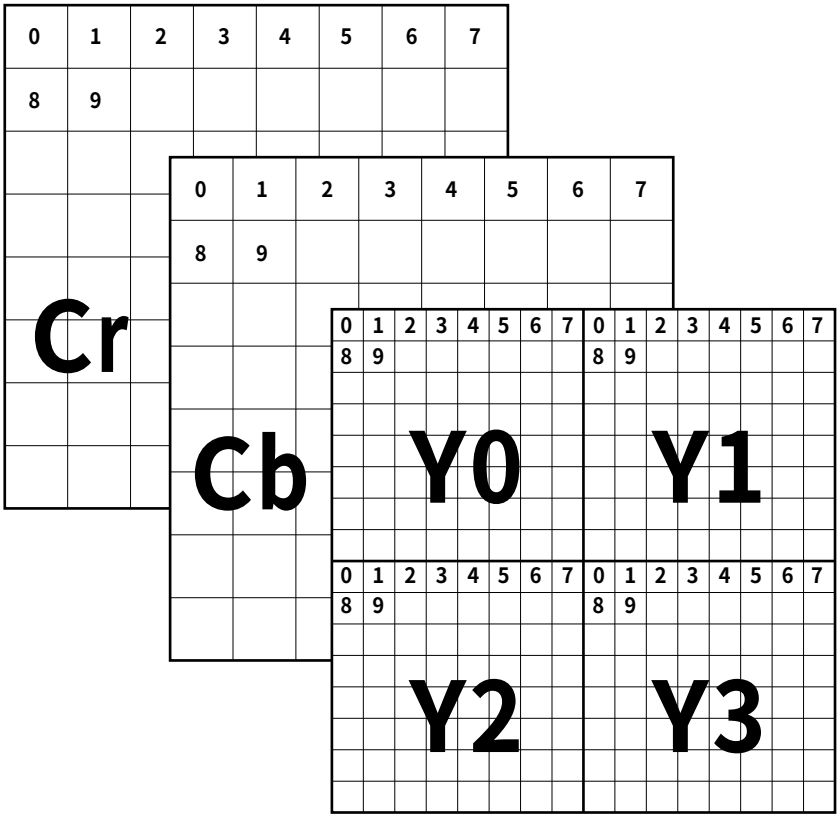
表２－45　MCUとブロック

サンプル比	MCU単位	ブロック
4：1：1（H：V＝2：2）	16×16	Y:4／Cb:1／Cr:1ブロック
4：1：1（H：V＝4：1）	８×32	Y:4／Cb:1／Cr:1ブロック
2：1：1（H：V＝2：1）	８×16	Y:2／Cb:1／Cr:1ブロック
1：1：1（H：V＝1：1）	８×８	Y:1／Cb:1／Cr:1ブロック

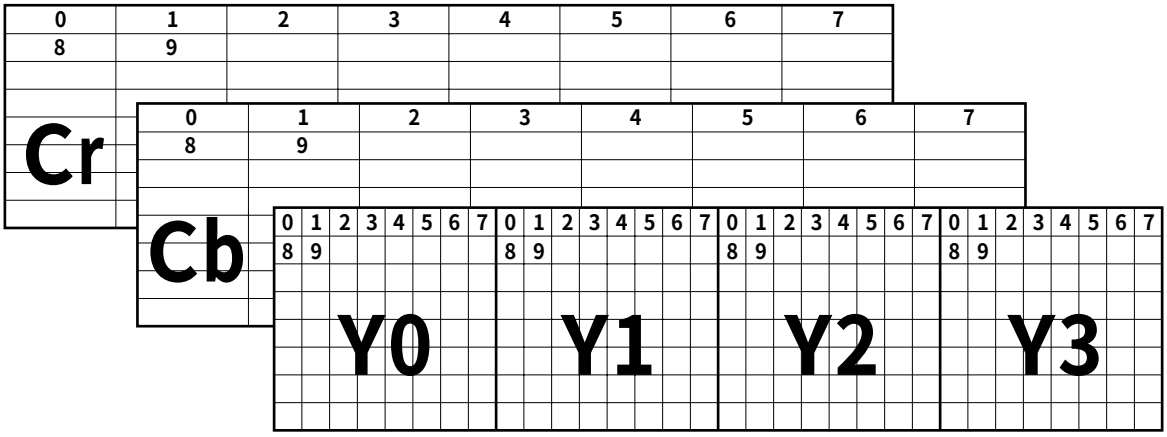
保守／廃止

図２－41　1MCU分の画像データ（1/2）

（a）4：1：1（H：V＝2：2）の場合



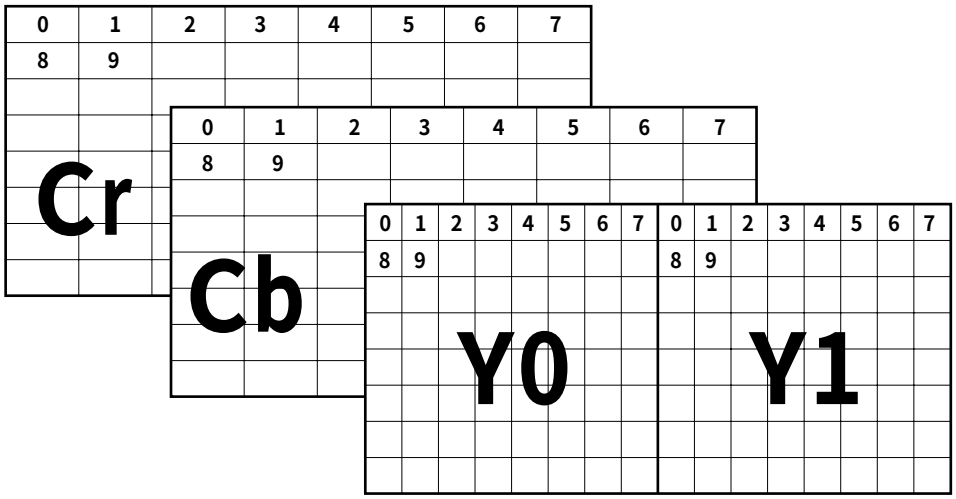
（b）4：1：1（H：V＝4：1）の場合



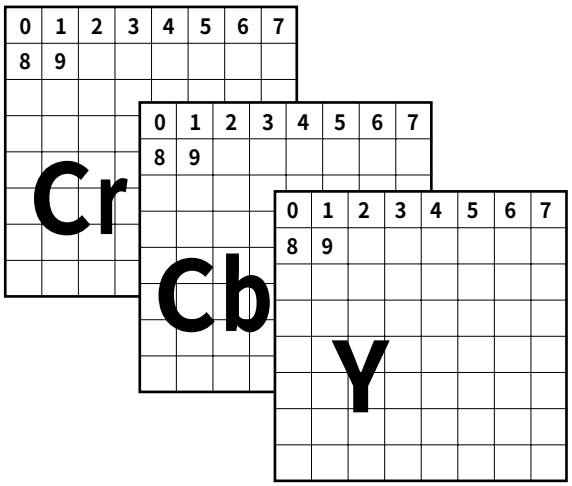
保守／廃止

図２－41　1MCU分の画像データ（2/2）

（c）2：1：1（H：V＝2：1）の場合



（d）1：1：1（H：V＝1：1）の場合



圧縮では，サンプル比１：１：１を除いて，色差成分（Cb/Cr）については隣り合ういくつかのピクセルの平均をとる必要があります。この平均をとる処理をサンプリングと呼んでいます。

表２－46　色差成分のサンプリング

サンプル比	サンプリング
4：1：1（H：V＝2：2）	縦２×横２ピクセル分の色差成分の平均をとる
4：1：1（H：V＝4：1）	横４ピクセル分の色差成分の平均をとる
2：1：1（H：V＝2：1）	横２ピクセル分の色差成分の平均をとる

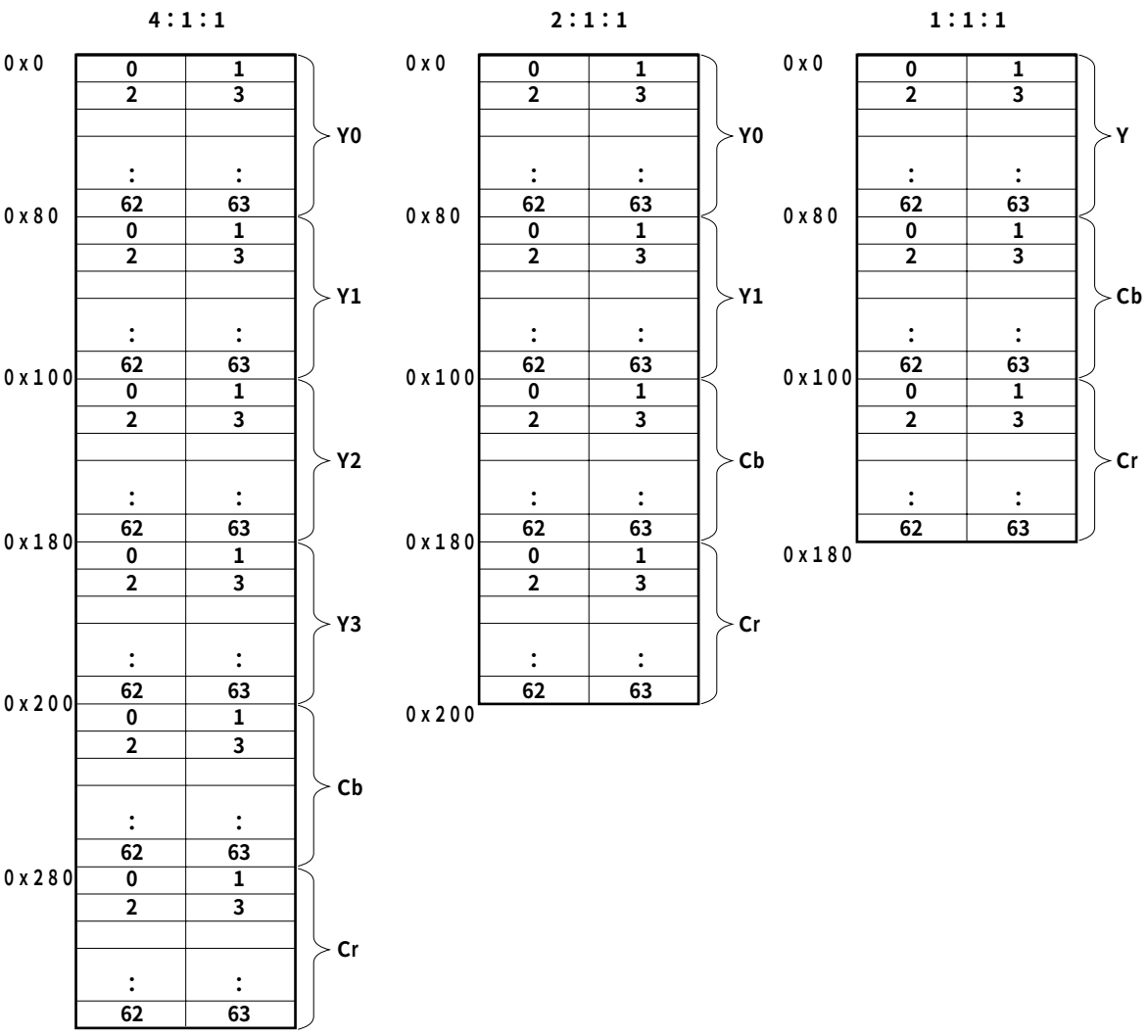
保守／廃止

2.6.3　画像データ用バッファ

1MCU分の画像データは，MCUバッファ（AP70732-B03の場合は構造体の最後のメンバ，AP705100-B03の場合は内部RAMワーク・エリア）に入れられます。

データの入る位置は次に示すとおりに決められています。

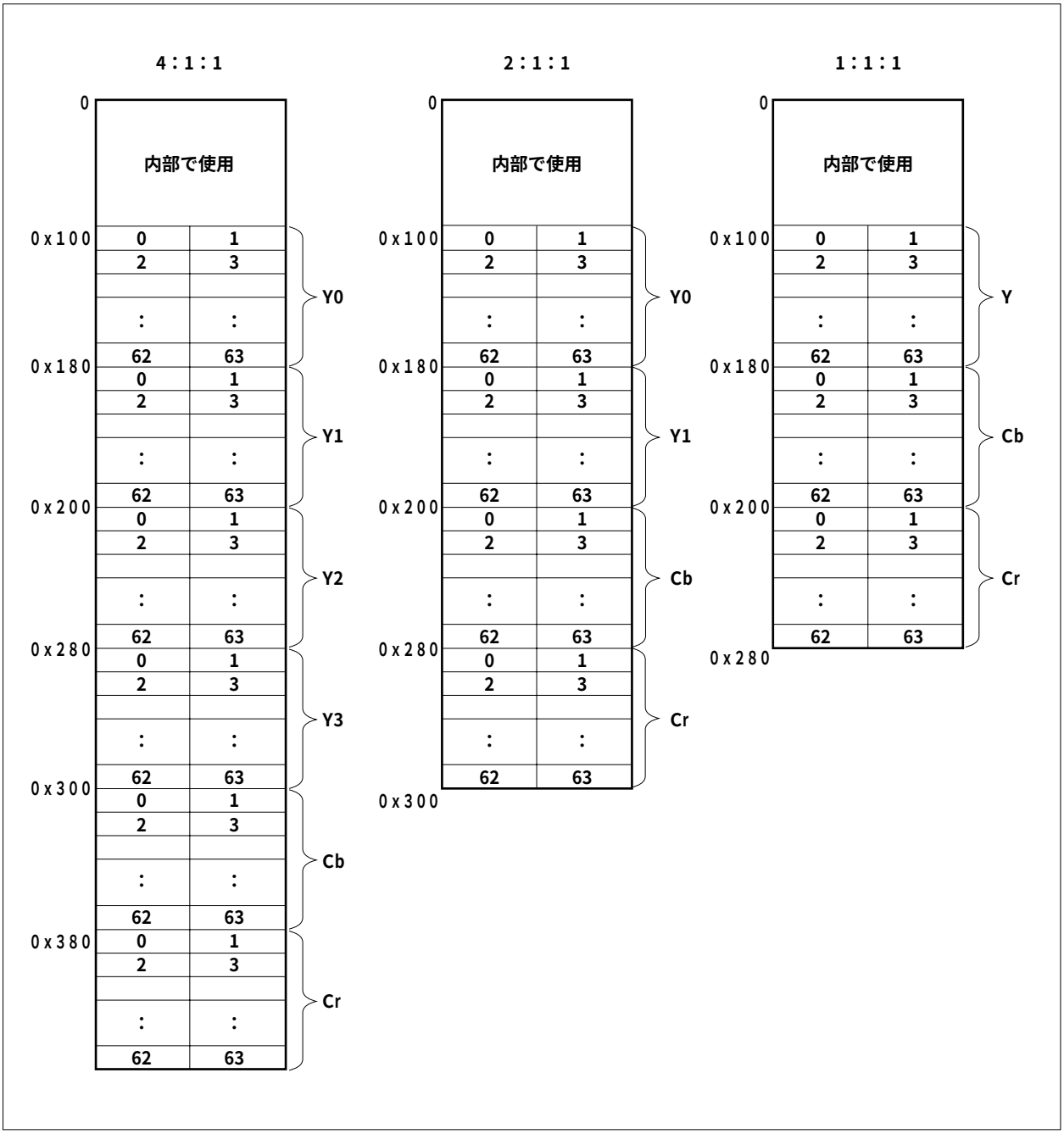
図２－42　MCUバッファの画像データ（AP70732-B03）



１画素の１色素分のデータ（８ビット）は，MCUバッファ内では，上位８ビットに０が入れられ，unsigned short型（16ビット）として扱われます。

保守／廃止

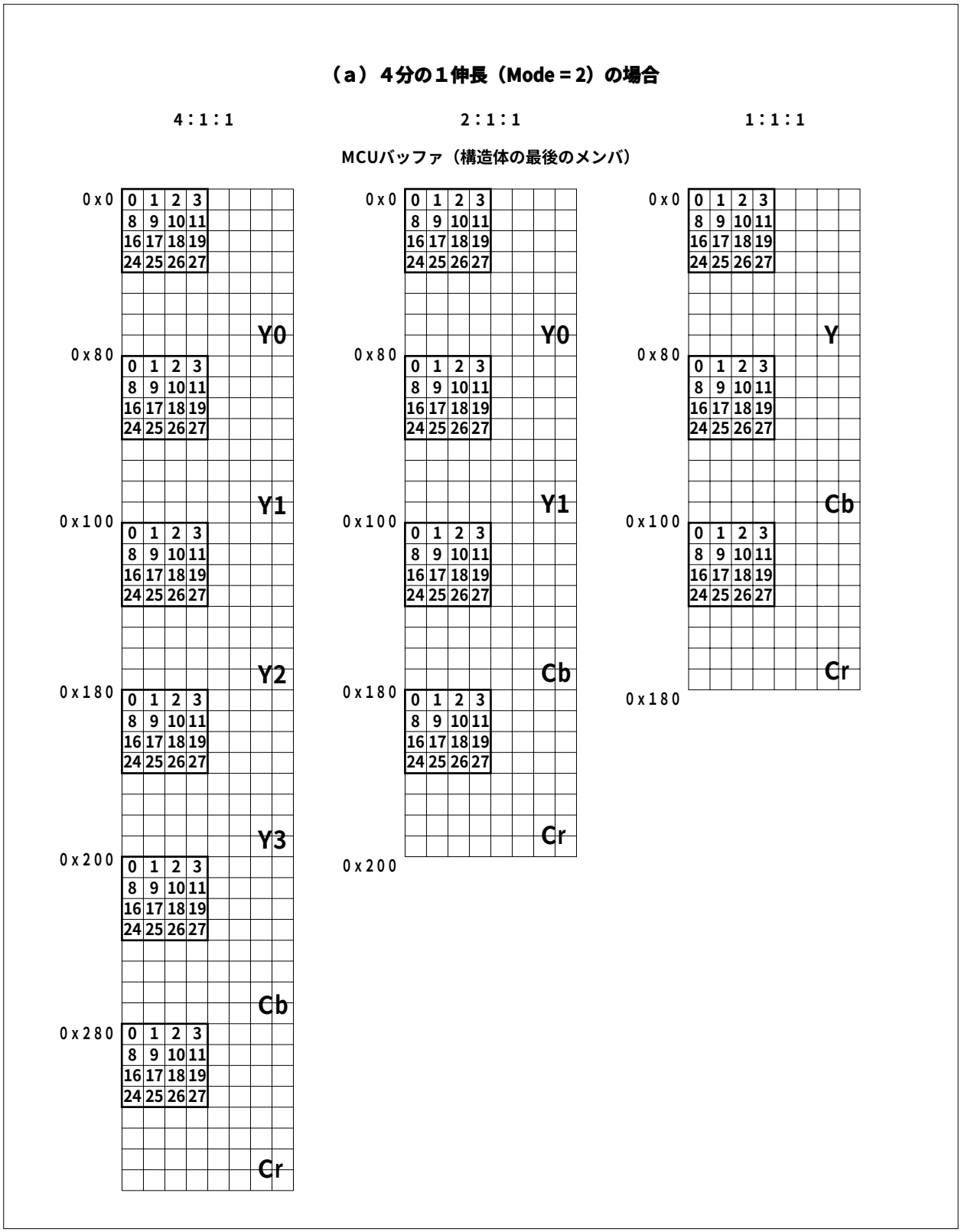
図２－43　内部RAMの画像データ用バッファ（AP705100-B03）



保守／廃止

縮小して伸長するモードの場合，同じ領域（縮小しない場合）のうち次の部分（太枠部分）が意味を持ちます。

図２－44　縮小伸長モードの場合の画像データ（AP70732-B03）（1/3）



保守／廃止

図２－44　縮小伸長モードの場合の画像データ（AP70732-B03）（2/3）

（b）16分の１伸長（Mode = 3）の場合

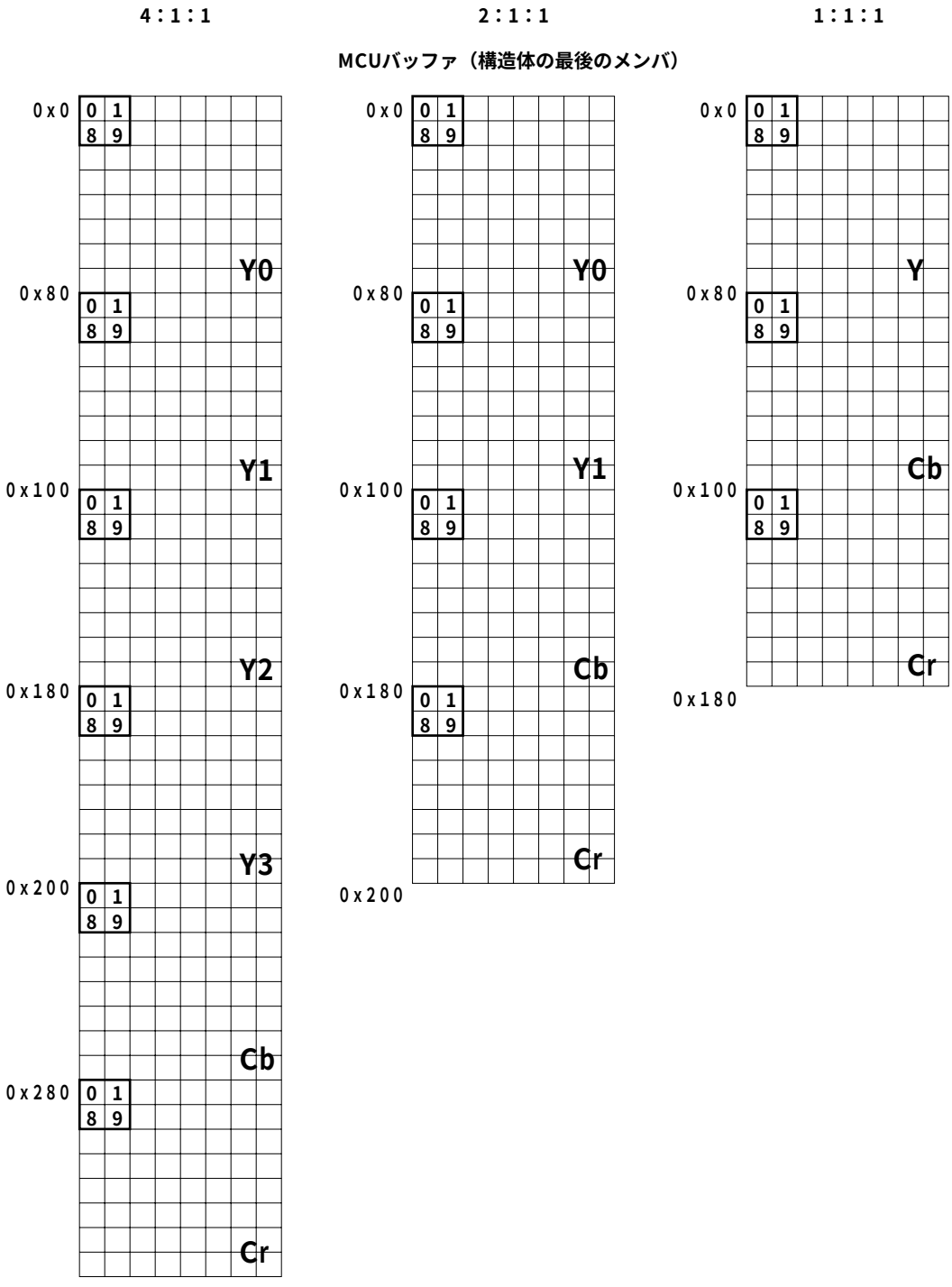
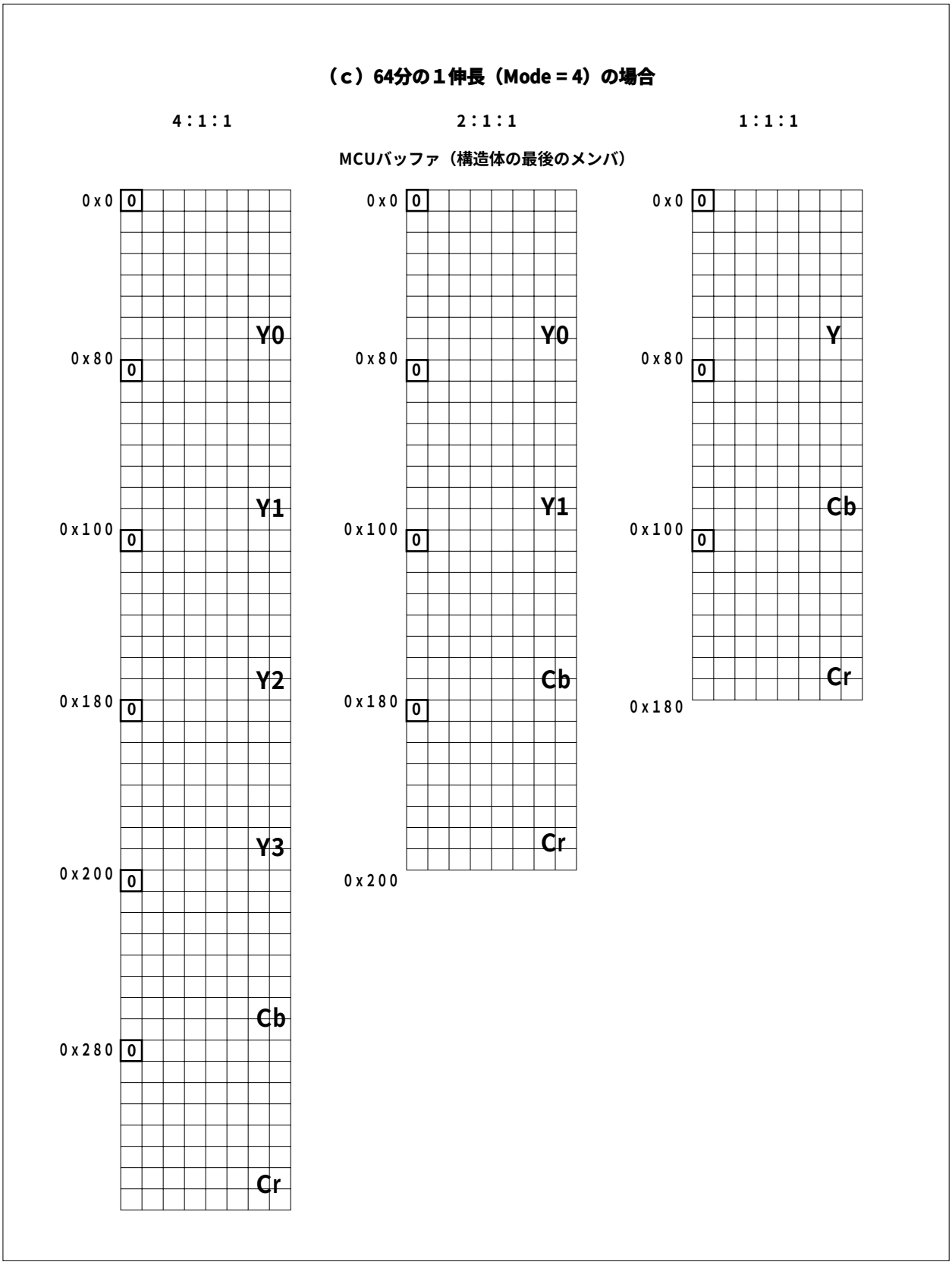


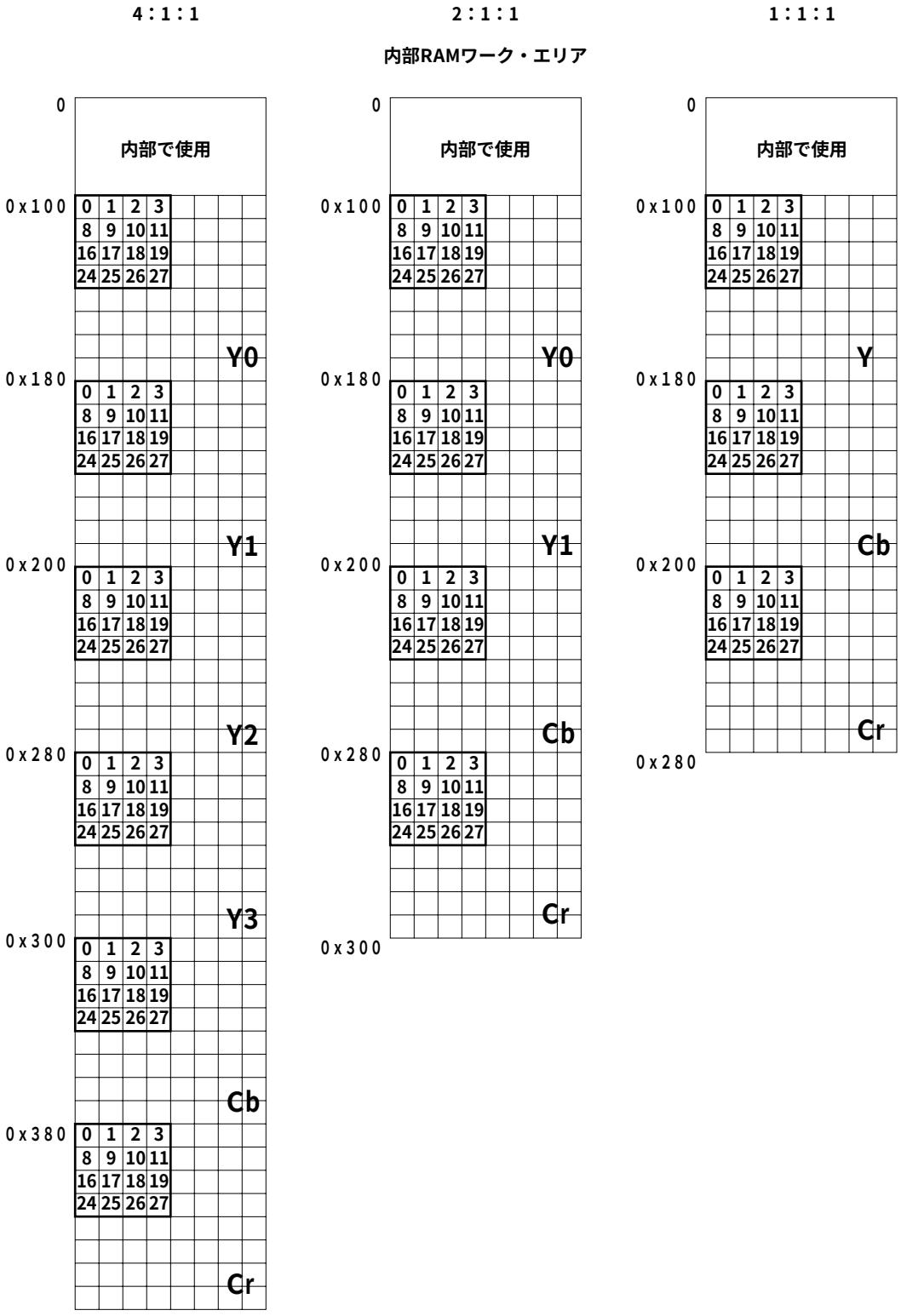
図２－44　縮小伸長モードの場合の画像データ（AP70732-B03）（3/3）



保守／廃止

図２－45　縮小伸長モードの場合の画像データ（AP705100-B03）（1/3）

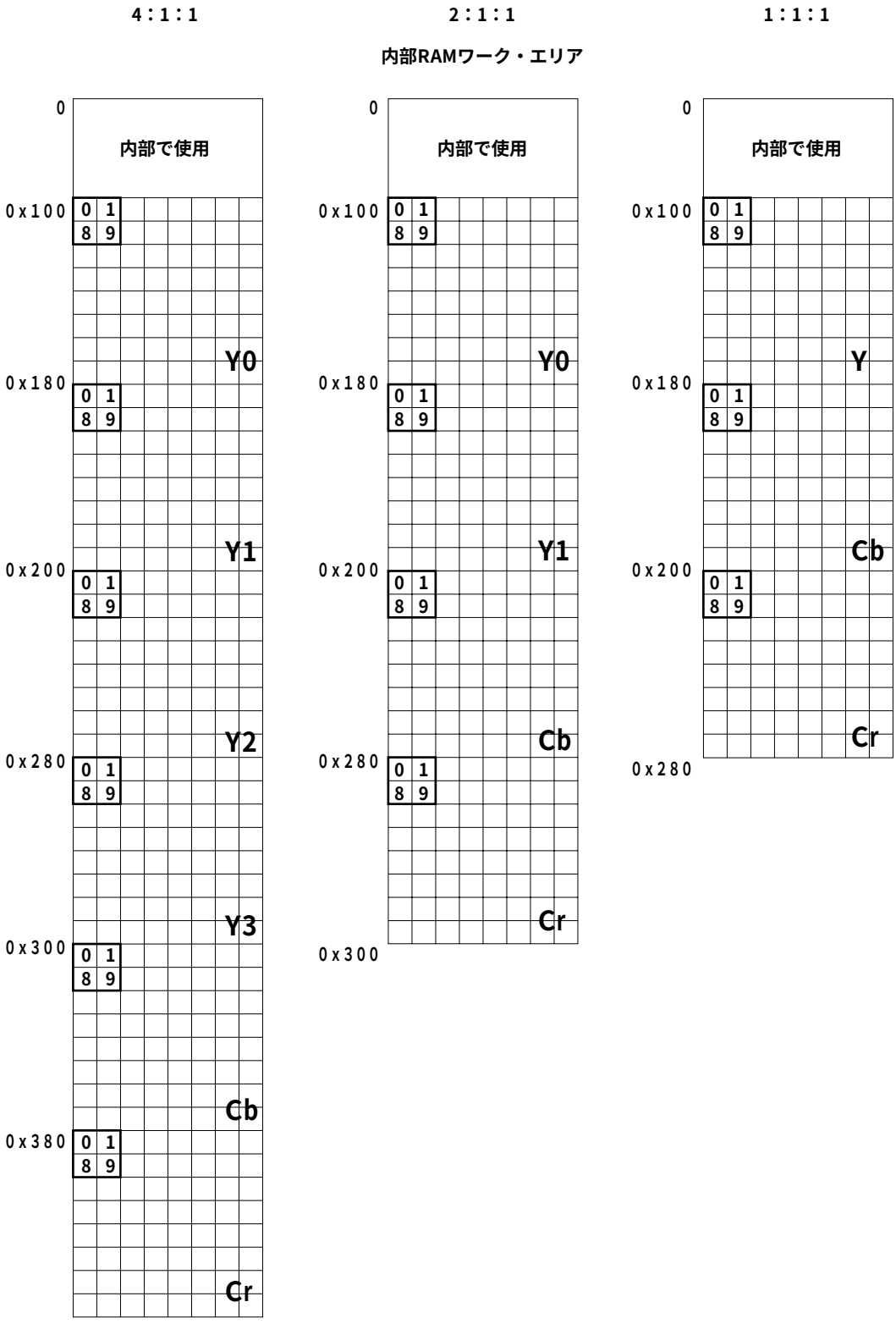
（ａ）４分の１伸長（Mode＝2）の場合



保守／廃止

図２－45　縮小伸長モードの場合の画像データ（AP705100-B03）（2/3）

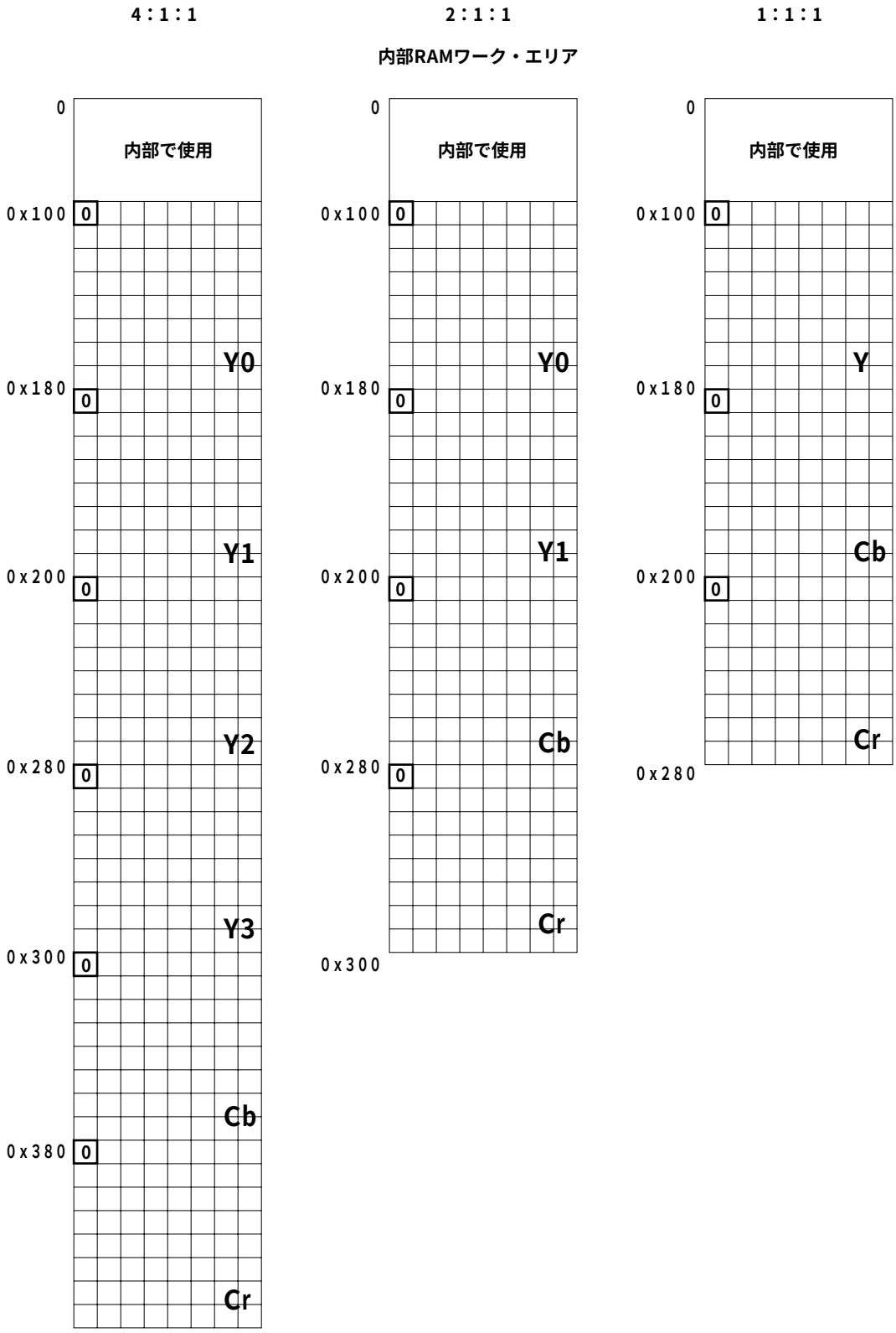
（b）16分の１伸長（Mode = 3）の場合



保守／廃止

図２－45　縮小伸長モードの場合の画像データ（AP705100-B03）（3/3）

（ｃ）64分の１伸長（Mode＝4）の場合



保守／廃止

２．６．４　カスタマイズする場合に必要な関数

圧縮処理で必要な関数は，VRAMから1MCU分を内部RAMワーク・エリアのデータを代入するべき場所にストアする関数です。サンプル比によって異なる関数を作らなければなりません。

表２－47　圧縮のカスタマイズ対象関数

関数名	サンプル比
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）

一方，基本伸長の方は，縮小モードがあるためさらに多くの関数を作らなければなりません。

表２－48　基本伸長のカスタマイズ対象関数

関数名	サンプル比
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）〔４分の１縮小〕
void jpeg_putMCU412（DJINFO＊jinfo）	4：1：1（H：V＝4：1）〔４分の１縮小〕
void jpeg_putMCU212（DJINFO＊jinfo）	2：1：1〔４分の１縮小〕
void jpeg_putMCU112（DJINFO＊jinfo）	1：1：1〔４分の１縮小〕
void jpeg_putMCU224（DJINFO＊jinfo）	4：1：1（H：V＝2：2）〔16分の１縮小〕
void jpeg_putMCU414（DJINFO＊jinfo）	4：1：1（H：V＝4：1）〔16分の１縮小〕
void jpeg_putMCU214（DJINFO＊jinfo）	2：1：1〔16分の１縮小〕
void jpeg_putMCU114（DJINFO＊jinfo）	1：1：1〔16分の１縮小〕
void jpeg_putMCU228（DJINFO＊jinfo）	4：1：1（H：V＝2：2）〔64分の１縮小〕
void jpeg_putMCU418（DJINFO＊jinfo）	4：1：1（H：V＝4：1）〔64分の１縮小〕
void jpeg_putMCU218（DJINFO＊jinfo）	2：1：1〔64分の１縮小〕
void jpeg_putMCU118（DJINFO＊jinfo）	1：1：1〔64分の１縮小〕

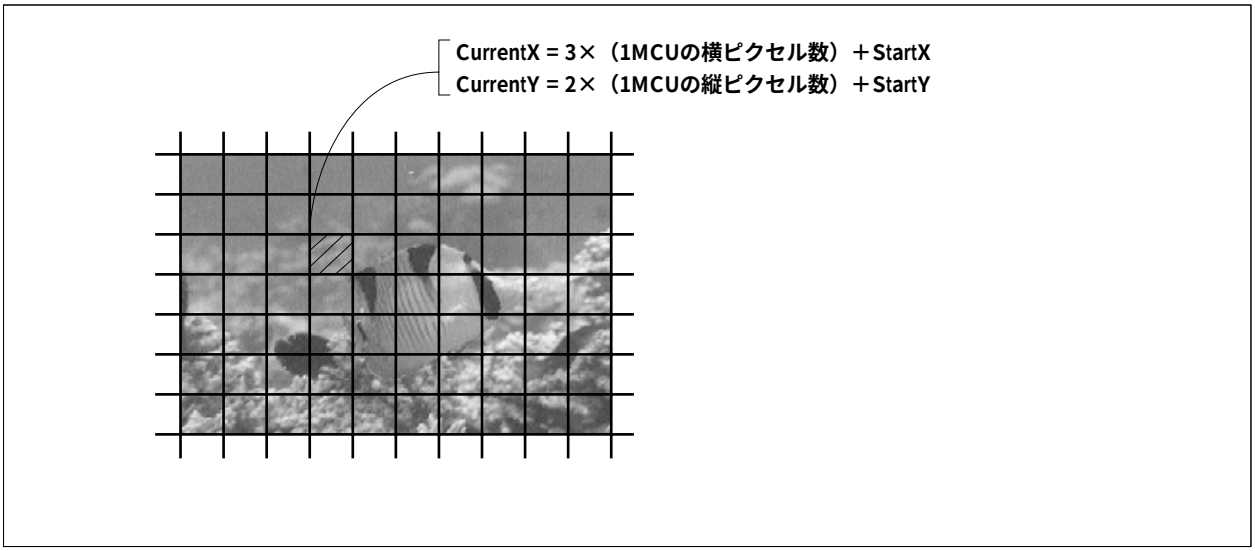
それぞれの関数の引き数はJPEG構造体（圧縮ではCJINFO，伸長ではDJINFO）１つだけです。特にアセンブラで記述する場合は，r20-r29，spの内容は使う前に保存し，関数終了時に元に戻すなどのＣ言語の規約に沿うようにしてください。

それぞれの関数を作る際に必要な情報は次のものです。

表２－49　カスタマイズに必要な情報

メンバ	意　　味
VRAM_Bptr	VRAMの先頭アドレス
CurrentX（short型）	VRAMの横ピクセル座標
CurrentY（short型）	VRAMの縦ピクセル座標
IRAM_Buff_Bptr	内部RAMワーク・エリアの先頭アドレス

図２－46　CurrentX/CurrentY



★ 第３章　プログレッシブ対応追加ライブラリ仕様

追加ライブラリはISO/IEC　10918-2の画像データの適合性検査を行い，A, C, E, G, Kの検査データについて正常に伸長することを確認しています。

3.1　機　　能

AP705100-B03で用意されている追加ライブラリ群を使うことにより実現できる伸長処理の主な機能について説明します。

なお，追加伸長処理を行う際の演算精度については，基本ライブラリの演算精度と同じです。**2.1.3　演算精度**を参照してください。

3.1.1　プログレッシブ・フォーマットのサンプリングとMCU

JPEGが処理を行う最小単位をMCU（Minimum Coded Unit）と呼び，そのMCUをY/Cb/Crに分離し，８×８ピクセル単位にしたものをブロックと呼びます（**1.2.1（3）サンプリングとMCU参照**）。

色コンポーネント数が3の場合，サンプル比４：１：１（H：V = ２：２）では，16×16ピクセルが一つのMCUとなります。このサンプル比のMCUはY（輝度）成分4ブロック，Cb（色差）成分１ブロック，Cr（色差）成分１ブロックからなります。

このようなMCUの大きさとブロックの個数は，SOFマーカ・セグメント内に含まれるHi, Viの値により次のように規定されます。

MCUの横ピクセル数はMax（H_0, H_1, …）×８
MCUの縦ピクセル数はMax（V_0, V_1, …）×８
$\Sigma_i Hi \times Vi \leqq 10$（ISO/IEC 10918-1による制限）
$\Sigma_i Hi \times Vi \leqq 20$（ISO/IEC 10918-3による拡張フォーマットの制限）

たとえば，サンプル比４：１：１（H：V = ２：２）の場合，Hi，Viの値は次のようになります。

$H_0 = 2$，$V_0 = 2$
$H_1 = 1$，$V_1 = 1$
$H_2 = 1$，$V_2 = 1$

これを式にあてはめると次のようになり，MCUの大きさは16×16ピクセルとなります。

Max（H_0, H_1, H_2）×8 = 16
Max（V_0, V_1, V_2）×8 = 16

次に，色コンポーネント数が4，サンプル比１：２：３：４のように複雑なサンプル比の場合を考えます。Hi, Viの値が次のようであったとします。

$H_0 = 1$，$V_0 = 1$
$H_1 = 1$，$V_1 = 2$
$H_2 = 3$，$V_2 = 1$
$H_3 = 1$，$V_3 = 4$

これを式にあてはめると次のようになり，MCUの大きさは24×32ピクセルとなります。

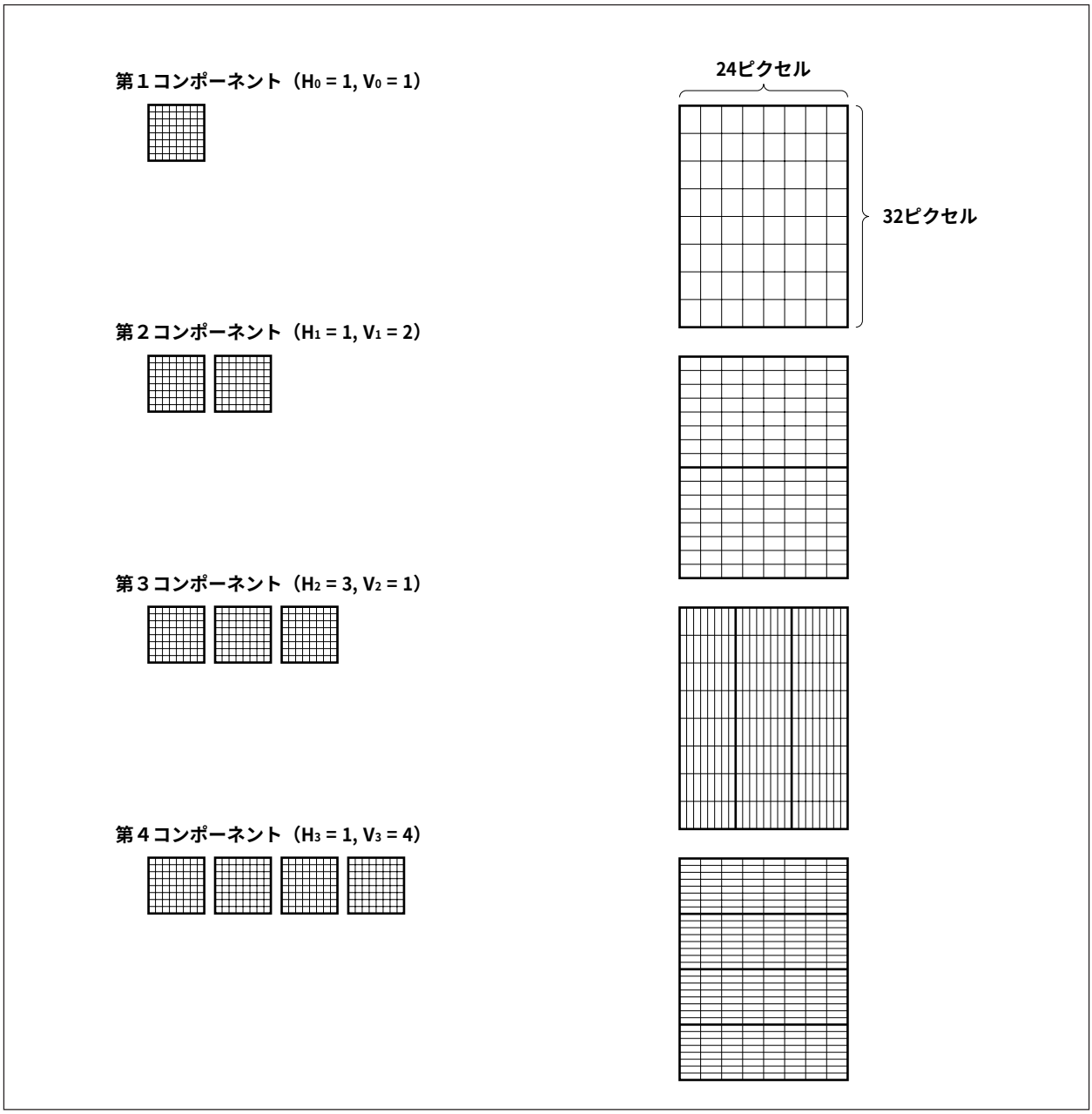
Max（H_0，H_1，H_2，H_3）×8 = 24

Max（V_0，V_1，V_2，V_3）×8 = 32

このとき，第１コンポーネント（H_0 = 1, V_0 = 1）は，24×32ピクセルに引き伸ばされます。

第２コンポーネント（H_1 = 1, V_1 = 2）は２ブロックで24×32ピクセルとなるので，１ブロックでは24×16ピクセルに引き伸ばされます。同様に，第３コンポーネント（H_2 = 3, V_2 = 1）は１ブロックが8×32ピクセルに，第４コンポーネント（H_3 = 1, V_3 = 4）は１ブロックが24×8ピクセルに引き伸ばされます。

図３－１　サンプリングとMCU（サンプル比１：２：３：４の場合）



保守／廃止

3.1.2　色空間について

AP705100-B03追加ライブラリでは，色空間を次のように規定しています。

・単色フォーマット：輝度（JFIF規定準拠）
・３色フォーマット：YCbCr（JFIF規定準拠）
・４色フォーマット：CMYK, YCCK

入力したJPEGファイルが4色フォーマットの場合，ファイル・ヘッダの情報からCMYK形式かYCCK形式かを自動判別し，次の計算式で処理を行います。

（1）CMYK形式の場合

C：第１コンポーネント，M：第２コンポーネント，Y：第３コンポーネント，K：第４コンポーネント，R, G, B：出力の（R：G：B）

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；

備考　VRAM形式がRGBではなくYCbCrの場合，この式で得られた（R：G：B）の値を（Y：Cb：Cr）に変換します。

（2）YCCK形式の場合

Yin：第１コンポーネント，C1in：第２コンポーネント，C2in：第３コンポーネント，Kin：第４コンポーネント，Yout，Cbout，Crout：出力の（Y：Cb：Cr）

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

備考　VRAM形式がYCbCrではなくRGBの場合，この式で得られた（Y：Cb：Cr）の値を（R：G：B）に変換します。

保守／廃止

3.1.3　プログレッシブの逆DCT変換

８×８要素からなる１ブロックに対してDCT変換した結果の64要素をDCT係数と呼びます。

DCT変換した結果のDCT係数をジグザグ順に並べなおすと低周波成分→高周波成分と並びます。最初の１要素だけをDC成分と呼び，そのブロックの平均の色レベルを示します。その他の63要素はAC成分と呼びます（DCT変換については，**1.2.1（4）DCT変換**を参照してください）。

64要素のDCT係数に対して逆DCT変換を行うと元の画像を復元することができます。AC1-AC63の部分をすべてゼロにしたような要素を逆DCT変換し全体の画像を再現すると，８×８単位のモザイクのような画像が得られます。DC成分とAC1-AC5の成分のみを有効にしてそれ以外をゼロにしたデータからは，ボンヤリとした画像が得られます。これがプログレッシブ・アルゴリズムの基本的な考え方となっています。

3.1.4　スキャン

JPEGファイルの圧縮データ部分は，SOSセグメントから始まる「スキャン」と呼ばれる単位で分割されています（SOSセグメントについては，**図１－24　SOSセグメント**を参照してください）。

スキャン・ヘッダであるSOSセグメントには，DCT係数の開始番号を指定するSs領域，DCT係数の終了番号を指定するSe領域があります。

ベースラインのようにDC成分-AC63成分を一度に圧縮する場合はSs = 0，Se = 63（= 0x3F）となります。プログレッシブでDC成分のみを圧縮するスキャンではSs = 0，Se = 0のように指定されます。

プログレッシブ・フォーマットでは通常，DCT係数をスキャンごとに次のように分割して圧縮します。

第１スキャンにDC成分のみ，第２スキャンにAC1-AC5，…

同じプログレッシブ・フォーマットの中でも，それぞれのDCT係数を分割しない方法をスペクトラル・セクションと呼びます。また，それぞれのDCT係数の値を次のように上位ビット，下位ビットに分割する方式をサクセッシブ・アプロキシメーションと呼びます。

第１スキャンにDC成分の２ビット目以上
第２スキャンにAC1-AC5の２ビット目以上，…

サクセッシブ・アプロキシメーション符号化では‘ ２ビット目以上 ’を表すのにSOSセグメントでAh = 0，Al = 2と指定します。‘ ０ビット目-１ビット目 ’を表すのにAh = 2，Al = 0と指定します。

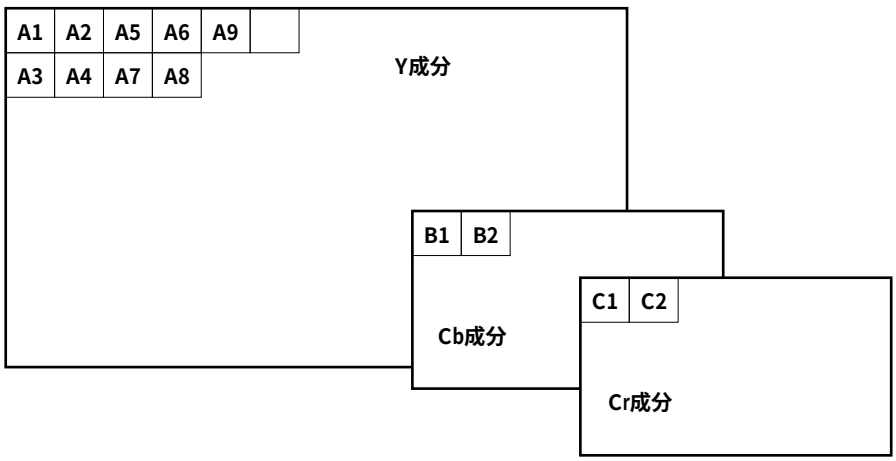
3.1.5　MCU符号化順序

プログレッシブ・フォーマットではDC成分とAC成分は別のスキャンで符号化し，DC成分のスキャンはAC成分のスキャンよりも先に符号化しなければなりません。また，DC成分スキャンについては，Y成分Cb成分Cr成分をまとめて1つのスキャンに符号化するブロック・インタリーブが許可されていますが，AC成分スキャンでは単独の色コンポーネントごとに符号化しなければなりません。

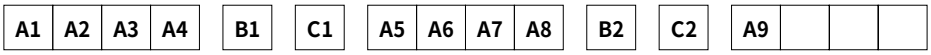
たとえば，サンプル比４：１：１（H：V = ２：２）でブロック・インタリーブを許可したフォーマットでは，MCUの符号化順序は次のようになります。

保守／廃止

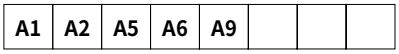
図３－２　MCUの符号化順序（４：１：１（H：V＝２：２），ブロック・インタリーブ・フォーマットの場合）



ブロック・インタリーブの符号化順



AC係数（Y成分）の符号化順



備考　　は１ブロックを表します。

スキャン内に含まれる色コンポーネント数が1個だけの場合，サンプル比によらず，スキャン順序はブロック単位で左から右へ，上から下へ行います（ISO/IEC 10918-1準拠）。

保守／廃止

３.１.６　追加伸長時のオプション

追加伸長時の主なオプションを示します。

（１）追加伸長処理の強制終了

実行中の追加伸長処理を強制終了できます。

このオプションは，JPEGEXINFO構造体で指定します。

（２）描画タイミング

伸長処理のどの段階で描画を行うかを設定できます。

このオプションは，JPEGEXINFO構造体で指定します。

（３）スタッフィング・ビット，スタッフィング・バイト

JPEGファイル内のスタッフィング・ビットの値をチェックできます。また，JPEGファイル内のスタッフィング・バイトを認めるかどうかを設定できます。

これらのオプションは，JPEGEXINFO構造体で指定します。

（４）伸長時のパス回数

追加伸長処理のパス回数を設定できます。

このオプションは，JPEGEXINFO構造体で指定します。

（５）DNLマーカ

DNLマーカ（ライン数の再定義）の存在を認めるかどうかを設定できます。

このオプションは，JPEGEXINFO構造体で指定します。

（６）画像拡大／縮小

画像の拡大／縮小伸長を設定できます。

このオプションは，JPEGEXINFO構造体で指定します。

（７）クリッピング

伸長時のクリッピングをピクセル単位で設定できます。

このオプションは，JPEGEXVIDEO構造体で指定します。

保守／廃止

３.２　追加ライブラリのリンク

リンク時にライブラリを選択できます。

追加ライブラリで選択できるライブラリを表３－１に示します。ライブラリ選択については，サンプルのmakefileを参考にしてください。

表３－１　リンク時に指定可能なライブラリ

ライブラリ名	ライブラリの内容
libjprg.a	追加ライブラリ本体
libjprgd.a	libjprg.aのディバグ・バージョン
libjprog.a	putMCUのシンボルを解決するためのライブラリ

（１）通常のライブラリ指定

NECライブラリ

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

GHSライブラリ

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

（２）ディバグ・ライブラリ指定

ディバグ・ライブラリを使用する場合には，libjprg.aの代わりにlibjprgd.aを指定します。

NECライブラリ

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

GHSライブラリ

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

（３）JPEGEXputMCU関数を使用しない場合の指定

JPEGEXputMCU関数で使用されるシンボルは，libjprog.aで定義されています。
JPEGEXputMCU関数を使用しない場合，libjprog.aをリン力で指定する必要はありません。
JPEGEXputMCU関数を使用しない場合には，次の２つがあります。

（ａ）JPEGEXputMCU関数を使用せず，次に示す関数がソース・ファイルで定義されている場合には，libjprog.aをリン力で指定する必要はありません。

- jpeg_putMCU221
- jpeg_putMCU411
- jpeg_putMCU211
- jpeg_putMCU111

（ｂ）基本ライブラリのputMCU関数を使用する場合には，libjprog.aをリン力で指定する必要はありません。
JPEGEXINFO構造体のメンバPolicy（**３.４.３（３）Policy**参照）で必要なオプションを設定したうえで，リンク時に次のように指定します。この例では，YCbCrのputMCUライブラリを使用しています。

NECライブラリ

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

GHSライブラリ

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

保守／廃止

３.３　追加ライブラリの構造体

追加ライブラリの伸長処理に用いられる構造体について説明します。

３.３.１　JPEGEXINFO構造体

JPEGEXINFO構造体で追加伸長処理のパラメータ設定を行います。この構造体の先頭アドレスは，追加伸長メイン関数に引き数として渡されます。JPEGEXINFO構造体メンバの設定については，**３.４.３　JPEGEXINFO構造体のパラメータ設定**を参照してください。

表３－２　JPEGEXINFO構造体

メンバ	型	内　　容	IN/OUT
TaskID	int	タスクのID番号	IN
Mode	int	通常伸長処理／伸長処理の強制終了の選択	IN
Policy	int	伸長処理のオプション設定	IN
ratio	int	画像拡大／縮小率設定	IN
ErrorState	int	エラー・ステータス番号	OUT
Work	struct JPEGEXWORK	JPEGEXWORK構造体先頭アドレス	IN
Video	struct JPEGEXVIDEO	JPEGEXVIDEO構造体先頭アドレス	IN
Inf	struct JPEGEXFrmINFO	JPEGEXFrmINFO構造体先頭アドレス	OUT

３.３.２　JPEGEXWORK構造体

JPEGEXWORK構造体で，追加ライブラリが使用可能なワーク・エリアを指定します。この構造体の先頭アドレスをJPEGEXINFO構造体のメンバWorkに設定します。

JPEGEXWORK構造体メンバの設定については，**３.４.４　JPEGEXWORK構造体のパラメータ設定**を参照してください。

表３－３　JPEGEXWORK構造体

メンバ	型	内　　容	IN/OUT
Work1	unsigned int	ワーク・エリア先頭アドレス	IN
Work1Len	unsigned int	ワーク・エリア・サイズ（バイト数）	IN
Work1Used	unsigned int	使用ワーク・エリア・サイズ（バイト数）	OUT
Work2	unsigned int	ワーク・エリア先頭アドレス	IN
Work2Len	unsigned int	ワーク・エリア・サイズ（バイト数）	IN
Work2Used	unsigned int	使用ワーク・エリア・サイズ（バイト数）	OUT

保守／廃止

３.３.３　JPEGEXVIDEO構造体

JPEGEXVIDEO構造体は描画関連の設定を行う構造体です。この構造体の先頭アドレスをJPEGEXINFO構造体のメンバVideoに指定します。

VRAM関連のメンバ（VRAMxxx）には，VRAMの構造を規定する値を設定します。

追加伸長処理時にクリッピングを行う場合，クリッピング関連のメンバ（Clipxxx）に適切な値を設定することによりクリッピングが行われます。クリッピングを行わない場合には，クリッピング関連メンバ（Clipxxx）には表３－５に示すダミー値を設定してください。

JPEGEXVIDEO構造体のメンバ設定については，**３.４.５　JPEGEXVIDEO構造体のパラメータ設定**を参照してください。

なお，JPEGEXputMCU関数を作成する場合には，JPEGEXVIDEO構造体の各メンバには表３－５に示すダミー値を設定してください。

表３－４　JPEGEXVIDEO構造体

メンバ	型	内　　容	IN/OUT
VRAMAddress	unsigned char*	VRAM先頭アドレス	IN
VRAMWidth	int	VRAMの横幅	IN
VRAMHeight	int	VRAMの縦幅	IN
VRAMPixel	int	VRAMの横1ピクセル分のアドレス差	IN
VRAMLine	int	VRAMの縦1ピクセル分のアドレス差	IN
VRAMGap0	int	Yピクセル（またはRピクセル）のバイト・オフセット	IN
VRAMGap1	int	Cbピクセル（またはGピクセル）のバイト・オフセット	IN
VRAMGap2	int	Crピクセル（またはBピクセル）のバイト・オフセット	IN
ClipStartX	int	クリッピング開始位置（Ｘ座標） クリッピングを行わない場合は，ダミー値０を設定	IN
ClipStartY	int	クリッピング開始位置（Ｙ座標） クリッピングを行わない場合は，ダミー値０を設定	IN
ClipWidth	int	クリッピング横サイズ（ピクセル） クリッピングを行わない場合は，ダミー値0x7FFFFFFFを設定	IN
ClipHeight	int	クリッピング縦サイズ（ピクセル） クリッピングを行わない場合は，ダミー値0x7FFFFFFFを設定	IN

表３－５　JPEGEXVIDEO構造体のダミー設定値

メンバ	ダミー値	メンバ	ダミー値
VRAMAddress	設定の必要なし	VRAMGap1	設定の必要なし
VRAMWidth	0x7FFFFFFF	VRAMGap2	設定の必要なし
VRAMHeight	0x7FFFFFFF	ClipStartX	0
VRAMPixel	設定の必要なし	ClipStartY	0
VRAMLine	設定の必要なし	ClipWidth	0x7FFFFFFF
VRAMGap0	設定の必要なし	ClipHeight	0x7FFFFFFF

保守／廃止

３.３.４　JPEGEXBUFF構造体

JPEGEXBUFF構造体は，JPEGファイルを格納するJPEGバッファを指定する構造体です。JPEGファイル取得関数（JPEGEXGetJpegStream）には，この構造体の先頭アドレスを引き数として渡します。JPEGEXBUFF構造体のメンバ設定については，**３.５.１　JPEGファイル取得関数**を参照してください。

表３－６　JPEGEXBUFF構造体

メンバ	型	内　　容	IN/OUT
TaskID	int	タスクID番号	OUT（上書き可）
JPEGBUFF	unsigned char*	JPEGバッファの先頭アドレス	IN
JPEGBUFFLEN	unsigned int	JPEGバッファのサイズ（バイト数）	IN

３.３.５　JPEGEXMCUSTR構造体

JPEGEXMCUSTR構造体には，MCUバッファの構造を規定するパラメータ，およびデータ出力に関するパラメータが追加ライブラリにより設定されます。この構造体の先頭アドレスを，ＭＣＵデータ出力関数（JPEGEXputMCU）の第４引き数として，また，JPEGEXpset関数の第１引き数として指定します。

表３－７　JPEGEXMCUSTR構造体

メンバ	型	内　　容	IN/OUT
component	unsigned char	色コンポーネント数 １：輝度のみ ３：Y, Cb, Crの３色 ４：４色	OUT
adobeflag	char	４色時の出力方式（4色時のみ有効） ０：CMYK １：YCbCr ２：YCCK	OUT
hf［4］	unsigned char	MCUバッファの横方向ブロック数	OUT
vf［4］	unsigned char	MCUバッファの縦方向ブロック数	OUT
VRAMAddress	unsigned char*	JPEGEXVIDEO構造体の設定値が追加ライブラリにより格納される	OUT
VRAMWidth	int		OUT
VRAMHeight	int		OUT
VRAMPixel	int		OUT
VRAMLine	int		OUT
VRAMGap0	int		OUT
VRAMGap1	int		OUT
VRAMGap2	int		OUT
ClipStartX	int	実際にクリッピングされるべきサイズが追加ライブラリにより格納される	OUT
ClipStartY	int		OUT
ClipWidth	int		OUT
ClipHeight	int		OUT
hfMax	unsigned char	MCUの横幅 （MCUの横サイズはhfMax×８ピクセル）	OUT
vfMax	unsigned char	MCUの縦幅 （MCUの縦サイズはvfMax×８ピクセル）	OUT

保守／廃止

JPEGEXMCUSTR構造体メンバcomponent，hf［4］，vf［4］に設定された値により，MCUバッファの構造が規定されます。図３－３にその例を示します。

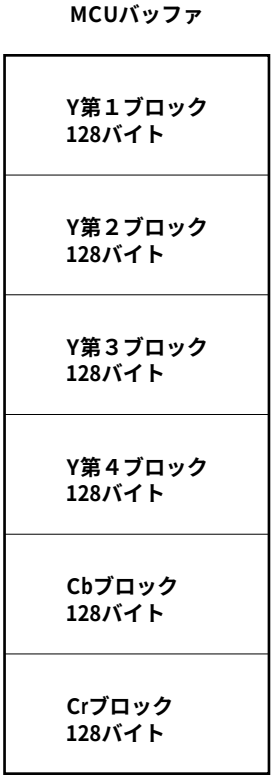
図３－３　JPEGEXMCUSTR構造体のメンバ設定値とMCUバッファの構造

JPEGEXMCUSTR構造体のメンバ設定値（例）

component = 3

hf［0］= vf［0］= 2

hf［1］= vf［1］= hf［2］= vf［2］= 1



保守／廃止

３.４　追加伸長処理の実行

３.４.１　追加伸長メイン関数

分　類　追加伸長処理系

関数名　JPEGEXdecode

形　式　int　JPEGEXdecode（struct JPEGEXINFO* JPInfo）；

引き数　JPEGEXINFO構造体の先頭アドレス

返り値　返り値の内容を表３－８に示します。

表３－８　追加伸長メイン関数の返り値

返り値		内　　容
定義名	数値	
DecodeStatusComplete	1	正常終了
DecodeStatusTerminate	2	強制終了
DecodeStatusError	－1	エラー終了
DecodeStatusNotRunning	－2	強制終了対象のプロセスは動作中ではない

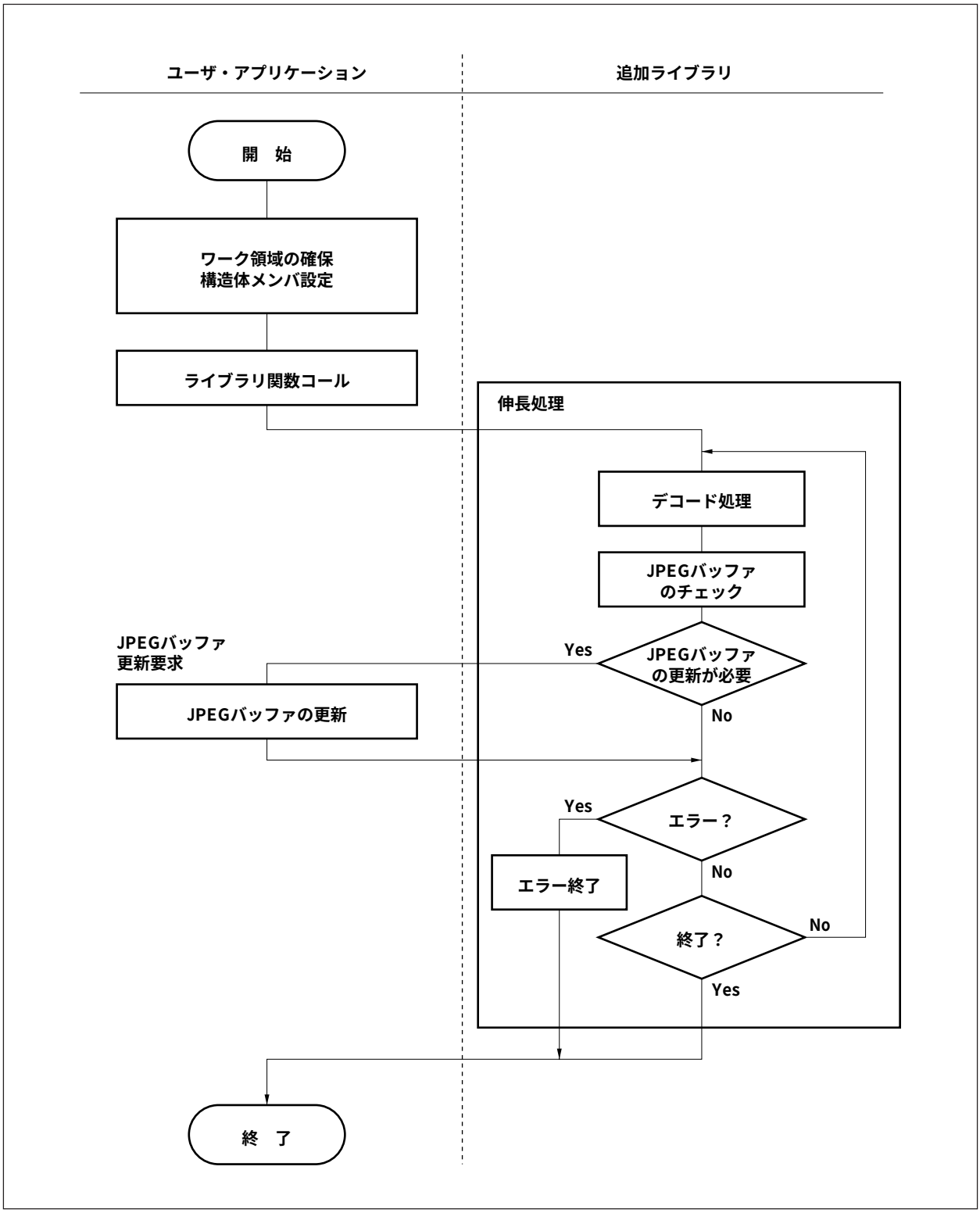
この関数をコールする前に，JPEGEXINFO構造体，JPEGEXWORK構造体，JPEGEXVIDEO構造体のメンバ設定を行う必要があります。

保守／廃止

３.４.２　追加伸長処理フロー

追加伸長処理の基本的な流れを示します。

図３－４　追加伸長処理フロー



保守／廃止

３.４.３　JPEGEXINFO構造体のパラメータ設定

伸長メイン関数をコールする前に，JPEGEXINFO構造体で追加伸長処理に必要なパラメータの設定を行います。

表３－９　JPEGEXINFO構造体

メンバ	型	内　　容	IN/OUT
TaskID	int	タスクのID番号	IN
Mode	int	通常伸長処理／伸長処理の強制終了の選択	IN
Policy	int	伸長処理のオプション設定	IN
ratio	int	画像拡大／縮小率設定	IN
ErrorState	int	エラー・ステータス番号	OUT
Work	struct JPEGEXWORK	JPEGEXWORK構造体先頭アドレス	IN
Video	struct JPEGEXVIDEO	JPEGEXVIDEO構造体先頭アドレス	IN
Inf	struct JPEGEXFrmINFO	JPEGEXFrmINFO構造体先頭アドレス	OUT

（１）TaskID

このTaskIDの値は，マルチタスク環境下で複数のタスクを起動する場合に，それらのタスクを区別するためのものです。各タスクごとに個別のJPEGEXINFO構造体が必要となりますので，それぞれのTaskIDには異なる値を設定してください。シングルタスクで使用する場合には，設定する必要はありません。

なお，この値はJPEGEXBUFF構造体のメンバTaskIDに代入されます。

（２）Mode

通常の伸長処理を行うか，伸長処理の強制終了を指示するかを設定します。

表３－10　追加伸長処理のモード設定

定義名	数値	内　　容
ModeStart	1	通常の伸長モードです。
ModeTerminate	－1	実行中の伸長処理を強制終了させます。

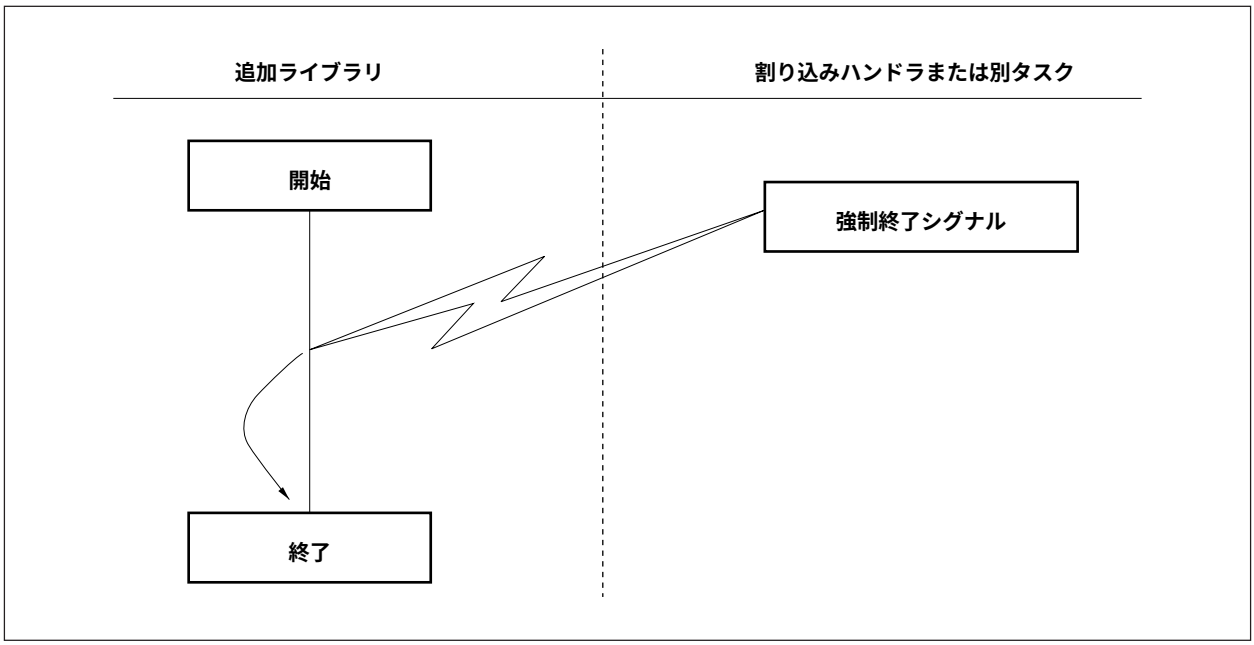
普通にJPEG伸長を開始する場合にはModeStartを指定してください。

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

ModeTerminateを指定すると，実行中の追加伸長処理を強制終了できます。伸長動作中のライブラリで指定したJPEGEXINFO構造体と同じ構造体を使って，割り込みハンドラから，あるいは，OSを使っている場合には別タスクからJPEGEXdecode関数を関数コールすることによって，実行中の追加ライブラリに強制終了を促すシグナルを送ります。

保守／廃止

図３－５　ModeTerminate指定時の追加伸長処理強制終了



保守／廃止

（３）Policy

Policyは図３－６に示す２バイトの領域に，オプション・ビットを備えています。

Policyでは，表３－11に示すオプションの設定を行います。

図３－６　Policyのビット構成

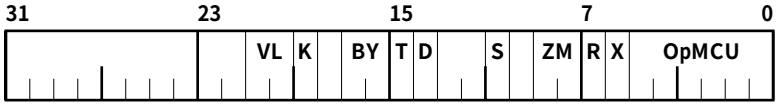


表３－11　Policyでのオプション設定（1/2）

オプション名	ビット位置	ビット名	意　味
VL	21	VideoOutLastOnly	描画タイミングの設定 １：プログレッシブで途中状態を表示せず最後のみ表示 ０：途中段階も表示
	20	LuminanceOutOnly	描画タイミングの設定 １：輝度成分の更新されたスキャンのみ表示 ０：すべてのスキャンで表示
K	19	BitStuffCheck	スタッフィング・ビットのチェック １：行う ０：行わない
BY	17	ByteStuffDisable	スタッフィング・バイト １：許可しない ０：許可する
	16	ByteStuffEnable	スタッフィング・バイト（ByteStuffDisable = 0のとき有効） １：0x10000まで許容 ０：セグメント間4バイトまで許容
T	15	2passEnable	伸長処理のパス回数設定 １：２パスで伸長する ０：１パスで伸長する
D	14	DNLEnable	DNLマーカ １：許容する ０：許容しない
S	11	UsePset	JPEGEXpset関数 １：JPEGEXpset関数をユーザが作成する ０：使用しない
ZM	9	VideoZoomLinear	画像の拡大／縮小 ZM = 01：拡大／縮小する 11：　〃 10：線形フィルタにより拡大／縮小する 00：拡大／縮小しない（等倍で伸長）
	8	VideoZoomNormal	

保守／廃止

表３－11　Policyでのオプション設定（2/2）

オプション名	ビット位置	ビット名	意　味
R	7	PutMCURGB	画像出力の設定 １：RGBで出力する ０：YCbCrで出力する
X	6	UseExPutMCU	JPEGEXputMCU関数を １：使用する ０：使用しない
OpMCU	5	UsePutMCUOnly	ユーザ作成のputMCU以外を １：伸長しない ０：伸長する
	4	UsePutMCU	ユーザ作成のputMCU関数を １：使用する ０：使用しない
	3	UsePutMCU22	ユーザ作成のputMCU22関数を １：使用する ０：使用しない
	2	UsePutMCU41	ユーザ作成のputMCU41関数を １：使用する ０：使用しない
	1	UsePutMCU21	ユーザ作成のputMCU21関数を １：使用する ０：使用しない
	0	UsePutMCU11	ユーザ作成のputMCU11関数を １：使用する ０：使用しない

備考　UsePutMCUオプションを使わず（基本ライブラリでユーザが作成したputMCUxxxライブラリを使用せず），追加ライブラリのJPEGEXputMCU関数を直接書き直して，追加ライブラリをカスタマイズすることもできます。その方法については，**３.６　追加ライブラリのカスタマイズ**を参照してください。

保守／廃止

（ａ）VideoOutLastOnly / LuminanceOutOnly（VLオプション）

これらは，描画タイミングを設定するオプションです。

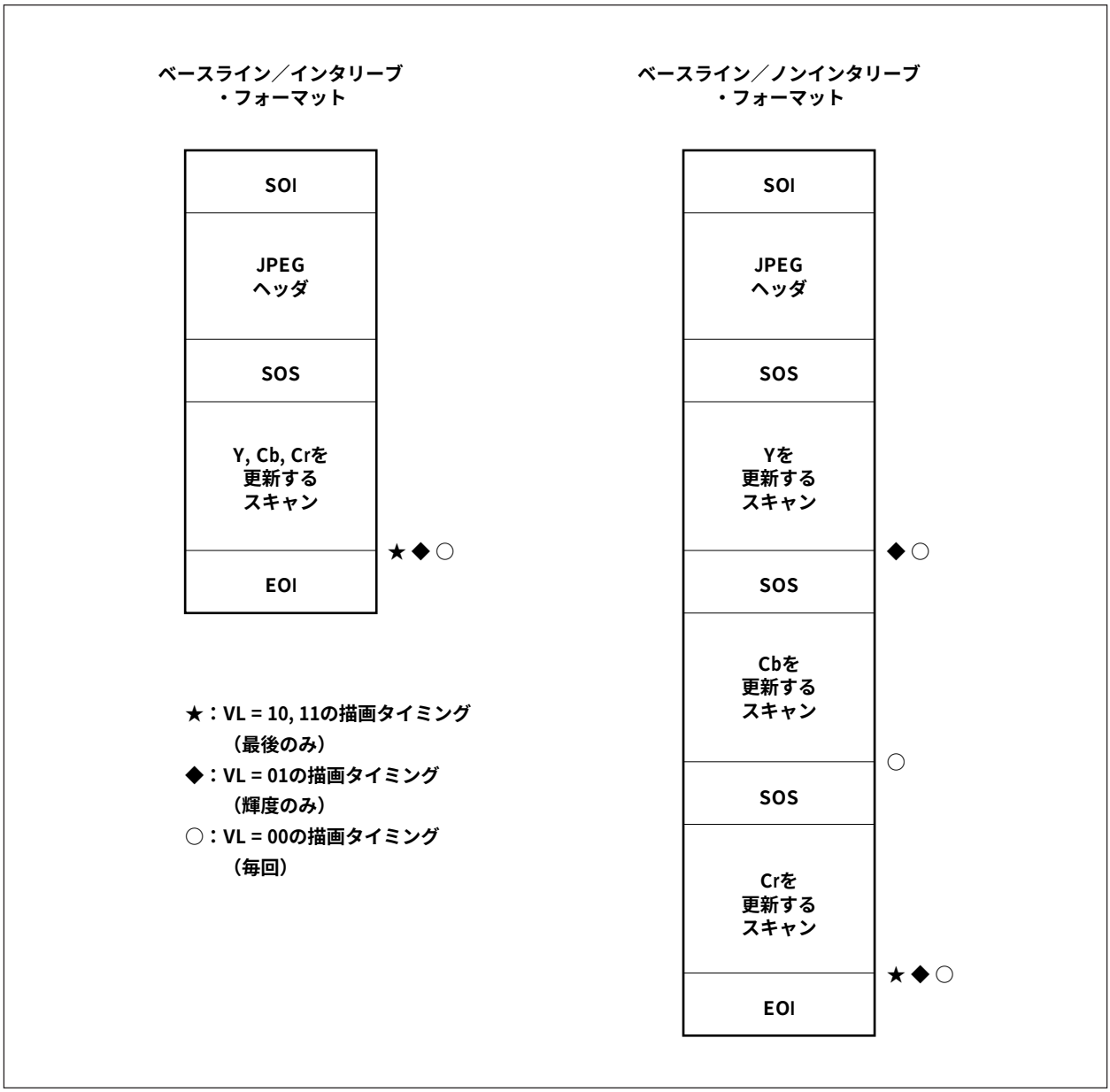
VideoOutLastOnlyオプションを指定すると，伸長の途中段階での描画はせず，伸長処理の最後の時点で描画します。

LuminanceOutOnlyオプションは，VideoOutLastOnly = 0のときに有効となるオプションです（VideoOutLastOnly = 1のとき，LuminanceOutOnlyは参照されません）。

LuminanceOutOnly = 0のとき，輝度成分の更新されたスキャンごとに描画します。

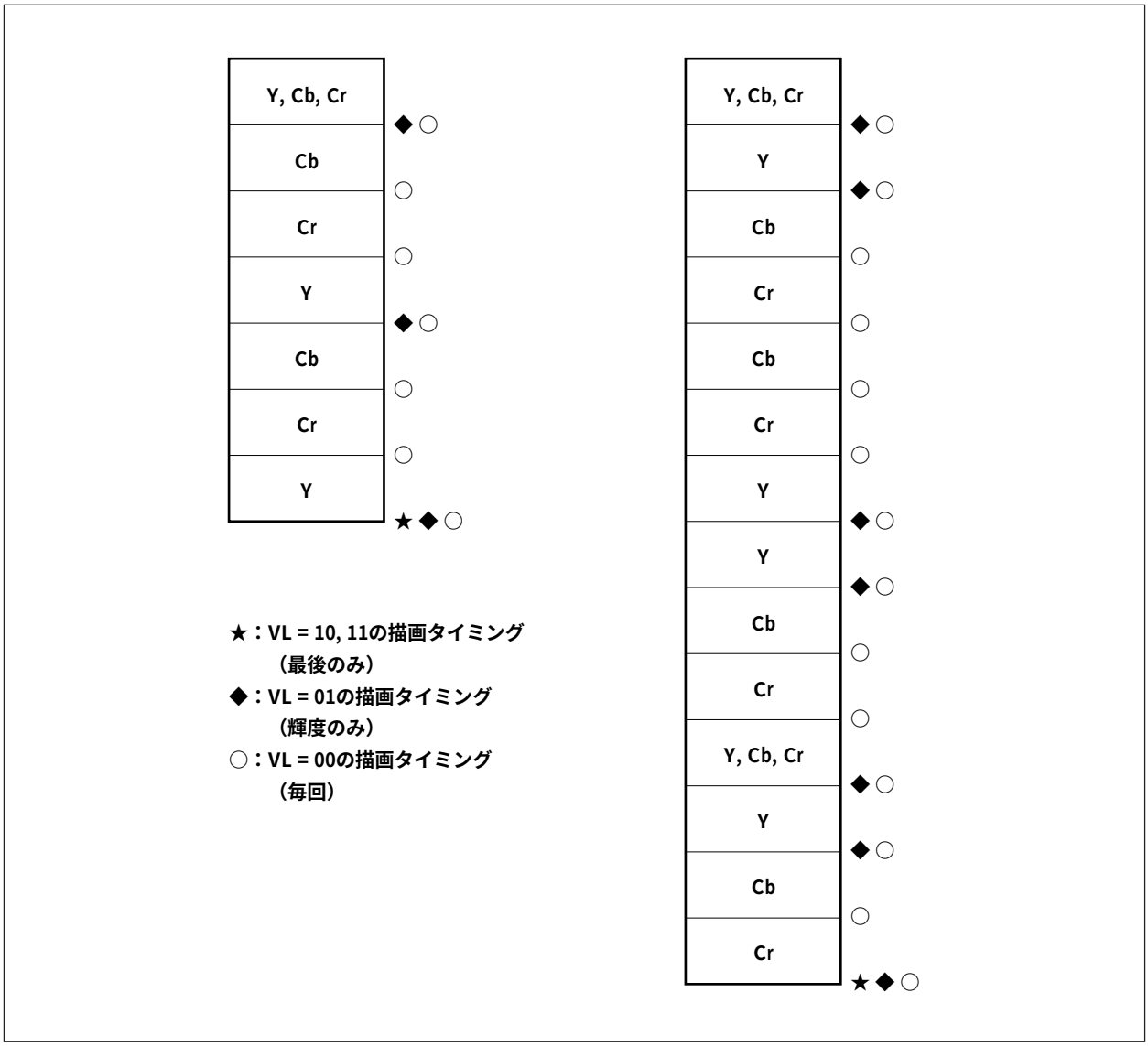
LuminanceOutOnly = 1のとき，すべてのスキャンごとに描画します。

図３－７　ベースライン・フォーマットの描画タイミング



保守／廃止

図３－８　プログレッシブ・フォーマット描画タイミング



保守／廃止

（ｂ）BitStuffCheck（Kオプション）

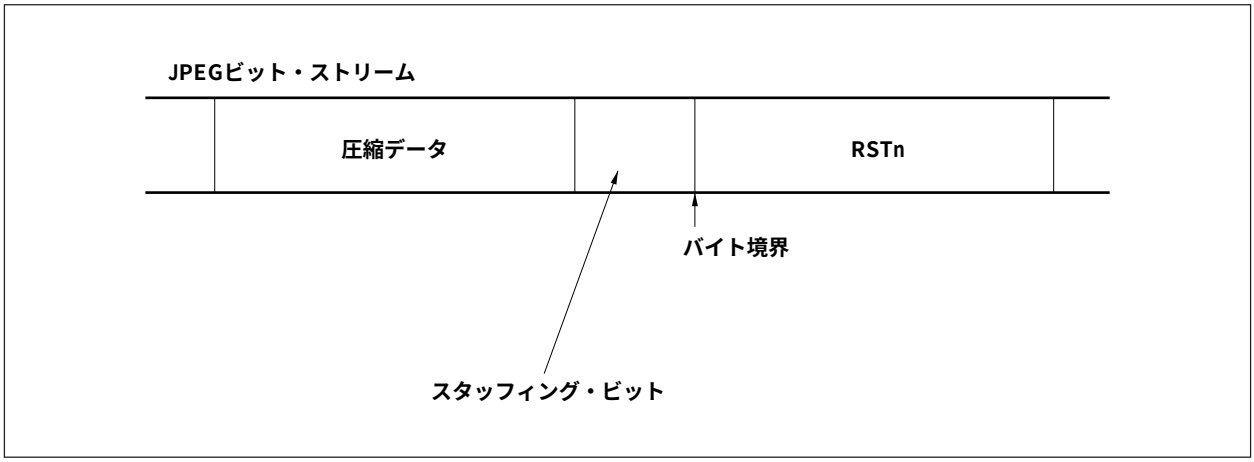
これはスタッフィング・ビットのチェックを行うかどうかを設定するオプションです。

JPEGファイルの圧縮データはビット単位で扱われ，SOFやDHTセグメント，RSTnマーカなどの各マーカはバイト単位で扱われます。そのため，圧縮データからマーカに切り替わる部分（EOI, SOS, RSTn, DNLなど）で１ビット-７ビットの隙間が生じる場合があります。この隙間をスタッフィング・ビットといい，ISO/IEC 10918-1ではこのスタッフィング・ビットの値を‘ １ ’とするように定めています。

追加ライブラリではBitStuffCheck = 1のとき，JPEGファイル内のスタッフィング・ビットが‘ １ ’であるか，‘ ０ ’かをチェックします。チェックした結果，‘ ０ ’のビットが見つかるとワーニング処理を行います。

なお，JPEGファイルのスタッフィング・ビットの値が‘ １ ’でも‘ ０ ’でも通常は問題ありません。

図３－９　スタッフィング・ビット



保守／廃止

（ｃ）ByteStuffDisable/ByteStuffEnable（BYオプション）

JPEGファイル内のセグメントとセグメントの間の隙間をスタッフィング・バイトといいます。たとえば，SOIセグメントとそれに続くAPP0セグメントの間に１バイトの0x00がある場合，このスタッフィング・バイトはJPEGファイルとしては意味のないものです。しかし，ISO/IEC 10918-1ではJPEGファイル内のスタッフィング・バイト存在の可否について，特に規定がありません。

このオプションで，スタッフィング・バイトの存在を認めるかどうかをユーザが設定します。

表３－12　ByteStuffDisable/ByteStuffEnable（スタッフィング・バイト）オプション

設定値	意　味
BY = 10 BY = 11	スタッフィング・バイトの存在を否定します。 この場合，スタッフィング・バイトを含むようなJPEGファイルを伸長すると，ほとんどの場合「マーカ・エラー」でエラー終了します。
BY = 01	スタッフィング・バイトを0x10000まで許容します。 しかし，スタッフィング・バイトの中にJPEGのマーカと間違えるようなバイト列があった場合には誤動作の原因となりますので，注意が必要です。
BY = 00	ディフォールト。セグメント間4バイトまでのスタッフィング・バイトを許容します。

（ｄ）2passEnable（Ｔオプション）

このオプションでは，追加伸長処理のパス回数の設定を行います。

2passEnable = 1のとき，伸長処理は２パスで行われ，2passEnable = 0のとき，伸長処理は１パスで行われます。ただし，DNLEnable = 1（**3.4.3（3）（e）DNLEnable（Dオプション）**参照）の場合には，無条件に2passEnable = 1と見なされます。

伸長処理の１パスと２パスの違いを表３－13に示します。

表３－13　パス回数による伸長処理の違い

項　目	１パス	２パス
ワーク・エリア・サイズ	豊富に必要	少なくてよい
例）４：１：１（640×480 pixel）の場合	約1 Mバイト	約5 Kバイト
例）１：１：１（640×480 pixel）の場合	約2 Mバイト	
実行時間	高速	低速
機能制限	なし	・JPEGバッファの更新不可 （JPEGバッファ内のデータ伸長後，処理中止） ・ハフマン・テーブルの多重定義不可 （多重定義以降の伸長処理中止）

備考　ワーク・エリア・サイズの例は，目安です。

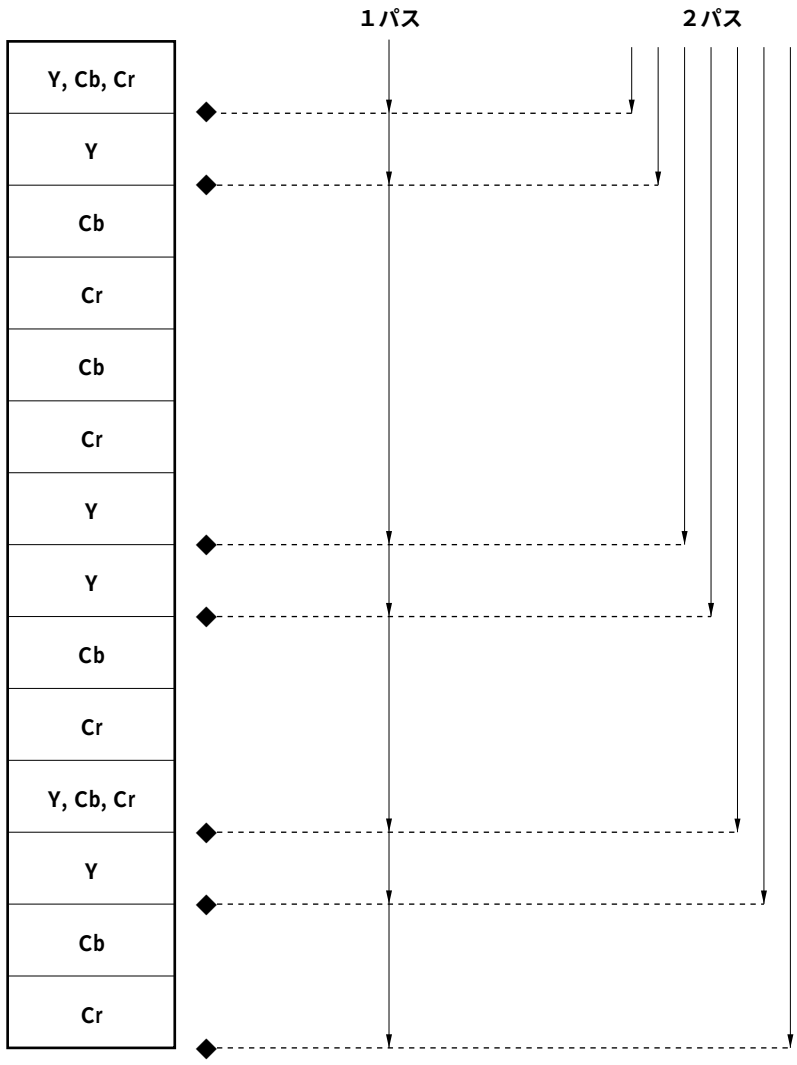
ワーク・エリア・サイズはJPEGEXWORK構造体で設定するものです。２パスでは伸長する画像のサイズによらずワーク・エリア・サイズは約5 Kバイトです。１パスでは伸長したDCT係数をすべて保存しておくため，非常に多くのワーク・エリアが必要となります。

１パスの設定がなされても十分なワーク・エリアが得られなかった場合，追加ライブラリは自動的に２パス設定に切り替わります。

２パスの場合には，描画を行うたびにJPEGファイルの先頭から圧縮データをなぞるので，処理速度が非常に遅くなります。また，２パスでは描画を行いながら圧縮データのデコードを行うので，画像の表示速度も遅くなります。

保守／廃止

図３－10　追加伸長処理のパス回数と描画タイミング

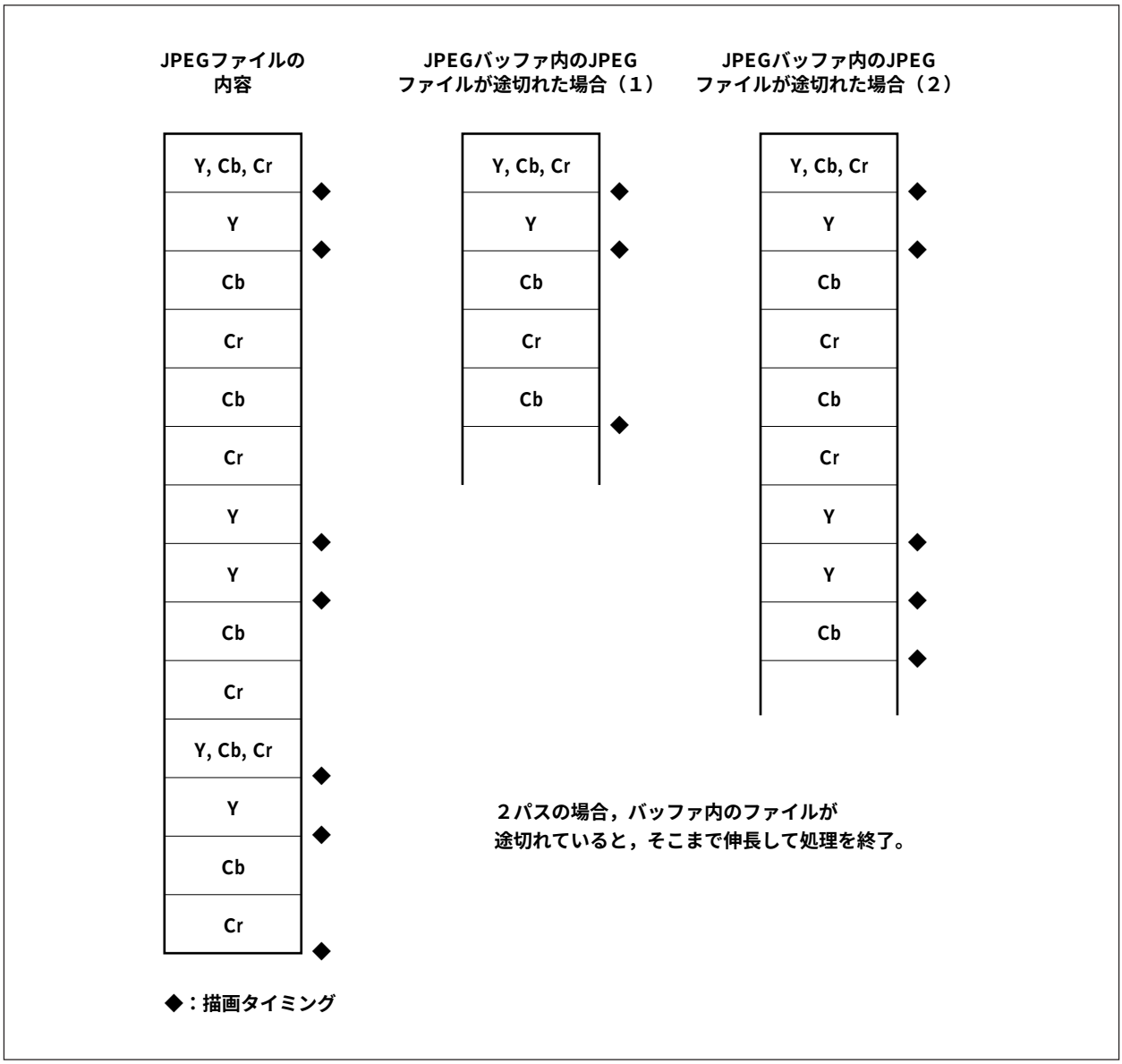


◆：描画タイミング

保守／廃止

また，JPEGバッファ内に，指定されたJPEGファイルが入りきらずに途切れた場合，２パスではJPEGバッファの内容を入れ替えて伸長処理を継続できません。この場合には，最初に指定されたJPEGバッファ内のJPEGファイルを伸長した時点で処理を終了します。

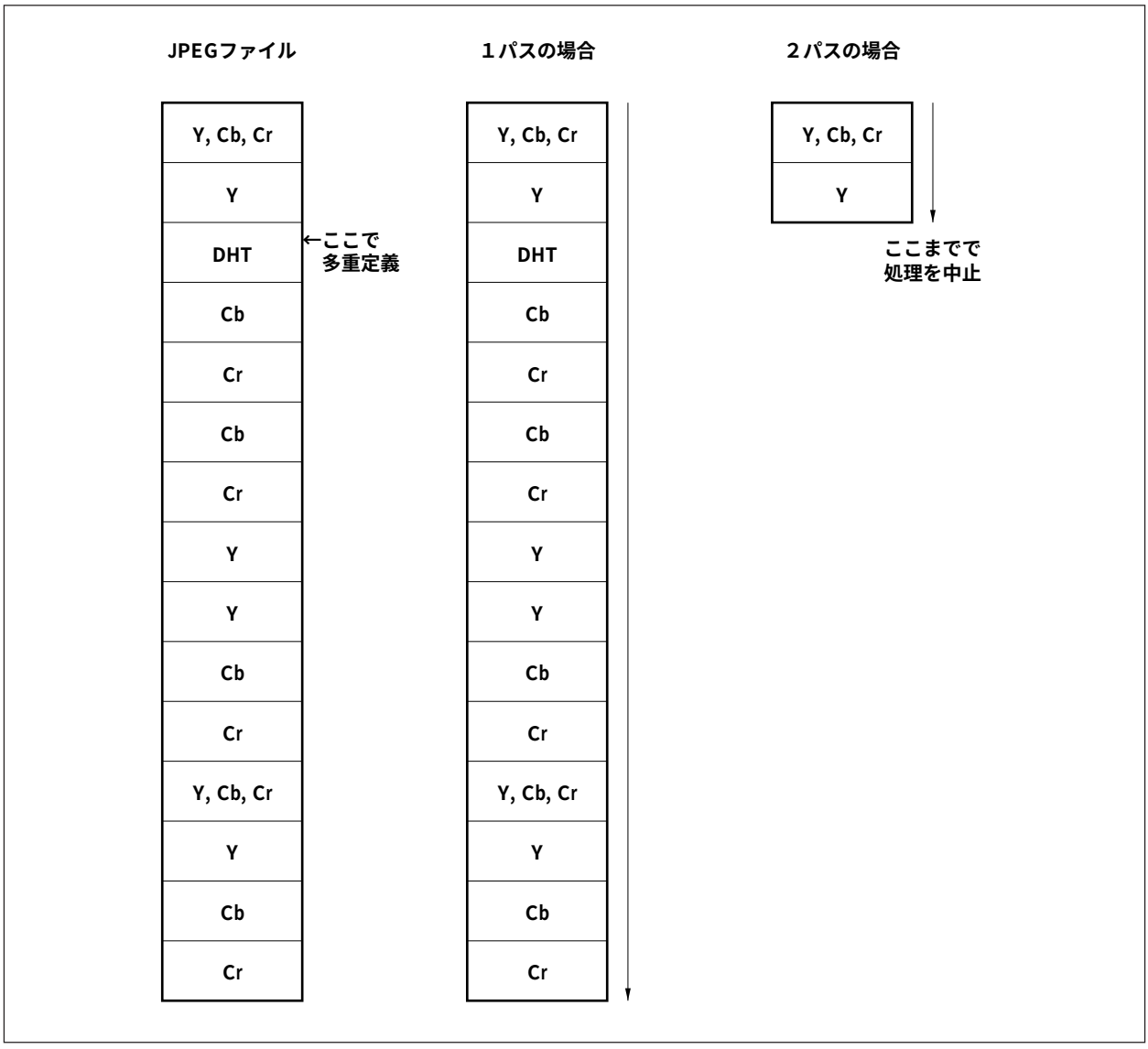
図３－11　JPEGバッファ内のJPEGファイルが途切れた場合の伸長処理（２パス）



保守／廃止

同じID番号のハフマン・テーブルが多重定義された場合，１パスの設定では，ワーク・エリアのサイズが許せばそのまま伸長を続行しますが，２パスの設定では多重定義されたところまで伸長を行い，それ以降の伸長を中止します。

図３－12　ハフマン・テーブル多重定義時のパス回数による伸長処理の違い



保守／廃止

（ｅ）DNLEnable（Dオプション）

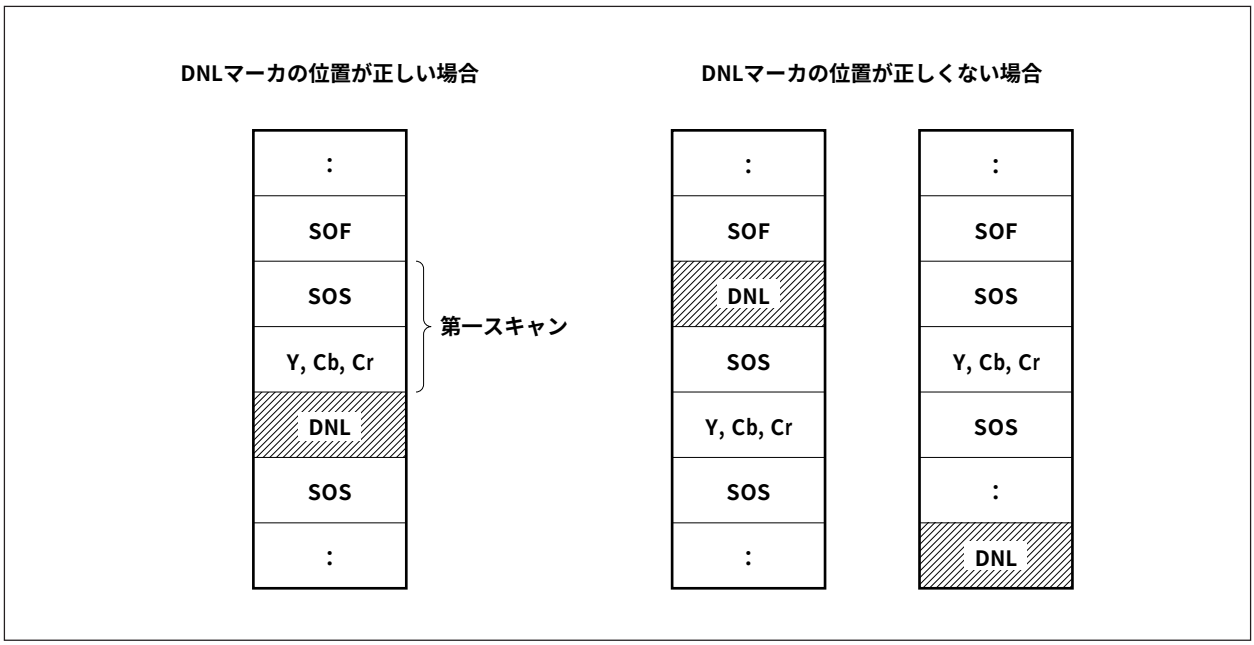
これは，DNLマーカ（ライン数の再定義）を含むJPEGファイルの伸長の可否を設定するオプションです。DNLEnable = 1のとき，DNLマーカを含むJPEGファイルを伸長できます。DNLEnable = 0のとき，DNLマーカを含むJPEGファイルの伸長処理はできません。

DNLEnable = 1にすると強制的に2パスでの伸長処理となります（2passEnableオプションの設定は無視されます）。

DNLセグメントはSOFマーカ内で指定されたY（画像の縦ピクセル数）の値を訂正するものです。DNLマーカの位置は第一スキャン直後と規定されており（ISO/IEC 10918-1準拠），これに反する場合，追加ライブラリではエラーとなります。

なお，通常のJPEGファイルではDNLマーカが用いられることはほとんどありません。

図３－13　JPEGファイル内のDNLマーカの位置



（ｆ）UsePset（Sオプション）

JPEGEXpset関数をユーザがカスタマイズする（**3.6　追加ライブラリのカスタマイズ**参照）場合に設定するオプションです。

JPEGEXpset関数をユーザがカスタマイズする場合には，UsePset = 1に設定します。

UsePset = 0のとき，追加ライブラリはpset関数を，インライン展開させたモジュールに動的に切り替える場合があります（設定されたすべてのオプションと伸長するJPEGファイルの種類に応じた関数に動的に切り替えます）。

ただし，ユーザ・カスタマイズした基本ライブラリのｐｕｔＭＣＵ関数を使用するように設定（UsePutMCU = 1（**3.4.3（3）（ｊ）UsePutMCU（OpMCUオプション）**参照））し，基本ライブラリが呼ばれると，UsePset = 1の場合でもJPEGEXpset関数は呼ばれません。

保守／廃止

（ｇ）VideoZoomLinear / VideoZoomNormal（ZMオプション）

これらは，画像の拡大伸長を設定するオプションです。この機能はAP705100-B03追加ライブラリのputMCU関数で実現されている機能です。基本ライブラリのputMCU関数やユーザ・カスタマイズされたputMCU関数を使用した場合の動作は保証しません。

VideoZoomLinearは，VideoZoomNormal = 0の場合にのみ有効となります。

表３－14　VideoZoomLinear/VideoZoomNormal（拡大伸長）オプション

設定値	意　　味
ZM = 01 ZM = 11 （VideoZoomNormal）	拡大します。 倍率はJPEGEXINFO構造体のメンバratio÷8が適用されます。
ZM = 10 （VideoZoomLinear）	線形一次補間という方式で拡大します。 倍率はJPEGEXINFO構造体のメンバratio÷8が適用されます。 線形一次補間方式での拡大は，VideoZoomNormalの方式よりも処理時間がかかります。 線形一次補間は一種のフィルタです。MCU単位で処理を行っているため，MCU内部は滑らかになりますが，MCU境界がかえって目立ってしまう場合があります。
ZM = 00	拡大を行わずに等倍で伸長します。 JPEGEXINFO構造体のメンバratioの値は無視されます。

（ｈ）PutMCURGB（Rオプション）

画像の出力方式を設定するオプションです。PutMCURGB = 1のとき，YCbCrではなくRGBで画像出力を行います。PutMCURGB = 0のとき，YCbCrで画像出力を行います。

この機能はAP705100-B03追加ライブラリのputMCU関数で実現されている機能です。基本ライブラリのputMCU関数や，ユーザ・カスタマイズされたputMCU関数を使用した場合には機能しません（影響がありません）。

（ｉ）UseExPutMCU（Xオプション）

JPEGEXputMCU関数をユーザがカスタマイズする場合に設定するオプションです。

JPEGEXputMCU関数をユーザがカスタマイズする場合には，UseExPutMCU = 1に設定します（JPEGEXputMCU関数のカスタマイズについては，**３.６　追加ライブラリのカスタマイズ**を参照してください）。

UseExPutMCU = 0の場合，追加ライブラリ内部でJPEGEXputMCU関数とその代替関数を動的に切り替えます。UseExPutMCU = 1の場合，追加ライブラリは動的な切り替えを行わず，常にJPEGEXputMCU関数を呼び出します。

保守／廃止

（ｊ）UsePutMCU（OpMCUオプション）

基本ライブラリのカスタマイズ機能（**2.6　基本ライブラリのカスタマイズ**参照）を用いてユーザが作成したputMCU関数を，追加ライブラリでも使用可能に設定するオプションです。

ユーザ作成のputMCU関数を追加ライブラリで使用する場合には，UsePutMCUOnlyまたはUsePutMCUのどちらかのビットをセットし，さらに，ユーザ作成の関数に対応するオプション・ビットをセットします。

たとえば，putMCU22に対応したJPEGファイルを伸長する場合には，UsePutMCUOnly = 1またはUsePutMCU = 1，さらにUsePutMCU22 = 1に設定することで，ユーザ作成のputMCU関数が追加ライブラリにより呼び出されます。

表３－15　UsePutMCUオプション

設定値	意　味
UsePutMCUOnly = 1	ユーザ作成のputMCU関数以外は伸長しない
UsePutMCU = 1	ユーザ作成のputMCU関数を使用する
UsePutMCU22 = 1	ユーザ作成のputMCU22関数を使用する
UsePutMCU41 = 1	ユーザ作成のputMCU41関数を使用する
UsePutMCU21 = 1	ユーザ作成のputMCU21関数を使用する
UsePutMCU11 = 1	ユーザ作成のputMCU11関数を使用する

UsePutMCU = 1設定時，伸長するJPEGファイルに対応するオプション（UsePutMCU22-UsePutMCU11）がセットされていなかった場合には，AP705100-B03追加ライブラリのputMCU関数が呼び出されます。しかし，UsePutMCUOnly = 1設定時に，伸長するJPEGファイルに対応するオプション（UsePutMCU22-UsePutMCU11）がセットされていなかった場合には，AP705100-B03追加ライブラリは伸長処理を中止し，エラー終了します。この意味でUsePutMCUよりUsePutMCUOnlyオプションの方が優先順位が高くなります。

表３－16に，OpMCUオプションの設定値ごとのライブラリ動作例を示します。

表３－16　OpMCUオプション設定値とライブラリ動作例

OpMCUオプション設定値（例）	putMCU22関数対応のJPEGファイル入力時	putMCU41関数対応のJPEGファイル入力時
001000	追加ライブラリのputMCU関数呼び出し	追加ライブラリのputMCU関数呼び出し
011000	ユーザ定義putMCU22関数呼び出し	追加ライブラリのputMCU関数呼び出し
101000	ユーザ定義putMCU22関数呼び出し	伸長処理を中止
111000	ユーザ定義putMCU22関数呼び出し	伸長処理を中止

保守／廃止

（４）ratio

JPEGEXINFO構造体のメンバPolicyで画像拡大／縮小オプション（VideoZoomLinear/VideoZoomNormal）を有効にした場合，その拡大／縮小率をこのメンバratioで指定します。

ratioには，実際の倍率に８を掛けて整数に丸めた値を代入してください。負あるいはゼロの値が指定された場合には，値‘１’が指定されたものとみなします。

（５）ErrorState

伸長処理中にエラーが発生した場合，メンバErrorStateにエラー番号が書き込まれます。エラー番号の内容については**３.４.６　追加伸長時のエラー内容**を参照してください。

（６）Work

JPEGEXWORK構造体の先頭アドレスをメンバWorkに設定します。JPEGEXWORK構造体は追加ライブラリが使用可能なワーク・エリアを設定する構造体です（**３.４.４　JPEGEXWORK構造体のパラメータ設定**参照）。

（７）Video

JPEGEXVIDEO構造体の先頭アドレスをメンバVideoに設定します。JPEGEXVIDEO構造体は描画関連の設定を行う構造体です（**３.４.５　JPEGEXVIDEO構造体のパラメータ設定**参照）。

（８）Inf

ユーザはこのメンバInfを意識する必要はありません。メンバInfには，追加ライブラリ自身がJPEGEXFrmINFO構造体の先頭アドレスを設定します。JPEGEXFrmINFO構造体は追加ライブラリが伸長処理を行うために必要な変数を格納するための構造体で，ワーク・エリア上に確保されます。

保守／廃止

３.４.４　JPEGEXWORK構造体のパラメータ設定

伸長メイン関数をコールする前に，JPEGEXWORK構造体で追加ライブラリが使用可能なワーク・エリアを指定します。

表３－17　JPEGEXWORK構造体

メンバ	型	内　　容	IN/OUT
Work1	unsigned int	ワーク・エリア先頭アドレス	IN
Work1Len	unsigned int	ワーク・エリア・サイズ（バイト数）	IN
Work1Used	unsigned int	使用ワーク・エリア・サイズ（バイト数）	OUT
Work2	unsigned int	ワーク・エリア先頭アドレス	IN
Work2Len	unsigned int	ワーク・エリア・サイズ（バイト数）	IN
Work2Used	unsigned int	使用ワーク・エリア・サイズ（バイト数）	OUT

（１）ワーク・エリアの指定

追加ライブラリが使用可能なワーク・エリアの先頭アドレスをWork1/Work2のどちらかに，使用可能なサイズ（バイト数）をWork1Len/Work2Lenに指定してください。追加ライブラリ終了後に，実際に使用したバイト数がWork1Used/Work2Usedに格納されます。

また，内蔵データRAMをワーク・エリアとして使用可能な場合には，Work1xxx/Work2xxxの残っている方に，その先頭アドレス，使用可能サイズを指定してください。

（２）内蔵RAMのワーク・エリア

内蔵RAMをワーク・エリアとして使用可能な場合，追加ライブラリはMCUバッファとDCTテンポラリ・バッファを内蔵RAMに確保しようとします。MCUバッファとDCTテンポラリ・バッファのサイズを表３－18に示します。

表３－18　MCUバッファとDCTテンポラリ・バッファのサイズ

バッファ	サイズ
MCUバッファ	768バイト（４：１：１のJPEGファイルの場合） 384バイト（１：１：１のJPEGファイルの場合）
DCTテンポラリ・バッファ	256バイト

保守／廃止

３.４.５　JPEGEXVIDEO構造体のパラメータ設定

伸長メイン関数をコールする前に，JPEGEXVIDEO構造体で追加伸長処理の画像出力に必要なパラメータ（VRAM構成，クリッピング）の設定を行います。

表３－19　JPEGEXVIDEO構造体

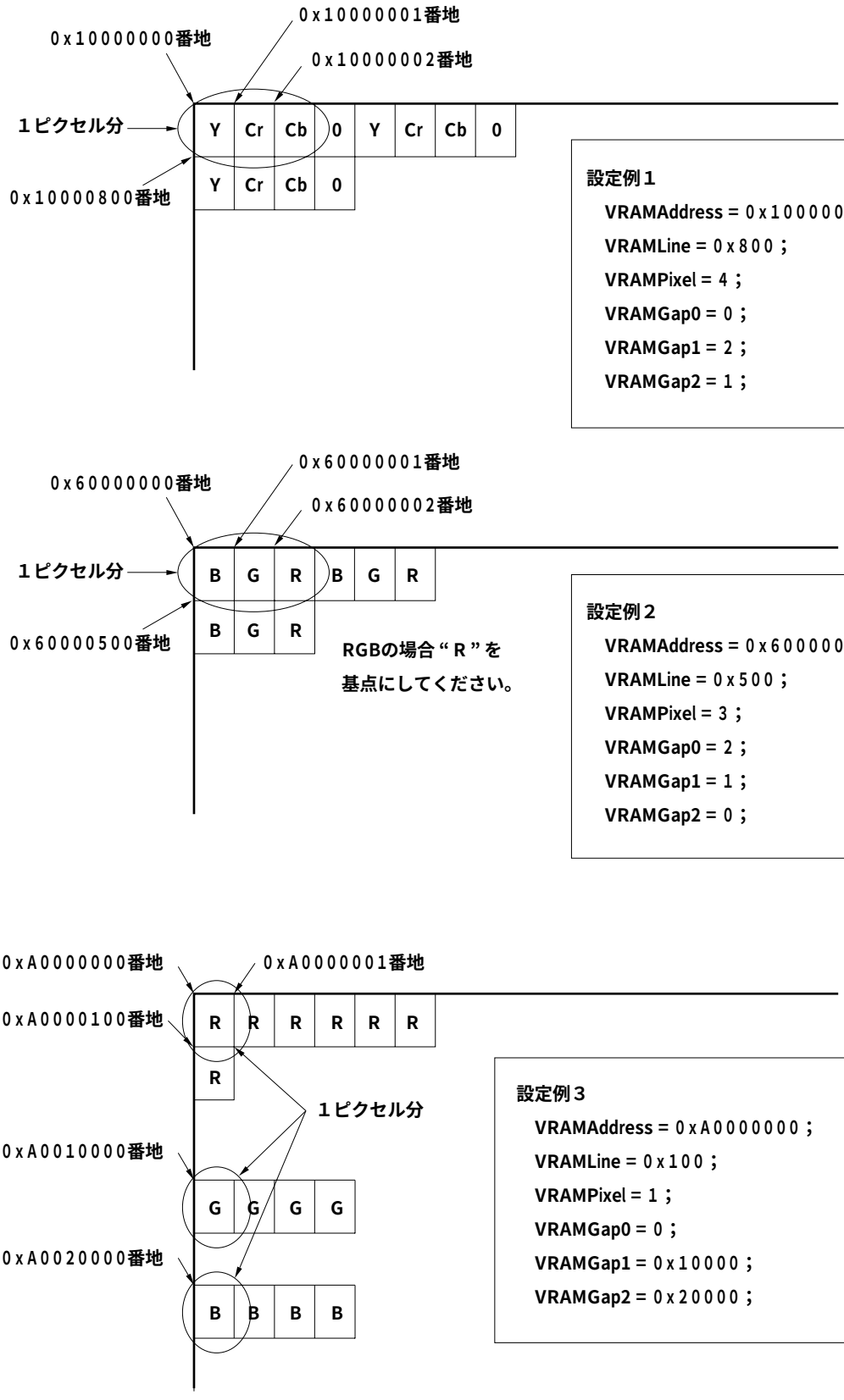
メンバ	型	内　　容	IN/OUT
VRAMAddress	unsigned char*	VRAMの先頭アドレス	IN
VRAMWidth	int	VRAMの横幅（ピクセル数）	IN
VRAMHeight	int	VRAMの縦幅（ピクセル数）	IN
VRAMPixel	int	VRAMの横1ピクセル分のアドレス差	IN
VRAMLine	int	VRAMの縦1ピクセル分のアドレス差	IN
VRAMGap0	int	Yピクセル（またはRピクセル）のバイト・オフセット	IN
VRAMGap1	int	Cbピクセル（またはGピクセル）のバイト・オフセット	IN
VRAMGap2	int	Crピクセル（またはBピクセル）のバイト・オフセット	IN
ClipStartX	int	クリッピング開始位置（Ｘ座標） クリッピングを行わない場合は，ダミー値0を設定	IN
ClipStartY	int	クリッピング開始位置（Ｙ座標） クリッピングを行わない場合は，ダミー値0を設定	IN
ClipWidth	int	クリッピング横サイズ（ピクセル） クリッピングを行わない場合は，ダミー値0x7FFFFFFFを設定	IN
ClipHeight	int	クリッピング縦サイズ（ピクセル） クリッピングを行わない場合は，ダミー値0x7FFFFFFFを設定	IN

（１）VRAM構成

VRAM関連メンバ（VRAMxxx）の設定例を図３－14に示します。

保守／廃止

図３－14　追加ライブラリのVRAM関連メンバ設定例



保守／廃止

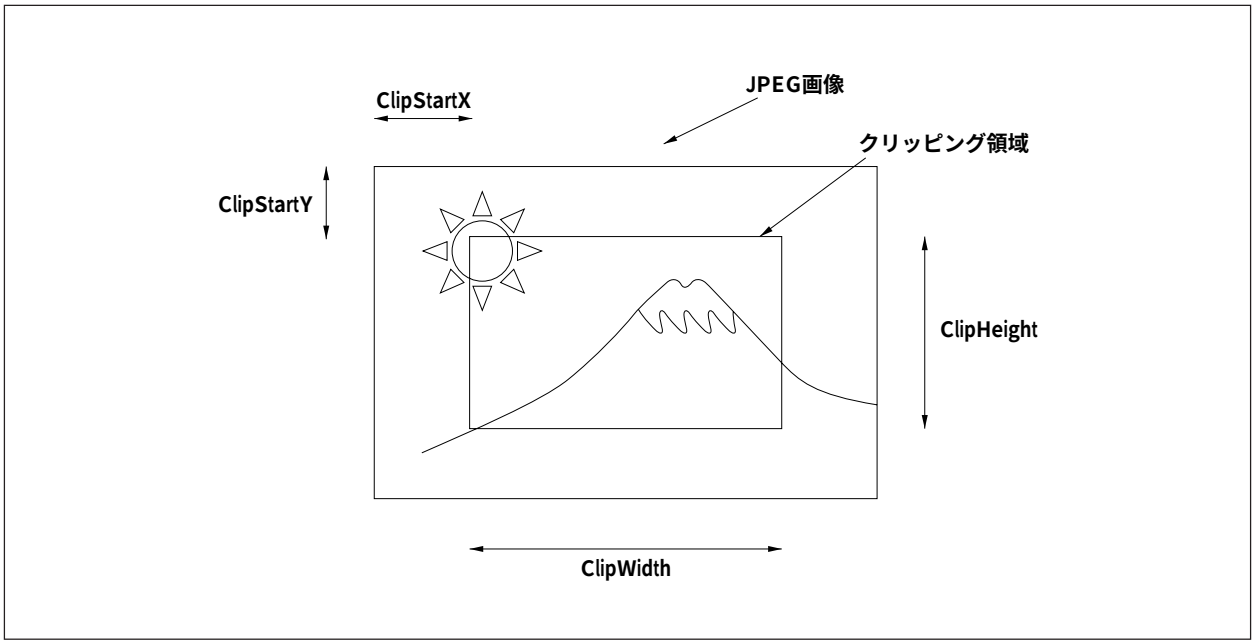
（2）クリッピングの設定

追加伸長時のクリッピングは，クリッピング関連メンバ（Clipxxx）にダミー値以外の値を設定した場合に行われます。クリッピングを行わない場合はクリッピング関連のメンバに表３－19に示すダミー値を代入してください。

クリッピング関連メンバ（Clipxxx）で指定されるJPEG画像の領域を図３－15に示します。

クリッピングされた画像を描画する位置を変更する場合は，VRAMAddressメンバの値を調節してください。

図３－15　クリッピング領域



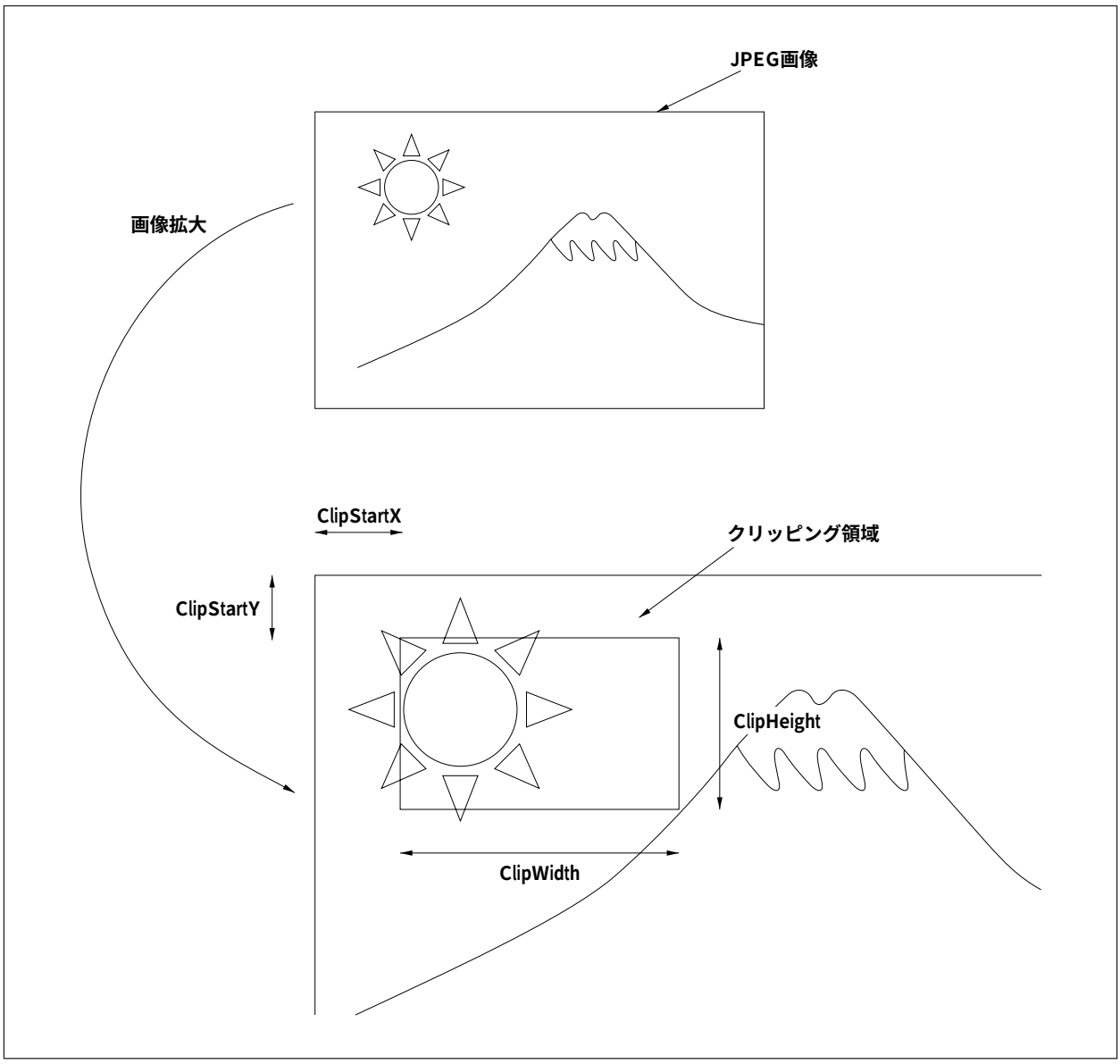
保守／廃止

（３）クリッピング設定と拡大／縮小設定との関係

JPEGEXINFO構造体メンバPolicyのオプション（３.４.３（３）（ｇ）VideoZoomLinear/VideoZoomNormal（ZMオプション）参照）で拡大／縮小を設定した場合には，拡大／縮小された画像に対してClipStartX，ClipStartY，ClipWidth，ClipHeightの値が適用されます。

図３－16に拡大／縮小伸長時のクリッピング領域を示します。

図３－16　拡大／縮小伸長時のクリッピング領域



保守／廃止

３.４.６　追加伸長時のエラー内容

追加ライブラリがエラー終了した場合，JPEGEXINFO構造体のメンバErrorStateに格納される番号とその内容を表３－20に示します。

表３－20　追加ライブラリのエラー内容

エラー・メッセージ	番号	エラー内容
ErrorMode	0x1000	JPEGEXINFO構造体のメンバModeに設定された値に誤りがあります。
ErrorAllocate	0x1001	ワーク・エリアのサイズが不足し処理が行えません。
ErrorMultiFrame	0x1002	マルチフレーム・フォーマットには対応していません。処理を終了します。
ErrorMultiScan	0x1003	マルチスキャンのため，伸長できません。
ErrorMultiDQT	0x1004	JPEGファイル中に同じ番号の量子化テーブルが多重定義されています。
ErrorMultiDHT	0x1005	JPEGファイル中に同じ番号のハフマン・テーブルが多重定義されています。
ErrorJPEGBuffLen	0x1006	JPEGバッファのサイズが小さすぎます。
ErrorJPEGMarker	0x1007	JPEGマーカ解析中にエラーが見つかりました。 未知のマーカが現れました。
ErrorSOIMarker	0x1008	SOI以外のマーカから始まっています。
ErrorDQTsegment	0x1009	DQTセグメントに誤りがあります。
ErrorDQTsegmentTq	0x100A	DQTセグメントに記述された量子化テーブル番号がJPEG規格外です。
ErrorDQTsegmentPq	0x100B	DQTセグメントに記述された量子化テーブル精度の値がJPEG規格外です。
ErrorSOFsegment	0x100C	SOFセグメントに誤りがあります。
ErrorSOFsegmentNf	0x100D	SOFセグメントに指定された色コンポーネントの数が多すぎます。
ErrorSOFsegmentSF	0x100E	SOFセグメントに記述されたサンプリング・ファクタの値がJPEG規格外です。
ErrorSOFsegmentTq	0x100F	SOFセグメントに記述された量子化テーブル番号がJPEG規格外です。
ErrorDHTsegment	0x1010	DHTセグメントに誤りがあります。
ErrorDHTsegmentTc	0x1011	DHTセグメントに記述されたテーブル番号（Tc）がJPEG規格外です。
ErrorDHTsegmentTh	0x1012	DHTセグメントに記述されたテーブル番号（Th）がJPEG規格外です。
ErrorSOSsegment	0x1014	SOSセグメントに誤りがあります。
ErrorSOSsegmentCi	0x1015	SOSセグメントに記述された色コンポーネントID番号がSOFセグメントに記述されたIDの中に見つかりません。
ErrorSOSsegmentTq	0x1016	そのスキャンを伸長するための量子化テーブルが定義されていません。
ErrorSOSsegmentTa	0x1017	そのスキャンを伸長するためのAC成分用ハフマン・テーブルが定義されていません。
ErrorSOSsegmentTd	0x1018	そのスキャンを伸長するためのDC成分用ハフマン・テーブルが定義されていません。
ErrorDCcode	0x1019	DC係数デコード中に圧縮データにエラーが見つかりました。
ErrorACcode	0x101A	AC係数デコード中に圧縮データにエラーが見つかりました。
ErrorHuffcode	0x101B	予想されない位置にマーカが現れました。 プログレッシブ・デコード中に圧縮データにエラーが見つかりました。
ErrorDNLsegment	0x101C	予想されない位置にDNLマーカが現れました。
ErrorRSTsegment	0x101D	予想されない位置にRSTnマーカが現れました。
ErrorDRIsegment	0x101E	DRIセグメントに誤りがあります。
ErrorDNLnot1stScan	0x101F	DNLセグメントの位置が最初のスキャンの直後にありません。
ErrorPutMCUfunc	0x1020	PolicyでUsePutMCUOnlyが選択されていますが，伸長しようとするJPEGファイルのサンプル比に対応するputMCU関数がありません。

保守／廃止

３.４.７　追加伸長時のワーニング内容

追加ライブラリ実行中にワーニングが発生した場合，JPEGEXWarning関数がコールされ，その第一引き数としてワーニング番号が渡されます。ワーニング番号とその内容を表３－21に示します。

表３－21　追加ライブラリのワーニング内容

ワーニング・メッセージ	番号	ワーニング内容
WarningBitStuff	0x2000	スタッフィング・ビット・チェックを行った結果，エラーが見つかりました。無視して処理を続行します。
WarningProgACInterleave	0x2001	プログレッシブ・フォーマットでAC係数はノンインタリーブでなければなりません（JPEG規格外）。 問題がないので処理を続行します。
WarningBlock20	0x2002	1MCU内のブロックの総数が20個を越えています（JPEG拡張規格外）。 問題がないので処理を続行します。
WarningBlock10	0x2003	1MCU内のブロックの総数が10個を越えています（JPEG規格外）。 問題がないので処理を続行します。
WarningDQTbaselinePq	0x2004	ベースラインで16ビット精度量子化テーブルが定義されています（JPEG規格外）。 問題がないので処理を続行します。
WarningDHTbaselineTh	0x2005	ベースラインでハフマン・テーブル番号２，３が定義されています（JPEG規格外）。 問題がないので処理を続行します。
WarningAPP14	0x2006	APPセグメント（APP14）がありましたが，APP14セグメント内の色空間識別子の値が０，１，２以外でした。 次のようにみなして処理を続行します。 単色の場合：モノクロ ３色の場合：YCbCr ４色の場合：最後の色を無視してYCbCr
WarningMultiDQT	0x2007	１つ以上のスキャン後に量子化テーブルが定義されています。問題がないので処理を続行します。
WarningMultiDHT	0x2008	１つ以上のスキャン後にハフマン・テーブルが定義されています。問題がないので処理を続行します。
WarningSOFYDNL	0x2009	DNLセグメントの位置が最初のスキャンの直後ではありませんが，このDNLで定義される値を画像の縦ピクセル数とみなして処理を行います。

保守／廃止

３.５　オーバライト関数

追加ライブラリが用意する関数のうち，ユーザが上書き可能なものをオーバライト関数と呼びます。オーバライト関数のうち，JPEGファイル取得関数は必ず上書きして定義しなければなりません。そのほかのオーバライト関数はオプションですので，上書きしてもしなくてもかまいません。

３.５.１　JPEGファイル取得関数

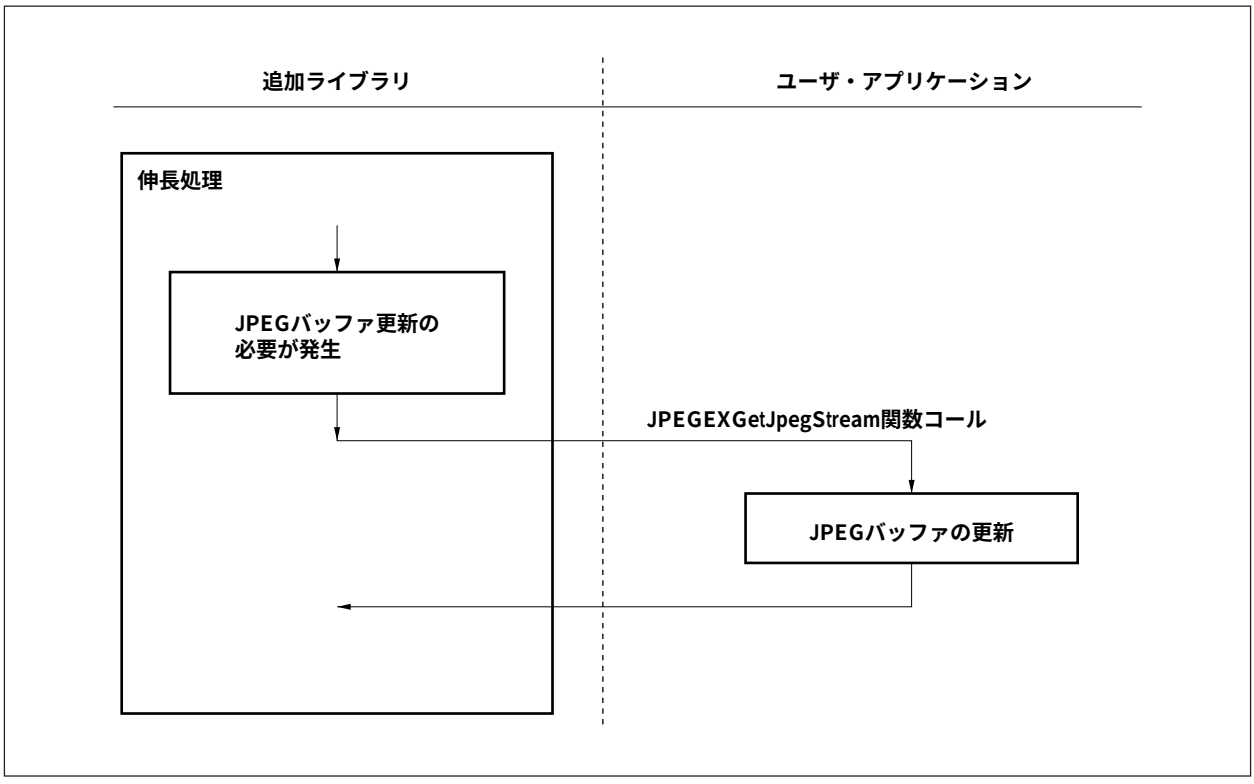
関数名　JPEGEXGetJpegStream
形　式　void　JPEGEXGetJpegStream（struct JPEGEXBUFF＊ JPBUFF）；
引き数　JPEGEXBUFF構造体の先頭アドレス
返り値　なし

JPEGバッファの内容を更新する場合，JPEGEXBUFF構造体にメンバ設定をしてから，JPEGファイル取得関数をコールします。この関数は，ユーザ・アプリケーションで必ず定義してください。

追加ライブラリからJPEGファイル更新要求があった場合，ユーザ・アプリケーションでこの関数をコールして，JPEGバッファの内容を更新します。

追加伸長処理が1パス・モードの場合には，追加ライブラリからJPEGファイル更新要求があるたびにこの関数がコールされます。追加伸長処理が2パス・モードの場合，この関数は一度しかコールされません（パス設定の詳細については，**３.４.３（３）（ｄ）2passEnable（Tオプション）**を参照してください）。

図３－17　JPEGバッファの更新処理



JPEGEXBUFF構造体のメンバ設定は次のように行います（JPEGEXBUFF構造体はヘッダ・ファイルjpegex.hファイルに定義されています）。

表３－22　JPEGEXBUFF構造体

メンバ	型	内　　容	IN/OUT
TaskID	int	タスクID番号	OUT（上書き可）
JPEGBUFF	unsigned char*	JPEGバッファの先頭アドレス	IN
JPEGBUFFLEN	unsigned int	JPEGバッファのサイズ（バイト数）	IN

（１）TaskID

TaskIDは，ワーク・エリア内にJPEGEXBUFF構造体を確保する際に，JPEGEXINFO構造体メンバTaskID（**３.４.３（１）TaskID**参照）の値で初期化されます。

このTaskIDの値は，マルチタスク環境下で動作する場合に，そのタスクのIDとなります。シングルタスクで使用する場合には，このTaskIDを無視してかまいません。また，JPEGEXGetJpegStream関数内で，このメンバTaskIDの値を上書きしてもかまいません。追加ライブラリは，初期化後にはこのメンバの値を参照することはありません。

（２）JPEGBUFF

JPEGBUFFには，JPEGバッファの先頭アドレスを設定します。

（３）JPEGBUFFLEN

JPEGBUFFLENには，JPEGバッファのサイズ（バイト数）を設定します。

JPEGバッファのサイズはできるだけ大きな値，できれば伸長するJPEGファイルがすべて入るサイズを確保してください。JPEGバッファのサイズが32バイト以下の場合，追加ライブラリは動作しません。

保守／廃止

３.５.２　APPマーカ関数

関数名　JPEGEXdecodeAPP

形　式　void　JPEGEXdecodeAPP（int APPnumber, int JpegStreamOffset, int Length）；

引き数　第１引き数：APPnマーカの番号

０：APP0（0xFF 0xE0）

１：APP1（0xFF 0xE1）

２：APP2（0xFF 0xE2）

：

15：APP15（0xFF 0xEF）

16：COM（0xFF 0xFE）

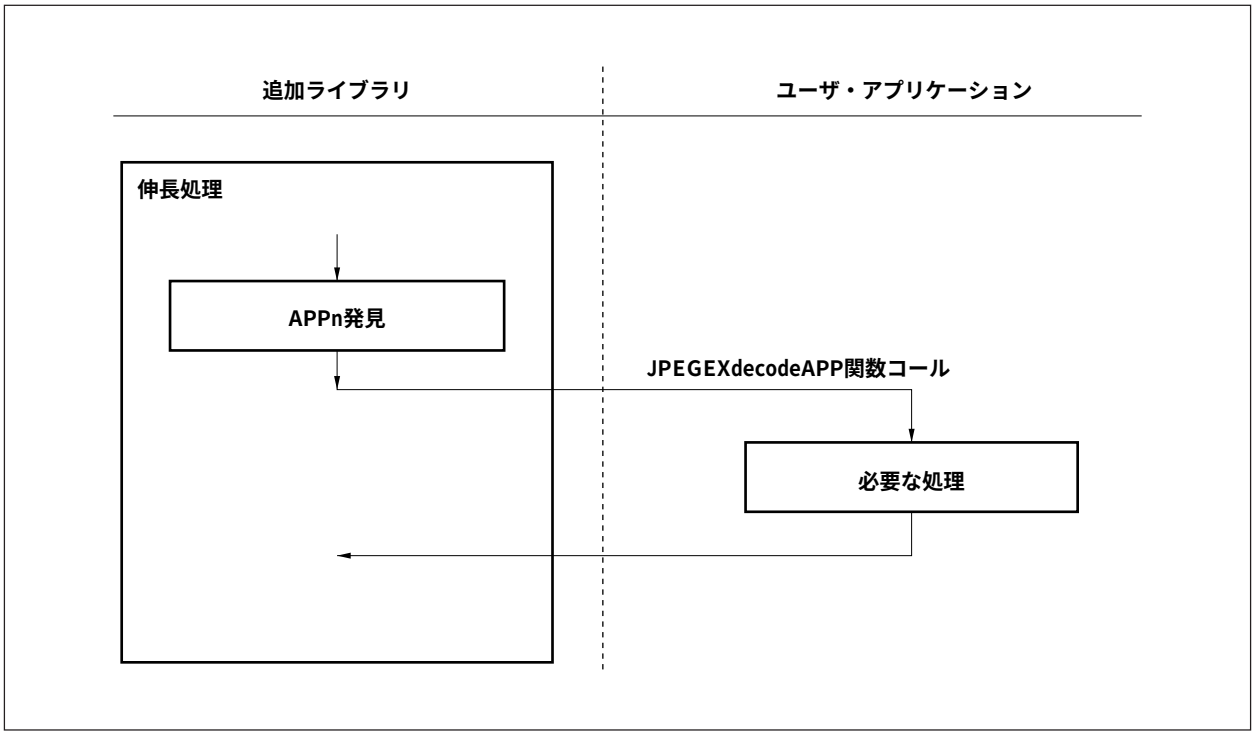
第２引き数：APPnセグメントの（JPEGファイル先頭からの）オフセット・バイト数

第３引き数：APPnセグメントの長さ（バイト数）

返り値　なし

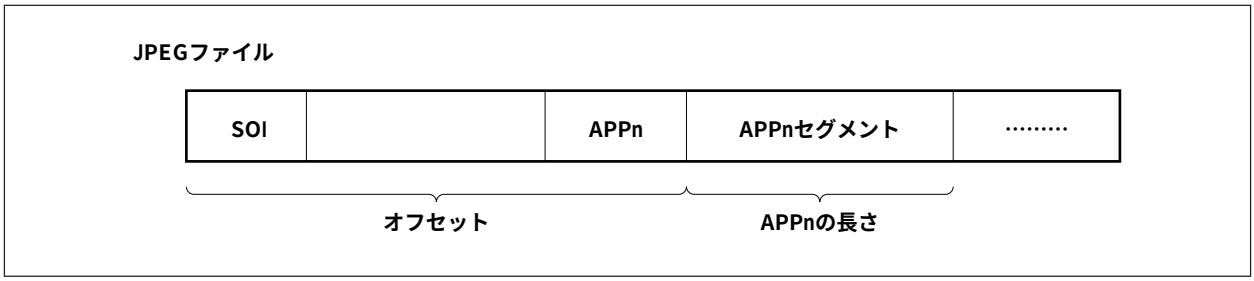
追加ライブラリはAPPnセグメントが見つかった場合にこの関数をコールします。この関数はオプションです。必要な処理がある場合には，この関数をユーザが上書きできます。ユーザがこの関数を上書きしない場合は，ディフォールトのJPEGEXdecodeAPP関数が呼ばれます。ディフォールトの関数は何もしません。

図３－18　APPマーカ発見時の処理



保守／廃止

図３－19　APPnセグメントのオフセットと長さ



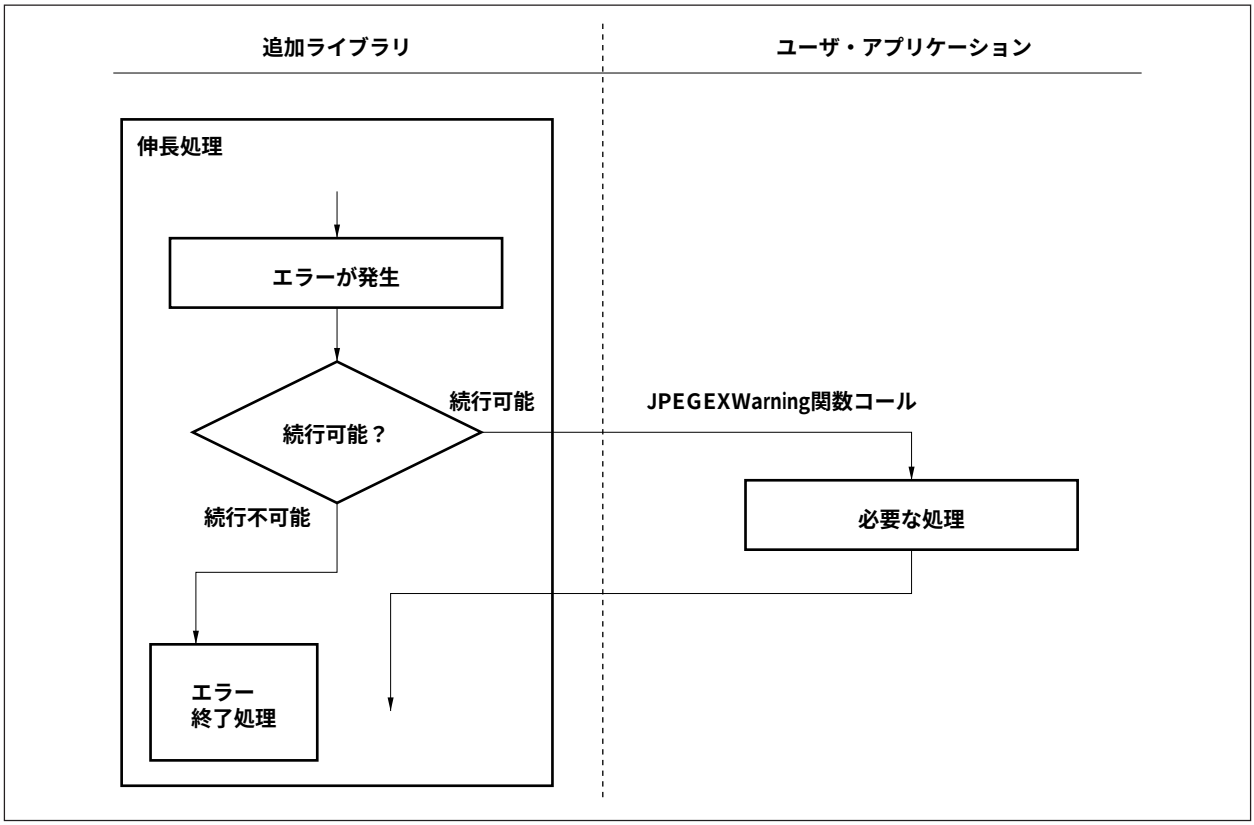
3.5.3　ワーニング・メッセージ関数

関数名　JPEGEXWarning
形　式　void　JPEGEXWarning（int WarningNumber）；
引き数　ワーニング・メッセージ番号
返り値　なし

追加伸長処理中に，処理全体を終了するほどではないエラーが発生した場合，追加ライブラリはこの関数をコールします。この関数はオプションです。必要な処理がある場合には，この関数をユーザが上書きできます。ユーザがこの関数を上書きしない場合は，ディフォールトのJPEGEXWarning関数が呼ばれます。ディフォールトの関数は何もしません。

引き数となるワーニング・メッセージ番号の内容については**3.4.7　追加伸長時のワーニング内容**を参照してください。

図３－20　続行可能なエラー発生時の処理



保守／廃止

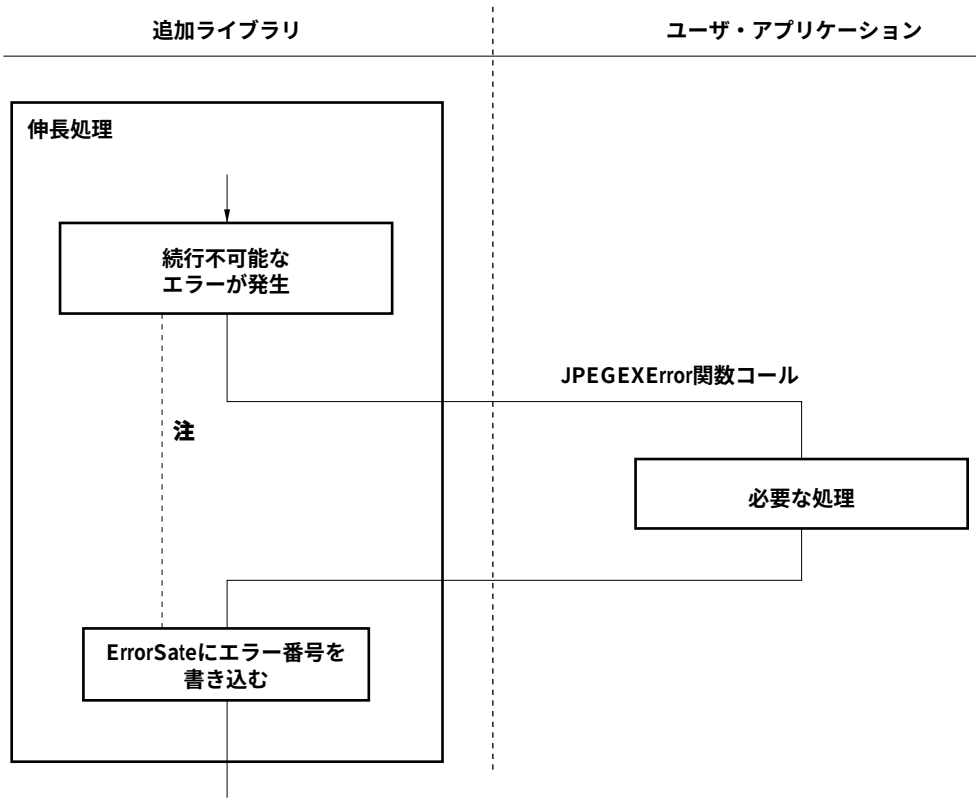
３.５.４　ディバグ・ライブラリのエラー・メッセージ関数

関数名　JPEGEXError
形　式　void　JPEGEXError（char* msg）;
引き数　エラー・メッセージ文字列へのポインタ
返り値　なし

この関数はオプションです。ディバグ・ライブラリ（**３.２　追加ライブラリのリンク**参照）を使用した場合のみコールされます。追加伸長処理中に処理続行不可能なエラーが発生した場合，伸長処理をエラー終了する直前に追加ライブラリはこの関数をコールします。通常，エラーが発生した場合には，JPEGEXINFO構造体のメンバErrorStateにエラー番号のみが書き込まれます。一方この関数がコールされると，エラー番号に対応したASCIIコード文字列でエラー内容を知らせるため，エラー内容が分かりやすくなります。さらに必要な処理がある場合には，この関数をユーザが上書きすることができます。

引き数となるエラー・メッセージ文字列については**３.４.６　追加伸長時のエラー内容**を参照してください。

図３－21　エラー発生時のディバグ・ライブラリの処理



注　ディバグ・ライブラリを使用しないときのルートです。JPEGEXError関数はコールされません。

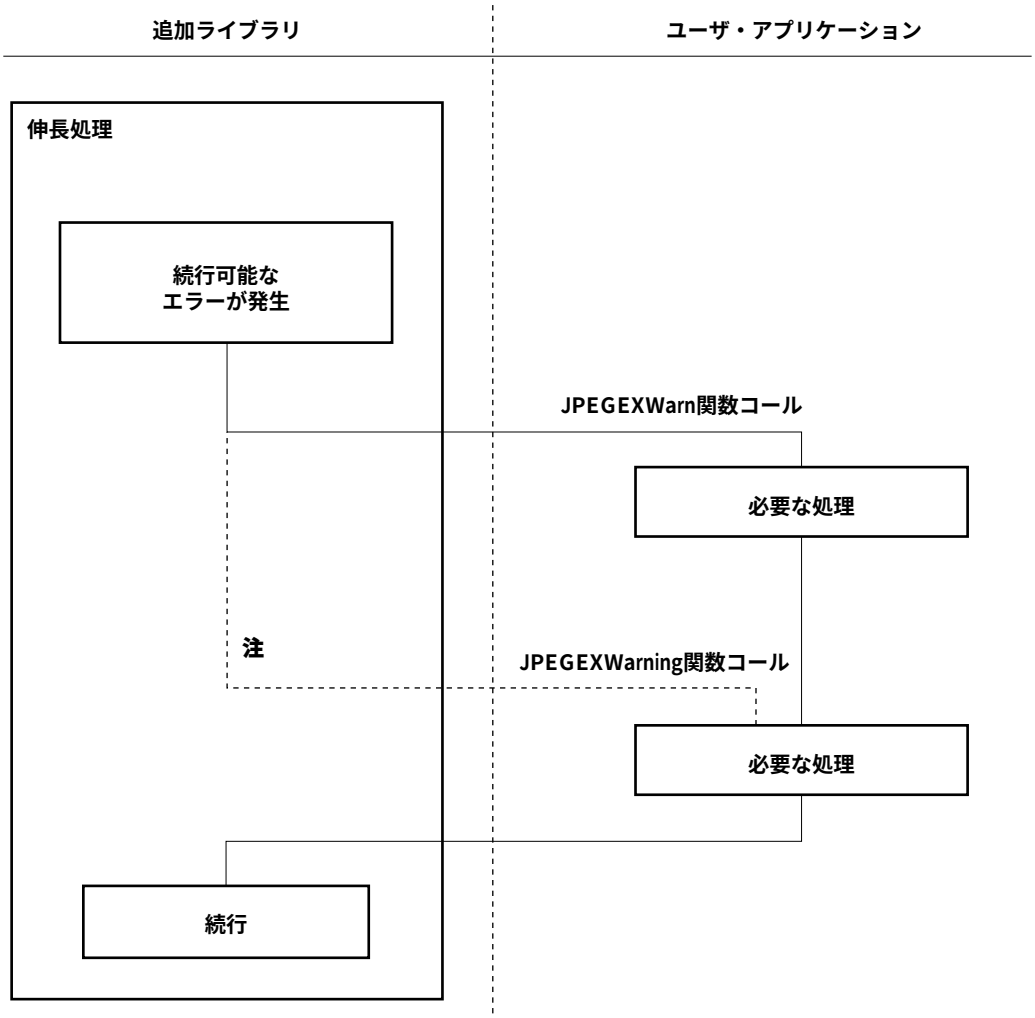
保守／廃止

3.5.5　ディバグ・ライブラリのワーニング・メッセージ関数

関数名　JPEGEXWarn
形　式　void　JPEGEXWarn（char* msg）;
引き数　ワーニング・メッセージ文字列へのポインタ
返り値　なし

この関数はオプションです。ディバグ・ライブラリ（**3.2　追加ライブラリのリンク**参照）を使用した場合のみコールされます。追加伸長処理中に処理続行が可能なエラー（ワーニング）が発生した場合，追加ライブラリはJPEGEXWarning関数をコールする直前に，この関数をコールします。通常，ワーニングが発生した場合JPEGEXWarning関数の引き数となるのはワーニング・メッセージ番号のみです。一方この関数がコールされると，ワーニング番号に対応したASCIIコード文字列でワーニング内容を知らせるため，その内容が分かりやすくなります。さらに必要な処理がある場合には，この関数をユーザが上書きできます。ワーニング・メッセージについては**3.4.6　追加伸長時のワーニング内容**を参照してください。

図３－22　ワーニング発生時のディバグ・ライブラリの処理



注　ディバグ・ライブラリを使用しないときのルートです。JPEGEXWarn関数はコールされません。

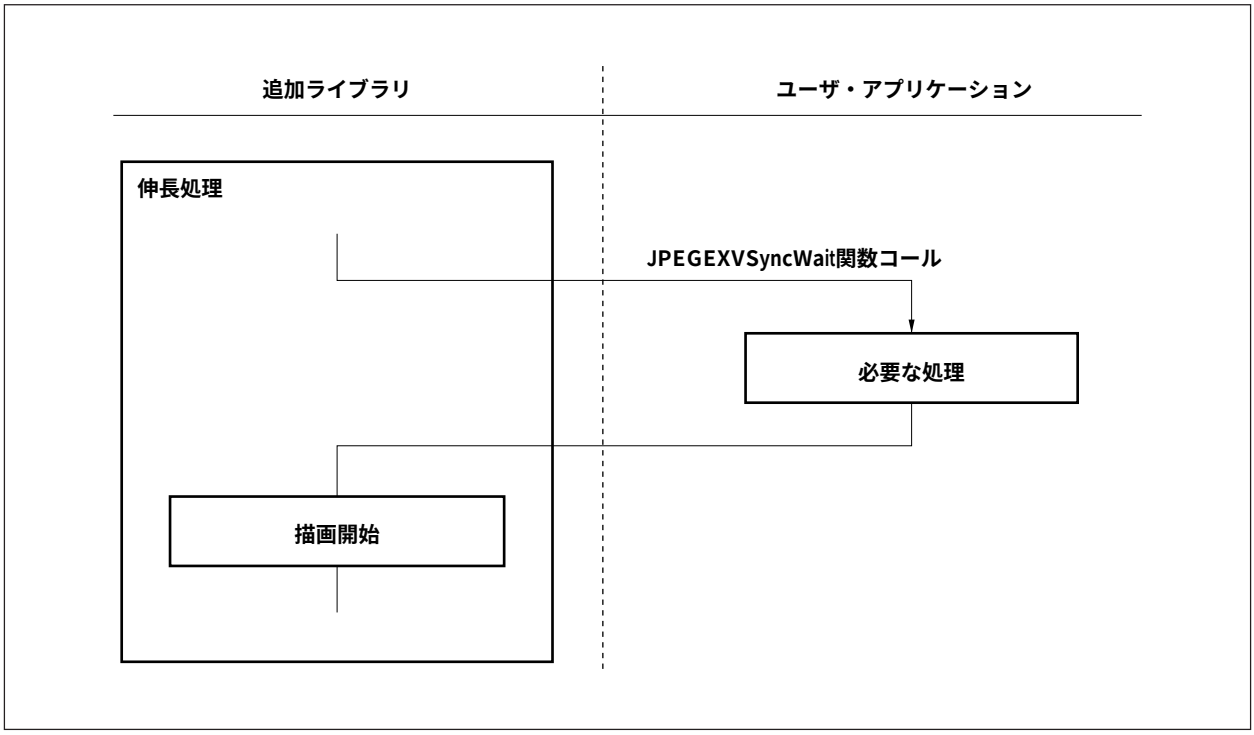
保守／廃止

3.5.6　表示タイミング調整関数

関数名　JPEGEXVSyncWait
形　式　void　JPEGEXVSyncWait（）；
引き数　なし
返り値　なし

追加ライブラリが描画を開始する前に，この関数がコールされます。この関数はオプションです。表示タイミングを調整するなどの必要な処理がある場合には，この関数をユーザが上書きできます。ユーザがこの関数を上書きしない場合は，ディフォールトのJPEGEXVSyncWait関数が呼ばれます。ディフォールトの関数は何もしません。

図３－23　描画開始前の処理



保守／廃止

３.５.７　MCUデータ出力関数

関数名　JPEGEXputMCU

形　式　void　JPEGEXputMCU（short* mcubuff, int Y, int X, struct JPEGEXMCUSTR* MCUstr）;

引き数　第１引き数：MCUバッファへのポインタ

第２引き数：描画するMCUの左上隅座標（Y）

第３引き数：描画するMCUの左上隅座標（X）

第４引き数：JPEGEXMCUSTR構造体の先頭アドレス

返り値　なし

この関数はVRAMに画像を描画する関数です。伸長処理で最もハードウエアに左右されるこの部分をユーザが上書きして作成できるようにしています。この関数はオプションです。

JPEGEXputMCU関数を作成して使用する場合には，JPEGEXINFO構造体のメンバPolicyのUseExPutMCUビットをセットする必要があります（**３.４.３（３）（ｉ）UseExPutMCU（Xオプション）**参照）。このビットがセットされていない場合，追加ライブラリはJPEGEXputMCU関数とその代替関数を動的に切り替えます。

JPEGEXputMCU関数を上書きする場合には，**３.６　追加ライブラリのカスタマイズ**を参照してください。

保守／廃止

３.５.８　ピクセル・データ出力関数

関数名　JPEGEXpset

形　式　この関数は，画像出力方式の設定により，引き数が異なります。

YCbCr出力の場合：

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

RGB出力の場合：

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

引き数　第１引き数：JPEGEXMCUSTR構造体の先頭アドレス

第２引き数：描画するピクセルのY座標

第３引き数：描画するピクセルのX座標

第４引き数：YCbCr出力の場合，Y色素データ

RGB出力の場合，R色素データ

第５引き数：YCbCr出力の場合，Cb色素データ

RGB出力の場合，G色素データ

第６引き数：YCbCr出力の場合，Cr色素データ

RGB出力の場合，B色素データ

返り値　なし

この関数はVRAMに画像を描画する関数です。伸長処理で最もハードウエアに左右されるこの部分をユーザが上書きして作成できるようにしています。この関数はオプションです。

JPEGEXpset関数を作成して使用する場合には，JPEGEXINFO構造体のメンバPolicyのUsePsetビットを１に設定する必要があります（**３.４.３（３）（ｆ）UsePset（Sオプション）**参照）。このビットが０の場合，追加ライブラリはJPEGEXpset関数とその代替関数（pset関数をインライン展開させた関数）を動的に切り替える場合があります。

また，PolicyのUsePutMCUビット（**３.４.３（３）（ｊ）UsePutMCU（OpMCUオプション）**参照）を１に設定すると，JPEGEXputMCU関数ではなく，基本ライブラリのputMCU関数がコールされる場合があります。このとき，JPEGEXpset関数はコールされません。

なお，変数Cy, Cu, Cvおよび変数，R, G, Bの値は0x00～0xFFです。

pset関数のCソース例を次に示します。

```
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;
}
```

保守／廃止

3.6　追加ライブラリのカスタマイズ

追加ライブラリの画像出力部分は，関連する関数を上書き（オーバライト）することによってカスタマイズが可能です。

使用する画像メモリの仕様や画像出力オプションの設定によっては，必ず画像出力部分をカスタマイズしなければなりません。その例として，次のような場合があります。

・画像メモリの色空間が24ビットRGB（24ビットYCbCr）でない場合。
・画像メモリがst.b/st.h/st.wでアクセスできない場合。

画像出力部分をカスタマイズするには，JPEGEXpset関数またはJPEGEXputMCU関数を変更します。JPEGEXpset関数は，画面上に点を描画する関数で，追加ライブラリ内部のputMCU関数から呼ばれます。また，JPEGEXputMCU関数は，MCU単位で画面出力する関数で，カスタマイズ専用の関数です（カスタマイズしない場合には，追加ライブラリ内部のputMCU関数が呼ばれます）。

3.6.1　簡易的なカスタマイズ

JPEGEXpset関数をオーバライトすることにより，容易に画像出力部分をカスタマイズできます。JPEGEXpset関数の仕様については，**3.5.8　ピクセル・データ出力関数**を参照してください。

JPEGEXpset関数をオーバライトして使用する場合には，JPEGEXINFO構造体のメンバPolicyでJPEGEXpset関数を使用するように，オプション設定する必要があります（**3.6.3　カスタマイズにおけるオプション設定**参照）。

3.6.2　高度なカスタマイズ

システム全体のパフォーマンスを上げるためには，JPEGEXpset関数に加え，JPEGEXputMCU関数をオーバライトし画像出力部分をカスタマイズします。JPEGEXputMCU関数をオーバライトして使用することにより，アドレス計算や引き数のスタックへのストア／ロードなどの冗長な部分を少なくできます。

JPEGEXputMCU関数の仕様については，**3.5.7　MCUデータ出力関数**を参照してください。

3.6.3　カスタマイズにおけるオプション設定

JPEGEXINFO構造体メンバPolicyには画像出力に関するオプションがあります。JPEGEXpset関数，JPEGEXputMCU関数をカスタマイズして使用する場合には，これらのオプションを表３－23に示す値に設定してください。

表３－23　カスタマイズにおけるPolicyオプション設定値

優先順位	オプション・ビット名	JPEGEXpset関数 使用時の設定値	JPEGEXpset関数／ JPEGEXputMCU関数 使用時の設定値
高	UsePutMCUOnly	0	0
↑	UsePutMCU	0	0
	VideoZoomNormal	任意	任意
	VideoZoomLinear	任意	任意
↓	UseExPutMCU	0	1
低	UsePset	1	任意

保守／廃止

なお，これらのオプションには優先順位があります。優先順位の高いオプションの値が1にセットされるとそのオプションで指定された画像出力関数が使用されることになり，それより優先順位の低いオプションで指定された画像出力関数は使用されない場合があります。

（1）UsePutMCUOnly

このオプションは基本ライブラリでユーザが作成したputMCU関数を使用するためのオプションです。このオプションの値が1にセットされると，UseExPutMCU／UsePsetオプションの値は無視され，JPEGEXputMCU関数，JPEGEXpset関数はコールされません。

（2）UsePutMCU

このオプションは基本ライブラリでユーザが作成したputMCU関数を使用するためのオプションです。次に示す３つの条件がすべて成立する場合，UseExPutMCUオプションは無視され，JPEGEXputMCU関数はコールされません。

- UsePutMCU = 1
- PolicyのUsePutMCU22, UsePutMCU41, UsePutMCU21, UsePutMCU11オプション（**3.4.3（3）（j）UsePutMCU（OpMCUオプション）**参照）のいずれかの値が1
- Policyでセットした**UsePutMCUxx**オプションに対応するサンプル比のJPEGファイル（**表２－48　基本伸長のカスタマイズ対象関数**参照）を伸長する

（3）VideoZoomNormal/VideoZoomLinear

これらのオプションの値が1にセットされると，追加ライブラリは必ずJPEGEXpset関数をコールします（したがって，JPEGEXpset関数をオーバライトする必要があります）。この場合，UseExPutMCUオプションの値は無視され，JPEGEXputMCU関数はコールされません。

3.6.4　カスタマイズ方法例

次ページにJPEGEXputMCU関数およびJPEGEXpset関数を定義したCソース例を示します。この例の中でJPEGEXpset関数は何度もコールされる部分です。追加ライブラリの処理速度を向上させるためには，このように頻繁にコールされる関数をインライン展開させるなどの工夫をする必要があります。

保守／廃止

```
#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);
                       }
                  }
        }
}
```

第４章　インストレーション

４.１　インストレーション手順

（１）UNIX版の場合

```
tar xvof　/dev/fd0
```

上記コマンドでリリース媒体からハード・ディスクに内容をコピーしてください（デバイスの指定はそれぞれの環境により異なります）。

（２）Windows版の場合

コピー・コマンドかファイラ（ファイル操作アプリケーション）でリリース媒体からハード・ディスクに内容をコピーしてください。

自己解凍形式になっていますので，解凍してください。

注意　ハード・ディスクにコピーされたディレクトリ構成は，特に指定はありません。

保守／廃止

４.２　サンプル作成手順

JPEGライブラリの指定を行うファイル“ archive ”を作成します。コマンド・ラインでjparc830を起動してください。詳しい作成手順は**２.２.１　リンク時のライブラリ選択**を参照してください。

ライブラリのパス名は，相対パスでも絶対パスでも指定できます（この例では相対パスで指定しています）。

../../lib732（UNIX-AP70732-B03）
../../lib830（UNIX-AP705100-B03）
..￥..￥lib732（Windows-AP70732-B03）
..￥..￥lib830（Windows-AP705100-B03）

コンパイルについては，次に示すように指定してください。

make　-f　makergb　（VRAMがRGB形式の場合）
make　-f　makeycc　（VRAMがYCbCr形式の場合）

ただし，RGBの場合はmain.cの中の“ ＃define　RGB ”の部分を有効にしてください。

コンパイラがWindows版のNEC製CA732/CA830コンパイラ・パッケージの場合は，VSH（ブイシェル）で，makeの代わりにvmakeとしてコンパイルしてください。

GHSコンパイラ使用時は，コンパイルするとアドレス情報がファイル“ jpeg.map ”に残されます。

NECコンパイラ使用時にアドレス情報が必要な場合には，次に示すダンプ・コマンドを実行してください。

dump732 -h jpeg.elf　＞　jpeg.map　　（AP70732-B03）
dump830 -h jpeg.elf　＞　jpeg.map　　（AP705100-B03）

４.３　サンプル動作手順

実機かシミュレータが必要です。

ダウンロード後，PC（プログラム・カウンタ）を　_ _start（グローバル・シンボル，start.sに記述）に設定してください。_ _exitにブレークポイントを設定してください。

ブレークポイントは必要に応じていくつか設定してください。

圧縮プログラム実行終了時には，jpegファイル格納用のバッファにでき上がったjpegファイルが格納されています。サンプルでは，関数get_Fsize（ ）の返り値がjpegファイルのサイズ（バイト数）です。

伸長プログラム実行終了時には，仮想のVRAMに伸長した結果の画像データが書き込まれます。サンプルでは，さらに，仮想のVRAMからBMPファイルを作成する関数が呼ばれます。この関数の実行終了時には，BMPファイル格納用のバッファにBMPファイルが書き込まれています。この関数の返り値がBMPファイルのサイズ（バイト数）になっています。

付録A　サンプル・プログラム・ソース・リスト（AP70732-B03用）

```
/************************************************************
 *  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*/
/*** VRAM を 8 or 16 行単位でアクセスする際にはこの define を有効にする ***/
/*#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 */

/* VRAM 関係の数値の define */
#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)

/* 画像の縦横ピクセル数 */
#define   IMAGE_WIDTH          224
#define   IMAGE_HEIGHT         144

/* 構造体の宣言 */
CJINFO              CJinfo;             /* structure for jpeg compress library */
DJINFO              DJinfo;             /* structure for jpeg decompress library */
APPINFO             cAppinfo, dAppinfo; /* structure for APPn segment */
/* 外部 RAM ワーク・エリアの宣言
 (ここでは、圧縮ライブラリと伸長ライブラリを同時に動かすことは考えていない)*/
unsigned char       WorkArea[0x1000];  /* library work area */
#define    JPEGBUFFSIZE       0x10000
/*#define  JPEGBUFSIZE        1*/
unsigned char       jpegbuffer[JPEGBUFFSIZE];
/* APPn セグメントに埋め込む文字列の例 */
unsigned char       str1[] = "JPEG middle-ware library.";
unsigned char       str2[] = "V810(uPD70732) 32-bit RISC Microcomputer";
/* コメント・マーカ */
unsigned char jpeg_COMStr[] = "This is a Comment Marker.";
```

保守／廃止

```
/***** 圧縮サンプル・プログラム *****/
void
move_jpeg()
{
        /* jpegbuffer の内容を退避する */
}

#ifdef    TINY_VRAM
void
replace_vram()
{
        /* VRAM を更新する */
}

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 */
```

保守／廃止

```
        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;
        CJinfo.DHT_AC_C_Bptr = (char *)DHT_markerChrominanceAC;
}

int
compress_test()
{

/*        12 MCU 分のビット数の目標値 */
#define    AVR_BITS   80*12

        int       Quality;
        int       i;
```

保守／廃止

```
        int         HEAP[3];
        short       xy[12][2]; /* 12 test point, ( x, y ) */
        short       width_tmp, height_tmp;

/*      以下のような 12 個の MCU に対してテストを行う */
/*      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   */
```

保守／廃止

```
        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];
        }
        /* 適切と思われる Quality ( 0 - 100 )を返す */
        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 */
```

保守／廃止

```
/***** 伸長／解析サンプル・プログラム *****/
/***** 解析サンプル・プログラム *****/
void
next_jpeg()
{
        /* jpegbuffer を更新する */
}

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);
                /* APPn セグメントに関して解析を行う場合 */
                /* 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 ? */
                }

        }
}

/***** 伸長サンプル・プログラム *****/
/*#define CLIPPING*/
#ifdef    TINY_VRAM
void
take_out_vram()
{
        /* VRAM の内容を退避する */
}
#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);
                /* APPn セグメントに関して解析を行う場合 */
```

保守／廃止

```
                /* 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 ? */
                }
        }
}
```

〔メ　モ〕

付録B　サンプル・プログラム・ソース・リスト（AP705100-B03用）

B.1　基本ライブラリ用サンプル・プログラム・ソース・リスト

```
/************************************************************
 *  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*/
/*** VRAM を 8 or 16 行単位でアクセスする際にはこの define を有効にする ***/
/*#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 */

/* VRAM 関係の数値の define */
#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)

/* 画像の縦横ピクセル数 */
#define   IMAGE_WIDTH           224
#define   IMAGE_HEIGHT          144

/* 構造体の宣言 */
CJINFO              CJinfo;           /* structure for jpeg compress library */
DJINFO              DJinfo;           /* structure for jpeg decompress library */
APPINFO             cAppinfo, dAppinfo; /* structure for APPn segment */
/* 外部 RAM ワーク・エリアの宣言
 (ここでは、圧縮ライブラリと伸長ライブラリを同時に動かすことは考えていない)*/
unsigned char       WorkArea[0x1000];   /* library work area */
#define    JPEGBUFFSIZE       0x10000
/*#define  JPEGBUFSIZE        1*/
unsigned char       jpegbuffer[JPEGBUFFSIZE];
/* APPn セグメントに埋め込む文字列の例 */
unsigned char       str1[] = "JPEG middle-ware library.";
unsigned char       str2[] = "V830(uPD705100) 32-bit RISC Microcomputer";
```

保守／廃止

```
/* コメント・マーカ */
unsigned char jpeg_COMStr[] = "This is a Comment Marker.";

/***** 圧縮サンプル・プログラム *****/
void
move_jpeg()
{
          /* jpegbuffer の内容を退避する */
}

#ifdef    TINY_VRAM
void
replace_vram()
{
          /* VRAM を更新する */
}

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 */
                    /* この場合 0x200 から 0x5FF までを使用します。 */
          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;
```

保守／廃止

```
        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()
{
                /* この場合 0x200 から 0x5FF までを使用します。 */
                /* 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()
{
```

保守／廃止

```
/*      12 MCU 分のビット数の目標値 */
#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;

/*      以下のような 12 個の MCU に対してテストを行う */
/*      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;
        }
```

保守／廃止

```
        /* 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];
        }
        /* 適切と思われる Quality ( 0 - 100 )を返す */
        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 */
```

保守／廃止

```
/***** 伸長／解析サンプル・プログラム *****/
/***** 解析サンプル・プログラム *****/
void
next_jpeg()
{
        /* jpegbuffer を更新する */
}

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 */
                /* この場合 0x200 から 0x2FF までを使用します。 */
        DJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* APPn セグメントに関して解析を行う場合 */
                /* 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 ? */
                }
        }
}


/***** 伸長サンプル・プログラム *****/
/*#define  CLIPPING*/
#ifdef     TINY_VRAM
void
take_out_vram()
{
        /* VRAM の内容を退避する */
}
#endif     /* TINY_VRAM */

int
decompress()
{
        int     ret;

        DJinfo.ErrorState = 0;          /* initialize */
                /* jpeg buffer start address */
```

保守／廃止

```
        DJinfo.JPEG_Buff_Bptr = jpegbuffer;
                /* jpeg buffer terminate address */
        DJinfo.JPEG_Buff_Eptr = (jpegbuffer + JPEGBUFFSIZE);
                /* internal data RAM work area start address */
                /* この場合 0x200 から最大 0x5FF までを使用します。 */
        DJinfo.IRAM_Buff_Bptr = (int *)0x200;
                /* APPn セグメントに関して解析を行う場合 */
                /* 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 ? */
                }
        }
}
```

保守／廃止

★ B.2　追加ライブラリ用サンプル・プログラム・ソース・リスト

（1）1パス・モードのサンプル

```
#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）2パス・モードのサンプル

```
#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 );
}
```

〔メ　モ〕

保守／廃止

★ 付録C　JPEGサンプル・ファイル（AP705100-B03追加ライブラリ用）

C.1　fishp3.jpg（プログレッシブ・スペクトラル・セレクション・フォーマット）

SOI	
APP0	JFIF
APPD	サムネール（ベースラインのJPEG形式）
COM	
APPE	
DQT	
SOF2	
DHT	
SOS	FF DA 00 0C 03 01 01 02 11 03 11 00 00 00
圧縮データ	Y, Cb, Cr成分のDC係数
SOS	FF DA 00 08 01 02 00 01 05 00
圧縮データ	Cb成分のAC1-AC5
SOS	FF DA 00 08 01 03 00 01 05 00
圧縮データ	Cr成分のAC1-AC5
SOS	FF DA 00 08 01 01 00 01 05 00
圧縮データ	Y成分のAC1-AC5
SOS	FF DA 00 08 01 02 02 06 3F 00
圧縮データ	Cb成分のAC6-AC63
SOS	FF DA 00 08 01 03 02 06 3F 00
圧縮データ	Cr成分のAC6-AC63
SOS	FF DA 00 08 01 01 01 06 3F 00
圧縮データ	Y成分のAC6-AC63
EOI	

保守／廃止

C.2　fishp4.jpg（プログレッシブ・サクセッシブ・アプロキシメーション・フォーマット）

SOI	
～	fishp3.jpgと同じ
DHT	
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 00
圧縮データ	Y, Cb, Cr成分のDC係数
SOS	FF DA 00 08 01 02 00 01 05 01
圧縮データ	Cb成分のAC1-AC5下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 03 00 01 05 01
圧縮データ	Cr成分のAC1-AC5下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 01 00 01 05 01
圧縮データ	Y成分のAC1-AC5下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 02 02 06 3F 01
圧縮データ	Cb成分のAC6-AC63下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 03 02 06 3F 01
圧縮データ	Cr成分のAC6-AC63下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 01 01 06 3F 01
圧縮データ	Y成分のAC6-AC63下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 02 03 01 3F 10
圧縮データ	Cb成分のAC1-AC63下位１ビット
SOS	FF DA 00 08 01 03 03 01 3F 10
圧縮データ	Cr成分のAC1-AC63下位１ビット
SOS	FF DA 00 08 01 01 03 01 3F 10
圧縮データ	Y成分のAC1-AC63下位１ビット
EOI	

保守／廃止

C.3　fishp5.jpg（プログレッシブ・サクセッシブ・アプロキシメーション・フォーマット）

SOI	
≀	fishp3.jpgと同じ
DHT	
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 01
圧縮データ	Y, Cb, Cr成分のDC係数，下位１ビットを除く上位ビット
SOS	FF DA 00 08 01 01 00 01 05 02
圧縮データ	Y成分のAC1-AC5下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 02 00 01 05 02
圧縮データ	Cb成分のAC1-AC5下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 03 00 01 05 02
圧縮データ	Cr成分のAC1-AC5下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 02 02 06 3F 02
圧縮データ	Cb成分のAC6-AC63下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 03 02 06 3F 02
圧縮データ	Cr成分のAC6-AC63下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 01 01 06 3F 02
圧縮データ	Y成分のAC6-AC63下位２ビットを除く上位ビット
SOS	FF DA 00 08 01 01 03 01 3F 21
圧縮データ	Y成分のAC1-AC63下位２ビット目
SOS	FF DA 00 08 01 02 03 01 3F 21
圧縮データ	Cb成分のAC1-AC63下位２ビット目
SOS	FF DA 00 08 01 03 03 01 3F 21
圧縮データ	Cr成分のAC1-AC63下位２ビット目
SOS	FF DA 00 0C 03 01 03 02 11 03 11 00 00 10
圧縮データ	Y, Cb, Cr成分のDC係数最下位ビット
SOS	FF DA 00 08 01 01 03 01 3F 10
圧縮データ	Y成分のAC1-AC63最下位ビット
SOS	FF DA 00 08 01 02 03 01 3F 10
圧縮データ	Cb成分のAC1-AC63最下位ビット
SOS	FF DA 00 08 01 03 03 01 3F 10
圧縮データ	Cr成分のAC1-AC63最下位ビット
EOI	

〔メ　モ〕

保守／廃止

付録D　総合索引

D.1　数字で始まる語句の索引

D.2　アルファベットで始まる語句の索引

保守／廃止

【W】

【Y】

保守／廃止

D.3　50音で始まる語句の索引

【ら行】

【わ行】

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン（インフォメーションセンター）
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−548−8899
FAX　：044−548−7900
E-mail ：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-8001	東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454−1111
中部支社	〒460-8525	愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222−2170 (052)222−2190
関西支社	〒540-8551	大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06)6945−3178 (06)6945−3200 (06)6945−3208

北海道支社	札　幌	(011)251−5599	宇都宮支店	宇都宮	(028)621−2281	北陸支社	金　沢	(076)232−7303
東北支社	仙　台	(022)267−8740	小山支店	小　山	(0285)24−5011	京都支社	京　都	(075)344−7824
岩手支店	盛　岡	(019)651−4344	甲府支店	甲　府	(055)224−4141	神戸支社	神　戸	(078)333−3854
郡山支店	郡　山	(024)923−5511	長野支社	松　本	(0263)35−1662	中国支社	広　島	(082)242−5504
いわき支店	いわき	(0246)21−5511	静岡支社	静　岡	(054)254−4794	鳥取支店	鳥　取	(0857)27−5311
長岡支店	長　岡	(0258)36−2155	立川支社	立　川	(042)526−5981,6167	岡山支店	岡　山	(086)225−4455
水戸支店	水　戸	(029)226−1717	埼玉支社	大　宮	(048)649−1415	松山支店	松　山	(089)945−4149
土浦支店	土　浦	(0298)23−6161	千葉支社	千　葉	(043)238−8116	九州支社	福　岡	(092)261−2806
群馬支店	高　崎	(027)326−1255	神奈川支社	横　浜	(045)682−4524			
太田支店	太　田	(0276)46−4011	三重支店	津	(059)225−7341			

保守／廃止

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］μSAP705100-B03, μSAP70732-B03　ユーザーズ・マニュアル
（U11052JJ4V0UM00（第４版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）（　　）
ご住所（　　）
お電話番号（　　）
お仕事の内容（　　）
お名前（　　）

１．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは
NEC販売員，特約店販売員，NEC半導体ソリューション技術本部員，
その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ