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-B02
μSAP705100-B02
μSAP70732-B02

JBIG ミドルウエア

保守／廃止

対象デバイス

μSAP703000-B02：V850ファミリ™
μSAP705100-B02：V830ファミリ™
μSAP70732-B02　：V810ファミリ™

資料番号　U11057JJ3V1UM00（第3版）
発行年月　October 1997 NS

保守／廃止

〔メ　モ〕

目　次　要　約

保守／廃止

〔メ　モ〕

保守／廃止

T.82，T.85 に関わる特許権について

ITU-T 勧告 T.82，T.85 に準拠したシステムについては，複数の特許権が存在しております。

これらの特許権に関する必要な権利処理は，お客様の方にてご対応いただきますようお願いいたします。

当社は，これらの特許権に関して一切責任を負いかねますのでご了承ください。

M7A 94.11

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
p. 15	1.3.3(1), (2)に V830 ファミリ追加
p. 18	1.3.4(2)V830 ファミリ　追加
p. 19	1.3.5(2)V830 ファミリ　追加
p. 26	2.1.4　状態遷移　記述変更
P. 33	2.2.1(12)reset　説明追加
p. 29, 35, 47, 52	Tx, mrk_Tx, mrk_yAT, next_AT_line 追加
P. 35	(17) Tx, mrk_Tx, mrk_yAT, next_AT_line　説明追加および変更
p. 36	表２－４　reset, SDNORM／SDRST による Tx, mrk_Tx, mrk_yAT, next_AT_line の設定　追加
p. 36	図２－７　mrk_yAT と L0（１ストライプあたりのライン数）の関係　追加
p. 38	図２－８　ATMOVE 有効範囲　追加
p. 39	図２－９　ユーザ側 AT 情報設定例　追加
p. 40	図２－10　ATMOVE 圧縮処理例　追加
p. 41	図２－11　ATMOVE 伸長処理例　追加
p. 44	(18)newlen　⑥　説明追加
p. 48, 49	表２－６，２－７　記述追加
p. 51	2.2.2(2)伸長処理系　記述追加
p. 53, 54	表２－９，２－10　記述追加
p. 59	3.1(2)V830 ファミリ　追加
p. 63	3.2.2 V830 ファミリ　追加
p. 72	図４－１　圧縮系システム例　追加
p. 73	図４－２　伸長系システム例　追加
p. 75	図４－４　V830 ファミリ・メモリ・マップ例（V830 の場合）　追加
p. 83	付録Ｂ　総合索引　追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

保守／廃止

はじめに

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

目　的　このマニュアルは，次の構成に示す μSAP703000-B02，705100-B02，70732-B02，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
- システム例
- 付　録

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××××または××××H
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）　$2^{20}=1024^2$

保守／廃止

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

V810 ファミリに関する資料

資料名／品名	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
V805™	U10917J	U10661J	U10082J
V810™	U10691J		
V820™	U11705J	U11708J	
V821™	U11678J	U10077J	

V830 ファミリに関する資料

資料名／品名	データ・シート	ユーザーズ・マニュアル
V830™	U11483J	U10064J

V850 ファミリに関する資料

品名／資料名		データ・シート	ユーザーズ・マニュアル		活用表	
			ハードウエア編	アーキテクチャ編	レジスタ	インストラクション
V851™	μPD703000（マスクROM） μPD703001（ROMレス）	U10987J	U10935J	U10243J	U10662J	U10229J
	μPD70P3000（PROM）	U10988J				
V852™	μPD703002（マスクROM）	U11826J	U10038J		U10513J	
	μPD70P3002（PROM）	U11827J				
V853™	μPD703003	作成予定	U10913J		—	
	μPD70F3003	U12036J				
V854™	μPD703008，703008Y	作成予定	U11969J		—	
	μPD70F3008，70F3008Y	作成予定				

保守／廃止

V810 ファミリ開発ツールに関する資料

資料名		資料番号
IE-70732-BX-A ユーザーズ・マニュアル		U10667J
IE-70732-MC ユーザーズ・マニュアル		EEU-5016
IE-70742-BX ユーザーズ・マニュアル		U10630J
IE-70741-BX ユーザーズ・マニュアル		U11963J
CA732 ユーザーズ・マニュアル	操作編（UNIX™ベース）	U11013J
	操作編（Windows™ベース）	U11068J
	アセンブリ言語編	U11016J
	C言語編	U11010J
ID732 ユーザーズ・マニュアル	操作編（UNIX ベース）	EEU-5021
	操作編（Windows ベース）	作成予定
	インストレーション編（UNIX ベース）	EEU-5022
	インストレーション編（Windows ベース）	作成予定
RX732 ユーザーズ・マニュアル	基礎編	U10346J
	インストレーション編	U10347J
	テクニカル編	U10490J
AZ732 ユーザーズ・マニュアル	操作編	U10488J
RX-FX732 ユーザーズ・マニュアル	基礎編	U11313J

V830 ファミリ開発ツールに関する資料

資料名		資料番号
IE-705100-MC-EM1　ユーザーズ・マニュアル		U11869J
CA830　ユーザーズ・マニュアル	操作編（UNIX ベース）	U11013J
	操作編（Windows ベース）	U11068J
	アセンブリ言語編	U11014J
	C言語編	U11010J
ID830　ユーザーズ・マニュアル	操作編（UNIX ベース）	作成予定
	操作編（Windows ベース）	作成予定
	インストレーション編（Windows ベース）	作成予定
RX830　ユーザーズ・マニュアル	基礎編	U11730J
	インストレーション編	U11731J
	テクニカル編	U11713J

保守／廃止

V850 ファミリ開発ツールに関する資料

資料名		資料番号
IE-703000-MC-A ユーザーズ・マニュアル ハードウエア編		U10887J
IE-703002-MC ユーザーズ・マニュアル		U11595J
IE-703003-MC-EMI ユーザーズ・マニュアル		U11596J
CA850 ユーザーズ・マニュアル	操作編（UNIX ベース）	U11013J
	操作編（Windows ベース）	U11068J
	アセンブリ言語編	U10543J
	C 言語編	U11010J
ID850 ユーザーズ・マニュアル	操作編（Windows ベース）	U11196J
RX850 ユーザーズ・マニュアル	基礎編	U11037J
	インストレーション編	U11038J
	テクニカル編	U11117J
AZ850 ユーザーズ・マニュアル	操作編	U11181J

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

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

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

保守／廃止

目　次

図　の　目　次

保守／廃止

表 の 目 次

保守／廃止

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す。従来，ハードウエアが行っていた処理をソフトウエアで実現したものです。RISC という高性能プロセッサの出現，そしてRISC が手軽にシステムに組み込める環境が整ってきたために，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。

NEC では V800 シリーズ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。

この製品は，FAX コーデック機能を提供するミドルウエアです。

1.2　JBIG

JBIG（Joint Bi-level Image Group）は，静止2値画像（白と黒の色だけを持つ画像）の標準符号化方式で，ITU-T 勧告 T.82 として勧告されました。JBIG FAX 版は，FAX 用として新たに T.85 として勧告されました。以降では JBIG は T.82，T.85 共通の説明，JBIG FAX 版は T.85 だけの説明を表します。

・高い圧縮性能

従来の MH/MR/MMR 方式よりも文字中心の文書で約 1.1～1.5 倍の高い圧縮性能があります。また，自然画像に対しても圧縮／伸長できるようになりました。

・情報の保存性

圧縮／伸長による情報の欠落がありません(JPEG 方式は欠落する場合があります）。

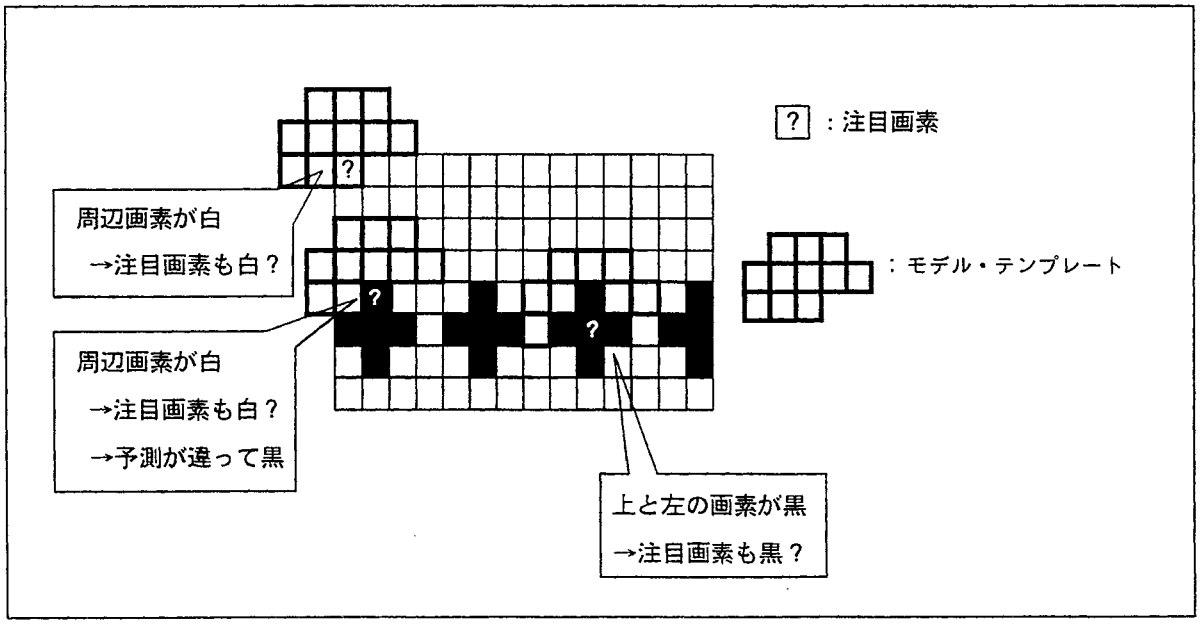
1.2.1　JBIGの符号化

JBIG による符号化は，図1－1で示すように周辺画素（モデル・テンプレート）の状態から注目画素の白と黒を予測して，予測が違う場合だけを符号化します。また，符号データの量を減少させるため，その予測を学習して予測を当たりやすくします。

復号化の場合も同じように予測して学習するため，予測が違う場合だけの情報で十分です。

符号は算術符号化により 0 以上 1 未満の 2 進数小数で表します（**1.2.9 AAE(算術符号化)**参照）。

図1－1　JBIGの符号化

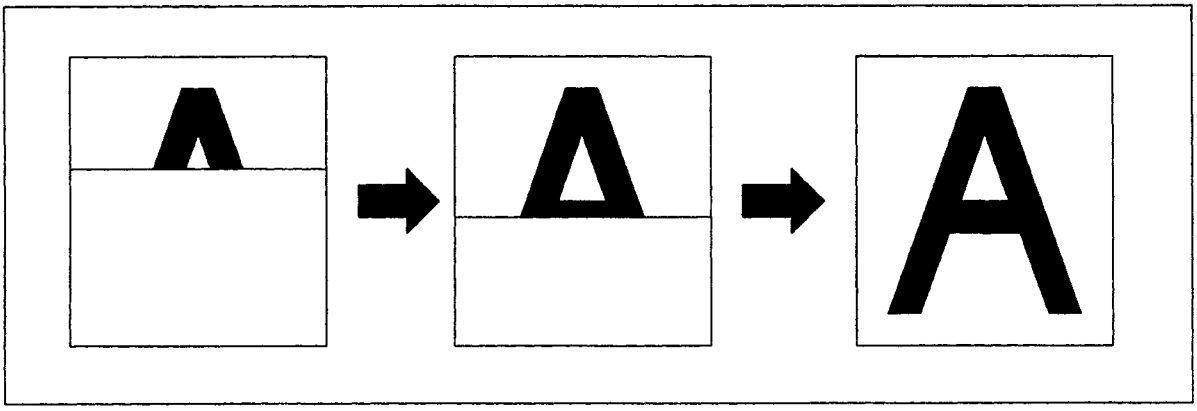


1．2．2　JBIGの伝送方法

JBIGはシーケンシャル伝送（画像を上から順に従って伝送）とプログレッシブ伝送（最初は粗い画像を伝送して，追加情報の伝送によって画質の精度を上げていく伝送）の2つの方式をサポートしています。プログレッシブ伝送は階層（レイヤ）的な伝送方式です。

JBIG FAX版ではシーケンシャル伝送だけサポートしています。図1−2にシーケンシャル伝送について示します。

図1－2　シーケンシャル伝送



1．2．3　シーケンシャル伝送

2値画像データのシーケンシャル伝送処理例を次に示します。

図1－3　2値画像データのシーケンシャル伝送処理例

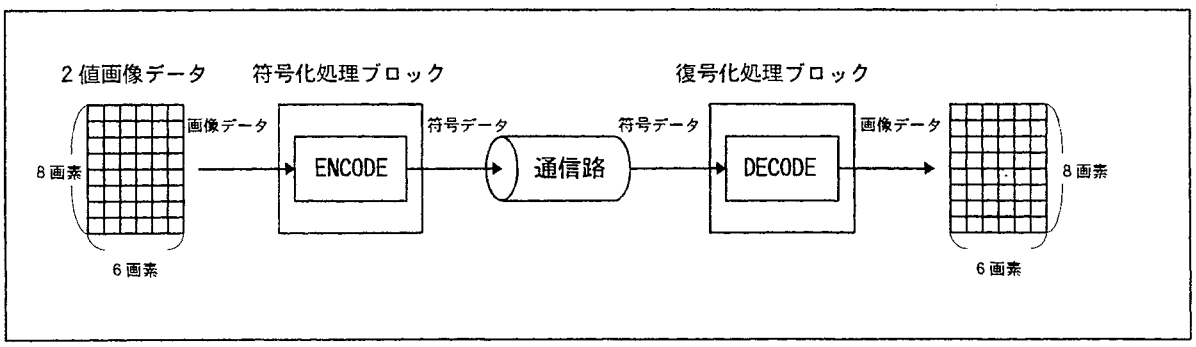


図1－3の2値画像データのレイヤ0（階層）を伝送するために必要なJBIG FAX版フリー・パラメータを次に示します。

・差分レイヤ[注1]の数（D）＝0
・ビット・プレーン[注2]の数（P）＝1
・レイヤにおける水平方向画素数（Xd）＝6
・レイヤにおける垂直方向画素数（Yd）＝8
・レイヤにおけるストライプ[注3]の画素数（L0）＝4

注1. プログレッシブ伝送に使用するレイヤです。シーケンシャル伝送の場合は、必要がないため差分レイヤの数は0になります。

2. JBIG FAX版は白黒の2値画像のためビット・プレーンの数は1になります。2以上はカラー対応時に使用します。

3. 1ページを数ラインごとに分割したブロックです。

1．2．4　符号化処理

JBIG FAX 版の符号化処理では，次に示す 4 つのブロックで構成されています。

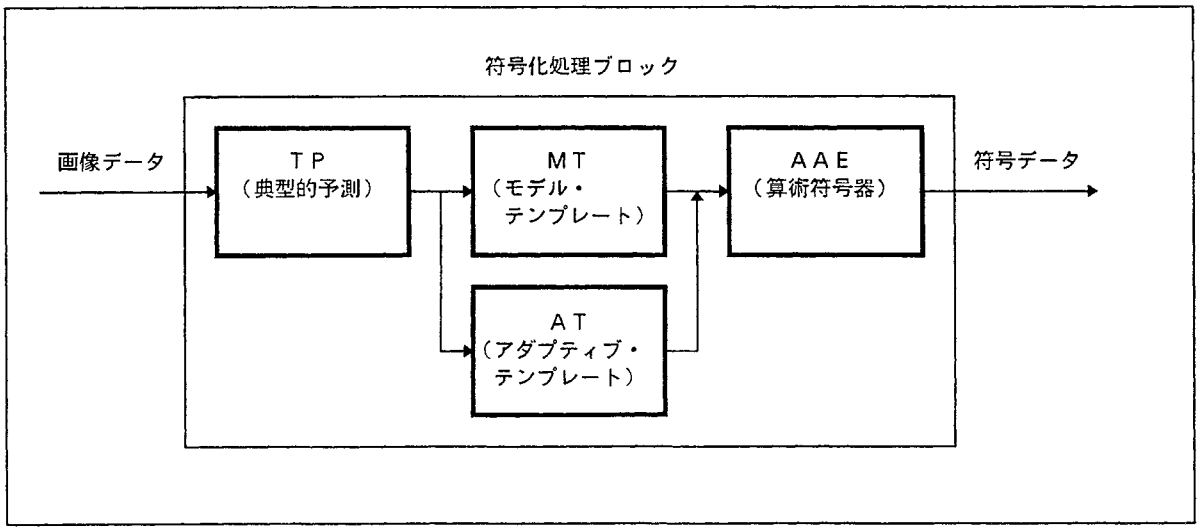
TP ：符号化対象画素数を減少させる処理です。符号化するラインが 1 つ前のラインと同じ場合，1 ビットの情報で表して符号化しません（**1.2.6 TP(Typical Prediction)**参照）。

MT ：符号化する注目画素の周辺画素パターンです（**1.2.7 MT(Model Template)**参照）。

AT ：モデル・テンプレート(MT)の参照画素の 1 つで，位置を移動できる特別な画素です。ディザ画像のような一定周期に相関がある画素の符号化に効果があります（**1.2.8 AT(Adaptive Template)**参照）。

AAE：算術符号器です。与えられた画素はモデル・テンプレート(MT)の内容から白黒を予測，学習して符号データを出力します（**1.2.9 AAE(算術符号化)**参照）。

図1－4　符号化処理ブロック図



1．2．5　復号化処理

JBIG FAX 版の復号化処理では，次に示す 4 つのブロックで構成されています。

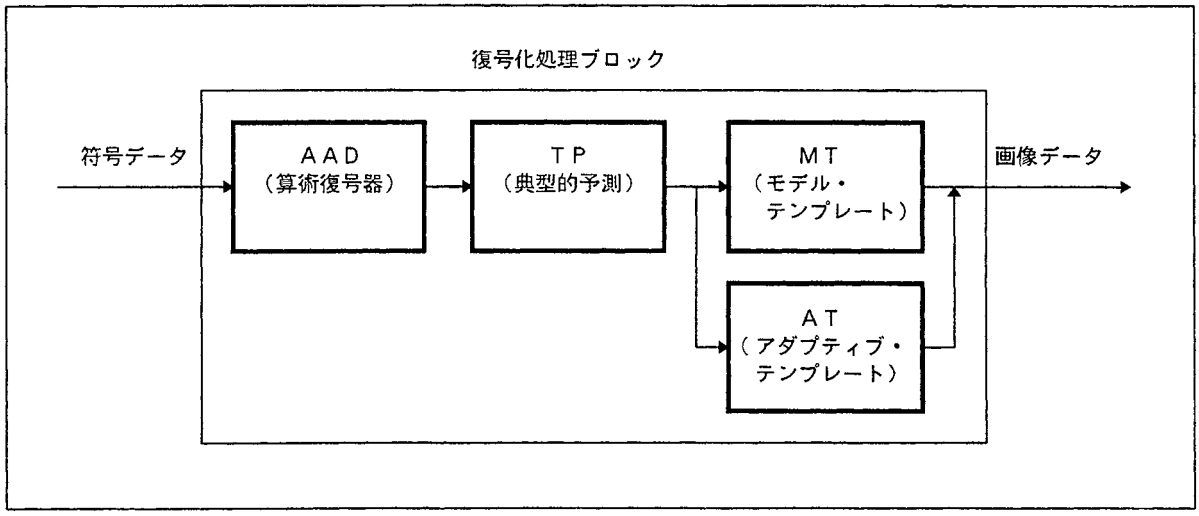
AAD：算術復号器です。

TP ：復号化対象画素数を減少させる処理です。復号化するラインが 1 つ前のラインと同じ場合，1 ライン前の画素データを出力します（**1.2.6 TP(Typical Prediction)**参照）。

MT ：復号化する注目画素の周辺画素パターンです（**1.2.7 MT(Model Template)**参照）。

AT ：モデル・テンプレート(MT)の参照画素の 1 つで，位置を移動できる特別な画素です。ディザ画像のような一定周期に相関がある画素の符号化に効果があります（**1.2.8 AT(Adaptive Template)**参照）。

図１－５　復号化処理ブロック図

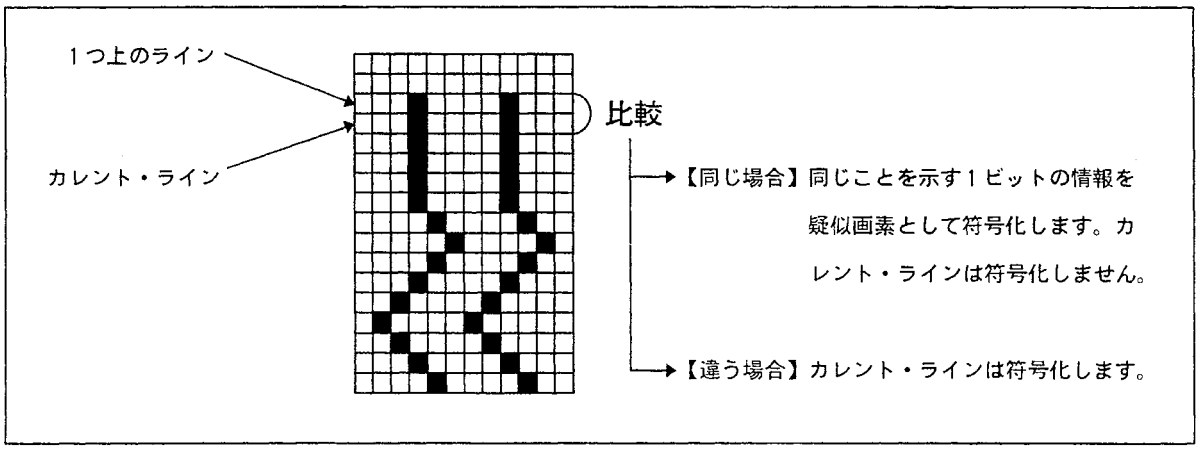


１．２．６　ＴＰ（Typical Prediction）

符号化または復号化する対象画素数を減少させる処理です。符号化処理の場合，符号化するライン（カレント・ライン）が１つ上のラインと同じ場合，１ビットの情報で表して符号化しません。復号化処理の場合，復号化するラインが１つ上のラインと同じ場合，１つ上のラインの画素データを出力します。

図１－６にカレント・ラインの画素との比較を示します。カレント・ラインと１つ上のラインを比較してカレント・ラインと同じ場合は，同じことを示す１ビットの情報を疑似画素として符号化して，カレント・ラインは符号化しません。この処理によって符号化の対象画素が減り，符号データを減少することができます。

図１－６　カレント・ラインとの比較

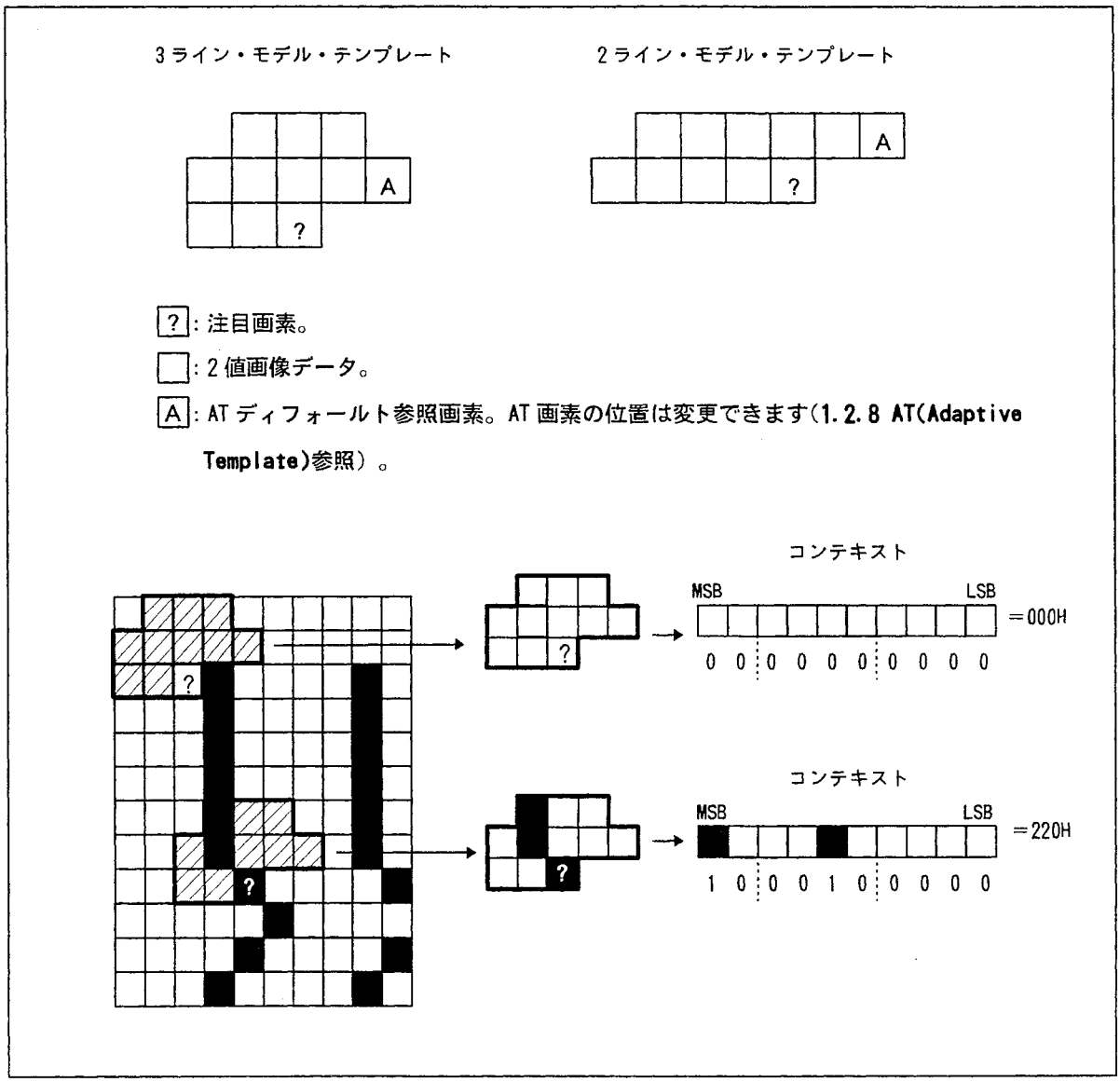


1．2．7　ＭＴ（Model Template）

モデル・テンプレートは，符号化するときに予測モデルとして使用する参照画素パターンです。モデル・テンプレートには，3ライン・モデル・テンプレートと2ライン・モデル・テンプレートがあります。

図1－7にモデル・テンプレート（二次元）からコンテキスト（モデル・テンプレートを一次元化したもの）にする手順を示します。

図1－7　モデル・テンプレート

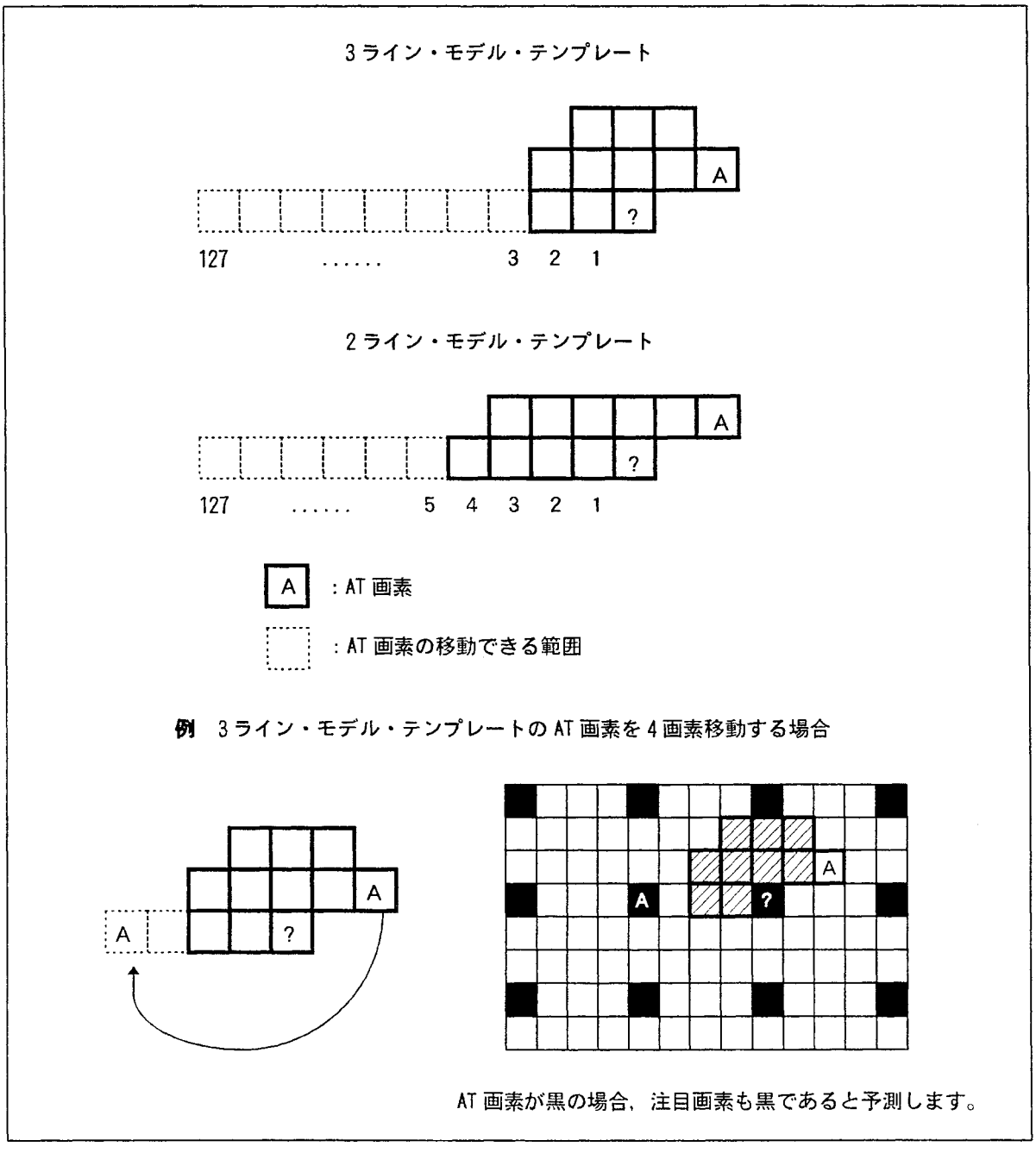


1．2．8　AT（Adaptive Template）

ATはモデル・テンプレートの参照画素の1つです。図1−8の画素“A”を移動させると，ディザ画像のように一定周期で強い相関がある画素の符号化に効果があります。

AT画素は，最大127画素まで移動できます。ただし，モデル・テンプレート内の画素と重ならないように設定してください。

図1－8　AT画素



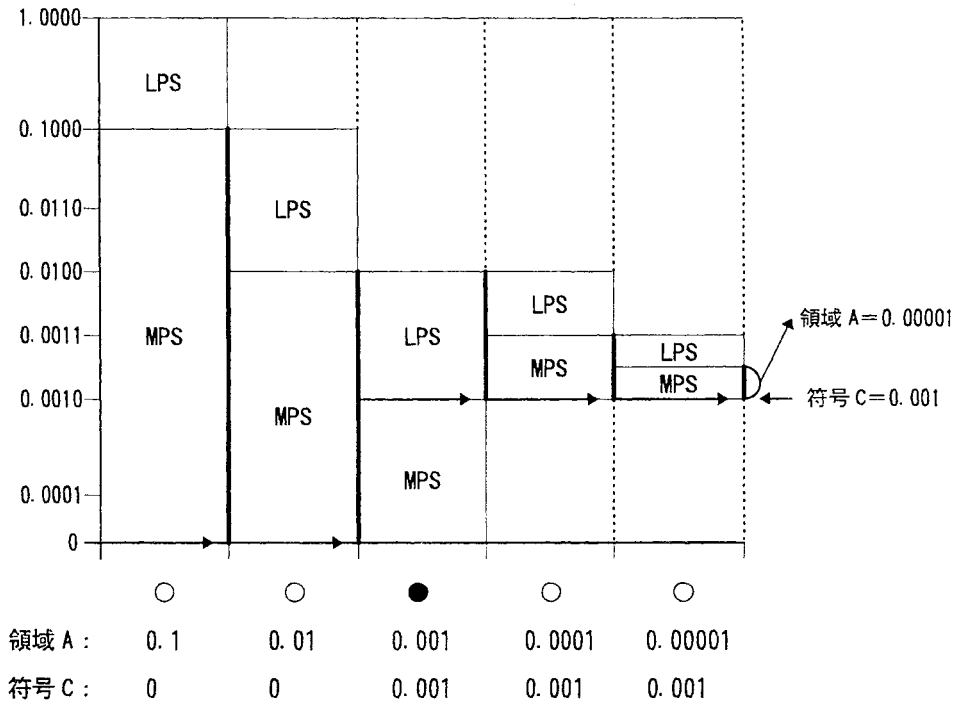
1．2．9　AAE（算術符号化）

JBIGでは周辺画素（モデル・テンプレート）の状態から注目画素の白黒を予測します。その予測結果を算術符号化を使用して，0以上1未満の2進数小数で示します。

図1－9に白の予測が当たる確率1／2，外れる確率1／2とした場合の符号化の例を示します。符号化するシンボルは，多く現れる方を優勢シンボル(MPS)，少ない方を劣勢シンボル(LPS)と呼びます。図1－9では白が現れる方をMPSとしています。

図1－10に白の予測が当たる確率1／2，外れる確率1／2とした場合の復号化の例（符号Cのデータを復号化する例）を示します。

図1－9　算術符号化の例（P0＝0.1，P1＝0.1の場合）



A：白黒の順列が現れる確率を示す領域（初期値＝1.00）。

C：データとなる値（初期値＝0）。

P0：MPSの生起確率。

P1：LPSの生起確率。

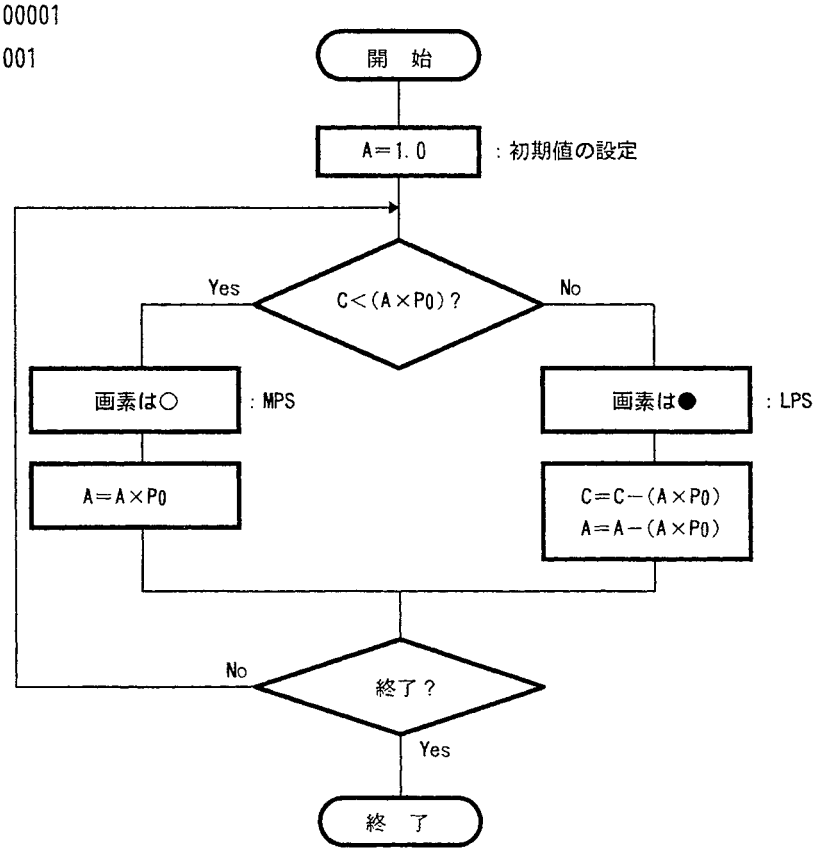
【符号化の式】　領域A：MPSの場合　A=A×P0，　LPSの場合　A=A×P1

符号C：MPSの場合　C=C，　LPSの場合　C=C+(A×P0)

図１－10　復号化処理フロー例

領域 A＝0.00001

符号 C＝0.001



A　：白黒の順列が現れる確率を示す領域（初期値＝1.00）。

C　：データとなる値（初期値＝0）。

P_0：MPS の生起確率。

P_1：LPS の生起確率。

1 回目のループ：C＝0.001＜(1.0×0.1)＝0.1 になり 1 画素目は○

A＝1.0×0.1＝0.1

2 回目のループ：C＝0.001＜(0.1×0.1)＝0.01 になり 2 画素目は○

A＝0.1×0.1＝0.01

3 回目のループ：C＝0.001＝(0.01×0.1)＝0.001 になり 3 画素目は●

C＝0.001－(0.01×0.1)＝0

A＝0.01－(0.01×0.1)＝0.01－0.001＝0.001

4 回目のループ：C＝0＜(0.001×0.1)＝0.0001 になり 4 画素目は○

A＝0.001×0.1＝0.0001

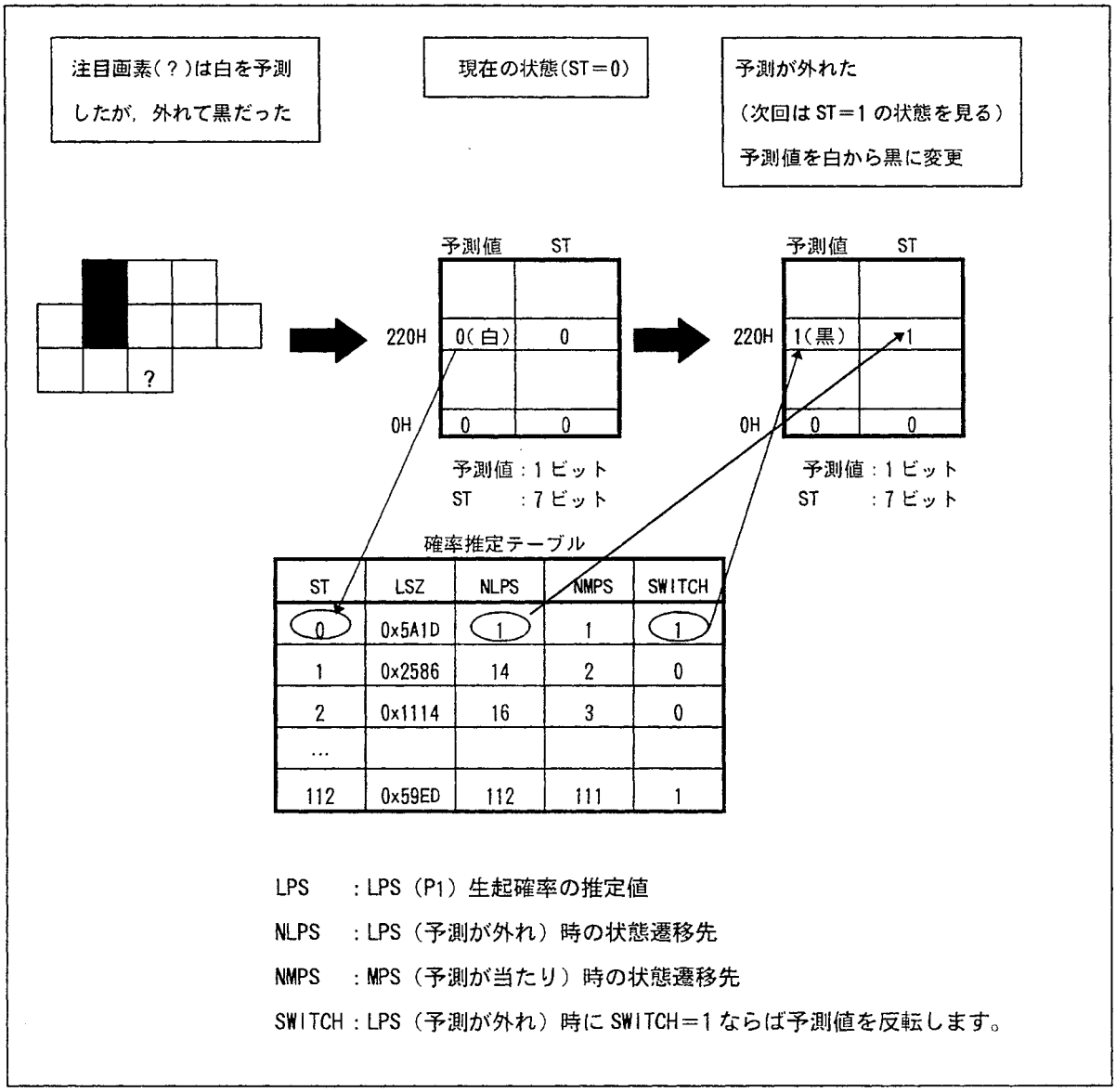
5 回目のループ：C＝0＜(0.0001×0.1)＝0.000001 になり 5 画素目は○

算術符号化に使用するコンテキスト・テーブルと確率推定テーブルについて説明します。

コンテキスト・テーブルは，モデル・テンプレート（10 画素）の値から予測する画素の値とその状態番号を記憶するテーブルです。

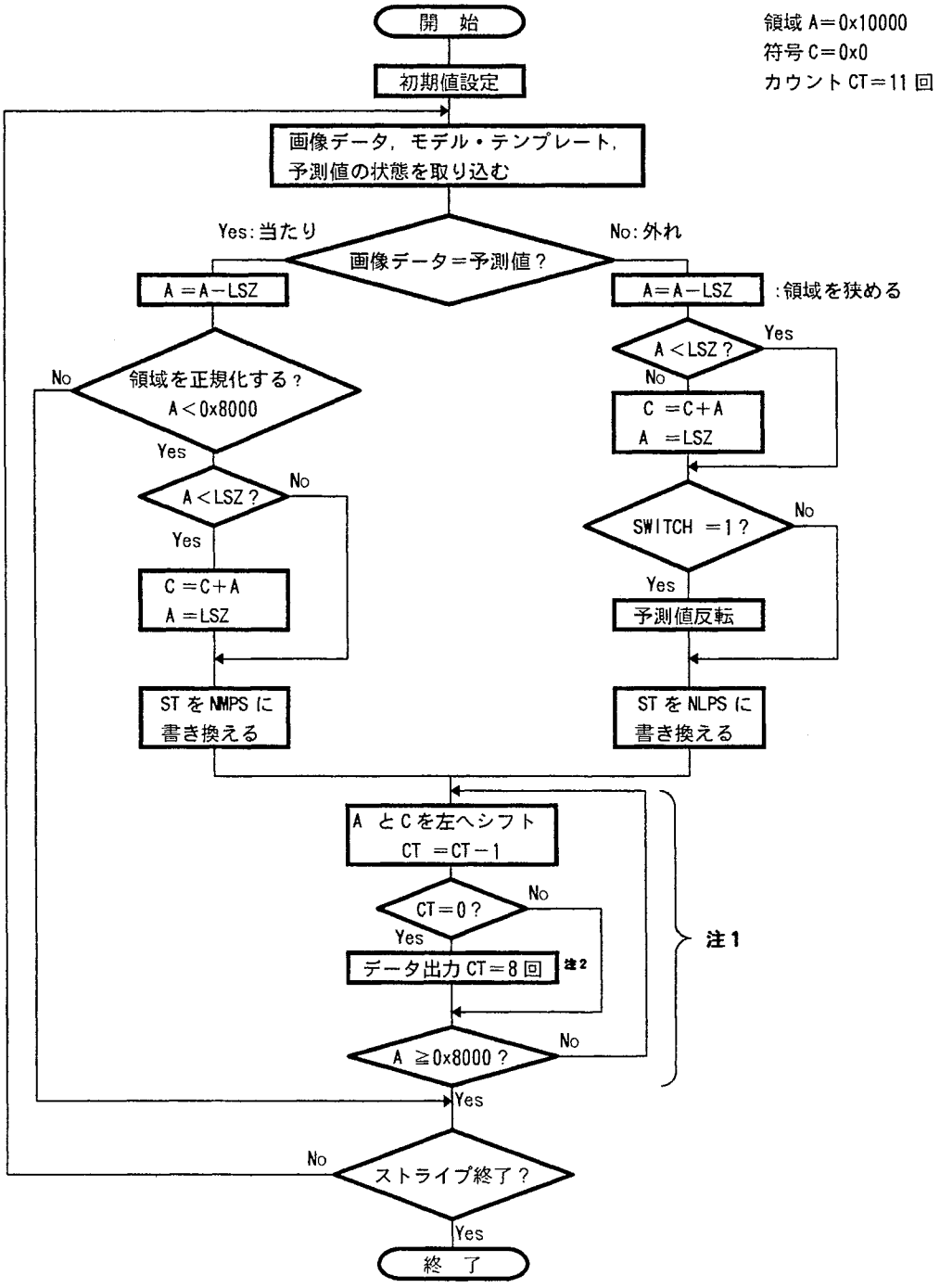
確率推定テーブルは，LPS の領域幅（LSZ），コンテキスト・テーブルが示す次の状態遷移番号（NLPS, NMPS），予測値反転（SWITCH）を示すテーブルです。これらの値は統計的手法によって求められたものです。

図1－11　コンテキスト・テーブルと確率推定テーブル



算術符号化処理の概略フローを図1－12 に示します。今まで 0 以上 1 未満で示していた領域 A の値を 0x8000 以上 0x10000 以下の 16 進数で示します。

図1－12　算術符号化処理の概略フロー



注1. 正規化(Renormalization)
A の領域が 0x8000 より小さくなった場合，A の値を LSZ より大きくするため，A≧0x8000 になるまで A と C の値を左へシフト(2 倍)します。

2. 符号データ出力。

領域 A，符号 C のレジスタ構成とカウント CT を説明します。

表１－１　レジスタ構成

	MSB　　LSB
領域 A	00000000 0000000a aaaaaaaa aaaaaaaa
符号 C	0000cbbb bbbbbsss xxxxxxxx xxxxxxxx

備考　a：白黒の順列が現れる確率を示す領域で 0x00000～0x10000 を示します。
b：符号データの 8 ビットを示します。
x：領域 A のデータを受け取るビット。
s：正規化で左へシフトするときの途中ビット。
c：キャリー・ビット。

カウント CT は，符号 C から JBIG の符号データ(8 ビット)を取り出すためのカウンタです。

CT へ設定する値は，初期設定が 11 回，次から 8 回になります。表 1－1 に示したように値が変わります。

“x”の部分が符号データとして扱われる“b”の部分までのシフト数は 11 回で，その符号データを出力したあと“s”の部分が“b”の部分までのシフト数は 8 回だからです。

1.2.10　JBIG符号データの構造

符号データ（BIE）は，ヘッダ（BIH）とデータ（BID）から構成されています。

BIE	BIH	DL			1バイト	最初に送るレイヤ
		D			1バイト	差分レイヤ数
		P			1バイト	ビット・プレーン数
		—			1バイト	予約
		Xd			4バイト	レイヤdにおける水平方向の画素数
		Yd			4バイト	レイヤdにおける垂直方向の画素数
		L0			4バイト	レイヤdにおけるストライプのライン数
		Mx			1バイト	AT画素に許される最大水平オフセット
		My			1バイト	AT画素に許される最大垂直オフセット
		Order	—	(MSB)	1ビット	予約
			—		1ビット	予約
			—		1ビット	予約
			—		1ビット	予約
			HITOLO 注1		1ビット	上位から転送
			SEQ 注1		1ビット	シーケンシャル転送
			ILEAVE 注1		1ビット	転送順序1
			SMID 注1	(LSB)	1ビット	転送順序2
		Options	—	(MSB)	1ビット	予約
			LRLTWO		1ビット	テンプレートの設定
			VLENGTH		1ビット	NEWLENマーカの設定
			TPDON 注1		1ビット	TPの設定
			TPBON		1ビット	TPの設定
			DPON 注1		1ビット	DPの設定
			DPPRIV 注1		1ビット	DPの設定
			DPLAST 注1	(LSB)	1ビット	DPの設定
		DPTABLE 注1			0/1728バイト	DPの設定
	BID 注2	フローティング・マーカ・セグメント 注3	ATMOVE 注4	ESC	1バイト	エスケープ(0xFF)
				ATMOVE	1バイト	AT移動(0x06)
				yAT	4バイト	AT移動開始ライン
				Tx	1バイト	ストライプごとのAT水平方向移動画素数
				Ty 注1	1バイト	ストライプごとのAT垂直方向移動画素数
			NEWLEN 注5	ESC	1バイト	エスケープ(0xFF)
				NEWLEN	1バイト	新しいライン数(0x05)
				Yd	4バイト	ページのライン数の再定義
			COMMENT 注6	ESC	1バイト	エスケープ(0xFF)
				COMMENT	1バイト	コメント(0x07)
				Lc	4バイト	コメント長
				コメント	可変長	Lcで指定した長さのコメント
		SDE	PSCD		可変長	ストライプ・データ
			ESC		1バイト	エスケープ(0xFF)
			SDNORM/SDRST		1バイト	0x02(ストライプ・データ終了) / 0x03(ストライプ・データ終了とリセット)
		フローティング・マーカ・セグメント				
		SDE				
		⋮				
		フローティング・マーカ・セグメント				
		SDE				

注 1. JBIG FAX 版では使用しません。

2. BID を途中で終了する場合は，次に示すアボート・マーカ（2 バイト）を付加してください。

   ESC(1 バイト：0xFF)

   ABORT(1 バイト：0x04)

3. AT 画素の移動，垂直方向画素数(Yd)の変更やコメント追加を行う情報を与えます。

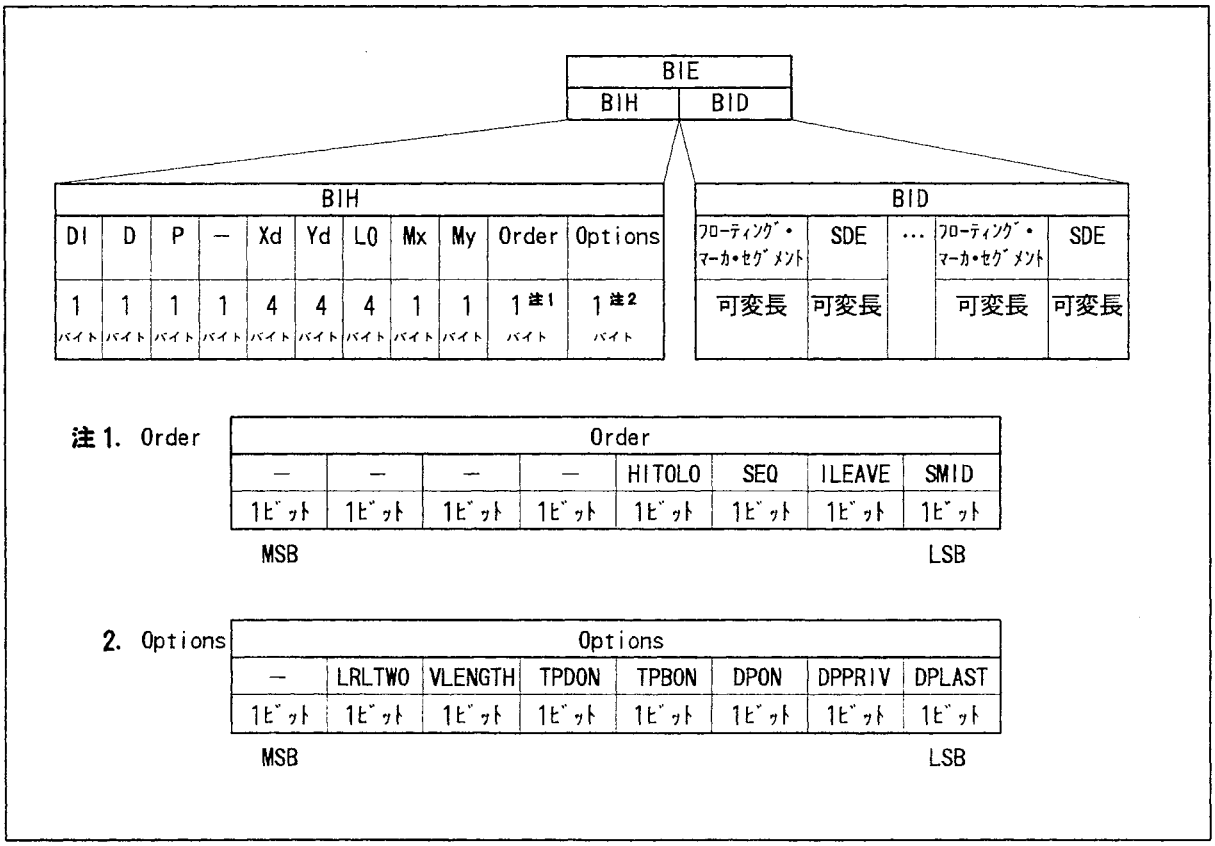
4. AT 画素移動の情報を与えます。

5. 垂直方向画素数(Yd)変更の情報を与えます。

6. コメントの情報を与えます。

注意 SDNORM はストライプ・データの最後を示します。SDRST はストライプ・データの最後を示し，画像の先頭と同様に TP，DP の機能をすべてリセットします。したがって，SDRST によるリセットは圧縮効率を劣化させるので注意してください。

図 1 －13 ＪＢＩＧ符号データの構造



1.3 製品概要

1.3.1 特　徴

・指定ライン分の圧縮／伸長処理。
・JBIG フリー・パラメータのサポート（ITU-T T.85 勧告に準拠）。
・NEC/GHS の C コンパイラの C 言語からの呼び出しができます。
・NEC/GHS のリアルタイム OS に対応（リエントラント可能）。
・JBIG の符号データの取り扱いは，LSB_first，MSB_first の 2 種類が可能。

備考 GHS：Green Hills Software, Inc.

1.3.2 機　能

（1）圧縮処理系

指定した画像データ・バッファおよび JBIG フリー・パラメータより圧縮処理を行い，圧縮符号データ・バッファへ符号データを出力します。

（2）伸長処理系

指定した受信バッファおよび JBIG フリー・パラメータより伸長処理を行い，画像データ・バッファへ伸長した画像データを出力します。

1.3.3 動作環境

（1）動作対象 CPU

・V810 ファミリ（μSAP70732-B02）
★ ・V830 ファミリ（μSAP705100-B02）
・V850 ファミリ（μSAP703000-B02）

（2）対象リンカ

・NEC 製 V810 ファミリ・リンカ(Ver.2.00 以上)
★ ・NEC 製 V830 ファミリ・リンカ(Ver.1.00 以上)
・NEC 製 V850 ファミリ・リンカ(Ver.1.00 以上)
・GHS 製 ELF 版リンカ(Ver.1.8.7B 以上)

（3）必要メモリ・サイズ

（a）ROM（約14Kバイト）

・プログラム

圧縮系ライブラリ　LSB_first 版（約 6K バイト）

MSB_first 版（約 6K バイト）

伸長系ライブラリ　LSB_first 版（約 8K バイト）

MSB_first 版（約 8K バイト）

・確率推定テーブル（約 1K バイト）

（b）RAM

必要メモリ・サイズは画像サイズ，圧縮符号データ・バッファ・サイズなどにより異なります。

・画像データ・バッファ（最低 3 ライン分以上，（1 ストライプ分のライン数＋参照ライン）分のメモリ・サイズを推奨）

・圧縮符号データ・バッファ

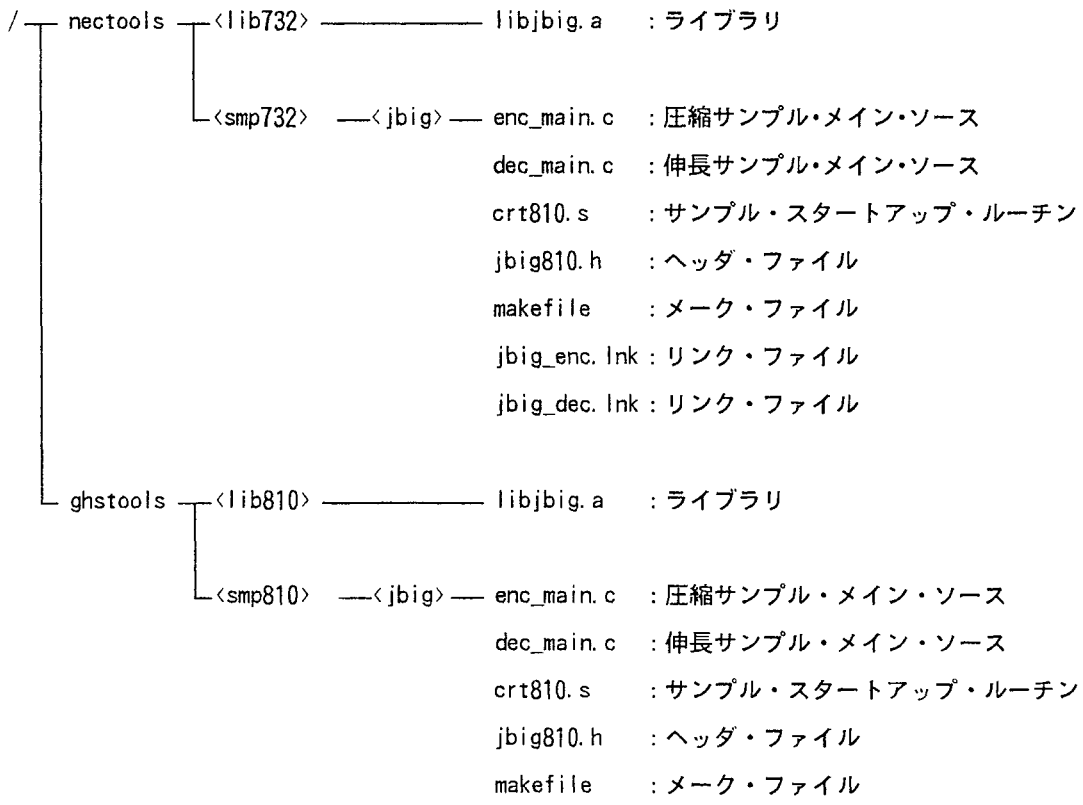
・MPS_ST テーブル（1K バイト）

・入出力パラメータ＋内部情報領域（圧縮／伸長系共通：236 バイト）

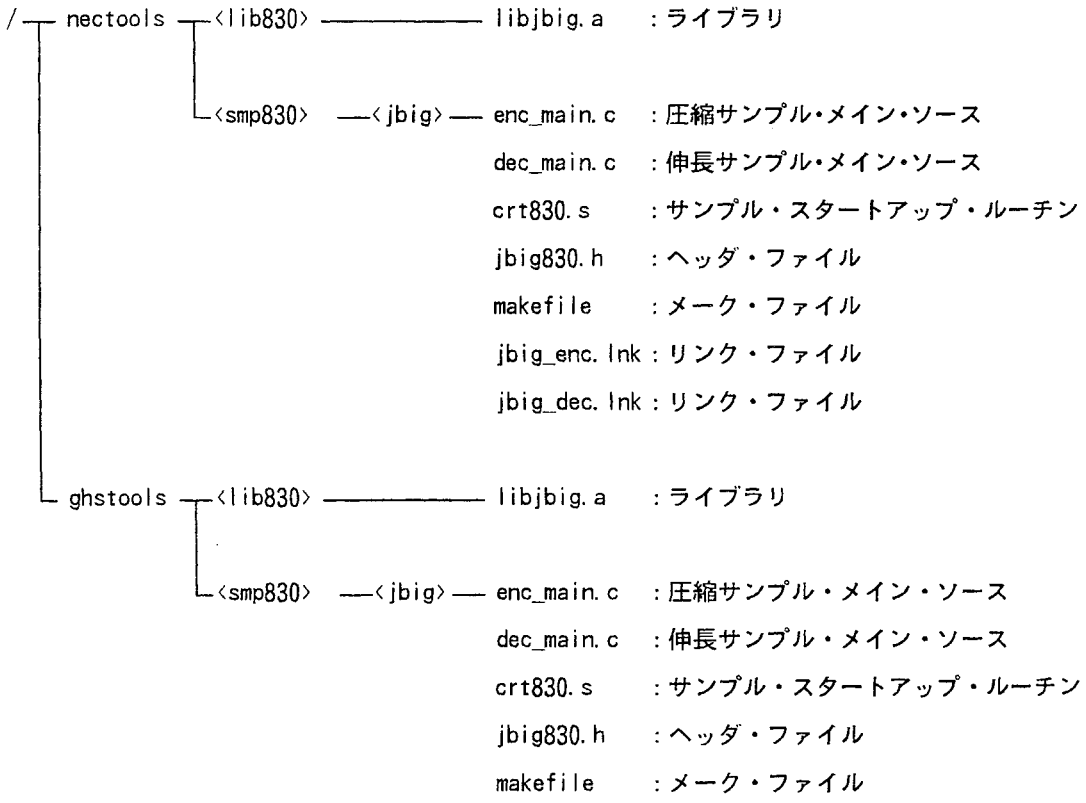
1．3．4　ディレクトリ構成

パッケージの内容を次に示します。

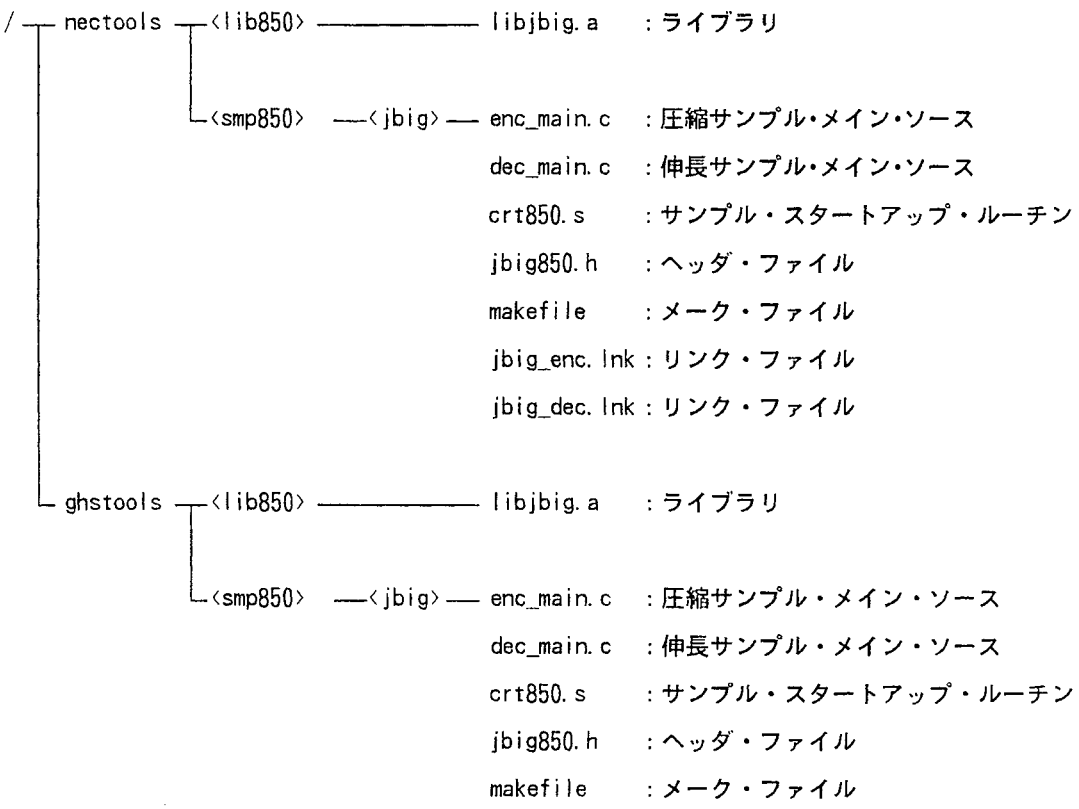
（1）V810ファミリ



★　　（2）V830ファミリ



（3）V850 ファミリ



1.3.5 性　能

（1）V810 ファミリ

条件：CPU　V810(25MHz), 32 ビット・バス, キャッシュ・オン
　　JBIG　TPBON=ON, LRLTWO=OFF, AT=default, Layer=Lowest
　　データ　ITU-T chart1(1728×2376 ドット)

性能：圧縮時間　約 1.8 秒
　　伸長時間　約 1.8 秒

★　（2）V830 ファミリ

条件：CPU　V830(内部：100MHz, 外部：33MHz), 32 ビット・バス,
　　JBIG　TPBON=ON, LRLTWO=OFF, AT=default, Layer=Lowest
　　データ　ITU-T chart1(1728×2376 ドット)

性能：圧縮時間　約 0.33 秒
　　伸長時間　約 0.48 秒

（3）V850ファミリ（目標値）

条件：CPU　V850ファミリ(33MHz)，16ビット・バス

JBIG　TPBON=ON，LRLTWO=OFF，AT=default，Layer=Lowest

データ　ITU-T chart1(1728×2376ドット)

性能：圧縮時間：約1.0秒

伸長時間：約1.2秒

第２章　ライブラリ仕様

２．１　処理概要

２．１．１　標準仕様

JBIG FAX 版についての標準勧告（T. 85）では次に示すフリー・パラメータが定義されています。

表２－１　フリー・パラメータ(1/2)

パラメータ	サイズ	T. 85 勧告	機　　能
DI	1 バイト	0 固定	最初に送るレイヤ
D	1 バイト	0 固定	差分レイヤ数
P	1 バイト	1 固定	ビット・プレーン数（1：2 値画像）
－	1 バイト	Don't care	フィル
Xd 注	4 バイト	1-0xFFFFFFFF	レイヤ d における水平方向画素数
Yd 注	4 バイト	1-0xFFFFFFFF	レイヤ d における垂直方向画素数
L0 注	4 バイト	1-Yd	１ストライプ当たりのライン数
Mx	1 バイト	0-127	AT 画素の最大水平オフセット
My	1 バイト	0 固定	AT 画素の最大垂直オフセット

注　このライブラリで設定できるパラメータです。

表２－１　フリー・パラメータ(2/2)

ﾊﾟﾗﾒｰﾀ	ﾋﾞｯﾄ名	ﾋﾞｯﾄ位置	T.85 勧告	機　　能
Order	－	7	－	－
	－	6	－	－
	－	5	－	－
	－	4	－	－
	HITOLO 注1	3	0 固定	上位から下位へ
	SEQ 注1	2	0 固定	シーケンシャル
	ILEAVE 注1	1	0 固定	インタリーブされたビット・プレーン
	SMID 注1	0	0 固定	中央ループはストライプ
Options	－	7	－	－
	LRLTWO 注2	6	0/1	使用するテンプレートの指定 0：3 ライン・テンプレートの使用 1：2 ライン・テンプレートの使用
	VLENGTH 注2	5	0/1	NEWLEN マーカの設定 0：設定なし 1：設定の可能性あり
	TPDON	4	0 固定	差分レイヤ TP を使用しない
	TPBON 注2	3	0/1	最低解像度レイヤ TP の指定 0：使用しない 1：使用する
	DPON	2	0 固定	DP を使用しない
	DPPRIV	1	0 固定	DP テーブルを使用しない
	DPLAST	0	0 固定	最後の DP を使用しない

注 1. ストライプの処理手順を定義します。

2. このライブラリで設定できるパラメータです。

２．１．２　ライブラリの処理

（１）符号データの MSB_first，LSB_first 処理

符号データは，MSB_first，LSB_first の２種類のライブラリを用意しています。

	MSB_first 符号データ	LSB_first 符号データ
圧縮処理	jbig_enc_m()	jbig_enc_l()
伸長処理	jbig_dec_m()	jbig_dec_l()

（２）処理単位と中断処理

圧縮／伸長処理は指定ライン分の処理をします。また，１ストライプ分の処理を終了した場合，SDNORM/SDRST マーカを付加（圧縮処理時）／検出（伸長処理時）し，１ストライプ終了を示すステータスをライブラリの返り値とします。１ページ分の処理を終了した場合，ページ終了を示すステータスをライブラリの返り値とします。

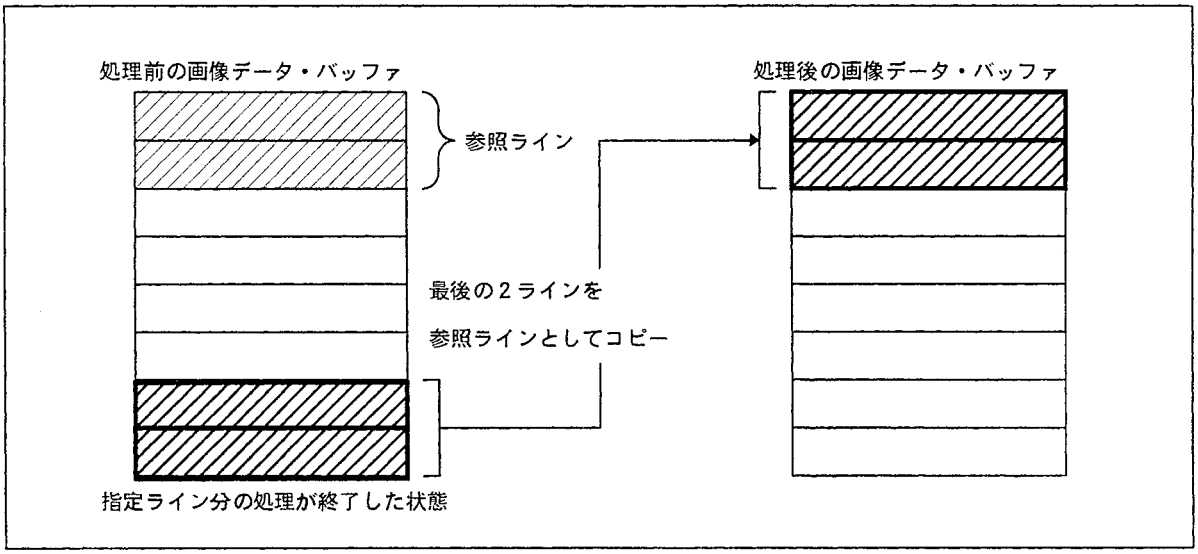
（３）ＢＩＨ（ＪＢＩＧヘッダ）の設定

BIH（JBIG ヘッダ）の設定は，このライブラリでは行っていません。また，BIH の VLENGTH ビットと NEWLEN マーカはチェックしますが，それ以外の BIH と入出力パラメータ間のチェックは行っていません。たとえば，BIH の Mx（AT 画素に許される最大水平オフセット）と ATMOVE マーカの Tx（AT 画素の水平方向のオフセット）の比較はチェックしません。

（４）参照ラインの設定

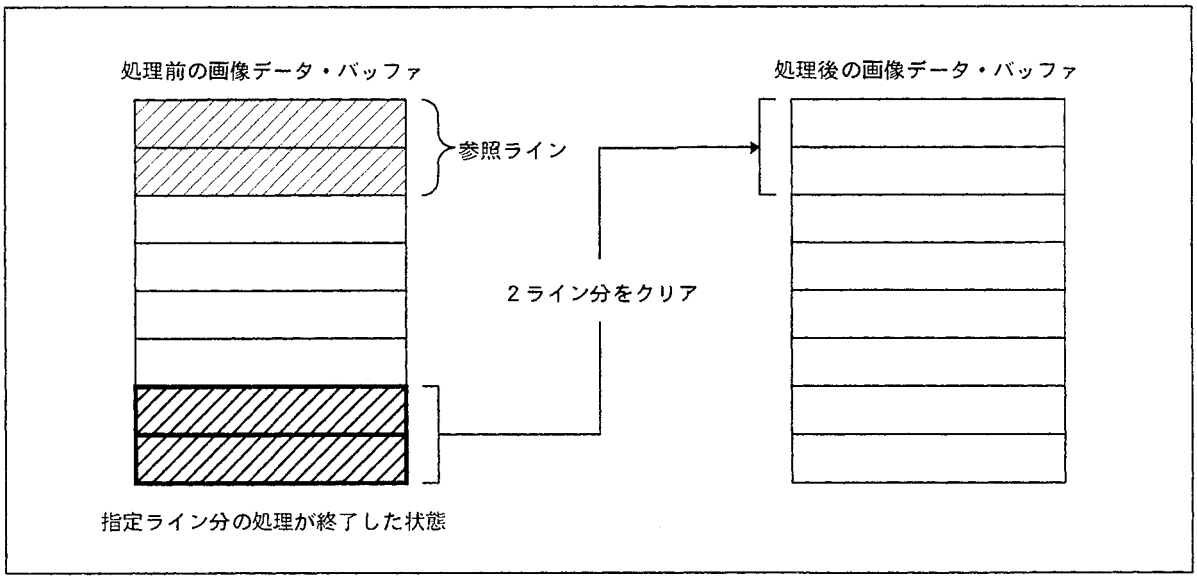
圧縮／伸長処理は指定ライン分を処理したあと，画像データ・バッファの最後の２ライン（３ライン・モデル・テンプレート使用時）または１ライン（２ライン・モデル・テンプレート使用時）を参照ラインの内容として指定されたバッファにコピーします。

図２－１　参照ラインとしてのコピー



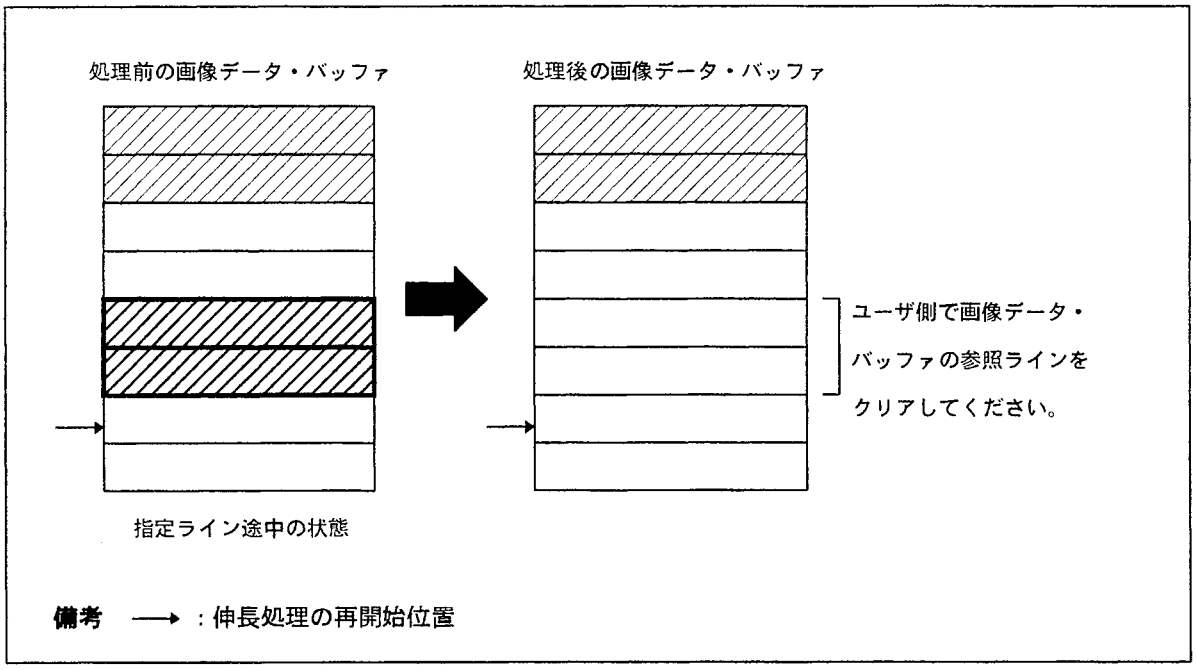
ただし，ストライプ終了で SDRST が付加された場合は，指定されたバッファを参照ライン分クリアします。

図２－２　ストライプ終了がＳＤＲＳＴの場合



注意　伸長処理系の画像データ・バッファの途中にストライプ終了で SDRST が付加された場合は画像データをクリアしないので，ユーザ側で画像データ・バッファの参照ラインをクリアしてください。

図２－３　ストライプ終了，画像データ・バッファの途中から再開する場合



（５）MPS，STテーブルのクリア

SDRST がセットされた場合，ストライプ終了後にライブラリで処理します。

２．１．３　画像データと符号データの取り扱い

（１）画像データと符号データの読み出し／書き込み方式

画像データは，スキャナで最初に走査されるビットをバイトの LSB(最下位ビット)から順に格納します。

符号データは，LSB_first と MSB_first の２種類の方式で読み出し／書き込みを行います。

データ種別	読み出し／書き込み方式
画像データ	LSB_first
符号データ	MSB_first または LSB_first

（２）マーカの取り扱い

次に示すマーカを自動付加（圧縮処理時）または自動検出（伸長処理時）します。

SDNORM/SDRST

ATMOVE

NEWLEN

ABORT

COMMENT（伸長処理だけ）

RESERVE（伸長処理だけ）

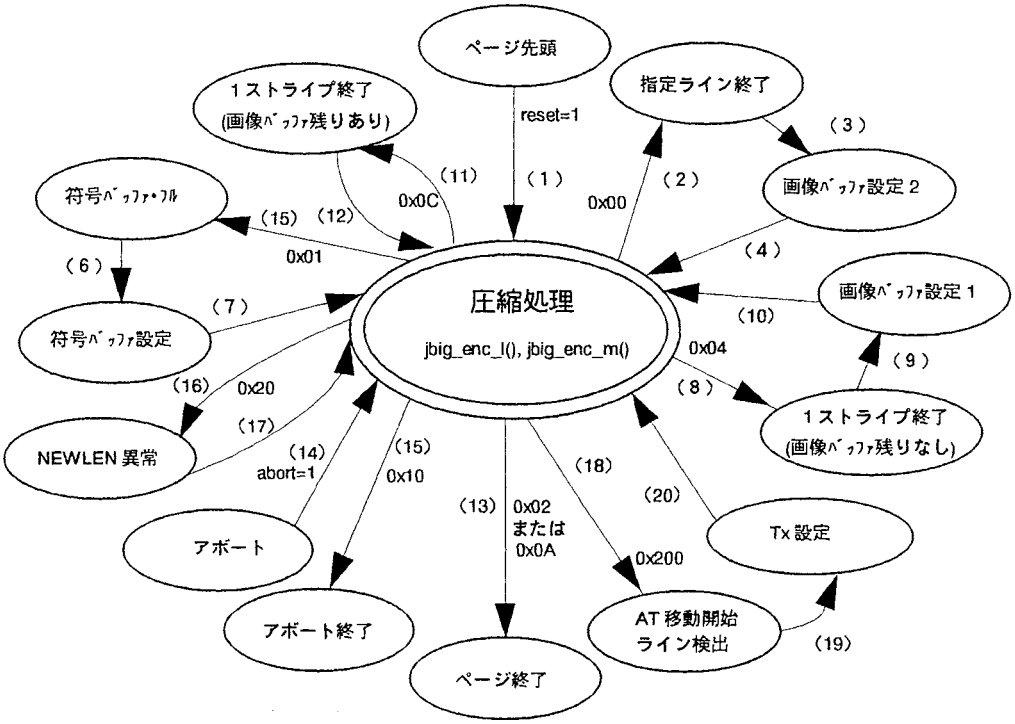
（３）スタッフ・バイトの取り扱い

自動付加（圧縮処理時）または自動廃棄（伸長処理時）します。

★　２．１．４　状態遷移

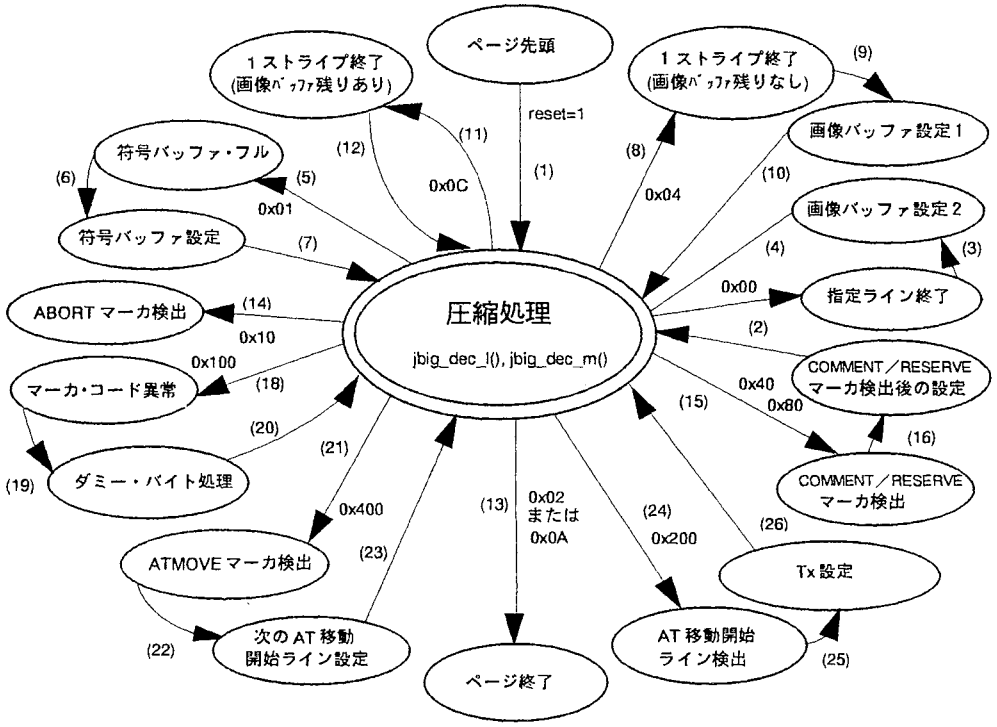
次に，各処理時の状態遷移図を示します。

図２－４　圧縮処理の状態遷移



遷移番号	説　明	返り値
（１）	圧縮処理開始(reset=1 設定)	–
（２）	指定ライン終了	0x00
（３）	画像バッファ設定	–
（４）	圧縮処理再開	–
（５）	符号バッファ・フル	0x01
（６）	符号バッファ設定	–
（７）	圧縮処理再開	–
（８）	１ストライプ終了（画像バッファ残りなし）	0x04
（９）	画像バッファ設定	–
（10）	圧縮処理再開	–
（11）	１ストライプ終了（画像バッファ残りあり）	0x0C
（12）	圧縮処理再開	–
（13）	ページ終了	0x02, 0x0A
（14）	アボート(abort=1 設定)	–
（15）	アボート終了	0x10
（16）	NEWLEN 異常	0x20
（17）	NEWLEN 異常後，圧縮処理再開	–
（18）	AT 移動開始ライン検出	0x200
（19）	Tx 設定	–
（20）	圧縮処理再開	–
上記以外	遷移不可能	–

図２－５　伸長処理の状態遷移



遷移番号	説　明	返り値
(１)	伸長処理開始(reset=1 設定)	－
(２)	指定ライン終了	0x00
(３)	画像バッファ設定	－
(４)	伸長処理再開	－
(５)	符号バッファ・フル	0x01
(６)	符号バッファ設定	－
(７)	伸長処理再開	－
(８)	１ストライプ終了（画像バッファ残りなし）	0x04
(９)	画像バッファ設定	－
(10)	伸長処理再開	－
(11)	１ストライプ終了（画像バッファ残りあり）	0x0C
(12)	伸長処理再開	－
(13)	ページ終了	0x02, 0x0A
(14)	ABORT マーカ検出	0x10
(15)	COMMENT/RESERVE マーカ検出	0x40, 0x80
(16)	COMMENT/RESERVE マーカ処理	－
(17)	伸長処理再開	－
(18)	異常マーカ・コード検出	0x100
(19)	ダミー・バイト処理(invalid_code_sts=0x01 のときのみ)	－
(20)	伸長処理再開	
(21)	ATMOVE マーカ検出	0x400
(22)	次の AT 移動開始ライン設定	－
(23)	伸長処理再開	－
(24)	AT 移動開始ライン検出	0x200
(25)	Tx 設定	－
(26)	伸長処理再開	－
上記以外	遷移不可能	－

２．２　関数仕様

各ライブラリを呼び出すとき（Ｃ言語記述）の仕様を次に示します。

２．２．１　構造体（パラメータ）

JBIG の圧縮／伸長の関数で共通して使用しているパラメータ(J_PARA)を次に示します。

表２－２　パラメータ（J_PARA）

メンバ名	型	説　　明
*pixel_buf	unsigned char	画像データ・バッファ・アドレス
*next_pixel_buf	unsigned char	次の画像データ・バッファ・アドレス
pixel_buf_line	unsigned int	画像データ・ライン数
*code_buf	unsigned char	圧縮符号データ・バッファ・アドレス
code_buf_size	unsigned int	圧縮符号データ・バッファ残りサイズ
*Mps_St_tbl	unsigned char	Mps_St テーブル先頭アドレス
Xd	unsigned int	水平方向画素数
Yd	unsigned int	垂直方向画素数
line_cnt	unsigned int	累積ライン数カウンタ
L0	unsigned int	1 ストライプ当たりのライン数
Options	unsigned char	JBIG BIH Options Byte
reset	unsigned char	1:リセット実行
abort	unsigned char	1:ABORT
sdrst	unsigned char	1 ストライプ処理後，1:SDRST/0:SDNORM
newlen_err_sts	unsigned char	NEWLEN エラー・ステータス
invalid_code_sts	unsigned char	異常マーカ・コード・ステータス
★ Tx	unsigned char	AT 水平方向移動画素数
★ mrk_Tx	unsigned char	ストライプごとの AT 水平方向移動画素数（マーカ付加/検出用）
★ mrk_yAT	unsigned int	AT 移動開始ライン，0:ストライプ先頭（マーカ付加/検出用）
★ next_AT_line	unsigned int	次の AT 移動開始ライン，0:ページ先頭
newlen	unsigned int	ページのライン数の再定義，新しいライン数
length_err	unsigned int	NEWLEN マーカ異常時の length 値
Lc	unsigned int	コメント長
restart_adr	unsigned int	初期化／再開フラグ
reg_area[10]	unsigned int	レジスタ退避エリア
jbg_val[31]	unsigned int	JBIG 変数退避エリア（中断時専用）

（１）＊pixel_buf

画像データ・バッファ・アドレスを示します。

pixel_buf は，圧縮対象ラインのアドレスを指定します。参照ラインのアドレスは設定しないでください。

指定ライン分を処理したあとの pixel_buf は，next_pixel_buf で指定したアドレスが設定されます。

このメンバはワードでアラインした値を入力してください。

（２）＊next_pixel_buf

次の画像データ・バッファ・アドレスを示します。

next_pixel_buf は，指定ライン分を処理したあとの次の画像データ・バッファのカレント・アドレスを指定します。

指定ライン分を処理したあと，画像データ・バッファの最後の２ライン(3 ライン・モデル・テンプレート使用時)，または１ライン(2 ライン・モデル・テンプレート使用時)を next_pixel_buf 以前にコピーします。また，このとき pixel_buf はカレント・ラインを示します。

このメンバはワードでアラインした値を入力してください。

（３）pixel_buf_line

画像データ・バッファに設定してある画像データのライン数を示します。

必ず 1(1 ライン)以上の値を指定してください。画像バッファに必要なライン数は，Options の LRLTWO の値によって異なります。

LRLTWO=0：カレント・ライン前の２ライン(参照ライン)を含む３ライン分以上必要です。

LRLTWO=1：カレント・ライン前の１ライン(参照ライン)を含む２ライン分以上必要です。

指定ライン分を処理した場合は，設定した値を出力します。

pixel_buf_line の値は次の場合だけ残りのライン数を出力します。

・指定ライン分の処理が終了しないで，ストライプが終了またはページが終了したとき。

・圧縮符号データ・バッファ・フルによって中断したとき。

（４）＊code_buf

圧縮符号データ・バッファ・アドレスを示します。

code_buf は，ライブラリにより符号データが１バイト書き込まれる（読み出される）と＋１されます。

（５）code_buf_size

圧縮符号データ・バッファ残りサイズを示します。

code_buf_size は，ライブラリにより符号データが１バイト書き込まれる（読み出される）と－１されます。圧縮符号データ・バッファ残りサイズが０バイト(返り値が 0x01)の場合は，再設定してください。

（６）*Mps_St_tbl

Mps_St テーブルの先頭アドレスを示します。

Mps_St テーブルは，1K バイト必要です。

（７）Xd

水平方向画素数を示します。

ページ処理中には変更できません。JBIG フリー・パラメータと同じキー・ワードを使用しています。

圧縮処理時：８の倍数を指定してください。

伸長処理時：特に制限はありません。

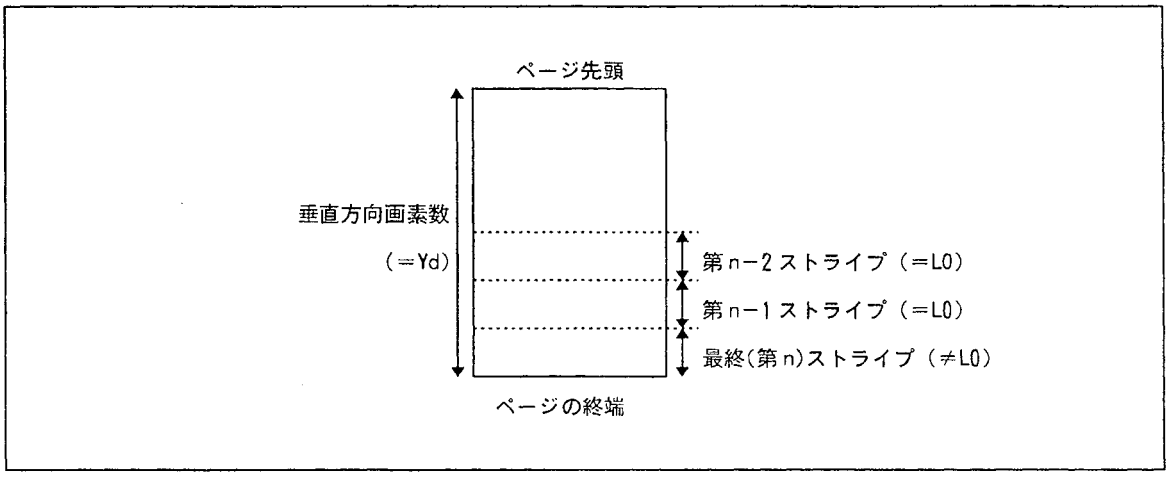
（８）Yd

垂直方向画素数を示します。

ページ処理中には変更できません。１ストライプ当たりのライン数(L0)との関係を次に示します。

Yd は，L0 の整数倍に一致するとは限りません。最終ストライプは L0 と一致しなくても，１ストライプとして処理します。JBIG フリー・パラメータと同じキー・ワードを使用しています。

図２－６　１ストライプ当たりのライン数(L0)との関係



（９）line_cnt

累積ライン数カウンタは，ライブラリで処理したライン数の累積を示します。

１ページまたは１ストライプのライン数が不明のときに参照してください。reset＝1によってクリアされます。

(10) L0

１ストライプ当たりのライン数を示します。

L0は，ページ処理中に変更できません。最終ストライプのライン数はL0と一致しなくても１ストライプとして処理します（**(8)Yd** 参照）。JBIGフリー・パラメータと同じキー・ワードを使用しています。

(11) Options

JBIG BIHのOptions Byteです。

Optionsはページ処理中に変更できません。JBIGフリー・パラメータと同じキー・ワードを使用しています。次にOptionsについて示します。

表２－３　Options

ﾋﾞｯﾄ位置	ﾋﾞｯﾄ名	説　　明
7	－	0固定
6	LRLTWO	モデル・テンプレートの指定 0：3ライン・モデル・テンプレートを使用 1：2ライン・モデル・テンプレートを使用
5	VLENGTH	NEWLENマーカの設定 0：設定なし 1：設定の可能性あり
4	－	0固定
3	TPBON	最低解像度レイヤTPの指定 0：使用しない 1：使用する
2-0	－	0固定

注意　2.1.1　標準仕様のOptionsのビット5で設定しているVLENGTHとNEWLENマーカの関係をチェックし，VLENGTH＝0でNEWLENマーカが付加（圧縮処理時）または検出（伸長処理時）された場合，NEWLEN エラー・ステータス(0x01)を返します。また，圧縮処理時は NEWLEN マーカ異常として関数の返り値(0x20)を返しますが，伸長処理時は NEWLEN マーカを無視して処理を続けます。

(12) reset

このライブラリを初期化します。

リセット(reset=1) でこのライブラリを実行すると，Mps_St テーブル，QM コーダ内部レジスタ，参照ライン(Options により 1 ライン/2 ライン)，restart_adr，line_cnt，newlen_err_sts，length_err はクリアされます。また，伸長処理時は newlen，sdrst，Lc，Tx，mrk_Tx，mrk_yAT，invalid_code_sts もクリアされ next_AT_line は 0xFFFFFFFF がセットされます。リセット後は，reset＝0 になります。ページの先頭で必ずリセットしてください。 ★

(13) abort

途中で処理を中断します。

・圧縮処理時

abort＝1 でこのライブラリを実行すると，アボート(強制終了)します。アボート(強制終了) は画像データ・バッファの先頭だけ受け付け，flush(圧縮コードの掃き出し)動作を行い，ABORT マーカを圧縮符号データ・バッファに出力して，圧縮処理を打ち切ります。アボート(強制終了) 後は，abort＝0 になります。

・伸長処理時

このライブラリの伸長処理では使用しません。

伸長処理の ABORT 処理について次に示します。

・ABORT マーカ・コードの検出時は，関数の返り値(0x10)として返します。

・ABORT マーカ・コードの検出時で伸長処理を打ち切ります。

(14) sdrst

1 ストライプ終了後の状態を設定(圧縮処理時)，または示します(伸長処理時)。sdrst＝1 は SDRST，sdrst＝0 は SDNORM です。

・圧縮処理時

1 ストライプ処理が終了したとき，次に示すようなマーカ・コードが圧縮符号データ・バッファに出力されます。

FFH，03H	FFH，02H
↑　↑	↑　↑
ESC　SDRST	ESC　SDNORM

1 ストライプ処理の終了後に参照され，1 (SDRST)の場合，JBIG 変数，Mps_St テーブルの内容がすべてクリアされ next_AT_line は 0xFFFFFFFF がセットされます。

・伸長処理時

SDRST または SDNORM のマーカ・コードが自動検出された場合，sdrst には 1 または 0 の値が設定されます。リセット(reset=1)によりクリアされます。

なお，ストライプ処理の終了時に sdrst が 1（SDRST）の場合，JBIG 変数および Mps_St テーブルの内容がすべてクリアされ next_AT_line は 0xFFFFFFFF がセットされます。

★

また，ストライプ処理の終了時に sdrst が 1（SDRST)の場合は，画像データ・バッファの参照ラインもクリアされます。参照ラインのクリアについては，画像データ・バッファの状態で異なるので注意してください。詳細については，**2.1.2（4）参照ラインの設定**を参照してください。

(15) newlen_err_sts

NEWLEN マーカ異常時に次のような NEWLEN エラー・ステータスを示します。NEWLEN マーカ正常時は 0 を返します。リセット(reset＝1)によりクリアされます。

0x01：圧縮処理時，Options：VLENGTH = 0 の状態で newlen ≠ 0 が設定された場合
　　伸長処理時，Options：VLENGTH = 0 の状態で NEWLEN マーカを検出した場合
0x02：圧縮処理時，newlen で指定された値が，累積ライン数カウンタより小さい，または垂直方向画素数より大きい場合
　　伸長処理時，NEWLEN マーカ検出時の length 値が，累積ライン数カウンタより小さい，または垂直方向画素数より大きい場合

なお，NEWLEN マーカ異常時はこのメンバのほか，length_err も同時に返します。

圧縮処理では，関数の戻り値(0x20)を返しますが，伸長処理ではステータスを設定するだけで，処理は続行します。また，圧縮，伸長処理ともに newlen_err_sts が返されるときは，length_err に length 値が設定されています。

(16) invalid_code_sts

このメンバは伸長処理時にだけ意味を持ちます。

関数の返り値として 0x100（マーカ・コードの異常）が返されるとき，次のようなエラー・ステータスを示します。リセット(reset＝1)によりクリアされます。

0x01：ストライプ終了時，END マーカがなく，ダミー・バイト（0x00）を検出
0x02：ストライプ終了時，END マーカがなく，ダミー・バイト以外を検出
0x03：ATMOVE，NEWLEN，COMMENT，RESERVE，SDRST/SDNORM，ABORT，スタッフ・バイト以外を検出

関数の返り値として 0x100 が返される場合，圧縮符号データ・バッファ・アドレスと圧縮符号データ・バッファ残りサイズは，異常コード検出直前の値を保持しています。invalid_code_sts が 0x01 の場合，ユーザ側で圧縮符号データ・バッファ・アドレスのインクリメントと圧縮符号データ・バッファ残りサイズのデクリメントを行い，再度伸長処理を呼び出すことでダミー・バイトを処理できます。invalid_code_sts は次に ESC コードが検出されたときにクリアされます。

ダミー・バイトが2バイト以上ある場合，次のようにしてダミー・バイトを処理できます。

・invalid_code_sts が 0 になるまで伸長処理の呼び出しを繰り返す
・圧縮符号データ・バッファ・アドレスを ESC コードのアドレスに進め，その分圧縮符号データ・バッファ残りサイズをデクリメントしてから，伸長処理を呼び出す

★　**(17) Tx, mrk_Tx, mrk_yAT, next_AT_line**

AT 処理には次に示す入出力パラメータを使用します。

●Tx, mrk_Tx, mrk_yAT

Tx, mrk_Tx は，ストライプごとに AT を水平方向に移動させる画素数を示します。
Tx は実際の AT 処理に使用され，mrk_Tx はマーカ・コード付加（検出）のみに使用されます。
mrk_yAT は，AT の移動開始ライン（ストライプからのライン数）を示しマーカ・コード付加（検出）のみに使用されます。
mrk_yAT＝0 の場合は，ストライプの先頭です。
AT の移動開始ラインは，それぞれのストライプに対して 0 からリスタートとするため mrk_yAT は，mrk_yAT＜LO となるように設定してください。
mrk_Tx, mrk_yAT の設定は，ストライプ先頭時に行ってください。それ以外では動作を保証しておりません。

●next_AT_line

next_AT_line は次の AT の移動開始ラインを示します（ページ先頭からの累積ラインで指定されます）。next_AT_line＝0 の場合はページの先頭です。
圧縮処理時に AT 処理を行わない場合は，ページ先頭時に next_AT_line に 0xFFFFFFFF をセットしてください。
なお，AT 移動開始ライン検出時には next_AT_line に 0xFFFFFFFF が設定されます。
reset，SDNORM／SDRST による Tx, mrk_Tx, mrk_yAT, next_AT_line の設定は次のとおりです。
また圧縮処理時に AT 処理を行わない場合は，ページ先頭時に Tx, mrk_Tx, mrk_yAT をゼロ・クリアしてください。

★　　表２－４　reset，SDNORM／SDRST による Tx，mrk_Tx，mrk_yAT，next_AT_line の設定

（ａ）圧縮処理のとき

	Tx	mrk_Tx	mrk_yAT	next_AT_line
reset=1				
SDNORM				
SDRST	0	0	0	0xFFFFFFFF

（ｂ）伸長処理のとき

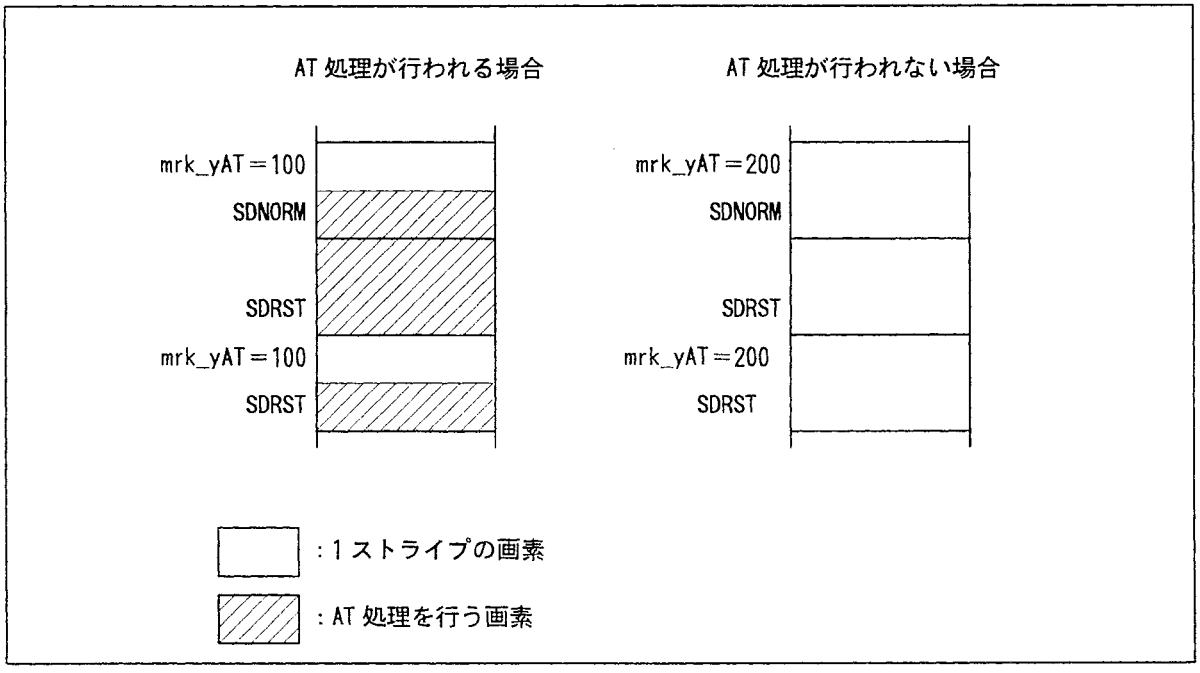
	Tx	mrk_Tx	mrk_yAT	next_AT_line
reset=1	0	0	0	0xFFFFFFFF
SDNORM				
SDRST	0	0	0	0xFFFFFFFF

備考　空欄にはユーザ設定値が入ります。

mrk_yAT と L0（１ストライプあたりのライン数）の関係は次に示すとおりです。

mrk_yAT＞L0 を設定した場合，AT 移動開始ラインは，それぞれのストライプに対して 0 から リスタートとするため，AT 処理に入りません。

図２－７　mrk_yAT と L0（１ストライプあたりのライン数）の関係



next_AT_line と line_cnt（累積ライン・カウンタ）の関係を次に示します。

next_AT_line＜line_cnt を設定した場合，AT 移動開始ラインを検出できないため，AT 処理に入れません。

AT 処理を行うときは必ず next_AT_line≧line_cnt となるように設定してください。

AT 処理の例として**図２－８　ATMOVE 有効範囲**，**図２－９　ユーザ側 AT 情報設定例**，**図２－１０　ATMOVE 圧縮処理例**，**図２－１１　ATMOVE 伸長処理例**を参照してください。

・圧縮処理時

ATMOVE マーカは１ストライプに１つだけ付加できます。
また，ストライプ先頭で次に示すようなマーカ・コードを圧縮符号データ・バッファに出力します。

```
FFH, 06H, XXH, XXH, XXH, XXH, XXH, 00H
 ↑    ↑          ↑          ↑    ↑
ESC  ATMOVE     mrk_yAT     mrk_Tx  fixed
```

AT がディフォールト(mrk_Tx＝0)の場合は，mrk_Tx＝0 を示すマーカ・コードは圧縮符号データ・バッファに出力しません。しかし，SDNORM で mrk_Tx≠0 の AT 処理から mrk_Tx＝0 に戻るとき，mrk_Tx＝0 を示すマーカ・コードは出力します。

圧縮処理時の AT 処理の手順は次のとおりです。

①ストライプ先頭時に，mrk_Tx には水平方向に移動させる画素数，mrk_yAT には AT の移動開始ライン（ストライプ先頭からのライン数），next_AT_line には次の AT 移動開始ライン（ページ先頭からの累積ライン）をセットし，圧縮処理を呼び出します。
ATMOVE マーカ・コードを圧縮符号データ・バッファへ出力します。

②next_AT_line で指定された次の AT 移動開始ラインを検出すると，関数の返り値（0x200）を返します。next_AT_line には，0xFFFFFFFF が設定されます。

③Tx に水平方向に移動させる画素数をセットし，圧縮処理を呼び出すことにより AT 処理を開始します。

・伸長処理時

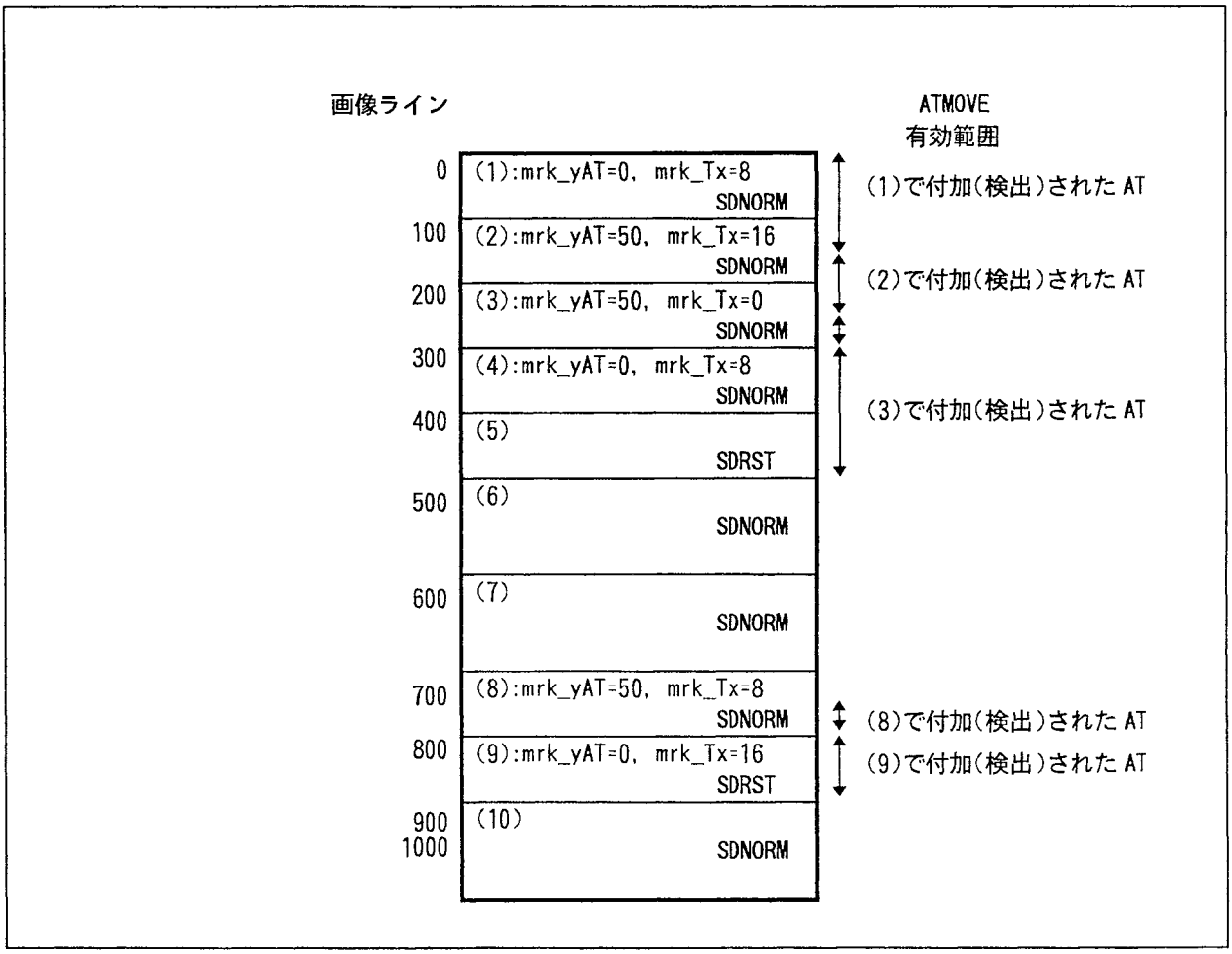
伸長処理時の AT 処理の手順は次のとおりです。

①ストライプ先頭で AT マーカ・コードが自動検出された場合，AT の移動開始ライン（ストライプからのライン数）を mrk_yAT に，AT 水平方向移動画素数を mrk_Tx に設定し，関数の返り値（0x400）を返します。
②次に AT 処理を開始するライン（ページ先頭からの累積ライン）を next_AT_line にセットし，伸長処理を呼び出します。
③ next_AT_line で指定された次の AT 移動開始ラインを検出すると，関数の返り値（0x200）を返します。next_AT_line には，0xFFFFFFFF が設定されます。
④ Tx に水平方向に移動させる画素数をセットし，伸長処理を呼び出すことにより AT 処理を開始します。

ATMOVE 有効範囲と AT 情報設定例，および ATMOVE 圧縮処理例／伸長処理例を次に示します。

★

図２－８　ATMOVE 有効範囲



★ **図2－9　ユーザ側 AT 情報設定例**

(1)のストライプ先頭で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	8	0	0

mrk_yAT, mrk_Tx, next_AT_line をセット

(1)のストライプ途中で AT 移動開始ライン検出後（0 ライン）

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	8	0xFFFFFFFF	8

Tx をセット

(2)のストライプ先頭で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	16	150	8

mrk_yAT, mrk_Tx, next_AT_line をセット

(2)のストライプ途中で AT 移動開始ライン検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	16	0xFFFFFFFF	16

Tx をセット

(3)のストライプ先頭で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	0	250	16

mrk_yAT, mrk_Tx, next_AT_line をセット

(3)のストライプ途中で AT 移動開始ライン検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	0	0xFFFFFFFF	0

Tx をセット

(4)のストライプ先頭時で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	8	300	0

mrk_yAT, mrk_Tx, next_AT_line をセット

(4)のストライプ途中で AT 移動開始ライン検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	8	0xFFFFFFFF	8

Tx をセット

(5)のストライプ先頭時

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	0	0xFFFFFFFF	8

(6), (7)のストライプ先頭時

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	0	0xFFFFFFFF	0

(8)のストライプ先頭で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	8	750	0

mrk_yAT, mrk_Tx, next_AT_line をセット

(8)のストライプ途中で AT 移動開始ライン検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
50	8	0xFFFFFFFF	8

Tx をセット

(9)のストライプ先頭で付加／検出後

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	16	800	8

mrk_yAT, mrk_Tx, next_AT_line をセット

(9)のストライプ途中で AT 移動開始ライン検出後

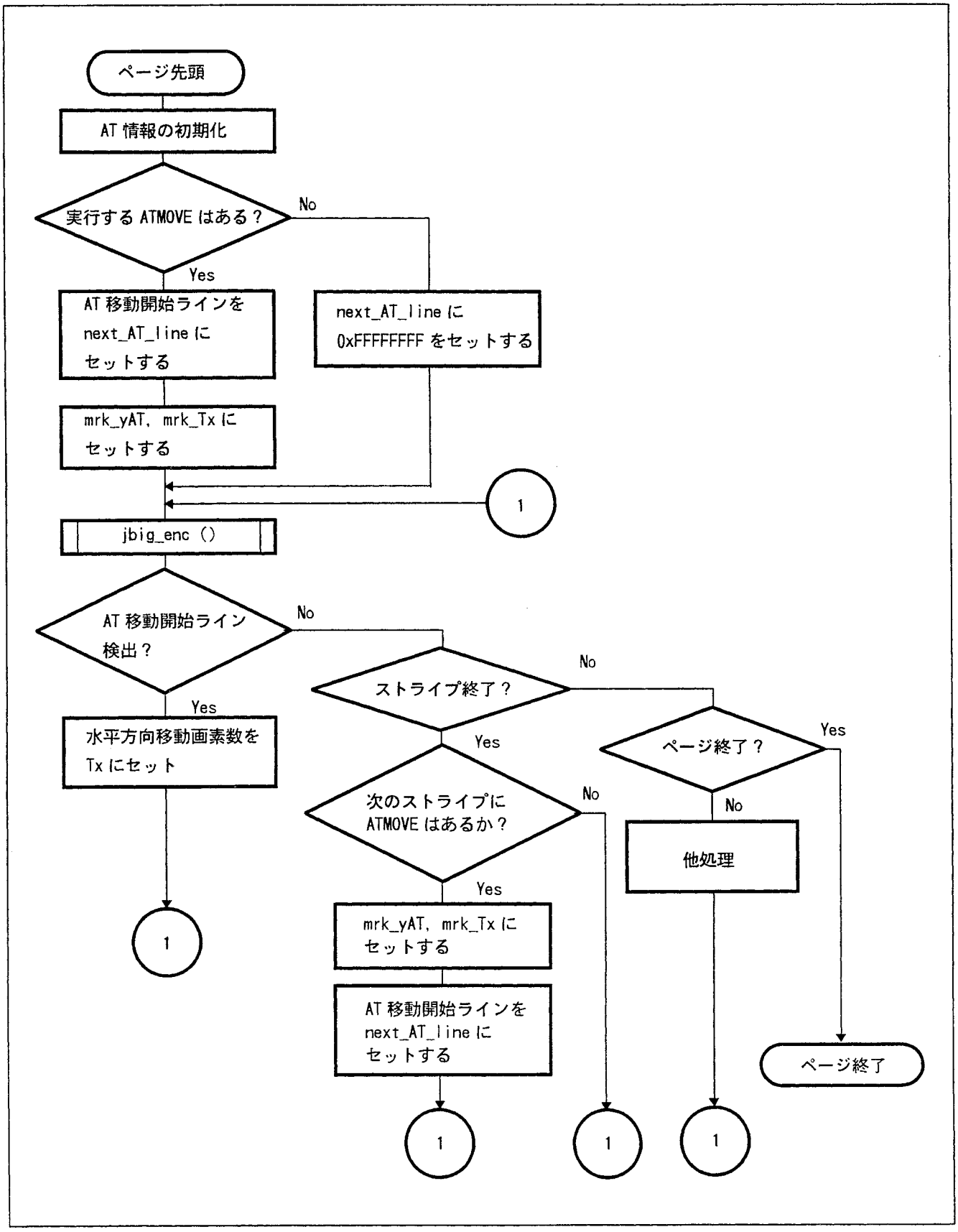
mrk_yAT	mrk_Tx	next_AT_line	Tx
0	16	0xFFFFFFFF	16

Tx をセット

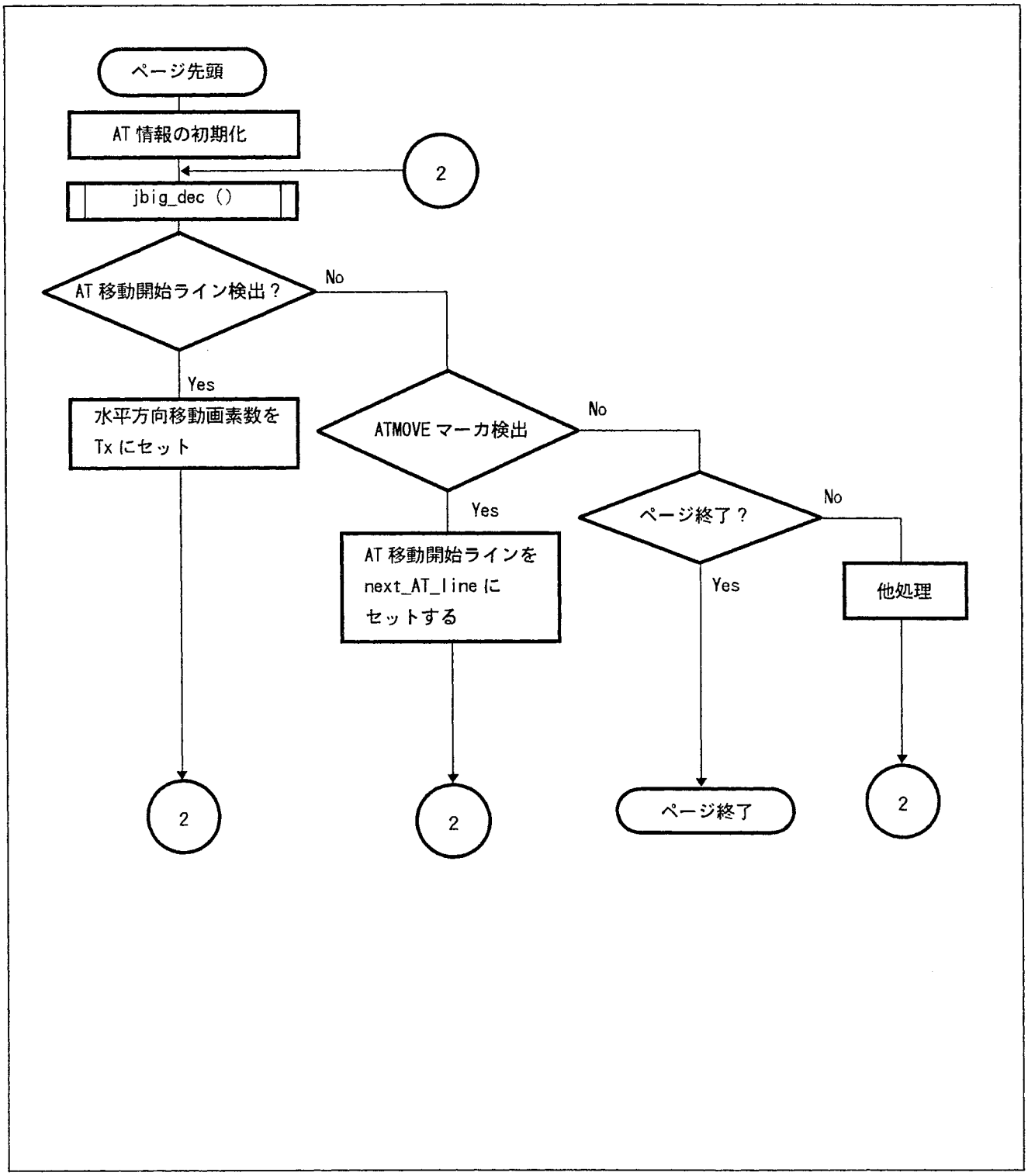
(10)のストライプ先頭時

mrk_yAT	mrk_Tx	next_AT_line	Tx
0	0	0xFFFFFFFF	0

★ 図２－10　ATMOVE 圧縮処理例



★ 図２－11　ATMOVE 伸長処理例



(18) newlen

ページ・ライン数の再設定を示します。

newlen の値が Yd で指定された値に変わって新しいページのライン数として処理します。

・圧縮処理時

newlen の値をページのライン数(length)として NEWLEN マーカを圧縮符号データ・バッファに出力します。NEWLEN マーカ出力後，newlen の値は 0 を出力します。

newlen への設定は，ストライプ先頭か画像バッファ先頭時に行ってください。符号バッファ先頭で設定した場合の動作は保証していません。

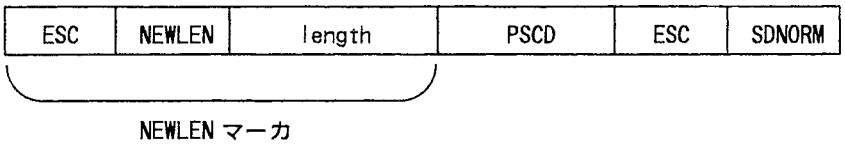
newlen ＝ 0：NEWLEN マーカを圧縮符号データ・バッファへ出力しません。このとき，ページ先頭で newlen をクリアしてください。リセット（reset=1）ではクリアされません。

newlen ≠ 0：VLENGTH = 1 で newlen の値が許容範囲内（Yd ≧ length 値 ≧ 累積ライン数カウンタ）であれば NEWLEN マーカを出力します。

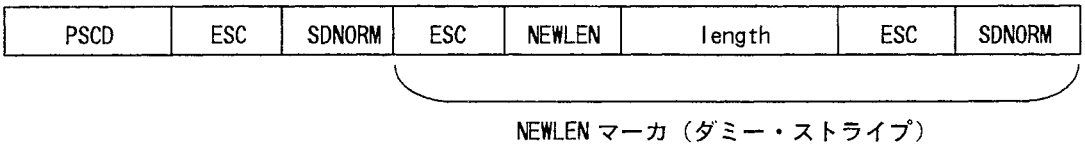
なお，最終ストライプ中の先頭以外で画像バッファ先頭のときに newlen を設定すると，NEWLEN マーカのあとに END マーカを出力します。

図２－12　newlen 設定時の出力データ

(a) ストライプ先頭のとき



(b) 最終ストライプ中の先頭以外で画像バッファ先頭のとき注



注　ページ終了となる場合のみ END マーカが付加されます（ダミー・ストライプとなります）。

次の場合，NEWLEN エラーとなります。このとき，NEWLEN マーカ異常として中断し，NEWLEN エラー・ステータス（newlen_err_sts），length_err を設定し，関数の返り値（0x20）を返します。

・Options：VLENGTH＝0 で，newlen≠0 を設定した場合
　newlen_err_sts には 0x01 が設定されます。
・length 値が許容範囲外の場合
（垂直方向画素数 ＜ length 値，または累積ライン数カウンタ ＞ length 値）
newlen_err_sts には 0x02，length_err に length 値を設定し，関数の返却値（0x20）を返します。

なお，NEWLEN エラーで中断したときは，newlen の値は 0 が出力されます。なお，NEWLEN エラー・ステータス（ newlen_err_sts ）と NEWLEN マーカ異常時の length 値（length_err）は，リセット（reset＝1）によりクリアされます。

・伸長処理時

NEWLEN マーカ・コードが自動検出された場合，新しいページのライン数として newlen に設定します。リセット（reset＝1）によりクリアされます。
ストライプ先頭時に NEWLEN マーカが検出された場合，新しいページのライン数として newlen に設定し，次にストライプの伸長処理を行います。ストライプ途中で END マーカ（SDNORM/SDRST）が検出された場合，次のように伸長処理を行います。

①仮想符号データ：0x00 として 1 ライン分の伸長処理を行ってから，処理を一時中断します。
② END マーカの次の符号が NEWLEN マーカかどうかを調べます。
③ NEWLEN マーカでない場合，仮想符号データ：0x00 を与えて伸長処理を再開します。
　この場合 L0 が終了条件となります。
　NEWLEN マーカの場合，VLENGTH をチェックします。
④ VLENGTH＝0 であれば，NEWLEN エラー・ステータス（ newlen_err_sts ）に 0x01，length_err に length 値が設定され，NEWLEN マーカと length 値は無視されます。すでに正常な NEWLEN マーカを検出していた場合はそのときの新しいページのライン数を，そうでない場合は初期値の Yd をページ終了判定とし，伸長処理を再開します。
⑤検出した length 値が許容範囲内（Yd ≧ length 値 ≧ 累積ライン数カウンタ）であれば，新しいページのライン数として newlen に設定し，このライン数を新しいページ終了条件として伸長処理を再開します。
　検出した length 値が許容範囲外であれば，NEWLEN エラー・ステータス（ newlen_err_sts ）に 0x02，length_err に length 値が設定され，NEWLEN マーカと length 値は無視されます。すでに正常な NEWLEN マーカを検出していた場合はそのときの新しいページのライン数を，そうでない場合は初期値の Yd をページ終了判定とし，伸長処理を再開します。

★　⑥ NEWLEN エラー・ステータス（newlen_err_sts）が，0x01 または 0x02 でページ終了した場合，圧縮符号データ・バッファ・アドレス（code_buf）は，length 値の直後に設定されます。

次の場合は NEWLEN エラーとなります。このとき，NEWLEN エラー・ステータス（newlen_err_sts）と length_err を設定し，伸長処理を続けます。

・Options：VLENGTH＝0 で，NEWLEN マーカを検出した場合
NEWLEN マーカは無視され，newlen_err_sts には 0x01，length 値を length_err に設定し伸長処理を続けます。
・length 値が許容範囲外の場合
（垂直方向画素数 ＜ length 値，または累積ライン数カウンタ ＞ length 値）
newlen_err_sts に 0x02，length 値を length_err に設定し伸長処理を続けます。

NEWLEN エラーのときは newlen の値は変化しません。なお，NEWLEN エラー・ステータス（newlen_err_sts）と NEWLEN マーカ異常時の length 値（length_err）は，リセット（reset＝1）によりクリアされます。

(19) length_err

NEWLEN マーカ異常時の length 値を示します。NEWLEN マーカ正常時は 0 を返します。
リセット（reset＝1）によりクリアされます。
圧縮処理時，関数の戻り値（0x20）を返します。伸長処理時は length 値を設定するだけで，処理を続けます。なお，NEWLEN マーカ異常時は，このメンバと一緒に newlen_err_sts も返されます。

(20) Lc

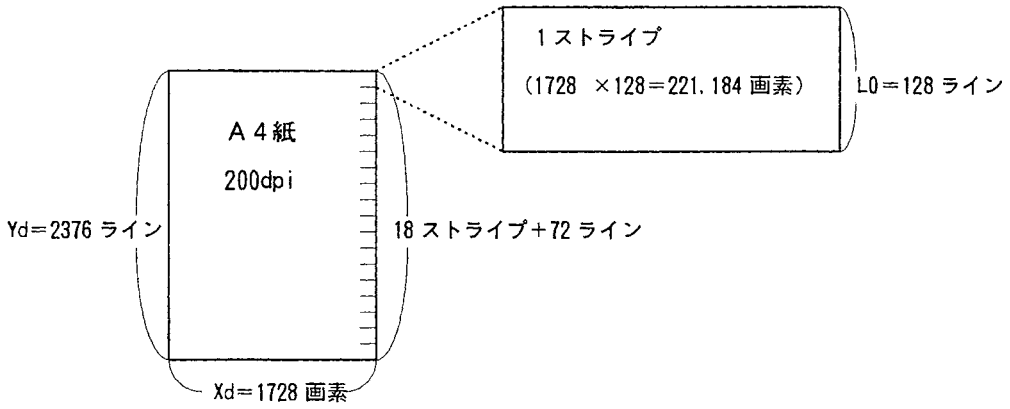
コメント・マーカのコメント長を示します。
・圧縮処理時
COMMENT マーカ出力はサポートしていません。

・伸長処理時
COMMENT マーカ・コードが自動検出された場合，コメント長を Lc に設定し，コメントの先頭アドレスを code_buf に設定します。関数の返り値(0x40)を返します。リセット（reset＝1）によりクリアされます。

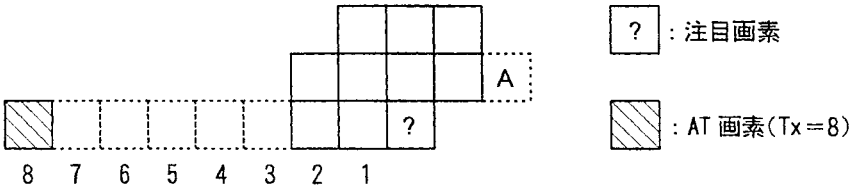
(21) restart_adr

指定ライン分の処理を終了しないで，このライブラリを再実行するために必要なフラグを示します。
restart_adr はページ先頭で，リセット（reset＝1）によってクリアされます。

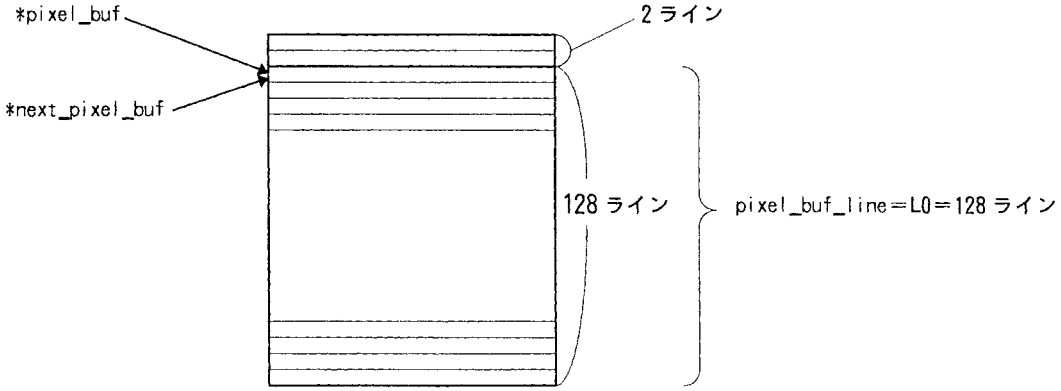
図２－13　パラメータの例



モデル・テンプレート（LRLTW0＝0 の場合）



画像データ・バッファ（LRLTW0＝0 の場合）



２．２．２　外部インタフェース

（１）圧縮処理系

（ａ）ＪＢＩＧ圧縮

・**分　　類**　圧縮処理系

・**関 数 名**　jbig_enc_m(), jbig_enc_l()

・**機能概要**　指定された画像データ・バッファおよびJBIGフリー・パラメータより圧縮処理を行い，圧縮符号データ・バッファへ符号データを出力します。

・**形　　式**　#include “jbig810.h”（V810ファミリ）
★　#include “jbig830.h”（V830ファミリ）
#include “jbig850.h”（V850ファミリ）
int jbig_enc_m(struct J_PARA *encdata)

・**引 き 数**　J_PARA

表２－５　圧縮処理系

メンバ名	型	入出力	説　　明
*pixel_buf	unsigned char	io	画像データ・バッファ・アドレス
*next_pixel_buf	unsigned char	i	次の画像データ・バッファ・アドレス
pixel_buf_line	unsigned int	io	画像データ・ライン数
*code_buf	unsigned char	io	圧縮符号データ・バッファ・アドレス
code_buf_size	unsigned int	io	圧縮符号データ・バッファ残りサイズ
*Mps_St_tbl	unsigned char	i	Mps_St テーブル先頭アドレス
Xd	unsigned int	i	水平方向画素数
Yd	unsigned int	i	垂直方向画素数
line_cnt	unsigned int	io	累積ライン数カウンタ
L0	unsigned int	i	1 ストライプ当たりのライン数
Options	unsigned char	i	JBIG BIH Options Byte
reset	unsigned char	io	1:リセット実行
abort	unsigned char	io	1:ABORT
sdrst	unsigned char	i	1 ストライプ処理後，1:SDRST/0:SDNORM
newlen_err_sts	unsigned char	o	NEWLEN エラー・ステータス
invalid_err_sts	unsigned char	－	異常マーカ・コード・ステータス
★ Tx	unsigned char	io	AT 水平方向移動画素数
★ mrk_Tx	unsigned char	io	AT 水平方向移動画素数（マーカ付加用）
★ mrk_yAT	unsigned int	io	AT 移動開始ライン，0:ストライプ先頭（マーカ付加用）
★ next_AT_line	unsigned int	io	次の AT 移動開始ライン，0:ページ先頭
newlen	unsigned int	io	ページのライン数の再定義，新しいライン数
length_err	unsigned int	o	NEWLEN マーカ異常時の length 値
Lc	unsigned int	－	コメント長
restart_adr	unsigned int	io	初期化／再開フラグ
reg_area[10]	unsigned int	io	レジスタ退避エリア
jbg_val[31]	unsigned int	io	JBIG 変数退避エリア（中断時専用）

備考　i：入力，o：出力，io：入出力，－：未使用

・返り値

表2－6に返り値の説明，表2－7に返り値の各ビットの説明を示します。

表2－6　返り値（圧縮処理系）

返り値	説　　明
0x200	AT移動開始ライン検出
0x20	NEWLENマーカ異常[注1]
0x10	アボート（強制終了）[注2]
0x0C	1ストライプ終了（画像データ・バッファ残りあり）
0x0A	ページ終了（画像データ・バッファ残りあり）
0x04	1ストライプ終了（画像データ・バッファ残りなし）
0x02	ページ終了（画像データ・バッファ残りなし）
0x01	圧縮符号データ・バッファ・フルによる中断
0x00	指定ライン分の圧縮正常終了

★

注1. エラーの詳細はnewlen_err_stsに格納されています。また，同時にnewlen=0，length_err=newlen設定値が出力されます。

2. 圧縮処理は再開できません。新規に圧縮を行う場合は，リセットしてページ先頭から始めてください。

注意　圧縮系は，0x01（圧縮符号バッファ・データ・フルによる中断）と，その他の返り値の状態が同時に発生する可能性がありますが，常に0x01が先に返ります。次に示す返り値については，同時に発生しません。
- 0X200（AT移動開始ライン検出）
- 0x20（NEWLENマーカ異常）

表２－７　返り値の各ビット（圧縮処理系）

ビット位置	説　　明
31-10	予約：0 固定
9	AT 移動開始ライン検出 0：未検出 1：検出
8-6	予約：0 固定
5	NEWLEN マーカ異常 0：正常 1：異常
4	アボート(強制終了)。 0：アボート(強制終了)しない 1：アボート(強制終了)する
3 注1	画像データ・バッファの状態を示します。 0：バッファ残りなし 1：バッファ残りあり
2	1 ストライプ終了。 0：未終了 1：終了
1	ページ終了。 0：未終了 1：終了
0	終了状態を示します。 0：符号化正常終了注2 1：中断注3

注 1. 1 ストライプ終了またはページ終了したときだけ，画像データ・バッファの残りがあるかないかが関係します。

2. 画像データの指定ライン終了，ストライプ終了，またはページ終了。

3. 圧縮符号データ・バッファ・フルによる中断。

・機　能

指定された画像データ・バッファおよび JBIG フリー・パラメータより，圧縮処理を行い，圧縮符号データ・バッファへ符号データを出力します。画像バッファ・ライン数で指定されたライン数分を圧縮します。

圧縮符号データ・バッファ残りサイズが 0 になったとき中断します。また，1 ストライプ分の処理を終了した場合，SDNORM/SDRST マーカを圧縮符号データ・バッファに出力し，1 ストライプ終了を示すステータスを返り値とします。ページ終了した場合もページ終了を示すステータスを返り値とします。

（b）画像データ

●ページが先頭のときは，参照ラインをすべて白に設定してください。

LRLTWO=0 の場合：前 2 ラインを参照ラインとしてすべて白にしてください。

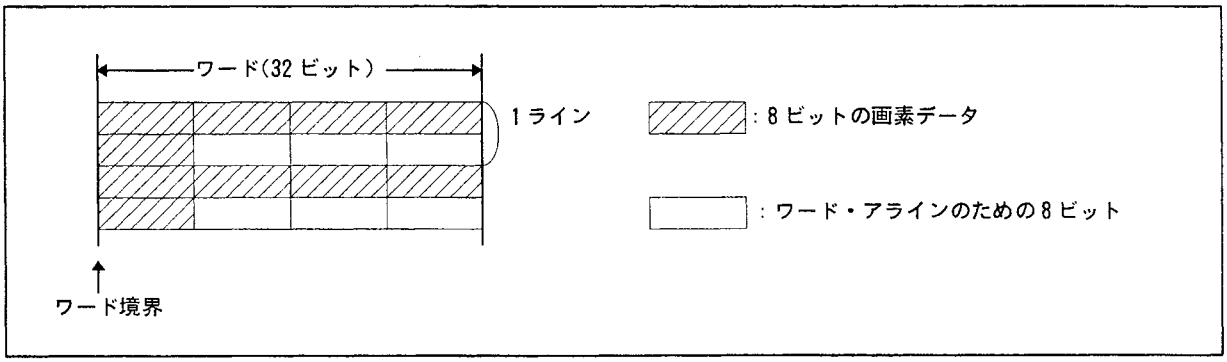
LRLTWO=1 の場合：前 1 ラインを参照ラインとしてすべて白にしてください。

参照ラインをすべて白に設定する場合は，reset に “1” を設定します。

●水平方向画素数は 8 の倍数にしてください。

●ライン先頭がワード境界となるように，画像データを置いてください。

図 2 －14　1 ラインが 40 画素の場合（圧縮処理系）



●ラインごとの画像データは，ワード境界で連続するように置いてください。

●画像データ・ライン数は，ストライプ終了時またはページ終了と圧縮符号データ・バッファ・フルにより中断したときに変化します。

●画像バッファ・ライン数の推奨値は，1 ストライプ当たりのライン数(LO)の定数倍または分数倍です。たとえば，1 ストライプ当たりのライン数を 9 ラインとすると，画像バッファ・ライン数の推奨値は定数倍の 9 ライン，18 ライン，27 ライン…，または分数倍の 1 ライン，3 ラインになります。

（２）伸長処理系

（ａ）ＪＢＩＧ伸長

・**分　　類**　伸長処理系

・**関 数 名**　jbig_dec_m(),　jbig_dec_l()

・**機能概要**　指定された受信バッファおよびJBIGフリー・パラメータより伸長処理を行い，
画像データ・バッファへ伸長した画像データを出力します。

・**形　　式**　#include “jbig810.h”（V810ファミリ）

★　#include “jbig830.h”（V830ファミリ）

#include “jbig850.h”（V850ファミリ）

```
int jbig_dec_m( struct J_PARA *decdata )
```

・**引 き 数**　J_PARA

表２－８　伸長処理系

メンバ名	型	入出力	説　　明
*pixel_buf	unsigned char	io	画像データ・バッファ・アドレス
*next_pixel_buf	unsigned char	i	次の画像データ・バッファ・アドレス
pixel_buf_line	unsigned int	io	画像データ・ライン数
*code_buf	unsigned char	io	圧縮符号データ・バッファ・アドレス
code_buf_size	unsigned int	io	圧縮符号データ・バッファ残りサイズ
*Mps_St_tbl	unsigned char	i	Mps_St テーブル先頭アドレス
Xd	unsigned int	i	水平方向画素数
Yd	unsigned int	i	垂直方向画素数
line_cnt	unsigned int	io	累積ライン数カウンタ
L0	unsigned int	i	1ストライプ当たりのライン数
Options	unsigned char	i	JBIG BIH Options Byte
reset	unsigned char	io	1:リセット実行
abort	unsigned char	－	未使用
sdrst	unsigned char	o	1ストライプ処理後，1:SDRST/0:SDNORM
newlen_err_sts	unsigned char	o	NEWLEN エラー・ステータス
invalid_err_sts	unsigned char	o	異常マーカ・コード・ステータス
★ Tx	unsigned char	io	AT 水平方向移動画素数
★ mrk_Tx	unsigned char	o	AT 水平方向移動画素数（マーカ検出用）
★ mrk_yAT	unsigned int	o	AT 移動開始ライン，0:ストライプ先頭（マーカ検出用）
★ next_AT_line	unsigned int	io	次の AT 移動開始ライン，0:ページ先頭
newlen	unsigned int	o	ページのライン数の再定義，新しいライン数
length_err	unsigned int	o	NEWLEN マーカ異常時の length 値
Lc	unsigned int	o	コメント長
restart_adr	unsigned int	io	初期化／再開フラグ
reg_area[10]	unsigned int	io	レジスタ退避エリア
jbg_val[31]	unsigned int	io	JBIG 変数退避エリア（中断時専用）

備考　i：入力，o：出力，io：入出力，－：未使用

・返り値

表２−９に返り値の説明，表２−10に返り値の各ビットの説明を示します。

表２－９　返り値（伸長処理系）

返り値	説　　明
★ 0x400	ATMOVE マーカ検出
★ 0x200	AT 移動開始ライン検出
0x100	マーカ・コードの異常[注1]
0x80	リザーブ・マーカ検出
0x40	コメント・マーカ検出
0x10	アボート（強制終了）[注2]
0x0C	１ストライプ終了（画像データ・バッファ残りあり）
0x0A	ページ終了（画像データ・バッファ残りあり）
0x04	１ストライプ終了（画像データ・バッファ残りなし）
0x02	ページ終了（画像データ・バッファ残りなし）
0x01	圧縮符号データ・バッファ・エンプティによる中断
0x00	指定ライン分の伸長正常終了

注 1. エラーの詳細は invalid_err_sts に格納されています。
invalid_err_sts＝１（ダミー・バイト検出）の場合はダミー・バイトを処理したあと，伸長処理を再開できます。その他の場合は伸長処理を再開できません。新規に伸長を行う場合は，リセットしてページ先頭から始めてください。

2. 伸長処理は再開できません。新規に伸長を行う場合は，リセットしてページ先頭から始めてください。

注意　伸長系は，0x01（圧縮符号バッファ・データ・エンプティによる中断）と，その他の返り値の状態が同時に発生する可能性がありますが，常に 0x01 が先に返ります。次に示す返り値については，同時に発生しません。
・0X200（AT 移動開始ライン検出）

表２－10　返り値の各ビット（伸長処理系）(1/2)

ビット位置	説　　明
31-11	予約：0 固定
10	ATMOVE マーカ検出 0：未検出 1：検出
9	AT 移動開始ライン検出 0：未検出 1：検出
8	マーカ・コード異常 0：正常 1：異常注
7	リザーブ・マーカ検出 0：未検出 1：検出

注　ATMOVE，NEWLEN，COMMENT，RESERVE，SDRST/SDNORM，ABORT，スタッフ・バイト以外の場合。または，ストライプが終了した次のデータが END マーカ(SDRST／SDNORM)でない場合。このとき，圧縮符号データ・バッファ・アドレス，圧縮符号データ残りバッファ・サイズは更新されず，検出直前の値を保持します。詳細は，**2.2.1(16) invalid_code_sts** を参照してください。

表２－10　返り値の各ビット（伸長処理系）(2/2)

ビット位置	説　　明
6	コメント・マーカ検出 0：未検出 1：検出
5	未使用
4	アボート（強制終了） 0：アボート（強制終了）しない 1：アボート（強制終了）する
3 注1	画像データ・バッファの状態を示します。 0：バッファ残りなし 1：バッファ残りあり
2	1ストライプ終了 0：未終了 1：終了
1	1ページ終了 0：未終了 1：終了
0	終了状態を示します。 0：伸長正常終了注2 1：中断注3

注 1. 1ストライプ終了またはページ終了したときだけ，画像データ・バッファの残りがあるかないかが関係します。

2. 画像データの指定ライン終了，ストライプの終了，またはページ終了。

3. 圧縮符号データ・バッファ・エンプティによる中断。

・機　能

指定された受信バッファおよびJBIGフリー・パラメータより伸長処理を行い，画像データ・バッファへ伸長した画像データを出力します。画像データ・ライン数で指定されたライン数分の伸長を行います。

圧縮符号データ・バッファ残りサイズが0になったとき中断します。また，1ストライプ分を処理した場合，SDNORM/SDRSTマーカを検出し，1ストライプ終了を示すステータスを返り値とします。ページ終了した場合もページ終了を示すステータスを返り値とします。

（b）画像データ

●ページが先頭のときは，参照ラインをすべて白に設定してください。

LRLTWO=0 の場合：前2ラインを参照ラインとしてすべて白にしてください。

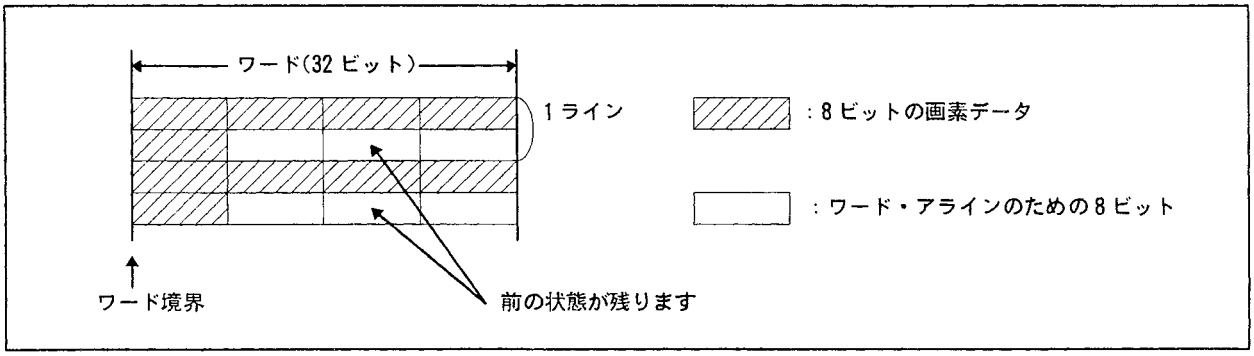
LRLTWO=1 の場合：前1ラインを参照ラインとしてすべて白にしてください。

すべての参照ラインを白に設定する場合は，reset に "1" を設定します。

●ライン先頭がワード境界となるように画像データを置いてください。

●ラインごとの画像データは，ワード境界で連続するように置いてください。次に示すワード・アラインのための8ビット部分は画素データが上書きされないで以前の状態が残るので注意してください。

図2－15　1ラインが40画素の場合（伸長処理系）



●画像データ・ライン数はストライプ終了時またはページ終了時/圧縮符号データ・バッファ・エンプティによる中断時に変化します。

●画像バッファ・ライン数の推奨値は，1ストライプ当たりのライン数(LO)の定数倍，または分数倍です。

たとえば，1ストライプ当たりのライン数を9ラインとすると，画像バッファ・ライン数の推奨値は，定数倍の9ライン，18ライン，27ライン…，または分数倍の1ライン，3ラインとなります。

画像バッファ・ライン数と1ストライプ当たりのライン数が，定数倍または分数倍でなく，かつSDRSTの場合（返り値：0x0C，sdrst＝1），次に伸長処理を行うときに参照ラインをクリアしてください。

（c）ABORT処理について

●ABORT マーカ・コード検出時は関数の返り値(0x10)として返します。

●ABORT マーカ・コード検出時点で伸長処理を中止します。

備考 1. RESERVE マーカ・コードが自動検出された場合，関数の返り値(0x80)として返します。また，RESERVE マーカ・コードの次のアドレスを code_buf に設定します。

2. スタッフ・バイトは自動廃棄されます。

〔メ　モ〕

保守／廃止

第３章　インストレーション

３．１　リンク手順

このライブラリ内で使用しているセクション名を次に示します。

表３－１　セクション

セクション名	属性	機　　能
.JBETEXT	text	JBIG 圧縮処理プログラム
.JBDTEXT	text	JBIG 伸長処理プログラム
.JBDATA	data	JBIG 圧縮／伸長テーブル

リンク手順を以下に示します。

（１）V810 ファミリ

(a) NEC版（ca732：Ver.2.00 以上）

ld732 -D 〈リンク・ディレクティブ〉〈.. オブジェクト・ファイル〉 .../libjbig.a
　　-o 〈出力ファイル〉

(b) GHS版（ELF版：Ver.1.8.7B 以上）

lx188 -o 〈出力ファイル〉 -sec 〈マップ・ファイル〉〈.. オブジェクト・ファイル〉 -L〈dir〉 -ljbig

★ **（２）V830 ファミリ**

(a) NEC版（ca830：Ver.1.00 以上）

ld830 -D 〈リンク・ディレクティブ〉〈.. オブジェクト・ファイル〉 .../libjbig.a
　　-o 〈出力ファイル〉

(b) GHS版（ELF版：Ver.1.8.8 以上）

lx -o 〈出力ファイル〉 -sec 〈マップ・ファイル〉〈.. オブジェクト・ファイル〉 -L〈dir〉 -ljbig

（３）V850 ファミリ

(a) NEC版（ca850：Ver.1.00 以上）

ld850 -D 〈リンク・ディレクティブ〉〈.. オブジェクト・ファイル〉 .../libjbig.a
　　-o 〈出力ファイル〉

(b) GHS版（ELF版：Ver.1.8.7B 以上）

lx188 -o 〈出力ファイル〉 -sec 〈マップ・ファイル〉〈.. オブジェクト・ファイル〉 -L〈dir〉 -ljbig

３．２　サンプルの作成手順

次に示すNEC版，GHS版のmakefileを実行します。

３．２．１　V810ファミリ

(1) NEC版（ca732：Ver.2.00以上）

・makefile

```
CC = ca732
AS = as732
LD = ld732

all: enc_main.elf dec_main.elf
enc_main.elf: crt810.o enc_main.o jbig_enc.lnk
    $(LD) -D jbig_enc.lnk crt810.o enc_main.o libjbig.a -o enc_main.elf
dec_main.elf: crt810.o dec_main.o jbig_dec.lnk
    $(LD) -D jbig_dec.lnk crt810.o dec_main.o libjbig.a -o dec_main.elf

enc_main.o: enc_main.c
    $(CC) -Wa,-cn -cpu 742 -c enc_main.c
dec_main.o: dec_main.c
    $(CC) -Wa,-cn -cpu 742 -c dec_main.c
crt810.o: crt810.s
    $(AS) crt810.s -o crt810.o
```

(a) NEC製リンカのオプション

-o 〈ファイル名〉

生成される実行ファイル名を指定します。

-D 〈リンク・ディレクティブ〉

セクション(.text, .data,...)の先頭アドレスを設定します。

以下に jbig_enc.lnk の内容を示します。

```
DATA : !LOAD ?RW V0x0 {.data = $PROGBITS ?AW;  JBDATA = $PROGBITS ?AW;};
TEXT : !LOAD ?RX V0x10000 {.text = $PROGBITS ?AX;  JBETEXT = $PROGBITS ?AX;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT;
```

以下に jbig_dec.lnk の内容を示します。

```
DATA : !LOAD ?RW V0x0 {.data = $PROGBITS ?AW;  JBDATA = $PROGBITS ?AW;};
TEXT : !LOAD ?RX V0x10000 {.text = $PROGBITS ?AX;  JBDTEXT = $PROGBITS ?AX;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT;
```

(b) サンプル・メイン・ソースのコンパイル

例　ca732 -c enc_main.c

　　└──────コンパイルだけ

詳細はNECのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

（2）GHS版（ELF版　Ver.1.8.7B以上）

・makefile

```
CC = cc810e
AS = as800
LD = lx188

all:enc_main.elf dec_main.elf
enc_main.elf: crt810.o enc_main.o
    $(LD) -o enc_main.elf -e __start -sec { .text 0x10000 : .JBETEXT :
       .data 0x0 : .JBDATA : .sdata : .sbss : .bss } crt810.o enc_main.o -L../../lib810
       -ljbig -M

dec_main.elf: crt810.o dec_main.o
    $(LD) -o dec_main.elf -e __start -sec { .text 0x10000 : .JBDTEXT :
       .data 0x0 : .JBDATA : .sdata : .sbss : .bss } crt810.o dec_main.o -L../../lib810
       -ljbig -M

enc_main.o: enc_main.c
    $(CC) -c -OA -G enc_main.c
dec_main.o: dec_main.c
    $(CC) -c -OA -G dec_main.c
crt810.o: crt810.s
    $(AS) -elf -cpu=V810 -o crt810.o crt810.s
```

（a）GHS製リンカのオプション

-o〈ファイル名〉

生成される実行ファイル名を指定します。

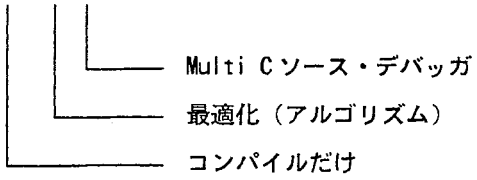
-sec { セクション　アドレス [: セクション　アドレス...] }

セクション(.text, .data, ...)の先頭アドレスを設定します。各セクションの指定は ":" で区切ります。アドレスを省略した場合，直前に指定したセクションに連続します。

（ｂ）サンプル・メイン・ソースのコンパイル

最適化オプション -OA を指定し，コンパイルを行っています。

例　cc810e -c -OA -G enc_main.c



詳細はGHSのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

★　３．２．２　V830ファミリ

（1）NEC版（ca830：Ver.1.00以上）

・makefile

```
CC = ca830
AS = as830
LD = ld830

all: enc_main.elf dec_main.elf
enc_main.elf: crt830.o enc_main.o jbig_enc.lnk
    $(LD) -D jbig_enc.lnk crt830.o enc_main.o libjbig.a -o enc_main.elf
dec_main.elf: crt830.o dec_main.o jbig_dec.lnk
    $(LD) -D jbig_dec.lnk crt830.o dec_main.o libjbig.a -o dec_main.elf

enc_main.o: enc_main.c
    $(CC) -cn -cpu 5100 -c enc_main.c
dec_main.o: dec_main.c
    $(CC) -cn -cpu 5100 -c dec_main.c
crt830.o: crt830.s
    $(AS) -cn -cpu 5100 crt830.s -o crt830.o
```

(a) NEC製リンカのオプション

-o 〈ファイル名〉

生成される実行ファイル名を指定します。

-D 〈リンク・ディレクティブ〉

セクション(.text, .data,...)の先頭アドレスを設定します。

以下に jbig_enc.lnk の内容を示します。

```
TEXT : !LOAD ?RX V0x10000000 {.text = $PROGBITS ?AX;  JBETEXT = $PROGBITS ?AX;};
DATA : !LOAD ?RW V0x10010000 {.data = $PROGBITS ?AW;  JBDATA = $PROGBITS ?AW;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT;
```

以下に jbig_dec.lnk の内容を示します。

```
TEXT : !LOAD ?RX V0x0x10000000 {.text = $PROGBITS ?AX;  JBDTEXT = $PROGBITS ?AX;};
DATA : !LOAD ?RW V0x10010000 {.data = $PROGBITS ?AW;  JBDATA = $PROGBITS ?AW;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT;
```

(b) サンプル・メイン・ソースのコンパイル

例　ca830 -cpu 5100 -c enc_main.c

└──── コンパイルだけ

詳細はNECのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

（２）GHS版（ELF版　Ver.1.8.8以上）

・makefile

```
CC = cc830
AS = as800
LD = lx

all:enc_main.elf dec_main.elf
enc_main.elf:crt.830.o enc_main.o
    $(LD) -o enc_main.elf -e __start -sec { .text 0x10000000 : .JBETEXT :
       .data 0x10010000 : .JBDATA : .sdata : .bss : .sbss } crt830.o enc_main.o
       -L../../lib830 -ljbig -M

dec_main.elf: crt830.o dec_main.o
    $(LD) -o dec_main.elf -e __start -sec { .text 0x10000000 : .JBDTEXT :
       :.data 0x10010000 : .JBDATA : .sdata : .sbss : .bss } crt830.o dec_main.o
       -L../../lib830 -ljbig -M

enc_main.o: enc_main.c
    $(CC) -c -OA -G enc_main.c
dec_main.o: dec_main.c
    $(CC) -c -OA -G dec_main.c
crt830.o: crt830.s
    $(AS) -elf -cpu=V830 -o crt830.o crt830.s
```

（ａ）GHS製リンカのオプション

-o〈ファイル名〉

生成される実行ファイル名を指定します。

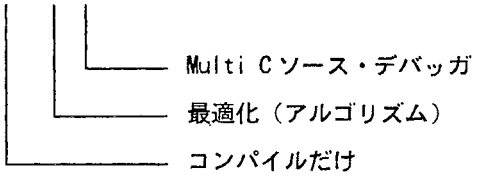
-sec { セクション　アドレス [: セクション　アドレス...] }

セクション(.text, .data, ...)の先頭アドレスを設定します。各セクションの指定は ":" で区切ります。アドレスを省略した場合，直前に指定したセクションに連続します。

（b）サンプル・メイン・ソースのコンパイル

最適化オプション -OA を指定し，コンパイルを行っています。

例　cc830 -c -OA -G enc_main.c



詳細はGHSのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

3．2．3　V850ファミリ

（1）ＮＥＣ版（ca850：Ver.1.00以上）

・makefile

```
CC = ca850
AS = as850
LD = ld850

all: enc_main.elf dec_main.elf
enc_main.elf: crt850.o enc_main.o jbig_enc.lnk
    $(LD) -D jbig_enc.lnk crt850.o enc_main.o libjbig.a -o enc_main.elf
dec_main.elf: crt850.o dec_main.o jbig_dec.lnk
    $(LD) -D jbig_dec.lnk crt850.o dec_main.o libjbig.a -o dec_main.elf

enc_main.o: enc_main.c
    $(CC) -cpu 3000 -c enc_main.c
dec_main.o: dec_main.c
    $(CC) -cpu 3000 -c dec_main.c
crt850.o: crt850.s
    $(AS) -cn -cpu 3000 crt850.s -o crt850.o
```

（a）NEC製リンカのオプション

-o〈ファイル名〉

生成される実行ファイル名を指定します。

-D〈リンク・ディレクティブ〉

セクション（.text, .data, ...）の先頭アドレスを設定します。

以下に jbig_enc.lnk の内容を示します。

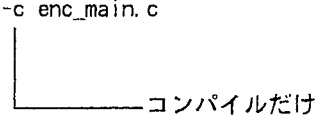
```
DATA1: !LOAD ?RW V0x1000 {.JBDATA = $PROGBIT ?AW;};
TEXT: !LOAD ?RX V0x2000 {.text = $PROGBITS ?AX; JBETEXT = $PROGBITS ?AX;};
DATA2: !LOAD ?RW V0x100000 {.data = $PROGBITS ?AW;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL & __tp_TEXT;
__ep_DATA @ %EP_SYMBOL;
```

以下に jbig_dec.lnk の内容を示します。

```
DATA1: !LOAD ?RW V0x1000 {.JBDATA = $PROGBIT ?AW;};
TEXT: !LOAD ?RX V0x2000 {.text = $PROGBITS ?AX; JBDTEXT = $PROGBITS ?AX;};
DATA2: !LOAD ?RW V0x100000 {.data = $PROGBITS ?AW;};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL & __tp_TEXT;
__ep_DATA @ %EP_SYMBOL;
```

（b）サンプル・メイン・ソースのコンパイル

例　ca850 -c enc_main.c



詳細はNECのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

（2）GHS版（ELF版　Ver.1.8.7B以上）

・makefile

```
CC = cc850e
AS = as850e
LD = lx188

all: enc_main.elf dec_main.elf
enc_main.elf: crt850.o enc_main.o
    $(LD) -o enc_main.elf -e __start -sec {.text 0x2000: .JBETEXT: .JBDATA 0x1000:
          .data 0x100000: .sdata: .bss: .sbss} crt850.o enc_main.o -L../../lib850 -ljbig -M

dec_main.elf: crt850.o dec_main.o
    $(LD) -o dec_main.elf -e __start -sec {.text 0x2000: .JBDTEXT: .JBDATA 0x1000:
          .data 0x100000: .sdata: .sbss: .bss} crt850.o dec_main.o -L../../lib850 -ljbig -M

enc_main.o: enc_main.c
    $(CC) -c -OA -G enc_main.c
dec_main.o: dec_main.c
    $(CC) -c -OA -G dec_main.c
crt850.o: crt850.s
    $(AS) -elf -o crt850.o crt850.s
```

（a）GHS製リンカのオプション

-o〈ファイル名〉

生成される実行ファイル名を指定します。

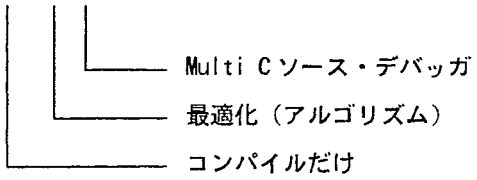
-sec { セクション　アドレス [: セクション　アドレス...] }

セクション(.text, .data, ...)の先頭アドレスを設定します。各セクションの指定は“:”で区切ります。アドレスを省略した場合，直前に指定したセクションに連続します。

（ｂ）サンプル・メイン・ソースのコンパイル

最適化オプション -OA を指定し，コンパイルを行っています。

例　cc850e -c -OA -G enc_main.c



詳細はGHSのリンカ，コンパイラのユーザーズ・マニュアルを参照してください。

３．３　シンボル名規約

このライブラリ内で使用するシンボル名は，すべて先頭に“ jbig_”が付加されています。“ jbig_”は“ミドルウエア識別名”です。関数名の場合は“ミドルウエア識別名＋関数機能名”になります。ユーザ・アプリケーション内で使用するシンボル名には注意してください。

〔メ　モ〕

第４章　システム例

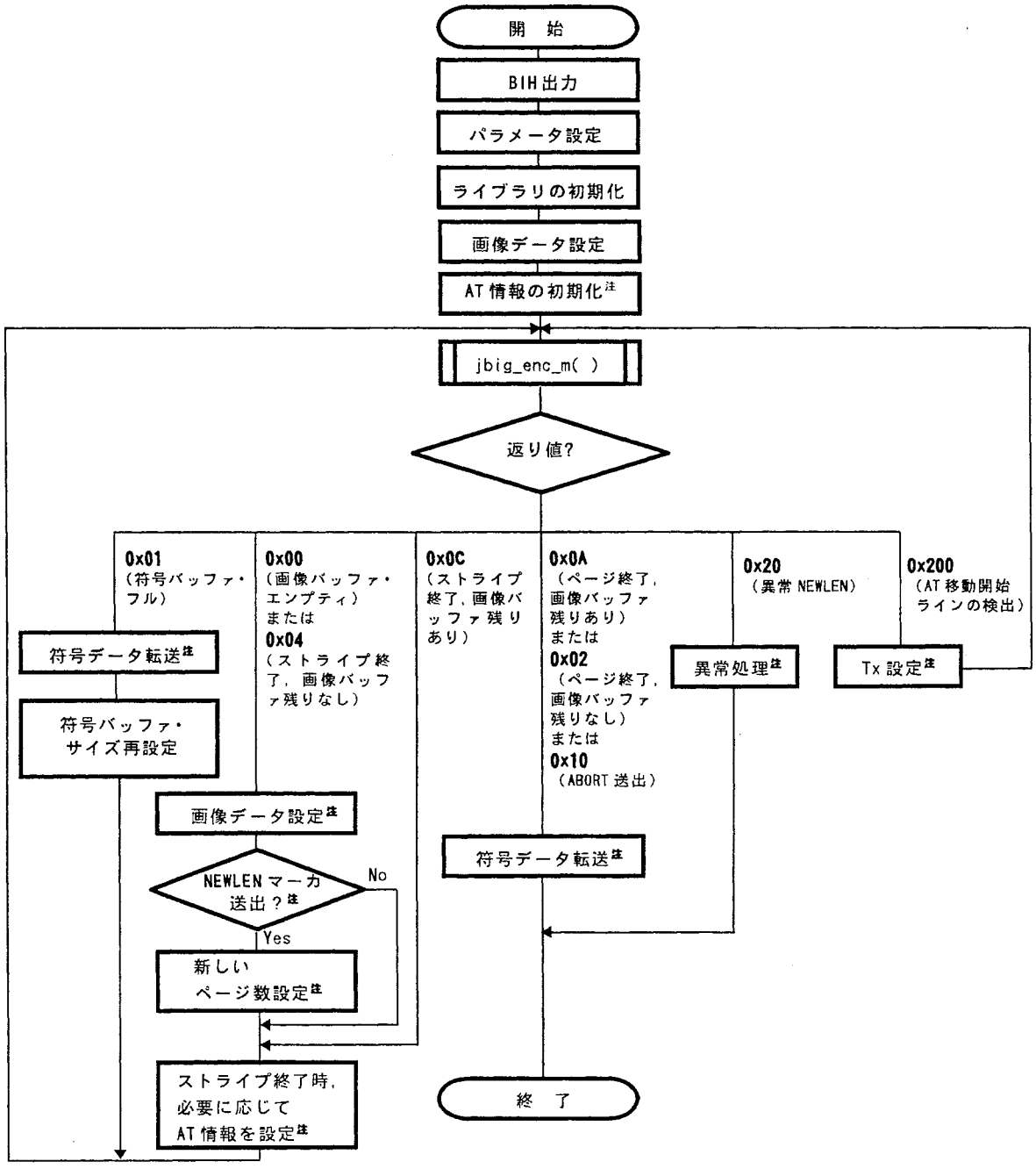
この章では，JBIG による圧縮／伸長処理のシステム例を示します。また，システム例のメイン・ソースについては**付録Ａ　サンプル・ソース・リスト**を参照してください（このサンプルは仕様のすべてを満たすものではありません）。

４．１　圧縮系システム例

指定された画像データ・バッファおよび JBIG フリー・パラメータより，圧縮処理を行い，圧縮符号データ・バッファに符号データを出力します。画像バッファ・ライン数で指定されたライン数分の圧縮を行います。

圧縮符号データ・バッファの残りサイズが 0 になったときに中断します。また，１ストライプ分の処理が終了した場合，SDNORM/SDRST マーカを圧縮符号データ・バッファに出力して，１ストライプの終了を示すステータスを返り値とします。

★ 図４－１　圧縮系システム例

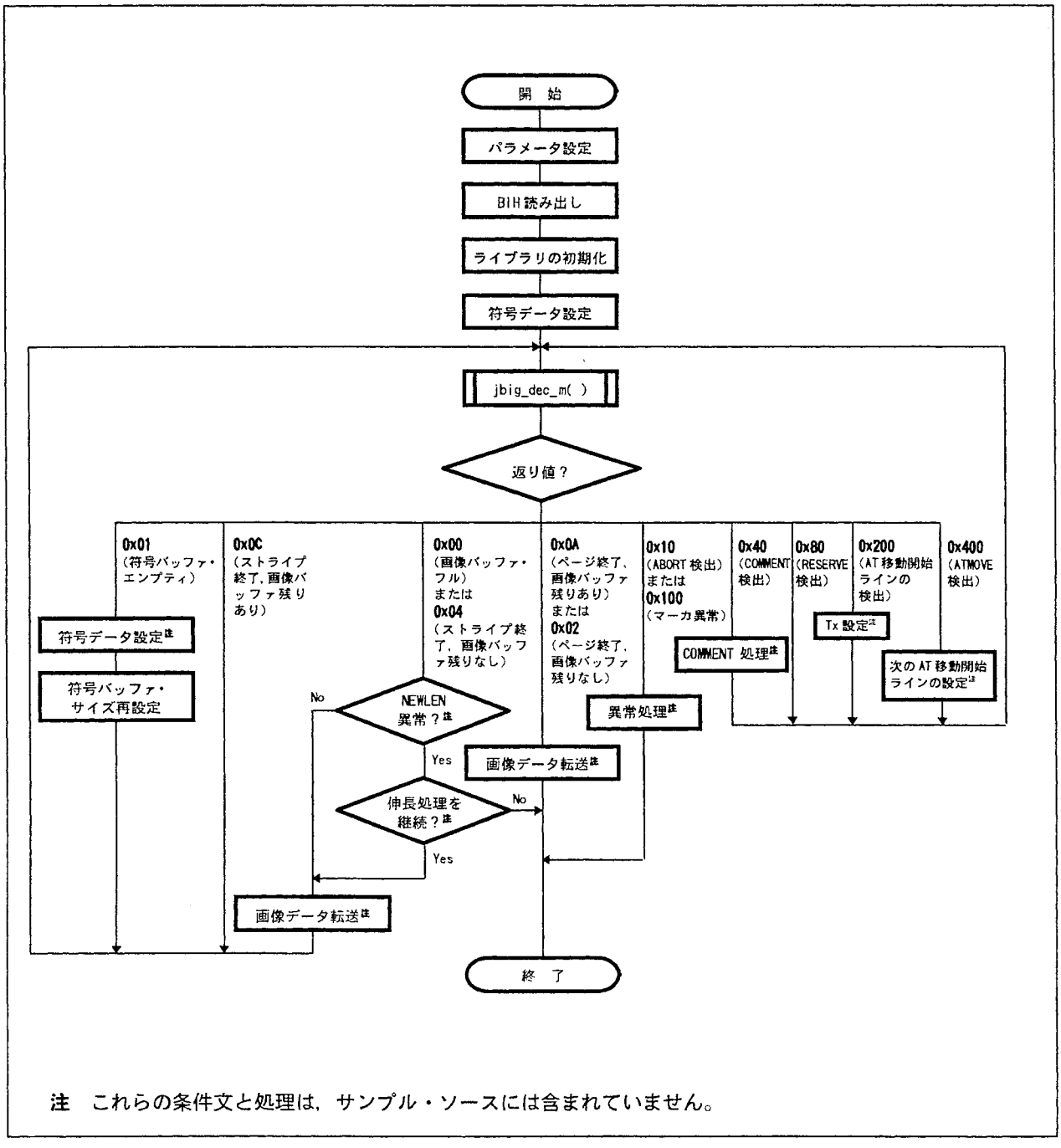


注　これらの条件文と処理は，サンプル・ソースには含まれていません。

４．２　伸長系システム例

指定された受信バッファおよびJBIGフリー・パラメータより伸長処理を行い，画像データ・バッファに伸長した画像データを出力します。画像データ・ライン数で指定されたライン数分の伸長を行います。圧縮符号データ・バッファの残りサイズが０になったときに中断します。また，１ストライプ分の処理が終了した場合は，SDNORM/SDRSTマーカを検出して，１ストライプ終了を示すステータスを返り値とします。

★　**図４－２　伸長系システム例**

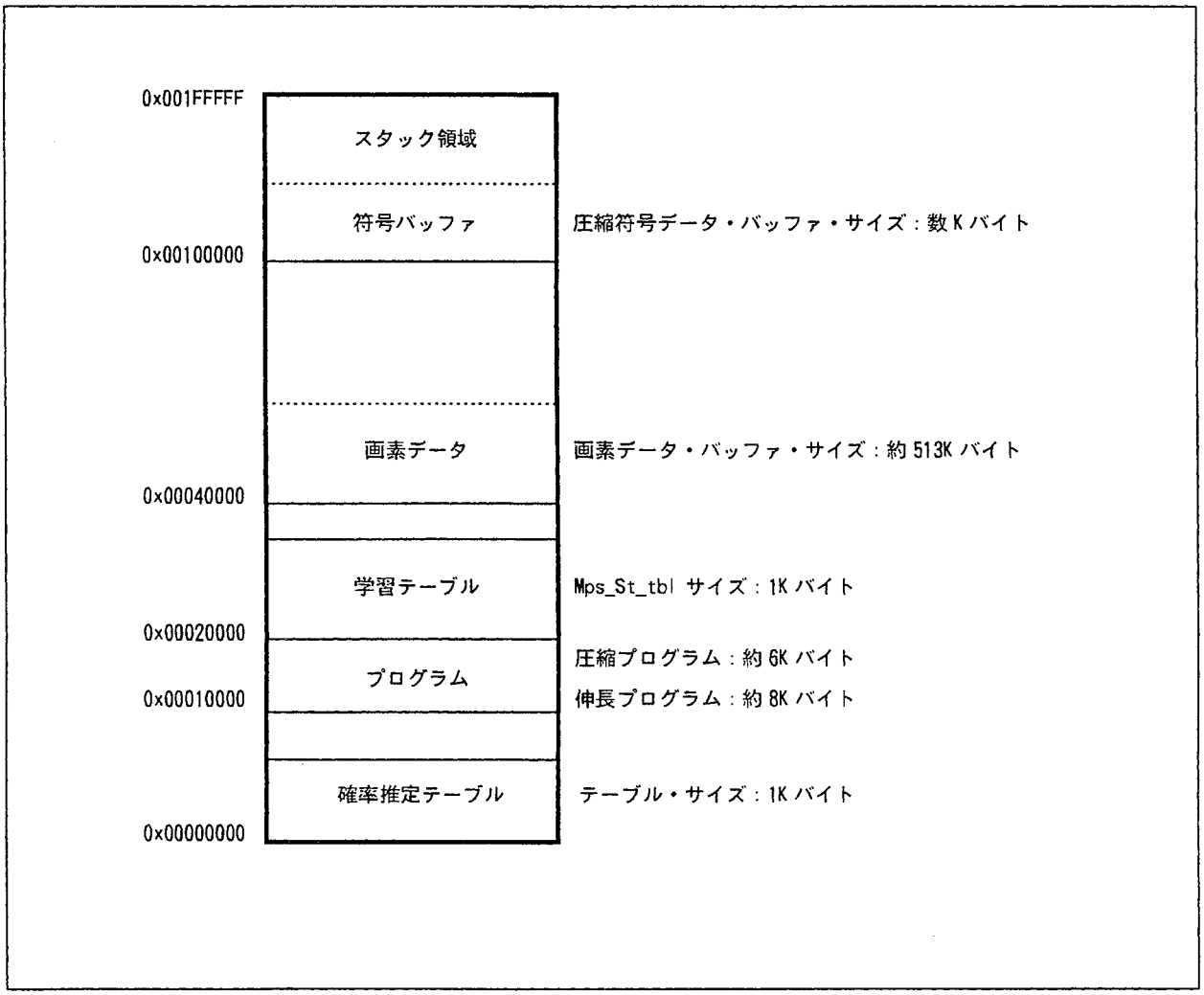


注　これらの条件文と処理は，サンプル・ソースには含まれていません。

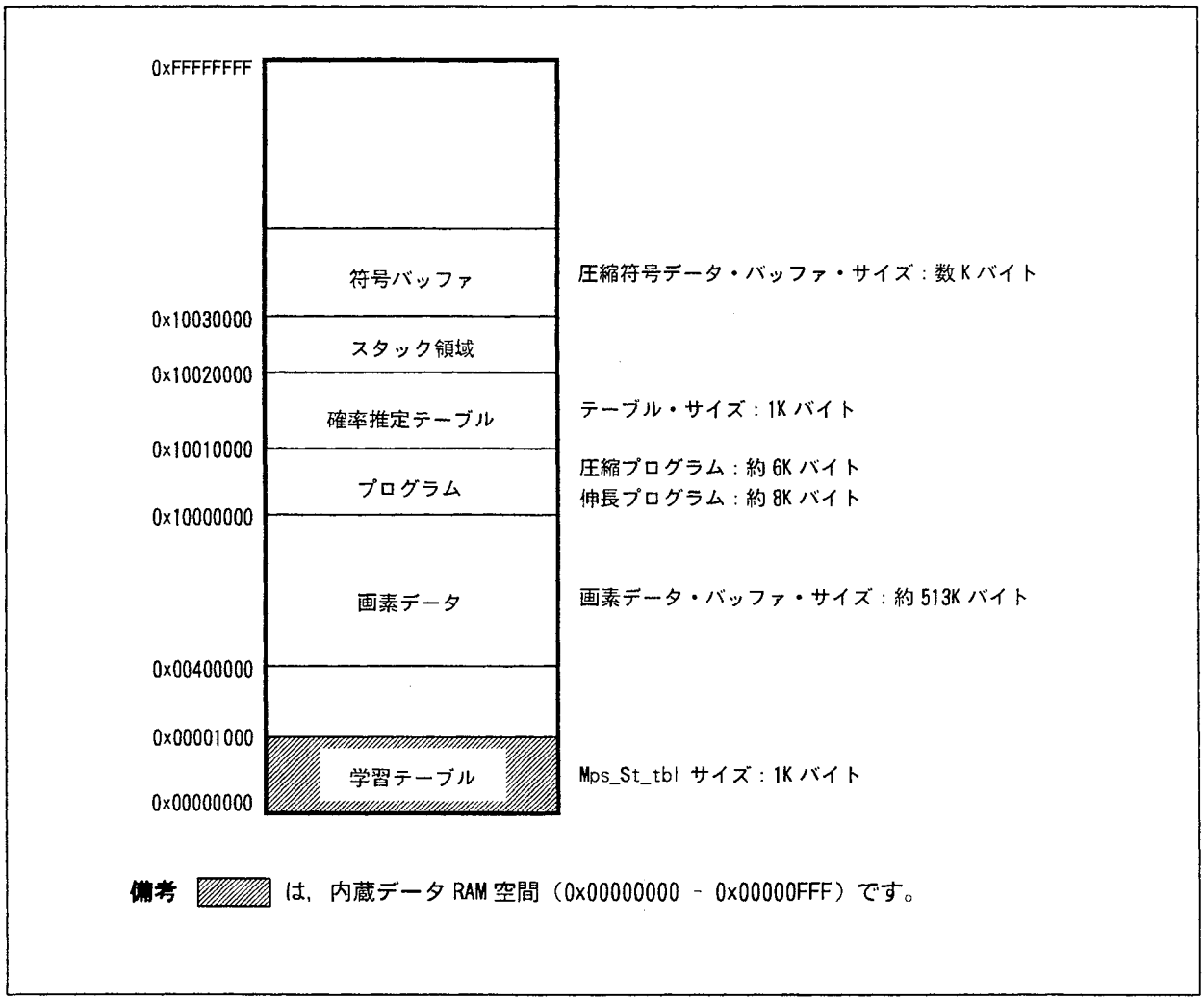
４．３　メモリ・マップ例

次にサンプル・プログラム（圧縮／伸長）のメモリ・マップ例を示します。

図４－３　V810 ファミリ・メモリ・マップ例

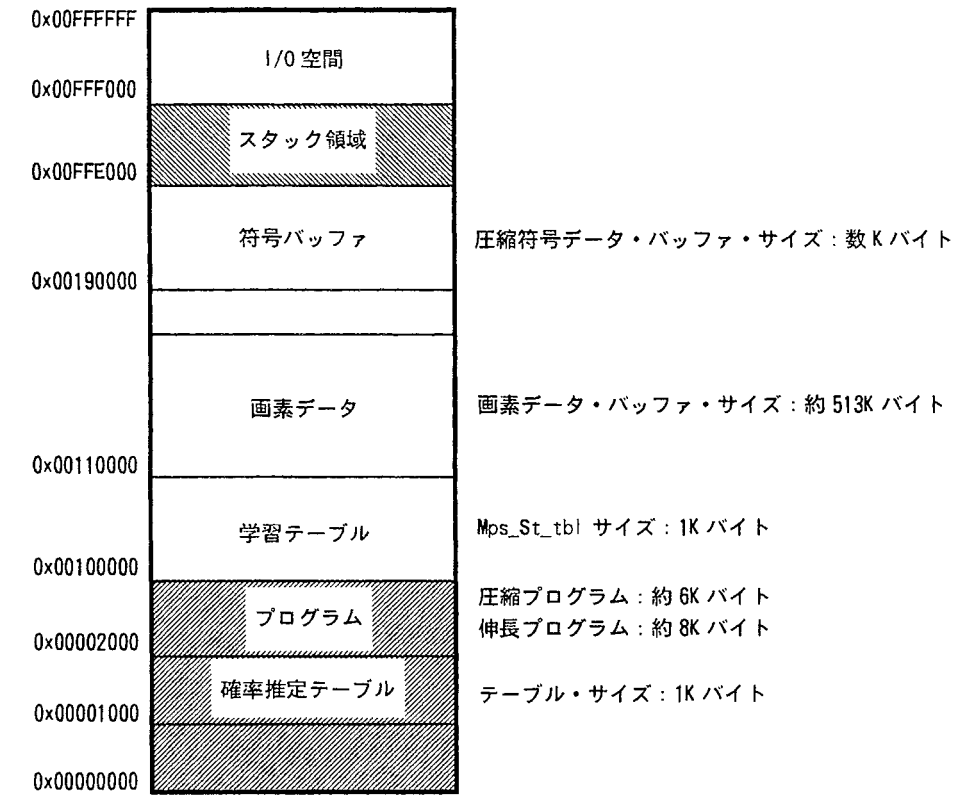


★ **図４－４　V830 ファミリ・メモリ・マップ例（V830 の場合）**



備考 ▨ は，内蔵データ RAM 空間（0x00000000 － 0x00000FFF）です。

図４－５　V850 ファミリ・メモリ・マップ例（V851 の場合）



備考　　は，内蔵 ROM 空間（0x00000000 - 0x000FFFFF）です。

　　は，内蔵 RAM 空間（0x00FFE000 - 0x00FFEFFF）です。

付録A　サンプル・ソース・リスト

A．1　JBIG圧縮サンプル・ソース

```
#include "jbig810.h"

/* BIH */
#define BIH_DL      0
#define BIH_D       0
#define BIH_P       1
#define BIH_FILL    0
#define BIH_XD      1728
#define BIH_YD      2376
#define BIH_L0      128
#define BIH_MX      0x7F
#define BIH_MY      0
#define BIH_ORDER   0
#define VLENGTH     0x20
#define TPBON       0x08  /* TPBON */
#define LRLTWO      0x40  /* 3 LINE */

/* jbig_enc_m() */
#define PIXEL_BUF         0x40000L
#define PIXEL_BUF_LINE    72
#define CODE_BUF          0x100000L
#define CODE_BUF_SIZE     256
#define MPS_ST_BUF        0x20000L
#define PAGE1             2376
#define STRIPE1           128
#define XD                1728
#define NORMAL_END        0x00
#define CODE_FULL         0x01
#define PAGE_END_FULL     0x02
#define PAGE_END          0x0A
#define STRIPE_END_FULL   0x04
#define STRIPE_END        0x0C
#define ABORT_END         0x10
#define NEWLEN_ERR        0x20

main()
{
            struct J_PARA encdata;
            register unsigned int cnt;
            register unsigned int pixel_byte;
            register unsigned int status;

            encdata.code_buf = (unsigned char *)CODE_BUF;

            /* BIH output */
            *encdata.code_buf++ = (unsigned char)BIH_DL;
            *encdata.code_buf++ = (unsigned char)BIH_D;
            *encdata.code_buf++ = (unsigned char)BIH_P;
```

```
*encdata.code_buf++ = (unsigned char)BIH_FILL;

*encdata.code_buf++ = (unsigned char)(BIH_XD>>24);
*encdata.code_buf++ = (unsigned char)(BIH_XD>>16);
*encdata.code_buf++ = (unsigned char)(BIH_XD>>8);
*encdata.code_buf++ = (unsigned char)BIH_XD;

*encdata.code_buf++ = (unsigned char)(BIH_YD>>24);
*encdata.code_buf++ = (unsigned char)(BIH_YD>>16);
*encdata.code_buf++ = (unsigned char)(BIH_YD>>8);
*encdata.code_buf++ = (unsigned char)BIH_YD;

*encdata.code_buf++ = (unsigned char)(BIH_L0>>24);
*encdata.code_buf++ = (unsigned char)(BIH_L0>>16);
*encdata.code_buf++ = (unsigned char)(BIH_L0>>8);
*encdata.code_buf++ = (unsigned char)BIH_L0;

*encdata.code_buf++ = (unsigned char)BIH_MX;
*encdata.code_buf++ = (unsigned char)BIH_MY;
*encdata.code_buf++ = (unsigned char)BIH_ORDER;
*encdata.code_buf++ = (unsigned char)(TPBON|VLENGTH);

/* jbig_enc_m() parameter set */
encdata.pixel_buf = (unsigned char *)PIXEL_BUF;
encdata.pixel_buf_line = PIXEL_BUF_LINE;
encdata.code_buf_size = CODE_BUF_SIZE;
encdata.Mps_St_tbl = (unsigned char *)MPS_ST_BUF;
encdata.Xd = XD;
encdata.Yd = PAGE1;
encdata.line_cnt = 0;
encdata.L0 = STRIPE1;
encdata.Options = (TPBON|VLENGTH);          /* TP=on, 3tmp */
encdata.newlen = 0;
encdata.sdrst = 0;                          /* SDNORM      */
encdata.abort = 0;                          /* ABORT       */
encdata.Tx = 0;                             /* AT = default */
encdata.yAT = 0;
encdata.restart_adr = 0;
encdata.reset = 1;                          /* reset       */

pixel_byte = PIXEL_BUF_LINE*(encdata.Xd/8);
encdata.next_pixel_buf = (unsigned char *)(PIXEL_BUF + pixel_byte);

/******************/
/* pixel data set */
/******************/
for(;;){
      switch(status = jbig_enc_m(&encdata)){
              case CODE_FULL          :encdata.code_buf_size = CODE_BUF_SIZE;
                                       break;
              case STRIPE_END         :break;
              case PAGE_END           :goto page_end;
              case ABORT_END          :if( encdata.code_buf_size != CODE_BUF_SIZE )
```

```
                                        /**********************/
                                        /* code data forward */
                                        /**********************/
                                        abort_int();
                        case STRIPE_END_FULL    :
                        case NORMAL_END         :/*******************/
                                                 /* pixel data set */
                                                 /*******************/
                                        encdata.next_pixel_buf = (unsigned char *)
                                        (encdata.pixel_buf + pixel_byte);
                                        break;
                        case PAGE_END_FULL      :goto page_end;
                        case NEWLEN_ERR         :newlen_err();
            }
        }
page_end:;

}

abort_int(){
            exit(1);
}

newlen_err(){
            exit(1);
}
```

A．2　JBIG伸長サンプル・ソース

```
#include "jbig810.h"

/* jbig_dec_m() */
#define PIXEL_BUF       0x40000L
#define PIXEL_BUF_LINE  72
#define CODE_BUF        0x100000L
#define CODE_BUF_SIZE   0x10000
#define MPS_ST_BUF      0x20000L
#define PAGE1           2376

#define     NORMAL_END      0x00
#define     CODE_BUF_FULL   0x01
#define     PAGE_END_FULL   0x02
#define     PAGE_END        0x0a
#define     STRIPE_END_FULL 0x04
#define     STRIPE_END      0x0c
#define     ABORT_END       0x10
#define     COMMENT_END     0x40
#define     RESERVE_END     0x80
#define     MARKER_ERR      0x100

main()
{
      struct J_PARA decdata;
      register unsigned int status, pixel_byte;
      register unsigned int pixel_buf_next = PIXEL_BUF;

      decdata.pixel_buf = (unsigned char *)PIXEL_BUF;
      decdata.pixel_buf_line = PIXEL_BUF_LINE;
      decdata.code_buf = (unsigned char *)CODE_BUF;
      decdata.code_buf_size = CODE_BUF_SIZE - 20;        /* BIH 20 byte          */
      decdata.pixel_buf = (unsigned char *)PIXEL_BUF;
      decdata.restart_adr = 0;
      decdata.Mps_St_tbl = (unsigned char *)MPS_ST_BUF;

      decdata.line_cnt= 0;
      decdata.code_buf += 4;

      decdata.Xd  = (*decdata.code_buf++) << 24;         /* read BIH (Xd)        */
      decdata.Xd |= ((*decdata.code_buf++) << 16);
      decdata.Xd |= ((*decdata.code_buf++) <<8);
      decdata.Xd |= *decdata.code_buf++;

      decdata.Yd  = (*decdata.code_buf++) << 24;         /* read BIH (Yd)        */
      decdata.Yd |= ((*decdata.code_buf++) << 16);
      decdata.Yd |= ((*decdata.code_buf++) <<8);
      decdata.Yd |= *decdata.code_buf++;

      decdata.L0  = (*decdata.code_buf++) << 24;         /* read BIH (L0)        */
      decdata.L0 |= ((*decdata.code_buf++) << 16);
      decdata.L0 |= ((*decdata.code_buf++) <<8);
      decdata.L0 |= *decdata.code_buf++;
```

```
        decdata.code_buf += 3;

        decdata.Options = (*decdata.code_buf++) & 0xff;      /* read BIH (Options)    */
        decdata.reset = 1;                                   /* reset */

        pixel_byte = PIXEL_BUF_LINE * (decdata.Xd / 8);
        decdata.next_pixel_buf = (unsigned char *)(PIXEL_BUF + pixel_byte);

        for(;;){
              switch(status = jbig_dec_m(&decdata)){
                    case CODE_BUF_FULL       :decdata.code_buf_size = CODE_BUF_SIZE;break;
                    case STRIPE_END          :break;
                    case PAGE_END            :goto page_end;
                    case STRIPE_END_FULL     :
                    case NORMAL_END          :/**********************/
                                              /* pixel data forward */
                                              /**********************/
                                             decdata.next_pixel_buf = (unsigned char *)
                                             (decdata.pixel_buf + pixel_byte) ;
                                             break;
                    case PAGE_END_FULL       :goto page_end;
                    case ABORT_END           :abort_err();
                    case COMMENT_END         :comment_end();
                    case RESERVE_END         :break;
                    case MARKER_ERR          :marker_err();
              }
        }

page_end:;

}

abort_err(){
        exit(1);
}

comment_end(){
/*******
```

保守／廃止

〔メ　モ〕

保守／廃止

★

付録B　総合索引

B．1　５０音で始まる語句の索引

【か行】

【さ行】

【た行】

【は行】

【ま行】

【ら行】

Ｂ．２　アルファベットで始まる語句の索引

【Ａ】

【Ｍ】

【Ｔ】

保守／廃止

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01 東京都港区芝五丁目７番１号（NEC本社ビル）	東京	(03)3454-1111 (大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460 名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540 大阪市中央区城見一丁目４番24号（NEC関西ビル）	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社 札幌	(011)251-5599	太田支店 太田	(0276)46-4011	福井支店 福井	(0776)22-1866
東北支社 仙台	(022)267-8740	宇都宮支店 宇都宮	(028)621-2281	富山支店 富山	(0764)31-8461
岩手支店 盛岡	(019)651-4344	小山支店 小山	(0285)24-5011	三重支店 津	(0592)25-7341
郡山支店 郡山	(0249)23-5511	長野支社 松本	(0263)35-1662	京都支社 京都	(075)344-7824
いわき支店 いわき	(0246)21-5511	甲府支店 甲府	(0552)24-4141	神戸支社 神戸	(078)333-3854
長岡支店 長岡	(0258)36-2155	埼玉支社 大宮	(048)649-1415	中国支社 広島	(082)242-5504
土浦支店 土浦	(0298)23-6161	立川支社 立川	(0425)26-5981	鳥取支店 鳥取	(0857)27-5311
水戸支店 水戸	(029)226-1717	千葉支社 千葉	(043)238-8116	岡山支店 岡山	(086)225-4455
神奈川支社 横浜	(045)682-4524	静岡支社 静岡	(054)254-4794	松山支店 松山	(089)945-4149
群馬支店 高崎	(0273)26-1255	北陸支社 金沢	(076)232-7303	九州支社 福岡	(092)261-2806

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 応用ソフトウエア技術部	〒210 川崎市幸区塚越三丁目484番地	川崎	(044)548-8860	半導体 インフォメーションセンター FAX(044)548-7900 (FAXにてお願い致します)
半導体販売技術本部 東日本販売技術部	〒108-01 東京都港区芝五丁目７番１号（NEC本社ビル）	東京	(03)3798-9619	
半導体販売技術本部 中部販売技術部	〒460 名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋	(052)222-2125	
半導体販売技術本部 西日本販売技術部	〒540 大阪市中央区城見一丁目４番24号（NEC関西ビル）	大阪	(06) 945-3383	

保守／廃止

アンケート記入のお願い

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

［ドキュメント名］μSAP703000-B02，μSAP705100-B02，μSAP70732-B02 ユーザーズ・マニュアル
（U11057JJ3V1UM00（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（　　）					
（　　）					

キリトリ

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

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

４．ご意見，ご要望

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

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

NEC半導体インフォメーションセンター
FAX：（044）548-7900

保守／廃止