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

μSAP703000-B03

JPEG ミドルウエア

対象デバイス
　V850 シリーズ™

資料番号　U10684JJ3V0UM00（第３版）
発行年月　May 2002 N　CP(K)

〔メ　モ〕

目次要約

V850シリーズ，V850/SA1, V850/SB1, V850/SB2, V850/SC1, V850/SC2, V850/SC3, V850/SF1, V850/SV1, V850E/MS1, V850E/MS2, V850E/MA1, V850E/MA2, V850E/IA1, V850E/IA2, V853は日本電気株式会社の商標です。

Green Hills Softwareは米国Green Hills Software, Inc.の商標です。

UNIXはX/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。

Windowsは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

TRONは，The Real-Time Operating system Nucleusの略称です。

ITRONは，Industrial TRONの略称です。

- **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

本版で改訂された主な箇所(1/2)

箇　所	内　　容
p.19	1.2.1　JPEGの概要の記述を変更
p.20	1.2.1（2）YCbCr/RBGを変更
p.21	1.2.1（3）サンプリングとMCUを変更
p.36	1.2.2（1）（c）DQTマーカを変更
p.37	1.2.2（1）（d）DHTマーカを変更
p.39	1.2.2（1）（f）SOFnマーカを変更
p.40	1.2.2（1）（g）SOSマーカを変更
p.46	1.3　システム概要に追加ライブラリの記述を追加
p.51	図1－18　各ライブラリとサンプル・ソースを変更
p.54	表1－8　基本ライブラリROMサイズ（単位：バイト）を変更
p.54	表1－9　追加ライブラリROMサイズ（単位：バイト）を追加
p.56	表1－11　追加ライブラリRAMサイズ（単位：バイト）を追加
p.60	2.2.1（1）（b）jpeg_Compressの機能を変更
p.64	表2－1　ユーザ設定メンバー一覧に注を追加
p.70	2.2.2（1）（b）jpeg_Decompressの機能を変更
p.75	表2－10　伸長結果情報メンバー一覧を変更
p.79	2.2.3（1）（b）jpeg_Analysisの機能を変更
p.82	表2－14　ユーザ設定メンバー一覧に注意を追加
p.83	表2－16　解析結果情報メンバー一覧を変更
p.85	表2－18　JPEG対応フォーマットを変更
p.93	図2－16　JPEGファイル設定方法を変更
p.93	2.3.4　DNLマーカを変更
p.98	2.4　VRAMの構成を変更
p.99	図2－20　基本ライブラリのVRAM関連メンバ設定例を変更
p.118	2.7.2（4）Samplingを変更
p.120	2.7.2（13）VRAM_W_Pixelを変更
p.120	2.7.2（14）VRAM_H_Pixelを変更
p.122	2.7.2（29）APP_Info_Bptrを変更
p.123	2.7.2（33）DNLを変更
p.124	2.7.2（34）CI1, CI2, CI3を変更
p.125	2.7.3　APPINFO構造体の構造を変更
p.127	表2－33　シンボル一覧を変更
p.130	2.7.6（2）jpeg_Compressを変更
p.132	2.7.6（5）jpeg_Decompressを変更
p.134	2.7.6（8）jpeg_Analysisを変更

本版で改訂された主な箇所(2/2)

箇　所	内　　容
p.138	**第３章　プログレッシブ対応追加ライブラリ仕様**を追加
p.185	**４.１（２）UNIX™版の場合**を変更
p.185	**４.２　サンプル・プログラム実行手順**を変更
p.202	**Ａ.２　追加伸長処理サンプル・プログラム（1passモード）**を追加
p.206	**Ａ.３　基本圧縮，追加伸長処理サンプル・プログラム**を追加

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。

はじめに

対象者　このマニュアルは，V850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μSAP703000-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
- インストレーション
- 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「μSAP703000-B03」という製品名を「AP703000-B03」の名称で統一して説明しています。

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。

○V850シリーズに関する資料

品名＼資料名		データ・シート	ユーザーズ・マニュアル	
			ハードウエア編	アーキテクチャ編
V853™	μPD703003A, 703004A, 703025A, 703003A(A), 703025A(A)	U13188J	U10913J	U10243J
	μPD70F3003A, 70F3025A, 70F3003A(A)	U13189J		
V850/SA1™	μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY	U14526J	U12768J	
	μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AY	U14527J		
V850/SB1™	μPD703031A, 703031AY, 703033A, 703033AY, 70F3033A, 70F3033AY	U14734J	U13850J	
	μPD703032A, 703032AY, 70F3032A, 70F3032AY	U14893J		
V850/SB2™	μPD703034A, 703034AY, 703035A, 703035AY, 70F3035A, 70F3035AY	U14780J		
	μPD703037A, 703037AY, 70F3037A, 70F3037AY	U14894J		
V850/SC1™, V850/SC2™, V850/SC3™	μPD703068Y, 703069Y, 703088Y, 703089Y, 70F3089Y	作成予定	U15109J	
V850/SF1™	μPD703078Y, 703079Y, 70F3079Y	U15183J	U14665J	
V850/SV1™	μPD703038, 703038Y, 703039, 703039Y, 703040, 703040Y, 703041, 703041Y	U13953J	U14462J	
	μPD70F3038, 70F3038Y, 70F3040, 70F3040Y	U14462J		
V850E/MS1™	μPD703100-33, 703100-40, 703101-33, 703102-33	U13995J	U12688J	U12197J
	μPD703100A-33, 703100A-40, 703101A-33, 703102A-33	U14168J		
	μPD70F3102-33	U13844J		
	μPD70F3102A-33	U13845J		
V850E/MS2™	μPD703130	U15390J	U14985J	
V850E/MA1™	μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A)	U14792J	U14359J	U14559J
	μPD70F3107A, 70F3107A(A)	U14618J		
V850E/MA2™	μPD703108	作成予定	U14980J	
V850E/IA1™	μPD70F3116, 70F3116(A), 70F3116(A1)	U15299J	U14492J	
V850E/IA2™	μPD703114, 70F3114	作成予定	U15195J	

○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
IE-703002-MC（V853, V850/SA1, V850/SB1, V850/SB2, V850/SF1, V850/SV1用インサーキット・エミュレータ）		U11595J
IE-V850E-MC（V850E/IA1, V850E/IA2用インサーキット･エミュレータ）， IE-V850E-MC-A（V850E/MA1, V850E/MA2用インサーキット・エミュレータ）		U14487J
IE-703003-MC-EM1（V853用インサーキット・エミュレータ・オプション・ボード）		U11596J
IE-703017-MC-EM1（V850/SA1用インサーキット・エミュレータ・オプション・ボード）		U12898J
IE-703037-MC-EM1（V850/SB1, V850/SB2用インサーキット・エミュレータ・オプション・ボード）		U14151J
IE-703040-MC-EM1（V850/SV1用インサーキット・エミュレータ・オプション・ボード）		U14337J
IE-703079-MC-EM1（V850/SF1用インサーキット・エミュレータ・オプション・ボード）		U15447J
IE-703102-MC（V850E/MS1用インサーキット・エミュレータ）		U13875J
IE-703102-MC-EM1, IE-703102-MC-EM1-A（V850E/MS1用インサーキット・エミュレータ・オプション・ボード）		U13876J
IE-703107-MC-EM1（V850E/MA1用インサーキット・エミュレータ・オプション・ボード）		U14481J
IE-703117-MC-EM1（V850E/IA1用インサーキット・エミュレータ・オプション・ボード）		U14770J
CA850（Ver.2.30以上） （Ｃコンパイラ・パッケージ）	操作編	U14568J
	Ｃ言語編	U14566J
	プロジェクト・マネージャ編	U14569J
	アセンブリ言語編	U14567J
CA850（Ver.2.40以上） （Ｃコンパイラ・パッケージ）	操作編	U15024J
	Ｃ言語編	U15025J
	プロジェクト・マネージャ編	U15026J
	アセンブリ言語編	U15027J
ID850（Ver.2.40以上）（統合ディバッガ）	操作編　Windows®ベース	U15181J
SM850（Ver.2.40以上）（システム・シミュレータ）	操作編　Windowsベース	U15182J
SM850（Ver.2.00以上）（システム・シミュレータ）	外部部品ユーザ・オープン・インタフェース仕様編	U14873J
RX850（Ver.3.13以上）（リアルタイムOS）	基礎編	U13430J
	インストレーション編	U13410J
	テクニカル編	U13431J
RX850 Pro（Ver.3.13）（リアルタイムOS）	基礎編	U13773J
	インストレーション編	U13774J
	テクニカル編	U13772J
RX-NET（TCP/IPライブラリ）		U15083J
RD850（Ver.3.01）（タスク・ディバッガ）		U13737J
RD850 Pro（Ver.3.01）（タスク・ディバッガ）		U13916J
AZ850（Ver.3.0）（システム・パフォーマンス・アナライザ）		U14410J
PG-FP3（フラッシュ・メモリ・プログラマ）		U13502J
PG-FP4（フラッシュ・メモリ・プログラマ）		U15260J

・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

目　　次

図の目次（1/2）

図の目次（2/2）

表の目次（1/3）

表番号　　タイトル，ページ

表の目次（2/3）

表の目次（3/3）

表番号　　タイトル，ページ

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すようにチューニングされたソフトウエア群のことです。

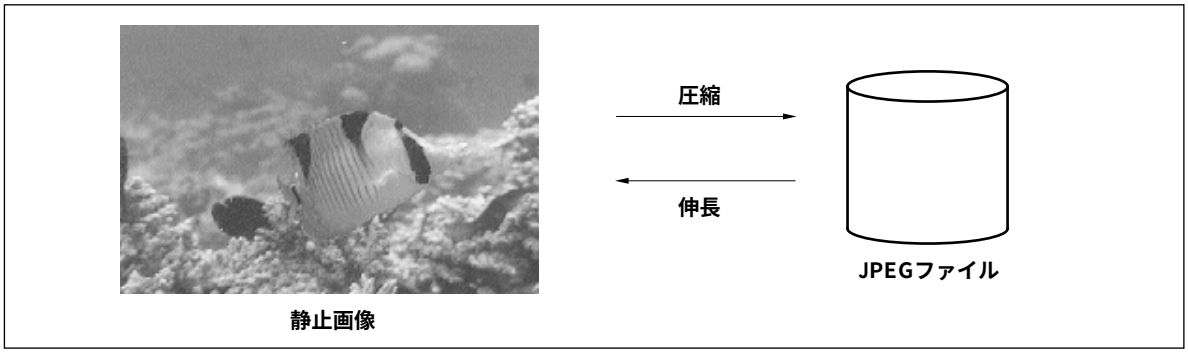
現在では高性能RISC（Reduced Instruction Set Computer）プロセッサが比較的安く市場に投入され，従来，専用ハードウエアに頼っていた処理を「高性能RISCプロセッサ」＋「ソフトウエア」というアプローチで実現できるようになりました。この「ソフトウエア」をミドルウエアと呼んでいます。

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



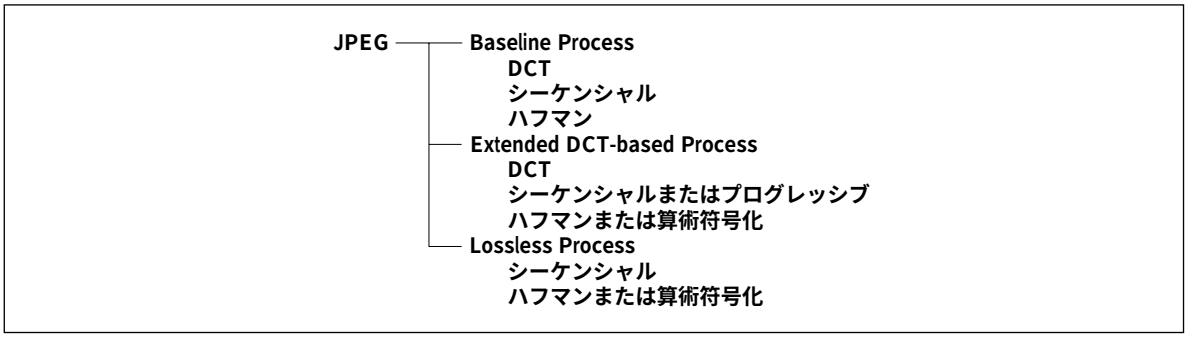
1.2.1　JPEGの概要

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。AP703000-B03では最も一般的なBaseline DCTに加えExtended DCT（プログレッシブを含む）と呼ばれるものに対応しています。

★　AP703000-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付けられます。

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。

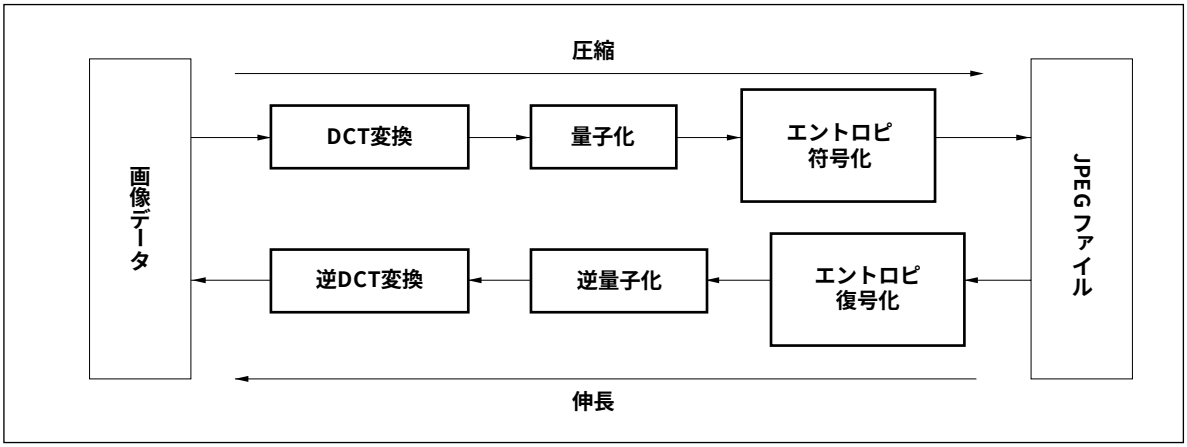
図１－２　JPEGの分類



（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，baseline DCTではハフマンを応用した技術を用いています。

AP703000-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。

★　（2）YCbCr/RGB

JPEGでは画像をYCbCrという色空間で圧縮／伸長します。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。

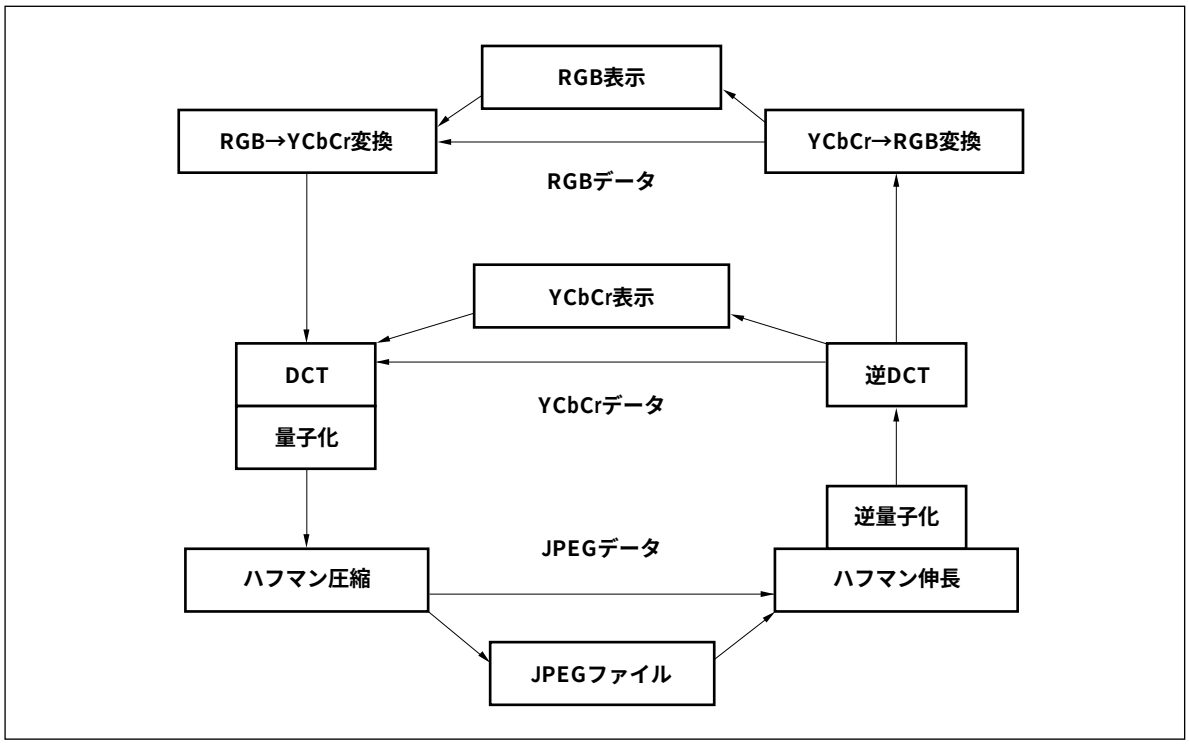
YCbCrのYは輝度（明るさの指標），Cb/Crは色差（Cbは緑から青への色調の差，Crは緑から赤への色調の差）のことで，RGBとの間には次のような変換式が成り立ちます。

次の式において，RGBがそれぞれ0-255の範囲であった場合，Yの範囲は0-255, Cb/Crは－128-127となります。

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

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

図１－４　JPEG処理概要



（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４：２：２，１MCU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４：４：４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	4：2：2 ［H：V＝2：1］	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	4：4：4 ［H：V＝1：1］	Y：１ブロック，Cb：１ブロック，Cr：１ブロック

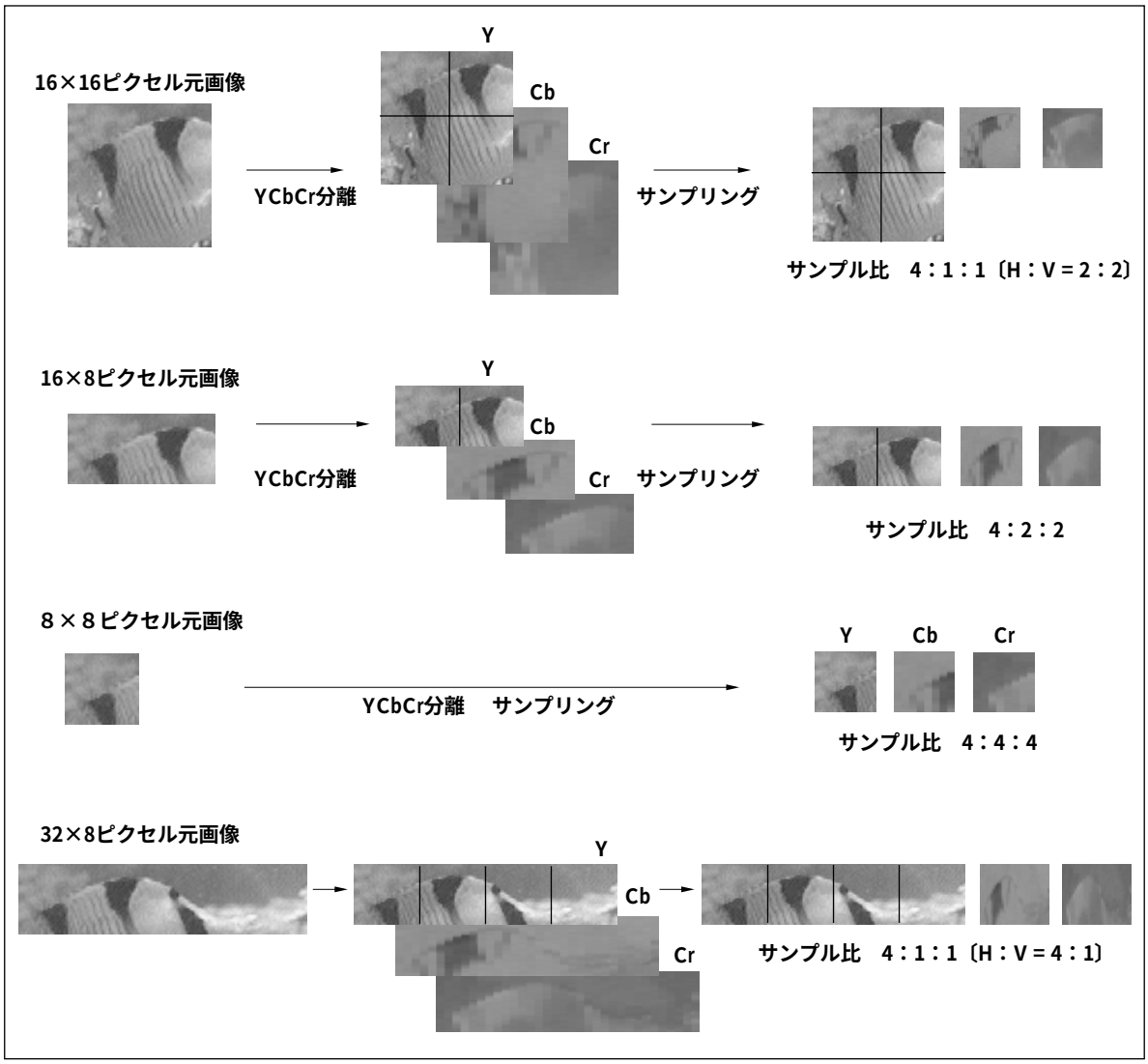
備考　H：MCUの横方向サンプリング比率
　　V：MCUの縦方向サンプリング比率

★ 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，AP703000-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，AP703000-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。

図１－５　画像のサンプリング



JPEGでは４：４：４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の場合		1MCUあたり	総ブロック数
	横	縦	ブロック数	
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
４：２：２［H：V＝２：１］	40分割	60分割	4	9600
４：４：４［H：V＝１：１］	80分割	60分割	3	14400

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

この表のように，サンプル比４：１：１よりも４：４：４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。

（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

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

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

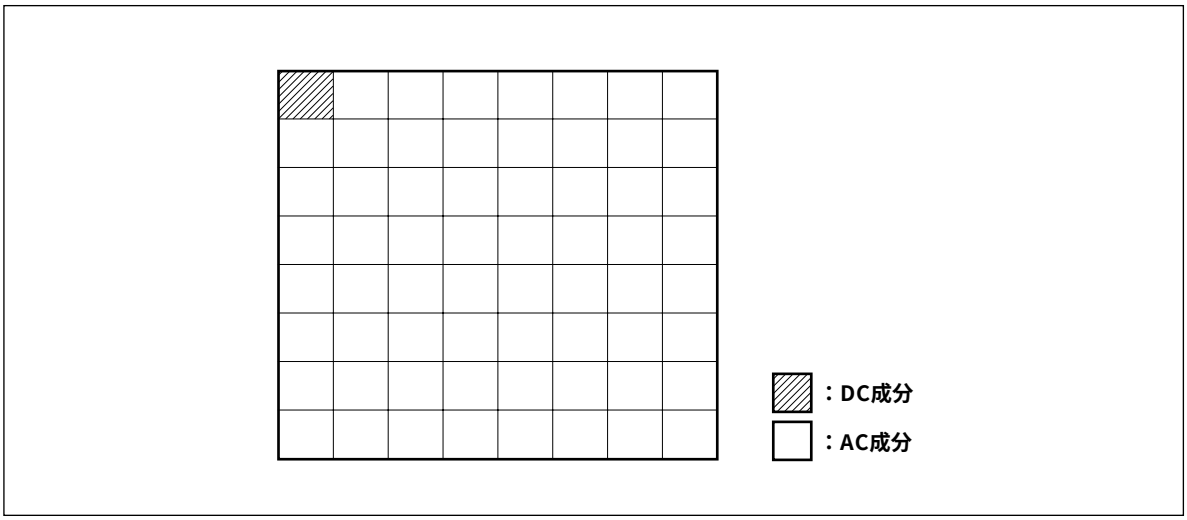
$$= 1 \qquad (w \neq 0)$$

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。

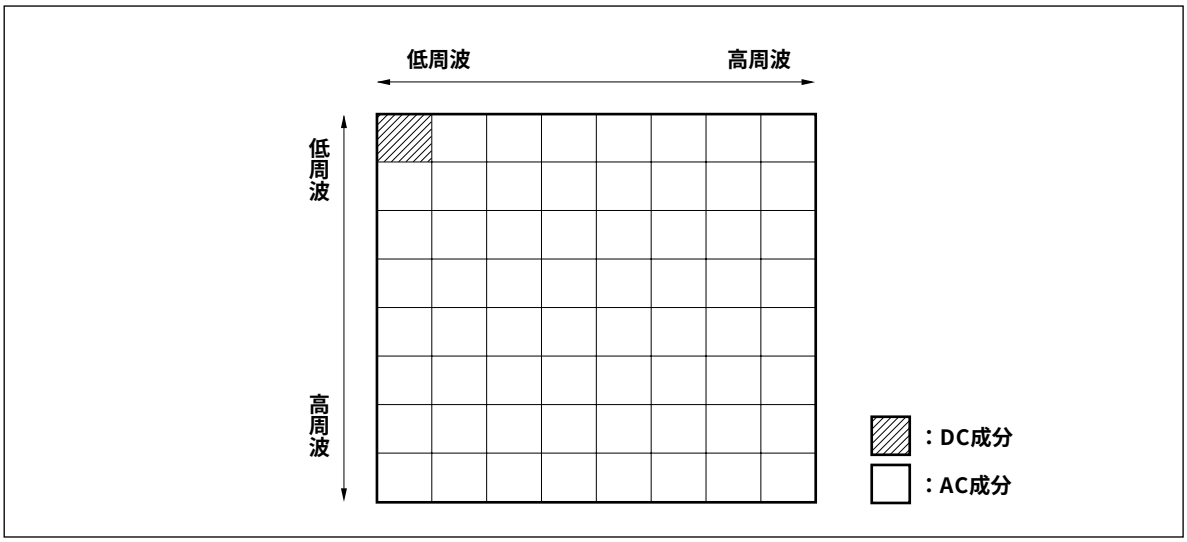
８×８ピクセルの画像データをDCT変換したとき，先頭の１要素はマトリクス全体の平均値を表し，それ以外の63要素はマトリクス内の色の歪み具合を表します。そこで信号処理では，DCT変換後のマトリクスの最初の１要素とそれ以外の63要素との特徴の違いから，最初の１要素だけをDC成分（直流成分）と呼び，残りの63要素をAC成分（交流成分）と呼んでいます。

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



（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

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

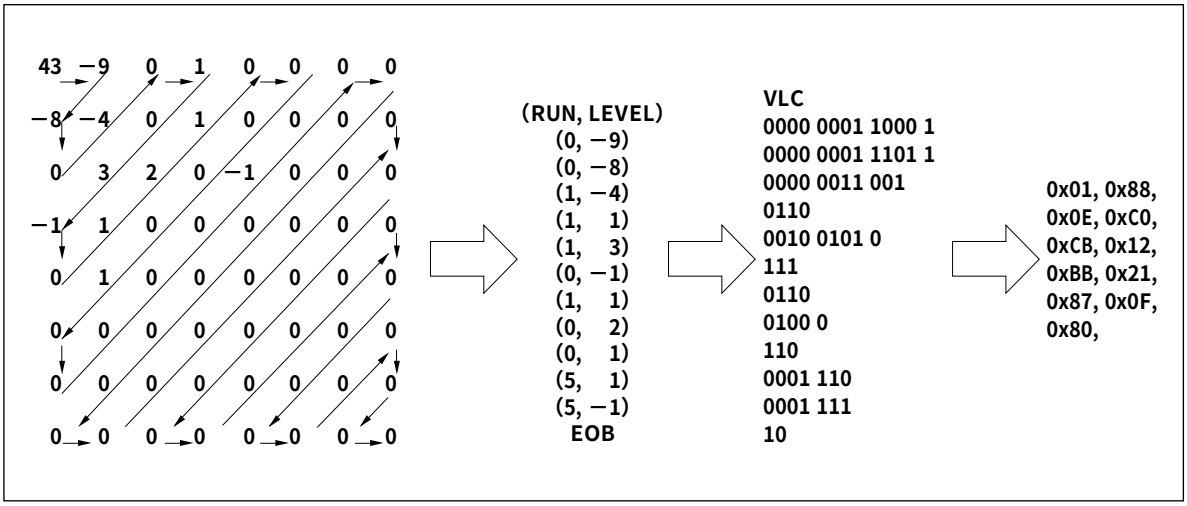
量子化された８×８行列の例

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

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

BaselineDCT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係数については，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。

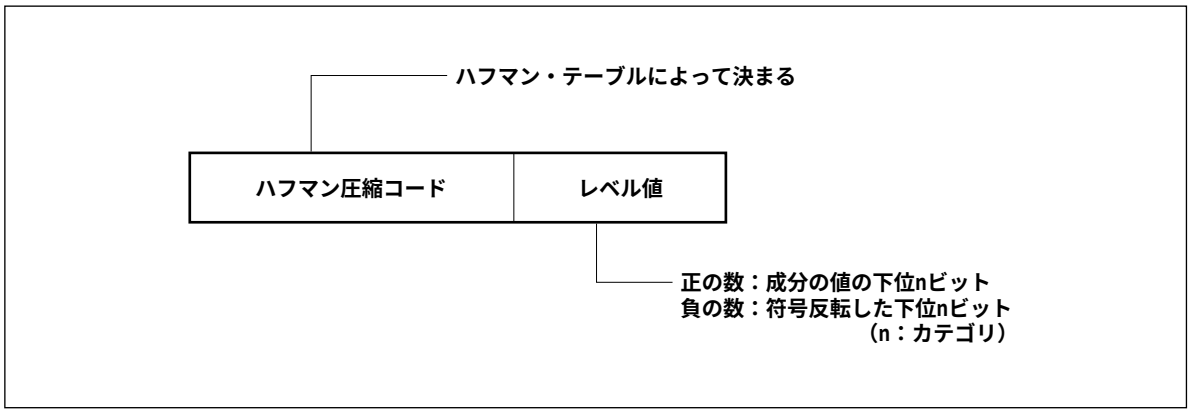
たとえば，輝度（Y）のDC成分用のハフマン・テーブルが，次のような規約になっていたとします。

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に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
⋮

（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１－11　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

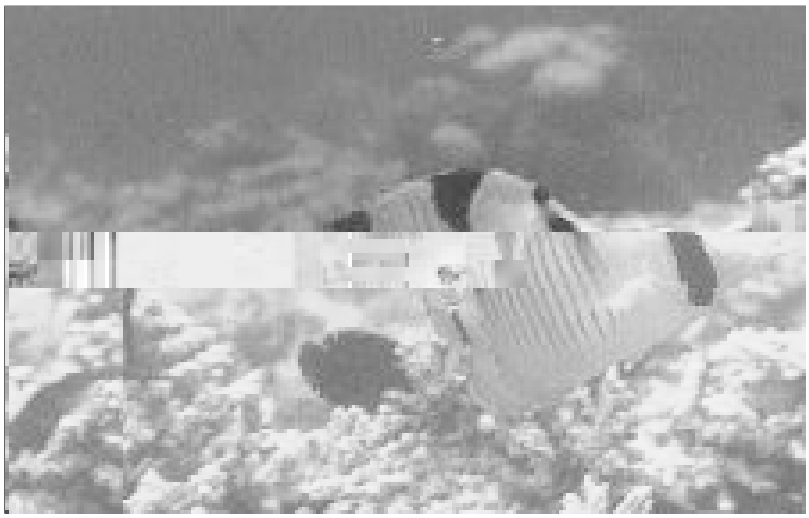
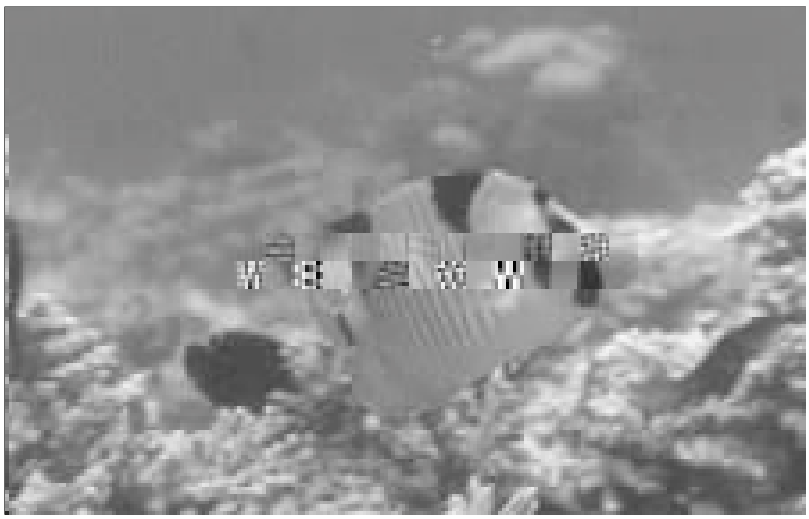


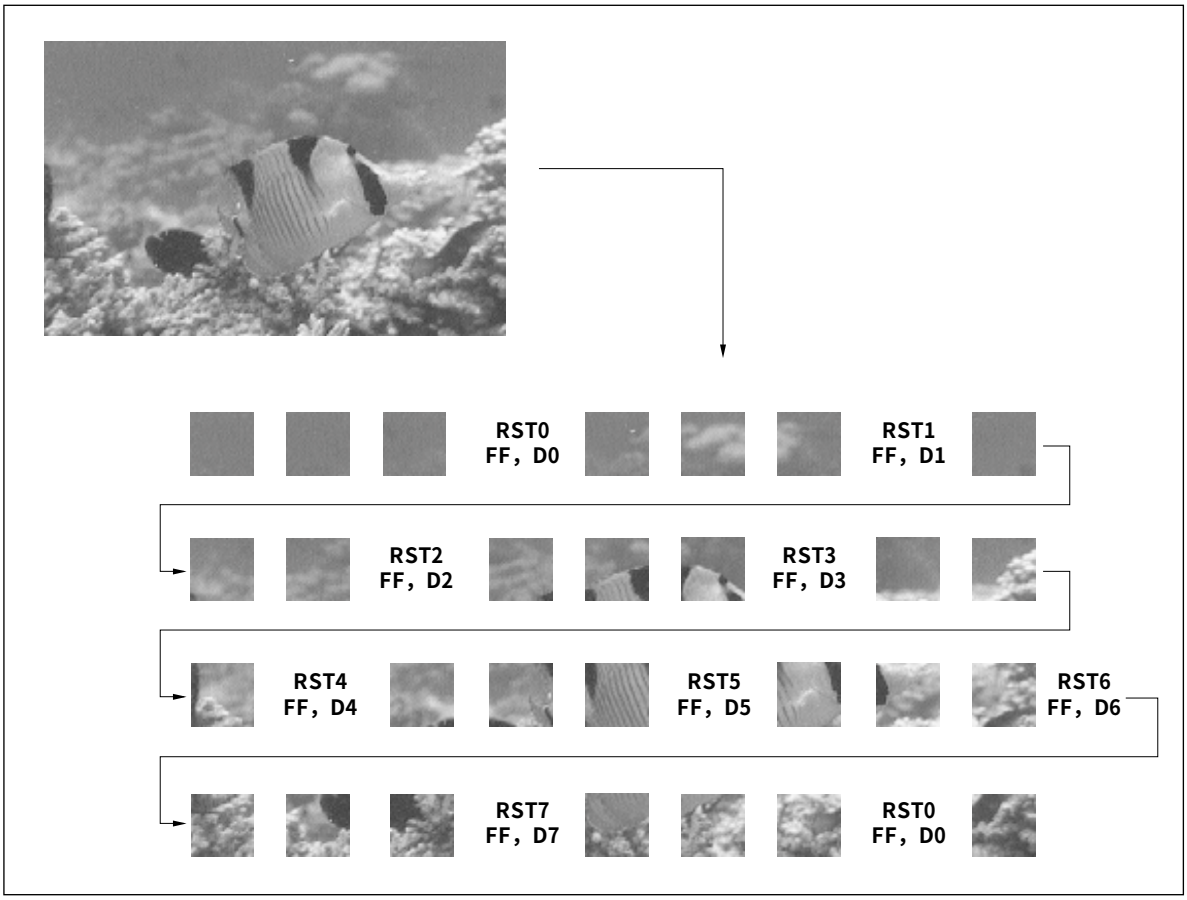
図１－12　リスタート・マーカを用いて途中から正しく伸長を再開できた例



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１－13　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４：２：２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。

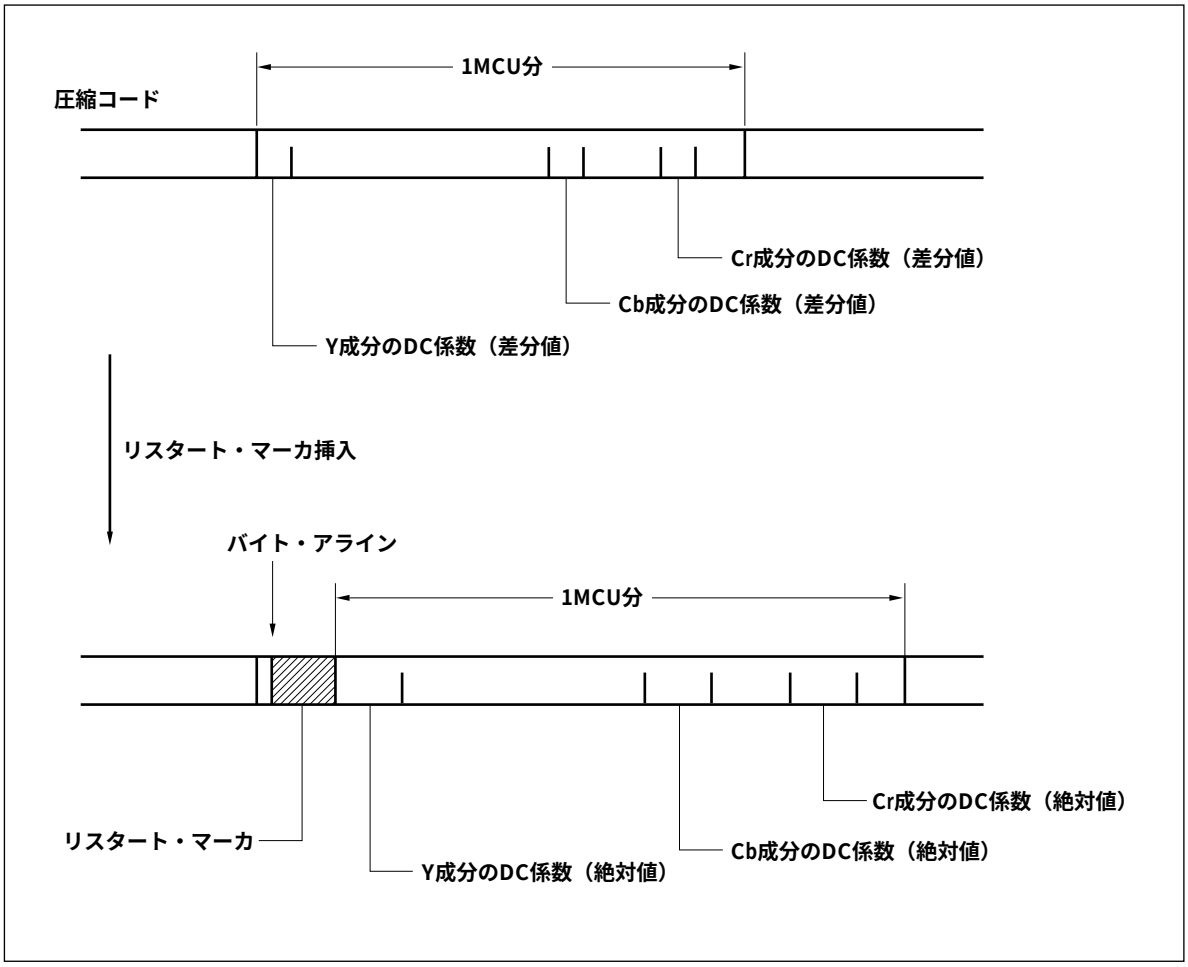
MCU（最小圧縮単位）：16×８ピクセル
リスタート・マーカ：MCU２個おき
したがって，

（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４：２：２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１－14　リスタート・マーカによるファイル・サイズの増加

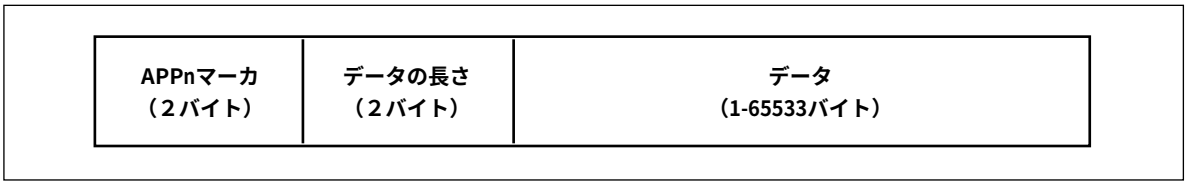


（8）APPnマーカ

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１－15　APPnセグメントの構造



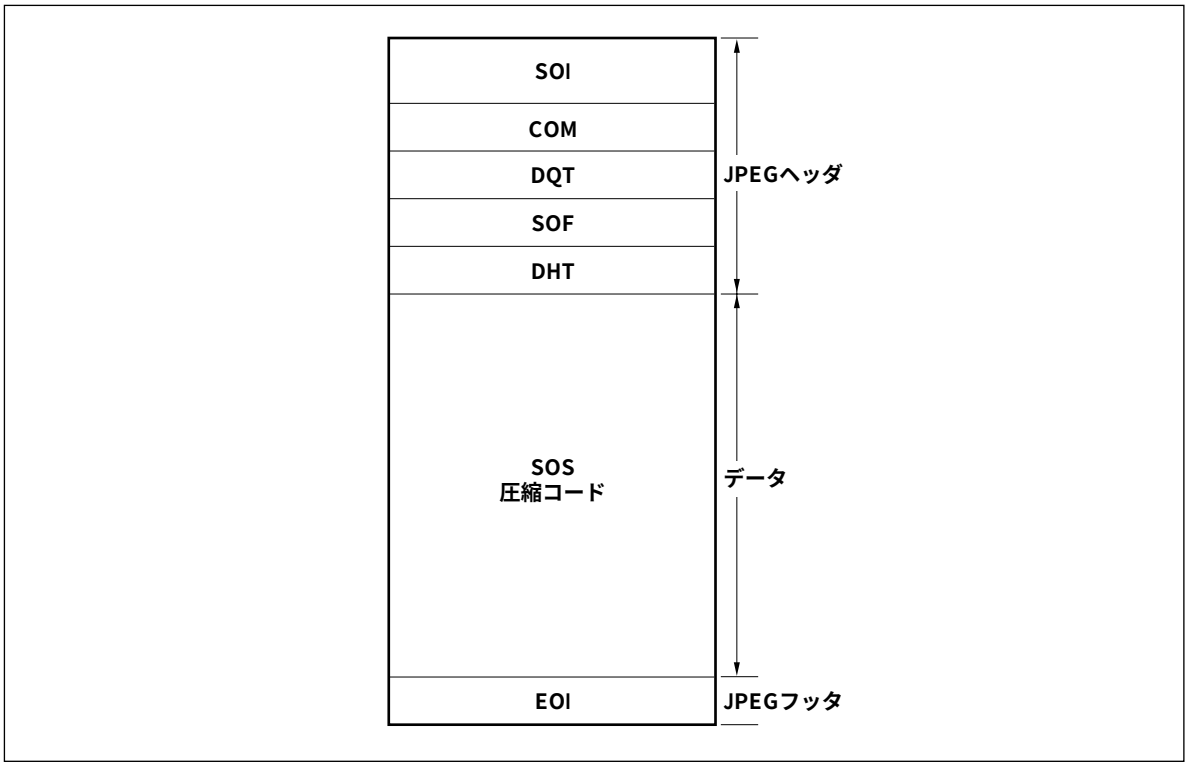
APPnマーカは0xFF,0xE0から0xFF,0xEFまでの16種類で，各APPnセグメントに対応しています。

AP703000-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。

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１－16　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。

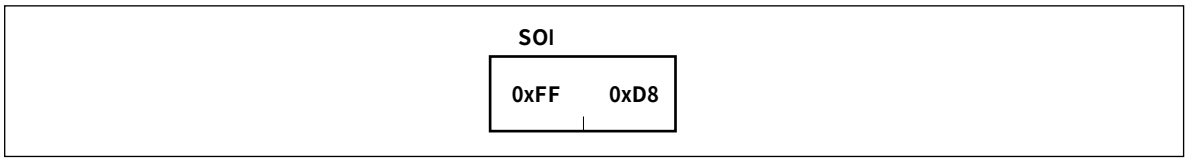
JPEGヘッダの各セグメント（COM, DQT, SOF, DHTなど）の順序に決まりはありません。

次にJPEGのマーカを示します。

表1－4　JPEGマーカ

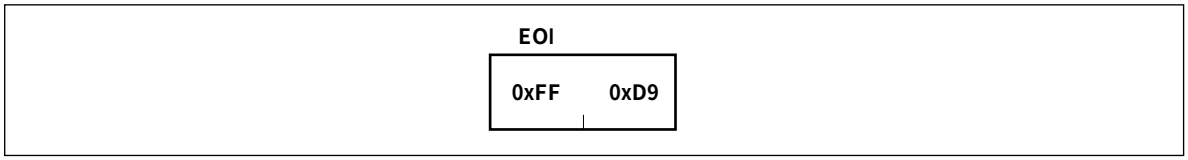
値	内　　容
0xFF 0x00	非マーカ（圧縮データの0xFF）
0xFF 0x01	TEM（算術圧縮の場合のテンポラリ・マーカ）
0xFF 0x02-0xFF 0xBF	RES（予約）
0xFF 0xC0	SOF0マーカ（Baseline DCT（Huffman））
0xFF 0xC1	SOF1マーカ（Extended sequential DCT（Huffman））
0xFF 0xC2	SOF2マーカ（Progressive DCT（Huffman））
0xFF 0xC3	SOF3マーカ（Spatial（sequential）lossless（Huffman））
0xFF 0xC4	DHT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	RSTm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



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。

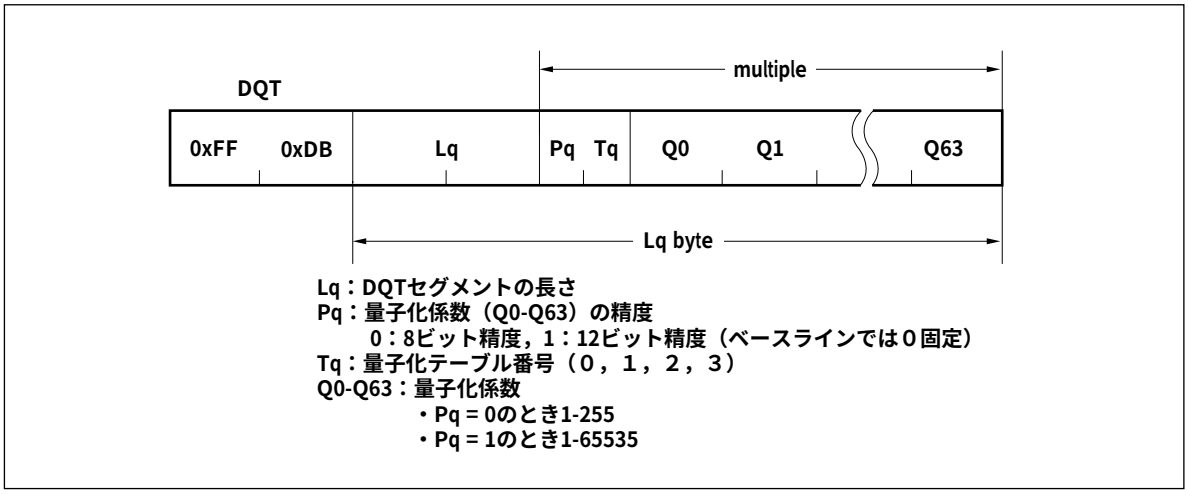
（b）EOI（End of image）マーカ



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。

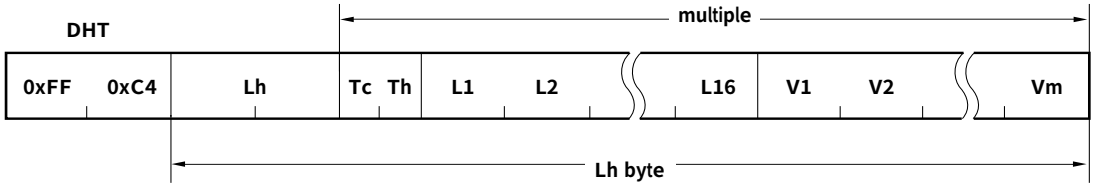


Lq：DQTセグメントの長さ
★ Pq：量子化係数（Q0-Q63）の精度
　　0：8ビット精度，1：12ビット精度（ベースラインでは０固定）
★ Tq：量子化テーブル番号（０，１，２，３）
Q0-Q63：量子化係数
　　・Pq = 0のとき1-255
　　・Pq = 1のとき1-65535

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。

（d）DHT（Define Huffman table（s））マーカ

ハフマン・テーブルを定義します。



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　：0, 1
・拡張ベースライン／プログレッシブ：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

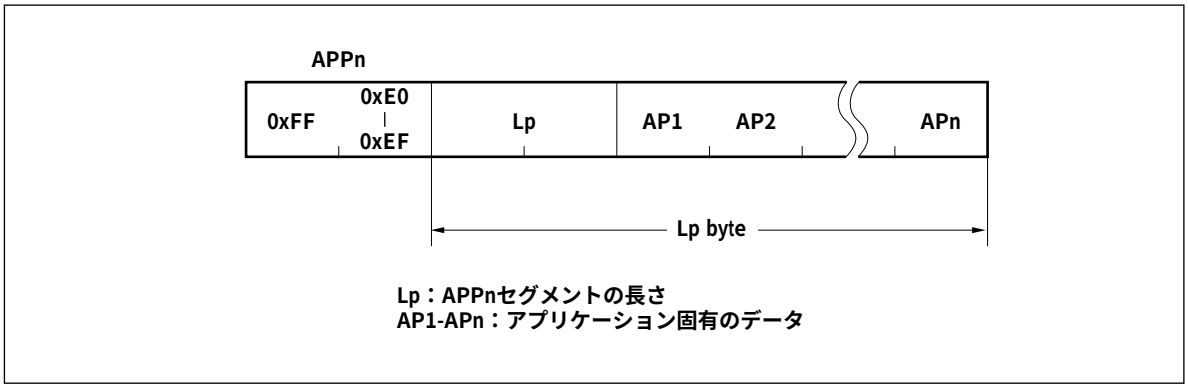
00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00

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



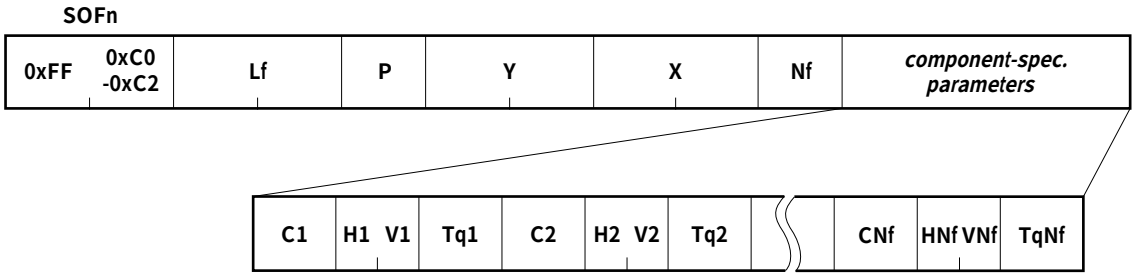
Lp：APPnセグメントの長さ
AP1-APn：アプリケーション固有のデータ

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。

（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セグメントでは，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。SOFnセグメントは特にフレーム・ヘッダとも呼ばれています。

また，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。



★

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
P：コンポーネントのビット数（24ビット・フルカラーの場合は，それぞれの色要素に対して８ビットを割り当てています。このとき，P = 8となります）
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（YCbCrの場合は色の数が３なので，Nf = 3）
Cn：ｎ番目のコンポーネントの番号
Hn：ｎ番目のコンポーネントの横方向サンプリング比
Vn：ｎ番目のコンポーネントの縦方向サンプリング比
Tqn：ｎ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

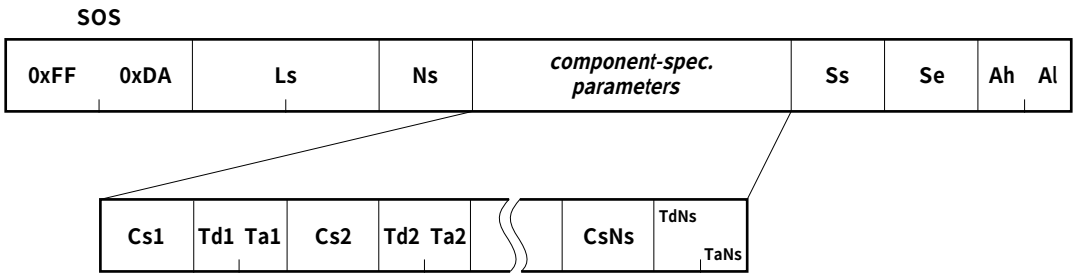
0xFF 0xC0, 0x00, 0x11, 0x08, 0x00, 0x90, 0x00, 0xE0, 0x03,
0x01, 0x22, 0x00, 0x02, 0x11, 0x01, 0x03, 0x11, 0x01

このセグメントの内容が上記のような値の場合は，次のような意味になります。

１番目のコンポーネント（Y）のサンプリング比が２×２で量子化テーブル番号が０
２番目のコンポーネント（Cb）のサンプリング比が１×１で量子化テーブル番号が１
３番目のコンポーネント（Cr）のサンプリング比が１×１で量子化テーブル番号が１

★ **（g）SOS（Start of scan）マーカ**

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。



Ls　：SOSセグメントの長さ
Ns　：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のいずれか
Csn：n番目のコンポーネントのID
Tdn：n番目のコンポーネントDC係数のハフマン・テーブル番号
　　ベースライン　　：0, 1
　　拡張ベースラインまたはプログレッシブ：0-3
Tan：n番目のコンポーネントのAC係数のハフマン・テーブル番号
　　ベースライン　　：0, 1
　　拡張ベースラインまたはプログレッシブ：0-3
Ss　：DCT係数の開始番号
　　ベースラインまたは拡張ベースライン：0に固定
　　プログレッシブ　　：0-63
Se　：DCT係数の終了番号
　　ベースラインまたは拡張ベースライン：63に固定
　　プログレッシブ　　：Ss－63（ただし，Ss = 63のときSe = 0固定）
Ah　：DCT係数の上位ビット
　　ベースラインまたは拡張ベースライン：0に固定
　　プログレッシブ　　：0-13
Al　：DCT係数の下位ビット
　　ベースラインまたは拡張ベースライン：0に固定
　　プログレッシブ　　：0-13

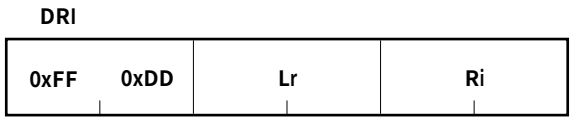
（h）DRI（Define restart interval）マーカとRSTm（Restart interval termination）マーカ

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係数は差分情報なので，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係数の値が必要ですが，リスタート・マーカの直後のDC係数は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。



Lr：DRIセグメントの長さ（4）
Ri：リスタート・インターバル

（2）データ

ヘッダ部分がバイト単位でファイルに格納されているのに対して，データ部分はビット単位でファイルに格納されています（データ全体ではバイトの倍数になります）。

データ部分についてまず注意しなければならないのが，マーカと区別するために，0xFFというデータが出現した場合にはその直後に0x00を挿入してあるという点です。したがって，伸長のときには，0xFFの直後の0x00は無視しなければなりません。そのためには，データをメモリから１バイトずつ持ってきて，それが0xFFであるか否かをそのつど比較する必要があります。データを２バイト／４バイト単位でメモリから持ってくるというような高速化のテクニックは使えません。

（3）ハフマン・デコードの実際

例

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

ハフマン・テーブル・セグメントが上記のような値の場合，このセグメントの内容から，次のように解釈します。

１ビット長のハフマン・コードは０個
２ビット長のハフマン・コードは‘ 00 ’の１個であり対応する値は‘ 0x00 ’
３ビット長のハフマン・コードは‘ 010 ’,‘ 011 ’,‘ 100 ’,‘ 101 ’,‘ 110 ’の５個であり，対応する値はそれぞれ，‘ 0x01 ’,‘ 0x02 ’,‘ 0x03 ’,‘ 0x04 ’,‘ 0x05 ’である。
４ビット長のハフマン・コードは‘ 1110 ’の１個であり対応する値は‘ 0x06 ’
.....

圧縮された画像データを伸長するたびにこの解釈を行っていたのでは，処理に時間がかかりすぎます。また，まったく同じ処理を何度も繰り返すのは時間の無駄でもあります。実際には，圧縮された画像データの伸長を開始する前に，ハフマン・テーブル・セグメントを解析して簡易的なテーブルを作成し，伸長時はこのテーブルを参照します。

（a）ハフマン・テーブル・イニシャライズ

JPEGの規格書では，次のようなテーブルの作成を推奨しています。

表１－５　規格書が推奨するハフマン・テーブル

ビット数	コードの最小値	コードの最大値	最小コードの番号
１ビット	不定	－1	不定
２ビット	0x0	0x0	0x0
３ビット	0x2	0x6	0x1
４ビット	0xE	0xE	0x6
５ビット	0x1E	0x1E	0x7
６ビット	0x3E	0x3E	0x8
７ビット	0x7E	0x7E	0x9
８ビット	0xFE	0xFE	0xA
９ビット	0x1FE	0x1FE	0xB
10ビット	不定	－1	不定
11ビット	不定	－1	不定
12ビット	不定	－1	不定
13ビット	不定	－1	不定
14ビット	不定	－1	不定
15ビット	不定	－1	不定
16ビット	不定	－1	不定

前述のハフマン・コード・セグメントの例では，３ビットのハフマン・コードの最小値は‘ 010 ’（= 0x02），最大値は‘ 110 ’（= 0x06），また最小コードの‘ 010 ’は‘ 00 ’に続く２番目（0x1番目），という意味になります。

（b）DC係数のデコード

具体例として表１－５のようなハフマン・テーブル・セグメントが与えられた場合を考えます。圧縮コードが次のような値であったとします。

0xDA 0x49（= 1101 1010 0100 1001）

まず最初に，圧縮データから初めの１ビット‘ １ ’（= 0x1）を，ハフマン・コード・テーブルの１ビットの最大値と比較します。

0x1＞－1（テーブルの１ビットの最大値）

したがって，圧縮コードは１ビット・コードではないことが分かります。

次に初めの２ビット‘ 11 ’（= 0x3）をテーブルの２ビットの最大値と比較します。

0x3＞0x0（テーブルの２ビットの最大値）

したがって２ビット・コードでもないことが分かります。

初めの３ビット‘ 110 ’（= 0x6）をテーブルの３ビットの最大値と比較します。

0x6 = 0x6（テーブルの３ビットの最大値）

これで３ビットの圧縮コードであることが確定します。

次に，この３ビットの圧縮コード‘ 110 ’が何番目のハフマン・コードであるかを求めます。圧縮コード‘ 0x6 ’からハフマン・テーブルの３ビットの最小値‘ 0x2 ’を引きます。

0x6－0x2 = 0x4

したがって，３ビット・コードとしては，0x4番目のコードであることが分かります。３ビットの最小コード‘ 0x2 ’は（テーブルを参照すれば）全体としては0x1番目のコードであるので，次のように計算します。

0x4＋0x1 = 0x5

これで全体としては0x5番目のコードであることがわかります。

次に，ハフマン・コード・セグメントを参照します。

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

このうち0x5番目の値は‘ 05 ’です。これは圧縮コードの次の“ 5 ”ビットがレベル値であることを意味しています。

次に，圧縮コードからデコードの終わった３ビット‘ 110 ’を捨てておきます。

0xDA 0x49（= 1101 1010 0100 1001）

↓

0xD2 0x48（=　　1 1010 0100 1001）

このうち初めの５ビットは，‘ 11010 ’（= 0x1A）です。この場合は一番先頭の符号ビットが‘ １ ’（‘ 正 ’であることを意味する）なので，求める値はそのまま‘ 0x1A ’です。これでDC係数のデコードの完了です。

（c）AC係数のデコード

基本的にはDC係数のデコードと同じですが，次の点が異なります。

・ハフマン・コード・テーブルが異なる。

・ハフマン・コード・セグメントからもってきた値の上位４ビットがゼロランの長さを表し，下位４ビットがレベル値のビット数を表す。

（4）逆量子化

逆量子化とは，ハフマン・デコードの結果得られた８×８マトリクスに，量子化セグメントで定義された量子化行列を掛け合わせる処理です。

この処理は，メモリから読んできた２つの値の積を求め，別のメモリ番地に書き込むだけの処理なので，ジグザグ・スキャンを元の順番に直す処理も同時に行います。同時に行うことでメモリ・アクセスの回数が減り，全体として高速化されます。

ハフマン・デコードで得られたマトリクスの要素のほとんどは，値が‘０’（ゼロ）です。また，一般的なＣＰＵでは乗算やメモリ・アクセス（読み書き）には何クロックもかかります。そこで，AP703000-B03では，要素の値が‘０’であるか否かを判定する式を１つ挿入し，‘０’であった場合ははじきます。

（5）DCT/逆DCTの実際

DCT変換は，次のような１次元ずつの計算を二度（縦と横）繰り返す処理と同等です。

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

２次元８次（N = 8；i.e.8×8マトリクス）の場合には，必要な定数は32個ですが，三角関数の公式などで絞り込むことができ，実際に必要な定数は次の７個になります。

cos（nπ/16）　（n = 0, 1, 2,...., 31）

↓

cos（nπ/16）　（n = 1, 2, 3, 4, 5, 6, 7）

また,あらかじめ次の値を求めておくと，乗算の回数が減ります。

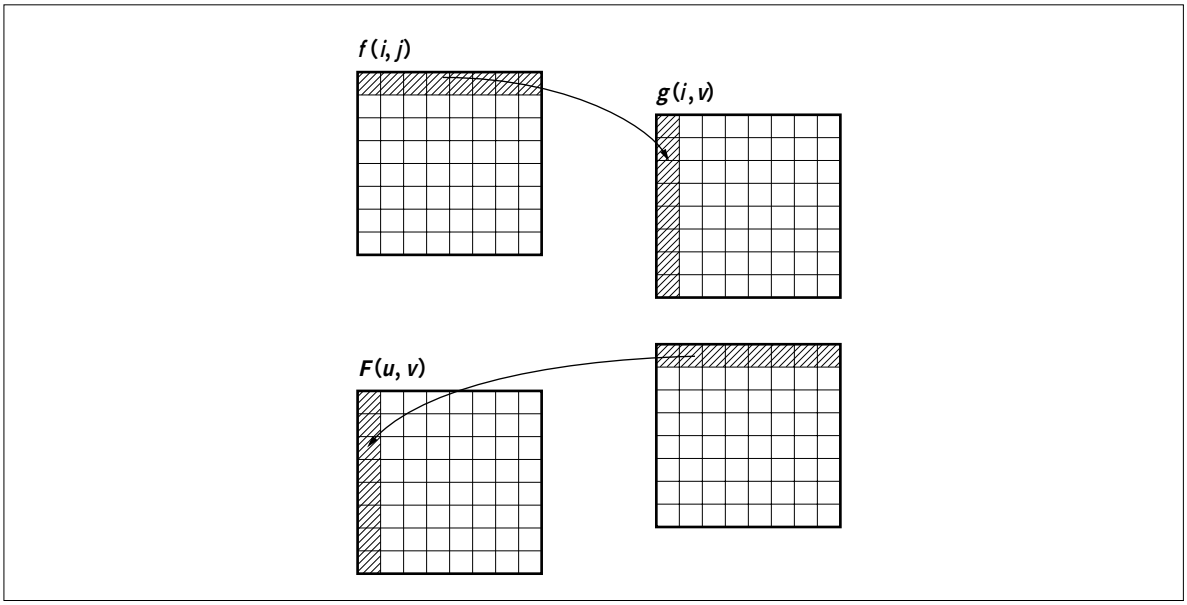
$f(0, j) \pm f(7, j)$, $f(1, j) \pm f(6, j)$, ...

さらに，DCTの式を次のようにすると，縦と横のルーチンを共通化できます。

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

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

図１－17　DCT/逆DCT変換アルゴリズム



実際にはDCTのルーチンに，引き数として*f*(*i*, *j*)のポインタ，ワーク領域*g*(*i*, *v*)のポインタを渡しています。このワーク領域を高速RAMに割り付けることで，処理を高速化できます。

★ 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１－６　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
AP703000-B03 （V850シリーズ対応）	基本ライブラリ（圧縮）	ベースライン
	基本ライブラリ（伸長）	
	基本ライブラリ（解析）	
	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

AP703000-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付けられます。

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あ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。

（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（輝度DC用，輝度AC用，色差DC用，色差AC用）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（リスタート・インターバル）

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，JPEGファイル内のDRIセグメントの値を使用します。

（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。

基本伸長では，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。

追加伸長では，APPnセグメント検出時にJPEGEXdecodeAPP関数（**3.5.3（2）JPEGEXdecodeAPP**参照）をコールします。

（6）DNLセグメント

圧縮では，DNLセグメントの挿入／非挿入の選択が可能です。

基本伸長では，DNLセグメントを自動認識します。

追加伸長では，オプションの設定が必要です（**3.6.1（3）（e）DNLEnable（Dオプション）**参照）。

（7）コンポーネントID

圧縮では，任意のコンポーネントIDを設定できます。

伸長では，コンポーネントIDを自動認識します。

（8）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。

1.3.3　基本ライブラリの特徴

（1）JPEGファイル格納用バッファ

圧縮／伸長／解析いずれも基本ライブラリを実行するためにJPEGファイルを格納するバッファが必要です。バッファ・サイズより大きなJPEGファイルを扱う場合は，バッファの終端で処理を中断します。ユーザがバッファ処理（圧縮時のバッファ退避，伸長／解析時のバッファ更新）を行い，基本ライブラリを再コールすることにより，処理を継続します。JPEGファイル格納用バッファのサイズは，１バイト以上の任意の値を設定できます。

（2）ライン処理

圧縮／伸長いずれも，1MCUライン単位で処理を中断できます。

この機能を使用することにより，画像メモリ（ex. VRAM）が１枚の画像すべてを保持できないようなシステムの場合でも，圧縮／伸長が可能です。

（3）カラーとモノクロ

圧縮／伸長いずれも，カラー画像とモノクロ画像に対応しています。

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

また，使用されるサンプル比を限定することで，コード・サイズの節約も可能です。

1.3.4　追加ライブラリの特徴

（1）JPEGファイル格納用バッファ

追加ライブラリを実行するために，JPEGファイルを格納するバッファが必要です。ライブラリは，ユーザ作成のJPEG取得関数（JPEGEXGetJpegStream）（**3.5.3（1）JPEGEXGetJpegStream**参照）をコールすることでJPEGバッファを更新します。

JPEGファイル格納用バッファのサイズは，32バイトより大きな値で設定してください（**3.6.4 JPEGEXBUFF構造体**参照）。

（2）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

（3）サンプル比

任意のサンプル比に対応しています。

（4）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。

（5）色要素

単色，３色フォーマットに加え，４色フォーマットにも対応しています。

（6）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（**3.6.3　JPEGEXVIDEO構造体**参照）。

（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（**3.6.1　JPEGEXINFO構造体**参照）。

（8）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（**3.6.1　JPEGEXINFO構造体**参照）。

1.3.5　基本ライブラリと追加ライブラリの違い

AP703000-B03の基本ライブラリと追加ライブラリの違いを表１－７に示します。

表１－７　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	×	○[注]
		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） ４：２：２（H：V＝２：１） ４：４：４	任意のサンプル比に対応
	色要素		単色，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		×	ピクセル単位
	伸長時の縮小／拡大		×	n/8の比率で，縮小／拡大が可能

注　16ビット量子化テーブルに対応していません。16ビットの量子化テーブルを持つJPEGファイルを伸長した場合，異常終了します（**3.7　追加伸長時のエラー内容**参照）。

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。

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。

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。

★ **図1－18　各ライブラリとサンプル・ソース（1/3）**

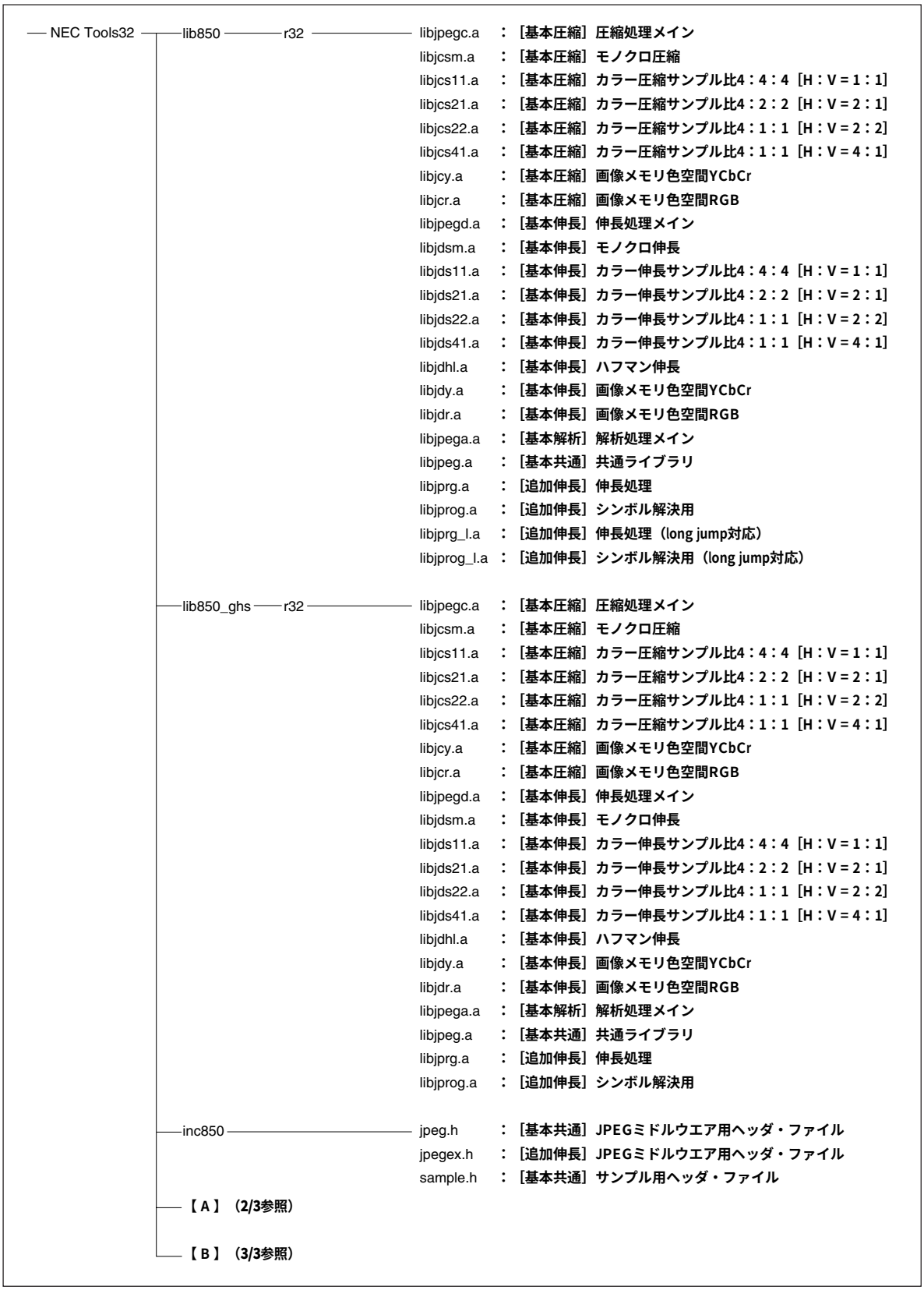


図1－18　各ライブラリとサンプル・ソース（2/3）

図1－18　各ライブラリとサンプル・ソース（3/3）

1.3.7 動作環境

（1）対象CPU

V850シリーズ

（2）使用コンパイラ・パッケージ

GHS（Green Hills Software, Inc.）社製コンパイラ

CCV850 Ver.4.0.2

NEC製コンパイラ・パッケージ

CA850 Ver.2.40

（3）メモリ容量

各ライブラリで必要なROM/RAMサイズは次のとおりです。ただし,合計はサンプル比の選択などにより異なります。

また，RGBを使用する場合はさらにメモリが必要となりますので，ご注意ください。

★ 表1－8　基本ライブラリROMサイズ（単位：バイト）

	必須バイト数	オプション（対応サンプル比の選択）				
		4：4：4[H：V＝1：1]	4：2：2[H：V＝2：1]	4：1：1[H：V＝2：2]	4：1：1[H：V＝4：1]	mono
圧縮処理	6.0 K	＋0.6 K	＋0.8 K	＋1.2 K	＋1.2 K	＋0.5 K
伸長処理	5.0 K	＋0.5 K	＋0.6 K	＋0.8 K	＋0.8 K	＋0.5 K
解析処理	2.0 K	－	－	－	－	－

★ 表1－9　追加ライブラリROMサイズ（単位：バイト）

	NEC製コンパイラ	GHS製コンパイラ
追加ライブラリ	48 K	55 K
追加ライブラリ（long jump対応）	55 K	64 K

表１－10　基本ライブラリRAMサイズ（単位：バイト）（1/2）

（a）圧縮処理

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③ワーク２	3072	
④MCUバッファ	128-768	128：対応サンプル比　mono 384：対応サンプル比　mono, 4：4：4 512：対応サンプル比　mono, 4：4：4, 4：2：2 768：対応サンプル比　mono, 4：4：4, 4：2：2, 4：1：1
⑤スタック	128	
⑥JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑦画像メモリ	（システム依存）	画像イメージ格納メモリ（たとえば，VRAM）。
⑧APPINFO構造体	96	APPnセグメントを使用する場合のみ必要。
⑨APPnデータ	（システム依存）	APPnセグメントを使用する場合のみ必要。
小計	4372[注]	

注　①～⑤を足した値です（MCUバッファは最大時）。

（b）伸長処理

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③ワーク２	8192	
④MCUバッファ	128-768	128：対応サンプル比　mono 384：対応サンプル比　mono, 4：4：4 512：対応サンプル比　mono, 4：4：4, 4：2：2 768：対応サンプル比　mono, 4：4：4, 4：2：2, 4：1：1
⑤スタック	128	
⑥JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑦画像メモリ	（システム依存）	画像イメージ格納メモリ（たとえば，VRAM）。
小計	9492[注]	

注　①～⑤を足した値です（MCUバッファは最大時）。

表１－10　基本ライブラリRAMサイズ（単位：バイト）（2/2）

（ｃ）解析処理

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③スタック	128	
④JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑤APPINFO構造体	96	APPnセグメントを使用する場合のみ必要。
⑥APPnデータ	（システム依存）	APPnセグメントを使用する場合のみ必要。
小計	532[注]	

注　①～③を足した値です。

★ 表１－11　追加ライブラリRAMサイズ（単位：バイト）

処理系	内　容	サイズ	備　　考
追加ライブラリ１パス設定	ワーク・エリア[注]	約 2 M	このRAMサイズは，640×480ピクセル，３色の場合のみ，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	－
追加ライブラリ２パス設定	ワーク・エリア[注]	約 5000	－
	スタック	約 500	－
	小計	約 5500	－

注　追加ライブラリのワーク・エリアの詳細については，**表３－10　パス回数による伸長処理の違い**を参照してください。

備考　ライブラリを実行させるためには，表に示す必要バイト数以外に，JPEGファイルを格納するバッファ，画像データを保持するメモリ（VRAMなど：伸長した画像データを出力するためのメモリ）が必要です。

第２章　基本ライブラリ仕様

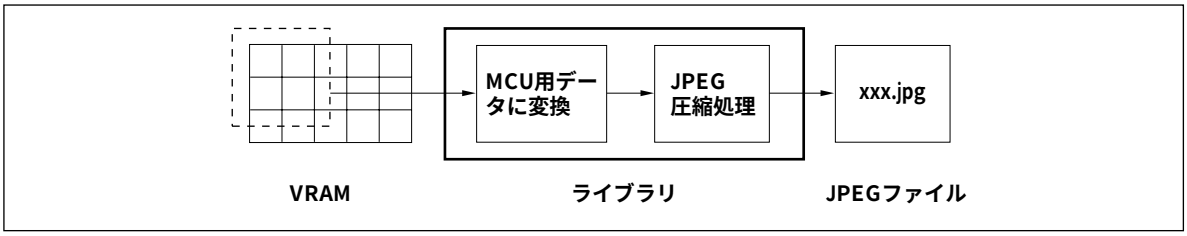
2.1　機　　能

AP703000-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３つの処理が実現できます。

（1）圧縮処理

画像データを圧縮し，JPEGファイルを生成します。

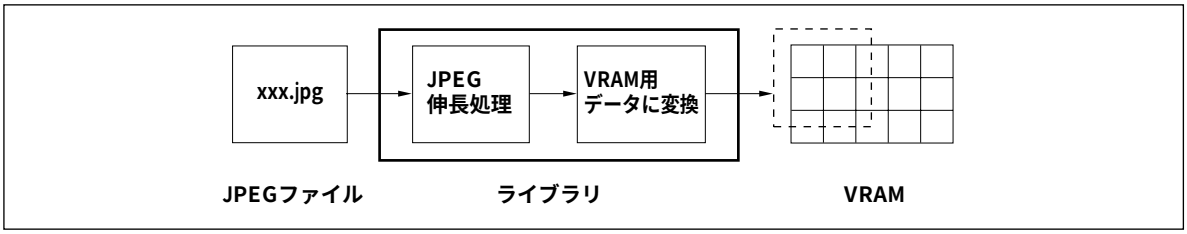
図２－１　圧縮処理



（2）伸長処理

JPEGファイルを伸長し，画面に表示します。

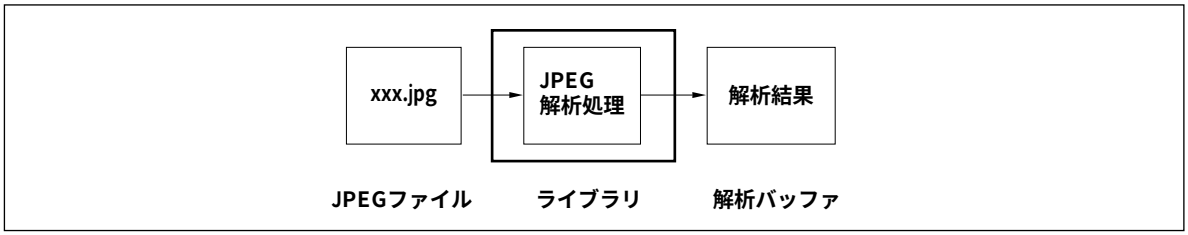
図２－２　伸長処理



（３）解析処理

JPEGファイルから画像サイズなどの情報を得ます。

図２－３　解析処理



２.２　処理詳細

２.２.１　圧縮処理

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。

（１）関数

分類	関数名	機能概要
圧縮処理	jpeg_CompressInit	JPEG圧縮ライブラリ初期化
	jpeg_Compress	JPEG圧縮処理メイン
	jpeg_getMCU_M[注]	VRAMアクセス関数（モノクロ圧縮）
	jpeg_getMCU_11[注]	VRAMアクセス関数（カラー圧縮サンプル比4：4：4［H：V＝1：1］）
	jpeg_getMCU_21[注]	VRAMアクセス関数（カラー圧縮サンプル比4：2：2［H：V＝2：1］）
	jpeg_getMCU_22[注]	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝2：2］）
	jpeg_getMCU_41[注]	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝4：1］）

注　カスタマイズする場合のみ参照してください。

（ａ）jpeg_CompressInit

分　　類　圧縮処理

関 数 名　jpeg_CompressInit

機能概要　JPEG圧縮ライブラリの初期化

形　　式　#include “ jpeg.h ”

void jpeg_CompressInit（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**２.７.２　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG圧縮ライブラリを初期化し，圧縮処理を実行可能な状態にします。

（b）jpeg_Compress

分　　類　圧縮処理

関 数 名　jpeg_Compress

機能概要　JPEG圧縮メイン

形　　式　#include “ jpeg.h ”

long jpeg_Compress（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ退避要求）
JPEGRET_CONT2	継続２（画像データ更新要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます）

★　機　　能　*info*で指定されるJPEG圧縮を行います。

異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_VRAM	圧縮画像の領域指定が不正です。
JPEGERR_USF	圧縮できないサンプル比です。
JPEGERR_IMY	圧縮画像の縦サイズが不正です。
JPEGERR_FID	コンポーネントIDの値が重複しています。
JPEGERR_DLH	DHTヘッダのLhの値が不正です。
JPEGERR_DCL	DC成分用ハフマン・テーブルが不正です。
JPEGERR_JPB	JPEGバッファ・サイズが不正です。

（２）メモリ資源

圧縮ライブラリを使用する場合に最低限必要なメモリ（RAM）容量は，次のとおりです。

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③ワーク２	3072	
④MCUバッファ	128-768	128：対応サンプル比　mono 384：対応サンプル比　mono, 4：4：4 512：対応サンプル比　mono, 4：4：4, 4：2：2 768：対応サンプル比　mono, 4：4：4, 4：2：2, 4：1：1 **２.５　MCUバッファの構造**参照。
⑤スタック	128	
⑥JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑦画像メモリ	（システム依存）	画像イメージ格納メモリ（たとえば，VRAM）。**２.４　VRAMの構成**参照。 最低，1MCUライン分は，必要。
⑧APPINFO構造体	96	APPnセグメントを使用する場合のみ必要。
⑨APPnデータ	（システム依存）	APPnセグメントを使用する場合のみ必要。
小計	4372[注]	

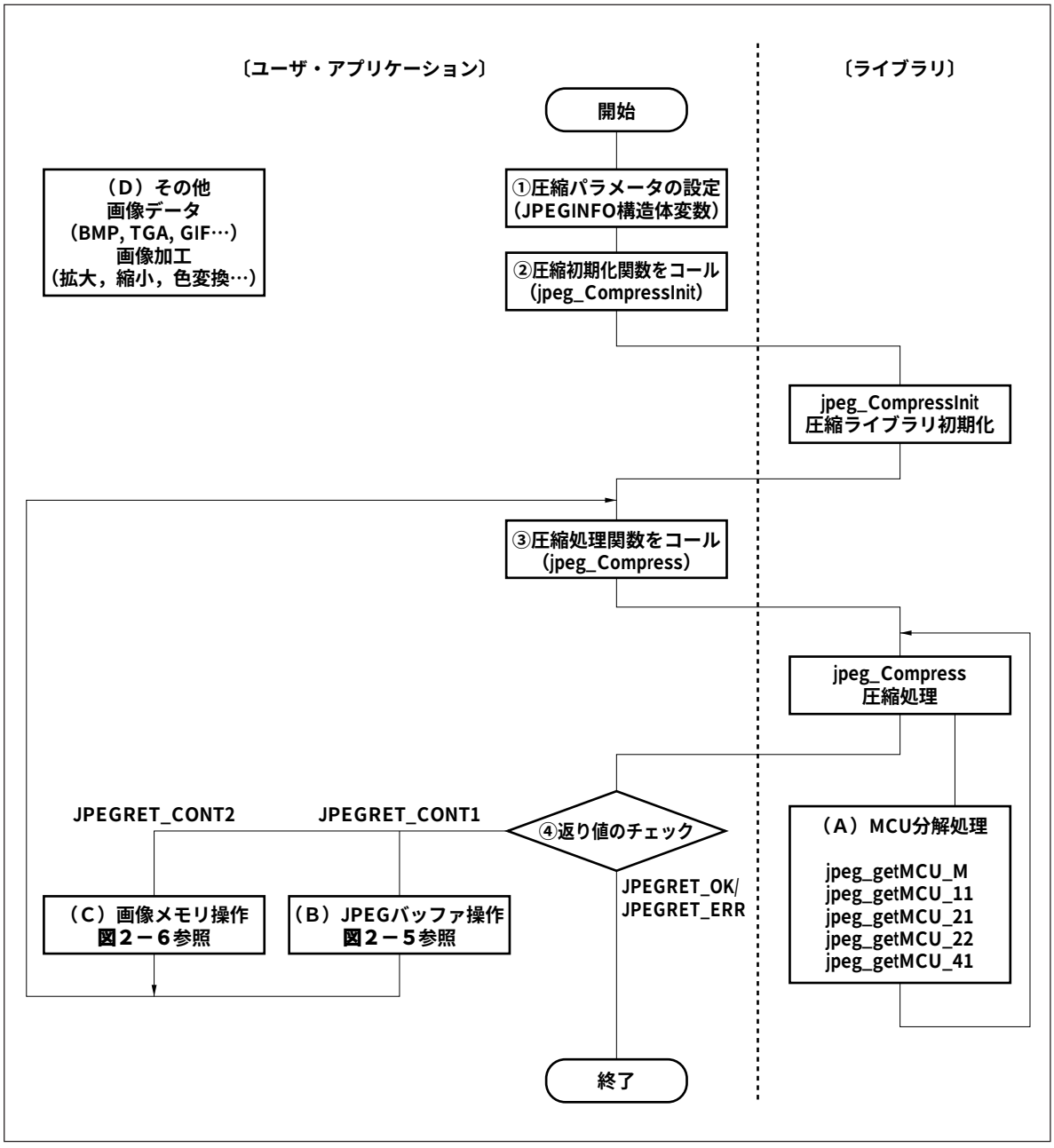
注　①～⑤を足した値です（MCUバッファは，最大時）。

注意　バイト／ハーフワード／ワード単位でランダム・アクセスできるメモリが必要です。

（３）圧縮処理フロー

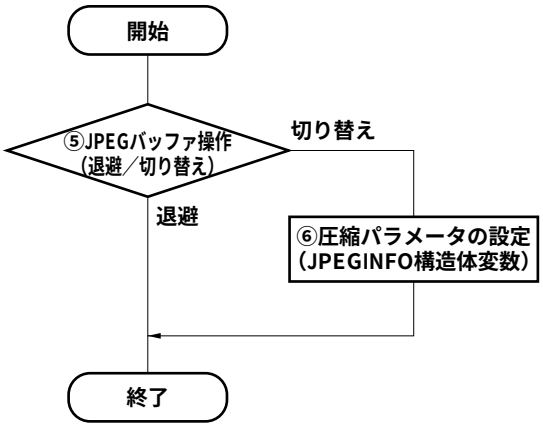
圧縮ライブラリを使用したアプリケーションの処理の流れを図２－４から図２－６に示します。

図２－４　圧縮処理の流れ



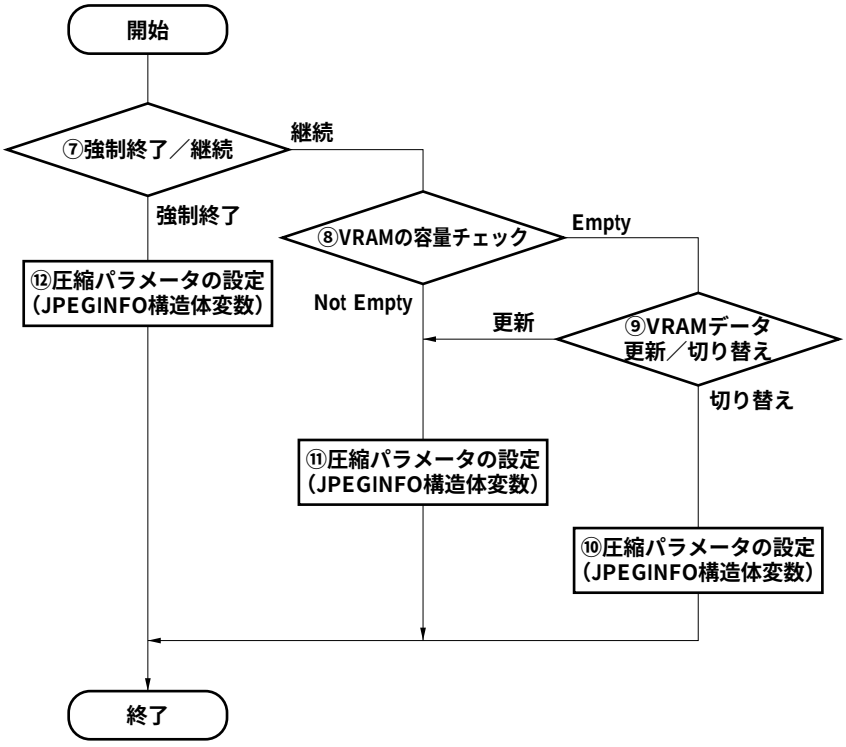
圧縮ライブラリを使用する場合にユーザ・アプリケーション部分の処理として最低限必要な処理は，図２－４の①～④です。必要に応じて（A），（B），（C），（D）の処理も必要となる場合があります。

図２－５　JPEGバッファ操作



備考　１枚分のJPEGデータを格納するメモリが用意できず，何回かに分けて圧縮する場合に必要な処置です。

図２－６　画像メモリ操作



備考　ライン単位圧縮機能を使用する場合に必要な処理です。圧縮画像を外部記憶媒体や通信で転送する場合の処理例です。

①　圧縮パラメータの設定

ヘッダ・ファイル“jpeg.h”で定義されているJPEGINFO構造体の変数を定義し，メンバを設定してください（**２.７.２　JPEGINFO構造体の構造**を参照）。新規に圧縮処理を行う際には，必ず最初に設定してください。継続の場合は，再設定する必要はありません。

表２－１　ユーザ設定メンバー覧

メンバ	設定するデータ	
Quality	量子化係数	
Sampling	サンプル比	
Restart	リスタート・インターバル	
Width	圧縮画像の横サイズ	
Height	圧縮画像の縦サイズ	
StartX	圧縮画像の開始ｘ座標	
StartY	圧縮画像の開始ｙ座標	
VRAM_Bptr	VRAMの先頭アドレス	
VRAM_W_Pixel	VRAMの横方向のピクセル数	
VRAM_H_Pixel	VRAMの縦方向のピクセル数	
VRAM_Line_Byte	VRAMの１ライン・バイト数	
VRAM_Pixel_Byte	VRAMの１ピクセル・バイト数	
VRAM_Gap1_Byte	同一ピクセルのY/Cb成分またはR/G成分のVRAMアドレス差分バイト数	
VRAM_Gap2_Byte	同一ピクセルのY/Cr成分またはR/B成分のVRAMアドレス差分バイト数	
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス	
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス	
MCU_Buff_Bptr	MCUバッファの先頭アドレス	
DQT_Y_Bptr	輝度成分用量子化テーブル先頭アドレス	
DQT_C_Bptr	色差成分用量子化テーブル先頭アドレス	
DHT_DC_Y_Bptr	輝度DC成分用ハフマン・テーブル先頭アドレス[注1]	
DHT_DC_C_Bptr	色差DC成分用ハフマン・テーブル先頭アドレス[注1]	
DHT_AC_Y_Bptr	輝度AC成分用ハフマン・テーブル先頭アドレス[注1]	
DHT_AC_C_Bptr	色差AC成分用ハフマン・テーブル先頭アドレス[注1]	
APP_Info_Bptr	APPnマーカに対応する場合：APPINFO構造体変数の先頭アドレス APPnマーカに対応しない場合：０	
Work1_Bptr	ライブラリ・ワーク・エリア１用バッファ先頭アドレス	
Work2_Bptr	ライブラリ・ワーク・エリア２用バッファ先頭アドレス	
Mode	ライブラリ動作モード	JPEGMODE_NORMAL（通常圧縮モード） １回の関数コールで１画面すべて圧縮します。１画面分の画像が格納できるだけのVRAMが必要です。
		JPEGMODE_LINE（ライン圧縮モード） １回の関数コールで1MCUラインのみ圧縮します。1MCUライン分の画像が格納できるだけのVRAMが必要です。複数回，関数コールすることで１画面分の圧縮を実現します。
DNL[注2]	DNLマーカを使用する場合　：JPEGDNL_ON DNLマーカを使用しない場合：JPEGDNL_OFF	
CI1, CI2, CI3	コンポーネントID[注3]	

★ **注１．ユーザ指定のハフマン・テーブルを使用するときは，２.３.２（１）ハフマン・テーブルの制限事項**を参照してください。

★ ２．Heightを０とした場合には，JPEGDNL_ONとしてください。JPEGDNL_OFFとした場合には異常終了します。

★ ３．CI1, CI2, CI3にはそれぞれ異なる値を設定してください。同じ値を設定した場合は異常終了します。

② **圧縮初期化関数をコール［jpeg_CompressInit］**

「①圧縮パラメータの設定」で設定した構造体のポインタを引き数に圧縮処理初期化関数をコールしてください。

この関数は，圧縮処理に必要なデータを初期化し，圧縮ライブラリを実行可能な状態にします。この関数は，新規に圧縮を行う際には，必ず最初に一度コールしてください。継続の場合は，この関数をコールする必要はありません。

③ **圧縮処理関数をコール［jpeg_Compress］**

「②圧縮初期化関数をコール」で使用した構造体のポインタを引き数に圧縮処理関数をコールしてください。

④ **返り値のチェック［JPEGRET_OK/JPEGRET_ERR/JPEGRET_CONT1/JPEGRET_CONT2］**

jpeg_Compressの返り値は次のとおりです。

表２－２　jpeg_Compressの返り値

返り値	意　味	説　明
JPEGRET_OK	正常終了	正常に終了しました。 新規に圧縮を行う場合は，再度**「①圧縮パラメータの設定」**から始めてください。
JPEGRET_ERR	異常終了	エラーを検出したため，処理を終了しました。 エラーの詳細は，JPEGINFO構造体のメンバ“ ErrorState ”に格納されています。 新規に圧縮を行う場合は，再度**「①圧縮パラメータの設定」**から始めてください。
JPEGRET_CONT1	継続１	指定されたJPEGファイル格納用バッファが一杯になったため，処理を中断しました。強制終了し新規に圧縮を行う場合は，再度**「①圧縮パラメータの設定」**から始めてください。 継続する場合は，**「⑤バッファ操作」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。
JPEGRET_CONT2	継続２	ライン圧縮モードを選択した場合にのみ発生します。 1MCUラインの圧縮が終了したため，処理を中断しました。 強制終了し新規に圧縮を行う場合は，再度**「①圧縮パラメータの設定」**から始めてください。 継続する場合は，**「⑧VRAMの容量チェック」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。 ライン圧縮モードを終了する場合は，**「⑫圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

表２－３　圧縮結果情報メンバ一覧

メンバ	格納されるデータ
ErrorState	エラー・ステータス
FileSize	JPEGファイル・サイズ

⑤　JPEGバッファ操作（退避／切り替え）

JPEGファイル格納用バッファの中身の退避／切り替えを選択します。

JPEGバッファの中身を退避し，同じバッファを使用し継続する場合は，再度**「③圧縮処理関数をコール」**から始めてください。

JPEGバッファを切り替える場合は，**「⑥圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

⑥　圧縮パラメータの設定

表２－４　圧縮パラメータ・メンバー覧（１）

メンバ	設定するデータ
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス

⑦　強制終了／継続

圧縮の途中ですが，圧縮データを掃き出し強制的にEOIを付加し終了させたい場合は，**「⑫圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

継続する場合は，**「⑧VRAMの容量チェック」**，**「⑨VRAMデータ更新／切り替え」**を選択し，**「⑩圧縮パラメータの設定」**，**「⑪圧縮パラメータの設定」**を実行したあと，再度**「③圧縮処理関数をコール」**から始めてください。

⑧　VRAMの容量チェック

VRAMが空の場合（画像データがない場合）は，**「⑨VRAMデータ更新／切り替え」**を選択し，**「⑩圧縮パラメータの設定」**，**「⑪圧縮パラメータの設定」**を実行したあと，再度**「③圧縮処理関数をコール」**から始めてください。

VRAMが空でない場合は，**「⑪圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

⑨　VRAMデータ更新／切り替え

VRAMデータの更新／切り替えを選択します。

切り替えの場合は，**「⑩圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

更新の場合は，**「⑪圧縮パラメータの設定」**を実行し，再度**「③圧縮処理関数をコール」**から始めてください。

⑩　圧縮パラメータの設定

表２－５　圧縮パラメータ・メンバ一覧（２）

メンバ	設定するデータ
StartX	圧縮画像の開始x座標
StartY	圧縮画像の開始y座標
VRAM_Bptr	VRAMの先頭アドレス
VRAM_W_Pixel	VRAMの横方向ピクセル数
VRAM_H_Pixel	VRAMの縦方向ピクセル数
VRAM_Line_Byte	VRAMの１ライン・バイト数
VRAM_Pixel_Byte	VRAMの１ピクセル・バイト数
VRAM_Gap1_Byte	同一ピクセルのY/Cb成分または，R/G成分のVRAMアドレス差分バイト数
VRAM_Gap2_Byte	同一ピクセルのY/Cr成分または，R/B成分のVRAMアドレス差分バイト数

⑪　圧縮パラメータの設定

表２－６　圧縮パラメータ・メンバ一覧（３）

メンバ	設定するデータ
StartX	圧縮画像の開始x座標
StartY	圧縮画像の開始y座標

⑫　圧縮パラメータの設定

表２－７　圧縮パラメータ・メンバ一覧（４）

メンバ	設定するデータ
Mode	強制終了モード（JPEGMODE_STOP）

２.２.２　伸長処理

伸長処理では，JPEGファイルを伸長し，画面に表示します。

（１）関数

分類	関数名	機能概要
伸長処理	jpeg_DecompressInit	JPEG伸長ライブラリ初期化
	jpeg_Decompress	JPEG伸長処理メイン
	jpeg_putMCU_M[注]	VRAMアクセス関数（モノクロ伸長）
	jpeg_putMCU_11[注]	VRAMアクセス関数（カラー伸長サンプル比4：4：4［H：V＝1：1］）
	jpeg_putMCU_21[注]	VRAMアクセス関数（カラー伸長サンプル比4：2：2［H：V＝2：1］）
	jpeg_putMCU_22[注]	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝2：2］）
	jpeg_putMCU_41[注]	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝4：1］）

注　カスタマイズする場合のみ参照してください。

（ａ）jpeg_DecompressInit

分　　類　伸長処理

関 数 名　jpeg_DecompressInit

機能概要　JPEG伸長ライブラリの初期化

形　　式　#include “ jpeg.h ”

void jpeg_DecompressInit（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**２.７.２　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG伸長ライブラリを初期化し，伸長処理を実行可能な状態にします。

（ｂ）jpeg_Decompress

分　　類　伸長処理

関 数 名　jpeg_Decompress

機能概要　JPEG伸長メイン

形　　式　#include “ jpeg.h ”

long jpeg_Decompress（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ更新要求）
JPEGRET_CONT2	継続２（画像データ退避要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます。）

★　　機　　能　*info*で指定されるJPEG伸長を行います。

異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_VRAM	伸長画像の領域指定が不正です。
JPEGERR_USF	伸長できないサンプル比です。
JPEGEER_QPQ	DQTヘッダのPqの値が０以外の値に設定されています。
JPEGERR_QTQ	DQTヘッダの量子化テーブル番号（Tq）が０，１以外の値です。
JPEGERR_DHT	DHTヘッダのTc, Thの値が不正です。
JPEGERR_SNS	SOSヘッダのNs値（コンポーネント数）が不正です（１，３以外）。
JPEGERR_SCD	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。
JPEGERR_SSS	SOSヘッダのSsの値が０ではありません。
JPEGERR_SSE	SOSヘッダのSeの値が63ではありません。
JPEGERR_SAA	SOSヘッダのAh, Alの値が０ではありません。
JPEGERR_SFP	SOFヘッダのPに８以外の値が設定されています。
JPEGERR_SFN	SOFヘッダのNfの値が不正です（１，３以外）。
JPEGERR_UKM	未知のマーカが現れました。
JPEGERR_OTH	圧縮データ中に不正なマーカが現れました。
JPEGERR_RST	RSTmマーカが不正です。
JPEGERR_SOS	そのほかのSOSヘッダ・エラーです（コンポーネントIDが不正）。
JPEGERR_SOF	SOFヘッダが定義されていないか，２つ以上定義されています。
JPEGERR_FTQ	必要な量子化テーブルが定義されていません。
JPEGERR_STH	必要なハフマン・テーブルが定義されていません。
JPEGERR_IMY	画像の縦サイズが不正です。
JPEGERR_FID	SOFヘッダのコンポーネントIDが重複しています。
JPEGERR_SID	SOSヘッダのコンポーネントIDが重複しています。
JPEGERR_QEL	量子化テーブルの要素に０が含まれています。
JPEGERR_VIJ	ハフマン・テーブルの要素が重複しています。
JPEGERR_JPB	JPEGバッファ・サイズが不正です。
JPEGERR_HUF	圧縮データが不正です。

（２）メモリ資源

伸長ライブラリを使用する場合に最低限必要なメモリ（RAM）容量は，次のとおりです。

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③ワーク２	8192	
④MCUバッファ	128-768	128：対応サンプル比　mono 384：対応サンプル比　mono, 4：4：4 512：対応サンプル比　mono, 4：4：4, 4：2：2 768：対応サンプル比　mono, 4：4：4, 4：2：2, 4：1：1 **２.５　MCUバッファの構造**参照。
⑤スタック	128	
⑥JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑦画像メモリ	（システム依存）	画像イメージ格納メモリ（たとえば，VRAM）。**２.４　VRAMの構成**参照。 最低，1MCUライン分は，必要。
小計	9492[注]	

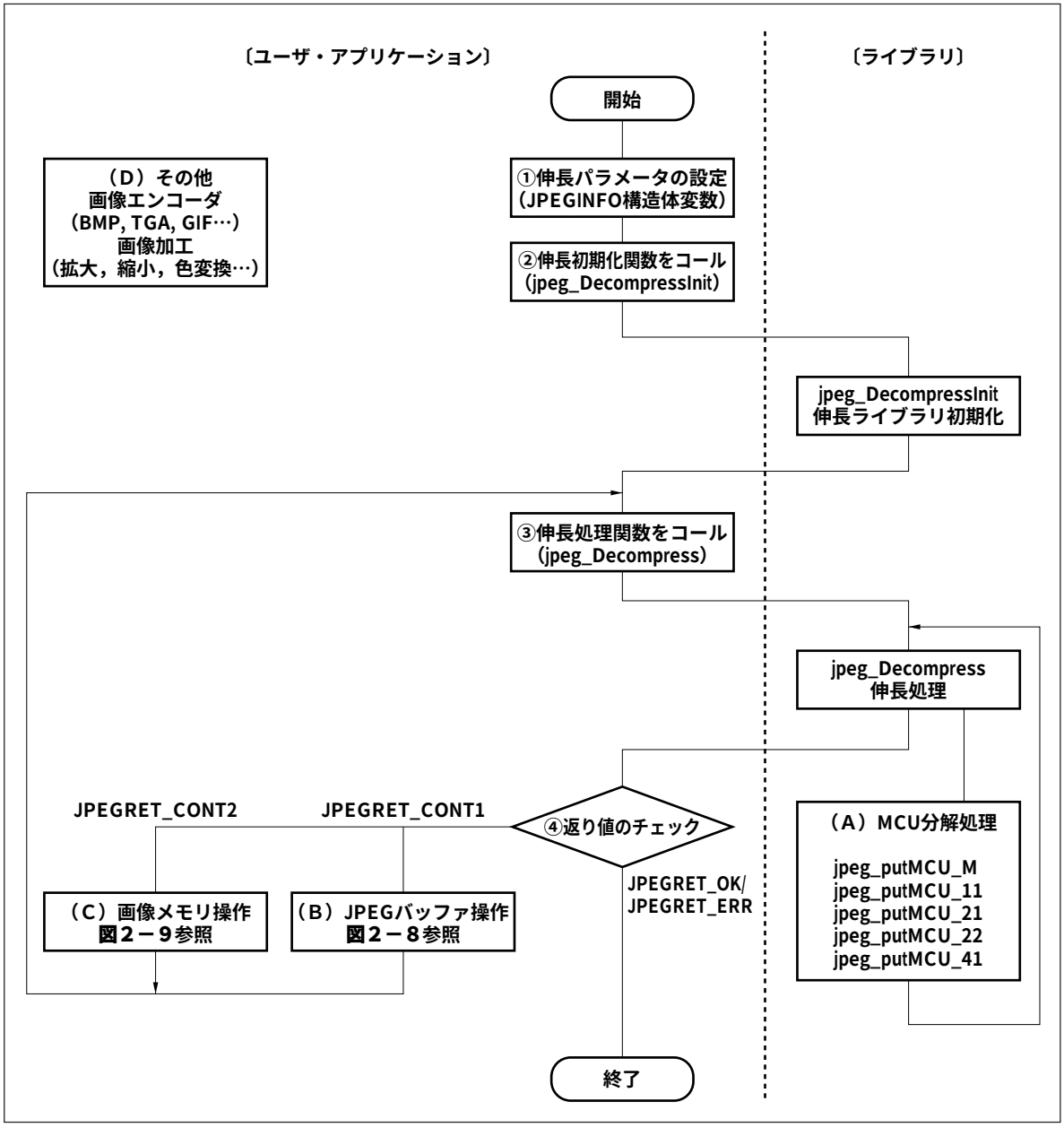
注　①～⑤を足した値です（MCUバッファは最大時）。

注意　バイト／ハーフワード／ワード単位でランダム・アクセスできるメモリが必要です。

（３）伸長処理フロー

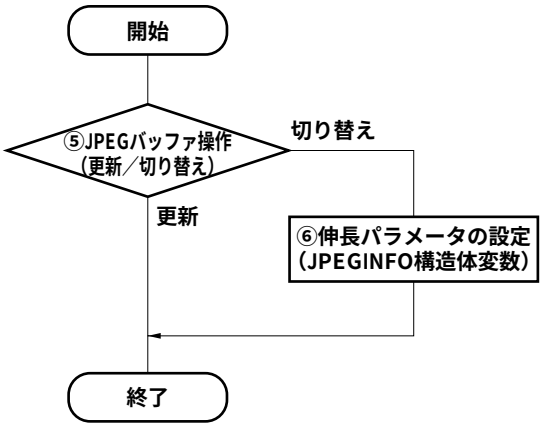
伸長ライブラリを使用したアプリケーションの処理の流れを図２－７から図２－９に示します。

図２－７　伸長処理の流れ



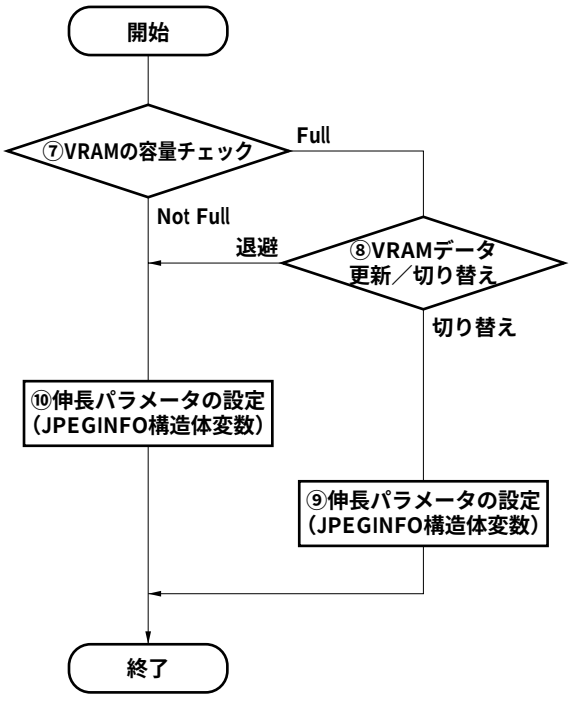
伸長ライブラリを使用する場合にユーザ・アプリケーション部分の処理として最低限必要な処理は，図２－７の①～④です。必要に応じて（A），（B），（C），（D）の処理も必要となる場合があります。

図２－８　JPEGバッファ操作



備考　１枚分のJPEGデータを格納するメモリが用意できず，何回かに分けて伸長する場合に必要な処理です。

図２－９　画像メモリ操作



備考　ライン単位伸長機能を使用する場合に必要な処理です。伸長画像を外部記憶媒体や通信で転送する場合の処理例です。

① 伸長パラメータの設定

ヘッダ・ファイル“jpeg.h”で定義されているJPEGINFO構造体の変数を定義し，メンバを設定してください（**2.7.2　JPEGINFO構造体の構造**を参照）。新規に伸長処理を行う際には，必ず最初に設定してください。継続の場合は，再設定する必要はありません。

表２－８　ユーザ設定メンバ一覧

メンバ	設定するデータ	
StartX	伸長画像の開始ｘ座標	
StartY	伸長画像の開始ｙ座標	
VRAM_Bptr	VRAMの先頭アドレス	
VRAM_W_Pixel	VRAMの横方向のピクセル数	
VRAM_H_Pixel	VRAMの縦方向のピクセル数	
VRAM_Line_Byte	VRAMの１ライン・バイト数	
VRAM_Pixel_Byte	VRAMの１ピクセル・バイト数	
VRAM_Gap1_Byte	同一ピクセルのY/Cb成分またはR/G成分のVRAMアドレス差分バイト数	
VRAM_Gap2_Byte	同一ピクセルのY/Cr成分またはR/B成分のVRAMアドレス差分バイト数	
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス	
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス	
MCU_Buff_Bptr	MCUバッファの先頭アドレス	
APP_Info_Bptr	0	
Work1_Bptr	ライブラリ・ワーク・エリア１用バッファ先頭アドレス	
Work2_Bptr	ライブラリ・ワーク・エリア２用バッファ先頭アドレス	
Mode	ライブラリ動作モード	JPEGMODE_NORMAL（通常伸長モード） １回の関数コールで１画面すべて伸長します。１画面分の画像が格納できるだけのVRAMが必要です。
		JPEGMODE_MCULINE（ライン伸長モード） １回の関数コールで1MCUラインのみ伸長します。1MCUライン分の画像が格納できるだけのVRAMが必要です。複数回，関数コールすることで１画面分の伸長を実現します。

② 伸長初期化関数をコール［jpeg_DecompressInit］

「①伸長パラメータの設定」で設定した構造体のポインタを引き数に伸長処理初期化関数をコールしてください。

この関数は，伸長処理に必要なデータを初期化し，伸長ライブラリを実行可能な状態にします。この関数は，新規に伸長を行う際には，必ず最初に一度コールしてください。継続の場合は，この関数をコールする必要はありません。

③ 伸長処理関数をコール［jpeg_Decompress］

「②伸長初期化関数をコール」で使用した構造体のポインタを引き数に伸長処理関数をコールしてください。

④　返り値のチェック［JPEGRET_OK/JPEGRET_ERR/JPEGRET_CONT1/JPEGRET_CONT2］

jpeg_Decompressの返り値は次のとおりです。

表２－９　jpeg_Decompressの返り値

返り値	意　味	説　明
JPEGRET_OK	正常終了	正常に終了しました。 新規に伸長を行う場合は，再度**「①伸長パラメータの設定」**から始めてください。
JPEGRET_ERR	異常終了	エラーを検出したため，処理を終了しました。 エラーの詳細は，JPEGINFO構造体のメンバ“ ErrorState ”に格納されています。 新規に伸長を行う場合は，再度**「①伸長パラメータの設定」**から始めてください。
JPEGRET_CONT1	継続１	指定されたJPEGファイル格納用バッファ内の伸長を終了したが，JPEGファイル終了コードがまだ発見できないため，処理を中断しました。 強制終了し新規に伸長を行う場合は，再度**「①伸長パラメータの設定」**から始めてください。 継続する場合は，**「⑤JPEGバッファ操作」**を実行し，再度**「③伸長処理関数をコール」**から始めてください。
JPEGRET_CONT2	継続２	ライン伸長モードを選択した場合にのみ発生します。 1MCUラインの伸長が終了したため，処理を中断しました。 強制終了し新規に伸長を行う場合は，再度**「①伸長パラメータの設定」**から始めてください。 継続する場合は，**「⑦VRAMの容量チェック」**を実行し，再度**「③伸長処理関数をコール」**から始めてください。

★ 表２－10　伸長結果情報メンバー覧

メンバ	格納されるデータ
ErrorState	エラー・ステータス
FileSize	JPEGファイル・サイズ
Sampling	サンプル比
Restart	リスタート・インターバル
Width	画像の横サイズ
Height	画像の縦サイズ
CI1	コンポーネントID1
CI2	コンポーネントID2
CI3	コンポーネントID3

⑤　JPEGバッファ操作（更新／切り替え）

JPEGファイル格納用バッファの中身の更新／切り替えを選択します。

JPEGバッファの中身を更新し，同じバッファを使用し継続する場合は，再度「③伸長処理関数をコール」から始めてください。

JPEGバッファを切り替える場合は，「⑥伸長パラメータの設定」を実行し，再度「③伸長処理関数をコール」から始めてください。

⑥　伸長パラメータの設定

表２－11　伸長パラメータの設定（１）

メンバ	設定するデータ
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス

⑦　VRAMの容量チェック

VRAMに空きがない場合（画像データで満たされた場合）は，「⑧VRAMデータ退避／切り替え」を選択し，「⑨伸長パラメータの設定」／「⑩伸長パラメータの設定」を実行したあと，再度「③伸長処理関数をコール」から始めてください。

VRAMに空きがある場合は，「⑩伸長パラメータの設定」を実行し，再度「③伸長処理関数をコール」から始めてください。

⑧　VRAMデータ退避／切り替え

VRAMデータの退避／切り替えを選択します。

切り替えの場合は，「⑨伸長パラメータの設定」を実行し，再度「③伸長処理関数をコール」から始めてください。

更新の場合は，「⑩伸長パラメータの設定」を実行し，再度「③伸長処理関数をコール」から始めてください。

⑨　伸長パラメータの設定

表２－12　伸長パラメータの設定（２）

メンバ	設定するデータ
StartX	伸長画像の開始x座標
StartY	伸長画像の開始y座標
VRAM_Bptr	VRAMの先頭アドレス
VRAM_W_Pixel	VRAMの横方向ピクセル数
VRAM_H_Pixel	VRAMの縦方向ピクセル数
VRAM_Line_Byte	VRAMの１ライン・バイト数
VRAM_Pixel_Byte	VRAMの１ピクセル・バイト数
VRAM_Gap1_Byte	同一ピクセルのY/Cb成分または，R/G成分のVRAMアドレス差分バイト数
VRAM_Gap2_Byte	同一ピクセルのY/Cr成分または，R/B成分のVRAMアドレス差分バイト数

⑩　伸長パラメータの設定

表２－13　伸長パラメータの設定（３）

メンバ	設定するデータ
StartX	伸長画像の開始x座標
StartY	伸長画像の開始y座標

２.２.３　解析処理

解析処理では，JPEGファイルを解析し，次に示すデータを得ます。

① サンプル比

② リスタート・インターバル

③ 画像のサイズ（横／縦）

④ JPEGファイル・サイズ

⑤ アプリケーション・データ（APPデータ）

★ ⑥ コンポーネントID

（１）関数

関数名	機　能
jpeg_AnalysisInit	JPEG解析処理ライブラリ初期化
jpeg_Analysis	JPEG解析処理

（ａ）jpeg_AnalysisInit

分　　類　解析処理

関 数 名　jpeg_AnalysisInit

機能概要　JPEG解析ライブラリの初期化

形　　式　#include “ jpeg.h ”

void jpeg_AnalysisInit（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**２.７.２　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG解析ライブラリを初期化し，解析処理を実行可能な状態にします。

（ｂ）jpeg_Analysis

分　　類　解析処理

関 数 名　jpeg_Analysis

機能概要　JPEG解析メイン

形　　式　#include “ jpeg.h ”

long jpeg_Analysis（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ更新要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます。）

機　　能　*info*で指定されるJPEG解析を行います。

異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_SNS	SOSヘッダのNs値（コンポーネント数）が不正です（１，３以外）。
JPEGERR_SCD	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。
JPEGEER_SSS	SOSヘッダのSsの値が０ではありません。
JPEGERR_SSE	SOSヘッダのSeの値が63ではありません。
JPEGERR_SAA	SOSヘッダのAh, Alの値が０ではありません。
JPEGERR_SFP	SOFヘッダのPに８以外の値が設定されています。
JPEGERR_SFN	SOFヘッダのNfの値が不正です（１，３以外）。
JPEGERR_UKM	未知のマーカが現れました。
★ JPEGERR_JPB	JPEGバッファ・サイズが不正です。

（2）メモリ資源

解析ライブラリを使用する場合に最低限必要なメモリ（RAM）容量は，次のとおりです。

名　　称	サイズ	備　　考
①JPEGINFO構造体	148	
②ワーク１	256	内蔵RAMへのメモリ・マッピングを推奨。
③スタック	128	
④JPEGバッファ	（システム依存）	圧縮データ格納メモリ。
⑤APPINFO構造体	96	APPnセグメントを使用する場合のみ必要。
⑥APPnデータ	（システム依存）	APPnセグメントを使用する場合のみ必要。
小計	532[注]	

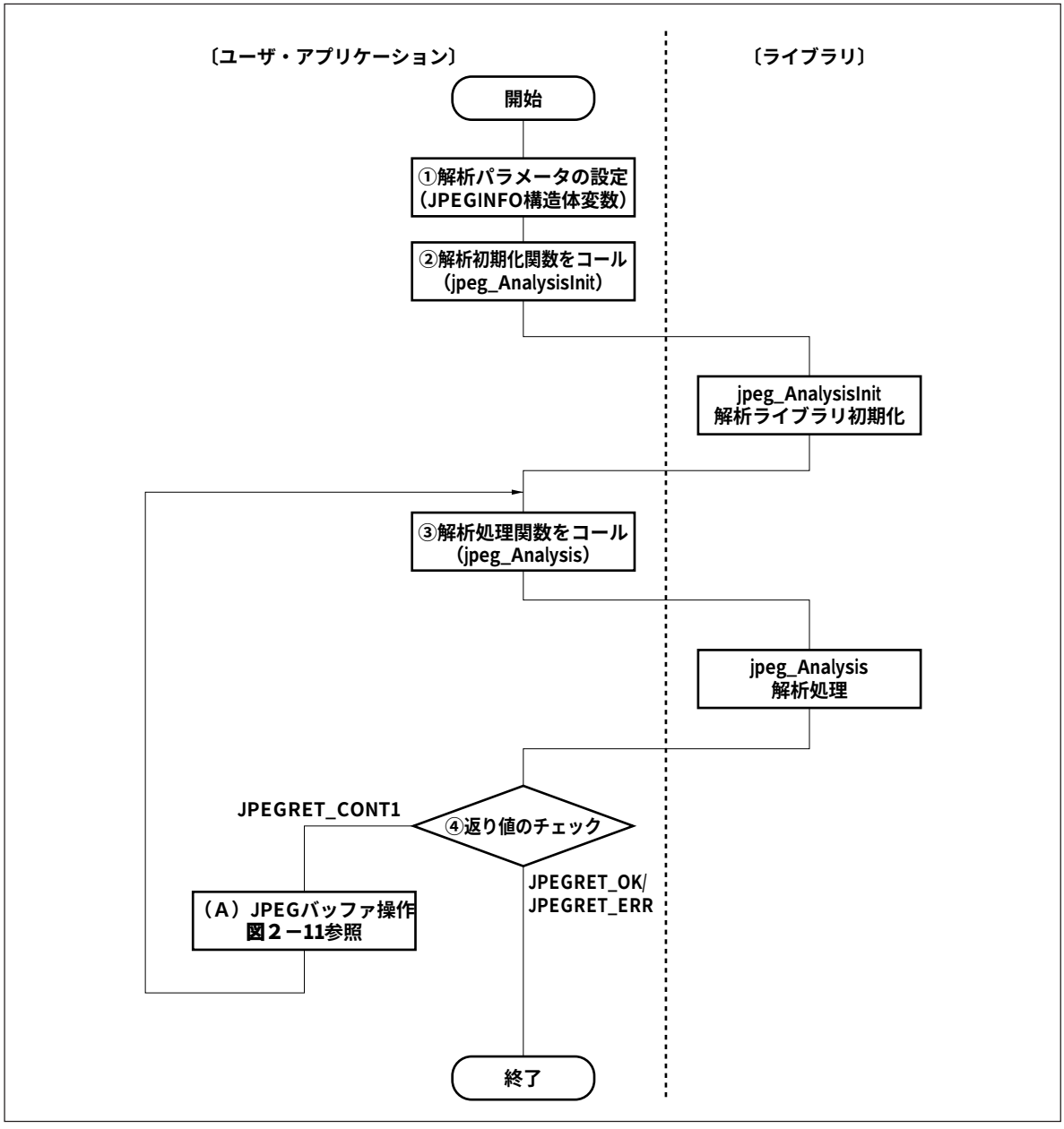
注　①～③を足した値です。

注意　バイト／ハーフワード／ワード単位でランダム・アクセスできるメモリが必要です。

（３）処理フロー

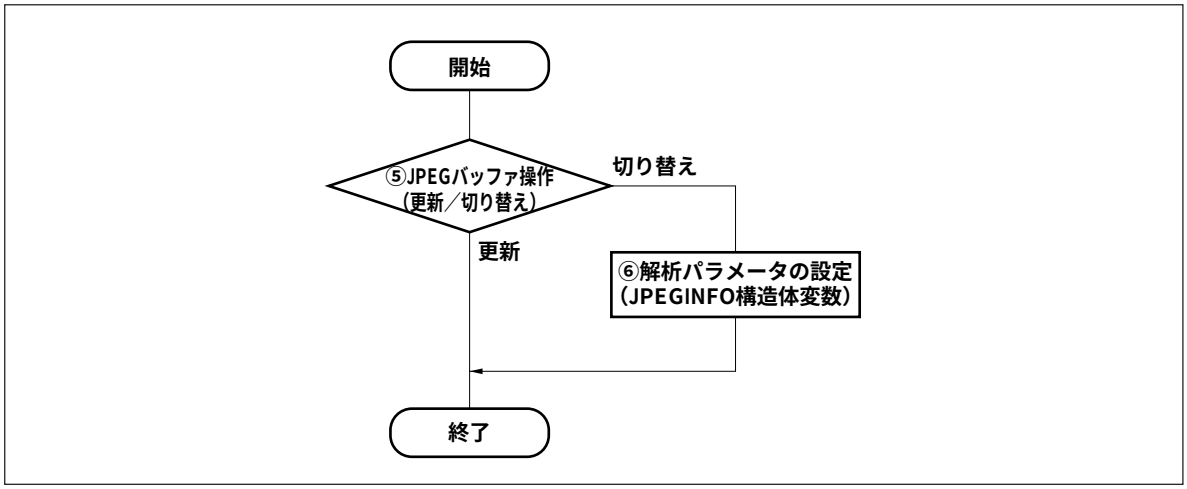
解析ライブラリを使用したアプリケーションの処理の流れを図２－10，図２－11に示します。

図２－10　解析処理の流れ



解析ライブラリを使用する場合にユーザ・アプリケーション部分の処理として最低限必要な処理は，図２－10の①～④です。必要に応じて（A）の処理も必要となる場合があります。

図２－11　JPEGバッファ操作



① 解析パラメータの設定

ヘッダ・ファイル“jpeg.h”で定義されているJPEGINFO構造体の変数を定義し，メンバを設定してください（２.７.２　JPEGINFO構造体の構造を参照）。新規に解析処理を行う際には，必ず最初に設定してください。継続の場合は，再設定する必要はありません。

表２－14　ユーザ設定メンバー覧

メンバ	設定するデータ
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス
APP_Info_Bptr	APPnマーカに対応する場合：APPINFO構造体変数の先頭アドレス APPnマーカに対応しない場合：0
Work1_Bptr	ライブラリ・ワーク・エリア用バッファ先頭アドレス

★ 注意　APP_Info_Bptrに０以外の値を設定した場合，解析初期化関数（jpeg_AnalysisInit）の中で構造体の初期化を行います。設定した値から96バイト分０クリアされますので，必要なデータは退避させてから関数をコールしてください。

② 解析初期化関数をコール［jpeg_AnalysisInit］

「①解析パラメータの設定」で設定した構造体のポインタを引き数に解析処理初期化関数をコールしてください。

この関数は，解析処理に必要なデータを初期化し，解析ライブラリを実行可能な状態にします。この関数は，新規に解析を行う際には，必ず最初に一度コールしてください。継続の場合は，この関数をコールする必要はありません。

③ 解析処理関数をコール［jpeg_Analysis］

「②解析初期化関数をコール」で使用した構造体のポインタを引き数に解析処理関数をコールしてください。

④　返り値のチェック［JPEGRET_OK/JPEGRET_ERR/JPEGRET_CONT1］

jpeg_Analysisの返り値は次のとおりです。

表２－15　jpeg_Analysisの返り値

返り値	意　味	説　明
JPEGRET_OK	正常終了	正常に終了しました。 新規に解析を行う場合は，再度**「①解析パラメータの設定」**から始めてください。
JPEGRET_ERR	異常終了	エラーを検出したため，処理を終了しました。 エラーの詳細は，JPEGINFO構造体のメンバ“ ErrorState ”に格納されています。 新規に解析を行う場合は，再度**「①解析パラメータの設定」**から始めてください。
JPEGRET_CONT1	継続１	指定されたJPEGファイル格納用バッファ内の解析を終了したが，JPEGファイル終了コードがまだ発見できないため，処理を中断しました。 強制終了し新規に解析を行う場合は，再度**「①解析パラメータの設定」**から始めてください。 継続する場合は，**⑤JPEGバッファ操作（更新／切り替え）**を実行し，再度**「③解析処理関数をコール」**から始めてください。

★

表２－16　解析結果情報メンバー覧

メンバ	格納されるデータ
ErrorState	エラー・ステータス
Sampling	サンプル比
Restart	リスタート・インターバル
Width	画像の横サイズ
Height	画像の縦サイズ
FileSize	JPEGファイル・サイズ
APP_Info_Bptr	初期パラメータ設定時に，このメンバにAPPINFO構造体変数のアドレスを設定した場合だけに，指定されたAPPINFO構造体変数のメンバにAPPnマーカのアドレスとデータ・サイズが格納されます。
CI1	コンポーネントID1
CI2	コンポーネントID2
CI3	コンポーネントID3

⑤　JPEGバッファ操作（更新／切り替え）

JPEGファイル格納用バッファの中身の更新／切り替えを選択します。

JPEGバッファの中身を更新し，同じバッファを使用し継続する場合は，再度「③解析処理関数をコール」から始めてください。

JPEGバッファを切り替える場合は，「⑥解析パラメータの設定」を実行し，再度「③解析処理関数をコール」から始めてください。

⑥　解析パラメータの設定

表２－17　解析パラメータの設定

メンバ	設定するデータ
JPEG_Buff_Bptr	JPEGファイル格納バッファの先頭アドレス
JPEG_Buff_Eptr	JPEGファイル格納バッファの末尾アドレス

２.３　JPEGフォーマット

２.３.１　対応フォーマット

AP703000-B03では，次のフォーマットに対応しています。

★ 表２－18　JPEG対応フォーマット

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	×
ロスレス方式（可逆方式）		×
ハイアラーキカル方式		×
サンプル比	カラー	4：4：4 [H：V＝1：1] 4：2：2 [H：V＝2：1] 4：1：1 [H：V＝2：2] 4：1：1 [H：V＝4：1]
	モノクロ	H：V＝1：1 H：V＝1：2 H：V＝1：3 H：V＝1：4 H：V＝2：1 H：V＝2：2 H：V＝2：3 H：V＝2：4 H：V＝3：1 H：V＝3：2 H：V＝3：3 H：V＝4：1 H：V＝4：2
コンポーネント		単色，３色
量子化テーブル		２面（圧縮），４面（伸長）
ハフマン・テーブル		４面
COM（コメント・マーカ）		○（伸長）
APPn（アプリケーション・マーカ）		○
DRI/RSTm（リスタート・マーカ）		○
DNL（ライン数再定義マーカ）		○

備考　○：対応している

×：対応していない

２.３.２　DHTマーカ

（１）ハフマン・テーブルの制限事項

AP703000-B03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良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，AP703000-B03の圧縮ルーチンは，そのような場合でも正常終了（返り値JPEGRET_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。

（ａ）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

（ｂ）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２－12　DHTセグメント

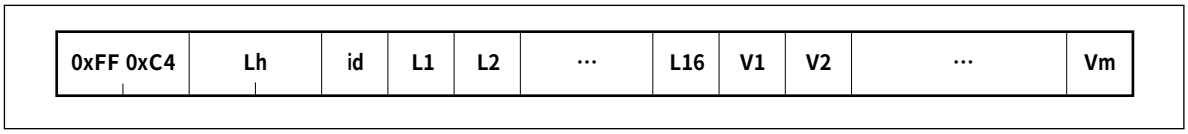


表２－19　DC/AC成分の値とビット長

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

（ｉ）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。

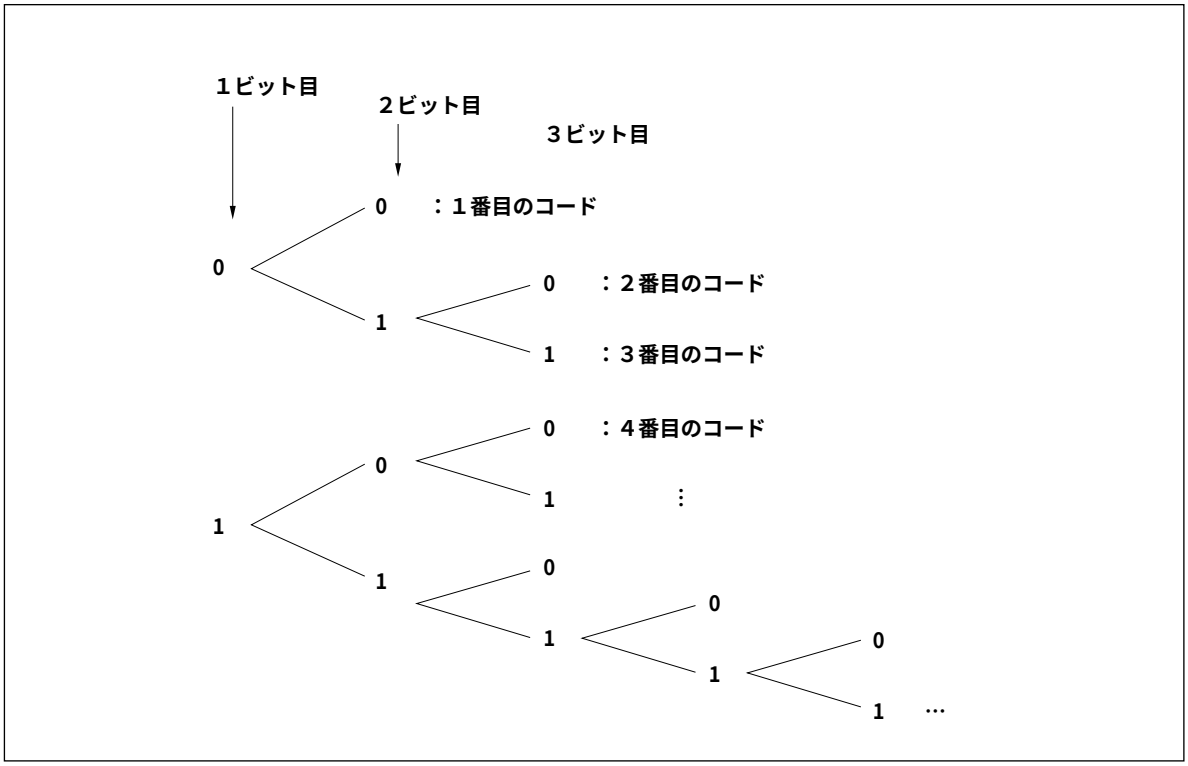
00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00

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，次の図のように，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２－13　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，AP703000-B03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。

（ⅱ）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に近づくにしたが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，AP703000-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，AP703000-B03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ビットがカテゴリを，上位４ビッ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，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，AP703000-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。

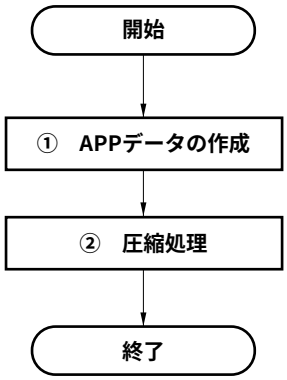
２.３.３　APPnマーカ

APPnマーカに対応した場合の各処理手順を次に示します。

（１）圧縮処理

APPnマーカに対応した場合の圧縮ライブラリを使用したアプリケーションの処理の流れを次に示します。

図２－14　APPnマーカを使用した場合の圧縮処理の流れ



①　APPデータの作成

APP用データを作成し，ヘッダ・ファイル“jpeg.h”で定義されているAPPINFO構造体の変数を定義し，メンバの設定を行ってください。

詳細は，**２.７.３　APPINFO構造体の構造**を参照してください。

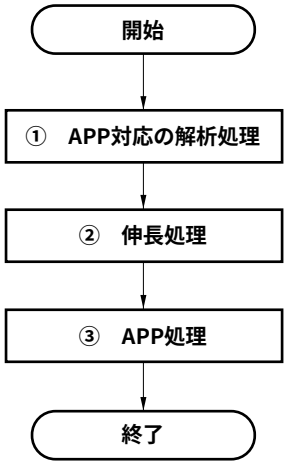
②　圧縮処理

詳細は，**２.２.１　圧縮処理**を参照してください。

（２）伸長処理

APPnマーカに対応した場合の伸長ライブラリを使用したアプリケーションの処理の流れを次に示します。

図２－15　APPnマーカを使用した場合の伸長処理の流れ



①　APP対応の解析処理

詳細は，**２.２.３　解析処理**を参照してください。

②　伸長処理

詳細は，**２.２.２　伸長処理**を参照してください。

③　APP処理

①でAPPINFO構造体変数に格納されたデータをもとに，ユーザがAPPnマーカに対応した処理を行ってください。

（３）解析処理

APPnマーカに対応した場合の解析ライブラリの使用方法は，**２.２.３　解析処理**を参照してください。

２.３.４　DNLマーカ

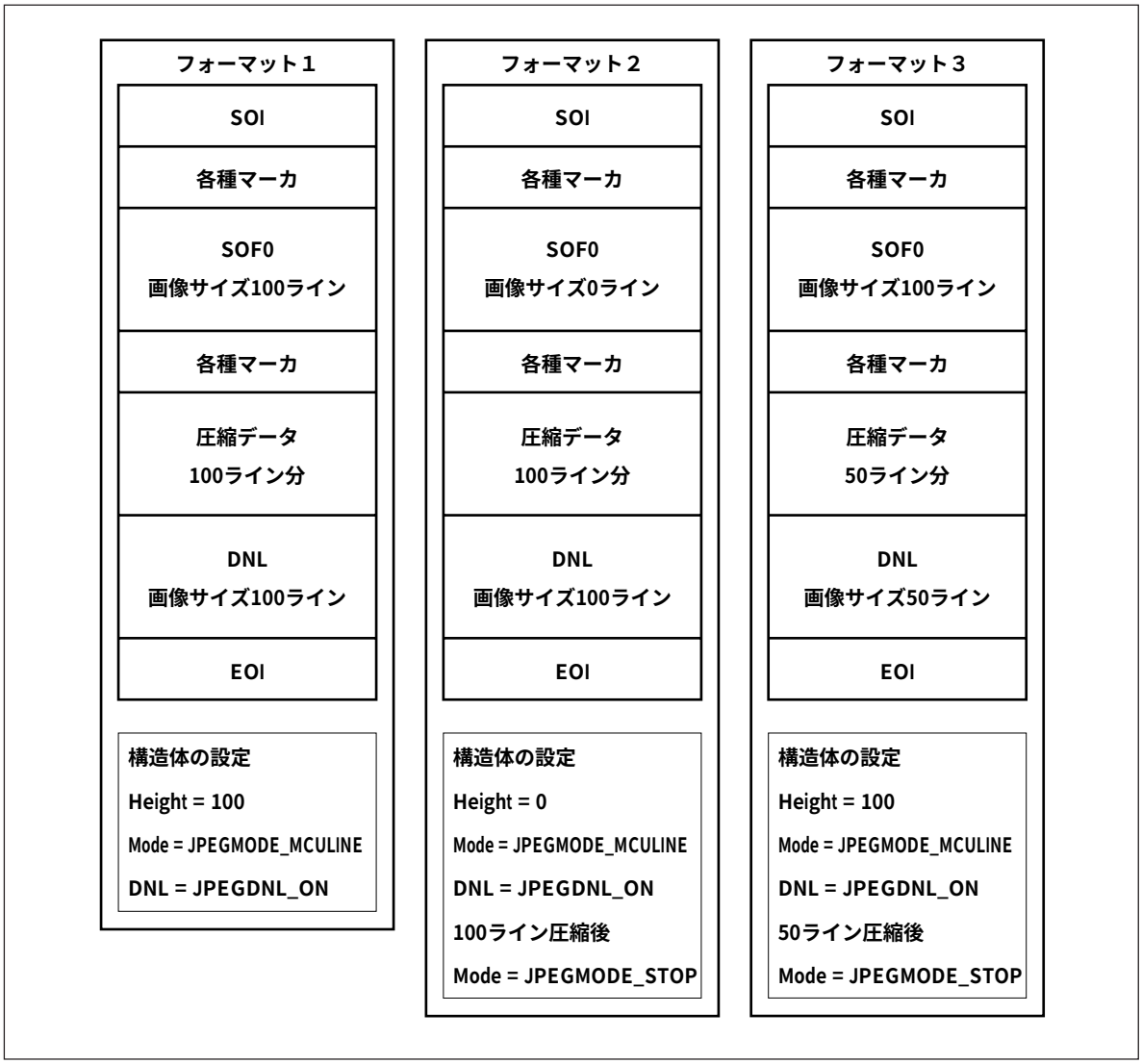
DNLマーカとは，画像のライン数（高さ）を再定義する場合に使用し，圧縮のデータの最後に位置します。

BaselineDCTでの扱いは，オプションとなっています。

通常，画像サイズ（ライン数）は，SOFnセグメントに定義されていますが，DNLセグメントが定義されている場合，DNLセグメントで定義されるライン数が優先されます。また，SOFnセグメントで画像サイズが０ラインと定義されている場合，DNLセグメントは必須となります。

一般的な使用方法，基本ライブラリで実現する場合のパラメータの設定方法は，次のとおりです。

図２－16　JPEGファイルの設定方法



★ ★

画像サイズ（ライン数）が圧縮開始段階で未知の場合は，フォーマット２のように構造体メンバ設定することで実現可能です。なお，SOFセグメントのライン数が０以外であり，圧縮データ・ライン数がSOFセグメントのライン数を越えるような場合には，正常に圧縮／伸長を行えません。

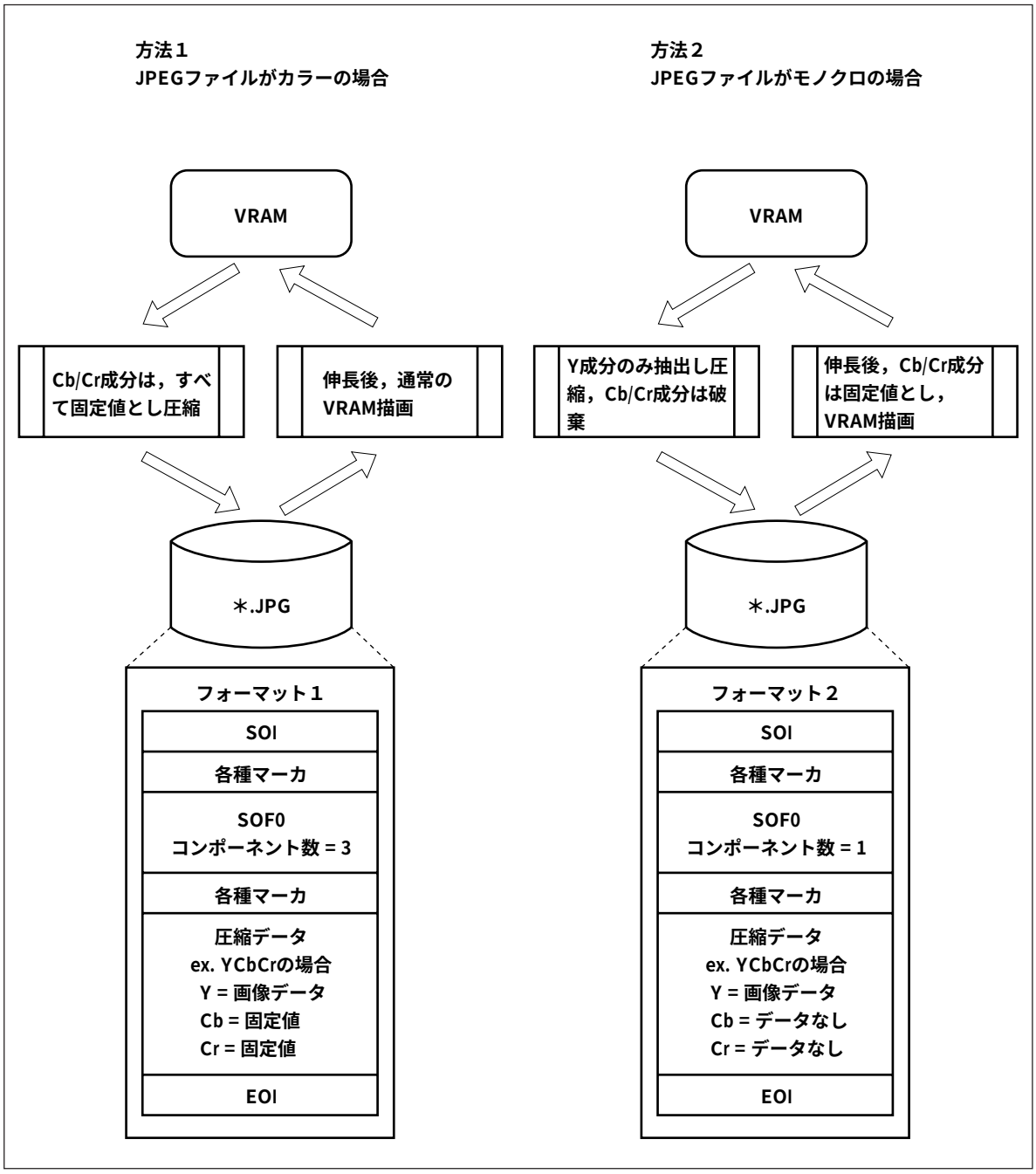
★

備考　DNLマーカは，ほかのオプション・マーカ（ex.DRI, RSTm）と比較し，対応しているツールが多くはありません。汎用性のあるJPEGファイルを作成したい場合は，DNLマーカの使用は，控えてください。

２.３.５　モノクロ・フォーマット対応

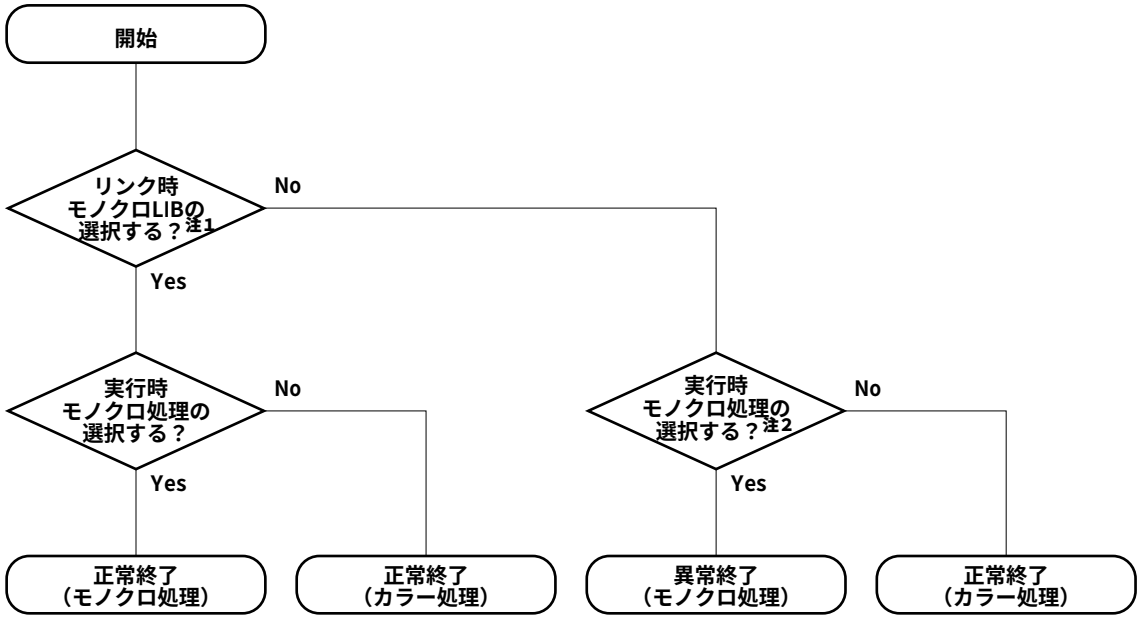
基本ライブラリは，モノクロ画像の圧縮／伸長手順として，次の２つの実現方法を提供します。

図２－17　モノクロ・フォーマット



なお，「方法２」でのライブラリの使用方法と実行結果は，次のとおりです。

図２－18　基本ライブラリの選択と実行結果



注１．圧縮　libjcsm.a

伸長　libjdsm.a

２．メンバ「Sampling」に指定する値により決定します。

・カラー

JPEGSAMPLE11（0x11）　JPEGSAMPLE21（0x21）　JPEGSAMPLE22（0x22）

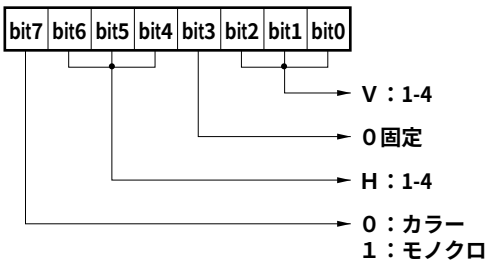
JPEGSAMPLE41（0x41）

・モノクロ

JPEGSAMPLEMONO（0x80）と下記値とのOR値

0x11　0x12　0x13　0x14　0x21　0x22　0x23　0x24　0x31　0x32　0x33　0x41　0x42

メンバ「Sampling」のビットの意味は，次のとおりです。



２.４　VRAMの構成

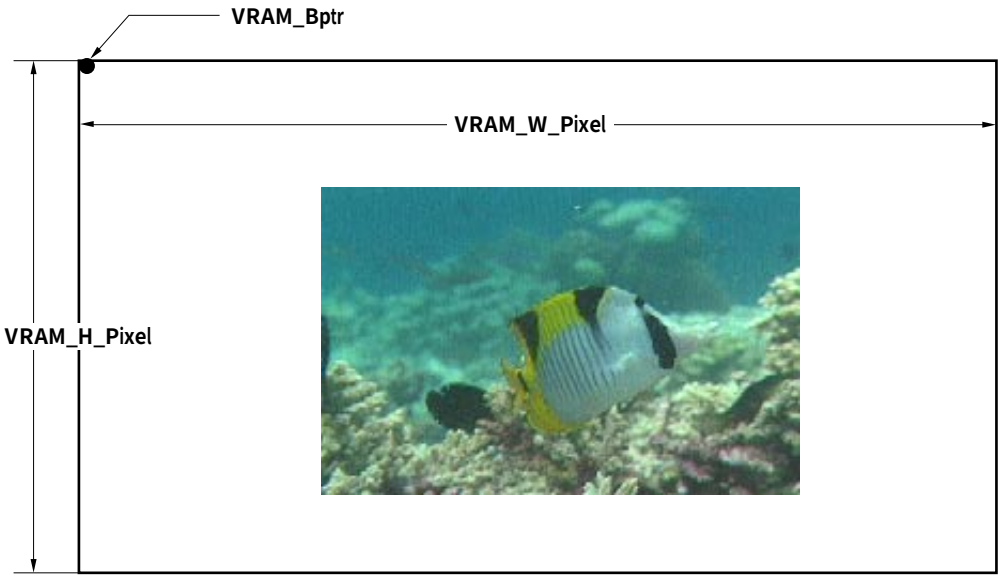
ディフォールトのVRAMアクセス・ルーチンでは，VRAMをY/Cb/CrあるいはR/G/Bのそれぞれが256階調（１バイト分）の深みを持つもので，LD.B命令／ST.B命令でアクセス可能なメモリとして想定しています。

次に示すメンバの値は，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。

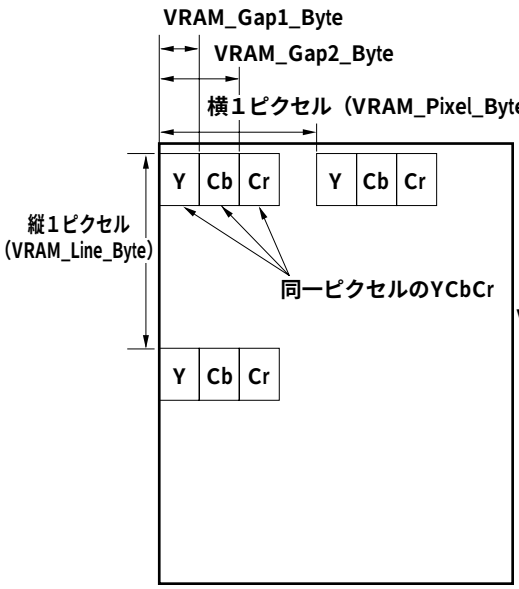
表２－20　VRAMの構成

メンバ	説　　明
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
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

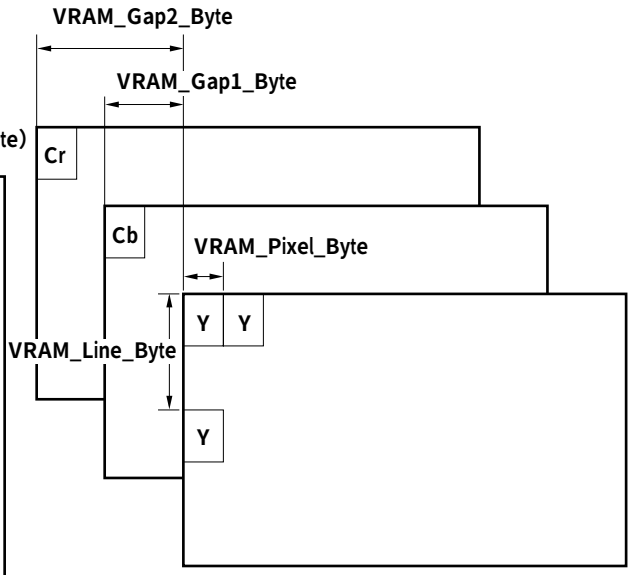
図２－19　VRAMの構成



例１



例２



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，次のメンバの値をチェックします。

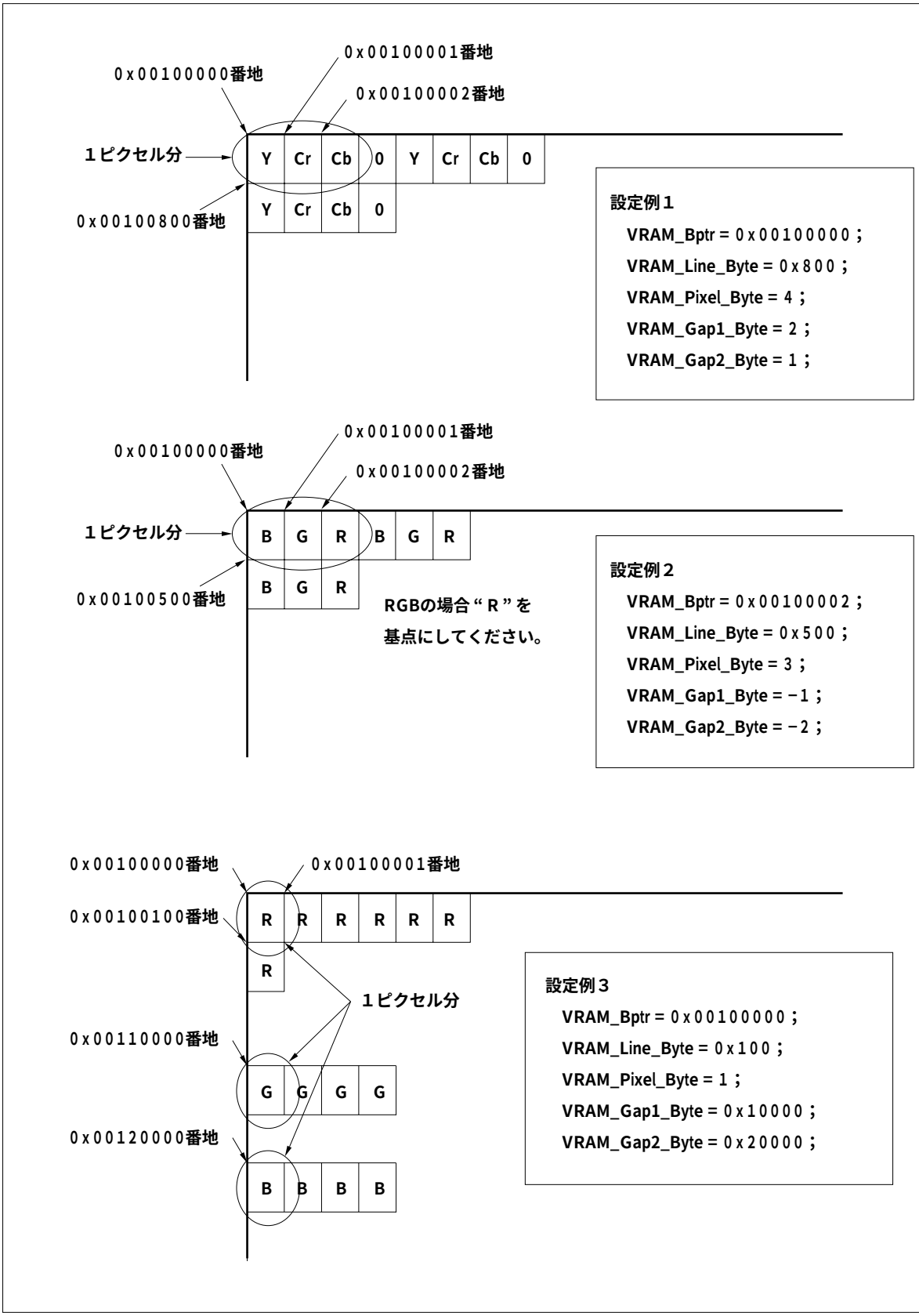
VRAM_W_PixelとStartX＋Widthの大小関係
VRAM_H_PixelとStartY＋Heightの大小関係

ＶＲＡＭアクセス部分をカスタマイズ（２．６　基本ライブラリのカスタマイズ参照）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，次のJPEGINFO構造体のメンバは設定許容値内であれば不定値で構いません。

- VRAM_Bptr
- VRAM_Line_Byte
- VRAM_Pixel_Byte
- VRAM_Gap1_Byte
- VRAM_Gap2_Byte

VRAM関連のメンバの設定例を次に示します。

★ **図２－20　基本ライブラリのVRAM関連メンバ設定例**



２.５　MCUバッファの構造

基本ライブラリ（圧縮処理／伸長処理）を使用する際には，ユーザがデータ格納用にMCUバッファを用意する必要があります。

MCUバッファは，圧縮時，MCU単位でサンプリングされたデータが格納されます。格納されたデータには，DCT処理が施されます。VRAMがRGB方式の場合は，YCbCr方式に変換してから格納します。また，伸長時，逆DCT処理の結果がMCU単位で出力されます。

次にMCUバッファの構成を示します。

表２－21　MCUバッファの構成

宣　　言	サイズ（バイト）	対応サンプル比
short *name*［1］［64］	128	モノクロ
short *name*［3］［64］	384	モノクロ，４：４：４［H：V＝１：１］
short *name*［4］［64］	512	モノクロ，４：４：４［H：V＝１：１］，４：２：２［H：V＝２：１］
short *name*［6］［64］	768	モノクロ，４：４：４［H：V＝１：１］，４：２：２［H：V＝２：１］，４：１：１［H：V＝２：２］，４：１：１［H：V＝４：１］

バッファは次の要領で構成してください。

（１）アプリケーション内でユーザが実体を定義し，JPEGINFO構造体変数のメンバへポインタを渡してください。JPEGINFO構造体については，**２.７.１　データ型詳細**を参照してください。

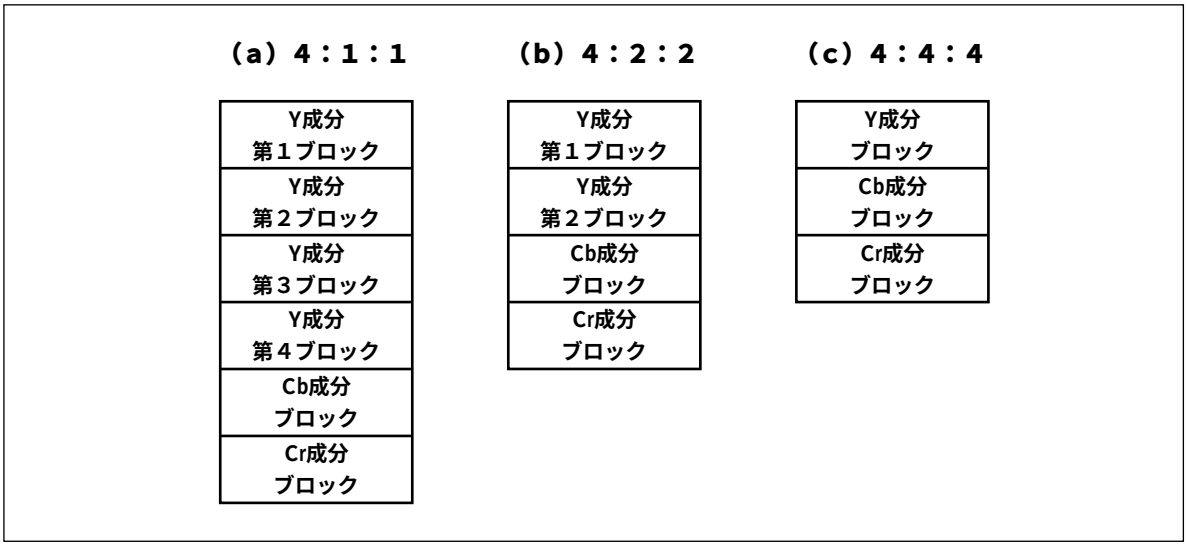
（２）データは８ビットですが，内部でワーク・バッファとしても扱いますので，必ずshort型（16ビット）で宣言してください。

（３）*name*は任意の名前を付けてください。

（４）バッファは，JPEGINFO構造体変数１つにつき必ず１つ用意してください。

なお，各MCUは次のようにバッファに格納してください。

図２－21　MCUバッファへの格納



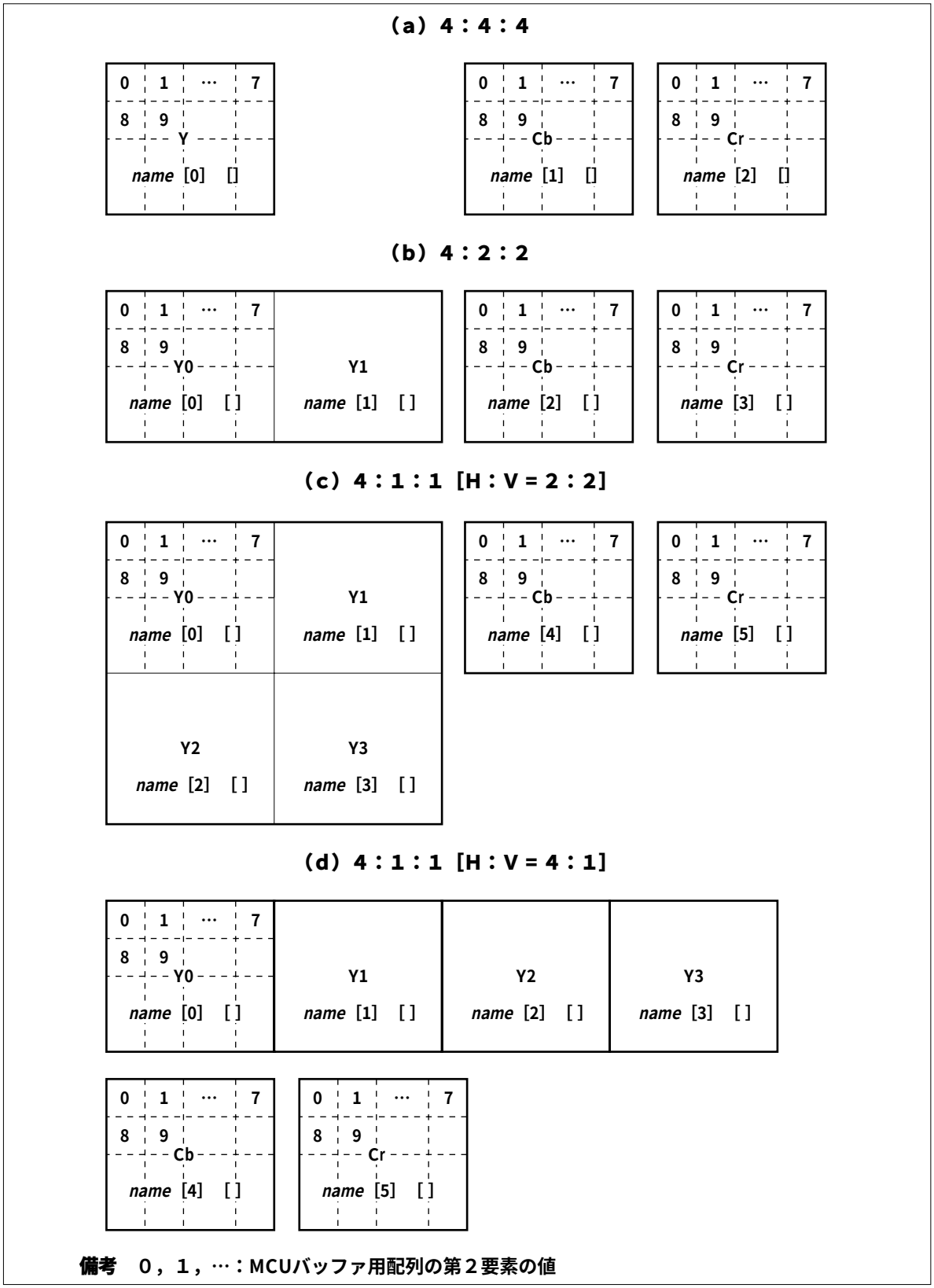
また，次に示すバッファを定義してください。

表２－22　バッファの定義

処　理	必要サイズ
圧縮	圧縮サンプル比に対応するもの。 **例**　対応ライブラリにかかわらず，サンプル比4：2：2で圧縮する場合，バッファとして*name*［4］［64］を定義します。
伸長	対応ライブラリの中でバッファ・サイズが最大のもの。 **例**　ライブラリをサンプル比4：4：4と4：2：2に対応させた場合，バッファとして*name*［4］［64］を定義します。
解析	－

次にバッファの内容を示します。

図２－22　MCUバッファの内容



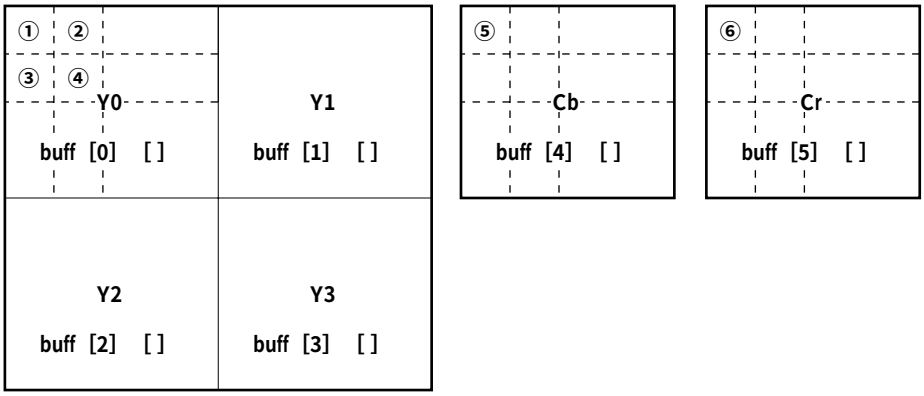
備考　０，１，…：MCUバッファ用配列の第２要素の値

サンプル比4：1：1でMCUバッファをbuff［6］［64］と定義した例を次に示します。

表２－23　1MCU分のデータの格納（4：1：1の場合）

配　列	格納データ
buff［0］［ ］	Y成分（左上８×８ピクセル分）
buff［1］［ ］	Y成分（右上８×８ピクセル分）
buff［2］［ ］	Y成分（左下８×８ピクセル分）
buff［3］［ ］	Y成分（右下８×８ピクセル分）
buff［4］［ ］	Cb成分（16×16ピクセル分：ただし１つの要素は２×２ピクセルの平均）
buff［5］［ ］	Cr成分（16×16ピクセル分：ただし１つの要素は２×２ピクセルの平均）

図２－23　左上２×２ピクセル分のデータの格納



成分	格納位置
Y成分	①buff［0］［0］，②buff［0］［1］，③buff［0］［8］，④buff［0］［9］
Cb成分	⑤buff［4］［0］
Cr成分	⑥buff［5］［0］

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）。

ここでは，VRAMアクセス部分の関数について，ディフォールトを使用しないで，ユーザが作成する場合の仕様について説明します。

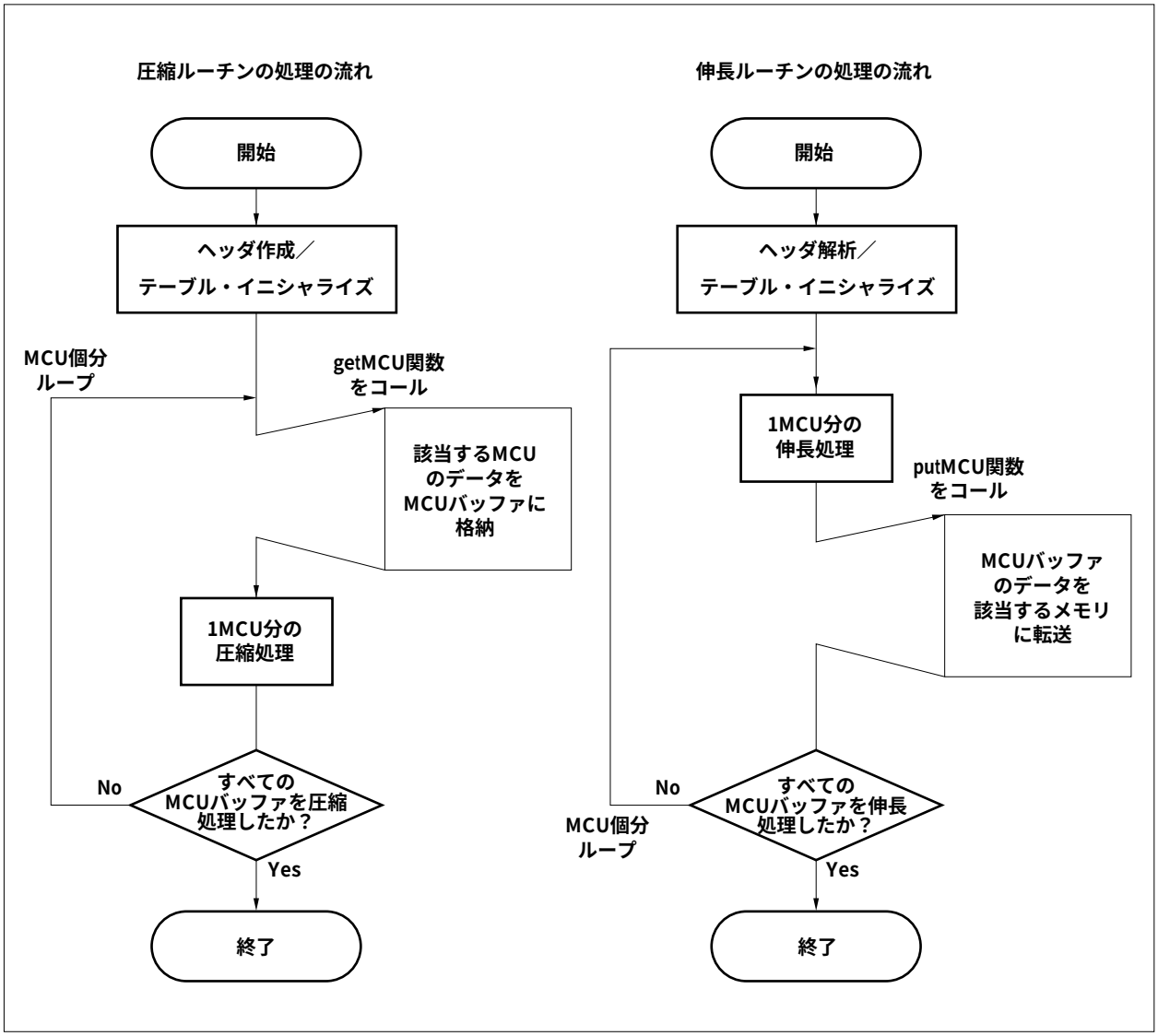
なおAP703000-B03には，VRAMアクセス関数をユーザが作成する際の参考用サンプル・ソースとしてvramacc0.s/vramacc1.cが含まれています。

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）。

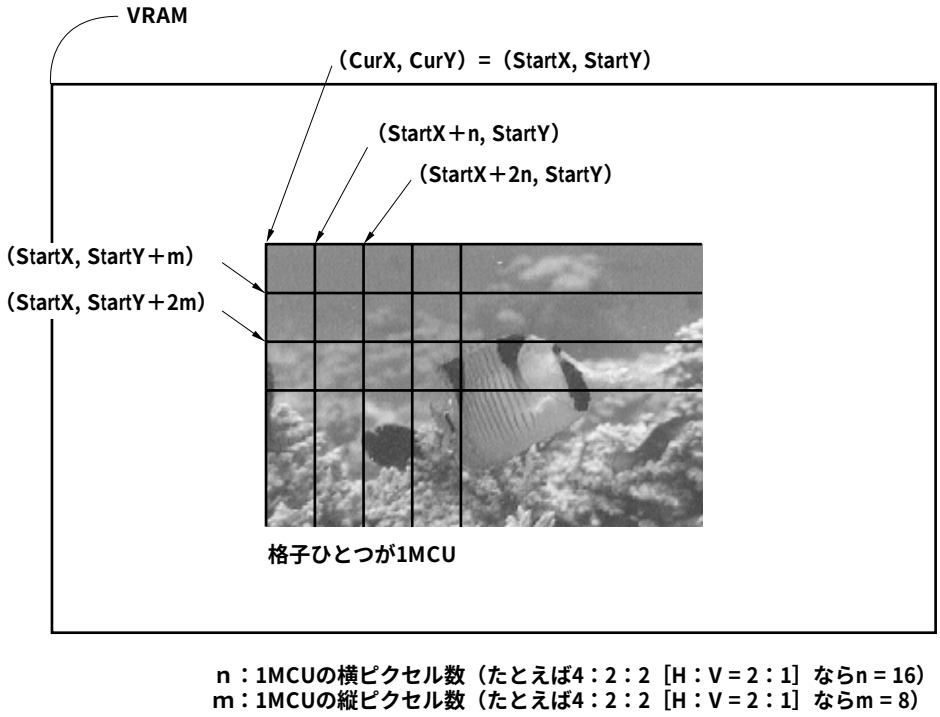
画像メモリへのアクセスは，getMCU関数，putMCU関数で行います。

図２－24　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。

図２－25　構造体のメンバCurrentX/CurrentY



n：1MCUの横ピクセル数（たとえば4：2：2［H：V = 2：1］ならn = 16）
m：1MCUの縦ピクセル数（たとえば4：2：2［H：V = 2：1］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，JPEGINFO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は基本ライブラリ側が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関数が基本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（Y/Cb/Crそれぞれ０-255）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２－24　MCUの単位

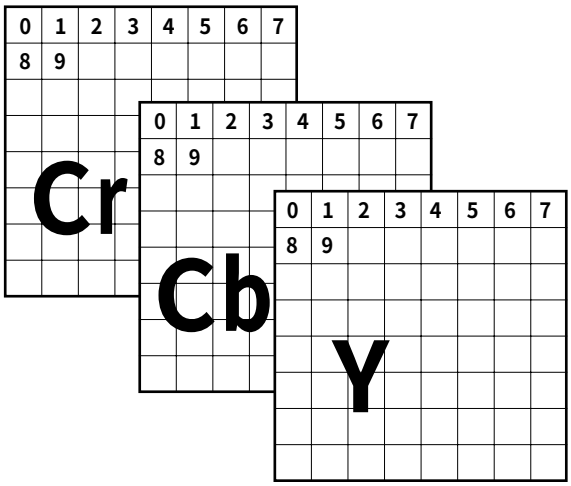
サンプル比	MCU単位
4：4：4［H：V = 1：1］	横８×縦８ピクセル
4：2：2［H：V = 2：1］	横16×縦８ピクセル
4：1：1［H：V = 2：2］	横16×縦16ピクセル
4：1：1［H：V = 4：1］	横32×縦８ピクセル

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

図２－26　1MCU分の画像データ（1/2）

（a）4：4：4［H：V＝1：1］の場合



（b）4：2：2［H：V＝2：1］の場合

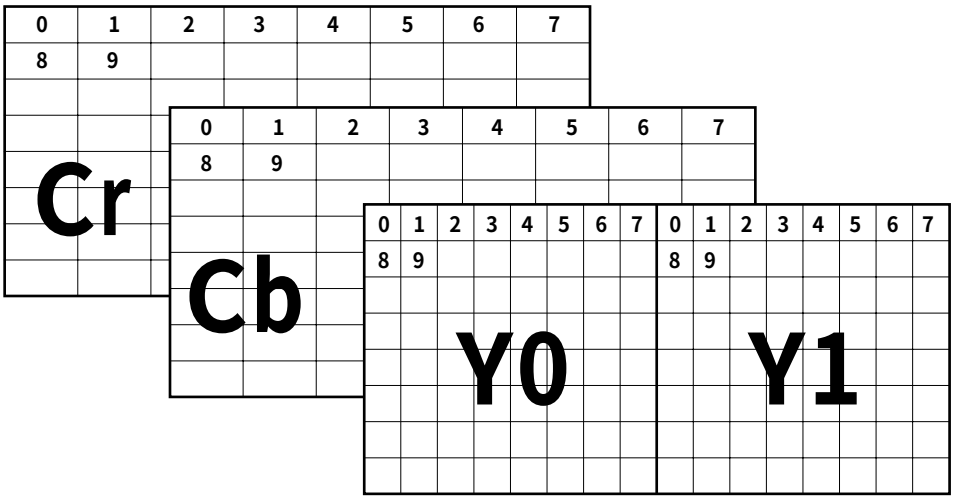
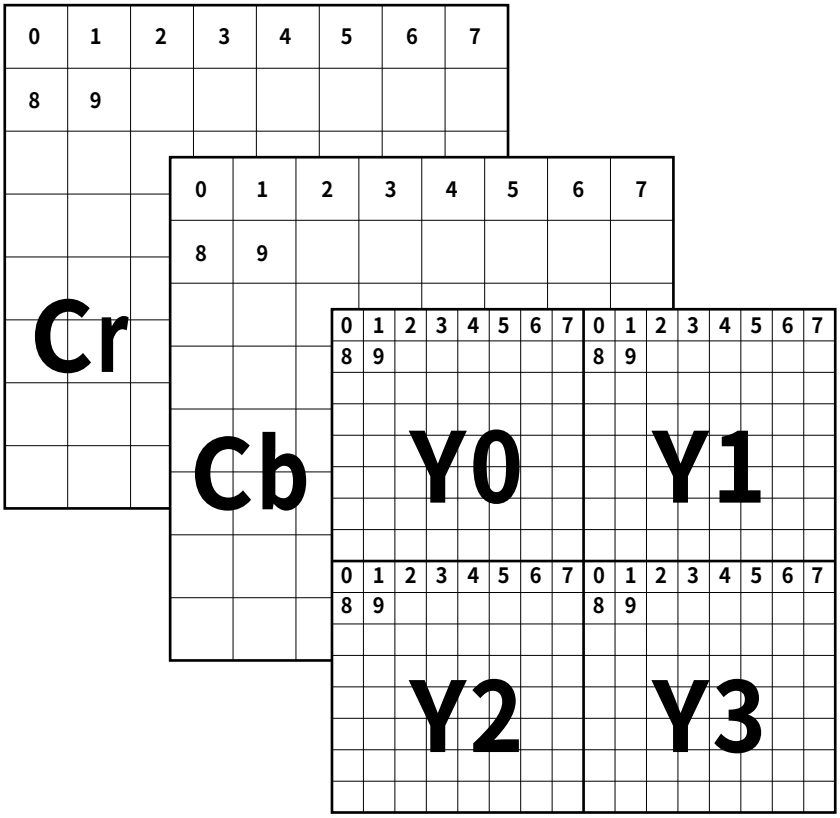
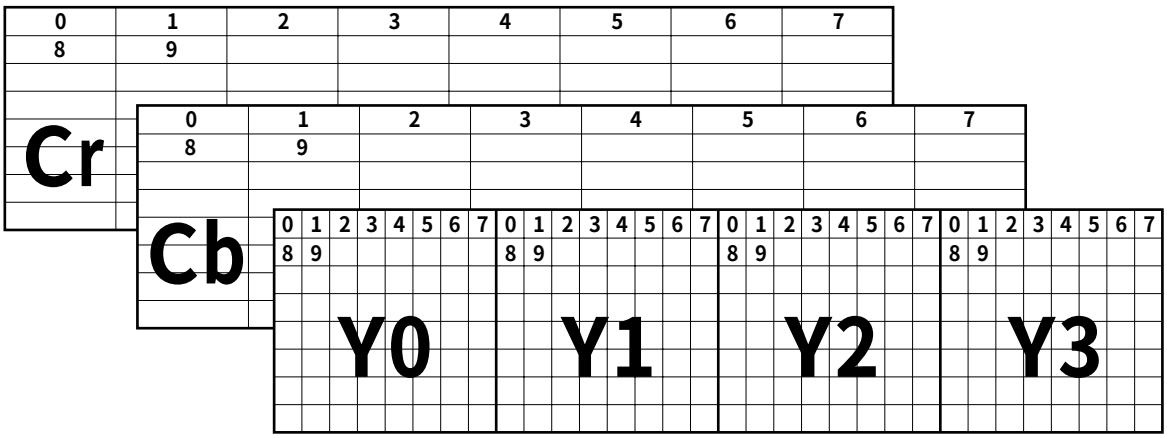


図２－26　1MCU分の画像データ（2/2）

（c）4：1：1［H：V＝2：2］の場合



（d）4：1：1［H：V＝4：1］の場合



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２－25　MCUとブロック

サンプル比	MCU単位	ブロック
4：4：4［H：V＝1：1］	8×8	３ブロック（Ｙ成分１ブロック，Cb成分１ブロック，Cr成分１ブロック）
4：2：2［H：V＝2：1］	16×8	４ブロック（Ｙ成分２ブロック，Cb成分１ブロック，Cr成分１ブロック）
4：1：1［H：V＝2：2］	16×16	６ブロック（Ｙ成分４ブロック，Cb成分１ブロック，Cr成分１ブロック）
4：1：1［H：V＝4：1］	32×8	６ブロック（Ｙ成分４ブロック，Cb成分１ブロック，Cr成分１ブロック）

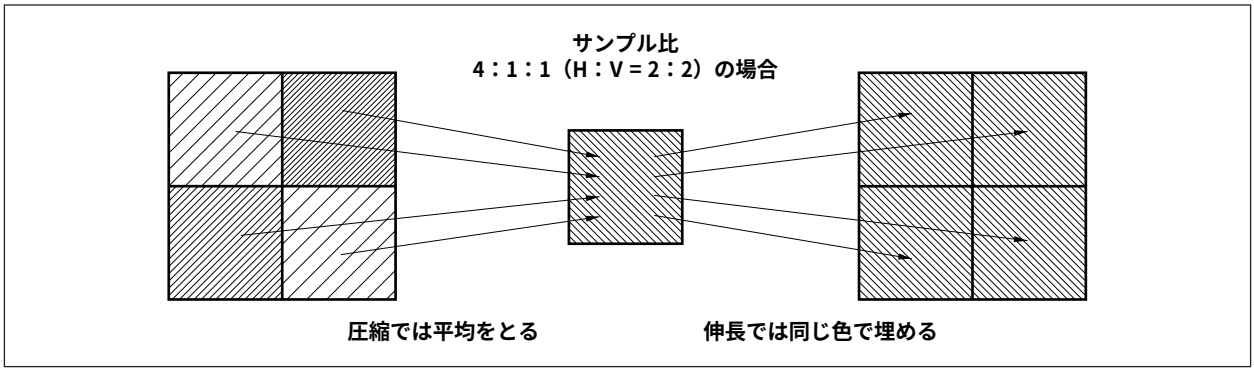
圧縮処理では，色差成分（Cb/Cr）について隣り合ういくつかのピクセルの平均をとる必要があります（サンプル比4：4：4は除く）。この平均をとる処理をサンプリングと呼んでいます。

伸長処理では，逆に圧縮で平均をとったピクセルに対して同じ色差成分値を書き出します。

表２－26　色差成分のサンプリング

サンプル比	サンプリング
4：4：4［H：V＝1：1］	平均はとらない
4：2：2［H：V＝2：1］	横２ピクセル分の色差成分の平均をとる
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

図２－27　サンプリング

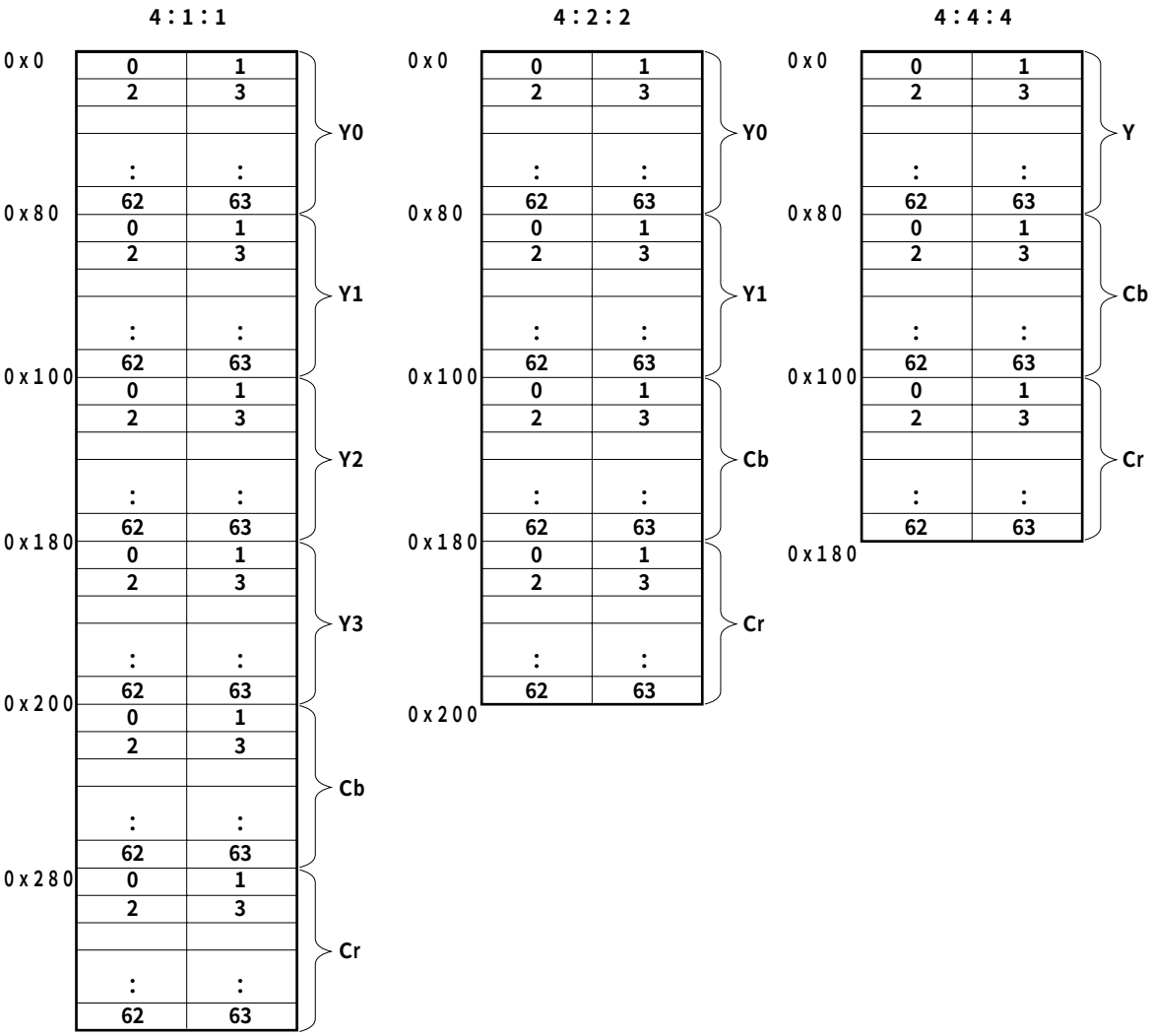


2.6.3　MCUバッファ

1MCU分の画像データは，MCUバッファに入れられます。

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２－28　MCUバッファの画像データ



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。

２．６．４　圧縮時のVRAMアクセス関数

圧縮の場合，VRAMアクセスのために必要な関数は次のとおりです。

表２－27　圧縮時に使用するVRAMアクセス関数

関数名	機能概要
jpeg_getMCU_M	VRAMアクセス関数（モノクロ圧縮）
jpeg_getMCU_11	VRAMアクセス関数（カラー圧縮サンプル比4：4：4［H：V＝1：1］）
jpeg_getMCU_21	VRAMアクセス関数（カラー圧縮サンプル比4：2：2［H：V＝2：1］）
jpeg_getMCU_22	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝2：2］）
jpeg_getMCU_41	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝4：1］）

これらの関数は，VRAMからMCU単位でデータを取り出し，YCbCr形式に変換して，MCUバッファに所定の形式で格納します。不要なサンプル比の関数は，作成する必要はありません。なお，関数名はユーザが任意につけることはできません。必ず表２－27と同じ関数名にしてください。

それぞれの関数仕様は次のようになっています。

関 数 名　jpeg_getMCU_M（モノクロ圧縮）
　　jpeg_getMCU_11（カラー圧縮サンプル比4：4：4［H：V＝1：1］）
　　jpeg_getMCU_21（カラー圧縮サンプル比4：2：2［H：V＝2：1］）
　　jpeg_getMCU_22（カラー圧縮サンプル比4：1：1［H：V＝2：2］）
　　jpeg_getMCU_41（カラー圧縮サンプル比4：1：1［H：V＝4：1］）

機能概要　画像メモリからデータをサンプリングし，MCUバッファへ格納

形　　式　#include “ jpeg.h ”
　　void jpeg_getMCU_M（JPEGINFO ＊*info*）
　　void jpeg_getMCU_11（JPEGINFO ＊*info*）
　　void jpeg_getMCU_21（JPEGINFO ＊*info*）
　　void jpeg_getMCU_22（JPEGINFO ＊*info*）
　　void jpeg_getMCU_41（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**２．７．２　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　*info*のメンバCurrenX/CurrentYで指定される画像メモリの現在位置から1MCU分のデータをサンプリングしMCUバッファへ格納します。

備　　考　jpeg_getMCU_xx関数をアセンブラで記述する場合は，Ｃ言語規約に従って記述してください。

それぞれのサンプル比に対応する1MCUの大きさは，表２－25のようになります。VRAMがRGBの場合には，それぞれのピクセルをYCbCrに変換したものが対応します。

MCUバッファへのデータの格納形式は，**２．６．３　MCUバッファ**を参照してください。

jpeg_getMCU_xx関数には，次の情報が必要です。

JPEGINFO->MCU_Buff_Bptr　：MCUバッファの先頭アドレス
JPEGINFO->CurrentX　：対応するMCUのｘ座標
JPEGINFO->CUrrentY　：対応するMCUのｙ座標

2.6.5　伸長時のVRAMアクセス関数

伸長の場合，VRAMアクセスのために必要な関数は次のとおりです。

表２－28　伸長時に使用するVRAMアクセス関数

関数名	機能概要
jpeg_putMCU_M	VRAMアクセス関数（モノクロ伸長）
jpeg_putMCU_11	VRAMアクセス関数（カラー伸長サンプル比4：4：4［H：V＝1：1］）
jpeg_putMCU_21	VRAMアクセス関数（カラー伸長サンプル比4：2：2［H：V＝2：1］）
jpeg_putMCU_22	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝2：2］）
jpeg_putMCU_41	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝4：1］）

これらの関数は，MCUバッファに所定の形式で格納されているYCbCr形式のデータをVRAMに描画します。不要なサンプル比の関数は，作成する必要がありません。なお，関数名はユーザが任意につけることはできません。必ず表２－28と同じ関数名にしてください。

それぞれの関数仕様は次のようになっています。

関 数 名　jpeg_putMCU_M（モノクロ伸長）
　　jpeg_putMCU_11（カラー伸長サンプル比4：4：4［H：V＝1：1］）
　　jpeg_putMCU_21（カラー伸長サンプル比4：2：2［H：V＝2：1］）
　　jpeg_putMCU_22（カラー伸長サンプル比4：1：1［H：V＝2：2］）
　　jpeg_putMCU_41（カラー伸長サンプル比4：1：1［H：V＝4：1］）
機能概要　MCUバッファのデータを画像メモリに展開
形　　式　#include “ jpeg.h ”
　　void jpeg_putMCU_M（JPEGINFO ＊*info*）
　　void jpeg_putMCU_11（JPEGINFO ＊*info*）
　　void jpeg_putMCU_21（JPEGINFO ＊*info*）
　　void jpeg_putMCU_22（JPEGINFO ＊*info*）
　　void jpeg_putMCU_41（JPEGINFO ＊*info*）
引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし
機　　能　*info*のメンバCurrenX/CurrentYで指定される画像メモリの現在位置へ1MCU分のデータを展開します。
備　　考　これら関数をアセンブラで記述する場合は，Ｃ言語規約に従って記述してください。

それぞれのサンプル比に対応する1MCUの大きさは，表２－25のようになります。VRAMがRGBの場合には，それぞれのピクセルをYCbCrに変換したものが対応します。

MCUバッファへのデータの格納形式は，**2.6.3　MCUバッファ**を参照してください。

これらの関数には，次の情報が必要です。

JPEGINFO->MCU_Buff_Bptr　：MCUバッファの先頭アドレス
JPEGINFO->CurrentX　：対応するMCUのｘ座標
JPEGINFO->CUrrentY　：対応するMCUのｙ座標

2.6.6　アセンブラ・コーディングのワン・ポイント

このルーチンをアセンブラで記述する場合，ロード／ストア命令の16ビット・ディスプレースメントを利用すると，処理が高速化できます。

たとえば，次のように実行するとします。

```
＊（mcu＋0）= Y0；
＊（mcu＋1）= Y1；
＊（mcu＋2）= Y2；
＊（mcu＋3）= Y3；
```

このとき，レジスタに次のような内容が格納されているとします。

```
r7：mcu
r10：Y0
r11：Y1
r12：Y2
r13：Y3
```

そこで，次のように記述すると，アドレス計算が大幅に短縮されて，結果として高速化されます。

```
st.h　r10, 0x0000［r7］
st.h　r11, 0x0002［r7］
st.h　r12, 0x0004［r7］
st.h　r13, 0x0006［r7］
```

また，次のようにアセンブラ・ファイル内でシンボルの値を定義しておきます（Ｃ言語の*#define*と同じ，GHS版の場合）。

```
VRAM_GAP1 = 1 … VRAMの同一ピクセルのＹとCbとのアドレス差
VRAM_GAP2 = 2 … VRAMの同一ピクセルのＹとCrとのアドレス差
```

r8に，アクセスしようとするVRAMのピクセルのＹ成分のアドレスが入っているとして，次のように命令を実行します。

```
ld.b  0x0000［r8］, r10
ld.b  VRAM_GAP1［r8］, r11
ld.b  VRAM_GAP2［r8］, r12
andi 0x00FF, r10, r10
andi 0x00FF, r11, r11
andi 0x00FF, r12, r12
```

この結果，レジスタには次の内容が格納されます。

r10：Ｙ成分の値
r11：Cb成分の値
r12：Cr成分の値

２.７　基本ライブラリ・レファレンス

２.７.１　データ型詳細

データ型は，表２－29に示す２種類があります。これらの構造体の変数，およびメンバにアドレスを渡す変数は，すべて外部変数で宣言してください。

表２－29　データ型

分　類	データ型名	データ型	サイズ	機能概要
共通	JPEGINFO	構造体	148バイト	圧縮，伸長，解析パラメータ設定
	APPINFO	構造体	96バイト	APPnマーカ情報保存

２.７.２　JPEGINFO構造体の構造

JPEGINFO構造体のメンバ構成を次に示します。

この構造体は，JPEGの圧縮／伸長／解析処理で共通して使用されます。

表２－30　JPEGINFO構造体のメンバ

メンバ	型	サイズ（バイト）	IN/OUT			説　明
			圧縮	伸長	解析	
ErrorState	long	4	OUT	OUT	OUT	エラー・ステータス
FileSize	long	4	OUT	OUT	OUT	JPEGファイル・サイズ
Quality	char	1	IN	RSV	RSV	量子化パラメータ
Sampling	char	1	IN	OUT	OUT	サンプル比
Restart	unsigned short	2	IN	OUT	OUT	リスタート・インターバル（DRI/RSTm）の設定
Width	unsigned short	2	IN	OUT	OUT	画像の横サイズ
Height	unsigned short	2	IN	OUT	OUT	画像の縦サイズ
StartX	unsigned short	2	IN	IN	RSV	画像の始点ｘ座標
StartY	unsigned short	2	IN	IN	RSV	画像の始点ｙ座標
CurrentX	unsigned short	2	RSV	RSV	RSV	カレントｘ座標
CurrentY	unsigned short	2	RSV	RSV	RSV	カレントｙ座標
VRAM_Bptr	unsigned char＊	4	IN	IN	RSV	VRAM先頭アドレス
VRAM_W_Pixel	unsigned short	2	IN	IN	RSV	VRAM仕様（幅　：横方向ピクセル数）
VRAM_H_Pixel	unsigned short	2	IN	IN	RSV	VRAM仕様（高さ：縦方向ピクセル数）
VRAM_Line_Byte	short	2	IN	IN	RSV	VRAM仕様（縦１ピクセル分アドレス差分バイト数）
VRAM_Pixel_Byte	short	2	IN	IN	RSV	VRAM仕様（横１ピクセル分アドレス差分バイト数）
VRAM_Gap1_Byte	short	2	IN	IN	RSV	VRAM仕様（Y-Cb/R-Gアドレス差分バイト数）
VRAM_Gap2_Byte	short	2	IN	IN	RSV	VRAM仕様（Y-Cr/R-Bアドレス差分バイト数）
JPEG_Buff_Bptr	unsigned char＊	4	IN	IN	IN	JPEGファイル格納バッファの先頭アドレス
JPEG_Buff_Eptr	unsigned char＊	4	IN	IN	IN	JPEGファイル格納バッファの末尾アドレス
MCU_Buff_Bptr	short＊	4	IN	IN	RSV	MCUバッファの設定
RGB_Buff_Bptr	short＊	4	RSV	RSV	RSV	予約フィールド
DQT_Y_Bptr	char＊	4	IN	RSV	RSV	輝度成分用量子化テーブルのアドレス
DQT_C_Bptr	char＊	4	IN	RSV	RSV	色差成分用量子化テーブルのアドレス
DHT_DC_Y_Bptr	char＊	4	IN	RSV	RSV	輝度DC成分用ハフマン・テーブルのアドレス
DHT_DC_C_Bptr	char＊	4	IN	RSV	RSV	色差DC成分用ハフマン・テーブルのアドレス
DHT_AC_Y_Bptr	char＊	4	IN	RSV	RSV	輝度AC成分用ハフマン・テーブルのアドレス
DHT_AC_C_Bptr	char＊	4	IN	RSV	RSV	色差AC成分用ハフマン・テーブルのアドレス
APP_Info_Bptr	APPINFO＊	4	IN	RSV	IN	APPINFO構造体変数のアドレス
Work1_Bptr	long＊	4	IN	IN	IN	ライブラリ用ワーク・エリア１のアドレス
Work2_Bptr	long＊	4	IN	IN	IN	ライブラリ用ワーク・エリア２のアドレス
Mode	long	4	IN	IN	RSV	動作モード
DNL	long	4	IN	RSV	RSV	DNLセグメントの設定
CI1	long	4	IN	OUT	OUT	コンポーネントID1
CI2	long	4	IN	OUT	OUT	コンポーネントID2
CI3	long	4	IN	OUT	OUT	コンポーネントID3
User	long	4	IN	IN	IN	ユーザ定義領域
JPEG_Buff_Cptr	unsigned char＊	4	OUT	OUT	OUT	JPEGファイル格納バッファのカレント・アドレス
Work3	char	28	RSV	RSV	RSV	ライブラリ用ワーク・エリア３
合計		148				

備考　IN　　：ユーザが値を設定してください。

OUT　：基本ライブラリが値を設定します（返却値）。

RSV　：基本ライブラリの予約フィールドです。

各メンバの詳細は，次のとおりです。

（１）ErrorState

分類	IN/OUT	アドレス値	説　　明
圧縮処理	OUT	0x00000000	エラー・ステータスを格納します。
伸長処理	OUT	｜	エラー・ステータスの詳細に関しては，**2.7.6　関数**を参照してください。
解析処理	OUT	0xFFFFFFFF	

（２）FileSize

分類	IN/OUT	アドレス値	説　　明
圧縮処理	OUT	0x00000000	JPEGファイルのサイズを格納します。
伸長処理	OUT	｜	
解析処理	OUT	0xFFFFFFFF	

（３）Quality

分類	IN/OUT	値	説　　明
圧縮処理	IN	0-100	量子化計数を設定してください。
伸長処理	RSV	－	－
解析処理	RSV	－	－

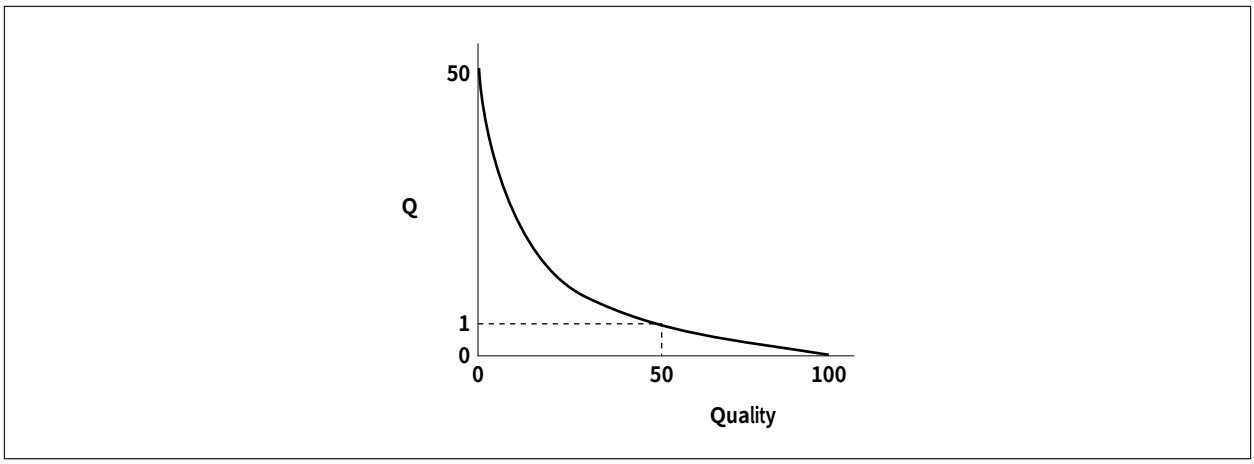
AP703000-B03では，量子化テーブルの値を簡単に変更できるように，量子化パラメータ（Quality）を用意しています。

AP703000-B03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 = 50/Quality（Quality = 0の場合，プログラム上で自動的に１になります。）

Quality≧50の場合：Q = 2－Quality/50

図２－29　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２－31　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

★ （４）Sampling

分類	IN/OUT	値	説　明
圧縮処理	IN	JPEGSAMPLE11 JPEGSAMPLE21 JPEGSAMPLE22 JPEGSAMPLE41 JPEGSAMPLEMONO	サンプル比を設定してください。 この情報は，JPEGヘッダに付加されます。 モノクロを使用する際には，任意のサンプル比とORした値を設定してください（**２.３.５　モノクロ・フォーマット対応**参照）。
伸長処理	OUT	JPEGSAMPLE11 JPEGSAMPLE21 JPEGSAMPLE22 JPEGSAMPLE41 JPEGSAMPLEMONO	サンプル比は，JPEGファイルから自動的に格納されます。 伸長ライブラリでは，どのサンプル比が使われているか自動的に判別するので，考慮する必要はありません。 モノクロを使用する際には，任意のサンプル比とORした値を設定してください（**２.３.５　モノクロ・フォーマット対応**参照）。
解析処理	OUT	上位４ビット［0x1-0x4］ 下位４ビット［0x1-0x4］	サンプル比は，JPEGファイルから自動的に格納されます。

JPEGSAMPLEnは，圧縮／伸長処理用としてヘッダ・ファイル“jpeg.h”で定義されています。

Sampling （サンプル比）	JPEGSAMPLE22（4：1：1［H：V=2：2］） JPEGSAMPLE41（4：1：1［H：V=4：1］）	JPEGSAMPLE21 （4：2：2［H：V＝2：1］）	JPEGSAMPLE11 （4：4：4［H：V＝1：1］）
色彩	普通	←→	鮮明
ファイル・サイズ	基準値（１倍）	約4/3倍	約２倍

★ **備考**　モノクロ・フォーマットの場合，色彩，ファイル・サイズ共にサンプル比には依存しません。

（５）Restart

分類	IN/OUT	値	説　明
圧縮処理	IN	0	RSTmマーカ/DRIマーカを挿入しません。
		1-65535	RSTmマーカを挿入します。リスタート・インターバル（DRI）は，設定された値です。
伸長処理	OUT	0	RSTmマーカ/DRIマーカは，挿入されていません。
		1-65535	RSTmマーカが挿入されています。リスタート・インターバル（DRI）は，格納された値です。
解析処理	OUT	0	RSTmマーカ/DRIマーカは，挿入されていません。
		1-65535	RSTmマーカが挿入されています。リスタート・インターバル（DRI）は，格納された値です。

（６）Width

分類	IN/OUT	値	説　　明
圧縮処理	IN	1-65535	JPEG圧縮画像の横サイズを設定してください。
伸長処理	OUT	1-65535	JPEG圧縮画像の横サイズがJPEGファイルから自動的に格納されます。
解析処理	OUT	1-65535	JPEG圧縮画像の横サイズがJPEGファイルから自動的に格納されます。

圧縮／伸長処理の場合は，サンプル比（Sampling）により「値」に制限があります。

Sampling	Width
4：4：4［H：V = 1：1］	８の倍数
4：2：2［H：V = 2：1］	16の倍数
4：1：1［H：V = 2：2］	16の倍数
4：1：1［H：V = 4：1］	32の倍数

（７）Height

分類	IN/OUT	値	説　　明
圧縮処理	IN	0-65535	JPEG圧縮画像の縦サイズを設定してください。
伸長処理	OUT	0-65535	JPEG圧縮画像の縦サイズがJPEGファイルから自動的に格納されます。
解析処理	OUT	0-65535	JPEG圧縮画像の縦サイズがJPEGファイルから自動的に格納されます。

圧縮／伸長処理の場合は，サンプル比により「値」に制限があります。

Sampling	Height
4：4：4［H：V = 1：1］	８の倍数
4：2：2［H：V = 2：1］	８の倍数
4：1：1［H：V = 2：2］	16の倍数
4：1：1［H：V = 4：1］	８の倍数

（８）StartX

分類	IN/OUT	値	説　　明
圧縮処理	IN	0-32767	JPEG圧縮画像の圧縮開始ｘ座標を設定してください。
伸長処理	IN	0-32767	JPEG圧縮画像の伸長開始ｘ座標を設定してください。
解析処理	RSV	－	－

（９）StartY

分類	IN/OUT	値	説　　明
圧縮処理	IN	0-32767	JPEG圧縮画像の圧縮開始ｙ座標を設定してください。
伸長処理	IN	0-32767	JPEG圧縮画像の伸長開始ｙ座標を設定してください。
解析処理	RSV	－	－

（10）CurrentX

分類	IN/OUT	値	説　　明
圧縮処理	RSV	0-32767	JPEG圧縮画像の圧縮処理カレントｘ座標が格納されます。 関数jpeg_getMCU_xxをカスタマイズする場合のみ，その関数内での参照のみ許可します。
伸長処理	RSV	0-32767	JPEG圧縮画像の伸長処理カレントｘ座標が格納されます。 関数jpeg_putMCU_xxをカスタマイズする場合のみ，その関数内での参照のみ許可します。
解析処理	RSV	－	－

（11）CurrentY

分類	IN/OUT	値	説　明
圧縮処理	RSV	0-32767	JPEG圧縮画像の圧縮処理カレントｙ座標が格納されます。 関数jpeg_getMCU_xxをカスタマイズする場合のみ，その関数内での参照のみ許可します。
伸長処理	RSV	0-32767	JPEG圧縮画像の伸長処理カレントｙ座標が格納されます。 関数jpeg_putMCU_xxをカスタマイズする場合のみ，その関数内での参照のみ許可します。
解析処理	RSV	－	－

（12）VRAM_Bptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000 ｜ 0xFFFFFFFF	画像メモリ（VRAM）の先頭アドレスを設定してください。
伸長処理	IN		
解析処理	RSV	－	－

（13）VRAM_W_Pixel

分類	IN/OUT	値	説　明
★ 圧縮処理	IN	1-10912	画像メモリ（VRAM）の横方向ピクセル数を設定してください。
伸長処理	IN		
解析処理	RSV	－	－

（14）VRAM_H_Pixel

分類	IN/OUT	値	説　明
★ 圧縮処理	IN	1-10912	画像メモリ（VRAM）の縦方向ピクセル数を設定してください。
伸長処理	IN		
解析処理	RSV	－	－

（15）VRAM_Line_Byte

分類	IN/OUT	値	説　明
圧縮処理	IN	1-32767	画像メモリ（VRAM）の縦方向１ピクセル分アドレス差分バイト数を設定してください。
伸長処理	IN		
解析処理	RSV	－	－

（16）VRAM_Pixel_Byte

分類	IN/OUT	値	説　明
圧縮処理	IN	1-32767	画像メモリ（VRAM）の横方向１ピクセル分アドレス差分バイト数を設定してください。
伸長処理	IN		
解析処理	RSV	－	－

（17）VRAM_Gap1_Byte

分類	IN/OUT	値	説　明
圧縮処理	IN	－32768 ｜ ＋32767	画像メモリ（VRAM）がYCbCrの場合は，同一ピクセルＹ成分/Cb成分のアドレス差分バイト数。 画像メモリ（VRAM）がRGBの場合は，同一ピクセルＲ成分/Ｇ成分のアドレス差分バイト数。
伸長処理	IN		
解析処理	RSV	－	－

（18）VRAM_Gap2_Byte

分類	IN/OUT	値	説　明
圧縮処理	IN	－32768	画像メモリ（VRAM）がYCbCrの場合は，同一ピクセルＹ成分/Cr成分のアドレス差分バイト数。
伸長処理	IN	｜	
		＋32767	画像メモリ（VRAM）がRGBの場合は，同一ピクセルＲ成分/Ｂ成分のアドレス差分バイト数。
解析処理	RSV	－	－

（19）JPEG_Buff_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000	JPEGバッファの先頭アドレスを設定してください。
伸長処理	IN	｜	
解析処理	IN	0xFFFFFFFF	

（20）JPEG_Buff_Eptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000	JPEGバッファの末尾アドレスを設定してください。
伸長処理	IN	｜	「末尾アドレス」=「先頭アドレス」＋「バッファ・サイズ」とします。
解析処理	IN	0xFFFFFFFF	

（21）MCU_Buff_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000	MCUバッファの先頭アドレスを設定してください。
伸長処理	IN	｜	
		0xFFFFFFFF	
解析処理	RSV	－	－

（22）RGB_Buff_Bptr

分類	IN/OUT	値	説　明
圧縮処理	RSV	－	－
伸長処理	RSV		
解析処理	RSV		

（23）DQT_Y_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000	輝度成分用量子化テーブルのアドレスを設定してください。
		｜	基本ライブラリが提供するテーブルは，jpeg_DQT_Yです。
		0xFFFFFFFF	
伸長処理	RSV	－	－
解析処理	RSV		

（24）DQT_C_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000	色差成分用量子化テーブルのアドレスを設定してください。
		｜	基本ライブラリが提供するテーブルは，jpeg_DQT_Cです。
		0xFFFFFFFF	
伸長処理	RSV	－	－
解析処理	RSV		

（25）DHT_DC_Y_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000 ｜ 0xFFFFFFFF	輝度DC成分用ハフマン・テーブルのアドレスを設定してください。 基本ライブラリが提供するテーブルは，jpeg_DHT_DC_Yです。 ハフマン・テーブルを作成する際は**2.3.2（1）ハフマン・テーブルの制限事項**を参照してください。
伸長処理	RSV	－	－
解析処理	RSV		

（26）DHT_DC_C_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000 ｜ 0xFFFFFFFF	色差DC成分用ハフマン・テーブルのアドレスを設定してください。 基本ライブラリが提供するテーブルは，jpeg_DHT_DC_Cです。 ハフマン・テーブルを作成する際は**2.3.2（1）ハフマン・テーブルの制限事項**を参照してください。
伸長処理	RSV	－	－
解析処理	RSV		

（27）DHT_AC_Y_Bptr

分類	IN/OUT	アドレス値	説　明
圧縮処理	IN	0x00000000 ｜ 0xFFFFFFFF	輝度AC成分用ハフマン・テーブルのアドレスを設定してください。 基本ライブラリが提供するテーブルは，jpeg_DHT_AC_Yです。 ハフマン・テーブルを作成する際は**2.3.2（1）ハフマン・テーブルの制限事項**を参照してください。
伸長処理	RSV	－	－
解析処理	RSV		

（28）DHT_AC_C_Bptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000 ｜ 0xFFFFFFFF	色差AC成分用ハフマン・テーブルのアドレスを設定してください。 基本ライブラリが提供するテーブルは，jpeg_DHT_AC_Cです。 ハフマン・テーブルを作成する際は**2.3.2（1）ハフマン・テーブルの制限事項**を参照してください。
伸長処理	RSV	－	－
解析処理	RSV		

（29）APP_Info_Bptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000	APPnマーカを挿入しません。
		0x00000000 ｜ 0xFFFFFFFF	APPnマーカを挿入します。APPnセグメントの内容を示すAPPINFO構造体変数のアドレスを設定してください。
伸長処理	RSV	－	－
解析処理	IN	0x00000000	APPnマーカの解析は行いません。
		0x00000000 ｜ 0xFFFFFFFF	APPnマーカの解析を行います。解析結果を格納するAPPINFO構造体変数のアドレスを設定してください。 APPINFO構造体は，解析初期化関数jpeg_AnalysisInitで０にクリアされます。

★

（30）Work1_Bptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000	ライブラリ用高速ワーク・エリアのアドレスを設定してください。
伸長処理	IN	｜	ワーク・エリアは，ユーザが「long型で256バイト」のバッファを宣言してください。
解析処理	IN	0xFFFFFFFF	また，このワーク・エリアを内部RAMに配置すると処理速度向上に寄与します。

（31）Work2_Bptr

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000	ライブラリ用低速ワーク・エリアのアドレスを設定してください。
		｜	ワーク・エリアは，ユーザが「long型で3072バイト」のバッファを宣言してください。
伸長処理	IN	0xFFFFFFFF	ライブラリ用低速ワーク・エリアのアドレスを設定してください。
			ワーク・エリアは，ユーザが「long型で8192バイト」のバッファを宣言してください。
解析処理	RSV	－	－

（32）Mode

分類	IN/OUT	値	説　明
圧縮処理	IN	JPEGMODE_NORMAL	通常圧縮モード。１回の関数コールで１画面すべて圧縮します。１画面分の画像が格納できるだけのVRAMが必要です。
		JPEGMODE_MCULINE	ライン圧縮モード。１回の関数コールで1MCUラインのみ圧縮します。1MCUライン分の画像が格納できるだけのVRAMが必要です。複数回，関数コールすることで１画面分の圧縮を実現します。
		JPEGMODE_STOP	圧縮処理を強制的に終了させます。ライン圧縮モード（JPEGMODE_MCULINE）で強制的に終了させたい場合に使用します。 構造体メンバDNLの値がJPEGDNL_OFFの場合は，EOIマーカを付加後，終了します。 構造体メンバDNLの値がJPEGDNL_ONの場合は，DNLマーカ/EOIマーカを付加後，終了します。 下記２つの条件が満たされた場合のみ，次のライブラリの呼び出し時に設定できます。 ①ライン処理モード（JPEGMODE_MCULINE）で圧縮。 ②最後のライブラリ呼び出し時の返却値がVRAM更新（JPEGRET_CONT2）。
伸長処理	IN	JPEGMODE_NORMAL	通常伸長モード。１回の関数コールで１画面すべて伸長します。１画面分の画像が格納できるだけのVRAMが必要です。
		JPEGMODE_MCULINE	ライン伸長モード。１回の関数コールで1MCUラインのみ伸長します。1MCUライン分の画像が格納できるだけのVRAMが必要です。複数回，関数コールすることで１画面分の伸長を実現します。
解析処理	RSV	－	－

（33）DNL

分類	IN/OUT	値	説　明
圧縮処理	IN	JPEGDNL_ON	圧縮処理終了後，DNLマーカを付加します。
		JPEGDNL_OFF	圧縮処理終了後，DNLマーカを付加しません。
伸長処理	RSV	－	－
解析処理	RSV		

★ **注意　Heightが０の場合は必ずJPEGDNL_ONにしてください。**

（34）CI1, CI2, CI3

分類	IN/OUT	値	説　明
★ 圧縮処理	IN	0-255	各コンポーネントIDが異なるように値を設定してください。
伸長処理	OUT	0-255	SOFn/SOSセグメントで指定されるコンポーネントIDが設定されます。
解析処理	OUT		

（35）User

分類	IN/OUT	値	説　明
圧縮処理	IN	0x00000000	ユーザが自由に使用できます。基本ライブラリは，このメンバを参照しません。
伸長処理	IN	｜	
解析処理	IN	0xFFFFFFFF	

（36）JPEG_Buff_Cptr

分類	IN/OUT	値	説　明
圧縮処理	OUT	－	JPEGバッファの最終アクセス位置（アドレス）を格納します。
伸長処理	OUT		圧縮時には，最終書き込み位置，伸長／解析時には，最終読み込み位置を格納します。
解析処理	OUT		このメンバは，ReadOnlyですのでユーザの書き込みは禁止します。

（37）Work3

分類	IN/OUT	値	説　明
圧縮処理	RSV	－	－
伸長処理	RSV		
解析処理	RSV		

２．７．３　APPINFO構造体の構造

APPINFO構造体は，圧縮処理／解析処理でAPPnセグメントに対応する場合に使用します（圧縮処理では，JPEGファイルにAPPnセグメントを挿入する場合。解析処理では，JPEGファイル中のAPPnセグメントの情報を獲得する場合）。

圧縮処理／解析処理でAPPnセグメントに対応する場合は，APPINFO構造体変数を宣言し，先頭アドレスを
★ JPEGINFO構造体のメンバ「APP_Info_Bptr」に設定してください。「APP_Info_Bptr」が０でない場合，解析初期化関数jpeg_AnalysisInit中ですべてのメンバが０にクリアされます。

圧縮処理／解析処理でAPPnセグメントに対応しない場合，および伸長処理では，APPINFO構造体変数の宣言は不要です。

表２－32　APPINFO構造体のメンバ

メンバ	型	サイズ（バイト）	IN/OUT			説　明
			圧縮	伸長	解析	
APP00_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	APPn（n = 0-15）セグメントのデータのアドレス。
APP01_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	［圧縮］
APP02_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	0x00000000
APP03_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	APPnセグメントを挿入しません。
APP04_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	0x00000000以外
APP05_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	APPnセグメントを挿入します。挿入するデータを格納
APP06_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	したバッファを宣言し，アドレスを設定してください。
APP07_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	［伸長］
APP08_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	設定を禁止します。
APP09_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	［解析］
APP10_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	0x00000000
APP11_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	APPnセグメントは，挿入されていません。
APP12_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	0x00000000以外
APP13_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	APPnセグメントが挿入されています。
APP14_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	データのアドレスをJPEGファイル内の絶対アドレ
APP15_Buff_Bptr	unsigned char＊	4	IN	RSV	OUT	スで格納します。
APP00_BuffSize	unsigned short	2	IN	RSV	OUT	APPn（n = 0-15）セグメントのデータのサイズ。
APP01_BuffSize	unsigned short	2	IN	RSV	OUT	［圧縮］
APP02_BuffSize	unsigned short	2	IN	RSV	OUT	0x00000000
APP03_BuffSize	unsigned short	2	IN	RSV	OUT	APPnセグメントを挿入しません。
APP04_BuffSize	unsigned short	2	IN	RSV	OUT	0x00000000以外
APP05_BuffSize	unsigned short	2	IN	RSV	OUT	APPnセグメントを挿入します。挿入するデータを格納
APP06_BuffSize	unsigned short	2	IN	RSV	OUT	したバッファを宣言し，データ・サイズを設定してくだ
APP07_BuffSize	unsigned short	2	IN	RSV	OUT	さい。最大データ・サイズは0x0000FFFDとなります。
APP08_BuffSize	unsigned short	2	IN	RSV	OUT	［伸長］
APP09_BuffSize	unsigned short	2	IN	RSV	OUT	設定を禁止します。
APP10_BuffSize	unsigned short	2	IN	RSV	OUT	［解析］
APP11_BuffSize	unsigned short	2	IN	RSV	OUT	0x00000000
APP12_BuffSize	unsigned short	2	IN	RSV	OUT	APPnセグメントは，挿入されていません。
APP13_BuffSize	unsigned short	2	IN	RSV	OUT	0x00000000以外
APP14_BuffSize	unsigned short	2	IN	RSV	OUT	APPnセグメントが挿入されています。
APP15_BuffSize	unsigned short	2	IN	RSV	OUT	データ・サイズを格納します。
合計		96				

備考　IN　：ユーザが値を設定してください。
　　OUT　：基本ライブラリが値を設定します（返却値）。
　　RSV　：基本ライブラリの予約フィールドです。

2.7.4 定　　数

★ 表２－33　シンボル一覧（1/2）

シンボル	値	圧縮	伸長	解析	意　　味
JPEGRET_OK	0x00000000	○	○	○	正常終了
JPEGRET_CONT1	0x00000001	○	○	○	継続１（JPEGバッファによるライブラリの中断）
JPEGRET_CONT2	0x00000002	○	○	×	継続２（VRAMによるライブラリの中断）
JPEGRET_ERR	0xFFFFFFFF	○	○	○	異常終了
JPEGERR_VRAM	0x00000001	○	○	×	VRAM領域指定が不正
JPEGERR_USF	0x00000002	○	○	×	不正なサンプリング比
JPEGERR_QPQ	0x00000003	×	○	×	DQTのPq値が不正（０以外）
JPEGERR_QTQ	0x00000004	×	○	×	DQTのTq値（量子化テーブル番号）が不正（０，１以外）
JPEGERR_DHT	0x00000005	×	○	×	DHTのTc/Th値が不正
JPEGERR_SNS	0x00000006	×	○	○	SOSのNs値（コンポーネント数）が不正（１，３以外）
JPEGERR_SCD	0x00000007	×	○	○	SOSのTan/Tdn値（ハフマン・テーブル番号）が不正
JPEGERR_SSS	0x00000008	×	○	○	SOSのSs値が不正（０以外）
JPEGERR_SSE	0x00000009	×	○	○	SOSのSe値が不正（63以外）
JPEGERR_SAA	0x0000000A	×	○	○	SOSのAh/Al値が不正（０以外）
JPEGERR_SFP	0x0000000B	×	○	○	SOFのP値が不正（８以外）
JPEGERR_SFN	0x0000000C	×	○	○	SOFのNf値が不正（１，３以外）
JPEGERR_UKM	0x0000000D	×	○	○	未知のマーカが発生
JPEGERR_RST	0x0000000E	×	○	×	RSTmマーカが不正
JPEGERR_OTH	0x0000000F	×	○	×	圧縮データが不正（マーカ検出）
JPEGERR_SOS	0x00000010	×	○	×	そのほかのSOSセグメント・エラー（コンポーネトIDが不正）
JPEGERR_SOF	0x00000011	×	○	×	SOFが未定義。または複数定義
JPEGERR_FTQ	0x00000012	×	○	×	伸長に必要な量子化テーブルが未定義
JPEGERR_STH	0x00000013	×	○	×	伸長に必要なハフマン・テーブルが未定義
JPEGERR_IMY	0x00000014	○	○	×	伸長に必要な量子化テーブルが未定義
JPEGERR_FID	0x00000015	○	○	×	SOFのコンポーネントIDが重複
JPEGERR_SID	0x00000016	×	○	×	SOSのコンポーネントIDが重複
JPEGERR_QEL	0x00000017	×	○	×	量子化テーブルの要素が不正（０）
JPEGERR_VIJ	0x00000018	×	○	×	ハフマン・テーブルの要素が不正（重複）
JPEGERR_JPB	0x00000019	○	○	○	JPEGバッファ・サイズが不正（０以下）
JPEGERR_DLH	0x0000001A	○	×	×	DHTのLhの値が不正（２以下）
JPEGERR_DCL	0x0000001B	○	×	×	DC成分用DHTのLの値が不正（17以上）
JPEGERR_HUF	0x0000001C	×	○	×	圧縮データが不正（デコード・エラー）

備考　○：使用します。

×：使用しません。

表２－33　シンボル一覧（2/2）

シンボル	値	圧縮	伸長	解析	意　味
JPEGSAMPLE11	0x00000011	○	○	○	Y成分H：V＝1：1
JPEGSAMPLE12	0x00000012	○	○	○	Y成分H：V＝1：2
JPEGSAMPLE13	0x00000013	○	○	○	Y成分H：V＝1：3
JPEGSAMPLE14	0x00000014	○	○	○	Y成分H：V＝1：4
JPEGSAMPLE21	0x00000021	○	○	○	Y成分H：V＝2：1
JPEGSAMPLE22	0x00000022	○	○	○	Y成分H：V＝2：2
JPEGSAMPLE23	0x00000023	○	○	○	Y成分H：V＝2：3
JPEGSAMPLE24	0x00000024	○	○	○	Y成分H：V＝2：4
JPEGSAMPLE31	0x00000031	○	○	○	Y成分H：V＝3：1
JPEGSAMPLE32	0x00000032	○	○	○	Y成分H：V＝3：2
JPEGSAMPLE33	0x00000033	○	○	○	Y成分H：V＝3：3
JPEGSAMPLE41	0x00000041	○	○	○	Y成分H：V＝4：1
JPEGSAMPLE42	0x00000042	○	○	○	Y成分H：V＝4：2
JPEGSAMPLECOLOR	0x00000000	○	○	○	カラーJPEG
JPEGSAMPLEMONO	0x00000080	○	○	○	モノクロJPEG
JPEGMODE_NORMAL	0x00000000	○	○	×	通常動作モード
JPEGMODE_MCULINE	0x00000001	○	○	×	MCUライン動作モード
JPEGMODE_STOP	0x00000002	○	×	×	強制中断
JPEGDNL_OFF	0x00000000	○	×	×	DNLマーカ挿入
JPEGDNL_ON	0x00000001	○	×	×	DNLマーカ非挿入

備考　○：使用します。

×：使用しません。

2.7.5　外部変数

基本ライブラリが提供するディフォールトのテーブル・データは，次のとおりです。

表２－34　外部変数一覧

分類	外部変数名	型	サイズ（バイト）	説　明
圧縮処理	jpeg_DQT_Y	char	64	輝度成分用量子化テーブル（DQTセグメントのテーブル部のみ定義）
	jpeg_DQT_C	char	64	色差成分用量子化テーブル（DQTセグメントのテーブル部のみ定義）
	jpeg_DHT_DC_Y	char	約50	輝度DC成分用ハフマン・テーブル（DHTセグメント）
	jpeg_DHT_DC_C	char	約50	色差DC成分用ハフマン・テーブル（DHTセグメント）
	jpeg_DHT_AC_Y	char	約300	輝度AC成分用ハフマン・テーブル（DHTセグメント）
	jpeg_DHT_AC_C	char	約300	色差AC成分用ハフマン・テーブル（DHTセグメント）

2.7.6　関　　数

基本ライブラリが提供する関数は，次のとおりです。

表２－35　関数一覧

分類	関数名	機能概要
圧縮処理	jpeg_CompressInit	JPEG圧縮ライブラリ初期化
	jpeg_Compress	JPEG圧縮処理メイン
	jpeg_getMCU_M注	VRAMアクセス関数（モノクロ圧縮）
	jpeg_getMCU_11注	VRAMアクセス関数（カラー圧縮サンプル比4：4：4［H：V＝1：1］）
	jpeg_getMCU_21注	VRAMアクセス関数（カラー圧縮サンプル比4：2：2［H：V＝2：1］）
	jpeg_getMCU_22注	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝2：2］）
	jpeg_getMCU_41注	VRAMアクセス関数（カラー圧縮サンプル比4：1：1［H：V＝4：1］）
伸長処理	jpeg_DecompressInit	JPEG伸長ライブラリ初期化
	jpeg_Decompress	JPEG伸長処理メイン
	jpeg_putMCU_M注	VRAMアクセス関数（モノクロ伸長）
	jpeg_putMCU_11注	VRAMアクセス関数（カラー伸長サンプル比4：4：4［H：V＝1：1］）
	jpeg_putMCU_21注	VRAMアクセス関数（カラー伸長サンプル比4：2：2［H：V＝2：1］）
	jpeg_putMCU_22注	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝2：2］）
	jpeg_putMCU_41注	VRAMアクセス関数（カラー伸長サンプル比4：1：1［H：V＝4：1］）
解析処理	jpeg_AnalysisInit	JPEG解析ライブラリ初期化
	jpeg_Analysis	JPEG解析処理メイン

注　カスタマイズする場合のみ参照してください。

（1）jpeg_CompressInit

分　　類　圧縮処理（1/7）

関 数 名　jpeg_CompressInit

機能概要　JPEG圧縮ライブラリの初期化

形　　式　#include “ jpeg.h ”

　　void jpeg_CompressInit（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG圧縮ライブラリを初期化し，圧縮処理を実行可能な状態にします。

（2）jpeg_Compress

分　　類　圧縮処理（2/7）

関 数 名　jpeg_Compress

機能概要　JPEG圧縮メイン

形　　式　#include “ jpeg.h ”

　　long jpeg_Compress（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ退避要求）
JPEGRET_CONT2	継続２（画像データ更新要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます）

★　機　　能　*info*で指定されるJPEG圧縮を行います。

　　異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_VRAM	圧縮画像の領域指定が不正です。
JPEGERR_USF	圧縮できないサンプル比です。
JPEGERR_IMY	圧縮画像用の縦サイズが不正です。
JPEGERR_FID	コンポーネントIDの値が重複しています。
JPEGERR_DLH	DHTヘッダのLhの値が不正です。
JPEGERR_DCL	DC成分用ハフマン・テーブルが不正です。
JPEGERR_JPB	JPEGバッファ・サイズが不正です。

（３）jpeg_getMCU_M/jpeg_getMCU_11/jpeg_getMCU_21/jpeg_getMCU_22/jpeg_getMCU_41

分　　類　圧縮処理（3/7～7/7）

関 数 名　jpeg_getMCU_M（モノクロ圧縮）
jpeg_getMCU_11（カラー圧縮サンプル比4：4：4［H：V＝1：1］）
jpeg_getMCU_21（カラー圧縮サンプル比4：2：2［H：V＝2：1］）
jpeg_getMCU_22（カラー圧縮サンプル比4：1：1［H：V＝2：2］）
jpeg_getMCU_41（カラー圧縮サンプル比4：1：1［H：V＝4：1］）

機能概要　画像メモリからデータをサンプリングし，MCUバッファへ格納

形　　式　#include “ jpeg.h ”
void jpeg_getMCU_M（JPEGINFO ＊*info*）
void jpeg_getMCU_11（JPEGINFO ＊*info*）
void jpeg_getMCU_21（JPEGINFO ＊*info*）
void jpeg_getMCU_22（JPEGINFO ＊*info*）
void jpeg_getMCU_41（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　*info*のメンバCurrenX/CurrentYで指定される画像メモリの現在位置から1MCU分のデータをサンプリングしMCUバッファへ格納します。

備　　考　jpeg_getMCU_xx関数をアセンブラで記述する場合は，C言語規約に従って記述してください。

（４）jpeg_DecompressInit

分　　類　伸長処理（1/7）

関 数 名　jpeg_DecompressInit

機能概要　JPEG伸長ライブラリの初期化

形　　式　#include “ jpeg.h ”
void jpeg_DecompressInit（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG伸長ライブラリを初期化し，伸長処理を実行可能な状態にします。

（５）jpeg_Decompress

分　　類　伸長処理（2/7）

関 数 名　jpeg_Decompress

機能概要　JPEG伸長メイン

形　　式　#include “ jpeg.h ”

long jpeg_Decompress（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ更新要求）
JPEGRET_CONT2	継続２（画像データ退避要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます）

★　機　　能　*info*で指定されるJPEG伸長を行います。

異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_VRAM	伸長画像の領域指定が不正です。
JPEGERR_USF	伸長できないサンプリング比です。
JPEGERR_QPQ	DQTヘッダのPqの値が０以外の値に設定されています。
JPEGERR_QTQ	DQTヘッダの量子化テーブル番号（Tq）が０，１以外の値です。
JPEGERR_DHT	DHTヘッダのTc, Thの値が不正です。
JPEGERR_SNS	SOSヘッダのNs値（コンポーネント数）が不正です（１，３以外）。
JPEGERR_SCD	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。
JPEGERR_SSS	SOSヘッダのSsの値が０ではありません。
JPEGERR_SSE	SOSヘッダのSeの値が63ではありません。
JPEGERR_SAA	SOSヘッダのAh, Alの値が０ではありません。
JPEGERR_SFP	SOFヘッダのPに８以外の値が設定されています。
JEPGERR_SFN	SOFヘッダのNfの値が不正です（１，３以外）。
JPEGERR_UKM	未知のマーカが現れました。
JPEGERR_OTH	圧縮データ中に不正なマーカが現れました。
JPEGERR_RST	RSTnマーカが不正です。
JPEGERR_SOS	その他のSOSヘッダ・エラーです（コンポーネントIDが不正）。
JPEGERR_SOF	SOFヘッダが定義されていないか，２つ以上定義されています。
JPEGERR_FTQ	必要な量子化テーブルが定義されていません。
JPEGERR_STH	必要なハフマン・テーブルが定義されていません。
JPEGERR_IMY	画像の縦サイズが不正です。
JPEGERR_FID	SOFヘッダのコンポーネントIDが重複しています。
JPEGERR_SID	SOSヘッダのコンポーネントIDが重複しています。
JPEGERR_QEL	量子化テーブルの要素に０が含まれています。
JPEGERR_VIJ	ハフマン・テーブルの要素が重複しています。
JPEGERR_JPB	JPEGバッファ・サイズが不正です。
JPEGERR_HUF	圧縮データが不正です。

（６）jpeg_putMCU_M/jpeg_putMCU_11/jpeg_putMCU_21/jpeg_putMCU_22/jpeg_putMCU_41

分　　類　伸長処理（3/7～7/7）

関 数 名　jpeg_putMCU_M（モノクロ伸長）

jpeg_putMCU_11（カラー伸長サンプル比4：4：4［H：V＝1：1］）

jpeg_putMCU_21（カラー伸長サンプル比4：2：2［H：V＝2：1］）

jpeg_putMCU_22（カラー伸長サンプル比4：1：1［H：V＝2：2］）

jpeg_putMCU_41（カラー伸長サンプル比4：1：1［H：V＝4：1］）

機能概要　MCUバッファのデータを画像メモリに展開

形　　式　#include “ jpeg.h ”

void jpeg_putMCU_M（JPEGINFO ＊*info*）

void jpeg_putMCU_11（JPEGINFO ＊*info*）

void jpeg_putMCU_21（JPEGINFO ＊*info*）

void jpeg_putMCU_22（JPEGINFO ＊*info*）

void jpeg_putMCU_41（JPEGINFO ＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　*info*のメンバCurrenX/CurrentYで指定される画像メモリの現在位置へ1MCU分のデータを展開します。

備　　考　jpeg_putMCU_xx関数をアセンブラで記述する場合は，C言語規約に従って記述してください。

（７）jpeg_AnalysisInit

分　　類　解析処理（1/2）

関 数 名　jpeg_AnalysisInit

機能概要　JPEG解析ライブラリの初期化

形　　式　#include “ jpeg.h ”

void jpeg_AnalysisInit（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　なし

機　　能　JPEG解析ライブラリを初期化し，解析処理を実行可能な状態にします。

（８）jpeg_Analysis

分　　類　解析処理（2/2）

関 数 名　jpeg_Analysis

機能概要　JPEG解析メイン

形　　式　#include “ jpeg.h ”

long jpeg_Analysis（JPEGINFO＊*info*）

引 き 数

引き数	型	説　　明
info	JPEGINFO＊	JPEGINFO構造体[注]へのポインタ

注　JPEGINFO構造体については，**2.7.2　JPEGINFO構造体の構造**を参照してください。

返 却 値　エラー・コード

エラー・コード	説　　明
JPEGRET_OK	正常終了
JPEGRET_CONT1	継続１（JPEGバッファ更新要求）
JPEGRET_ERR	異常終了（エラー内容は，JPEGINFO構造体のメンバ「ErrorState」に格納されます）

★　機　　能　*info*で指定されるJPEG解析を行います。

異常終了の場合，*info*のメンバ「ErrorState」に格納される値は，次のとおりです。

ErrorStateの値	説　　明
JPEGERR_SNS	SOSヘッダのNs値（コンポーネント数）が不正です（１，３以外）。
JPEGERR_SCD	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。
JPEGERR_SSS	SOSヘッダのSsの値が０ではありません。
JPEGERR_SSE	SOSヘッダのSeの値が63ではありません。
JPEGERR_SAA	SOSヘッダのAh, Alの値が０ではありません。
JPEGERR_SFP	SOFヘッダのPに８以外の値が設定されています。
JEPGERR_SFN	SOFヘッダのNfの値が不正です（１，３以外）。
JPEGERR_UKM	未知のマーカが現れました。
JPEGERR_JPB	JPEGバッファ・サイズが不正です。

２.７.７　セクション

基本ライブラリが定義（予約）するセクションは，次のとおりです。

表２－36　基本ライブラリが使用するセクション

処　理	セクション名	属　性		マッピング		説　　明
		NEC版	GHS版	ROM	RAM	
圧縮処理	.JPCTEXT	.text	.text	○	○	テキスト（命令コード）
	.JPCTBL	.const	.rodata	○	○	テーブル・データ（読み取り専用）
	.JPCDATA	.data	.data	×	○	初期値ありデータ
	.JPCBSS	.bss	.bss	×	○	初期値なしデータ
伸長処理	.JPDTEXT	.text	.text	○	○	テキスト（命令コード）
	.JPDTBL	.const	.rodata	○	○	テーブル・データ（読み取り専用）
	.JPDDATA	.data	.data	×	○	初期値ありデータ
	.JPDBSS	.bss	.bss	×	○	初期値なしデータ
解析処理	.JPATEXT	.text	.text	○	○	テキスト（命令コード）
	.JPATBL	.const	.rodata	○	○	テーブル・データ（読み取り専用）
	.JPADATA	.data	.data	×	○	初期値ありデータ
	.JPABSS	.bss	.bss	×	○	初期値なしデータ
共通処理	.JPJTEXT	.text	.text	○	○	テキスト（命令コード）
	.JPJTBL	.const	.rodata	○	○	テーブル・データ（読み取り専用）
	.JPJDATA	.data	.data	×	○	初期値ありデータ
	.JPJBSS	.bss	.bss	×	○	初期値なしデータ

備考　○：配置可能

×：配置不可能

２.７.８　シンボル名規約

基本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。ユーザがアプリケーション内でシンボルなどを定義する際は，重複などのないようご注意ください。

表２－37　基本ライブラリのシンボル名規約

分　類	説　　明
関数	先頭に「jpeg_」を付加しています。
変数	先頭に「jpeg_」を付加しています。
定数	先頭に「JPEG」を付加しています。
データ型	「JPEGINFO」，「APPINFO」
セクション	先頭に「.JP」を付加しています。

２.７.９　基本ライブラリの選択

基本ライブラリとして提供するファイルは，次のとおりです。

表２－38　基本ライブラリのファイル名

分　類	リンク優先順位	ファイル名	説　明	備　　考
圧縮処理	1	libjpegc.a	圧縮処理メイン	JPEG圧縮を使用する場合は，必ず選択してください。
	2	libjcsm.a	モノクロ圧縮	対応させる圧縮方式をすべて選択してください。
	3	libjcs11.a	カラー圧縮サンプル比4：4：4［H：V＝1：1］	
	4	libjcs21.a	カラー圧縮サンプル比4：2：2［H：V＝2：1］	
	5	libjcs22.a	カラー圧縮サンプル比4：1：1［H：V＝2：2］	
	6	libjcs41.a	カラー圧縮サンプル比4：1：1［H：V＝4：1］	
	7	libjcy.a	画像メモリ色空間YCbCr	ディフォールトの“jpeg_getMCU_xx（）”を使用する場合は，どちらかを必ず選択してください。
	8	libjcr.a	画像メモリ色空間RGB	
伸長処理	9	libjpegd.a	伸長処理メイン	JPEG伸長を使用する場合は，必ず選択してください。
	10	libjdsm.a	モノクロ伸長対応	対応させる伸長方式をすべて選択してください。
	11	libjds11.a	カラー伸長サンプル比4：4：4［H：V＝1：1］	
	12	libjds21.a	カラー伸長サンプル比4：2：2［H：V＝2：1］	
	13	libjds22.a	カラー伸長サンプル比4：1：1［H：V＝2：2］	
	14	libjds41.a	カラー伸長サンプル比4：1：1［H：V＝4：1］	
	15	libjdhl.a	ハフマン伸長	JPEG伸長を使用する場合は，必ず選択してください。
	16	libjdy.a	画像メモリ色空間YCbCr	ディフォールトの“jpeg_putMCU_xx（）”を使用する場合は，どちらかを必ず選択してください。
	17	libjdr.a	画像メモリ色空間RGB	
解析処理	18	libjpega.a	解析処理メイン	JPEG解析を使用する場合は，必ず選択してください。
共通処理	19	libjpeg.a	共通処理	基本ライブラリを使用する場合は，必ず選択してください。

表２－39　基本ライブラリ選択例

処理方法	条　　件	選択ライブラリとリンク順
圧縮処理	対応サンプル比　：4：4：4［H：V＝1：1］，4：2：2［H：V＝2：1］ jpeg_getMCU　：ライブラリのディフォールト（YCbCr方式）使用	libjpegc.a libjcs11.a libjcs21.a libjcy.a libjpeg.a
伸長処理	対応サンプル比　：4：2：2［H：V＝2：1］ jpeg_putMCU　：ライブラリのディフォールト（RGB方式）使用	libjpegd.a libjcs21.a libjdhl.a libjcr.a libjpeg.a
解析処理	なし	libjpega.a libjpeg.a

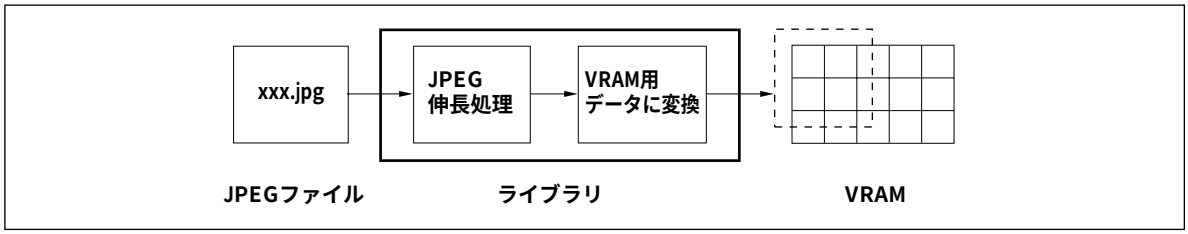
★

第３章　プログレッシブ対応追加ライブラリ仕様

3.1　機　　能

追加ライブラリを用いることにより，基本ライブラリでは対応できなかった画像の伸長が可能となります。
主な機能については，**1.3.4　追加ライブラリの特徴**を参照してください。
JPEGファイルを伸長し，画面に表示します。

図３－１　追加伸長処理



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。

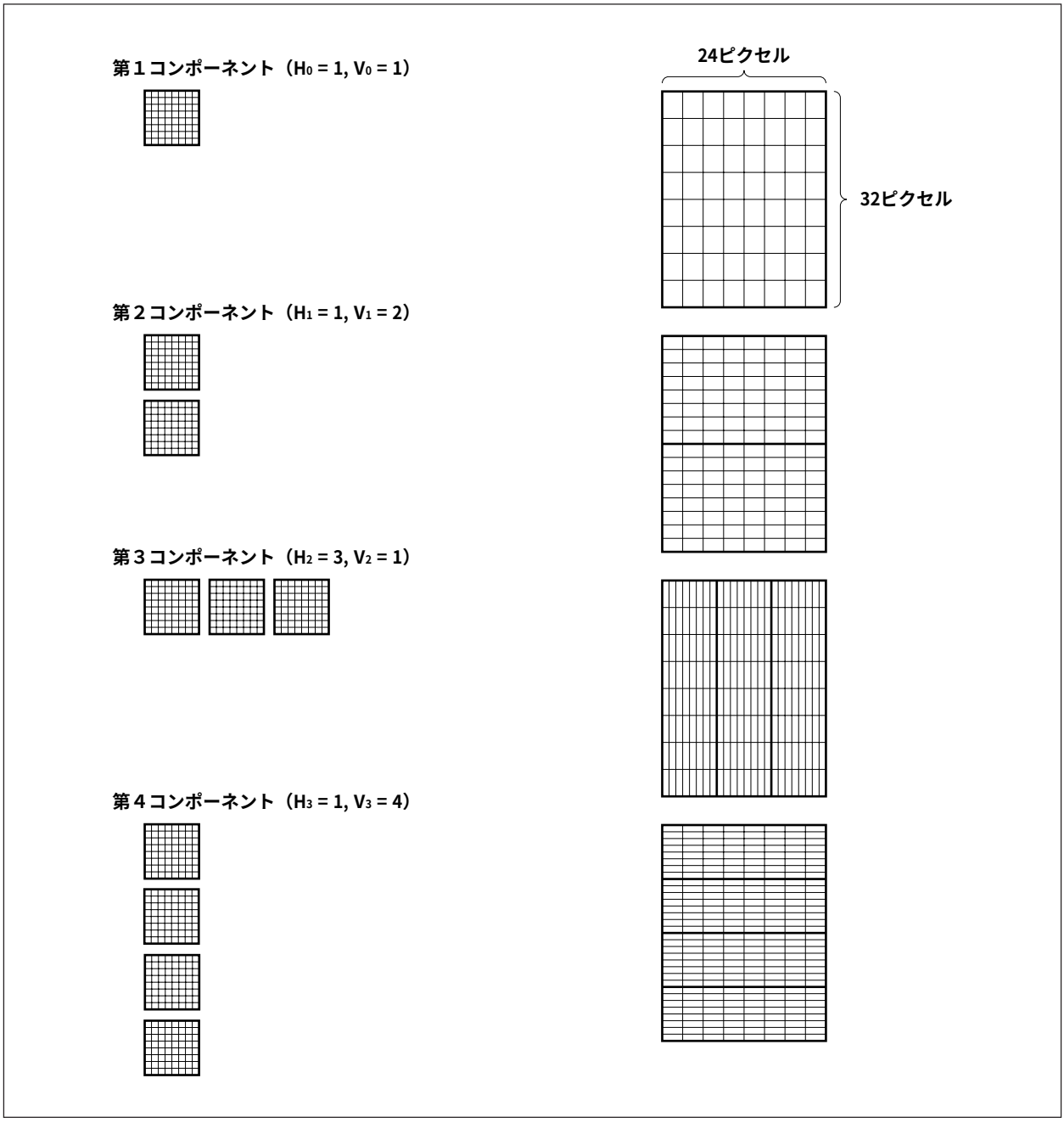
Max（H_0，H_1，H_2，H_3）×8 = 24

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３－２　サンプリングとMCU（サンプル比１：２：３：４の場合）



３.１.２　色空間について

AP703000-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＞255）R = 255；
G = M＋K；if（G＜0）G = 0；if（G＞255）G = 255；
B = Y＋K；if（B＜0）B = 0；if（B＞255）B = 255；

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 = 255-C1in；
Crout = 255-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。

３.１.３　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）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。

（３）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。

（４）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。

（５）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。

（６）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。

３.２　シンボル名規約

追加ライブラリ内で使用するシンボル名などは，次のとおりです。

表３－１　追加ライブラリのシンボル名規約

分　　類	説　　明
関数	先頭に「JPEGEX」を付加しています。
定数（参照専用ROMデータ）	先頭に「JPEGEX」を付加しています。
定数	先頭に「JPEGEX」を付加しています。
データ型	先頭に「JPEGEX」を付加しています。
セクション	先頭に「.JPD」を付加しています。

３.３　セクション

追加ライブラリが定義（予約）するセクションは次のとおりです。

表３－２　追加ライブラリが使用するセクション

処理	セクション名		属　性	マッピング		説　明
	NEC製	GHS製		ROM	RAM	
追加伸長処理	.JPD.text	.JPD.TEXT	.text	○	○	テキスト（命令コード）
	－	.JPDRODATA	.rodata	○	○	テーブル・データ（読み取り専用）
			.data	○	○	初期化ありデータ
	.JPD.const	－	.const	○	○	テーブル・データ（読み取り専用）
	.JPD.data	－	.data	×	○	初期化ありデータ

３.４　ライブラリの選択

追加ライブラリとして提供するファイルは，次のとおりです。

表３－３　追加ライブラリのファイル名

分　類	リンク優先順位	ファイル名	説　明	備　　考
追加伸長処理	1	libjprg.a	追加伸長処理メイン	追加伸長を使用する場合は，どちらかを必ず選択してください。
	2	libjprg_l.a	追加伸長処理メイン（long jump対応）	
	3	libjprog.a	シンボル解決用	追加伸長処理メインにlibjprg.aを使用する場合はlibjprog.aを，libjprg_l.aを使用する場合はlibjprog_l.aを選択してください。
	4	libjprog_l.a	シンボル解決用（long jump対応）	

３.５　関数仕様

追加ライブラリ処理では基本ライブラリで対応できなかった画像の伸長処理を行います。

なお，追加ライブラリは伸長処理のみで圧縮処理，解析処理は含まれてません。

表３－４　追加伸長処理関数

分　類	関数名	説　　明
追加伸長処理	JPEGEXdecode注１	追加伸長処理メイン関数
	JPEGEXGetJpegStream注２	JPEGデータ取得
	JPEGEXdecodeAPP注３	APPnセグメント処理
	JPEGEXWarning注３	ワーニング処理
	JPEGEXVSyncWait注３	表示タイミング調整処理
	JPEGEXputMCU注３	VRAM描画処理
	JPEGEXpset注３	ピクセル・データ出力

注１．この関数をコールする前に，JPEGEXINFO構造体，JPEGEXWORK構造体，JPEGEXVIDEO構造体，JPEGEXBUFF構造体のメンバ設定を行う必要があります。

２．必須カスタマイズ関数です。ユーザ・アプリケーションで必ず定義してください。

３．カスタマイズする場合のみ参照してください。

3.5.1　追加伸長処理

（1）JPEGEXdecode

分　　類　追加伸長処理

関 数 名　JPEGEXdecode

機能概要　JPEG伸長処理メイン

形　　式　#include “ jpegex.h ”

int JPEGEXdecode（struct JPEGEXINFO ＊*JPInfo*）

引 き 数

引き数	型	説　　明
JPInfo	JPEGEXINFO＊	JPEGEXINFO構造体[注]へのポインタ

注　JPEGEXINFO構造体については，**3.6.1　JPEGEXINFO構造体**を参照してください。

返 却 値

エラー・コード	説　明
JPEGEXDecodeStatusComplete	正常終了
JPEGEXDecodeStatusTerminate	強制終了
JPEGEXDecodeStatusError	エラー終了
JPEGEXDecodeStatusNotRunning	強制終了対象のプロセスは動作中ではない

機　　能　JPInfoで指定されるJPEG伸長を行います。

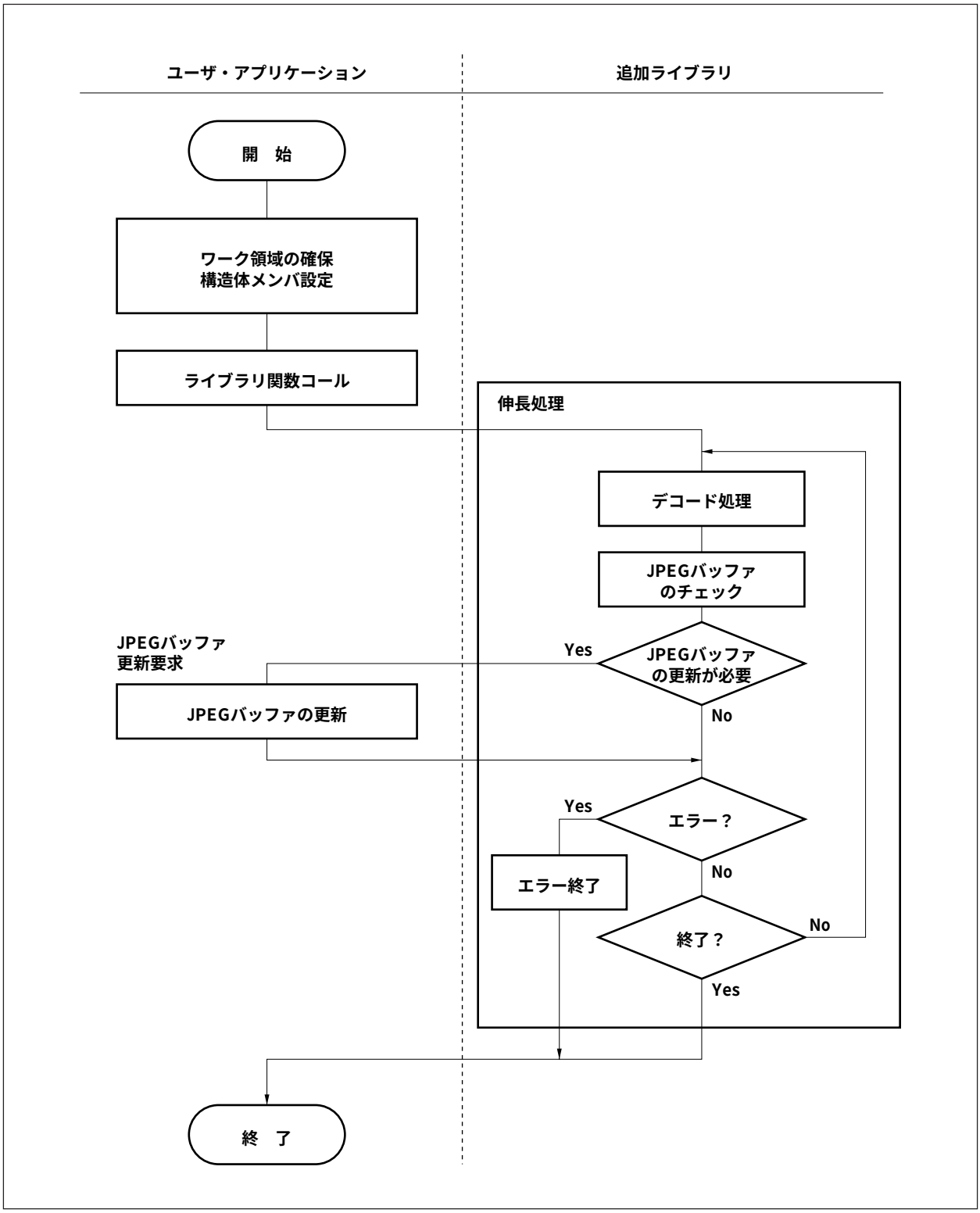
異常終了の場合，JPInfoのメンバ「ErrorState」で，エラー内容を確認してください（**3.7　追加伸長時のエラー内容**参照）。

制限事項　追加ライブラリでは16ビット量子化テーブルに対応していません。16ビットの量子化テーブルを持つJPEGファイルを伸長した場合，異常終了します。

3.5.2　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３－３　追加伸長処理フロー



（１）伸長パラメータの設定

ヘッダ・ファイル“ jpegex.h ”で定義されている４つの構造体（JPEGEXINFO型，JPEGEXWORK型，JPEGEXVIDEO型，JPEGEXBUFF型）の変数を定義し，メンバを設定してください（**３.６　追加ライブラリの構造体**を参照）。

新規に伸長処理を行う際には，必ず最初に設定してください。

構造体	メンバ	設定するデータ
JPEGEXINFO	TaskID	タスクのID番号
	Mode	通常伸長処理／伸長処理の強制終了
	Policy	伸長処理のオプション
	ratio	画像拡大／縮小率
	Work	JPEGEXWORK構造体アドレス
	Video	JPEGEXVIDEO構造体アドレス
	User1	ユーザ定義用メンバ
	User2	ユーザ定義用メンバ
JPEGEXWORK	Work1	ワーク・エリア１先頭アドレス
	Work1Len	ワーク・エリア１サイズ（バイト）
	Work2	ワーク・エリア２先頭アドレス
	Work2Len	ワーク・エリア２サイズ（バイト）
JPEGEXVIDEO	VRAMAddress	VRAM先頭アドレス
	VRAMWidth	VRAMの横幅
	VRAMHeight	VRAMの縦幅
	VRAMPixel	VRAMの横１ピクセルのアドレス差
	VRAMLine	VRAMの縦１ピクセルのアドレス差
	VRAMGap0	Yピクセル（またはRピクセル）のバイト・オフセット
	VRAMGap1	Cbピクセル（またはGピクセル）のバイト・オフセット
	VRAMGap2	Crピクセル（またはBピクセル）のバイト・オフセット
	ClipStartX	クリッピング開始位置（X座標）
	ClipStartY	クリッピング開始位置（Y座標）
	ClipWidth	クリッピング横サイズ（ピクセル）
	ClipHeight	クリッピング縦サイズ（ピクセル）

（２）伸長処理関数をコール〔JPEGEXdecode〕

「（１）伸長パラメータの設定」で設定した構造体のポインタを引き数に伸長処理関数をコールしてください。

（３）返却値のチェック

JPEGEXdecodeの返却値は次のとおりです。

返却値	意　味	説　　明
JPEGEXDecodeStatusComplete	正常終了	正常に終了しました。
JPEGEXDecodeStatusTerminate	強制終了	伸長処理を強制終了しました。
JPEGEXDecodeStatusError	異常終了	エラーを検出したため，処理を終了しました。 エラーの詳細は，JPEGEXINFO型構造体のメンバ（ErrorState）に格納されています。
JPEGEXDecodeStatusNotRunning	正常終了	強制終了対象のプロセスは動作中でないため，処理を終了しました。

注意　新規に伸長を行う場合は，再度（１）伸長パラメータの設定から始めてください。

伸長処理終了後，JPEGEXWORK構造体でワークの使用サイズが確認できます。

メンバ	格納されるデータ
Work1Used	使用ワーク・エリア１サイズ（バイト）
Work2Used	使用ワーク・エリア２サイズ（バイト）

３.５.３　カスタマイズ関数

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カスタマイズ関数と呼びます。カスタマイズ関数のうち，JPEGEXGetJpegStream関数は必ず上書きして定義しなければなりません。そのほかのカスタマイズ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。

（１）JPEGEXGetJpegStream

分　　類　追加伸長処理

関 数 名　JPEGEXGetJpegStream

機能概要　JPEGデータ取得

形　　式　#include “ jpegex.h ”

　　void　JPEGEXGetJpegStream（struct JPEGEXBUFF ＊JPBUFF）

引 き 数

引き数	型	説　　明
JPBUFF	JPEGEXBUFF＊	JPEGEXBUFF構造体[注]へのポインタ

注　JPEGEXBUFF構造体については，**３.６.４　JPEGEXBUFF構造体**を参照してください。

返 却 値　なし

機　　能　JPEGバッファにJPEGデータを格納します。

　　JPEXBUFF構造体のメンバJPEGBUFおよびJPEGBUFFLENに値を設定します。

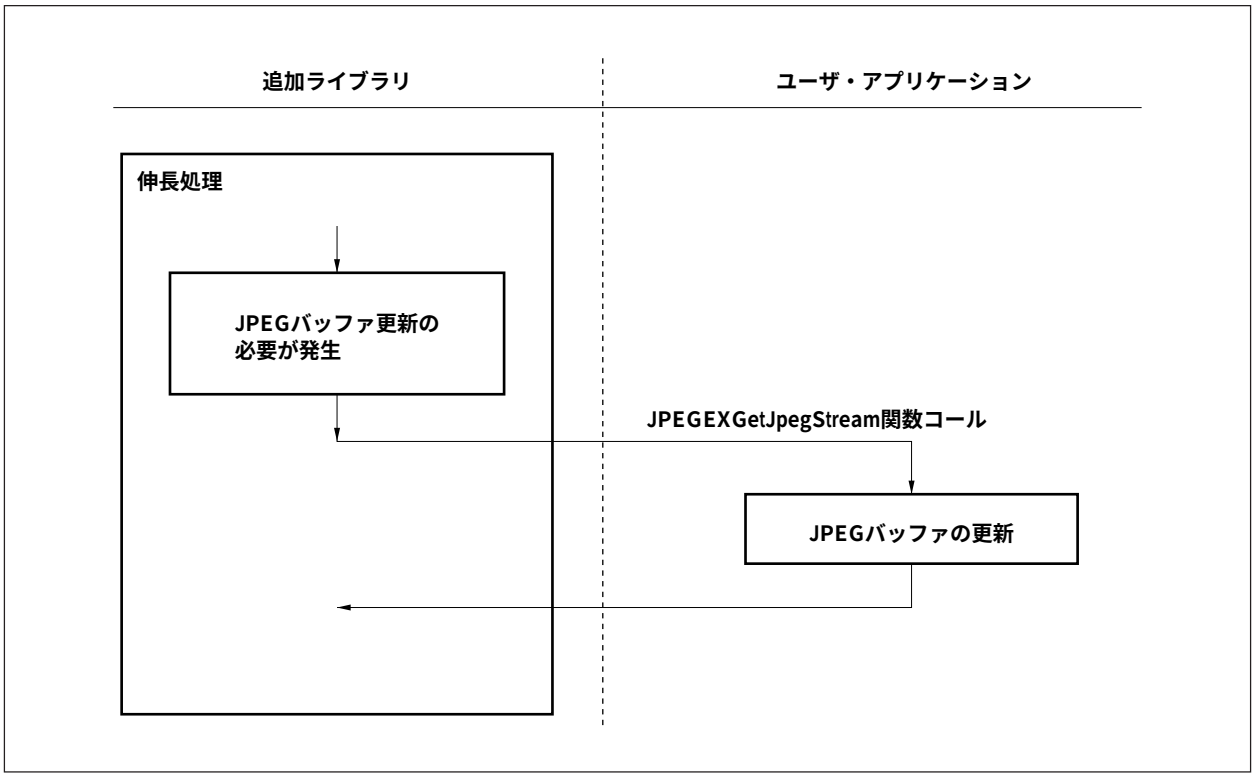
JPEGEXGetJpegStream関数は，ユーザ・アプリケーションで必ず定義してください。

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，JPEGEXGetJpegStream関数をコールします。

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・アプリケーション側で，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，**３.６.１（３）（ｄ）2passEnable（Tオプション）**を参照してください）。

図３－４　JPEGバッファの更新処理



（2）JPEGEXdecodeAPP

分　　類　追加伸長処理

関 数 名　JPEGEXdecodeAPP

機能概要　APPnセグメント処理

形　　式　#include “ jpegex.h ”

void JPEGEXdecodeAPP（int APPnumber, int JpegStreamOffset, int Length）

引 き 数

引き数	型	説　　明
APPnumber	int	0 ： APP0 （0xFF 0xE1） 1 ： APP1 （0xFF 0xE2） ： 15 ： APP15 （0xFF 0xEF） 16 ： COM （0xFF 0xFE）
JpegStreamOffset	int	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
Length	int	APPnセグメントの長さ（バイト数）

返 却 値　なし

機　　能　APPnセグメントの処理を行います。

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３－５　APPマーカ発見時の処理

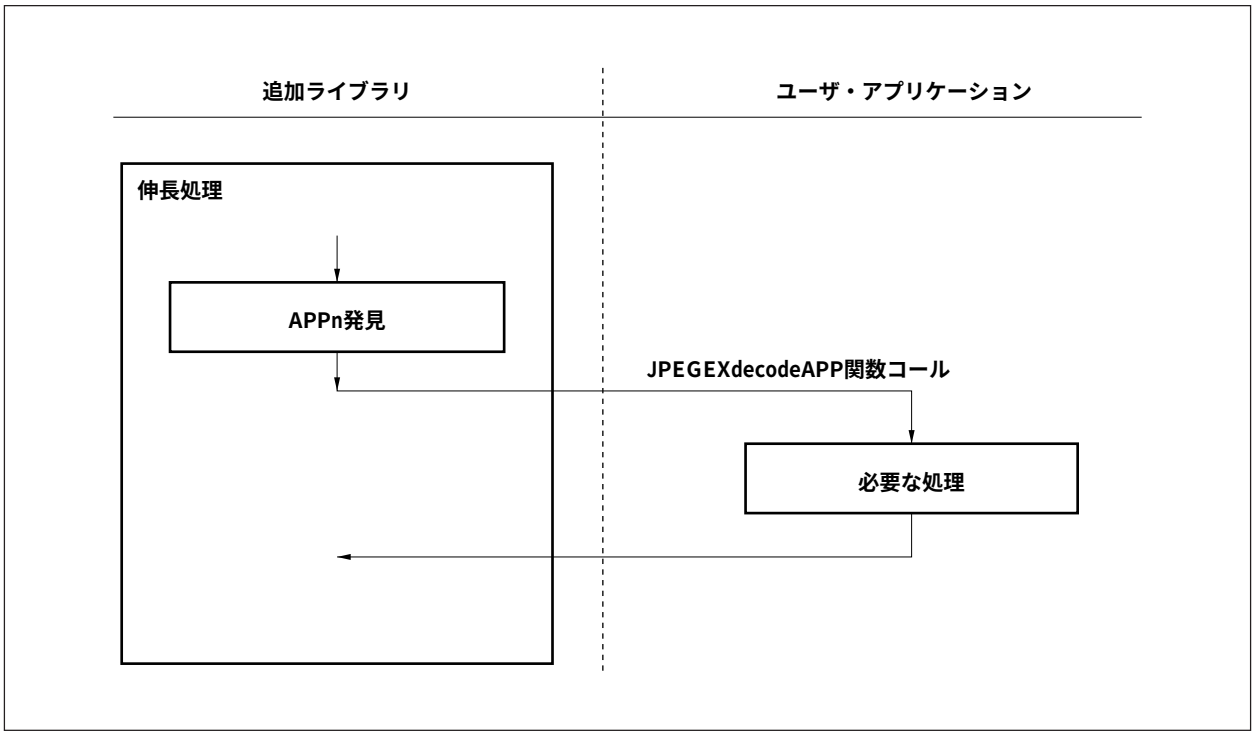
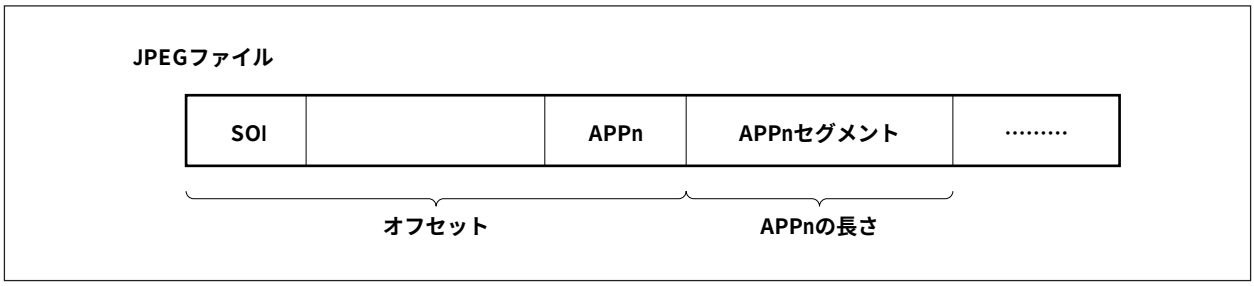


図３－６　APPnセグメントのオフセットと長さ



（３）JPEGEXWarning

分　　類　追加伸長処理

関 数 名　JPEGEXWarning

機能概要　ワーニング処理

形　　式　#include “ jpegex.h ”

　　void JPEGEXWarning（int WarningNumber）

引 き 数

引き数	型	説　　明
WarningNumber	int	ワーニング・メッセージ番号

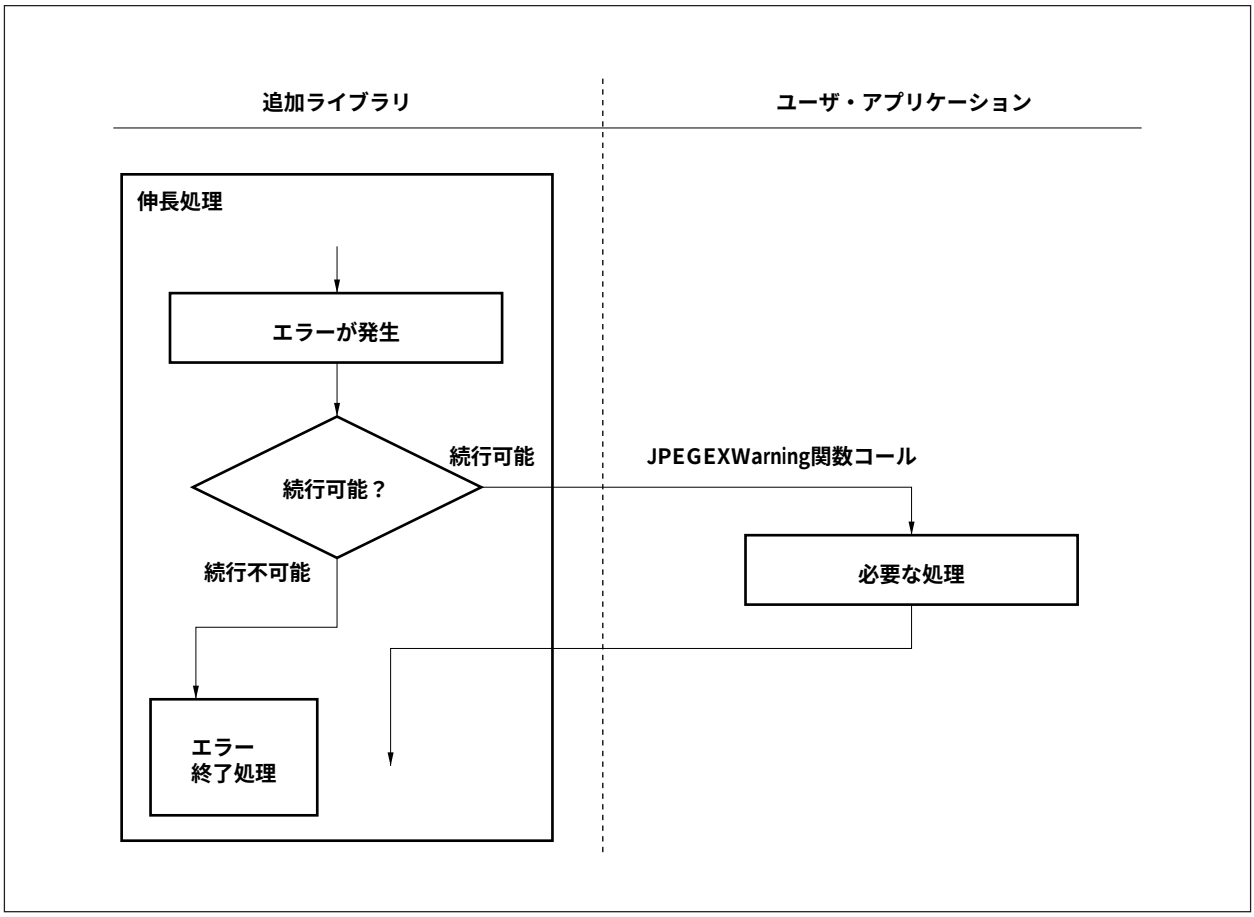
返 却 値　なし

機　　能　ワーニング時の処理を行います。

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**３．８　追加伸長時のワーニング内容**を参照してください。

図３－７　続行可能なエラー発生時の処理



（４）JPEGEXVSyncWait

分　　類　追加伸長処理

関 数 名　JPEGEXVSyncWait

機能概要　表示タイミング調整処理

形　　式　#include “ jpegex.h ”

　　void JPEGEXVSyncWait（void）

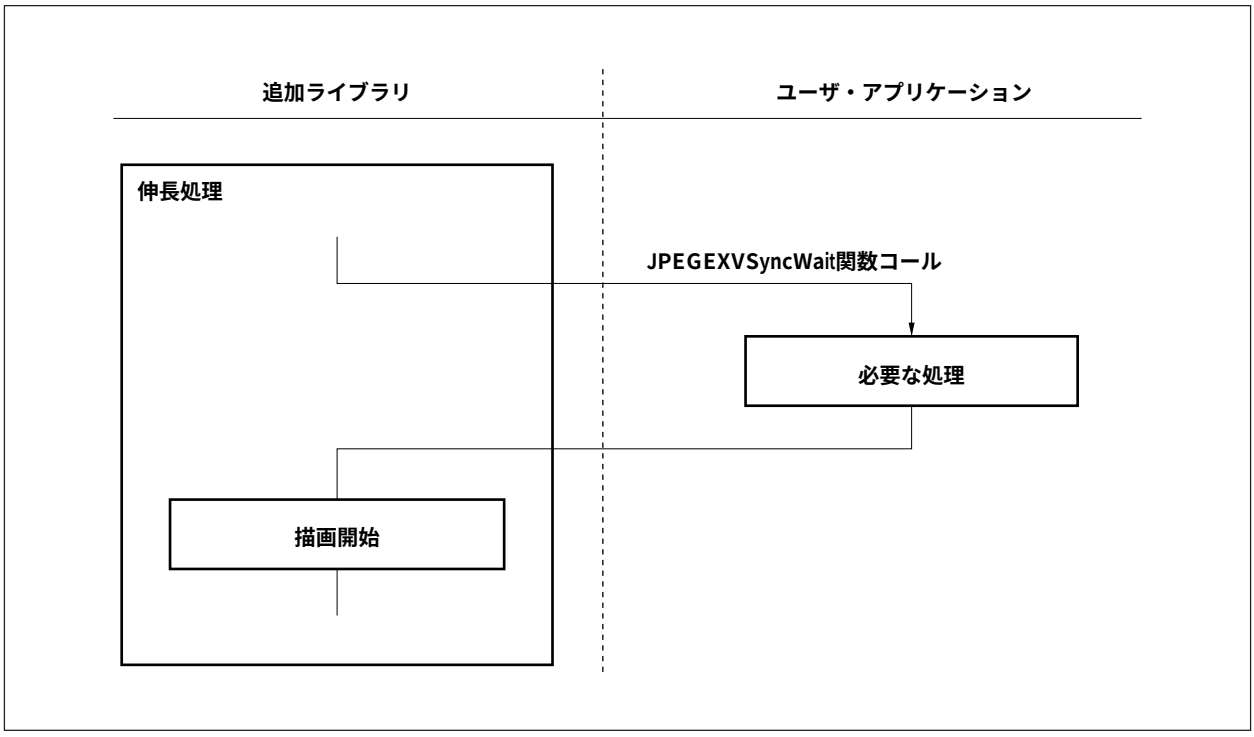
引 き 数　なし

返 却 値　なし

機　　能　表示タイミングの調整を行います。

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３－８　描画開始前の処理



（６）JPEGEXputMCU

分　　類　追加伸長処理

関 数 名　JPEGEXputMCU

機能概要　VRAM描画処理

形　　式　#include “ jpegex.h ”

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

引 き 数

引き数	型	説　　明
mcubuff	short＊	MCUバッファへのポインタ
Y	int	描画するMCUの左上隅の縦方向座標
X	int	描画するMCUの左上隅の横方向座標
MCUstr	struct JPEGEXMCUSTR＊	JPEGEXMCUSTR構造体の先頭アドレス

返 却 値　なし

機　　能　VRAMへ画像を表示します。

この関数はVRAMに画像を描画する関数です。伸長処理で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（**3.6.1（3）（h）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，**3.10　追加ライブラリのカスタマイズ**を参照してください。

第1引き数であるMCUバッファには，0-255の値がshort型で格納されています。詳しいMCUバッファの構造は，**2.5　MCUバッファの構造**を参照してください。なお４コンポーネントの場合は，**3.1.2　色空間について**もあわせて参照してください。

カスタマイズする際の画像メモリに関する設定は，次のようになります。

表３－５　画像描画関数カスタマイズ時のVRAM設定値

	構造体メンバ	カスタマイズ関数		
		JPEGEXputMCU	JPEGEXpset	両関数
JPEGEXINFO構造体	Video	必須	必須	必須
JPEGEXVRAM構造体	VRAMAddress	必須	ダミー値可	ダミー値可
	VRAMWidth	ダミー値可	必須	ダミー値可
	VRAMHeight	ダミー値可	必須	ダミー値可
	VRAMPixel	必須	ダミー値可	ダミー値可
	VRAMLine	必須	ダミー値可	ダミー値可
	VRAMGap0	必須	ダミー値可	ダミー値可
	VRAMGap1	必須	ダミー値可	ダミー値可
	VRAMGap2	必須	ダミー値可	ダミー値可
	ClipStartX	0	0	0
	ClipStartY	0	0	0
	ClipWidth	0x7FFFFFFF	0x7FFFFFFF	0x7FFFFFFF
	ClipHeight	0x7FFFFFFF	0x7FFFFFFF	0x7FFFFFFF

（７）JPEGEXpset

分　　類　追加伸長処理

関 数 名　JPEGEXpset

機能概要　ピクセル・データ出力

形　　式　#include “ jpegex.h ”

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出力方式の場合 ：

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

・RGB出力方式の場合

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

引 き 数

引き数	型	説　　明
MCUstr	struct JPEGEXMCUSTR＊	JPEGEXMCUSTR構造体の先頭アドレス
y	int	描画するピクセルのY座標
x	int	描画するピクセルのX座標
Cy , R	int	YCbCr出力の場合，Y色素データ RGB出力の場合，R色素データ
Cu , G	int	YCbCr出力の場合，Cb色素データ RGB出力の場合，G色素データ
Cv , B	int	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 却 値　なし

機　　能　VRAMへ画像を表示します。

この関数はVRAMに画像を描画する関数です。伸長処理で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。

カスタマイズを行う際は**表３－５　画像描画関数カスタマイズ時のVRAM設定値**を参照してください。

なお，変数Cy, Cu, Cvおよび変数，R, G, Bの値は0-255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;
}
```

３.６　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。

３.６.１　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。

表３－６　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	IN/OUT
User1	int	ユーザ定義用メンバ	IN/OUT
User2	int	ユーザ定義用メンバ	IN/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。

伸長処理を開始する前に強制終了モードを設定しないでください。

表３－７　追加伸長処理のモード設定

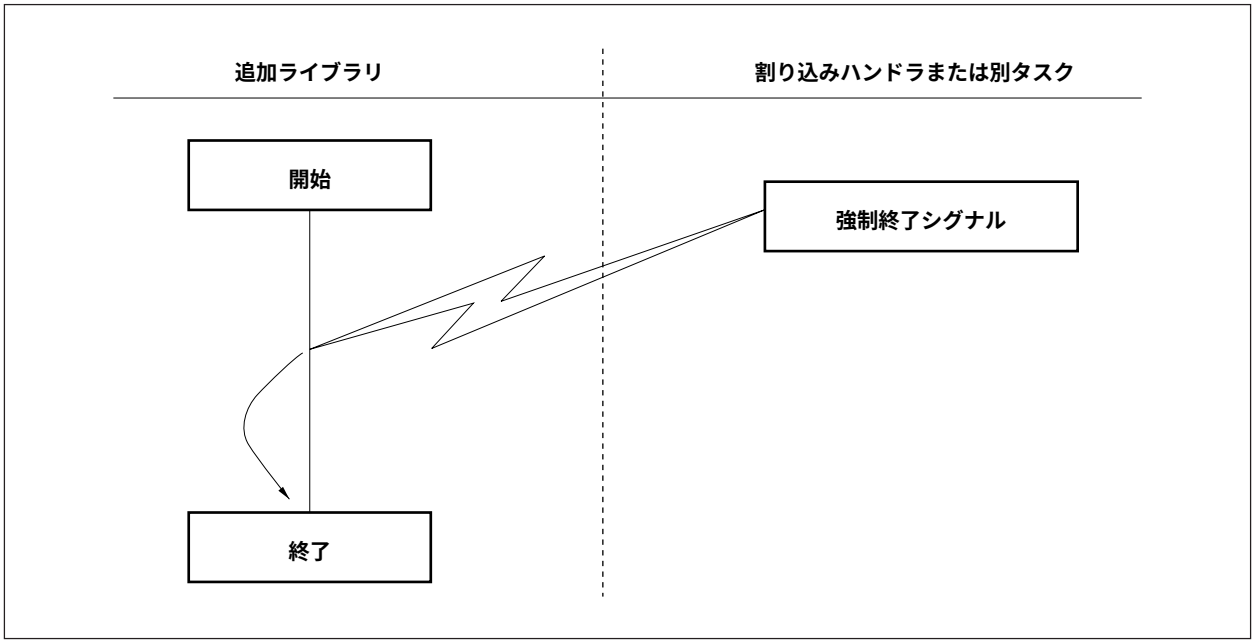
定義名	数値	内　　容
JPEGEXModeStart	1	通常の伸長モードです。
JPEGEXModeTerminate	－1	実行中の伸長処理を強制終了させます。

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

```
struct JPEGEXINFO JPINFO;
main ()
 {
          JPINFO.Mode = JPEGEXModeStart;
          JPEGEXdecode (&JPINFO);
 }
```

JPEGEXModeTerminate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。

図３－９　JPEGEXModeTerminate指定時の追加伸長処理強制終了



（３）Policy

Policyは図３－10に示す４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３－８に示すオプションの設定を行います。

図３－10　Policyのビット構成

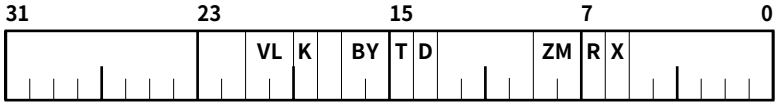


表３－８　Policyでのオプション設定

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（VideoOutLastOnly = 0のとき有効） 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（DNLEnable = 1のときは自動的に２パスで伸長） 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
ZM	9	VideoZoomLinear	画像の拡大／縮小（UseExPutMCU = 1のときは自動的に等倍で伸長） 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	
R	7	PutMCURGB	画像出力の設定（UseExPutMCU = 0のとき有効） 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

注意　この表で示されたビット以外の空きビットは，必ず０を設定してください。

（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３－11　ベースライン・フォーマットの描画タイミング

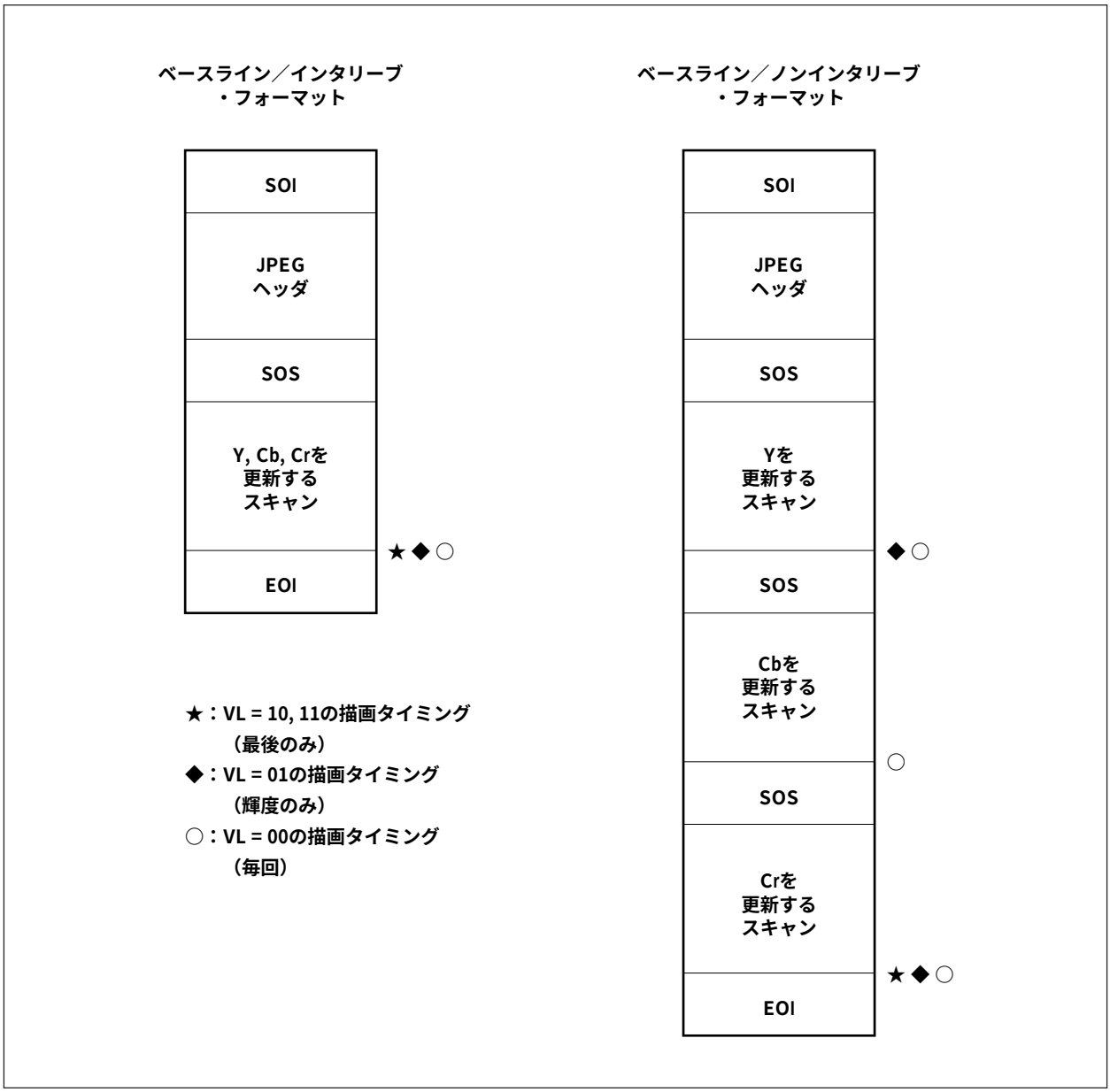
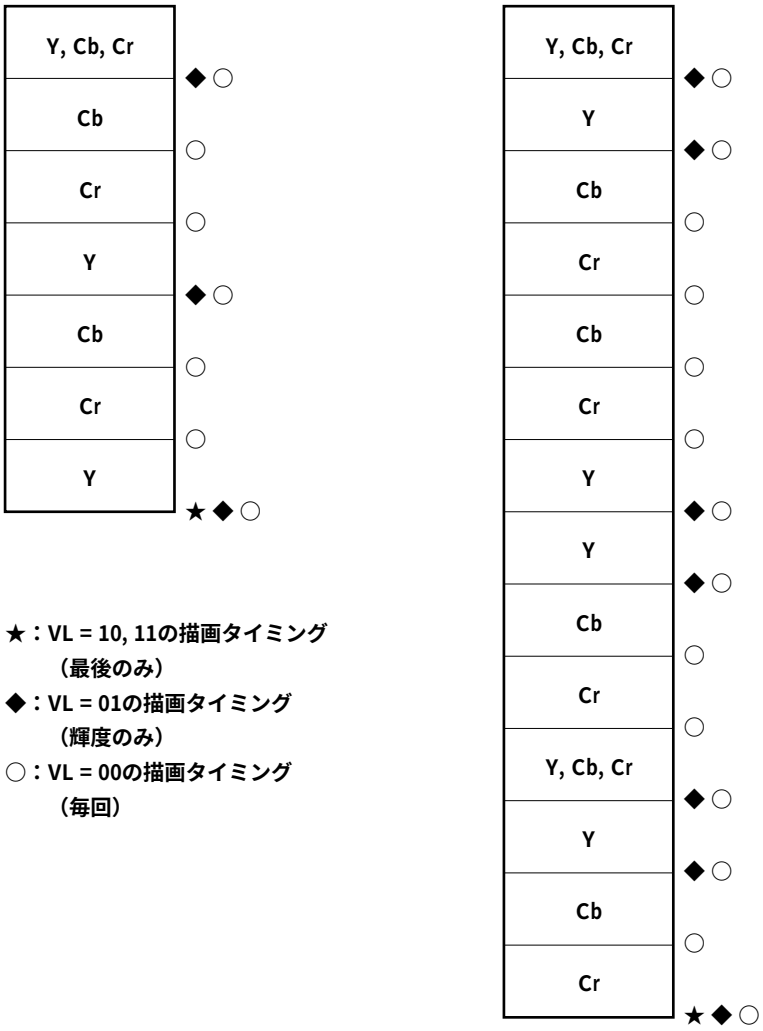


図３－12　プログレッシブ・フォーマット描画タイミング



（ｂ）BitStuffCheck（Kオプション）

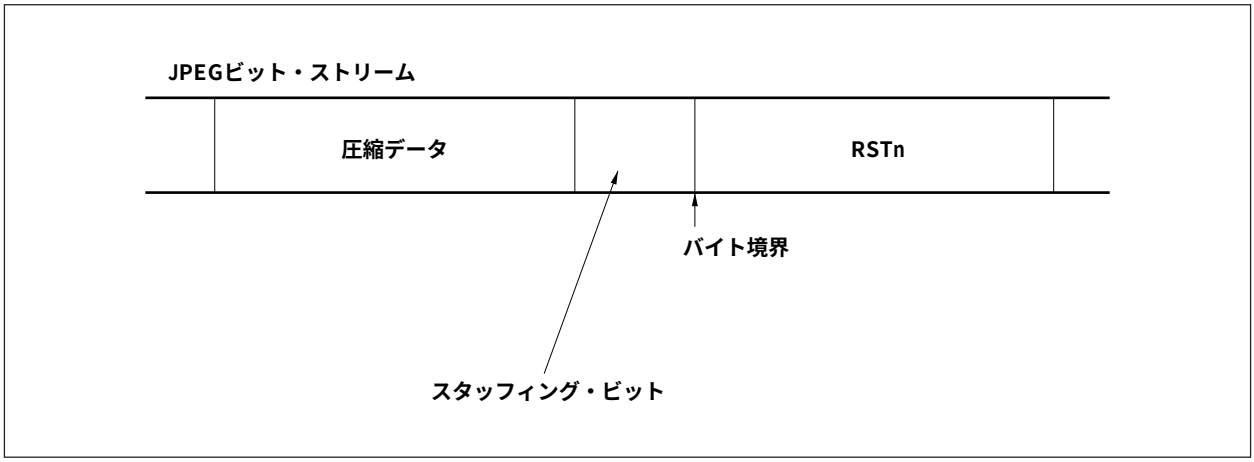
これはスタッフィング・ビットのチェックを行うかどうかを設定するオプションです。

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（**３.５.３（３）JPEGEXwarning**参照）。

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３－13　スタッフィング・ビット



（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３－９　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， 「マーカ・エラー：0x00001007」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，JPEGEXdecode関数の動作は不定です。
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（**3.6.1（３）（ｅ）DNLEnable（Dオプション）**参照）の場合には，無条件に２パスで伸長されます。

伸長処理の１パスと２パスの違いを表３－10に示します。

表３－10　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）

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。

ワーク・エリアの必要サイズ（単位はバイト）の計算式は次のとおりです。

・１パス・モード：

ワーク・エリア・サイズ

= 1408＋（128×1 MCUのブロック数）＋（128×量子化テーブルの数）
＋（208×DC成分用ハフマン・テーブルの数）＋（432×AC成分用ハフマン・テーブルの数）
＋（108×SOSマーカの個数）＋（128×伸長する画像の総ブロック数）

・２パス・モード：

ワーク・エリア・サイズ

= 1408＋（128×1 MCUのブロック数）＋（128×量子化テーブルの数）
＋（208×DC成分用ハフマン・テーブルの数）＋（432×AC成分用ハフマン・テーブルの数）
＋（404×SOSマーカの個数）

注意　クリッピングを使用して伸長する場合も，元の画像の総ブロック数を用いて計算してください。
ただし，次の条件を満たす時は１パス，２パスともにさらに次のサイズが必要となります。
・SOSマーカが15以上の場合：ワーク・エリア・サイズ＋（68×（SOSマーカの個数－14））

〔計算例1〕

次の条件で伸長する場合

・画像サイズ = 640×480（ベースライン形式）
・サンプル比 = 4：1：1（H：V = 2：2）
・量子化テーブル数 = 2
・DC成分用ハフマン・テーブルの数 = 2
・ AC成分用ハフマン・テーブルの数 = 2
・ SOSマーカの個数 = 1

サンプル比 4：1：1（H：V = 2：2）から画像のブロック数を算出すると次のようになります。

画像の総ブロック数 =（元画像の横方向のピクセル数×元画像の縦方向のピクセル数）
／（Y成分のMCUのピクセル数）×（1MCUのブロック数の合計）
=（640×480）／（16×16）×（4＋1＋1）
= 7200

したがってワーク・エリア・サイズは，次のようになります。

①　１パス・モード時

ワーク・エリア・サイズ

= 1408＋（128×（4＋1＋1））＋（128×2）＋（208×2）＋（432×2）＋（108×1）＋（128×7200）
= 1408＋768＋256＋416＋864＋108＋921600
= 925420バイト

② パス・モード時

ワーク・エリア・サイズ

= 1408＋（128×（4＋1＋1））＋（128×2）＋（208×2）＋（432×2）＋（404×1）

= 1408＋768＋256＋416＋864＋404

= 4116バイト

〔計算例２〕

次の条件で伸長する場合

・画像サイズ = 640×480（プログレッシブ形式）

・サンプル比 = 4：1：1（H：V = 2：2）

・量子化テーブル数 = 2

・DC成分用ハフマン・テーブルの数 = 2

・AC成分用ハフマン・テーブルの数 = 3

・SOSマーカの個数 = 7

サンプル比 4：1：1（H：V = 2：2）から画像のブロック数を算出すると次のようになります。

画像の総ブロック数 =（画像のピクセル数）／（Y成分のMCUのピクセル数）×（1MCUのブロック数の合計）

=（640×480）／（16×16）×（4＋1＋1）

= 7200

したがってワーク・エリアのサイズは，次のようになります。

① １パス・モード時

ワーク・エリア・サイズ

= 1408＋（128×（4＋1＋1））＋（128×2）＋（208×2）＋（432×3）＋（108×7）＋（128×7200）

= 1408＋768＋256＋416＋1296＋756＋921600

= 926500 バイト

② ２パス・モード時

ワーク・エリア・サイズ

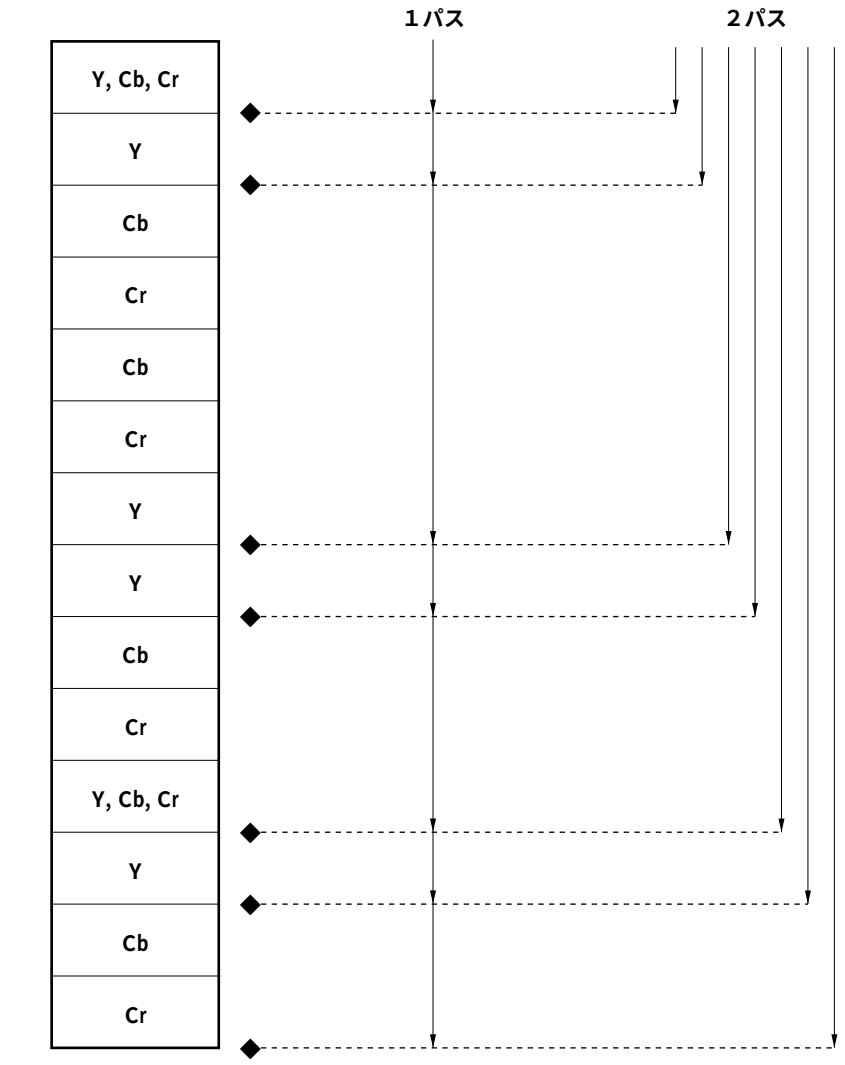
= 1408＋（128×（4＋1＋1））＋（128×2）＋（208×2）＋（432×3）＋（404×7）

= 1408＋768＋256＋416＋1296＋2828

= 6972バイト

備考　２パスの場合には，描画を行うたびにJPEGファイルの先頭が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。

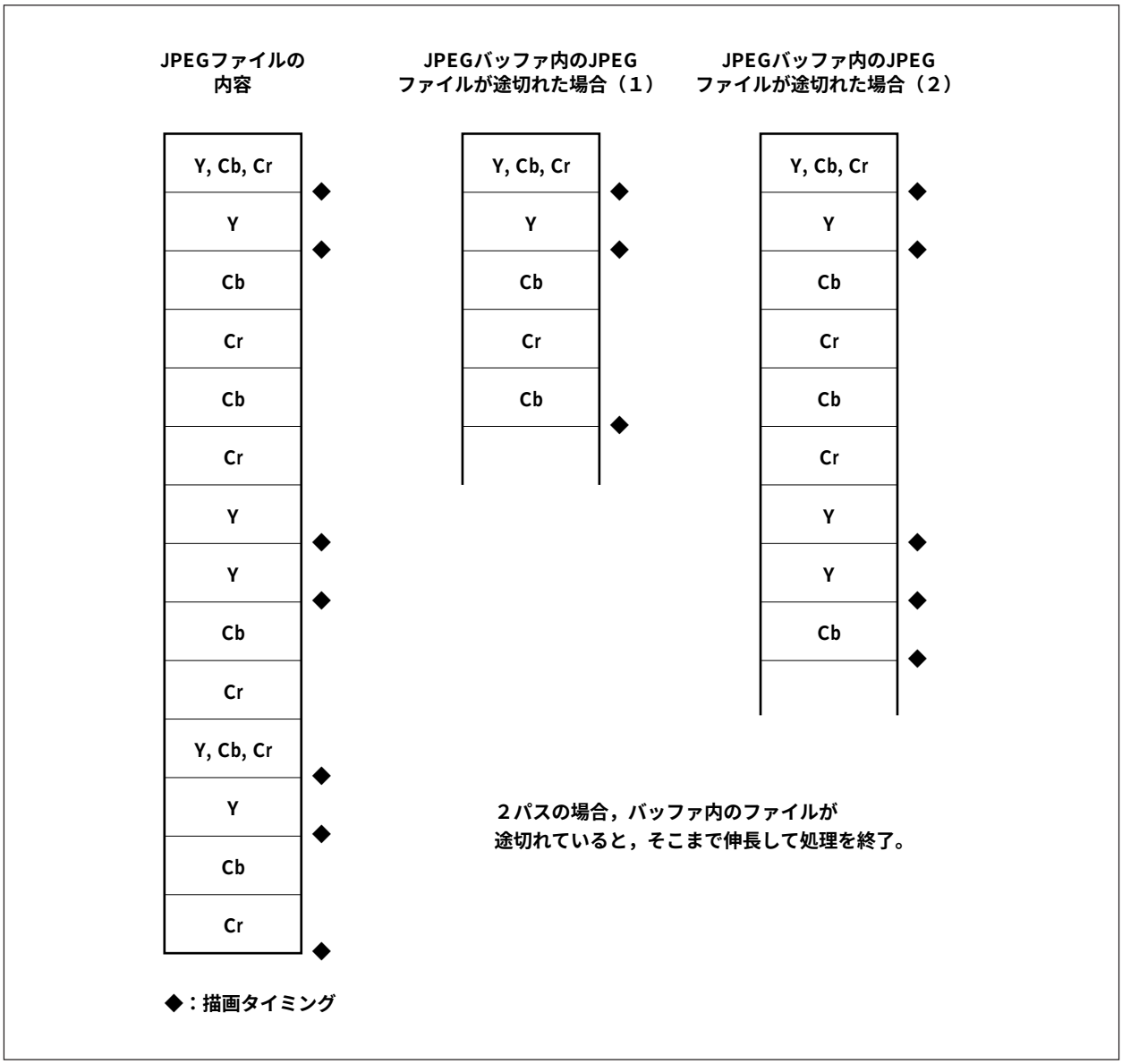
図３－14　追加伸長処理のパス回数と描画タイミング



◆：描画タイミング

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（**３.６.４（３）JPEGBUFFLEN**参照）。

図３－15　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）



（ｅ）DNLEnable（Dオプション）

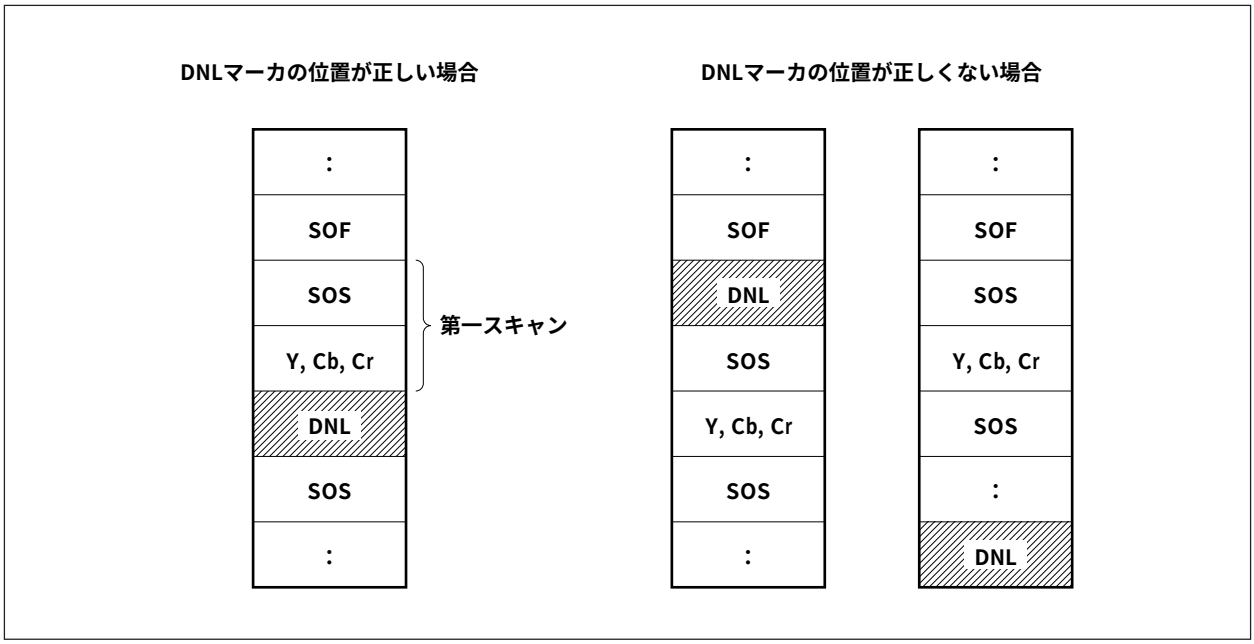
これは，DNLマーカ（ライン数の再定義）を含むJPEGファイルの伸長の可否を設定するオプションです。DNLEnable = 1のとき，DNLマーカを含むJPEGファイルを伸長できます。DNLEnable = 0のとき，DNLマーカを含むJPEGファイルを伸長すると異常終了（ErrorState：0x0000101C）します。

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３－16　JPEGファイル内のDNLマーカの位置



（ｆ）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。ZMオプションを設定する場合（拡大／伸長を行う場合）には，UseExPutMCUオプションはセットしないでください。UseExPutMCU = 1とした場合，ZMオプションは無視され等倍で伸長します。

表３－11　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。

（ｇ）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。

PutMCURGBは，UseExPutMCU = 0のとき有効となります。

（ｈ）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，**３.10　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。

VideoZoomLinear/VideoZoomNormalオプションでZM = 00以外を用いる場合，必ずUseExPutMCUオプションはUseExPutMCU = 0としてください。UseExPutMCU = 1とした場合は，JPEGEXputMCU関数をコールします。

（４）ratio

JPEGEXINFO構造体のメンバPolicyで画像拡大／伸長オプション（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。

ratioに設定できる値の範囲は次のとおりです。

（ａ）通常拡大／伸長（VideoZoomNormal設定時）の場合

ratioが必ず次の式を満たすように設定してください。

1≦ratio≦65536

（ｂ）線形一次補間（VideoZoomLinear設定時）拡大／伸長の場合

ratioが必ず次の式を満たすように設定してください。

1≦ratio≦256

（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**３.７　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（**３.６.２　JPEGEXWORK構造体**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（**３.６.３　JPEGEXVIDEO構造体**参照）。

（８）Inf

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。

伸長モードでライブラリをコールする際には，初期値０を設定してください。

３.６.２　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。

表３－12　JPEGEXWORK構造体

メンバ	型	内　　容	IN/OUT
Work1	unsigned int	ワーク・エリア１先頭アドレス	IN
Work1Len	unsigned int	ワーク・エリア１サイズ（バイト数）	IN
Work1Used	unsigned int	使用ワーク・エリア１サイズ（バイト数）	OUT
Work2	unsigned int	ワーク・エリア２先頭アドレス	IN
Work2Len	unsigned int	ワーク・エリア２サイズ（バイト数）	IN
Work2Used	unsigned int	使用ワーク・エリア２サイズ（バイト数）	OUT

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，または両方に，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，追加ライブラリではワーク・エリア１を優先しますので，ワーク・エリア１を高速メモリにマッピングすることを推奨します。

３.６.３　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３－13に示すダミー値を設定してください。

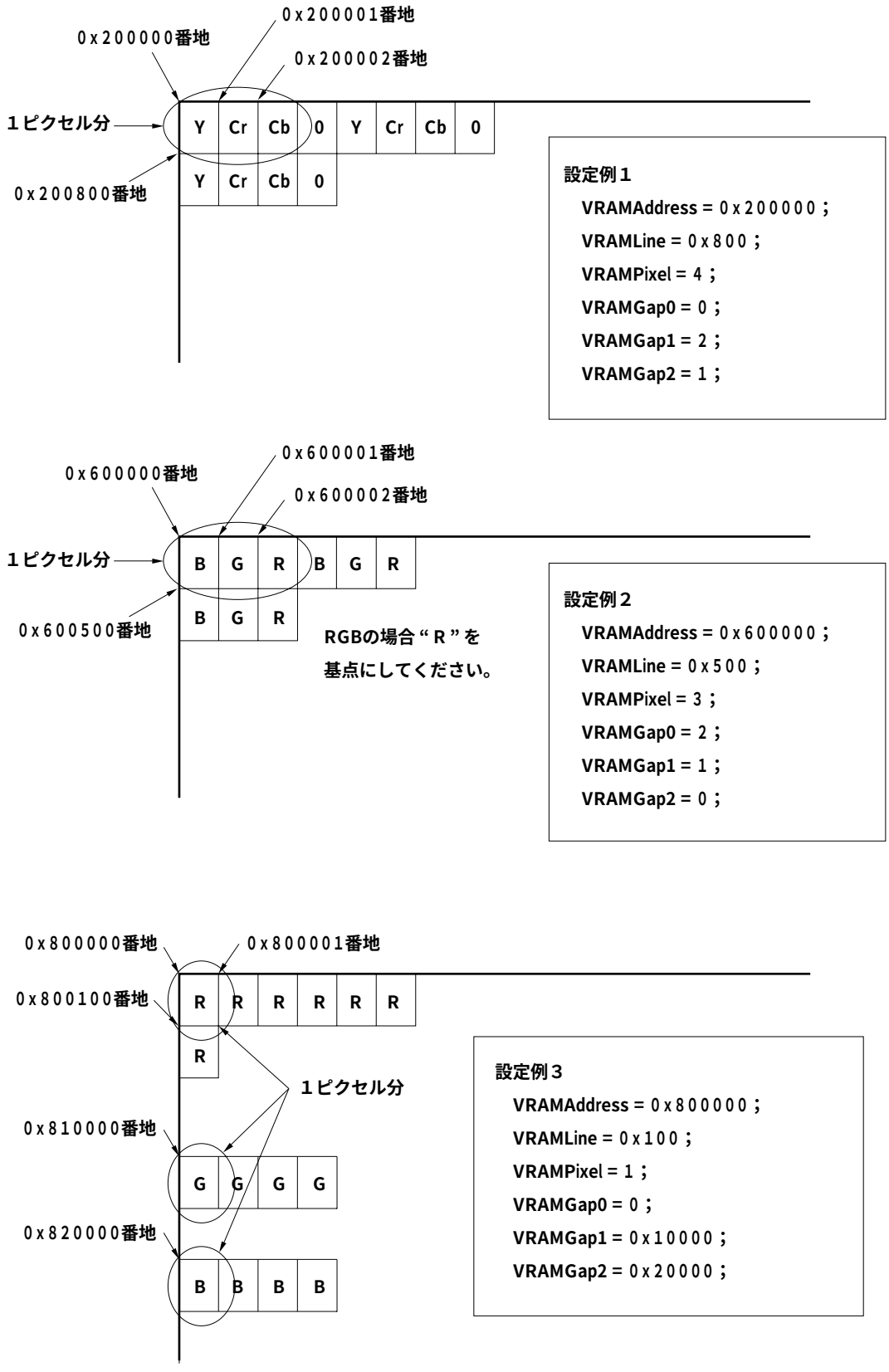
表３－13　JPEGEXVIDEO構造体

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

（１）VRAM構成

VRAM関連メンバ（VRAMxxx）の設定例を図３－17に示します。

図３－17　追加ライブラリのVRAM関連メンバ設定例



（2）クリッピングの設定

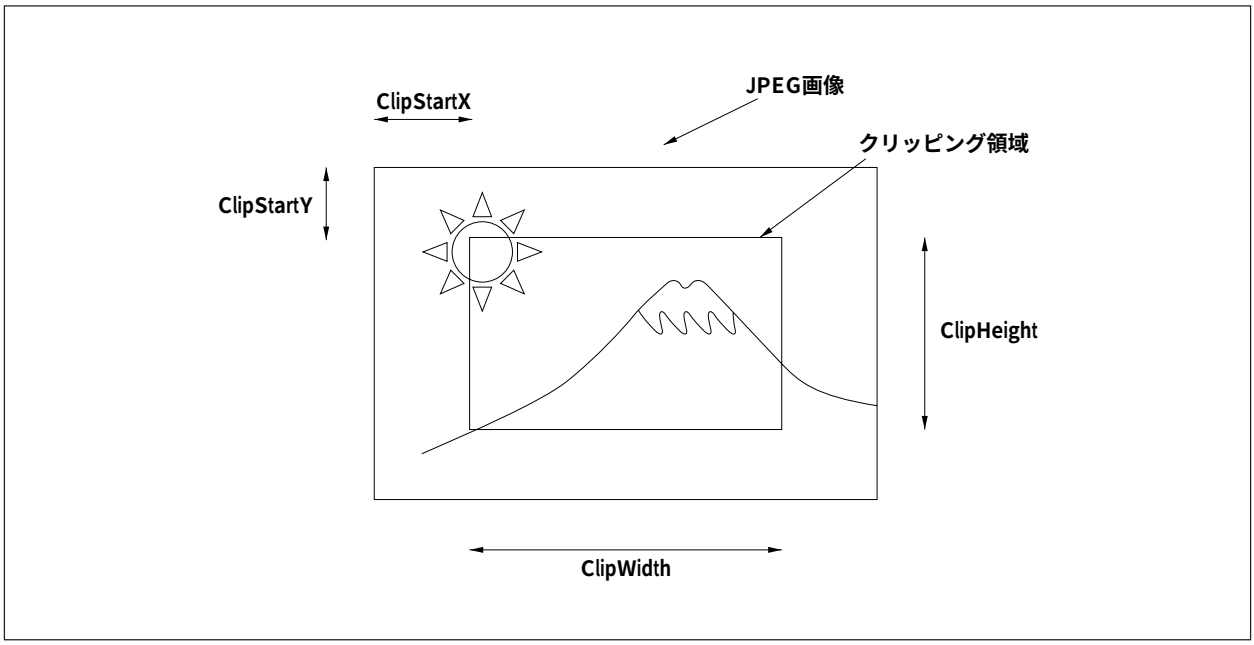
追加伸長時のクリッピングは，クリッピング関連メンバ（Clipxxx）にダミー値以外の値を設定した場合に行われます。クリッピングを行わない場合はクリッピング関連のメンバに表３－13に示すダミー値を代入してください。

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３－18に示します。

ClipStartX, ClipStartYにはJPEG画像の左上を原点（0 , 0）とする座標を設定してください。ClipWidth, ClipHeightにはクリッピングするピクセル数を設定してください。

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３－18　クリッピング領域

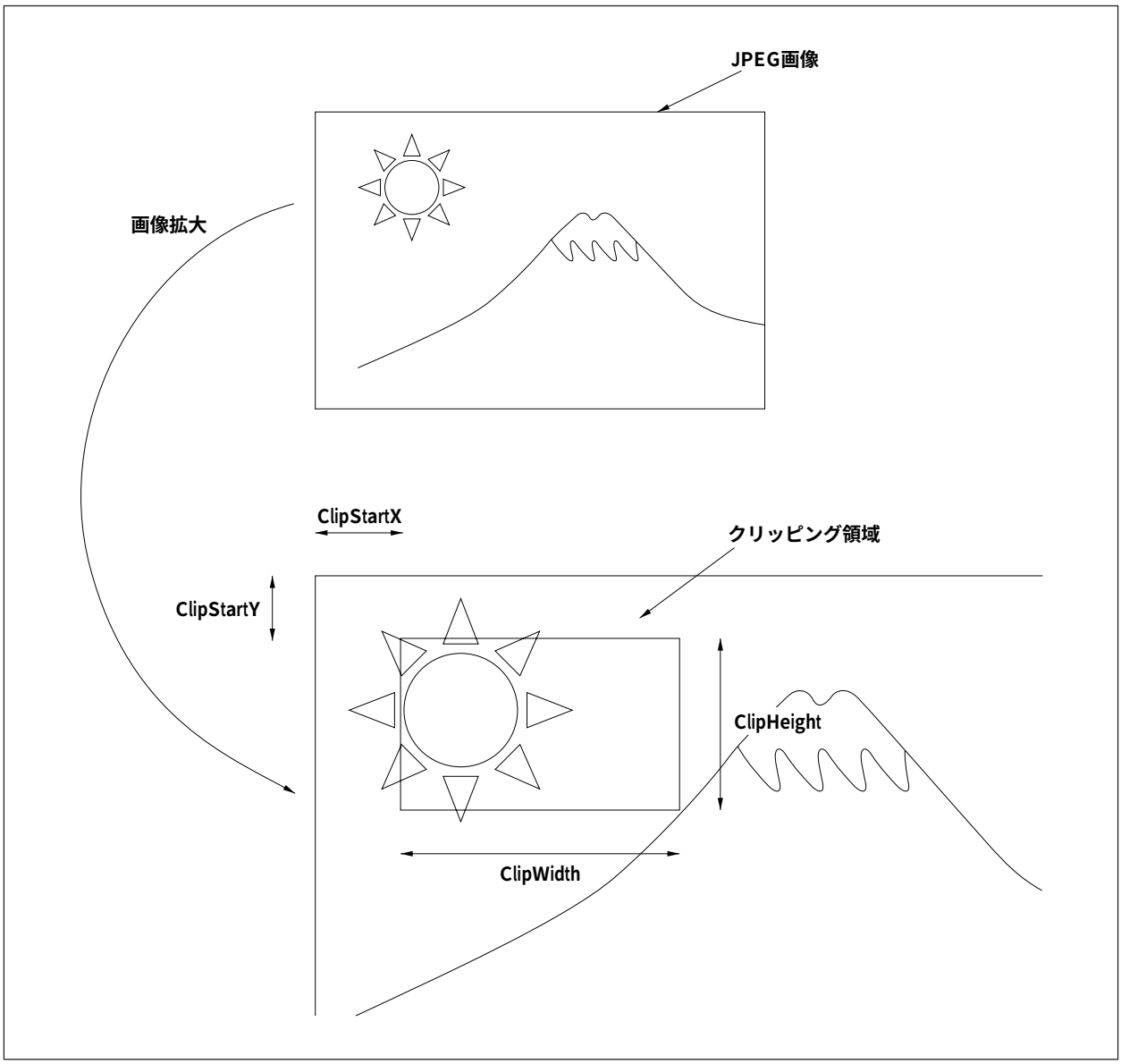


（３）クリッピング設定と拡大／縮小設定との関係

JPEGEXINFO構造体メンバPolicyのオプション（３.６.１（３）（ｆ）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３－19に拡大／縮小伸長時のクリッピング領域を示します。

図３－19　拡大／縮小伸長時のクリッピング領域



３.６.４　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。

表３－14　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（３.６.１（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。

２パス・モードで伸長する場合には，JPEGBUFFLENに伸長するJPEGファイル・サイズ以上の値を設定してください。JPEGファイル・サイズよりJPEGバッファ・サイズが小さい場合，追加ライブラリの動作は不定（正常，もしくは異常終了）です。

１パス・モードで伸長する場合にはJPEGバッファ・サイズはできるだけ大きな値（推奨値は，伸長するJPEGファイルと同じサイズです）を設定してください。

JPEGファイル・サイズよりJPEGバッファ・サイズが小さい場合，JPEGEXdecode関数はJPEGファイル取得関数（３.５.３（１）JPEGEXGetJpegStream参照）をコールし，JPEGバッファを更新します。

JPEGバッファのサイズが32バイト以下の場合，追加ライブラリは異常終了（ErrorState：0x00001006）します。

３.６.５　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３－15　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

３.７　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３－16に示します。

表３－16　追加ライブラリのエラー内容（1/2）

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	マルチスキャンのフォーマットが不正なため，伸長できません。
ErrorMultiDQT	0x1004	第１スキャン以降に同じ番号の量子化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８以外です。
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	DNLセグメントに誤りがあります。 DNLマーカは許可されていません。
ErrorRSTsegment	0x101D	RSTnマーカに誤りがあります。
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。

表３－16　追加ライブラリのエラー内容（2/2）

エラー・メッセージ	番号	エラー内容
ErrorSOIMarker2	0x1021	SOIマーカが複数定義されています。
ErrorDQTelement	0x1022	DQTセグメントに記述されたテーブル要素がJPEG規格外です。
ErrorDHTelement	0x1023	DHTセグメントに記述されたテーブル要素がJPEG規格外です。
ErrorImageX	0x1024	SOFセグメントに記述された画像横ピクセル数がJPEG規格外です。
ErrorSOFsegmentID2	0x1025	SOFセグメントに重複するコンポーネントIDが定義されています。
ErrorSOSMarker	0x1026	SOSセグメントが定義されていません。
ErrorSOSsegmentID2	0x1027	SOSセグメントに重複するコンポーネントIDが定義されています。
ErrorSOSsegmentSS	0x1028	SOSセグメントに記述されたSsがJPEG規格外です。
ErrorSOSsegmentSA	0x1029	圧縮データ情報が重複しています。
ErrorDataNotEnough	0x102A	reserve
ErrorSOF0segmentP	0x102B	サンプル・ビット精度が12ビットになっています（ベースライン）。
ErrorImageY	0x102C	SOFまたはDNLセグメントに記述された画像縦ピクセル数がJPEG規格外です。
ErrorDNLMarker2	0x102E	DNLセグメントが複数定義されています。
ErrorSOSnotAlloc	0x102F	reserve

３.８　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１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３－17に示します。

表３－17　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。
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
WarningSOFYDNL	0x2009	縦ピクセル数に０以外の値が設定されていますが，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。

３.９　追加ライブラリ定数一覧

追加ライブラリで定義している（jpegex.h）定数の一覧を示します。

表３－18　追加ライブラリ定数（1/3）

シンボル	値	IN/OUT	意　　味
JPEGEXModeStart	0x00000001	IN	通常伸長処理モード（JPEGEXINFO構造体Mode）
JPEGEXModeTerminate	0xFFFFFFFF	IN	強制終了モード（JPEGEXINFO構造体Mode）
JPEGEXVideoOutLastOnly	0x00200000	IN	プログレッシブJPEG伸長時，最終画像のみ表示 （JPEGEXINFO構造体Policy）
JPEGEXLuminanceOutOnly	0x00100000	IN	プログレッシブJPEG伸長時，輝度成分ごとに表示 （JPEGEXINFO構造体Policy）
JPEGEXBitStuffCheck	0x00080000	IN	スタッフィング・ビット・チェック（JPEGEXINFO構造体Policy）
JPEGEXByteStuffDisable	0x00020000	IN	スタッフィング・バイト許可（JPEGEXINFO構造体Policy）
JPEGEXByteStuffEnable	0x00010000	IN	スタッフィング・バイト0x10000バイトまで許容 （JPEGEXINFO構造体Policy）
JPEGEX2passEnable	0x00008000	IN	2パス・モード（JPEGEXINFO構造体Policy）
JPEGEXDNLEnable	0x00004000	IN	DNLマーカ許可（JPEGEXINFO構造体Policy）
JPEGEXUsePset	0x00000800	IN	ユーザ作成のJPEGEXpset関数使用 （JPEGEXINFO構造体Policy）
JPEGEXVideoZoomLinear	0x00000200	IN	線形フィルタによる拡大／縮小 （JPEGEXINFO構造体Policy）
JPEGEXVideoZoomNormal	0x00000100	IN	拡大／縮小（JPEGEXINFO構造体Policy）
JPEGEXPutMCURGB	0x00000080	IN	RGB出力（JPEGEXINFO構造体Policy）
JPEGEXUseExPutMCU	0x00000040	IN	ユーザ作成のJPEGEXputMCU関数使用 （JPEGEXINFO構造体Policy）
JPEGEXWarningBitStuff	0x00002000	OUT	スタッフィング・ビットが不正 （JPEGEXWarning（）引き数）
JPEGEXWarningProgACInterleave	0x00002001	OUT	プログレッシブJPEGのAC係数がインタリーブ （JPEGEXWarning（）引き数）
JPEGEXWarningBlock20	0x00002002	OUT	１MCUブロック数が20以上（JPEGEXWarning（）引き数）
JPEGEXWarningBlock10	0x00002003	OUT	１MCUブロック数が10以上（JPEGEXWarning（）引き数）
JPEGEXWarningDHTbaselineTh	0x00002005	OUT	ベースラインJPEGでハフマンテーブル番号に2, 3を使用 （JPEGEXWarning（）引き数）
JPEGEXWarningAdobe	0x00002006	OUT	APP14セグメント内の空間識別子の値が0, 1, 2以外 （JPEGEXWarning（）引き数）
JPEGEXWarningSOFYDNL	0x00002009	OUT	縦ピクセル数に0以外の値が設定されている （JPEGEXWarning（）引き数）
JPEGEXDecodeStatusComplete	0x00000001	OUT	正常終了（JPEGEXdecode返却値）
JPEGEXDecodeStatusTerminate	0x00000002	OUT	強制終了（JPEGEXdecode返却値）
JPEGEXDecodeStatusError	0xFFFFFFFF	OUT	異常終了（JPEGEXdecode返却値）
JPEGEXDecodeStatusNotRunning	0xFFFFFFFE	OUT	強制終了対象のプロセスは動作中でない （JPEGEXdecode返却値）
JPEGEXErrorMode	0x00001000	OUT	伸長モード・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorAllocate	0x00001001	OUT	ワーク・エリア・サイズ不足 （JPEGEXINFO構造体 ErrorState）

表３－18　追加ライブラリ定数（2/3）

シンボル	値	IN/OUT	意　　味
JPEGEXErrorMultiFrame	0x00001002	OUT	マルチ・フレーム・フォーマット未対応 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorMultiScan	0x00001003	OUT	マルチ・スキャン・フォーマット・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorMultiDQT	0x00001004	OUT	量子化テーブル多重定義 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorJPEGBuffLen	0x00001006	OUT	JPEGバッファ・サイズ不足 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorJPEGMarker	0x00001007	OUT	JPEGマーカ・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOIMarker	0x00001008	OUT	SOIマーカなし（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDQTsegment	0x00001009	OUT	DQTセグメント・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDQTsegmentTq	0x0000100A	OUT	DQTセグメントの量子化テーブル番号エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDQTsegmentPq	0x0000100B	OUT	DQTセグメントの量子化テーブル精度エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOFsegment	0x0000100C	OUT	SOFセグメント・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOFsegmentNf	0x0000100D	OUT	SOFセグメントのコンポーネント数エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOFsegmentSF	0x0000100E	OUT	SOFセグメントのサンプリング・ファクタ・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOFsegmentTq	0x0000100F	OUT	SOFセグメントの量子化テーブル番号エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDHTsegment	0x00001010	OUT	DHTセグメント・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDHTsegmentTc	0x00001011	OUT	DHTセグメントのハフマン・テーブル・クラス・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDHTsegmentTh	0x00001012	OUT	DHTセグメントのハフマン・テーブル番号エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegment	0x00001014	OUT	SOSセグメント・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentCi	0x00001015	OUT	コンポーネントID未定義 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentTq	0x00001016	OUT	量子化テーブル未定義（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentTa	0x00001017	OUT	AC成分用ハフマン・テーブル未定義 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentTd	0x00001018	OUT	DC成分用ハフマン・テーブル未定義 （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDCcode	0x00001019	OUT	DC成分ハフマン・デコード・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorACcode	0x0000101A	OUT	AC成分ハフマン・デコード・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorHuffcode	0x0000101B	OUT	圧縮データ中にマーカ（RST以外）出現 （JPEGEXINFO構造体 ErrorState）

表３－18　追加ライブラリ定数（3/3）

シンボル	値	IN/OUT	意　　味
JPEGEXErrorDNLsegment	0x0000101C	OUT	DNLマーカ無許可，またはDNLセグメント・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorRSTsegment	0x0000101D	OUT	RSTマーカ・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDRIsegment	0x0000101E	OUT	DRIセグメント・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDNLnot1stScan	0x0000101F	OUT	DNLセグメント位置エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOIMarker2	0x00001021	OUT	SOIマーカ多重定義（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDQTelement	0x00001022	OUT	DQTセグメント要素エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDHTelement	0x00001023	OUT	DHTセグメント要素エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorImageX	0x00001024	OUT	画像横サイズ・エラー （JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOFsegmentID2	0x00001025	OUT	SOFセグメントのコンポーネントID重複定義（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSMarker	0x00001026	OUT	SOSセグメントなし（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentID2	0x00001027	OUT	SOSセグメント・コンポーネントID重複定義（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentSS	0x00001028	OUT	SOSセグメント・スペクトラル・セレクション・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSsegmentSA	0x00001029	OUT	SOSセグメント・サクセッシブ・アプロキシメーション・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDataNotEnough	0x0000102A	OUT	reserve
JPEGEXErrorSOF0segmentP	0x0000102B	OUT	SOF0セグメント・サンプル・ビット精度エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorImageY	0x0000102C	OUT	画像縦サイズ・エラー（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorDNLMarker2	0x0000102E	OUT	DNLセグメント多重定義（JPEGEXINFO構造体 ErrorState）
JPEGEXErrorSOSnotAlloc	0x0000102F	OUT	reserve

３.10　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）。

３.10.１　簡易的なカスタマイズ

JPEGEXpset関数をカスタマイズすることにより，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，**３.５.３（７）JPEGEXpset**を参照してください。

JPEGEXpset関数をカスタマイズした場合，ディフォールトのJPEGEXpset関数は使われません。

３.10.２　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，（**３.５.３（６）JPEGEXputMCU**）を参照してください。

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。JPEGEXputMCU関数をカスタマイズして使用する場合には，PolicyのUseExPutMCUオプションを１にセットしてください（**３.６.１（３）Policy**参照）。

３.10.３　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。

```
#include "jpegex.h"
#defineR_Cr     (int)(1.40200*0x4000)
#defineG_Cb     (int)(-0.34414*0x4000)
#defineG_Cr     (int)(-0.71414*0x4000)
#defineB_Cb     (int)(1.77200*0x4000)
#defineshiftbit 14

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;
}

void JPEGEXputMCU(short*mcubuff,int Y,int X,struct JPEGEXMCUSTR*MCUstr)
{
    int x,y,w,h,Cy,Cu,Cv,R,G,B;
    int hf0,hf1,hf2,vf0,vf1,vf2,hfs0,hfs1,hfs2;
    int x0,x1,x2,y0,y1,y2;
    short* mcubuff1;
    short* mcubuff2;

    hf0=MCUstr->hf[0];vf0=MCUstr->vf[0];
    hf1=MCUstr->hf[1];vf1=MCUstr->vf[1];
    hf2=MCUstr->hf[2];
        w=(MCUstr->hfMax)*8;h=(MCUstr->vfMax)*8;
    mcubuff1=mcubuff+((vf0*hf0)<<6);
    mcubuff2=mcubuff1+((vf1*hf1)<<6);
    vf0=MCUstr->vfs[0];
    vf1=MCUstr->vfs[1];
    vf2=MCUstr->vfs[2];
    hfs0=MCUstr->hfs[0];
    hfs1=MCUstr->hfs[1];
    hfs2=MCUstr->hfs[2];
    for(y=0;y<h;y++){
        y0=(y>>vf0);
        y1=(y>>vf1);
        y2=(y>>vf2);
    for(x=0;x<w;x++){
        x0=(x>>hfs0);
        x1=(x>>hfs1);
        x2=(x>>hfs2);
               Cy=*(mcubuff+(((x0>>3)+(y0>>3)*hf0)<<6)+(y0&7)*8+(x0&7));
               Cu=*(mcubuff1+(((x1>>3)+(y1>>3)*hf1)<<6)+(y1&7)*8+(x1&7));
               Cv=*(mcubuff2+(((x2>>3)+(y2>>3)*hf2)<<6)+(y2&7)*8+(x2&7));
               R=(((Cy<<shiftbit)+(Cv-0x80)*R_Cr+(1<<(shiftbit-1)))>>shiftbit);
               G=(((Cy<<shiftbit)+(Cu-0x80)*G_Cb+(Cv-0x80)*G_Cr+(1<<(shiftbit-1)))>>
shiftbit);
               B=(((Cy<<shiftbit)+(Cu-0x80)*B_Cb+(1<<(shiftbit-1)))>>shiftbit);
               Cy=(R<0)? 0:(R>255)?255:R;
               Cu=(G<0)? 0:(G>255)?255:G;
               Cv=(B<0)? 0:(B>255)?255:B;
        JPEGEXpset(MCUstr,Y+y,X+x,Cy,Cu,Cv);
      }
   }
}
```

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）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）を使用して媒体からハード・ディスクに内容をコピーしてください。

ハード・ディスクにコピーする際のディレクトリ構成は，特に指定しません。

（２）UNIX™版の場合

★ ハード・ディスクに“ NEC_JPEG.tar ”をコピーし，コマンド“ tar xovf NEC_JPEG.tar ”を使用してファイルを復元してください。デバイス指定は，使用環境により異なります。

ハード・ディスクにコピーする際のディレクトリ構成は，特に指定しません。

★ ４.２　サンプル・プログラム実行手順

パッケージには，圧縮処理／伸長処理／解析処理のサンプル・プログラムのソースが添付されています。システム例としてご活用ください。

４.２.１　基本ライブラリのサンプル・プログラム概要

（１）サンプル・プログラムの概要

ビット・マップ・ファイル（Targa形式）をダウンロードしライブラリでJPEG圧縮します。作成したJPEGファイルをライブラリでJPEG伸長／JPEG解析します。なお，実行する場合は，実機かシミュレータが必要です。

メモリ・マッピングは，次のとおりです。

図４－１　基本ライブラリ・メモリ・マッピング例（V850E/MS1）

0x00FFFFFF 0x00FFF000	周辺I/O領域	未使用
0x00FFEFFF 0x00FFE000	内蔵RAM領域	未使用
0x00FFDFFF 0x00100000	外部メモリ	0x00800000-0x008FFFFF スタック データ（サンプル） 0x00200000-0x002FFFFF 画像メモリ（VRAM）
0x000FFFFF 0x00000160	内蔵ROM領域	プログラム （ライブラリ，サンプル） 参照専用データ （ライブラリ，サンプル）
0x0000015F 0x00000000	割り込みベクタ・テーブル	（リセット・ベクタ使用）

（２）コンパイルとリンク

（ａ）ディレクトリの移動

作業ディレクトリへ移動します。移動先は，次のとおりです。

表４－１　ファイルの移動先（１）

	Windows版	UNIX版
NEC版	\NECTools32\smp850\jpeg\base	/NECTools32/smp850/jpeg/base
GHS版	\NECTools32\smp850_ghs\jpeg\base	/NECTools/smp850_ghs/jpeg/base

（ｂ）サンプル・ソースの変更

makefileに記述されているTOOLDIRの設定を変更してください。

start.sにMM/DWC1/DWC2/BCC...の設定を記述してください。MMは，外部拡張モード（ex.V850E/MS1の場合0x07）を設定してください（ソフトウエア・シミュレータで動作させる場合，この設定は，必ずしも必要ではありません）。

（ｃ）make

コマンド“ make ”を実行してください。

（３）実機または，シミュレータでの実行

（ａ）ディバッガ，シミュレータの設定

外部メモリを使用しているので次の設定を行ってください。

MM：外部拡張モード

外部メモリ設定：0x00100000-0x00FFEFFFを使用可能に

（ｂ）ダウンロード

プログラム（実行形式）

ビット・マップ・ファイル（_tgaFile0にfish.tgaをダウンロード）。

（ｃ）ブレークポイントの設定

main（）中の「tp02（）が終了する場所」にブレークポイントを設定

（ｄ）実行

（ｅ）アップロード

JPEGファイル（圧縮結果：_jpegFile1から0x2327バイト）

ビット・マップ・ファイル（伸長結果：_tgaFile1から0x17A12バイト）

（ｆ）確認

errの値が0x00000000か確認

（ｅ）でアップロードしたデータを画像ビューアで確認

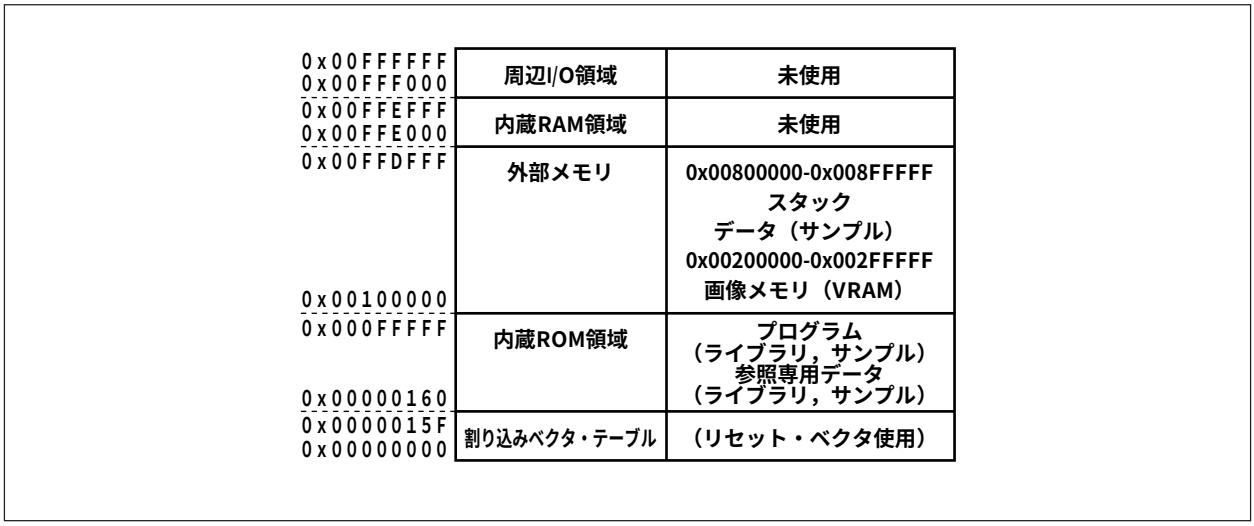
４.２.２　追加ライブラリのサンプル・プログラム概要

（１）サンプル・プログラムの概要

JPEGバッファにJPEGをダウンロードし，追加ライブラリでJPEG伸長します。なお，実行する場合は，実機かシミュレータが必要です。

メモリ・マッピングは，次のとおりです。

図４－２　追加ライブラリ・メモリ・マッピング例（V850E/MS1）



（２）コンパイルとリンク

（ａ）ディレクトリの移動

作業ディレクトリへ移動します。移動先は，次のとおりです。

表４－２　ファイルの移動先（２）

	Windows版	UNIX版
NEC版	\NECTools32\smp850\jpeg\ex	/NECTools32/smp850/jpeg/ex
GHS版	\NECTools32\smp850_ghs\jpeg\ex	/NECTools32/smp850_ghs/jpeg/ex

（ｂ）サンプル・ソースの変更

makefileに記述されているTOOLDIRの設定を変更してください。

start.sにMM/DWC1/DWC2/BCC...の設定を記述してください。MMは，外部拡張モード（ex.V850E/MS1の場合0x07）を設定してください（ソフトウエア・シミュレータで動作させる場合，この設定は，必ずしも必要ではありません）。

（ｃ）make

コマンド“ make ”を実行してください。

（３）実機または，シミュレータでの実行

（ａ）ディバッガ，シミュレータの設定

外部メモリを使用しているので次の設定を行ってください。

MM：外部拡張モード

外部メモリ設定：0x00100000-0x00FFEFFFを使用可能に

（ｂ）ダウンロード

プログラム（実行形式）

ビット・マップ・ファイル（_tgaFile1にbasesmp.jpgまたはprogsmp.jpgをダウンロード）。

（ｃ）ブレークポイントの設定

main（）中の「JPEGEXdecode（）が終了する場所」にブレークポイントを設定

（ｄ）実行

（ｅ）アップロード

ビット・マップ・ファイル（伸長結果：_tgaFile1から0x38412バイト）

（ｆ）確認

errの値が0x00000001か確認

（ｅ）でアップロードしたデータを画像ビューアで確認

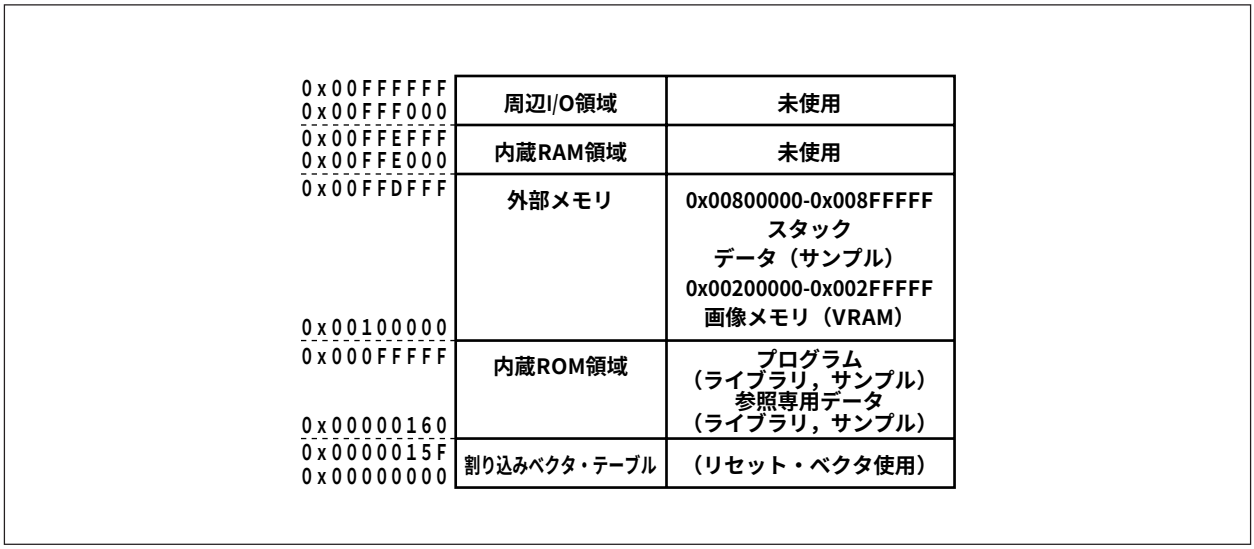
４.２.３　基本ライブラリ，追加ライブラリのサンプル・プログラム概要

（１）サンプル・プログラムの概要

JPEGバッファにJPEGをダウンロードし，追加ライブラリでJPEG伸長します。なお，実行する場合は，実機かシミュレータが必要です。

メモリ・マッピングは，次のとおりです。

図４－３　基本ライブラリ／追加ライブラリ・メモリ・マッピング例（V850E/MS1）



（２）コンパイルとリンク

（ａ）ディレクトリの移動

作業ディレクトリへ移動します。移動先は，次のとおりです。

表４－３　ファイルの移動先（３）

	Windows版	UNIX版
NEC版	\NECTools32\smp850\jpeg\both	/NECTools32/smp850/jpeg/both
GHS版	\NECTools32\smp850_ghs\jpeg\both	/NECTools32/smp850_ghs/jpeg/both

（ｂ）サンプル・ソースの変更

makefileに記述されているTOOLDIRの設定を変更してください。

start.sにMM/DWC1/DWC2/BCC...の設定を記述してください。MMは，外部拡張モード（ex.V850E/MS1の場合0x07）を設定してください（ソフトウエア・シミュレータで動作させる場合，この設定は，必ずしも必要ではありません）。

（ｃ）make

コマンド“ make ”を実行してください。

（３）実機または，シミュレータでの実行

（ａ）ディバッガ，シミュレータの設定

外部メモリを使用しているので次の設定を行ってください。

MM：外部拡張モード

外部メモリ設定：0x00100000-0x00FFEFFFを使用可能に

（ｂ）ダウンロード

プログラム（実行形式）

ビット・マップ・ファイル（_tgaFile0にfish.tgaをダウンロード）。

（ｃ）ブレークポイントの設定

main（）中の「JPEGEXdecode（）が終了する場所」にブレークポイントを設定

（ｄ）実行

（ｅ）アップロード

JPEGファイル（圧縮結果：_jpegFile1から0x2327バイト）

ビット・マップ・ファイル（伸長結果：_tgaFile1から0x17A12バイト）

（ｆ）確認

errの値が0x00000001か確認

（ｅ）でアップロードしたデータを画像ビューアで確認

付録A　サンプル・プログラム・ソース・リスト

A.1　基本ライブラリ・サンプル・プログラム

あとに示す，サンプル・プログラム・ソース・リストは表A－1～表A－3のような設定で実行した場合の例です。

表A－1　圧縮処理の設定

項　目	構造体メンバ	設定値	説　　明
画像メモリ（VRAM）	VRAM_Bptr	&（vramArea［2］）	画像メモリ：uchar vramAre［］
	VRAM_W_Pixel	320	画像サイズ：320［W］×240［H］
	VRAM_H_Pixel	240	座標位置：（0, 0）-（319, 239）
	VRAM_Line_Byte	1280	画像格納順序：左→右，上→下
	VRAM_Pixel_Byte	4	色空間：24ビットRGB
	VRAM_GAP1_Byte	－1	pixel-format：B, G, R, reserveの順。
	VRAM_GAP2_Byte	－2	B = G = R = rev = 1［バイト］
	Mode	JPEGMODE_NORMAL	動作モード：通常圧縮
JPEG圧縮情報１	Width	320	画像サイズ：320［W］×240［H］
	Height	240	
	StartX	0	座標位置：（0, 0）-（319, 239）
	StartY	0	
JPEG圧縮情報２ （JPEG特有の設定）	Cl1	0x01	コンポーネント１のID：0x01
	Cl2	0x02	コンポーネント２のID：0x02
	Cl3	0x03	コンポーネント３のID：0x03
	Quality	75	Qualityパラメータ：75
	Sampling	JPEGSAMPLE21	サンプル比：4：2：2［H：V = 2：1］
	DQT_Y_Bptr	&（jpeg_DQT_Y［0］）	量子化テーブル：
	DQT_C_Bptr	&（jpeg_DQT_C［0］）	ライブラリ・ディフォールト
	DHT_DC_Y_Bptr	&（jpeg_DHT_DC_Y［0］）	ハフマン・テーブル：
	DHT_DC_C_Bptr	&（jpeg_DHT_DC_C［0］）	ライブラリ・ディフォールト
	DHT_AC_Y_Bptr	&（jpeg_DHT_AC_Y［0］）	
	DHT_AC_C_Bptr	&（jpeg_DHT_AC_C［0］）	
	Restart	0	RSTmマーカ：挿入しない
	DNL	JPEGDNL_OFF	DNLマーカ：挿入しない
	APP_Info_Bptr	0	APPnマーカ：挿入しない
JPEG格納メモリ	JPEG_Buff_Bptr	&（jpegBuff［0］）	JPEGファイル格納用バッファ
	JPEG_Buff_Eptr	&（jpegBuff［0］）＋0x8000	
ライブラリ・ワーク	MCU_Buff_Bptr	&（mcuBuff［0］　［0］）	MCUバッファ
	Work1_Bptr	&（work1Buff［0］）	ライブラリ・ワーク１
	Work2_Bptr	&（work2Buff［0］）	ライブラリ・ワーク２
JPEGINFO構造体変数		jpegInfo	
対応するプログラムの位置		L121-L197	

表A－2　伸長処理の設定

項目	構造体メンバ	設定値	説明
画像メモリ（VRAM）	VRAM_Bptr	&（vramArea［2］）	画像メモリ：uchar vramAre［］
	VRAM_W_Pixel	320	画像サイズ：320［W］×240［H］
	VRAM_H_Pixel	240	座標位置：（0, 0）-（319, 239）
	VRAM_Line_Byte	1280	画像格納順序：左→右，上→下
	VRAM_Pixel_Byte	4	色空間：24ビットRGB
	VRAM_GAP1_Byte	－1	pixel-format：B, G, R, reserveの順。
	VRAM_GAP2_Byte	－2	B = G = R = rev = 1［バイト］
	Mode	JPEGMODE_NORMAL	動作モード：通常伸長
JPEG伸長情報１	StartX	0	座標位置：（0, 0）-（w, h）
	StartY	0	
JPEG伸長情報２（JPEG特有の設定）	APP_Info_Bptr	0	APPnマーカ：この関数では，０固定
JPEG格納メモリ	JPEG_Buff_Bptr	&（jpegBuff［0］）	JPEGファイル格納用バッファ
	JPEG_Buff_Eptr	&（jpegBuff［0］）＋0x8000	
ライブラリ・ワーク	MCU_Buff_Bptr	&（mcuBuff［0］［0］）	MCUバッファ
	Work1_Bptr	&（work1Buff［0］）	ライブラリ・ワーク１
	Work2_Bptr	&（work2Buff［0］）	ライブラリ・ワーク２
JPEGINFO構造体変数		jpegInfo	
対応するプログラムの位置		L198-L257	

表A－3　解析処理の設定

項目	構造体メンバ	設定値	説明
JPEG解析情報２（JPEG特有の設定）	APP_Info_Bptr	0	APPnマーカ：APPnマーカを解析しない
JPEG格納メモリ	JPEG_Buff_Bptr	&（jpegBuff［0］）	JPEGファイル格納用バッファ
	JPEG_Buff_Eptr	&（jpegBuff［0］）＋0x8000	
ライブラリ・ワーク	Work1_Bptr	&（work1Buff［0］）	ライブラリ・ワーク１
JPEGINFO構造体変数		jpegInfo	
対応するプログラムの位置		L258-L294	

```
001  /*=====================================================================  */
002  /*Copyright (C) NEC Corporation 1995,2002                               */
003  /*NEC CONFIDENTIAL AND PROPRIETARY                                      */
004  /*All rights reserved by NEC Corporation.                               */
005  /*Use of copyright notice does not evidence publication                 */
006  /*---------------------------------------------------------------------  */
007  /*80[character/line], 4[character/tab]                                  */
008  /*=====================================================================  */
009
010  /*=====================================================================  */
011  /*インクルードファイル                                                   */
012  /*=====================================================================  */
013  #include  <string.h>
014  #include  "jpeg.h"
015  /*=====================================================================  */
016  /*共通定義                                                               */
017  /*=====================================================================  */
018  #define   B2H(x)          (x >> 1)
019  #define   B2W(x)          (x >> 2)
020  #define   H2B(x)          (x << 1)
021  #define   H2W(x)          (x >> 1)
022  #define   W2B(x)          (x << 2)
023  #define   W2H(x)          (x << 1)
024  #define   RET_OK          0
025  #define   RET_ERR         (-1)
026  #define   COLOR_YCBCR     0
027  #define   COLOR_RGB       1
028  /*=====================================================================  */
029  /*JPEG-ミドルウエアを使用するために必要な定義                            */
030  /*=====================================================================  */
031  JPEGINFO  jpegInfo;                                    /* JPEGINFO構造体 */
032  #define   DEF_JPEGBUFFSIZE 0x00008000                  /* JPEG Buffer    */
033  U08       jpegBuff[DEF_JPEGBUFFSIZE];                  /* JPEG buffer    */
034  S16       mcuBuff[6][64];                              /* MCU buffer     */
035  APPINFO   appInfo;                                     /* APPINFO構造体  */
036  S32       work1Buff[B2W(0x00000100)];                  /* work1          */
037  S32       work2Buff[B2W(0x00002000)];                  /* work2          */
038  /*=====================================================================  */
039  /*JPEG-ミドルウエアを使用するために必要な定義(VRAMの設定)                */
040  /*=====================================================================  */
041  #define   DEF_VRAM_W      320
042  #define   DEF_VRAM_H      240
043  #define   DEF_VRAM_PIXEL  4
044  #define   DEF_VRAM_LINE   (DEF_VRAM_W * DEF_VRAM_PIXEL)
045  #define   DEF_VRAM_SIZE   (DEF_VRAM_H * DEF_VRAM_LINE)
046  #define   DEF_VRAM_GAP1   -1
047  #define   DEF_VRAM_GAP2   -2
048  #define   DEF_VRAM_COLOR  COLOR_RGB
049  U08       vramArea[DEF_VRAM_SIZE];
050  /*=====================================================================  */
051  /*本サンプルを使用するために必要な定義                                   */
052  /*=====================================================================  */
053  typedefstruct  {
054  PU08 Bp;                                             /* 開始アドレス     */
055  U16    imgWidth;                                     /* 表示領域幅(pixel)*/
056  U16    imgHeight;                                    /* 表示領域高(pixel)*/
057  S16    ByteLine;                                     /* 1ライン(byte)    */
058  S16    BytePixel;                                    /* 1ピクセル(byte)  */
059  S16    ByteGap1;                                     /* Gap1(byte)       */
060  S16    ByteGap2;                                     /* Gap2(byte)       */
```

```
061  S32    ColorType;                                          /* 色空間          */
062  } VRAMINFO;
063  typedef  VRAMINFO * PVRAMINFO;
064  typedefstruct  {
065  S16    X;                                                  /* 開始x座標       */
066  S16    Y;                                                  /* 開始y座標       */
067  S16    W;                                                  /* 幅              */
068  S16    H;                                                  /* 高さ            */
069  } RECT;
070  typedef  RECT * PRECT;
071  VRAMINFO vramInfo;                                         /* VRAM構成        */
072  RECT     imgRect;                                          /* 画像サイズと位置 */
073  U08      tgaFile0[DEF_VRAM_SIZE+18];                       /* TGAFILE1領域    */
074  U08      tgaFile1[DEF_VRAM_SIZE+18];                       /* TGAFILE2領域    */
075  U08      jpegFile1[0x00010000];                            /* JPEGFILE1領域   */
076  /*=====================================================================  */
077  /*関数プロトタイプ宣言                                                    */
078  /*=====================================================================  */
079  S32      tpCompress( void );
080  S32      tpDecompress( void );
081  S32      tpAnalyze( void );
082  S32      VRAM2TGA( PVRAMINFO, PRECT, PU08 );
083  void     TGA2VRAM( PVRAMINFO, PRECT, PU08 );
084  void     FIX_YCC2RGB_CCIR601( U08, U08, U08, PU08, PU08, PU08 );
085  void     FIX_RGB2YCC_CCIR601( U08, U08, U08, PU08, PU08, PU08 );
086  /*=====================================================================  */
087  /*メインプログラム                                                        */
088  /*=====================================================================  */
089  void main( void )
090  {
091  S32  err;
092
093  /* ---------------------------------------------------------    */
094  /* VRAM構成をVRAMINFO構造体に設定                                 */
095  /* ---------------------------------------------------------    */
096  vramInfo.Bp        = &(vramArea[0]);
097  vramInfo.imgWidth  = DEF_VRAM_W;
098  vramInfo.imgHeight = DEF_VRAM_H;
099  vramInfo.ByteLine  = DEF_VRAM_LINE;
100  vramInfo.BytePixel = DEF_VRAM_PIXEL;
101  vramInfo.ByteGap1  = DEF_VRAM_GAP1;
102  vramInfo.ByteGap2  = DEF_VRAM_GAP2;
103  vramInfo.ColorType = DEF_VRAM_COLOR;
104  err                = RET_OK;
105  /* ---------------------------------------------------------    */
106  /* Stop!                                                        */
107  /* [1]TGAFILEをダウンロード(FISH.TGA->tgaFile0)                   */
108  /* ---------------------------------------------------------    */
109  err |= tpCompress();                    /* 圧縮のテスト        */
110  err |= tpDecompress();                  /* 伸長のテスト        */
111  err |= tpAnalyze();                     /* 解析のテスト        */
112  /* ---------------------------------------------------------    */
113  /* Stop!                                                        */
114  /* [1]エラー値をチェック(err==RET_OK)?(OK):(NG)                   */
115  /* [2]JPEGFILEをアップロード(jpegFile1->FISH??R1.JPG)             */
116  /* [3]TGAFILEをアップロード(tgaFile1->FISH??R2.TGA)               */
117  /*                        "??"=SAMPLE値(11,21,22,...)           */
118  /* ---------------------------------------------------------    */
119  return;
120  }
```

```
121  /*=====================================================================  */
122  /*圧縮テストプログラム                                                    */
123  /*=====================================================================  */
124  S32  tpCompress( void )
125  {
126  PU08 file;
127  S32    size;
128
129  imgRect.X                = 0;
130  imgRect.Y                = 0;
131  TGA2VRAM( &vramInfo, &imgRect, &(tgaFile0[0]) );
132  memset( (PU08)(&jpegInfo), 0x00, sizeof(JPEGINFO) );
133  jpegInfo.Quality         = 75;
134  jpegInfo.Sampling        = JPEGSAMPLE21;
135  jpegInfo.Restart         = 0;
136  jpegInfo.Width           = imgRect.W;
137  jpegInfo.Height          = imgRect.H;
138  jpegInfo.StartX          = imgRect.X;
139  jpegInfo.StartY          = imgRect.Y;
140  jpegInfo.VRAM_Bptr       = vramInfo.Bp;
141  jpegInfo.VRAM_W_Pixel    = vramInfo.imgWidth;
142  jpegInfo.VRAM_H_Pixel    = vramInfo.imgHeight;
143  jpegInfo.VRAM_Line_Byte  = vramInfo.ByteLine;
144  jpegInfo.VRAM_Pixel_Byte = vramInfo.BytePixel;
145  jpegInfo.VRAM_Gap1_Byte  = vramInfo.ByteGap1;
146  jpegInfo.VRAM_Gap2_Byte  = vramInfo.ByteGap2;
147  jpegInfo.JPEG_Buff_Bptr  = &(jpegBuff[0]);
148  jpegInfo.JPEG_Buff_Eptr  = jpegInfo.JPEG_Buff_Bptr + DEF_JPEGBUFFSIZE;
149  jpegInfo.MCU_Buff_Bptr   = &(mcuBuff[0][0]);
150  jpegInfo.DQT_Y_Bptr      = jpeg_DQT_Y;
151  jpegInfo.DQT_C_Bptr      = jpeg_DQT_C;
152  jpegInfo.DHT_DC_Y_Bptr   = jpeg_DHT_DC_Y;
153  jpegInfo.DHT_DC_C_Bptr   = jpeg_DHT_DC_C;
154  jpegInfo.DHT_AC_Y_Bptr   = jpeg_DHT_AC_Y;
155  jpegInfo.DHT_AC_C_Bptr   = jpeg_DHT_AC_C;
156  jpegInfo.APP_Info_Bptr   = 0;
157  jpegInfo.Work1_Bptr      = &(work1Buff[0]);
158  jpegInfo.Work2_Bptr      = &(work2Buff[0]);
159  jpegInfo.Mode            = JPEGMODE_NORMAL;
160  jpegInfo.DNL             = JPEGDNL_OFF;
161  jpegInfo.CI1             = 0x01;
162  jpegInfo.CI2             = 0x02;
163  jpegInfo.CI3             = 0x03;
164  file = &(jpegFile1[0]);
165  size = 0;
166  jpeg_CompressInit( &jpegInfo );
167  for( ; ; ) {
168     switch( jpeg_Compress( &jpegInfo ) ) {
169     case JPEGRET_OK:
170       size = jpegInfo.FileSize - size;
171       memcpy( file, jpegInfo.JPEG_Buff_Bptr, size );
172       return( RET_OK );
173     case JPEGRET_CONT1:
174       size = jpegInfo.FileSize - size;
175       memcpy( file, jpegInfo.JPEG_Buff_Bptr, size );
176       file += size;
177       size = jpegInfo.FileSize;
178       break;
179     case JPEGRET_CONT2:
180       if( jpegInfo.Mode != JPEGMODE_MCULINE ) {
```

```
181       return( RET_ERR );
182       }
183       jpegInfo.StartX =  imgRect.X;
184       if( (jpegInfo.Sampling & JPEGSAMPLEMONO) == 0 ) {
185         jpegInfo.StartY += JPEGGET_MCUV( jpegInfo.Sampling );
186       } else {
187         jpegInfo.StartY += 8;
188       }
189       break;
190     case JPEGRET_ERR:
191       return( RET_ERR );
192     default:
193       return( RET_ERR );
194     }
195   }
196   return( RET_ERR );
197   }
198   /*=====================================================================  */
199   /*伸長テストプログラム                                                    */
200   /*=====================================================================  */
201   S32  tpDecompress( void )
202   {
203   PU08 file;
204   S32    size;
205
206   imgRect.X                = 0;
207   imgRect.Y                = 0;
208   memset( (PU08)(&jpegInfo), 0x00, sizeof(JPEGINFO) );
209   jpegInfo.StartX          = imgRect.X;
210   jpegInfo.StartY          = imgRect.Y;
211   jpegInfo.VRAM_Bptr       = vramInfo.Bp;
212   jpegInfo.VRAM_W_Pixel    = vramInfo.imgWidth;
213   jpegInfo.VRAM_H_Pixel    = vramInfo.imgHeight;
214   jpegInfo.VRAM_Line_Byte  = vramInfo.ByteLine;
215   jpegInfo.VRAM_Pixel_Byte = vramInfo.BytePixel;
216   jpegInfo.VRAM_Gap1_Byte  = vramInfo.ByteGap1;
217   jpegInfo.VRAM_Gap2_Byte  = vramInfo.ByteGap2;
218   jpegInfo.JPEG_Buff_Bptr  = &(jpegBuff[0]);
219   jpegInfo.JPEG_Buff_Eptr  = jpegInfo.JPEG_Buff_Bptr + DEF_JPEGBUFFSIZE;
220   jpegInfo.MCU_Buff_Bptr   = &(mcuBuff[0][0]);
221   jpegInfo.APP_Info_Bptr   = 0;
222   jpegInfo.Work1_Bptr      = &(work1Buff[0]);
223   jpegInfo.Work2_Bptr      = &(work2Buff[0]);
224   jpegInfo.Mode            = JPEGMODE_NORMAL;
225   file = &(jpegFile1[0]);
226   size = jpegInfo.JPEG_Buff_Eptr - jpegInfo.JPEG_Buff_Bptr;
227   memcpy( jpegInfo.JPEG_Buff_Bptr, file, size );
228   jpeg_DecompressInit( &jpegInfo );
229   for( ; ; ) {
230     switch( jpeg_Decompress( &jpegInfo ) ) {
231     case JPEGRET_OK:
232       imgRect.W = jpegInfo.Width;
233       imgRect.H = jpegInfo.Height;
234       VRAM2TGA( &vramInfo, &imgRect, &(tgaFile1[0]) );
235       return( RET_OK );
236     case JPEGRET_CONT1:
237       memcpy( jpegInfo.JPEG_Buff_Bptr, file, size );
238       break;
239     case JPEGRET_CONT2:
240       if( jpegInfo.Mode != JPEGMODE_MCULINE ) {
```

```
241         return( RET_ERR );
242       }
243       jpegInfo.StartX =  imgRect.X;
244       if( (jpegInfo.Sampling & JPEGSAMPLEMONO) == 0 ) {
245         jpegInfo.StartY += JPEGGET_MCUV( jpegInfo.Sampling );
246       } else {
247         jpegInfo.StartY += 8;
248       }
249       break;
250     case JPEGRET_ERR:
251       return( RET_ERR );
252     default:
253       return( RET_ERR );
254     }
255   }
256   return( RET_ERR );
257   }
258   /*====================================================================== */
259   /*解析テストプログラム                                                   */
260   /*====================================================================== */
261   S32  tpAnalyze( void )
262   {
263   PU08 file;
264   S32    size;
265
266   imgRect.X                = 0;
267   imgRect.Y                = 0;
268   memset( (PU08)(&jpegInfo), 0x00, sizeof(JPEGINFO) );
269   jpegInfo.JPEG_Buff_Bptr   = &(jpegBuff[0]);
270   jpegInfo.JPEG_Buff_Eptr   = jpegInfo.JPEG_Buff_Bptr + DEF_JPEGBUFFSIZE;
271   jpegInfo.APP_Info_Bptr    = 0;
272   jpegInfo.Work1_Bptr       = &(work1Buff[0]);
273   file = &(jpegFile1[0]);
274   size = jpegInfo.JPEG_Buff_Eptr - jpegInfo.JPEG_Buff_Bptr;
275   memcpy( jpegInfo.JPEG_Buff_Bptr, file, size );
276   jpeg_AnalysisInit( &jpegInfo );
277   for( ; ; ) {
278     switch( jpeg_Analysis( &jpegInfo ) ) {
279     case JPEGRET_OK:
280       imgRect.W = jpegInfo.Width;
281       imgRect.H = jpegInfo.Height;
282       return( RET_OK );
283     case JPEGRET_CONT1:
284       memcpy( jpegInfo.JPEG_Buff_Bptr, file, size );
285       break;
286     case JPEGRET_CONT2:
287     case JPEGRET_ERR:
288       return( RET_ERR );
289     default:
290       return( RET_ERR );
291     }
292   }
293   return( RET_ERR );
294   }
295   /*====================================================================== */
296   /*VRAM/Targa変換プログラム                                               */
297   /*====================================================================== */
298   S32  VRAM2TGA( PVRAMINFO info, PRECT rect, PU08 tga )
299   {
300   S16    sx, sy, x, y;
```

```
301  S16    width, height;
302  S16    pixel, line;
303  PU08 vp;
304  U08    p0, p1, p2;
305  S32    size;
306
307  sx     = (S16)(rect->X);
308  sy     = (S16)(rect->Y);
309  width  = (S16)(rect->W);
310  height = (S16)(rect->H);
311  pixel  = (S16)(info->BytePixel);
312  line   = (S16)(info->ByteLine);
313  *(tga +  0) = 0;
314  *(tga +  1) = 0;
315  *(tga +  2) = 2;
316  *(tga +  3) = 0;
317  *(tga +  4) = 0;
318  *(tga +  5) = 0;
319  *(tga +  6) = 0;
320  *(tga +  7) = 0;
321  *(tga +  8) = 0;
322  *(tga +  9) = 0;
323  *(tga + 10) = 0;
324  *(tga + 11) = 0;
325  *(tga + 12) = (U08)(width & 0xFF);
326  *(tga + 13) = (U08)((width >> 8) & 0xFF);
327  *(tga + 14) = (U08)(height & 0xFF);
328  *(tga + 15) = (U08)((height >> 8) & 0xFF);
329  *(tga + 16) = 24;
330  *(tga + 17) = 0x20;
331  tga  += 18;
332  size =  18;
333  for( y = 0; y < height; y ++ ){
334    for( x = 0; x < width; x ++ ){
335      vp =  info->Bp;
336      vp += (S32)((S16)(sx + x) * pixel);
337      vp += (S32)((S16)(sy + y) * line);
338      p0 = *vp;
339      p1 = *(vp + (info->ByteGap1));
340      p2 = *(vp + (info->ByteGap2));
341      if( info->ColorType == COLOR_YCBCR ) {
342        FIX_YCC2RGB_CCIR601( p0, p1, p2, &p0, &p1, &p2 );
343      }
344      *(tga + 0) = p2;
345      *(tga + 1) = p1;
346      *(tga + 2) = p0;
347      tga  += 3;
348      size += 3;
349    }
350  }
351  return( size );
352  }
353
354  void TGA2VRAM( PVRAMINFO info, PRECT rect, PU08 tga )
355  {
356  S16       sx, sy, x, y;
357  S16       width, height;
358  S16       pixel, line;
359  PU08 vp;
360  U08       p0, p1, p2;
```

```
361
362  sx     =  (S16)(rect->X);
363  sy     =  (S16)(rect->Y);
364  width  =  ((S16)(*(tga+12)) & 0xFF);
365  width |= (((S16)(*(tga+13)) & 0xFF) << 8);
366  height =  ((S16)(*(tga+14)) & 0xFF);
367  height|= (((S16)(*(tga+15)) & 0xFF) << 8);
368  pixel  =  (S16)(info->BytePixel);
369  line   =  (S16)(info->ByteLine);
370  rect->W=  width;
371  rect->H=  height;
372  tga     += 18;
373  for( y = 0; y < height; y ++ ){
374    for( x = 0; x < width; x ++ ){
375      vp = info->Bp;
376      vp += (S32)((S16)(sx + x) * pixel);
377      vp += (S32)((S16)(sy + y) * line);
378      p2 = *(tga + 0);
379      p1 = *(tga + 1);
380      p0 = *(tga + 2);
381      if( info->ColorType == COLOR_YCBCR ) {
382        FIX_RGB2YCC_CCIR601( p0, p1, p2, &p0, &p1, &p2 );
383      }
384      *vp              = p0;
385      *(vp + (info->ByteGap1))  = p1;
386      *(vp + (info->ByteGap2))  = p2;
387      tga += 3;
388    }
389  }
390  return;
391  }
392  /*=========================================================================  */
393  /*CCIR601-1色変換(RGB/YCbCr)プログラム                                        */
394  /*=========================================================================  */
395  #defineS08MIN           ((S08)0xFFFFFF80)
396  #defineS08MAX           ((S08)0x0000007F)
397  #defineU08MIN           ((U08)0x00000000)
398  #defineU08MAX           ((U08)0x000000FF)
399  #defineZERO             ( 0)
400  #definePOSITIVE         ( 1)
401  #defineNEGATIVE         (-1)
402  #defineMIN(x,min)       (((x)>(min))?(min):(x))
403  #defineMAX(x,max)       (((x)<(max))?(max):(x))
404  #defineSAT(x,min,max)   (((x)<(min))?(min):(((x)>(max))?(max):(x)))
405  #defineSIGN(x)          (((x)==ZERO)?(ZERO):(((x)>ZERO)?(POSITIVE):(NEGATIVE)))
406  #defineABS(x)           (((x)>=ZERO)?((x)):(-(x)))
407  #defineFLT2FIX(x,p)     ((S16)((ABS((x))*(1<<(p))    )*(SIGN((x)))))
408  #defineRNDFLT2FIX(x,p)  ((S16)((ABS((x))*(1<<(p))+0.5)*(SIGN((x)))))
409  #defineFIX2INT(x,p)     ( (x)              >>(p))
410  #defineRNDFIX2INT(x,p)  (((x)+(1<<((p)-1)))>>(p))
411  #defineFLT2INT(x)       ((S16)((ABS((x))    )*(SIGN((x)))))
412  #defineRNDFLT2INT(x)    ((S16)((ABS((x))+0.5)*(SIGN((x)))))
413  #defineFLT_RGB2YCC_R2Y  ( 0.29900)
414  #defineFLT_RGB2YCC_G2Y  ( 0.58700)
415  #defineFLT_RGB2YCC_B2Y  ( 0.11400)
416  #defineFLT_RGB2YCC_R2CB (-0.16874)
417  #defineFLT_RGB2YCC_G2CB (-0.33126)
418  #defineFLT_RGB2YCC_B2CB ( 0.50000)
419  #defineFLT_RGB2YCC_R2CR ( 0.50000)
420  #defineFLT_RGB2YCC_G2CR (-0.41869)
```

```
421  #defineFLT_RGB2YCC_B2CR (-0.08131)
422  #defineFIX_RGB2YCC_FP   (14)
423  #defineFIX_RGB2YCC_R2Y  (RNDFLT2FIX(FLT_RGB2YCC_R2Y, FIX_RGB2YCC_FP))
424  #defineFIX_RGB2YCC_G2Y  (RNDFLT2FIX(FLT_RGB2YCC_G2Y, FIX_RGB2YCC_FP))
425  #defineFIX_RGB2YCC_B2Y  (RNDFLT2FIX(FLT_RGB2YCC_B2Y, FIX_RGB2YCC_FP))
426  #defineFIX_RGB2YCC_R2CB (RNDFLT2FIX(FLT_RGB2YCC_R2CB,FIX_RGB2YCC_FP))
427  #defineFIX_RGB2YCC_G2CB (RNDFLT2FIX(FLT_RGB2YCC_G2CB,FIX_RGB2YCC_FP))
428  #defineFIX_RGB2YCC_B2CB (RNDFLT2FIX(FLT_RGB2YCC_B2CB,FIX_RGB2YCC_FP))
429  #defineFIX_RGB2YCC_R2CR (RNDFLT2FIX(FLT_RGB2YCC_R2CR,FIX_RGB2YCC_FP))
430  #defineFIX_RGB2YCC_G2CR (RNDFLT2FIX(FLT_RGB2YCC_G2CR,FIX_RGB2YCC_FP))
431  #defineFIX_RGB2YCC_B2CR (RNDFLT2FIX(FLT_RGB2YCC_B2CR,FIX_RGB2YCC_FP))
432  #defineFLT_YCC2RGB_Y2R  ( 1.00000)
433  #defineFLT_YCC2RGB_CB2R ( 0.00000)
434  #defineFLT_YCC2RGB_CR2R ( 1.40200)
435  #defineFLT_YCC2RGB_Y2G  ( 1.00000)
436  #defineFLT_YCC2RGB_CB2G (-0.34414)
437  #defineFLT_YCC2RGB_CR2G (-0.71414)
438  #defineFLT_YCC2RGB_Y2B  ( 1.00000)
439  #defineFLT_YCC2RGB_CB2B ( 1.77200)
440  #defineFLT_YCC2RGB_CR2B ( 0.00000)
441  #defineFIX_YCC2RGB_FP   (14)
442  #defineFIX_YCC2RGB_Y2R  (RNDFLT2FIX(FLT_YCC2RGB_Y2R, FIX_YCC2RGB_FP))
443  #defineFIX_YCC2RGB_CB2R (RNDFLT2FIX(FLT_YCC2RGB_CB2R,FIX_YCC2RGB_FP))
444  #defineFIX_YCC2RGB_CR2R (RNDFLT2FIX(FLT_YCC2RGB_CR2R,FIX_YCC2RGB_FP))
445  #defineFIX_YCC2RGB_Y2G  (RNDFLT2FIX(FLT_YCC2RGB_Y2G, FIX_YCC2RGB_FP))
446  #defineFIX_YCC2RGB_CB2G (RNDFLT2FIX(FLT_YCC2RGB_CB2G,FIX_YCC2RGB_FP))
447  #defineFIX_YCC2RGB_CR2G (RNDFLT2FIX(FLT_YCC2RGB_CR2G,FIX_YCC2RGB_FP))
448  #defineFIX_YCC2RGB_Y2B  (RNDFLT2FIX(FLT_YCC2RGB_Y2B, FIX_YCC2RGB_FP))
449  #defineFIX_YCC2RGB_CB2B (RNDFLT2FIX(FLT_YCC2RGB_CB2B,FIX_YCC2RGB_FP))
450  #defineFIX_YCC2RGB_CR2B (RNDFLT2FIX(FLT_YCC2RGB_CR2B,FIX_YCC2RGB_FP))
451
452  void FIX_RGB2YCC_CCIR601( U08 R, U08 G, U08 B, PU08 Y, PU08 Cb, PU08 Cr )
453  {
454  S16  r, g, b;
455  S32  y, cb, cr;
456
457  r   =  (S16)R;
458  g   =  (S16)G;
459  b   =  (S16)B;
460  y   =  (FIX_RGB2YCC_R2Y *r)+(FIX_RGB2YCC_G2Y *g)+(FIX_RGB2YCC_B2Y *b);
461  cb  =  (FIX_RGB2YCC_R2CB*r)+(FIX_RGB2YCC_G2CB*g)+(FIX_RGB2YCC_B2CB*b);
462  cr  =  (FIX_RGB2YCC_R2CR*r)+(FIX_RGB2YCC_G2CR*g)+(FIX_RGB2YCC_B2CR*b);
463  y   =  FIX2INT(y, FIX_RGB2YCC_FP);
464  cb  =  FIX2INT(cb,FIX_RGB2YCC_FP);
465  cr  =  FIX2INT(cr,FIX_RGB2YCC_FP);
466  y   =  SAT(y, U08MIN,U08MAX);
467  cb  =  SAT(cb,S08MIN,S08MAX);
468  cr  =  SAT(cr,S08MIN,S08MAX);
469  y   += 0x00000000;
470  cb  += 0x00000080;
471  cr  += 0x00000080;
472  *Y  =  (U08)y;
473  *Cb =  (U08)cb;
474  *Cr =  (U08)cr;
475  return;
476  }
477
478  void FIX_YCC2RGB_CCIR601( U08 Y, U08 Cb, U08 Cr, PU08 R, PU08 G, PU08 B )
479  {
480  S16  y, cb, cr;
```

```
481  S32  r, g, b;
482
483  y  =  (S16)Y;
484  cb =  (S16)Cb;
485  cr =  (S16)Cr;
486  y  -= 0x00000000;
487  cb -= 0x00000080;
488  cr -= 0x00000080;
489  r  =  (FIX_YCC2RGB_Y2R*y)+(FIX_YCC2RGB_CB2R*cb)+(FIX_YCC2RGB_CR2R*cr);
490  g  =  (FIX_YCC2RGB_Y2G*y)+(FIX_YCC2RGB_CB2G*cb)+(FIX_YCC2RGB_CR2G*cr);
491  b  =  (FIX_YCC2RGB_Y2B*y)+(FIX_YCC2RGB_CB2B*cb)+(FIX_YCC2RGB_CR2B*cr);
492  r  =  FIX2INT(r,FIX_YCC2RGB_FP);
493  g  =  FIX2INT(g,FIX_YCC2RGB_FP);
494  b  =  FIX2INT(b,FIX_YCC2RGB_FP);
495  r  =  SAT(r,U08MIN,U08MAX);
496  g  =  SAT(g,U08MIN,U08MAX);
497  b  =  SAT(b,U08MIN,U08MAX);
498  *R =  (U08)r;
499  *G =  (U08)g;
500  *B =  (U08)b;
501  return;
502  }
```

★ A.2　追加伸長処理サンプル・プログラム（1passモード）

あとに示す，サンプル・プログラム・ソース・リストは表Ａ－４のような設定で実行した場合のみの例です。

表Ａ－４　追加伸長処理の設定（１）

項　目	構造体メンバ	設定値	説　　明
JPEG伸長情報（JPEGEXINFO構造体）	Mode	JPEGEXModeStart	動作モード：通常伸長
	Policy	JPEGEXLuminanceOutOnly ｜JPEGEXPutMCURGB	伸長時オプション：輝度成分ごとに描画 RGB出力
	Work	&JPWORK	struct JPEGEXWORK JPWORK
	Video	&JPVIDEO	struct JPEGEXVIDEO JPVIDEO
ライブラリ・ワーク（JPEGEXWORK構造体）	Work1	Work1Area	ワーク・エリア1：int Work1Area〔0x200〕
	Work1Len	0x800	
	Work2	Work2Area	ワーク・エリア2：int Work2Area〔0x14000〕
	Work2Len	0x50000	
画像メモリ（VRAM）（JPEGEXVIDEO構造体）	VRAMAddress	&tgaFile1［20］	VRAM：unsigned char tgaFile1［］
	VRAMWidth	320	VRAMサイズ：320〔W〕×240〔H〕
	VRAMHeight	240	座標位置：（0,0）-（319,239）
	VRAMPixel	3	画像格納順序：左→右，上→下
	VRAMLine	960	色空間：24ビットRGB
	VRAMGap0	0	pixel-format：B, G, Rの順
	VRAMGap1	－1	B = G = R = 1〔バイト〕
	VRAMGap2	－2	
	ClipStartX	0	クリッピングなし（ダミー値を設定）
	ClipStartY	0	
	ClipWidth	0x7FFFFFFF	
	ClipHeight	0x7FFFFFFF	
JPEG格納メモリ（JPEGEXBUFF構造体）	JPEGBUFF	&jpegFile1［0］	JPEGファイル格納用バッファ：unsigned char jpegFile1〔100000〕
	JPEGBUFFLEN	100000	
対応するプログラムの位置		L067-L124	

```
001  /*****************************************************************/
002  /*  Copyright (C) NEC Corporation 1995,2002                      */
003  /*  NEC CONFIDENTIAL AND PROPRIETARY                             */
004  /*  All rights reserved by NEC Corporation.                      */
005  /*  Use of copyright notice does not evidence publication        */
006  /*****************************************************************/
007  /***********************************************
008  *     include
009  ***********************************************/
011  #include "jpegex.h"
012
013  /***********************************************
014  **  Definition of the global variable
015  ***********************************************/
016  #define    JPEG_BUFSIZE   100000
017  #define    TMP_BUFSIZE    0x1000
018
019  #define    VRAM_WIDTH     320
020  #define    VRAM_HEIGHT    240
021  #define    VRAM_PIXEL     3
```

```
022  #define   VRAM_GAP1       -1
023  #define   VRAM_GAP2       -2
024  #define   VRAM_LINE       ( VRAM_WIDTH * VRAM_PIXEL )
025  #define   WORK1LENGTH     0x800
026  #define   WORK2LENGTH     0x50000
027
028  /* structure definition */
029  struct JPEGEXINFO JPINFO;
030  struct JPEGEXWORK JPWORK;
031  struct JPEGEXVIDEO JPVIDEO;
032
033  /***********************************************
034  **    Reference of the global variable
035  ***********************************************/
036  unsigned char      jpegFile1[JPEG_BUFSIZE];
037  unsigned char      tgaFile1[VRAM_WIDTH * VRAM_HEIGHT * VRAM_PIXEL + 18];
038  unsigned char      VRAM[VRAM_WIDTH * VRAM_HEIGHT * VRAM_PIXEL];
039
040  unsigned int       Work1Area[WORK1LENGTH/4];
041  unsigned int       Work2Area[WORK2LENGTH/4];
042
043  unsigned char      jpegtmp[TMP_BUFSIZE];
044  unsigned char      *jpegptr = (unsigned char*)(&(jpegFile1[0]));
045
046  /***********************************************
047  **   Function definition
048  ***********************************************/
049  void jd_proc(void);
050  void main(void);
051  void JPEGEXGetJpegStream(struct JPEGEXBUFF*);
052  void fillMem( unsigned char * dst, signed long size );
053  void tga_head( unsigned char * tga );
054
055  /***********************************************
056  *  main
057  ***********************************************/
058  void main(void)
059  {
060
061         /*  the decompression processing */
062         tga_head( &tgaFile1[0] );
063
064         jd_proc();
065  }
066
067  void jd_proc(void)
068  {
069         int ret;
070
071         fillMem( &tgaFile1[18], (signed long)(VRAM_HEIGHT * VRAM_LINE) );
072
073         /* jpeg decompress parameter */
074         JPINFO.Mode           = JPEGEXModeStart;
075         JPINFO.Policy         = JPEGEXLuminanceOutOnly | JPEGEXPutMCURGB;
076         JPINFO.Work           = &JPWORK;
077         JPWORK.Work1          = (unsigned int*)Work1Area;
078         JPWORK.Work1Len       = (int)WORK1LENGTH;
079         JPWORK.Work2          = (unsigned int*)Work2Area;
080         JPWORK.Work2Len       = (int)WORK2LENGTH;
081         JPINFO.Video          = &JPVIDEO;
```

```
082         JPVIDEO.VRAMAddress   = (unsigned char*)&tgaFile1[20];
083         JPVIDEO.VRAMWidth     = VRAM_WIDTH;
084         JPVIDEO.VRAMHeight    = VRAM_HEIGHT;
085         JPVIDEO.VRAMPixel     = VRAM_PIXEL;
086         JPVIDEO.VRAMLine      = VRAM_LINE;
087         JPVIDEO.VRAMGap0      = 0;
088         JPVIDEO.VRAMGap1      = VRAM_GAP1;
089         JPVIDEO.VRAMGap2      = VRAM_GAP2;
090         JPVIDEO.ClipStartX    = 0;
091         JPVIDEO.ClipStartY    = 0;
092         JPVIDEO.ClipWidth     = 0x7FFFFFFF;
093         JPVIDEO.ClipHeight    = 0x7FFFFFFF;
094
095         ret = JPEGEXdecode(&JPINFO);
096         if(ret == JPEGEXDecodeStatusComplete){
097             /* normal ending */
098             return;
099         }
100         if(ret == JPEGEXDecodeStatusError){
101             /* unusual ending */
102             return;
103         }
104         if(ret == JPEGEXDecodeStatusTerminate){
105             /* The compulsion ending */
106             return;
107         }
108         if(ret == JPEGEXDecodeStatusNotRunning){
109             /* The object is not running */
110             return;
111         }
112 }
113
114
115 void JPEGEXGetJpegStream(struct JPEGEXBUFF* JPBUFF)
116 {
117         int i;
118
119         for(i = 0; i<TMP_BUFSIZE; i++){
120             jpegtmp[i] = *jpegptr++;
121         }
122         JPBUFF->JPEGBUFF = jpegtmp;
123         JPBUFF->JPEGBUFFLEN = TMP_BUFSIZE;
124 }
125
126 /****************
127 *   FillMem     *
128 *****************/
129 void fillMem( unsigned char * dst, signed long size )
130 {
131 for( ; size > 0; size-- ) {
132             *dst = 0x00;
133             dst++;
134     }
135
136             return;
137 }
138
139 /****************
140 *    VRAM2TGA   *
141 ****************/
```

```
142  void tga_head( unsigned char * tga )
143  {
144              signed short    width, height;
145
146              width  = (signed short)VRAM_WIDTH;
147              height = (signed short)VRAM_HEIGHT;
148              *(tga +  0) = 0;
149              *(tga +  1) = 0;
150              *(tga +  2) = 2;
151              *(tga +  3) = 0;
152              *(tga +  4) = 0;
153              *(tga +  5) = 0;
154              *(tga +  6) = 0;
155              *(tga +  7) = 0;
156              *(tga +  8) = 0;
157              *(tga +  9) = 0;
158              *(tga + 10) = 0;
159              *(tga + 11) = 0;
160              *(tga + 12) = (unsigned char)(width & 0xFF);
161              *(tga + 13) = (unsigned char)((width >> 8) & 0xFF);
162              *(tga + 14) = (unsigned char)(height & 0xFF);
163              *(tga + 15) = (unsigned char)((height >> 8) & 0xFF);
164              *(tga + 16) = 24;
165              *(tga + 17) = 0x20;
166  }
```

★ A.3　基本圧縮，追加伸長処理サンプル・プログラム

あとに示す，サンプル・プログラム・ソース・リストは表Ａ－５，表Ａ－６のような設定で実行した場合のみの例です。

表Ａ－５　圧縮処理の設定（１）

項　目	構造体メンバ	設定値	説　　明
画像メモリ（VRAM）	VRAM_Bptr	&（vramArea［2］）	画像メモリ：uchar vramAre［］
	VRAM_W_Pixel	320	画像サイズ：320〔W〕×240〔H〕
	VRAM_H_Pixel	240	座標位置：（0, 0）－（319, 239）
	VRAM_Line_Byte	1280	画像格納順序：左→右, 上→下
	VRAM_Pixel_Byte	4	色空間：24ビットRGB
	VRAM_Gap1_Byte	－1	pixel-format：B, G, R, reserveの順
	VRAM_Gap2_Byte	－2	B = G = R = rev = 1〔バイト〕
	Mode	JPEGMODE_NORMAL	動作モード：通常圧縮
JPEG圧縮情報1	Width	320	画像サイズ：320〔W〕×240〔H〕
	Height	240	
	StartX	0	座標位置：（0, 0）－（319, 239）
	StartY	0	
JPEG圧縮情報2（JPEG特有の設定）	CI1	0x01	コンポーネント１のID：0x01
	CI2	0x02	コンポーネント２のID：0x02
	CI3	0x03	コンポーネント３のID：0x03
	Quality	75	Qualityパラメータ：75
	Sampling	JPEGSAMPLE21	サンプル比：4：2：2 〔H：V = 2：1〕
	DQT_Y_Bptr	&（jpeg_DQT_Y［0］）	量子化テーブル：
	DQT_C_Bptr	&（jpeg_DQT_C［0］）	ライブラリ・ディフォールト
	DHT_DC_Y_Bptr	&（jpeg_DHT_DC_Y［0］）	ハフマン・テーブル：
	DHT_DC_C_Bptr	&（jpeg_DHT_DC_C［0］）	ライブラリ・ディフォールト
	DHT_AC_Y_Bptr	&（jpeg_DHT_AC_Y［0］）	
	DHT_AC_C_Bptr	&（jpeg_DHT_AC_C［0］）	
	Restart	0	RSTmマーカ：挿入しない
	DNL	JPEGDNL_OFF	DNLマーカ：挿入しない
	APP_Info_Bptr	0	APPnマーカ：挿入しない
JPEG格納メモリ	JPEG_Buff_Bptr	&（jpegBuff［0］）	JPEGファイル格納用バッファ
	JPEG_Buff_Eptr	&（jpegBuff［0］）+0x8000	
ライブラリ・ワーク	MCU_Buff_Bptr	&（mcuBuff［0］　［0］）	MCUバッファ
	Work1_Bptr	&（work1Buff［0］）	ライブラリ・ワーク１
	Work2_Bptr	&（work2Buff［0］）	ライブラリ・ワーク２
JPEGINFO構造体変数		jpegInfo	
対応するプログラムの位置		L101-L211	

表A－6　圧縮処理の設定（2）

項　目	構造体メンバ	設定値	説　　明
JPEG伸長情報 （JPEGEXINFO構造体， 変数：JPINFO）	Mode	JPEGEXModeStart	動作モード：通常伸長
	Policy	JPEGEXLuminanceOutOnly ｜JPEGEXPutMCURGB ｜JPEGEX2passEnable	伸長時オプション：輝度成分ごとに描画 RGB出力 ２パス・モード伸長
	Work	＆JPWORK	struct JPEGEXWORK JPWORK
	Video	＆JPVIDEO	struct JPEGEXVIDEO JPVIDEO
ライブラリ・ワーク （JPEGEXWORK構造体， 変数：JPWORK）	Work1	Work1Area	ワーク･エリア１： int Work1Area〔0x40〕
	Work1Len	0x100	
	Work2	Work2Area	ワーク･エリア２： int Work2Area〔0x800〕
	Work2Len	0x2000	
画像メモリ（VRAM） （JPEGEXVIDEO構造体， 変数：JPVIDEO）	VRAMAddress	&tgaFile1［2］	VRAM：unsigned char tgaFile1［］
	VRAMWidth	320	VRAMサイズ：320〔W〕 x 240〔H〕
	VRAMHeight	240	座標位置：（0, 0）-（319, 239）
	VRAMPixel	4	画像格納順序：左→右，上→下 色空間：24ビットRGB pixel-format：B, G, R, reserveの順。 B = G = R = rev = 1〔バイト〕
	VRAMLine	1280	
	VRAMGap0	0	
	VRAMGap1	－1	
	VRAMGap2	－2	
	ClipStartX	0	クリッピングなし （ダミー値を設定）
	ClipStartY	0	
	ClipWidth	0x7FFFFFFF	
	ClipHeight	0x7FFFFFFF	
JPEG格納メモリ （JPEGEXBUFF構造体， 変数：JPBUFF）	JPEGBUFF	&jpegFile1［0］	JPEGファイル格納用バッファ： unsigned char jpegFile1〔0x8000〕
	JPEGBUFFLEN	0x00008000	
対応するプログラムの位置		L213-L272	

```
001  /*****************************************************************/
002  /*  Copyright (C) NEC Corporation 1995,2002                      */
003  /*  NEC CONFIDENTIAL AND PROPRIETARY                             */
004  /*  All rights reserved by NEC Corporation.                      */
005  /*  Use of copyright notice does not evidence publication        */
006  /*****************************************************************/
007
008  /*****************************************************************/
009  /*      Include Header File                                       */
010  /*****************************************************************/
011  /*     base       */
012  #include    "jpeg.h"
013  #include    "sample.h"
014  /*  ex  */
015  #include  "jpegex.h"
016
017  /*****************************************************************/
018  /*     Definition Constant                                       */
019  /*****************************************************************/
020  #define    DEF_QUALITY              75
021  #define    DEF_SAMPLE               (JPEGSAMPLE11)
022  #define    DEF_RST                  0
023
024  #define    TGA_FILE_SIZE            (DEF_VRAM_SIZE + 18)
```

```
025  #define    JPEG_FILE_SIZE            0x00010000
026  #define    JPEG_WORK1_SIZE           0x00000100
027  #define    JPEG_WORK2_SIZE           0x00002000
028  #define    JPEG_BUFF_SIZE            0x00008000
029
030  /*****************************************************************/
031  /*         Definition Type & Struct                              */
032  /*****************************************************************/
033  /* None */
034
035  /*****************************************************************/
036  /*         Definition Macro                                      */
037  /*****************************************************************/
038  /* None */
039
040  /*****************************************************************/
041  /*         Definition Global(External) Variable                 */
042  /*****************************************************************/
043  /*     base       */
044  JPEGINFO   jpegInfo;
045  APPINFO    appInfo;
046  /*     ex         */
047  struct     JPEGEXINFO JPINFO;
048  struct     JPEGEXWORK JPWORK;
049  struct     JPEGEXVIDEO JPVIDEO;
050
051  S32        work1Buff[B2W(JPEG_WORK1_SIZE)];
052  S32        work2Buff[B2W(JPEG_WORK2_SIZE)];
053  S16        mcuBuff[6][64];
054  U08        jpegBuff[JPEG_BUFF_SIZE];
055  U08        vramArea[DEF_VRAM_SIZE];      /* VRAM Area   */
056  U08        tgaFile0[TGA_FILE_SIZE];      /* DownLoad 1  */
057  U08        tgaFile1[TGA_FILE_SIZE];      /* UpLoad 2,4  */
058  U08        jpegFile1[JPEG_FILE_SIZE];    /* UpLoad 1,3  */
059  VRAMINFO   vramInfo;
060  RECT       imgRect;
061
062  /*****************************************************************/
063  /*     Definition Global(External) Function                      */
064  /*****************************************************************/
065  /*     base       */
066  S32    tp01( void );
067  S32    tpCompress( PJPEGINFO, PU08, PRECT );
068  void   InitApplication( void );
069  /*     ex     */
070  void   jd_proc(void);
071  void   JPEGEXGetJpegStream(struct JPEGEXBUFF*);
072
073  /*****************************************************************/
074  /*     Function                                                  */
075  /*****************************************************************/
076  /*****************************************************************/
077  /*     main                                                      */
078  /*****************************************************************/
079  void   main( void )
080  {
081        S32    err;
082
083        InitApplication();
084        err = RET_OK;
```

```
085         /* ------------------------------------------------------*/
086         /* Stop!                                                 */
087         /* DownLoad "FISH.TGA" -> "tgaFile0"                     */
088         /* ------------------------------------------------------*/
089         err |= tp01();        /* JPEG Compress   base            */
090         jd_proc();            /* JPEG Decompress ex              */
091         /* ------------------------------------------------------*/
092         /* Stop!                                                 */
093         /* Check "err" = RET_OK -> OK                            */
094         /* UpLoad "jpegFile1"->"FISH??R1.JPG"(result tp01)       */
095         /* UpLoad "tgaFile1 "->"FISH??R2.TGA"(result jd_proc)    */
096         /* ------------------------------------------------------*/
097
098         return;
099   }
100
101   /*************************************************************/
102   /*  tp01                                                     */
103   /*************************************************************/
104   S32    tp01( void )
105   {
106         imgRect.X      = 0;
107         imgRect.Y      = 0;
108         imgRect.W      = vramInfo.imgWidth;
109         imgRect.H      = vramInfo.imgHeight;
110         fillVRAM( &vramInfo, &imgRect );
111         TGA2VRAM( &vramInfo, &imgRect, &(tgaFile0[0]) );
112
113         fillMem( (PU08)(&jpegInfo), sizeof(JPEGINFO) );
114         jpegInfo.Quality          = DEF_QUALITY;
115         jpegInfo.Sampling         = DEF_SAMPLE;
116         jpegInfo.Restart          = DEF_RST;
117         jpegInfo.Width            = imgRect.W;
118         jpegInfo.Height           = imgRect.H;
119         jpegInfo.StartX           = imgRect.X;
120         jpegInfo.StartY           = imgRect.Y;
121         jpegInfo.VRAM_Bptr        = vramInfo.Bp;
122         jpegInfo.VRAM_W_Pixel     = vramInfo.imgWidth;
123         jpegInfo.VRAM_H_Pixel     = vramInfo.imgHeight;
124         jpegInfo.VRAM_Line_Byte   = vramInfo.ByteLine;
125         jpegInfo.VRAM_Pixel_Byte  = vramInfo.BytePixel;
126         jpegInfo.VRAM_Gap1_Byte   = vramInfo.ByteGap1;
127         jpegInfo.VRAM_Gap2_Byte   = vramInfo.ByteGap2;
128         jpegInfo.JPEG_Buff_Bptr   = &(jpegBuff[0]);
129         jpegInfo.JPEG_Buff_Eptr   = jpegInfo.JPEG_Buff_Bptr + JPEG_BUFF_SIZE;
130         jpegInfo.MCU_Buff_Bptr    = &(mcuBuff[0][0]);
131         jpegInfo.DQT_Y_Bptr       = jpeg_DQT_Y;
132         jpegInfo.DQT_C_Bptr       = jpeg_DQT_C;
133         jpegInfo.DHT_DC_Y_Bptr    = jpeg_DHT_DC_Y;
134         jpegInfo.DHT_DC_C_Bptr    = jpeg_DHT_DC_C;
135         jpegInfo.DHT_AC_Y_Bptr    = jpeg_DHT_AC_Y;
136         jpegInfo.DHT_AC_C_Bptr    = jpeg_DHT_AC_C;
137         jpegInfo.APP_Info_Bptr    = 0;
138         jpegInfo.Work1_Bptr       = &(work1Buff[0]);
139         jpegInfo.Work2_Bptr       = &(work2Buff[0]);
140         jpegInfo.Mode             = JPEGMODE_NORMAL;
141         jpegInfo.DNL              = JPEGDNL_OFF;
142         jpegInfo.CI1              = 0x01;
143         jpegInfo.CI2              = 0x02;
144         jpegInfo.CI3              = 0x03;
```

```
145
146         return( tpCompress( &jpegInfo, &(jpegFile1[0]), &imgRect ) );
147   }
148
149   /*************************************************************/
150   /*          tpCompress                                       */
151   /*************************************************************/
152   S32         tpCompress( PJPEGINFO info, PU08 file, PRECT rect )
153   {
154               S32       size;
155
156               size = 0;
157               jpeg_CompressInit( info );
158               for( ; ; ) {
159                     switch( jpeg_Compress( info ) ) {
160                     case JPEGRET_OK:
161                           size = info->FileSize - size;
162                           copyMem( info->JPEG_Buff_Bptr, file, size );
163                           return( RET_OK );
164                     case JPEGRET_CONT1:
165                           size = info->FileSize - size;
166                           copyMem( info->JPEG_Buff_Bptr, file, size );
167                           file += size;
168                           size = info->FileSize;
169                           break;
170                     case JPEGRET_CONT2:
171                           if( info->Mode != JPEGMODE_MCULINE ) {
172                                 return( RET_ERR );
173                           }
174                           info->StartX =  rect->X;
175                           if( (info->Sampling & JPEGSAMPLEMONO) == 0 ) {
176                           info->StartY += JPEGGET_MCUV( info->Sampling );
177                           } else {
178                                 info->StartY += 8;
179                           }
180                           break;
181                     case JPEGRET_ERR:
182                           return( RET_ERR );
183                     default:
184                           return( RET_ERR );
185                     }
186               }
187               return( RET_ERR );
188   }
189
190   /*************************************************************/
191   /*          Init Application                                 */
192   /*************************************************************/
193   void        InitApplication( void )
194   {
195   #if         1
196               vramInfo.Bp           = DEF_VRAM_BP;
197               vramInfo.Ep           = (PU08)(vramInfo.Bp + DEF_VRAM_SIZE);
198   #else
199               vramInfo.Bp           = &(vramArea[0]);
200               vramInfo.Ep           = &(vramArea[0]) + DEF_VRAM_SIZE;
201   #endif
202               vramInfo.imgWidth     = DEF_VRAM_W;
203               vramInfo.imgHeight    = DEF_VRAM_H;
204               vramInfo.ByteLine     = DEF_VRAM_LINE;
```

```
205             vramInfo.BytePixel    = DEF_VRAM_PIXEL;
206             vramInfo.ByteGap1     = DEF_VRAM_GAP1;
207             vramInfo.ByteGap2     = DEF_VRAM_GAP2;
208             vramInfo.ColorType    = DEF_VRAM_COLOR;
209
210             return;
211     }
212
213     /*************************************************************/
214     /*          jd_proc                                          */
215     /*************************************************************/
216     void jd_proc(void)
217     {
218             int ret;
219
220             fillMem( &tgaFile1[0], (signed long)(TGA_FILE_SIZE) );
221
222             /* jpeg decompress parameter */
223             JPINFO.Mode           = JPEGEXModeStart;
224             JPINFO.Policy         = JPEGEXLuminanceOutOnly | JPEGEXPutMCURGB;
225             JPINFO.Policy         |= JPEGEX2passEnable;
226             JPINFO.Work           = &JPWORK;
227             JPWORK.Work1          = (unsigned int*)work1Buff;
228             JPWORK.Work1Len       = (int)JPEG_WORK1_SIZE;
229             JPWORK.Work2          = (unsigned int*)work2Buff;
230             JPWORK.Work2Len       = (int)work2Buff;
231             JPINFO.Video          = &JPVIDEO;
232             JPVIDEO.VRAMAddress   = (unsigned char*)&tgaFile1[2];
233             JPVIDEO.VRAMWidth     = DEF_VRAM_W;
234             JPVIDEO.VRAMHeight    = DEF_VRAM_H;
235             JPVIDEO.VRAMPixel     = DEF_VRAM_PIXEL;
236             JPVIDEO.VRAMLine      = DEF_VRAM_LINE;
237             JPVIDEO.VRAMGap0      = 0;
238             JPVIDEO.VRAMGap1      = DEF_VRAM_GAP1;
239             JPVIDEO.VRAMGap2      = DEF_VRAM_GAP2;
240             JPVIDEO.ClipStartX    = 0;
241             JPVIDEO.ClipStartY    = 0;
242             JPVIDEO.ClipWidth     = 0x7FFFFFFF;
243             JPVIDEO.ClipHeight    = 0x7FFFFFFF;
244
245             ret = JPEGEXdecode(&JPINFO);
246             if(ret == JPEGEXDecodeStatusComplete){
247                    /* normal ending */
248                    VRAM2TGA( &vramInfo, &imgRect, &(tgaFile1[0]) );
249                    return;
250             }
251             if(ret == JPEGEXDecodeStatusError){
252                    /* unusual ending */
253                    return;
254             }
255             if(ret == JPEGEXDecodeStatusTerminate){
256                    /* The compulsion ending */
257                    return;
258             }
259             if(ret == JPEGEXDecodeStatusNotRunning){
260                    /* The object is not running */
261                    return;
262             }
263     }
264
```

```
265  /*************************************************************/
266  /*         JPEGEXGetJpegStream                               */
267  /*************************************************************/
268  void JPEGEXGetJpegStream(struct JPEGEXBUFF* JPBUFF)
269  {
270             JPBUFF->JPEGBUFF = jpegFile1;
271             JPBUFF->JPEGBUFFLEN = JPEG_FILE_SIZE;
272  }
```

付録B　総合索引

B.1　50音で始まる語句の索引

B.2　数字，アルファベットで始まる語句の索引

〔メ　モ〕

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
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：info@lsi.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
広　島　(082)242−5504
仙　台　(022)267−8740

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
前　橋　(027)243−6060
鳥　取　(0857)27−5313
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL(アドレス)　http://www.ic.nec.co.jp/

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］μSAP703000-B03　ユーザーズ・マニュアル
（U10684JJ3V0UM00（第３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，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。

日本電気（株）NECエレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ