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

μSAP77016-B22

JPEG ビデオ・コーデック・ミドルウエア

対象デバイス

μPD77110
μPD77111
μPD77112
μPD77113A
μPD77114
μPD77115
μPD77210
μPD77213
μPD77214

資料番号 U16554JJ1V0UM00（第 1 版）
発行年月 February 2003 NS CP（N）

〔メ　モ〕

目次要約

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。

- 本資料に記載されている内容は2003年2月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品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，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，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，ソフトウエアおよびこれらに関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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「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。

  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

  特別水準：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

  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，生命維持のための装置またはシステム等

  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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対象者　このマニュアルは，μPD77016 ファミリ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。

μPD77016 ファミリは，μPD7701×ファミリ（μPD77015, 77016, 77017, 77018A, 77019）と，μPD77111 ファミリ（μPD77110, 77111, 77112, 77113A, 77114, 77115），μPD77210 ファミリ（μPD77210, 77213, 77214）の総称です。
ただし，このマニュアルでは，μPD77110, 77111, 77112, 77113A, 77114, 77115, 77210, 77213, 77214 を対象デバイスにしています。

目　的　このマニュアルは，μPD77016 ファミリの応用システムを設計，開発する際にサポートするミドルウエアを，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されています。

- 概　説
- ライブラリ仕様
- インストレーション
- サンプル使用例
- サンプル・プログラム・ソース

読み方　このマニュアルの読者は，電気，論理回路やマイクロコンピュータ，C 言語に関する一般的知識が必要となります。

μPD77111 ファミリのハードウエア機能を知りたいとき
→**μPD77111 ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77210 ファミリのハードウエア機能を知りたいとき
→**μPD77210 ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77016 ファミリの命令機能を知りたいとき
→**μPD77016 ファミリ　ユーザーズ・マニュアル　命令編**を参照してください。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号の名称に上線）
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	：本文中の補足説明
数の表記	：2 進数 … ××××または 0b××××
	10 進数 … ××××または 0t××××
	16 進数 … 0x××××

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。

μPD77016 ファミリに関する資料

資料名 / 品名	パンフレット	データ・シート	ユーザーズ・マニュアル		アプリケーション・ノート	
			アーキテクチャ編	命令編	基本ソフトウエア編	ライブラリ編
μPD77110	U12395J	U12801J	U14623J	U13116J	U11958J	U12021J
μPD77111						
μPD77112						
μPD77113A		U14373J				
μPD77114						
μPD77115		U14867J				
μPD77210		U15203J	U15807J			
μPD77213						
μPD77214		作成予定				

開発ツールに関する資料

資 料 名		資料番号
HSM77016　ユーザーズ・マニュアル		U11602J
WB77016　ユーザーズ・マニュアル	言語編	U10078J
	操作編	U11506J
ID77016　ユーザーズ・マニュアル		U10118J
CC77016　ユーザーズ・マニュアル		U15037J
RX77016　ユーザーズ・マニュアル	機能編	U14397J
	コンフィギュレーション・ツール編	U14404J
RX77016　アプリケーション・ノート	HOST API 編	U14371J

規格に関する資料

資 料 名
ISO/IEC 10918（JPEG 国際標準規格）

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　次

図の目次

図番号　　タイトル，ページ

表の目次

表番号　　タイトル，ページ

第1章 概　　説

1.1　ミドルウエア

ミドルウエアとは，プロセッサの性能をできるだけ引き出せるようにチューニングされたソフトウエア群で，従来，ハードウエアが行っていた処理をソフトウエアで実現したものです。

DSP という高性能プロセッサの出現，そして DSP が手軽にシステムに組み込める環境がそろってきたため，ミドルウエアという概念が現実のものとなってきました。

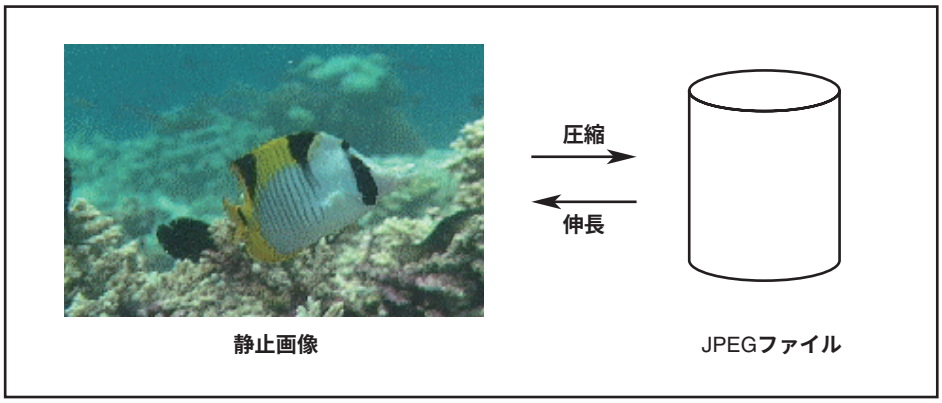
当社は，μPD77016 ファミリ用にマルチメディア・システムを実現する要素技術を提供しています。たとえば音声コーデック，画像データの圧縮／伸長といったミドルウエアをタイムリに提供し，お客様のシステム開発を支援します。

μSAP77016-B22 は，JPEG 画像のデコード／エンコード機能を提供するミドルウエアです。

1.2　JPEG とは

JPEG 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。この規格に関しては ISO/IEC から勧告書（10918-1, 10918-2）が発行されています。

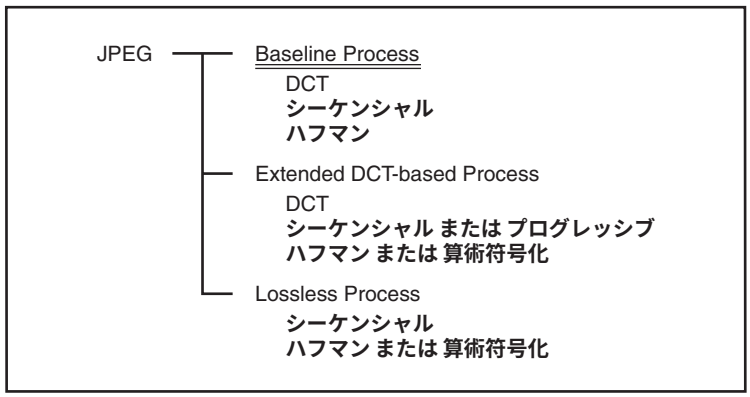
図 1-1　画像の圧縮／伸長



1. 3　JPEG 画像 CODEC

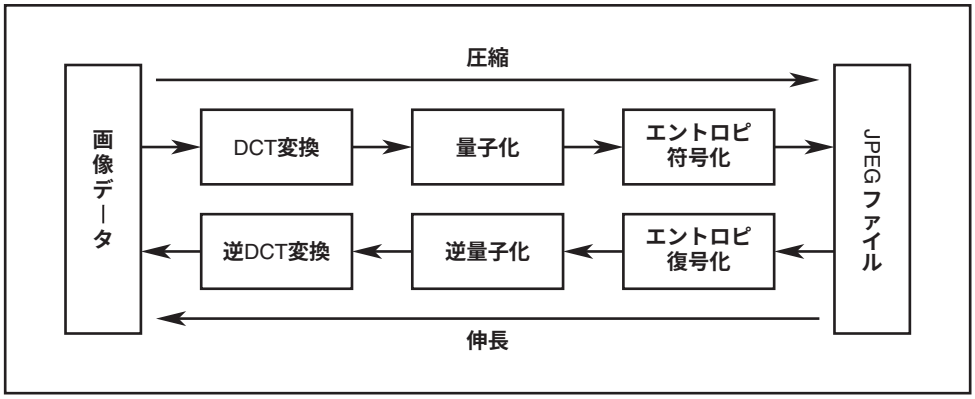
JPEG にはいくつかのバリエーションがあります。最初に画像の概略を表示し，しだいに鮮明な表示へと変化させていくプログレッシブや，画像が圧縮前の画像に完全に復元できるロス・レスなどです。μSAP77016-B22 では，最も一般的な Baseline DCT に対応しています。

図 1-2　JPEG の分類



JPEG 圧縮では，①DCT 変換，②量子化，③エントロピ符号化の 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エントロピ復号化，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 処理の流れ



備考 1. 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。

2. 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ような周波数成分情報を切り落とす処理です。

3. 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 ではハフマン符号化を応用した技術を用いています。

1. 3. 1　色空間（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す。また，伸長した結果を表示する前に，YCbCr から RGB に変換する処理が必要になります。

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−127 〜 +128 となります。

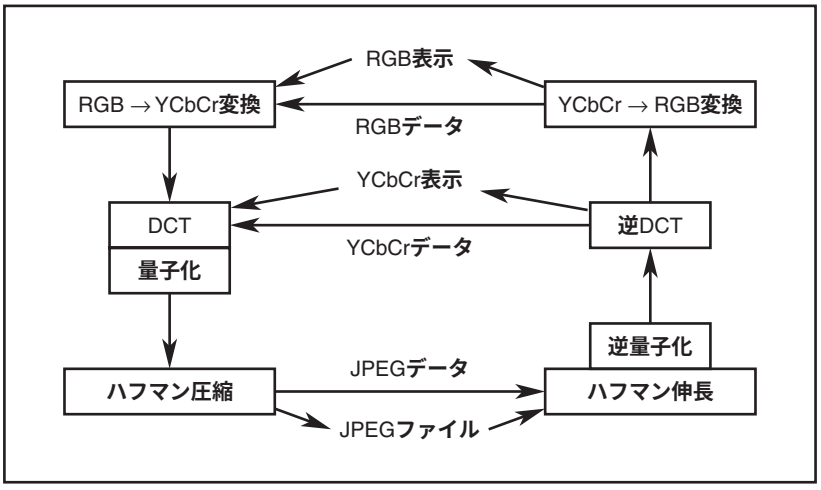
数式 1-1　色空間変換式

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

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.40200 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.7720 & 0 \end{pmatrix} \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix}$$

注意　μSAP77016-B22 は，色空間変換機能（YCbCr ⇔ RGB）には対応していません。機能実装の際は，付録サンプル・プログラム・ソースを参考にしてください。

図 1-4　JPEG 処理概要



1.3.2 サンプリングと MCU

JPEG が処理を行う最小単位を MCU（Minimum Coded Unit）といいます。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いいます。

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 ブロックとるようなものを「サンプル比が 4:1:1」といいます。同様に，1 つの MCU から Y を 2 ブロック，Cb と Cr をそれぞれ 1 ブロックとるようなものを 4:2:2，1 MCU から Y/Cb/Cr をそれぞれ 1 ブロックずつとるようなものを 4:4:4 といいます。

表 1-1　サンプル比と MCU

MCU	サンプル比	ブロック
横 16 ピクセル縦 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
横 32 ピクセル縦 8 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
横 16 ピクセル縦 8 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
横 8 ピクセル縦 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

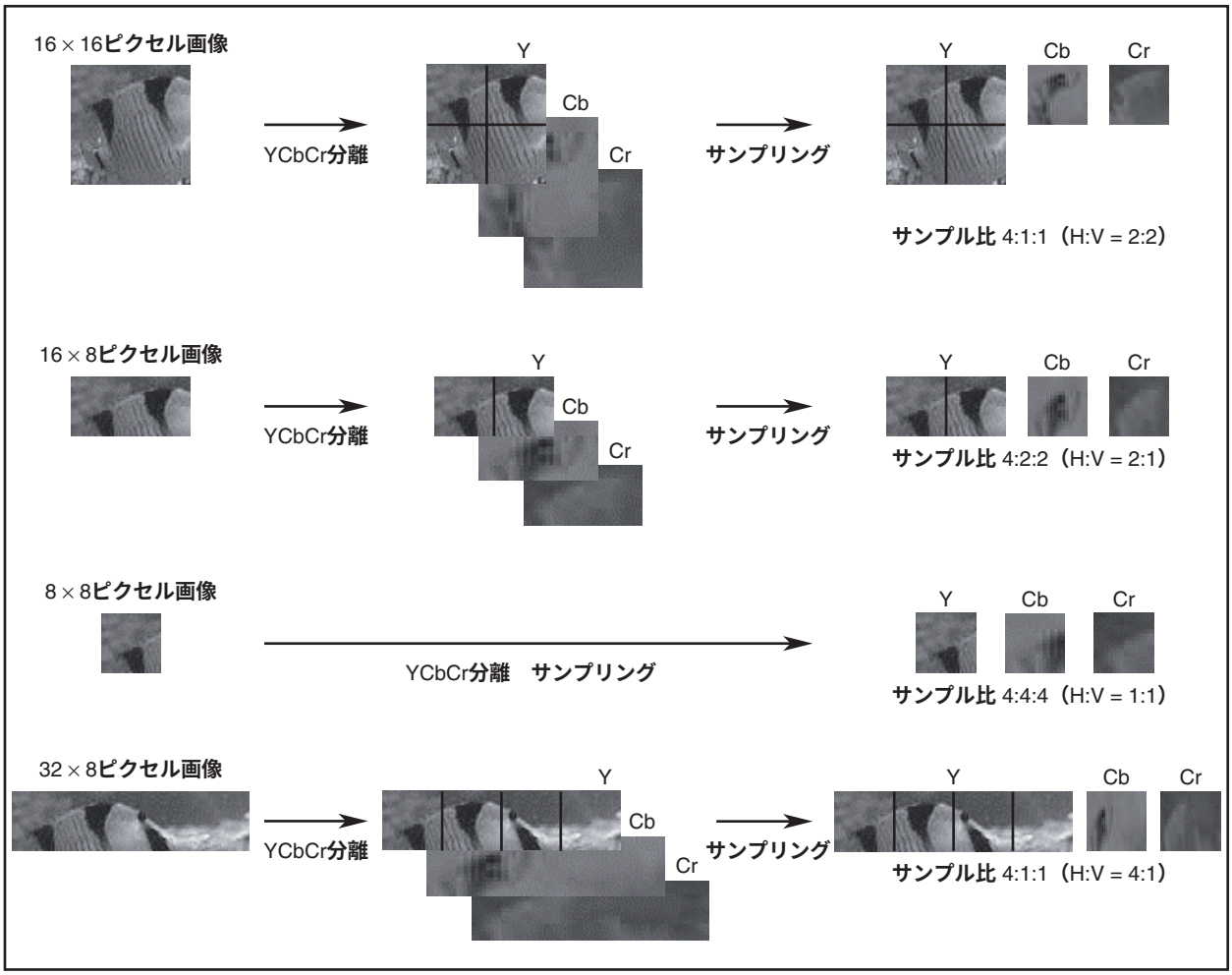
V：MCU の縦方向サンプリング比率

注意　JPEG で許されているサンプル比は表 1-1に記載されたもの以外にもありますが，μSAP77016-B22 では，この表にあるサンプル比だけに対応しています。

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」といいます。

図 1-5　画像サンプリング



JPEG では 4:4:4 よりも 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対して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，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。サンプル比が 4:1:1 のときよりも 4:4:4 のときの方が全体としての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長くかかり，でき上がった JPEG ファイルのサイズも大きくなります。

1.3.3 リスタート・マーカ

JPEG 規格では圧縮したコードの途中に目印となる 2 バイトを挿入するオプションが認められています。

挿入する 2 バイトをリスタート・マーカといいます。リスタート・マーカは，全部で 8 種類あり，値が 0xFF, 0xD0 から 0xFF, 0xD7 までと決められています。

リスタート・マーカは，JPEG の画像を絵の途中から下だけを伸長したい場合に使用します。また，JPEG ファイルを転送するときにファイルの途中でビット誤りが生じてしまったとき，リスタート・マーカを用いない JPEG ファイルだとそれ以降のデータが正しく伸長できません。しかし，その JPEG ファイルがリスタート・マーカを用いたものであれば，次のリスタート・マーカが見つかった時点から伸長を正しく再開できます。

図 1-6　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

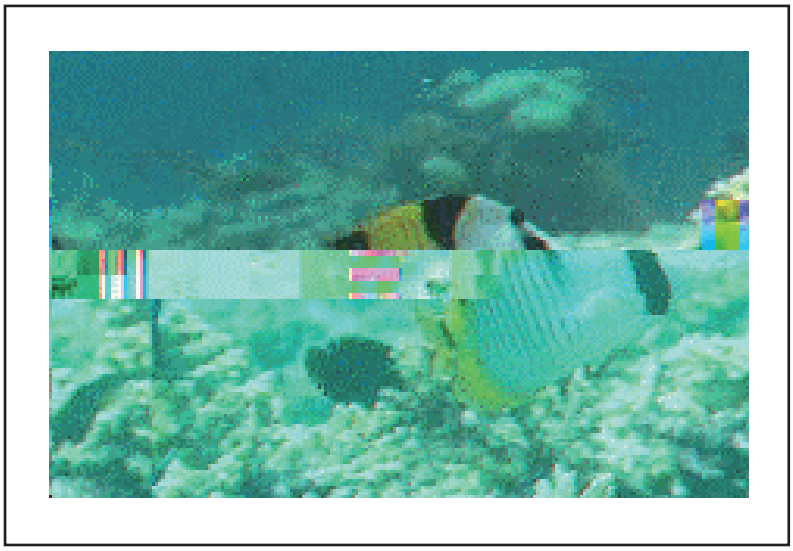
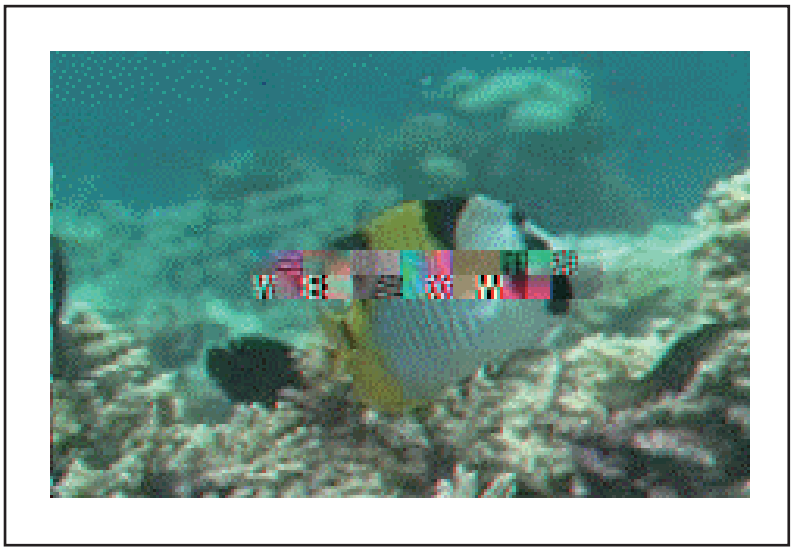


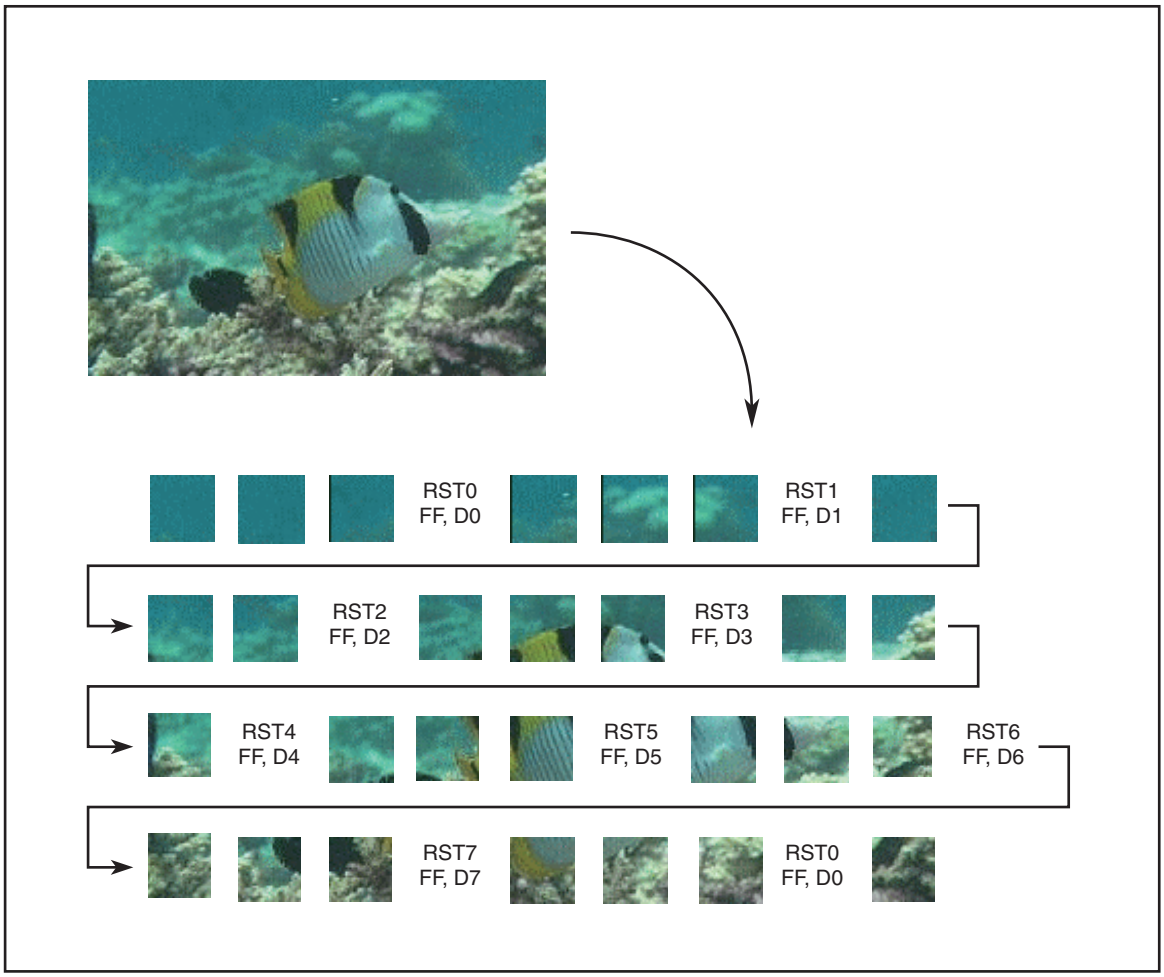
図 1-7　リスタート・マーカを用いて途中から正しく伸長を再開できた例



リスタート・マーカは，m 個の MCU おきに圧縮コード中に挿入されると決められていて，RST0, RST1, RST2, ... , RST7 の順に用いられます。RST7 の次は RST0 に戻ります。この m の値をリスタート・インターバルといいます。

たとえば，リスタート・インターバルが 3 であった場合には 図 1-8 のようなイメージになります。

図 1-8　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定量的に求めることができます。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，次のように計算されます。

（640 × 480）÷（16 × 8）÷ 2 = 1200 個

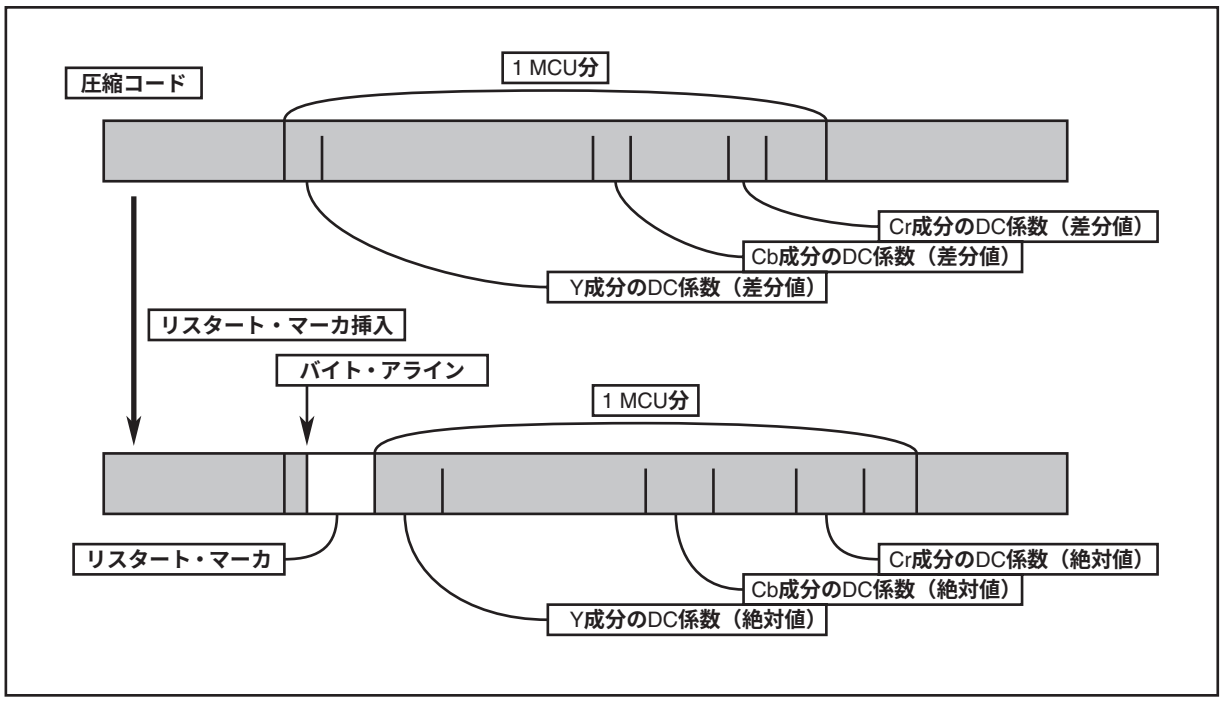
備考　サンプル比が 4:2:2 なので MCU（最小圧縮単位）は 16 × 8 ピクセルとなります。
リスタート・インターバルが 2 なので，MCU が 2 個おきにリスタート・マーカが挿入されます。

リスタート・マーカはバイト境界に置かれます。一方，圧縮コードは，ビット単位で置かれています。リスタート・マーカを挿入するときには，そのアドレスをバイト境界とするためビット“ 1 ”を隙間に埋めます。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。差分値よりも情報量が多いため，一般にリスタート・マーカ直後の DC 成分の圧縮コードの方がそれ以外の DC 成分の圧縮コードよりも大きくなります。

そのため，リスタート・マーカ 1 個当りのバイト数は，マーカそのものの 2 バイトよりも大きく，ほぼ 4 バイト弱となります（多少，ばらつきがあります）。

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・マーカを用いなかった場合に比べて，ファイル・サイズが約 4800 バイト（1200 個×約 4 バイト）余計に必要になります。

図 1-9　リスタート・マーカによるファイル・サイズの増加



1. 3. 4　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 ファイル中（正確には JPEG ヘッダの中）に埋め込んだり，JPEG ファイルから取り出したりすることが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）が用意されています。

アプリケーション・データ・セグメントには 16 種類あり，次のような構造になっています。

図 1-10　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，APPn マーカを用いるかどうかはアプリケーション側に任されています。

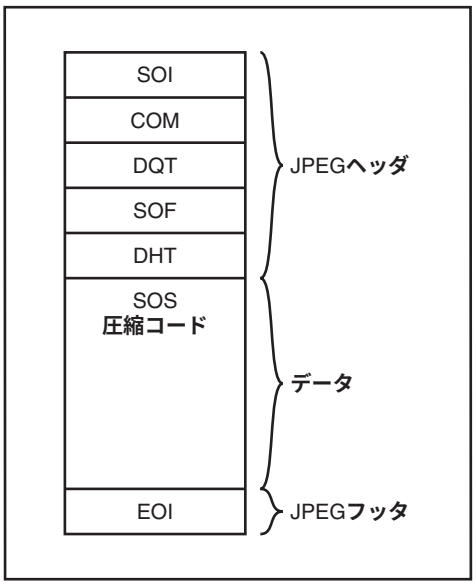
μSAP77016-B22 では圧縮時に APPn セグメントを使うかどうかを指定できるようになっています。

1.4　JPEGのファイル・フォーマット

JPEGのファイルは，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成り立ちます。

ヘッダ部分はすべてバイト単位で書かれています（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が，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 1-11　JPEGファイル・フォーマット



1.4.1　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。この仕様のため，JPEGファイルを0xFFでサーチすることで，その中で用いられているすべてのマーカを検出することができます。ただし，0xFFはヘッダ部分だけではなく，圧縮データの中にも出てきます。この場合には0xFFの直後に圧縮データとしては意味のない0x00を挿入する決まりになっています。「0xFF, 0x00」はマーカではなく，圧縮されたデータ「0xFF」のことを示します。

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。

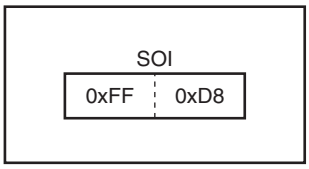
表 1-2　JPEG マーカ

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（ハイアラーキカル・プログレッションの定義）
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）

（1）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。

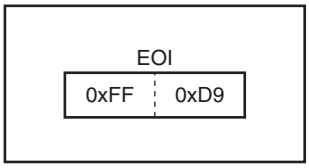
図 1-12　SOI (Start Of Image) マーカ



（2）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。

図 1-13　EOI (End Of Image) マーカ

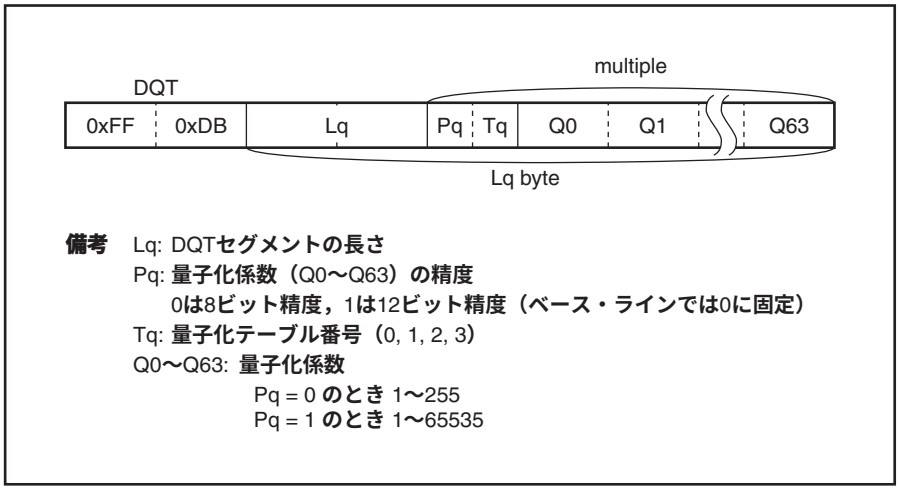


（3）DQT (Define Quantization Table(s)) マーカ

量子化テーブルを定義します。

通常は，輝度成分用（Luminance quantization table）と色差成分用（Chrominance quantization table）の 2 つを持っています。

図 1-14　DQT (Define Quantization Table(s)) マーカ

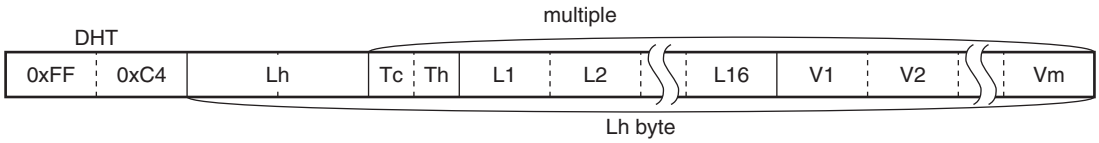


備考　Lq: DQTセグメントの長さ
Pq: 量子化係数（Q0～Q63）の精度
0は8ビット精度，1は12ビット精度（ベース・ラインでは0に固定）
Tq: 量子化テーブル番号（0, 1, 2, 3）
Q0～Q63: 量子化係数
Pq = 0 のとき 1～255
Pq = 1 のとき 1～65535

(4) DHT (Define Huffman Table(s)) マーカ

ハフマン・テーブルを定義します。

図 1-15　DHT (Define Huffman Table(s)) マーカ



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

たとえば，L1～L16 が，「00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00」の場合，次のようになります。

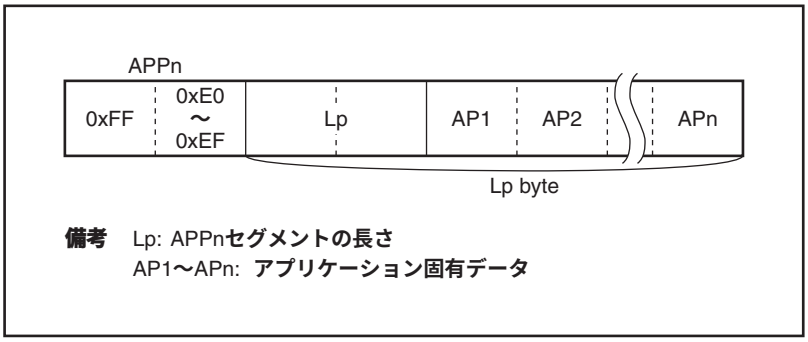
1 ビット長のコードが，0 個
2 ビット長のコードが，00 の 1 個
3 ビット長のコードが，010, 011, 100, 101, 110 の 5 個
4 ビット長のコードが，1110 の 1 個
5 ビット長のコードが，11110 の 1 個
6 ビット長のコードが，111110 の 1 個
7 ビット長のコードが，1111110 の 1 個
8 ビット長のコードが，11111110 の 1 個
9 ビット長のコードが，111111110 の 1 個
それ以外のコード長はないことを意味します。

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 2 番目の圧縮コードなので V2 です。

(5) APPn (Reserved for APPlication segments) マーカ

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用できるセグメントです。たいていの場合は 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。

図 1-16　APPn (Reserved for APPlication segments) マーカ



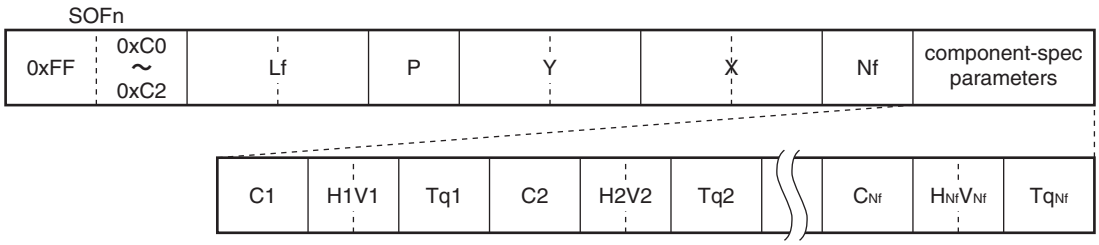
備考　Lp: APPnセグメントの長さ
AP1～APn: アプリケーション固有データ

(6) SOFn (Start Of Frame) マーカ

JPEG では JPEG ファイルから SOI マーカと EOI マーカを除いた中身の部分をフレームといいます。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 セグメントは特別にフレーム・ヘッダともいいます。

また，JPEG では Y/Cb/Cr などの色要素をコンポーネントといいます。

図 1-17　SOFn (Start Of Frame) マーカ



備考　SOFn:　SOF0のとき0xFF 0xC0（ベース・ライン）
SOF1のとき0xFF 0xC1（拡張ベース・ライン）
SOF2のとき0xFF 0xC2（プログレッシブ）
Lf:　SOFnセグメントの長さ
P:　DCT係数の精度 8: 8ビット精度
12: 12ビット精度
Y:　画像の縦サイズ（ピクセル数）
X:　画像の横サイズ（ピクセル数）
Nf:　コンポーネント数　1: モノクロ（単色）
3: カラー（3色）
4: CMYK, YCCK（4色）
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ル比
Vn:　n番目のコンポーネントの縦方向サンプル比
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

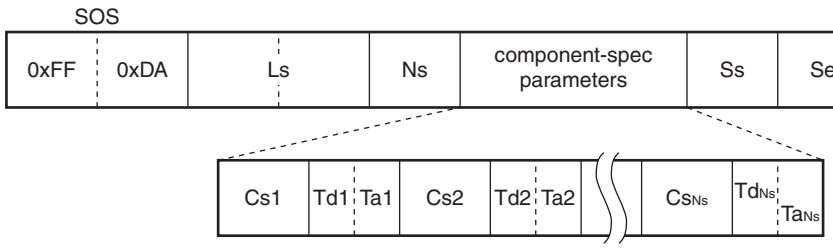
たとえば，このセグメントの内容が，「0xFF 0xC0, 0x00, 0x11, 0x08, 0x00, 0x90, 0x00, 0xE0, 0x03, 0x01, 0x22, 0x00, 0x02, 0x11, 0x01, 0x03, 0x11, 0x01」の場合，次のようになります。

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

(7) SOS (Start Of Scan) マーカ

SOS セグメントはスキャン・ヘッダともいいます。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，ハフマン・テーブル番号の指定などを行っています。

図 1-18　SOS (Start Of Scan) マーカ



備考　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のDC係数のハフマン・テーブル番号
ベース・ライン：0 または 1
拡張ベース・ラインおよびプログレッシブ：0, 1, 2, 3
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 または 1
拡張ベース・ラインおよびプログレッシブ：0, 1, 2, 3
Ss: DCT係数の開始番号
ベース・ラインおよび拡張ベース・ライン：0 に固定
プログレッシブ：0〜63
Se: DCT係数の終了番号
ベース・ラインおよび拡張ベース・ライン：63 に固定
プログレッシブ：Ss〜63（ただし，Ss = 63のときはSe = 0に固定）
Ah: DCT係数の上位ビット
ベース・ラインおよび拡張ベース・ライン：0 に固定
プログレッシブ：0〜13
Al: DCT係数の下位ビット
ベース・ラインおよび拡張ベース・ライン：0 に固定
プログレッシブ：0〜13

（8）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度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・マーカが挿入されることになります。

図 1-19　DRI (Define Restart Interval) マーカと RSTm (REStart interval termination) マーカ

たとえば，4 個置きにマーカを入れる場合には，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 係数を差分値として圧縮するため，それぞれの 8 × 8 ブロックを伸長するために，1 つ前の DC 係数の値が必要です。リスタート・マーカの直後の DC 係数は，0 との差分をとります。

1.5 製品概要

1.5.1 ライブラリ構成

μSAP77016-B22 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 1-3　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
μPD77016 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）	

(1) VRAM と座標の（x, y）

VRAM アクセスの部分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。この部分は，ユーザがシステムにあわせて記述する必要があります。

(2) 量子化テーブル

圧縮処理では，輝度成分用と色差成分用の 2 面の量子化テーブルを指定してください。この量子化テーブルは，ディフォールトでも用意されています。

また，Quality パラメータを用意し，このパラメータに 0～100 の値を設定することで，量子化テーブルの値は一律に定数倍され，要素の値が 1～255 の範囲内に収まるように加工されます。

指定した量子化テーブルを加工しないで用いる場合は Quality パラメータに値 50 を設定してください。

伸長処理では，JPEG ファイル内の DQT ヘッダに書かれている値を用います。

(3) ハフマン・テーブル

圧縮処理では，輝度 DC 用／輝度 AC 用／色差 DC 用／色差 AC 用の 4 面のテーブルを指定してください。このハフマン・テーブルはディフォールトでも用意されています。

伸長処理では，JPEG ファイル内の DHT ヘッダに書かれている値を用います。

(4) リスタート・マーカ

圧縮処理では，リスタート・マーカを用いる／用いないの選択ができます。用いる場合，リスタート・インターバルの値を変えることができます。

伸長処理では，JPEG ファイル内の DRI セグメントの値を用います。

(5) APPn セグメント，COM セグメント

圧縮処理では，APPn セグメント，COM セグメントを挿入するように指定することができます。

伸長処理では，APPn セグメント，COM セグメントはスキップされます。

1.5.2 ライブラリの特徴

(1) サンプル比

圧縮，伸長処理ともに次の 4 つのサンプル比をサポートしています。

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

(2) JPEG ファイル格納用バッファ

圧縮，伸長処理いずれも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。

JPEG ファイルのサイズは一定ではないので，μSAP77016-B22 では指定した JPEG バッファの最後までデータが来た場合に処理を中断し，バッファ処理（圧縮ではバッファの退避／伸長ではバッファの更新）後，ルーチンを再度呼び出すことで処理が再開される仕様になっています。

JPEG ファイル用のバッファのサイズは 1 ワード以上，任意のワード数に設定可能ですが，圧縮／伸長処理とバッファ処理の間には必ずレジスタ・ディスパッチ処理が入りますので，この回数があまり増えないように，十分なサイズを確保してください。

(3) ワーク・エリア（MCU データ格納用バッファ）

圧縮，伸長処理いずれもライブラリを実行するために YCbCr データを格納する MCU バッファが必要です。

YCbCr データはワーク・エリアに確保されます。ワーク・エリアのサイズは 768 ワード（固定）になります。

μSAP77016-B22 では，1 MCU 単位で YCbCr データの入出力を行います。サンプル比によってワーク・エリア内の YCbCr データの配置アドレスが変わるので注意してください。

1.5.3 対応ファイル・フォーマット

表 1-4　対応ファイル・フォーマット一覧

ファイル・フォーマット		基本ライブラリ
符号化方式	ベース・ライン	○
	拡張ベース・ライン	×
	プログレッシブ	×
	その他	×
符号化順序	インタリーブ	○
	ノン・インタリーブ	×
DNL マーカ対応		×
サンプル比		4:1:1 (H:V = 2:2) 4:1:1 (H:V = 4:1) 4:2:2 (H:V = 2:1) 4:4:4 (H:V = 1:1)
コンポーネント		3 色
量子化テーブル		4 面[注1]
ハフマン・テーブル		4 面
ロスレス方式（可逆方式）		×
ハイアラーキカル方式		×
COM（コメント・マーカ）		○[注2]
APPn（アプリケーション・マーカ）		○[注2]
DRI/RSTm（リスタート・マーカ）		○
DNL（ライン数再定義マーカ）		×

注 1. 伸長処理のみ 4 面対応，圧縮処理は 2 面

2. 伸長処理では読み飛ばされます。

1.5.4 動作環境

(1) 動作対象 DSP

μPD77110, 77111, 77112, 77113A, 77114, 77115, 77210, 77213, 77214

(2) 必要メモリ

μSAP77016-B22 は表 1-5のメモリ・サイズを使用します

表 1-5 必要メモリ・サイズ

メモリ	種別		必要メモリ・サイズ（word）	
			エンコーダ	デコーダ
命令メモリ	ー		1987	2047
X メモリ	RAM	スタティック領域	37	35
		スクラッチ領域	768 + 1 注	768 + 1 注
	ROM		744	98
Y メモリ	RAM	スタティック領域	1828	2136
		スクラッチ領域	0	0
	ROM		0	10

注 スクラッチ領域は，入出力バッファ（JPEG バッファおよびワーク・エリア（MCU バッファ））の領域です。入出力バッファについては，2.9 入出力バッファを参照してください。
JPEG バッファを 1 word，MCU バッファを 768 word としてありますが，JPEG バッファには可能な限り大きなメモリ領域を割り当ててください。

備考 命令メモリの 1 ワードは，32 ビットです。X メモリ，Y メモリの 1 ワードは，16 ビットです。

(3) ソフトウエア・ツール（Windows®版）

表 1-6 ソフトウエア・ツール

対象 DSP	ソフトウエア・ツール
μPD77110 ファミリ	WB77016（ワークベンチ（アセンブラ／リンカ））
	HSM77016（ハイスピード・シミュレータ）
	ID77016（ディバッガ）
μPD77210 ファミリ	Atair Developer Studio（ワークベンチ（アセンブラ／リンカ））
	μPD7721x ハイスピード・シミュレータ
	μPD7721x ディバッガ

備考 これらの DSP 用ソフトウエア・ツールは Atair 社製です。

1.5.5 性　能

μSAP77016-B22 を使用して圧縮／伸長したときの演算時間（実測値）を表 1-7に示します。

・測定条件

シミュレータ：μPD7721x ハイスピード・シミュレータ

DSP：μPD77213（120 MHz）

評価画像：任意の画像ファイル（ビットマップ形式，320×240 ピクセル，24 ビット RGB）を使用。

評価結果：評価画像を圧縮／伸長したときの時間を測定します。画像データ入出力関数，および割り込みハンドラなどの時間は含みません。

圧縮パラメータ： jpegEnc_quality = 50

jpegEnc_restartInterval = 0

JPEGBuffer = 1000

表 1-7　JPEG 画像 CODEC の演算量

コーデック・モード	サンプル・レート [ms]			
	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)
エンコード	101.2	72.1	57.6	57.3
デコード	112.0	81.5	66.0	65.9

備考　この値は，当社評価時の実測値です。入力される画像によって変動します。この値は，最悪値を保証するものではありません。

1.5.6 ディレクトリ構成

μSAP77016-B22 のパッケージ内容を示します。

- library
  - jpegdec_ram.lib ：RAM品種用JPEGデコード・ライブラリ
  - jpegdec_rom.lib ：ROM品種用JPEGデコード・ライブラリ
  - jpegenc_ram.lib ：RAM品種用JPEGエンコード・ライブラリ
  - jpegenc_rom.lib ：ROM品種用JPEGエンコード・ライブラリ
  - jpeg_dec.h ：RAM/ROM品種用JPEGデコード・ライブラリ・ヘッダ・ファイル
  - jpeg_enc.h ：RAM/ROM品種用JPEGエンコード・ライブラリ・ヘッダ・ファイル
- sample
  - dsp_jpeg_sample.AtairPWS ：サンプル・プロジェクト・ワーク・スペース
  - dsp_jpeg_sample.AtairWS ：サンプル・ワーク・スペース
  - upd77210.model ：μPD77210モデル・ファイル
  - decoder
    - decoder.AtairPRJ ：デコード用サンプル・プロジェクト・ファイル
    - hostrd8.tmg ：データ出力用タイミング・ファイル
    - hostwr8.tmg ：データ入力用タイミング・ファイル
    - lib （ライブラリ格納用フォルダ）
    - rel （中間ファイル出力用フォルダ）
    - sources
      - decoder_sample.asm ：デコード用サンプル・ソース・ファイル
      - yuvtorgb.asm ：YUV → RGB変換用サンプル・ソース・ファイル
      - macro.h ：マクロ定義用ヘッダ・ファイル
      - upd7721x.h ：μPD7721x用ヘッダ・ファイル
  - encoder
    - encoder.AtairPRJ ：エンコード用サンプル・プロジェクト・ファイル
    - hostrd8.tmg ：データ出力用タイミング・ファイル
    - hostwr8.tmg ：データ入力用タイミング・ファイル
    - lib （ライブラリ格納用フォルダ）
    - rel （中間ファイル出力用フォルダ）
    - sources
      - encoder_sample.asm ：エンコード用サンプル・ソース・ファイル
      - rgbtoyuv.asm ：RGB → YUV変換用サンプル・ソース・ファイル
      - macro.h ：マクロ定義用ヘッダ・ファイル
      - upd7721x.h ：μPD7721x用ヘッダ・ファイル
  - tool
    - bmp2dat11.exe ：bmp → dat変換
    - bmp2dat22.exe ：bmp → dat変換
    - dat2jpg.exe ：dat → jpeg変換
    - jpg2dat.exe ：jpeg → dat変換
    - dat2bmp11.exe ：dat → bmp変換
    - dat2bmp22.exe ：dat → bmp変換
    - readme.txt ：データ変換用ツールの説明書
- readme.txt ：使用上の注意

各ディレクトリの概要は次のとおりです。

- library

  ライブラリ・ファイル，ヘッダ・ファイルを格納しています。

- sample

  DSP ツール用のプロジェクト・ファイル，サンプル・プログラムのソース・ファイルおよびタイミング・ファイルを格納しています。また，ビットマップ画像および JPEG 画像を入力用の DAT ファイルにするための実行ファイルも格納しています。

第2章 ライブラリ仕様

2.1 ライブラリ概要

μSAP77016-B22 では，表 2-1の 4 つの関数を用意しています。

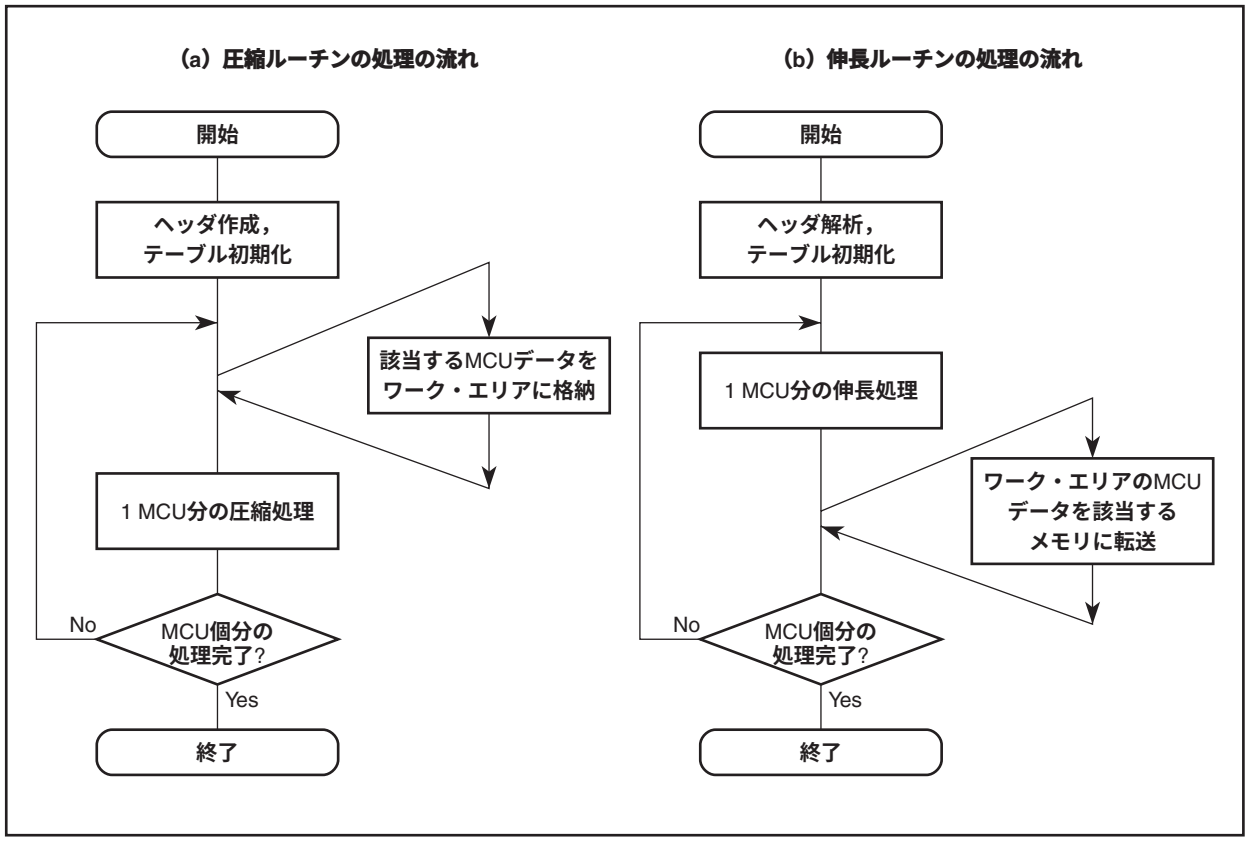
表 2-1 ライブラリ関数一覧

関 数 名	機　　能
jpegEnc_Compress	圧縮処理
jpegEnc_GetVersion	エンコーダのバージョン情報取得
jpegDec_Decompress	伸長処理
jpegDec_GetVersion	デコーダのバージョン情報取得

2. 2　処理フロー

μSAP77016-B22 ライブラリでは，画像データを MCU 単位で処理します。圧縮処理では MCU 単位で，画像入力→DCT 変換→量子化→ハフマン符号化を行います。伸長処理では MCU 単位で，ハフマン復号化→逆量子化→逆 DCT 変換→画像出力を行います。

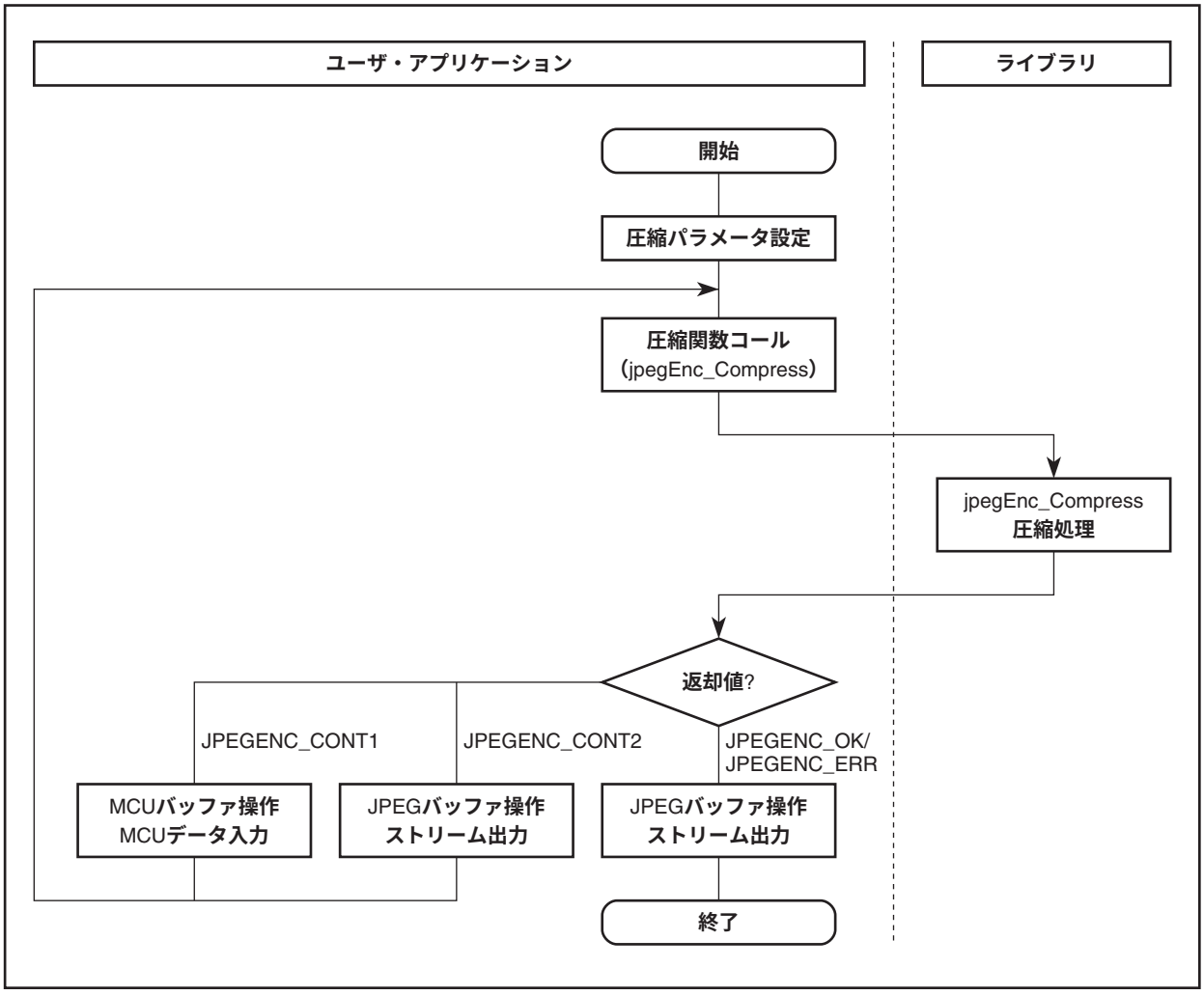
図 2-1　μSAP77016-B22 の処理の流れ



2. 2. 1　圧縮処理フロー

ユーザ・アプリケーションを含めた圧縮処理フローを図 2-2に示します。

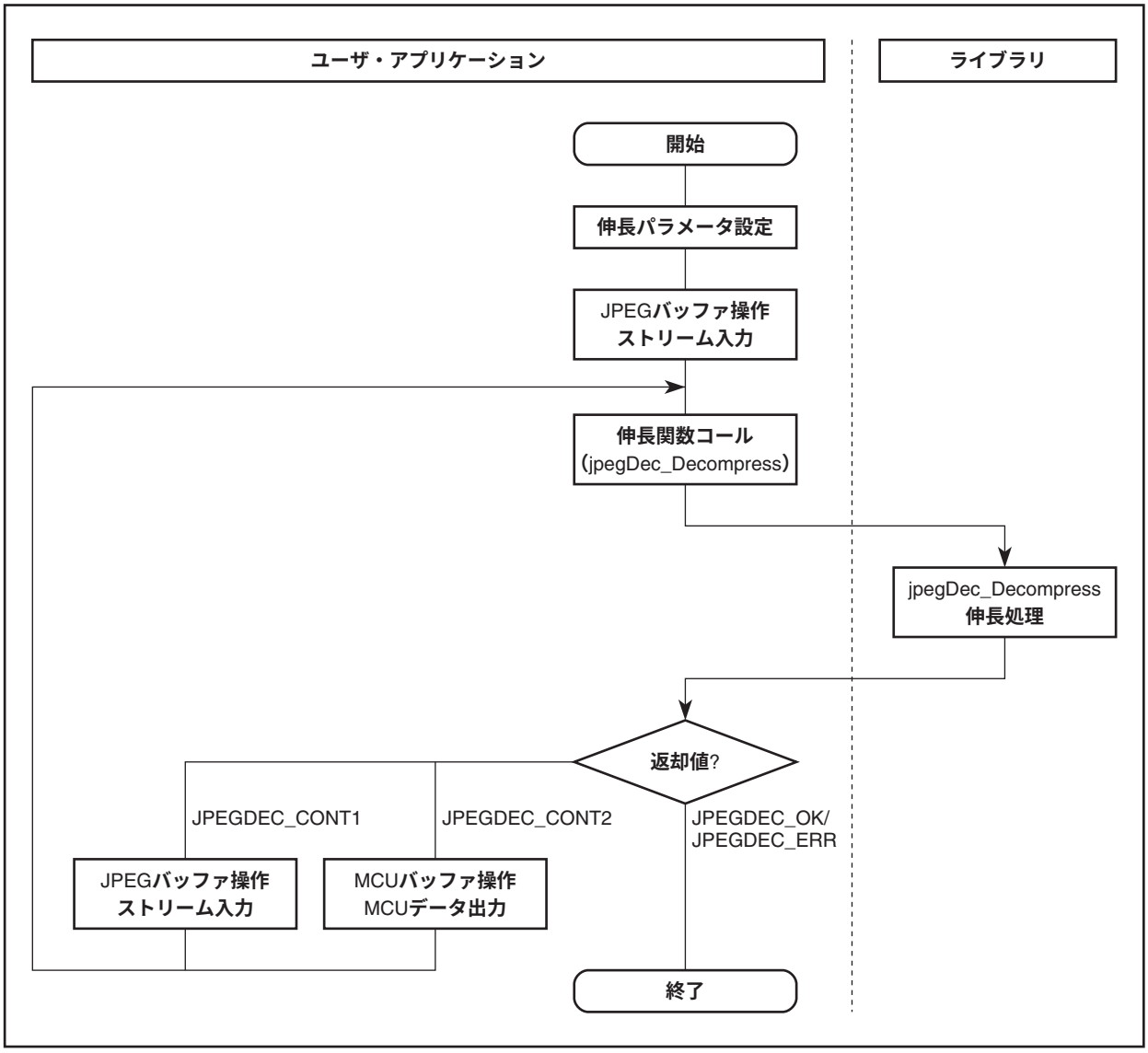
図 2-2　圧縮処理フロー



2. 2. 2　伸長処理フロー

ユーザ・アプリケーションを含めた伸長処理フローを図 2-3に示します。

図 2-3　伸長処理フロー



2. 3　jpegEnc_Compress 関数

jpegEnc_Compress 関数は，設定されたパラメータに従って，JPEG エンコード（1MCU）を行います。

【 分　類 】圧縮処理系

【関 数 名】jpegEnc_Compress

【機能概要】JPEG 圧縮処理

【 形　式 】call jpegEnc_Compress

【引 き 数】

変数名	メモリ	説　明
jpegEnc_returnValue	1 word:X	ステータス（最初のエンコード時は 0 にしてください。）
jpegEnc_restartInterval	1 word:X	JPEG ファイルのリスタート・インターバルの挿入間隔の設定
jpegEnc_imageWidth	1 word:X	水平方向の画像サイズ（ピクセル数）
jpegEnc_imageHeight	1 word:X	垂直方向の画像サイズ（ピクセル数）
jpegEnc_sampleRatio	1 word:X	サンプル比
jpegEnc_quality	1 word:X	量子化パラメータ
jpegEnc_compNumC1	1 word:X	コンポーネント・ナンバ 1（Y 成分）
jpegEnc_compNumC2	1 word:X	コンポーネント・ナンバ 2（Cb 成分）
jpegEnc_compNumC3	1 word:X	コンポーネント・ナンバ 3（Cr 成分）
jpegEnc_workAreaBptr	1 word:X	ワーク・エリアの先頭アドレス
jpegEnc_jpegBuffBptr	1 word:X	JPEG データ・バッファの先頭アドレス
jpegEnc_jpegBuffEptr	1 word:X	JPEG データ・バッファの終端アドレス
jpegEnc_qTblYBptr	1 word:X	輝度成分用量子化テーブル先頭アドレス
jpegEnc_qTblCBptr	1 word:X	色差成分用量子化テーブル先頭アドレス
jpegEnc_dHTDCYBptr	1 word:X	輝度 DC 用ハフマン・テーブル先頭アドレス
jpegEnc_dHTDCCBptr	1 word:X	色差 DC 用ハフマン・テーブル先頭アドレス
jpegEnc_dHTACYBptr	1 word:X	輝度 AC 用ハフマン・テーブル先頭アドレス
jpegEnc_dHTACCBptr	1 word:X	色差 AC 用ハフマン・テーブル先頭アドレス
jpegEnc_aPPTblBptr	1 word:X	APP セグメント構造体の先頭アドレス

【返 り 値】

変数名	メモリ	説　明
jpegEnc_returnValue	1 word:X	ステータス
jpegEnc_fileSize	2 word:X	ファイル・サイズ
jpegEnc_errorState	1 word:X	エラー・コード

ステータスが JPEGENC_ERROR の場合は，jpegEnc_errorState にエラー・コードが格納されます。

【使用レジスタ】すべてのレジスタ

【ハードウエア・リソースメント】

最大スタック・レベル　8

最大ループ・スタック・レベル　2

最大リピート回数　16

サイクル数（任意の画像ファイルの 1MCU の圧縮に必要なサイクル数）　41052

備考　この値は，当社評価時の実測値です。入力される画像によって変動します。この値は，最大サイクル数を保証するものではありません。

2. 3. 1　jpegEnc_Compress 関数の引き数

(1) jpegEnc_returnValue

返り値の初期化を行います。

圧縮処理を起動する前に，一度だけ 0 を設定し，初期化してください。

変数名	メモリ	設定値
jpegEnc_returnValue	1 word:X	0

注意　処理を中断したあと再開する場合，μSAP77016-B22 はこの jpegEnc_returnValue の値を見て，最初の起動か，それとも再開かを判別します。このため，起動前の設定以外では初期化しないでください。

また，最初の jpegEnc_Compress 関数の呼び出し時には，MCU データをワーク・エリアに配置しないでください。

(2) jpegEnc_restartInterval

リスタート・マーカの挿入間隔を指定します。

設定値をリスタート・インターバルとし，この個数の MCU ごとに RSTm マーカを挿入します。

変数名	メモリ	設定値
jpegEnc_restartInterval	1 word:X	0 ～ 65535

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

リスタート・インターバルを有効にした場合，RSTm マーカにより，JPEG ファイルのサイズはリスタート・マーカ 1 個につき，4 バイト弱増えます。

(3) jpegEnc_imageWidth，jpegEnc_imageHeight

圧縮する画像の縦横のピクセル数を指定します。

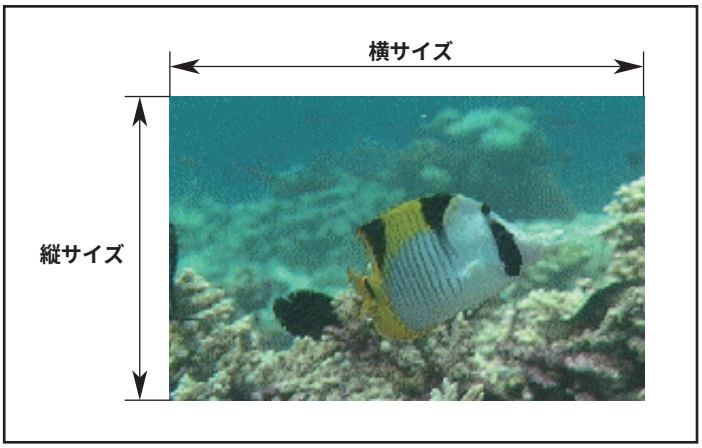
変数名	メモリ	設定値
jpegEnc_ImageWidth	1 word:X	8 ～ 65535
jpegEnc_ImageHeight	1 word:X	8 ～ 65535

値の単位はピクセル数です。設定できる値には次のような制限があります。

表 2-2　横サイズ／縦サイズの制限

サンプル比	jpegEnc_imageWidth	jpegEnc_imageHeight
4:1:1 (H:V = 2:2)	16 の倍数	16 の倍数
4:1:1 (H:V = 4:1)	32 の倍数	8 の倍数
4:2:2 (H:V = 2:1)	16 の倍数	8 の倍数
4:4:4 (H:V = 1:1)	8 の倍数	8 の倍数

図 2-4　画像の横サイズ／縦サイズ



(4) jpegEnc_sampleRatio

サンプル比を設定します。

μSAP77016-B22 がサポートしているサンプル比は次の 4 種類です。

変数名	メモリ	サンプル比	設定値（シンボル名）
jpegEnc_sampleRatio	1 word:X	4:1:1 (H:V = 2:2)	0x22（JPEGENC_SAMPLING22）
		4:1:1 (H:V = 4:1)	0x41（JPEGENC_SAMPLING41）
		4:2:2 (H:V = 2:1)	0x21（JPEGENC_SAMPLING21）
		4:4:4 (H:V = 1:1)	0x11（JPEGENC_SAMPLING11）

(5) jpegEnc_quality

量子化パラメータの値を設定することで，容易に量子化テーブルの値を変更できるようになっています。

量子化パラメータの値によって，画質，JPEG ファイル・サイズが影響を受けます。

変数名	メモリ	設定値
jpegEnc_quality	1 word:X	0 ～ 100

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 2-3　Quality パラメータの設定値と画質の関係

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

(6) jpegEnc_compNumC1，jpegEnc_compNumC2，jpegEnc_compNumC3

コンポーネント・ナンバを指定します。

変数名	メモリ	設定値
jpegEnc_compNumC1	1 word:X	0 ～ 255
jpegEnc_compNumC2	1 word:X	0 ～ 255
jpegEnc_compNumC3	1 word:X	0 ～ 255

通常は，jpegEnc_compNumC1 = 1, jpegEnc_compNumC2 = 2, jpegEnc_compNumC3 = 3 **に設定します。**
重複する番号を選ぶことはできません。

(7) jpegEnc_workAreaBptr

ワーク・エリアの先頭アドレスを設定します。

変数名	メモリ	設定値
jpegEnc_workAreaBptr	1 word:X	**ワーク・エリアの先頭アドレス**

注意　ワーク・エリアのサイズは 768 ワード固定です。
ワーク・エリアは，X メモリ上に存在する必要があります。

(8) jpegEnc_jpegBuffBptr，jpegEnc_jpegBuffEptr

JPEG **バッファの先頭アドレスと終端アドレスを設定します。**

変数名	メモリ	設定値
jpegEnc_jpegBuffBptr	1 word:X	JPEG **バッファの先頭アドレス**
jpegEnc_jpegBuffEptr	1 word:X	JPEG **バッファの先頭アドレス＋バッファ・サイズ**－1

注意　JPEG バッファ・サイズは，1 以上，65535 以下に設定してください。
JPEG バッファは，X メモリ上に存在する必要があります。

(9) jpegEnc_qTblYBptr，jpegEnc_qTblCBptr

量子化テーブルの先頭アドレスを設定します。

変数名	メモリ	設定値
jpegEnc_qTblYBptr	1 word:X	**輝度成分用量子化テーブルの先頭アドレス**
jpegEnc_qTblCBptr	1 word:X	**色差成分用量子化テーブルの先頭アドレス**

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

ディフォールトの量子化テーブルを使用する場合は，jpegEnc_defaultQTblYBptr，jpegEnc_defaultQTblBptr **をそれぞれ指定してください。**

図 2-5　ディフォールトの量子化テーブル

(a) 輝度成分のディフォールト量子化テーブル

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

(b) 色差成分のディフォールト量子化テーブル

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

(10) jpegEnc_dHTDCYBptr，jpegEnc_dHTDCCBptr，jpegEnc_dHTACYBptr，jpegEnc_dHTACCBptr

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

変数名	メモリ	設定値
jpegEnc_dHTDCYBptr	1 word:X	**輝度** DC **用ハフマン・テーブルの先頭アドレス**
jpegEnc_dHTDCCBptr	1 word:X	**色差** DC **用ハフマン・テーブルの先頭アドレス**
jpegEnc_dHTACYBptr	1 word:X	**輝度** AC **用ハフマン・テーブルの先頭アドレス**
jpegEnc_dHTACCBptr	1 word:X	**色差** AC **用ハフマン・テーブルの先頭アドレス**

輝度成分用と色差成分用にそれぞれ 64 **バイトの量子化テーブルを指定してください。**

(a) μ**SAP77016-B22 で用意されているハフマン・テーブルを使用する**

μSAP77016-B22 **で用意されているテーブルを用いるには，次の配列名の先頭を指定してください。**

- **輝度成分** DC **用：** jpegEnc_defaultDHTDCYBptr
- **輝度成分** AC **用：** jpegEnc_defaultDHTACYBptr
- **色差成分** DC **用：** jpegEnc_defaultDHTDCCBptr
- **色差成分** AC **用：** jpegEnc_defaultDHTACCBptr

(b) ユーザが独自に用意したハフマン・テーブルを使用する

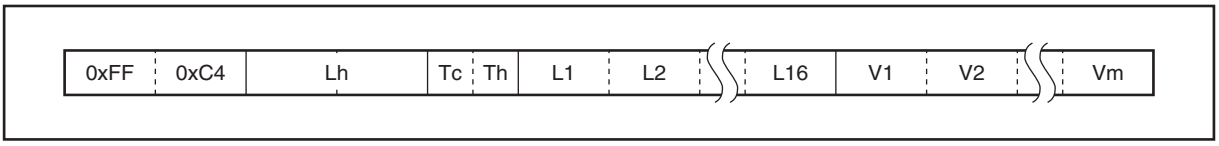
用意されたテーブルを用いずに，ユーザが独自に用意したハフマン・テーブルと差し替えることもできます。

しかし，適切でないハフマン・テーブルを指定した場合，画像によっては正常に圧縮されない場合があります。また，正常に圧縮されなくても，圧縮ルーチンは正常終了（返り値 JPEGENC_OK）します。

このため，ハフマン・テーブルをユーザが作成する際は次の 2 つの条件を満たすようにしてください。

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

図 2-6　DHT セグメント



DHT セグメントの L1～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。

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，μSAP77016-B22 では，そのチェックを行いません。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。

表 2-4　DC/AC 成分の値とビット長

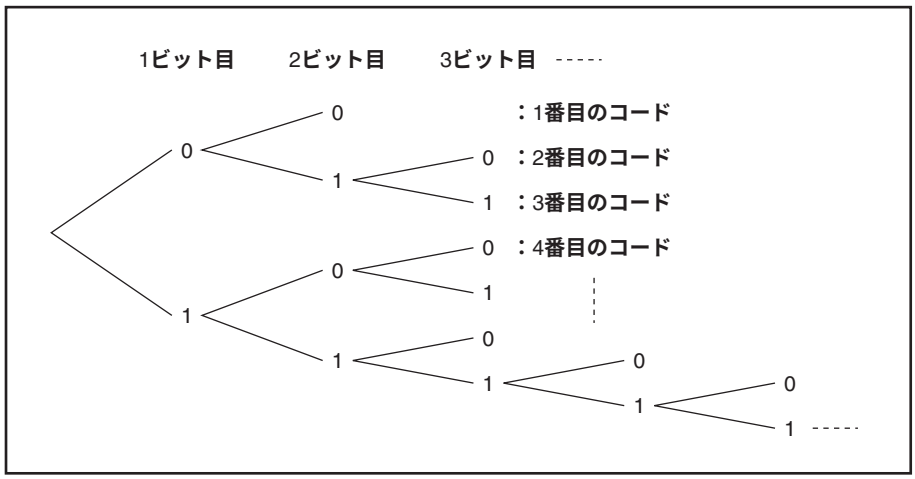
成分の値	カテゴリ
0	0
−1, 1	1
−3, −2, 2, 3	2
−7～−4, 4～7	3
−15～−8, 8～15	4
−31～−16, 16～31	5
−63～−32, 32～63	6
−127～−64, 64～127	7
−255～−128, 128～255	8
−511～−256, 256～511	9
−1023～−512, 512～1023	10
−2047～−1024, 1024～2047	11

たとえば，L1～L16 が「00, 01, 05, 01, 01, 01, 01, 01, 01, 00, 00, 00, 00, 00, 00, 00」の場合，次の意味になります。

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 2-7　圧縮コードの値の確定



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「0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B (m = 12)」の場合，次の意味になります。

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 0x0～0xB です。

一般には，画像を圧縮した場合，カテゴリ 2 とカテゴリ 1 のビット長が最も多く分布し，カテゴリ 11 に近づくに従って出現確率が低くなります。画像によっては，カテゴリ 8, 9, 10, 11 のビット長がまったく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。

しかし，カテゴリ 8 以上の値がないハフマン・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に，μSAP77016-B22 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，圧縮コードを正常に埋め込んだものと解釈して正常終了します。この 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，μSAP77016-B22 に付属の AC 成分用のハフマン・テーブル（jpegEnc_defaultDHTACYBptr）では，次のようになっています。

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

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

この例の場合には，次の意味になります。

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

AC 係数の場合も DC 係数の場合と同様に，カテゴリの小さいものほど出現確率は高く，大きいものほど出現頻度が小さくなります。また，ゼロ・ランは 0 のものが最も多く，大きくなるにつれて出現確率が低下します。そこで，ゼロ・ランとカテゴリが大きい部分に相当する圧縮コードを持たないようなハフマン・テーブルを作ることが考えられます。

しかし，μSAP77016-B22 では，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。

（11）jpegEnc_aPPTblBptr

APP **ヘッダを埋め込みます。**

APP **ヘッダ構造体の先頭アドレスを指定します。**

変数名	メモリ	設定値
jpegEnc_aPPTblBptr	1 word:X	APP **構造体の先頭アドレス**（0x0000～0xFFFF:X）

N **ワードのサイズを持つ** APP **構造体を埋め込む例を表に示します。**

表 2-5　APP 構造体

メモリ・アドレス	メモリ	設定値
jpegEnc_aPPTblBptr:X	1 word:X	N（0～65535, APP **ヘッダ・サイズ**）
jpegEnc_aPPTblBptr:X ＋ 1 ～ jpegEnc_aPPTblBptr:X ＋ N	N word:X	APP **ヘッダ本体**（0xFF, 0xE*～）

備考　APP **ヘッダ・サイズ**（N）**に** 0 **を指定した場合は，**APP **マーカは埋め込まれません。**

APP **ヘッダ以外のデータ・セグメントも埋め込むことが可能です。**

2. 3. 2　jpegEnc_Compress 関数の返り値

(1) jpegEnc_returnValue

ライブラリの圧縮ルーチンは，エンコード処理の終了，MCU，JPEG バッファによる中断，各種エラーをステータスとして返します。

変数名	メモリ	意　味	返り値
jpegEnc_returnValue	1 word:X	正常終了	0x0 (JPEGENC_OK)
		MCU 中断	0x1 (JPEGENC_CONT1)
		JPEG バッファ中断	0x2 (JPEGENC_CONT2)
		エラー終了注	0xFFFF (JPEGENC_ERR)

注　エラー終了の場合は，jpegEnc_errorState にエラー・コードが格納されます。

注意　処理を中断したあと再開する場合，μSAP77016-B22 はこの jpegEnc_returnValue の値を見て，一番最初の起動か，それとも再開かを判別します。このため，起動前の設定以外では初期化しないでください。

(2) jpegEnc_fileSize

処理が正常終了した際に，でき上がった JPEG ファイルのバイト数を出力します。

変数名	メモリ	設定値
jpegEnc_fileSize	2 word:X	ファイル・サイズ（バイト）

(3) jpegEnc_errorState

μSAP77016-B22 の圧縮ルーチンは，なんらかの原因で処理が正常終了できなかった場合に，jpegEnc_errorState にエラー・ステータスの値を代入し，処理を中止します。その際，返り値として jpegEnc_returnValue に JPEGENC_ERR を返します。

変数名	メモリ	意　味	返り値
jpegEnc_errorState	1 word:X	サンプル比の値が不正	0x1 (JPEGENC_ERR_SAMP)
		イメージ・サイズがサンプル比に対応していない	0x2 (JPEGENC_ERR_IMAGESIZE)
		JPEG バッファの値が不正	0x3 (JPEGENC_ERR_JPEGBUFF)
		コンポーネント番号の値が不正	0x4 (JPEGENC_ERR_COMPNUM)
		DHT テーブルの値が不正	0x5 (JPEGENC_ERR_DHT)

2. 4　jpegEnc_GetVersion 関数

jpegEnc_GetVersion **関数は，エンコーダのバージョン情報を取得します。**

【 分　類 】圧縮処理系

【 関 数 名 】jpegEnc_GetVersion

【機能概要】バージョン情報取得

【 形　式 】call jpegEnc_GetVersion

【 引 き 数 】なし

【 返 り 値 】

変数名	メモリ	説　　明
r0h	-	**メジャー・バージョン情報**
r0l	-	**マイナ・バージョン情報**

エンコーダのバージョン番号を 32 **ビットの値でレジスタに返します。**

例　r0 = 0x00’0x0001’0x0100 **の場合，バージョン：**V1.01

【使用レジスタ】r0

【ハードウエア・リソースメント】

最大スタック・レベル	1
最大ループ・スタック・レベル	0
最大リピート回数	0
最大サイクル数	6

2. 5　jpegDec_Decompress 関数

jpegDec_Decompress 関数は，設定されたパラメータに従って，JPEG デコード（1 MCU）を行います。

【 分　類 】伸長処理系

【 関 数 名 】jpegDec_Decompress

【機能概要】JPEG 伸長処理

【 形　式 】call jpegDec_Decompress

【引 き 数 】

変数名	メモリ	説　明
jpegDec_returnValue	1 word:X	ステータス（最初のデコード時は 0 にしてください。）
jpegDec_workAreaBptr	1 word:X	ワーク・エリアの先頭アドレス
jpegDec_jpegBuffBptr	1 word:X	JPEG データ・バッファの先頭アドレス
jpegDec_jpegBuffEptr	1 word:X	JPEG データ・バッファの終端アドレス

【 返 り 値 】

変数名	メモリ	説　明
jpegDec_returnValue	1 word:X	ステータス
jpegDec_imageWidth	1 word:X	水平方向の画像サイズ（ピクセル数）
jpegDec_imageHeight	1 word:X	垂直方向の画像サイズ（ピクセル数）
jpegDec_sampleRatio	1 word:X	サンプル比
jpegDec_errorState	1 word:X	エラー・コード

エラーが発生した場合は，jpegDec_errorState にエラー・コードが格納されます。

【使用レジスタ】すべてのレジスタ

【ハードウエア・リソースメント】

最大スタック・レベル　9

最大ループ・スタック・レベル　3

最大リピート回数　62

サイクル数（任意の画像ファイルの 1MCU の伸張に必要なサイクル数）

- jpegDec_returnValue に 0 をセットした場合　154302
- jpegDec_returnValue に 0 以外をセットした場合　52526

備考　この値は，当社評価時の実測値です。入力される画像によって変動します。この値は，最大サイクル数を保証するものではありません。

2. 5. 1　jpegDec_Decompress 関数の引き数

(1) jpegDec_returnValue

返り値の初期化を行います。

圧縮処理を起動する前に，一度だけ 0 を設定し，初期化してください。

変数名	メモリ	設定値
jpegDec_returnValue	1 word:X	0

注意　処理を中断したあと再開する場合，μSAP77016-B22 はこの jpegDec_returnValue の値を見て，最初の起動か，それとも再開かを判別します。このため，起動前の設定以外では初期化しないでください。

また，最初の jpegDec_Decompress 関数の呼び出し時に JPEG データを JPEG バッファに配置してください。

(2) jpegDec_workAreaBptr

ワーク・エリアの先頭アドレスを設定します。

変数名	メモリ	設定値
jpegDec_workAreaBptr	1 word:X	ワーク・エリアの先頭アドレス

注意　ワーク・エリアのサイズは 768 ワード固定です。ワーク・エリアは X メモリ上に存在する必要があります。

(3) jpegDec_jpegBuffBptr，jpegDec_jpegBuffEptr

JPEG バッファの先頭アドレスと終端アドレスを設定します。

変数名	メモリ	設定値
jpegDec_jpegBuffBptr	1 word:X	JPEG バッファの先頭アドレス
jpegDec_jpegBuffEptr	1 word:X	JPEG バッファの先頭アドレス＋JPEG バッファ・サイズ－1

注意　JPEG バッファサイズは１以上，65535 以下に設定してください。

JPEG バッファは X メモリ上に存在する必要があります。

2. 5. 2　jpegDec_Decompress 関数の返り値

(1) jpegDec_returnValue

μSAP77016-B22 の伸長ルーチンは，デコード処理の終了，MCU，JPEG バッファによる中断，各種エラーをステータスとして返します。

変数名	メモリ	意　味	返り値
jpegDec_returnValue	1 word:X	正常終了	0x0 (JPEGDEC_OK)
		JPEG バッファ中断	0x1 (JPEGDEC_CONT1)
		MCU 中断	0x2 (JPEGDEC_CONT2)
		エラー終了注	0xFFFF (JPEGDEC_ERR)

注　エラー終了の場合は，jpegDec_errorState にエラー・コードが格納されます。

(2) jpegDec_imageWidth，jpegDec_imageHeight

伸長する画像の縦横のピクセル数を返します。

変数名	メモリ	設定値
jpegDec_imageWidth	1 word:X	1 ～ 65535
jpegDec_imageHeight	1 word:X	1 ～ 65535

(3) jpegDec_sampleRatio

サンプル比を返します。

μSAP77016-B22 がサポートしているサンプル比は次の 4 種類です。

変数名	メモリ	サンプル比	設定値（シンボル名）
jpegDec_sampleRatio	1 word:X	4:1:1 (H:V = 2:2)	0x22 (JPEGDEC_SAMPLING22)
		4:1:1 (H:V = 4:1)	0x41 (JPEGDEC_SAMPLING41)
		4:2:2 (H:V = 2:1)	0x21 (JPEGDEC_SAMPLING21)
		4:4:4 (H:V = 1:1)	0x11 (JPEGDEC_SAMPLING11)

(4) jpegDec_errorState

μSAP77016-B22 の伸長ルーチンは，なんらかの原因で処理が正常終了できなかった場合に，jpegDec_errorState にエラー・ステータスの値を代入し，処理を中止します。表 2-6にエラー・ステータスを示します。

表 2-6　jpegDec_errorState

返り値（シンボル名）	意　　味
0x01 (JPEGDEC_ERR_JPEGBUFF)	JPEG バッファの値が不正です。
0x02 (JPEGDEC_ERR_SAMP)	対応していないサンプル比です。
0x03 (JPEGDEC_ERR_SOI_DEC)	SOI セグメントが見つかりませんでした。
0x04 (JPEGDEC_ERR_DQT_MANY)	DQT セグメントが多すぎます。
0x05 (JPEGDEC_ERR_DQT_DEC)	DQT セグメントの内容が不正または対応していません。
0x06 (JPEGDEC_ERR_SOF_DEC)	SOF セグメントの内容が不正または対応していません。
0x07 (JPEGDEC_ERR_DHT_MANY)	DHT セグメントが多すぎます。
0x08 (JPEGDEC_ERR_DHT_DEC)	DHT セグメントの内容が不正または対応していません。
0x09 (JPEGDEC_ERR_SOS_DEC)	SOS セグメントの内容が不正または対応していません。
0x0A (JPEGDEC_ERR_DRI_DEC)	DRI ヘッダの内容が不正または対応していません。
0x0B (JPEGDEC_ERR_RST_DEC)	RST マーカの内容が不正または対応していません。
0x0C (JPEGDEC_ERR_EOI_DEC)	EOI セグメントが見つかりませんでした。
0x0D (JPEGDEC_ERR_UNKNOWMARK)	未知のセグメントが見つかりました。
0x0E (JPEGDEC_ERR_FILE_COMPNUM)	コンポーネント・ナンバが対応していません。
0x0F (JPEGDEC_ERR_SAMPL)	サンプル比の値が不正または対応していません。
0x10 (JPEGDEC_ERR_FILE_DQTNUM)	DQT セグメントの内容が不正または対応していません。
0x11 (JPEGDEC_ERR_FILE_DHTNUM)	DHT セグメントの内容が不正または対応していません。
0x12 (JPEGDEC_ERR_DATA)	データ部にエラーがあります。
0x13 (JPEGDEC_ERR_SEGMENT)	セグメントが不正です。

2. 6　jpegDec_GetVersion 関数

jpegDec_GetVersion 関数は，デコーダのバージョン情報を取得します。

【 分　類 】伸長処理系

【関 数 名】jpegDec_GetVersion

【機能概要】バージョン情報取得

【 形　式 】call jpegDec_GetVersion

【引 き 数】なし

【返 り 値】

変数名	メモリ	説　　明
r0h	-	メジャー・バージョン情報
r0l	-	マイナ・バージョン情報

デコーダのバージョン番号を 32 ビットの値でレジスタに返します。

例　r0 = 0x00’0x0001’0x0100 の場合，バージョン：V1.01

【使用レジスタ】r0

【ハードウエア・リソースメント】

最大スタック・レベル	1
最大ループ・スタック・レベル	0
最大リピート回数	0
最大サイクル数	6

2. 7　VRAM アクセス関数のカスタマイズ

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。ユーザは，画像入出力の部分をシステムに合わせて作成する必要があります。

2. 7. 1　必要な関数

圧縮処理で必要な関数は，VRAM から 1 MCU 分のデータをワーク・エリアにロードする関数です。サンプル比によって異なる関数を作る必要があります。

伸長処理で必要な関数は，ワーク・エリアから 1 MCU 分のデータを VRAM にストアする関数です。サンプル比によって異なる関数を作る必要があります。

2. 7. 2　画像の取り扱い

μSAP77016-B22 では，画像データを MCU 単位で処理します。圧縮では MCU 単位で，画像入力（YCbCr）→DCT 変換→量子化→ハフマン符号化を行い，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（YCbCr）を行います。

圧縮時は，MCU のデータが Y/Cb/Cr の形式でワーク・エリアに格納されるように VRAM アクセス関数を作成してください。伸長時は，逆に MCU に相当するデータが Y/Cb/Cr の形式で格納されているため，そのデータを VRAM に転送するように VRAM アクセス関数を作成してください。

表 2-7　MCU の単位

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

μSAP77016-B22 の圧縮時の DCT 変換の入力，伸長時の逆 DCT 変換の出力は RGB 形式ではなく，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 から YCbCr へ，または YCbCr から RGB への変換が必要になります。

2. 8　サンプル比とブロック

圧縮時は，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 2-8　MCU とブロック

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

図 2-8　1 MCU 分の画像データ　4:1:1 (H:V = 2:2) の場合

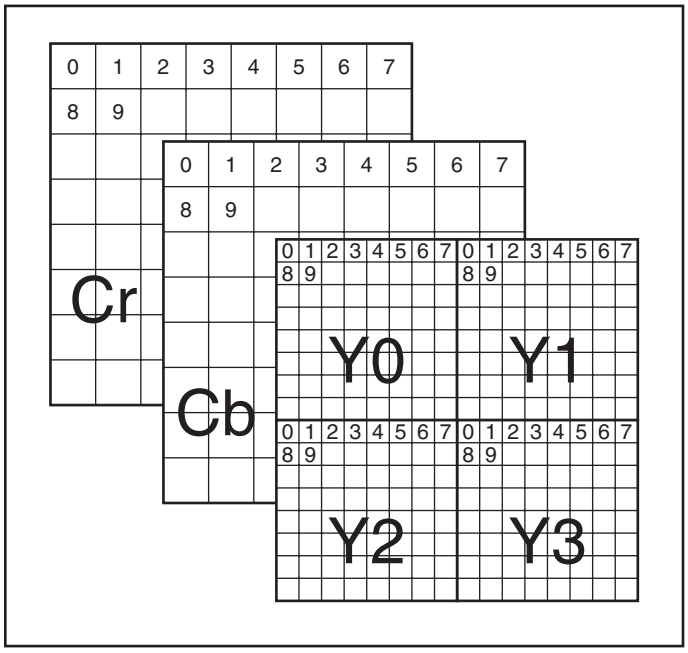


図 2-9　1 MCU 分の画像データ　4:1:1 (H:V = 4:1) の場合

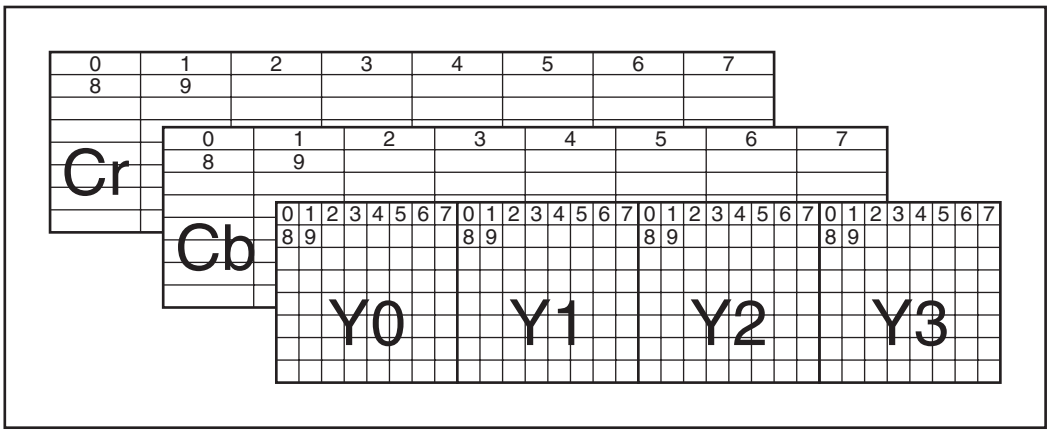


図 2-10　1 MCU 分の画像データ　4:2:2 (H:V = 2:1) の場合

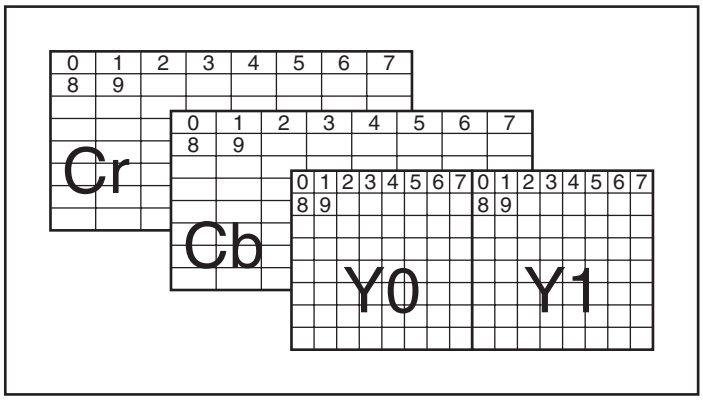
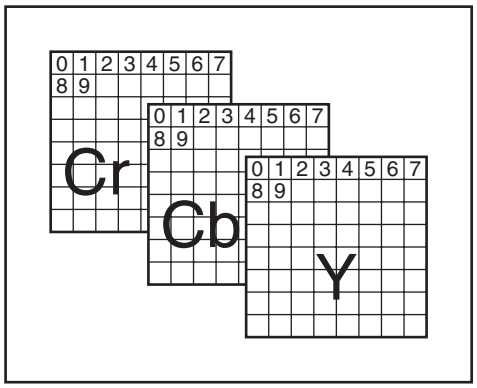


図 2-11　1 MCU 分の画像データ　4:4:4 (H:V = 1:1) の場合



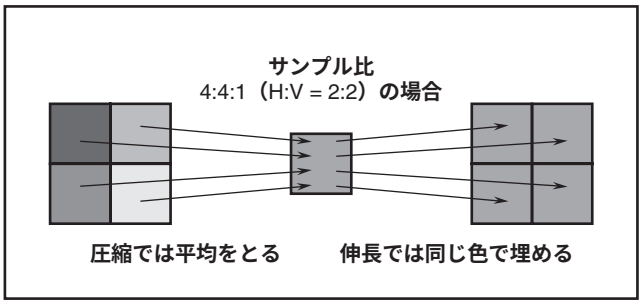
圧縮時の色差成分（Cb／Cr）は，サンプル比 4:4:4 (H:V = 1:1) の場合を除いて，隣り合ういくつかのピクセルの色差成分の平均をとります。この平均をとる処理をサンプリングといいます。

表 2-9　色差成分のサンプリング

サンプル比	サンプリング
4:1:1（H:V = 2:2）	横 2×縦 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
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

伸長処理では，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。

図 2-12　サンプリング



2. 9　入出力バッファ

ユーザは，入出力バッファ（JPEG バッファ，ワーク・エリア（MCU バッファ））の領域を X メモリに確保する必要があります。

この領域は，MCU 画像データ（YUV）およびビット・ストリーム・データの入出力用の領域です。1 フレームのエンコード／デコード処理のあと，ユーザが自由に使用することができます。

エンコード／デコード処理中に画像データおよびビット・ストリーム・データの領域を操作した場合，正常な動作を保証できません。

エンコード／デコード処理時にユーザが入出力バッファを使用できるタイミングは，次のとおりです。

・エンコーダの入出力バッファを使用できるタイミング

① jpegEnc_Compress 関数からの戻り値が JPEGENC_CONT1 の場合，ワーク領域（MCU バッファ）を使用することができます。

（次の MCU データを入力し，jpeg_Compress 関数を呼び出すまでの間）

② jpegEnc_Compress 関数からの戻り値が JPEGENC_CONT2 の場合，JPEG バッファ領域を使用することができます。

（JPEGStream を出力後，jpeg_Compress 関数を呼び出すまでの間）

・デコーダの入出力バッファを使用できるタイミング

① jpegDec_Decompress 関数からの戻り値が JPEGDEC_CONT1 の場合，JPEG バッファ領域を使用することができます。

（次の JPEGStream を入力し，jpeg_Decompress 関数を呼び出すまでの間）

② jpegDec_Decompress 関数からの戻り値が JPEGDEC_CONT2 の場合，ワーク領域（MCU バッファ）を使用することができます。

（MCU データを出力後，jpeg_Decompress 関数を呼び出すまでの間）

注意 1. 入出力バッファ（JPEG バッファ，ワーク・エリア（MCU バッファ））は，両方とも X メモリ領域に配置する必要があります。

2. MCU 画像データの配置領域は，サンプル・レートによって異なります。

例　入出力バッファ領域の指定例

```
WORK_SIZE       EQU     768         ; 768固定
JPEG_BUFSIZE    EQU     1000        ; User define

__JPEG_DATA xramseg
JPEG_Buffer:
       ds      JPEG_BUFSIZE
WORK_Area:
       ds      WORK_SIZE
```

図 2-13，図 2-14に入出力バッファ（JPEG バッファ，ワーク・エリア（MCU バッファ））のデータ配置イメージを示します。

図 2-13　JPEG バッファ

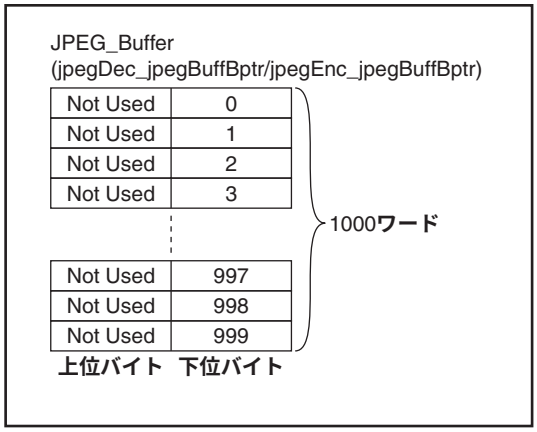
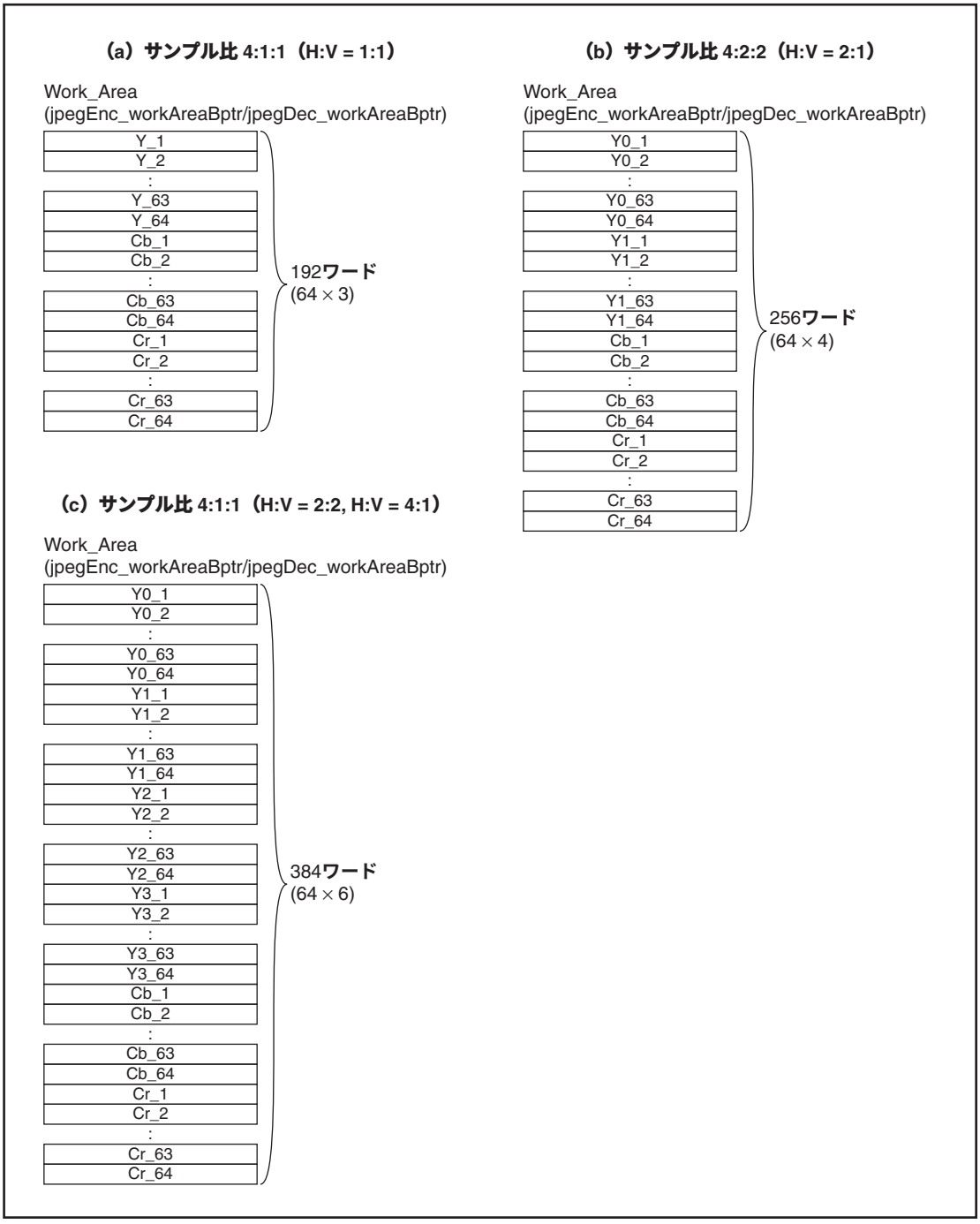


図 2-14　ワーク・エリア（MCU バッファ）



第3章 インストレーショ ン

3.1　インストレーション手順

μSAP77016-B22（JPEG ビデオ・コーデック・ミドルウエア）の提供媒体は，CD-ROM です。ホスト・マシンへのインストール手順を次に示します。

① 提供媒体を CD-ROM ドライブにセットします。任意のディレクトリ（例：C:¥DSPTools）の下にファイルをコピーします。ここでは，Q ドライブから C ドライブへコピーした場合を示します。

```
Q:¥>xcopy /s *.* C:¥DSPTools<CR>
```

② ファイルがコピーされたことを確認します。各ディレクトリについては，1. 5. 6　ディレクトリ構成を参照してください。

```
C:¥>dir C:¥DSPTools<CR>
```

3. 2　シンボル命名規約

μSAP77016-B22 で使用するシンボルの命名規約を表 3-1 に示します。他のアプリケーションを組み合わせて使用するときは，重複しないようにしてください。

表 3-1　シンボル命名規約

分　類	命名規則
関数名	jpegEnc_Xxxx, jpegDec_Xxxx
変数領域名	jpegEnc_xxxx, jpegDec_xxxx
その他	__JPEG_XXXX（先頭のアンダスコアは 2 つ）

備考　XXXX，Xxxx，xxxx は任意の英字とします。X は大文字，x は小文字です。

第4章 サンプル使用例

4.1 サンプル・プログラムを使用したビルド／シミュレーション環境

μSAP77016-B22 の圧縮／伸長処理のビルド／シミュレーション環境を次に示します。

【ソフトウエア環境】

- Atair Developer Studio（ワークベンチ（アセンブラ／リンカ）（Atair 社製）
- μPD7721x ハイスピード・シミュレータ（Atair 社製）
- サンプル・プログラム／タイミング・ファイル（μSAP77016-B22 に含まれています。1. 5. 6 ディレクトリ構成を参照してください。）

4.2 操作方法

サンプル・プログラムを使用した圧縮／伸長処理のビルド／シミュレーション手順を次に示します。

4.2.1 エンコーダの場合

(1) RAM 品種用 JPEG エンコード・ライブラリ（jpegenc_ram.lib）およびエンコード・ライブラリ・ヘッダ・ファイル（jpeg_enc.h）を library ディレクトリから sample¥encoder¥lib ディレクトリへコピーします。

(2) Atair Developer Studio を起動し，sample ディレクトリの uPD77210.model（モデル・ファイル）を読み込みます。

(3) sample ディレクトリの dsp_jpeg_sample.AtairPWS を読み込み，encoder プロジェクトをアクティブにします。

(4) sample¥encoder¥sources ディレクトリの encoder_sample.asm を編集（エンコード・パラメータを設定）したあと，ビルドを行い，encoder.lnk を作成します。

(5) インプット・データ・ファイル（BMP 形式）を用意します。
例 rgb.bmp（320 x 240 ピクセル）

(6) 変換ツール（sample¥tool ディレクトリの bmp2dat11.exe または bmp2dat22.exe）を使用し，用意したインプット・データ・ファイル（rgb.bmp）を dat 形式（rgb.dat）に変換します。

(7) sample¥encoder ディレクトリの hostwr8.tmg ファイルの入力ファイル名を作成したインプット・データ・ファイルに合わせて変更します。

(8)　sample¥encoder ディレクトリの hostrd8.tmg ファイルの出力ファイル名を設定します。

(9)　μPD7721x ハイスピード・シミュレータを起動し，sample ディレクトリの uPD77210.model を読み込みます。

(10)　μPD7721x ハイスピード・シミュレータに，(4)で作成した encoder.lnk と(7)，(8)で編集した hostwr8.tmg, hostrd8.tmg を読み込みます。

(11)　JPEGCompress_OK:, JPEGCompress_NG: にブレーク・ポイントを設定します。

(12)　μPD7721x ハイスピード・シミュレータを実行します。

(13)　しばらくすると，(11) で設定したアドレスで停止します。
JPEGCompress_OK で停止した場合は，エンコードが正常終了したことを示します。
JPEGCompress_NG で停止した場合は，正常にエンコードが終了しなかったことを示します。

(14)　sample¥encoder ディレクトリに (8) で指定した出力ファイルが作成されます。

(15)　sample¥tool ディレクトリの dat2jpg.exe を使用し，出力ファイルを jpg ファイルに変換します。

(16)　JPEG ビューワで出力ファイルが正常に作成されたことを確認します。
出力ファイルの画像は，bmp2dat11.exe または bmp2dat22.exe での変換時に上下反転されます。画像が上下反転されていても，ミドルウエアの動作は正常です。

4. 2. 2　デコーダの場合

(1)　RAM 品種用 JPEG デコード・ライブラリ（jpegdec_ram.lib）およびデコード・ライブラリ・ヘッダ・ファイル（jpeg_dec.h）を library ディレクトリから sample/decoder/lib ディレクトリへコピーします。

(2)　Atair Developer Studio を起動し，sample ディレクトリの uPD77210.model（モデル・ファイル）を読み込みます。

(3)　sample ディレクトリの dsp_jpeg_sample.AtairPWS を読み込み，decoder プロジェクトをアクティブにします。

(4)　sample¥decoder¥sources ディレクトリの decoder_sample.asm のビルドを行い，decoder.lnk を作成します。

(5)　インプット・データ・ファイル（JPEG 形式）を用意します。
例　rgb.jpg （320 × 240 ピクセル，H:V = 2:2）

(6)　sample¥tool ディレクトリの jpg2dat.exe を使用し，rgb.jpg を dat 形式（rgb.dat）に変換します。

(7)　sample¥decoder ディレクトリの hostwr8.tmg ファイルを入力ファイル名に合わせて変更します。

(8)　sample¥decoder ディレクトリの hostrd8.tmg ファイルに出力ファイル名を設定します。

(9)　μPD7721x ハイスピード・シミュレータを起動し，sample ディレクトリの uPD77210.model を読み込みます。

(10)　μPD7721x ハイスピード・シミュレータに，(4)で作成した decoder.lnk と (7)，(8) の hostwr8.tmg, hostrd8.tmg を読み込みます。

(11)　JPEGDecompress_OK:,JPEGDecompress_NG:にブレーク・ポイントを設定します。

(12)　μPD7721x ハイスピード・シミュレータを実行します。

(13)　しばらくすると，(11) で設定したアドレスで停止します。
JPEGDecompress_OK で停止した場合はデコードが正常終了したことを示します。
JPEGDecompress_NG で停止した場合は正常にデコードが終了しなかったことを示します。

(14)　μPD7721x ハイスピード・シミュレータをリセットします。

(15)　sample¥decoder ディレクトリに (8) で指定した出力ファイルが作成されます。

(16)　sample¥tool ディレクトリの dat2bmp11.exe または dat2bmp22.exe を使用し，出力ファイルを bmp ファイルに変換します。

(17)　BMP ビューワで出力ファイルが正常に作成されたことを確認します。
出力ファイルの画像は，dat2bmp11.exe または dat2bmp22.exe での変換時に上下反転されます。画像が上下反転されていても，ミドルウエアの動作は正常です。

付　録　サンプル・プログラム・ソース

付録.1　デコード用サンプル・プログラム

・デコード用サンプル・プログラム・ソース（decoder_sample.asm）


```
;---------------------------------------------------------------------
;-     File information                                              -
;---------------------------------------------------------------------
;-     Name       : decoder_sample.asm                               -
;-     Type       : ASM Programming Language  Application            -
;-     Version    : V1.0                                             -
;-     Date       : 2002 NOV 15                                      -
;-     CPU        : uPD7701x Family                                  -
;-     Assembler  : Atair Developer Studio Version 1.01 (Build83)    -
;-     About      : NEC uPD7701x Family JPEG Middleware Library      -
;-                  Application Sample                               -
;---------------------------------------------------------------------
;-     Copyright (C) NEC Coporation 2002                             -
;-     NEC CONFIDENTIAL AND PROPRIETARY                              -
;-     All rights reserved by NEC Coporation                         -
;-     Use of copyright notice does not evidence publicaton          -
;---------------------------------------------------------------------
;---------------------------------------------------------------------
;-    Include                                                        -
;---------------------------------------------------------------------
#include "jpeg_dec.h"
#include "macro.h"
#include "upd7721x.h"

;---------------------------------------------------------------------
;-     function                                                      -
;---------------------------------------------------------------------
extrn YUVtoRGB;YUV -> RGB sample (yuvtorgb.asm)

public JPEGDecompress_OK
public JPEGDecompress_NG

;---------------------------------------------------------------------
;-     Parameters                                                    -
;---------------------------------------------------------------------
JPEG_BUFSIZE    EQU    1000          ; ストリーム・バッファ・サイズ (User define)

__JPEG_BUFFER  xramseg
JPEG_Buffer:
       ds    JPEG_BUFSIZE
jpeg_Work_Area:
       ds    768                    ;768words固定
```

(2/6)

```
;---------------------------------------------------------------------
;-     RGB buffer                    ; User define                   -
;---------------------------------------------------------------------
RGB_BUFSIZE    EQU    768            ; User define 192x(192(4:4:4),384(4:2:2),768(4:1:1))
RGB_Buffer:
       ds    RGB_BUFSIZE
public RGB_Buffer

;---------------------------------------------------------------------
;-     local                                                         -
;---------------------------------------------------------------------
;ファイル終端検出用(EOI)
JPEG_BufferPrev8bit:
       ds     1
SOF_headercounter:
       ds     1
;---------------------------------------------------------------------
;-    interrupter vector segment                                     -
;---------------------------------------------------------------------
__INTVECTOR IMSEG at 0x200
       %DefVectJmp(StartUp)          ;ivReset
       %DefVectNop                   ;ivPriv
       %DefVectNop                   ;ivAddr
       %DefVectNop                   ;ivSpare0
       %DefVectNop                   ;ivINT1
       %DefVectNop                   ;ivINT2
       %DefVectNop                   ;ivINT3
       %DefVectNop                   ;ivINT4
       %DefVectNop                   ;ivSI1
       %DefVectNop                   ;ivSO1
       %DefVectNop                   ;ivSI2
       %DefVectNop                   ;ivSO2
       %DefVectJmp(InPutInt)         ;ivHI（ストリーム入力用）
       %DefVectJmp(OutPutInt)        ;ivHO（画像出力用）
       %DefVectNop                   ;ivSpare1
       %DefVectNop                   ;ivSpare2

;---------------------------------------------------------------------
;      program segment                                                -
;---------------------------------------------------------------------
__JPEG_SAMPLE IMSEG at 0x240
;-------------------------------------------------
;-     main                                      -
;-------------------------------------------------
StartUp:
       FINT                          ;
       %DisableInterrupt()           ;
       %InitRegister()               ;

              r2l = 0x10             ;
              *ICR8:X =r2l           ;
              *ICR9:X =r2l           ;
```

(3/6)

```
              clr(r0)                                ;EOI検出用
              *JPEG_BufferPrev8bit:X = r0l           ;
              r0l = 0xFFFF                           ;
              *SOF_headercounter:X = r0l             ;

       ;--------------------------------
       ;-  set Parameters               -
       ;--------------------------------
       clr(r0)                                 ;
       *jpegDec_returnValue:x = r0l            ;
       r0l = jpeg_Work_Area                    ;
       *jpegDec_workAreaBptr:x = r0l           ;
       r0l = JPEG_Buffer                       ;
       *jpegDec_jpegBuffBptr:x = r0l           ;
       r0  = r0 + (JPEG_BUFSIZE-1)             ;
       *jpegDec_jpegBuffEptr:x = r0l           ;

       jmp Decompress_CONT1                    ;ストリーム入力
       ;--------------------------------
       ;-  Main loop                    -
       ;--------------------------------

Main_loop:
       call jpegDec_Decompress                 ;1MCUデコード

       clr(r0)                                 ;
       r0l = *jpegDec_returnValue:x            ;
       r1  = r0 ^ JPEGDEC_CONT1                ;JPEGデータ入力
       if(r1==0) jmp Decompress_CONT1          ;
       r1  = r0 ^ JPEGDEC_CONT2                ;1MCU出力
       if(r1==0) jmp Decompress_CONT2          ;
       r1  = r0 ^ JPEGDEC_OK                   ;正常終了
       if(r1==0) jmp JPEGDecompress_OK         ;
       jmp JPEGDecompress_NG                   ;異常終了

Decompress_CONT1:
              dp0 = JPEG_Buffer                ;
              clr(r2)                          ;
              r2l = JPEG_BUFSIZE               ;
              r5l = 1                          ; transfer flag
              r4l = *JPEG_BufferPrev8bit:X     ;

       %DisableInterrupt()                     ;
       %SetSR_HI_Enable()                      ; HI Enable
       %EnableInterrupt()                      ;

              r0l = *HST:X                     ;HREM UNMASK
              r0 = r0 & 0xFEFF                 ;
       *HST:X = r0l                            ;ストリーム入力再開

              if(r5!=0) jmp $                  ;ストリーム入力終了待ち
       jmp Main_loop                           ;
```

(4/6)

```
Decompress_CONT2:
              r7l = *jpegDec_sampleRatio:X    ;サンプル比
              call YUVtoRGB                   ;YUV -> RGB 変換(1MCU)

           dp1 =  RGB_Buffer                  ;
       r2l = *jpegDec_sampleRatio:X           ;
              call SetRGBSize                 ;
       r5l = 1                                ; set flag

       r0l = *dp1++                           ;
       r0  = r0 sll 8                         ;
       *HDT:y = r0l                           ;1byte書き込み
       r2  = r2 - 1                           ;

              r0l = *HST:X                    ;HWEM UNMASK
              r0 = r0 & 0xFDFF                ;
       *HST:X = r0l                           ;画像出力再開

       %DisableInterrupt()                    ;
       %SetSR_HO_Enable()                     ; HO Enable
       %EnableInterrupt()                     ;

              if(r5!=0) jmp $                 ;画像出力終了待ち
       jmp Main_loop                          ;

JPEGDecompress_OK:
              jmp $;                            正常終了

JPEGDecompress_NG:
              jmp $;                            異常終了

;1MCU(RGB)サイズ設定関数
;input  r2l = SAMPLINGRATIO(0x11,0x21,0x22,0x41)
;output r2l = RGBSIZE(192,384,768)
SetRGBSize:
              r2 = r2 & 0xFFFF                ;
              r3 = r2 - 0x11                  ;
              if(r3==0) jmp _0x11             ;
              r3 = r2 - 0x21                  ;
              if(r3==0) jmp _0x21             ;
              r3 = r2 - 0x22                  ;
              if(r3==0) jmp _0x22             ;
              r3 = r2 - 0x41                  ;
              if(r3==0) jmp _0x41             ;
              jmp JPEGDecompress_NG           ;
_0x11:
              r2l = 192                       ;R(8x8)+G(8x8)+B(8x8)
              ret                             ;
_0x21:
              r2l = 384                       ;(R(8x8)+G(8x8)+B(8x8))*2
              ret                             ;
_0x22:
_0x41:
              r2l = 768                       ;(R(8x8)+G(8x8)+B(8x8))*4
              ret                             ;
```

(5/6)

```
;-------------------------------------------------------------------------
;-      Int handler                                                      -
;-------------------------------------------------------------------------
;画像出力用割り込みハンドラ
;dp1 = RGBバッファ・アドレス
;r2  = 転送サイズ
;r5  = 転送終了フラグ
OutPutInt:
              r0l = *ICR9:X                   ;
       if(r2>0) jmp _OutContinue              ;

       %DisableInterrupt()                    ;
       %SetSR_HO_Disable()                    ;
       %EnableInterrupt()                     ;

              r0l = *HST:X                    ;HREM MASK
              r0 = r0 | 0x200                 ;
       *HST:X = r0l                           ;画像出力停止

       clr(r5)                                ;
       jmp _OutRet                            ;

_OutContinue:
       r0h=*dp1++                             ; perform output from HDO
       r0  = r0 sll 8                         ;
       *HDT:x=r0h                             ;
       r2  = r2 - 1                           ;
_OutRet:
       reti                                   ;

;ストリーム入力用割り込みハンドラ
;dp0 = ストリーム・バッファ・アドレス
;r2  = 転送サイズ
;r4  = ストリーム・データ待避レジスタ(EOI検出用)
;r5  = 転送終了フラグ
InPutInt:
              r0l = *ICR8:X                   ;
              clr(r0)                         ;
              r0l = *HDT:X                    ;
              r0 = r0 srl 8                   ;
              *dp0++ = r0l                    ;

              r4 = r4 sll 8                   ;16bitsヘッダ検出
              r4 = r4 | r0                    ;
              r4 = r4 & 0xFFFF                ;
```

(6/6)

```
_SOF_Check:
              r0 = r4 - 0xFFD8                ;SOFのヘッダをカウント
              r6l = *SOF_headercounter:X      ;(APPヘッダ内のサムネイル画像のEOIヘッダの誤検出を防ぐため)
              if(r0==0)r6=r6+1                ;
              r6 = r6 & 0xFFFF                ;

_EOI_Check:
              r0 = r4 - 0xFFD9                ;EOI Check
       if(r0==0)r6=r6-1                       ;
              if(r6<0)clr(r2)                 ;
              if(r6<0)jmp _EOI_Found          ;*SOF_headercounter:Xの値がマイナスの場合のみEOI検出を有効にする
              *SOF_headercounter:X = r6l      ;

_Size_Check:
              r2 = r2 - 1                     ;Size = Size - 1
              if(r2>0)jmp _InContinue         ;

_EOI_Found:
              *JPEG_BufferPrev8bit:X = r4l    ;

       %DisableInterrupt()                    ;ストリーム入力終了処理
       %SetSR_HI_Disable()                    ;
       %EnableInterrupt()                     ;

       clr(r5)                                ; set transfer end flag

              r0l = *HST:X                    ;HWEM MASK
              r0 = r0 | 0x100                 ;
       *HST:X = r0l                           ;ストリーム入力停止

_InContinue:
       reti                                   ;

end
```

付録.2　エンコード用サンプル・プログラム

・エンコード用サンプル・プログラム・ソース（encoder_sample.asm）

（1/7）

```
;---------------------------------------------------------------------
;-     File information                                              -
;---------------------------------------------------------------------
;-     Name       : encoder_sample.asm                               -
;-     Type       : ASM Programming Language  Application            -
;-     Version    : V1.0                                             -
;-     Date       : 2002 NOV 15                                      -
;-     CPU        : uPD7701x Family                                  -
;-     Assembler  : Atair Developer Studio Version 1.01 (Build83)    -
;-     About      : NEC uPD7701x Family JPEG Middleware Library      -
;-                  Application Sample                               -
;---------------------------------------------------------------------
;-     Copyright (C) NEC Coporation 2002                             -
;-     NEC CONFIDENTIAL AND PROPRIETARY                              -
;-     All rights reserved by NEC Coporation                         -
;-     Use of copyright notice does not evidence publicaton          -
;---------------------------------------------------------------------
;---------------------------------------------------------------------
;-    Include                                                        -
;---------------------------------------------------------------------
#include "jpeg_enc.h"
#include "macro.h"
#include "upd7721x.h"

;---------------------------------------------------------------------
;-     function                                                      -
;---------------------------------------------------------------------
extrn RGBtoYUV ;RGB -> YUV sample (rgbtoyuv.asm)

public JPEGCompress_OK
public JPEGCompress_NG

;---------------------------------------------------------------------
;-     Parameters                                                    -
;---------------------------------------------------------------------
JPEG_BUFSIZE     EQU    1000      ; ストリーム・バッファ・サイズ (User define)
IMAGE_WIDTH      EQU    160       ; 水平画素サイズ               (User define)
IMAGE_HEIGHT     EQU    120       ; 垂直画素サイズ               (User define)
RESTARTINTERVAL  EQU    0         ; リスタート・マーカ間隔       (User define)
QUALITY          EQU    75        ; 画質                         (User define) (0-100)
SAMPLINGRATIO    EQU    0x21      ; コンポーネント               (User define) (0x22,0x11,0x21,0x41)
COMPNUM1         EQU    1         ; コンポーネント番号1          (User define)
COMPNUM2         EQU    2         ; コンポーネント番号2          (User define)
COMPNUM3         EQU    3         ; コンポーネント番号3          (User define)

__JPEG_APP_INFO xramseg
APP_Info:
;       dw 0x12                              ;header Size
;       dw 0xff,0xe0                         ;APP0(JFIF)
;       dw 0x00,0x10,0x4a,0x46,0x49,0x46,0x00,0x01  ;
;       dw 0x01,0x01,0x00,0x61,0x00,0x61,0x00,0x00  ;
```

(2/7)

```
       dw 0                                    ;APP none(header Size = 0)

__JPEG_BUFFER xramseg
JPEG_Buffer:
       ds    JPEG_BUFSIZE
jpeg_Work_Area:
       ds     768                              ;768words固定

;---------------------------------------------------------------------
;-     RGB buffer                     ; User define                   -
;---------------------------------------------------------------------
RGB_BUFSIZE     EQU     768            ; User define 192x(192(4:4:4),384(4:2:2),768(4:1:1))
RGB_Buffer:
       ds     RGB_BUFSIZE             ;

public RGB_Buffer

jpeg_StreamSizeH:      ds    1;出力済みストリーム・サイズ
jpeg_StreamSizeL:      ds    1;


;---------------------------------------------------------------------
;-     interrupter vector segment                                    -
;---------------------------------------------------------------------
IntVect  IMSEG at 0x200
       %DefVectJmp(StartUp)           ;ivReset
       %DefVectNop                    ;ivPriv
       %DefVectNop                    ;ivAddr
       %DefVectNop                    ;ivSpare0
       %DefVectNop                    ;ivINT1
       %DefVectNop                    ;ivINT2
       %DefVectNop                    ;ivINT3
       %DefVectNop                    ;ivINT4
       %DefVectNop                    ;ivSI1
       %DefVectNop                    ;ivSO1
       %DefVectNop                    ;ivSI2
       %DefVectNop                    ;ivSO2
       %DefVectJmp(InPutInt)          ;ivHI（画像入力用）
       %DefVectJmp(OutPutInt)         ;ivHO（ストリーム出力用）
       %DefVectNop                    ;ivSpare1
       %DefVectNop                    ;ivSpare2

;---------------------------------------------------------------------
;      program segment                                               -
;---------------------------------------------------------------------
__JPEG_SAMPLE imseg at 0x240
;---------------------------------------------------------------------
;-     main                                                          -
;---------------------------------------------------------------------
StartUp:
       FINT                           ;
       %DisableInterrupt()            ;
       %InitRegister()                ;
```

(3/7)

```
r0l = 0x10                      ;
*ICR8:X =r0l                    ;
*ICR9:X =r0l                    ;

clr(r0)                         ;
*jpeg_StreamSizeH:X = r0l       ;
*jpeg_StreamSizeL:X = r0l       ;

;--------------------------------
;-  set Parameters               -
;--------------------------------
clr(r0)                                 ;
*jpegEnc_returnValue:x = r0l            ;
r0l = RESTARTINTERVAL                   ;
*jpegEnc_restartInterval:x = r0l        ;
r0l = IMAGE_WIDTH                       ;
*jpegEnc_imageWidth:x = r0l             ;
r0l = IMAGE_HEIGHT                      ;
*jpegEnc_imageHeight:x = r0l            ;
r0l = SAMPLINGRATIO                     ;
*jpegEnc_sampleRatio:x = r0l            ;
r0l = QUALITY                           ;
*jpegEnc_quality:x = r0l                ;
       r0l = COMPNUM1                   ;
       *jpegEnc_compNumC1:X = r0l       ;
       r0l = COMPNUM2                   ;
       *jpegEnc_compNumC2:X = r0l       ;
       r0l = COMPNUM3                   ;
       *jpegEnc_compNumC3:X = r0l       ;

r0l = jpeg_Work_Area                    ;
*jpegEnc_workAreaBptr:x = r0l           ;
r0l = JPEG_Buffer                       ;
*jpegEnc_jpegBuffBptr:x = r0l           ;
r0  = r0 + (JPEG_BUFSIZE-1)             ;
*jpegEnc_jpegBuffEptr:x = r0l           ;
r0l = jpegEnc_defaultQTblYBptr          ;
       *jpegEnc_qTblYBptr:x = r0l       ;
r0l = jpegEnc_defaultQTblCBptr          ;
*jpegEnc_qTblCBptr:x = r0l              ;
r0l = jpegEnc_defaultDHTDCYBptr         ;
*jpegEnc_dHTDCYBptr:x = r0l             ;
r0l = jpegEnc_defaultDHTDCCBptr         ;
*jpegEnc_dHTDCCBptr:x = r0l             ;
r0l = jpegEnc_defaultDHTACYBptr         ;
*jpegEnc_dHTACYBptr:x = r0l             ;
r0l = jpegEnc_defaultDHTACCBptr         ;
*jpegEnc_dHTACCBptr:x = r0l             ;

r0l = APP_Info                          ;
*jpegEnc_aPPTblBptr:x = r0l             ;
```

(4/7)

```
       ;--------------------------------
       ;-  Main loop                    -
       ;--------------------------------

Main_loop:
       call jpegEnc_Compress             ;1MCUエンコード

       clr(r0)                           ;
       r0l = *jpegEnc_returnValue:x      ;
       r1  = r0^JPEGENC_CONT1            ;1MCU入力
       if(r1==0) jmp Compress_CONT1      ;
       r1  = r0^JPEGENC_CONT2            ;JPEGデータ出力
       if(r1==0) jmp Compress_CONT2      ;
       r1  = r0^JPEGENC_OK               ;正常終了
       if(r1==0) jmp Compress_OK         ;
       jmp JPEGCompress_NG               ;異常終了

;画像入力(1MCU,RGB)
Compress_CONT1:
       dp0 =  RGB_Buffer                 ;

              r2l = SAMPLINGRATIO        ;
       call SetRGBSize                   ;
       r5l = 1                           ; transfer flag

       %DisableInterrupt()               ;
       %SetSR_HI_Enable()                ;
       %EnableInterrupt()                ;

              r0l = *HST:X               ;HREM UNMASK
              r0 = r0 & 0xFEFF           ;
       *HST:X = r0l                      ;画像入力再開

       if(r5!=0) jmp $                   ;1MCU分のRGBデータ入力終了待ち
       r7l = SAMPLINGRATIO               ;
       call RGBtoYUV                     ; RGB -> YUV 変換(1MCU)

              jmp Main_loop;

;JPEGストリーム出力
Compress_CONT2:
       dp1 =  JPEG_Buffer                ; 転送元アドレス(dp1)
       clr(r3)                           ;
       r3l = JPEG_BUFSIZE                ; 転送サイズ(r3l)
       r5l = 1                           ; set flag

       r0l = *dp1++                      ; 1byte書き込み
       r0  = r0 sll 8                    ;
       *HDT:y = r0l                      ;
       r3  = r3 - 1                      ;

              r0l = *HST:X               ;HWEM UNMASK
              r0 = r0 & 0xFDFF           ;
       *HST:X = r0l                      ;ストリーム出力再開
```

(5/7)

```
        %DisableInterrupt()                ;
        %SetSR_HO_Enable()                 ;
        %EnableInterrupt()                 ;

        if(r5!=0) jmp $                    ;JPEGストリーム出力終了待ち

                r0h = *jpeg_StreamSizeH:X  ;
                r0l = *jpeg_StreamSizeL:X  ;
                r0 = r0 + JPEG_BUFSIZE     ; 転送済みサイズ=転送済みサイズ＋転送サイズ
                *jpeg_StreamSizeH:X = r0h  ;
                *jpeg_StreamSizeL:X = r0l  ;

        jmp Main_loop;

;JPEGエンコード終了
Compress_OK:
        clr(r0)                            ;
                clr(r3)                    ;
        r0h = *jpegEnc_fileSize+0:X        ;JPEGファイル・サイズ
                r0l = *jpegEnc_fileSize+1:X ;
                r3h = *jpeg_StreamSizeH:X  ;転送済みJPEGファイル・サイズ
                r3l = *jpeg_StreamSizeL:X  ;
        r3  = r0 - r3                      ;
        if(r3==0) jmp  JPEGCompress_OK     ;

        dp1 =  JPEG_Buffer                 ;
                r5l = 1                        ; set flag

        r0l = *dp1++                       ;
        r0  = r0 sll 8                     ;
        *HDT:y = r0l                       ;
        r3  = r3 - 1                       ;

                r0l = *HST:X               ;HWEM UNMASK
                r0 = r0 & 0xFDFF           ;
        *HST:X = r0l                       ;ストリーム出力再開

        %DisableInterrupt()                ;
        %SetSR_HO_Enable()                 ; HO Enable
        %EnableInterrupt()                 ;

        if(r5!=0) jmp $                    ;JPEGストリーム出力終了待ち

JPEGCompress_OK:
                jmp $;                       正常終了

JPEGCompress_NG:
                jmp $;                       異常終了
```

(6/7)

```
;1MCU(RGB)サイズ設定関数
;input  r2l = SAMPLINGRATIO(0x11,0x21,0x22,0x41)
;output r2l = RGBSIZE(192,384,768)
SetRGBSize:
              r2 = r2 & 0xFFFF            ;
              r3 = r2 - 0x11              ;
              if(r3==0) jmp _0x11         ;
              r3 = r2 - 0x21              ;
              if(r3==0) jmp _0x21         ;
              r3 = r2 - 0x22              ;
              if(r3==0) jmp _0x22         ;
              r3 = r2 - 0x41              ;
              if(r3==0) jmp _0x41         ;
              jmp JPEGCompress_NG         ;
_0x11:
              r2l = 192                   ;R(8x8)+G(8x8)+B(8x8)
              ret                         ;
_0x21:
              r2l = 384                   ;(R(8x8)+G(8x8)+B(8x8))*2
              ret                         ;
_0x22:
_0x41:
              r2l = 768                   ;(R(8x8)+G(8x8)+B(8x8))*4
              ret                         ;
;---------------------------------------------------------------------
;-     Int handler                                                   -
;---------------------------------------------------------------------
;ストリーム出力用割り込みハンドラ
;dp0 = ストリームバッファアドレス
;r3  = 転送サイズ
;r5  = 転送終了フラグ
OutPutInt:
              r0l = *ICR9:X               ;
       if(r3>0) jmp OutContinue           ;ストリーム転送終了?

       %DisableInterrupt()                ;ストリーム・データ転送終了処理
       %SetSR_HO_Disable()                ;
       %EnableInterrupt()                 ;

              r0l = *HST:X                ;HREM MASK
              r0 = r0 | 0x200             ;
       *HST:X = r0l                       ;ストリーム出力停止

       clr(r5)                            ;
       jmp OutRet                         ;

OutContinue:
       r0h =*dp1++                        ; perform output from HDO
       r0  = r0 sll 8                     ;
       *HDT:x=r0h                         ;
       r3  = r3 - 1                       ;
OutRet:
       reti                               ;
```

(7/7)

```
;画像入力用割り込みハンドラ
;dp0 = RGBバッファ・アドレス
;r2  = 転送サイズ
;r5  = 転送終了フラグ
InPutInt:
              r0h =*HDT:x                  ; perform input from HDI
       r0  = r0 srl 8                      ;
       *dp0++=r0h                          ;
              r0l = *ICR8:X                ;
       r2  = r2 - 1                        ;
       if(r2>0) jmp InContinue             ;1MCU転送終了?
ReadInEnd:
       %DisableInterrupt()                 ;1MCU転送終了処理
       %SetSR_HI_Disable()                 ;
       %EnableInterrupt()                  ;

       clr(r5)                             ; set transfer end flag

              r0l = *HST:X                 ;HWEM MASK
              r0 = r0 | 0x100              ;
       *HST:X = r0l                        ;画像入力停止

InContinue:
       reti                                ;

end
```

付録.3　データ入力／出力用タイミング・ファイル

・データ出力用タイミング・ファイル（hostrd8.tmg）

```
;-------------------------------------------------------------------
;-     Copyright (C) NEC Coporation 2002                           -
;-     NEC CONFIDENTIAL AND PROPRIETARY                            -
;-     All rights reserved by NEC Coporation                       -
;-     Use of copyright notice does not evidence publicaton        -
;-------------------------------------------------------------------
; file:        HOSTRD8.TMG
; purpose:     Demonstrate eight bit data transfer via host interface
;              Read from high-byte of built-in host port's HDT register
;              Output data values to data file HOSTRD8.DAT
; software:    HOSTDEMO.LNK must be loaded

set pin hcs = 1               ; terminate any read access, which might
set pin hrd = 1               ; be active

open output "output.dat"      ; data are stored in file "output.dat"
output format showbase unsigned hex,   ; select output format
do                            ; loop to read 10 bytes of data
   wait cond pin hre == 0     ; wait untill data is availiable
   wait cond pin hcs == 1     ; and no write is in progress
   set pin hcs = 0            ; perform the access...
   set port ha = 1            ; select lower byte of HDT
   set pin hrd = 0            ; start output
   wait 100ns                 ; access duration
   output port hd & 0xFF      ; output host data to file (mask sign bits)
   set pin hrd = 1            ; end output
   set pin hcs = 1            ; end host port access
enddo

close output
end
```

・データ入力用タイミング・ファイル（hostwr8.tmg）

```
;-------------------------------------------------------------------
;-     Copyright (C) NEC Coporation 2002                           -
;-     NEC CONFIDENTIAL AND PROPRIETARY                            -
;-     All rights reserved by NEC Coporation                       -
;-     Use of copyright notice does not evidence publicaton        -
;-------------------------------------------------------------------
; file:        HOSTWR8.TMG
; purpose:     Demonstrate eight bit data transfer via host interface
;              Input data values from file HOSTWR8.DAT
;              Write to high-byte of built-in host port's HDT register
; software:    HOSTDEMO.LNK must be loaded.

set pin hcs = 1               ; terminate any write access, which might
set pin hwr = 1               ; be active

set pin hcs  = 0              ; initialize low byte with 0
set port ha = 0               ;
set pin hwr = 0               ;
set port hd = 0               ;
wait 100ns                    ;
set pin hwr = 1               ;
set pin hcs = 1               ;
set port hd = @               ;

open input "input.dat"        ; data are read from the file "input.dat"
input format hex              ; select hex input format
do                            ; loop to read 10 bytes of data
   wait 1000ns                ; access duration
   wait cond pin hwe == 0     ; wait till write is allowed
   wait cond pin hcs == 1     ; and no read is in progress
   set pin hcs = 0            ; perform the access...
   set port ha = 1            ; select higher byte of HDT
   set pin hwr = 0            ; start input
   input port hd              ; input host data from file
   wait 100ns                 ; access duration
   set pin hwr = 1            ; end input
   set pin hcs = 1            ; end access
enddo

close input
end
```

〔メ　モ〕

【発　行】

NECエレクトロニクス株式会社

〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

【ホームページ】

NECエレクトロニク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.necel.co.jp/

【営業関係お問い合わせ先】

下記のページに最新版のお問い合わせ先が記載されています。
URL(アドレス)　http://www.necel.com/ja/contact/contact_j.html

【技術的なお問い合わせ先】

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

【資料請求先】

NECエレクトロニクス特約店または上記ホームページ記載の営業関係お問い合わせ先へお申し付けください。
